

UC Santa Cruz

UC Santa Cruz Electronic Theses and Dissertations

Title

Improving Clinically Relevant Classification of Gene Expression Datasets Using Attribute Classifiers as Features

Permalink

<https://escholarship.org/uc/item/78p2r99m>

Author

Durbin, Kenneth James

Publication Date

2018

Peer reviewed|Thesis/dissertation

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

**IMPROVING CLINICALLY RELEVANT CLASSIFICATION OF
GENE EXPRESSION DATASETS USING ATTRIBUTE
CLASSIFIERS AS FEATURES**

A dissertation submitted in partial satisfaction of the
requirements for the degree of

DOCTOR OF PHILOSOPHY

in

BIOINFORMATICS

by

K. James Durbin

September 2018

The Dissertation of K. James Durbin
is approved:

Professor Josh Stuart, Chair

Professor David Haussler

Professor Scot Lokey

Lori Kletzer
Vice Provost and Dean of Graduate Studies

Copyright © by
K. James Durbin
2018

Table of Contents

List of Figures	vi
List of Tables	x
Abstract	xi
Dedication	xv
1 Introduction	1
1.1 The Need for Prognostic Prediction	1
1.2 Gene signatures	3
1.2.1 70-gene Signature Example	4
1.2.2 Performance Limitations	7
1.2.3 Signature Stability and Interpretability	9
1.3 Representation Based Improvements to Prognostic Classification .	10
1.3.1 Importance of Representation in Classifier Construction . .	12
1.3.2 Pathway Activity Inference using Condition Responsive Gene Sets	15
1.3.3 Inferred Pathway Activities	20
1.3.4 Meta-gene Attractors	23
1.4 Semantic Attributes	27
1.4.1 Semantic Attributes in Machine Vision	29
1.4.2 Biological Semantic Attributes	31
1.4.3 Novel Contribution of Semantic Attributes	34
2 Aim 1: Tools	37
2.1 WekaMine: large scale model selection, training, and application .	38
2.1.1 Phases of Pipeline	40
2.1.2 Model Selection	40
2.1.3 Model Training and Application	47
2.1.4 Experiment Design and Reproducibility	48
2.2 Viewtab: A Big Data Spreadsheet	49

2.3	Csvsql: Treating CSV Files as Databases	56
2.4	Grapnel: Java library	60
3	Aim 2: Semantic Attribute Results	63
3.0.1	Building Models for UMLS Terms in GEO	63
3.0.2	Pooling From UMLS Term Hierarchies	64
3.0.3	Basic Test Building UMLS Based Classifier	68
3.0.4	UMLS Compentium Conclusion	68
4	Survival Prediction	69
4.1	Survival Prediction With Semantic Attributes	69
4.1.1	Mutation Attribute Classifiers	69
4.1.2	Clinical Attribute Classifiers	73
4.1.3	MEMo Event Attribute Classifiers	73
4.1.4	All models	74
4.1.5	Semantic Transform Results	74
4.1.6	Informative Semantic Attributes	76
5	Drug Sensitivity Prediction	79
5.1	Drug Sensitivity Prediction With Semantic Attributes	79
5.1.1	Modeling Tissue Type with URSA	80
5.1.2	Modeling Chromatin State with Gene Expression Classifiers	82
5.1.3	Modeling Gene Essentiality with Gene Expression Classifiers	92
5.1.4	Drug Sensitivity With Gene Level Features	96
5.1.5	Applying Semantic Attributes to Drug Sensitivity	100
5.1.6	Essentiality And Drug Targets	121
5.1.7	Chromatin Features And Drug Targets	126
5.1.8	Mutation Attributes vs Mutations	139
6	Aim 3: Sample Psychic	146
6.1	Design	147
6.2	User Interface	150
6.3	CIRM Single Cell Report Card	161
7	Conclusion	168
A	Normalization	173
A.0.1	Exponential Normalization	174
A.0.2	Cross-tissue Normalization Test	175
A.0.3	Stem Cell Cross-Platform + Cross-Tissue Cross Classifica- tion Test	177
A.0.4	Semantic Attribute Classification with Exponential Normal- ization	179

B	WekaMine Components and Usage	181
B.1	WekaMine Scripts	181
B.2	WekaMine Experiment DSL	181
B.2.1	Experiments	181
B.2.2	Configuration File	182
B.2.3	Example	182
B.2.4	Special Keywords	183
B.2.5	More Realistic Example	183
C	Semantic Classifier Performance	186
D	Drug Targets And Activities	190

List of Figures

1.1	Conceptual Stages in Prognostic Classification	5
1.2	Supervised Classifier Construction for 70-gene Prognostic Signature	6
1.3	Most Random Signatures Significantly Associated With Breast Cancer Outcome	11
1.4	City Block Toy Example	13
1.5	City Block Toy Example Showing Power of Representation	14
1.6	Condition Responsive Gene Sets	17
1.7	PAC Classifier Performance	18
1.8	CORG Pathway Activities for Two Lung Cancer Datasets	19
1.9	Conversion of a genetic pathway diagram into a PARADIGM model	22
1.10	Idealized Subnetwork vs Full Network	23
1.11	Metagene Attractor Predictor	26
1.12	Use of semantic and similarity attributes in face classification . . .	32
1.13	Semantic Attribute Features For Scenes	33
1.14	Semantic Attributes for Clinical Classification	35
2.1	Building Best Models	41
2.2	No Best Classifier	43
2.3	viewtab view of clinical data file from [11]	52
2.4	viewtab view of clinical data file from [11] with scatter plot of two variables.	53
2.5	viewtab view of clinical data showing zoom of points in scatter plot.	54

2.6	viewtab view of clinical data file from [TCGAPANCAN] showing distribution of days to death in LUSC metadata.	55
2.7	viewtab view of single cell data showing distribution of tissue types.	56
2.8	viewtab view of clinical data filtered by keyword "Upper"	57
3.1	Original free-text description in GEO submission	65
3.2	MMTx terms derived from free-text	66
3.3	Parent child hierarchy for Neoplasm Metastasis	67
3.4	Parent child hierarchy for mitochondrial encephalomyopathies	67
4.1	Top Mutation Classifiers	71
4.2	TP53 Cross-Tissue Classifier Performance	72
4.3	Predicting Survival With Semantic Attributes	75
4.4	Predicting Survival With Semantic Attributes	77
5.1	Ontology-aware tissue features.	81
5.2	Training ChromHMM on Epigenetic Marks	83
5.3	chromHMM 15 State Model	84
5.4	chromHMM states across a number of tissues.	85
5.5	Distribution of ChromHMM Contiguous Calls	86
5.6	Topological Domains	88
5.7	Summarizing epigenomic states in chromatin domains.	89
5.8	Building classifiers for epigenomic states in chromatin domains.	90
5.9	Creating samples by domain dataset.	92
5.10	K-means clustering of domains.	93
5.11	Gene Essentiality Screening	95
5.12	Drug sensitivity, gene feature classifiers.	98
5.13	Best Drug Sensitivity Classifiers Without Feature Transform	99
5.14	Drug Sensitivity Classifiers With COMBINED Attributes	101
5.15	Delta Median Drug Performance, Good Classifiers	103
5.16	Delta Median Drug Performance, All Classifiers	104
5.17	Delta Between Best Performing Classifiers	107

5.18	Drugs along with their delta performance COMBINED vs gene level.	108
5.19	Drug Sensitivity Classifiers With MEMO Attributes	110
5.20	Delta Distribution MEMO Attributes	111
5.21	Drug Sensitivity Classifiers With Mutation Attributes	113
5.22	Delta Distribution Mutation Attributes	114
5.23	Drug Sensitivity Classifiers With Chromatin Attributes	116
5.24	Delta Distribution Chromatin Attributes	117
5.25	Drug Sensitivity Classifiers With Gene Essentiality Attributes . .	119
5.26	Delta Distribution Chromatin Attributes	120
5.27	Essentiality Features vs Activities	123
5.28	Pathway enrichment for Essentiality Genes	125
5.29	Genomic Region Around Most Informative Chromatin Feature . .	128
5.30	Genomic Region Around 2nd Most Informative Chromatin Feature	130
5.31	Genomic Region Around 5th Most Informative Chromatin Feature	132
5.32	Performance of Chromatin Attributes	136
5.33	Bits in Chromatin Features	136
5.34	Recurrence of Informative Chromatin Attributes	137
5.35	Drug Sensitivity Accuracy vs Individual Attribute Information . .	138
5.36	Recurrence Frequency of Chromatin Attributes	139
5.37	Precision of Mutation Attributes on CCLE	140
5.38	Recall of Mutation Attributes on CCLE	141
5.39	Semantic mutation vs actual mutation in drug sensitivity.	143
5.40	Expression plus semantic attribues and expression plus actual mutations.	145
6.1	SamplePsychic Upload	151
6.2	SamplePsychic Selection	152
6.3	SamplePsychic Main Results View	153
6.4	SamplePsychic Filter Results	155
6.5	SamplePsychic Filter Results By Score	156

6.6	SamplePsychic Sample Summary View	157
6.7	SamplePsychic Report Card View	158
6.8	SamplePsychic Results Clustering	159
6.9	SamplePsychic Cluster Parameters	160
6.10	Distribution of Fraction of Genes Present	162
6.11	First five levels in Allen Brain Atlas Ontology.	163
6.12	Sixth level in Allen Brain Atlas Ontology.	163
6.13	Train on full gene set, test on full gene set.	165
6.14	Train on full gene set, test on simulated single cell.	166
6.15	Train on simulated single cell set, test on simulated single cell. . .	167
A.1	Exponential Normalization Applied to Cross-Tissue Classification	176
A.2	Exponential Normalization in Cross-Platform Classification	178
A.3	Exponential Normalization in Semantic Attribute Classification .	180

List of Tables

4.1	Top 20 semantic attributes for LUSC survival prediction.	78
C.1	Performance of 117 Classifier Set	186
D.1	COMBINED Drug Classifier ROCs and Their Targets	190

Abstract

Improving Clinically Relevant Classification of Gene Expression Datasets Using Attribute Classifiers as Features

by

K. James Durbin

Predicting clinical outcomes from high-throughput genome-wide data, such as microarrays and RNASeq, is a promising element of both future clinical practice as well as a critical component of research into the molecular underpinnings of disease. A range of statistical and machine learning techniques have been applied to create predictive signatures of clinically relevant patient outcomes from genome-wide data. Most often these techniques have been applied directly to gene-level features, such as RNA expression levels or gene copy numbers. In this thesis I explore a novel method of generating prior-knowledge informed features on which to build predictive models. This method is inspired by analogy to a similar problem in machine vision. Every image recognition system starts with pixels, but pixels are semantically very far removed from concepts we may wish to identify in images such as, for example, a party. By analogy, gene expression levels may be several levels of abstraction removed from the clinically relevant outcomes that are our target. In machine vision a technique called, by some, "semantic attributes", is one way to try to bridge this "semantic gap". The idea is to build classifiers for components of the image that are more concrete, or for which there is additional data, then to use the output of these lower level classifiers to inform an overall scene classifier. Similarly, the idea explored in this thesis is to build classifiers for high-level target concepts on the output of lower-level feature classifiers. That is, classifiers are first built for a range of biological states at various levels of abstraction using genome-wide data known to correspond to those

states. This compendium of cell-state classifiers, or "semantic attributes", is then applied to patient genome-wide datasets. The resulting vector of classifier confidences are then used as features in the clinical prediction task. By incorporating prior knowledge, using a larger and wider range of input data (e.g. including data from different tissues), and reducing the dimensionality of the data by projecting it onto smaller number of knowledge-based dimensions, resulting prognostic predictors can show improved accuracy, improved signature stability across different training sets, improved interpretability of the resulting models as well as highlighting clues to novel contributing factors to patient prognosis.

I have applied this technique to the task of survival prediction in four TCGA datasets: glioblastoma (GBM), ovarian serous cystadenocarcinoma (OV), kidney renal clear cell carcinoma (KIRC), and lung squamous cell carcinoma (LUSC) using a compendium of 119 semantic attribute classifiers. The resulting predictors performed comparably to mRNA based classifiers in two of the four tissues (GBM,OV), worse in one tissue (KIRC), and strikingly better in LUSC. The most informative semantic attribute in the LUSC survival predictor was a semantic attribute for progesterone receptor status that was trained on BRCA data. This illustrates how this technique can take advantage of data outside of the study cohort (here data from an entirely different tissue, BRCA) as well as how the semantic attribute technique can potentially reveal surprising biological connections with potential therapeutic consequences (e.g. the importance of progesterone receptor activity in lung cancer). I present some experiments using fine scaled tissue classifiers to improve survival prediction, and show that the presence or absence of stem cell predictors is a significant feature. I present substantial efforts to use chromatin structure features to improve performance of drug sensitivity prediction, and a much more successful effort to use gene essentiality features to

improve drug sensitivity prediction. Combining all of these panels of semantic attributes results in the most improvement in drug sensitivity prediction.

In addition to these experimental results, I will also present a wide range of software and tools developed to support this effort. Foremost among them is wekaMine, a pipeline for automating large scale classifier experiments, including parameter tuning and classifier chaining, using a domain specific language to describe experiments. wekaMine is built on top of an open source library I created called Grapnel that provides a wide range of tools for data analysis, including a tool, csvsql, for performing SQL queries on tab/comma delimited data files. I will present an open source "big data" spreadsheet application I wrote called viewtab. Viewtab allows one to view, sort, plot, and compute basic statistics on tab file datasets that are much too large to handle in traditional spreadsheets like Excel or Numbers, thus enabling quick exploration of new datasets.

Finally, I will present Sample Psychic, a web app that presents a compendium of classifiers that can be used to illuminate the states of cells from their expression profiles. Expression data uploaded to Sample Psychic can be transformed into semantic attribute space and downloaded for other purposes.

To summarize, the aims this dissertation satisfies are:

- Aim1: Machine Learning Pipeline And Tools
 - wekaMine: A suite of machine learning tools to...
 - * Perform large scale model selection experiments
 - * Facilitate building thousands of classifiers
 - * Encapsulate classifiers into portable models.
 - viewtab: A Big Data Viewer
 - csvsql: Data/Meta-data wrangling tool

- Aim2: Demonstrate prognostic improvements using semantic attributes.
 - Survival Prediction w/ TCGA derived Semantic Attributes
 - Survival Prediction w/ Tissue type Semantic Attributes
 - Drug Sensitivity Prediction w/ Chromatin and Gene Essentiality Attributes
- Aim3: Sample Psychic, Webservice for WekaMine Models
 - Semantic attribute transformations.
 - Cell Type Classifier for CIRM Single Cell Data

Dedicated to Shelley Durbin

Chapter 1

Introduction

1.1 The Need for Prognostic Prediction

One key motivation for prognostic prediction is tailoring patient therapies. Historically patients are treated as a homogeneous group within a given clinical disease type, even though the outcomes of these patients often vary markedly. For example, in one breast cancer study, while chemotherapy or hormonal therapy reduced distant metastasis by one third, 70-80% of patients see no survival difference [1]. Even more strikingly, in a recent ten year study by Albain et al. [2] examining Tamoxafin timing and adjuvant chemotherapy response of patients in node-negative ER positive breast cancer, there was found to be a benefit overall for patients receiving adjuvant chemotherapy in addition to Tamoxafin. However, only 5% of the adjuvant chemotherapy patients actually realized this benefit. The remaining 95% of the patients receiving adjuvant therapy had to endure the many side effects of adjuvant cytotoxic chemotherapy such as alopecia, vomiting, nausea, myelosuppression, and even ovarian failure [3], as well as the costs and time of the treatment, without receiving any compensatory benefit. Notably, in a retrospective analysis, all of the patients who did respond to adjuvant cyto-

toxic chemotherapy were classified in the highest risk subgroup of the 21-gene recurrence signature (OncotypeDX) [4], suggesting the possibility that the likely responders could have been identified in advance sparing the remaining patients of the many negative aspects of the adjuvant therapy.

Before the genomic era this kind of prognosis was performed with such clinical variables as tumor size, nodal status, grade, and immunohistochemistry responses such as estrogen receptor status [5, 6]. The reviews of Vehtari et al. [7] and of Reis-Filho and Pusztai [8] suggest that using traditional features such as those used by Adjuvant! Online and the Nottingham prognostic index have been successful in reducing mortality overall, but that still, even under these protocols almost 60% of patients receive adjuvant chemotherapy while only about 2-15% benefit from it. These tests are adequate for reducing group mortality and adjuvant therapy, but fail to provide adequate guidance for the individual patient. The promise of achieving better prognosis with genomic information is a natural extension to this since it is widely understood that cancer is largely a genetic disease arising from genomic or epigenetic alterations in the programs of cells [9, 10]. Indeed, it has been demonstrated that genomic signatures can in some cases outperform these clinical variables in prognostic prediction. For example, 't Veer et al. [11] performed a comparison of the Adjuvant! Online system which assigns breast cancer risk according to tumor size, nodal status, grade, and ER status with their 70-gene signature. In this comparison it was found that 62% of patients were rated as high-risk with conventional criteria but low-risk with gene signature, and 32% conversely. In the disagreeing cases the genomic test predicted outcome more accurately, with 10-year survival rate for low-risk patients from the genomic test of 89% versus 69% for the low-risk patients from conventional criteria [12, 13].

Since the first gene expression signature in breast cancer by Golub [14], many

genomic signatures have appeared in the literature. Breast cancer in particular has been a testing ground for prognostic signatures, with at least 47 published signatures as of 2011 [15], including at least six commercially available genomic assays [13, 16] and one, MammaPrint, with FDA clearance. These prognostic signatures have proven a valuable aid to therapy with use of such tests increasing so that approximately 30% of stage I and 13% of stage II ER- breast cancers are now tested. Moreover, among those receiving such tests, treatment plans changed in approximately 25-30% of the cases based on the test results [17, 18].

1.2 Gene signatures

So what are gene signatures? The term "signature" is often used ambiguously to describe multiple distinct things. The first and most generic use is as a simple list of genes whose expression pattern is somehow connected with some biological state or medical condition. These are the genes involved in that condition, though information about how exactly they are involved is not contained in the list. The second use is as a prognostic signature where there is typically some information about relative expression levels and a classification decision procedure to assign samples into categories. GeneSigDB [19] and MSigDB [20] are examples of the first kind of gene signature database. These databases contain simple lists of genes associated with biological conditions. These lists are often used with something like Gene Set Enrichment Analysis (GSEA) [21] to determine the significance of overlap between sets of genes whose differential activity is observed in some experiment and curated sets of genes known to be involved in some biological state. These 'bag of genes' signatures do not, by themselves, contain the information needed to build a prognostic classifier. For a signature to be prognostic of some condition additional information (if only an assumption) is needed about

how these genes are involved (e.g. is TP53 over expressed or underexpressed in the condition?) and some procedure is needed to assign new expression profiles to a category based on this information. For example, one may know that 70 genes are associated with breast cancer metastasis, but that list by itself does not tell us what expression pattern indicates metastasis and what expression pattern indicates no-metastasis. A prognostic classifier is a combination of some gene selection and/or feature creation procedure and a decision procedure that specifies how to make a prognosis based on gene expression patterns or the patterns of derived features. Figure 1.1

1.2.1 70-gene Signature Example

There are a variety of ways to build an actual prognostic classifier from simple correlation of genes with some response variable and nearest neighbor evaluation to classify test samples. to complex machine learning algorithms. It is instructive to consider one of the earliest and most successful gene signatures, the 70-gene signature of 't Veer et al. [11] as an illustration. This gene signature was built from the microarray expression results of 78 breast cancer patients. Patients were classified by the detection of distant metastasis in the five year post diagnosis period, with those with distant metastasis considered poor prognosis patients and those with no five year distance metastasis as good prognosis patients. 't Veer et al. [11] identified a subset of genes that were associated with this metastasis status by correlating the signals of each of the 5000 genes in the microarray assay with the metastasis status of the 78 patients Figure 1.2. The significantly correlated genes were retained as a set of 231 candidate prognostic genes. The expression levels of these 231 genes were averaged for the 34 poor prognosis patients to produce an average poor prognosis expression template, and the average of the expression

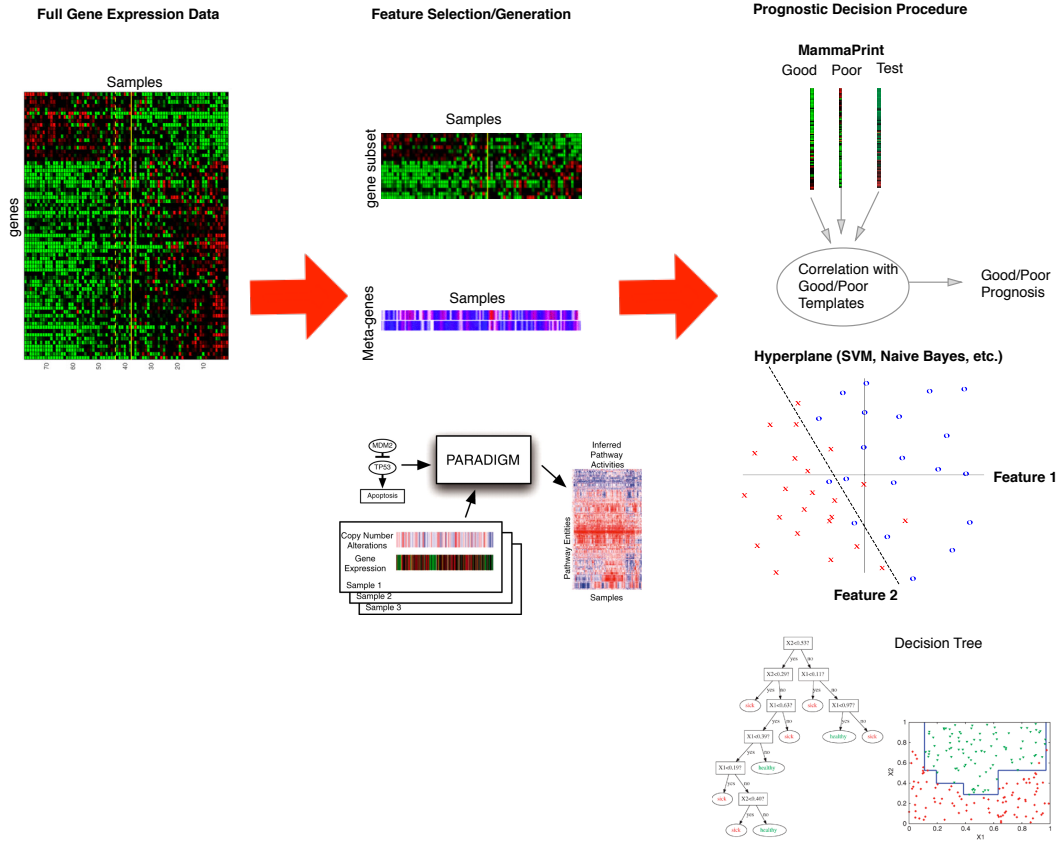


Figure 1.1: Conceptual Stages in Prognostic Classification. Given gene expression data for a set of patients there are two conceptual stages in a prognostic classifier, generating the features and defining a decision procedure. **(A)** Generating the features used can take many forms: simply choosing a subset of the genes (often ambiguously referred to as 'signatures'), generating a set of meta-genes that each somehow summarize the expression of a set of genes, or modeling the pathway activity of genes to get pathway activation levels as features, as well as many others. **(B)** Given a set of features, there are many ways to implement a decision procedure. One may simply correlate new samples with average expression between exemplars of two prognosis, or a hyperplane may be found in k-dimensional space that separates samples into two prognostic groups (where k is the number of features, whether genes, meta-genes, pathways, etc.). Or one may build a decision tree that examines features sequentially in a hierarchical fashion to arrive at a prognosis. And there are many other decision procedures as well, which I will collectively call "classifiers" in this work. Pathway diagram from [22].

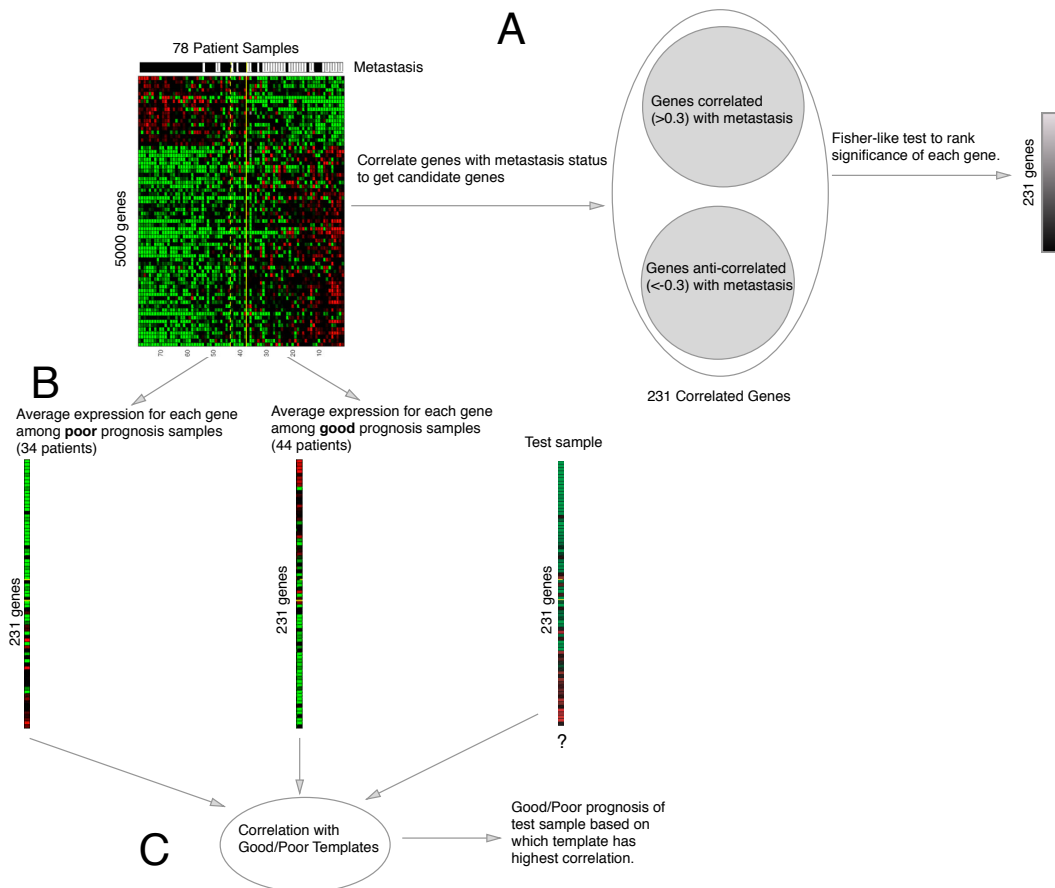


Figure 1.2: 70-gene signature. (A) [11] find a subset of genes that are correlated with the five year occurrence of metastasis. Genes with $|correlation| > 0.3$ are chosen as candidate predictor genes. (B) An average expression profile (meta-gene) is computed from the metastasis samples and another for the no-metastasis samples. (C) A prognosis for a test sample is produced by correlating that sample with each of the template meta-genes. The template with the highest correlation is the prognosis of the test sample.

levels of 231 genes were computed across the 44 good prognosis patients to obtain an average good prognosis expression template. To classify a sample of unknown prognosis the same set of 231 genes of the test sample are correlated to these two templates to see which it is more similar to, the most similar template determining the prognosis call. 't Veer et al. [11] further refine this classifier by determining what subset of this initial set of genes provides the lowest error rate in an iterated leave-one-out cross validation setting. In each iteration one of the 78 samples is held out as the test sample and the remaining used to compute good/poor prognosis expression templates. The prognosis call of the hold-out is evaluated on these templates and then the process is repeated until all 78 samples have been the test sample. A Fisher-like score is used to rank the 231 genes and on each leave-one-out cross validation experiment a smaller set of genes is used, cutting off the gene with the lowest rank each time. It was found that the best error rate occurred at 70-genes, with more genes apparently adding noise and fewer genes apparently eliminating some useful information. Using this simple classifier they achieved an accuracy of 68% (3 poor patients classified as good, 18 good patients classified as poor). Note that in the literature the term "70-gene signature" is often used interchangeably, and sometimes confusingly, between the simple list of 70-genes identified in this study and the full prognostic procedure used to assign patients to good or poor prognosis categories. This 70-gene signature forms the basis for the commercially available test MammaPrint.

1.2.2 Performance Limitations

Although these results are both promising and clinically useful, as Pawitan et al. [17] review, the actual reported accuracy is probably overstated since the feature (gene) selection procedure involved samples that were later used in testing,

which allows some information about the test samples to "leak" into the decision procedure. This flaw was pointed out by Ransohoff [23] and surveyed in Drier and Domany [24]. Such "information leaks" lead to overfitting the decision procedure to the cohort at hand resulting in overoptimistic estimates of algorithm performance on unseen data. Such overoptimistic performance results stemming from improper use of test data in gene selection may plague many published signatures as noted by Ein-Dor et al. [25], Dupuy and Simon [26] and Michiels, Koscielny, and Hill [27]. This suggests that when looking at the reported accuracy of such signatures the room for performance improvement may be even larger than is apparent from reported performance figures. Even taking the performance at face value, though, there is substantial room for improvement. In this case, if adjuvant therapy were based solely on the genetic test, three patients (3.8%) who would have benefitted from it would have not received it, while 18 patients who do not benefit from it (23%) would have to endure cytotoxic therapy which was of no use to them. Given the human costs of the outcomes, even small improvements in performance are worth pursuing.

Many other signatures have been developed since the pioneering 70-gene signature, both many different lists of potentially prognostic genes, and also many different decision procedures applied to these gene lists. Although many of these signatures predict outcome of ER-positive breast cancer, predicting ER-negative breast cancer remains a challenge [8], with even specifically tailored signatures separately trained on ER-negative tumors failing to perform well under meta-analysis scrutiny [28].

1.2.3 Signature Stability and Interpretability

As these gene signatures proliferated it became clear that the sets of prognostic genes identified had very little overlap (Ein-Dor et al. [25] and Drier and Domany [24]). Michiels et al. Michiels, Koscielny, and Hill [29] reported that gene lists derived from seven published cancer studies were highly unstable, and other signatures that followed also contained differing gene lists (Fan et al. [30]). Ein-Dor [31] show that this is perhaps not surprising given the large difference between the dimensionality of features and the dimensionality of the samples. They illustrate that, for example, if 100 samples are used to rank 10,000 genes, selecting the top genes as prognostic, repeating this procedure will produce a gene set with only 2-3% overlap. From a purely prognostic point of view this lack of overlap between genes in different signatures (gene sets) might not be a problem, so long as the resulting prognostic signatures (classifiers) have reasonable performance and between signature concordance. Reyal et al. [32], however, examine nine different gene signatures and found that while all nine sets of genes perform similarly in terms of overall performance at assigning patients to a poor or good prognosis, the concordance in classification between these signatures is low. Only 50% of the patients were classified into the same prognosis category by all gene sets.

The failure of published gene signatures to identify stable sets of prognostic genes raises the question of whether many of these models are simply overfitting the training data or, on the other hand, whether these various gene sets might be merely different subsets of genes from the same underlying pathways and so perhaps more in concordance with one another than is at first apparent. Venet, Dumont, and Detours [15] evaluated the association of 47 published gene signatures with breast cancer outcome. They found that most of the published signatures were not significantly better predictors than random signatures of the same

size (Figure 1.3), and that more than 90% of random signatures were significant outcome predictors. This calls into question the idea that the specific sets of genes themselves have carried much information. Further drawing into question the actual importance of the specific genes in these signatures, **HaibeKains:2008dr** examined 13 different combinations of gene sets, dimensionality reduction methods, and machine learning algorithms. They found that complex methods were not significantly better than the simplest univariate model based on a single proliferation gene.

1.3 Representation Based Improvements to Prognostic Classification

A number of approaches to improving prognostic prediction and signature stability/interpretability have been pursued in recent years. Three such approaches will be discussed here, gene sets (Lee et al. [33]), pathway inference (Vaske et al. [22]), and metagene attractors (Cheng, Yang, and Anastassiou [34], Cheng, Yang, and Anastassiou [35]). Finally, I introduce a new technique that I call "semantic attributes". All four of these approaches share in common that they focus on the gene selection/feature creation stage of building a prognostic classifier (Figure 1.1) and seek to transform gene expression data into a more biologically meaningful representation, leaving the exact classifier/decision procedure as a hyper-parameter to be chosen.

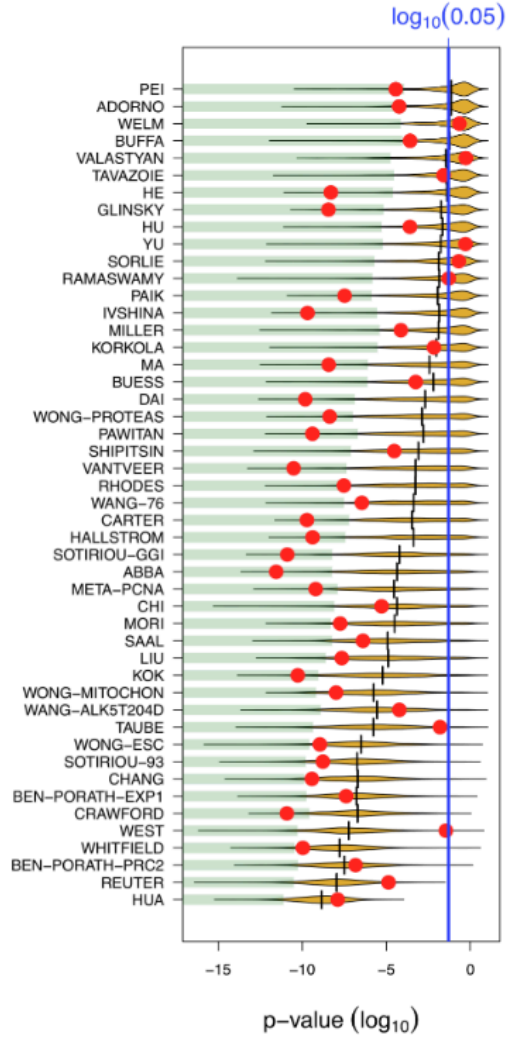


Figure 1.3: Most published signatures are not significantly better outcome predictors than random signatures of identical size. The x-axis denotes the p-value of association with overall survival. Red dots stand for published signatures, yellow shapes depict the distribution of p-values for 1000 random signatures of identical size, with the lower 5% quantiles shaded in green and the median shown as black line. Signatures are ordered by increasing sizes. Figure from: Venet, Dumont, and Detours [15]

1.3.1 Importance of Representation in Classifier Construction

The importance of representation in machine learning, which overlaps with dimensionality reduction, is well known (Clarke et al. [36], Wang and Gehan [37], Guyon et al. [38], Domingos [39]). In order to further motivate the importance of representation and perhaps convey an intuitive sense of why it matters it may be useful to examine a toy example. The goal in this toy problem is to learn whether pedestrians consider two destinations in a city to be "within walking distance". The input is a set of location pairs and pedestrian rating of "walkable" or "not-walkable". If the input is given as a set of city-block coordinates, a North-South street number and it's intersecting East-West street number, this prediction task can be quite difficult. As can be seen in Figure 1.4, none of the four coordinate input variables is correlated with the output nor, in fact, are any pair of input variables correlated with the output.

A classifier attempting to find some decision boundary in this representation will face difficulties. A classifier with a simple hypothesis space, a linear decision boundary say (e.g. Linear-SVM, Naive Bayes), will not be able to separate good and bad examples at all (Figure 1.5). A classifier with a more complex hypothesis space (e.g. Polynomial-SVM, Decision Tree) may be able to carve out a better decision boundary for the training samples, but this more complex boundary runs the risk of overfitting the training set: drawing a complex boundary that fits the training data but which does not capture any generalizable relationship of the data that will work on new test samples. Of course, the relevant semantic idea for two destinations to be "walkable" is the geometric distance between the two points. While coordinates contain all the information needed to compute the distance the raw coordinates by themselves tend to make this relationship obscure. If, however,

A

NS Street Start	EW Street Start	NS Street End	EW Street End	Walking Distance	Walkable
12	3	35	44	64	No
49	40	21	11	57	No
2	29	36	44	49	No
49	40	23	19	47	No
22	17	28	50	39	No
15	14	46	22	39	No
41	22	12	12	39	No
2	45	50	31	34	No
17	13	22	41	33	No
11	21	11	51	30	No
33	50	45	9	29	No
21	22	10	4	29	No
22	2	37	16	29	No
34	30	31	5	28	No
27	11	29	32	23	No
10	10	22	12	14	No
30	26	3	40	13	No
29	7	14	10	12	No
34	4	18	8	12	No
33	10	23	11	9	Yes
11	19	20	2	8	Yes
5	38	4	33	6	Yes
29	6	7	24	4	Yes
36	30	45	24	3	Yes
32	24	32	21	3	Yes
41	43	42	45	3	Yes
12	22	12	24	2	Yes
13	41	12	43	1	Yes
34	22	47	9	0	Yes

B

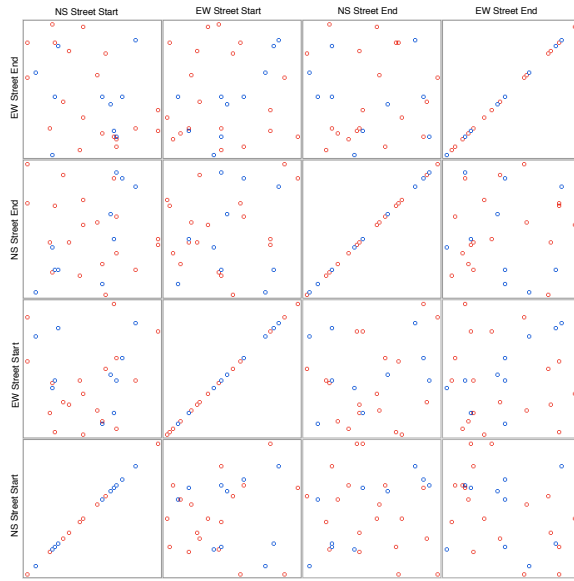


Figure 1.4: Toy City Block Example(A) Toy example to predict whether pedestrian considers destinations "in walking distance" from city block coordinates. (B) Using the raw coordinates, none of the four coordinates (omitting the derived walking distance field), nor any pair of the four coordinates, are correlated with the prediction variable.

we make this concept evident by transforming the coordinate representation of the pairs of destinations into the distance between the destinations, which can be done with the very simple formula $D_{walking} = |(NS_2 - NS_1)| + |(EW_2 - EW_1)|$, the prediction task suddenly becomes trivial (Figure 1.5,C).

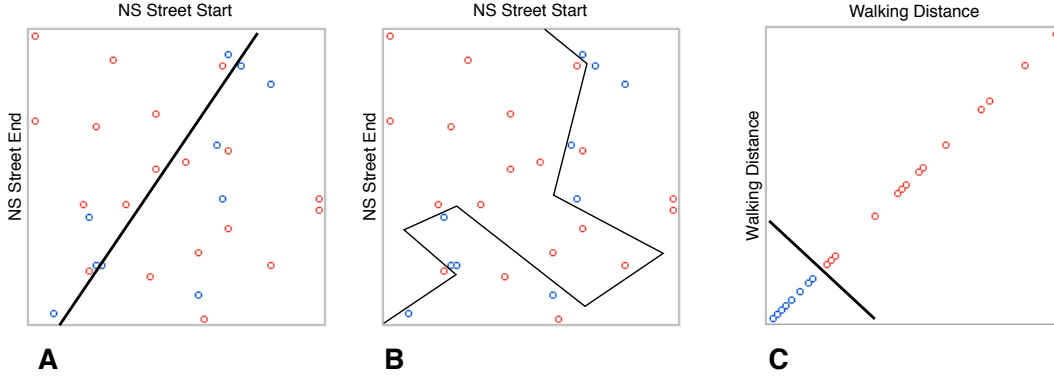


Figure 1.5: Effect of representation in Toy City Block Example(A) In the coordinate representation there is no simple line that separates positive and negative examples. (B) Using a more complicated boundary function it is possible to draw a better boundary with fewer errors but the possibility of overfitting grows with the complexity of the boundary. (C) When the input data is transformed with a simple formula into walking distance, it is easy to separate positive and negative examples in a way that is likely to generalize.

To illustrate the effect of this representation change I performed a simple test with the data from Figure 1.4 and found that both linear and quadratic support vector machines have a cross-validated accuracy of only 50%, and a fourth-order SVM, which in some sense is able to consider the interaction of four variables at once and so has more chance of finding a boundary in this four coordinate scenario, achieves only 62%. With the derived variable "walking distance" added, however, a linear SVM achieves a cross-validated accuracy of 98%. Indeed, in this transformed coordinate space very simple classifiers, single node decision trees and Naive Bayes, achieved perfect cross-validated accuracy of 100%. Although this is just a toy example, with no noise or other complicating factors, it illustrates the point that representation can be the dominant factor in the performance of a

classifier. Indeed, for many successful machine learning endeavors the largest part of the effort goes into feature engineering to develop meaningful features from raw data (Domingos [39]). It is worth observing that while the learning algorithms that underlie prognostic prediction (Support Vector Machines, Random Forests, and so on) are very generic and mature, with the same algorithms applied unaltered to a wide range of problems, the problem of feature engineering is typically very domain specific and usually draws on a significant amount of domain knowledge from the field of application (geometry, finance, vision, biology, etc.). I will briefly review three feature transformation techniques to illustrate the idea of feature space transformation in practice. It's worth noting that all three make use of some form of background biological knowledge in order to transform their feature spaces. A key motivation for the semantic attribute technique explored here is the ability of the technique to employ a wide variety of different kinds of background knowledge in a semi-automated way.

1.3.2 Pathway Activity Inference using Condition Responsive Gene Sets

In order to address both the stability issues with traditional signatures as well as to improve the accuracy of prognostic predictions several groups have turned to ways to create features from sets of genes defined by curated PPI networks (Chuang et al. [40], Taylor et al. [41]) or curated pathways (Abraham et al. [42], Staiger et al. [43]). Among these, Lee et al. [33] is the most widely cited. In their method Lee et al. [33] use gene sets from the MsigDB (Liberzon et al. [20]) to represent pathways and try to define a pathway activity for each sample based on these gene sets. Like many early pathway approaches, they do not consider the actual known or hypothesized interactions in the pathway but merely use the set

of genes of the pathway as a proxy for that pathway. In their method they explicitly use the target output to identify the subset of genes from each pathway set that are most discriminative (via a t-test) for the target. This makes their method a fully supervised method. They call this subset of a pathway set the “condition responsive genes” or CORGs of the pathway, and they call their technique overall Pathway Activity from CORGs (PAC). They then compute an activity level for each sample over each pathway that is proportional to the sum of the z-scores for each sample and each CORG gene relative to the target condition (Figure 1.6).

The result is a transformation of the genes by samples input dataspace into a pathway activity by samples input space. They tested the ability of this feature space to enhance classification by using these derived features to train a logistic regression classifier. They compared their results to gene features alone, principal components analysis (PCA) dimensionality reduction, and simple mean and median pathway gene scores and found that across several datasets this technique had equal or better performance (see Figure 1.7).

In addition to this improved performance using this technique has the advantage of adding a straightforward explanatory component to the classification. By clustering pathway activities between good and poor prognosis patients it is possible to get a picture of the underlying pathways driving the prognosis prediction (Figure 1.8).

Although Lee et al. [33] found improvements in their own tests of the PAC procedure a subsequent evaluation of three pathway methods by Staiger et al. [43] casts some doubt on the efficacy of this and two related procedures. In their evaluation Staiger et al. [43] look at the pathway based feature transformations of Chuang et al. [40], Lee et al. [33], and Taylor et al. [41]. The algorithm of Chuang et al. [40] defines an activity level for subnetworks in a protein-protein interaction

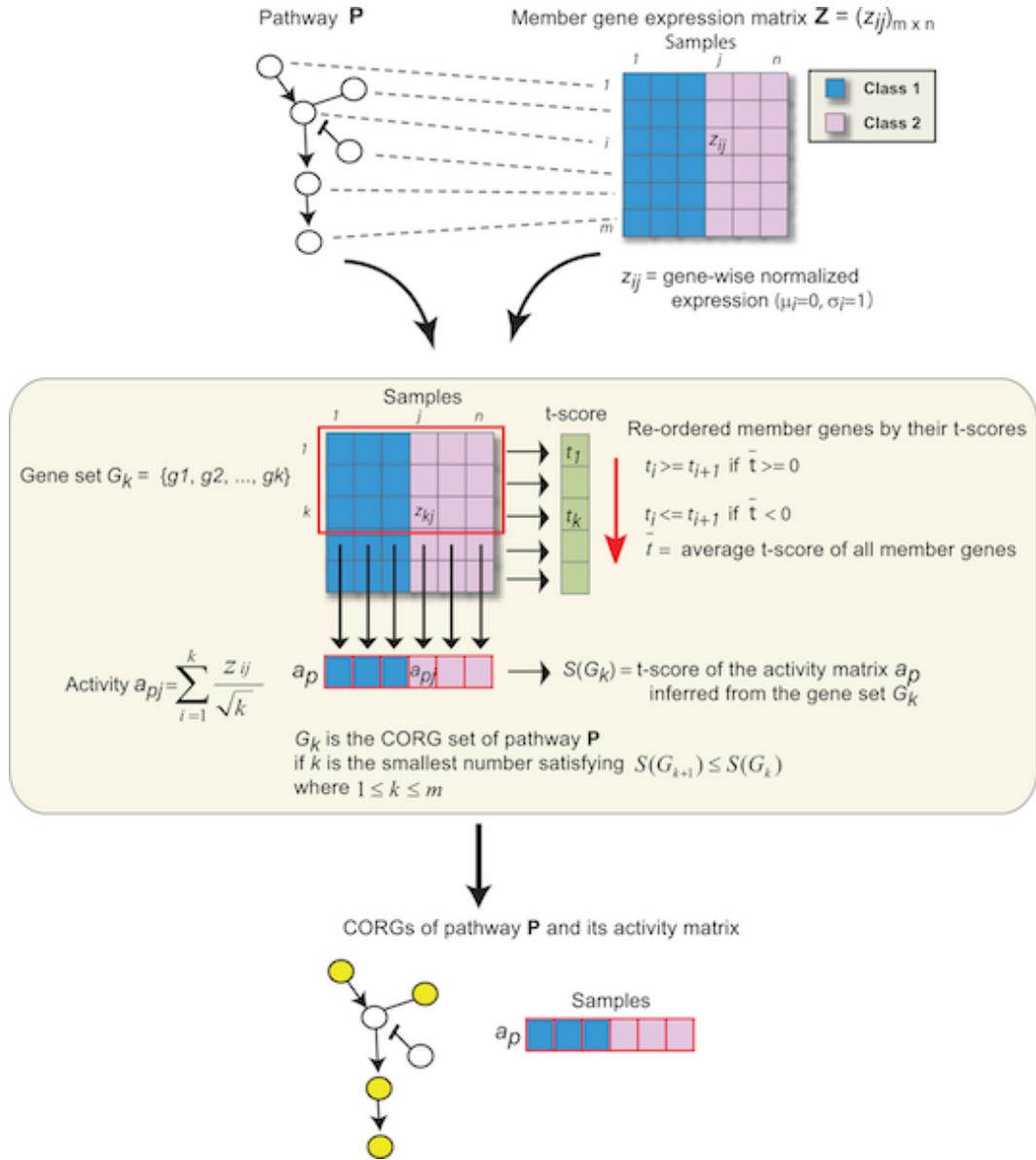


Figure 1.6: Condition Responsive Gene Sets, Gene expression profiles of patient samples drawn from each subtype of diseases (e.g., good or poor prognosis) are transformed into a pathway activity matrix. For a given pathway, the activity is a combined z-score derived from the expression of its individual key genes. After overlaying the expression vector of each gene on its corresponding protein in the pathway, key genes which yield most discriminative activities are found. The pathway activity matrix is then used to train a classifier. From Lee et al. [33] (Note: although this figure shows the connectivity of the pathway, only the set of genes for the pathway is actually used.)

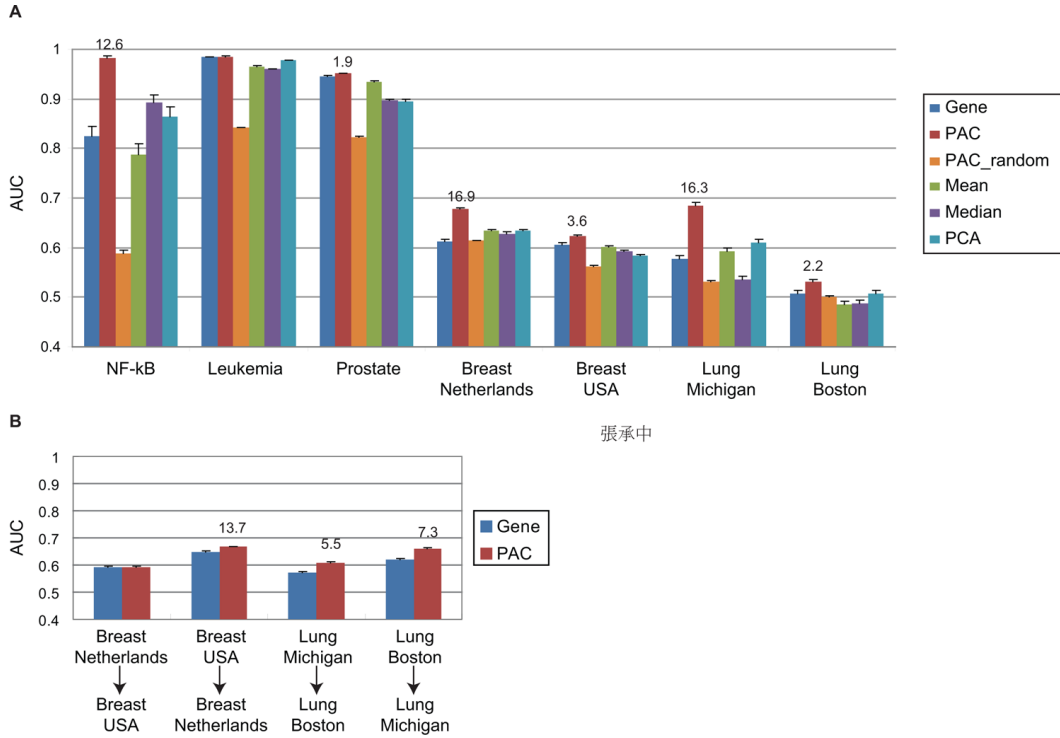


Figure 1.7: PAC Accuracy within (A) and across (B) datasets Bar chart of Area Under ROC Curve (AUC) classification performance of CORG-based pathway markers (PAC), conventional pathway markers (Mean, Median, and PCA), and individual genes. Classification performance is summarized as mean of AUC over 100 runs of 5-fold cross-validation within a dataset. To compute PAC_{random} , the AUC values of 1000 sets of random gene sets were averaged. Numbers above the red bars are $-\log(p - value)$ from the Wilcoxon signed-rank test on the 500 AUCs of PAC against those of Gene (only the ones with p-value, 0.05 are shown). The p-values measure the significance of difference between PAC and gene-based classification. Figure from Lee et al. [33]

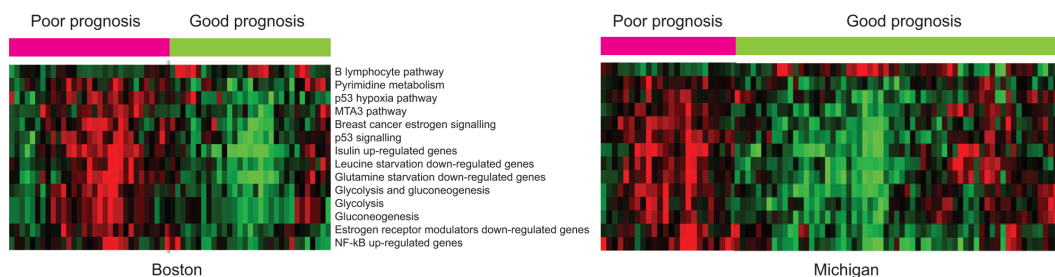


Figure 1.8: PAC Pathway Activity of top markers in two lung cancer datasets Activities were inferred from CORGs identified from each dataset. Green/red blocks indicate pathways (rows) that are up-/down- regulated in patients (columns) of specific prognosis (above color bars: pink and green indicate poor and good prognosis, respectively). Pathways are clustered based on the similarity of their activities across patients. Figure from: Lee et al. [33]

(PPI) network that is proportional to the average expression values of the genes in the subnetwork. They then do a greedy search over subnetworks to identify discriminative subnetworks relative to the target prediction (i.e. metastatic or non-metastatic). Taylor et al. [41] attempt to identify highly connected proteins in the interaction network, proteins that they call hubs, and detect changes in the correlation of samples with these hubs and the genes the hubs immediately interact with. Although Taylor et al. [41] do not actually use their technique to enhance classifier performance, Staiger et al. [43] reasonably see it as another feature transformation technique similar in spirit and evaluate this feature transformation alongside Chuang et al. [40] and Lee et al. [33]. In addition to the three pathway based feature extraction methods they also select subsets of prognostic genes by calculating a t-stastic between mRNA distributions of two patient groups and choosing the highest k associated genes. Staiger et al. [43] evaluate these three feature transformations and one feature selection technique on six breast cancer data sets using three different classifiers as the decision procedure: nearest mean classifier, logistic regression, and the 3 nearest neighbor classifier. Over many validation folds, although Lee et al. [33] performed best at the tail of the distri-

bution of AUC (area under the ROC curve) values, ultimately the difference in performance between Lee et al. [33] and a selection of single best genes was not significant.

1.3.3 Inferred Pathway Activities

Vaske et al. [22] move beyond the realm of bags-of-genes as representatives of pathways to actually utilizing the connectivity of pathways to model patient-specific genomic alterations. They note that mutations in different genes of cells often participate in a common pathway and alter the pathway output in the same direction (e.g. Ciriello et al. [44]). They further note that the collective knowledge about the interaction between genes and their phenotypes is increasing, and that databases such as Reactome (Joshi-Tope [45]), KEGG (Kanehisa [46]), and National Cancer Institute (NCI) Pathway Interaction Database (PID) include not just gene sets but also curated information about the interactions between genes and their phenotypic consequences, making this knowledge accessible to inference algorithms. Vaske et al. [22] use the hypothesized causal relationships implied by these pathways to construct a probabilistic graphical model mirroring the central dogma of molecular biology. The specific algorithmic model they use is a special case of a Bayesian network (Heckerman, Geiger, and Chickering [47]) called a factor graph (Kschischang, Frey, and Loeliger [48]). A Bayesian network is a graph with nodes that represent unknown parameters and edges that represent conditional dependencies between the nodes. The translation of the central dogma and other gene, protein, and pathway relationships into such a graph is conceptually straightforward, if difficult in many technical details. For example, the action of a transcription factor on the transcription of a gene is a conditional dependency, an edge, between that transcription factor and that gene. Similarly, the transla-

tion of a protein from mRNA is a conditionally dependent on the transcription of the gene, etc. Such dependencies can be enumerated from constituent proteins to complexes, from signaling proteins to other transcription factors, and from various proteins to conceptual pathways or cellular events such as apoptosis. In addition, various conditional relationships, such as OR (productA produced if productB OR productC produced), can be modeled in this way. Some kind of background prior on the various unknowns must be assigned, often with global background training on some cohort of samples. Once constructed and initialized with priors, known information from individual patients, such as mRNA expression levels from microarray or RNASeq experiments, or protein levels from PPI, mutations, or any combination of these can be used to update the probabilities of the nodes in the graph. An inference algorithm can then propagate these updated probabilities through the graph to update the posterior probabilities of all the other nodes. The net result is a set of integrated pathway activities (IPAs) for a specific patient (Figure 1.9).

While the focus of Vaske et al. [22] is the elucidation of causal relationships between genomic alterations and cellular events or responses rather than predicting patient outcomes, they clustered a set of glioblastoma multiform (GBM) patients based on pathway perturbations and found that the members of these clusters have significantly different survival outcomes suggesting the potential utility of the technique in prognostic prediction.

One of the problems faced by pathway approaches using the interactions between terms is the very high connectivity among genes in pathways. This high connectivity greatly complicates attempts to identify clear subnetwork in a pathway and this makes isolating causal effects challenging (Figure 1.10). If modeling pathways and their interactions doesn't prove to be an effective feature trans-

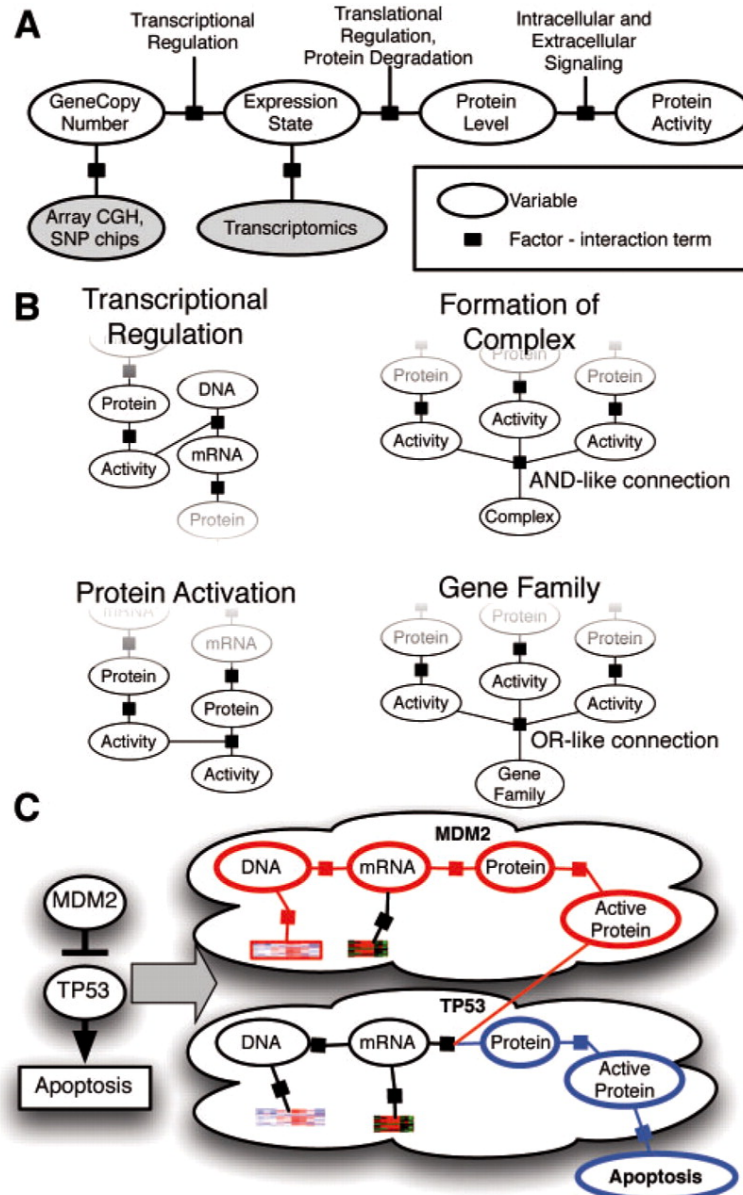


Figure 1.9: PARADIGM A. Data on a single patient is integrated for a single gene using a set of four different biological entities for the gene describing the DNA copies, mRNA and protein levels, and activity of the protein. **B.** PARADIGM models various types of interactions across genes including transcription factors to targets (upper-left), subunits aggregating in a complex (upper-right), post-translational modification (lower-left) and sets of genes in a family performing redundant functions (lower-right). **C.** Toy example of a small sub-pathway involving P53, an inhibitor MDM2, and the high level process, apoptosis as represented in the model. Figure from Vaske et al. [22]

formation this high connectivity may be one factor for why. Another possible limitation to the pathway approach are limits in our pathway knowledge. Many of the links are merely hypothesized, some may be erroneous, and many real links may be missing from our pathway databases.

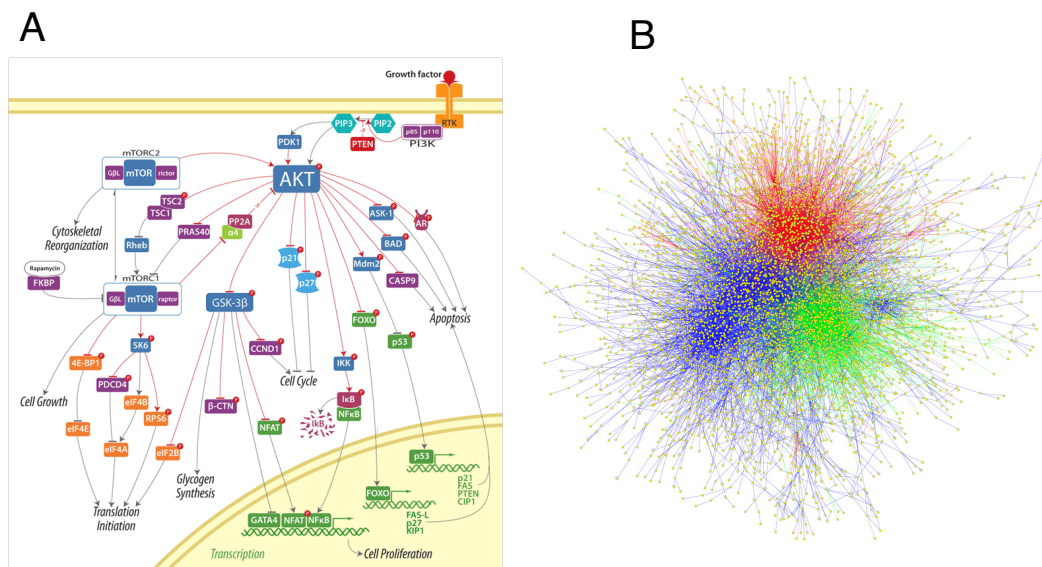


Figure 1.10: (A) Idealized subnetwork of a set of protein interactions and (B) full network of interactions. Given the high connectivity of biological pathways even exploring a few edges out from a modest number of nodes quickly connects large portions of the graph. Figure from Ferrell [49]

1.3.4 Meta-gene Attractors

Taking another direction from the pathway centric approach, Cheng, Yang, and Anastassiou [34] have developed a feature space transformation based on the idea of deriving signatures for specific cellular events by employing an iterative meta-gene algorithm. Like many other techniques the goals of this transformation are twofold. First they desire to identify stable metagene signatures that correspond to biomolecular events such as cell differentiation or the presence of an amplicon, and then they seek to use these metagenes to transform noisy low-

level gene expression signals into a more prediction-meaningful representation in order to improve prognosis prediction or subtype identification. Cheng, Yang, and Anastassiou [34] contrast their approach to the popular Non-negative matrix factorization (NMF, Brunet et al. [50]), by highlighting the latter’s focus on dimensionality reduction and representing a particular corpus of data as accurately as possible as opposed to merely seeking stable surrogates for biomolecular events.

A metagene for Cheng, Yang, and Anastassiou [34] is a weighted linear combination of gene expression values. The authors sometimes ambiguously use the term ‘metagene’ to refer to the final sum of such a linear combination, and sometimes as a reference to the vector of weights in this linear term. The metagene weights are derived by a simple iterative algorithm that will tend to converge to a stable set of values. The key computation in their approach is a measure of distance between two vectors of values, $J(\mathbf{V}_1, \mathbf{V}_2)$. A variety of measures might work, such as correlation, but the authors choose mutual information. Given this measure, the algorithm in outline follows. An initial vector of metagene weights, $\mathbf{M}_0$ are chosen randomly. The weight for each gene i , w_i , is updated by measuring the distance J between a vector of that gene’s expression values across samples, $\mathbf{G}_i$, and the random weights of the initial metagene $\mathbf{M}_0$. Doing this for every gene g_i yields a new vector of weights, $\mathbf{M}_1$. This new vector then becomes the comparison vector and the process is repeated, either until the change in the weight vector gets smaller than some threshold or for a fixed number of iterations. Pseduocode

can be seen in Algorithm 1.

input : A set of gene expression vectors for each gene $\mathbf{G}_i : i \in [1, n]$ and a measure of association between two vectors $J(\mathbf{V}_1, \mathbf{V}_2)$ (e.g. mutual information) output : A set of weights for metagene $\mathbf{M}$ $\mathbf{M}_0 = (w_1, w_2, \dots, w_n) \mid w_i = rand(0, 1);$ while $iterations < maxIterations$ do <table border="0"> <tr> <td style="border-left: 1px solid black; padding-left: 10px;"> $\mathbf{M}_1 = (w_1, w_2, \dots, w_n) \mid w_i = J(\mathbf{G}_i, \mathbf{M});$ $\mathbf{M}_0 = \mathbf{M}_1;$ </td> </tr> </table> end	$\mathbf{M}_1 = (w_1, w_2, \dots, w_n) \mid w_i = J(\mathbf{G}_i, \mathbf{M});$ $\mathbf{M}_0 = \mathbf{M}_1;$
$\mathbf{M}_1 = (w_1, w_2, \dots, w_n) \mid w_i = J(\mathbf{G}_i, \mathbf{M});$ $\mathbf{M}_0 = \mathbf{M}_1;$	

Algorithm 1: Meta-gene Attractor Algorithm

Each time the algorithm is run with a different random starting point it has a chance to converge on a different attractor. Running this algorithm many times on six ovarian, two breast cancer, and two colon cancer datasets yielded 61 general attractors that spanned all of the datasets as well as 241 additional attractors that were specific to individual datasets. To apply these metagene weights to transform new data the authors selected only the genes with a certain minimum weight, so a metagene derived from thousands of genes might in the end only depend on three to ten genes. The authors applied this technique to the Sage Bionetworks / DREAM7 Breast Cancer Prognostic Challenge *Sage Bionetworks-DREAM Breast Cancer Prognosis Challenge* [51]. They selected fifteen of the metagenes identified for feature space transformation, 3 from the set of global metagenes, and 12 from the breast specific set of metagenes. Using these metagenes they transformed each sample into a metagene activity matrix Figure 1.11.

This transformed dataset was used to train three classifiers whose outputs were combined in a weighted voting scheme to generate final predictions. This challenge

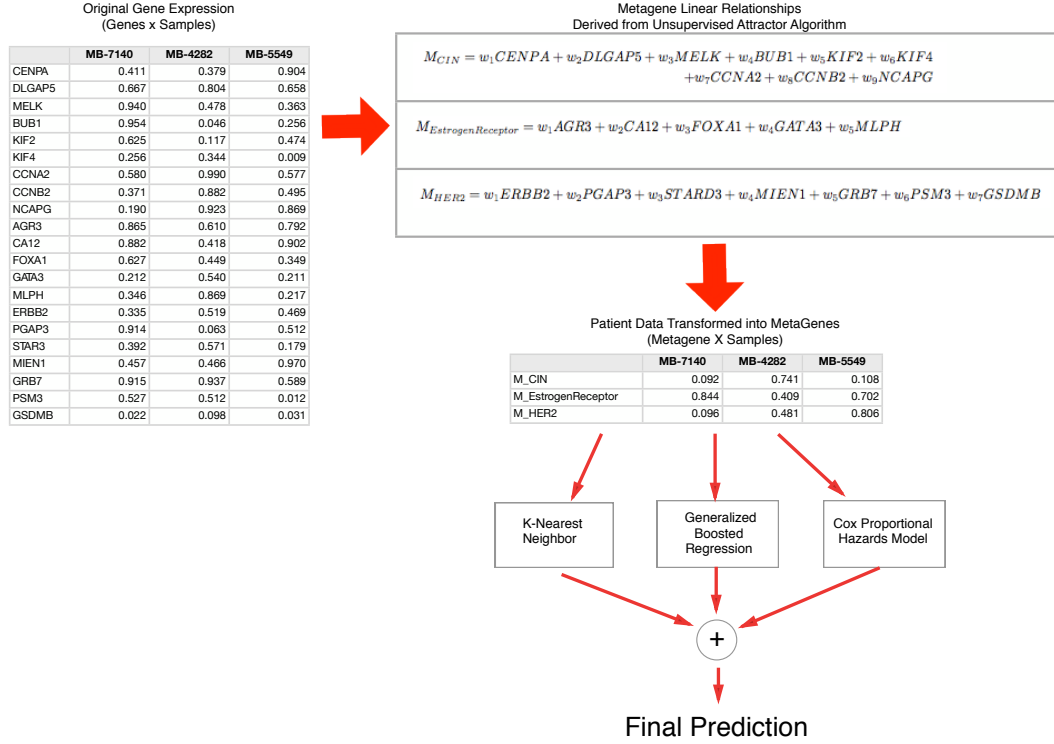


Figure 1.11: Metagene Attractor Predictor A linear combination of gene expression values yields a metagene value. Transforming a vector of gene expression values for a given patient into a vector of metagene values for each patient transforms the data from gene space to metagene space. The transformed data is then used as input to three different classifier algorithms whose independent outputs are combined in a weighted voting scheme to arrive at a final prediction for a given sample.

was evaluated with a concordance index (CI) measure. The CI measures the relative frequency of correct ordering for randomly chosen pairs given the survival ranking output by a predictor. For random predictions the average CI is 0.5. The final CI of the Cheng, Yang, and Anastassiou [34] model on an independent dataset was 0.756, which was the winning model out of 350 entrants and over 1700 submitted models.

It is worth highlighting that the fifteen meta-genes used were derived from a range of datasets beyond those in the training set provided by Synapse. One thing this potentially illustrates is the utility bringing in more data and more information from beyond the training cohort in a given experiment. The amount of data available in most experiments is so small compared to the dimensionality of the gene features that it may be that even weak information from outside sources can have a significant impact.

1.4 Semantic Attributes

There are several possible limitations to these approaches. Gene sets such as used by Lee et al. [33] may face limitations from the lack of information about the relationship between genes in a pathway and, in any case, Staiger et al. [43] have cast doubt on the efficacy of their technique as compared to simple single gene models. Pathway approaches such as PARADIGM (Vaske et al. [22]) that use state of the art belief updating algorithms and detailed knowledge about how biological entities interact hold the promise of producing detailed patient specific inferences about the state of cells from the integration of various kinds of experimental data. On the other hand, the dense connectivity of the underlying knowledge graph coupled with uncertainty and possible errors and omissions in the pathway knowledge base could potentially undermine some of the utility of

this method.

Looking for new knowledge-driven ways to improve the feature space for prognostic classification I introduce the idea of "semantic attributes". The idea of a semantic attribute is to build a classifier, possibly a complex non-linear classifier, for some known cellular state and then to use these classifiers to transform data in gene expression representation into a representation based on cellular events.

As Clarke et al. [36] note, there are many difficulties with using gene-level features. Regulation or expression of some genes depends on cell and tissue types and sometimes even varies among individuals. Some genes can perform different functions in different cells and different genes can sometimes perform the same function in different contexts. This highly context dependent nature of the effect of genes greatly complicates connecting genes to specific phenotypic outcomes. Further complicating the picture is the fact that many different cellular states may be superimposed at the same time. A cell may be in the state of responding to a shock, it may be replicating, it may be in the state of being a cell of a particular tissue or of a particular subclass of cancer cell, or it may be in all of these states and more simultaneously. Tumors, in any case, are not homogenous monoclonal masses but complex assemblages of stromal and epithelial cells, with cancer cells actively promoting certain intercellular interactions such as releasing PDGF to stimulate reciprocal IGF-1 (insulin-like growth factor-1) release from stromal cells, encouraging the growth of endothelial cells, etc. A. [52]. Looking globally at gene expression data it may, in some cases, be easy enough to pick up on a dominant global state, a proliferation signal say or a signal of high metabolism, and to develop classifiers for phenotypes that are correlated with this global state, but this may be only capturing a crude level of association between gene states and phenotypes. In other cases it may be advantageous to decompose the expression

profile into its membership in these individual cell states and use these states as the features in the phenotype classification.

At a technical level, many classification schemes struggle to overcome platform biases, tissue biases, and even batch biases (Shi et al. [53], Luo, Schumacher, and Scherer [54], and Chen et al. [55]). By decomposing the classification problem into a series of more tightly focused classification tasks with possibly more training data, and more heterogeneous training data, and then combining these focused tasks it may be possible to overcome some of these limitations. For example, one can potentially obtain data for P53 mutated cells from a variety of tissues, platforms, and disease states. Using such heterogeneous data it is possible we can train a P53 mutation classifier that is somewhat independent of platform, tissue, or other confounding variables. In predicting breast cancer patient survival, the pool of samples for which we have survival information is seriously limited, but the pool of samples from which we can produce a P53 classifier may be much larger. It may thus be possible to build a P53 mutation classifier that is more robust and independent of sample variation than a classifier that attempts to predict higher level phenotypes, like survival, directly from the very limited set of survival data available. A collection of such mid-level feature classifiers applied to the limited pool of survival samples available may be able to better illuminate what is going on in the cells of those samples than globally looking at all the genes of just those patients.

1.4.1 Semantic Attributes in Machine Vision

A similar set of problems arises in the area of computer vision and these provide a useful set of analogies, as well as a successful antecedent, for the approach explored in this work. While pixels in an image capture a wealth of information

about a visual scene, pixels themselves are a very low level representation relative to the objects one may want to identify in a scene. Typically a visual scene is first decomposed into higher level features like lines, edges or, more realistically, sets of scale invariant features such as produced by the SIFT (Lowe [56]) feature extraction system. Even these features are somewhat semantically motivated, unlike principle components analysis, insofar as we have a prior expectation of what kinds of features are important in a visual scene. More recently, however, several groups have attempted to improve the performance of some image classification tasks by employing overtly semantic features (Kumar et al. [57], Parikh and Grauman [58], and Su and Jurie [59]). Kumar et al. [57], for example, looked at the task of face verification, that is of building a classifier to reliably identify an image as containing an image of a particular person, say Halley Berry. The features a human might describe for the image such as: adult, male, asian, blond hair, etc. they called "semantic features", and the training set for these features were human curated (see Figure 1.12). Another set of features that they call "similarity" features attempts to capture the idea of similarity of facial features (eyes, nose, mouth, etc.) to a limited set of reference faces. Classifiers were trained for both the semantic and the similarity features, and images were scored by a suite of such classifiers to get a matrix of classifier outputs that supplied the input to an overall face recognition classifier (e.g. a classifier that says the image is/isn't Halle Berry). This method of using the output of low level concept classifiers as the input for a face classifier was tested on the Labeled Faces in the Wild (Huang et al. [60]) database of labeled faces of people in unconstrained settings. Semantic feature classifiers improved the classification error rate by 24% over state of the art methods and simile classifiers by 26%, and combining the two kinds of features achieved a 31% improvement in error rate over state of the art methods. Li et al.

[61] have taken this idea and extended it to the idea of scene recognition, where the goal is to label an image as, for example, "a sailing scene", or "a rowing scene". In this case low level features can easily pick up on, say, the "water" aspect of these images, but have more difficulty correctly identifying the activity without some sub-scene level semantic features. Li et al. [61] call these issues the "semantic gap" between the image representations (pixels or even SIFT-level features) and certain image recognition goals. It is worth noting that the individual attribute classifiers in these experiments were far from perfect. In both Su and Jurie [59] and Kumar et al. [57], for example, the semantic attributes had individual accuracies ranging from 65% up to 95%, with at least half of the attributes below 80%. This is important because one natural concern with the semantic attribute idea is whether the imperfect semantic attribute classifiers might, in effect, just be adding more noise to the input and doing more harm to the classification problem than they help. The range of accuracies for semantic attributes in these successful applications suggests that semantic attribute classifiers need not be absurdly accurate to be effective and that classifiers can make use of a combination of features that convey weak information.

1.4.2 Biological Semantic Attributes

By analogy to these image classification tasks, predicting endpoints like survival, metastasis, drug response, or other clinical variables from microarray data could be imagined a bit like recognizing a face or the kind of scene from pixels. Gene expression is a low level representation with a large semantic gap relative to the target endpoints. Similarly, there are a wealth of known biological concepts that have a lower semantic gap and for which gene expression profiles have been produced, concepts for which we could train biological semantic attribute clas-

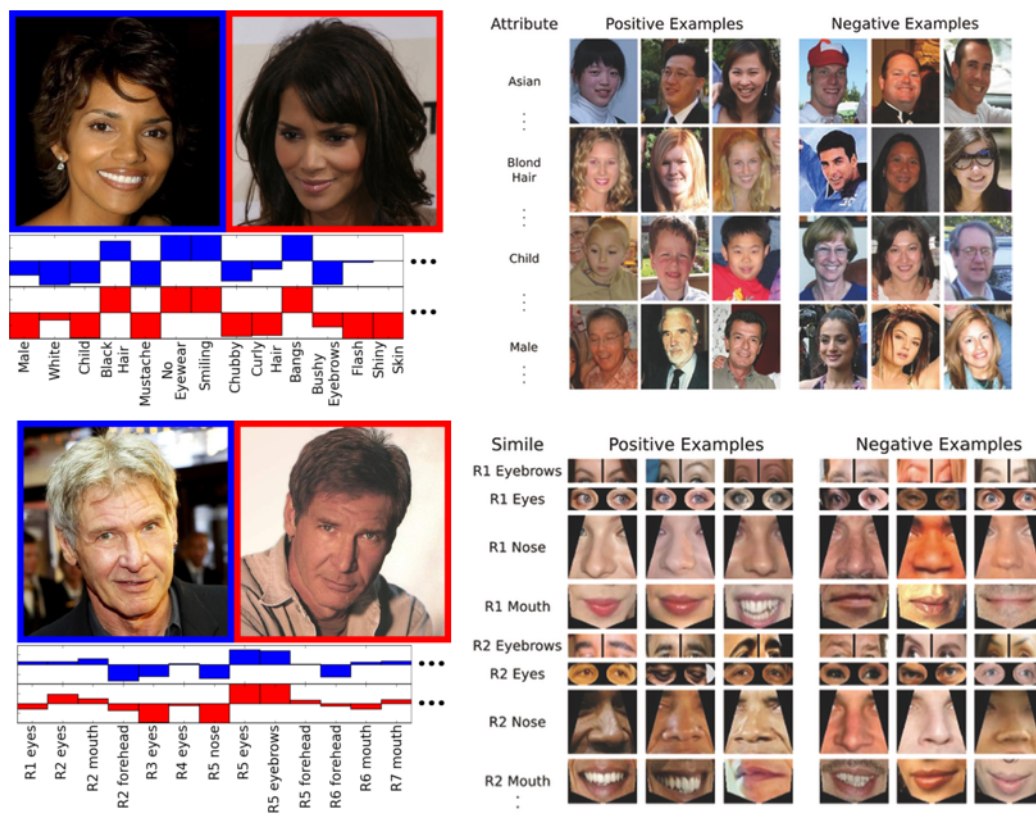


Figure 1.12: In (A) Kumar et al. [57] create training sets for semantic features using images that are human scored as containing a range of semantic attributes. In (B) they take photos of facial features for a number of people as positive examples and photos of those same features from a range of different people as negative examples and build "mouth like subject R1", "mouth like subject R2", etc. classifiers. These simile classifiers are then used as semantic attributes in the face recognition task.

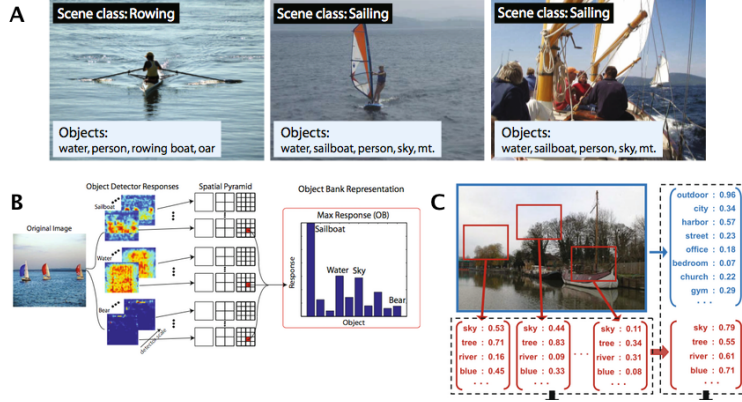


Figure 1.13: (A) Distinguishing between complex scenes can be aided by decomposing the scene into semantic attributes. In these complex scenes, the first and second image are more globally similar, although a human would classify the middle and right images as belonging to the same event class. (B) This can be done globally, over the whole image using semantic categories that can then be combined to make scene classifications Li et al. [61], (C) or it can be done by splitting the scene into independent object regions and classifying those subregions into semantic attributes that can be combined for scene recognition. Su and Jurie [59]

sifiers Figure 1.14. Some of these concepts are available as clinical variables in some studies, things like tumor grade, immunohistochemistry responses and so on. Other concepts could include genomic or cellular events like the presence of specific non-synonymous mutations, or genomic instability. Still other concepts could include cell states, such as tissue type, tissue layer, cell cycle states, transcriptional states, endocytotic state, infection with a viral agent, response to chemical agents, etc. Each of these concepts could capture an aspect of possible cell states and functioning and the combination of such attributes possessed by a particular cell could potentially be the basis for improved classifiers for certain clinically relevant endpoints. At first semantic features can be drawn from concepts thought to be relevant to cancer prognosis and treatment but eventually the range of features can be expanded in hopes of discovering previously unsuspected connections to cancer prognosis (and this same idea can be applied to other diseases and other classification tasks as well).

1.4.3 Novel Contribution of Semantic Attributes

Although mentioned in earlier sections, it is useful to summarize the some of novel contributions of the semantic attribute idea over previous techniques:

1. More accurate prediction of prognostic outcomes
2. Better interpretability of prognostic classifiers.
3. Possibility of discovering unique contributing factors to classification
4. Able to integrate wide range of external sources of knowledge.
5. Able to use high confidence knowledge.
6. Able to add knowledge incrementally.

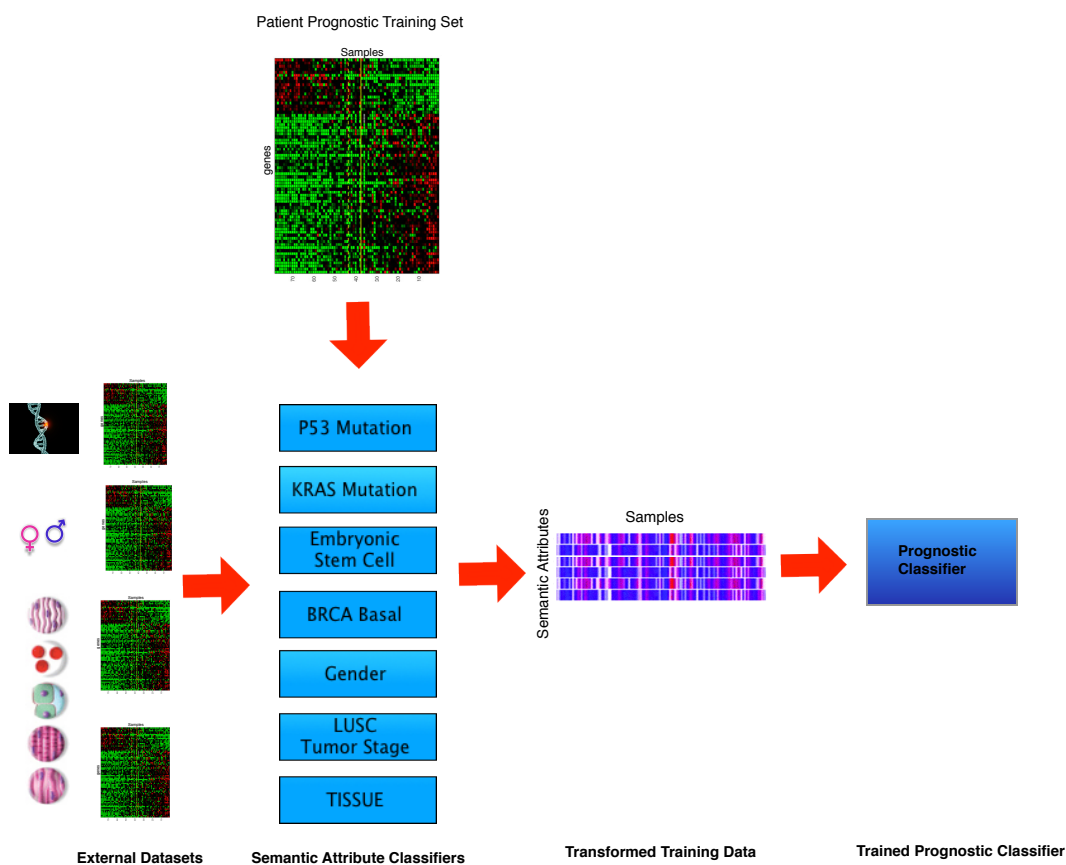


Figure 1.14: Semantic Attributes A wide variety of biological concepts can be used to create semantic attributes, everything from mutation status to gender to similarity to known subtypes in other cancer tissues. A wide range of external data can be tapped to train semantic classifiers. Patient prognostic data can be transformed from genes by samples dataset to a semantic attribute by samples dataset, and this transformed data can be used to train a prognostic classifier.

7. Able to leverage public data beyond experimental cohort.

More accurate predictions are expected to arise from the dimensionality reduction of projecting data onto knowledge based dimensions as well as the ability to separate out contextual factors in prognosis. Prediction is expected to be enhanced also by the ability to use a large amount of external data, many thousands of samples per semantic attribute, versus just the few hundred in a given study cohort. Incorporating a new concept into the semantic attribute repertoire is a simple matter of labeling samples according to their membership in the concept and training a classifier for it.

Given the finding that signatures based on individual genes are highly unstable this makes interpreting raw gene signatures quite problematic. Semantic attributes can improve on the interpretability gene based data by directly encoding a range of knowledge task. As will be discussed in the first set of results, a semantic attribute classifier may indicate that the most prognostic thing in lung cancer is progesteron receptor status, an interesting outcome on it's own independent of the improvement in classification accuracy.

Chapter 2

Aim 1: Tools

Building semantic attribute classifiers means building large numbers of classifiers from a wide variety of datasets. While there are many tools that make building single classifiers fairly turnkey, there are not so many tools that enable building classifiers in bulk and applying them in bulk to new data as required for semantic attributes. What is more, in practice there is no single best classifier algorithm for all classification tasks, nor a single best choice of hyper-parameters for any single algorithm. In fact, there are "no-free-lunch" theorems in learning theory which suggest that this is not merely a limitation in practice but a deep fact about what it means to generalize from seen examples to unseen examples[62][Whitley:2006vf]. Any such generalization necessarily taps into some sort of bias inherent in the algorithm to inform it how to make this "inductive leap". Sometimes this bias may be as simple as a bias for linear relationships, as with linear regression, or for quadratic or spline surfaces through a high dimensional vector space, or may involve assumptions about the compactness of the data. In practice, it turns out that even with the same kind of data, say RNASeq data, different target classes may best be captured by different kinds of classifiers. So in addition to needing to build and apply classifiers in bulk, building

semantic attribute classifiers also requires performing a search through a space of algorithms and hyperparameters to identify a reasonable approximation to the best classifier (the best is, of course, unknowable and would require an infeasible search through the space of options).

In order to support this effort I developed a series of tools to help with data wrangling, classifier model selection, building classifiers in bulk, and applying classifiers in bulk. Most of these tools were written in Java, both for performance (modern Java is very high performance, usually within a factor of 2X of C-language code and because of the ease of deploying Java applications both across platforms and in server settings.

Each of the following tools will be discussed in upcoming sections, with some fine details left for appendices:

- WekaMine a large scale model selection/application pipeline.
- Viewtab a Big Data Spreadsheet
- Csvsql a program to perform SQL queries on csv files.
- Grapnel a library of code for working with data in Java

2.1 WekaMine: large scale model selection, training, and application

In support of the other aims of this work, I have implemented a very mature suite of machine learning tools called WekaMine (Durbin [63]). WekaMine is based on the Weka (Hall et al. [64] and Frank et al. [65]) machine learning library and benefits from the wide range of algorithms provided by that library.

Weka is a mature library that implements a very wide variety of machine learning algorithms both new and old. While Weka itself is suitable for small scale experiments it does not have the functionality needed to evaluate algorithms on a large scale or to deploy trained models on a large scale. WekaMine aims to address the major issues in conducting repeatable machine learning experiments and also to address the often neglected aspect of actually building deployable models based on these experiments. Semantic attribute creation described in this work involves model selection, building, and deploying of many thousands of classifiers, so WekaMine was designed to operate on that scale, automating every step of a real-world ML pipeline as much as possible. While WekaMine was developed for bioinformatics machine learning pipeline creation, it is a general purpose machine learning framework suitable for any machine learning problem. WekaMine can be used as a stand alone bulk pipeline (`wekaMine`), or as a set of scripts that can be run individually (`wmFilter`, `wmSelectAttributes`, etc.) for different aspects of data mining. The aim of `wekaMine` is to make performing large scale machine learning model selection, including across a large computer cluster, as easy as performing a single training session with typical packages. `wekaMine` is also meant to be called as library functions from within any Java Virtual Machine (JVM) based language, such as Java, Groovy, Scala, or Jython code (e.g. for a web service or to implement complex meta-classifier algorithms). The major contributions of `wekaMine` are:

- Convenient command line tools
- Built-in compute cluster support.
- Flexible domain specific language (DSL) to describe experiments.
- Folds file support to generate repeatable validation folds.

- Additional discretizers (e.g. Quartile, Bimodality)
- Additional attribute selectors (e.g. Fisher Linear Discriminant)
- Additional classifiers (e.g. Balanced Random Forest)
- Fully encapsulated serialized models.
- Per-sample and per-feature score outputs.
- Automatic plots of experimental factors.
- Additional validation techniques (e.g. leave pair out)

2.1.1 Phases of Pipeline

There are three broad phases of a machine learning pipeline: model selection, model training, and model deployment. I use the term "model" instead of "classifier" to highlight that a deployable classifier necessarily includes normalization and feature selection (also known as dimensionality reduction), and possibly a discretization, as well as a trained classifier. "Model" is taken to be the combination of a normalization scheme, a subset of attributes, a discretization for numeric attributes, and a trained classifier. In some cases, a null background model for the classifier is also bundled as part of the model. An overview of the pipeline can be seen in Figure 2.1 and each phase is discussed in more detail below.

2.1.2 Model Selection

There is no single best algorithm for all prediction tasks. Among the several hundred different targets I have built classifiers for, each of Linear SVM,

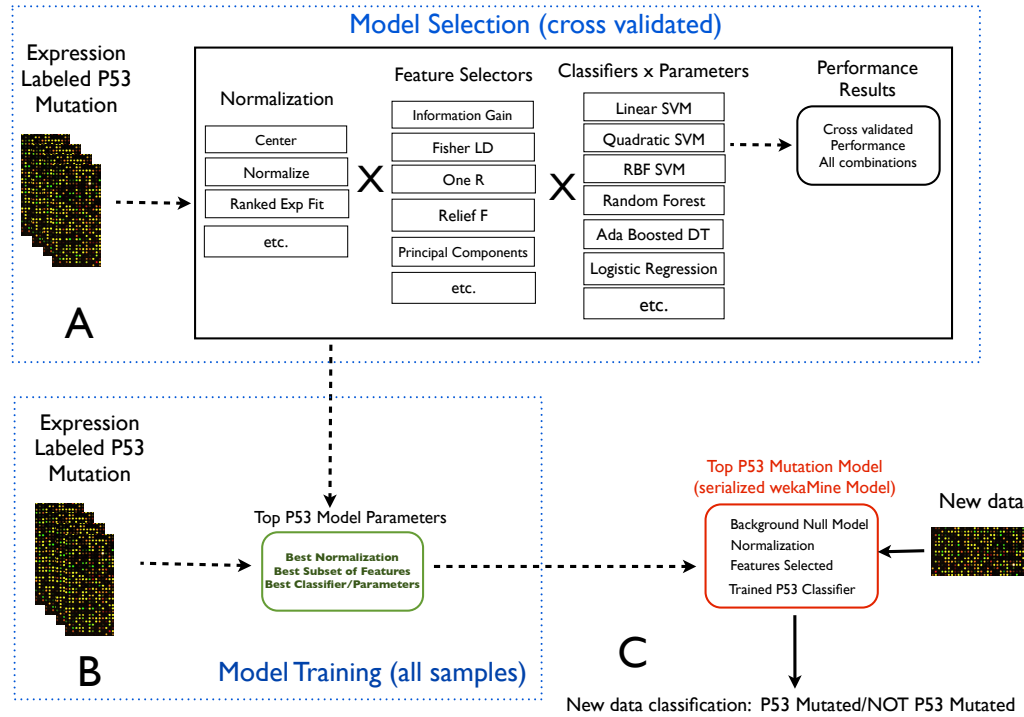


Figure 2.1: Building best classifiers for concept classes is a two stage process. The first stage (**A**) is model selection. In model selection a combination of a specific preprocessing step, attribute selection algorithm, classifier, and classifier parameters are used to train a model on the training portion of a stratified cross-validation fold. The performance of this combination is evaluated on a test fold and saved. This is repeated for each combination of preprocessing, attribute selection, classifier, and parameters. The best performing combination of steps for that particular target class can then be picked out. (**B**) This best combination of steps is used to train and save a serialized WekaMine model (using all available data) that can then be applied (**C**) to new data to obtain classifications for that target class.

Quadratic SVM, Radial Basis Function SVM, Logistic Regression, Balanced Random Forest, and Ada Boosted decision stumps have performed best for some subset of the target classes, both in terms of average performance and in terms of best performance (see Figure 2.2).

Moreover, almost every combination of this list of classifier algorithms and similar list of attribute selection algorithms, Information Gain, Relief F attribute selection, CFS Subset selection, perform best on some subset of problems. Partly this is due to the intrinsic shape of the decision boundary and the "inductive bias" (Haussler [66]) of the algorithms (the implicit shape of the hypothesis space entailed by a particular algorithm). Partly this is due to features of the dataset. Balanced Random Forest, for example, is a modification of the Random Forest algorithm to attempt to handle highly imbalanced datasets, that is data where there are many more examples of one class than there are examples of another class (Mohammed Khalilia [67]). Random Forest is an ensemble learning algorithm that combines a collection of decision trees via voting to arrive at a more robust generalization than with a single decision tree. Balanced Random Forest, which I have implemented for WekaMine, seeks to improve upon this by ensuring that the samples seen by each tree in the forest is presented with a balanced set of data, either by resampling the more rare examples or subsampling the more common samples. Given adequate data, some other algorithm might perform better than BRF for a given target class, but with limited and highly imbalanced data BRF sometimes produces the most robust (accurate and generalizable) predictors. Moreover, even within a given algorithm the performance of the classifier can often depend on the choice of parameters (e.g. the regularization parameter C in an SVM). As a result, it is necessary to perform a search over algorithms and parameters to determine the best choice for a given target class. This multi-dimensional

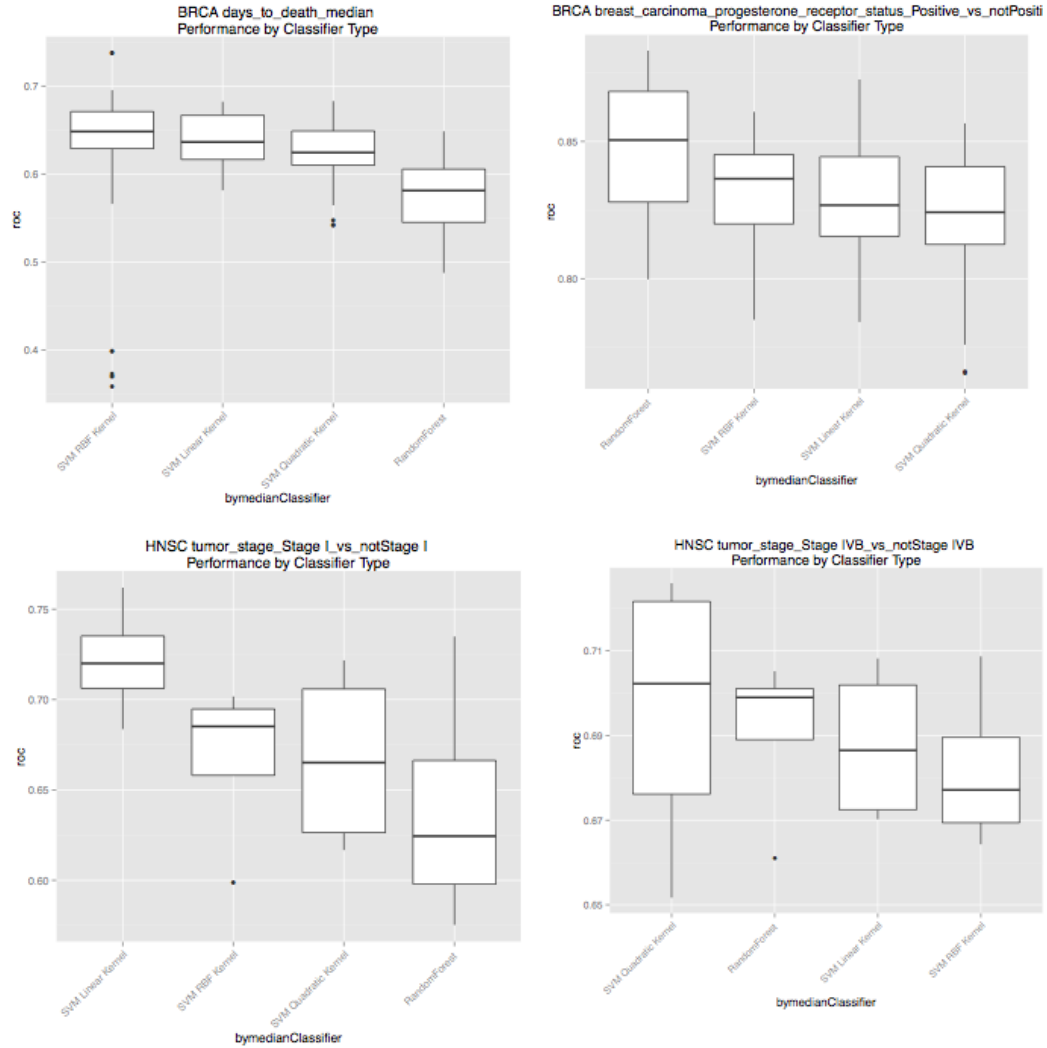


Figure 2.2: No Best Classifier Performance of four different classifier algorithms over four different semantic attributes. The distributions reflect different choices of parameters and different feature selections. Different classifiers perform best for different algorithms, and the differences in performance are stable across a variety of parameters for each algorithm suggesting that a particular classifier type is intrinsically better matched to a particular problem.

search is called model selection.

Cross Validation

Model selection requires that we train a set of algorithms on one set of data and evaluate the performance of that set of algorithms on data not seen by that training set. In many machine learning experiments, and especially in gene expression classification, the number of samples available is often quite small and so it is important to get the most out of this limited amount of data. In such situations it has been standard practice in machine learning research to perform cross-validated experiments. The goal is to get a performance estimate that is not too biased while making maximum use of limited data. In a cross-validated experiment the data is divided into k disjoint subsets, called folds. The algorithm is trained on $k-1$ of these subsets and its performance is evaluated by applying it to the k -th subset and recording its prediction performance on that subset. This procedure is then repeated k -times, each time with a different subset as the test set and the remaining data as the training subset. The performance of the algorithm is estimated as the average of the performance across these k -folds. In the ideal cross-validation experiment the folds will also be stratified, meaning that each fold contains approximately the same ratio of positive and negative examples (in binary classification; the same overall distribution in multi-label classification). WekaMine automatically handles generating stratified cross validation folds and saves a record of the folds chosen called a "fold set". A stratified fraction of samples is also automatically set aside by WekaMine and not used for any part of model selection or model training as a check against overfitting hyperparameters.

Model Performance

WekaMine uses the area under the receiver operator curve as the standard measure of performance, although the output produced by a model selection experiment is rich enough, including TP/FP/TN/FN, and even sample by sample classifications to compute any other measure after the fact. Accuracy can be a poor choice of performance measure because the accuracy can often vary with the choice of a cutoff that trades coverage for precision and accuracy can be made to seem higher than expected by silently accepting low coverage. Computing a receiver operator curve (ROC) gives a picture of this tradeoff and the area under the ROC (AUC) gives a point estimate of the performance that is less likely to give a false impression of the quality of an algorithm. AUC as a point measure is not without issues. It can be noisy itself and obscures somewhat the TPR/FPR tradeoff by summarizing in a single number. Of course, any point measure would be improved by estimating confidence intervals for the measure but in practice this can be challenging. More recently, Airola et al. [68] show that AUC may be a poor estimate of true predictor performance in low-sample high-feature settings. Unfortunately there is not much in terms of alternatives, as AUC still remains one of the best ways to evaluate performance with a point estimate. There is some suggestion in Airola et al. [68] that using bootstrapping and leave-pair-out CV may be slightly more robust. Leave-pair-out CV is currently being implemented in wekaMine so that another measure can be obtained.

Supervised Attribute Selection

Machine learning algorithms do not typically function alone but in conjunction with other data pre-processing steps. Data is often normalized in some way, so that all samples are in the same numerical space. Another pre-processing step

is typically some form of attribute selection. Some machine learning algorithms degrade in the presence of many irrelevant attributes and in the domain of gene expression classification the number of attributes is large and most are very noisy relative to the target. So typically some form of attribute selection or dimensionality reduction technique is employed before presenting the data to the classifier algorithm. Attribute selection falls into two categories: unsupervised and supervised. In unsupervised attribute selection some transformation is applied to the data that does not depend on the class being predicted. Unsupervised attribute selection includes, for example, techniques that remove all but one of a set of highly correlated attributes, or transforming the data into the k principal components that account for some percentage of the variance in the data. In unsupervised attribute selection it is possible to transform all of the data at one time, outside of the model-selection cross-validation experiment. Supervised attribute selection attempts to pick attributes that are known to be related to the output somehow. A simple but effective supervised attribute selection method is information gain. For each attribute the entropy of that attribute is computed relative to the target class. Expressed in bits this tells how many bits of information you learn about the output by looking at just that single attribute. Attributes can then be ranked based on this measure and the top k attributes selected to be passed onto the classifier. One of the most common mistakes in model selection is improperly applying a supervised attribute selection algorithm. It is necessary to apply the supervised attribute selection **only** to the training subset of data in a cross validation experiment. If one applies supervised attribute selection to all of the data at once, then information about the test set is effectively *leaking* through to the training set (Drier and Domany [24]). In effect, the classifier gains information about the test set which will make it seem like it generalizes better than it does.

This error is surprisingly common and can have a fairly significant positive bias in the performance estimate of an algorithm on a classification task. WekaMine automatically applies supervised attribute selection algorithms only to the training subset of a cross-validation experiment.

2.1.3 Model Training and Application

For many experiments model selection is all we are interested in. For example, we may wish to evaluate whether or not a novel normalization improves performance, or whether one kind of data is better than another (e.g. microarray vs RNAseq vs CNV) for predicting a target class. In this case model selection itself is the experiment, and the list of algorithms, parameters, and their respective cross validated performances output by WekaMine allows us to see how that performance varies with different algorithms or inputs. In other cases the purpose of model selection is to actually produce a model that can be applied to new data. In this instance, once the best combination of normalization, feature selection, classifiers, and parameters is determined for a particular target class, a classifier must be trained for application to new data. For this purpose we want to use all the training data available and the final classifier is trained from this data. Since the classifier is inherently tied to a particular normalization of the data, and a specific subset of the features, WekaMine bundles these together into a single serialized "WekaMine Model File" (.wmm file). This gives a single file that can be easily curated and which contains all of the information needed to apply this model to new data. A WekaMine script, `wmClassify`, can take this model file and a data file and automatically generate the subset of attributes needed for the model (the attributes must be in the same namespace, such as HUGO gene names, of course), apply the normalization needed, and apply the classifier to this feature reduced

normalized dataset to produce new classifications. Normalization itself can be dependent on the exact nature of the input data, the tissue, batch, platform, and so on, but I have deployed a novel normalization scheme that will be discussed in a later section to reduce this dependency as much as possible. These encapsulated models also help with the curation of very heterogeneous sets of models, using many different combinations of algorithms. In the semantic attribute application of this work, for example, I hadve used many hundreds of trained models and individual ones of these models may be built with an arbitrary combination of algorithms that is best for that target class. The WekaMineModel serialization presents a common abstract interface to any subsequent code so that this very heterogeneous collection of models can be applied as easily as a single algorithm applied repeatedly.

2.1.4 Experiment Design and Reproducibility

A justifiably growing concern in the bioinformatics community is the reproducibility of experiments. WekaMine addresses this reproducibility aspect in several ways. First, WekaMine experiments are specified using a simple domain specific language (DSL) that gives a compact description of all the experiments that will occur in a model selection experiment. This DSL makes it easy to specify complex multi-dimensional experiments in a very readable way. The details of this DLS are documented in the WekaMine appendix. Another factor in the experiment can be the exact folds chosen. WekaMine generates folds by generating a "folds file" that describes for each target attribute which samples are test samples in which fold. Folds are generated randomly, though stratified, but the "folds file" drives the actual pipeline and documents what folds were actually used. The folds file and the experiment DSL file and the data are all one

needs to exactly duplicate even complex WekaMine experiments. The serialized encapsulation of WekaMine models itself aids reproducibility by making it easy to share completely trained and ready to use models so others can deploy a model without any setup beyond installing WekaMine and downloading the model file. And finally, WekaMine is currently available as easy-to-install open source project on github: <https://github.com/jdurbin/wekaMine>, along with detailed documentation: <https://github.com/jdurbin/wekaMine/wiki>. At least six different users have downloaded and successfully installed wekaMine from the online documentation verifying it's comparative ease of use.

2.2 Viewtab: A Big Data Spreadsheet

Bioinformatics work often involves obtaining data from many different sources and integrating that data to produce some new analysis. The work described in this dissertation is a particularly acute example. In order to explore building classifiers for many different concepts it was necessary to obtain and explore many different data sets and meta-data files. Often these datasets come with minimal information. Faced with figuring out what is in a file, or figuring out which of several files might contain a particular piece of information that I need, the first step is often just finding some way to look inside the file. Most of the data files I have had to process for this dissertation came in the form of tab or comma delimited files. When such files are small they can be opened in commercial spreadsheets such as Excel or Numbers. These spreadsheet programs often have difficulty handling tables that are tens of thousands of rows by hundreds or thousands of columns though. Most simply can not open such large files, or if they can they become unusably sluggish and difficult. Some of these programs can not easily be launched from the command line, where bioinformatics workers

spend most of their time, which is an additional impediment as one has to launch the program and then navigate to the file in question from some arbitrary top directory. When examining lots of files, or examining intermediate files produced by ongoing pipelines, this can be a cumbersome step.

Bioinformaticians have a number of ways to approach this situation. The first is of course simply using "cat" to begin to get some idea what is in the file, but large files line wrap in ways that make a simple cat completely unreadable. This can be addressed by piping commands together with some obscure options like:

```
cat test.csv | column -s, -t | less -#2 -N -S
```

This at least presents the data in a column format and allows one to scroll sensibly through rows and columns. It's still very unweildy, though, and doesn't allow you to sort columns, get any idea what the range of values are in a column, easily move from top to middle and back in a file, and many other things one would like to do to explore a data file.

To address this problem I designed and create a custom spreadsheet program to meet the following requirements:

- Must be launchable from the command line.
- Must be launchable locally or remotely.
- Must handle files at least 50,000 x 5,000 in dimension.
- Must read file and display it very quickly (typically 10s or less).
- Responsiveness of GUI must be high even with large files.
- Should be able to sort columns
- Should be able to get a plot of the distribution of values in a row/column

- Should be able to see distribution of categorical values.
- Scatter plots of two columns easy.
- Correlation between columns.
- Plots can be formatted and saved to files.
- Columns re-orderable
- Rows searchable/restricted by keyword.

The resulting program, written in Java for performance and portability, is called "viewtab", and it can be invoked from the command line like:

```
viewtab test.csv
```

It takes about seven seconds for viewtab to display a 18,000 x 150 gene expression file on a 2014 MacBook Pro, 2.6Ghz, 16GB RAM machine. For comparison, it took Apple Numbers a full two minutes to display the file. Even then, the resulting spreadsheet was virtually unusable because each horizontal or vertical scroll resulted in display artifacts and dozens of seconds of additional delay. Loading a larger data file, say the entire set of CCLE expression data for 1037 cell lines, a 17,206 x 1037 table, viewtab opens and displays this file in 20 seconds (2.8x the time for a 6.6x larger file. Five seconds or so are fixed startup), after which it can be manipulated as easily as the 6.6x smaller file. Numbers, in contrast, attempts to open this file for a couple of minutes before failing.

An example of viewing a clinical data file can be seen in Figure 2.3.

From such a file one can select a pair of columns and get a scatter plot of the two columns, along with the correlation between the values in the columns, as in Figure 2.4. Here we see that the correlation between survival time and time to distant metastasis is strong in this dataset. This is extremely useful for getting

Features	Posnodes	TIMEsurvival	EVENTdeath	TIMErecur...	ESR1	StGallen	EVENTmeta	C1fromData	Prediction	NIH	Clused
VJV-14	0	8.233	0	8.233	1	1	0	0.726	1	1	0.77
VJV-13	0	8.167	0	8.167	1	0	0	0.41	0	0	0.28
VJV-12	0	7.858	0	7.858	0	0	0	-0.305	0	0	-0.3
VJV-11	0	5.804	0	5.804	1	0	0	0.549	0	0	0.37
VJV-396	1	10.231	0	10.231	0	0	0	0.028	0	0	0.028
VJV-257	1	2.587	1	1.717	0	0	1	0.467	1	0	0.467
VJV-395	1	15.077	0	11.211	1	0	1	0.519	1	0	0.519
VJV-256	1	9.506	0	8.988	1	0	1	0.459	1	0	0.459
VJV-398	0	8.424	0	8.424	0	0	0	-0.16	0	0	-0.32
VJV-259	0	8.964	0	5.517	1	1	1	0.453	1	0	0.453
VJV-397	0	8.772	1	4.767	1	0	1	0.203	0	0	0.203
VJV-258	1	5.353	1	5.117	1	0	1	0.195	0	0	0.195
VJV-392	1	6.171	0	6.171	1	0	0	-0.021	0	0	-0.021
VJV-391	0	6.021	0	3.51	1	0	0	0.693	1	0	0.693
VJV-394	0	5.708	0	5.708	1	0	0	0.675	1	0	0.675
VJV-393	1	5.574	0	5.574	1	0	0	-0.099	0	0	-0.099
VJV-9	0	10.294	0	10.294	1	0	0	0.651	1	0	0.67
VJV-8	0	8.802	0	8.802	0	0	0	-0.073	0	0	-0.09
VJV-251	0	9.407	0	9.407	1	0	0	0.084	0	0	0.31
VJV-7	0	10.138	0	10.138	0	0	0	0.815	1	0	0.79
VJV-250	0	11.365	0	5.569	1	1	0	0.67	1	0	0.77
VJV-6	0	11.157	0	11.157	1	0	0	0.772	1	0	0.69
VJV-252	0	9.916	0	9.123	1	0	1	0.222	0	0	-0.01
VJV-4	0	12.997	0	12.997	1	0	0	0.261	0	0	0.33
VJV-17	0	7.866	0	7.866	1	0	0	0.828	1	0	0.79

Figure 2.3: viewtab view of clinical data file from [11]

a quick initial view of what variables may be related to one another in a dataset. Within the scatter plots it's possible to select a region and zoom in on that region, as illustrated in Figure 2.5.

The distribution of numeric values can be obtained for a selected row or column as in Figure 2.6. In the case of non-numeric data the distribution shows the distribution of counts for each factor value, as in Figure 2.7. In addition to being useful in final data analysis, the ability to quickly examine tables of data like this is a huge benefit to the initial stages of data wrangling, determining the ranges of values to expect from different columns of meta-data and determining which meta-data columns contain the kind of data you are looking for.

Since bioinformatics data files are very large, it can be difficult to find a particular sample or subset of samples within the file by visual inspection. Also, sometimes one is interested in plotting values for some subset. To facilitate this, viewtab has a built-in text filter that will filter rows to include only rows that contain the text typed. For example, one can restrict the view of data in a LUSC

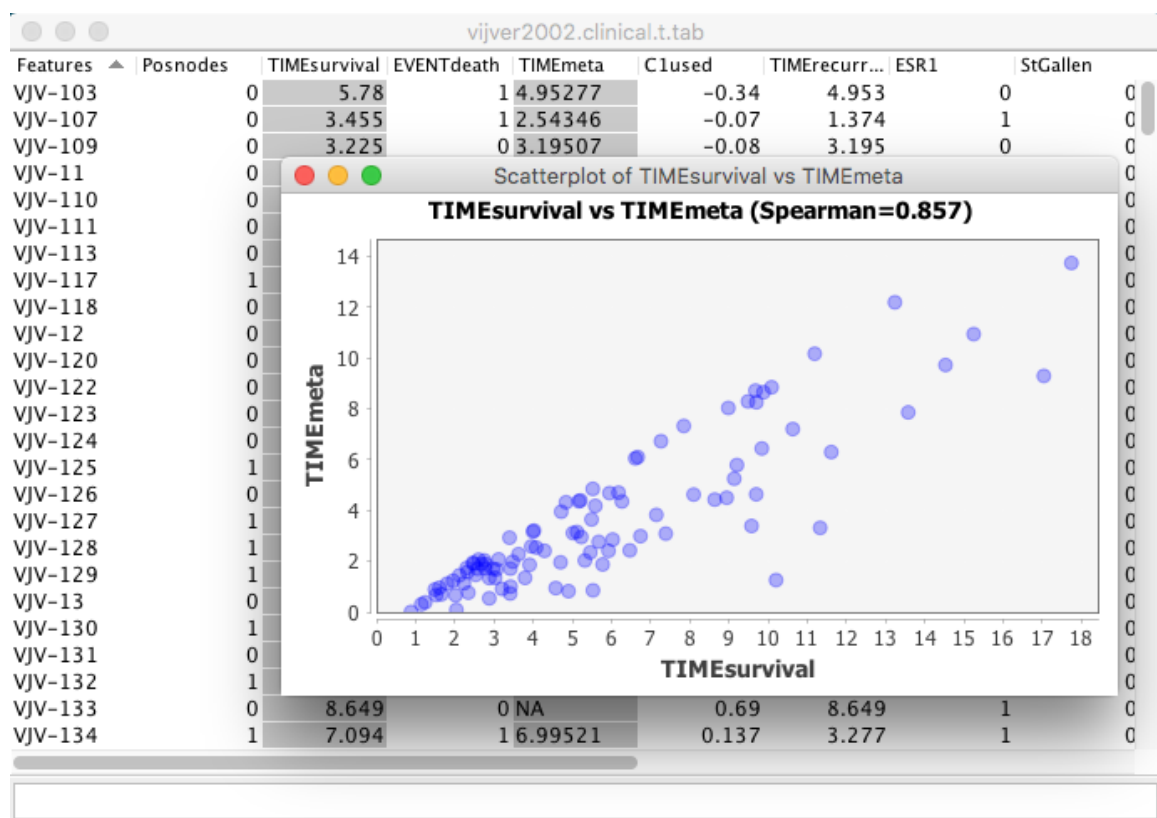


Figure 2.4: viewtab view of clinical data file from [11] with scatter plot of two variables.

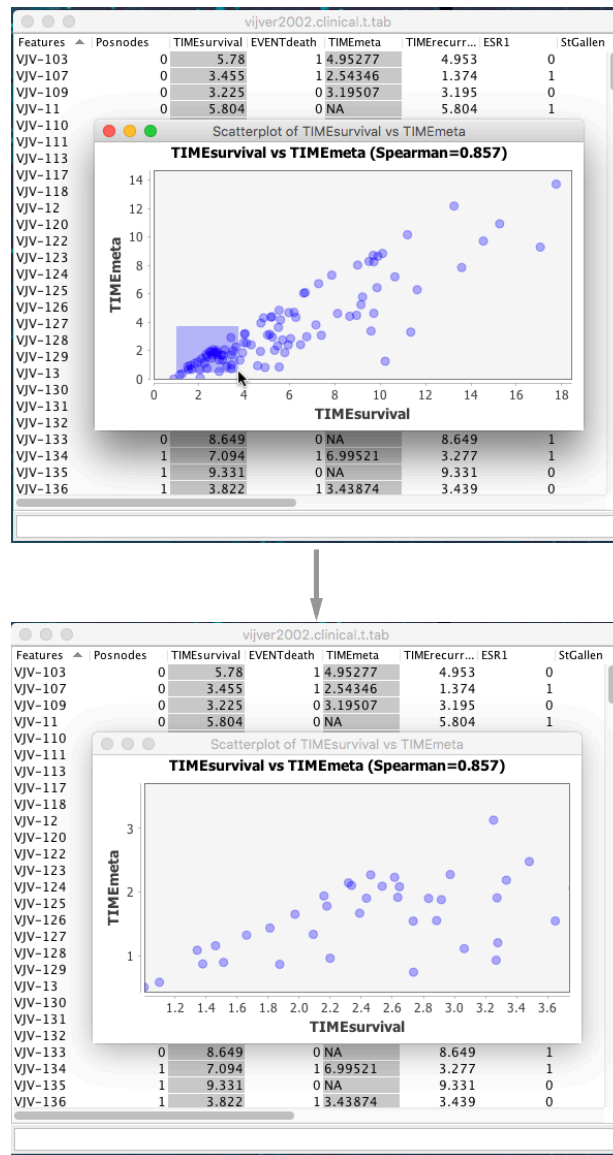


Figure 2.5: viewtab view of clinical data showing zoom of points in scatter plot.

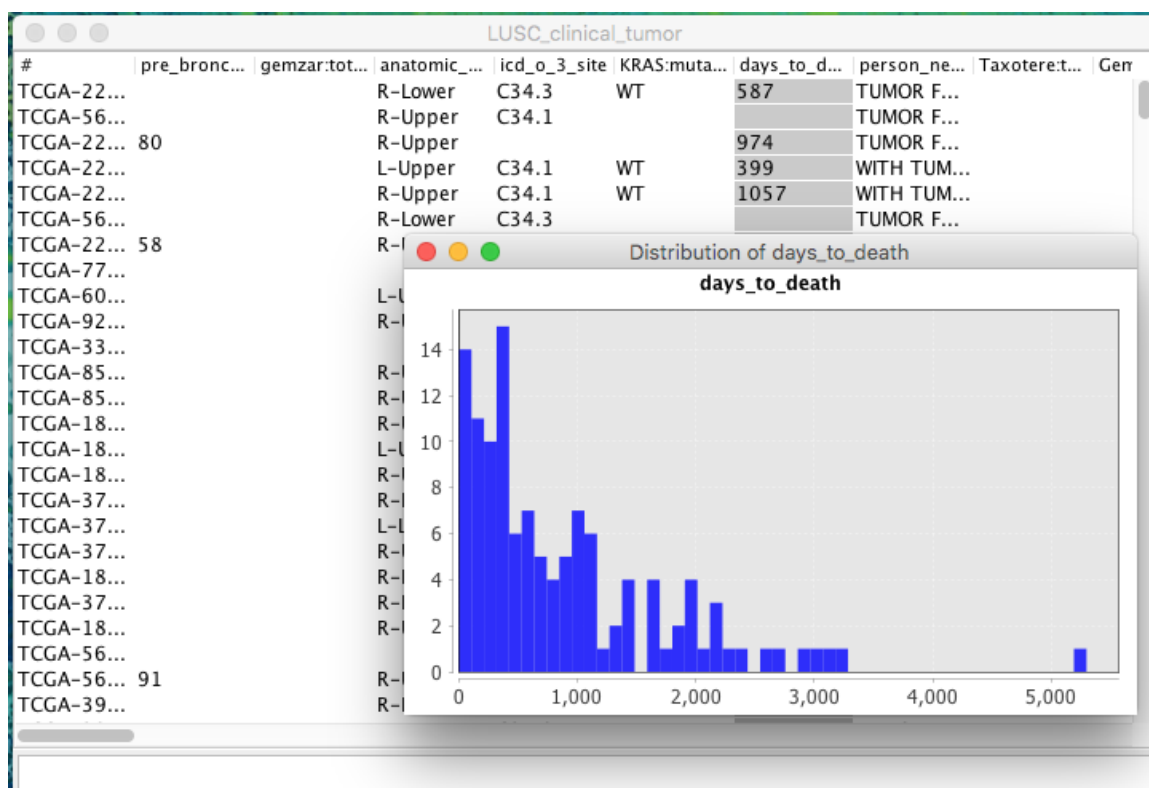


Figure 2.6: viewtab view of clinical data file from [TCGAPANCAN] showing distribution of days to death in LUSC metadata.

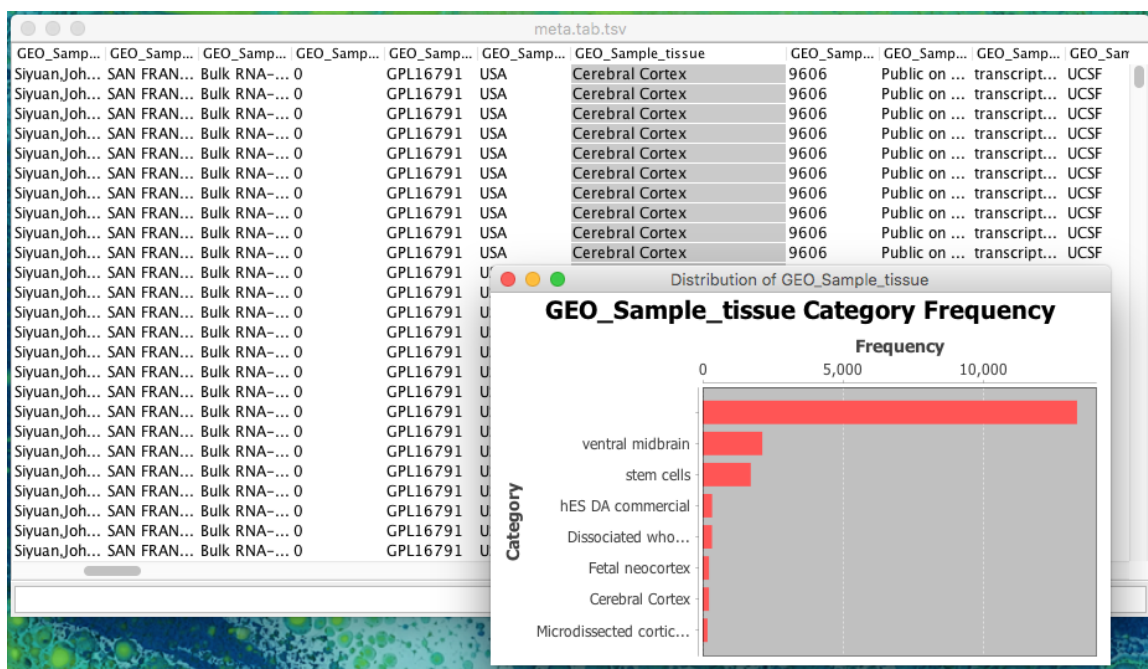


Figure 2.7: viewtab view of single cell data showing distribution of tissue types.

clinical file by, for example, typing "Upper" into the filter entry box. viewtab will then show only those rows which contain the text "Upper", as in Figure 2.8. Any plotting that is performed is performed only on the values that pass the filter, so if one plots days-to-death while the "Upper" filter is in place, only samples from the upper part of the lung will appear in the plot.

Taking these features together, viewtab stands as a significant contribution of this work.

2.3 Csvsql: Treating CSV Files as Databases

Viewing data in a spreadsheet is one way to explore and manipulate tab delimited data files. Often there is a need to extract certain subsets of information out of a data file, or examine the data file in various quantitative ways, to ask question such as, "How many samples from the upper quadrant of the lung also

LUSC_clinical_tumor									
#	pre_bronc...	gemzar:tot...	anatomic_...	icd_o_3_site	KRAS:muta...	days_t...	person_ne...	Taxotere:t...	C
TCGA-34...			L-Upper	C34.1	WT	999	TUMOR F...		
TCGA-22...	80		R-Upper	C34.1		974	TUMOR F...		
TCGA-21...			L-Upper	C34.1		964	WITH TUM...		
TCGA-22...			R-Upper	C34.1	WT	915	WITH TUM...		
TCGA-34...			R-Upper	C34.1		881	WITH TUM...		
TCGA-52...			L-Upper	C34.9		835	WITH TUM...		
TCGA-22...			R-Upper	C34.1	WT	822	TUMOR F...		
TCGA-77...			L-Upper	C34.1		803	WITH TUM...		
TCGA-77...			L-Upper	C34.1		632			
TCGA-33...			R-Upper	C34.1	WT	601	WITH TUM...		
TCGA-22...			R-Upper	C34.1		573	WITH TUM...		
TCGA-39...			R-Upper	C34.1	WT	544	TUMOR F...		
TCGA-77...			L-Upper	C34.1		538	WITH TUM...		
TCGA-22...			R-Upper	C34.1	WT	53	TUMOR F...		
TCGA-39...			R-Upper	C34.1	WT	52			
TCGA-22...			L-Upper	C34.1	WT	505			
TCGA-94...	71		L-Upper	C34.1		5			
TCGA-77...			R-Upper	C34.1		5			
TCGA-22...			L-Upper	C34.1	WT	444	TUMOR F...		
TCGA-22...			L-Upper	C34.1	WT	399	WITH TUM...		
TCGA-18...			L-Upper	C34.1	WT	371	WITH TUM...		
TCGA-22...			L-Upper	C34.1	WT	358			
TCGA-22...			L-Upper	C34.1	WT	357	TUMOR F...		
TCGA-21...			L-Upper	C34.1	WT	329	WITH TUM...		
TCGA-77...			L-Upper			3253			
TCGA-46...			L-Upper	C34.1	WT	303	TUMOR F...		

Figure 2.8: viewtab view of clinical data filtered by keyword "Upper"

have a valid days-to-death value?", or to extract all of the rows out of one file that match a key or search term in a separate file.

In many applications data that needs to be queried and manipulated in these ways is stored in a database, such as commercially available Oracle database or the open source MySQL. Both of these databases can then be queried using a special query language called SQL (Structured Query Language). Often we would like to perform exactly the kinds of queries that SQL allows but without the overhead of creating a database, a task that typically involves considerable setup for each table of data one wishes to store.

To bridge this gap I wrote software called csvsql which allows ordinary tab or comma separated files to be queried as if they were tables in a database. csvsql achieves this by examining the tab files and creating a database in RAM on-the-fly using the Java based h2 database engine. With csvsql it's possible to perform full SQL queries on csv table files, even including multi-file joins.

With csvsql, any query that is valid sql for the h2 database engine is available to be used on csv or tab files. The use of csvsql is most easily illustrated by examples. Here are some common kinds of SQL queries and how they would be written with csvsql to operate on csv files:

```
csvsql "select score from people.csv where age < 40"
csvsql "select name,score from people.csv where age <50 and score > 100"
csvsql "select sum(score) from people.csv where age < 40"
csvsql "select people.name,children.child
        from people.csv,children.csv
        where people.name=children.name"
```

When a selection is performed, the output is provided in the same format as the input file. So in the case of selecting name and score from people.csv, the output will be a csv formatted table with name and score as headings. By piping the output of csvsql to another file, csvsql enables the extraction of a subset of

data from one to create a new file, like these examples:

```
csvsql "select name,score from people.csv where score > 80" > over80.csv
csvsql "select * from LUSC_meta.tab where anatomic_site like '%Upper%'" > upper.csv
```

When referencing csv files, full path names work fine:

```
csvsql "select * from /users/data/people.csv where age > 40"
csvsql "select people.name from /users/data/people.csv where age > 40"
csvsql "select name from /users/data/people.csv where age > 40"
csvsql "select name,age from /users/data/people.csv where age > 40"
```

It's also possible to do queries with sum and average and group by and other SQL manipulations like:

```
csvsql "select sum(score) from people.csv where age < 40"
csvsql "select alcohol,count(*) from beer.tab group by alcohol"
```

If children.csv is a file with same key name as people, it is possible to do a multi-file join query like:

```
csvsql "select people.name,children.child from people.csv,
children.csv where people.name=children.name"
```

You can also enter the query on multiple lines like:

```
csvsql "
> select people.name,children.child
> from people.csv,children.csv
> where people.name=children.name and people.age < 40"
```

Get a distribution of values in the file like:

```
csvsql "select alcohol,count(*) from beer.tab group by alcohol"
```

rename an output column:

```
csvsql "select brand,alcohol,price as SalesPrice from beer.tab order by price"
```

alias names for convenience:

```
csvsql "select p.name,c.child from people.csv as p, children.csv as c where p.name=c.name"
```

And other standard SQL commands like:

```
csvsql "select brand,count(*) from beer.tab group by brand having count(*) > 2"
csvsql "select * from beer.tab where brand like 'Red%'"
csvsql "select distinct(type) from beer.tab"
```

Files can be either tab delimited or comma delimited and must end in one of .csv, .tsv, or .tab. It is necessary refer to the file by file name at least once in the query, but thereafter it can be referred to either with the file name or the root name (e.g. beer for beer.tab).

The main reason for writing csvsql is to work with fairly large data and meta-data files, so performance is important. It is common with gene expression and sample meta-data files to process csv files containing between 100k and 1M records, with usually a few hundred columns. csvsql can perform a select on a file 170k rows in length and output 16k results to a file in 3 seconds on a 2015 MacBook Pro.

Being able to treat every tab or csv file one encounters as though it were already a populated database is a tremendous aid to data wrangling tasks and is another significant tool contribution of this work.

2.4 Grapnel: Java library

Grapnel is an open source Java library written for this work. It's named after a grappling hook, which is a tool that allows a person to grab onto something at a distance. Grapnel provides the core library functions used by viewtab, csvsql, and wekaMine.

Grapnel includes:

- `grapnel.charts`: Support for common kinds of charts: line chart, xyplot, hist. Based on JFreeChart but includes lots of sugar to make it easier to make commonly used charts and support for saving them in various formats. Also

has support for creating a chart and displaying it in a GUI with a single command.

- `grapnel.stat`: Statistical classes.
 - `MixtureModel` Class to compute parameters of a mixture model given data (based on SSJ), and to classify data into most-likely mixtures.
 - `KolmogorovSmirnov` class to compute KolmogorovSmirnov statistics from data.
 - `QuantileNormalization` classes to perform quantile normalization.
 - `Sampling Wrappers` to simplifying sampling from lists.
- `grapnel.util`: Core functionality of `grapnel`.
 - `DoubleTable` Implements a high-performance 2D table of doubles accessible by index or name. Backed by `[colt]` (<http://acs.lbl.gov/software/colt/>) `DenseDoubleMatrix`. Includes syntatic sugar to allow `[]` notation, `eachRow` closures, etc. from Groovy and functionality to read/write tables to files in a fairly high performance way.
 - `Table` Implements a high-performance 2D object table. Same functionality as `DoubleTable` generalized to a table of objects. Not as efficient as `DoubleTable` for numeric data, but still fairly efficient. `DynamicTable` Implements a dynamically allocatable 2D table (a 2D Map, essentially). Row and column keys can be any comparable object. Backed by Google `HashBasedTable` in guava Good performance with lots of Groovy syntax sugar. Read/write to file functionality.
 - `MultidimensionalMap` for the creation of HashMaps of arbitrary dimensions.

- CounterMap Map that counts unique occurrences of keys.
 - FileUtils Number of file utilities, fastCountLines, determineSeparator, etc.
 - ImageUtils Utilities for saving AWT/Swing components as JPG/PNG/GIF images.
 - OnlineTable Class to allow access to a tabular data file one-row-at-a-time by column name.
 - Parser Class to parse options. Fork of Groovy Option Parser.
 - SSHPortForwarder Class to do port forwarding from within Java/-Groovy (say, for example, to access a database behind a firewall). Based on jsch.
- grapnel.weka Additional algorithms for Weka
 - BalancedRandomForest An implementation of a balanced random forest algorithm.
 - ExponentialNormalizationFilter Normalize data by ranking and fitting to exponential distribution.
 - FishersLDEval Attribute ranking based on Fisher's linear discriminant.
 - BimodalityIndexFilter Model attributes as a mixture model, replace each value with a bimodality index based on that model.

Chapter 3

Aim 2: Semantic Attribute Results

3.0.1 Building Models for UMLS Terms in GEO

One possible benefit of the semantic attribute method is if the generation of semantic attributes can be automated so that bringing in diverse semantic attributes opens up the possibility of discovering novel and unexpected connections. I explored building semantic attributes by semi-automated mining the National Center for Biotechnology Information (NCBI) Gene Expression Omnibus (GEO, [69][70]). In addition to providing a diverse set of semantic attribute models, there is the potential to greatly expand the amount of data available to train each model. While any given experiment in the database is small, across many such experiments the number of samples available to train a model can be substantial.

Unfortunately, the data is not stored in a way that makes pooling data into large experiment/control groups particularly easy. There is, for example, no universal tag in the data for cancer samples versus non-cancer samples. Rather, the experimental and control data sets are distinguished only by free-text entered by

researchers who most often use very specific terms to indicate the nature of the data (e.g. retinal blastoma). Thus a central challenge of building semantic models from GEO datasets is identifying relevant pools of gene expression data.

3.0.2 Pooling From UMLS Term Hierarchies

UMLS Terms

The National Library of Medicine (NLM, part of the National Institutes of Health) has created a set of resources known as the Unified Medical Language System. UMLS is a set of lexical tools to help in the retrieval and interpretation of medical information from unstructured sources. This includes an extensive structured database of biomedical terms and their relationships (e.g. retinal-blastoma is-a-type-of cancer), as well as text-mining tools to read in free text and derive relevant terms from that free text ([71]). The UMLS database and tools can be used as the basis for pooling microarray experiments.

Pooling with UMLS

The strategy for data pooling was to use the UMLS text mining tools to create lists of terms for each experiment downloaded from the GEO database, then to use the term relationships to identify common parent terms across several experiments. For example, parent terms for "retinal blastoma" might be "eye", "eye diseases", "eye cancer", and "cancer". Each of these parent terms thus picks out a subset of all of the microarray data that can be used to create positive and negative examples of the term. We could, for example, pool all of the data with parent concept "eye" in one set of positive examples, and a similar sized pool of data without "eye" as the parent concept as the set of negative examples, and use these two pools in order order to try to create a classifier for the expression

patterns found in eye tissue versus other tissues. Similarly to look at cancer tissue versus non-cancer tissue, and so on. This pooling based on UMLS hierarchy terms will both allow one to pool data from various researchers for the same condition, as well as to slice and repool the data in ways not intended or envisioned by those who originally produced the data (e.g. the retinal blastoma dataset was produced to compare normal eye and retinal blastoma eye expression patterns, not to distinguish eye tissue expression patterns from heart tissue expression patterns, though it will necessarily contain information useful for the latter classification as well).

MMTx Term Generation

A substantial fraction of the GEO database was downloaded and processed by running the MetaMap Transfer program ([72]) on the free text descriptions of all of the microarray experiments to extract out UMLS standard lexical terms. Typically about five hundred lines of output including about forty unique terms are generated per experiment. An example of the original GEO description can be seen in Figure 3.1 and a sample of the MMTx terms in Figure 3.2.

```
DATABASE   Geo
DATASET    GDS1062
title Squamous cell carcinoma of the oral cavity with lymph node metastasis
description Expression profiling of 18 primary squamous cell carcinomas of the oral
cavity with or without lymph node metastasis. Expression in lymph node metastasis also
examined. Results used to identify gene expression signature in primary tumors that
indicates the presence of lymph node metastasis.
type       gene expression array-based
pubmed_id 15558013
platform   GPL96
platform_organism Homo sapiens
```

Figure 3.1: Original free-text description in GEO submission

title	1000	C0585362	squamous cell carcinoma of oral cavity	squamous cell carcinoma of mouth
title	909	C0686619	lymph node metastasis	secondary malignant neoplasm of lymph node
description	687	C0585362	squamous cell carcinoma of oral cavity	
description	909	C0686619	lymph node metastasis	
description	1000	C0024264	lymph	lymphocyte
description	1000	C0011334	cavity	<u>nos</u>
description	1000	C0185117	expression	expression procedure

Figure 3.2: MMTx terms derived from free-text

UMLS Term Hierarchies

The UMLS terms themselves do not contain all the information needed in order to pool data. From "retinal blastoma" we want to discover that this is a kind of cancer, what tissue it is from, etc. I have implemented software to query the UMLS database with a concept identifier from the MMTx output and perform a bi-directional depth first search to identify all of the parents and children of the term in the database. Since the queries take a non-trivial amount of time, and many terms have the same parents or children, nodes are cached so that the expensive remote lookup of any given node only occurs once. Two examples of such hierarchies can be seen in Figure 3.3 and Figure 3.4. Examining these hierarchies brought to light the unanticipated problem of cycles in the "hierarchy". Most often these seem to be related to terms that are not elsewhere classified (NEC), so I adopted the heuristic strategy of automatically breaking these links whenever they were encountered. Also apparent is that while there are parent terms that correspond to broader conditions that we could pool from the datasets, these are still quite technical and may not directly overlap with related terms found just by exploring the parent/child hierarchy. Because of these and similar issues, the overall process ends up requiring a fair amount of human input to resolve ambiguities.

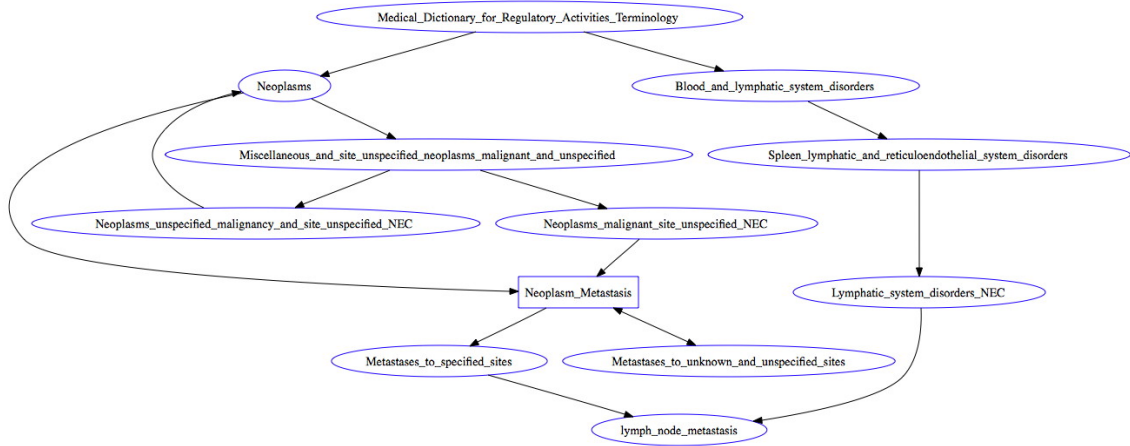


Figure 3.3: Parent child hierarchy for Neoplasm Metastasis

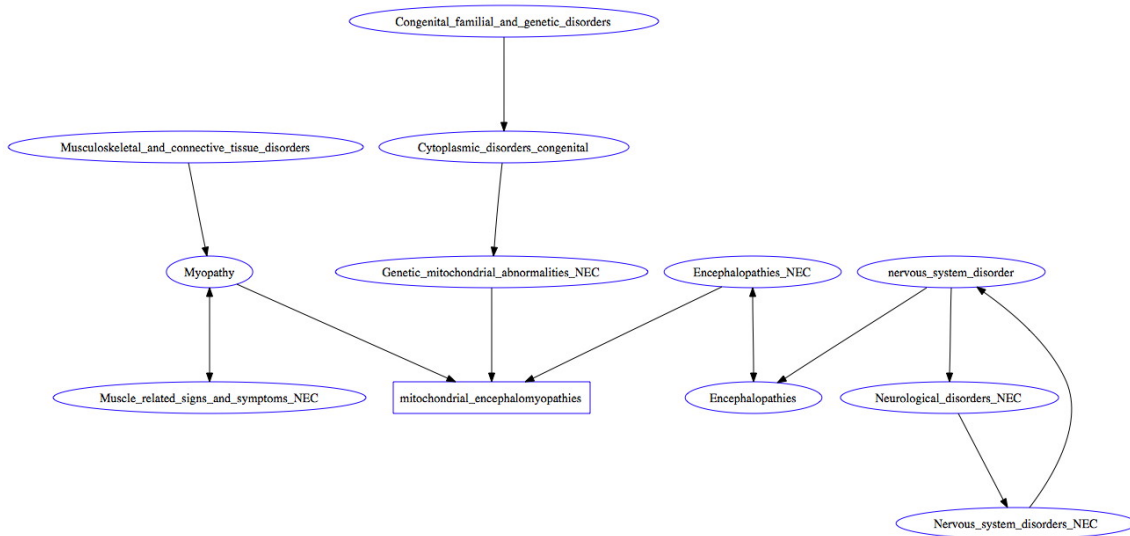


Figure 3.4: Parent child hierarchy for mitochondrial encephalomyopathies

3.0.3 Basic Test Building UMLS Based Classifier

As a test of the pipeline, I used UMLS parent terms extracted from MMTX labels to simply identify "cancer" vs "normal" samples. In this test I identified eighty eight cancer related experiments by looking up more specific terms in the hierarchy. These experiments together cover 19218 different genes (attributes), although any one experiment often only has 5000-8000 genes. WekaMine automatically handles merging datasets with different overlapping subsets of features, so these are easily combined into one large training set of "cancer" samples. A similar sized dataset of randomly chosen samples from GEO without the parent term "cancer" in the hierarchy were assembled as the contrast set. Using this dataset the best WekaMine model had an ROC of 0.963 at identifying a sample as a "cancer" sample. A similar small scale test looked for "lung" tissue samples. I identified 24 data sets comprising a total of 20409 genes. The same sized not-lung pool was created from an additional 125 data sets automatically extracted. The WekaMine best model ROC for a "lung" vs "not-lung" semantic attribute classifier was 0.973.

3.0.4 UMLS Compendium Conclusion

Although these early tests seemed promising overall the task of curating where and how to break the graph of terms proved to require a lot of manual intervention. I explored various ways to automate this task more, including using more advanced text mining tools than MMTx, but ultimately decided that this approach expanded the scope of the thesis too much and subsequently focus on semantic attribute datasets that are pre-compiled and curated by external resources. Returning to UMLS term mining of large expression databases is an interesting problem that will be left to future work.

Chapter 4

Survival Prediction

4.1 Survival Prediction With Semantic Attributes

As an initial test of the semantic attribute idea in practice I looked at the task of survival prediction in four of the The Cancer Genome Atlas (TCGA) cohorts: glioblastoma (GBM), ovarian serous cystadenocarcinoma (OV), kidney renal clear cell carcinoma (KIRC), and lung squamous cell carcinoma (LUSC), a total of 634 samples. For the semantic attributes I used a combination of mutation attributes, clinical attributes, and mutual exclusivity module events identified by the MEMo method (Ciriello et al. [44]). These attributes were trained using TCGA RNASeq data for all eleven of the pancan tumor types.

4.1.1 Mutation Attribute Classifiers

Mutations are a potentially useful semantic attribute since they represent a stable aspect of the genotype that may inform expression patterns in many different phenotypes. A particular mutation may be present in many different kinds of tissues and their effects may persist across many different cell states. As a result,

a large amount of data may be available to train classifiers for many mutations and, subsequently, identifying which mutations are active in a cell state may help categorize the phenotypic meaning of observed expression patterns. Using mutation data compiled by the TCGA project (Synapse entity ID: syn1710680) I extracted all of the non-synonymous mutation calls for eleven pancancer tumor types. Samples with mutation rate > 12 per 10^6 bases were considered 'hypermutated' phenotypes and were omitted from the training set for individual mutations (per [73]). Mutations with less than 20 examples were omitted as having too few examples for training. This left 99 mutations across eleven tissues for which WekaMine model selection was performed. Of these, 33 produced cross-validated accuracies that exceeded the majority classifier accuracy (e.g. if you just always guessed the most common mutation category). Some mutations are very common in some tissues (e.g. FBXW7 has non-silent mutations in 92% of BLCA samples) so beating the majority can for many mutations be a very high bar. The performance of the top 20 tissue-specific mutations can be seen in Figure 4.1.

One concern with building semantic models in general is the question of whether it is possible to train a model in one dataset and apply it meaningfully in another dataset. As described in the appendix, one aspect of this is our novel exponential normalization to normalize samples against batch and tissue specific effect. In the mutation context I evaluated the cross-tissue applicability of these models by performing a variety of cross-tissue classifier experiments. In these experiments I trained the mutations on $N-1$ tissues and tested the classifier on a hold-out tissue. I then repeated this for N tissues in a cross-validated way. Figure 4.2 illustrates the results of one such experiment.

Mutation	roc	majority accuracy	deltaroc
LAML_NPM1_NonSilent	0.984	0.537	0.447
BRCA_TP53_NonSilent	0.912	0.549	0.363
BLCA_TP53_NonSilent	0.867	0.508	0.359
BRCA_PIK3CA_NonSilent	0.857	0.521	0.336
UCEC_PTEN_NonSilent	0.863	0.541	0.322
UCEC_CTNNB1_NonSilent	0.977	0.678	0.299
LUAD_TP53_NonSilent	0.871	0.593	0.278
COADREAD_TP53_NonSilent	0.835	0.557	0.277
GBM_EGFR_NonSilent	0.858	0.607	0.25
COADREAD_KRAS_NonSilent	0.807	0.566	0.241
KIRC_PBRM1_NonSilent	0.831	0.591	0.239
UCEC_TP53_NonSilent	0.938	0.7	0.238
LUAD_KRAS_NonSilent	0.848	0.652	0.196
GBM_TP53_NonSilent	0.777	0.615	0.161
LUAD_KEAP1_NonSilent	0.939	0.793	0.146
LAML_DNMT3A_NonSilent	0.752	0.613	0.139
LAML_RUNX1_NonSilent	0.959	0.853	0.106
BLCA_NFE2L2_NonSilent	0.983	0.892	0.09
BLCA_ARID1A_NonSilent	0.777	0.688	0.09

Figure 4.1: Top 20 Mutation classifiers, showing the roc area-under the curve, the majority accuracy that would result from guessing the most common mutation, and the delta between the majority accuracy and the roc.

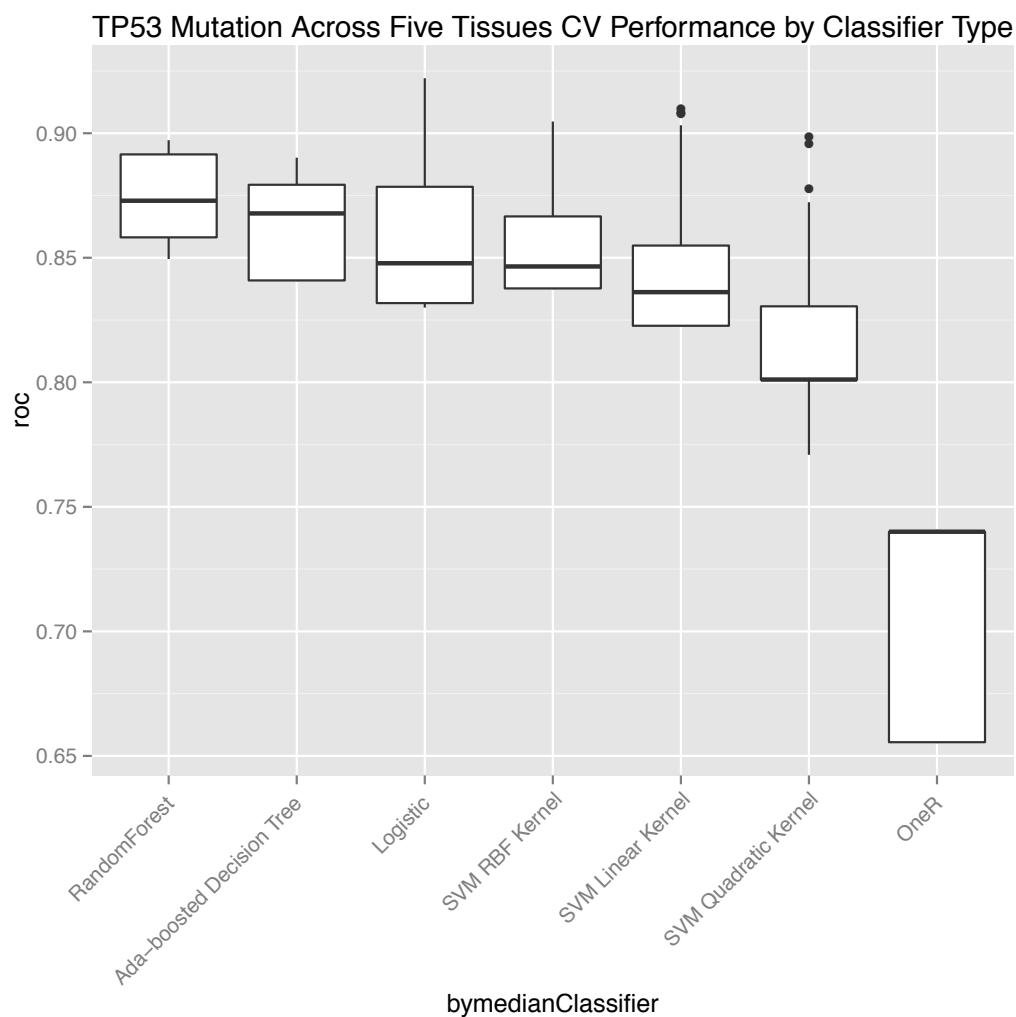


Figure 4.2: For this experiment I trained classifiers on four tissues, tested on the fifth and repeated this process five times to assess performance across tissues. Across a range of algorithms cross-tissue mutation classifiers manage to perform almost as well as within tissue classifiers.

4.1.2 Clinical Attribute Classifiers

Clinical information provided with each tumor type were passed through a dichotimization filter (`wmDichotomize`) to obtain dichotomized set of clinical attributes. `wmDichotomize` will take a k-variabed attribute and convert it to k binary attributes (e.g. `TumorStage = A,B,C` becomes three binary variables, `TumorStageA_vs_NotA`, `TumorStageB_vs_NotB`, `TumorStageC_vs_notC`). It also transforms numeric variables into three binary variables, one each corresponding to values above/below the upper quartile, above/below median, and above/below lower quartile. In all there were 142 dichotomized clinical attributes. Out of these, survival time and trivial attributes were removed leaving 109 clinical variables. Of these, 36 produced classifiers in cross-validated model selection that beat the majority classifier.

4.1.3 MEMo Event Attribute Classifiers

The MEMo (Mutual Exclusivity Modules, Ciriello et al. [44]) method uses correlation analysis to identify modules of genes that are suspected to be in the same biological process and for which alteration events within the modules are mutually exclusive (e.g. PI3K, P53, and Rb alterations seem to be mutually exclusive in GBM). The events produced from this analysis are either amplifications, deletions, or mutations of genes, groups of genes, or physical locations on the genome (e.g. amplification of the set: LOC100288778, B4GALNT3, CCDC77, KDM5A, NINJ2, SLC6A13, IQSEC3, WNK1). MEMo analysis was run on the eleven pancancer tumor types and produced a total of 1295 events across these eleven tumors. Using again a minimum cutoff of 20 samples per event in order to have a potentially trainable sample this was reduced to 171 events on which WekaMine model selection was performed. Out of these, 52 produced models with

cross-validated ROC that beats the majority classifier for that event.

4.1.4 All models

In all, 121 semantic attribute models were created each of which performed better than majority classifier for that attribute’s class. Models with $< 1\%$ improvement over majority were eliminated, leaving 117 models. The full list of models, their cross validated roc, and their improvement over majority classifier can be found in Appendix C

4.1.5 Semantic Transform Results

As a comparison set, I performed WekaMine model selection experiments on mRNA data from RNASeq for four different tumor types. The same WekaMine model selection experiment was run using semantic attributes. The results can be seen in Figure 4.3. These plots show the variation in performance with different choices of WekaMine pipeline algorithms and tuning parameters over the same fixed set of five 5X cross-validation folds (e.g. every point in the distribution is the average performance over a five times 5x CV experiment, a total of 25 experiments). In this test, semantic attributes perform noticeably better in predicting LUSC survival and noticeably worse in KIRC survival and comparably in GBM and OV. Predicting KIRC survival is the easiest prognosis between the four tumor types so it may be that there is a simple gene signature that provides most of the prognostic power for KIRC or it could be that the set of semantic attributes simply are not relevant to KIRC survival. To be a success semantic attributes need not perform best on every problem but only to be a reliably better approach on some subset of problems.

This test was expanded on by building classifiers also from different kinds

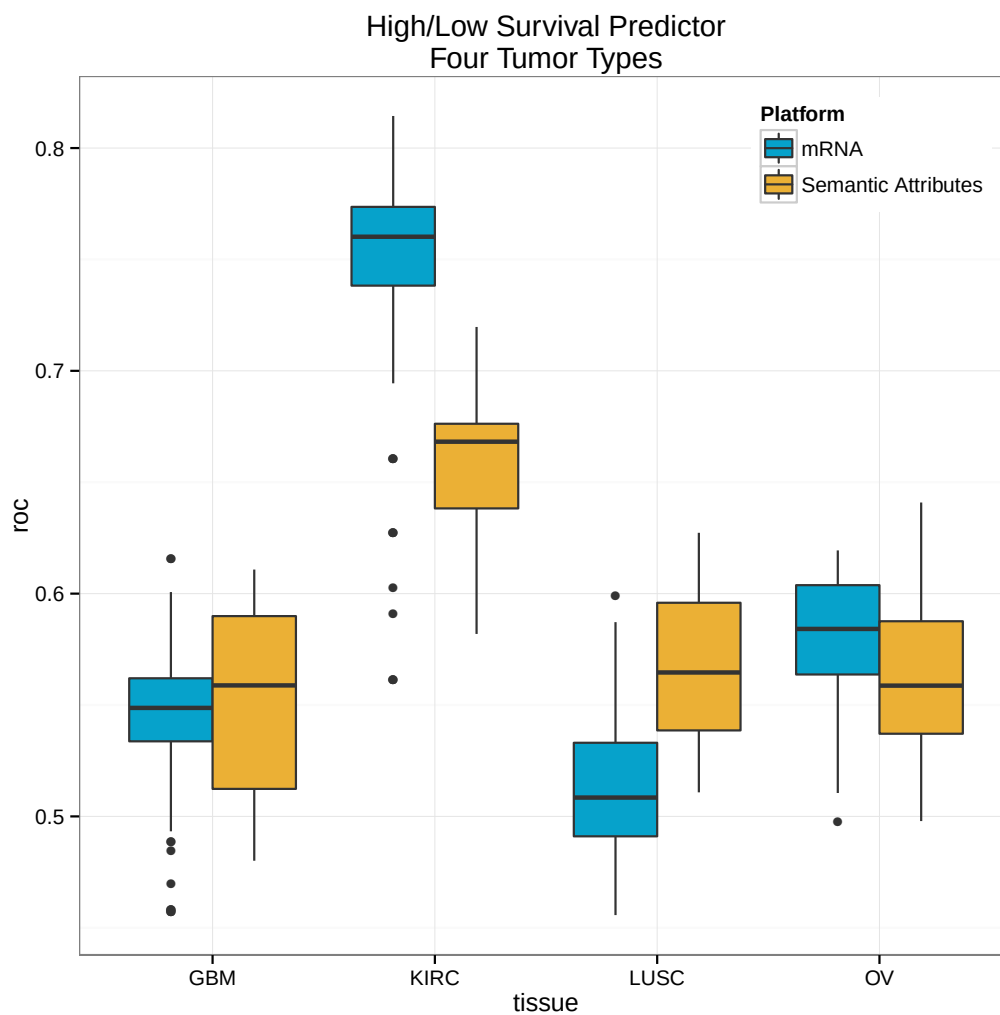


Figure 4.3: Survival predictors were built and validated in a 5 times 5x cross-validation scheme for four different tumor types using the 117 classifier set of semantic attributes in Table C.1. Semantic attributes outperform gene expression in LUSC with the median semantic classifier performing 5% better than median mRNA classifier and the best semantic classifier performing 3% better than the best mRNA classifier.

of data: micro-RNA (miRNA), messenger RNA from RNASeq (mRNA), copy number variation (CNV), and reverse phase protein array (RPPA). The results can be seen in Figure 4.4. In general, the semantic attributes performed comparably with other platforms. In LUSC there is a marked difference observed between RPPA data and other platforms, a difference replicated by the TCGA pancancer predictions working group [74]. In this case semantic attributes outperformed the three non-RPPA platforms and performed comparably with RPPA data. In some ways this is the most striking result here since the semantic attributes are built from mRNA data yet perform more on par with protein data, suggesting that both protein expression and semantic attributes may capture something that is obscured in the overall mRNA expression profile. And finally, while the average semantic attribute classifier in OV did not perform as well as the average mRNA classifier, the very best classifier for OV survival is a semantic attribute classifier.

4.1.6 Informative Semantic Attributes

Two of the benefits of semantic attributes are the potential interpretability of the resulting classifiers and the ability to use data from widely different sources. It is therefore interesting to ask what cross-tissue semantic attributes are informative in the prediction task. Table 4.1 lists the top 20 features from the best survival classifier for LUSC, along with the information gain of each feature. Notably, the most informative feature is a BRCA trained classifier for progesterone status. The possible influence of estrogen and progesterone receptors in lung cancer is the subject of a growing body of evidence as well as the potential target for endocrine therapies (Ishibashi [75] and Kazmi et al. [76]). The seventh most informative feature is a feature for classifying gender trained on KIRC data. Gender, which is itself a complex trait, has also been suggested as a factor in lung cancer incidence,

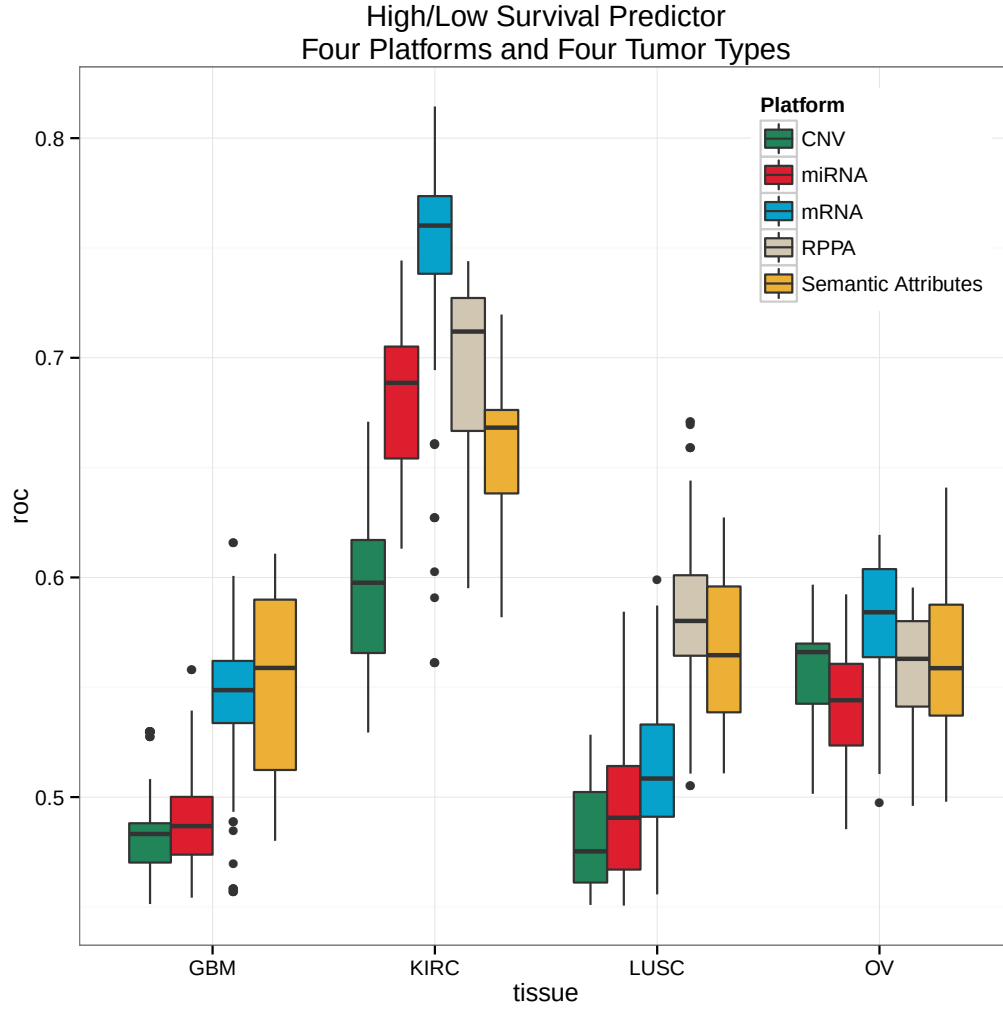


Figure 4.4: Survival predictors were built and validated in a 5 times 5x cross-validation scheme for four different tumor types over four platforms (GBM did not have RPPA data). Using the 117 classifier set of semantic attributes in Table C.1. Here RPPA data improves upon semantic attributes in LUSC.

Semantic Attribute	Information Gain (bits)
BRCAbreast_progesterone_receptor_status(Positive)	0.151
HNSC_TP53_MUTATION(TRUE)	0.0436
LUSC_ANO1_AMPLIFICATION(TRUE)	0.0294
BRCA_CCND1_AMPLIFICATION(TRUE)	0.0228
OV_MECOM_AMPLIFICATION(TRUE)	0.0203
BRCAbreast_progesterone_receptor_status(Negative)	0.0119
KIRCgender(FEMALE)	0.0115
UCECPTEN_NonSilent(NonSilentMut)	0.0109
LUADKRAS_NonSilent(NonSilentMut)	0.0106
GBM_EGFR_AMPLIFICATION(TRUE)	0.0102
LUADTP53_NonSilent(NonSilentMut)	0.0075
LUSC_WHSC1L1_LETM2_AMPLIFICATION(TRUE)	0.0072
BRCAlab_proc_her2_neu_immunohistochemistry_receptor_status(Negative)	0.0070
KIRC_PBRM1_MUTATION(TRUE)	0.0043
BLCA_TP53_MUTATION(TRUE)	0.0041
BRCAbreast_estrogen_receptor_status(Negative)	0.0035
UCEC_TP53_MUTATION(TRUE)	0.0035
UCEC_CTNNB1_MUTATION(TRUE)	0.0032
HNSC_C9orf53_CDKN2A_DELETION(TRUE)	0.0025
UCECTP53_NonSilent(NonSilentMut)	0.0024

Table 4.1: Top 20 semantic attributes for LUSC survival prediction.

severity, and progression (Carey et al. [77]). The tenth most informative feature, EGFR amplification trained on GBM data. EGFR expression is a predictor of survival for chemotherapy plus cetuximab (Pirker et al. [78]). These sensible top semantic attributes copuled with the improved performance of the LUSC semantic attribute classifier over the source mRNA data suggests that semantic attributes are able to use data from quite different sources and highlight how the resulting features are more easily interpreted than gene based signatures.

Chapter 5

Drug Sensitivity Prediction

5.1 Drug Sensitivity Prediction With Semantic Attributes

It's previously been observed that the efficacy of anti-cancer therapies is related to tumor subtype, progression, and pathway activities [79] [Peggs:2003bz] [80]. Being able to identify the therapeutic response of a patient from an analysis of the molecular states of tumor cells is therefore a key goal of precision medicine. One approach to exploring the relationship between molecular states and drug responsiveness is to examine human cancer cell lines screened against various compounds. The Cancer Cell Line Encyclopedia has performed screens on a large collection of small molecules on hundreds of cell lines which have been molecularly characterized with gene expression assays [81]. A number of studies have examined building predictors of cell response from this drug response dataset using modern machine learning tools [82] [83]. The goal in the present work is to examine if semantic attribute classifiers are able to boost the performance of these predictors over standard machine learning algorithms working alone.

In addition to the semantic attribute panels already introduced, I will bring in three additional semantic attribute panels for this problem: URSA tissue type classifiers, chromatin state attributes, and gene essentiality attributes.

Building these three semantic attribute panels will be discussed in the following sections, followed by the results of applying these, and previously discussed, semantic attributes to the drug sensitivity problem. After performance is reviewed I will discuss the utility and interpretability of attributes.

5.1.1 Modeling Tissue Type with URSA

Human tissues are composed of many different cell types. These cell types might be viewed as the major background state of cells against which other cellular changes take place. It seems sensible, then, to seek semantic attributes which capture this background cell state. Rather than build a cell-type classification system from scratch, I chose to use the URSA [84] system as a cell type identifier. URSA uses a form of Bayesian Network Correction, with the Brenda Tissue Ontology [85] as the network, to model the dependencies between cell types. A series of one-vs-all support vector machine classifiers for each tissue type was created and the output of these individual binary classifiers is "corrected" to a final call using constraints found in the Brenda Tissue Ontology Figure 5.1.

Applying the URSA tissue classifications to gene expression vectors gives a vector of tissue type probabilities which can be used as a semantic tissue attribute in subsequent prediction tasks.

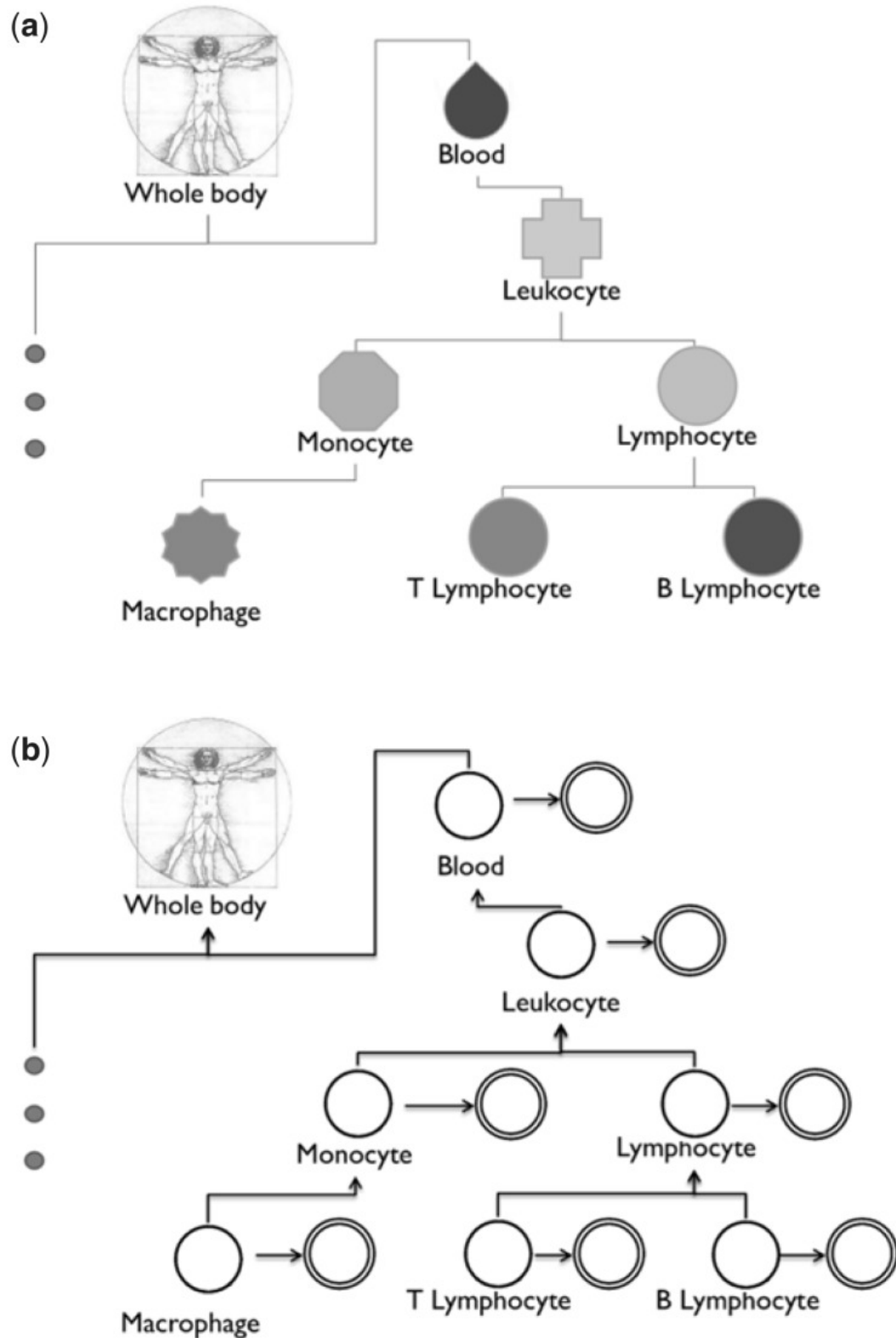


Figure 5.1: URSA is an ontology aware tissue classification system. a) Shows a section of the Brenda Tissue Ontology [85]. b) URSA uses a form of Bayesian Network Correction to model the dependencies between cell types, informing the most likely cell classification from noisy individual cell-type classifiers. Figure from [84].

5.1.2 Modeling Chromatin State with Gene Expression Classifiers

The first of two additional semantic attribute panels I will consider is a panel to consider epigenetic and chromatin states.

Epigenomics Roadmap

The human genome sequence is, modulo de novo mutations, fixed across human cell types. The epigenomic marks and chromatin state of the genome, however, varies considerably and in many cases systematically across cell types and contribute to the gene expression patterns that determine different cell types and states ([86],[87]). The role of chromatin states in cell type canalization suggests that knowing these states of cells can be a useful tool for identifying the background states of cells, and that information might in turn be useful in trying to determine secondary aspects of cells, such as their responsiveness to drugs and compounds.

Typically, however, we will not have direct measurements of epigenetic or chromatin states for the samples we are interested in due to the difficulty and expense of generating an epigenomic map of a cell. We do, however, have abundently available gene expression measurements, either from microarrays or RNASeq experiments. If it is possible to derive some aspect of epigenomic state from gene expression, to build models of epigenomic state from expression patterns, that state information could then be used as an informative semantic feature for other analysis tasks.

The Epigenomics Roadmap Project [88] provides us with both expression data and extensive mapping of epigenomic marks and chromatin states for 111 reference human epigenomes. This data can be used to link chromatin states to expression

data to build models of chromatin state that can be derived from expression.

The first step in approaching this problem is coming up with a training set of mappings between chromatin states and RNASeq data and there we run into the large size of epigenomic state datasets. Epigenomic marks are mapped at the single base resolution, so for each sample there are billions of state calls across five core histone marks, not including DNA methylation and chromatin accessibility measurements.

ChromHMM

In order for the Roadmap Project itself to summarize and start to make sense of this data from the diverse set of epigenomic marks, ChIP-seq data for 5 core marks (H3K27me3, H3K36me3, H3K4me1, H3K4me3, H3K9ac, H3K9me3) + H3K27ac were computationally integrated using a hidden Markov Model (HMM) packaged as open source software ChromHMM [89]. Figure 5.2 shows an example of ChromHMM converting the base pair resolution signal from six epigenetic marks into a single track of state calls. ChromHMM produces a state call for each non-overlapping 200bp window.

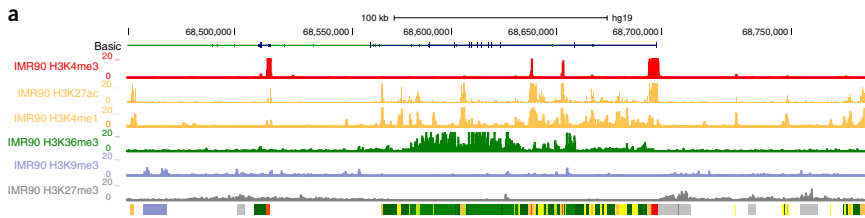


Figure 5.2: Tracks of multiple histone marks from one cell type (IMR90) showing the histone marks and the corresponding ChromHMM states (bottom) Fig from [89].

Broadly the state calls can be categorized as corresponding to active or repressed chromatin states but can be further broken down into sets of states asso-

ciated more fine grained aspects of chromatin regulation such as active transcription or the formation of heterochromatin. Figure 5.3 gives a breakdown of the state calls both in broad and more fine grained terms.

chromHMM 15-state Model

Major Category	Minor Category (9)	State (15)	Color Code	Description
Active				
	Tstart	TssA		Active transcription start site
		TssAFlnk		
	Transcription	TxFlnk		Transcribed state showing both promoter and enhancer signals
		Tx		Actively transcribed states
		TxWk		
	Enhancer	Enh		Enhancer states
		EnhG		
	ZincFinger	ZNF/Rpts		Zinc finger protein gene associated state
Repressed/ Inactive				
	Heterochromatin	Het		Constitutive heterochromatin
	Bivalent	TssBiv		Bivalent regulatory states
		BivFlnk		
		EnhBiv		
	Repressed	ReprPC		Repressed polycomb states
		ReprPCWk		
	Quiescent	Quies		

Figure 5.3: Major and minor categories of epigenomic state mapped to chromHMM states.

The chromatin states annotated by ChromHMM for any given region of the genome varies considerably across tissue types, as can be seen in Figure 5.4. This tissue-stable pattern of states is what we hope to capture in some way in a classifier model for use as semantic attributes.

Blocks of ChromHMM Calls

While ChromHMM reduces the original multi-track chromatin marking data down to a single 15-state track for each celltype sample there are still on the order of 15 million ChromHMM calls per sample. Some way to reduce this to a smaller

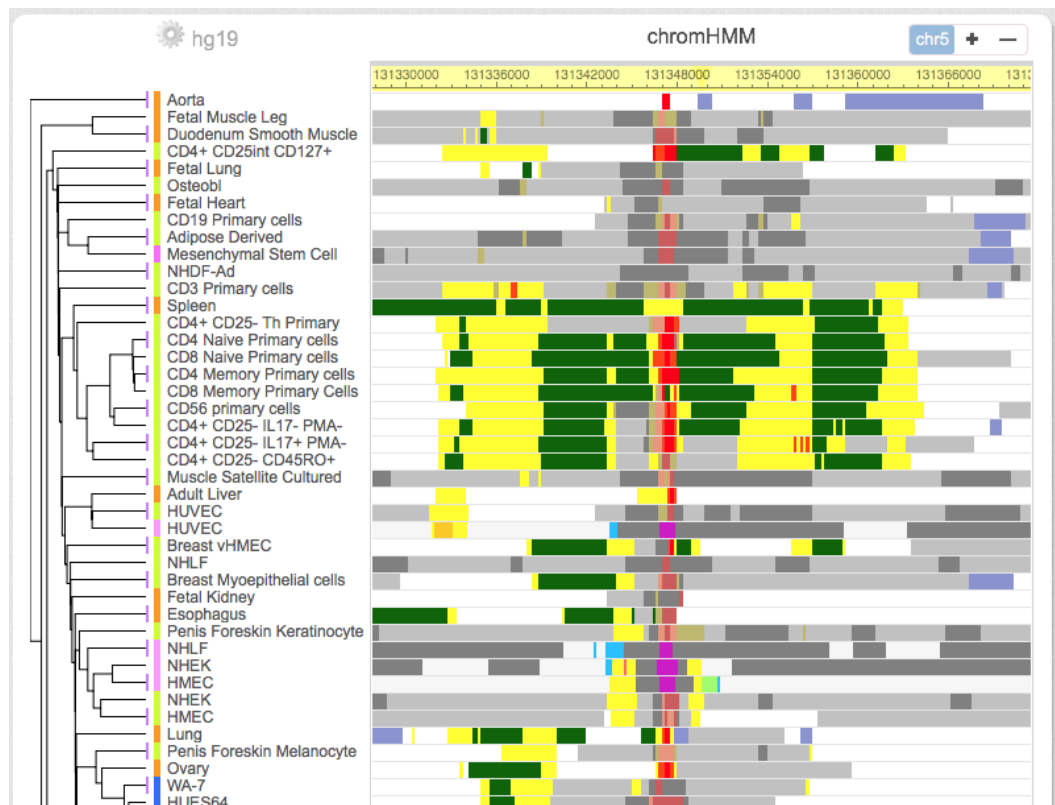


Figure 5.4: Chromatin states identified by chromHMM in a region of Chromosome 5 across a range of tissues.

Distribution of chromHMM feature Sizes E011

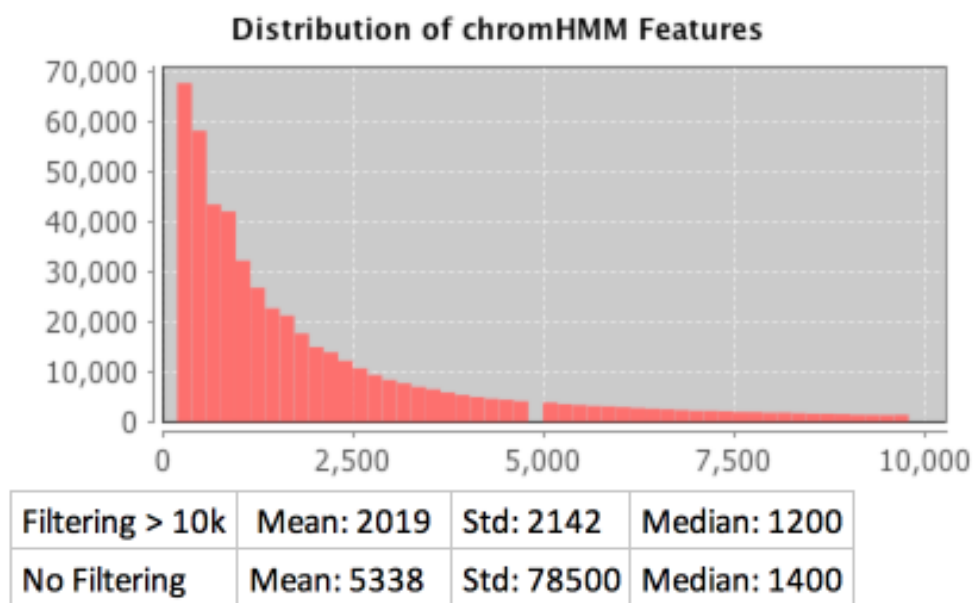


Figure 5.5: Spatially sorted ChromHMM calls are scanned for contiguous blocks. 510,000 blocks were identified. The distribution of block sizes is shown here.

informative set is needed in order to produce a set of semantic attribute classifiers.

One approach to simplifying this information is to look at continuous blocks of same-call ChromHMM states, since although the calls are made at 200bp blocks, many consecutive blocks may have the same 15-state value. I wrote software to scan the ChromHMM bed file of calls to find such contiguous blocks. The size of the blocks varied considerably from 200bp to over 80 Kbp, though blocks over 10 Kbp are rare. The distribution of contiguous block sizes is shown in Figure 5.5. As this distribution makes clear, the bulk of contiguous blocks are quite small in size. As a result, simplifying down to contiguous blocks would only reduce the number of distinct state values per sample to 510,000. I investigated the feasibility of building classifier models at this resolution with a modest degree of success. Details of these experiments can be found in Appendix XX. Ultimately, the combination of the modest accuracy of these models with the tremendous scale of trying to model 510,000 separate genome positions, led me to abandon this approach in favor of further simplifications.

Chromatin Domains

Another approach to simplifying this information is to summarize the data over topological domains in chromatin structure as illuminated by Hi-C experiments [90]. Hi-C allows the mapping of the frequency of contact between different physical locations on the genome, and the distribution of these contacts gives information about the 3-dimensional proximity, and so conformation, of chromatin Figure 5.6. The topological domains so identified appear to be stable across cell types and define distinct regions in chromatin structure.

One way to try to reduce the complexity of chromHMM states is to summarize these states on a per-domain basis. These domains are of variable size, but are

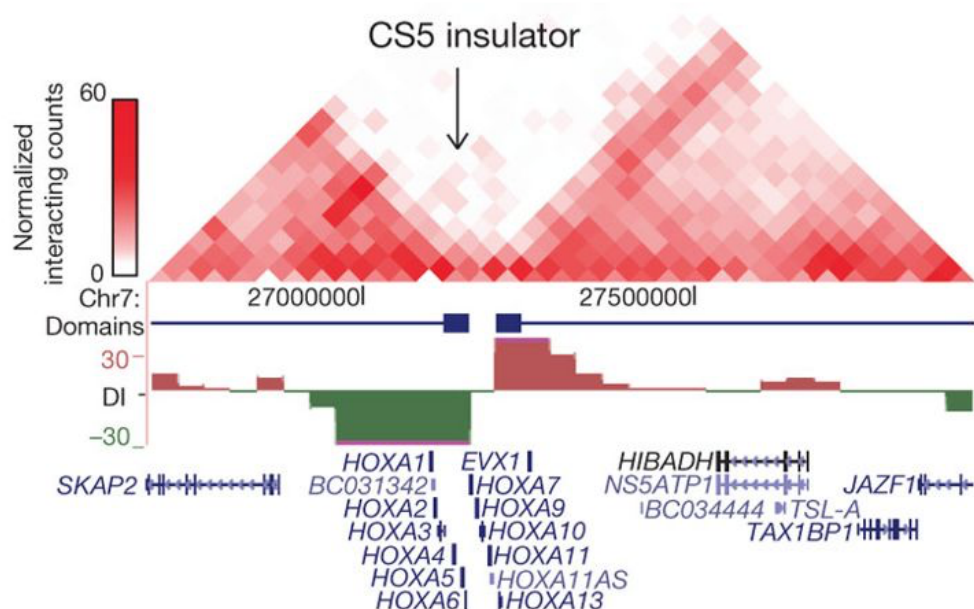


Figure 5.6: Two-dimensional heat map of chromosome interactions surrounding the Hoxa locus and CS5 insulator in IMR90 cells.

typically on the order of 1 Mbp in size. Across the genome there are 3127 domains defined by [90]. I have restricted my analysis to binary classifiers, so the 15 ChromHMM states will be modeled as 15 binary classifiers indicating the presence or absence of a given state. For this domain-level modeling of ChromHMM states, the training value for each classifier is taken to be the fraction of bases in that domain that have a particular ChromHMM state call, as illustrated in Figure 5.7. The result is a collection of 46,905 individual binary states, each one a combination of a genomic coordinate region (domain) and a particular ChromHMM call fraction Figure 5.8.

Figure 5.7

Figure 5.8

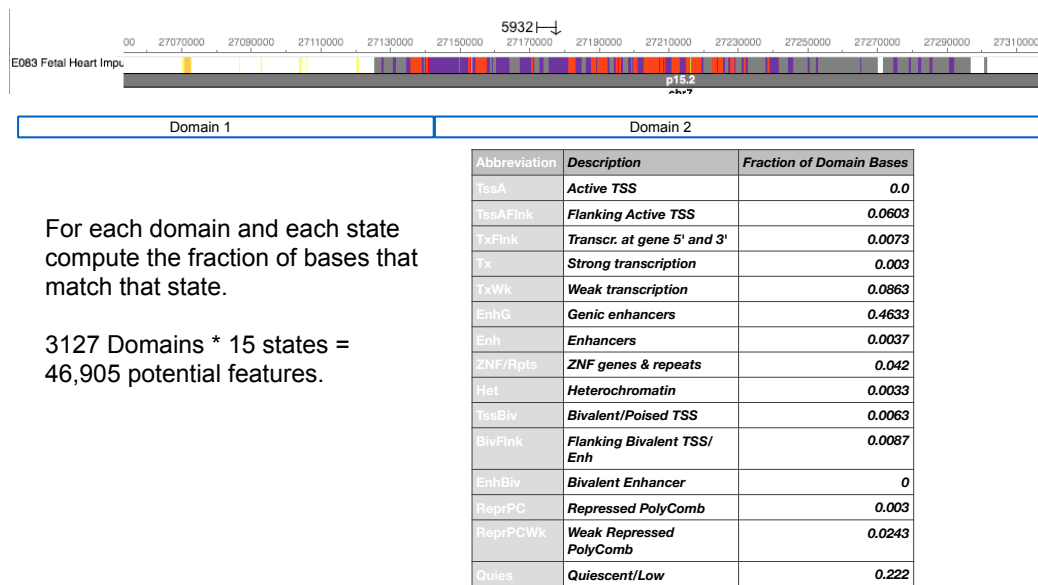


Figure 5.7: Using chromatin domains determined by Hi-C data, the fraction of particular epigenomic states in that domain are used to provide a summary of the chromatin state in that domain.



Figure 5.8: Each domain classifier is a combination of a genomic location and a ChromHMM call fraction.

Clustering Chromatin States

Performing model search to build 49,000 individual binary classifiers would still be an prohibitively expensive computationally task. While this might be something worth pursuing for future work, to test out the idea that information about chromatin structure might make usable semantic features I was willing to discard some of this fine grained information in order to reduce the size of the computational problem.

Fortunately, since chromatin structure is highly correlated across regions that opens up an avenue for a simplification: identify the highly correlated chromatin mark states and then build a representative expression classifier from each correlated group. To do this I performed k-means clustering, with $k=100$, on each of the 15 chromHMM chromatin states datasets, to identify 100 representative clusters of chromatin states across those datasets, as illustrated in Figure 5.9 and Figure 5.10. Each cluster is then taken to capture some essential information about large scale variations in chromatin structure. Doing this for all 15 states produces a much more maneagable 1500 separate chromatin state clusters which constitute the meta-data input for training classifiers.

Building Models

Using the wekaMine pipeline, I performed a model selection experiment on these 1500 target classes to find the best combination of algorithm and hyper-paramters for each one. Such model-selection experiments typically involve about 100 cluster jobs per classifier, so a total of 150,000 compute cluster jobs were performed to identify the best models. Final classifiers for each chromatin mark cluster were then built to be used as semantic attributes for other classification tasks, such as cell state, drug prediction, and others.

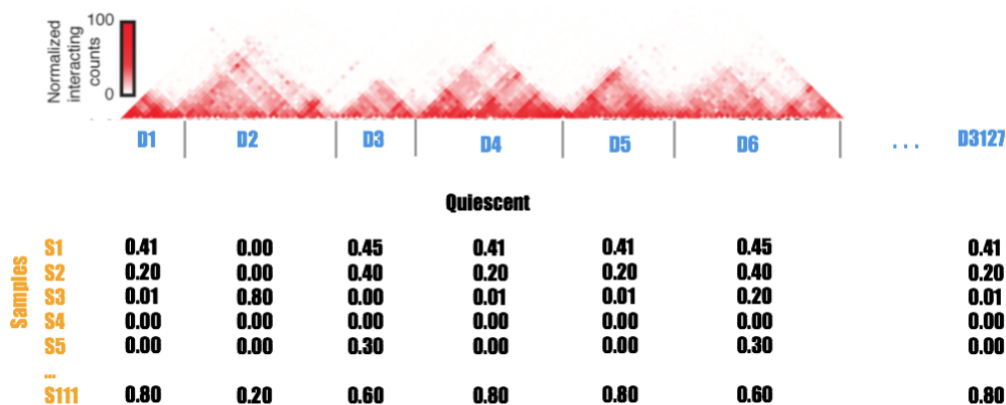


Figure 5.9: For each of fifteen chromatin states a state value per domain is created.

To summarize the process:

- Model each of 15 states as binary variable
- Summarize with fraction within a Hi-C domains.
- Cluster to derive representative states
- 1500 classifiers to represent entire genome

5.1.3 Modeling Gene Essentiality with Gene Expression Classifiers

Project Achilles at the Broad Institute is an ongoing project to identify gene essentiality across a range of cell lines [91, 80, 92, 93]. By selectively knocking out genes using RNAi and, more recently, CRISPR-Cas9, they are developing a map of genes essential for cell survival. In human cancer cell lines, there may be many mutations and rearrangements that affect the cell's self-regulation in various

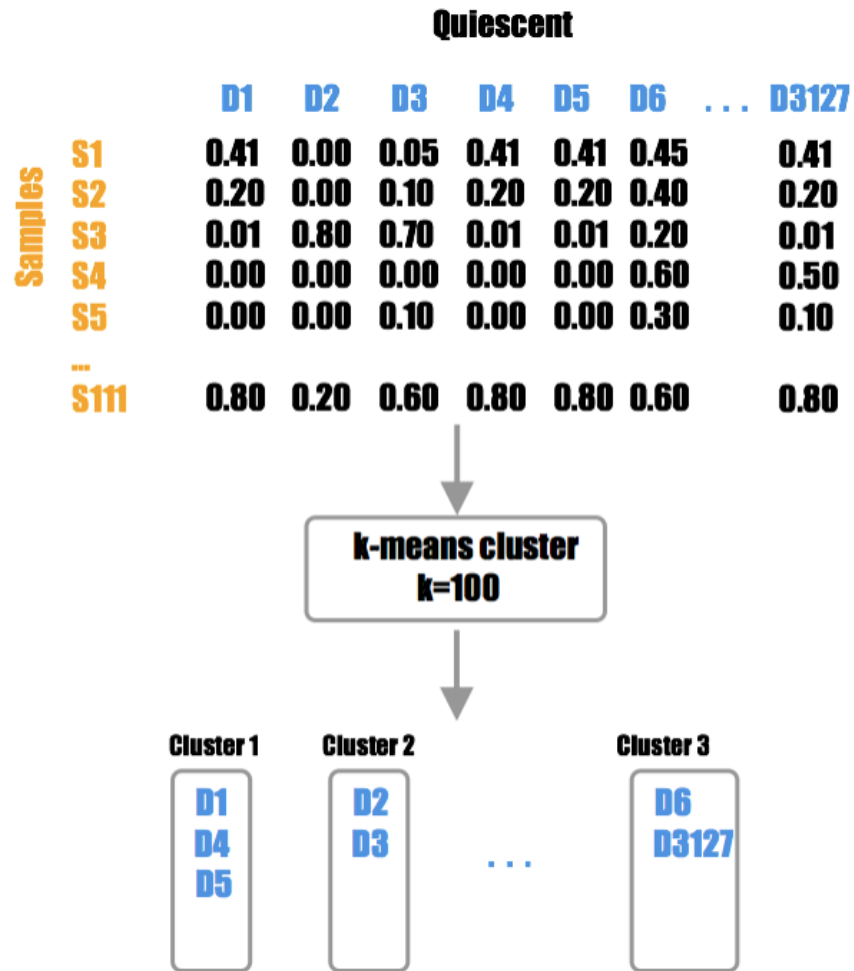


Figure 5.10: K-means clustering is performed across domains to identify domains that are co-varying. A representative domain is chosen from each cluster to represent that cluster in subsequent processing.

ways. These effects result in cell-specific vulnerabilities when, for example, one pathway of a normally redundant pathway is not present, making the genes in the alternative pathway now essential for the cell's survival.

Using this gene essentiality data from Project Achilles along with cell-line matched data for gene expression, it's possible to build predictors of gene essentiality from the expression profile. To build semantic attribute classifiers for gene essentiality I used a snapshot of the gene essentiality data assembled for the The Broad DREAM 9 challenge [94]. This data was produced by screening approximately 98,000 shRNAs targeting around 17,000 genes. After 16 population doublings, or 40 days in culture, the effect of shRNA infection on cell viability is assessed. Gene essentiality is measured by the shRNA levels at the end of the screen compared to the initial DNA pool. To account for the effect of knockdowns of unintended genes the shRNA data was processed with the package DEMETER [93], which models response levels as a linear combination of target specific gene effects and off-target effects, to obtain an estimate of the gene specific essentiality values. The final values are scaled to log fold change, with lower values representing higher dependency on the gene, Figure 5.11.

To build models of gene essentiality I first binarized the data with `wmDichotomize` using an upper/lower quartile filter. For this filter `wmDichotomize` computes the upper and lower quartiles of all of the log fold change values. Samples with values above the upper quartile cutoff were assigned an "upperQuartile" label and samples with values below the lower quartile cutoff were assigned as "lowerQuartile". Values in the middle were treated as unknown values by the classifiers and effectively omitted from the training sample. This is justified because only the most extreme values carry a significant signal, with intermediate values working more as a kind of background noise that may degrade the confidence of

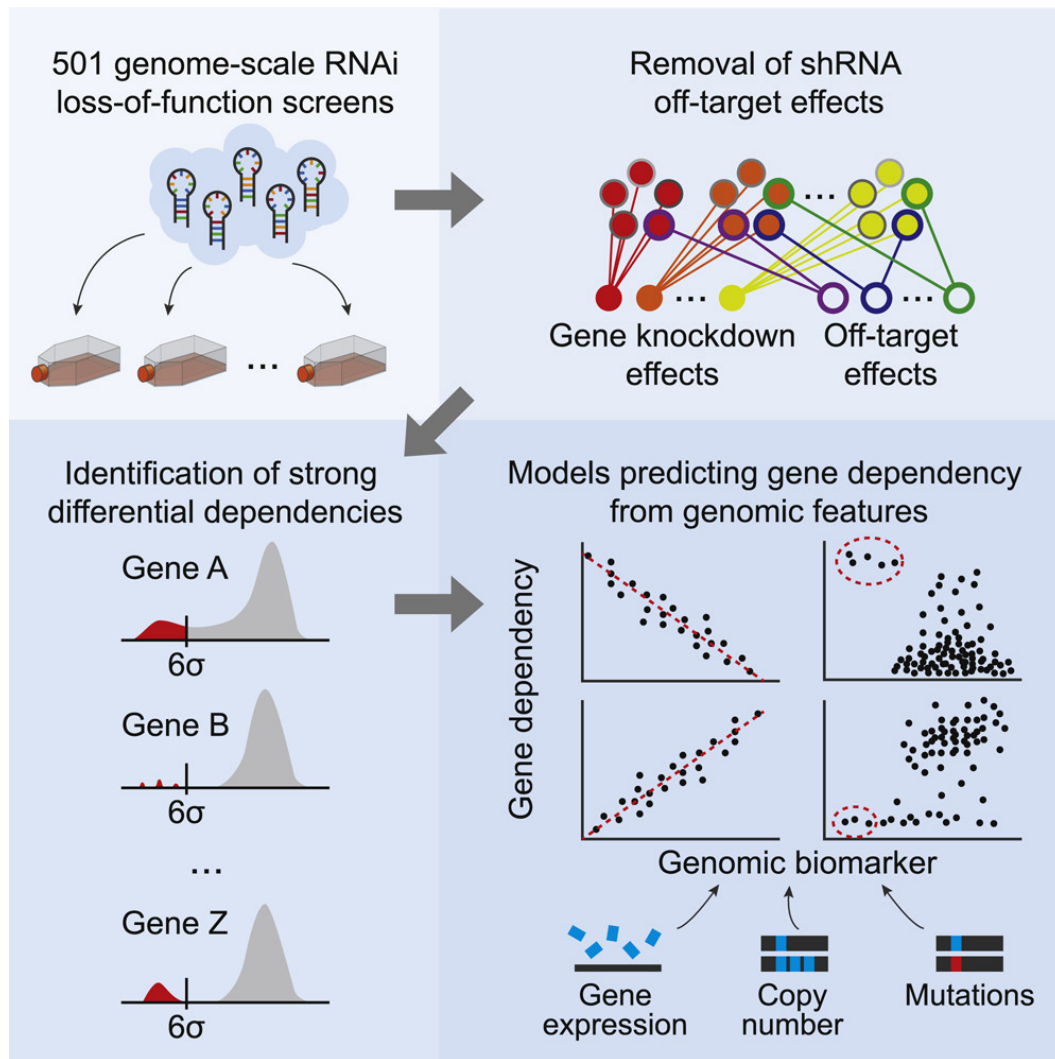


Figure 5.11: Using RNAi loss-of-function screens gene knockdown effects are isolated. Among these, strong differential knockdown dependencies among genes are identified. Using matched genomic features, such as gene expression, predictors of gene essentiality can be built. Figure from Tsherniak et al. [93]

the calls that are made.

5.1.4 Drug Sensitivity With Gene Level Features

To model drug sensitivity I used data from the Cancer Cell Line Encyclopedia (CCLE), specifically Cancer Therapeutics Response Portal (CTRP v2, 2015) [95] [96] [97] [98]. This is a dataset of 545 small molecules and select combinations screened against 907 cancer cell lines. Of these, 887 cell lines had data for all 545 small molecules.

Cell lines were grown in their preferred mediums and treated with compounds at eight concentrations for 72 hours. Sensitivity was measured using CellTiter-Glo to measure ATP levels as a proxy for cell number. For each cell line a dose-response curve is generated and an EC50 score is calculated, where the EC50 is the concentration of a compound on a drug response curve where 50% of it's maximum effect is observed.

In order to build classifiers for drug responsiveness I first dichotomized these EC50 values using `wmDichotomize` into a high-sensitivity and low-sensitivity set. In order to reduce noise from indeterminate outcomes `wmDichotomize` assigned high-sensitivity to drugs with EC50 scores greater than or equal to the 75% quartile of values for that drug, and low-sensitivity to drugs with EC50 scores less than or equal to the 25% quartile of values. Values that fell between these two quartiles were treated as "unknown response", and were effectively removed from the training dataset. From the initial dataset of 1036 samples, this transformation eliminated 149 samples as being ambiguous, leaving 887 unambiguous samples for training.

A `wekaMine` model selection experiment was performed on this dataset with five re-randomized replications of 10x cross-validation for each algorithm and hy-

perparameter combination. Algorithms considered were random forest, logistic regression, linear kernel support vector machines, and quadratic kernel support vector machines. Of the resulting classifiers, only those that performed better than chance by 5% were retained as having some degree of learning over simply guessing. Of the 545 small molecule models, 487 met this minimum quality threshold.

The results of the model selection experiment are illustrated in Figure 5.12. For the purposes of this plot, and many later comparisons, I further restrict consideration to "good" classifiers, which I have taken to be classifiers with a median roc across experiments of 0.75 or better. While classifiers with performance below this but still above chance do reflect some level of information capture and generalization in the classifiers, and so can show the influence of semantic attributes on performance, classifiers with such low performance are not generally good enough that they would be used in an application.

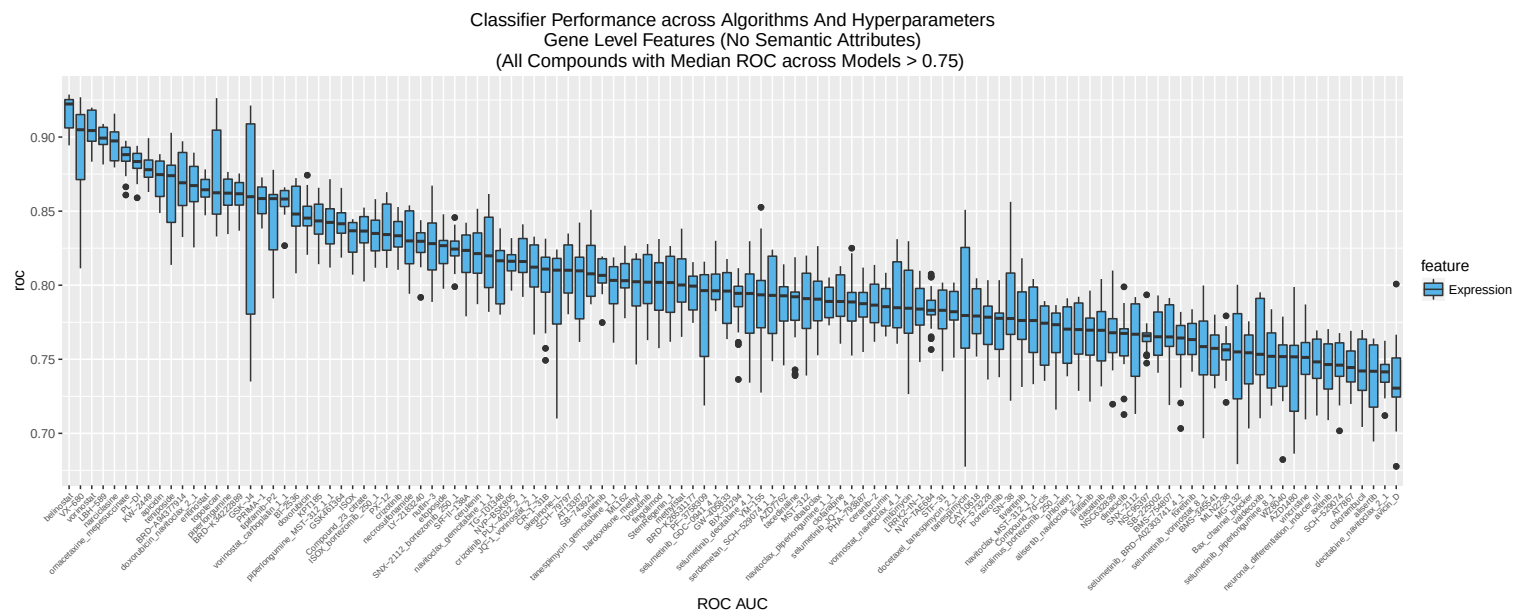


Figure 5.12: Performance of drug sensitivity classifiers built with gene-level expression features only. Performance is shown as varying across machine learning algorithms and hyperparameters. Each point in the distribution is the result of a 5 times 10x cross-validation experiment. Only compounds with median ROC ≥ 0.75 are shown.

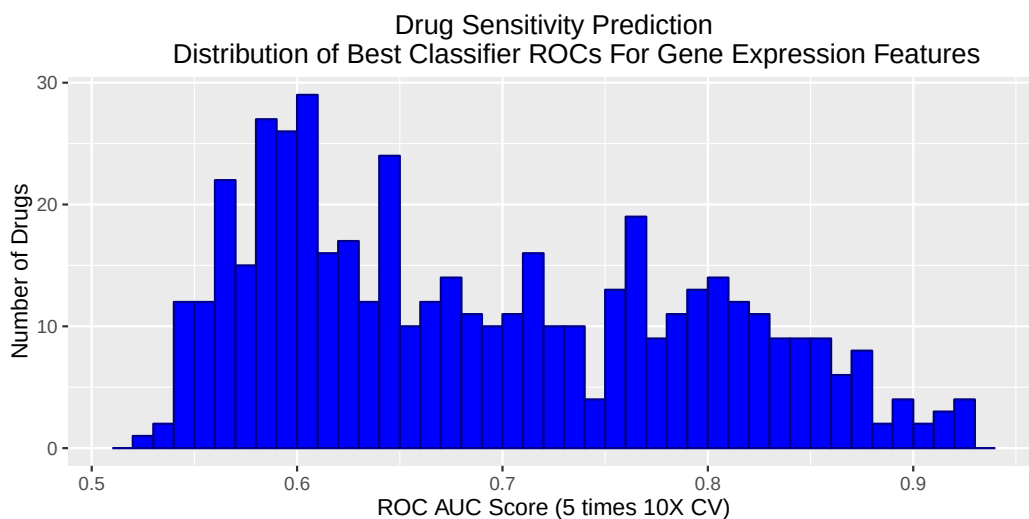


Figure 5.13: Distribution of ROC values for the best classifiers built from expression data without feature space transformation

While looking at the whole range of algorithm and hyperparameter choices allows us some insight into the robustness of our ability to build classifiers with a particular dataset and particular pre-processing, for actual applications of drug sensitivity classifiers we are most interested in the best possible classifier/hyperparameter choice for each drug prediction target. This distribution is shown in Figure 5.13. Of the 487 drug sensitivity classifiers shown, 158 have roc scores ≥ 0.75 , 93 have roc scores ≥ 0.80 , and 9 with roc scores ≥ 0.90 .

5.1.5 Applying Semantic Attributes to Drug Sensitivity

COMBINED Attribute Performance

All together, I combined the semantic attribute features from MEMo events (52 attributes), mutations (229 attributes), URSA tissue types (166 attributes), and the two additional transformation, chromatin state (1065 attributes) and gene essentiality (4307 attributes), into a combined collection of attribute classifiers. This set of 5882 semantic features I will call the "COMBINED" set of attributes.

I applied each of the 5882 classifiers in the COMBINED set to the gene expression profiles of 1036 CCLE samples to obtain a transformed vector for each CCLE sample. Using the 887 dichotomized drug response values as described in 5.1.3 I ran a wekaMine model selection experiment to gather information about the performance of models built with different classifiers and hyperparameters. The result of these model selection experiments is shown in Figure 5.14. The plot combined the results from Figure 5.12 so that the range of performance across algorithms and hyperparameters can be seen for each drug both for the data transformed with the COMBINED set of feature transformations and for raw expression data (Expression).

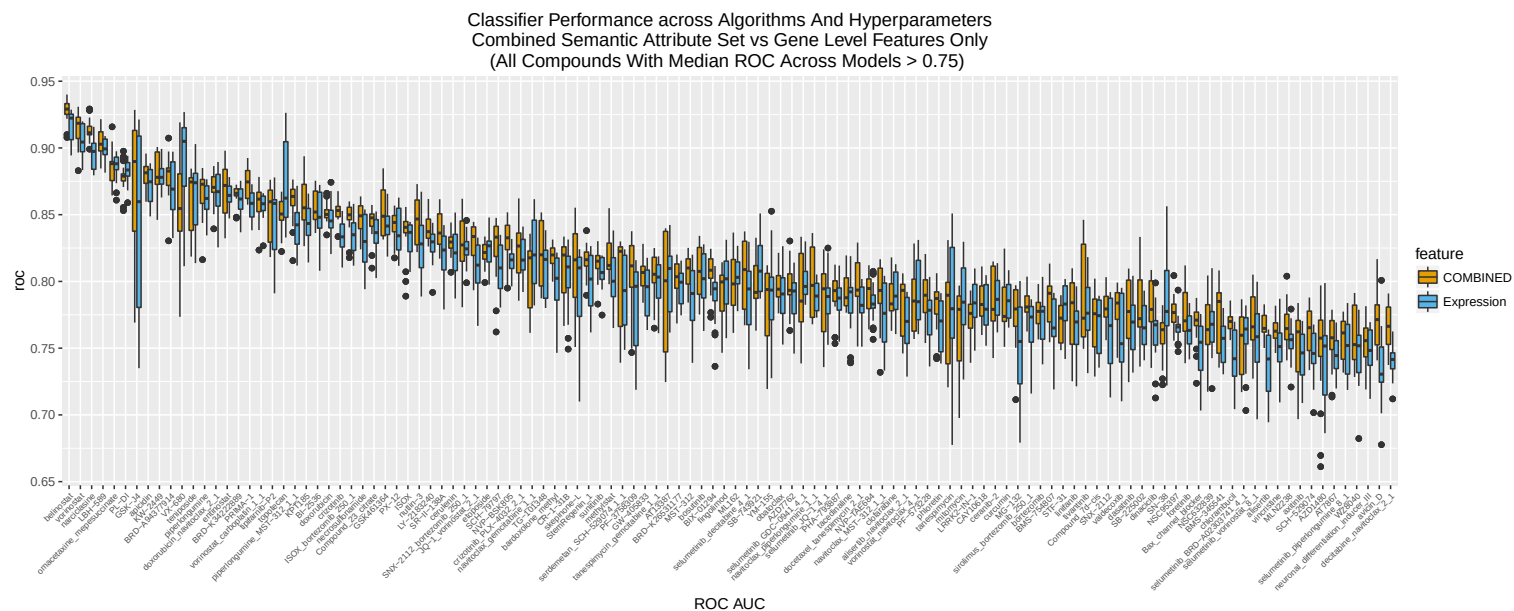


Figure 5.14: Performance of drug sensitivity classifiers built with combined set of 5882 semantic attributes. Performance is shown as varying across machine learning algorithms and hyperparameters. Each combination of algorithms and hyperparameters (e.g. point in the distribution) is the result of a 5 times 10x cross-validation experiment. Only compounds with median ROC ≥ 0.75 are shown.

COMBINED Performance Across All Algorithms

I take two approaches to evaluating the performance of semantic attributes on this data. The first asks the question, "Do semantic attributes improve the performance of classifiers on average across algorithms?", and the second asks the question, "Do semantic attributes improve the performance of the best algorithm available?". The first question gets at the robustness of semantic attributes, since an effect that is seen across a wide range of algorithms and parameters is a more robust effect than one that is tied to a specific algorithm. Also, although each evaluation itself is the result of five randomized replications of 10x cross validation, fifty total experiments, looking at data across parameters and algorithms increases the sample size. The second question is, of course, the most important question for application, since ultimately any research or clinical application of gene expression classifiers will want to use only the best possible algorithm/transformation combination.

Considering the aggregate question first, we can compute the median performance for each drug across algorithms/hyperparameters using gene level features, giving a set of median values for each drug, and then compute the median performance for each drug across algorithms/hyperparameters using COMBINED transformed data. The difference between the median performance for raw and transformed data I will call the median delta model performance. The distribution of these deltas is shown in Figure 5.15

As the figure illustrates, the median performance difference of the classifiers is shifted noticeably from the 0 mean expected if benefits were randomly distributed between conditions. To validate this visual impression, I look at the distribution of roc values from both conditions. Unlike the delta distributions, the distribution of roc values is not obviously gaussian in shape, and so I use a Wilcoxon rank sum

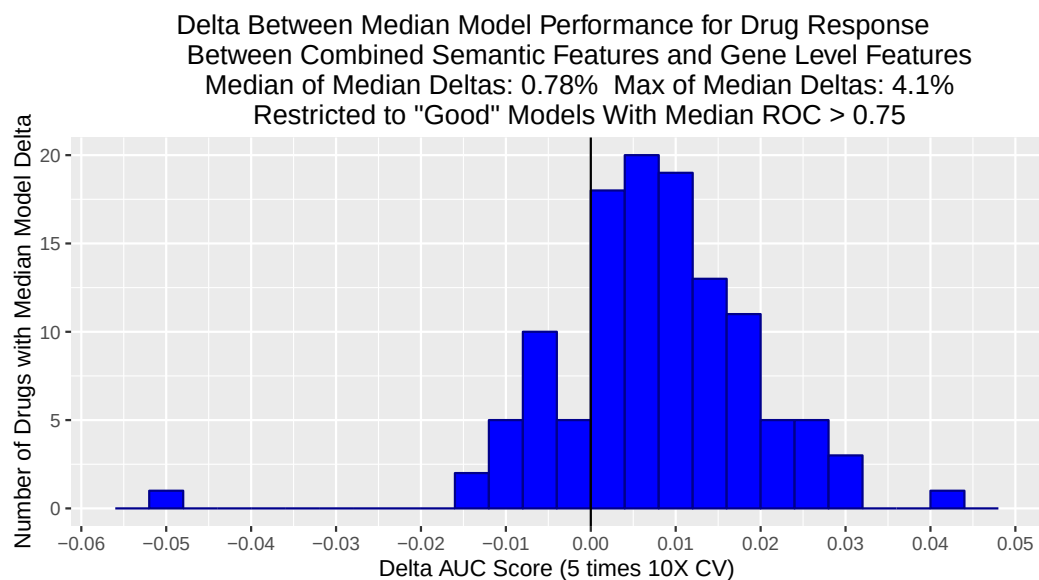


Figure 5.15: Computing the median performance across algorithms for each drug, the median performance for each drug without semantic transform is compared to the median performance for each drug with semantic transform, giving a delta value. The distribution of deltas is shown. Overall, the median of these median-value deltas is 0.78% "Good" classifiers only shown.

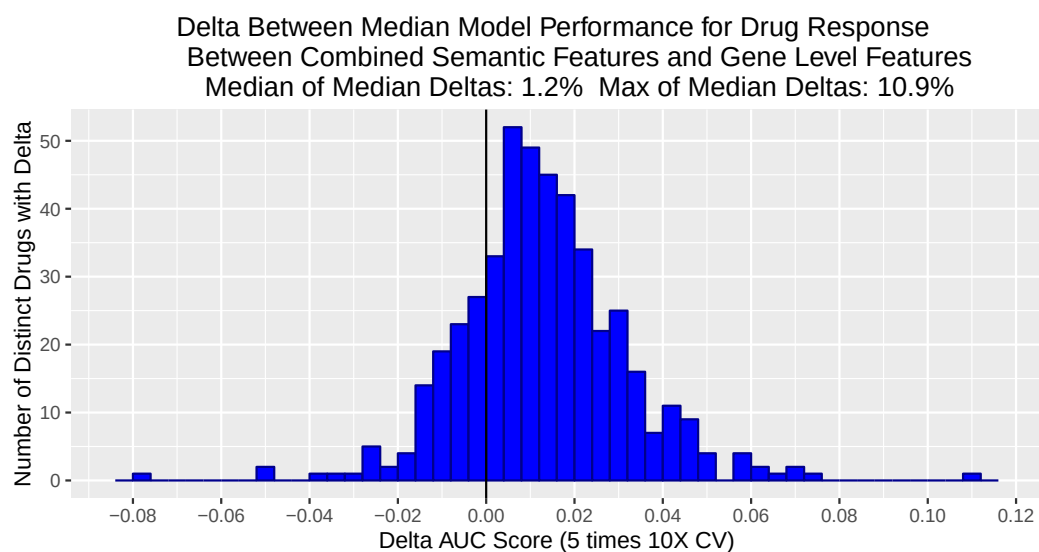


Figure 5.16: Computing the median performance across algorithms for each drug, the median performance for each drug without semantic transform is compared to the median performance for each drug with semantic transform, giving a delta value. The distribution of deltas is shown. Overall, the median of these median-value deltas is 1.2% All classifiers shown.

test to evaluate the null hypothesis of no location shift between the distributions. Using this test I find the null hypothesis of no shift to be rejected with p-value $< 2.2\text{e-}16$.

Figure 5.15 is restricted to only the "good" classifiers, those with median roc ≥ 0.75 , so as to be directly comparable to Figure 5.14 which is similarly restricted.

It's potentially instructive to look at the distribution of deltas over the entire range of classifiers, good and bad. Figure 5.16 shows a similar improvement, here 1.2%. Since on the low end many classifiers simply fail to generalize at all, there is a lot of variance among poor performing classifiers, resulting in some large deltas on both sides of the distribution. Even so, none of these high delta classifiers were boosted from the poor classifier set into the "good" classifier category of ≥ 0.75 roc. As a result, this view of the results only interesting as a check on restricted plot.

COMBINED Performance Among Best Algorithms

While the differences in median performance across algorithms and parameters gives the most robust view of whether or not semantic attributes are able to boost performance, and the answer is clearly "yes, but very modestly", this doesn't address the question of whether semantic attributes might be useful in further research or clinical settings, since in those settings we are interested in the best possible classifier for each drug sensitivity prediction. If most of the difference in performance seen in Figure 5.15 occurs in average or below average classifiers it might be the case that best classifiers are not improved in any notable degree. To assess this, I look at the distribution of deltas between the best gene-level classifier and the best COMBINED semantic attribute classifier. Each such result is the product of a five times 10X cross-validation experiment. The results are shown in

Figure 5.17. The overall improvement is smaller among the best algorithms, at 0.97% median delta, and the maximum improvement is only 4%, but even so there are 56 drugs with an improvement of 1% or more, 14 with an improvement of 2% or more, and 3 with an improvement of 3% or more. Figure 5.18 shows the list of ≥ 0.75 roc drug sensitivity predictors and their delta performance compared to gene level features. Discussion of patterns within the most improved drugs and the semantic features these classifiers use will occur in a later section.

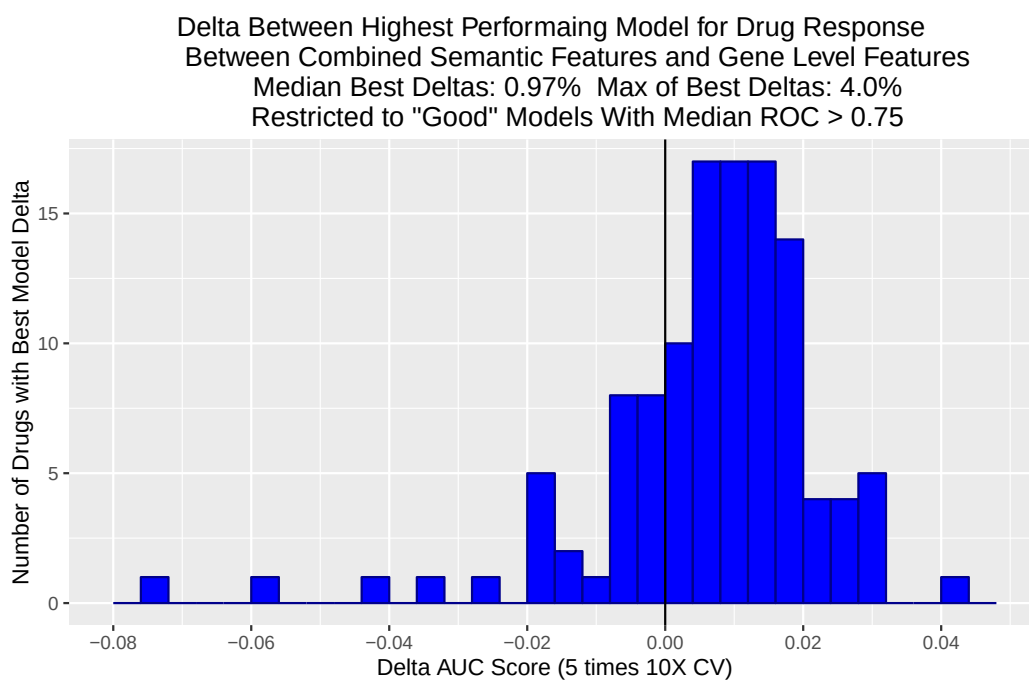
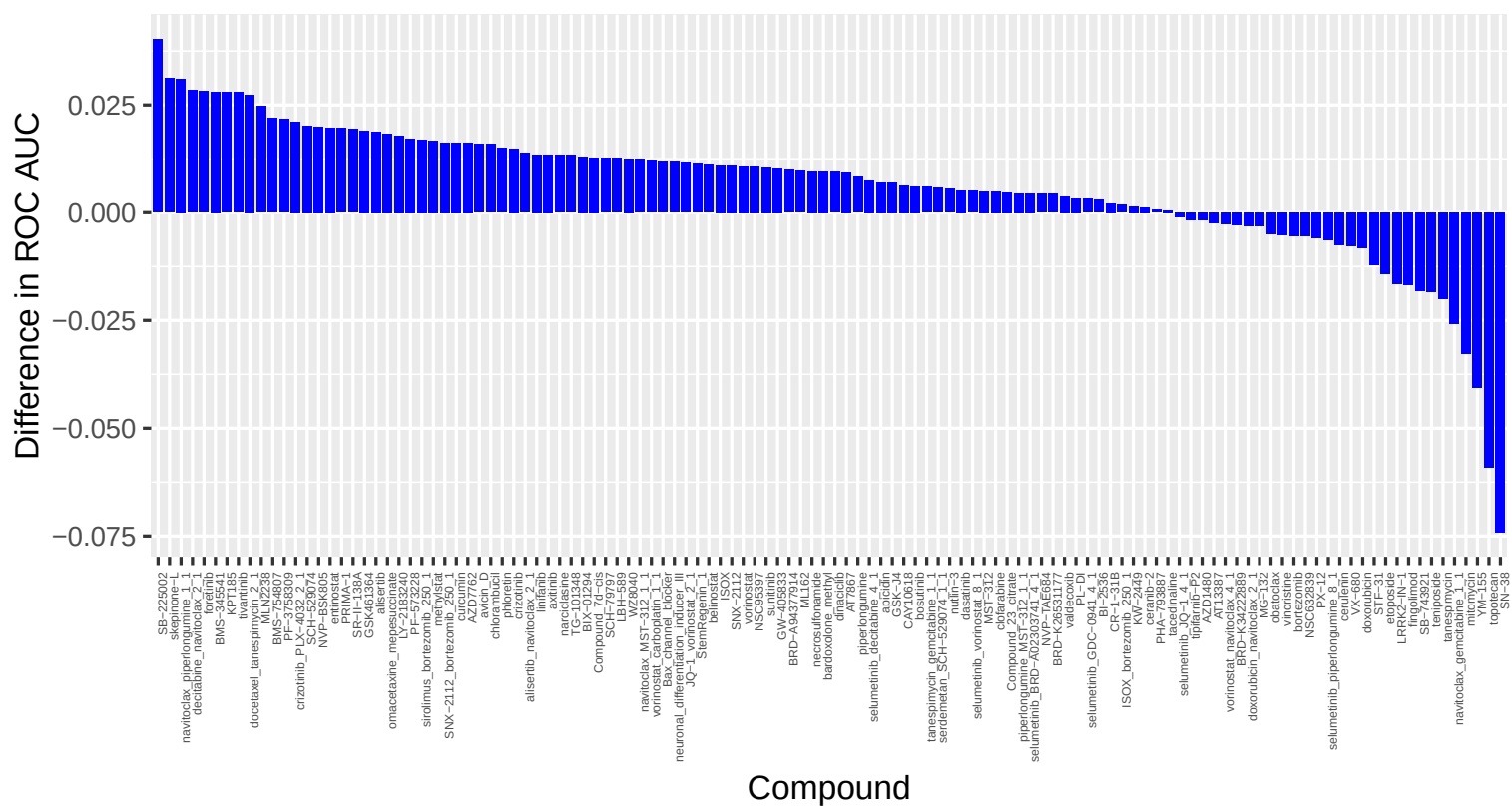


Figure 5.17: Distribution of deltas between the best gene level feature classifier for each drug and the best COMBINED semantic attribute transformed classifier for each drug. "Good" classifiers only shown.

Difference Between Maximum AUC of Combined Semantic Attributes Across Algorithms and Hyperparameters and Gene Level Expression Across Algorithms and Hyperparameters Restricted to "Good" Models With Median ROC > 0.75



Looking at the individual panels that make up the COMBINED set of attributes allows us to see how well each set of attributes would perform alone.

URSA Attribute Performance

URSA tissue type calls were collected as a binary vector of tissue-type probabilities, yielding 166 attribute classifiers in total. This transformation was used as input data for drug response prediction and a wekaMine model selection experiment was performed to evaluate performance over algorithms and hyperparameters. Zero of drug sensitivity models from this experiment had an ROC score over 0.65. This is not surprising, on reflection, because in practice URSA gives one or two positive probability values, corresponding to the most likely tissues, for each sample. Such an extremely sparse vector is not going to be useful, on its own, for building classifiers.

MEMO Attribute Performance

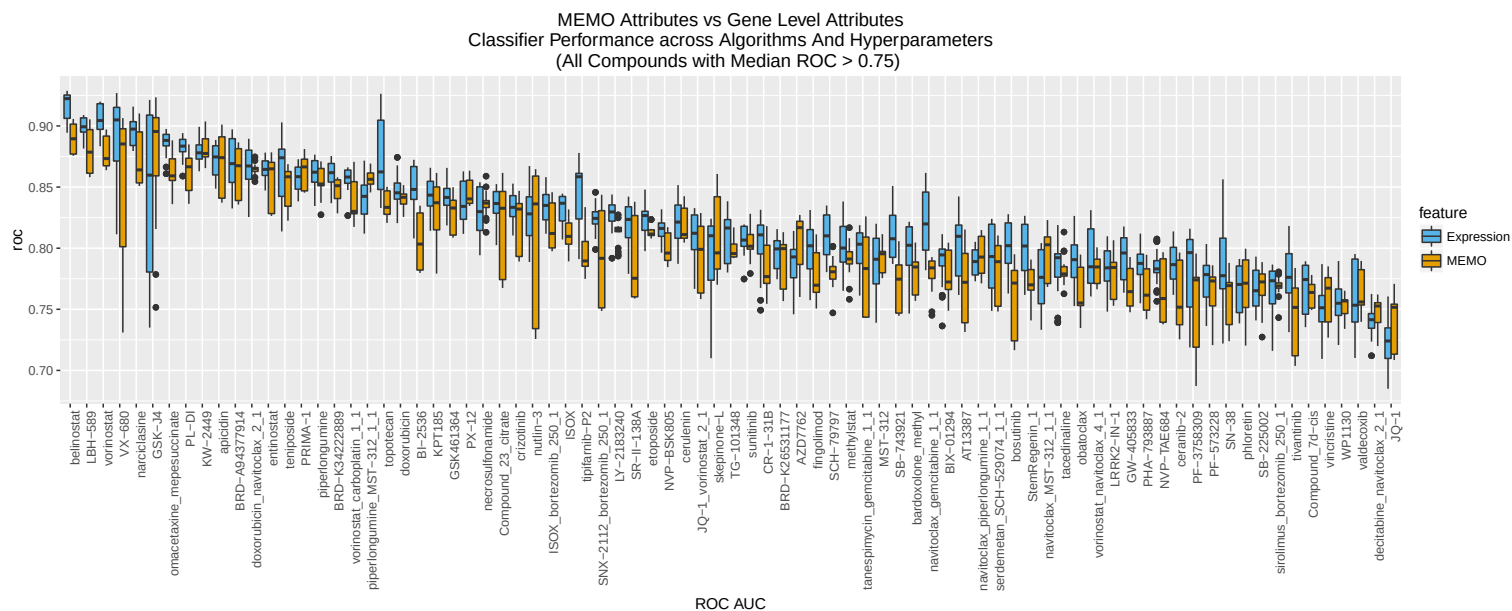


Figure 5.19: Performance of drug sensitivity classifiers built with MEMO semantic attributes. Performance is shown as varying across machine learning algorithms and hyperparameters. Each combination of algorithms and hyperparameters (e.g. point in the distribution) is the result of a 5 times 10x cross-validation experiment. Only compounds with median ROC ≥ 0.75 are shown.

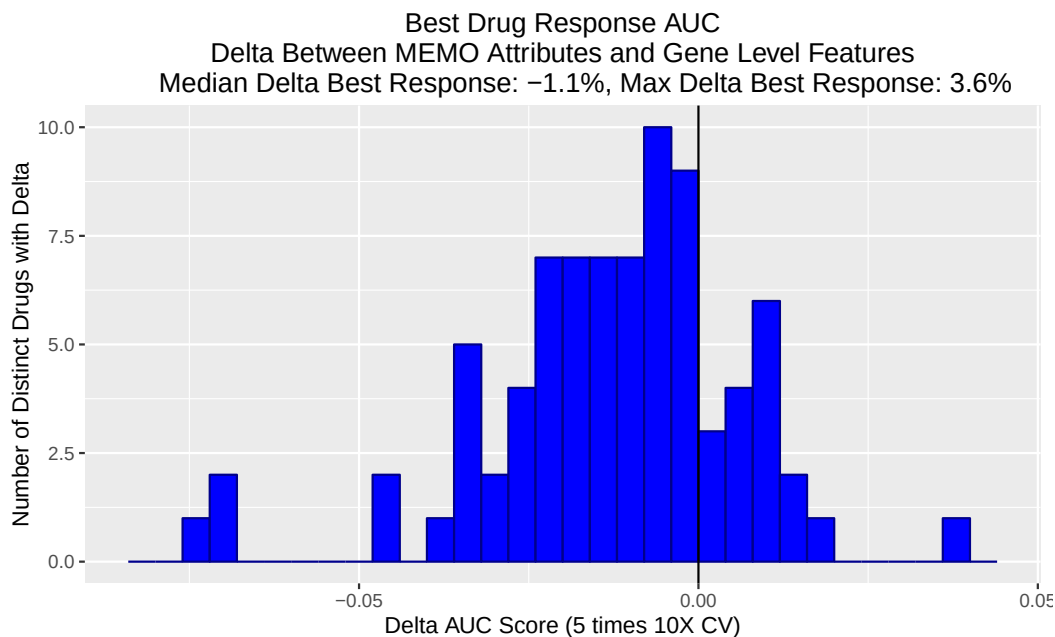


Figure 5.20: Distribution of deltas between the best gene level feature classifier for each drug and the best MEMO semantic attribute transformed classifier for each drug. "Good" classifiers, median ROC ≥ 0.75 , shown.

The MEMO event attribute set consisting of 52 MEMO state classifiers was applied to the gene expression profiles 1036 CCLE samples to obtain a transformed vector for each CCLE sample. Using the 887 dichotomized drug response values as described in 5.1.3. I ran a Weka Mine model selection experiment to gather performance information to compare with gene-level performance. The results are shown in Figure 5.19. The distribution of deltas is shown in Figure 5.20, showing a

-1.1% median difference in scores, which is not an improvement over gene features alone. It's somewhat notable, however, that a mere 52 attributes can produce modestly good classifiers overall.

Mutation Attribute Performance

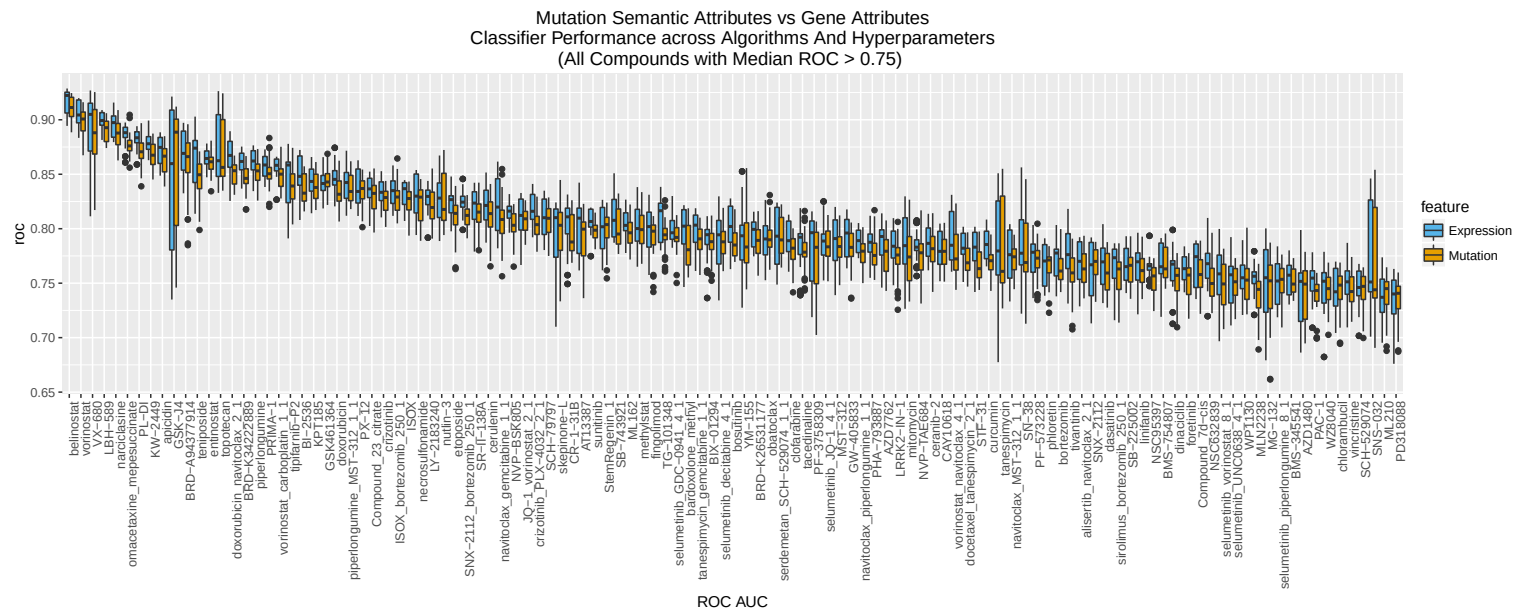


Figure 5.21: Performance of drug sensitivity classifiers built with mutation semantic attributes. Performance is shown as varying across machine learning algorithms and hyperparameters. Each combination of algorithms and hyperparameters (e.g. point in the distribution) is the result of a 5 times 10x cross-validation experiment. Only compounds with median ROC ≥ 0.75 are shown.

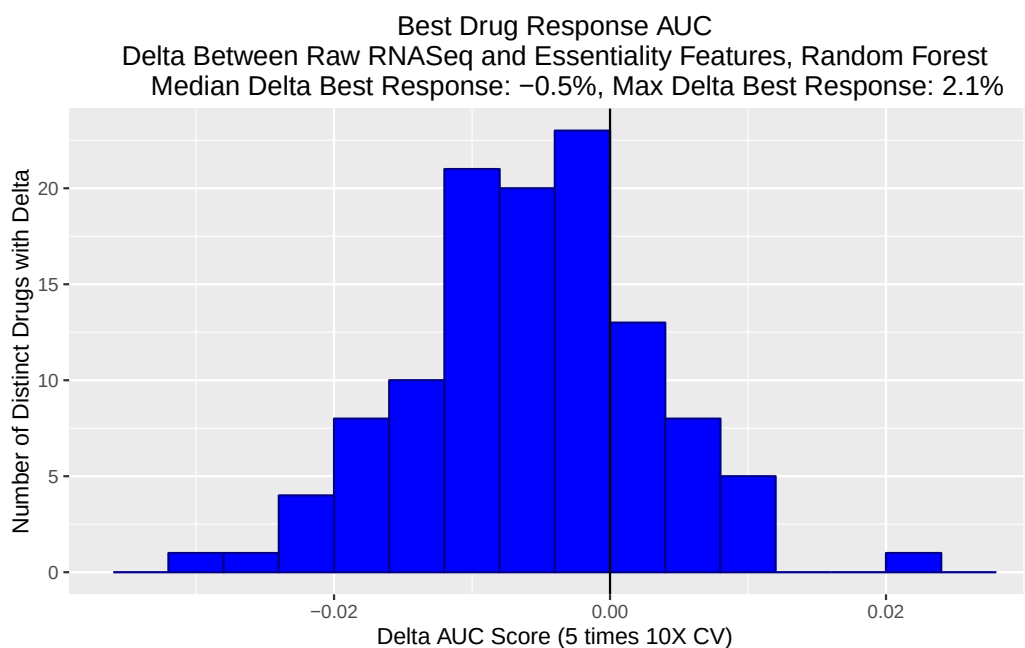


Figure 5.22: Distribution of deltas between the best gene level feature classifier for each drug and the best Mutation semantic attribute transformed classifier for each drug. "Good" classifiers, median ROC ≥ 0.75 , shown.

The mutation event attribute set consisting of 229 mutation state classifiers was applied to the gene expression profiles 1036 CCLE samples to obtain a transformed vector for each CCLE sample. Using the 887 dichotomized drug response values as described in 5.1.3 I ran a Weka Mine model selection experiment to gather performance information to compare with gene-level performance. The results are shown in Figure 5.21. The distribution of deltas is shown in Figure 5.22,

showing a -0.5% median difference in scores, which is not an improvement over gene features alone.

A discussion of how mutation attributes compare to actual mutation data will be presented in a later section.

Chromatin Attribute Performance

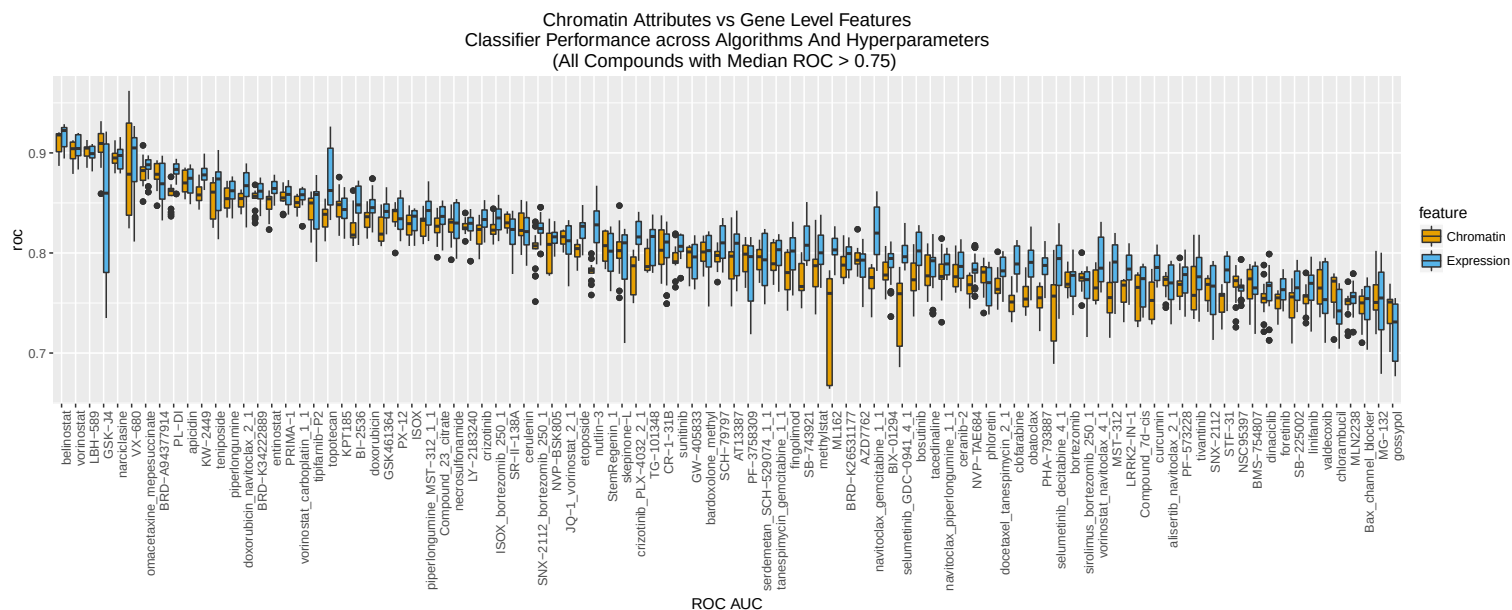


Figure 5.23: Performance of drug sensitivity classifiers built with Chromatin semantic attributes. Performance is shown as varying across machine learning algorithms and hyperparameters. Each combination of algorithms and hyperparameters (e.g. point in the distribution) is the result of a 5 times 10x cross-validation experiment. Only compounds with median ROC ≥ 0.75 are shown.

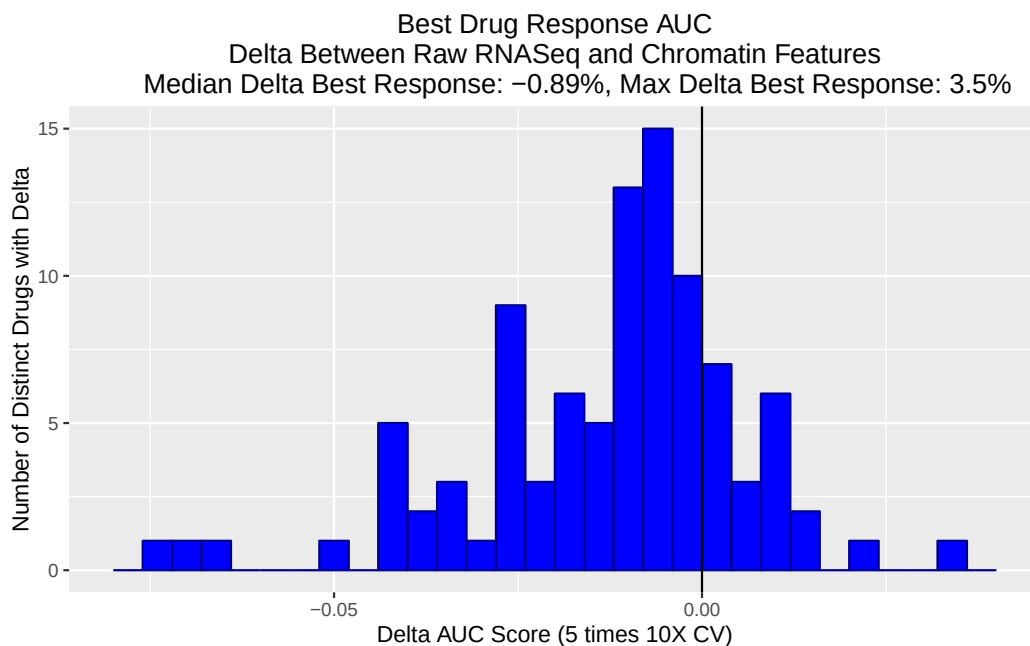


Figure 5.24: Distribution of deltas between the best gene level feature classifier for each drug and the best Chromatin semantic attribute transformed classifier for each drug. "Good" classifiers, median ROC ≥ 0.75 , shown.

The Chromatin attribute set consisting of 1065 chromatin state classifiers was applied to the gene expression profiles 1036 CCLE samples to obtain a transformed vector for each CCLE sample. Using the 887 dichotomized drug response values as described in 5.1.3 I ran a wekaMine model selection experiment to gather performance information to compare with gene-level performance. The results are shown in Figure 5.23. The distribution of deltas is shown in Figure 5.24,

showing a -0.89% median difference in scores, which is not an improvement over gene efeatures alone.

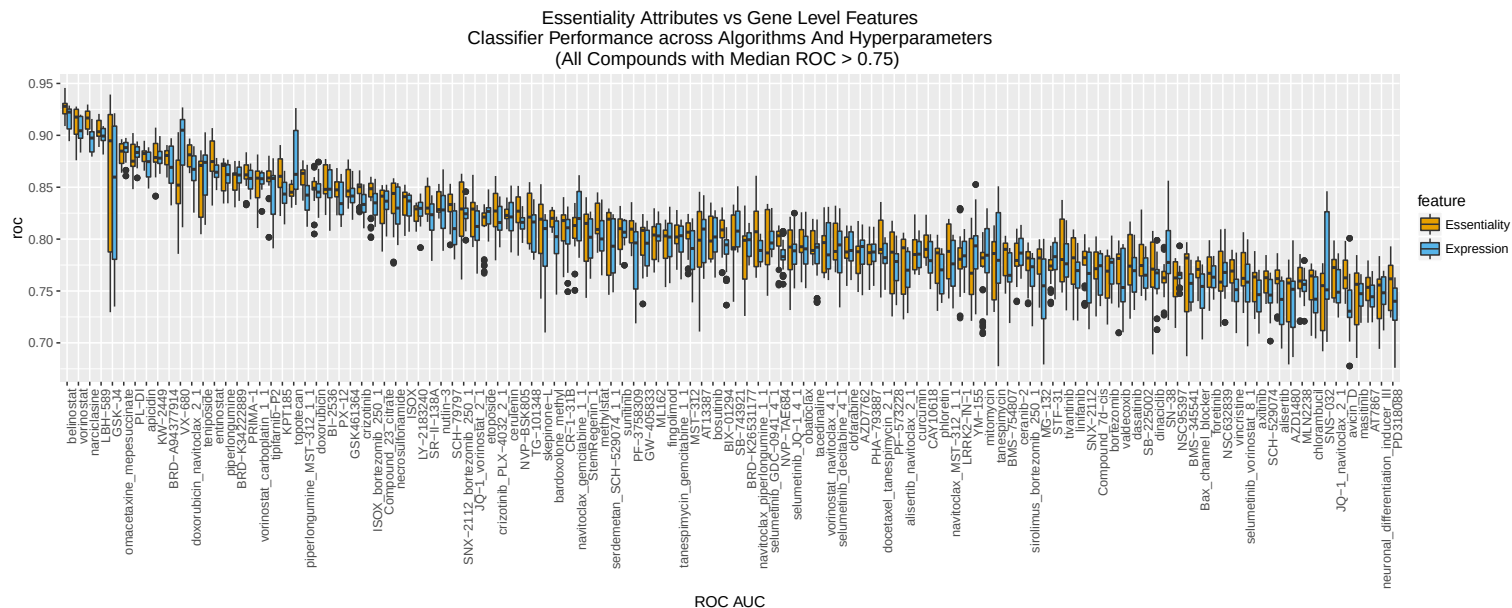


Figure 5.25: Performance of drug sensitivity classifiers built with gene essentiality semantic attributes. Performance is shown as varying across machine learning algorithms and hyperparameters. Each combination of algorithms and hyperparameters (e.g. point in the distribution) is the result of a 5 times 10x cross-validation experiment. Only compounds with median ROC ≥ 0.75 are shown.

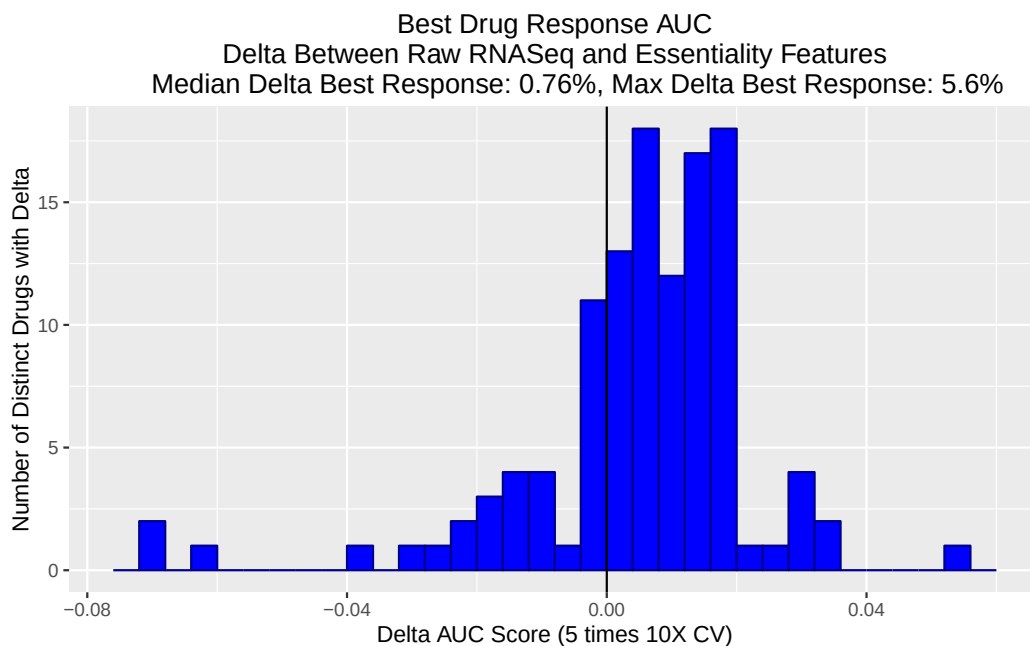


Figure 5.26: Distribution of deltas between the best gene level feature classifier for each drug and the best Chromatin semantic attribute transformed classifier for each drug. "Good" classifiers only shown.

Essentiality Attribute Performance

The gene essentiality attribute set consisting of 4307 gene essentiality classifiers was applied to the gene expression profiles 1036 CCLE samples to obtain a transformed vector for each CCLE sample. Using the 887 dichotomized drug response values as described in 5.1.3 I ran a wekaMine model selection experiment

to gather performance information to compare with gene-level performance. The results are shown in Figure 5.25. The distribution of deltas is shown in Figure 5.26, showing a 0.76% median improvement in scores. The essentiality panel was the only one to show an overall improvement on it's own, and it's improvement is the same order of magnitude of the improvement in the COMBINED set, suggesting that essentiality is the major driver of COMBINED performance. Nonetheless, the COMBINED set does have a larger improvement than essentiality alone, so it seems that there is some synergistic effect between the essentiality attributes and some of the other attributes. In a later section I will examine the actual features used by classifiers in the most improved drug classifiers.

5.1.6 Essentiality And Drug Targets

Essentiality Contribution to Specific Targets

Most of the compounds in the drug prediction set have known or suspect targets and method of activities. The targets and activities were compiled for each of the drug targets from ?? and can be found in Appendix D. An interesting question to explore is to see how genes whose essentiality helps predict drug sensitivity relate to drug targets or method of activities.

To examine this I map the essentiality genes used in each drug classifier to the targets for that drug. All total there are 154 compound classifiers. Removing compound combinations leaves 115 single compound classifiers. As a measure of the strength of association of each essentiality gene with the drug classifier I extract the essentiality attributes from each classifier along with their weights within that classifier. In order to make the weights comparable I restrict this analysis to random forest classifiers. As an aside, it is notable that among gene essentiality classifiers the best classifier is most often random forest (specifically 20

logistic regression, 8 SVM, and 126 Random Forest), whereas with gene expression based classifiers the best classifier is most often an SVM (specifically 49 logistic regression, 85 SVM, 25 Random Forest). Speculatively, this may be because gene essentiality attributes provide information that better splits the dataset serially along single features, whereas gene expression must rely on relations among many genes at once to obtain a separating hyperplane.

Restricting the analysis to random forest classifiers leaves 91 out of the 115 classifiers of single compounds. Over these 91 classifiers, there are 78 activities/-targets.

For each of 78 activities, I examined drugs that mapped to that activity and extracted the actual classifier weights for each drug essentiality attribute. When there are multiple drugs that map to the same activity, the average of the attribute weights across drugs is taken. The resulting table of gene essentiality attributes by targets/activities was then clustered with hierarchical clustering (using correlation as a similarity measure and average linkage).

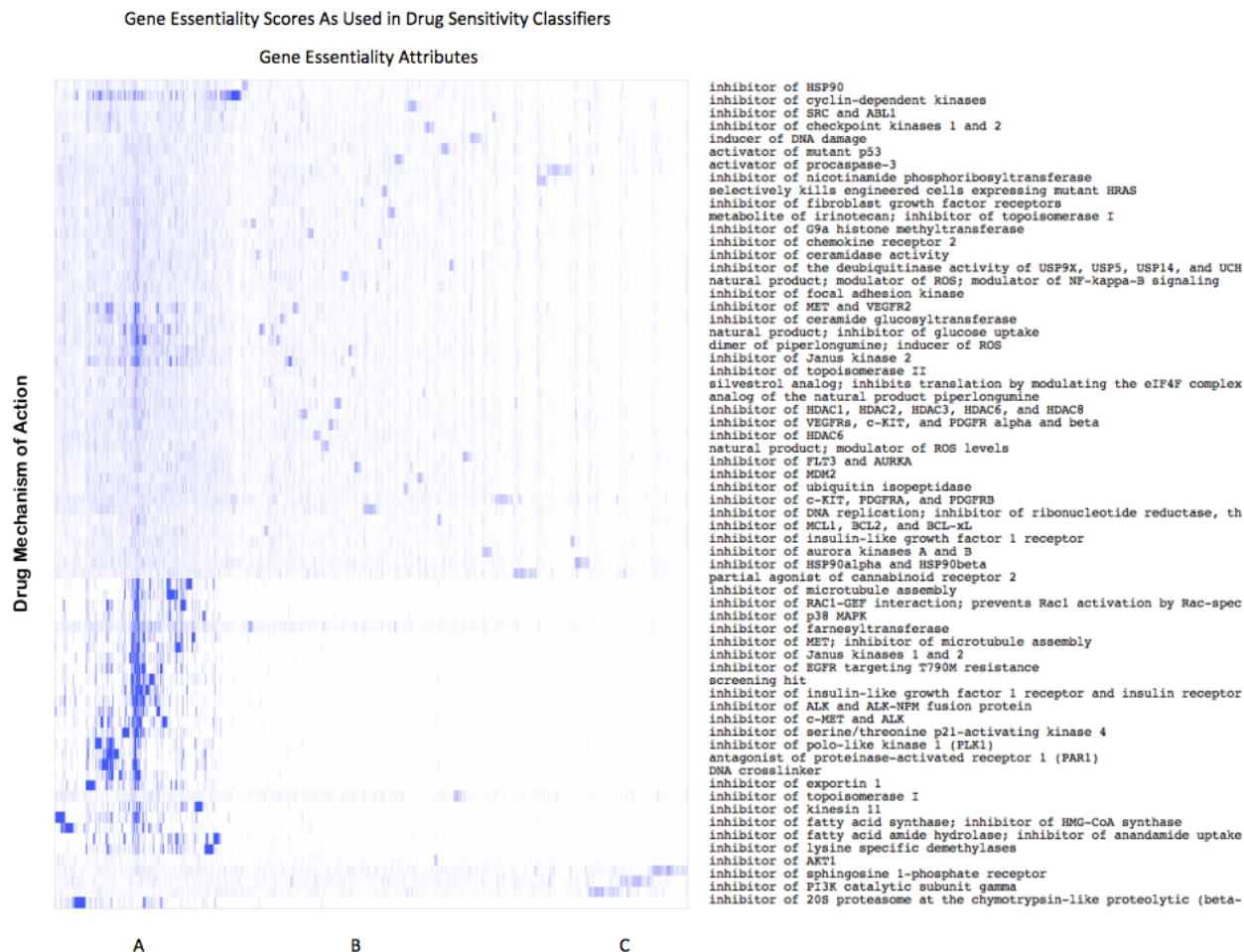


Figure 5.27: Heatmap showing how much each gene essentiality attribute contributes to the drug classifiers for a given method of activity. The region in A includes genes centered on the MYC pathway involved in transcription and proliferation.

A plot of this clustering, showing essentiality genes in columns and putative method of action in rows can be seen in Figure 5.27. The plot shows how much each gene essentiality attribute contributes to the drug classifiers for each method of activity. A number of obvious associations are apparent in this plot. Glucose uptake activities appear in clusters together, histone deacetylases appear in nearby arms, AURKA is closely associated with aurora kinase B, proteasome inhibitors are together, fatty acid processing inhibitors are together, and ALK activities cluster together as well. The region in A includes genes centered on the MYC pathway, see Figure ??, involved in transcription and proliferation and is broadly useful in predicting drug sensitivity across many different activities. There is evidence in this plot in sections B and C of some specific modules tied to specific mechanisms of activity, but initial pathway activities for these collections of genes were weak so the nature of these modules is not yet clear.

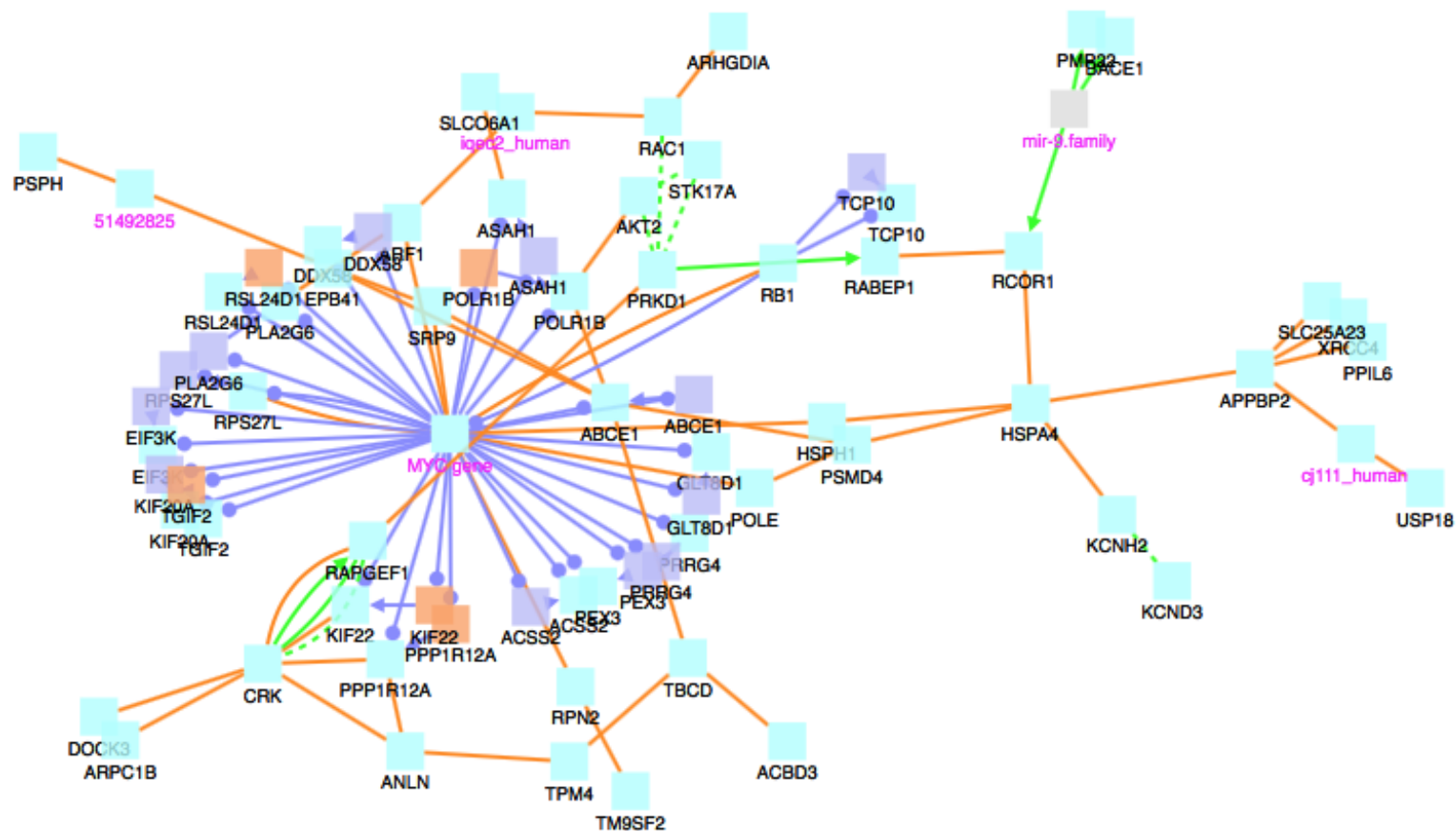


Figure 5.28: Pathway enrichment for essential genes that are useful across most activities (section A in previous plot). This pathway is centered on MYC and is involved in general in transcription and cell proliferation.

5.1.7 Chromatin Features And Drug Targets

Semantically transformed expression data can form the basis of a good classifier in a couple of ways. One is simply to provide a collection of vectors in high dimensional space from which various partitioning hyperplanes can be constructed. With a large collection of such vectors one can almost always construct classifiers with performance above chance even if the vectors are not connected to the target in any important way. Another is for the transformation to include subtle but distributed information about the target class, information that helps boost performance but which is spread thinly over many vectors and so hard to identify. The final, and most interesting way that semantic transformations may be useful is if the semantic content of the classifier is actually determinative. That is, if it's actually the case that presence of stem cells, or the similarity of cancer cells to stem cells, is a driving factor behind the target class of, say, survival.

While analyzing this question quantitatively is beyond the scope of this thesis, we can nonetheless take a look at some of the features in our drug sensitivity classifiers to get a sense whether semantic content of the features might be involved in the performance improvements we see.

Most Informative Individual Chromatin Features

Among the semantic features, in some ways the most interesting are chromatin features, since those go directly to the regulatory state of the cell. These features are also the most heavily processed of the features in the COMBINED set of features, which were derived from a series of simplifying assumptions. To explore this in an anecdotal way, I computed the information gain of each semantic feature across all drugs and made a sorted list of the most informative features among the "good" (median roc >0.75) classifiers. Looking at the top 15 non-redundant

Drug	Chromatin Attribute	Drug Classifier ROC	Information Gain
VX-680	B_chr3_150680000_RepressedPolycomb	0.9108	0.746
GSK-J4	B_chr13_24480000_Quiescent	0.9203	0.704
VX-680	B_chr11_92320000_RepressedPolycomb	0.9108	0.664
VX-680	B_chr1_95080000_Quiescent	0.9108	0.664
VX-680	B_chr8_134280000_Quiescent	0.9108	0.664
VX-680	B_chr11_66560000_Quiescent	0.9108	0.664
VX-680	B_chr11_100800000_Quiescent	0.9108	0.664
belinostat	B_chr12_63440000_Quiescent	0.939	0.517
narciclasine	B_chr10_23440000_Quiescent	0.9211	0.465
vorinostat	B_chr11_9520000_Quiescent	0.9203	0.457
vorinostat	B_chr1_68000000_Quiescent	0.9203	0.456
vorinostat	B_chr10_72320000_Quiescent	0.9203	0.454
apicidin	B_chr8_118400000_StrTranscription	0.8923	0.424
teniposide	B_chr1_110400000_RepressedPolycomb	0.8841	0.42
tipifarnib-P2	B_chr3_158360000_Quiescent	0.8692	0.407

chromatin attributes gives us the following table:

It's notable that all of these highest information gain chromatin features are associated with well performing classifiers, 0.87-0.939 roc.

Genomic Regions of Most Informative Chromatin Features

So what kinds of genes are in the regions identified by these chromatin state classifiers? In the USCS Human Genome Browser (hg19), we see in Figure 5.29 a 200Kbp region around chr3:150680000. This region corresponds to a chromatin semantic attribute that, by itself, provides 0.746 bits of information towards predicting VX-680 drug sensitivity. In this region are two genes, Homo sapiens clarin 1 (CLRN1), which is associated with eye disorders, and Homo sapiens mediator complex subunit 12-like (MED12L), which is involved in transcriptional coactivation of nearly all RNA polymerase II-dependent genes [99]. It could be that this feature is merely coincidentally useful, but it's also possible that it represents genuine information about the regulatory state of this region that is important to drug responsiveness. The possibly wide range of effects regulation of MED12L may have are at least plausibly consistent with the idea that regulation of this region affects drug responsiveness.

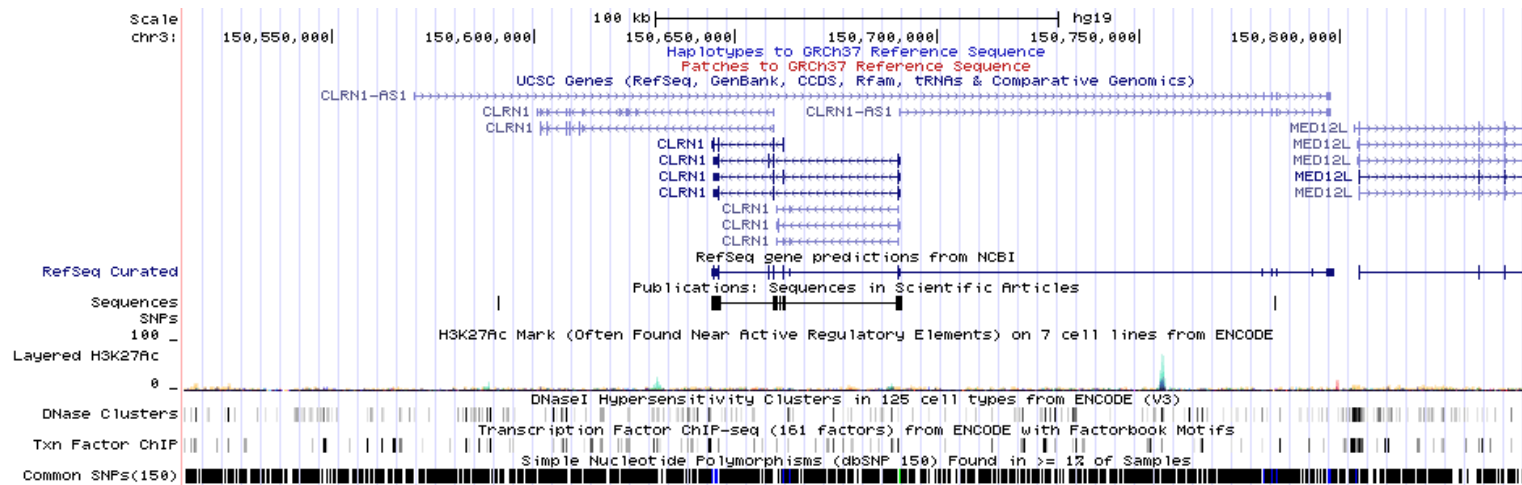


Figure 5.29: Roughly 200Kbp region around chr3:150,680,000 showing two genes. This region corresponds to a chromatin semantic attribute that, by itself, provides 0.746 bits of information towards predicting VX-680 drug sensitivity.

Looking at the next region, a feature that by itself provides 0.704 bits of information about the sensitivity of the compound GSK-J4, we can examine chr13:24,480,000. A 400Kbp region around this point is illustrated in Figure 5.30.

In this region we see five genes (and one pseudo-gene) two of which are implicated in cell-death/apoptosis pathways, specifically Homo sapiens tumor necrosis factor receptor superfamily, member 19 (TNFRSF19), and Homo sapiens C1q and tumor necrosis factor related protein 9B (C1QTNF9B). Both of these genes are capable of inducing apoptosis. [99]. The plausible connection between regulation of this region and drug sensitivity, that is the survival and proliferation of cells under treatment with the compound, seems easy to imagine. This is only suggestive, of course, but it lends some credibility to the idea that these semantic features actually reflect information about the concept they attempted to capture, namely chromatin mediated regulation of genomic regions.

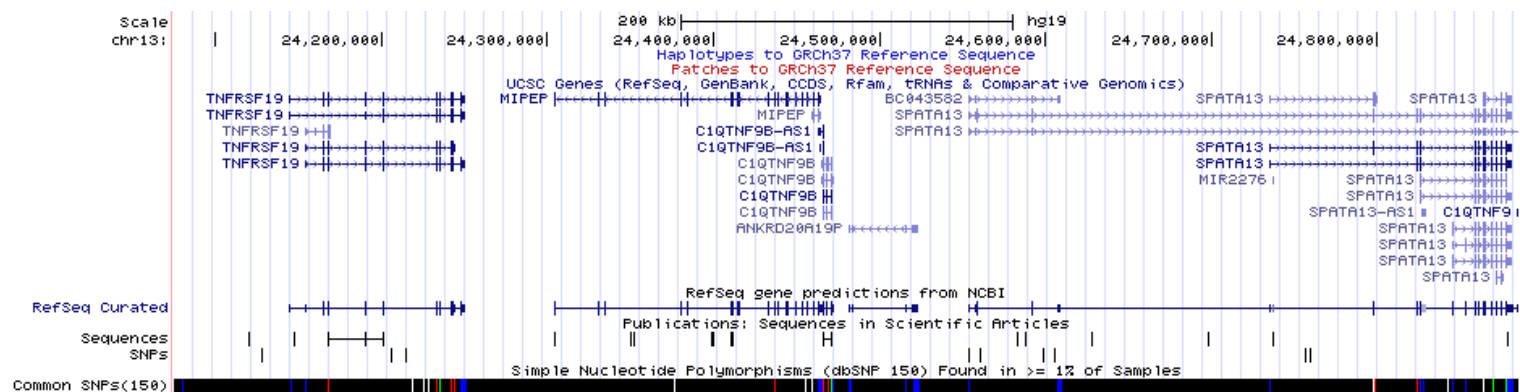


Figure 5.30: Roughly 400Kbp region around chr13:24,480,000 showing six genes. This region corresponds to a chromatin semantic attribute that, by itself, provides 0.704 bits of information towards predicting GSK-J4 drug sensitivity. Two of these genes are apoptosis related.

Skipping a couple of features, whose regions contain genes for FAT3, ABCD3 (an ATP-binding cassette), and F3 (a coagulation factor), we can look at chr8:134,280,000, a feature that provides 0.664 bits of information toward predicting VX-680 responsiveness. A 100Kbp region around this location is shown in Figure 5.31 showing two genes. One of these genes, Homo sapiens WNT1 inducible signaling pathway protein 1 (WISP1), "may be downstream in the WNT1 signaling pathway that is relevant to malignant transformation. It is expressed at a high level in fibroblast cells, and overexpressed in colon tumors". The other, Homo sapiens N-myc downstream regulated 1 (NDRG1), is necessary for p53-mediated apoptosis[99]. Once again, this is merely suggestive, but this is another set of genes among top chromatin attributes whose connection to cell proliferation and cell death seems clear.

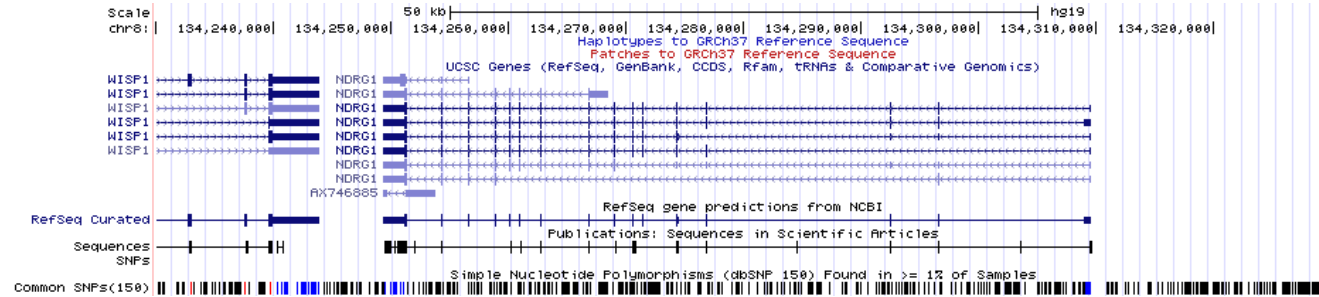


Figure 5.31: Roughly 100Kbp region around chr8:134,280,000 showing two genes. This region corresponds to a chromatin semantic attribute that, by itself, provides 0.664 bits of information towards predicting VX-680 drug sensitivity. One of these genes (WISP1) is involved in the WNT1 pathway involved in malignant transformation, the other, NDRG1 is necessary for p53 mediated apoptosis.

Skipping down further, to another drug entirely, we see that the feature chr12:6,344,000 Quiescent is a feature that provides 0.52 bits of information toward the most accurate drug classifier of all, belinostat (roc=0.939). In this region we find Homo sapiens CD9 molecule (CD9). "The encoded protein functions in many cellular processes including differentiation, adhesion, and signal transduction, and expression of this gene plays a critical role in the suppression of cancer cell motility and metastasis"[99].

Looking further, at narciclasine, chr10:23440000 Quiescent, we find Homo sapiens phosphatidylinositol-5-phosphate 4-kinase, type II, alpha (PIP4K2A), "involved in the regulation of secretion, cell proliferation, differentiation, and motility", and Homo sapiens armadillo repeat containing 3 (ARMC3). "ARM domain-containing proteins, such as ARMC3, function in signal transduction, development, cell adhesion and mobility, and tumor initiation and metastasis"[99].

And still further on, vorinostat, chr1:68,000,000 Quiescent, where we find Homo sapiens interleukin 23 receptor (IL23R), which is a member of a group of genes whose transcript levels are increased following "stressful growth arrest conditions and treatment with DNA-damaging agents"[99].

For teniposide, chr1:110,400,000 RepressedPolycomb, there is in that region Homo sapiens EPS8-like 3 (EPS8L3), a gene that "encodes a protein that is related to epidermal growth factor receptor pathway", and Homo sapiens glutathione S-transferase mu 5 (GSTM5), one of a class of enzymes that "functions in the detoxification of electrophilic compounds, including carcinogens, therapeutic drugs, environmental toxins and products of oxidative stress, by conjugation with glutathione"[99].

And finally consider, tipifarnib-P2, with feature chr3:158,360,000 Quiescent. At this location is the gene Homo sapiens myeloid leukemia factor 1 (MLF1). This

oncoprotein is thought to be involved in the determination of hemopoetic cells [99]. The drug Tipifarnib, as it turns out, is a drug that is used in the treatment of acute myeloid leukemia [Thomas:wk]. All of these snapshots are anecdotal in nature, and so do not constitute proof that the semantic content of these features is driving the performance of these classifiers, but they are suggestive. Especially given the indirect nature of the association (chromatin mark->chromatin state-> classifier for chromatin state-> gene in region associated with drug response).

It is an obvious direction for future work to try to assess the degree to which the specific semantic content of individual features drives predictor accuracy or whether the suggestive associations seen here are merely coincidental.

Chromatin Features of VX-680

The targets and activities were compiled for each of the drug targets from ?? and can be found in Appendix D. The drug classifier with the most information gain across various target attributes is VX-680, as seen in Table 5.1.7.

VX-680 is a compound that has been found to be an inhibitor of the Aurora kinases [100]. Aurora-A was "first isolated as the product of gene BTAK (breast tumor amplified kinase, also named STK15) on chromosome 20q13, a region that is frequently amplified in breast, colorectal, and bladder tumors as well as ovarian, prostate, neuroblastoma, and cervical cancer cell lines" [101][102].

An aurora kianase A inhibitor will have little effect in cells that are not actually expressing Aurora kinase A, so it's to be expected that transcription of aurora kianase is a strong indicator of the potential effectiveness of an AURKA inhibitor. AURKA is found on chromosome 20q13 at genomic region chr20:54,944,445-54,967,351 (all coordinates are with reference to UCSC hg19 assembly). Nearby, also on 20q13, is a highly informative chromatin attribute for VX-680, chr20, location

56240000, StrTranscription. This attribute alone conveys 0.53 bits of information towards VX-680 sensitivity classification. Strong transcription of this region likely means strong transcription for the nearby AURKA region, which is consistent with the idea that this chromatin feature could be a proxy for the positive transcriptional regulation of AURKA. Related, immediately by this region, at location chr20:56,223,448-56,265,680, is the gene PMEPA1. PMEPA1 is itself overexpressed in some colon, breast, and ovarian cancers and along with AURKA, appears to be involved in chromosomal instability in certain cancers [Sheffer:wy].

AURKA is embedded in a network of activators and repressors. We would expect that the absence of transcriptional activation for repressors for AURKA and the presence of transcriptional activation of activators of AURKA to be markers for VX-680 sensitivity. In normal cells, APC down-regulates Aurora-A [102]. APC is located at chr5:112,073,556-112,181,936 and there is a nearby chromatin attribute for VX-680 sensitivity B_chr5_111880000_Quiescent, which carries 0.21 bits of information. Similarly, Cdh1 is required for the downregulation of AURKA, and in its absence AURKA transcription levels should rise. Cdh1 is located at chr16:68,771,195-68,869,444, and there is a nearby chromatin feature in the VX-680 drug predictor, B_chr16_68840000_Quiescent, which carries 0.53 bits of information about VX-680 sensitivity. Both of these repressor genes are in regions where the VX-680 feature is for a Quiescent state, which is consistent with the hypothesis that silent repressors would indicate VX-680 sensitivity.

Lack of Specificity in Chromatin Features

While these positive associations between specific chromatin features and drug sensitivity are encouraging, excitement over the interpretability of chromatin semantic attributes may be premature. Chromatin attributes generally outperform

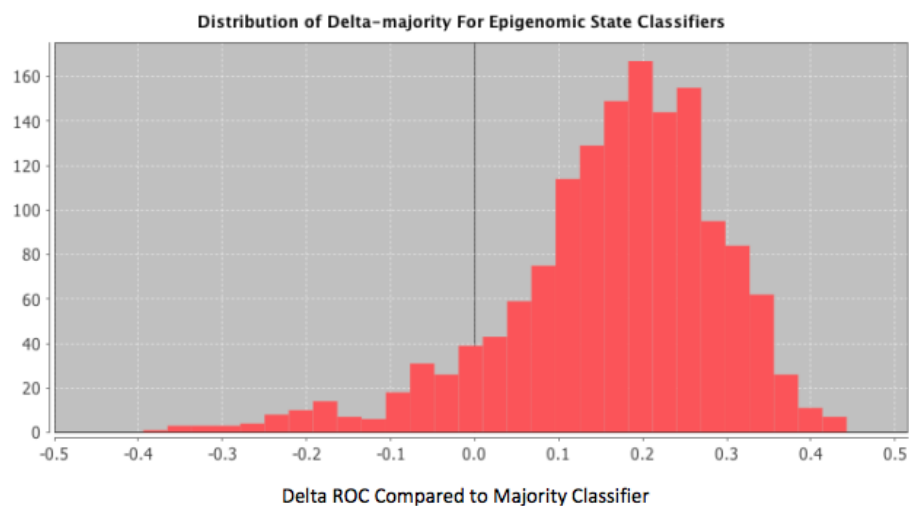


Figure 5.32: Chromatin attributes generally outperform majority classifier by large margins, with most over 20% better than guessing. Whatever real world state these attributes represent, it is easily predicatable from gene expression data.

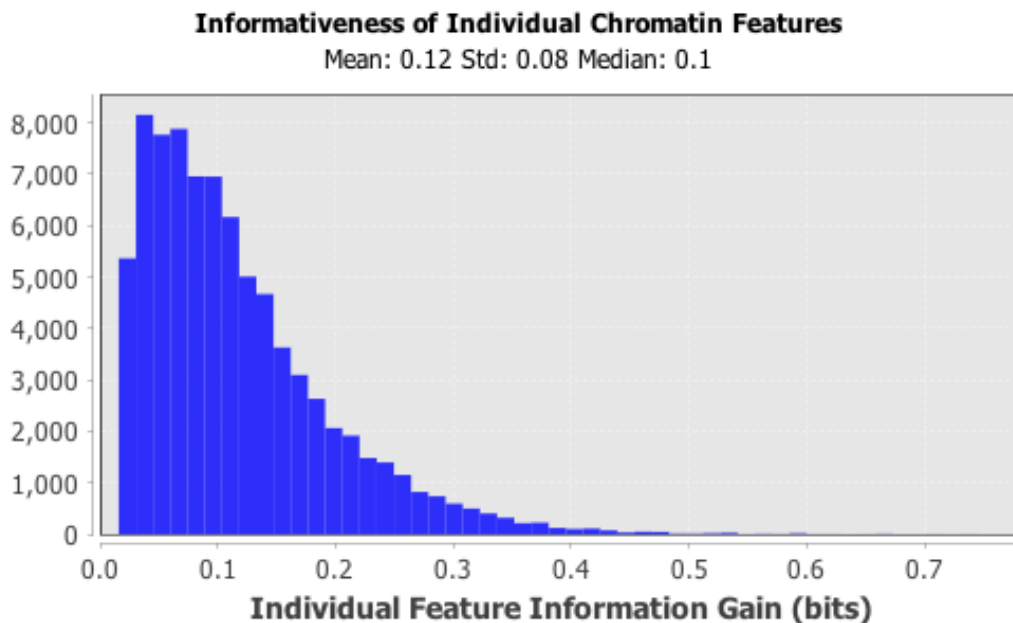


Figure 5.33: In addition to being accurate classifiers, chromatin attributes generally convey a lot of bits of information toward their drug targets.

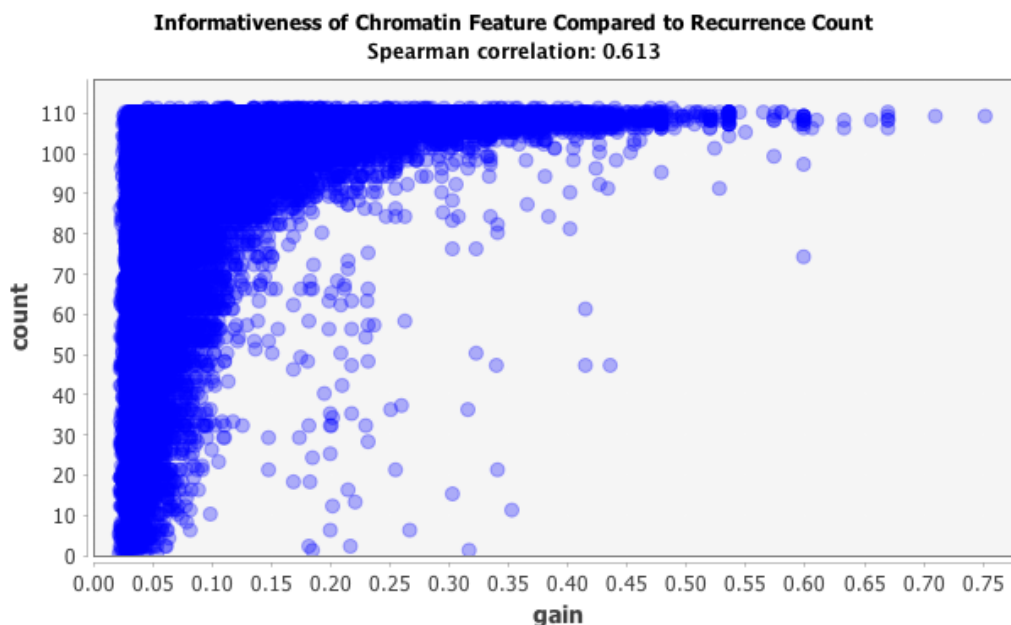


Figure 5.34: There is a reasonably strong correlation between the amount of information each chromatin attribute conveys toward a target and how many different drug sensitivity classifiers it appears in.

majority classifier by large margins, with most over 20% better than guessing as can be seen in figure Figure 5.32. Whatever real world state these attributes represent, it is easily predicatable from gene expression data. In addition to being accurate classifiers, chromatin attributes generally convey a lot of bits of information toward their drug targets, as seen in Figure 5.33. Moreover, informative chromatin attributes tend to recur across drug targets Figure 5.34. Also, the informativeness of individual attributes is well correlated with the performance of drug classifiers it's a part of, Figure 5.35. All of this is good news for chromatin features in drug prediction. The bad news here is that most chromatin attributes that appear in any of these drug classifiers appear in almost all of these drug classifiers. This implies that the specificity of chromatin attributes is quite poor. Seeing a semantic attribute for strong transcription of the AURKA region is exciting, since that's exactly what one would expect as an informative feature for

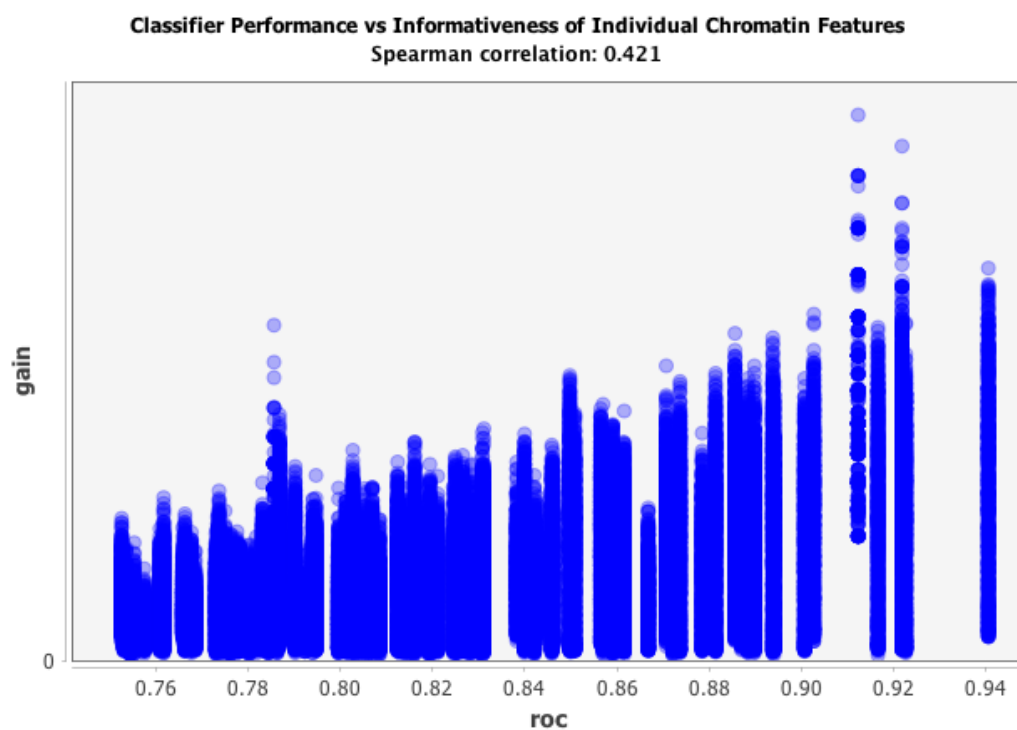


Figure 5.35: There is a significant correlation between the informativeness of individual chromatin attributes and the overall performance of the drug classifier, though there is a large spread at all performance levels.

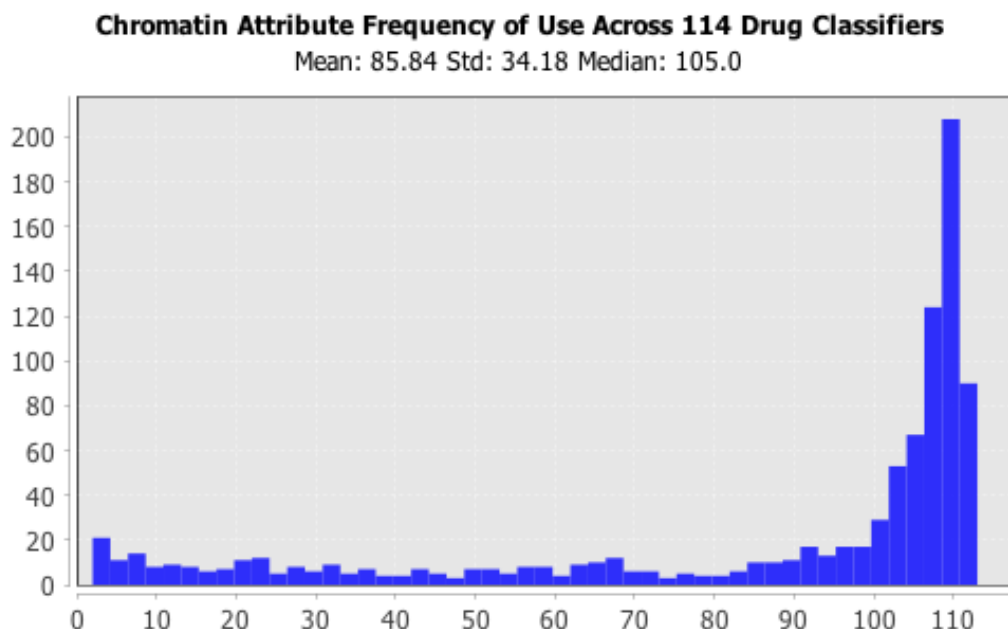


Figure 5.36: Most chromatin attributes appear in most drug classifiers. This implies that the specificity of chromatin attributes is quite poor.

VX-680, an AURKA inhibitor, to be. However, this same feature occurs in 112 of the 115 drug classifiers, with a mean bits of information of 0.2 bits!

While these features seem to be useful in general way, contributing to the modest performance improvements seen, they are not specific enough to the drug target to be interpretable in the way that the semantic attribute hypothesis posited. The implications for work going forward will be discussed in the conclusion.

5.1.8 Mutation Attributes vs Mutations

In this section I investigate the degree to which mutation attributes capture the actual mutation state of CCLE samples and how their performance predicting drug response for samples compares to using actual mutation data.

At the time of writing this section cluster resources were unavailable to me to perform a normal sized wekaMine model experiment, so for the purposes of this

section I performed a truncated experiment. In the truncated experiment I look at only the top nine drugs from mutation chaining and compared their performance as semantic attributes for this drug prediction task to the performance of actual mutation data only, expression data enriched with actual mutation data, expression data enriched with semantic attributes, and finally expression only. In this truncated experiment I examine only random forest and linear SVMs using fixed hyper-parameters.

Mutation Attribute Precision/Recall

For the CCLE actual mutation data I obtained the mutation data MAF file from ?? and processed this into a mutations by cell line table. I applied the 229 mutation models to 841 CCLE cell line samples to obtain a comparison set of mutation calls. For each target mutation I then computed the precision and recall of the mutation attribute relative to the actual mutation calls for that cell line.

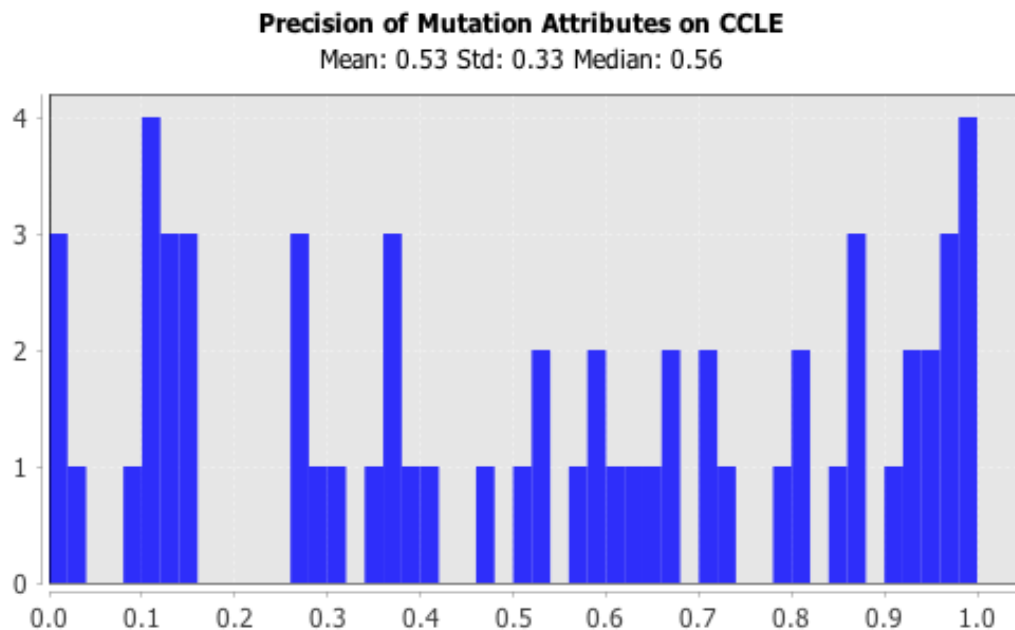


Figure 5.37: Precision of mutation attributes on CCLE relative to actual mutation calls.

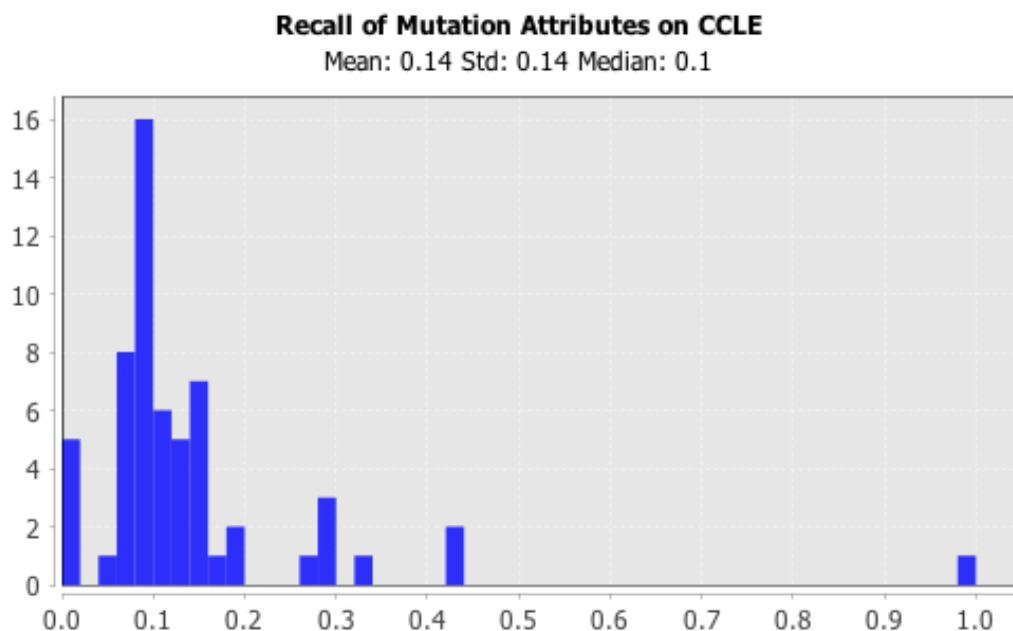


Figure 5.38: Recall of mutation attributes on CCLE relative to actual mutation calls

The results of this analysis can be seen in Figure 5.37 and Figure 5.38. As this shows, the precision of the mutation classifiers is often adequate and sometimes good, but the recall is pretty much uniformly dismal. Although I have conducted cross-tissue mutation experiments with convincing results for some mutations, it may be that batch effects or other complications across datasets and tissues make accurate mutation prediction difficult or that the good examples are unusually easy to detect.

This highlights a general difficulty and potential pitfall of semantic attributes. Without a large and diverse corpus to train semantic attributes, it may be easy to train semantic attributes that perform well with respect to the assigned label but which do not actually directly carry the semantic content of the label. A "TP53 mutation" classifier might be an accurate classifier for the assigned TP53 label in training data, but may actually be keying off of something that is merely

correlated with that label within the dataset. This possibility makes interpreting semantic attributes, one of the desired payoffs of this approach, harder to achieve.

Actual Mutations as Attributes in Drug Sensitivity

So how do actual mutations perform as features in drug sensitivity prediction compared to mutation attributes? To examine this I performed a truncated model selection experiment on the top nine drugs from mutation attribute chaining and also on raw mutations alone. The results are shown in figure Figure 5.39. Clearly, semantic mutation features alone perform vastly better than mutations alone. This is not surprising because semantic mutation features take into account a wide range of gene features as inputs before transforming them into a score. The semantic attribute chained matrix is dense, since every single mutation provides some score that is generally non-zero. One way to think of it is as similarity matrix, where every element measures, in some sense, the similarity of a particular sample to the training samples that have the given mutation. This set of dense vector projections into gene space provides lots of opportunities for classifier algorithms to find combinations of those vectors that partition the space along drug sensitivity lines, resulting ultimately in fairly decent classifiers. In contrast, the actual mutation matrix is sparse, consisting mostly of zeros representing mutations that are not present in a given sample. Such a sparse matrix contains a small amount of total information and doesn't provide much flexibility for constructing boundary hyperplanes and results in classifiers that are too poor to be of much practical use. It is remarkable, however, that even with this sparse matrix classifiers are able to beat guessing by 2-11%. Note that belinostat failed in the actual mutation case because there were too few mutations in their samples.

Mutation Semantic Attribute				Actual Mutation			
drug	majority	roc	delta	attribute	majorityAcc	roc	delta
belinostat	0.564	0.899	0.335	GSK-J4	0.568	0.64	0.072
GSK-J4	0.523	0.854	0.331	KW-2449	0.504	0.569	0.064
KW-2449	0.51	0.886	0.376	LBH-589	0.535	0.56	0.024
LBH-589	0.535	0.893	0.358	narciclasine	0.518	0.556	0.038
narciclasine	0.507	0.899	0.393	PL-DI	0.508	0.527	0.019
PL-DI	0.518	0.867	0.349	topotecan	0.5	0.546	0.046
topotecan	0.5	0.846	0.346	VX-680	0.529	0.642	0.113
vorinostat	0.521	0.895	0.374				
VX-680	0.514	0.895	0.381				

Figure 5.39: Semantic mutation attributes alone compared to actual mutation calls alone as a basis for building drug sensitivity classifiers.

Next I compared combining chained mutation attributes with expression data and combining actual mutations with expression data. As can be seen, adding expression data improves performance over either mutation attributes or actual mutations alone, though there is not much difference between actual mutations and chained mutations conditions.

Expression + Mutation Semantic Attribute				Expression + Actual Mutations			
attribute	majorityAcc	roc	delta	attribute	majorityAcc	roc	delta
belinostat	0.564	0.93	0.366	belinostat	0.564	0.932	0.368
GSK-J4	0.523	0.928	0.405	GSK-J4	0.523	0.915	0.392
KW-2449	0.51	0.895	0.385	KW-2449	0.512	0.898	0.386
LBH-589	0.535	0.917	0.382	LBH-589	0.535	0.913	0.378
narciclasine	0.507	0.92	0.414	narciclasine	0.508	0.922	0.414
PL-DI	0.518	0.898	0.38	PL-DI	0.518	0.894	0.376
topotecan	0.5	0.864	0.364	topotecan	0.5	0.866	0.366
vorinostat	0.521	0.921	0.4	vorinostat	0.52	0.918	0.398
VX-680	0.514	0.93	0.416	VX-680	0.514	0.924	0.41

Figure 5.40: Expression plus semantic attributes and expression plus actual mutations.

Chapter 6

Aim 3: Sample Psychic

Sample Psychic is a web application created to provide an easy way for researchers to utilize the many classifiers created in this work. With Sample Psychic, large sets of classifiers can be applied to uploaded data (RNASeq or microarray) in order to gain various insights into the states of the cells represented in the data. In addition to allowing one to explore the results of these classifications online it also provides access to semantic attributes for further work, allowing one to download vectors of classifier probabilities for each uploaded sample for use in further analysis. Sample Psychic is currently being used to provide a "report card" feature for single cell data from the CIRM Brain of Cells single cell brain development initiative. Some of the major features of Sample Psychic include:

- Tight integration with Weka Mine.
- Able to apply thousands of classifiers to data.
- Classifiers organized into sets.
- Able to explore tens of thousands of classifications.
 - Where each classification fits on a background distribution.

- Genes and their weights used in each classifier
- Information about those genes
- Ability to filter results in various ways.
- Ability to download classifier result vectors.
- Clustering and visualtion of classification vectors with t-SNe
- Ability to bookmark results in web browser.
- Ability to produce results off-line and view in browser.

6.1 Design

Weka Mine was designed with a web application like Sample Psychic in mind. In particular, Weka Mine was designed to isolate detailed knowledge of processing steps from the Web Application, allowing changes in models or addition of new models to occur without having to touch any Sample Psychic code. It achieves this by encapsulating all the information needed to apply a classifier to new samples in a single file, the Weka Mine model file (.wmm file), separating those algorithm details from the design, or even knowledge, of the Sample Psychic web app. This file includes the unsupervised filters used, the normalization used to process the data, the set of attributes used by the classifier (and code to reconcile those attributes with uploaded data that may not have the same number of features), the classifier itself, which may be one of a range of classifiers, the classifier hyper parameters, and a set of background classifications either from an external background dataset or from a label-permuted bootstrap background (These background samples can be used as an independent measure of the strength of a classification result from

the built in score, reported as a probability, that each classifier algorithm individually returns.)

For the purposes of Sample Psychic Weka Mine model files are grouped together into "signature sets". Signature sets provide a logical grouping of signatures by category, but signature sets are also usually derived from the same training data and share the same background samples used for the background null model. For example, the 158 individual CCLE drug classifiers were all trained with CCLE RNASeq data and all evaluated on 2213 randomly selected TCGA PANCAN 4.5 samples for a background null model. Weka Mine was designed from the ground up to make model selection and training of a signature set as easy as model selection and training for a single classifier. Sample Psychic was designed so that simply dropping a directory of .wmm files into a directory is all that is required to add signatures.

To illustrate, taking a 545 x 887 drugs x samples file it is possible to perform model selection on 548 drug sensitivity measures building 158 classifiers from the best combination and then deploying them to Sample Psychic involving only the following sequence of commands (some options omitted for readability):

```
wmDichotomize -c ec50_compounds > ec50_dichoto
wmModelSelection -i ec50_dichoto -d CCLE_expr -k 10 > selection.jobs
para create selection.jobs
para push
wmBestModels -D 0.05 raw/*.tab > best.tab
wmTrainModel -i ec50_dichoto -d CCLE_expr -B TCGASamples -R best.tab -o models
mv models /data/samplepsychichome/signatures/CCLE_drug
```

Listing 6.1: Building Signature Sets for Sample Psychic

wmDichotomize splits numeric variables on median or upper/lower quartile, and for categorical variables it expands a single multi-class categorical variable into a series of corresponding binary variables. wmModelSelection creates a parasol-

ready list of jobs (-k specifies how much work each job should do). The two para commands launch the thousands of jobs in the selection.jobs file. wmBestModels picks out the best model for each of the 545 drugs, subject to the constraint specified here that it must beat a majority classifier by 5%. wmTrainModels trains classifiers using the best algorithm/hyperparameters and outputs them to a models directory. It also applies each model to all the samples in TCGASamples and saves the distribution along with the model. Finally copying the model directory to the Sample Psychic signatures directory lets Sample Psychic pick up the results. That's it, from data and meta-data file through thousands of cluster jobs to evaluate models to final deployment.

Sample Psychic can be used in two modes. In it's interactive mode users upload gene expression data to Sample Psychic, select model sets to apply, apply those model sets, and view the results. In the off-line mode, a wmApplySignatureSets script can be used to perform the work of Sample Psychic's engine off-line and the hash-identified results can be copied into Sample Psychic results directory for viewing in the web application. For modestly sized datasets of tens or hundreds of samples all the steps involved take a few to tens of seconds to complete, and so Sample Psychic can be used as an interactive classification application in real-time. For larger datasets, involving thousands of samples and applying signature sets containing thousands of classifiers, it may be more convenient to perform the classifications off-line and simply use Sample Psychic as a viewer.

Sample Psychic saves results in a result file, and each session is assigned a unique hash. The same hashed file name is created by wmApplySignatureSets. This hash key can be used in the Sample Psychic URL to call up previously analyzed datasets. Importantly for some projects, a list of URLs can then be created to link data on one web page to a Sample Psychic session for viewing.

Sample Psychic is built with the Vaadin [103] web component toolkit, which is itself built on top of Google Web Toolkit [104]. Google Web Toolkit allows the development of complex web applications almost entirely in Java, without the need to write difficult-to-maintain Java Script or worry about browser-specific quirks. The toolkit provides lots of high-performance components for web applications and is used to build several of Google’s online properties. Vaadin is a web application framework that takes the low-level components of GWT and creates more complex components ready-for use. The most prominent Vaadin component used in Sample Psychic is a Grid, which is like a spreadsheet table for the browser. A Vaadin Grid automatically handles dynamic loading of data as the user scrolls, allowing for fast webpage start-up time a webpage even if it is intended to display tens of thousands of rows of data. While there are some initial costs to using a framework like Vaadin, the long term benefits in maintainability of code and performance of the resulting application are expected to result in a net reduction in long term effort.

6.2 User Interface

Sample Psychic accepts user data in tab delimited files Figure 6.1. So long as very basic processing of the RNASeq data has been applied to convert the RNASeq read counts into RPKM, FPKM, or similar per-gene abundance measure, Sample Psychic can work with the data. Because batch effects and pre-processing steps are often so large for gene expression profiles, Sample Psychic uses a ranked based normalization scheme that first ranks genes then fits the ranked list to a unit exponential, as described in Appendix A. As that appendix details, this heavy handed normalization costs a little in absolute performance but provides the robustness needed to work with data that comes from many different sources.

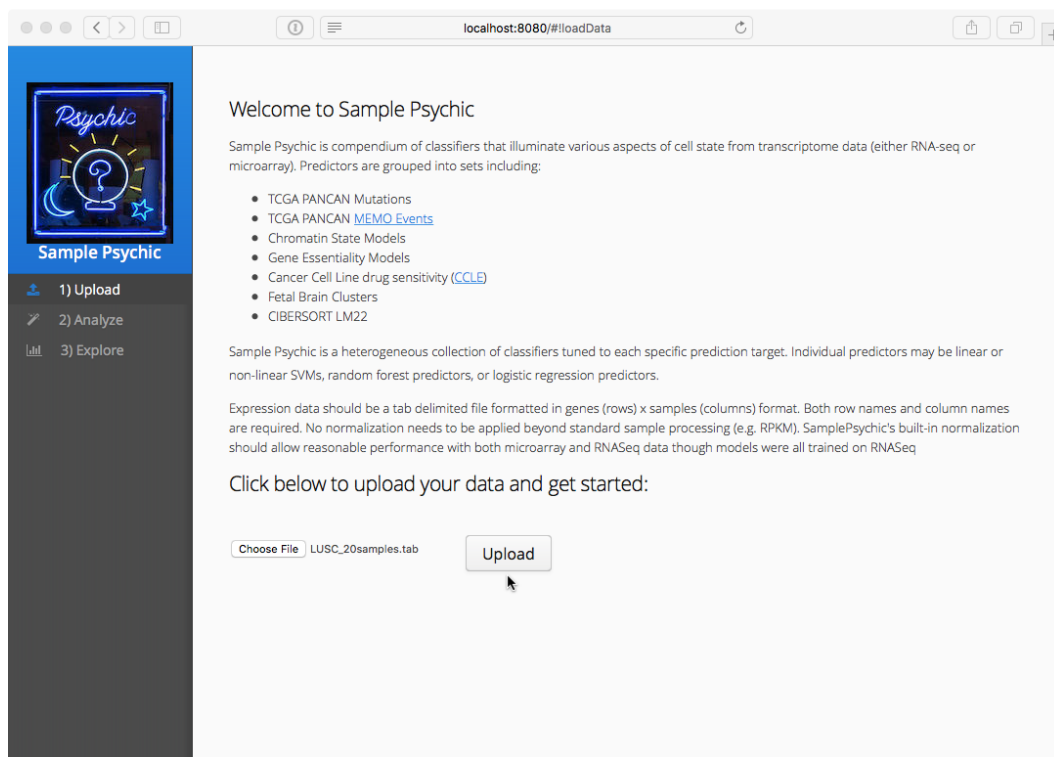


Figure 6.1: Upload data screen for Sample Psychic.

This normalization even allows microarray data to be used as well as RNASeq data, though with an added small hit to performance.

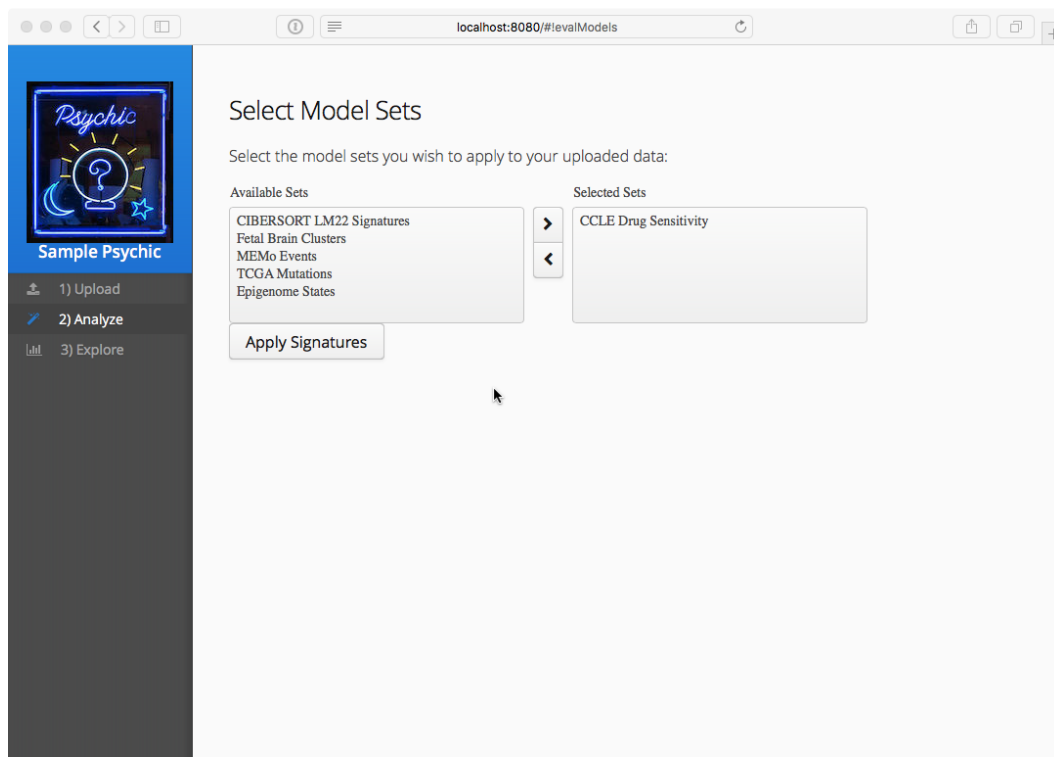


Figure 6.2: One or more sets of models can be selected. Each set of models is a heterogeneous mixture of classifier types tuned to each target class.

Once data has been uploaded users are presented with a signature selection page that allows them to choose which signature sets they would like to apply to their data, as in Figure 6.2. One or more signaiture sets can be selected. Once signature sets are selected, "Apply Models" will apply all of the signatures in those sets to the uploaded data. Once the classifications are complete, the user it taken to a page that allows access to all the results, as in figure Figure 6.3.

Applying hundreds of classifiers to hundreds of samples results in tens of thousands of classifier calls. Simply listing such results in a single web page results in slow loading of the webpage and often slow and difficult browsing once loaded. The use of Vaadin Grid for the results solves these two problems, since only vis-

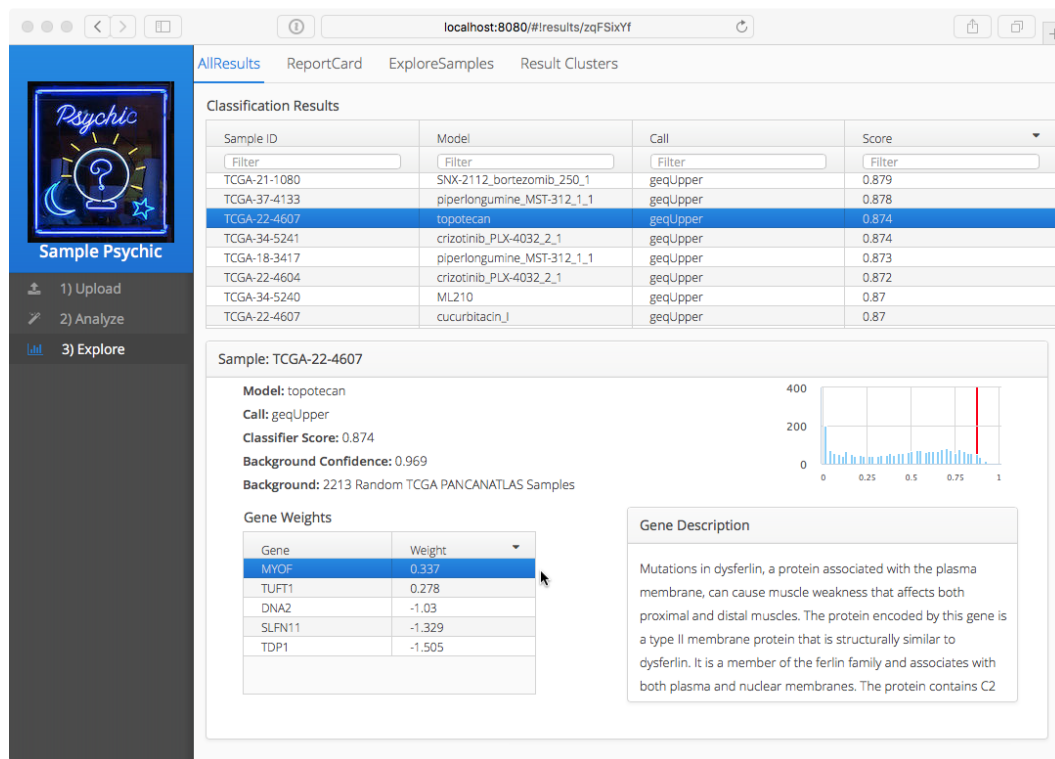


Figure 6.3: A scrollable list of classifier results. Information about selected result is shown. Information includes where this sample's classifier result falls on a distribution of background samples, the p-value of that background level, the weights of genes in the model, and brief information about each selected gene. Each column can be sorted or filtered.

ible data is transferred over the network, and Vaadin Grid intelligently provides dynamic loading of data as the user scrolls to it.

When a result is clicked on a summary panel of information about that result is displayed. This summary panel contains information about the model and the classifier output for this sample. Information about each model includes the distribution of model scores across a background sample. The summary panel shows both the p-value of the classification on this background and also a plot of the background distribution highlighting where the current sample falls on the distribution. The genes used by the classifier are shown with a proxy for their gene weights. In the case of logistic regression and support vector machines these weights are simply the per-gene coefficient of the hyperplane. In the case of random forest I have written special code in wekaMine to perform a DFS search over all trees, summing up the number of leaves dependent on each feature (gene) in each tree. This sum, divided by the total number of leaves, is the score for that feature (gene) in the random forest. Clicking on a gene in the gene weights list will bring up a short description of the gene derived from the USCS Genome Browser gene description tables, which are derived from several sources annotated at the bottom of each description.

Figure 6.5

Even though the Vaadin Grid component enables the display of tens of thousands of results, that's still more than humans can wade through by hand. To help manage this complexity each column of the data can be sorted. Each column can also apply a filter to the rows of data displayed. Thus if one wants to see only results for the drug "topotecan" it's possible to enter that word, or part of it, in the filter column for models, and then only rows that have the matching text in them for that column will be shown Figure 6.4. Numeric columns can be filtered

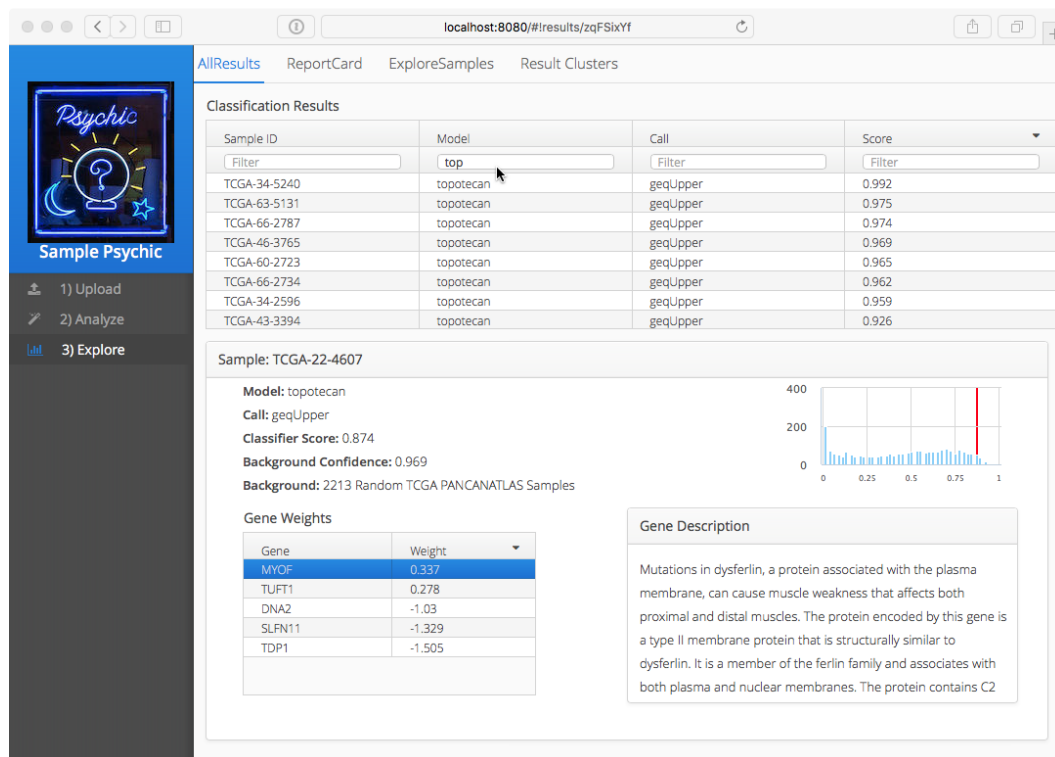


Figure 6.4: Since the list of results can include many thousands of entries, it's possible to filter each column to narrow the results shown. Shown here is filtering the model column to show only rows where the model name includes "top".

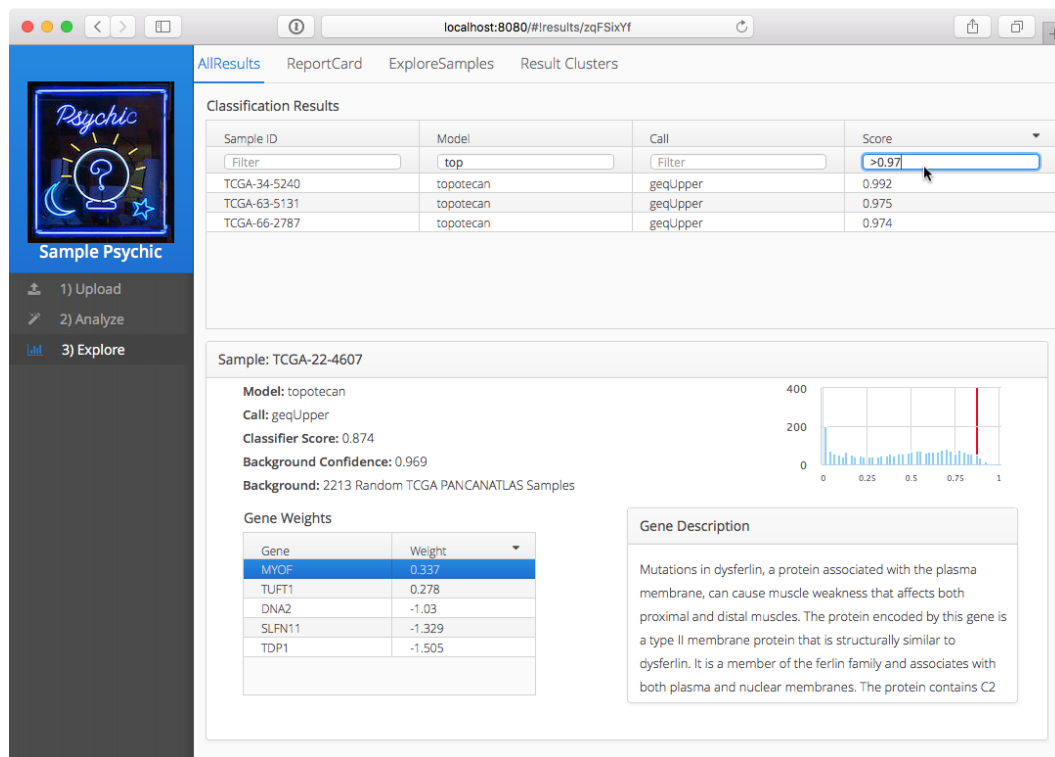


Figure 6.5: Numeric columns can be filtered by entering comparisons, such as ">0.9" or "=0.95". Multiple column filters return the intersection, so here is shown all models containing "top" with a score > 0.97

by value or by a value range. To see only results with a score over 0.95, one can enter ">0.95" into the filter field and only rows with that score constraint will be shown. The filters applied to separate columns are intersected, so entering "topo" into the model column and ">0.95" in the score column will show only rows where both the model name contains "topo" and where the score is ">0.95".

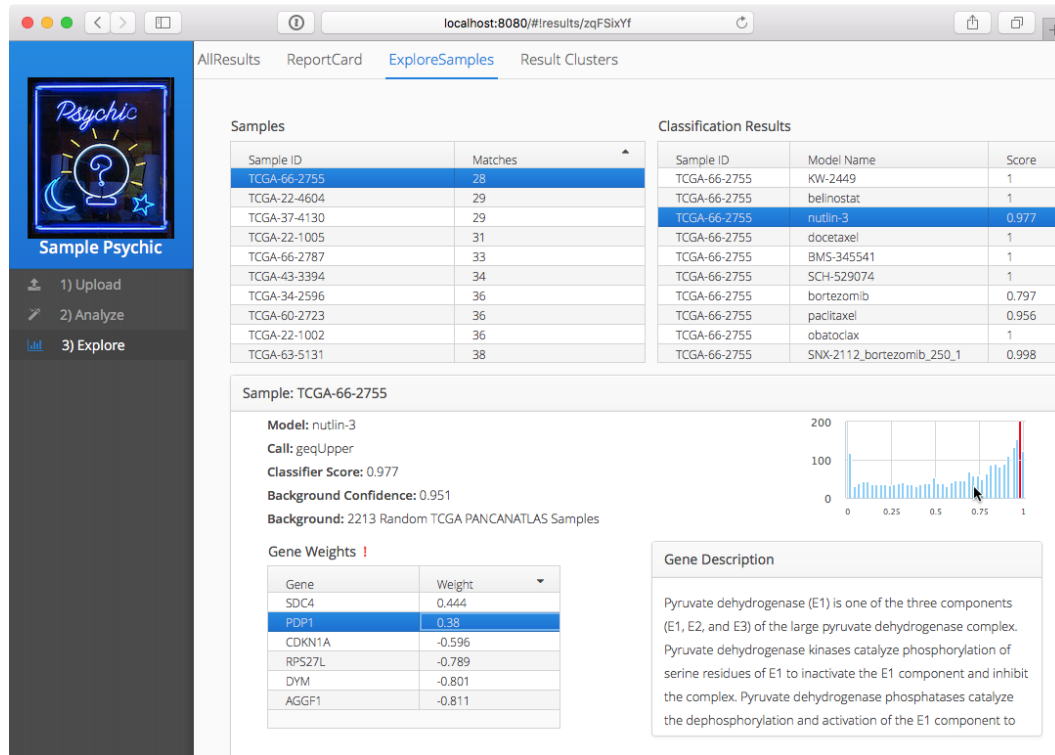


Figure 6.6: Sample summary is an alternate way to explore by-sample results. Left grid selects a sample, right grid shows results for that sample. Selecting result on right displays summary information about the result.

The "Sample Summary" page, Figure 6.6, provides another way to browse all of the results that is oriented around samples. On the "Sample Summary" page the left hand grid contains a list of samples and how many models returned scores over a basic threshold of goodness (e.g. 0.70) for each sample. Clicking on a sample name populates the results column with just the results for that sample. Clicking on a sample

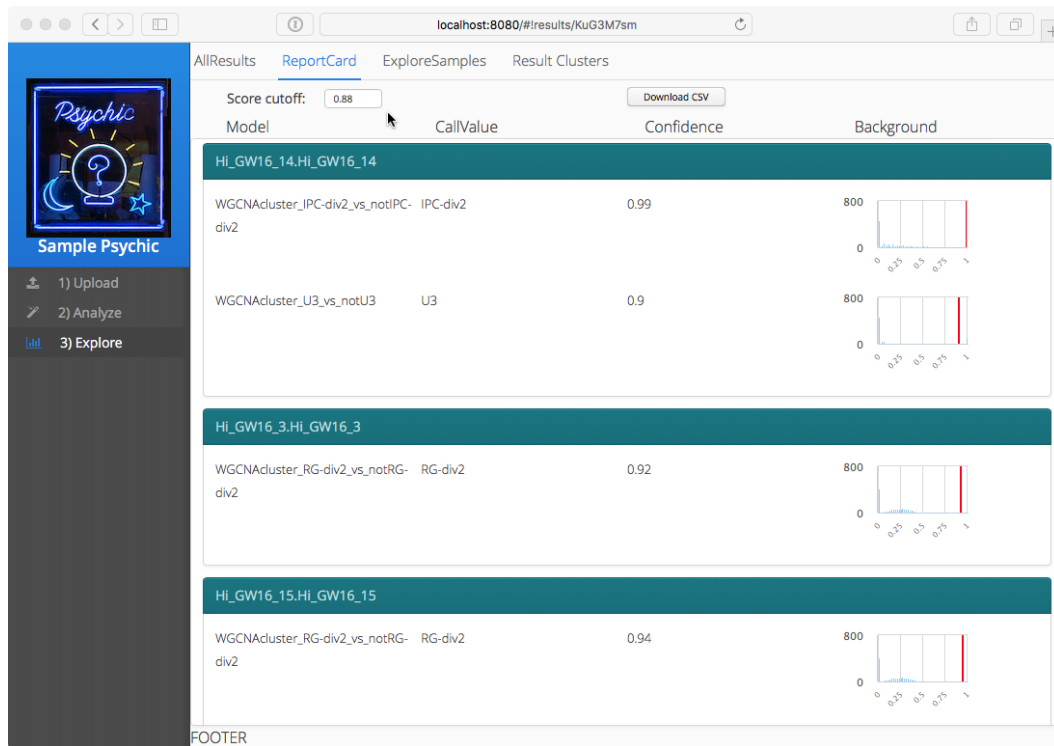


Figure 6.7: Sample report card view groups each sample with top hits.

The report card page, as shown in Figure 6.7, is provided to give a short by-sample view of only the very top classifier results. Since it doesn't use the Vaadin Grid framework, but is built in more traditional way, all of the results are loaded at once. As a result, it can become sluggish if the report threshold is set too low and so includes too many results. Still, this clean view focused on each sample is preferred for some applications.

A signature set applied to a set of samples produces a vector of classification calls for each sample (a vector of semantic attributes). To explore how these samples relate to each other in terms of these semantic attributes these classification vectors can be clustered and the results visualized. The "Results Clustering" page does this, applying the t-SNe algorithm [VanDerMaaten: __XO1JDSF] to produce a 2D visualization of this high-dimensional similarity. Figure 6.8. Nearby

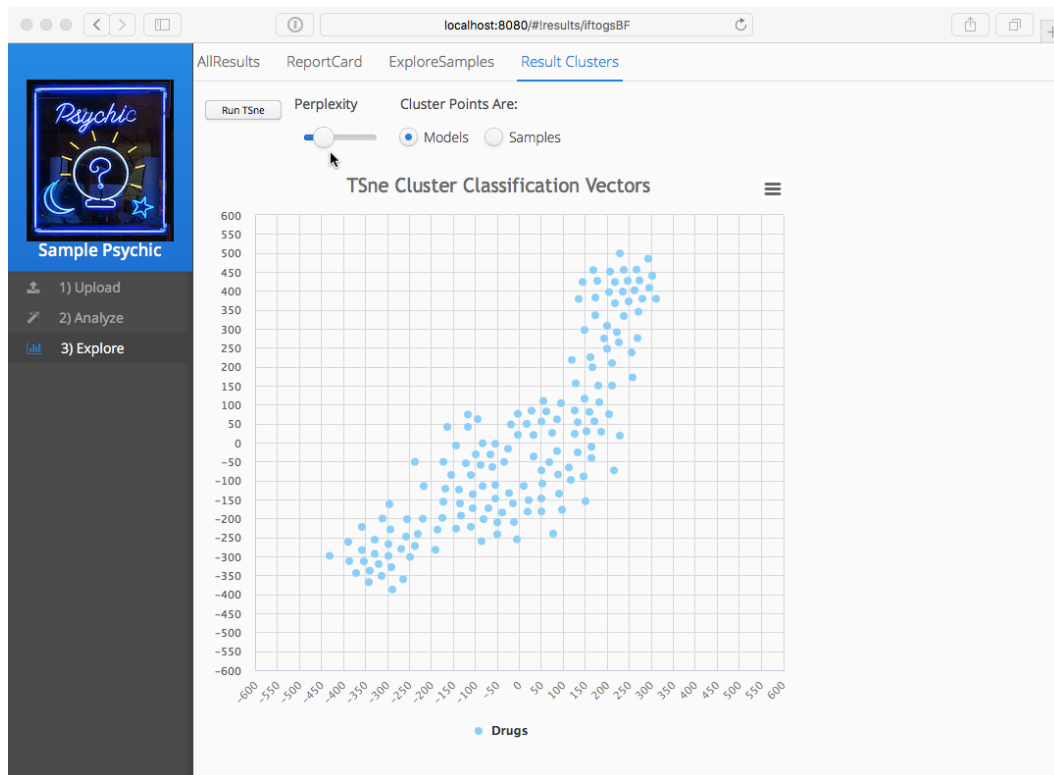


Figure 6.8: The vector classifications for each sample can be used to cluster either the samples or the models to indicate which samples are most similar with respect to the applied classifiers or to indicate which models are most similar across the applied samples. This high-dimensional similarity is visualized in two dimensions here with the t-SNE algorithm [VanDerMaaten:_XO1JDSF].

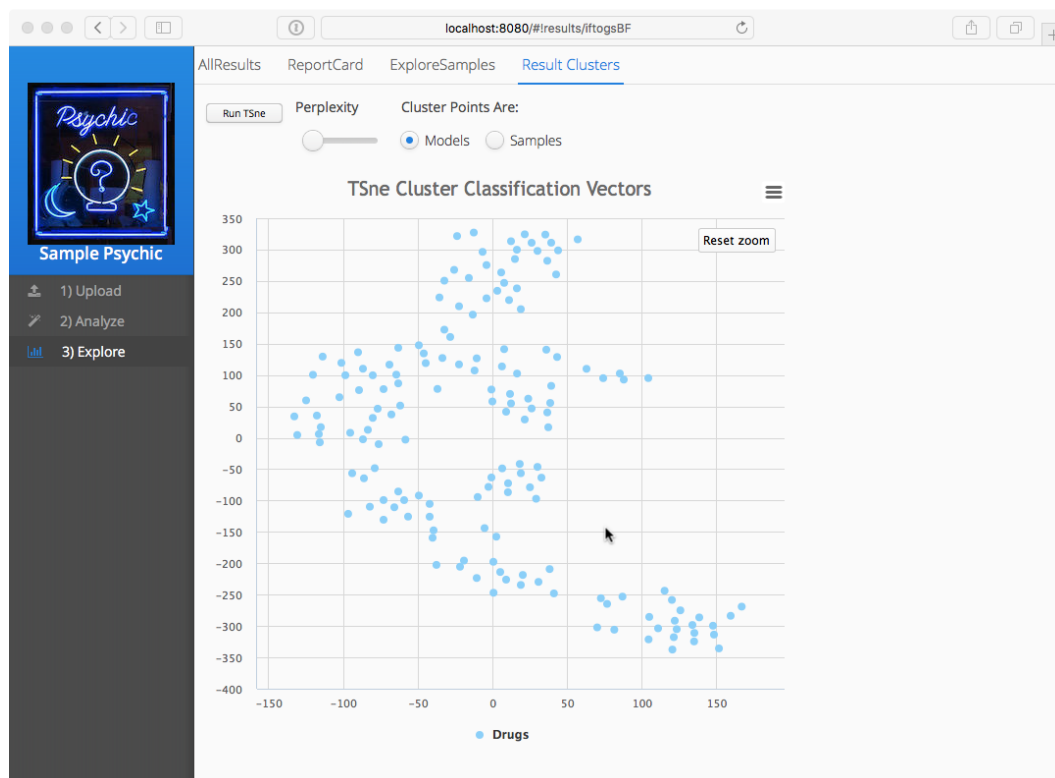


Figure 6.9: The t-SNE algorithm parameter "perplexity" can be adjusted to tweak the visualization.

points (samples) in the visualization are more similar in their classification vectors. For example, if the signature sets applied are drugs, nearby samples would have more similar profiles of drug response predictors than samples further away. Hovering the mouse over a sample identifies the sample and the list of models (e.g. drugs) that the sample had high classification scores for. Since this visualization is sensitive to a "perplexity" parameter, a slider is provided to adjust that parameter. Figure 6.9. It's also possible to turn this comparison around and ask "Which models are similar to each other given this dataset?". For that, a radio-button is provided to produce a clustering where each point in the visualization is a model instead of a sample.

6.3 CIRM Single Cell Report Card

Because of its modular design and tight integration with WekaMine, it is easy to adapt Sample Psychic to new uses. One such use is to provide a report card view for the California Institute of Regenerative Medicine (CIRM) single cell initiatives, Brain of Cells and Heart of Cells.

Single-cell data presents special challenges for building classifiers. In a large collection of cells the transcript profile across cells will reflect state that is the union of many individual states. This union of gene transcripts may reflect a state that no individual cell may be in. A consequence of this is that at any given time a single cell will be lacking measurable transcripts for many genes.

To investigate the consequences of this for building classifiers for single cell data, and especially for applying classifiers built on non-single cell data to single cell data, I performed experiments simulating single cell gene dropouts on Allen Brain Atlas (ABA) data [105].

Using pilot embryonic brain single cell data we can see that the typical sample

has only about 30% of the available genes represented Figure 6.10.

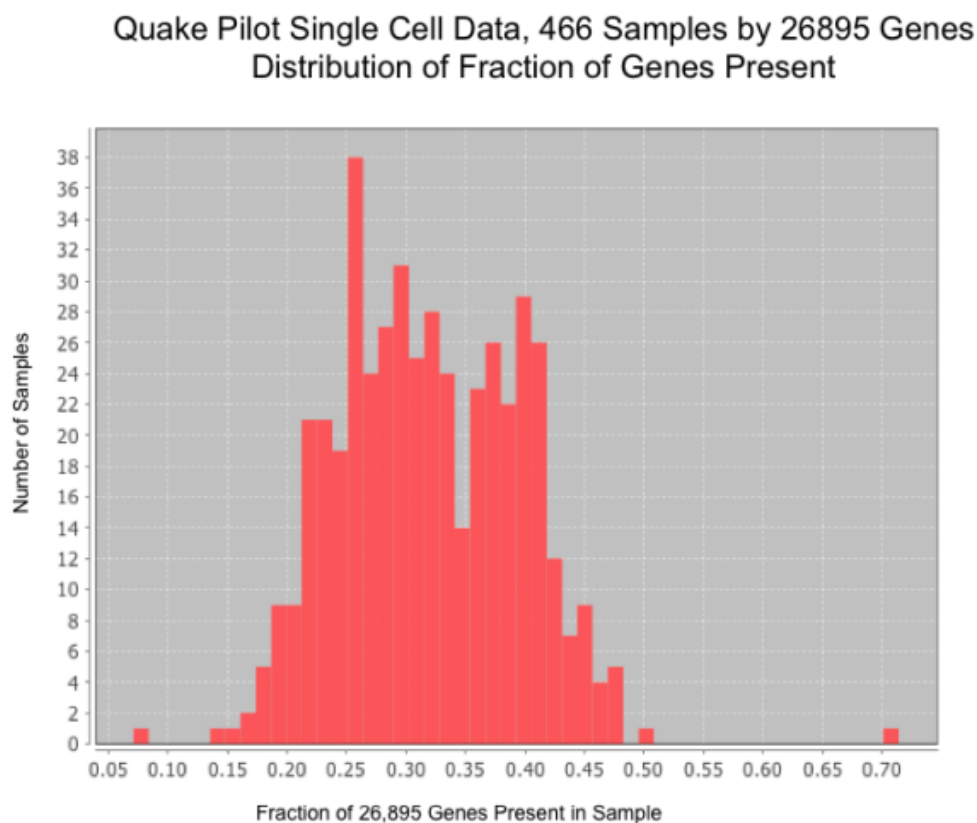


Figure 6.10: Distribution of fraction of genes present in single cell pilot data.

The Allen Brain Atlas RNASeq dataset I use here contains 3702 samples covering 29,131 genes and other transcripts. The ABA includes a detailed tissue ontology for all samples, see figure Figure 6.11 and Figure 6.12. One approach to providing a report card for CIRM samples is to build classifiers for ontology levels at different depths in the ontology tree. To investigate the effects of dropouts on this I built models for the fifty nine tissue types at the lowest depth in the tree. Using the empirical distribution of gene frequency in pilot data to produce simulated single-cell data I explored how performance is affected when classifiers are built on full data and on simulated single-cell data.



Figure 6.11: First five levels in Allen Brain Atlas Ontology.

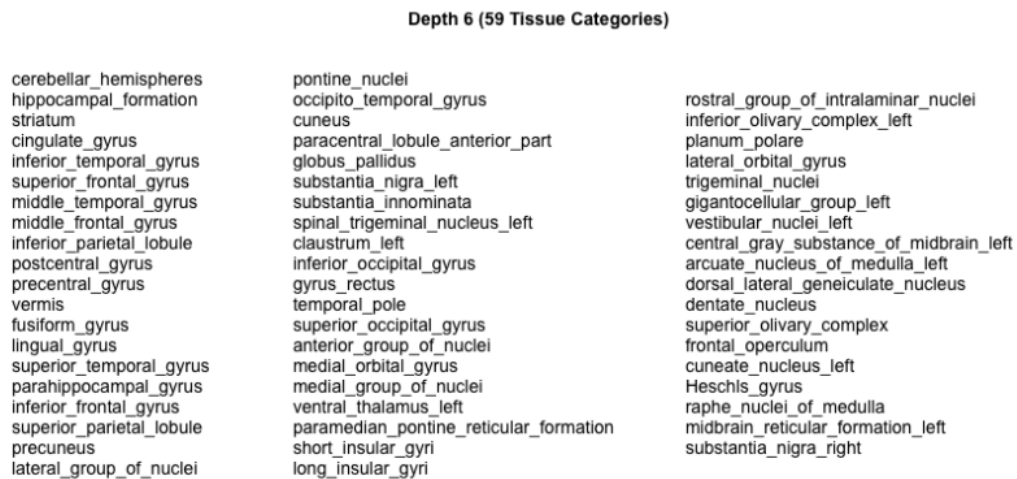


Figure 6.12: Sixth level in Allen Brain Atlas Ontology with fifty nine tissue types.

As a reference, a WekaMine model selection experiment was performed on fifty nine Allen Brain Atlas tissues using the full gene set for both the training and test data, with results shown in Figure 6.13. Next I performed a WekaMine model selection experiment on these fifty nine Allen Brain Atlas tissues using full gene sets in the training data for each cross validation fold, but using simulated single-cell drop-outs in the test set. As can be seen in Figure 6.14, the median and best performance is comparable but the variance in performance increases notably. Finally I performed a WekaMine model selection experiment using simulated single cell gene sets in both the training data and test data of each cross validation fold. As can be seen in Figure 6.15 simulating single cell drop-outs in the training data reduces the variance in performance considerably compared to using full gene feature datasets as input.

Other strategies are possible than simulating single-cell dropouts stochastically. For example, one could try to build models using only the subset of most commonly expressed genes that are likely to appear in a high fraction of single cell data. While more experiments are needed to determine the best way to construct models to apply to single cell data, the simulated single cell experiment here shows that it's possible to train on full gene sets and apply to single cell data. These experiments also show that some adjustment to the training data to match gene drop-out expectations from single cell data is prudent.

BASELINE

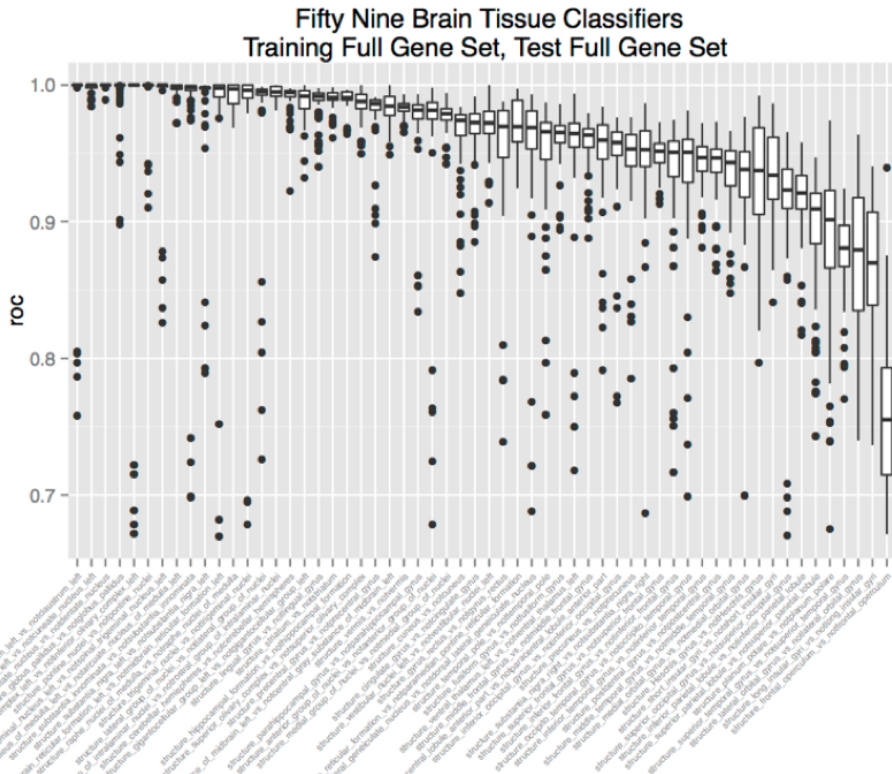


Figure 6.13: WekaMine model selection experiment performed on fifty nine Allen Brain Atlas tissues. Both training and test use full gene sets. Variation across algorithms and parameters.

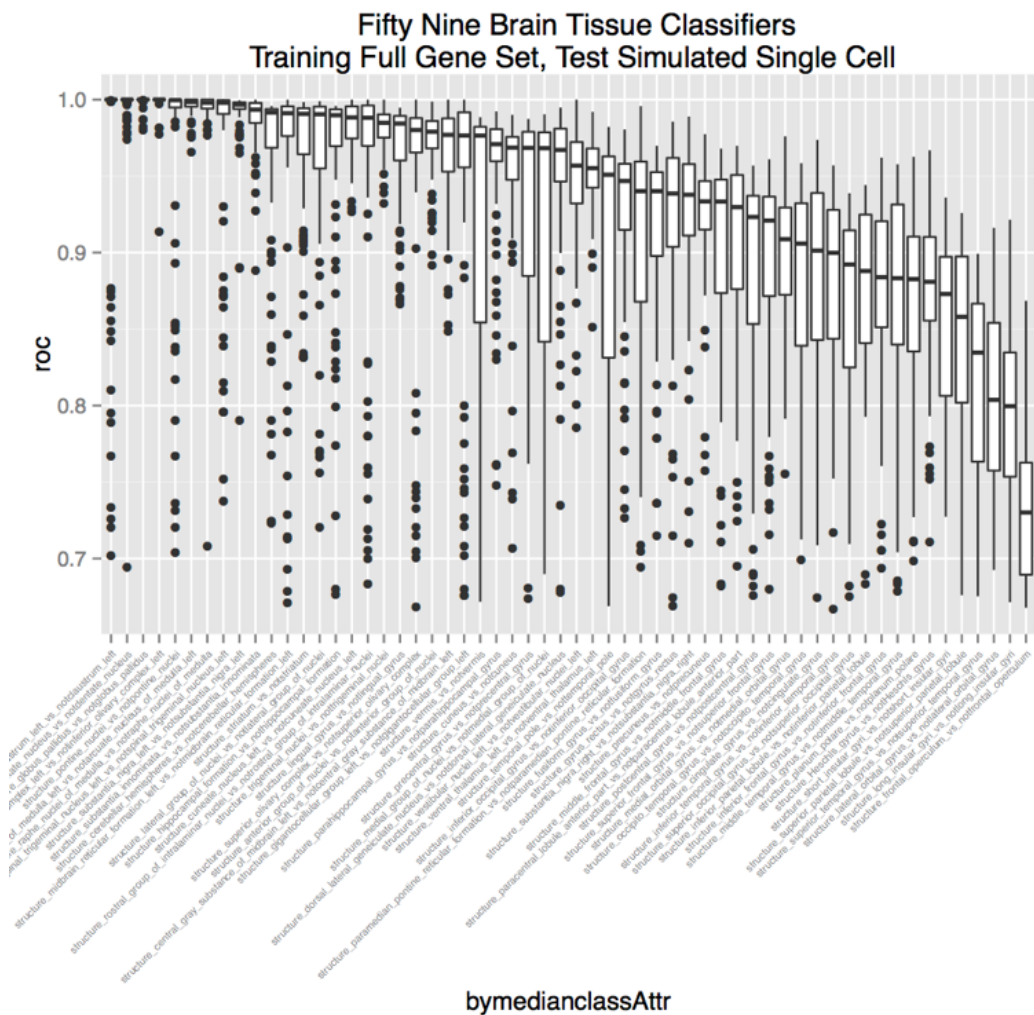


Figure 6.14: WekaMine model selection experiment performed on fifty nine Allen Brain Atlas tissues. Variation across algorithms and parameters. Each point represents a 10x cross-validation experiment. Each cross-validation fold was constructed so that training data had full gene set but test set was simulated to match single cell gene drop-out distribution.

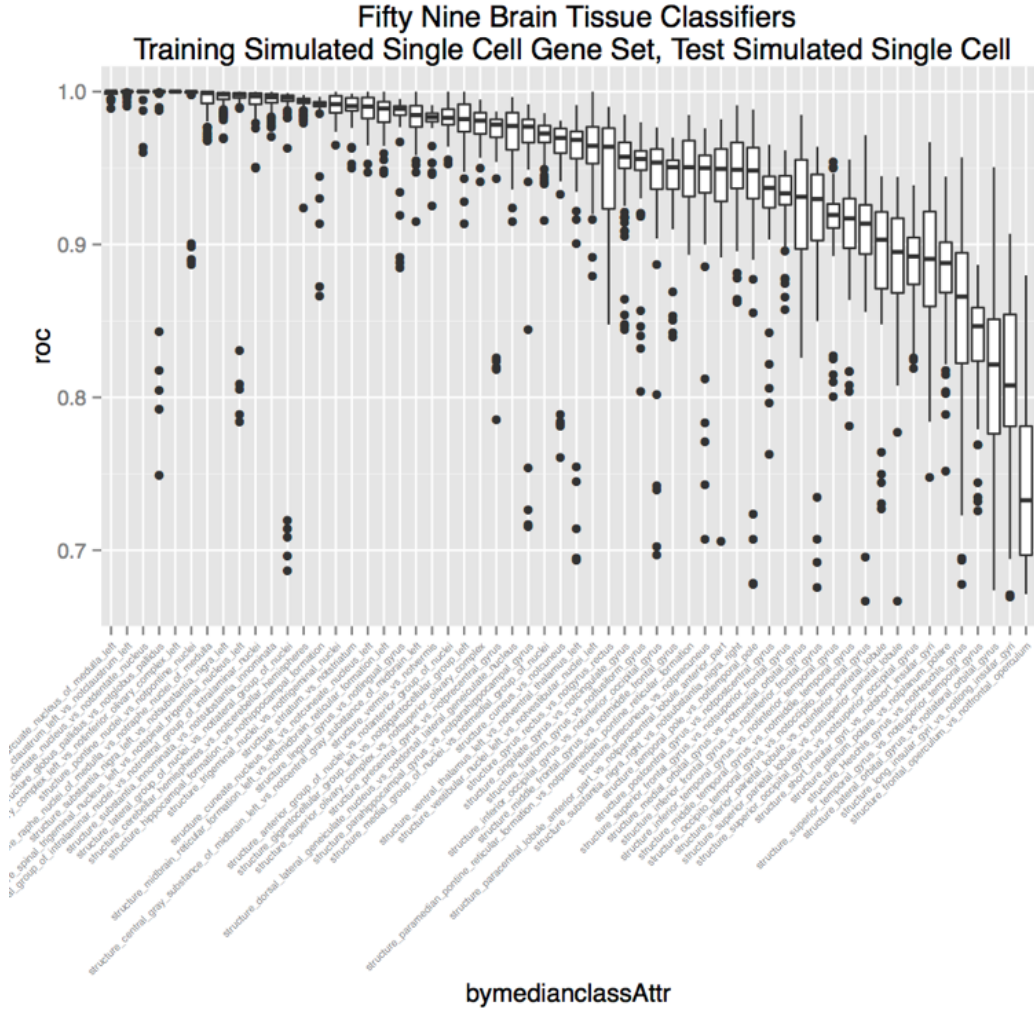


Figure 6.15: WekaMine model selection experiment performed on fifty nine Allen Brain Atlas tissues. Variation shown is across algorithms and parameters with each point representing a 10x cross-validation experiment. Each cross-validation fold was constructed so that both training data and test set were simulated to match single cell gene drop-out distribution.

Chapter 7

Conclusion

To conclude this effort it is useful to look back on what I had hoped the contribution of semantic attributes to gene expression classification would be. These were:

1. More accurate prediction of prognostic outcomes
2. Better interpretability of prognostic classifiers.
3. Possibility of discovering unique contributing factors to classification
4. Able to integrate wide range of external sources of knowledge.
5. Able to use high confidence knowledge.
6. Able to add knowledge incrementally.
7. Able to leverage public data beyond experimental chort.

Many of these benefits were, indeed, realized. It was possible to integrate a wide range of external sources of knowledge, from mutation data to classifications to classifications output from a completely independent classification system (here

URSA). This knowledge was, in many cases, high confidence information. It was possible to add attributes incrementally. As we saw when examining chromatin features for drug sensitivity classification in section 5.1.7, high information gain features were correlated with high performance classifiers, so that using attribute selection is able to choose among a pallet of semantic attributes without introducing excessive noise. I was also able to leverage public data from sources removed from the original experimental chort. So on those measures I can tick off the boxes.

Any fair assessment of the technique, though, must linger on the first two contributions which are the contributions that most motivated this approach. The first is increased accuracy. Over a range of problems I was able to occasionally find combinations of problems and semantic attributes that genuinely improved accuracy. In every case, though, the improvements were exceedingly modest, a few percentage points on already poor classifiers, or the sort of improvements on good classifiers that require careful experiments to even uncover. Against these very modest gains must be weighed the very large effort required to create, and eventually maintain, a collection of semantic attributes. In the machine vision literature one goal of semantic attributes not listed above was a reduction in the considerable work of feature engineering that historically has consumed so much of the effort in machine learning. However, as discussed in 3, efforts to create an automated pipeline to generate a large pallet of semantic attributes from public collections like GEO proved prohibitive, largely because of the significant human knowledge that was required to interpret the sample descriptions. Turning to pre-curved datasets was more tractable, but I found that even with a streamlined machine learning pipeline that allowed me to build hundreds or thousands of models at a time, the effort to build semantic attributes from curated datasets was

enormous. Where I started with one classification problem to optimize, say drug essentiality prediction, I ended up with hundreds or thousands of classification problems to optimize across a wide variety of domains, each with their own special problems (what mutations to count, how to collapse mutation categories, how to even begin to model chromatin state, etc). In principle this could be a one-time effort, but the problem-specificity of useful attributes, coupled with the need to maintain this diverse set of classifiers over time, along with the code base they are based on, cuts into the long-term utility. Perhaps for a ten percent gain in classification accuracy it would be worth investing such ongoing effort, but for sub-percentage point gains it is simply not viable.

Interpretability of prognostic classifiers was, in many ways, the most promising aspect of semantic attributes. In fact, given that clinical prediction is an environment of ongoing discovery of underlying mechanisms (unlike, say, identifying zebras in photos, where the zebra concept is already understood), interpretability is a useful enough goal that it could be worthwhile even with no, or even slightly reduced, prediction accuracy. I examined the relationship between gene essentiality predictions and mechanism of action of drugs in 5.1.6, and find there that the contribution of essentiality predictions is diffuse across many predictors. No obvious patterns presented themselves, leaving us in a position not much different from having a pallet of gene expression levels to examine. For mutation features I found in 5.1.8 that the mutation classifiers did not reliably identify mutations with enough specificity to be interpretable. And while chromatin features yielded a number of tantalizing and suggestive associations in 5.1.7, ultimately they suffer from the same lack of specificity found in mutation attributes. It seems that when semantic attributes work at all at improving performance, they do so in a diffuse way that is spread across many semantic attribute classifiers, reducing the ability

to learn anything from the semantic label of the classifiers.

A classifier built on top of diffuse network of hard-to-interpret features will remind anyone familiar with recent trends in machine learning of neural networks. In recent years, given increased computing power and exploding datasets, some of the biggest impacts in machine learning have come from work on deep neural networks (commonly referred to as "deep learning") [106]. Models have been built that contain billions of parameters, and have yielded substantial improvements in performance over other approaches. The internal layers of these networks can be viewed as a feature space transformation, much as I aimed for with semantic attributes, except where I have laboriously built my semantic attribute latent feature space by curating and training thousands of classifiers, deep neural networks can use unsupervised training to provide a way to build such an internal space automatically from training data [107]. As with image and speech processing applications where an explosion of training data made such approaches feasible, gene expression data is becoming more and more plentiful.

Taking these things together, my recommendation for future work would be to shift from the semantic attribute approach to an approach rooted in semi-supervised deep learning. Such an approach would first train a deep network on a large corpus of data in an unsupervised way, using either a stacked denoising autoencoder architecture [Vincent:vu], where noise is added to input samples and the deep network is trained to reconstruct the original samples from the output, or possibly using a variational auto encoder [108]. After training such a network in an unsupervised way on a corpus of gene expression data, the supervised clinical training task would use the outputs of hidden layers when presented with gene expression profiles as transformed and dimensionality reduced features for the supervised learning step, which could use either fast traditional algorithms such

as support vector machines and random forest.

To this end I have begun the process of integrating deep learning tools, specifically Deep Learning for Java (dl4j) [109], into my toolkit. For stand-alone neural network training I am exploring implementing a search over network topologies with syntax similar to:

```
def layerExperiments =  
[[[1999,800],[800,200],[200,50],[50,2]]  
[[49,20],[20,12],[12,12],[12,12],[12,6],[6,2]],  
[[49,49],[49,24],[24,12],[12,12],[12,2]],  
[[49,49],[49,24],[24,12],[12,6],[6,2]]  
[[49,24],[24,16],[16,16],[16,12],[12,2]],  
[[49,20],[20,12],[12,12],[12,12],[12,2]],  
[[49,20],[20,12],[12,6],[6,2]],  
[[49,24],[24,12],[12,2]],  
[[49,12],[12,6],[6,2]],  
[[49,24],[24,2]],  
[[49,12],[12,2]],  
[[49,20],[20,2]],  
[[49,6],[6,2]]
```

I have also begun experiments with auto-encoders. In one test I created a mixture of breast carcinoma data (BRCA) and Lung Squamous Cell Carcinoma data (LUSC). I trained the auto-encoder on this mixed dataset with a two node hidden layer. The output of the hidden layer when presented with new samples showed a 0.8 Spearman correlation with the BRCA/LUSC label. There are many issues to explore in pre-processing the data, the best network architecture, the number of nodes in the hidden layer, and so on, along with the considerable computational resources needed, but in light of the issues and modest results of semantic attributes I plan to invest effort going forward into this semi-supervised technique.

Appendix A

Normalization

It is common when running multiple experiments with different oligonucleotide arrays to find significant non-biological sources of variation in the arrays. Normalization is an attempt to minimize these sources of variation. The most commonly used normalization methods rely on having access to all the data from the multiple arrays/sources/tissues at one time (e.g. loess and contrast normalization). One of the most successful of these is quantile normalization Bolstad et al. [110]. The essential idea of quantile normalization is to force the feature values for each sample to have the same distribution. What this equivalent distribution is will depend on the data being normalized and requires having all of the values in hand at one time. However, the goals of this work require the prior training and curation of models that can be applied to never before seen data. Using a whole data normalization such as quantile normalization would require data to be re-normalized and models retrained with the newly normalized data each time the models are applied to a new batch of samples. Not only does this require a considerable amount of extra computation, it requires that all of the training data be kept at hand when using the models. Given that this work calls for the creation of hundreds or possibly thousands of models, both the re-computation time and the

data curation resources would prove prohibitive if models had to be recomputed with each new data application. This has led us to consider online alternatives. The success of the quantile normalization method led us to consider a closely related idea. Rather than try to fit all the samples to some distribution that is dynamically determined from all of the samples, what if we fit all of the values to a single fixed distribution?

A.0.1 Exponential Normalization

I have implemented and tested this idea in a number of scenarios related to the aims of this work. A unit exponential distribution was chosen as the distribution to fit. Partly this is because RNASeq data is already exponentially distributed, and partly based on an intuition that this would tend to reduce the magnitude of background genes while exaggerating the magnitude of outlier genes. In particular, for exponential normalization we first rank the samples p_i where for each i $p_{i-1} < p_i$ and then apply the transform

$$F^{-1}(p_i; \lambda) = \frac{-\ln(1 - p_i)}{\lambda}, 0 \leq p_i \leq 1 \quad (\text{A.1})$$

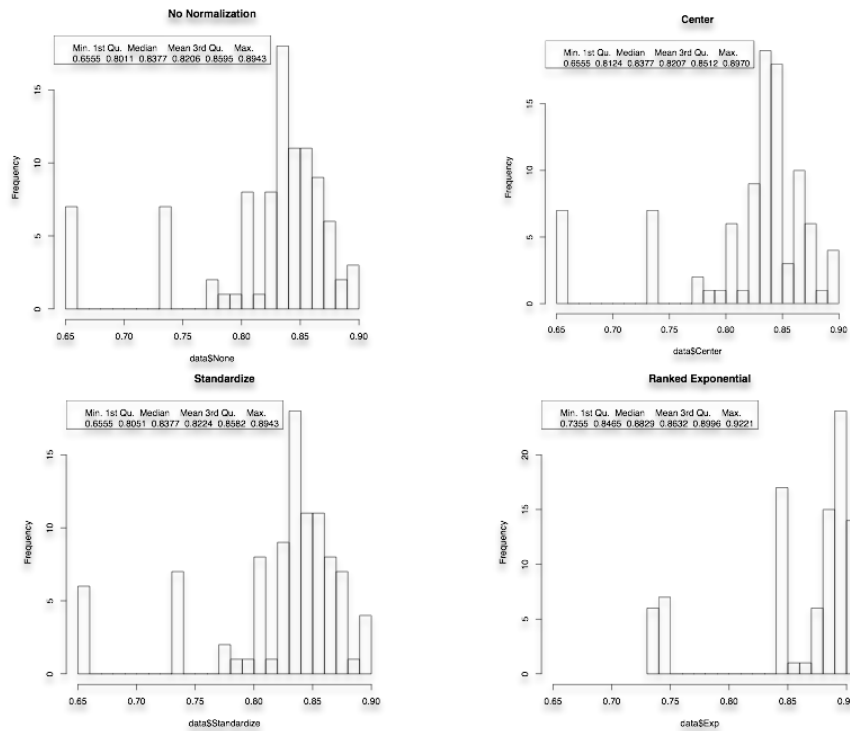
Whatever the original distribution of the input data, this transformation will result in data with an exponential distribution with mean λ , where $\lambda = 1$ is typical. Importantly, new samples can be fit to this distribution without knowledge of prior training data. For this preliminary work I have tested this normalization scheme along side several normalization schemes available in Weka and have validated that it outperforms these for my intended applications. These tests are documented in the sections that follow and serve to validate not only exponential normalization but also the semantic attribute uses I am building. For the completion of Aim 2 I will further test this normalization against baseline array methods as well as

other fixed transformations to complete the validation of it's efficacy.

A.0.2 Cross-tissue Normalization Test

Using the TCGA PANCAN 8 dataset I took the five tissue datasets that had a minimum number of TP53 mutations (at least 20 non-synonymous mutations per tissue) and built classifiers for these non-synonymous TP53 mutations. For the plots shown in Figure A.1 I trained the classifier on four of the datasets holding out the fifth dataset and then evaluated the trained classifier on the held out set. So the classifiers were tested on samples from a tissue they have never seen in training. I repeated this 5x for a 5x cross-tissue validation. The average roc across all five folds is reported here. It is already notable that TP53 classifiers trained on four tissues perform well on a tissue type never seen before. This lends support to the assumption that strong semantic attribute classifiers can be built with heterogeneous data and applied meaningfully to data from different tissues, etc. This experiment was with different classifier algorithms (Random Forest, Balanced Random Forest, SVM with linear, quadratic, and radial basis kernels), different supervised attribute selection methods (Information Gain, ReliefF, Linear Forward Selection) and four different normalization methods (No normalization, Center, Standardize, and Exponential Normalization). The plots here show the distribution of average roc values across these methods for each of the four normalizations attempted. In this smallish experiment there were 100 combinations of classifier/supervised attribute selection and parameters considered per normalization. The distribution of roc scores gives a picture of how the normalization affects the performance. It's clear that on average and in terms of the best combination, the exponential normalization does best (0.9 median vs 0.84 median) in this cross-tissue classification task.

Normalization:TP53 Cross-tissue CV, 100 algorithms/params



18

Figure A.1: Exponential Normalization Applied to Cross-Tissue Classification

A.0.3 Stem Cell Cross-Platform + Cross-Tissue Cross Classification Test

This test looks at the scenario where we are applying a model trained on one tissue (e.g. stem cells) and on one platform (microarray) then applied to other tissues (nine tumor types) on both microarray and rnaseq data. This is the most extreme scenario for the semantic attribute approach to classification. In this scenario we are applying a stem cell model trained on microarray data to rnaseq data from tumor tissues and want to assess the “stemness” of the tumor tissues. The ideal normalization here would have perfect correlation between the output of these classifiers applied to microarray data and applied to RNAseq data. In this experiment I performed a wekaMine model selection of stem cell classifiers on microarray data from GEO (data and labels curated by Dan Carlin) and trained a classifier for the best model that used each of these normalizations: None, Normalize, Exponential Normalize. In all three cases the average auc of the best model was over 0.95, so this is an comparatively easy prediction task. Each of these best classifiers (3 classifiers * 3 normalizations) was applied to a set of 1434 TCGA Pancan (Freeze 4.5) samples, from nine tissues (BRCA,COAD,GBM,KIRC,LUAD,LUSC,OV,READ, and UCEC) that have both microarray and RNASeq data. The following scatter plots, Figure A.2, show that for each of the three models (two shown, one summarized), the classifier calls between microarray and rnaseq datasets were most highly correlated when using exponential normalization. There is no known truth in these cases, as we are essentially measuring the “stemness” of random cancer samples, but it shows that at least a measure of the information in the model survives the platform change using exponential normalization but not otherwise.

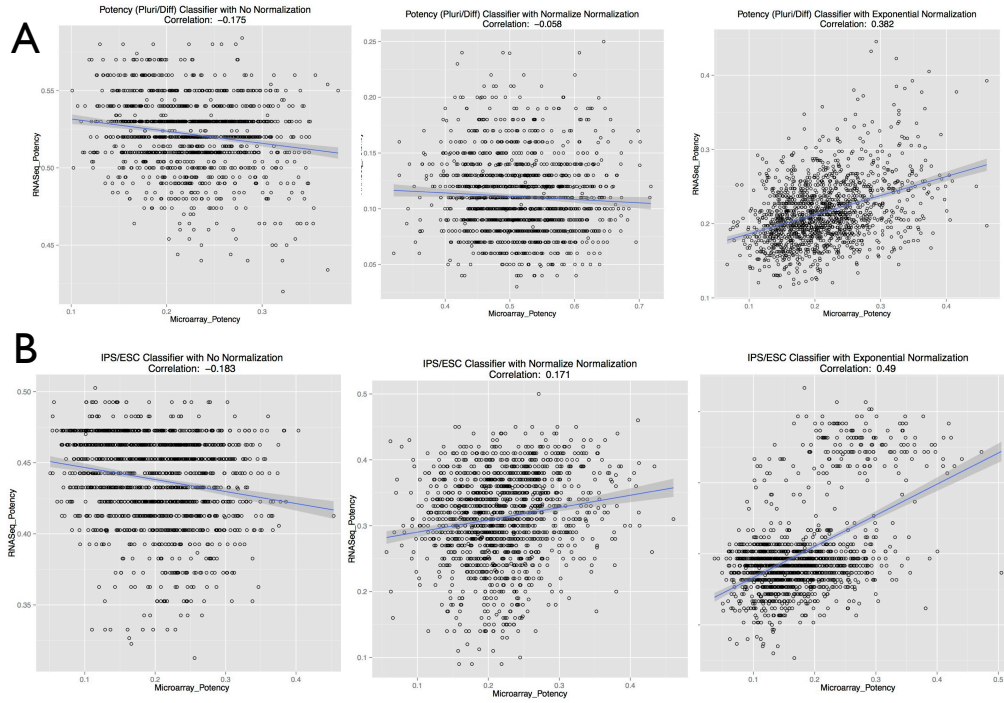


Figure A.2: In (A) classifiers were trained to predict whether a cell is pluripotent or fully differentiated. The Spearman's ranked correlation between classifier calls were -0.175, -0.058, and 0.382 for no normalization, normalize normalization, and exponential normalization respectively. In (B) classifiers were trained to identify induced pluripotent stem cells versus embryonic stem cells. In this case the correlation between the microarray model calls and the rnaseq model calls were -0.183, 0.171, and 0.49 respectively for no normalization, normalize normalization, and exponential normalization. In a third case, not shown, models were trained to distinguish between early embryonic stem cells and partially differentiated but still multi-potent stem cells. In this case the correlations were 0.009, 0.163, and 0.357 for no normalization, normalize, and exponential normalization.

A.0.4 Semantic Attribute Classification with Exponential Normalization

In this experiment, a set of 120 classifiers for MEMO events, mutations, and clinical measurements were applied to 634 KIRC, GBM, OV, and LUSC samples with survival data (the pancan prediction working group’s dichotomized set downloaded from Synapse). The output of these classifiers was used as the input for a combined pancancer survival classifier as semantic attributes. High/Low survival was predicted for each sample in a 5x 5X CV model selection experiment. A number of normalization schemes were attempted, including standardize, normalize, exponential normalization, and each of those preceded by a log transformation. The plot below Figure A.3 shows that in this case, the best chained predictions came from exponential normalized samples, though the best median performance comes from standardize. In these plots the five sets of CV folds were fixed for every point so that what is varying in the distribution is the choice of attribute selectors, classifiers, classifier parameters, and number of features.

In all three of these experiments the goal was validating exponential normalization as a usable normalization for the semantic attribute applications of this work. Given the good performance of this normalization in these tasks, as part of this thesis I will evaluate this normalization scheme against other schemes in a more systematic way starting with more direct experiments and including a variety of choices for distribution mapping.

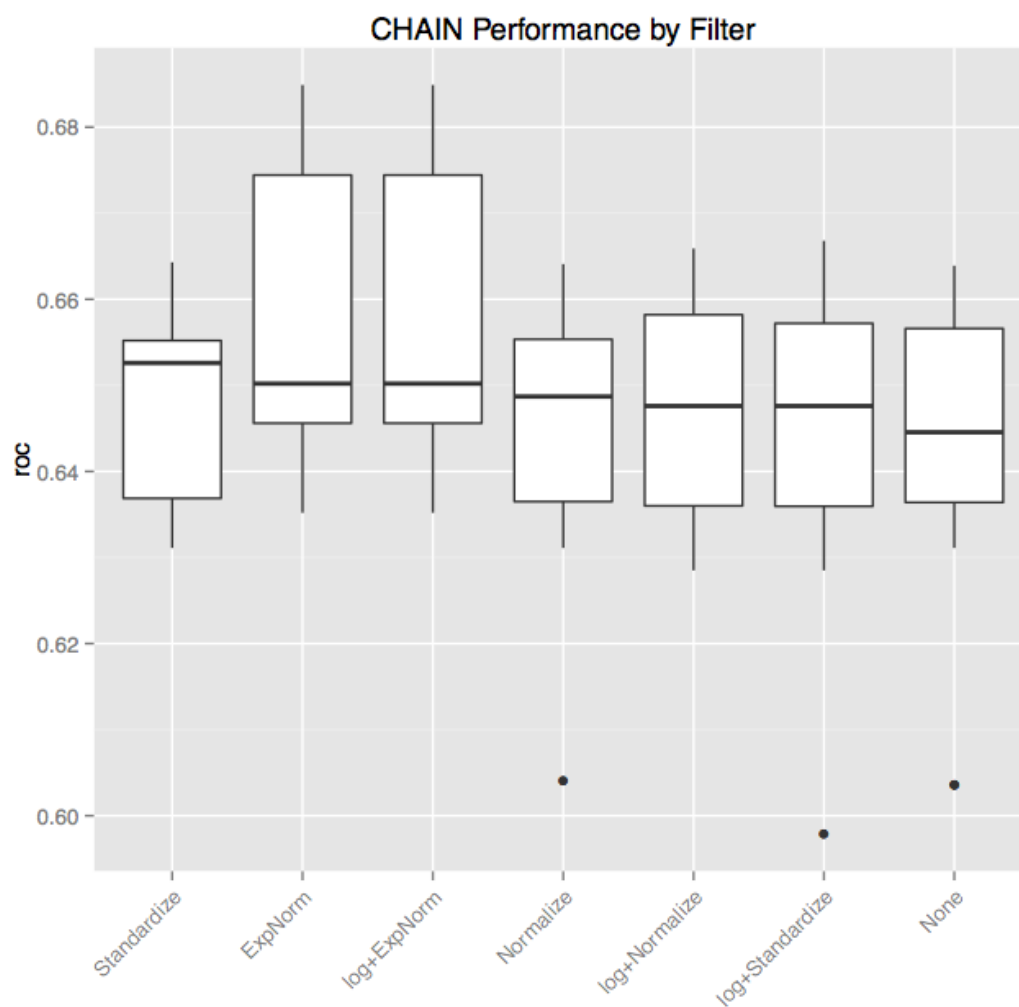


Figure A.3: Exponential Normalization in Semantic Attribute Classification (previously called "classifier chaining")

Appendix B

WekaMine Components and Usage

B.1 WekaMine Scripts

B.2 WekaMine Experiment DSL

B.2.1 Experiments

A wekaMine config file documents one or more experiments. An experiment has the following fields:

- classifier (e.g. SMO, etc.)
- attributeEvaluation (e.g. InformationGain)
- attributeSearch (e.g. Ranker)
- numAttributes for attribute evaluator to return.
- class Attribute (e.g. TIMEsurvival)

- discretization (e.g. median, quartile, bimodality-dichotimization, etc.)

Each of these fields can have a wide range of parameters and even include whole other algorithms. For example, support vector machines have a variety of parameters, and also a variety of kernel algorithms to choose from. Boosting classifiers are meta-classifiers, and specify whole other classifiers as one of their parameters. The range of possible combinations explodes very quickly. In order to help keep this multi-dimensional search through algorithm space manageable, wekaMine implements a string substitution configuration file.

B.2.2 Configuration File

The configuration file has a few basic principles:

Terms Any word followed by "`= [list]`" will expand to that list

Keyword Any word beginning with a `$` is a keyword and will attempt to be expanded from previously defined terms.

Range Any pair of braces is assumed to contain a start, stop, and increment to define a range of values: `start,stop,increment`

B.2.3 Example

This is probably easiest to see with an example. Consider a set of experiments where we look at `TIMESurvival` and `PLATINUM_FREE_INTERVAL_MONTHS`. Say we want to look at SMO using the default kernel trying three different values. Additionally, we want to try FisherLD and InformationGain as attribute selectors. A config file for this might look like:

```

expand{
    eval = ['durbin.weka.FisherLDEval','weka.attributeSelection.InfoGainAttributeEval']
    search = ['weka.attributeSelection.Ranker']
    classifier = ['weka.classifiers.functions.SMO -M -C {1,3,1} -V -1 -W 1']
    classAttribute = ['PLATINUM_FREE_INTERVAL_MONTHS','TIMESurvival']
    discretization = [median]
    experiments = [
        '$atEval,$atSearch,$numAttributes,$classifier,$classAttr,$discretization'
    ]
}

```

This configuration file would produce the following list of experiments:

```

durbin.weka.FisherLDEval,Ranker,weka.classifiers.functions.SMO -C 1 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,PLATINUM_FREE_INTERVAL_MONTHS,median
durbin.weka.FisherLDEval,Ranker,weka.classifiers.functions.SMO -C 2 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,PLATINUM_FREE_INTERVAL_MONTHS,median
durbin.weka.FisherLDEval,Ranker,weka.classifiers.functions.SMO -C 3 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,PLATINUM_FREE_INTERVAL_MONTHS,median
durbin.weka.FisherLDEval,Ranker,weka.classifiers.functions.SMO -C 1 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,TIMESurvival,median
durbin.weka.FisherLDEval,Ranker,weka.classifiers.functions.SMO -C 2 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,TIMESurvival,median
durbin.weka.FisherLDEval,Ranker,weka.classifiers.functions.SMO -C 3 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,TIMESurvival,median
InfoGainAttributeEval,Ranker,weka.classifiers.functions.SMO -C 1 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,PLATINUM_FREE_INTERVAL_MONTHS,median
InfoGainAttributeEval,Ranker,weka.classifiers.functions.SMO -C 2 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,PLATINUM_FREE_INTERVAL_MONTHS,median
InfoGainAttributeEval,Ranker,weka.classifiers.functions.SMO -C 3 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,PLATINUM_FREE_INTERVAL_MONTHS,median
InfoGainAttributeEval,Ranker,weka.classifiers.functions.SMO -C 1 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,TIMESurvival,median
InfoGainAttributeEval,Ranker,weka.classifiers.functions.SMO -C 2 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,TIMESurvival,median
InfoGainAttributeEval,Ranker,weka.classifiers.functions.SMO -C 3 -L 0.0001 -P 1.0E-12 -N 0 -V -1 -W 1,200,TIMESurvival,median

```

So in this case, our config file expanded to 12 experiments.

B.2.4 Special Keywords

Specifying "none" as the classifier does attribute selection only. Specifying "none" as the discretization similarly performs no discretization. One can give explicit discretization cutoffs, for example here are three different cutoffs to try:

```
discretization = ['5;10','4.5;5','4;12']
```

B.2.5 More Realistic Example

A more realistic example follows. In this example we are trying a variety of SMO kernels, a variety of classifier algorithms, a variety of attribute selectors, several discretizations, over several class attributes. This configuration file specifies over 300 separate experiments.

```

// An actual working configuration example..
//
//
// Expand section contains all the keyword and number range expansions that will
// generate a list of experiments. That is, all
// per-experiment variables will be documented in expand.
expand{
    atSearch=['weka.attributeSelection.Ranker']
    atEval=['weka.attributeSelection.GainRatioAttributeEval',
            'weka.attributeSelection.OneRAttributeEval -S 1 -F 10 -B 6',
            'weka.attributeSelection.InfoGainAttributeEval'
    ]

    // Try 1st,2nd,3rd, and 4th degree polynomials kernels.
    kernel = [
        'weka.classifiers.functions.supportVector.PolyKernel -C 250007 -E {1,4,1}'
    ]

    // Try SMO with C values from 1 to 3, and for each of those try all four
    // kernels specified in the kernel section
    //
    classifier = [
        'weka.classifiers.functions.SMO -M -C {1,3,1} -V -1 -W 1 -K "$kernel"',
        'weka.classifiers.trees.J48 -C 0.25 -M 2',
        'weka.classifiers.trees.RandomForest -I 2 -K 0 -S 1',
        'weka.classifiers.misc.HyperPipes',
        'weka.classifiers.bayes.NaiveBayes'
    ]

    // Clinical attributes to try to predict. Can list them individually,
    // or use keyword 'ALL' to indicate to classify on ALL clinical parameters.
    classAttr = ['TIMESurvival','TIMEmeta','TIMErecurrence']

    // Different ways to discretize the class variable.
    discretization = ['median','quartile','4.99;5.0']

    // Or, if no discretization is desired, you can specify none or nominal:
    // none means treat numbers as numeric and non-numbers as nominal
    // nominal means treat everything as nominal (i.e. numbers are nominal codes)
    //discretization = ['nominal']

```

```

    // -1 says to let attribute eval choose number of attributes.
    numAttributes = ['20','100','200','1000','-1']

    experiments = [
        '$atEval,$atSearch,$numAttributes,$classifier,$classAttr,$discretization'
    ]
}

params{
    censoredAttribute = "EVENTdeath"
    cvFolds = 5
    // -1 seed means seed with time clock...
    cvSeed = -1
}

```

Appendix C

Semantic Classifier Performance

Table C.1: Performance of 117 classifiers compared to majority classifier accuracy. ROC value is average ROC across a 5 times 5x Cross-validation experiment.

LAMLNPM1_NonSilent	0.984	0.537	0.447
BRCATP53_NonSilent	0.912	0.549	0.363
BLCATP53_NonSilent	0.867	0.508	0.359
GBM_EGFR_HYPER_NODE_AMPLIFICATION	0.873	0.523	0.350
BRCAPIK3CA_NonSilent	0.857	0.521	0.336
KIRCgender	0.987	0.656	0.331
LAMLlab_procedure_monocyte_result_percentage	0.865	0.535	0.330
UCECPTEN_NonSilent	0.863	0.541	0.322
COADREAD_TP53_HYPER_NODE_MUTATION	0.828	0.520	0.308
UCECCTNNB1_NonSilent	0.977	0.678	0.299
OV_MYC_HYPER_NODE_AMPLIFICATION	0.826	0.531	0.294
GBM_KLHL9HYPER_NODE_DELETION	0.873	0.586	0.287
BRCA_TP53_HYPER_NODE_MUTATION	0.911	0.630	0.281
LUADTP53_NonSilent	0.871	0.593	0.278
COADREADTP53_NonSilent	0.835	0.557	0.277
LUSCgender	1.000	0.726	0.274
KIRC_PBRM1_HYPER_NODE_MUTATION	0.877	0.608	0.269
GBMEGFR_NonSilent	0.858	0.607	0.250
BRCAbreast_carcinoma_progesterone_receptor	0.883	0.634	0.249
COADREADKRAS_NonSilent	0.807	0.566	0.241
KIRCPBRM1_NonSilent	0.831	0.591	0.239
UCECTP53_NonSilent	0.938	0.700	0.238

Model	Average ROC	Majority Accuracy	Improvement Over Majority
UCEC_PTEN_HYPER_NODE_MUTATION	0.863	0.633	0.230
BLCA_C9orf53_CDKN2A_HYPER_NODE_	0.880	0.651	0.229
KIRCtumor_stage_Stage I_vs_notStage I	0.746	0.521	0.225
UCEC_CTNNB1_HYPER_NODE_MUTATION	0.925	0.702	0.223
COADREAD_KRAS_HYPER_NODE_MUTATION	0.790	0.567	0.222
BRCAbreast_carcinoma_progesterone_recepto	0.910	0.688	0.222
UCEC_TP53_HYPER_NODE_MUTATION	0.929	0.716	0.213
BRCAdays_to_death_median	0.738	0.524	0.213
BLCA_TP53_HYPER_NODE_MUTATION	0.704	0.500	0.204
LAMLdays_to_death_median	0.721	0.521	0.200
LAML_NPM1_HYPER_NODE_MUTATION	0.947	0.750	0.197
LUADKRAS_NonSilent	0.848	0.652	0.196
HNSCtumor_stage_Stage IVA_vs_notStage IVA	0.700	0.509	0.191
OVdays_to_death_median	0.715	0.526	0.190
LUSC_C9orf53_CDKN2A_HYPER_NODE_	0.912	0.722	0.190
BLCAdays_to_death_median	0.695	0.522	0.174
BRCAlab_proc_her2_neu_immunohistochemi	0.700	0.527	0.173
BRCAbreast_carcinoma_estrogen_receptor_st	0.963	0.793	0.170
GBMTP53_NonSilent	0.777	0.615	0.161
BRCAbreast_carcinoma_estrogen_receptor_st	0.899	0.738	0.161
HNSC_C9orf53_CDKN2A_HYPER_NODE_	0.890	0.729	0.161
LUSC_WHSC1L1_LETM2_HYPER_NODE_	0.950	0.796	0.154
LUADKEAP1_NonSilent	0.939	0.793	0.146
OV_MECOM_HYPER_NODE_AMPLIFICATION	0.812	0.672	0.140
BRCA_CCND1_HYPER_NODE_AMPLIFICATION	0.941	0.801	0.139
GBM_TP53_HYPER_NODE_MUTATION	0.811	0.672	0.139
LAMLDNMT3A_NonSilent	0.752	0.613	0.139
BRCAdays_to_death_lowerquartile	0.863	0.725	0.138
LUADdays_to_death_median	0.656	0.521	0.136
BRCA_ERBB2_HYPER_NODE_AMPLIFICATION	0.991	0.856	0.134
UCECdays_to_death_median	0.714	0.583	0.131
HNSC_TP53_HYPER_NODE_MUTATION	0.840	0.716	0.124
BRCA_MYC_HYPER_NODE_AMPLIFICATION	0.857	0.740	0.117
GBM_CDK4_MARCH9_TSPAN31_HYPER_	0.916	0.802	0.114
OV_CCNE1_HYPER_NODE_AMPLIFICATION	0.898	0.787	0.111
LUSC_ANO1_HYPER_NODE_AMPLIFICATION	0.983	0.875	0.108
LAMLRUNX1_NonSilent	0.959	0.853	0.106

Model	Average ROC	Majority Accuracy	Improvement Over Majority
BLCANFE2L2_NonSilent	0.983	0.892	0.090
BLCAARID1A_NonSilent	0.777	0.688	0.090
LAML_FLT3_HYPER_NODE_MUTATION	0.833	0.745	0.088
LUAD_ARNT_SETDB1_HYPER_NODE_A	0.921	0.836	0.085
KIRCdays_to_death_median	0.593	0.511	0.082
BRCA_TUBD1_HYPER_NODE_AMPLIFIC	0.952	0.872	0.080
HNSC_WHSC1L1_LETM2_HYPER_NODE_	0.989	0.909	0.080
BLCAajcc_cancer_metastasis_stage_code_M	0.674	0.598	0.076
LUADdistant_metastasis_pathologic_spread_	0.710	0.635	0.075
UCEC_ARID1A_HYPER_NODE_MUTATIC	0.735	0.662	0.073
OV_CACNA1A_BRD4_NOTCH3_HYPER_	0.916	0.843	0.073
KIRC_VHL_HYPER_NODE_MUTATION	0.649	0.577	0.073
GBMdays_to_death_median	0.605	0.537	0.068
HNSCdays_to_death_median	0.573	0.505	0.068
KIRC_SETD2_HYPER_NODE_MUTATION	0.935	0.867	0.068
OV_SAMD4B_HYPER_NODE_AMPLIFICA	0.917	0.849	0.068
KIRC_KDM5C_HYPER_NODE_MUTATIO	0.984	0.919	0.065
BLCAajcc_cancer_metastasis_stage_code_M	0.653	0.588	0.064
BRCA_KCTD14_HYPER_NODE_AMPLIFI	0.955	0.892	0.063
LUSCdays_to_death_median	0.636	0.575	0.061
LUAD_SFTA3_HYPER_NODE_AMPLIFIC	0.920	0.860	0.060
BRCA_ZNF703_HYPER_NODE_AMPLIFIC	0.925	0.866	0.060
LAML_DNMT3A_HYPER_NODE_MUTATI	0.814	0.760	0.053
KIRCSETD2_NonSilent	0.905	0.852	0.053
UCECPIK3CA_NonSilent	0.567	0.517	0.049
LAMLIDH2_NonSilent	0.879	0.833	0.046
LUADdistant_metastasis_pathologic_spread_	0.733	0.689	0.044
UCEC_KRAS_HYPER_NODE_MUTATION	0.834	0.795	0.039
UCECdays_to_death_lowerquartile	0.798	0.760	0.038
COADREADBRAF_NonSilent	0.956	0.918	0.038
LUSC_NFE2L2_HYPER_NODE_MUTATIO	0.889	0.853	0.037
BRCAGATA3_NonSilent	0.917	0.882	0.035
KIRCBAP1_NonSilent	0.909	0.874	0.035
BRCA_GOLPH3L_MCL1_ENSA_HYPER_	0.860	0.825	0.035
BLCAF7XW7_NonSilent	0.956	0.922	0.034
UCECARID1A_NonSilent	0.759	0.725	0.033
LUSCNFE2L2_NonSilent	0.879	0.848	0.031
UCECKRAS_NonSilent	0.871	0.841	0.030

Model	Average ROC	Majority Accuracy	Improvement Over Majority
BRCA_ZNF217_HYPER_NODE_AMPLIFIC	0.926	0.905	0.020
HNSC_EGFR_HYPER_NODE_AMPLIFICA	0.918	0.898	0.019
LUAD_C9orf53_CDKN2A_HYPER_NODE_	0.823	0.805	0.018
UCEC_PIK3R1_HYPER_NODE_MUTATIO	0.690	0.673	0.017
GBMPTEN_NonSilent	0.632	0.615	0.016
UCEC_MECOM_HYPER_NODE_AMPLIFI	0.925	0.910	0.015
LAMLIDH1_NonSilent	0.855	0.840	0.014
BRCA_GATA3_HYPER_NODE_MUTATIO	0.909	0.896	0.013
OV_KRAS_HYPER_NODE_AMPLIFICATI	0.852	0.840	0.012
HNSCdays_to_death_lowerquartile	0.743	0.731	0.011
LUSC_BCL11A_HYPER_NODE_AMPLIFIC	0.891	0.881	0.010

Appendix D

Drug Targets And Activities

Table D.1: COMBINED Drug Classifier ROCs and Their Targets

VX-680	0.9108	AURKA;AURKB;AURKC	inhibitor of aurora kinases
GSK-J4	0.9203	KDM6A;KDM6B	inhibitor of lysine-specific demethylases
belinostat	0.939	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
vorinostat	0.9203	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
BRD-A94377914	0.9012	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
narciclasine	0.9211	RHOA	activates cellular activity of RhoA; modulator of Rho/Rho kinase/LIM kinase/cofilin signaling
BRD-K33514849	0.7842		screening hit
LBH-589	0.9151	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
teniposide	0.8841	TOP2A;TOP2B	inhibitor of topoisomerase II
apicidin	0.8923	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
tipifarnib-P2	0.8692	FNTA	inhibitor of farnesyltransferase
PRIMA-1	0.8883	TP53	re-activator of the pro-apoptotic activity of mutant p53

drug	roc	target	activity
piperlongumine	0.8799		natural product; modulator of ROS levels
PX-12	0.8483	TXN	inhibitor of thioredoxin-1
PL-DI	0.8869		dimer of piperlongumine; inducer of ROS
KW-2449	0.8992	AURKA;FLT3	inhibitor of FLT3 and AURKA
BRD-K34222889	0.8722		analog of the natural product piperlongumine
entinostat	0.8926	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
LY-2183240	0.8555	FAAH	inhibitor of fatty acid amide hydrolase; inhibitor of anandamide uptake
KPT185	0.8855	XPO1	inhibitor of exportin 1
necrosulfonamide	0.855		inhibitor of downstream signaling of RIP3 associated with MLKL
doxorubicin	0.8601	TOP2A	inhibitor of topoisomerase II
ISOX	0.8494	HDAC6	inhibitor of HDAC6
LRRK2-IN-1	0.7854	DCLK1;LRRK2	inhibitor of leucine-rich repeat kinase 2; inhibitor of doublecortin-like kinase
topotecan	0.8576	TOP1	inhibitor of topoisomerase I
etoposide	0.8297	TOP2A	inhibitor of topoisomerase II
sunitinib	0.8294	FLT1;FLT3;KDR;KIT;PDGFRA;PDGFR	inhibitor of VEGFRs, c-KIT, and PDGFR alpha and beta
skepinone-L	0.849	MAPK14	inhibitor of p38 MAPK
SR-II-138A	0.8576	EIF4A2;EIF4E;EIF4G1	silvestrol analog; inhibits translation by modulating the eIF4F complex
GSK461364	0.877	PLK1	inhibitor of polo-like kinase 1 (PLK1)
cerulenin	0.8385	FASN;HMGCS1	inhibitor of fatty acid synthase; inhibitor of HMG-CoA synthase
crizotinib	0.8599	ALK;MET	inhibitor of c-MET and ALK
tacedinaline	0.8147	HDAC1;HDAC2;HDAC3;HDAC6;HDAC8	inhibitor of HDAC1, HDAC2, HDAC3, HDAC6, and HDAC8
BI-2536	0.8706	PLK1	inhibitor of polo-like kinase 1 (PLK1)
TG-101348	0.8444	JAK2	inhibitor of Janus kinase 2
NVP-BSK805	0.8486	JAK2	inhibitor of Janus kinase 2
NSC95397	0.8013	CDC25A;CDC25B;CDC25C	inhibitor of cell division cycle 25 phosphatase (CDC25)
methylstat	0.8447	KDM3A;KDM4A;KDM4B;KDM4C;KDM4D	inhibitor of lysine specific demethylases

drug	roc	target	activity
CR-1-31B	0.8251	EIF4A2;EIF4E;EIF4G1	silvestrol analog; inhibits translation by modulating the eIF4F complex
AZD7762	0.8235	CHEK1;CHEK2	inhibitor of checkpoint kinases 1 and 2
AT13387	0.8368	HSP90AA1	inhibitor of HSP90
BRD-K26531177	0.811		analog of the natural product piperlongumine
PF-3758309	0.8271	PAK4	inhibitor of serine/threonine p21-activating kinase 4
BIX-01294	0.818	EHMT2	inhibitor of G9a histone methyltransferase
bortezomib	0.7888	PSMB1;PSMB2;PSMB5;PSMD1;PSMD	inhibitor of 26S proteasome
AZD1480	0.7933	JAK1;JAK2	inhibitor of Janus kinases 1 and 2
SCH-79797	0.8406	F2R	antagonist of proteinase-activated receptor 1 (PAR1)
GW-405833	0.8197	CNR2	partial agonist of cannabinoid receptor 2
chlorambucil	0.7817		DNA alkylator
fingolimod	0.8055	S1PR1	inhibitor of sphingosine 1-phosphate receptor
SB-225002	0.8231	CXCR2	inhibitor of chemokine receptor 2
NSC632839	0.801	USP13;USP5	inhibitor of ubiquitin isopeptidase
ceranib-2	0.813	ACER1;ACER2;ACER3;ASAH1;ASAH	inhibitor of ceramidase activity
SNX-2112	0.7982	HSP90AA1;HSP90B1	inhibitor of HSP90alpha and HSP90beta
phloretin	0.8038	SLC5A1	natural product; inhibitor of glucose uptake
SN-38	0.7723	TOP1	metabolite of irinotecan; inhibitor of topoisomerase I
MLN2238	0.7998	PSMB5	inhibitor of 20S proteasome at the chymotrypsin-like proteolytic (beta-5) site
tivantinib	0.8413	MET	inhibitor of MET; inhibitor of microtubule assembly
serdemetan	0.7603	MDM2	inhibitor of MDM2
MG-132	0.7927	PSMB1;PSMB2;PSMB5;PSMD1;PSMD	inhibitor of the proteasome
obatoclax	0.8175	BCL2;BCL2L1;MCL1	inhibitor of MCL1, BCL2, and BCL-xL
PF-573228	0.812	PTK2	inhibitor of focal adhesion kinase
valdecoxib	0.7926	PTGS2	inhibitor of cyclooxygenase-2 (COX2)

drug	roc	target	activity
nutlin-3	0.8653	MDM2	inhibitor of p53-MDM2 interaction
NVP-TAE684	0.8032	ALK	inhibitor of ALK and ALK-NPM fusion protein
mitomycin	0.7933		DNA crosslinker
dinaciclib	0.8026	CDK1;CDK2;CDK5;CDK9	inhibitor of cyclin-dependent kinases
masitinib	0.7736	KIT;PDGFRA;PDGFRB	inhibitor of c-KIT, PDGFRA, and PDGFRB
bosutinib	0.8288	ABL1;SRC	inhibitor of SRC and ABL1
SB-743921	0.8266	KIF11	inhibitor of kinesin 11
clofarabine	0.8146		inducer of DNA damage
JQ-1	0.7649	BRDT	inhibitor of bromodomain (BRD) and extra-C terminal domain (BET) proteins
PIK-93	0.772	PIK3CG	inhibitor of PI3K catalytic subunit gamma
WP1130	0.783	UCHL5;USP14;USP5;USP9X	inhibitor of the deubiquitinase activity of USP9X, USP5, USP14, and UCH37
curcumin	0.8145		natural product; modulator of ROS; modulator of NF-kappa-B signaling
CIL55A	0.7512		screening hit
BMS-754807	0.8071	IGF1R	inhibitor of insulin-like growth factor 1 receptor and insulin receptor
docetaxel	0.7824		inhibitor of microtubule assembly
STF-31	0.7849	NAMPT	inhibitor of nicotinamide phosphoribosyltransferase
PHA-793887	0.8116	CDK1;CDK2;CDK4;CDK5;CDK7;CDK	inhibitor of cyclin-dependent kinases
MST-312	0.8182	TERT	inhibitor of telomerase reverse transcriptase
BRD-A86708339	0.7644		screening hit
gossypol	0.7595	BCL2;BCL2L1;LDHA;LDHB;LDHC	inhibitor of lactate dehydrogenase; inhibitor of BCL2 family members
daporinad	0.7762	NAMPT	inhibitor of nicotinamide phosphoribosyltransferase
PDMP	0.7663	UGCG	inhibitor of ceramide glucosyltransferase
foretinib	0.8018	KDR;MET	inhibitor of MET and VEGFR2
vincristine	0.7716		inhibitor of microtubule assembly
PAC-1	0.7749	CASP3	activator of procaspase-3

drug	roc	target	activity
NSC23766	0.7746	RAC1;TIAM1;TRIO	inhibitor of RAC1-GEF interaction; prevents Rac1 activation by Rac-specific guanine nucleotide exchange factors (GEFs) TrioN and Tiam1
BMS-345541	0.7981	IKBKB	inhibitor of IKK-2
SCH-529074	0.7804	TP53	activator of mutant p53
linifanib	0.8065	FLT1;FLT3;KDR	inhibitor of VEGFRs
NVP-ADW742	0.7539	IGF1R	inhibitor of insulin-like growth factor 1 receptor
ML162	0.8267		selectively kills engineered cells expressing mutant HRAS
axitinib	0.7767	FLT1;FLT3;KDR;KIT;PDGFRA;PDGFR	inhibitor of VEGFRs, c-KIT, and PDGFR alpha and beta
CAY10618	0.8013	NAMPT	inhibitor of nicotinamide phosphoribosyltransferase
alisertib	0.7789	AURKA;AURKB	inhibitor of aurora kinases A and B
gemcitabine	0.7522	CMPK1;RRM1;TYMS	inhibitor of DNA replication; inhibitor of ribonucleotide reductase, thymidylate synthetase, and cytidine monophosphate (UMP-CMP) kinase
MK-2206	0.7517	AKT1	inhibitor of AKT1
WZ8040	0.7909	EGFR	inhibitor of EGFR targeting T790M resistance
SMER-3	0.7672	CUL1;SKP1	inhibitor of E3-ubiquitin ligase
tanespimycin	0.8238	HSP90AA1	inhibitor of HSP90
AT7867	0.7749	AKT1;AKT2;AKT3;RPS6KB2	inhibitor of AKT1/2/3 and S6K
AZD4547	0.7745	FGFR1;FGFR2;FGFR3	inhibitor of fibroblast growth factor receptors
YM-155	0.806	BIRC5	inhibitor of survivin expression
CD-437	0.7561	RARG	agonist of retinoic acid receptor gamma
dasatinib	0.8048	EPHA2;KIT;LCK;SRC;YES1	inhibitor of SRC, YES1, EPHA2, c-KIT, and LCK
ML210	0.774		selectively kills engineered cells expressing mutant HRAS
triptolide	0.7531		natural product; inhibitor of RNA polymerase II
ouabain	0.753	ATP1A1;ATP1A2;ATP1A3;ATP1A4;ATP1A5	cardiac glycoside; inhibitor of the Na ⁺ /K ⁺ -ATPase

drug	roc	target	activity
trametinib	0.7882	MAP2K1;MAP2K2	inhibitor of MEK1 and MEK2
SNS-032	0.772	CDK16;CDK17;CDK2;CDK7;CDK9;CDK10	inhibitor of cyclin-dependent kinases
PD318088	0.7787	MAP2K1;MAP2K2	inhibitor of MEK1 and MEK2

Bibliography

- [1] Early Breast Cancer Trialists' Collaborative Group. "Polychemotherapy for early breast cancer: an overview of the randomised trials". English. In: *The Lancet* 352.9132 (Sept. 1998), pp. 930–942. DOI: [10.1016/S0140-6736\(98\)03301-7](https://doi.org/10.1016/S0140-6736(98)03301-7). URL: <http://linkinghub.elsevier.com/retrieve/pii/S0140673698033017>.
- [2] Kathy S Albain et al. "Adjuvant chemotherapy and timing of tamoxifen in postmenopausal patients with endocrine-responsive, node-positive breast cancer: a phase 3, open-label, randomised controlled trial". In: *The Lancet* 374.9707 (Dec. 2009), pp. 2055–2063.
- [3] Alastair J J Wood, Charles L Shapiro, and Abram Recht. "Side Effects of Adjuvant Treatment of Breast Cancer". In: *The New England journal of medicine* 344.26 (June 2001), pp. 1997–2008.
- [4] Soonmyung Paik et al. "A Multigene Assay to Predict Recurrence of Tamoxifen-Treated, Node-Negative Breast Cancer". In: *The New England journal of medicine* 351.27 (Dec. 2004), pp. 2817–2826.
- [5] DC Allred, Jennet M Harvey, Melora Berardo, and Gary M Clark. "Prognostic and predictive factors in breast cancer by immunohistochemical analysis." In: *Modern pathology: an official journal of the United States and Canadian Academy of Pathology, Inc* 11.2 (1998), p. 155.
- [6] Patrick L Fitzgibbons et al. "Prognostic Factors in Breast Cancer". In: *archivesofpathology.org* ().
- [7] B van der Vegt, G H de Bock, H Hollema, and J Wesseling. "Microarray methods to identify factors determining breast cancer progression: potentials, limitations, and challenges." In: *Critical reviews in oncology/hematology* 70.1 (Apr. 2009), pp. 1–11.
- [8] Jorge S Reis-Filho and Lajos Pusztai. "Gene expression profiling in breast cancer: classification, prognostication, and prediction". In: *The Lancet* 378.9805 (Nov. 2011), pp. 1812–1823.
- [9] John S Bertram. "The molecular biology of cancer". In: *Molecular Aspects of Medicine* 21.6 (Dec. 2000), pp. 167–223.

- [10] B Vogelstein et al. “Cancer Genome Landscapes”. In: *Science (New York, NY)* 339.6127 (Mar. 2013), pp. 1546–1558.
- [11] Laura J van ’t Veer et al. “Gene expression profiling predicts clinical outcome of breast cancer”. In: *Nature* 415.6871 (Jan. 2002), pp. 530–536.
- [12] Jolien M Bueno-de-Mesquita et al. “Use of 70-gene signature to predict prognosis of patients with node-negative breast cancer: a prospective community-based feasibility study (RASTER)”. In: *The Lancet Oncology* 8.12 (Dec. 2007), pp. 1079–1087.
- [13] Christos Sotiriou and Lajos Pusztai. “Gene-Expression Signatures in Breast Cancer”. In: *The New England journal of medicine* 360.8 (Feb. 2009), pp. 790–800.
- [14] T R Golub. “Molecular Classification of Cancer: Class Discovery and Class Prediction by Gene Expression Monitoring”. In: *Science (New York, NY)* 286.5439 (Oct. 1999), pp. 531–537.
- [15] David Venet, Jacques E Dumont, and Vincent Detours. “Most Random Gene Expression Signatures Are Significantly Associated with Breast Cancer Outcome”. In: *Plos Computational Biology* 7.10 (Oct. 2011), e1002240.
- [16] Grazia Arpino et al. “Gene expression profiling in breast cancer: A clinical perspective”. In: *The Breast* 22.2 (Apr. 2013), pp. 109–120.
- [17] Yudi Pawitan et al. “Breast Cancer Research | Full text | Gene expression profiling spares early breast cancer patients from adjuvant therapy: derived and validated in two population-based cohorts”. In: *Breast Cancer Research* 7.6 (2005), R953.
- [18] Maysa Abu-Khalf and Lajos Pusztai. “Influence of genomics on adjuvant treatments for pre-invasive and invasive breast cancer”. In: *The Breast* 22 (Aug. 2013), S83–S87.
- [19] A C Culhane et al. “GeneSigDB: a manually curated database and resource for analysis of gene expression signatures”. In: *Nucleic Acids Research* 40.D1 (Dec. 2011), pp. D1060–D1066.
- [20] A Liberzon et al. “Molecular signatures database (MSigDB) 3.0”. In: *Bioinformatics (Oxford, England)* 27.12 (June 2011), pp. 1739–1740.
- [21] Aravind Subramanian et al. “Gene set enrichment analysis: A knowledge-based approach for interpreting genome-wide expression profiles”. In: *pnas.org* ().
- [22] C J Vaske et al. “Inference of patient-specific pathway activities from multi-dimensional cancer genomics data using PARADIGM”. In: *Bioinformatics (Oxford, England)* 26.12 (June 2010), pp. i237–i245.

- [23] David F Ransohoff. “Gene-expression signatures in breast cancer.” In: *The New England journal of medicine* 348.17 (Apr. 2003), 1715–7– author reply 1715–7.
- [24] Yotam Drier and Eytan Domany. “Do Two Machine-Learning Based Prognostic Signatures for Breast Cancer Capture the Same Biological Processes?” In: *PLoS ONE* 6.3 (Mar. 2011), e17795.
- [25] L Ein-Dor, I Kela, G Getz, D Givol, and E Domany. “Outcome signature genes in breast cancer: is there a unique set?” In: *Bioinformatics (Oxford, England)* 21.2 (Jan. 2005), pp. 171–178.
- [26] Alain Dupuy and Richard M Simon. “Critical Review of Published Microarray Studies for Cancer Outcome and Guidelines on Statistical Analysis and Reporting”. In: *jnci.oxfordjournals.org* ().
- [27] Stefan Michiels, Serge Koscielny, and Catherine Hill. “Prediction of cancer outcome with microarrays: a multiple random validation strategy”. In: *The Lancet* 365.9458 (Feb. 2005), pp. 488–492.
- [28] Pratyaksha Wirapati et al. “Meta-analysis of gene-expression profiles in breast cancer: toward a unified understanding of breast cancer sub-typing and prognosis signatures”. In: *Breast Cancer Research* 10.4 (2008), R65.
- [29] Stefan Michiels, Serge Koscielny, and Catherine Hill. “Prediction of cancer outcome with microarrays: a multiple random validation strategy”. In: *Lancet* 365.9458 (2005), pp. 488–492.
- [30] Cheng Fan et al. “Concordance among Gene-Expression–Based Predictors for Breast Cancer”. In: *The New England journal of medicine* 355.6 (Aug. 2006), pp. 560–569.
- [31] L Ein-Dor. “Thousands of samples are needed to generate a robust gene list for predicting outcome in cancer”. In: *Proceedings of the National Academy of Sciences* 103.15 (Apr. 2006), pp. 5923–5928.
- [32] Fabien Reyat et al. “A comprehensive analysis of prognostic signatures reveals the high predictive capacity of the Proliferation, Immune response and RNA splicing modules in breast cancer”. In: *Breast Cancer Research* 10.6 (2008), R93.
- [33] Eunjung Lee, Han-Yu Chuang, Jong-Won Kim, Trey Ideker, and Doheon Lee. “Inferring Pathway Activity toward Precise Disease Classification”. In: *Plos Computational Biology* 4.11 (Nov. 2008), e1000217.
- [34] Wei-Yi Cheng, Tai-Hsien Ou Yang, and Dimitris Anastassiou. “Biomolecular Events in Cancer Revealed by Attractor Metagenes”. In: *Plos Computational Biology* 9.2 (Feb. 2013), e1002920.

- [35] W Y Cheng, T H O Yang, and D Anastassiou. “Development of a Prognostic Model for Breast Cancer Survival in an Open Challenge Environment”. In: *Science Translational Medicine* 5.181 (Apr. 2013), 181ra50–181ra50.
- [36] Robert Clarke et al. “The properties of high-dimensional data spaces: implications for exploring gene and protein expression data”. In: *Nature Reviews Cancer* 8.1 (Jan. 2008), pp. 37–49.
- [37] Antai Wang and Edmund A Gehan. “Gene selection for microarray data analysis using principal component analysis”. In: *Statistics in Medicine* 24.13 (2005), pp. 2069–2087.
- [38] I Guyon, J Weston, S Barnhill, and V Vapnik. “Gene selection for cancer classification using support vector machines”. In: *Machine Learning* (2002).
- [39] Pedro Domingos. “A few useful things to know about machine learning”. In: *Communications of the ACM* 55.10 (Oct. 2012), p. 78.
- [40] Han-Yu Chuang, Eunjung Lee, Yu-Tsueng Liu, Doheon Lee, and Trey Ideker. “Network-based classification of breast cancer metastasis”. In: *Molecular Systems Biology* 3 (Oct. 2007).
- [41] Ian W Taylor et al. “Dynamic modularity in protein interaction networks predicts breast cancer outcome.” In: *Nature Biotechnology* 27.2 (Feb. 2009), pp. 199–204.
- [42] Gad Abraham, Adam Kowalczyk, Sherene Loi, Izhak Haviv, and Justin Zobel. “Prediction of breast cancer prognosis using gene set statistics provides signature stability and biological context”. In: *BMC bioinformatics* 11.1 (2010), p. 277.
- [43] Christine Staiger et al. “A Critical Evaluation of Network and Pathway-Based Classifiers for Outcome Prediction in Breast Cancer”. In: *PLoS ONE* 7.4 (Apr. 2012), e34796.
- [44] G Ciriello, E Cerami, C Sander, and N Schultz. “Mutual exclusivity analysis identifies oncogenic network modules”. In: *Genome research* 22.2 (Feb. 2012), pp. 398–406.
- [45] G Joshi-Tope. “Reactome: a knowledgebase of biological pathways”. In: *Nucleic Acids Research* 33.Database issue (Dec. 2004), pp. D428–D432.
- [46] M Kanehisa. “KEGG: Kyoto Encyclopedia of Genes and Genomes”. In: *Nucleic Acids Research* 28.1 (Jan. 2000), pp. 27–30.
- [47] David Heckerman, Dan Geiger, and David M Chickering. “Learning Bayesian networks: The combination of knowledge and statistical data”. In: *Machine Learning* 20.3 (Sept. 1995), pp. 197–243.

- [48] F R Kschischang, B J Frey, and H A Loeliger. “Factor graphs and the sum-product algorithm”. In: *IEEE Transactions on Information Theory* 47.2 (2001), pp. 498–519.
- [49] James E Ferrell. “Q&A: Systems biology”. In: *Journal of Biology* 8.1 (2009), p. 2.
- [50] J P Brunet, P Tamayo, T R Golub, and J P Mesirov. “Metagenes and molecular pattern discovery using matrix factorization”. In: *Proceedings of the National Academy of Sciences* 101.12 (Mar. 2004), pp. 4164–4169.
- [51] *Sage Bionetworks-DREAM Breast Cancer Prognosis Challenge*. 2012. URL: <http://www.the-dream-project.org/challenges/sage-bionetworks-dream-breast-cancer-prognosis-challenge>.
- [52] Weinberg Robert A. *The Biology of Cancer*. New York,USA: Garland Science, 2007.
- [53] L Shi, G Campbell, W D Jones, and F Campagne. “The MicroArray Quality Control (MAQC)-II study of common practices for the development and validation of microarray-based predictive models.” In: *Nature* (2010).
- [54] J Luo, M Schumacher, and A Scherer. “A comparison of batch effect removal methods for enhancement of prediction performance using MAQC-II microarray gene expression data”. In: *The ...* (2010).
- [55] C Chen, K Grennan, J Badner, D Zhang, and E Gershon. “Removing batch effects in analysis of expression microarray data: an evaluation of six batch adjustment methods”. In: *PLoS ONE* (2011).
- [56] D G Lowe. “Object recognition from local scale-invariant features”. In: *Proceedings of the Seventh IEEE International Conference on Computer Vision*. IEEE, 1150–1157 vol.2.
- [57] Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. “Attribute and simile classifiers for face verification”. In: *ieeexplore.ieee.org* (2009), pp. 365–372.
- [58] D Parikh and K Grauman. “Interactively building a discriminative vocabulary of nameable attributes”. In: *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. 2011, pp. 1681–1688.
- [59] Yu Su and Frédéric Jurie. “Improving Image Classification Using Semantic Attributes”. In: *International Journal of Computer Vision* 100.1 (May 2012), pp. 59–77.
- [60] Gary B. Huang, Marwan Mattar, Tamara Berg, and Erik Learned-miller. *Labeled faces in the wild: A database for studying face recognition in unconstrained environments*. 2007. URL: <http://vis-www.cs.umass.edu/lfw/>.

- [61] Li-Jia Li, Hao Su, Yongwhan Lim, and Li Fei-Fei. “Objects as Attributes for Scene Classification”. In: *Trends and Topics in Computer Vision*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 57–69.
- [62] C Giraud-Carrier, F Provost of the ICML-2005 Workshop on, and 2005. “Toward a justification of meta-learning: Is the no free lunch theorem a show-stopper”. In: *researchgate.net* ().
- [63] K. James Durbin. *WekaMine, A machine learning toolkit for large scale model selection, training, and curation*. 2011. URL: <https://github.com/jdurbin/wekaMine/wiki>.
- [64] Mark Hall et al. “The WEKA data mining software”. In: *ACM SIGKDD Explorations Newsletter* 11.1 (Nov. 2009), pp. 10–18.
- [65] Eibe Frank, Mark Hall, Len Trigg, Geoffrey Holmes, and Ian H Witten. “Data mining in bioinformatics using Weka”. In: *Bioinformatics (Oxford, England)* 20.15 (Oct. 2004), pp. 2479–2481.
- [66] D Haussler. “Quantifying inductive bias: AI learning algorithms and Valiant’s learning framework”. In: *Artificial Intelligence* 36.2 (Sept. 1988), pp. 177–221.
- [67] Sounak Chakraborty Mihail Popescu Mohammed Khalilia. “Predicting disease risks from highly imbalanced data using random forest”. In: *BMC Medical Informatics and Decision Making* 11 (2011), p. 51.
- [68] Antti Airola, Tapio Pahikkala, Willem Waegeman, Bernard De Baets, and Tapio Salakoski. “A comparison of AUC estimators in small-sample studies”. In: (2009).
- [69] Tanya Barrett et al. “NCBI GEO: mining millions of expression profiles—database and tools”. eng. In: *Nucleic Acids Res* 33.Database issue (Jan. 2005), pp. D562–6. DOI: [10.1093/nar/gki022](https://doi.org/10.1093/nar/gki022). URL: http://nar.oxfordjournals.org/cgi/content/full/33/suppl_1/D562.
- [70] Ron Edgar, Michael Domrachev, and Alex E Lash. “Gene Expression Omnibus: NCBI gene expression and hybridization array data repository”. eng. In: *Nucleic Acids Res* 30.1 (Jan. 2002), pp. 207–10.
- [71] Olivier Bodenreider. “The Unified Medical Language System (UMLS): integrating biomedical terminology”. eng. In: *Nucleic Acids Res* 32.Database issue (Jan. 2004), pp. D267–70. DOI: [10.1093/nar/gkh061](https://doi.org/10.1093/nar/gkh061).
- [72] John D Osborne, Simon Lin, Lihua Zhu, and Warren A Kibbe. “Mining biomedical data using MetaMap Transfer (MMtx) and the Unified Medical Language System (UMLS)”. eng. In: *Methods Mol Biol* 408 (Jan. 2007), pp. 153–69.

- [73] Donna M Muzny et al. “Comprehensive molecular characterization of human colon and rectal cancer”. In: *Nature* 487.7407 (July 2012), pp. 330–337.
- [74] Xinsen Xu et al. “Assessing the clinical utility of genomic expression data across human cancers”. In: *Oncotarget* 7.29 (June 2016), pp. 45926–45936.
- [75] H Ishibashi. “Progesterone Receptor in Non-Small Cell Lung Cancer—A Potent Prognostic Factor and Possible Target for Endocrine Therapy”. In: *Cancer Research* 65.14 (July 2005), pp. 6450–6458.
- [76] Nadiyah Kazmi et al. “The role of estrogen, progesterone and aromatase in human non-small-cell lung cancer”. In: *Lung Cancer Management* 1.4 (Dec. 2012), pp. 259–272.
- [77] Michelle A Carey et al. “It’s all about sex: gender, lung development and lung disease”. In: *Trends in Endocrinology & Metabolism* 18.8 (Oct. 2007), pp. 308–313.
- [78] Robert Pirker et al. “EGFR expression as a predictor of survival for first-line chemotherapy plus cetuximab in patients with advanced non-small-cell lung cancer: analysis of data from the phase 3 FLEX study”. In: *The Lancet Oncology* 13.1 (Jan. 2012), pp. 33–42.
- [79] Charles Fert , Fabrice Andr , and Jean-Charles Soria. “Molecular circuits of solid tumors: prognostic and predictive tools for bedside use”. In: *Nature Reviews Clinical Oncology* 7.7 (June 2010), pp. 367–380.
- [80] Hiu Wing Cheung et al. “Systematic investigation of genetic vulnerabilities across cancer cell lines reveals lineage-specific dependencies in ovarian cancer”. In: *Proceedings Of The National Academy Of Sciences Of The United States Of America* 108.30 (July 2011), pp. 12372–12377.
- [81] Jordi Barretina et al. “The Cancer Cell Line Encyclopedia enables predictive modelling of anticancer drug sensitivity.” In: *Nature* 483.7391 (Mar. 2012), pp. 603–607.
- [82] Chayaporn Suphavilai, Denis Bertrand, and Niranjana Nagarajan. “Predicting Cancer Drug Response Using a Recommender System.” In: *Bioinformatics (Oxford, England)* (June 2018).
- [83] IN SOCK JANG, ELIAS CHAIBUB NETO, JUSTIN GUINNEY, Stephen H Friend, and ADAM A MARGOLIN. “SYSTEMATIC ASSESSMENT OF ANALYTICAL METHODS FOR DRUG SENSITIVITY PREDICTION FROM CANCER CELL LINE DATA”. In: *Proceedings of the Pacific Symposium*. WORLD SCIENTIFIC, Nov. 2013, pp. 63–74.

- [84] Young-suk Lee, Arjun Krishnan, Qian Zhu, and Olga G Troyanskaya. “Ontology-aware classification of tissue and cell-type signals in gene expression profiles across platforms and technologies.” In: *Bioinformatics (Oxford, England)* 29.23 (Dec. 2013), pp. 3036–3044.
- [85] Marion Gremse et al. “The BRENDA Tissue Ontology (BTO): the first all-integrating ontology of all organisms for enzyme sources.” In: *Nucleic Acids Research* 39.Database issue (Jan. 2011), pp. D507–13.
- [86] Peter A Jones. “Functions of DNA methylation: islands, start sites, gene bodies and beyond.” In: *Nature reviews Genetics* 13.7 (May 2012), pp. 484–492.
- [87] Zachary D Smith and Alexander Meissner. “DNA methylation: roles in mammalian development”. In: *Nature reviews Genetics* 14.3 (Feb. 2013), pp. 204–220.
- [88] Roadmap Epigenomics Consortium et al. “Integrative analysis of 111 reference human epigenomes.” In: *Nature* 518.7539 (Feb. 2015), pp. 317–330.
- [89] Jason Ernst and Manolis Kellis. “Chromatin-state discovery and genome annotation with ChromHMM.” In: *Nature protocols* 12.12 (Dec. 2017), pp. 2478–2492.
- [90] Jesse R Dixon et al. “Topological domains in mammalian genomes identified by analysis of chromatin interactions”. In: *Nature* 485.7398 (Apr. 2012), pp. 376–380.
- [91] Biao Luo et al. “Highly parallel identification of essential genes in cancer cells”. In: *Proceedings Of The National Academy Of Sciences Of The United States Of America* 105.51 (Dec. 2008), pnas.0810485105–20385.
- [92] Glenn S Cowley et al. “Parallel genome-scale loss of function screens in 216 cancer cell lines for the identification of context-specific genetic dependencies.” In: *Scientific data* 1 (2014), p. 140035.
- [93] Aviad Tsherniak et al. “Defining a Cancer Dependency Map”. In: *Cell* 170.3 (July 2017), 564–576.e16.
- [94] *Broad-DREAM Gene Essentiality Prediction Challenge*. 2014. URL: <https://www.synapse.org/#!Synapse:syn2384331/wiki/62825>.
- [95] Amrita Basu et al. “An interactive resource to identify cancer genetic and lineage dependencies targeted by small molecules.” In: *Cell* 154.5 (Aug. 2013), pp. 1151–1161.
- [96] Brinton Seashore-Ludlow et al. “Harnessing Connectivity in a Large-Scale Small-Molecule Sensitivity Dataset.” In: *Cancer discovery* 5.11 (Nov. 2015), pp. 1210–1223.

- [97] *The Broad Institute: Screening for Dependencies in Cancer Cell Lines Using Small Molecules*. 2015. URL: <https://ocg.cancer.gov/ctd2-data-project/broad-institute-screening-dependencies-cancer-cell-lines-using-small-molecules-0>.
- [98] *CTD2 Data Portal*. 2012. URL: <https://ocg.cancer.gov/programs/ctd2/data-portal>.
- [99] Kim D Pruitt, Tatiana Tatusova, William Klimke, and Donna R Maglott. “NCBI Reference Sequences: current status, policy and new initiatives.” In: *Nucleic Acids Research* 37.Database issue (Jan. 2009), pp. D32–6.
- [100] Elizabeth A Harrington et al. “VX-680, a potent and selective small-molecule inhibitor of the Aurora kinases, suppresses tumor growth in vivo.” In: *Nature Medicine* 10.3 (Mar. 2004), pp. 262–267.
- [101] Antonino B D’Assoro, Tufia Haddad, and Evanthia Galanis. “Aurora-A Kinase as a Promising Therapeutic Target in Cancer”. In: *Frontiers in Oncology* 5.40 (Jan. 2016), p. 65.
- [102] Anqun Tang et al. “Aurora kinases: novel therapy targets in cancers.” In: *Oncotarget* 8.14 (Apr. 2017), pp. 23937–23954.
- [103] *Vaadin Web Application Framework for Java*. 2010. URL: <https://vaadin.com>.
- [104] *Google Web Toolkit*. 2010. URL: <http://www.gwtproject.org/overview.html>.
- [105] *Allen Brain Atlas*. 2012. URL: <http://human.brain-map.org>.
- [106] M I Jordan and T M Mitchell. “Machine learning: Trends, perspectives, and prospects.” In: *Science (New York, NY)* 349.6245 (July 2015), pp. 255–260.
- [107] G E Hinton and R R Salakhutdinov. “Reducing the dimensionality of data with neural networks.” In: *Science (New York, NY)* 313.5786 (July 2006), pp. 504–507.
- [108] D P Kingma and M Welling. “Auto-encoding variational bayes [J]”. In: (2013).
- [109] *Deep Learning for Java*. 2015. URL: <https://deeplearning4j.org>.
- [110] B M Bolstad, R A Irizarry, M Astrand, and T P Speed. “A comparison of normalization methods for high density oligonucleotide array data based on variance and bias”. In: *Bioinformatics (Oxford, England)* 19.2 (Jan. 2003), pp. 185–193.