UNIVERSITY OF CALIFORNIA SAN DIEGO

# Investigation on the Effects of Curriculum Learning through a Simulation Study

A thesis submitted in partial satisfaction of the
requirements for the degree
Master of Science

in

Computer Science

by

Edward Wong

Committee in charge:

Professor Kamalika Chaudhuri, Chair
Professor Yoav Freund
Professor Ndapa Nakashole

2018

The thesis of Edward Wong is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California San Diego

2018

DEDICATION

To my uncles Alan and John, for their unconditional support throughout my life.

TABLE OF CONTENTS


Signature Page . . . iii

Dedication . . . iv

Table of Contents . . . v

List of Figures . . . vii

List of Tables . . . ix

Acknowledgements . . . x

Abstract of the Thesis . . . xi

Chapter 1 Introduction . . . 1
1.1 Background . . . 1
1.2 Problem Statement . . . 2

Chapter 2 Related Work . . . 4
2.1 Related Work on Curriculum Learning . . . 4
2.2 Related work on Alternative Forms of Learning . . . 5
2.2.1 Self-Paced Learning . . . 5
2.2.2 Active Learning . . . 6

Chapter 3 Synthetic Data With Linear Classifiers . . . 7
3.1 Introduction . . . 7
3.2 Problem Setting . . . 7
3.2.1 Stochastic Gradient Descent . . . 8
3.2.2 Exponentiated Gradient Descent . . . 8
3.2.3 Loss Function - Logistic Loss . . . 9
3.3 Methodology . . . 10
3.3.1 Data . . . 10
3.4 Curriculum Metrics . . . 10
3.4.1 Baselines . . . 13
3.5 Algorithms Selection . . . 14
3.6 Experimental Results . . . 16
3.6.1 Hardness from Margin . . . 16
3.6.2 Hardness from Irrelevant Features . . . 18
3.7 Discussion . . . 19
3.7.1 Hardness from Margin . . . 19
3.7.2 Hardness from Irrelevant Features . . . 20

Chapter 4 Real Data With Linear Classifiers . . . 22
4.1 Introduction . . . 22
4.2 Problem Setting . . . 22
4.2.1 Stochastic Gradient Descent . . . 23
4.2.2 Loss Function - Logistic Loss . . . 23
4.3 Methodology . . . 24
4.3.1 Data . . . 24
4.4 Curriculum Metrics . . . 24
4.4.1 Baselines . . . 25
4.5 Algorithm Selection . . . 26
4.6 Experimental Results . . . 26
4.7 Discussion . . . 34

Chapter 5 Neural Networks . . . 36
5.1 Introduction . . . 36
5.2 Problem Setting . . . 37
5.2.1 Loss Function - Cross Entropy Loss . . . 37
5.2.2 Activation Functions . . . 37
5.2.3 Back Propagation . . . 38
5.2.4 Output Layer - Softmax Regression . . . 38
5.3 Methodology . . . 39
5.3.1 Data . . . 39
5.4 Curriculum Metrics . . . 41
5.4.1 MNIST Dataset . . . 41
5.4.2 Shape Dataset . . . 41
5.4.3 Baselines . . . 42
5.4.4 Shape Dataset . . . 43
5.5 Neural Network Model Selection . . . 43
5.5.1 MNIST Dataset . . . 43
5.5.2 Shape Dataset . . . 45
5.6 Experimental Results . . . 45
5.6.1 MNIST Dataset . . . 45
5.6.2 Shape Dataset . . . 49
5.7 Discussion . . . 49

Chapter 6 Conclusion . . . 52
6.1 Future Work . . . 53

Bibliography . . . 54

LIST OF FIGURES


Figure 3.1: Objective Function Value Trace on Test Set . . . . . 15
Figure 3.2: Test Error Rate Trace . . . . . 15
Figure 3.3: Hardness measured by margin. $c = 1$, $\lambda = 0.1$. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard, Light Blue = Half-Half, Magenta = Easy-Half 15
Figure 3.4: Objective Function Value Trace on Test Set . . . . . 17
Figure 3.5: Test Error Rate Trace . . . . . 17
Figure 3.6: Hardness measured by $L_1$ of irrelevant features. $D = 16$, $c = 4$. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard . . . . . 17

Figure 4.1: Objective Function Value Trace on Test Set . . . . . 27
Figure 4.2: Test Error Rate Trace . . . . . 27
Figure 4.3: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c$ = 1. PCA reduced dimensionality to 50. Data is MNIST 3 v 5. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard . . . . . 27
Figure 4.4: Objective Function Value Trace on Test Set . . . . . 29
Figure 4.5: Test Error Rate Trace . . . . . 29
Figure 4.6: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c$ = 1. PCA reduced dimensionality to 75. Data is MNIST 3 v 5. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard . . . . . 29
Figure 4.7: Objective Function Value Trace on Test Set . . . . . 31
Figure 4.8: Test Error Rate Trace . . . . . 31
Figure 4.9: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c$ = 1. PCA reduced dimensionality to 50. Data is MNIST 1 v 8. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard . . . . . 31
Figure 4.10: Objective Function Value Trace on Test Set . . . . . 33
Figure 4.11: Test Error Rate Trace . . . . . 33
Figure 4.12: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c$ = 1. PCA reduced dimensionality to 75. Data is MNIST 1 v 8. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard . . . . . 33

Figure 5.1: Example Rectangle . . . . . 40
Figure 5.2: Example Triangle . . . . . 40
Figure 5.3: Example Ellipse . . . . . 40
Figure 5.4: Samples from the Shape dataset . . . . . 40
Figure 5.5: Example Square . . . . . 44
Figure 5.6: Example Equilateral Triangle . . . . . 44
Figure 5.7: Example Circle . . . . . 44
Figure 5.8: Samples from Easy Shapes Only . . . . . 44
Figure 5.9: The Architecture of the Convolutional Neural Network . . . . . 46
Figure 5.10: Test Error Rate Trace . . . . . 47
Figure 5.11: Training Error Rate Trace . . . . . 47

Figure 5.12: Error Rate Figures for MNIST on Neural Networks . . . . . 47
Figure 5.13: Test Error Rate Trace Zoomed In . . . . . 48
Figure 5.14: Training Error Rate Trace Zoomed In . . . . . 48
Figure 5.15: Zoomed In Error Rate Figures for MNIST on Neural Networks . . . . . 48
Figure 5.16: Test Error Rate Trace . . . . . 50
Figure 5.17: Training Error Rate Trace . . . . . 50
Figure 5.18: Error Rate Figures for Shape Dataset with Convolutional Neural Networks 50

LIST OF TABLES


Table 3.1: The final error rate averaged over 20 runs using SGD and synthetic data with the margin as the metric of hardness. . . . . 16
Table 3.2: The final objective function value averaged over 20 runs using SGD and synthetic data with the margin as the metric of hardness. . . . . 18
Table 3.3: The final error rate averaged over 20 runs using EGD and noisy synthetic data with the L-1 norm of the irrelevant features as the metric of hardness. . . . . 19
Table 3.4: The final objective function value averaged over 20 runs using EGD and noisy synthetic data with the L-1 norm of the irrelevant features as the metric of hardness. . . . . 19

Table 4.1: The final error rate averaged over 20 runs using SGD and MNIST 3 v 5 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness. 28
Table 4.2: The final objective function value averaged over 20 runs using SGD and MNIST 3 v 5 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness. . . . . 28
Table 4.3: The final error rate averaged over 20 runs using SGD and MNIST 3 v 5 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness. 30
Table 4.4: The final objective function value averaged over 20 runs using SGD and MNIST 3 v 5 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness. . . . . 30
Table 4.5: The final error rate averaged over 20 runs using SGD and MNIST 1 v 8 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness. 32
Table 4.6: The final objective function value averaged over 20 runs using SGD and MNIST 1 v 8 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness. . . . . 32
Table 4.7: The final error rate averaged over 20 runs using SGD and MNIST 1 v 8 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness. 34
Table 4.8: The final objective function value averaged over 20 runs using SGD and MNIST 1 v 8 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness. . . . . 34

Table 5.1: The final test error rate averaged over 10 runs using a neural network and MNIST as the dataset. . . . . 45
Table 5.2: The final test error rate averaged over 10 runs using a convolutional neural network and generated shapes as the dataset. . . . . 49

ACKNOWLEDGEMENTS

I am so appreciative to have had the opportunity to embark on this journey. I would like to thank Kamalika Chaudhuri for constantly providing support and guidance throughout the course of the project.

I would also like to thank Shuang Song for working with me and being a mentor to me throughout it all.

ABSTRACT OF THE THESIS

**Investigation on the Effects of Curriculum Learning through a Simulation Study**

by

Edward Wong

Master of Science in Computer Science

University of California San Diego, 2018

Professor Kamalika Chaudhuri, Chair

Machine learning has consistently proved to be useful in many applications. An integral facet allowing for machine learning is the training step. The training step of the models may take up excessive time and hardware resources. This is especially relevant when considering the fact that the performance of the models depend heavily on the aptitude of the training step. Depending on the scale of the application, any increase in performance during the training step may pay large dividends regarding the efficiency and success of the model as a whole. A heuristic commonly implemented as part of the training steps within models is the ordering of training samples with some predefined knowledge, more popularly known as curriculum learning. The common assumption resides in the belief that applying curriculum learning will generally improve

results. In this study, experiments were performed through the utilization of curriculum learning, while isolating certain variables to analyze the effect of curriculum learning as a whole. Within the experiments, different metrics of hardness will be explored. The training process will train on samples which will be ordered based on the different metrics of hardness. The orderings typically covered are Easy-to-Hard, Hard-to-Easy, and Random. Easy-to-Hard matches the analogy of how humans and animals typically learn. In Easy-to-Hard, the learner is presented with the easiest samples followed by progressively harder samples. In Hard-to-Easy, the learner is presented with the hardest samples followed by progressively easier samples. In Random, the learner is presented with samples that maintain no inherent ordering. Results obtained during this study reveal that curriculum learning does indeed have an effect on performance, but the effects can both be positive or negative, depending on the the metric of hardness. Such results garnered evidence that, when selecting a curriculum strategy for a setting, one should use care as the effectiveness of the training can be either diminished or enhanced based on the metric of hardness chosen.

# Chapter 1

# Introduction

## 1.1 Background

The main characteristic of supervised learning is that there is a requirement that the dataset is labeled. Machine learning typically starts with a pre-defined model. The goal is for the model to be able to produce the correct label or output from the given data or input. To do so, the model is trained against a labeled dataset. By training, the models parameters are optimized in order to correctly classify samples from the same underlying distribution in which the training set was drawn from.

Curriculum learning is based in the belief that learning process may become faster or the performance of the learner is improved when the learning process is structured. This is not a new or unique belief, as structured learning is universally used amongst humans and animals, as seen in the public school systems or within animal training programs. In Mathematics, individuals typically learn counting before learning addition and subtraction. After addition and subtraction, most students will then learn multiplication and division. This trend reveals how structured learning allows for concepts to build on top of one another, just as when humans learn mathematics. In the context of humans and animals, the process of learning in small incremental

steps is called shaping [9].

In this work, the idea of structured learning shall be used but within a context of machine learning. Experimentation will be conducted exclusively through supervised learning. Structure shall be applied to the learning or the training process of certain machine learning models under certain settings to determine whether or not it does provide additional utility. The goal is to develop an understanding of when curriculum learning works for the purpose of supervised learning.

## 1.2 Problem Statement

Curriculum-based learning used within machine-learning applications may increase the efficiency of the learning process as a whole when structured in specific ways that enable machine-learning algorithms to converge faster and at lower error rates. The objective of this research is to investigate when and where curriculum learning works through experimentation with examples by which isolated variables are obtainable; such examples include the manner in which the hardness of the data is quantified as well as the types of data being assessed.

Curriculum learning in real-life comes from static knowledge about what concepts build on each other or what concepts are easier. For example, counting may be easier to learn than addition. In addition, one may need to learn how to count before one can learn how to add. For these reasons, humans typically learn counting before then learning addition. However, this is somewhat of an abstract notion of how to define hardness and how to structure a curriculum. To be able to apply curriculum learning in a supervised learning setting, the quantification of the hardness of a sample has to be well-defined. In this paper, different measurements of hardness will be explored. The measurements of hardness can range from the margin from the decision boundary for a sample, to the amount of artificial noise added to a sample, or in a shape classification experiment, shapes such as circles, squares, and equilateral triangles are considered

easier than the more general class of ellipses, rectangles, and triangles.

The types of data that may undergo experimentation are synthetic and real-world data. Synthetic, or generated data, is easier to experiment with, as certain properties may be guaranteed, and therefore experimental results are clearer and trends may become more apparent. However, in order to be relevant within a real-world context, curriculum learning strategies must be analyzed with respect to not only synthetic data, but also real-world data in order to correctly assess the effects of each through comparison.

The machine learning algorithms used to construct models from data may be impacted by different curriculum learning approaches in numerous ways. Within this study, experimentation will occur on models that begin with a simple linear classifier, ranging all the way to a deep convolutional neural network.

Experiments regarding synthetic data with linear classifiers will be the topic of Chapter 3. The metrics of hardness explored in Chapter 3 will be the margin from the decision boundary and the amount of artificial noise added to each sample. In Chapter 4, the experiments will be dealing with real data. For measuring hardness in this case, a pretrained model will be used to estimate the margin. The topic of Chapter 4 will be the experiments with neural networks. A difference between 1 and the probability generated by the softmax layer for the true class of a sample will be used as a substitute for margin. In addition, a different notion of hardness will be used for the shape recognition problem. Out of the the general class of shapes for ellipses, rectangles, and triangles, only circles, squares, and equilateral triangles will be considered easy samples.

# Chapter 2

# Related Work

Prior work on this topic broadly falls into three categories – work on curriculum learning and other related concepts such as self-paced learning and active-learning.

## 2.1 Related Work on Curriculum Learning

Previous research conducted by Bengio, Louradour, Collobert, and Weston [1] introduced curriculum learning for machine learning settings. Their research [1] suggests that applying curriculum learning may improve the speed of convergence during the training process. In addition, curriculum learning could possibly improve the quality of the local minima which the model converges towards. The aforementioned research [1] generalizes on the work done in previous research, where the goal was to learn a simple grammar in a recurrent neural network. The training process focused on learning easier aspects of the grammar, meanwhile gradually expanding its resources and learning the more intricate details of the grammar [5]. Curriculum learning has since been applied successfully to achieve state of the art results in many machine

learning problems, such as facial recognition [11]. The work here attempts to replicate the shape experiments done in [1]. In addition, the work here will explore different settings in which to apply curriculum learning.

## 2.2 Related work on Alternative Forms of Learning

In supervised learning, the training set utilized is not necessarily unstructured. Rather, consideration may be taken when structuring the data for training purposes. These alternative forms of learning adhere the same to the idea that there exists samples that are more beneficial than others for training a model at specific points in the training process.

### 2.2.1 Self-Paced Learning

One alternative form of learning would be self-paced learning. Both self-paced and supervised learning involve the idea that structured learning may lead to improved learner performance or improved learning speeds. In curriculum learning, the structure comes from static prior knowledge, or well-defined metrics of hardness. In self-paced learning, the structure is dynamically defined through the specific needs of the learner as they develop [6]. For example, instead of a teacher fixing a curriculum for a student, the analogy would be the curriculum being dynamically adjusted according to the students abilities. Self-paced learning has also been applied successfully in machine learning settings, such as in latent variable models. A sample may be considered easy if the model is confident in determining the value of the hidden variable, or when the model is confident in in predicting the true output of said sample. Applying a self-paced

learning algorithm outperforms state of the art methods for learning latent structural SVM model in four applications object localization, noun phrase coreference, motif finding and handwritten digit recognition [10]. The work in this paper differentiates from self-paced learning in that a structuring of learning will come from some predefined notion, and not dynamically with respect to the current learner.

### 2.2.2 Active Learning

Another alternative form of learning is active learning. In active learning within a machine learning context, the learner has a dataset in which each sample's label is initially unknown, but the label for a specific sample may be obtained at some cost [4, 12]. The goal in active-learning is to learn a classifier without inducing too much cost. There are many settings where samples are costly to label, such as in image classification problems due to their necessity for manual human effort for labeling. Active learning allows for a cost efficient process for classifying images because it provides a framework by which cost is considered when seeking to learn a classifier [7]. The work in this paper differentiates from active learning in that there is no cost of labeling, as the labels are known in supervised learning. In active-learning, the goal is to select good samples to obtain labels for. Inversely, in curriculum learning and self-paced learning, the labels are already known and the goal is to discover what orderings of data allows for an expedited learning process by the learner.

# Chapter 3

# Synthetic Data With Linear Classifiers

## 3.1 Introduction

The properties of real data cannot be guaranteed, and often, it may be difficult dealing with the inherent complexities often found in real-world data. Synthetic data is cleaner in that it maintains the advantage of guaranteed properties as dedicated by the way in which the data was generated. Therefore, it is necessary to first examine the results of the application of certain curriculum learning strategies against synthetic data.

## 3.2 Problem Setting

In this section, a linear classification problem will be considered through the use of synthetic data. For any sample $x$ in $d$-dimensional space, the classification label of $x$ will be either positive or negative, as are binary labels, and the objective is to learn a vector $w$ such that

$w^T \cdot x >= 0$ for positive samples of $x$ and $w^T \cdot x < 0$ for negative samples. To achieve this, an initialization of the vector $w$ is made, followed by the use of certain machine learning algorithms in order to update $w$. The machine learning algorithms used in this chapter are Stochastic Gradient Descent and Exponentiated Gradient Descent.

### 3.2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an algorithm that may be used for the purpose of learning the classifier vector $w$ through the iterative updating of $w$ in order to lower the loss function [13]. Starting with some initial $w_0 \in W$ where $W$ is the set of all possible vectors in n-dimensional space, the following update is iteratively applied:

$$w_{t+1} = w_t - \eta_t \left(\nabla \ell(w_t, x_{i_t}, y_{i_t}) + \lambda w_t\right)$$

where $\eta_t = c/\sqrt{t}$

The hyperparameters $\lambda$ and $c$ remain optimized through the use of a hold-out or validation set.

### 3.2.2 Exponentiated Gradient Descent

Exponentiated Gradient Descent (EGD) is another algorithm used to learn the classifier vector $w$ [8]. EGD consists of two vectors, $w^+$ and $w^-$, which are the positive and negative parts of the classifier $w$ respectively. Therefore, $w = w^+ - w^-$, such that for a pre-defined scaling factor $D$, $\sum_{i=1}^{d}(w_i^+ + w_i^-) = D$ after each iteration. Both vectors are initialized as $w_1^+ = w_1^- =$

$(D/(2d),\ldots,D/(2d))$. For $t = 1,\ldots,T$, the algorithm calculates the gradient $\nabla f(w_t)$ at the current $w_t = w_t^+ - w_t^-$, and updates $w^+$ and $w^-$ as follows:

$$w_{t+1,i}^+ = D\frac{w_{t,i}^+ \exp(-\eta_t D[\nabla f(w_t)]_i)}{Z_t}$$

$$w_{t+1,i}^- = D\frac{w_{t,i}^- \exp(\eta_t D[\nabla f(w_t)]_i)}{Z_t}$$

where

$$Z_t = \sum_{i=1}^{d} w_{t,i}^+ \exp(-\eta_t D[\nabla f(w_t)]_i) + w_{t,i}^- \exp(\eta_t D[\nabla f(w_t)]_i)$$

is used for normalization and $\eta_t$ is the learning rate. We choose $\eta_t = c/\sqrt{t}$ for some constant $c$. As with the Stochastic Gradient Descent method, all hyperparameters are optimized by using a hold-out or validation set.

### 3.2.3 Loss Function - Logistic Loss

For both the SGD and EGD methods, the algorithms are defined in terms of a loss function. For both the SGD and EGD methods, implementation of the logistic loss function is enacted, defined as:

$$\ell(w_t, x_{i_t}, y_{i_t}) = \frac{-yx}{1+\exp(yw^\top x)}$$

## 3.3 Methodology

### 3.3.1 Data

The synthetically generated data samples $x \in D$, where $D$ is the data set and $k$ is the number of dimensions, will have the following properties:

1. $\forall x \in D \ \sqrt{\sum_{i=1}^{k} x_i^2} = 1$

    - This means that each sample lies on the unit hypersphere in the $k$-th dimension, which is to say that each sample has a l-2 norm of 1.

2. Let $f$ be the probability density function that represents the relative frequency of generating a particular sample for dataset $D$. Within the context of this experiment, $f(x^i) = f(x^k)$ where $x^i$ and $x^k$ have a l-2 norm of 1.

    - This is another way of stating that samples are generated uniformly on the unit hypersphere, therefore identifying that at any point on the hypersphere, it is as likely to appear within the dataset $D$ as any other.

## 3.4 Curriculum Metrics

All the samples within this experiment's dataset $D$ lie on the unit hypersphere. To formulate this into a classification problem, it is necessary to have a mechanism by which to apply a classification label per individial sample. If $k$ is the dimensionality of the samples $x \in D$, then a start may be had through the generation of a random vector $w^*$, which is also $k$-dimensional. $w^*$

is used to label each sample $x \in D$ in the following way.

$\forall x \in D$

1. If $x^T \cdot w^* >= 0$ then we label this sample $x$ as 1, or positive

2. If $x^T \cdot w^* < 0$ then we label this sample $x$ as 0, or negative

The vector $w^*$ essentially defines a hyperplane through the origin, which splits the dataset into 2 sets, a positive set and a negative set. The classification problem becomes that of determining which of the sets a given sample $x \in D$ lies in.

**Hardness by Margin**

The hyperplane defined by $w^*$ is the decision boundary for this classification problem. Intuitively, samples further from the decision boundary habitually represent easily classified samples, whereas samples that are closer to the decision boundary tend to remain more difficult to classify. The trend regarding the difficulty in classifying samples according to their relation to the decision boundary shall be called 'Hardness by Margin', and it will be quantified by the margin between $w^*$ and a sample $x$. Given a sample $x^e$ and a sample $x^h$, sample $x^e$ is considered easier than sample $x^h$ if and only if:

$$|w^{*T} \cdot x^e| > |w^{*T} \cdot x^h|$$

**Hardness by Irrelevant Features**

After defining $w*$ and concurrently labeling every sample $x \in D$, the addition of irrelevant features is applied to both $w^*$ and every sample $x \in D$. This allows for the testing of the effect of curriculum learning when the measurement of hardness is quantified by the noisiness of data samples. To do this, $n$ irrelevant dimensions are added, bringing the total dimensionality of the dataset to $n+k$, continued by the application of the following operations to $w^*$, as well as to each sample $x \in D$:

1. Append a vector of $n$ 0's to $w*$ to produce a $w^*$ that includes dimensions of noise.
    - 0's are appended here as each of the additional $n$ dimensions are irrelevant.
2. For all $x \in D$, generation of an $n$-dimensional vector, by which each of the $n$ entries are uniformly distributed from [1,1], is carried out. Then, a scaling factor is applied, also drawn from a uniform distribution of [.5,1.5].
    - The scaling factor is responsible for the mechanism in which irrelevant features may become emphasized in some samples more so than in others, allowing for a larger variance in noise for the samples within the dataset.

After these operations, a hardness function may be defined for each sample based on the noisiness maintained within the irrelevant features. This hardness function of $h(x)$ may be defined as the following:

$$h(x) = \sum_{i=k+1}^{n+k} |x_i|$$

The output of the hardness function is simply the sum of the absolute value of all the irrelevant features. Therefore, the example $x_e$ is considered to be easier than the example $x_h$ if $h(x_e) < h(x_h)$, which is to say that sample $x_e$ contains less noise in terms of the dimensions of irrelevant features present.

### 3.4.1 Baselines

To analyze the effect of curriculum learning on machine learning efficiency, it is necessary to define several different approaches to training, all of which may be comparable. For this experiment, five different approaches shall be taken. Given a training set of real-world data samples, the samples shall be presented in training in the following manner:

1. Easy-to-Hard – The samples are sorted by the metric of hardness, and the easiest samples will be used first for algorithmic training, followed by progressively harder samples.

2. Random – The samples are in an arbitrary order. This is a common practice used in machine learning applications, by which there are no inherent ordering of samples used for algorithmic training.

3. Hard-to-Easy – The samples are sorted by the metric of hardness, and the hardest samples will be used first for algorithmic training, followed by progressively easier samples. This is a reversal of the Easy-to-Hard approach.

4. Hard-Half - This is similar to the Hard-To-Easy approach, differentiated by the exception of using the hardest half of the dataset.

5. Easy-Half - This is similar to the Easy-To-Hard approach, except that we only use the easiest half of the dataset.

Note that Hard-Half and Easy-Half methodologies are only used in the Hardness by Margin experiments for the purpose of analyzing the effects of a curriculum-based approach being implemented on the learning efficiency of the algorithm.

## 3.5 Algorithms Selection

**SGD - Hardness by Margin**

For the experiments implementing the Hardness by Margin as the metric of difficulty for a given sample, implementation of the Stochastic Gradient Descent (SGD) algorithm is used towards the development of the classification model.

**EGD - Hardness by Irrelevant Features**

For experiments implementing Hardness by Irrelevant Features as the metric of difficulty, use of the Exponentiated Gradient Descent (EGD) algorithm is administered. Use of EGD rather than SGD for this experiment is due to the nature of EDG, as EGD is an algorithm that works particularly well when there are irrelevant features within the data.

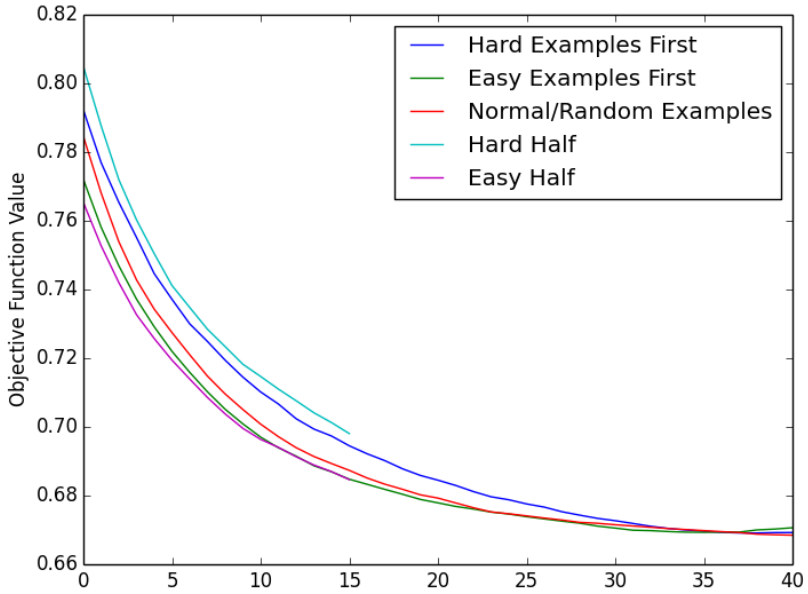


**Figure 3.1**: Objective Function Value Trace on Test Set

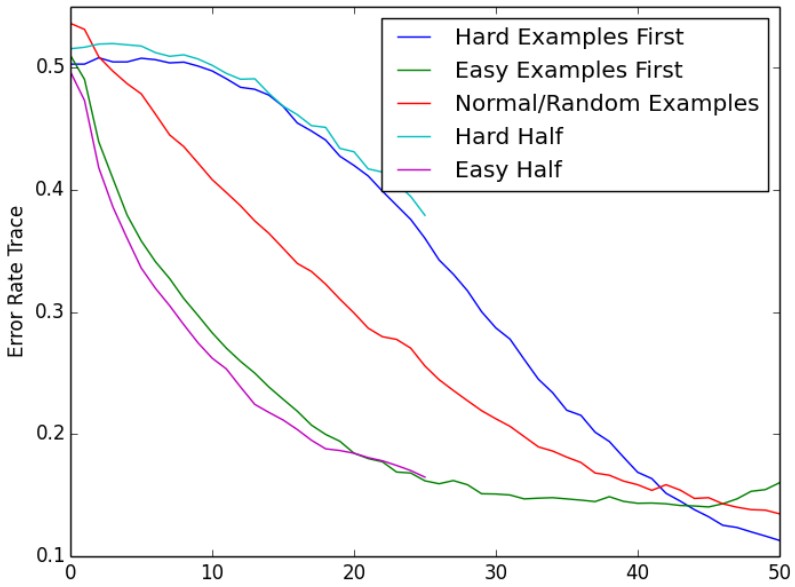


**Figure 3.2**: Test Error Rate Trace

**Figure 3.3**: Hardness measured by margin. $c = 1, \lambda = 0.1$. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard, Light Blue = Half-Half, Magenta = Easy-Half

## 3.6 Experimental Results

### 3.6.1 Hardness from Margin

Figure 3.3 depicts the results of the Hardness by Margin experiment, where the hardness of a sample has been measured based on the margin between the perfect classifier vector $w^*$ and the given sample x. With the generation of 1500 synthetic data samples, samples are split into a training set of 500 samples, a validation/holdout set of 500 samples, and a test set of 500 samples, with each sample featuring 25 dimensions that are made up of 25 relevant features, allowing for 0 irrelevant features. The hyperparameters were $c = 1$, $\lambda = 0.1$. The error rate trace and the objective function value trace consist of the function output averaged over 20 runs at constant intervals. The final error rate and final objective function value are based on the function outputs extrapolated from the final $w$, obtained only after training on every sample in the training set, averaged over the 20 runs.

**Table 3.1**: The final error rate averaged over 20 runs using SGD and synthetic data with the margin as the metric of hardness.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.173 | 0.026 |
| Random | 0.138 | 0.024 |
| Hard-to-Easy | 0.114 | 0.027 |
| Hard-Half | 0.345 | 0.053 |
| Easy-Half | 0.172 | 0.033 |

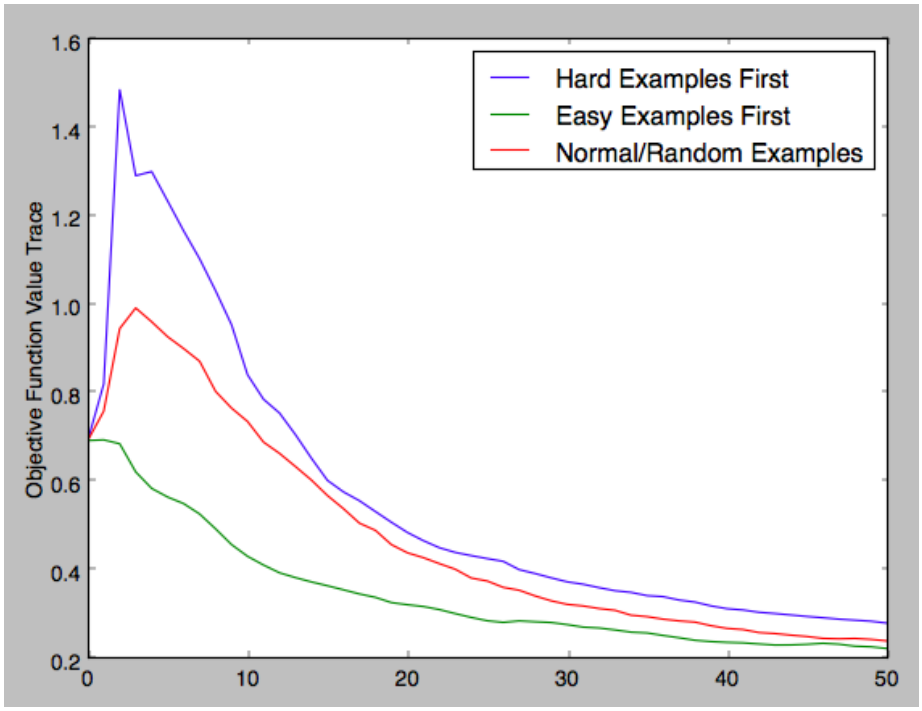


**Figure 3.4**: Objective Function Value Trace on Test Set

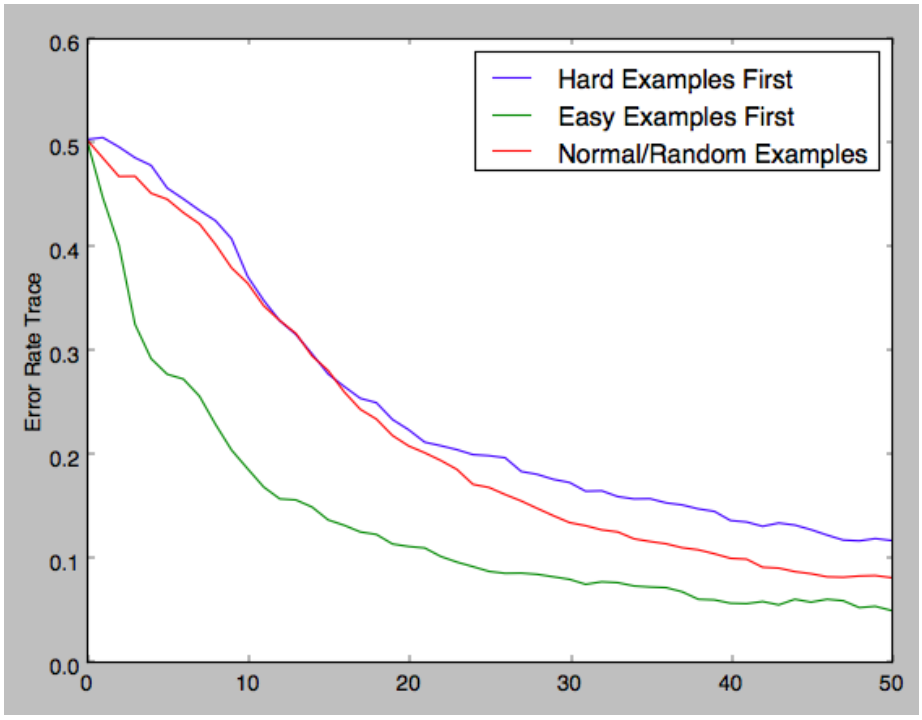


**Figure 3.5**: Test Error Rate Trace

**Figure 3.6**: Hardness measured by $L_1$ of irrelevant features. $D = 16$, $c = 4$. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard

**Table 3.2**: The final objective function value averaged over 20 runs using SGD and synthetic data with the margin as the metric of hardness.

| | Average Objective Function Value | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.672 | 0.0033 |
| Random | 0.671 | 0.0028 |
| Hard-to-Easy | 0.670 | 0.0030 |
| Hard-Half | 0.692 | 0.0084 |
| Easy-Half | 0.686 | 0.0072 |

### 3.6.2 Hardness from Irrelevant Features

Figure 3.6 shows the results of the Hardness from Irrelevant Features, where the hardness of a sample is measured based on the $L_1$ norm of irrelevant features. 750 synthetic data samples are then generated and split into a training set of 250 samples, a validation/holdout set of 250 samples, and a test set of 250 samples, with $d = 5$ relevant features and $k = 95$ irrelevant features. The hyperparameters are maintained as $c = 4$ and scaling factor $D = 16$. The error rate trace and objective function value trace consist of the function output averaged over 20 runs being sampled at constant intervals. The final error rate and final objective function value are based on the function outputs extrapolated from the final $w$, obtained only after training on every sample in the training set, averaged over the 20 runs.

**Table 3.3**: The final error rate averaged over 20 runs using EGD and noisy synthetic data with the L-1 norm of the irrelevant features as the metric of hardness.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.0556 | 0.022 |
| Random | 0.0818 | 0.036 |
| Hard-to-Easy | 0.114 | 0.046 |

**Table 3.4**: The final objective function value averaged over 20 runs using EGD and noisy synthetic data with the L-1 norm of the irrelevant features as the metric of hardness.

| | Average Objective Function Value | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.228 | 0.0032 |
| Random | 0.242 | 0.0046 |
| Hard-to-Easy | 0.281 | 0.0051 |

## 3.7 Discussion

The trends identified from the results of both experiments (Hardness by Margin and Hardness by Irrelevant Features) were present even when changing the dimensionality of the data and the number of samples in the data set.

### 3.7.1 Hardness from Margin

As seen from Figure 3.3, there are clear results when applying curriculum based training. In the beginning of algorithm training, the Easy-to-Hard curriculum resulted in a lower test error rate than the Random and Hard-to-Easy curriculums. From this, it is identified that a great marginal utility is clear when training the algorithm on easy examples initally.

At the conclusion of Easy-to-Hard training, where the algorithm is training on the hardest

samples in the data set, it was found that test error increased. This contradicts previous assumptions which maintain that large quantities of data training should generally produce better results. Findings showed that training on the hardest samples as defined by Hardness by Margin in-fact provided negative utility when the model is relatively converged.

Consequently, we see that although initial training using the Hard-to-Easy approach produced a higher test error rate than Random and Easy-to-Hard, the final error rate was lower, attributable to the fact training ended with the easiest samples.

These results exemplify the way in which the easiest samples in the data set provided much more utility for training, concurrently exemplifying how training on the hardest examples may inadvertently have an adverse effect on model performance when the model is relatively converged. Curriculum learning that focuses primarily on the easiest examples in a data set, as defined by Hardness by Margin, presents the potential of allowing for potential faster convergence and convergence to a more accurate model.

### 3.7.2 Hardness from Irrelevant Features

As seen from Figure 3.6, there are also clear effects that arise from applying curriculum based training within a different context. The learning rate of the algorithm decreases as a function of time, and experimental results reveal that the test errors produced by the final model were highest when the algorithm trained on the most difficult samples first, attributable to the fact that the learning rate was highest.

These results provide evidence which support training on the samples that are least noisy within the irrelevant dimensions, produces a more accurate classification model especially when

the learning rate is relatively high.

# Chapter 4

# Real Data With Linear Classifiers

## 4.1 Introduction

The goal of applying curriculum learning against machine learning applications lies in a desire to develop a system that does not simply improve outcomes derived from synthetic data, but rather improves outcomes applicable towards real-world scenarios involving real-world data. Curriculum strategies should be examined and analyzed in order to assess the effects observed when used against real-world data.

## 4.2 Problem Setting

To analyze the effects curriculum learning has on real-world data sets, a linear classification problem shall be formulated using real-world data. For any sample $x$ in $d$-dimensional space, the label of $x$ will either be positive or negative, as are binary labels. The goal is to learn a vector

$w$ such that $w^T \cdot x >= 0$ for positive examples $x$ and $w^T \cdot x < 0$ for negative samples. To set up such an experiment, initializing of the vector $w$ and concurrent application of machine learning algorithms are enacted in order to update its weights for the purpose of improving the accuracy of the classification model. In this section, the Stochastic Gradient Descent (SGD) algorithm is utilized.

### 4.2.1 Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an algorithm that may be used to learn the classifier vector $w$ through iterative updating $w$, which may lower the loss function. Starting with some initial $w_0 \in W$ where $W$ is the set of all possible vectors in n-dimensional space, we iteratively apply the following update:

$$w_{t+1} = w_t - \eta_t \left( \nabla \ell(w_t, x_{i_t}, y_{i_t}) + \lambda w_t \right)$$

where $\eta_t = c/\sqrt{t}$

The hyperparameters $\lambda$ and $c$ are optimized by using a hold-out or validation set.

### 4.2.2 Loss Function - Logistic Loss

Stochastic Gradient Descent (SGD) is defined in terms of a loss function. The logistic loss function will be implemented, defined as:

$$\ell(w_t, x_{i_t}, y_{i_t}) = \frac{-yx}{1 + \exp(yw^\top x)}$$

## 4.3 Methodology

### 4.3.1 Data

The MNIST (Modified National Institute of Standards and Technology) dataset shall be utilized, which is a large dataset of handwritten numeric digits commonly used to formulate pattern recognition and machine learning problems. To construct a binary linear classification problem, it is necessary to reduce the number of classes of samples from the MNIST dataset down to two classes. Therefore, filtering samples from the MNIST dataset through only 2 digits allows for one of the digits to be positive and the other to be negative.

#### Preprocessing of Data

PCA (Principle Component Analysis) will be used to reduce the dimensionality of the samples [15]. The primary function of the PCA serves to create a manageable classification problem, allowing for a quicker running and iteration of the experiment through lower dimensionality within the problem.

## 4.4 Curriculum Metrics

In the previous experiments on synthetic data, samples were generated uniformly on the hypersphere. From this, it is allowable to randomly select a vector $w^*$ that defined a hyperplane dividing the hypersphere into two distinct regions, by which samples are concurrently labeled according to which side of the hyperplane they appeared present within. In doing so, it was made

possible to define an objective margin between a given sample and the decision boundary, which was then used to quantify the hardness of that sample.

With real-world data, a vector $w^*$ is no longer able to be defined, resulting in a splitting of the data perfectly into the two regions necessary prior to quantify hardness for the samples. However, it is possible to learn a sufficient $w^*$ through running a normal training algorithm for a linear classifier on the data. The resulting $w^*$ may be used to define the hyperplane. By measuring the margin of each sample from this hyperplane, the hardness of a given sample is quantifiable.

**Hardness by Margin**

After learning a vector $w$ that defines the decision boundary in order to classify samples, Hardness by Margin may be defined in the same manner as the synthetic data. Intuitively, a sample that lies closer to the hyperplane, may be defined as being harder to classify, and a sample lying further away from the hyperplane as being easier. The measured margin between $w^*$ and a sample $x$ is then used to quantify the hardness of that sample. Given a sample $x^e$ and a sample $x^h$, one may say that the sample $x^e$ is easier than sample $x^h$ if and only if

$$|w^{*T} \cdot x^e| > |w^{*T} \cdot x^h|$$

### 4.4.1 Baselines

To analyze the effect of curriculum learning on machine learning efficiency, it is necessary to define several different approaches to training, all of which may be comparable. For this experiment, three different approaches shall be taken. Given a training set of real-world data

samples, the samples shall be presented in training in the following manner:

1. Easy-to-Hard – The samples are sorted by the metric of hardness, with the easiest samples being trained first followed by the training of progressively harder samples.

2. Random – The samples are in an arbitrary order. This is commonly used in practices regarding machine learning applications, where there is no inherent ordering of samples used when training the algorithm.

3. Hard-to-Easy – The samples are sorted by the metric of hardness, with the hardest samples trained first followed by progressively easier samples. This is the reverse of the Easy-to-Hard approach.

## 4.5 Algorithm Selection

For this experiment, we will use the Stochastic Gradient Descent (SGD) algorithm to learn the vector $w$.

## 4.6 Experimental Results

**MNIST 3 v 5 PCA Reduced to 50 dimensions**

Figure 4.3 shows the results of MNIST 3 v 5, where the algorithm trained on the data set only after reducing the dimensionality of the samples to 50. In order to formulate the appropriate linear classification problem, samples taken from MNIST were solely the digit 3 or the digit 5. By

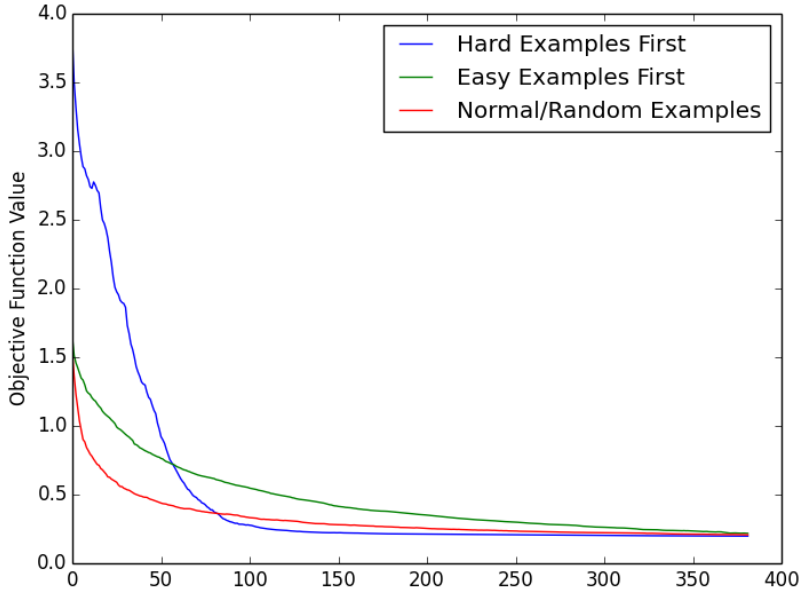


**Figure 4.1**: Objective Function Value Trace on Test Set

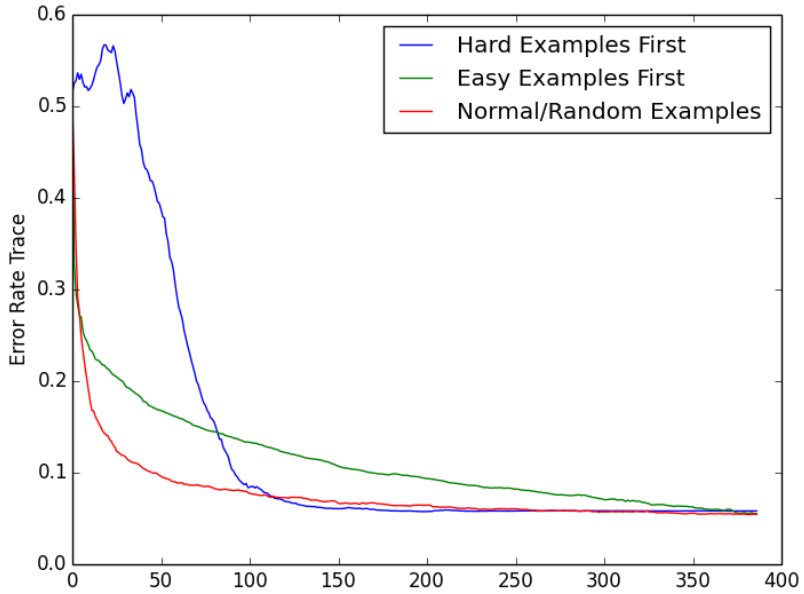


**Figure 4.2**: Test Error Rate Trace

**Figure 4.3**: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c = 1$. PCA reduced dimensionality to 50. Data is MNIST 3 v 5. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard

first running a training algorithm on the data set as normally performed, a vector $w$ was learned that yielded a 0.058 % test error. This was then used to measure and quantify the hardness by margin of each of the samples within the data set. The hyperparameters of the Stochastic Gradient Descent algorithm used in this experiment were $\lambda = 0.01$ and $c = 1$. The error rate trace and the objective function value trace were then averaged over 20 runs at constant intervals.

**Table 4.1**: The final error rate averaged over 20 runs using SGD and MNIST 3 v 5 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.0594 | 0.0076 |
| Random | 0.0575 | 0.0074 |
| Hard-to-Easy | 0.0592 | 0.0048 |

**Table 4.2**: The final objective function value averaged over 20 runs using SGD and MNIST 3 v 5 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Objective Function Value | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.234 | 0.0177 |
| Random | 0.217 | 0.0187 |
| Hard-to-Easy | 0.205 | 0.01062 |

**MNIST 3 v 5 PCA Reduced to 75 dimensions**

Figure 4.6 shows the results of MNIST 3 v 5, where the algorithm trained on the data set after reduction of the dimensionality of the samples to 75. In order to formulate the appropriate linear classification problem, samples taken from MNIST were solely the digit 3 or the digit 5. By

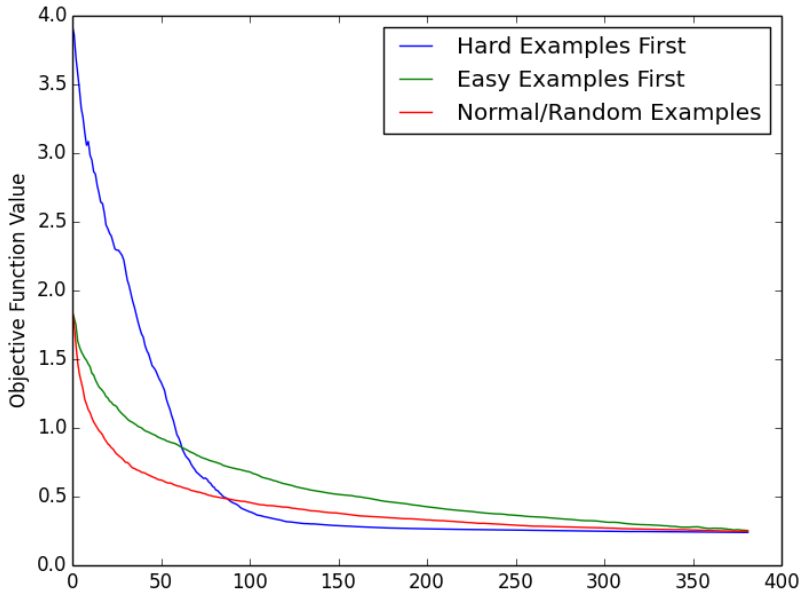


**Figure 4.4**: Objective Function Value Trace on Test Set

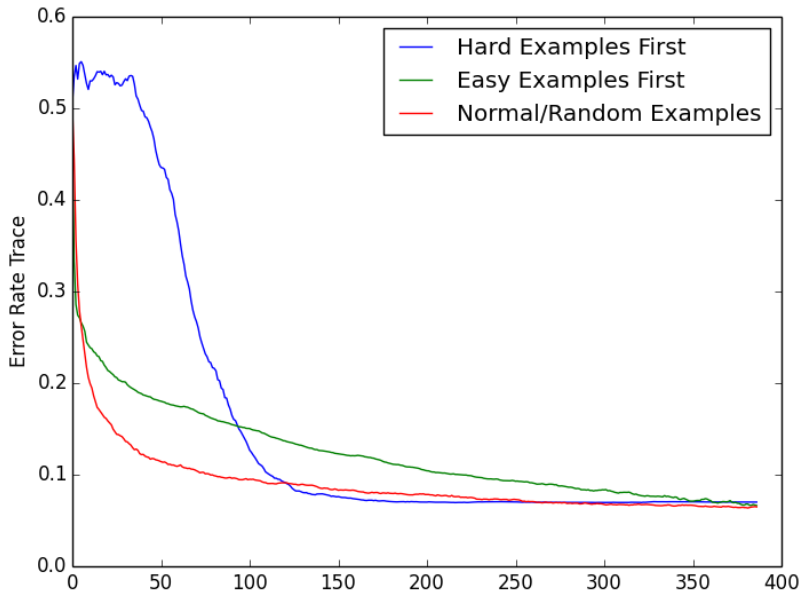


**Figure 4.5**: Test Error Rate Trace

**Figure 4.6**: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c = 1$. PCA reduced dimensionality to 75. Data is MNIST 3 v 5. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard

first running a training algorithm on the data set as normally performed, a vector $w$ was learned that yielded a 0.068 % test error. This was then used to measure and quantify the hardness by margin of each of the samples within the data set. The hyperparameters of the Stochastic Gradient Descent algorithm used in this experiment were $\lambda = 0.01$ and $c = 1$. The error rate trace and the objective function value trace were then averaged over 20 runs at constant intervals.

**Table 4.3**: The final error rate averaged over 20 runs using SGD and MNIST 3 v 5 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.0579 | 0.0073 |
| Random | 0.0546 | 0.0045 |
| Hard-to-Easy | 0.0589 | 0.0053 |

**Table 4.4**: The final objective function value averaged over 20 runs using SGD and MNIST 3 v 5 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Objective Function Value | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.2428 | 0.0180 |
| Random | 0.2313 | 0.0097 |
| Hard-to-Easy | 0.2171 | 0.0103 |

### MNIST 1 v 8 PCA Reduced to 50 dimensions

Figure 4.9 shows the results of MNIST 1 v 8, where the algorithm trained on the data set after reduction of the dimensionality of the samples to 50. In order to formulate the appropriate linear classification problem, samples taken from MNIST were solely the digit 1 or the digit 8.

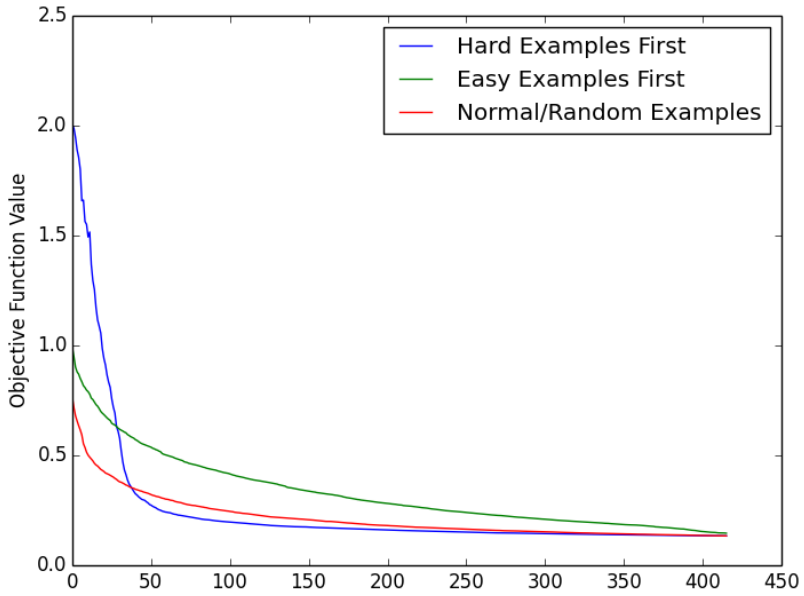


**Figure 4.7**: Objective Function Value Trace on Test Set

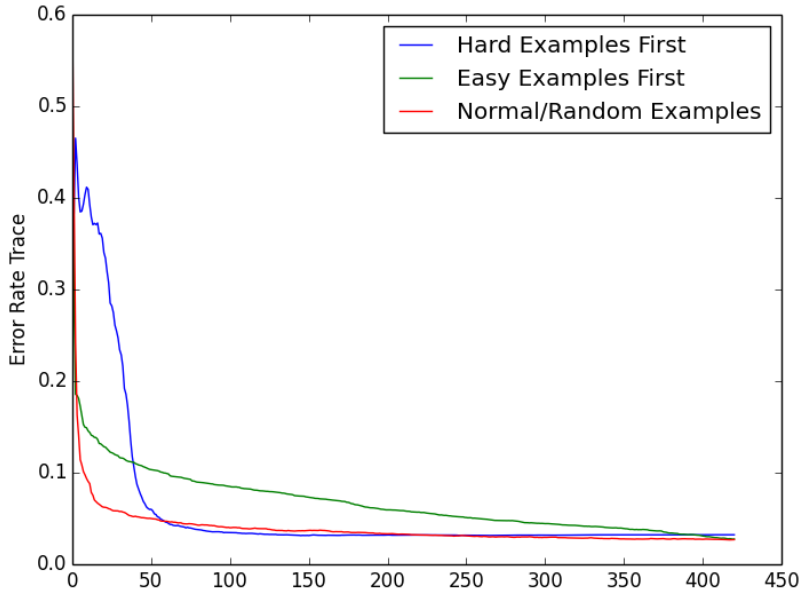


**Figure 4.8**: Test Error Rate Trace

**Figure 4.9**: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c = 1$. PCA reduced dimensionality to 50. Data is MNIST 1 v 8. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard

By first running the appropriate training algorithm on the data set as normally performed, a vector $w$ was learned that yielded a 0.028 % test error. This was then used to measure and quantify the hardness by margin of each of the samples within the data set. The hyperparameters of the Stochastic Gradient Descent algorithm used in this experiment were $\lambda = 0.01$ and $c = 1$. The error rate trace and the objective function value trace were then averaged over 20 runs at constant intervals.

**Table 4.5**: The final error rate averaged over 20 runs using SGD and MNIST 1 v 8 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.0292 | 0.0038 |
| Random | 0.0305 | 0.0043 |
| Hard-to-Easy | 0.0319 | 0.0047 |

**Table 4.6**: The final objective function value averaged over 20 runs using SGD and MNIST 1 v 8 (50 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Objective Function Value | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.1582 | 0.0190 |
| Random | 0.1368 | 0.0129 |
| Hard-to-Easy | 0.1368 | 0.0096 |

### MNIST 1 v 8 PCA Reduced to 75 dimensions

Figure 4.12 shows the results of MNIST 1 v 8, where the algorithm trained on the data set after reduction of the dimensionality of the samples to 75. In order to formulate the appropriate

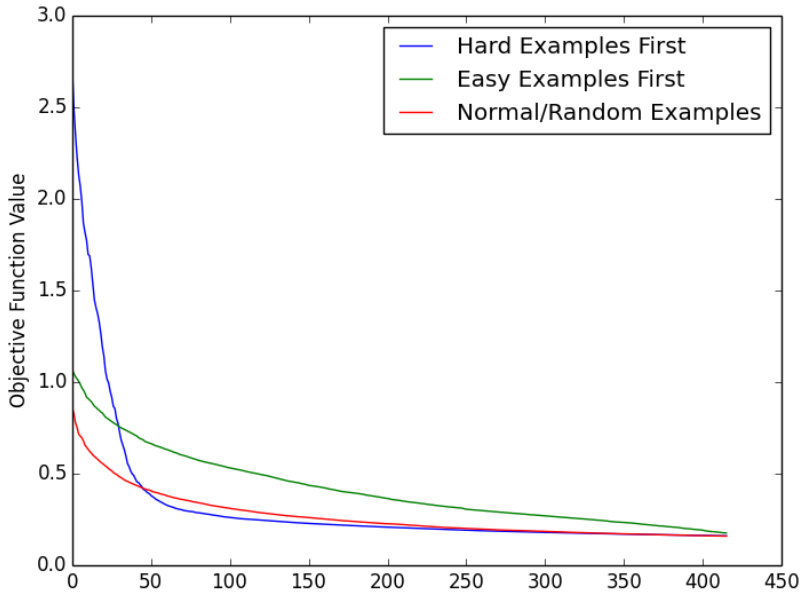


**Figure 4.10**: Objective Function Value Trace on Test Set

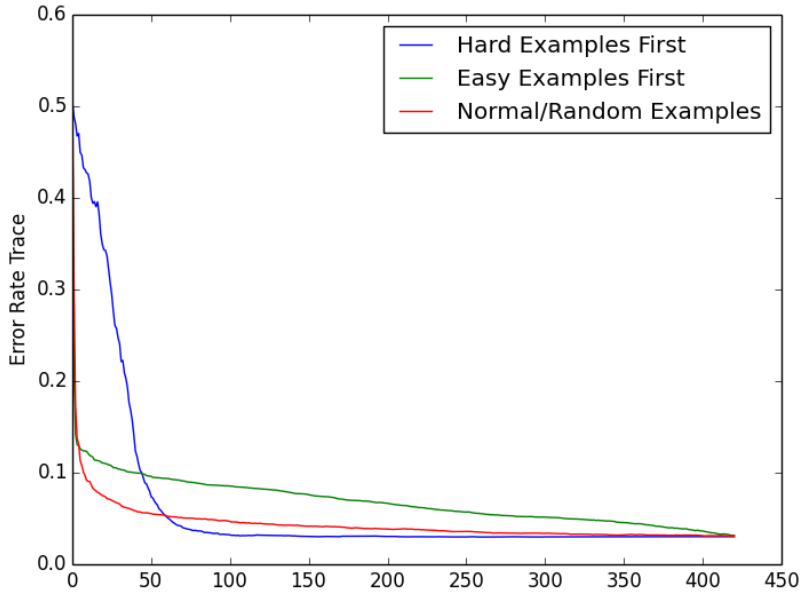


**Figure 4.11**: Test Error Rate Trace

**Figure 4.12**: Hardness measured by margin of a pretrained $w$. $\lambda = 0.01$, $c = 1$. PCA reduced dimensionality to 75. Data is MNIST 1 v 8. Blue = Hard-to-Easy, Red = Random, Green = Easy-to-Hard

linear classification problem, samples taken from MNIST were solely the digit 1 or the digit 8. By first running a training algorithm on the data set as normally performed, a vector $w$ was learned that yielded a 0.029 % test error. This was then used to measure and quantify the hardness by margin of each of the samples within the data set. The hyperparameters of the Stochastic Gradient Descent algorithm used in this experiment were $\lambda = 0.01$ and $c = 1$. The error rate trace and the objective function value trace were then averaged over 20 runs at constant intervals.

**Table 4.7**: The final error rate averaged over 20 runs using SGD and MNIST 1 v 8 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.0321 | 0.0036 |
| Random | 0.0326 | 0.0043 |
| Hard-to-Easy | 0.0313 | 0.0029 |

**Table 4.8**: The final objective function value averaged over 20 runs using SGD and MNIST 1 v 8 (75 dimensions after PCA) as the dataset with the margin as the metric of hardness.

| | Average Objective Function Value | Standard Deviation |
|---|---|---|
| Easy-to-Hard | 0.1770 | 0.0162 |
| Random | 0.1669 | 0.0172 |
| Hard-to-Easy | 0.1518 | 0.0091 |

## 4.7 Discussion

The trends identified within real-world data experimentation yielded different results than those found in synthetic data experimentation. In order to accurately compare the impact

varying levels of classification difficulty had on the effects of curriculum learning, both a hard classification problem, which was the problem of classifying the digit 3 versus 5, and an easy classification problem, which was classifying the digit 1 versus 8, were experimented upon. In order to see if dimensionality may be correlated with observable trends in the experimental results, experimentation occured when varying the dimensionality reduction of the data using PCA.

The main trends observed identified that Easy-to-Hard maintained a slower convergence rate than Random and Hard-to-Easy. Random and Easy-to-Hard had a large marginal decrease in error rate initially, whereas Hard-to-Easy would start off with a relatively much higher error rate, but eventually decreased in error rate much faster. Although each converged at a relatively similar error rates, it is clear that the rate of convergence varies within different curriculum strategies. Hard-to-Easy tends to converge quickest despite having higher error rates early on in training.

# Chapter 5

# Neural Networks

## 5.1 Introduction

Curriculum learning strategies are often implemented when training neural networks, as it offers the opportunity for faster convergence rates and improved model accuracy. Due to the common usage of neural networks, it remains imperative that experimentation continues regarding neural networks uses in order to provide more comprehensive and necessary analysis. In the following experiments, curriculum strategies will be applied towards training neural networks in an attempt to develop understanding of the effects of curriculum based training on such neural networks.

## 5.2 Problem Setting

In this section, multi-class classification problems on both real and synthetic data shall be considered. The machine learning model being applied to the data is a neural network. Neural networks are defined by the hidden layers of neural units that lie between the input and output layers within the network, as well as by the number of neural units and activation functions of said hidden layers.

### 5.2.1 Loss Function - Cross Entropy Loss

Here, a Cross Entropy Loss function for multiple classes is implemented during experimentation.

$$E = -\Sigma_n \Sigma_{k=1}^{c} t_k^n \ln y_k^n$$

### 5.2.2 Activation Functions

In order to achieve non-linearity, different types of activation functions are applied to the hidden layers within the neural network [2]. Some of these activation functions include:

1. Sigmoid - $g(x) = \frac{1}{1+e^x}$
2. Tanh - $g(x) = tanh(x)$
3. ReLu - $g(x) = max(0,x)$

### 5.2.3 Back Propagation

During training of a neural network, the goal is learning the weights for each of the connections between the units in a neural network. Back propagation is a generalization of the gradient descent algorithm for neural networks which contains hidden layers [14]. The updated rule is as follows:

$$w_{ij} = w_{ij} - \eta * \delta_j * z_i$$

where $w_{ij}$ is the weight for the connection from node $i$ to node $j$, $z_i$ is the activation at node i, $\eta$ is the stepsize. If $j$ is an output unit, $\delta_j$ is defined as:

$$\delta_j = (t_j - y_j)$$

where $t_j$ is the target output and $y_j$ is the current output. However if $j$ is a hidden unit, $\delta_j$ is defined as:

$$\delta_j = g'(a_j) \sum_k \delta_k * w_{jk}$$

where $g'(a_j)$ is the gradient of the activation function.

### 5.2.4 Output Layer - Softmax Regression

A softmax layer will be used for the output layer of the neural network. The softmax layer will normalize the outputs, allowing for their interpretations as probabilities. The following may be applied against the softmax layer:

$$y_k = \frac{exp(a_k)}{\Sigma_{k'} exp(a_{k'})}$$

where $a_k$ is the sum of the activations into the output unit $k$. The classification decision of the neural network is defined as the output with the highest probability value, i.e., $argmax_k(y_k)$.

## 5.3 Methodology

### 5.3.1 Data

#### MNIST

As with previous experimentation using real-world data, the use of the MNIST dataset of handwritten digits are again used. Within this experiment, the MNIST naturally presents multi-class classification problem as there are 10 numeric digits.

#### Generated Shapes

Initial experimentation included an attempt to replicate the findings within a 2009 Curriculum Learning study [1]. Within the context of the 2009 study, the classification problem aims to distinguish between several different shapes. The shapes that the dataset consists of are ellipses, rectangles, and triangles. Random generation of these shapes were used to fill the dataset, and the sizes and orientations of each of the shapes were chosen randomly as well. Examples of the shapes can be seen in 5.4.

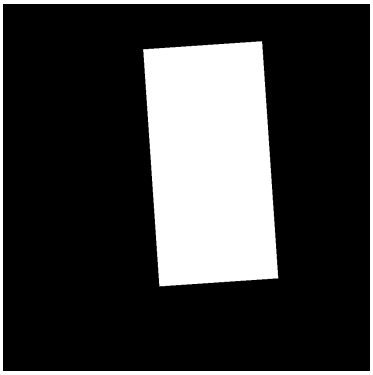

**Figure 5.1**: Example Rectangle

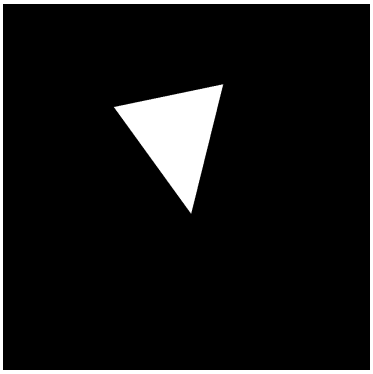

**Figure 5.2**: Example Triangle

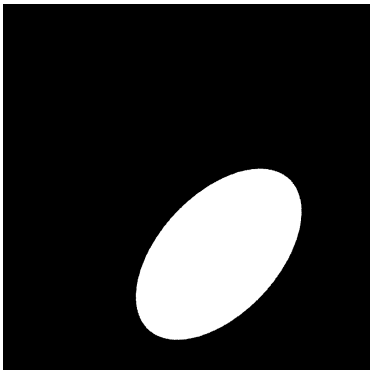

**Figure 5.3**: Example Ellipse

**Figure 5.4**: Samples from the Shape dataset

## 5.4 Curriculum Metrics

### 5.4.1 MNIST Dataset

As in previous experiments, it is necessary to quantify hardness for each of the examples. To do so, a neural network is trained to classify the digits of MNIST, and concurrently it uses the output of the trained neural network to quantify the hardness of a sample. Recall that the output layer of the neural network is a softmax layer. This allows for the interpretation of the output from the forward propagation of a sample in a neural networks as the probability a sample will be in each of the possible classes, e.g., $y_k = Pr(x \in k)$. After training a neural network on the data set, sample $x$ may be ran through it. Quantification of the hardness of sample $x$ where the true class is $c$ may then be obtained as follows:

$$1 - Pr(x \in c)$$

The higher the value, the harder the sample is considered. Intuitively, one would consider that, if the true class is of a sample $x$ is $c$, a higher value for $1 - Pr(x \in c)$ would mean the neural net struggled when classifying the sample $x$ as class $c$. This is attributable to $Pr(x \in c)$, which is the probability of the sample belonging to that class as relatively low.

### 5.4.2 Shape Dataset

In the experiment from the 2009 Curriculum Learning paper[1], which is undergoing the attempted replication, the multi-class classification problem involves distinguishing between

ellipses, rectangles, and triangles. The 2009 Curriculum Learning paper states that the easy examples are circles, squares, and equilateral triangles, and for the purposes of this study, hardness will be interpreted in the same way [1].

### 5.4.3 Baselines

#### MNIST Dataset

For the following experiments, three different ways to train the neural network based on the hardness of the samples will be defined. The training therefore proceeds in the following ways:

1. Easy-to-Hard – The samples are sorted by the metric of hardness, and the easiest samples will be used first for training, followed by progressively harder samples.

2. Random – The samples are in an arbitrary order. This is commonly used in practice of machine learning applications, where there is no inherent ordering of samples used when training the algorithm.

3. Hard-to-Easy – The samples are sorted by the metric of hardness, and the hardest samples will be used first for training, followed by progressively easier samples. This is the reverse of the Easy-to-Hard approach.

### 5.4.4 Shape Dataset

In the case of shape identification, the distinction between a sample being hard or easy is a binary property. If the shape is a circle, square, or an equilateral triangle, the sample is considered easy. If it is not any of the above, it is considered hard. Training of the neural network with the following curriculum strategies shall proceed as follows:

1. Easy Shapes Only - The training set consists of only circles, squares, and equilateral triangles. Examples of easy shapes may be seen in Figure 5.8.

2. Hard Shapes Only - The training set consists of ellipses which are not circles, rectangles which are not squares and triangles which are not equilateral.

3. Easy and Hard Shapes Only - The training set consists of a random mixture of easy and hard shapes.

## 5.5 Neural Network Model Selection

### 5.5.1 MNIST Dataset

To quantify the hardness of the samples within MNIST, using the output from a pre-trained neural network allows for the training of a new neural network against the data, providing a probability value for quantify the hardness of each sample. Use of a neural network with 1 hidden layer constructed of 64 hidden units with logistic activation functions. Once the hardness of each of the samples has been quantified, the data sets are then prepared according to each appropriate

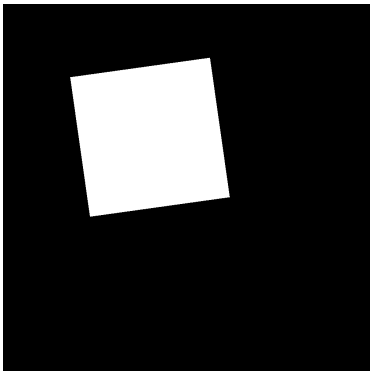

**Figure 5.5**: Example Square

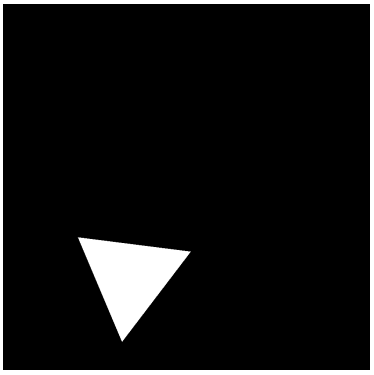

**Figure 5.6**: Example Equilateral Triangle

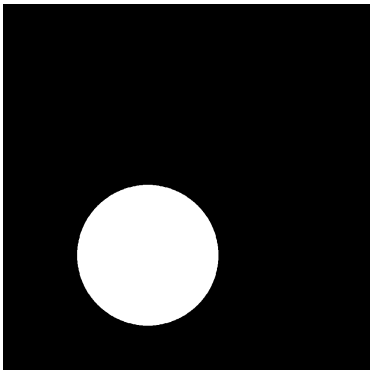

**Figure 5.7**: Example Circle

**Figure 5.8**: Samples from Easy Shapes Only

method as described in Baselines, and experimental results are produced through training with a neural network of the same architecture.

### 5.5.2 Shape Dataset

We use a convolutional neural network for this image classification problem [3]. The architecture is specified in Figure 5.9.

## 5.6 Experimental Results

### 5.6.1 MNIST Dataset

To achieve consistent, stable gradients, the use of batched gradient descent or back propagation with 10 examples per batch is implemented. A stepsize of 0.01 or $\eta = 0.01$ is also used.

**Table 5.1**: The final test error rate averaged over 10 runs using a neural network and MNIST as the dataset.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy to Hard | 0.1152 | 0.062 |
| Random | .1055 | 0.047 |
| Hard to Easy | 0.1031 | 0.052 |

Within the results, the error rate trace for both the training set and the test set is present, including a magnified version aiding in identifying the trends in Figure 5.18 and Figure 5.15. The traces for the error rates are averaged over 10 runs.

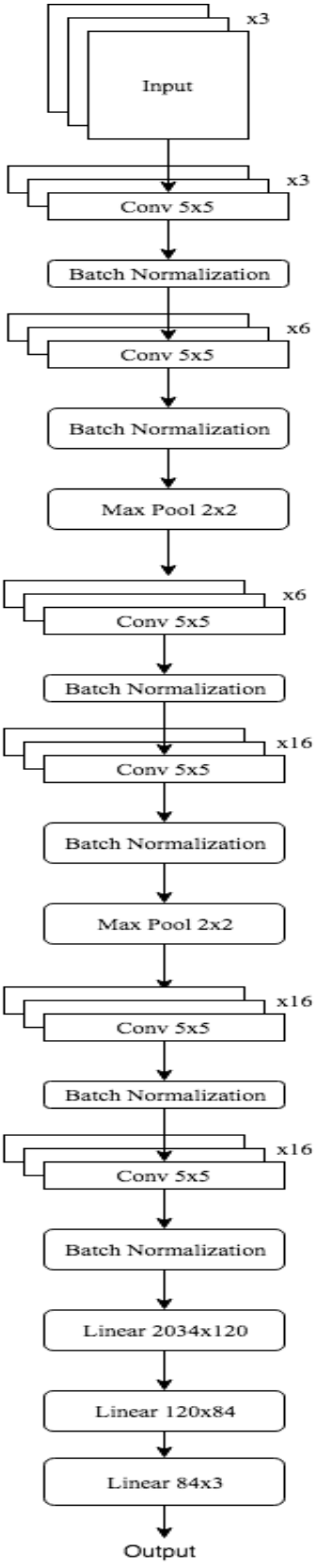


**Figure 5.9**: The Architecture of the Convolutional Neural Network

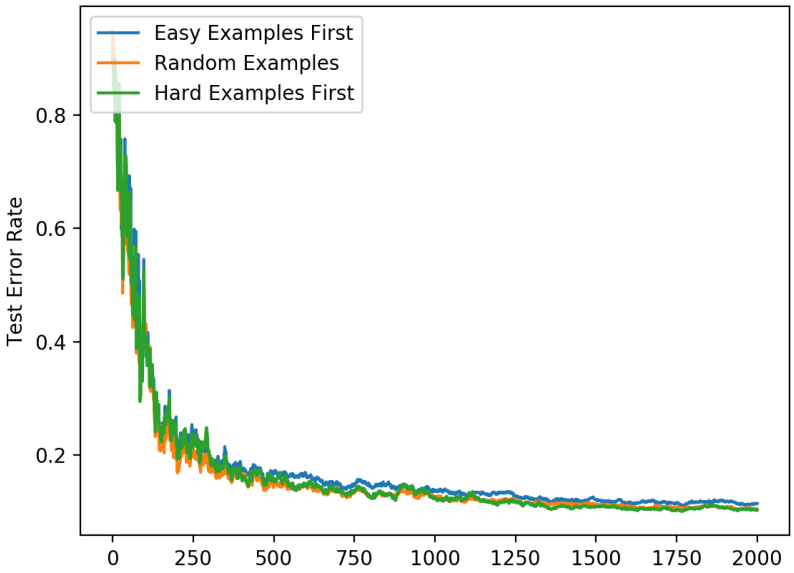


**Figure 5.10**: Test Error Rate Trace

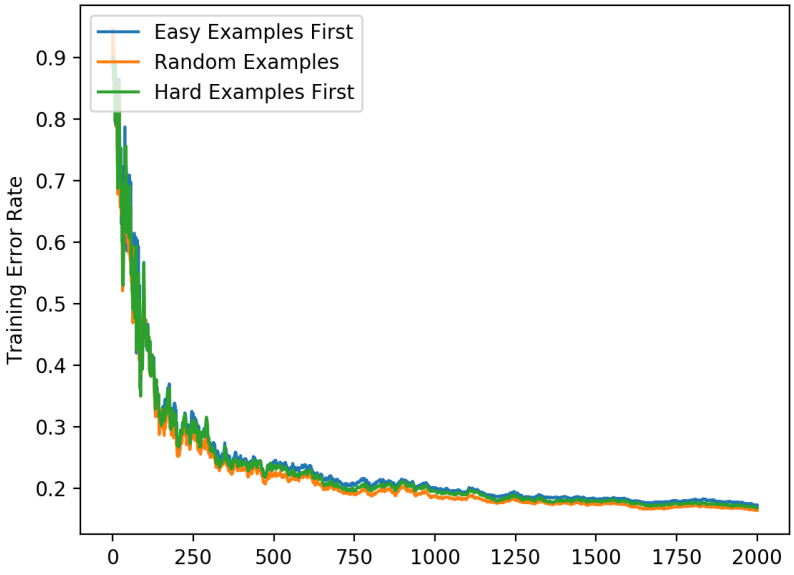


**Figure 5.11**: Training Error Rate Trace

**Figure 5.12**: Error Rate Figures for MNIST on Neural Networks

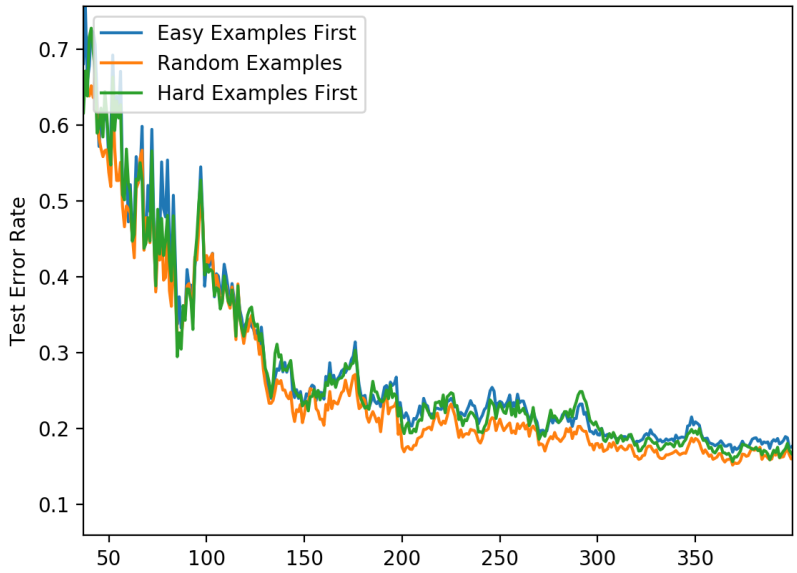


**Figure 5.13**: Test Error Rate Trace Zoomed In

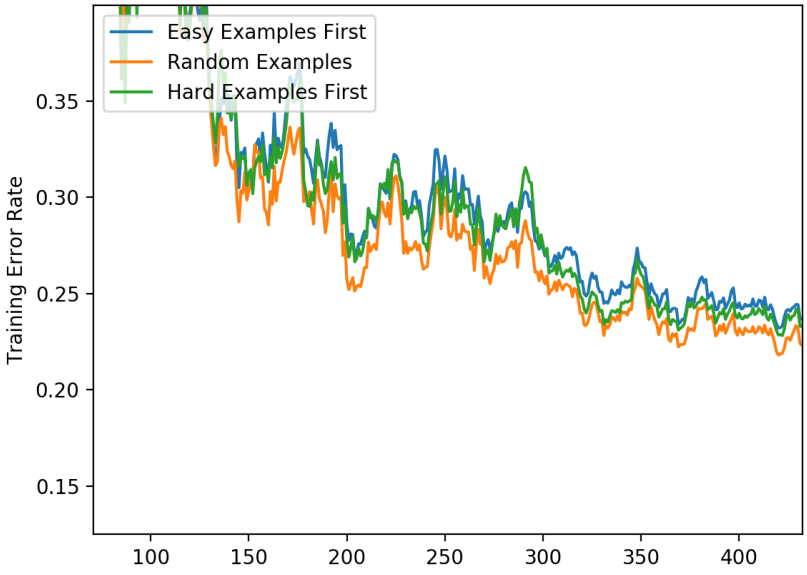


**Figure 5.14**: Training Error Rate Trace Zoomed In

**Figure 5.15**: Zoomed In Error Rate Figures for MNIST on Neural Networks

### 5.6.2 Shape Dataset

For the shape dataset, batch sizes of 1000 samples are used. The Adam optimizer was also used, which is an extension of the Stochastic Gradient Descent algorithm, in order to train the neural network weights while implementing a learning rate of 0.001. The following metrics and graphs are averaged over 10 runs:

**Table 5.2**: The final test error rate averaged over 10 runs using a convolutional neural network and generated shapes as the dataset.

| | Average Error Rate | Standard Deviation |
|---|---|---|
| Easy Shapes Only | 0.1112 | 0.0223 |
| Easy and Hard Shapes | 0.0955 | 0.0194 |
| Hard Shapes Only | 0.1141 | 0.0211 |

## 5.7 Discussion

Analysis of the experimental results yielded a number of trends. One of the trends discovered was that, artificially applying a curriculum strategy towards training the neural network actually produced slightly worse results than if the training was performed naively. In the case of the MNIST dataset, although all the strategies converge to a relatively similar final test error rate, the magnified images of the graphs show that applying no curriculum (i.e. random sample ordering) produced a marginally lower error rates.

In the Shapes Dataset case, it was found that filtering the training set in order to consist solely of easy shapes for the Easy Shapes Only curriculum strategy resulted in lower performance. The curriculum that produced the lowest test error rate was when the training set used a mixture

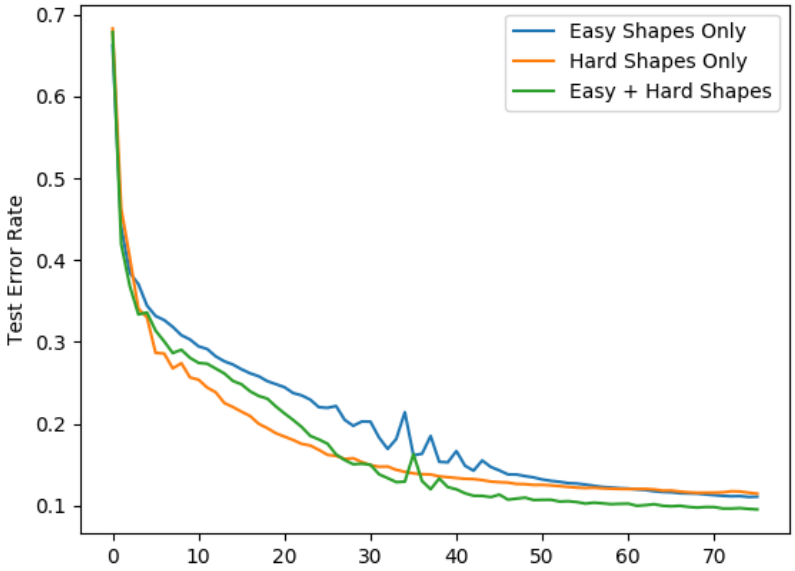


**Figure 5.16**: Test Error Rate Trace

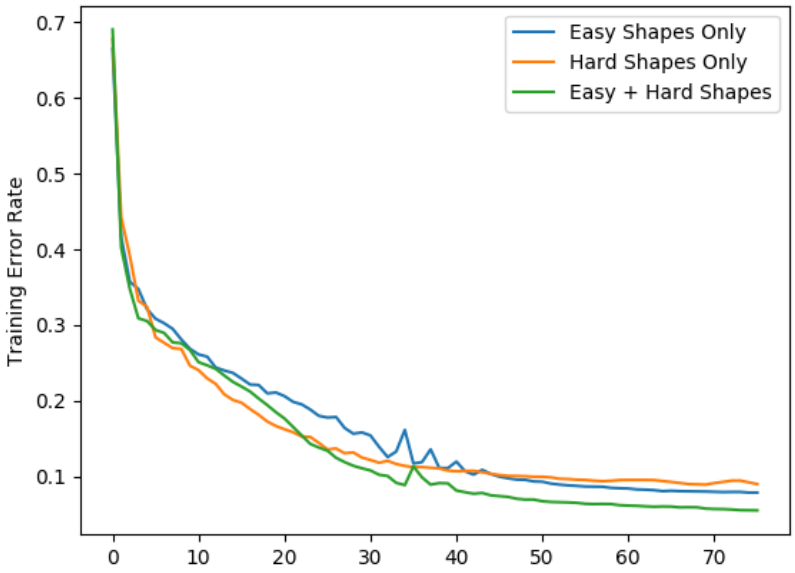


**Figure 5.17**: Training Error Rate Trace

**Figure 5.18**: Error Rate Figures for Shape Dataset with Convolutional Neural Networks

of both easy shapes and hard shapes. One possible rationale for this discovery comes from the fact that easy shapes and hard shapes better represents the distribution from which the dataset was generated from. Training only on easy shapes as defined in [1] may not necessarily help to generalize the ability of the convolutional neural network in classifying images between ellipses, rectangles, and triangles.

# Chapter 6

# Conclusion

Results from the experiments conducted for the purpose of this study clearly identify trends by which applying curriculum learning provides an effect on the performance amongst the final classifiers and convergence rates. In the experiments involving synthetic data and linear classifiers, it was found that there exists curriculum strategies which may improve final classifier performance and also potentially improving the speed of convergence. Another observation encountered revealed that, depending on the definition of hardness or difficulty of a sample, training may garner a decrease in performance when the hardest samples are used in training when the classifier was relatively converged.

In experiments involving real data and linear classifiers, it was found that although each of the methods of training presented relatively similar final test error rates. Hard-to-Easy or an anti-curriculum strategy tended to present worst performance initially before then converging quicker than the other methods.

In experiments based on neural networks, it was found that the application of curriculum

strategy may bring about adverse effects on both the final classifier performance.

When comparing these findings to other forms of testing, it was clear certain scenarios do exist where the application of curriculum learning may be beneficial. However, there also exists settings where applying particular curriculum strategies could ultimately have an adverse effect overall. These findings lead to the belief that one should use caution when selecting an appropriate curriculum strategy for any specific machine learning setting.

## 6.1 Future Work

In this study, specific settings were examined through the application of a particular curriculum learning strategy followed by analysis of the results. However, for the cases in which applying a curriculum strategy did not work, it remains feasible for one to believe that there may exist a curriculum learning strategy which may be effective. Should there exist a curriculum learning strategy which may prove effective, it may be beneficial to examine whether it is possible to develop a methodical process for uncovering the appropriate curriculum strategy in each unique machine learning setting. Should this not be true, it would mean perhaps an appropriate curriculum strategy exists only in specific scenarios.

All concepts and finding expressed in this report were accomplished through examining experimental results. A deeper understanding of the effects of curriculum learning may be understood through theoretical results. Theoretical results may explain our experimental results derived from this work, and it may also provide better guidance on how and when to apply curriculum learning in novel machine learning settings.

# Bibliography


[1] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 41–48, New York, NY, USA, 2009. ACM.

[2] Tianping Chen and Hong Chen. Universal approximation to nonlinear operators by neural networks with arbitrary activation functions and its application to dynamical systems. *IEEE Transactions on Neural Networks*, 6(4):911–917, Jul 1995.

[3] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, June 2012.

[4] Sanjoy Dasgupta. Two faces of active learning. *Theoretical Computer Science*, 412(19):1767 – 1781, 2011. Algorithmic Learning Theory (ALT 2009).

[5] Jeffrey L. Elman. Learning and development in neural networks: The importance of starting small. *Cognition*, 48(1):71–99, 1993.

[6] Lu Jiang, Deyu Meng, Qian Zhao, Shiguang Shan, and Alexander Hauptmann. Self-paced curriculum learning, 2015.

[7] A. J. Joshi, F. Porikli, and N. Papanikolopoulos. Multi-class active learning for image classification. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2372–2379, June 2009.

[8] Jyrki Kivinen and Manfred K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Information and Computation*, 132(1):1 – 63, 1997.

[9] Kai A. Krueger and Peter Dayan. Flexible shaping: How learning in small steps helps. *Cognition*, 110(3):380 – 394, 2009.

[10] M. P. Kumar, Benjamin Packer, and Daphne Koller. Self-paced learning for latent variable models. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1189–1197. Curran Associates, Inc., 2010.

[11] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[12] Burr Settles. Active learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 6(1):1–114, 2012.

[13] Ohad Shamir. Making gradient descent optimal for strongly convex stochastic optimization. *CoRR*, abs/1109.5647, 2011.

[14] P. J. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, Oct 1990.

[15] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and Intelligent Laboratory Systems*, 2(1):37 – 52, 1987. Proceedings of the Multivariate Statistical Workshop for Geologists and Geochemists.