Title
A Matter of Time: Faster Percolator Analysis via Efficient SVM Learning for Large-Scale Proteomics

Permalink
https://escholarship.org/uc/item/3842s8qc

Journal
Journal of Proteome Research, 17(5)

ISSN
1535-3893

Authors
Halloran, John T
Rocke, David M

Publication Date
2018-05-04

DOI
10.1021/acs.jproteome.7b00767

Peer reviewed

Published in final edited form as:
J Proteome Res. 2018 May 04; 17(5): 1978–1982. doi:10.1021/acs.jproteome.7b00767.

A Matter of Time: Faster Percolator Analysis via Efficient SVM Learning for Large-Scale Proteomics

John T. Halloran[†] and **David M. Rocke**[‡]
[†]Department of Public Health Sciences, University of California, Davis, CA
[‡]Division of Biostatistics, University of California, Davis, CA

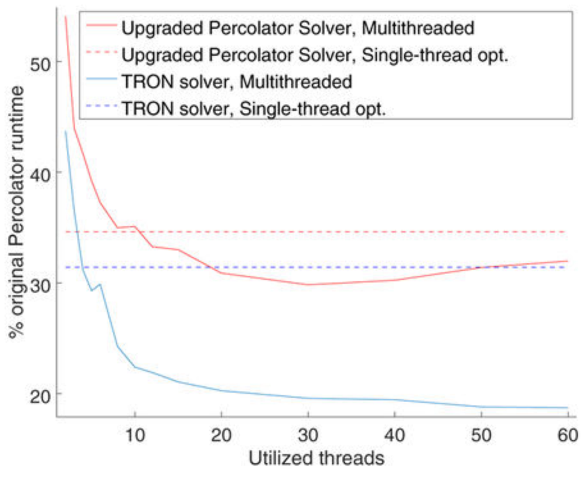
Abstract

Percolator is an important tool for greatly improving the results of a database search and subsequent downstream analysis. Using support vector machines (SVMs), Percolator recalibrates peptide-spectrum matches (PSMs) based on the learned decision boundary between targets and decoys. In order to improve analysis time for large-scale datasets, we update Percolator's SVM learning engine through software and algorithmic optimizations, rather than heuristic approaches which necessitate the careful study of their impact on learned parameters across different search settings and datasets. We show that by optimizing Percolator's original learning algorithm, l_2-SVM-MFN, large-scale SVM learning requires nearly only a third of the original runtime. Furthermore, we show that by employing the widely-used Trust Region Newton (TRON) algorithm in the stead of l_2-SVM-MFN, large-scale Percolator SVM learning is reduced to nearly only a fifth of the original runtime. Importantly, these speedups only affect the speed at which Percolator converges to a global solution and do not alter recalibration performance. Both the upgraded version of l_2-SVM-MFN and TRON are optimized within the Percolator codebase for multithreaded and single-thread use, and available under Apache license at bitbucket.org/jthalloran/percolator_upgrade.

Graphical Abstract

jthalloran@ucdavis.edu, Phone: 530-752-2793.

Supporting Information
Differences, in the number of significant PSMs at $q = 0.01$, between solving the exact large-scale problem (using Percolator's original solver, L2-SVM-MFN) and several approximate solutions found using downsampling (Supplementary Figure S–1).



Keywords

tandem mass spectrometry; machine learning; support vector machine; percolator; TRON

Introduction

Originally released a decade ago, Percolator[1] has risen as an integral tool in many tandem mass spectrometry (MS/MS) pipelines for accurate post-processing analysis of a database search. This growing prominence is greatly owed to Percolator's continued development[2,3] and its synergy with many popular search engines.[4–8] Recent work involving large-scale datasets focused on speeding up Percolator runtime by training on smaller, randomly sampled sets of PSMs (called *downsampling*[3]) to approximate the large-scale SVM parameters. In this approximate approach, the size of the random sample to be used for training is chosen at the discretion of the user, without general guarantees. However, choosing this user-specified parameter may not be generally obvious given Percolator's underlying machine learning method (i.e., randomly removing many data points, which may include support vectors of the SVM decision boundary being approximated, potentially alters the quality of the approximated parameters; see Supplementary Figure S–1) and the highly variable nature of MS/MS data (i.e., datasets and identified PSMs vary significantly based on dataset sizes, machine types, search settings, organisms, digesting enzymes, etc.). Rather than focus on an approximate approach, we instead investigate speeding up Percolator runtime through algorithmic and software improvements to its SVM learning engine. As such improvements generally speed-up Percolator training time without affecting the quality of learned parameters, the work described herein also complements future efforts where downsampling (or a similar approximate approach) is employed.

In this work, we investigate two non-heuristic speedups to Percolator post-processing: extensive optimization of the current SVM learning engine, l_2-SVM-MFN,[9] and utilizing a newer, state-of-the-art SVM learning algorithm which minimizes the same objective function, Trust Region Newton (TRON),[10,11] widely used for large-scale machine learning problems. Herein, we refer to Percolator's out-of-the-box SVM solver as l_2-SVM-MFN, our optimized version of Percolator's SVM solver as l_2-SVM-MFN*, and our implementation of the Trust Region Newton algorithm optimized for use within Percolator as TRON. l_2-SVM-MFN* and TRON reduce Percolator's out-of-the-box, large-scale SVM learning time by up to an average of **65.19%** and **79.37%**, respectively, on the benchmarked large-scale datasets for multithreaded environments. For environments limited to a single thread, specialized implementations of l_2-SVM-MFN* and TRON reduce large-scale Percolator learning time by respective averages of **60.65%** and **69.95%** on the benchmarked large-scale datasets. All optimizations were written within Percolator (version 3.01, downloaded May 31, 2017) with no dependencies on external packages. The resulting software is freely available as open-source software under the original Percolator license at bitbucket.org/jthalloran/percolator_upgrade.

Methods

SVM learning in Percolator

As input, Percolator receives the target and decoy PSMs for a database search algorithm (such as Sequest,[12] MS-GF+,[13] X!Tandem,[14] or DRIP[8,15]) along with features detailing each PSM computed during the search (e.g., score, peptide mass, mass deviation from the observed spectrum's precursor mass, etc.). Three-fold cross-validation is then used[2] (along with further cross-validation nested within each outer fold) to estimate high-quality training PSMs using target-decoy q-values[16,17] and train a discriminative classifier. The discriminative classifier learned is a linear SVM, wherein the hyperplane which maximizes the soft-margin between the target and decoy training PSMs is computed. Note that the parameters learned in an SVM are determined solely by the support vectors, i.e., the datapoints lying on the margin of the learned decision boundary.

The soft-margin SVM formulation is convex, so that a global solution is guaranteed. Percolator's SVM formulation includes an l_2 regularization term, which preserves convexity while improving generalization to unseen data. SVMs are well-suited for classifying target and decoy PSMs, as they are robust and may be trained quickly for a linear SVM like Percolator's. Once cross-validation is finished, the learned hyperplanes are than merged to form Percolator's final decision boundary and used to rescore PSMs for improved calibration. Percolator currently supports multithreading (using OpenMP) for cross-validation, wherein one SVM is trained per thread within a nest.

SVM solver—Although a general convex method (i.e., gradient descent, conjugate gradient descent, Newton's method, or L-BFGS) may be used to solve Percolator's SVM formulation, highly efficient algorithms have been developed by the machine learning community for SVM learning. At the time of its initial release in Ref. 1, Percolator's SVM learning algorithm (l_2-SVM-MFN[9]) was considered state-of-the-art and particularly efficient for large-scale problems. Subsequently, Trust Region Newton (TRON)[10] was introduced to solve the same SVM objective and shown to converge quickly on a variety of datasets.[18] TRON is a second-order algorithm wherein a region around the current solution, called the *trust region*, is adjusted based on the approximated reduction of the objective function. A truncated Newton step within the trust region is then calculated and used to update the objective weights, and the overall process is repeated until convergence. TRON remains state-of-the-art and in widespread use, with recent work investigating practical improvements to the algorithm.[19,20]

Software details

The optimized version of Percolator is freely available for download, based on Percolator version 3.01 (downloaded May 31, 2017). Percolator's implementation of l_2-SVM-MFN, originating from the C++ implementation of SVM$_{lin}$,[21] was optimized through a combination of extensive code restructuring, use of low-level linear algebra functions, and the use of multithreading in critical, bottleneck computations (development outcomes are further discussed in the Results section). TRON, based on the C implementation from LIBLINEAR[11] (version 2.11, downloaded April 24, 2017), was similarly extensively optimized for use within the Percolator codebase.

For multithreaded compute environments, computations are distributed across a user-specified number of CPU threads using the option -nr NUMTHREADS, where NUMTHREADS defaults to the maximum number of system threads if the specified value is greater than the max. For compute environments where multithreading is not an option, l_2-SVM-MFN* and TRON are specially optimized to utilize a single thread; for each objective function evaluation within an iteration, a single low-level matrix-vector multiply is performed to quickly evaluate the parameters (i.e., the learned hyperplane) calculated in the previous iteration. The single thread implementation requires slightly more memory than normal Percolator to correctly format the data matrix for the low-level matrix-vector product call, although this is negligible in practice (e.g., the memory overhead was ~ 1.6% of standard Percolator memory use, or 90 MB, for the Wu dataset).

Datasets

All Percolator pin files utilized in this work (both large-scale and development data) are available for download at jthalloran.ucdavis.edu.

Large-scale datasets—Our two large-scale benchmark sets are based on the two datasets (i.e., the Kim dataset[22] and Wu dataset[23]) and search parameters used to benchmark timing results in Ref. 3. The Kim data was generated from 17 adult tissues, seven fetal tissues, and six hematopoietic cell types, collected using an LTQ Orbitrap Velos and Elite equipped with an Easy-nLC II nanoflow LC systems. The Wu dataset, created in a study of human protein abundance variation, consists of spectra generated from 51 lysates of lymphoblastoid cell lines, where peptides were labeled with TMT 6-plex, and collected using an LTQ Orbitrap Velos. All spectra were searched with Tide[7] using Crux version 3.1 (downloaded June 21, 2017). To vary decoy generation techniques, the Kim and Wu decoy databases were generated by peptide reversal and shuffling, respectively, using Tide index. 4,459,463 spectra of the Wu dataset were searched against the IPI Human database ver. 3.74 (source, accessed: May 22, 2014) using a tryptic digestion, Tide's default fragment mass tolerance, a ±10 ppm precursor mass window, up to two missed cleavages, oxidation of methionine, and TMT labeling of N-terminal amino acids, resulting in 8,313,602 target and decoy PSMs. 4,084,132 spectra of the Kim dataset were searched against the human Swiss-Prot and

Swiss-Prot+TrEMBL databases (source, accessed: July 24 , 2017) concatenated with a database of common contaminants (source, accessed: July 24, 2017) using a semi-tryptic digestion, Tide's default fragment mass tolerance, a ±10 ppm precursor mass tolerance window, up to two missed cleavages, up to two oxidations of methionine per peptide, and variable acetylation of N-termini, resulting in 7,710,057 target and decoy PSMs.

Development datasets—During development, two small-scale datasets were used to debug and chart the relative improvement of successive optimizations. The first, which we refer to as Yeast, consists of 35,467 *Saccharomyces cerevisiae* spectra collected using a tryptic digestion followed by acquisition using low-resolution precursor scans and low-resolution fragments ions (further described in Ref. 1). The second, which we refer to as Plasmodium, consists of 12,594 spectra collected from a *Plasmodium falciparum* sample digested using Lys-C and labeled with an isobaric TMT relabeling agent, collected using high-resolution precursor scans and high-resolution fragment ions (further described in Ref. 24).

Decoy peptides were created by shuffling target peptides. Plasmodium was searched using Tide with a ±50 ppm precursor mass window, Lys-C, a fixed carbamidomethylation, and a fixed TMT labeling of lysine and N-terminal amino acids, resulting in 23,922 target and decoy PSMs. Yeast was searched using Tide with a ±3 Thomson precursor mass window, trypsin without proline suppression, no missed cleavages, and a fixed carbamidomethylation, resulting in 140,346 target and decoy PSMs. To further stress test the algorithms during development, seventeen in-house features detailing each PSM were added to the Yeast and Malaria pin files output by Tide. XCorr *p*-values were also calculated, using Tide, for the low-resolution MS2 Yeast dataset and appended to the corresponding pin file.

Experimental environment

All Percolator experiments were run using a multicore compute server with one terabyte of RAM, comprised of Intel Xeon E7–4830 v3 CPUs clocked at 2.10 GHz. For all timing tests, multithreading in Percolator's cross-validation procedure was disabled to reduce extra scheduling overhead and accurately measure the speed of the different SVM learning algorithms.

Results

Development

Optimization progress was marked by several major development stages:

1. **Initial** - successful implementation of the learning algorithm within Percolator's SVM framework. This refers to TRON, although l_2-SVM-MFN is also illustrated for this stage in Figure 1 for completeness.
2. **Restructured** - code restructuring and simplification. This stage focused on streamlining the code of the SVM learning algorithm. l_2-SVM-MFN* most benefited from this, as large portions of code were significantly restructured and condensed.

3. **low-level** - many operations were optimized using low-level linear algebra functions (this greatly reduced TRON runtime due to Percolator's dense, rather than sparse, feature representation).

4. **single-thread** - the SVM learning algorithm was optimized for use with a single thread (objective-function evaluations in both TRON and l_2-SVM-MFN* were optimized using a low-level matrix-vector multiply).

5. **nr-k** - multithreading with *k* threads. For both learning algorithms, the matrix-vector multiply in the single-thread optimization is instead parallelized. For TRON, this includes parallelization of the Hessian evaluation and computation of the gradient (further discussed in Ref. 25). For l_2-SVM-MFN*, this new work included parallelization of major computational bottlenecks in its conjugate gradient and line search procedures.

The benchmarked runtime on the development datasets after each stage is illustrated in Figure 1. Multithreading was anticipated to have a significant impact on large-scale experiments, but was not expected to improve on the highly optimized single-thread implementation for the development datasets due to their small size (as turned out to be the case for l_2-SVM-MFN*, where the single-thread optimized implementation performed best on the dev sets and multithreading was far more impactful on the large-scale runtime). However, owing to the efficiency of the algorithm and its streamlined design, TRON with multiple threads proved to be extremely fast even on such small datasets, significantly outperforming its single-thread optimized counterpart. We note that, due to the reliance of l_2-SVM-MFN on a relative stopping heuristic to terminate the algorithm early, the learned parameters using TRON may differ slightly. However, no substantial difference in performance has been observed due to any slight difference in learned parameters (Figures 2 and 4).

Attempting to improve the speed further by mixing the single-thread optimization with multithreading was found not to improve runtime, and obfuscated the codebase.

Large-scale timing results

Runtimes are plotted as the ratio of TRON or l_2-SVM-MFN* runtime divided by the runtime of the original l_2-SVM-MFN Percolator implementation. All Percolator runtimes (using l_2-SVM-MFN, l_2-SVM-MFN*, and TRON learning algorithms) were averaged over ten runs, for a total of 620 timing tests conducted. The Percolator speedup afforded using l_2-SVM-MFN* and TRON with multiple threads was tested using -nr set to 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 30, 40, 50, and 60 (plotted in blue in Figure 3). When -nr is set to 1, the single-thread optimized versions of TRON and l_2-SVM-MFN* are utilized (plotted in red in Figure 3). Only SVM training time was evaluated in all reported runtimes, measured as the elapsed time from training start to stop.

TRON and l_2-SVM-MFN* improve Percolator runtime for large-scale analysis in all experiments. l_2-SVM-MFN* optimized for single-threading reduces Percolator runtime by 55.91% and 65.38% (2.27 and 2.89 fold speedup) for the Kim and Wu datasets, respectively. l_2-SVM-MFN* with multithreading reduced Percolator runtime by 60.21% and 70.17%

(2.51 and 3.35 fold speedup) for the Kim and Wu datasets, respectively. The single-thread optimized TRON implementation reduces Percolator runtime by 71.31% and 68.59% (3.49 and 3.18 fold speedup) for the Kim and Wu datasets, respectively. For multithreaded environments, TRON reduces Percolator runtime by 77.46% and 81.28% (4.44 and 5.34 fold speedup) for the Kim and Wu datasets, respectively. This saves hours of Percolator runtime in nearly all cases (TRON with nr ≥ 10 finished in under an hour on both datasets), without any degradation in recalibration performance.

Conclusions

We've shown that Percolator runtime may be significantly improved through algorithmic speedups to its current SVM learning algorithm, l_2-SVM-MFN. Moreover, Percolator runtime may be even further improved using the state-of-the-art TRON learning algorithm. For large-scale analysis, these speedups save several hours of analysis time, both for multithreaded and single-threaded compute environments. Importantly, as these improvements are algorithmic and software optimizations, they do not compromise the learned parameters (and subsequent recalibration performance) and are complementary to the future analysis and utilization of approximate methods such as downsampling.

Supplementary Material

Refer to Web version on PubMed Central for supplementary material.

Acknowledgments:

This work was supported by the National Center for Advancing Translational Sciences (NCATS), National Institutes of Health, through grant UL1 TR001860.

References

(1). Kall L; Canterbury J; Weston J; Noble WS; MacCoss MJ A semi-supervised machine learning technique for peptide identification from shotgun proteomics datasets. Nat. Methods 2007, 4, 923–25. [PubMed: 17952086]

(2). Granholm V; Noble WS; Kall L A cross-validation scheme for machine learning algorithms in shotgun proteomics. BMC bioinformatics 2012, 13, S3.

(3). Matthew T; MacCoss MJ; Noble WS; Käll L Fast and accurate protein false discovery rates on large-scale proteomics data sets with percolator 3.0. Journal of The American Society for Mass Spectrometry 2016, 27, 1719–1727. [PubMed: 27572102]

(4). Brosch M; Yu L; Hubbard T; Choudhary J Accurate and sensitive peptide identification with Mascot Percolator. J. Proteome Res 2009, 8, 3176–3181. [PubMed: 19338334]

(5). Granholm V; Kim S; Navarro JC; Sjolund E; Smith RD; Kall L Fast and accurate database searches with MS-GF+ Percolator. J. Proteome Res 2013, 890–897. [PubMed: 24344789]

(6). Xu M; Li Z; Li L Combining percolator with X! Tandem for accurate and sensitive peptide identification. Journal of proteome research 2013, 12, 3026–3033. [PubMed: 23581882]

(7). McIlwain S; Tamura K; Kertesz-Farkas A; Grant CE; Diament B; Frewen B; Howbert JJ; Hoopmann MR; Kall L; Eng JK; MacCoss MJ; Noble WS Crux: rapid open source protein tandem mass spectrometry analysis. J. Proteome Res 2014, 13, 4488–4491. [PubMed: 25182276]

(8). Halloran JT; Bilmes JA; Noble WS Dynamic Bayesian Network for Accurate Detection of Peptides from Tandem Mass Spectra. Journal of Proteome Research 2016, 15, 2749–2759. [PubMed: 27397138]

(9). Keerthi SS; DeCoste D A modified finite Newton method for fast solution of large scale linear SVMs. Journal of Machine Learning Research 2005, 6, 341–361.

(10). Lin C-J; Weng RC; Keerthi SS Trust region newton methods for large-scale logistic regression. Proceedings of the 24th international conference on Machine learning. 2007; pp 561–568.

(11). Fan R-E; Chang K-W; Hsieh C-J; Wang X-R; Lin C-J LIBLINEAR: A library for large linear classification. Journal of machine learning research 2008, 9, 1871–1874.

(12). Eng JK; McCormack AL; Yates JR, III An approach to correlate tandem mass spectral data of peptides with amino acid sequences in a protein database. 1994, 5, 976–989.

(13). Kim S; Pevzner PA MS-GF+ makes progress towards a universal database search tool for proteomics. 2014, 5, 5277.

(14). Craig R; Beavis RC TANDEM: matching proteins with tandem mass spectra. Bioinformatics 2004, 20, 1466–1467. [PubMed: 14976030]

(15). Halloran JT; Rocke DM Gradients of Generative Models for Improved Discriminative Analysis of Tandem Mass Spectra. Advances in Neural Information Processing Systems. 2017; pp 5728–5737.

(16). Storey JD A direct approach to false discovery rates. J. R. Stat. Soc 2002, 64, 479–498.

(17). Keich U; Kertesz-Farkas A; Noble WS Improved False Discovery Rate Estimation Procedure for Shotgun Proteomics. J. Proteome Res 2015, 14, 3148–3161. [PubMed: 26152888]

(18). Lin C-J; Weng RC; Keerthi SS Trust region newton method for large-scale logistic regression. Journal of Machine Learning Research 2008, 9, 627–650.

(19). Lee M-C; Chiang W-L; Lin C-J Fast matrix-vector multiplications for large-scale logistic regression on shared-memory systems. Data Mining (ICDM), 2015 IEEE International Conference on. 2015; pp 835–840.

(20). Hsia C-Y; Zhu Y; Lin C-J A study on trust region update rules in Newton methods for large-scale linear classification. Asian Conference on Machine Learning. 2017; pp 33–48.

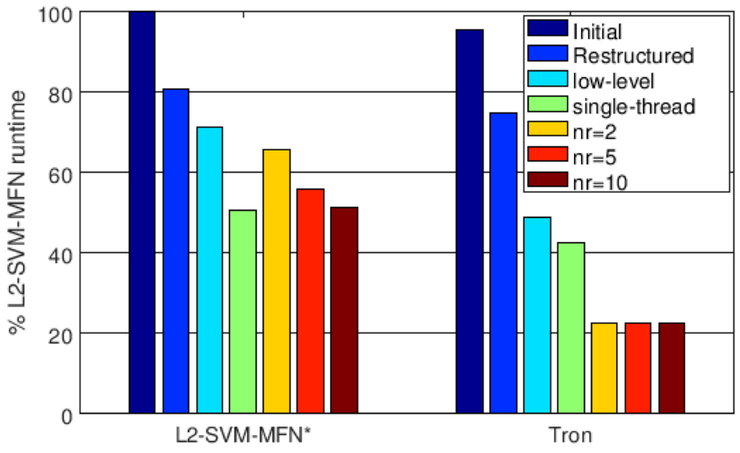
(21). Sindhwani V; Keerthi SS Newton methods for fast solution of semi-supervised linear SVMs. Large scale kernel machines 2007, 155–174.

(22). Kim M-S; Pinto SM; Getnet D; Nirujogi RS; Manda SS; Chaerkady R; Madugundu AK; Kelkar DS; Isserlin R; Jain S A draft map of the human proteome. Nature 2014, 509, 575. [PubMed: 24870542]

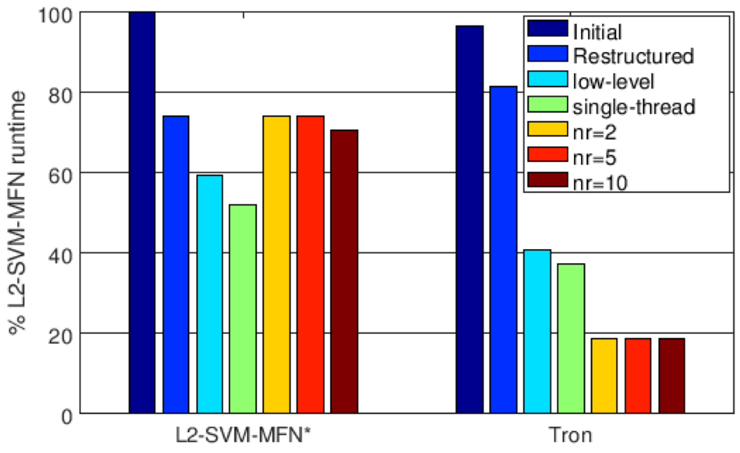
(23). Wu L; Candille SI; Choi Y; Xie D; Jiang L; Li-Pook-Than J; Tang H; Snyder M Variation and genetic control of protein abundance in humans. Nature 2013, 499, 79–82. [PubMed: 23676674]

(24). Pease BN; Huttlin EL; Jedrychowski MP; Talevich E; Harmon J; Dillman T; Kannan N; Doerig C; Chakrabarti R; Gygi SP; Chakrabarti D Global analysis of protein expression and phosphorylation of three stages of Plasmodium falciparum intraerythrocytic development. J. Proteome Res 2013, 12, 4028–4045. [PubMed: 23914800]

(25). Chiang W-L; Lee M-C; Lin C-J Parallel Dual Coordinate Descent Method for Large-scale Linear Classification in Multi-core Environments. KDD. 2016; pp 1485–1494.



(a) Yeast



(b) Plasmodium

Figure 1:

Percentage of original Percolator SVM learning runtime for the developments datasets after each set of major l_2-SVM-MFN* and TRON speedups. SVM learning using the original l_2-SVM-MFN took 174.6 seconds and 27.2 seconds for the Yeast and Plasmodium datasets, respectively.

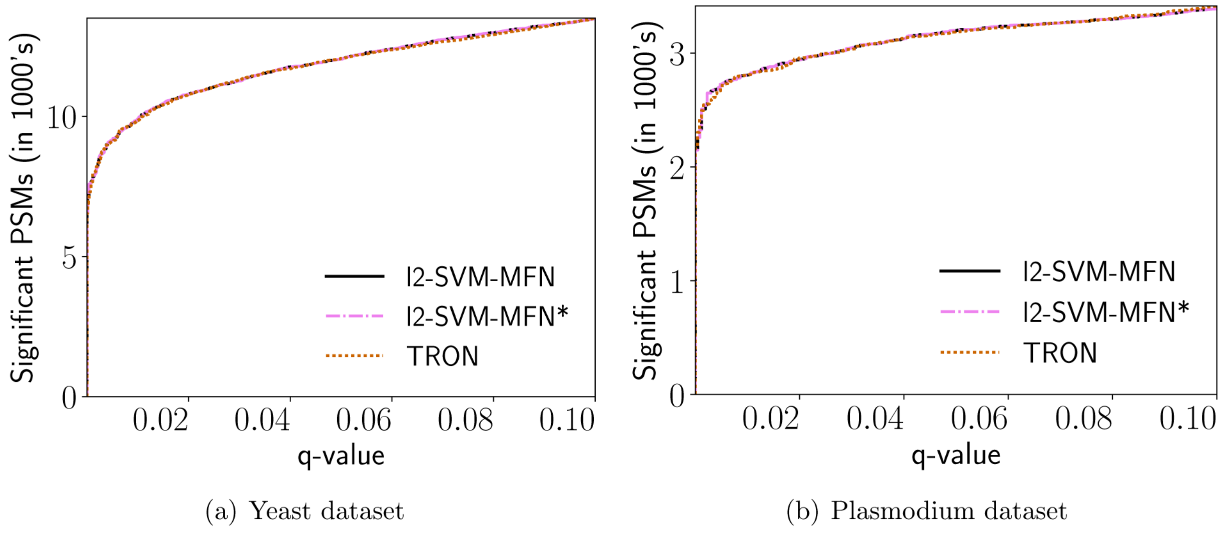


Figure 2:
Percolator postprocessing accuracy over the development sets using the different SVM learning algorithms. Plotted are the *q*-values versus number of significant PSMs after Percolator recalibration of target and decoy PSMs using l_2-SVM-MFN, l_2-SVM-MFN*, and TRON algorithms.

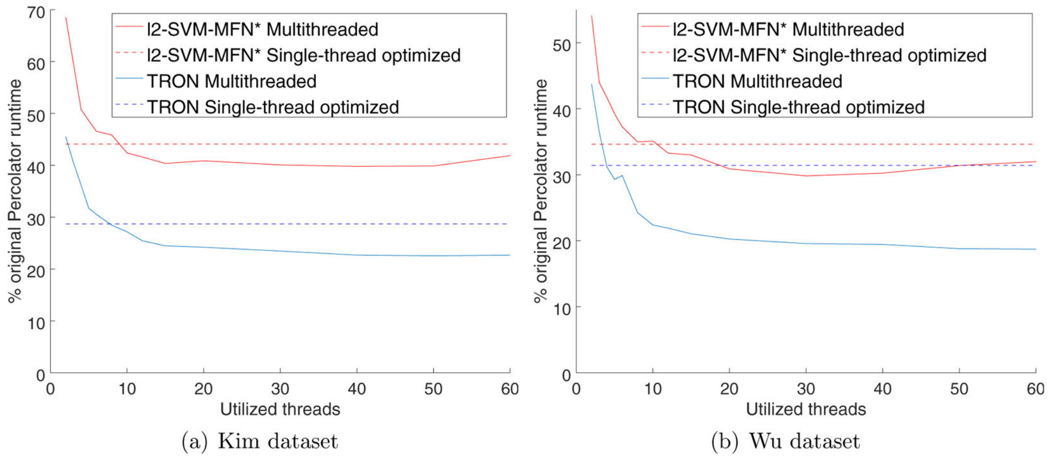


Figure 3:

Percolator runtime using l_2-SVM-MFN* and TRON versus the original implementation of l_2-SVM-MFN. The *y*-axis denotes the runtime of Percolator using l_2-SVM-MFN* (plotted in red) and TRON (plotted in blue) divided by the runtime of Percolator using l_2-SVM-MFN. The *x*-axis denotes the number of threads utilized for l_2-SVM-MFN* multithreading (solid red curve) and TRON multithreading (solid blue curve). The runtimes of l_2-SVM-MFN* and TRON optimized for a single thread are illustrated in dashed red and dashed blue, respectively. All reported search times are the average of ten runs (the average l_2-SVM-MFN Percolator runtime was 2.94 hours and 3.89 hours for the Kim and Wu datasets, respectively).

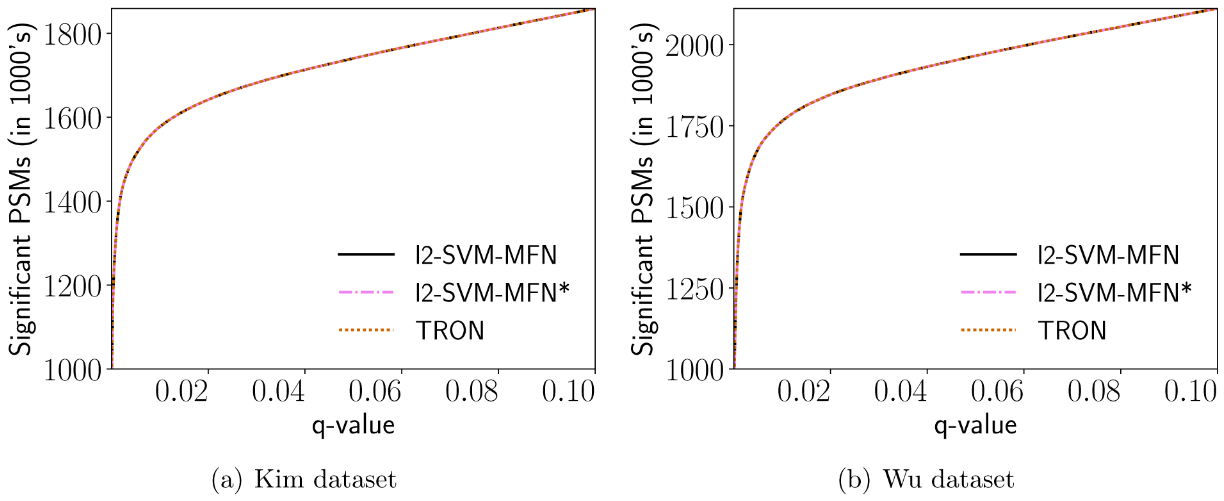


Figure 4:

Percolator postprocessing accuracy over large-scale benchmark datasets. Plotted are the *q*-values versus number of significant PSMs after Percolator recalibration of target and decoy PSMs using l_2-SVM-MFN, l_2-SVM-MFN*, and TRON algorithms.