

RESEARCH ABSTRACT
Nikos Karampatziakis

My research interests are broad covering both theory and applications of machine learning in real world problems. I particularly focus on large scale problems, online learning, optimization, and structured input/output prediction. In my work I derive and apply principled machine learning approaches to solve challenging problems with large scale being a common theme among them. This reflects the ever-increasing volumes of structured and unstructured data that characterize today's and tomorrow's real world problems.

Some examples of the projects I have worked on include online learning of a context tree under resource constraints [20], motivated by predicting a program's system calls for anomaly detection (typical executions consist of billions of system calls), a novel structured output prediction approach to malicious code detection where the inference problem reduces to weighted interval scheduling [19] (typical programs consist of millions of bytes), and new and robust types of updates for online gradient descent (OGD) based on solutions of differential equations [18]. The latter type of updates also enable an extremely fast active learning algorithm [3]. Again OGD and active learning are typically applied to data sets that do not fit in memory or that are too big to be fully labeled. I give a brief overview of all my projects and future goals below.

Contributions

Online Learning of Prediction Suffix Trees: Consider the task of deciding if a monitored application is behaving maliciously. During the execution of a program, we can intercept the system calls it makes to the operating system. It is widely believed that this sequence of system calls characterizes the behavior of a program. If at any time the predicted system call is very different from the actual system call we can report this to the administrator. The model we use to decide which system call will occur next is a prediction suffix tree (also known as context tree), a relatively exotic model (though it has recently been used in impressive applications [24]), which has several advantages over n -grams. Hence predicting the next system call given the previous ones arises very naturally as online learning of a prediction suffix tree.

For this prediction task I designed a multiplicative update online algorithm to learn the parameters of a prediction suffix tree. Besides enjoying strong worst-case bound on regret, the amount of space (parameters) used by the model should be as small as possible without sacrificing much of the prediction performance. This enables us to do faster inference and maintain a small memory footprint while monitoring the application. Furthermore, we do not use any a priori assumptions on how large the learned model should be; the algorithm figures this out by itself. In [20] we present an algorithm that nicely balances the trade-off between the number of parameters and the cumulative loss of the algorithm. The context used to make a prediction scales logarithmically with the number of mistakes the algorithm makes. Furthermore, the algorithm has a mistake bound similar to that of a standard multiplicative update algorithm. Finally, our experiments showed that this algorithm learns smaller and more accurate models than other existing algorithms [11], because the multiplicative update is more suitable for learning our context tree.

Structured Prediction for Binary Code Analysis: A nice application of structured input/output prediction arises in binary code analysis: given a binary executable, determine what it does and whether it's malicious. First we need to find where the actual instructions inside the executable are. This is nontrivial because the executable does not have any demarcations indicating, say, the beginning of a block of code. Furthermore the task is not as easy as tagging the whole byte sequence with a Hidden Markov Model. There are long range dependencies and a Markovian assumption is not always appropriate. The task is also not as hard as sequence segmentation, because, once we know where a block of code starts, we can deduce where it will end: instructions are executed sequentially until an instruction that changes the flow of control is reached. What we want to predict is a *schedule*, a set of intervals that do not overlap. In [19] I present a support vector machine for predicting such schedules and showed that it outperforms (discriminatively trained) Hidden Markov Models while having the same running time for learning and inference. Both learning and inference are based on a variation of weighted interval scheduling algorithm whose running time scales linearly in the size of the sequence

In follow-up ongoing work, the schedule is treated as a hidden variable and the model is a conditional random field that is trained to predict whether the program is malicious or not, by summing over all (exponentially many) schedules, in time linear in the length of the sequence.

New Algorithms for the Contextual Bandit Problem: The contextual bandit problem is an important abstraction for capturing exploration/exploitation trade-offs. Before making its decision, the algorithm is given some context and must perform almost as well as the best policy from a set of policies of interest (say, all decision trees of depth up to k) that map this context to an action. As usual, in bandit problems we do not observe the rewards of the actions we did not choose. In the vast majority of applications involving exploration/exploitation trade-offs, context information is readily available. Prior to this work, contextual bandit algorithms either achieved low regret but had to enumerate over a potentially exponential-sized set of policies [5], or they had polynomial complexity but did not achieve optimal regret [21]. The algorithm we describe in [13] has both properties in the iid setting and only assumes oracle access to a cost sensitive learning algorithm. An interesting feature of our new algorithm is that it uses a completely different proof technique, based on a new minimax argument, that may lead to new insights about the problem.

Better Updates for Online Gradient Descent: Online Gradient Descent (OGD) is a simple and surprisingly effective learning algorithm. It can be derived formally by arguing that a learning algorithm should minimize the (first order Taylor approximation of the) loss it attains on the current example while staying close to what it already knows. In this view, OGD is like an Euler integrator for solving the gradient flow differential equation. In [18] we observe that for linear models and for many common loss functions we can solve this differential equation in closed form for each example. The new updates satisfy the same regret guarantees as previous update rules for OGD and are more robust in the exact setting of the learning rate. Furthermore, they work well when the examples can have large importance weights attached to them as it often happens in exploration settings or covariate shift problems. These updates are used by default in VowpalWabbit, an open source project focused on fast learning, essentially eliminating parameter tuning.

Efficient Active Learning: The above updates imply that it is possible to have a fast algorithm that works correctly for problems where large importance weights are inevitable. This is the case for example in the active learning approach of [4]. In [3], we empirically showed that basic gradient descent cannot handle those large importance weights correctly and effectively negates the advantage of active learning. The updates we proposed in [18] are robust to large importance weights and lead to an extremely fast active learning algorithm that can typically save a factor of 3 to 10 in the number of labeled examples. I have implemented this in the VowpalWabbit software.

Empirical Comparison of Supervised Learning in High Dimensions: In a project with Rich Caruana whose somewhat surprising results are reported in [7] we studied the performance of popular learning algorithms in high dimensional binary classification problems. We found that methods based on random projections such as random forests, can sometimes outperform linear classifiers. The insights of our work (and some of our code) were subsequently used by the winning team in the KDD Cup 2009, which involved predicting well on a high dimensional data set.

Parallel Learning: The running time of a learning algorithm depends on the architecture of the computer on which it runs. A few years ago, CPU frequencies stopped increasing because of fundamental physical constraints. Moore’s law still continues to hold though and is now manifested via the number of cores of a typical computer. On a single core, we know that, asymptotically, online gradient descent (OGD) is essentially optimal [6]. What happens when the architecture is parallel? Some preliminary results on this line of research are reported in [16] and in a book chapter [17]. The most exciting thing about research in this area is that it is driven both by theory and by technology advances. New architectures appear all the time. For example, GPUs can now be found in commodity machines on the cloud. Finally, some learning algorithms are inherently more parallel than others, which means that, as technology advances, we need to always reevaluate what we consider to be the fastest algorithm.

Future Research

One of my main goals in the future is to turn large scale learning and learning in exploration settings into a technology. By “technology” I mean, to develop methods for these settings that provably work and work

well with little or no tuning. Below I describe some directions that can contribute towards this goal.

Exploration: The algorithm we developed in [13] is optimal and polynomial, but not efficient. We only exploit convexity of a certain optimization problem. However it can also be formulated as a second order cone program (SOCP) albeit with an exponential number of variables and constraints. A beautiful result of Ben-Tal and Nemirovski [2] roughly says that we can approximate SOCPs extremely well by slightly larger linear programs. The linear program is still non-trivial because of the exponential number of constraints and variables however there exist cases when such linear programs can be efficiently solved [12].

With the above exception, most other algorithms are derived and analyzed using a potential function. In [9] a new doubly exponential potential and corresponding algorithm were proposed for the full feedback expert setting with appealing properties like no tuning parameters. It would be interesting to obtain similar results in the bandit setting or when side information is available to the algorithm.

Empirically, a very effective way for handling the exploration/exploitation trade-off in the stochastic setting is an extremely simple-to-state scheme called Thompson Sampling [8], used by Microsoft to recommend ads [14]. When rewards are generated from an unknown fixed distribution it empirically outperforms well-tuned variants of the UCB policy [8]. Recently [1] showed logarithmic regret for the case of Bernoulli rewards. The cases of Gaussian rewards and contextual bandits do not have an analysis. Furthermore, the analysis for the Bernoulli case does not reflect the empirical superiority of Thompson sampling.

Fast and reliable stopping for stochastic gradient descent: The holy grail of stochastic optimization for machine learning is to be able to find a model with the same generalization error as a batch learning algorithm while performing only one pass over the training data. We are still quite far from this goal especially for medium sized datasets, for which stochastic gradient is the fastest algorithm but one pass is not enough. When do we stop training? Many machine learning tasks are convex and hence it is possible to estimate the suboptimality of a candidate solution. I have lately been thinking of two cheap ways to compute estimates of (an upper bound on) the duality gap as we run stochastic gradient descent. For large problems, such stopping criteria open the possibility of learning without even going through the whole input.

Stochastic Bundle Methods: Bundle methods are sophisticated convex optimization procedures. For regularized machine learning problems they need $O(\log(\epsilon^{-1}))$ gradient evaluations to reach an ϵ -suboptimal solution [23]. Some lines of research have used stochastic gradients when exact gradient calculation is expensive [25] and massaged the standard bundle methods which aren't robust to noisy gradients. Can we do better? Obviously we cannot get around lower bounds for black box stochastic gradient descent. To work around this barrier ideas similar to those in [22] can be employed. The proposed scheme works well with stochastic gradients and bears some similarity with bundle methods. With respect to convergence, the methods in [22] only improve the constants. Is it possible that by exploiting more the structure of the problems we care about in machine learning we can achieve better rates of convergence?

Online algorithms that do more per iteration: In today's computers, carefully engineered online learning software is bound by the speed by which it can read data (from the network or a disk –even a solid state one). In the meantime there are many applications that require more than learning a single linear classifier from the given data. Examples include maintaining uncertainty about the model, tracking, and second order algorithms but there are many others. Maintaining uncertainty information is useful both for practitioners who care about the reliability of a prediction as well as in the design of new algorithms especially when exploration is necessary. An online algorithm that maintains a committee of models and measures disagreement among them is certainly implementable without any slowdown. A committee can also be used for tracking in the spirit of [15] who prove good adaptivity using a strong notion of regret. Second order algorithms can provide benefits in theory, such as improved regret in multiclass classification with bandit feedback [10] and in practice since in many applications many inputs are extremely correlated (e.g. the presence of a bigram feature implies the presence of the two unigram features that comprise it). In high dimensional problems we cannot maintain the full covariance matrix necessary for capturing all correlations in the data. However because of the extreme aforementioned correlations and the nature of most datasets the intrinsic dimension of the data may be small. What can we say about learning algorithms that maintain cheap low rank approximations of the covariance when the data actually live in a low dimensional subspace?

References

- [1] S. Agrawal and N. Goyal. Analysis of Thompson Sampling for the Multi-Armed Bandit Problem. *Arxiv preprint arXiv:1111.1797*, 2011.
- [2] A. Ben-Tal and A. Nemirovski. On polyhedral approximations of the second-order cone. *Mathematics of Operations Research*, pages 193–205, 2001.
- [3] A. Beygelzimer, D. Hsu, N. Karampatziakis, J. Langford, and T. Zhang. Efficient active learning. In *Workshop on Learning on Online Trading of Exploration and Exploitation 2*, 2011.
- [4] A. Beygelzimer, D. Hsu, J. Langford, and T. Zhang. Agnostic Active Learning Without Constraints. In *Proc. of Neural Information Processing Systems (NIPS)*, 2010.
- [5] Alina Beygelzimer, John Langford, Lihong Li, Lev Reyzin, and Robert E. Schapire. Contextual bandit algorithms with supervised learning guarantees. *Journal of Machine Learning Research - Proceedings Track*, 15:19–26, 2011.
- [6] L. Bottou and O. Bousquet. The tradeoffs of large scale learning. *Advances in neural information processing systems*, 20:161–168, 2008.
- [7] Rich Caruana, Nikos Karampatziakis, and Ainur Yessenalina. An empirical evaluation of supervised learning in high dimensions. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 96–103, New York, NY, USA, 2008. ACM.
- [8] Olivier Chapelle and Lihong Li. An Empirical Evaluation of Thompson Sampling. *Neural Information Processing Systems (NIPS)*, 2011.
- [9] Kamalika Chaudhuri, Yoav Freund, and Daniel Hsu. A parameter-free hedging algorithm. *Advances in Neural Information Processing Systems*, pages 1–9, 2009.
- [10] Koby Crammer and Claudio Gentile. Multiclass classification with bandit feedback using adaptive regularization. In *ICML*, pages 273–280, 2011.
- [11] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The power of selective memory: Self-bounded learning of prediction suffix trees. *Advances in Neural Information Processing Systems*, 17, 2004.
- [12] M. Dudík and G.J. Gordon. A sampling-based approach to computing equilibria in succinct extensive-form games. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, pages 151–160. AUAI Press, 2009.
- [13] Miroslav Dudík, Daniel Hsu, Satyen Kale, Nikos Karampatziakis, John Langford, Lev Reyzin, and Tong Zhang. Efficient optimal learning for contextual bandits. In *UAI*, pages 169–178, 2011.
- [14] T. Graepel, J.Q. Candela, T. Borchert, and R. Herbrich. Web-scale bayesian click-through rate prediction for sponsored search advertising in microsoft’s bing search engine. In *Proceedings of the Twenty-Seventh International Conference on Machine Learning (ICML-10)*, pages 13–20. Citeseer, 2010.
- [15] E. Hazan and C. Seshadhri. Efficient learning algorithms for changing environments. In *Proceedings of the 26th Annual International Conference on Machine Learning (ICML)*, page 50, 2009.
- [16] Daniel Hsu, Nikos Karampatziakis, and John Langford. Parallel online learning. In *Workshop on Learning on Cores, Clusters and Clouds*, 2010.
- [17] Daniel Hsu, Nikos Karampatziakis, John Langford, and Smola Alex. *Scaling Up Machine Learning*, chapter Parallel Online Learning. Cambridge University Press, 2011.

- [18] N. Karampatziakis and J. Langford. Online importance weight aware updates. In *Uncertainty in Artificial Intelligence*, 2011.
- [19] Nikos Karampatziakis. Static analysis of binary executables using structural svms. In *Neural Information Processing Systems (NIPS)*, 2010.
- [20] Nikos Karampatziakis and Dexter Kozen. Learning prediction suffix trees with Winnow. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 489–496. ACM, 2009.
- [21] J. Langford and T. Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. *Advances in Neural Information Processing Systems*, 20, 2007.
- [22] Y. Nesterov. Primal-dual subgradient methods for convex problems. *Mathematical programming*, 120(1):221–259, 2009.
- [23] C.H. Teo, SVN Vishwanthan, A.J. Smola, and Q.V. Le. Bundle methods for regularized risk minimization. *The Journal of Machine Learning Research*, 11:311–365, 2010.
- [24] S. Venkataraman, A. Blum, D. Song, S. Sen, and O. Spatscheck. Tracking dynamic sources of malicious activity at internet-scale. In *Proc. of Neural Information Processing Systems (NIPS)*, 2009.
- [25] Chun-Nam John Yu and Thorsten Joachims. Training Structural SVMs with Kernels Using Sampled Cuts. In *KDD*, pages 794–802, 2008.