SVM Learning of IP Address Structure for Latency Prediction

Robert Beverly
MIT CSAIL
rbeverly@csail.mit.edu

Karen Sollins
MIT CSAIL
sollins@csail.mit.edu

Arthur Berger
MIT/Akamai
awberger@csail.mit.edu

ABSTRACT

We examine the ability to exploit the hierarchical structure of Internet addresses in order to endow network agents with predictive capabilities. Specifically, we consider Support Vector Machines (SVMs) for prediction of round-trip latency to random network destinations the agent has not previously interacted with. We use kernel functions to transform the structured, yet fragmented and discontinuous, IP address space into a feature space amenable to SVMs. Our SVM approach is accurate, fast, suitable to on-line learning and generalizes well. SVM regression on a large, randomly collected data set of 30,000 Internet latencies yields a mean prediction error of 25ms using only 20% of the samples for training. Our results are promising for equipping end-nodes with intelligence for service selection, user-directed routing, resource scheduling and network inference. Finally, feature selection analysis finds that the eight most significant IP address bits provide surprisingly strong discriminative power.

Keywords

IP networks, SVM, Machine Learning, Latency, Prediction

1. INTRODUCTION

The hierarchical structure of the Internet and address allocation policies [9] provide locality to IP addresses. This work exploits the IP address structure in order to endow network agents with predictive abilities. Based on prior interaction with the network, we examine an agent's ability to successfully predict latency to random hosts the agent has not previously interacted with. Specifically, we cast the problem of predicting round-trip network latency to an IP destination as an application of machine learning.

IP addresses must be globally unique and are therefore assigned by regional registries which are in turn governed by a central body, the Internet Assigned Numbers Authority (IANA). Wherever possible, addresses are delegated in a semi-organized and hierarchical fashion with the intention that networks, organizations and geographic regions receive contiguous blocks of IP address space. Contiguous allocations permit aggregation of routing announcements, preserve router memory and reduce BGP convergence time. While Internet IP address space maintains hierarchy and structure, it has evolved organically over time. As a result the address space is discontinuous, variable and fragmented [4, 12] as evidenced by the approximately 200,000 BGP entries in the global routing table [10]. Our research investigates the use of kernel functions to transform the Internet address space into a feature space amenable to Support Vector Machine (SVM) [18] learning methods.

The ability to learn and predict network latencies to random destinations is potentially useful in a variety of practical applications. For example, our results provide an estimation accuracy granularity suitable for: service selection, where multiple copies of the same resource are geographically distributed; user-directed routing via overlays or IPv6 logical interfaces; resource scheduling in grid environments; and network inference.

In the larger architectural context, we believe machine learning is particularly well-suited to providing intelligence in today's Internet and future networks. Machine learning techniques are most successful in complex and dynamic environments which can accommodate and recover from infrequent predictive errors. For example, an agent's prediction of the best server among a set may be incorrect but is not fatal. The incorrect prediction may simply lower performance or cause the agent to adaptively choose a different server. Further, the high-levels of aggregation and traffic in the network provide a large and continuous training set for on-line learning and adaptation. We hope our work serves as a step forward in the area of latency prediction and general network intelligence using recent statistical learning methods. The primary contributions of this paper are:

1. Validation of SVMs and kernel functions as methods to learn on the basis of the Internet address space.

2. A feature selection analysis of the informational content of IP addresses. We find that eight bits provide strong discriminative power in determining latency.

3. An estimation accuracy within 30% of the true value for approximately three-quarters of the latency predictions on a large, live Internet data set. We obtain this performance without any prior interaction with the target, using only 20% of our data samples for training the SVM.

SIGCOMM'06 Workshops September 11-15, 2006, Pisa, Italy.

2. LATENCY PREDICTION

Latency prediction is a difficult, but relevant research topic with many potential practical applications; Section 4 details several applications that motivate this research. Using recursive DNS queries, King [8] estimates latencies between arbitrary pairs of Internet hosts. King's method assumes DNS servers are in close proximity to the hosts they are authoritative for and requires an active probe. Vivaldi [6] is a scheme which defines a synthetic coordinate system in order to predict latencies. Vivaldi is a distributed algorithm that requires nodes to query each other to establish their relative position in the coordinate space. Meridian [19] is a distributed network location system using concentric ring queries rather than a virtual coordinate system.

We hypothesize that agents can exploit the inherent IP address locality to their advantage. An agent in the network can be any device that is attempting to make decisions based upon previous interactions with the network[1]. It is reasonable to believe that latency, congestion and throughput are much more likely to be consistent amongst destinations all within the same subnetwork as compared to random nodes drawn from the entire network[2]. In the case of latency, the fundamental speed of light limit imposes a lower bound on round-trip delay measurements in the face of network noise and variation.

In this Section, we summarize SVMs using a simple example of placing IP addresses into a latency class. With this intuition, we describe our SVM-based methodology and data set. In contrast to prior work, our effort is designed for individual, intelligent network agents and does not presume any additional network infrastructure, overlays or network-layer assistance. Our technique is predictive on the basis of prior learning: an agent forms a latency estimate for a random, remote end-node with which it has never previously interacted.

2.1 SVM Methodology

SVMs work well in many learning situations because they generalize to unseen data: the machine is defined by only a subset of the training points, or support vectors. For basic classification into two types, SVMs find a hyperplane that provides a maximal separation between classes. This optimal hyperplane is orthogonal to the shortest line connecting the convex hulls of the two classes in some dimensional space. The support vectors are exactly those data points which define this shortest line. Thus, SVMs maximize the minimal margin. Additional data points do not affect the final solution unless they redefine the margin. For this reason, SVMs are amenable to continuous, adaptive on-line learning, a desirable property in network environments.

The semi-structured IP address space is a challenging foundation for learning latency. While an agent's latency to machines on a particular subnetwork may be within a tight bound, there exist other subnetworks with an identical bound that are numerically distant in the IP space. Analysis of geographic locality in BGP prefixes [7] finds that autonomous systems commonly advertise multiple discontiguous prefixes corresponding to a single location. Because of this address discontinuity, we turn to kernel functions [18].

[1]Our intent is that every autonomous agent in the network build an independent view of the network in order to form predictions that maximize individual utility.

[2]Intra-network consistency is naturally not absolute, but our work is concerned with providing a *most likely prediction* for applications that can compensate for occasional errors.

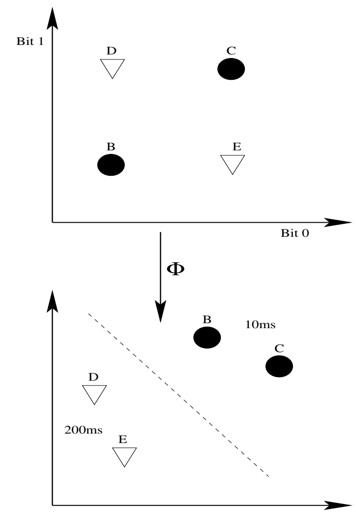

Figure 1: Separating XOR inputs. Four IP addresses belong to two distinct latency classes, circles and triangles. Although the points are not separable in the input space, a mapping Φ to a three-dimensional space gives a separating hyper-plane.

Let the input to the learning algorithm be pairs of d dimensional vectors $\mathbf{x} \in \mathbb{R}^d$ and corresponding y labels: $(\mathbf{x}, y)$. For latency prediction, the input vector is an IP address $\mathbf{x} \in \{0,1\}^{32}$ and $y \in \mathbb{R}$ is a latency. A kernel Φ transforms the IP address $\mathbf{x}$ in the *input space* into a higher dimensional *feature space*. SVM kernels allow us to find the separating hyper-plane in the feature space without carrying out the actual transformation in the input space.

As an instructive example, consider an agent A interacting with nodes B, C, D and E. Figure 1 plots the four nodes by their two most significant IP bits to provide a graphical intuition of SVM kernels. A's latency to B and C is 10ms, while D and E are 200ms away. We depict 10ms nodes with circles and 200ms nodes with triangles. Clearly, no linear separator exists in the input space. However, as shown in the lower half of the Figure, the feature space defined by the kernel transformation Φ yields a separator.

SVM regression [17, 11] is a similarly-posed optimization problem. Rather than predicting class labels, the machine estimates $y \in \mathbb{R}$. We use *mySVM* [16], an SVM regression package for the results in this paper.

2.2 Data Set

To collect data for our experiments, we use a simple active measurement procedure. We select unsigned 32-bit integers at random until one is found as a valid IP address in a public global routing table. Based on the approximately 1.8B publicly advertised addresses, filtering with the BGP table reduces our search space by approximately half. If the randomly selected destination responds to ICMP echo requests, i.e. "ping," we record the average of five ping times from our measurement host as the round-trip latency. Our data set consists of approximately 30,000 randomly selected $(IP, Latency)$ pairs. The data is available as a public resource from: `http://ana.csail.mit.edu/latency`

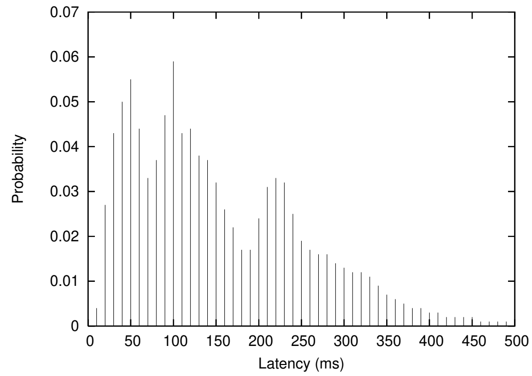

Figure 2: Probability mass function of 30,000 node latencies in data set.

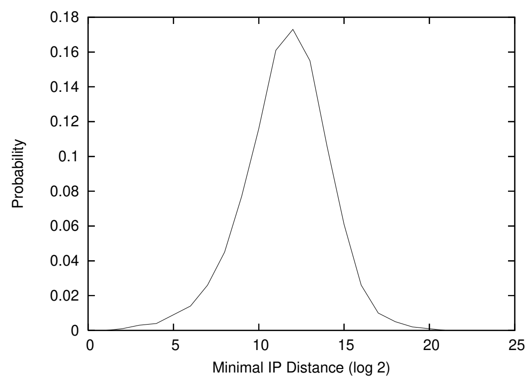

Figure 3: Probability mass function of IP address dispersion in data set.

Figure 2 displays the probability mass function of latencies as observed in our data set. The distribution is non-trivial, with multiple modes and a long tail, posing a reasonable challenge to a learning algorithm.

The selection of training points is crucial to any learning algorithm. 30,000 addresses out of the approximately 1.8B advertised in the global routing table is quite sparse. If our test points are close to points in the training set, we expect the learning to over-perform. We wish to ensure that our training set is suitably well-distributed in order to generalize to random predictions. One metric of distribution is address dispersion. To compute address dispersion, we find the numeric difference between each address and the next closest address. For the set of 30,000 addresses $\{A\}$, the minimum dispersion of address i is:

$$min_i = \operatorname*{argmin}_j \left(|i-j|\right)\ \forall j \in \{A-i\}$$

Figure 3 shows the probability mass function of IP address dispersion in our data set, i.e. $\Pr(\lfloor \log_2 min_i \rfloor = x)\ \forall i \in \{A\}$. Approximately 82% of the addresses have a minimal separation of 2^{10} or greater.

3. RESULTS

In this Section, we use SVM regression for latency prediction. Learning algorithms require optimization along several dimensions. We begin by analyzing the training complexion: which features of the IP address provide the most discriminatory power and what size training set generalizes well. Given a suitable training set, we examine prediction error and error distribution.

3.1 SVM Regression

We use SVM regression to produce a latency prediction. IP addresses are simply unsigned 32-bit integers. We transform the IP addresses into a 32 dimension input space where each bit of the address corresponds to a dimension. Thus, the input to the SVM machine is an IP address bit vector $\mathbf{x}$ while the labels y are floating point round-trip latency numbers.

To reduce possible dependence on the choice of training set, all results we present are the average of five independent experiments. We randomly permute the order of the data set so that, after splitting, the training and test samples are different between experiments. In this way, we ensure the generality, a critical measure of the effectiveness of learning algorithms.

Performance is measured in terms of deviation from the true latency in the data set. Let $V(f(\cdot), y)$ be the real-valued loss function between the prediction $f(\cdot)$ and the true value y. We first establish a baseline against which to determine the effectiveness of the learning. The most naïve approach is to simply always predict the mean latency of the training samples: $f(\mathbf{x}) = \bar{y} = \frac{\sum y_i}{|y|}$. The mean latency of our data set is $\bar{y} = 122$ms. A mean prediction strategy with an absolute loss function, $V(f(\mathbf{x}), y) = |\bar{y} - y|$, yields a mean prediction error of approximately 70ms. Thus, results lower than 70ms metric indicate effective learning.

We experiment with linear, polynomial and Gaussian kernels and empirically obtain the best results with a fifth-degree polynomial. The remainder of our results are based on using a fifth-degree polynomial kernel.

3.2 Problem Dimension

The selection of training complexion is crucial to creating a machine that generalizes well and operates efficiently. We examine the informational content of each bit, or "feature," in the IP address. Let $\boldsymbol{\theta}$ be a feature vector where $\theta_i \in \mathbf{x}$. Intuitively, the most significant bits correspond to large networks and should provide the most discriminatory power. Here "most-significant features" correspond directly to BGP prefix masks, i.e. `192.160.0.0/12`. We run the regression SVM algorithm against our data set using 4000 points for training while varying the number of input features. For example, the first 12 features of IP address `192.168.1.1` is the bit vector $\boldsymbol{\theta} =$`110000001010`.

We plot the SVM prediction mean error as a function of the dimensionality of the input space in Figure 4. We see that four or fewer bits yields virtually no information; the mean error is nearly identical to that in a mean prediction strategy. However, with only four more bits of input address, the regression achieves a mean error around 33ms. The optimal number of most significant features is between 12 and 14, after which test error begins to increase. This increase in test error is symptomatic of over-training as the least significant bits of the IP address add no discretionary strength.

Next, we use greedy feature selection to determine which bits of the IP address are most valuable to the regression algorithm. Greedy selection simply finds, in succession, the

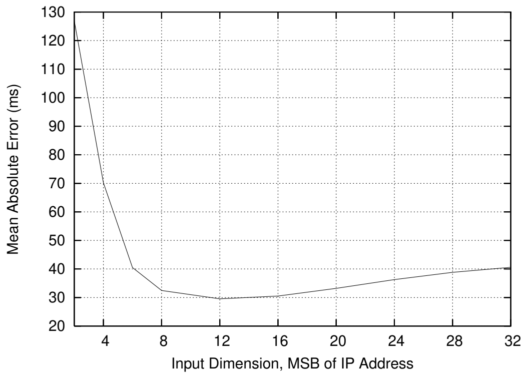

Figure 4: Latency prediction mean error vs. input dimension, i.e. number of most significant bits of input IP address.

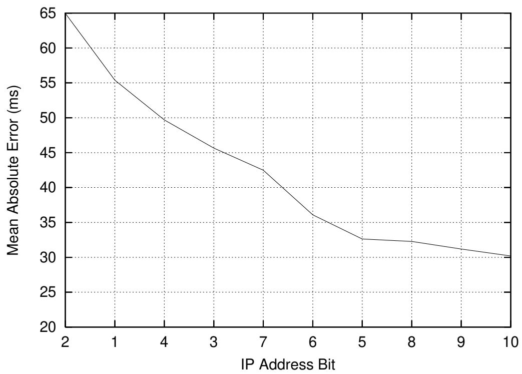

Figure 5: Feature selection: mean error vs. sequential bit chosen via greedy strategy.

next single feature that minimizes test error. For an error function $V(f(\boldsymbol{\theta}), \cdot)$, find the next feature θ_i from the remaining features not previously selected in $\theta_1, \ldots, \theta_{i-1}$. Thus, we evaluate each potential feature i for the fixed set of $i-1$ features. Formally:

$$\theta_i \leftarrow \underset{j}{\operatorname{argmin}}\, V(f(\boldsymbol{\theta}, x_j), y) \forall x_j \notin \theta_1, \ldots, \theta_{i-1}$$

Figure 5 depicts the prediction error as a function of features found in greedy feature selection. The x-axis shows which feature is selected; for example the first five best features are, in order, 2 1 4 3 7.

Thus, *the majority of the discriminatory power is comprised of the first eight bits of address*. This is a powerful result, but perhaps unsurprising given the traditional assignment of classful "net A" address blocks to large organizations and networks. Error continues to decrease to a minima around 12 bits after which the additional features begin to over-fit the training data and hinder the regression performance.

3.3 Training Size

Given the optimal features found via feature selection, we next attempt to optimize the balance between training and test size. Figure 6 shows the mean absolute error in milliseconds as a function of training size along with a 70ms line indicating the learning bound. Using 4000 of the data

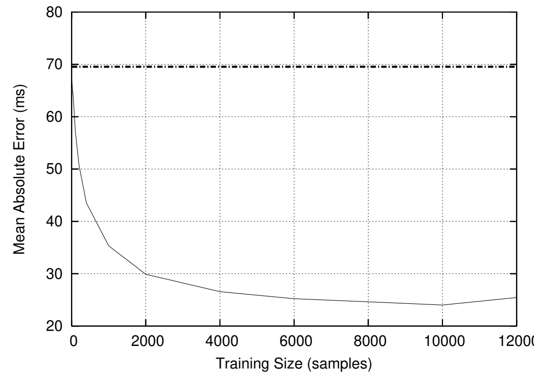

Figure 6: Latency prediction mean error vs. training size. The 70ms line represents a naïve mean prediction strategy.

points as training, we obtain an average error of 26.6ms across all latency predictions in the test set. 6000 training points yields 25ms average error. We select 6000 training points for the remainder of our experiments. Given the inherent noise in the network, the ability to predict latencies within 25ms of their actual value is readily acceptable for applications such as resource scheduling, service selection and user-directed routing.

A second metric of performance is to run SVM regression on a false data set identical in size to our real data, but with random features. In other words, for every measured latency, we assign a random IP address to create the false data set. Regression on the false data set with the same kernel parameters and training size yields a mean absolute error of 84ms, again lending credence to our original hypothesis: it is possible to leverage the structure of the network in order to form latency predictions.

3.4 Regression Performance

Using the first 12 features, i.e. bits, from 6000 of our samples for training yields a good balance between performance and exploitation. Given this training set, we examine the distribution of latency prediction errors. While the previous Section demonstrates a mean error of 25ms, it is important to understand the character of the errors. Figure 7 gives a scatter plot of measured versus predicted latencies. Perfect predictions will fall along a line at a 45 degree angle from the axes, while poor predictions are points that strongly deviate from the line.

Figure 8 shows the cumulative probability distribution of prediction error in milliseconds. 11% of the predictions are within 2ms of the correct value, while more than 80% are within 40ms of our measured latency. The distribution has a long tail however, indicating that while the majority of predictions are quite close, there is a relatively even distribution of infrequent errors greater than 60ms.

Finally, figure 9 presents the cumulative distribution of ratios between predicted and measured latencies. Ratios less than one indicate an underestimate of latency while those larger than one indicate an overestimate. With our SVM prediction method, approximately 61% of the estimates are within a 20% error, i.e. with ratios between 0.8 and 1.2, while approximately three-quarters of the predictions are within 30% of the actual value.

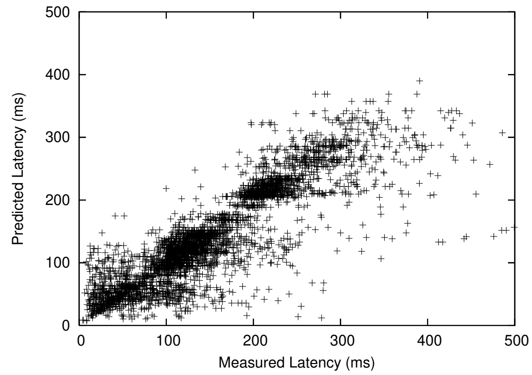

Figure 7: Scatter plot of measured vs. predicted latencies.

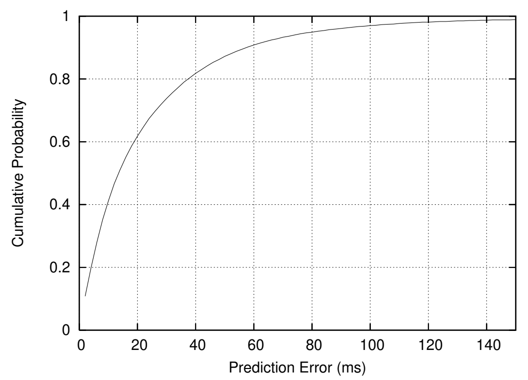

Figure 8: Cumulative distribution function of latency prediction errors given 6000 training samples.

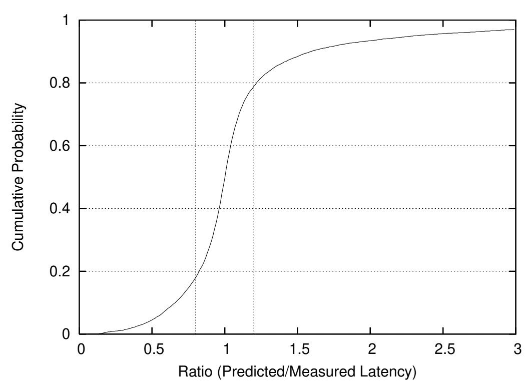

Figure 9: Latency estimation performance: cumulative distribution of prediction ratios.

3.5 Prediction Speed

Latency predictions must be fast in order to be practical for the wide-range of potential applications. Since we apply known SVM methods in a novel arena, we present empirical performance results rather than computational complexity measures. Joachims [11] contains an analysis of several techniques to reduce the time and space complexity of the quadratic programming required in SVMs.

A significant cost is collecting the training points, however several points bear notice. First, the host may continually collect training data through its normal interaction with the network, for instance using latency from the TCP three-way handshake. In this mode of operation, the data are no longer random and hence the model may be overly specific to IP addresses within networks the host typically interacts with. Yet, this specificity is a desirable feature for most hosts and servers. Because SVMs are amenable to online learning, the model adapts to provide the best generality and performance given the workload the host expects to encounter.

Second, it is reasonable for the host to periodically collect new, random training data. For instance, the host can spend weekend or night time hours collecting data and updating its model. The frequency of these updates and the performance degradation of the model without updating is a subject for future research.

On a 1.66GHz 32-bit Intel x86 architecture, training and creating the machine using 6,000 data points requires 27.5s, or ∼4.6ms per data point. Once trained, the prediction of latency to an arbitrary IP address takes approximately 1.5ms. Thus, the SVM regression is fast and practical for a wide-variety of agents and applications.

4. APPLICATIONS

Prediction of network latencies to random destinations is potentially useful in a variety of practical applications which motivate this research:

1. Service Selection: To balance load and optimize performance, a resource may be distributed over a set of geographically distributed servers. These servers may coordinate to form service selection decisions on the basis of current network performance and the origin of a request as well as the object requested [15]. Service selection is also an important problem in peer-to-peer (P2P) networks where popular data is replicated among many nodes. A search in a P2P file sharing network may result in many potential peers offering the file. Latency prediction enables an alternate architecture where intelligence is shifted to the end-nodes. For instance, consider a web service existing in multiple, distributed locations advertised via a set of DNS address records. An intelligent resolver agent's first choice for the given resource can be guided by our learning algorithm. The client predicts which server is closest from among the set of all potential addresses for the given resource. Note that an incorrect prediction is not fatal; nothing precludes the agent from selecting a different server if the first proves to be a poor choice. Both clients and servers benefit in such an architecture without explicit coordination.

2. User-directed Routing: Currently network end-nodes have no control over the route their data takes through the network to a destination. However, the continued adoption of IPv6, with multiple per-provider logical interfaces, and research efforts such as NIRA [20] and RON [1], are poised to give nodes coarse routing control. In an IPv6 world with its provider-assigned addressing model, hosts will have a combinatorial number of interfaces. When forming decisions on how to best send traffic to a particular destination, learning algorithms can significantly narrow the host's search space. Similarly, a mobile device choosing from many different possible wireless networks could form decisions based on prior interactions with each. In fact, any agent can build a "routing table" without formal participation in a routing protocol or receiving routing announcements.

3. Resource Scheduling: Web-servers endowed with predictive abilities might tailor content depending on the anticipated latency of the remote end-point or perform opportunistic scheduling [2]. Additionally, the grid computing community would like to predict transfer times in order to perform distributed scheduling efficiently [14].

4. Network Inference: Researchers frequently use structural models of the Internet including routing tables. However, publicly available routing tables [13] provide only a highly aggregated view and from limited vantage points. In many cases, it would be useful to understand the internal structure and address assignments of individual networks. A classification algorithm such as we propose could be used to infer detailed topological properties of networks.

5. CONCLUSIONS AND FUTURE WORK

An emerging architectural tussle is where in the network, if at all, intelligence should be placed [3]. Increasingly, end-nodes must be intelligent participants in the network through learning, prediction and classification. While a complete view of the network may be best harnessed by a distributed knowledge plane [5], we find significant value also to end-nodes acting as autonomous learning agents.

Our results are the first to examine latency prediction from machine learning on the IP address space. Using SVM regression, our results show an estimation performance within 30% of the true value for approximately three-quarters of the latency predictions on a large, live Internet data set. The results produced by our method are encouraging to enable a range of applications including service selection, user-directed routing, resource scheduling and network inference.

We plan to continue the research by investigating alternate feature geometries in order to use linear kernels and better represent the IP address structure. We wish to explore other network applications of SVMs and construction of autonomous agents on network test beds. These agents will model our vision of how a web server, peer-to-peer node or IPv6 host might act intelligently. An important question for future research is how network topology changes over time affect performance and thus how often the model must be retrained. Finally, we wish to investigate on-line learning as a means for agents to adapt to a changing network environment without explicit retraining.

Acknowledgments

We thank Mike Afergan, Steve Bauer, Srikanth Kandula and our reviewers for their invaluable comments and suggestions.

6. REFERENCES

[1] D. Andersen, H. Balakrishnan, M. Kaashoek, and R. Morris. Resilient overlay networks. In *Symposium on Operating System Principles*, Oct. 2001.

[2] M. Arlitt, B. Krishnamurthy, and J. C. Mogul. Predicting short-transfer latency from TCP arcana: A trace-based validation. In *Proceedings of Internet Measurement Conference*, 2005.

[3] M. S. Blumenthal and D. D. Clark. Rethinking the design of the internet: the end-to-end arguments vs. the brave new world. *ACM Transactions on Internet Technology*, 1(1), Aug. 2001.

[4] T. Bu, L. Gao, and D. Towsley. On characterizing BGP routing table growth. *Computer Networks*, 45(1):45–54, 2004.

[5] D. D. Clark, C. Partridge, C. Ramming, and J. Wroclawski. A knowledge plane for the internet. In *Proceedings of ACM SIGCOMM*, Aug. 2003.

[6] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. Vivaldi: A decentralized network coordinate system. In *Proceedings of SIGCOMM*, Aug. 2004.

[7] M. Freedman, M. Vutukuru, N. Feamster, and H. Balakrishnan. Geographic locality of ip prefixes. In *Proceedings of ACM Internet Measurement Conference*, Oct. 2005.

[8] K. Gummadi, S. Saroiu, and S. Gribble. King: Estimating latency between arbitrary internet end hosts. In *Internet Measurement Workshop*, Nov. 2002.

[9] K. Hubbard, M. Kosters, D. Conrad, D. Karrenberg, and J. Postel. Internet Registry IP Allocation Guidelines. RFC 2050, Nov. 1996.

[10] G. Huston. BGP reports. `http://bgp.potaroo.net`.

[11] T. Joachims. Making large-scale SVM learning practical. In B. Schlkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*. MIT Press, 1999.

[12] X. Meng, Z. Xu, B. Zhang, G. Huston, S. Lu, and L. Zhang. IPv4 address allocation and the BGP routing table evolution. *ACM SIGCOMM CCR*, 35(1):71–80, 2005.

[13] D. Meyer. University of Oregon RouteViews, 2000. `http://www.routeviews.org`.

[14] Y. Qiao, J. Skicewicz, and P. Dinda. An empirical study of the multiscale predictability of network traffic. In *Proceedings of IEEE HPDC*, 2003.

[15] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings of IEEE INFOCOM*, June 2002.

[16] S. Rüping. mySVM. `http://www-ai.cs.uni-dortmund.de/SOFTWARE/MYSVM/`.

[17] V. Vapnik, S. Golowich, and A. Smola. Support vector method for function approximation, regression estimation and signal processing, 1996.

[18] V. N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, 1995.

[19] B. Wong, A. Slivkins, and E. G. Sirer. Meridian: A lightweight network location service without virtual coordinates. In *Proceedings of SIGCOMM*, Aug. 2005.

[20] X. Yang. NIRA: A new internet routing architecture. In *Proceedings of ACM SIGCOMM Workshop on Future Directions in Network Architecture*, Aug. 2003.