Statistical Learning of ISP Peering Policies

Michael Cafarella and Daniel Lowd
Department of Computer Science and Engineering
University of Washington
Seattle, WA 98195 U.S.A.
mjc@cs.washington.edu, lowd@cs.washington.edu

December 10, 2004

Abstract

Spring et al's research on path inflation uses a large number of traceroute probes to deduce facts about ISP policies [3]. They are able to learn an ISP's network topology, its individual router policies, and the overall ISP peering strategy. However, the authors use only very basic heuristics to make deductions from the data. Also, collecting the necessary traceroutes is a substantial burden. Finally, a relationship between two ISPs is characterized only as being in one of several categories.

We use a number of different machine learning techniques to make similar deductions. We believe these techniques are well-grounded in statistics. They enable us to describe a peering relationship with much greater precision than the commonly-considered ISP policies.

1 Introduction

Many of the inner workings of the internet are unknown and difficult to determine. While the basic protocols are standard, network topology and routing policies are manually set by individual ISPs and are generally considered proprietary. This is the natural outcome of allowing distributed management, a policy that has greatly helped the internet to grow. Unfortunately, this makes studying the internet, debugging internet problems, and planning for its future more difficult.

In the Rocketfuel project, Spring et al. [4] used traceroute sampling to map ISPs' internal networks. They used Planetlab sites to launch a number of traceroutes to various points on the internet. They recorded the routers that each trace traversed, and used this information to generate a topology for the network. They use a number of heuristic techniques to tie router addresses and DNS names to geographic real-world coordinates. Thus Rocketfuel can generate a geography-fitted network topology for the traceroute-instrumented ISPs.

In [3], Spring et al. study the problem of *path inflation*. Network path inflation can happen in several scenarios, but one of the most common is during inter-ISP traffic exchange. There is no protocol for ISPs to easily share information about their internal topologies. Further, ISPs may have different business motivations for exchanging traffic at points that are less than geographically optimal. By comparing observed traffic handoffs in traceroute data with the geographically-extracted information from the Rocketfuel paper, the authors can compute each ISP's traffic handling policies.

While Rocketfuel's geographic-tying techniques have yielded good results, they risk being overly brittle or simplistic, since they rely on hand-coded knowledge and manual experimentation. They also rely on tremendous amounts of data. Finally, inter-ISP peering policies are

characterized as being just one of *early-exit*, *late-exit*, *load-balancing*, or a few subclasses of *late-exit*. We would like to describe these policies in greater detail.

We take the goals of [4, 3] as our own, but try to apply modern machine learning techniques. We focus on the problem of predicting inter-ISP peering policy. Specifically, given source and destination IP addresses and some knowledge of the network, how can we predict which peering point packets are most likely to take? Each ISP's peering decision is modelled as a probability distribution over the border routers between itself and the destination ISP.

It might seem strange to model the largely deterministic phenomenon of routing policy probabilistically. However, there are solid reasons for doing so:

- Probabilistic statements can model uncertainty on the predictor's part, not just probabilistic behavior of the modelled process. Operating with small amounts of data means that the learner will need to generalize from limited training; it's misleading to pretend such generalization will be correct 100% of the time.
- An ISP is an agglomeration of policies, at the router and at the peering ISP level. Probabilities are useful as a shorthand for describing the total impact of those policies.
- Although we assume most peering policies are deterministic, this isn't guaranteed.
- Probabilistic modeling can be combined with a cost model to enable utility-based policy analysis.

2 Problem Modelling

In this section, we frame the specific learning problem. Given a source IP address and a destination IP address in two different ISP networks, we wish to predict the peering point used, where traffic goes from one network to the other. Training data for this problem consists of a set of traceroutes covering a number of paths traversing these same two networks, but none from the specific source to the specific destination. (Clearly, these traceroutes must also reveal enough of the topology to make the problem feasible: if, for example, the learning algorithm has never seen the peering point that is used, then it will not be able to predict it.)

2.1 Data processing

In general, we can assume that each traceroute contains the following information along the observed route:[1]

1. source IP in ISP A
2. some number of routers in ISP A
3. peer router in ISP A
4. peer router in ISP B
5. some number of routers in ISP B
6. destination IP in ISP B

Each step in the traceroute is labelled with the RTT from the source.

The two peers in the middle constitute the peering point, where traffic is exchanged between ISPs A and B. We seek to learn the probability distribution

$$Pr(peer|\text{source-in-A}, \text{dest-in-B})$$

As discussed before, a substantial advantage of learning a probability distribution is that it

[1] At least, this is the form we would like the data to take. In fact, the traceroute data is very messy. Formats vary, routers are unavailable, some RTTs are unavailable, there is not always a DNS name for every IP address encountered, and there are many other problems, as well. Processing this data required large amounts of time and programming, but we were eventually able to massage our huge set of traceroutes into the format described here.

allows us to quantify our uncertainty. We can generate the most likely prediction by selecting the peering pair with highest probability, given the end-points.

After finding the most popular ISP pairs in a set of traceroutes, we use this algorithm to infer router policies:

1. Split the data into sets S and S'.

2. With set S, build a model of the network graph in the ISPs in question. Adjacent steps in the traceroute indicate connected machines. We average all RTTs for a given linked machine pair, and label the link with that value.

3. For each traceroute in set S', generate a set of tuples that consist of:

   - Number of border routers between the source and the destination ISP. (Draw this information from the graph constructed using set S.)
   - The index for the border router actually chosen by the current traceroute.
   - For each border router,
     - Shortest-path RTT from source to the relevant border router. Shortest path is computed over the set S graph.
     - Shortest-path RTT from the relevant border router to the destination. Shortest path is computed similarly.

We then feed this tuple set to the learner, described in the next section. The learner will compute the probability of the source ISP choosing each of the possible border routers in order to get the packet to the destination ISP.

It may seem slightly odd to build an explicit model of the ISP router graph topology before using a more statistical technique with different data. However, we think there is a good justification for this. This allows us to focus on the special problem of modelling policies given topological data, while avoiding using topological data from the test sets. Since the learner is not supposed to know the eventual peering decision for a given route in the test set, it is important that we do not peer behind the training set veil and also use it for topology. We do not use the constructed graph as a mechanism for observing router decisions or for gathering statistics on peering decisions. It exists only as a means for generating the fairly limited data for the learning step.

Ideally, we would prefer to avoid constructing the graph altogether by using geography in place of latency. It is used to compute the set of possible peering points, and to compute the shortest-path RTT distance from the source to those points. If we were given the set of possible peering points, and if we had geographic data about the source and peer points, then we could drop the graph construction altogether. Indeed, since data that we use for building this graph is data that we cannot use for later learning, we would like to build the topology with as little data as possible. This is an issue we could explore more in future work.

Since not every possible source/destination pair is reflected in the chosen set S, there will be lots of sources or destinations that do not appear in the generated graph. So, we do not actually compute the shortest-path distance between the true source and destination. Instead, we compute the distance between the ISP routers that are immediately adjacent to those machines. In practice, the resulting distance measures are very similar, if not identical. Since even knowing a single hop of the chosen route could be considered cheating, it would be even better to remove this assumption using routing tables or subnet guesses.

There are other pieces of information that could be useful for the learner, such as geographic distances. However, we concentrate exclusively on measured RTT values.

Some related problems we ignore for now include inferring the complete path from source to destination, traversing multiple ISPs, and learning a collection of ISP routing policies at once. The complete path problem in the two-ISP case can be decomposed into predicting the probability of each peering point and then predicting the probability of each internal ISP path. Learning internal ISP policies has already been done with some effectiveness; there may be interesting problems, but we do not focus on them here. Traversing multiple ISPs is harder, since the policies of multiple ISPs may be relevant at once.

Learning a collection of two-ISP routing policies is also an interesting problem: since some ISP policies are not wholly independent, knowledge of a few ISP policies can make learning additional ISP policies more effective. Some of these relationships can probably be captured by hierarchical Bayesian methods, which link a collection of independent but related models via a prior distribution over their parameters.

2.2 Learning Task Formulation

As stated above, we need to compute the probability of the source ISP choosing each of the possible peering points to the destination ISP. We formulate the probability for a router R_i as follows:

$$Pr(R_i|src, dest) \propto e^{\lambda_1 RTT(src,R_i)+\lambda_2 RTT(R_i,dest)}$$

We can turn this into a true probability by normalizing over all the border routers for a given router decision:

$$Pr(R_i|src, dest) = \frac{e^{\lambda_1 RTT(src,R_i)+\lambda_2 RTT(R_i,dest)}}{\sum_j e^{\lambda_1 RTT(src,R_j)+\lambda_2 RTT(R_j,dest)}}$$

The router R_i with the highest probability is the one predicted to be the source ISP's chosen peering point.

Except for the values λ_1 and λ_2, the precomputed router connectivity graph gives us all the information necessary to compute $Pr(R_i|src, dest)$. The learner's main task is to choose these λ values. We choose them such that we maximize the probability of seeing the test tuple data. In statistical literature this is called *maximum likelihood estimation*.

We can also interpret these λ values as weights that determine the relative cost to the routing ISP of packet latency in the source ISP versus packet latency in the destination ISP. We can very naturally express the peering policies from [3] in terms of these λ values.

- A *random* policy is followed when $\lambda_1 = \lambda_2 = 0$. No latency times are considered at all, and all R_i are equal.
- A fully *early-exit* policy is followed when $\lambda_1 \leq 0; \lambda_2 = 0$. The packet latency in the destination ISP is not considered at all when the source ISP determines the peering point.
- A fully *late-exit* policy is followed when $\lambda_1 = 0; \lambda_2 \leq 0$. The packet latency in the source ISP is not considered at all when the source ISP determines the peering point.
- An *optimal* policy is followed when $\lambda_1 = \lambda_2 = w; w \geq 0$. Latency in both ISPs is considered with equal weight.

We want our learner to predict routing policies with greater accuracy than any of the above "stereotypical" policies. Put another way, when we use the λ values from our learner to evaluate $Pr(R_i|src, dest)$, the resulting probabilities should be more accurate than when we use λ values from *random*, *early-exit*, or the other enumerated policies. If our learner succeeds, then we can say our system describes an ISP's policies with more precision than the standard methods do.

But if the learner's λ values cannot beat those of the standard methods, then none of our goals for a statistical approach can be fulfilled. We will neither describe the system in greater detail, and smaller amounts of traceroute data will likely make performance even worse.

index	Source ISP	Destination ISP
1	francetelecom.net	opentransit.net
2	alter.net	level3.net
3	ebone.net	sprintlink.net
4	above.net	level3.net
5	att.net	level3.net
6	gblx.net	level3.net
7	qwest.net	level3.net
8	bbnplanet.net	level3.net
9	oceanic.net	netenterprise.net
10	bbnplanet.net	att.net
11	ebone.net	level3.net

Table 1: Labels for each of the ISP pairs, referred to by ID for most of the paper.

3 Experiments

We now present prediction results from a number of different methods for learning the λ values. We also compare them against the standard policies listed above. Although finding the right learner took some time, we were eventually able to predict peering decisions with reasonably high accuracy.

3.1 Methodology

We ran all experiments using data from the 10 most popular ISP pairs seen in our dataset. For each of the pairs, we regenerate the graph used for topology and distance.

We present both log-likelihood and accuracy results for every experiment. Accuracy is just a measure of how often the policy was able to predict the router that the ISP actually chose. It is expressed as percentage of the routing decisions seen in the data. When the policy emits router probabilities that result in a tie for the most-likely, the prediction is scored with fractional credit (depending on how many routers were tied).

Log-likelihood is a measure of how likely the dataset is, given the computed λ values. This shows how well a policy predicts the probabilities for all possible peering decisions, not just the one actually chosen. The raw numbers do not have an easily-understood intuition, but larger numbers are better.

All λ values are learned using gradient-ascent, to maximize the log-likelihood of the training data. Since the log likelihood of our training data is a convex function, there is a single, global maximum. This guarantees that gradient ascent will not get stuck in a local maximum, but will converge to the optimal weights. We used the implementation of Powell's method from [1] to perform the actual optimization.

All results are also compared against the stereotypical policies enumerated in the previous section.

Table 1 lists the ISP pairs we studied. Later tables will refer to them only by ID.

Table 2 shows some basic statistics about the data. Each ISP pair's dataset had between 200 and 600 routes to process. We divide the data into train and test sets using 10-way cross-validation; that is, the available data was divided into training and test sets 10 times, each time with a different tenth of the dataset used for testing. All results come from averaging modeled probabilities over these 10 different train/test sets.

Table 3 describes some of the ambiguity found in the data. ISPs will sometimes make different routing decisions even though all the RTT distances are exactly the same. It's not possible to say whether such conflicts mean that the ISP is actually making different decisions based on identical inputs, or whether the data we have selected to learn from simply does not capture every factor the ISP incorporates. From the learner's perspective, the ISP seems to be routing at least somewhat randomly. This random behavior places a ceiling on how well any learner can be expected to perform.

ISP Pair	Total routes	Unique RTT sets	Unique routes
1	217	17	24
2	445	199	237
3	571	115	115
4	524	127	173
5	518	190	190
6	508	170	180
7	506	263	265
8	409	190	193
9	366	323	324
10	391	250	283

Table 2: Statistics about the processed dataset. Two traceroutes produce indistinguishable "RTT sets" when they have identical source-to-peer and peer-to-destination RTT times. Two traceroutes produce identical routes when they produce identical RTT sets and choose the same peer. There are more unique routes than unique RTT sets, indicating that identical RTT values can result in choosing different peers.

ISP Pair	Ambiguous routes	Max log-likelihood	Max accuracy
1	76.49%	-0.5238	65.43%
2	42.69%	-0.5869	74.60%
3	0.00%	0.0000	100.00%
4	78.24%	-1.4519	41.03%
5	0.00%	0.0000	100.00%
6	55.90%	-0.1245	95.07%
7	22.33%	-0.0197	99.60%
8	1.95%	-0.0122	99.26%
9	0.81%	-0.0052	99.72%
10	39.64%	-0.2582	84.65%

Table 3: Ambiguity in the dataset means that routing decisions with identical RTT-distances may be route differently. This ambiguity may reflect that even a complete RTT "fingerprint" is not sufficient to characterize routing decisions. These conflicts place a theoretical ceiling on the predictive power of any learner.

3.2 Basic Model

We first tried to learn λ values using the traceroute data tuples described exactly as in Section 2. Recall that each routing decision described by a traceroute is presented as a set of RTT pairs. There is an RTT pair for each of the possible peering points between the source and destination ISPs. Each peering point's RTT pair has the shortest-path RTT distance from the source to the peering point, and the shortest-path RTT distance from the peering point to the destination.

Accuracy results for the basic model are presented in Table 4. Our learner predicts actual ISP behavior better than the other policies in six out of ten scenarios. It does especially poorly for ISP pairs 4, 6, and 8.

Note that in the case of 4, at least, the dataset ambiguity places an unusually restrictive ceiling on our prediction accuracy.

Note that in Table 5, ISPs where the basic model is the best predictor and still relatively poor, the log-likelihood is relatively low. ISP 2 is the best example of this phenomenon.

3.3 Basic Model with Router Weights

Our performance with the basic model was somewhat mediocre, so we ran a short experiment. We threw out the weighted pair of distances, and tried a simple-minded predictor.

We formerly computed the probability of choosing a peering point as a λ-weighted function of the RTT pairs. Instead, we tried to compute that probability as the simple percentage of the routes that go through a peering point of the total number of routes (that are used for training). (This is known as the marginal probability for the peering point.) So we now use:

$$Pr(R_i) = \frac{\#\text{ training routes using } R_i}{\#\text{ training routes}}$$

Amazingly, this was a much better predictor than our basic model above. For example, the most accurate policy for ISP 6 was the *random* policy (see Table 6), which scored 7.69%. The marginal model obtains accuracy of 74.6% simply by predicting the most common peer. In the log-likelihood results, this model wins over the basic model or the best stereotypical model in 8 of 10 ISP pairs.

It was exciting to see a big jump in performance, but cast our basic model in a terrible light. We did better by ignoring source and destination altogether. A very dumb strategy of simply choosing the most popular peering point could beat our learned weights.

3.4 Extended Model

It seemed obvious that we could try extending our basic model with some of the per-peering point information that the marginal approach used. So we extended the basic model very simply. We add a new λ value for each peering point being considered. We now use the formula:

$$Pr(R_i|src, dest) = \frac{e^{\lambda_1 RTT(src,R_i)+\lambda_2 RTT(R_i,dest)+\lambda_{R_i}}}{\sum_j e^{\lambda_1 RTT(src,R_j)+\lambda_2 RTT(R_j,dest)+\lambda_{R_j}}}$$

We used to consider an ISP's peering policy as a simple pair of weights. We can still consider that same pair as setting general policy, but now there is also a special set of "modifying weights", to favor or disfavor certain routers outside of the overall policy. This might better reflect the engineered routes that we believe ISPs often create.

This change allowed for substantially better performance, as can be seen in Table 6 and Table 7. In fact, some of these results come fairly close to the theoretical maximum performance described in Table 3.

4 Conclusion

Learning peering policies is a difficult task overall. Peering policies can be very complex, or arbitrary. There are many factors that go into a

ISP Pair	basic model	early	late	opt	rand
1	**40.09%**	0.00%	**40.09%**	0.00%	8.33%
2	**11.14%**	9.04%	7.82%	10.94%	2.43%
3	**96.49%**	96.23%	48.24%	**96.49%**	50.00%
4	0.19%	0.49%	1.68%	1.62%	**2.63%**
5	**99.71%**	69.98%	60.23%	75.48%	33.33%
6	2.36%	0.00%	6.03%	0.00%	**7.69%**
7	43.37%	**43.77%**	27.3%	41.69%	16.66%
8	6.84%	10.75%	**30.56%**	4.27%	16.66%
9	**57.37%**	**57.37%**	54.37%	**57.37%**	20.00%
10	**44.37%**	21.86%	32.07%	3.32%	8.33%

Table 4: Accuracy results for the basic model. The best result for each ISP is in **bold type**.

ISP Pair	basic model	early	late	opt	rand
1	**-1.9712**	-2.3128	-1.9719	-2.3614	-2.3756
2	**-3.5757**	-3.5774	-3.6282	-3.6118	-3.6319
3	-0.1324	-0.0857	-0.6812	**-0.0833**	-0.6812
4	**-3.3818**	-3.6344	-3.5694	-3.5694	-3.5694
5	**-0.0051**	-0.4082	-0.8817	-0.3204	-1.0778
6	**-2.1889**	-2.4474	-2.2076	-2.5154	-2.5154
7	-0.9368	**-0.9362**	-1.4393	-1.0863	-1.7570
8	**-1.5280**	-1.5341	-1.7490	-1.5349	-1.7490
9	-1.2699	-1.2723	-1.2885	**-1.2687**	-1.5666
10	**-1.2344**	-1.9229	-1.5084	-2.3799	-2.4229

Table 5: Log-likelihood results for the basic model. Larger numbers are better. The best result for each ISP is in **bold type**.

ISP Pair	extended model	basic model	marginal model	best-stereotypical
1	**40.09%**	**40.09%**	39.62%	**40.09%**
2	**33.48%**	11.14%	15.27%	10.94%
3	**100.00%**	96.49%	99.64%	96.49%
4	**37.52%**	0.19%	11.82%	2.63%
5	**100.00%**	99.71%	60.03%	75.48%
6	**90.15%**	2.36%	74.60%	7.69%
7	**96.83%**	43.37%	53.55%	43.77%
8	**92.66%**	6.84%	56.23%	30.56%
9	**72.67%**	57.37%	57.37%	57.37%
10	46.29%	44.37%	**49.10%**	32.07%

Table 6: Accuracy results for the extended model. The best-stereotypical column lists the highest accuracy of either *early*, *late*, *opt*, or *rand* from Table 4. The best result for each ISP is in **bold type**.

ISP Pair	extended	basic	marg	non-marg
1	**-1.1588**	-1.9712	-1.2856	-1.9719
2	**-1.6687**	-3.5757	-2.4446	-3.5774
3	**0.0000**	-0.1324	-0.0209	-0.0833
4	-2.6094	-3.3818	**-2.4541**	-3.5694
5	**0.0000**	-0.0051	-0.7215	-0.3204
6	**-0.4501**	-2.1889	-1.0562	-2.2076
7	**-0.1267**	-0.9368	-1.0564	-0.9362
8	**-0.3013**	-1.5280	-1.1625	-1.5341
9	**-0.9959**	-1.2699	-1.2365	-1.2687
10	-1.1333	-1.2344	**-1.0456**	-1.5084

Table 7: Log-likelihood results for the extended model. The best-stereotypical column lists the highest log-likelihood of either *early*, *late*, *opt*, or *rand* from Table 5. The best result for each ISP is in **bold type**.

peering decision, including many that are unlikely to be captured in any dataset gathered through exclusively traceroute-based means.

However, our very simple weight-based model is able to solve a large amount of the problem. They capture most of the top-ten ISP relationships more accurately than a number of stereotypical policy descriptions. This suggests a machine learning approach is worth also pursuing when the dataset is more impoverished. While standard techniques might be very brittle in the face of small amounts of data, statistical ones often are not.

We also learn that policies depend on much more than RTTs: certain routers may be preferred or dispreferred independent of their respective latencies. Although we knew some engineered routes add substantial latency, we were surprised by the extent to which this affected our modelling accuracy and log likelihoods. We assumed that network structure would account for most of the learnable patterns.

After taking router preference into account, our learned models bested every stereotypical baseline, as well as the marginal model, except for one ISP pair. For two of the datasets, we predict the test set perfectly; for several others, we come close to the theoretical upper bounds. This suggests that weight-based approaches using RTT and router weights can effectively model peering behavior in many cases. Simple analysis of the model weights can also yield an understanding of the tradeoffs being made by the peering policy, the extent to which latency in source and destination ISP are considered, as well as which routers are preferred.

5 Future Work

Our top priority for future work is to employ a relational learning technique. A relational technique would allow us to model the relationships between all routers in the system, and infer routing policy both at the router level and the ISP level.

While traditional machine learning operates on a fixed set of features that define each instance, relational methods use relations and first-order logic. Consider the task of predicting whether or not someone smokes. If people who smoke tend to be friends with other people who smoke, then one's friends may be a good predictor of this value. In the traditional machine learning approach, we could use the variables *NumberOfSmokingFriends*, *NumberOfNonSmokingFriends*, and *Smokes*. By applying an appro-

priate algorithm, we could learn how the number of smoking and non-smoking friends affects the probability of an individual smoking.

A similar mechanism would probably work well with routing relationships. Each router's policies depends on global ISP policies, but also on local connectivity, geography, and link congestion. A different set of predicates is required, but the fundamental idea is the same. Of the competing probabilistic relational models in existence, we would recommend using Markov Logic Networks (MLNs), due to their versatility and scalability [2].

In addition, we would like to extend our learners with additional features. We could add information about geographical distance, link bandwidth, or even business details.

6 Acknowledgements

The authors would like to thank Pedro Domingos and Ratul Mahajan for their advice and help.

References

[1] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, 1993.

[2] Matt Richardson and Pedro Domingos. Markov logic networks: A unifying framework for statistical relational learning. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, pages 49–54. IMLS, 2004.

[3] Neil Spring, Ratul Mahajan, and Thomas Anderson. Quantifying the causes of path inflation. In *ACM SIGCOMM*, 2003.

[4] Neil Spring, Ratul Mahajan, David Wetherall, and Thomas Anderson. Measuring isp topologies with rocketfuel. *IEEE/ACM Transactions on Networking*, 12(1):2–16, 2004.