IP Geolocation with Two-tiered Neural Networks

Hao Jiang Jeanna N. Matthews
Clarkson University

Abstract

The ability to accurately determine the geographic location of an arbitrary IP address has potential in many applications. Previous methods that are based on observing the relationship between network delay and physical distance are inaccurate. Methods based on delay similarity are more accurate but inefficient because they require information on a large number of landmark nodes near the destination to be collected and maintained. We propose a method that can overcome both problems. Our method maintains a stable collection of network observers and landmark nodes that covers the target area. Observers are network nodes from which we can issue measurement command such as *ping* and *traceroute*. Landmarks are IP addresses which are reachable from observers and for which the physical locations are well-known. With measurement results collected from these landmarks, we train a two-tier neural network that estimates the geolocation of arbitrary IP addresses. The first tier neural network locates a smaller region in which the target IP resides, and then the second tier estimates its location more accurately within that region. We intentionally limit the size of landmark collection to a controllable and reasonable number, and show that with only a limited number of landmarks, we are able to retain a high accuracy that is similar to previous methods. In our experiment, with 1547 landmarks across US territory, the median error of our estimation is 4.1 km on the entire test dataset. On half of the test regions which contain more than 100 landmarks, the median error can be decreased to 3.7 km.

1 Introduction

The purpose of IP geolocation is to estimate the geographic location of a given IP address. Accurate IP geolocation has potential in many applications. For example, knowing the geographic location of a new client, a distributed system like Netflix and Amazon AWS can dispatch the requests to a nearest datacenter node and thus ensure a better service quality. Online Ad systems like Google AdWords can provide their viewers more accurate information about local discount activities. If an online credit card transaction happens in a region far away from the owner's usual region of activity, we can suspect the existence of credit card fraud and temporarily suspend the transaction for further inspection.

With assistance from client side, IP geolocation can reach high accuracy. For example, devices that implement W3C Geolocation API[19] will provide their location information retrieved from GPS, WiFi and other channels to webservers they access. If the target IP provides web services, we have more hint to geolocate it. Calder et al. [3] show their effort of geolocating Google web servers based on the servers' load balance. In addition, it helps to have physical access to the region where target IP resides in. Craig et al.[18] demonstrate using a mobile monitor to capture leaked WiFi signals from residences and use them to geolocate a given IP. However, requiring physical access or web service on the node in question greatly limits the applicability of these methods.

There has been substantial work in the more general problem of locating an arbitrary IP address using only network measurement methods such as *ping* and *traceroute*. Much of this previous work is based on the idea that a simple relationship exists between the physical distance of two network nodes and the round trip time between them. This assumption allows the problem to be described either as a regression problem or an optimization problem. Different mathematical models, such as quadratic programming[8], semi-definite programming [11], Bézier curves[23] and polynomial regression[5] have been used to solve the problem. The state-of-the-art result in this category, to the best of our knowledge, is given by [23] and [5], which achieve an estimation with

an median error of $\sim$ 20 miles. The accuracy of these methods suffers because especially in the wide area network case, a simple relationship between distance and latency may not exist. First, the network latency is affected by multiple factors such as cable types and count of routers between nodes. Also, considering the effort of maintenance, network cables connecting two cities may follow transportation routes instead of going in a straight line. Second, latitude and longitude forms a non-Euclidean coordinate system, while previous models simulate them using Euclidean. The impact may be small in a local area, but becomes significant in a wide area. Together, these facts fundamentally limit the accuracy of these delay model-based methods.

In [20], Wang et al. propose a novel method. They notice the inevitable estimation error in previous methods. Yet, in practice they find that the measured distance by these methods is in general proportional to the real distance. Thus instead of calculating the estimation directly, they compare the measured result from target with results from IP addresses with known locations called "landmark"s. More specifically, they first use constraint-based geolocation described in [8] to geolocate a large region in which the target IP address resides, then collect any facilities that host their websites locally in that region as landmarks. They measure these landmarks, find the landmark that has the closest measured distance to the target, and use the landmark's location as an estimation. This method gives the best result of any IP geolocation we have seen so far, with a median error of 690m - 2250m. Despite the achievement, this method is still not perfect. One problem is that the accuracy of this method highly depends on the density of landmarks collected in the target area. In a region where fewer landmarks can be collected, the system performance will degrade quickly. Landmark collection and verification is also tedious work and limits the performance of entire process. Though the landmarks can be cached, they may no longer be valid due to various reasons such as addresses changing. Excessive number of landmarks will slow down the update of measurement results and may limit its application in real world scenarios.

We gain two important insights from previous work. First, for IP addresses that are geographically adjacent, their measurement results from same observer should be similar. Hence we can collect measurement data from landmarks together with their locations, and use the data to train a model that describes the distribution of location conditioned on measurement results. We then use this model to estimate the location of an IP address from its measurement result. Second, we notice that in a small region (of radius 100-300 km), this estimation problem will be much simpler than it is in a wide area. For example, in a small area the earth surface can be seen as an Euclidean space. The IPs that are physically adjacent will more likely share the same ISP and use similar type of cables. In addition, based on our observation, it takes only a single hop for most subnets to reach another in such a small region. Thus the latency between them can be linear to their distance. Thus in a small region, using only local data to estimate an IP's location may lead to a better result than using the entire dataset.

With these insights, we design and implement a new IP geolocation approach. Our work consists of two parts: data collection and model training. We first collect landmarks from different data sources, then measure these landmarks from different observers. These measurement results, together with their known location in latitude and longitude, form our training data. We build our model as a two-tier neural network. Tier 1 is trained with entire dataset. It is responsible for estimating a region the target IP resides in. Tier 2 is trained with only the data in the region indicated by the tier 1 stage.

We collect datasets from three different sources - Ripe Atlas probes, universities websites and city government websites. Ripe Atlas probes reside primarily in residential networks and academic networks. University websites are in academic networks and city government websites use primarily commercial networks. This provides our dataset substantial diversity. Our dataset is far larger than most that are used in previous research, thus providing a more general and reliable result. In a US based test, with 1547 landmarks, we have a median error of 4.1 km.

Landmark density in a region is the primary factor that affects the accuracy of our method. We present an extensive numerical analysis showing how the performance varies with landmark density. This result provides our readers who want to adopt our method a reference of the size of landmarks they need to maintain for good performance. Moreover, we show that half of these landmarks have more than 100 landmarks within distance of 300 km. In these case, we can further decrease the median error to 3.7 km. While in regions with less than 50 landmarks, the median error increases to 6 km.

The rest of the paper is organized as follows. Section 2 is an introduction of our two-tiered network structure. Section 3 presents our data collection process. Section 4 describe our experimental results and analysis. Section 5 discusses possible improvements to the approach. Section 6 discusses related work.

2 Two-tiered Neural Network

Our system consists of two tiers, each of which contains a neural network trained with data collected from landmarks. Both tiers take measurement results from a target IP as input and output a latitude-longitude coordinate. The first tier is trained using all of our training data including Ripe Atlas probes, university websites and city government websites. It is expected to locate a large region that contains target IP. The second tier is then trained with only data from that region, to get the final estimate of target IP's geolocation. In this section, we describe the framework of training data generation, introduce some notations we will use in subsequent parts, and finally provide a detailed description of the design of each stage. We will introduce more detail about data collection in next section.

2.1 Training Data Generation

Assume that we have N landmarks, and k observers in the network. We send *ping* requests to each landmark from the observers, collect RTT as a vector $\mathbf{x}_n$ of size k. The landmark's physical location in latitude and longitude is recorded as vector t_n of size 2. Thus with the entire landmark set, we have an input matrix $\mathbf{X}$ of size $k \times N$, and a target matrix $\mathbf{t}$ of size $2 \times N$.

ping and *traceroute* are two of the most common used tools in Internet measurements. *ping* is simple and fast. *traceroute* provides more information about network path and hop-to-hop latency. In [16], Pelsser et al. discussed the potential problem using *ping* to do measurement. Despite this finding, we choose *ping* as our primary measurement method. We do this because it is straight forward to translate *ping* results to network input. While we are still working on looking for a valid representation of *traceroute* result as network input, we haven't yet found a satisfactory solution. This is also limited by the quota allocated by Ripe Atlas network we use, which allow us to do more *ping* measure than *traceroute*.

We adopt multiple methods to mitigate the impact of variance in our input data. First, by adding a regularization term to the error function, we make the network invariant to the variance in input. This is detailed in the next section. Second, our input data is designed to be multi-dimensional, hence variance in one dimension will be diminished in the final output. In practice, we didn't observe a big impact from the variance in *ping* results.

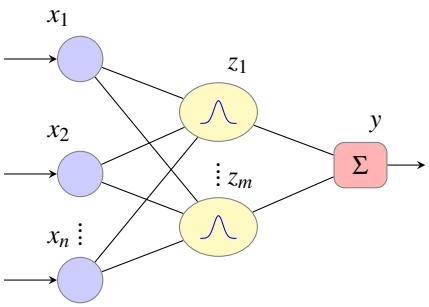

Figure 1: Radial-basis function network

2.2 Radial-basis Function Network for Tier 1

In tier one, our target is to find the region in which the target IP resides. This is equivalent to finding the region that has the closest distance to the target in the Euclidean space formed by measured result. We achieve this by using a radial-basis function network(RBF)[12]. A Radial-basis function is a function in the form $h(x) = h(||x - x_c||)$. Each radial basis function has a center x_c, and the output only relies on distance $||x - x_c||$. A Radial-basis function network is a weighted linear combination of radial-basis functions. Figure 1 shows a radial-basis function network, where x_i are inputs, z_j are outputs from radial-basis functions, and y is the network output. Let $\mathbf{x} = [x_1, x_2, ..., x_n]^T$ be the input vector, $\mathbf{x}_j$ be the center point for the j-th radial-basis function. We have

$$z_j = h(||\mathbf{x} - \mathbf{x}_j||)$$
$$y = \sum_j^m w_j z_j$$

where w_i are the weight parameters to be learned from the dataset.

We first apply K-mean clustering algorithm[13] to the entire set, determine an appropriate number of clusters, and get center point of these clusters. With these center points we then create radial-basis functions centered at them and train the network.

We choose all landmarks from the lower 48 US states. With Wikipedia data of US extreme points[21], we calculated a minimal rectangle R = $[x_{min}, y_{min}, x_{max}, y_{max}]$ enclosing the lower 48 states. We then use the data to construct a regularization term Ω_b for the network. This regularization term is basically a basin-shape function giving a near zero value to points with in a region and a big value for points outside of that region. In other words, it "punishes" the results that fall out of the given region. With this term added to error function, the network will

tend to output results within the region.

$$\Omega_b = \alpha_b(\text{sigmoid}(-x+x_{min}) + \text{sigmoid}(x-x_{max}) + \text{sigmoid}(-y+y_{min}) + \text{sigmoid}(y-y_{max}))$$

where

$$\text{sigmoid}(x) = \frac{1}{1+\exp(-\lambda x)}$$

is the logistic regression function, α_b is the weight factor for this term and λ is the factor controlling the steepness of basin side.

For an input $\mathbf{x}$ and target value $\mathbf{t}$, we have $\mathbf{t}' = y(\mathbf{x})$ where y is the network function and $\mathbf{t}'$ is the network output. The error is given by $\varepsilon = ||\mathbf{t}-\mathbf{t}'||$. We measure the mean value of ε on our dataset as ε_m. We then create a circular region centered on $\mathbf{t}'$, with radius ε_m. This region is the output from tier 1. In the case this region doesn't contain enough landmark, we will enlarge the radius until either we have enough landmarks in the region or an upper bound is met.

2.3 MLP Network and Comparison Network for Tier 2

Within the region identified by tier one, we use a multilayer perceptron(MLP) network to do a more precise estimation that focuses only on that region. A MLP network is a variant of neural network that consists of multiple layers of units (aka neurons). Each neuron is a computation unit with multiple input and single output. It takes a weighted sum of previous layer's output (for the first layer, it is the network's input), apply an activation function and use the result as its output. Figure 2 shows a general example of two layers MLP, in which x_i are inputs to the network, z_i are outputs from hidden layers and y_i represent outputs from the network. We have

$$z_j = h_{\text{h}}(\sum_i^n w_{ij}x_i)$$
$$y_k = h_{\text{o}}(\sum_j^m w_{jk}z_j)$$

where h_{h} is the hidden layer activation function and h_{o} is the output layer activation function. The weight factors w_{ij} are the network parameters we need to determine.

We use a two layer MLP network with a logistic activation function in hidden layer , and a linear function in output layer. This structure had been proven capable of simulating arbitrary functions with enough neurons[4]. We chose the size of neurons in hidden layer to be twice as much as input size after observing that the target value distribution follows a relative simple model in a local

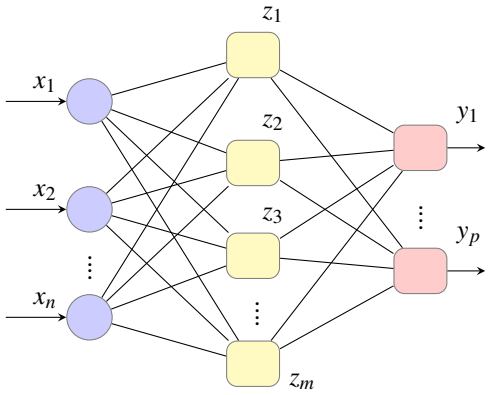

Figure 2: Two-layer MLP network

area. We then use standard techniques including error propagation and Levenberg-Marquardt method to train the algorithm.

To deal with variance in dataset we mentioned in previous section, we add a regularization term Ω_v to mitigate its impact. In [2] Bishop gives an overview of such techniques in general, and we apply it to our special case.

$$\Omega_v = \sum_{i=1}^{n}\sum_{k=1}^{m}\sum_{t=1}^{N} J_{kti}^2$$

where α_v is the weight factor for this term and

$$J_{kti} = \frac{\partial y_k}{\partial x_{ti}}$$

is the $[k,i]$ element of the Jacobian Matrix of the network near data point $\mathbf{x}_t$. J_{tki} can be evaluated effectively using numerical differentiation.

Within the region in which the target IP resides, we use a regularization term based on logistic sigmoid function to constrain the estimated value to this region.

$$\Omega_r = \alpha \cdot \text{sigmoid}(||x - x_{\text{center}}|| - \varepsilon)$$

where α and sigmoid is the same as described in previous section, x_{center} is the center for target region and ε is the radius.

The training process of MLP is a non-linear optimization process. This means some results may represent local minima rather than reaching a global minimum. We mitigate this by repeating the training process multiple times and choose the one that perform best on the entire dataset.

To compare performance of MLP network with other regression models, we develop a comparison network.

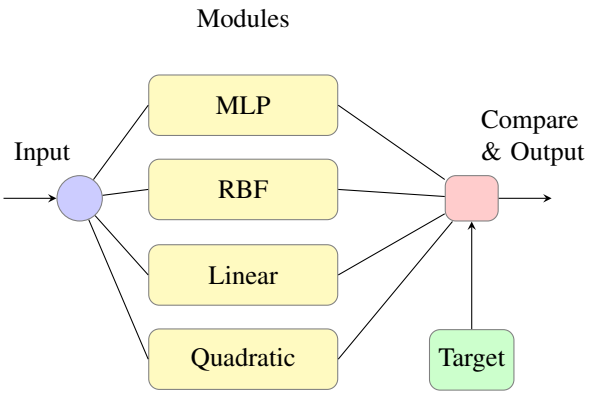

Figure 3: Comparison Network

This is a framework that allow us to easily compare estimation performance from different models. The structure is shown in Figure 3. In comparison network, each regression model is implemented as a module. The framework copies input to different modules simultaneously and collect estimation output from them. It then compares these output value with target values, calculate error and generate report. As a comparison to MLP network, we again use RBF network in the second tier.

In addition to estimate the location of target IP, we are also interested in the endemicity hypothesis. We build a comparison network that apply simple regression algorithms to the same input data, and compare their performance with MLP performance. If any of these regression algorithms gives a similar performance as MLP, we have a strong evidence that these data points actually has an analytical distribution. In our current implements, we start with two regression methods: Linear and Quadratic regression. This structure can be easily extended to higher order regression or other models. We use Volterra series to further simplify the calculation. A discrete P-order Volterra series is defined as

$$h(x) = h_0 + \sum_{p=1}^{P} H_p(x)$$

where

$$H_p(x) = \sum_{\tau_1=a}^{b} \sum_{\tau_2=a}^{b} \dots \sum_{\tau_p=a}^{b} h(\tau_1, \dots, \tau_p) \prod_{i=1}^{p} x(n-\tau_i)$$

$h(\tau_1, \dots, \tau_p)$ is called Volterra kernel. A p-order Volterra series can be viewed as a linear combination of all terms with order less than or equal to p. We use first-order Volterra series for linear regression and second-order Volterra series for quadratic regression. The Volterra kernel in both case can be easily calculated by adopting least square method.

3 Data Collection

We use Ripe Atlas network[17] to collect measurement data. All local processing is done using a desktop with AMD A10-6800K CPU and 8G memory. The desktop is installed with Ubuntu 14.04 and Sun JDK 1.8.0_05. The system is implemented using Java. Our test data and source code is available from the project website.

Ripe Atlas network is comprised of thousands of small hardware devices("probes") that connect to Internet. These probes spread across the world. Through platform API, we can issue *ping* and *traceroute* command from these probes and collect results. To make sure that measurement results capture the characteristic of the distribution of landmarks' geographic location, we need the observers to be distributed as evenly among the target area as possible. We choose 14 probes from Ripe Atlas network as observers considering that a) the region should contain enough backup nodes so that we can maintain a relative stable observation b) the geographic distribution of observers should be even. Figure 4 shows their locations.

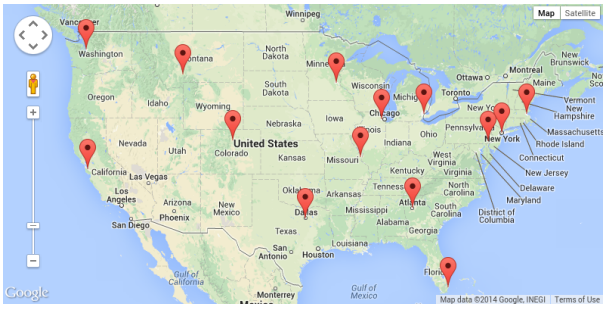

Figure 4: Observer Locations

3.1 Landmarks

We collect our landmarks from three datasets: Ripe Atlas probes, university websites and city government websites. Figure 5 shows the landmark distribution. Table 1 list the detail of these landmarks.

Ripe Atlas Probes: Probe owners are required to provide physical locations of their probes when they register them. This information, together with probes' IPs, is made available to us. These probes are designed to respond to *ping* and *traceroute*, as far as their residency networks don't filter these requests out. This makes them the perfect choice for landmarks. At the time we wrote this paper, Ripe Atlas manages 637 public and active probes in the United States. Comparing to PlanetLab platform that was widely used in previous research

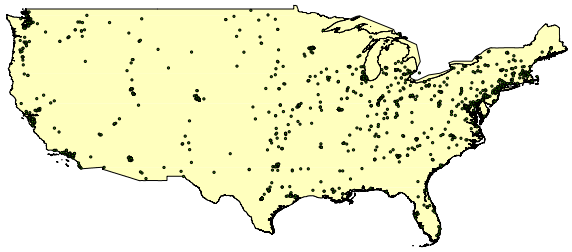

Figure 5: Landmark Distribution

Category	Raw	Valid	Reachable
Ripe Atlas Probes	637	637	429
University Websites	2170	1858	826
City Government Websites	2880	740	292
Total	5687	3235	1547

Raw: All landmark candidates
Valid: Landmarks after filtering and cross-validation
Reachable: Landmarks that respond to *ping*

Table 1: Landmark Detail

[11, 23, 9], Ripe Atlas provides a lightweight hardware solution and thus lower the threshold to participants. It has a better coverage (PlanetLab has less than 300 sites in US) and is increasing fast. We update probe information from Ripe Atlas database in a daily basis, adding new probes to our dataset.

University Website: Academic institutions are believed to have a higher chance to hold their webserver locally. Thus University dataset is widely used in previous research as either landmarks or test sets [8, 11, 20]. However the size of those datasets are limited($\sim$150) and thus is too small for our propose. We target at maintaining a complete university dataset that covers ALL universities in US. With information from [22], we retrieve a list containing 2170 distinct entries of university names and their states. For each entry, we use scripts to automatically query Google Websearch API[7] for the corresponding website, then use *host* and *whois* service to retrieve corresponding IP address, alias name and the owner organization name.

We adopt various automatic methods to wash out invalid data. We first exclude entries that use cloud service such as Amazon AWS and Rackspace Hosting to hold their webserver. We do this by looking for organizations that don't contain keywords like "university", "college" or "institute" in their names, but own more than one IP address from different universities. Table 2 lists top cloud providers providing web hosting service for universities.

Hosting Service	Count
Rackspace Hosting	134
Amazon.com	102
Media Temple	34
Unified Layer	33
GoDaddy.com, LLC	29
Linode	27

Table 2: Top Cloud Providers for University Websites

We also pay close attention to websites that own multiple IP addresses on different subnets. An organization with a large allocation of IP addresses space has potential to be a cloud service provider. We extract these organizations and further use Google to check their website for keywords like "network service", "web hosting", "cloud". Organizations contain those keywords in their websites will the considered cloud service provider and IP addresses belong to them will be removed.

We are left with 1858 IP addresses after the washing operations, and the corresponding campus' geographic locations are again obtained through Google Websearch API. This construct our University Website dataset. In addition, our result again confirms the assumption that academic websites are mostly locally hosted.

City Dataset: University dataset contains landmarks spreading among most part of the United States. However, their coverage is more dense in eastern area and western coast, leaving gaps in Mid-west areas. We hope that the introduction of City dataset can mitigate this situation.

Most US local government have their own websites and we want to extract those who are hosting their websites locally. We retrieve US atlas data from [15], which gives us 38186 cities and towns together with their geographic location and population. We assume that bigger cities with more population has a higher chance to host their webserver locally. We choose top 60 cities from each state, again make use of Google's Websearch API[7] to query the city name together with its state to retrieve its website, and apply similar methods that we used to process University Dataset to it. We also cross-validate the IP addresses using MaxMind GeoLite City Database[14]. After washing, we get 740 entries, which comprise our city dataset. Unlike the case of university dataset, in which more than 85% of the universities host their website locally, only 25% of the cities are found hosting their website locally.

Dataset	Unreachable	Partial Failure
Probe	196(32.18%)	12(1.97%)
University	809(43.97%)	28(1.52%)
City	437(58.58%)	11(1.47%)

Table 3: Failed ping in different datasets

3.2 Performing Measurement

With 3235 potential landmarks collected, we use Ripe Atlas to do our measurement work. We choose to use *ping* to measure the landmarks primarily considering performance and Ripe Atlas usage quota. However, we believe that *traceroute* definitely provides more information than *ping* and has potential to provide us a more accurate result. We have a plan to address this in future research.

We do *ping* to each landmark for 5 times, retrieving the average RTT as the measurement result. We ignore a landmark if *ping* from any probe to it fails. If the landmark is not reachable from all observers, it should be caused by either network configuration or firewall settings. Such landmarks are marked and will be discarded if the failure persists among multiple round of test. Partial error, in which case this landmark is reachable by some observers but not by others. This may just caused by temporary network jamming. This leaves us with only 1547 valid result. Table 3 shows the failure distribution in each dataset.

We also notice that measurement result may vary with time due to network topology changes or hardware upgrades. Thus to keep an up-to-date measurement result set is crucial to the accuracy of our measurement. We choose to update our measurement result on a monthly basis.

4 Experimental Result

In this section, we show our experimental results with the new approach. To measure the ability of our method to predict the location of unknown IP addresses, we use leave-one-out cross-validation. That is, for each landmark, we will first take it away from the dataset, use the remaining data to train the model and estimate the landmark's location with the model. The error is the difference between landmark's real location and estimated location.

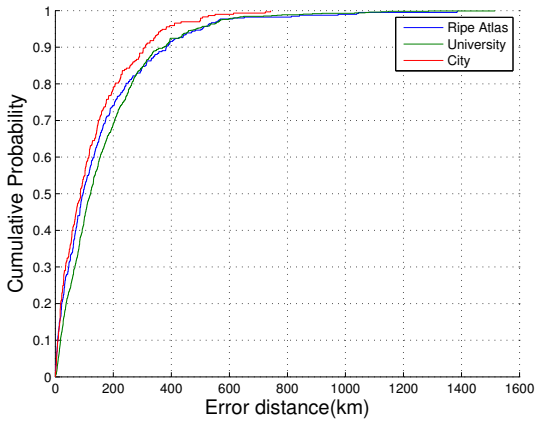

Figure 6: Error Distribution in Different Dataset

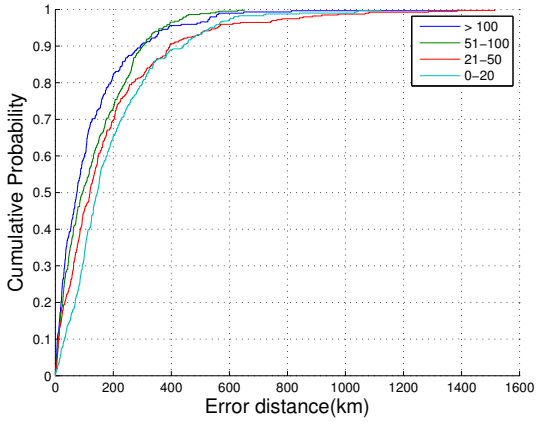

Figure 7: Error related to Landmark Density

4.1 Tier 1

Figure 6 shows the cumulative distribution function (cdf) of error in estimation results with Radial-basis function network in Tier 1. With the curiosity to know whether our method will have preference to network type(e.g. academic network, residence network and commercial network) , we separate the results by landmark sources. As we have mentioned before, Ripe Atlas probes belong primarily to academic network and residence network. University websites use academic network and city government websites use commercial network. Thus any preference to network type should lead to difference in their cdfs. In the figure we notice that there's no significant difference in three cdfs. This shows that our method has no obvious preference to academic or commercial networks. The median error in all three datasets is around 110 km. This figure also shows us that over 85% errors are less than 300 km.

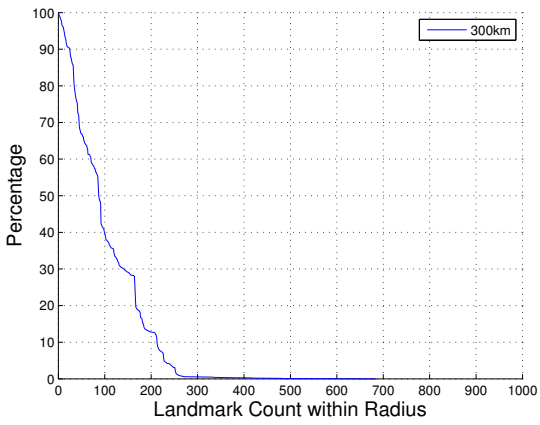

Figure 8: Landmark Density

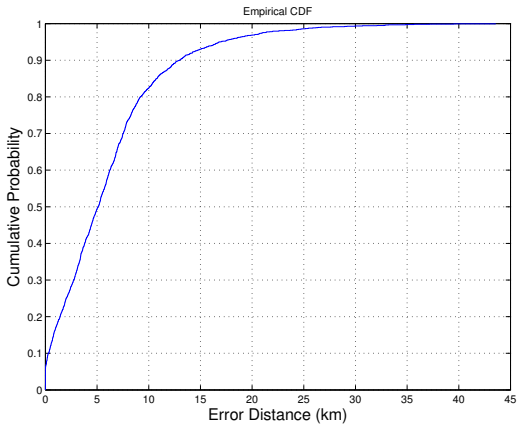

Figure 9: MLP Error Distribution for Tier 2

As is shown in Figure 5, our landmark distribution is not even. There are more landmarks in eastern half of the US and on the West coast and fewer in the mid-west. We are also interested in whether our estimation results are impacted by landmark densities in an area. In Figure 7 we separate the landmarks into 4 bins by the number of landmarks that can be found in its surrounding with a radius of 300 km and show their cdfs. Not surprisingly, the errors are larger in regions with fewer landmarks. However, the difference is small. This shows that the radial-basis function network is relatively resilient to landmark density variance.

In Figure 8, we show how many landmarks can be found within a circle centered on each landmark with radius of 300 km. (x, y) in the graph means y% landmarks have no less than x landmarks surrounding it. From the graph we know that 70% of the landmarks has at least 50 landmarks in surroundings. We choose 300 km as our estimation region radius.

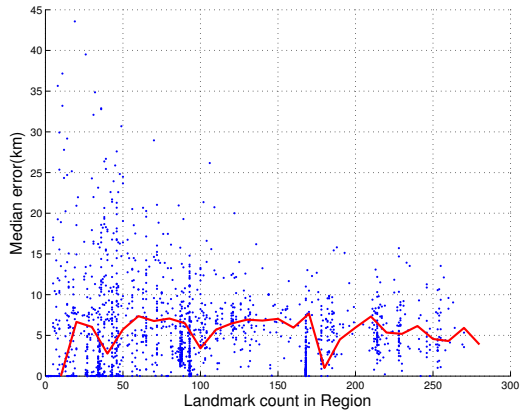

Figure 10: MLP Error related to Landmark Counts

4.2 Tier 2

To test the performance of tier 2, we construct 1547 sub datasets. Each sub dataset consists of one centered landmark along with all other landmarks within 300 km of it. With each sub dataset, we use all surrounding landmarks to estimate the center landmark's location and obtain an error distance.

We empirically choose the MLP hidden layer size to be 60% of the input data size. We repeatedly train the training process 50 times for each dataset, each time randomly split the input dataset to 80% training set, 10% test set and 10% validation set. The network with smallest sum of squared error value on the training set is used to do final estimation. Figure 9 shows the error distribution. It can be seen that the median error is 5.1 km, and in over 80% cases we have an error of less that 10 km.

We are also interested in how MLP performance varies with landmark density. We plot such relationship in Figure 10. Here x axis denotes the landmark count in each sub dataset, and y axis is the error distance in km. The blue spots shows error distance of each region. We also study how the median error changes with landmark density. We separate the data into 30 bins based on their landmark count (0-300) and get the median error in each bin. The red line in the figure shows this trend. It can be seen that the median error becomes slightly smaller with the increase of landmark density. In half of the regions where we have more than 100 landmarks, MLP have achieved a median error of 3.7 km, while in regions where we have less than 50 landmarks, the median error is around 6 km.

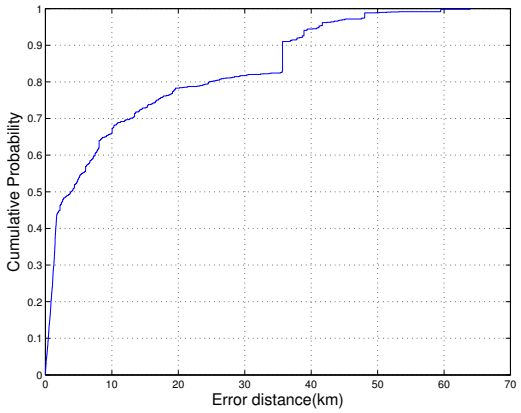

Figure 11: RBF Error Distribution for Tier 2

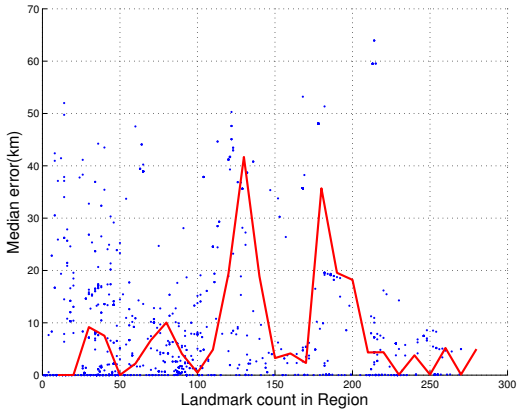

Figure 12: RBF Error related to Landmark Counts

We repeat the same experiment using RBF network. In Figure 11 we show the error distribution. It can be seen RBF gives a slightly better result than MLP with a median error of 4.1 km. However, it has a longer tail and in the worst case the error distance reaches over 60 km. On the contrast, MLP has a worst case of around 35 km. We also show how RBF performance varies with different landmark densities in Figure 12. It can be noticed that RBF's result is not as stable as MLP in different regions. We can notice two obvious peaks near 130 and 180 to reach around 40 km, while in MLP the median error is stably around 5 km. We also notice that the median error of RBF doesn't show a decreasing trend with the increasing of landmark density. Actually when counting the median error in regions with more than 100 landmarks, we get an even slightly higher result of 4.4 km. This result shows us than in regions with higher landmark densities (¿100), MLP has better performance than RBF. Overall, these results proves our hypothesis that using only local

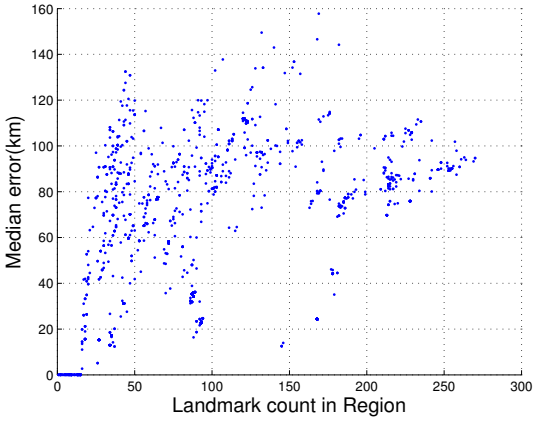

Figure 13: Linear regression Error related to Landmark Counts

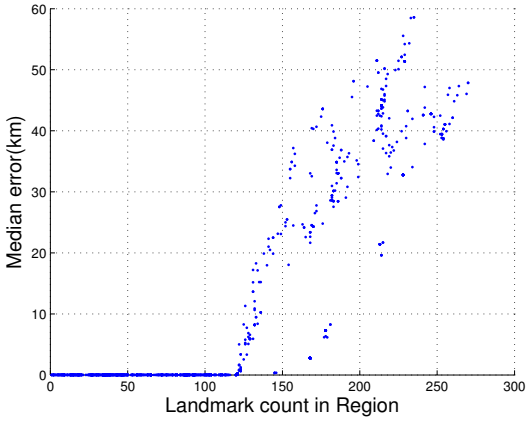

Figure 14: Quadratic regression Error related to Landmark Counts

landmarks can lead to a better estimation result.

We also demonstrate our result of using linear regression and quadratic regression model to fit data in sub datasets. In linear regression case, We have 14 variables corresponding with the 14 input to the network and 1 variables representing a constant shift. Thus this model can fit dataset with less than 15 data points. However if it cannot fit dataset with more than 15 data points, we could assume that there's no linear relationship between network latency and its positions. Figure 13 shows the result using linear regression model. In the figure we can see that when the dataset size exceeds 15, the performance deteriorate instantly. This shows that there's no linear relationship even in a small region.

Similarly, in quadratic regression case we have 105 quadratic cross terms, 14 linear terms and 1 constant

term, which in total gives us 120 variables. Figure 14 shows the experiment result. When dataset size exceeds 120, the median error of estimation increase rapidly. These results show that although our experiments reveals that we can use nearby landmark to estimate an IP's location, we cannot describe this relation in simple linear or quadratic models.

5 Discussion

5.1 Landmark Collection

Our method is a data-based method. Hence landmark density has a big impact on the accuracy. In previous section we have shown that although landmark density doesn't show big impact to the radial-basis function at tier one, the performance of MLP network in tier two deteriorate sharply when there's not enough input. Although we adopted multiple approaches to mitigate the problem, the best method to solve this problem is to collect enough landmark in the target area. There are several ways in which we can increase the landmark count in a specific area. In [20] Wang et al. introduce their data-mining based method to collect landmarks in a specific area. We can also make use of existing IP databases. Although their accuracy are still in doubt [10], these IP databases provide us a valuable hint of looking for possible IP addresses in some areas. With cross-validation using our radial-basis network, we can efficiently wash out invalid data and leave only valid landmarks. We are currently working on collecting landmark information from IP databases and will update our result using the new landmarks.

5.2 Ping vs. Traceroute

Many arguments have been raised on the suitability of *ping* as Internet measurement. Pelsser et al.[16] test the performance of *ping* on both international connections and domestic networks. They found that there exists a large variance from *ping* result due to network load balance. However, due to the performance limitation of Ripe Atlas platform, we choose *ping* in our current experiment. To mitigate the possible impact from the inaccuracy of *ping* result, we append a regulation term to the error function that makes the network invariant to this variance. Another problem caused by *ping* is the high failure rate in measurement. In 3235 landmarks we collected, only 1547 respond to all the *ping* requests. This is not surprising knowing that many gateways and routers will filter ICMP requests to prevent DDOS attack. But it greatly reduces the number available landmarks. *paris-traceroute*[1] is a measurement tool that solves the inaccuracy due to load balancing. It also supports TCP protocol to avoid ICMP filtering. We are working on collecting measurement results from our landmark set using *paris-traceroute* and compare the result with our current result with *ping*.

In addition to these advantages provided by *paris-traceroute*, we also want to make use of the network path information it provides to further increase the accuracy of our estimation. To do this, a key point is to find a way to map network paths to vector space while preserving the information the paths convey. In our observation, we find that as to identify the similarity of the destination, we need only to compare the last several hops that are close to the destination. Repeating IPs in these hops means shared routers and may be used to increase the accuracy of estimation.

6 Related Work

A noticeable category of previous work in IP geolocation is work based on a delay model. These methods start with the assumption that network latency has a direct relationship with physical distance between nodes. The physical distance is then mapped to a Euclidean space in latitude and longitude. Mathematical models can then be applied to this space and the locations of unknown nodes are estimated based on other known nodes.

Many effort had been done in this category. Gueye et al.[8] propose Constraint-based Geolocation (CBG), which send *ping* to a target from a series of observers. With experiments, they estimate an upper bound of the physical distance between these observers and the target based on RTT. This allows them to treat the problem as a quadratic programming and solve it to get the position of target IP. Katz-Bassett et al. [11] extend this idea by taking intermediate hops into account and propose Topology-based Geolocation (TBG). They use *traceroute* instead of *ping*, which allows them to collect latencies between hops. They take both the upper bound and the lower bound between nodes into account, and relax the problem to a semi-definite programming. Wong et al. [23] use Bézier curves to address the region target IP may stay in, rather than doing a point estimation. Dong et al. [5] use polynomial to replace linear relationship in the modeling and adopt a method similar to [11] to solve it.

These methods give promising results with a median error of around 20 miles ([23, 5]) However, they suffer from the inaccuracy caused by two factors. First, the

model describing the relationship between latency and physical location may be a mixture of models with different parameters instead of a single one. Second, latitude and longitude construct a roughly spherical space rather than a Euclidean space, which leads to slight difference in the translation between distance and their coordinates.

Wang et al. [20] propose a very promising method based on measurement similarity. They start by using CBG to determine a big area that target IP resides in, then collect landmarks in that area using data-mining techniques. With the analysis that physically nearby IP addresses should have similar measurement results, they compare the estimated distance between landmarks and target IP, choosing the closest landmark as the estimation to the target IP. We are very motivated by the detailed description to data-mining methods the authors use in their paper to collect landmarks. In our experiment, we suffer from inadequate landmarks in Mid-west region. By adopting their methods we have the hope to increase landmarks in some specific region and thus further improve the accuracy.

Data based methods is another fruitful category in IP geolocation field. Our method partially belongs to this category. In [24] Youn et al. describe a method based on gradient-descent and forced direction. Eriksson et al.[6] introduce their method based on Bayesian approach to increase IP geolocation estimation accuracy. These methods are good examples showing data-based methods' potential in IP geolocation. We believe that these methods can be improved by taking endemicity property into account.

Craig et al. [18] gives a solution to the last-meter problem, allowing the location of an IP address to be mapped to a physical street address. They adopt a method that requires the house to be located has a wireless router, which can be satisfied easily. They first locate the region of target IP based on the result from [20], then make use of a mobile observer(generally a vehicle) that is capable of monitoring wireless signals to scan that area. By sending to the target IP a packet with specific pattern and monitor that pattern, they are able to locate the house that router resides in.

Another method worth mentioning is IP databases. These databases provide a quick way to address IP geolocation problem. However, this method also suffers from problems such as outdated data, incomplete coverage and inaccurate information. Huffaker et al.[10] conduct a survey on available IP databases in market. Their results shows that IP databases have a median error (88 km $\sim$ 727 km) higher than state-of-the-art IP geolocation methods and thus is incapable of being used to do IP geolocation alone. However, IP databases has their advantages such as wide coverage and fast access speed. We can combine two methods by use IP databases to look for landmark candidates in a specific region, then update IP database entries regularly using our method.

7 Conclusion

In this paper we have shown a novel approach that accurately geolocates arbitrary IP address with fixed number of landmarks. Starting by observing the key problem preventing previous model-based methods from achieving high accuracy, we make a hypothesis that an important characteristic of IP location distribution conditioned on its measurement result is endemism, which means nearby points have similar distributions in their measurement results. To prove this hypothesis, we create a two-tier neural network system. The first tier searches for a small region that target IP resides in, and the second tier accurately locate IP address in that region. Thus by a divide-and-conquer method, instead of building one single model, our approach actually build a mixture of local models. To train and test our system, we collect a large dataset covering academic network, commercial network and residential network. We also build a system that regular update measurement results from these landmarks and train the system to keep our estimation result up-to-date.

We test our system on this dataset extensively, showing a similar accuracy as previous state of the art, with a fixed landmark dataset. This proves our hypothesis about endemism in the distribution, also opens a new door for people who want to further investigate on this topic. We also do a series of detailed analysis to show the application range as well as the limit of our method. Comparing to previous model-based methods, our system is capable to achieve a higher accuracy by addressing the endemism ignored before. Comparing to previous similarity-based methods, we showed that by adopting model prediction, far less landmarks are required to achieve similar accuracy. This accurate and logistic-efficient approach can be expected to have wide applications in real world.

Acknowledgement

We thank Professor Yaoqing Liu for providing us access to Ripe Atlas network, Professor Thomas Anderson and Professor Ethan Katz-Bassett for their valuable suggestions and comments, Othman Ait Maatallah for discussing with us the mathematical models used in this paper.

References

[1] Brice Augustin, Xavier Cuvellier, Benjamin Orgogozo, Fabien Viger, Timur Friedman, Matthieu Latapy, Clémence Magnien, and Renata Teixeira. Avoiding Traceroute Anomalies with Paris Traceroute. In *IMC '06*, pages 153–158, New York, NY, USA, 2006. ACM.

[2] Christopher Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[3] Matt Calder, Xun Fan, Zi Hu, Ethan Katz-Bassett, John Heidemann, and Ramesh Govindan. Mapping the Expansion of Google's Serving Infrastructure. In *IMC '13*, pages 313–326, New York, NY, USA, 2013. ACM.

[4] G. Cybenko. Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2(4):303–314, 1989.

[5] Ziqian Dong, Rohan D.W. Perera, Rajarathnam Chandramouli, and K.P. Subbalakshmi. Network measurement based modeling and optimization for {IP} geolocation. *Computer Networks*, 56(1):85 – 98, 2012.

[6] Brian Eriksson, Paul Barford, Joel Sommers, and Robert Nowak. A Learning-based Approach for IP Geolocation. In *PAM '10*, pages 171–180, Berlin, Heidelberg, 2010. Springer-Verlag.

[7] Google Inc. Google Websearch API, July 2014.

[8] Bamba Gueye, Artur Ziviani, Mark Crovella, and Serge Fdida. Constraint-based Geolocation of Internet Hosts. *IEEE/ACM Trans. Netw.*, 14(6):1219–1232, December 2006.

[9] Zi Hu, John Heidemann, and Yuri Pradkin. Towards Geolocation of Millions of IP Addresses. In *IMC '12*, pages 123–130, New York, NY, USA, 2012. ACM.

[10] B. Huffaker, M. Fomenkov, and k. claffy. Geocompare: a comparison of public and commercial geolocation databases - Technical Report . Technical report, Cooperative Association for Internet Data Analysis (CAIDA), May 2011.

[11] Ethan Katz-Bassett, John P. John, Arvind Krishnamurthy, David Wetherall, Thomas Anderson, and Yatin Chawathe. Towards IP Geolocation Using Delay and Topology Measurements. In *IMC '06*, pages 71–84, New York, NY, USA, 2006. ACM.

[12] R.P. Lippmann. Pattern classification using neural networks. *Communications Magazine, IEEE*, 27(11):47–50, Nov 1989.

[13] S. Lloyd. Least squares quantization in pcm. *Information Theory, IEEE Transactions on*, 28(2):129–137, Mar 1982.

[14] MaxMind. GeoLite City Database. http://dev.maxmind.com/geoip/legacy/geolite/.

[15] National Atlas. Cities and Towns of the United States, October 2013.

[16] Cristel Pelsser, Luca Cittadini, Stefano Vissicchio, and Randy Bush. From Paris to Tokyo: On the Suitability of Ping to Measure Latency. In *IMC '13*, pages 427–432, New York, NY, USA, 2013. ACM.

[17] RIPE NCC. RIPE Atlas Network. https://atlas.ripe.net/.

[18] Craig A. Shue, Nathanael Paul, and Curtis R. Taylor. From an IP Address to a Street Address: Using Wireless Signals to Locate a Target. In *Presented as part of the 7th USENIX Workshop on Offensive Technologies*, Berkeley, CA, 2013. USENIX.

[19] W3C. Geolocation API Specification. http://dev.w3.org/geo/api/spec-source.html, July 2014.

[20] Yong Wang, Daniel Burgener, Marcel Flores, Aleksandar Kuzmanovic, and Cheng Huang. Towards Street-level Client-independent IP Geolocation. In *NSDI '11*, pages 27–27, Berkeley, CA, USA, 2011. USENIX Association.

[21] Wikipedia. Extreme points of the United States, 2014.

[22] Wikipedia. Lists of American institutions of higher education, 2014.

[23] Bernard Wong, Ivan Stoyanov, and Emin Gün Sirer. Octant: A Comprehensive Framework for the Geolocalization of Internet Hosts. In *NSDI '07*, pages 23–23, Berkeley, CA, USA, 2007. USENIX Association.

[24] I Youn, B.L. Mark, and D. Richards. Statistical geolocation of internet hosts. In *ICCCN 2009*, pages 1–6, Aug 2009.