# Learning and Predicting Flow Duration and Rate

Kok-Kiong Yap, Kwong-Aik Goh and Myunghwan Kim
a final report for CS229, Fall 2008-09

December 12, 2008

## 1 Introduction

OpenFlow is an exciting Stanford technology, which enabled flow-based networking with centralized control. The architecture is intended to open up opportunities for networking. Here, we explore an interesting possibility which is to provide network monitoring and dynamic traffic engineering in network. However, these applications require the central controller to be aware of the rate and duration of each flow. While this information can be readily polled, it is often difficult to do so for each and every flow, given the number of flows in a network.

In an OpenFlow network, each flow can be identified with the source, destination, entry switch, transport type and etcetera. This often communicates a fair amount of information on the nature of the flow. For example, TCP traffic on port 22 is likely to be secure shell connections. Exploiting this, this project attempts to investigate the feasibility of learning and predicting flow rates based on the above mentioned metrics in a network. This allows for adaptation of the prediction in accordance of the different networks. For example, an enterprise network is likely to have different traffic combined to Google's data center or Stanford's network.

This project investigates the accuracy of various learning algorithms in prediction of flow rate and duration in a network. To compare different algorithms, exact data are logged from a network and used in quick assessments of the algorithms' suitability to the problem. Network data are recorded.

## 2 Problem Setup

Supervised learning algorithms is applied to problem above. This is because the outputs, *i.e.*, flow rate and duration of the training sets, are known values, which are in the space of positive real numbers.

### 2.1 Data acquisition

To evaluate accuracy and efficiency of the various learning algorithms, we log over 69,000 flows in an OpenFlow network, for a single user. The inputs associated with each flow are

1. Ingress port and switch data-path identity
2. VLAN identity and type
3. Ethernet source and destination address
4. IP source and destination address
5. Transport source and destination address

With these, we seek to predict the flow rate and duration of each flow. In this project, we use hold-out cross-validation to assess the accuracy of each learning algorithm. Here, we use 49,585 flows as the training set while using the rest of the 20,837 flows to estimate the generalization error.

| Output Types | ... | Class 1 | Class 2 | Class 3 | Class 4 | Class 5 | Class 6 | ... |
|---|---|---|---|---|---|---|---|---|
| Durations(sec) | ... | $1 \sim 10$ | $10 \sim 100$ | $100 \sim 1$K | 1K $\sim$ 10K | 10K $\sim$ 100K | 100K $\sim$ 1M | ... |
| Rates(bps) | ... | $1 \sim 10$ | $10 \sim 100$ | $100 \sim 1$K | 1K $\sim$ 10K | 10K $\sim$ 100K | 100K $\sim$ 1M | ... |

**Table 1:** Output Classes

### 2.2 Ouput classification

We discretize the continuous output into a few classes rather than exact values. This allows us to use more advanced algorithms such as GDA and Naive Bayes. We intend to discretize flow rates and durations into the following classes respectively. Note that we choose classes to discretize along a logarithm scale due to the large range of values and the heavy-tailed distribution of flow rate and duration.

## 3 Learning Algorithm

We implement and test the following learning algorithms to obtain our goals. In this section, we describe what is the mathematical problem of each algorithm and how it is applied. We predict each output independently, thus only for the definition, we state the problem using one output feature, $y \in \mathbf{R}^m$ where $m = 49,585$ is the number of training samples. Then, we obtain the cross-validation error by examining $m_{CV} = 20,837$ hold-out samples.

### 3.1 Linear regression(LR)

$$\min_\theta \frac{1}{2}\sum_{j=1}^{m}(y^{(j)} - \theta^T x^{(j)})^2$$

where $x^{(j)} \in \mathbf{R}^{(n+1)}$ is each training sample whose components indicate input features and an intercept term($n = 10$). Every input feature is definitely a discrete class label, neither continuous nor meaningful in a vector space. However, these features can be managed as coded values, therefore we can regard models differently according to coding scheme. We do not gurantee that this coding scheme is the best one, but at least we know this provides a lower bound of the prediction. Then, we classify outputs based on the values, $\theta^T x^{(i)}$ for $i = 1, 2, ..., m_{CV}$.

### 3.2 Generalized Linear Model(GLM)

For most of network flows, we can imagine that their durations are short and rates are low. It could be a great guess that given network features, durations and flow rates are exponentially distributed, $y|x;\theta \sim \text{Exp}(\lambda)$. Under this assumption, we can try to fit the generalized linear model, since Exp($\lambda$) belongs to exponential family where $\eta = -\lambda, b(y) = 1, T(y) = y, a(\eta) = -\log(-\eta)$. We solve the following problem,

$$\min_\theta \frac{1}{2}\sum_{j=1}^{m}(y^{(j)} - h_\theta(x^{(j)}))^2$$

where $h_\theta(x) = -\frac{1}{\theta^T x}$ and input features are defined as same as section 3.1. Similar concept about the input features can be applied here. Then, we classify outputs based on the values, $h_\theta(x^{(i)})$ for $i = 1, 2, ..., m_{CV}$.

### 3.3 Gaussian Discriminant Analysis(GDA)

Another thought is that there exists a centroid for each class and farther points from the centroid is less likely to belong to the given class. One of possible assumption is that network flows in the same class are normally distributed. After estimating of $\Sigma$ (common covariance), $\mu_k$ (a centroid for each class), $\phi_k$ (probability for each class) according to maximum likelyhood, we select a class that maximizes $p\ (y = k|x)$. This class is equivalent to

$$\arg\max_k \quad \exp(-\frac{1}{2}\ (x-\mu_k)^T\ \Sigma^{-1}\ (x-\mu_k))\ \phi_k$$

where input features, $x$, are exactly same as section 3.1, therefore the concept is still same.

### 3.4 Simple Naive Bayes(SNB)

We can make a rough assumption that every input feature is conditionally independent given output, $y$. Thus, by estimating each probability $p\ (x_i|y)$ for $i = 1, 2, ..., n$, we need to solve

$$\arg\max_k \quad p(y)\ \prod_{i=1}^{n} p\ (x_i|y)$$

Unlike above cases, Naive Bayes works indepently of the coding scheme, since this algorithm does not use input features' actual value. Rather, it counts every event and estimate the probability of each feature given outputs.

### 3.5 Complex Naive Bayes(CNB)

Acutually, we cannot guarantee conditional independence described in section 3.4. It is easy to show a counterexample; for instance, when $x_i$ and $x_j$ are respectively a source IP address and a source transport port, $p(x_j|y, x_i)$ is different since different server provides different services. Or, it is possible that $x_i$ does not offer any service because $x_i$ is a client. Therefore, at least, coupling IP address and transport port for both source and destination seems necessary. The class number to choose is

$$\arg\max_k \quad p(y)\ p(x_1, x_2|y)\ p(x_3, x_4|y)\ \prod_{i=5}^{n} p\ (x_i|y)$$

where $x_1$, $x_2$, $x_3$, and $x_4$ respectively indicate source IP address, source transport port, destination IP address, and destnation transport port.

### 3.6 Support Vector Machine(SVM)

The basic idea of this algorithm is similar to section 3.3, however its detail process is totally different. We do one-versus-all SVM which partitions the problem to 5 independent problems and picks up the class corresponding to the largest function value $f(x) = w^T\phi(x)+b$. Since this algorithm is rooted on the geometric distance, it is more important to define the distance between two flows than in the other algorithms. However, the difference of class labels mentioned in section 3.1 has no meaning. For example, we cannot say that $a_1$ =10.10.10.255 is closer to $a_2$ =10.10.11.0 than $a_3$ =10.10.11.10, even though $\mathbf{dist}(a_1, a_2) < \mathbf{dist}(a_2, a_3)$. A kernel method gives a great possibility to transform the input features, then we solve the following problem for each binary classification,

$$\begin{array}{ll} \max_\alpha & \sum_{i=1}^m \alpha_i - \frac{1}{2}\sum_{i,j=1}^m y^{(i)}y^{(j)}\alpha_i\alpha_j K(x^{(i)}, x^{(j)}) \\ \text{s.t.} & 0 \le \alpha_i \le C \\ & \sum_{i=1}^m \alpha_i y^{(i)} = 0 \end{array}$$

where $K(x^{(i)}, x^{(j)}) = \exp(-\frac{1}{2}\sum_{p=1}^n 1\{x_p^{(i)} \neq x_p^{(j)}\})$. The computational complexity is too large for the entire training set, we apply the randomized algorithm in [LR06].

# 4 Result

Using Matlab, we make a prediction and evaluate the error rate for each algorithm. The table shows their error rates; on the other hand, Figures display the error distributions. The error rate of the best fit is below 1% for duration and approximately 25% for flow rate in terms of CV error.

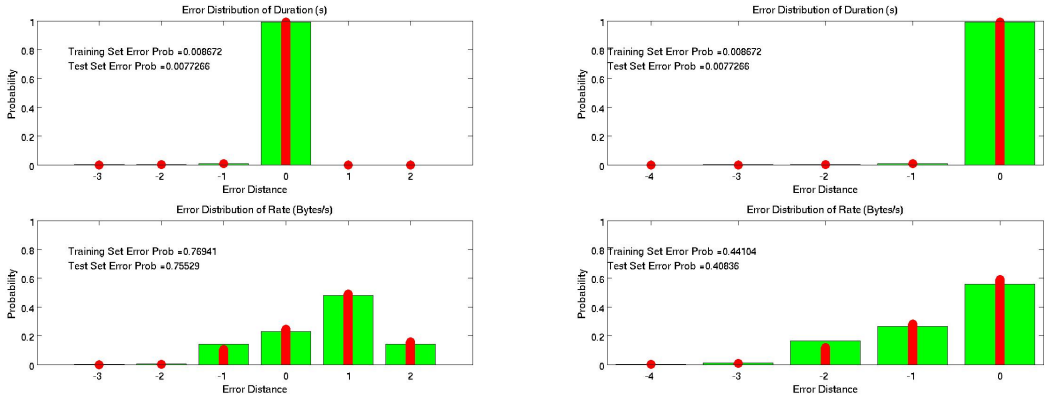


**Figure 1:** LR(left), GLM(right) / Duration(upper), Flow rate(lower)

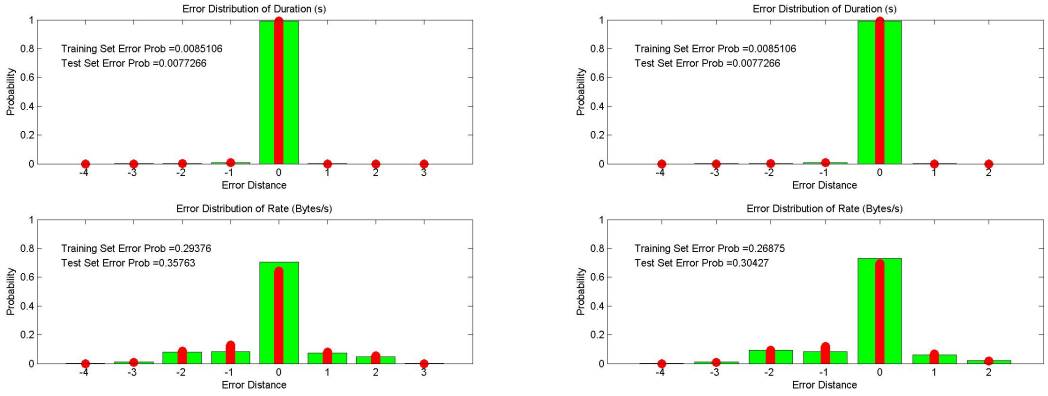


**Figure 2:** SNB(left), CNB(right) / Duration(upper), Flow rate(lower)

| Algorithm | Training error(duration) | CV error(duration) | Training error(flow rate) | CV error(flow rate) |
|---|---|---|---|---|
| LR | 0.0087 | 0.0077 | 0.7694 | 0.7553 |
| GLM | 0.0087 | 0.0077 | 0.4410 | 0.4084 |
| SNB | 0.0086 | 0.0077 | 0.2938 | 0.3576 |
| CNB | 0.0085 | 0.0077 | 0.2688 | 0.3043 |
| GDA | 0.0268 | 0.0313 | 0.3456 | 0.2136 |
| SVM | 0.0090 | 0.0079 | 0.4018 | 0.4266 |

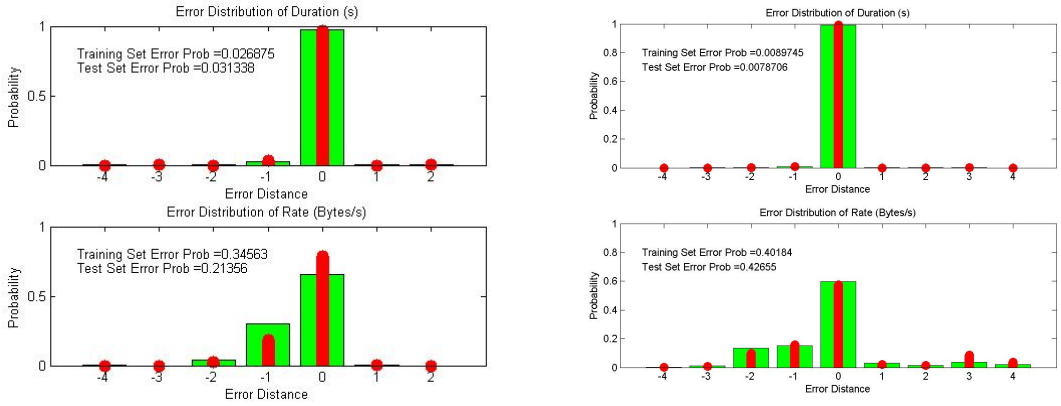


**Figure 3:** GDA(left), SVM(right) / Duration(upper), Flow rate(lower)

# 5 Conclusion and Future Works

As shown in the previous section, the accuracy of duration-prediction is very good, while that of rate-prediction is not. Unfortunately, in our data set, duration does not vary much compared to flow-rate. When the duration becomes to change a lot, we cuold not be sure that its error rate still remains low.

To evaluate how our models work well, we need to know the lower bound of error rate, Bayes rate. However, to estimate the Bayes rate, statistically many samples with the same network features are required. Due to the issue, with leaving it as the future work, we see the results in the practical view.

For the flow rate, over 25% error is not good enough for the prediction to be useful. Assuming that it might be possible to improve the accuracy, we consider reasons why they do not work well. One possible reason is that there exist nonlinear factors in the features. Because we usually apply linear algorithms, it may not reflect any high nonlinear factors. Another reason which we can come up with is that the number of features is not enough. For instance, if the flow rate of TCP stream depend on TCP state rather than IP address or transport port, then our feature selection is wrong.

# References

[LR06] Yumao Lu and Vwani Roychowdhury. Parallel randomized support vector machine. In *PAKDD*, 2006.