Online Learning for URL classification

Onkar Dalal ONKAR@STANFORD.EDU
496 Lomita Mall, Palo Alto, CA 94306 USA

Devdatta Gangal DEVDATTA@GMAIL.COM
Yahoo!, 701 First Ave, Sunnyvale, CA 94089 USA

Abstract

This work consists of a review of online algorithms for URL classification followed by some extensions and tweaking of these methods to make them efficient in terms of computational time and memory. We found out that the trade-off in error for these extensions are fairly comparable for some of these algorithms. We also applied two more kernel based methods namely Forgetron and Projectron.

1. Introduction

In this project, we study the online algorithms for classification of the URLs as malicious or benign. We analyze and reproduce the earlier work on this and then extend the same and analyze the change in error for these extensions.

1.1 Outline of report

We begin with the previous work on this data in section 2. Given the enormous size of the data and number of features, we have focused on applying feature selection and Forgetron like techniques to these methods. In section 3, we give our methods for restricting the size of feature set for the 4 methods. It also talks about other kernel-based algorithms: Forgetron and Projectron and our modification to Forgetron. The report ends with conclusion and references.

1.2 Data

The data was taken from the UC Irvine ML Repository. It is a multivariate, time-series data consisting of approximately 2.3 million data points with 3.2 million attributes per point. The data is spanned over 120 days and the URLs come with a label +1 or −1 based on whether they are malicious or benign. One of the important features of the data is sparseness which compensates for its large size.

2. Previous Work

This section is reproduction of the work in [4]. We studied the four algorithms suggested by them and reran them to match their results. This was essential to understand of the algorithms and further suggest extensions in section 3.

2.1 Basic Algorithms

We study six algorithms for online classifications i.e. a system in which we receive pairs (X_t, y_t) at every point t, such that X_t is the input feature vector and $y_t \in \{-1, +1\}$ is the actual classification of the data point at time t. However, we do not get the label y_t before we have to classify it using the old data $(X_1, y_1), (X_2, y_2), ..., (X_{t-1}, y_{t-1})$, where each X_t is a feature vector and $y_t \in \{-1, +1\}$ is the label. The label prediction is given by $h_t(X_t)$, which for linear classifiers is $h_t(X) = sign(w_t \bullet X)$, where w is the weight function.

2.1.1 Online Perceptron

This is a classic linear classifier that updates the weights in following manner for every misclassified sample:

$$w_{t+1} \leftarrow w_t + y_t x_t \tag{1}$$

This is a very simple, fast and memory efficient algorithm but all misclassifications are treated similarly in this algorithm.

2.1.2 Logistic Regression with Stochastic Gradient Descent

The Stochastic Gradient Descent (SGD) is perfectly suited for online applications. Here SGD is applied to logistic regression to optimize the log-likelihood function with following update rule for parameters:

$$w_{t+1} \leftarrow w_t + \gamma \frac{\partial L_t}{\partial w} = w_t + \gamma \Delta_t x_t \qquad (2)$$

where L_t is the log-likelihood for sigmoid distribution σ and $\Delta_t = \frac{y_t+1}{2} - \sigma(w_t \bullet x_t)$ is the difference between the actual and the predicted likelihood that the label is +1. The training rate γ is kept constant. This update is similar to the Perceptron, but the training is proportional to Δ_t and the model is updated even if the prediction is correct.

2.1.3 Passive-Aggressive (PA) Algorithm

In this algorithm, the following optimization problem is solved for each sample to minimize the change in model [1]:

$$\begin{aligned} &w_{t+1} \leftarrow arg\min_w ||w_t - w||^2 \\ &s.t. \quad y_i(w \bullet x_t) \geq 1 \end{aligned} \qquad (3)$$

If the prediction is below some confidence level - i.e., $y_t(w_t \bullet x_t) \leq 1$, we update the parameters as follows:

$$w_{t+1} \leftarrow w_t + \alpha_t y_t x_t \qquad (4)$$

where $\alpha_t = \max\{\frac{1-y_t(w_t \bullet x_t)}{||x_t||^2}, 0\}$. The use of confidence measure while classification makes this algorithm better in practice.

2.1.4 Confidence-Weighted(CW) algorithm

As given in [3], in this algorithm, we maintain the weights and a confidence measure on them (as opposed to confidence on classification in PA). The weights w_i's are modeled as Gaussian random variables with mean μ_i and variance Σ_i. At each step we make minimal change to the model to classify x_t with probability η. This is equivalent to minimizing the KL divergence between the Gaussians under the constraint of confidence measure:

$$\begin{aligned} &(\mu_{t+1}, \Sigma_{t+1}) \leftarrow arg\min_{(\mu,\Sigma)} D_{KL}(\mathcal{N}(\mu,\Sigma)||\mathcal{N}(\mu_t,\Sigma_t)) \\ &s.t. \quad y_i(\mu \bullet x_t) \geq \Phi^{-1}(\eta)\sqrt{x_t^T \Sigma x_t} \end{aligned} \qquad (5)$$

where Φ is the c.f.d of standard normal. This corresponds to the following update rule:

$$\begin{aligned} \mu_{t+1} &\leftarrow \mu_t + \alpha_t y_t \Sigma_t x_t \\ {\Sigma_{t+1}}^{-1} &\leftarrow {\Sigma_t}^{-1} + \alpha_t \, \phi \, u_t^{-1/2} diag^2(x_t) \end{aligned} \qquad (6)$$

Since every feature here is treated differently in terms of confidence, the features with less confidence are updated more aggressively. However, the heavy computation per update makes this slower than the rest.

2.2 Reworked Results

The error percentage calculated at the end of each day is plotted for the 4 methods. We see the CW is the best algorithm among the four. PA is marginally better than logistic regression. Perceptron being the most efficient happens to have the maximum error among the four. The table below gives the value of

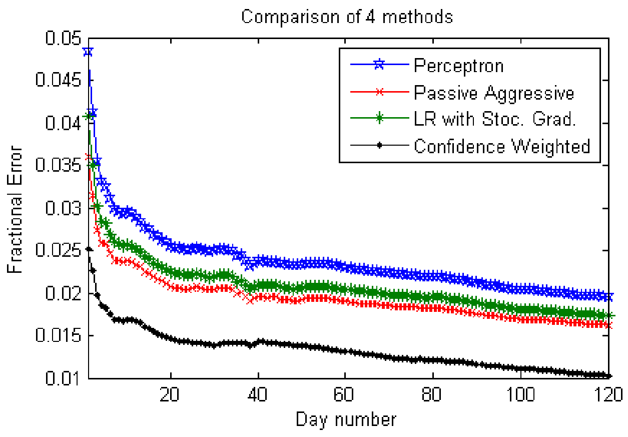

Figure 1. Errors for the four algorithms

the error towards the end of the experiment by which the algorithms have learned the model.

Algorithm	Error
Online Perceptron	1.95-1.99%
Logistic Regression SGD	1.73-1.77%
Passive-Aggresive	1.62-1.66%
Confidence Weighted	1.03-1.07%

3. Extensions

As extensions to the previous work, we have looked at following algorithms and variations of earlier algorithms in this project:

- Restrict the size of weight vector to consider only the valuable features and see the changes in errors for the four algorithms described in section 2
- Forgetron: A kernel based algorithm, which stores the examples with a fixed budget on number of support vectors [2]
- Projectron: A kernel based algorithm, which checks the projection of new data point onto support set and update only if it is sufficiently off [5]

3.1 Correct Feature selection (Limited W)

3.1.1 Motivation

Since each of the URL has over 3.3 million features, we want to store and utilize only the important features. Our motivation for limiting the memory usage (by limiting size of W) comes from the applications like Web Search where classification needs to be done in minuscule times. Another application for quick, memory efficient online classification would be to decide whether a user is a bot or not. It is important for the business to know which top features are shown by bots in contrast to real people.

3.1.2 Growth in number of features

We also looked at growth of the feature set for each of the four algorithms. The results are represented in Fig 2. As expected we can see that Perceptron quickly updates all the features as compared to linear growth in LRSD and very sub-linear growth for PA and CW.

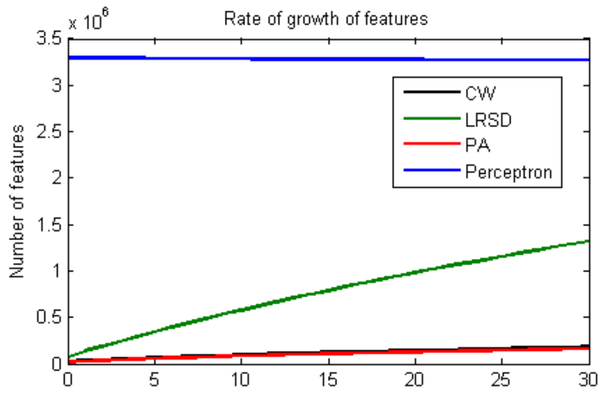

Figure 2. Growth in size of feature set with time

3.1.3 Budgeting the features

For all the four algorithms, we restrict the features selected to a budget B. We implement this budgeting by choosing only the top B features where the W vector has changed the most. We run our budgeting after every update. We observe that by setting the number of features to a budget B=0.1%-0.2% of the total features in the data, the accuracy is still comparable to otherwise. The results of this experiment for perceptron, LRSD and PA are plotted in Fig 3. The results for CW are plotted in Fig 4.

3.1.4 Observations

- Although Perceptron uses most of the features, if we restrict the number of features, the error

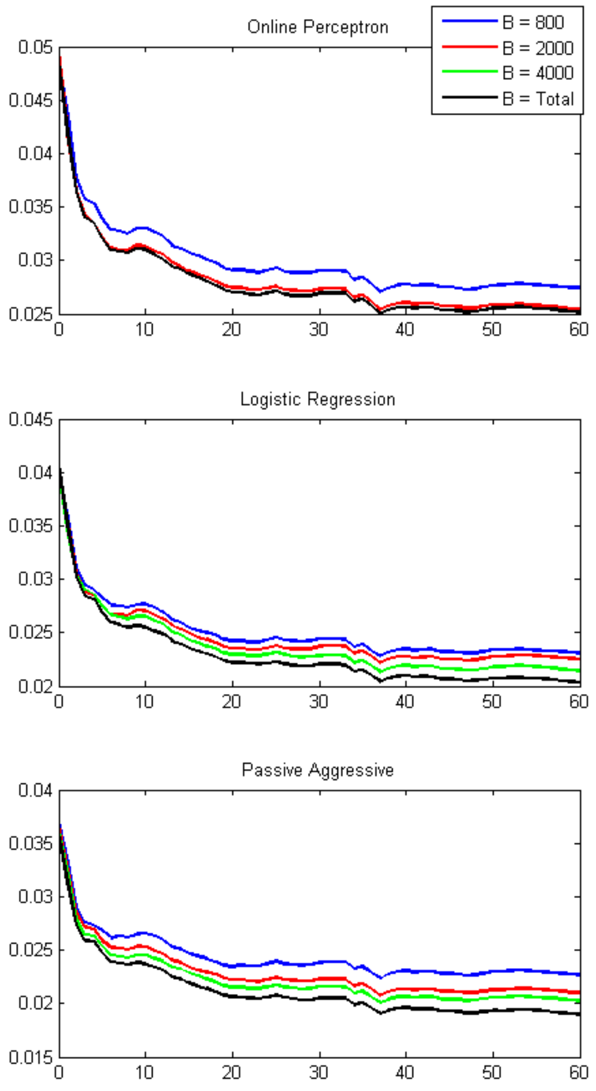

Figure 3. Error dependence on budget B for Perceptron, Logistic Regression and Passive Aggressive

change for $B = 2000$ or 4000 is about 1%.

- Even though PA uses less features (200,000) with a budget of 2000 features, the performance goes down for PA and LRSD by about 10%.
- Restricting the features in CW decreases the performance of the algorithm by a substantial degree (20-50%). This could be because CW not only uses the weight of the feature, but also the confidence in the feature as an important measure of how important that feature is.

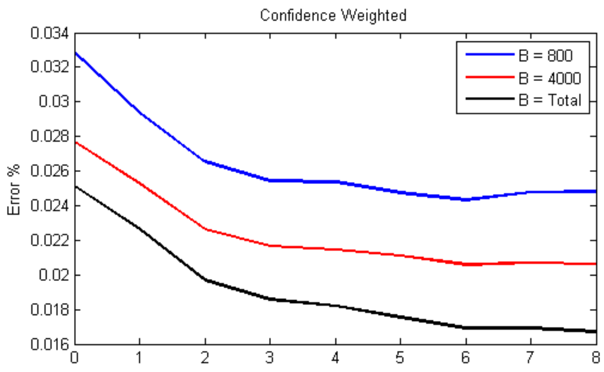

Figure 4. Error dependence on budget B for Confidence Weighted

3.2 Forgetron

Perceptron becomes very effective when used along with Kernels. But the kernel computations are costly and memory intensive. Forgetron is a kernel based online-learning algorithm with a budget on the kernel size. Forgetron assigns heavier weight to recent data points and lighter weight to old data. The exact algorithm is given in [2]. However, there are few disadvantages to the Forgetron.
At every update, Forgetron reduces the weight of the previous stored examples in the Kernel. Slowly for large B, after many updates, the weights of the very old stored example vectors become negligible. This makes the updates slower and does not add too much improvement in the errors. Re-weighting process is expensive We have tried a simplified version of Forgetron which stores latest B misclassified examples and weighs them equally. This runs 2-3 times faster and requires less memory. And for $B \geq 1000$, Forgetron stores the old samples but their weights gradually become negligible. After 10-15 days, oue simplified version performs better than Forgetron. The results for these for different values of B are given in Fig 5.

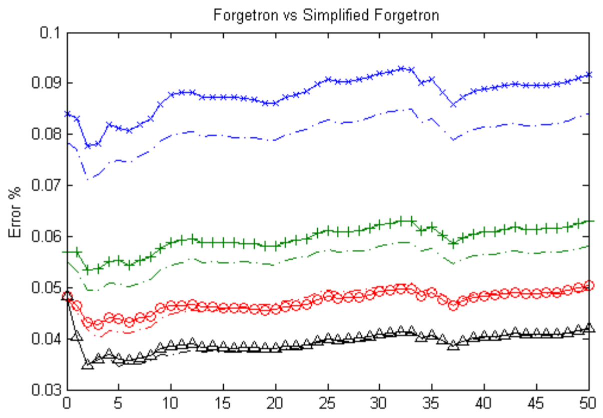

Figure 5. Comparison of actual Forgetron (dotted) and simplified Forgetron (solid) for B=200(blue), B=400(green), B=1000(red), B=2000(black)

3.3 Projectron

Another kernel based algorithm we applied was Projection. In this algorithm, whenever a data point is misclassified, we check if it lies in span of existing support set and update the weights. If however it does not lie in the support set, it gets added to the support. The details of this algorithm are in [5]. The main disadvantage of this was the magnitude of calculations required. This was the slowest of the six algorithms and the errors for linear kernel were worse than perceptron. The results are given in Fig 6.

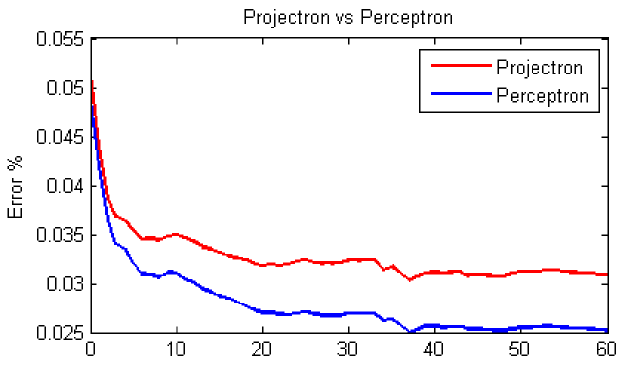

Figure 6. Projectron vs Perceptron

4. Conclusion

We started with reproducing the results in [4]. As expected Perceptron is the most efficient, but low on accuracy and CW even if slower compared to the rest, has the best accuracy. We observed that algorithms like Perceptron and LRSG use almost all the features however PA and CW add new features at a sub-linear rate. To exploit this fact, we restricted our feature size to a small but significant subset (0.1%) of the features. We observed that Perceptron works with same accuracy for restricted weights, however, the errors in CW increase substantially. In the second half, we applied Forgetron and Projectron for classification. We found that the errors in Forgetron and Projectron are worse than Perceptron. This can be attributed to the loss of information in both the algorithms. The algorithms also turn out to be slower than the other four because of the kernel calculations. However, the simplified Forgetron works very well for moderately high budgets ($B \geq 1000$).

References

[1] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:585, 2006.

[2] O. Dekel, S. Shalev-Shwartz, and Y. Singer. The Forgetron: A kernel-based perceptron on a fixed budget. 2005.

[3] M. Dredze, K. Crammer, and F. Pereira. Confidence-weighted linear classification. In *Proceedings of the 25th international conference on Machine learning*, pages 264–271. ACM, 2008.

[4] J. Ma, L.K. Saul, S. Savage, and G.M. Voelker. Identifying suspicious URLs: an application of large-scale online learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM New York, NY, USA, 2009.

[5] F. Orabona, J. Keshet, and B. Caputo. The Projectron: a bounded kernel-based Perceptron. In *Proceedings of the 25th international conference on Machine learning*, pages 720–727. ACM, 2008.