Critical Evaluation of Feed-forward Neural Network and Support Vector Machine Classifiers for Phishing Websites Dataset

Mark Longhurst & Thomas Martin

31st March 2019

INM427 Coursework

https://github.com/neural-computing/INM427-neural-computing-coursework

Abstract

This paper reports on a critical evaluation of two machine learning models in the task of identifying phishing websites. The models considered are a feed-forward neural network and support vector machine classifier. For each model, k-fold cross-validation was performed to determine the best configuration of hyperparameters in training. The performance of each trained model was then checked against a test set, using a confusion matrix and other derived metrics. Against these evaluation metrics, the feed-forward neural network achieved better performance.

I. Introduction

Phishing websites represent a subset of the wider phishing problem [1]. In general, phishing relates to any attempt to fraudulently obtain sensitive personal information in an electronic communication. Phishing websites are websites that trick users into believing they are on an otherwise non-phishing website, typically using a range of frontend web technologies in order to gain this information [2]. This is an increasingly important problem due to the increasing reliance on web-based services [1].

This paper aims to evaluate the performance of two models in the task of identifying phishing websites based on 9 features. The models considered in this paper belong to feed-forward neural networks and support vector machine (SVM) family of models. We use cross-validation to perform hyperparameter tuning from the parameter space.

The paper is organised as follows, section 2 provides an exploratory overview of the dataset used in the project, section 3 details the approach taken in the project to train and compare the models, section 4 discusses the results of the previous process, with section 5 providing a final conclusion.

I.I Feed-Forward Neural Network (FNN)

Feed-forward neural networks refer to the generalised, multilayer perceptron model. These model consist of nodes or neurons arranged in input and output layers, typically with one or more hidden layers in between. Individual nodes are connected by edges called "weights", which denote the strength of the relationship between any node pair. During the training process, these weights are adjusted following an algorithm such as gradient descent, which is a systematic process of determine the impact a given node had on the output produced in a forward pass [3]. They are high performing models in supervised tasks especially where the dataset is large, high dimensional, and unstructured [4].

I.II Support Vector Machine (SVM)

Support vector machine (SVM) classifiers determine the classification of data points by finding the hyperplane of maxmimum margin separating the classes of the dataset. SVM is an appropriate for finding nonlinear boundaries, even in case of high-dimensional datasets using a kernel function [5]. Though typically used in binary classification tasks, SMVs can be generalised to non-binary classification problems by following either one-vs-one or one-vs-all classification algorithm [6].

Compared to a feed-forward neural network, SVM classifiers operate similarly to a shallow neural network, however SVMs are generally thought to produce more easily understandable models.

II. Dataset

The dataset used in this study was taken from the UCI Machine Learning Repository, originally collected from the Phishtank data archive [7]. The dataset contains features corresponding to 1353 websites, classed as either non-phishing, unknown, or phishing, encoded as 1, 0, and -1 respectively. There is a slight imbalance between these classes occurring with a frequency of 548, 702 and 103 for non-phishing, phishing and unknown samples respectively. However, this imbalance is not dramatic enough to require additional sampling methods. The 9 features and target contained within this dataset were: SFH, popUpWindow, SSLfinal_State, Request_URL, URL_of_Anchor, web_traffic, URL_Length, age_of_domain, having_IP_Address, and Result.

All features take one of 3 values 1, 0 and -1 and the framework that was originally used to produce them is explained within [8].

III. Methodology

This sections outlines the general approach taken to train and test either model. as well a approaches specific to each model.

III.I General Approach

Whilst prototyping, the initial dataset was split into a training and testing dataset in a proportion of 70% and 30% respectively. Once the models were built, a grid search methodology was employed during the initial training process, grid-search was used to select the optimal hyper-parameter configuration.

At this point, k-fold cross validation was leveraged instead of randomly generating test/train splits. "k" was set to 5 effectively delivering 80% of the dataset for training and 20% for testing within an particular fold. Cross-validation is especially useful to get a better understanding of how well the data models remain unbiased and generalise in the case of relatively small datasets - the original dataset has ~1000 rows. This is because when selecting a small number samples from a dataset, we cannot reliably trust that the resulting collection contains the same distributions/patterns as the original dataset.

The test accuracy over each of the "k" folds was then averaged to determine the overall test accuracy allowing an optimal set of hyper-parameters for each model to then be selected.

To evaluate the performance of the trained models against the test set, confusion matrices were plotted. These provide a simple visual breakdown of the classifiers performance. Additionally metrics of precision and recall can be used to understand the classifiers accuracy with respect to particular class an well as its sensitivity (true-positive rate). Such an approach also helps given there is a class imbalance, which accuracy considered alone could lead to a biased classifier.

III.II Feed-Forward Neural Network (FNN)

Initially a fully connected feed-forward neural network was built utilizing the sigmoid function as the neuron activation function for all nodes. The number of input and output layer nodes was kept consistent, equal to the number of predictor variables plus bias, and classes respectively. This can be seen in Figure 1.

Weights were randomly initialised for each epoch within a given threshold, to ensure that networks weights can reach different minima for each run. The weights of the network were updated following a backpropagation algorithm, updating weights according to the mean squared error of the output layer nodes. An early stopping threshold was also set to ensure the training was stopped for a given epoch if the updated accuracy changed little from its previous value.

During the training stage, the following hyperparameter were tuned:

- Hidden nodes - number of fully connected nodes within the hidden layer, varied at increments of 5 between 10 and 60
- Learning rate - rate at which the models weights are updated during the back-propagation, varied at increments of 0.05, between 0.001 and 0.1

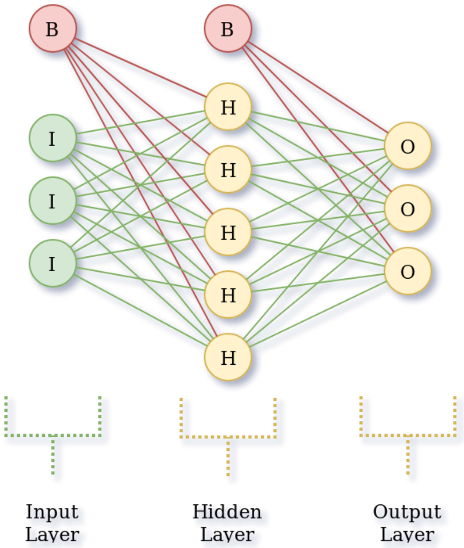

Figure 1: Feed Forward Neural Network, where input, hidden, output layer and bias nodes are indicated with letters I, H, O, and B respectively.

- Momentum - level of inertia added when modifying the models weights, varied at increments of 0.025, between 0.005 and 0.05
- Early stopping threshold - smallest delta allowed when updating the models weights before training is halted, this varid as increments of 0.05, between 0.001 and 0.1

III.III Support Vector Machine (SVM)

Next a multiclass SVM model was constructed using MATLAB's "fitcecoc" method. The SVM state is completely defined by the initial hyperparameter set. Given that the task is a mutliclass classification problem, a one-vs-all algorithm was used to produce the classifications.

The following hyperparameters were tuned:

- Kernel - this determines how the SVM performs comparisons between data points to determine the hyperplane of maximum margin. The selection available consisted of linear, RBF, and polynomial kernels
- Box Constraint - this controls the penalty on data-points that are misclassfied by the margin and overall helps with overfitting. Increasing the box constraint will reduce the number of support vectors the SVM margin uses. This was varied between 0.05 and 1, at increments of 0.3
- Kernel scale - this is a measure of the variance of the data points. This was varied between 0.1 and 1, at increments of 0.3
- Shrinkage period - this is the number of iterations between reductions of the active set. This was varied between 1 and 10, at increments of 3

IV. Results

IV.I Model Selection

In total, 128 models were run for both the Neural Net and SVM. The following tables reproduces the top ten configurations for both models, ordered by test accuracy.

Top 10 Configurations for SVM

Table 1: Top 10 model configurations for SVM

Box Constraint	Kernel Scale	Shrinkage Period	Kernel	Train Accuracy	Test Accuracy
0.65	1	10	rbf	0.9398	0.8877
0.95	1	10	rbf	0.9535	0.8847
0.65	1	1	rbf	0.9398	0.8840
0.95	1	7	rbf	0.9535	0.8840
0.95	1	4	rbf	0.9535	0.8832
0.95	1	1	rbf	0.9535	0.8825
0.65	1	7	rbf	0.9398	0.8810
0.95	0.7	4	rbf	0.9609	0.8766
0.65	1	4	rbf	0.9398	0.8721
0.95	0.7	7	rbf	0.9609	0.8707

Top 10 Configurations for FNN

Table 2: Top 10 model configurations for FNN

Hidden Layer Nodes	Learning Rate	Momentum	Early Stopping Threshold	Train Accuracy	Test Accuracy
26	0.046	0.005	0.001	0.9496	0.8926
26	0.046	0.03	0.001	0.9398	0.8926
34	0.046	0.005	0.001	0.9498	0.8926
34	0.031	0.03	0.001	0.9452	0.8911
18	0.046	0.03	0.001	0.9411	0.8911
34	0.046	0.03	0.001	0.9435	0.8911
30	0.046	0.005	0.001	0.9491	0.8904
22	0.046	0.005	0.001	0.9428	0.8889
30	0.046	0.03	0.001	0.9459	0.8889
22	0.031	0.005	0.001	0.9431	0.8881

From tables 1 and 2 we can see that all of the best performing SVM models used a radial basis function (rbf) kernel with a kernel scale of 1. Coupling this with a box constraint of 0.65 and shrinkage period of 10 built the best SVM model and this will be carried forward into the model comparison. Similarly with the FNN we can see that the top 3 models all achieved almost identical test accuracy. This suggests that either 26 or 34 hidden nodes, a momentum of either 0.005 and 0.03, a learning rate of 0.046 and a stopping threshold of 0.001 delivers the best performance. We selected 26 hidden nodes instead of 34 to try and reduce the training time of the model.

IV.II Model Comparison

The reported training accuracy for the most successful configurations for both models indicates that FNN performed better overall, if only slightly: 94.96% v 93.89%. More interesting perhaps is the choice of hyperparameters that produced the top performing models. In particular, the RBF kernel was the best kernel choice for the SVM classifier. The FNN demonstrated a high variability in the number of hidden layer nodes which produced the best performance.

To determine the most effective model for this dataset a confusion matrix was produced using the best model choice on the test data. Accuracy is reported for the test set only, which is defined as the percentage of true positives identified over all classes out of the total samples considered.

FNN Confusion Matrix

SVM Confusion Matrix

Figures 2 and 3 contain the confusion matrices for the kfold test accuracy achieved by the FNN and SVM models respectively. Considering the accuracy alone, we see very comparable performances between the two models as shown in table 3. The

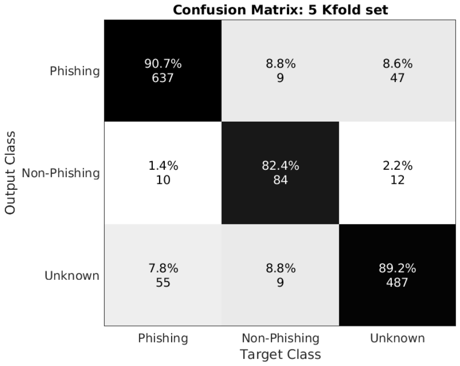

Figure 2: 5-fold confusion matrix for Best Neural Net Model

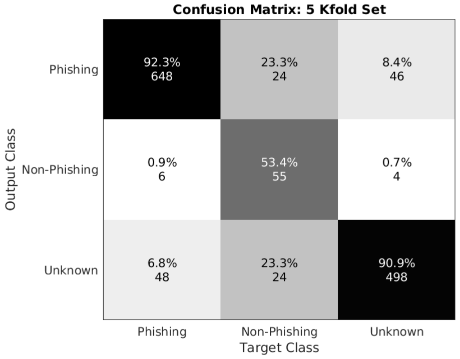

Figure 3: 5-fold confusion matrix for Best SVM Model

feed-forward network out performs the SVM model by about 0.5% and it appears to be significantly better at correctly identifying emails as "non-phishing".

Table 3: Overall Test Accuracy

	Accuracy
FNN	89.3%
SVM	88.8%

However, as the dataset contains imbalanced classes, accuracy alone is not the best way of determining the performance of classifier - as it could simply be a consequence of a model biased towards the most frequent class. When determining whether a website is phishing or not, it can be considered that the best model is the one that correctly identifies the phishing websites i.e. reduces the number of phishing website false negatives. This is precisely because, in optimising for the highest proportion of websites correctly identified as phishing we can ensure the best experience for a user: there is a strong guarantee that phishing websites are handled appropriately, without simply identifying most websites as phishing or unknown. From this perspective, considering the precision and recall with respect to each class will aid in comparing the performance of each model.

Table 4: Class Specific Precision, Recall and F1-Scores. These were calculated assuming by picking each class inturn to be the positive class and assuming that the other classes represent negative samples

Class	Model	Precision	Recall	F1-Score
Phishing	FNN	0.912	0.907	0.909
Phishing	SVM	0.903	0.923	0.913
Non-Phishing	FNN	0.792	0.824	0.808
Non-Phishing	SVM	0.846	0.534	0.655
Unknown	FNN	0.883	0.892	0.887
Unknown	SVM	0.874	0.909	0.891

V. Conclusion

In this paper, we considered the performance of two types of models, SVMs and FFNs, to correctly classify websites as either non-phishing, phishing, or unknown. It was found that both sets of models demonstrated comparable performance in this task at both the training and testing stage. For both models as well,

Given that this is a multiclass classification problem, a confusion matrix proved to be the most immediate and effective means to make meaningful comparison between the two trained model. From consideration of the problem domain, the f1-score a single optimising metric for either model, as this can be taken as a metric that ensure the best experience for an end-user. In this context, the FNN classifier performed slightly better to SVM.

Models were selected based on the test accuracy achieved during the kfold training process. The dataset used within this study contains a fairly large class imbalance with the "non-phishing" class under represented by a ratio of roughly 7-1. This may mean that using accuracy for model selection is unfairly biased towards models that are able to better predict "phishing" websites at the expense of "non-phishing" websites.

Although this paper demonstrated that the two models independently have good performance as determined by the recall metric, it is generally considered that ensemble methods generally give better performance [9]. Rather than comparing, maybe the two methods could be used in conjunction to achieve a better overall performance. This is especially the case for models that perform better in some instances compared to another

VI. References

[1] Wikipedia contributors. "Phishing." Wikipedia. Last accessed 29th March 2019

[2] "Phishing Web Site Methods", https://www.webcitation.org/5w9Z2iACi?url =http://www.fraudwatchinternational.com/ phishing- fraud/phishing-web-site-methods/. Last accessed 27th March 2019

[3] "Feed-Forward Networks", https://cs.stanford.edu/people/eroberts/courses/soco/projects/neural-networks/Architecture/ feedforward.html. Last accessed 27th March 2019

[4] J. Mahanta, "Introduction to Neural Networks, Advantages and Applications", https://towardsdatascience.com/introductio n-to-neural-networks-advantages-and-applications-96851bd1a207. Last accessed 30th March 2019

[5] L. Auria & R. A. Moro. (2008). "Support Vector Machines (SVM) as a Technique for Solvency Analysis". German Institute for Economic Research

[6] https://nlp.stanford.edu/IR-book/html/htmledition/multiclass-svms-1.html. Last accessed 29th March 2019

[7] N, Abdelhamid, Website Phishing Data Set, https://archive.ics.uci.edu/ml/da tasets/Website+Phishing. Last accessed 21st March 2019

[8] Abdelhamid, Neda & Ayesh, Aladdin & Thabtah, Fadi. (2014). "Phishing detection based Associative Classification data mining". Expert Systems with Applications. 41. 5948–5959. 10.1016/j.eswa.2014.03.019.

[9] V. Smolyakov, "Ensemble Learning to Improve Machine Learning Results", https://blog.statsbot.co/ensemble-learning-d1dcd548e936. Last accessed 31st March 2019