A Neural Network Decision Support System for Predicting the Popularity of Online News

Ekta Sardana ekta@wisc.edu
Jie Feng jfeng33@wisc.edu
Roney Michael rmichael2@wisc.edu

University of Wisconsin-Madison

Abstract: The rapid expansion of the Internet user-base has resulted in a tectonic shift in the way we interact with media. News is increasingly moving online with companies vying to maximize page-views which are intricately tied to their generated revenue. This presents an exciting opportunity to explore learning systems for predicting the popularity of online news articles prior to their publication. In this report, we detail our efforts in exploring one such system built leveraging artificial neural networks. We found that simpler networks perform better with respect to accuracy and precision, while more complex nets result in better recall and F1 score.

Keywords: Supervised Learning, Classification, Neural Networks, Online News, Popularity

I. Introduction

Intelligent Decision Support Systems (IDSSs) are entities which analyze data and provide suggestions which businesses may use to guide their decision making process. These systems wade through an abundance of data, which is often too great for a human to make sense of, and come to (hopefully) meaningful conclusions.

Digital journalism involves the act of publishing news articles to the Internet as opposed to the conventional paper medium. With the ever-increasing size of the Internet user populace, this has become a highly competitive space with numerous businesses vying for greater popularity among Internet users. Thus predicting the popularity of online news has become a research trend [1]. Earlier a story breaks, greater the number of page-views and greater the revenue. Hence, to maximize profit, businesses need to identify and promote articles likely to be popular. Popularity can be measured as: number of likes, shares or comments. Given the vastness of the space of topics, articles, writers, this poses an excellent opportunity for IDSSs to be employed. Better decisions in this regard may well be the make-or-break factor in determining the immediate virality of news articles and in the long term, could strongly influence companies' future.

II. Approach

For building our IDSS, we have used Neural Networks (NNs). The basic idea behind neural network is to model a “learning” system as a topology of smaller, simpler learning systems. It is loosely based on how neurons in biological brain communicate to one another to produce distinct inputs from outputs based on some inherent parameters which are adapted over time to refine the generated output based on an evaluated error. This lends itself to the idea that the predictive system would get better with time as the number of input training instances increases and the model can be learned to arbitrary complexity depending on available computational power. We will not be detailing the functioning of NNs here as it is beyond the scope of the report, but assume that readers are already familiar with the same.

III. Evaluation

Dataset

The dataset used to train the neural network is the “Online News Popularity” dataset which was graciously donated to the University of California, Irvine Machine Learning Repository by K. Fernandes et al., following their work on evaluating the performance of a few popular machine learning approaches on this task [1]. While performing our literature review, we noticed that though their work delved deep into the methods of Random Forest (RF), Adaptive Boosting (AdaBoost), Support Vector Machine (SVM), K-Nearest Neighbor (KNN) and Naive Bayes (NB) based classifiers, the popular class of Neural Networks had been left out. Since NNs may be trained up to arbitrary depths and associated complexity while at the same time being tunable by means of numerous parameters, we explored an IDSS built using NNs for this task.

The dataset comprises of 60 attributes of which one (the number of shares) is the target attribute, another (the URL of the article) is non-predictive and the rest are input features such as keywords, digital media content, earlier popularity of news referenced in the article [1]. To be consistent with [1] in the interpretation of results, we built a classifier instead of a regressor with an article being identified as *POPULAR* if it has 1400 or more shares and *UNPOPULAR* else; the dataset used has a fairly even split of positive and negative instances given this threshold (21154 positive and 18490 negative instances). We normalized the instances feature-wise since all are numeric and have vastly different upper and lower bounds. This idea may be irrelevant for models such as those using RFs, where each attribute is evaluated in isolation, but it

assumes great importance in case of NNs due to their intrinsic nature of working with a linear combination of the provided inputs. For the purpose of this project, the *POPULAR* class is considered as the positive class since the base idea here is for a business to identify popular articles prior to publication.

Neural Network Parameters

In this report we focussed mainly on the tuning of the following parameters:

- Learning rate
- Number of epochs
- Number of hidden layers
- Number of units per hidden layer

Different types of activation units (*Sigmoid* vs. Linear Rectifier), error functions (*Mean Categorical Cross-entropy* vs. Mean Squared Error) and learning rules (*Stochastic Gradient Descent* vs. RMSprop) were tried, with the italicised methods being chosen in each case. We also tried to flush out how varying some of the aforementioned parameters in conjunction with one another affects the network. Additionally, we experimented with batching of updates to the NN with respect to the time taken to train the system. All relevant weights and bias terms are initialized to random values in the interval [0, 1) as is the default for the scikit library we leverage [5].

Measures and Baseline

We evaluated our predictive model based on five metrics - accuracy (Acc), precision (Pre), recall (Rec), F-1 score (F1S) and area under the ROC curve (AUC) - used in [1]. In this task, the most relevant metrics are **Pre** (a measure of correctly classifying both classes) and **AUC** (a measure of the model's discriminatory power as compared to a random classifier); Rec (a measure of identifying as many *POPULAR* articles as possible) is an interesting statistic as well, but we will not be using it in tuning our parameters as it is biased toward positive predictions. As the dataset is split fairly evenly in both classes, we expect these measures to be a fair representation of our model.

We use the results stated in [1] as the baseline expectation for these metrics; the results are listed below with the greatest performance achieved has been bolded for each metric. Note here that the purpose of this report is not to engineer a model to be a better predictor, but is to study the effects of tuning various parameters in NNs; the numbers stated for the metrics are ballpark figures which exhibit the learnability of the task at hand from the standpoint of a peer-reviewed academic publication.

Model	Acc	Pre	Rec	F1S	AUC
RF	**0.67**	0.67	**0.71**	**0.69**	**0.73**
AdaBoost	0.66	0.68	0.67	0.67	0.72
SVM	0.66	0.67	0.68	0.66	0.71
KNN	0.62	0.66	0.55	0.60	0.67
NB	0.62	**0.68**	0.49	0.57	0.65

Validation

We used 40-fold cross-validation in our system with a rolling window approach as was detailed in [1]. The entire data-set is randomly shuffled before each experiment to produce uncorrelated pseudo-random ordering. Each training window comprised of 10,000 instances and the corresponding test window of the next 1,000 instances. The process is repeated in steps of 1,000 instances till the point when all instances have been tested with at least once.

System

Since the object of this project was primarily to evaluate neural network based models and not to build one ourselves, we made use of the popular Scikit learn library for neural networks [5], which provides a reasonably easy-to-use wrapper around the more-powerful, but esoteric pylearn2 learning module.

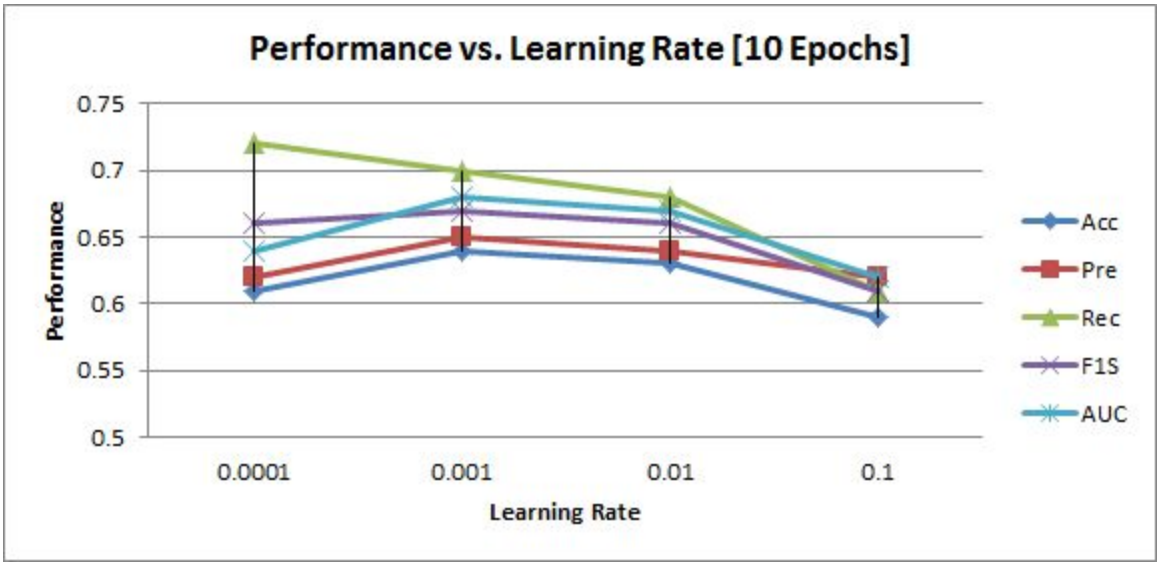
The observations and inferences gleaned from our experimental runs are made note of in the following section. We will use the general term *performance* to refer to these metrics altogether. The values of parameters set are indicated where relevant. We use 10 epochs where not explicitly indicated. The output layer has *Softmax* units and all hidden layers has *Sigmoid* units. All other values are the defaults stated in [6] (learning_rate=0.01, momentum=0.9, learning_rule=u'sgd', etc.).

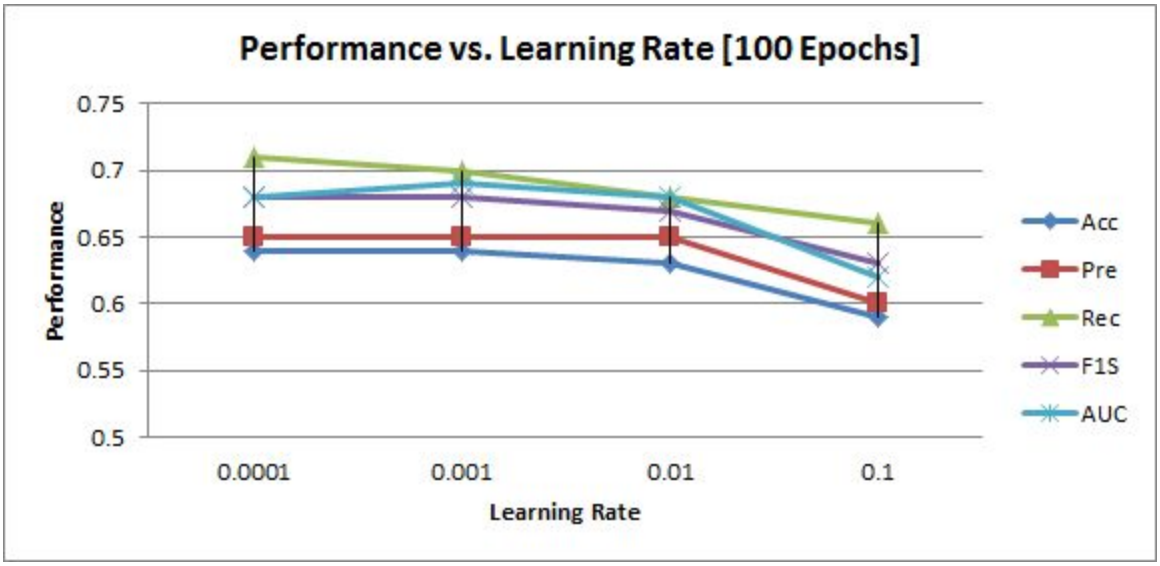
IV. Results and Discussion

Learning Rate and Number of Epochs

We evaluated the variation in performance while simultaneously adjusting both the learning rate (learning_rate) and the number of epochs (n_iter), since the two were closely interlinked. For simplicity in computation, a NN with no hidden layers was used for this evaluation. The best overall performance was obtained with a learning rate of 0.001 and 100 epochs.

learning_rate	n_iter	Acc	Pre	Rec	F1S	AUC
0.0001	100	0.64	0.65	0.71	0.68	0.68
0.0001	10	0.61	0.62	0.72	0.66	0.64
0.001	**100**	**0.64**	**0.65**	**0.70**	**0.68**	**0.69**
0.001	10	0.64	0.65	0.70	0.67	0.68
0.01	100	0.63	0.65	0.68	0.67	0.68
0.01	10	0.63	0.64	0.68	0.66	0.67
0.1	100	0.59	0.60	0.66	0.63	0.62
0.1	10	0.59	0.62	0.61	0.61	0.62





It is interesting to note here that with learning_rate=0.1, the Acc indicates trashing, while with learning_rate=0.0001 and n_iter=10, the Acc indicates that the NN may not have converged yet.

Number of Units per Hidden Layer

Various number of units (num_unit) were tried for each hidden layer in accordance with prior research by Karsoliya et al. [3]. The data provided here was calculated on a NN with a single hidden layer. Since there was a tie between two trials when considering Acc and AUC, Rec was used as the tie-breaker.

num_units	Acc	Pre	Rec	F1S	AUC
30	0.63	0.65	0.69	0.67	0.68
60	**0.63**	**0.64**	**0.71**	**0.67**	**0.68**
90	0.63	0.64	0.71	0.67	0.67

Number of Hidden Layers

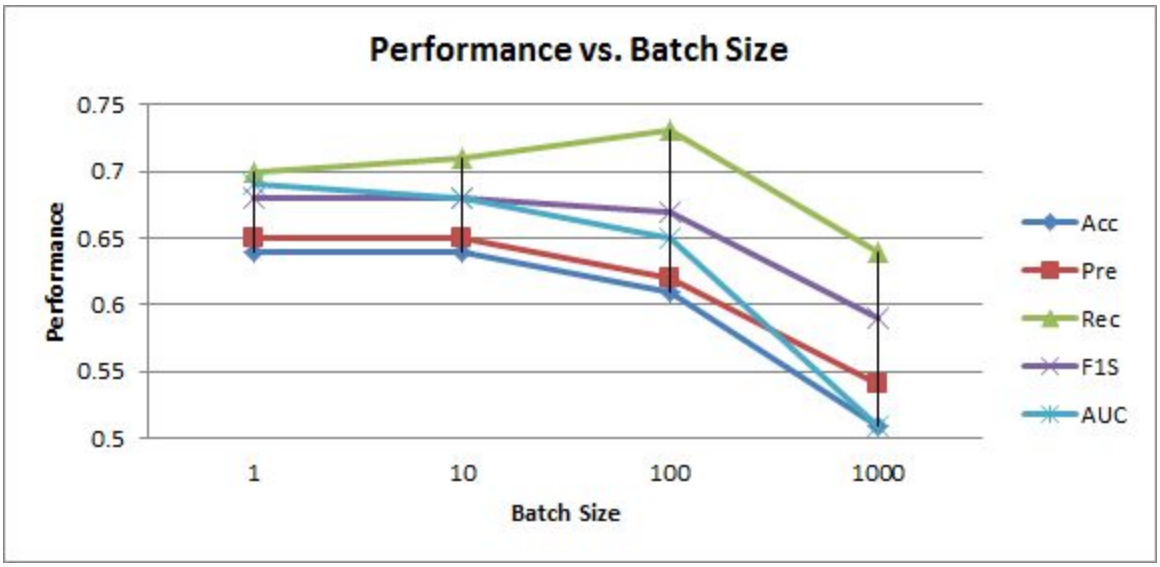
We tuned the number of hidden layers in a NN with 60 units per layer. Acc and AUC were highest for the first two trials and Rec was used as the deciding factor.

num_layers	Acc	Pre	Rec	F1S	AUC
0	0.64	0.65	0.70	0.68	0.69
1	**0.64**	**0.65**	**0.71**	**0.68**	**0.69**
2	0.64	0.64	0.71	0.68	0.68
3	0.64	0.65	0.69	0.67	0.68

Batching

A large portion of the computation involved in training a NN is owed to updating the weights involved. Typical NNs make online weight updates (once for each training instance) which is time consuming. Batching is a workaround which lets us speed up this process by trading off performance by only applying updates in batches. We evaluated this on a NN with no hidden layers, with learning_rate=0.001 & n_iter=100.

batch_size	Acc	Pre	Rec	F1S	AUC
1000	0.51	0.54	0.64	0.59	0.51
100	0.61	0.62	0.73	0.67	0.65
10	0.64	0.65	0.71	0.68	0.68
1	**0.64**	**0.65**	**0.70**	**0.68**	**0.69**



Note that the best performance in terms of our primary metrics of Acc and AUC was obtained with online training as expected, but batch_size=10 gave reasonably comparable performance while requiring only a tenth of the time.

Consolidated Model

Putting the results from our previous trials together, our consolidated model has exactly 1 hidden layer of 60 units and was trained with learning_rate=0.001 and n_iter=100. The results obtained were equivalent to our previous best (when using the same topology with learning_rate=0.01 and n_iter=10). We conclude this to be the best performance obtainable from a NN model for the task while adjusting only the aforementioned parameters.

Acc	Pre	Rec	F1S	AUC
0.64	0.65	0.71	0.68	0.69

V. Conclusion

This report explored the effects of tuning the learning rate, number of epochs, number of units per hidden layer, number of hidden layers and batch size for a neural network IDSS. Our models were evaluated using the “Online News Popularity” dataset obtained from the UCI Machine Learning Repository. The final model showcased fair to good performance as compared to previous work by Fernandes et al. on this task for all metrics except precision. Overall we found that simpler networks perform better with respect to accuracy and precision, while more complex nets result in better recall and F1 score.

VI. Acknowledgements

We would like to specially thank Dr. Mark Craven, professor in the Department of Biostatistics and Medical Informatics at the University of Wisconsin, and affiliate faculty member in the Department of Computer Sciences, for his help and guidance throughout this project.

We are also grateful to Kelwin Fernandes and his group for making their dataset freely available and for proactively providing us more information about the target task.

VII. Bibliography

1. Fernandes, Kelwin, Pedro Vinagre, and Paulo Cortez. "A Proactive Intelligent Decision Support System for Predicting the Popularity of Online News." Progress in Artificial Intelligence. Springer International Publishing, 2015. 535-546.
2. Bashiri, M., and A. Farshbaf Geranmayeh. "Tuning the parameters of an artificial neural network using central composite design and genetic algorithm." Scientia Iranica 18.6 (2011): 1600-1608.
3. Karsoliya, Saurabh. "Approximating number of hidden layer neurons in multiple hidden layer BPNN architecture." International Journal of Engineering Trends and Technology 3.6 (2012): 713-717.
4. Wilson, D. Randall, and Tony R. Martinez. "The general inefficiency of batch training for gradient descent learning." Neural Networks 16.10 (2003): 1429-1451.
5. Champandard, Alex J., et al. "aigamedev/scikit-neuralnetwork" AiGameDev.com. Web. <https://github.com/aigamedev/scikit-neuralnetwork>.
6. "Scikit-neuralnetwork Documentation" <http://scikit-neuralnetwork.readthedocs.org>.