Comparative study of Support Vector Machines and Feed-forward Multilayer Perceptrons

Vaida Gulbinskaitė, Gediminas Sadaunykas

INM427 Neural Computing – City, University of London

1. INTRODUCTION

In 2013, the NBA has started using a new statistical analysis technology, called SportsVU (Official NBA release 2013), which allows tracking of every single movement a basketball player makes on the court from the very beginning of the game to the very end. Basketball is one the greatest big data adopters in the sports industry, as team managers started using big data insights in order to make adaptations in their game strategy (e.g. the famous Houston Rockets' 'Moreyball' (Partnow, 2016)).

Deep learning in basketball has already solved problems that were unthinkable decades ago. For example, Loeffelholz et al 2009 examined the application of neural networks as a predictive tool for the success of basketball teams in NBA. Markoski et al 2011 used an artificial neural network in order to determine how the referee should move in the basketball court in order to maximise the visibility of players' actions. Shah and Romijnders 2016 applied deep learning algorithms on ball tracking data in order to predict whether the three-point shot was successful. Wang and Zemel 2016 used data extracted from SportsVU in order to solve a difficult task of offensive play call classification.

Only a few machine learning algorithms are successfully applied in the industry, professional sports, as well as academia; such as Support Vector Machines (SVM) and Artificial Neural Networks (ANN) to name just a few. A study, designed and conducted by Caruana and Mizil (2006), presented an empirical evaluation of different machine learning techniques. SVMs and ANNs performed well when judged on accuracy, overall ranked at 4th and 5th place (out of 10) respectively. On the other side, there were datasets on which SVMs and ANNs performed better (went up to 2nd place) or worse (down to 7th place). It suggests that there is no 'free lunch' in machine learning - different algorithms have different capabilities and therefore should be applied in different settings.

In this paper, we will compare and evaluate SVMs and feed-forward multilayer perceptron trained with backpropagation, (MLP) algorithms. We will investigate and compare their performance, based on speed and accuracy. Our models will be finely tuned via exhaustive grid search with cross-validation, over various sets of hyper-parameters. Due to the depth of data and our mutual passion for basketball, we have selected NBA 2015 data in order to run our study.

2. DATASET

2.1 About the dataset

In order to perform the analysis, we have used a NBA 2015 shot log data, taken from Kaggle (Kaggle.com). This dataset contains 128,069 observations, however using the full dataset caused substantial slowdowns in the search for optimal solutions ,therefore we have decided to use the first 20,000 instances. Data also has 21 features, out of which, we have selected 5 continuous, numeric variables, that hold time (touch time, shot clock), distance (shot distance, closest defender distance) and basketball specific (dribbling - the number of deceiving actions with a ball, before shot attempt) parameters. We have selected a binary value, indicating whether the shot was made or missed as an output.

2.2 Data preparation

Before the analysis, the data was explored and wrangled in Python. Using our general domain knowledge, we have noticed outliers in one of our input variables - touch time, with values more than 24sec (limit of time allowance one attack) or less than 0. Since outliers only accounted for 0.002% of the data, we decided to remove these observations from our dataset. Another input variable, shot clock, had 4.54% of the data missing. We have decided to replace missing values with median, due to its outlier insensitive properties, and speed of imputation. The output variable (shot result) had to be converted from a string variable to categorical variable (from 'made', 'missed' to made = 1 and missed = -1).

	SHOT RESULT									
	MISSED (-1) 9,115 samples					MADE (1) 10,885 samples				
	Mean	Std.	Variance	Min	Max	Mean	Std.	Variance	Min	Max
TOUCH_TIME	2.76	2.89	8.36	0.00	22.80	2.42	2.63	6.92	0.00	22.00
CLOSE_DEF_DIST	4.13	2.63	6.92	0.00	23.90	4.15	2.89	8.35	0.00	52.60
SHOT_DIST	15.09	8.47	71.77	0.10	46.30	11.72	8.73	76.24	0.10	37.10
DRIBBLES	1.95	3.34	11.13	0.00	30.00	1.63	2.98	8.90	0.00	26.00
SHOT_CLOCK	11.88	5.46	29.86	0.00	24.00	13.02	5.66	32.02	0.00	24.00

Table 1: *Descriptive statistics of input variables, segmented by output variable.*

3. METHODS

3.1. Feed-forward Multilayer Perceptron

Evolved from McCulloch and Pitts neuron (McCulloch and Pitts 1943) multilayer perceptron is now a relatively old and established deep learning algorithm, which contains input layer, one or more hidden layers and an output layer. Each layer is composed of one or more artificial neurons in parallel. Each layer of neurons is connected to a layer above and assigned with biases. Each connection is assigned with weights. Neurons are activated by using either threshold function or a semi-linear function (e.g. Sigmoid function). Multilayer perceptron does not hold assumptions about the distribution of the data, it can therefore model any[1] non - linear function and has the capacity to generalise unseen data to a great accuracy (Gardner and Dorling 1998).

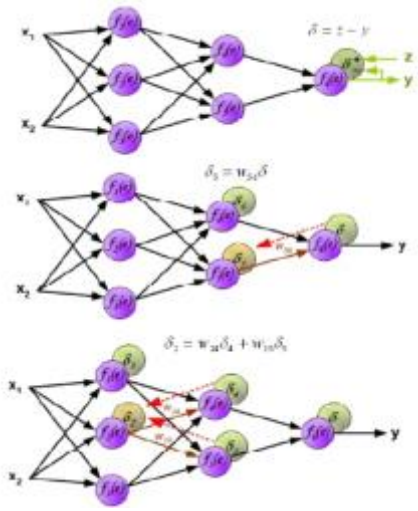

Figure 1: *Example of a part of a process of MPLs training with backpropagation Mariusz Bernacki (2005)*

The feed-forward Multilayer Perceptron algorithm is quite simplistic in nature and, therefore, by feeding input forward just once, output classification is rarely accurate (Rumelhart and Hinton 1986). For that reason, multilayer perceptron has a set of complex training algorithms (e.g. backpropagation), which greatly increases the number of optimizable parameters.

Backpropagation (Pineda 1987) is one of the most popular methods to train an artificial neural network. It is used in conjunction with an optimisation method (e.g. gradient descent). The premise of backpropagation algorithm is the calculation of error signal (δ) by subtracting output value (y) from target output value (z). The cumulative error is then fed back to earlier layers of neurons, weights and biases are adjusted so that error signal is minimised. Once the error is calculated on full data, and all weights are adjusted, one epoch of training is completed and second one starts. (**Figure 1**). There are various stopping criteria, for instance, early stopping – training ceases once error on validation dataset does not drop for a specified number of epochs.

Backpropagation is optimized by adjusting several parameters, for example: Learning rate (η) influences the quality and speed of learning; momentum (μ) - assists the algorithm in finding global minimum by preventing it from being stuck in local suboptimal solutions; learning rate decay (λ) – is regularization term, treating overfitting.

Training neural network with backpropagation could be a very time and processing power consuming task. Number of parameters, that make MLP, trained with backpropagation, a highly adaptable algorithm could also be a 'curse'. Learning rate too large (close to 1) could make the algorithm sensitive to random

[1] Universal approximation theory, states that feed-forward network with single hidden layer and finite number of neuros can approximate any function. (Cybenko 1989) https://en.wikipedia.org/wiki/Universal_approximation_theorem

noise. Large momentum (close to 1) can make the algorithm vulnerable to overshooting - which causes the algorithm miss the global minimum.

3.2 Support Vector Machines

SVM solves the constrained optimization problem, by searching for a hyperplane, which separates data points with the greatest geometrical/functional margin. The formal definition of a problem is derived from Lagrange duality, solving for Karush-Kuhn-Tucker conditions:

$$min_{w,b,\vartheta} \quad \frac{1}{2}w^T w + C\sum_{i=1}^{l}\vartheta_i \qquad s.t \quad y_i(w^T\emptyset(x_i)+b) \geq 1-\vartheta_i \qquad and \qquad \vartheta_i \geq 0$$

Instance-label pairs are defined as (x_i, y_i), where $x_i \in R^n$ and $y \in \{1, -1\}$. C is the parameter penalizing the error term. (C > 0) Input data vectors x_i are mapped into a higher dimensional space by the $\emptyset$ function. Kernel function ($K(x_i, x_j) = \emptyset(x_i)^T\emptyset(x_j)$) is used in order to improve the efficiency of computation in very high dimensional spaces [2].(Hsu et al,. 2003). Several widely used kernels include: linear (no kernel), polynomial, sigmoid, radial basis function (Gaussian)[3]. The Gaussian kernel is used in this study and is defined as: $K(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2), \gamma > 0$. **figure 2** shows two parallel lines, representing support vectors and a dashed line, which represents the maximum margin separator.

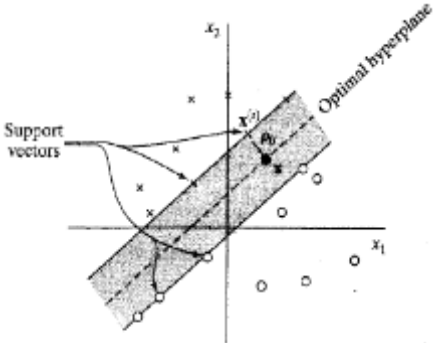

***Figure 2**: Maximization of a margin in two-dimensional space. (Haykin 1998)*

Interestingly, one of the major drawbacks to SVMs comes from some of its major assets: efficiency in the absence of domain knowledge incorporation; and guarantee for a global optimum solution. (Haykin 1998). As a result, there is limited control over the number of support vectors selected, and there is no implicit mechanism to account for the prior knowledge. Therefore, the main solving algorithm – Sequential Minimal Optimization (SMO)- is relatively slow, even when 'kernel trick' is applied.[4]

3.2 Link between SVMs and Feed-Forward MLPs

Collobert (2004) presents a careful theoretical comparison between the deep learning models we chose to compare. It shows that under some assumptions, MLPs are linear SVMs, maximizing the margin in a single hidden layer space. [5] In other words- hidden units in MLP attempt to find a separating hyperplane in the subset of an input space. Both methods share the speed issue; MLPs mainly due to the number of adjustable parameters, and SVMs due to property of finding guaranteed global optimum.

There is one main difference between these two approaches. Feed-Forward MLPs are usually trained with backpropagation which minimizes error function, whether it's a regression or a classification type of problem. However, in the classification task, SVMs minimize the number of training examples that fall between the margins of separation. (between support vectors) On the other hand, for non-linear regression task, it minimizes insensitive loss function. (Haykin 1998)

4. ANALYSIS

4.1 Hypothesis Statement

We have trained two deep learning algorithms (MPL and SVM) in order to determine which one is more suitable to solve the supervised classification problem, posed in the NBA 2015 shot data set. Accuracy of both models expected to surpass the prior probability of positive class 55%. We also hypothesise that SVM will achieve higher accuracy scores than MPL with one layer of hidden neurons, however at the expense of training time.

.

---

[2] Complexity of calculating $K(x_i, x_j)$ is O(n) – linear in data size, while $\emptyset(x_i)$ is $O(n^2)$. https://en.wikipedia.org/wiki/Kernel_method

[3] Mercer condition is necessary and sufficient for kernel to be valid, more on Mercer theorem: https://en.wikipedia.org/wiki/Mercer's_theorem

[4] A rule of thumb, to assume $O(n^2 * l)$ SMO complexity for SVM with RBF kernel, where n-number of data instances, l-number of features. https://en.wikipedia.org/wiki/Sequential_minimal_optimization

[5] If an MLP is minimum of cost function with added weight decay parameters, then it maximizes the margin in the hidden layer space. In other words, (w, b) is solution of the SVM problem , for the hidden layer found by the MLP. Collobert (2004)

4.2 Methodology

Analysis with both models was performed in parallel and could be split into 5 steps. First of all, the data was prepared, as explained in section 2.2 which involved some necessary operations, like missing value and outlier treatments. It was conducted in Python 3.5. Some other transformations, like standardization[6] of variables and transposing it into vectors, as well as core analysis, were performed in Matlab 2016b. As a part of the data preparation process, it was split into 80%(16 000 observations) training data, and 20% (4000 observations) validation data.

The second part, exhaustive grid-search, took the longest due to the vast number of parameter combinations tested. Grids of various parameters for both models could be found in **table 2.** A grid search was performed on validation data, with 10 - fold cross-validation. This means that each combination of variables was trained on 90% (3600 observations) of validation data, and tested on 10% (400 observations). Such methodology, although very computationally costly, modulates overfitting of parameters to the particular subset of data, as well as reducing the effect of chance in comparison of parameters.

Once the best combination of parameters was identified, the model was adjusted and the remaining 80% of the data was fed into it. Multiple models have been trained and evaluated with all the same parameters, however, different sizes of data in the increments of 1000 observations were used. This way, we were able to deduct the learning curves and compare the effectiveness of our models against marginal gain in data quantity.

The fourth step consisted of feeding the training data into the models with the best parameters once again, in order to find the scores along with the predicted labels. It was necessary for comparison of accuracy, performed in later steps. Models were trained with 10 fold cross-validation.
It is important to note that the 10 fold cross –validation exercise was used instead of splitting the data into another subset – testing data. This is due to the enhanced utilization of data, provided by k –fold cross-validation.

Instead of using the simple classification error, obtained in previous steps, we have decided to plot confusion matrices, and subsequent derivation of precision, recall and f1 scores in the fifth step. This has greatly improved the process of comparing the accuracy between the two models.

Despite the efforts to make the comparative analysis as similar as possible, some steps did not apply to both models. For example, the method for initializing weights, - in MLP, is quite important, because of the value of breaking the symmetry between the hidden units (Lecun et., al 1998). We have noticed this by experimenting with different methods, e.g. setting them to 0, drawing from uniform distributions and then scaling down. Different initializations gave different results, even when a high number of epochs was used. As such, drawing weights from the uniform distribution and scaling down by the inverse of square root of the number of inputs, as advocated by (Lecun et., al 1998), was chosen. Biases were initialized to equal 0.

MLP		SVM	
Paramter Name	Paramter Grid	Paramter Name	Paramter Gird
Hidden Layers	1	Box Constraint	[2^-5; 2^-4; 2^-3; 2^-2; 2^-1; 2^0; 2^1; 2^2; 2^3; 2^4; 2^5; 2^6; 2^7; **2^8**; 2^9: 2^10]
Number of neurons in the hidden layer	[50, (**75),** 100, 300]	Gausian Kernel Sigma	[2^-10;2^-9;2^-8;2^-7;2^-6; 2^-5; 2^-4; 2^-3; 2^-2; 2^-1; 2^0; 2^1; 2^2; **2^3**; 2^4; 2^5;]
Learning rate	[0.01, 0.05, 0.1, 0.5, (**0.55)**, 0.9]		
Learning rate decay	[0.1, 0.4, 0.5, 0.6, 0.8, (**0.99)]**		
Momentum	**[(0),** 0.1, 0.3, 0.4, 0.6, 0.9]		
Number of maximum epochs	[100, 500 , (**1,000)**]		

Table 3: *MPL and SVM regularisation parameters*

[6] Gaussian standardization, where means are subtracted, and values are normalized by the standard deviation.

5. Results and Critical Evaluation

5.1 Efficiency

Algorithms were run on different machines, however both machines are quite similar in computational power. As a result, such differences were not accounted for. In total, there were 256 parameter combinations tested for SVM, and 1,125 combinations tested for MLP. It took a little over 4h2minutes, for SVM, and 7hr29 minutes for MLP. On average testing SVM combinations took twice as long, compared to MPL. However, MLPs required further manual testing, due to the fact, that none of the 1,025 combinations, provided results better than chance (accuracy of 0.55) Additional testing required another 4-5 hours. Best parameters are marked with bold font for SVM, and in bold, within brackets, for MLP. (table 2)

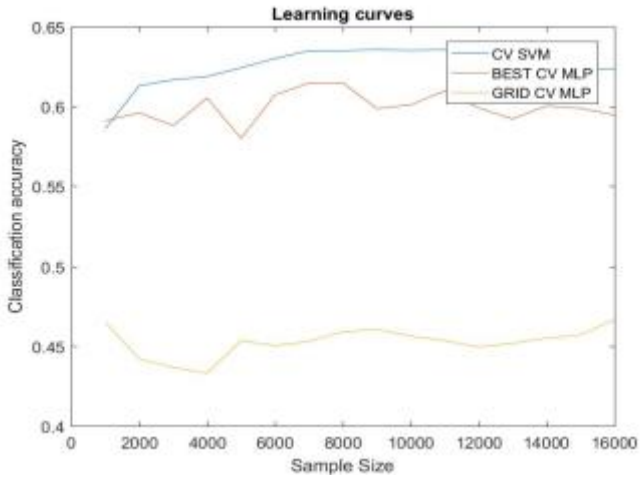

Figure 4: *Learning curves, of best SVM, best MLP, and grid-searched MLP.*

Efficiency could also be described by the capacity to pick up generalizable patters from different data sizes. This algorithm, which requires less data to achieve a certain accuracy, is considered superior to the one requiring more of it, especially if their computational requirements are similar. The distinction here is insignificant as both algorithms pick up the majority of its generalizing power within first 2-3 thousand observations. (figure 3) The accuracy of SVM follows a slight upward slope, for a couple of extra percent in accuracy by the time it is trained with 7000 observations. The accuracy of the best MLP, is more hectic, but it never drops to the levels of grid-searched MLP.

5.2 Accuracy

Classification accuracy, as expected, is a couple of percentage points higher for SVM, than for best MLP. However, this distinction becomes apparent only after the first couple thousands observations. The difference is also never greater than 5 percentage points. However, a 5 percent drop in the accuracy of MLP would make it indistinguishable from a random model, so is quite significant. Confusion matrices (figure 3) reveal a more detailed view. It shows that both models output positive[7] class disproportionally more often. Recalls are 81.37% and 85.8%, for MLP and SVM respectively. SVMs slightly more precise (61.2, against 60) which leads to the harmonic mean, or f1 scores being higher for SVM. Exploration of accuracy results confirms the advantage of an SVM, by small margin.

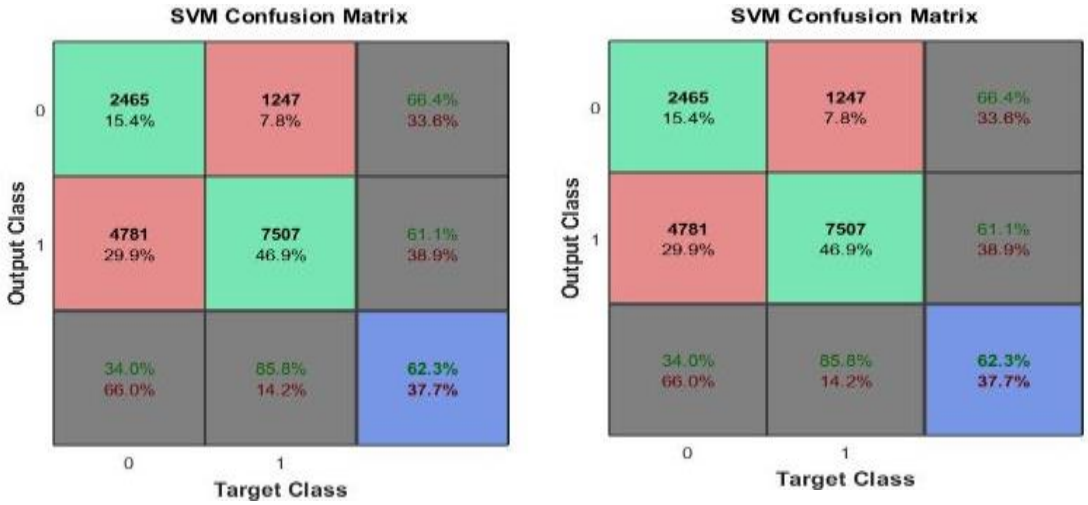

Figure 4: *Confusion matrices, of best MLP and SVM models, for the training data.*

[7] Classes in confusion matrix represented as [0 1], as opposed to [-1 1].

5.3 Discussion

Parameter tuning required the most time and effort. SVMs were tuned via a regularization parameter – box constraint, and kernel scaling parameter- sigma. MLPs on other hand, had five. Interestingly, the absence of momentum improved the model, which was unexpected and could potentially be a result of a flat and nondeterministic error surface. Also, learning decay required a very high value, suggesting a high sensitivity to neuron saturation. However, this list of parameters is not exhaustive, one could experiment with other weight and bias initialization functions, add extra regularization methods (e.g. dropout). Also, our MLP only had one hidden layer, even with the universal approximation rule, it is likely that extra layers could add some delicacy in pattern recognition, which might have led it to be more comparable with SVM. Then, cost and benefit analysis would need to be done, to investigate whether additional time, and computational requirements are worth it.

Despite having truly powerful generalizability, and flexibility properties, both algorithms have some substantial drawbacks. First of all, both are relatively slow: MLP due extensive tuning; SVM due to consequences of global optimization. Also, they both are tough to interpret. In our example, it is extremely challenging to investigate what effect did touch time, or defenders distance have on the outcome of the shot. Classic models, such as logistic regression, have advantage in this domain. Also, Random Forests, and some derivations of it, share the neural networks ability to discover non-linear relationships, but also provides feature ranking via entropy criteria, and even logistic regression-like feature coefficients. (Saabas, 2016)

6. CONCLUSION

In this paper we evaluated two deep learning algorithms: Support Vector Machines (SVM) and Feed forward Multilayer Perceptrons trained with backpropagation algorithms. We have evaluated both models on accuracy and efficiency and determined that although, in total, MPL took longer to run, on average, SVM iterations took longer than MPL iterations. We have also determined that the learning curve for SVM is much smoother. Overall, SVM achieves a greater accuracy than MPL, although by a small margin.

We have learned that finding the right parameters for MPL can be very time consuming and can take a lot of trial and error, as 1,025 different combinations (that ran for almost 7.5 hours) did not give sufficient results and ultimately, the optimal MPL parameters had to be found manually (which took an additional 4-5 hours).

We have outlined that it is almost impossible to determine which factors have the most impact on the outcome of the shot, perhaps this study could be replicated by using traditional machine learning algorithms (e.g. logistic regression). It would also be interesting to run the study with a different kind of artificial neural network, for example, recurrent neural network, which would be tuned to analyse chunks in time, and look for evidence of 'hot hand'. [8]

We have used NBA 2015 data in order to run the study. We have treated both 2 point and 3 point shots in the same manner. Perhaps model accuracy could have been improved by treating these separately - as overall 2 point shots are more likely to be made than 3 point shots, making 3 point shots more susceptible to miss-classification. The dataset that we have obtained had distance recorded in radius from the basket rather than the exact location in the arena. Having these parameters are likely to have also improved the chance of classifying shots accurately.

[8] Hot hand is wide held belief that sequence of made shots, would positively influence the probability of the next shot.

7. REFERENCES

Official NBA release (2013) *NBA partners with stats LLC for tracking technology*. Available at: http://www.nba.com/2013/news/09/05/nba-stats-llc-player-tracking-technology.

Partnow, S. (2016) *MoreyBall, Goodhart's law, and the limits of Analytics*, VICE sports. Available at: https://sports.vice.com/en_us/article/moreyball-goodharts-law-and-the-limits-of-analytics.

Shah, R.; Romijnders, R, (2016) *Applying Deep Learning to Basketball Trajectories*. CoRR abs/1608.03793

Wang, K.C.; Zemel R. *Classifying NBA Offensive Plays Using Neural Networks* MIT Slolan Sports Analytics Conference 2016

Bernard Loeffelholz-Earl Bednar-Kenneth Bauer *Predicting NBA Games Using Neural Networks* - Journal of Quantitative Analysis in Sports – 2009

B. Markoski, P. Pecev, L. Ratgeber, M. Ivkovic and Z. Ivankovic, "Appliance of neural networks in Basketball - Basketball board for Basketball Referees," *2011 IEEE 12th International Symposium on Computational Intelligence and Informatics (CINTI)*, Budapest, 2011, pp. 133-137. doi: 10.1109/CINTI.2011.6108486

Kaggle.com. (2016). NBA shot logs | Kaggle. [online] Available at: https://www.kaggle.com/dansbecker/nba-shot-logs.

Gardner, M. and Dorling, S. (1998). Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. Atmospheric Environment, 32(14-15), pp.2627-2636.

Bernack M.(2005) *Backpropagation [online*] Home.agh.edu.pl . Available at http://home.agh.edu.pl/~vlsi/AI/backp_t_en/backprop.html

Warren McCulloch and Walter Pitts, "A Logical Calculus of Ideas Immanent in Nervous Activity", 1943, *Bulletin of Mathematical Biophysics* 5:115–133.

Rumelhart, David E.; Hinton, Geoffrey E.; Williams, Ronald J. (8 October 1986). "Learning representations by back-propagating errors". *Nature*. **323** (6088): 533–536.

Pineda, F. J. (1987). Generalization of back-propagation to recurrent neural networks. Physical Review Letters, 59(19), 2229–2232. http://doi.org/10.1103/physrevlett.59.2229

Saabas, Ando. "Random Forest Interpretation with Scikit-learn." *Diving into Data*. N.p., 12 Aug. 2015. Web. 13 Dec. 2016

Caruana, R. and Mizil, A. (2006). An Empirical Comparison of Supervised Learning Algorithms.

Collobert, R. and Begio, S. (2004). Links between Perceptrons, MLPs and SVMs.

Haykin, S. (1998). *Neural Networks: A Comprehensive Foundation*. 2nd ed. Prentice Hall PTR Upper Saddle River, NJ, USA ©1998.

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection.

Lecun, Y., Bottou, L., Bengio, Y. and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11), pp.2278-2324.

Hsu, C., Chang, C. and Lin, C. (2003). A Practical Guide to Support Vector Classification.