Assignment 1, Deep Learning Fundamentals, 2020
Perceptron implementation and testing on the Pima Diabetes dataset

XXXX
XXXX
XXXX
XXXX

Abstract

Classification Machine Learning algorithms take a input vector and classify it in discrete classes. One of the first classifications algorithms to be invented, and one of the simplest neural networks, is the Perceptron, that deals with binary classification problems by adjusting a set of weights to find an optimum hyperplane to separate the classes. In this research, a classic single-layer Perceptron algorithm was implemented to create a diabetes classification model based on patient data, using the Pima Indians diabetes dataset. Moreover, the single-layer Perceptron was improved by adding a hidden layer, using sigmoid functions and implementing a back propagation algorithm to train the weights. The results of these algorithms were compared with the Support Vector Machine (SVM) and Random Forest (RF) model performances. This comparison yielded that the SVM outperformed all the other method (overall accuracy=0.7857), being followed by the multi-layer Perceptron (overall accuracy=0.7532), although the latter detected slightly more True Positives, fact that can be relevant in the context of disease prediction. The inclusion of an extra layer in the Perceptron algorithm increased its performance, showing that in this case, a complexity gain in the neural network can produce better classification performance.

1. Introduction

In the field of Pattern recognition and machine learning, classification problems are one of the most common tasks. In these kind of problems, an algorithms takes a vector X as input, and assigns it to one or mane K discrete classes [4]. Considering that the labeled data take values in a discrete set, the input space can be divided in regions that can be labeled according to the classification. Boundaries of these regions can be smooth or rough, depending on the function applied by the algorithm [6]. When lineal functions are used to estimate these decision boundaries or surfaces, we can refer to them as part of the ”linear models for classification” family of algorithms. While, data sets whose classes can be separated by lineal functions, are often called to be lineally separable [4]

The Perceptron algorithm is one of the first linear models for classification, invented by Rosenblatt [10], is often considered one of the first neural networks. The Perceptron deals with binary (two class) classification problems [4], by adjusting a set of weights to find a separating hyperplane in the training data, by minimizing the distance of misclassified data to the decision surface or boundary [6].

In the field of classification problems, the diagnosis or determination of he risk of diseases based on clinical data is a recurrent field of study. These classification algorithms can help decision makers to predict the patient outcome based on the patient data [3], making appropriate and well-timed decisions. One of the diseases that has been extensively researched in this field is the Diabetes Mellitus ()DM), this chronic condition is characterized by high levels of glucose in the blood, caused either by the failure of the pancreas to produce enough insulin or the inefficiency of the body to efficiently use it [13].

The objective of this research is to implement the Perceptron algorithm and to test if it can effectively predict the Diabetes Mellitus disease based on patient data, using the PIMA diabetes dataset. Additionally, the inclusion of a hidden layer to implement a multi-layer Perceptron (MLP) is proposed to be tested. Finally, this research aims to compare the single layer and multi-layer Perceptron performances against other machine learning algorithms, such as Support Vector Machines (SVM) and Random Forest (RF).

2. Related work

The prediction of medical diseases is one of the fields where different machine learning methods has been extensively tested, with an special emphasis on the prediction of Diabetes Mellitus, where several models have been implemented [14, 1, 11, 13], as an early diagnosis of the disease can be beneficial for the patient.

Multiple different machine learning methods can be found in the literature on the prediction of Diabetes Mellitus. As an example, Santhanam & Padmavathi [11] used K-means and genetic algorithms to reduce the dimension of the dataset to integrate it into a Support Vector Machine(SVM) algorithm, archiving 96.71% accuracy in the PIMA Indians Diabetes dataset, while Ahmad et al [1] used a hybrid Genetic Algorithm together with a multi-layer Perceptron to archive 80.4% accuracy o the same dataset.

Using a different dataset from the University of Virginia, Xu et al [14] used a random forest model to predict the risk of Diabetes Mellitus based on readily available indicators (age, waist, weight etc), archiving 84.13% of accuracy when sufficient data is provided.

3. Methods

3.1. Pima Indian Diabetes Dataset

The Pima Indian Diabetes dataset consists in the information of 768 patients from a population close to Phoenix, Arizona, USA [13], all of the patients are woman over the age 21 [14] belonging to the Pima indiginous community. In this dataset, 268 (34.9%) of the patients tested positive for Diabetes Mellitus (DM), while 500 (65.1%) tested negative. Each row of data contains 8 attributes [13], apart from the classes label (Positive DM or Negative DM):

- Number of pregnancies
- Glucose concentration in a oral tolerance test
- Skin fold thickness in the triceps
- Body mass index (BMI)
- Serum insulin (2h)
- Diastolic blood pressure
- Diabetes pedigree function
- Age of the patient

3.2. Data preprocessing

When machine learning algorithms are applied, considerable performance problems in the predictions can be attributed to low data quality, especially in clinical data [8]. Given this background, basic data preprocessing was applied to increase the performance of the models. Therefore, following the recommendations of Wu **et al** [13], it was noted that the features ”Diastolic Blood Pressure” and ”Body mass index (BMI)” presented zero values, that were interpreted as missing data, as the human body can not present these values. Thus, as in We **et al** [13], the zero values were replaced by the mean of the valid observations in these two variables. After the handling of these missing values, the dataset was scaled to have values between -1 and +1 in all the features, using a standard lineal transformation:

$$x' = (b-a) \times \frac{x - min(x)}{max(x) - min(x)} + a \tag{1}$$

where b is the new maximum (in this case +1), a in the new minimum (-1), x is the original feature and x' is the transformed feature. The output label (the diabetes diagnosis) was also transformed to -1 (Negative DM) and +1 (Positive DM), to comply with the requirement of the Perceptron algorithm

3.3. Single-layer Perceptron implementation

The Perceptron is a a two class classifier that uses the input vector a set of weights w, over the input vector x (Figure 1), that can be previously transformed [4] (as in this case). These elements are used to construct a linear model of the form:

$$y(x,w) = f(w^T x) \tag{2}$$

Then, a nonlinear activation function is used to transform this numerical output a into a discrete label using the following sign function [4]

$$f(a) = \begin{cases} +1, & \text{if } a \geq 0 \\ -1, & \text{if } a < 0 \end{cases} \tag{3}$$

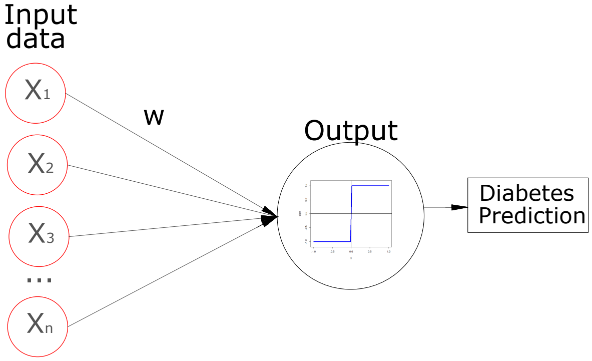

Figure 1. Single layer Perceptron

The weights of the Perceptron algorithm are iteratively trained using the following equation, that evaluate the predictions for each sample i and compare it with the true label, using gradient descent to adjust the weights for each feature

According to Ripley [9], the Perceptron algorithm will perform poorly in the case of a non linearly separable dataset, as optimizing the number of error made by this algorithm is hard. Also, in nonlinear separable problem, this algorithm may not converge in a fixed set of weights, being dependent in the number of iterations in the training.

$$w_(t+1) = w_t + \eta \sum_{i=1}^{n} y_i x_i Error_i \tag{4}$$

$$Error_i = \begin{cases} 1, & \text{if } predicted_i \neq y_i \\ 0, & \text{if } predicted_i = y_i \end{cases} \tag{5}$$

Where w is the vector of weights, y_i are true label in the dataset and x_i is the input value of the feature

In this case, the initial weights (w) were generated using random numbers coming from a normal distribution of mean = 0 and standard deviation = 0.1

3.4. Multi-layer Perceptron implementation

In order to improve the single layer Perceptron, a hidden layer with a variable number of layers was implemented in between the Output and the Input layer. Also, in order to reflect non linear boundaries [12], a sigmoid function of the following form was used to activate the hidden and output layer:

$$sigmoid(x) = \frac{1}{1 - e^{-x}} \tag{6}$$

As we can see in Figure 2, the multi-layer Perceptron the input data (x) feed the hidden layer nodes by appling a $w1$ set of weights, just as in the single layer perceptron, then, these hidden layer nodes are activated with the above mentioned sigmoid equation, feeding the output layer though the application of the $w2$ set of weights. This output layer is also activated by a sigmoid function. Finally, the diabetes prediction is determined by a threshold of 0.5 in the final output, as the sigmoid function generates values from 0 to 1. Considering this particularity, for this specific case, the label values were encoded as 1: Positive DM and 0: Negative DM.

In this case, the algorithm has two sets of weights ($w1, w2$), the first one correspons to a matrix of dimensions $[numberoffeatures, numberofhiddenlayernodes]$, while $w2$ is a vector of length equal to the number of hidden layer nodes. As explained by [12], in order to train them, a iterative back-propagation workflow has to be implemented, consisting in three steps

(i) Forward pass

First each hidden layer node is activated using the sigmoid function, getting the values from the input values multiplied by the set of $w1$ weights corresponding to the hidden layer h_i

$$h_i(x, w1) = S(w1_i^T x) \tag{7}$$

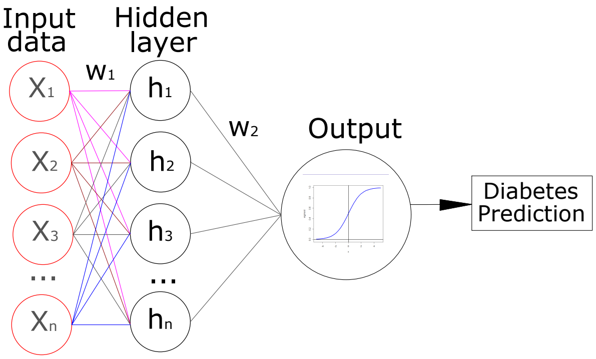

Figure 2. Multi-layer Perceptron

Then the output layer is activated using the input from the hidden layers, by multiplying the hidden post activation values with the second set of weights ($w2$)

$$o(h, w2) = S(w2^T h) \tag{8}$$

(ii) Backward pass

The backward pass computes the error in the output layer and then passes those values to the previous layer, in this case the hidden layer

$$b(output) = o(h, w2) - y_i \tag{9}$$

$$b_i(hidden) = b(output) \nabla S(w2^T h) * x2_i \tag{10}$$

iii) Weight update

In order to update the weights, a step real positive value(η) is defined, and the backward passes of the respective layers are used to update each set of weights

$$w1_i(t+1) = w1_i(t) - \eta \sum_{i=1}^{n} b_i(hidden) x_i w1_i \nabla S(w1^T x_i) \tag{11}$$

$$w2_i(t+1) = w1_i(t) - \eta \sum_{i=1}^{n} b_i(output) h_i w2_i \nabla S(w2^T h_i) \tag{12}$$

3.5. Performance metrics

The dataset under study corresponds to an unbalanced dataset (there is much more negative DM than nositive DM), and as such, some considerations have to be made in order to correctly compare methods and asses the performance of the models [7]. therefore, two different metrics were used for comparison

Accuracy: Although this metric can be misleading in the case of unbalanced datasets, it is the most commonly used metric [7] and can help compare the overall performance of the method:

$$accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{13}$$

where TP is the number of true positive, TP: true negative, FP: false positives and FN: false negatives.

In order to use a metric than can asses the performance in this unbalanced problem, the F1 score was used, as it balances the precision and recall of the algorithm, being a metric that has useful properties in unbalanced cases [7]

$$precision = \frac{TP}{TP + FP} \tag{14}$$

$$recall = \frac{TP}{TP + FN} \tag{15}$$

$$F1 = 2\frac{precision * recall}{precision + recall} \tag{16}$$

3.6. Training, parameter tuning and testing

In order to obtain the best possible model to predict the occurrence of Diabetes Mellitus (DM) in future data, the complete dataset was partitioned into training (80%) and testing data (20%), as the data is imbalanced (it contains 34.9% of samples with positive DM), a stratified sampling was used to ensure that this proportion is approximately maintained in the training and testing data.

The single-layer Perceptron presents two parameters that can impact the performance of the model: the step value and the number of iterations (epochs) in the training. In order to pick the best possible values, a grid search was implemented. Therefore, the training data was split into 4 groups using stratified sampling to maintain the proportion of positive and negative labels. Then 4-fold cross validation was used to test all the possible combinations of values for these parameters in the following vectors: $step = (0.1, 0.2, 0.3...1.0)$ and $n_{\text{iterations}} = (10, 20, 30...100)$

In each of the parameter combinations, the F1 index was calculated in order to asses the performance of the model, and the obtained values were averaged between the 4-fold operations, giving a mean F1 value for each combination of parameters

Once all the possible combinations of parameter values were tested, the combination of parameters with the biggest F1 score was picked. These parameters were used to where the above mentioned metrics were calculated to asses the model performance.

In the case of the multi-layer Perceptron, the parameter tuning also included the testing of different number of nodes in the hidden layer, this, the grid search included the following values: $step = (0.1, 0.3, 0.5, 0.7, 0.9, 1.1)$, $n_{\text{iterations}} = (10, 30, 50, 70, 90, 110)$ and $nodes_{\text{hiddenlayer}} = (3, 4, 5)$

3.7. Comparison with other machine learning methods

In order to asses the performance of the Perceptron models against other well-established machine learning models. Following the related literature [14, 11], a Random Forest classifier and a Support Vector Machine model were tested to predict Diabetes Mellitus (DM), using the same pre-processing steps and score metrics than those used in the Perceptron implementations

Random Forest: This bagging method was first introduced by Breiman [5], and consists in the a large collection of de-correlated decision trees, using a user-defined number of trees (n_{trees}), the algorithm draws a bootstrap sample of the training data, to grow a decision tree of the subset data, selecting a random number of variables in each split (max_{features}), to then determine the best split point among the drawn variables, generating two child nodes. After all the trees are grown, forming a *Random Forest*, new testing data can be predicted by obtaining a class vote from each decision tree in the forest, and then classifying using majority vote [6].

According to Hastie [6], Random Forest algorithms can perform remarkably well with very little tuning, while Breiman [5] claims that their results can be highly interpretable. Moreover, this model is quite robust in the presence of noisy variables, but can perform poorly with a small max_{features} [5]

In this case, the parameters that were tuned using k-fold cross validation were the maximum number of features in each split(max_{features}), and the number of trees (n_{trees}). In this case, the vectors for the grid search were: $max_{\text{features}} = (1, 2, 3...8)$ and $n_{\text{trees}} = (100, 300, 500...2100)$

Support Vector Machine: This supervised classification algorithm was developed by Vapnik [12], and consist in an hyperplane that separates a high dimensional space defined by the input variables into discrete classes. This hyperplane is defined to have the largest possible distances to the closest point of either class [6], thus, maximizing the margin (M) between two classes.

$$Max\, M \quad constraned\ by\ y_1(x_i^T\theta + \theta_0) \geq M \tag{17}$$

This algorithm can define an hyperplane even when classes overlap (non linearly separable), maximizing the margin but allowing some points to be in the wrong side of the boundary, defining the slack variables xi_i, thus, the margin constrain for defining the hyperplane can be modified as [6]

$$y_1(x_i^T\theta + \theta_0) \geq M(1 - \xi_i) \quad (18)$$

if we redefine M as $M = 1/||\theta||$, we can write the previous equation as [6]:

$$min\frac{1}{2}||\theta||^2 + C\sum_{i=1}^{n}\xi_i \quad (19)$$

This equation can be then solved using Lagrange multipliers [6]

$$L_p = \frac{1}{2}||\theta||^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i[y_1(x_i^T\theta + \theta_0) - (1 - \xi_i)] - \sum_{i=1}^{n}\mu_i\xi_i \quad (20)$$

Although this algorithm initially use lineal boundaries between the different classes, it can be defined by other kinds of boundaries form, using a ”kernel trick” (more details in Hastie [6])

Support vector machine are widely used in the medical sciences, and can perform greatly in image classification and other pattern recognition tasks. Although some practitioners have criticized SVM method for being difficult to interpret and explain. This kind of method is commonly compared with neural networks, as both use sigmoid functions and the finding of optimal hyperplanes [2]

In this case, the tuning parameters were the cost and the type of kernel, that were tested in the cross validation stage, using grid search with $cost = (0.5, 1.0, 1.5...10)$ and $kernel = (linear, polynomial, radial, sigmoid)$).

4. Code

The code, along with the requirements and setup instructions for the reproduction of the above-mentioned method, can be found in the following GitHub repository: `https://github.com/juliancabezas/deep_learning_perceptron`

5. Results and experimental analysis

5.1. Parameter tuning

The k-fold testing of different parameters gave the result of Table 1, where it is possible to see that the RF shows a better F1 score using the maximum number of trees that was trained, and a multiple number of features of 4 (half of the dataset). On the other hand, the single layer and the multi-layer Perceptron show similar performances (0.6455 and 0.6377 respectively), with the single layer Perceptron showing the best performance with 20 iterations to train the weights, while the multi-layer Perceptron shows 70, as the procedure to train them is more complex. Finally, the best performance of the Support Vector Machine (SVM) model was archived with a lineal kernel an a cost of 3.0.

Algorithm	Tuned parameters	F1
Perceptron	$n_{\text{iter}} = 20$ & $step = 0.7$	0.6455
MLP	$n_{\text{iter}} = 70$, $step = 0.9$ & $nodes_{\text{hidden}} = 5$	0.6382
SVM	$cost = 3.0$ & $kernel = linear$	0.6377
RF	$max_{\text{features}} = 4$ & $n_{\text{trees}} = 1100$	0.6751

Table 1. Parameter tuning (MLP = Multi-Layer Peceptron, SVM = Support Vector Machine, RF = Random Forest)

5.2. Model testing

The results of the application of the different models in the test dataset 1 show that the best results in terms of the F1 score, that generates a compromise between recall and precision, are presented in the single-layer Perceptron (0.65 against 0.64 in the multi-layer Perceptron and the SVM models). While comparing the accuracy, the Support Vector Model (SVM) clearly outperforms the Perceptron models and the Random Forest (RF) model, generating an overall better model. The poor accuracy of the single layer Perceptron, that is just slightly bigger that the trivial solution (classifying all samples as negative DM) can be attributed to the fact that the data is non linearly separable [9]

Regarding the Perceptron models, the inclusion of the hidden layer containing 5 nodes and the use of the sigmoid function to activate them generates significant improvements in the accuracy (0.70 to 0.75) and Kappa values (0.41 to 0.45), showing that in this case, increasing the complexity of the neural network can be beneficial for its prediction capacity

Algorithm	F1	Accuracy
Perceptron	0.6565	0.7078
MLP	0.6415	0.7532
SVM	0.6451	0.7857
RF	0.6111	0.7272

Table 2. Model performance in the test data (MLP = Multi-Layer Perceptron, SVM = Support Vector Machine, RF = Random Forest)

As the F1 values are relatively similar between the single-layer Perceptron, the MLP and the SVM, we can determine, by looking at the accuracy, that the two best performing models are the MLP and the SVM, with the SVM obtaining an accuracy 3% greater than what was obtained by the SVM. Altough, when looking at the details of the confusion matrix of these two algorithms, the SVN appears to detect a great percentage of the negative cases (91%), while on the other hand, missing a significant number of positive cases (44%) (Figure 3). In the case of the diagnosis of diseases, this consideration can have a great impact in the utility of the model, preventing the decision makers to perform early diagnosis of this disease.

		Prediction		
		Positive DM	**Negative DM**	**Total**
Label value	**Pos. DM**	TP=30	FN=24	54
	Neg. DM	FP=9	TN=91	100
	Total	39	115	154

Figure 3. Confusion matrix for the Support Vector Machine model

On the other hand, the multi-layer Perceptron, although producing a lower overall accuracy, is able to detect a larger value of positive cases (63%), but still missing 37% of the positive cases (Figure 4)

		Prediction		
		Positive DM	**Negative DM**	**Total**
Label value	**Pos. DM**	TP=34	FN=20	54
	Neg. DM	FP=18	TN=82	100
	Total	52	102	154

Figure 4. Confusion matrix for the multi-layer Perceptron model

These results indicate that neither of this models could be used in a medical environment, as could mislead the practitioners into missing the diagnosis of a potentially serious disease.

Although the performance of the tested models was far from ideal (specially in the medical field), this research showed that the inclusion of a simple hidden layer can generate a simple multi-layer Perceptron model that outperformed more modern models as Random Forest, and can compete with well tested models as SVM. Giving the insight that the inclusion of more complexity in the model (more layer and/or nodes) could lead to a increased performance.

6. Conclusion

The results of this experiment shows that the single-layer Perceptron, even being a relatively old and simple model, can give results that can be comparable with newer models such as random forest, while clearly being outperformed by other more complex models as SVM. The increase in performance of the Perceptron when adding an extra layer, shows the beginning of the developing of deep neural networks, that benefit from the inclusion of multiple layers to solve non-linear and complex problems.

We can note, with this research, that although relative ”classic” machine learning algorithms, such as Random Forest or Support Vector Machine, can be easy to implement, tune and interpret, relatively simple neural networks can easily compete with them, and that the more modern one will probably outperform them, showing that the future of machine learning will be seriously influenced by deep neural networks.

In future research, extra hidden layers, with different activation functions, can be tested to increase to generate a deeper multi-layer Perceptron, possibly increasing the accuracy of the prediction, generating useful models for the medical community and taking advantage of the modern computational capacities.

References

[1] Fadzil Ahmad, Nor Ashidi Mat Isa, Zakaria Hussain, and Muhammad Khusairi Osman. Intelligent medical disease diagnosis using improved hybrid genetic algorithm - Multilayer perceptron network. *Journal of Medical Systems*, 37(2), 2013.

[2] Jason Bell. *Machine Learning : Hands-On for Developers and Technical Professionals*. John Wiley & Sons Inc., Indianapolis, IN, USA, 1 edition, 2014.

[3] Riccardo Bellazzi and Blaz Zupan. Predictive data mining in clinical medicine: Current issues and guidelines, 2008.

[4] Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer, New York, NY, USA, 2006.

[5] L Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[6] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. 2009.

[7] Ryan Hoens and Nitesh V. Chawla. Imbalanced Datasets: From Sampling to Classifier. In Haibo He and Yunqian Ma, editors, *Imbalanced Learning: Foundations, Algorithms and Applications*, chapter 3, pages 44–107. Wiley, Hoboken, New Jersey, USA, 2013.

[8] Arnaud J.F. Installé, Thierry Van Den Bosch, Bart De Moor, and Dirk Timmerman. Clinical data miner: An electronic

case report form system with integrated data preprocessing and machine-learning libraries supporting clinical diagnostic model research. *Journal of Medical Internet Research*, 16(10):e28, 2014.

[9] B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge University Press, Cambridge, UK, 1996.

[10] F. Rosenblatt. The Perceptron - A Perceiving and Recognizing Automaton, 1957.

[11] T Santhanam and M S Padmavathi. Application of K-Means and Genetic Algorithms for Dimension Reduction by Integrating SVM for Diabetes Diagnosis. *Procedia - Procedia Computer Science*, 47:76–83, 2015.

[12] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer, New York, NY, USA, 1995.

[13] Han Wu, Shengqi Yang, Zhangqin Huang, Jian He, and Xiaoyi Wang. Type 2 diabetes mellitus prediction model based on data mining. *Informatics in Medicine Unlocked*, 10(August 2017):100–107, 2018.

[14] Weifeng Xu, Jianxin Zhang, Qiang Zhang, and Xiaopeng Wei. Risk prediction of type II diabetes based on random forest model. In *Proceedings of the 3rd IEEE International Conference on Advances in Electrical and Electronics, Information, Communication and Bio-Informatics, AEEICB 2017*, pages 382–386. IEEE, 2017.