Assignment 1: Predict diabetes via Perceptron

Moaz Mohamed
The University of Adelaide
Adelaide SA 5005
a1779177@student.adelaide.edu.au

1. Introduction

Over the last few years, deep learning had a significant impact in the medical field. The application of numerous Deep learning algorithms have been implemented in different sector of the medical field. In this paper Perceptron algorithm will be explored and its performance will be investigated on classification task to predict diabetes on the provided dataset.In addition to a complete review of the Perceptron algorithm describing its strength and weakness. An alternative algorithm will be introduced and explored that can mitigate some of the inherit weakness of Perceptron.

2. Methodology and Background

2.1. Perceptron

The Perceptron algorithm was invented by Frank Rosenblatt at Cornell university. [1] Perceptron is a supervised linear classifier. In binary classification the Perceptron will find a hyperplane that is able to separate the two classes assuming that the two classes are linearly separable.[2] [3]

the idea behind the perceptron was inspired by the neurons in the brain work. hence a neuron will only fire at the signal reaches a certain threshold. in turn the perceptron emulates such mechanism.

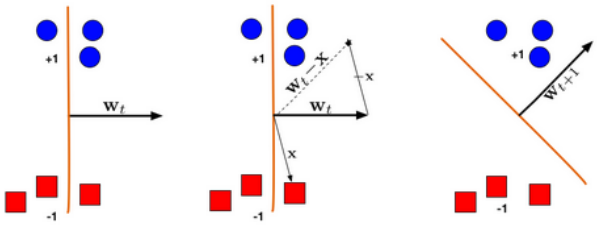

Figure 1: Perceptron update

The classification of a class depends on the data point location from the hyperplane. if its same direction as $\overrightarrow{w}$ or in opposite direction and a hyperplane is created that is Orthogonal to $\overrightarrow{w}$ hence the hyperplane becomes the decision boundary. the classification is characterized by the following equation $y_i\left(w_i^T x_i + b\right)$. assuming the binary outcomes are $(1, -1)$ in both cases for the correct classification the previous terms should be $y_i\left(w_i^T x_i + b\right) \geq 0$. the idea behind the learning of a perceptron is to continuously update the $\overrightarrow{w}$ until all data points are correctly classified by $w_{i+1} = w_i + \eta\left(t_i - o_i\right) x_i$ as alluded to in figure 1.

for easier implementation the bias term can be absorbed as such

$$\begin{bmatrix} w \\ b \end{bmatrix}^T \begin{bmatrix} x_i \\ 1 \end{bmatrix} \tag{1}$$

which allows for a straight forward update for the perceptron. Gradient Descent can also be utilized in the original perceptron algorithm by allowing by having an linear activation function instead of unite step function(cite).

2.2. Support Vector Machines

An alternative approach to pattern recognition is using Support Vector Machines(SVM). Which in hindsight very similar to the perceptron algorithm but SVM ensures finding a hyperplane taking into consideration the finding the hyperplane that can separate between the two classes with maximum margin. By finding the set of points x_j to the boundary $|w^t x_j + w_0| = 1$ hence the distance from the closet sample x_j to $g(x)$ is

$$\frac{|w^t x_j + w_0|}{\|w\|} = \frac{1}{\|w\|} \tag{2}$$

where $g\left(x\right) = w^t x + w_0$
thus the margin is

$$m = \frac{2}{\|w\|} \tag{3}$$

which can be constructed as a quadratic function to minimize

$$j\left(w\right) = \frac{1}{2} \times \|w\|^2 \tag{4}$$

constraint to

$$y_j\left(w^T x_j + w_0\right) \geq 1, \forall j \tag{5}$$

Such formulation does work on finding the optimal hyperplane and it can be solved with quadratic optimization tool such as CVXOPT OR MOSER. a slack variable can also be added to relax the SVM with C as regularization parameter.

minimization of

$$\|w\|_2^2 + c \times \frac{1}{n} \sum_{j=1}^{n} \xi_j \tag{6}$$

subject to

$$\begin{aligned} y_j \left(w^T X_j + w_0 \right) &\geq 1 - \xi j \\ \xi_j &\geq 0, \forall n \end{aligned} \tag{7}$$

Kuhn-Tucker theorem and Lagrange multipliers can be utilized to drive the previous problem from primal problem to dual problem.

$$L_D(\alpha) = \sum_{i}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j < X_i^T X_j > \tag{8}$$

subject to

$$\alpha_j \geq 0 \forall j, \sum_{j=1}^{n} \alpha_j y_j = 0 \tag{9}$$

and the $\overrightarrow{w}$ and the bias term can be calculated as follows.

$$W = \sum_{j=1}^{n} \alpha_j y_j x_j \tag{10}$$

$$w_0 = \frac{1}{y_j} - w^T x_j \tag{11}$$

It is interesting to note that in equation 10 the weight vector can be formulated as summation of alpha, label and the support vector. instead of in the primal problem where the $\overrightarrow{w}$ is obtained directly from solving the quadratic optimization problem. Such difference allows for the usage for something called the "Kernel Trick".

$$L_D(\alpha) = \sum_{j=1}^{n} \alpha_j - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j K(x_j, x_i) \tag{12}$$

$$K(x_j, x_i) = \left(x_i^T x_j + 1 \right)^p \tag{13}$$

$$K(x_i, x_j) = \exp\left(-\frac{1}{2\sigma^2} \times \|x_j - x_i\|^2 \right) \tag{14}$$

the optimization problems for both equations 12 and 8 except for the (x_j, x_i) in equation 12 is being applied

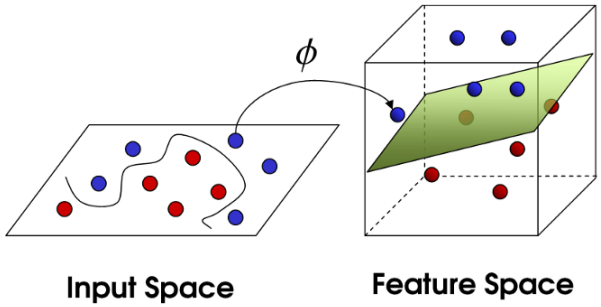

Figure 2: Mapping

to a kernel and the data is being projected into higher dimensions. here the SVM is exploiting cover's theorem."A complex pattern-classification problem, cast in a high-dimensional space non-linearly, is more likely to be linearly separable than in a low-dimensional space, provided that the space is not densely populated." [4] Which increases the complexity of the SVM by applying the kernel trick and enabling it to classify classes even in non-separable cases as demostrated in figure 2

3. Experimental Analysis.

3.1. Perceptron

The perceptron algorithm does converge on hyperplane that the two classes. but it doesn't converge on hyperplane that separate the two classes with the biggest margin between the two classes. in addition the algorithm inherit inability to classify class in non-linearly separable data as demonstrated in figure 3 and 4. In-fact the algorithm will loop forever assuming no criteria for stoppage wasn't added.

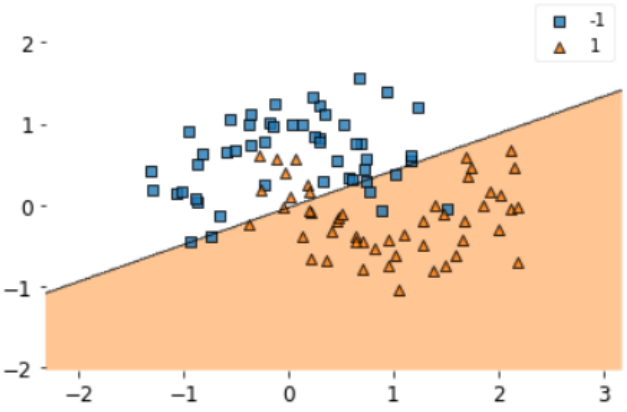

Figure 3: Perceptron decision boundary in linearly separable data

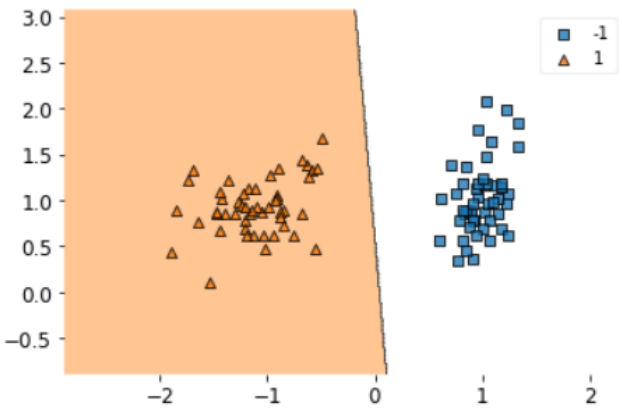

Figure 4: Perceptron decision boundary in non-linearly separable data

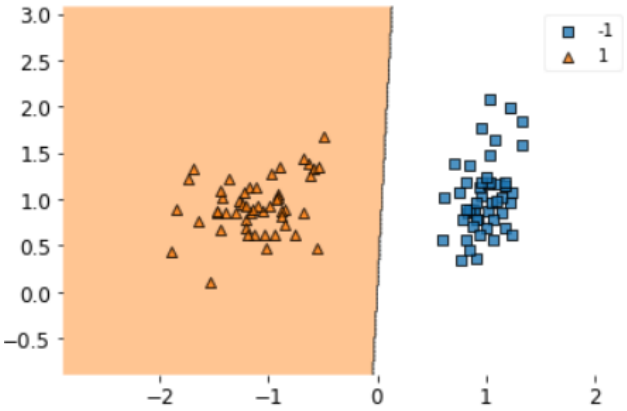

Figure 6: SVM decision boundary in linearly separable data

3.2. SVM Primal and Dual

By utilizing the support vectors in eq 10 SVM is able to find the support vectors that can maximize the margin between the two classes as demonstrated in equation 8 but SVM without utilizing a kernel SVM has the same weakness as Perceptron in that regard. But when a kernel is utilized and project the data into higher dimensions SVM is capable of finding a hyperplane between the two classes as demonstrated in figure 7.

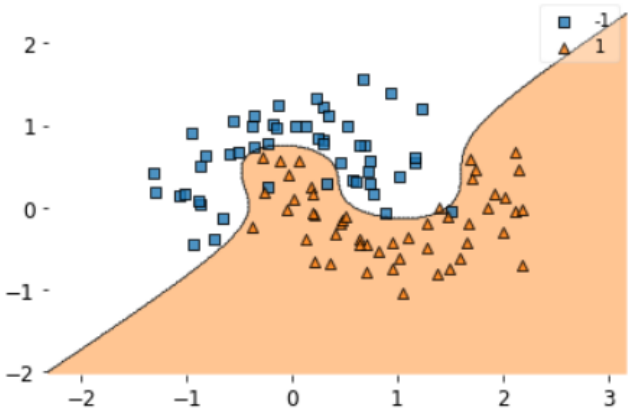

Figure 7: SVM with polynomial kernel applied.Decision boundary in non-linearly separable data

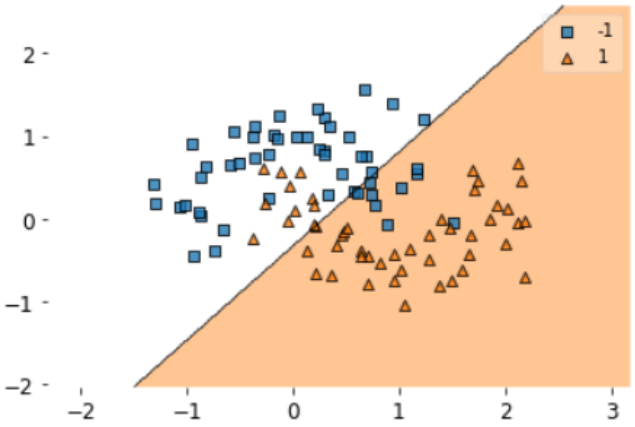

Figure 5: SVM decision boundary in non-linearly separable data

4. Performance on diabetes data-set

In table 1 are the results. with aggressive hyperparameter tuning all algorithms did perform with no apparent close contender. the diabetes dataset is full of missing values and noise with careful data cleaning steps i expect better results.

Algorithm	Accuracy	precision	F1
Perceptron	72%	60%	57%
SVM	73%	63%	58%
SVM + Polynomial	72%	62%	57%
SVM + Gaussian	71%	58%	58%

Table 1: Results

5. Conclusion

Perceptron and different variations of Support vector machines algorithm have been explored and analyzed. mathematical formulation of all the algorithms have been studied. decision boundaries for have been made for a careful study of the behavior of various algorithms in different data situations. Advantageous and weakness have been identified for all the explored algorithms. Perceptron and its creator have created and paved the way for the emergence of the field of machine learning. all subsequent algorithms builds on top of the weakness of the previous work as seen in perceptron, SVM and later kernalised SVM.

6. Code

all the code for this assignment is provided in this GitHub repo `https://github.com/RoastedKernel/uni_Machine_Learning`

References

[1] F. Rosenblatt, "The perceptron: A probabilistic model for information storage and organization in the brain," *Psychological Review*, vol. 65, no. 6, pp. 386–408, 1958.

[2] C. Murphy, P. Gray, and G. Stewart, "Verified perceptron convergence theorem," *MAPL 2017 - Proceedings of the 1st ACM SIGPLAN International Workshop on Machine Learning and Programming Languages, co-located with PLDI 2017*, pp. 43–50, 2017.

[3] M. Minsky and S. Papert, "Perceptrons: expanded edition," *MIT Press Cambridge MA*, vol. 522, p. 20, 1969.

[4] M. Cover, "Inequalities Applications Pattern," pp. 326–334, 1965.