Machine learning lab assignment 1 Linear-threshold classifiers

Kevin Serrano, *EMARO+, UNIGE*

***Abstract*—This lab report introduces the concept of linear classifier and its performance on different 2D binary problems. A manual design approach is taken in the sens that there is no algorithm used for determining the decision boundary. Instead, a graphical trial-and-error method is used. Further analysis for robustness against random noise are presented and discussed.**

***Keywords*—linear, classifier, machine learning.**

I. Introduction

Classification is one of the most widely used techniques in machine learning with a broad scope of applications, including spam detection, image classification, medical diagnostic and risk assessment. The main goal is to predict a certain class y from a set of inputs X. Linear classifiers are amongst the most practical classification methods.

Maximum likelihood estimation (MLE) and gradient descent are the most widely known approaches for linear classification. Nonetheless, this lab requires manual design, meaning that none of these mentioned approaches is used [1].

II. Approach

For this lab assignment, 3 two-dimensional datasets are provided. The classification in all of these cases is binary, that is, only two classes can be predicted. Additionally, they are all linearly separable. The line dividing the two classes, also called decision boundary, will be manually chosen with trial-and-error and looking at graphical information.

Once we obtain a valid linear classifier we'll induce random noise to the decision boundary and test its prediction accuracy. Summing it all up, the main steps to follow are:

1) Program a simple classifier.
2) Visualize datasets by plotting them.
3) Set decision boundary parameters by trial-and-error.
4) Add random noise in the following form.
   a) Set the parameter vector w to unit norm.
   b) Create a random vector r of the same dimension as w with a range of $[-1, 1]$.
   c) Set perturbation vector r to unit norm.
   d) Multiply it by a constant p.
   e) Add the noise to the parameters $w_p = w + p\,r$
5) Run the classifier using a high number of different perturbation vectors r, e.g. 100
6) Increase p from 1% to 10% the norm of w in 10 steps and test the classifier at each step.
7) Perform a result analysis.

III. Experimental results

Three datasets were used for this lab:

- `dataset1.txt`: 7 observations.
- `dataset2.txt`: 7 observations.
- `iris-2class.txt`: modified version of the iris dataset where classes 2 and 3 are merged such that we obtain a binary problem. 150 observations.

MATLAB was the selected environment for this lab. In the following graphs we'll represent class 1 with a red cross + and class -1 with a blue circle ∘. the decision boundary will simple by a black line clearly diving both classes.

The hyperplane acting as the decision boundary, in this case two-dimensional, is defined by a set of parameters represented by a vector w.

$$w = [w_0 \quad w_1 \quad w_2] \tag{1}$$

The minimum number of parameters for representing a line in 2D space is two, however, this line will always pass through the origin. To solve this problem we introduce a bias which translates the line to any desired position in the 2D plane. The resulting hyperplane equations is:

$$w_0 + w_1\,x_1 + w_2\,x_2 = 0 \tag{2}$$

where

- w_0 is the bias term.
- w_1 is the weight associated with feature x_1
- w_2 is the weight associated with feature x_2

An extra column of ones is added as an extra feature for the bias parameter, such that:

$$X = [\mathbf{1} \quad X] \tag{3}$$

A. Linear classifier

The linear classifier implemented is a rather simple one, where each observation is classified using the following:

$$y = \begin{cases} -1 & \text{if} \quad X\,w^T > 0 \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

B. Dataset 1

The unit norm vector w that *best* splits the data is:

$$w = [-0.5886 \quad 0.4851 \quad 0.6468] \tag{5}$$

The decision boundary obtained with this w is displayed in figure 1.

A total of 200 random trials were carried out for every step of p. As norm p increases, the probability of getting errors in the prediction process increases.

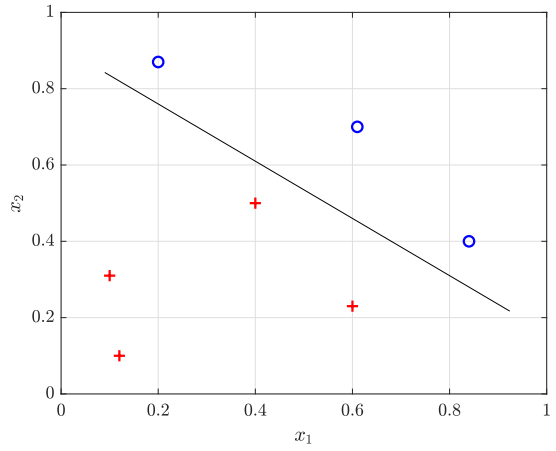

Fig. 1: Dataset 1 with decision boundary.

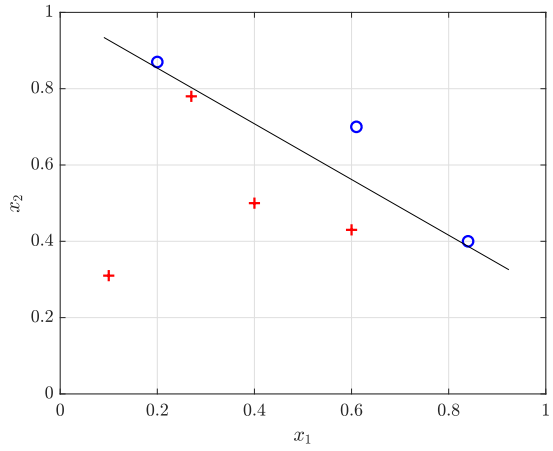

Fig. 3: Dataset 2 with decision boundary.

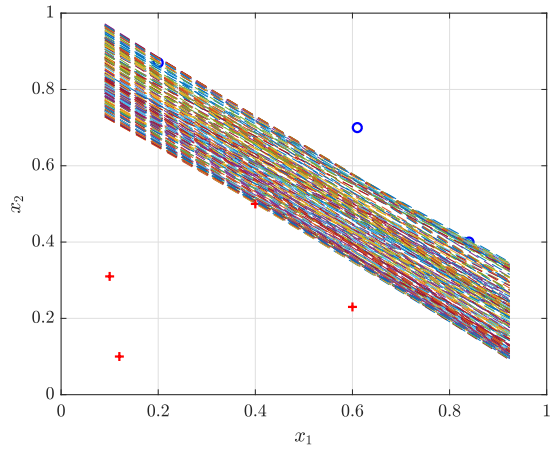

Fig. 2: Overlapped random noise with $p = 0.05$.

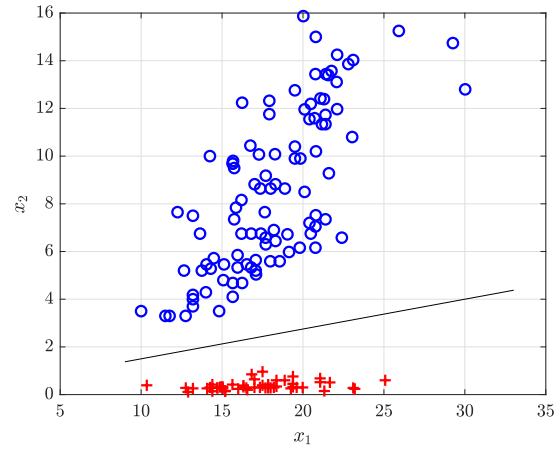

Fig. 4: Iris modified dataset with decision boundary.

C. Dataset 2

The corresponding unit norm parameter vector is:

$$w = [-0.6283 \quad 0.4587 \quad 0.6283] \tag{6}$$

As it can be seen from figure 3, the observations lying on the upper-left corner of the graph are almost tangent to the decision boundary, making them prone to being misclassified when inducing noise.

D. Iris modified dataset

The parameter vector w in unit form is:

$$w = [-0.2408 \quad -0.1204 \quad 0.9631] \tag{7}$$

An interesting behavior is observed when adding noise. In contrast with the previous datasets, the feature range in the iris dataset allows us to realize that the noise acquires a conic shape. Points lying far from the origin will be more sensitive to noise than those lying closer. Curiously enough, the observations seem to follow this conic shape but it won't always be like this.

IV. Analysis

Prediction errors differ from each datasets since they all have a different spread of data. We'll start off by comparing the first two which are fairly similar between each other. Looking at the average number of errors in figure 6 we can immediately distinguish that the second dataset if far more prone to errors even with small perturbations. The way data is spread in such dataset makes it hard to find a hyperplane that accurately divides both classes, in other words, a great number of observations are lying closer to the decision boundary making them possible candidates for misclassification.

As for the modified iris dataset, even though the number of observations is greater than in the previous two, the average

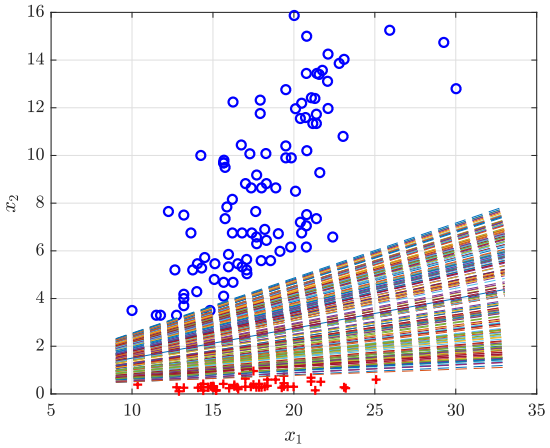

Fig. 5: Conic shaped random noise with $p = 0.1$.

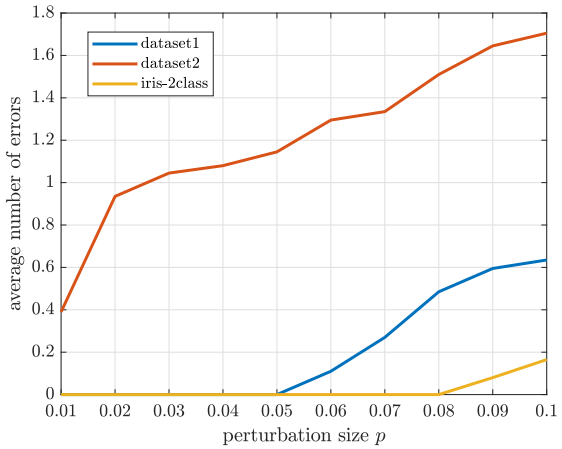

Fig. 6: Error average vs perturbation size.

number of errors is considerably low. The particular spread of the data makes it easier to split the two classes and there are a vast amount of hyperplanes that could accurately classify both classes.

V. Conclusion

The manual approach taken for defining the hyperplane's parameters isn't optimal. While a good decision boundary can be found using visual information, we highly rely on how the data is spread in order to achieve good performance when inducing noise into our chosen parameters.

The error analysis should be made using percentages. In these cases the difference isn't much because the first two of them have the same number of observations. However, when comparing bigger datasets, a percentage can give us more intuition on how well our classifier is performing.

Feature preprocessing could be implemented to avoid getting a conic-shaped noise.

Appendix A
Auxiliary Visualization Plots

These plots were omitted from the main section of the report since they do not contribute with relevant information that could be used for discussion.

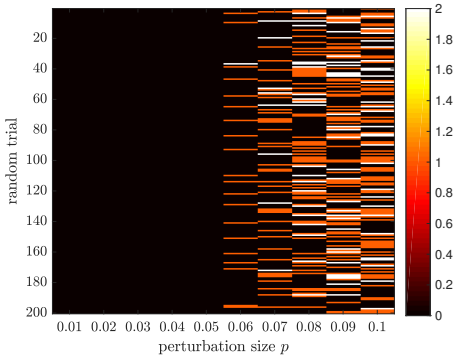

Fig. 7: Dataset 1, number of errors per trial.

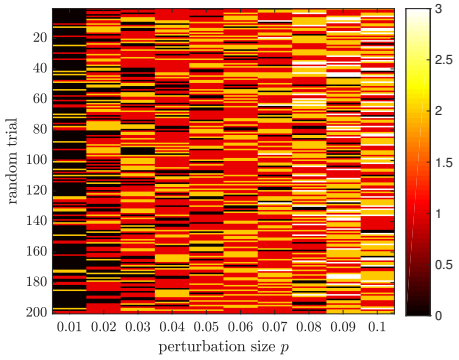

Fig. 8: Dataset 2, number of errors per trial.

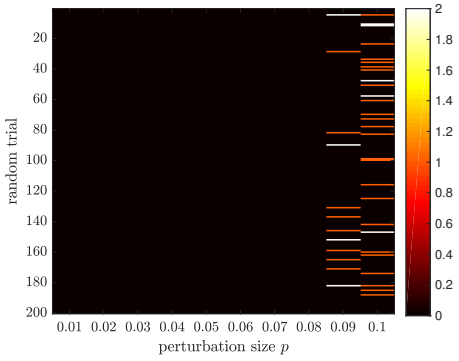

Fig. 9: Modified iris dataset, numbers of error per trial.

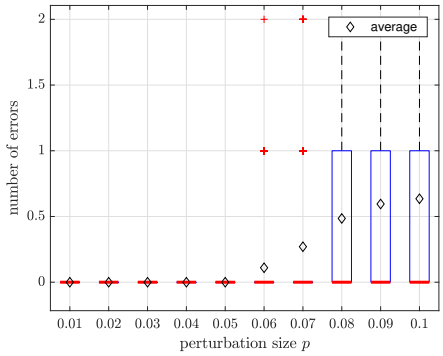

Fig. 10: Dataset 1, error boxplot.

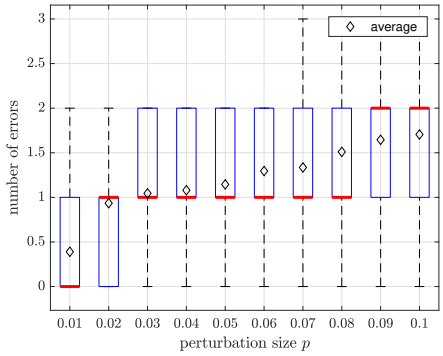

Fig. 11: Dataset 2, error boxplot.

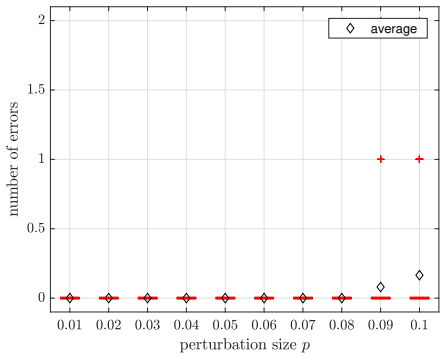

Fig. 12: Modified iris dataset, error boxplot.

REFERENCES

[1] C. Guestrin and E. Fox, "Machine learning: Classification," University of Washington, Tech. Rep.