

The perceptron learning model

Machine Learning II (2023-2024)
UMONS

1 Exercise 1

Prove that the PLA eventually converges to a linear separator for separable data. The following steps will guide you through the proof. Let \mathbf{w}^* be an optimal set of weights (one which separates the data). The essential idea in this proof is to show that the PLA weights $\mathbf{w}(t)$ get "more aligned" with \mathbf{w}^* with every iteration. For simplicity, assume that $\mathbf{w}(0) = 0$.

- (a) Let $\rho = \min_{1 \leq n \leq N} y_n(\mathbf{w}^{*T} \mathbf{x}_n)$. Show that $\rho > 0$.
- (b) Show that $\mathbf{w}^T(t) \mathbf{w}^* \geq \mathbf{w}^T(t-1) \mathbf{w}^* + \rho$, and conclude that $\mathbf{w}^T(t) \mathbf{w}^* \geq t\rho$. [**Hint:** Use induction]
- (c) Show that $\|\mathbf{w}(t)\|^2 \leq \|\mathbf{w}(t-1)\|^2 + \|\mathbf{x}(t-1)\|^2$. [**Hint:** $y(t-1)(\mathbf{w}^T(t-1)\mathbf{x}(t-1)) \leq 0$ because $\mathbf{x}(t-1)$ was misclassified by $\mathbf{w}(t-1)$.]
- (d) Show by induction that $\|\mathbf{w}(t)\|^2 \leq tR^2$, where $R = \max_{1 \leq n \leq N} \|\mathbf{x}_n\|$.
- (e) Using (b) and (d), show that

$$\frac{\mathbf{w}^T(t)}{\|\mathbf{w}(t)\|} \mathbf{w}^* \geq \sqrt{t} \frac{\rho}{R},$$

and hence prove that

$$t \leq \frac{R^2 \|\mathbf{w}^*\|^2}{\rho^2}$$

[**Hint:** $\frac{\mathbf{w}^T(t) \mathbf{w}^*}{\|\mathbf{w}(t)\| \|\mathbf{w}^*\|} \leq 1$. Why?]

In practice, PLA converges more quickly than the bound $\frac{R^2 \|\mathbf{w}^*\|^2}{\rho^2}$ suggests. Nevertheless, because we do not know ρ in advance, we can't determine the number of iterations to converge, which does pose a problem if the data is non-separable.

2 Exercise 2

Let us create our own target function f and data set \mathcal{D} and see how perceptron learning algorithm works.

- (a) First, take $d = 2$ so you can visualize the problem, and choose a random line in the plane as your target function, where one side of the line maps to $+1$ and the other maps to -1 . Choose the input \mathbf{x}_n of the data set as random points in the plane, and evaluate the target function on each \mathbf{x}_n to get the corresponding output y_n . Now generate a data set of size 20. Plot the examples $\{(\mathbf{x}_n, y_n)\}$ as well as the target function f on a plane. Be sure to mark the examples from different classes differently and add labels to the axes of the plot.
- (b) Run the perceptron learning algorithm on the dataset above. Report how long it takes to converge, number of updates that the algorithm takes before converging. Plot the examples $\{(\mathbf{x}_n, y_n)\}$, the target function f , and the final hypothesis g in the same figure. Comment on whether f is close to g .
- (c) Repeat everything in (b) with another randomly generated data set of size 20. Compare your results with (b).
- (d) Repeat everything in (b) with another randomly generated data set of size 100. Compare your results with (b).
- (e) Repeat everything in (b) with another randomly generated data set of size 1000. Compare your results with (b).
- (f) Modify the algorithm such that it takes $\mathbf{x}_n \in \mathbb{R}^{10}$ instead of \mathbb{R}^2 . Randomly generate a linearly separable data set of size 1000 with $\mathbf{x}_n \in \mathbb{R}^{10}$ and feed the dataset to the algorithm. How many updates does the algorithm take to converge?
- (g) Repeat the algorithm on the same data set as (f) for 100 experiments. In the iterations of each experiment, pick $\mathbf{x}(t)$ randomly instead of deterministically. Plot histogram for the number of updates that the algorithm takes to converge.
- (h) Summarize your conclusions with respect to accuracy and running time as function of N and d .

3 Exercise 3

The perceptron learning algorithm works like this: In each iteration t , pick a random $(\mathbf{x}(t), y(t))$ and compute the ‘signal’ $s(t) = \mathbf{w}^T(t)\mathbf{x}(t)$. If $y(t) \cdot s(t) \leq 0$, update \mathbf{w} by

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + y(t) \cdot \mathbf{x}(t),$$

One may argue that this algorithm does not take the ‘closeness’ between $s(t)$ and $y(t)$ into consideration. Let’s look at another perceptron learning algorithm. In each iteration, pick a random $(\mathbf{x}(t), y(t))$ and compute $s(t)$. If $y(t) \cdot s(t) \leq 1$, update \mathbf{w} by

$$\mathbf{w}(t+1) \leftarrow \mathbf{w}(t) + \eta \cdot (y(t) - s(t)) \cdot \mathbf{x}(t),$$

where η is a constant. That is, if $s(t)$ agrees with $y(t)$ well (their product is > 1), the algorithm does nothing. On the other hand, if $s(t)$ is further from $y(t)$, the algorithm changes $\mathbf{w}(t)$ more. In this problem, you are asked to implement this algorithm and study its performance.

- (a) Generate a training data set of size 100 similar to that used in Exercise 2. Generate a test data set of size 10,000 from the same process. To get g , run the algorithm above with $\eta = 100$ on the training data set until a maximum of 1,000 updates has been reached. Plot the training data set, the target function f , and the final hypothesis g on the same figure. Report the error on the test set.
- (b) Use the data set in (a) and redo everything with $\eta = 1$.
- (c) Use the data set in (a) and redo everything with $\eta = 0.01$.
- (d) Use the data set in (a) and redo everything with $\eta = 0.0001$.
- (e) Compare the results that you get from (a) to (d).

The algorithm above is a variant of the so-called Adaline (Adaptive Linear Neuron) algorithm for perceptron learning.

Note: This lab is based on Abu-Mostafa et al., 2012.

References

Abu-Mostafa, Y. S., Magdon-Ismail, M., & Lin, H.-T. (2012). *Learning from data*. AMLBook.