Perceptron Learning Algorithm

Matthieu R. Bloch

1 A bit of geometry

Definition 1.1. *Dataset* $\{\mathbf{x}_i, y_i\}_{i=1}^N$ *is* linearly separable *if there exists* $\mathbf{w} \in \mathbb{R}^d$ *and* $b \in \mathbb{R}$ *such that*

$$\forall i \in [\![1, N]\!] \quad y_i = \text{sgn}(\mathbf{w}^\intercal \mathbf{x} + b) \qquad y_i \in \{\pm 1\}$$

By definition $\text{sgn}(x) = +1$ *if* $x > 0$ *and* -1 *else. The affine set* $\{\mathbf{x} : \mathbf{w}^\intercal \mathbf{x} + b = 0\}$ *is then called a* separating hyperplane.

As illustrated in Fig. 1, it is important to note that $\mathcal{H} \triangleq \{\mathbf{x} : \mathbf{w}^\intercal \mathbf{x} + b = 0\}$ is *not* a vector space because of the presence of the offset b It is an *affine* space, meaning that it can be described as $\mathcal{H} = \mathbf{x}_0 + \mathcal{V}$, where $\mathbf{x}_0 \in \mathcal{H}$ and $\mathcal{V}$ is a vector space. Make sure that this is clear and check for yourself.

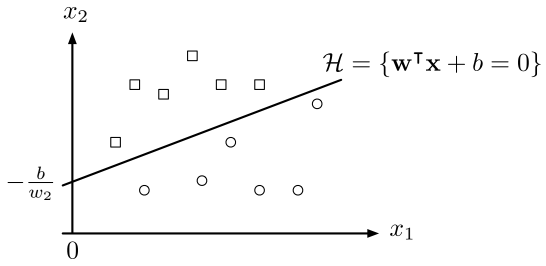

Figure 1: Ilustration of linearly separable dataset

Lemma 1.2. *Consider the hyperplane* $\mathcal{H} \triangleq \{\mathbf{x} : \mathbf{w}^\intercal \mathbf{x} + b = 0\}$. *The vector* $\mathbf{w}$ *is orthogonal to all vectors parallel to the hyperplane. For* $\mathbf{z} \in \mathbb{R}^d$, *the distance of* $\mathbf{z}$ *to the hyperplane is*

$$d(\mathbf{z}, \mathcal{H}) = \frac{|\mathbf{w}^\intercal \mathbf{z} + b|}{\|\mathbf{w}\|_2}$$

.

Proof. Consider $\mathbf{x}, \mathbf{x}'$ in $\mathcal{H}$. Then, by definition, $\mathbf{w}^\intercal \mathbf{x} + b = 0 = \mathbf{w}^\intercal \mathbf{x}' + b$ so that $\mathbf{w}^\intercal (\mathbf{x} - \mathbf{x}') = 0$. Hence, $\mathbf{w}$ is orthogonal to all vectors parallel to $\mathcal{H}$.

Consider now any point $\mathbf{z} \in \mathbb{R}^d$ and a point $\mathbf{x}_0 \in \mathcal{H}$. The distance of $\mathbf{z}$ to $\mathcal{H}$ is the distance between $\mathbf{z}$ and its orthogonal projection onto $\mathcal{H}$, which we can compute as $d(\mathbf{z}, \mathcal{H}) = \frac{|\mathbf{w}^\intercal (\mathbf{z} - \mathbf{x}_0)|}{\|\mathbf{w}\|_2}$. Then,

$$|\mathbf{w}^\intercal (\mathbf{z} - \mathbf{x}_0)| = |\mathbf{w}^\intercal \mathbf{z} + b| . \tag{1}$$

■

2 The Perceptron Learnign Algorithm

The Perceptron Learnign Algorithm (PLA) was proposed by Rosenblatt to identify a separating hyperplane in a linearly separarable dataset $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ if it exist. We assume that every vector $\mathbf{x} \in \mathbb{R}^{d+1}$ with $\mathbf{x}_0 = 1$, so that we can use the shorthand $\theta^\intercal \mathbf{x} = 0$ to describe a affine hyperplane. The principle of the algorithm is the following.

1. Start from a guess $\theta^{(0)}$.
2. For $j \geqslant 1$, iterate over the data points (in any order) and update

$$\theta^{(j+1)} = \begin{cases} \theta^{(j)} + y_i \mathbf{x}_i \text{ if } y_i \neq \operatorname{sgn}\left(\theta^{(j)\intercal} \mathbf{x}_i\right) \\ \theta^{(j)} \text{ else} \end{cases} \tag{2}$$

Geometric view of PLA The effect of the PLA update is illustrated in Fig. 2. Note that the update of $\theta^{(j+1)}$ not only changes the overall hyperplane, and not just the associated vector space. This is best seen in Fig. 2, where the offset changes and not just the slope of the separator.

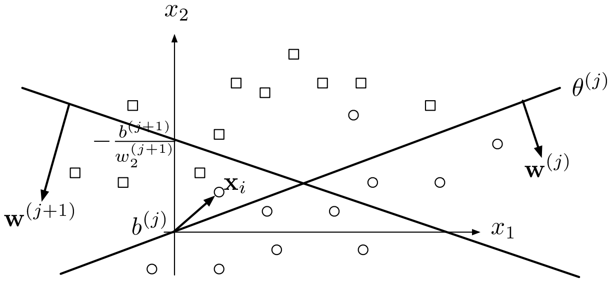

Figure 2: PLA update

Gradient descent view of PLA Consider a loss function called the "perceptron loss" defined as

$$\ell(\theta) \triangleq \sum_{i=1}^{N} \max(0, -y_i \theta^\intercal \mathbf{x}_i). \tag{3}$$

Intuitively, the loss penalizes misclassified points (according to θ) with a penalty proportional to how badly they are misclassified. Setting $\ell_i(\theta) \triangleq \max(0, -y_i \theta^\intercal \mathbf{x}_i)$, we have

$$\nabla \ell_i(\theta) = \begin{cases} 0 \text{ if } y_i \theta^\intercal \mathbf{x}_i > 0 \\ -y_i \mathbf{x}_i \text{ if } y_i \theta^\intercal \mathbf{x}_i < 0 \\ [0, 1] \times -y_i \mathbf{x}_i \text{ if } \theta^\intercal \mathbf{x}_i = 0 \end{cases} \tag{4}$$

The case of equality $\theta^\intercal \mathbf{x}_i = 0$ corresponds to the point where the loss function $\ell_i(\theta)$ is not differentiable. In such case, we have to use a *subgradient* of ℓ_i at θ, which is any vector $\mathbf{v}$ such that for all θ', $\ell_i(\theta) - \ell_i(\theta') \geqslant \mathbf{v}^\intercal(\theta - \theta')$. A subgradient is not unique and the set of subgradients is usually denoted $\partial \ell_i(\theta)$. Let us now apply a stochastic gradient descent algorithm with a step size of 1 to the loss function. We obtain

1. Start from a guess $\theta^{(0)}$.
2. For $j \geqslant 1$, iterate over the data points (in any order) and update

$$\theta^{(j+1)} = \theta^{(j)} - \nabla \ell_i(\theta) = \begin{cases} \theta^{(j)} + y_i \mathbf{x}_i \text{ if } - y_i {\theta^{(j)}}^\intercal \mathbf{x}_i > 0 \\ \theta^{(j)} \text{ if } - y_i {\theta^{(j)}}^\intercal \mathbf{x}_i < 0 \\ \theta^{(j)} - \mathbf{v} \text{ where } \mathbf{v} \in \partial \ell_i(\theta) \text{ if } \theta^\intercal \mathbf{x}_i = 0 \end{cases} \tag{5}$$

Note that (5) is almost identical to (2). The PLA udpate rule is essentially a stochastic gradient descent that treats the case of subgradients with its own rule.

Theorem 2.1. *Consider a linearly separable data set* $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$. *The number of updates made by the PLA because of classification errors is bounded and the PLA eventually identifies a separating hyperplane.*

Proof. By assumption, there exists a separating hyperplane $\mathcal{H}$ with parameter $\theta \triangleq [b\, \mathbf{w}^\intercal]^\intercal$. Note that

$$\min_i d(\mathbf{x}_i, \mathcal{H}) = \min_i \frac{|\theta^\intercal \mathbf{x}_i|}{\|\mathbf{w}\|_2}. \tag{6}$$

Upon setting $\tilde{\mathbf{w}} \triangleq \frac{\mathbf{w}}{\|\mathbf{w}\|_2}$ and $\tilde{b} \triangleq \frac{b}{\|\mathbf{w}\|_2}$, remark that hyperplanes $\{\mathbf{x} : \mathbf{w}^\intercal \mathbf{x} + b = 0\}$ and $\{\mathbf{x} : \tilde{\mathbf{w}}^\intercal \mathbf{x} + \tilde{b} = 0\}$ are identical and we can assume without loss of generality that we use a parameter $\tilde{\theta} = [\tilde{b}\, \tilde{\mathbf{w}}^\intercal]^\intercal$ such that

$$\min_i d(\mathbf{x}_i, \mathcal{H}) = \min_i \left| \tilde{\theta}^\intercal \mathbf{x}_i \right| \triangleq \rho. \tag{7}$$

Consider a situation with a positive error, for which $\text{sgn}({\theta^{(j)}}^\intercal \mathbf{x}) = -1$ but $y = +1$. In such case,

$$ {\theta^{(j+1)}}^\intercal \tilde{\theta} = (\theta^{(j)} + \mathbf{x})^\intercal \tilde{\theta} = {\theta^{(j)}}^\intercal \tilde{\theta} + \underbrace{\mathbf{x}^\intercal \tilde{\theta}}_{\geqslant \rho} \geqslant {\theta^{(j)}}^\intercal \tilde{\theta} + \rho. \tag{8}$$

Consider now a situation with a negative error, for which $\text{sgn}({\theta^{(j)}}^\intercal \mathbf{x}) = +1$ but $y = -1$. In such case, we have again

$$ {\theta^{(j+1)}}^\intercal \tilde{\theta} = (\theta^{(j)} - \mathbf{x})^\intercal \tilde{\theta} = {\theta^{(j)}}^\intercal \tilde{\theta} - \underbrace{\mathbf{x}^\intercal \tilde{\theta}}_{\leqslant -\rho} \geqslant {\theta^{(j)}}^\intercal \tilde{\theta} + \rho. \tag{9}$$

We can conclude that if we have made m PLA updates after j steps, it must hold that

$$ {\theta^{(j+1)}}^\intercal \tilde{\theta} \geqslant {\theta^{(0)}}^\intercal \tilde{\theta} + m\rho. \tag{10}$$

Define now $\tau \triangleq \max_i \|\mathbf{x}_i\|_2$. Consider a situation with positive error and note that

$$\|\theta^{(j+1)}\|_2^2 = \|\theta^{(j)} + \mathbf{x}\|_2^2 = \|\theta^{(j)}\|_2^2 + \|\mathbf{x}\|_2^2 + 2 \underbrace{\mathbf{x}^\intercal \theta^{(j)}}_{\leqslant 0} \leqslant \|\theta^{(j)}\|_2^2 + \tau^2 \tag{11}$$

Similarly, for a situation with a negative error, we have

$$\|\theta^{(j+1)}\|_2^2 = \|\theta^{(j)} - \mathbf{x}\|_2^2 = \|\theta^{(j)}\|_2^2 + \|\mathbf{x}\|_2^2 - 2 \underbrace{\mathbf{x}^\intercal \theta^{(j)}}_{\geqslant 0} \leqslant \|\theta^{(j)}\|_2^2 + \tau^2 \tag{12}$$

We can therefore conclude that if we have made m error after j steps, it must hold that

$$\|\theta^{(j+1)}\|_2^2 \leqslant \|\theta^{(0)}\|_2^2 + m\tau^2. \tag{13}$$

We finally tie in (10) and (13) using Cauchy-Schwarz inequality.

$$\theta^{(0)\intercal}\tilde{\theta} + m\rho \leqslant \theta^{(j+1)\intercal}\tilde{\theta} \leqslant \|\theta^{(j+1)}\|_2\|\tilde{\theta}\|_2 \leqslant \|\tilde{\theta}\|_2\sqrt{\|\theta^{(0)}\|_2^2 + m\tau^2}. \tag{14}$$

Since we assumed (without losing much generality) that $\theta^{(0)} = 0$, we obtain that the number m of errors must satisfy

$$m \leqslant \frac{\|\tilde{\theta}\|_2^2\tau^2}{\rho^2}. \tag{15}$$

In other words, if after going sufficiently many points in the dataset, if we have made more than $\frac{\|\tilde{\theta}\|_2^2\tau^2}{\rho^2}$ updates because of errors, we must have found a separating hyperplane. ■

The result of Theorem 2.1 is quite remarkable because the dimension of the data does not appear and the order in which the data points are processed has no incidence. Nevertheless, the convergence can be very slow, especially if the ratio $\frac{\tau}{\rho}$ in (15) is very small. Note that we may *not* know $\frac{\tau}{\rho}$ ahead of time, so that we cannot not guarantee how long it will take for the algorithm to find a separating hyperplane.

3 Maximum margin hyperplane

Although the PLA is guaranteed to find a separating hyperplane in linearly separable data, not all separating hyperplanes are equally useful. Consider the situation illustrated in Fig. 3, which shows two valid separating hyperplanes for linearly separable dataset in $\mathbb{R}^2$. Intuitively, $\mathcal{H}_1$ is likely to be sensitive to statistical variations in the data set because it is too close to some of the points in the class. In contrast, $\mathcal{H}_2$ has some *margin* that is likely to make the prediction more robust.

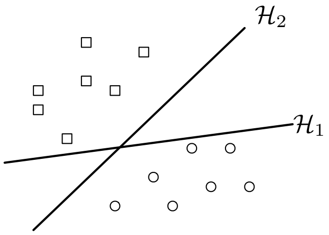

Figure 3: All separating hyperplanes are equal but some are more equal than others.

Definition 3.1. *The margin of a separating hyperplane* $\mathcal{H} \triangleq \{\mathbf{x} : \mathbf{w}^\intercal\mathbf{x} + b = 0\}$ *for a linearly separable dataset* $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ *is*

$$\rho(\mathbf{w}, b) \triangleq \min_{i\in[\![1,N]\!]} \frac{|\mathbf{w}^\intercal\mathbf{x}_i + b|}{\|\mathbf{w}\|_2} \tag{16}$$

The maximum margin hyperplane is then defined as $\mathcal{H}^* \triangleq \{\mathbf{x} : \mathbf{w}^{*\intercal}\mathbf{x} + b^* = 0\}$ *such that*

$$(\mathbf{w}^*, b^*) = \underset{\mathbf{w},b}{\textit{argmax}}\, \rho(\mathbf{w}, b). \tag{17}$$

Intuitively, the maximum margin hyperplane leads to a more robust separation of the classes and therefore benefits from a better generalization. For linearly separable datasets with $\mathcal{Y} = \{\pm 1\}$, it is also convenient to write the separating hyperplane in canonical form.

Definition 3.2. *The canonical form* $(\mathbf{w}, b)$ *of a separating hyperplane is such that*

$$\forall i \in [\![1, N]\!]\; y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geqslant 1 \textit{ and } \exists i^* \in [\![1, N]\!] \textit{ s.t. } y_{i^*}(\mathbf{w}^\intercal \mathbf{x}_{i^*} + b) = 1. \tag{18}$$

The canonical form can always be obtained by normalizing $\mathbf{w}$ and b by $\min_i |\mathbf{w}^\intercal \mathbf{x}_i + b|$. By rewriting all hyperplanes in canonical form, the maximum margin hyperplane can be characterized as follows.

$$\begin{aligned}(\mathbf{w}^*, b^*) &= \operatorname*{argmax}_{\mathbf{w},b} \rho(\mathbf{w}, b) && (19)\\ &= \operatorname*{argmax}_{\mathbf{w},b} \frac{1}{\|\mathbf{w}\|_2} \text{ s.t. } \forall i \in [\![1, N]\!]\; y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geqslant 1 && (20)\\ &= \operatorname*{argmin}_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|_2^2 \text{ s.t. } \forall i \in [\![1, N]\!]\; y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geqslant 1 && (21)\end{aligned}$$

Note that in (20), we have dropped the condition $\exists i^* \in [\![1, N]\!] \quad \text{s.t.} \quad y_{i^*}(\mathbf{w}^\intercal \mathbf{x}_{i^*} + b) = 1$. This can be justified a posteriori by the fact that the maximum margin hyperplane must satisfy the constraint with equality. We will discuss this again later after reviewing a bit of constrained convex optimization. The vectors $\mathbf{x}_i$ such that $y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) = 1$ are called *support vectors* and will reappear when we discuss support vector machines. In (21), we have used the fact that maximizing $\frac{1}{\|\mathbf{w}\|_2}$ is equivalent to minimizing $\frac{1}{2}\|\mathbf{w}\|_2^2$. The choice of the quadratic function is motivated by its nice analytical properties that make the numerical optimization more stable. The good news is that the optimization in (21) is a constrained *quadratic* program that we know how to solve extremely efficiently.

4 Non-linearly separable data

The previous discussion hinges on the fact that the dataset is linearly separable. In reality, this is unlikely to happen in practice because their might be noise in the labels or because the exists no true separation between classes. In such case we need to relax the notion of maximum margin hyperplane and consider *soft margins*. Specifically, if a dataset is *not* linearly separable, then it is impossible to guarantee that $\forall i \in [\![1, N]\!]\; y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geqslant 1$. The solution is therefore to introduce positive slack variables $\boldsymbol{\xi} = \{\xi_i\}_{i=1}^N$ and only seek to enforce

$$\forall i \in [\![1, N]\!]\; y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geqslant 1 - \xi_i. \tag{22}$$

When $\xi_i > 1$, we effectively allow the classification to make an error. Of course, we would like to ensure that this does not happen much and rather than fixing the $\boldsymbol{\xi}$ ahead of time, we make them part of the optimization.

Definition 4.1. *For a chosen* $C > 0$*, the optimal soft-margin hyperplane is*

$$\mathop{\textit{argmin}}_{\mathbf{w},b,\boldsymbol{\xi}} \frac{1}{2}\|\mathbf{w}\|_2^2 + \frac{C}{N}\sum_{i=1}^N \xi_i \textit{ s.t. } \forall i \in [\![1, N]\!]\; y_i(\mathbf{w}^\intercal \mathbf{x}_i + b) \geqslant 1 - \xi_i \textit{ and } \xi_i \geqslant 0. \tag{23}$$

Note that $\sum_{i=1}^{N}\xi_i \geqslant 0$ can be viewed as the *cost* incurred when misclassifying a datapoint. The bigger ξ_i, the larger the cost. The parameter C allows the user to tradeoff the minimization of $\|\mathbf{w}\|_2$, which controls the margin, and the minimization of $\boldsymbol{\xi}$, which controls the penalty incurred when misclassifying. As illustrated in Fig. 4, a small value of C makes the cost of misclassification negligible and the optimization will favor a hyperplane that separates the data well and ignores a few misclassified outliers. In contrast, a large value of C will result in a hyperplane that avoids misclassifications as much as possible.

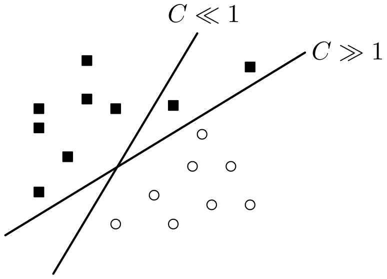

Figure 4: Soft-margin hyperplanes

Regardless of the soft-margin constraints, there exist datasets that are never remotely close to linearly separable. Fig. 5 provides an illustration in $\mathbb{R}^2$. The data cannot be properly classified with a linear classifier because it is intrinsically not linear in the features x_1 and x_2. Note, however, that the data could be separated in principle using the rule $x_1^2+x_2^2 \gtrless 1$.

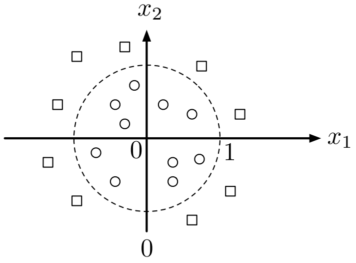

Figure 5: Example of non-linearly separable data

Note that the rule $x_1^2+x_2^2 \gtrless 1$ is a linear classifier on the *nonlinear* features x_1^2 and x_2^2. This suggests that we could transform the feature vector *before* using a linear classifier using the map $\Phi : \mathbb{R}^d \to \mathbb{R}^p$ such that

$$\Phi : \begin{bmatrix} x_1 \\ \vdots \\ x_d \end{bmatrix} \to \begin{bmatrix} \phi_1(\mathbf{x}) \\ \vdots \\ \vdots \\ \phi_p(\mathbf{x}) \end{bmatrix}. \tag{24}$$

This may seem like a good idea, but note that this is not really telling us how to create the non-linear features and how many to generated. Note that if $p \gg n$, there is a risk of overfitting the training set by using linear features that separate the training data well but have no chance of generalizing well.

Remark 4.2. *We could have chosen to apply a nonlinear transform after classifying, e.g.,* $\Phi(\mathbf{w}^\intercal\mathbf{x}+b)$*. However the choice of using* $\mathbf{w}^\intercal\Phi(\mathbf{x})+b$ *allows us to reuse the linear classifier seen earlier and will allow us to introduce the concept of kernels later on.*

5 To go further

More details on soft margin optimization can be found in [1, Section 12.2].

References

[1] T. Hastie, R. Tibshirani, and J. H. Friedman, *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*, ser. Springer series in statistics. Springer, 2009.