The Perceptron Algorithm

Kristoffer Stensbo-Smidt

14th December 2015

The *perceptron algorithm* is an algorithm for supervised linear classification. Restricting ourselves to considering only binary classification, the most basic linear classifier is a hyperplane separating the two classes of our dataset.

More formally, assume that we have a normal vector $\mathbf{w} \in \mathbb{R}^D$ defining a hyperplane. Binary classification is typically performed by defining a function f, such that

$$
\begin{aligned}
f(\mathbf{x}) &= \mathbf{w}^\mathsf{T}\mathbf{x} + b && (1)\\
&= b + \sum_{i=1}^{D} w_i x_i, && (2)
\end{aligned}
$$

where $\mathbf{x} \in \mathbb{R}^D$ is an input sample. The plane is then simply defined as $f(\mathbf{x}) = 0$ and classification is done by assigning class $+1$ to samples $\mathbf{x}'$ where $f(\mathbf{x}') > 0$ and, similarly, class -1 to samples where $f(\mathbf{x}') < 0$. We call these two classes $\mathcal{C}_1$ and $\mathcal{C}_2$, respectively.

In a practical setting, we need to tune the hyperplane to a dataset, called the *training set*, such that it separates the classes of the samples. We can then use the tuned (or *trained*) hyperplane to classify new, unknown samples. Therefore, assume we have a dataset $S = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$, where $\mathbf{x} \in \mathbb{R}^D$ and $t = \{-1, +1\}$ indicating the classes of the samples. We now want to adapt the hyperplane to this dataset – in other words, $\mathbf{w}$ and b must be learned from the data.

The million dollar question is now: how do we learn a normal vector $\mathbf{w}$ that separates our training set? This is exactly the question that the perceptron algorithm answers.

To ease the notation, we will in the following use the definitions $\mathbf{w} = (w_1, \ldots, w_D, b)^\mathsf{T}$ and $\mathbf{x} = (x_1, \ldots, x_D, 1)^\mathsf{T}$, so $f(\mathbf{x}) = \mathbf{w}^\mathsf{T}\mathbf{x}$. The text is based on Bishop (2006).

The perceptron

To find the optimal hyperplane for our dataset, we need to minimise the *error* that it makes when trying to separate our training set. The error function used by the perceptron is known as the *perceptron criterion*. To derive this, remember that we want $f(\mathbf{x}_i) > 0$ for every sample $\mathbf{x}_i$ in class $\mathcal{C}_1$ and $f(\mathbf{x}_i) < 0$ for every sample $\mathbf{x}_i$ in class $\mathcal{C}_2$. Because we have used $t = \{-1, +1\}$ to encode the class, the problem reduces to ensuring that $t_i f(\mathbf{x}_i) > 0$ for all samples $\mathbf{x}_i$. Any misclassified sample will have $t_i f(\mathbf{x}_i) < 0$, so to minimise the error we need to minimise the error function

$$
E(\mathbf{w}) = -\sum_{i \in \mathcal{M}} t_i \mathbf{w}^\mathsf{T}\mathbf{x}_i, \qquad (3)
$$

where $\mathcal{M}$ denotes the set of all misclassified samples. Eq. (3) is exactly the perceptron criterion. Any $\mathbf{w}$ minimising this quantity is therefore a hyperplane separating the classes of the training set, assuming such a separation exists[1].

Eq. (3) is, however, not differentiable, so we need another approach to minimising this function. We will use an *iterative* approach, i.e., we update the normal vector $\mathbf{w}$ in

[1] In fact, any hyperplane fully separating the classes will have $E(\mathbf{w}) = 0$.

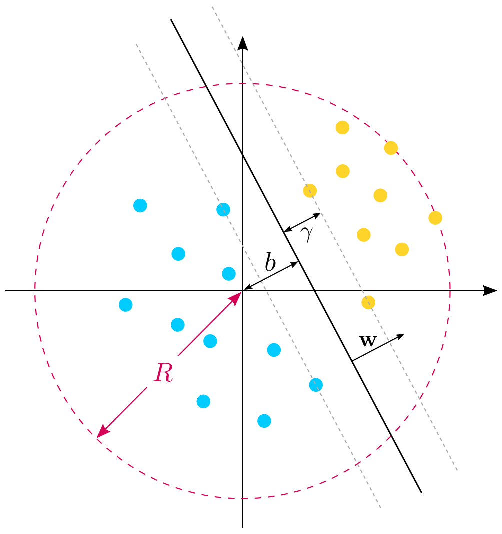

Figure 1 A dataset consisting of two classes separated by a hyperplane $\mathbf{w}^{\mathsf{T}}\mathbf{x} + b = 0$. The functional margin γ is the shortest distance from the hyperplane to any of the samples. The distance from the hyperplane to the origin is b.

steps such that it, on average, moves closer to the minimum of Eq. (3). More formally, we obtain the normal vector of iteration $k+1$ as follows:

$$\mathbf{w}_{k+1} = \mathbf{w}_k + \Delta\mathbf{w}. \tag{4}$$

The minimisation scheme we will use is known as *stochastic gradient descent* (SGD) and updates the normal vector each time it encounters a misclassified point. In SGD $\Delta\mathbf{w} = -\eta\nabla E(\mathbf{w})$, where η is the so-called learning rate parameter. Thus,

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \eta\nabla E(\mathbf{w}) \tag{5}$$
$$= \mathbf{w}_k + \eta t_i \mathbf{x}_i. \tag{6}$$

To update the normal vector, all we have to do is to add (or subtract) one of the misclassified samples.

This concludes the perceptron learning algorithm. To find a hyperplane separating the classes of the training set, we continuously apply Eq. (6) until no misclassified points are left. Note, however, that each time we update the normal vector, some of the previously correctly classified samples may become misclassified, so the perceptron learning algorithm (Eq. (6)) is not guaranteed to reduce the overall error with each update. The *perceptron convergence theorem*, however, states that if it is possible to separate the two classes of the dataset with a hyperplane, then the perceptron learning algorithm is *guaranteed* to find it in a finite number of iterations.

The perceptron convergence theorem

To prove the perceptron convergence theorem, we need to introduce some definitions. Fig. 1 illustrates the situation of a hyperplane separating a dataset consisting of two classes. The first definition we will need is the concept of a containing sphere centred at the origin. This sphere will have a radius R, such that

$$R = \max_i \|\mathbf{x}_i\|. \tag{7}$$

The second definition we will need is that of the *functional margin*. This is simply the distance from the hyperplane to a sample $\mathbf{x}_i$,

$$\gamma_i = t_i \mathbf{w}^{\mathsf{T}}\mathbf{x}_i. \tag{8}$$

When talking about *the* functional margin, we are referring to the functional margin of the entire dataset, defined as the minimum of all functional margins:

$$\gamma = \min_i t_i \mathbf{w}^\mathsf{T} \mathbf{x}_i \,. \tag{9}$$

With these definitions in place, we can now continue to prove the convergence theorem.

The perceptron convergence theorem was proved by Block (1962) and Novikoff (1962). The following version is based on that in Cristianini and Shawe-Taylor (2000).

Theorem 1 (Block, Novikoff). *Let the training set $S = \{(\mathbf{x}_1, t_1), \ldots, (\mathbf{x}_n, t_n)\}$ be contained in a sphere of radius R about the origin. Assume the dataset to be linearly separable, and let $\mathbf{w}_{\text{opt}}$, $\|\mathbf{w}_{\text{opt}}\| = 1$, define the hyperplane separating the samples, having functional margin $\gamma > 0$. We initialise the normal vector as $\mathbf{w}_0 = \mathbf{0}$. The number of updates, k, of the perceptron algorithms is then bounded by*

$$k \leq \left(\frac{2R}{\gamma}\right)^2 . \tag{10}$$

Proof. Though the proof can be done using the augmented normal vector and samples defined in the beginning, the notation will be a lot easier if we introduce a different augmentation: $\hat{\mathbf{w}} = (\mathbf{w}^\mathsf{T}, b/R)^\mathsf{T} = (w_1, \ldots, w_D, b/R)^\mathsf{T}$ and $\hat{\mathbf{x}} = (\mathbf{x}^\mathsf{T}, R)^\mathsf{T} = (x_1, \ldots, x_D, R)^\mathsf{T}$.

We first derive an upper bound on how fast the normal vector grows. As the hyperplane is unchanged if we multiply $\hat{\mathbf{w}}$ by a constant, we can set $\eta = 1$ without loss of generality. Let $\hat{\mathbf{w}}_{k+1}$ be the updated (augmented) normal vector after the kth error has been observed.

$$\begin{aligned}
\|\hat{\mathbf{w}}_{k+1}\|^2 &= (\hat{\mathbf{w}}_k + t_i \hat{\mathbf{x}}_i)^\mathsf{T} (\hat{\mathbf{w}}_k + t_i \hat{\mathbf{x}}_i) &(11)\\
&= \hat{\mathbf{w}}_k^\mathsf{T} \hat{\mathbf{w}}_k + \hat{\mathbf{x}}_i^\mathsf{T} \hat{\mathbf{x}}_i + 2 t_i \hat{\mathbf{w}}_k^\mathsf{T} \hat{\mathbf{x}}_i &(12)\\
&= \|\hat{\mathbf{w}}_k\|^2 + \|\hat{\mathbf{x}}_i\|^2 + 2 t_i \hat{\mathbf{w}}_k^\mathsf{T} \hat{\mathbf{x}}_i \,. &(13)
\end{aligned}$$

Since an update was triggered, we know that $t_i \hat{\mathbf{w}}_k^\mathsf{T} \hat{\mathbf{x}}_i \leq 0$, thus

$$\begin{aligned}
\|\hat{\mathbf{w}}_k\|^2 + \|\hat{\mathbf{x}}_i\|^2 + 2 t_i \hat{\mathbf{w}}_k^\mathsf{T} \hat{\mathbf{x}}_i &\leq \|\hat{\mathbf{w}}_k\|^2 + \|\hat{\mathbf{x}}_i\|^2 &(14)\\
&= \|\hat{\mathbf{w}}_k\|^2 + (\|\mathbf{x}_i\|^2 + R^2) &(15)\\
&\leq \|\hat{\mathbf{w}}_k\|^2 + 2R^2 \,. &(16)
\end{aligned}$$

This implies that $\|\hat{\mathbf{w}}_k\|^2 \leq 2kR^2$, thus

$$\|\hat{\mathbf{w}}_{k+1}\|^2 \leq 2(k+1)R^2 \,. \tag{17}$$

We then proceed to show how the inner product between an update of the normal vector and $\hat{\mathbf{w}}_{\text{opt}}$ increase with each update:

$$\begin{aligned}
\hat{\mathbf{w}}_{\text{opt}}^\mathsf{T} \hat{\mathbf{w}}_{k+1} &= \hat{\mathbf{w}}_{\text{opt}}^\mathsf{T} \hat{\mathbf{w}}_k + t_i \hat{\mathbf{w}}_{\text{opt}}^\mathsf{T} \hat{\mathbf{x}}_i &(18)\\
&\geq \hat{\mathbf{w}}_{\text{opt}}^\mathsf{T} \hat{\mathbf{w}}_k + \gamma &(19)\\
&\geq (k+1)\gamma \,, &(20)
\end{aligned}$$

since $\hat{\mathbf{w}}_{\text{opt}}^\mathsf{T} \hat{\mathbf{w}}_k \geq k\gamma$. We therefore have

$$k^2 \gamma^2 \leq (\hat{\mathbf{w}}_{\text{opt}}^\mathsf{T} \hat{\mathbf{w}}_k)^2 \leq \|\hat{\mathbf{w}}_{\text{opt}}\|^2 \|\hat{\mathbf{w}}_k\|^2 \leq 2kR^2 \|\hat{\mathbf{w}}_{\text{opt}}\|^2 \,, \tag{21}$$

where we have made use of the Cauchy-Schwarz inequality. As $k^2\gamma^2$ grows faster than $2kR^2$, Eq. (21) can hold if and only if

$$k \leq 2\|\hat{\mathbf{w}}_{\text{opt}}\|^2 \frac{R^2}{\gamma^2} \,. \tag{22}$$

As $b \leq R$, we can rewrite the norm of the normal vector:

$$\|\hat{\mathbf{w}}_{\mathrm{opt}}\|^2 = \|\mathbf{w}_{\mathrm{opt}}\|^2 + \frac{b^2}{R^2} \leq \|\mathbf{w}_{\mathrm{opt}}\|^2 + 1 = 2 . \tag{23}$$

The bound on k now becomes

$$k \leq 4\frac{R^2}{\gamma^2} = \left(\frac{2R}{\gamma}\right)^2 , \tag{24}$$

which therefore bounds the number of updates necessary to find the separating hyperplane. □

References

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Block, H. D. (1962). The perceptron: A model for brain functioning. i. *Reviews of Modern Physics 34*(1), 123–134.

Cristianini, N. and J. Shawe-Taylor (2000). *An introduction to support vector machines and other kernel-based learning methods*. Cambridge university press.

Novikoff, A. B. J. (1962). On convergence proofs on perceptrons. In *Proceedings of the Symposium on the Mathematical Theory of Automata*, pp. 615–622.