CS 337, Fall 2023

Mistake Bounds of The Perceptron Classifier

Scribes: Ashutosh Singh, Avadhoot Gorakh Jadhav, Gohil Megh Hiteshkumar
Govind Kumar, Aman Sharma, Vir Wankhede
(***Equal contribution by all scribes***)
Edited by: Dhiraj Kumar Sah

September 12, 2023

Recap

Perceptron is a linear model of classification. The model aims to estimate the best hyperplane that separates the training data. Equation of hyperplane: $\mathbf{w^T x = 0}$. It makes weight updates by seeing train instances one at a time. Weight updates are triggered **only when** Perceptron makes a mistake. For a given $\mathbf{x}$, perceptron predicts label as $\text{sgn}(\mathbf{w}^T\mathbf{x})$. If the predicted label does not match the actual label, then the weight vector is moved either towards or away from $\mathbf{x}$ as follows:

$$\mathbf{w_{t+1}} = \mathbf{w_t} + \mathbf{x} \quad \text{if } y = 1$$
$$\mathbf{w_{t+1}} = \mathbf{w_t} - \mathbf{x} \quad \text{if } y = -1$$

Perceptron Algorithm

```
1: $\mathbf{w} \leftarrow \mathbf{w_0}$
2: for iter $= 1, \ldots, MaxIter$ do
3:    for all $(\mathbf{x}, y) \in D$ do
4:       $\hat{y} \leftarrow \text{sgn}(\mathbf{w}^T\mathbf{x})$
5:       if $\hat{y} \neq y$ then
6:          $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
7:       end if
8:    end for
9: end for
10: return $\mathbf{w}$
```

Perceptron Loss

For an example $(\mathbf{x_i}, y_i)$ the perceptron loss is: $\max(0, -y_i\mathbf{w^T x_i})$
So, the loss for the entire dataset will be:

$$L_{per}(\mathbf{w}, \mathcal{D}) = \sum_i \max(0, -y_i\mathbf{w^T x_i})$$

In stochastic Gradient Descent, the weight update will be:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta\nabla_w L_{per}(\mathbf{w}, (x_i, y_i))$$

The gradient of above loss function does not exist when $\mathbf{w^T x_i} = \mathbf{0}$.
So, we use subgradient functions.

For a convex function $f(x)$, a vector $\mathbf{g}$ is a subgradient of f at $\mathbf{w_0}$ if the following condition holds for all $\mathbf{w}$:

$$\forall w, f(\mathbf{w}) - f(\mathbf{w_0}) \geq g^T(\mathbf{w} - \mathbf{w_0})$$

If $\mathbf{f}$ is convex and differentiable, $\nabla f(x)$ is a subgradient of $\mathbf{f}$ at $\mathbf{x}$. Also, subgradients are not necessarily strictly decreasing.

The perceptron loss function isn't differentiable at all points so we can use a subgradient in place of the gradient. The subgradient of the perceptron loss, $\mathbf{g}$ can be written as

$$g = \begin{cases} 0, & \text{if} \quad y\mathbf{w}^T\mathbf{x} > 0 \\ -y\mathbf{x}, & \text{if} \quad y\mathbf{w}^T\mathbf{x} < 0 \\ ky\mathbf{x}, & \text{if} \quad y\mathbf{w}^T\mathbf{x} = 0, k \in [-1, 0] \end{cases}$$

If we pick $k = -1$, then

$$g = \begin{cases} 0, & \text{if} \quad y\mathbf{w}^T\mathbf{x} > 0 \\ -y\mathbf{x}, & \text{if} \quad y\mathbf{w}^T\mathbf{x} \leq 0 \end{cases}$$

Thus, the SGD weight update rule would become

$$w \leftarrow w + y\mathbf{x} \text{ if } y\mathbf{w}^T\mathbf{x} \leq 0$$

Convergence bound of perceptron algorithm

Consider a linearly separable dataset $\mathcal{D}$, i.e., $\exists\mathbf{u}$ such that,

$$y = sgn(\mathbf{u^T x}) \quad \forall(\mathbf{x}, y) \in D$$

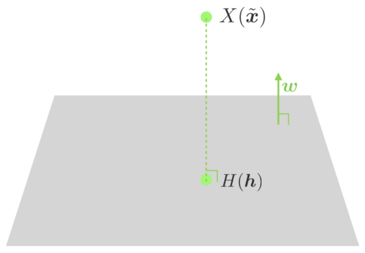
Without loss of generality, Let's assume:

- $\mathbf{u}$ is a unit vector, $||\mathbf{u}||_2 = 1$
- All $\mathbf{x_i} \in \mathcal{D}$ are scaled to lie within a Euclidean ball of unit radius, i.e. $||\mathbf{x_i}||_2 \leq 1$.

Def^n **: margin of separation** (γ) is defined as,

$$\gamma = \min_{\mathsf{x} \in \mathcal{D}} |\mathsf{u}^\mathsf{T}\mathsf{x}| \tag{1}$$

Distance between point and hyperplane



The equation of the hyperplane in the figure is $\mathbf{w}^\mathbf{T}\mathbf{x} + \mathbf{b} = 0$. We have to find the distance of point X from the hyperplane. Now consider a point H on the hyperplane such that vector $\tilde{\mathbf{x}} - \mathbf{h}$ is orthogonal to the plane. that is $\mathbf{w}$ is parallel to $\tilde{\mathbf{x}} - \mathbf{h}$. So,

$$\begin{aligned}
\tilde{\mathbf{x}} - \mathbf{h} &= k\mathbf{w} \\
\therefore \mathbf{h} &= \tilde{\mathbf{x}} - k\mathbf{w} \\
\therefore \mathbf{w}^\mathbf{T}(\tilde{\mathbf{x}} - k\mathbf{w}) + \mathbf{b} &= 0 \\
\therefore \mathbf{w}^\mathbf{T}\tilde{\mathbf{x}} + \mathbf{b} &= k||\mathbf{w}||_2^2 \\
\therefore k &= \frac{\mathbf{w}^\mathbf{T}\tilde{\mathbf{x}} + \mathbf{b}}{||\mathbf{w}||_2^2}
\end{aligned}$$

Now, the distance between the hyperplane and point X is the same as the distance between point X and H. So, the distance between hyperplane $\mathbf{w}^\mathbf{T}\mathbf{x} + \mathbf{b}$ and point $\tilde{\mathbf{x}}$ is:

$$\begin{aligned}
||\tilde{\mathbf{x}} - \mathbf{h}||_2 &= ||k\mathbf{w}||_2^2 \\
&= |k| \cdot ||\mathbf{w}||_2^2 \\
&= \frac{|\mathbf{w}^\mathbf{T}\tilde{\mathbf{x}} + \mathbf{b}|}{||\mathbf{w}||_2}
\end{aligned}$$

So, the distance between hyperplane $\mathbf{w}^\mathbf{T}\mathbf{x} + \mathbf{b}$ and point $\tilde{\mathbf{x}}$ is $\frac{|\mathbf{w}^\mathbf{T}\tilde{\mathbf{x}} + \mathbf{b}|}{||\mathbf{w}||_2}$

So, the distance of any point $\tilde{x}$ from the given hyperplane $\mathbf{u}^\mathbf{T}\mathbf{x} = 0$ will be:

$$\frac{|\mathbf{u}^\mathbf{T}\tilde{\mathbf{x}}|}{||\mathbf{u}||_2}$$

Here, $\mathbf{u}$ is a unit vector. So, the distance is $|\mathbf{u}^\mathbf{T}\tilde{\mathbf{x}}|$.
So, γ is the minimum distance of a point $\mathbf{x} \in D$ from the hyperplane $\mathbf{u}^\mathbf{T}\mathbf{x} = 0$

Theorem 1. ***Novikoff's Theorem:*** *If there exist a unit vector* $\mathbf{u}$ *s.t.* $y\mathbf{u}^\mathbf{T}\mathbf{x} \geq \gamma \quad \forall(x, y) \in D$ *then perceptron algorithm will make no more than* $\frac{1}{\gamma^2}$ *mistake on training dataset* $\mathcal{D}$ *when initialisation is* $\mathbf{w_0} = \mathbf{0}$.

That is, the number of mistakes made by the perceptron algorithm -

- **Does not** depend on the size of feature space
- **Does not** depend on the number of examples
- **Only** depends on the margin of separation

Proof. We will monitor the growth of the following quantities across weight updates

- $\mathbf{w^T u}$
- $||\mathbf{w}||_2^2$

Why these quantities?

The desired weight vector is $\mathbf{u}$. We want our trained weight vector $\mathbf{w}$ to be more and more like $\mathbf{u}$. We want $\mathbf{w}$ to be more and more parallel to $\mathbf{u}$. So, we want the minimum angle between $\mathbf{w}$ and $\mathbf{u}$. So, we have to maximize the $\mathbf{w^T u}$ value. Now value of $\mathbf{w^T u}$ can increase due to two reasons,

- Value of $||\mathbf{w}||_2$ increases
- The angle between them decreases, i.e. $\mathbf{w}$ comes closer to $\mathbf{u}$

But, we want to minimize the angle, so we keep track of $||\mathbf{w}||_2$ to check that the angle between the vectors is decreasing.

First, we will see what happens to $\mathbf{w^T u}$ after each iteration. Let the weight at i^{th} iteration be $\mathbf{w}_i$ and after update be $\mathbf{w}_{i+1}$. The difference between $\mathbf{w_{i+1}^T u}$ and $\mathbf{w_i^T u}$ is,

$$\begin{aligned}\mathbf{w_{i+1}^T u} - \mathbf{w_i^T u} &= (\mathbf{w}_i + y\mathbf{x})^\mathbf{T}\mathbf{u} - \mathbf{w_i^T u}\\ \mathbf{w_{i+1}^T u} &= \mathbf{w_i^T u} + y\mathbf{x^T u}\\ \mathbf{w_{i+1}^T u} &\geq \mathbf{w_i^T u} + \gamma\end{aligned}$$

Since, $\mathbf{w_0}$ is initialised to 0.

$$\boxed{\mathbf{w_k^T u} \geq k\gamma} \tag{2}$$

Now, $||\mathbf{w}||_2^2$ at each weight update increases by at most 1.

$$\begin{aligned}
\|\mathbf{w} + y\mathbf{x}\|^2 &= (\mathbf{w} + y\mathbf{x})^\mathsf{T}(\mathbf{w} + y\mathbf{x}) \\
&= \|\mathbf{w}\|^2 + 2y\mathbf{w}^\mathbf{T}\mathbf{x} + y^2\|\mathbf{x}\|^2 \\
&= \|\mathbf{w}\|^2 + 2y\mathbf{w}^\mathbf{T}\mathbf{x} + \|\mathbf{x}\|^2 \\
&\leq \|\mathbf{w}\|^2 + \|\mathbf{x}\|^2 && \because y\mathbf{w}^\mathbf{T}\mathbf{x} \leq 0 \\
&\leq \|\mathbf{w}\|^2 + 1 && \because \text{assumption that } \|\mathbf{x}\| \leq 1
\end{aligned}$$

Hence, after **k** weight updates,

$$\|\mathbf{w}_k\|^2 \leq \|\mathbf{w}_0\|^2 + k$$

since $\mathbf{w}_0$ is initialised to 0.

$$\boxed{\|\mathbf{w}_k\|^2 \leq k} \tag{3}$$

so from (2) and (3) we get,

$$\sqrt{k} \geq \|\mathbf{w_k}\| \geq \mathbf{w_k^T u} \geq k\gamma$$

$$\boxed{k \leq \frac{1}{\gamma^2}}$$

□

Hence, the number of mistakes made by the perceptron algorithm is bounded. As the perceptron algorithm updates weight only when there is a mistake, the mistake bound implies that the convergence is guaranteed.

Why is k independent of the number of examples?

Suppose the mistake in predicting an example $(\mathbf{x}, y)$ caused a weight update. But the change in hyperplane won't just correctly classify this point $(\mathbf{x}, y)$. It can classify multiple points correctly. Hence, the total number of iterations for the perceptron to converge does not depend on the number of examples in the dataset.

Limitation of Perceptron Algorithm

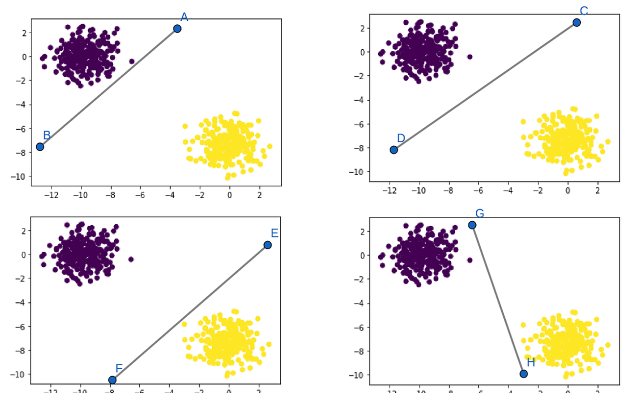


Figure 1: Possible Separation Hyperplanes

Consider Figure 1. All the hyperplanes AB, CD, EF, and GH classify data correctly. But, the hyperplanes AB, EF and GH on slight perturbation to the dataset will give incorrect predictions for examples that are closer to the hyperplane.

Hyperplane CD will give correct prediction even if there are perturbations because the closest point to the hyperplane is at the maximum distance for both groups. The problem with the perceptron algorithm is that it does not account for the margin of separation while choosing the hyperplane.

Objective: Maximize the margin of separation while correctly predicting all data points. Hence, the desirable hyperplane is the one whose margin of separation is higher for both groups while predicting all the training points correctly.

One of the possible solutions to this problem is a **Support Vector Machine**, which maximizes the margin while predicting correct values.