

Hyperplane ($\vec{x} \in \mathbb{R}^n$): In P -dimension, Hyperplane is flat affine space in $P-1$ -dimension.

• Affine space: An affine space is what is left of a vector space after you've forgotten which point is the origin.

↳ kind of subspace of Vector space

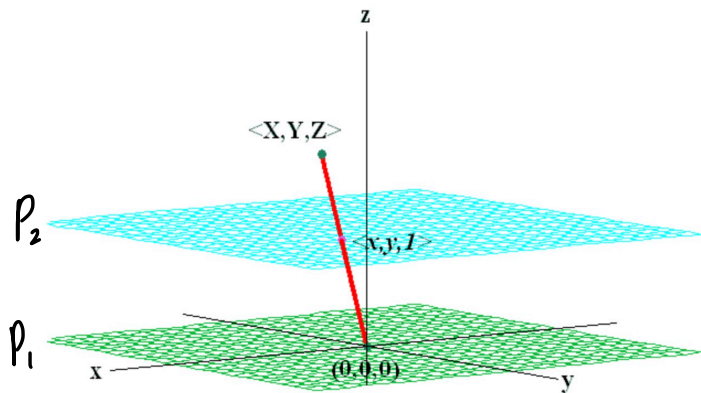
↳ In Vector space, all of vector is regard as same if those vectors have same norm / direction. Those are clearly different vector, but they are regard as same vector.

↳ So Affine space = Vector space + location (by point)

↳ ex) In 2-dimension space, $y=2x+1$ isn't including $(0,0)$. It is still Line.

⇒ Affine space $\Rightarrow W_1x_1 + W_2x_2 + \dots + W_px_p = b$ $\nexists \vec{0}$

Consider the vector space \mathbb{R}^3 . Inside \mathbb{R}^3 we can choose two planes, as in the picture below. We'll call the green one P_1 and the blue one P_2 . The plane P_1 passes through the origin but the plane P_2 does not. It is a standard homework exercise in linear algebra to show that the P_1 is a **sub-vector space** of \mathbb{R}^3 but the plane P_2 is **not**. However, the plane P_2 **looks** almost exactly the same as P_1 , having the exact same, flat geometry, and in fact P_2 and P_1 are simply translates of one another. This plane P_2 is a classical example of an affine space.



⇒ Affine space = subspace but it isn't including origin $(0,0)$



• Hyperplane $\Rightarrow W^T x + b = 0$ 인 직선

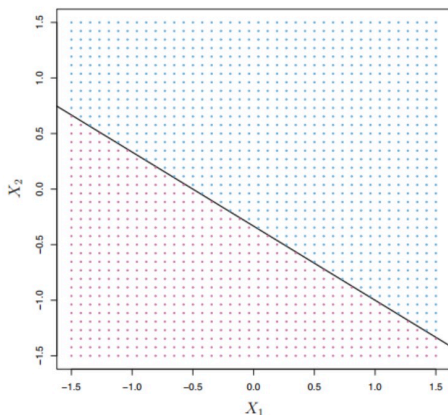


FIGURE 9.1. The hyperplane $1 + 2X_1 + 3X_2 = 0$ is shown. The blue region is the set of points for which $1 + 2X_1 + 3X_2 > 0$, and the purple region is the set of points for which $1 + 2X_1 + 3X_2 < 0$.

Q. Weight value W 와 hyperplane θ 가 다른 개념이
 We initialize $W^{(0)} = [2, 3]$, $W_0^{(0)} = 1$. So $\theta^{(0)} = W^{(0)T} x + W_0^{(0)}$
 $= \begin{bmatrix} 2 \\ 3 \end{bmatrix}^T x + 1$

↳ This is not data! it's variable.

So if $W_0 = 0$, which means hyperplane goes through the origin, $W = \theta$.

p차원 공간에서, 초평면은 p-1 차원의 평평한 affine 부분 공간입니다. 예를 들어, 2차원 공간에서 초평면은 평평한 1차원 부분공간입니다. 즉, 선입니다. 3차원에서 초평면은 평평한 2차원 부분공간이며 이는 평면입니다. p > 3인 경우에는 초평면을 시각화하기가 어렵지만 p-1 차원의 평평한 부분 공간인 것은 여전히 유효합니다.

2차원에서 초평면의 수학적 정의는 다음의 방정식으로 정의됩니다.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 = 0$$

2차원의 초평면은 1차원 선입니다. 2차원 파라미터 $\beta_0, \beta_1, \beta_2$ 가 존재하여 X_1, X_2 는 초평면 상 임의의 점입니다.

위 식은 임의의 p차원으로 확장할 수 있습니다. 즉 p차원에 대한 초평면은 아래의 식으로 정의됩니다.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p = 0$$

p차원 공간의 점 X_1, X_p 가 위 식을 만족하면 X는 초평면 상에 있습니다.

X가 아래 식을 만족하면 X는 초평면의 한쪽에 놓인다는 것을 의미합니다.

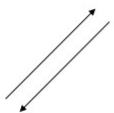
$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p > 0.$$

반면에 아래 식을 만족하면 X는 초평면의 다른 한쪽에 놓입니다.

$$\beta_0 + \beta_1 X_1 + \beta_2 X_2 + \dots + \beta_p X_p < 0.$$

따라서 초평면은 p차원 공간을 두개로 분할한다고 생각할 수 있습니다. X가 분리된 두 공간 중 어느 부분에 속하는지 판별하려면 초평면 식에 X를 대입하여 0보다 큰지 작은지를 확인하면 됩니다.

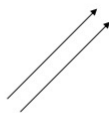
• Cosine similarity: Similarity between two vectors is calculated via Cosine angle



코사인 유사도 : -1



코사인 유사도 : 0



코사인 유사도 : 1

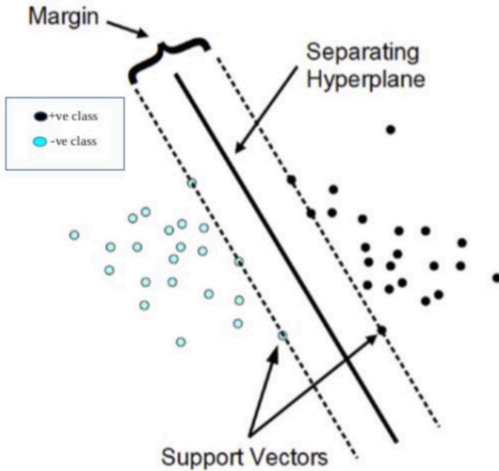
$$\text{Similarity} = \cos(\theta)$$

$$= \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Angle between A and B

$$(Using A \cdot B = \|A\| \cdot \|B\| \cdot \cos(\theta))$$

Margin



Margin: Referring shortest distance from each training observation to the given hyper-plane.

So hyperplane that has biggest margin is optimal separating hyperplane.

Formula: $y_i \cdot \frac{\theta^T x_i + \theta_0}{\|\theta\|}$ (θ is hyperplane, (x, y) is labeled data)

$= y_i \cdot \frac{W^T x_i + b}{\|W\|}$ (Cosine distance)

So classifier divide properly, y_i and $W^T x_i + b (= \hat{y}_i)$ have the same sign. \Rightarrow Positive value

한번 예시를 통해서 이해해보고자 한다. 위의 그림과 같은 linear classifier, h 가 있고, 이를 구성하는 hyperplane θ , θ_0 이 있다고 가정해보자

$$\theta = \begin{bmatrix} 1 \\ -1 \end{bmatrix}, \theta_0 = 1$$

그러면 이 hyperplane을 이용해서 linear classifier를 표현하면 다음과 같이 된다.

$$\theta^T x + \theta_0 = 0$$

보다시피 위의 좌표계에는 3개의 점이 있는데, 각각의 margin은 다음과 같이 구할 수 있다.

$$y^{(1)} \cdot \frac{\theta^T x^{(1)} + \theta_0}{\|\theta\|} = \frac{1 \cdot 2 + 1}{\sqrt{2}} = \frac{2+1}{\sqrt{2}} = \frac{3}{\sqrt{2}}$$

$$y^{(2)} \cdot \frac{\theta^T x^{(2)} + \theta_0}{\|\theta\|} = 1 \cdot \frac{1 + 1}{\sqrt{2}} = \frac{2}{\sqrt{2}} = \sqrt{2}$$

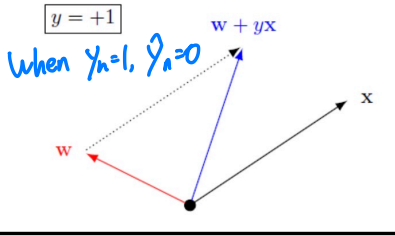
$$y^{(3)} \cdot \frac{\theta^T x^{(3)} + \theta_0}{\|\theta\|} = -1 \cdot \frac{-3 + 1}{\sqrt{2}} = \frac{-2}{\sqrt{2}} = -\sqrt{2}$$

참고로 $x^{(1)}$ 은 linear classifier가 잘못 구별했기 때문에, 이에 대한 margin도 음수가 나왔다. 그렇기 때문에 전체 dataset에 대한 margin은 위에서 구한 margin중 가장 작은 값이 되기 때문에 $-\frac{\sqrt{2}}{2}$ 가 되겠다.

Perceptron

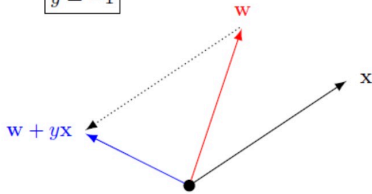
- Activation function [Unit step function: 0 or 1 $\frac{1}{2}$ at MS Edge
Sign function: -1 or 1]

But assume that Activation function is Unit step function. And assume $\eta=1$ / $b=0$ (in $z=w^T x + b$)



When $y_n=0, \hat{y}_n=1$

$y = -1$



The Perceptron implements $G(z) = \text{USF}(w^T x)$,
And perceptron picks misclassified point such that

$$\hat{y}_n \neq y_n$$

and updates $w \leftarrow w + \Delta w$

$$= w + \eta(y_n - \hat{y}_n)x_n,$$

$$\eta(y_n - \hat{y}_n)x_n = (y_n - \hat{y}_n)x_n = \begin{cases} 0 & \text{if } y_n = \hat{y}_n \\ -x_n & \text{if } y_n=0, \hat{y}_n=1 \\ x_n & \text{if } y_n=1, \hat{y}_n=0 \end{cases}$$

So we can update $w \leftarrow w + \Delta w = \begin{cases} w + x_n & \text{if } y_n=1, \hat{y}_n=0 \\ w - x_n & \text{if } y_n=0, \hat{y}_n=1 \end{cases}$

Thm. Linear separability of Perceptron (Using USF)

if $y_i(w^T x_i + b) \geq 0$, The dataset $D = \{(x_1, y_1), (x_2, y_2), \dots\}$ is linearly separable.

$\underbrace{\text{in } D_i}_{= \hat{y}_k}$ it means when $y_k=1, \hat{y}_k=1$
 $y_k=0, \hat{y}_k=0$ } real value = predict value

Why Perceptron converges when a classes are linearly separable?

ref: <https://nbviewer.org/github/metamath1/ml-simple-works/blob/master/perceptron/perceptron.ipynb> → Use sign function as activate function

• Perceptron convergence theorem ($y' = \hat{y}$, $y = Y$), $(y = \{-1, 1\})$ → 1 or -1 (Only for two-class)

Theorem. Assume that there exists some parameter vector θ^* such that $\|\theta^*\| = 1$, and some $\gamma > 0$ such that for all $i = 1 \dots n$, $y_i(\theta^* \cdot x_i) \geq \gamma$. Assume in addition that for all $i = 1 \dots n$, $\|x_i\| \leq R$. Then the perceptron algorithm makes at most $\frac{R^2}{\gamma^2}$ errors. (The definition of an error is as follows: an error occurs whenever we have $y_i \neq \hat{y}_i$ in the algorithm.)

So ideal hyperplane $\theta^* \leftarrow w^T x + b = 0$, and assume all of the above assumptions. (①~③)

And also say $\theta^{(0)} = 0$ (Initial value = 0), $\theta^{(k)}$ = The hyperplane after k^{th} misclassification. (= after k^{th} update)

b through Origin → 원점을 가장 근접하게 끌고 더 정확함

$$\text{And } y_i(\theta^* \cdot x_i) = \frac{y_i(\theta^* \cdot x_i)}{\|\theta^*\|} \quad (\because \|\theta^*\| = 1 \text{ is assumed})$$

γ = margin $> 0 \Rightarrow$ it means in θ^* , Predictions are always True. = Linearly separable

Goal of Perceptron is current hyperplane similarity changing to ideal hyperplane θ^* .

We can evaluate similarity between $\theta^{(k)}$ and θ^* via Cosine similarity.

$$\cos(\theta^{(k)}, \theta^*) = \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^{(k)}\| \cdot \|\theta^*\|} = \left(\frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|^2} \right) \cdot \left(\frac{1}{\|\theta^{(k)}\|} \right)$$

$$\Delta W = \eta(y_i - \hat{y}_i)x_i \quad \text{in } y = \{0, 1\}$$

$$= \eta y_i x_i \quad \text{in } y = \{-1, 1\}$$

1)

$$\frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} \text{ or } y_i \neq \hat{y}_i \text{ then } \theta^{(k)} = (W_{i-1} + \Delta W)^T x + W_0 + \Delta W_0$$

$$= (W_{i-1} + y_i x_i) x + 0 \quad \text{s.t. } W_0 = 0 \text{ \& } y = \{-1, 1\}$$

$$= \theta^{(k-1)} + y_i x_i \quad \because \text{both } W_{i-1} \text{ and } y_i x_i \text{ is Vector.}$$

x_i 가 1이면 y_i

Equation
↓
Vector comparison

$$\therefore \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} = \frac{(\theta^{(k-1)} + y_i x_i) \cdot \theta^*}{\|\theta^*\|} = \frac{\theta^{(k-1)} \cdot \theta^*}{\|\theta^*\|} + \frac{y_i (x_i \cdot \theta^*)}{\|\theta^*\|}$$

$$\geq \frac{\theta^{(k-1)} \cdot \theta^*}{\|\theta^*\|} + \gamma$$

Each update of $\theta^{(k)}$ accumulate $+ y_i x_i$. if $\theta^{(0)} = 0$, θ^1 will be $y_1 x_1$,
 θ^2 will be $2 \cdot y_1 x_1 + x_2 \dots$

$$\text{So } \frac{\theta^{(k+1)} \cdot \theta^*}{\|\theta^*\|} = \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} + \gamma \geq k\gamma \quad \therefore \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} \geq k\gamma$$

$$2) \text{ In } \frac{1}{\|\theta^{(k)}\|},$$

$$\|\theta^{(k)}\|^2 = \|\theta^{(k-1)} + y_i x_i\|^2 = \|\theta^{(k-1)}\|^2 + 2y_i \theta^{(k-1)T} x_i + \|x_i\|^2$$

if (x_i, y_i) is classified incorrectly, $y_i \theta^{(k-1)T} x_i \leq 0$

$$\leq \|\theta^{(k-1)}\|^2 + \|x_i\|^2$$

$$\leq \|\theta^{(k-1)}\|^2 + R^2 \quad \text{s.t. } \|x_i\| \leq R$$

$$\leq \|\theta^{(0)} + k(y_i x_i)\|^2 + R^2$$

$$\leq kR^2$$

$$\leq kR^2, \quad \therefore \frac{1}{\|\theta^{(k)}\|} \geq \frac{1}{\sqrt{k}R}$$

$\therefore \cos(\theta^{(k)}, \theta^*) \geq (kR) \cdot \frac{1}{\sqrt{k}R} = \sqrt{k} \cdot \frac{\sigma}{R}$, since the value of the cosine is at most 1, we have

$$1 \geq \sqrt{k} \cdot \frac{\sigma}{R}$$

$$k \leq \left(\frac{R}{\sigma}\right)^2$$

So When using the Perceptron algorithm for classification.

at most $\left(\frac{R}{\sigma}\right)^2$ classification errors will be made. \square

Ref

CHAPTER 3

The Perceptron

First of all, the coolest algorithm name! It is based on the 1943 model of neurons made by McCulloch and Pitts and by Hebb. It was developed by Rosenblatt in 1962. At the time, it was not interpreted as attempting to optimize any particular criteria; it was presented directly as an algorithm. There has, since, been a huge amount of study and analysis of its convergence properties and other aspects of its behavior.

Well, maybe “neocognitron,” also the name of a real ML algorithm, is cooler.

1 Algorithm

Recall that we have a training dataset \mathcal{D}_n with $x \in \mathbb{R}^d$, and $y \in \{-1, +1\}$. The Perceptron algorithm trains a binary classifier $h(x; \theta, \theta_0)$ using the following algorithm to find θ and θ_0 using τ iterative steps:

PERCEPTRON(τ, \mathcal{D}_n)

```
1   $\theta = [0 \ 0 \ \dots \ 0]^T$ 
2   $\theta_0 = 0$ 
3  for  $t = 1$  to  $\tau$ 
4      for  $i = 1$  to  $n$ 
5          if  $y^{(i)} (\theta^T x^{(i)} + \theta_0) \leq 0$ 
6               $\theta = \theta + y^{(i)} x^{(i)}$ 
7               $\theta_0 = \theta_0 + y^{(i)}$ 
8  return  $\theta, \theta_0$ 
```

We use Greek letter τ here instead of T so we don't confuse it with transpose!

Intuitively, on each step, if the current hypothesis θ, θ_0 classifies example $x^{(i)}$ correctly, then no change is made. If it classifies $x^{(i)}$ incorrectly, then it moves θ, θ_0 so that it is “closer” to classifying $x^{(i)}, y^{(i)}$ correctly.

Note that if the algorithm ever goes through one iteration of the loop on line 4 without making an update, it will never make any further updates (verify that you believe this!) and so it should just terminate at that point.

Let's check dimensions. Remember that θ is $d \times 1$, $x^{(i)}$ is $d \times 1$, and $y^{(i)}$ is a scalar. Does everything match?

Study Question: What is true about \mathcal{E}_n if that happens?

Example: Let h be the linear classifier defined by $\theta^{(0)} = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\theta_0^{(0)} = 1$. The diagram below shows several points classified by h . However, in this case, h (represented by the bold line) misclassifies the point $x^{(1)} = \begin{bmatrix} 1 \\ 3 \end{bmatrix}$ which has label $y^{(1)} = 1$. Indeed,

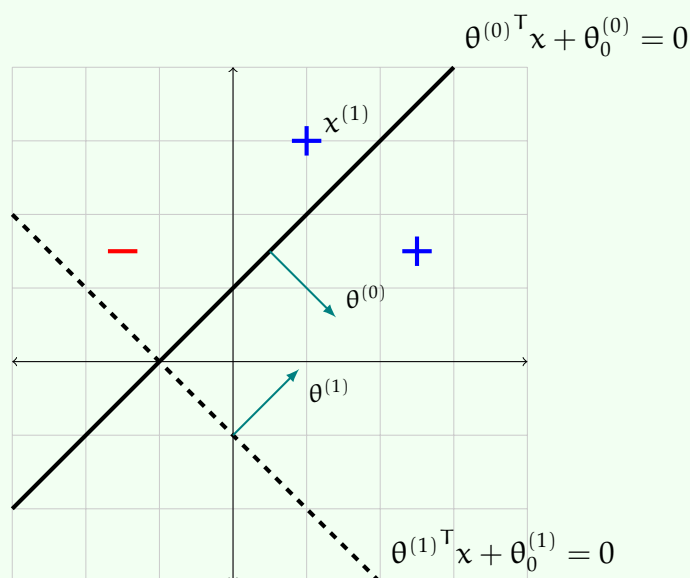
$$y^{(1)} (\theta^T x^{(1)} + \theta_0) = [1 \quad -1] \begin{bmatrix} 1 \\ 3 \end{bmatrix} + 1 = -1 < 0$$

By running an iteration of the Perceptron algorithm, we update

$$\theta^{(1)} = \theta^{(0)} + y^{(1)} x^{(1)} = \begin{bmatrix} 2 \\ 2 \end{bmatrix}$$

$$\theta_0^{(1)} = \theta_0^{(0)} + y^{(1)} = 2$$

The new classifier (represented by the dashed line) now correctly classifies that point, but now makes a mistake on the negatively labeled point.



A really important fact about the perceptron algorithm is that, if there is a linear classifier with 0 training error, then this algorithm will (eventually) find it! We'll look at a proof of this in detail, next.

2 Offset

Sometimes, it can be easier to implement or analyze classifiers of the form

$$h(x; \theta) = \begin{cases} +1 & \text{if } \theta^T x > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Without an explicit offset term (θ_0), this separator must pass through the origin, which may appear to be limiting. However, we can convert any problem involving a linear separator *with* offset into one with *no* offset (but of higher dimension)!

Consider the d -dimensional linear separator defined by $\theta = [\theta_1 \ \theta_2 \ \dots \ \theta_d]$ and offset θ_0 .

- to each data point $x \in \mathcal{D}$, append a coordinate with value $+1$, yielding

$$x_{\text{new}} = [x_1 \ \dots \ x_d \ +1]^T$$

- define

$$\theta_{\text{new}} = [\theta_1 \ \dots \ \theta_d \ \theta_0]^T$$

Then,

$$\begin{aligned} \theta_{\text{new}}^T \cdot x_{\text{new}} &= \theta_1 x_1 + \dots + \theta_d x_d + \theta_0 \cdot 1 \\ &= \theta^T x + \theta_0 \end{aligned}$$

Thus, θ_{new} is an equivalent $((d+1)$ -dimensional) separator to our original, but with no offset.

Consider the data set:

$$\begin{aligned} X &= [[1], [2], [3], [4]] \\ Y &= [[+1], [+1], [-1], [-1]] \end{aligned}$$

It is linearly separable in $d = 1$ with $\theta = [-1]$ and $\theta_0 = 2.5$. But it is not linearly separable through the origin! Now, let

$$X_{\text{new}} = \begin{bmatrix} [1] & [2] & [3] & [4] \\ [1] & [1] & [1] & [1] \end{bmatrix}$$

This new dataset is separable through the origin, with $\theta_{\text{new}} = [-1, 2.5]^T$.

We can make a simplified version of the perceptron algorithm if we restrict ourselves to separators through the origin:

PERCEPTRON-THROUGH-ORIGIN(τ, \mathcal{D}_n)

```

1   $\theta = [0 \ 0 \ \dots \ 0]^T$ 
2  for  $t = 1$  to  $\tau$ 
3      for  $i = 1$  to  $n$ 
4          if  $y^{(i)} (\theta^T x^{(i)}) \leq 0$ 
5               $\theta = \theta + y^{(i)} x^{(i)}$ 
6  return  $\theta$ 
```

We list it here because this is the version of the algorithm we'll study in more detail.

3 Theory of the perceptron

Now, we'll say something formal about how well the perceptron algorithm really works. We start by characterizing the set of problems that can be solved perfectly by the perceptron algorithm, and then prove that, in fact, it can solve these problems. In addition, we provide a notion of what makes a problem difficult for perceptron and link that notion of difficulty to the number of iterations the algorithm will take.

3.1 Linear separability

A training set \mathcal{D}_n is *linearly separable* if there exist θ, θ_0 such that, for all $i = 1, 2, \dots, n$:

$$y^{(i)} (\theta^T x^{(i)} + \theta_0) > 0 .$$

Another way to say this is that all predictions on the training set are correct:

$$h(x^{(i)}; \theta, \theta_0) = y^{(i)} .$$

And, another way to say this is that the training error is zero:

$$\mathcal{E}_n(h) = 0 .$$

3.2 Convergence theorem

The basic result about the perceptron is that, if the training data \mathcal{D}_n is linearly separable, then the perceptron algorithm is guaranteed to find a linear separator.

We will more specifically characterize the linear separability of the dataset by the *margin* of the separator. We'll start by defining the margin of a point with respect to a hyperplane.

First, recall that the distance of a point x to the hyperplane θ, θ_0 is

$$\frac{\theta^T x + \theta_0}{\|\theta\|} .$$

Then, we'll define the *margin* of a *labeled point* (x, y) with respect to hyperplane θ, θ_0 to be

$$y \cdot \frac{\theta^T x + \theta_0}{\|\theta\|} .$$

This quantity will be positive if and only if the point x is classified as y by the linear classifier represented by this hyperplane.

Study Question: What sign does the margin have if the point is incorrectly classified? Be sure you can explain why.

Now, the *margin* of a *dataset* \mathcal{D}_n with respect to the hyperplane θ, θ_0 is the *minimum* margin of any point with respect to θ, θ_0 :

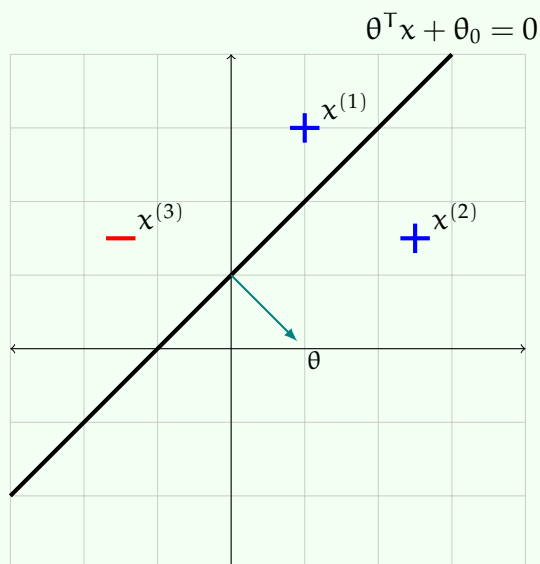
$$\min_i \left(y^{(i)} \cdot \frac{\theta^T x^{(i)} + \theta_0}{\|\theta\|} \right) .$$

The margin is positive if and only if all of the points in the data-set are classified correctly. In that case (only!) it represents the distance from the hyperplane to the closest point.

If the training data is *not* linearly separable, the algorithm will not be able to tell you for sure, in finite time, that it is not linearly separable. There are other algorithms that can test for linear separability with run-times $O(n^{d/2})$ or $O(d^{2n})$ or $O(n^{d-1} \log n)$.

Example: Let h be the linear classifier defined by $\theta = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$, $\theta_0 = 1$.

The diagram below shows several points classified by h , one of which is misclassified. We compute the margin for each point:



$$y^{(1)} \cdot \frac{\theta^T x^{(1)} + \theta_0}{\|\theta\|} = 1 \cdot \frac{-2 + 1}{\sqrt{2}} = -\frac{\sqrt{2}}{2}$$

$$y^{(2)} \cdot \frac{\theta^T x^{(2)} + \theta_0}{\|\theta\|} = 1 \cdot \frac{1 + 1}{\sqrt{2}} = \sqrt{2}$$

$$y^{(3)} \cdot \frac{\theta^T x^{(3)} + \theta_0}{\|\theta\|} = -1 \cdot \frac{-3 + 1}{\sqrt{2}} = \sqrt{2}$$

Note that since point $x^{(1)}$ is misclassified, its margin is negative. Thus the margin for the whole data set is given by $-\frac{\sqrt{2}}{2}$.

Theorem 3.1 (Perceptron Convergence). *For simplicity, we consider the case where the linear separator must pass through the origin. If the following conditions hold:*

- (a) *there exists θ^* such that $y^{(i)} \frac{\theta^{*T} x^{(i)}}{\|\theta^*\|} \geq \gamma$ for all $i = 1, \dots, n$ and for some $\gamma > 0$ and*
- (b) *all the examples have bounded magnitude: $\|x^{(i)}\| \leq R$ for all $i = 1, \dots, n$,*

then the perceptron algorithm will make at most $\left(\frac{R}{\gamma}\right)^2$ mistakes. At this point, its hypothesis will be a linear separator of the data.

Proof. We initialize $\theta^{(0)} = 0$, and let $\theta^{(k)}$ define our hyperplane after the perceptron algorithm has made k mistakes. We are going to think about the angle between the hypothesis we have now, $\theta^{(k)}$ and the assumed good separator θ^* . Since they both go through the origin, if we can show that the angle between them is decreasing usefully on every iteration, then we will get close to that separator.

So, let's think about the cos of the angle between them, and recall, by the definition of dot product:

$$\cos(\theta^{(k)}, \theta^*) = \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\| \|\theta^{(k)}\|}$$

We'll divide this up into two factors,

$$\cos(\theta^{(k)}, \theta^*) = \left(\frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} \right) \cdot \left(\frac{1}{\|\theta^{(k)}\|} \right), \quad (3.1)$$

and start by focusing on the first factor.

Without loss of generality, assume that the k^{th} mistake occurs on the i^{th} example $(x^{(i)}, y^{(i)})$.

$$\begin{aligned} \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} &= \frac{(\theta^{(k-1)} + y^{(i)} x^{(i)}) \cdot \theta^*}{\|\theta^*\|} \\ &= \frac{\theta^{(k-1)} \cdot \theta^*}{\|\theta^*\|} + \frac{y^{(i)} x^{(i)} \cdot \theta^*}{\|\theta^*\|} \\ &\geq \frac{\theta^{(k-1)} \cdot \theta^*}{\|\theta^*\|} + \gamma \\ &\geq k\gamma \end{aligned}$$

where we have first applied the margin condition from (a) and then applied simple induction.

Now, we'll look at the second factor in equation 3.1. We note that since $(x^{(i)}, y^{(i)})$ is classified incorrectly, $y^{(i)} (\theta^{(k-1)T} x^{(i)}) \leq 0$. Thus,

$$\begin{aligned} \|\theta^{(k)}\|^2 &= \|\theta^{(k-1)} + y^{(i)} x^{(i)}\|^2 \\ &= \|\theta^{(k-1)}\|^2 + 2y^{(i)} \theta^{(k-1)T} x^{(i)} + \|x^{(i)}\|^2 \\ &\leq \|\theta^{(k-1)}\|^2 + R^2 \\ &\leq kR^2 \end{aligned}$$

where we have additionally applied the assumption from (b) and then again used simple induction.

Returning to the definition of the dot product, we have

$$\cos(\theta^{(k)}, \theta^*) = \frac{\theta^{(k)} \cdot \theta^*}{\|\theta^{(k)}\| \|\theta^*\|} = \left(\frac{\theta^{(k)} \cdot \theta^*}{\|\theta^*\|} \right) \frac{1}{\|\theta^{(k)}\|} \geq (k\gamma) \cdot \frac{1}{\sqrt{k}R} = \sqrt{k} \cdot \frac{\gamma}{R}$$

Since the value of the cosine is at most 1, we have

$$\begin{aligned} 1 &\geq \sqrt{k} \cdot \frac{\gamma}{R} \\ k &\leq \left(\frac{R}{\gamma} \right)^2. \end{aligned}$$

□

This result endows the margin γ of \mathcal{D}_n with an operational meaning: when using the Perceptron algorithm for classification, at most $(R/\gamma)^2$ classification errors will be made, where R is an upper bound on the magnitude of the training vectors.