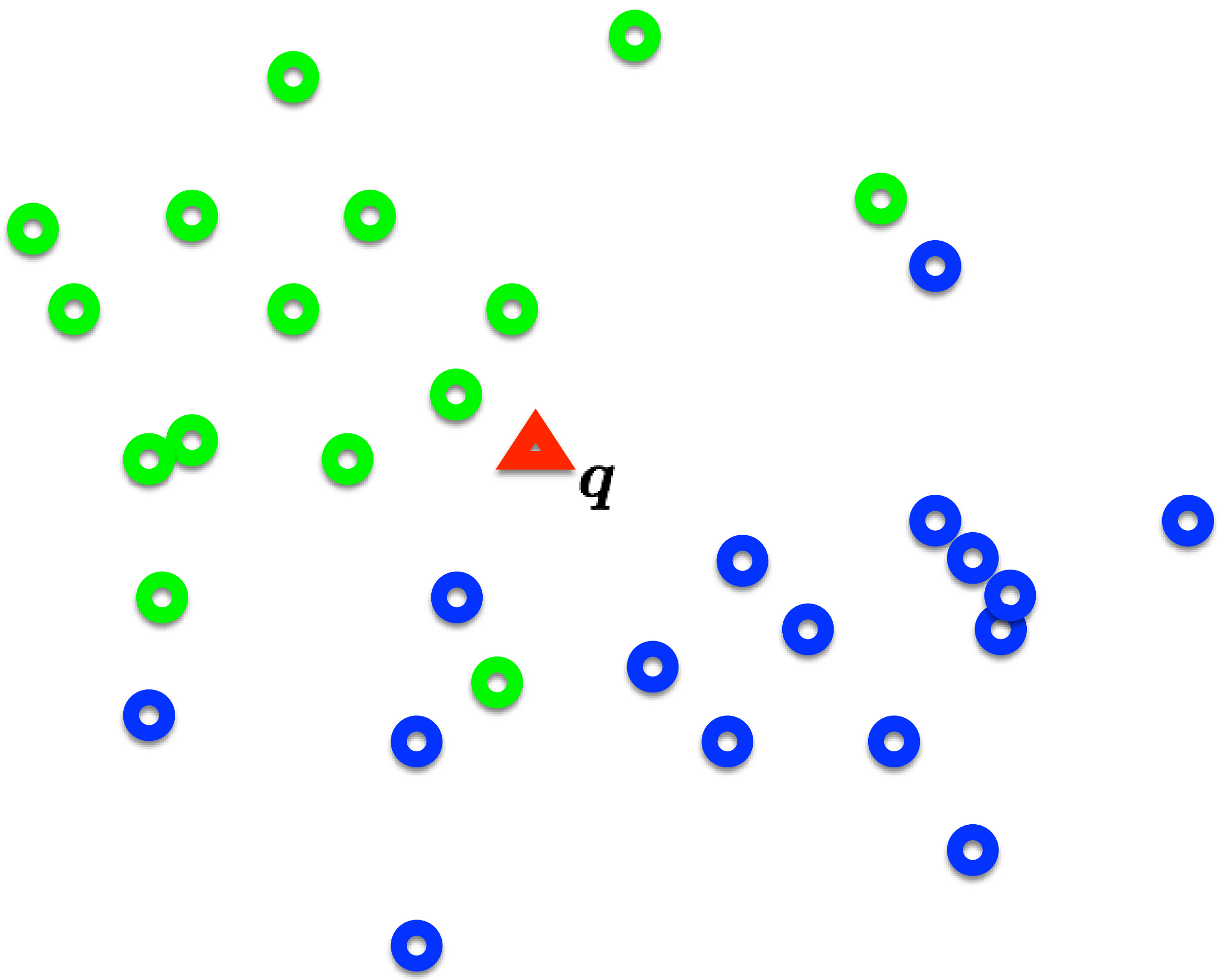
Lecture 6:
Machine Learning 2

Sung In Cho

Div. AISW
Dongguk Univ.

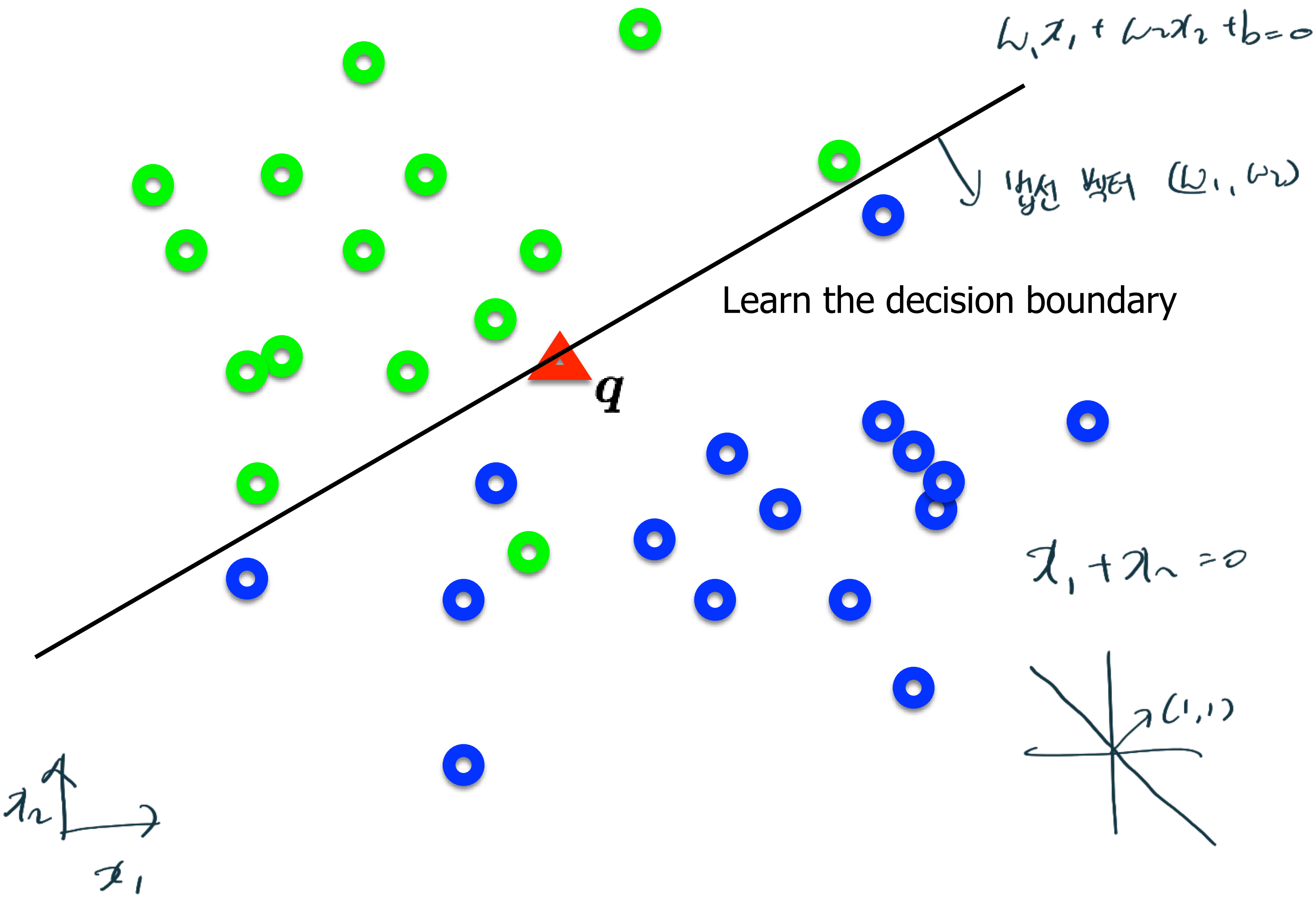
Concept of Support Vector Machine (SVM)

Distribution of data from two classes



Which class does q belong too?

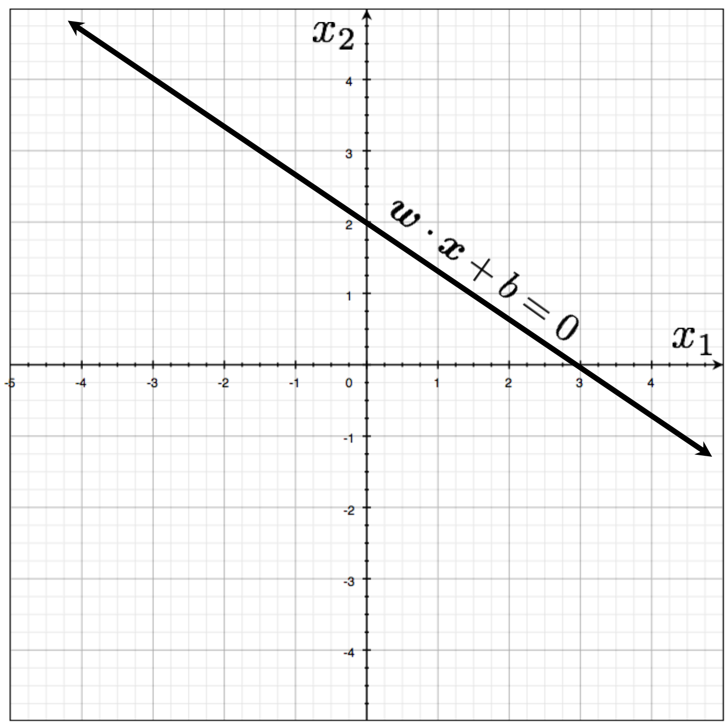
Distribution of data from two classes



First we need to understand hyperplanes…

Hyperplanes (lines) in 2D

$$w_1x_1 + w_2x_2 + b = 0$$



A line can be written as dot product plus a bias

$$[w_1 \ w_2] \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + b$$

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

$$\boldsymbol{w} \in \mathcal{R}^2$$

Another version, add a weight 1 and push the bias inside

$$[w_1 \ w_2 \ b] \begin{bmatrix} x_1 \\ x_2 \\ 1 \end{bmatrix}$$

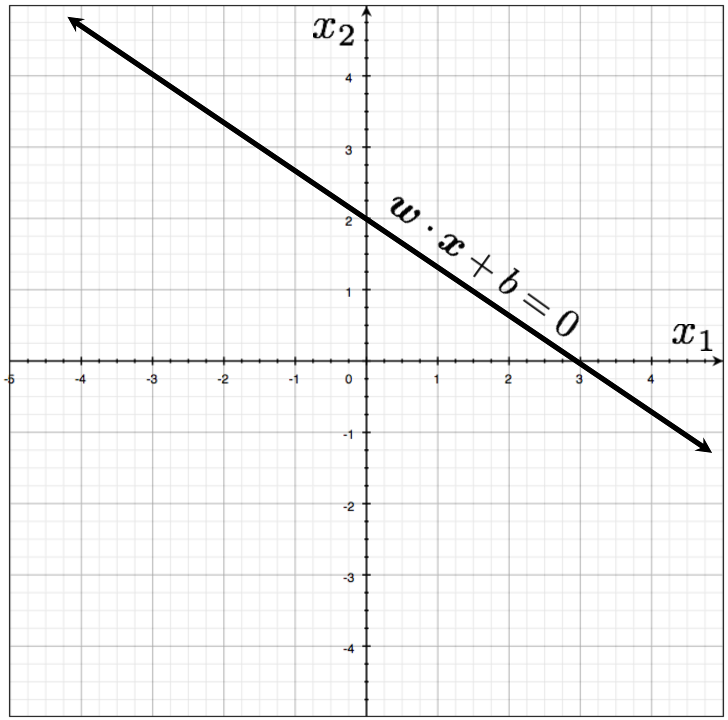
$$\boldsymbol{w} \cdot \boldsymbol{x} = 0$$

$$\boldsymbol{w} \in \mathcal{R}^3$$

Hyperplanes (lines) in 2D

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$ (offset/bias outside) $$\boldsymbol{w} \cdot \boldsymbol{x} = 0$$ (offset/bias inside)

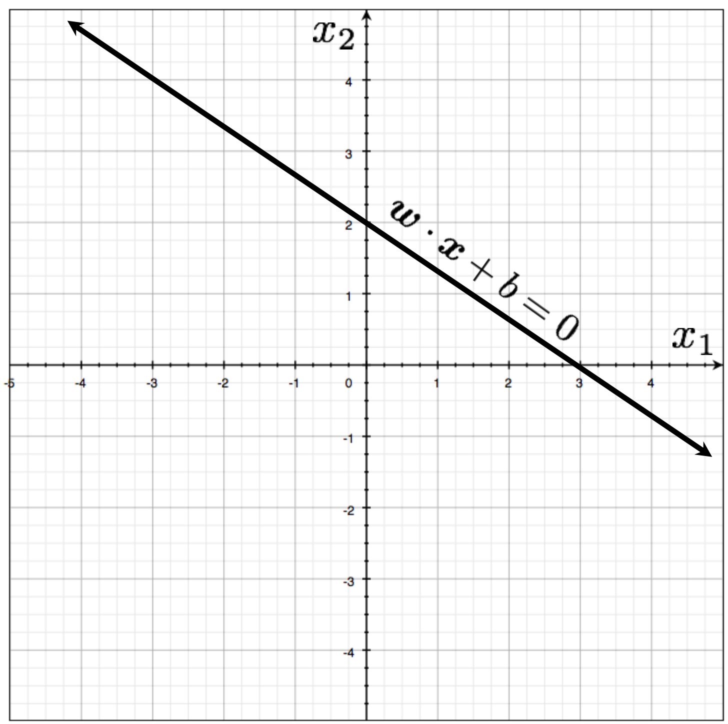
$$w_1 x_1 + w_2 x_2 + b = 0$$



Hyperplanes (lines) in 2D

$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ (offset/bias outside) $\boldsymbol{w} \cdot \boldsymbol{x} = 0$ (offset/bias inside)

$$w_1 x_1 + w_2 x_2 + b = 0$$



Important property:

Free to choose any normalization of w

The line

$$w_1 x_1 + w_2 x_2 + b = 0$$

and the line

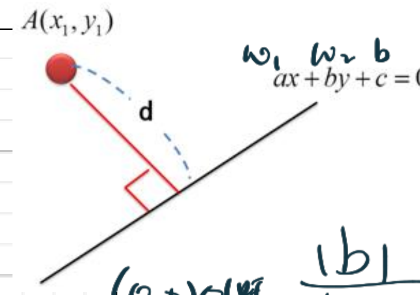
$$\lambda(w_1 x_1 + w_2 x_2 + b) = 0$$

define the same line

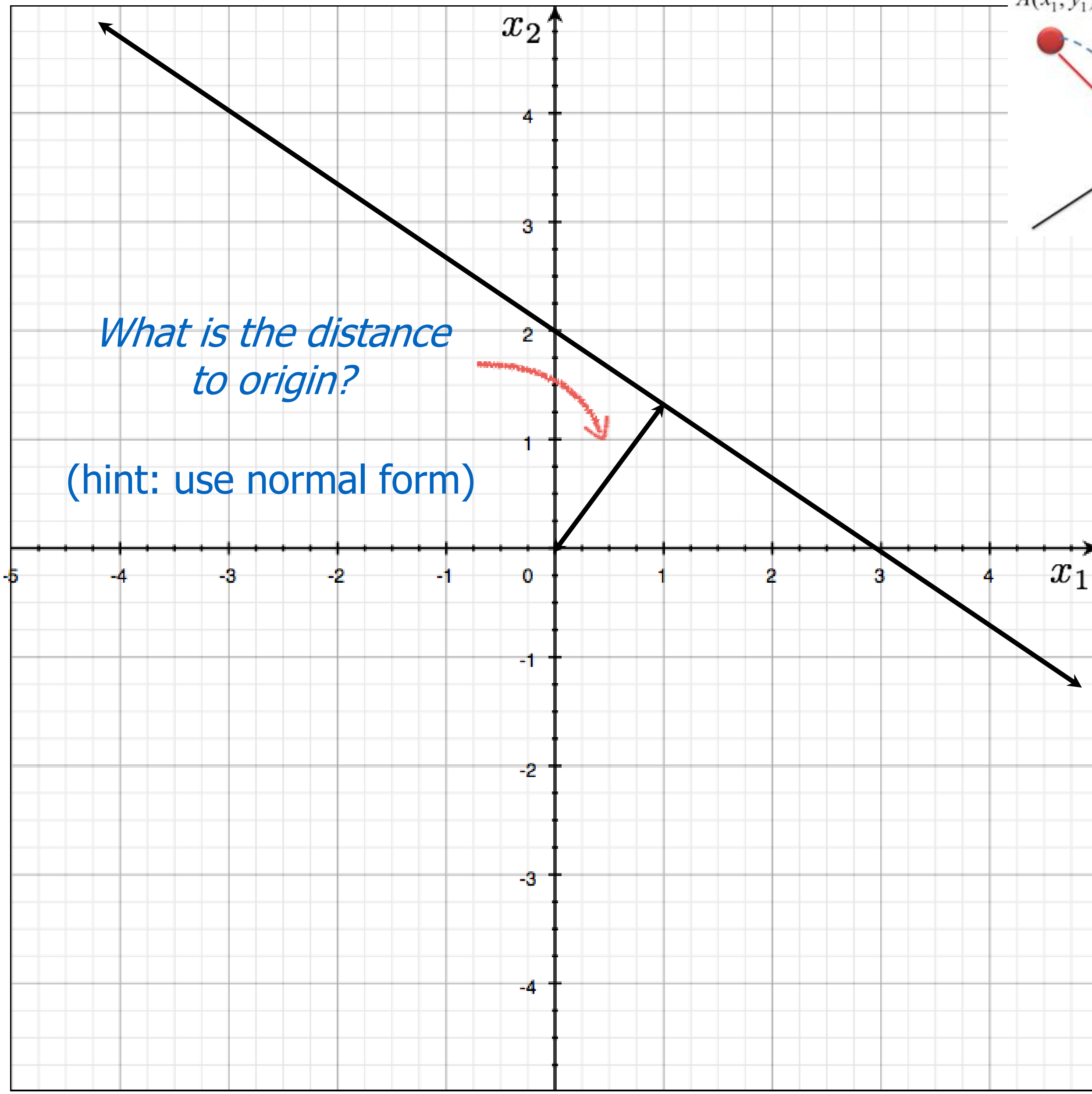
scaling을 해도

직선이 바뀌지 않음 → 방향성이 중요!

$$d = \frac{|ax_1 + by_1 + c|}{\sqrt{a^2 + b^2}}$$



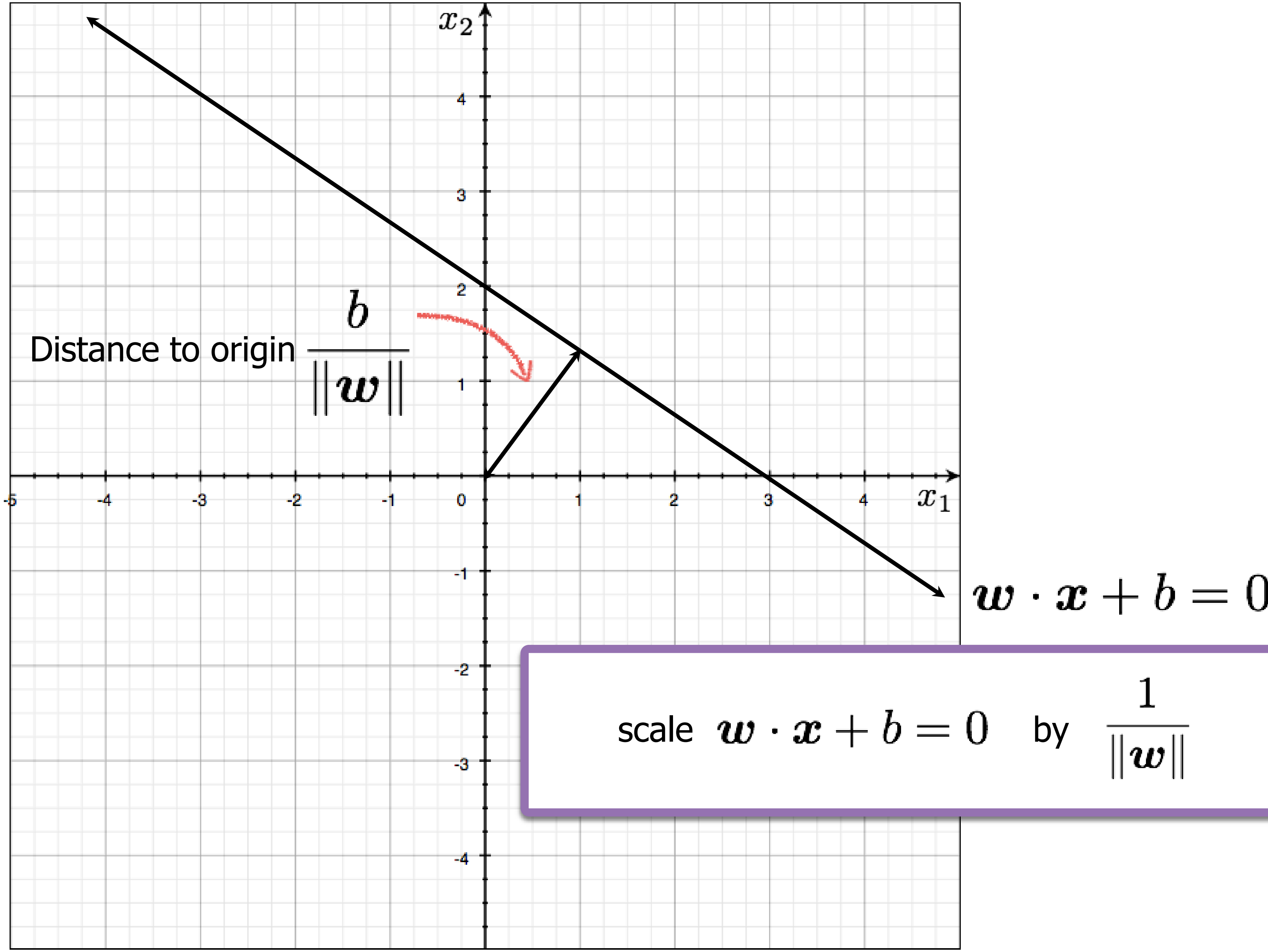
(0,0)이면, $\frac{|b|}{\|w\|}$



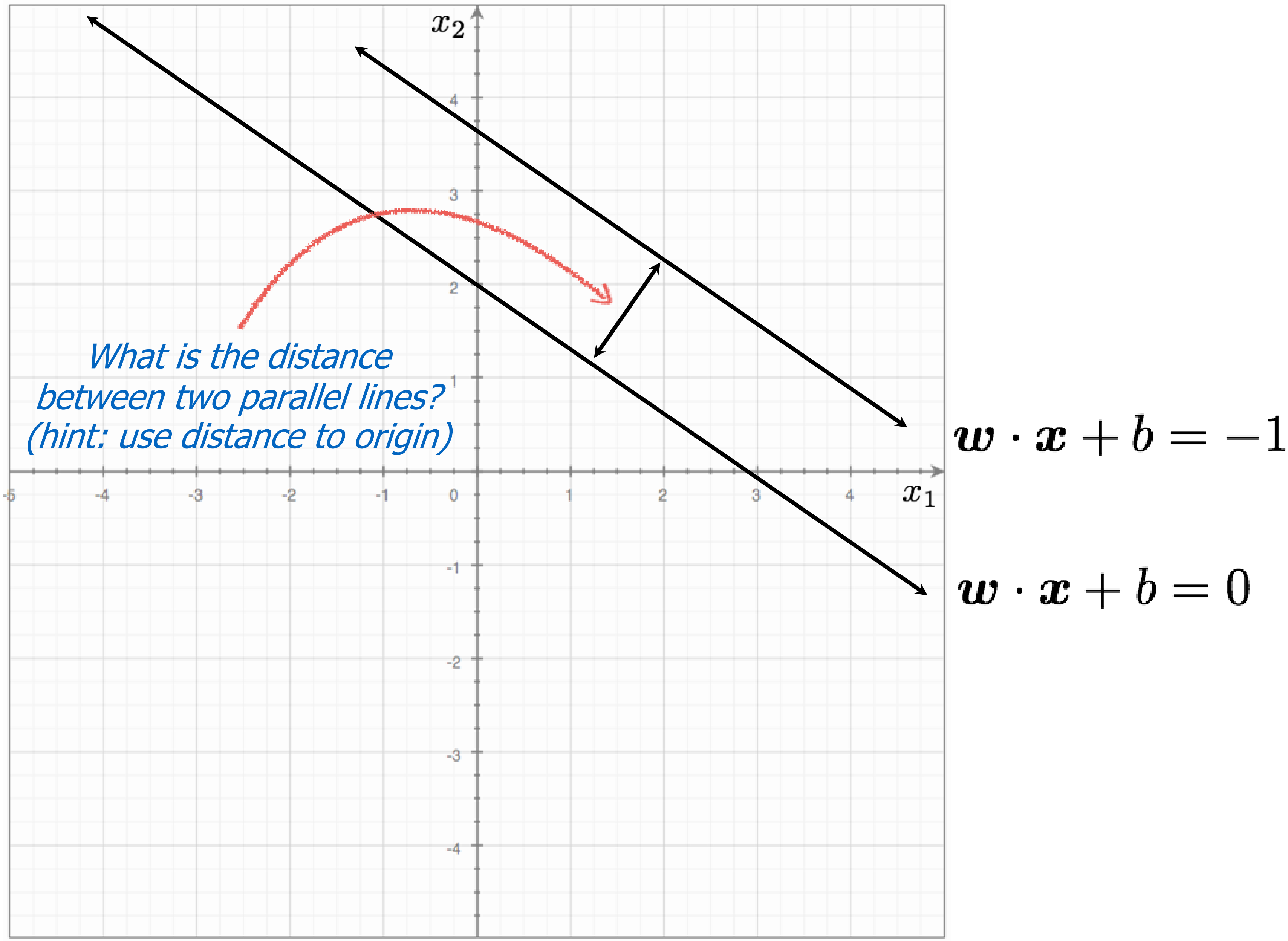
What is the distance to origin?

(hint: use normal form)

$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$



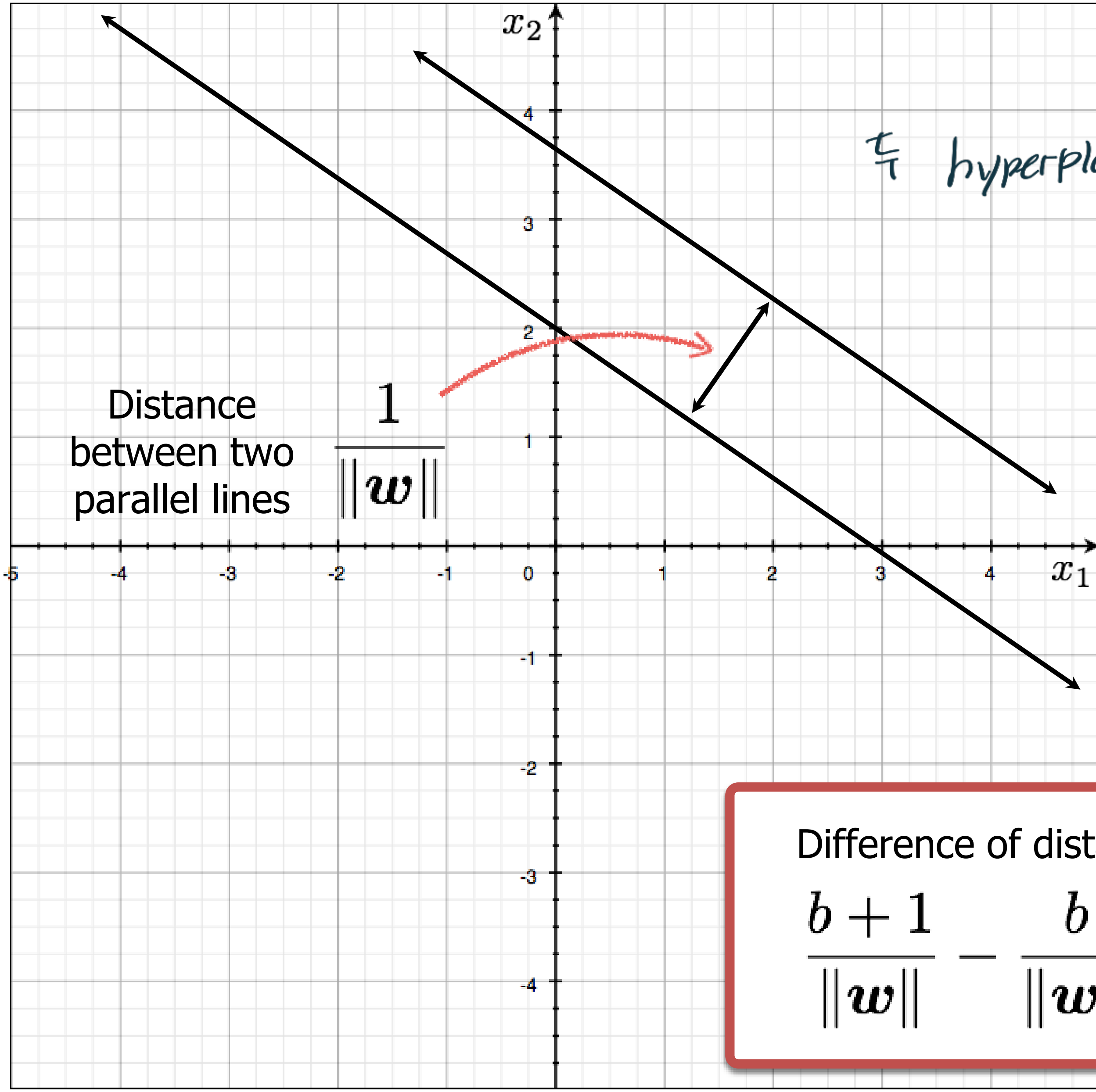
scale $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$ by $\frac{1}{\|\boldsymbol{w}\|}$



What is the distance between two parallel lines?
(hint: use distance to origin)
$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$
$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$
x_1
x_2

3D, 4 Dimension으로 바뀌어도 수식은 변하지 않는다.

두 hyperplane 간의 거리



Distance between two parallel lines $\dfrac{1}{\|\boldsymbol{w}\|}$

$\boldsymbol{w} \cdot \boldsymbol{x} + b = -1$

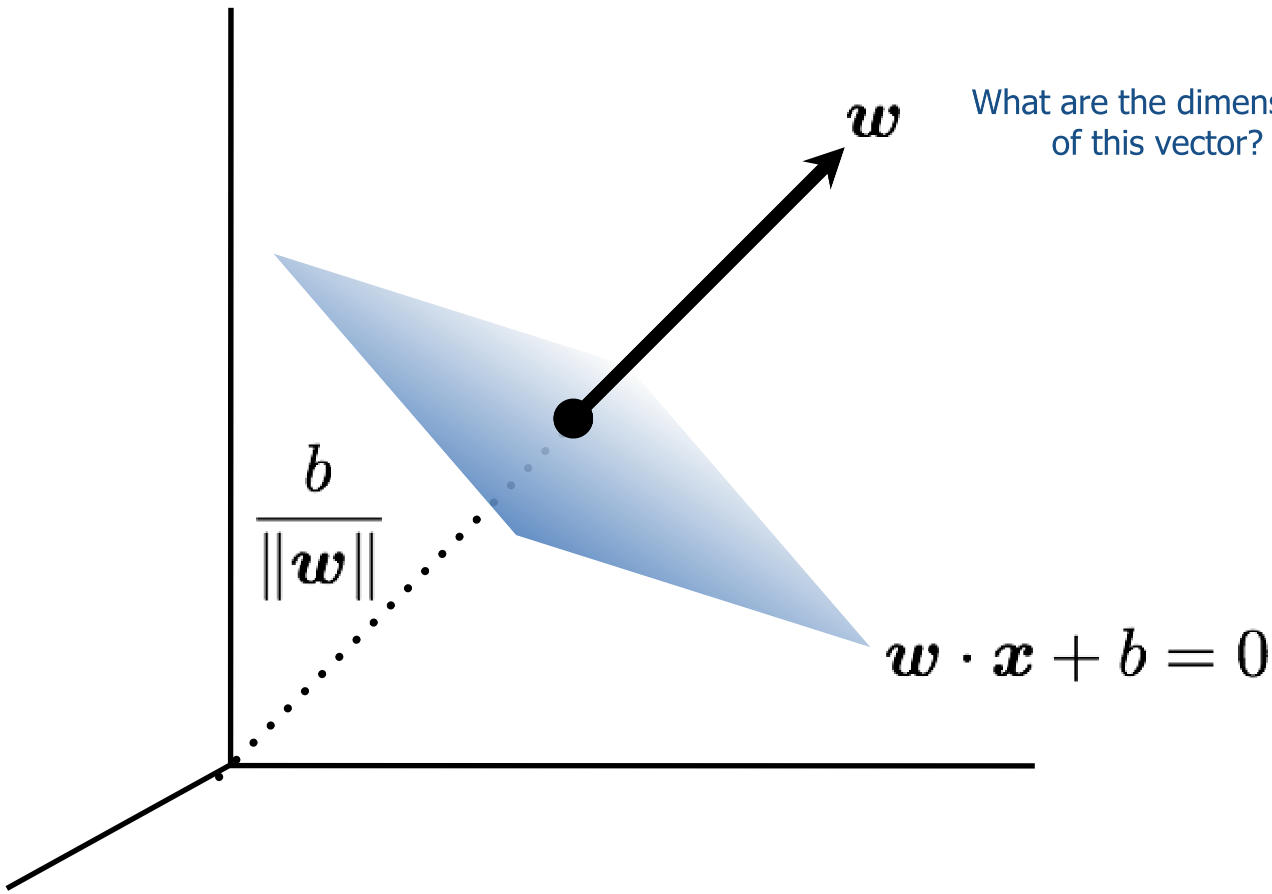
$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$

Difference of distance to origin

$$\frac{b+1}{\|\boldsymbol{w}\|} - \frac{b}{\|\boldsymbol{w}\|} = \frac{1}{\|\boldsymbol{w}\|}$$

Now we can go to 3D …

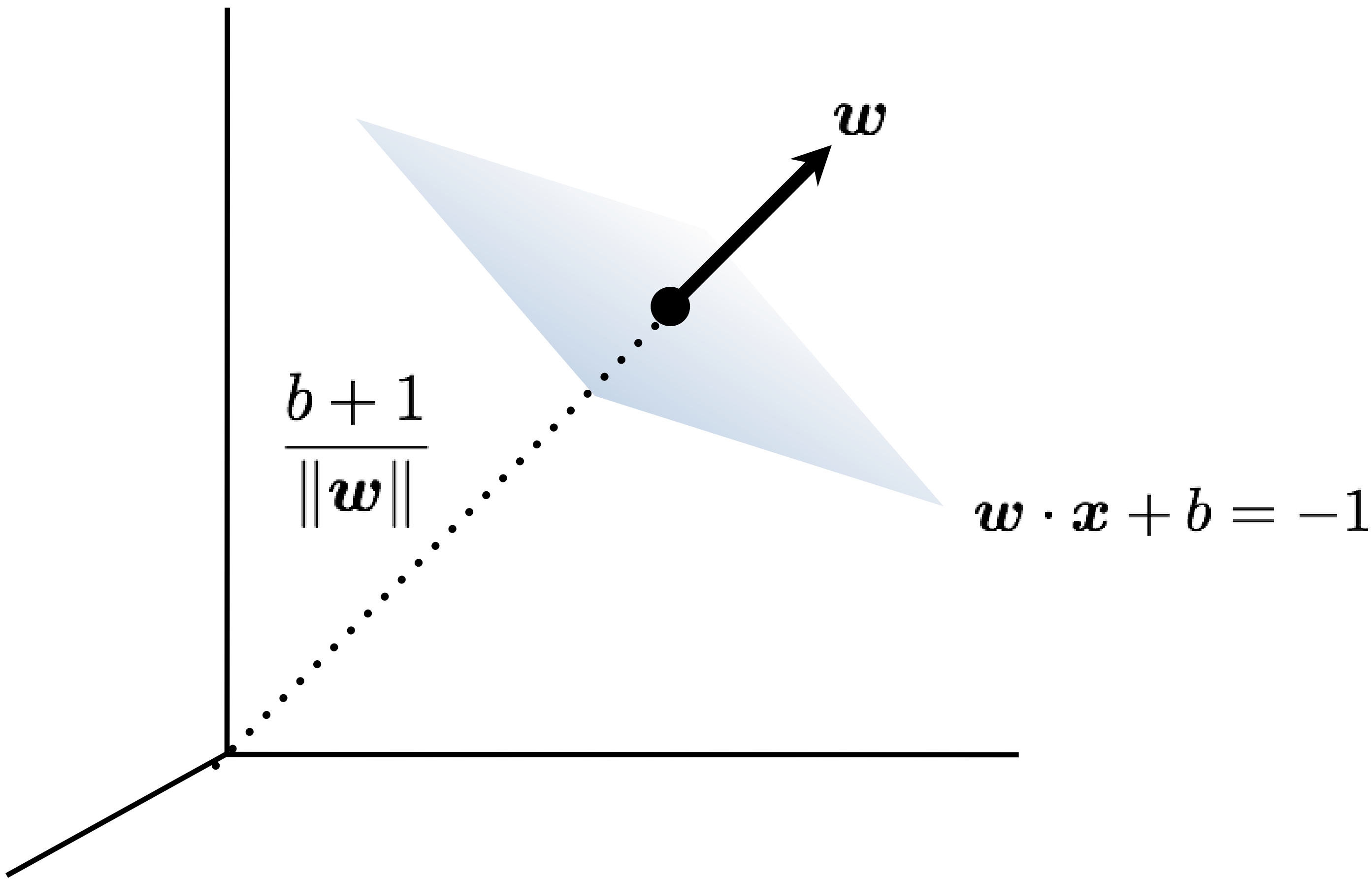
Hyperplanes (planes) in 3D



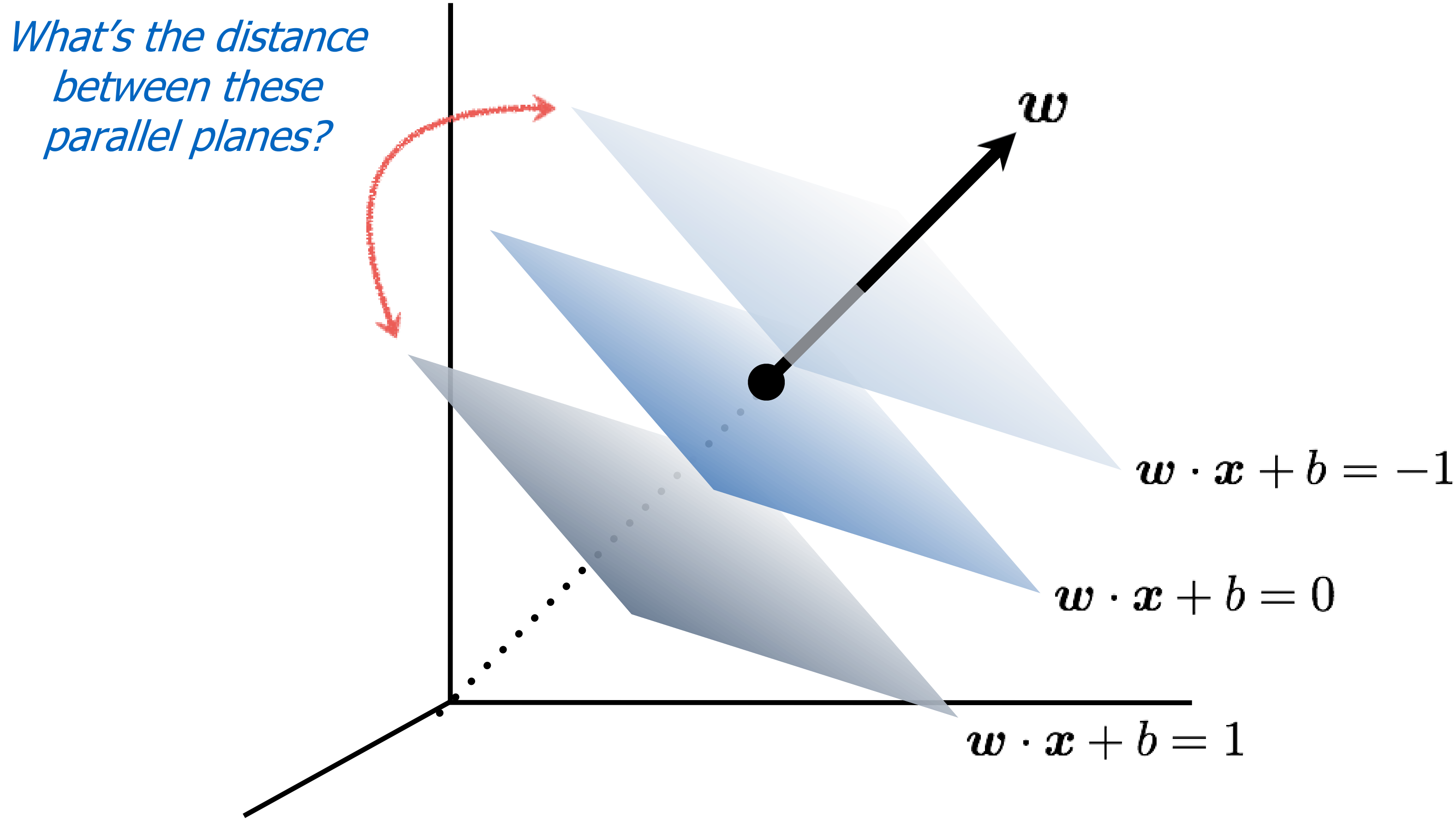
What are the dimensions of this vector?

*What happens if you change **b**?*

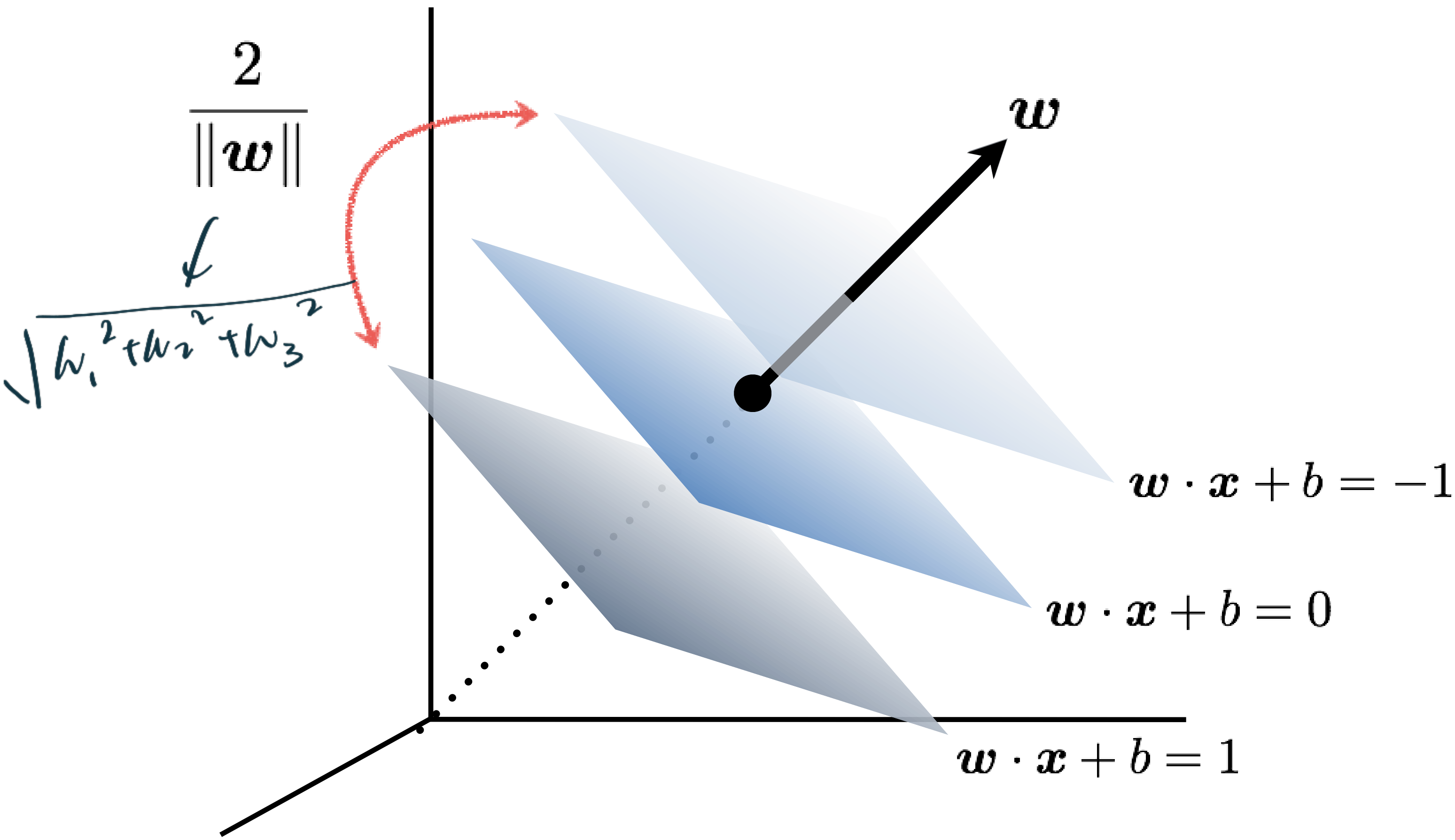
Hyperplanes (planes) in 3D



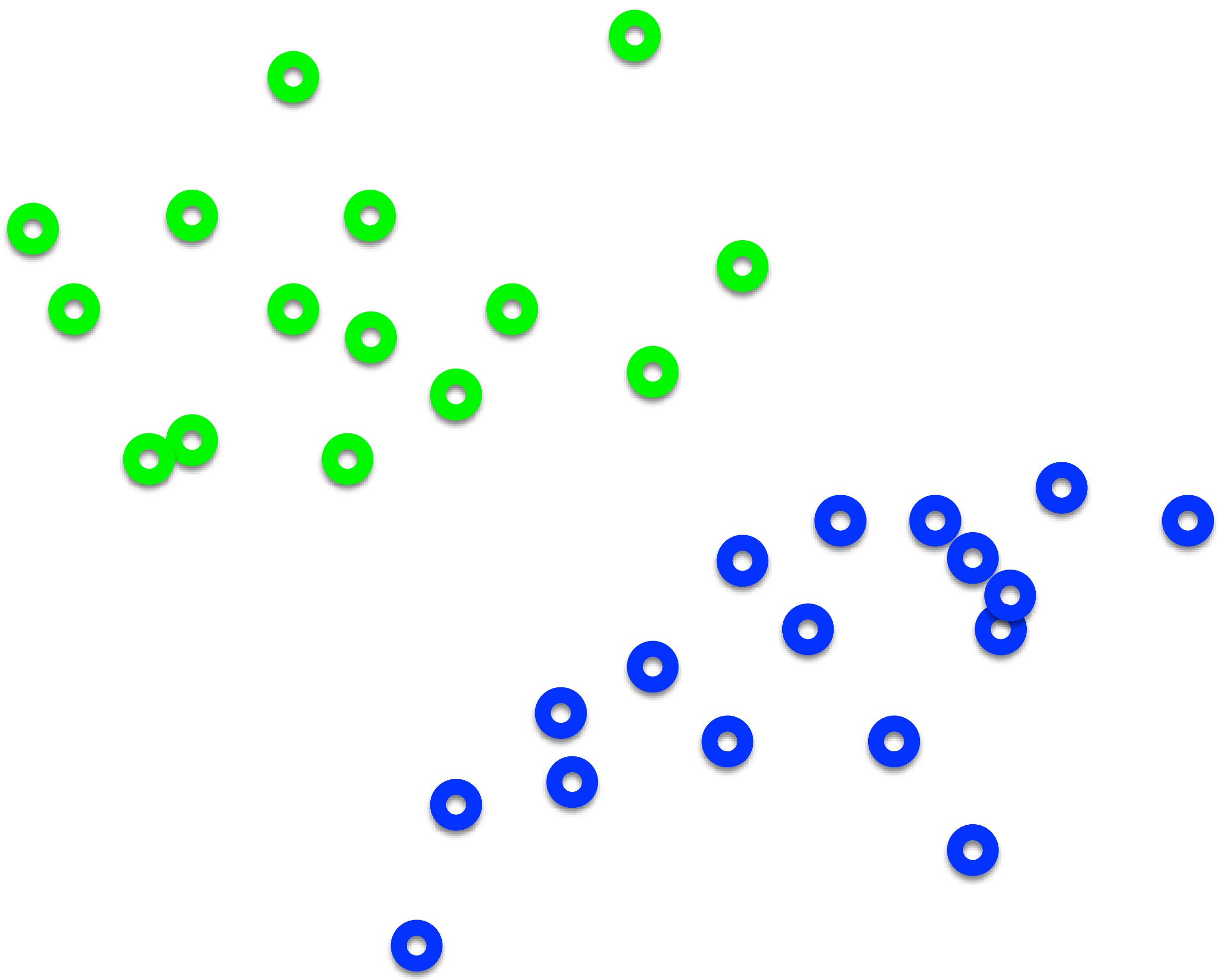
Hyperplanes (planes) in 3D



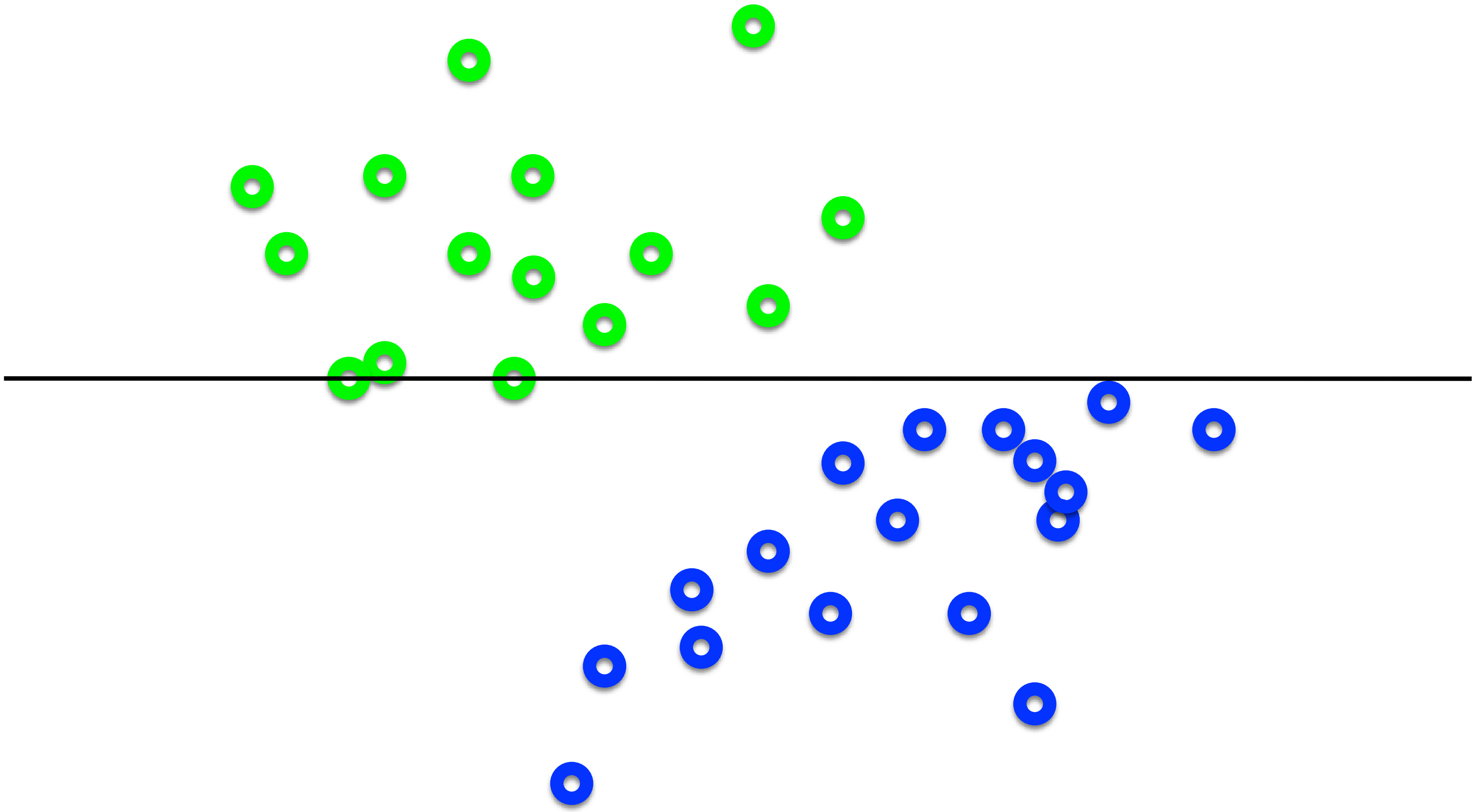
Hyperplanes (planes) in 3D



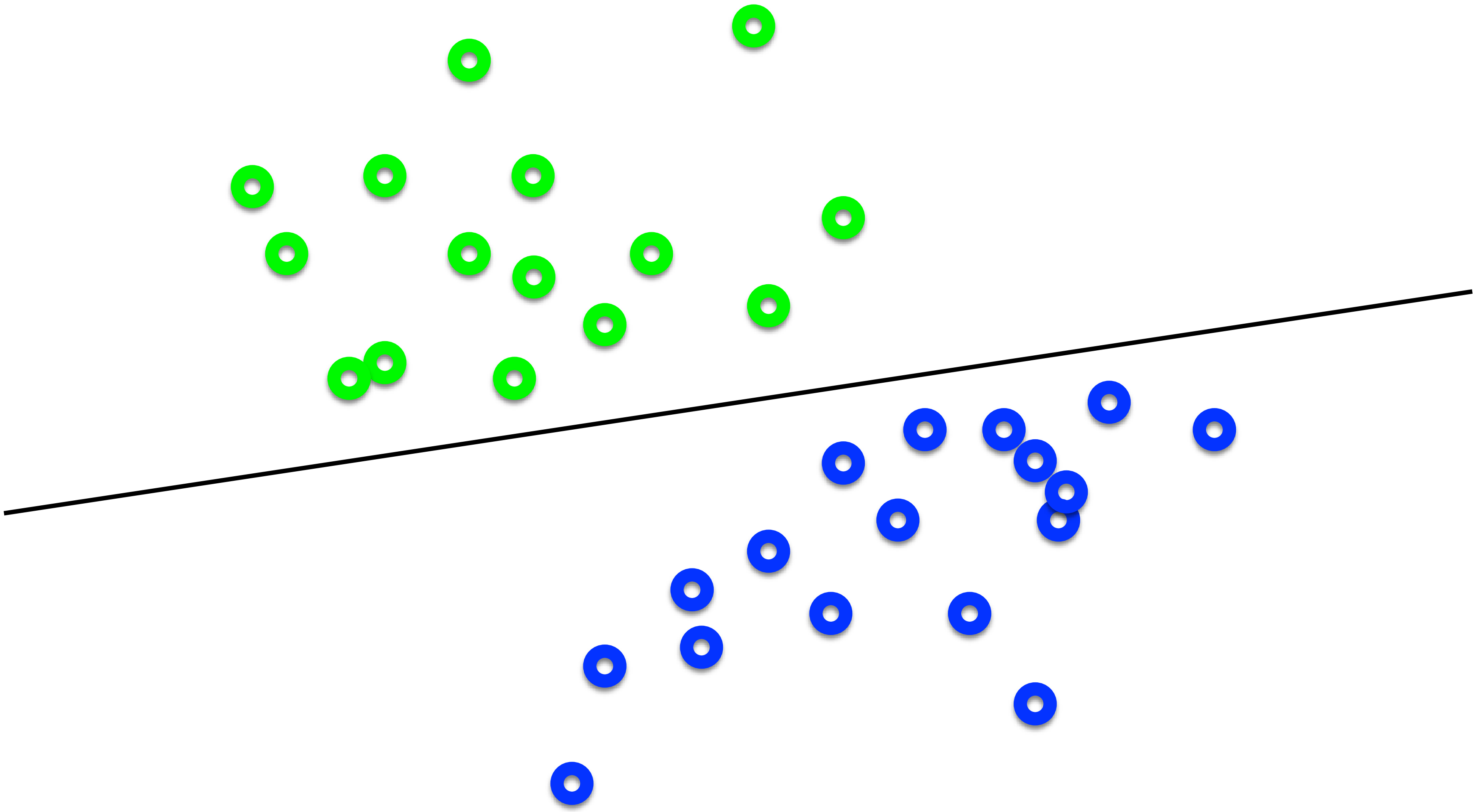
What’s the best **w**?



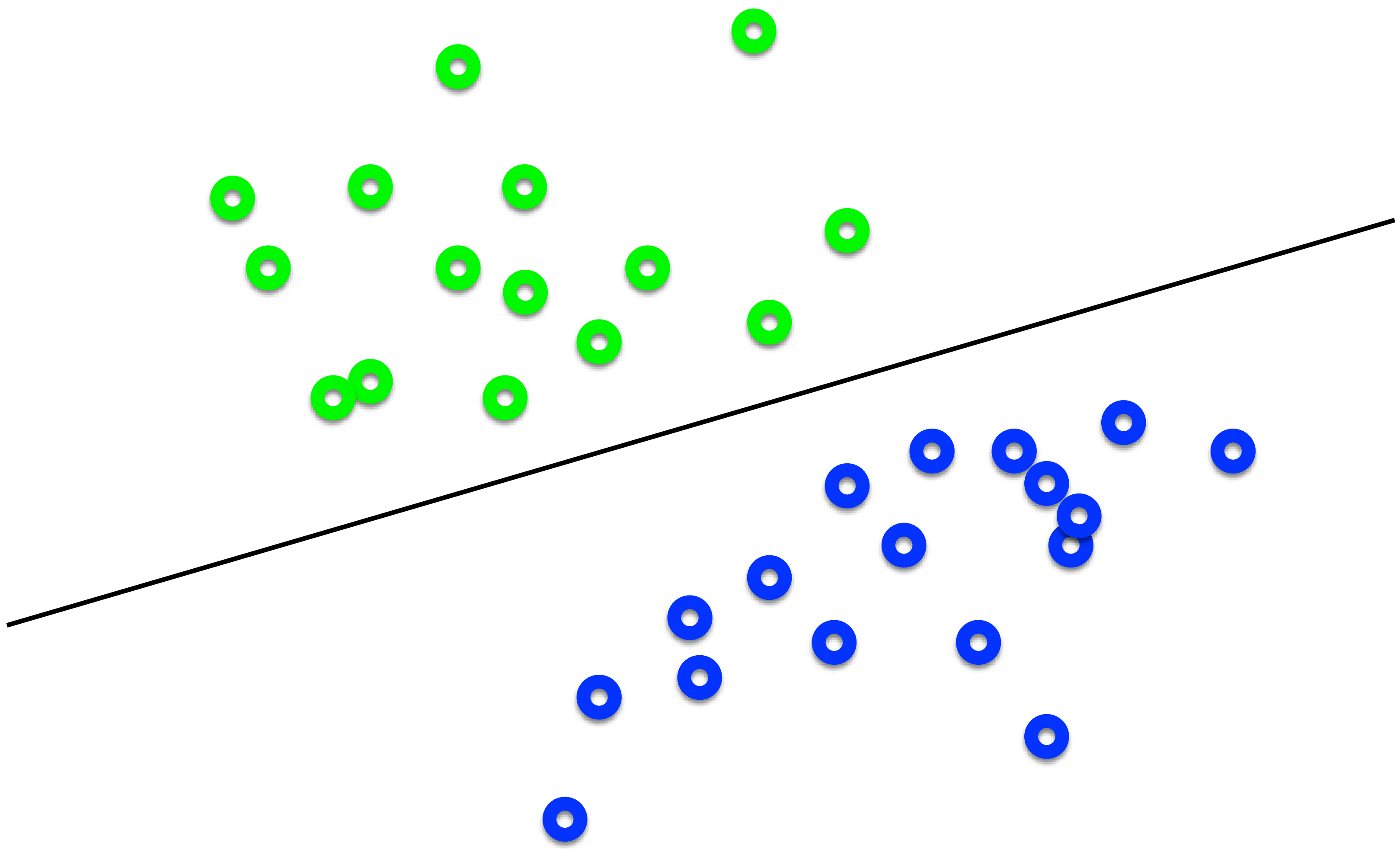
What’s the best **w**?



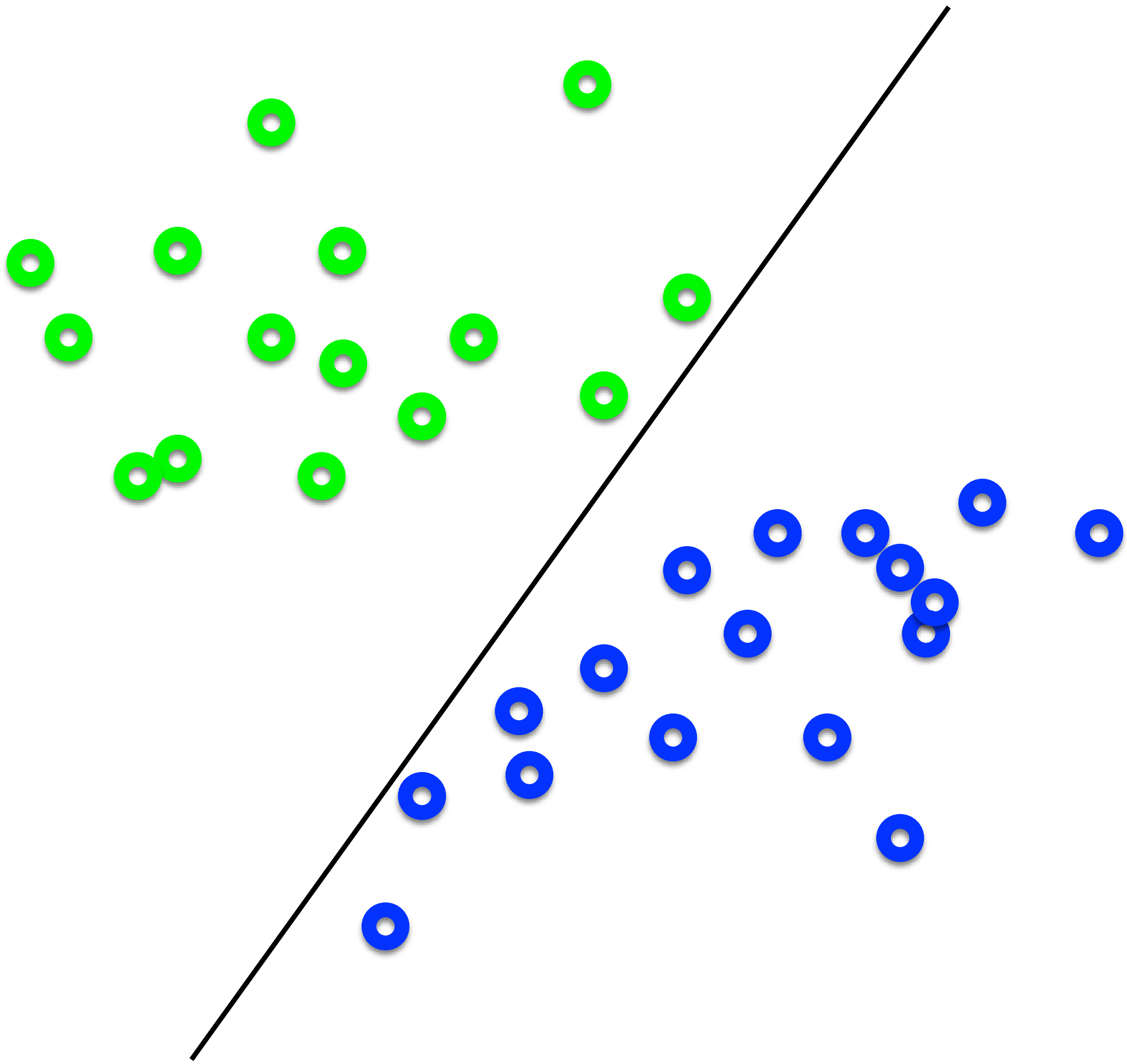
What's the best **w**?



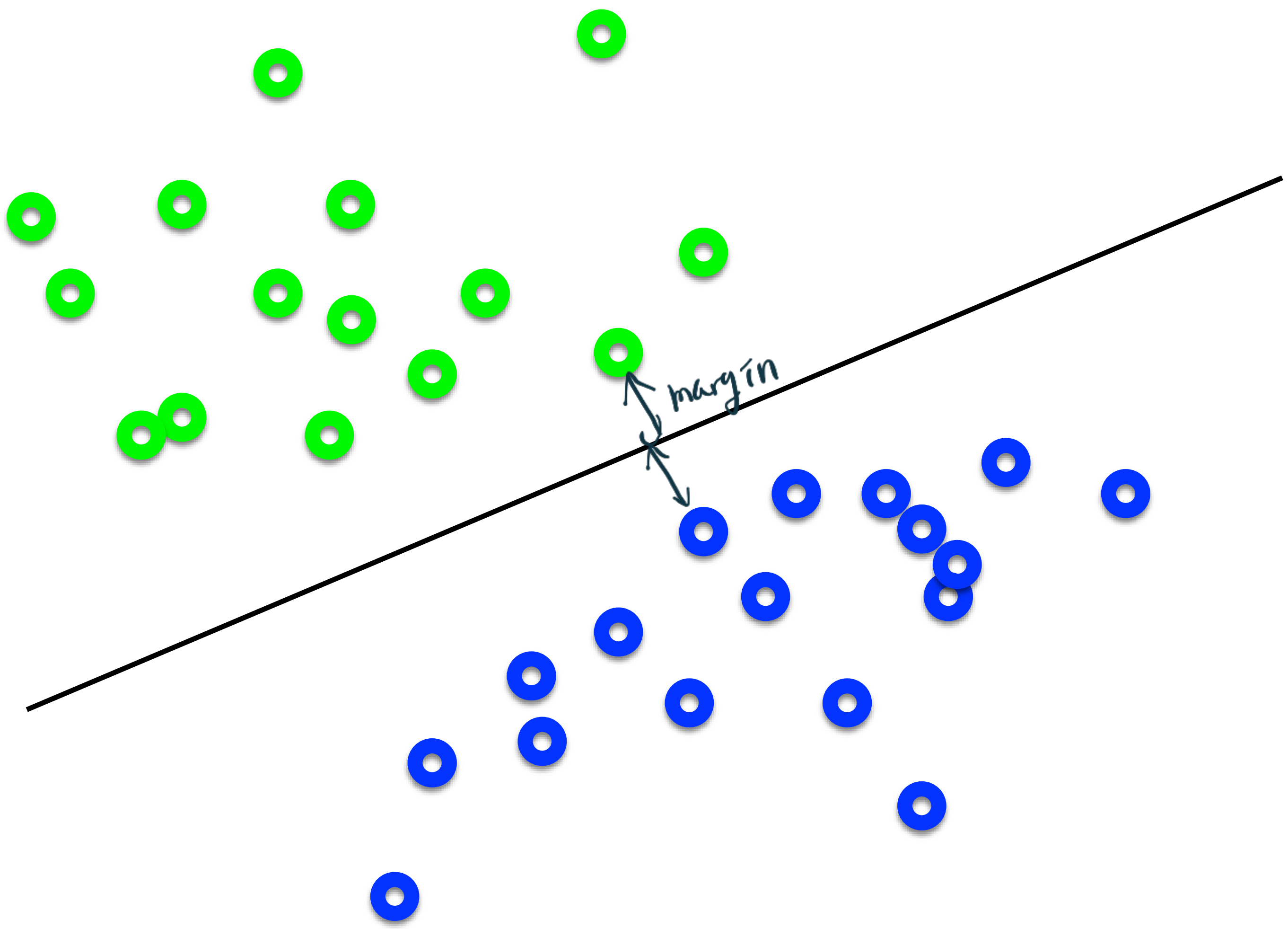
What’s the best **w**?



What's the best **w**?

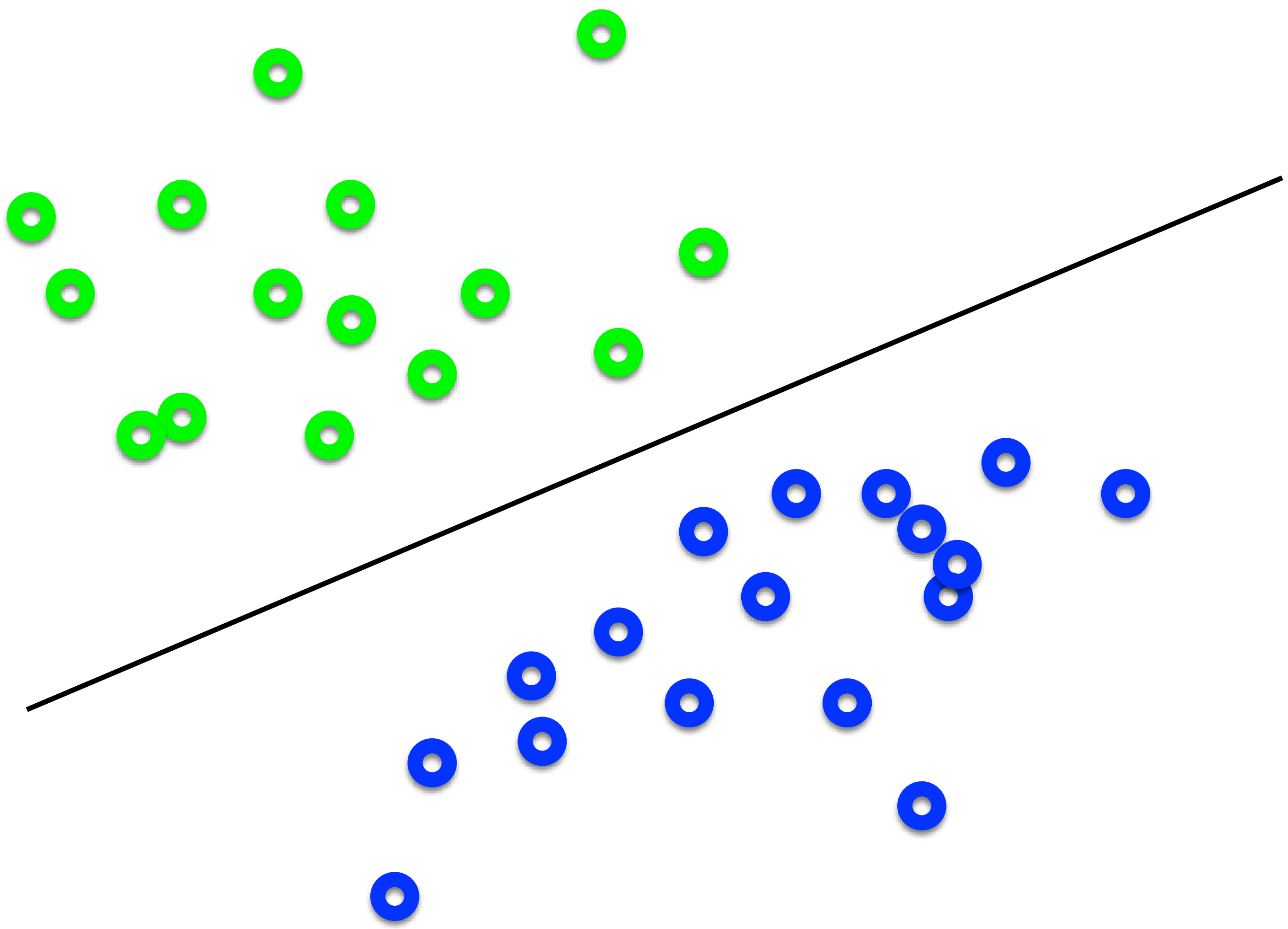


What's the best **w**?



Intuitively, the line that is the farthest from all interior points

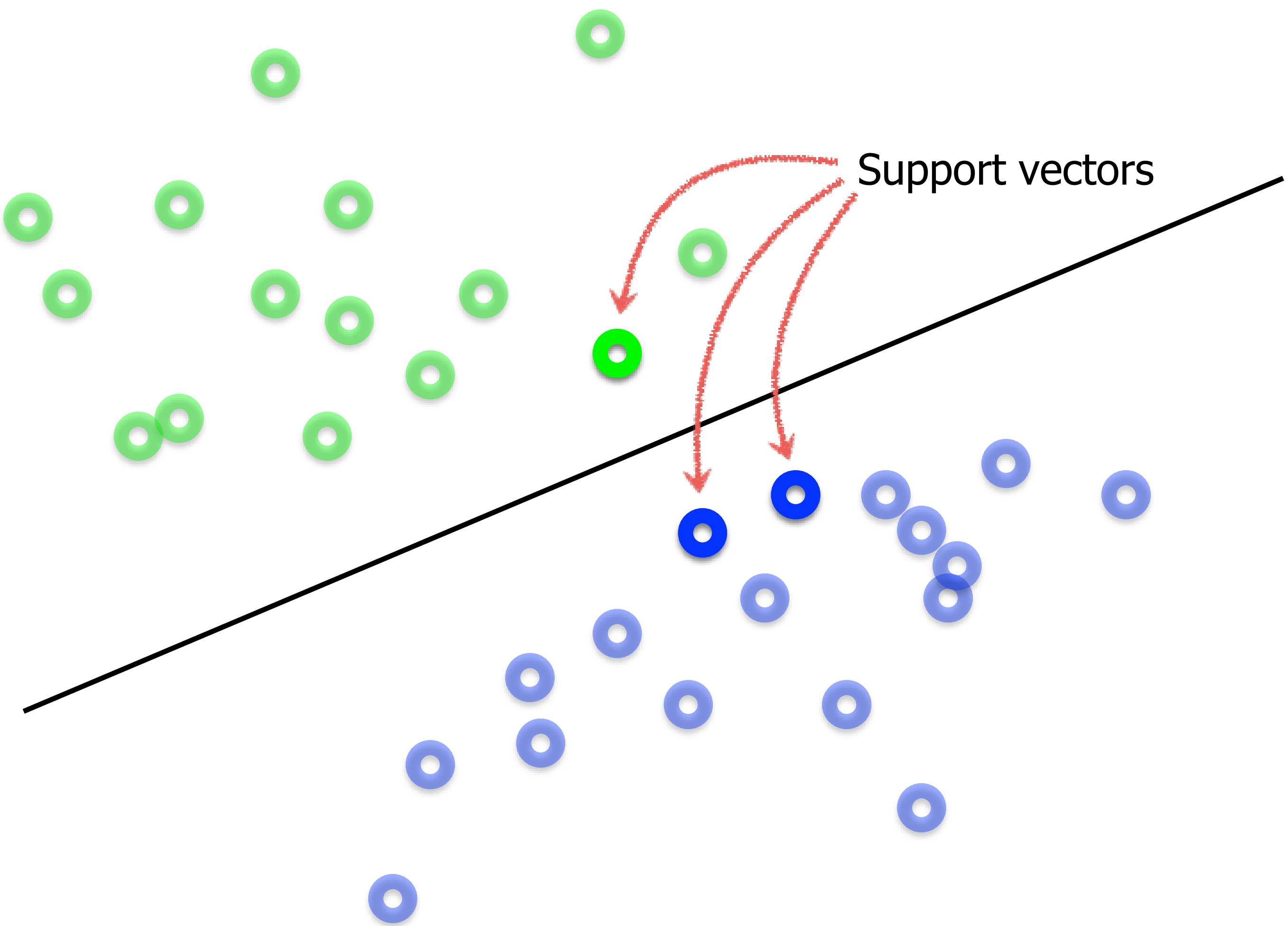
What's the best **w**?



Maximum Margin solution:

most stable to perturbations of data

What's the best **w**?



Want a hyperplane that is far away from 'inner points'

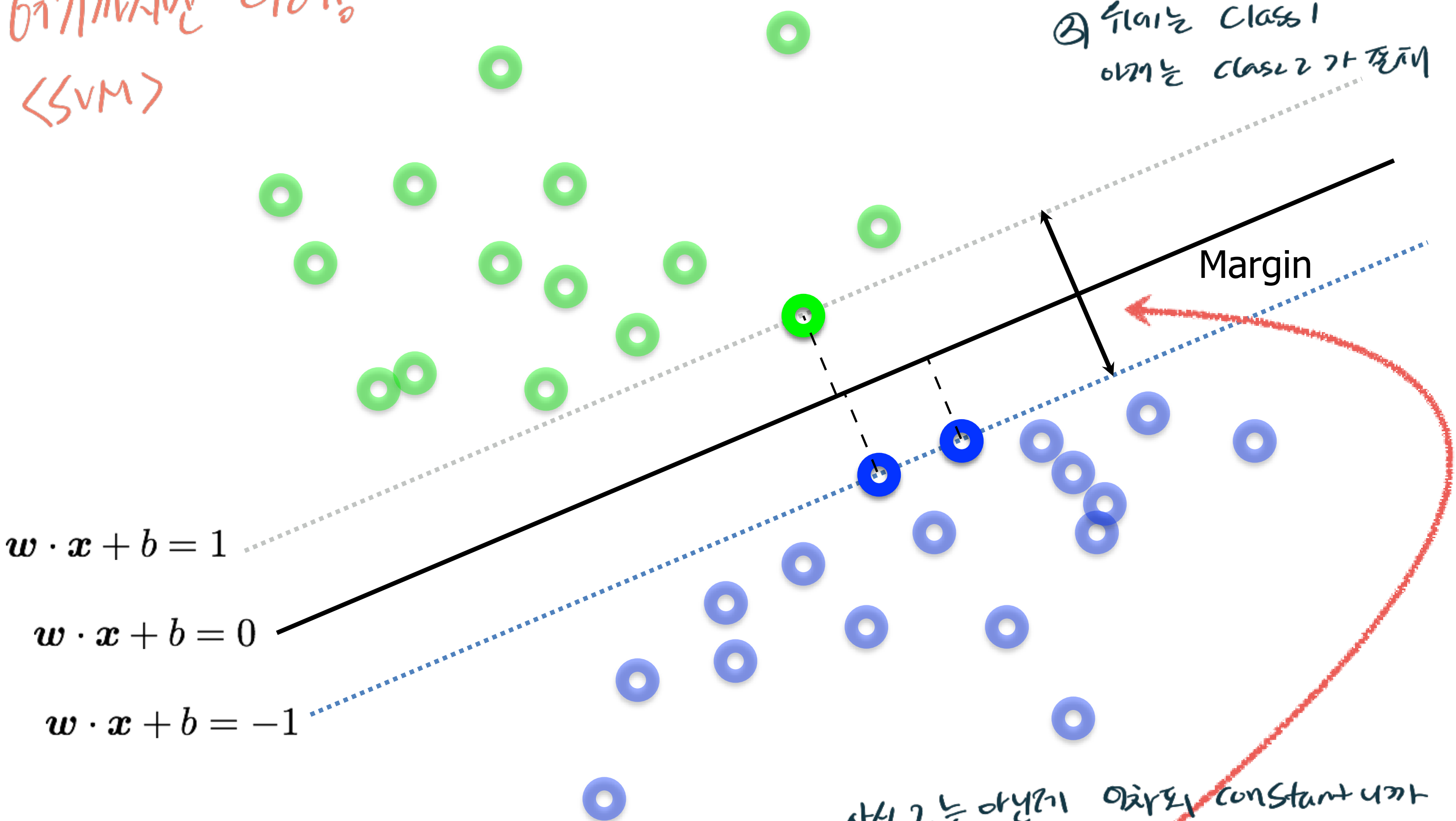
Find hyperplane **w** such that ...

Constraints : ① Support vector는 지나야 한다.

② 위에는 class 1 아래는 class 2가 존재

여기까지만 아해!

<SVM>



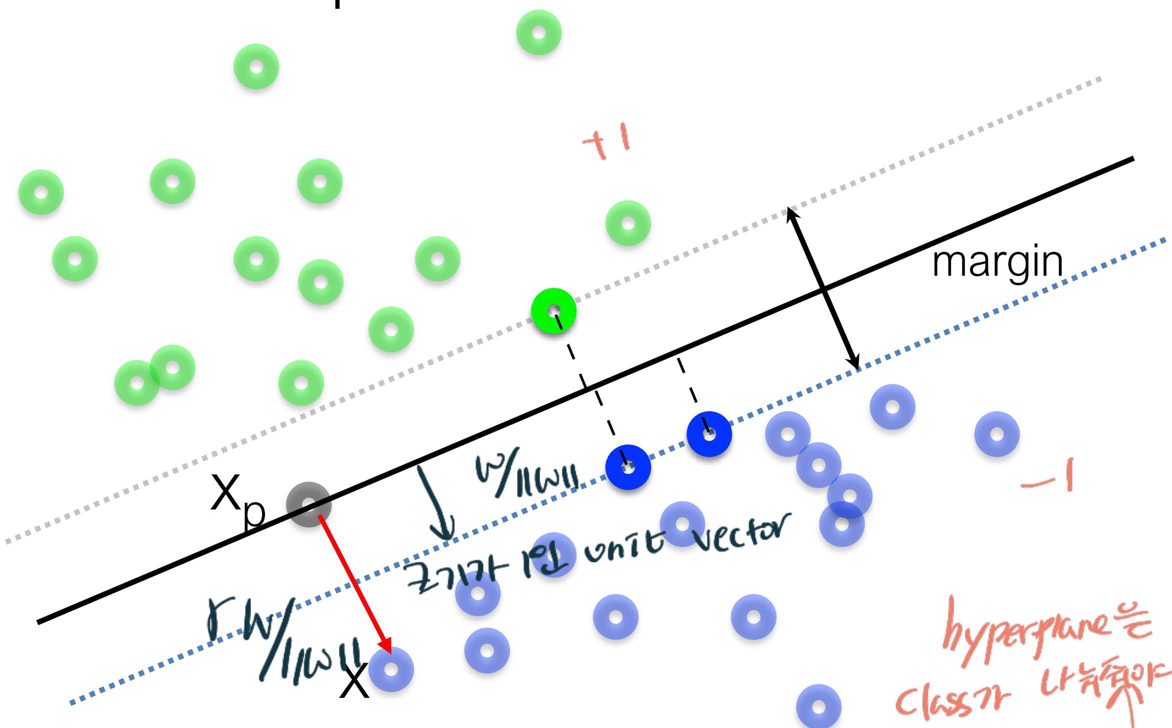
사실 2는 아닌데 어차피 constant니까 상관없음.

The gap between parallel hyperplanes $\dfrac{2}{\|\boldsymbol{w}\|}$ is maximized

$\|w\|$는 minimize

Another explanation

선 위에 X_p라는 점이 있다고 가정



hyperplane은 기준으로 class가 나눠줘야 함.

Let's say
$f(\mathbf{x})=\mathbf{w}\cdot\mathbf{x}+b$
우리가 찾는 hyper plane

A point x on the boundary has
$f(\mathbf{x})= \mathbf{w}\cdot\mathbf{x}+b=0$

(초평면)
A positive point x has
$f(\mathbf{x})= \mathbf{w}\cdot\mathbf{x}+b=a,\ a>0$

a는 constant라 1로 or 2로 바꿔도 됨.

Maximize margin
$r=a/||\mathbf{w}||$

$\max_{w,b} 2r= 2a/||\mathbf{w}||$
s.t. $(wx_j+b)y_j \geq a,\ \forall j$
label 1 or -1

a is an arbitrary number and can be normalized

$\min_{w,b} ||\mathbf{w}||$
s.t. $(wx_j+b)y_j \geq 1,\ \forall j$

We are going to measure the distance between an arbitrary point **x** and a point $\mathbf{x_p}$ on the boundary and on the perpendicular line from x to the boundary

unit vector

$x=x_p+r\cdot w/||w||,\ f(x_p)=0$

$f(x) = w\cdot x+b = w\cdot(x_p+ r\cdot w/||w||)+b = \cancel{wx_p+b} + r\cdot w\cdot w/||w|| = r||w||$

$\frac{w_1^2+w_2^2}{\sqrt{w_1^2+w_2^2}}$

The distance is $r = f(x)/||w||$

Can be formulated as a maximization problem

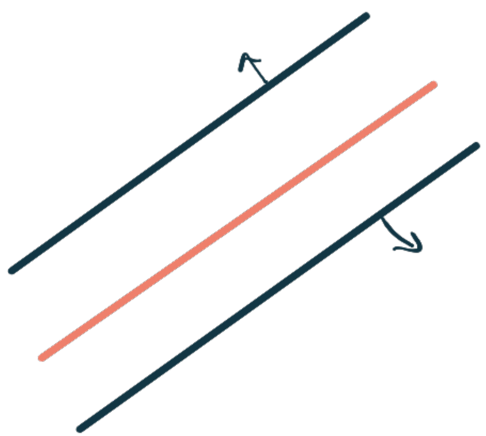
$$\max_{\boldsymbol{w}} \frac{2}{\|\boldsymbol{w}\|}$$

$$\text{subject to } \boldsymbol{w} \cdot \boldsymbol{x}_i + b \begin{array}{l} \geq +1 \quad \text{if} \quad y_i = +1 \\ \leq -1 \quad \text{if} \quad y_i = -1 \end{array} \text{ for } \quad i = 1, \ldots, N$$

What does this constraint mean?

Label of the data point

Why is it +1 and -1?



Can be formulated as a maximization problem

$$\max_{\boldsymbol{w}} \frac{2}{||\boldsymbol{w}||}$$

$$\text{subject to } \boldsymbol{w} \cdot \boldsymbol{x}_i + b \begin{array}{l} \geq +1 \quad \text{if} \quad y_i = +1 \\ \leq -1 \quad \text{if} \quad y_i = -1 \end{array} \text{ for } \quad i = 1, \ldots, N$$

Equivalently,

Where did the 2 go?

$$\min_{\boldsymbol{w}} ||\boldsymbol{w}||$$

$$\text{subject to } \quad y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \quad \text{for} \quad i = 1, \ldots, N$$

What happened to the labels?

‘Primal formulation’ of a linear SVM

$$\boxed{\min_{\boldsymbol{w}} \|\boldsymbol{w}\|}$$

Objective Function

$$\text{subject to } \boxed{y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1} \text{ for } \; i = 1, \ldots, N$$

Constraints

This is a convex quadratic programming (QP) problem

(a unique solution exists)

$$\min_{\boldsymbol{w}} \|\boldsymbol{w}\|$$

Objective Function

$$\text{subject to } \boxed{y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1} \text{ for } i = 1, \ldots, N$$

$$\boxed{y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 = 0 \quad \textit{for } \mathbf{x}_i \in \text{support vectors}}$$

Constraints

Lagrange Multiplier Method

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \right]$$

$$\textit{minimize w.r.t. } \mathbf{w} \textit{ and } b$$

$$\textit{maximize w.r.t. } \alpha_i \geq 0 \ \forall i$$

$$\alpha : \text{Lagrange multiplier}$$

- Finding Solution

$$\nabla_{\mathbf{w}} L = \mathbf{w} - \sum_i \alpha_i y_i x_i = 0 \qquad \qquad \mathbf{w} = \sum_i \alpha_i y_i x_i \quad ①$$

$$\nabla_b L = -\sum_i \alpha_i y_i = 0 \qquad \longrightarrow \qquad -\sum_i \alpha_i y_i = 0 \quad ②$$

If we know only the value of α, we can find w

Organize formulas into functions for α

$$\frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 \right]$$

$$= \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \left[y_i \left(\mathbf{w} \cdot \mathbf{x}_i \right) \right] = \sum_{i=1}^{N} \alpha_i + \frac{1}{2}\mathbf{w}^T \mathbf{w} - \mathbf{w}^T \mathbf{w}$$

$$= \sum_{i=1}^{N} \alpha_i - \frac{1}{2}\mathbf{w}^T \mathbf{w} = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j = L(\alpha)$$

Since the coefficient of the α is negative, the problem of finding the minimum value is replaced with the problem of finding the maximum value.

Find α that maximizes *L*
→ Dual Form

- Finding Solution

$$\max L(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j =$$

From ②

$$\sum_i \alpha_i y_i = 0,\ \alpha_i \geq 0,\ i = 1, \dots, n$$

From KKT condition (KKT complimentary slackness)

$$\alpha_i = 0 \quad or \quad y_i \left(w^T x_i + b \right) - 1 = 0$$

$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1=0$ → $\alpha_i \neq 0$, x_i is support vector
otherwise, xi is not support vector → it cannot influence to find the **w**

Concept of Adaboost

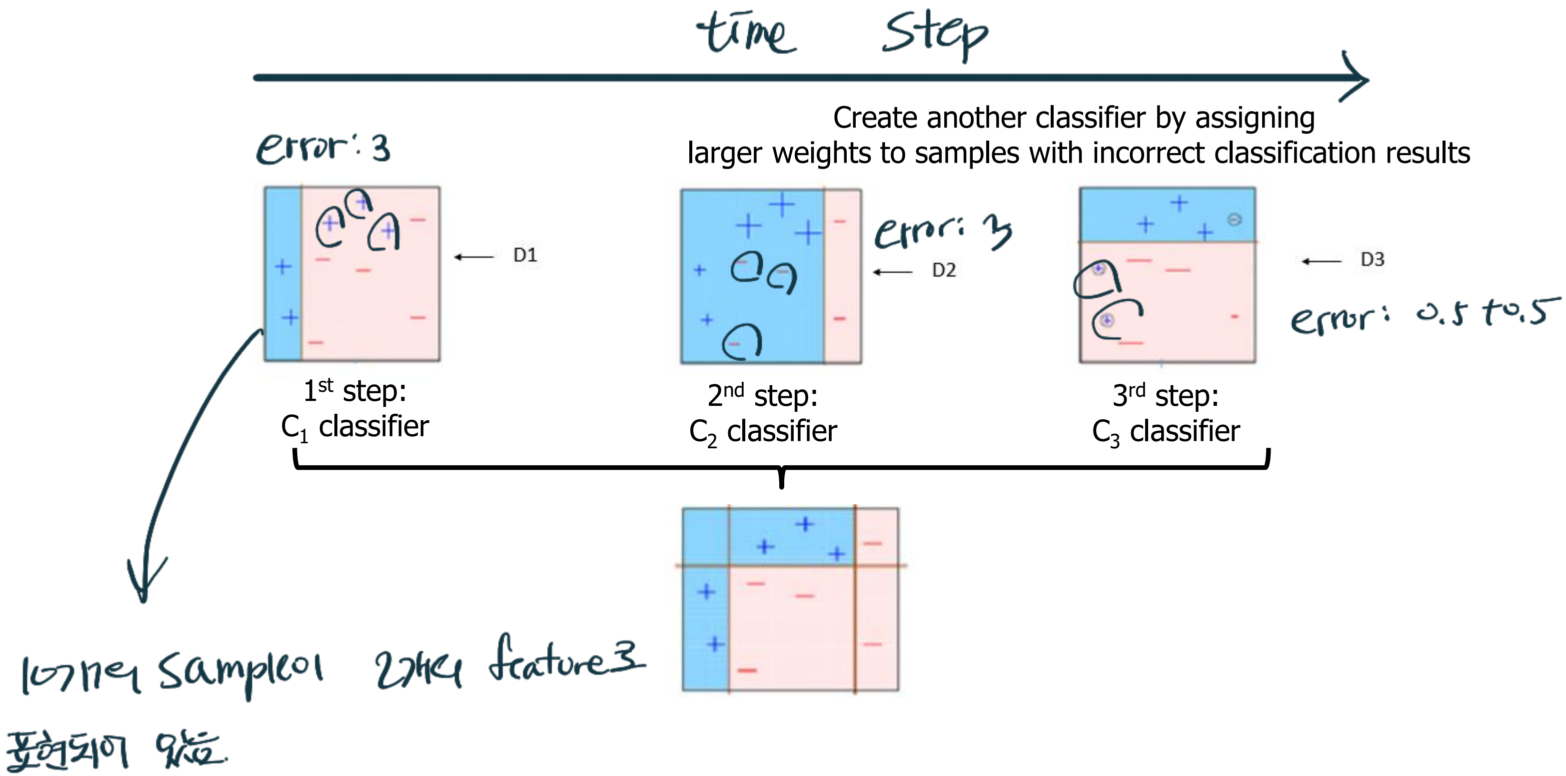
Classifier ensemble

- The importance of collaboration (ensemble)
  - Collection of individual abilities
  - Better results if you cooperate → Motivation of classifier ensemble

다양한 관점 (집단지성)

Adaboost

- Boosting
  - Most popular boosting: Adaboost



Create final strong classifier by using various weak classifiers

Adaboost

- Boosting
  - Most popular boosting: Adaboost

  - Using effective resampling
    - The boosting algorithm is created so that the kth classifier c_k and the next c_{k+1} are related

  - Samples in training set X are separated by correct or incorrect classification results by c_k
    - The correct sample lowers the weight, and the wrong sample is still a 'tricky' opponent, increasing the weight
    - c_{k+1} is learned considering the above weight

Adaboost

앙상블 기법은 $C_k(x_j)$를 하는 과정은 고려하지 않음.

vector : 1x feature 수

Label : scalar (0 or 1)

N=3
(0,0) → 0 $w_1=\frac{1}{3}$
(1,0) → 0 $w_2=\frac{1}{3}$
(1,1) → 1 $w_3=\frac{1}{3}$
k=2

C_1은 1st 원소만 가지고 0.5로 구분 $w_1=\frac{1}{4}$ $w_2=\frac{1}{2}$ $w_3=\frac{1}{4}$

C_2는 2nd 0.5

Input: Training sample, $X=\{(\mathbf{x}_1,t_1),(\mathbf{x}_2,t_2),\cdots,(\mathbf{x}_N,t_N)\}$, K is the number of classifier

Output: Classifier ensemble, $C=\{(c_k,\alpha_k),\ 1\le k\le K\}$ // α_k is the Confidence of classifier, c_k

K가 얼마나 많은만 한지

Classifier 1의 경우 $\varepsilon=0.3$

```
C=∅;
for(j=1 to N) w_j=1/N;  // Use the same weights initially
for(k=1 to K) {
  Learning classifier c_k considering w_1, w_2,..,w_N   // More weighted samples are more important
  ε=0;
  for(j=1 to N) if(c_k(x_j)≠t_j) ε=ε+w_j;  // Calculate error (sum of weights of wrong samples)
  if(ε<0.5) {  // Classifier c_k is taken only when the error is less than 0.5
    α_k = 1/2 log((1-ε)/ε);  // Confidence of classifier c_k
    for(j=1 to N)
      if(c_k(x_j)≠t_j) w_j=w_j×e^α;  // Increase the weight of wrong samples
      else w_j=w_j×e^-α;  // decrease the weight of correct samples
    Normalize the sum of the w_1, w_2,..,w_N to be 1.
    C=C∪(c_k,α_k);
  }
  else {c_k=Nil; C=C∪(c_k,0);}
}
```

$\alpha_k=\frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$

weight가 크면, 틀렸을때 error가 커짐. 더 중요하게 보겠다.

prediction과 실제 label이랑 다르면,

0.5보다 크면, 버린다.

error가 작으면 커짐.

← Classifier 추가

normalize 안 하면 scale이 달라짐

Adaboost

Input: Training sample, $X=\{(\mathbf{x}_1, t_1), (\mathbf{x}_2, t_2), \cdots, (\mathbf{x}_N, t_N)\}$, K is the number of classifier, t: ground truth

Output: Classifier ensemble, $C=\{(c_k, \alpha_k), 1\le k\le K\}$ // α_k is the Confidence of classifier, c_k

$C=\emptyset$;
for(j=1 to N) $w_j=1/N$;
for(k=1 to K) {
  Learning classifier c_k considering $w_1, w_2,..,w_N$
  $\varepsilon=0$;
  for(j=1 to N) if($c_k(\mathbf{x}_j)\neq t_j$) $\varepsilon=\varepsilon+w_j$;
  if($\varepsilon<0.5$) {
    $\alpha_k=\frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$;
    for(j=1 to N)
      if($c_k(\mathbf{x}_j)\neq t_j$) $w_j=w_j\times e^{\alpha}$;
      else $w_j=w_j\times e^{-\alpha}$;
    Normalize the sum of the $w_1, w_2,..,w_N$ to be 1.
    $C=C\cup(c_k, \alpha_k)$;
  }
  else {c_k=Nil; $C=C\cup(c_k, 0)$;}
}

Adaboost

Input: Training sample, $X=\{(\mathbf{x}_1,t_1),(\mathbf{x}_2,t_2),\cdots,(\mathbf{x}_N,t_N)\}$, K is the number of classifier, t: ground truth

Output: Classifier ensemble, $C=\{(c_k,\alpha_k), 1\le k\le K\}$ // α_k is the Confidence of classifier, c_k

$C=\emptyset$;

for(j=1 to N) $w_j=1/N$; // Use the same weights initially

for(k=1 to K) {

  Learning classifier c_k considering $w_1, w_2, .., w_N$ // More weighted samples are more important

  $\varepsilon=0$;

  for(j=1 to N) if($c_k(\mathbf{x}_j)\ne t_j$) $\varepsilon=\varepsilon+w_j$; // Calculate error (sum of weights of wrong samples)

  if($\varepsilon<0.5$) { // Classifier c_k is taken only when the error is less than 0.5

    $\alpha_k=\frac{1}{2}\log\left(\frac{1-\varepsilon}{\varepsilon}\right)$; // Confidence of classifier c_k

    for(j=1 to N)

      if($c_k(\mathbf{x}_j)\ne t_j$) $w_j=w_j\times e^{\alpha}$; // Increase the weight of wrong samples

      else $w_j=w_j\times e^{-\alpha}$; // decrease the weight of correct samples

    Normalize the sum of the $w_1, w_2, .., w_N$ to be 1.

    $C=C\cup(c_k,\alpha_k)$;

  }

  else {c_k=Nil; $C=C\cup(c_k,0)$;}

}