

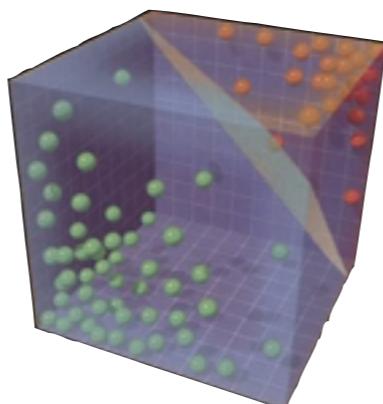


# Classification

16-385 Computer Vision (Kris Kitani)  
**Carnegie Mellon University**

# typical perception pipeline

**representation**

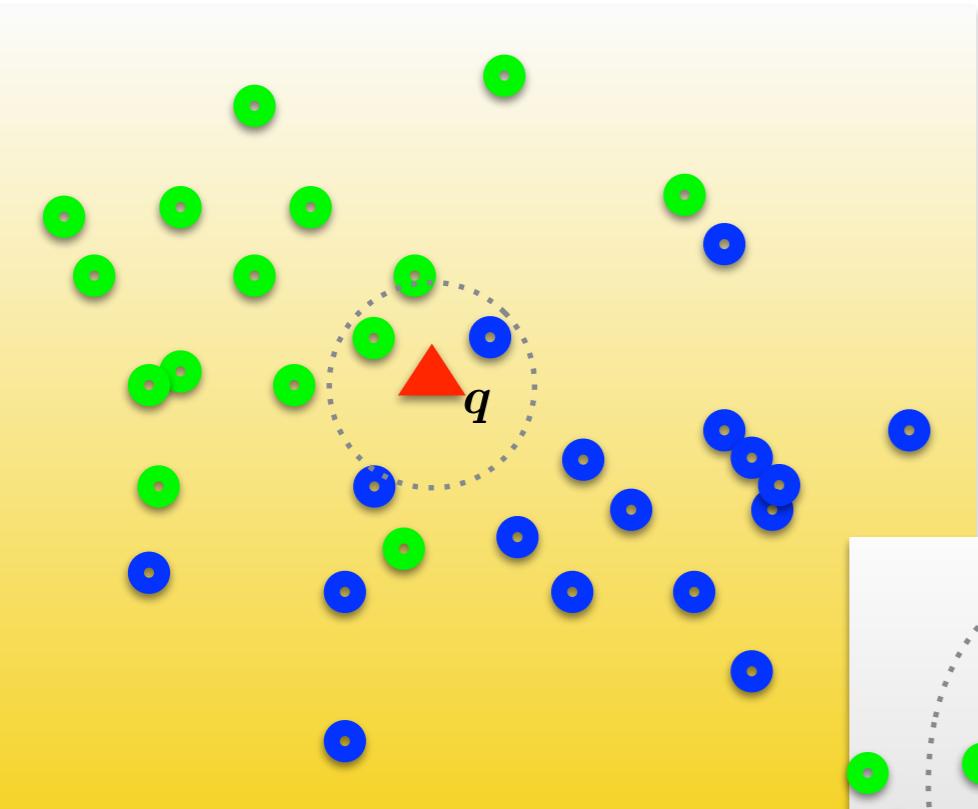


**classifier**

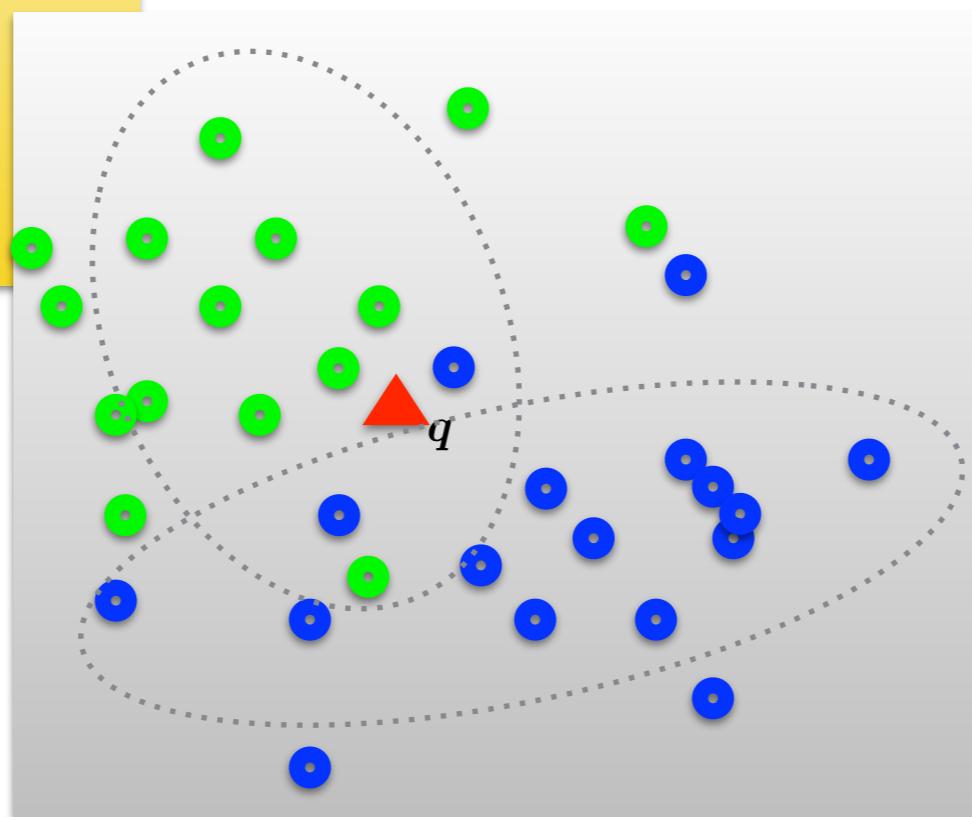


**output**

- Nearest Neighbor classifier
- Naive Bayes classifier
- Support Vector Machine

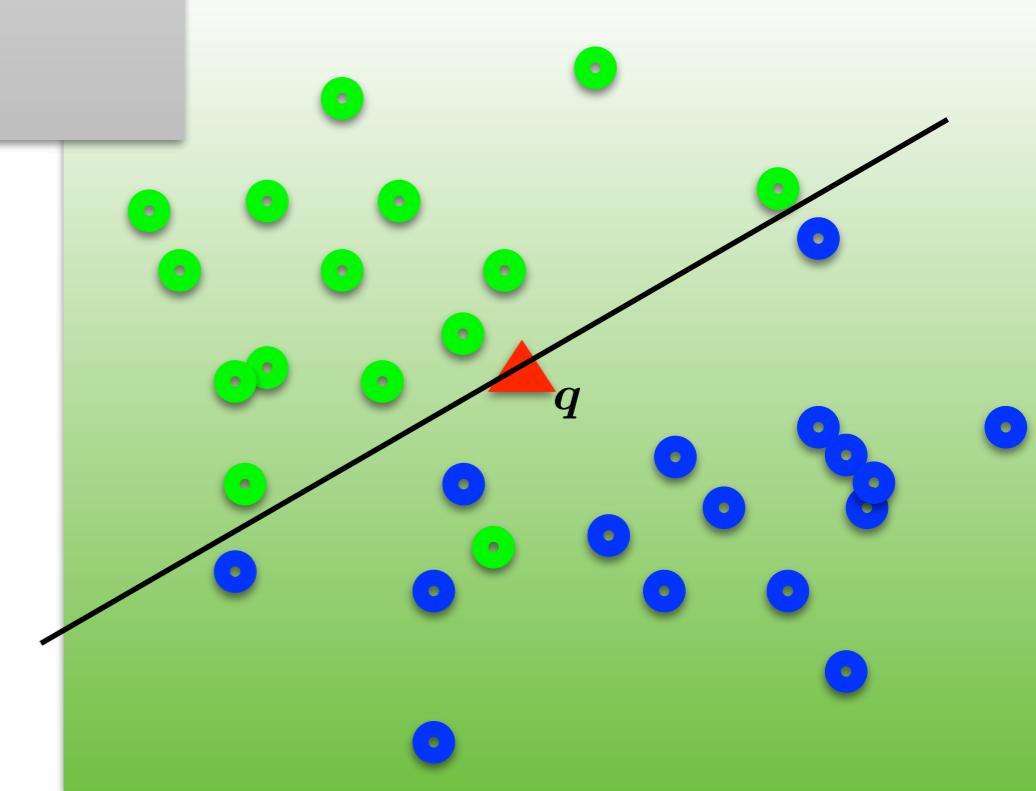


Nearest Neighbor

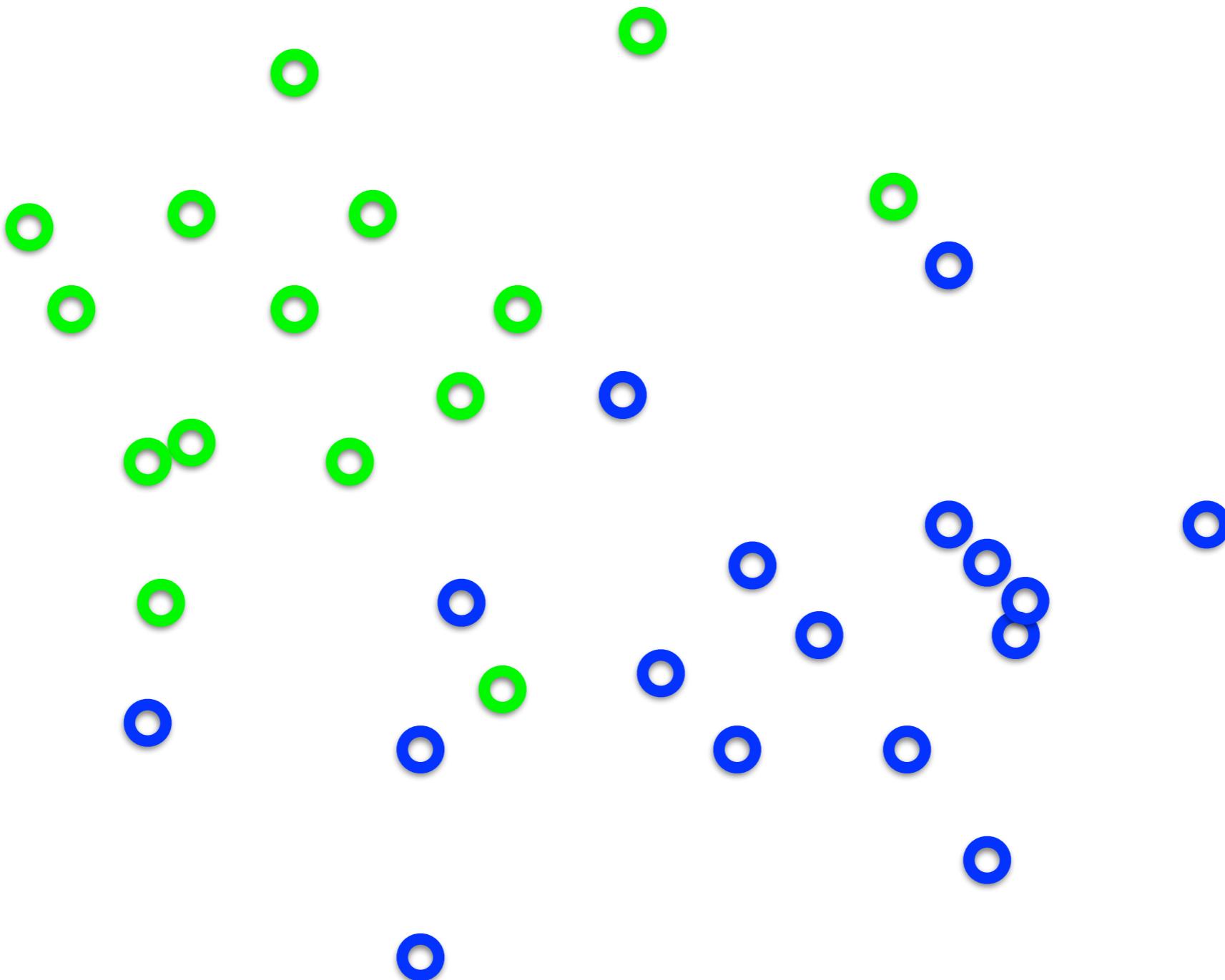


Naive Bayes

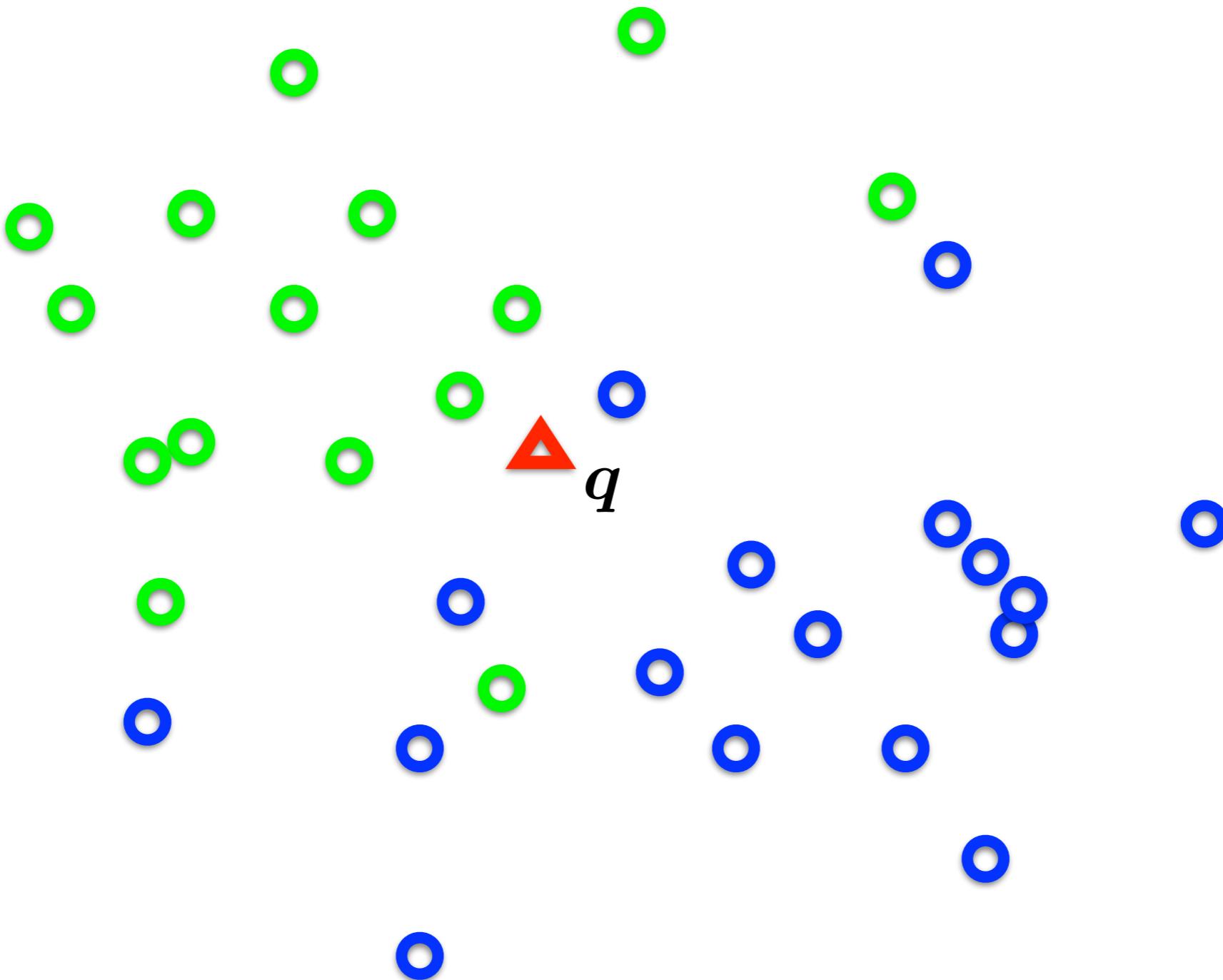
Support Vector Machine



# Distribution of data from two classes

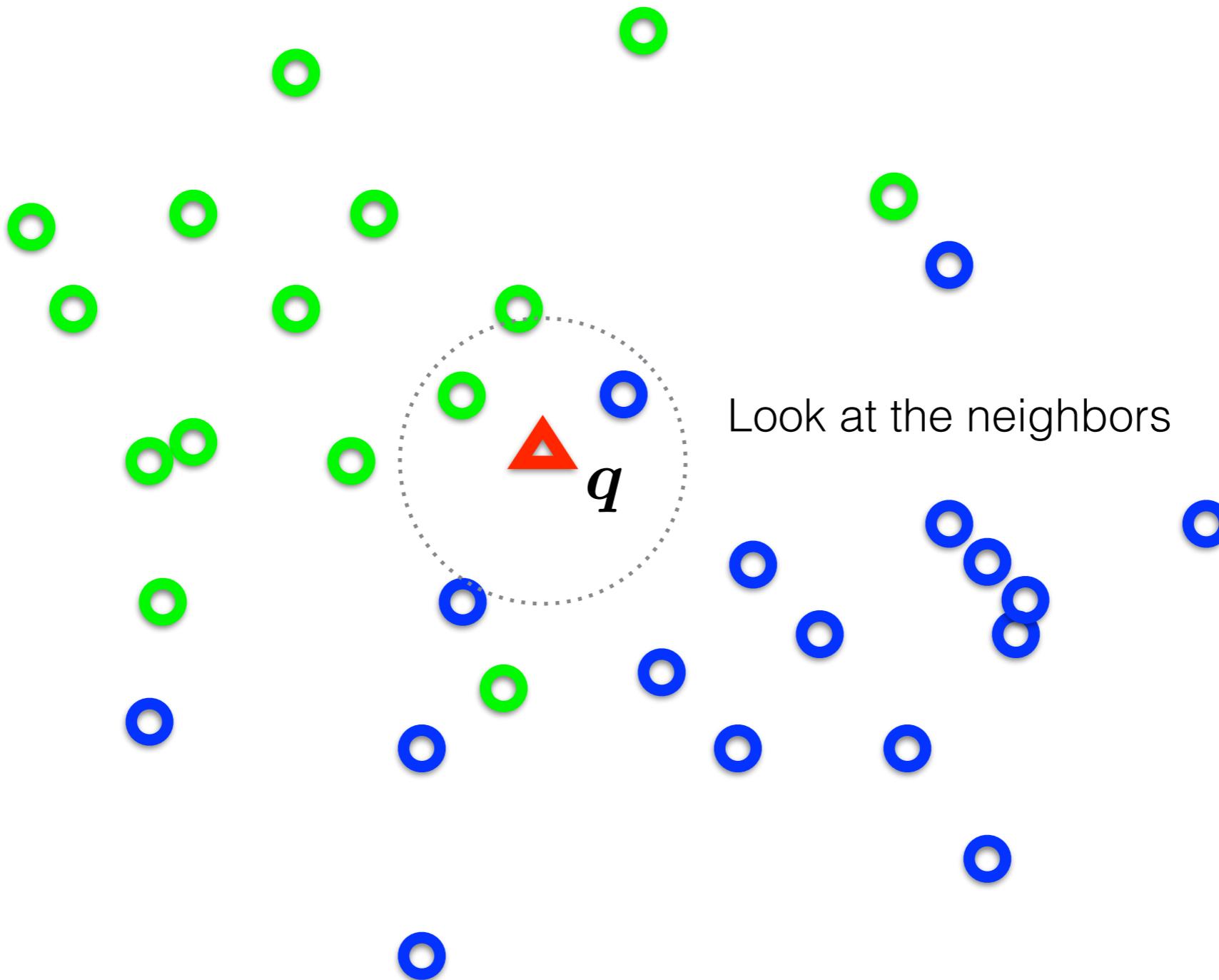


# Distribution of data from two classes



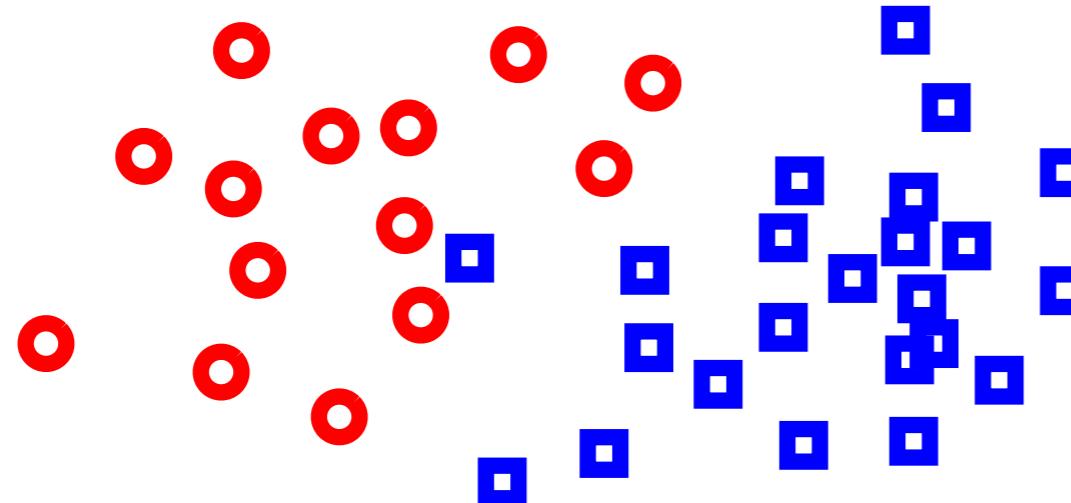
*Which class does  $q$  belong to?*

# Distribution of data from two classes



# K-nearest neighbor

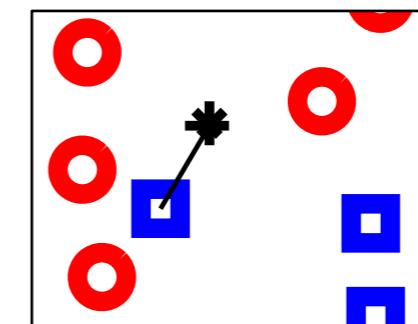
# K-Nearest Neighbor (KNN) Classifier



Non-parametric pattern classification approach  
Consider a two class problem where each sample consists of two measurements  $(x,y)$ .

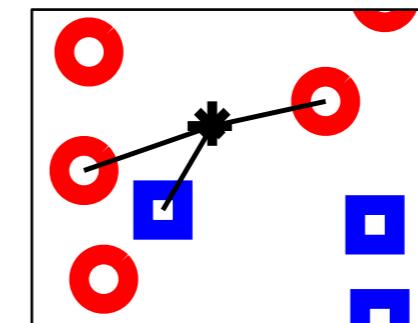
For a given query point  $q$ , assign the class of the nearest neighbor

$k = 1$



Compute the  $k$  nearest neighbors and assign the class by majority vote.

$k = 3$



# Nearest Neighbor is competitive

4028150880327726495529284686500876/71127400776386420140578214711366  
50711167679664143112241087634006330117113109975414895351982739901029  
8468682467933943144705960444612336459685600641865284554770782237018  
76953465018828357808571101378507110114527623028596972136418240510226  
93977149069842728100783331376131605747595849916501320348220251514889  
82049962335648092836457294912860704116759914592504108408989425198980  
35517216919955162286714604033223689853854520563283995794671313660901  
94368160413174951001162198403649071657525185470670258104571851900607  
8857389886823975629288168879180172075190209862393802111142972512199  
148534347507488153959369036398212868553941251514414435912233029009  
9319097549201051493361525220266012030255795808950325908846884546549  
69285457999216340783934656219260061287992047750564674307507426899404  
12845278113035703193631773084826529739099642972116747590821445161325  
90662367728608302983253980019513960141712379749939282718091017796999  
2101045282835177112976405078847858498138031715516574935471208160734  
283087884084458566309376893495891288681379011470817457121130621280766  
41992780136134111560707232522949810/61278000822722799275134941756283

## MNIST Digit Recognition

- Handwritten digits
- 28x28 pixel images:  $d = 784$
- 60,000 training samples
- 10,000 test samples

Yann LeCunn

	Test Error Rate (%)
Linear classifier (1-layer NN)	12.0
K-nearest-neighbors, Euclidean	5.0
K-nearest-neighbors, Euclidean, deskewed	2.4
K-NN, Tangent Distance, 16x16	1.1
K-NN, shape context matching	0.67
1000 RBF + linear classifier	3.6
SVM deg 4 polynomial	1.1
2-layer NN, 300 hidden units	4.7
2-layer NN, 300 HU, [deskewing]	1.6
LeNet-5, [distortions]	0.8
Boosted LeNet-4, [distortions]	0.7

## Pros

- simple yet effective

## Cons

- search is expensive (can be sped-up)
- storage requirements
- difficulties with high-dimensional data

# **What is the best distance metric between data points?**

- Typically Euclidean distance
- Locality sensitive distance metrics
- Important to normalize.  
Dimensions have different scales

## **How many K?**

- Typically  $k=1$  is good
- Cross-validation

# Distance metrics

$$D(\mathbf{x}, \mathbf{y}) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_N - y_N)^2}$$

Euclidean

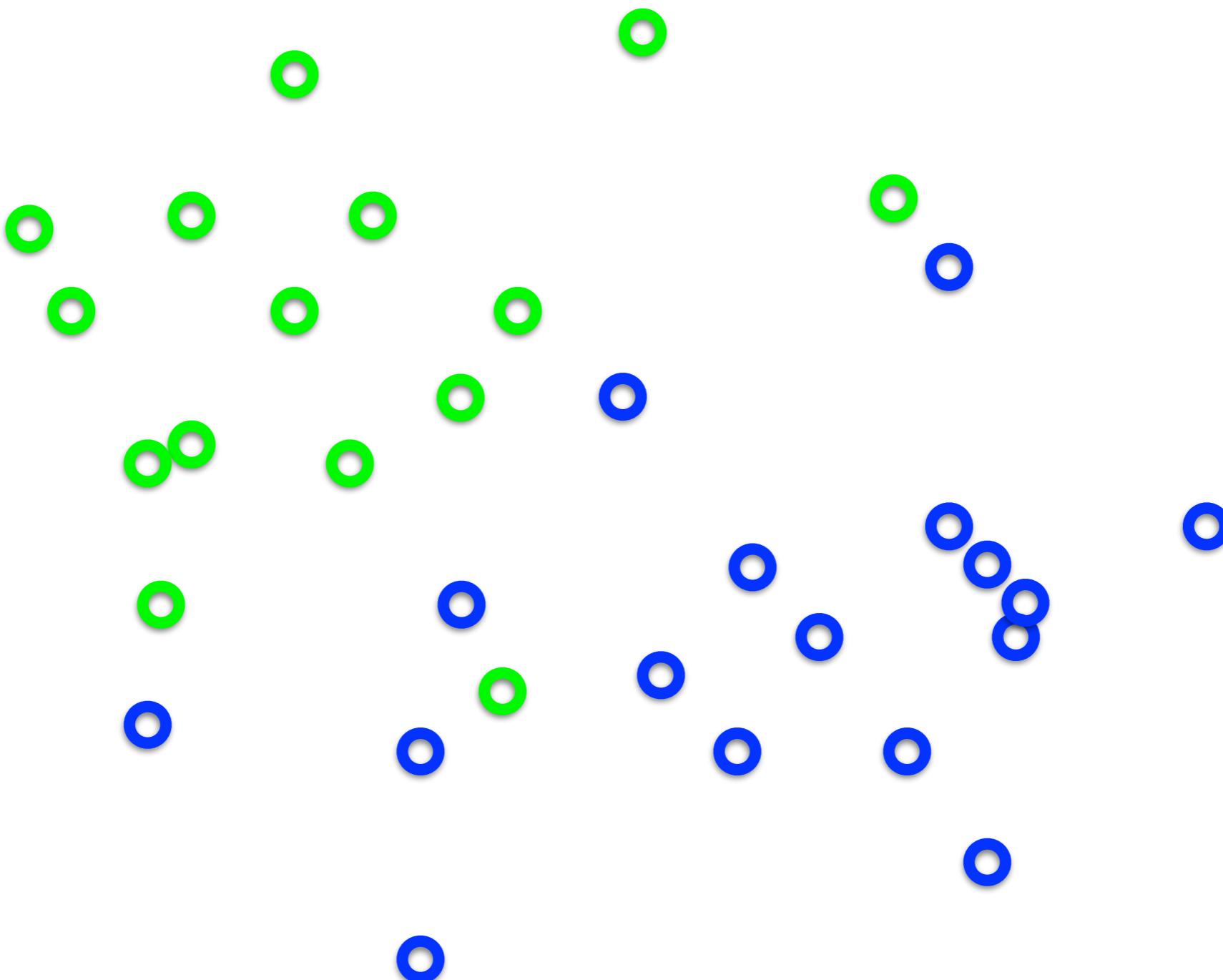
$$D(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\| \|\mathbf{y}\|} = \frac{x_1 y_1 + \cdots + x_N y_N}{\sqrt{\sum_n x_n^2} \sqrt{\sum_n y_n^2}}$$

Cosine

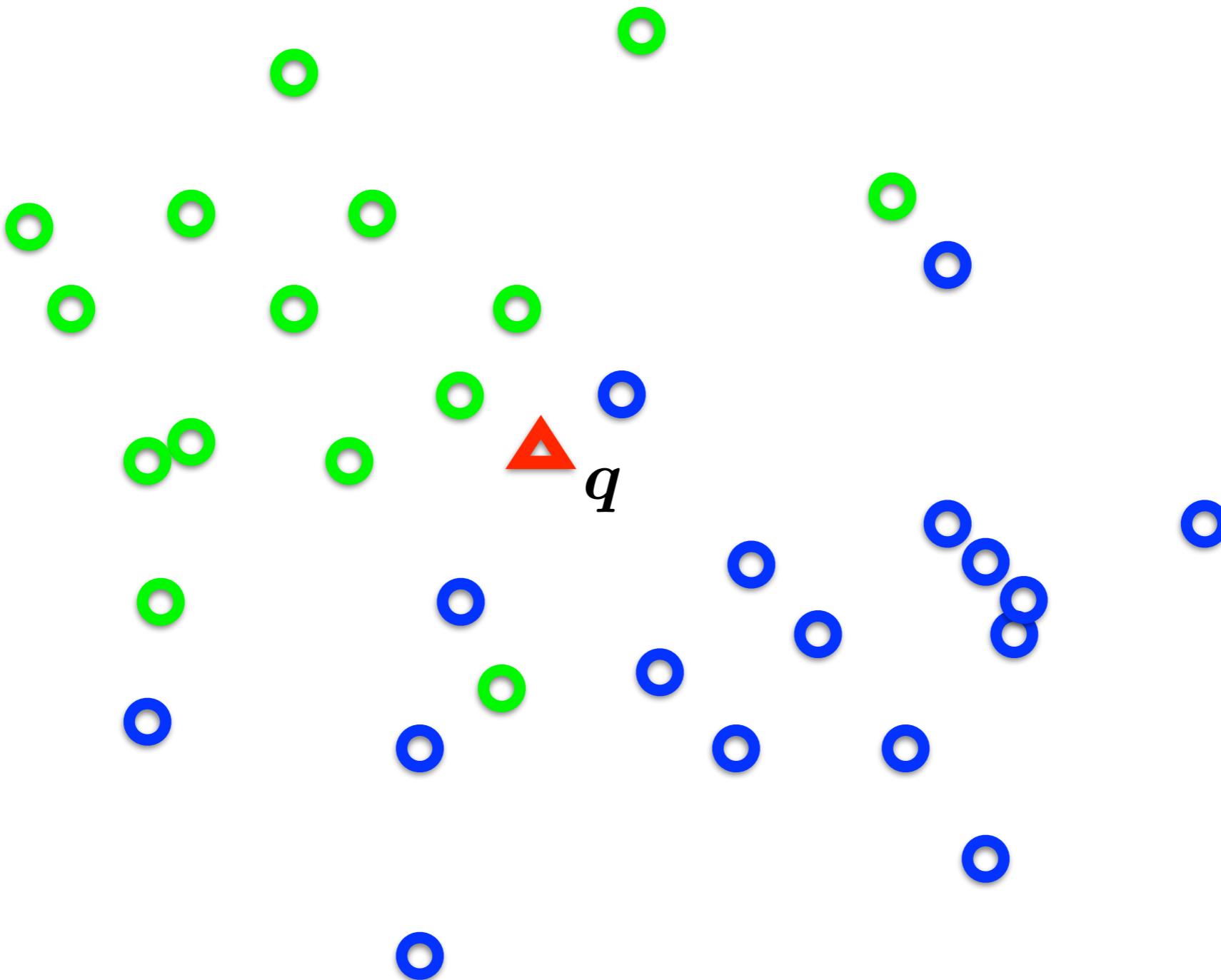
$$D(\mathbf{x}, \mathbf{y}) = \frac{1}{2} \sum_n \frac{(x_n - y_n)^2}{(x_n + y_n)}$$

Chi-squared

# Distribution of data from two classes

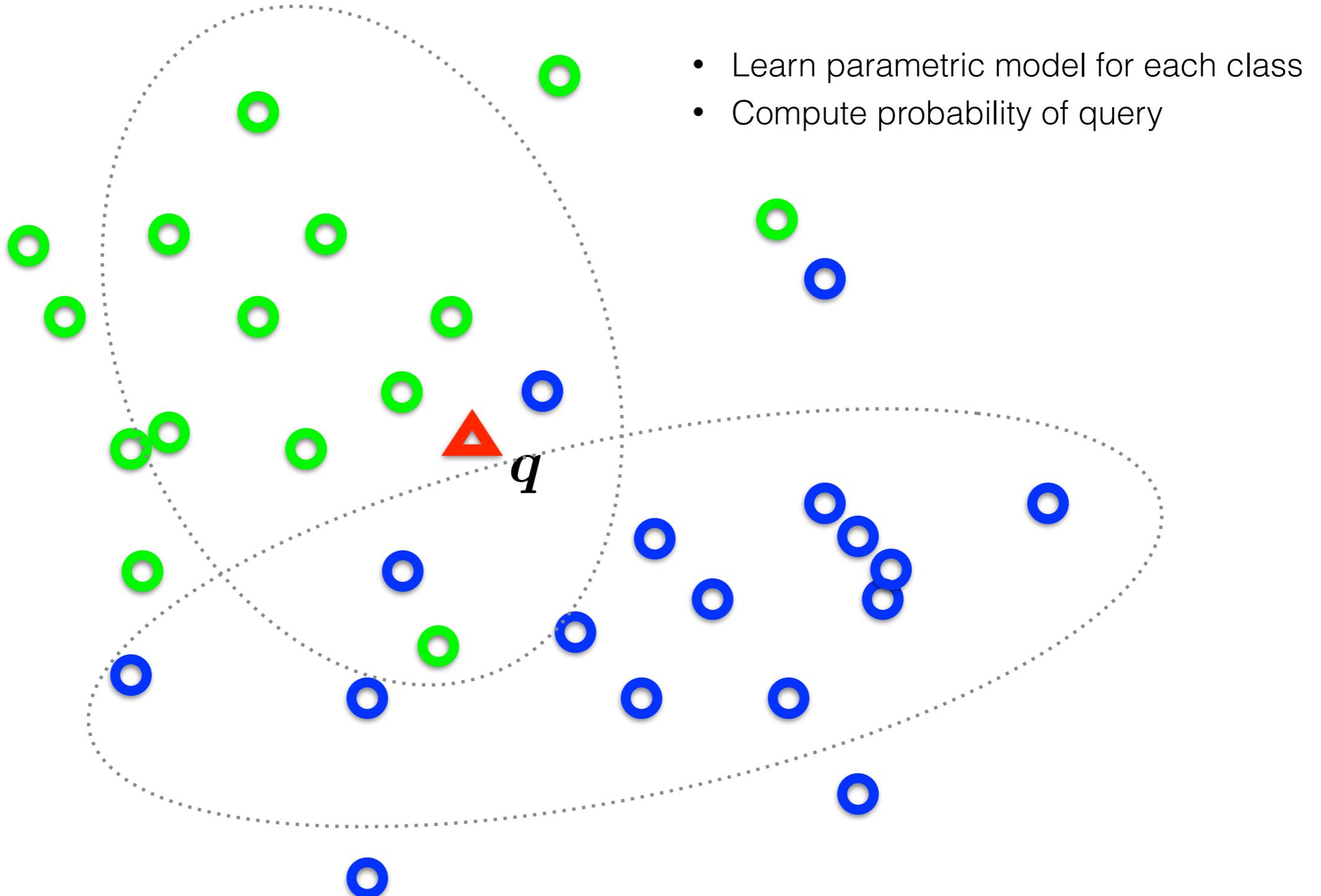


# Distribution of data from two classes



*Which class does  $q$  belong to?*

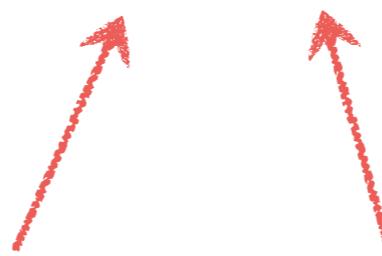
# Distribution of data from two classes



# Naive Bayes

This is called the posterior:  
the probability of a class  $z$  given the observed features  $X$

$$p(z|X)$$



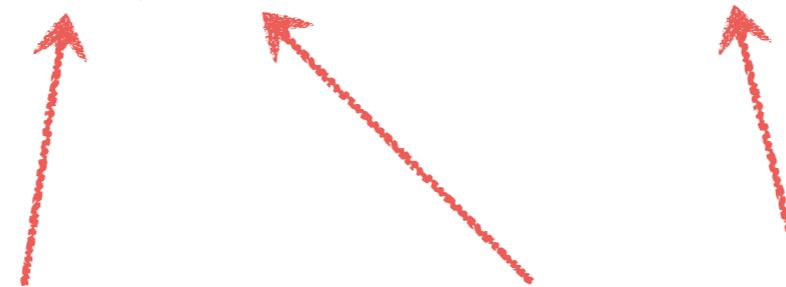
For classification,  $z$  is a  
discrete random variable  
(e.g., car, person, building)

$X$  is a set of observed feature  
(e.g., features from a single image)

(it's a function that returns a single probability value)

This is called the posterior:  
the probability of a class  $z$  given the observed features  $X$

$$p(z|x_1, \dots, x_N)$$



For classification,  $z$  is a  
discrete random variable  
(e.g., car, person, building)

Each  $x$  is an observed feature  
(e.g., visual words)

(it's a function that returns a single probability value)

## Recall:

The posterior can be decomposed according to  
**Bayes' Rule**

$$p(A|B) = \frac{\text{likelihood} \quad \text{prior}}{p(B)}$$

posterior

In our context...

$$p(z|x_1, \dots, x_N) = \frac{p(x_1, \dots, x_N | z)p(z)}{p(x_1, \dots, x_N)}$$

The naive Bayes' classifier is solving this optimization

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} p(z | \mathbf{X})$$

MAP (maximum a posteriori) estimate

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} \frac{p(\mathbf{X}|z)p(z)}{p(\mathbf{X})} \quad \text{Bayes' Rule}$$

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} p(\mathbf{X}|z)p(z) \quad \text{Remove constants}$$

To optimize this...

$$\hat{z} = \arg \max_{z \in \mathcal{Z}} p(z|X)$$

We need to compute this



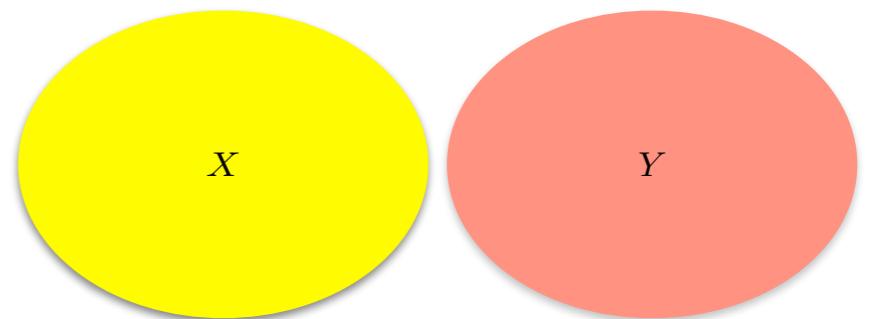
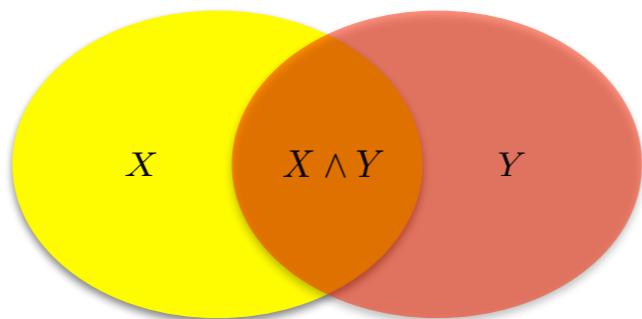
$$p(z|x_1, \dots, x_N) = \frac{p(x_1, \dots, x_N | z)p(z)}{p(x_1, \dots, x_N)}$$

Compute the likelihood...

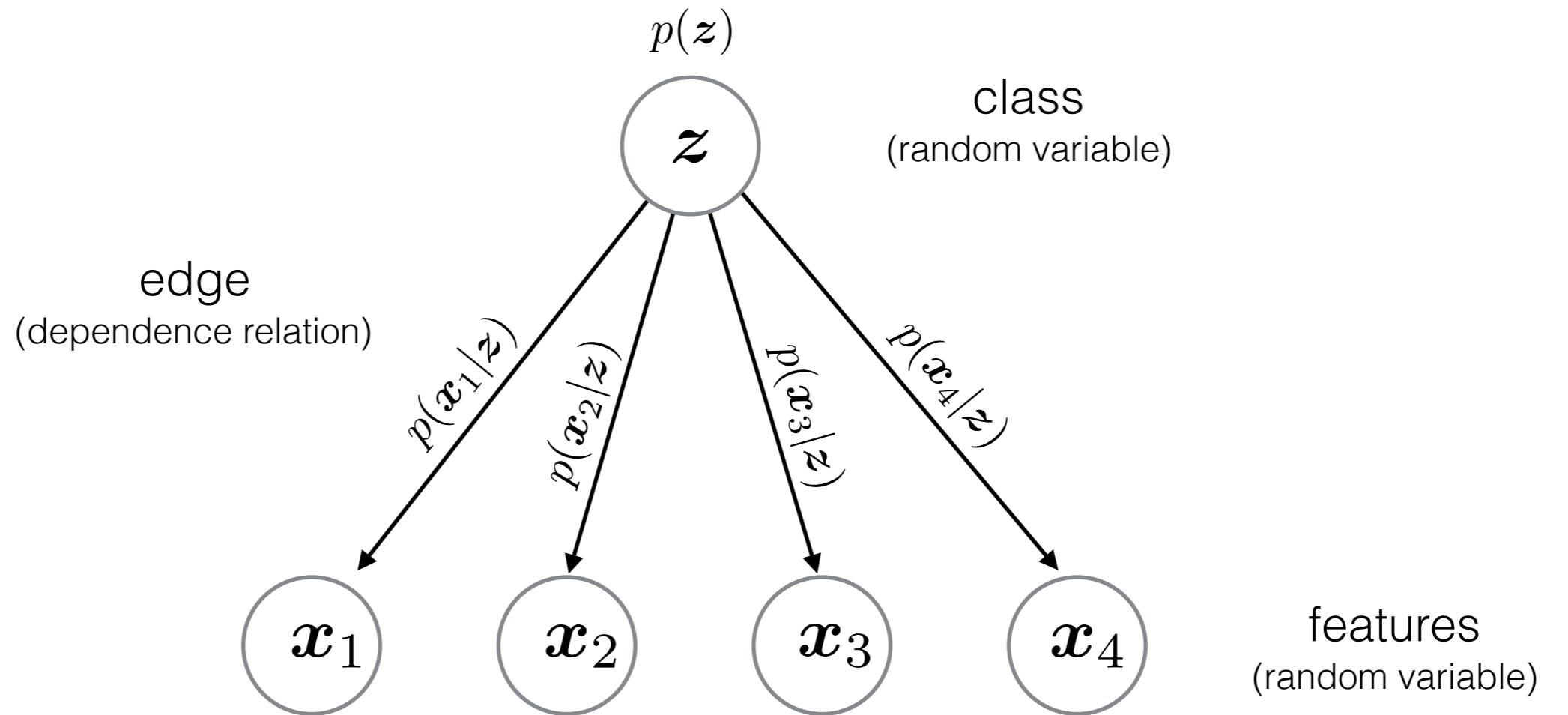
A naive Bayes' classifier assumes all features are  
***conditionally independent***

$$\begin{aligned} p(\mathbf{x}_1, \dots, \mathbf{x}_N | \mathbf{z}) &= p(\mathbf{x}_1 | \mathbf{z})p(\mathbf{x}_2, \dots, \mathbf{x}_N | \mathbf{z}) \\ &= p(\mathbf{x}_1 | \mathbf{z})p(\mathbf{x}_2 | \mathbf{z})p(\mathbf{x}_3, \dots, \mathbf{x}_N | \mathbf{z}) \\ &= p(\mathbf{x}_1 | \mathbf{z})p(\mathbf{x}_2 | \mathbf{z}) \cdots p(\mathbf{x}_N | \mathbf{z}) \end{aligned}$$

**Recall:**



$$p(x, y) = p(x|y)p(y) \quad p(x, y) = p(x)p(y)$$



Graphical model visualization

To compute the MAP estimate

Given (1) a set of known parameters

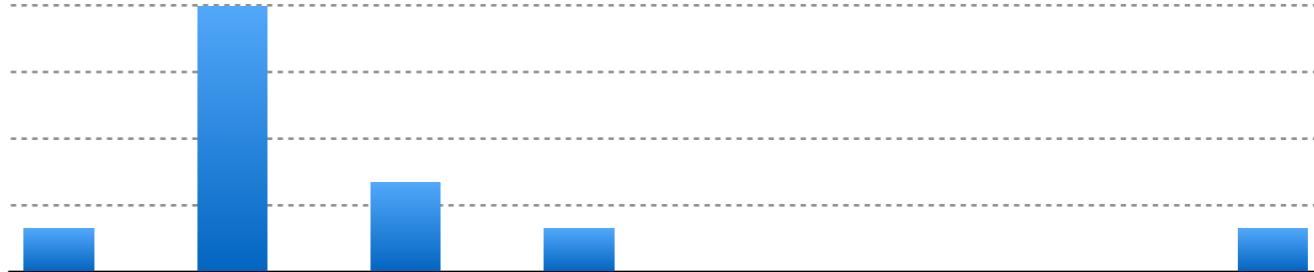
$$p(\mathbf{z}) \quad p(\mathbf{x}|\mathbf{z})$$

(2) observations

$$\{x_1, x_2, \dots, x_N\}$$

Compute which  $\mathbf{z}$  has the largest probability

$$\hat{\mathbf{z}} = \arg \max_{\mathbf{z} \in \mathcal{Z}} p(\mathbf{z}) \prod_n p(x_n | \mathbf{z})$$



count	1	6	2	1	0	0	0	1
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x z)	0.09	0.55	0.18	0.09	0.0	0.0	0.0	0.09

$$\begin{aligned}
 p(X|z) &= \prod_v p(x_v|z)^{c(w_v)} \\
 &= (0.09)^1 (0.55)^6 \cdots (0.09)^1
 \end{aligned}$$

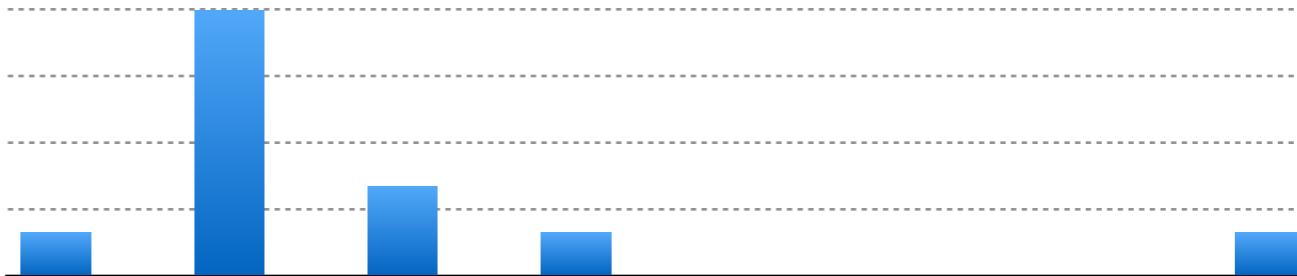
Numbers get really small so use log probabilities

$$\log p(X|z = \text{'grandchallenge'}) = -2.42 - 3.68 - 3.43 - 2.42 - 0.07 - 0.07 - 0.07 - 2.42 = -14.58$$

$$\log p(X|z = \text{'softrobot'}) = -7.63 - 9.37 - 15.18 - 2.97 - 0.02 - 0.01 - 0.02 - 2.27 = -37.48$$

\* typically add pseudo-counts (0.001)

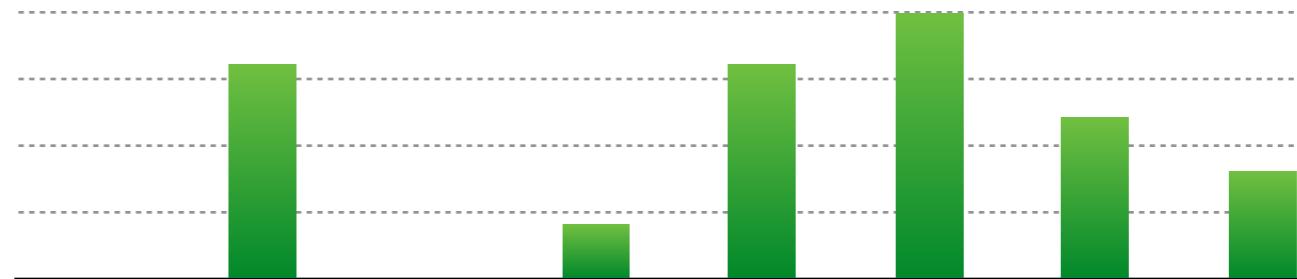
\*\* this is an example for computing the likelihood, need to multiply times **prior** to get posterior



count	1	6	2	1	0	0	0	1
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x z)	0.09	0.55	0.18	0.09	0.0	0.0	0.0	0.09

$$\log p(X|z=\text{grand challenge}) = -14.58$$

$$\log p(X|z=\text{bio inspired}) = -37.48$$



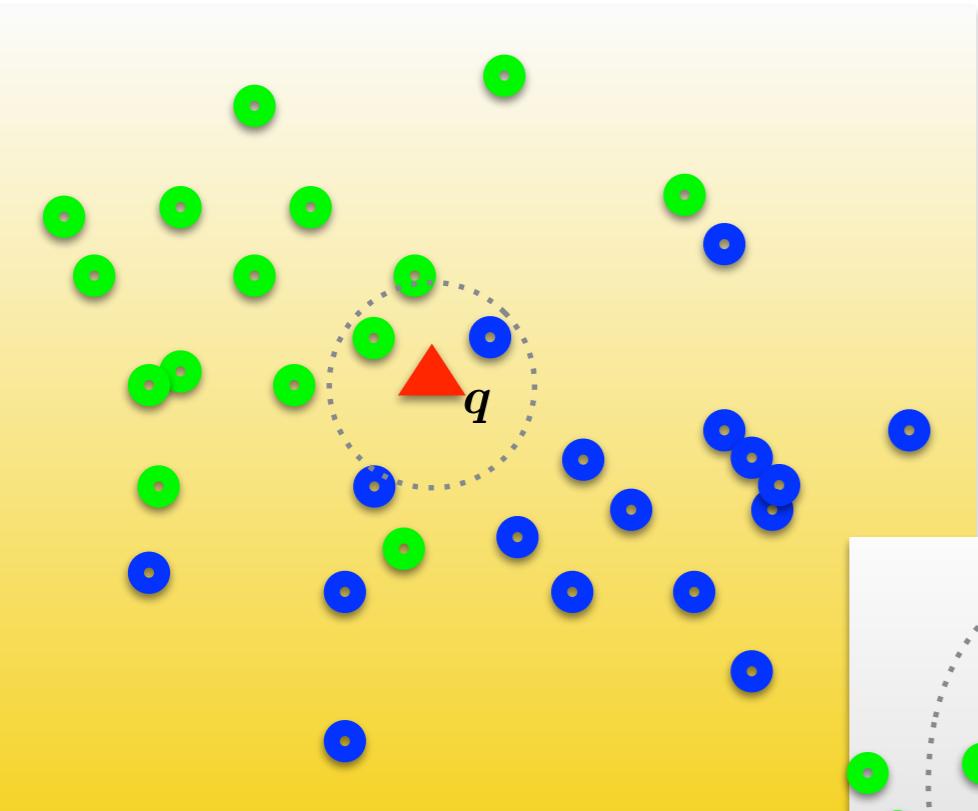
count	0	4	0	1	4	5	3	2
word	Tartan	robot	CHIMP	CMU	bio	soft	ankle	sensor
p(x z)	0.0	0.21	0.0	0.05	0.21	0.26	0.16	0.11

$$\log p(X|z=\text{grand challenge}) = -94.06$$

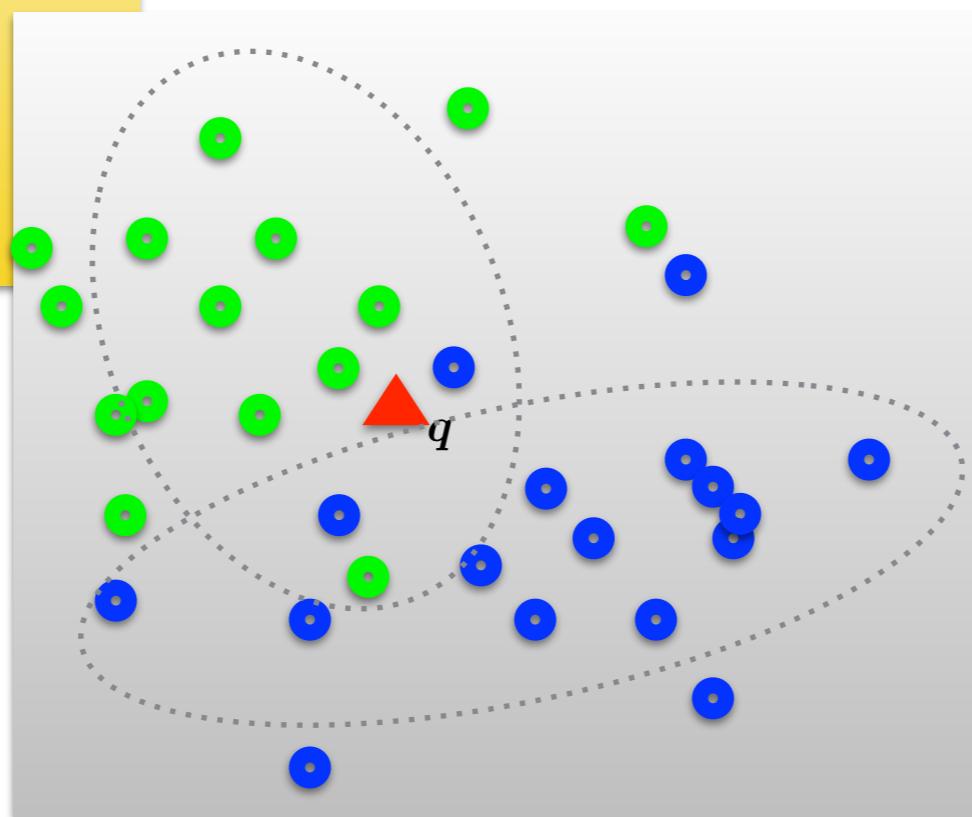
$$\log p(X|z=\text{bio inspired}) = -32.41$$

\* typically add pseudo-counts (0.001)

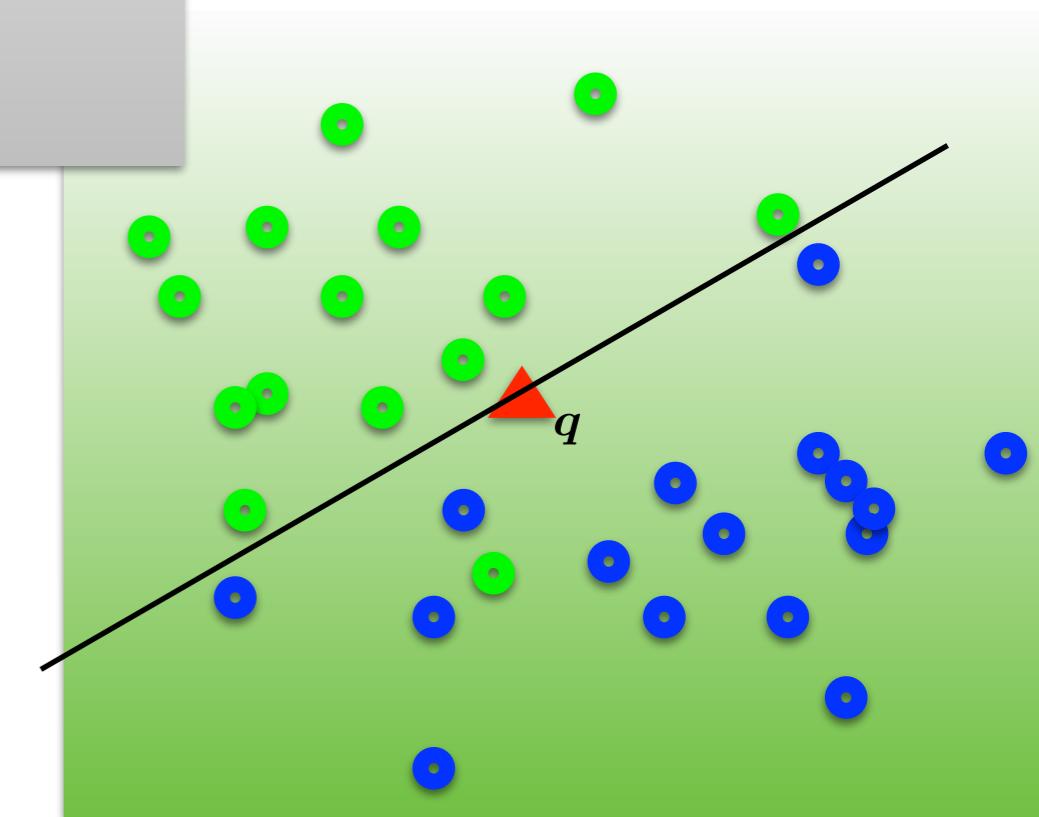
\*\* this is an example for computing the likelihood, need to multiply times prior to get posterior



Nearest Neighbor

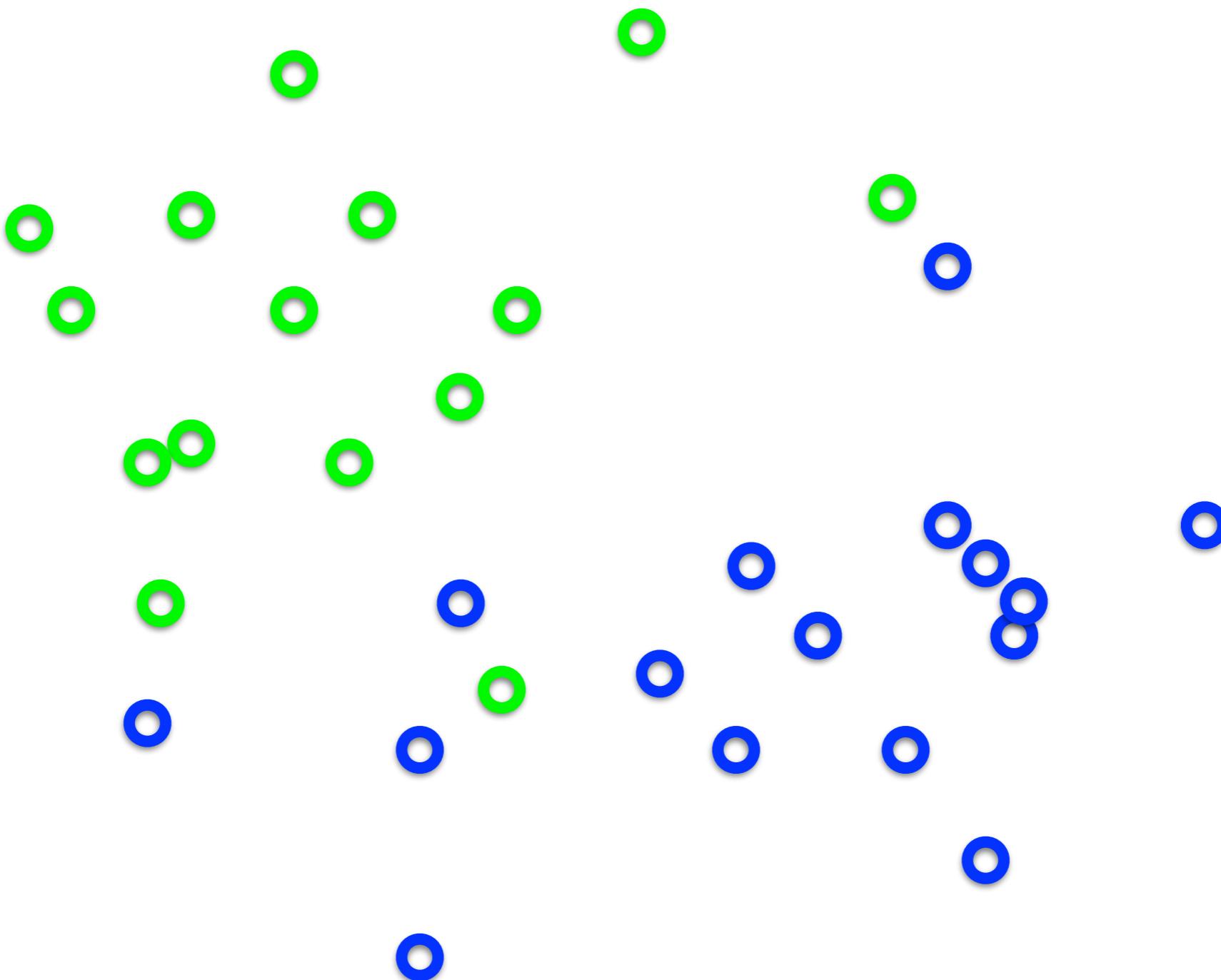


Naive Bayes

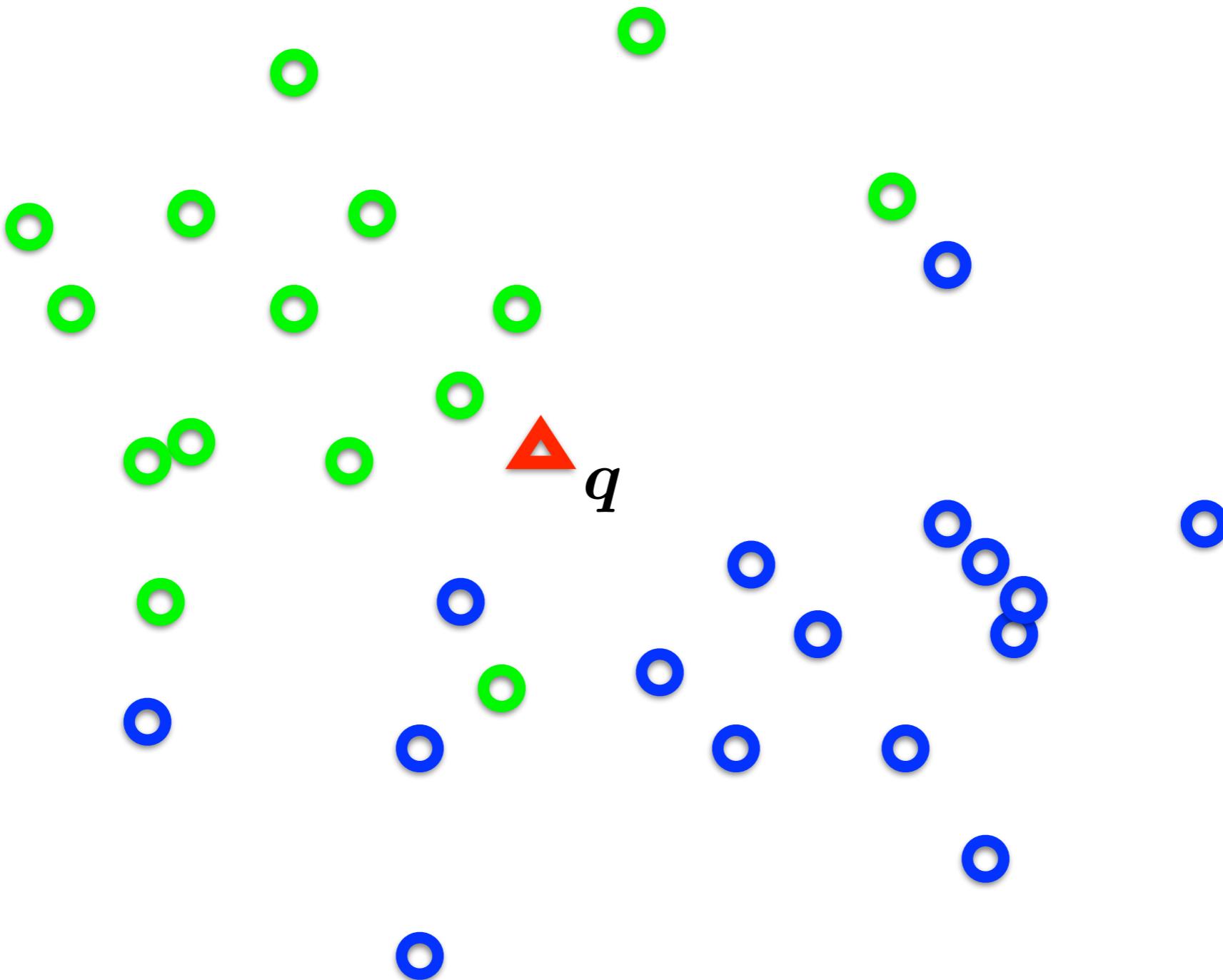


Support Vector Machine

# Distribution of data from two classes

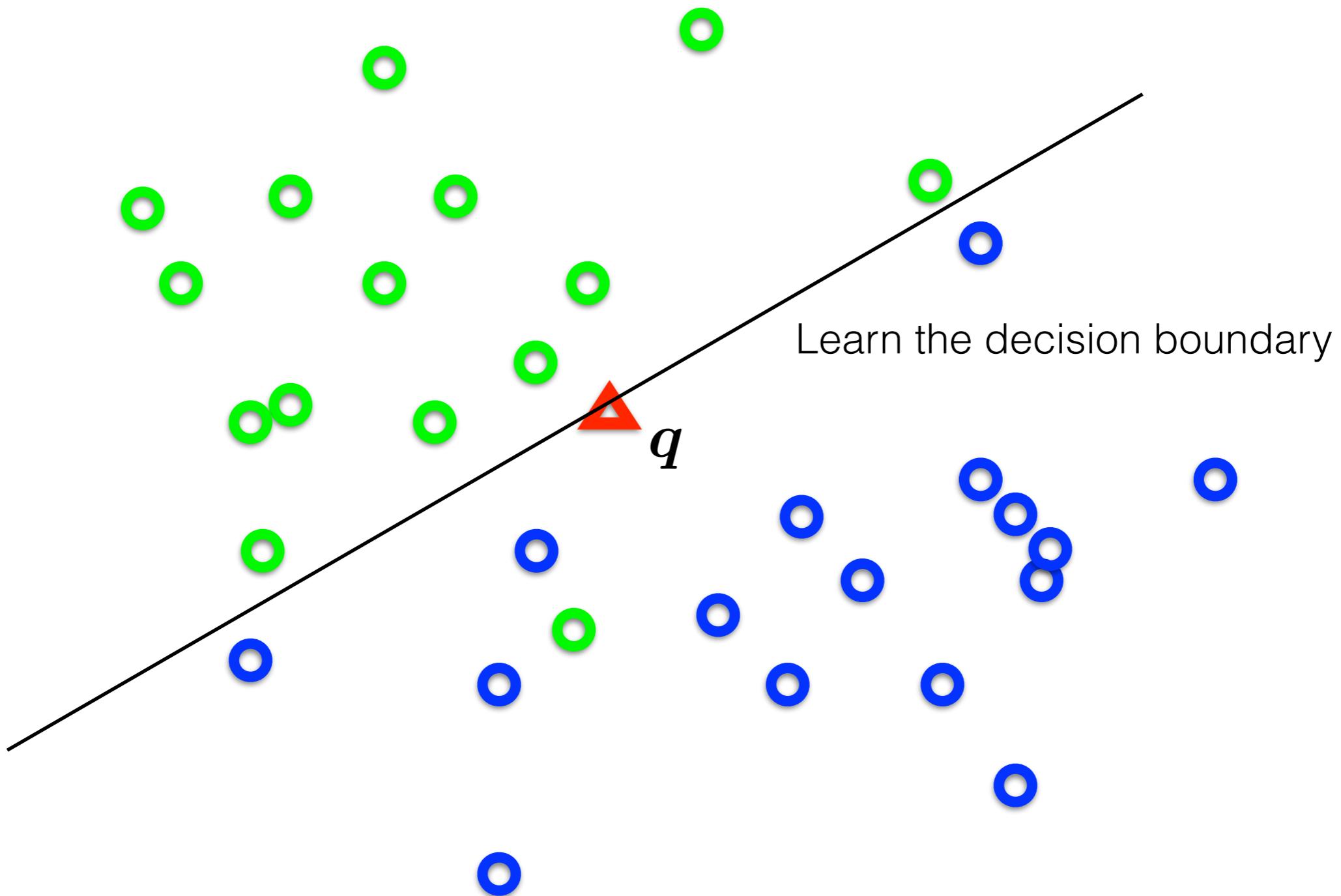


# Distribution of data from two classes



*Which class does  $q$  belong to?*

# Distribution of data from two classes

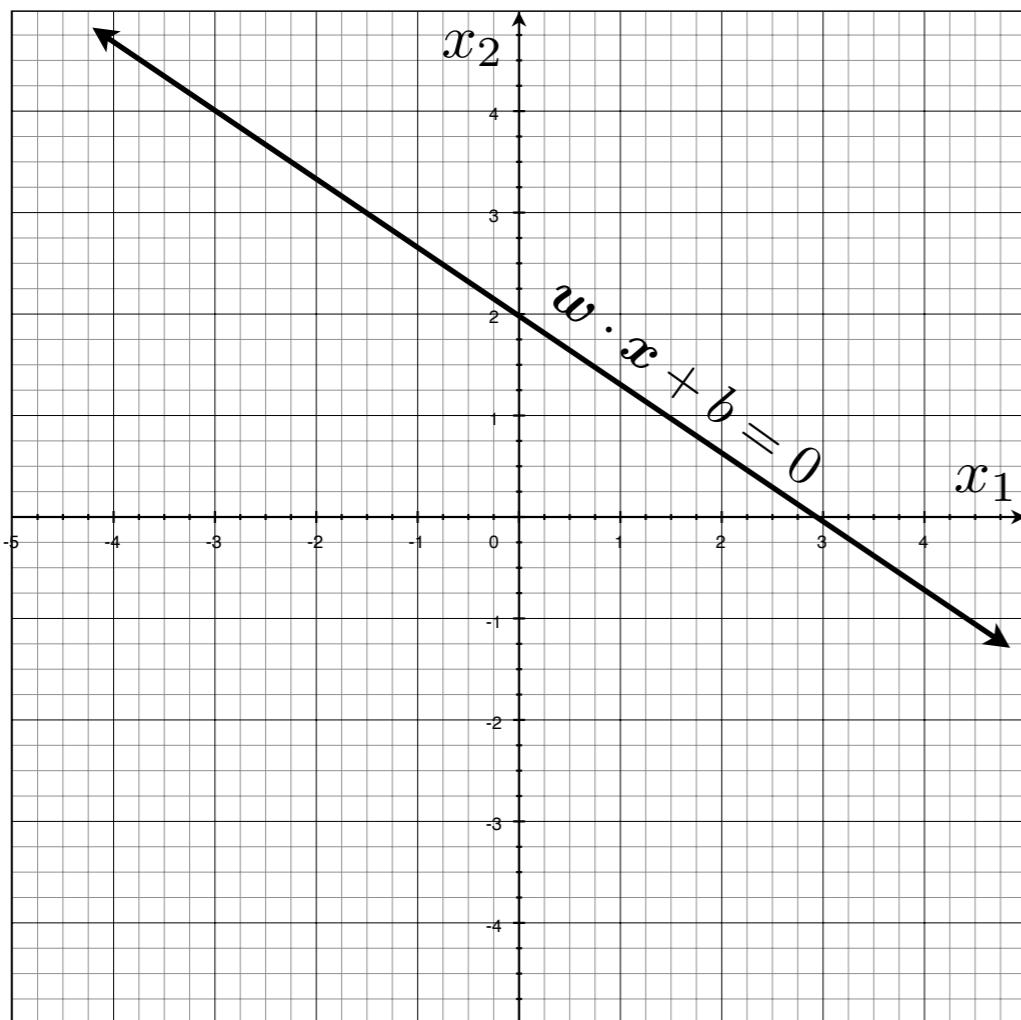


# Support Vector Machine

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

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

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

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

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

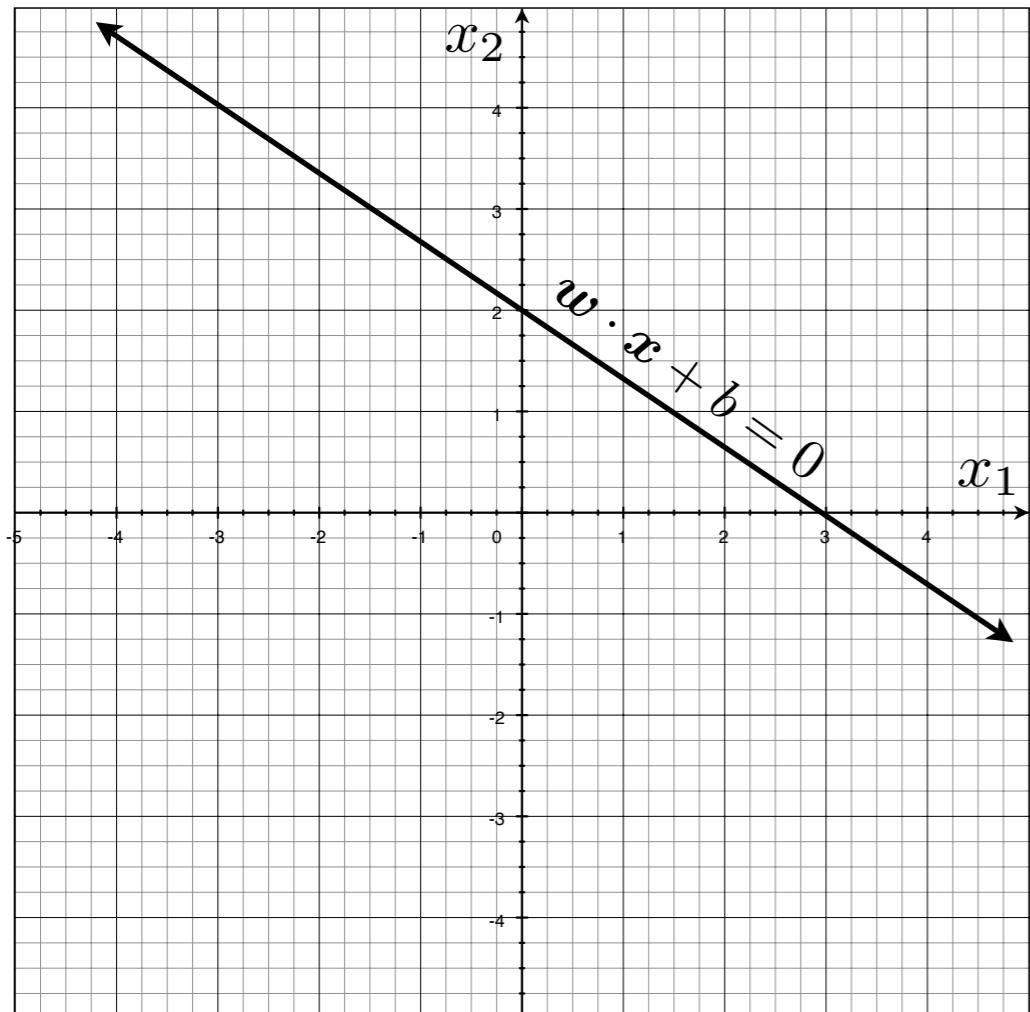
$$\mathbf{w} \in \mathcal{R}^3$$

# Hyperplanes (lines) in 2D

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

---

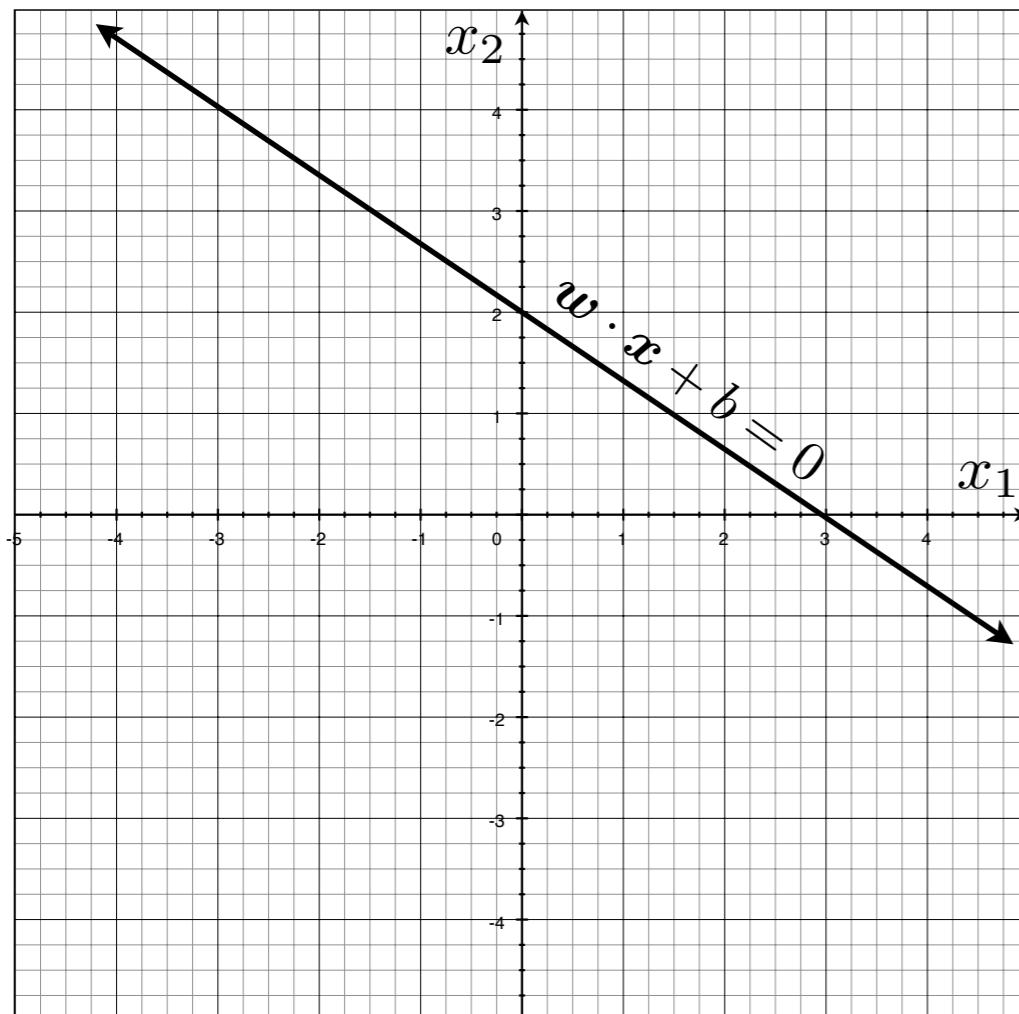
$$w_1x_1 + w_2x_2 + b = 0$$



# Hyperplanes (lines) in 2D

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

$$w_1x_1 + w_2x_2 + b = 0$$



**Important property:**  
*Free to choose any normalization of  $w$*

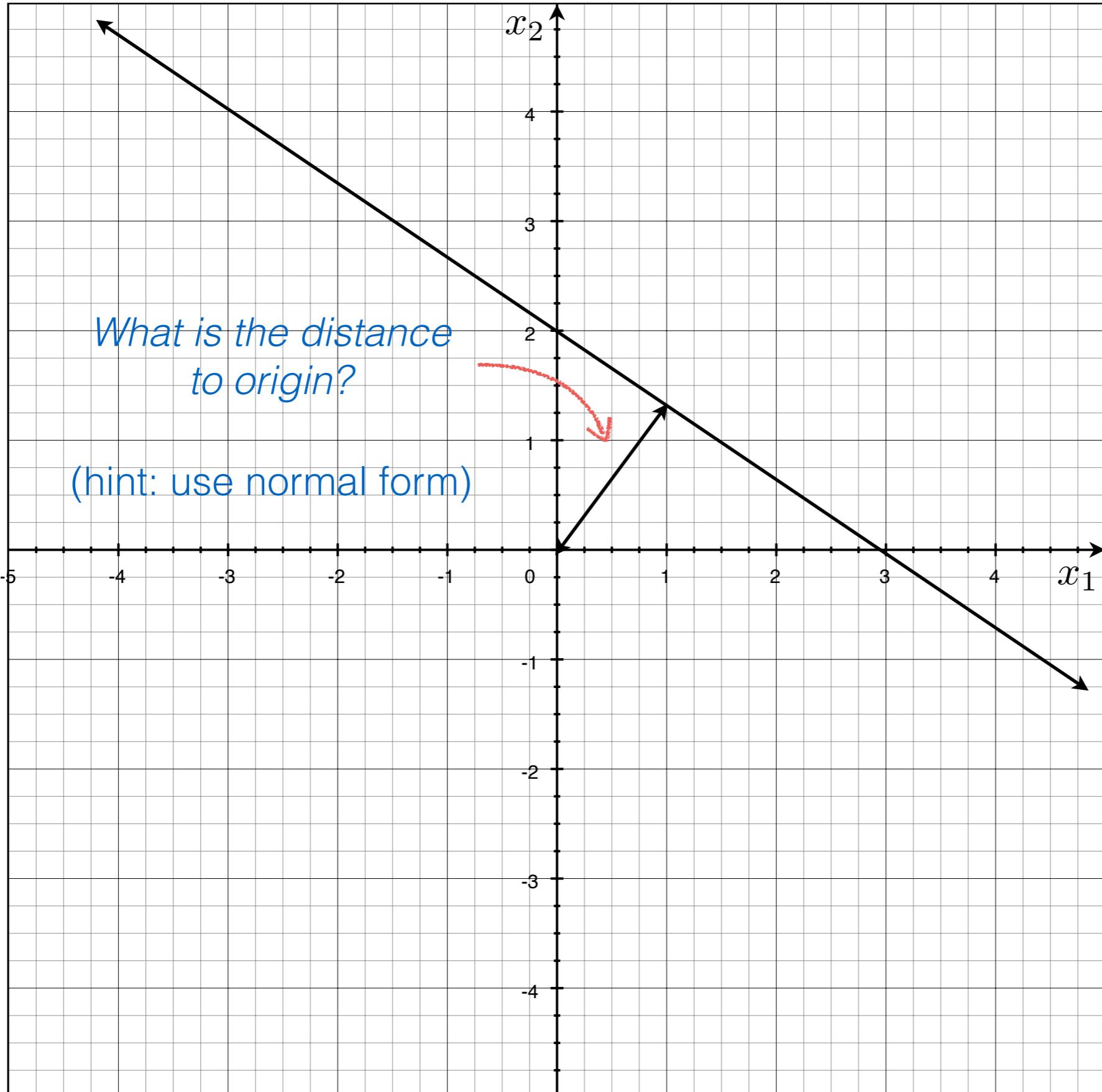
The line

$$w_1x_1 + w_2x_2 + b = 0$$

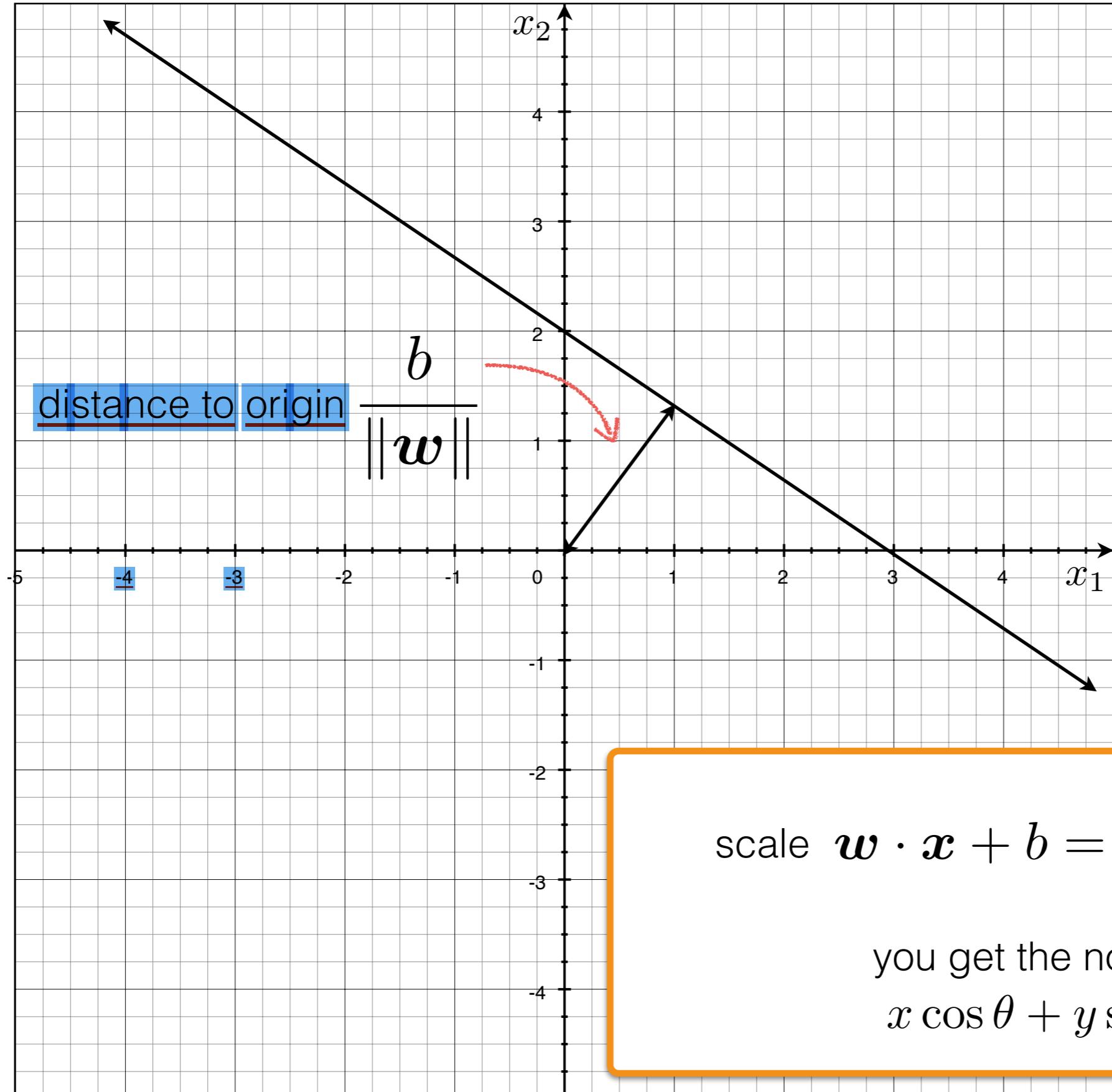
and the line

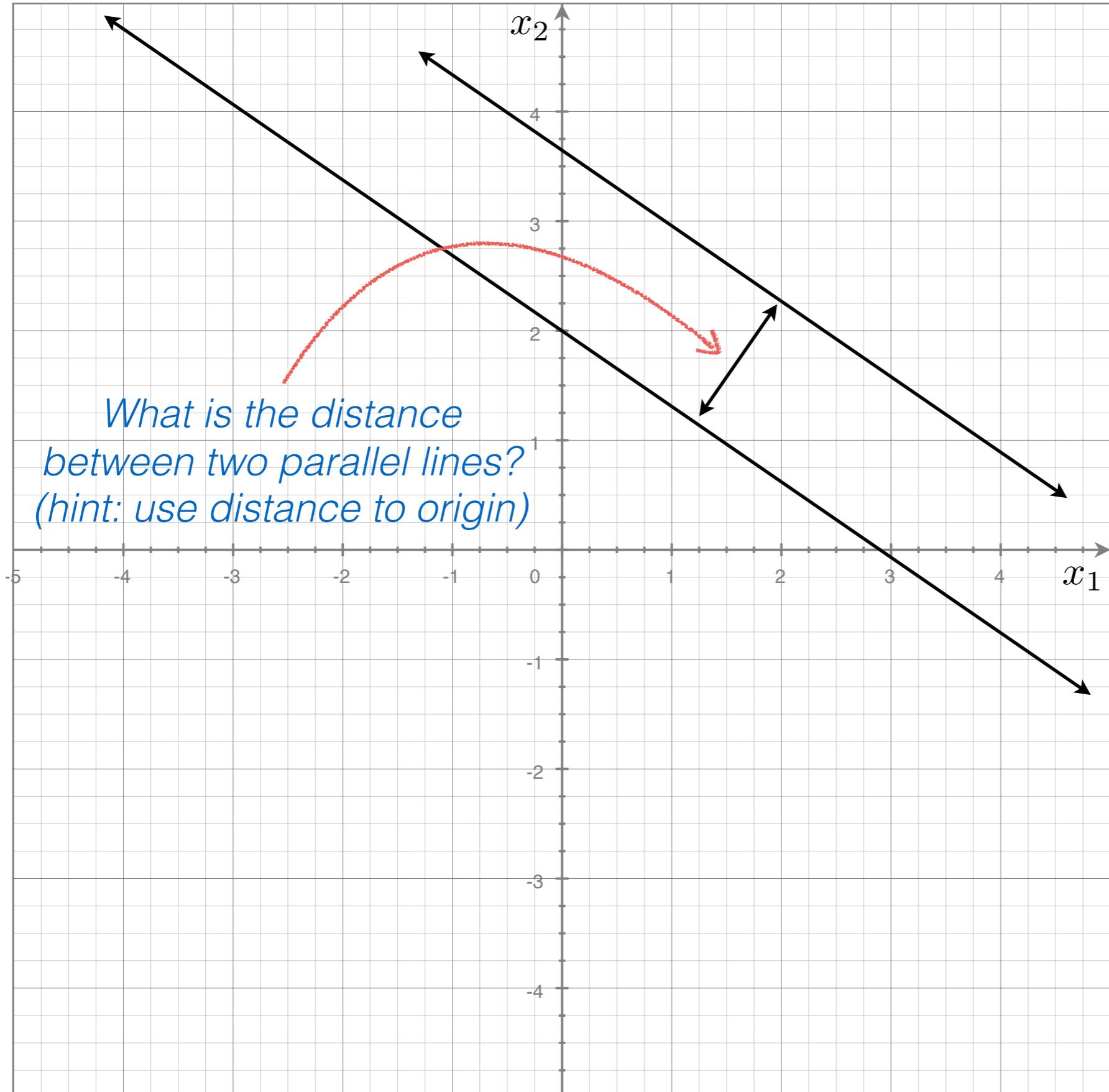
$$\lambda(w_1x_1 + w_2x_2 + b) = 0$$

define the same line



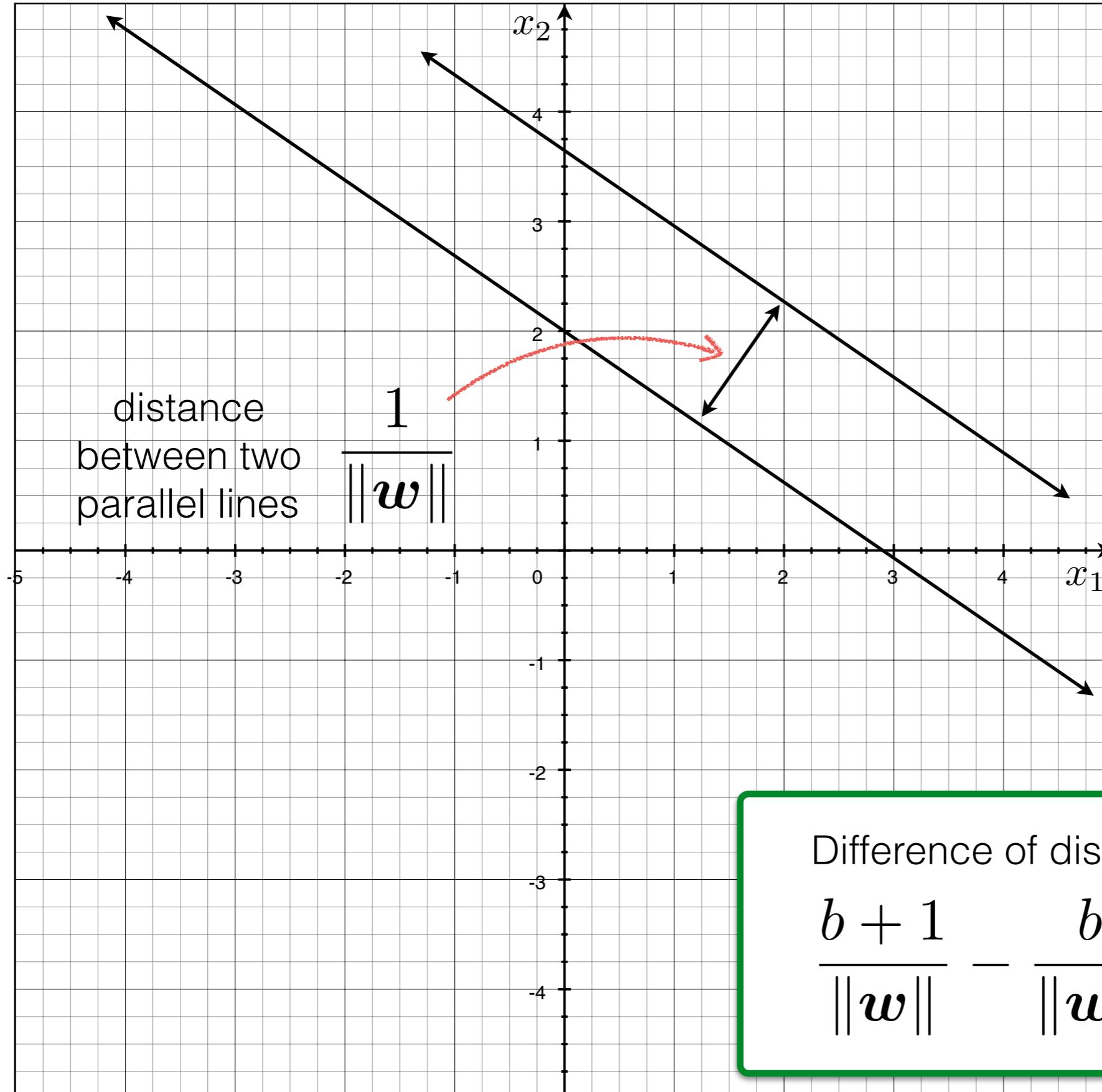
$$\mathbf{w} \cdot \mathbf{x} + b = 0$$





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

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

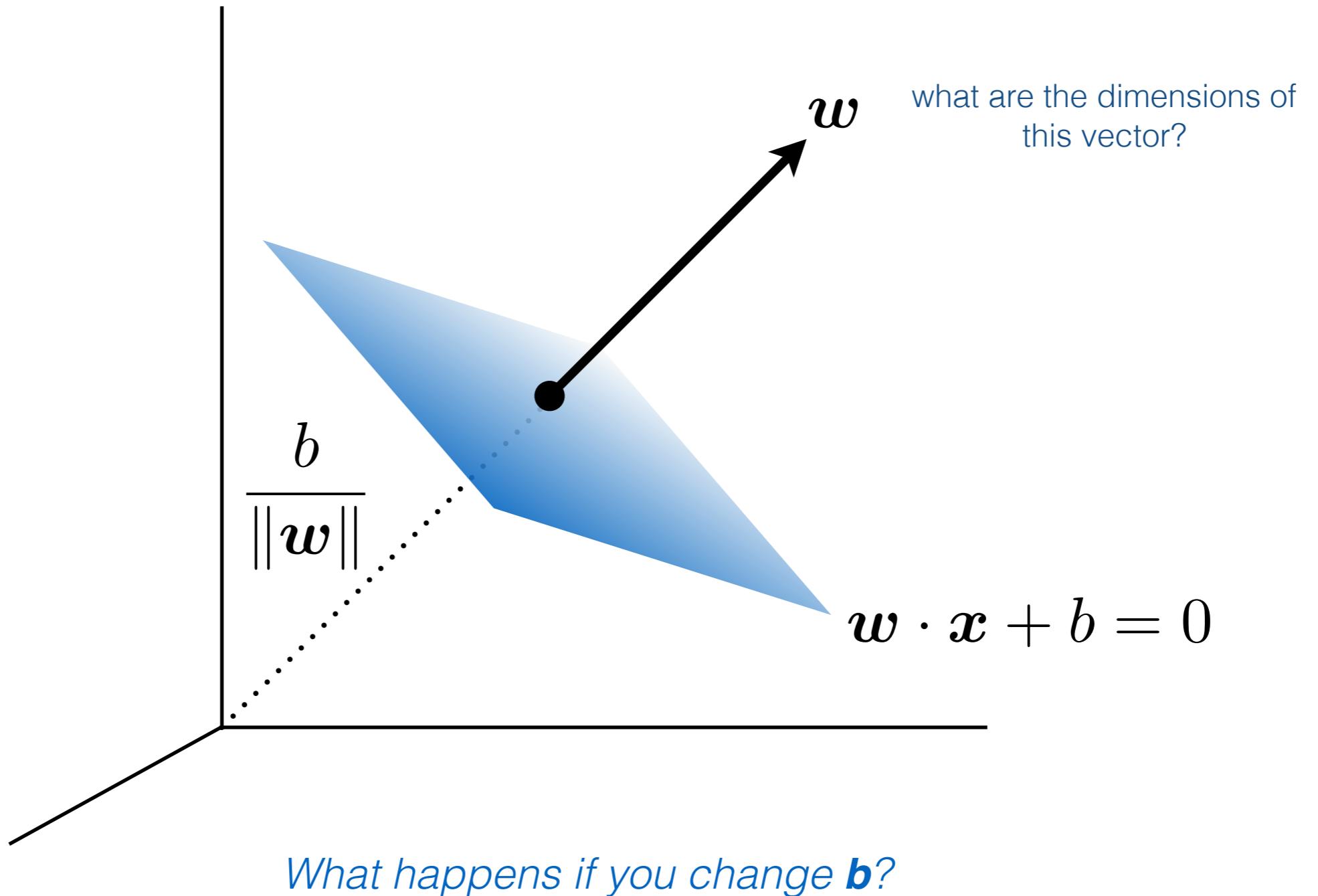


Difference of distance to origin

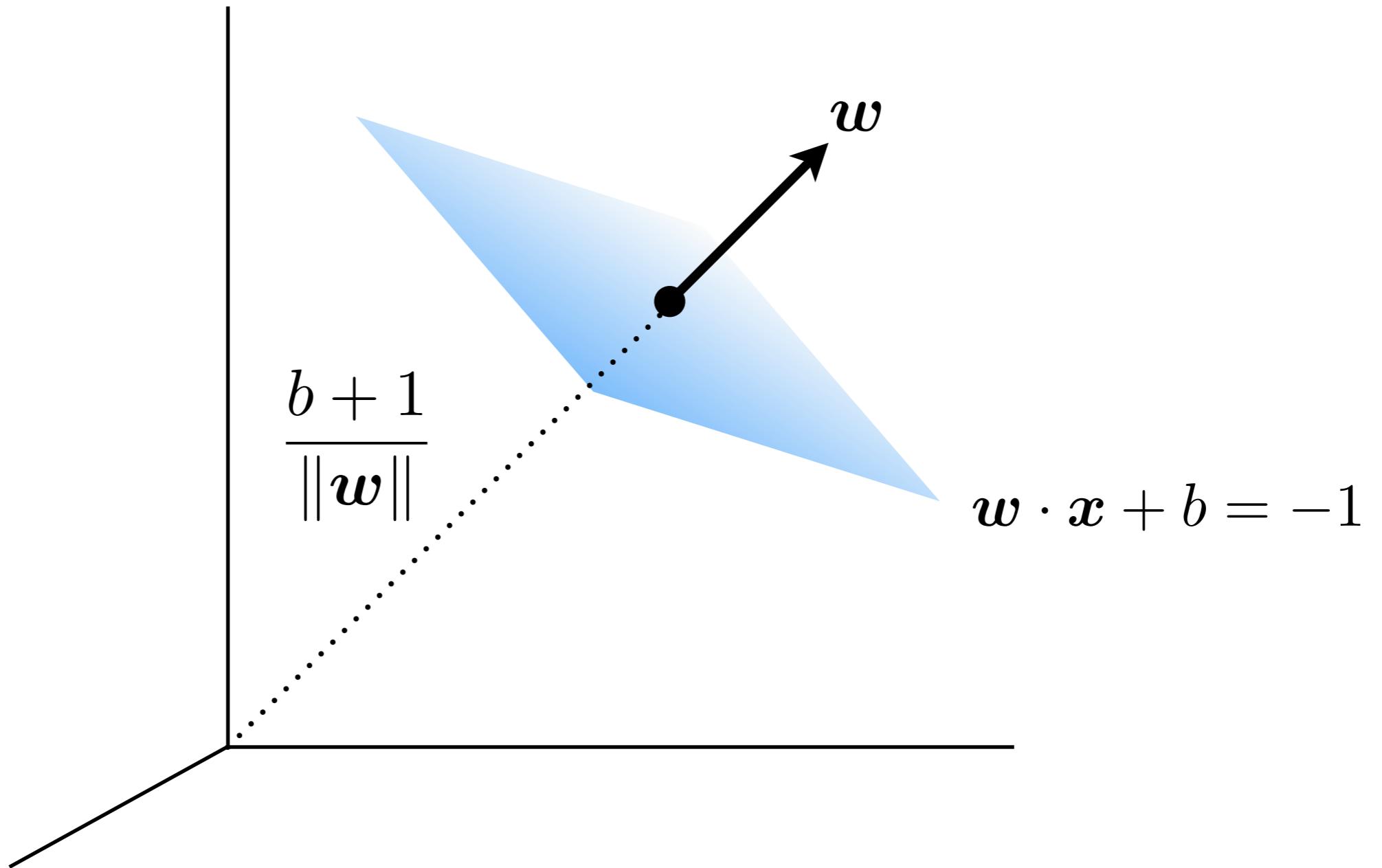
$$\frac{b+1}{\|w\|} - \frac{b}{\|w\|} = \frac{1}{\|w\|}$$

Now we can go to 3D ...

# Hyperplanes (planes) in 3D

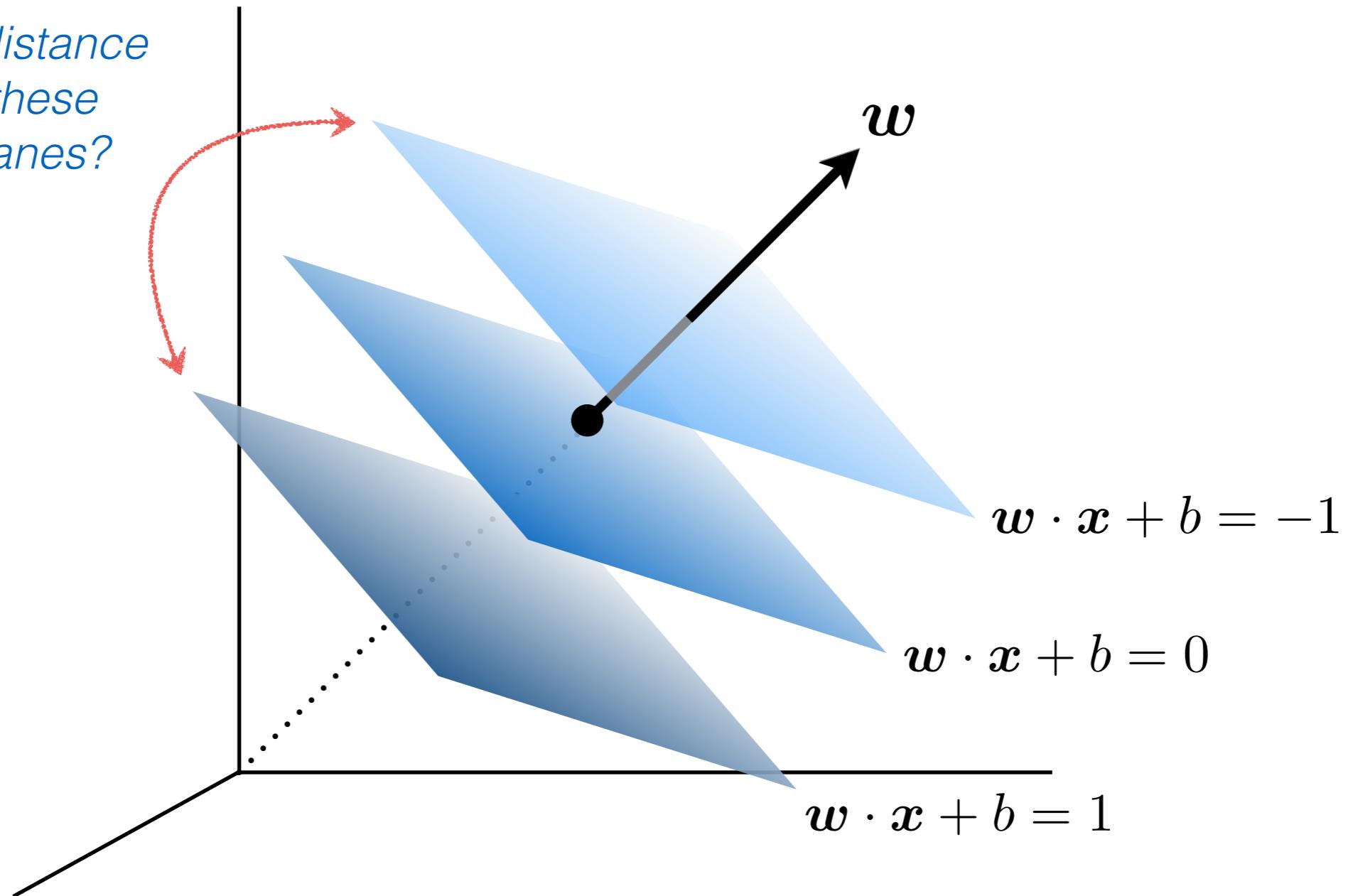


# Hyperplanes (planes) in 3D

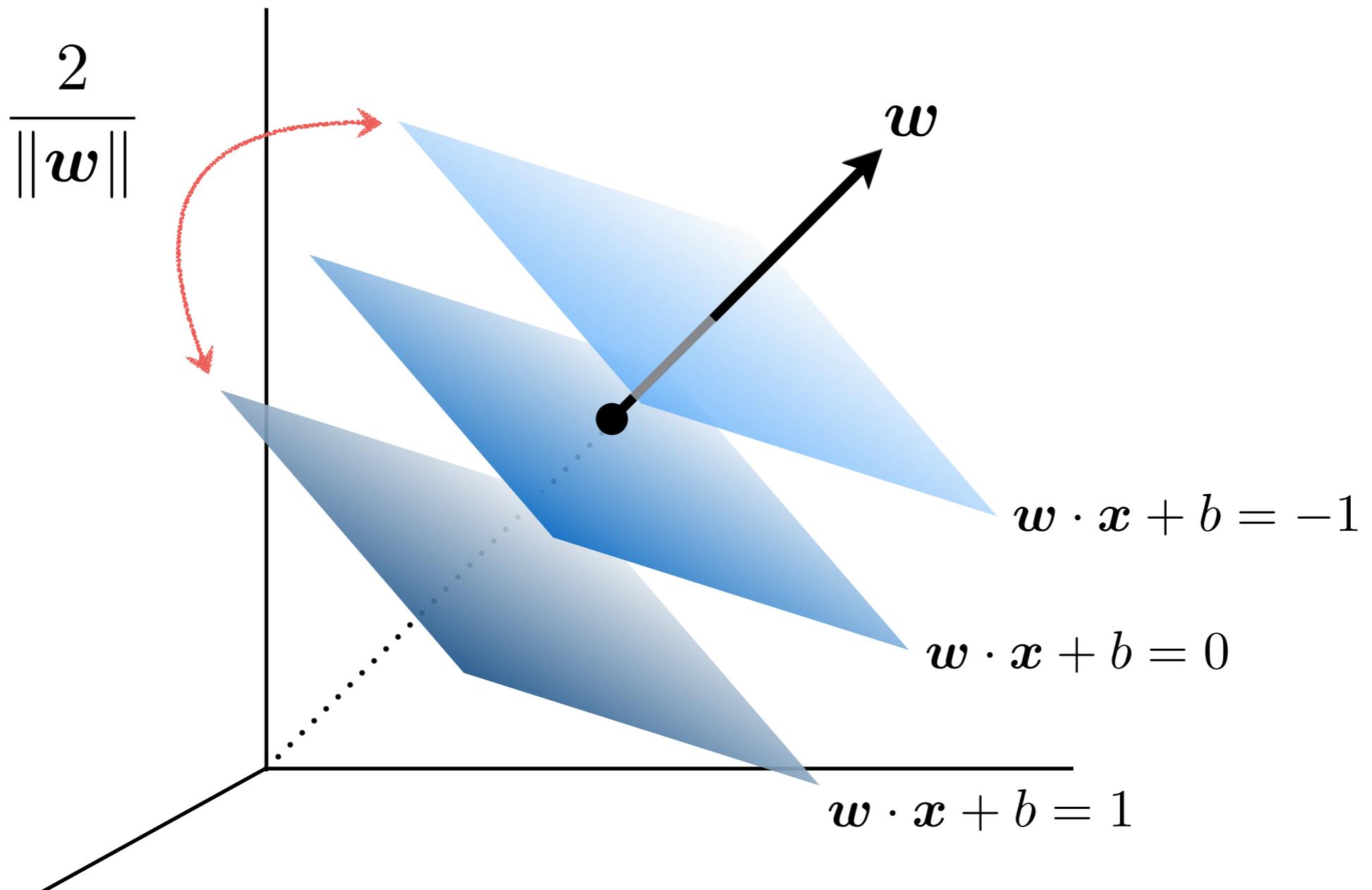


# Hyperplanes (planes) in 3D

*What's the distance  
between these  
parallel planes?*

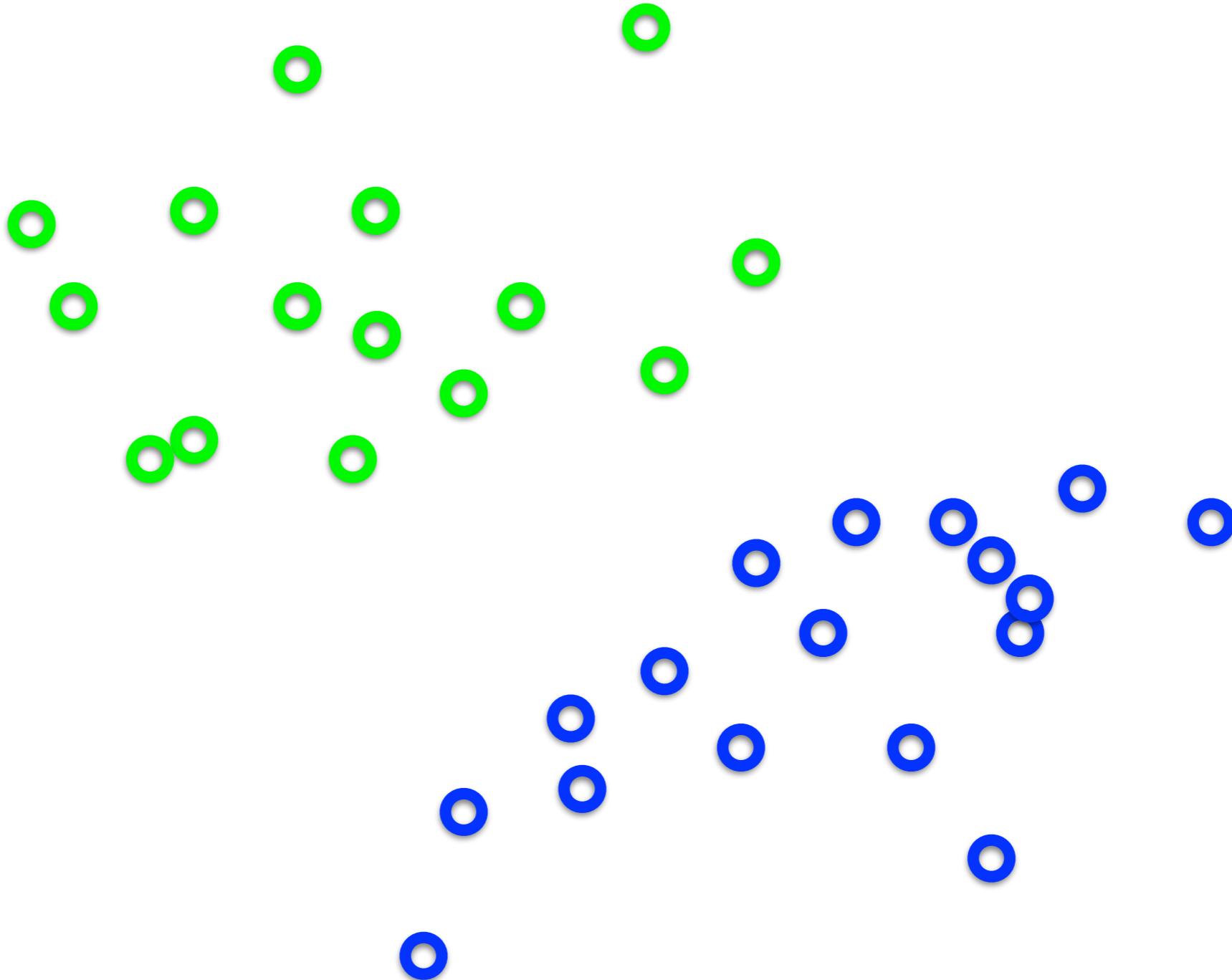


# Hyperplanes (planes) in 3D

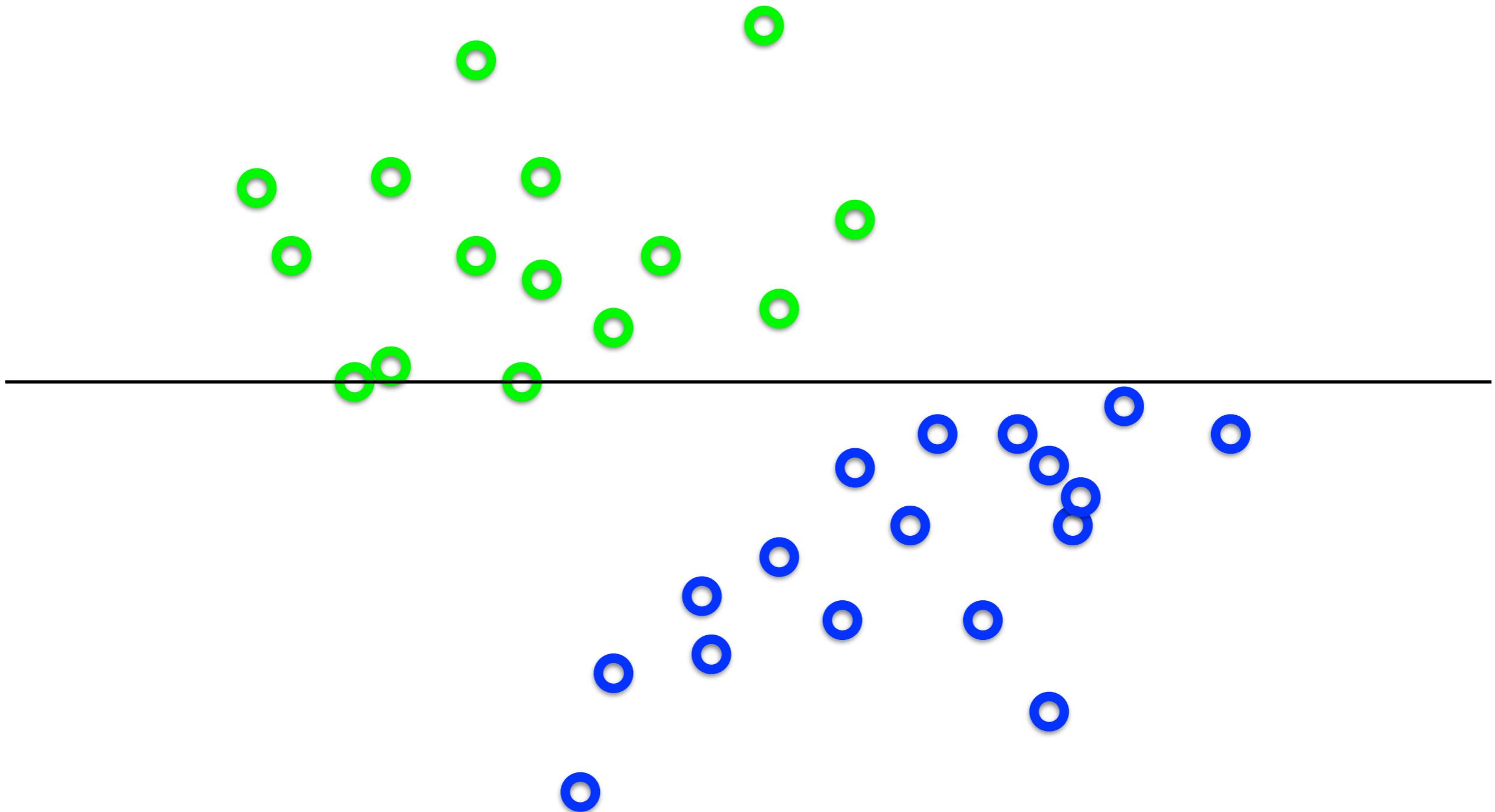


# Support Vector Machine

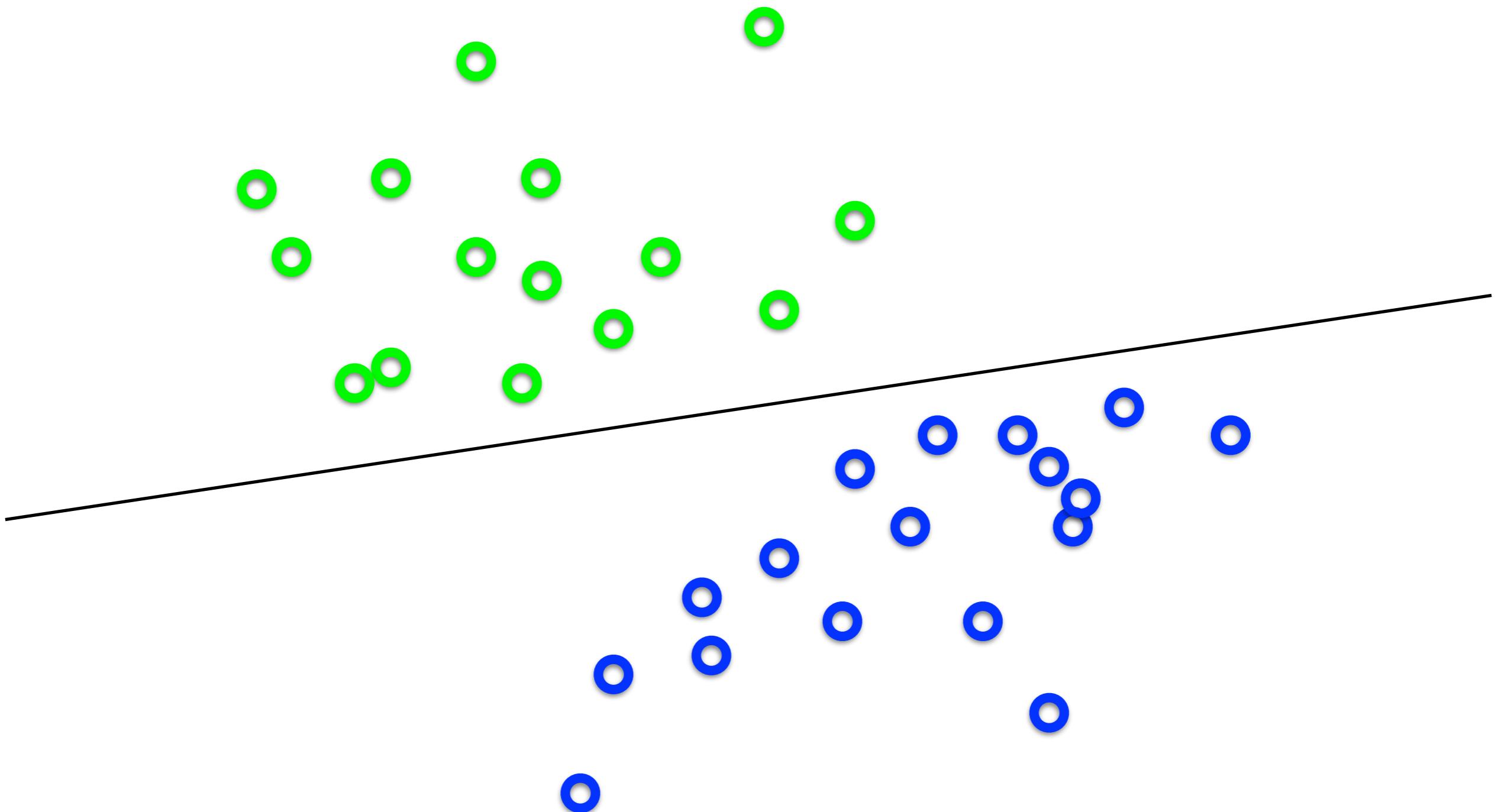
What's the best **w**?



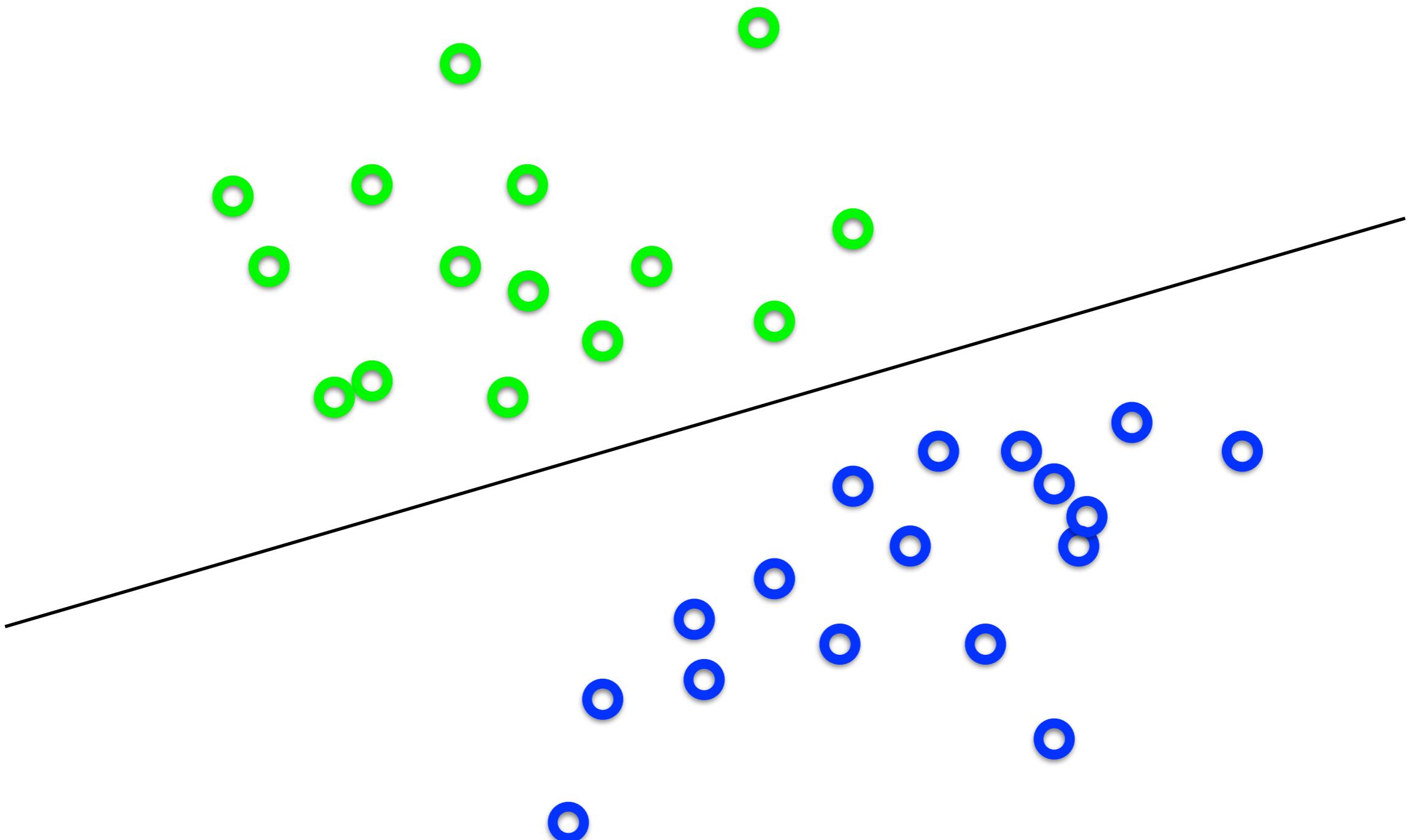
# What's the best **w**?



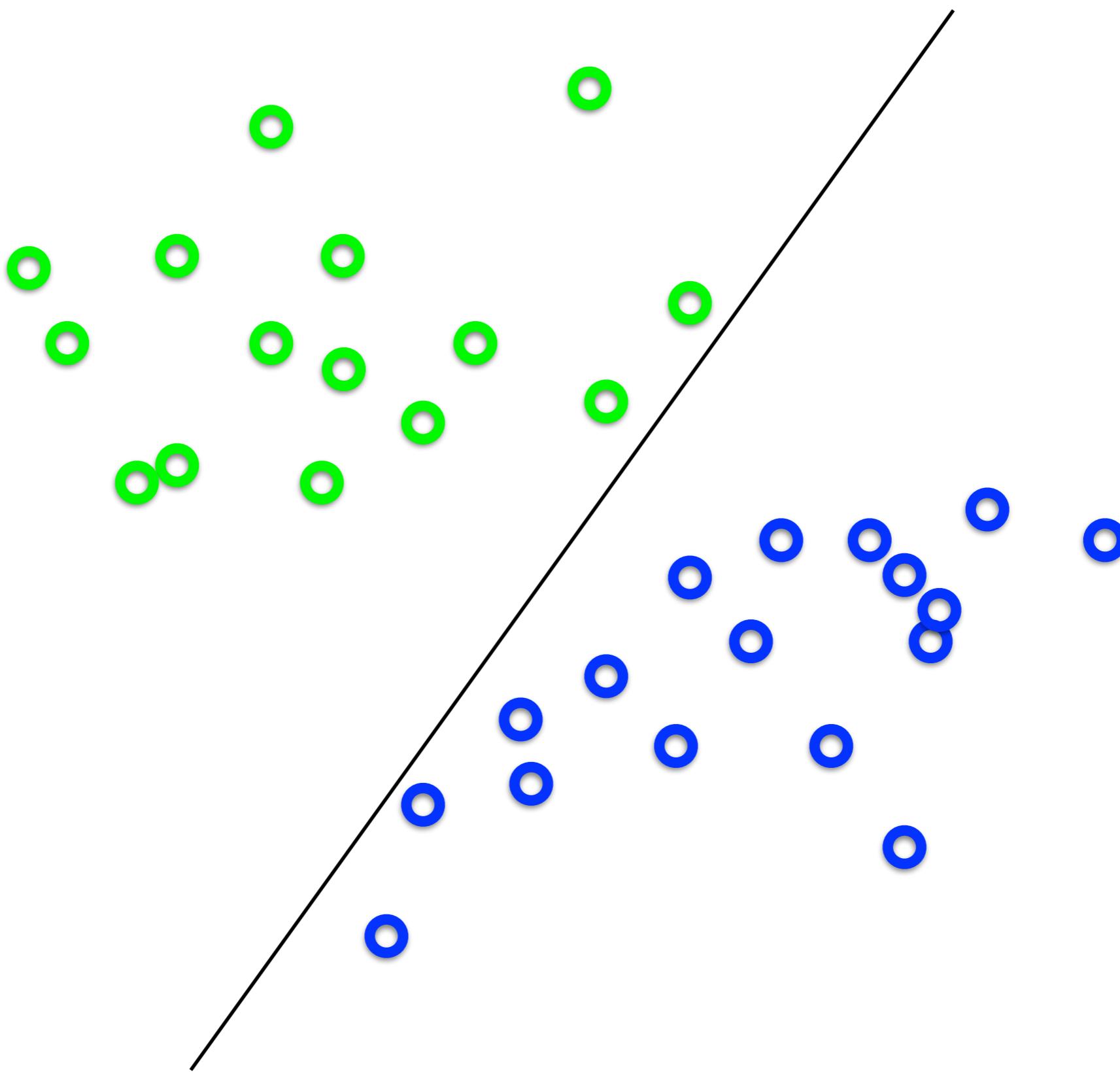
# What's the best $w$ ?



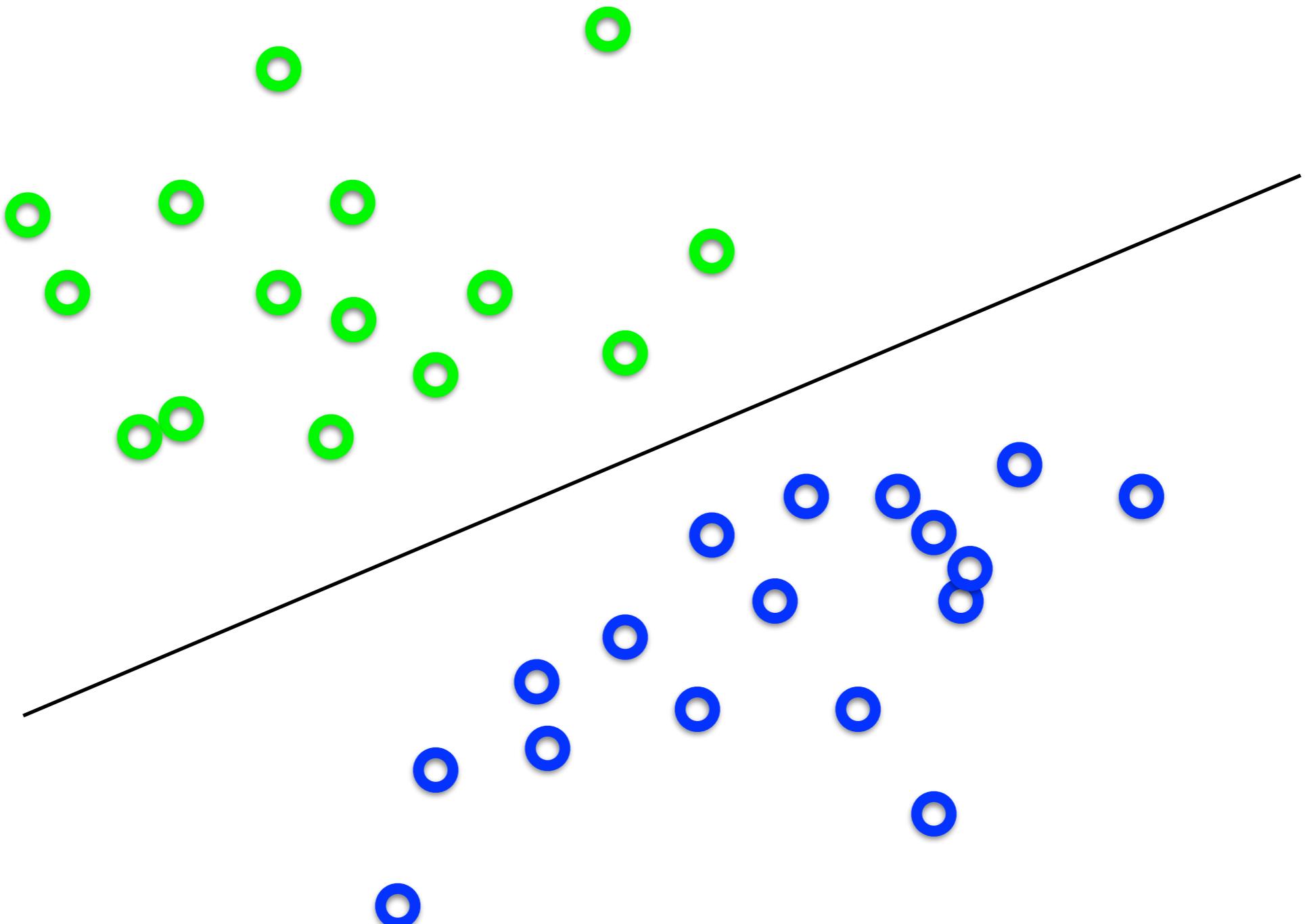
# What's the best $w$ ?



What's the best **w**?

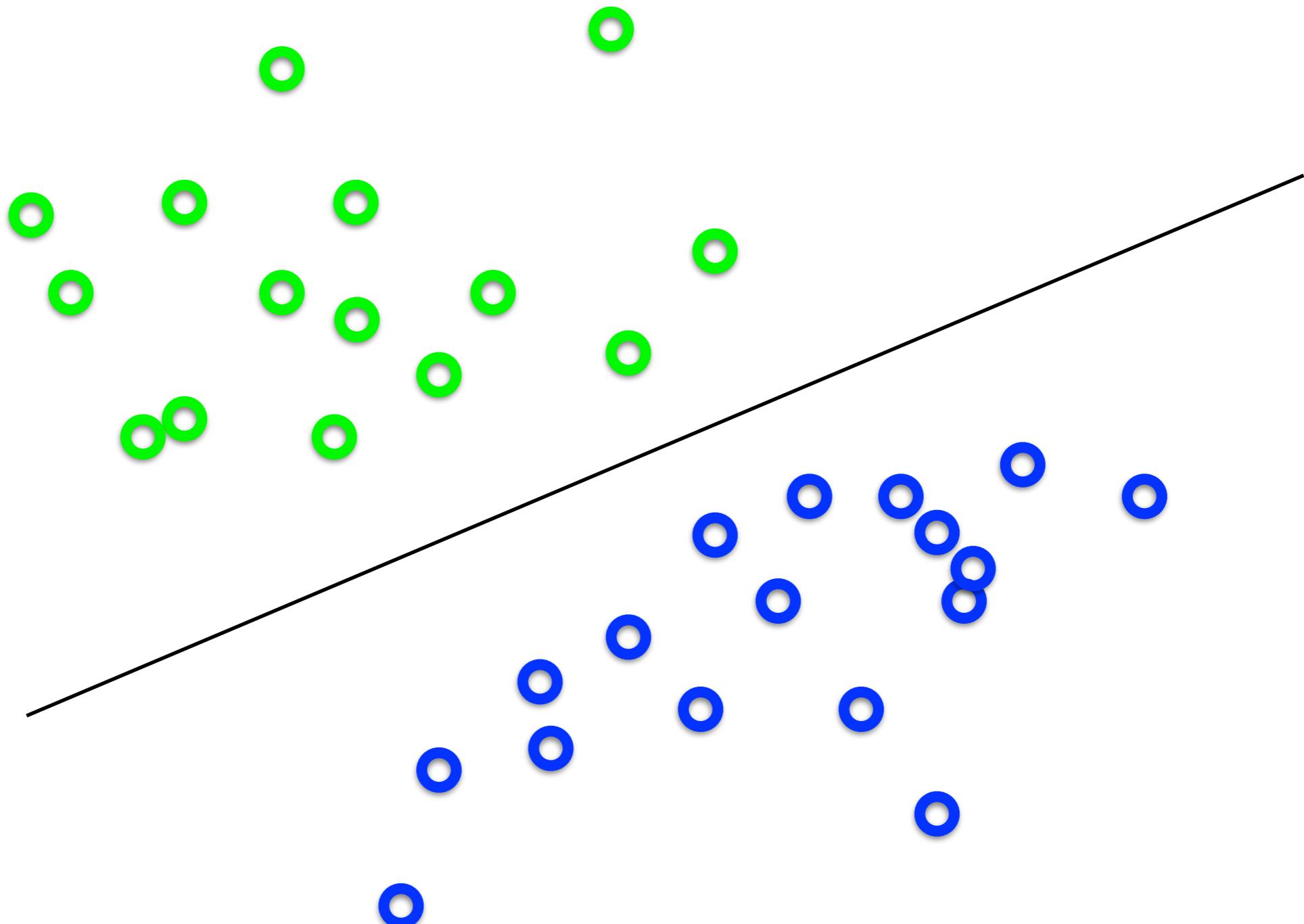


# What's the best $w$ ?



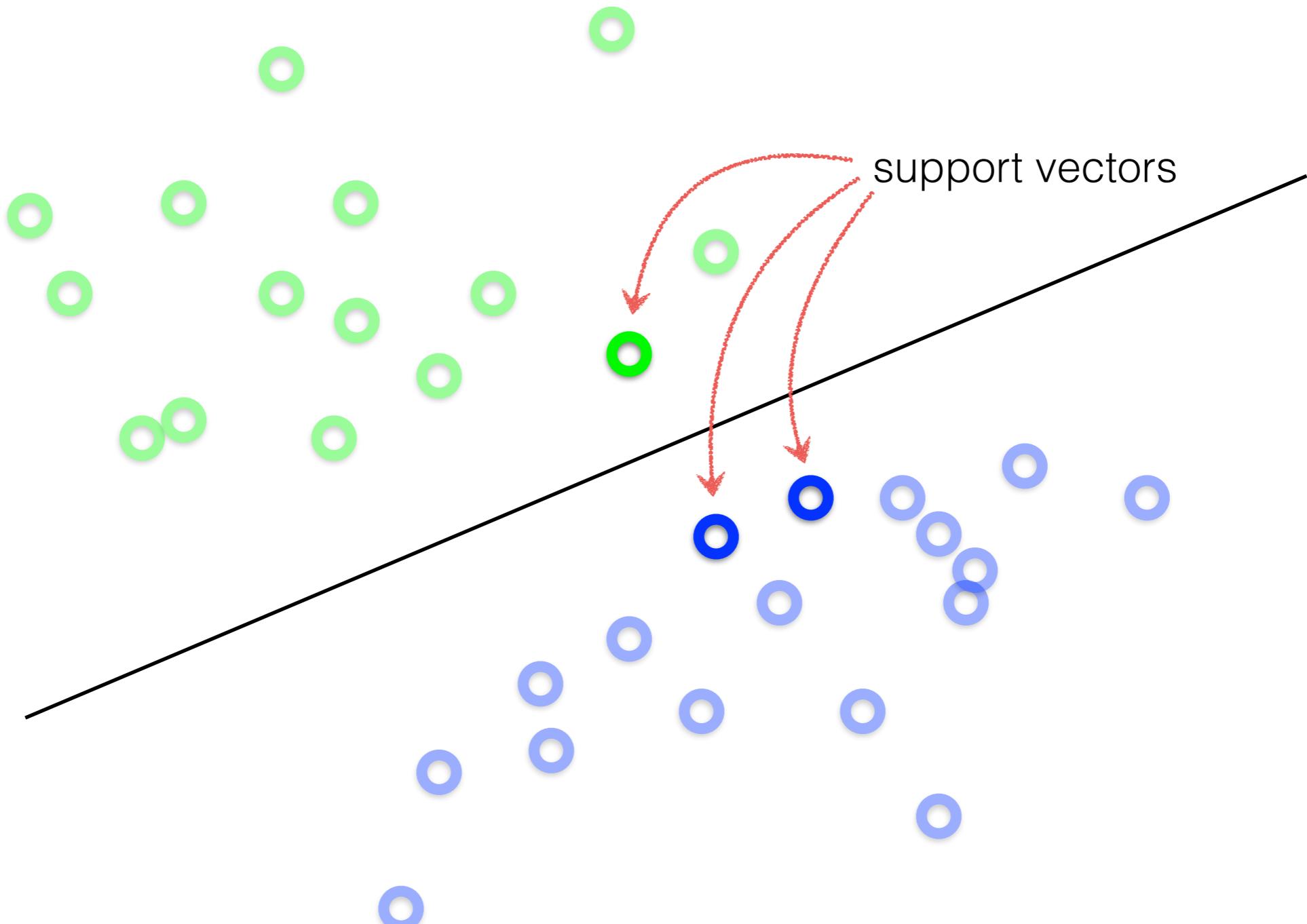
**Intuitively**, the line that is the  
farthest from all interior points

# What's the best $\mathbf{w}$ ?



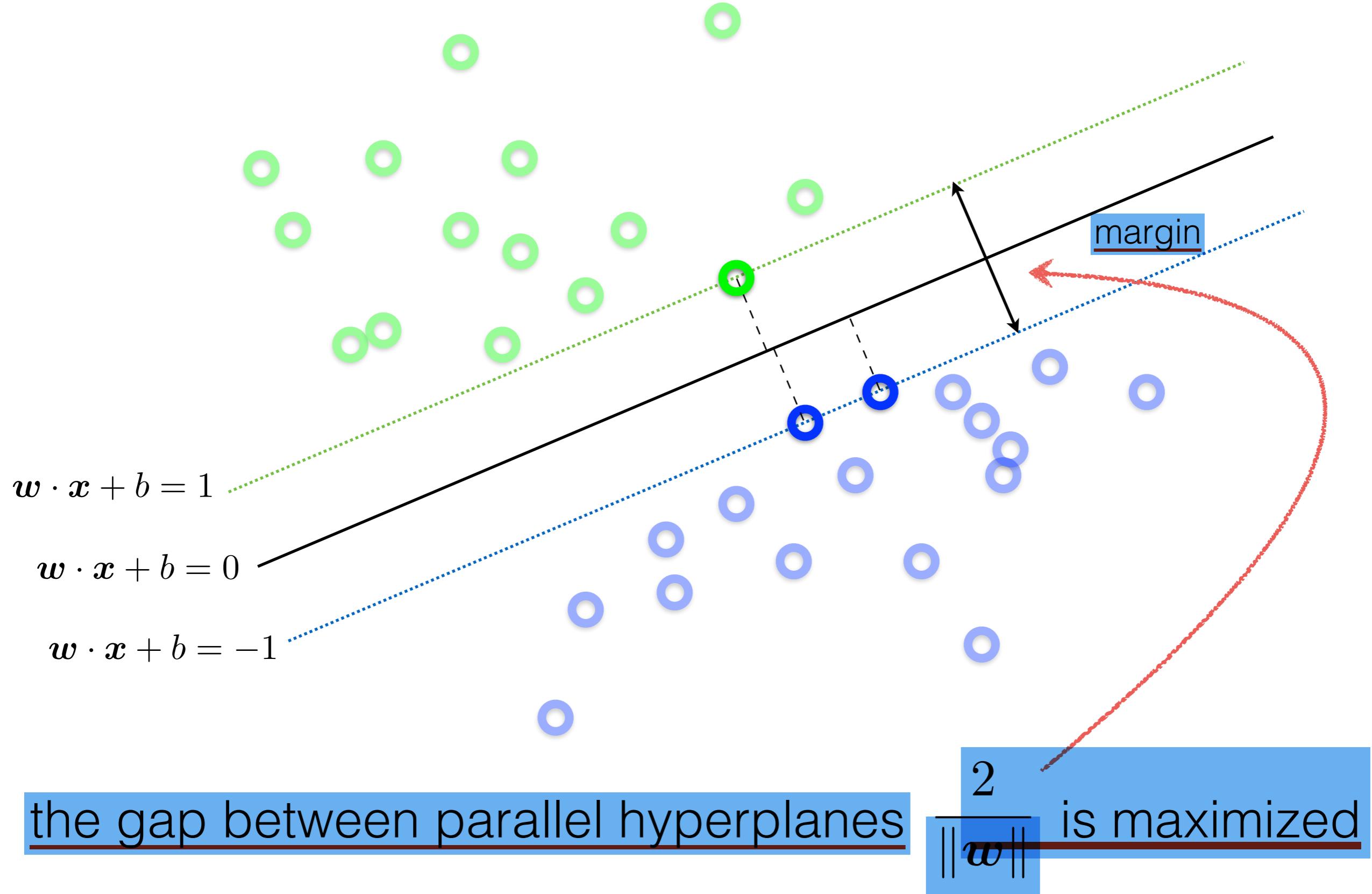
**Maximum Margin solution:**  
most stable to perturbations of data

# What's the best $\mathbf{w}$ ?



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

Find hyperplane  $\mathbf{w}$  such that ...



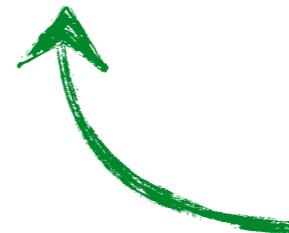
Can be formulated as a maximization problem

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

$$\text{subject to } \mathbf{w} \cdot \mathbf{x}_i + b \begin{cases} \geq +1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases} \text{ for } i = 1, \dots, N$$

*What does this constraint mean?*

Line should lie between these two hyperplanes.



label of the data point

*Why is it +1 and -1?*

Can be formulated as a maximization problem

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

subject to  $\mathbf{w} \cdot \mathbf{x}_i + b \begin{cases} \geq +1 & \text{if } y_i = +1 \\ \leq -1 & \text{if } y_i = -1 \end{cases}$  for  $i = 1, \dots, N$

Equivalently,

*Where did the 2 go?*

$$\min_{\mathbf{w}} \|\mathbf{w}\|$$

subject to  $y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \geq 1$  for  $i = 1, \dots, N$

*What happened to the labels?*

More general form so 2 and the y labels are generalized here

# ‘Primal formulation’ of a linear SVM

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

Objective Function

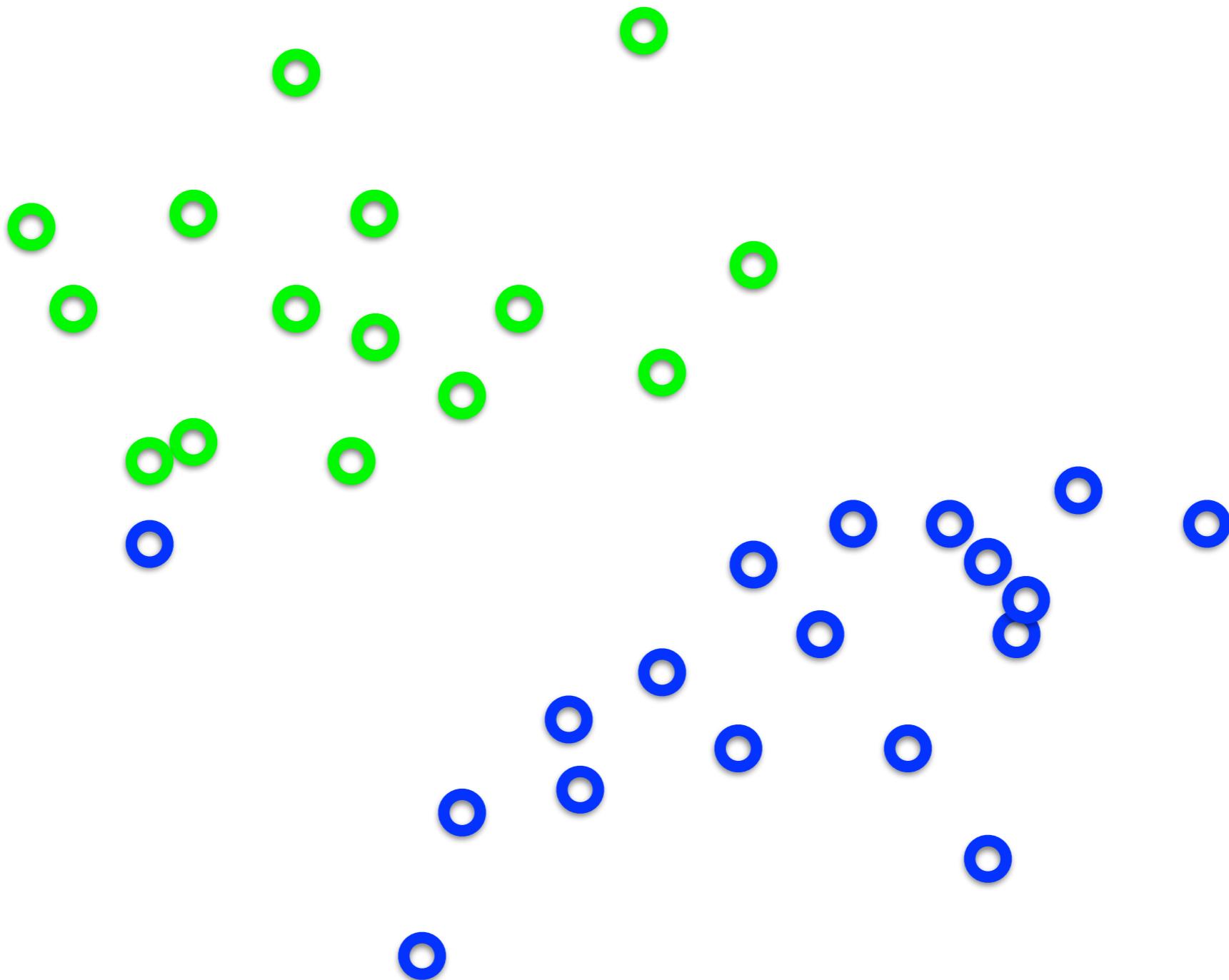
subject to  $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \quad \text{for } i = 1, \dots, N$

Constraints

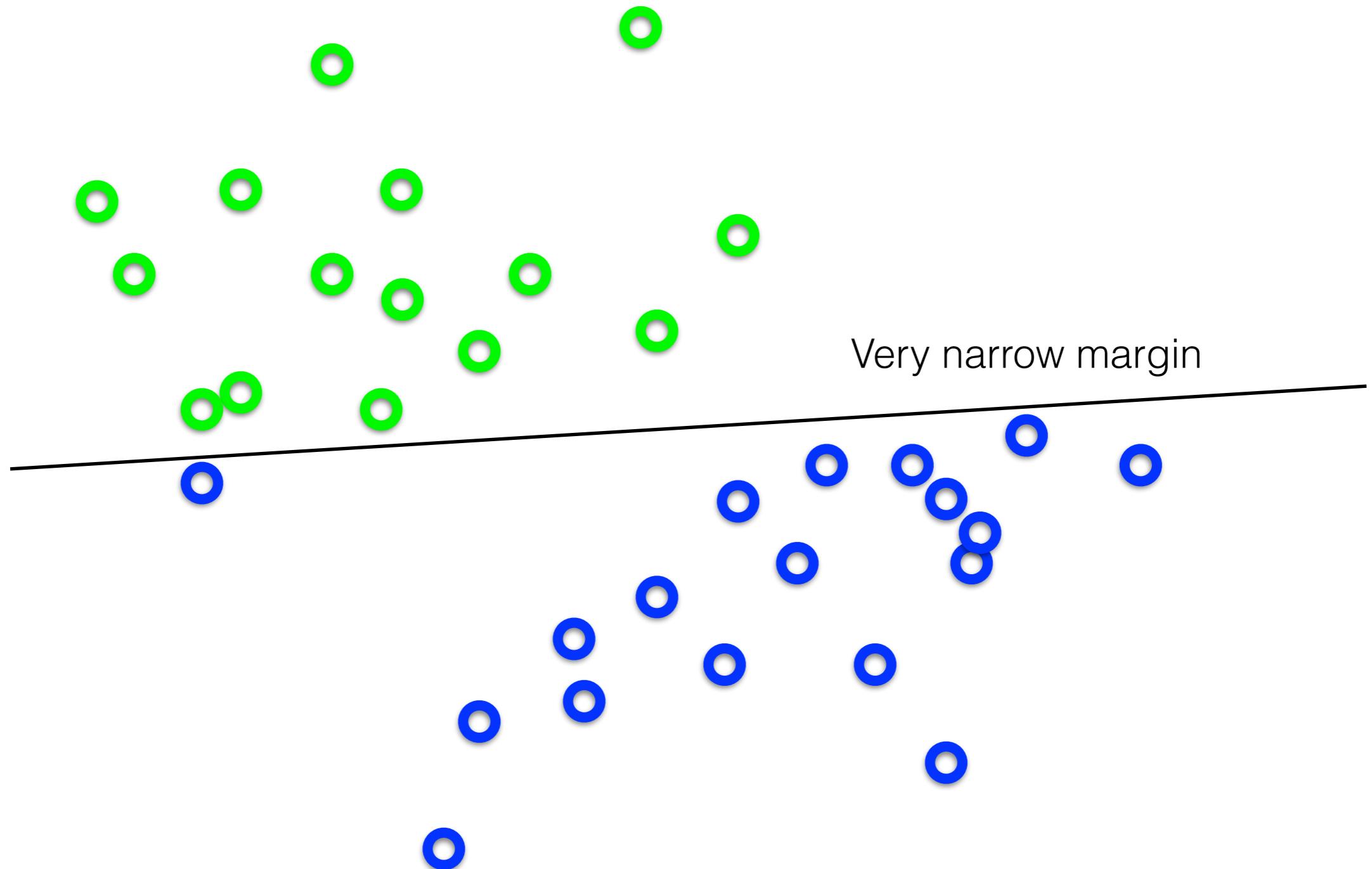
This is a convex quadratic programming (QP) problem  
(a unique solution exists)

‘soft’ margin

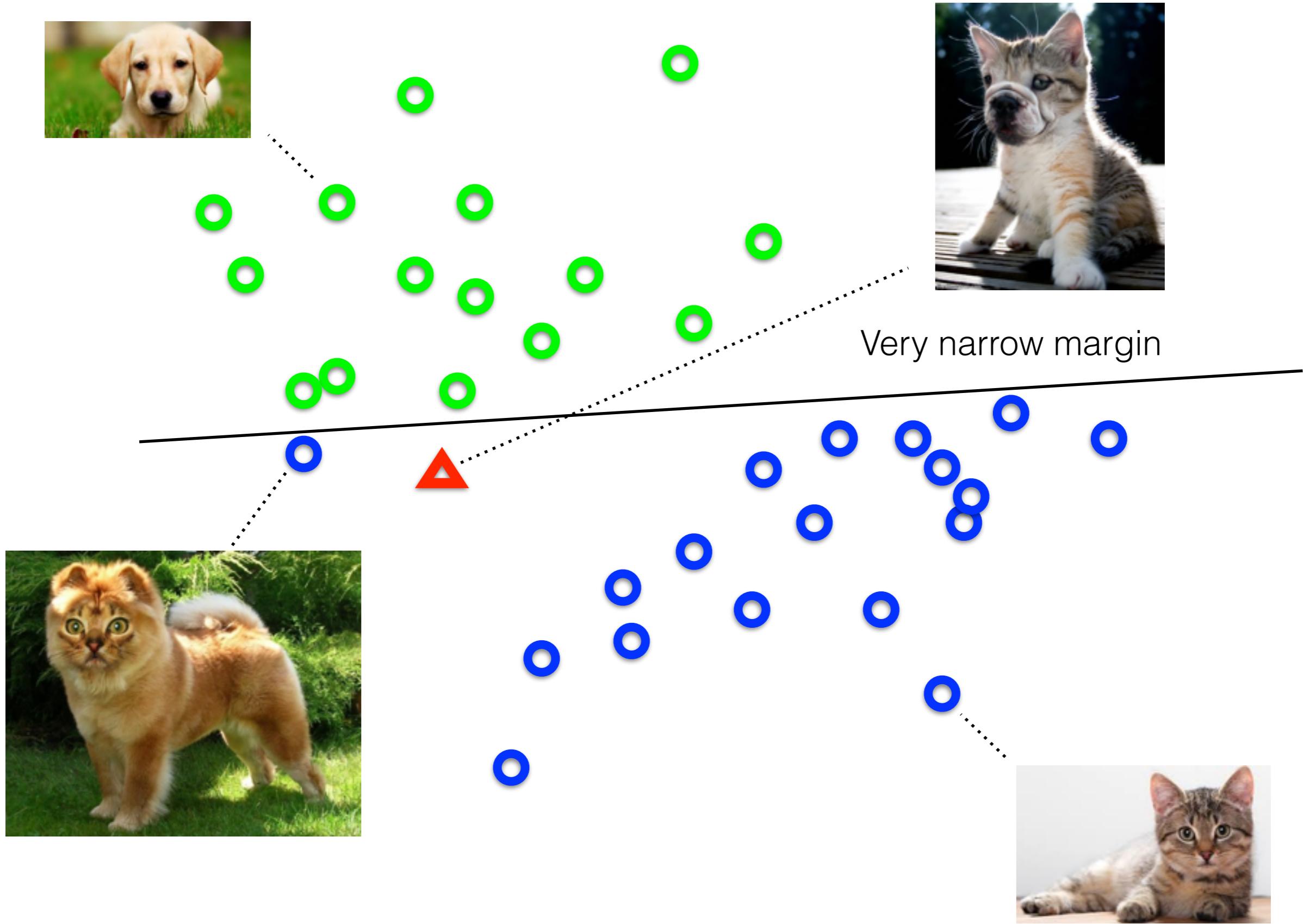
# What's the best **w**?



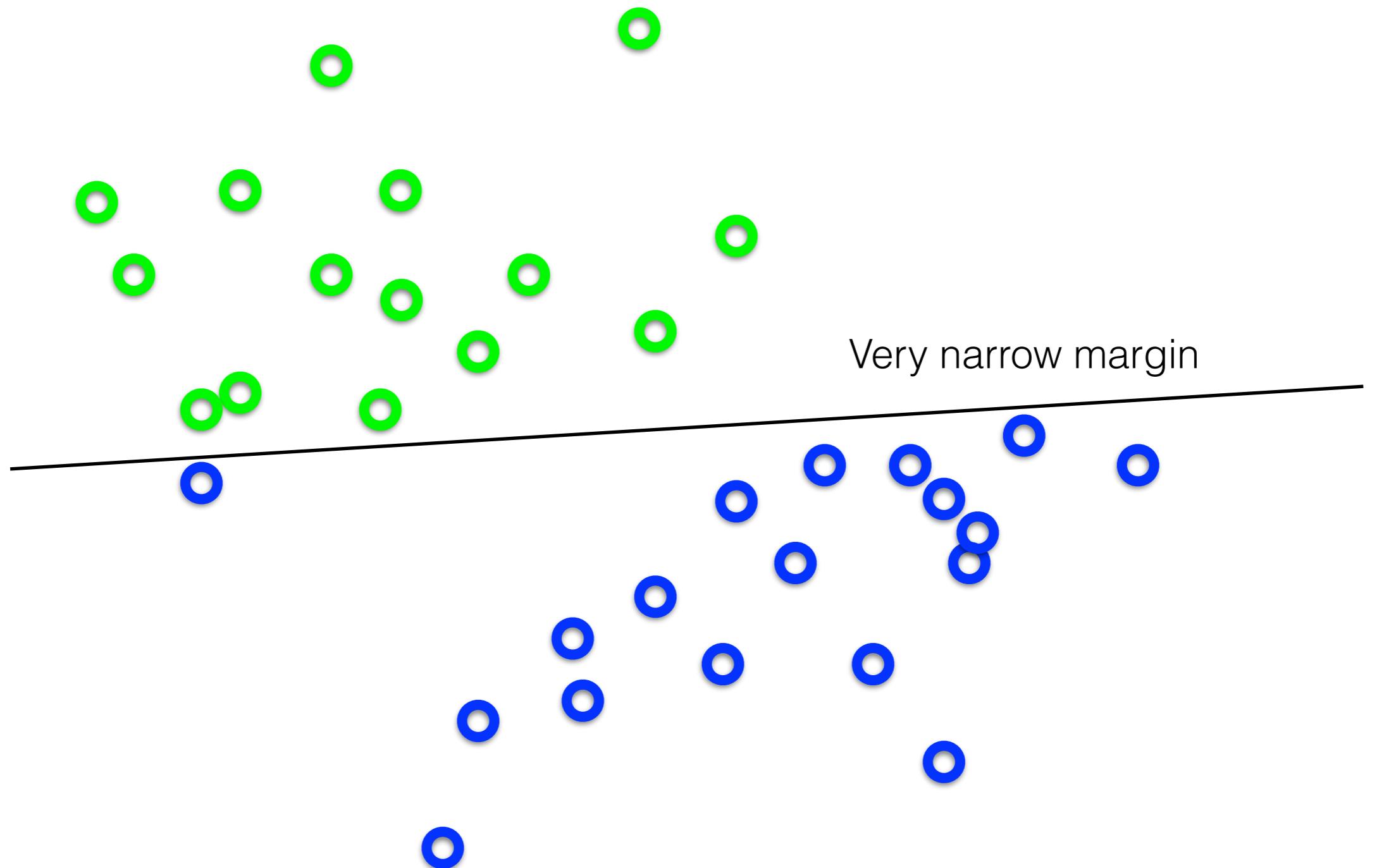
# What's the best $\mathbf{w}$ ?



# Separating cats and dogs

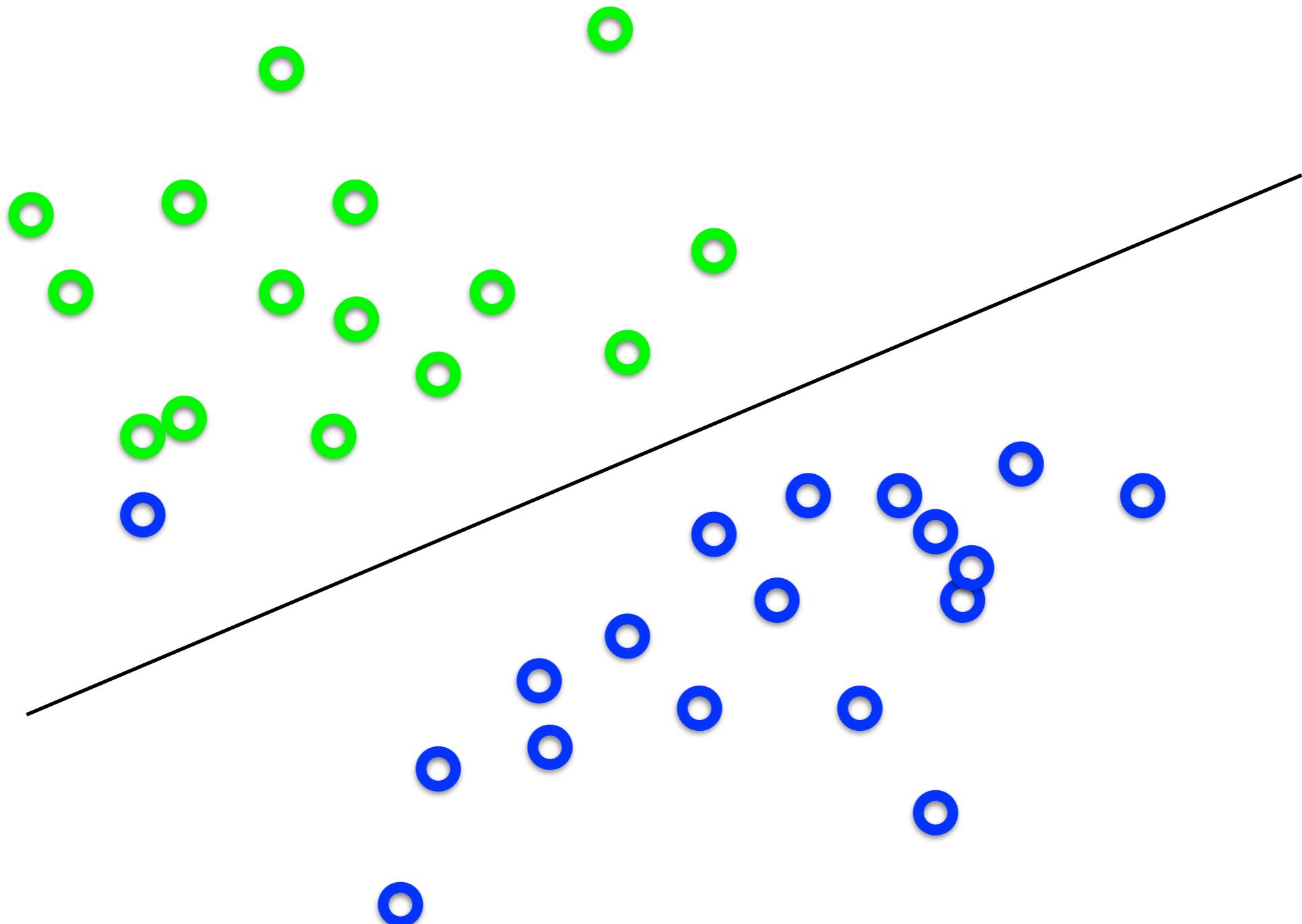


# What's the best $\mathbf{w}$ ?



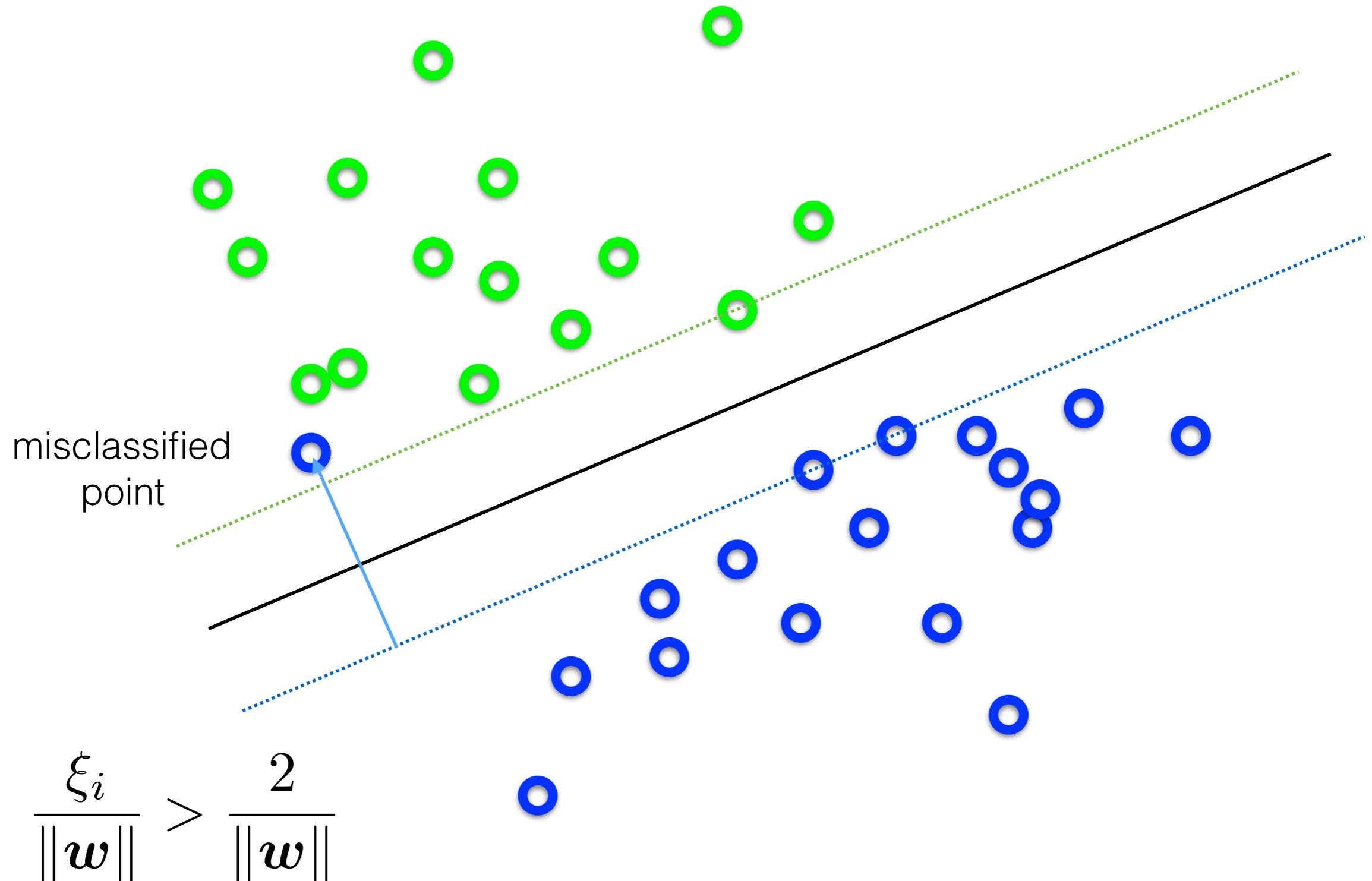
**Intuitively**, we should allow for some misclassification if we can get more robust classification

# What's the best **w**?



Trade-off between the MARGIN and the MISTAKES  
(might be a better solution)

## Adding slack variables $\xi_i \geq 0$



# ‘soft’ margin

objective

$$\min_{\boldsymbol{w}, \boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C \sum_i \xi_i$$

subject to

$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i$$

for  $i = 1, \dots, N$

# 'soft' margin

objective

$$\min_{\mathbf{w}, \xi} \|\mathbf{w}\|^2 + C \sum_i \xi_i$$

subject to

$$y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, N$$

The slack variable allows for mistakes,  
as long as the inverse margin is minimized.

# ‘soft’ margin

objective

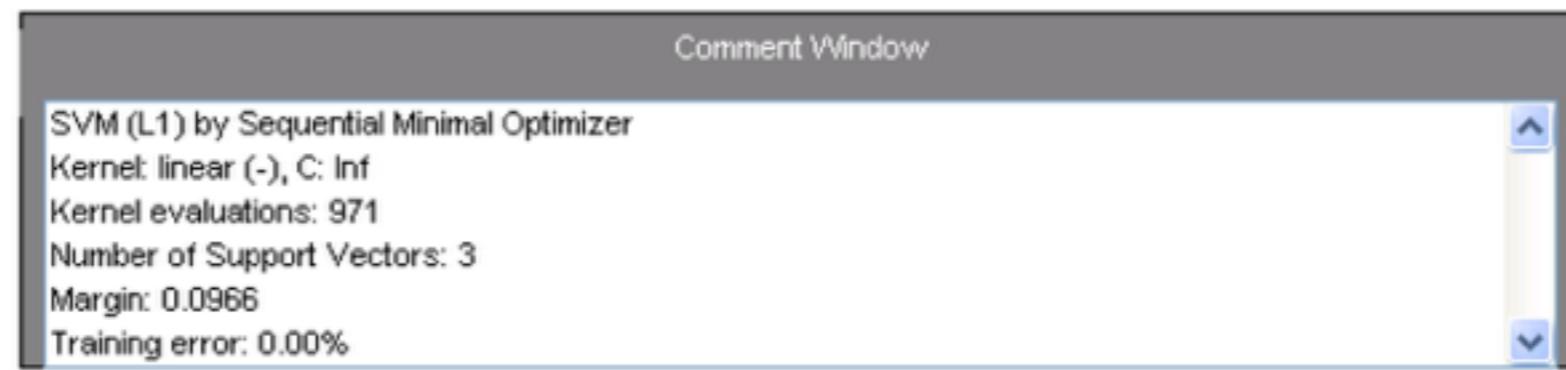
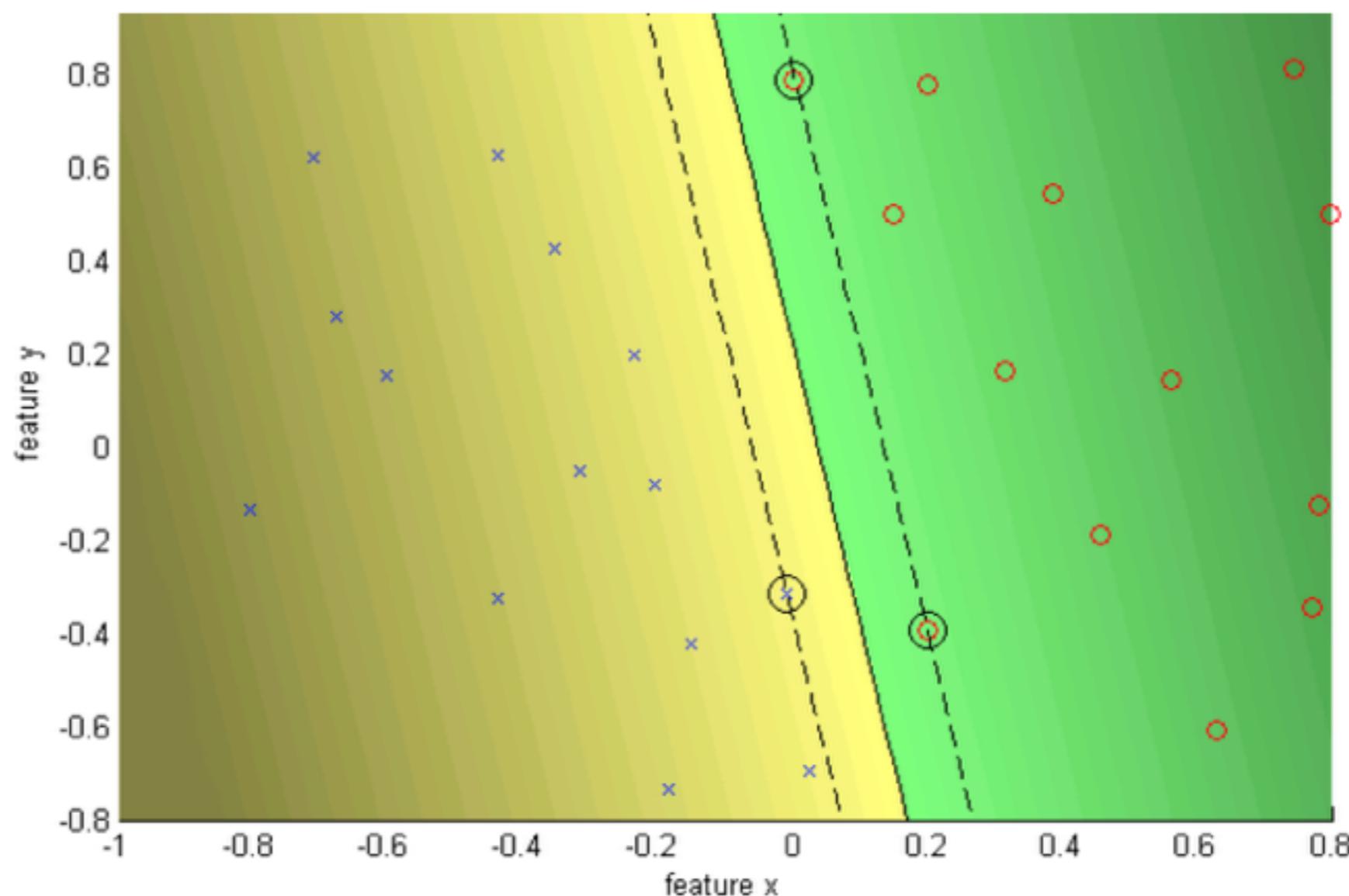
$$\min_{\boldsymbol{w}, \boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C \sum_i \xi_i$$

subject to

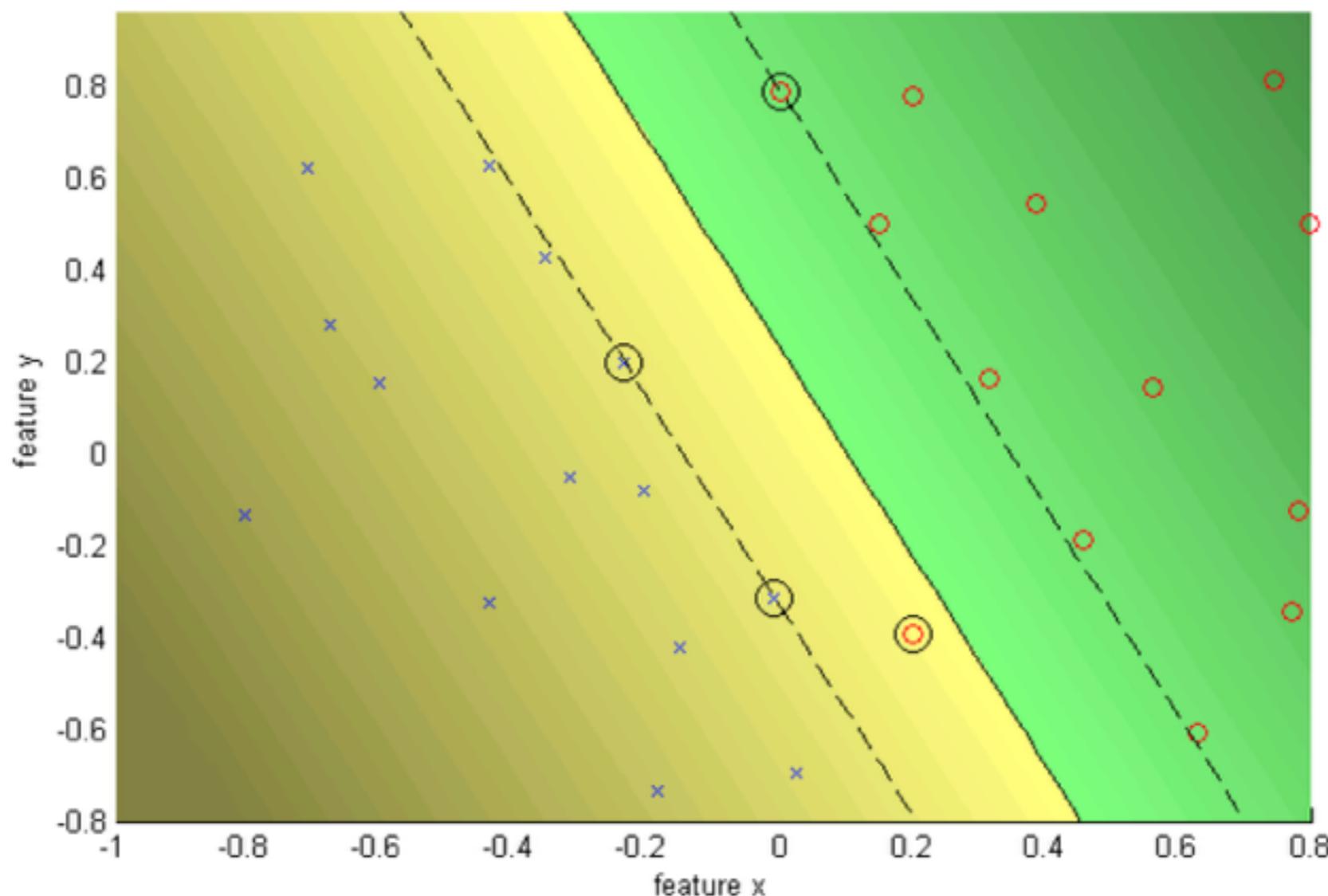
$$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \text{for } i = 1, \dots, N$$

- Every constraint can be satisfied if slack is large
- C is a regularization parameter
  - Small C: ignore constraints (larger margin)
  - Big C: constraints (small margin)
- Still QP problem (unique solution)

## C = Infinity hard margin



## C = 10 soft margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer  
Kernel: linear (-), C: 10.0000  
Kernel evaluations: 2645  
Number of Support Vectors: 4  
Margin: 0.2265  
Training error: 3.70%