Machine Learning

Classification

Overview

Problems:

1. Spam or Non-spam emails
2. Person re-identification (same or different person)

Classifier

- Classifier: A classifier is a special case of a hypothesis (nowadays, often learned by a machine learning algorithm). A classifier is a hypothesis or discrete-valued function that is used to assign (categorical) class labels to particular data points. In the email classification example, this classifier could be a hypothesis for labeling emails as spam or non-spam. However, a hypothesis must not necessarily be synonymous to a classifier. In a different application, our hypothesis could be a function for mapping study time and educational backgrounds of students to their future SAT scores.

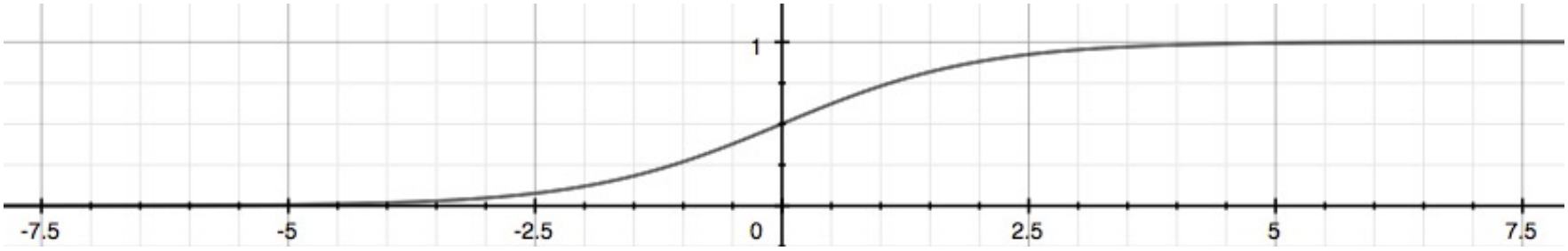
Logistic Regression Hypothesis

$$h_\theta(x) = g(\theta^T x) \tag{1}$$

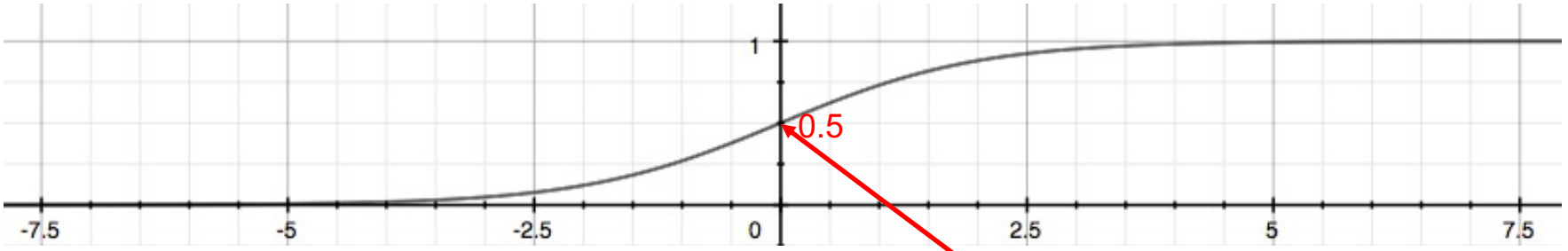
$$g(z) = \frac{1}{1 + e^{-z}} \tag{2}$$

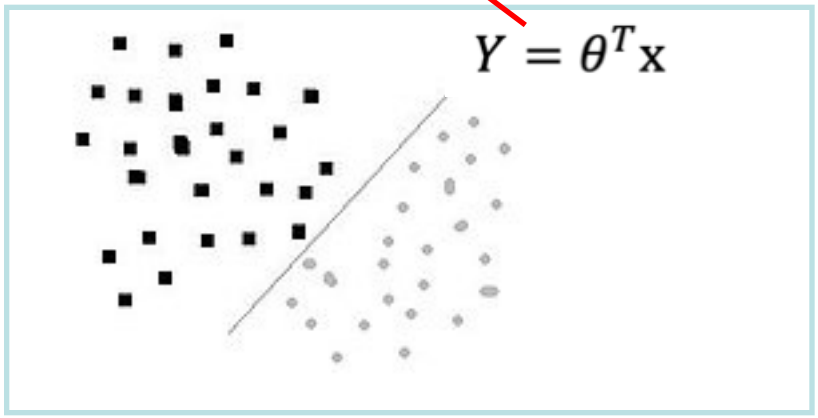
$$z = \theta^T x \tag{3}$$

$$g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \tag{4}$$



Decision Boundary





How to do Classification

- Training data for Classification

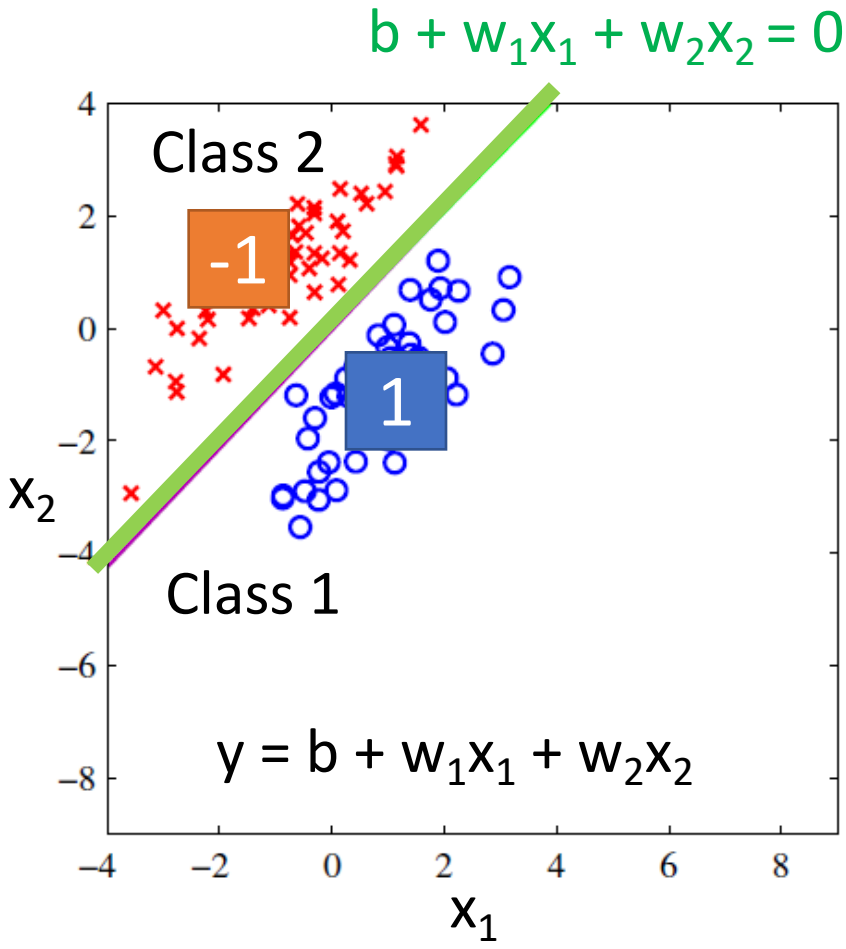
$$(x^1, \hat{y}^1) \qquad (x^2, \hat{y}^2) \qquad \cdots\cdots \qquad (x^N, \hat{y}^N)$$

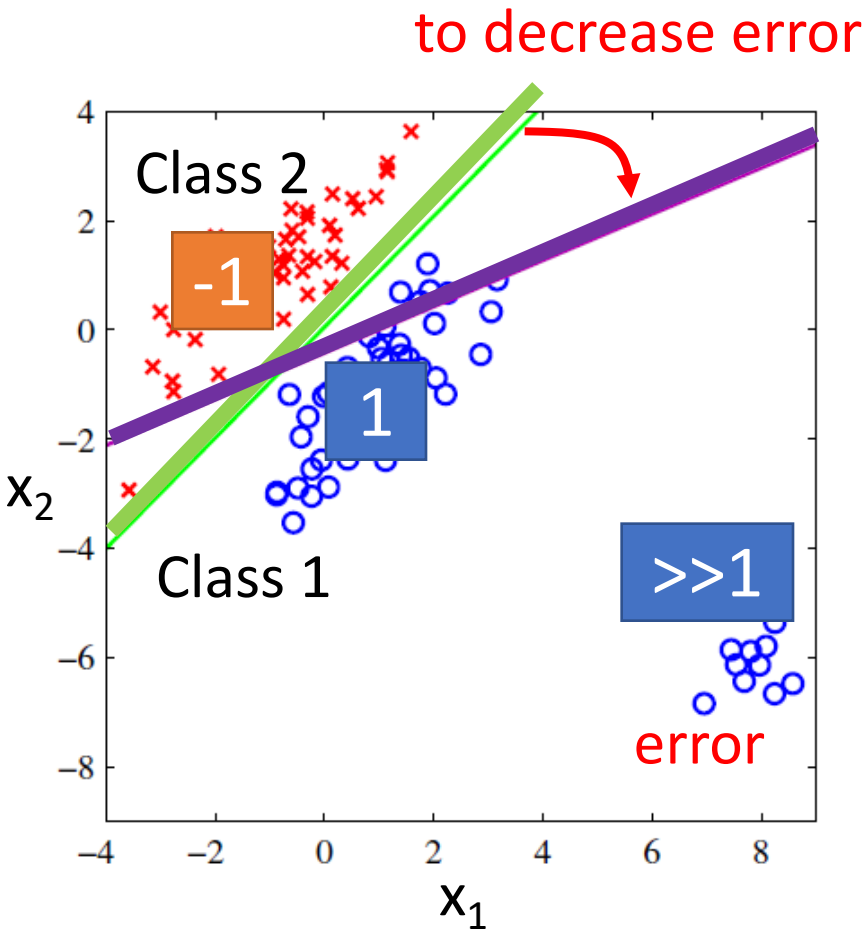
Classification as Regression?

Binary classification as example

Training: Class 1 means the target is 1; Class 2 means the target is -1

Testing: closer to 1 → class 1; closer to -1 → class 2





Penalize to the examples that are "too correct" …

(Bishop, P186)

- Multiple class: Class 1 means the target is 1; Class 2 means the target is 2; Class 3 means the target is 3 …… problematic

Image Classification

(assume given set of discrete labels)
{dog, cat, truck, plane, ...}

cat

Score function

class scores

Linear Classifier

define a **score function**

data (histogram)

$$f(x_i, W, b) = Wx_i + b$$

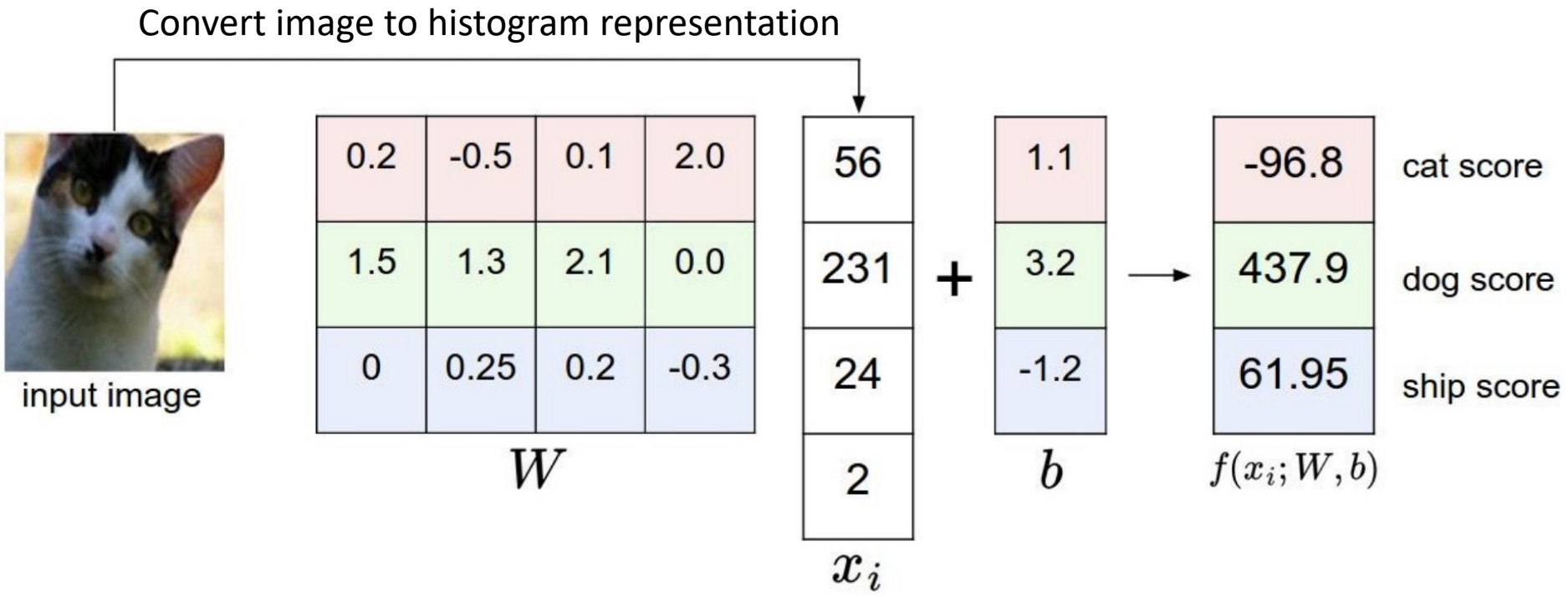
class scores

"weights"

"bias vector"

"parameters"

Example with an image with 4 pixels, and 3 classes (cat/dog/ship)



Machine Learning

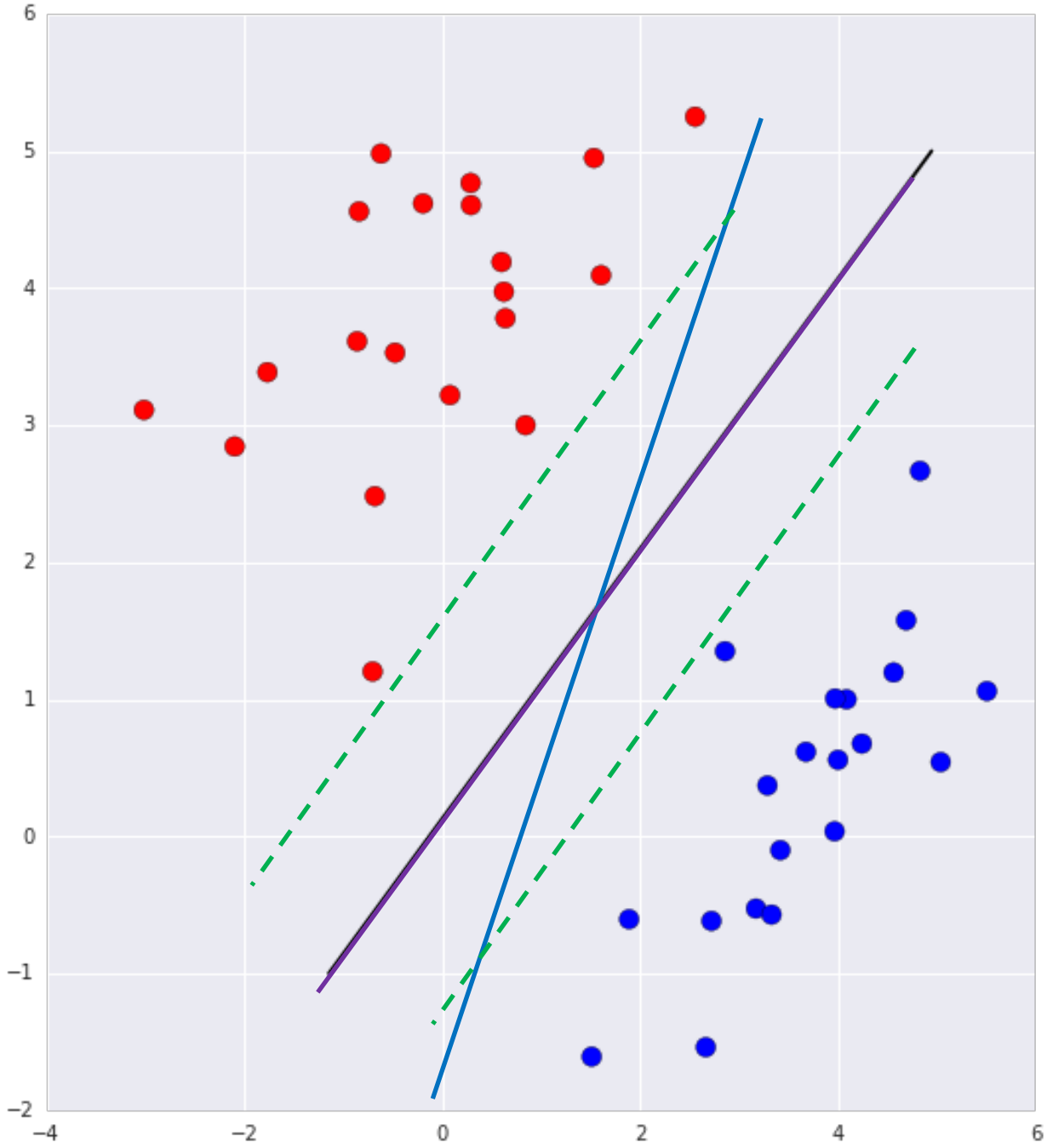
Support Vector Machine

Support Vector Machines (SVMs)

Support vector machines are an optimization based prediction approach used primarily for **binary classification**, and are able to achieve state-of-the-art prediction accuracy on many real-world tasks.

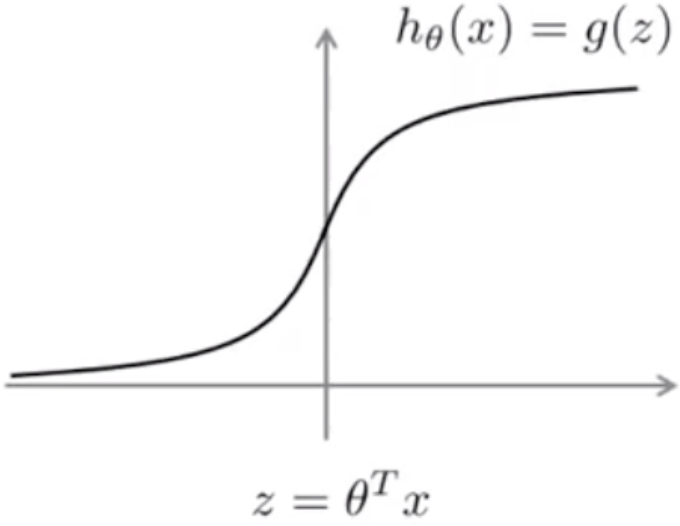
Key idea 1: Learn a **decision boundary** that optimally separates positive and negative training examples. (But what does it mean to be optimal?)

Key idea 2: Learn a **linear** decision boundary in high dimensional space corresponding to a **non-linear** decision boundary for the original problem.


6
5
4
3
2
1
0
−1
−2
−4
−2
0
2
4
6

From Logistic Regression to SVM

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}}$$



If y =1, we wish our predicted hypothesis value is close to 1, then $\theta^T x \gg 0$

If y =0, we wish our predicted hypothesis value is close to 1, then $\theta^T x \ll 0$

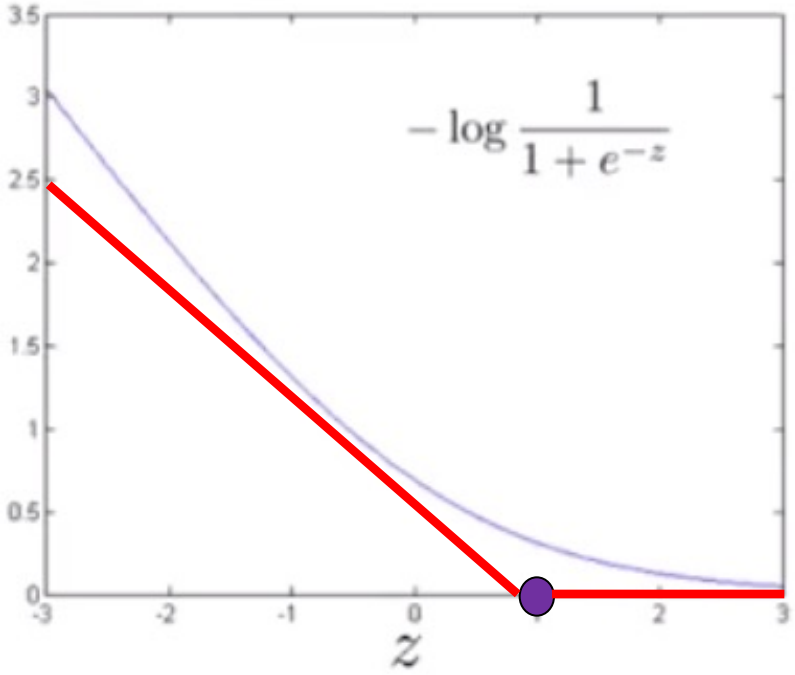
Cost function of Logistic Regression

Cost Function:

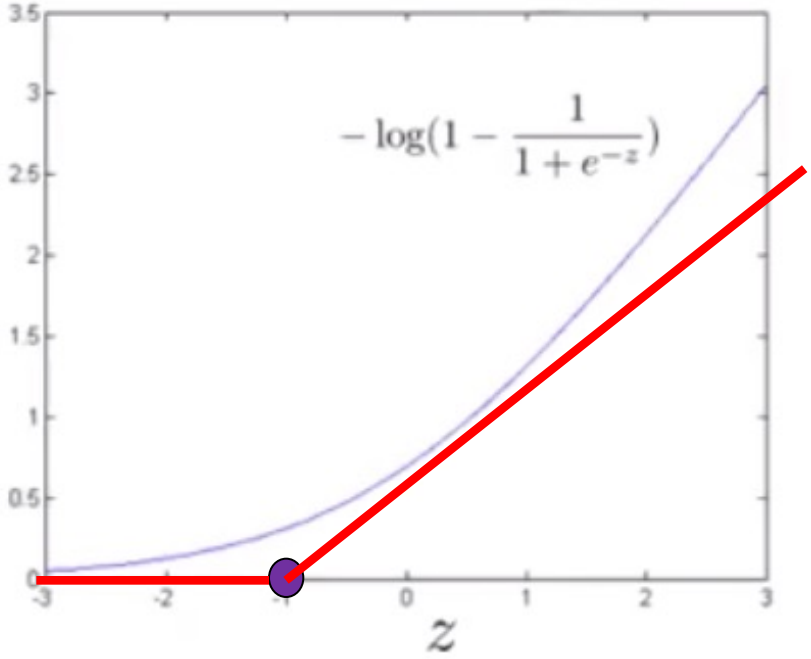
$$-(y \log h_\theta(x) - (1-y) \log(1 - h_\theta(x)))$$

$$= -y \log \frac{1}{1+e^{-\theta^T x}} - (1-y) \log(1 - \frac{1}{1+e^{-\theta^T x}})$$

y = 1

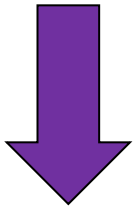

$-\log \frac{1}{1+e^{-z}}$
3.5
3
2.5
2
1.5
1
0.5
0
-3
-2
-1
0
2
3
z

y = 0


$-\log(1-\frac{1}{1+e^{-z}})$
3.5
3
2.5
2
1.5
1
0.5
0
-3
-2
0
1
2
3
z

Logistic regression:

$$\min_{\theta} \frac{1}{m}\left[\sum_{i=1}^{m} y^{(i)}\left(-\log h_\theta(x^{(i)})\right) + (1-y^{(i)})\left(\left(-\log(1-h_\theta(x^{(i)}))\right)\right]+\frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$



Support vector machine:

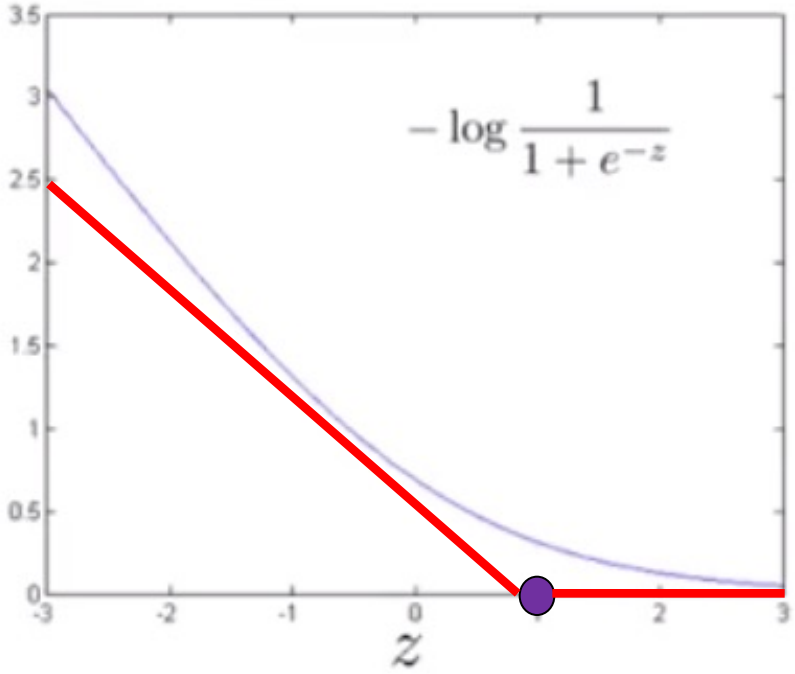
$$\min_{\theta} C\sum_{i=1}^{m}\left[y^{(i)}cost_1(\theta^T x^{(i)}) + (1-y^{(i)})cost_0(\theta^T x^{(i)})\right] + \frac{1}{2}\sum_{i=1}^{n}\theta_j^2$$

SVM Hypothesis

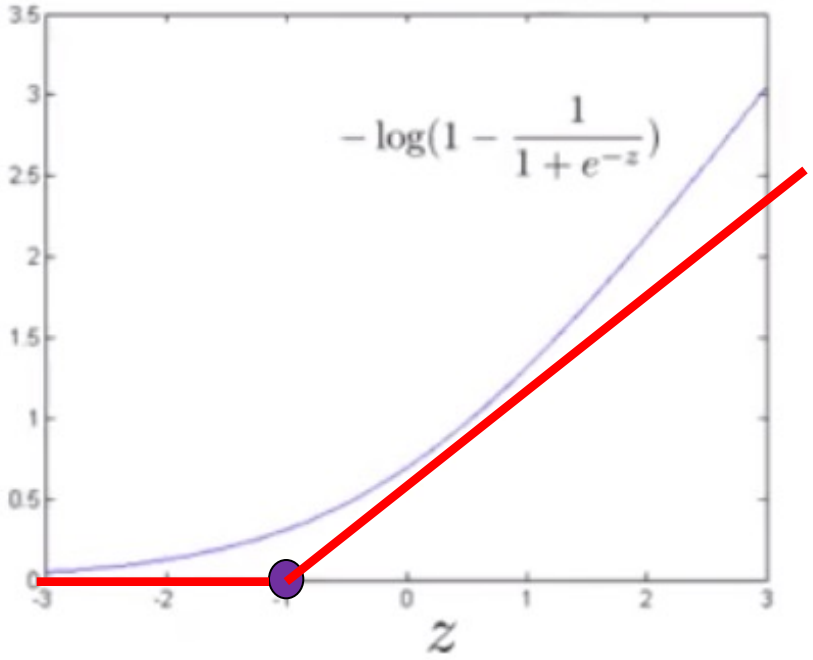
Support Vector Machine

$$\min_{\theta} C \sum_{i=1}^{m} \left[y^{(i)} cost_1(\theta^T x^{(i)}) + (1 - y^{(i)}) cost_0(\theta^T x^{(i)}) \right] + \frac{1}{2} \sum_{i=1}^{n} \theta_j^2$$

y = 1



y = 0



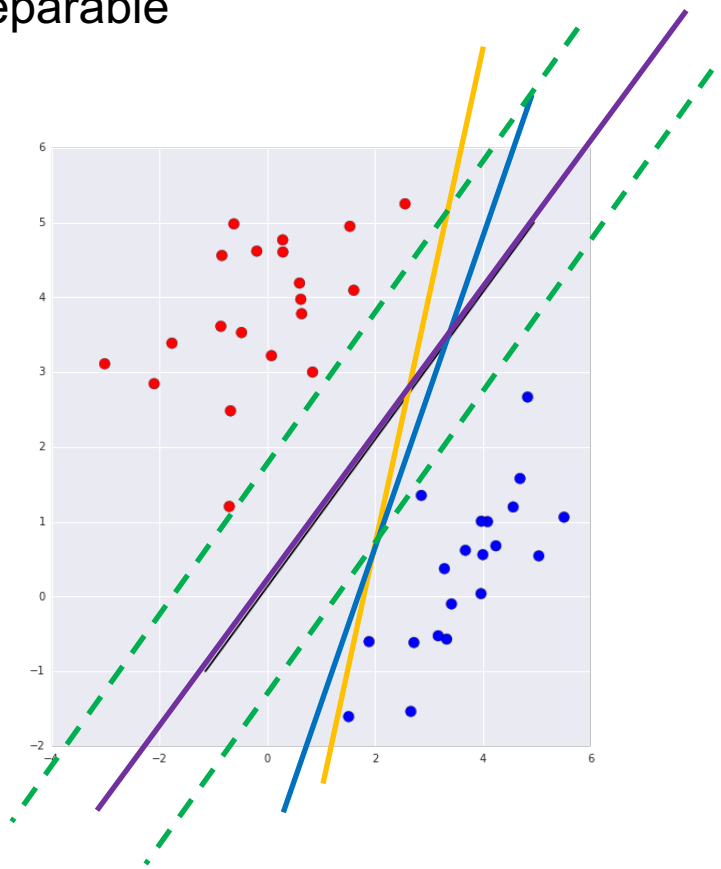
SVM Cost Function

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^T x^{(i)} \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$\theta^T x^{(i)} \leq -1 \qquad \text{if } y^{(i)} = 0$$

Example of linearly separable



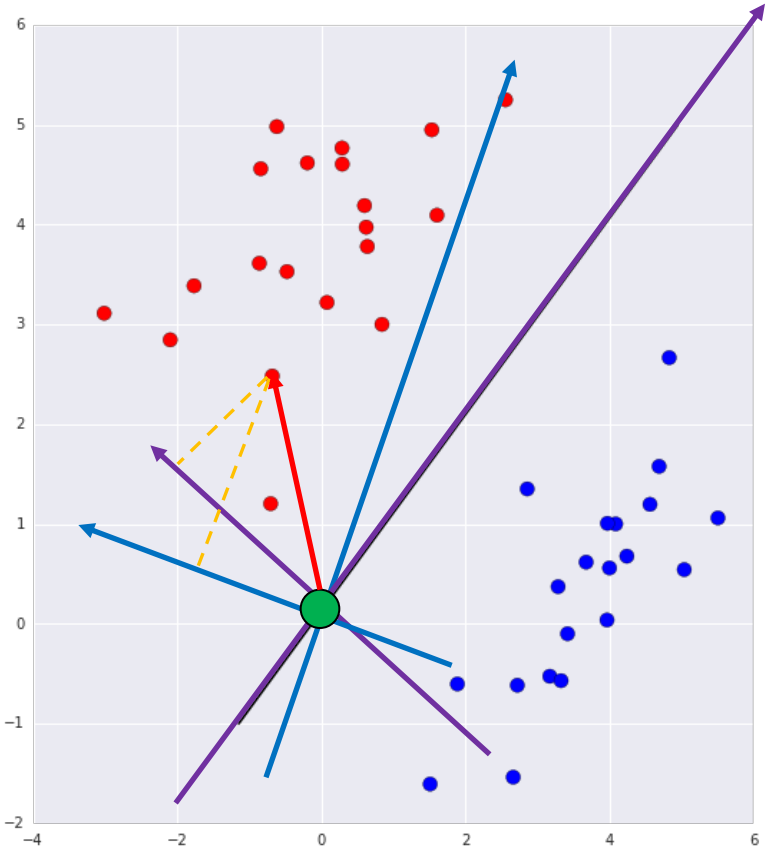
Large Margin Classifier

Why large margin works?

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^T x^{(i)} \geq 1 \qquad \text{if } y^{(i)} = 1$$

$$\theta^T x^{(i)} \leq -1 \qquad \text{if } y^{(i)} = 0$$



Recall Dot product

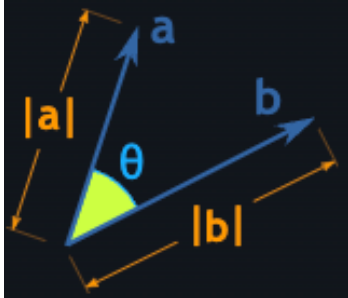
Calculating

The Dot Product is written using a central dot:

$$\mathbf{a} \cdot \mathbf{b}$$

This means the Dot Product of **a** and **b**

We can calculate the Dot Product of two vectors this way:



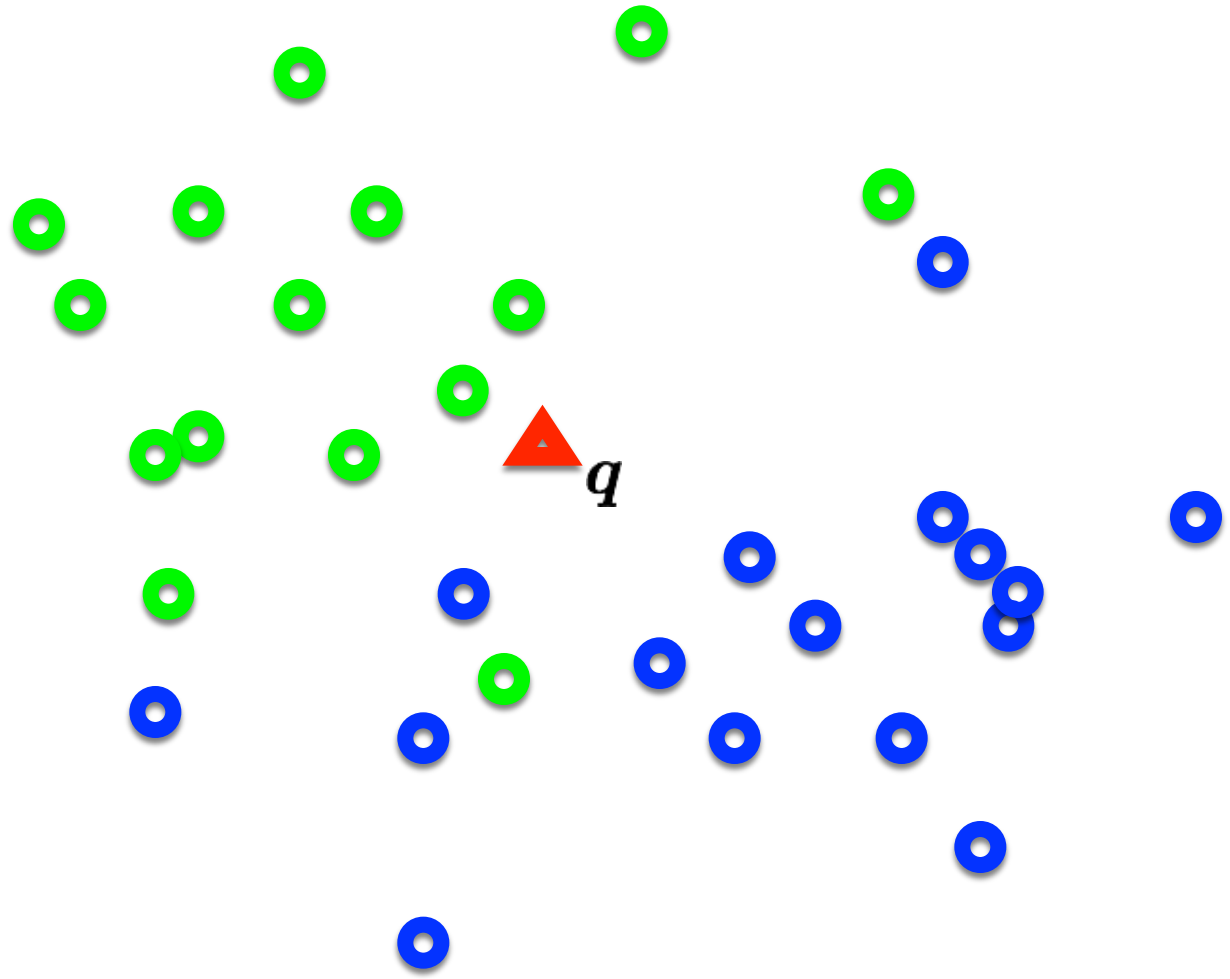
$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}| \times |\mathbf{b}| \times \cos(\theta)$$

Where:
|a| is the magnitude (length) of vector **a**
|b| is the magnitude (length) of vector **b**
θ is the angle between **a** and **b**

Ref to: https://www.mathsisfun.com/algebra/vectors-dot-product.html

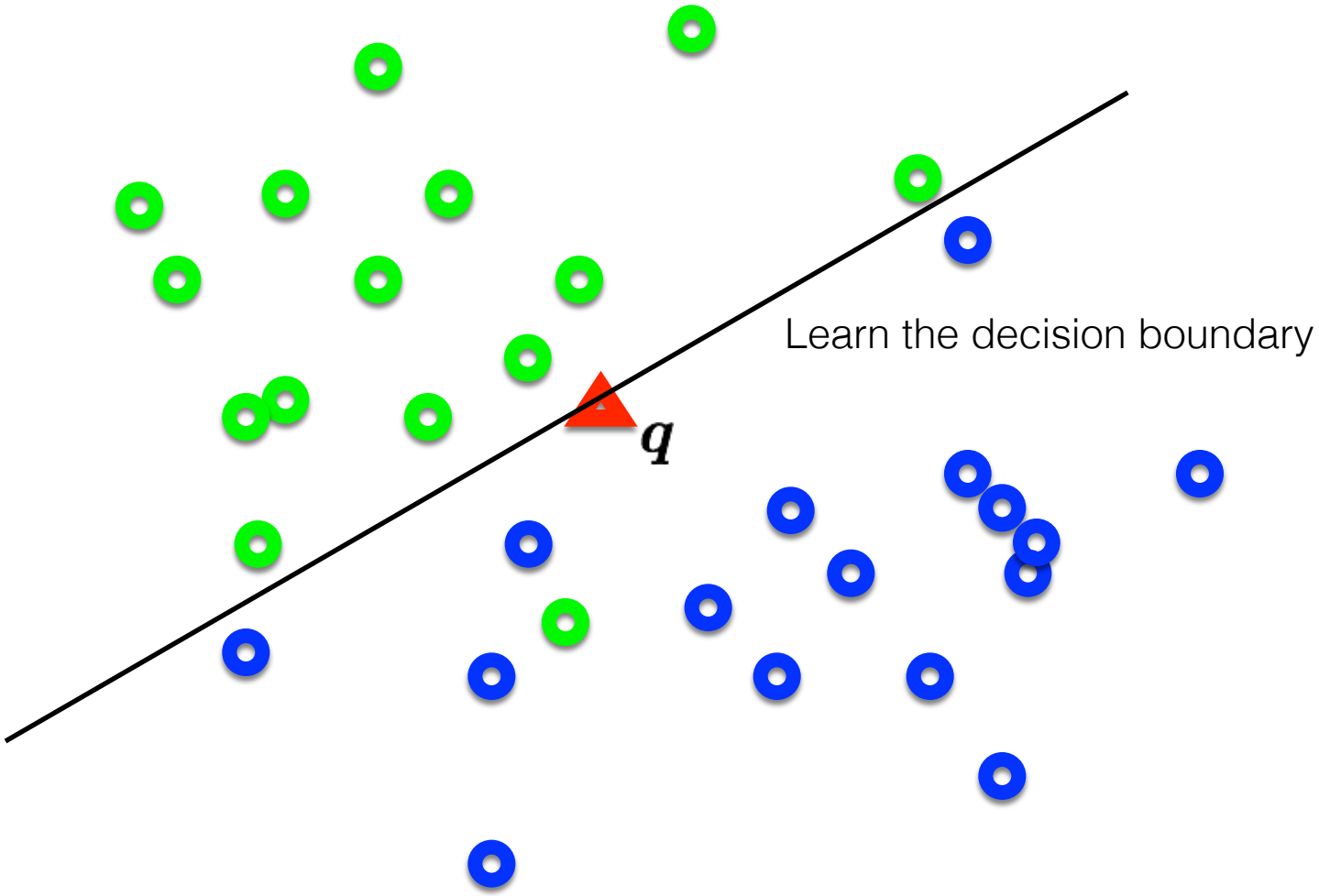
How to interpret SVM cost function?

Distribution of data from two classes



Which class does q belong too?

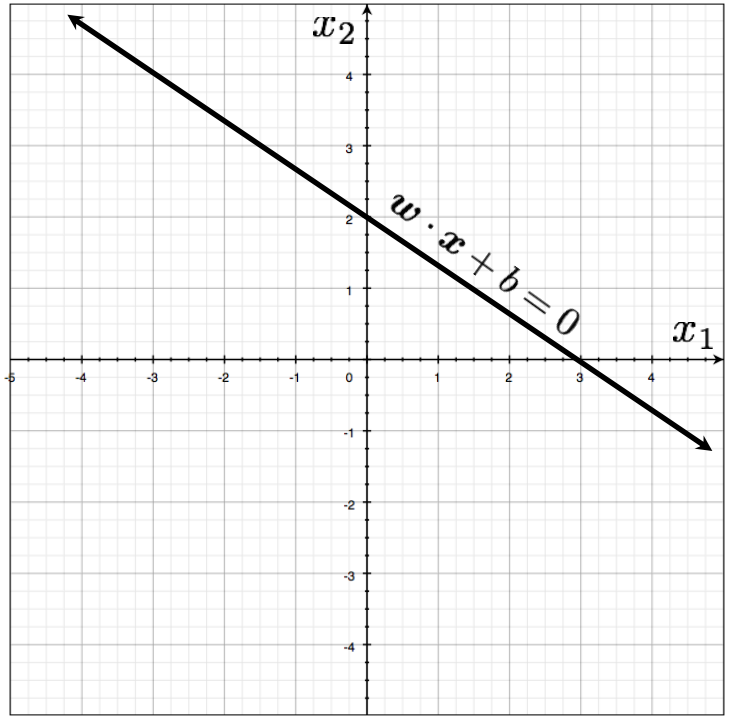
Distribution of data from two classes



First we need to understand hyperplanes…

Hyperplanes (lines) in 2D

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



a line can be written as
dot product plus a bias

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

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

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

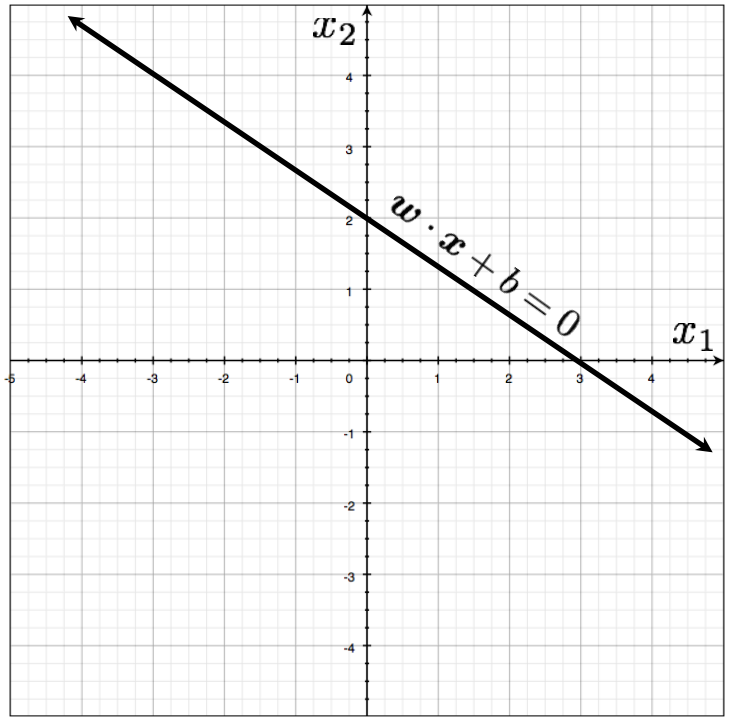
$$\boldsymbol{w} \cdot \boldsymbol{x} = 0$$

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

Hyperplanes (lines) in 2D

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

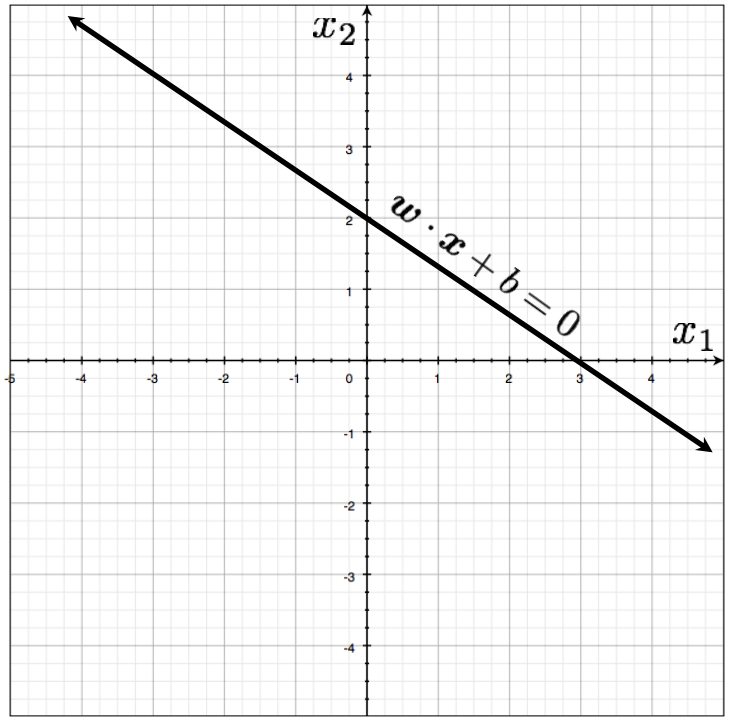
$$w_1 x_1 + w_2 x_2 + b = 0$$



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

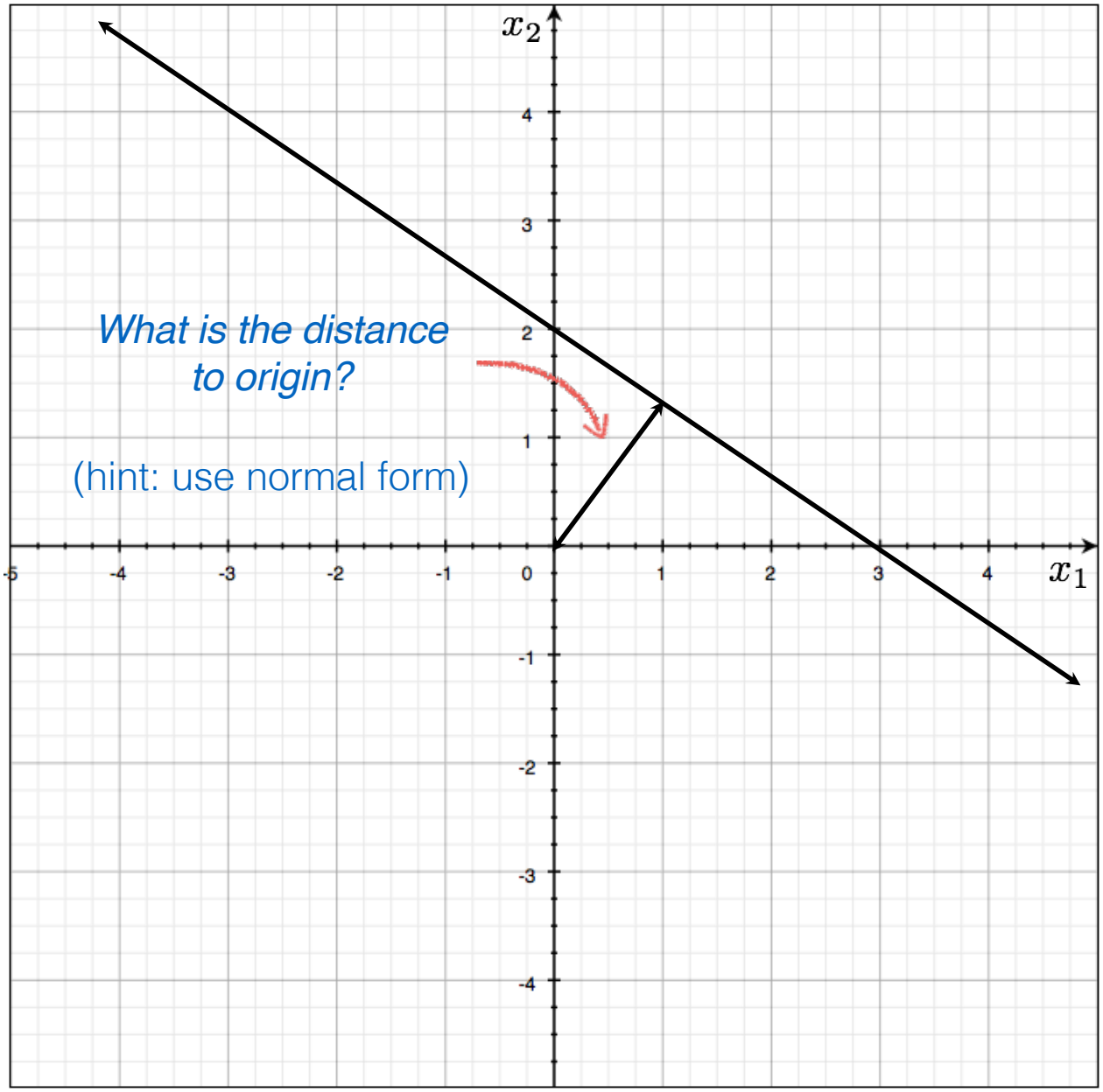
The line

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

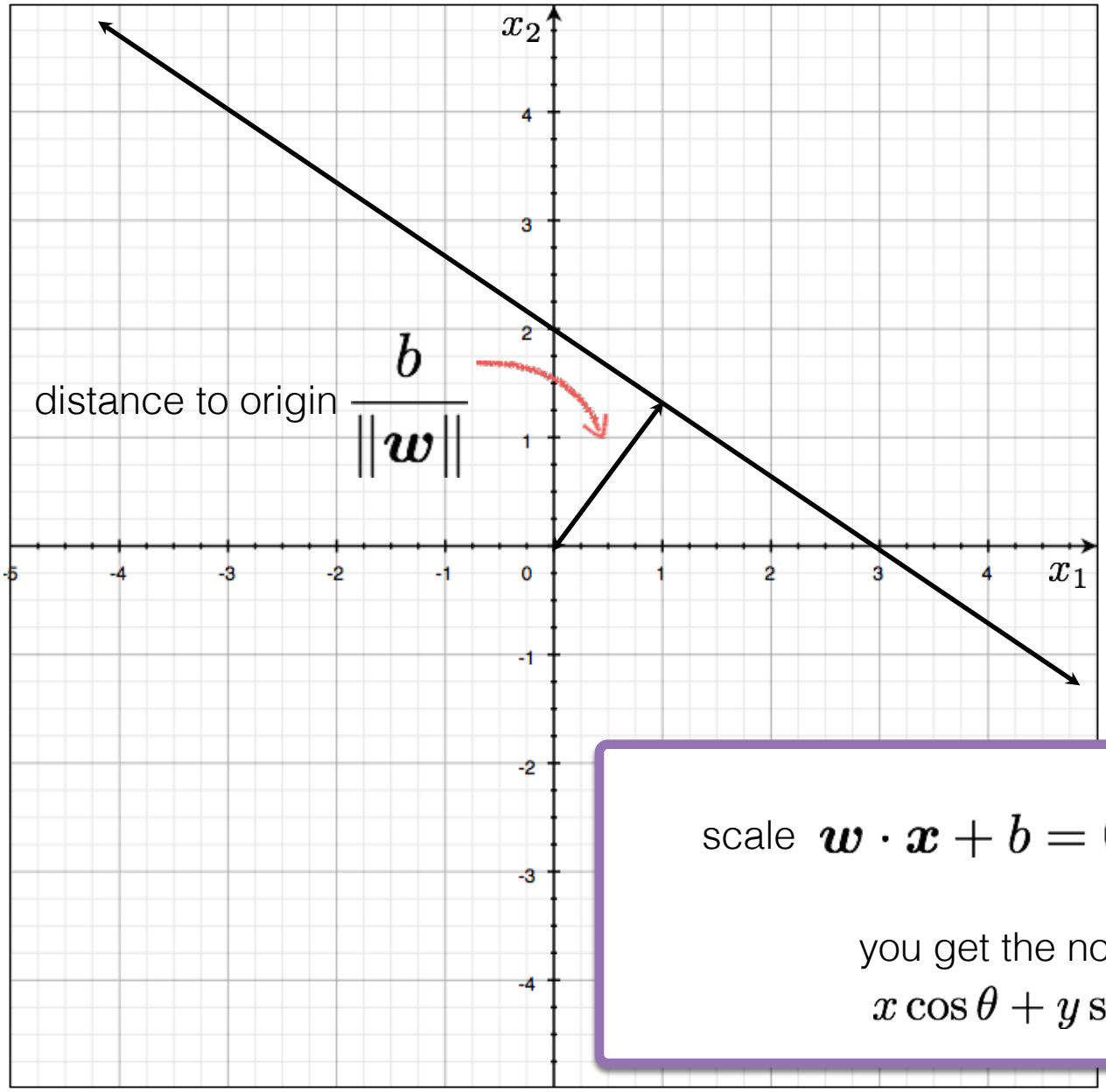
and the line

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

define the same line



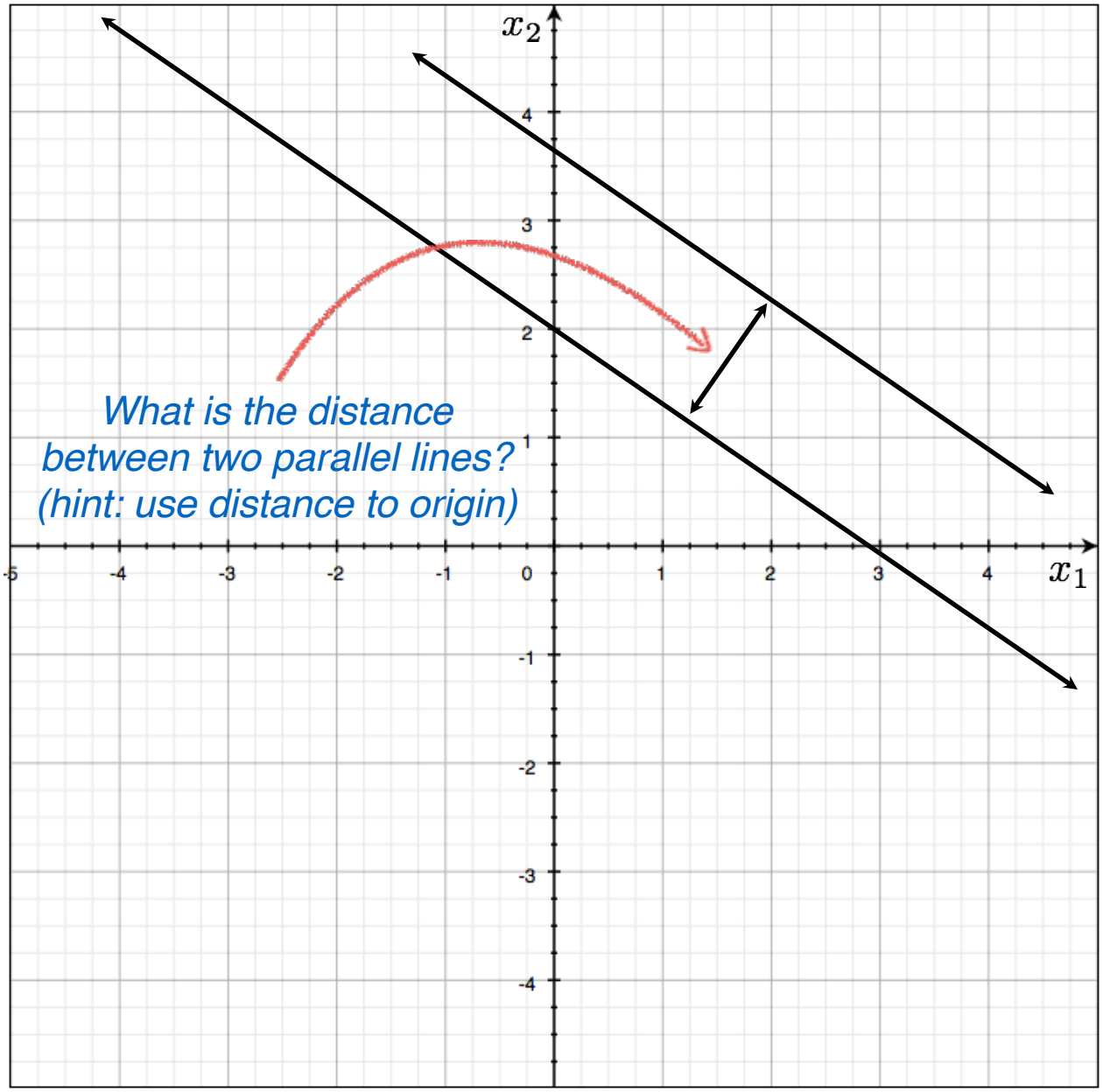
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$



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

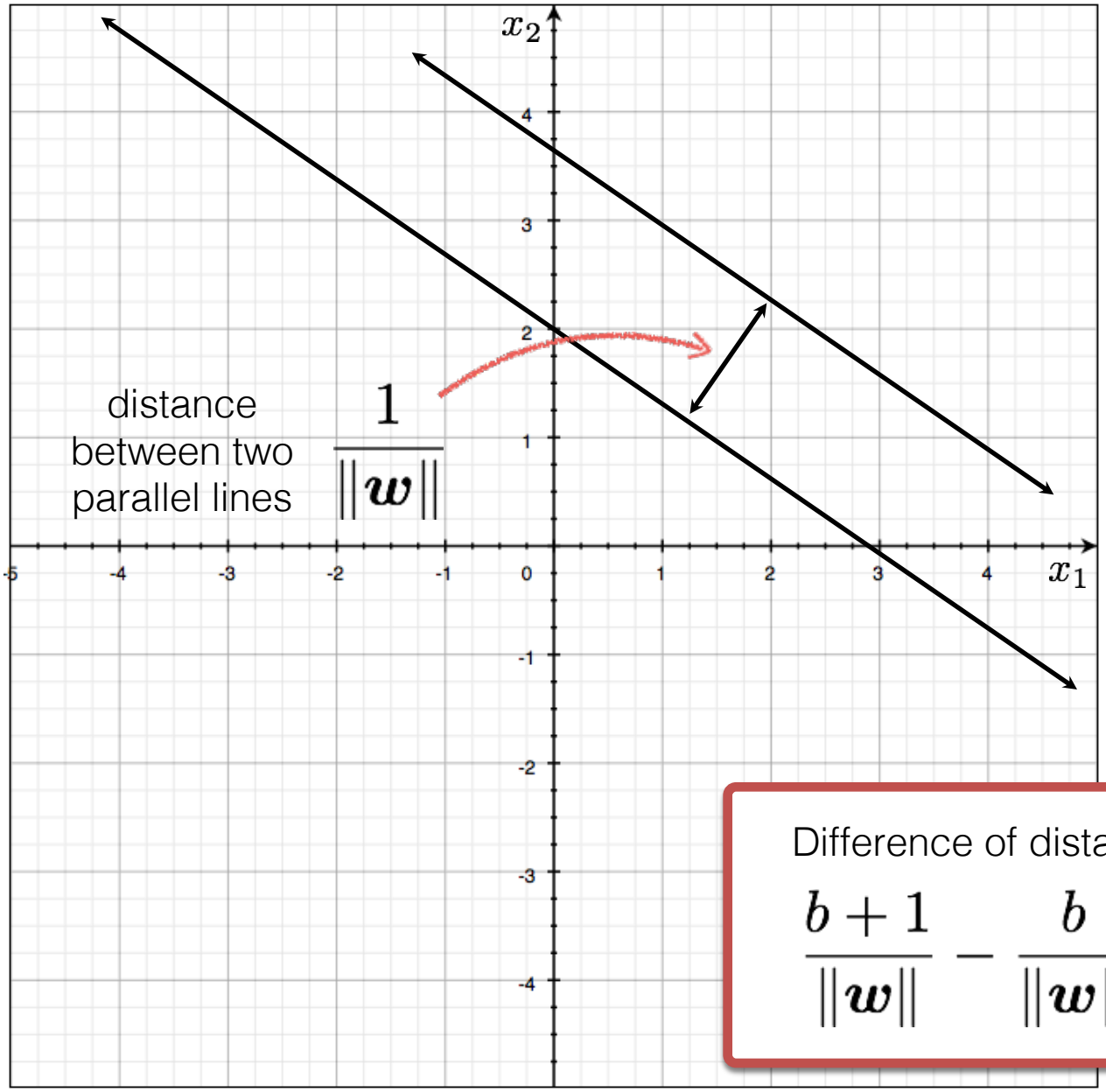
you get the normal form

$$x \cos\theta + y \sin\theta = \rho$$



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

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



distance between two parallel lines $\frac{1}{\|\boldsymbol{w}\|}$

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

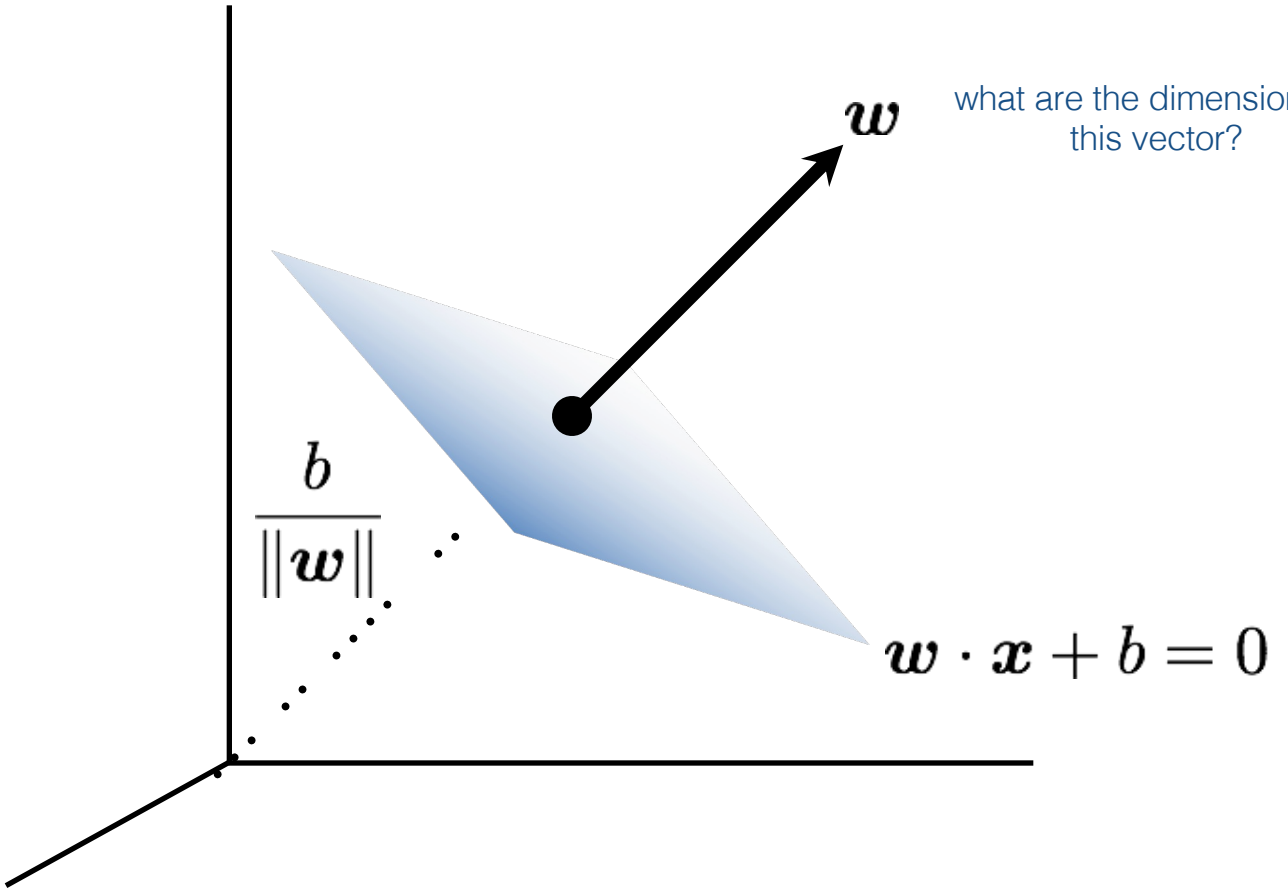
$$\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$$

Difference of distance to origin

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

Now we can go to 3D …

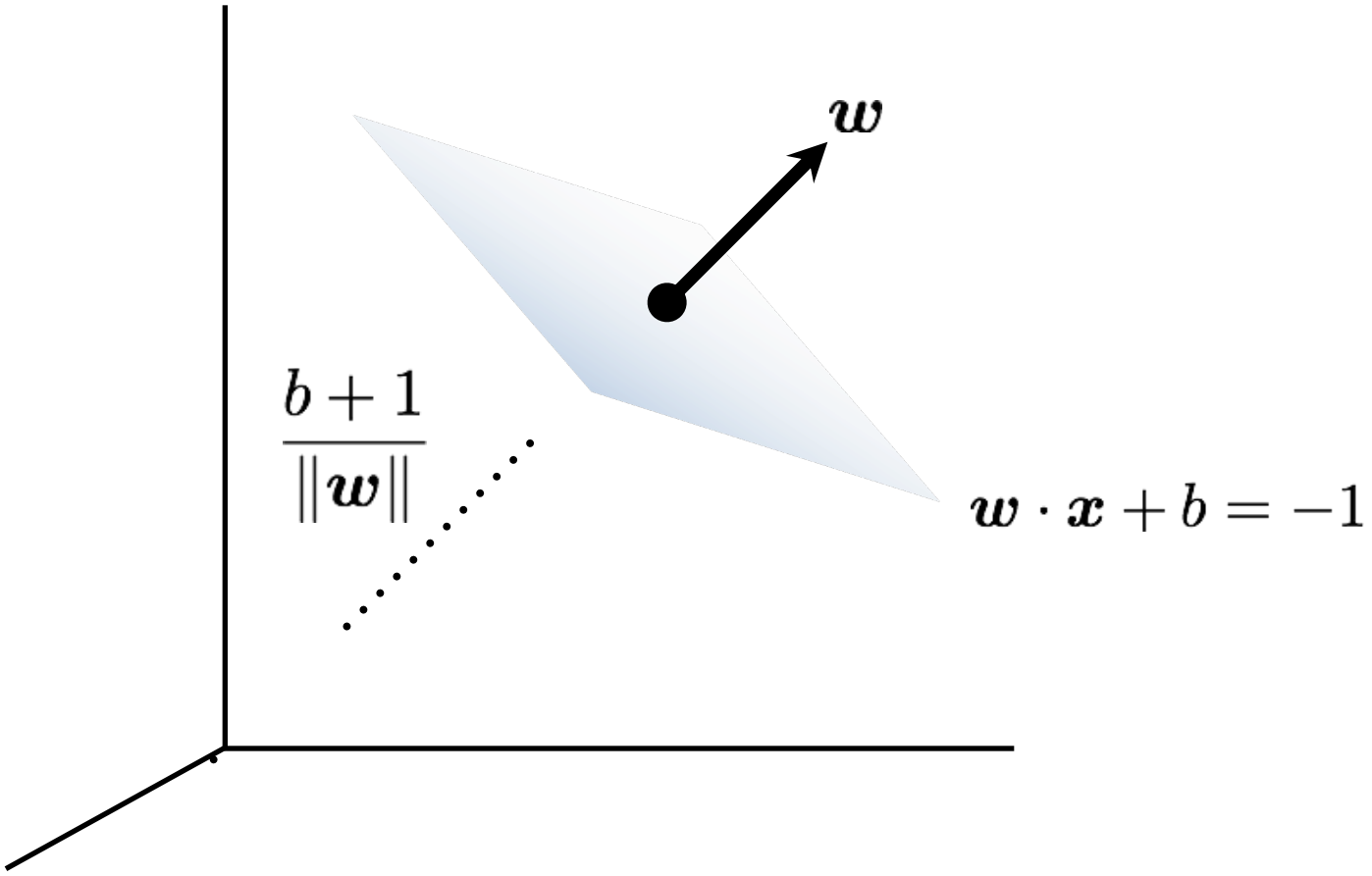
Hyperplanes (planes) in 3D



what are the dimensions of this vector?

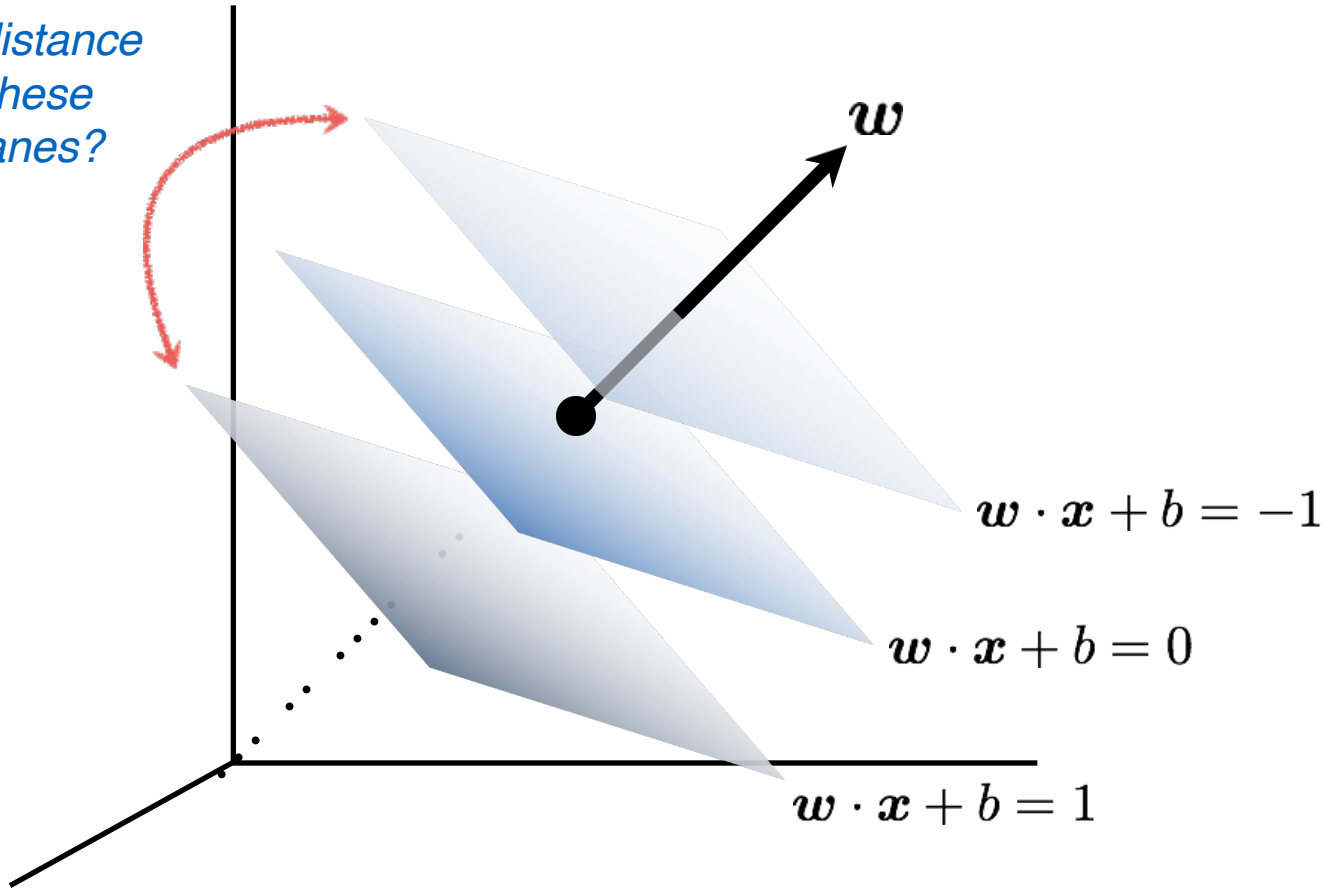
*What happens if you change **b**?*

Hyperplanes (planes) in 3D

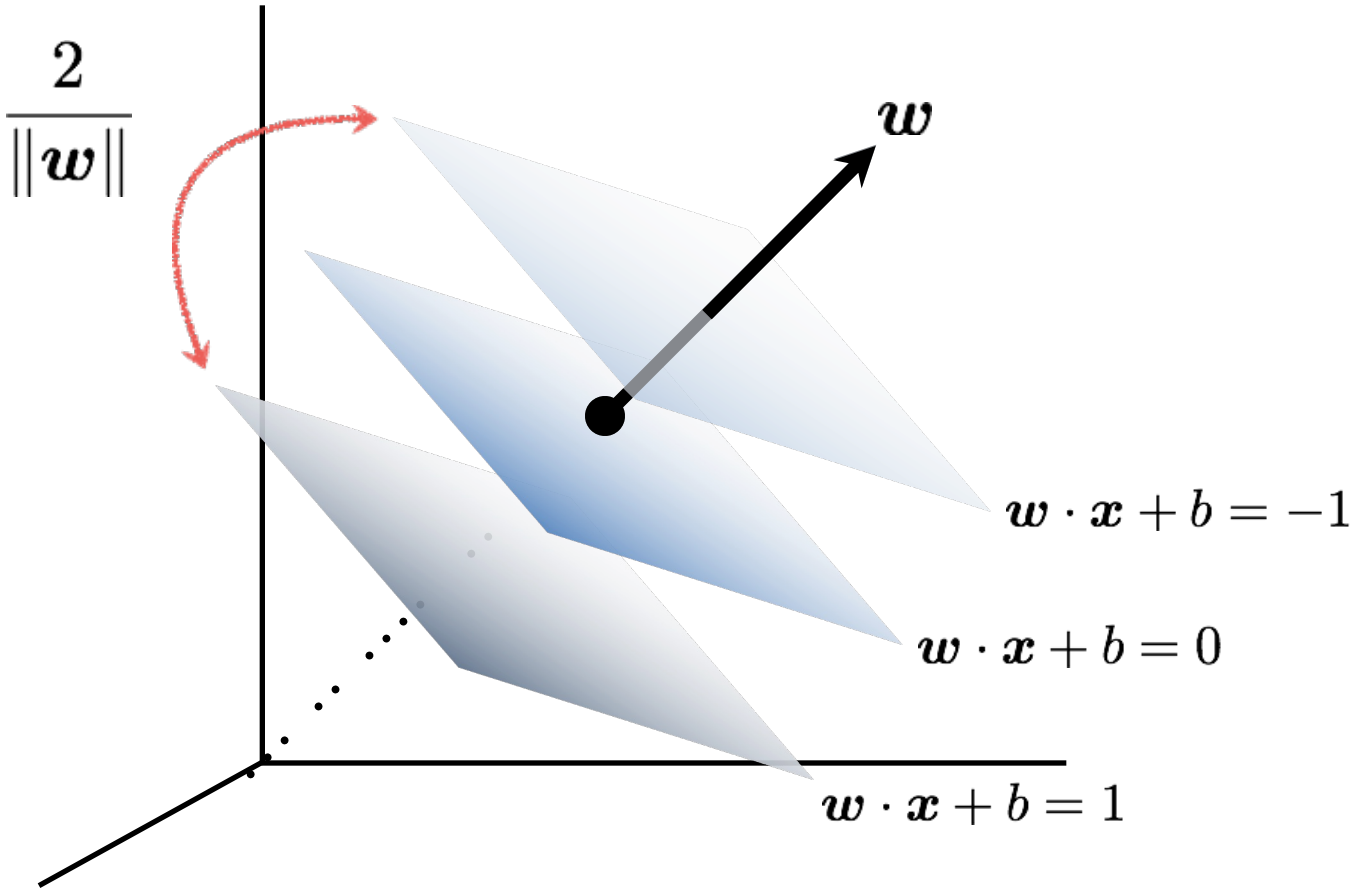


Hyperplanes (planes) in 3D

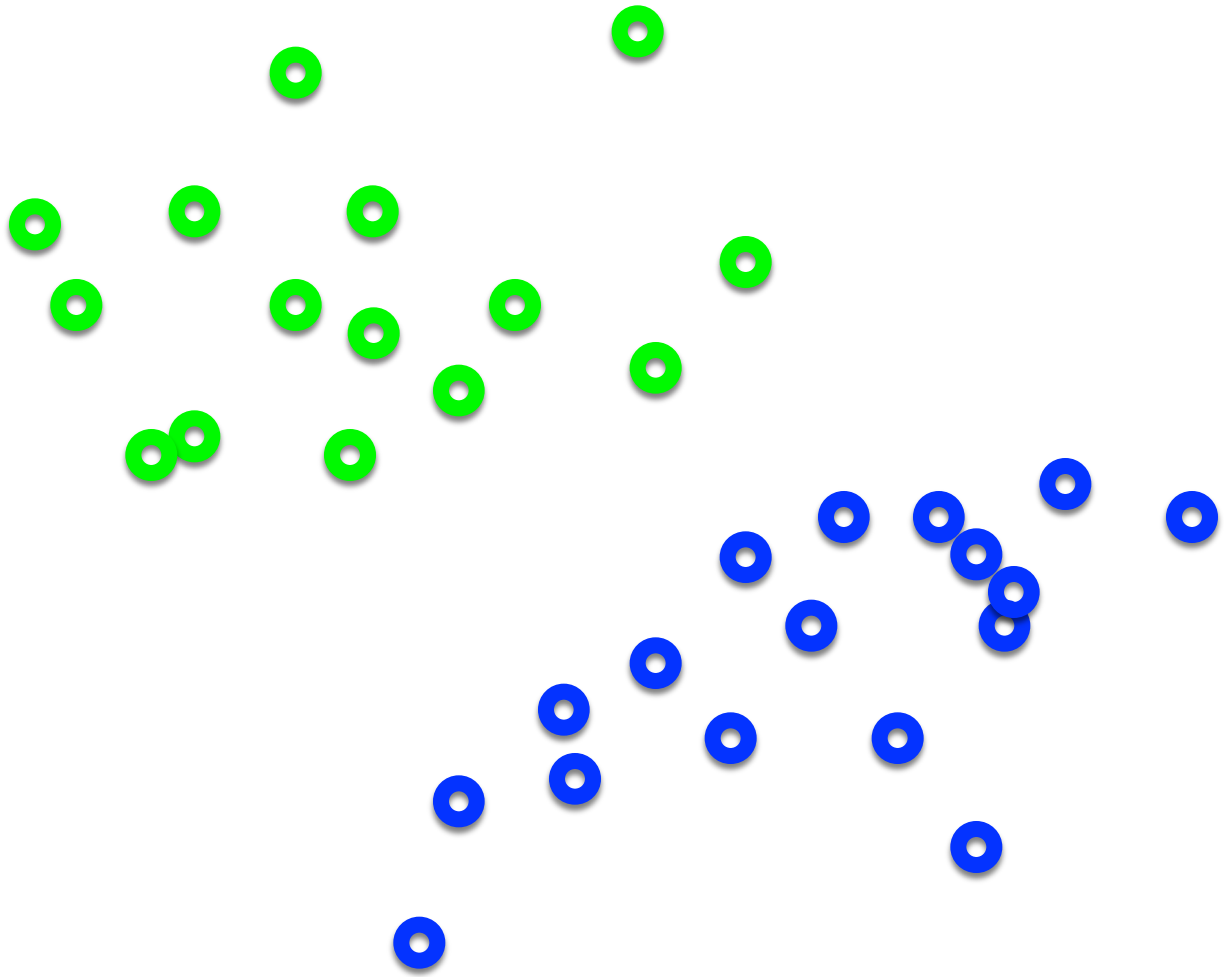
What's the distance between these parallel planes?



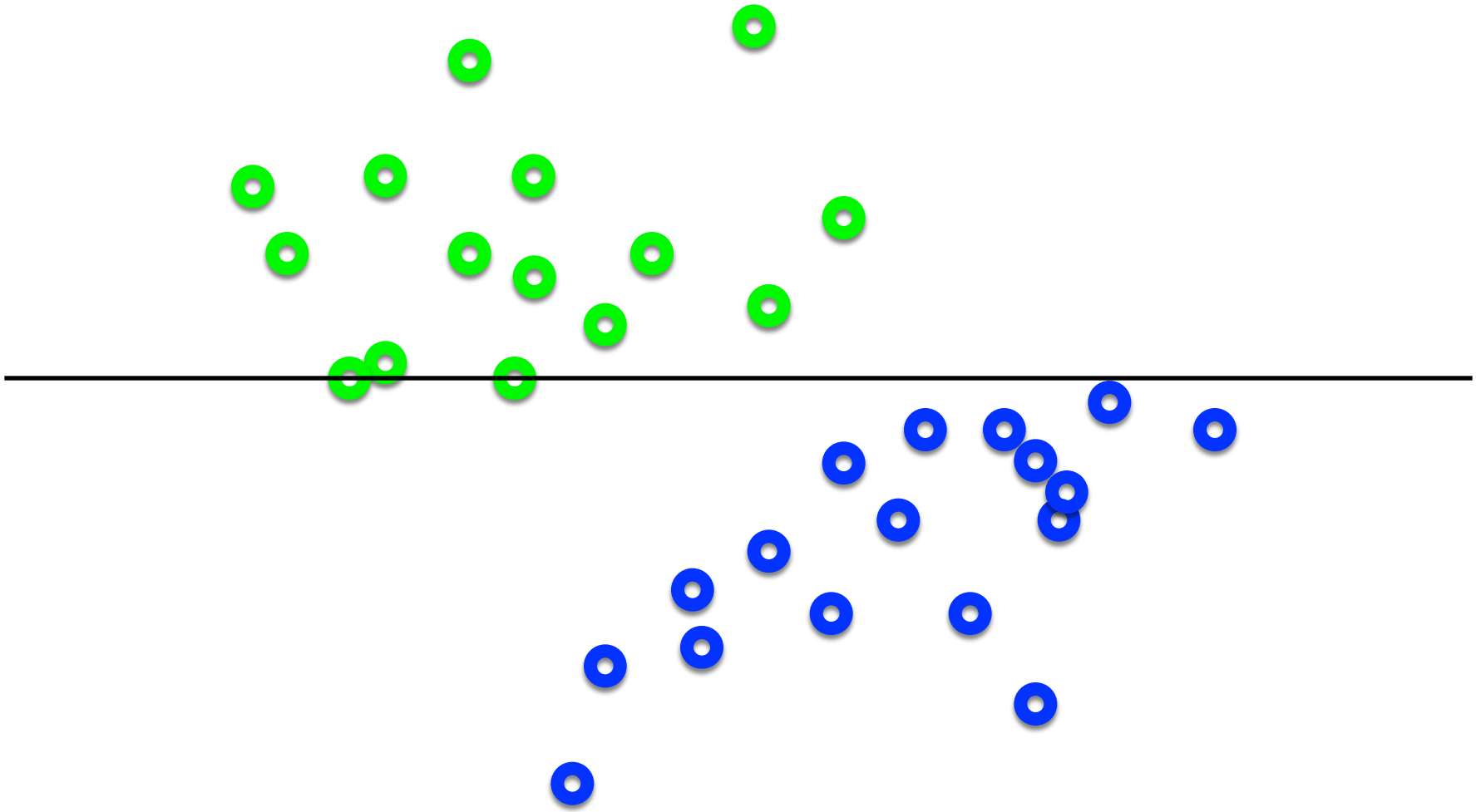
Hyperplanes (planes) in 3D



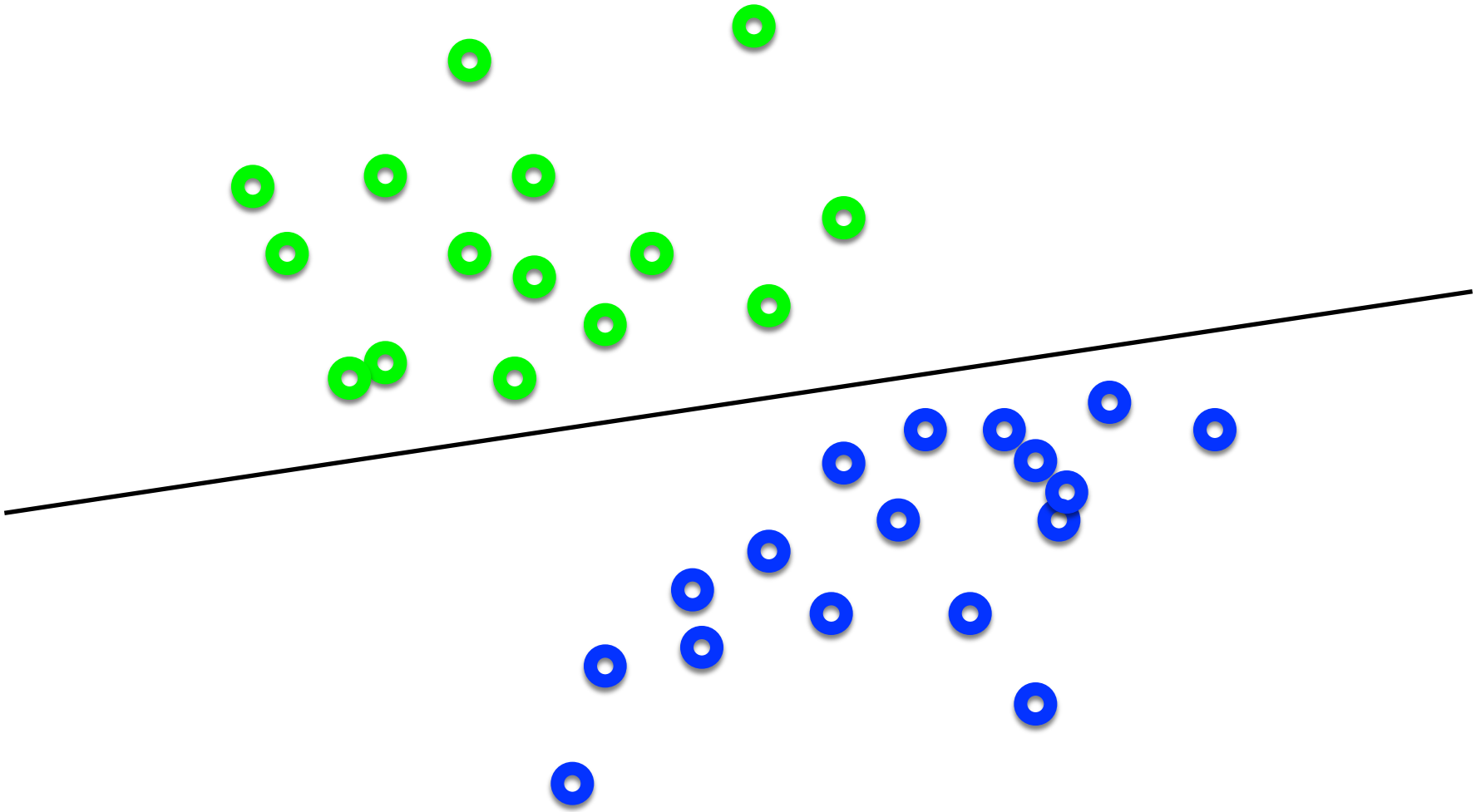
What's the best **w**?



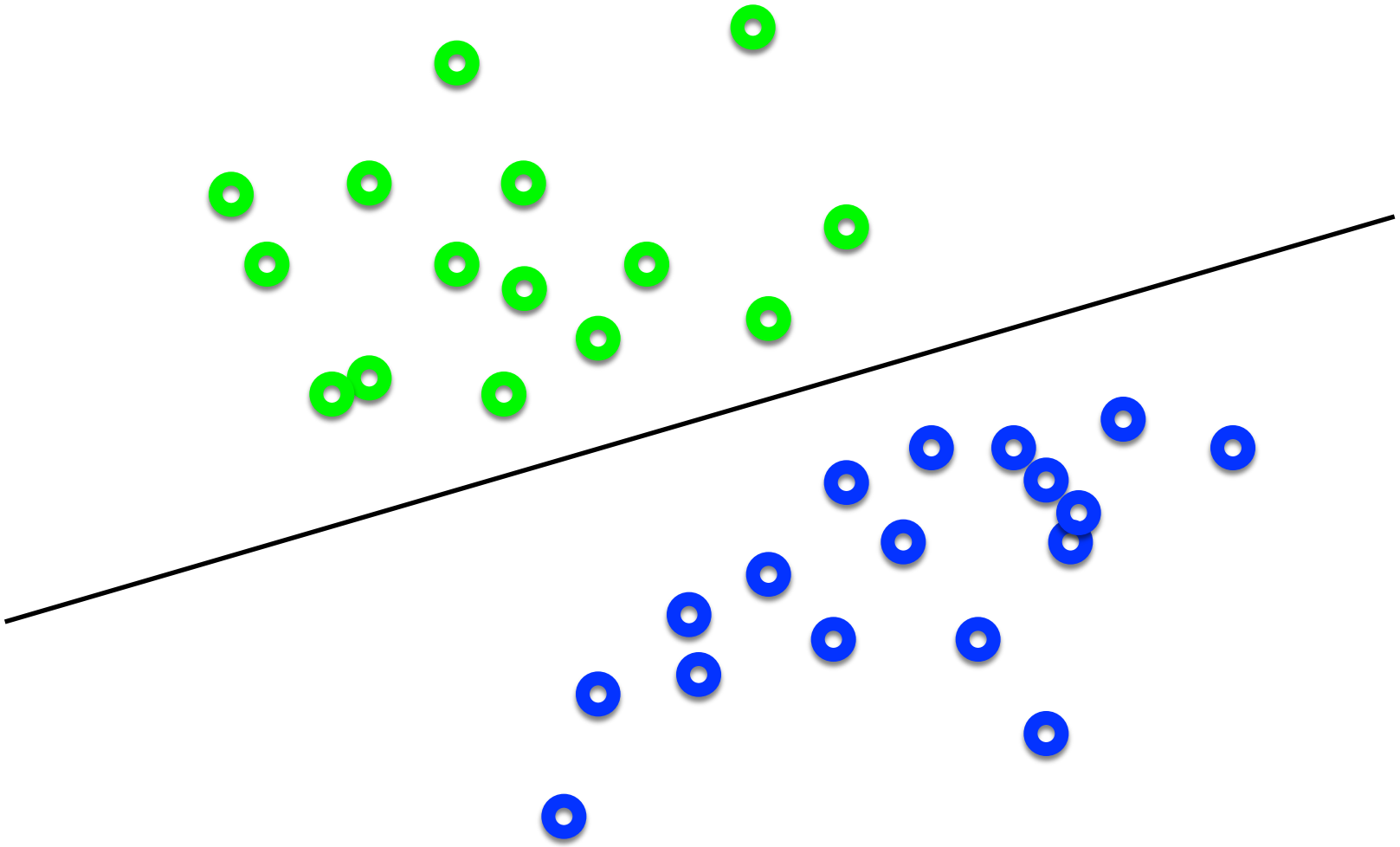
What's the best **w**?



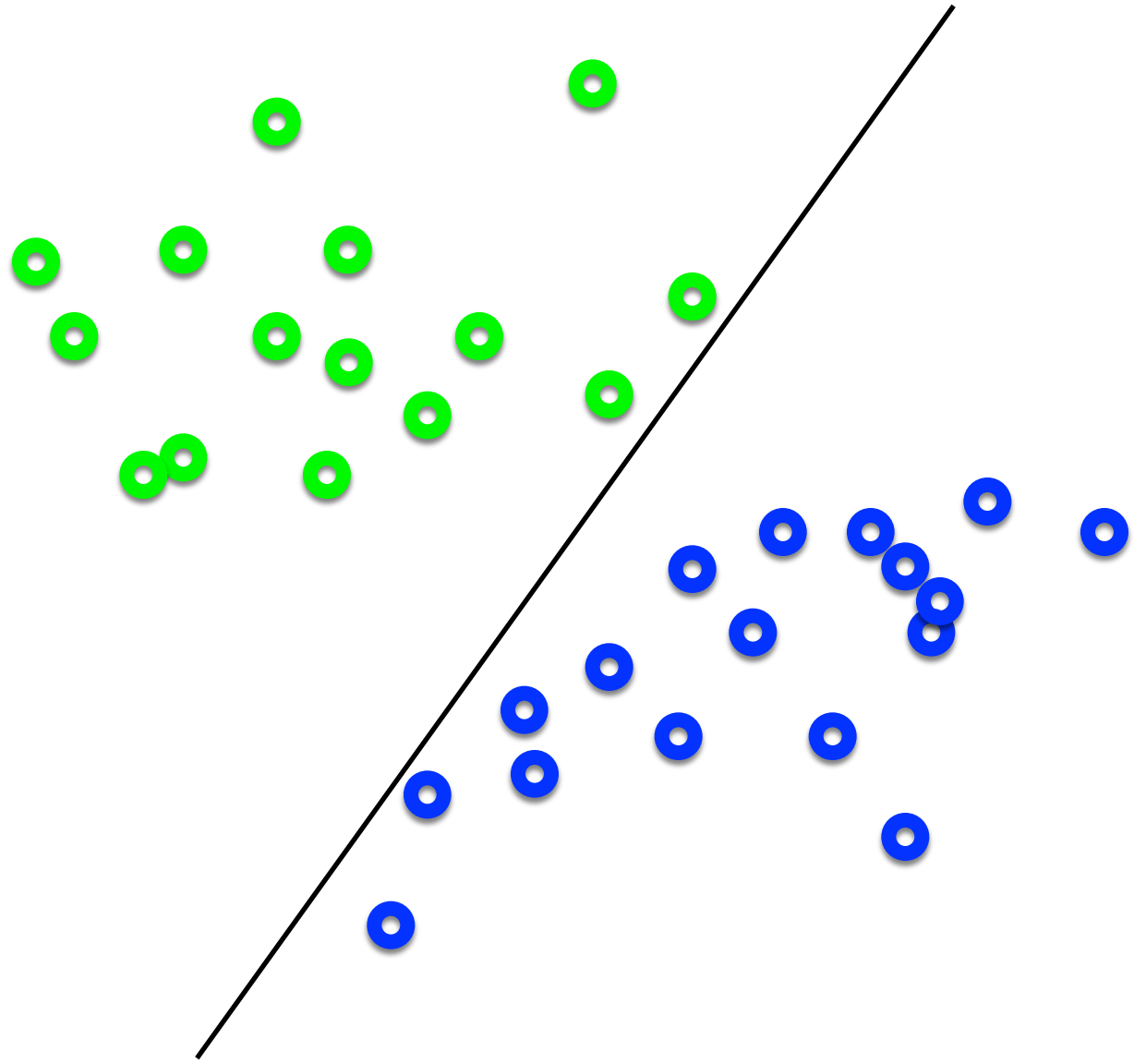
What's the best **w**?



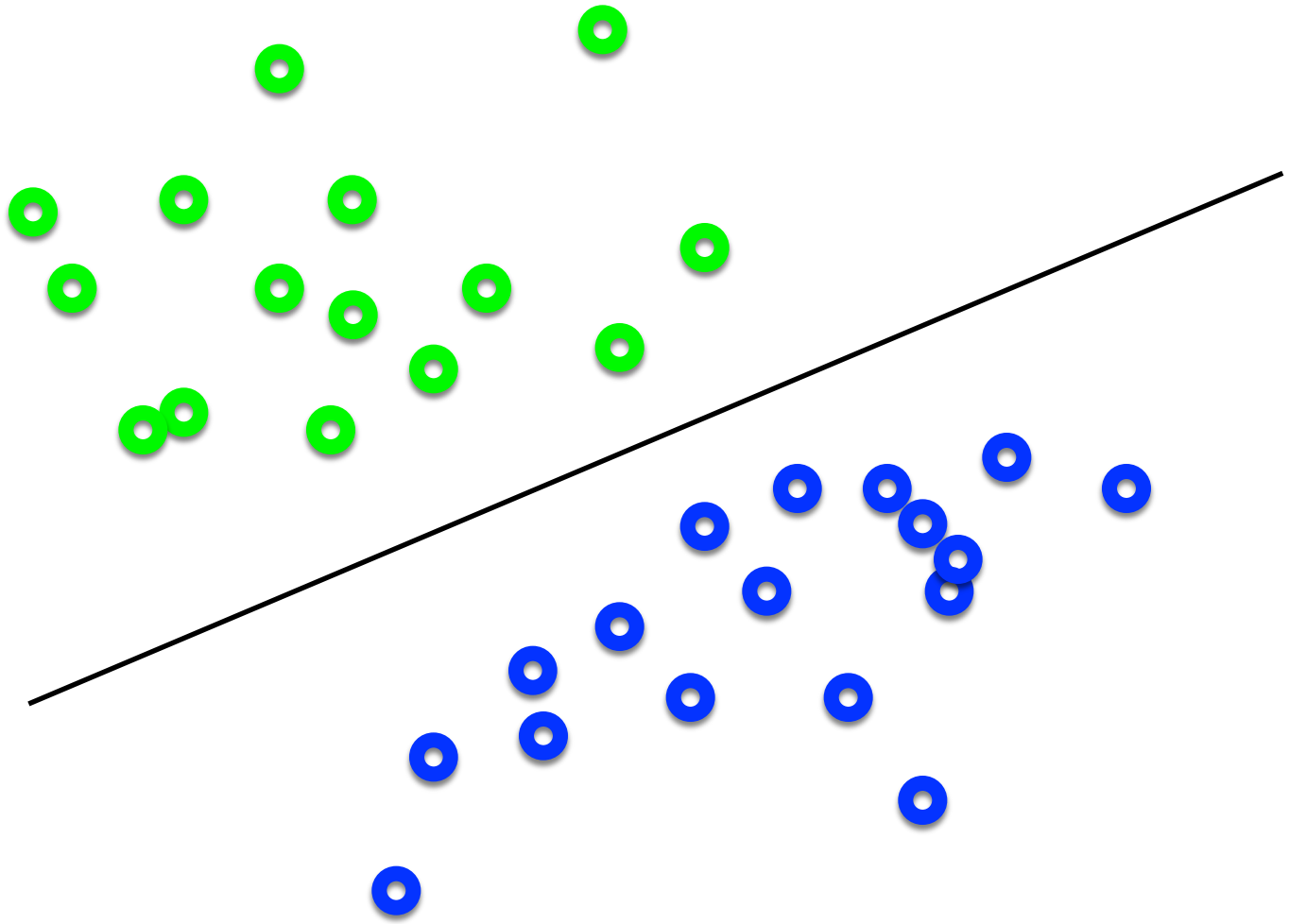
What's the best **w**?



What's the best **w**?

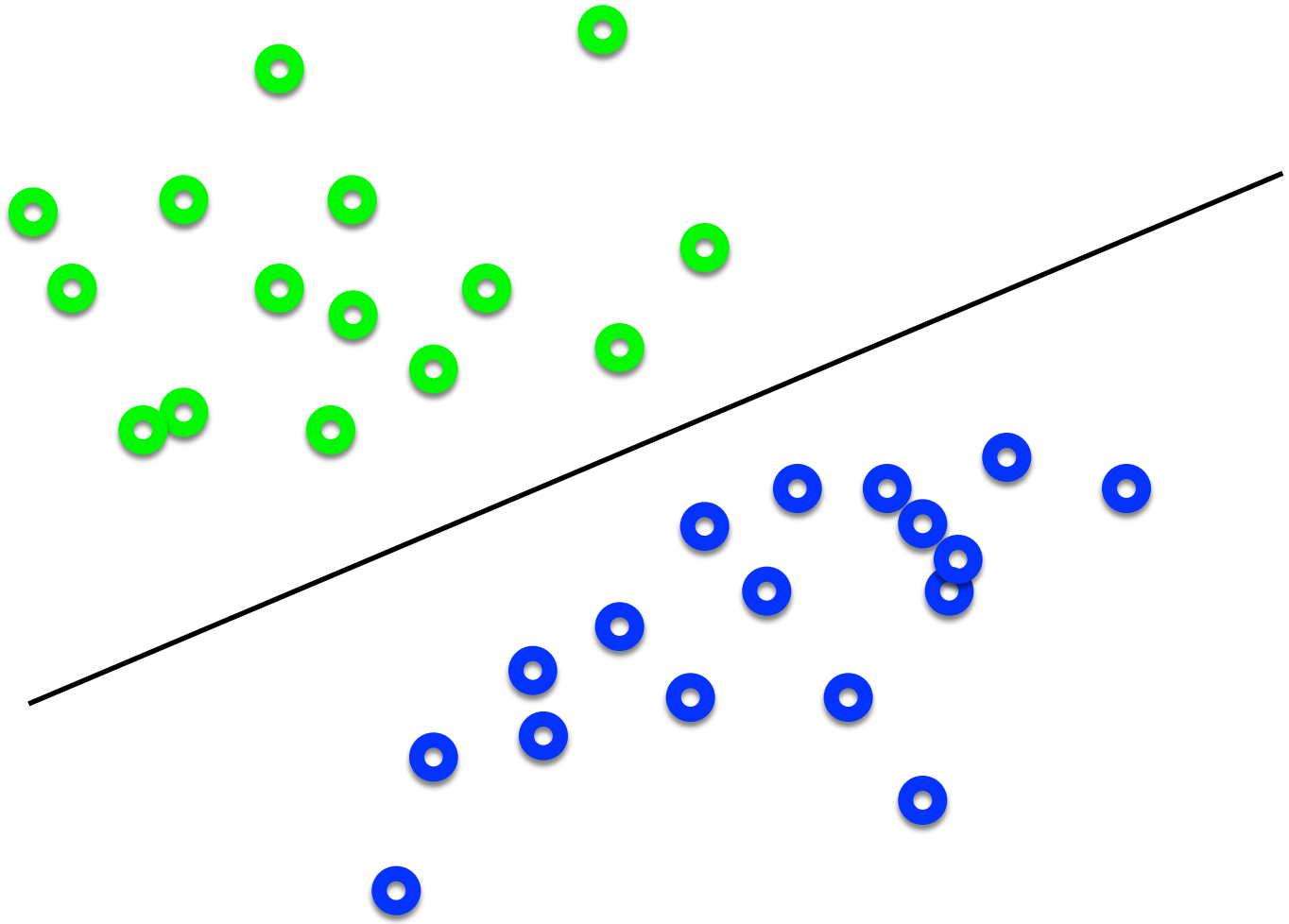


What's the best **w**?



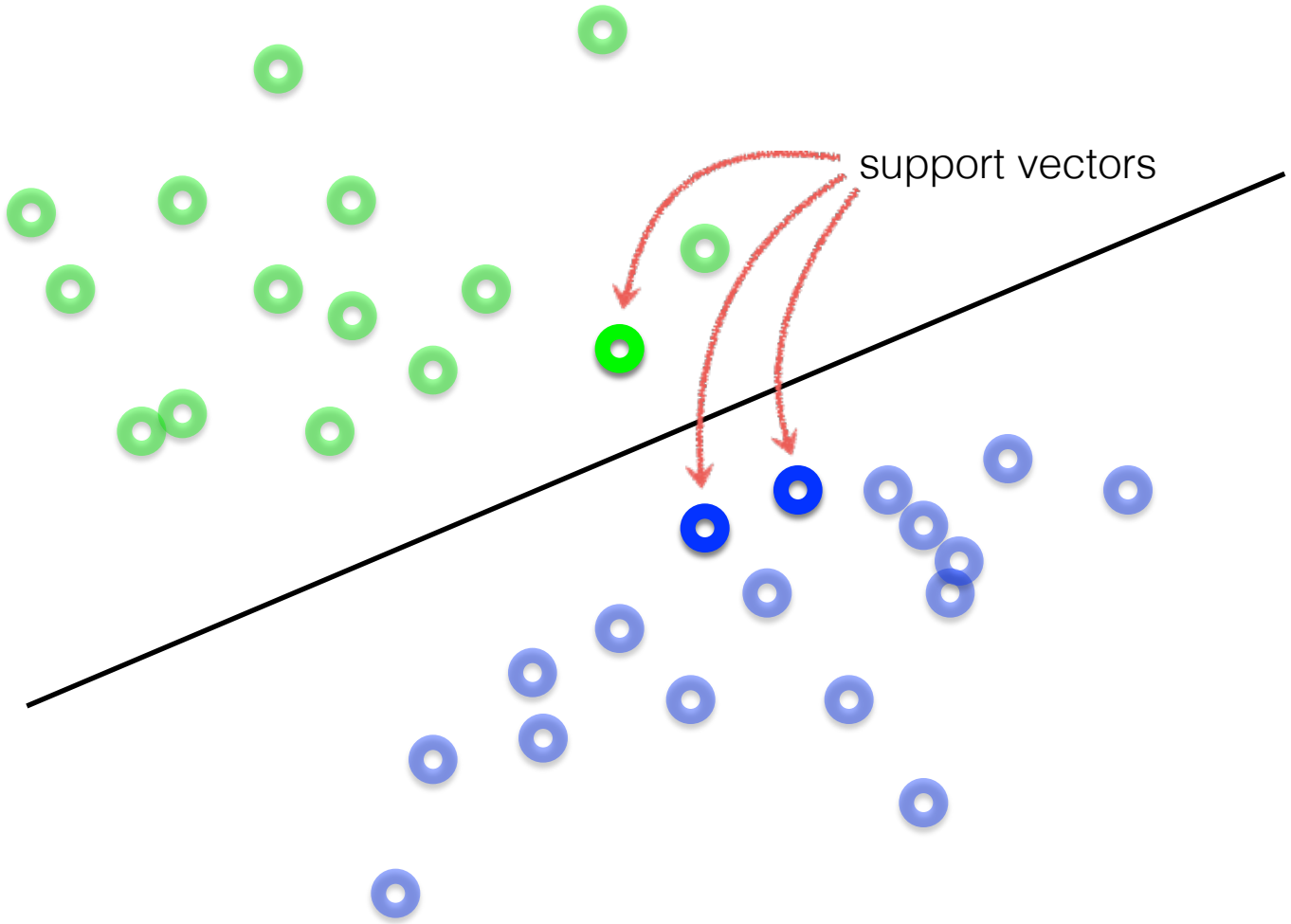
Intuitively, the line that is the farthest from all interior points

What's the best **w**?



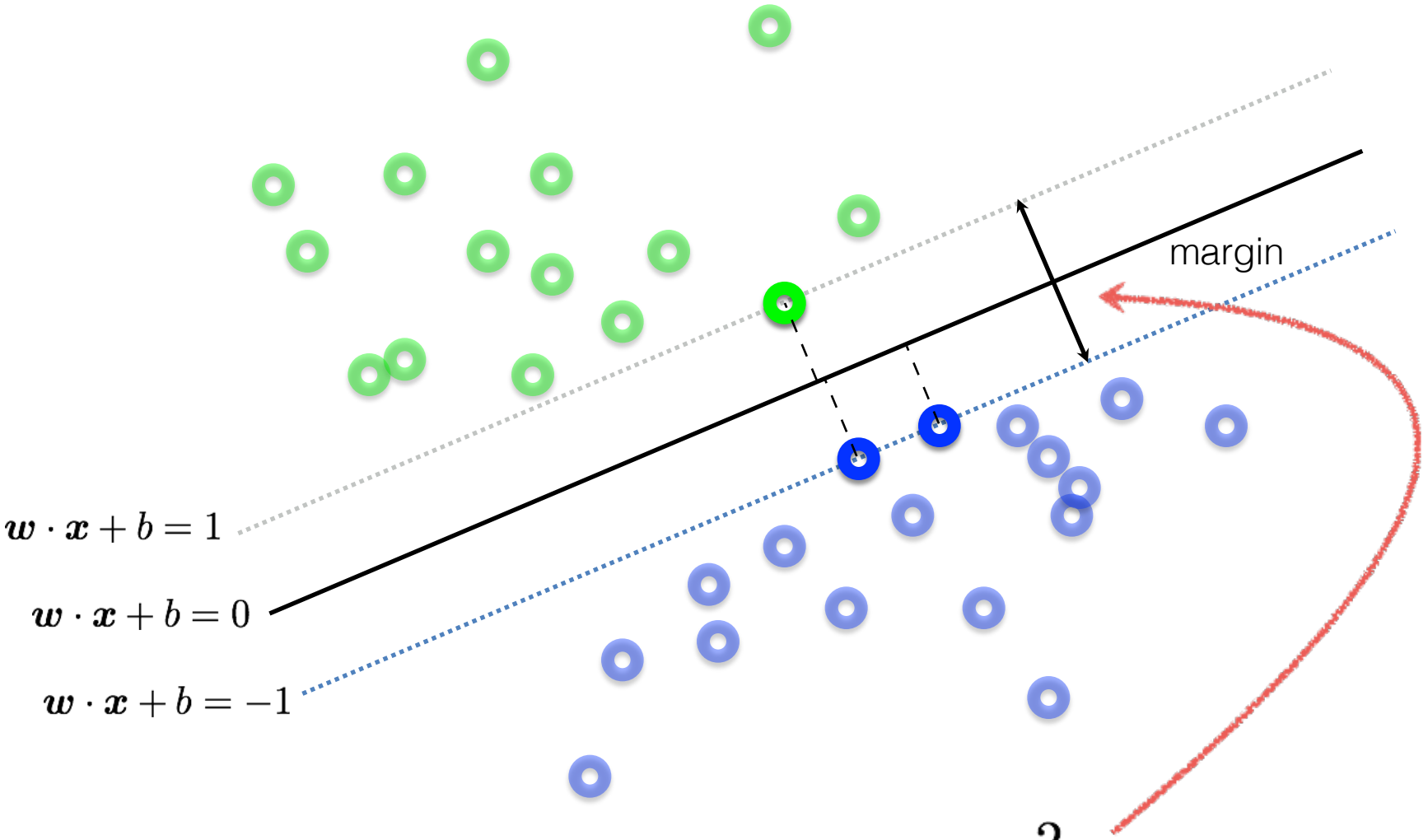
Maximum Margin solution:
most stable to perturbations of data

What's the best **w**?



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

Find hyperplane **w** such that …



the gap between parallel hyperplanes $\frac{2}{\|\boldsymbol{w}\|}$ is maximized

Can be formulated as a maximization problem

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

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

What does this constraint mean?

label of the data point

Why is it +1 and -1?

Can be formulated as a maximization problem

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

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

Equivalently,

Where did the 2 go?

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

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

What happened to the labels?

‘Primal formulation’ of a linear SVM

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

Objective Function

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

Constraints

This is a convex quadratic programming (QP) problem

(a unique solution exists)

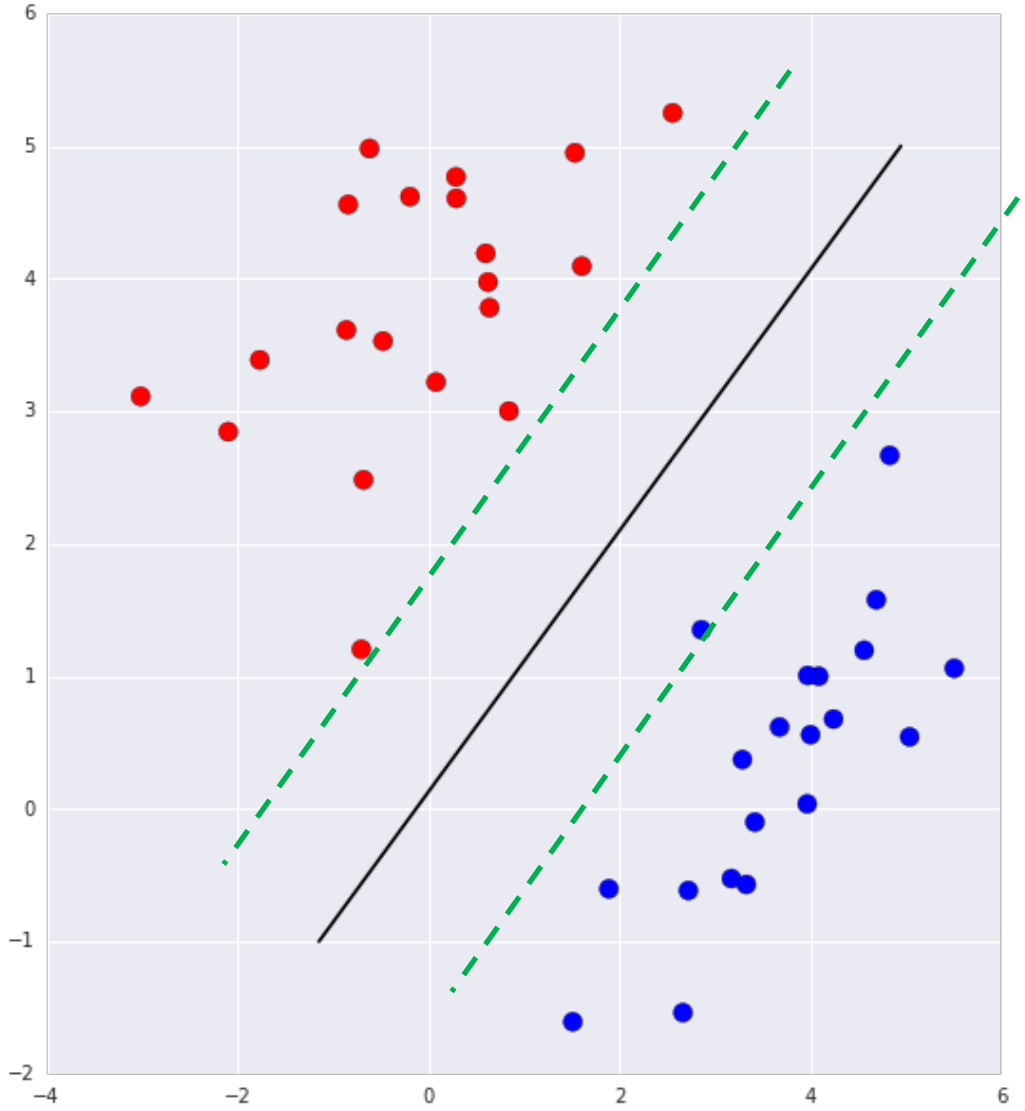
Revisit SVM Cost Function

$$\min_{\theta} \frac{1}{2} \sum_{j=1}^{n} \theta_j^2$$

$$\text{s.t.} \quad \theta^T x^{(i)} \geq 1 \qquad \text{if } y^{(i)} = 1$$

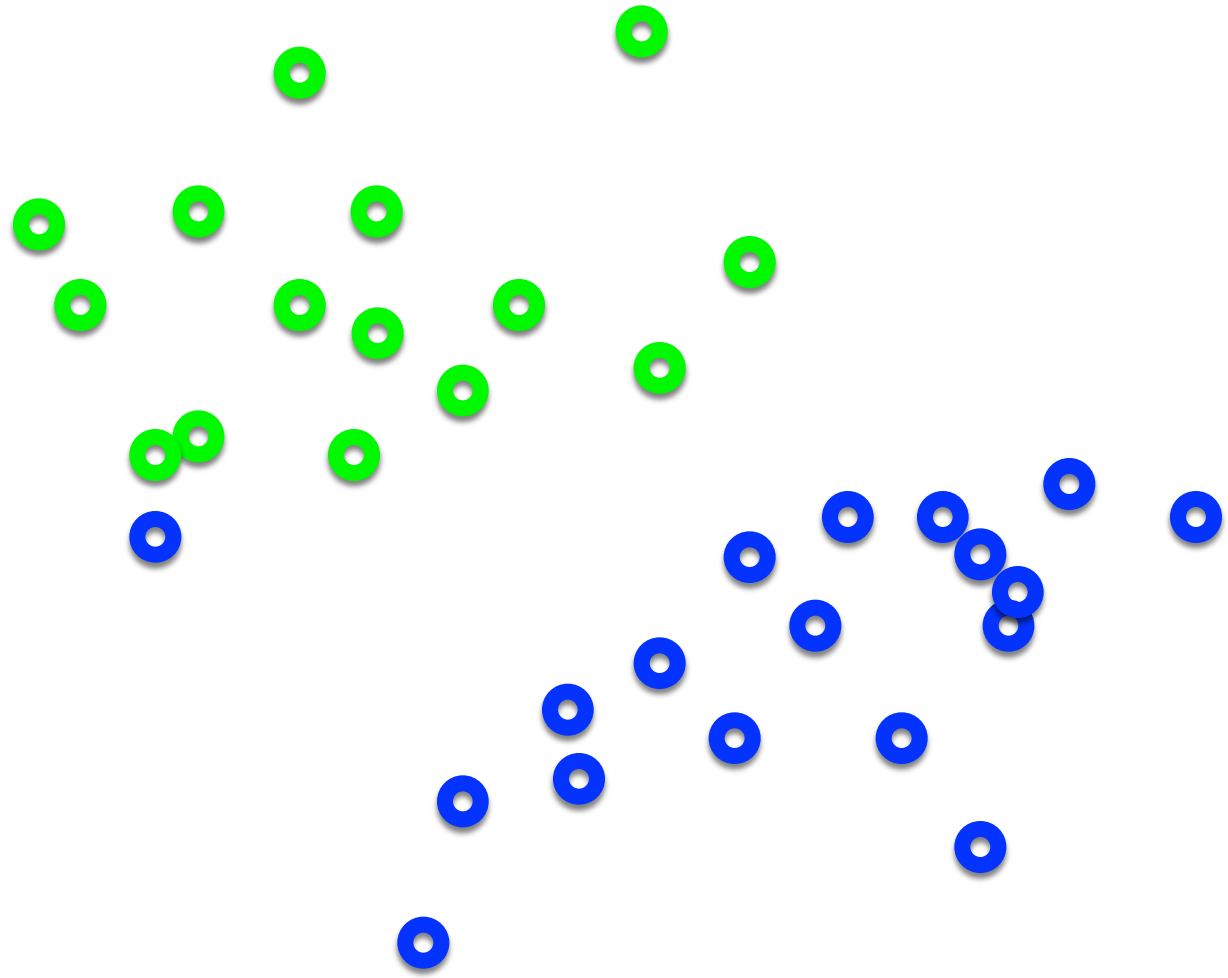
$$\theta^T x^{(i)} \leq -1 \qquad \text{if } y^{(i)} = 0$$

Large Margin Classifier

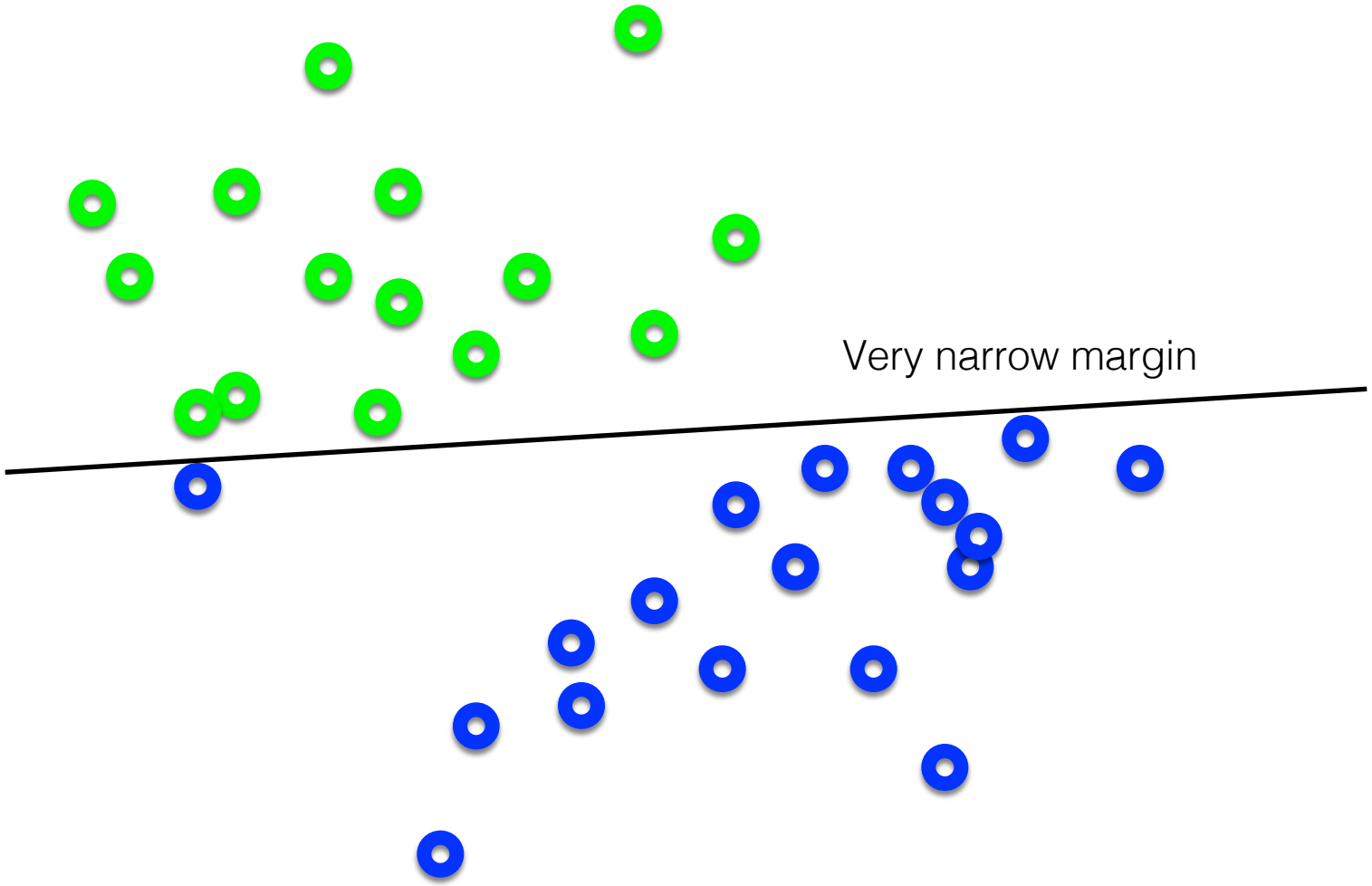


‘soft’ margin

What's the best **w**?



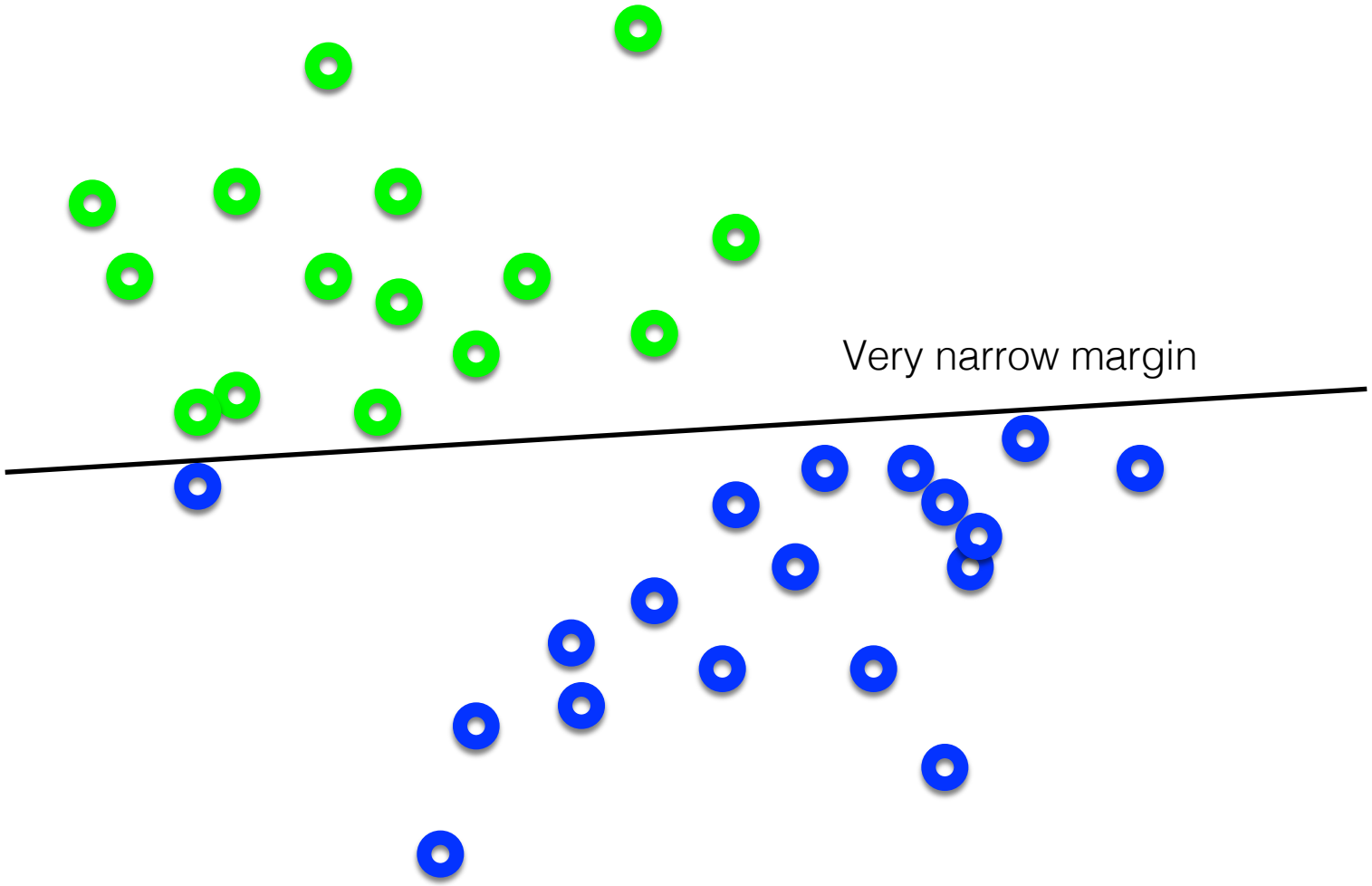
What's the best **w**?



Separating cats and dogs

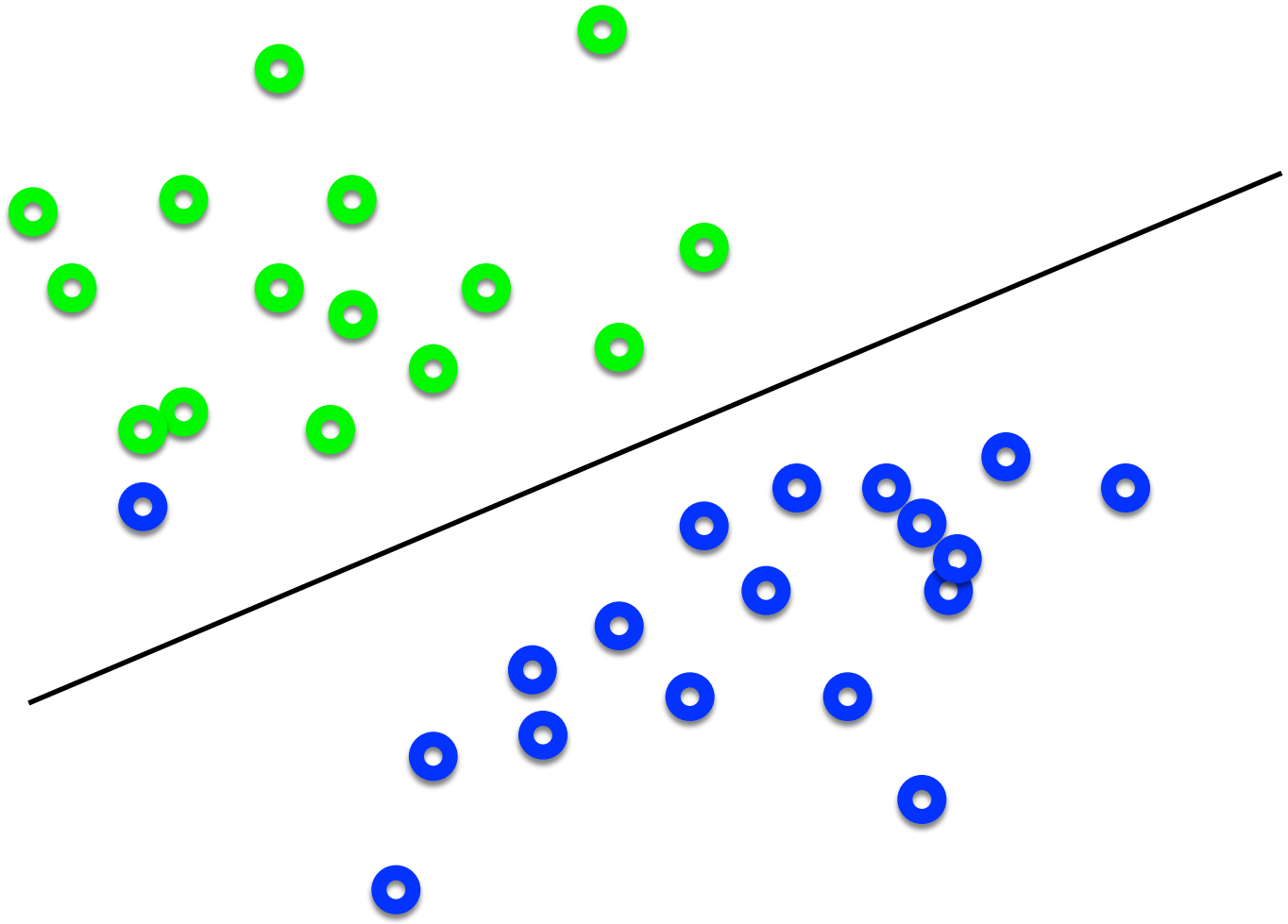
Very narrow margin

What's the best **w**?



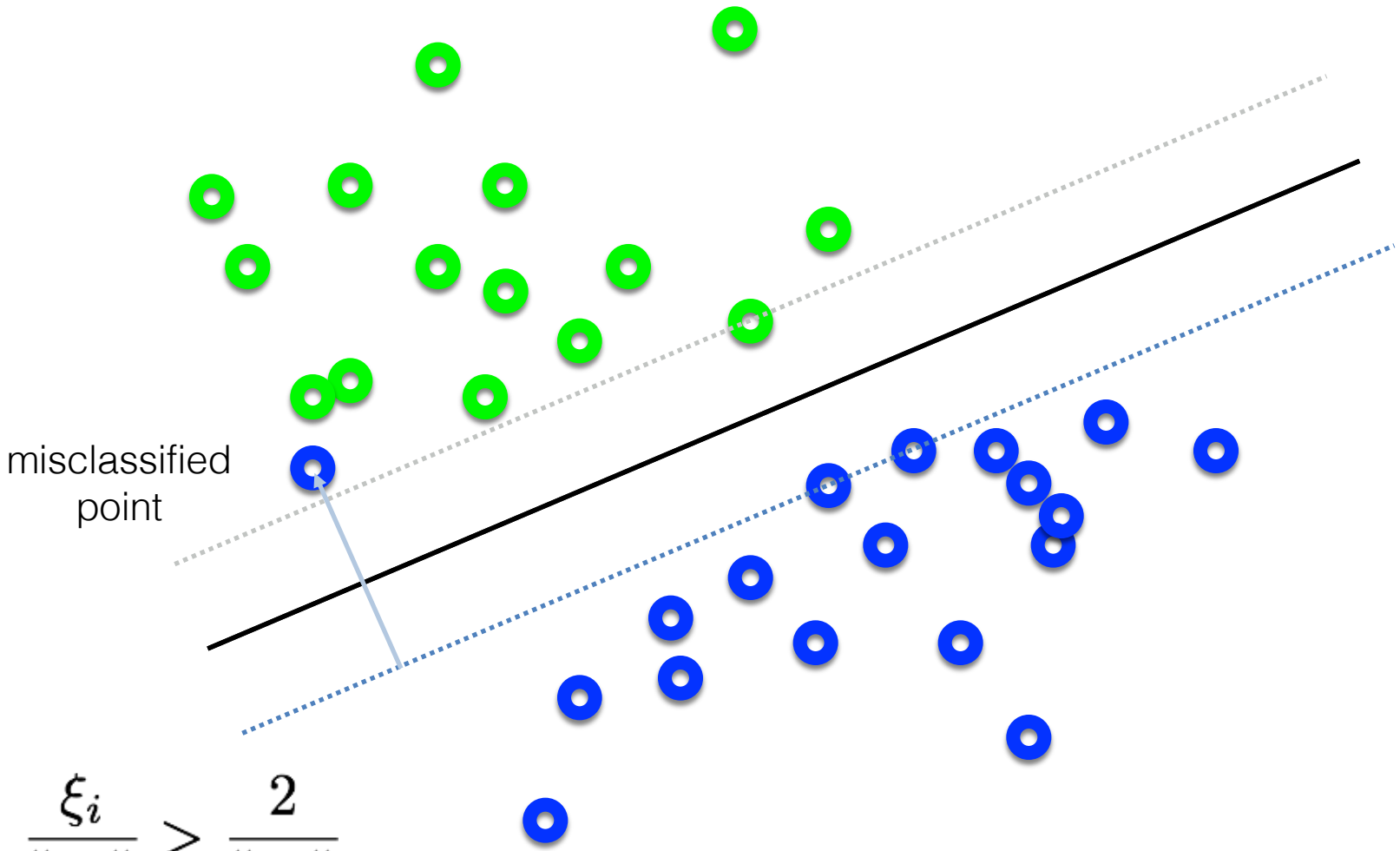
Intuitively, we should allow for some misclassification if we can get more robust classification

What's the best **w**?



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

Adding slack variables $\xi_i \geq 0$



$$\frac{\xi_i}{\|\boldsymbol{w}\|} > \frac{2}{\|\boldsymbol{w}\|}$$

‘soft’ margin

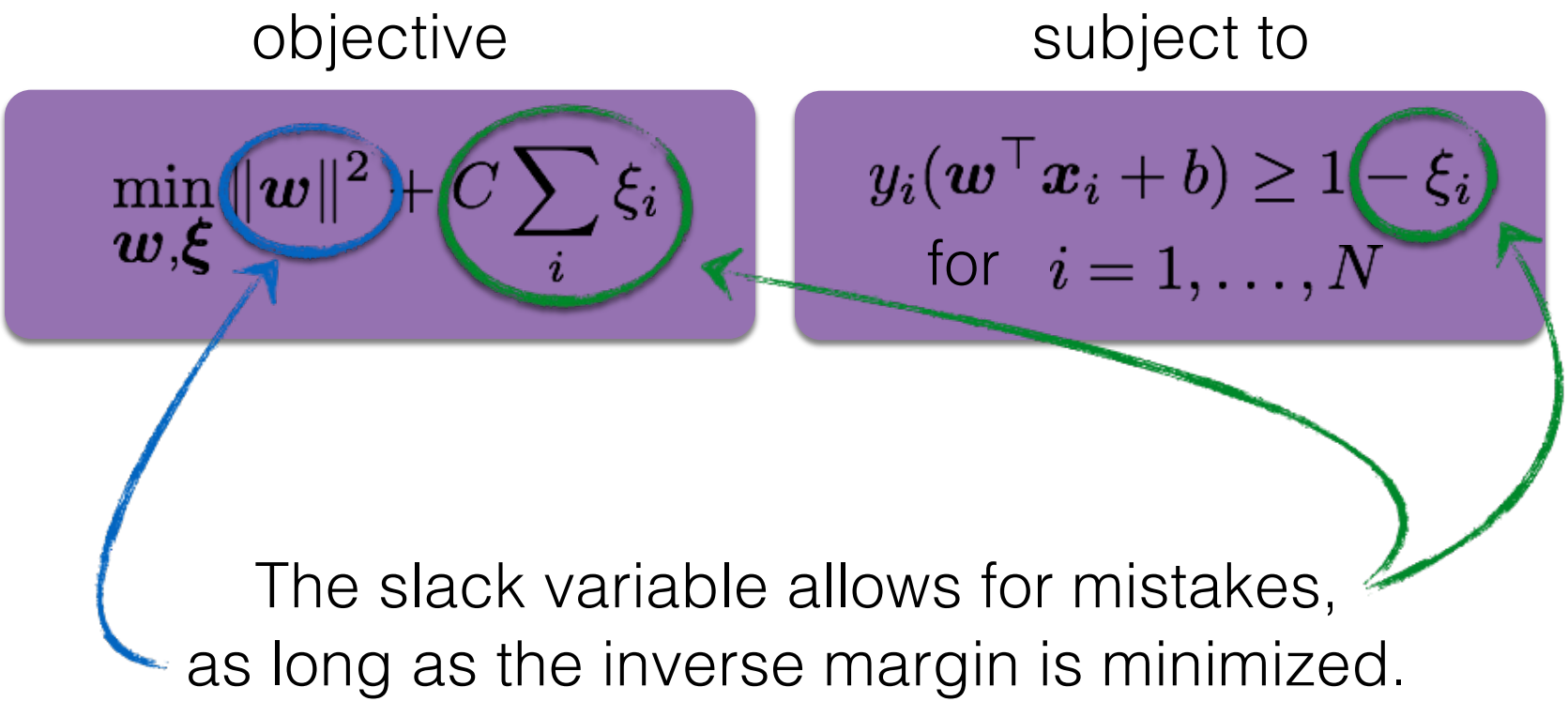
objective

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

subject to

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

‘soft’ margin


objective
subject to
$\min_{\boldsymbol{w},\boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C\sum_i \xi_i$
$y_i(\boldsymbol{w}^\top \boldsymbol{x}_i + b) \geq 1 - \xi_i$
for $i = 1, \ldots, N$
The slack variable allows for mistakes,
as long as the inverse margin is minimized.

‘soft’ margin

objective

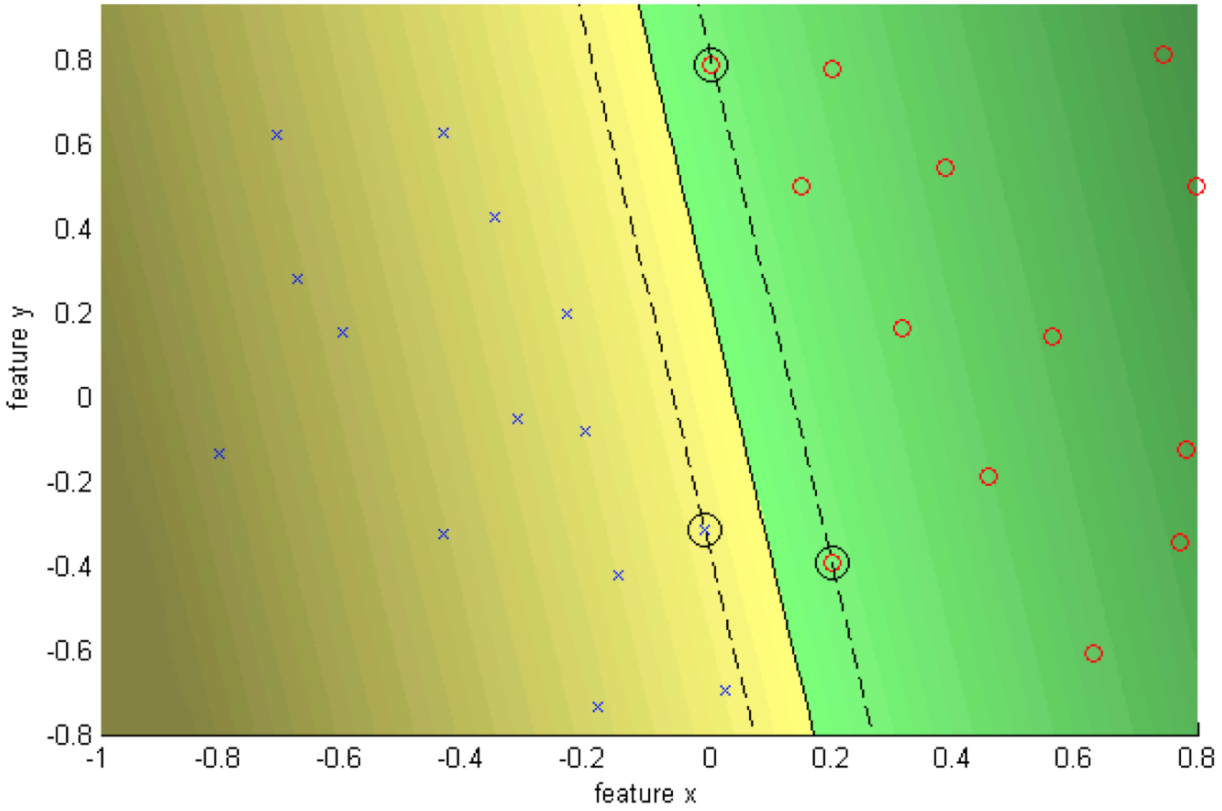
$$\min_{\boldsymbol{w},\boldsymbol{\xi}} \|\boldsymbol{w}\|^2 + C\sum_i \xi_i$$

subject to

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

- 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)

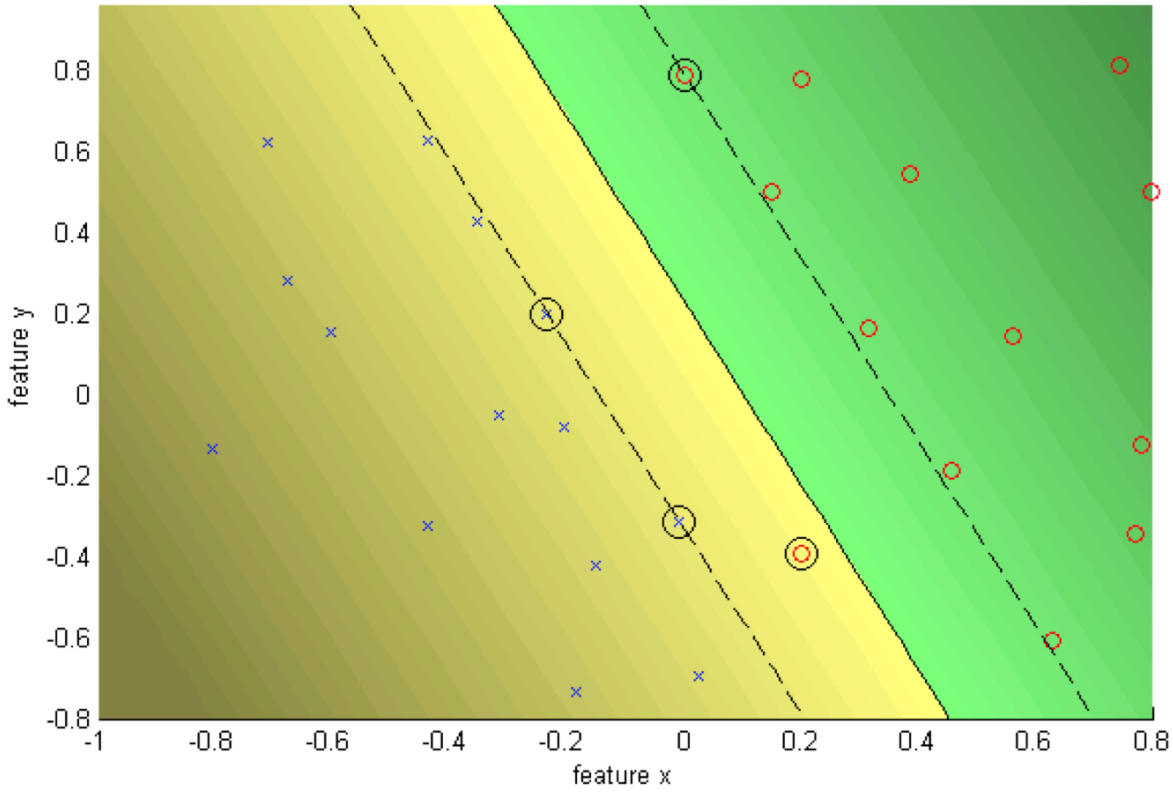
C = Infinity hard margin



Comment Window

SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: Inf
Kernel evaluations: 971
Number of Support Vectors: 3
Margin: 0.0966
Training error: 0.00%

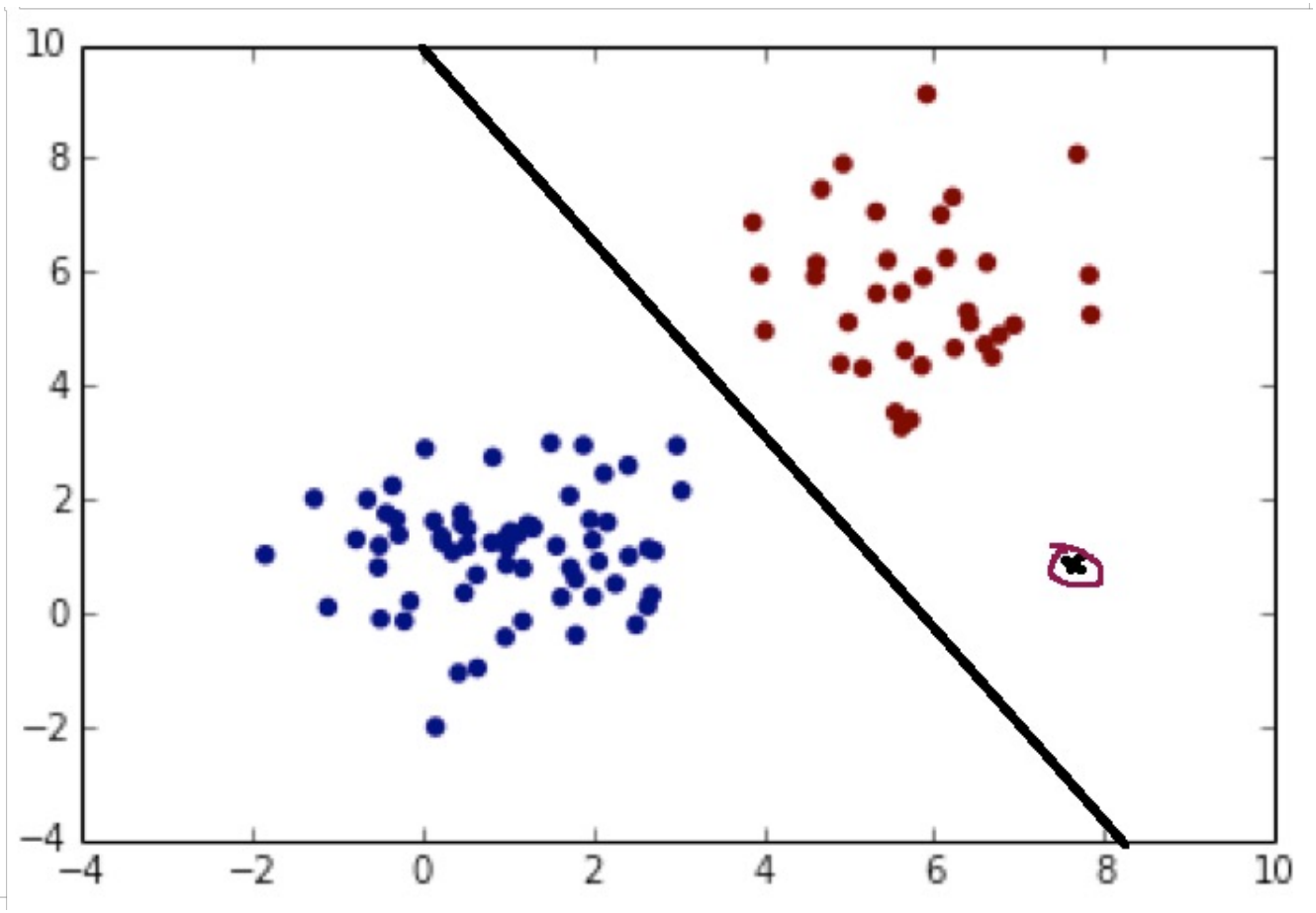
C = 10 soft margin



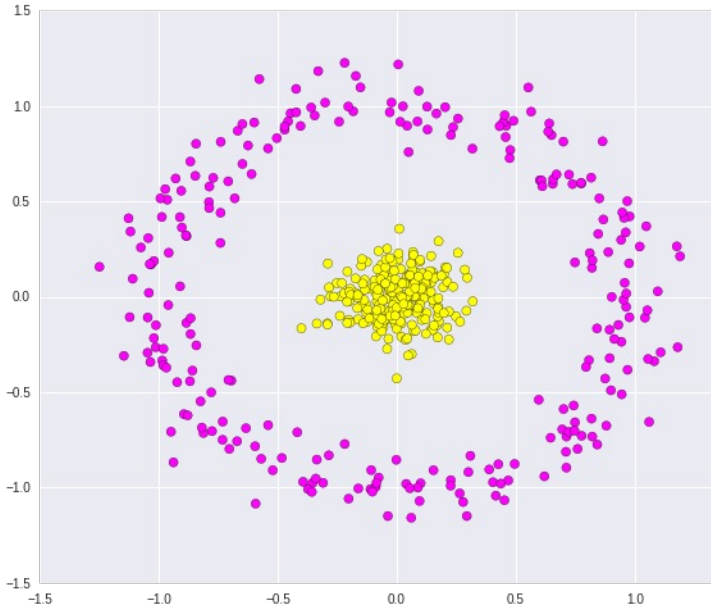
Comment Window

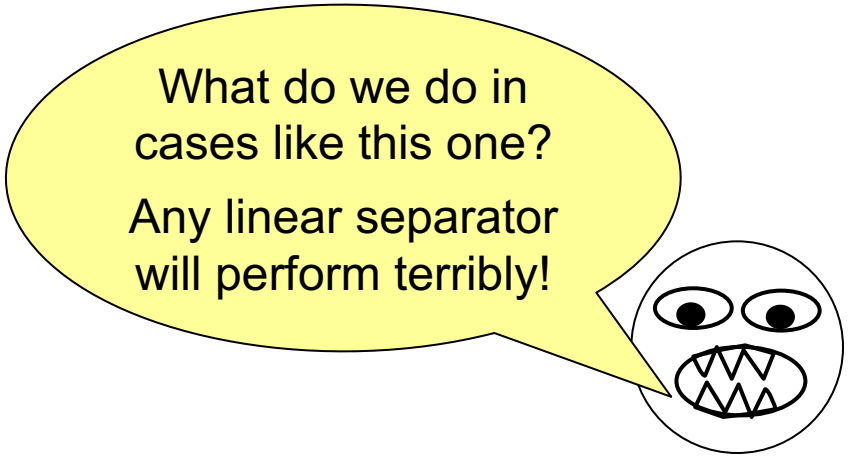
```
SVM (L1) by Sequential Minimal Optimizer
Kernel: linear (-), C: 10.0000
Kernel evaluations: 2645
Number of Support Vectors: 4
Margin: 0.2265
Training error: 3.70%
```

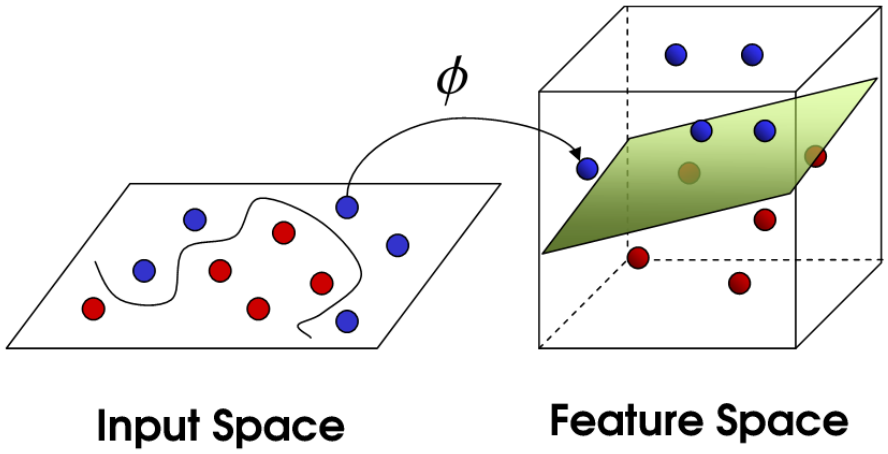
SVMs: the basic idea (linearly separable case)



Feature Transformation



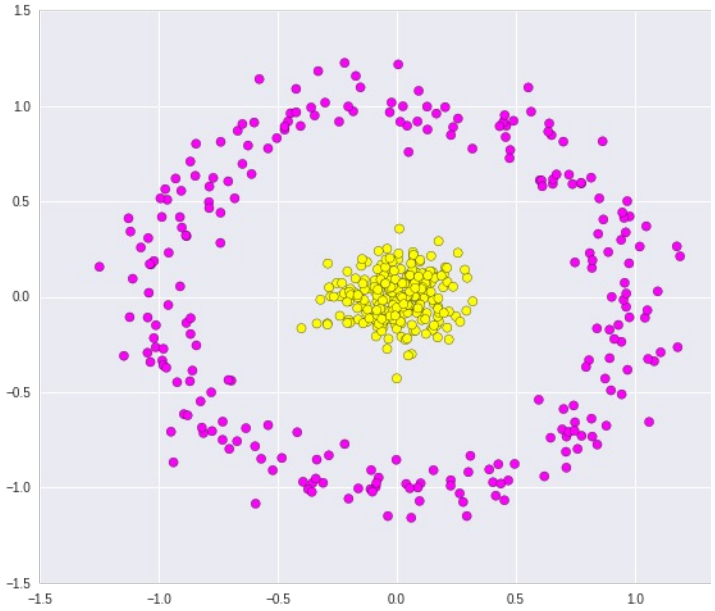




Solution:

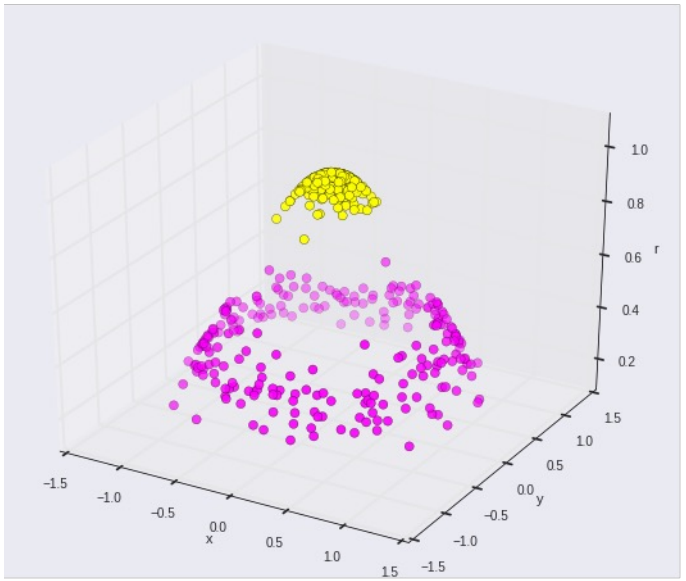
1) Map input space to a high-dimensional feature space.
2) Learn a linear decision boundary (hyperplane) in the high-dimensional space.
3) Map back to lower-dimensional space, giving a non-linear boundary.

Non-linear decision boundaries

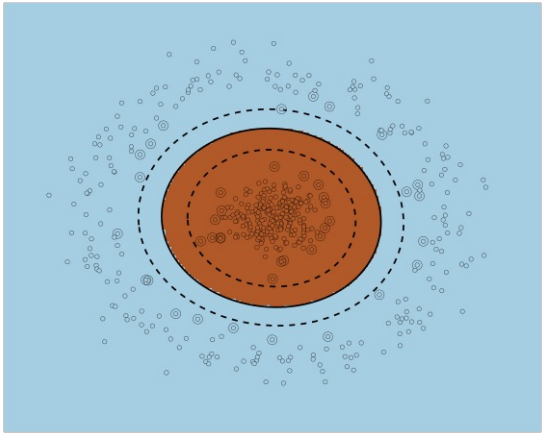


$$\phi : (x, y) \rightarrow (x, y, r)$$

$$r = e^{-(x^2+y^2)}$$



The resulting classifier perfectly separates the training data.



Some common kernel functions

Linear kernel: $\phi : x \to x \qquad K(x_i, x_j) = x_i \cdot x_j$

Non-linear kernels

Polynomial kernel: $K(x_i, x_j) = (\gamma(x_i \cdot x_j) + r)^d$

Sigmoid kernel: $K(x_i, x_j) = \tanh(\gamma(x_i \cdot x_j) + r)$

Gaussian kernel: $K(x_i, x_j) = \exp\left(-\gamma \|x_i - x_j\|^2\right)$

The Gaussian kernel is usually called the "radial basis function", or **RBF**, kernel. It is one of the most widely used choices of kernel and a good default option.

Very cool trick (the "kernel trick"): instead of mapping both x_i and x_j into a high-dimensional space and computing the dot product in that space, we can just compute a function $K(x_i, x_j)$ of the original data points.

This makes the QP efficiently solvable. To classify a test point x, we just need to compute $sign(\sum_j \alpha_j y_j K(x_j, x) + \rho)$.

Sum is just over the support vectors; other points have $\alpha_j = 0$.

Some advantages of SVMs

- Very good performance: though lately outshined by convolutional neural networks on some benchmarks (e.g., the MNIST digit recognition dataset) they often beat basically everything else.
- Theoretical guarantees about their generalization performance (accuracy for labeling test data) based on statistical learning theory.
- SVMs rely on convex optimization and do not get stuck in suboptimal local minima (neural networks have a big problem with these; similarly, decision trees rely on greedy search).
- Fairly robust to the curse of dimensionality → can effectively solve prediction problems with a large number of features.
- Flexible: can choose kernel to fit very complex decision boundaries.
- Will generally avoid overfitting with well-chosen parameters (but can certainly overfit for poorly chosen values).
- Classification of test points relies only on the support vectors → fast and memory efficient, especially when # of support vectors is small.