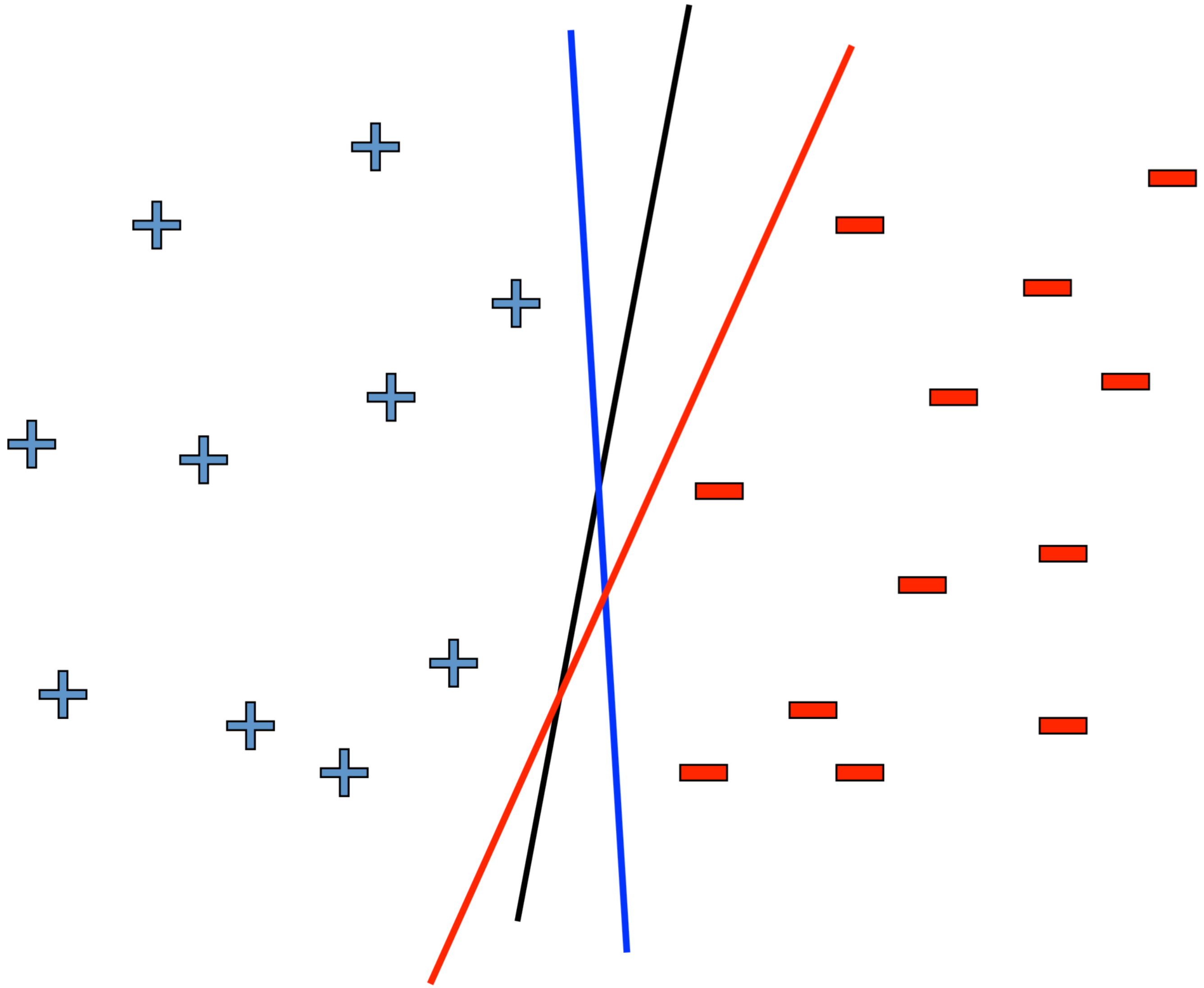
Machine Learning
Lecture 3

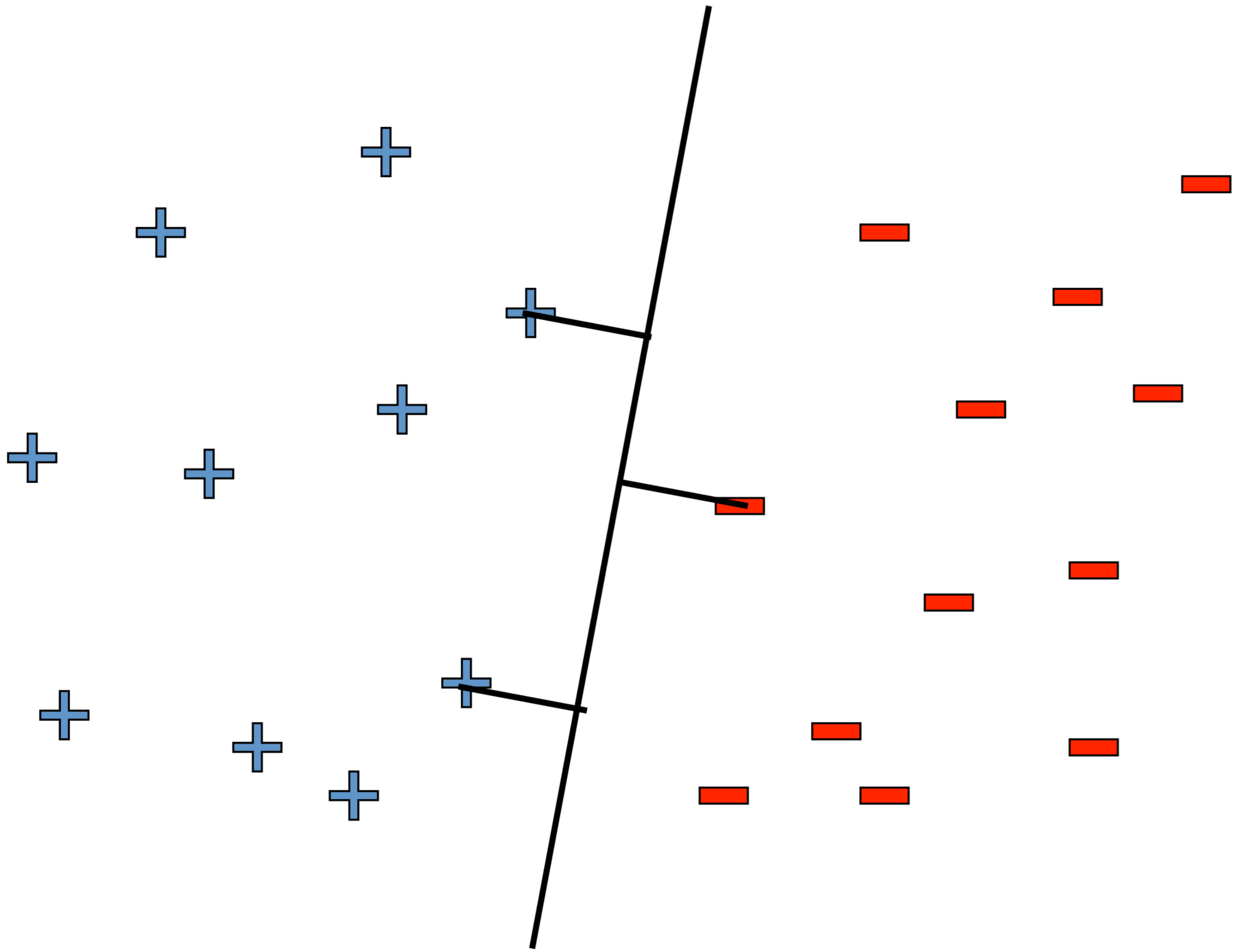
Yang Yang
Department of Computer Science & Engineering
Shanghai Jiao Tong University

Support Vector Machine

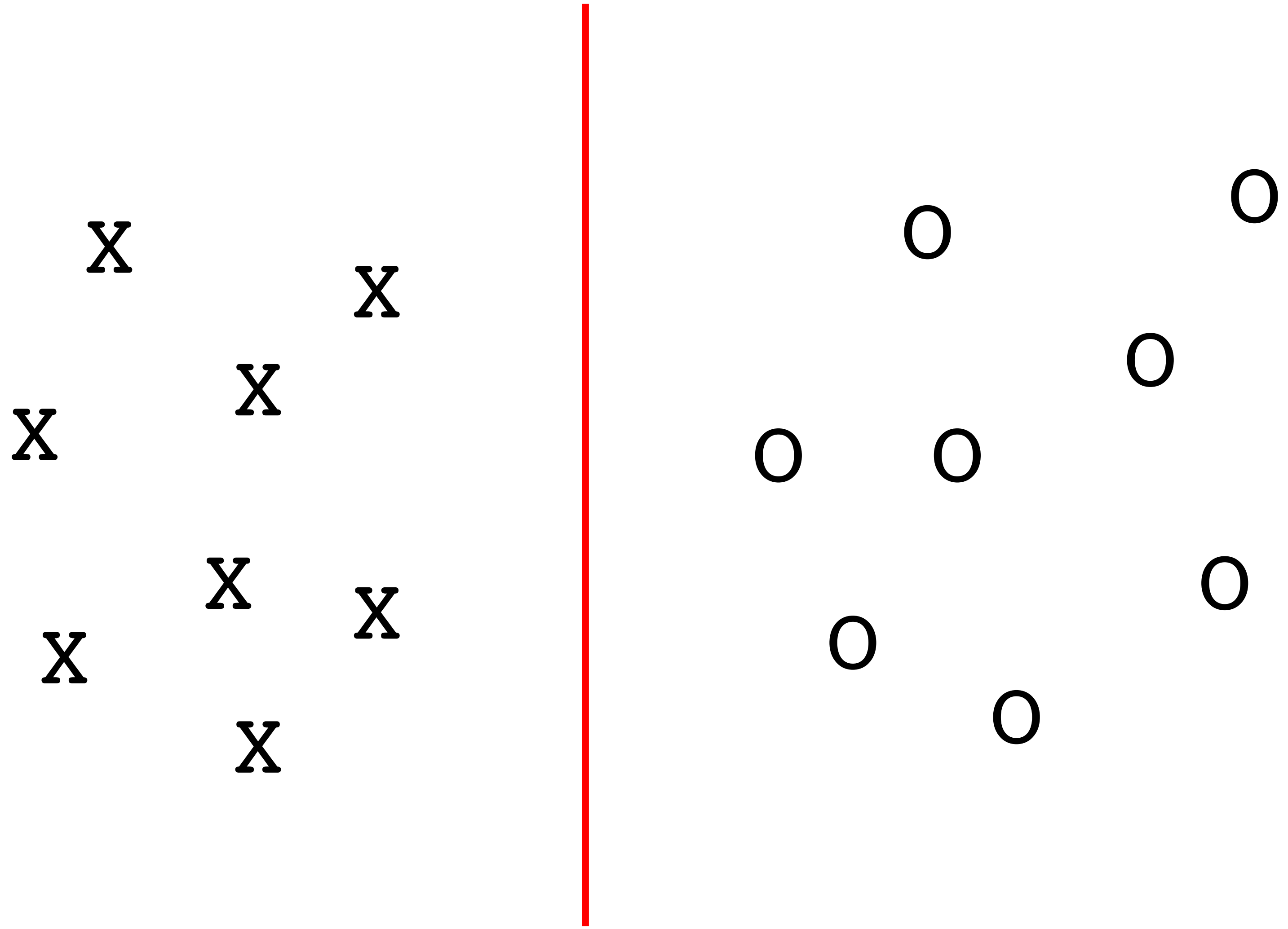
Linear classifiers–which line is better?



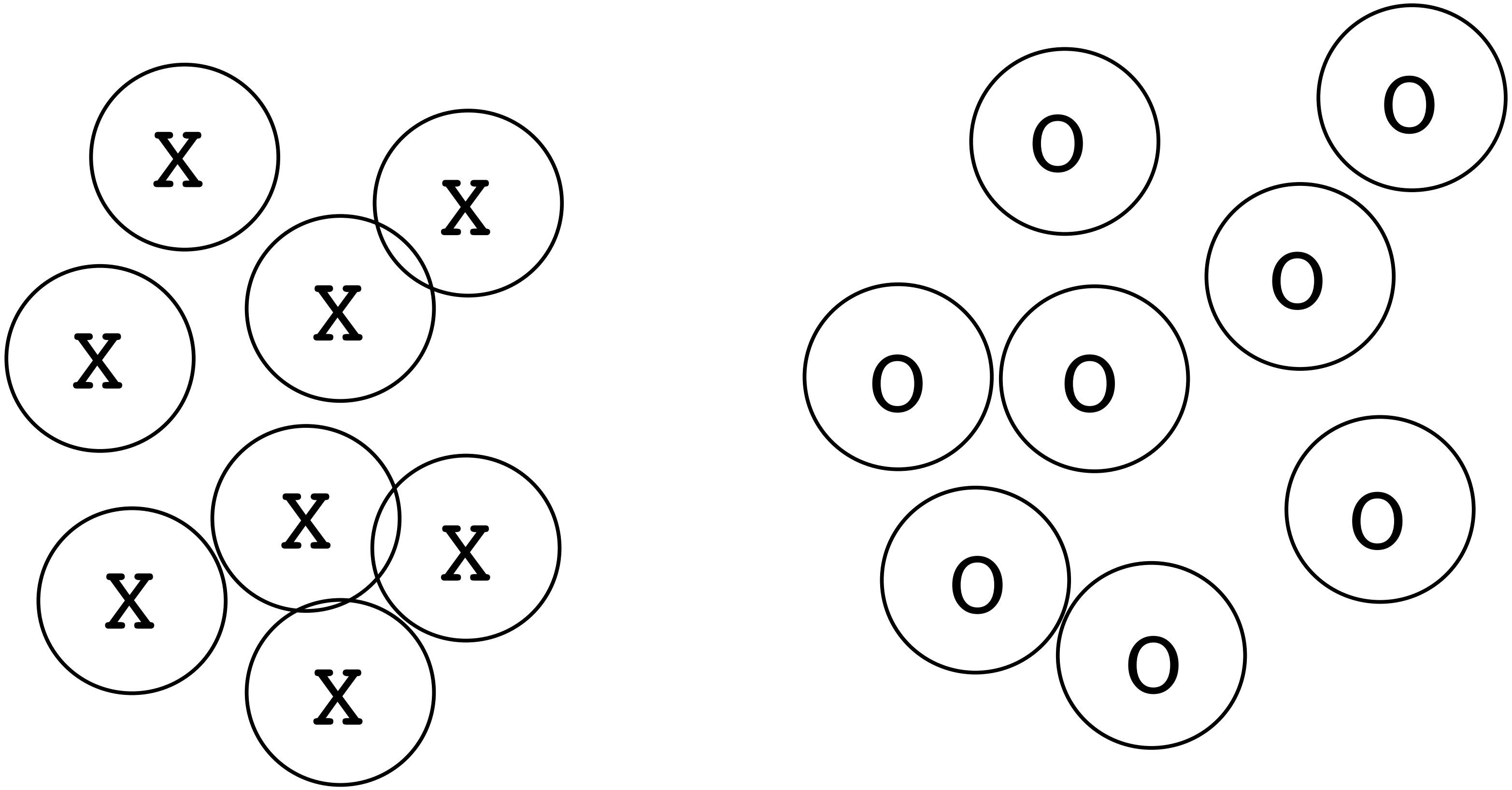
Pick the one with the largest margin!



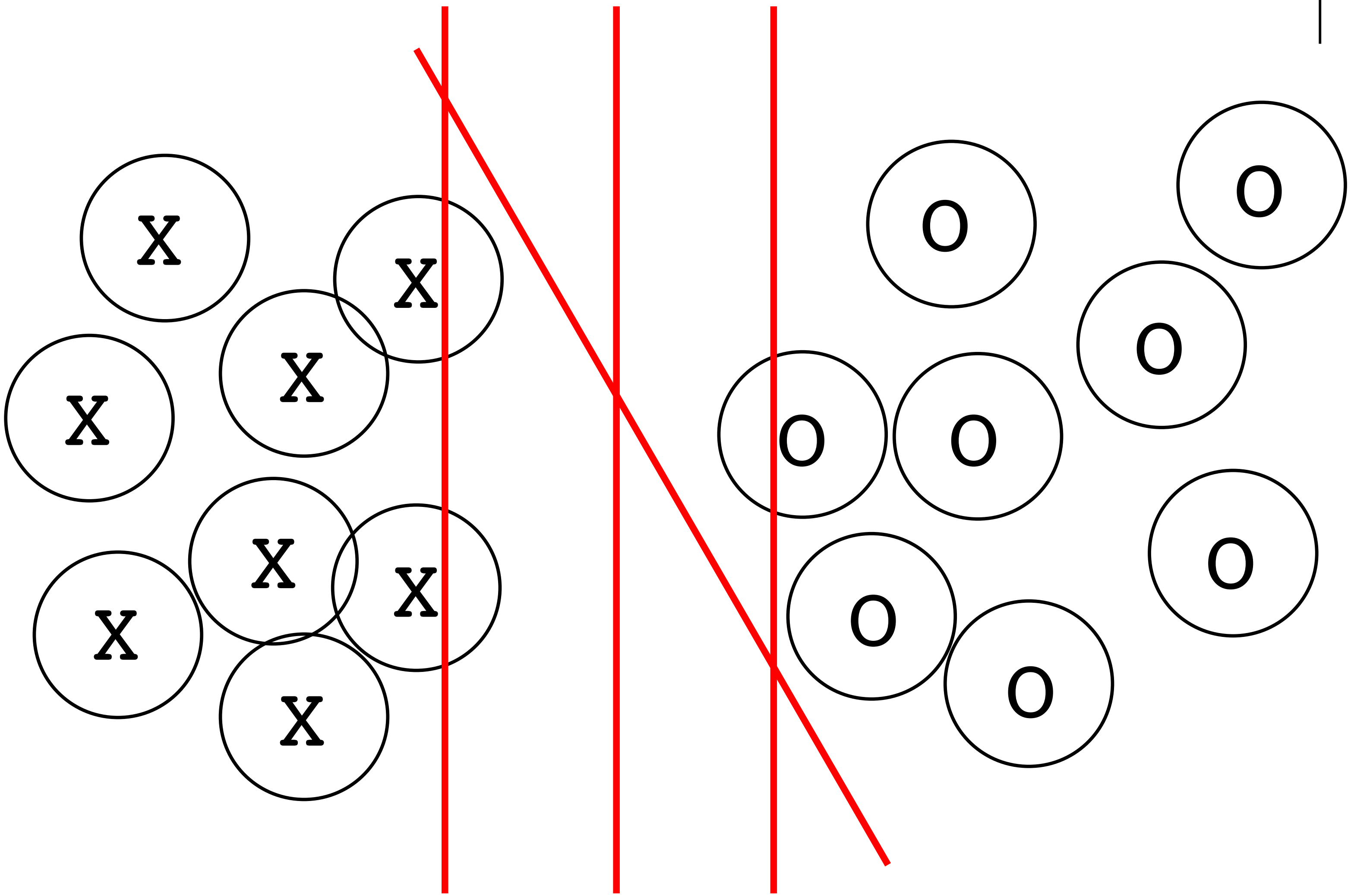
A “Good” Separator



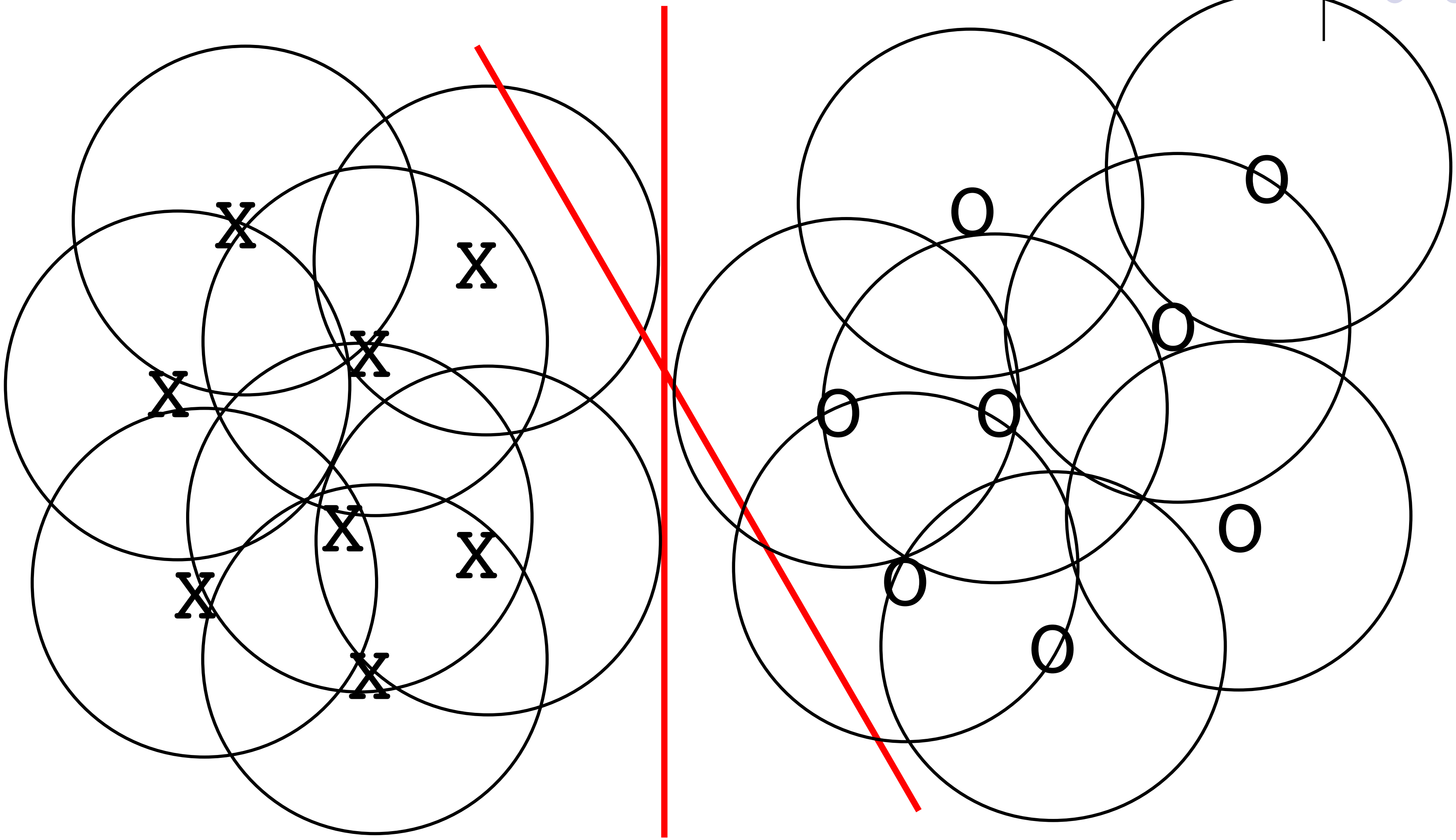
Noise in the Observations



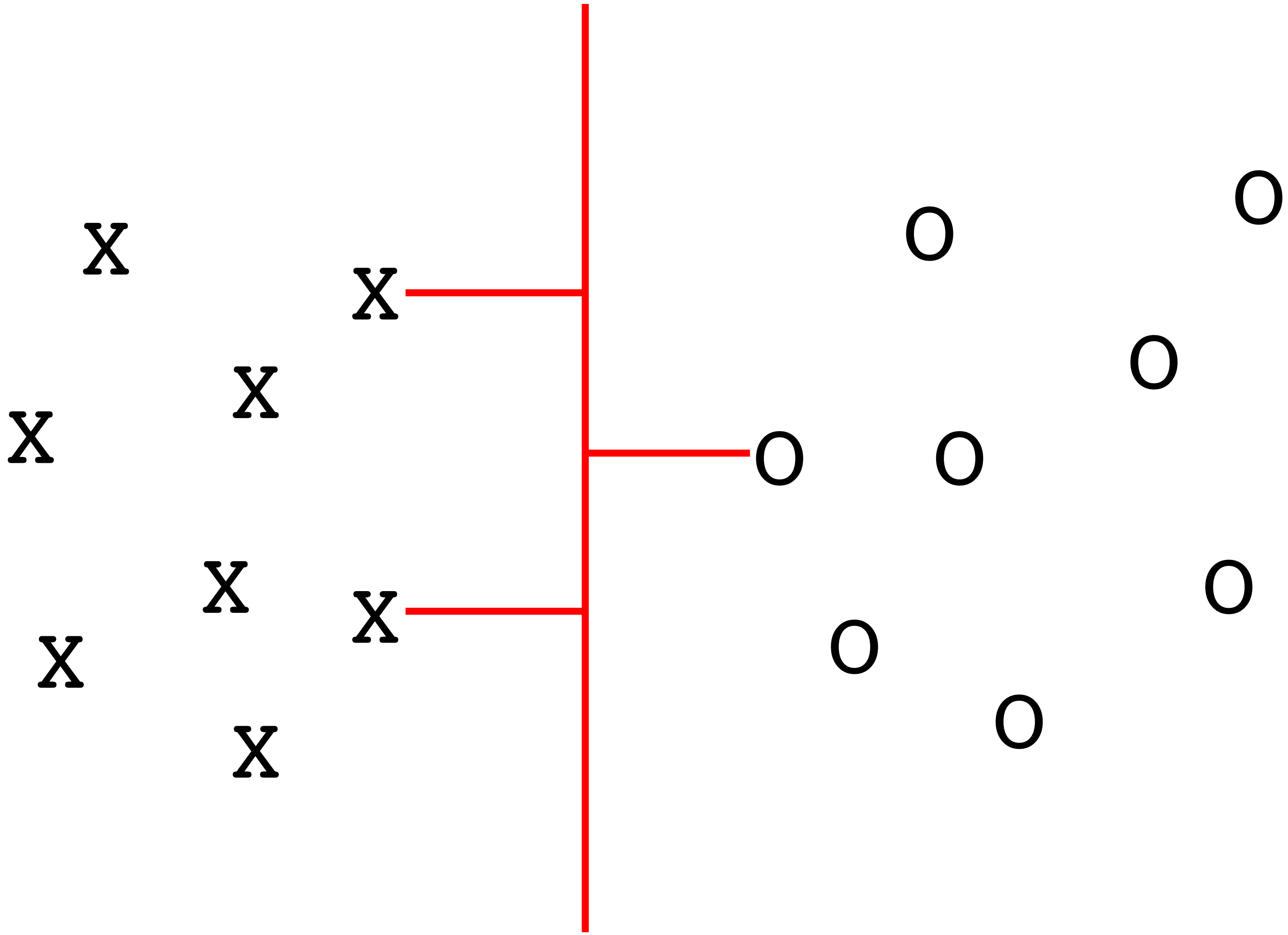
Ruling Out Some Separators



Lots of Noise



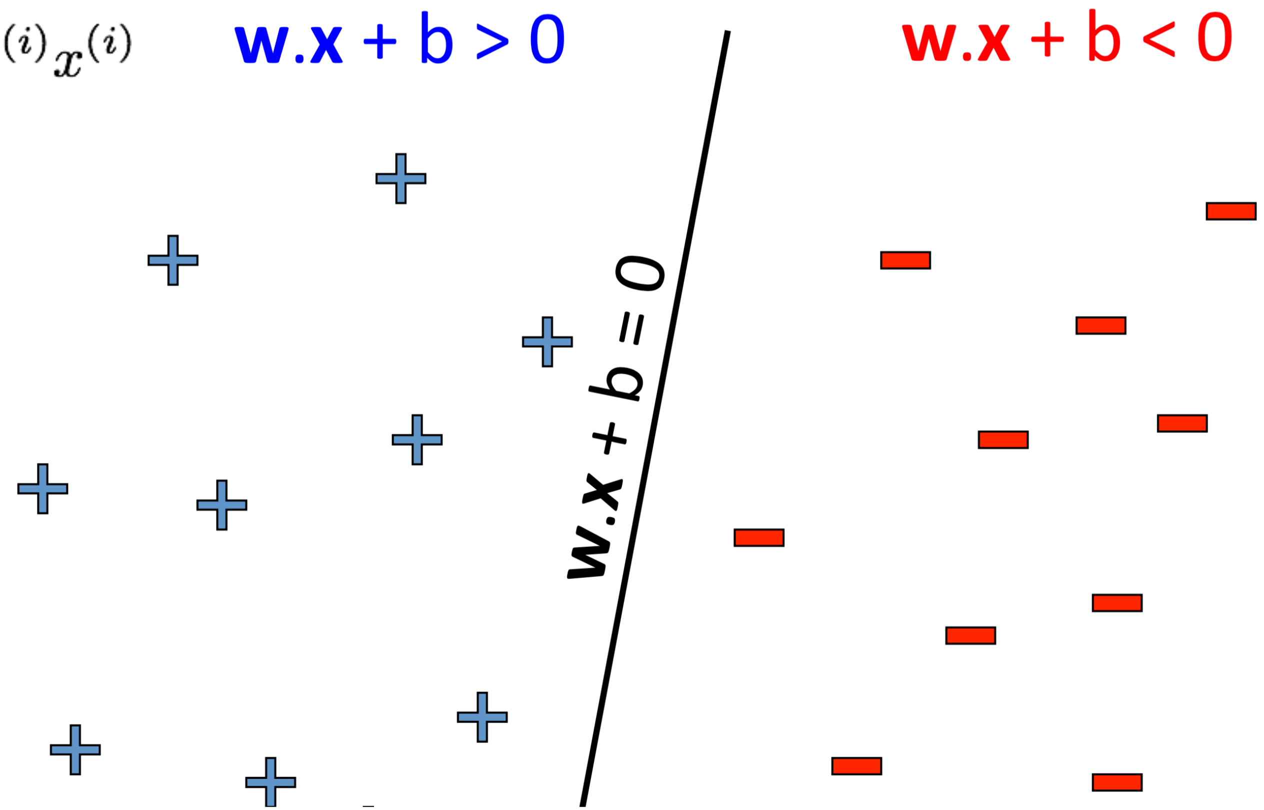
Maximizing the Margin



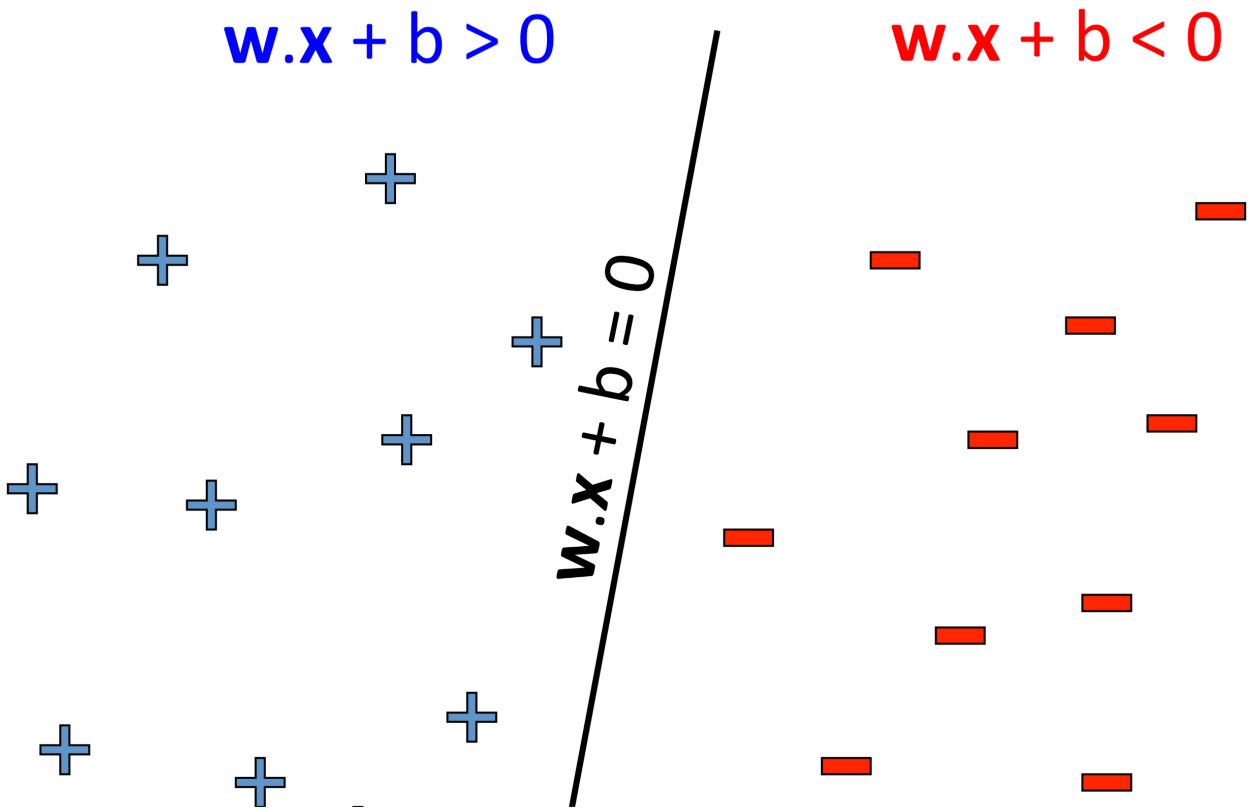
Parameterizing the decision boundary

$$w.x = \sum_{i=1}^{m} w^{(i)} x^{(i)}$$

m features

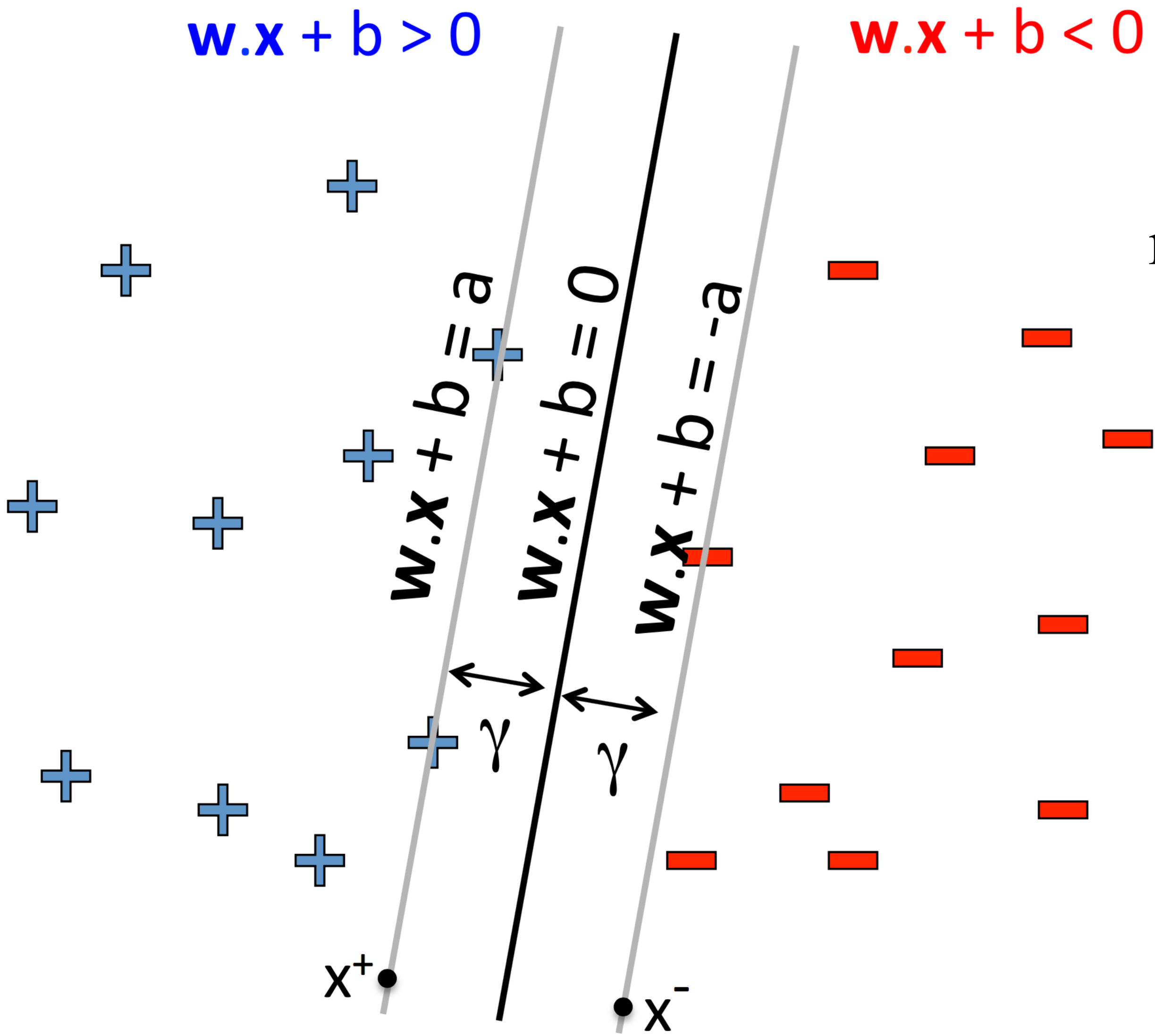


Parameterizing the decision boundary



"confidence" $= (w.x_j + b)y_j$
for j^{th} data point

Maximizing the margin



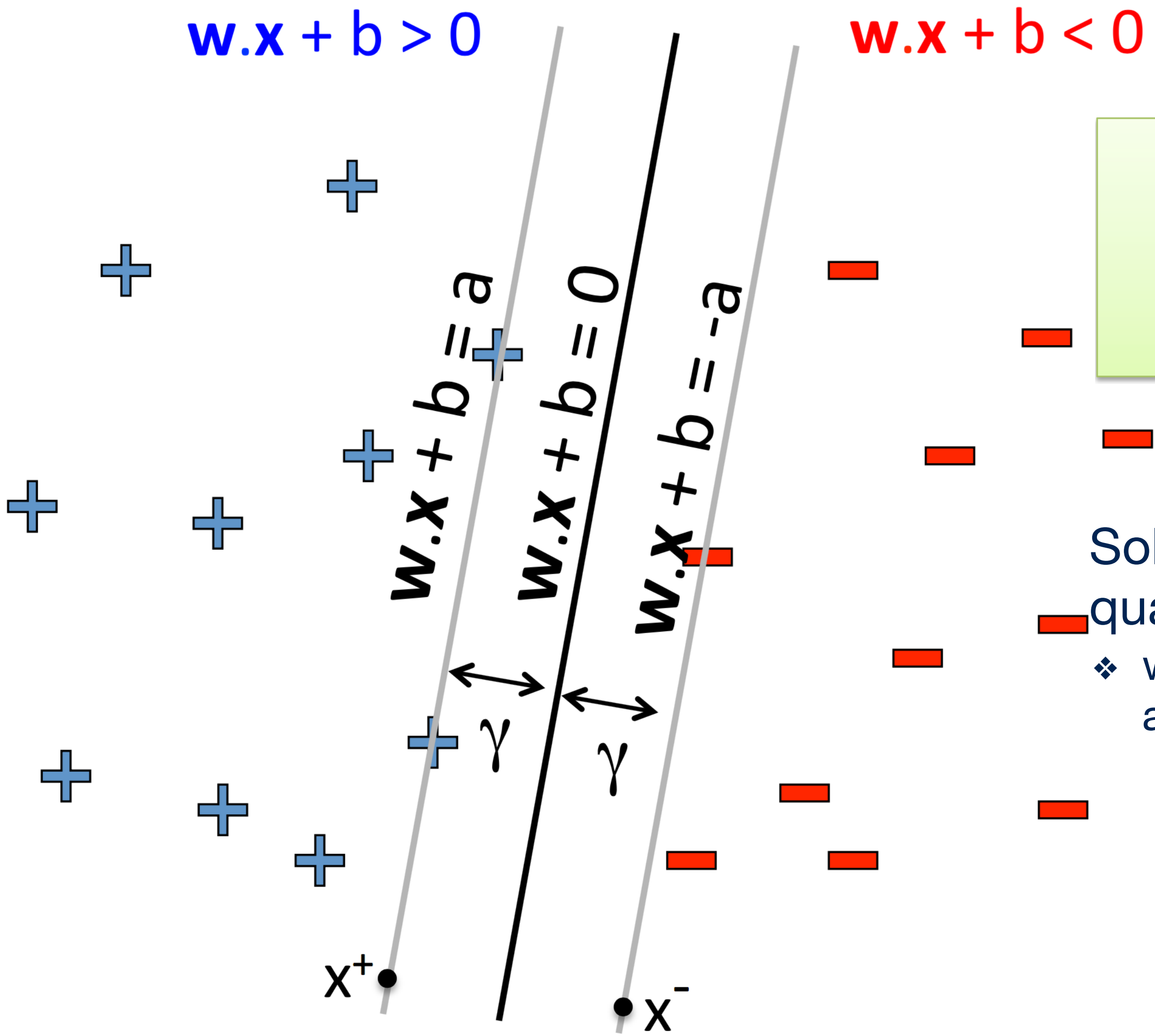
Distance between examples closest to the line/hyperplane

$$\text{margin} = 2\gamma = 2a/\|w\|$$

$$\max_{\mathbf{w},b} \; 2\gamma = 2a/\|w\|$$
$$\text{s.t. } (\mathbf{w}.\mathbf{x}_j+b)\, y_j \geq a \quad \forall j$$

Note: 'a' is arbitrary (can normalize equations by a)

Support Vector Machine



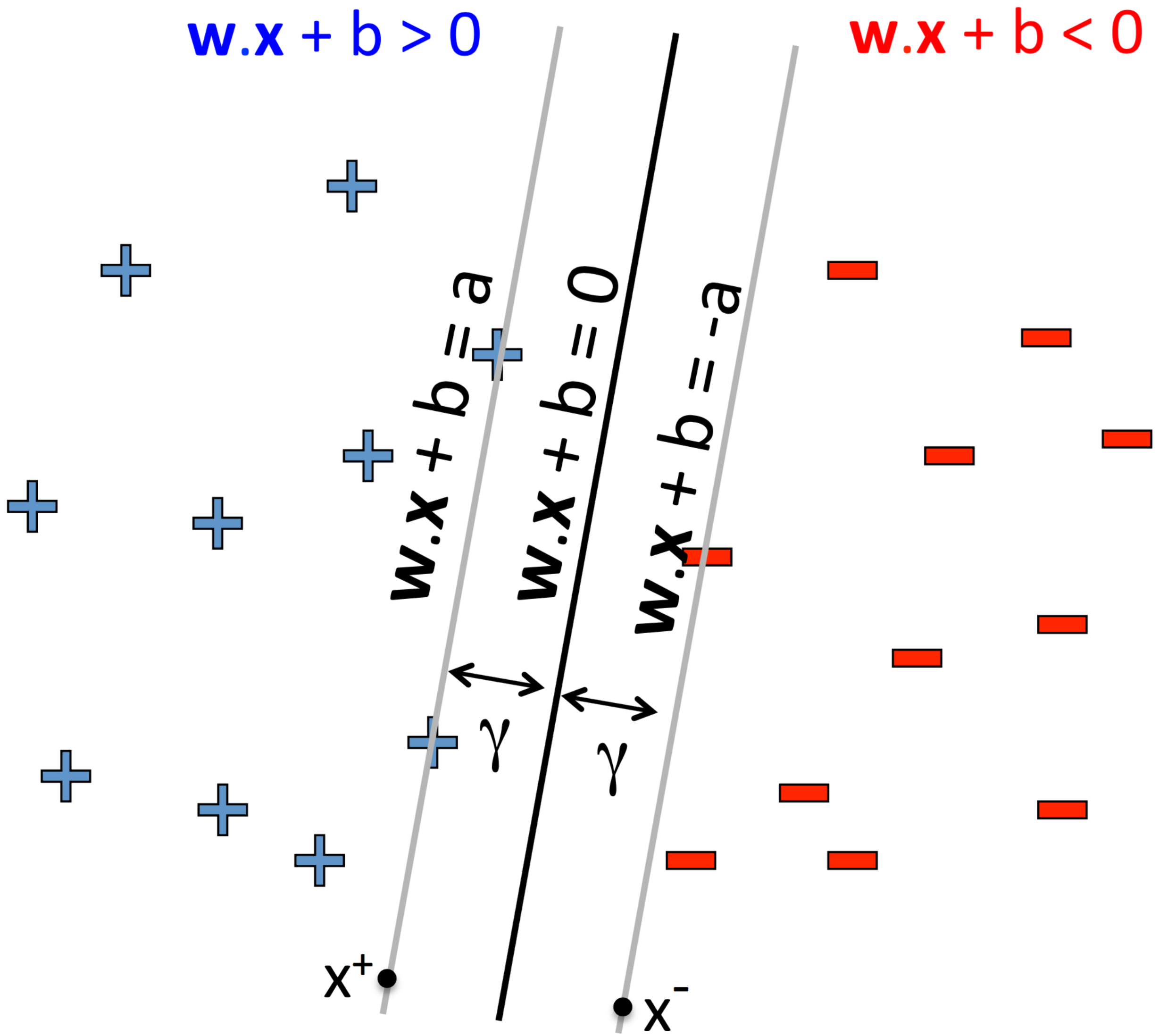
$$\min_{\mathbf{w},b} \; \mathbf{w}.\mathbf{w}$$

$$\text{s.t. } (\mathbf{w}.\mathbf{x}_j + b)\, y_j \geq 1 \quad \forall j$$

Solve efficiently by quadratic programming (QP)

- well-studied solution algorithms

Support Vector Machine



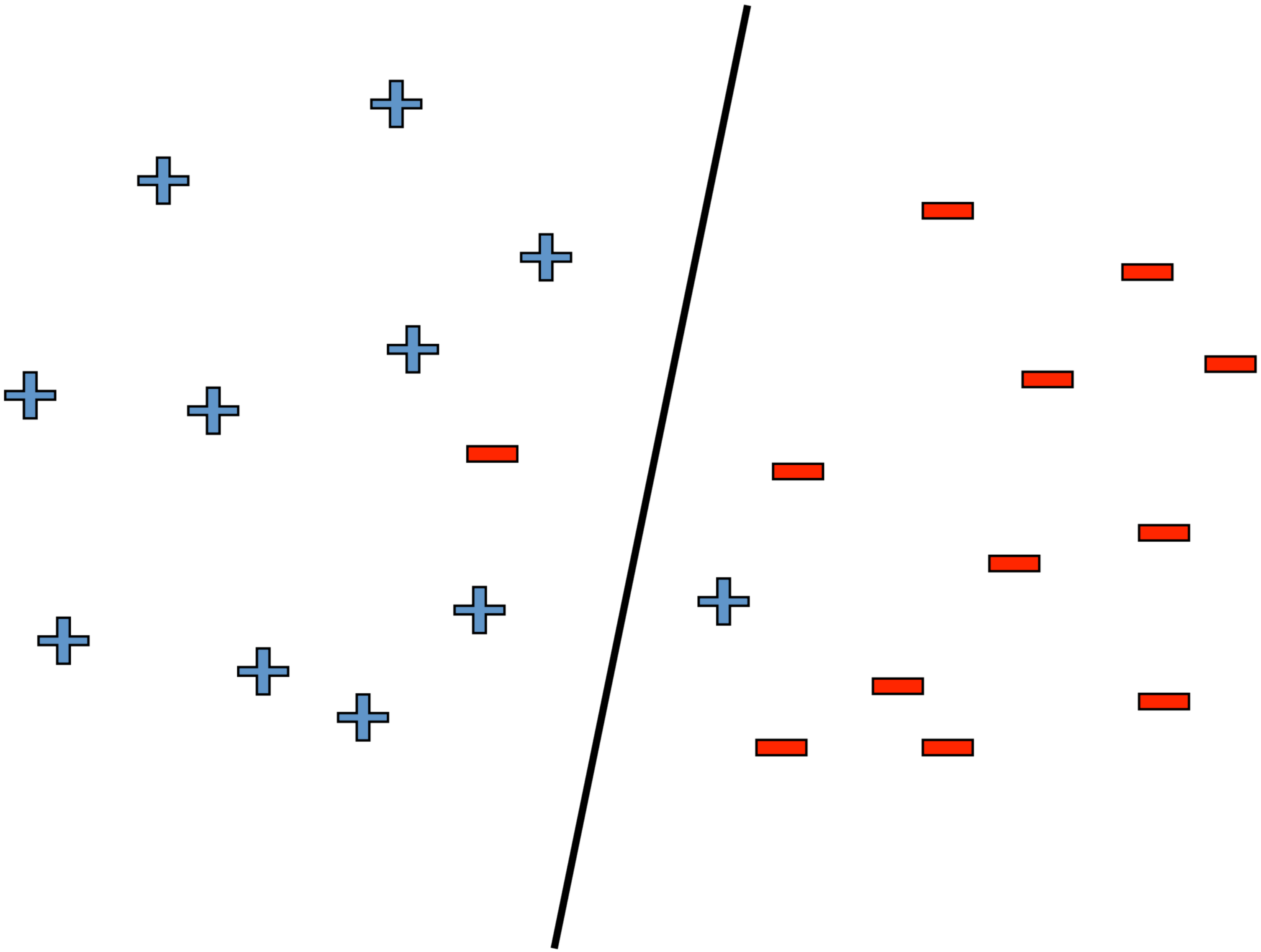
Linear hyperplane defined by "support vectors"
 i: $(\mathbf{w}.\mathbf{x}_i + b)\, y_i = 1$

Moving other points a little doesn't effect the decision boundary

Only need to store the support vectors to predict labels of new points

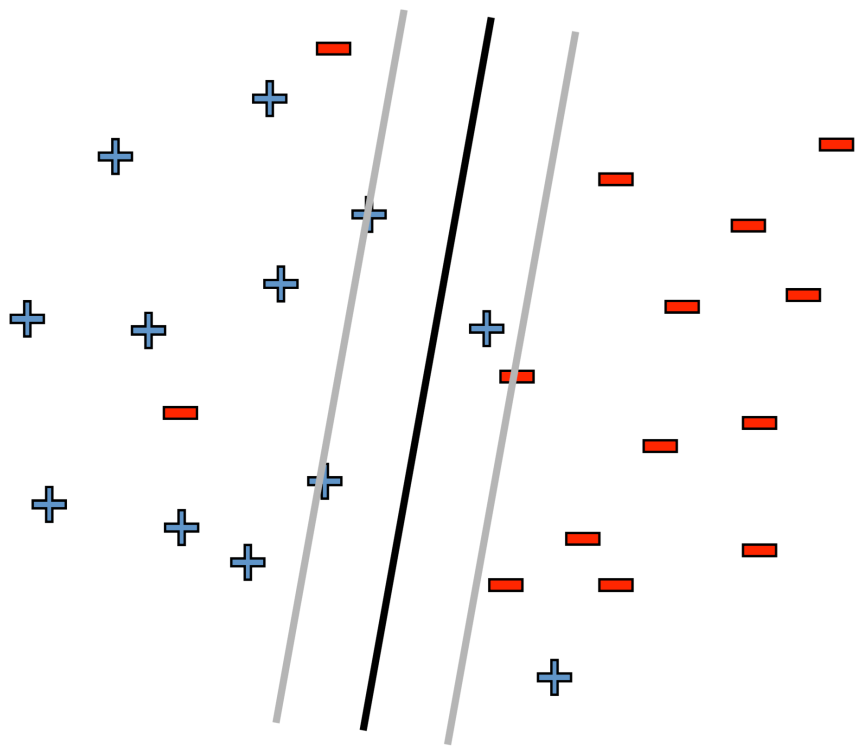
How many support vectors in linearly separable case?
 $\leq m+1$

What if data is not linearly separable?



What if data is still not linearly separable?

Allow “error” in classification



$$\min_{\mathbf{w},b} \ \mathbf{w}.\mathbf{w} + C \ \#\text{mistakes}$$
$$\text{s.t. } (\mathbf{w}.\mathbf{x}_j+b)\, y_j \geq 1 \quad \forall j$$

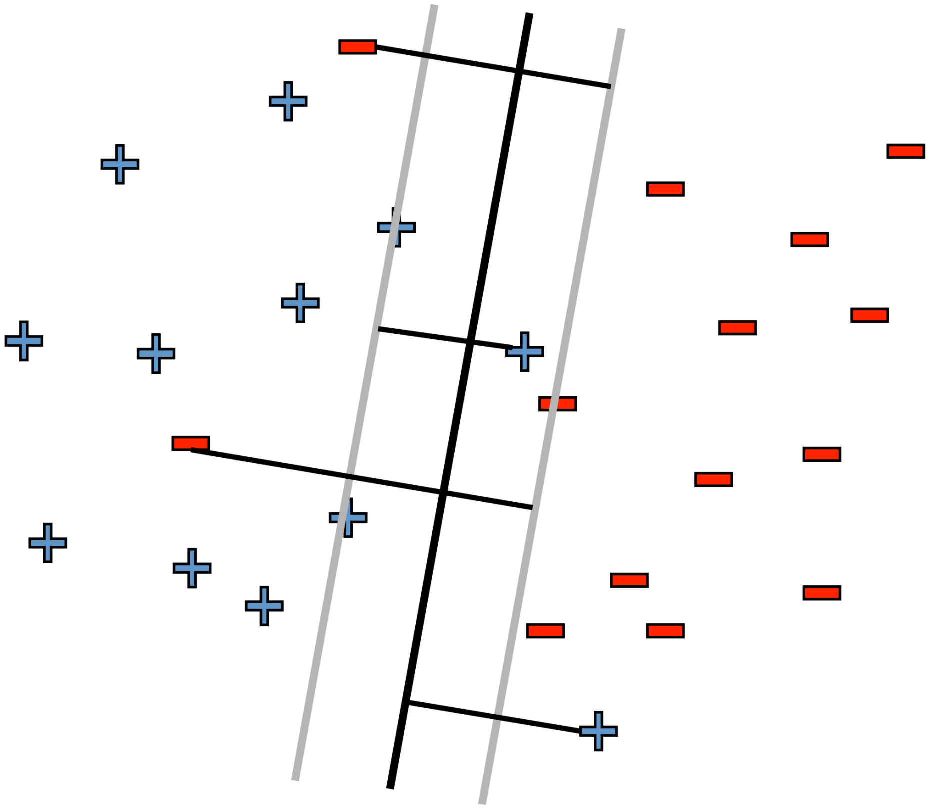
Maximize margin and minimize # mistakes on training data

C - tradeoff parameter

- Not convex
- 0/1 loss (doesn’t distinguish between near miss and bad mistake)

What if data is still not linearly separable?

Allow "error" in classification



Soft margin approach

$$\min_{\mathbf{w},b,\xi_j} \ \mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$

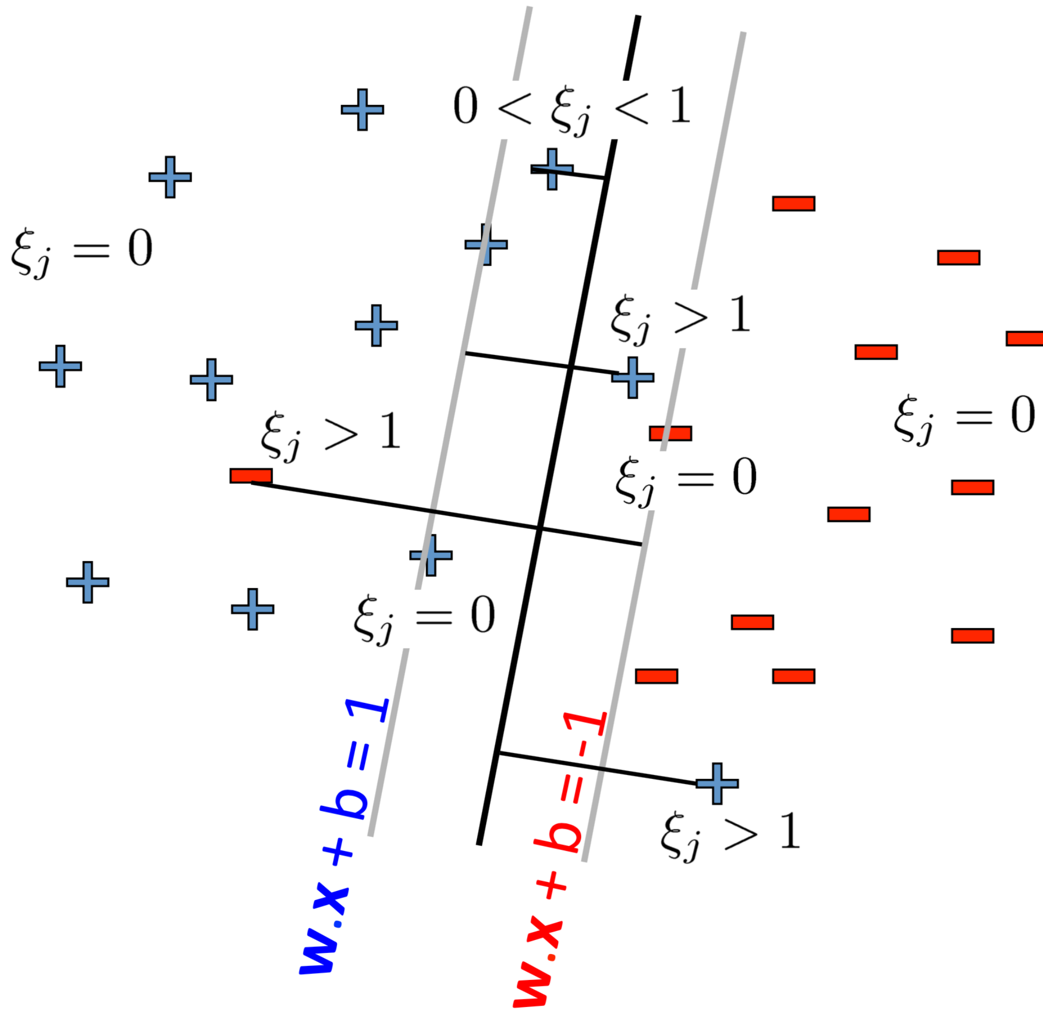
$$\text{s.t. } (\mathbf{w}.\mathbf{x}_j + b)\, y_j \geq 1 - \xi_j \quad \forall j$$

$$\xi_j \geq 0 \quad \forall j$$

ξ_j - "slack" variables
(>1 if x_j misclassifed)
pay linear penalty if mistake

C - tradeoff parameter(chosen by cross-validation)
convex!

Soft-margin SVM



- Soften the constraints:

  ($\mathbf{w}.\mathbf{x}_j+b$) $y_j \geq 1-\xi_j \;\forall j$

  $\xi_j \geq 0 \quad\quad \forall j$

  Penalty for misclassifying:

  $C\,\xi_j$

  How do we recover hard margin SVM?

  Set $C = \infty$

Slack variables as Hinge loss

Regularized loss

$$\xi_j = \text{loss}(f(x_j), y_j)$$

$$f(x_j) = \text{sgn}(\mathbf{w} \cdot \mathrm{x_j} + \mathrm{b})$$

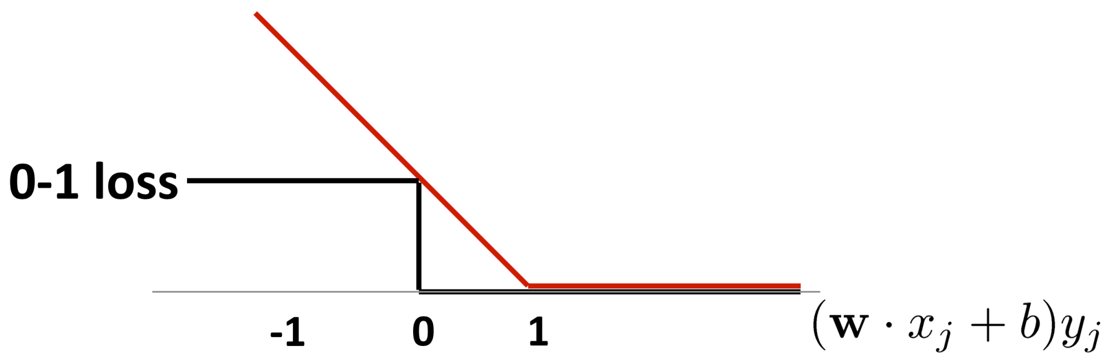
$$\min_{\mathbf{w},b,\xi_j} \mathbf{w}.\mathbf{w} + C \sum_j \xi_j$$

$$\text{s.t. } (\mathbf{w}.\mathbf{x}_j + b)\, y_j \geq 1 - \xi_j \quad \forall j$$

$$\xi_j \geq 0 \quad \forall j$$

$$\xi_j = (1 - (\mathbf{w} \cdot x_j + b) y_j))_+$$

Hinge loss



Hinge Loss

$$\underbrace{\text{argmin}_{\{w,b\}} w^t w}_{\text{regularization}} + \underbrace{\lambda \sum_{1}^{m} \max(1 - y_i(w^t x_i + b), 0)}_{\text{Loss: hinge loss}}$$

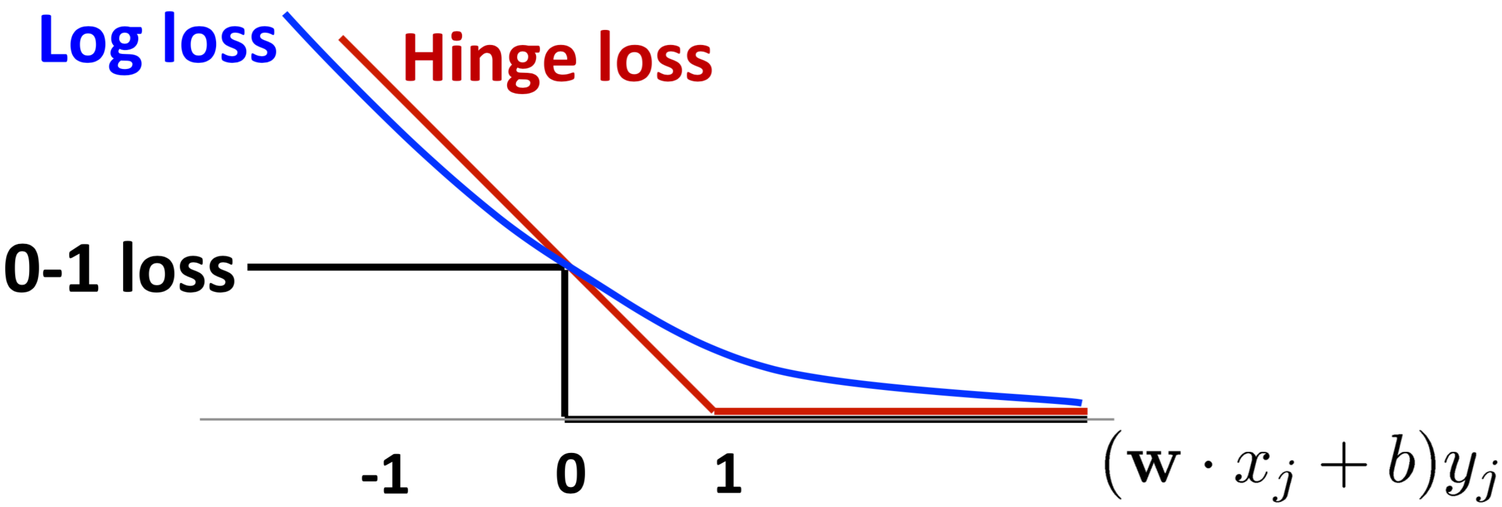
SVM vs. Logistic Regression

SVM : Hinge loss

$$\text{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$

Logistic Regression : Log loss

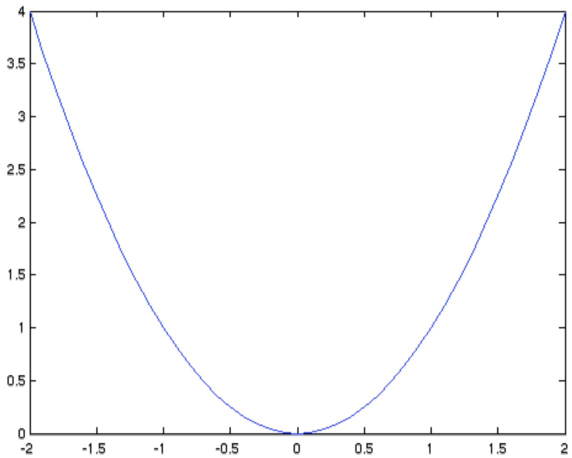
$$\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b)y_j})$$



Constrained Optimization

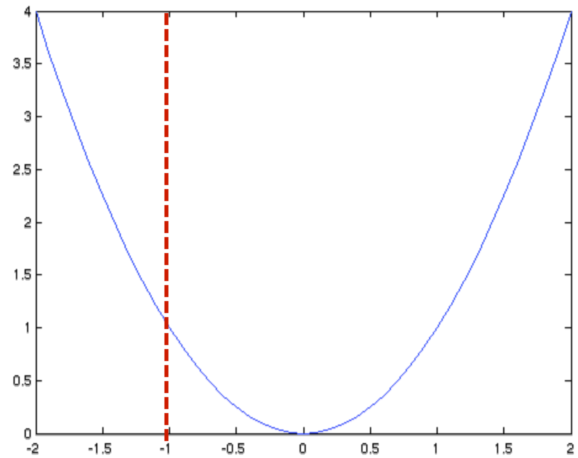
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq b$$

$$\min_x \; x^2$$



$$x^* = 0$$

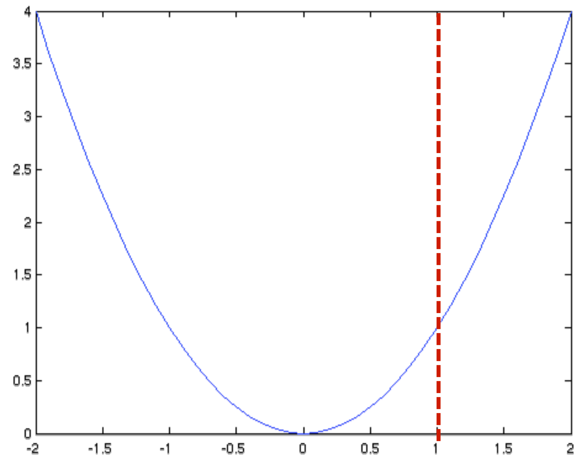
$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq -1$$



$$x^* = 0$$

Constraint inactive

$$\min_x \; x^2$$
$$\text{s.t.} \quad x \geq 1$$



$$x^* = 1$$

Constraint active

Digression to Lagrangian Duality

- The Primal Problem

Primal:

$$\begin{aligned} &\min_w \quad f(w) \\ &\text{s.t.} \quad g_i(w) \le 0, \quad i = 1, \ldots, k \\ &\qquad\quad h_i(w) = 0, \quad i = 1, \ldots, l \end{aligned}$$

The generalized Lagrangian:

$$\mathcal{L}(w, \alpha, \beta) = f(w) + \sum_{i=1}^{k} \alpha_i g_i(w) + \sum_{i=1}^{l} \beta_i h_i(w)$$

the α's ($\alpha_i \geq 0$) and β's are called the Lagarangian multipliers

Lemma:

$$\max_{\alpha, \beta, \alpha_i \ge 0} \mathcal{L}(w, \alpha, \beta) = \begin{cases} f(w) & \text{if } w \text{ satisfies primal constraints} \\ \infty & \text{o/w} \end{cases}$$

A re-written Primal:

$$\min_w \max_{\alpha, \beta, \alpha_i \ge 0} \mathcal{L}(w, \alpha, \beta)$$

Lagrangian Duality, cont.

- Recall the Primal Problem:

$$\min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta)$$

- The Dual Problem:

$$\max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta)$$

- **Theorem (weak duality):**

$$d^* = \max_{\alpha,\beta,\alpha_i \geq 0} \min_w \mathcal{L}(w,\alpha,\beta) \leq \min_w \max_{\alpha,\beta,\alpha_i \geq 0} \mathcal{L}(w,\alpha,\beta) = p^*$$

- **Theorem (strong duality):**

Iff there exist a saddle point of $\mathcal{L}(w,\alpha,\beta)$, we have

$$d^* = p^*$$

The KKT conditions

- If there exists some saddle point of $\mathcal{L}$, then the saddle point satisfies the following "Karush-Kuhn-Tucker" (KKT) conditions:

$$
\begin{aligned}
\frac{\partial}{\partial w_i}\mathcal{L}(w,\alpha,\beta) &= 0, \quad i=1,\dots,k \\
\frac{\partial}{\partial \beta_i}\mathcal{L}(w,\alpha,\beta) &= 0, \quad i=1,\dots,l \\
\alpha_i g_i(w) &= 0, \quad i=1,\dots,m \\
g_i(w) &\le 0, \quad i=1,\dots,m \\
\alpha_i &\ge 0, \quad i=1,\dots,m
\end{aligned}
$$

  - **Theorem**: If w^*, α^* and β^* satisfy the KKT condition, then it is also a solution to the primal and the dual problems.

Solving optimal margin classifier

- Recall our opt problem:

$$\max_{w,b} \quad \frac{1}{\|w\|}$$
$$\text{s.t} \quad y_i(w^T x_i + b) \geq 1, \quad \forall i$$

- This is equivalent to

$$\min_{w,b} \quad \frac{1}{2} w^T w$$
$$\text{s.t} \quad 1 - y_i(w^T x_i + b) \leq 0, \quad \forall i \qquad (*)$$

- Write the Lagrangian:

$$\mathcal{L}(w,b,\alpha) = \frac{1}{2} w^T w - \sum_{i=1}^{m} \alpha_i \left[y_i (w^T x_i + b) - 1 \right]$$

  - Recall that (*) can be reformulated as $\min_{w,b} \max_{\alpha_i \geq 0} \mathcal{L}(w,b,\alpha)$
    Now we solve its **dual problem**: $\max_{\alpha_i \geq 0} \min_{w,b} \mathcal{L}(w,b,\alpha)$

The Dual Problem

$$\mathcal{L}(w,b,\alpha)=\frac{1}{2}w^T w-\sum_{i=1}^{m}\alpha_i\left[y_i(w^T x_i+b)-1\right]$$

$$\max_{\alpha_i\geq 0}\min_{w,b}\mathcal{L}(w,b,\alpha)$$

- We minimize $\mathcal{L}$ with respect to w and b first:

$$\nabla_w\mathcal{L}(w,b,\alpha)=w-\sum_{i=1}^{m}\alpha_i y_i x_i=0, \qquad (*)$$

$$\nabla_b\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i y_i=0, \qquad (**)$$

Note that **(*)** implies:

$$w=\sum_{i=1}^{m}\alpha_i y_i x_i \qquad (***)$$

- Plug (***) back to $\mathcal{L}$, and using (**), we have:

$$\mathcal{L}(w,b,\alpha)=\sum_{i=1}^{m}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{m}\alpha_i\alpha_j y_i y_j(\mathbf{x}_i^T\mathbf{x}_j)$$

The Dual problem, cont.

- Now we have the following dual opt problem:

$$\max_\alpha \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

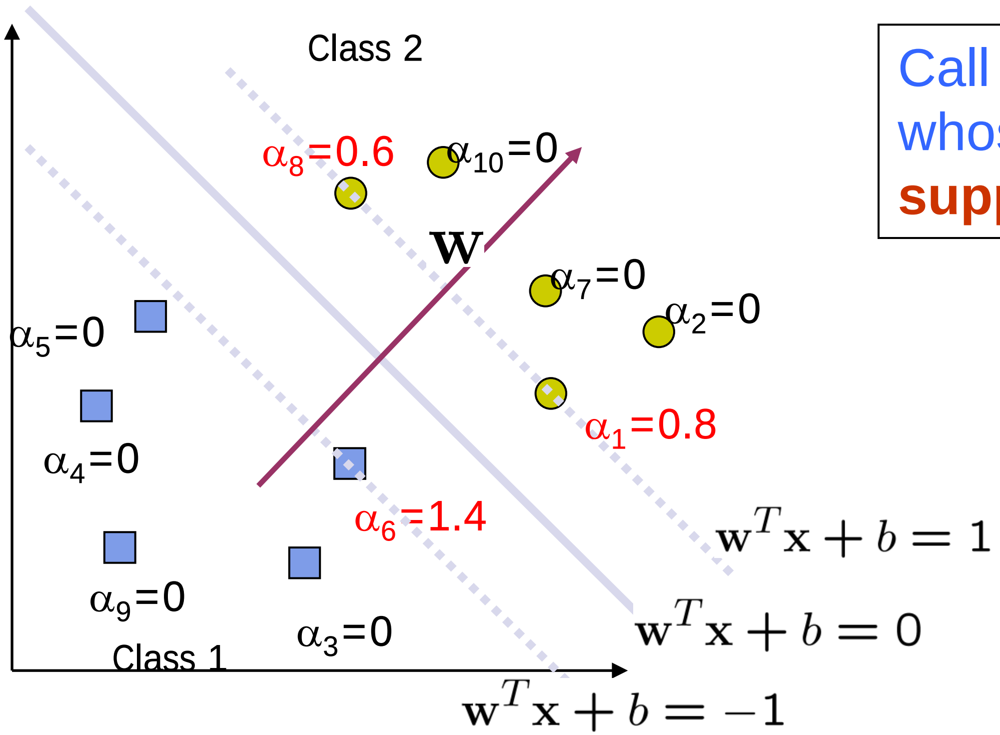
- This is, (again,) a **quadratic programming** problem.
  - A global maximum of α_i can always be found.
  - But what's the big deal??
  - Note two things:

1. *w* can be recovered by $w = \sum_{i=1}^{m} \alpha_i y_i \mathbf{x}_i$ See next …
2. The "kernel" $\mathbf{x}_i^T \mathbf{x}_j$ More later …

I. Support vectors

- Note the KKT condition --- only a few α_i's can be nonzero!!

$$\alpha_i g_i(w) = 0, \quad i = 1, \ldots, m$$



Call the training data points whose α_i's are nonzero the **support vectors** (SV)

Support vector machines

- Once we have the Lagrange multipliers $\{\alpha_i\}$, we can reconstruct the parameter vector w as a weighted combination of the training examples:

$$w = \sum_{i \in SV} \alpha_i y_i \mathbf{x}_i$$

- For testing with a new data $\boldsymbol{z}$
  - Compute

$$w^T z + b = \sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b$$

    and classify $\boldsymbol{z}$ as class 1 if the sum is positive, and class 2 otherwise
  - Note: w need not be formed explicitly

Interpretation of support vector machines

- The optimal $\boldsymbol{w}$ is a linear combination of a small number of data points. This “sparse” representation can be viewed as data compression as in the construction of kNN classifier

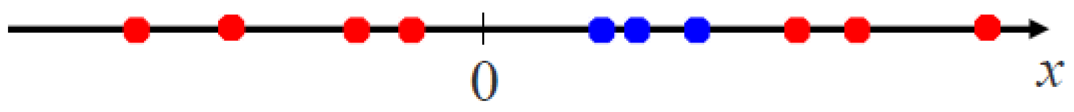
- To compute the weights $\{\alpha_i\}$, and to use support vector machines we need to specify only the inner products (or kernel) between the examples $\mathbf{x}_i^T \mathbf{x}_j$

- We make decisions by comparing each new example $\boldsymbol{z}$ with only the support vectors:

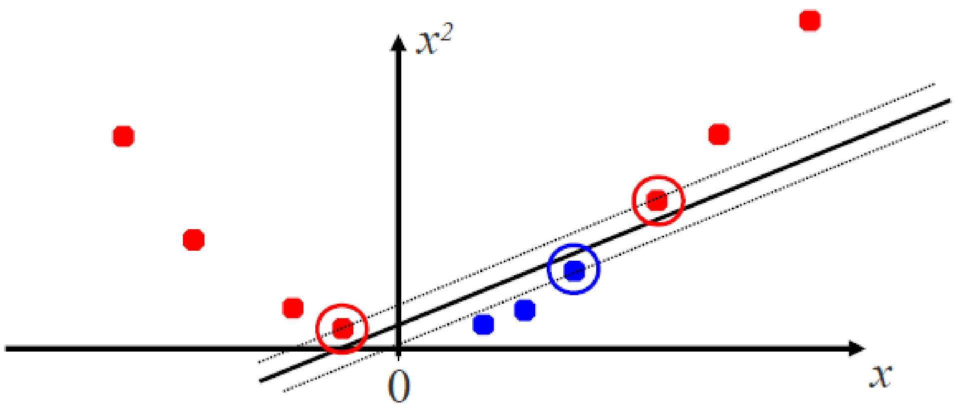
$$y^* = \operatorname{sign}\left(\sum_{i \in SV} \alpha_i y_i \left(\mathbf{x}_i^T z \right) + b \right)$$

II. The Kernel Trick

- Is this data linearly-separable?



- How about a quadratic mapping $\phi(x_i)$?



II. The Kernel Trick

- Recall the SVM optimization problem

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- The data points only appear as inner product
- As long as we can calculate the inner product in the feature space, we do not need the mapping explicitly
- Many common geometric operations (angles, distances) can be expressed by inner products
- Define the kernel function *K* by $K(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^T \phi(\mathbf{x}_j)$

II. The Kernel Trick

- Computation depends on feature space
  - Bad if its dimension is much larger than input space

$$\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j)$$

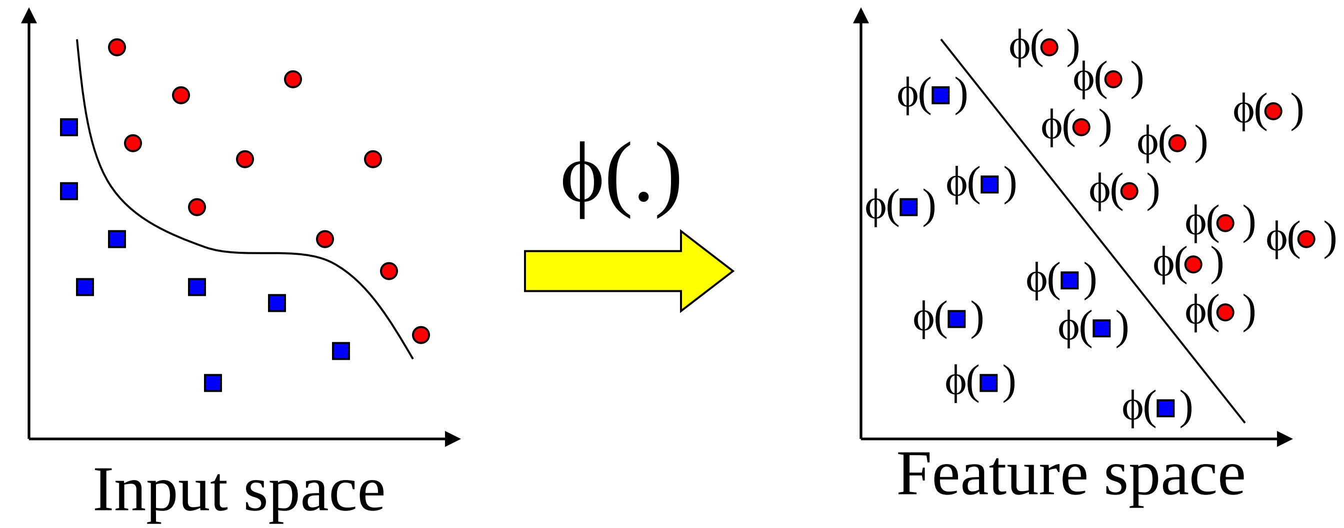
$$\text{s.t.} \quad \alpha_i \geq 0, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

Where $K(x_i,x_j) = \phi(x_i)^t \phi(x_j)$

$$y^*(z) = \operatorname{sign}\left(\sum_{i \in SV} \alpha_i y_i K(\mathbf{x}_i, z) + b \right)$$

Transforming the Data



Note: feature space is of higher dimension than the input space in practice

- Computation in the feature space can be costly because it is high dimensional
  - The feature space is typically infinite-dimensional!
- The kernel trick comes to rescue

An Example for feature mapping and kernels

- Consider an input $\mathbf{x}=[x_1,x_2]$
- Suppose $\phi(.)$ is given as follows

$$\phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) = 1, \sqrt{2}x_1, \sqrt{2}x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2$$

- An inner product in the feature space is

$$\left\langle \phi\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right), \phi\left(\begin{bmatrix} x_1' \\ x_2' \end{bmatrix}\right) \right\rangle =$$

- So, if we define the **kernel function** as follows, there is no need to carry out $\phi(.)$ explicitly

$$K(\mathbf{x},\mathbf{x}') = \left(1 + \mathbf{x}^T\mathbf{x}'\right)^2$$

More examples of kernel functions

- Linear kernel (we've seen it)

$$K(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

- Polynomial kernel (we just saw an example)

$$K(\mathbf{x}, \mathbf{x}') = \left(1 + \mathbf{x}^T \mathbf{x}'\right)^p$$

where p = 2, 3, … To get the feature vectors we concatenate all pth order polynomial terms of the components of x (weighted appropriately)

- Radial basis kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right)$$

In this case the feature space consists of functions and results in a non-parametric classifier.

The essence of kernel

- Feature mapping, but “without paying a cost”
  - E.g., polynomial kernel

$$K(x, z) = (x^T z + c)^d$$

  - How many dimensions we’ve got in the new space?
  - How many operations it takes to compute K()?

- Kernel design, any principle?
  - K(x,z) can be thought of as a similarity function between x and z
  - This intuition can be well reflected in the following “Gaussian” function (Similarly one can easily come up with other K() in the same spirit)

$$K(x, z) = \exp\left(-\frac{\|x - z\|^2}{2\sigma^2}\right)$$

  - Is this necessarily lead to a “legal” kernel? (in the above particular case, K() is a legal one, do you know how many dimension $\phi(x)$ is?

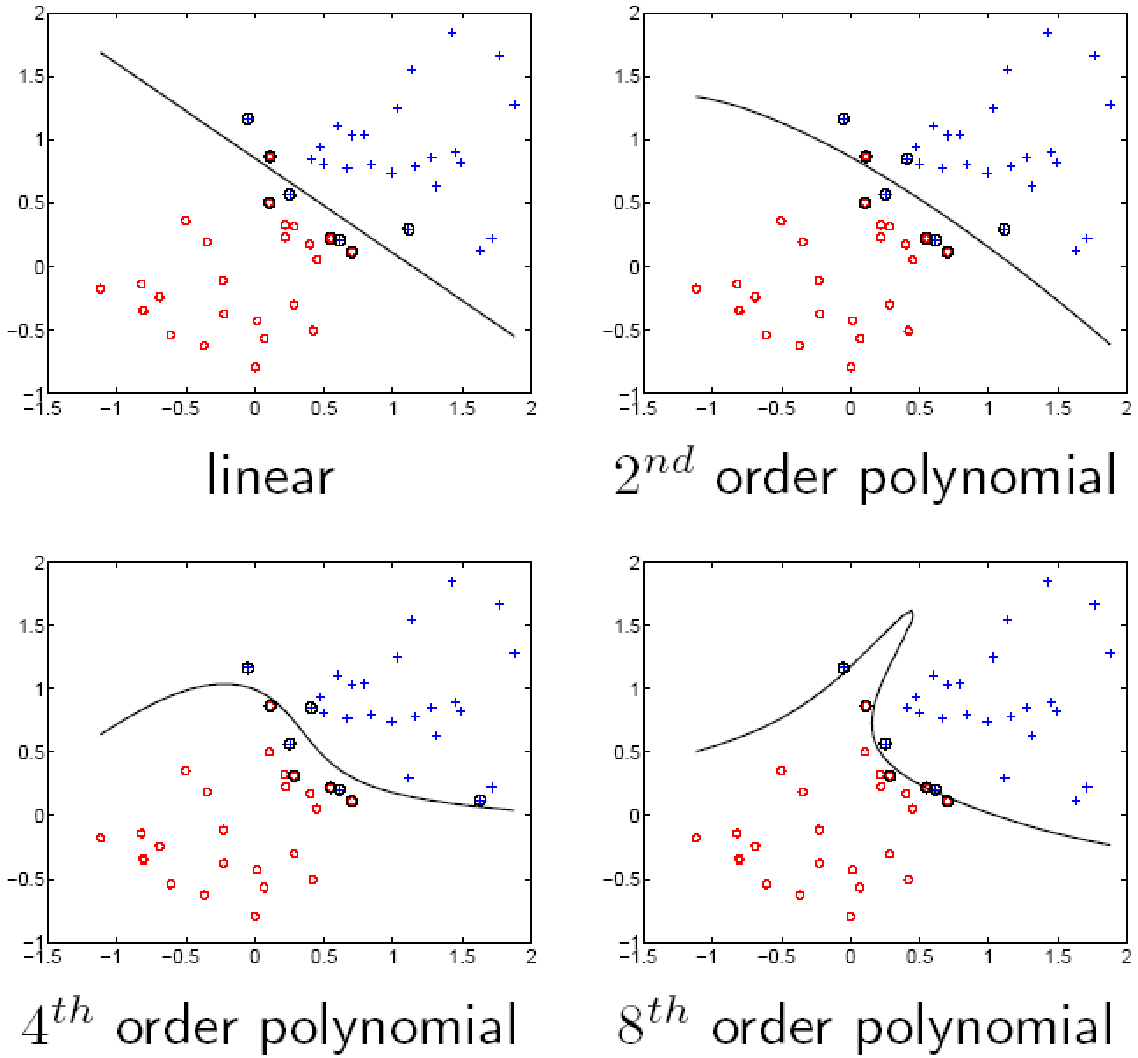
Kernel matrix

- Suppose for now that K is indeed a valid kernel corresponding to some feature mapping ϕ, then for $x_1, \ldots, x_m$, we can compute an $m \times m$ matrix $K = \{K_{i,j}\}$, where $K_{i,j} = \phi(x_i)^T \phi(x_j)$
- This is called a kernel matrix!
- Now, if a kernel function is indeed a valid kernel, and its elements are dot-product in the transformed feature space, it must satisfy:
  - Symmetry $K=K^T$

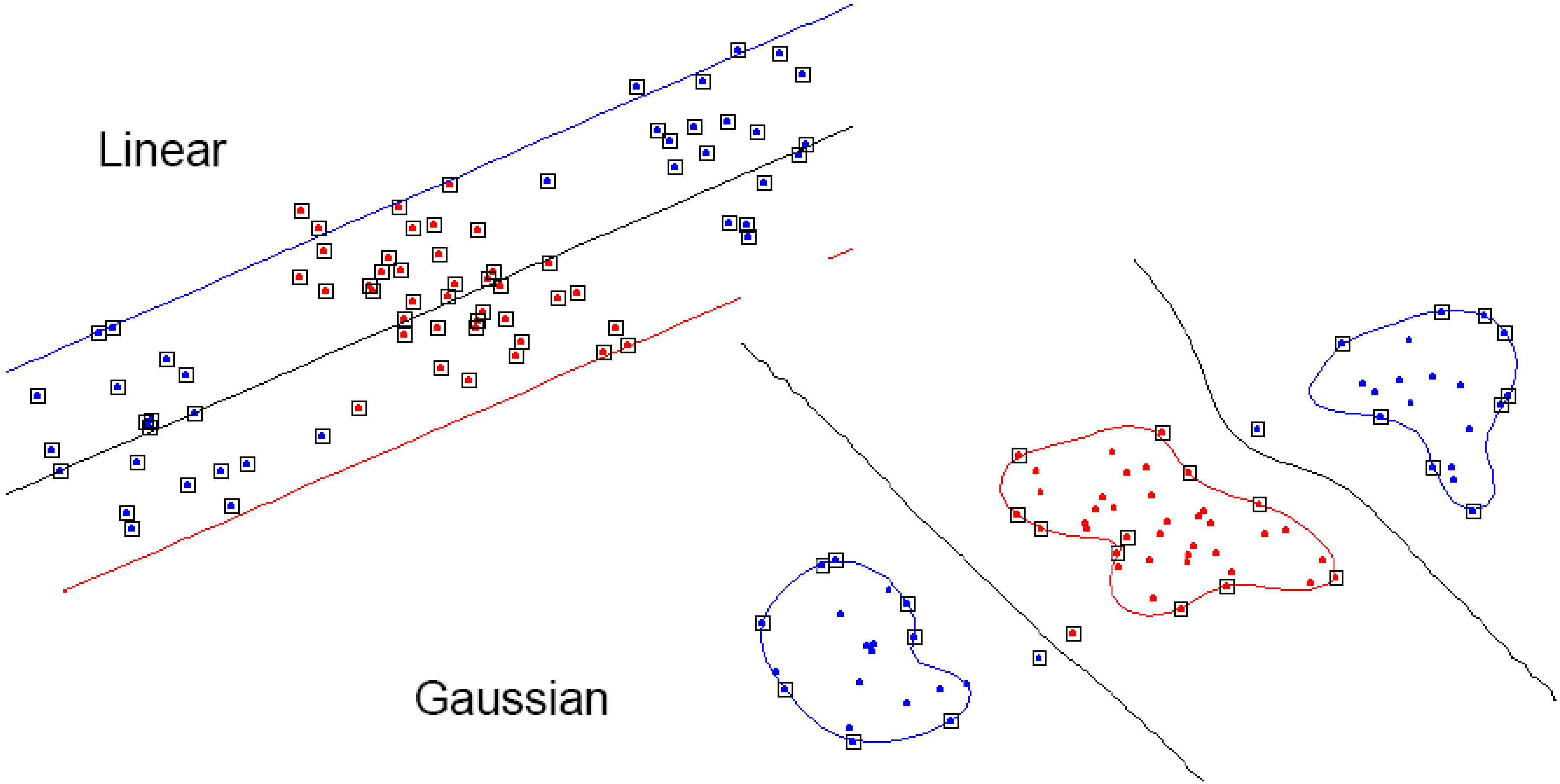
    proof $K_{i,j} = \phi(x_i)^T \phi(x_j) = \phi(x_j)^T \phi(x_i) = K_{j,i}$
  - Positive –semidefinite $y^T K y \geq 0 \quad \forall y$

    proof?
  - Mercer's theorem

SVM examples



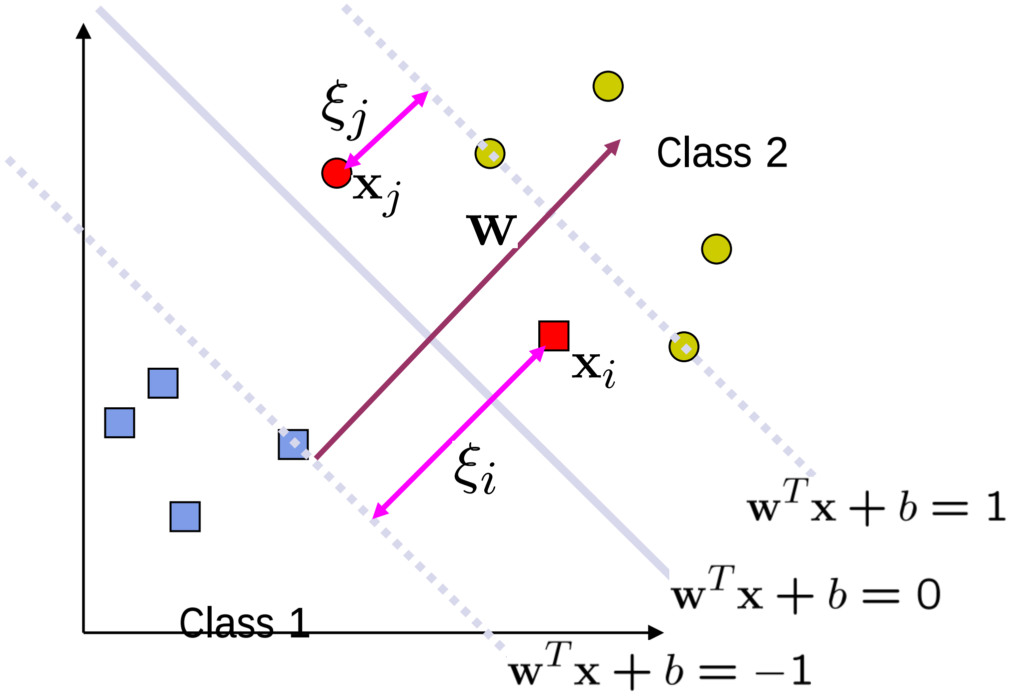
Examples for Non Linear SVMs – Gaussian Kernel



Example Kernel

- x_i is a bag of words
- Define $\phi(x_i)$ as a count of every n-gram up to n=k in x_i.
  - This is huge space 26^k
  - What are we measuring by $\phi(x_i)^t \phi(x_j)$?
- Can we compute the same quantity on input space?
  - Efficient linear dynamic program!
- Kernel is a measure of similarity
- Must be positive semi-definite

Non-linearly Separable Problems



- We allow "error" ξ_i in classification; it is based on the output of the discriminant function $\boldsymbol{w}^T\boldsymbol{x}+b$
- ξ_i approximates the number of misclassified samples

Soft Margin Hyperplane

- Now we have a slightly different opt problem:

$$\min_{w,b} \quad \frac{1}{2} w^T w + C \sum_{i=1}^{m} \xi_i$$

$$\text{s.t} \quad \begin{aligned} & y_i (w^T x_i + b) \geq 1 - \xi_i, \quad \forall i \\ & \xi_i \geq 0, \quad \forall i \end{aligned}$$

  - ξ_i are "slack variables" in optimization
  - Note that ξ_i=0 if there is no error for $\mathbf{x}_i$
  - ξ_i is an upper bound of the number of errors
  - *C* : tradeoff parameter between error and margin

The Optimization Problem

- The dual of this new constrained optimization problem is

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- This is very similar to the optimization problem in the linear separable case, except that there is an upper bound *C* on α_i now
- Once again, a QP solver can be used to find α_i

The SMO algorithm

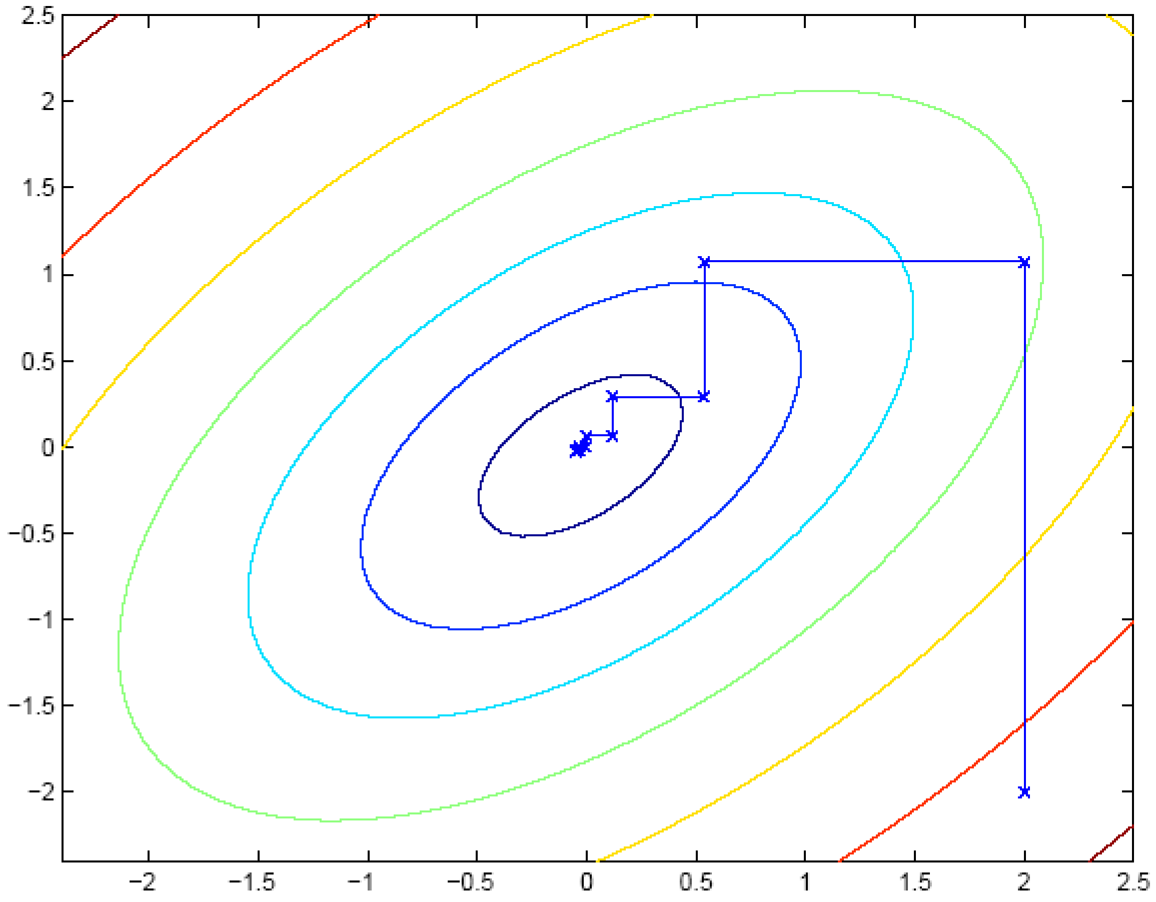
- Consider solving the unconstrained opt problem:

$$\max_{\alpha} W(\alpha_1, \alpha_2, \ldots, \alpha_m)$$

- We've already seen several opt algorithms!
  - ?
  - ?
  - ?

- Coordinate ascend:

Coordinate ascend



Sequential minimal optimization

- Constrained optimization:

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, m$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Question: can we do coordinate along one direction at a time (i.e., hold all $\alpha_{[-i]}$ fixed, and update α_i?)

The SMO algorithm

Repeat till convergence

1. Select some pair α_i and α_j to update next (using a heuristic that tries to pick the two that will allow us to make the biggest progress towards the global maximum).

2. Re-optimize J(α) with respect to α_i and α_j, while holding all the other α_k 's ($k \neq i$; j) fixed.

Will this procedure converge?

Convergence of SMO

$$\max_{\alpha} \quad \mathcal{J}(\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j (\mathbf{x}_i^T \mathbf{x}_j)$$

KKT:

$$\text{s.t.} \quad 0 \le \alpha_i \le C, \quad i = 1, \ldots, k$$

$$\sum_{i=1}^{m} \alpha_i y_i = 0.$$

- Let's hold $\alpha_3, \ldots, \alpha_m$ fixed and reopt J w.r.t. α_1 and α_2

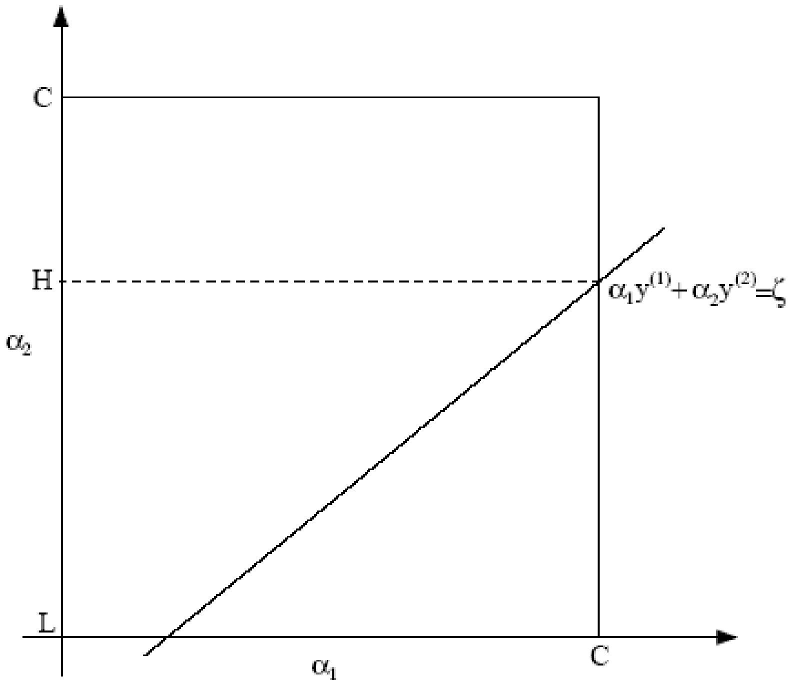
Convergence of SMO

- The constraints:

$$\alpha_1 y_1 + \alpha_2 y_2 = \xi$$

$$0 \leq \alpha_1 \leq C$$

$$0 \leq \alpha_2 \leq C$$



- The objective:

$$\mathcal{J}(\alpha_1, \alpha_2, \ldots, \alpha_m) = \mathcal{J}((\xi - \alpha_2 y_2) y_1, \alpha_2, \ldots, \alpha_m)$$

- Constrained opt:

Summary

- Max-margin decision boundary
- Constrained convex optimization
  - Duality
  - The KTT conditions and the support vectors
  - Non-separable case and slack variables
  - The SMO algorithm