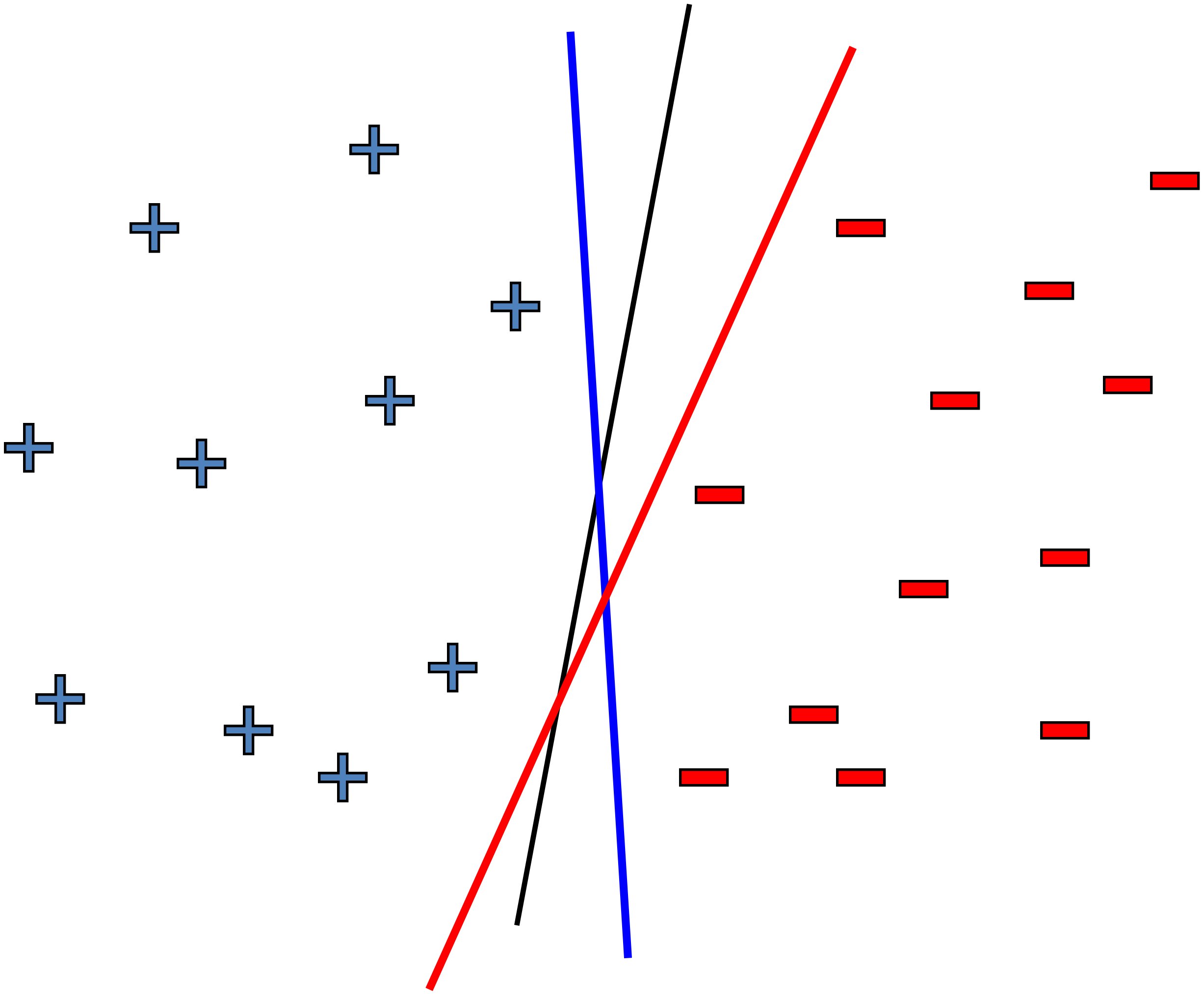
Support Vector Machines

Pradeep Ravikumar
Co-instructor: Manuela Veloso

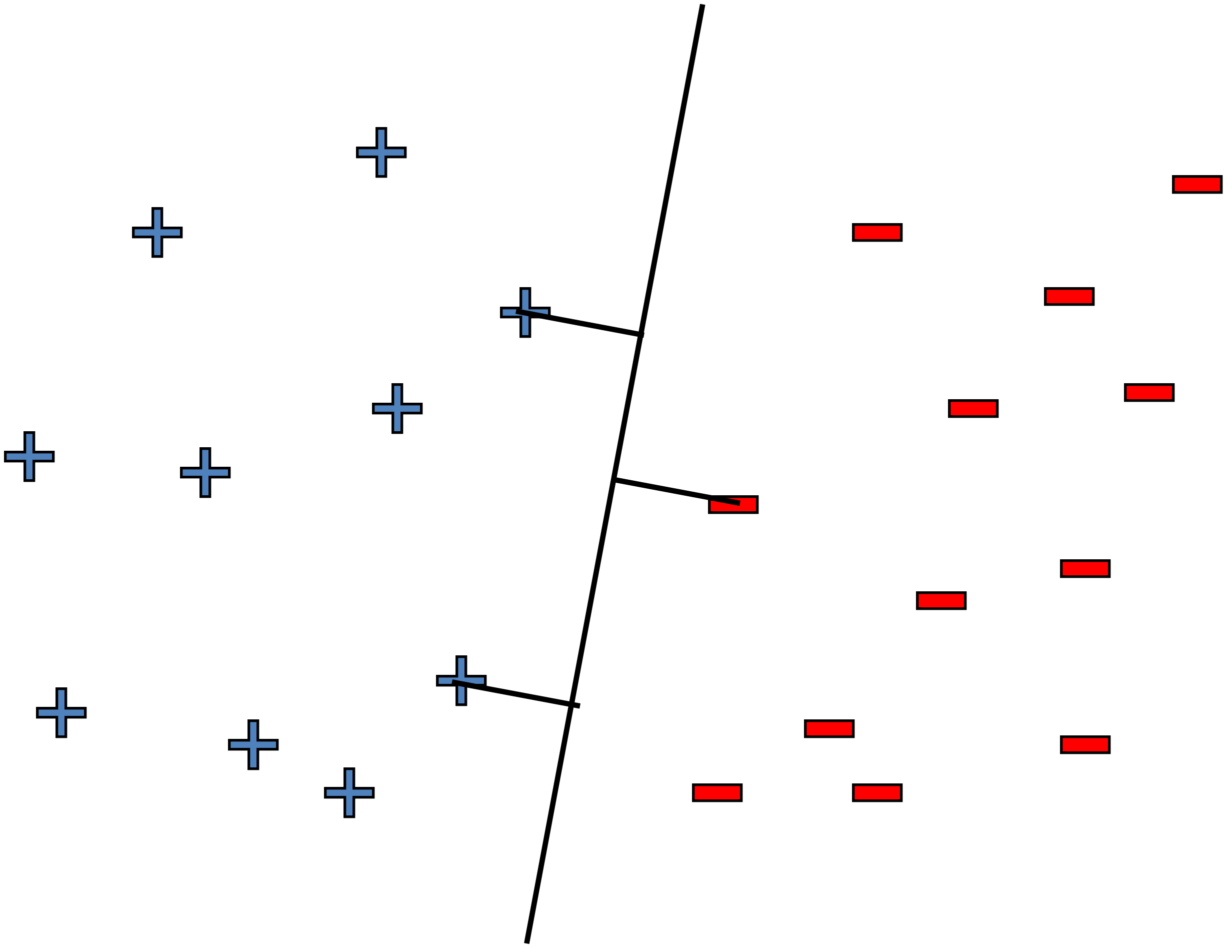
Machine Learning 10-701

At Pittsburgh G-20 summit …

Linear classifiers – which line is better?



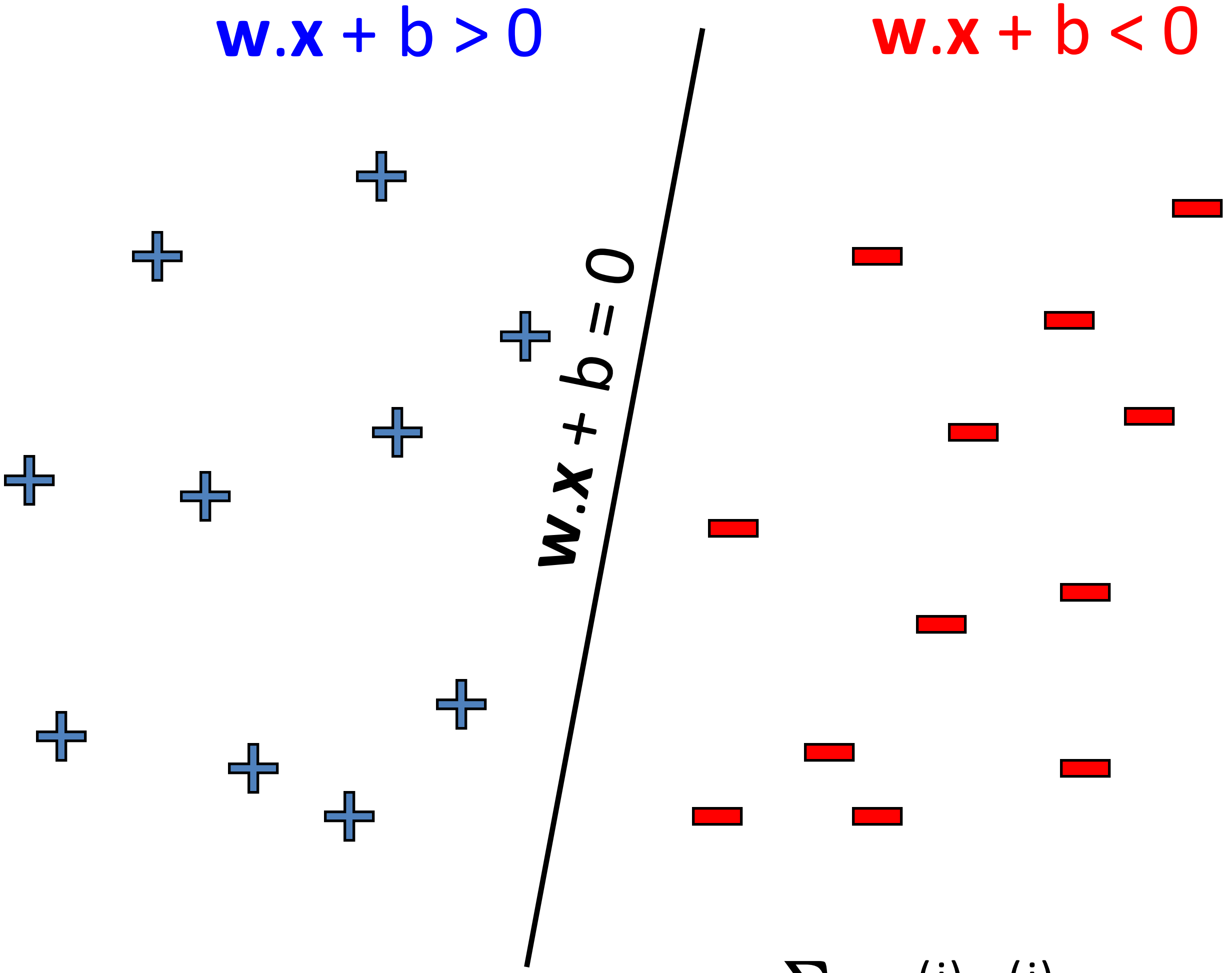
Pick the one with the largest margin!



Parameterizing the decision boundary

$\mathbf{w}.\mathbf{x} + b > 0$

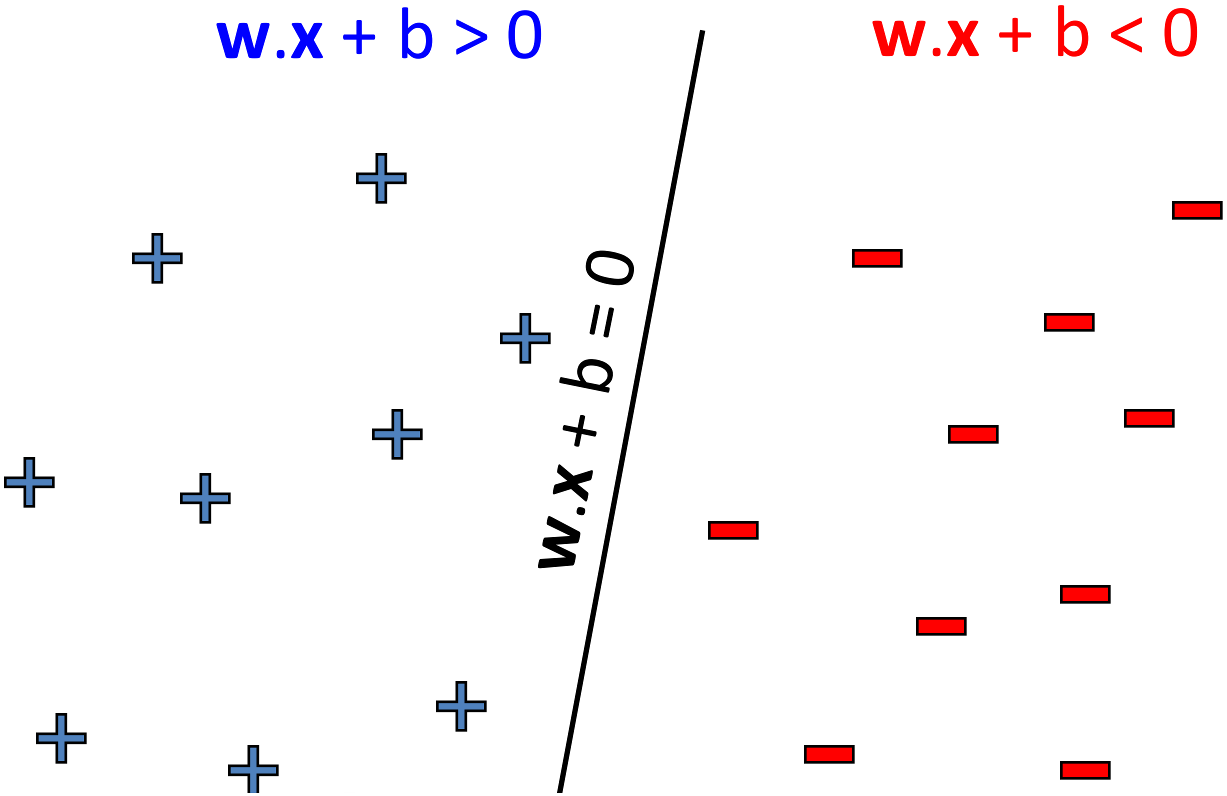
$\mathbf{w}.\mathbf{x} + b < 0$



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

$\mathbf{w}.\mathbf{x} = \sum_j w^{(j)} x^{(j)}$

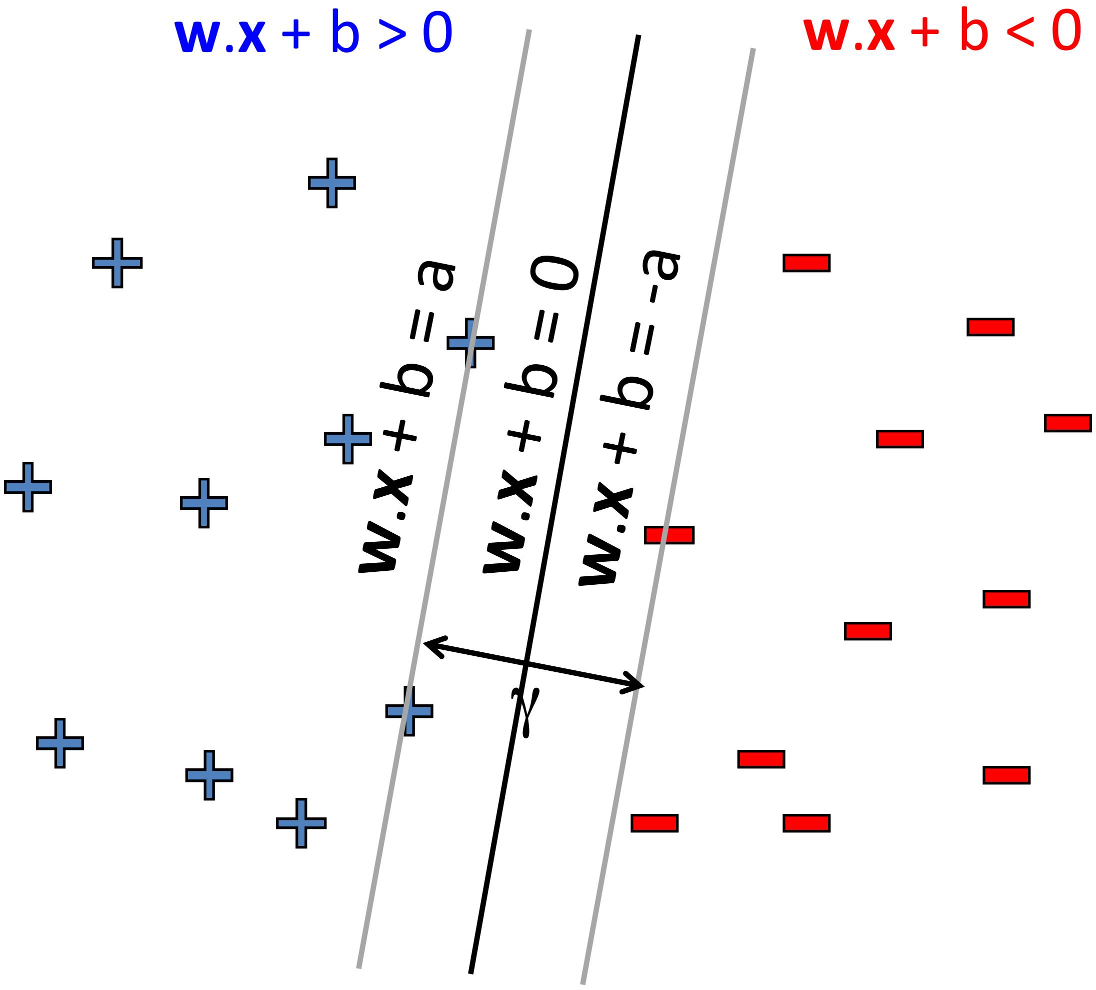
Parameterizing the decision boundary



$y_j \in \{-1, +1\}$ — class

"confidence" $= \left(\mathbf{w}.\mathbf{x}_j + b\right) y_j$

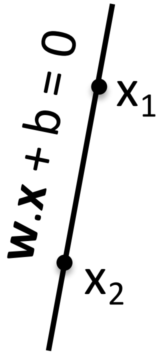
Maximizing the margin



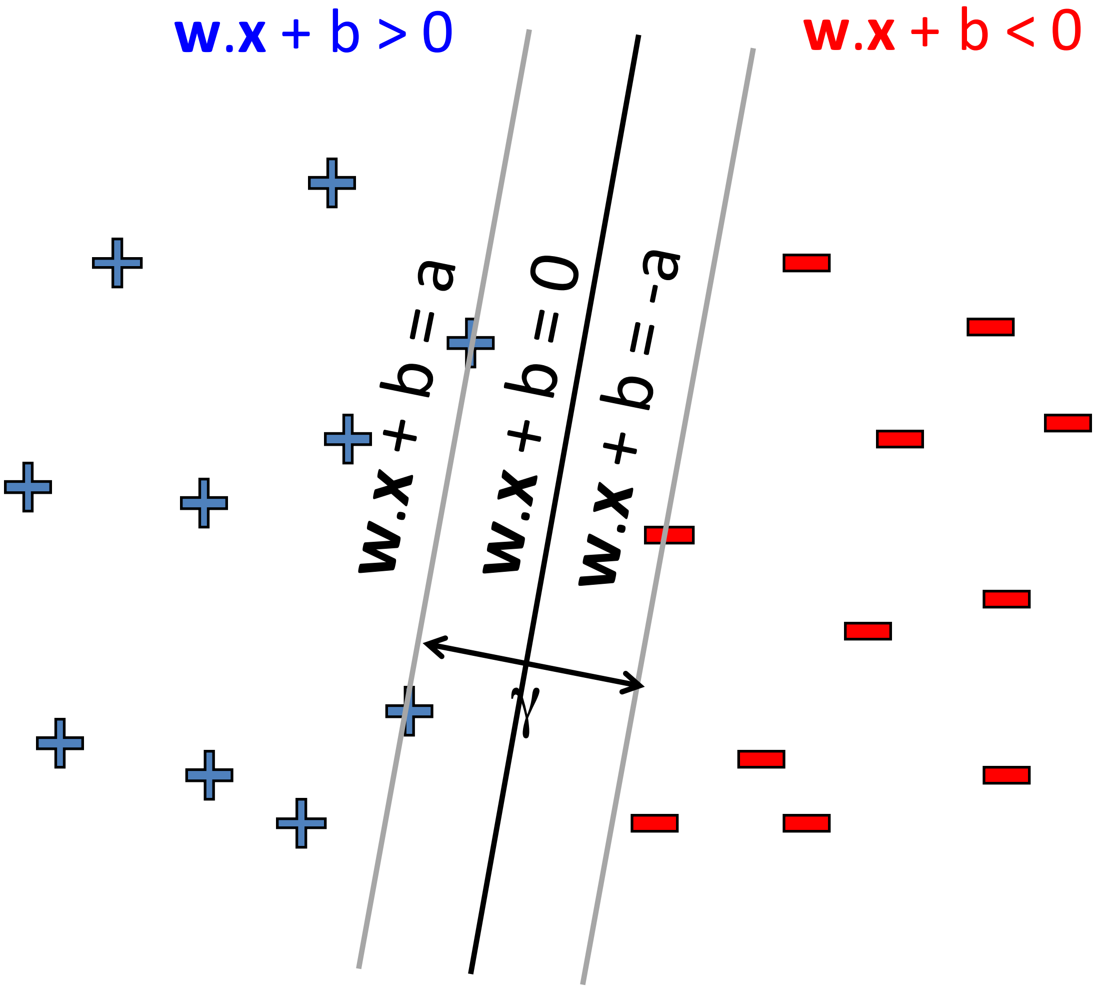
Distance of closest examples from the line/hyperplane

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

Step 1: **w** is perpendicular to lines since for any x_1, x_2 on line $\mathbf{w}.(\mathbf{x}_1 - \mathbf{x}_2) = 0$



Maximizing the margin



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

Step1: **w** is perpendicular to lines

Step 2: Take a point x_- on w.x +b = -a and move to point x_+ that is γ away on line w.x+b = a

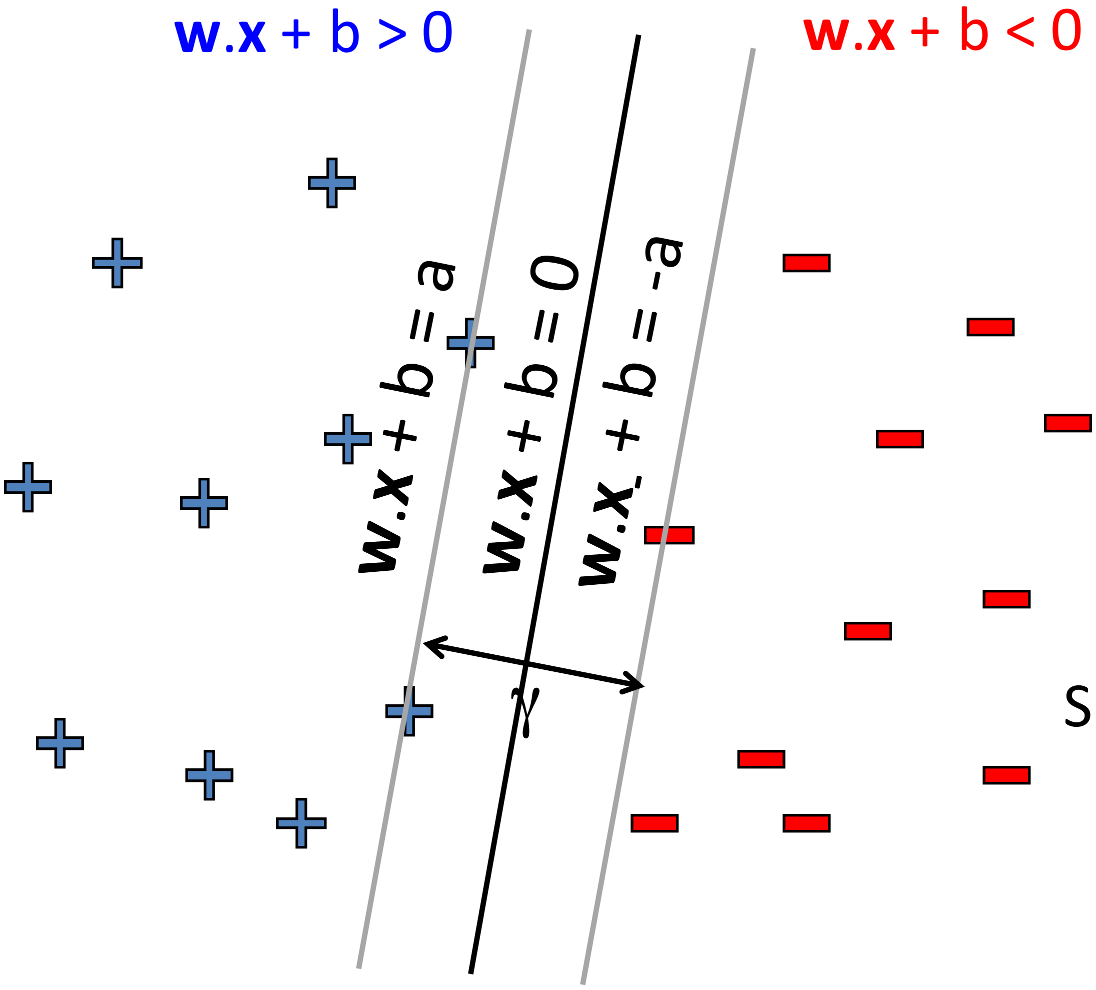
$$\mathbf{x}_+ = \mathbf{x}_- + \gamma \mathbf{w}/\|w\|$$

$$\mathbf{w}.\mathbf{x}_+ = \mathbf{w}.\mathbf{x}_- + \gamma \mathbf{w}.\ \mathbf{w}/\|w\|$$

$$a\text{-}b = \text{-}a\text{-}b + \gamma\|w\|$$

$$2a = \gamma\|w\|$$

Maximizing the margin



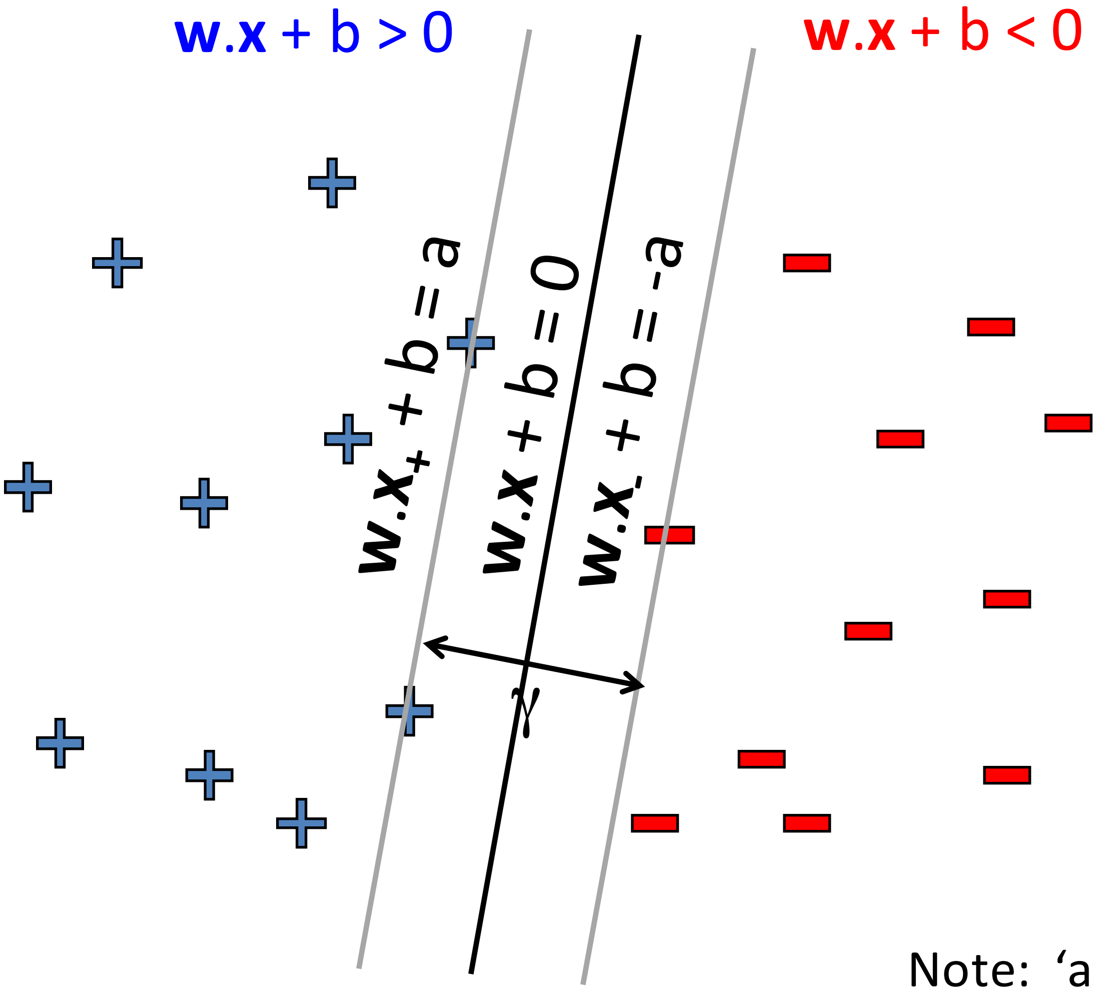
$\mathbf{w}.\mathbf{x} + b > 0$
$\mathbf{w}.\mathbf{x} + b < 0$
$\mathbf{w}.\mathbf{x} + b = a$
$\mathbf{w}.\mathbf{x} + b = 0$
$\mathbf{w}.\mathbf{x}_- + b = -a$
γ

Distance of closest examples from the line/hyperplane

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

Smaller margin ⇔ larger $\|w\|$

Maximizing the margin



Distance of closest examples from the line/hyperplane

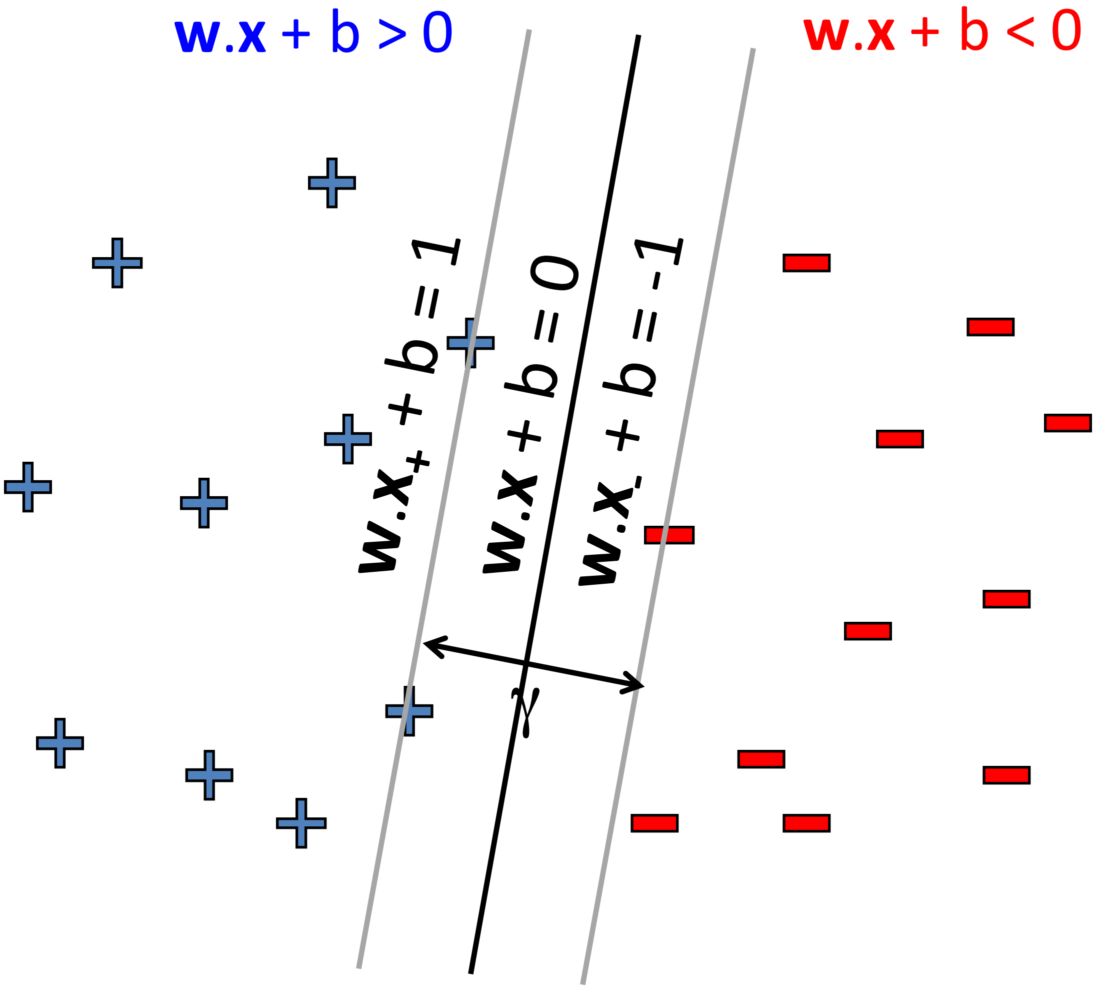
$$\boxed{\text{margin} = \gamma = 2a/\|w\|}$$

$$\max_{\mathbf{w},b} \ \gamma = 2a/\|w\|$$

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

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

Support Vector Machines

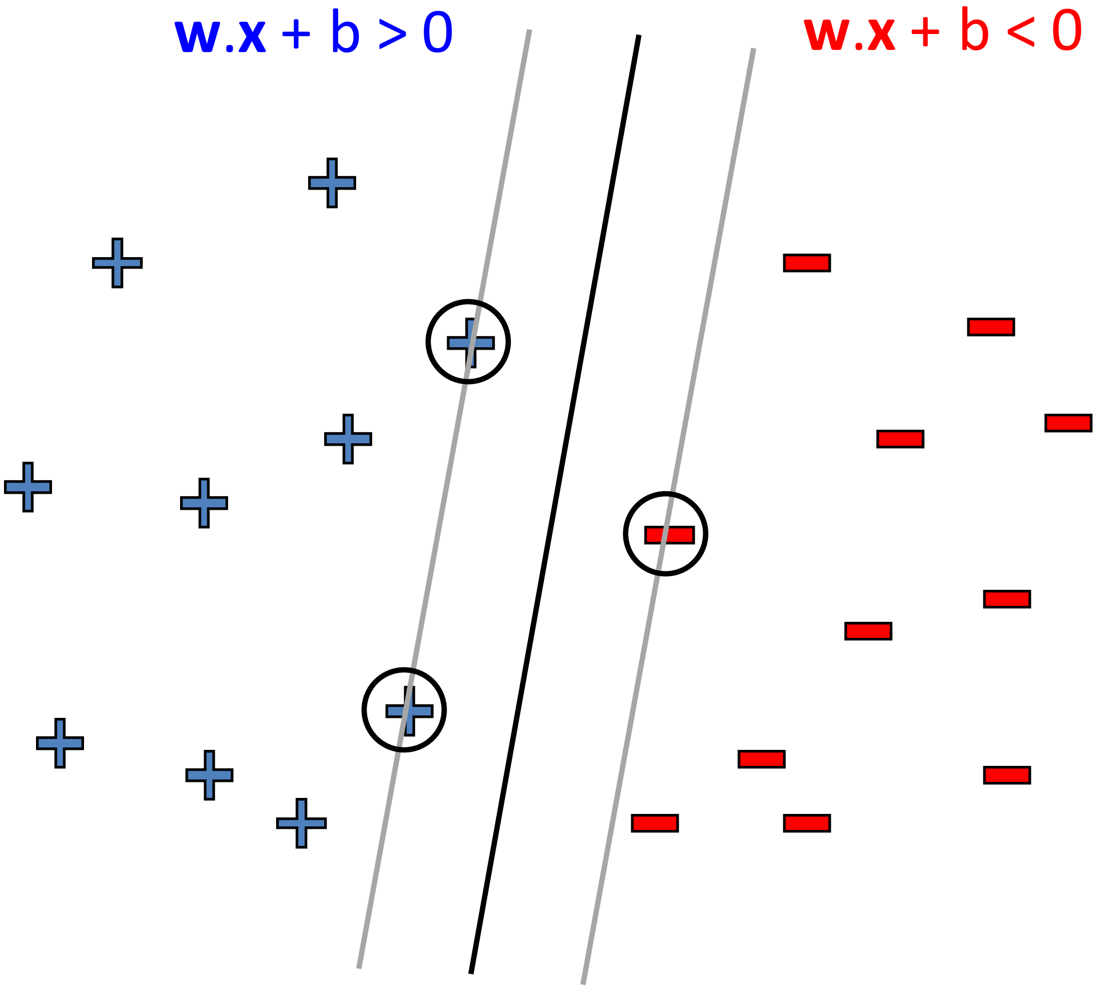

$\mathbf{w}.\mathbf{x} + b > 0$
$\mathbf{w}.\mathbf{x} + b < 0$
$\mathbf{w}.\mathbf{x}_+ + b = 1$
$\mathbf{w}.\mathbf{x} + b = 0$
$\mathbf{w}.\mathbf{x}_- + b = -1$
γ

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

Solve efficiently by quadratic programming (QP)

- Quadratic objective, linear constraints
- Well-studied solution algorithms

Support Vectors



$\mathbf{w}.\mathbf{x} + b > 0$

$\mathbf{w}.\mathbf{x} + b < 0$

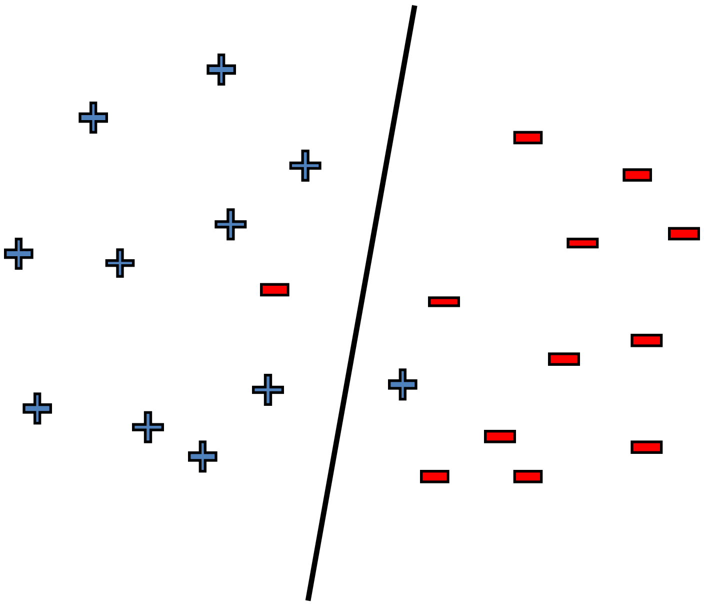
Linear hyperplane defined by "<u>support vectors</u>"

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

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

For support vectors $(\mathbf{w}.\mathbf{x}_j + b)\, y_j = 1$

What if data is not linearly separable?



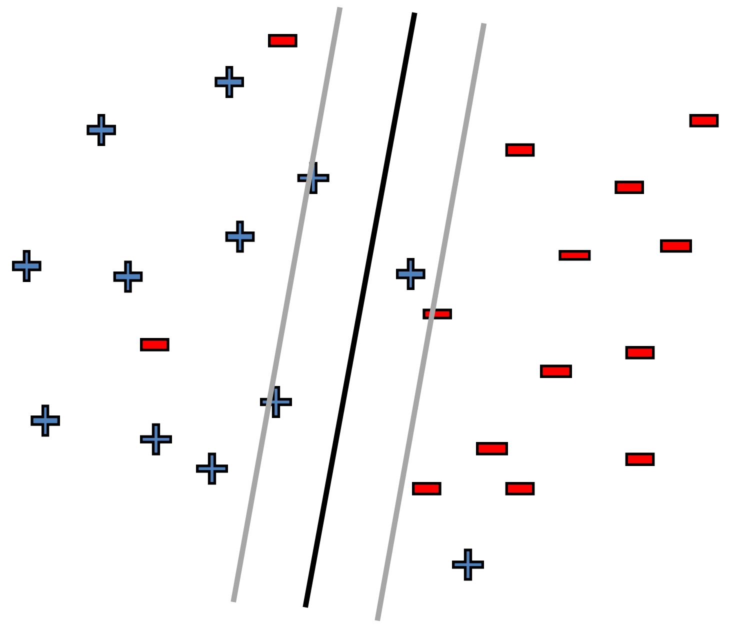
Use features of features of features of features….

x_1^2, x_2^2, x_1x_2, …., $\exp(x_1)$

But run risk of overfitting!

What if data is still not linearly separable?

Allow “error” in classification



Smaller margin ⇔ larger ‖w‖

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

Maximize margin and minimize # mistakes on training data

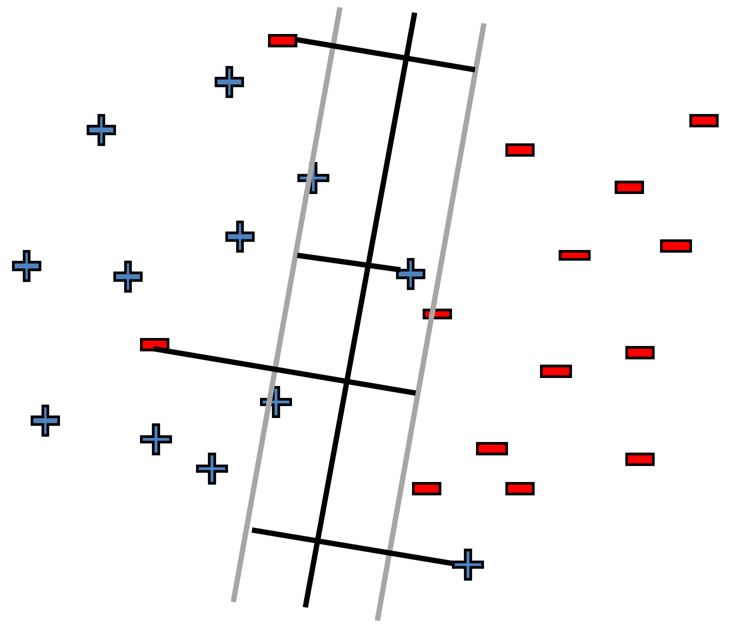
C - tradeoff parameter

Not QP ☹

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

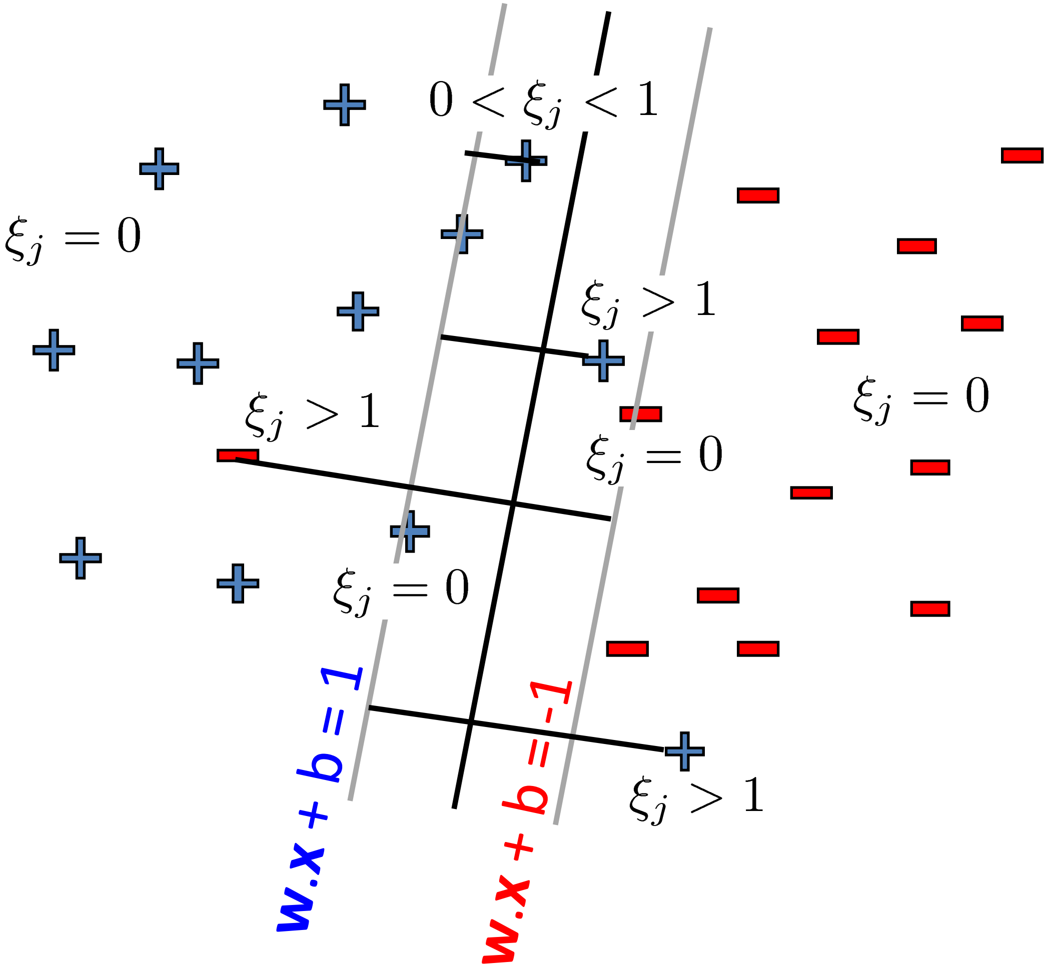
= (>1 if x_j misclassifed)

pay linear penalty if mistake

C - tradeoff parameter (chosen by cross-validation)

Still QP ☺

Soft-margin SVM



Soften the constraints:

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

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

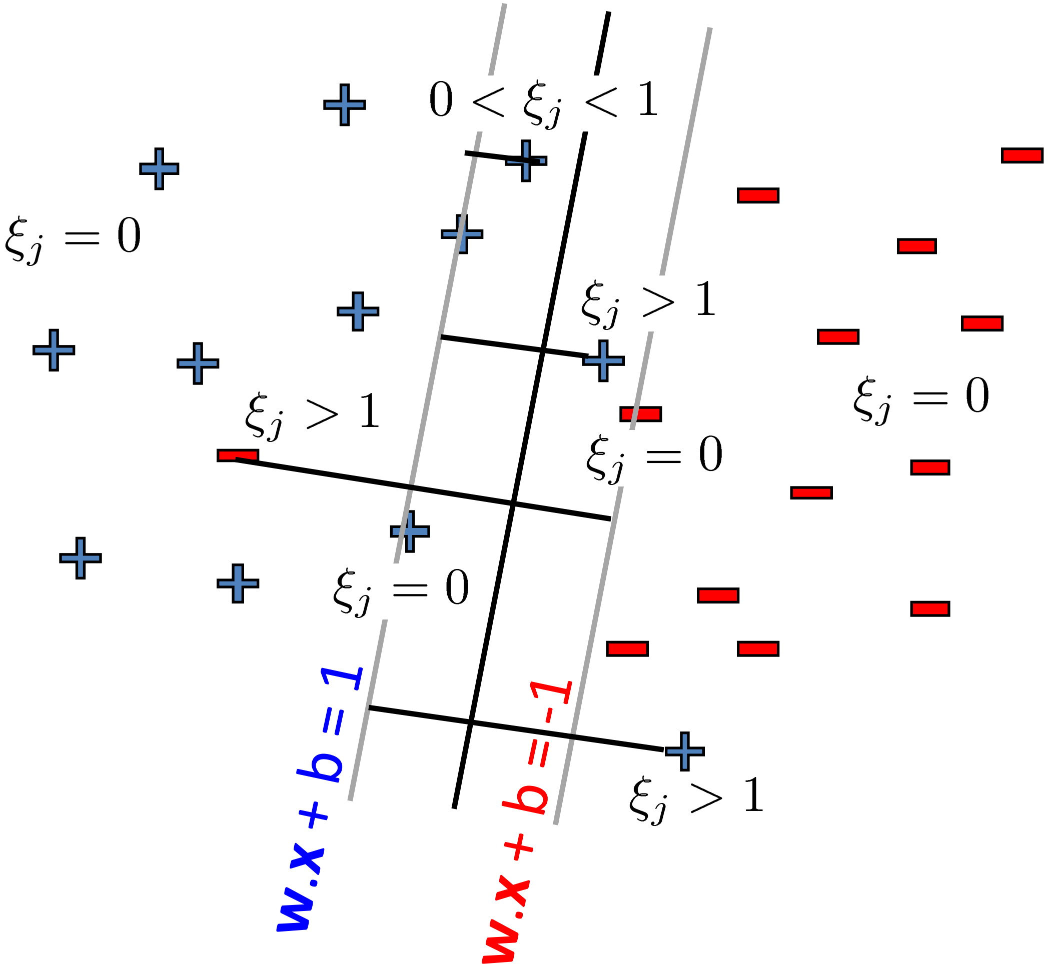
Penalty for misclassifying:

$$C\, \xi_j$$

How do we recover hard margin SVM?

Set C = ∞

Slack variables – Hinge loss

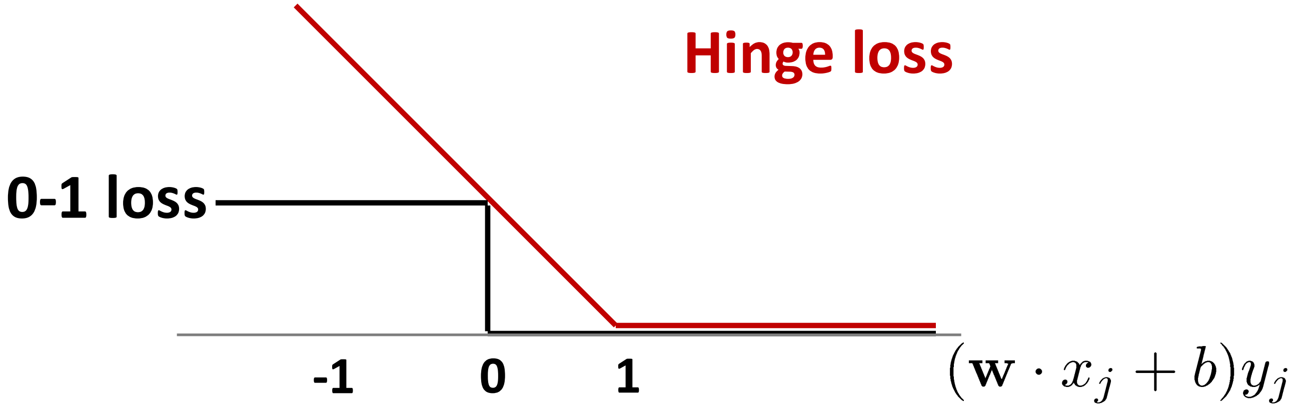


Notice that

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

Slack variables – Hinge loss

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



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

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

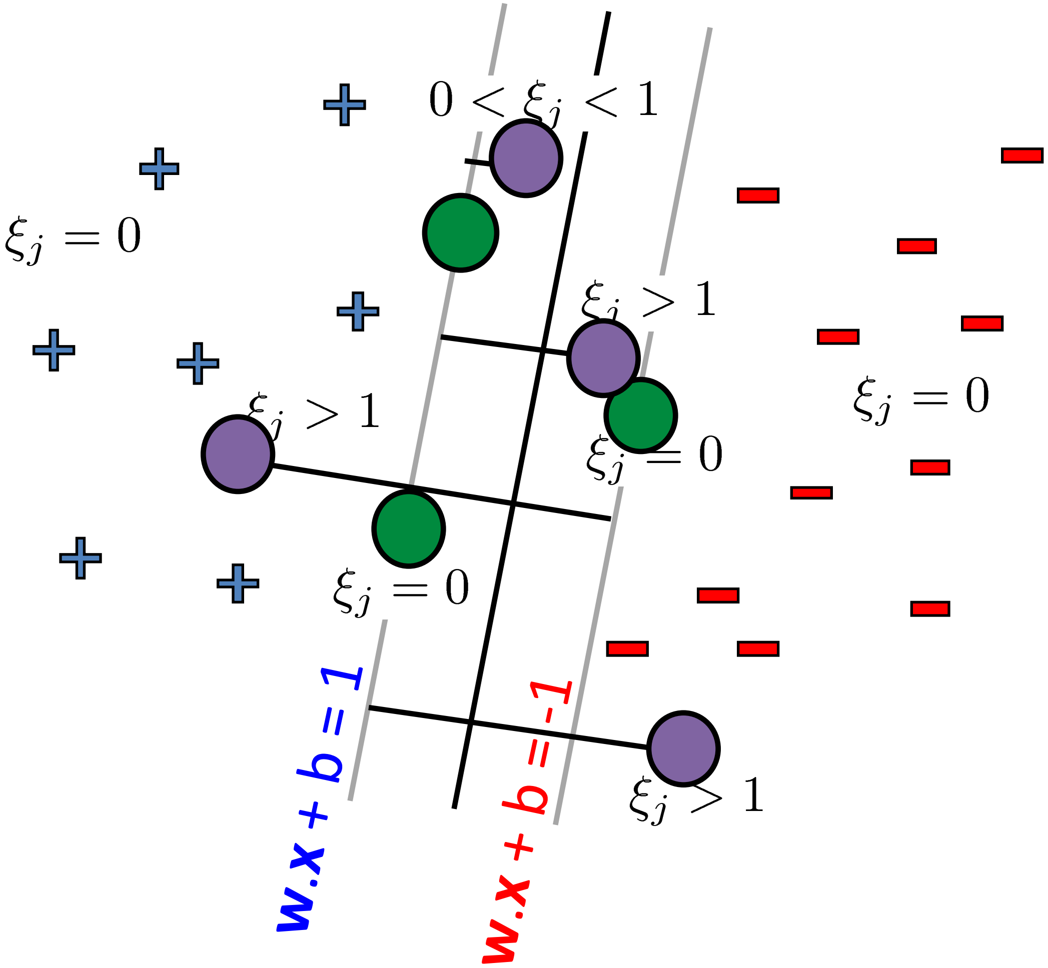
$$\xi_j \geq 0 \quad \forall j$$

⇔

Regularized hinge loss

$$\min_{\mathbf{w},b} \ \mathbf{w}.\mathbf{w} + C \sum_j (1-(\mathbf{w}.x_j+b)y_j)_+$$

Support Vectors



Margin support vectors

$\xi_j = 0$, $(\mathbf{w}.\mathbf{x}_j+b)\, y_j = 1$

(don't contribute to objective but enforce constraints on solution)

Correctly classified but on margin

Non-margin support vectors

$\xi_j > 0$

(contribute to both objective and constraints)

$1 > \xi_j > 0$ Correctly classified but inside margin

$\xi_j > 1$ Incorrectly classified

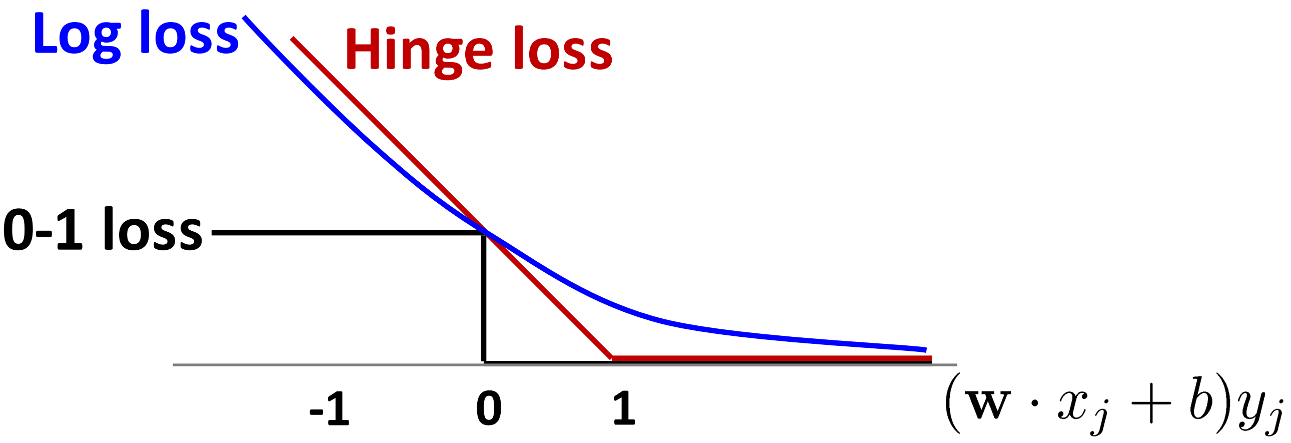
SVM vs. Logistic Regression

SVM : **Hinge loss**

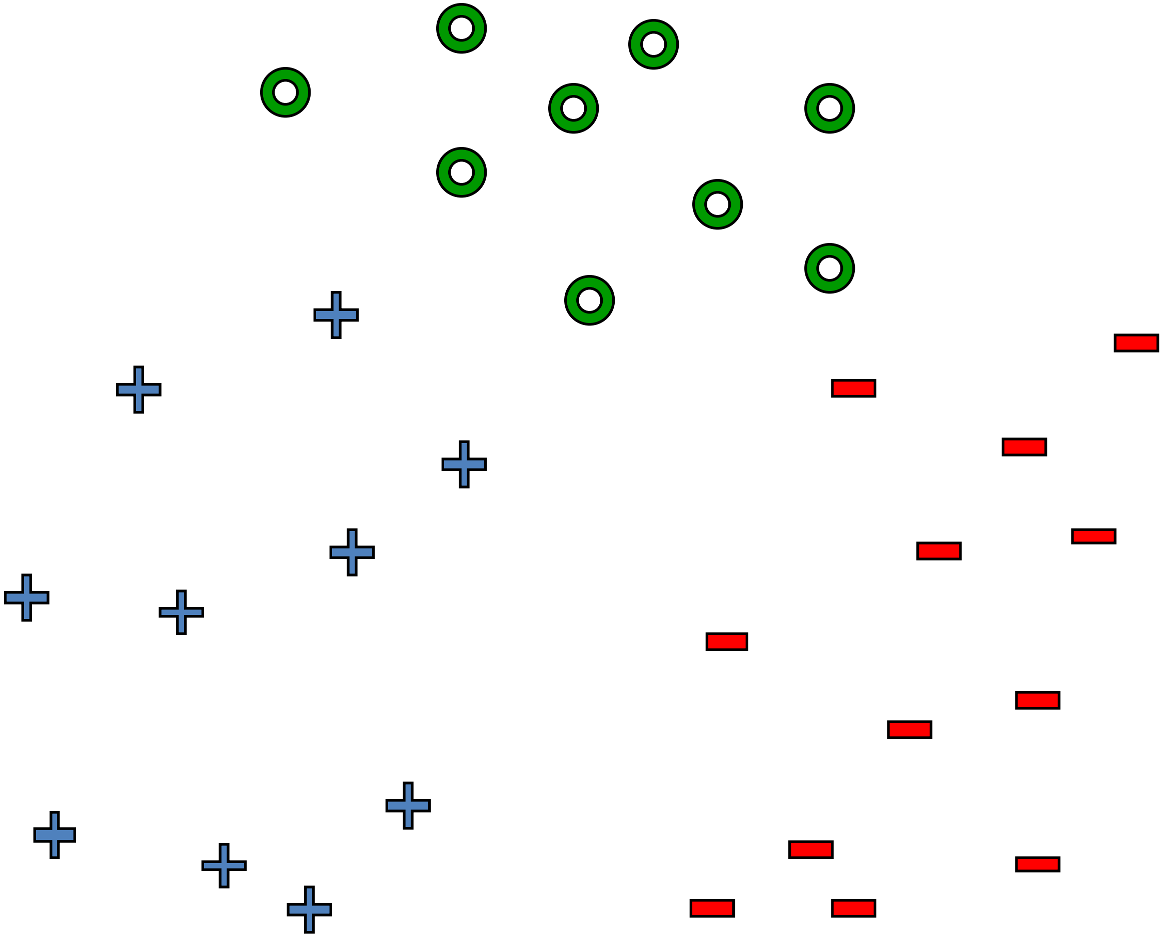
$$\text{loss}(f(x_j), y_j) \; = (1 - (\mathbf{w} \cdot x_j + b) y_j))_+$$

Logistic Regression : **Log loss** (-ve log conditional likelihood)

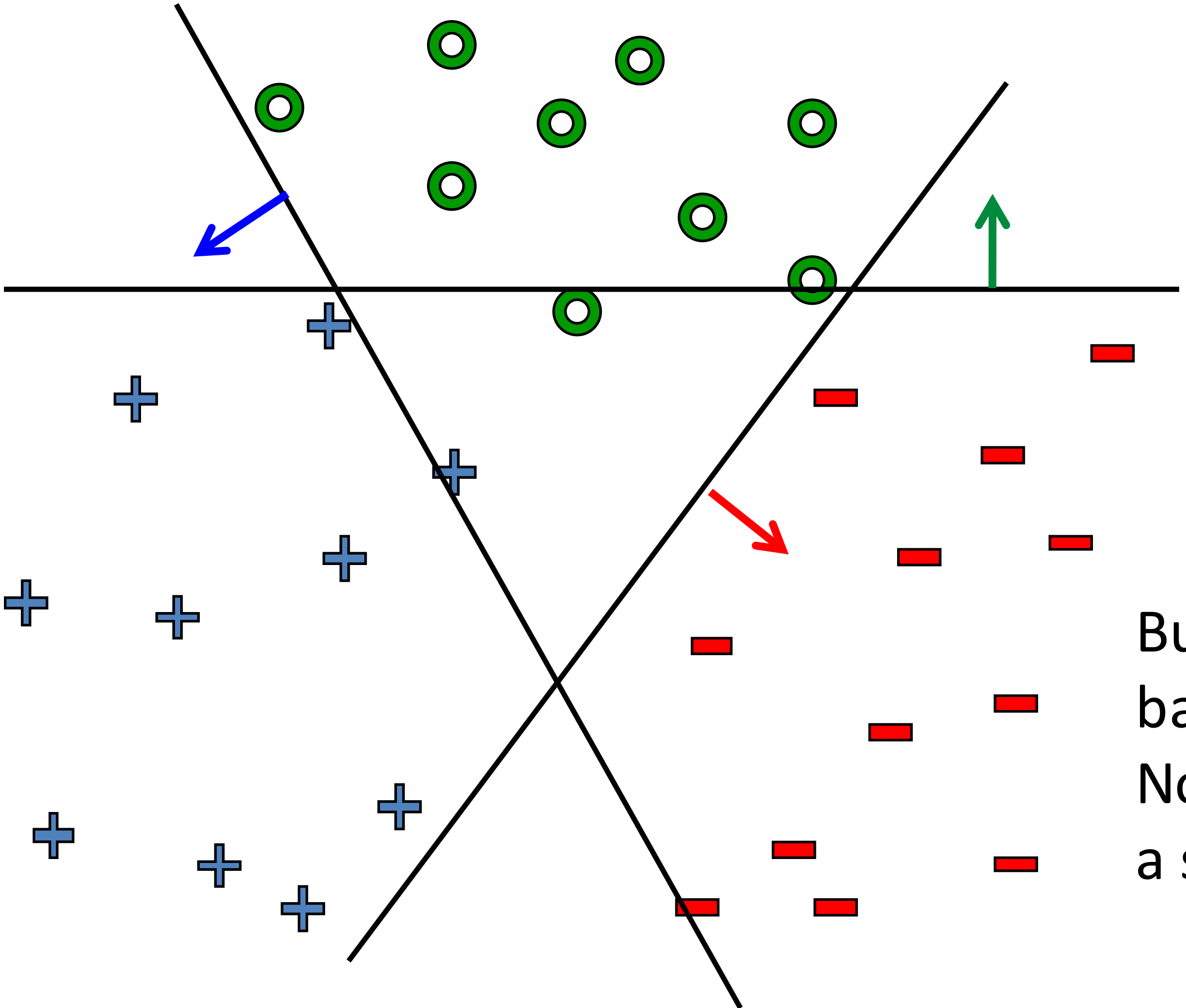
$$\text{loss}(f(x_j), y_j) \; = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b) y_j})$$



What about multiple classes?



One vs. rest



Learn 3 classifiers separately:

Class k vs. rest

$(\mathbf{w}_k, b_k)_{k=1,2,3}$

$$y = \arg\max_k \mathbf{w}_k.x + b_k$$

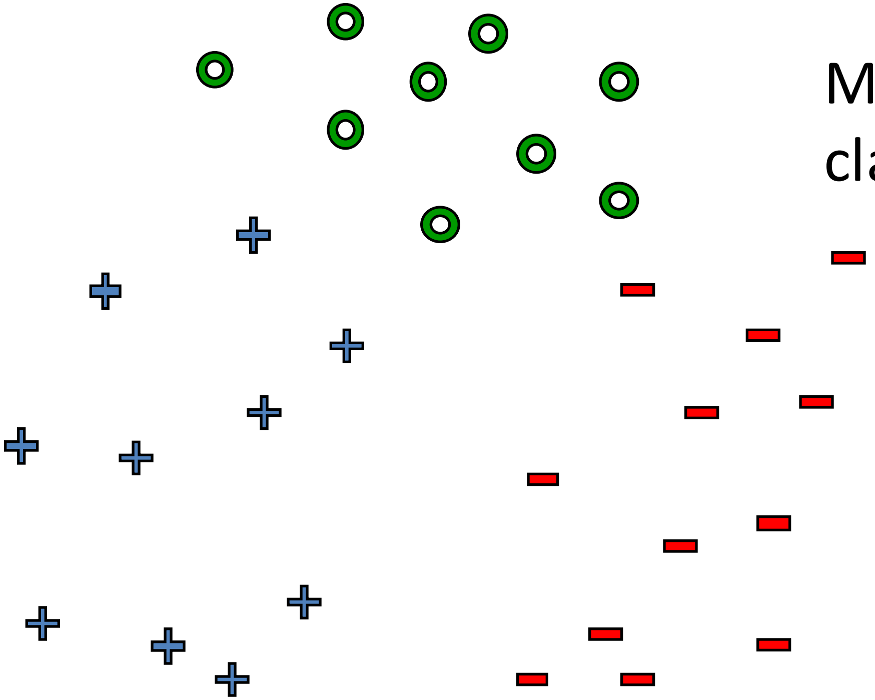
But $\mathbf{w}_k$s may not be based on the same scale. Note: $(a\mathbf{w}).x + (ab)$ is also a solution

Learn 1 classifier: Multi-class SVM

Simultaneously learn 3 sets of weights

$$\min_{\{w^{(y)}\},\{b^{(y)}\}} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)}$$

$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1, \ \forall y' \neq y_j, \ \forall j$$



Margin - gap between correct class and nearest other class

y = arg max$_k$ $\mathbf{w}^{(k)}$.x + b$^{(k)}$

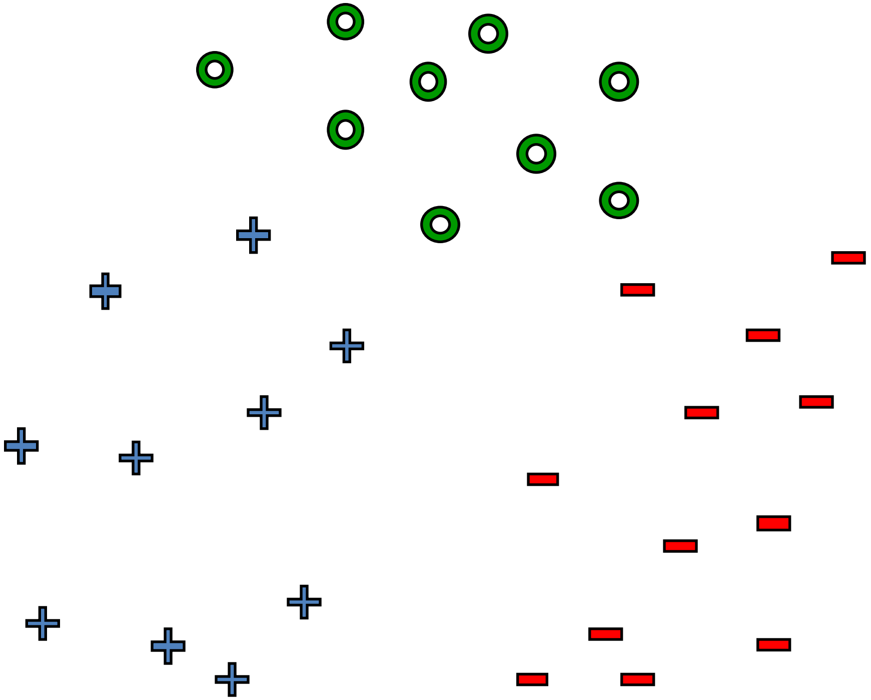
Learn 1 classifier: Multi-class SVM

Simultaneously learn 3 sets of weights

$$\text{minimize} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \sum_{y \neq y_j} \xi_j^{(y)} \quad \text{over } \{\mathbf{w}^{(y)}\}, \{b^{(y)}\}, \{\xi_j^{(y)}\}$$

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

$$\xi_j^{(y)} \geq 0 \qquad , \ \forall y \neq y_j, \ \forall j$$



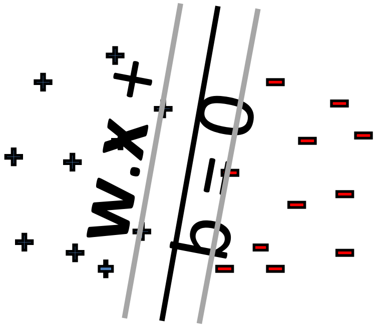
y = arg max $\mathbf{w}^{(k)}$.x + $b^{(k)}$

Joint optimization: $\mathbf{w}_k$s have the same scale.

SVM – linearly separable case

n training points $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$

d features $\mathbf{x}_j$ is a d-dimensional vector



- Primal problem: $$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$ $$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \ \forall j$$

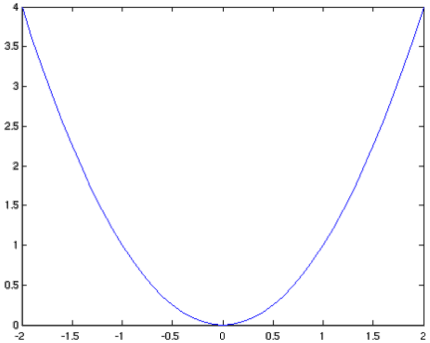
w – weights on features (d-dim problem)

- Convex quadratic program – quadratic objective, linear constraints
- But expensive to solve if d is very large
- Often solved in dual form (n-dim problem)

Constrained Optimization

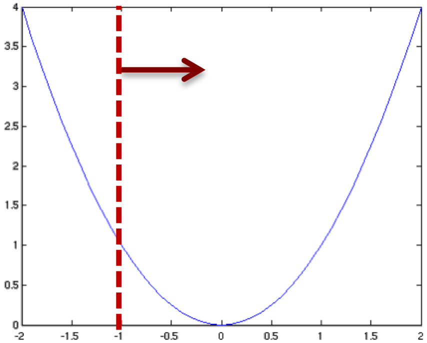
$$\min_x \; x^2 \quad \text{s.t.} \quad x \geq b \qquad\qquad x^* = \max(b, 0)$$

$\min_x \; x^2$



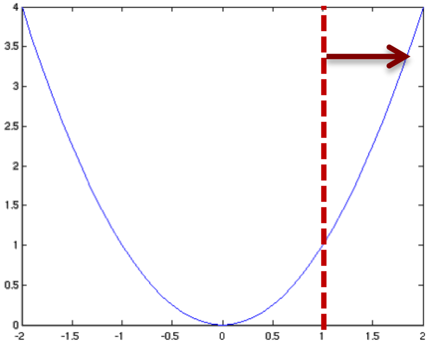
$x^* = 0$

$\min_x \; x^2$
s.t. $x \geq -1$



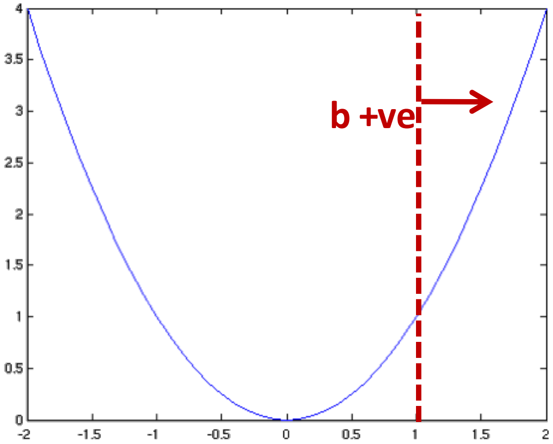
$x^* = 0$
Constraint inactive

$\min_x \; x^2$
s.t. $x \geq 1$



$x^* = 1$
Constraint active
and tight

Constrained Optimization – Dual Problem



$$x^* = b$$

α = 0 constraint is inactive

α > 0 constraint is active

Primal problem:

$$\begin{aligned} &\text{min}_x \quad x^2 \\ &\text{s.t.} \quad x \geq b \end{aligned}$$

Moving the constraint to objective function

Lagrangian:

$$\begin{aligned} &L(x, \alpha) = x^2 - \alpha(x - b) \\ &\text{s.t.} \quad \alpha \geq 0 \end{aligned}$$

Dual problem:

$$\begin{aligned} &\text{max}_\alpha \quad d(\alpha) \longrightarrow \text{min}_x \, L(x, \alpha) \\ &\text{s.t.} \quad \alpha \geq 0 \end{aligned}$$

Connection between Primal and Dual

Primal problem: p* = $\min_x \; x^2$ s.t. $x \geq b$

Dual problem: d* = $\max_\alpha \; d(\alpha)$ s.t. $\alpha \geq 0$

- **Weak duality:** The dual solution d* lower bounds the primal solution p* i.e. d* ≤ p*

  To see this, recall $L(x, \alpha) = x^2 - \alpha(x - b)$

  For every feasible x (i.e. x ≥ b) and feasible α (i.e. α ≥ 0) , notice that

  d(α) = $\min_x L(x, \alpha)$ ≤ p*

- **Dual problem (maximization) is always concave even if primal is not convex**

Connection between Primal and Dual

Primal problem: p* = $\min_x \; x^2$ s.t. $x \geq b$

Dual problem: d* = $\max_\alpha \; d(\alpha)$ s.t. $\alpha \geq 0$

- **Weak duality:** The dual solution d* lower bounds the primal solution p* i.e. d* ≤ p*
- **Strong duality:** d* = p* holds often for many problems of interest e.g. if the primal is a feasible convex objective with linear constraints

Connection between Primal and Dual

What does strong duality say about α^* (the α that achieved optimal value of dual) and x^* (the x that achieves optimal value of primal problem)?

Whenever strong duality holds, the following conditions (known as KKT conditions) are true for α^* and x^*:

- 1. $\bigtriangledown L(x^*, \alpha^*) = 0$ i.e. Gradient of Lagrangian at x^* and α^* is zero.
- 2. $x^* \geq b$ i.e. x^* is primal feasible
- 3. $\alpha^* \geq 0$ i.e. α^* is dual feasible
- 4. $\alpha^*(x^* - b) = 0$ (called as complementary slackness)

We use the first one to relate x^* and α^*. We use the last one (complimentary slackness) to argue that $\alpha^* = 0$ if constraint is inactive and $\alpha^* > 0$ if constraint is active and tight.

Solving the dual

Solving:

$$\max_\alpha \min_x \overbrace{x^2 - \alpha(x-b)}^{L(x,\alpha)}$$
$$\text{s.t.} \quad \alpha \geq 0$$

Optimization over x is unconstrained.

$$\frac{\partial L}{\partial x} = 2x - \alpha = 0 \Rightarrow x^* = \frac{\alpha}{2}$$

$$L(x^*, \alpha) = \frac{\alpha^2}{4} - \alpha\left(\frac{\alpha}{2} - b\right)$$
$$= -\frac{\alpha^2}{4} + b\alpha$$

Now need to maximize L(x*,α) over α ≥ 0

Solve unconstrained problem to get α′ and then take max(α′,0)

$$\frac{\partial}{\partial \alpha} L(x^*, \alpha) = -\frac{\alpha}{2} + b \;\; \Rightarrow \alpha' = 2b$$

$$\Rightarrow \alpha^* = \max(2b, 0) \qquad \Rightarrow x^* = \frac{\alpha^*}{2} = \max(b, 0)$$

α = 0 constraint is inactive, α > 0 constraint is active and tight

Dual SVM – linearly separable case

n training points, d features ($\mathbf{x}_1$, …, $\mathbf{x}_n$) where x_i is a d-dimensional vector

- Primal problem:

$$\text{minimize}_{\mathbf{w},b} \quad \frac{1}{2}\mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \forall j$$

w – weights on features (d-dim problem)

- Dual problem (derivation):

$$L(\mathbf{w}, b, \alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$
$$\alpha_j \geq 0, \ \forall j$$

a – weights on training pts (n-dim problem)

Dual SVM – linearly separable case

- Dual problem:

$$\max_{\alpha} \min_{\mathbf{w},b} L(\mathbf{w},b,\alpha) = \frac{1}{2}\mathbf{w}.\mathbf{w} - \sum_j \alpha_j \left[\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j - 1\right]$$

$$\alpha_j \geq 0, \ \ \forall j$$

$$\frac{\partial L}{\partial \mathbf{w}} = 0 \qquad \Rightarrow \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = 0 \qquad \Rightarrow \sum_j \alpha_j y_j = 0$$

If we can solve for αs (dual problem), then we have a solution for **w**,b (primal problem)

Dual SVM – linearly separable case

$$\text{maximize}_\alpha \quad \textstyle\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$$

$$\textstyle\sum_i \alpha_i y_i = 0$$
$$\alpha_i \geq 0$$

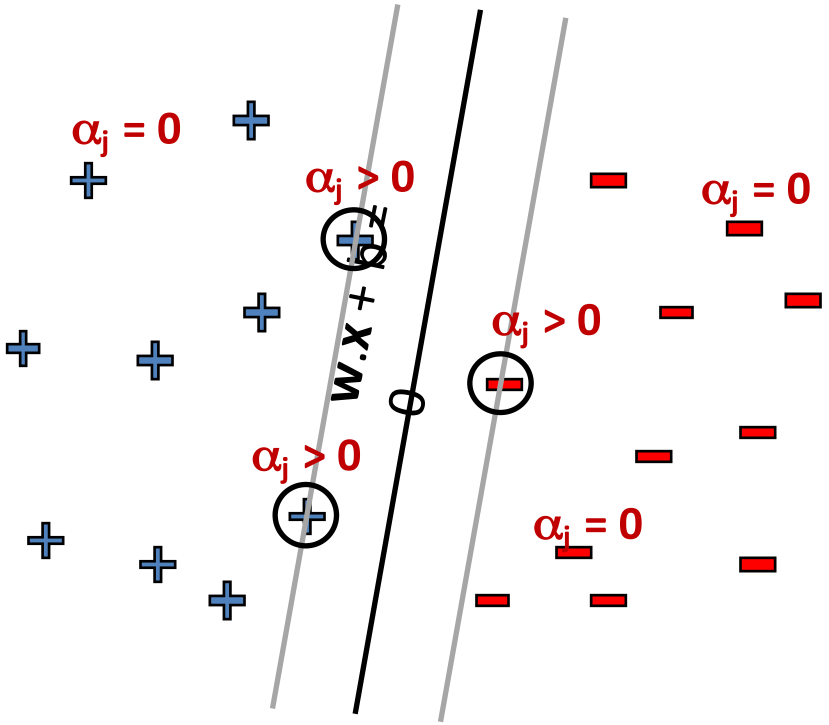
Dual problem is also QP

Solution gives α_js $\longrightarrow$

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

What about b?

Dual SVM: Sparsity of dual solution



$$\mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

Only few α_js can be non-zero : where constraint is active and tight

$(\mathbf{w}.\mathbf{x}_j + b)y_j = 1$

Support vectors – training points j whose α_js are non-zero

Dual SVM – linearly separable case

$$\text{maximize}_\alpha \quad \textstyle\sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$$

$$\textstyle\sum_i \alpha_i y_i = 0$$

$$\alpha_i \geq 0$$

Dual problem is also QP →

Solution gives α_js

$$\boxed{\begin{array}{l} \mathbf{w} = \displaystyle\sum_i \alpha_i y_i \mathbf{x}_i \\ \\ b = y_k - \mathbf{w}.\mathbf{x}_k \\ \text{for any } k \text{ where } \alpha_k > 0 \end{array}}$$

Use support vectors with α_k>0 to compute b since constraint is tight $(w.x_k + b)y_k = 1$

Dual SVM – non-separable case

- Primal problem:

$$\text{minimize}_{\mathbf{w},b,\{\xi_j\}} \tfrac{1}{2}\mathbf{w}.\mathbf{w} + C\sum_j \xi_j$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j,\ \ \forall j$$
$$\xi_j \geq 0,\ \ \forall j$$

$$\boxed{\begin{matrix}\alpha_j \\ \mu_j\end{matrix}}$$

Lagrange Multipliers

- Dual problem:

$$\max_{\alpha,\mu} \min_{\mathbf{w},b,\{\xi_j\}} L(\mathbf{w}, b, \xi, \alpha, \mu)$$
$$s.t. \alpha_j \geq 0 \quad \forall j$$
$$\mu_j \geq 0 \quad \forall j$$

Dual SVM – non-separable case

$$\text{maximize}_\alpha \quad \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i . \mathbf{x}_j$$

$$\sum_i \alpha_i y_i = 0$$

$$\boxed{C \geq} \alpha_i \geq 0$$

comes from $\frac{\partial L}{\partial \xi} = 0$

Intuition:

If C→∞, recover hard-margin SVM

Dual problem is also QP

Solution gives α_js →

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $C > \alpha_k > 0$

So why solve the dual SVM?

- There are some quadratic programming algorithms that can solve the dual faster than the primal, (specially in high dimensions d>>n)

- But, more importantly, the "**kernel trick**"!!! (later!)