Support Vector Machines (SVMs)

Aarti Singh

Machine Learning 10-315
Oct 21, 2020

MACHINE LEARNING DEPARTMENT
Carnegie Mellon
School of Computer Science

Discriminative Classifiers

Optimal Classifier:

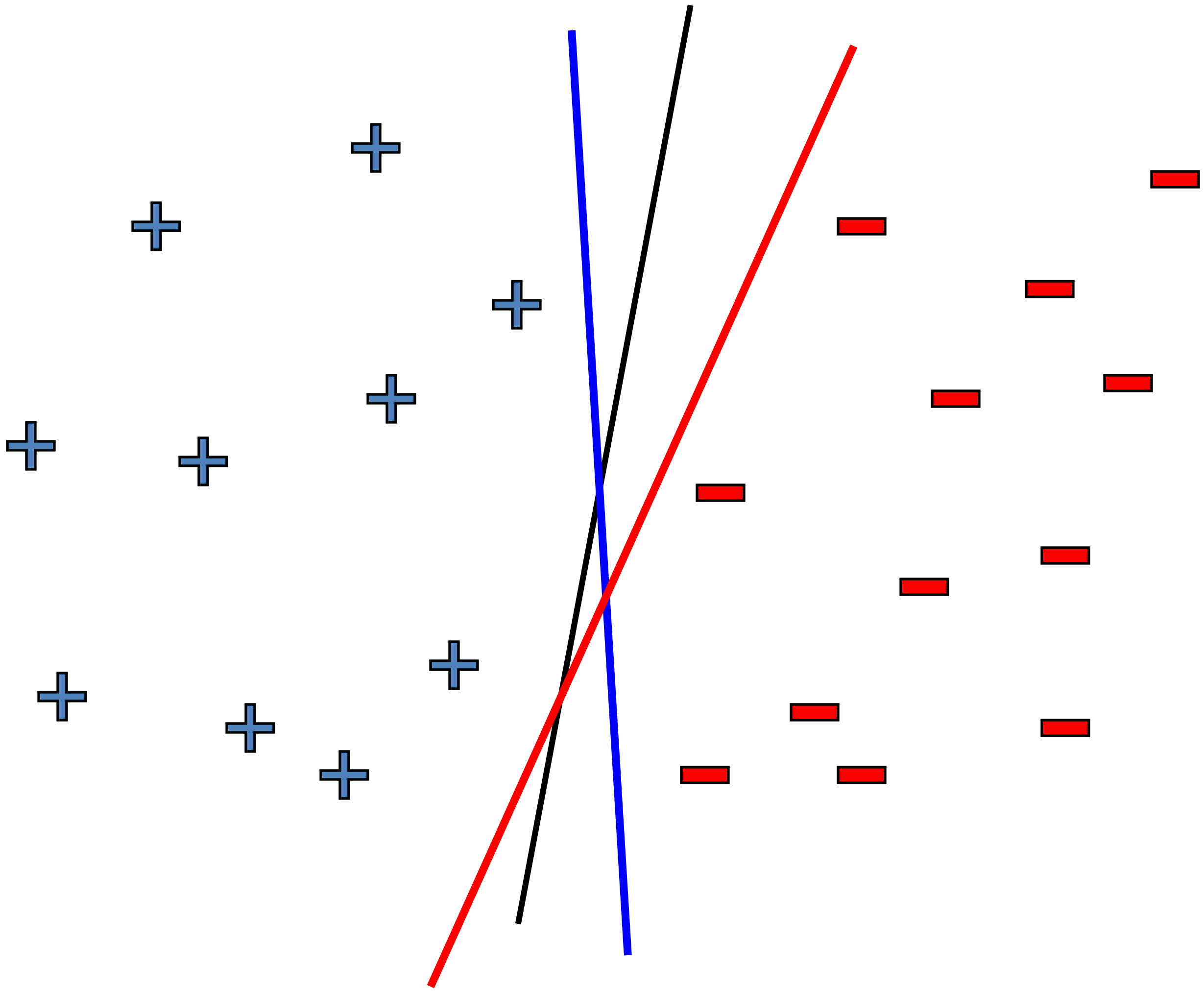
$$f^*(x) = \arg\max_{Y=y} P(Y = y|X = x)$$
$$= \arg\max_{Y=y} P(X = x|Y = y)P(Y = y)$$

Why not learn P(Y|X) directly? Or better yet, why not learn the decision boundary directly?

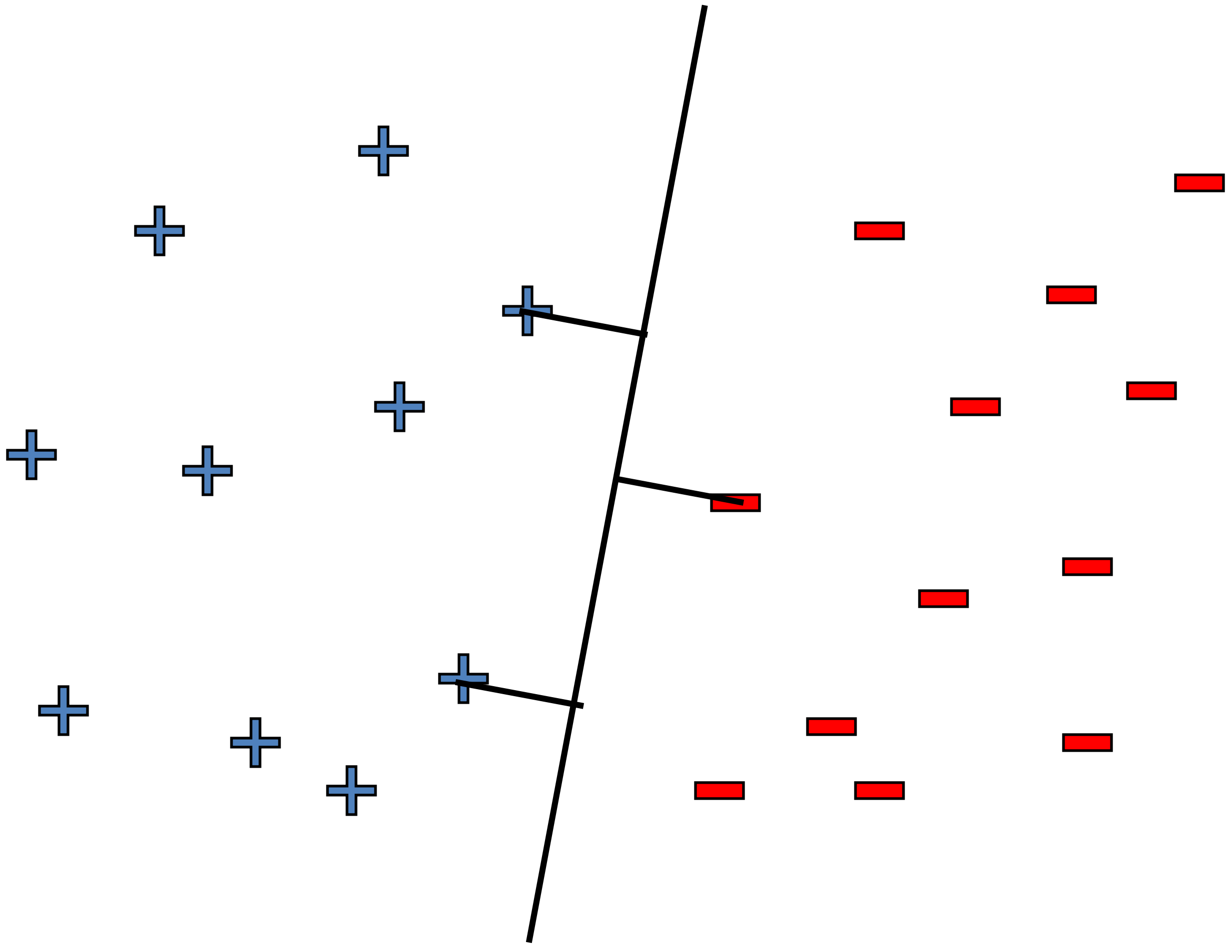
- Assume some functional form for P(Y|X) (e.g. Logistic Regression) or for the decision boundary (e.g. Neural nets, SVMs)
- Estimate parameters of functional form directly from training data

At Pittsburgh G-20 summit …

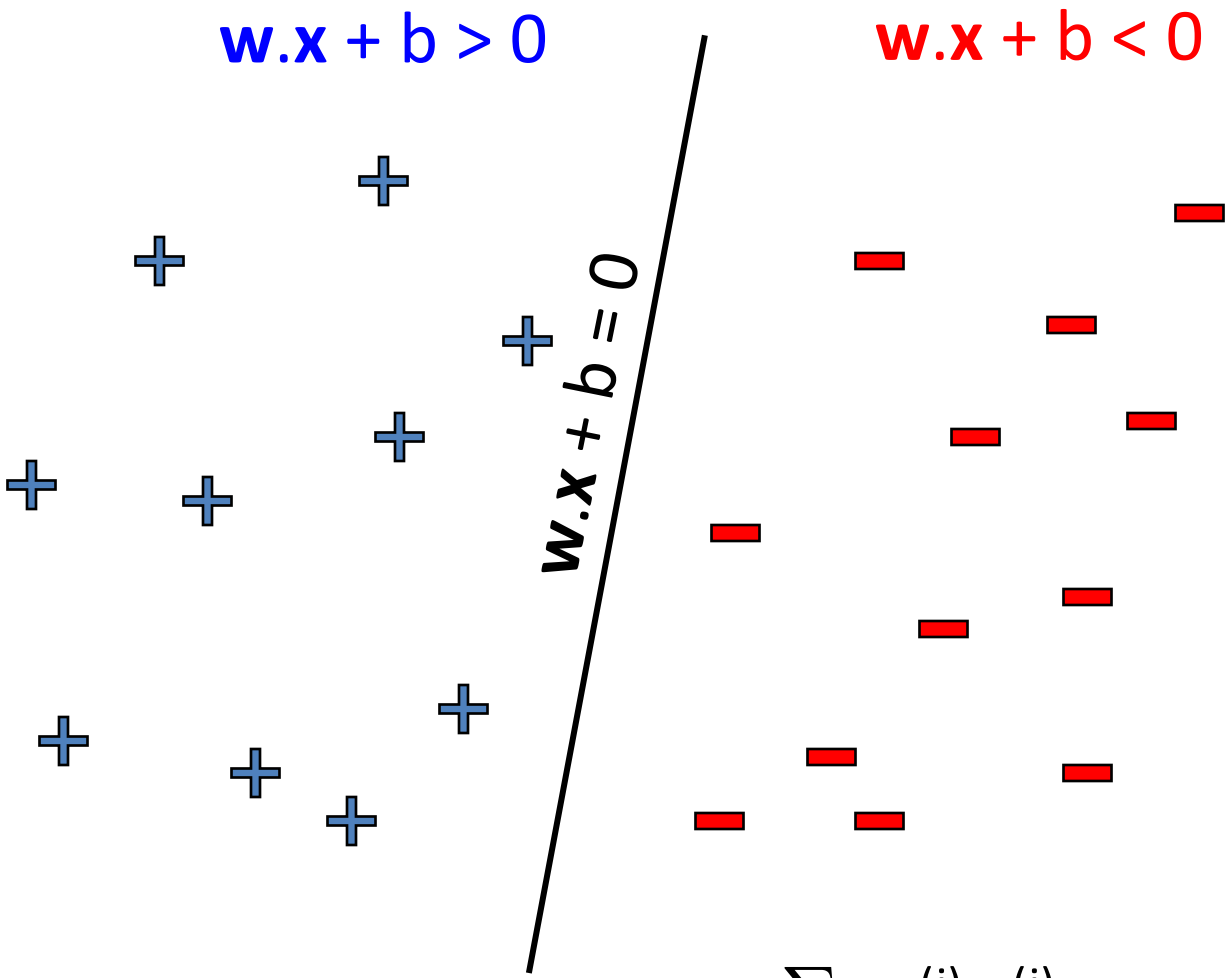
Linear classifiers – which line is better?



Pick the one with the largest margin!

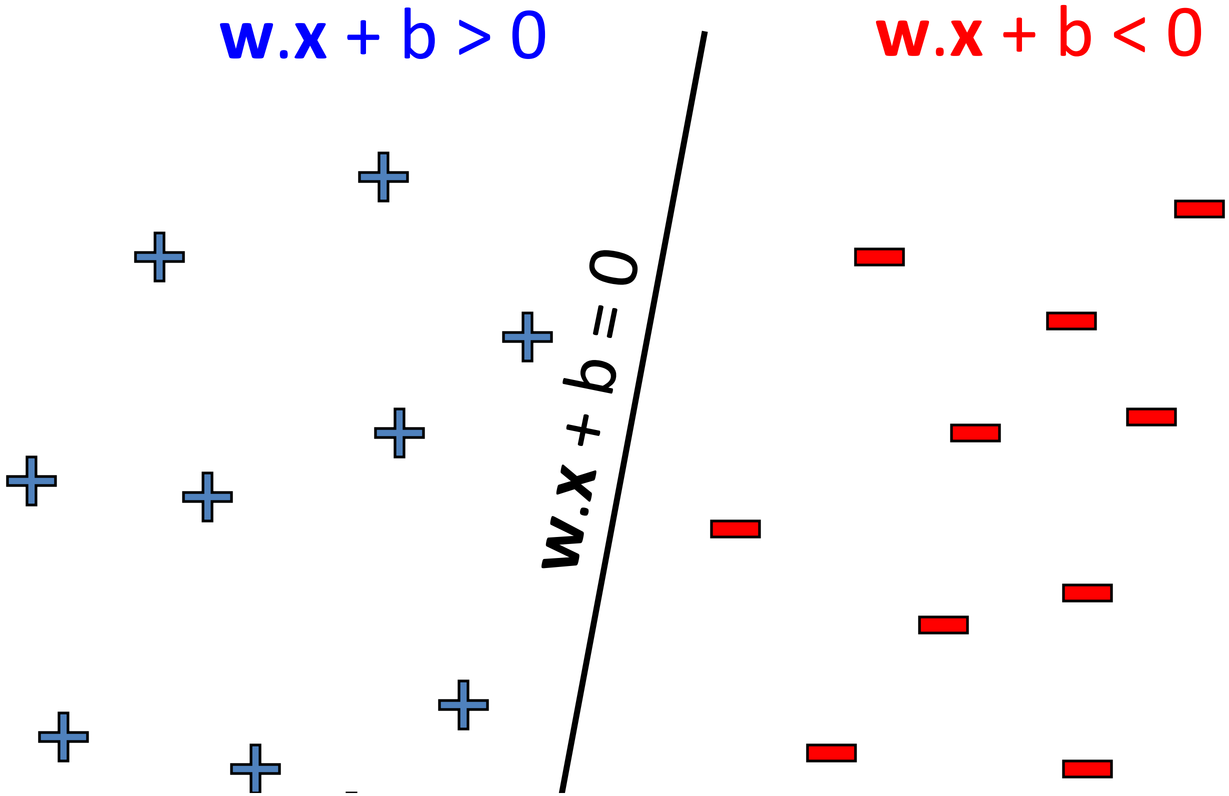


Parameterizing the decision boundary



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

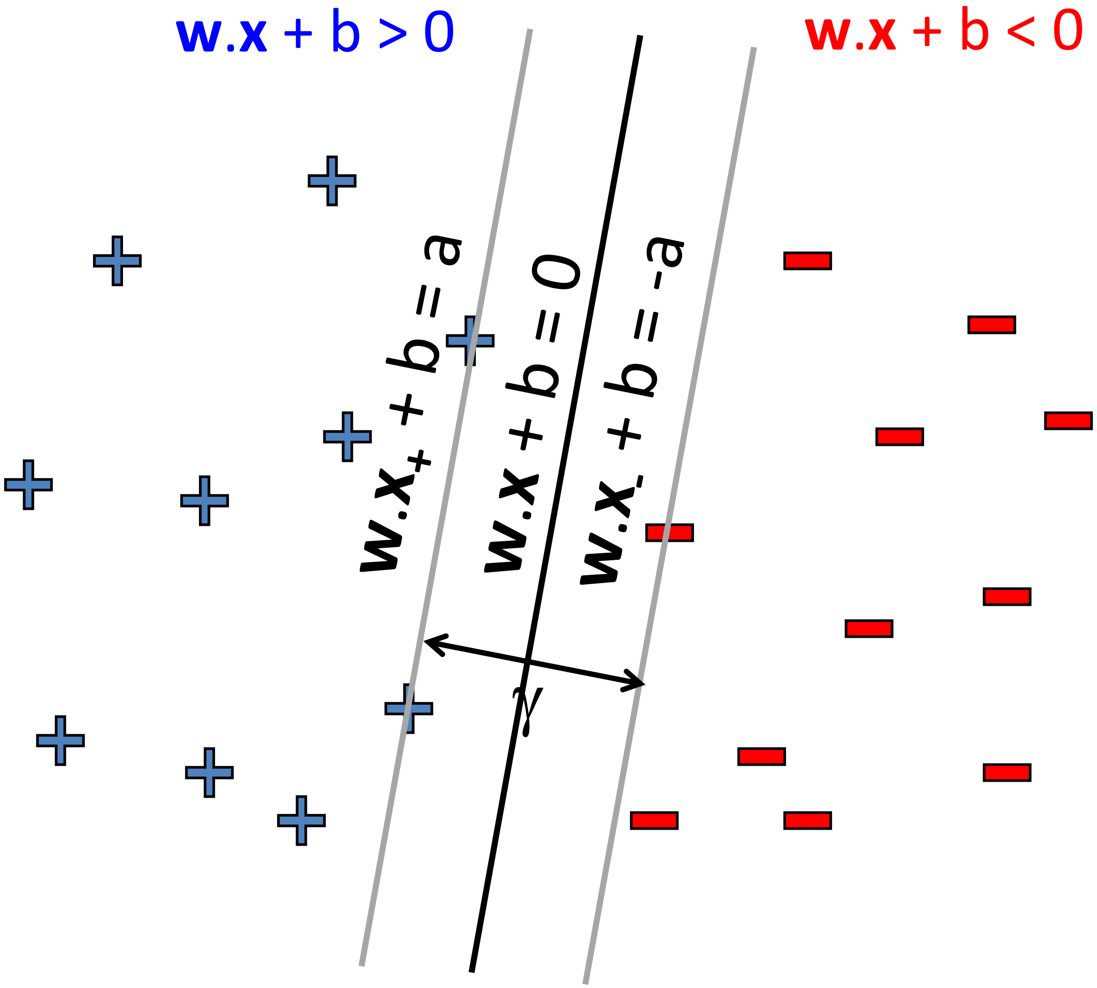
Parameterizing the decision boundary



$y_j \in \{-1, +1\}$ — class

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

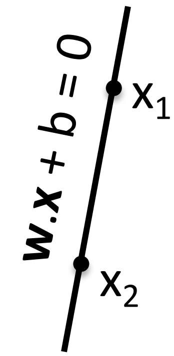
Maximizing the margin



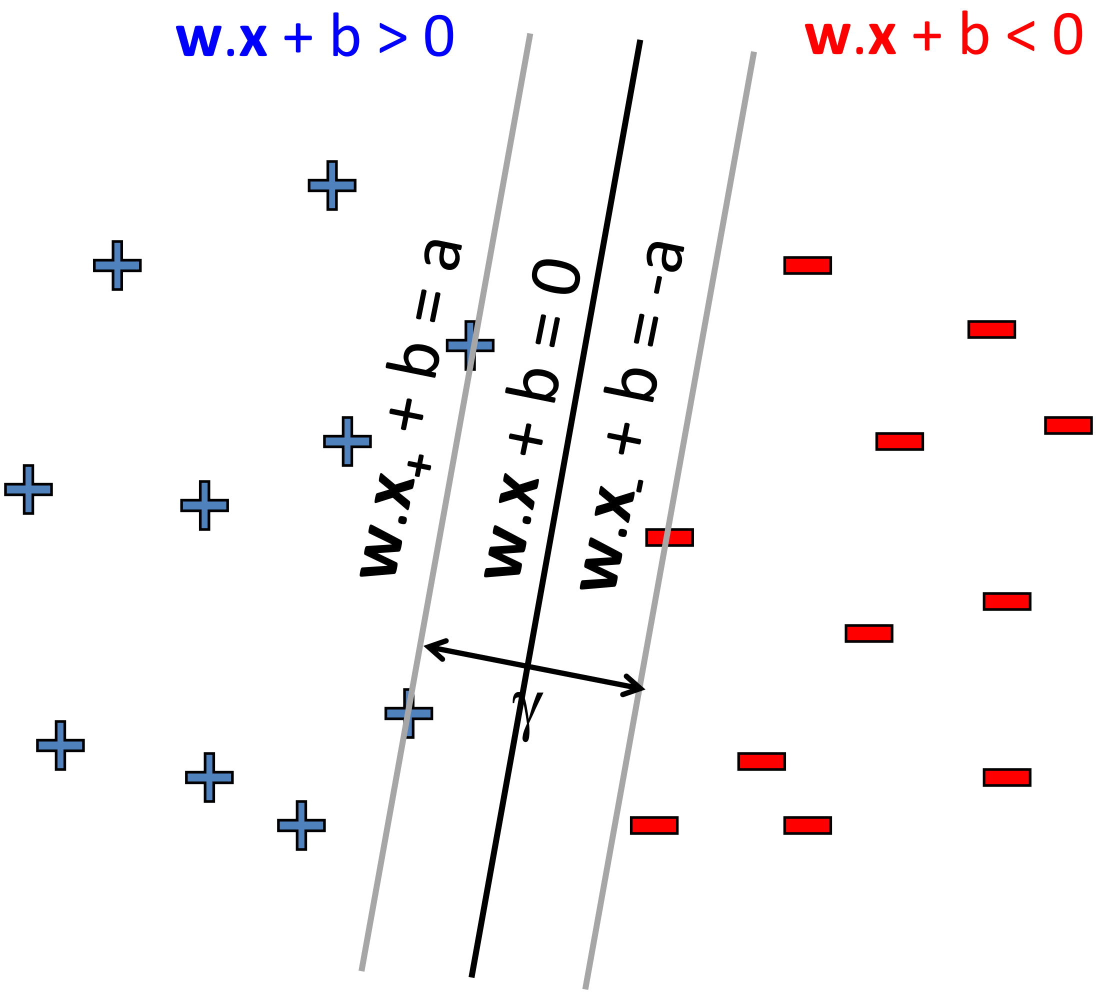
Distance of closest examples from the line/hyperplane

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

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



margin = γ = 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

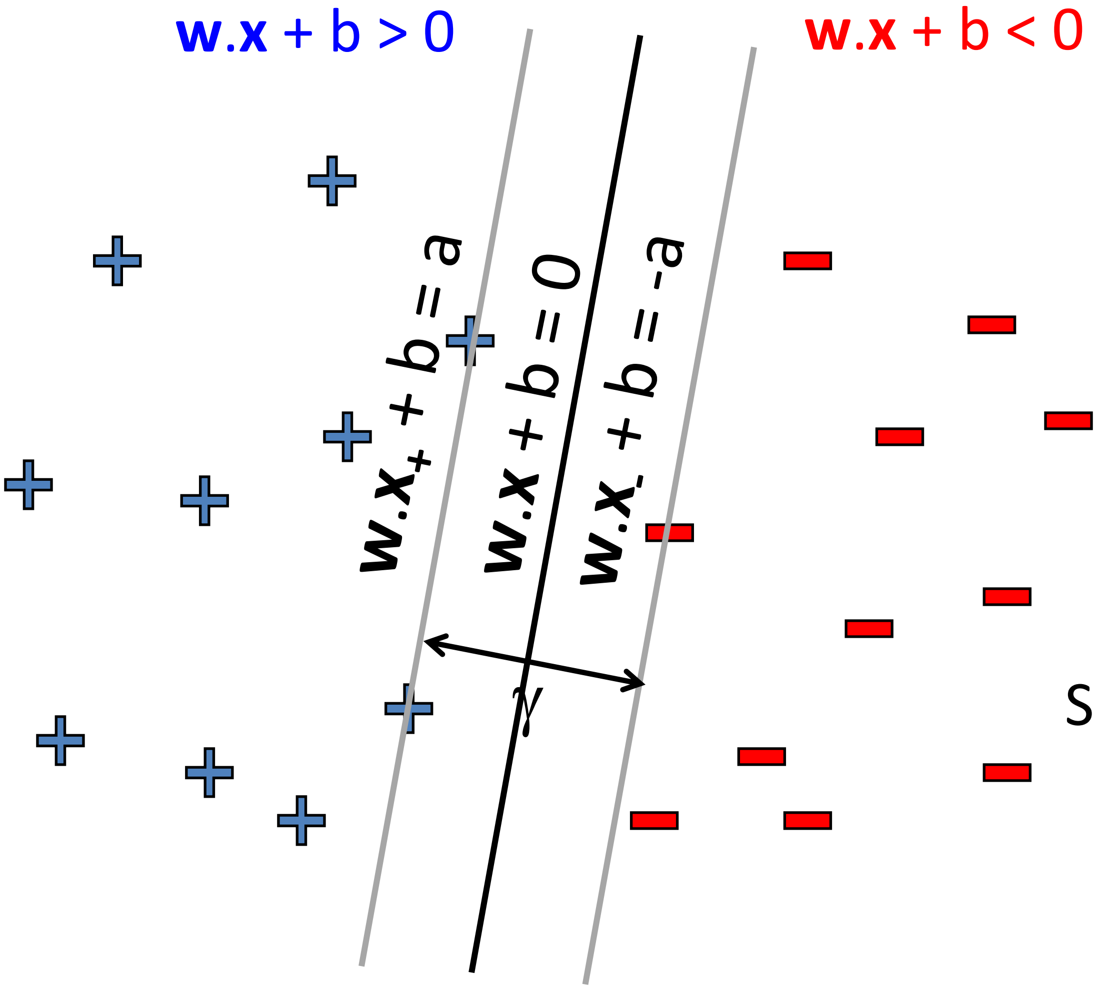
$$\mathbf{x}_+ = \mathbf{x}_- + \gamma \mathbf{w}/\|w\|$$

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

$$a-b = -a-b + \gamma\|w\|$$

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

Maximizing the margin

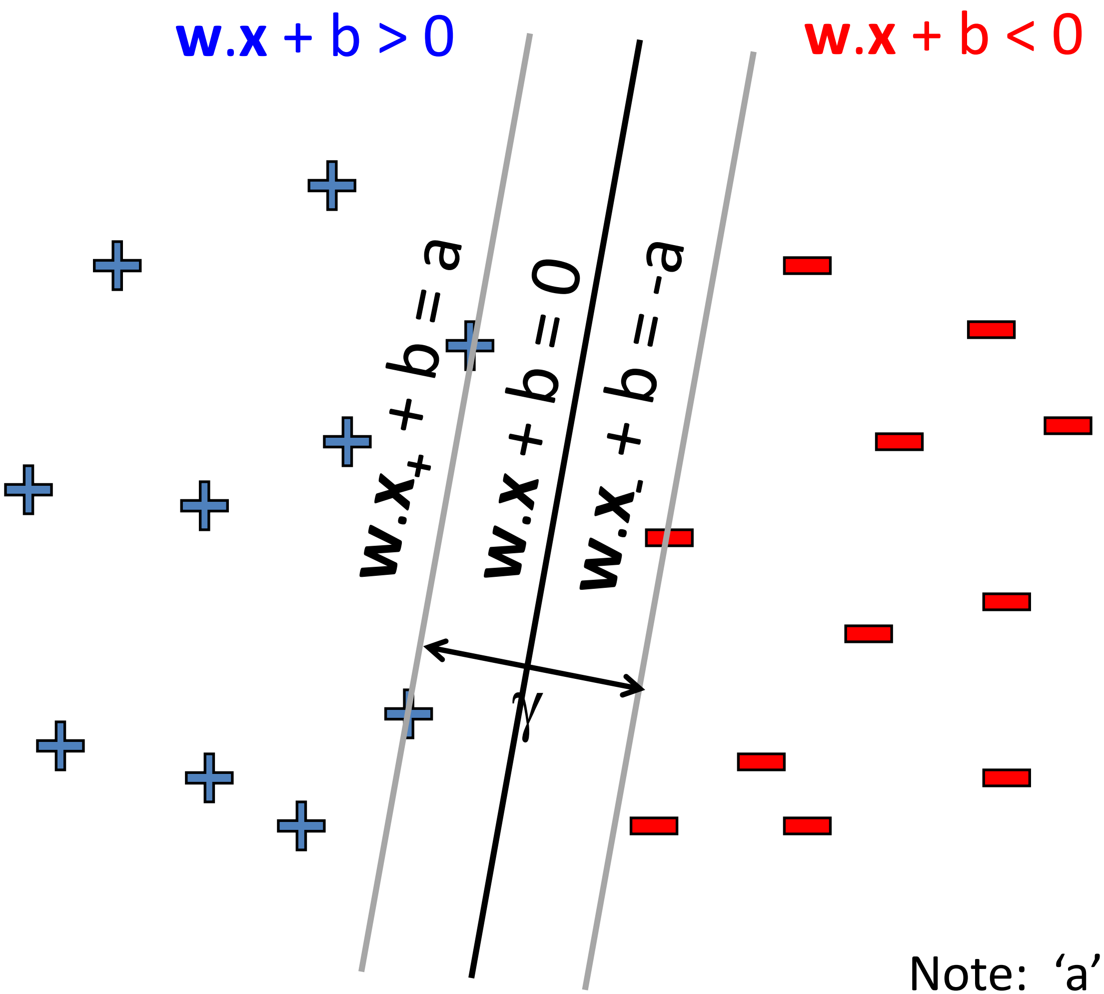


Distance of closest examples from the line/hyperplane

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

Smaller margin ⇔ larger $\|w\|$

Maximizing the margin



Distance of closest examples from the line/hyperplane

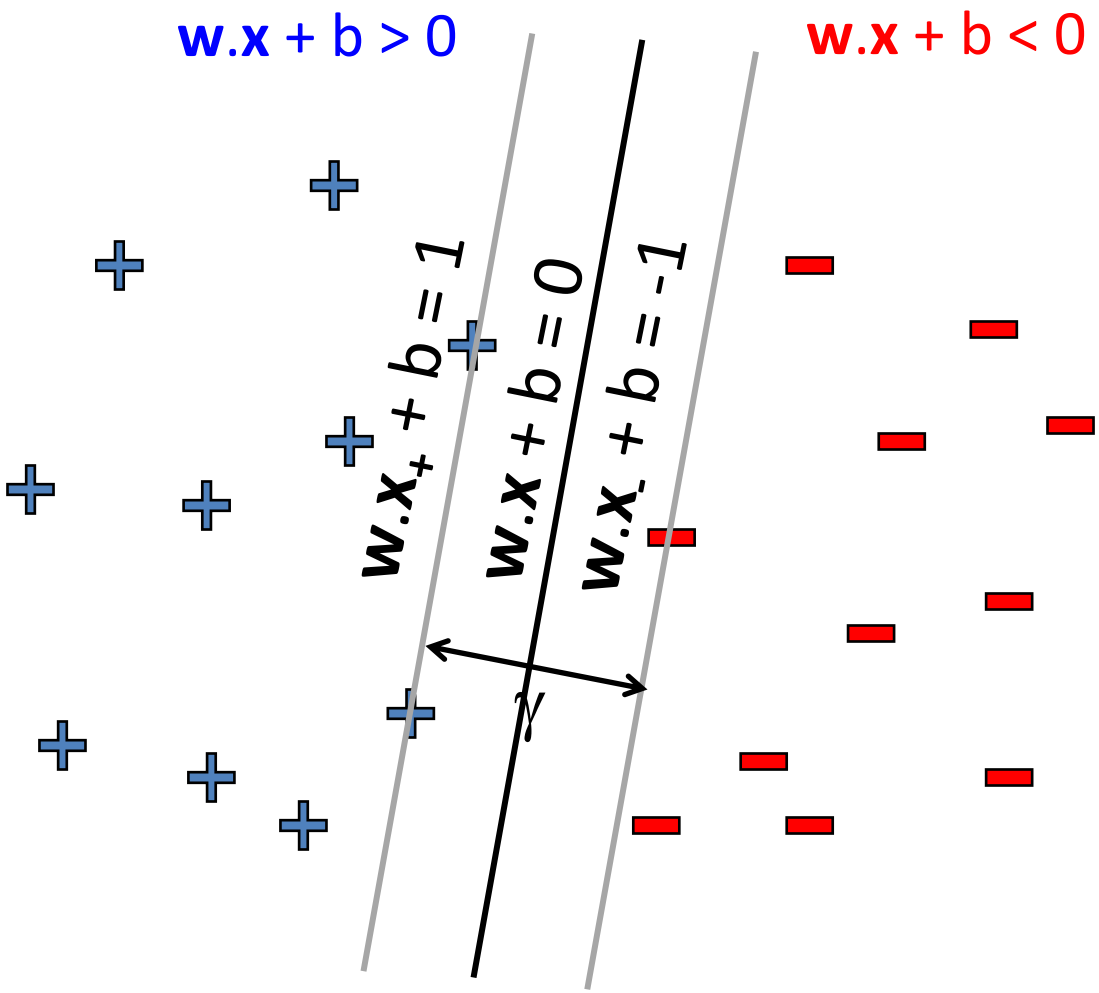
$$\text{margin} = \gamma = 2a/\|w\|$$

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

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

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

Support Vector Machines

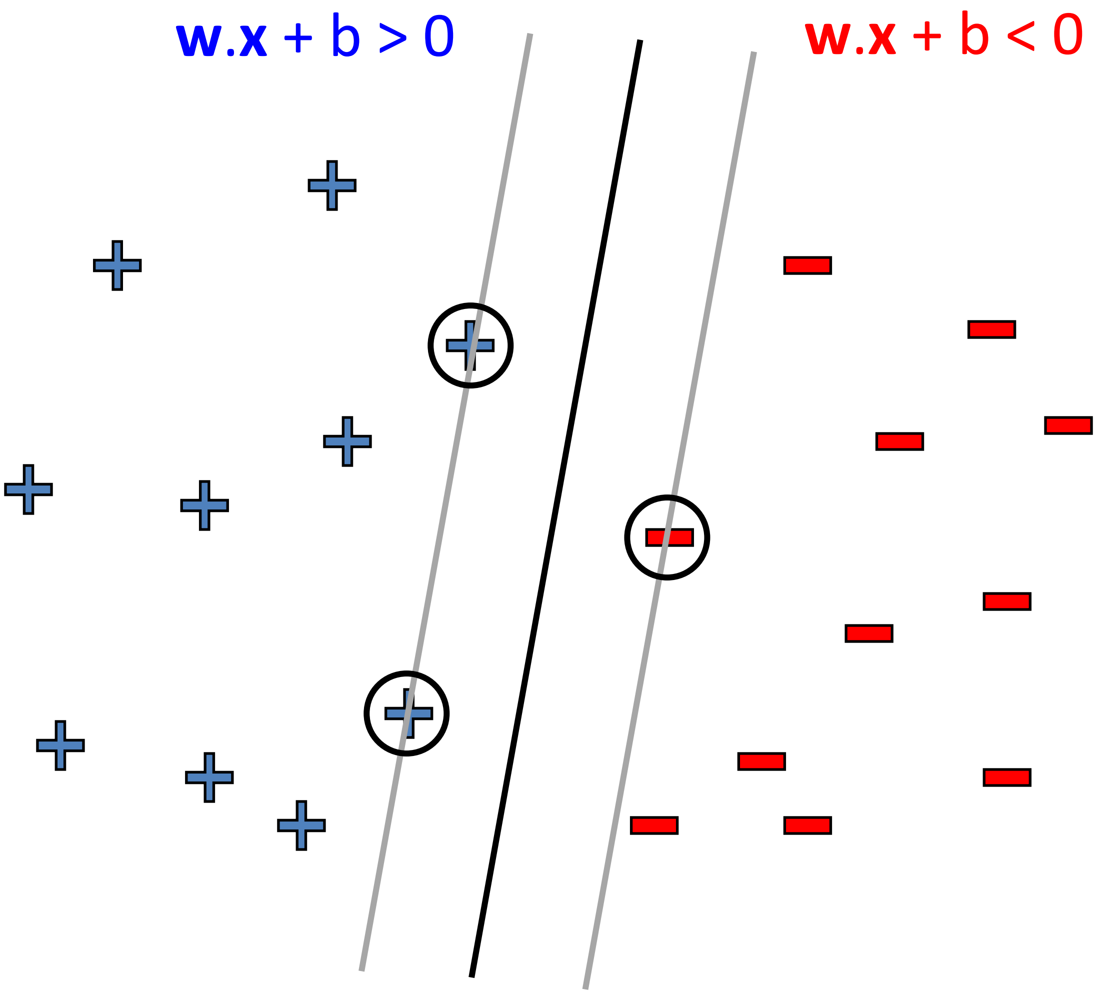


$$\min_{\mathbf{w},b} \ \mathbf{w}.\mathbf{w}$$
$$\text{s.t. } (\mathbf{w}.\mathbf{x}_j + b)\, y_j \geq 1 \quad \forall j$$

Solve efficiently by quadratic programming (QP)

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

Support Vectors



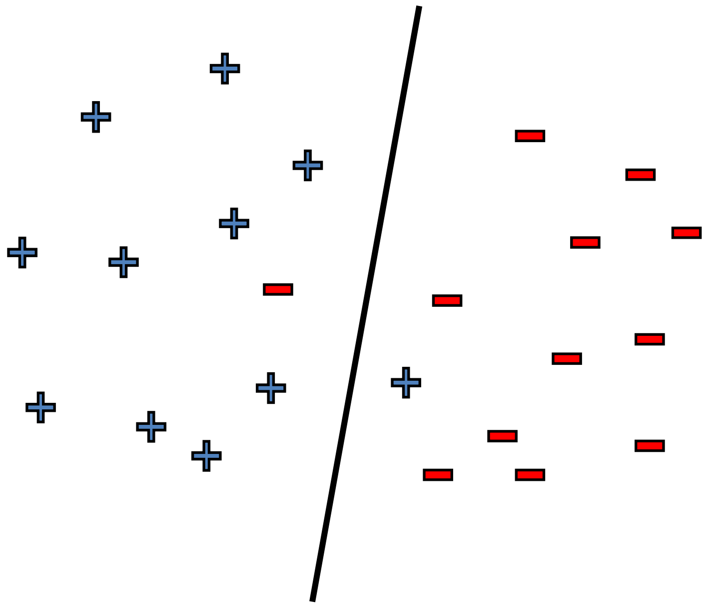
Linear hyperplane defined by "support vectors"

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?



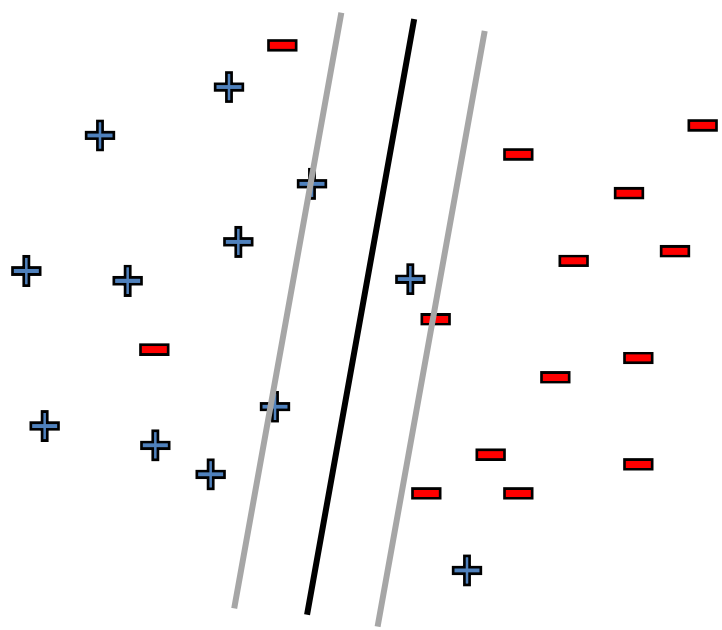
Use features of features of features of features….

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 $\|w\|$

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

Maximize margin and minimize # mistakes on training data

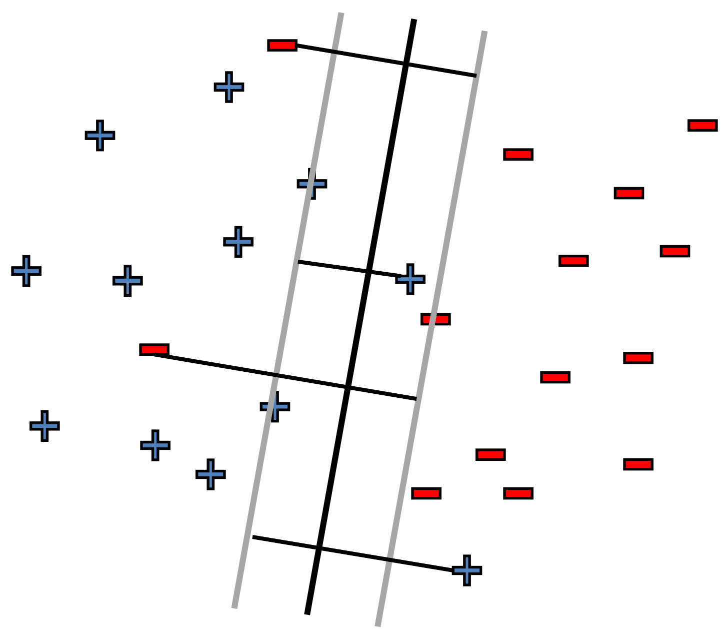
C - tradeoff parameter

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

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

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

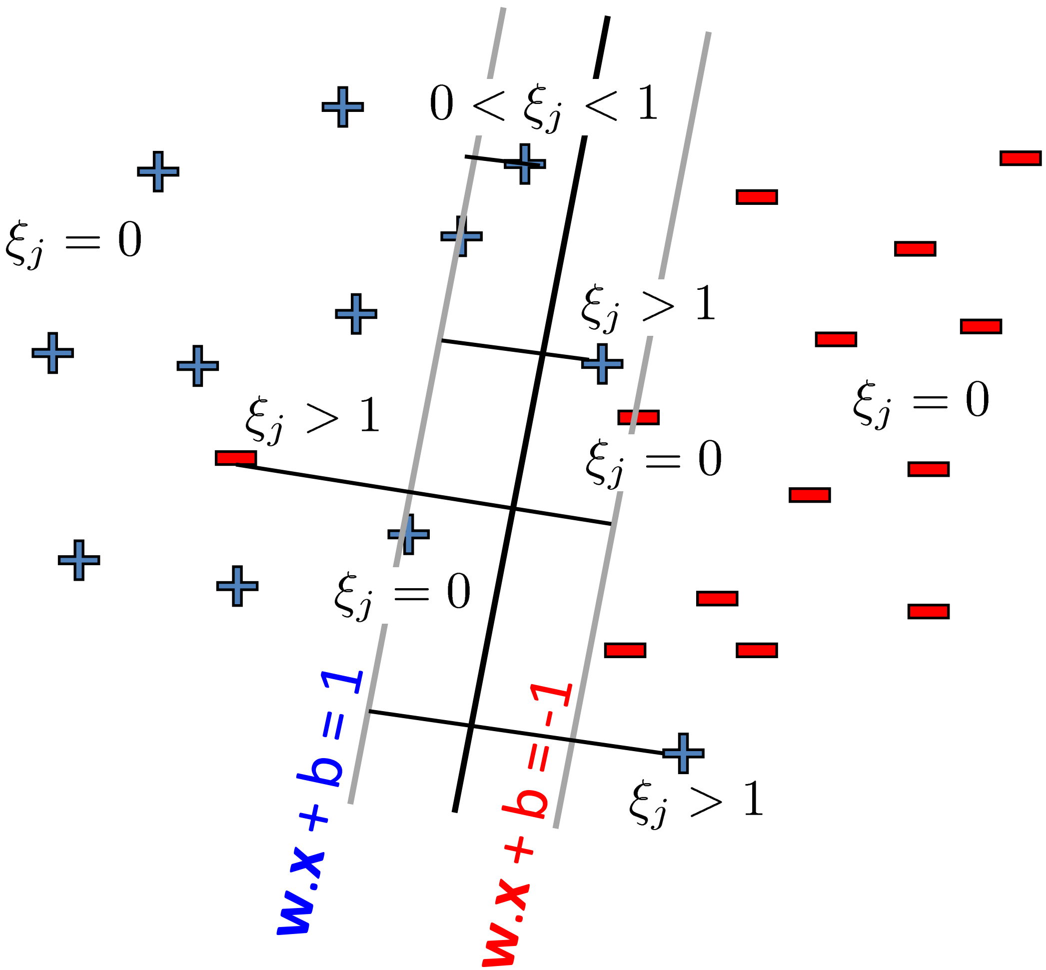
$$\xi_j \geq 0 \quad \forall j$$

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:

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

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

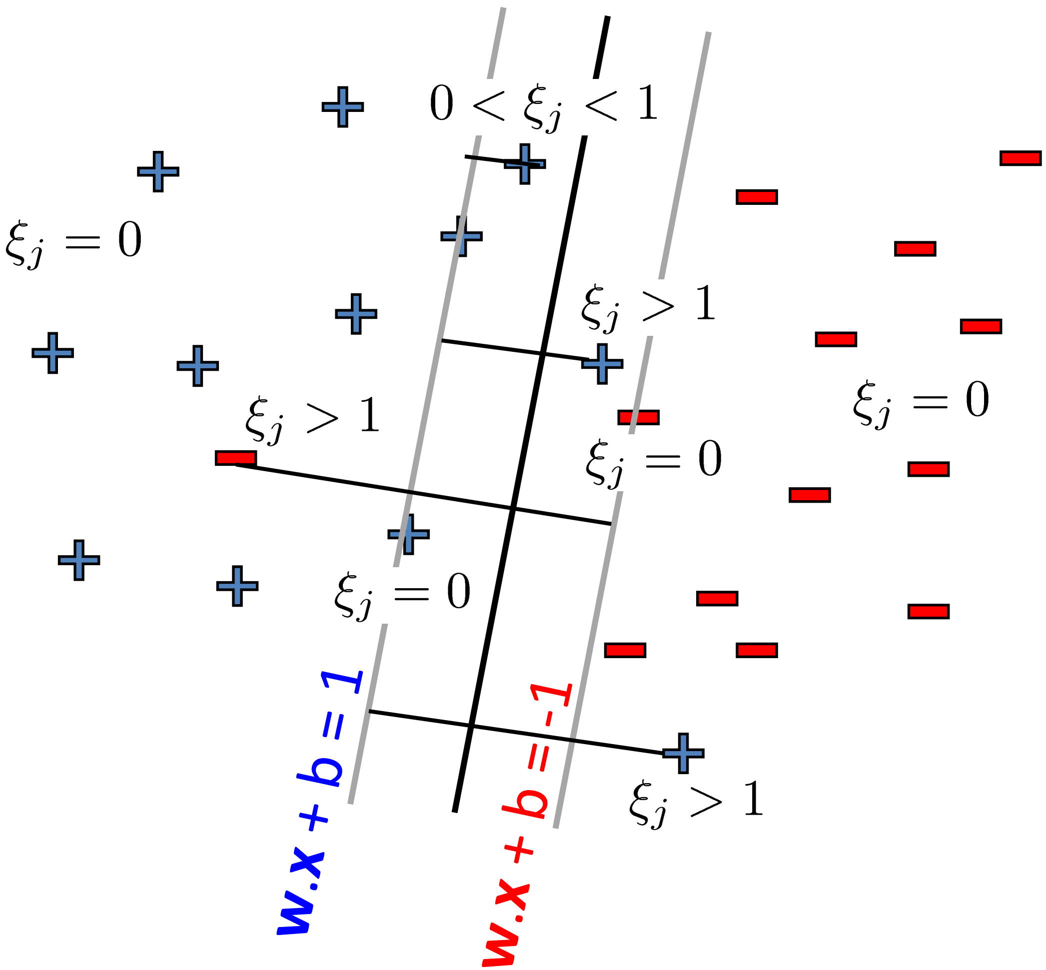
Penalty for misclassifying:

$$C\, \xi_j$$

How do we recover hard margin SVM?

Set $C = \infty$

Slack variables – Hinge loss

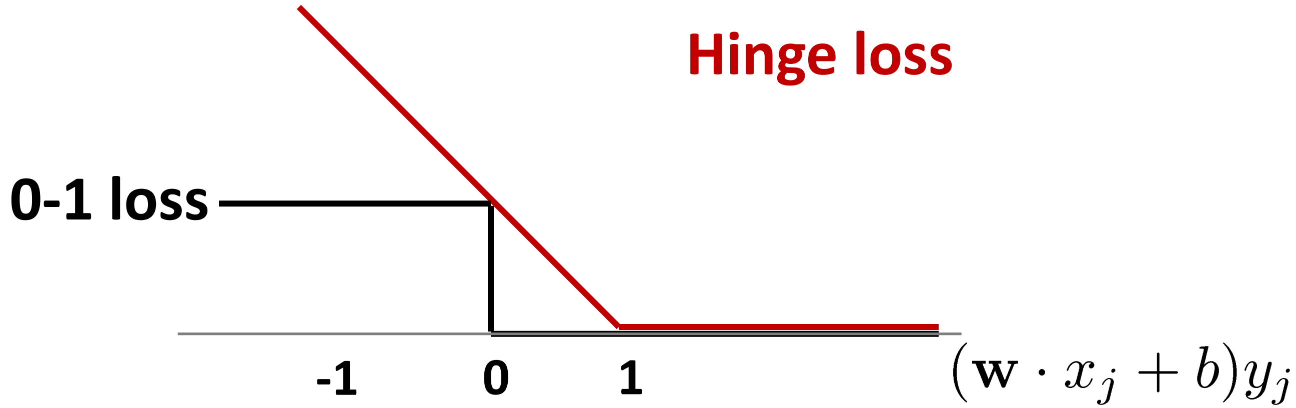


Notice that

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

Slack variables – Hinge loss

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



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

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

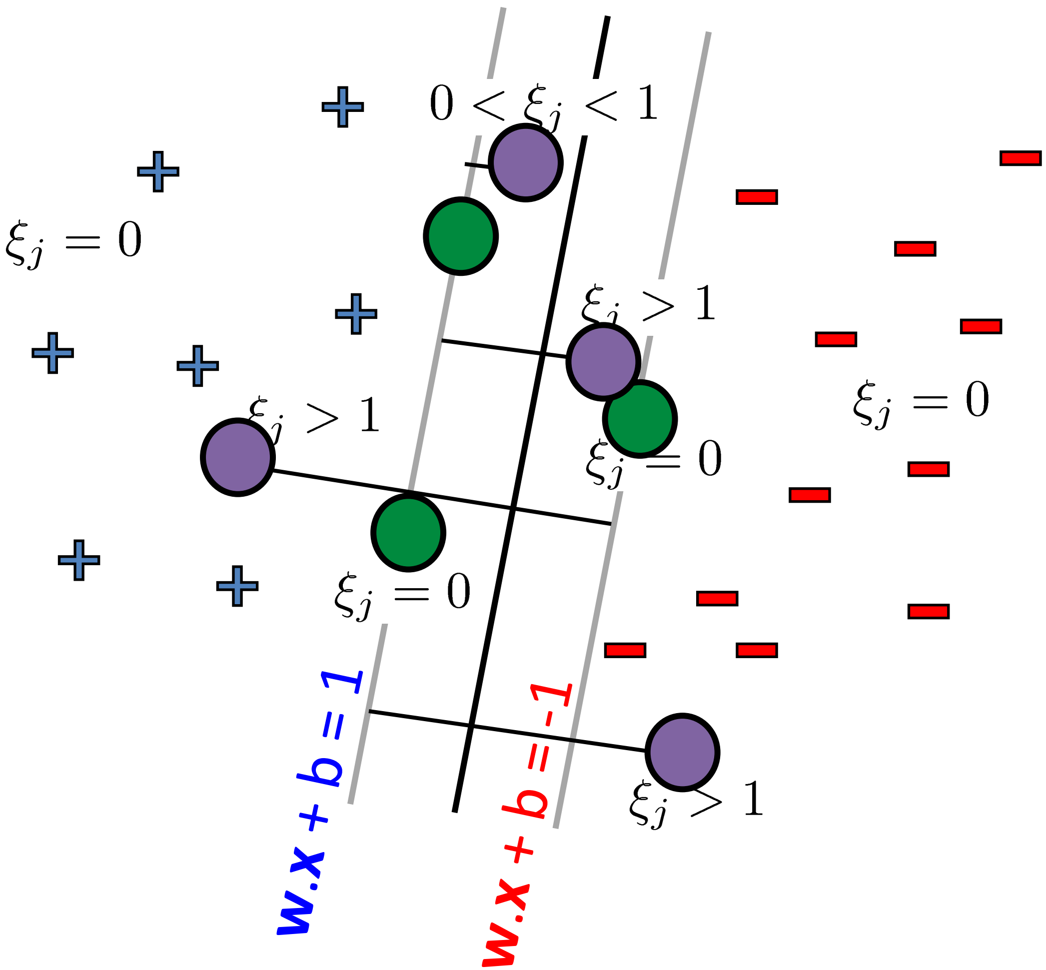
$$\xi_j \geq 0 \quad \forall j$$

⇔

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

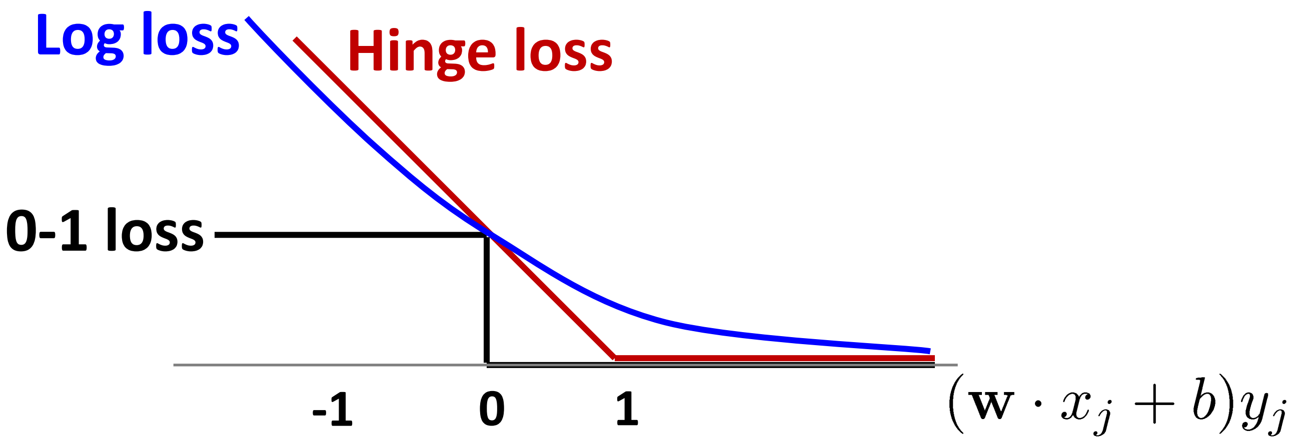
SVM vs. Logistic Regression

SVM : **Hinge loss**

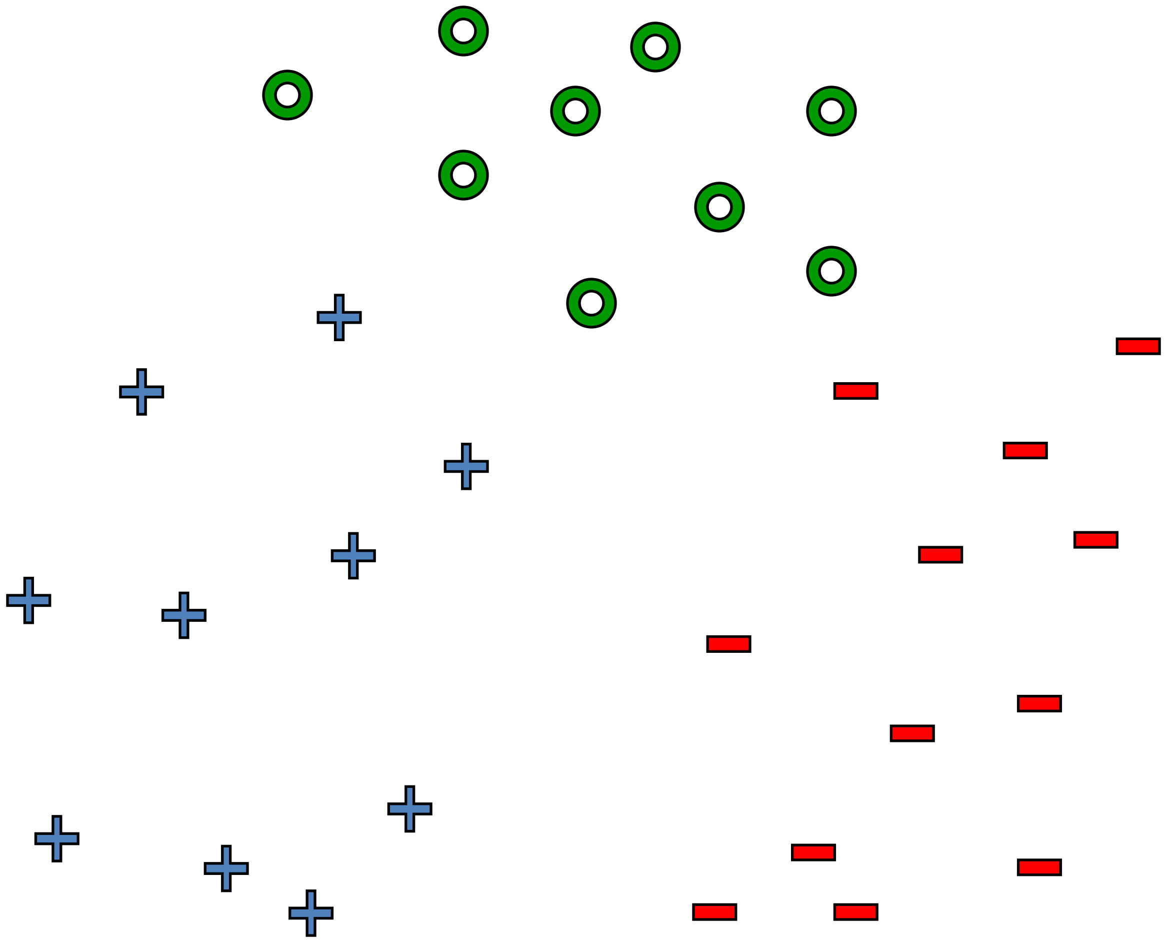
$$\text{loss}(f(x_j), y_j) = (1 - (\mathbf{w} \cdot x_j + b)y_j))_+$$

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

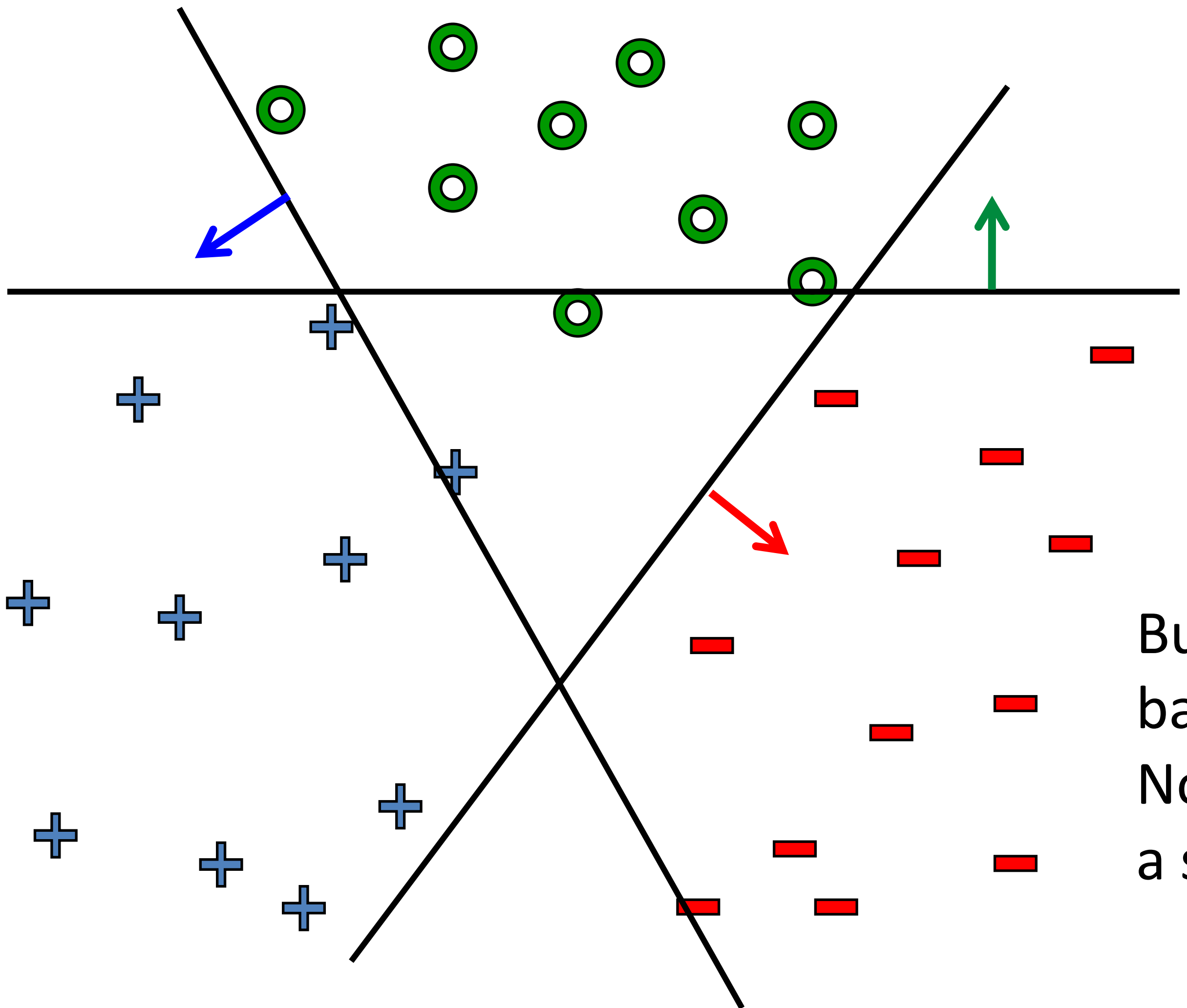
$$\text{loss}(f(x_j), y_j) = -\log P(y_j \mid x_j, \mathbf{w}, b) = \log(1 + e^{-(\mathbf{w} \cdot x_j + b)y_j})$$



What about multiple classes?



One vs. rest



Learn 3 classifiers separately:

Class k vs. rest

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

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

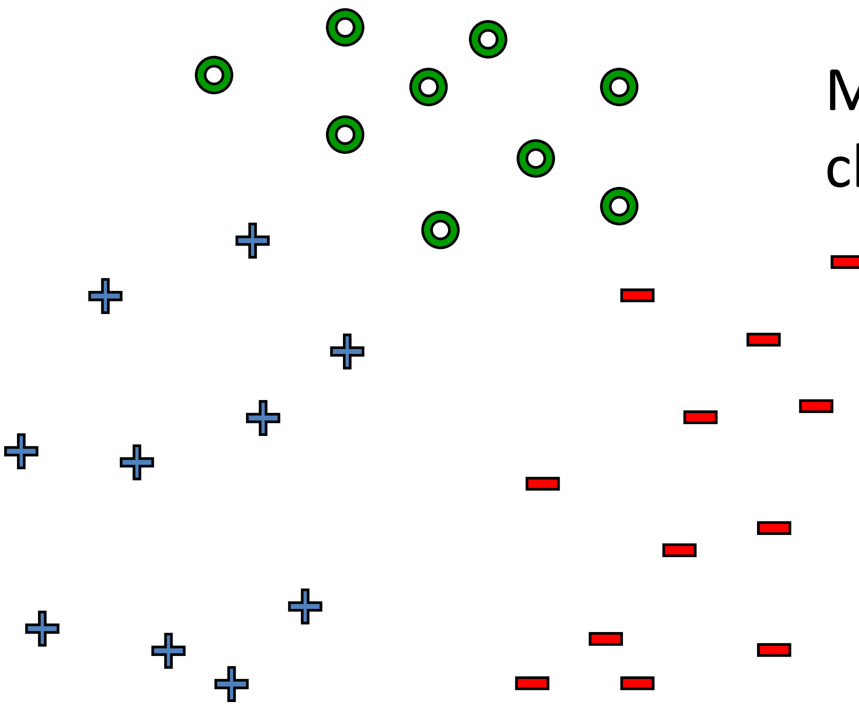
But $\mathbf{w}_k$s may not be based on the same scale.
Note: $(a\mathbf{w}).x + (ab)$ is also a solution

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

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

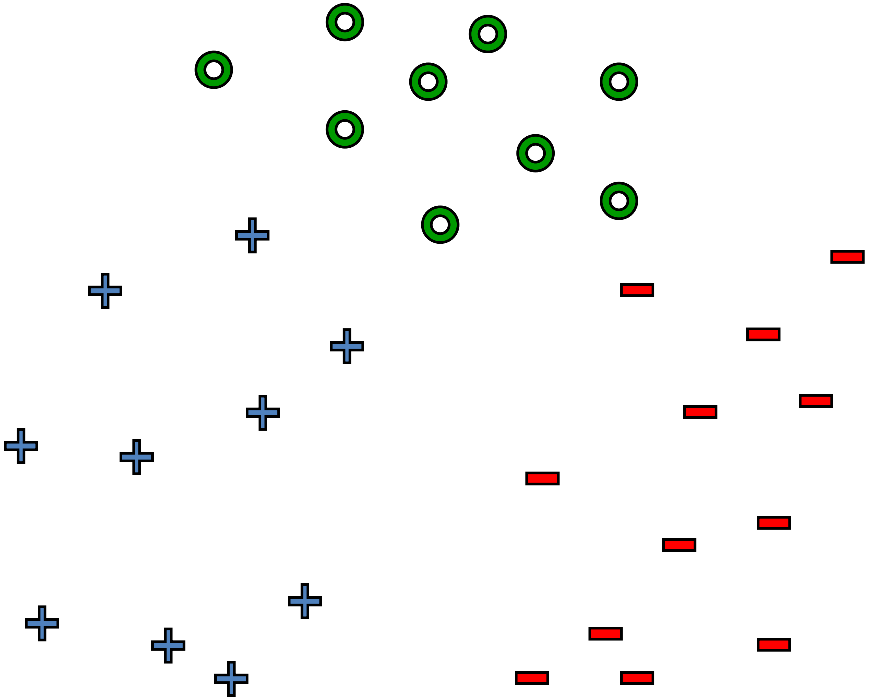
Learn 1 classifier: Multi-class SVM

Simultaneously learn 3 sets of weights

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

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

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



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

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