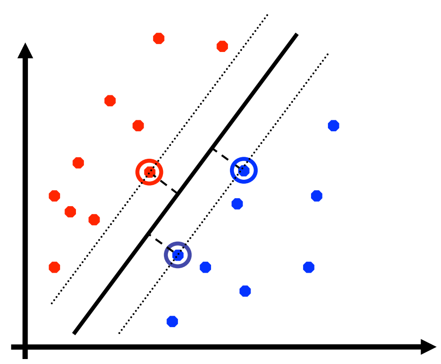
SVMs and Kernel Methods
Lecture 3

David Sontag
New York University

Slides adapted from Luke Zettlemoyer, Vibhav Gogate, and Carlos Guestrin

Today's lecture

- Dual form of soft-margin SVM
- Feature mappings & kernels
- Convexity, Mercer's theorem
- (Time permitting) Extensions:
  - Imbalanced data
  - Multi-class
  - Other loss functions
  - L1 regularization



Recap of dual SVM derivation

(Dual) $$\max_{\vec{\alpha}\geq 0} \; \min_{\vec{w},b} \; \frac{1}{2}||\vec{w}||^2 - \sum_j \alpha_j \left[(\vec{w}\cdot\vec{x}_j + b)\, y_j - 1 \right]$$

Can solve for optimal **w**, b as function of α:

$$\frac{\partial L}{\partial w} = w - \sum_j \alpha_j y_j x_j \quad \rightarrow \quad \mathbf{w} = \sum_j \alpha_j y_j \mathbf{x}_j$$

$$\frac{\partial L}{\partial b} = -\sum_j \alpha_j y_j \quad \rightarrow \quad \sum_j \alpha_j y_j = 0$$

Substituting these values back in (and simplifying), we obtain:

(Dual) $$\max_{\vec{\alpha}\geq 0,\, \sum_j \alpha_j y_j = 0} \; \sum_j \alpha_j - \frac{1}{2} \sum_{i,j} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j \right)$$

So, in dual formulation we will solve for α directly!

- **w** and b are computed from α (if needed)

Solving for the offset "b"

Lagrangian:

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

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

$\alpha_j > 0$ for some *j* implies constraint is tight. We use this to obtain *b*:

$$y_j \left(\vec{w} \cdot \vec{x}_j + b\right) = 1 \quad (1)$$

$$y_j y_j \left(\vec{w} \cdot \vec{x}_j + b\right) = y_j \quad (2)$$

$$\left(\vec{w} \cdot \vec{x}_j + b\right) = y_j \quad (3)$$

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

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

for any k where $\alpha_k > 0$

Dual formulation only depends on dot-products of the features!

$$\max_{\vec{\alpha}\geq 0,\ \sum_j \alpha_j y_j = 0} \ \sum_j \alpha_j - \frac{1}{2}\sum_{i,j} y_i y_j \alpha_i \alpha_j \left(\vec{x}_i \cdot \vec{x}_j\right)$$

First, we introduce a *feature mapping*:

$$\mathbf{x}_i \mathbf{x}_j \quad \rightarrow \quad \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

Next, replace the dot product with an equivalent *kernel* function:

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

$$K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$$

$$\textstyle\sum_i \alpha_i y_i = 0 \qquad \vec{\alpha} \geq 0$$

Do kernels need to be symmetric?

Classification rule using dual solution

$$y \leftarrow \text{sign}(\vec{w} \cdot \vec{x} + b)$$

Using dual solution

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \underbrace{(\vec{x}_i \cdot \vec{x})} + b\right]$$

dot product of feature vectors of new example with support vectors

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

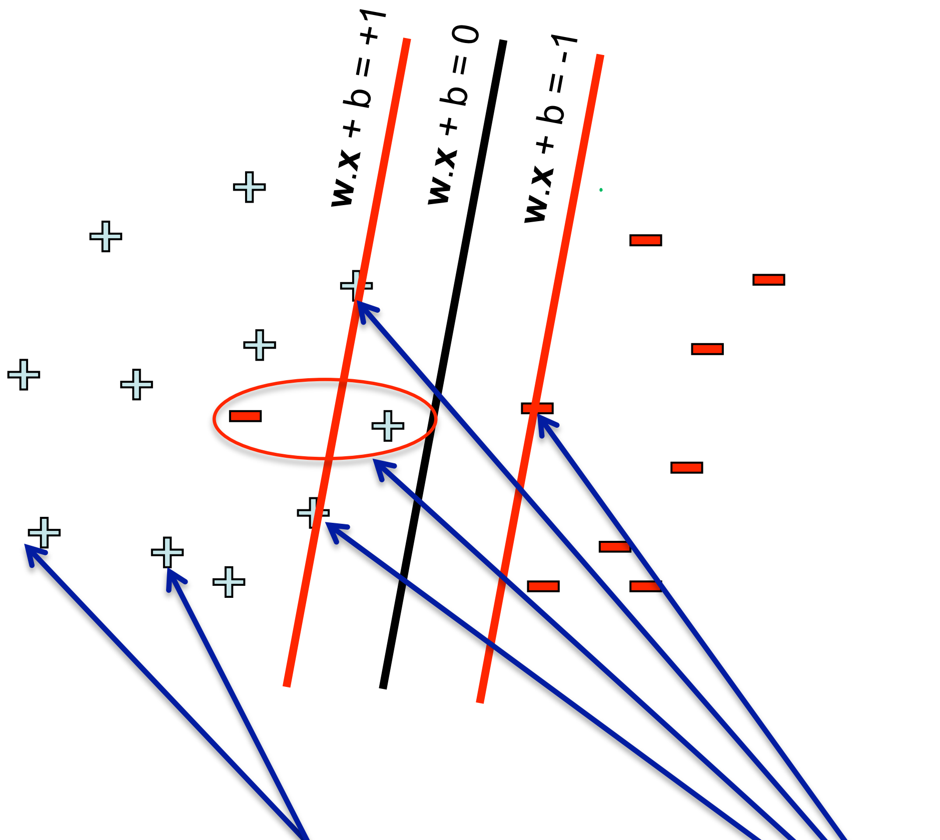
$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

for any k where $\alpha_k > 0$

Using a kernel function, predict with…

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i K(x_i, x) + b\right]$$

Dual SVM interpretation: Sparsity



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

Final solution tends to be sparse

- α_j=0 for most j
- don't need to store these points to compute w or make predictions

Non-support Vectors:

- α_j=0
- moving them will not change w

Support Vectors:

- $\alpha_j \geq 0$

Soft-margin SVM

Primal:

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

Solve for w,b, α :

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$
$$b = y_k - \mathbf{w}.\mathbf{x}_k$$

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

Dual:

$$\text{maximize}_{\alpha} \quad \sum_i \alpha_i - \tfrac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \mathbf{x}_j$$
$$\sum_i \alpha_i y_i = 0$$
$$C \geq \alpha_i \geq 0$$

What changed?

- Added upper bound of C on α_i!
- Intuitive explanation:
  - Without slack, $\alpha_i \rightarrow \infty$ when constraints are violated (points misclassified)
  - Upper bound of C limits the α_i, so misclassifications are allowed

Common kernels

- Polynomials of degree exactly *d*

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v})^d$$

- Polynomials of degree up to *d*

$$K(\mathbf{u},\mathbf{v}) = (\mathbf{u}\cdot\mathbf{v} + 1)^d$$

- Gaussian kernels

$$K(\vec{u},\vec{v}) = \exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right)$$

- Sigmoid

$$K(\mathbf{u},\mathbf{v}) = \tanh(\eta\mathbf{u}\cdot\mathbf{v} + \nu)$$

- And many others: very active area of research!

Polynomial kernel

d=1

$$\phi(u).\phi(v) = \begin{pmatrix} u_1 \\ u_2 \end{pmatrix} . \begin{pmatrix} v_1 \\ v_2 \end{pmatrix} = u_1 v_1 + u_2 v_2 = u.v$$

d=2

$$\phi(u).\phi(v) = \begin{pmatrix} u_1^2 \\ u_1 u_2 \\ u_2 u_1 \\ u_2^2 \end{pmatrix} . \begin{pmatrix} v_1^2 \\ v_1 v_2 \\ v_2 v_1 \\ v_2^2 \end{pmatrix} \begin{aligned} &= u_1^2 v_1^2 + 2 u_1 v_1 u_2 v_2 + u_2^2 v_2^2 \\ &= (u_1 v_1 + u_2 v_2)^2 \\ &= (u.v)^2 \end{aligned}$$

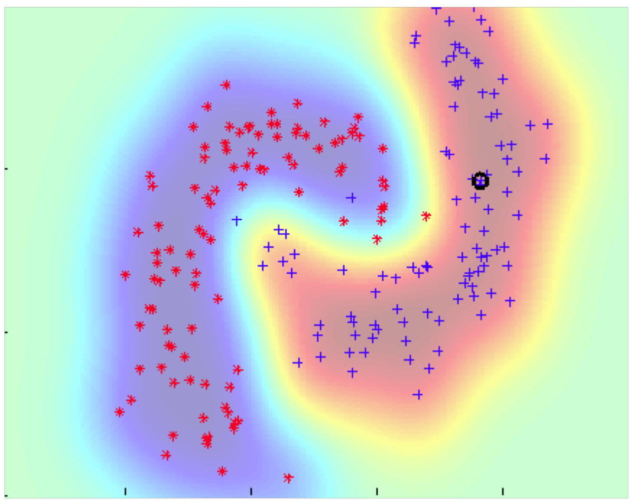
For any *d (we will skip proof):*

$$\phi(u).\phi(v) = (u.v)^d$$

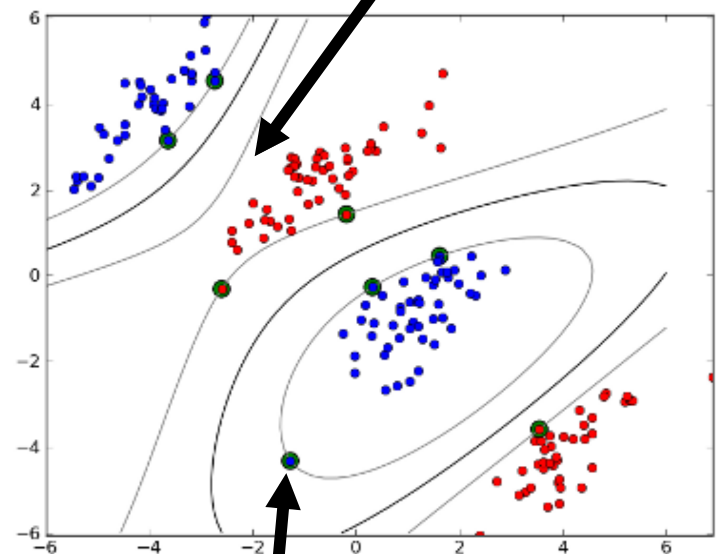
Polynomials of degree **exactly** *d*

Gaussian kernel

$$K(\vec{u},\vec{v}) = \exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right)$$



Level sets, i.e. $w \cdot \phi(x) = r$ for some r



Support vectors

$$y \leftarrow \text{sign}\left[\sum_i \alpha_i y_i \exp\left(-\frac{\|\vec{x}-\vec{x}_i\|_2^2}{2\sigma^2}\right) + b\right]$$

[Cynthia Rudin]

[mblondel.org]

Kernel algebra

kernel composition	feature composition
a) $k(\mathbf{x},\mathbf{v}) = k_a(\mathbf{x},\mathbf{v}) + k_b(\mathbf{x},\mathbf{v})$	$\boldsymbol{\phi}(\mathbf{x}) = (\boldsymbol{\phi}_a(\mathbf{x}), \boldsymbol{\phi}_b(\mathbf{x})),$
b) $k(\mathbf{x},\mathbf{v}) = f k_a(\mathbf{x},\mathbf{v}),\ f > 0$	$\boldsymbol{\phi}(\mathbf{x}) = \sqrt{f}\boldsymbol{\phi}_a(\mathbf{x})$
c) $k(\mathbf{x},\mathbf{v}) = k_a(\mathbf{x},\mathbf{v}) k_b(\mathbf{x},\mathbf{v})$	$\phi_m(\mathbf{x}) = \phi_{ai}(\mathbf{x})\phi_{bj}(\mathbf{x})$
d) $k(\mathbf{x},\mathbf{v}) = \mathbf{x}^T A \mathbf{v}$, A positive semi-definite	$\boldsymbol{\phi}(\mathbf{x}) = L^T\mathbf{x}$, where $A = LL^T$.
e) $k(\mathbf{x},\mathbf{v}) = f(\mathbf{x})f(\mathbf{v})k_a(\mathbf{x},\mathbf{v})$	$\phi(\mathbf{x}) = f(\mathbf{x})\phi_a(\mathbf{x})$

Q: How would you prove that the "Gaussian kernel" is a valid kernel?
A: Expand the Euclidean norm as follows:

$$\exp\left(-\frac{||\vec{u}-\vec{v}||_2^2}{2\sigma^2}\right) = \exp\left(-\frac{||\vec{u}||_2^2}{2\sigma^2}\right)\exp\left(-\frac{||\vec{v}||_2^2}{2\sigma^2}\right)\exp\left(\frac{\vec{u}\cdot\vec{v}}{\sigma^2}\right)$$

Then, apply (e) from above

To see that this is a kernel, use the Taylor series expansion of the exponential, together with repeated application of (a), (b), and (c):

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$$

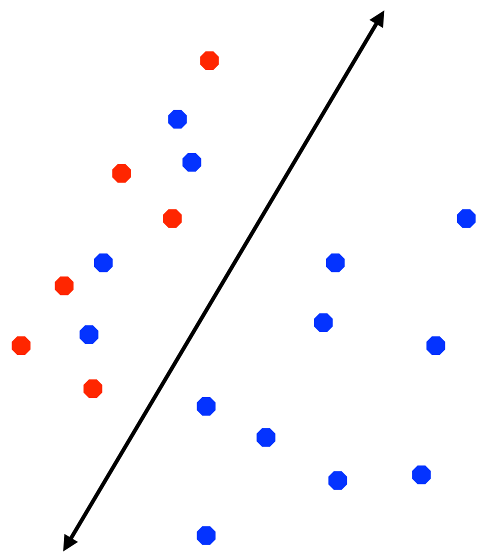
The feature mapping is infinite dimensional!

[Justin Domke]

Overfitting?

- Huge feature space with kernels: should we worry about overfitting?
  - SVM objective seeks a solution with large **margin**
    - Theory says that large margin leads to good generalization (we will see this in a couple of lectures)
  - But everything overfits sometimes!!!
  - Can control by:
    - Setting C
    - Choosing a better Kernel
    - Varying parameters of the Kernel (width of Gaussian, etc.)

How to deal with imbalanced data?



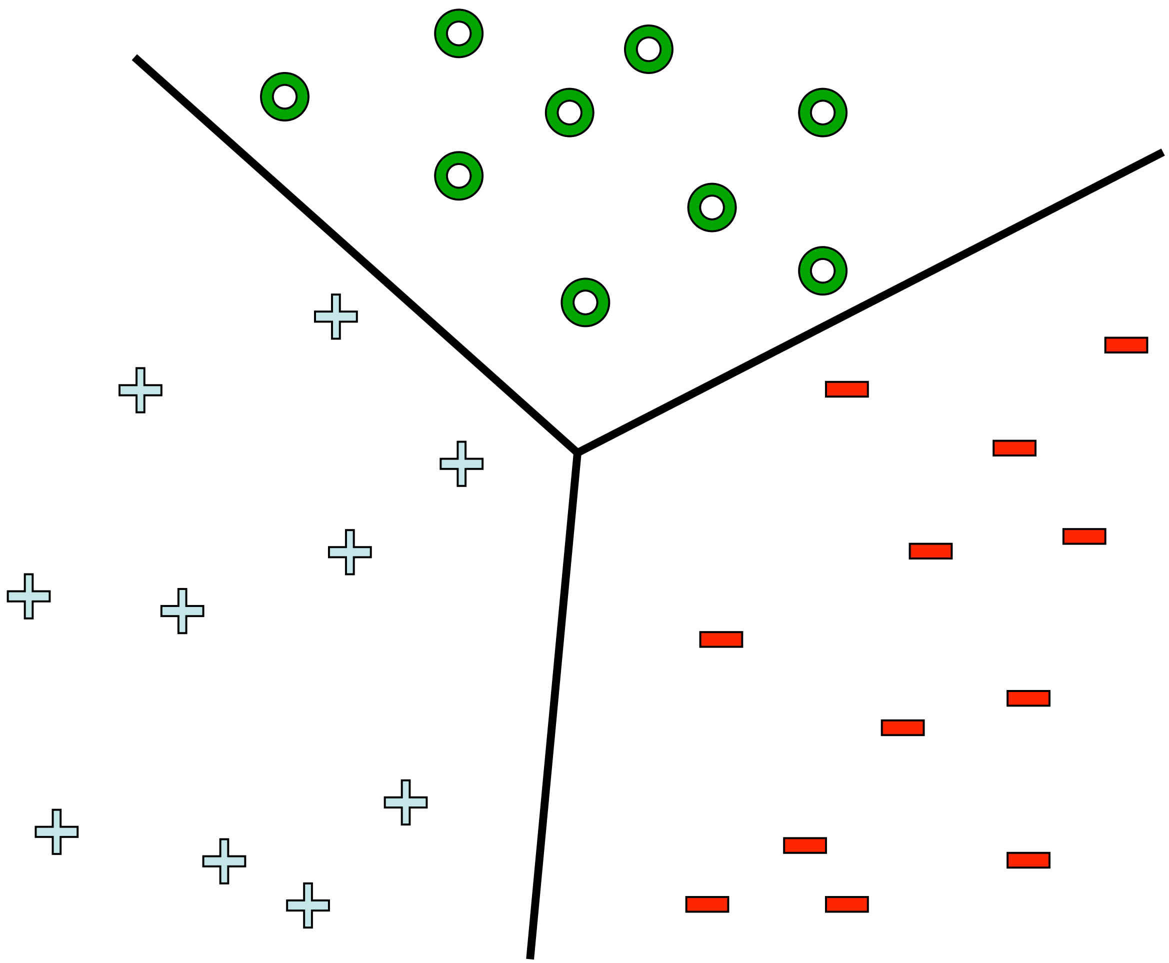
- In many practical applications we may have **imbalanced** data sets
- We may want errors to be equally distributed between the positive and negative classes
- A slight modification to the SVM objective does the trick!

$$N = N_+ + N_-$$

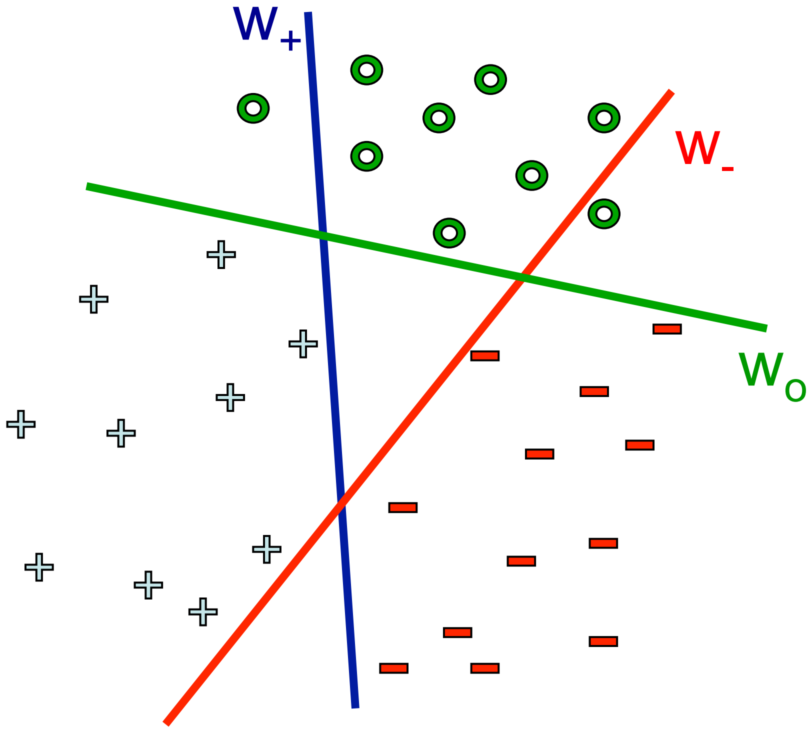
$$\min_{w,b} \ ||w||_2^2 + \frac{CN}{2N_+} \sum_{j:y_j=+1} \xi_j + \frac{CN}{2N_-} \sum_{j:y_j=-1} \xi_j$$

Class-specific weighting of the slack variables

How do we do multi-class classification?



One versus all classification



Learn 3 classifiers:

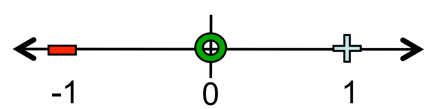
- \- vs {o,+}, weights w_-
- \+ vs {o,-}, weights w_+
- o vs {+,-}, weights w_o

Predict label using:

$$\hat{y} \leftarrow \arg\max_k \; w_k \cdot x + b_k$$

Any problems?

Could we learn this (1-D) dataset? →

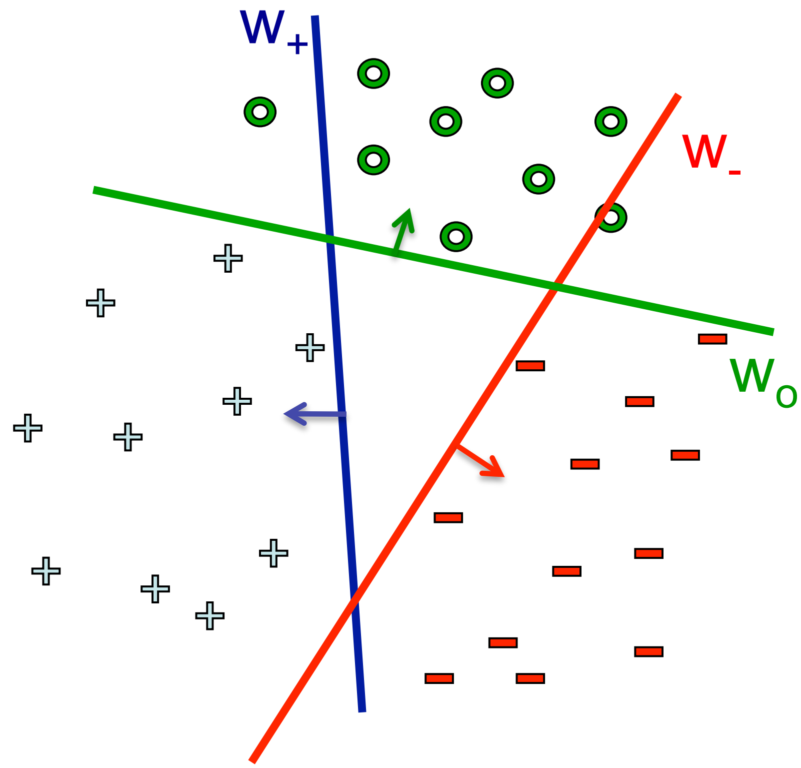


Multi-class SVM

Simultaneously learn 3 sets of weights:

•How do we guarantee the correct labels?

•Need new constraints!



The “score” of the correct class must be better than the “score” of wrong classes:

$$w^{(y_j)} \cdot x_j + b^{(y_j)} > w^{(y)} \cdot x_j + b^{(y)} \qquad \forall j,\ y \neq y_j$$

Multi-class SVM

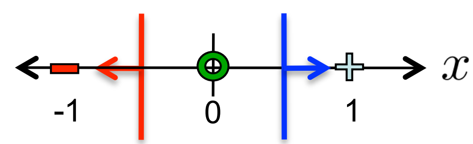
As for the SVM, we introduce slack variables and maximize margin:

$$\text{minimize}_{\mathbf{w},b} \quad \sum_y \mathbf{w}^{(y)}.\mathbf{w}^{(y)} + C\sum_j \xi_j$$
$$\mathbf{w}^{(y_j)}.\mathbf{x}_j + b^{(y_j)} \geq \mathbf{w}^{(y')}.\mathbf{x}_j + b^{(y')} + 1 - \xi_j, \ \forall y' \neq y_j, \ \forall j$$
$$\xi_j \geq 0, \ \forall j$$

To predict, we use:

$$\hat{y} \leftarrow \arg\max_k \ w_k \cdot x + b_k$$

Now can we learn it? →



$w_- = -1$

$b_- = -.5$

$w_+ = 1$

$b_+ = -.5$

$w_o = 0$

$b_o = .001$