Support vector machines
Lecture 4

David Sontag
New York University

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

Q: What does the Perceptron mistake bound tell us?

Theorem: The maximum number of mistakes made by the perceptron algorithm is bounded above by R^2/γ^2

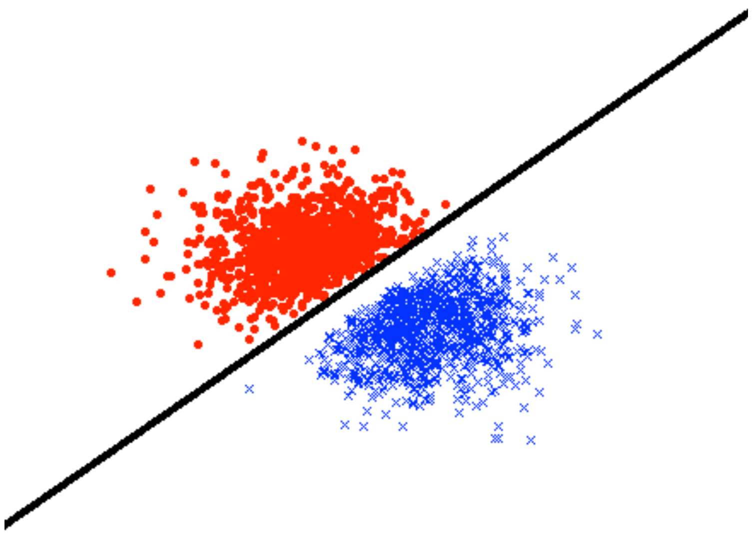
- **Batch learning:** setting we consider for most of the class.
  - Assume training data drawn from same distribution as future test data
  - Use training data to find the hypothesis

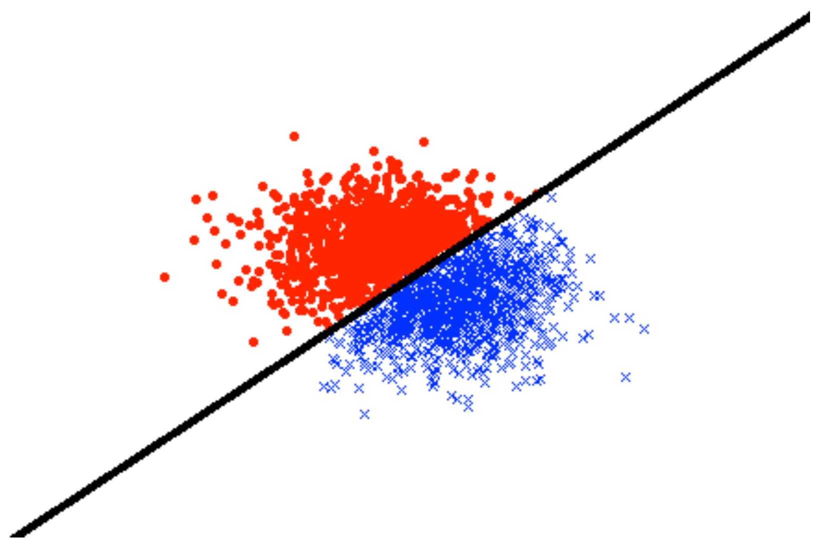
- The mistake bound gives us an upper bound on the perceptron **running time**
  - At least one mistake made per pass through the data
  - Running time is at most $N\left(\frac{R}{\gamma}\right)^2$ computed on **training** data

- Does not tell us anything about **generalization –** this is addressed by the concept of VC-dimension (in a couple lectures)

Q: What does the Perceptron mistake bound tell us?

Theorem: The maximum number of mistakes made by the perceptron algorithm is bounded above by R^2/γ^2





Demonstration in Matlab that Perceptron takes many more iterations to converge when there is a smaller margin (relative to R)

Online versus batch learning

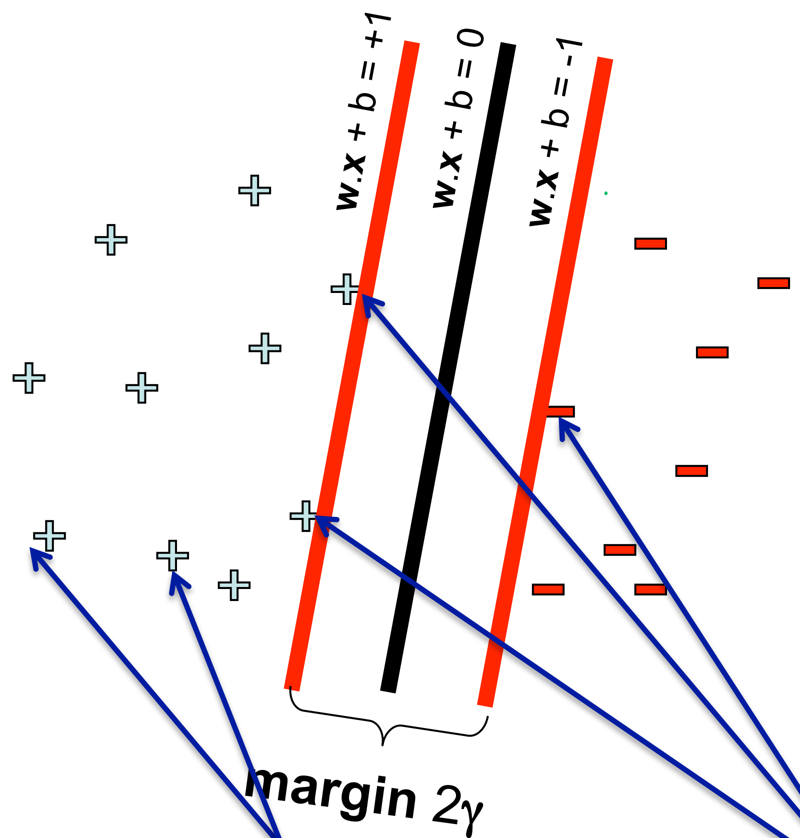
`Online Learning`

for $t = 1, 2, \ldots$
  receive question $\mathbf{x}_t \in \mathcal{X}$
  predict $p_t \in D$
  receive true answer $y_t \in \mathcal{Y}$
  suffer loss $l(p_t, y_t)$

- In the online setting we measure **regret**, i.e. the total cumulative loss
- No assumptions at all about the order of the data points!
- R and gamma refer to **all** data points (seen and future)
- Perceptron mistake bound tell us that the algorithm has bounded regret

[Shai Shalev-Shwartz, “Online Learning and Online Convex Optimization”, ‘11]

Recall from last lecture…
Support vector machines (SVMs)



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

- Example of a **convex optimization** problem
  - A quadratic program
  - Polynomial-time algorithms to solve!
- Hyperplane defined by **support vectors**
  - Could use them as a lower-dimension basis to write down line, although we haven't seen how yet

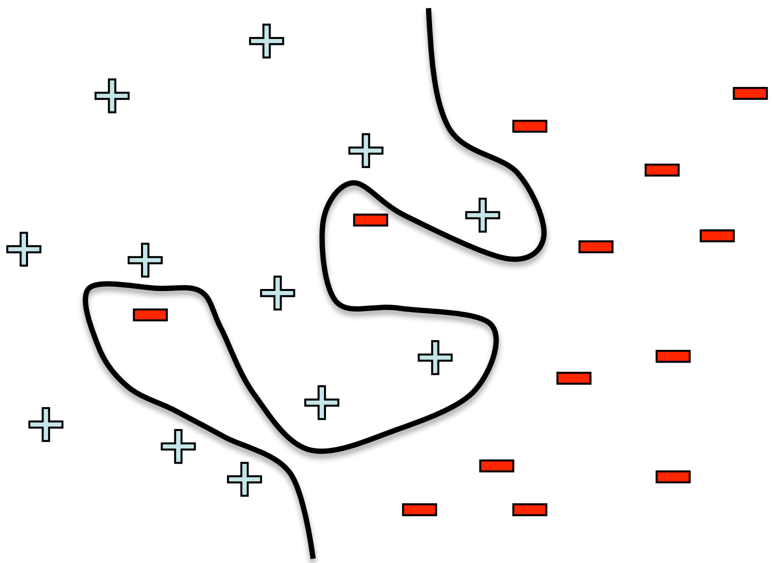
Non-support Vectors:
- everything else
- moving them will not change **w**

Support Vectors:
- data points on the canonical lines

What if the data is not linearly separable?

$$\left\langle x_i^{(1)}, \ldots, x_i^{(m)} \right\rangle \text{ — } m \text{ features}$$

$$y_i \in \{-1, +1\} \text{ — class}$$



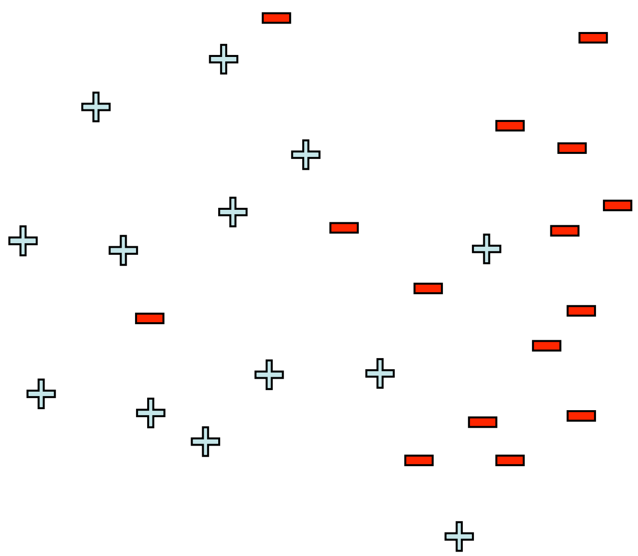
Add More Features!!!

$$\phi(x) = \begin{pmatrix} x^{(1)} \\ \ldots \\ x^{(n)} \\ x^{(1)}x^{(2)} \\ x^{(1)}x^{(3)} \\ \ldots \\ e^{x^{(1)}} \\ \ldots \end{pmatrix}$$

What about overfitting?

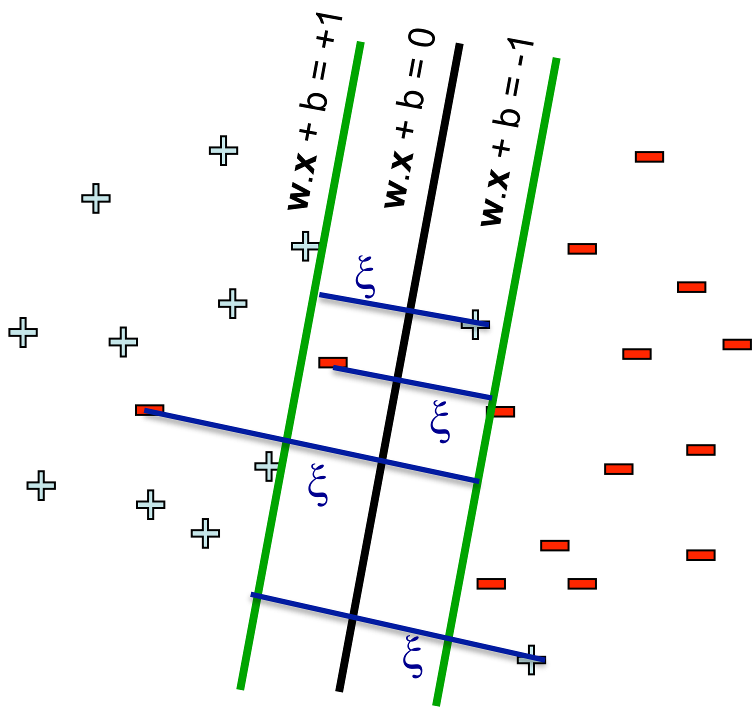
What if the data is not linearly separable?

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



- First Idea: Jointly minimize **w.w** and number of training mistakes
  - How to tradeoff two criteria?
  - Pick C using held-out data
- Tradeoff #(mistakes) and **w.w**
  - 0/1 loss
  - Not QP anymore
  - Also doesn't distinguish near misses and really bad mistakes
  - NP hard to find optimal solution!!!

Allowing for slack: "Soft margin" SVM



$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\,\xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad ,\forall j \quad \xi_j \geq 0$$

"slack variables"

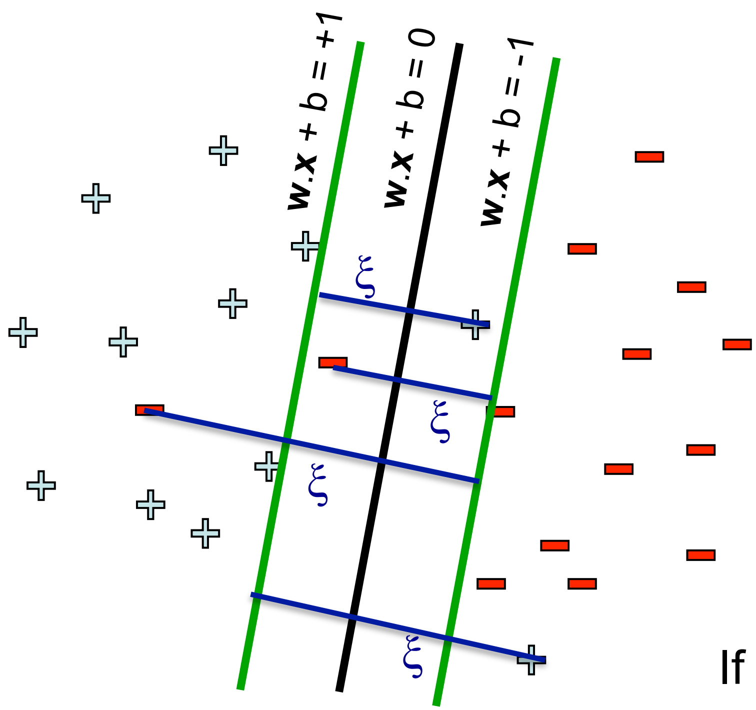
Slack penalty $C > 0$:

- $C=\infty$ → have to separate the data!
- $C=0$ → ignores the data entirely!

For each data point:

- If margin ≥ 1, don't care
- If margin < 1, pay linear penalty

Allowing for slack: "Soft margin" SVM



$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\,\xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad , \forall j \quad \xi_j \geq 0$$

"slack variables"

What is the (optimal) value of ξ_j as a function of **w** and b?

If $(w \cdot x_j + b)\, y_j \geq 1$, then $\xi_j = 0$

If $(w \cdot x_j + b)\, y_j < 1$, then $\xi_j = 1 - (w \cdot x_j + b)\, y_j$

$$\xi_j = \max\left(0, 1 - (w \cdot x_j + b)\, y_j\right)$$

Sometimes written as

$$\left(1 - (w \cdot x_j + b)\, y_j\right)_+$$

Equivalent hinge loss formulation

$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\,\Sigma_j\,\xi_j$$

$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1 - \xi_j \quad ,\forall j \quad \xi_j \geq 0$$

Substituting $\xi_j = \max\left(0, 1 - \left(w \cdot x_j + b\right) y_j\right)$ into the objective, we get:

$$\boxed{\min ||w||^2 + C \sum_j \max\left(0, 1 - \left(w \cdot x_j + b\right) y_j\right)}$$

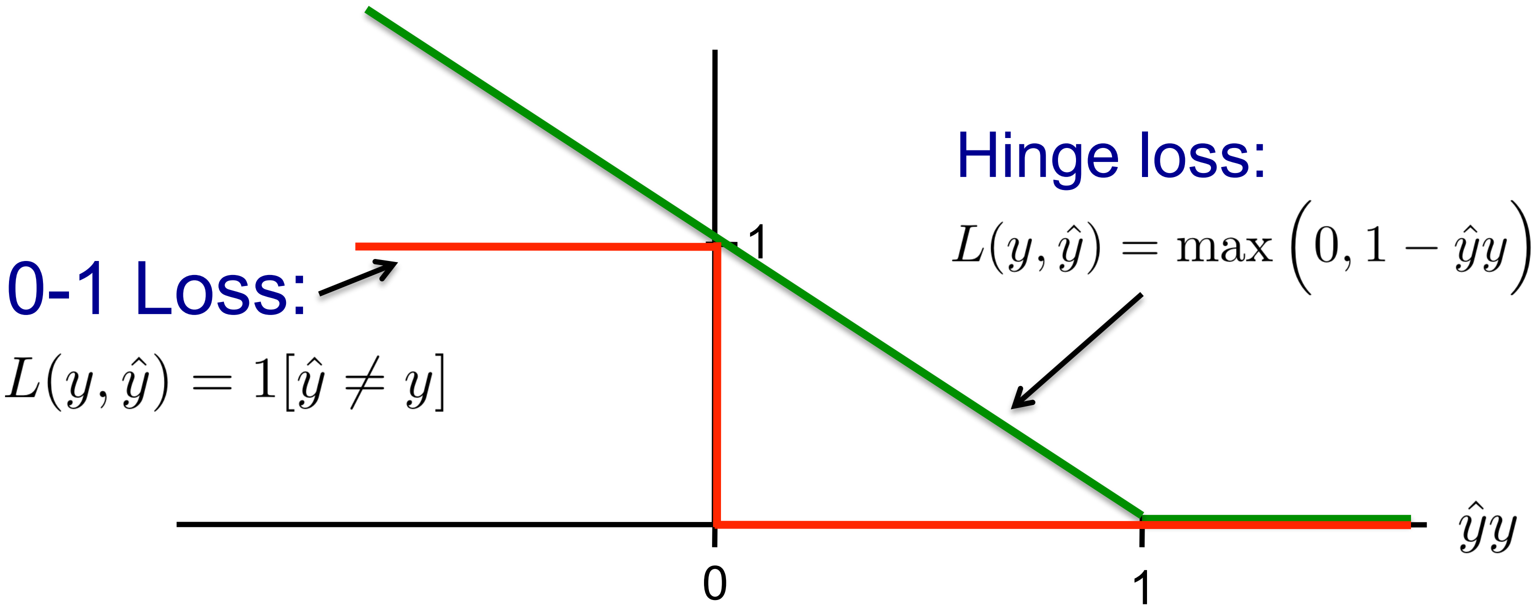
The **hinge loss** is defined as $L(y, \hat{y}) = \max\left(0, 1 - \hat{y}y\right)$

$$\min_{w,b} ||w||_2^2 + C \sum_j L(y_j, \mathbf{w} \cdot x_j + b)$$

This is called **regularization**; used to prevent overfitting!

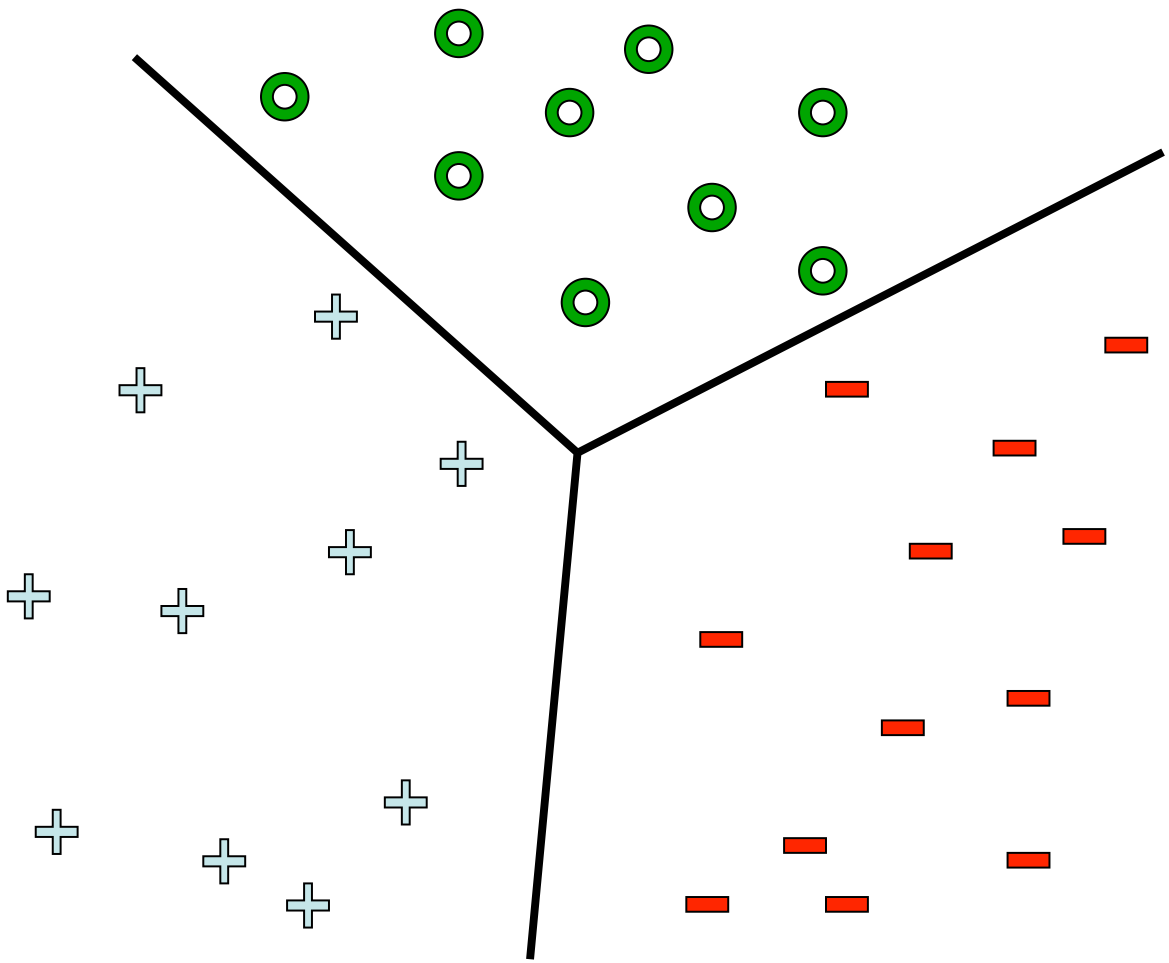
This part is empirical risk minimization, using the hinge loss

Hinge loss vs. 0/1 loss

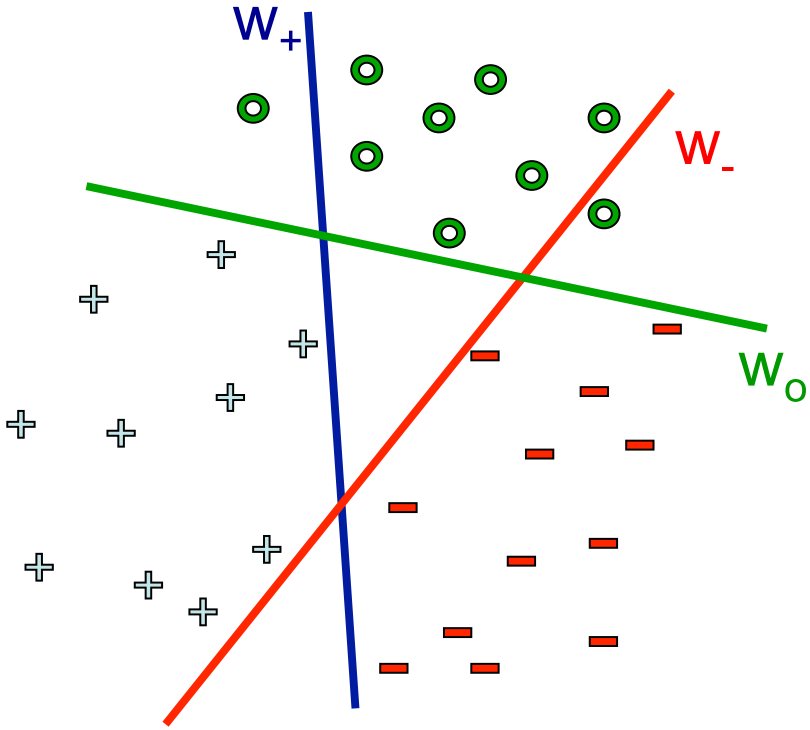


Hinge loss upper bounds 0/1 loss!

How do we do multi-class classification?



One versus all classification



Learn 3 classifiers:

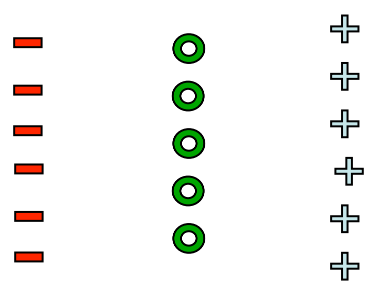
- \- vs {o,+}, weights w_-
- \+ vs {o,-}, weights w_+
- o vs {+,-}, weights w_o

Predict label using:

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

Any problems?

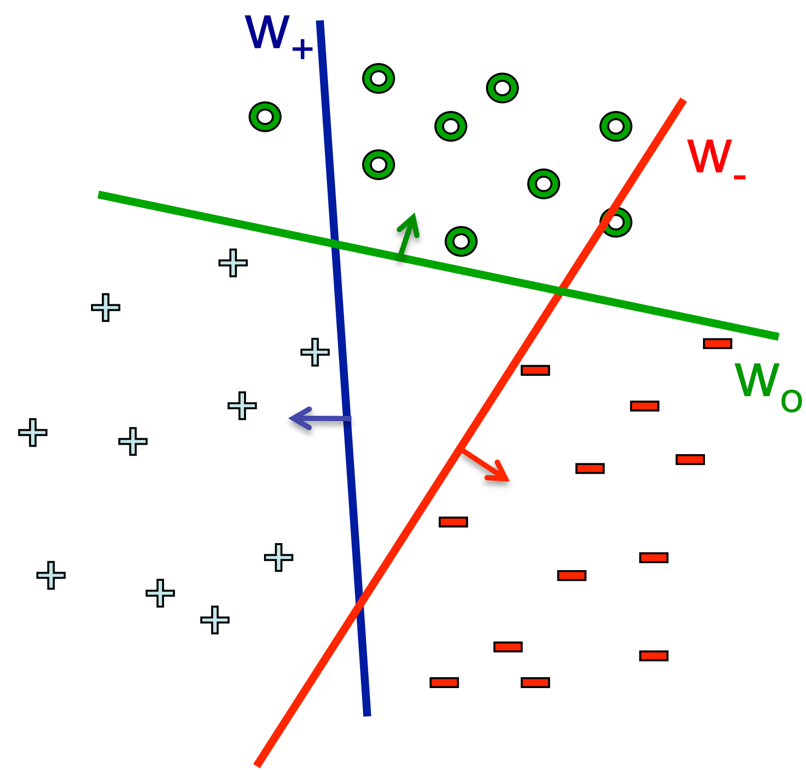
Could we learn this dataset? →



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

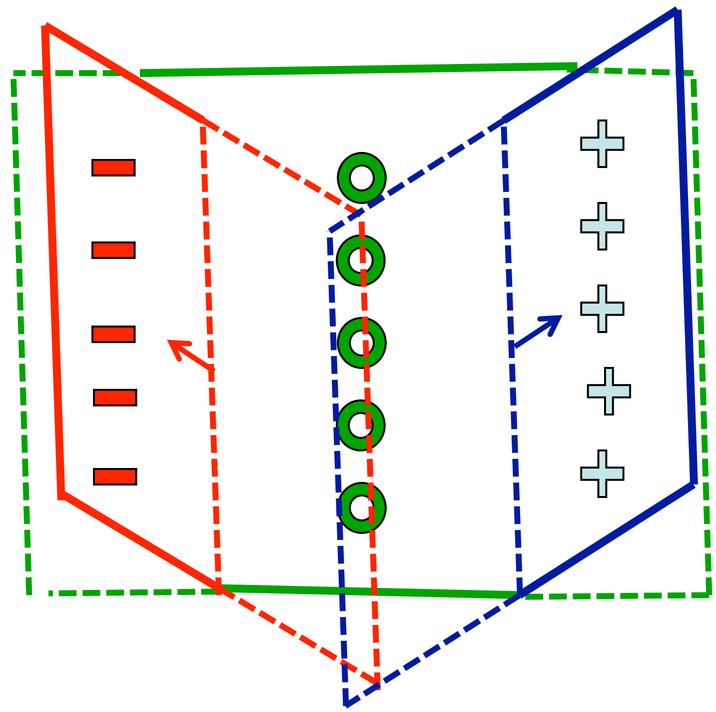
As for the SVM, we introduce slack variables and maximize margin:

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

To predict, we use:

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

Now can we learn it? →



What you need to know

- Perceptron mistake bound
- Maximizing margin
- Derivation of SVM formulation
- Relationship between SVMs and empirical risk minimization
  - 0/1 loss versus hinge loss
- Tackling multiple class
  - One against All
  - Multiclass SVMs

What's Next!

- Learn one of the most interesting and exciting recent advancements in machine learning
  - The "kernel trick"
  - High dimensional feature spaces at no extra cost!
- But first, a detour
  - Constrained optimization!