Data Management for Data Science

Lecture 17: Linear Classifiers and Support Vector Machines

Prof. Asoc. Endri Raço

Today’s Lecture

1. Linear classifiers

2. The Perceptron Algorithm

3. Support Vector Machines

Linear classifier

- Let's simplify life by assuming:
  - Every instance is a vector of real numbers, **x**=(x_1,…,x_n). (Notation: boldface **x** is a vector.)
  - There are only two classes, *y=(+1)* and *y=(-1)*
- A linear classifier is vector **w** of the same dimension as **x** that is used to make this prediction:

$$\hat{y} = \mathrm{sign}(w_1 x_1 + w_2 x_2 + \ldots + w_n x_n) = \mathrm{sign}(\mathbf{w} \cdot \mathbf{x})$$

$$\mathrm{sign}(x) = \begin{cases} +1 & \text{if } x \geq 0 \\ -1 & \text{if } < 0 \end{cases}$$

Example: Linear classifier

- Imagine 3 features (spam is "positive" class):
  1. free (number of occurrences of "free")
  2. money (occurrences of "money")
  3. BIAS (intercept, always has value 1)

$$w \cdot f(x)$$

$$\sum_i w_i \cdot f_i(x)$$

x	$f(x)$	w
"free money"	BIAS : 1 free : 1 money : 1 ...	BIAS : -3 free : 4 money : 2 ...

$$(1)(-3) +$$
$$(1)(4) +$$
$$(1)(2) +$$
$$\ldots$$
$$= 3$$

w.f(x) > 0 ➔ SPAM!!!

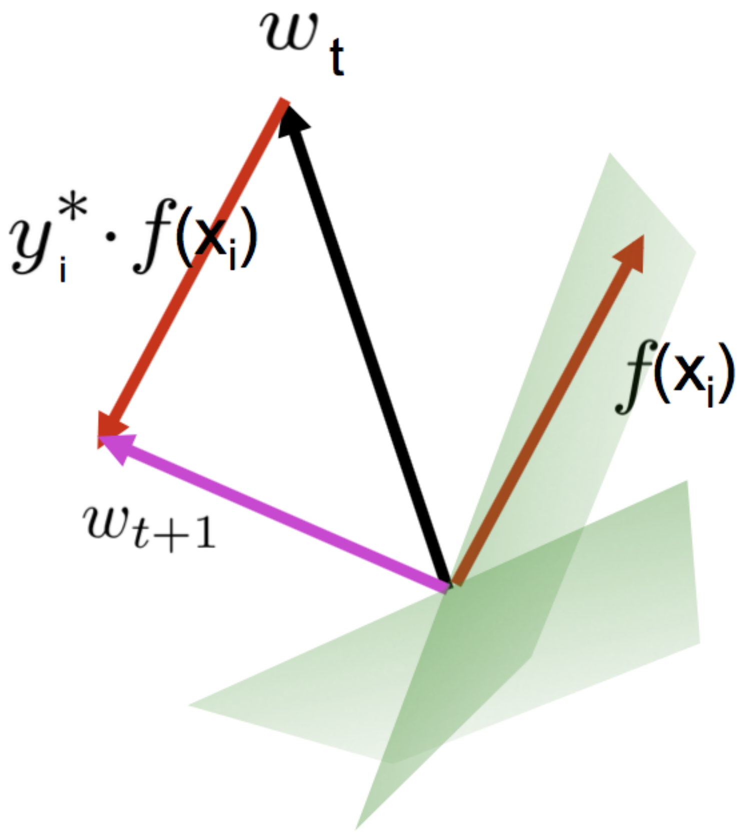
The Perceptron Algorithm to learn a Linear Classifier

- Start with weight vector = $\vec{0}$
- For each training instance (x_i, y_i^*):
  - Classify with current weights

$$y_i = \begin{cases} +1 & \text{if } \ w \cdot f(x_i) \geq 0 \\ -1 & \text{if } \ w \cdot f(x_i) < 0 \end{cases}$$

  - If correct (i.e., $y = y_i^*$), no change!
  - If wrong: update

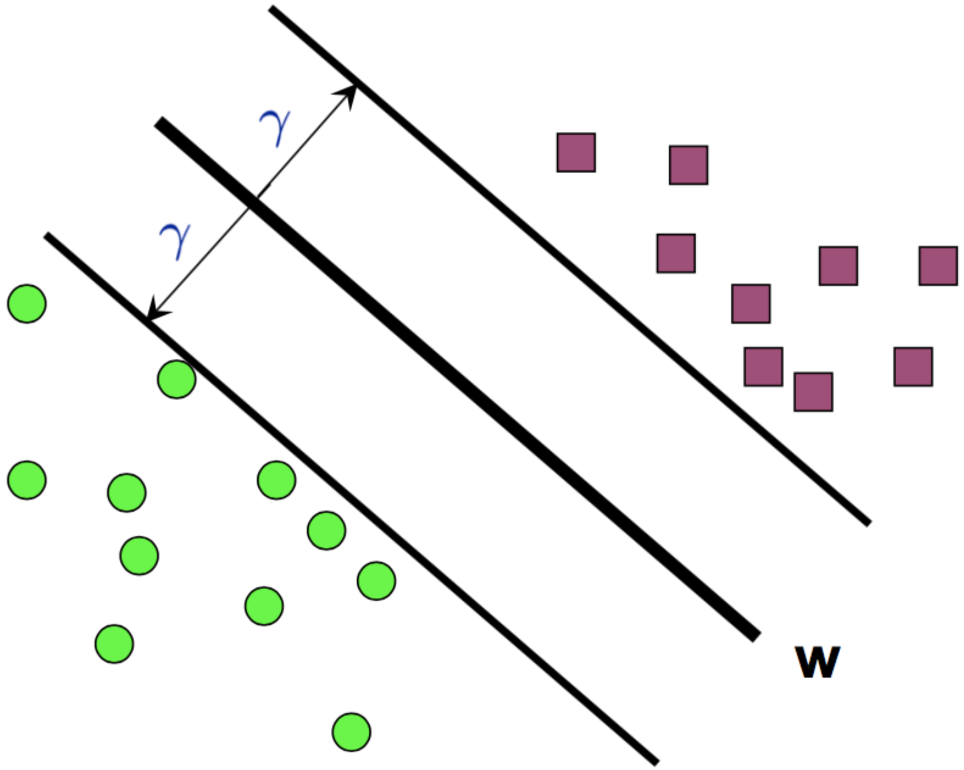
$$w = w + y_i^* f(x_i)$$



Definition: Linearly separable data

$\exists \mathbf{w}$ such that $\forall t$ $\quad y_t(\mathbf{w} \cdot \mathbf{x}_t) \geq \gamma > 0$

Called the *margin*



Equivalently, for $y_t = +1$,

$$w \cdot x_t \geq \gamma$$

and for $y_t = -1$,

$$w \cdot x_t \leq -\gamma$$

Does the perceptron algorithm work?

- Assume the data set *D* is linearly separable with margin γ, i.e.,

$$\exists \mathbf{w}^*, |\mathbf{w}^*|_2 = 1,\ \forall t, y_t \mathbf{x}_t^\top \mathbf{w}^* \geq \gamma$$

- Assume $|\mathbf{x}_t|_2 \leq R, \forall t$
- Theorem: The maximum number of mistakes made by the perceptron algorithm is bounded by R^2/γ^2

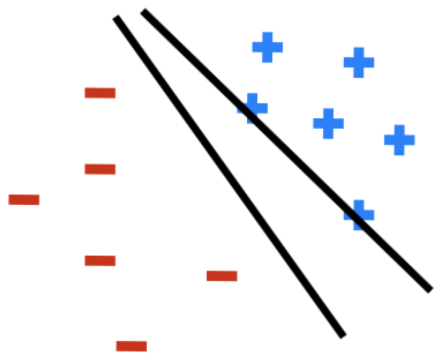
[Rong Jin]

Properties of the perceptron algorithm

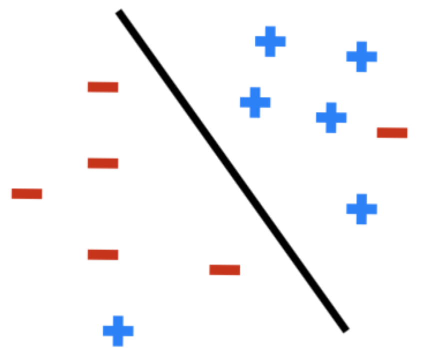
- Separability: some parameters get the training set perfectly correct

- Convergence: if the training is **linearly separable**, perceptron will eventually converge

Separable

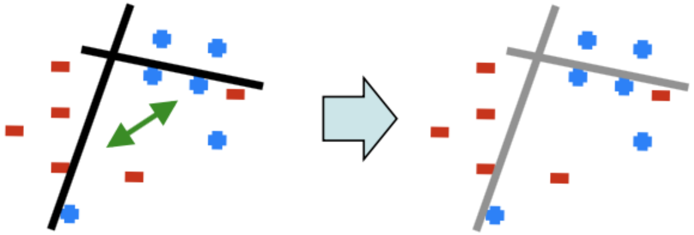


Non-Separable

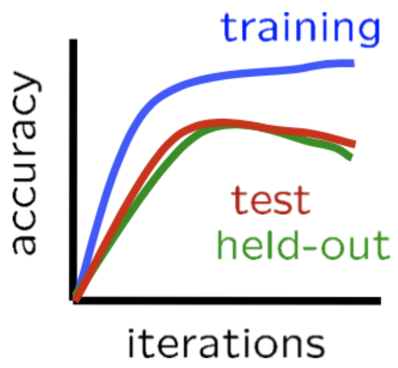


Problems with the perceptron algorithm

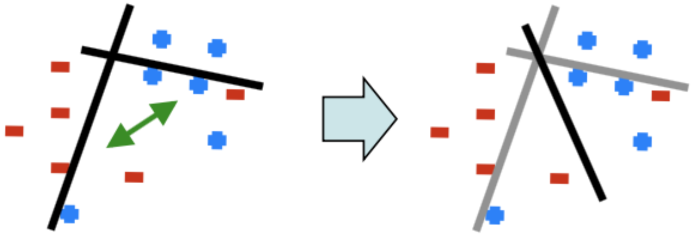
- **Noise**: if the data isn't linearly separable, no guarantees of convergence or accuracy



- Frequently the training data ***is*** linearly separable! Why?
  - When the number of features is much larger than the number of data points, there is lots of flexibility
  - As a result, Perceptron can significantly **overfit** the data **[We will see next week]**

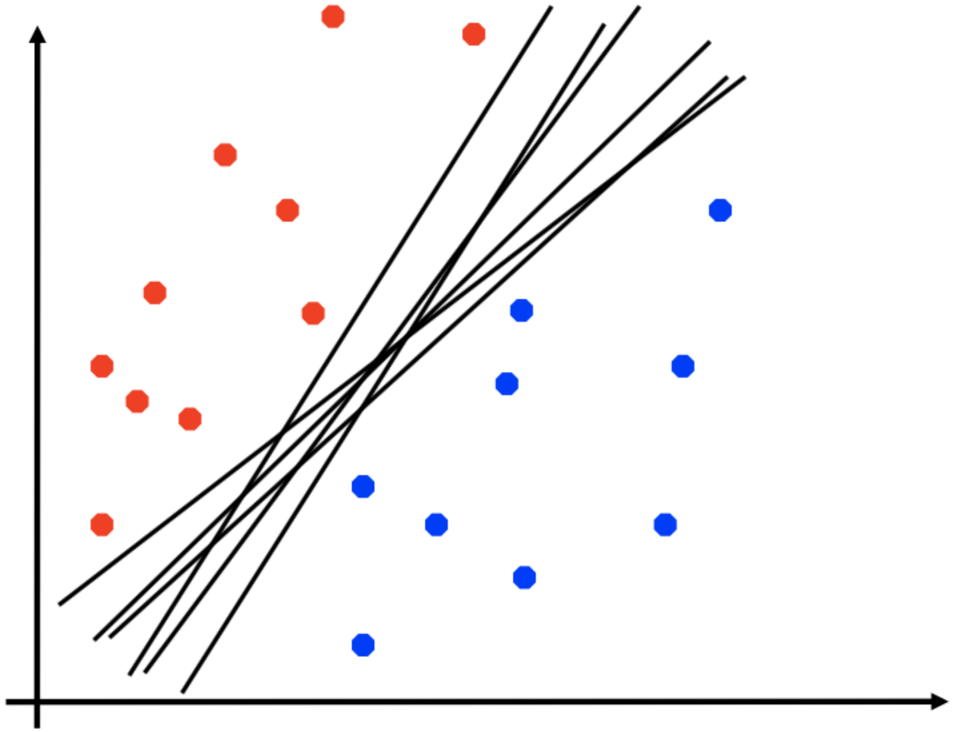


- **Averaged** perceptron is an algorithmic modification that helps with both issues
  - Averages the weight vectors across all iterations



Linear separators

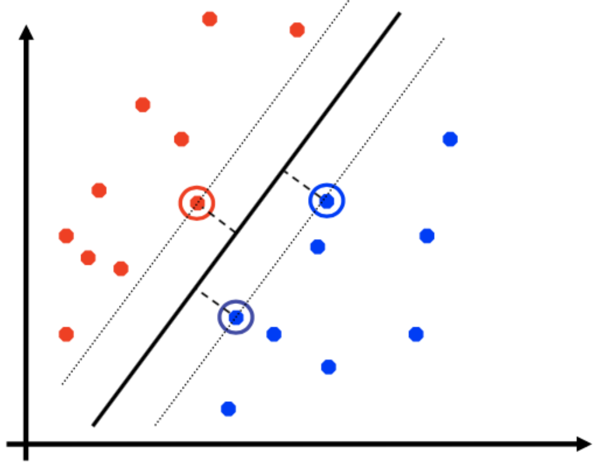
- Which of these linear separators is optimal?



Support Vector Machines

- SVMs (Vapnik, 1990's) choose the linear separator with the **largest margin**

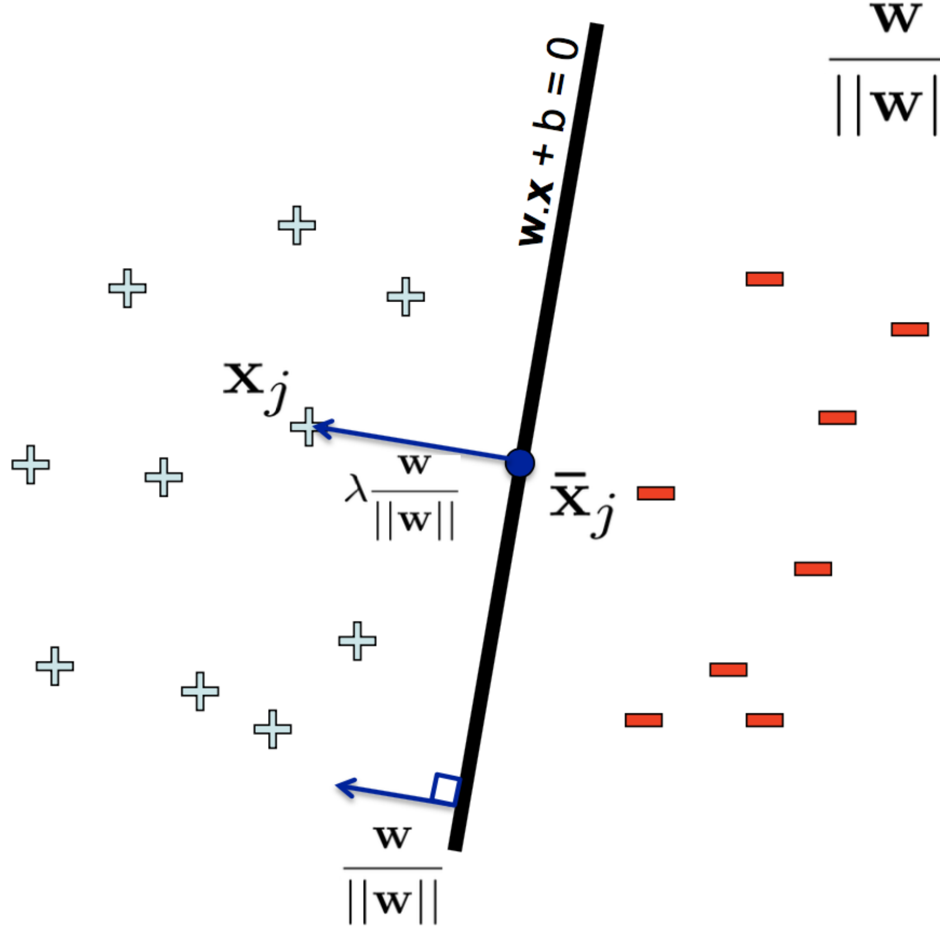
Robust to outliers!



V. Vapnik

- Good according to intuition, theory, practice
- SVM became famous when, using images as input, it gave accuracy comparable to neural-network with hand-designed features in a handwriting recognition task

Normal to a plane



$\frac{\mathbf{w}}{||\mathbf{w}||}$ -- unit vector parallel to w

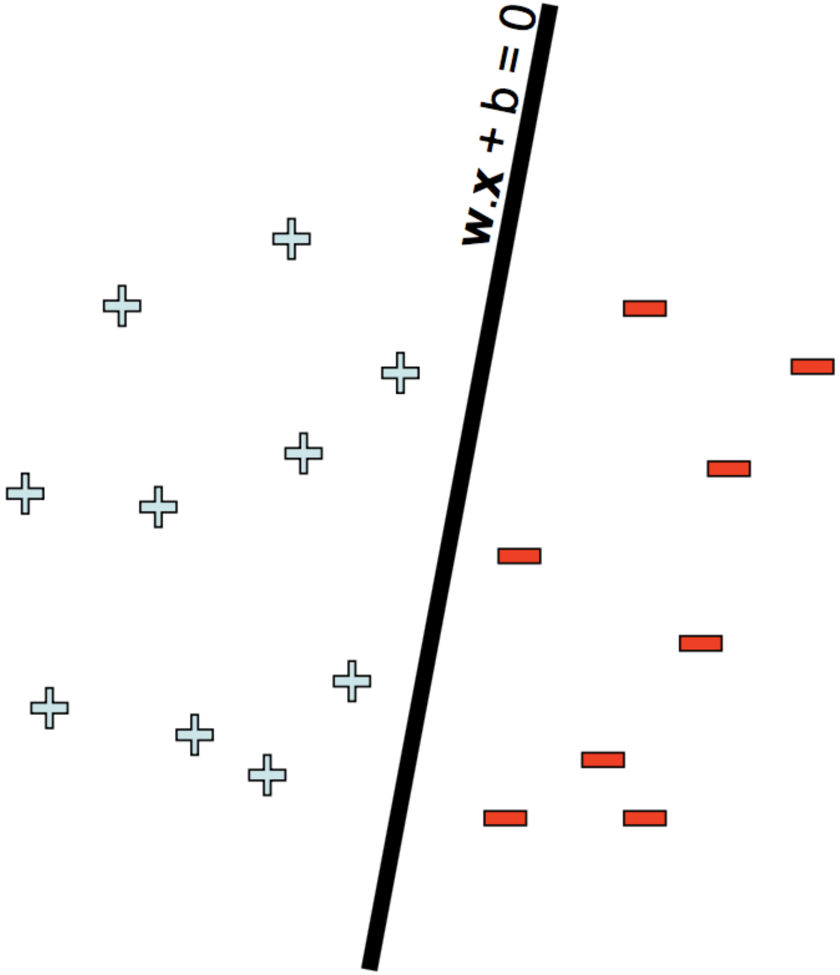
$\bar{\mathbf{x}}_j$ -- projection of x_j onto the plane

$$x_j - \bar{x}_j = \lambda \frac{w}{||w||}$$

λ is the length of the vector, i.e.

$$||x_j - \bar{x}_j|| = \frac{\lambda}{||w||} ||w|| = \lambda$$

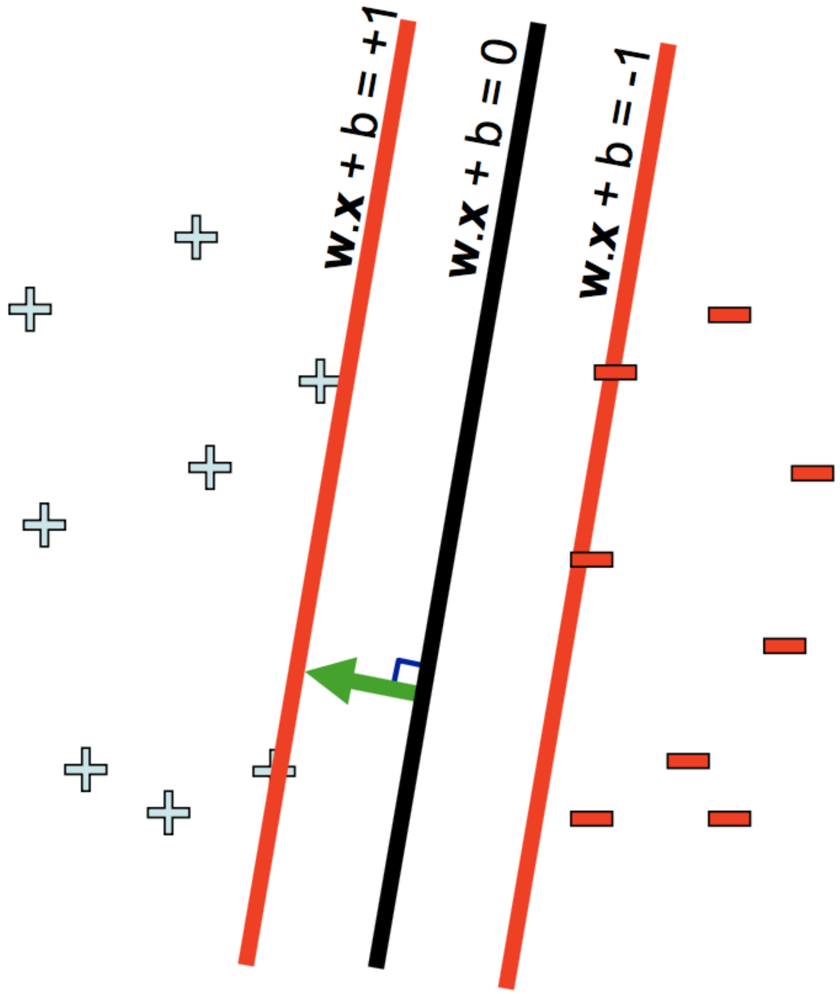
Scale invariance



Any other ways of writing the same dividing line?

- **w.x** + b = 0
- 2**w.x** + 2b = 0
- 1000**w.x** + 1000b = 0
- ….

Scale invariance



During learning, we set the scale by asking that, for all *t*,

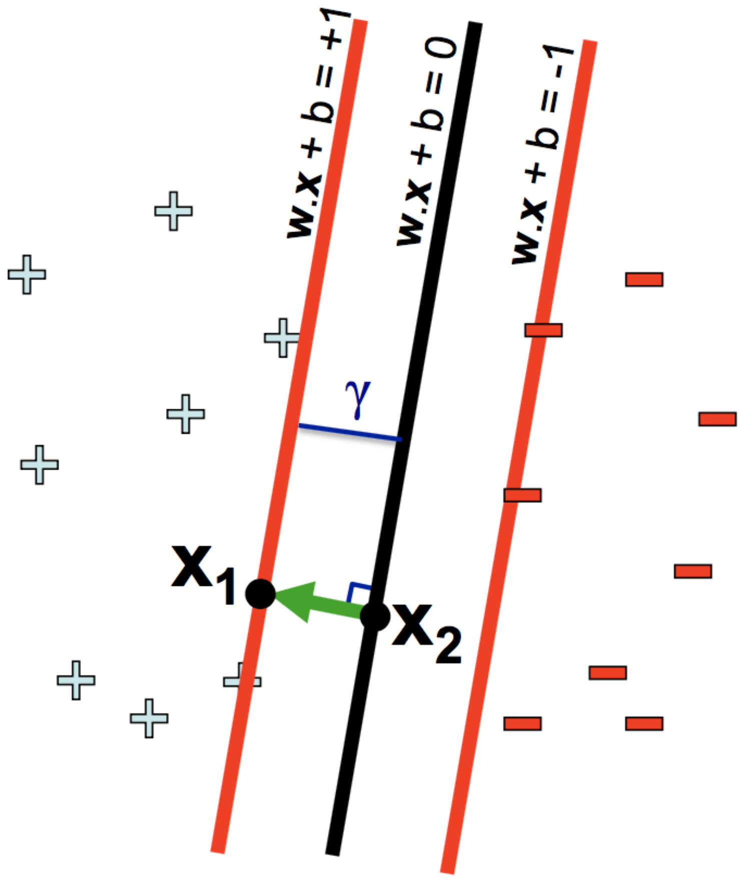
for $y_t = +1$, $w \cdot x_t + b \geq 1$

and for $y_t = -1$, $w \cdot x_t + b \leq -1$

That is, we want to satisfy all of the **linear** constraints

$$y_t \, (w \cdot x_t + b) \geq 1 \quad \forall t$$

What is γ as a function of **w**?



$$w \cdot x_1 + b = 1$$
$$-$$
$$w \cdot x_2 + b = 0$$

$$w \cdot (x_1 - x_2) = 1$$

Plug in

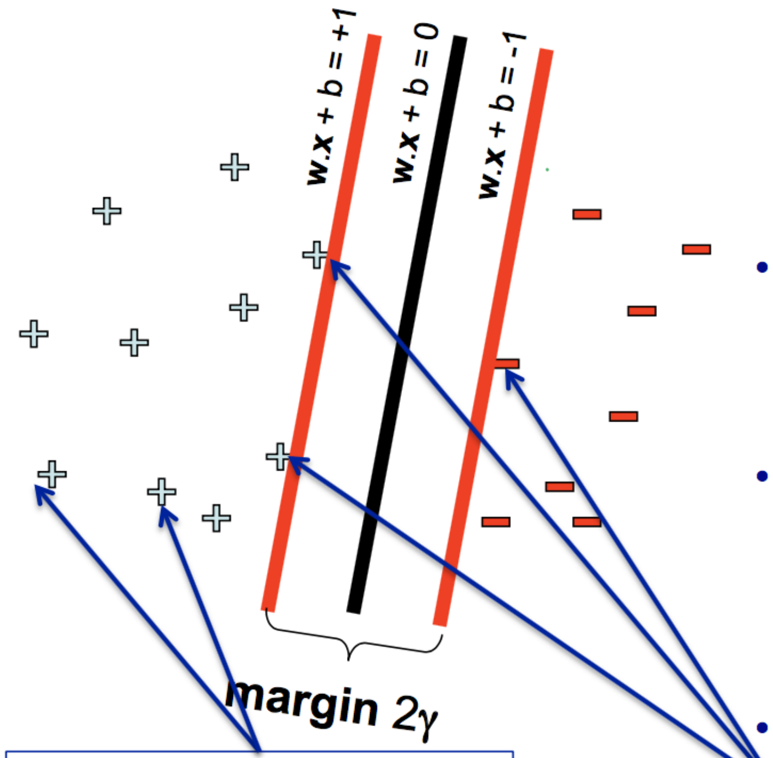
We also know that:
$$x_1 - x_2 = \gamma \frac{w}{||w||}$$

$$1 = w \cdot \left(\gamma \frac{w}{||w||}\right) = \frac{\gamma}{||w||} w \cdot w = \gamma ||w||$$

So, $\boxed{\gamma = \frac{1}{||w||}}$

Final result: can maximize margin by minimizing $||w||_2$!!!

Support Vector Machines (SVMs)



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

Support Vectors:
- data points on the canonical lines

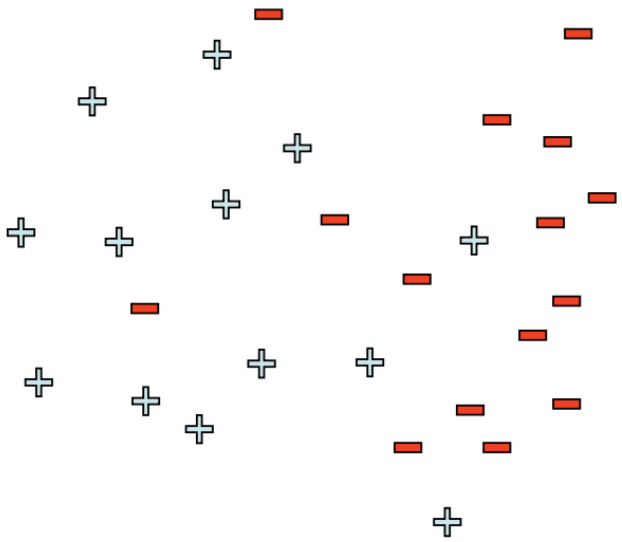
$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w}$$
$$\left(\mathbf{w}.\mathbf{x}_j + b\right) y_j \geq 1, \ \ \forall j$$

- 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
- More on these later

What if the data is not separable?

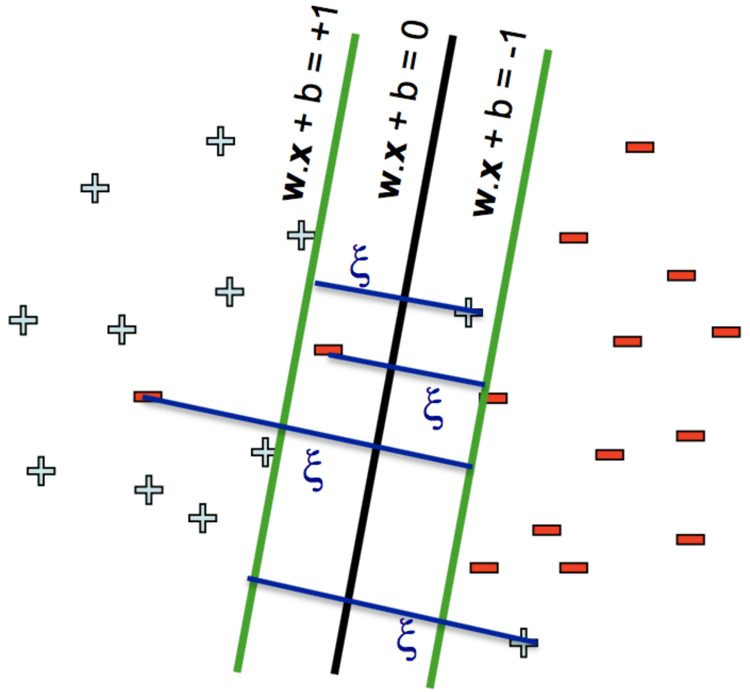
$$\text{minimize}_{\mathbf{w},b} \quad \mathbf{w}.\mathbf{w} + C\ \#(\text{mistakes})$$

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



- 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"

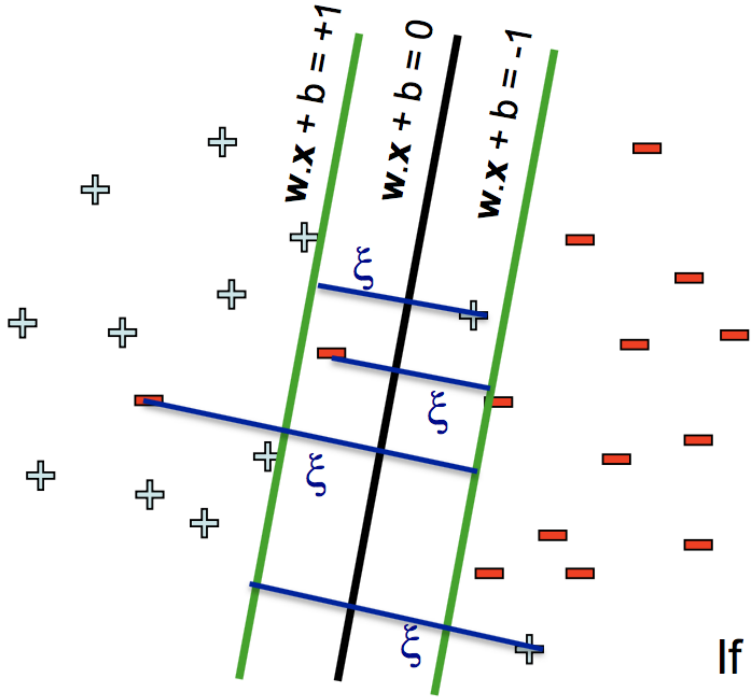
Slack penalty $C > 0$:

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

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

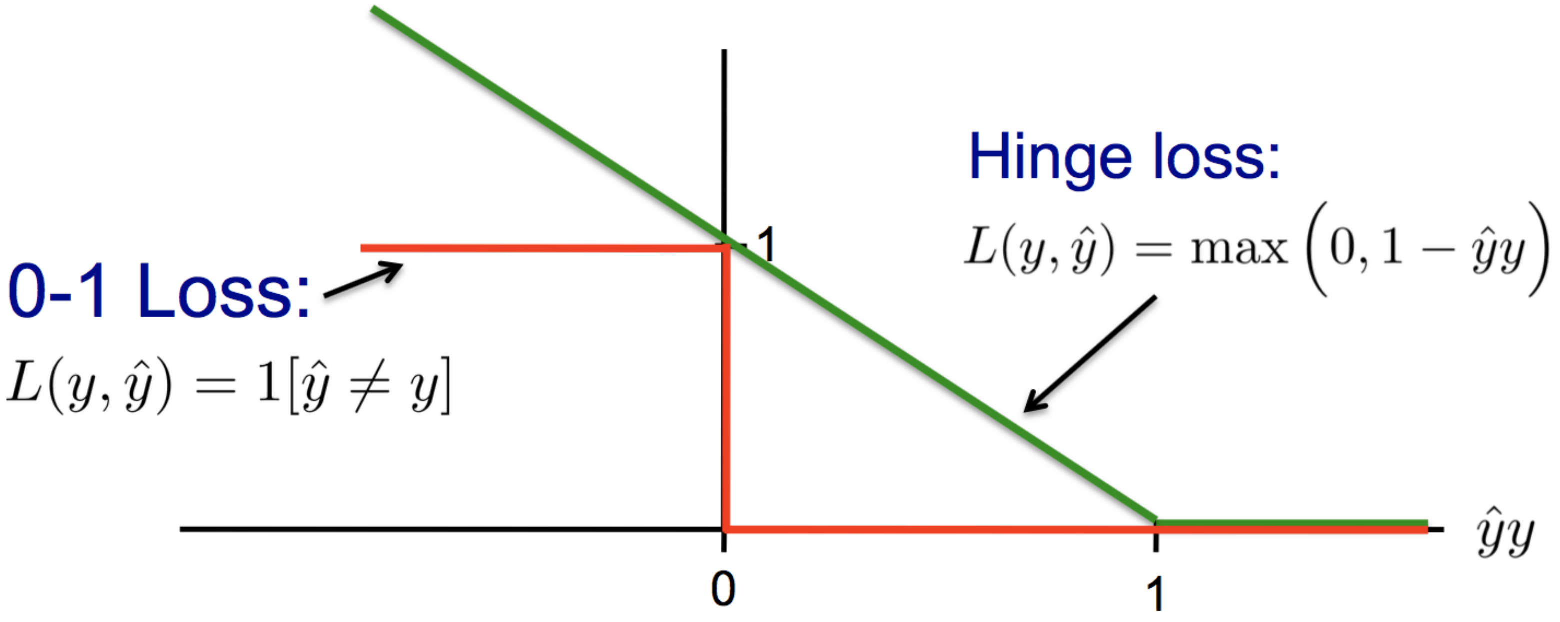
The **hinge loss** is defined as $L(y, \hat{y}) = \max\left(0, 1 - \hat{y}y\right)$

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

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

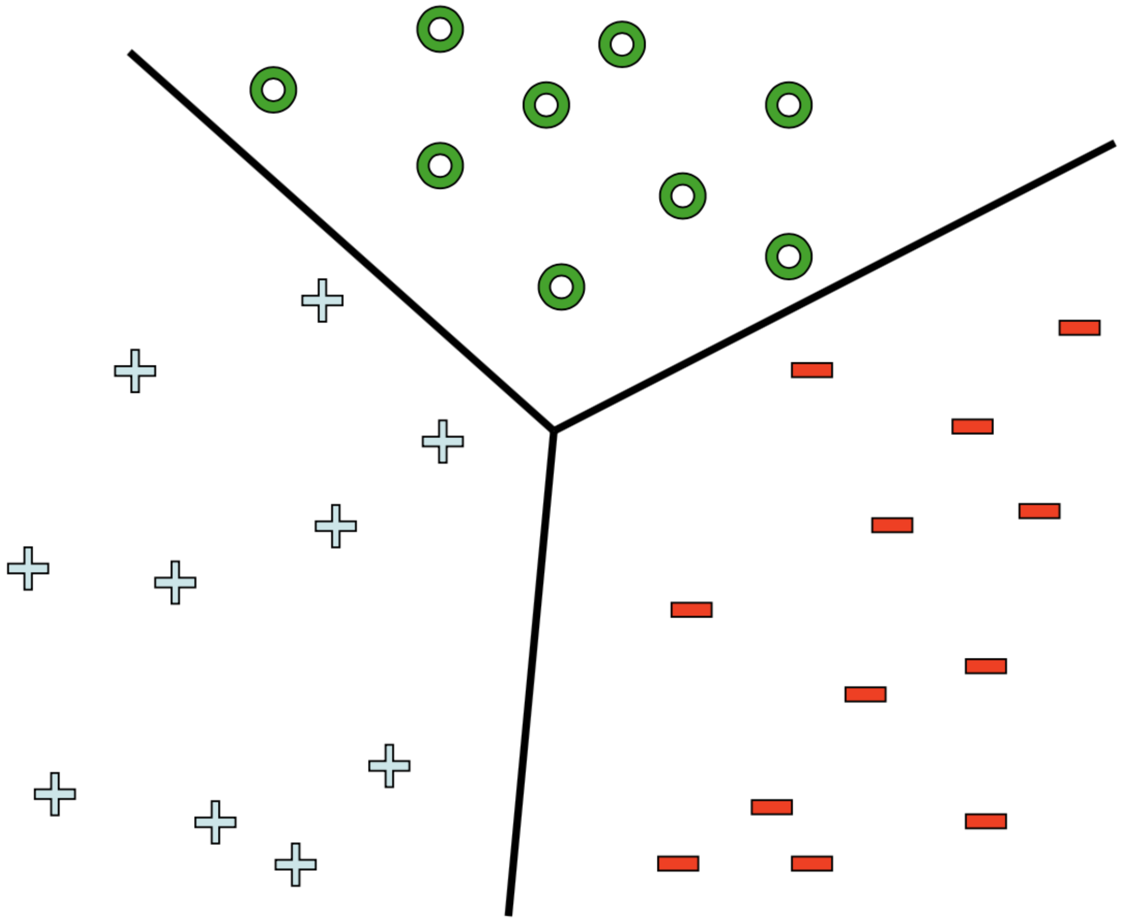
This part is empirical risk minimization, using the hinge loss

Hinge Loss vs 0-1 Loss

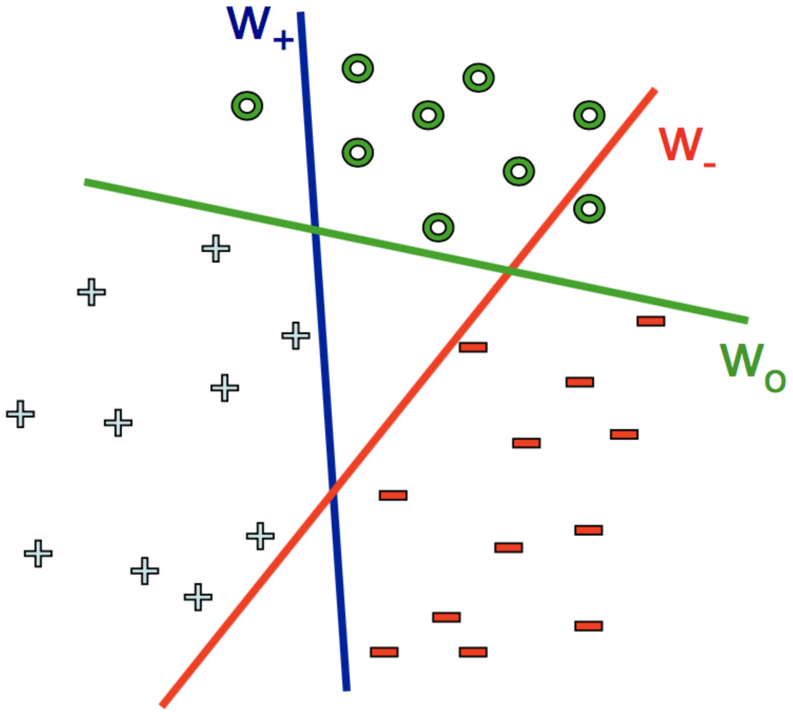


Hinge loss upper bounds 0/1 loss!

Multiclass SVM



One versus all classification



Learn 3 classifiers:

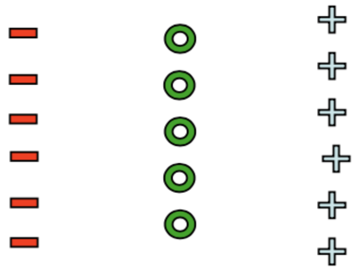
- \- vs {o,+}, weights w_-
- \+ vs {o,-}, weights w_+
- o vs {+,-}, weights w_o

Predict label using:

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

Any problems?

Could we learn this dataset? →



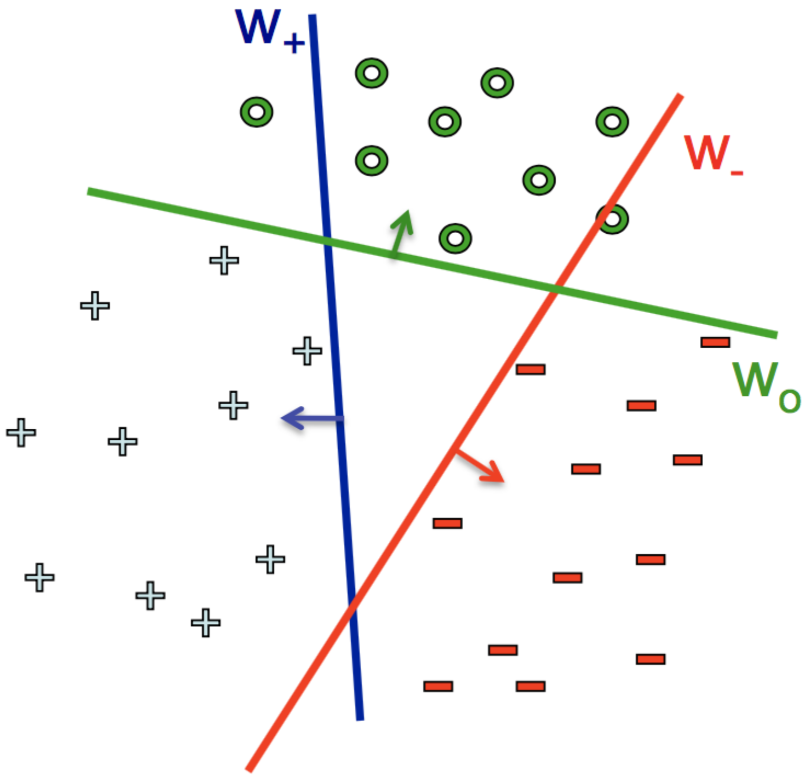
Multiclass SVM

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:



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

Multiclass SVM

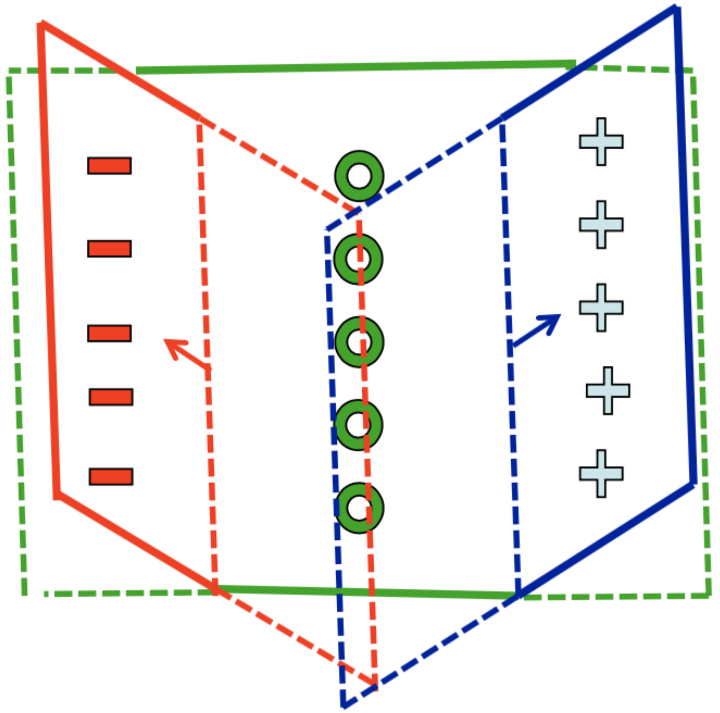
As for the SVM, we introduce slack variables and maximize margin:

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

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