Support Vector Machines

Adopted from slides by

Jure Leskovec, Anand Rajaraman, Jeff Ullman, http://www.mmds.org

and Alexander Ihler

Linear classifiers

- Classifier model: $f(x;\theta) = T(\theta x)$
- Loss function:

Class y = {0, 1}

$$J(\theta) = \frac{1}{m}\sum_i \left(y^{(i)}\phi(\theta x^{(i)}) + (1 - y^{(i)})\phi(-\theta x^{(i)})\right)$$

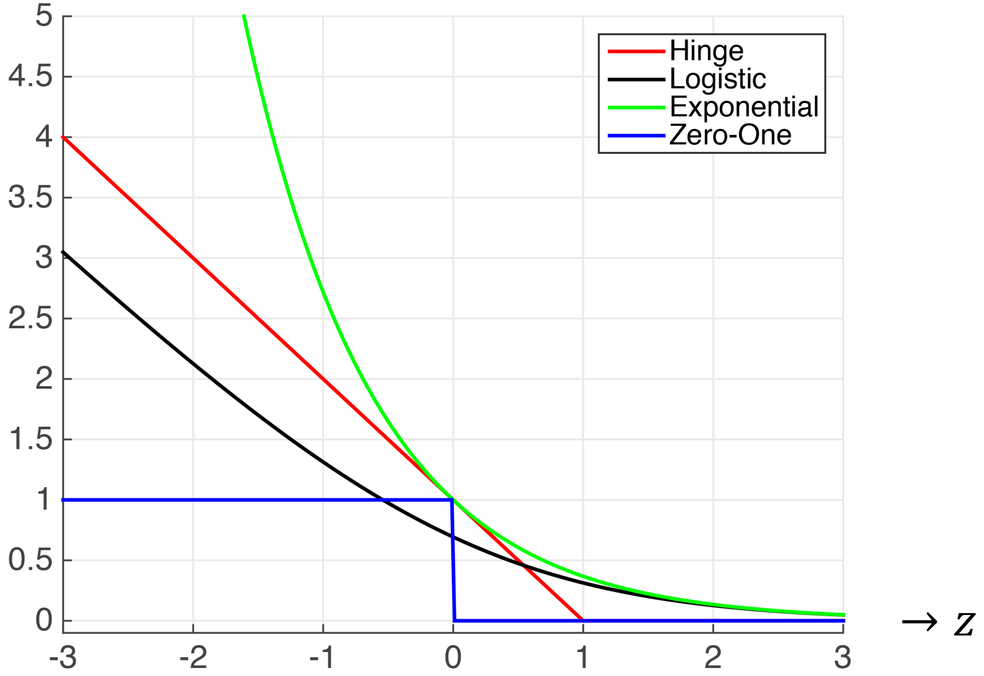
Class y = {-1, 1}

$$J(\theta) = \frac{1}{m}\sum_i \phi\left(y^{(i)}(\theta x^{(i)})\right)$$

Surrogate loss functions

- 0-1: $\mathcal{L}(z) = \mathbf{1}[z < 0]$
- Logistic: $\mathcal{L}(z) = -\log \sigma(z)$
- Exponential: $\mathcal{L}(z) = e^{-\beta z}$
- Hinge: $\mathcal{L}(z) = \max\{0, 1 - z\}$
- …

Logistic regression

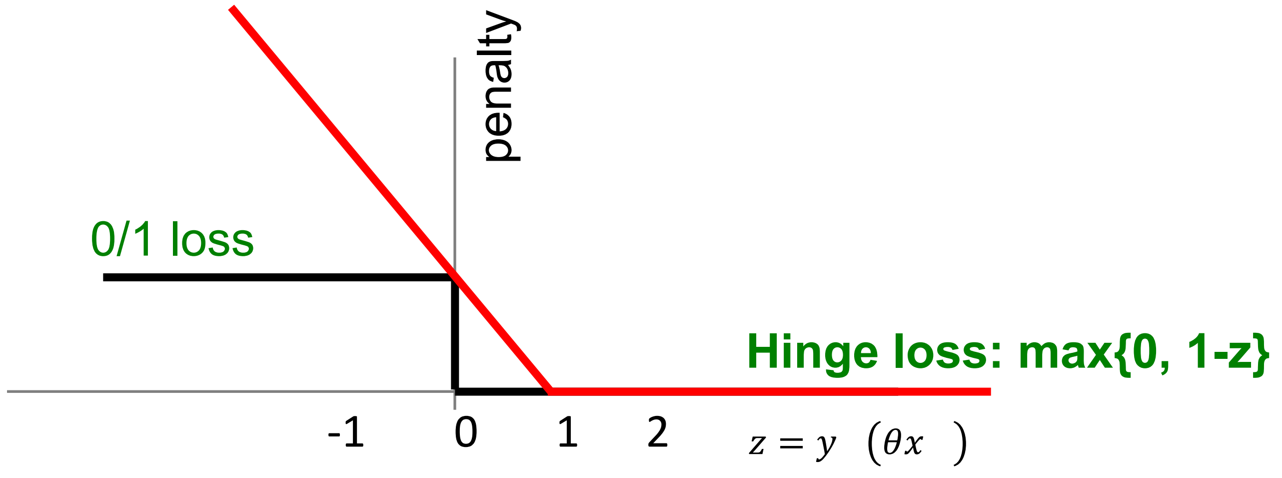


Support Vector Machines (SVM)

- Classifier: $f(x;\theta) = sign(\theta x)$
- Loss function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \max\{0, 1 - y^{(i)}(\theta x^{(i)})\} + \frac{\lambda}{2m}||\theta||$$

L_2 regularization

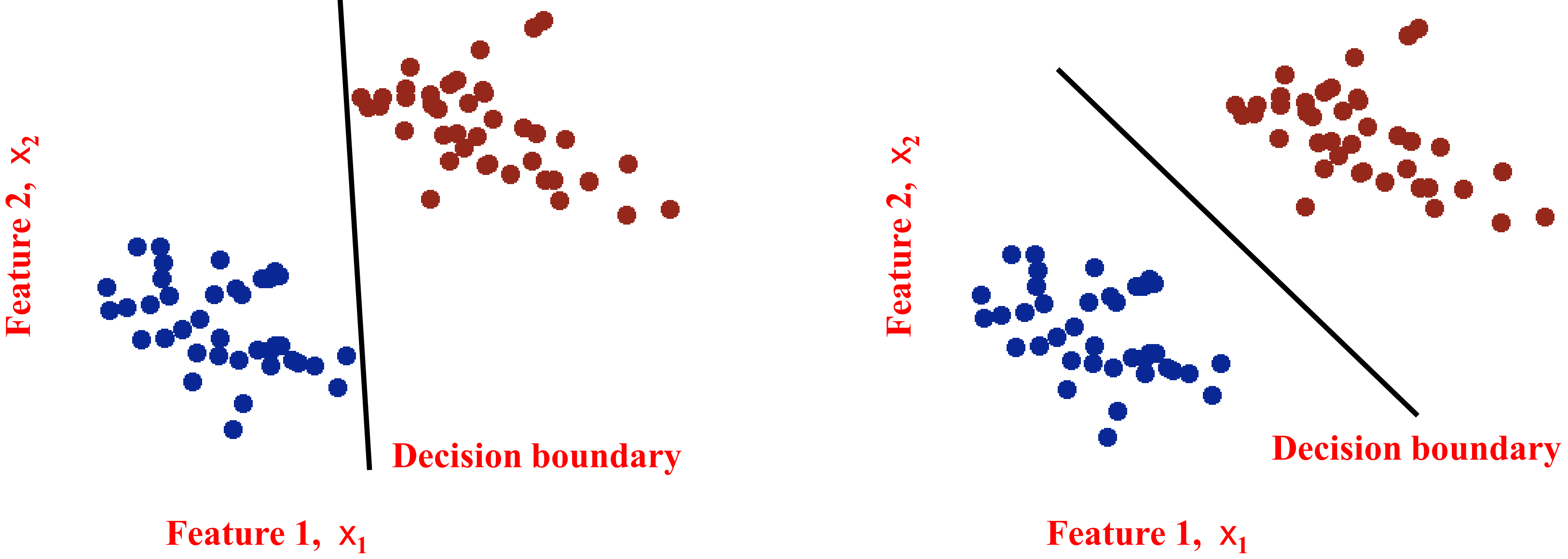


Linear classifiers

- Which decision boundary is "better"?
  - Both have zero training error (perfect training accuracy)
  - But, one of them seems intuitively better…

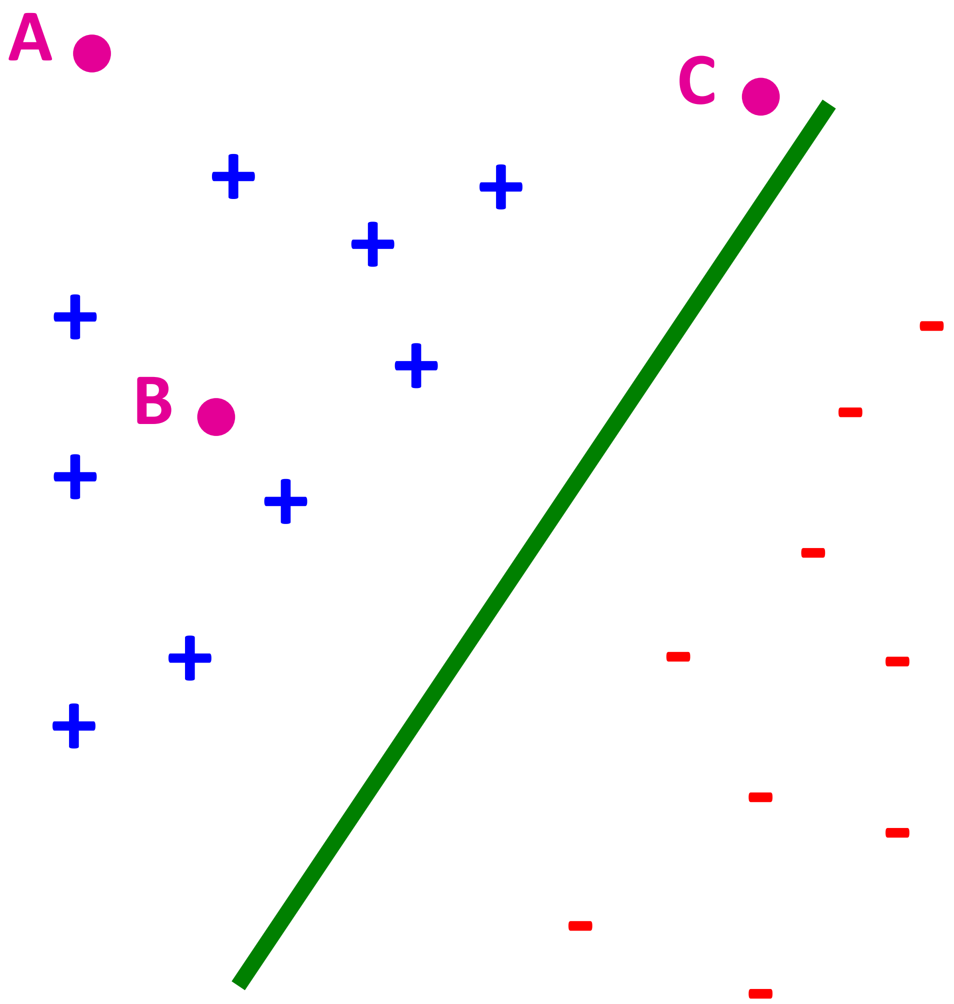
- How can we quantify "better",

  and learn the "best" parameter settings?



Largest Margin

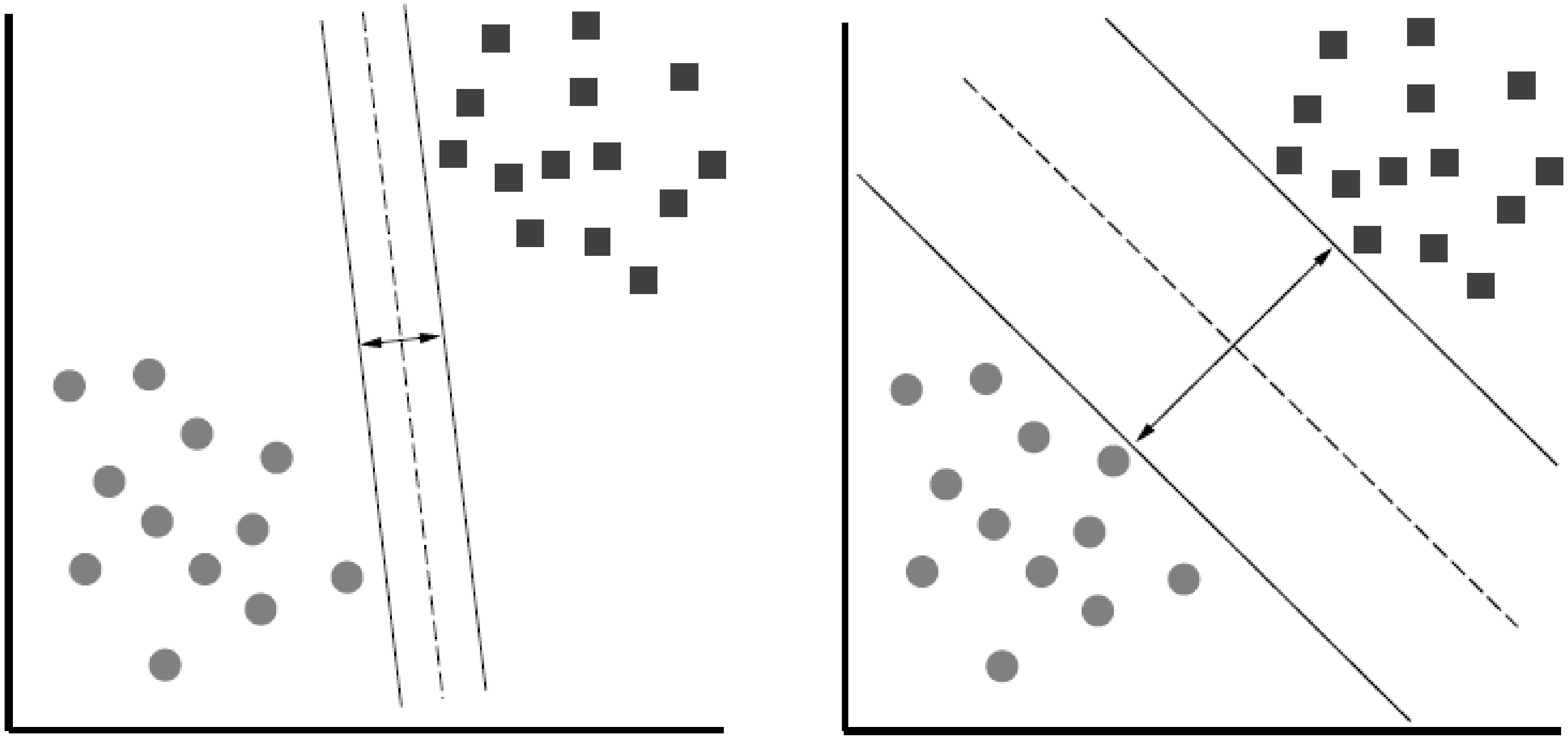
Margin:
(2 ×) the distance from the decision boundary to the closest example.



- **Distance from the separating hyperplane corresponds to the "confidence" of prediction**
- **Example:**
  - We are more sure about the class of **A** and **B** than of **C**

Largest Margin

- **Margin γ: Distance of closest example from the decision line/hyperplane**



The reason we define margin this way is due to theoretical convenience and existence of generalization error bounds that depend on the value of margin.

Notations

- Parameters:

$$\theta_0, \theta_1, \cdots, \theta_n \quad \Longrightarrow \quad b, \overbrace{w_1, w_2, \cdots, w_n}^{w}$$

- Data:
  - Training examples:

$$\left(x^{(1)}, y^{(1)}\right), \cdots, \left(x^{(m)}, y^{(m)}\right)$$

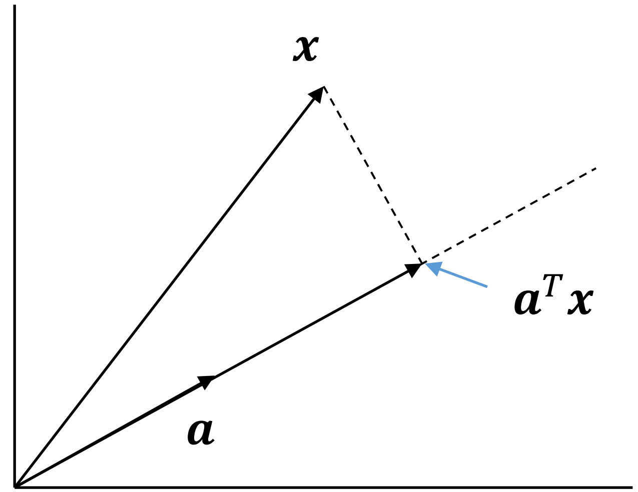
  - Each example i:

$$x^{(i)} = \left(x_1^{(i)}, \cdots, x_n^{(i)}\right)$$
$$y^{(i)} \in \{-1, +1\}$$

- Decision boundary:

$$\theta x = 0 \quad \Longrightarrow \quad w \cdot x + b = 0$$

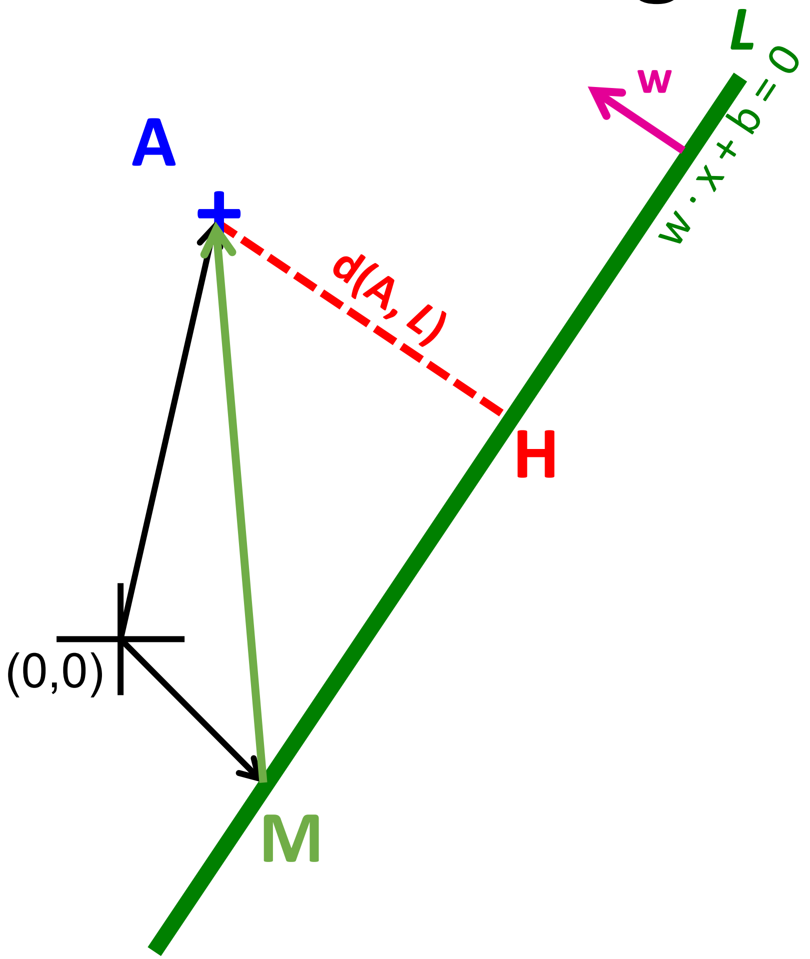
Projection: Using Inner Products



Projection of $\boldsymbol{x}$ along the direction $\boldsymbol{a}$ ($\|\boldsymbol{a}\| = 1$)

$$\mathbf{p} = \mathbf{a}\,(a^T x)$$
$$\|a\| = a^T a = 1$$

What is the margin?



Distance from a point to a line

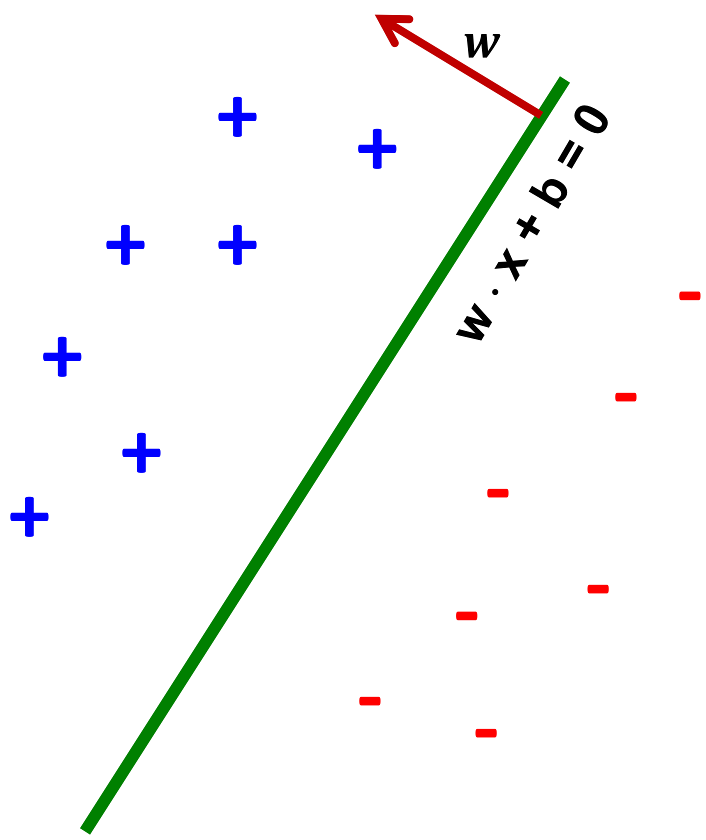
Note we assume $\|w\|_2 = 1$

- **Let:**
  - **Line L:** **w·x+**b = 0
  - **Point A**
  - **Point M** on line L

$d(\mathbf{A}, L) = AH$
$= w \cdot (A-M)$
$= w \cdot A - w \cdot M$
$= w \cdot \mathbf{A} + b$

Remember **w** · **M = - b** since **M** belongs to line **L**

Largest Margin



- Prediction = $\mathbf{sign}(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})$
- "**Confidence**" = $(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})\ \boldsymbol{y}$
- **For i-th datapoint:**

$$\boldsymbol{\gamma}^{(i)} = \left(\boldsymbol{w} \cdot \boldsymbol{x}^{(i)} + \boldsymbol{b}\right)\boldsymbol{y}^{(i)}$$

- **Want to solve:**

$$\max_{w,b} \min_{i} \gamma^{(i)}$$

- **Can rewrite as**

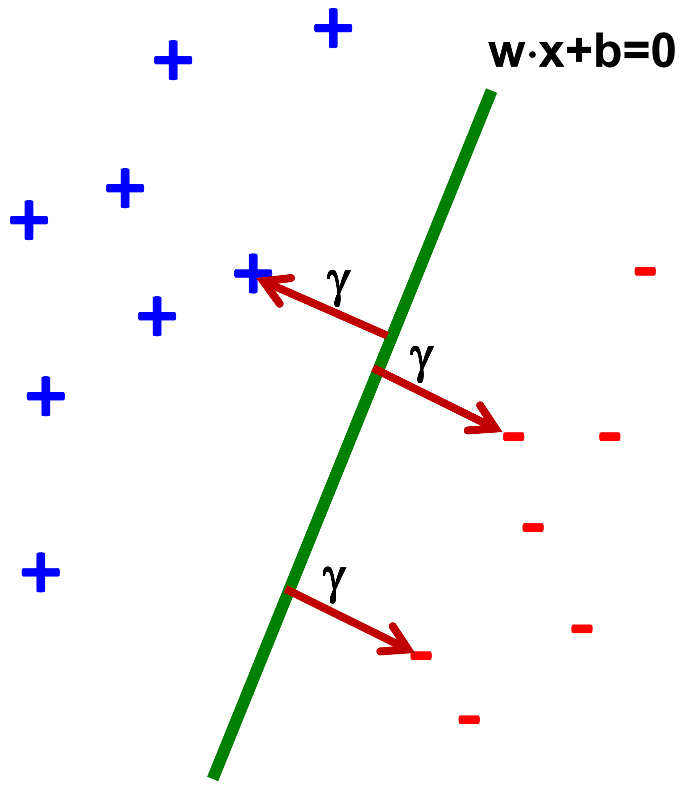
$$\max_{w,b} \gamma$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq \gamma$$

Support Vector Machine

- **Maximize the margin:**
  - **Good according to intuition, theory (c.f. "VC dimension") and practice**

$$\max_{w,b} \gamma$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq \gamma$$

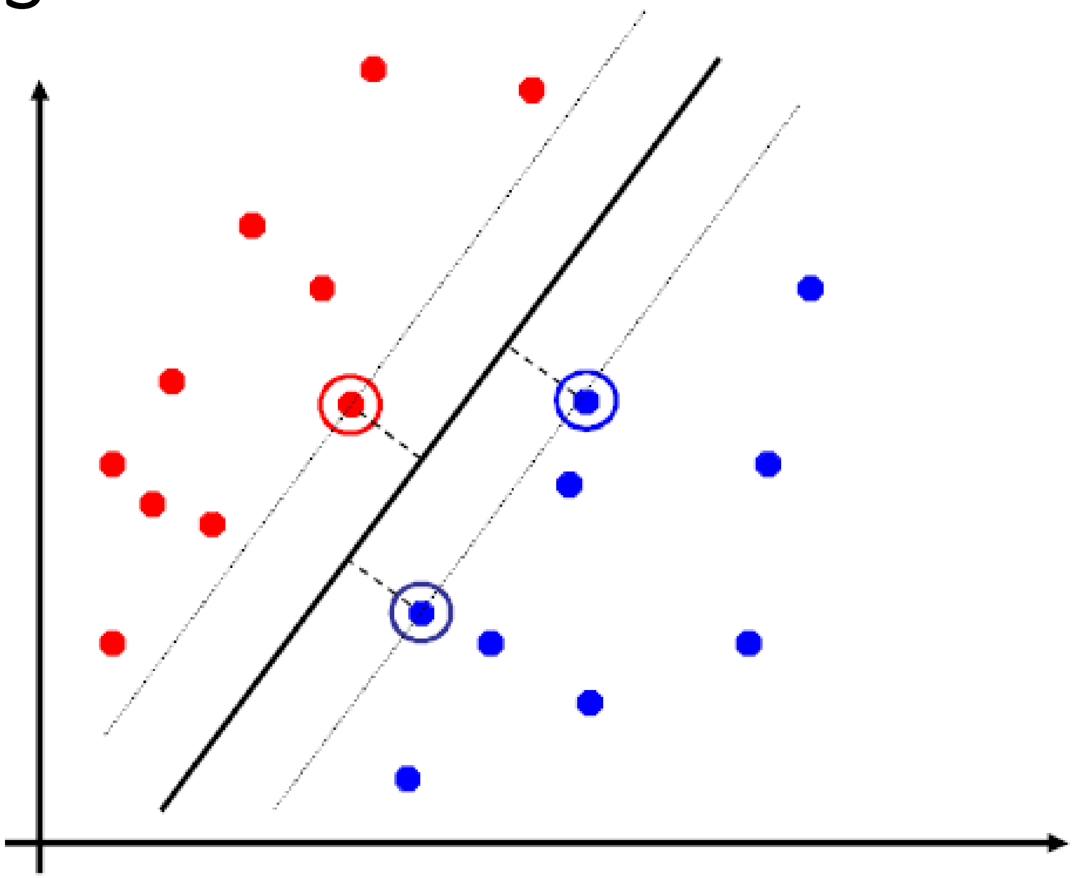
  - **γ is margin … distance from the separating hyperplane**



Maximizing the margin

Support Vector Machines

- **Separating hyperplane is defined by the support vectors**
  - Points on **+/-** planes from the solution
  - If you knew these points, you could ignore the rest
  - Generally, $n+1$ support vectors (for n dim. data)



Support Vector Machines

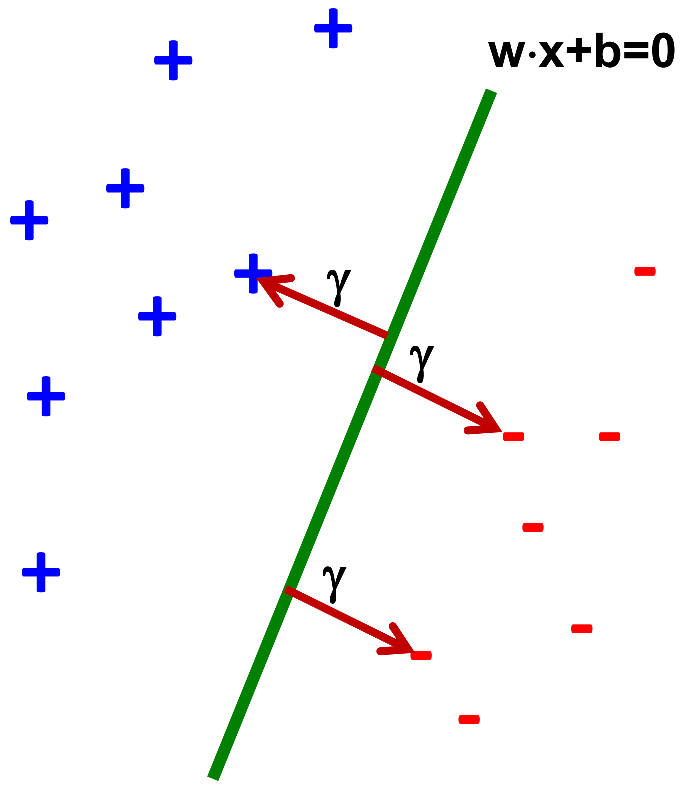
Deriving the margin

Support Vector Machine

- **Maximize the margin:**
  - **Good according to intuition, theory (c.f. “VC dimension”) and practice**

$$\max_{w,b} \gamma$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq \gamma$$

  - **γ is margin … distance from the separating hyperplane**

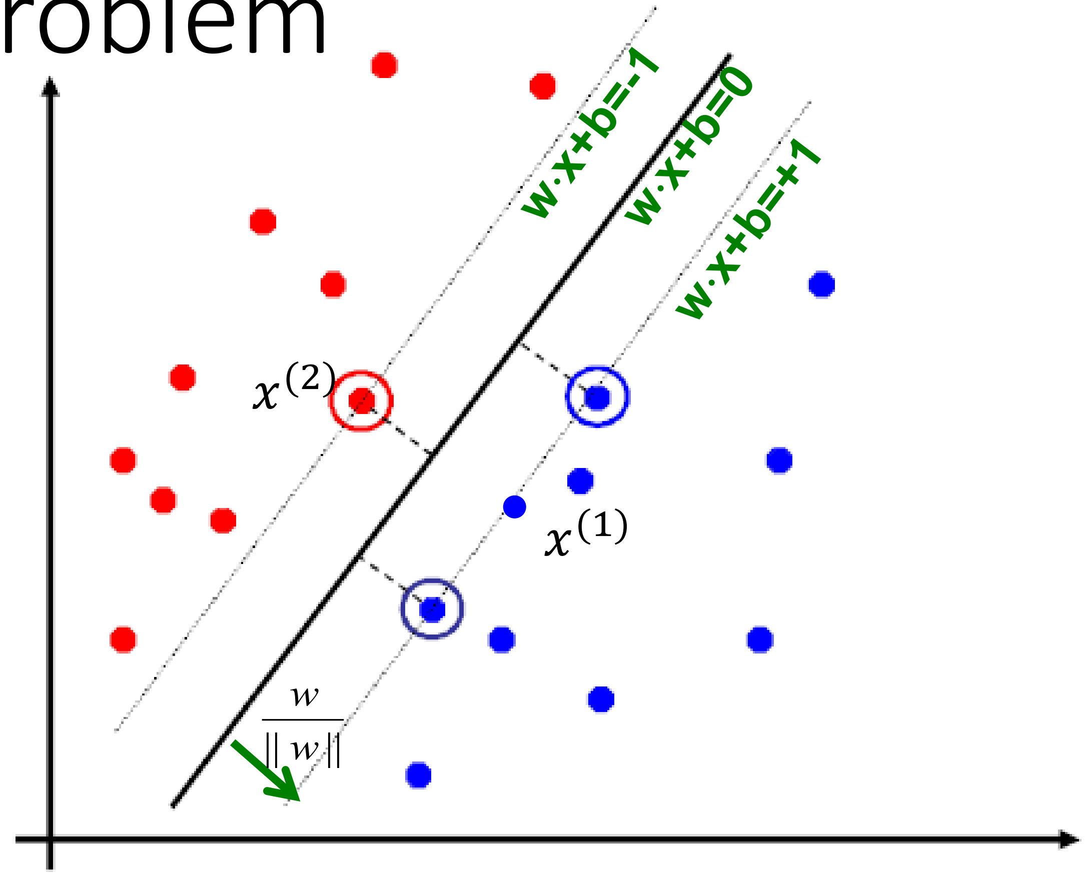


Maximizing the margin

Canonical Hyperplane: Problem

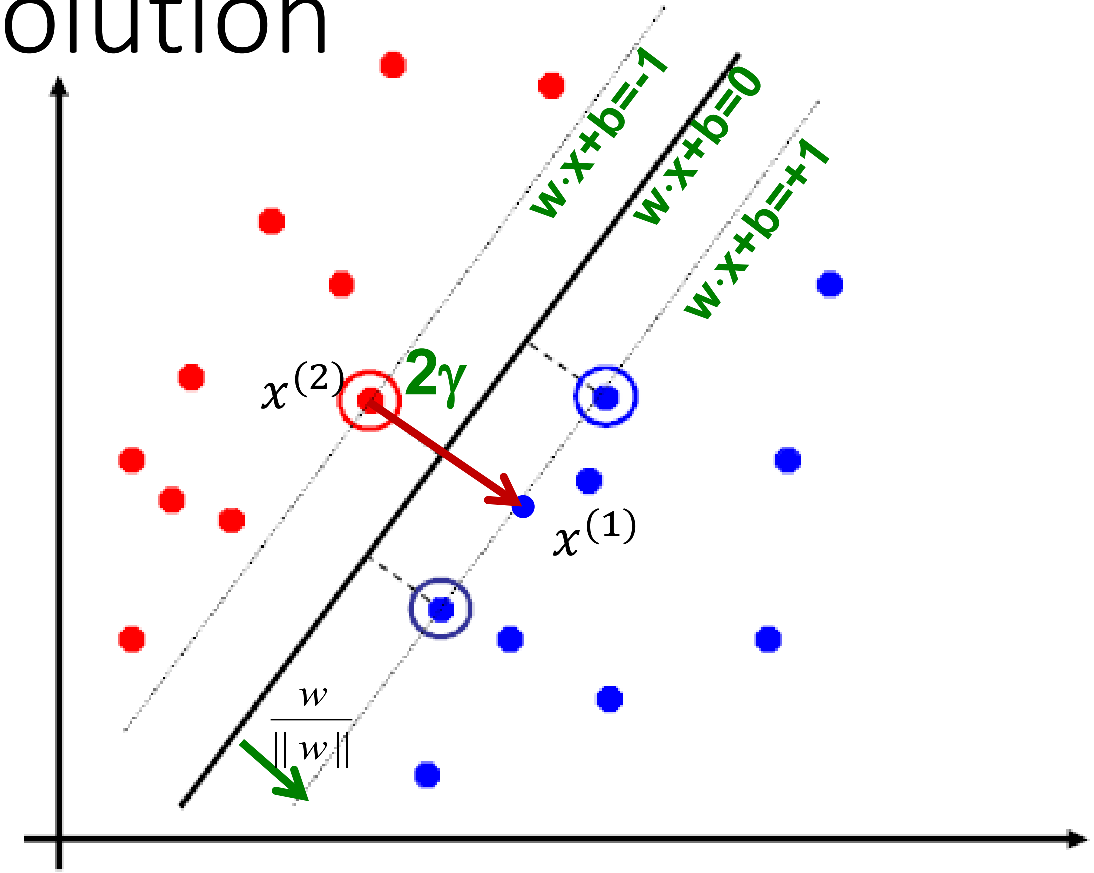
- **Problem:**
  - Let $(\boldsymbol{w} \cdot \boldsymbol{x} + \boldsymbol{b})\boldsymbol{y} = \boldsymbol{\gamma}$
    then $(\mathbf{2}\boldsymbol{w} \cdot \boldsymbol{x} + \mathbf{2}\boldsymbol{b})\boldsymbol{y} = \mathbf{2}\boldsymbol{\gamma}$
    - Scaling $\boldsymbol{w}$ increases margin!
- **Solution:**
  - Let's require **support vectors** $\boldsymbol{x}^{(j)}$ to be on the plane defined by:

$$\boldsymbol{w} \cdot \boldsymbol{x}^{(j)} + \boldsymbol{b} = \pm \mathbf{1}$$



Canonical Hyperplane: Solution

- **Want to maximize margin!**
- **What is the relation between x_1 and x_2?**
  - $\boldsymbol{x}^{(1)} = \boldsymbol{x}^{(2)} + 2\gamma \frac{\boldsymbol{w}}{||\boldsymbol{w}||}$
  - **We also know:**
    - $\boldsymbol{w} \cdot \boldsymbol{x}^{(1)} + \boldsymbol{b} = +1$
    - $\boldsymbol{w} \cdot \boldsymbol{x}^{(2)} + \boldsymbol{b} = -1$
- **So:**
  - $\boldsymbol{w} \cdot \boldsymbol{x}^{(1)} + \boldsymbol{b} = +1$
  - $\boldsymbol{w}\left(\boldsymbol{x}^{(2)} + 2\gamma \frac{\boldsymbol{w}}{||\boldsymbol{w}||}\right) + \boldsymbol{b} = +1$
  - $\underbrace{\boldsymbol{w} \cdot \boldsymbol{x}^{(2)} + \boldsymbol{b}}_{-1} + 2\gamma \frac{\boldsymbol{w} \cdot \boldsymbol{w}}{||\boldsymbol{w}||} = +1$



$$\Rightarrow \gamma = \frac{\|w\|}{w \cdot w} = \frac{1}{\|w\|}$$

Note:
$w \cdot w = \|w\|^2$

Maximizing the Margin

- **We started with**

$$\max_{w,b} \gamma$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq \gamma$$

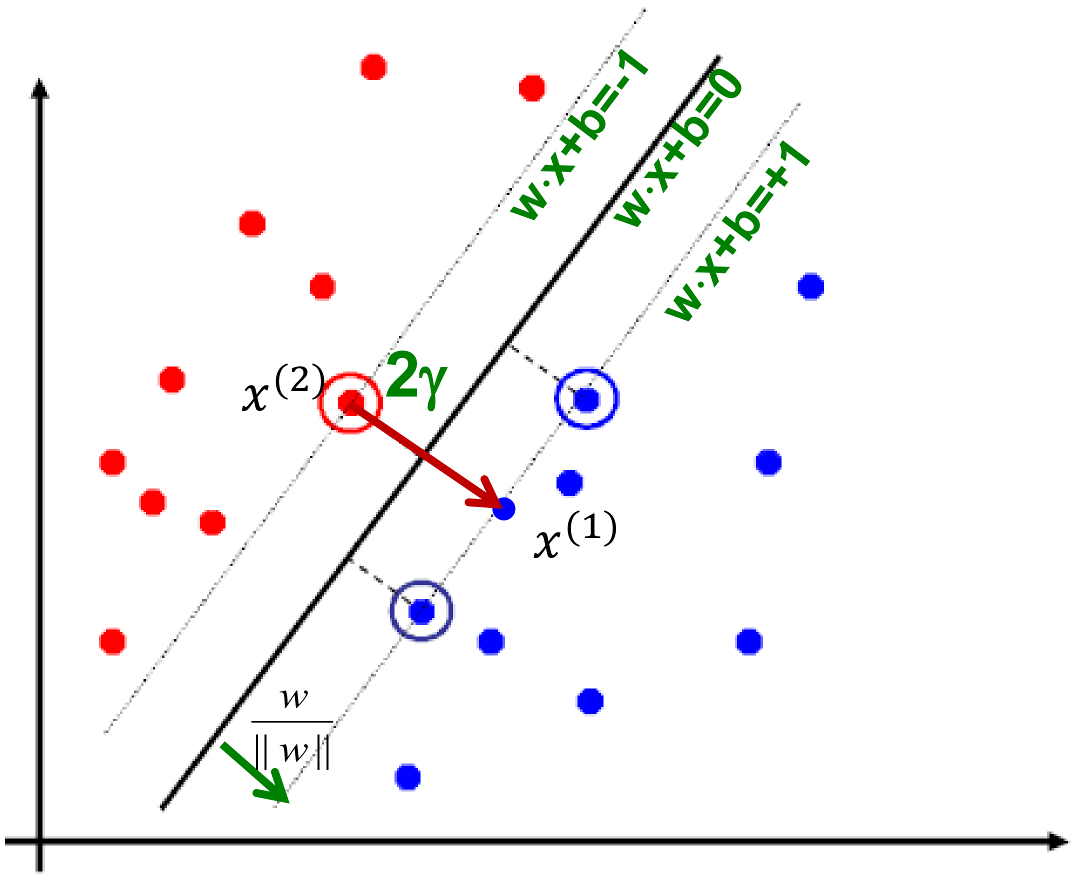
But *w* can be arbitrarily large!

- **We normalized and...**

$$\arg\max \gamma = \arg\max \frac{1}{\|w\|} = \arg\min \|w\| = \arg\min \tfrac{1}{2}\|w\|^2$$

- **Then:**

$$\min_{w,b} \frac{1}{2} ||w||^2$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1$$



Quadratic Programming

This is called SVM with "hard" constraints

A 1D Example

- Suppose we have three data points

  x = -3, y = -1

  x = -1, y = -1

  x = 2, y = 1

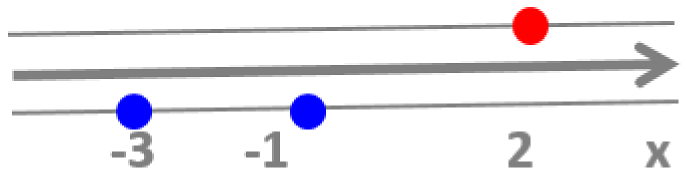
- Many separating perceptrons, $\text{sign}[wx + b]$
  - Anything with $wx + b \ = \ 0$ between -1 and 2

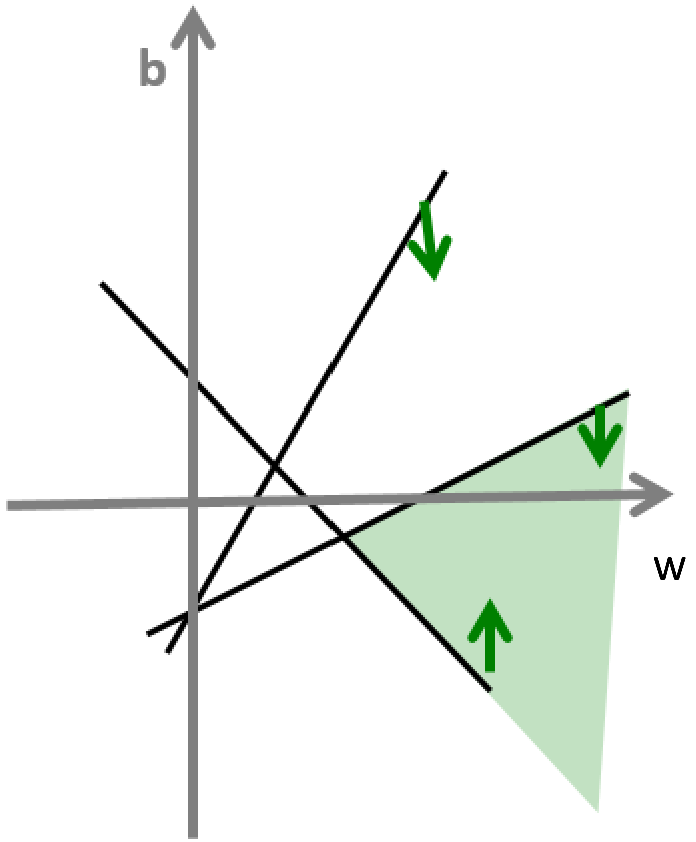
- We can write the margin constraints

  (-1)*(w (-3) + b) > 1 => b < 3w - 1

  (-1)*(w (-1) + b) > 1 => b < w - 1

  w (2) + b > 1 => b > -2w + 1





A 1D Example

- Suppose we have three data points

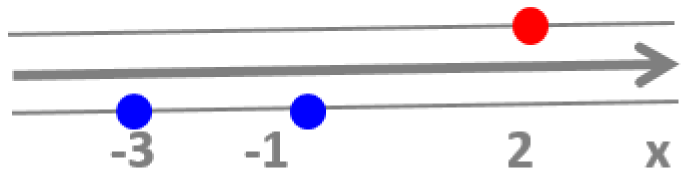
  x = -3, y = -1
  x = -1, y = -1
  x = 2, y = 1

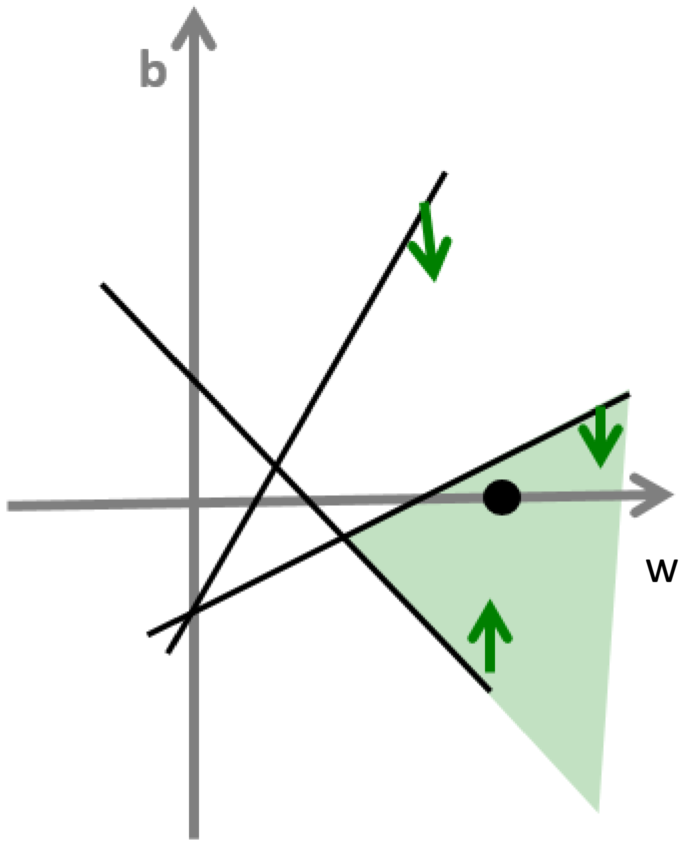
- Many separating perceptrons, $\text{sign}[wx + b]$
  - Anything with $wx + b \ = \ 0$ between -1 and 2

- We can write the margin constraints

  (-1)*(w (-3) + b) > 1 => b < 3w - 1
  (-1)*(w (-1) + b) > 1 => b < w - 1
  w (2) + b > 1 => b > -2w + 1

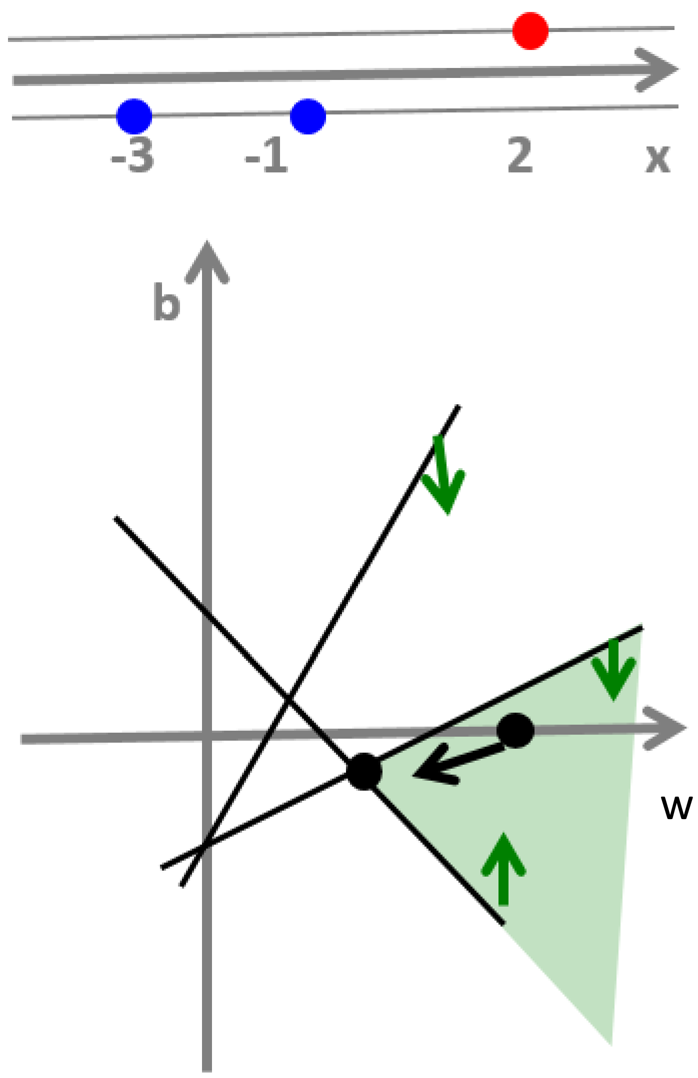
- Ex: w = 1, b = 0





A 1D Example

- Suppose we have three data points
  x = -3, y = -1
  x = -1, y = -1
  x = 2, y = 1
- Many separating perceptrons, $\text{sign}[wx + b]$
  - Anything with $wx + b \ = \ 0$ between -1 and 2
- We can write the margin constraints
  (-1)*(w (-3) + b) > 1 => b < 3w - 1
  (-1)*(w (-1) + b) > 1 => b < w - 1
  w (2) + b > 1 => b > -2w + 1
- Ex: w = 1, b = 0
- Minimize ||w|| => w = .66, b = -.33
  - Two data on the margin; constraints "tight"



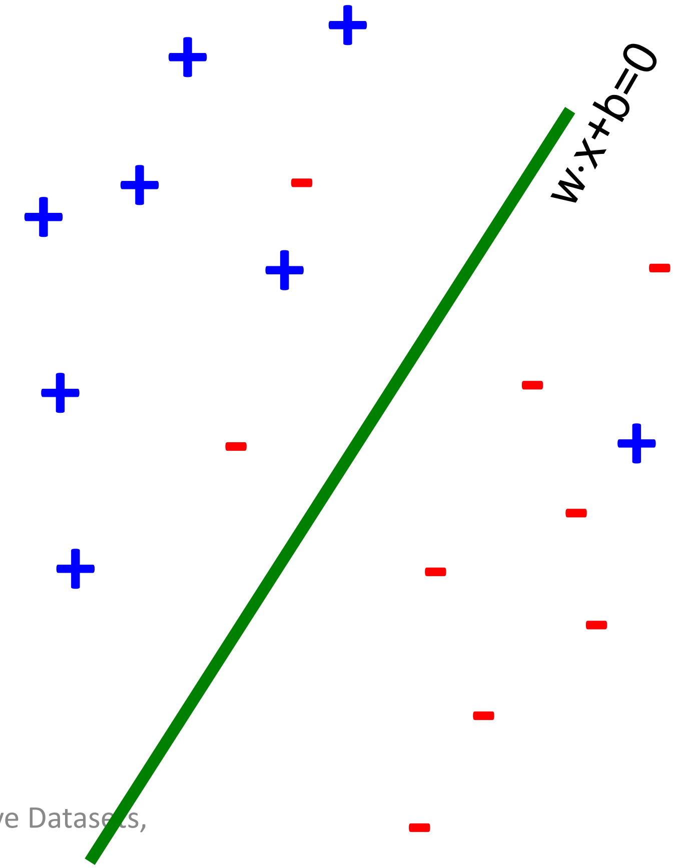
Non-linearly Separable Data

- **If data is not separable introduce penalty:**

$$\min_{w,b} \frac{1}{2} \|w\|^2 + \text{ C } \cdot (\text{\#number of mistakes})$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1$$

  - Minimize $\|w\|^2$ plus the number of training mistakes
  - Set C using cross validation

- **How to penalize mistakes?**
  - **All mistakes are not equally bad!**



Support Vector Machines

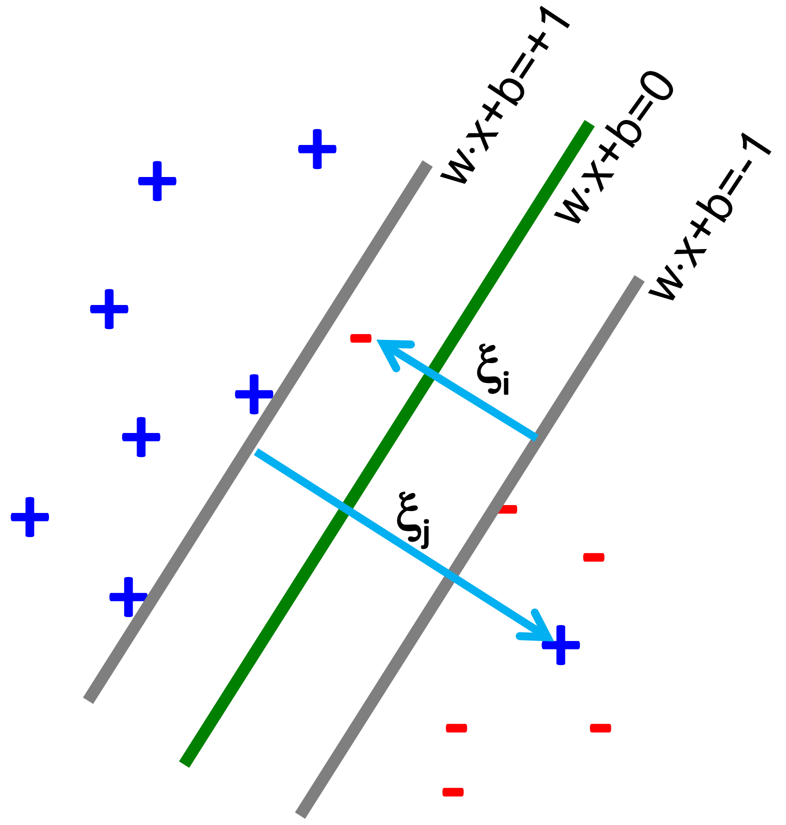
- **Introduce** **slack variables** **ξ_i**

$$\min_{w,b,\xi_i \geq 0} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i$$

- If point $x^{(i)}$ is on the wrong side of the margin then get penalty ξ_i

SVM with "soft" constraints



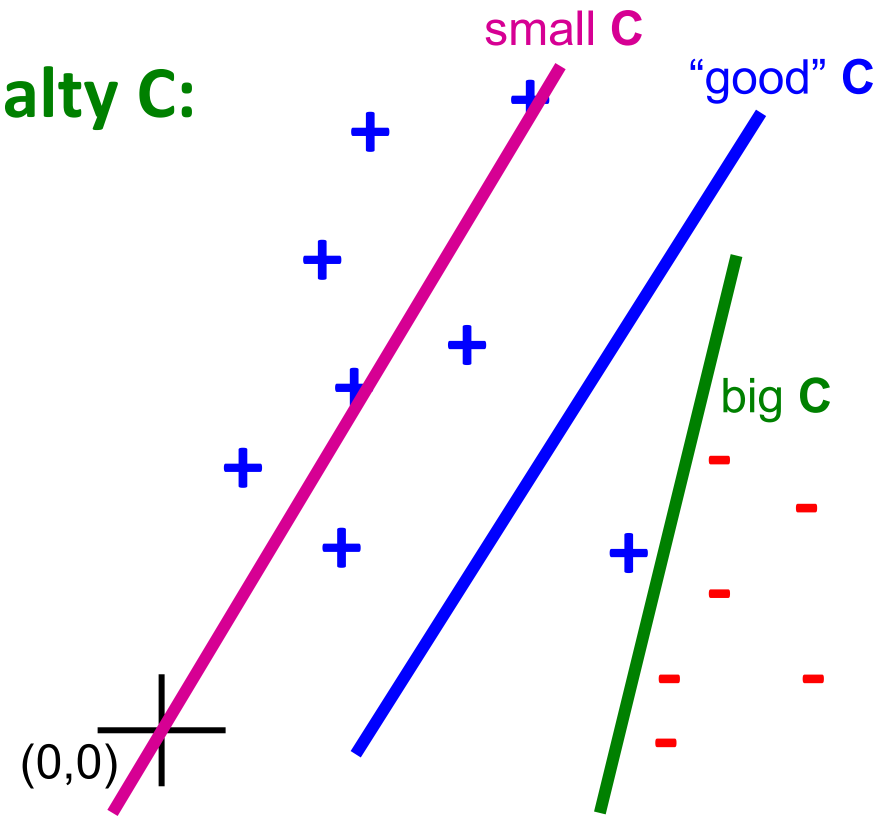
For each data point:
If margin ≥ 1, don't care
If margin < 1, pay linear penalty

Slack Penalty $\boldsymbol{C}$

$$\min_{w,b,\xi_i \geq 0} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i$$

- **What is the role of slack penalty C:**
  - **C=∞:** Only want to **w, b** that separate the data
  - **C=0:** Can set ξ_i to anything, then **w=0** (basically ignores the data)



Support Vector Machines

- Combine the constraints and the objective function

$$\min_{w,b,\xi_i \geq 0} \frac{1}{2}\|w\|^2 + C \cdot \sum_{i=1}^{n} \xi_i$$
$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i$$

$$\xi_i \geq 1 - y^{(i)}(w \cdot x^{(i)} + b)$$

⊕ $\xi_i \geq 0$

$$\xi_i = \begin{cases} 0, & \text{if } 1 - y^{(i)}(w \cdot x^{(i)} + b) \leq 0 \\ 1 - y^{(i)}(w \cdot x^{(i)} + b), & \text{if } 1 - y^{(i)}(w \cdot x^{(i)} + b) > 0 \end{cases}$$

$$\xi_i = \max\{0, 1 - y^{(i)}(w \cdot x^{(i)} + b)\}$$

Support Vector Machines

- **SVM in the "natural" form**

$$\underset{w,b}{\operatorname{argmin}}\ \underbrace{\frac{1}{2} w \cdot w}_{\text{Margin}} + C \cdot \sum_{i=1}^{n} \underbrace{\max\{0, 1 - y^{(i)}(w \cdot x^{(i)} + b)\}}_{\text{Empirical loss L (how well we fit training data)}}$$

Support Vector Machines

How to compute the margin?

SVM: How to estimate *w*?

$$\min_{w,b} \frac{1}{2} w \cdot w + C \cdot \sum_{i=1}^{n} \xi_i$$

$$s.t. \forall i, y^{(i)}(w \cdot x^{(i)} + b) \geq 1 - \xi_i$$

- **Want to estimate w and b!**
  - **Standard way:** Use a solver!
    - **Solver:** software for finding solutions to "common" optimization problems
- **Use a quadratic solver:**
  - Minimize quadratic function
  - Subject to linear constraints
- **Problem:** Solvers are inefficient for big data!

SVM: How to estimate *w*?

- **Want to minimize *J(w,b)*:**

$$J(w,b) = \frac{1}{2}\sum_{j=1}^{d}(w_j)^2 + C\sum_{i=1}^{n}\underbrace{\max\left\{0, 1 - y^{(i)}\left(\sum_{j=1}^{d} w_j x_j^{(i)} + b\right)\right\}}_{\text{Empirical loss } L(x^{(i)}, y^{(i)})}$$

- **Compute the gradient ∇_j w.r.t. w_j**

$$\nabla J_j = \frac{\partial J(w,b)}{\partial w_j} = w_j + C\sum_{i=1}^{n}\frac{\partial L(x^{(i)}, y^{(i)})}{\partial w_j}$$

$$\frac{\partial L(x^{(i)}, y^{(i)})}{\partial w_j} = 0 \qquad \text{if } y^{(i)}(w \cdot x^{(i)} + b) \geq 1$$

$$= -y^{(i)} x_j^{(i)} \qquad \text{else}$$

SVM: How to estimate *w*?

- **Batch Gradient Descent:**

Iterate until convergence:

- **For j = 1 … d**
  - **Evaluate:** $\nabla \boldsymbol{J_j} = \frac{\partial J(w,b)}{\partial w_j} = w_j + C \sum_{i=1}^{n} \frac{\partial L(x^{(i)}, y^{(i)})}{\partial w_j}$
  - **Update:**

$$\boldsymbol{w'_j} \leftarrow \boldsymbol{w_j} - \eta \nabla \boldsymbol{J_j}$$

- **w ← w'**

η…learning rate parameter
C… regularization parameter

SVM: How to estimate *w*?

- **Stochastic Gradient Descent**
  - Instead of evaluating gradient over all examples evaluate it for each **individual** training example

$$\nabla J_j(x^{(i)}) = w_j + C \cdot \frac{\partial L(x^{(i)}, y^{(i)})}{\partial w_j}$$

Notice: no summation over ***i*** anymore

- **Stochastic gradient descent:**

Iterate until convergence:
- **For i = 1 … n**
  - **For j = 1 … d**
    - **Compute:** $\nabla \boldsymbol{J}_j(x^{(i)})$
    - **Update:** $\boldsymbol{w'}_j \leftarrow \boldsymbol{w}_j - \eta \nabla \boldsymbol{J}_j(x^{(i)})$

Support Vector Machines

Example

Example: Text categorization

- **Example by Leon Bottou:**
  - **Reuters RCV1** document corpus
    - Predict a category of a document
      - One **vs.** the rest classification
  - ***n* = 781,000** training examples (documents)
  - 23,000 test examples
  - ***d* = 50,000** features
    - One feature per word
    - Remove stop-words
    - Remove low frequency words

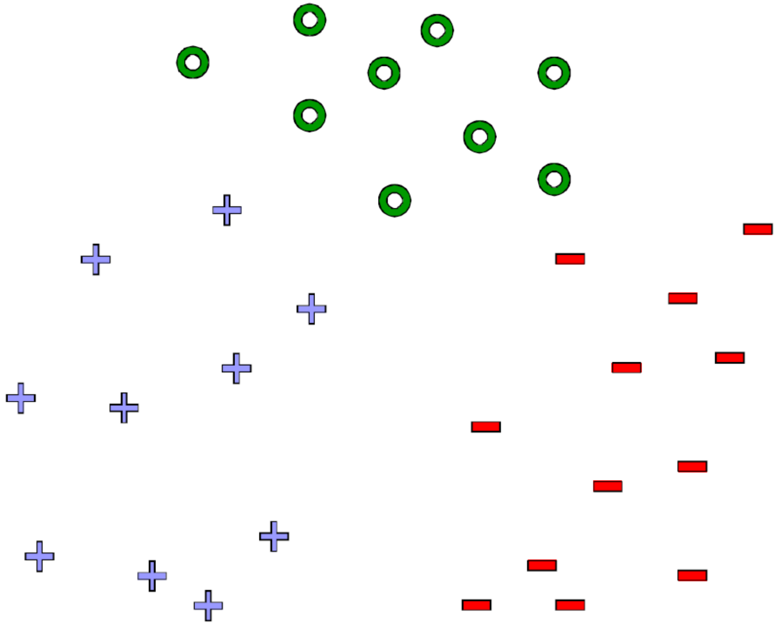
Example: Text categorization

- **Questions:**
  - **(1)** Is **SGD** successful at minimizing ***J(w,b)***?
  - **(2)** How quickly does **SGD** find the min of ***J(w,b)***?
  - **(3)** What is the error on a test set?

	Training time	*Value of J(w,b)*	*Test error*
Standard SVM	23,642 secs	0.2275	6.02%
"Fast SVM"	66 secs	0.2278	6.03%
SGD-SVM	1.4 secs	0.2275	6.02%

(1) SGD-SVM is successful at minimizing the value of ***J(w,b)***
(2) SGD-SVM is super fast
(3) SGD-SVM test set error is comparable

What about multiple classes?



- **Idea 1:**
  One against all
  Learn 3 classifiers
  - **+ vs. {o, -}**
  - **- vs. {o, +}**
  - **o vs. {+, -}**

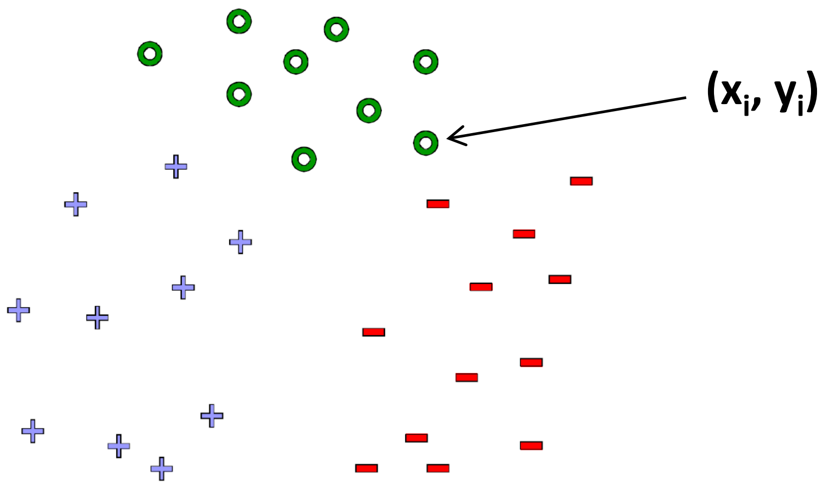
  Obtain:

  $\mathbf{w_+ \, b_+, \;\; w_- \, b_-, \;\; w_o \, b_o}$
- **How to classify?**
- Return class *c*
  $\mathbf{\arg\max_c \; w_c \, x + b_c}$

Multiclass SVM

- **Idea 2: Learn 3 sets of weights simoultaneously!**
  - For each class ***c*** estimate $\boldsymbol{w_c}$, $\boldsymbol{b_c}$
  - **Want the correct class y_i to have highest margin:**

$$\mathbf{w_{y_i} x_i + b_{y_i} \geq 1 + w_c x_i + b_c \quad \forall c \neq y_i \;, \forall i}$$



Multiclass SVM

- **Optimization problem:**

$$\min_{w,b} \frac{1}{2}\sum_{c}\|w_c\|^2 + C\sum_{i=1}^{n}\xi_i$$

$$w_{y_i}\cdot x_i + b_{y_i} \geq w_c \cdot x_i + b_c + 1 - \xi_i \qquad \forall c \neq y_i, \forall i$$

$$\xi_i \geq 0, \forall i$$

  - To obtain parameters $\boldsymbol{w_c}$, $\boldsymbol{b_c}$ (for each class $\boldsymbol{c}$) we can use similar techniques as for 2 class **SVM**

- **SVM is widely perceived a very powerful learning algorithm**