Învățare Automată (Machine Learning)

Bogdan Alexe,

bogdan.alexe@fmi.unibuc.ro

Master Informatică, anul I, 2018-2019, cursul 12

Recap - AdaBoost

- construct distribution $\mathbf{D}^{(t)}$ on {1,..., m}:
  - $\mathbf{D}^{(1)}(i) = 1/m$
  - given $\mathbf{D}^{(t)}$ and h_t: $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}}{Z_{t+1}}$

where Z_{t+1} normalization factor ($\mathbf{D}^{(t+1)}$ is a distribution): $Z_{t+1} = \sum_{i=1}^{n} D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}$

w_t is a weight: $w_t = \frac{1}{2} \ln(\frac{1}{\varepsilon_t} - 1) > 0$ as the error $\varepsilon_t < 0.5$

ε_t is the error of h_t on $\mathbf{D}^{(t)}$: $\varepsilon_t = \Pr_{i \sim D^{(t)}}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \times 1_{[h_t(x_i) \neq y_i]}$

If example $\mathbf{x}_i$ is correctly classified then $h_t(\mathbf{x}_i) = y_i$ so at the next iteration t+1 its importance (probability distribution) will be decreased to $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t}}{Z_{t+1}}$

If example x_i is misclassified then $h_t(x_i) \neq y_i$ so at the next iteration t+1 its importance (probability distribution) will be increased to $D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{w_t}}{Z_{t+1}}$

- output final/combined classifier h_{final}: $h_{final}(x) = sign(\sum_{t=1}^{T} w_t h_t(x))$

Generalization error for AdaBoost

A popular approach for constructing a weak learner is to apply the ERM rule with respect to a base hypothesis class $\mathcal{B}$ (e.g., ERM over $\mathcal{H}_{DS}^{d}$ = decision stumps in d dimensions).

Given a base hypothesis class $\mathcal{B}$ (e.g., decision stumps), the output of AdaBoost will be a member of the following class:

$$L(\mathcal{B},T) = \{h: \mathbf{R}^{d} \rightarrow \{-1,1\},\ h(x) = sign(\sum_{t=1}^{T} w_t h_t(x))\ ,\ w \in \mathbf{R}^{T},\ h \in \mathcal{B}\}$$

Remember the bias-complexity tradeoff: decompose the error of a predictor that chooses h_S from a restricted class $\mathcal{H}$ into two components:

$$L_D(h_S) = \varepsilon_{app} + \varepsilon_{est}$$

Generalization error for AdaBoost

$$L_D(h_S) = \varepsilon_{app} + \varepsilon_{est}$$

The approximation error (ε_{app})

the minimum risk achievable by a predictor in the hypothesis class $\mathcal{H}$

it is determined only by $\mathcal{H}$, enlarging it decreases the approximation error

$$\varepsilon_{app} = \min_{h \in H} L_D(h)$$

the expressiveness of $L(\mathcal{B}, T)$ grows with T

in other words, the approximation error decreases with T

The estimation error (ε_{est})

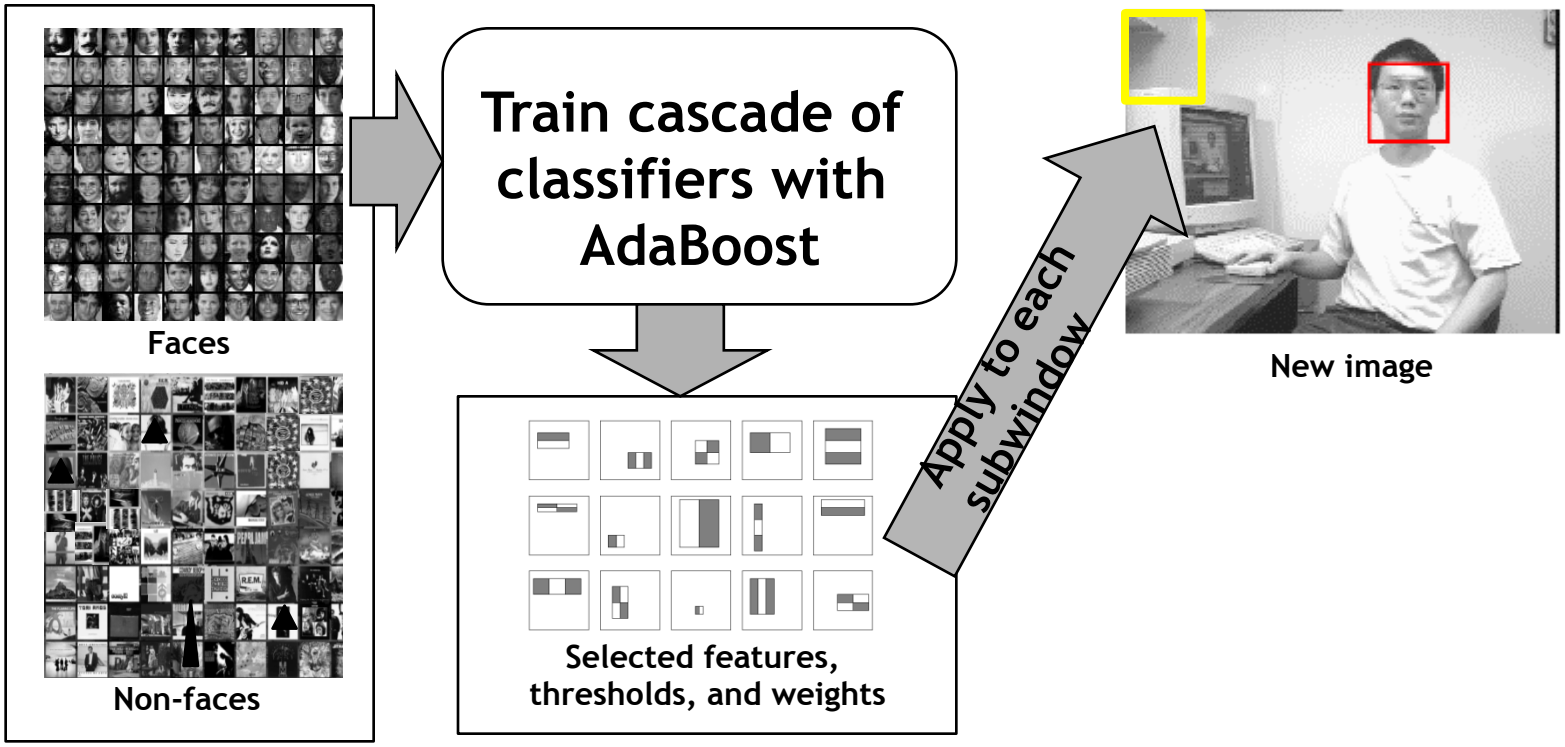
it measures how well our particular sample let us estimate the best classifier

the quality of this estimation depends on the training set size and on the size (complexity) of $\mathcal{H}$

we'll show that the estimation error increases with T

therefore, the parameter T of AdaBoost enables us to control the bias-complexity trade-off

Viola-Jones detector: summary



Train with 5K positives, 350M negatives
Real-time detector using 38 layer cascade
6061 features in all layers

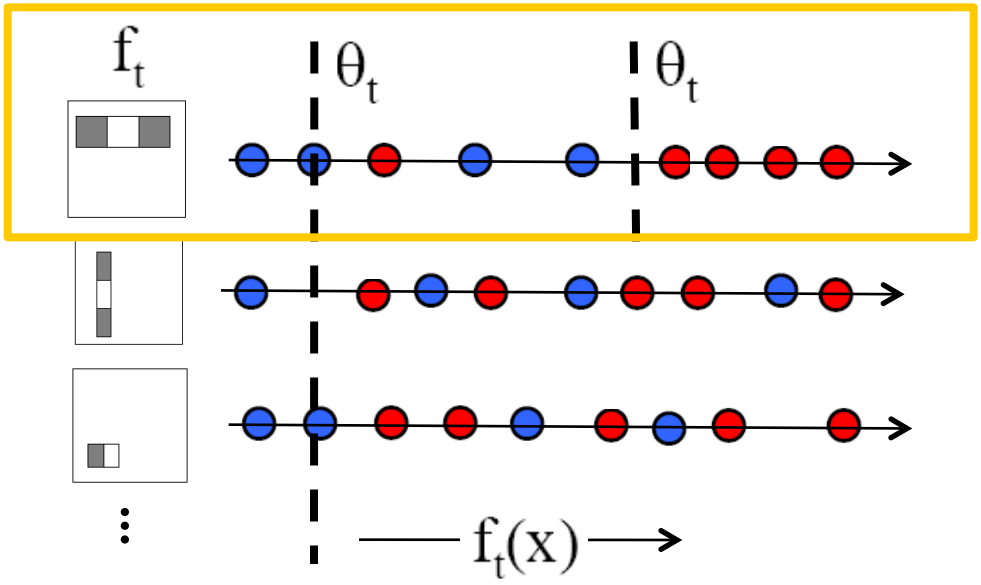
Implementation available in OpenCV, Matlab

P. Viola and M. Jones. *Rapid object detection using a boosted cascade of simple features.* CVPR 2001.

P. Viola and M. Jones. *Robust real-time face detection.* IJCV 57(2), 2004.

Viola-Jones detector: AdaBoost

- Want to select the single rectangle feature and threshold that best separates positive (faces) and negative (non-faces) training examples, in terms of *weighted* error.



Outputs of a possible rectangle feature on faces and non-faces.

Resulting weak classifier:

imum number of examples are misclassified. A weak classifier $h_j(x)$ thus consists of a feature f_j, a threshold θ_j and a parity p_j indicating the direction of the inequality sign:

$$h_j(x) = \begin{cases} 1 & \text{if } p_j f_j(x) < p_j \theta_j \\ 0 & \text{otherwise} \end{cases}$$

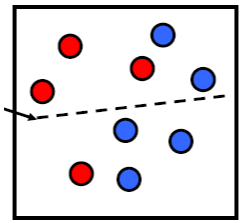
For next round, reweight the examples according to errors, choose another filter/threshold combo.

The resulting weak classifier is in fact from $\mathcal{H}_{DS}{}^{d} = \{h_{i,\theta,b}: \mathbf{R}^d \rightarrow \{-1,1\}$, $h_{i,\theta,b}(\mathbf{x}) = \text{sign}(\theta - x_i) \times b$, $1 \leq i \leq d$, $\theta \in \mathbf{R}$, $b \in \{-1,+1\}\}$

AdaBoost Algorithm

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

Start with uniform weights on training examples



$\{x_1, \ldots x_n\}$

For T rounds

Evaluate *weighted* error for each feature, pick best.

Re-weight the examples:
Incorrectly classified -> more weight
Correctly classified -> less weight

Final classifier is combination of the weak ones, weighted according to error they had.

Freund & Schapire 1995

Today's lecture: Overview

- Assignment 2
- Support Vector Machines

Assignment 2

Assignment 2 – good to know

- modified version of the assignment from last week
- 3 problems = 3.5 points
- 1 bonus problem = 1 point
- deadline: Wednesday, 5th of June 2019, 23:59
  - late submission policy: maximum 3 days allowed, -10% (= 0.35 points) for each day
  - submit hard copy

  OR
  - send a pdf with a scan of your solution to bogdan.alexe@fmi.unibuc.ro

- for every problem write clear explanations, proofs to justify your answer (if you write just some indications you will not get too many points)

- do not share/copy the solution with/from your colleagues: you + your colleague/s will get 0 points

Problem 1

Problem 1 (1 point)

Let $\mathcal{X} = \mathbf{R}$ and consider $\mathcal{H}$ the class of 3-piece classifiers (signed intervals):

$\mathcal{H} = \{h_{a,b,s}: \mathbf{R} \to \{-1,1\}, a \le b, s \in \{-1,+1\}\}$, where

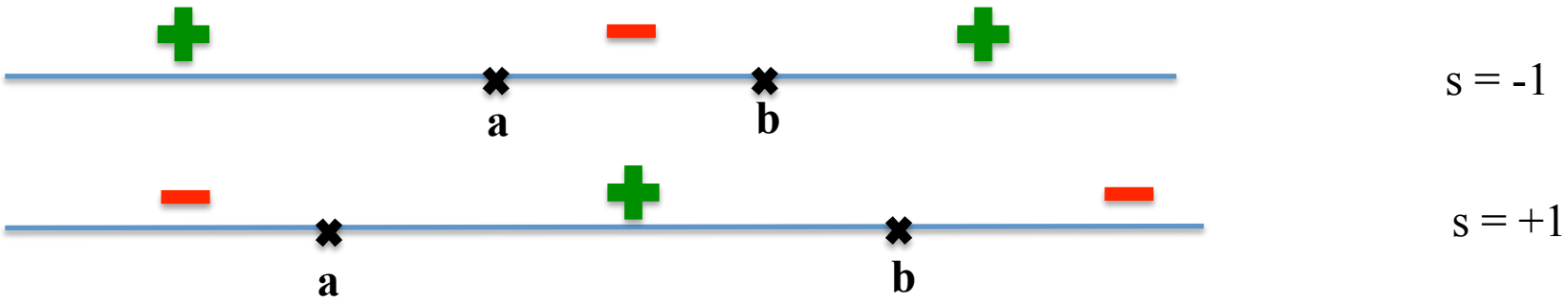
$$h_{a,b,s}(x) = \begin{cases} s & \text{if } x \in [a,b] \\ -s & \text{if } x \notin [a,b] \end{cases}$$

Give an efficient ERM algorithm for class $\mathcal{H}$ and compute its complexity for each of the following cases:

a. realizable case.
b. agnostic case.

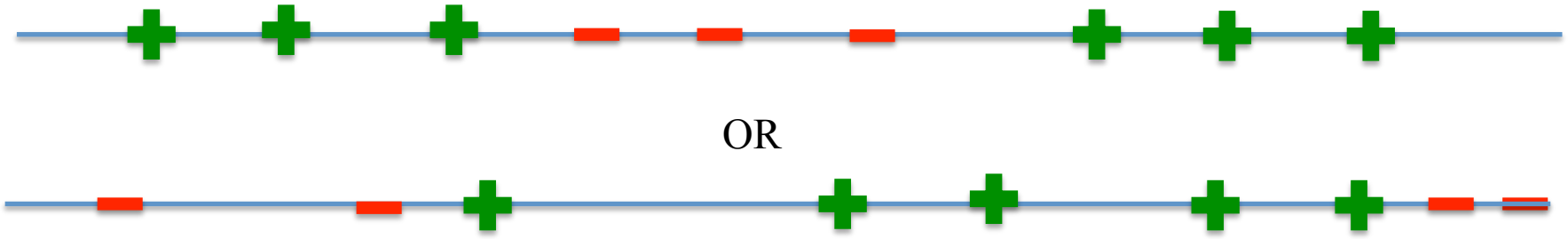
Justify the correctness of your algorithm.

Note: Solutions which do not give the most efficient algorithm in terms of complexity will be penalized.



Problem 1

Realizable case: there exist h^* in H that correctly labels all examples from a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$, $y_i = h^*(x_i)$

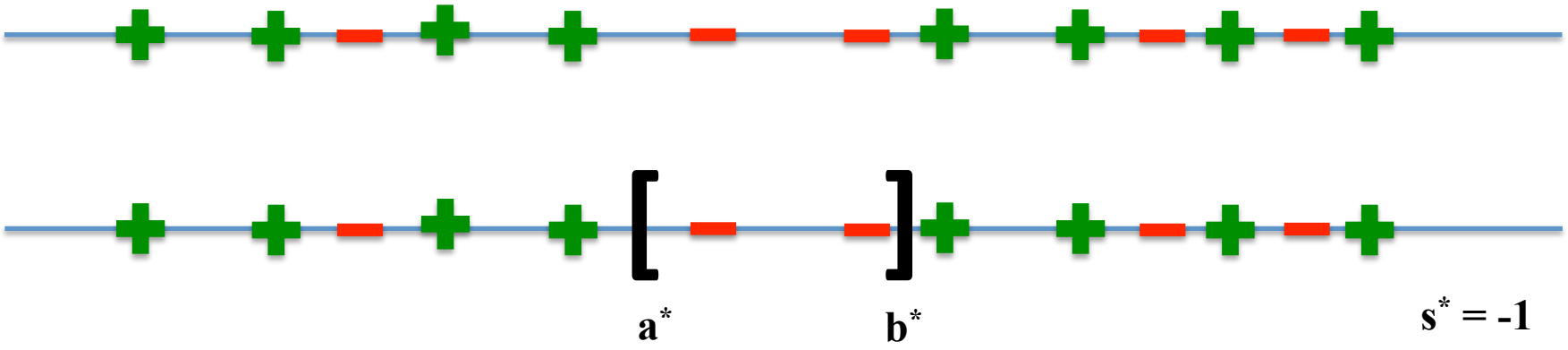


OR

Give the algorithm and compute its complexity as a function of *m* (number of examples). Make sure your algorithm works on particular cases.

Problem 1

Agnostic case = unrealizable case: there might not exist h^* in H that labels correctly all examples from a training set $S = \{(x_1, y_1), \ldots, (x_m, y_m)\}$



Give the algorithm and compute its complexity as a function of m (number of examples). Make sure your algorithm works on particular cases.

Problem 2

Problem 2 (1 point) - problem 8.2 in the book "Understanding Machine Learning: From Theory to Algorithms".

Let $\mathcal{H}_1, \mathcal{H}_2, \ldots$ be a sequence of hypothesis classes for binary classification. Assume that there is a learning algorithm that implements the ERM rule in the realizable case such that the output hypothesis of the algorithm for each class $\mathcal{H}_n$ only depends on $O(n)$ examples out of the training set. Furthermore, assume that such a hypothesis can be calculated given these $O(n)$ examples in time $O(n^2)$, and that the empirical risk of each such hypothesis can be evaluated in time $O(mn)$. For example, if $\mathcal{H}_n$ is the class of axis aligned rectangles in $\mathbf{R}^n$, we saw that it is possible to find an ERM hypothesis in the realizable case that is defined by at most $2n$ examples. Prove that in such cases, it is possible to find an ERM hypothesis for $\mathcal{H}_n$ in the unrealizable case in time $O(m \times n \times m^{O(n)})$.

O(n) in the book, we consider in the assignment $O(n^2)$

n is some parameter related to the size/complexity of domain/hypothesis class

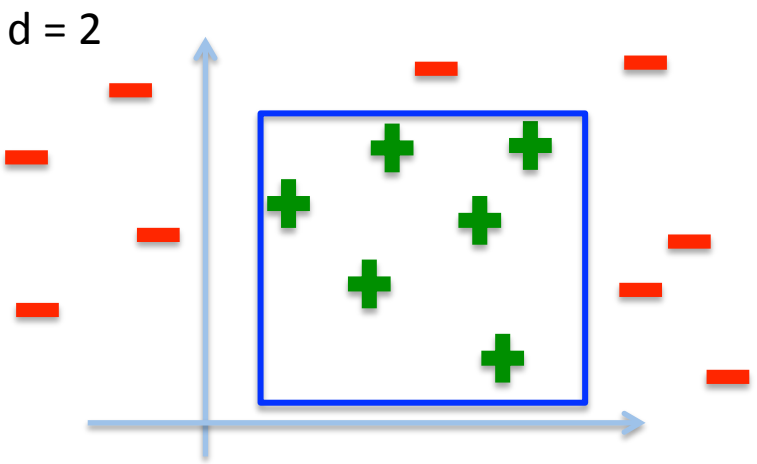
Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^d$ = the class of axis aligned rectangles in $\mathbf{R}^d$

$$H_{rec}^d = \{h_{a_1,b_1,a_2,b_2,\ldots,a_d,b_d} : R^d \to \{0,1\} \mid a_1 \le b_1, a_2 \le b_2, \ldots, a_d \le b_d, a_i \in R, b_i \in R\}$$

$$h_{a_1,b_1,a_2,b_2,\ldots,a_d,b_d}(x_1, x_2 \ldots, x_d) = \begin{cases} 1, if\, a_1 \le x_1 \le b_1, a_2 \le x_2 \le b_2, \ldots, a_d \le x_d \le b_d \\ 0, otherwise \end{cases}$$

- consider the realizable case:
  - there exists a rectangle h* in $\mathcal{H}_{\text{rec}}^d$ with real risk = 0



We have shown in the seminar class that:

- the algorithm that returns the rectangle enclosing all positive examples is ERM
- the output hypothesis it depends on maximum 2×d positive examples. Given these 2×d positive examples the output hypothesis can be computed in $O(d^2)$ operations as for each dimension, the algorithm has to find the minimal and the maximal values among the the exmples provided.
- empirical risk of a rectangle can be computed in O(md), as we check for each example from the training set and for each dimension if the points is inside the rectangle

Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{\text{rec}}^{\text{d}}$ = the class of axis aligned rectangles in $\mathbf{R}^{\text{d}}$

$$H_{rec}^{d} = \{h_{a_1,b_1,a_2,b_2,\ldots,a_d,b_d} : R^d \to \{0,1\} \mid a_1 \le b_1, a_2 \le b_2, \ldots, a_d \le b_d, a_i \in R, b_i \in R\}$$

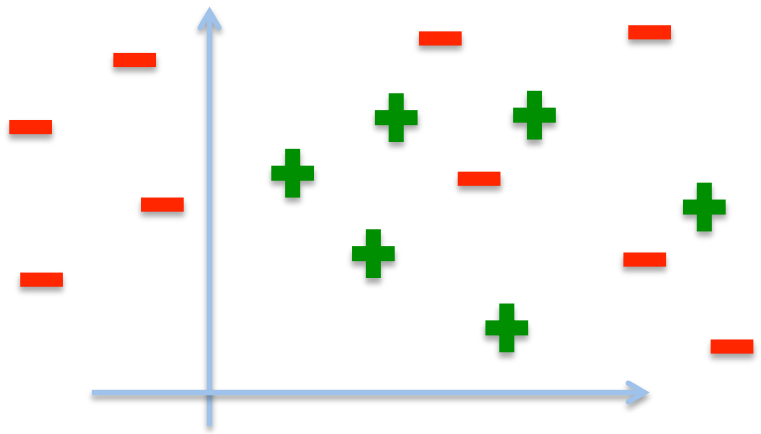
$$h_{a_1,b_1,a_2,b_2,\ldots,a_d,b_d}(x_1, x_2 \ldots, x_d) = \begin{cases} 1, if\, a_1 \le x_1 \le b_1, a_2 \le x_2 \le b_2, \ldots, a_d \le x_d \le b_d \\ 0, otherwise \end{cases}$$

- consider the agnostic case:
  - distribution $\mathcal{D}$ over $\mathcal{Z} = \mathbf{R}^{\text{d}} \times \{0,1\}$ (a sample could get both labels)
  - if there exist a labeling function f this might not be in $\mathcal{H}_{\text{rec}}^{\text{d}}$

Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{rec}^{d}$ = the class of axis aligned rectangles in $\mathbf{R}^d$
- consider that we have a sample S of size m
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in R^d?
  - go over all axis aligned rectangles in R^d and choose the best one (based on minimizing the error on the training data)

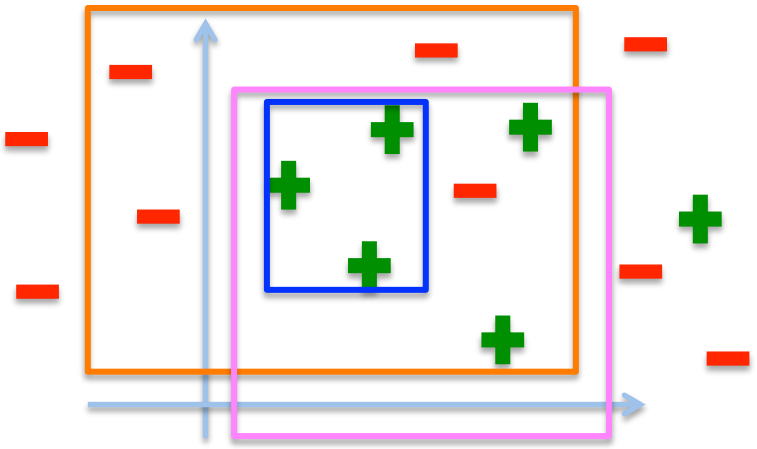
d = 2



Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{rec}^d$ = the class of axis aligned rectangles in $\mathbf{R}^d$
- consider that we have a sample S of size m
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in R^d?
  - go over all axis aligned rectangles in R^d and choose the best one (based on minimizing the error on the training data)

d = 2



error: (1 + 4)/ (6 + 9) = 5/15

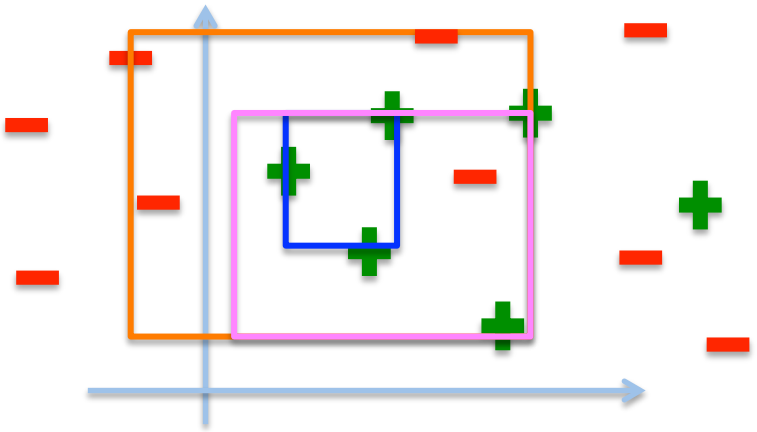
error: (3 + 0)/ (6 + 9) = 3/15

error: (1 + 1)/ (6 + 9) = 2/15

Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{rec}^d$ = the class of axis aligned rectangles in $\mathbf{R}^d$
- consider that we have a sample S of size m
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in R^d?
  - go over all axis aligned rectangles in R^d and choose the best one (based on minimizing the error on the training data)

d = 2



error: (1 + 4)/ (6 + 9) = 5/15

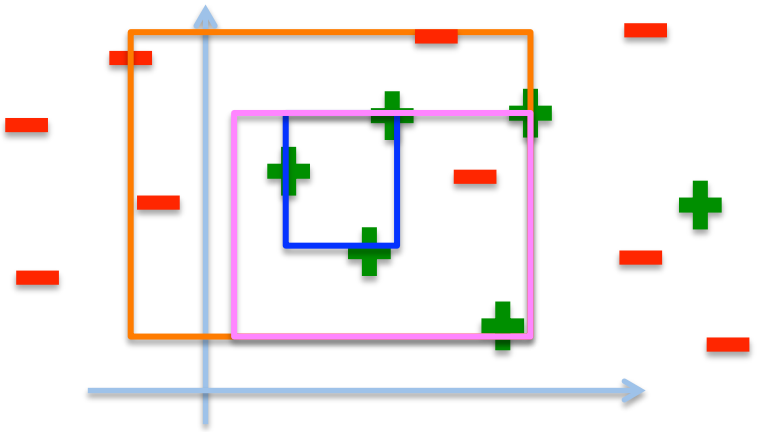
error: (3 + 0)/ (6 + 9) = 3/15

error: (1 + 1)/ (6 + 9) = 2/15

- the number of all possible rectangles can be reduced to all possible rectangles that have points of S on every boundary edge (very efficient algorithm)

Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{rec}^d$ = the class of axis aligned rectangles in $\mathbf{R}^d$
- consider that we have a sample S of size m
- what is the runtime of the ERM algorithm?
  - how long it will take to find the best rectangle in R^d?
  - Step 1: generate all the rectangles based on the sample points in R^d
  - Step 2: for each such rectangle compute the training error
  - Step 3: choose the rectangle with the smallest training error



error: (1 + 4)/ (6 + 9) = 5/15

error: (3 + 0)/ (6 + 9) = 3/15

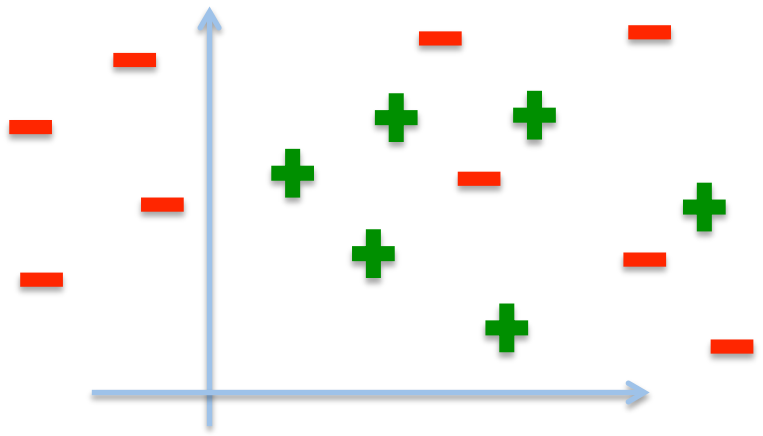
error: (1 + 1)/ (6 + 9) = 2/15

- how many possible rectangles can we construct based on the points in the sample S?

Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{rec}^{d}$ = the class of axis aligned rectangles in $\mathbf{R}^d$
- how many possible rectangles can we construct based on the points in the sample S?

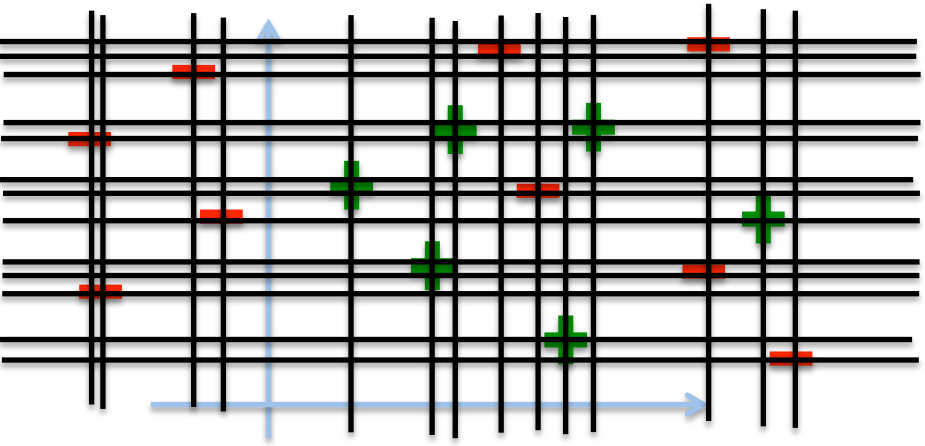
d = 2



Example: Learning axis aligned rectangles in $\mathbf{R}^d$

- $\mathcal{H}_{rec}^d$ = the class of axis aligned rectangles in $\mathbf{R}^d$
- how many possible rectangles can we construct based on the points in the sample S?

d = 2



Every such rectangle is determined by at most 2d points from S

So there are at most $|S|^{2d} = m^{2d}$ such rectangles.

For each rectangle (at most $O(m^{2d})$) we need to iterate over all examples ($O(m)$) to compute the training error ($O(d)$).

So, the runtime is: $O(d \times m^{2d+1})$

Problem 3

Problem 3 (1.5 points)

Consider the boosting algorithm described (page 4) in the article "Rapid object detection using a boosted cascade of simple features", P. Viola and M. Jones, CVPR 2001. Consider that the number of positives is equal with the number of negative examples ($l = m$).

a. Prove that the distribution w_{t+1} obtained at round t + 1 based on the algorithm described in the article is the same with the distribution $D^{(t+1)}$ obtained based on the procedure described in lecture 11 (slides 11-13).

b. Prove that the two final classifiers (the one described in the article and the one described in the lecture) are equivalent.

c. Assume that at each iteration t of AdaBoost, the weak learner returns a hypothesis h_t for which the error ε_t satisfies $\varepsilon_t \leq 1/2 - \gamma$, $\gamma > 0$. What is the probability that the classifier h_t (selected as the best weak learner at iteration t) will be selected again at iteration t+1? Justify your answer.

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ where $y_i = 0, 1$ for negative and positive examples respectively.
- Initialize weights $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$ for $y_i = 0, 1$ respectively, where m and l are the number of negatives and positives respectively.
- For $t = 1, \ldots, T$:
  1. Normalize the weights,
  $$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$
  so that w_t is a probability distribution.
  2. For each feature, j, train a classifier h_j which is restricted to using a single feature. The error is evaluated with respect to w_t, $\epsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- The final strong classifier is:
$$h(x) = \begin{cases} 1 & \sum_{t=1}^{T} \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{otherwise} \end{cases}$$
where $\alpha_t = \log \frac{1}{\beta_t}$

- construct distribution $\mathbf{D}^{(t)}$ on {1,..., m}:
  - $\mathbf{D}^{(1)}(i) = 1/m$
  - given $\mathbf{D}^{(t)}$ and h_t: $D^{(t+1)}(i) = \dfrac{D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}}{Z_{t+1}}$

where Z_{t+1} normalization factor ($\mathbf{D}^{(t+1)}$ is a distribution):

$$Z_{t+1} = \sum_{i=1}^{n} D^{(t)}(i) \times e^{-w_t h_t(x_i) y_i}$$

w_t is a weight: $w_t = \dfrac{1}{2} \ln(\dfrac{1}{\varepsilon_t} - 1) > 0$ as the error $\varepsilon_t < 0.5$

ε_t is the error of h_t on $\mathbf{D}^{(t)}$:

$$\varepsilon_t = \Pr_{i \sim D^{(t)}}[h_t(x_i) \neq y_i] = \sum_{i=1}^{m} D^{(t)}(i) \times 1_{[h_t(x_i) \neq y_i]}$$

If example $\mathbf{x}_i$ is correctly classified then $h_t(\mathbf{x}_i) = y_i$ so at the next iteration t+1 its importance (probability distribution) will be decreased to

$$D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{-w_t}}{Z_{t+1}}$$

If example x_i is misclassified then $h_t(x_i) \neq y_i$ so at the next iteration t+1 its importance (probability distribution) will be increased to

$$D^{(t+1)}(i) = \frac{D^{(t)}(i) \times e^{w_t}}{Z_{t+1}}$$

- output final/combined classifier h_{final}:

$$h_{final}(x) = sign(\sum_{t=1}^{T} w_t h_t(x))$$

Bonus Problem

Bonus Problem (1 point)

Consider H^d_{2DNF} the class of 2-term disjunctive normal form formulae consisting of hypothesis of the form h: $\{0,1\}^d \rightarrow \{0,1\}$,

$$h(\mathbf{x}) = A_1(\mathbf{x}) \vee A_2(\mathbf{x}),$$

where $A_i(\mathbf{x})$ is a Boolean conjunction of literals (in H^d_{conj}).

It is known that the class H^d_{2DNF} is not efficient properly learnable but can be learned improperly considering the class H^d_{2CNF}.

a. Give a γ-weak-learner algorithm for learning the class H^d_{2DNF} which is not a stronger PAC learning algorithm for H^d_{2DNF} (like the one considering H^d_{2CNF}). Prove that this algorithm is a γ-weak-learner algorithm for H^d_{2DNF}.

b. Compute the bound obtained with $L(\mathcal{B},T)$. Compare this bound with the bound based on VC-dimension of the class H^d_{2CNF}.

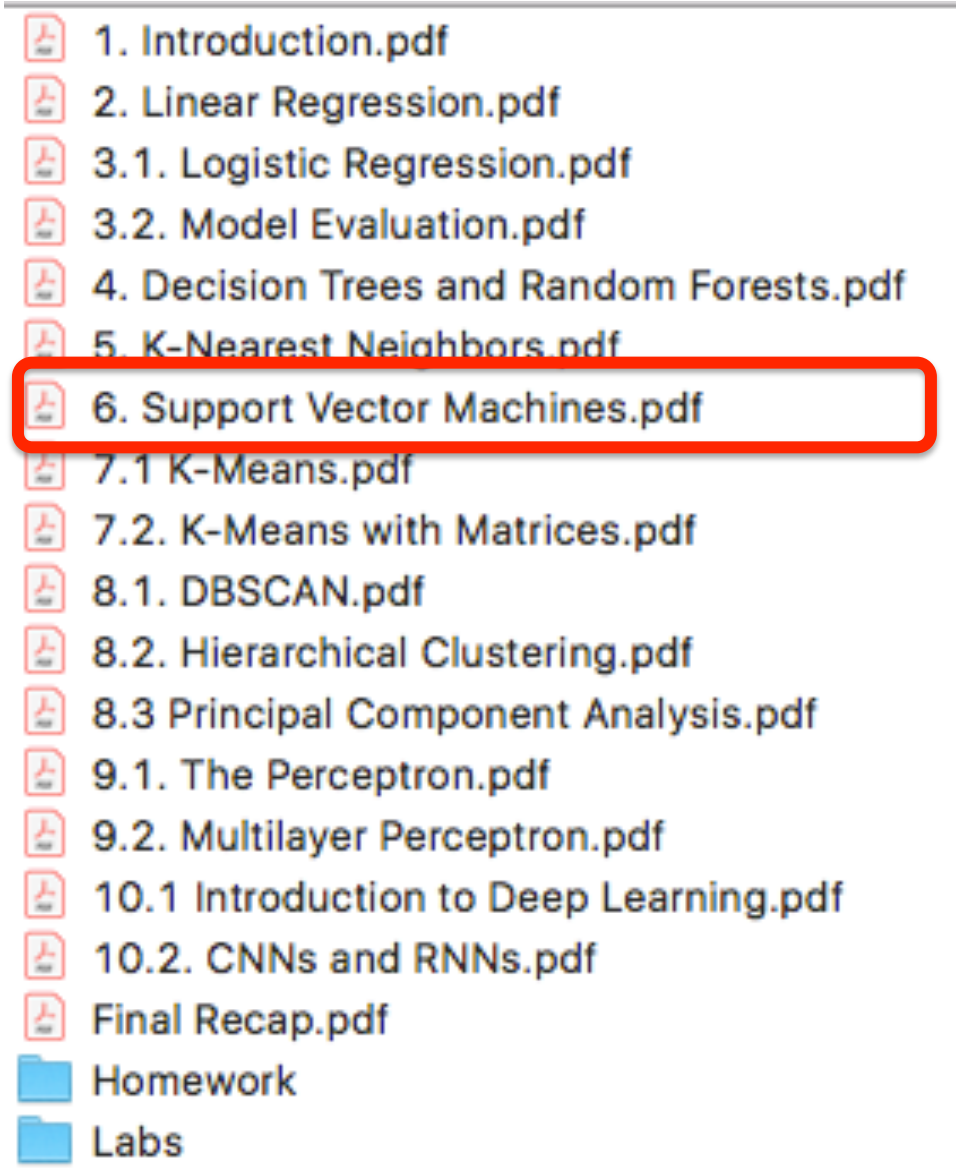
Hint: Find an algorithm that returns h(x)=0 or the disjunction of 2 literals.

Support vector machines

Practical Machine Learning (sem 1) …

- topicurile abordate în Învățare Automată cu Python din semestrul 1
- diverși algoritmi utilizați în învățarea automată
- materiale aici (Sparktech)

http://goo.gl/1iG3v9



Support Vector Machines

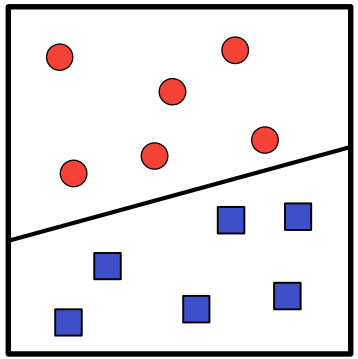
Separating data with the **largest margin**

Faculty of Mathematics and Computer Science, University of Bucharest
and
Sparktech Software

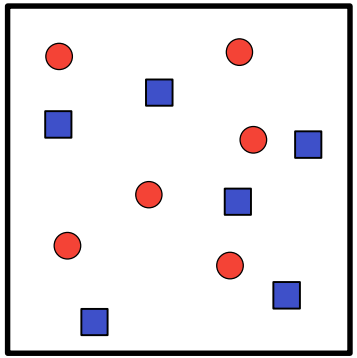
Academic Year 2018/2019, 1st Semester

Linear Separability

- Two sets of points are **linearly separable**, if there is at least one **hyperplane** which completely separates them.
- An **n-dimensional hyperplane** is a flat *n-1* dimensional subset of the space.
  - A *1d* hyperplane is a **point.**
  - A *2d* hyperplane is a **line.**
  - A *3d* hyperplane is a **plane.**



Separable



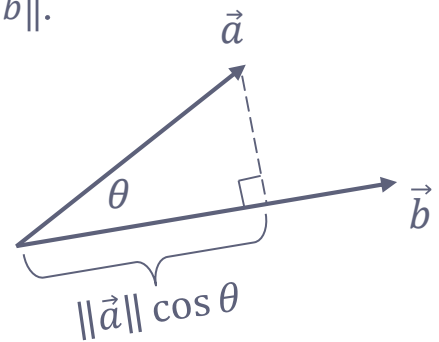
Non-separable

Refresher – Dot Product

- An operation which takes two vectors and returns a **scalar** value.
- Also called the *scalar product* or *inner product*.

$$\vec{a} \cdot \vec{b} = \langle \vec{a}, \vec{b} \rangle = \|\vec{a}\| \|\vec{b}\| \cos\theta = \sum_i a_i b_i$$

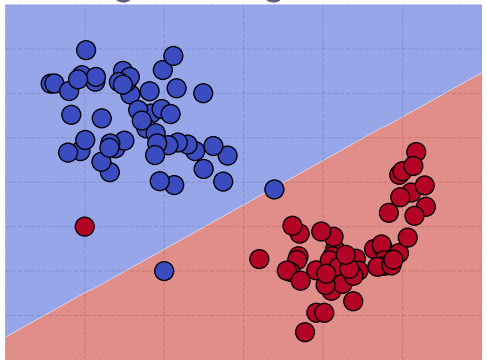
- It can be interpreted as a **similarity measure** between vectors.
- If we write it as $(\|\vec{a}\| \cos\theta)\|\vec{b}\|$, it can be viewed as the **length of the projection** of $\vec{a}$ on $\vec{b}$, measured in units of length $\|\vec{b}\|$.



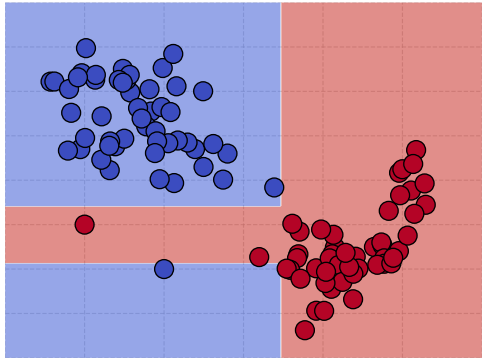
Decision Boundaries

- Classifications means learning a **good decision boundary**.
- Different algorithms have different *allowed hypotheses* and, therefore, different types of decision boundaries.

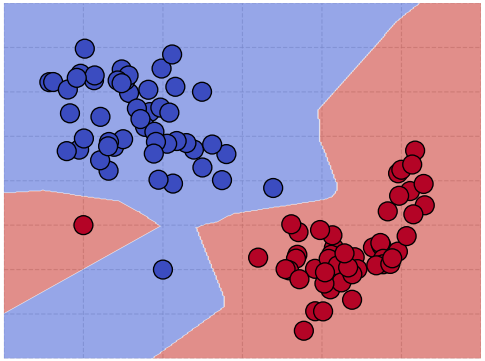
Logistic Regression



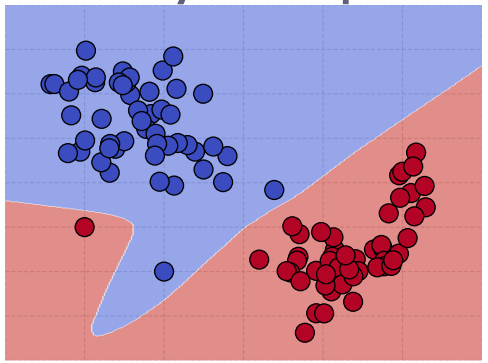
Decision Tree



KNN

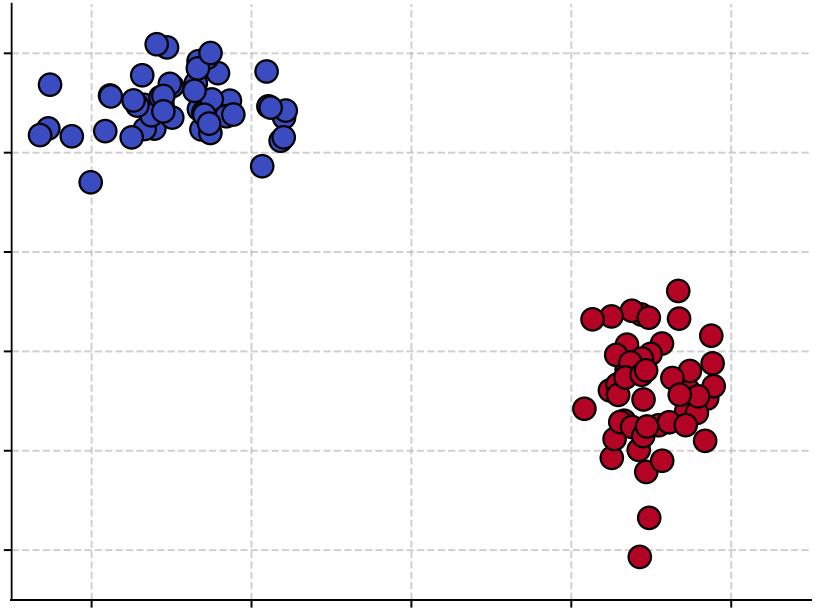


Multilayer Perceptron



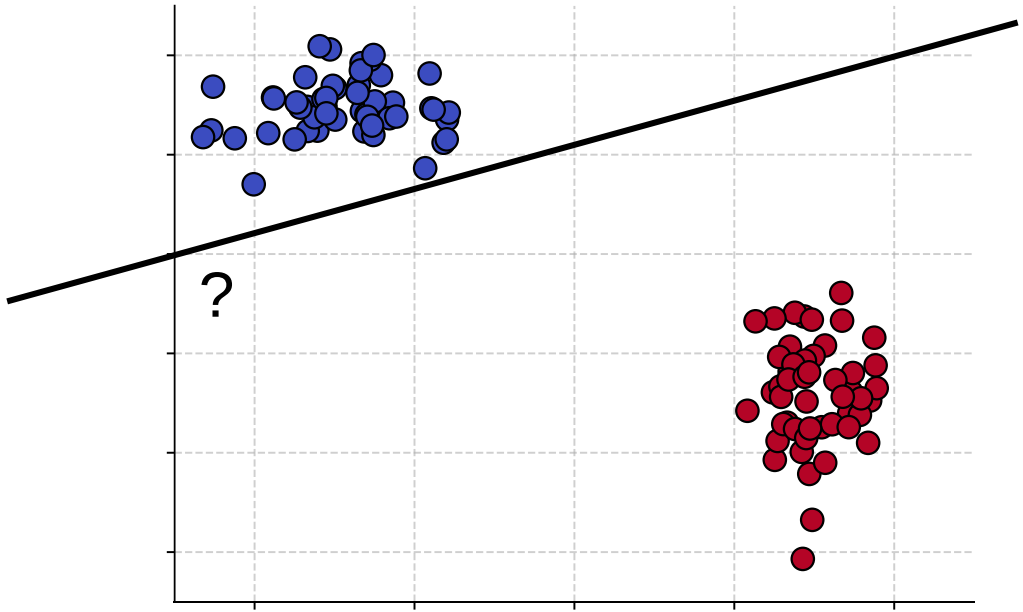
Decision Boundaries

- How would you draw a straight line through these two sets of points?



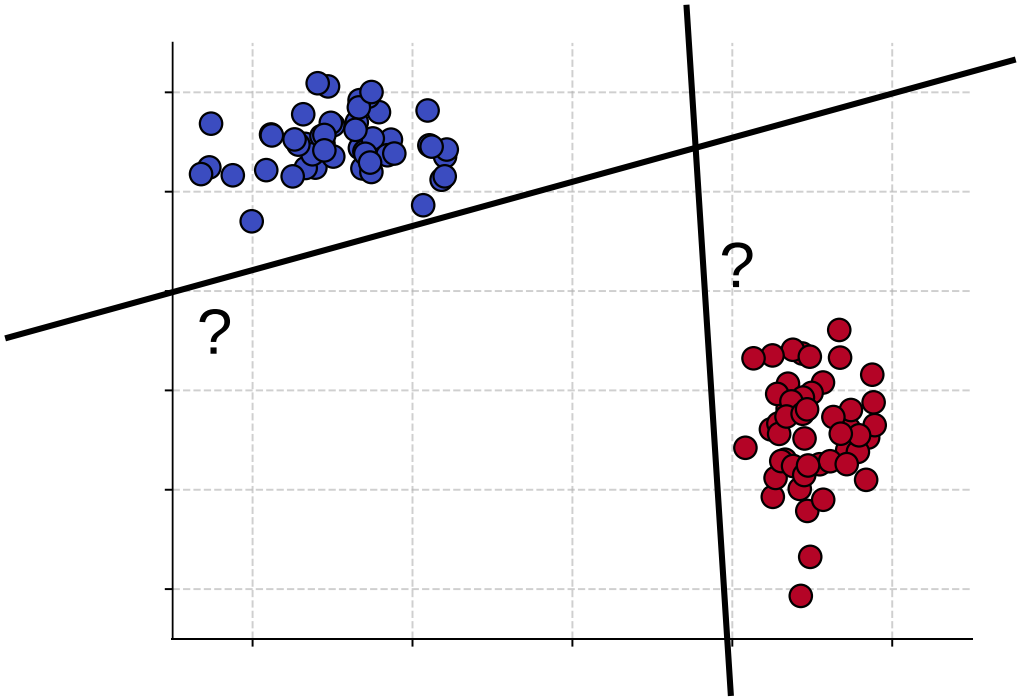
Decision Boundaries

- How would you draw a straight line through these two sets of points?



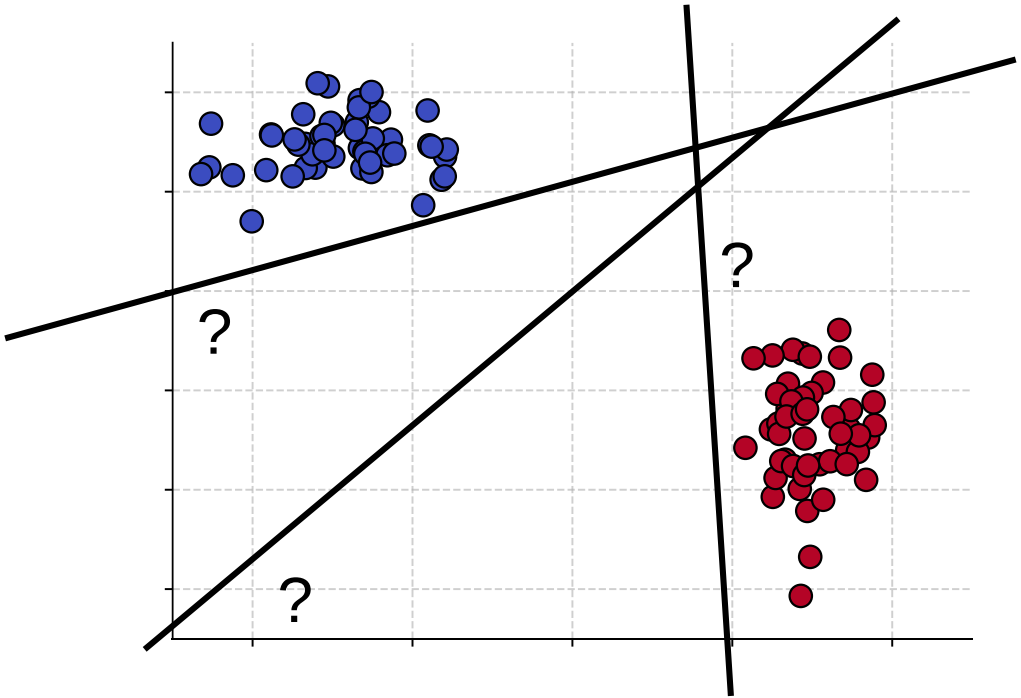
Decision Boundaries

- How would you draw a straight line through these two sets of points?



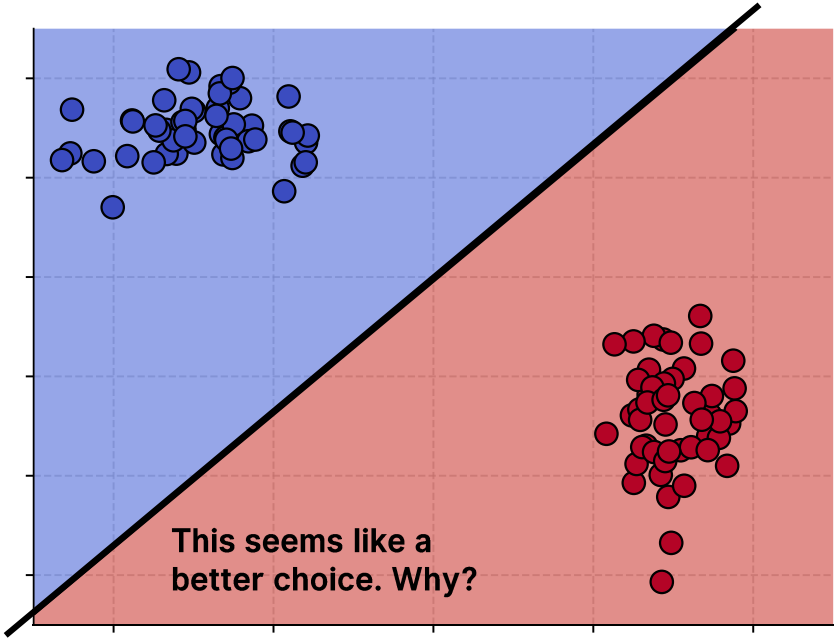
Decision Boundaries

- How would you draw a straight line through these two sets of points?



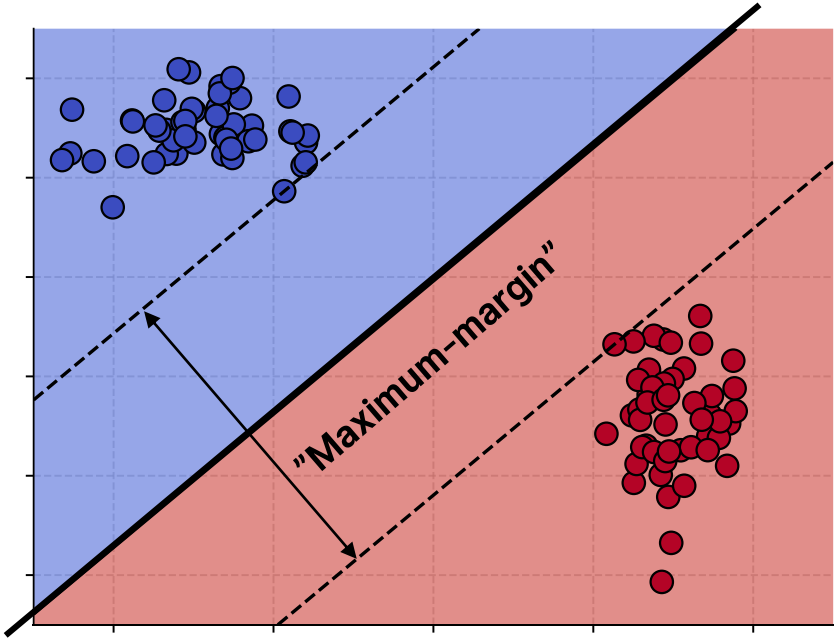
Decision Boundaries

- How would you draw a straight line through these two sets of points?



Decision Boundaries

- How would you draw a straight line through these two sets of points?



SVM Definition

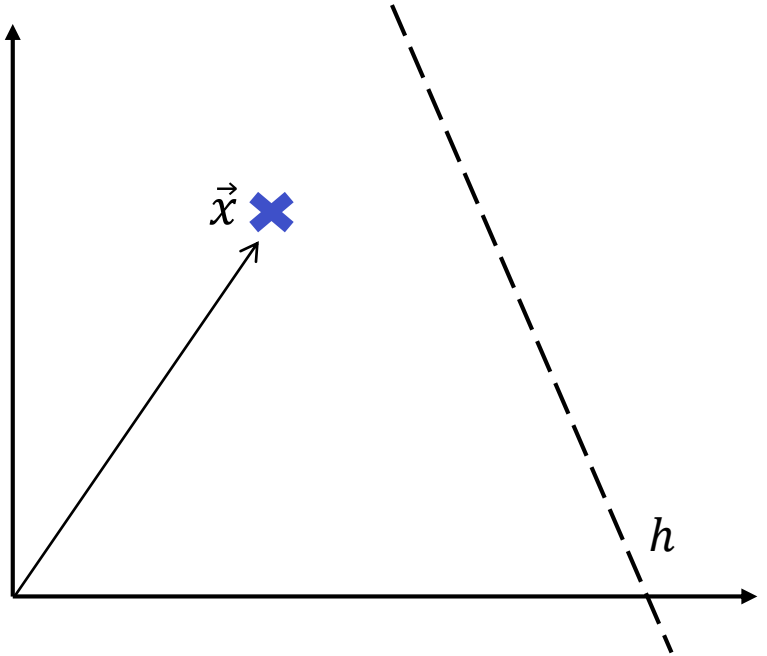
- A **Support Vector Machine** (**SVM**) is a *non-probabilistic binary linear classifier*.
  - **Classifier** – A supervised learning method which predicts a categorical class.
  - **Linear** – The decision boundary is an n-dimensional hyperplane.
  - **Binary** – It can learn to predict one of two classes (a “+” class and a ”-” class).
  - **Non-probabilistic** – The output of its *decision function* is not bounded, so it cannot be interpreted as probability.

There are methods of adapting an SVM to be used for **regression** problems and for making it **non-linear**, **multiclass** and/or **probabilistic**.

- An SVM tries to find a *separating* hyperplane, which is as far away from all training points at the same time (a **maximum-margin hyperplane**)
  - A point is classified as “+” or “-”, depending on which part of the hyperplane it lies.

Decision Rule

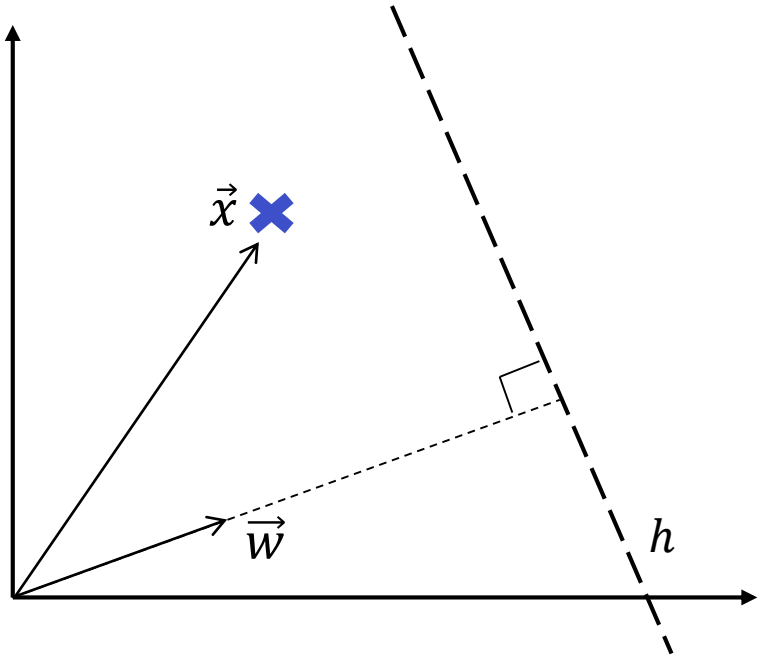
- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?



Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$



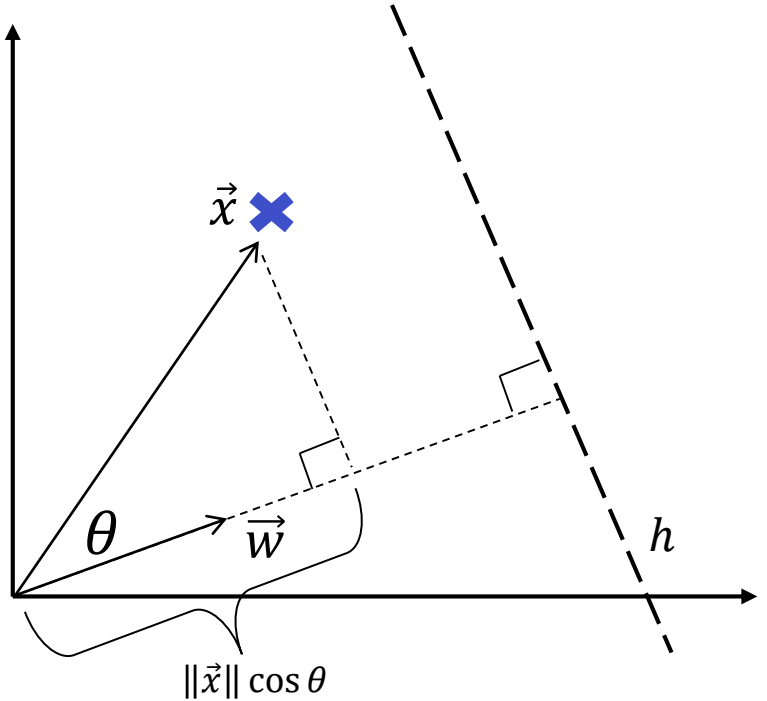
Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$

$$\langle \vec{x}, \vec{w} \rangle = (\|\vec{x}\| \cos\theta)\|\vec{w}\|$$

is proportional to the projection of $\vec{x}$ on $\vec{w}$



Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$

$\langle \vec{x}, \vec{w} \rangle = (\|\vec{x}\| \cos\theta)\|\vec{w}\|$

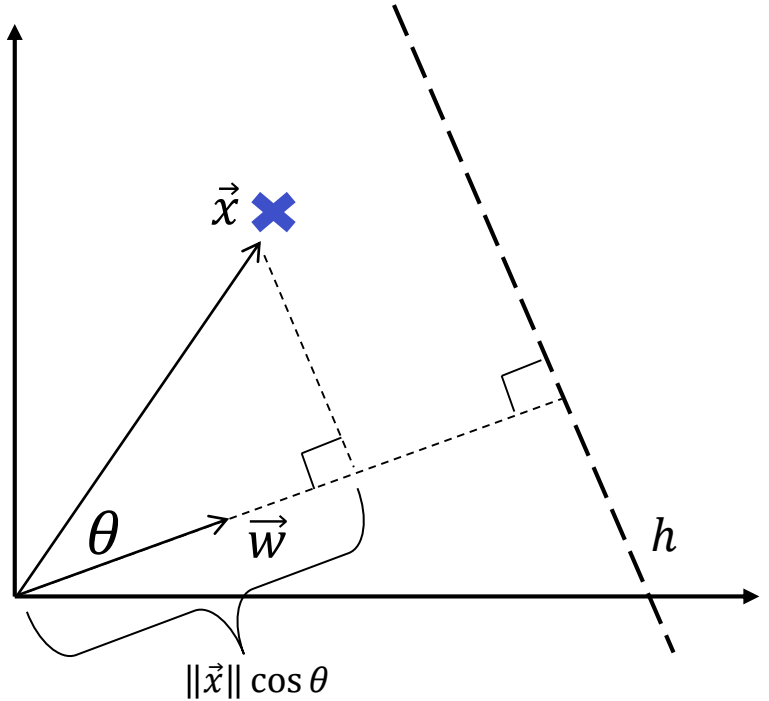
is proportional to the projection of $\vec{x}$ on $\vec{w}$

$\Longrightarrow$

$\exists c \in \mathbb{R}$ such that

if $\langle \vec{x}, \vec{w} \rangle \geq c$ then $\vec{x}$ is on the right side of h

(i.e. $\vec{x}$ is a "+" sample)



Decision Rule

- How do we **decide** if a point $\vec{x}$ is on some side of a hyperplane h?

Let $\vec{w} \perp h$

$\langle \vec{x}, \vec{w} \rangle = (\|\vec{x}\| \cos\theta)\|\vec{w}\|$

is proportional to the projection of $\vec{x}$ on $\vec{w}$

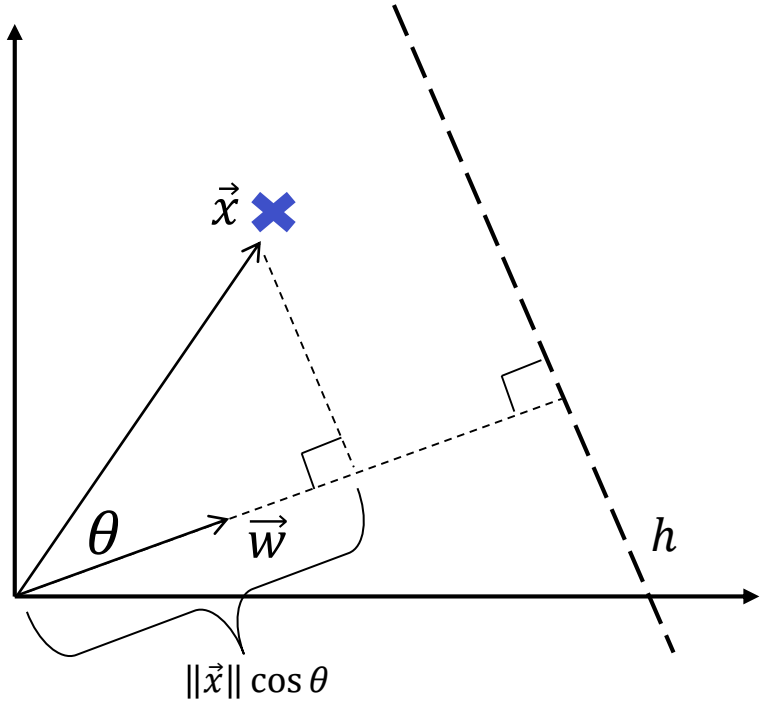
$\Longrightarrow$

$\exists c \in \mathbb{R}$ such that

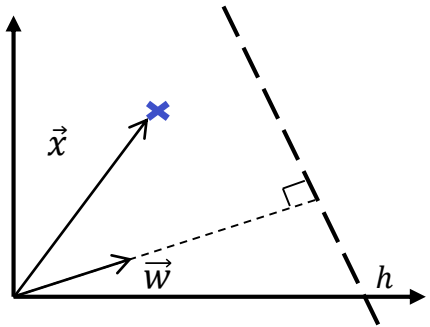
if $\langle \vec{x}, \vec{w} \rangle \geq c$ then $\vec{x}$ is on the right side of h

(i.e. $\vec{x}$ is a "+" sample)

Let $b = -c$



Decision Rule



$\langle \vec{x}, \vec{w} \rangle + b > 0 \Rightarrow \vec{x}$ is on the right side of h

$\langle \vec{x}, \vec{w} \rangle + b < 0 \Rightarrow \vec{x}$ is on the left side of h

$\langle \vec{x}, \vec{w} \rangle + b = 0 \Rightarrow \vec{x}$ is exactly on h

$|\langle \vec{x}, \vec{w} \rangle + b|$ is proportional to the distance from h

In fact, we can say that h is determined by $\vec{w}$ and b

- SVM decision rule:

$\vec{x}$ is a "+" sample if $\langle \vec{x}, \vec{w} \rangle + b \geq 0$

$\vec{x}$ is a "-" sample otherwise

- Which $\vec{w}$? Which b?

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ \mathrm{y}^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples:

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples:

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 0 \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b < 0 \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ \mathrm{y}^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples:

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 0 \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b < 0 \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

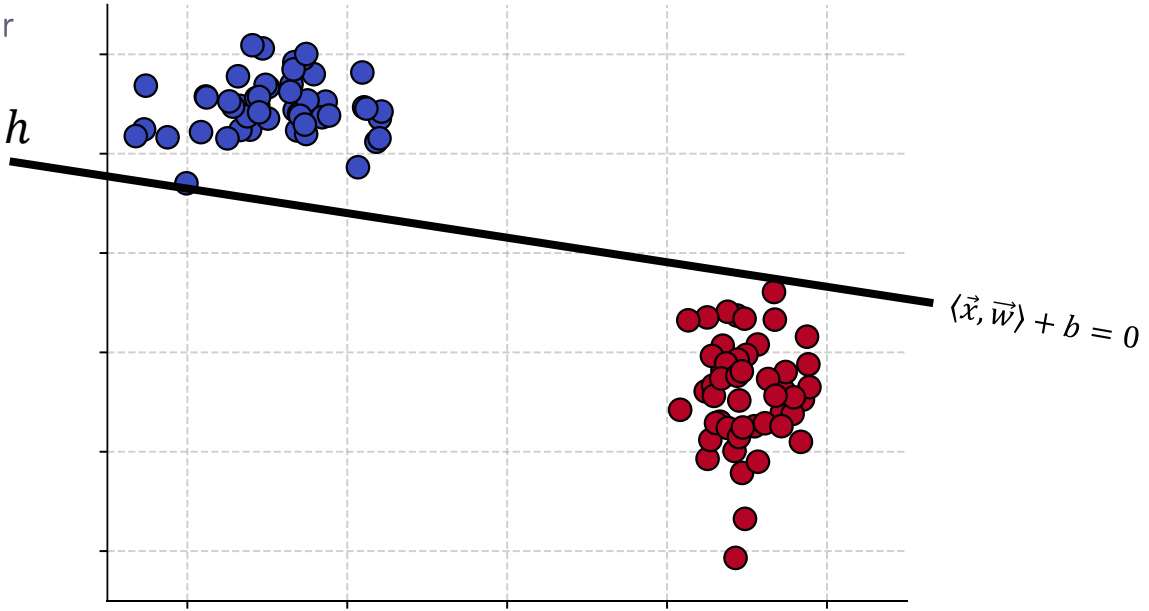
Are these conditions good enough?

Learning a "good" separating hyperplane

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 0 \qquad \forall \overrightarrow{x_+} \in \left\{ \vec{x}^{(i)} \middle| y^{(i)} = +1 \right\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b < 0 \qquad \forall \overrightarrow{x_-} \in \left\{ \vec{x}^{(i)} \middle| y^{(i)} = -1 \right\}$$

This hyperplane satisfies our conditions, but it has *absolutely no margin*.

We want it to have *some margin*, which we can later maximize.



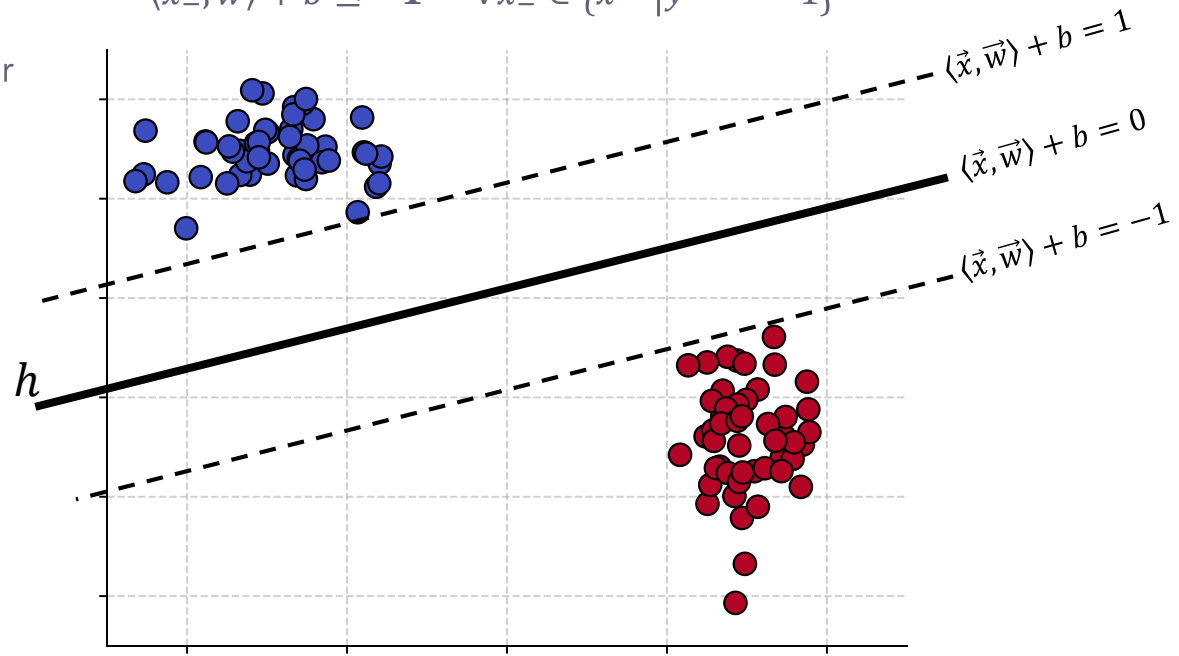
Learning a "good" separating hyperplane

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq 1 \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -1 \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

This hyperplane satisfies our conditions, but it has *absolutely no margin*.

We want it to have *some margin*, which we can later maximize.

We modify the conditions such that we force a **gap** around the *separating hyperplane* in which no training examples lie.



Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples and have **some margin between the classes:**

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq \mathbf{1} \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -\mathbf{1} \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples and have **some margin between the classes:**

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq \mathbf{1} \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -\mathbf{1} \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

- Combining the two inequations:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

Learning a “good” separating hyperplane

- Training set: $E = \{(\vec{x}^{(1)}, y^{(1)}), (\vec{x}^{(2)}, y^{(2)}), \dots, (\vec{x}^{(m)}, y^{(m)})\},\ \vec{x}^{(i)} \in \mathbb{R}^n,\ y^{(i)} \in \{+1, -1\}$
- We want to separate the ”+” and ”-” training samples and have **some margin between the classes:**

$$\langle \overrightarrow{x_+}, \vec{w} \rangle + b \geq \mathbf{1} \qquad \forall \overrightarrow{x_+} \in \{\vec{x}^{(i)} | y^{(i)} = +1\}$$
$$\langle \overrightarrow{x_-}, \vec{w} \rangle + b \leq -\mathbf{1} \qquad \forall \overrightarrow{x_-} \in \{\vec{x}^{(i)} | y^{(i)} = -1\}$$

- Combining the two inequations:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

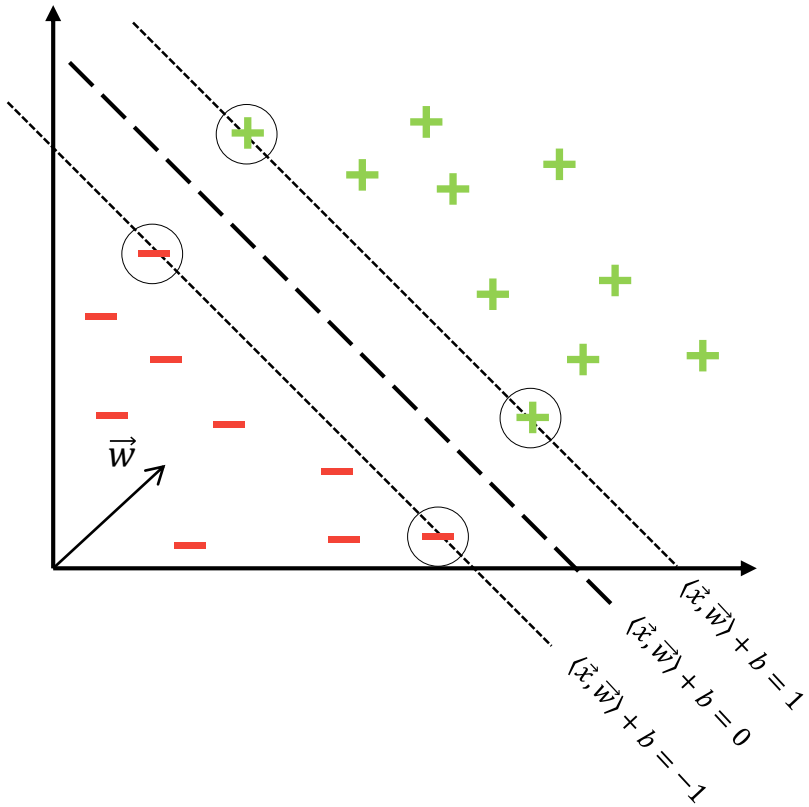
- For examples $(\vec{x}^{(i)}, y^{(i)})$ which lie exactly on the edges of the gap:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 = 0$$

Learning a "good" separating hyperplane

- For training examples $(\vec{x}^{(i)}, y^{(i)})$, which lie exactly on the edges of the gap:

$$y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 = 0$$

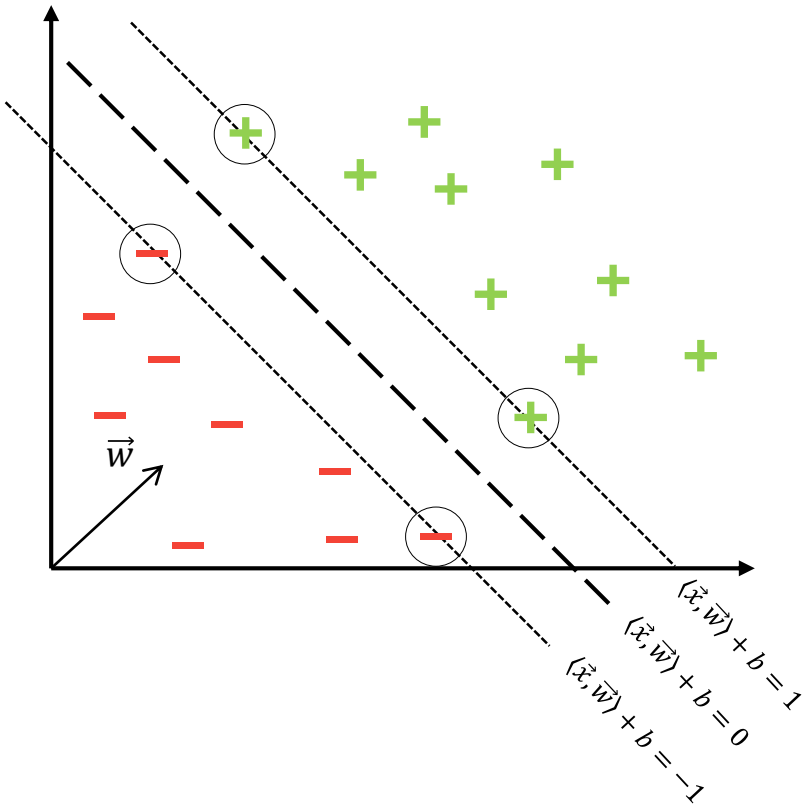


Learning a "good" separating hyperplane

- For training examples $\left(\vec{x}^{(i)}, y^{(i)}\right)$, which lie exactly on the edges of the gap:

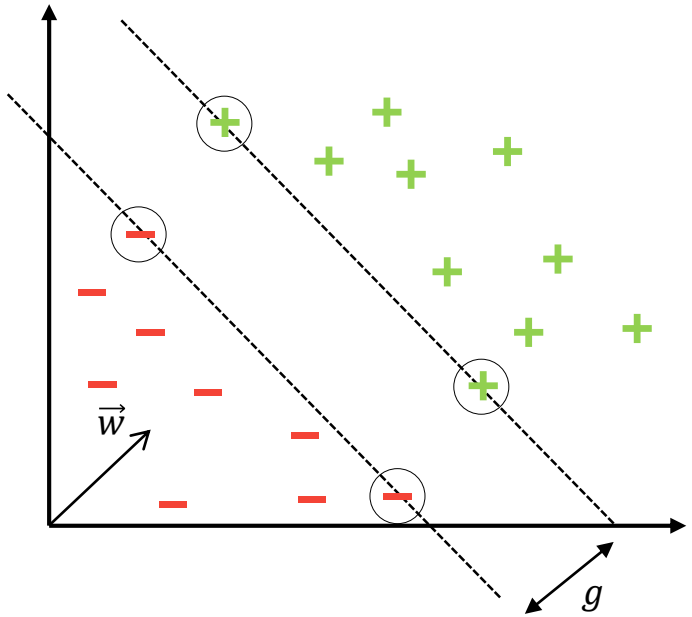
$$\boldsymbol{y^{(i)}\left(\left\langle \vec{x}^{(i)}, \vec{w} \right\rangle + b\right) - 1 = 0}$$

- We call these examples "**Support Vectors**"



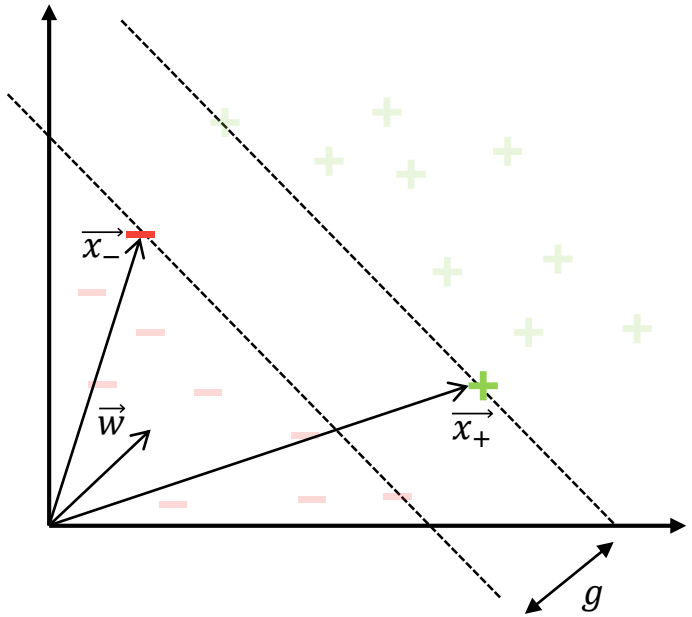
Making the margin “wide”

- How do we express the width of the gap?



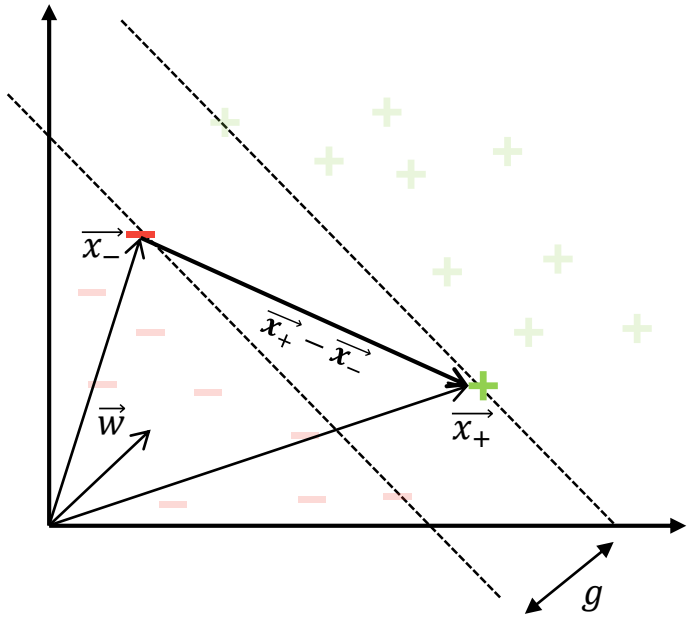
Making the margin “wide”

- How do we express the width of the gap?



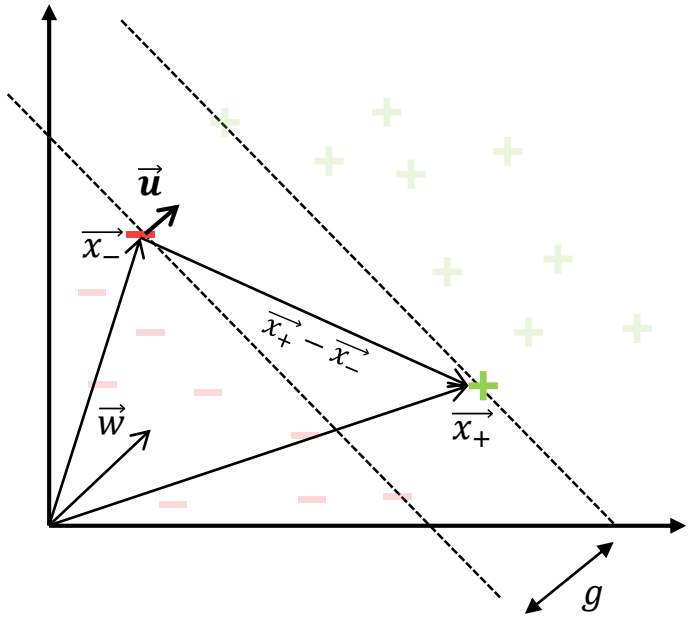
Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.



Making the margin “wide”

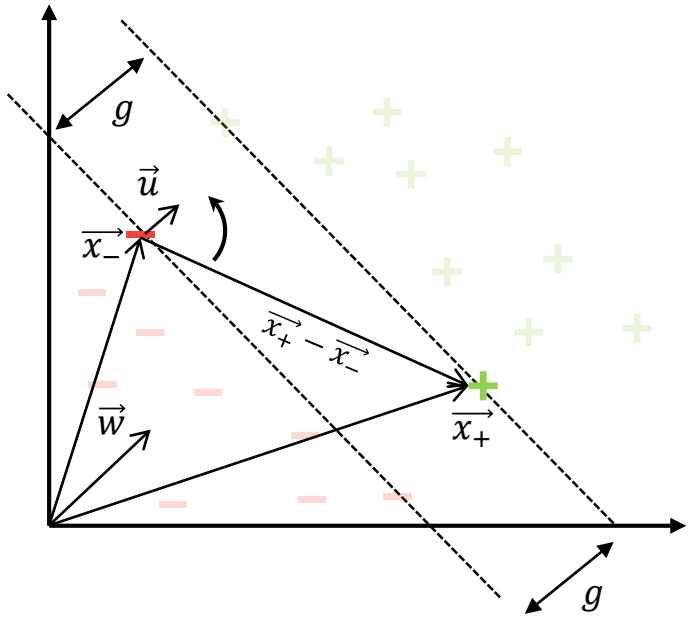
- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector



Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

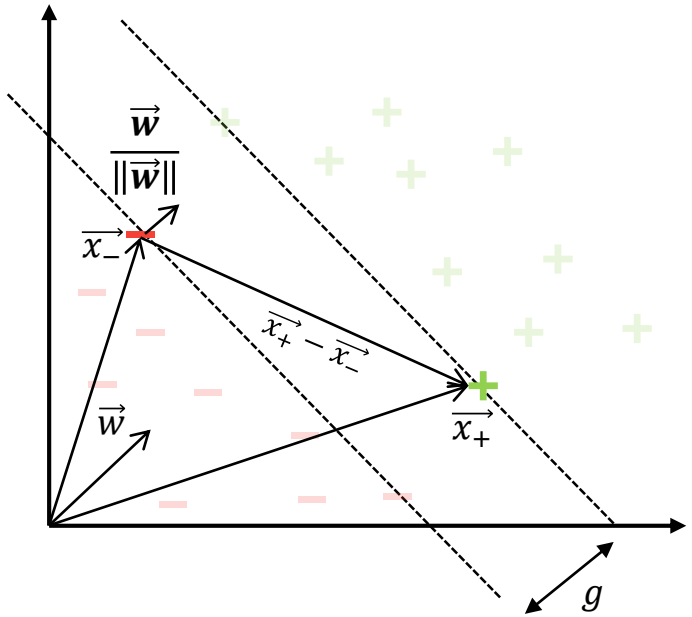
$$g = (\overrightarrow{x_+} - \overrightarrow{x_-}) \cdot \vec{u}$$



Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

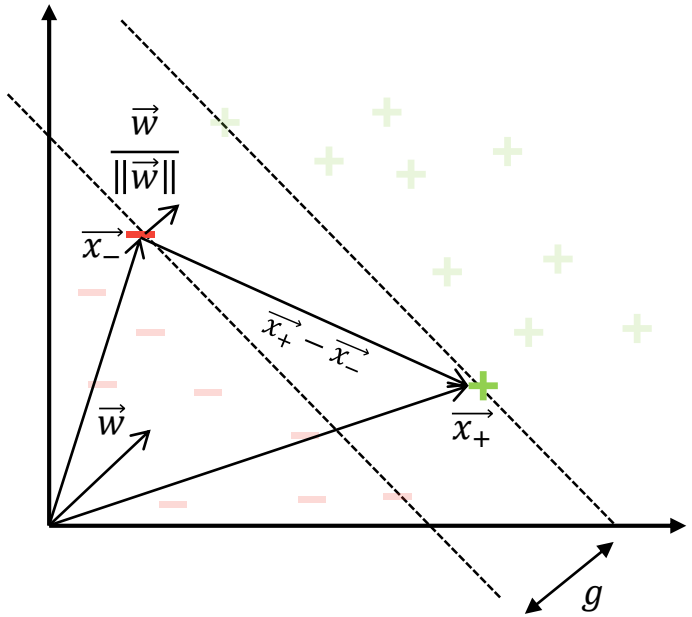
$$g = (\overrightarrow{x_+} - \overrightarrow{x_-}) \cdot \frac{\vec{w}}{\|\vec{w}\|}$$



Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

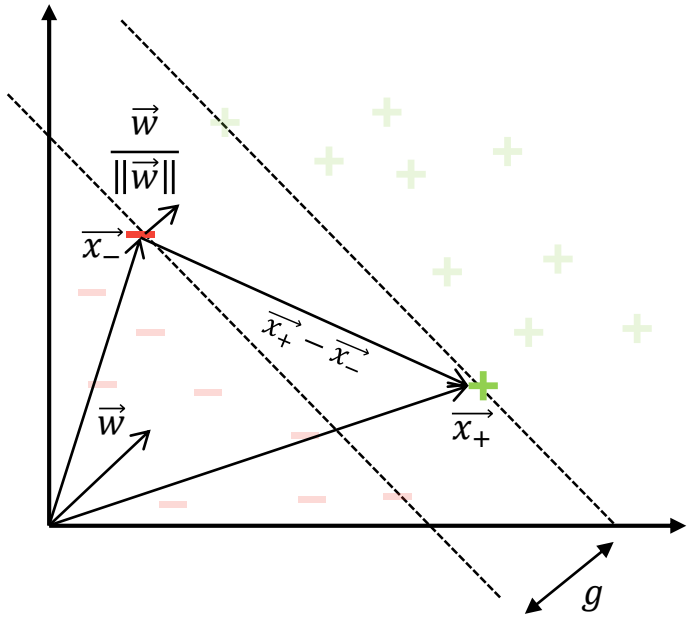


Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

- But $y(\langle \vec{x}, \vec{w} \rangle + b) - 1 = 0$ for $\vec{x}$ on the edge:



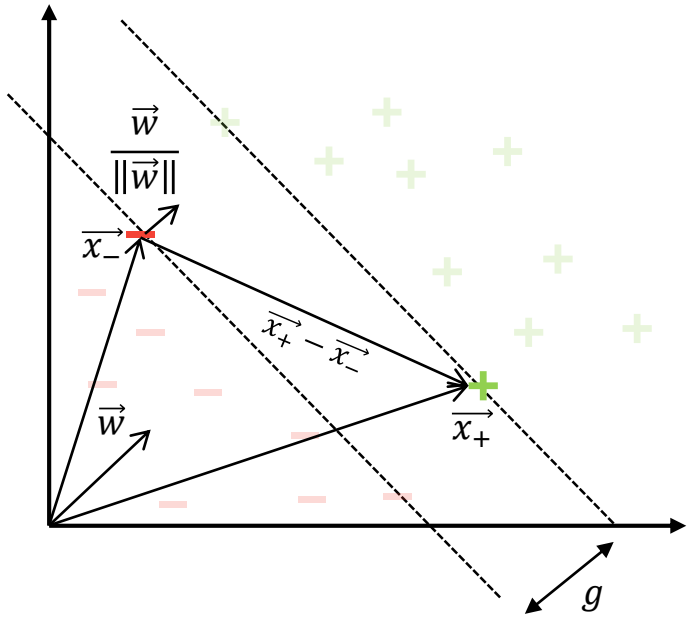
Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

- But $y(\langle \vec{x}, \vec{w} \rangle + b) - 1 = 0$ for $\vec{x}$ on the edge:

$$g = \frac{(1 - b) - (-1 - b)}{\|\vec{w}\|}$$



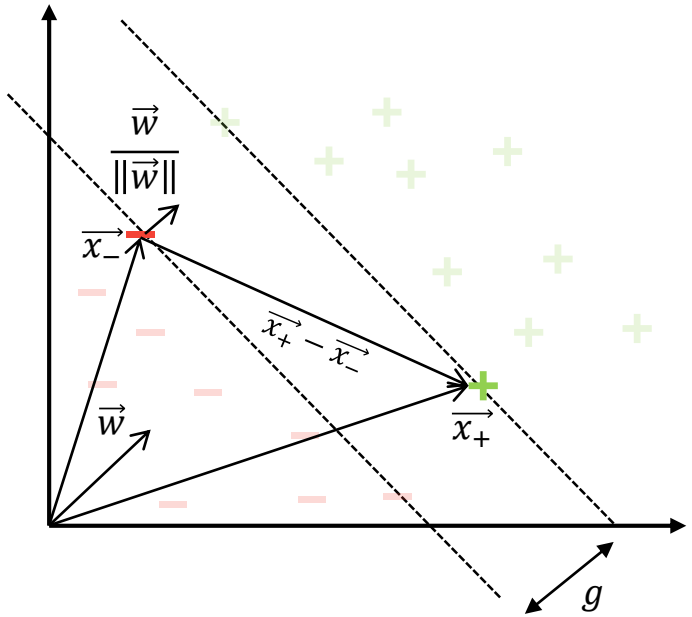
Making the margin “wide”

- How do we express the width of the gap?
- $(\overrightarrow{x_+} - \overrightarrow{x_-})$ is a vector from one side of the gap to the other.
- Let $\vec{u} \perp h$ be a unit vector

$$g = \frac{\langle \overrightarrow{x_+}, \vec{w} \rangle - \langle \overrightarrow{x_-}, \vec{w} \rangle}{\|\vec{w}\|}$$

- But $y(\langle \vec{x}, \vec{w} \rangle + b) - 1 = 0$ for $\vec{x}$ on the edge:

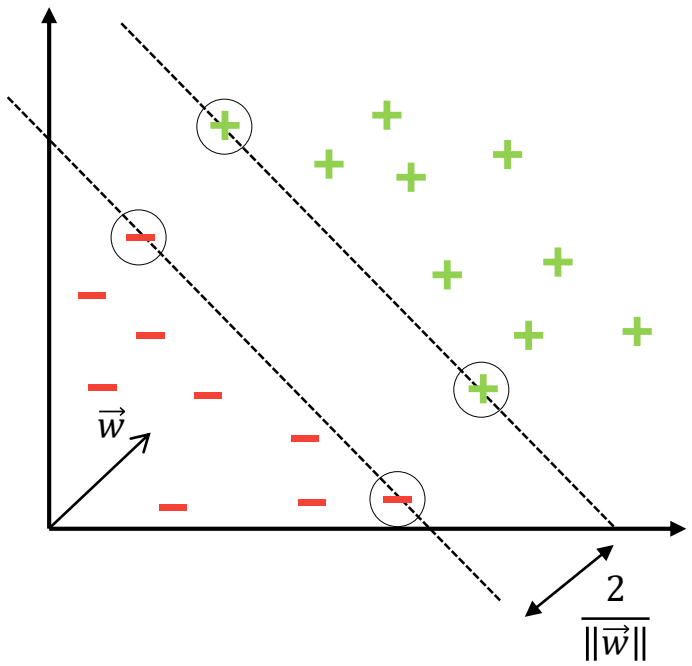
$$\boldsymbol{g = \frac{2}{\|\vec{w}\|}}$$



Making the margin “wide”

- How do we express the width of the gap?

$$g = \frac{2}{\|\vec{w}\|}$$



Making the margin "wide"

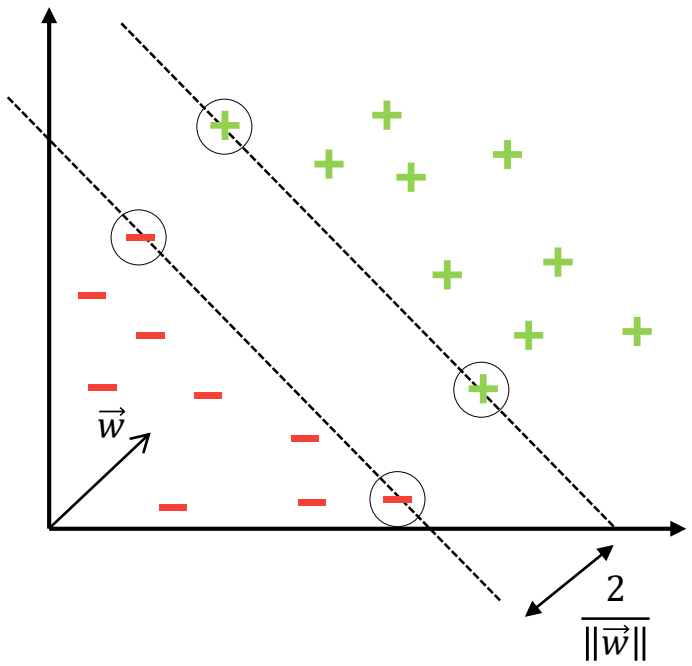
- How do we express the width of the gap?

$$g = \frac{2}{\|\vec{w}\|}$$

- We want to *maximize the gap:*

$$\text{maximize} \frac{2}{\|\vec{w}\|} \Rightarrow \text{minimize } \|\vec{w}\| \Rightarrow$$

$$\textbf{minimize} \frac{\|\vec{w}\|^2}{2}$$



What we have so far

SVM Primal Form

- The decision rule is:

$$\vec{x} \text{ is a “+” sample if } \langle \vec{x}, \vec{w} \rangle + \mathbf{b} \geq \mathbf{0}$$

- In order to obtain $\vec{w}$ and b we need to:

$$\text{minimize } \frac{\|\vec{w}\|^2}{2}$$

$$\text{subject to } y^{(i)}\left(\langle \vec{x}^{(i)}, \vec{w} \rangle + b\right) - 1 \geq 0$$

15.1.2 The Sample Complexity of Hard-SVM

Recall that the VC-dimension of halfspaces in $\mathbb{R}^d$ is $d+1$. It follows that the sample complexity of learning halfspaces grows with the dimensionality of the problem. Furthermore, the fundamental theorem of learning tells us that if the number of examples is significantly smaller than d/ϵ then no algorithm can learn an ϵ-accurate halfspace. This is problematic when d is very large.

To overcome this problem, we will make an additional assumption on the underlying data distribution. In particular, we will define a "separability with margin γ" assumption and will show that if the data is separable with margin γ then the sample complexity is bounded from above by a function of $1/\gamma^2$. It follows that even if the dimensionality is very large (or even infinite), as long as the data adheres to the separability with margin assumption we can still have a small sample complexity. There is no contradiction to the lower bound given in the fundamental theorem of learning because we are now making an additional assumption on the underlying data distribution.