Lecture 6: Linear classification

Machine Learning 2025

Federico Chiariotti (federico.chiariotti@unipd.it)

Università degli Studi di Padova

Lecture plan

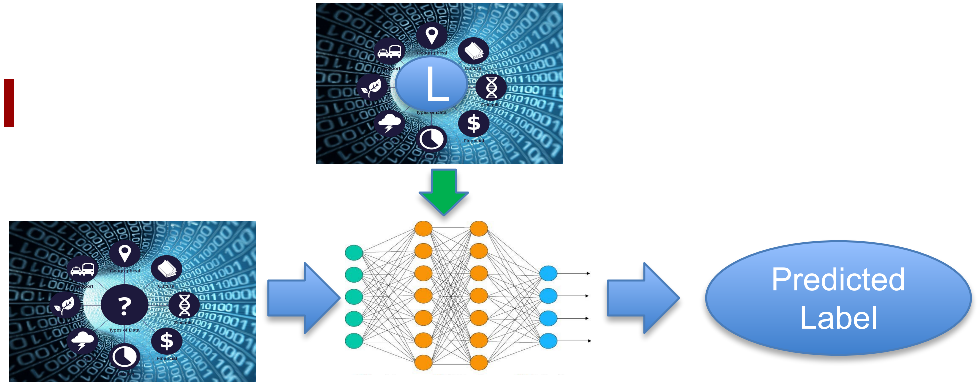
Date	#	Topic	Date	#	Topic	Date	#	Topic
Sep. 30	1	Introduction	Nov. 4	L2	*Model selection*	Nov. 28	12	CNNs
Oct. 7	2	PAC learning	Nov. 7	8	SVMs	???	13	PCA
Oct. 10	3	Uniform convergence	Nov. 11	L3	*Linear models*	Dec. 5	14	Clustering models
Oct. 14	L1	*Python basics*	???	9	Kernels	Dec. 9	L6	*Neural networks*
Oct. 17	4	VC dimension	Nov. 14	10	Ensemble models	Dec. 16	L7	*Clustering*
Oct. 21	5	Model selection	Nov. 18	L4	*SVMs*	Dec. 19	15	Reinforcement
Oct. 24	6	Linear classification	Nov. 21	11	Neural networks	???	L8	*Reinforcement*
Oct. 31	7	Linear regression	Nov. 25	L5	*Random forests*	???	16	Exercises and Q&A

IMPORTANT: no lecture on October 28!

Recap

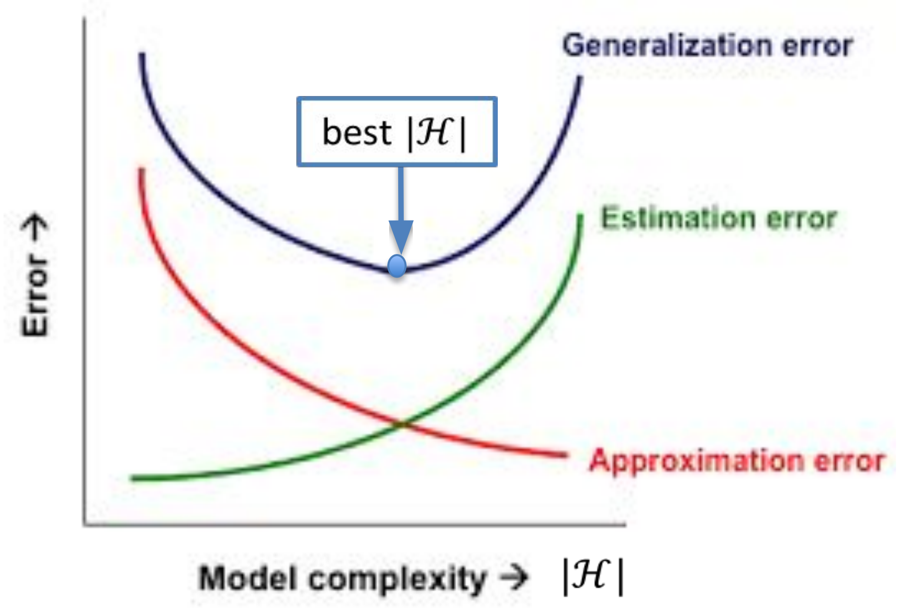
Università degli Studi di Padova

Supervised learning model

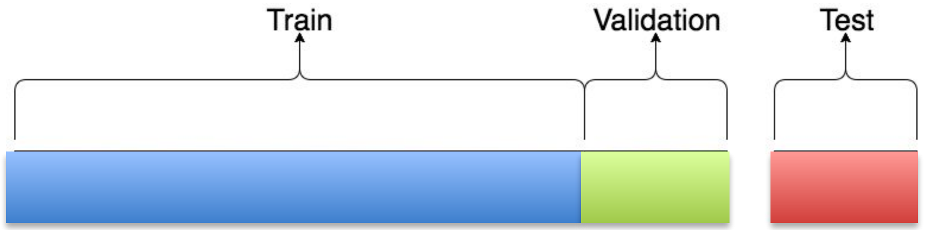


- Prediction rule $h : X \rightarrow Y$
  - The learner's output (hypothesis)
  - $A(S)$: hypothesis produced by algorithm A when it is fed training set S
- Data generation model
  - D is a distribution over X **(unknown to the machine learning algorithm)**
  - Instances are labeled according to $f : X \rightarrow Y$ **(unknown to the ML algorithm)**
  - Training set: sampling according to $D, y_i = f(x_i) \ \forall x_i \in S$
- Success metric
  - Classifier error: probability of predicting the wrong label over D

The bias-complexity trade-off



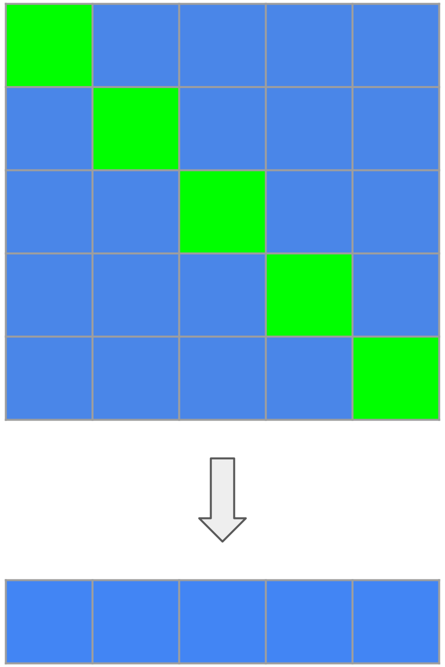
Test and validation sets



We divide our data in 3 parts:

➔ The training set is used to choose the best hypothesis with ERM
➔ The validation set is used to choose the best hyperparameters or subclass
➔ The test set is used to estimate the true loss

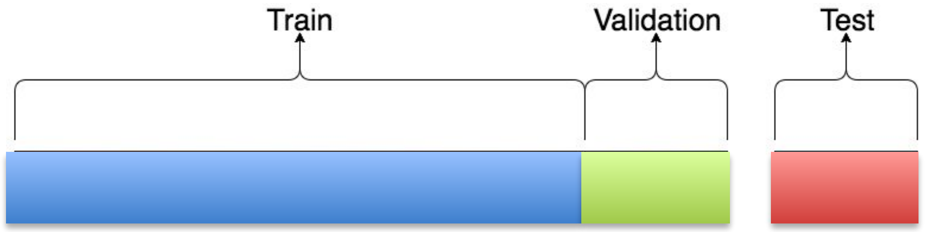
K-fold Cross Validation



Actually, we can reuse the validation set:

1. Split the training set in K *folds*
2. Perform validation using K-1 as the training set and the other one as the validation set
3. The **score** of each subclass is the average validation set
4. Choose the subclass with the highest score
5. Retrain on the whole training set

Test set contamination



The test set should be drawn from D, but it should be independent from the training (i.e., should not be used in any way, or contain the same samples)

If we use the test set to make decisions, we are overfitting on it: the only way to use it to estimate $L_D(h)$ without cheating is to **only use it at the end**. If the test set is contaminated (i.e., we use it during the learning process in any way), we need a new test set!

Part 1:
Linear models

Università degli Studi di Padova

Affine functions

Class of affine functions:

$$L_d = \left\{ h_{\mathbf{w},b}, \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R} \right\}$$

Each function h is:

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = b + \sum_{i=1}^{d} w_i x_i$$

Affine functions are linear functions with a sum. There are d+1 parameters (d weights, plus the bias)

Linear hypotheses

Binary classification:

Sign of the affine function

$$h_{\mathbf{w},b}(\mathbf{x}) = \text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b)$$

Regression:

Direct affine function

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b$$

A 2D example

Separating (hyper)plane:

$$w_1 x_1 + w_2 x_2 + b = 0$$

Plane definition:

$$x_2 = m x_1 + h$$

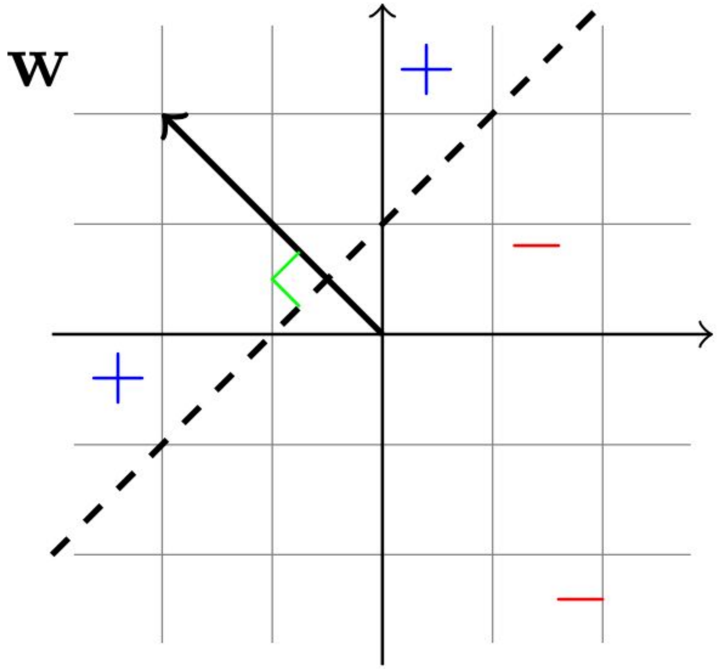
Intercept:

$$h = -\frac{b}{w_2}$$

Slope:

$$m = \frac{w_1}{w_2}$$

The weight vector is orthogonal to the hyperplane



Homogeneous coordinates

We can incorporate the bias into the weights by adding one dimension

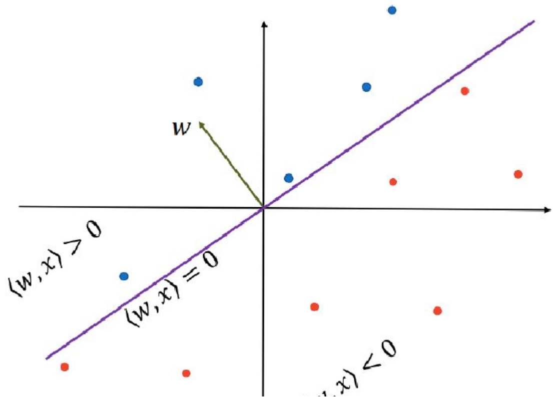
$$\mathbf{x}' = (1, x_1, x_2, \ldots, x_d) \quad \mathbf{w}' = (b, w_1, w_2, \ldots, w_d)$$

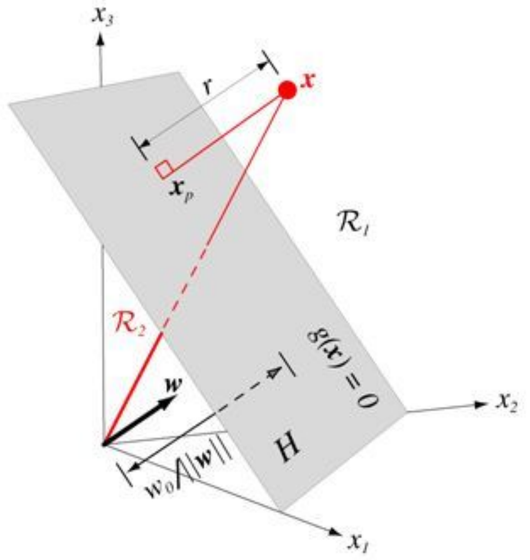
In this way, the affine function becomes a linear function

$$h_{\mathbf{w},b}(\mathbf{x}) = b + \sum_{i=1}^{d} w_i x_i \qquad h_{\mathbf{w}'}(\mathbf{x}') = 1 \times b + \sum_{i=1}^{d} w_i x_i$$

The result is exactly the same, but the notation is much more compact

Classification: halfspaces


w
$\langle w, x\rangle > 0$
$\langle w, x\rangle = 0$
$\langle w, x\rangle < 0$



Domain: $X \subset \mathbb{R}^d$

Labels: $Y = \{-1, 1\}$

Loss: 0-1

Hypothesis class: $H = \text{sign} \circ L_d$ $\qquad h_{\mathbf{w}'}(\mathbf{x}') = \text{sign}(\langle \mathbf{w}', \mathbf{x}' \rangle)$

2D: a line divides the two regions (d=2, d'=3)

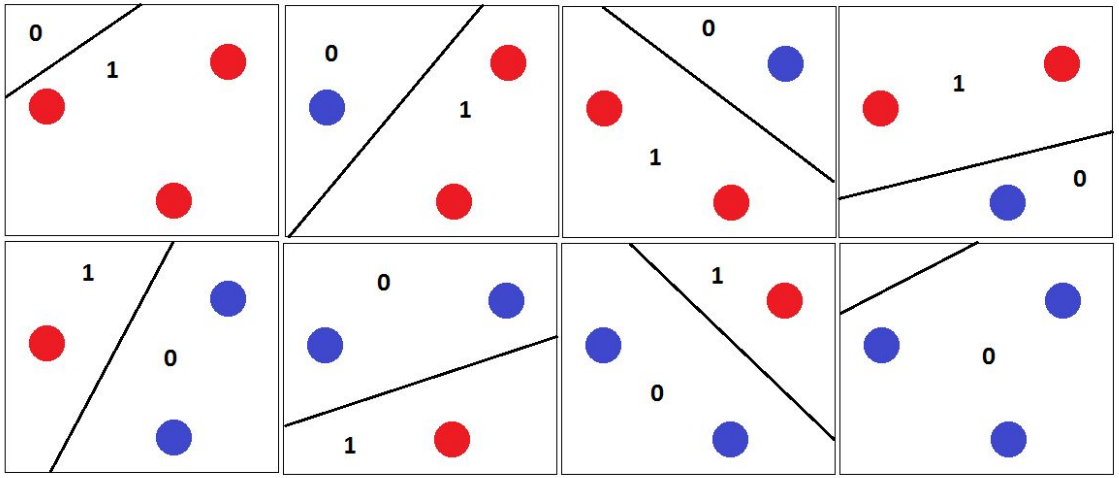
3D: a plane divides the two regions (d=3, d'=4)

The VC dimension of halfspaces

There are uncountably infinite possible halfspaces even in 2 dimensions (all possible straight lines on a plane)

The VC dimension of the halfspace hypothesis class in d dimensions is d+1, considering non-homogeneous coordinates

VC dimension in R^2



We can shatter any set of size 3 in 2 dimensions (we have all possible examples in the figure)

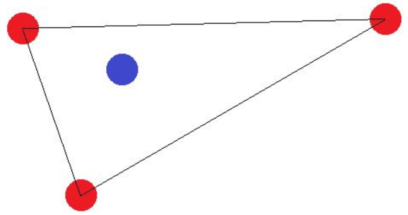
VC dimension in R^2

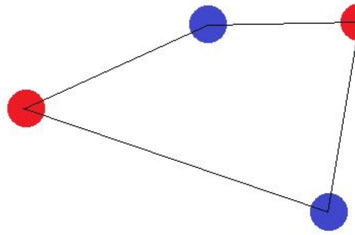
1. Point D is inside triangle ABC: 0001 is impossible

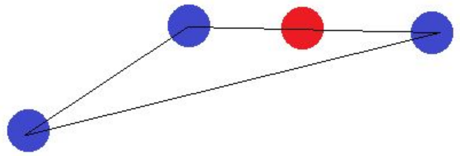
2. ABCD is a convex polygon: 0101 (on the two diagonals) is impossible

3. D is on the AB line: 0001 is impossible

No set of size 4 can be shattered: **VC=3**







Proof: first step (homogeneous case)

We show that VC≥d in the homogeneous case:

Let us consider the base $\mathrm{e}_1, \ldots, \mathrm{e}_d$, with $\mathrm{e}_i = (0, 0, \ldots, 0, 1, 0, \ldots, 0)$, non-zero only in i

If we set $w_i = y_i$, the set is always shattered: all vectors are orthogonal, and only the product of the i-th dimension with the i-th label is non-zero

Proof: second step (homogeneous case)

We then need to show that VC<d+1 in the homogeneous case:

If we have a set of d+1 vectors in d dimensions, they must be linearly dependent:

$$\exists \mathbf{a} \neq 0 : \sum_{i=1}^{d+1} a_i \mathbf{x}_i = 0$$

We can then split the indices in two sets:

$$I = \{i : a_i \geq 0\} \qquad J = \{j : a_j < 0\}$$

Proof: third step (homogeneous case)

We split the indices in two sets:

$$I = \{i : a_i \geq 0\} \qquad J = \{j : a_j < 0\}$$

If neither is empty, we need to keep the sum equal to 0, so we get

$$\sum_{i \in I} a_i \mathbf{x}_i = \sum_{j \in J} |a_j| \mathbf{x}_j$$

We can proceed *ad absurdum*: if the set can be shattered, we can create $\mathbf{w}$ such that

$$\langle \mathbf{w}, \mathbf{x}_j \rangle < 0 \; \forall j \in J \qquad \langle \mathbf{w}, \mathbf{x}_i \rangle > 0 \; \forall i \in I$$

Proof: final step (homogeneous case)

$$\langle \mathbf{w}, \mathbf{x}_j \rangle < 0 \; \forall j \in J \qquad\qquad \langle \mathbf{w}, \mathbf{x}_i \rangle > 0 \; \forall i \in I$$

We then multiply by a:

$$\sum_{i\in I} a_i \langle \mathbf{w}, \mathbf{x}_i \rangle = \sum_{i\in I} \langle a_i \mathbf{x}_i, \mathbf{w} \rangle > 0$$

$$\sum_{j\in J} |a_j| \langle \mathbf{w}, \mathbf{x}_j \rangle = \sum_{j\in J} \langle |a_j| \mathbf{x}_j, \mathbf{w} \rangle < 0$$

There is a contradiction with $\sum_{i\in I} a_i \mathbf{x}_i = \sum_{j\in J} |a_j| \mathbf{x}_j$

The contradiction proves that the VC dimension is d (in the homogeneous case), as no set of size d+1 can be shattered

Part 2: Linear classification

UNIVERSITÀ DEGLI STUDI DI PADOVA

Linear programming as ERM

Linear programming (LP) is an optimization framework that can be solved efficiently. We can pose the ERM rule as a constraint, using a dummy objective

$$\text{maximize } 1$$
$$\text{such that } \mathbf{A}\mathbf{w} \geq (1, \ldots, 1)^T$$

Each row of matrix **A** is simply given by $y_i \mathbf{x}_i$

There are optimized solvers for LP (e.g., simplex), but we don't need to implement them here and now

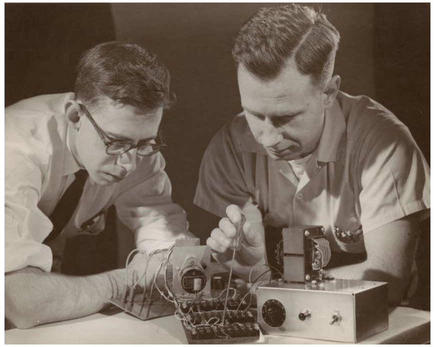
The perceptron

1958 (Rosenblatt): first iterative algorithm to find the separating hyperplane

At each step, the perceptron focuses on **a single** misclassified sample and pushes the boundary towards it

Always converges to ERM if we have realizability

REMEMBER: the perceptron cannot train on all samples at once!



The perceptron algorithm

Start: $\mathbf{w}^{(0)} = (0, \ldots, 0)$

Repeat until convergence:

1. Find a sample with $\text{sign}(\langle \mathbf{w}^{(i)}, \mathbf{x}_m \rangle) = -y_m$
2. Compute the new $\mathbf{w}^{(i+1)} = \mathbf{w}^{(i)} + \alpha \mathbf{x}_m y_m$

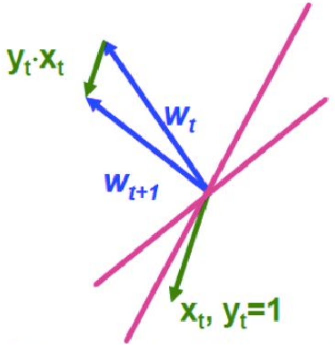
Parameter α is the learning rate (if omitted, it is set to 1)

Termination depends on realizability: if the training set is not linearly separable, the perceptron will go on forever

Pseudocode

```
Input: training set $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$
initialize $\mathbf{w}^{(1)} = (0, \ldots, 0)$;
for $t = 1, 2, \ldots$ do
    if $\exists i$ s.t. $y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle \leq 0$ then $\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + y_i \mathbf{x}_i$;
    else return $\mathbf{w}^{(t)}$;
```

Interpretation of update:



Note that:

$$\begin{aligned} y_i \langle \mathbf{w}^{(t+1)}, \mathbf{x}_i \rangle &= y_i \langle \mathbf{w}^{(t)} + y_i \mathbf{x}_i, \mathbf{x}_i \rangle \\ &= y_i \langle \mathbf{w}^{(t)}, \mathbf{x}_i \rangle + ||\mathbf{x}_i||^2 \end{aligned}$$

$\Rightarrow$ update guides $\mathbf{w}$ to be "more correct" on $(\mathbf{x}_i, y_i)$.

Perceptron convergence

If the training set is linearly separable, the perceptron will converge after fewer than $(RB)^2$ iterations, converging to an ERM rule, with

$$R = \max(||\mathbf{x}_i||)$$

$$B = \min\{||\mathbf{w}|| : y_i\langle\mathbf{w}, \mathbf{x}_i\rangle \geq 1\}\forall i$$

Proof: first step

It is simple to prove that the perceptron will stop at an ERM rule, as it only stops when no samples are misclassified.

We define a vector that satisfies ERM with the minimum weight:

$$\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}:\mathbf{A}\mathbf{w}\geq 1}||\mathbf{w}||$$

We need to bound the iteration T at which we reach convergence.

Proof: second step

$$\langle \mathbf{w}^*, \mathbf{w}_t \rangle - \langle \mathbf{w}^*, \mathbf{w}_{t-1} \rangle = \langle \mathbf{w}^*, \mathbf{w}_t - \mathbf{w}_{t-1} \rangle$$

During the t-th iteration, the weights are changed by a misclassified sample:

$$\mathbf{w}_t - \mathbf{w}_{t-1} = y_t \mathbf{x}_t$$

We then have

$$\langle \mathbf{w}^*, \mathbf{w}_t \rangle - \langle \mathbf{w}^*, \mathbf{w}_{t-1} \rangle = \langle \mathbf{w}^*, y_t \mathbf{x}_t \rangle$$

We can take y out of the inner product:

$$\langle \mathbf{w}^*, \mathbf{w}_t \rangle - \langle \mathbf{w}^*, \mathbf{w}_{t-1} \rangle = y_t \langle \mathbf{w}^*, \mathbf{x}_t \rangle$$

Proof: third step

$$\langle \mathbf{w}^*, \mathbf{w}_t \rangle - \langle \mathbf{w}^*, \mathbf{w}_{t-1} \rangle = y_t \langle \mathbf{w}^*, \mathbf{x}_t \rangle$$

By the definition of ERM, we have

$$y_i \langle \mathbf{w}^*, \mathbf{x}_i \rangle \geq 1 \ \forall i$$

At every step, the inner product between w* and w_t increases by at least 1:

$$\langle \mathbf{w}^*, \mathbf{w}_T \rangle \geq T$$

Proof: fourth step

We want to find a bound for $||\mathbf{w}_T||$. We have

$$||\mathbf{w}_t||^2 = ||\mathbf{w}_{t-1} + y_t\mathbf{x}_t||^2$$

We expand the square:

$$||\mathbf{w}_t||^2 = ||\mathbf{w}_{t-1}||^2 + 2y_t||\langle\mathbf{w}_{t-1}, \mathbf{x}_t\rangle||^2 + ||\mathbf{x}||^2$$

By definition, R is at least as large as the norm of any input vector:

$$||\mathbf{w}_t||^2 \leq ||\mathbf{w}_{t-1}||^2 + 2y_t\langle\mathbf{w}_{t-1}, \mathbf{x}_t\rangle + R^2$$

Since we have a misclassified sample, the product term is negative:

$$||\mathbf{w}_t||^2 \leq ||\mathbf{w}_{t-1}||^2 + R^2$$

Proof: fifth step

After T iterations, we have $||\mathbf{w}_T||^2 \leq TR^2$ and $\langle \mathbf{w}^*, \mathbf{w}_T \rangle \geq T$

By definition, $||\mathbf{w}^*|| = B$ $\left(B = \min\{||\mathbf{w}|| : y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1\} \forall i\right)$

We know that $||\mathbf{w}_T|| \, ||\mathbf{w}^*|| \leq \sqrt{T} RB$

Proof: final step

$$||\mathbf{w}_T||\,||\mathbf{w}^*|| \leq \sqrt{T}RB \qquad\qquad \langle \mathbf{w}^*, \mathbf{w}_T \rangle \geq T$$

We can apply the Cauchy-Schwarz inequality:

$$\langle \mathbf{w}_1, \mathbf{w}_2 \rangle \leq ||\mathbf{w}_1||\,||\mathbf{w}_2||$$

We get

$$T \leq \langle \mathbf{w}^*, \mathbf{w}_T \rangle \leq \sqrt{T}RB$$

Simplifying, we have

$$\sqrt{T} \leq RB$$

This is equivalent to the theorem statement (just square both sides)

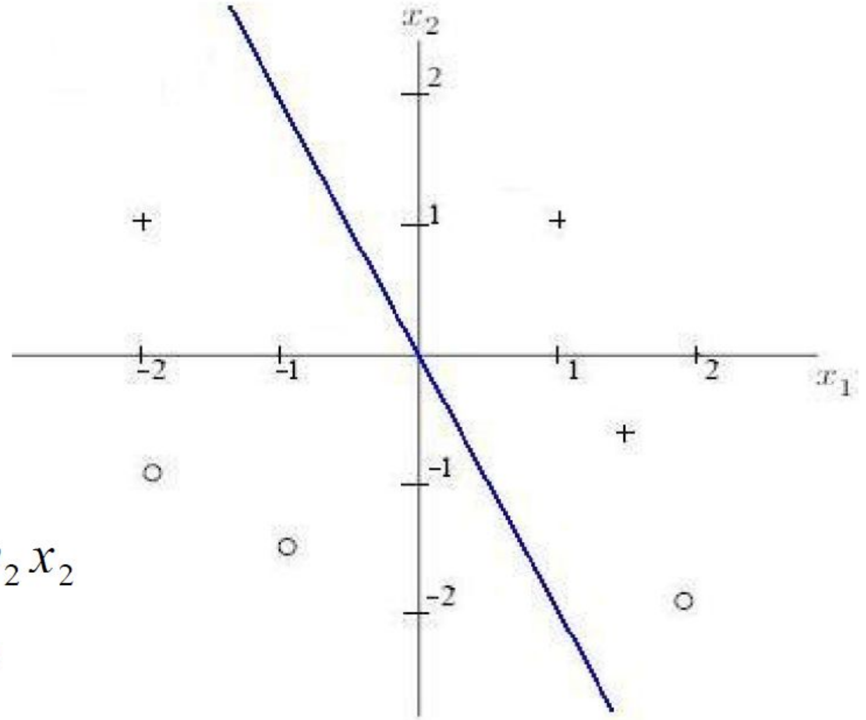
Perceptron: graphical example

Initial Values:

$\eta = 0.2$

$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$$\begin{aligned} 0 &= w_0 + w_1 x_1 + w_2 x_2 \\ &= 0 + x_1 + 0.5x_2 \\ \Rightarrow \quad x_2 &= -2x_1 \end{aligned}$$



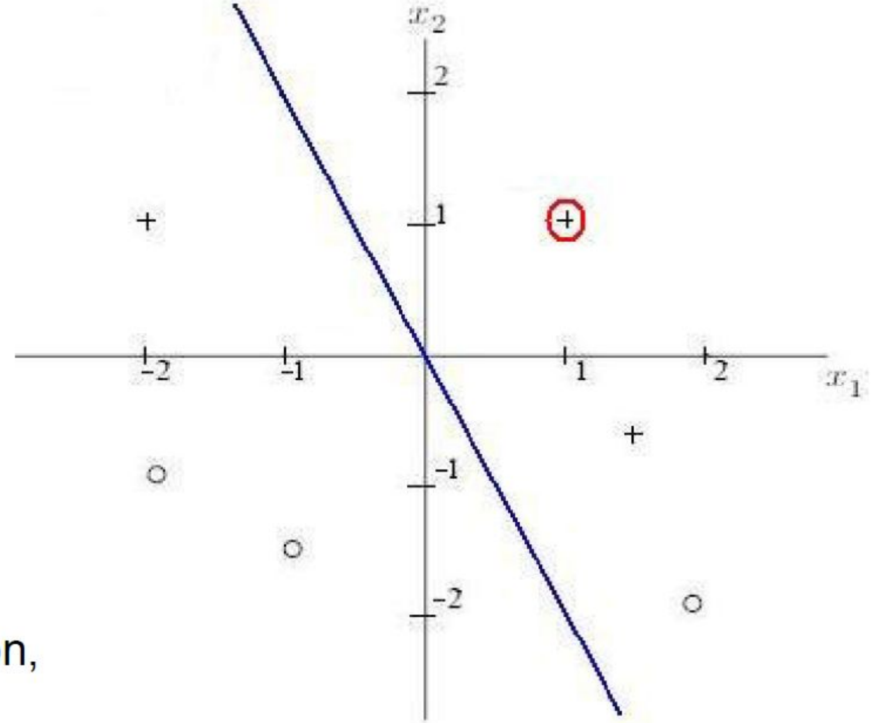
Perceptron: graphical example

$\eta = 0.2$

$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$x_1 = 1$, $x_2 = 1$
$w^T x > 0$

Correct classification, no action



Perceptron: graphical example

$\eta = 0.2$

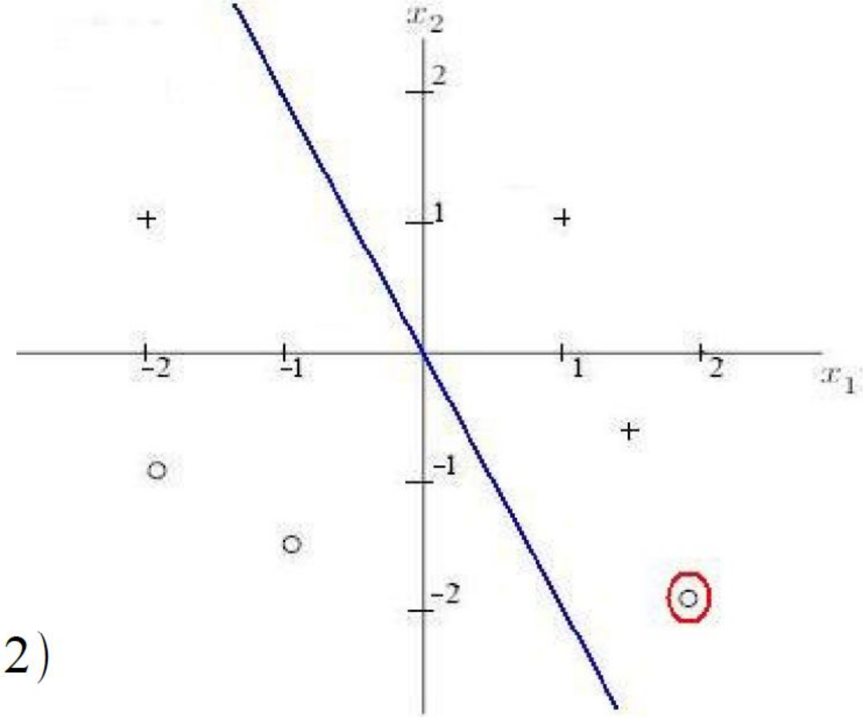
$$w = \begin{pmatrix} 0 \\ 1 \\ 0.5 \end{pmatrix}$$

$x_1 = 2, x_2 = -2$

$$w_0 = w_0 - 0.2 * 1$$

$$w_1 = w_1 - 0.2 * 2$$

$$w_2 = w_2 - 0.2 * (-2)$$



$$\boxed{\text{new } \mathbf{w} = \begin{bmatrix} -0.2 \\ 0.6 \\ 0.9 \end{bmatrix}}$$

Perceptron: graphical example

$\eta = 0.2$

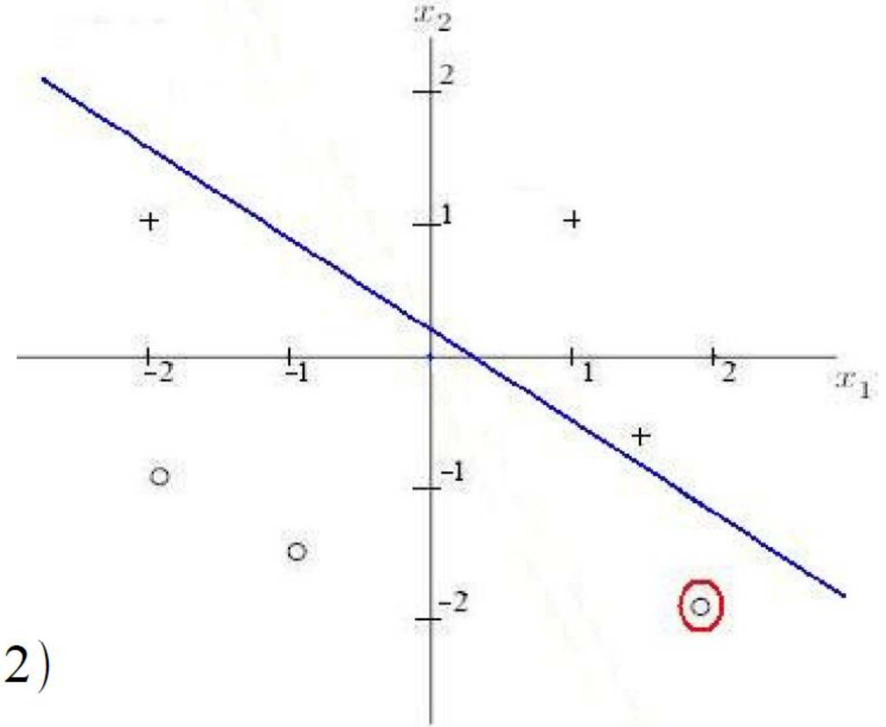
$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$x_1 = 2,\ x_2 = -2$

$$w_0 = w_0 - 0.2 * 1$$

$$w_1 = w_1 - 0.2 * 2$$

$$w_2 = w_2 - 0.2 * (-2)$$



Perceptron: graphical example

$\eta = 0.2$

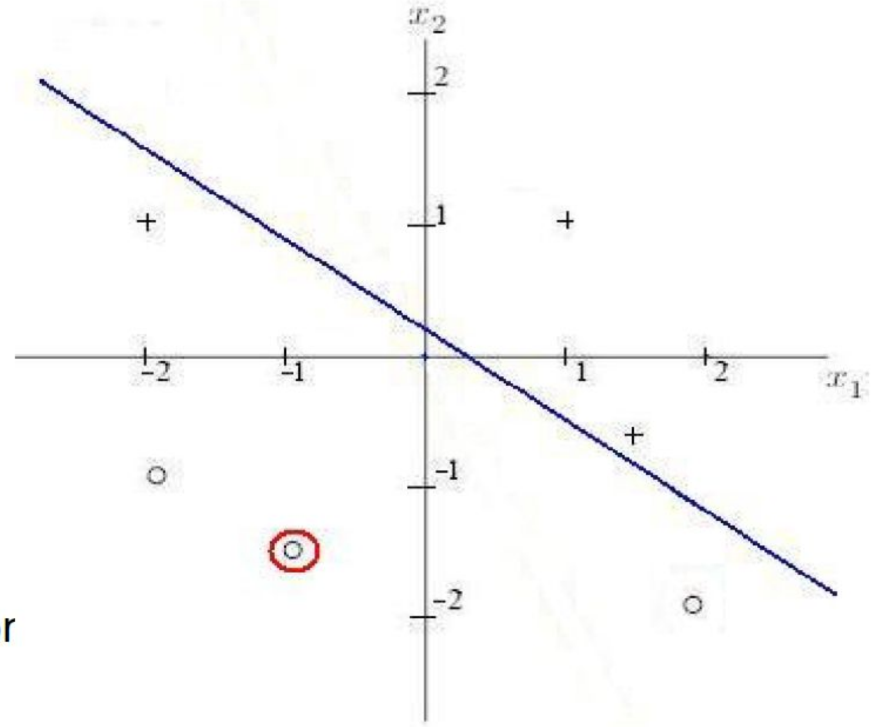
$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$x_1 = -1,\ x_2 = -1.5$

$w^T x < 0$

Correct classificatior

no action



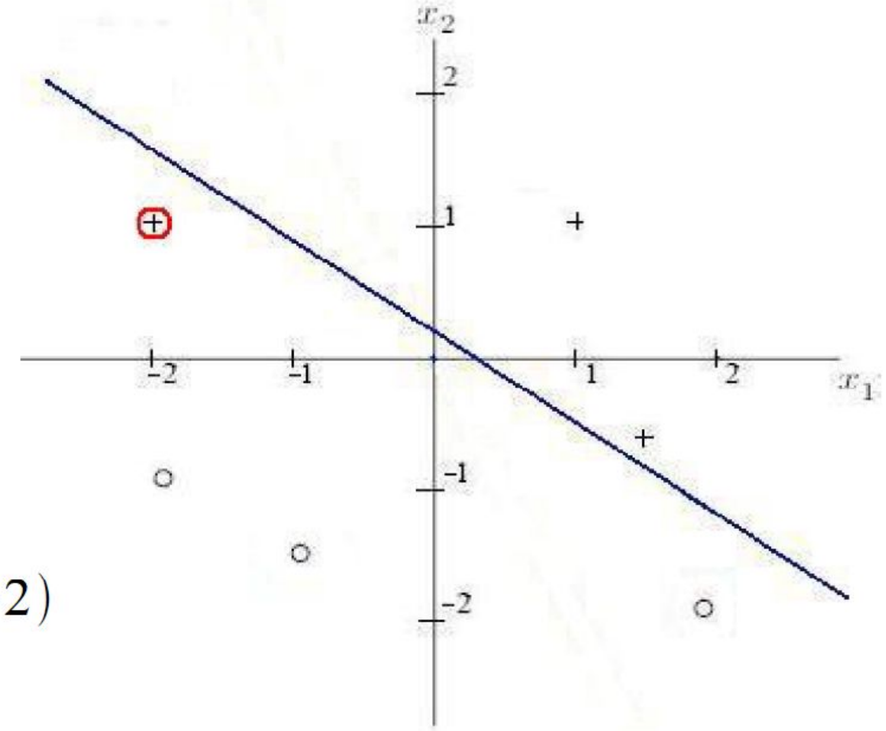
Perceptron: graphical example

$\eta = 0.2$

$$w = \begin{pmatrix} -0.2 \\ 0.6 \\ 0.9 \end{pmatrix}$$

$x_1 = -2,\ x_2 = 1$

$$w_0 = w_0 + 0.2 * 1$$
$$w_1 = w_1 + 0.2 * (-2)$$
$$w_2 = w_2 + 0.2 * 1$$



$$\boxed{new\ \mathbf{w} = \begin{bmatrix} 0 \\ 0.2 \\ 1.1 \end{bmatrix}}$$

Perceptron: graphical example

$\eta = 0.2$

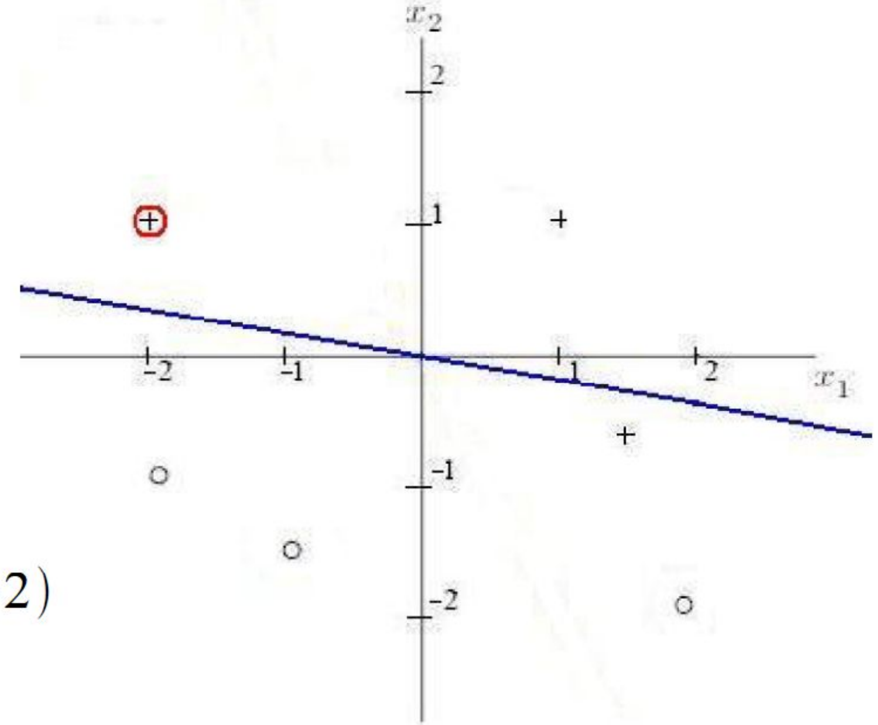
$$w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$$

$x_1 = -2,\ x_2 = 1$

$$w_0 = w_0 + 0.2 * 1$$

$$w_1 = w_1 + 0.2 * (-2)$$

$$w_2 = w_2 + 0.2 * 1$$



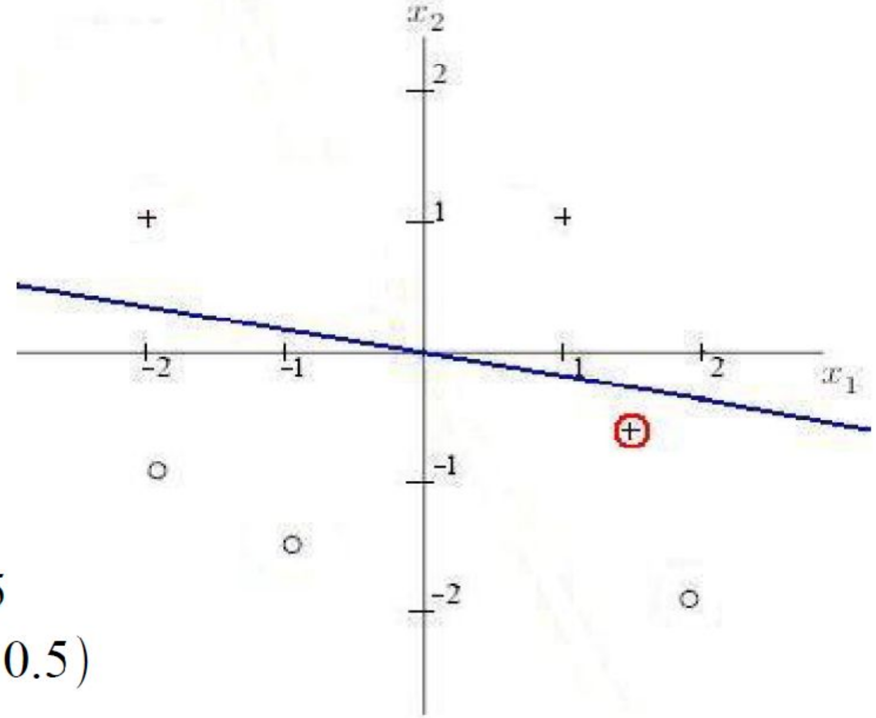
Perceptron: graphical example

$\eta = 0.2$

$$w = \begin{pmatrix} 0 \\ 0.2 \\ 1.1 \end{pmatrix}$$

$x_1 = 1.5,\ x_2 = -0.5$

$$w_0 = w_0 + 0.2 * 1$$
$$w_1 = w_1 + 0.2 * 1.5$$
$$w_2 = w_2 + 0.2 * (-0.5)$$



$$\boxed{new\ \boldsymbol{w} = \begin{bmatrix} 0.2 \\ 0.5 \\ 1 \end{bmatrix}}$$

Perceptron: graphical example

$\eta = 0.2$

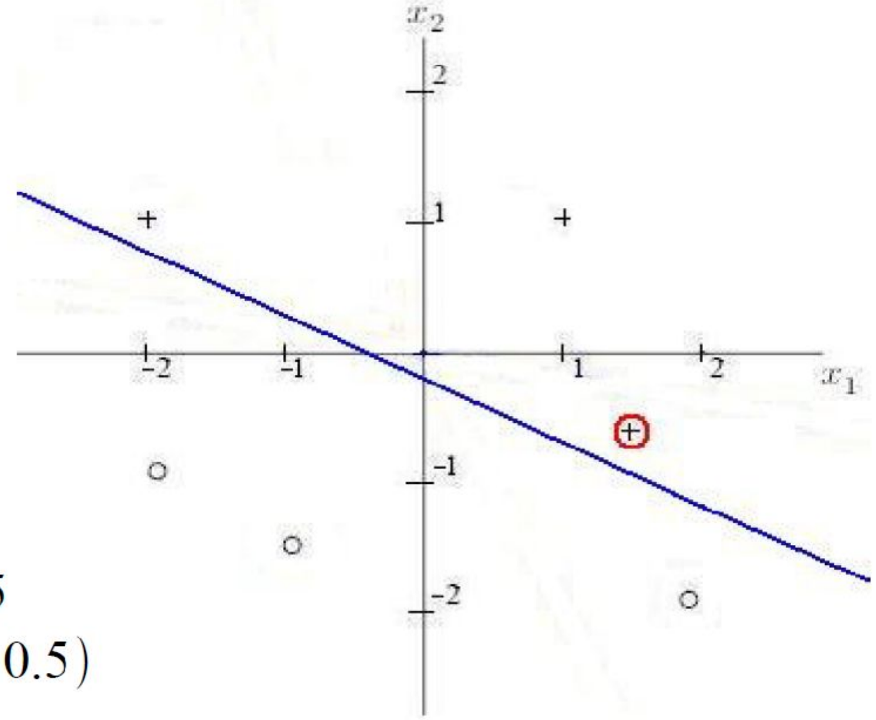
$$w = \begin{pmatrix} 0.2 \\ 0.5 \\ 1 \end{pmatrix}$$

$x_1 = 1.5,\ x_2 = -0.5$

$$w_0 = w_0 + 0.2 * 1$$

$$w_1 = w_1 + 0.2 * 1.5$$

$$w_2 = w_2 + 0.2 * (-0.5)$$



Stopping condition (ERM)

Perceptron: implementation notes

- ➔ Normalizing the input can reduce convergence time (the perceptron may swing a lot if one dimension has a much wider dynamic range)
- ➔ Randomizing the selection of the **single** sample to use at each iteration can reduce the risk of unlucky orderings (i.e., a lot of misleading samples in the first positions)
- ➔ **Pocket algorithm**: compute the error for each iteration, keep the best one so that we have an approximate solution if the set is not linearly separable
- ➔ The perceptron will reach the ERM solution, but depending on the training, it may reach *any* ERM solution