LEUPHANA
UNIVERSITÄT LÜNEBURG

INTRODUCTION TO MACHINE LEARNING

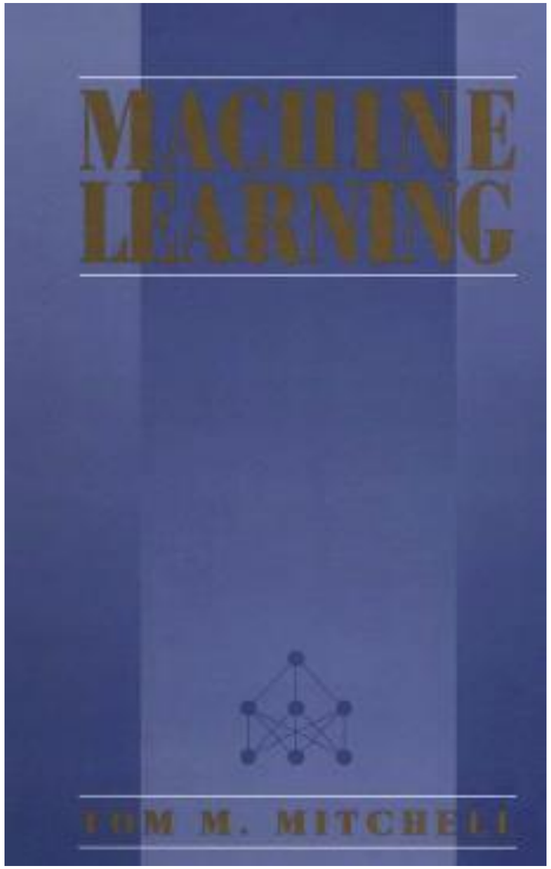
01. Linear classification
Winter 2020/2021

Agenda

- Introduction
- **Learning problem & linear classification**
- Linear models: regression & logistic regression
- Non-linear transformation, overfitting & regularization
- Support Vector Machines and kernel learning
- Neural Networks: shallow [and deep]
- Theoretical foundation of supervised learning
- Unsupervised learning

Defining a learning problem

Definition: A computer program is said to **learn** from experience **E** with respect to some class of tasks **T** and performance **P**, if its performance at tasks in T improves with E



Tom Mitchell (1998) Machine Learning

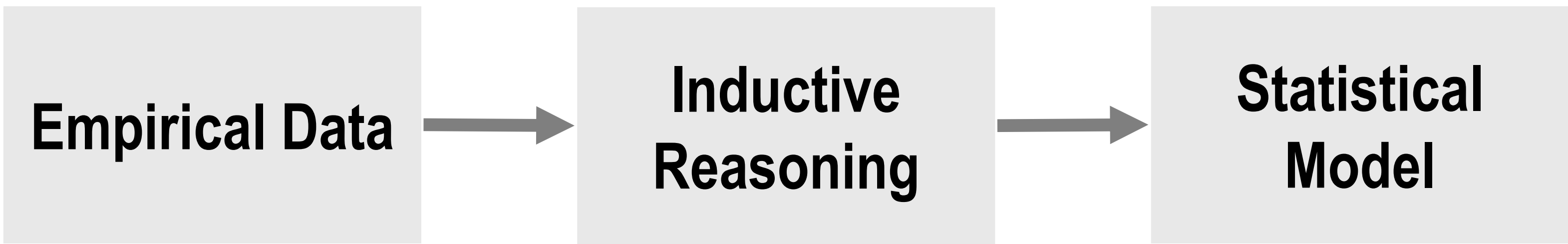
Example: Learn to play checkers

– T: play checkers
– P: % of games won
– E: opportunity to play against itself

Machine learning

— Machine learning is an „approach to achieve AI through systems that can learn from experience (data) to find patterns“

— We try to teach a computer to recognize patterns by providing examples, **rather than programming** it with specific rules.



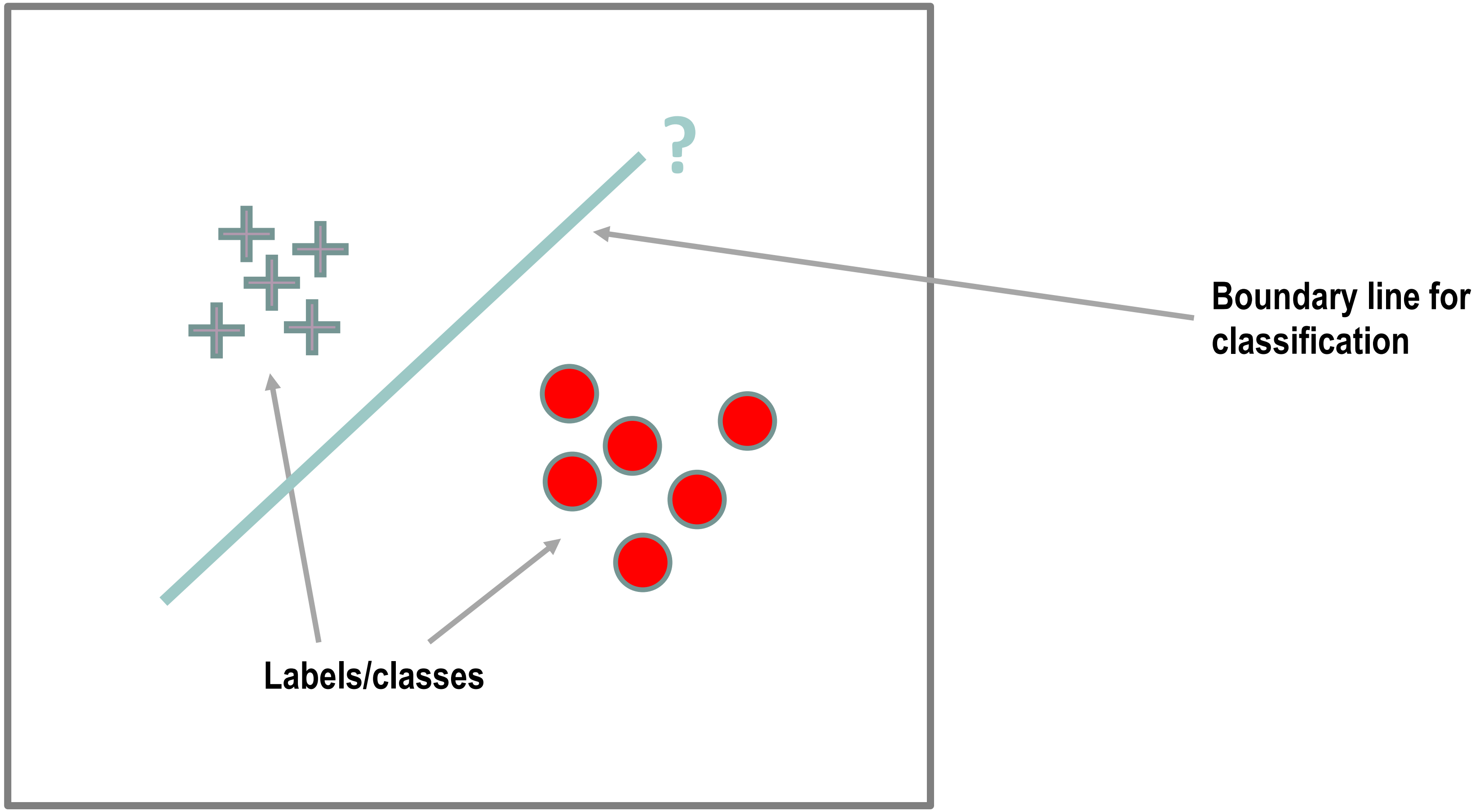
Source: Jason Mayes (2017)

Types of learning

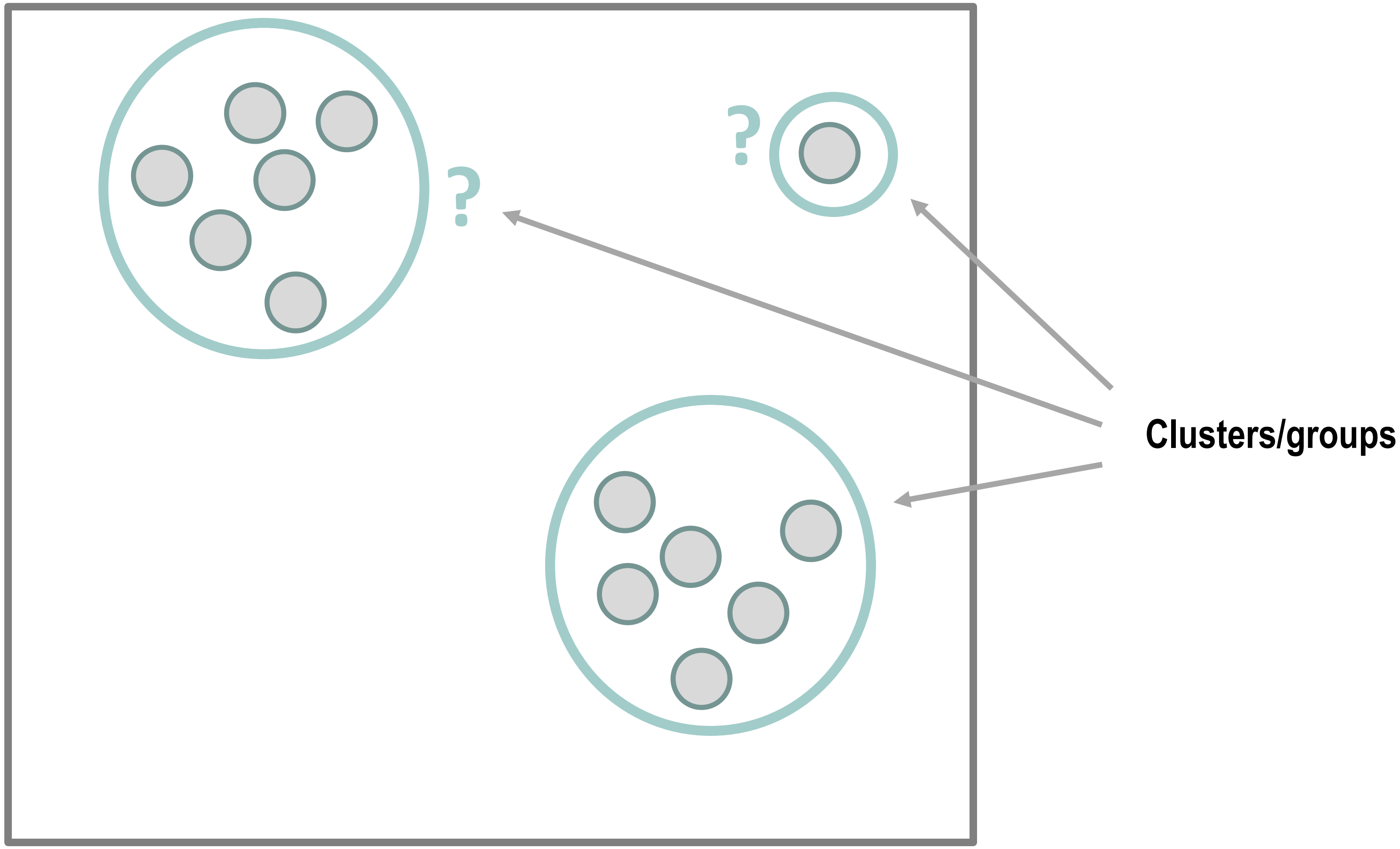
We have different types of learning, here are 4 important settings that you find in practice

- **Supervised learning (input, labels)**
- Unsupervised learning (input)
- Semi-supervised learning {lots of (input), some (input, labels)}
- Reinforcement learning

Supervised learning

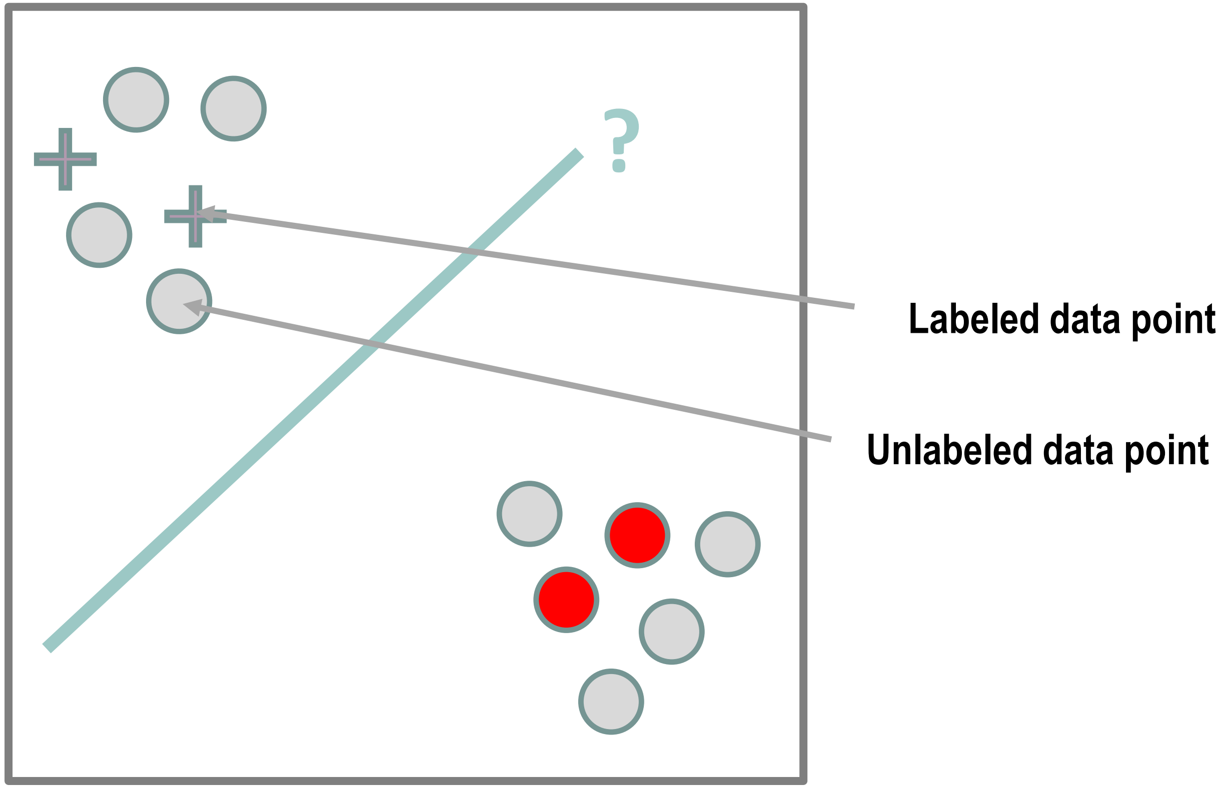


Unsupervised learning



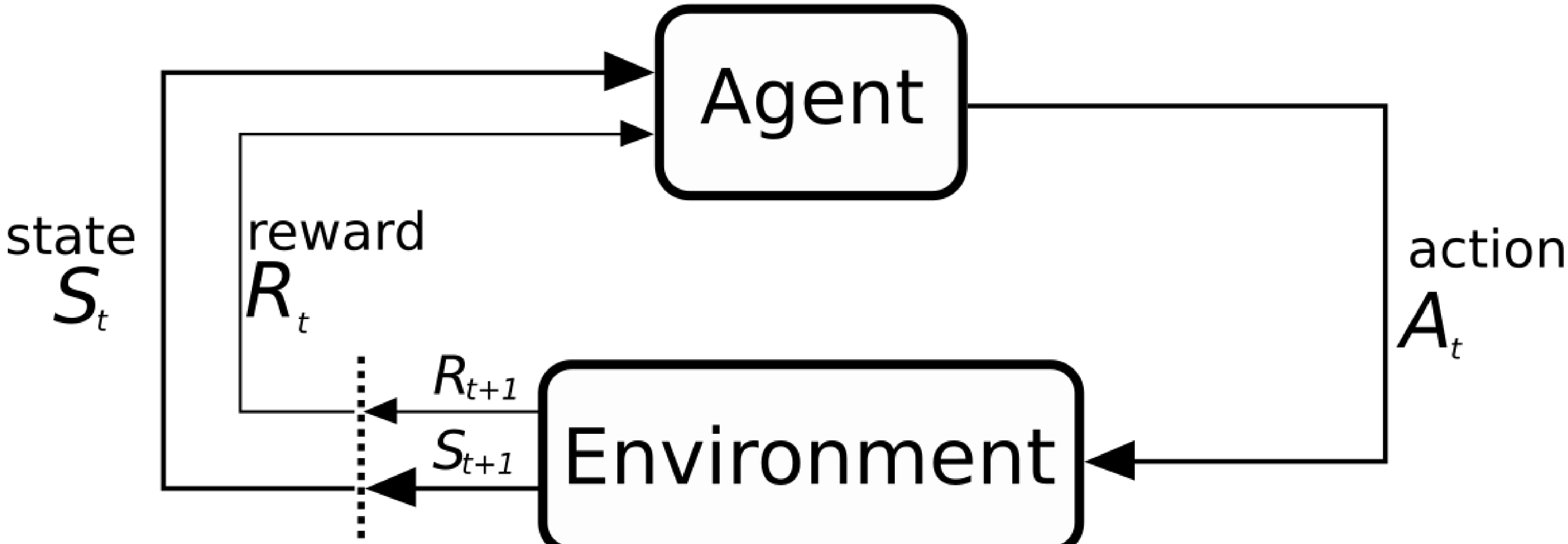
Semi-supervised learning

- Used when there is a mixture of labeled and unlabeled data
- Generally, the quantity of unlabeled data is much larger
- The goal is the same as in supervised learning, but with additional cluster information
- Useful when labeling is cost and time intensive; e. g. audio files, web pages, etc.



Source: Towards Data Science, Understanding Semi-supervised Learning

Reinforcement learning



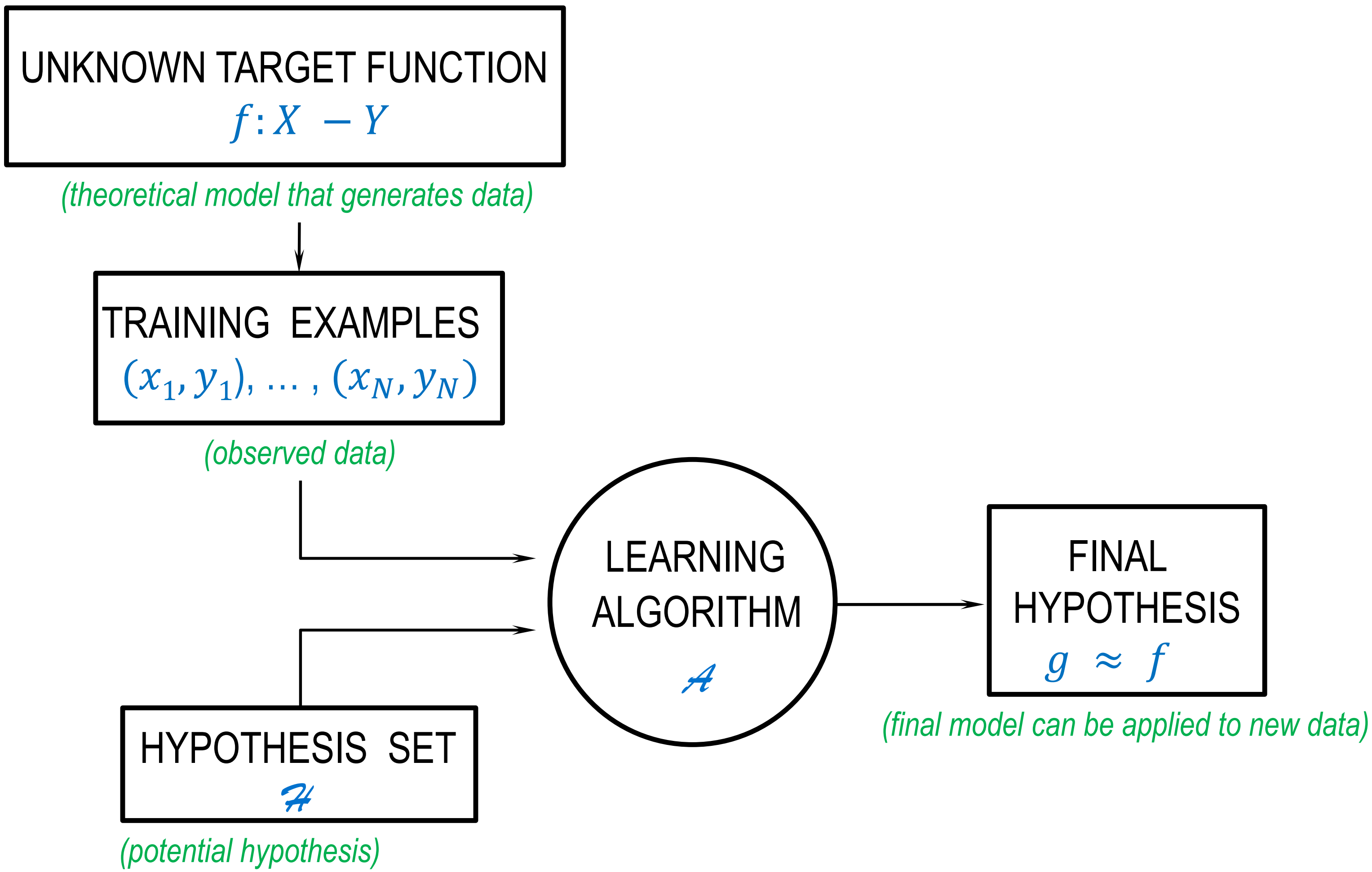
https://commons.wikimedia.org/wiki/File:Markov_diagram_v2.svg

Agenda

— Introduction
— **Learning problem & linear classification**
— Linear models: regression & logistic regression
— Non-linear transformation, overfitting & regularization
— Support Vector Machines and kernel learning
— Neural Networks: shallow [and deep]
— Theoretical foundation of supervised learning
— Unsupervised learning

Notation and terminology

Components of Learning



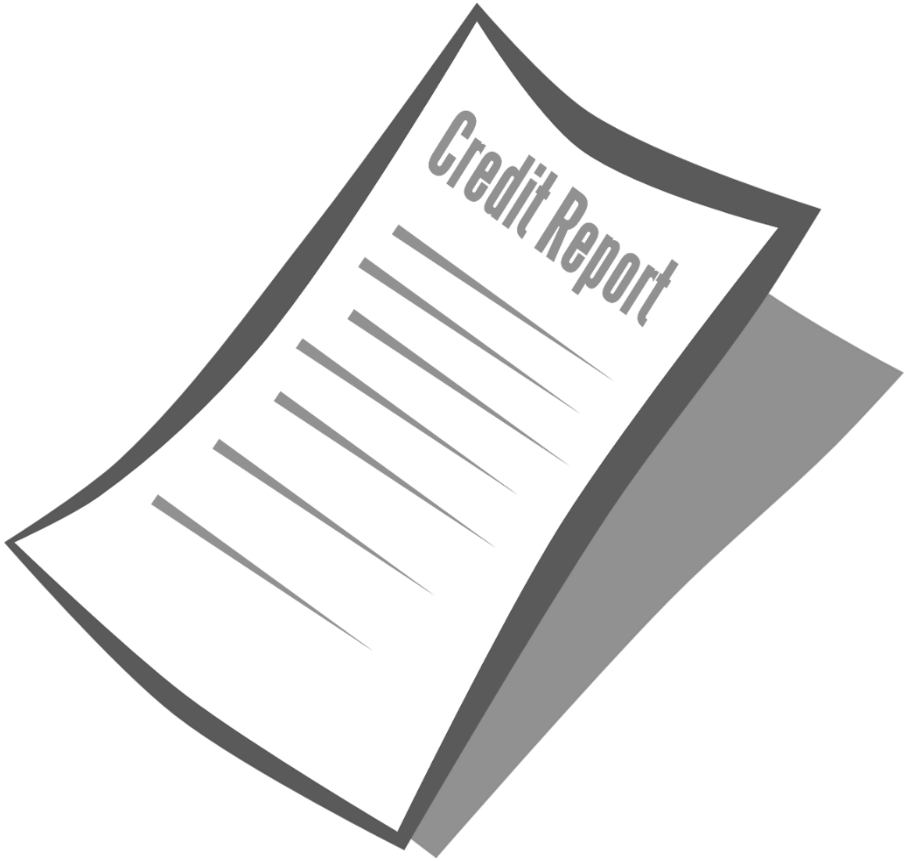
Source: Abu-Mustafa (2012) et al. LfD – many of the following charts are taken from this lecture series and correspond to the LFD book

Key issues in machine learning

Transform real-world problem into computable learning problem

— **Define label space**: what do we want to learn (e.g. playing checkers: board move vs. board value)?
— **Determine type of training data**/experience (e.g. given/biased, self-selected?)
— **Represent input data** (feature definition and extraction, functional form)
— Learning algorithm
— Evaluation (Is it a good model? Is it good enough?)
— What decisions shall be informed?

Credit scoring – a learning example

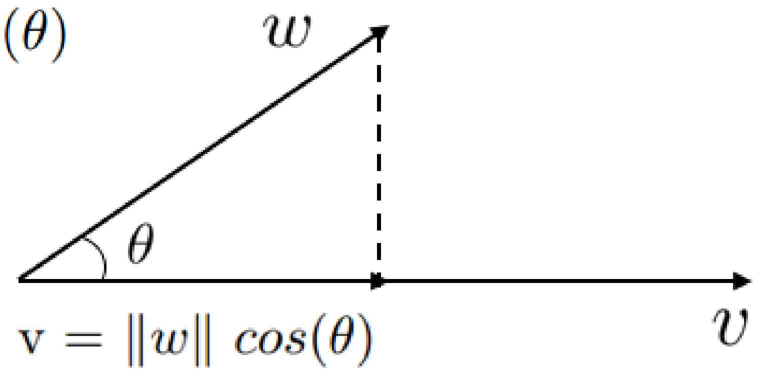


Source: https://www.needpix.com/

— Suppose we want to inform credit decisions of a bank. What information is included in a credit request from a customer?
— What features will have a positive impact on the credit decision?
— How could the set of hypothesis be represented?

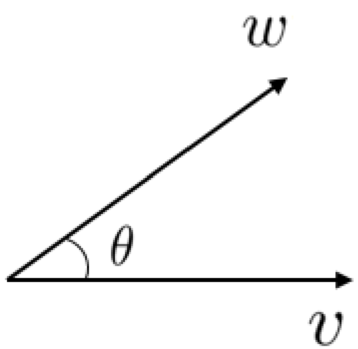
Dot product: a quick recap

$$w^T v = \|w\| \; \|v\| \; cos(\theta)$$

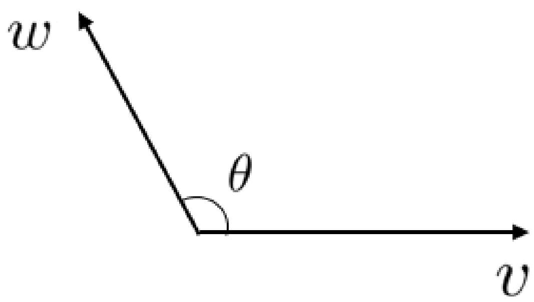


projection of w onto v $= \|w\| \; cos(\theta)$

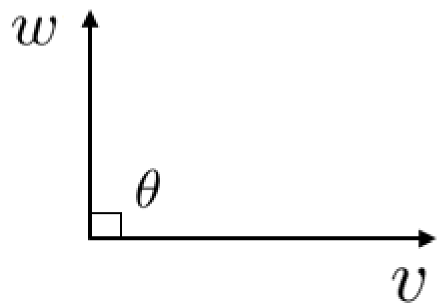
$w^T v > 0, \quad 0° \leq \theta < 90°$



$w^T v < 0, \quad 90° < \theta \leq 180°$



$w^T v = 0, \quad \theta = 90°$



A simple decision model

— Input vector $\boldsymbol{x} = [x_1, \dots, x_d]^T$

— Find weights for different inputs and compute *credit score*

$$\textit{credit score} = \sum_{i=1}^{d} w_i x_i \, .$$

— Approve credit if the *credit score* is acceptable.

Approve credit if $\sum_{i=1}^{d} w_i x_i \geq$ threshold, (*credit score* is good)

Deny credit if $\sum_{i=1}^{d} w_i x_i <$ threshold. (*credit score* is bad)

— **How to choose the "importance" weights $\boldsymbol{w_i}$**

input x_i is important ⇒ large weight $|w_i|$

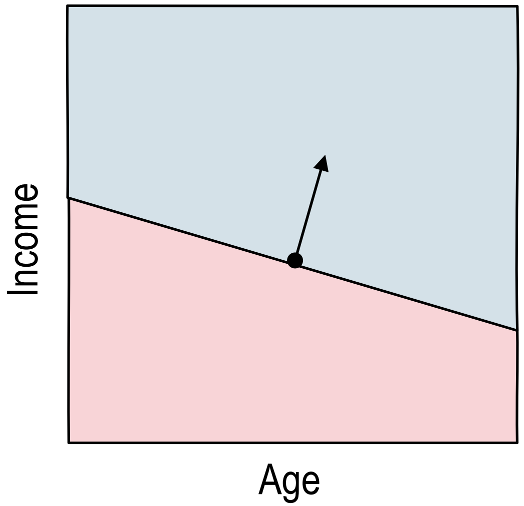
input x_i beneficial for credit ⇒ positive weight $w_i > 0$

input x_i detrimental for credit ⇒ negative weight $w_i < 0$

Simple decision model – revised version

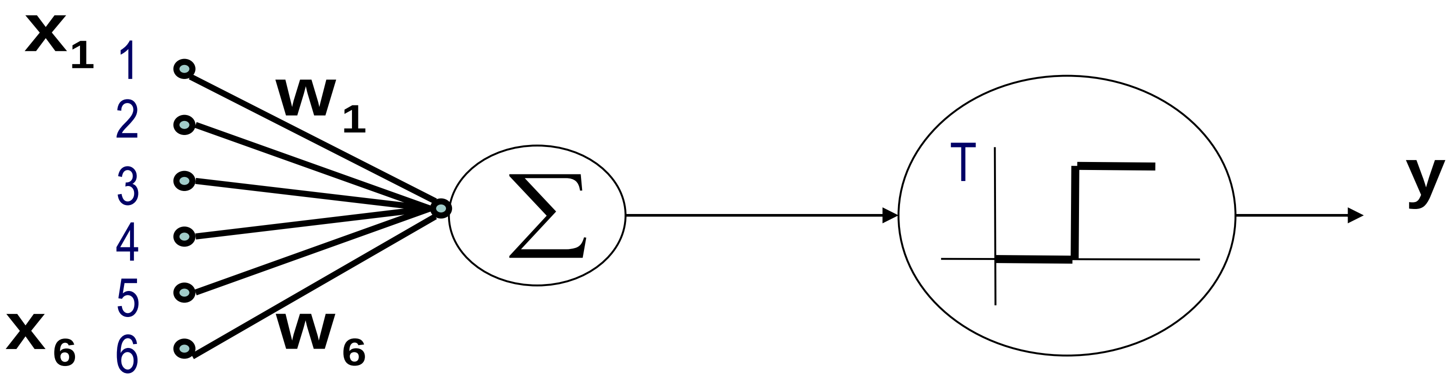
Visualizing the decision boundary

$$h(\mathbf{x}) = \mathrm{sign}(\mathrm{w}^{\mathrm{T}}\mathrm{x})$$



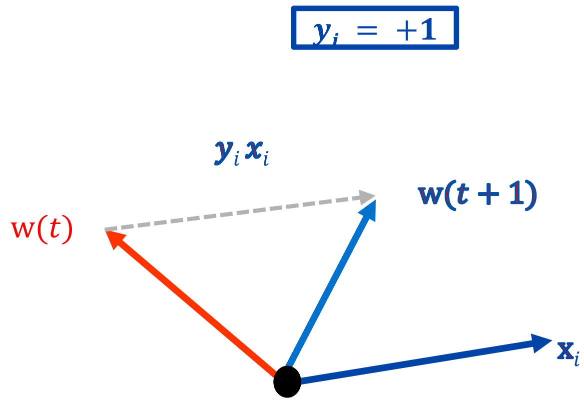
Online Learning - Perceptron

- Rosenblatt (1959) suggested that when a target output value is provided for a single neuron with fixed input, it can incrementally change weights and learn to produce the output using the Perceptron learning rule
- Online learning (“as new data comes in”), mistake driven algorithm
- Perceptron = Linear Threshold Unit



Source: Dan Ross: CS 446: Machine Learning

Perceptron Algorithm – basic idea



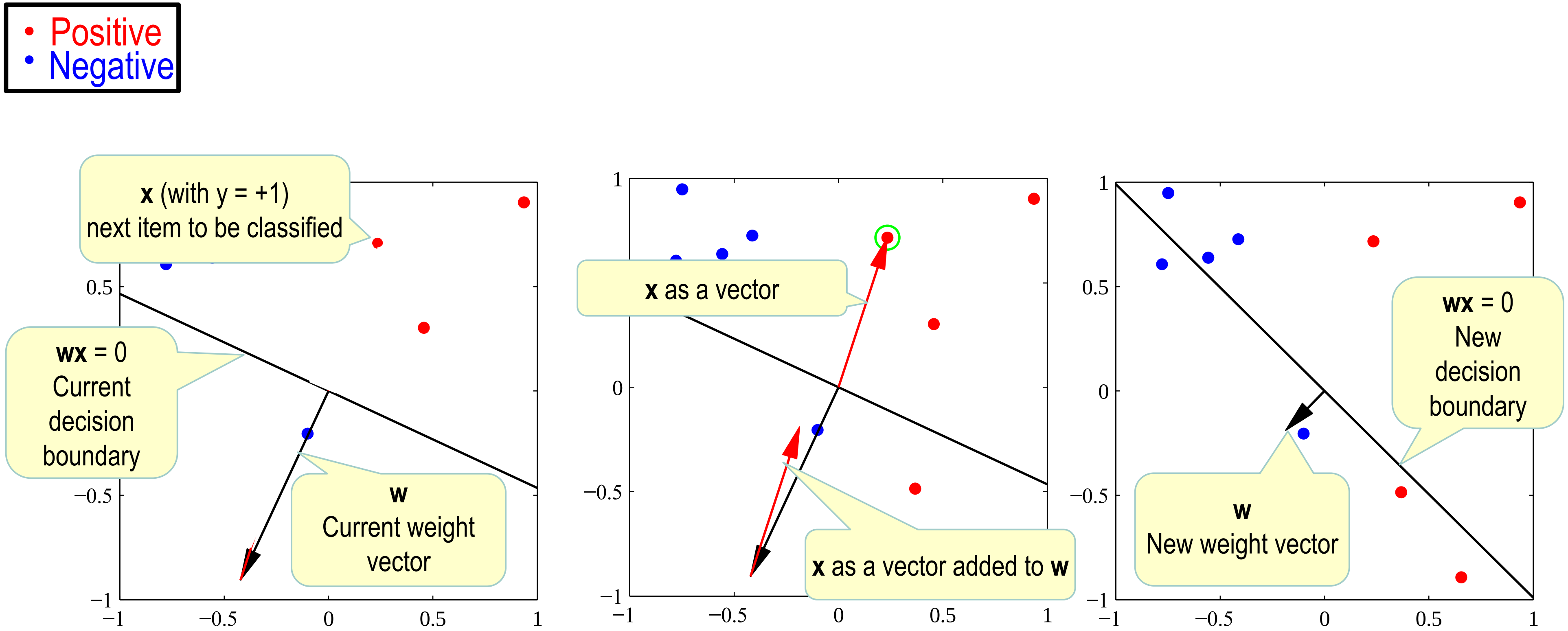
Assume that data is linearly separable

```
Start with arbitrary w(1) = 0
Do until all training data is correctly classified
      Pick any misclassified example (x_i, y_i)
      Update the weight:
```

$$w(t+1) = w(t) + \eta y_i x_i$$

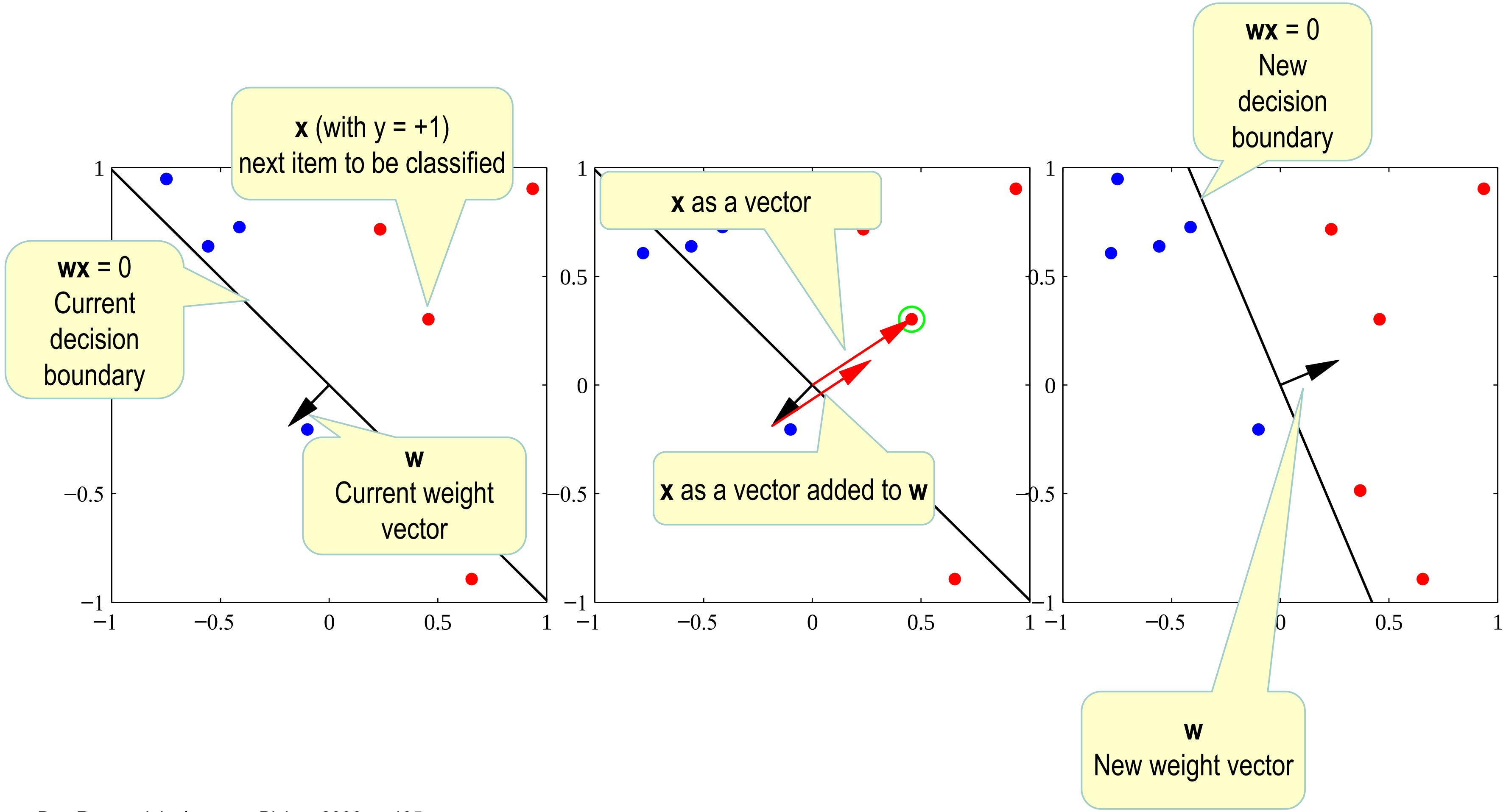
PLA implements incremental learning, how should we choose the learning rate?

How the algorithm adjusts the weight vector



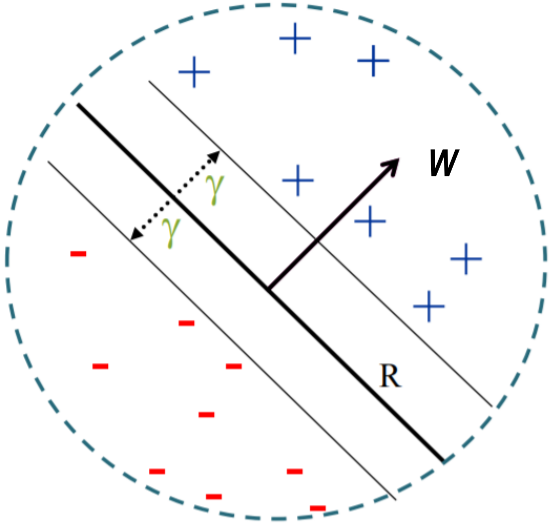
Source: Dan Ross, original source: Bishop 2006, p. 195

How the algorithm adjusts the weight vector



Source: Dan Ross, original source: Bishop 2006, p. 195

Number of mistakes the PLA makes is bound



Theorem (Novikoff 1962)

Let S be a training set and let

$$R = \max_{1 \le i \le n} \|x_i\|_2$$

Suppose that there exists a vector w such that $\|w\| = 1$ and $\gamma > 0$

$$y_i w^T x_i > \gamma$$ => *data is linearly separable*

for $1 \le i \le n$. Then the number of mistakes made by the online perceptron algorithm on S is at most

$$\left(\frac{R}{\gamma}\right)^2$$

Source: Machine Learning Department, School of Computer Science, Carnegie Mellon University

For a proof see e.g. Andrew Ng lecture notes (lecture 6 large margin classifiers)

Detailed PLA - pseudo code

Given a linearly separable training set S and learning rate $\eta \epsilon \mathbb{R}^+$

$\boldsymbol{w}_0 \leftarrow 0;\ b_0 \leftarrow 0;\ k \leftarrow 0$

$R \leftarrow \max_{1 \le i \le l} \|x_i\|$

repeat

for $i = 1$ to l

if $y_i(\langle w_k \cdot x_i \rangle + b_k) \le 0$ then

$w_{k+1} \leftarrow w_k + \eta y_i x_i$

$b_{k+1} \leftarrow b_k + \eta y_i R^2$

$k \leftarrow k + 1$

end if

end for

Until there are no mistakes within the *for* loop

Return the list (w_k, b_k)

Source: Christianini & Shawe-Taylor (2000), p. 12

Concluding remarks

— PLA always converges if data is linearly separable

— PLA and problem setting have everything we discussed to define a learning problem

— Try applying this to the Australia Rain dataset to predict the occurrence of rainfall. Does it converge? Why not?

Solution: Multi-Layer Perceptron (Neural Networks) or feature transformation

Historical remarks:

— Frank Rosenblatt simulated PLA on an IBM 704 in 1957 – built special hardware, were able to recognize characters from “photos”

— Already simple mechanisms can generate data that are not linearly separable (e.g. XOR) as discussed by Minsky and Papert (1969) → lead to a stop of research on neural networks

Implement perceptron (problem set 1)

Given a linearly separable training set S and learning rate $\eta \in \mathbb{R}^+$

$\boldsymbol{w}_0 \leftarrow 0;\ b_0 \leftarrow 0;\ k \leftarrow 0$

$R \leftarrow \max_{1 \le i \le l} ||x_i||$

repeat

for $i = 1$ to l

if $y_i(\langle w_k \cdot x_i \rangle + b_k) \le 0$ then

$w_{k+1} \leftarrow w_k + \eta y_i x_i$

$b_{k+1} \leftarrow b_k + \eta y_i R^2$

$k \leftarrow k + 1$

end if

end for

Until there are no mistakes within the *for* loop

Return the list (w_k, b_k)

Christianini & Shawe-Taylor (2000)

Any questions/ thoughts?