Course 2: Supervised Learning

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Summary

Last session

1. AI definition
2. Applications
3. Deep learning
4. Open issues

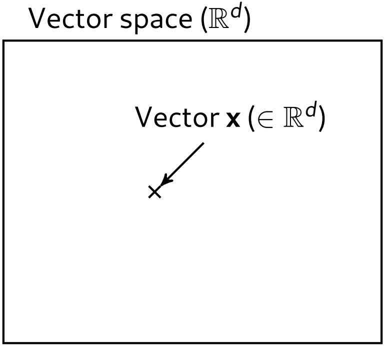
Today's session

- Learning from labeled examples
- Challenges of supervised learning

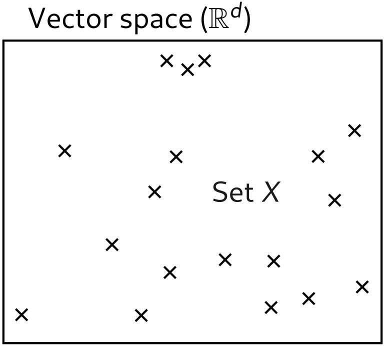
Notations

Vector space ($\mathbb{R}^d$)

Notations



Notations



Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...

Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

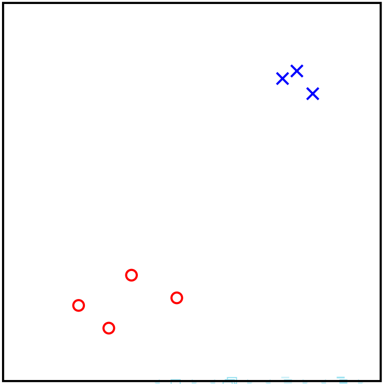
- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...

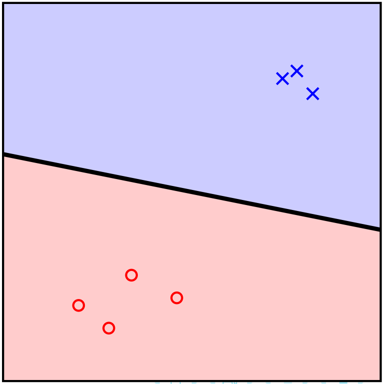


Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



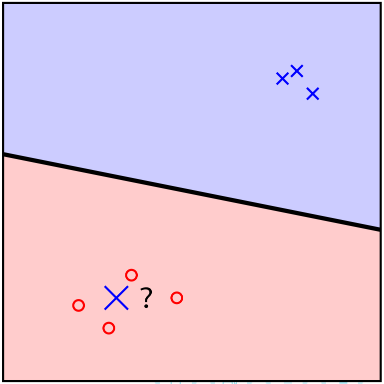
Supervised learning

Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction…

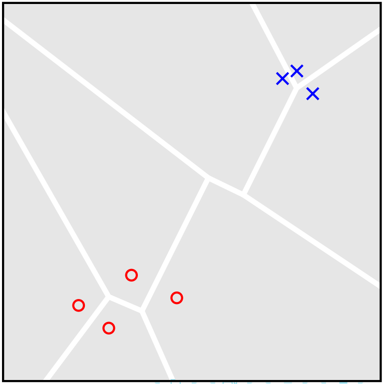


Definition

Supervised learning methods use **labels** $\hat{\mathbf{y}}$ associated with examples $\mathbf{x}$ to learn a function f such as $\hat{\mathbf{y}} \approx f(\mathbf{x})$, with the aim of **generalizing** ($\neq$ memorizing) to unlabeled examples.

Examples

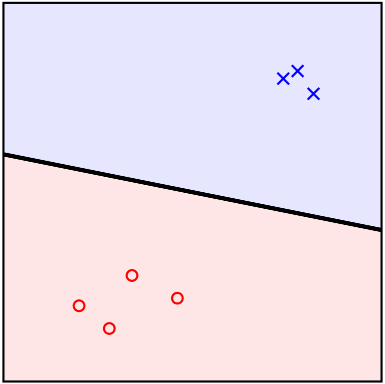
- **Regression** ($\mathbf{y}$ is scalar)
- **Classification** ($\mathbf{y}$ is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction…



Challenges of supervised learning (1/5)

An ill-defined problem

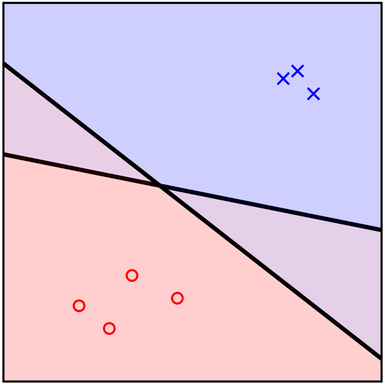
- An infinity of potential solutions, one must be the “best one” but is unreachable,
- $\Rightarrow$ requires **priors or constraints**.



Challenges of supervised learning (1/5)

An ill-defined problem

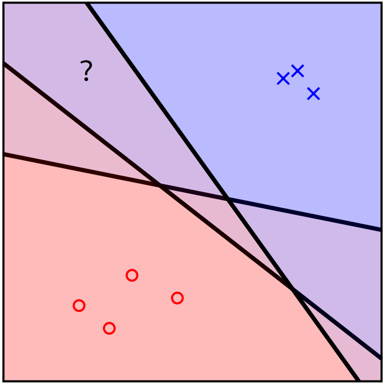
- An infinity of potential solutions, one must be the "best one" but is unreachable,
- $\Rightarrow$ requires **priors or constraints**.



Challenges of supervised learning (1/5)

An ill-defined problem

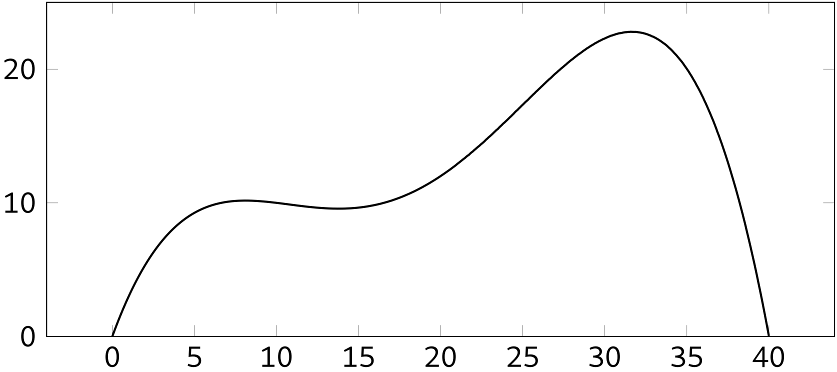
- An infinity of potential solutions, one must be the "best one" but is unreachable,
- $\Rightarrow$ requires **priors or constraints**.



Challenges of supervised learning (2/5)

Bias/variance trade-off

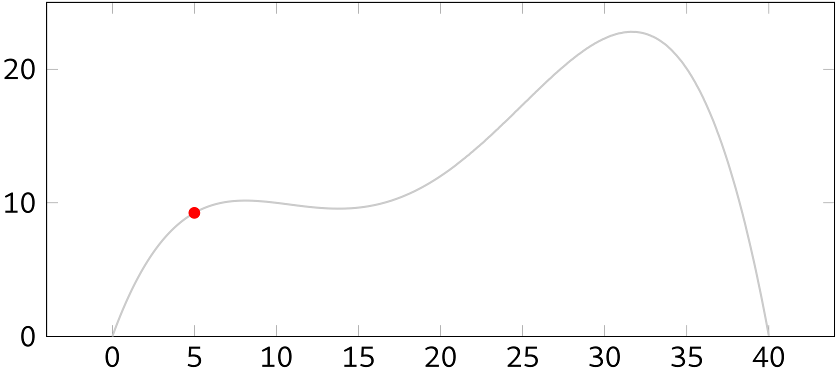
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

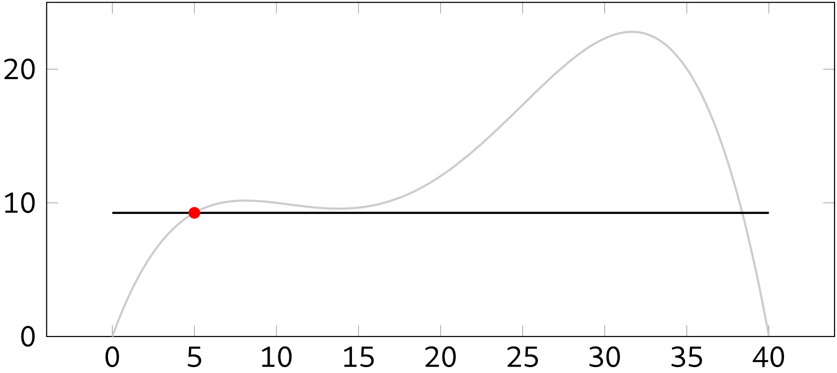
Bias/variance trade-off

- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Bias/variance trade-off

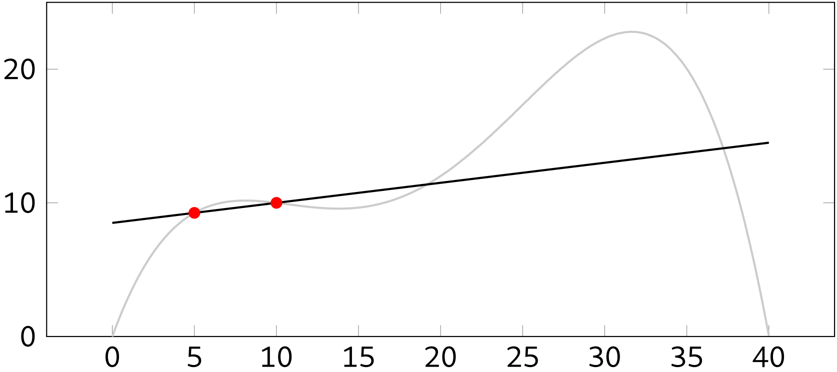
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

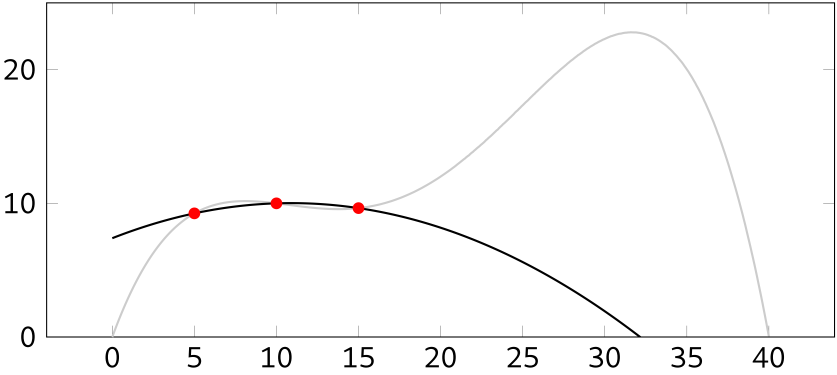
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

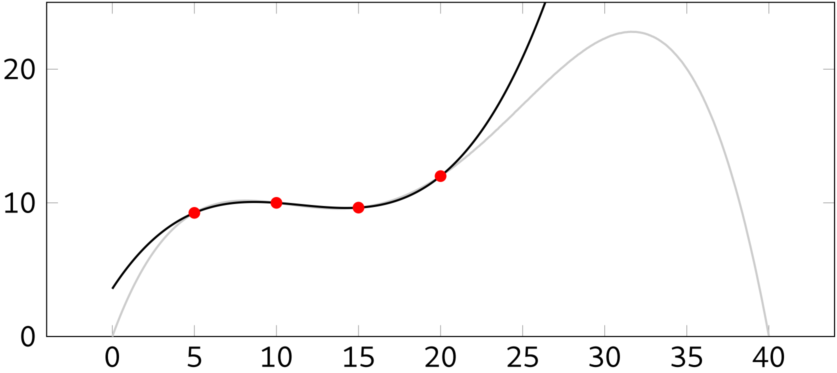
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

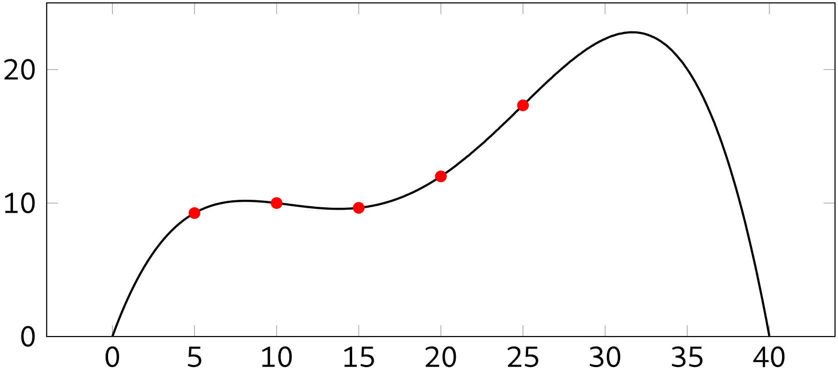
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

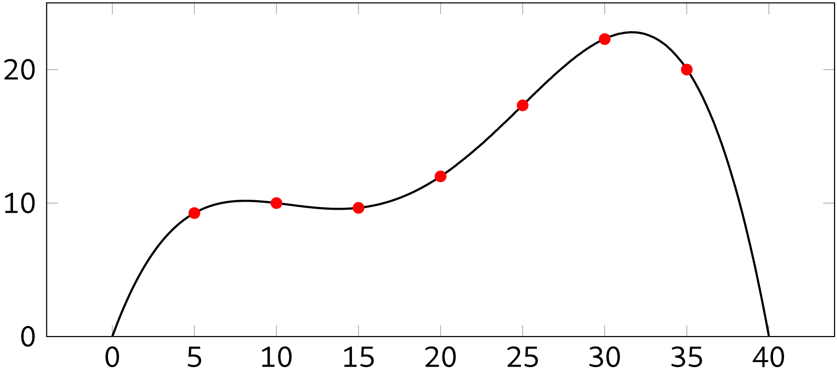
Bias/variance trade-off

- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Bias/variance trade-off

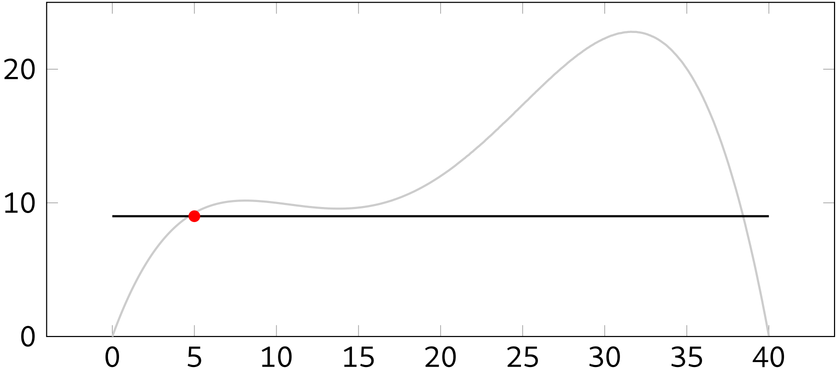
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

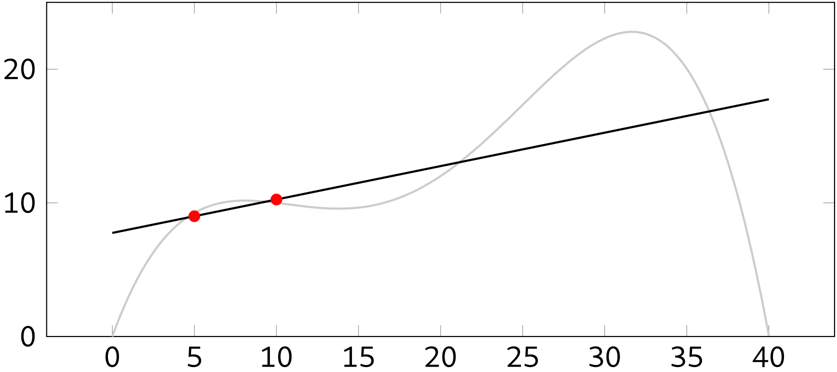
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

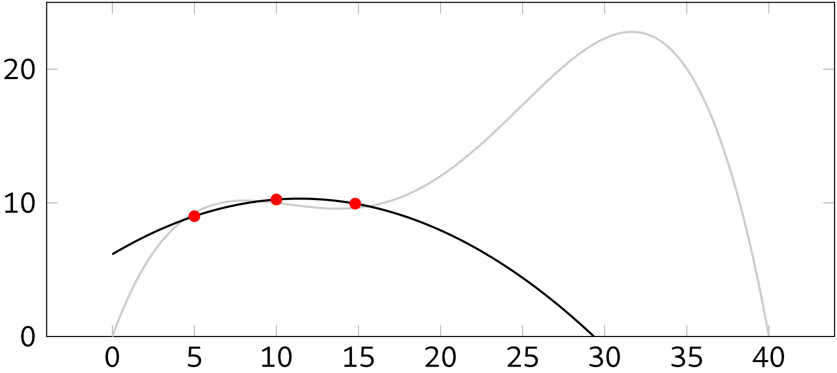
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

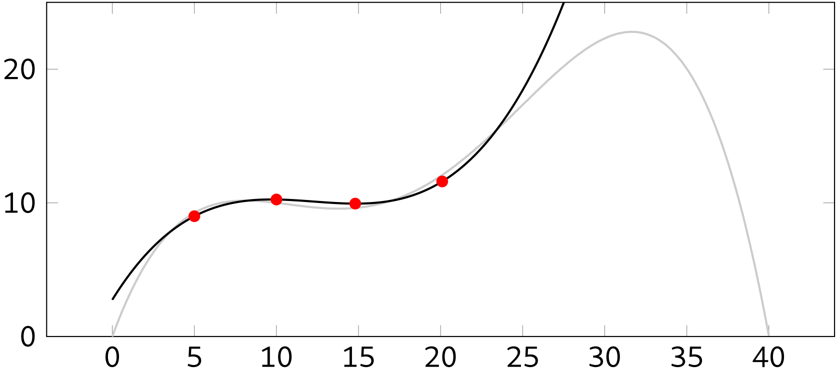
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

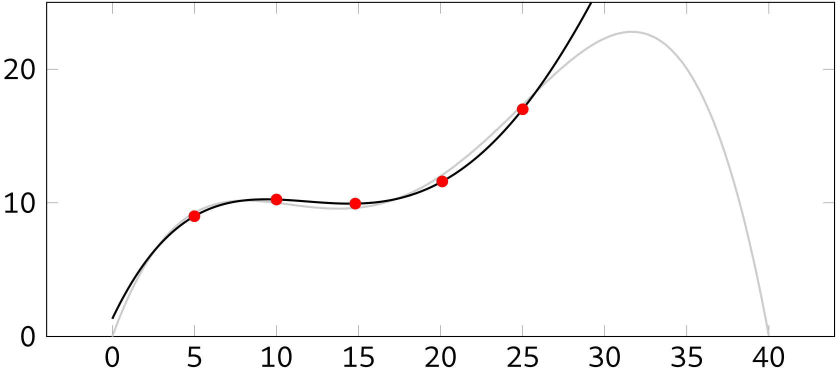
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

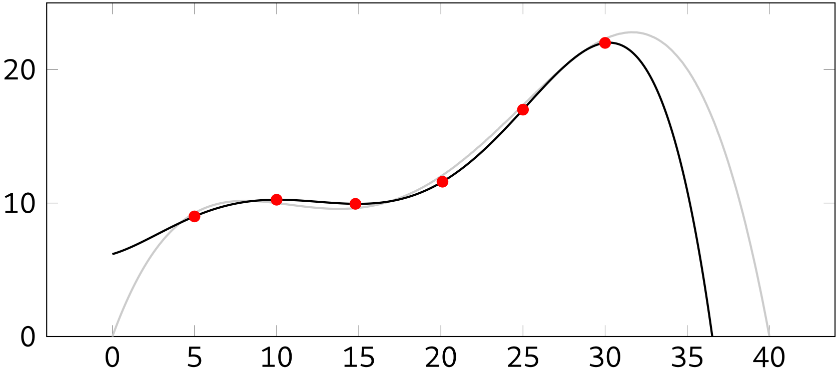
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

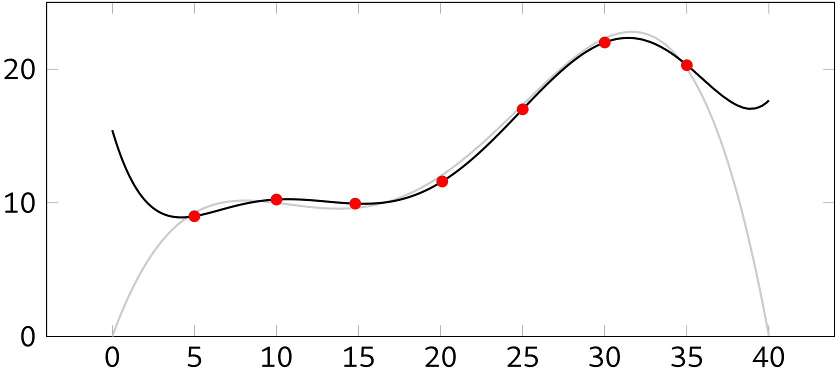
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

Bias/variance trade-off

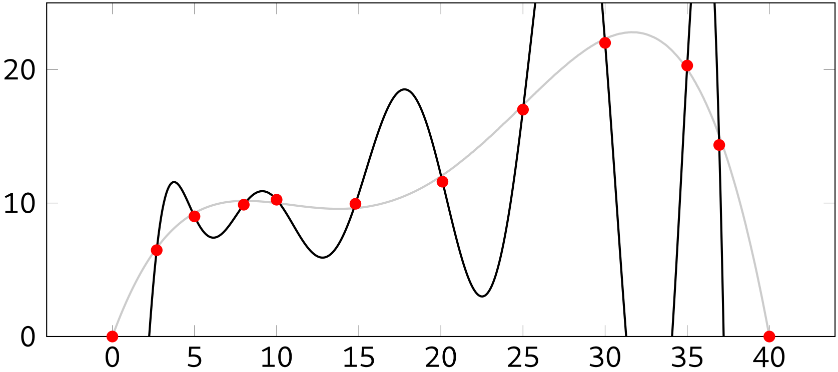
- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Challenges of supervised learning (2/5)

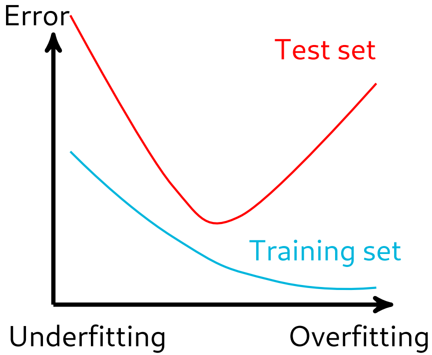
Bias/variance trade-off

- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Bias/variance trade-off

- A **simple** solution that almost matches is better than a complex one that fully matches,
- Mimicking is not learning: **overfitting** problem.



Crossvalidation

- To detect overfitting, split training dataset in two parts:
  1. A first part is used to train,
  2. A second part is used to validate,

Curse of dimensionality

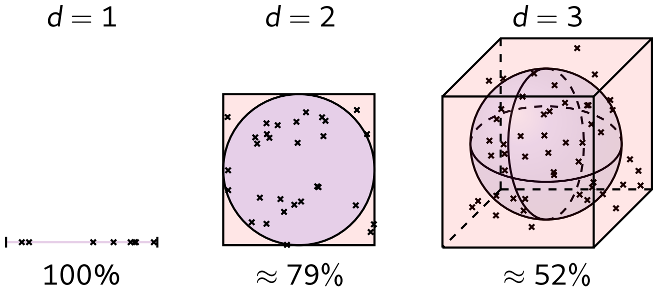
- Geometry is not intuitive in **high dimension**,
- Efficient methods in 2D are not necessarily still valid.

$$V_d^s = \frac{\pi^{d/2} R^d}{\Gamma(d/2+1)} \text{ versus } V_d^c = (2R)^d$$

see https://youtu.be/dZrGXYty3qc?t=533

Curse of dimensionality

- Geometry is not intuitive in **high dimension**,
- Efficient methods in 2D are not necessarily still valid.

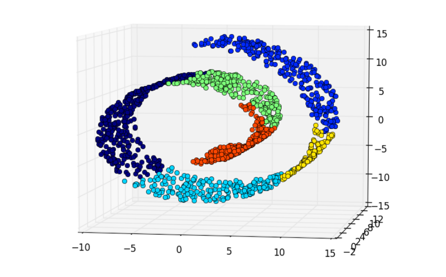


$$V_d^s = \frac{\pi^{d/2} R^d}{\Gamma(d/2+1)} \text{ versus } V_d^c = (2R)^d$$

see `https://youtu.be/dZrGXYty3qc?t=533`

Challenges of supervised learning (4/5)

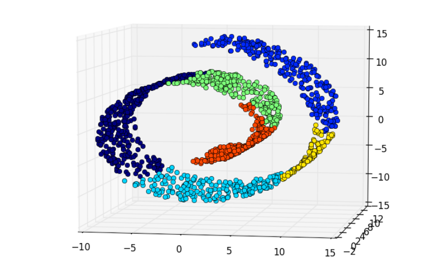
Riemannian manifolds



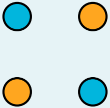
Linear separability and need for embedding

Challenges of supervised learning (4/5)

Riemannian manifolds



Linear separability and need for embedding



Challenges of supervised learning (5/5)

Computation time

Example on ImageNet, simply going through all images:

- $n = 10.000.000$, $d \approx 1.000.000$,
- $\approx 10^{13}$ elementary operations,
- $\approx$ 2h45 on a modern processor.

Scalability

- Finding the best solution to a problem would be feasible with unlimited computation time,
- But searching through the space of possible functions is often **untractable**,
- Solutions must be computationally reasonable, which is the true challenge today.

Challenges of supervised learning (5/5)

Computation time

Example on ImageNet, simply going through all images:

- $n = 10.000.000$, $d \approx 1.000.000$,
- $\approx 10^{13}$ elementary operations,
- $\approx$ 2h45 on a modern processor.

Scalability

- Finding the best solution to a problem would be feasible with unlimited computation time,
- But searching through the space of possible functions is often **untractable**,
- Solutions must be computationally reasonable, which is the true challenge today.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

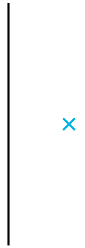
Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.



Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

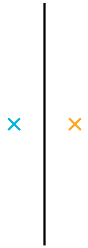
Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.



Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

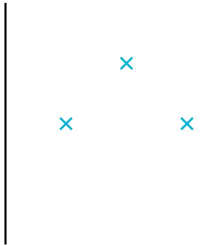
Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.



Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

Vapnik Chervonenki (VC) dimension

Definition

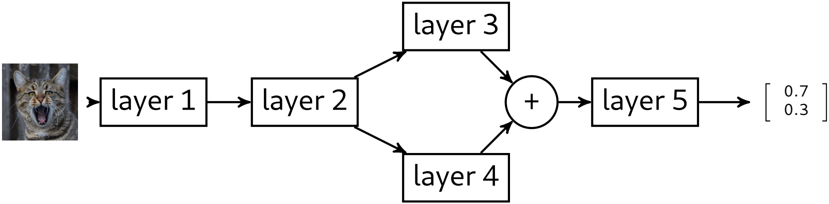
- Let us fix d,
- The **VC dimension** is a measure of the genericity of a method,
- It is the **maximum cardinality** of a set of vectors that the method is able to shatter in any possible way.

Consider for example lines to shatter set of points with $d = 2$.

VC is 3.

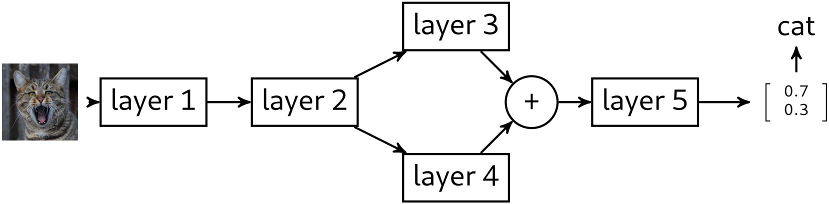
Inputs/outputs

- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



Inputs/outputs

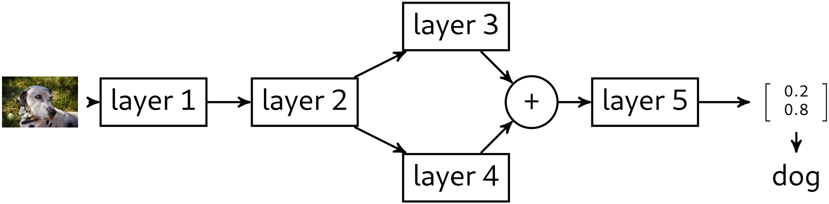
- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



The case of deep learning in classification

Inputs/outputs

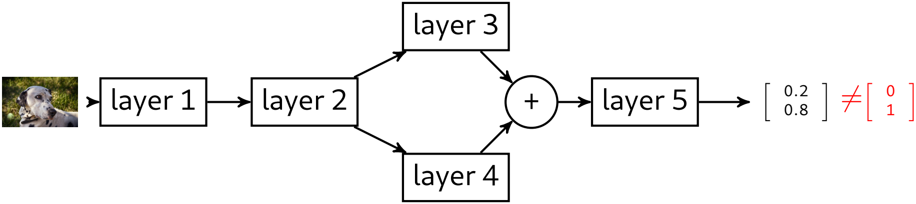
- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



The case of deep learning in classification

Inputs/outputs

- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



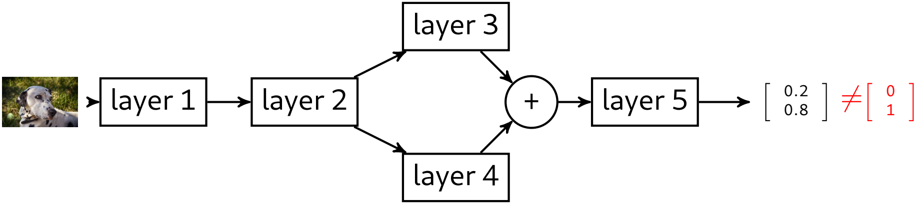
Loss and targets

- Labels are encoded as one-hot-bit vectors and called **targets**,
- Outputs are **softmaxed:** $\mathbf{y}_i \leftarrow \exp(\mathbf{y}_i)/\sum_j \exp(\mathbf{y}_j)$,
- Loss is typically **cross-entropy:** $-\log(\hat{\mathbf{y}}^\top \mathbf{y})$.

The case of deep learning in classification

Inputs/outputs

- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



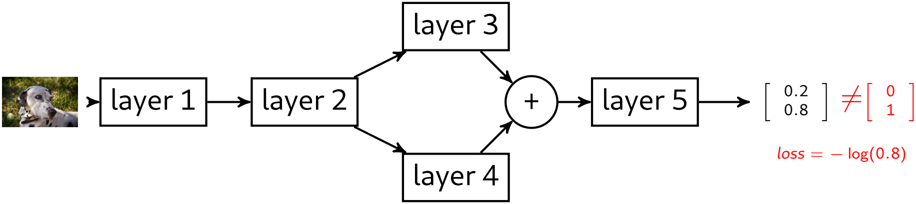
Loss and targets

- Labels are encoded as one-hot-bit vectors and called **targets**,
- Outputs are **softmaxed:** $\mathbf{y}_i \leftarrow \exp(\mathbf{y}_i)/\sum_j \exp(\mathbf{y}_j)$,
- Loss is typically **cross-entropy:** $-\log(\hat{\mathbf{y}}^\top \mathbf{y})$.

The case of deep learning in classification

Inputs/outputs

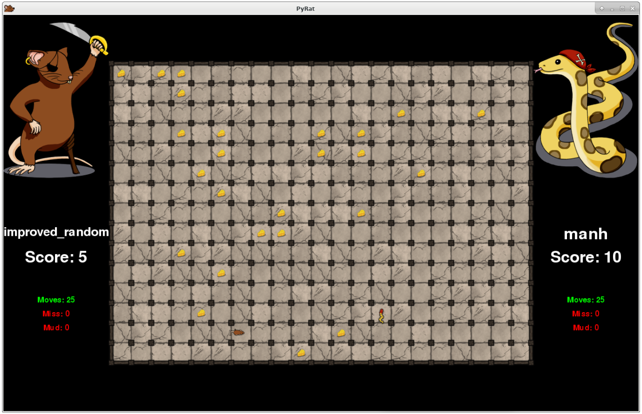
- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



Loss and targets

- Labels are encoded as one-hot-bit vectors and called **targets**,
- Outputs are **softmaxed**: $\mathbf{y}_i \leftarrow \exp(\mathbf{y}_i)/\sum_j \exp(\mathbf{y}_j)$,
- Loss is typically **cross-entropy**: $-\log(\hat{\mathbf{y}}^\top \mathbf{y})$.

Non-symmetric PyRat without walls / mud



Both players follow a deterministic greedy algorithm.

Supervised learning - Two tasks

- Predict the outcome of a game from the start configuration.
- Learn the next move using a dataset of winners

Lab Session 2 and assignments for Session 3

TP Supervised Learning (TP1)

- Basics of machine learning using sklearn (including new definitions / concepts) and pytorch
- Tests on PyRat datasets using the two tasks (predicting winner and predicting moves to play)

Project 1 (P1)

You will choose a supervised learning method. You have to prepare a Jupyter Notebook on this method, including:

- A brief description of the theory behind the method,
- Basic tests on simulated data to show the influence of parameters and hyperparameters
- Tests on PyRat Datasets on at least ONE of the two tasks (predicting winner or playing)

During Session 3 you will have 7 minutes to present your notebook.