Course 2: Supervised Learning

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

2022-02-11

Course 2: Supervised Learning

Course 2: Supervised Learning

Summary

Last session

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

Today's session

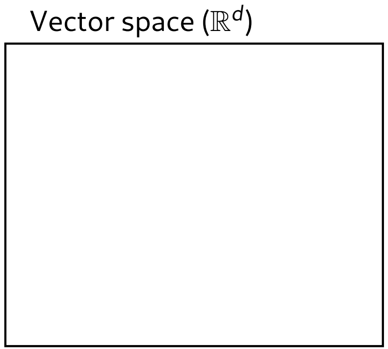
- Learning from labeled examples
- Challenges of supervised learning

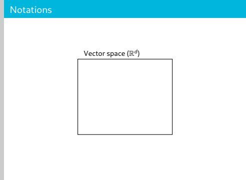
Summary

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

Today's session
- Learning from labeled examples
- Challenges of supervised learning

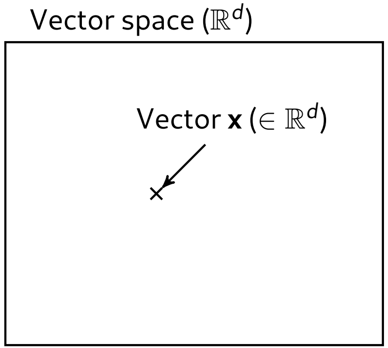
Notations

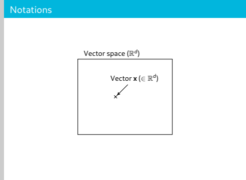




We denote a vector space of real values in dimension d. We will consider vectors x in this space, and the set big X of all such vectors.

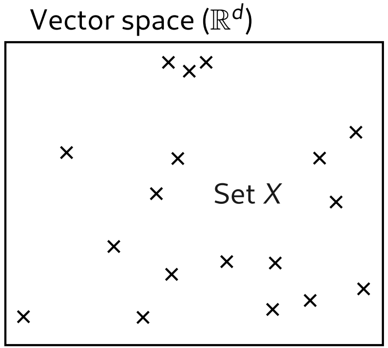
Notations

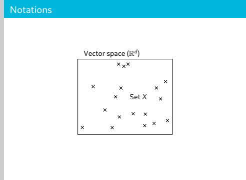




We denote a vector space of real values in dimension *d*. We will consider vectors *x* in this space, and the set big *X* of all such vectors.

Notations





We denote a vector space of real values in dimension d. We will consider vectors x in this space, and the set big X of all such vectors.

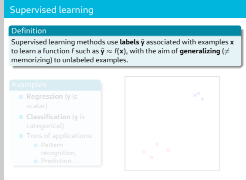
Supervised learning

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** (y is scalar)
- **Classification** (y is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



- We insist here one more time on the fact that learning is not memorizing.We need to say orally that an expert is needed to provide the labels, that is why it is "supervised".
- Give here a few examples of regression tasks (predicting the price of a product in the stock market, the age of a person based on his/her face, ...) and classification tasks (recognizing apples versus oranges.
- When the plot appears, say that for example if we have the points labeled in blue and the points labeled in red, a simple function could be learnt by just dividing the space in two regions.
- However if we present a new point (not part of training) that lies in the red region and is supposed to be "blue", then it means we are not generalizing.
- Finally, we present here another way to "learn", by defining the so-called Voronoi diagram (you can write it on the board), which

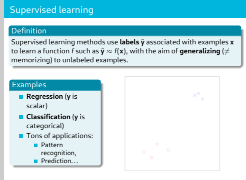
Supervised learning

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** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction...



- We insist here one more time on the fact that learning is not memorizing.We need to say orally that an expert is needed to provide the labels, that is why it is "supervised".
- Give here a few examples of regression tasks (predicting the price of a product in the stock market, the age of a person based on his/her face, ...) and classification tasks (recognizing apples versus oranges.
- When the plot appears, say that for example if we have the points labeled in blue and the points labeled in red, a simple function could be learnt by just dividing the space in two regions.
- However if we present a new point (not part of training) that lies in the red region and is supposed to be "blue", then it means we are not generalizing.
- Finally, we present here another way to "learn", by defining the so-called Voronoi diagram (you can write it on the board), which

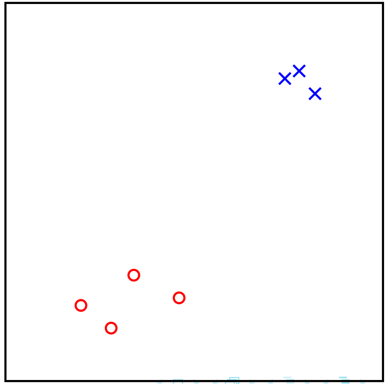
Supervised learning

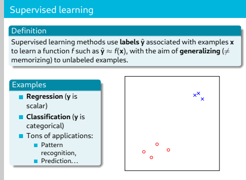
Definition

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** (**y** is scalar)
- **Classification** (**y** is categorical)
- Tons of applications:
  - Pattern recognition,
  - Prediction…





- We insist here one more time on the fact that learning is not memorizing.We need to say orally that an expert is needed to provide the labels, that is why it is "supervised".
- Give here a few examples of regression tasks (predicting the price of a product in the stock market, the age of a person based on his/her face, ...) and classification tasks (recognizing apples versus oranges.
- When the plot appears, say that for example if we have the points labeled in blue and the points labeled in red, a simple function could be learnt by just dividing the space in two regions.
- However if we present a new point (not part of training) that lies in the red region and is supposed to be "blue", then it means we are not generalizing.
- Finally, we present here another way to "learn", by defining the so-called Voronoi diagram (you can write it on the board), which

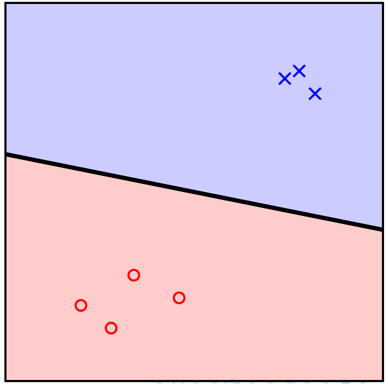
Supervised learning

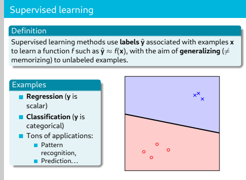
Definition

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…





- We insist here one more time on the fact that learning is not memorizing.We need to say orally that an expert is needed to provide the labels, that is why it is "supervised".
- Give here a few examples of regression tasks (predicting the price of a product in the stock market, the age of a person based on his/her face, ...) and classification tasks (recognizing apples versus oranges.
- When the plot appears, say that for example if we have the points labeled in blue and the points labeled in red, a simple function could be learnt by just dividing the space in two regions.
- However if we present a new point (not part of training) that lies in the red region and is supposed to be "blue", then it means we are not generalizing.
- Finally, we present here another way to "learn", by defining the so-called Voronoi diagram (you can write it on the board), which

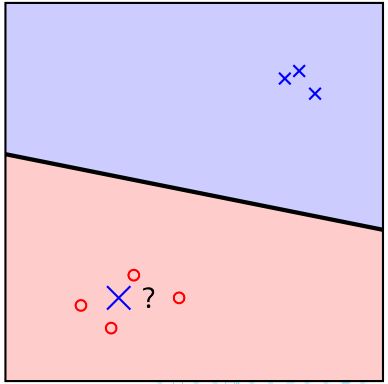
Supervised learning

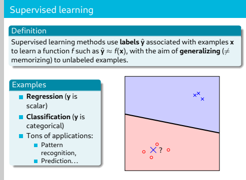
Definition

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...





- We insist here one more time on the fact that learning is not memorizing.We need to say orally that an expert is needed to provide the labels, that is why it is "supervised".
- Give here a few examples of regression tasks (predicting the price of a product in the stock market, the age of a person based on his/her face, ...) and classification tasks (recognizing apples versus oranges.
- When the plot appears, say that for example if we have the points labeled in blue and the points labeled in red, a simple function could be learnt by just dividing the space in two regions.
- However if we present a new point (not part of training) that lies in the red region and is supposed to be "blue", then it means we are not generalizing.
- Finally, we present here another way to "learn", by defining the so-called Voronoi diagram (you can write it on the board), which

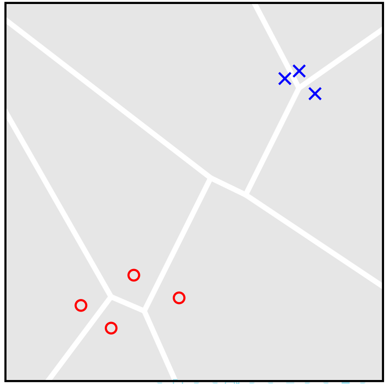
Supervised learning

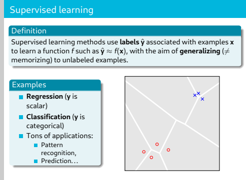
Definition

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…

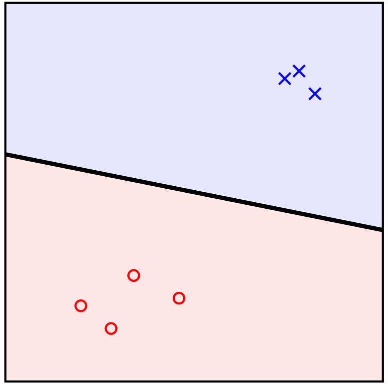


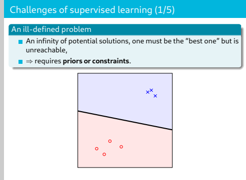


- We insist here one more time on the fact that learning is not memorizing.We need to say orally that an expert is needed to provide the labels, that is why it is "supervised".
- Give here a few examples of regression tasks (predicting the price of a product in the stock market, the age of a person based on his/her face, ...) and classification tasks (recognizing apples versus oranges.
- When the plot appears, say that for example if we have the points labeled in blue and the points labeled in red, a simple function could be learnt by just dividing the space in two regions.
- However if we present a new point (not part of training) that lies in the red region and is supposed to be "blue", then it means we are not generalizing.
- Finally, we present here another way to "learn", by defining the so-called Voronoi diagram (you can write it on the board), which

An ill-defined problem

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



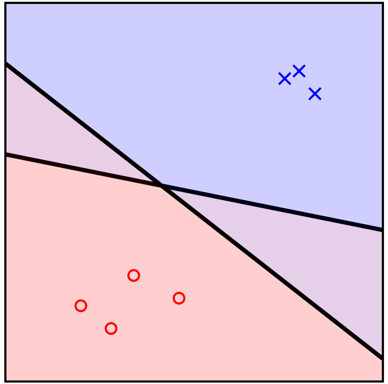


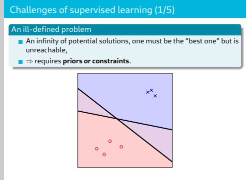
The point here is simply illustrate the fact that the solution is not unique. One way to find a solution that could be "better" than another one is to use prior knowledge or constraints of the problem at hand.

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**.

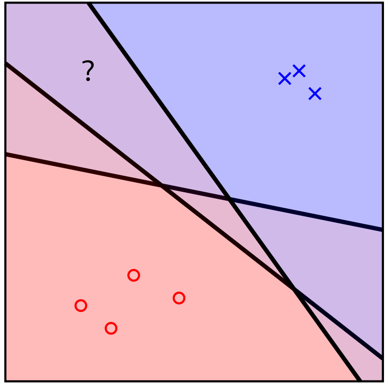


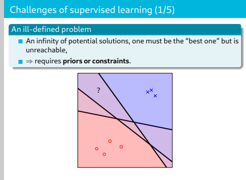


The point here is simply illustrate the fact that the solution is not unique. One way to find a solution that could be "better" than another one is to use prior knowledge or constraints of the problem at hand.

An ill-defined problem

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



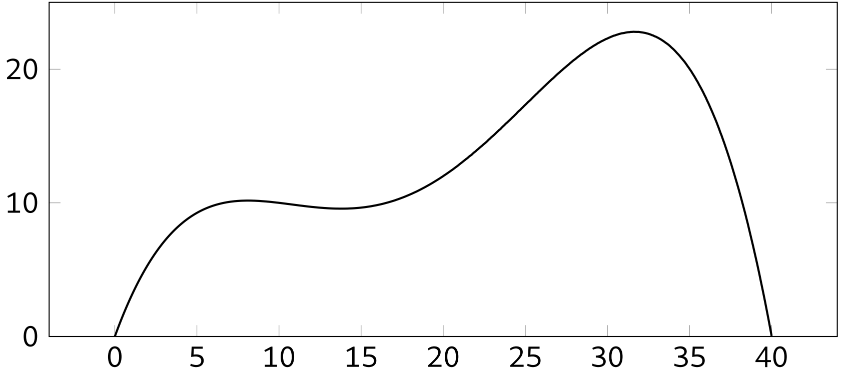


The point here is simply illustrate the fact that the solution is not unique. One way to find a solution that could be "better" than another one is to use prior knowledge or constraints of the problem at hand.

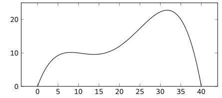
Challenges of supervised learning (2/5)

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.



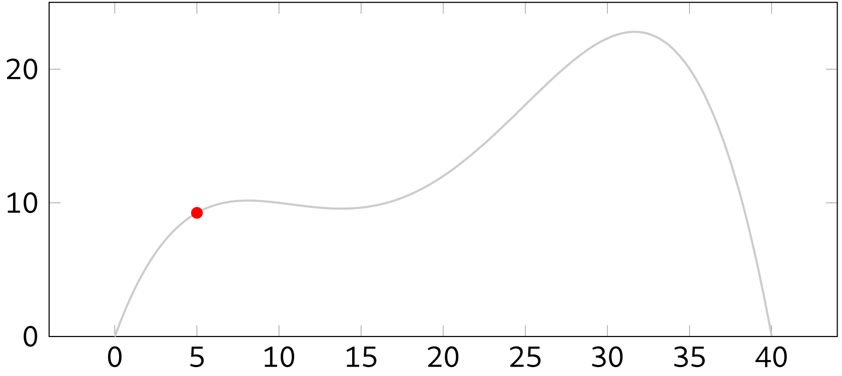
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

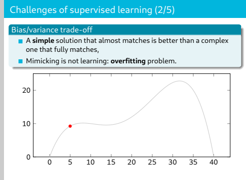
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





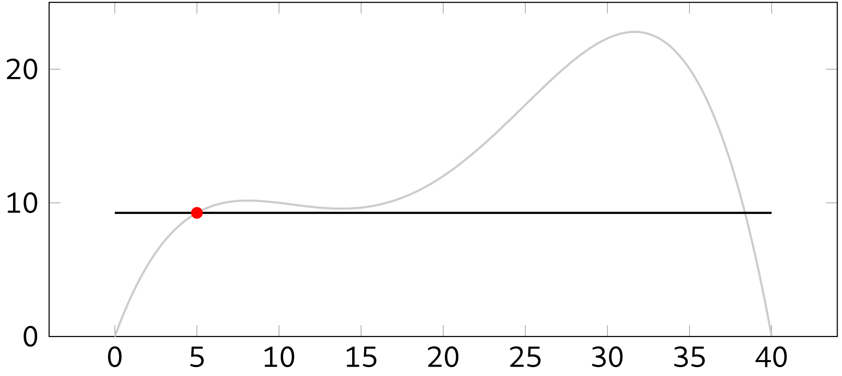
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

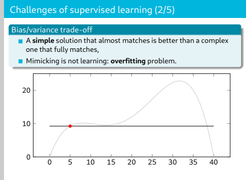
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





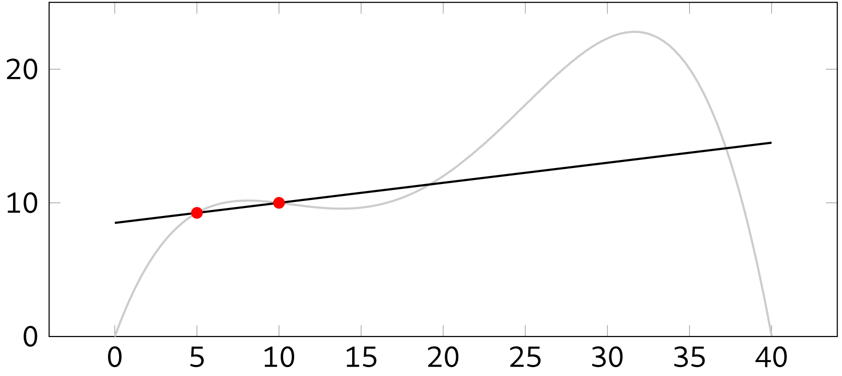
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

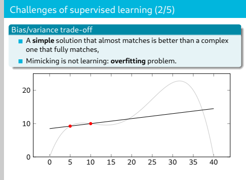
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.



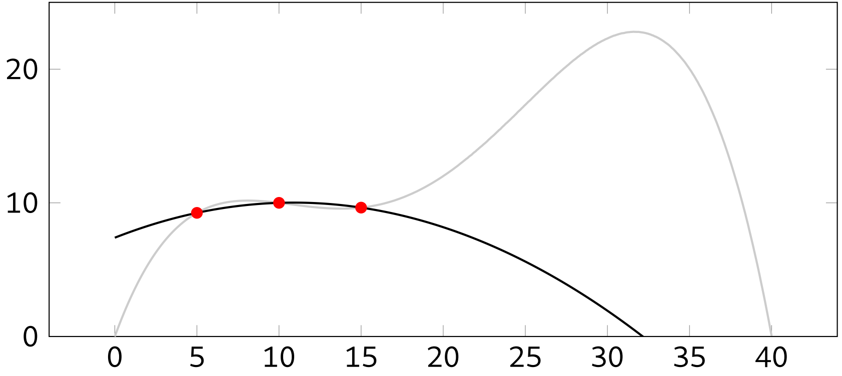


In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

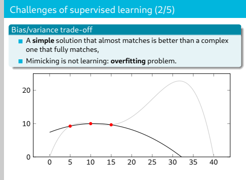
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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)



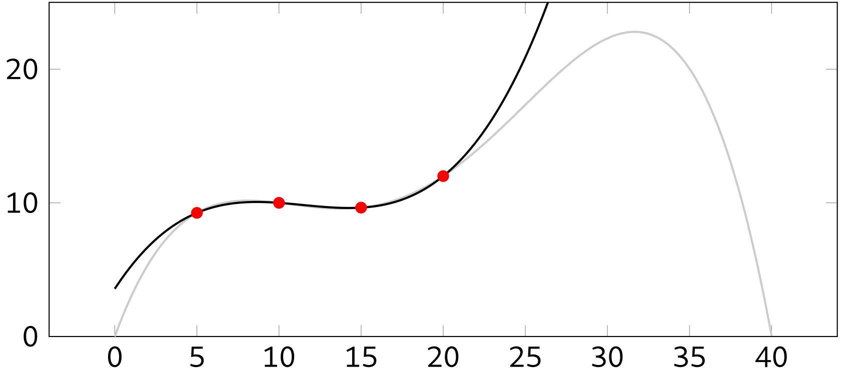
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

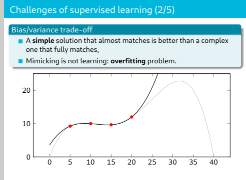
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





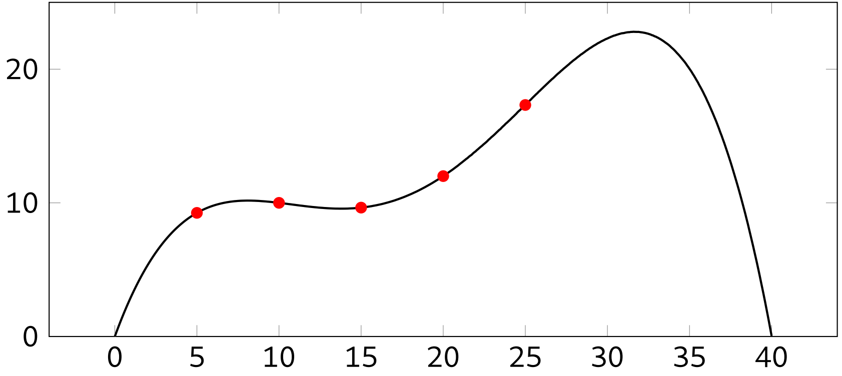
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

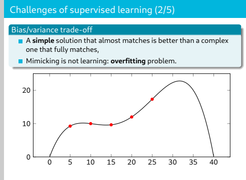
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





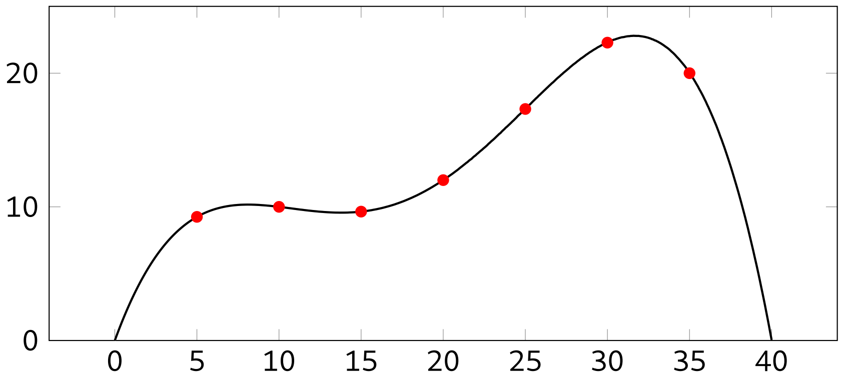
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

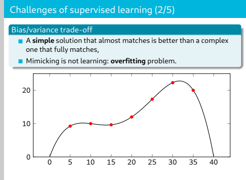
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





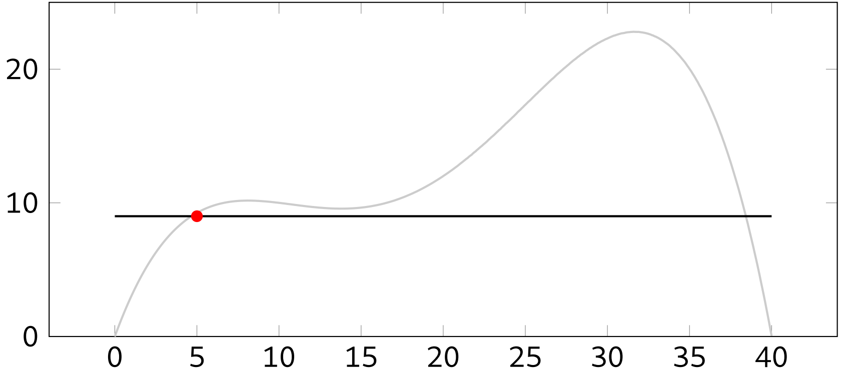
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

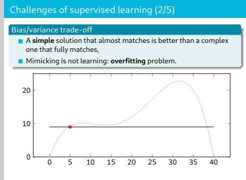
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.



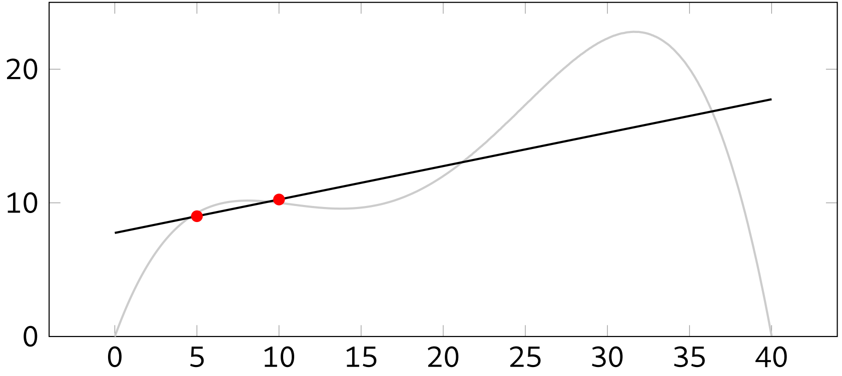


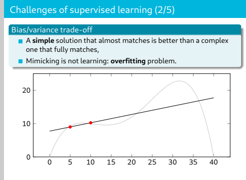
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





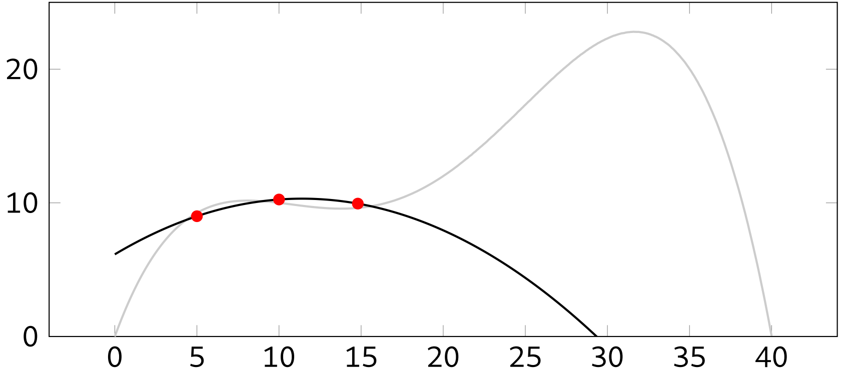
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

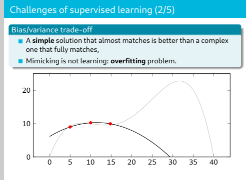
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





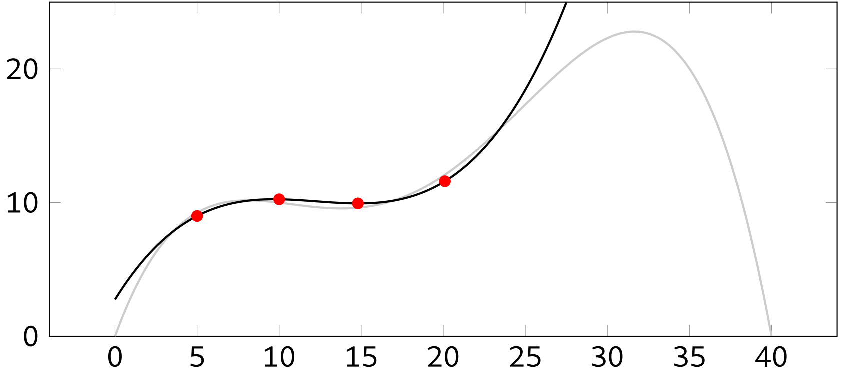
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

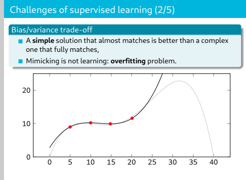
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.



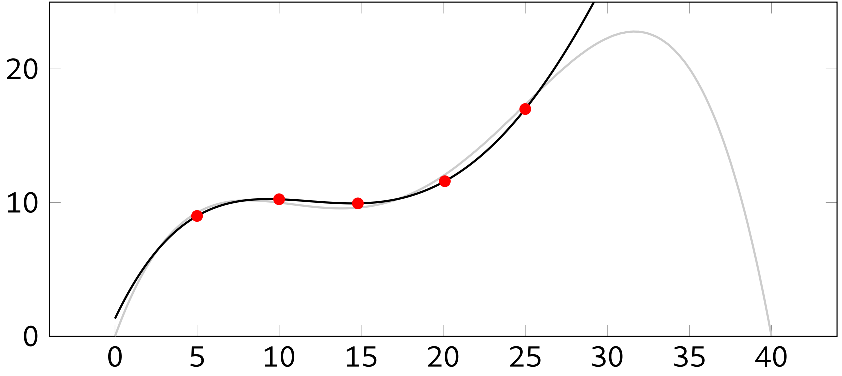


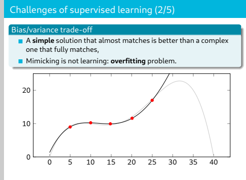
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





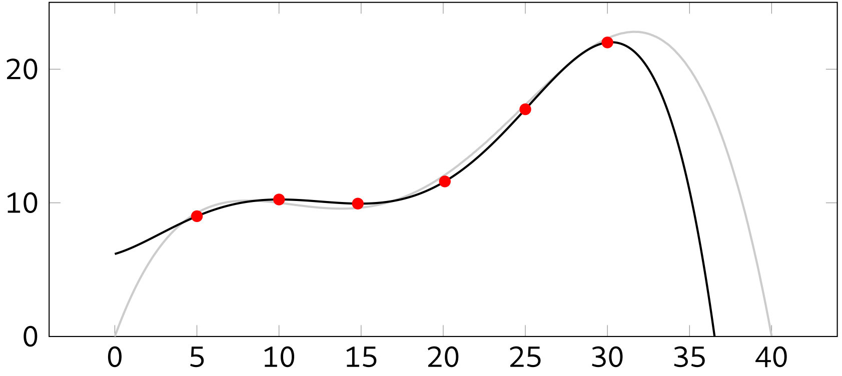
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

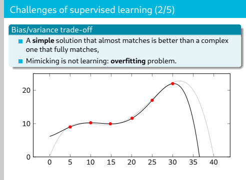
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





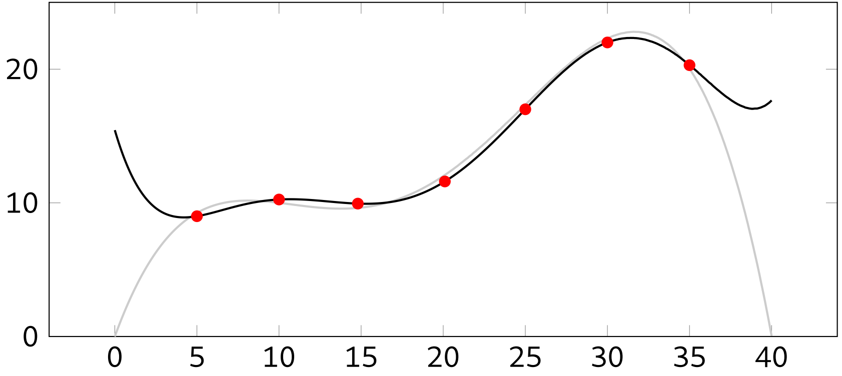
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

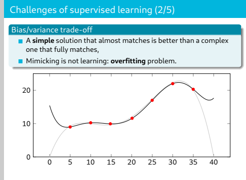
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





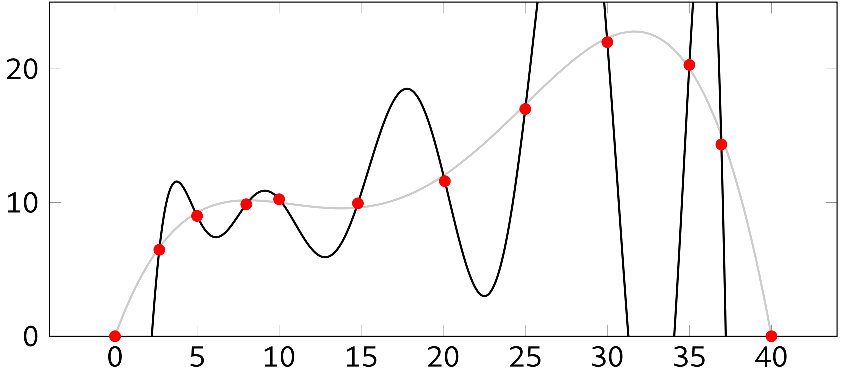
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

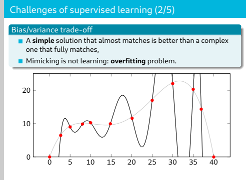
To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

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.





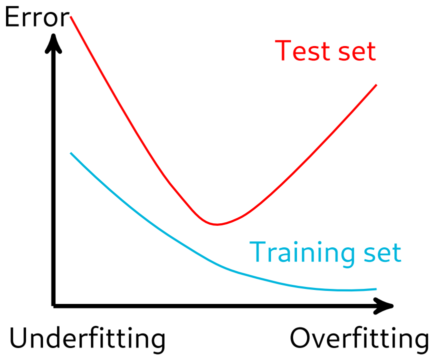
In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

Challenges of supervised learning (2/5)

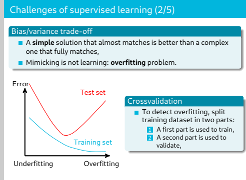
Bias/variance trade-off

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



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,



In the first part, the goal is to show what happens when trying to learn a polinomial function. With one point, we can learn a line, with two points a parabola, etc. If the 5 points are chosen carefully, it is possible to learn something very close to the "true" function. If we take five other points, we might end up with a result that is relatively close. However, if trying to learn polynomials of a higher order, the result can be completely off while the values taken at the training points (red) are correct.

To prevent this, explain what is cross-validation. The diagram on the left shows the error (in regression or classification). The X axis is illustrative, it doesn't correspond to something specific (although one could imagine it to correspond to order of a polynomial, epochs of training a neural net, ...) but it illustrates the situations of underfitting and overfitting.

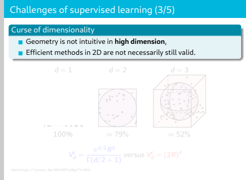
Challenges of supervised learning (3/5)

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

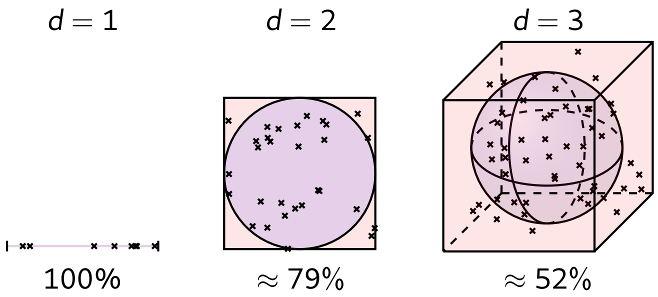


The point here is to show that when the dimension increases, the space tends to be more and more "empty". V_d^s is the volume of the hypershpere, and V_d^c is the volume of the hypercube. The crosses in the different figures are generated by each coordinates following a uniform distribution $\mathcal{U}(0, R)$ (so on average they have a value of $R/2$). When d increases, the ratio between the hypersphere and the hypercube becomes smaller and smaller, so that the majority of the volume of the hypercube lies in the corners. Therefore, the intuitions we have easily in 2D are not valid anymore, so we can imagine why it is difficult to build good classifiers in high dimensions.

Challenges of supervised learning (3/5)

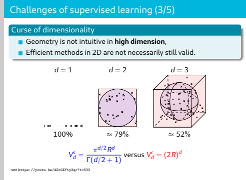
Curse of dimensionality

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



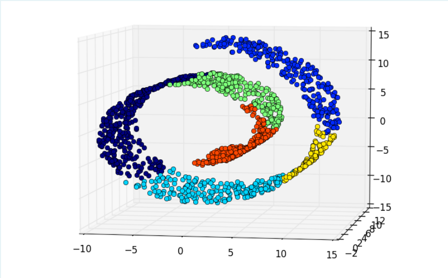
$$V_d^s = \frac{\pi^{d/2} R^d}{\Gamma(d/2+1)} \text{ versus } V_d^c = (2R)^d$$

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

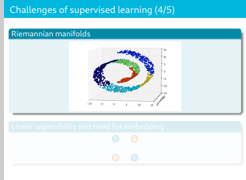


The point here is to show that when the dimension increases, the space tends to be more and more "empty". V_d^s is the volume of the hypershpere, and V_d^c is the volume of the hypercube. The crosses in the different figures are generated by each coordinates following a uniform distribution $\mathcal{U}(0, R)$ (so on average they have a value of $R/2$). When d increases, the ratio between the hypersphere and the hypercube becomes smaller and smaller, so that the majority of the volume of the hypercube lies in the corners. Therefore, the intuitions we have easily in 2D are not valid anymore, so we can imagine why it is difficult to build good classifiers in high dimensions.

Riemannian manifolds



Linear separability and need for embedding

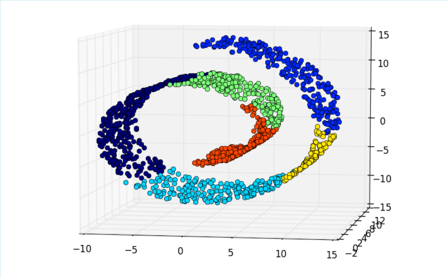


Top part : the point here is to show an example of a dataset in 4D, which is actually much simpler because it is 1D. A nice example to explain the swiss roll is to explain how to roll the cake to make it !

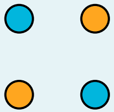
Bottom part : just explain the fact that even in very simple cases, there is no way to find a linear separator.

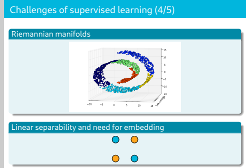
Challenges of supervised learning (4/5)

Riemannian manifolds



Linear separability and need for embedding





Top part : the point here is to show an example of a dataset in 4D, which is actually much simpler because it is 1D. A nice example to explain the swiss roll is to explain how to roll the cake to make it !

Bottom part : just explain the fact that even in very simple cases, there is no way to find a linear separator.

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.

This slide is pretty much self-explanatory. First, the goal is to show that just going through each image is very costly. Second, it is easy to explain why the space of possible functions quickly become so huge that it's not possible to search through it.

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.

This slide is pretty much self-explanatory. First, the goal is to show that just going through each image is very costly. Second, it is easy to explain why the space of possible functions quickly become so huge that it's not possible to search through it.

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.

The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.

The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.

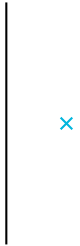
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

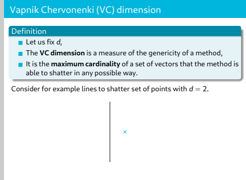
Vapnik Chervonenki (VC) dimension

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 $d = 2$.





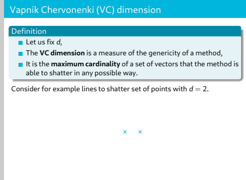
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.



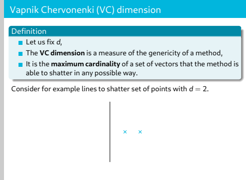
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.

The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.

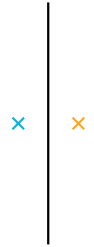
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

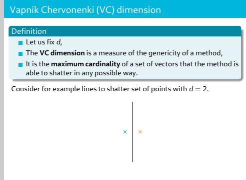
Vapnik Chervonenki (VC) dimension

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 $d = 2$.





The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.

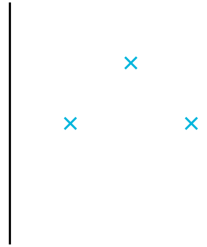
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

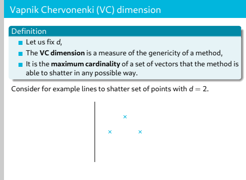
Vapnik Chervonenki (VC) dimension

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 $d = 2$.





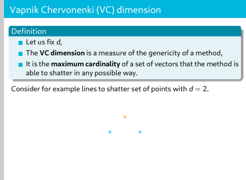
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.



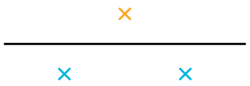
The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

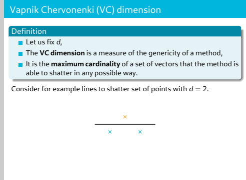
Vapnik Chervonenki (VC) dimension

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 $d = 2$.





The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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 $d = 2$.

The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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.

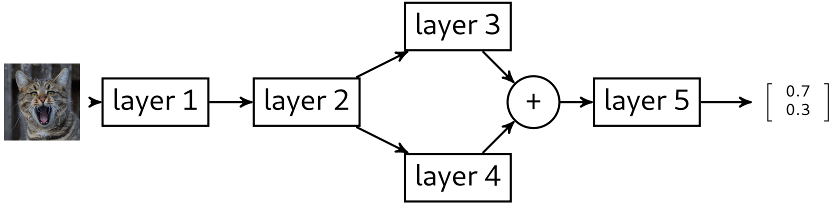
Vapnik Chervonenki (VC) dimension
Definition
Let us fix d,
The VC dimension is a measure of the genericity of a method,
It is the maximum cardinality of a set of vectors that the method is able to shatter in any possible way.
Consider for example lines to shatter set of points with $d = 2$.
VC is 3.

The goal of this slide is to show a theoretical limitation that can be easily be defined and demonstrated. Just comment through the animation.

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**.

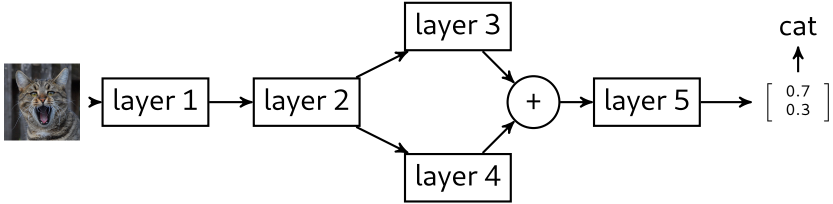


The case of deep learning in classification
Inputs/outputs
Often: inputs are raw signals or feature vectors,
Often: outputs are vectors which highest value indicate the category of the input.

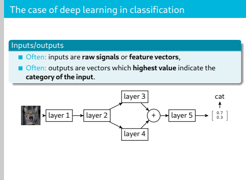
The case of deep learning in classification

Inputs/outputs

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

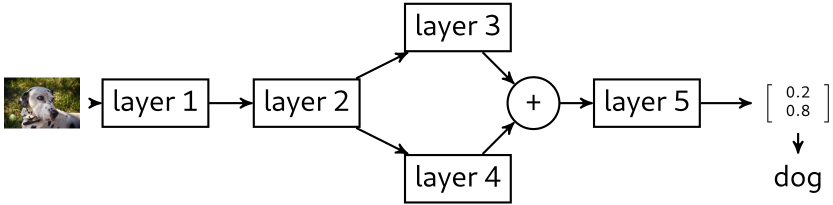


└The case of deep learning in classification



Inputs/outputs

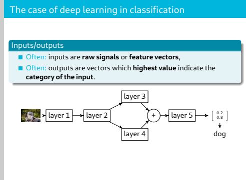
- Often: inputs are **raw signals** or **feature vectors**,
- Often: outputs are vectors which **highest value** indicate the **category of the input**.



2022-02-11

Course 2: Supervised Learning

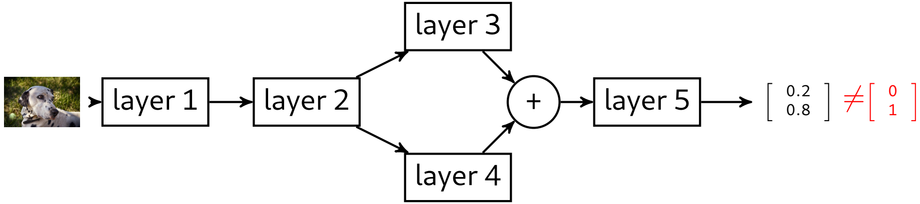
└The case of deep learning in classification



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**.



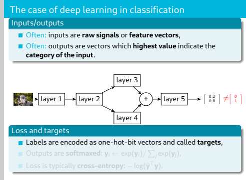
Loss and targets

- 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**: $-\log(\hat{\mathbf{y}}^\top \mathbf{y})$.

2022-02-11

Course 2: Supervised Learning

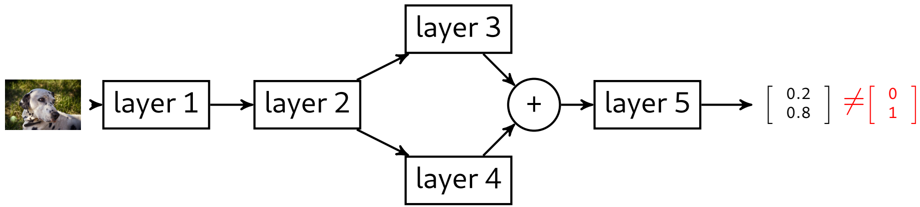
└─The case of deep learning in classification



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

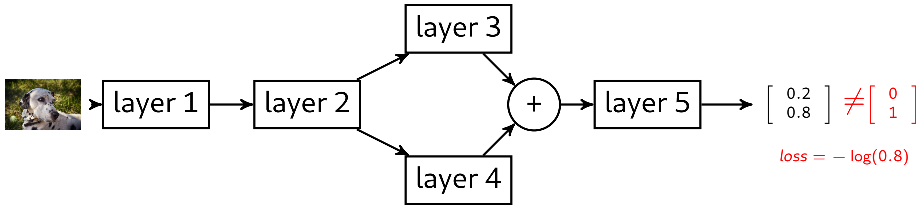
- Labels are encoded as one-hot-bit vectors and called **targets**,
- Outputs are **softmaxed**: $\mathbf{y}_i \leftarrow \exp(\mathbf{y}_i)/\sum_j \exp(\mathbf{y}_j)$,
- Loss is typically **cross-entropy**: $-\log(\hat{\mathbf{y}}^\top \mathbf{y})$.

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**.



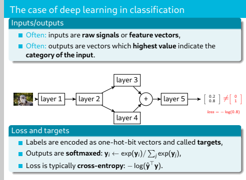
Loss and targets

- 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**: $-\log(\hat{\mathbf{y}}^\top \mathbf{y})$.

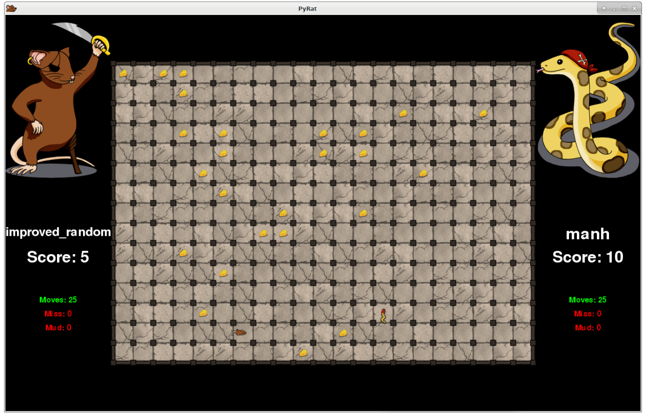
2022-02-11

Course 2: Supervised Learning

└ The case of deep learning in classification

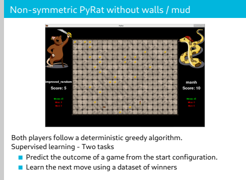


Non-symmetric PyRat without walls / mud



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



Here, we continue the "fil rouge" that will be followed during the whole course.
Ask the students "Can someone remind me what is the simplest deterministic greedy approach that can be taken by a player ?". The answer being "always take the closest piece of cheese".
For the first task :
The start configuration is the location of the pieces of cheese.
There are three possible outcomes : win python, win rat, and draw. So the chance level (expected accuracy of a random classifier) is 30 percent.
For the second task : There are four possible moves.

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.

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.

Here, it is important to tell them that we expect them to think about interpreting the result on the pyrat datasets. In addition, there are definitions in the Lab Session (accuracy, precision, recall and f1 score) that are important to learn.

IMPORTANT : tell them to remember that they have COMPLETE CONTROL on the generation of the pyrat datasets (size of the maze, number of pieces of cheese, ...). So they can use that to explore the problem.