Classification: Part II

Statistical Analysis and Document Mining

Spring 2021

Vasilii Feofanov

Université Grenoble Alpes

vasilii.feofanov@univ-grenoble-alpes.fr

Outline

Artificial Intelligence (AI)

- AI is an active field of study these days that explores different ways to develop machines capable to learn and solve problems.

Artificial Intelligence (AI)

- AI is an active field of study these days that explores different ways to develop machines capable to learn and solve problems.
- A great introduction to the history of the artificial intelligence you can find in the following video.

Figure: https://www.youtube.com/watch?v=8FHBh_OmdsM

AI and Human Brain Study

- One of the directions in AI is to mimic human intelligence: how human brain perceives information and make decisions.

AI and Human Brain Study

- One of the directions in AI is to mimic human intelligence: how human brain perceives information and make decisions.
- Promising deep neural networks were originally inspired by biological neurons, electrically active cells.

AI and Human Brain Study

- One of the directions in AI is to mimic human intelligence: how human brain perceives information and make decisions.
- Promising deep neural networks were originally inspired by biological neurons, electrically active cells.
- The brain has complex networks of connected to each other neurons that can receive, store or propagate information.

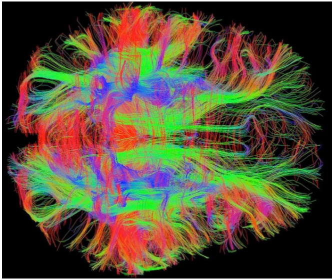
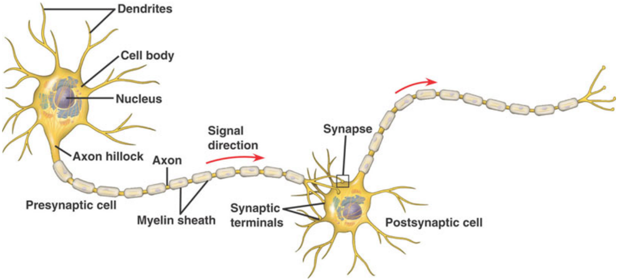


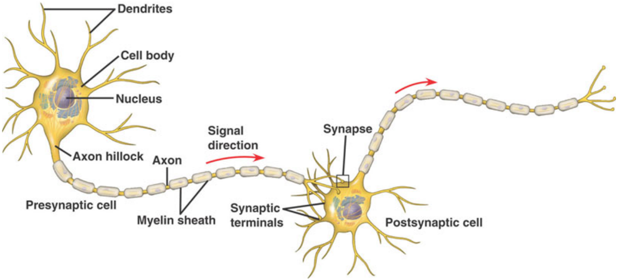
Figure: A map of nerve fibers (axons) in the human brain.

Biological Neuron



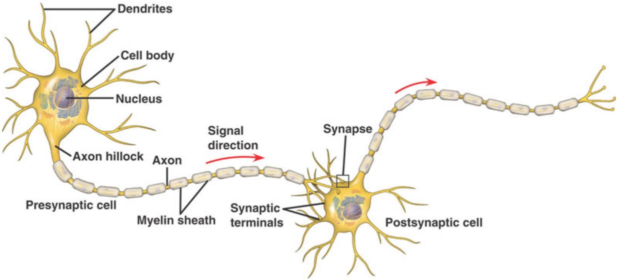
- One neuron sends a signal to another one through an axon to a synapse of a dendrite.

Biological Neuron



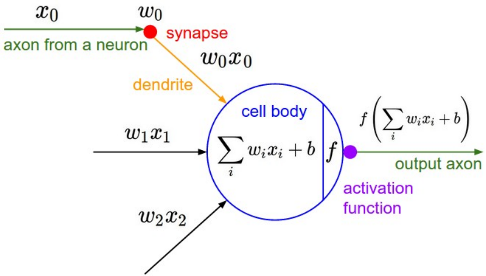
- One neuron sends a signal to another one through an axon to a synapse of a dendrite.
- The synapse process the information and can regularize its amplitude and its frequency.

Biological Neuron



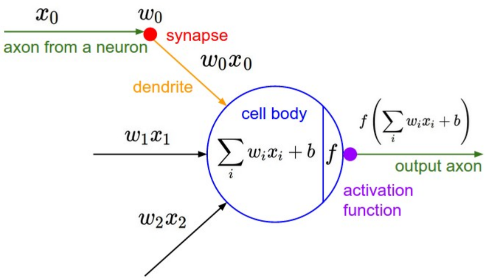
- One neuron sends a signal to another one through an axon to a synapse of a dendrite.
- The synapse process the information and can regularize its amplitude and its frequency.
- The cell body receives the processed information via the dendrite. The neuron decides then whether to fire it further.

Neuron Model (MuCulloch & Pitts, 1943)



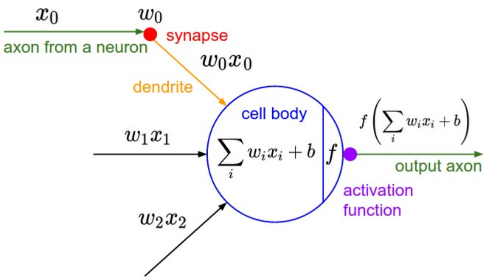
- In 1943, a simplified mathematical model of the neuron was proposed.

Neuron Model (MuCulloch & Pitts, 1943)



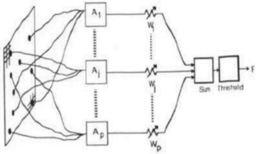
- In 1943, a simplified mathematical model of the neuron was proposed.
- Neurons $0, 1, 2 \ldots$ send information $x_0, x_1, x_2 \ldots$. The synapse determines the strength (weight) of information.

Neuron Model (MuCulloch & Pitts, 1943)



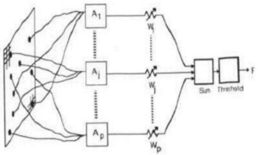
- In 1943, a simplified mathematical model of the neuron was proposed.
- Neurons $0, 1, 2 \ldots$ send information $x_0, x_1, x_2 \ldots$. The synapse determines the strength (weight) of information.
- The cell body sums up all information and decides via activation function whether to propagate it further.

Perceptron Machine (Rosenblatt, 1958)



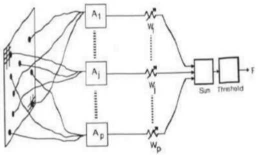
- Using the neuron model, Rosenblatt implemented an algorithm in hardware for image recognition of geometric figures.

Perceptron Machine (Rosenblatt, 1958)



- Using the neuron model, Rosenblatt implemented an algorithm in hardware for image recognition of geometric figures.
- An array of 20x20 photocells are randomly connected to form *features* (input neurons).

Perceptron Machine (Rosenblatt, 1958)



- Using the neuron model, Rosenblatt implemented an algorithm in hardware for image recognition of geometric figures.
- An array of 20x20 photocells are randomly connected to form *features* (input neurons).
- The weights represent the resistance of electric motors and are updated during learning by purely electric process.

- **Summation:** From an image we extract features $\mathbf{x} = (x_1, \ldots, x_d)$ and linearly combine them inside the decision neuron.

$$h_{\mathbf{w}}(\mathbf{x}) := \langle \mathbf{w}, \mathbf{x} \rangle + w_0 = \sum_{j=1}^{d} w_j x_j + w_0.$$

[1] We consider the binary classification framework, i.e. $\mathcal{Y} = \{-1, +1\}$.

Perceptron in a Nutshell

- **Summation:** From an image we extract features $\mathbf{x} = (x_1, \ldots, x_d)$ and linearly combine them inside the decision neuron.

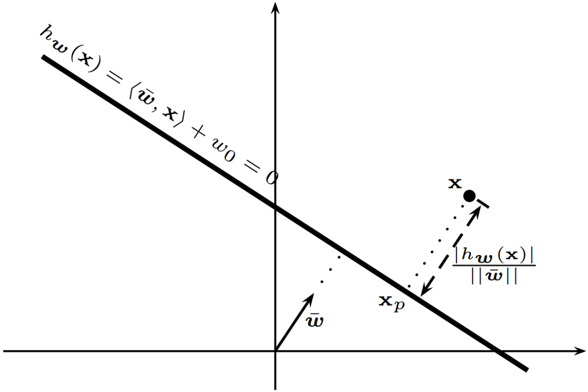
$$h_{\mathbf{w}}(\mathbf{x}) := \langle \mathbf{w}, \mathbf{x} \rangle + w_0 = \sum_{j=1}^{d} w_j x_j + w_0.$$

- **Activation:** We compare the sign of the true label y and the output $h_{\mathbf{w}}(\mathbf{x})$. If there is no mistake ($y h_{\mathbf{w}}(\mathbf{x}) \geq 0$), the activation is 1, otherwise, it is -1[1].

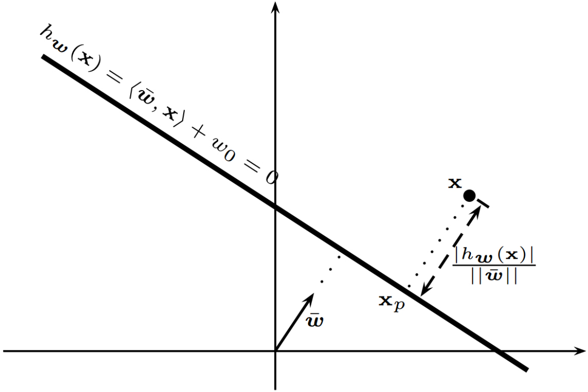
$$a(\mathbf{x}) = \begin{cases} +1, & \text{if } y h_{\mathbf{w}}(\mathbf{x}) \geq 0; \\ -1, & \text{if } y h_{\mathbf{w}}(\mathbf{x}) < 0. \end{cases}$$

[1] We consider the binary classification framework, i.e. $\mathcal{Y} = \{-1, +1\}$.

We want to find parameters $(w_0, \mathbf{w})$ such that the distance between the misclassified examples and the decision boundary is minimised.



By this, we push the boundary away from correctly classified examples that will have high margins. In this case, the margin $\frac{|h_{\mathbf{w}}(\mathbf{x})|}{\|\mathbf{w}\|}$ represents the confidence score.



Perceptron Error Minimisation

- We would like to minimise the *perceptron error*:

$$\hat{L}(\mathbf{w}, w_0) = -\sum_{i' \in \mathcal{I}} y_{i'}(\langle \mathbf{w}, \mathbf{x}_{i'} \rangle + w_0).$$

Perceptron Error Minimisation

- We would like to minimise the *perceptron error*:

$$\hat{L}(\mathbf{w}, w_0) = -\sum_{i' \in \mathcal{I}} y_{i'}(\langle \mathbf{w}, \mathbf{x}_{i'} \rangle + w_0).$$

- If we take derivatives with respect to the weights:

$$\frac{\partial \hat{L}(\mathbf{w})}{\partial w_0} = -\sum_{i' \in \mathcal{I}} y_{i'},$$

$$\nabla \hat{L}(\mathbf{w}) = -\sum_{i' \in \mathcal{I}} y_{i'} \mathbf{x}_{i'}.$$

Perceptron Error Minimisation

- We would like to minimise the *perceptron error*:

$$\hat{L}(\mathbf{w}, w_0) = -\sum_{i' \in \mathcal{I}} y_{i'}(\langle \mathbf{w}, \mathbf{x}_{i'} \rangle + w_0).$$

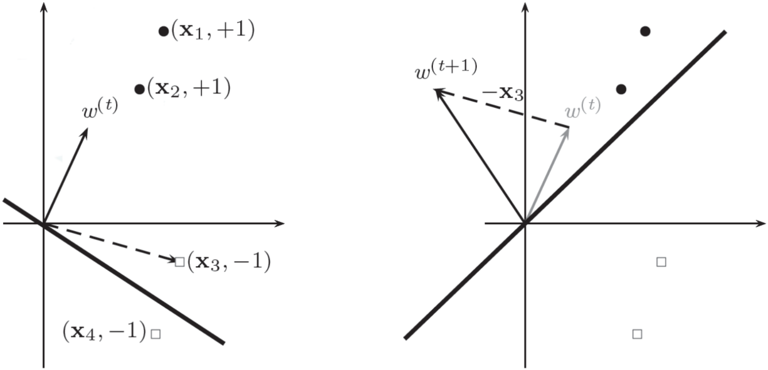
- If we take derivatives with respect to the weights:

$$\frac{\partial \hat{L}(\mathbf{w})}{\partial w_0} = -\sum_{i' \in \mathcal{I}} y_{i'},$$

$$\nabla \hat{L}(\mathbf{w}) = -\sum_{i' \in \mathcal{I}} y_{i'} \mathbf{x}_{i'}.$$

- Minimisation of the perceptron error is done by stochastic gradient descent algorithm:

$$\text{if } y(\langle \mathbf{w}, \mathbf{x} \rangle + w_0) < 0, \text{then } \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} \leftarrow \begin{pmatrix} w_0 \\ \mathbf{w} \end{pmatrix} + \eta \begin{pmatrix} y \\ y\mathbf{x} \end{pmatrix}.$$


$\bullet(\mathbf{x}_1, +1)$
$\bullet(\mathbf{x}_2, +1)$
$w^{(t)}$
$\square(\mathbf{x}_3, -1)$
$(\mathbf{x}_4, -1)\square$
$w^{(t+1)}$
$-\mathbf{x}_3$
$w^{(t)}$

Algorithm Perceptron

Input: Training set $\mathrm{S} = \{\mathbf{x}_i, y_i\}_{i=1}^n$.
Max number of iterations T.

Initialisation: Weights $\mathbf{w}^{(0)} \leftarrow \mathbf{0}, w_0 \leftarrow 0$;
Counter $t \leftarrow 0$, learning rate $\eta > 0$.

repeat

Choose randomly an example $(\mathbf{x}^{(t)}, y^{(t)}) \in \mathrm{S}$

if $y \cdot (\langle \mathbf{w}^{(t)}, \mathbf{x}^{(t)} \rangle + w_0) < 0$ **then**

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta\, y^{(t)}$$
$$\mathbf{w}^{(t+1)} \leftarrow \mathbf{w}^{(t)} + \eta\, y^{(t)}\, \mathbf{x}^{(t)}$$

end if

$t \leftarrow t + 1$

until $t > T$

Output: $(w_0^{(T)}, \mathbf{w}^{(T)})$.

Perceptron: Discussion

- What is the time complexity during training phase?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?
- How to tune learning rate η?

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?
- How to tune learning rate η?
- Does the algorithm converge to an exact solution?

Perceptron: Discussion

- What is the time complexity during training phase?
- What is the time complexity to predict a label for new $\mathbf{x}$?
- How to tune learning rate η?
- Does the algorithm converge to an exact solution?
- Under what condition the algorithm does not converge?

AI Winter

- Large success of the perceptron were followed by public disappointment and a sharp decline of AI funding.
- Minsky and Papert showed that the perceptron is not able to approximate the XOR operation, a non-linear problem.

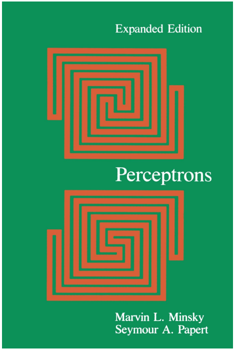
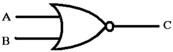


Figure: (Minsky & Papert, 1969)

Figure: Minsky and Papert in 1971.

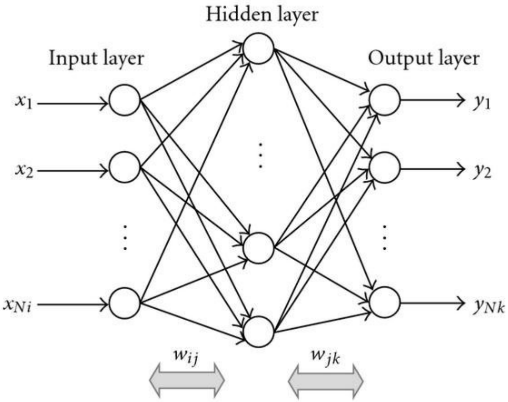


A	B	C
0	0	1
0	1	0
1	0	0
1	1	0

Figure: The XOR operation.

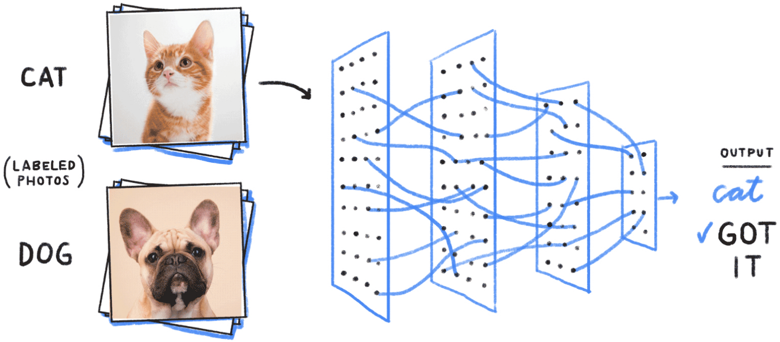
Backpropagation Learning (Rumelhart et al., 1985)

- In the 1980's, the perceptron returned as *multi-layer*: a hidden layer of neurons connects the input and the output layers.
- The weights are updated layer by layer propagating errors back through the whole network (*backpropagation*).



UGA Université Grenoble Alpes

- Nowadays, with large computational resources, the neural networks has become *deep* with a large number of hidden layers of different functionality.
- The goal is not only prediction itself, but also learning new appropriate data representation for better prediction.

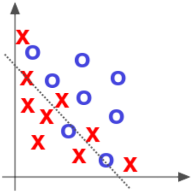


Outline

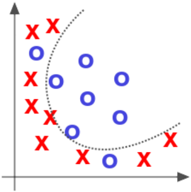
Emerging In Practice Questions

- How to select a learning model with the best accuracy score[2] on *unseen examples*?
- How to tune hyperparameters of an algorithm?
- Are there metrics that take into account imbalance in classes? Importance of one class over another one?

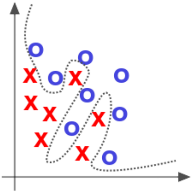
[2] The accuracy score is the proportion of correctly predicted examples.



(a) Undefitting



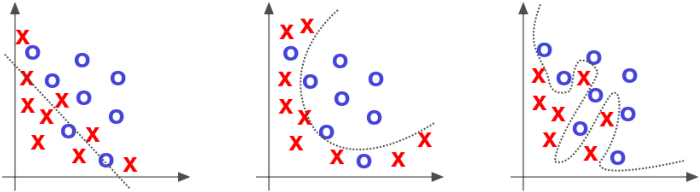
(b) Appropriate



(c) Overfitting

- **Underfitting:** The classifier does not fit data and has a large error value.

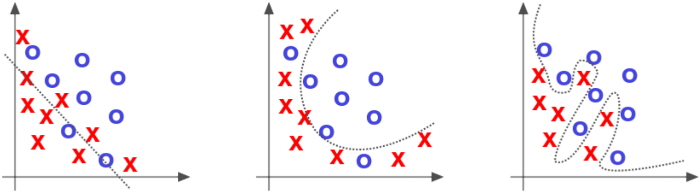
Performance of an Algorithm



(a) Undefitting (b) Appropriate (c) Overfitting

- **Underfitting:** The classifier does not fit data and has a large error value.
- **Overfitting:** Being perfect on training data, the classifier has poor generalization making lots of mistakes on new examples.

Performance of an Algorithm



(a) Undefitting (b) Appropriate (c) Overfitting

- **Underfitting:** The classifier does not fit data and has a large error value.
- **Overfitting:** Being perfect on training data, the classifier has poor generalization making lots of mistakes on new examples.
- **What we want:** The classifier does not take into account noise in the training data and approximates well the true boundary.

Misclassification Error Estimation

Given the data sample $S = \{\mathbf{x}_i, y_i\}_{i=1}^n$, are the following estimates of the misclassification error good (in terms of bias/variance)?

- Compute error on the same examples the model was trained?

Misclassification Error Estimation

Given the data sample $S = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, are the following estimates of the misclassification error good (in terms of bias/variance)?

- Compute error on the same examples the model was trained?
  - No! Among various models we will choose the one that is optimistically biased. This is exactly *overfitting*.
- Split the data into two parts: one is used for training, error is evaluated on the examples of the second one?

Misclassification Error Estimation

Given the data sample $S = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, are the following estimates of the misclassification error good (in terms of bias/variance)?

- Compute error on the same examples the model was trained?
  - No! Among various models we will choose the one that is optimistically biased. This is exactly *overfitting*.
- Split the data into two parts: one is used for training, error is evaluated on the examples of the second one?
  - This is appropriate. The method is widely used when the number of examples n is large. However, when n is small, we cherish every example we have due to the risk of *underfitting*.
- Data is divided by k folds, one fold is for evaluation and $k-1$ rest are for training; the errors are averaged over k rounds?

Misclassification Error Estimation

Given the data sample $S = \{\mathbf{x}_i, y_i\}_{i=1}^{n}$, are the following estimates of the misclassification error good (in terms of bias/variance)?

- Compute error on the same examples the model was trained?
  - No! Among various models we will choose the one that is optimistically biased. This is exactly *overfitting*.
- Split the data into two parts: one is used for training, error is evaluated on the examples of the second one?
  - This is appropriate. The method is widely used when the number of examples n is large. However, when n is small, we cherish every example we have due to the risk of *underfitting*.
- Data is divided by k folds, one fold is for evaluation and $k-1$ rest are for training; the errors are averaged over k rounds?
  - *Cross-validation* (CV) is the most popular approach. It reduces not only the bias, but also the data variability used for training.

CV: How Many Folds to Choose?

- The smaller k, the less number of examples we use for training. If k is large, we train too many classifiers.
- For medium/large sample sizes, usual choices of k are 5 or 10.
- If the sample size n is really small, the popular paradigm is to take $k = n$, a.k.a. the *leave-one-out* (LOO) cross-validation.

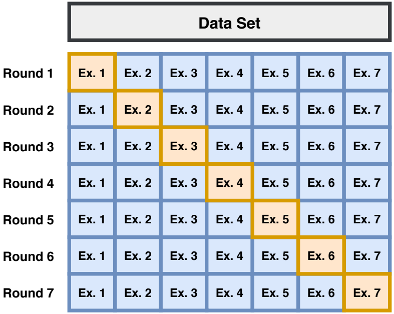


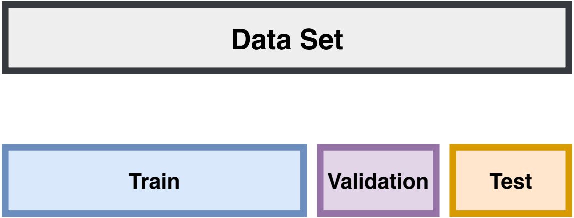
Figure: LOO for a data set with 7 examples (blue = train, orange = test).

- The CV can be also used to find the best hyperparameter (k for knn, η for perceptron, etc.).
- We define a grid of values (e.g. $\eta \in \{0.01, 0.1, 1\}$) and compute the CV score for each parameter value.
- Then, we select a model with the best CV-score.

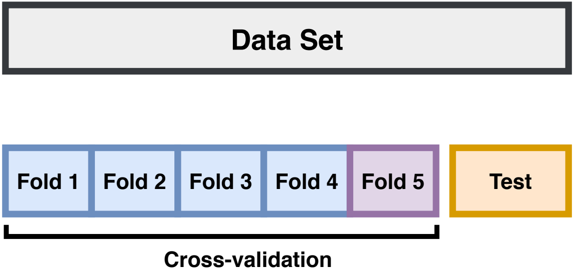
Hyperparameter Tuning

- The CV can be also used to find the best hyperparameter (k for knn, η for perceptron, etc.).
- We define a grid of values (e.g. $\eta \in \{0.01, 0.1, 1\}$) and compute the CV score for each parameter value.
- Then, we select a model with the best CV-score.

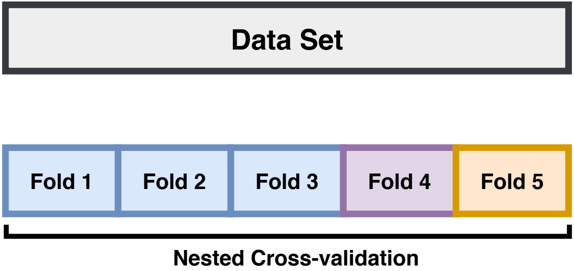
However, there is still a chance of overfitting. If you wanna be on the safe side, one of the following strategy can be applied.



- *Strategy 1:* Split into three parts: learn models on Train, choose the best hyperparameter on Validation, estimate the performance on Test.



- *Strategy 2:* Split into two parts: The first part is used to perform CV. By the cv-score we choose the best hyper-parameter, which is used then to learn a final model using all examples from the first part. Finally, the performance is estimated on Test.



- *Strategy 3:* We average the effect of choosing the best hyperparameter: at each round, $k-2$ folds are used for learning, orange fold is for estimating performance of the hyperparameter chosen on the purple fold. The final cv-score is the average performance over k orange folds.

Confusion Matrix

- To see the difference between the predicted and the true outputs in more detail, the confusion matrix can be used.
- Its (i, j)-entry is the number of examples that were assigned to a class j given the true label is i.

Prediction

Real

$n = 200$	-1	$+1$
-1	65	5
$+1$	7	123

Confusion Matrix

- To see the difference between the predicted and the true outputs in more detail, the confusion matrix can be used.
- Its (i, j)-entry is the number of examples that were assigned to a class j given the true label is i.
- In the binary classification, the entries are named as follows: True Negative (TN), False Positive (FP), False Negative (FN), True Positive (TP).

		Prediction	
		-1	$+1$
Real	-1	TN	FP
	$+1$	FN	TP

- The class imbalance is an often situation in real applications. For instance, the number of patients without a disease (-1) can be much larger than with it (+1).
- In this case, the accuracy score might be not the best choice to evaluate the prediction quality.

Class Imbalance

- The class imbalance is an often situation in real applications. For instance, the number of patients without a disease (-1) can be much larger than with it $(+1)$.
- In this case, the accuracy score might be not the best choice to evaluate the prediction quality.
- Example: Classifier 2 is better in terms of accuracy. However, we would like to choose Classifier 1, because it predicts less frequently "no disease" given a diseased patient (small `FN`).

Prediction

Real	$n = 200$	-1	$+1$
	-1	150	20
	$+1$	3	27

(a) Classifier 1.

Prediction

Real	$n = 200$	-1	$+1$
	-1	170	0
	$+1$	20	10

(b) Classifier 2.

Figure: Confusion matrices of two classifiers.

Sensitivity and Specificity

- If we want to minimize $\mathtt{FN}$, the *sensivity* (a.k.a. recall) score can be used to choose the best model:

$$\mathtt{SEN} = \frac{\mathtt{TP}}{\mathtt{TP} + \mathtt{FN}}.$$

- If we want to minimize $\mathtt{FP}$, the *specificity* score can be used:

$$\mathtt{SPE} = \frac{\mathtt{TN}}{\mathtt{TN} + \mathtt{FP}}.$$

- Finally, with the *balanced accuracy* we minimize both $\mathtt{FP}$ and $\mathtt{FN}$ discarding the effect of the class distribution:

$$\mathtt{BACC} = \frac{\mathtt{SPE} + \mathtt{SEN}}{2}.$$

To see a simple example of classfication in R, please go to Chamilo:

`Lectures` → `Classification Part II` → `R script with a classification example`