Introduction to ML Based on First Chapter of Géron’s Book

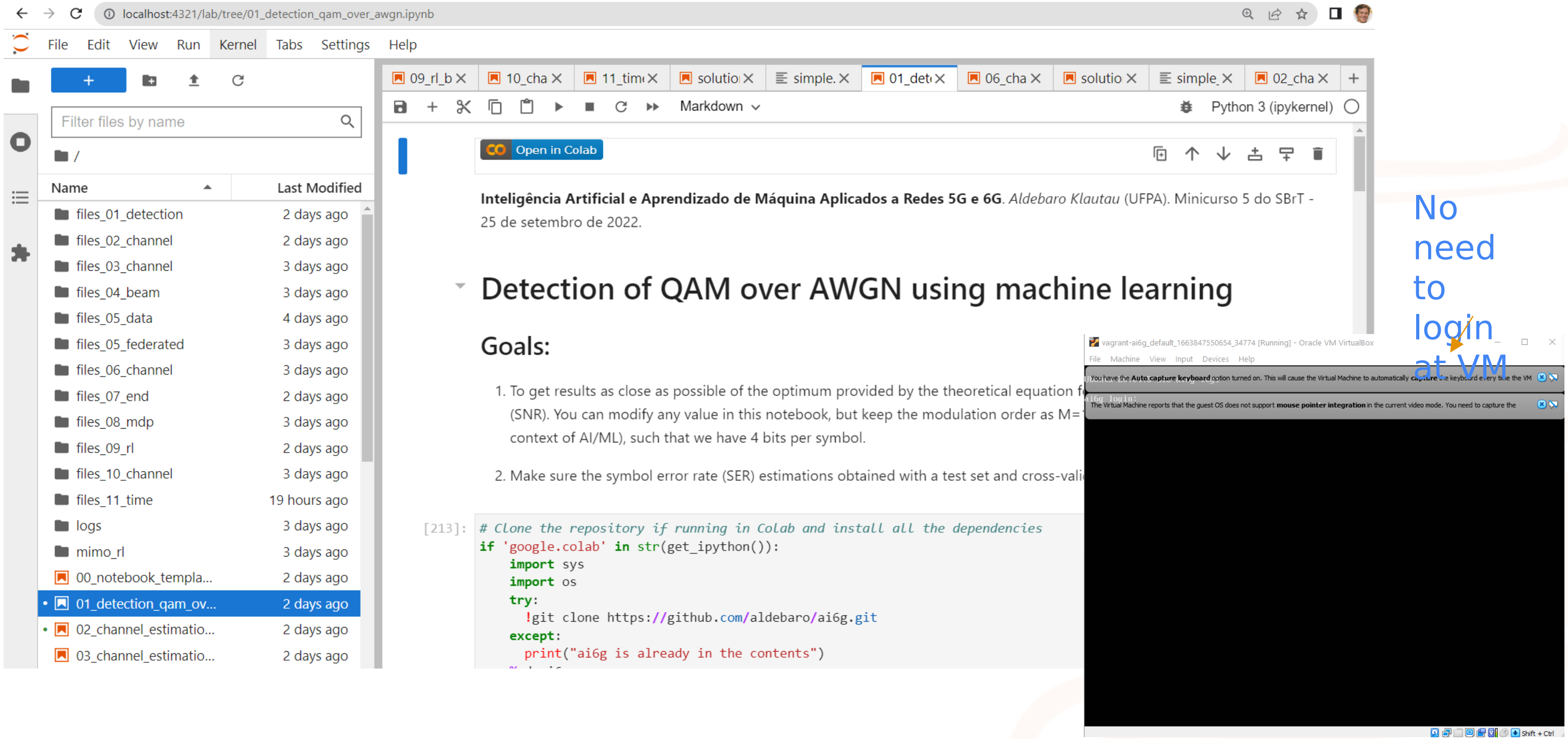
Aldebaro Klautau
Federal University of Pará (UFPA) / LASSE

Computational Intelligence Class

November 07, 2024

Extra!

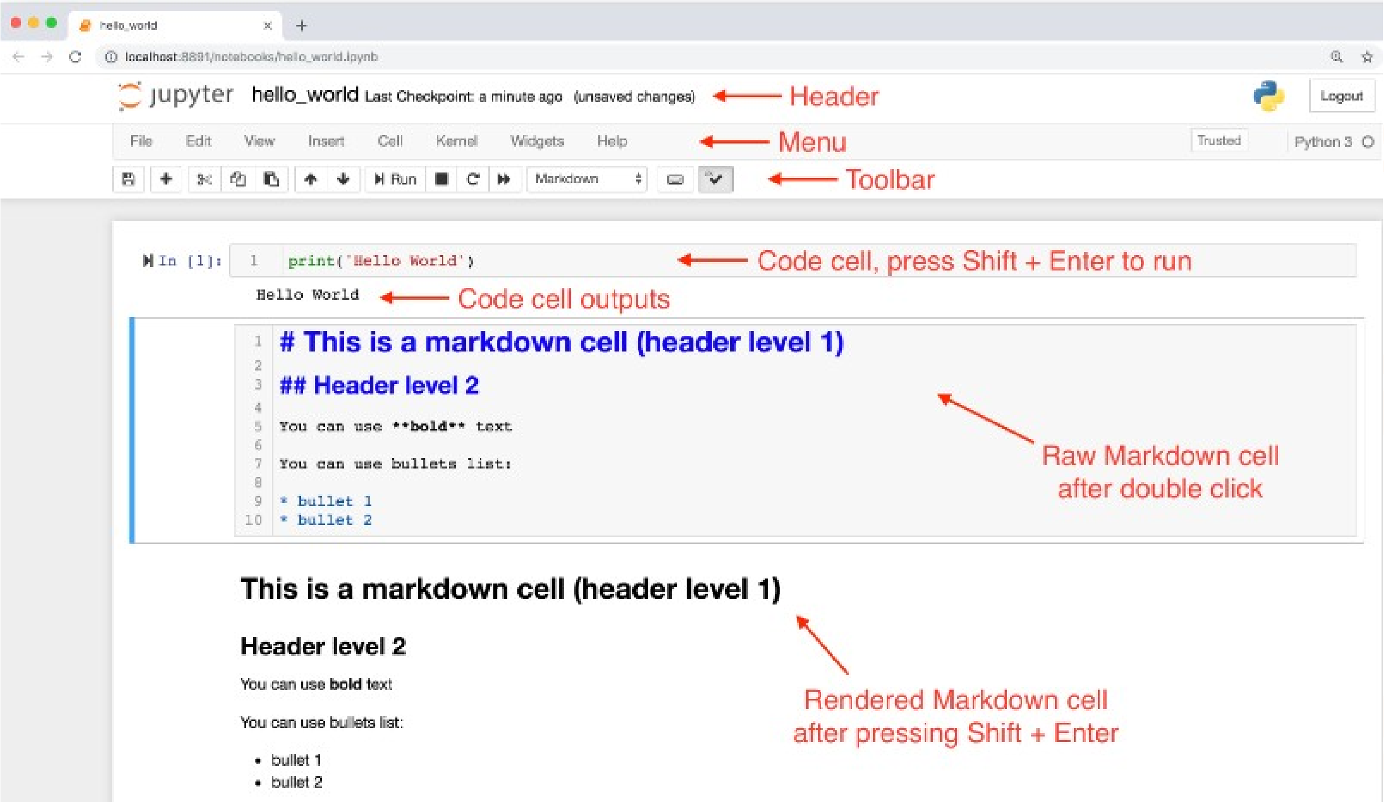
*Introduction to Jupyter notebooks



No
need
to
login
at VM

https://pythonnumericalmethods.berkeley.edu/notebooks/chapter01.04-Introduction-to-Jupyter-Notebook.html

*Introduction to Jupyter notebooks (2)



https://pythonnumericalmethods.berkeley.edu/notebooks/chapter01.04-Introduction-to-Jupyter-Notebook.html

Extra!

*Jupyter shortcuts, magic and shell commands

LASSE

Two different keyboard input modes:

- **Edit** mode: type code or text into a cell.
  Green cell border
- **Command** mode: notebook level commands. Gray cell border with a blue left margin

Shortcuts that work in both edit and command modes:

Shift + Enter - run the current cell, select below
Ctrl + Enter - run selected cells
Alt + Enter - run the current cell, insert below
Ctrl + S - save and checkpoint

Magic commands:

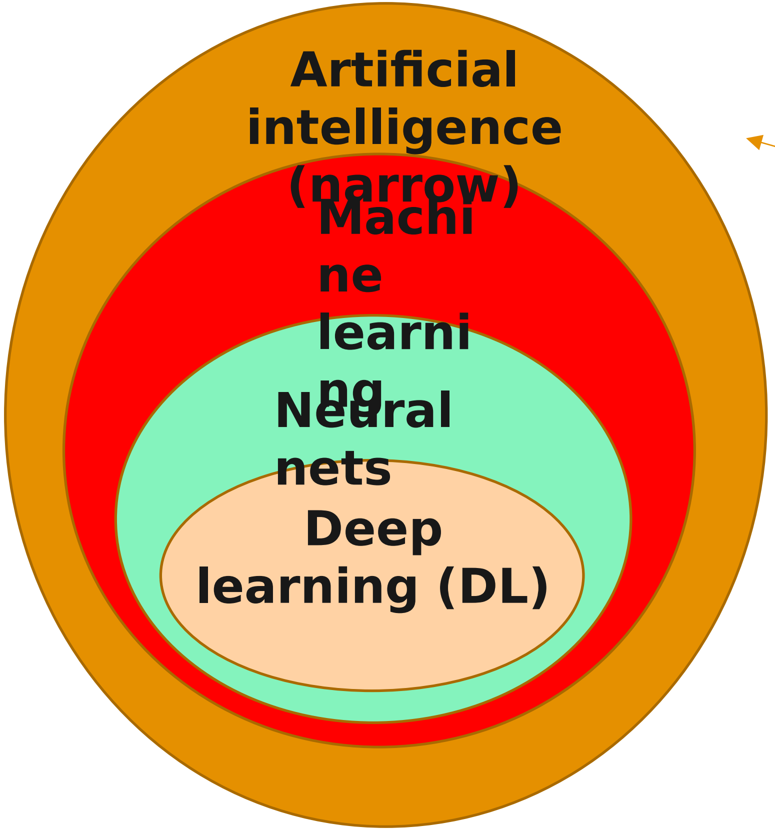
%matplotlib inline **- Display matplotlib graphs in notebook**
%run <file name> **- Run a file**
%%time **- Get an execution time**
%who **- List all variables**
%pinfo <variable> **- Get detailed information about variable**
%env **– List all environment variables**
%load_ext autoreload – **Reload modules**
%pip – **Install in current kernel (instead**

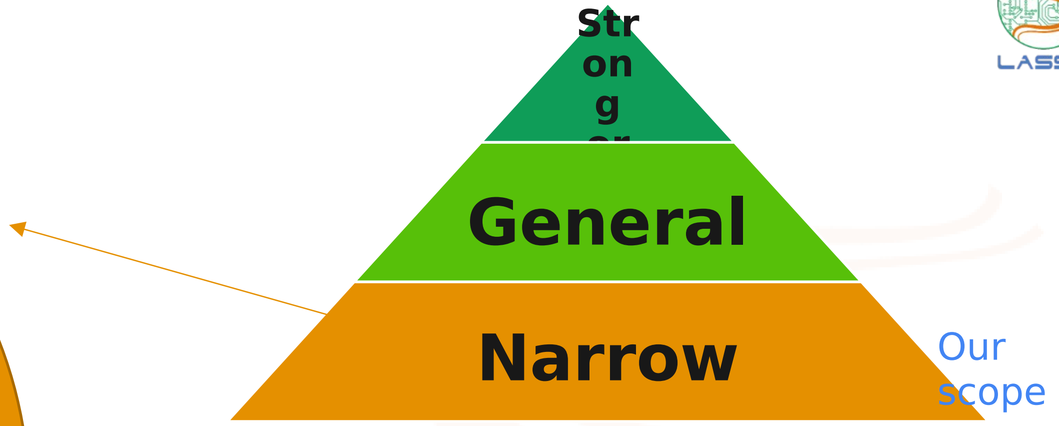
Shell commands in IPython / Jupyter:

Use the ! character as prefix to the command.
For instance:
!ls (on Linux)
!dir (on Windows)

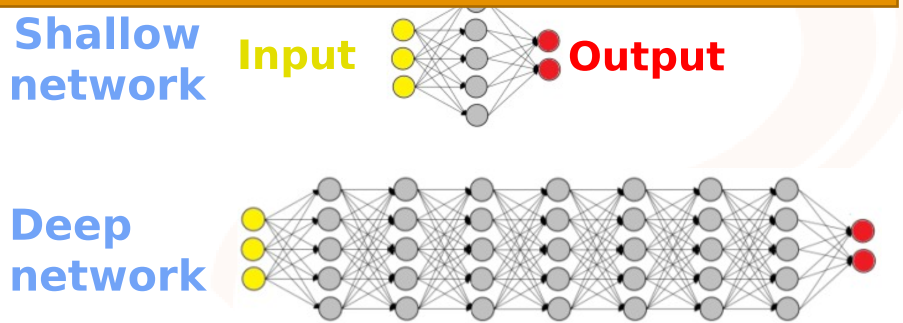
[1]
[2] ...3560
[3] https://jakevdp.github.io/PythonDataScienceHandbook/01.05-ipython-and-shell-commands.html

AI, Machine Learning and Deep learning (DL)





DL is a set of techniques to **train** and **deploy** neural networks with large number parameters



*Classification and regression problems

- Both rely on **supervised learning**: when training the model, we know the correct **output** *y*
- The **input** is a vector x=[x_1, …, x_N] with *N* **features**

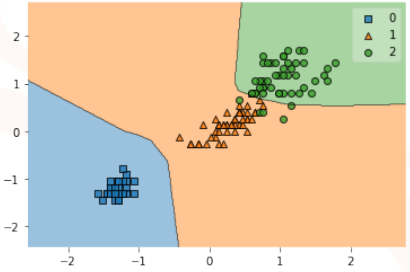
Differences:

Classification

- Output *y* is an element of a set {1,…, *Y*} of *Y* labels.
- Evaluation is based e.g. on misclassification (or error) rate

x_1, x_2, ⋮, x_N → classifier → *y*

Classification example: input [x_1, x_2], *N*=2 and

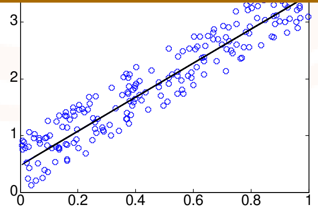


Regression

- Output *y* is a real number or vector (multivariate regression)
- Evaluation is based e.g. on the mean-squared error (MSE), mean absolute error

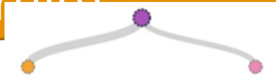
x_1, x_2, ⋮, x_N → regressor → *y*

Regression example: input x_1, *N*=1 and

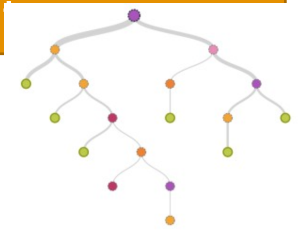


Learning algorithms (most support classification and regression)

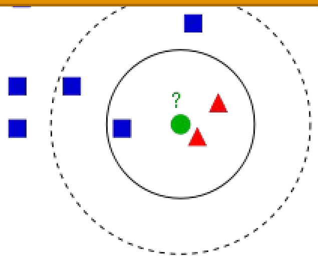
Decision stump (single if/else rule)



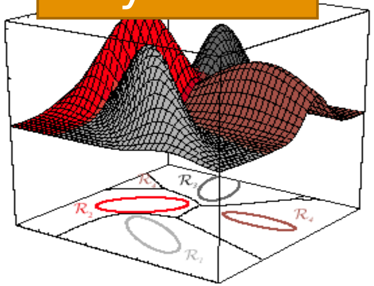
Decision



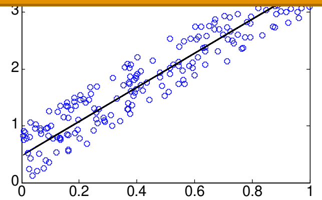
K-nearest neighbors (KNN)



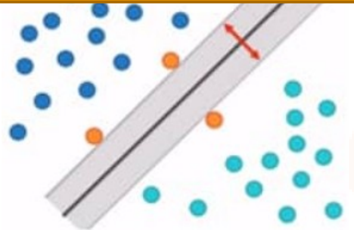
Naïve Bayes



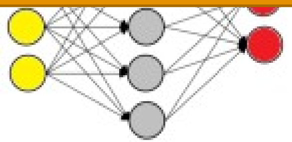
Linear regression



Support vector machine (SVM)

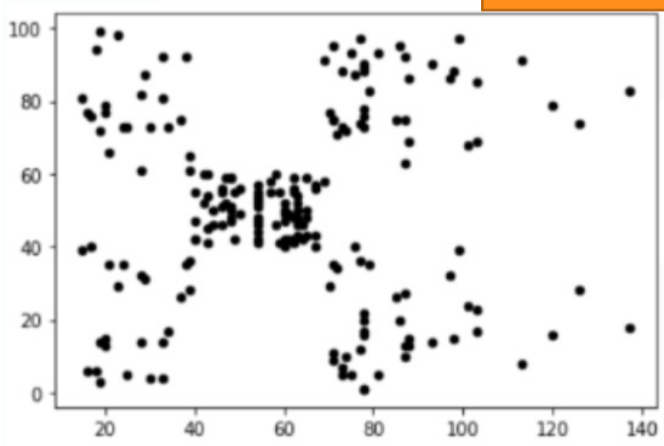


Artificial neural network (ANN)

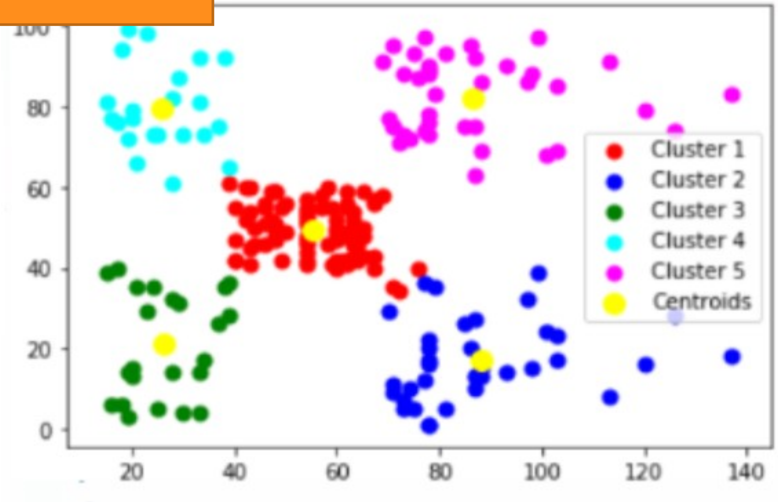


Alternative to supervised learning: unsupervised

Unsupervised learning



Unlabeled data

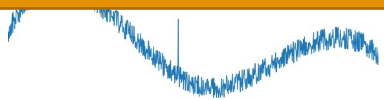


K-means clustering with K=5 centroids

Adapted from https://ai.plainenglish.io/what-is-k-means-clustering-3060791cb589?

Popular special case of unsupervised learning: anomaly detection

Examples of anomalies (univariate time series)



Spike

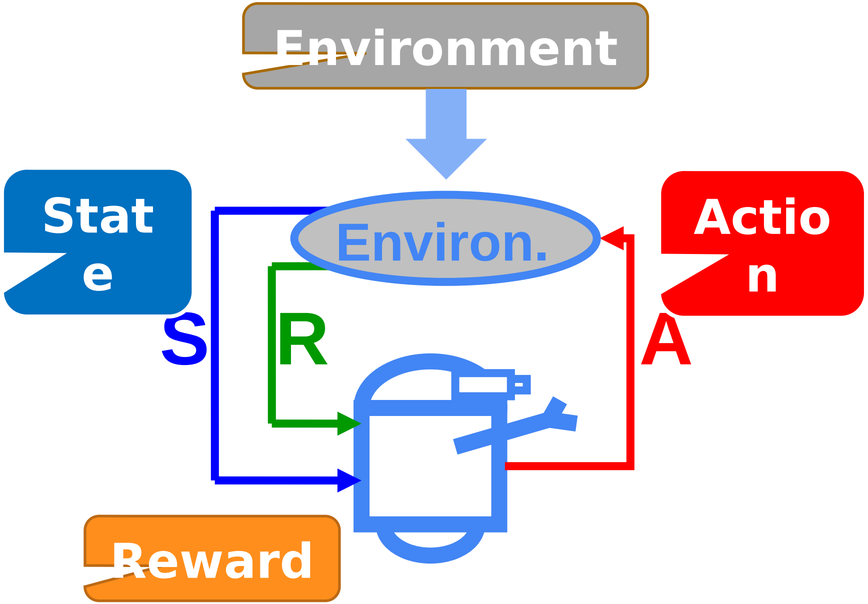
Level shift

Pattern change

Anomaly in seasonal patterns

[1] Anomaly Detection Toolkit (ADTK) - https://pypi.org/project/adtk/

Distinct from supervised and unsupervised learning: Reinforcement Learning (RL)



Online learning, no need for output labels.
Support to delayed reward

Goal: Find a policy that maximizes the ***return*** over a lifetime (episode, if not a continuing task), not the immediate ***reward***

Classification with Scikit-Learn

Simple classifiers (for two simple sets)

- Let us design a decision stump using the two simple sets below:

Training set

Length	Weight	Class y
12	3.2	0
10	0.5	1
14	2.8	0
14	2.4	0
13	1.8	1
13.8	1.5	0
11	1	1

Test set

Length	Weight	Class y
13	3.1	0
9	0.8	0
12.3	1.4	1
10	2.3	1

Getting familiar with the given data

- Because there are (only) two features, it is easy to visualize the training set

Training set

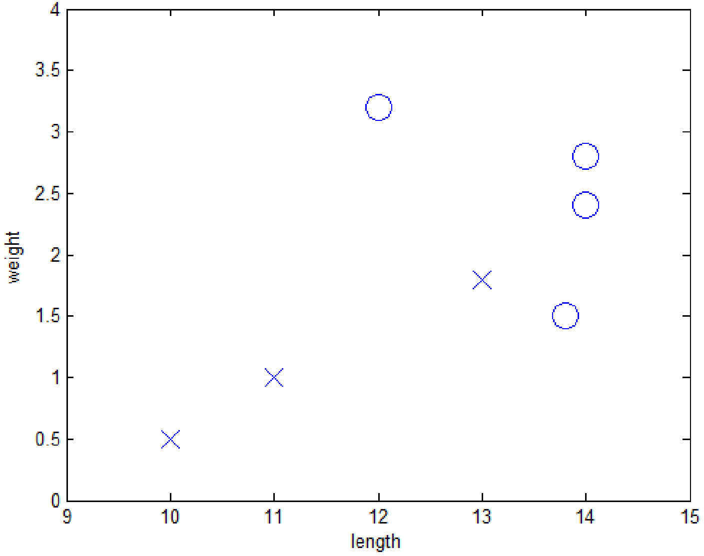
Length	Weight	Class *y*
12	3.2	0
10	0.5	1
14	2.8	0
14	2.4	0
13	1.8	1
13.8	1.5	0
11	1	1

train_set.txt - Bloco de n...

Arquivo Editar Formatar Exibir Ajuda

```
12    3.2  0
10    0.5  1
14    2.8  0
14    2.4  0
13    1.8  1
13.8  1.5  0
11    1    1
```

Text (ASCII) file



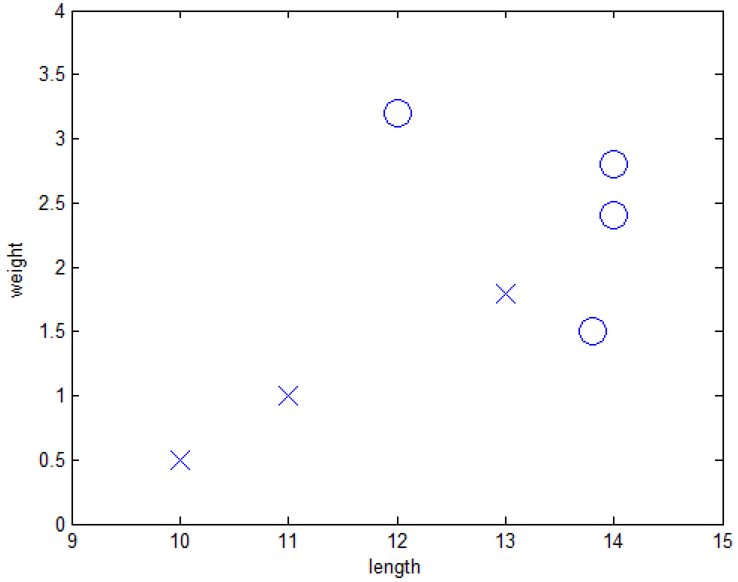
First classifier: decision stump

- Decision stump is a single if / else rule based on a chosen threshold value of a chosen feature
- First example:
  - if weight > 1
    - then class label is 0
  - This gives one error in training set!
- Another example:
  - if length < 12
    - then class label is 1
  - This also gives one error in training set!

train_set.txt - Bloco de n...

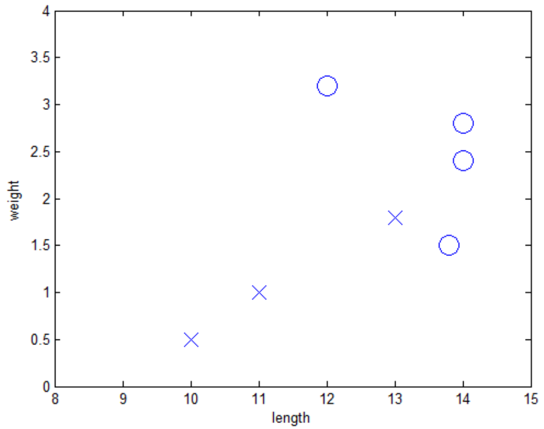
Arquivo Editar Formatar Exibir Ajuda

12	3.2	0
10	0.5	1
14	2.8	0
14	2.4	0
13	1.8	1
13.8	1.5	0
11	1	1

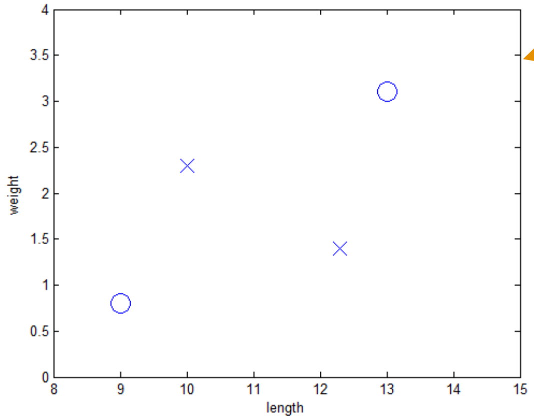


Test our decision stump

- Example:
  - if weight > 1
    - then class label is 0
  - Else
    - then class label is 1
  - This gives one error in training set!



- But gives three errors in this (strange) test set!



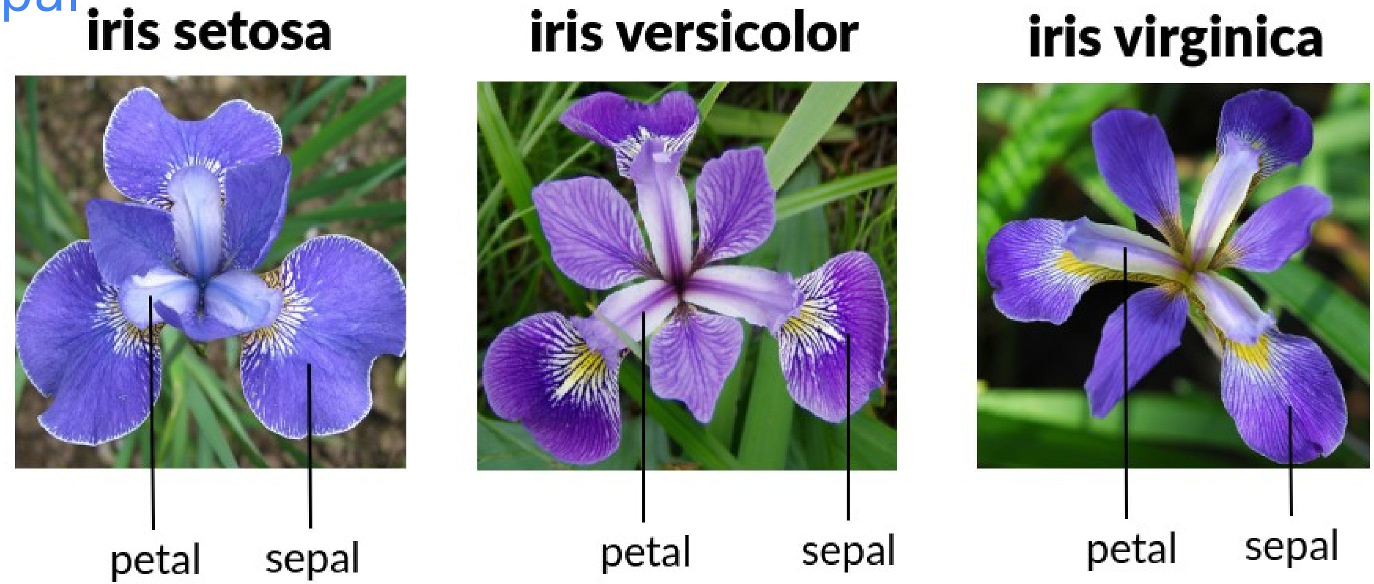
test_set.txt - Bloco de n...

Arquivo Editar Formatar Exibir Ajuda

13	3.1	0
9	0.8	0
12.3	1.4	1
10	2.3	1

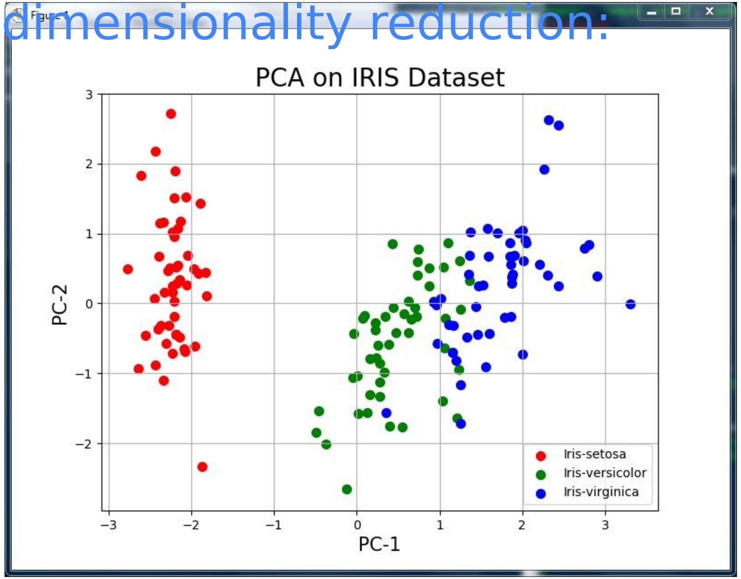
Another dataset: Iris

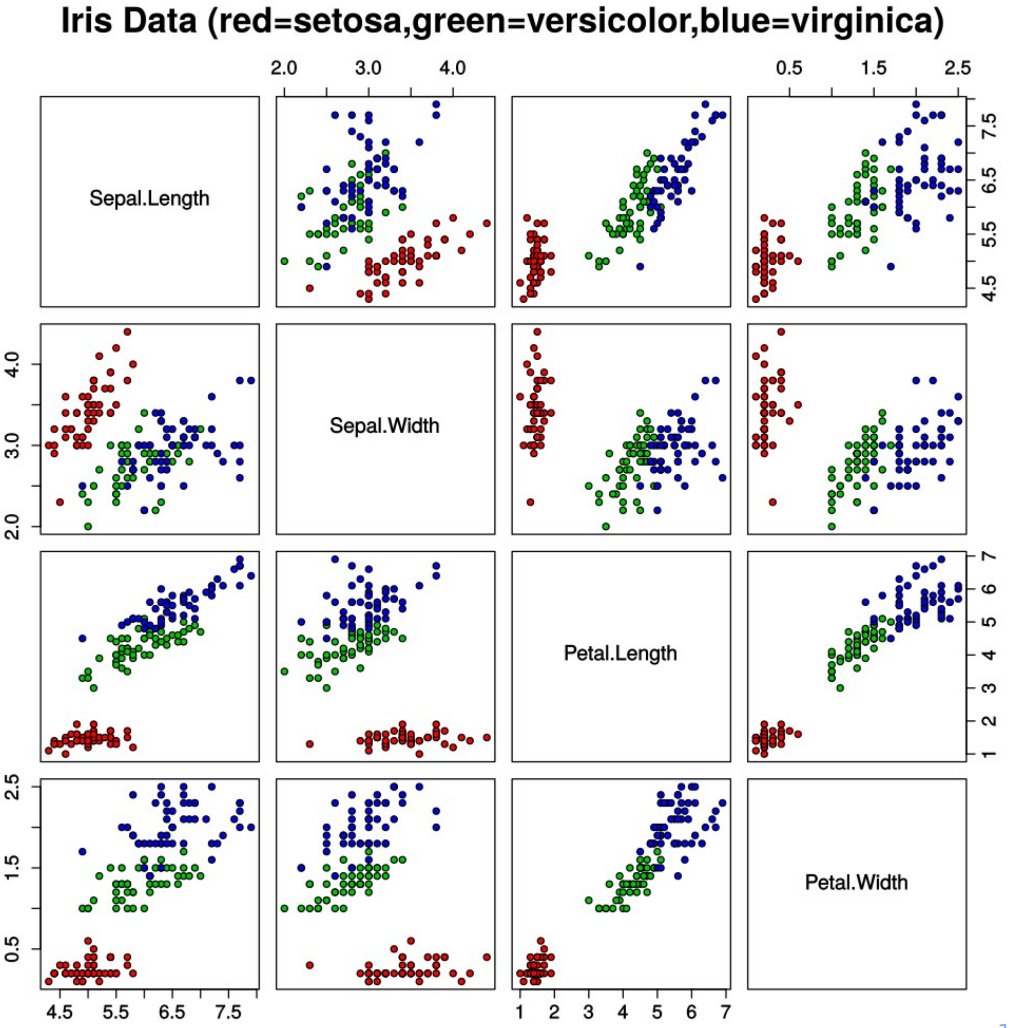
Iris has three classes with 50 examples of each class, and four input features: width and length for petal and sepal



Iris

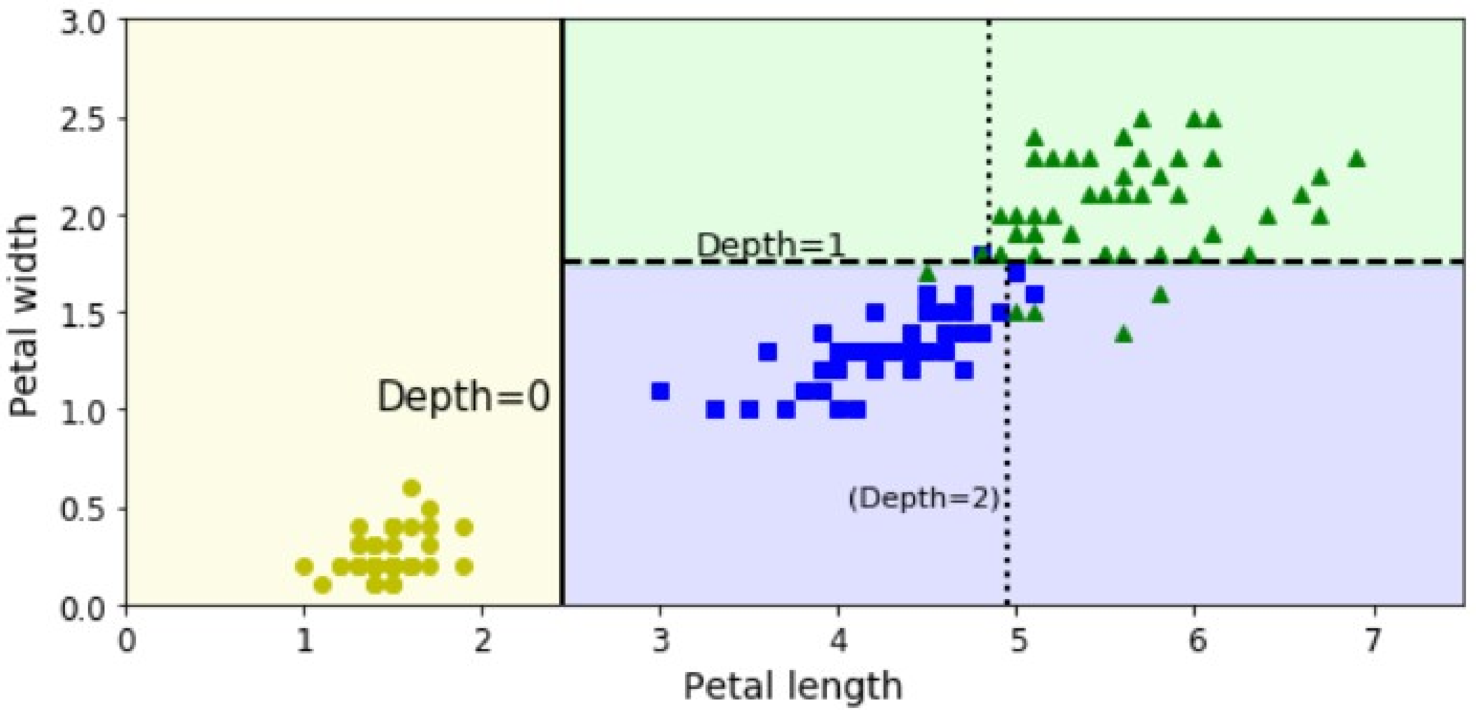
Principal component analysis (PCA) for dimensionality reduction:





https://en.wikipedia.org/wiki/Iris_flower_data_set

Exemplo: Dataset Iris



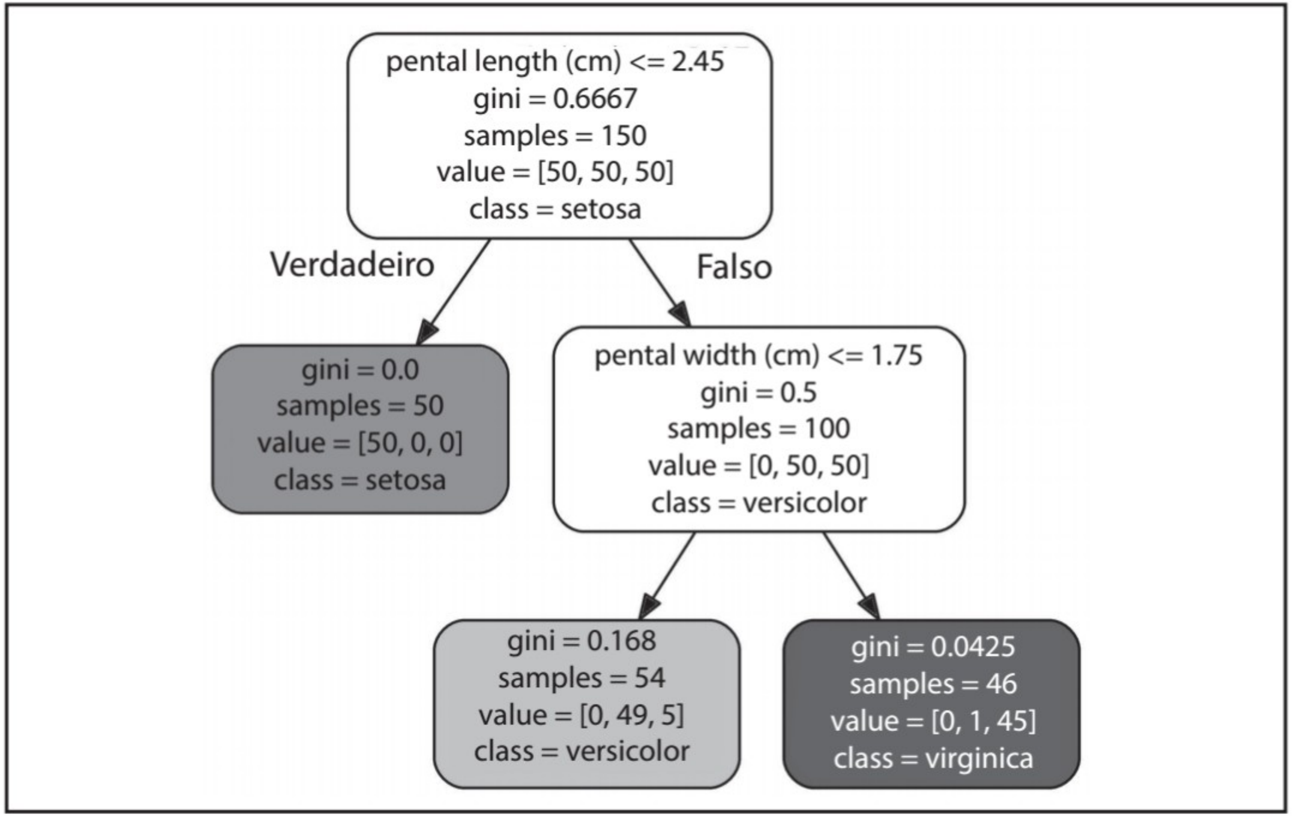
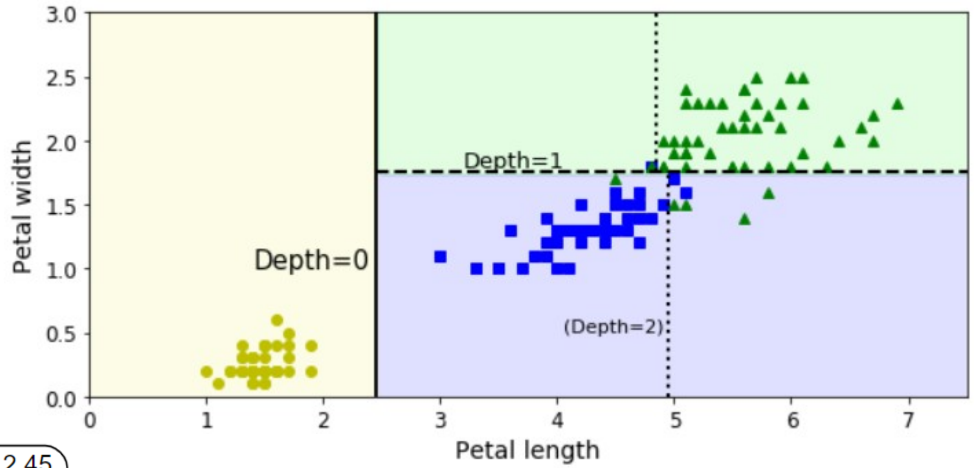
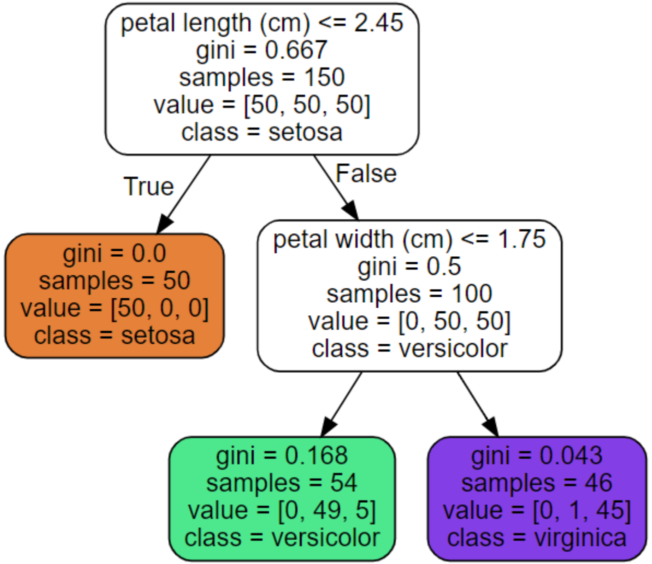


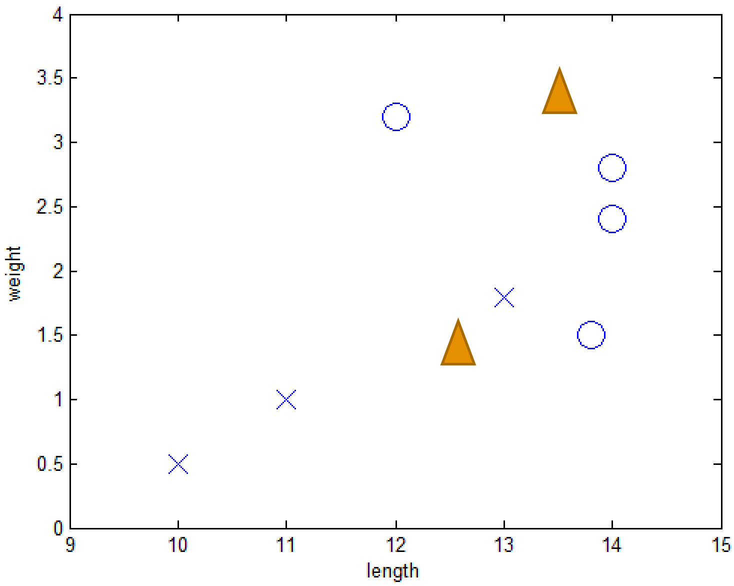
Figura 6-1. Árvore de Decisão da íris


Depth=0
Depth=1
(Depth=2)
Petal width
Petal length
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0
1
2
3
4
5
6
7


petal length (cm) <= 2.45
gini = 0.667
samples = 150
value = [50, 50, 50]
class = setosa
True
False
gini = 0.0
samples = 50
value = [50, 0, 0]
class = setosa
petal width (cm) <= 1.75
gini = 0.5
samples = 100
value = [0, 50, 50]
class = versicolor
gini = 0.168
samples = 54
value = [0, 49, 5]
class = versicolor
gini = 0.043
samples = 46
value = [0, 1, 45]
class = virginica

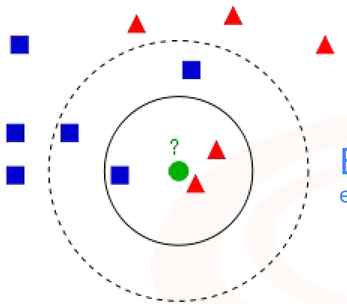
Nearest neighbor classifier

- The nearest neighbor (NN) classifier simply stores the whole training sequence and, according to the adopted distance measure (e.g. Euclidean distance) assigns to the tes example the same class of its nearest (smallest distance) neighbor (exampl of the stored training sequence
- The Euclidean distance corresponds to the squared value of the error vector norm
  - y represents the test vector
  - z represents a training example
  - Euclidean: distance(y, z) = $|| y - z ||^2$



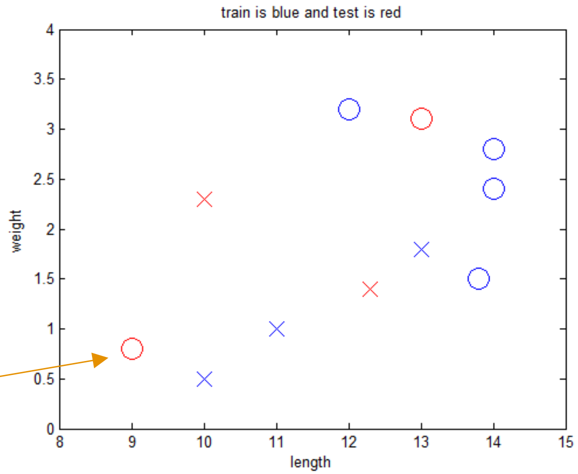
Generalizing the NN: K-nearest neighbors

- KNN classifier: choose K as an integer odd number (e.g. K=3 or 5 is widely adopted) and make the classifier to output the most popular label among the K nearest neighbor as final decision



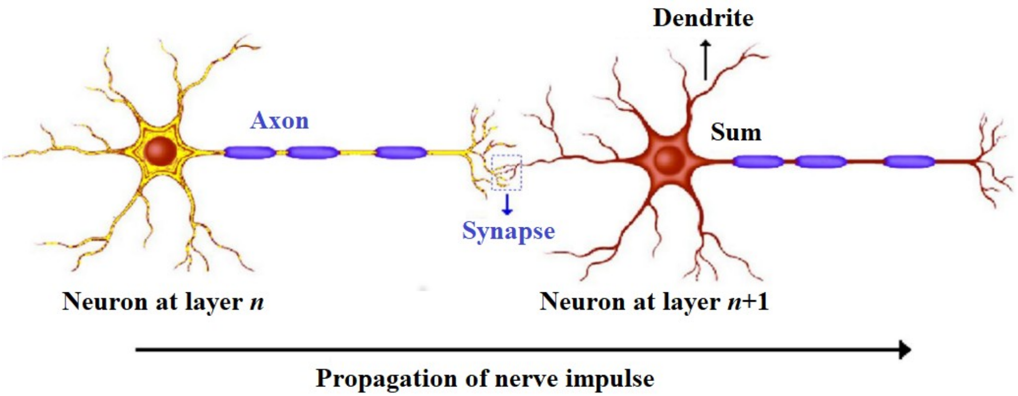
Example from Wikipedia
en.wikipedia.org/wiki/K-nearest_neighbors_algorit

- The previous NN classifier is equivalent to using K=1, which may be less robust to *outliers* than K > 1
- Example: assume that all red and blue examples compose a new training sequence. Note the outlier!

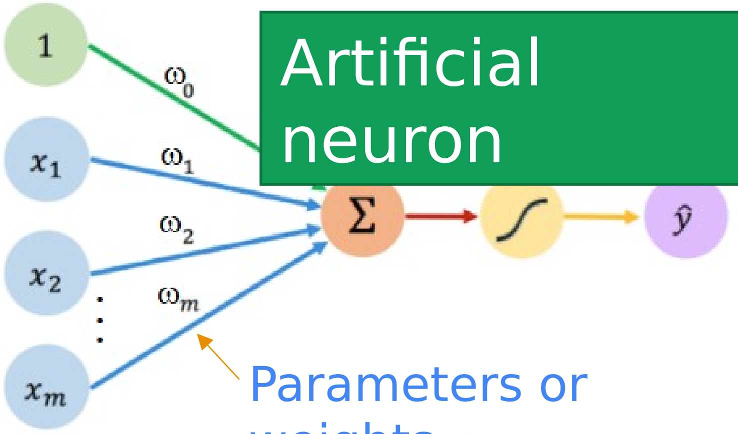


Artificial neural network (ANN or NN)

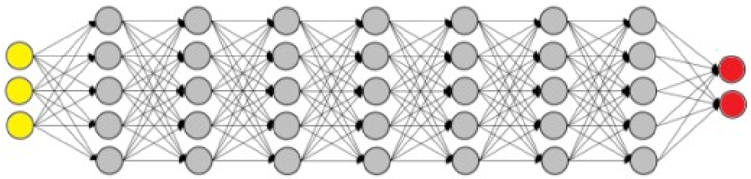
Biological neuron



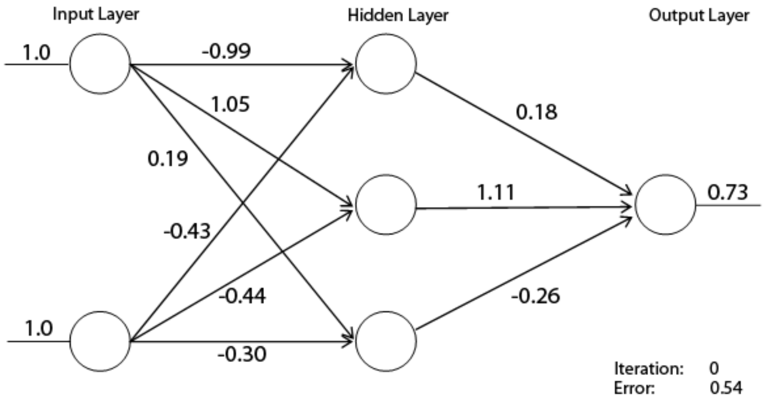
Artificial neuron



Parameters or weights ω



Input Hidden layers Output



Many layers and architectures in DL: dense (fully-connected), convolutional, recurrent, etc.

There is life outside the deep neural net world!

dmlc

XGBoost

https://dmlc.github.io

https://xgboost.ai

XGBoost is an optimized distributed gradient boosting library designed to be highly **efficient**, **flexible** and **portable**. It implements machine learning algorithms under the Gradient Boosting framework. The same code runs on major distributed environment (Hadoop, SGE, MPI) and can solve problems beyond billions of examples.

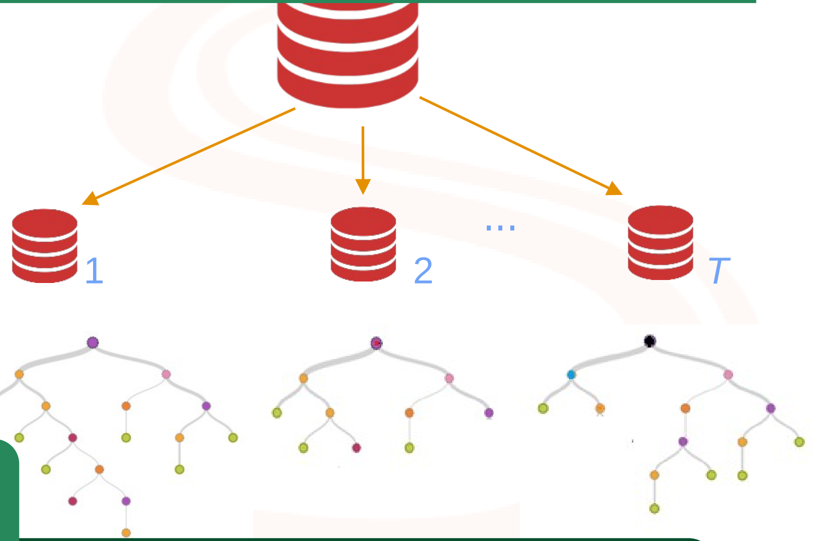
*Random forest: ensemble of decision trees

Extra!

Meta-algorithms → Bagging

Base learners → Decision trees

Random forest: model composed of T decision trees ("estimators" in sklearn)



Recipe:

trees to create a random forest

Bootstrap (sample with replacement)

Random features subsets

Final classification result by voting

Fast training and good generalization. State-of-art results for tabular data

Scikit-learn algorithms for classification

Relevant hyperparameters

Decision tree:	DecisionTreeClassifier(min_samples_leaf=5)
Decision stump:	DecisionTreeClassifier(max_depth=1)
Naïve Bayes	GaussianNB(priors=None, var_smoothing=1e-09)
K-nearest neighbors (KNN)	KNeighborsClassifier(n_neighbors=3, metric='euclidean')
Support vector machine (SVM)	SVC(C=1.0, kernel='rbf', decision_function_shape='ovo')
Linear (SVM)	LinearSVC(C=1.0, decision_function_shape='ovr')
Artificial neural network (ANN)	MLPClassifier(max_iter=500, hidden_layer_sizes=(100,))
Adaboost	AdaBoostClassifier(base_estimator=DecisionTreeClassifier(max_depth=1), n_estimators=50)
Random forest	RandomForestClassifier(n_estimators=30, max_depth=100)

Classification with scikit-learn

```
classifier = DecisionTreeClassifier(max_depth=20) #single tree
classifier = RandomForestClassifier(n_estimators=30,max_depth=10) #30 trees
classifier = SVC(gamma=1, C=1) #SVM with RBF kernel

classifier.fit(X_train, y_train) #training stage
y_predicted = classifier.predict(X_test) #test stage
print('Accuracy = ', accuracy_score(y_test, y_predicted))
```

Be aware that modern ML has sophisticated workflows that require time to set up and get familiar with

[1] https://scikit-lear

Tipo de treino

Classificação

Regressão

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. " O'Reilly Media, Inc.", 2019.

Métricas de performance

instâncias
Ex: casa

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. " O'Reilly Media, Inc.", 2019.

Métricas de performance

X= Valores dos atributos

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. " O'Reilly Media, Inc.", 2019.

Métricas de performance

Y = Resultado esperado

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m} \sum_{i=1}^{m} \left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)} \right)^2}$$

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. " O'Reilly Media, Inc.", 2019.

Métricas de performance

h= Função *machine learning*

Equation 2-1. Root Mean Square Error (RMSE)

$$\mathrm{RMSE}(\mathbf{X}, h) = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left(h\left(\mathbf{x}^{(i)}\right) - y^{(i)}\right)^2}$$

GÉRON, Aurélien. **Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems**. " O'Reilly Media, Inc.", 2019.

Feature scaling

<u>Min-max</u> scaling (*normalization*)

<u>Standard</u> scaling (*standardization*)

Feature scaling

<u>Min-max</u> scaling (*normalization*)

- Limita o *range* para valores entre 0 e 1 (ou outro range designado)
- **Desvantagem:** sujeito a ter a performance prejudicada por <u>*outliers*</u>, ou seja, valores que estão muito acima ou muito abaixo da média
- **Método de aplicação:**

Para cada amostra

1) Subtrair o valor pelo valor mínimo
2) Dividir o valor pela diferença entre valor mínimo e máximo

Ex: valor - min

Feature scaling

- Faz com que a distribuição dos dados passe a ter média 0 e variância unitária.
- **Desvantagem:** Não está restrito a um *range* específico
- **Método de aplicação:**

Para cada amostra

1) Subtrair o valor pelo valor médio
2) Dividir o valor pelo desvio padrão

Ex: valor - média

----

desvio padrão

Standard scaling (*standardization*)

Feature scaling

Ambos os métodos são aplicados independentemente em cada feature

Alguns **métodos afetados pela escala** dos valores das amostras:

- KNN
- Redes Neurais
- Regressão linear
- Regressão logística
- SVM

Alguns **métodos NÃO afetados pela escala** dos valores das amostras:

- Árvores de decisão (decision tree)
- Random Forest

Extra!

*Deep learning frameworks

DL training frameworks implement sophisticated algorithms, which use "backpropagation", automatic differentiation and stochastic gradient descent (SGD)

python | PyTorch | TensorFlow & Keras

Most used language

Facebook's / Meta's

Google's, TF versions 1 and 2, with high level Keras API

Deployment frameworks: facilitate pruning the models and quantizing the weights for acceleration

Qualcomm's AI Model Efficiency Toolkit

www.tinyml.org

TorchScript, Tensorflow Lite & PyTorch Quantization

Other tools: NVIDIA, Intel, etc.

Auxiliary tools for (shallow) machine learning, debugging, assessing models and running on cloud

Scikit learn | jupyter

It may not be trivial to set up your development workflow

Extra!

*Importance of proper management of Python environments

Example:
Keras in TF2 uses hdf5 file format

conda install
h5py==3.7.0

Keras folder with code that requires h5py:
d:\Programs\Anaconda3\envs\ai6g\Lib\site-packages\h5py

09/11/2022 05:25 PM   3,334,144 hdf5.dll
09/11/2022 05:25 PM   117,760 hdf5_hl.dll

Folder d:\Programs\Anaconda3\envs\ai6g\Lib\site-packages\keras\saving

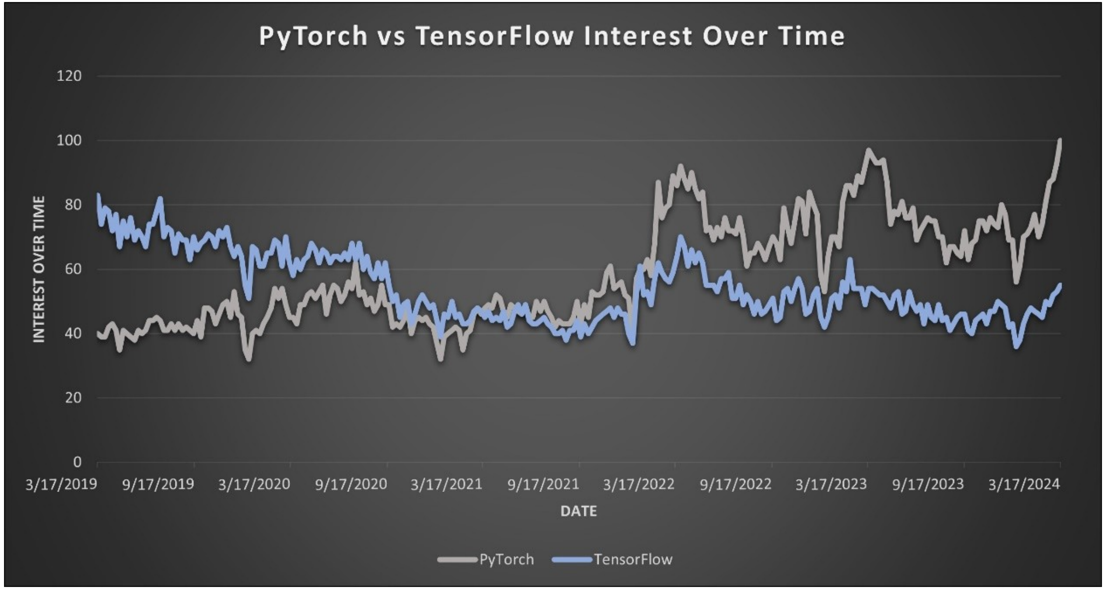
09/05/2022 02:22 PM   37,436

conda list

packages in environment at d:\Programs\Anaconda3\envs\ai6g:
#
Name   Version   Build Channel
absl-py   1.2.0   pypi_0   pypi
aom   3.4.0   h0e60522_1   conda-forge
asttokens   2.0.8   pyhd8ed1ab_0   conda-forge
...
gym   0.19.0   py39h832f523_0   conda-forge
h5py   3.7.0   pypi_0   pypi
...
tensorboard   2.9.1   pypi_0   pypi
tensorboard-data-server   0.6.1   pypi_0   pypi
tensorboard-plugin-wit   1.8.1   pypi_0   pypi
tensorflow   2.9.2   pypi_0

Extra!

*Tensorflow 2 versus Pytorch



TF2: Easier deployment to cloud, servers, mobile, and IoT devices: TensorFlow Serving and TensorFlow Lite

Pytorch: More models, for instance, at https://huggingface.co/
- 85% only PyTorch
- 8% only TF
- 7% both

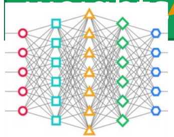
Tools such as: https://github.com/Lyken17/pytorch-OpCounter to count MACs / FLOPs

https://levelup.gitconnected.com/why-tensorflow-for-python-is-dying-a-slow-death-ba4dafcb37e6
https://www.v7labs.com/blog/pytorch-vs-tensorflow

Model selection with a validation set

Parameters

Values that are part of the deployed model, e.g. neural network (NN) weights

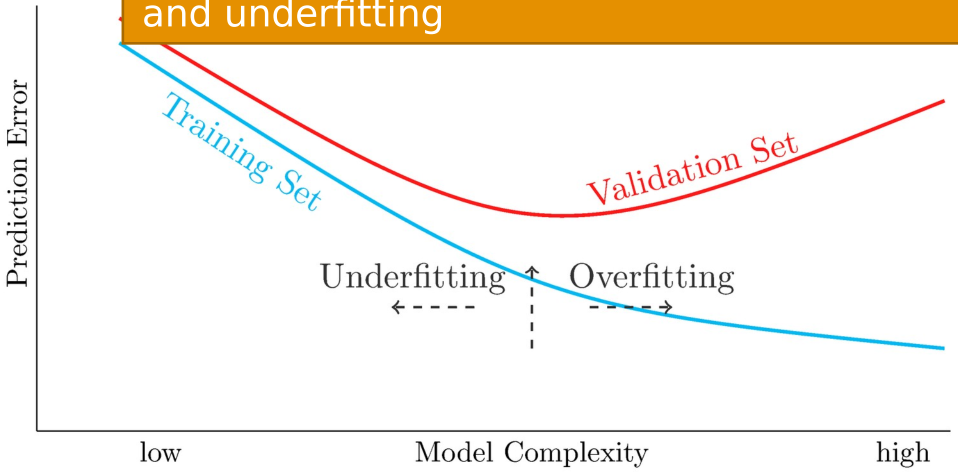


Hyperparameters

- Model hyperparameter: e.g, topology and size of a NN
- Algorithm hyperparameters: learning rate and mini-batch size

Model selection

Tune hyperparameters using a validation set to avoid overfitting and underfitting



(e.g. #

Modify hyperparameters that influence the model complexity and adopt the values that provide good performance on validation set

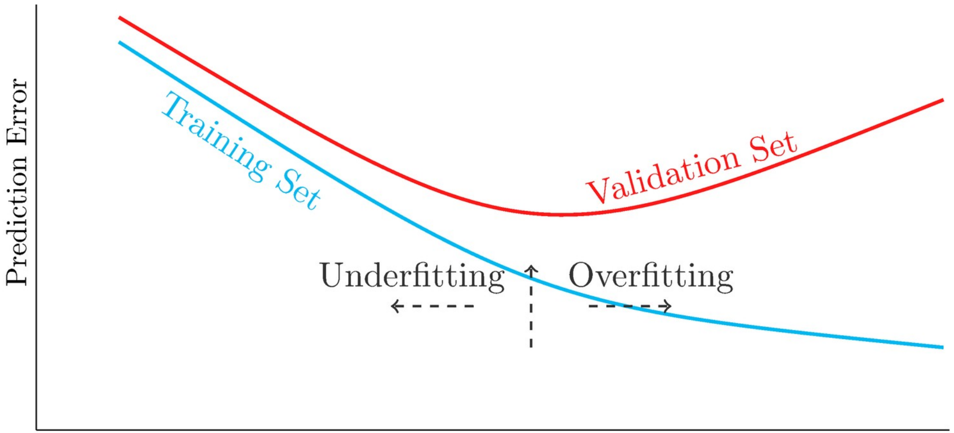
This strategy is also used to find the best value for a given hyperparameter

Model selection with a validation set (2)

DL models have many hyperparameters!

Model selection

Tune hyperparameters using a validation set to ~~avoid overfitting and underfitting~~ find their best values



0.01 hyperparameter (e.g. learning rate) 0.1

Some frameworks for automatic model selection:

Scikit-learn
https://scikit-learn.org/stable/model_selection.html

Optuna
https://optuna.org

KerasTuner
https://keras.io/keras_tuner

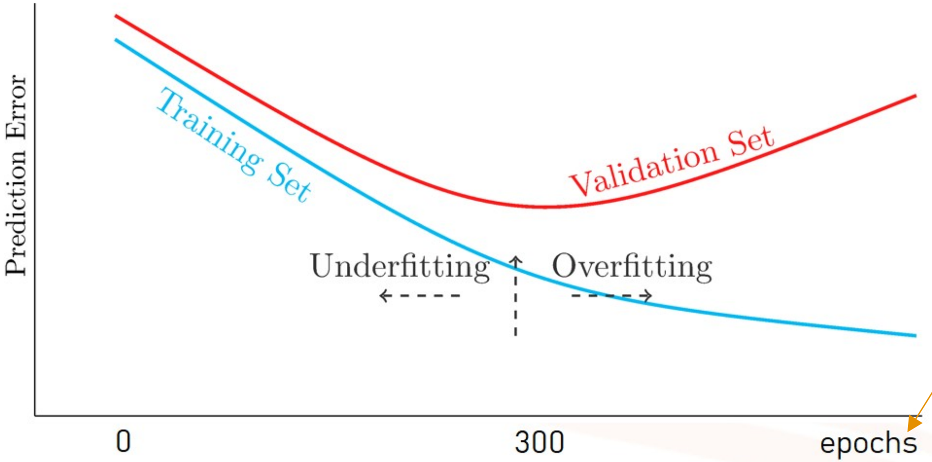
Early stop with a validation set

Assume model selection has indicated a set of hyperparameters

For how long should the model be trained? For how many **epochs**?

Epoch: one complete cycle over all training set examples that may have been shuffled

"Early stop": interrupt training before maximum number of epochs and keep model that maximizes performance on validation set



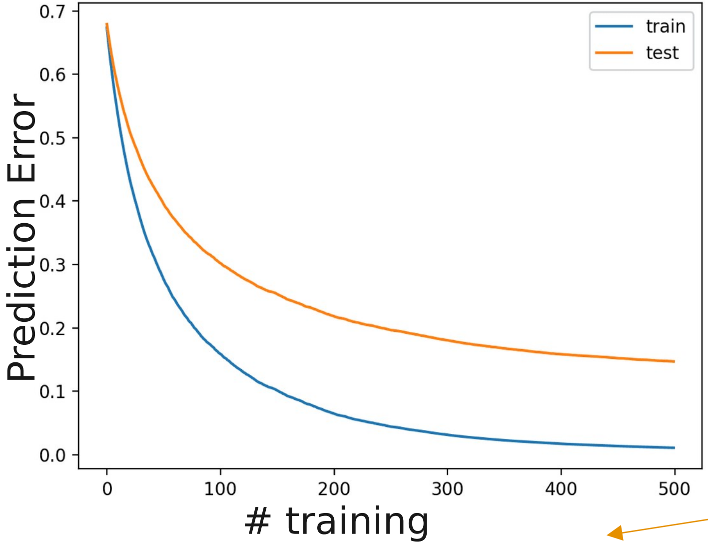
Not the model complexity, but number of epochs

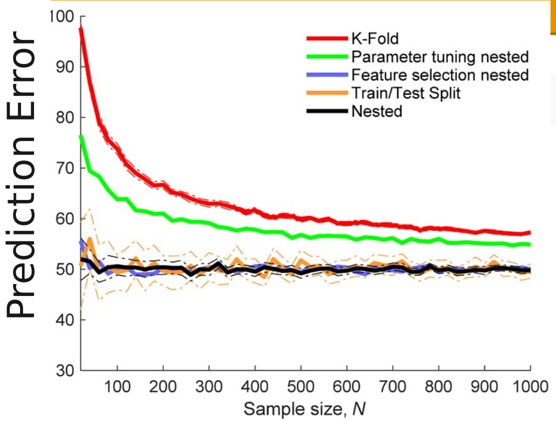
Sample complexity

Given a set of hyperparameters and a model trained with N epochs

What is the minimum number of training examples for good performance?

The sample complexity depends on the model, hyperparameters, data, learning algorithm, etc.





[1] Machine learning algorithm validation with a limited sample size", A. Vabalas et al, 2019

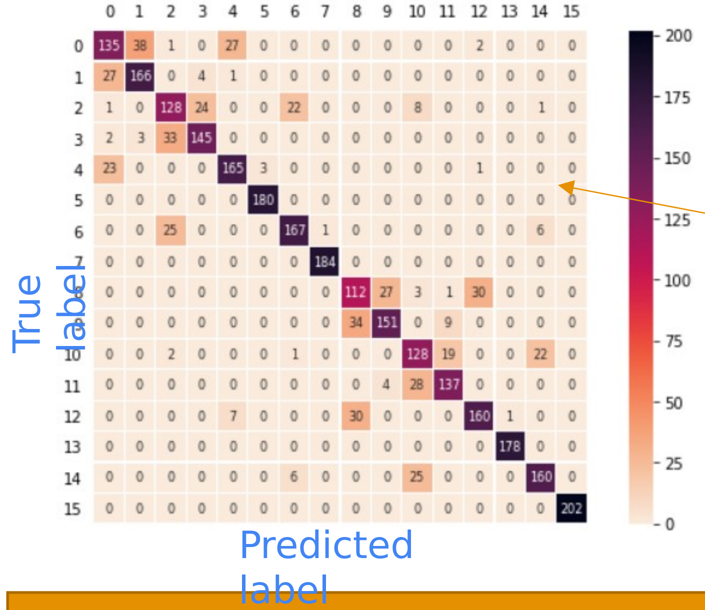
Table 1. The comparison on sample complexity to attain ϵ approximate point of stability.

Algorithm	Sample complexity
REINFORCE[13]	$\mathcal{O}(1/\epsilon^2)$
PGT[14]	$\mathcal{O}(1/\epsilon^2)$
GPOMDP[15]	$\mathcal{O}(1/\epsilon^2)$
SVRPG[4]	$\mathcal{O}(1/\epsilon^{5/3})$
HAPG[22]	$\mathcal{O}(1/\epsilon^3)$
IS-MBPG[18]	$\mathcal{O}(1/\epsilon^3)$
DP-RBPG (this study)	$\mathcal{O}(1/\epsilon^3)$

[2] A randomized block policy gradient algorithm with differential privacy in Content Centric Networks, L. Wang et al, 2021.

Not the model complexity, nor the number of epochs, but the number of

Evaluation



Confusion-matrix for classification

Generalization error: measure of how accurately an algorithm is able to predict outcome values for previously unseen data

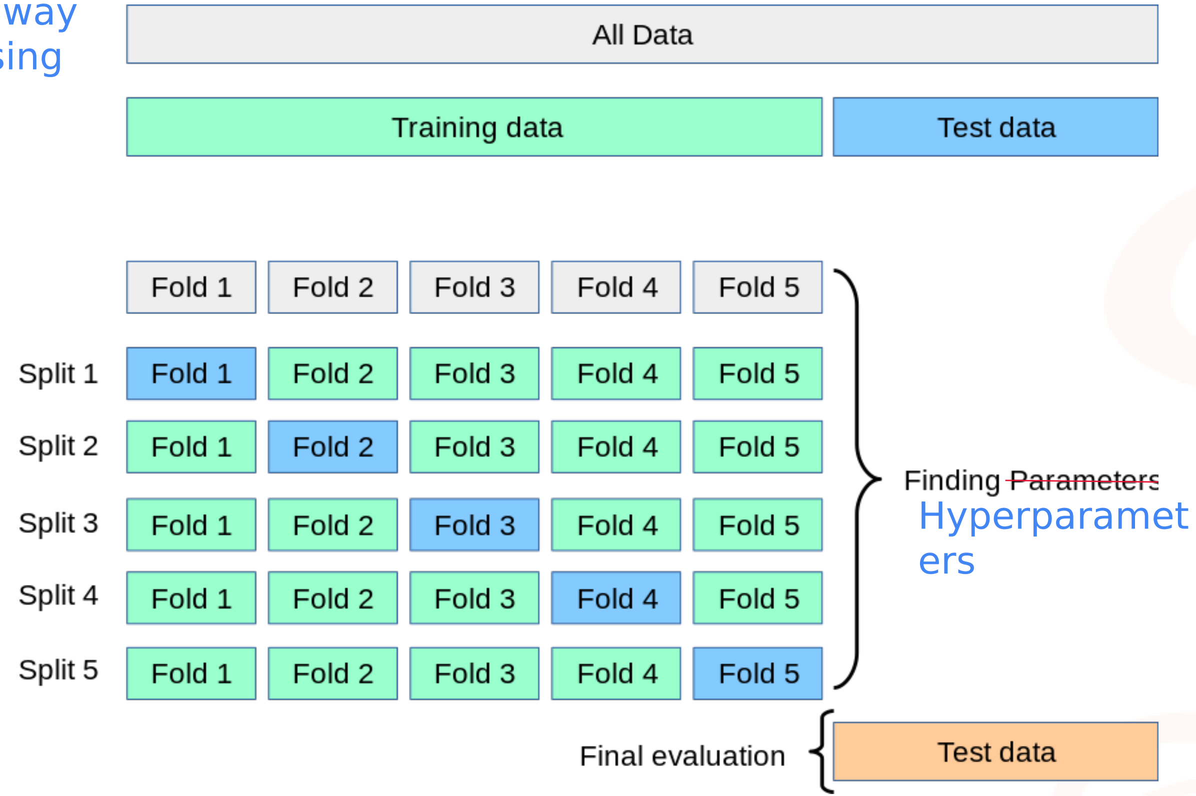
Generalization error estimates:

1. Using unseen **test dataset** (disjoint with the training set)
2. **Cross-validation** with *N* subsets (“splits” or “folds”)
3. **Leave-one-out**: cross-validation with a single example as the test set and all other examples composing the training set

Never estimate the generalization error using the training or validation sets!

Cross-validation (CV) for model selection and evaluation

One way of using CV:



Estimate generalization error

CV can be used for estimating the generalization error and /or hyperparameters

CV can also use all data

Final model and its parameters should be obtained using all data

https://scikit-learn.org/stable/modules/cross_validation.html

Experiment reproducibility is not only initializing RNG with a given seed

Three different pseudo random number generators (RNG) when simulating with Tensorflow

```
import numpy as np
import tensorflow as tf
import random as python_random
# Start random number generation in
# well-defined initial state.
np.random.seed(123) # Numpy
python_random.seed(456) #Core Python
tf.random.set_seed(789) #Tensorflow
```

For reproducibility, one needs also to properly provide dataset and code

https://keras.io/getting_started/faq/#how-can-i-obtain-reproducible-results-using-keras-during-development