Workshop: Introduction to Deep Learning

Lecture 1: Classical Machine Learning algorithms

Petia Georgieva

Department of Electronics Telecommunications and Informatics (DETI)
Institute of Electronics and Informatics Engineering of Aveiro (IEETA)
Intelligent Robotics and Systems Lab
University of Aveiro, Portugal (https://www.ua.pt/)
Email: petia@ua.pt

European Union-NextGenerationEU, through the National Recovery and Resilience Plan of the Republic of Bulgaria, project N0 BG-RRP-2.004-0005.

universidade de aveiro

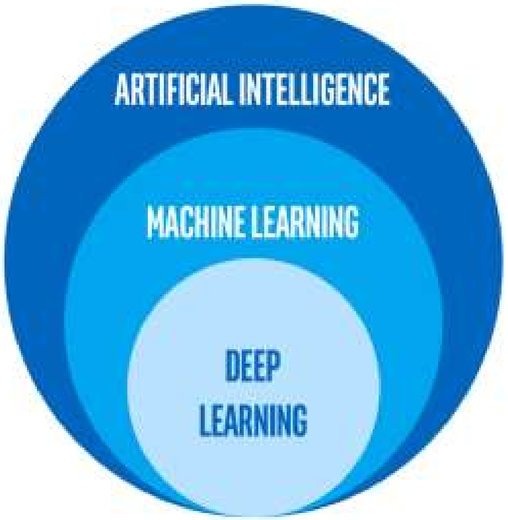
AI - the new Electricity

Artificial Intelligence (AI) is a general purpose technology that may influence every industry (similar to the internet, electricity) .

AI is based on algorithms of

Machine Learning (ML) & Deep Learning (DL)

This course: Introduction to ML & DL models



Content of this Workshop

Lecture 1: Classical Machine Learning algorithms

Lecture 2: Neural Network (NN) & Deep Learning (Conv NN, Unet)

Lecture 3: Deep Learning (YOLO, RNN, LSTM, GRU, Transformers)

Lectura 4: Wrap up of all topics. Questions&Answers.
Student short presentations (if you want)

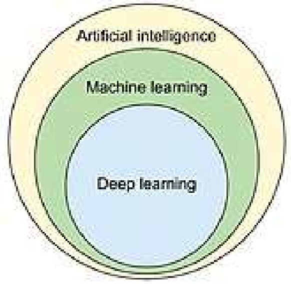
Outline (Lecture 1)

Part 1: Introduction

Part 2: Supervised learning
- Linear Regression
- Classification - Logistic regression
- Cost/loss function
- Gradient descent learning

Part 3. Model Performance evaluation
- Confusion matrix
- K-fold cross validation

Why Machine Learning ?



- **Sensors** get cheaper (e.g. widely available IoT devices)

- Exponential **growth of data** – IoT, medical records, biology, engineering, etc.
- **Data sources**: sound, vibration, image, electrical signals, accelerometer, temperature, pressure, LIDAR etc.
- Increasing **computational resources**.

- **Complex Applications:**

  ✓ Autonomous driving;
  ✓ Intelligent robotics;
  ✓ Computer Vision;
  ✓ Natural Language Processing (Speech recognition, Machine translation)
  ✓ 5G+ networks

Computer Vision Tasks

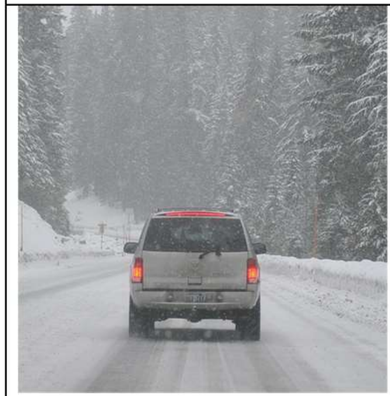
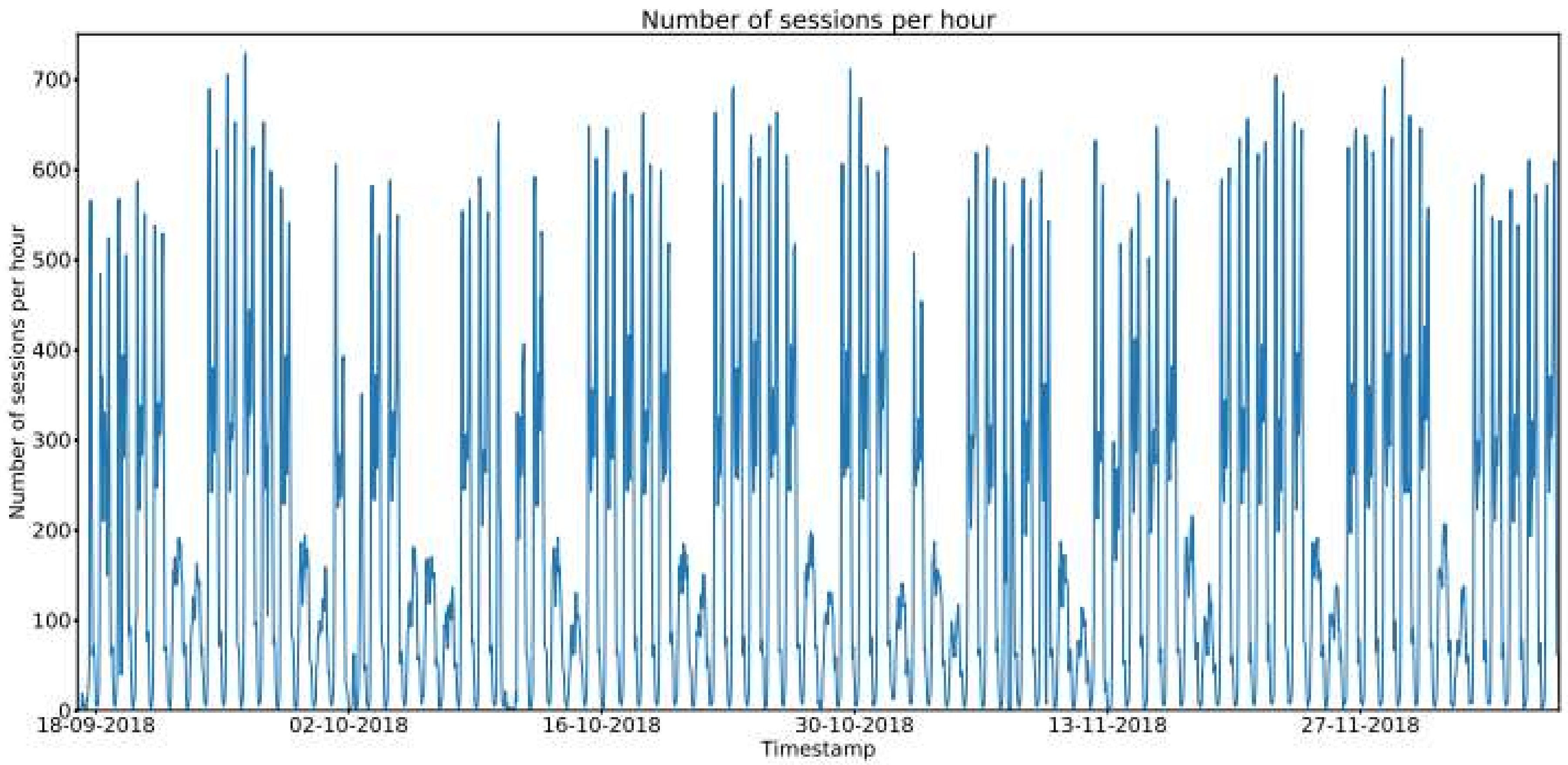
Image classification	Classification & Localization	Detection
	 b_x, b_y, b_h, b_w	

Image classification: input a picture into the model and get the class label (e.g. person, bike, car, background, etc.)

Classification & localization: the model outputs not only the class label of the object but also draws a bounding box (the coordinates) of its position in the image.

Object Detection: outputs the position and labels of several objects.

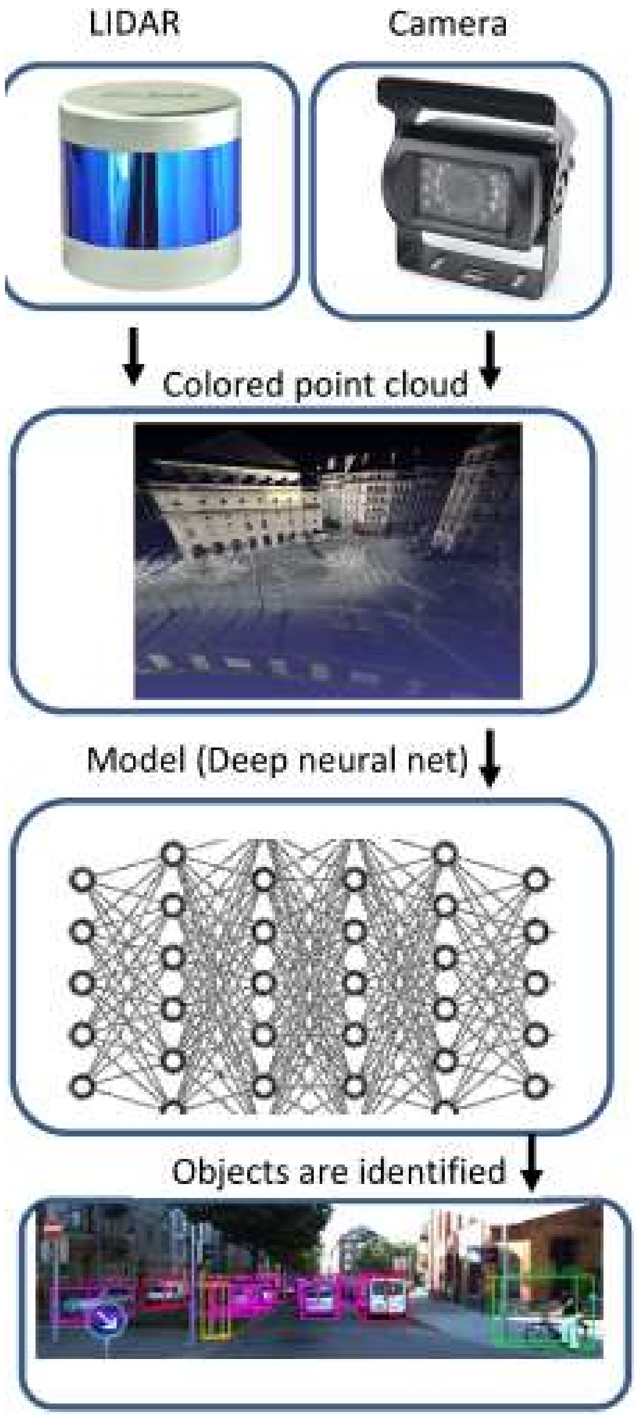
Time Series (TS) Data



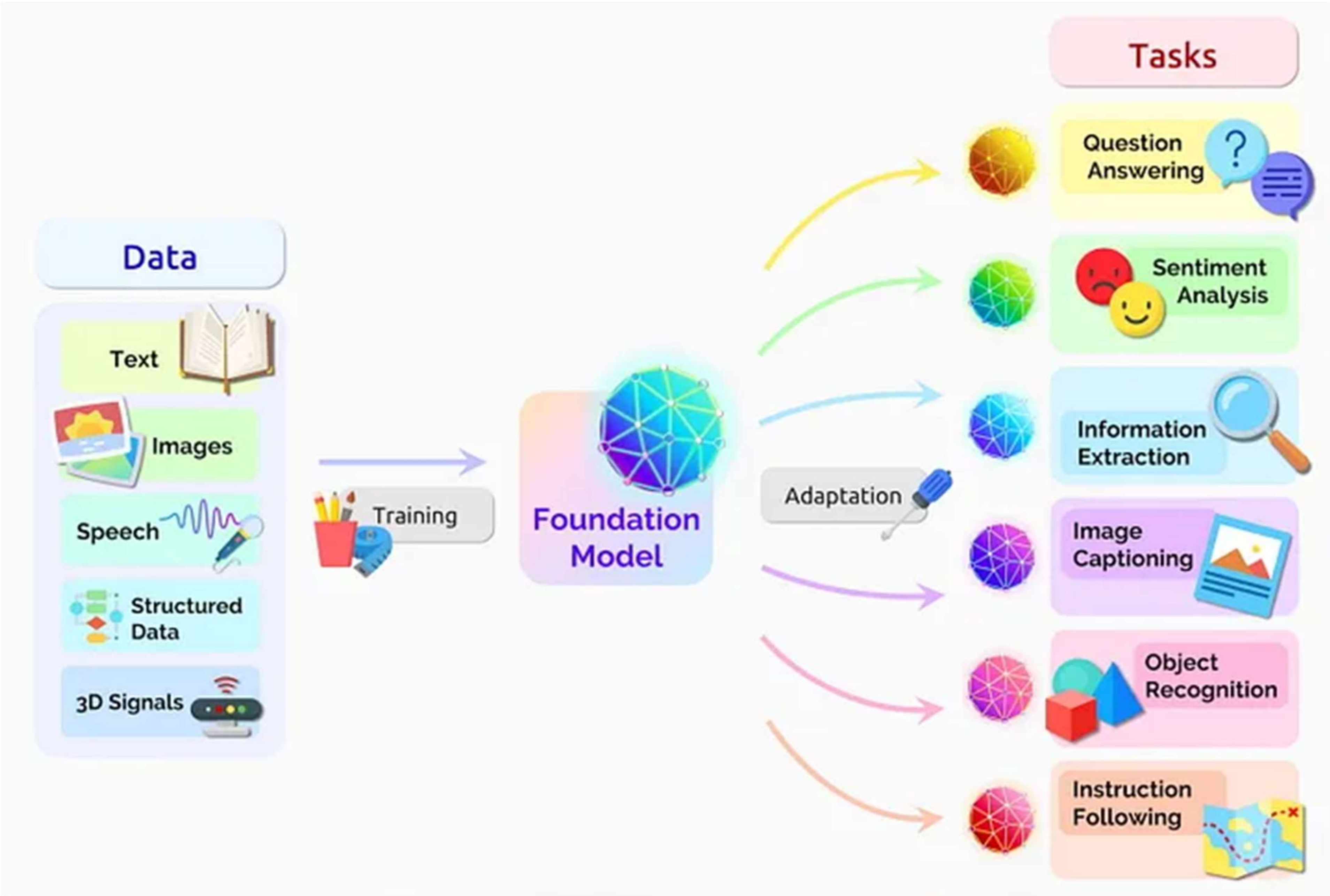
Time Series - collection of data points indexed based on the time they were collected . Most often, data are recorded at regular time intervals.

universidade de aveiro

Multimodal Object Detection

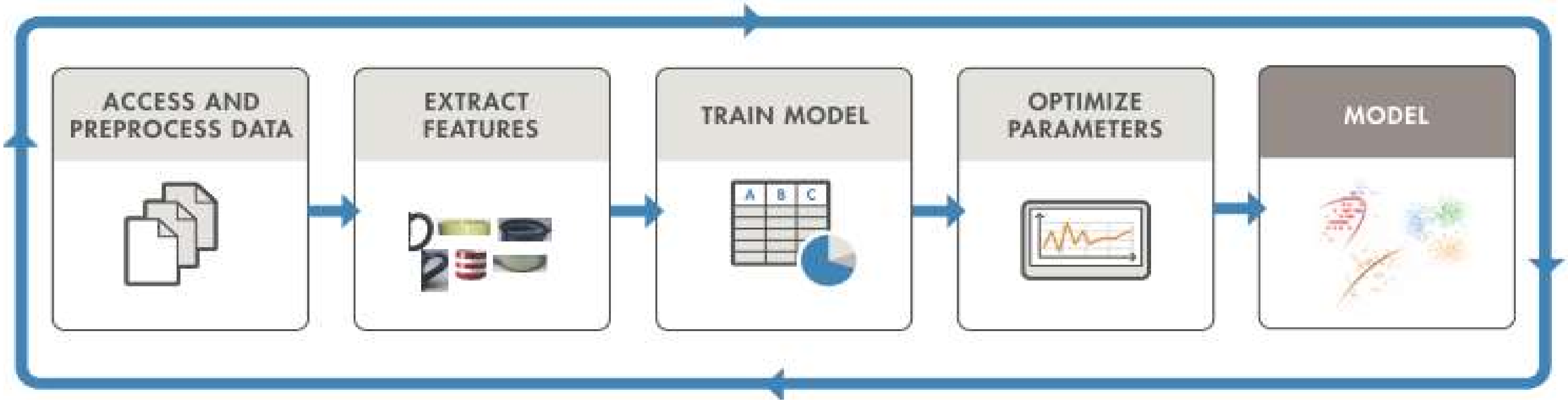


Multimodal generative AI

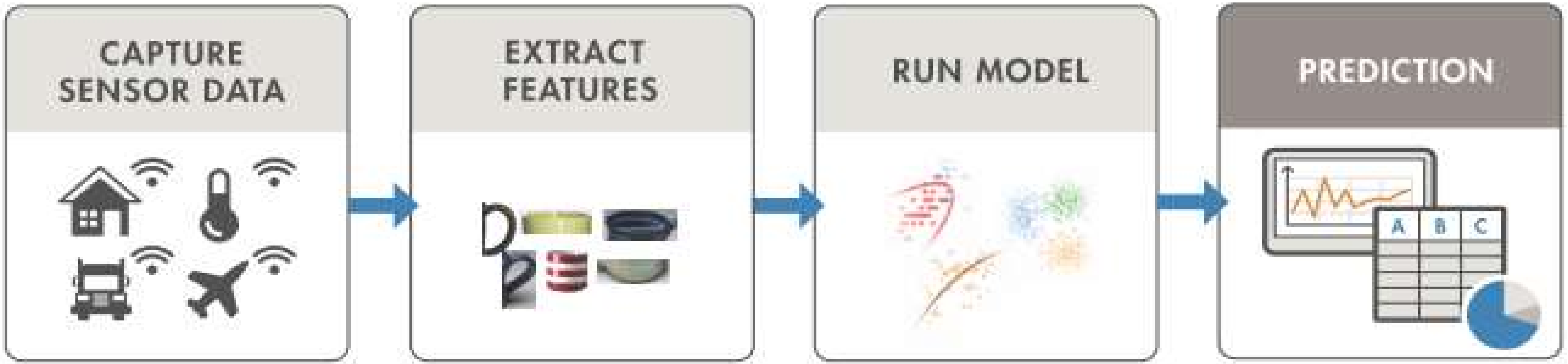


ML workflow

Train: Iterate until achieve satisfactory performance **(off-line)**



Predict: Integrate trained models into applications **(real time**)



universidade de aveiro

AI/Machine Learning Approaches

Supervised Learning

Given examples with "correct answer" (labeled examples)

(e.g. given dataset with spam/not-spam labeled emails)

Unsupervised Learning

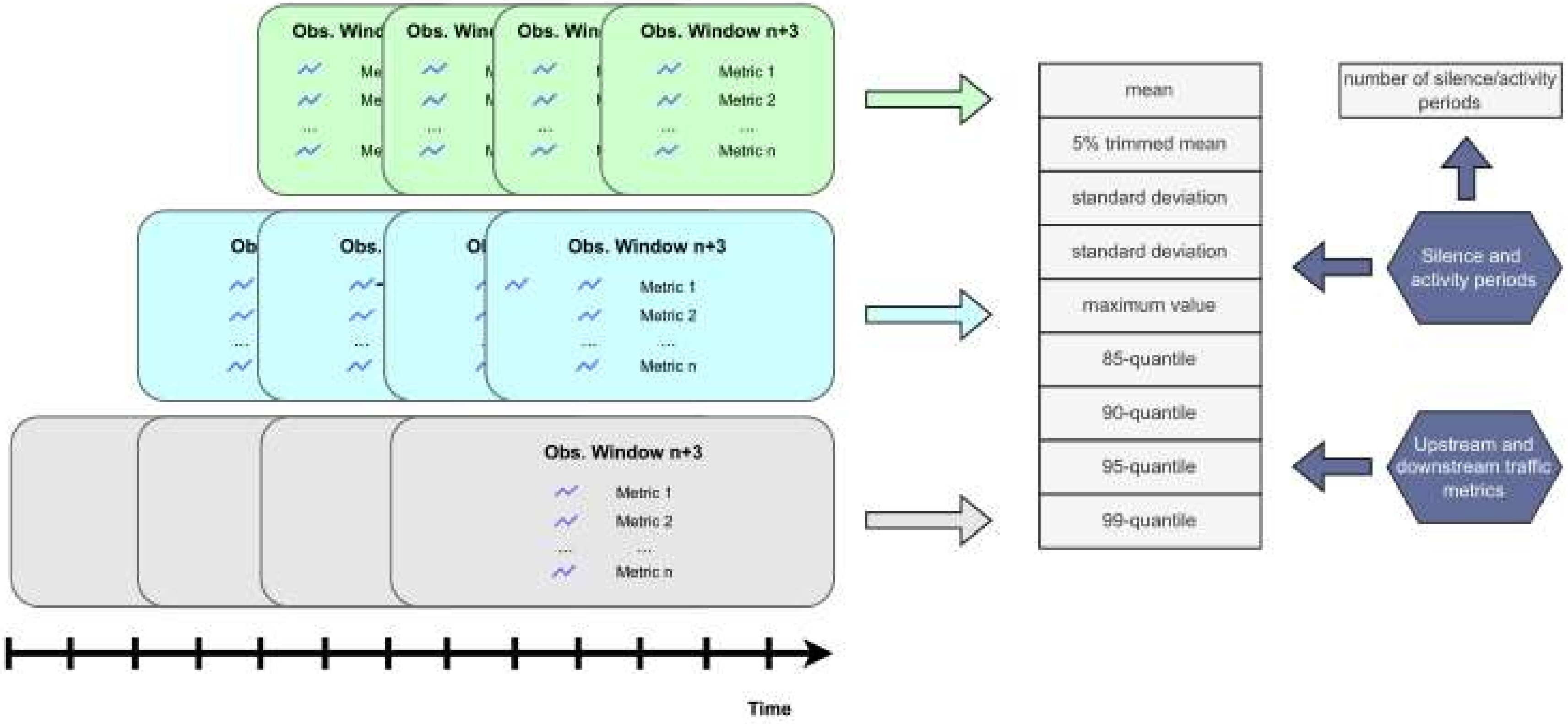
Given examples without answers (no labels).

Deep Learning

Automatically extract hidden features (in contrast to hand-crafted features). Need a lot of data (Big data) . Need for very high computational resources (GPUs).

GenAI (generative AI models) – ChatGPT, Large Language Models (LLM). Trained to generate new data (images, text, music) .

Hand-crafted features – example



Raw data:
collected upstream/downstream network traffic metrics:
uploaded packets (#, Bytes), downloaded packets (#, Bytes), silence/activity periods

Feature extraction (input vector x) - e.g. statistical metrics
mean, max, min, standard deviation, different quantiles, over multiple sub-windows

Class (label y) : Network traffic OK (0) / NOT OK (1)

Supervised Learning

Requires labeled data (examples with “correct answer”).

Regression: The model output is a real number

Ex. Time series forecasting (predict the network traffic)

Ex. Predict some physical variable/property based on other measurable variables

Classification: The model output is a label (e.g. 0, 1).

Ex. Learn to predict OK (0) or NOT OK (1) state of the network

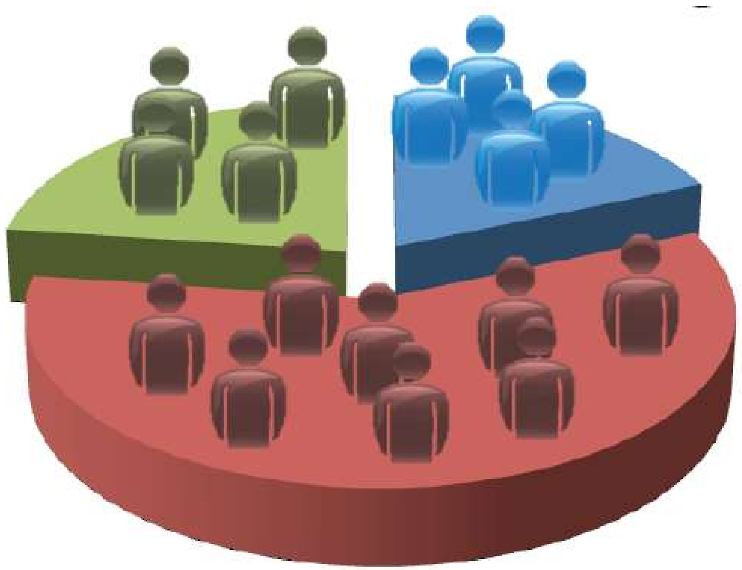
Multiclass – from medical images detect different types of cancer (0, 1,2,3, etc.)

Unsupervised Learning

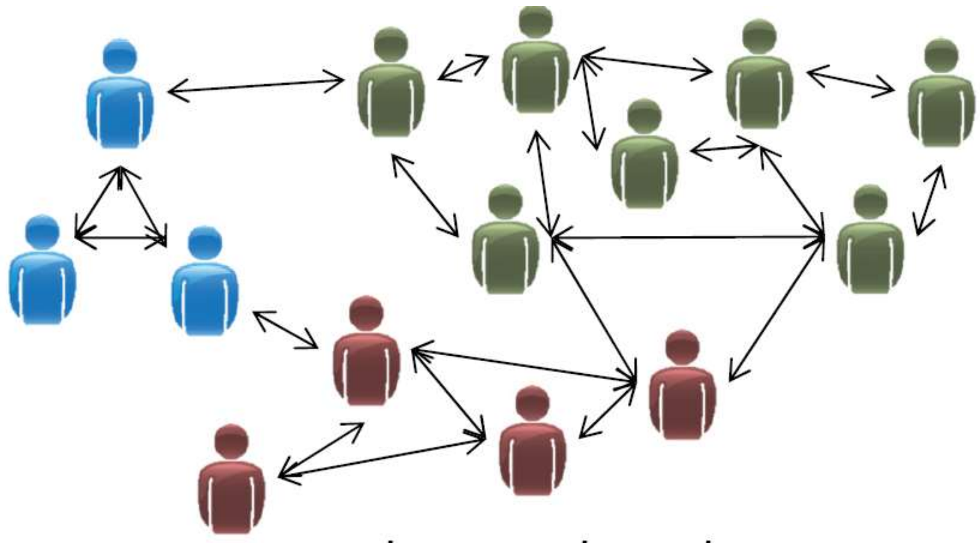
Given unlabeled data, NO correct answers.

Features: education, job, age, marital status, etc.

Market segmentation



Social network analysis



Clustering: Given a collection of examples (e.g. user profiles with a number of features). Each example is a point in the multidimensional space of features. Find a similarity measure that separates the points into clusters.

-***K-means clustering***

universidade de aveiro

Part 2. Supervised learning

2.1 Linear Regression

Standard Data format

x – input vector of features, attributes

y – output vector of labels, ground truth, target

m - number of training examples

n – number of features

$h_\theta(x)$ - model (hypothesis)

θ - vector of model parameters

Training set: data matrix X (m rows, n columns)

	feature x_1	feature x_2	.....	feature x_n	output(label) y
Example 1	$x_1^{(1)}$			$x_n^{(1)}$	$y^{(1)}$
Example 2	$x_1^{(2)}$			$x_n^{(2)}$	$y^{(2)}$
…					
Example i	$x_1^{(i)}$			$x_n^{(i)}$	$y^{(i)}$
…					
…					
Example m	$x_1^{(m)}$			$x_n^{(m)}$	$y^{(m)}$

Example

Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

Problem: Learning to predict the housing price as a function of living area & number of bedrooms.

	feature x_1	feature x_2	.....	feature x_n	output(label) y
Example 1	$x_1^{(1)}$			$x_n^{(1)}$	$y^{(1)}$
Example 2	$x_1^{(2)}$			$x_n^{(2)}$	$y^{(2)}$
...					
Example i	$x_1^{(i)}$			$x_n^{(i)}$	$y^{(i)}$
...					
...					
Example m	$x_1^{(m)}$			$x_n^{(m)}$	$y^{(m)}$

Linear Regression problem

Problem: Learning to predict the housing price as a function of living area & number of bedrooms.

Living area (feet^2)	#bedrooms	Price (1000\$s)
2104	3	400
1600	3	330
2400	3	369
1416	2	232
3000	4	540
⋮	⋮	⋮

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 = [\theta_0 \quad \theta_1 \quad \theta_2] \begin{bmatrix} x_0 = 1 \\ x_1 \\ x_2 \end{bmatrix} = \vec{\theta}^T \vec{x}$$

General Multi-variate/multiple linear regression model:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots + \theta_n x_n = \vec{\theta}^T \vec{x}$$

Linear Regression – iterative gradient descent algorithm

Inicialize model parameters *(e.g. θ =0)*
Repeat until J converge {

Compute Linear Regression Model =>

$$h_\theta(x) = \vec{\theta}^{\,T}\vec{x}$$

Compute cost/loss function =>
(Mean Squared Error - MSE)

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}\left(h_\theta(x^{(i)}) - y^{(i)}\right)^2$$

Goal =>

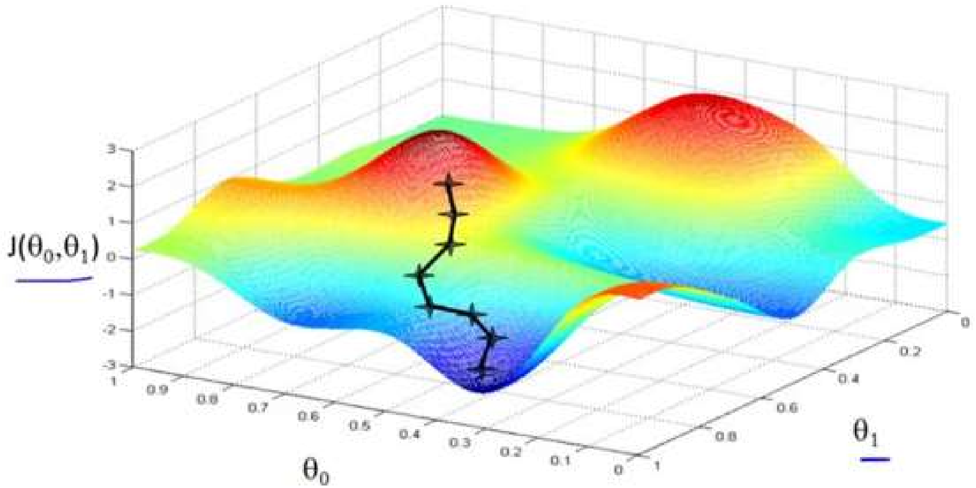
$$\min_\theta J(\theta)$$

Compute cost function gradients =>

$$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Update parameters =>
}

$$\theta_j := \theta_j - \alpha\frac{\partial}{\partial\theta_j}J(\theta)$$



alpha – learning rate > 0

Evaluation Metrics for Regression

$$MSE = \frac{1}{m}\sum_{i=1}^{m}\left[y_i - h_\theta(x^{(i)})\right]^2$$

Mean Squared Error

$$RMSE = \sqrt{\frac{1}{m}\sum_{i=1}^{m}\left[y_i - h_\theta(x^{(i)})\right]^2}$$

Root Mean Squared Error

$$MAE = \frac{1}{m}\sum_{i=1}^{m}\left|y_i - h_\theta(x^{(i)})\right|$$

Mean Absolute Error

Coefficient of determination

(R^2 score)

$$R^2 = 1 - \frac{\sum_{i=1}^{m} \left[y_i - h_\theta(x^{(i)})\right]^2}{\sum_{i=1}^{m} \left[y_i - \bar{y}\right]^2}$$

- Since R^2 is a proportion, it is always a number between 0 and 1.
- If R^2 = 1, all of the data points fall perfectly on the regression line. The predictor *x* accounts for *all* of the variations in *y*!
- If R^2 = 0, the estimated regression line is perfectly horizontal. The predictor *x* accounts for *none* of the variations in *y*!

R^2 is indicative of the level of explained variability in the data set.
The closer to 1, the better.

R^2 is the performance metrics (the score) by default in
Linear regression, Ridge regression, Lasso regression sklearn models.

Part 2. Supervised learning

2.2 Classification -

LOGISTIC REGRESSION (LOGIT)

Classification Problem

Email: Spam / NOT Spam?
Online Bank Transaction: Fraudulent (Yes /No) ?
Network traffic : OK/Malware

Binary classification:
y = 1: "positive class" (e.g. Malware traffic)
y = 0: "negative class" (e.g. OK traffic)

Find a model h(x) that outputs values between 0 and 1

$$0 <= h(x) <= 1$$

if h(x) >=0.5, predict "y=1"
if h(x) <0.5, predict "y=0"

Multiclass classification (*K* classes) => y= {0, 1, 2,..}
Build *K* binary classifiers, for each classifier one of the classes has label 1 all other classes take label 0 .

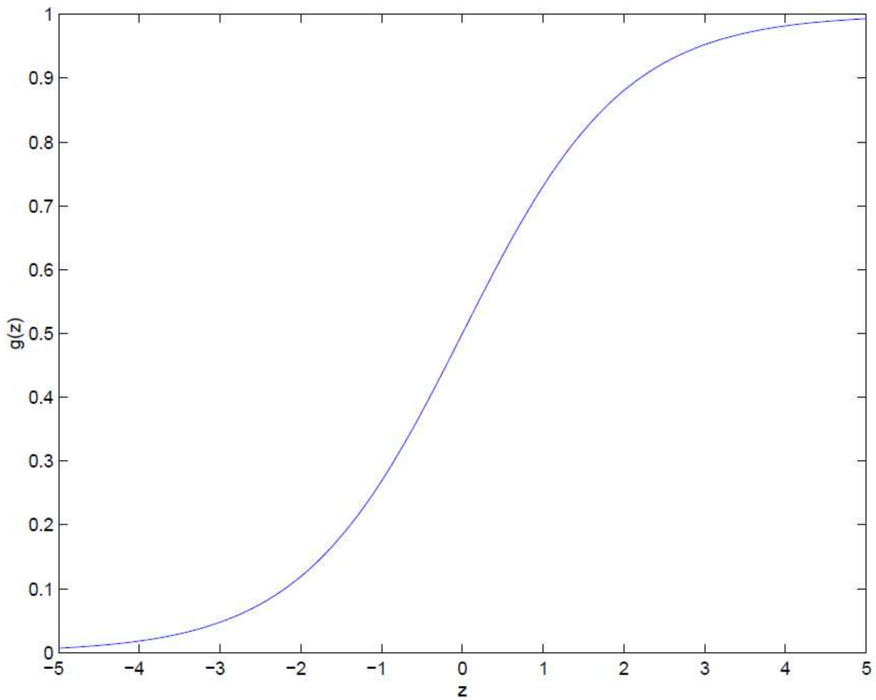
Logistic Regression

Given labelled data of m examples, n features
Labels $\{0,1\}$ => binary classification
x –vector of features; θ – vector of model parameters;
h(x) – logistic (sigmoid) model

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}} = \frac{1}{1+e^{-z}} = g(\theta^T x) = g(z)$$

$$z = \theta^T x = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \dots \theta_n x_n$$

Logistic (sigmoid) function



universidade de aveiro

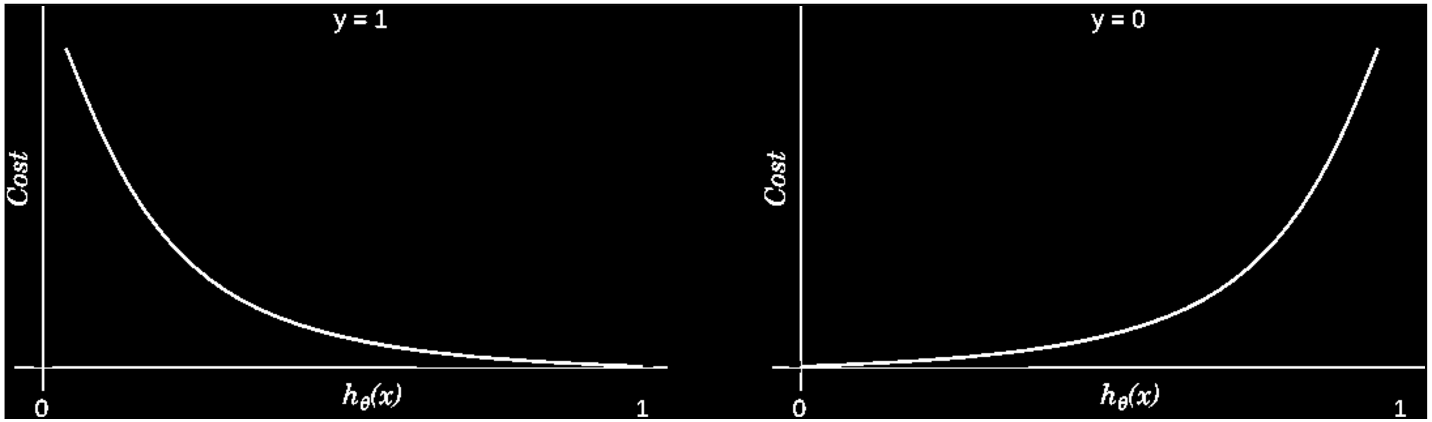
Logistic Regression Cost Function

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \mathrm{Cost}(h_\theta(x^{(i)}), y^{(i)})$$

$$\mathrm{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

Note: $y = 0$ or 1 always

$-\log(h_\theta(x))$ $\qquad$ $-\log(1-h_\theta(x))$



LogReg cost function combined into one expression :
(also known as binary Cross-Entropy or Log Loss function)

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Log Reg with gradient descent learning

Inicialize model parameters *(e.g. θ =0)*
Repeat until J converge {

Compute LogReg Model prediction => $$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Compute LogReg cost function => $$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Goal => $$\min_\theta J(\theta)$$

Compute cost function gradients => $$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

Update parameters => $$\theta_j := \theta_j - \alpha\frac{\partial}{\partial \theta_j}J(\theta)$$

}

alpha – learning rate > 0

Overfitting problem

Overfitting: If we have too many features the learned model may fit the training data very well but fails to generalize to new examples.

$x_1 =$ size of house
$x_2 =$ no. of bedrooms
$x_3 =$ no. of floors
$x_4 =$ age of house
$x_5 =$ average income in neighborhood
$x_6 =$ kitchen size
$\vdots$
x_{100}

Multi-variate regression model:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ... + \theta_n x_n = \vec{\theta}^T \vec{x}$$

How to deal with overfitting problem ?

1. Reduce number of features.
— Manually select which features to keep.

2. Regularization (add extra term in cost function)
Regularization methods shrink model parameters θ towards zero to prevent overfitting by reducing the variance of the model.

2.1 Ridge Regression (L2 norm)
— Reduce magnitude of θ (but never make them =0) => keep all features
— Works well when all features contributes a bit to the output y.

2.2 Lasso Regression (L1 norm)
- May shrink some of the elements of vector θ to become 0.
- Eliminate some of the features => Serve as feature selection

Regularization

Regularization is a popular method in ML to prevent overfitting by reducing the magnitude of the model trainable parameters θ .

1 Ridge Regression (L2 norm)

— Keep all the features, but reduces the magnitude of θ.
— Works well when each of the features contributes a bit to predict y.

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log(h_\theta(x^{(i)})) - (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

2 Lasso Regression (L1 norm)

- May shrink some coefficients of θ to exactly zero.
- Serve as a feature selection tools (reduces the number of features).

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m}\left[-y^{(i)}\log\left(h_\theta\left(x^{(i)}\right)\right) - \left(1-y^{(i)}\right)\log\left(1-h_\theta\left(x^{(i)}\right)\right)\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\left|\theta_j\right|$$

λ – regularization hyper-parameter

Regularized Linear Regression (cost function)

Unregularized cost function =>

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} \left(h_\theta(x^{(i)}) - y^{(i)} \right)^2$$

Regularized cost function
(add extra regularization term
don't regularize θ_0

Ridge Regression

$$J(\theta) = \frac{1}{2m} \left[\sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{n} \theta_j^2 \right]$$

$$\min_\theta J(\theta)$$

λ **> 0**
(alfa in python- sklearn library)

Part 3. Model Performance evaluation

Performance Evaluation – Confusion Matrix

	PREDICTED CLASS		
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	a (TP)	b (FN)
	Class=No	c (FP)	d (TN)

a: TP (true positive)

b: FN (false negative)

c: FP (false positive)

d: TN (true negative)

Python: from sklearn.metrics import confusion_matrix

Performance metric - Accuracy

	PREDICTED CLASS		
ACTUAL CLASS		Class=Yes	Class=No
	Class=Yes	(TP)	(FN)
	Class=No	(FP)	(TN)

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Accuracy - fraction of examples correctly classified.

1-Accuracy: Error rate (misclassification rate)

Limitation of Accuracy

- Consider binary classification (**Unbalanced data set**)
  - Class 0 has 9990 examples
  – Class 1 has 10 examples

- If model classify all examples as class 0, accuracy is 9990/10000 = 99.9 %

- Accuracy is misleading metrics because model does not classify correctly any example of class 1

  =>Use other performance metrics.

  => Find a way to balance the data set

(re-sampling methods: oversampling, under-sampling)

Performance metrics from Conf Matrix

True Positive Rate (TPR), Sensitivity, Recall
of all positive examples the fraction of correctly classified
(ex. skin cancer)

$$TPR = \frac{TP}{TP + FN}$$

True Negative Rate (TNR), Specificity
of all negative examples the fraction of correctly classified
(ex. spam/not spam emails)

$$TNR = \frac{TN}{TN + FP}$$

False Positive Rate (FPR) - how often an actual negative instance will be classified as positive, i.e. “false alarm” (ex. cyber attack)

$$FPR = 1 - TNR = \frac{FP}{FP + TN}$$

Precision - the fraction of correctly classified positive samples from all classified as positive

$$\text{Precision} = \frac{TP}{TP + FP}$$

universidade de aveiro

Combined performance metrics

F1 Score - weighted average of Precision and Recall

F1=2*(Recall * Precision) / (Recall + Precision)

Balanced Accuracy= (Recall+Specificity)/2

Deciding what to do next ?

Suppose you have trained a ML model on some data. When you test the trained model on a new set of data, it makes unacceptably large errors. What should you do ?

-- Get more training examples ?
-- Try smaller sets of features (feature selection) ?
-- Try getting additional features (feature engineering) ?
-- Try using different/nonlinear kernels ?
-- Try other values of the hyper parameters (e.g. regul. parameter) ?

Machine learning diagnostics = Model-centric approach

Run tests to gain insight what isn't working with the learning algorithm and how to improve its performance.
Diagnostics is time consuming , but can be a very good use of your time.

Simplest division: Train & Test subsets

- Training set (70%-80 %) : used to train the model
- Test set (30%-20%) : used to test the trained model

- **Optimize the model parameters with training data**
  (minimize some cost/loss function J)

<u>After the training stage is over (i.e. the cost function J converged)</u>

- Compute the MSE on test data (for regression problems)

$$E_{test}(\theta) = \frac{1}{m_{test}} \left[\sum_{i=1}^{m_{test}} \left(h_\theta\left(x_{test}^{(i)}\right) - y_{test}^{(i)} \right)^2 \right]$$

or

- **Compute the model accuracy or some other metric from the confusion matrix, on test data (for classification problems)**

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

universidade de aveiro

3 way split: Train/Dev/Test Sets

Choose ML model: Logistic Regression, Neural Network (NN), etc. ?
Choose model hyper-parameters:
- # of layers in NN ?
- What is the best learning rate ?
- What is the best regularization parameter (λ) ?
- What is the best degree of the polynomial regression model ?
- ……

Devide dataset in 3 sub-sets:
- Training set
- Cross Validation (CV) set = Development set = 'dev' set
- Test set

Traditional division for Small data set (up to 10000 examples) :
60% - 20% - 20%

Big data (1 million. examples): 98% - 1% - 1%

Model /hyper parameter selection

Step 1: Optimize parameters θ (to minimize some cost function J) using the same training set for all models. Compute some perf. metrics with the training data (i.e. error, accuracy) :

Training error =>

$$E_{train}(\theta) = \frac{1}{2m}\left[\sum_{i=1}^{m}\left(h_\theta\left(x^{(i)}\right) - y^{(i)}\right)^2\right]$$

Step 2: Test the optimized models from step 1 with the CV set and choose the model with the min CV error (or other performance metric with dev data):

Cross validation (CV)/dev error =>

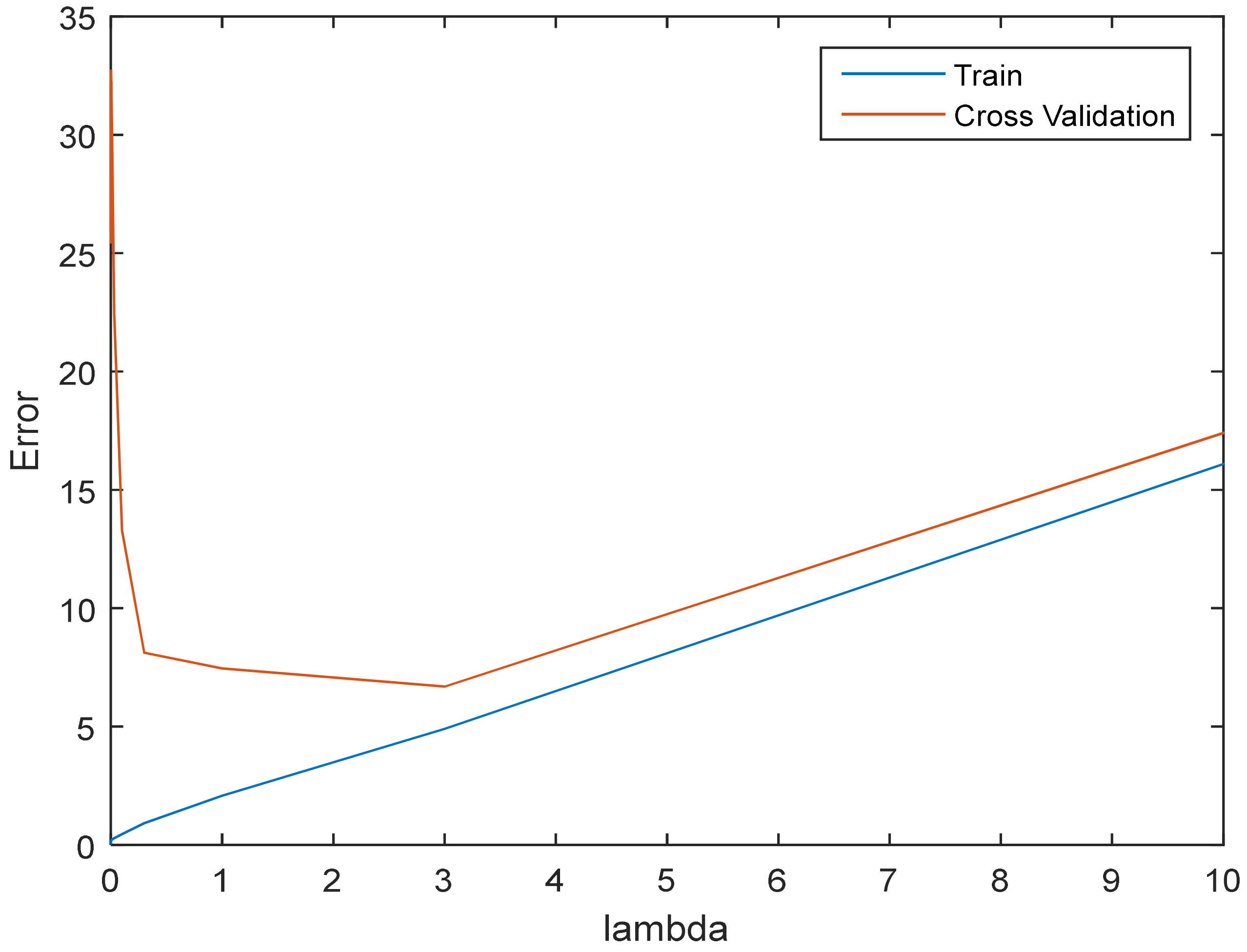
$$E_{cv}(\theta) = \frac{1}{2m_{cv}}\left[\sum_{i=1}^{m_{cv}}\left(h_\theta\left(x_{cv}^{(i)}\right) - y_{cv}^{(i)}\right)^2\right]$$

Step 3: Retrain the best model from step 2 with both train and CV sets starting from the parameters got at step 2. Test the retrained model with test set and compute test data perf. metric (***the real model performance !!!***):

Test error =>

$$E_{test}(\theta) = \frac{1}{2m_{test}}\left[\sum_{i=1}^{m_{test}}\left(h_\theta\left(x_{test}^{(i)}\right) - y_{test}^{(i)}\right)^2\right]$$

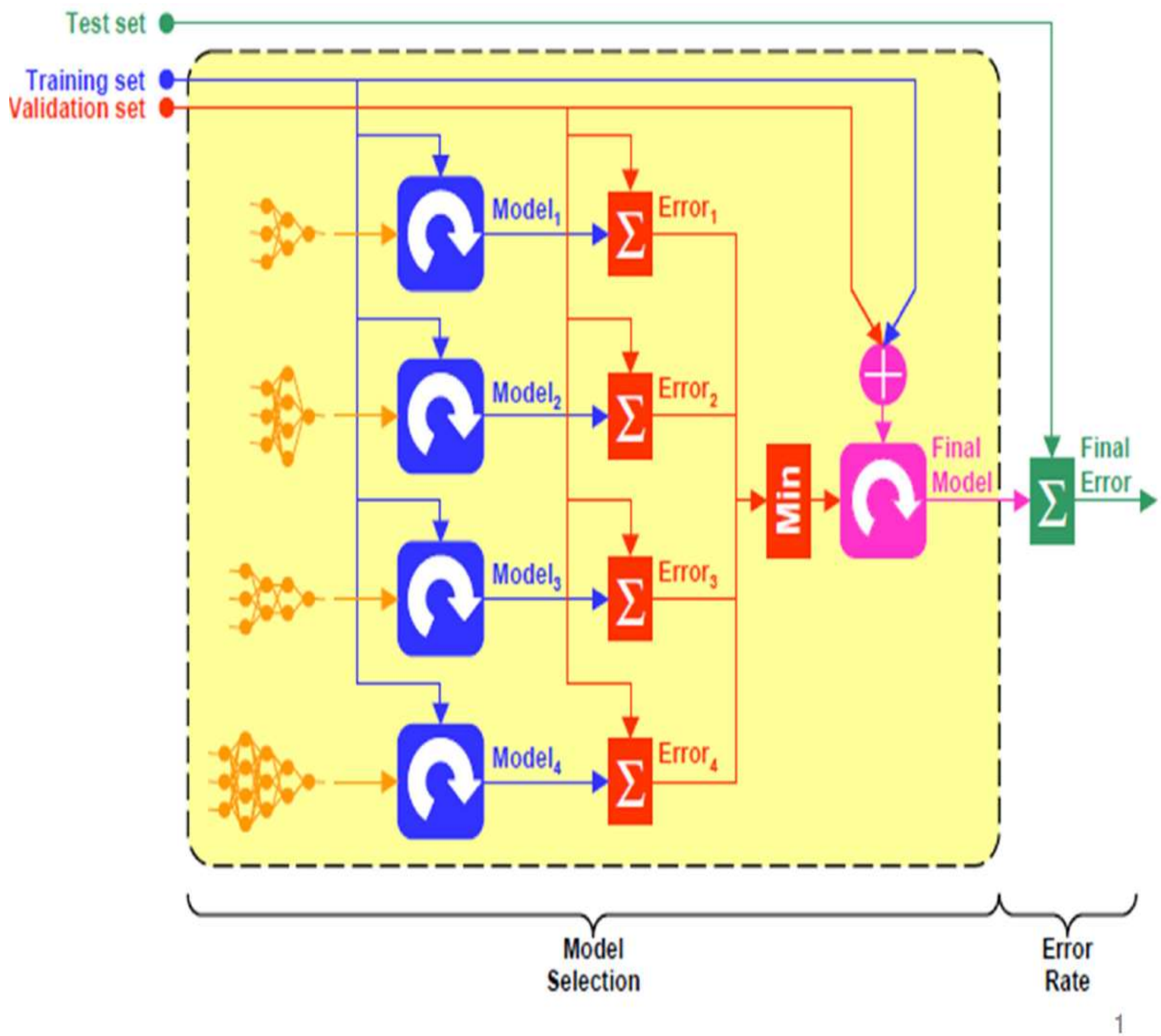
Example: Select best λ



Best λ =3

universidade de aveiro

Training/Valid (Dev)/Test subsets



The most credible is the performance metric with test data, not used for training or validation of the model.

K –fold Cross Validation

- Divide data into Training and Test subsets.
- Split Training data into K subsets (K folds).
- Use K-1 folds for training and the remaining fold for validation.
- The final validation error is the average error of K trainings.
- Choose the best model or the best hyper-parameter

the one that minimises the validation error.