NOVA IMS

Information Management School

5

MODELING: REGRESSION

Machine Learning for Marketing

Acreditações e Certificações

Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa

Summary

1. Introduction
2. Models' validation
3. Generalization and overfitting
4. Regression: measures of performance
5. Regression algorithms: linear regression
6. Regression algorithms: decision trees
7. Regression algorithms: linear neural networks
8. Parameters hyper tunning

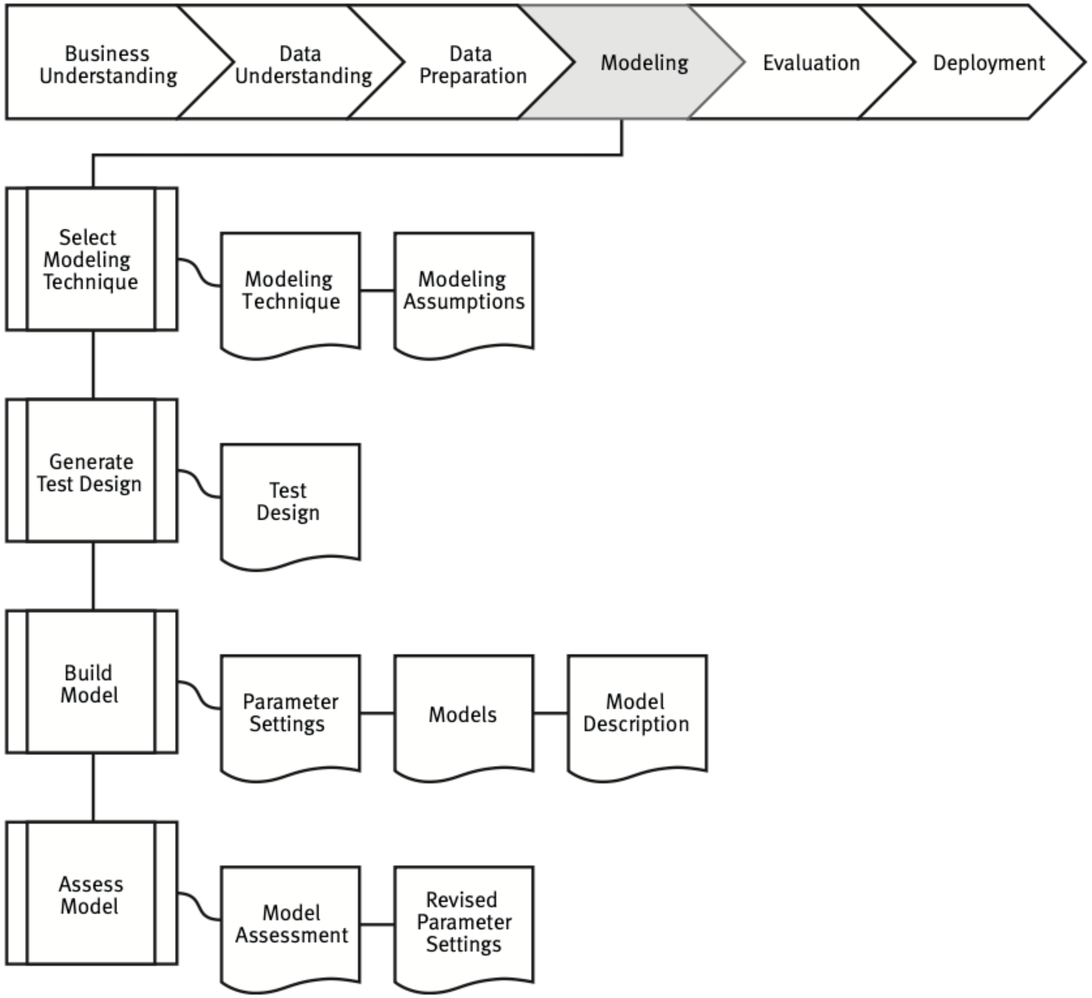
NOVA IMS
Information Management School

5.1

Introduction

Modeling: Regression

Modeling

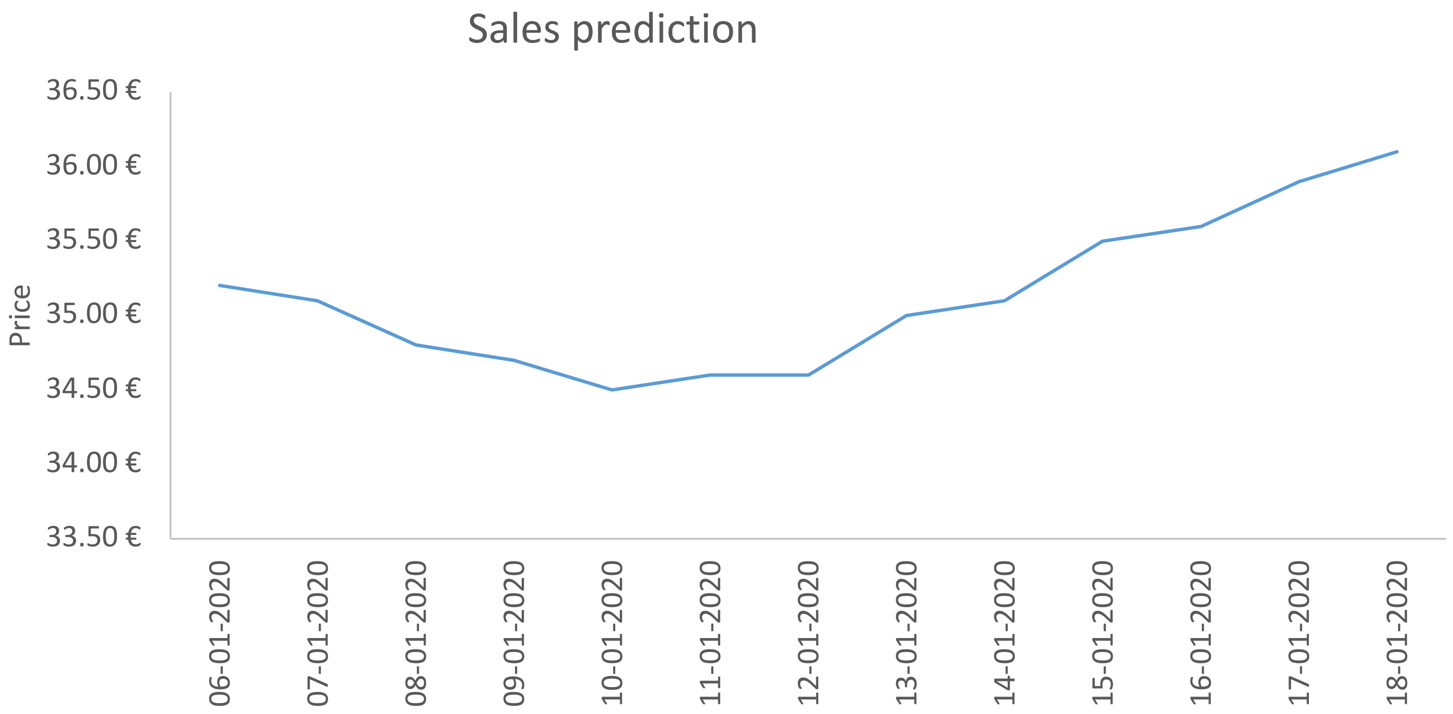

Business Understanding
Data Understanding
Data Preparation
Modeling
Evaluation
Deployment
Select Modeling Technique
Modeling Technique
Modeling Assumptions
Generate Test Design
Test Design
Build Model
Parameter Settings
Models
Model Description
Assess Model
Model Assessment
Revised Parameter Settings

Supervised Learning

- Uses **input** and **output** attributes to create a function to map inputs to outputs
- Employs a **TRAINING** dataset with M input/output attribute pairs: $(x_1, y_1), (x_2, y_2), \ldots (x_M, y_M)$
- Where y_j is generated by unknown function ***y = f (x)***
- **The goal** is to find a function **h** (hypothesis) that is close to the function **f** (an unknown function)
- The accuracy of a hypothesis is measured in a **TEST** dataset
- A hypothesis is said to **GENERALIZE** well when it correctly predicts the value of y in new observations

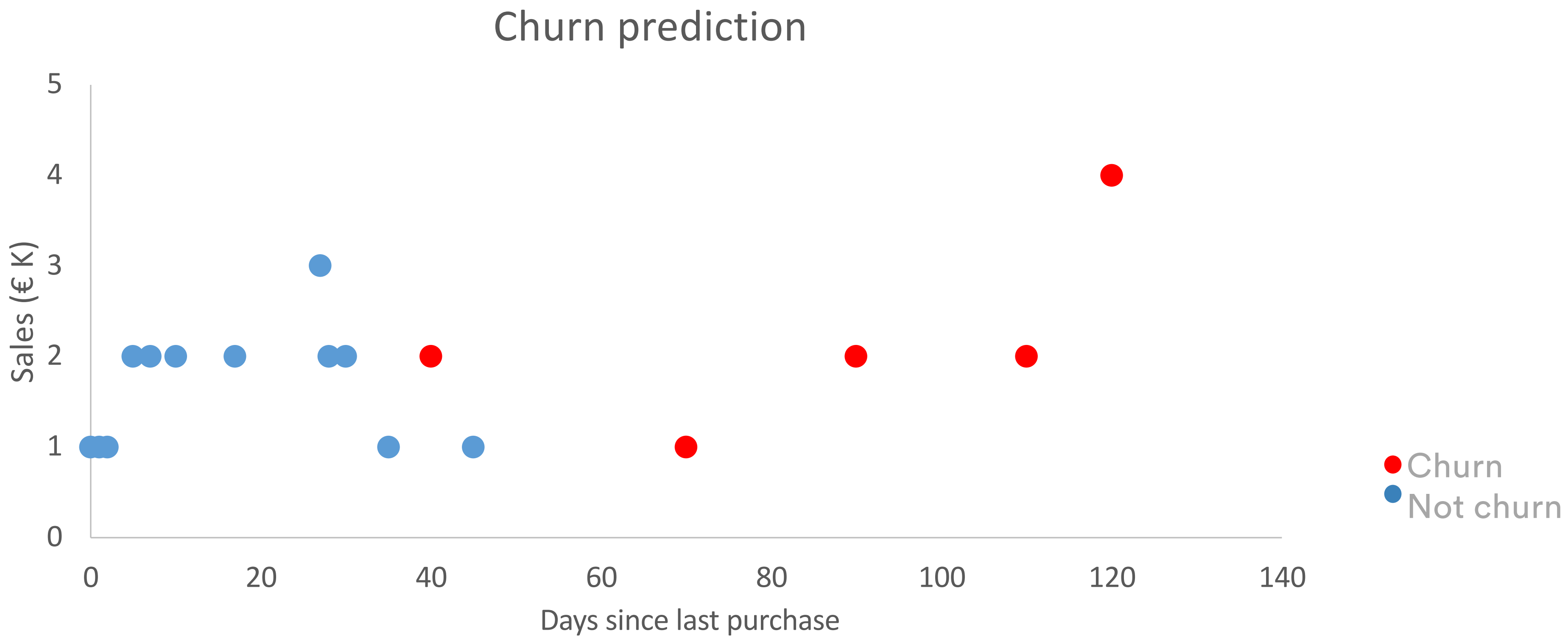
Regression

Predict **continuous** output

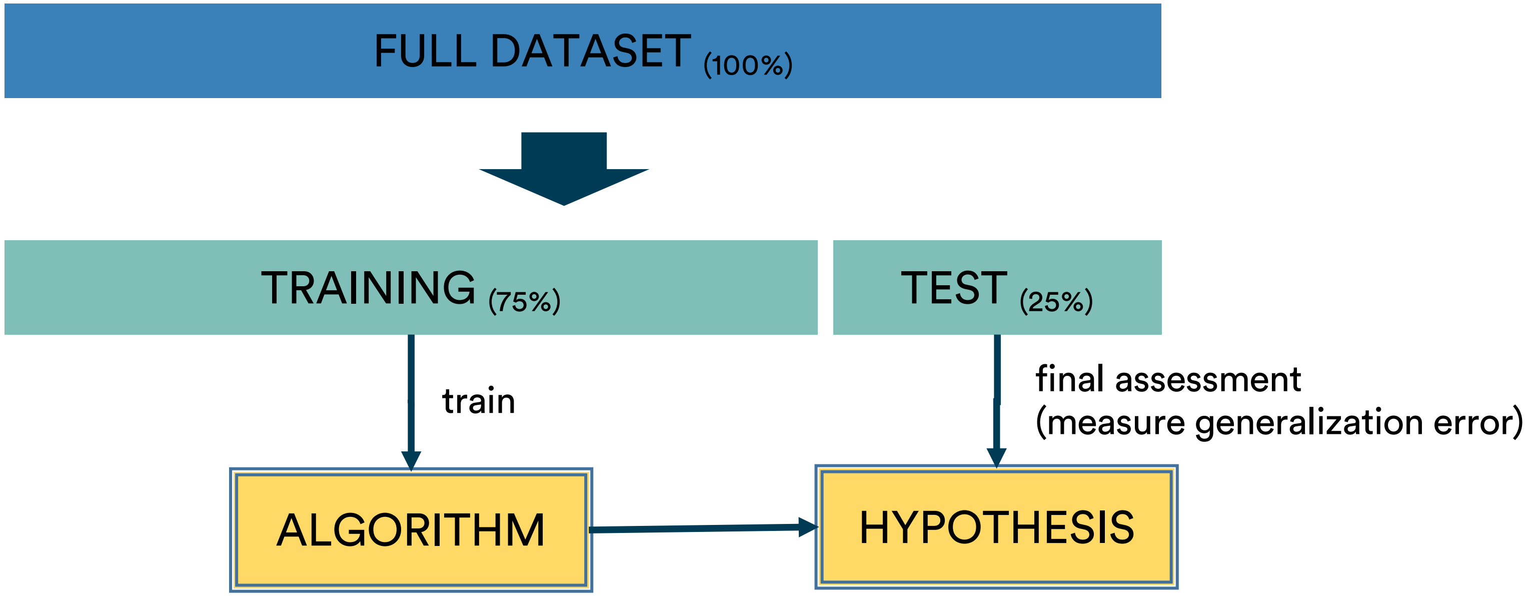


Classification

Predict **discrete** output



Hypothesis development



NOVA IMS
Information Management School

5.2

Models' validation

Modeling: Regression

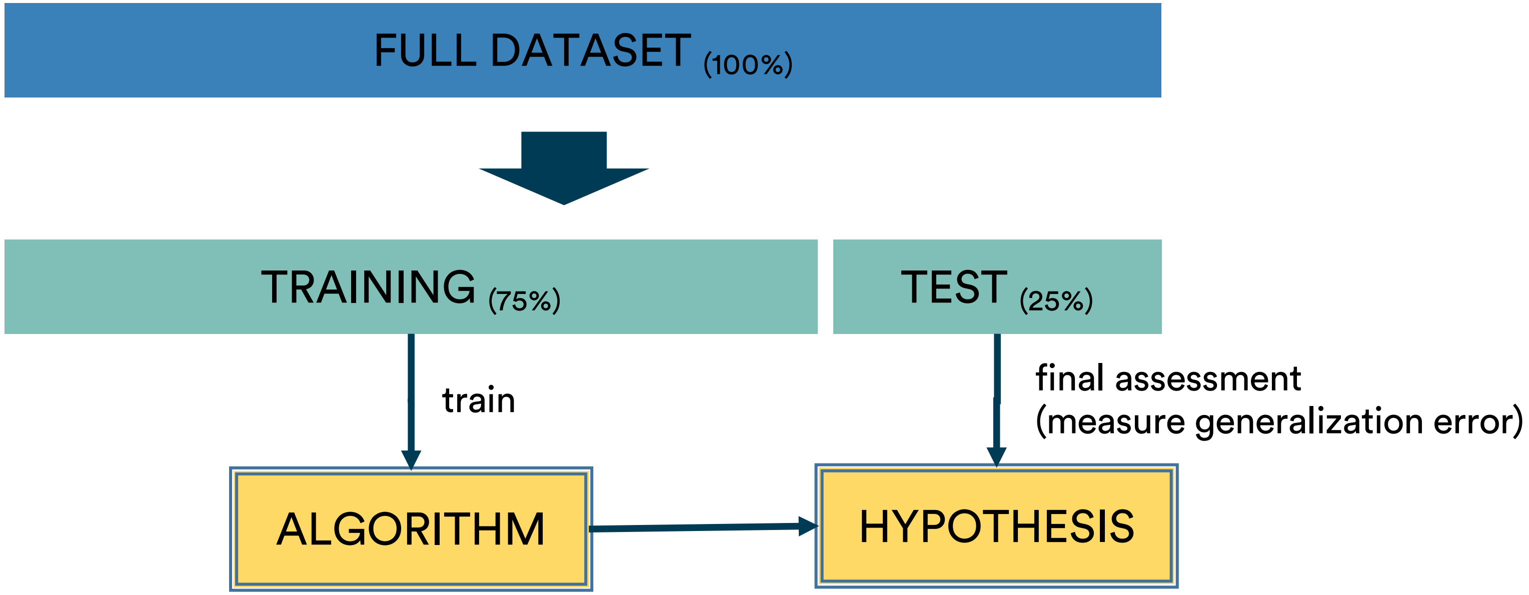
Performance measurement

Error rate: Proportion of errors that a hypothesis has – proportion of times *h(x) ≠ y* for a pair of attributes *(x,y)*

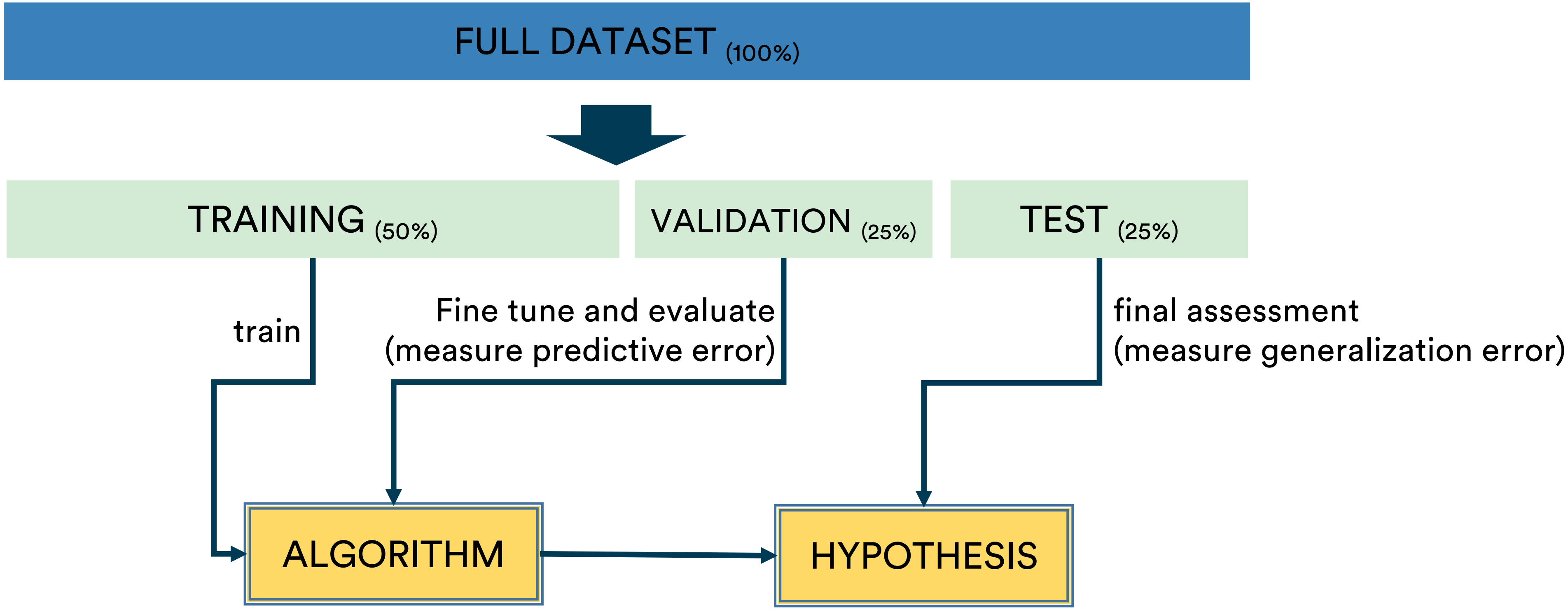
The more common techniques for error rate evaluation are:

- Holdout (2 and 3 splits)
- K-fold cross validation (CV)

Holdout (2 splits)



Holdout (3 splits)



FULL DATASET (100%)

TRAINING (50%)

VALIDATION (25%)

TEST (25%)

train

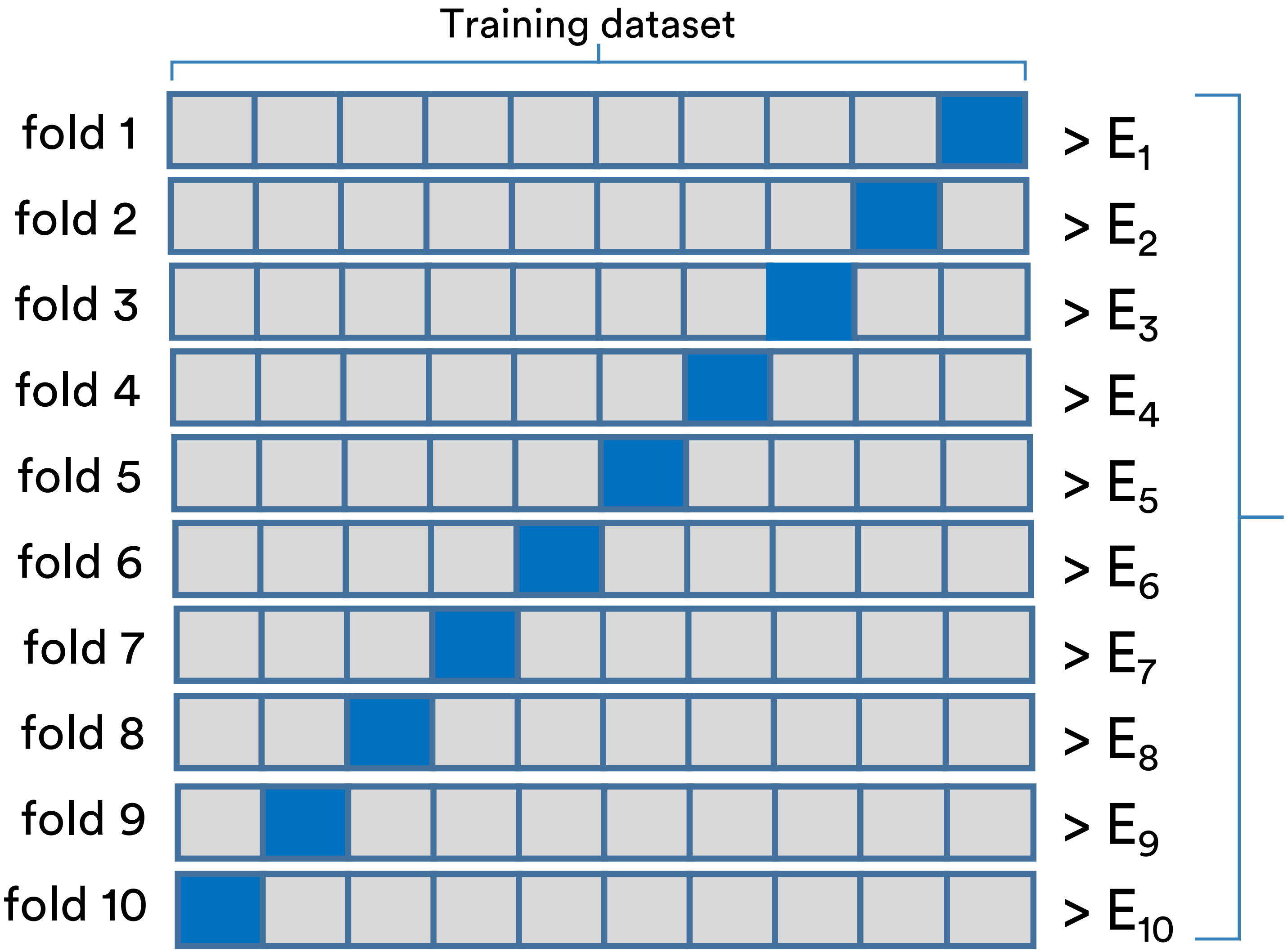
Fine tune and evaluate
(measure predictive error)

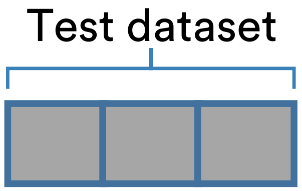
final assessment
(measure generalization error)

ALGORITHM

HYPOTHESIS

K-Fold cross validation





⬜ Training folds

🟦 Test fold

$$E = \frac{1}{10}\sum_{i=1}^{10} E_i$$

Example for K=10

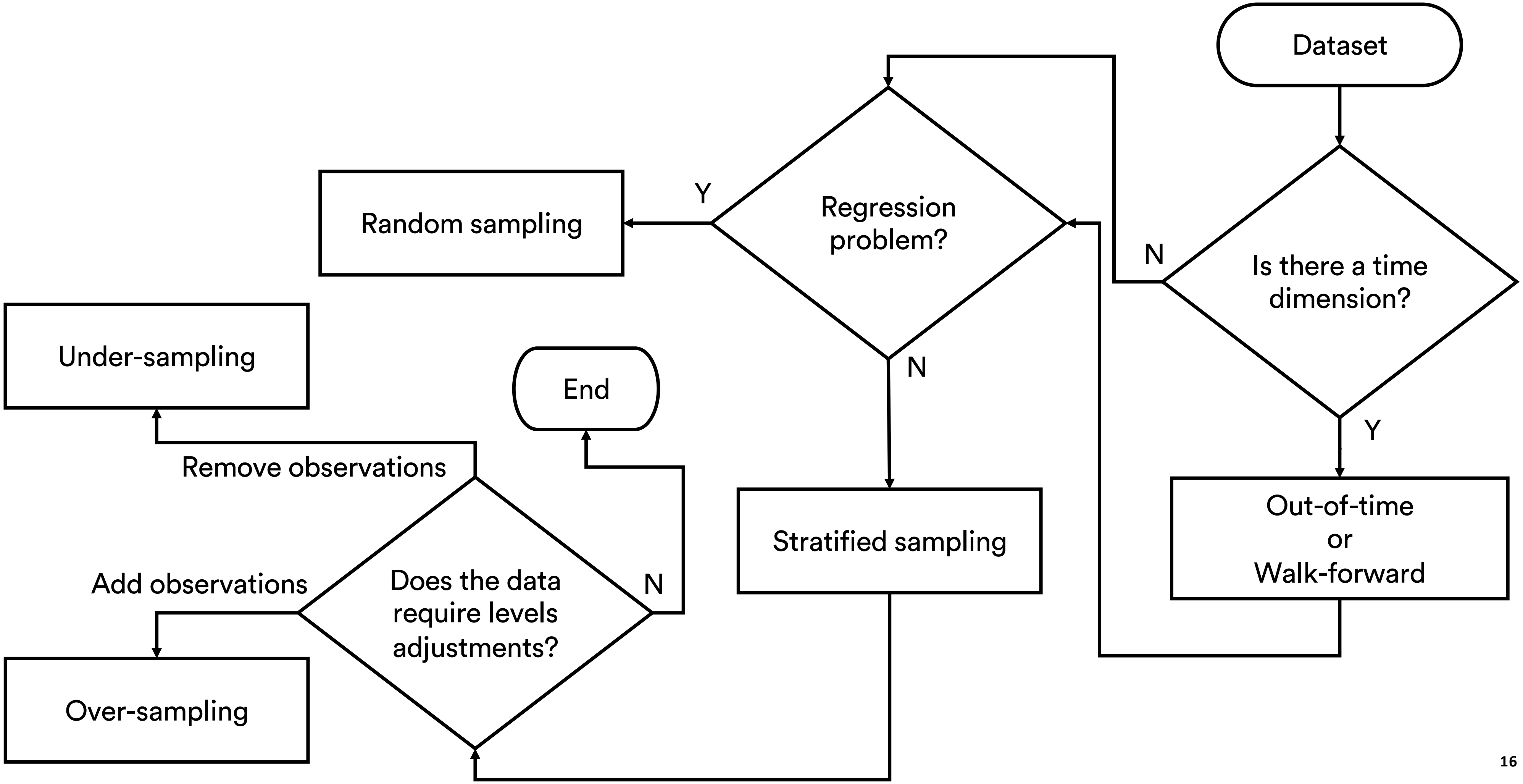
Steps in implementing K-Fold cross validation

1. The dataset is split into training and test dataset
2. The training dataset is (randomly) split into K-folds
3. Out of the K-folds, (K-1) fold is used for training
4. One-fold is used for validation
5. The model with specific hyperparameters is trained with training data (K-1 folds) and validation data as 1-fold. The performance of the model is recorded
6. Steps 3, 4, and 5 are repeated until each of the k-fold got used for validation purposes
7. Finally, the mean and standard deviation of the model performance is computed by taking all of the model scores calculated in step 5 for each of the K models
8. Steps 3 to 7 are repeated for different values of hyperparameters
9. Finally, the hyperparameters which result in the most optimal mean and the standard value of model scores get selected
10. The model is then trained using the training data set (step 2), and the model performance is computed on the test data set (step 1)

Sampling methods

- **Random sampling:** selects a random percentage of instances
- **Stratified sampling:** selects instances accordingly the relative frequencies of the levels of a specified stratification feature. This selection ensures that the sample presents a distribution similar to the population
- **Out-of-time:** selects instances based on the time dimension
- **Under-sampling:** used to adjust the the number of instances by the levels of a particular feature. Random under-sampling selects all instances of the *smallest* level, and then do random sampling for the same number of instances then the smaller level for the remaining levels
- **Over-sampling:** the opposite of under-sampling. In this case, selects all the instances from the *largest* level

Sampling methods – Basic selection diagram



Dataset

Is there a time dimension?

Y

Out-of-time
or
Walk-forward

N

Regression problem?

Y

Random sampling

N

Stratified sampling

Does the data require levels adjustments?

Remove observations

Under-sampling

Add observations

Over-sampling

N

End

Under/Over-sampling techniques

UNDER-SAMPLING

- Random under-sampling
- Cluster
- Tomek links
- Ensemble learning

OVER-SAMPLING

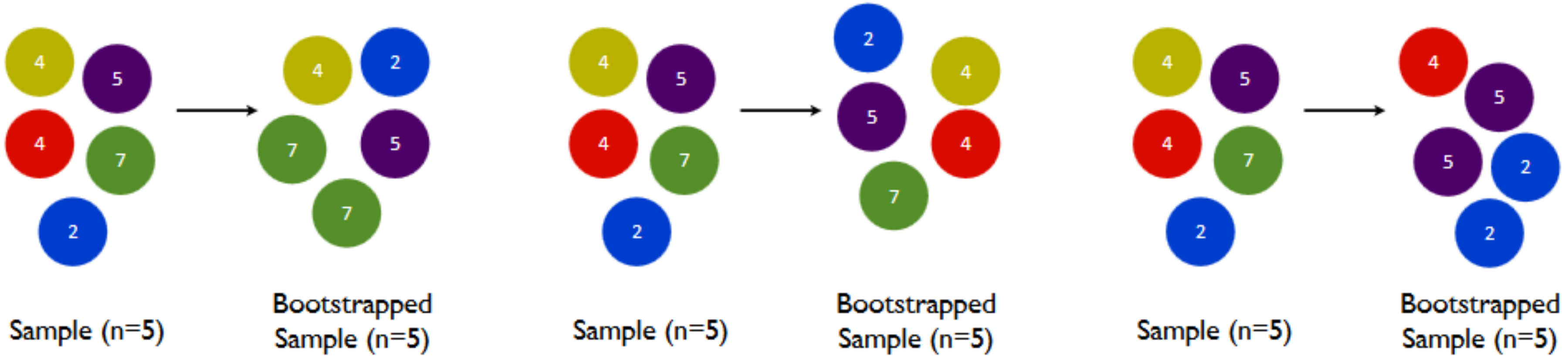
- Random oversampling
- SMOTE (Synthetic Minority Over-sampling Technique)
- SOMO (Self-Organizing Map Oversampling)
- ADASYN (Adaptive Synthetic Sampling Approach)

For more details check the “imbalanced-learn” package
https://imbalanced-learn.readthedocs.io/en/stable/index.html

Other methods: Bootstrap

Commonly applied with small data. Samples the data randomly, **with replacement.**

Usually done dozens or hundreds of times. On average, ~63% will be selected, leaving the remaining ~37% as the hold-out sample.


4
5
4
7
2
4
2
7
5
7
Sample (n=5)
Bootstrapped Sample (n=5)
4
5
4
7
2
2
4
5
4
7
Sample (n=5)
Bootstrapped Sample (n=5)
4
5
4
7
2
4
5
5
2
2
Sample (n=5)
Bootstrapped Sample (n=5)

[https://i.stack.imgur.com/5Nzqf.png]

5.3

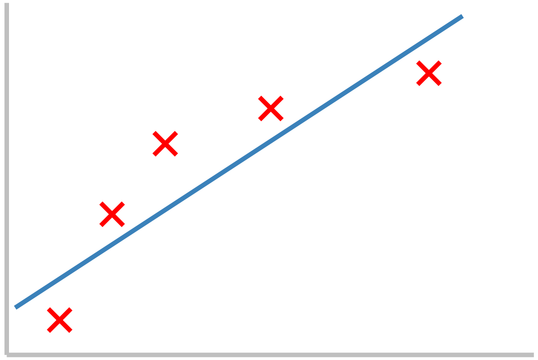
Generalization and Overfitting

Modeling: Regression

Generalization and Overfitting

Overfitting occurs when a model corresponds too closely or exactly to a dataset and therefore may fail to **generalize**, i.e., to predict future observations reliably (for example, if there are too many features, the model can “memorize” the data)

Underfitting vs Overfitting



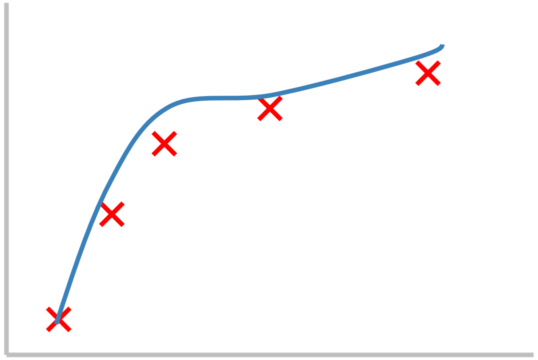
Underfit

High bias

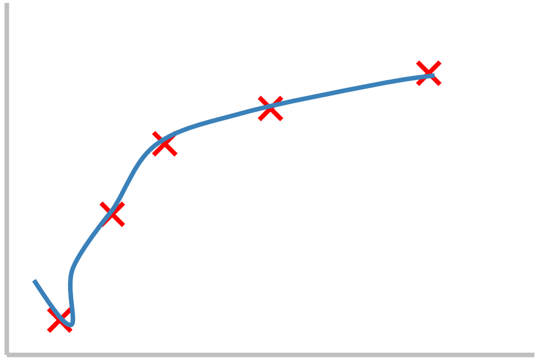
Bad performance on training data

Average predictions are far away from actual values

CV error ~ train error



Optimal



Overfit

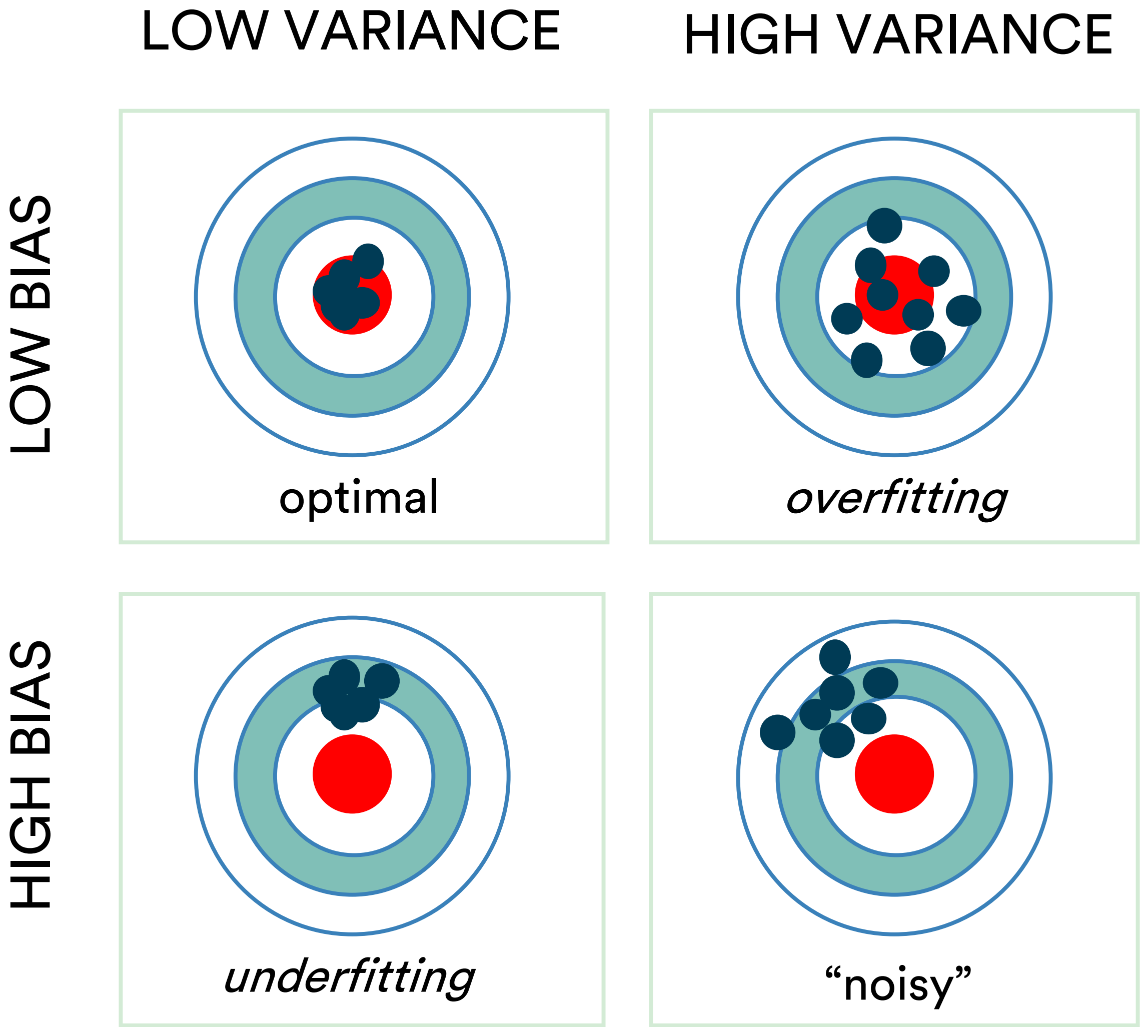
High variance

Capture all noise in the data

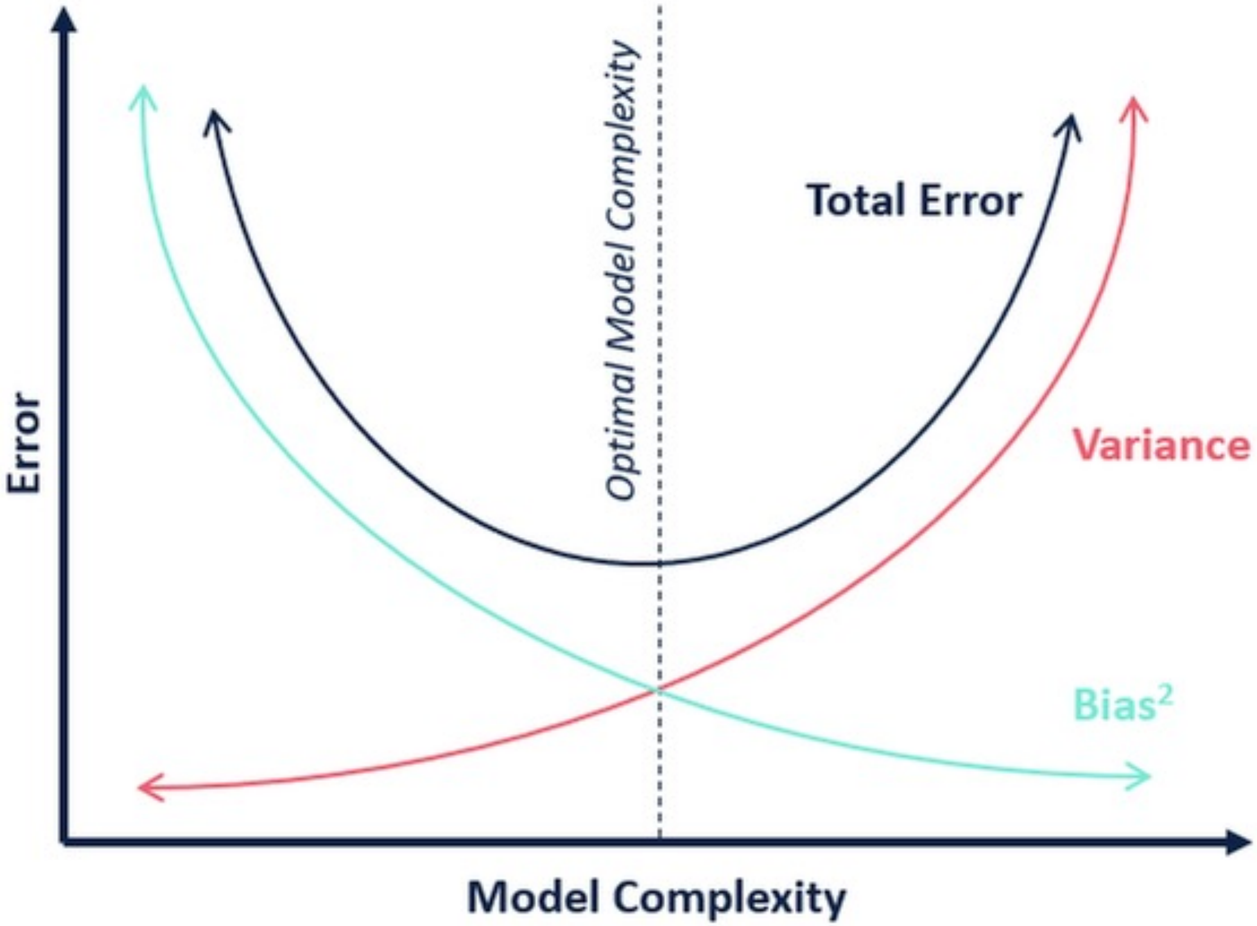
Good performance on training data but not on test data

CV error >> train error

Bias-Variance

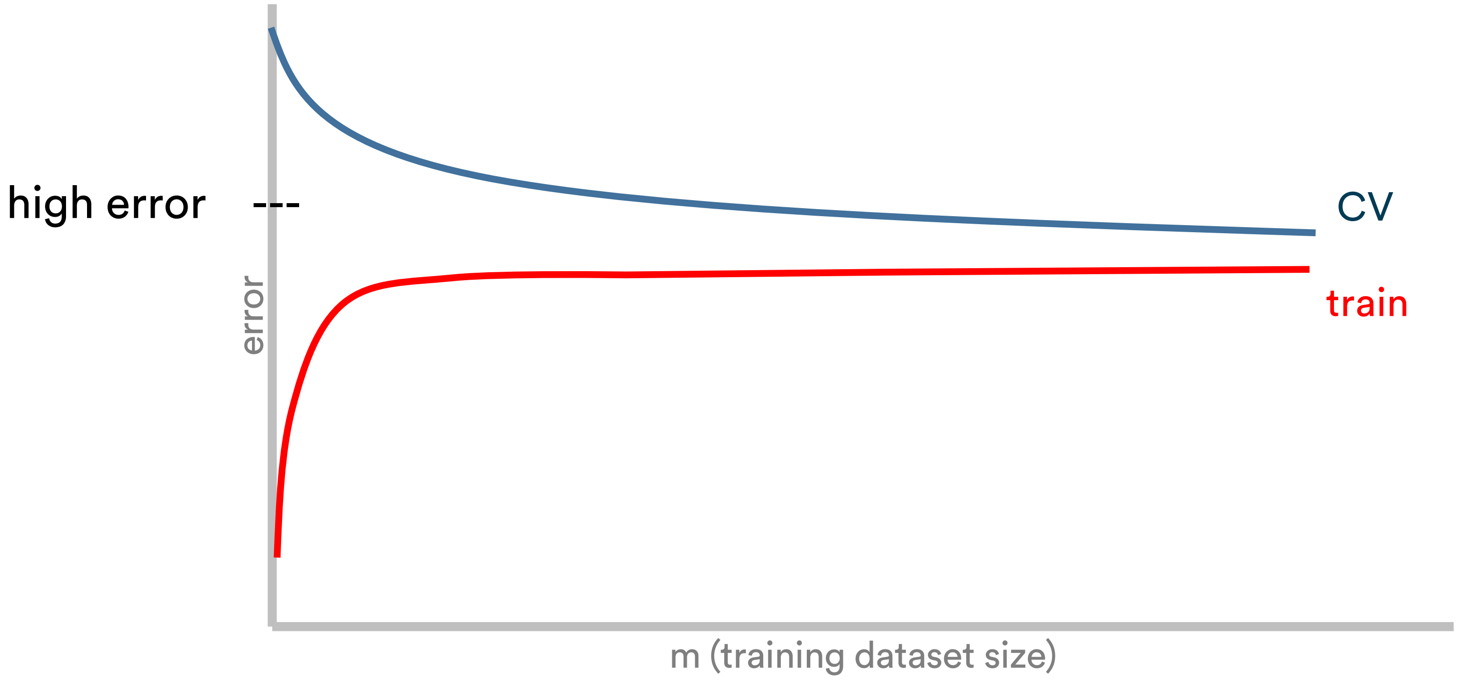


Bias-Variance tradeoff



https://www.analyticsvidhya.com

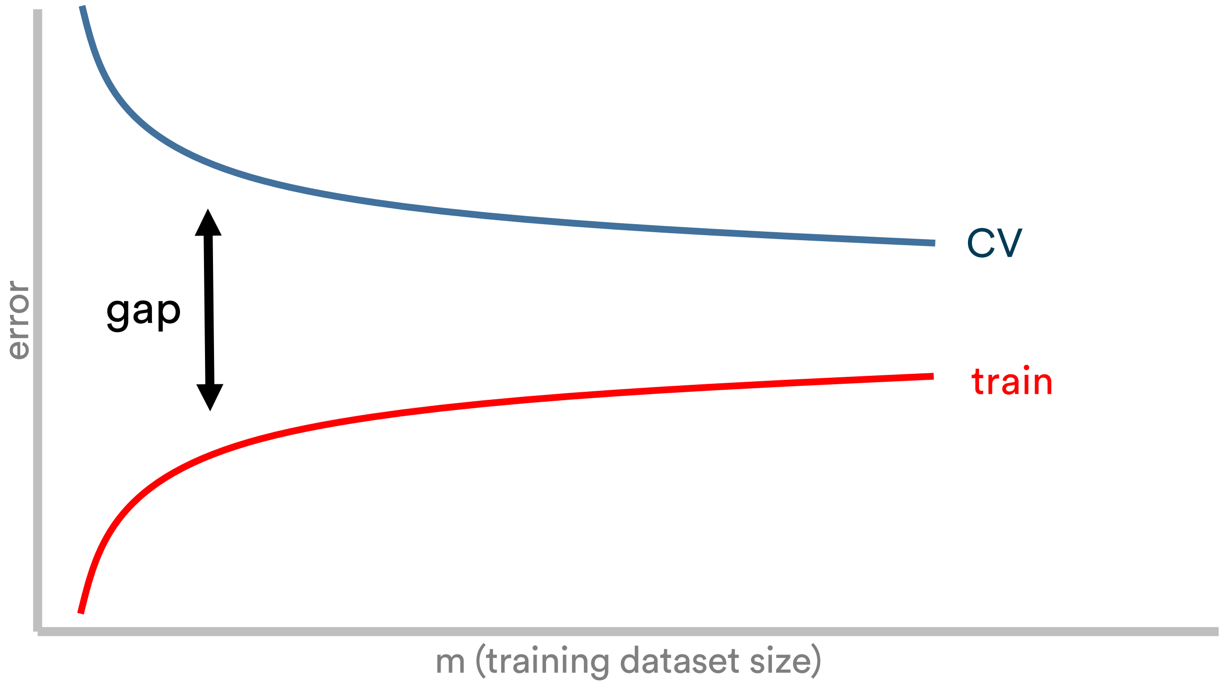
High Bias / Underfitting



How to solve underfitting

- Increase the number of features (for the model to capture the intricacy of the dataset)
- Create new re-engineered features (e.g., polynomials, ratios, etc.)
- Increase the number of observations
- Use K-fold cross-validation to find any bias it may exist

High Variance



How to solve overfitting

- Obtain more observations/instances (if possible)
- Reduce the number of features
  - Select features to keep (e.g., Chi-squared, Pearson correlation, etc.)
- Change modeling algorithm
- Regularization
  - Keep all features but penalize more complex hypothesis
- Employ ensemble models, but from different families

5.4

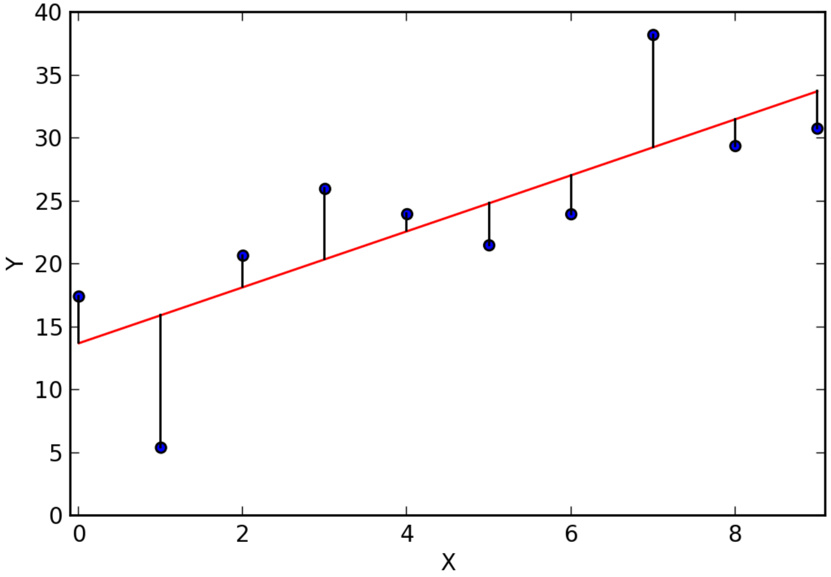
Regression: Measures of performance

Modeling: Regression

Residuals

- Are in the base of several performance measures
- Residuals are the difference between the observed value and the predicted value
- In regression, residuals should be normally distributed

$$Residual = y - \hat{y}$$



[https://commons.wikimedia.org]

Measures of performance

$$Mean\ Absolute\ Error\ (MAE) = \frac{1}{m}\sum_{j=1}^{m}\left|y_j - \hat{y}_j\right|$$

- Compute the absolute value before averaging
- Provides an average magnitude of the error, in the unit under analysis

Measures of performance

$$Mean\ Squared\ Error\ (MSE) = \frac{1}{m}\sum_{j=1}^{m}\left(y_j - \hat{y}_j\right)^2$$

- Average error
- If predictions are unbiased, this should be equal to 0

Measures of performance

$$Root\ Mean\ Squared\ Error\ (RMSE) = \sqrt{\frac{1}{m}\sum\nolimits_{j=1}^{m}\left(y_j - \hat{y}_j\right)^2}$$

- Compute the square root of MSE
- Error magnitude in the unit under analysis
- Penalizes outliers (in relation to MAE)

Ex.1: $y = [2,4,6,8]$, $\hat{y} = [4,6,8,10]$ MAE=2.0, RMSE=2.0

Ex.2: $y = [2,4,6,8]$, $\hat{y} = [4,6,8,12]$ MAE=2.5, RMSE=2.65

Measures of performance

$$R^2 = 1 - \frac{\sum_{j=1}^{m}\left(y_j - \hat{y}_j\right)^2}{\sum_{j=1}^{m}\left(y_j - \bar{y}\right)^2}$$

- Percent variance explained
- Employed for explanatory purposes. Explains how the independent variable(s) explain the dependent variable(s) variability

Measures of performance

$$Mean\ Absolute\ Percentage\ Error\ (MAPE) = \frac{1}{m}\sum_{j=1}^{m}\left|\frac{y_j - \hat{y}_j}{y}\right|$$

- Percentage equivalent of MAE
- Provides a relative size of the error compared to the actual target value

Measures of performance

$$Mean\ Percentage\ Error\ (MPE) = \frac{1}{m}\sum_{j=1}^{m}\left(\frac{y_j - \hat{y}_j}{y}\right)$$

- Similar to MAPE, but without being an absolute value
- Useful to show if there is any bias in terms of positive or negative errors

5.5

Regression algorithms: Linear regression

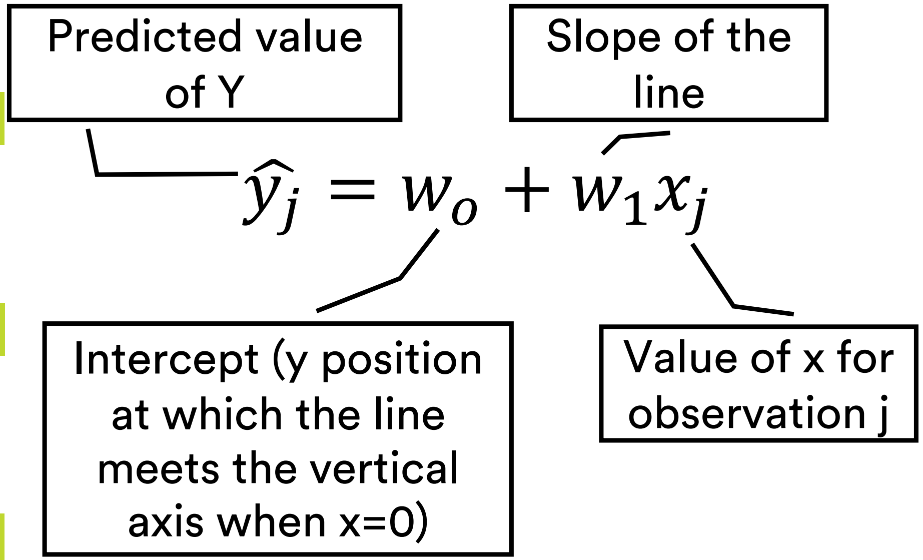
Modeling: Regression

Introduction

Regression algorithms: Linear regression

Linear regression

Attempts to model the relationship between two variables by fitting a linear equation to the observed data.



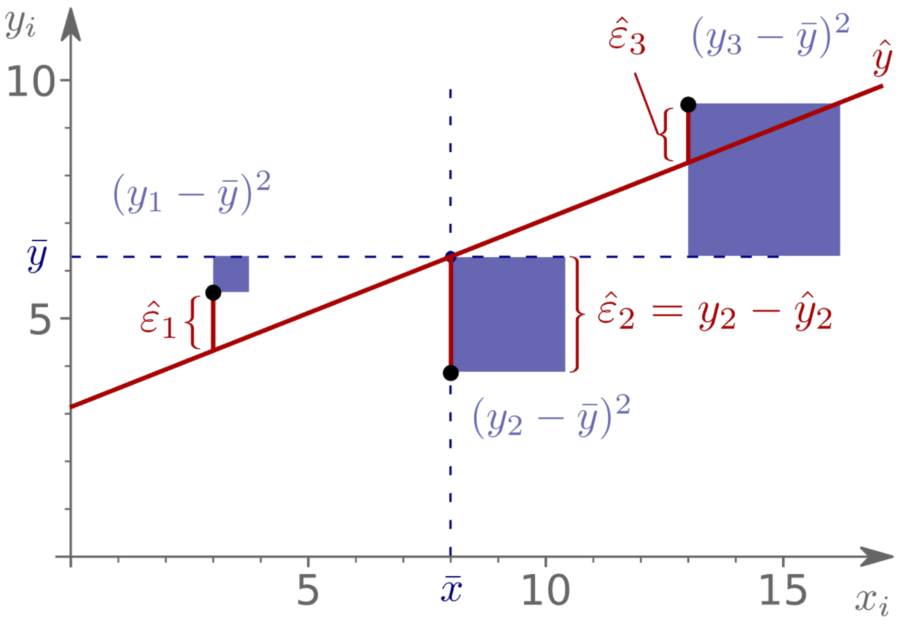
Predicted value of Y

Slope of the line

$$\widehat{y_j} = w_o + w_1 x_j$$

Intercept (y position at which the line meets the vertical axis when x=0)

Value of x for observation j



[https://commons.wikimedia.org]

Multivariate linear regression

$$\hat{y} = w_o + w_1x_1 + w_2x_2 + \cdots + w_nx_n$$

Where:

- $\hat{y}$ is the estimated value (output)
- w_o, w_1, w_2, w_n are the coefficients
- x_o, x_1, x_2, x_n are the inputs

Linear regression assumptions

- The relationship between the input variables and the output variable is assumed to be linear
- Inputs are uncorrelated with each other
- Inputs are normally distributed

Least squares optimization

Regression algorithms: Linear regression

Least squares optimization

Is the most common method for finding weights (fitting a regression line).

This method calculates the best-fitting line for the observed data by minimizing the sum of the squares of the residuals, i.e., the weights that generate the lowest sum of squared errors (**global minimum**).

$$Loss(h_w) = \sum_{j=1}^{m}\left(y_j - h_w(x_j)\right)^2 = \sum_{j=1}^{m}\left(y_j - \left(w_0 + w_1 x_j\right)\right)^2$$

Gradient descent algorithm (univariate)

One of the most important algorithms employed in machine learning. One of its uses is to find the weights for linear regression.

- Cost function:

$$J(w_0, w_1) = \frac{1}{2m}\sum_{j=1}^{m}\left(h_w\left(x_j\right) - y_j\right)^2$$

$$h_w\left(x_j\right) = w_0 + w_1 x_j$$

- The cost function is minimized when the partial derivatives in relation to w_0 and w_1 are zero

Gradient descent algorithm (univariate)

- Repeat until convergence:

$$w_j := w_j - \boxed{\alpha}\boxed{\frac{\partial}{\partial w_j} J(w_0, w_1)}$$

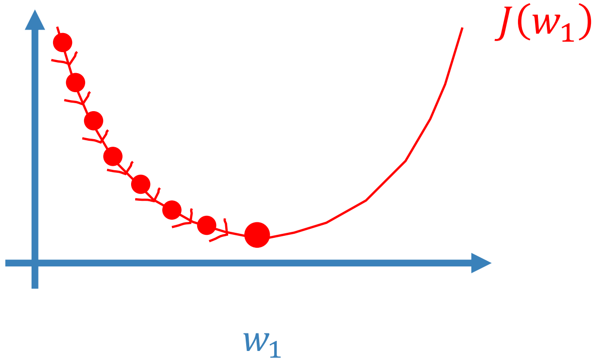
Learning rate

derivative

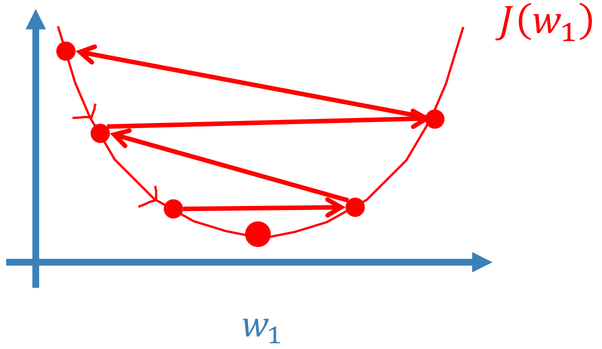
- $\min\big(J(w_o)\big) = 0$ therefore only $\min\big(J(w_1)\big)$ should be calculated

Gradient descent algorithm (univariate)

If α too small, *gradient descent* can be slow

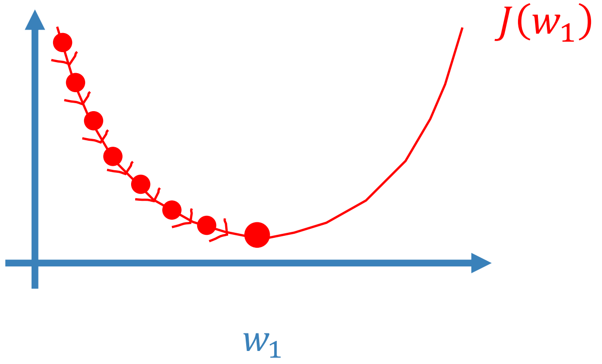


If α too large, *gradient descent* can overpass the minimum. Instead of converging, it can diverge

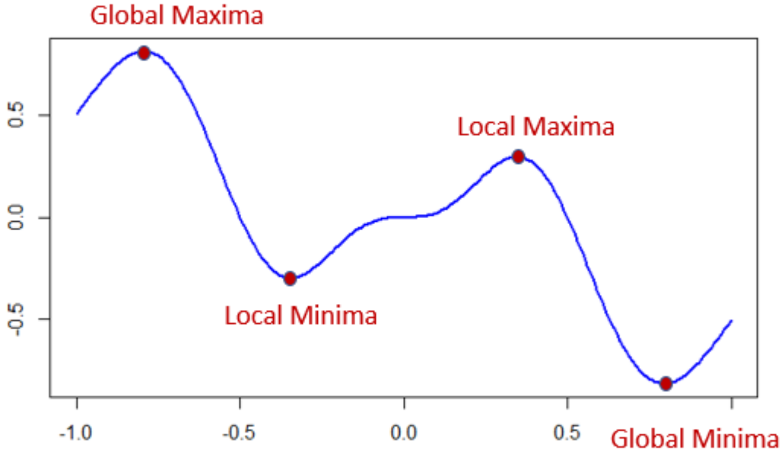


Gradient descent algorithm (univariate)

Even with a fixed *learning rate* α *gradient descent* can converge to the *local minimum*



As it approaches a *local minimum*, *gradient descent* automatically takes shorter steps. Therefore, it is never necessary to reduce α over the course of iterations



[https://www.i2tutorials.com/]

Multivariate linear regression

$$x = \begin{bmatrix} x_0 \\ \cdots \\ x_n \end{bmatrix}, where\ x_0 = 1 \qquad w = \begin{bmatrix} w_0 \\ \cdots \\ w_n \end{bmatrix}$$

The hypothesis is defined by:

$$h_w(x) = w_0x_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n = \ w^Tx$$

The cost function is:

$$J(w) = \frac{1}{2m}\sum_{j=1}^{m}\left(h_w(x_j) - y_j\right)^2$$

Matrices – Linear algebra (1/2)

- Identity matrix: $\mathrm{I} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$

- Inverse matrix. If A is a matrix m x m and it has an inverse matrix $A(A^{-1}) = (A^{-1})A = I$

$$A = \begin{bmatrix} 3 & 4 \\ 2 & 16 \end{bmatrix} \quad A^{-1} = \frac{1}{3\times16 - 4\times2} \begin{bmatrix} 16 & -4 \\ -2 & 3 \end{bmatrix} = \begin{bmatrix} 0.4 & -0.1 \\ -0.05 & 0.075 \end{bmatrix}$$

- Transpose matrix: $A = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \quad A^T = \begin{bmatrix} 1 & 4 & 7 \\ 2 & 5 & 8 \\ 3 & 6 & 9 \end{bmatrix}$

Matrices – Linear algebra (2/2)

Matrix multiplication:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

1x7 + 2x9 + 3x11 = 58

1x8 + 2x10 + 3x12 = 64

…

Gradient descent algorithm (multivariate)

- Repeat until convergence:

$$w_j := w_j - \alpha \frac{\partial}{\partial w_j} J(w)$$

- Analytically this could be solved with the equation:

$$w = (X^T X)^{-1} X^T y$$

Analytical equation to find weights

x_0	Size (sq. feet)	Rooms	Floors	Construction years	Price (K)
1	2104	5	1	45	460
1	1416	3	2	40	232
1	1534	3	2	30	315
1	852	2	1	36	178

$$x = \begin{bmatrix} 1 & 2104 & 5 & 1 & 45 \\ 1 & 1416 & 3 & 2 & 40 \\ 1 & 1534 & 3 & 2 & 30 \\ 1 & 852 & 2 & 1 & 36 \end{bmatrix} \quad y = \begin{bmatrix} 460 \\ 232 \\ 315 \\ 178 \end{bmatrix}$$

$$w = (X^T X)^{-1} X^T y$$

Gradient descent vs equation

m **training examples –** ***n*** **features**

GRADIENT DESCENT

- Needs to select α
- Requires several iterations
- Works well, even with a large *n* (>10 000)

EQUATION

- Does not need to define α
- Does not require iterations
- Needs to calculate $(X^T X)^{-1}$, a problem of $O(n^3)$ complexity
- Slow with a large *n* (>10 000)

Interpreting the model

Regression algorithms: Linear regression

Interpreting the model

- **Coefficients:**

  If inputs are normalized, the coefficients show the influence of inputs (the largest the magnitude, the larger the importance)

- ***p*-value:**

  The smaller the *p*-value, the more important it is

Variable	Coefficient	*p*
Constant	13.4704	0.0473
x_1	0.4547	0.0000
x_2	0.4478	0.0000
x_3	-0.5847	0.0002
x_4	-0.0545	0.0349
x_5	-0.0009	0.2237

Application exercise

Regression algorithms: Linear regression

Business problem

For an insurance company to make money, it needs to collect more in yearly premiums than it spends on medical care to its beneficiaries. As a result, insurers invest a great deal of time and money to develop models that accurately forecast medical expenses.

Medical expenses are difficult to estimate because the costliest conditions are rare and seemingly random. Still, some conditions are more prevalent for certain segments of the population. For instance, lung cancer is more likely among smokers than non-smokers, and heart disease may be more likely among the obese.

[use case from Lantz, B (2013)]

Business objective

Estimate the medical care expenses per individual. These estimates could be used to create actuarial tables which set the price of yearly premiums higher or lower depending on the expected treatment costs

Understanding key drivers of the estimates

Predict medical expenses

1. Copy from the datasets folder copy the dataset “medical_expenses.csv”
2. Copy and open the Jupyter notebook “PredictMedicalExpenses_Modeling.ipynb”
3. Follow the presentation of the notebook, answer questions, and explore the challenges

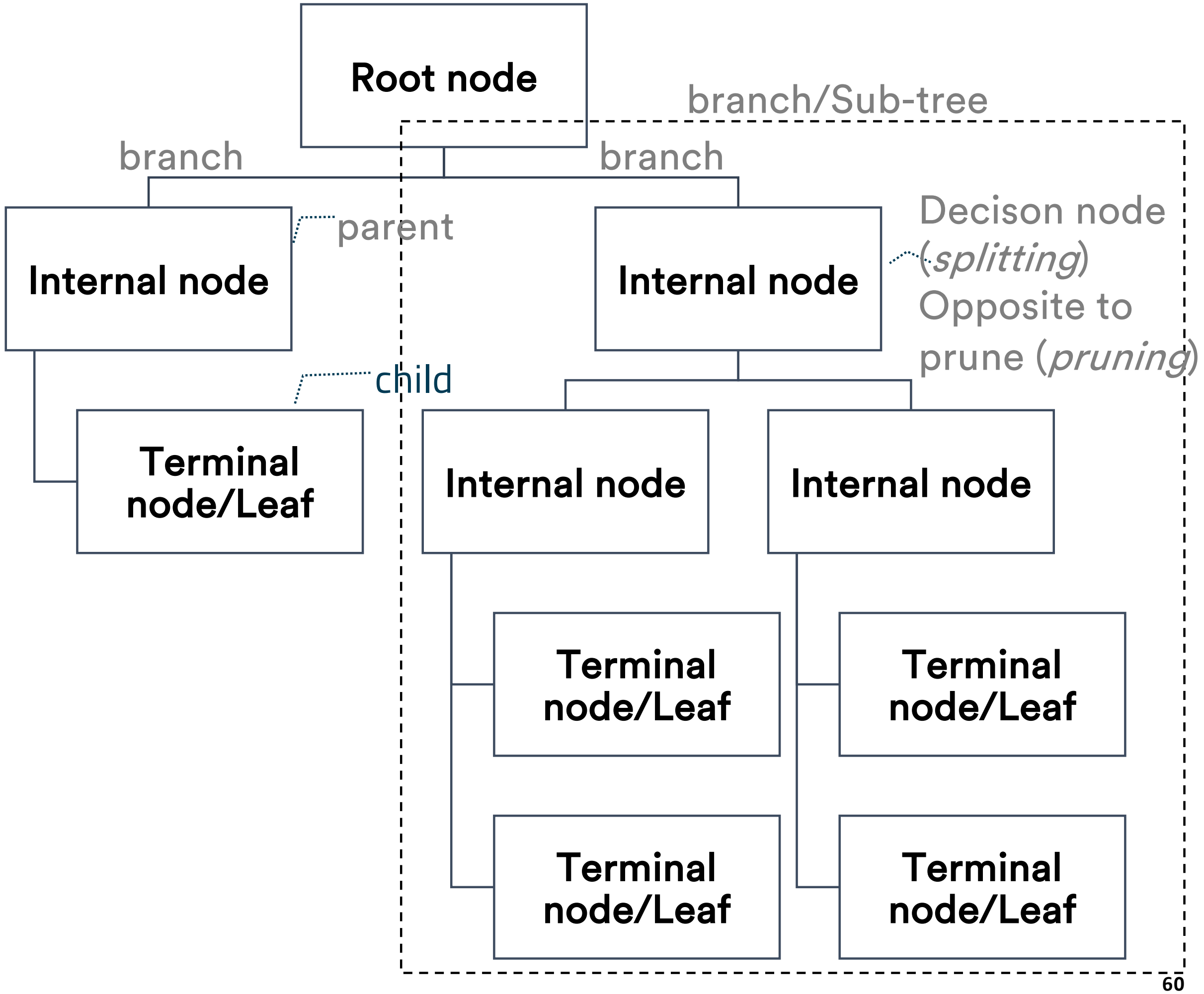
5.6

Regression algorithms: Decision Trees

Modeling: Regression

Decision trees

- One of the simplest and most commonly used algorithms in Machine Learning
- Inputs and outputs can be continuous as well as discrete



Pros and Cons

PROS

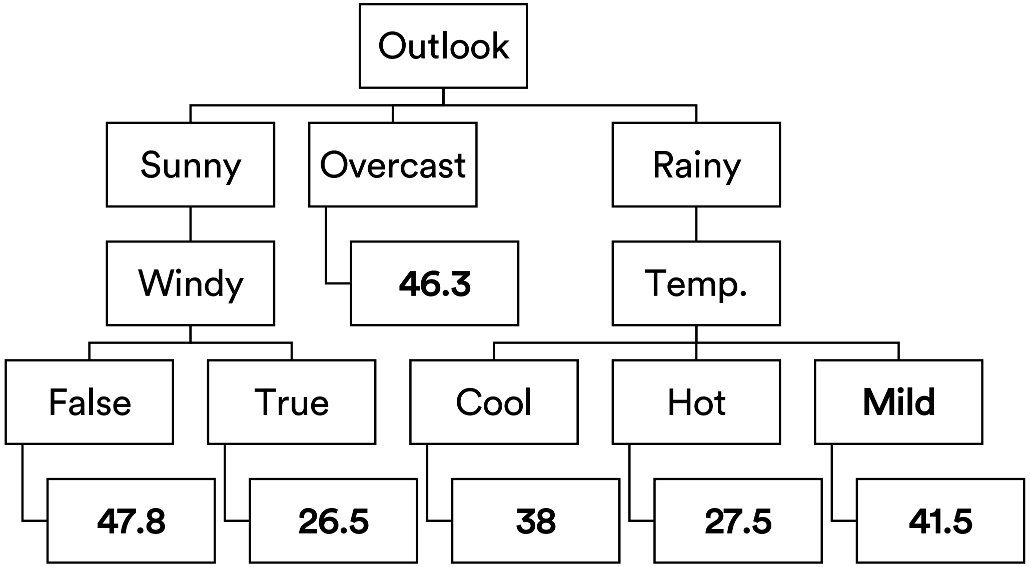
- Easy to understand
- Useful for data exploration
- Not influenced by outliers or missing values
- Nonparametric (do not assume any kind of distribution between inputs and outputs)
- Incorporate a feature selection mechanism

CONS

- Tend to overfit and not to generalize well
- By categorizing continuous variables may lose some information

Example: Estimate golf play hours

Day	Outlook	Temp.	Humidity	Windy	Hours played
1	Rainy	Hot	High	False	26
2	Rainy	Hot	High	True	30
3	Overcast	Hot	High	False	48
4	Sunny	Mild	High	False	46
5	Sunny	Cool	Normal	False	62
6	Sunny	Cool	Normal	True	23
7	Overcast	Cool	Normal	True	43
8	Rainy	Mild	High	False	36
9	Rainy	Cool	Normal	False	38
10	Sunny	Mild	Normal	False	48
11	Rainy	Mild	Normal	True	48
12	Overcast	Mild	High	True	62
13	Overcast	Hot	Normal	False	44
14	Sunny	Mild	High	True	30



How to choose which attribute to divide by

Tree algorithm	Splitting criterion	Input variables	Target variable	Binary or Multi-way splits
CART	Gini index, Two-ing	Categorical or continuous	Categorical or continuous	Binary
C5.0	Gain Ratio, based on Entropy	Categorical or continuous	Categorical	Binary (continuous variables) Multi-way (categorical variables)
CHAID	Chi-square	Categorical	Categorical	Binary or Multi-way
ID3	Information Gain or Standard Deviation Reduction	Categorical or continuous	Categorical or continuous	Binary or Multi-way

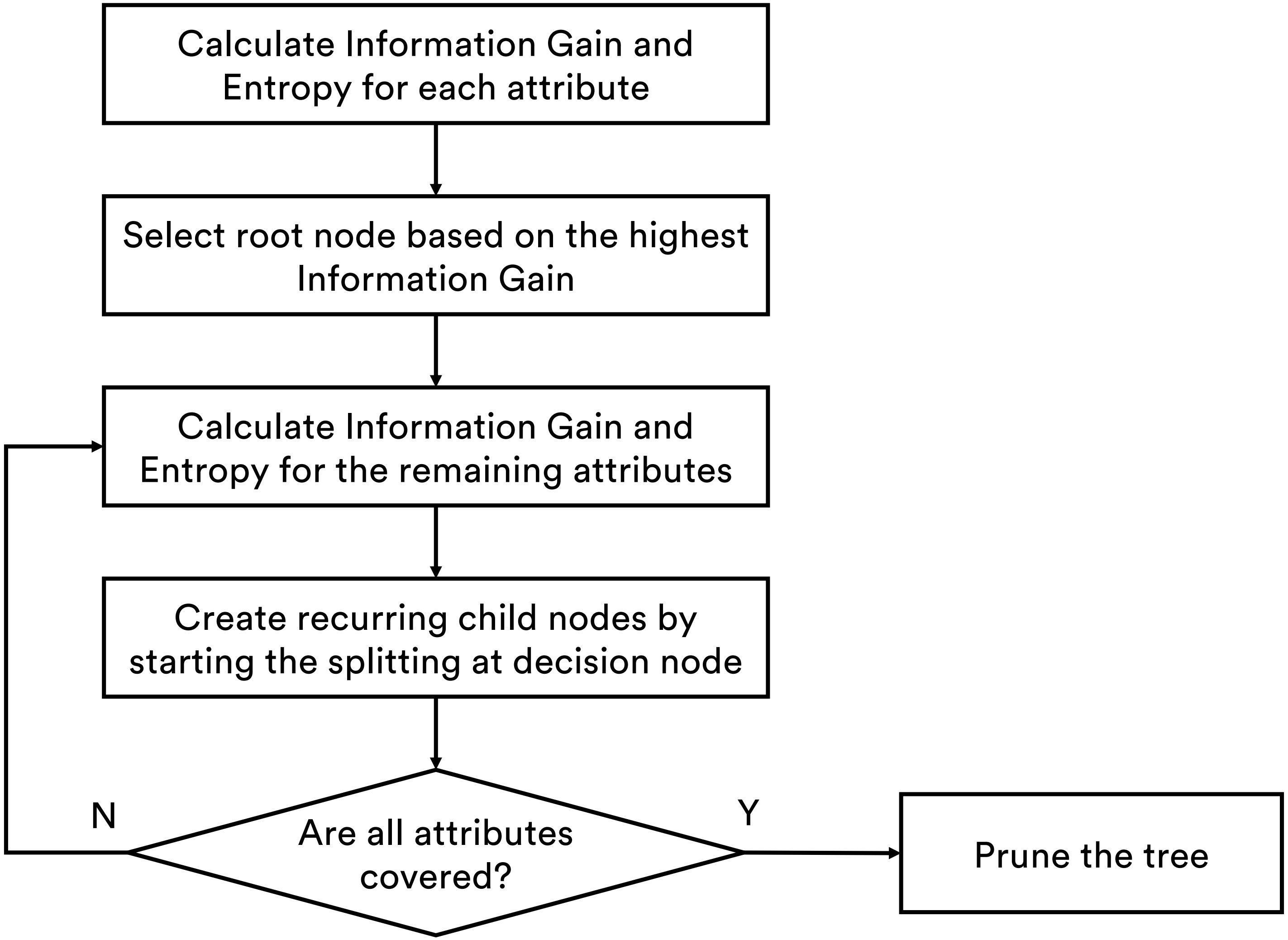
Algorithm example: C5.0 (1/2)

- Entropy:
  - From physics – a metric to measure the degree of disorder or randomness of a system
  - Formula for a dataset with C classes: $E = -\sum_i^C p_i log_2 p_i$
  - In a DT is used to measure the uncertainty of predicting the target
  - Consider a dataset with 1 blue, 2 green, and 3 red balls (6 instances):

$$E = -\left(\frac{1}{6} log_2\left(\frac{1}{6}\right) + \frac{2}{6} log_2\left(\frac{2}{6}\right) + \frac{3}{6} log_2\left(\frac{3}{6}\right)\right) = 1.46$$

- Information Gain:
  - Measure used to determine which attribute is the most useful for discriminating between target classes
  - Formula: $IG = Entropy(parent) - [average\ entropy(children)]$

Algorithm example: C5.0 (2/2)


Calculate Information Gain and Entropy for each attribute
Select root node based on the highest Information Gain
Calculate Information Gain and Entropy for the remaining attributes
Create recurring child nodes by starting the splitting at decision node
Are all attributes covered?
N
Y
Prune the tree

Tuning Decision Trees

- **Maximum depth**: The number of levels deep a tree can go. The deeper (complex) goes, the likely to generate overfitting
- **Minimum number of instances on terminal nodes:** limitation to number of instances on terminal nodes. The smaller the number, the likely to overfit
- **Minimum number of records in parent node:** threshold to the splitting node. Once there are fewer than the threshold, no further split is allowed at the branch

Practical considerations

- Decision trees incorporate only one variable for each split. If a modeler knows multivariate features that may produce better results, should consider create new features from different variables
- Decision trees tend to perform not so good when categorical variables suffer from high cardinality
- Single trees are often not as accurate as other algorithms. If model interpretation is not the top priority, consider building ensembles of trees

Application exercise

Regression algorithms: Decision Trees

Business problem

For an insurance company to make money, it needs to collect more in yearly premiums than it spends on medical care to its beneficiaries. As a result, insurers invest a great deal of time and money to develop models that accurately forecast medical expenses.

Medical expenses are difficult to estimate because the costliest conditions are rare and seemingly random. Still, some conditions are more prevalent for certain segments of the population. For instance, lung cancer is more likely among smokers than non-smokers, and heart disease may be more likely among the obese.

[use case from Lantz, B (2013)]

Business objective

Estimate the medical care expenses per individual. These estimates could be used to create actuarial tables which set the price of yearly premiums higher or lower depending on the expected treatment costs

Understanding key drivers of the estimates

Predict medical expenses

1. Copy from the datasets folder copy the dataset “medical_expenses.csv”
2. Copy and open the Jupyter notebook “PredictMedicalExpenses_Modeling_DT.ipynb”
3. Follow the presentation of the notebook, answer questions, and explore the challenges

5.7

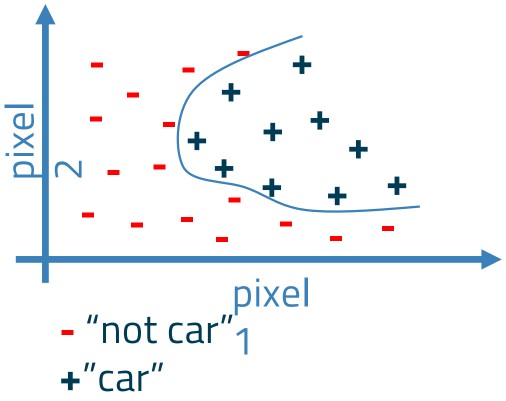
Regression algorithms: Neural Networks

Modeling: Regression

Neural networks

- Try to replicate brain function
- Very popular in the 80's and 90's, but loose some popularity over time
- New techniques allowed its resurgence
- Are the foundation of deep learning
- Create nonlinear hypotheses
- Very difficult to interpret (“black boxes”)

Complex and nonlinear problems



50 x 50 pixels images → 2 500 pixels

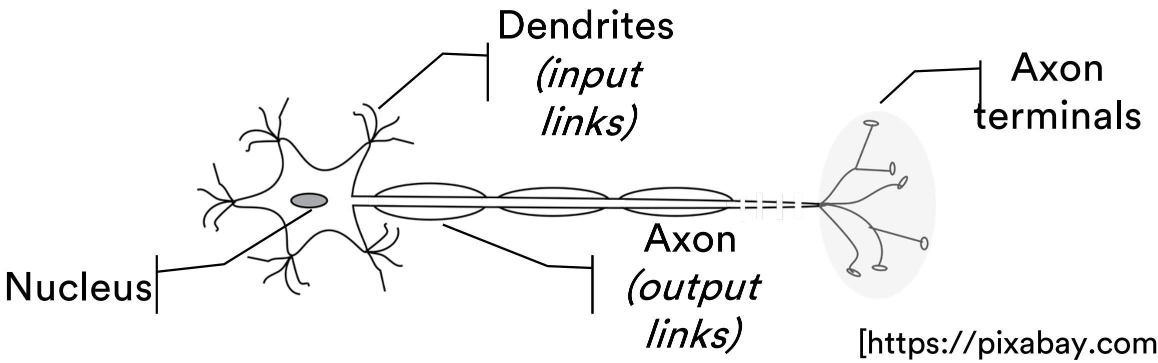
n= 2 500 (7 500 if RGB)

$$x = \begin{bmatrix} pixel\ 1\ intensity \\ pixel\ 2\ intensity \\ \dots \\ pixel\ 2\ 500\ intensity \end{bmatrix}$$

If features are quadratic $(x_i \times x_j) \approx 3$ million *features*

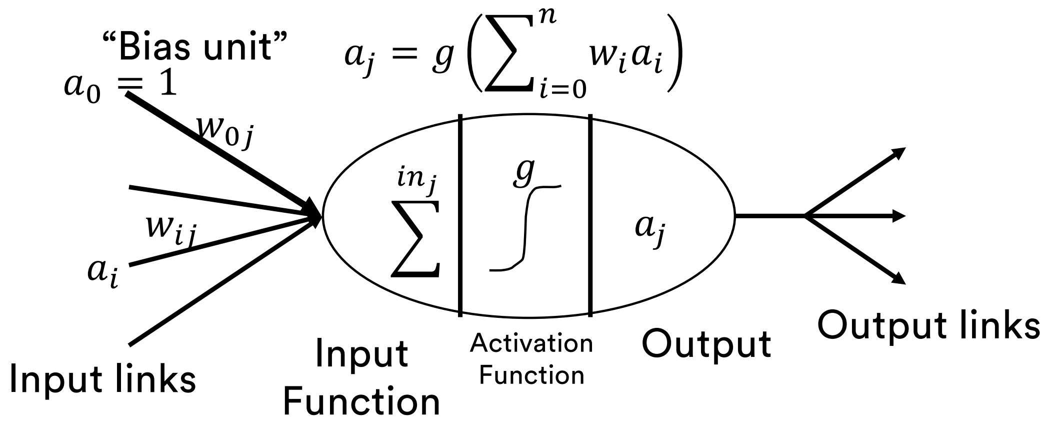
Neurons

BIOLOGICAL



[https://pixabay.com]

ARTIFICIAL

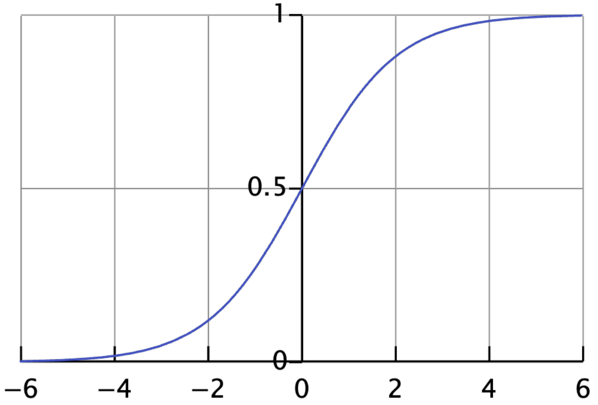


Activation function

Logistic/sigmoide function

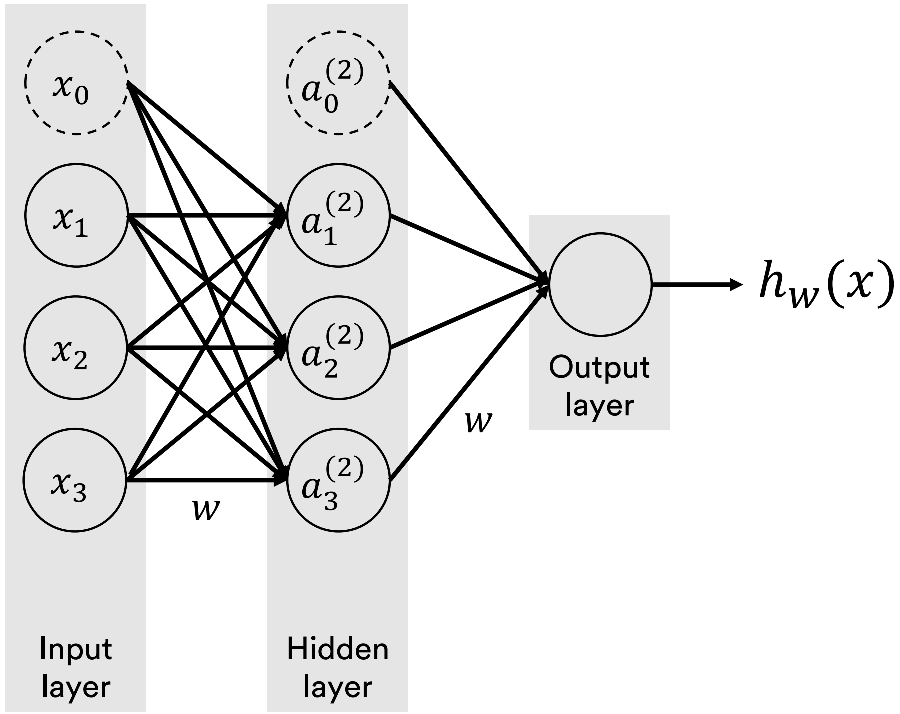
$$g(z) = \frac{1}{1 + e^{-z}}$$

$$h_w(x) = \frac{1}{1 + e^{-w^T x}}$$



[https://wikipedia.com]

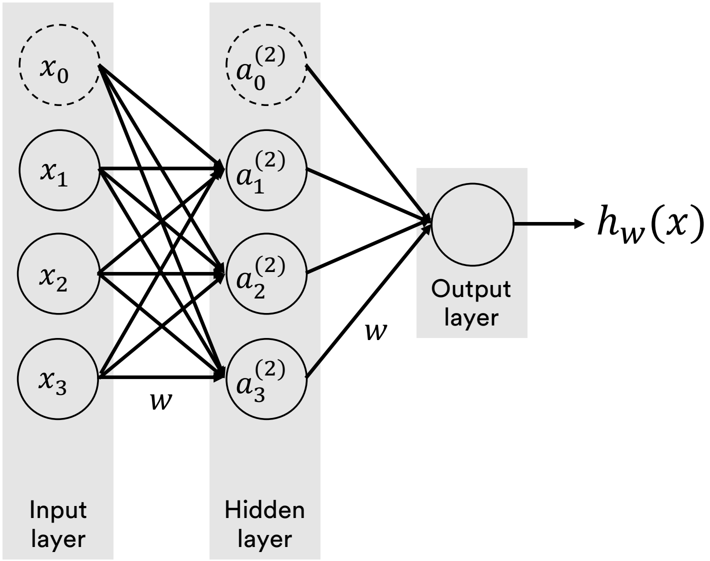
Neural network



$a_i^{(j)}$= activation of unit *i* in layer *j*

$w^{(j)}$= weight matrix of the function that controls the mapping of **layer** *j* to layer *j*+1

Neural network



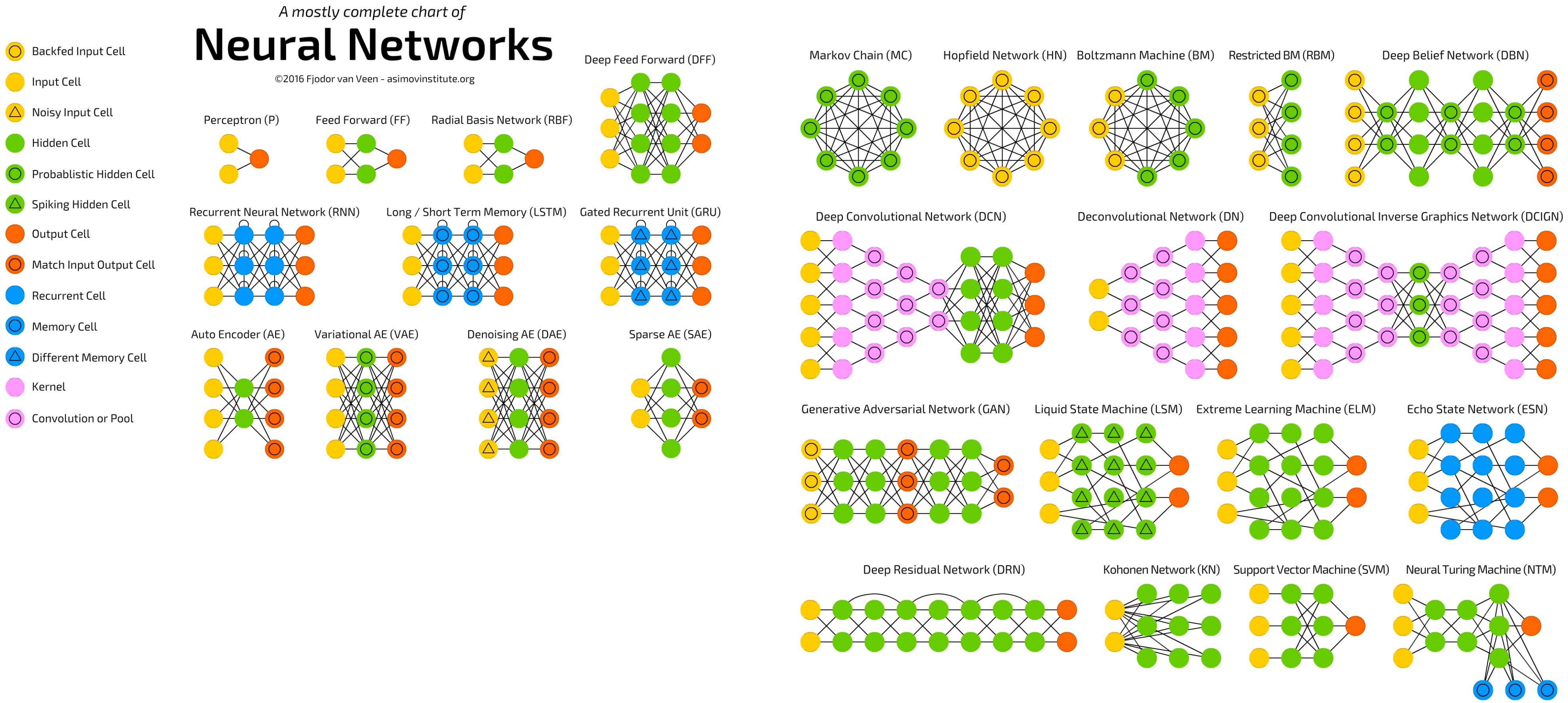
$a_i^{(j)}$= activation of unit *i* in layer *j*

$w^{(j)}$ = weight matrix of the function that controls the mapping of layer *j* to layer *j*+1

$$a_1^{(2)} = \mathrm{g}(w_{10}^{(1)}x_0 + w_{11}^{(1)}x_1 + w_{12}^{(1)}x_2 + w_{13}^{(1)}x_3)$$
$$a_2^{(2)} = \mathrm{g}(w_{20}^{(1)}x_0 + w_{21}^{(1)}x_1 + w_{22}^{(1)}x_2 + w_{23}^{(1)}x_3)$$
$$a_3^{(2)} = \mathrm{g}(w_{30}^{(1)}x_0 + w_{31}^{(1)}x_1 + w_{32}^{(1)}x_2 + w_{33}^{(1)}x_3)$$

$$h_w(x) = a_1^3 = \mathrm{g}(w_{10}^{(2)}a_0^{(2)} + w_{11}^{(2)}a_1^{(2)} + w_{12}^{(2)}a_2^{(2)} + w_{13}^{(2)}\, a_3^{(2)})$$

Neural network topologies


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

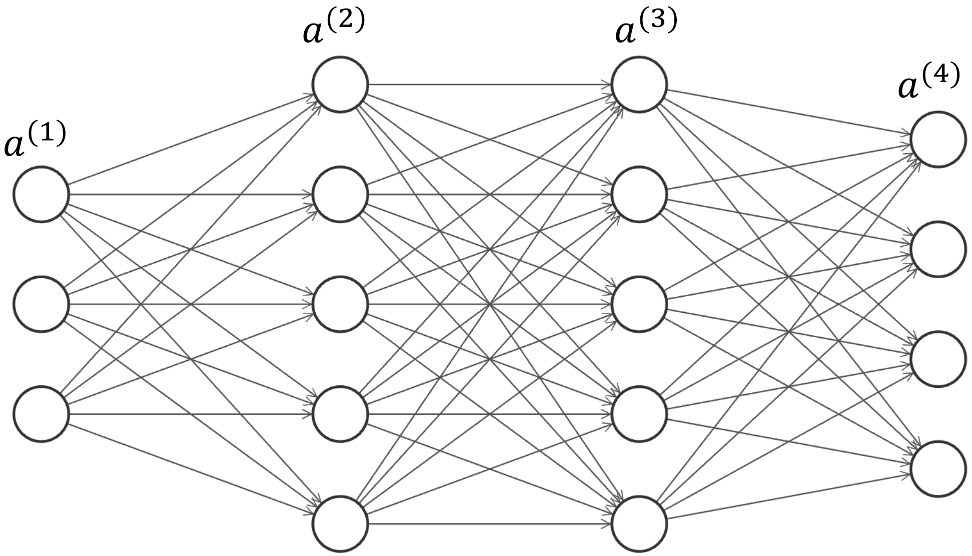
[https://towardsdatascience.com]

Common topologies

- **Feed-forward (forward propagation)**: Has connections only in one direction, i.e., forms a directed acyclic graph. Each node receives input from upstream nodes and delivers output to downstream nodes. There are no loops
- **Recurrent network (backward propagation)**: feeds its inputs its own outputs. This means that network activation levels form a dynamic system that can reach a steady state or exhibit oscillations or even chaotic behaviors

Forward propagation

$$a^{(1)} = x$$
$$z^{(2)} = w^{(1)}a^{(1)}$$
$$a^{(2)} = g\left(z^{(2)}\right) \quad (add\ a_0^{(2)})$$
$$z^{(3)} = w^{(2)}a^{(2)}$$
$$a^{(3)} = g\left(z^{(3)}\right) \quad (add\ a_0^{(3)})$$
$$z^{(4)} = w^{(3)}a^{(3)}$$
$$a^{(4)} = h_w(x) = g\left(z^{(4)}\right)$$



Back propagation Gradient computation

Intuition: $\delta_j^{(l)}$="error" of node *j* in *l*ayer *l*

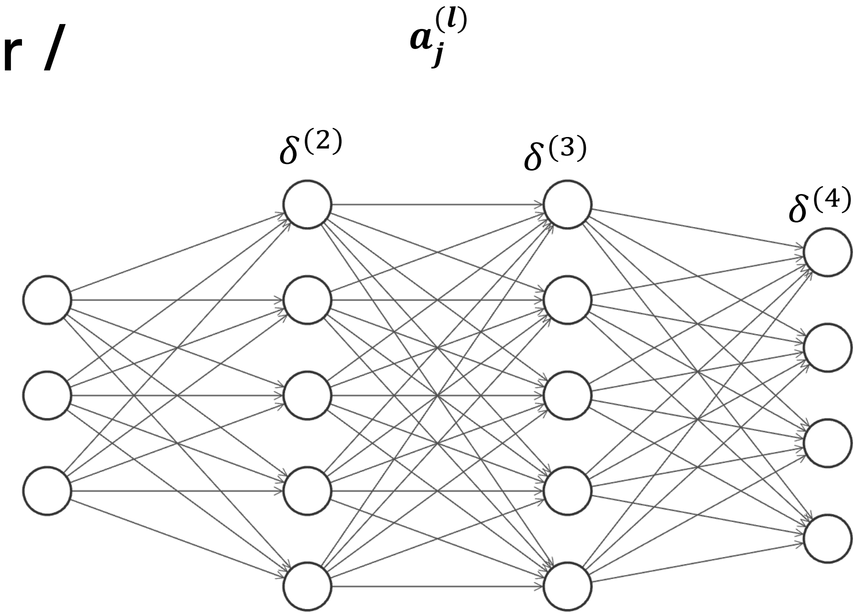
For each output node (layer L=4)

$$\delta_j^4 = a_j^{(4)} - y_j$$

$$\delta^{(4)} = a^{(4)} - y$$

$$\delta^{(3)} = (w^{(3)})^T \delta^{(4)} .* \mathrm{g}'(z^{(3)})$$

$$\delta^{(2)} = (w^{(2)})^T \delta^{(3)} .* \mathrm{g}'(z^{(2)})$$



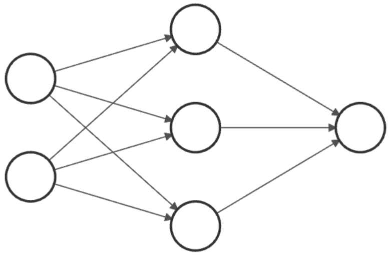
Practical considerations

- Like other numerical algorithms requires numeric input data
- Categorical variables must be transformed into numeric
- Do not work with missing data
- Usually perform better than k-NN, logistic regression, or decision trees

Neural networks and overfitting

Small networks

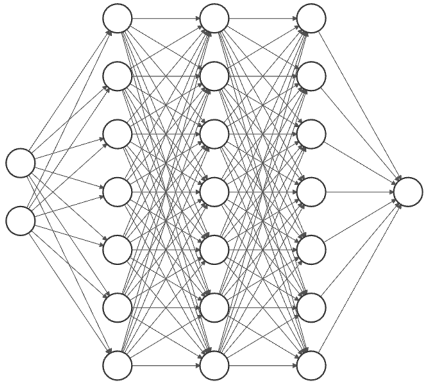
(few parameters; more prone to *underfitting*)



Computationally cheaper

Big networks

(many parameters; more prone to *overfitting*)



Computationally more expensive

Application exercise

Regression algorithms: Neural networks

Predict customer CLV

1. From the datasets folder copy the dataset “OnlineRetail.xlsx”
2. Copy and open the Jupyter notebook “PredictCustomerCLV.ipynb”
3. Follow the presentation of the notebook and explore the challenges

5.8

Regression algorithms: Parameters Hyper Tunning

Modeling: Regression

Hyper tunning: What it is

- Most algorithms (for regression and classification) have optimization parameters to optimize the learning process
- These parameters require different constrains, weights or learning rates to generalize different data patterns (e.g., the learning rate in a liner regression algorithm or the minimum number of observations in a decision tree node)
- The objective of hyper tuning is to automate the task of manually finding the values of the parameters that generate a model that minimizes the loss function

Approaches

- **Grid search** (AKA parameter sweep): exhaustive search through a subset of the hyperparameter space specified by the modeler
- **Random search:** evaluates parameters randomly in predefined domain and hyperparameter space
- **Bayesian optimization:** builds a probabilistic model of the function mapping it to the objective evaluated in a validation set
- **Evolutionary algorithms:** meta-heuristic approaches that use the value of the fitness function to find the best hyperparameters

Demo: Grid search

1. From the datasets folder copy the dataset “OnlineRetail.xlsx”
2. Copy and open the Jupyter notebook “PredictCustomerCLV_GS.ipynb”
3. Follow the presentation of the notebook and explore the challenges

Demo: Randomized search

1. From the datasets folder copy the dataset “OnlineRetail.xlsx”
2. Copy and open the Jupyter notebook “PredictCustomerCLV_RS.ipynb”
3. Follow the presentation of the notebook and explore the challenges

Demo: Optuna framework

1. From the datasets folder copy the dataset “OnlineRetail.xlsx”
2. Copy and open the Jupyter notebook “PredictCustomerCLV_Optuna.ipynb”
3. Follow the presentation of the notebook and explore the challenges

For more info check https://optuna.org

Questions?

Machine Learning for Marketing

Acreditações e Certificações

Instituto Superior de Estatística e Gestão da Informação
Universidade Nova de Lisboa