Introduction to Machine Learning

Supervised Learning

Luís Nunes Ricardo Ribeiro Sancho Oliveira

Iscte – Instituto Universitário de Lisboa

2022/2023

INSTITUTO UNIVERSITÁRIO DE LISBOA

Learning Types

Reinforcement Learning	Learn the best (sequence of) action(s) given state(s)
Search	Given a set of possible solutions, find the best (a reasonable) one
Unsupervised Learning	Group similar things
Supervised Learning	Predict outcome. Learn a rule/model given examples

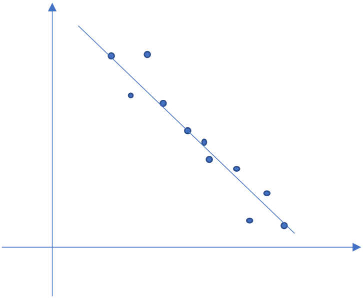
Outline

- Given a set of examples
  - Each example has
    - A set of values, one for each attribute
    - A desired output: a continuous value

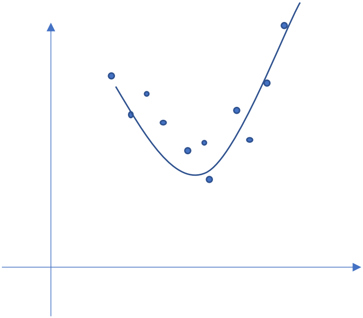
Example

Attributes	Square meters	140
	Number of rooms	4
	...	...
Output	Current house price	€150,000

Regression



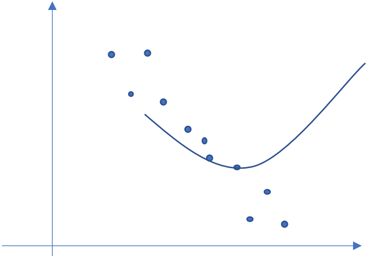
Linear



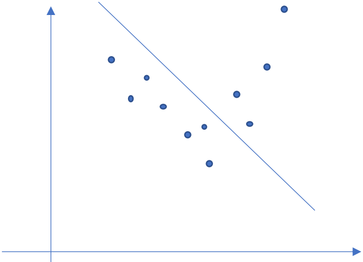
Non-linear

Regression

when the model does not fit data



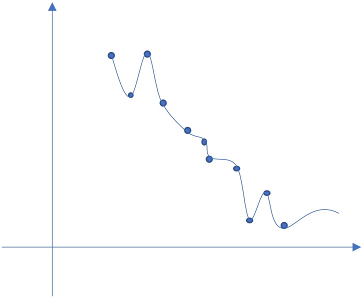
Model too complex

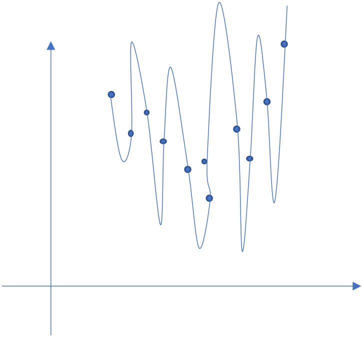


Model too simple

Regression

when the model does not generalize well (overfit)





Classification

problem

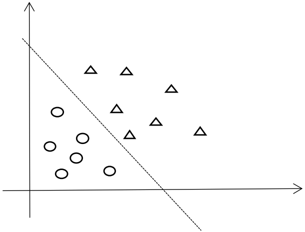
- Given a set of examples
  - Each example has
    - A set of values, one for each attribute
    - A desired output: a category

Example

Attributes	Age	40
	Current balance	5,000
	Had previous loans	No
	Loan value	10,000
Output	Will pay current loan?	No

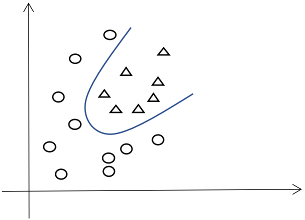
Classification

separable classes



Linear separation

Non-linear separation



Bias / Variance Dilemma

Bias error Model does/can not correctly represent the concept (underfit)

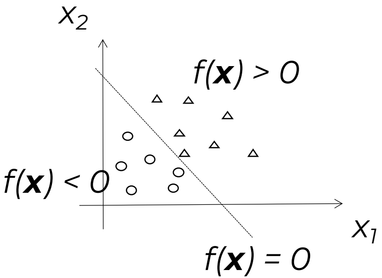
Variance error Model specializes in training set (overfit)

Mitigating variance error Regularization (favor smoother functions – output varies slowly with input)

Classification

linearly separable classes

$$f(\mathbf{x}) = w_0 + w_1 \cdot x_1 + w_2 \cdot x_2$$



Multilinear Regression

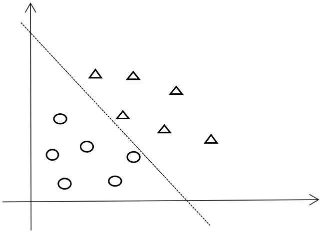
$$y = w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n + \varepsilon$$

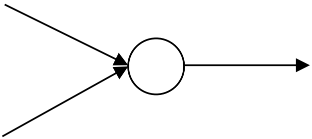
$$Y = WX + \varepsilon$$

- Assumptions
  - Relation between x_i and y is linear
  - All variables (x) have Normal distributions
  - Variables are independent and residual / error $(y(x_i) - \hat{y}(x_i))$ is constant
- Least Mean Squares

$$\hat{W} = (X^T X)^{-1} (X^T Y)$$

Classification with Perceptron

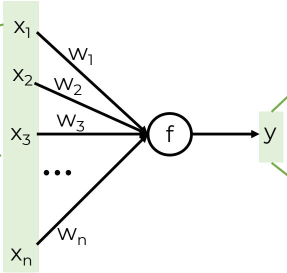




Artificial Neuron

An artificial neuron, in a feedforward neural network, receives a set of inputs and generates a single output.

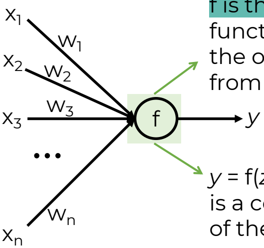
The input of an artificial neuron comes from all neurons of the previous layer or it is an external input



The output of an artificial neuron is a simple combination of the received inputs

The output of an artificial neuron is sent to all neurons of the next layer or is (part of) the network output.

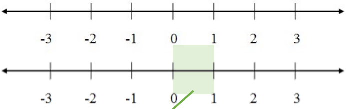
Artificial Neuron



f is the activation function. This function is used to generate the output of the neuron (y) from the neuron's inputs.

y = f(z), in which z is a combination of the inputs

$w_1x_1 + w_2x_2 + w_3x_3 + \ldots + w_nx_n$

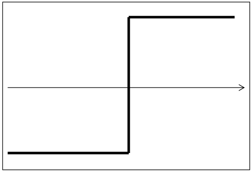


Activations should be in this range

Perceptron and Delta Rule

simple function and update rule for classification

$$f(x) = \begin{cases} 1, & w_0 + w_1x_1 + w_2x_2 + \cdots > 0 \\ -1, & w_0 + w_1x_1 + w_2x_2 + \cdots \leq 0 \end{cases}$$

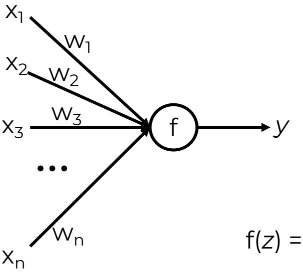


$$\Delta w_i = \alpha(f(x) - d)x_i$$

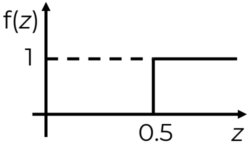
where d is the desired value

Boolean Operations Network

example



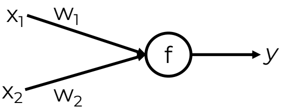
$$f(z) = \begin{cases} 0, \text{ if } z < 0.5 \\ 1, \text{ if } z \geq 0.5 \end{cases}$$



In which, $z = \sum_{i=1}^{n}(w_i x_i)$

Boolean Operations Network

AND



$$f(z) = \begin{cases} 0, \text{ if } z < 0.5 \\ 1, \text{ if } z \geq 0.5 \end{cases}$$

X_1	X_2	AND
0	0	0
0	1	0
1	0	0
1	1	1

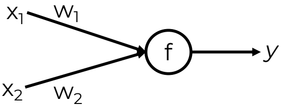
X_1	X_2
0	0
0	1
1	0
1	1

$W_1 = W_2 = 0.1$
$Y = f(0.1 \text{ x } X_1 + 0.1 \text{ x } X_2)$
$f(0.1{\times}0+0.1{\times}0) = f(0) = 0$
$f(0.1{\times}0+0.1{\times}1) = f(0.1) = 0$
$f(0.1{\times}1+0.1{\times}0) = f(0.1) = 0$
$f(0.1{\times}1+0.1{\times}1) = f(0.2) = 0$

$W_1 = W_2 = 0.3$
$Y = f(0.3 \text{ x } X_1 + 0.3 \text{ x } X_2)$
$f(0.3{\times}0+0.3{\times}0) = f(0) = 0$
$f(0.3{\times}0+0.3{\times}1) = f(0.3) = 0$
$f(0.3{\times}1+0.3{\times}0) = f(0.3) = 0$
$f(0.3{\times}1+0.3{\times}1) = f(0.6) = 1$

Boolean Operations Network

OR



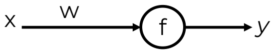
$$f(z) = \begin{cases} 0, \text{ if } z < 0.5 \\ 1, \text{ if } z \geq 0.5 \end{cases}$$

X_1	X_2	OR
0	0	0
0	1	1
1	0	1
1	1	1

X_1	X_2	$W_1 = W_2 = 0.6$
		$Y = f(0.6 \times X_1 + 0.6 \times X_2)$
0	0	$f(0.6\times0+0.6\times0) = f(0) = 0$
0	1	$f(0.6\times0+0.6\times1) = f(0.6) = 1$
1	1	$f(0.6\times1+0.6\times0) = f(0.6) = 1$
1	1	$f(0.6\times1+0.6\times1) = f(1.2) = 1$

Boolean Operations Network

NOT



$$f(z) = \begin{cases} 0, \text{ if } z < 0.5 \\ 1, \text{ if } z \geq 0.5 \end{cases}$$

X	NOT
0	1
1	0

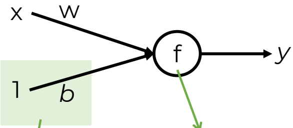
If the network correctly implemented the Boolean NOT, whenever x = 1, y = 0; and whenever x = 0, y = 1

But f(w×0) = f(0) = 0. Independently of the value of the weight w, f(w×0) = 0

Thus, this cannot be used to implement the Boolean NOT; something else is required: bias value

Boolean Operations Network

NOT



$$f(z) = \begin{cases} 0, \text{ if } z < 0.5 \\ 1, \text{ if } z \geq 0.5 \end{cases}$$

X	NOT
0	1
1	0

b is called the bias of the neuron

General expression of the output:

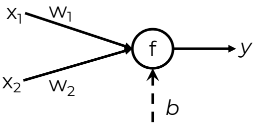
y = f(z), with z = $b + \sum_{i=1}^{n}(w_i x_i)$

X	1
0	1
1	1

W=-0.5; b = 0.5
Y = f(-0.5 x X + 0.5)
f(-0.5×0+0.5) = f(0.5) = 1
f(-0.5×1+0.5) = f(0) = 0

Boolean Operations Network

NAND



$$f(z) = \begin{cases} 0, \text{ if } z < 0.5 \\ 1, \text{ if } z \geq 0.5 \end{cases}$$

X_1	X_2	NAND
0	0	1
0	1	1
1	0	1
1	1	0

With different network parameters (weights and biases), it is possible to implement different relations between input and output

X_1	X_2	B=0.75; $W_1 = W_2 = -0.25$
		$Y = f(0.75 - 0.25\times(X_1+X_2))$
0	0	$f(0.75 - 0.25\times0) = f(0.75) = 1$
0	1	$f(0.75 - 0.25) = f(0.5) = 1$
1	0	$f(0.75 - 0.25) = f(0.5) = 1$
1	1	$f(0.75 - 0.25\times(1+1)) = f(0.25) = 0$

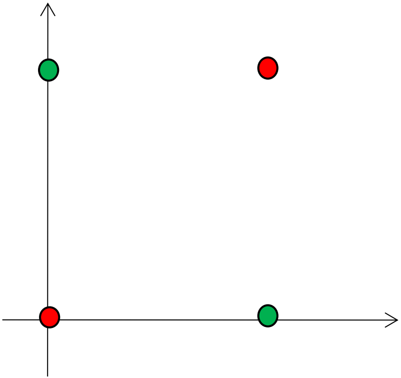
Learning

changing the network parameters

- Changing the parameters of the network (weights and biases), it is possible to implement different relations between input and output
- Thus, if we want a network to learn the input/output relation implicit in a given training set, the learning algorithm just needs to change the network parameters (connection weights and neuron biases)
- But there are certain input/output relations that required more complex networks than a single neuron besides the input layer

Non-linearly Separable Classes

Peceptron draws linear separation between classes, not suitable for all problems (XOR) [Minsky & Papert 69]



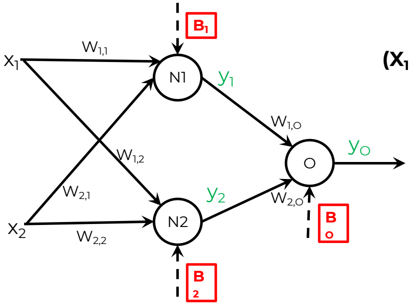
Boolean Operations Network

XOR

$XOR(x_1, x_2) = AND(OR(x_1, x_2), NAND(x_1, X_2))$

$(\mathbf{X_1 \vee X_2) \wedge \neg(X_1 \wedge X_2})$

X_1	X_2	XOR
0	0	0
0	1	1
1	0	1
1	1	0



N1 implements the Boolean OR of the two inputs x_1 and x_2

N2 implements the Boolean NAND of the two inputs x_1 and x_2

O implements the Boolean AND of the outputs of N1 and N2, y_1 and y2

Boolean Operations Network

XOR

$$XOR(x_1, x_2) = AND(OR(x_1, x_2), NAND(x_1, X_2))$$

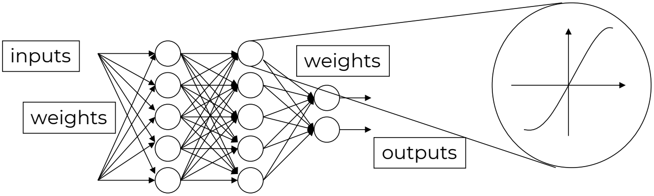
$$(X_1 \vee X_2) \wedge \neg(X_1 \wedge X_2)$$

		OR	NAND	AND	XOR
X_1	X_2	$Y_1 = f(0.6\times X_1 + 0.6\times X_1)$	$Y_2 = f(0.75 - 0.25\times(X_1 + X_2))$	$Y_O = f(0.3\times Y_1 + 0.3\times Y_2)$	
0	0	$f(0) = 0$	$f(0.75) = 1$	$f(0.3) = 0$	0
0	1	$f(0.6) = 1$	$f(0.5) = 1$	$f(0.6) = 1$	1
1	0	$f(0.6) = 1$	$f(0.5) = 1$	$f(0.6) = 1$	1
1	1	$f(1.2) = 1$	$f(0.25) = 0$	$f(0.3) = 0$	0

Artificial Neural Networks

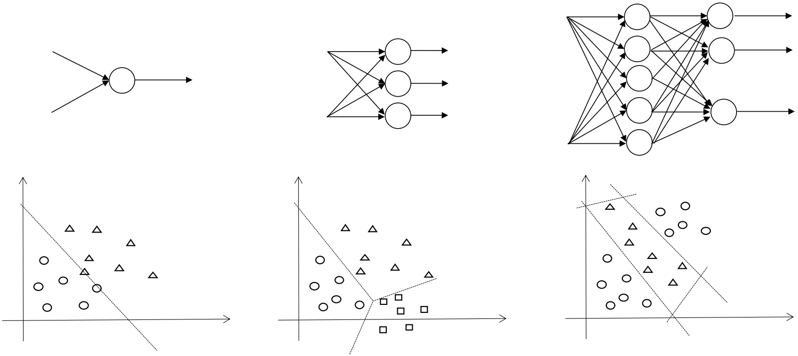
[Rumelhart, Hinton, Williams 86]

- Multilayer Perceptron
- Performance depends only on number of hidden layer units



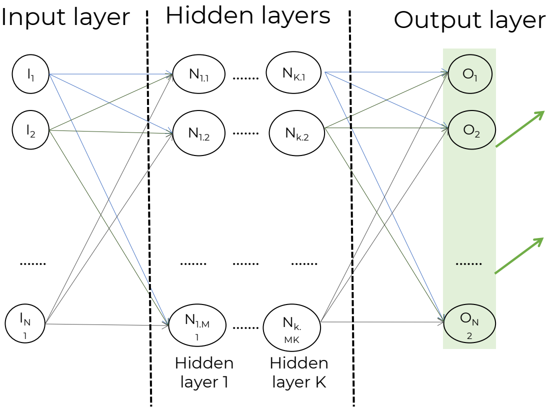
Classification with MLP

potencial for space division



Feedforward Neural Network with Backpropagation

Classification

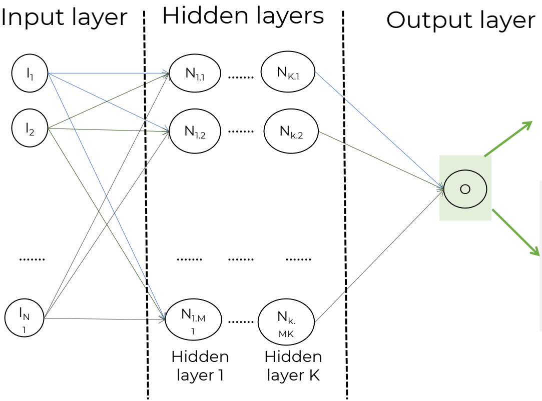


Each node of the output represents one class

The output class is the one corresponding to the node with the highest output value

Feedforward Neural Network with Backpropagation

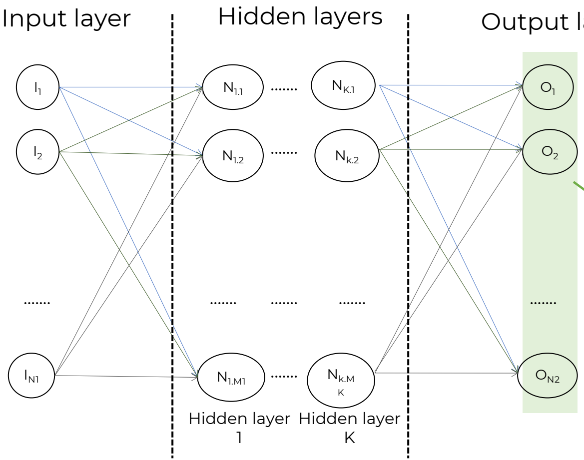
Regression



There is only a single output node

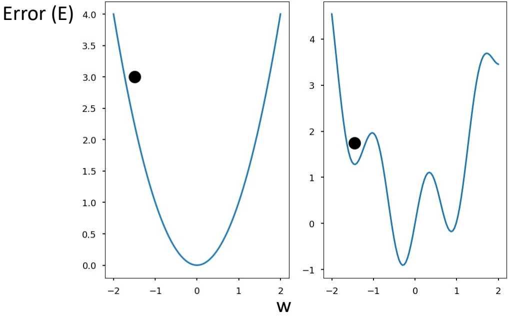
The value predicted for the numeric output variable is the output of the output node

Feedforward Neural Network with Backpropagation



It is the expected result?
No? -> Error calculation

Gradient Descent Algorithm



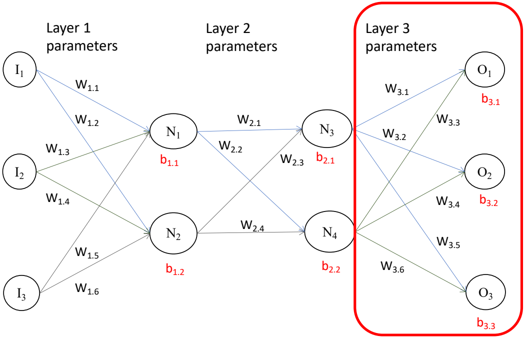
$$\frac{\partial E}{\partial w}$$

Derivative of the error

$$\Delta_P = -(\eta \times \frac{\partial E}{\partial P}),$$

in which P is a parameter, Δ_P is the update to be added to P, E is the error and η is a proportionality constant called the **learning rate**

Backpropagation



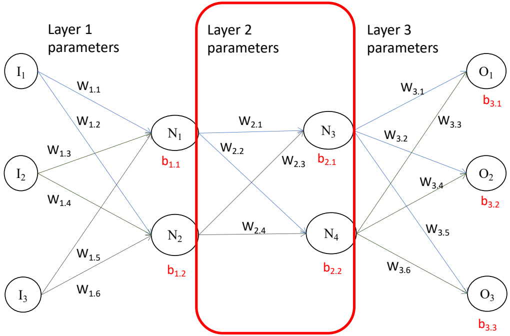
For instance,

W3.1 = W3.1 + $\Delta_{w3.1}$

$$\Delta_{w3.1} = -(\eta \times \frac{\partial E}{\partial w3.1})$$

$\frac{\partial E}{\partial w3.1}$ depends on the network outputs

Backpropagation



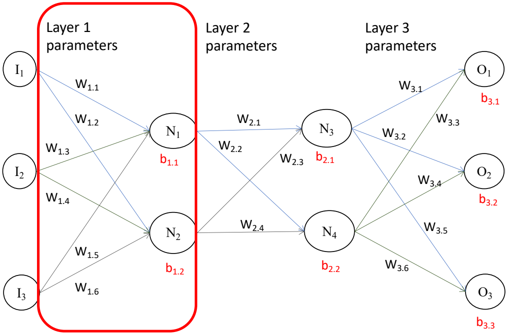
For instance,

$W2.1 = W2.1 + \Delta_{w2.1}$

$$\Delta_{w2.1} = -(\eta \times \frac{\partial E}{\partial w2.1})$$

$\frac{\partial E}{\partial w2.1}$ depends only on the network outputs and on the derivatives of the error with respect to the parameters of layer 3 **(already computed)**

Backpropagation



For instance,

b1.1 = b1.1 + $\Delta_{b1.1}$

$$\Delta_{b1.1} = -(\eta \times \frac{\partial E}{\partial b1.1})$$

$\frac{\partial E}{\partial b1.1}$ depends only on the network outputs and on the derivatives of the error with respect to the parameters of layer 2 **(already computed)**

Backpropagation

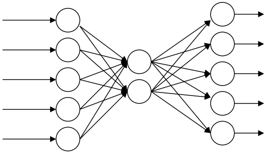
[Rumelhart, Hinton, Williams 86]

- Present each example $(x(i), d(i))$
- Calculate network response $x(i) : f(x(i))$
- Propagate error backwards (iteratively building error derivative at each layer)
- Save partial derivatives
- After all examples processed, update weights

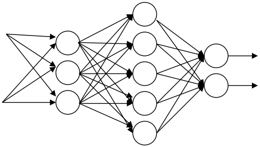
Neural Network

Compression vs Feature Generation

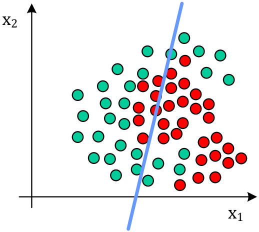
Compression



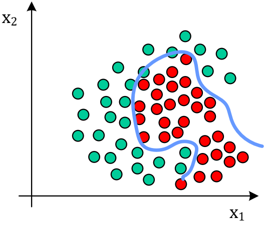
Feature generation



Activation Functions



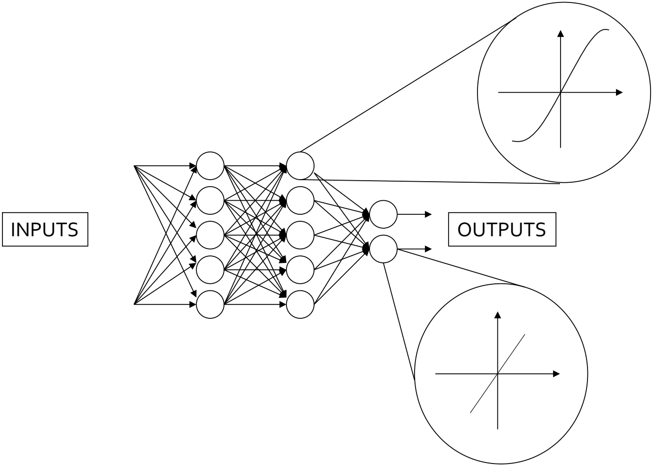
Linear activation function allows a linar separation



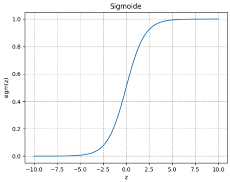
Non-linear activation function allow complex separations

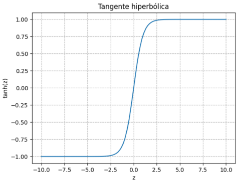
MPL

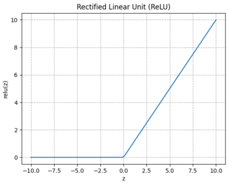
Linear outputs for regression



Activation Functions







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

$$sigm'(z) = sigm(z) \times sigm(1 - z)$$

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

$$tanh'(z) = 1 - tanh(z)^2$$

$$ReLU(z) = \begin{cases} x, se\ x > 0 \\ 0, se\ x \leq 0 \end{cases}$$

$$ReLU'(z) = \begin{cases} 1, se\ x > 0 \\ 0, se\ x \leq 0 \end{cases}$$

ANN/MLP+backpropagation

- Analytic methods of classification / regression deal badly with noise and are sensitive to numerical approximations
- ANN are:
  - Robust to noise and approximations
  - Based in simplified neuron model
  - Incremental training
  - Compress information of many examples in small model

Deep Learning

- Alternating prediction layers with feature decorrelation
- Techniques used were first thought of in the 70s and 80s
- ... now benefit from the massive computing power available

- Object Detection with Tensorflow API: https://www.youtube.com/watch?v=_zZe27JYi8Y
- Quadcopter Navigation in the Forest using Deep Neural Networks: https://www.youtube.com/watch?v=umRdt3zGgpU
- Real-time face recognition with Deep Learning technology: https://www.youtube.com/watch?v=B4m2RVFLbME

References

- Almeida, L.B.D., 1990. and Multilayer Perceptrons F. Emile & B. Russell, eds. Analysis, 12(12), pp.1167-1178.
- Minsky, M. & Papert, S., 1969. Perceptrons: An Introduction to Computational Geometry, MIT Press. Available at: http://www.computer.org/portal/web/csdl/doi/10.1109/T-C.1969.222718.
- Minsky, M.L. & Papert, S.A., 1988. Perceptrons: Expanded edition, The MIT Press. Available at: http://mitpress.mit.edu/book-home.tcl?isbn=0262631113.
- Rumelhart, D E, Hinton, G.E. & Williams, R.J., 1986. Learning internal representation by error propagation. In D E Rumelhart & J. L. McClelland, eds. Parallel Distributed Processing Explorations in the Microstructure of Cognition. Cambridge, MA: MIT Press, pp. 318-362.
- Rumelhart, David E, Widrow, B. & Lehr, M.A., 1994. The basic ideas in neural networks. Communications of the ACM, 37(3), pp.87-92. Available at: http://portal.acm.org/citation.cfm?doid=175247.175256.
- Silva, F.M. & Almeida, L.B., 1990. Acceleration Techniques for the Backpropagation Algorithm. Neural Networks Proc EURASIP Workshop, 412, pp.110-119.

k-Nearest Neighbours

- Find k patterns in the set most similar to the one to classify
- Select a class between those of known patterns (how? Most common? Only consider majority? ties?)
- Problems:
  - Define distance,
  - Define class selection,
  - Non-linear problems

k-Nearest Neighbours

Attribute Values

Sky	**Temperature**	**Humidity**	**Wind**	**Sea**	**Prediction**
Clear	Warm	Normal	String	Warm	Stable
Cloudy	Cold	High	Weak	Cold	Unstable
Rain					

- Can we find the pattern?

#	Sky	Temp.	Humid	Wind	Sea	Pred.	Go surf?
1	Clear	Warm	Normal	Strong	Warm	Stable	Yes
2	Clear	Warm	Normal	Strong	Warm	Unstable	No
3	Cloudy	Cold	High	Strong	Cold	Unstable	No
4	Clear	Warm	High	Strong	Cold	Stable	Yes
5	Rain	Cold	High	Strong	Warm	Stable	Yes
6	Rain	Cold	High	Weak	Warm	Unstable	No

Naïve Bayes Classifier

- Calculate probabilities and use Bayes theorem:

posterior

$$P(A \mid B) \equiv \frac{P(B \mid A)P(A)}{P(B)}$$

prior

P(B|A)/P(B): support

$$P(go = Yes \mid x = \ldots) \equiv \frac{P(x \mid go = Yes)P(go = Yes)}{P(x)}$$

x={cloudy, cold, normal, strong, warm, unstable}

Naïve Bayes Classifier

$$P(go = Yes \mid x = \ldots) \equiv \frac{P(x \mid go = Yes) P(go = Yes)}{P(x)}$$

x={rain, cold, high, strong, warm, stable}

P(go=Yes) = 3/6 : #Yes / # observations

P(x=...) = 1/6 : #patterns equal to x / # observations

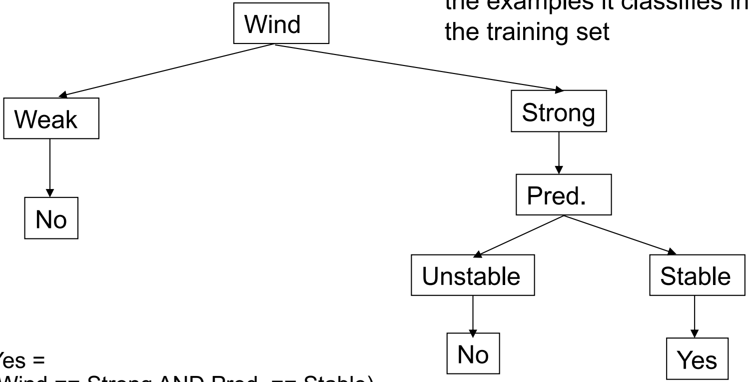
P(x=...|go=Yes) = 1/3 * 1/3 * 2/3 * 3/3 * 2/3 * 3/3 = 4/81 : probability of each attribute equal to its value in x

P(go = Yes | x = ...) = 0,148...

Decision Tree

Equivalent

Each node is associated to the examples it classifies in the training set



Yes =
(Wind == Strong AND Pred. == Stable)

Decision Tree

Homogeneity of a set

The **entropy** of a set is the measure of the diversity of its elements.
A set has the largest entropy if each of its elements belongs to a different class.
The smaller the entropy of a set, the larger its homogeneity!

$$\textbf{Entropy(Set)} = -\Sigma_{i=1, \ldots, n} [P(x_i) \times \log_2(P(x_i))]$$

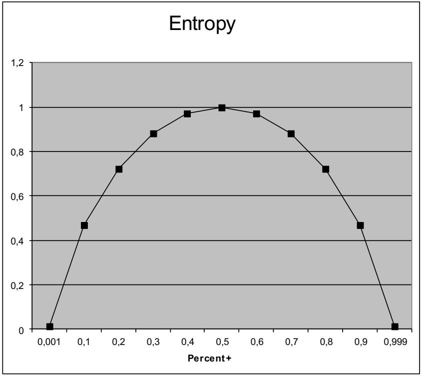
in which, $P(x_i)$ is the probability of picking an element of class x_i.

Example Classes: go_surf(no / yes)

$$\textbf{Entropy(Set)} = -[P(no)\times\log_2(P(no)) + P(yes)\times\log_2(P(yes))]$$

Decision Tree

Homogeneity of a set



Maximum entropy when p+ = p-, i.e. lower probability to predict example class

Decision Tree

The best split

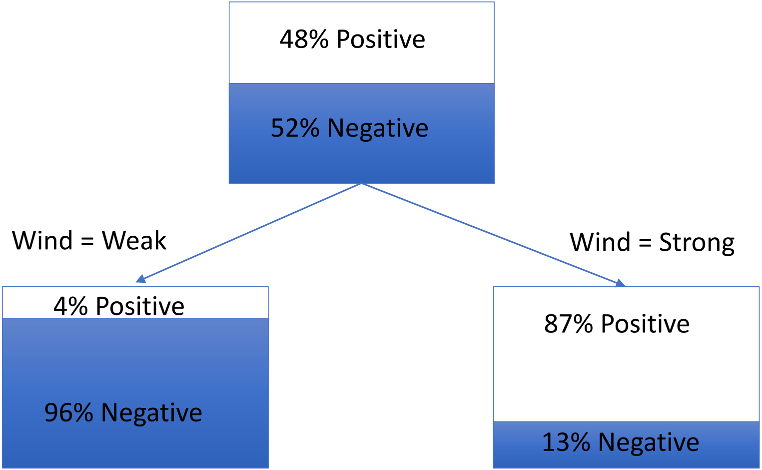
The **best split** is the split that results in the largest entropy reduction, that is, the largest **information gain (IG)**.

IG(Set/{Subset1, Subset2}) = Entropy(Set) – ((1/#Set) × (#Subset1 × Entropy(Subset1) + #Subset2 × Entropy(Subset2)))

in which, $S/\{S_1, S_2\}$ is the split of S into the two subsets S_1 and S_2, and #S is the cardinality of set S.

Decision Tree

Subsets and Gain



Decision Tree

ID3(Examples, Target-Attribute, Attributes)

- Create root
  - If p+ = 1: root = +
  - If p- = 1: root = -
  - If Attributes = ∅, root = most common target-value in examples
- A $\leftarrow$ Attribute with best information gain
- Root = A
- For each (v) possible for A:
  - Add branch A = v
  - ExamplesV = Set of examples where A=v
    - If ExamplesV == ∅: add branch with most common target-value in Examplesv
    - else branch = ID3(ExamplesV, A, Attributes – A)

Decision Tree

C4.5 / C5.0 (Quinlan 96)

- Similar to ID3, but ...
  - Support for continuous attributes: discretizes continuous attributes
  - Allows missing values: examples not used when calculating entropy
  - Allows different costs for attributes
  - Prunning

Learning ensembles

- Boosting (Kearns 88)
  - Can a set of weak learners create a single strong learner?
  - Classification combines the results of all the subtrees
  - Misclassified examples become more important for the error in each iteration
  - New trees are trained to fit the residual error
- Bagging - Bootstrap aggregating: (Breiman 96)
  - Selects randomly the subsets
  - Trains several learners,
  - Classification by voting, regression by averaging

XGBoost (eXtreme Gradient Boosting)

(Chen Guestrin 2016)

- An optimized Gradient Boosting Machine
- Uses many small trees
- Classifies an example by joining the scores of each of the various trees
- Train by adding trees that improve the result or pruning
- Trees are fitted to predict the residual error

Examples ML

- `https://www.youtube.com/watch?v=yeS8TJwBAFs`
  Not on today's subject ...
- `https://www.youtube.com/watch?v=wL7tSgUpy8w&t`

References

- `https://xgboost.readthedocs.io/en/latest/tutorials/model.html`
- Chen, Tianqi; Guestrin, Carlos (2016). "XGBoost: A Scalable Tree Boosting System". In Krishnapuram, Balaji; Shah, Mohak; Smola, Alexander J.; Aggarwal, Charu C.; Shen, Dou; Rastogi, Rajeev (eds.). Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, August 13-17, 2016. ACM. pp. 785–794. arXiv:1603.02754. doi:10.1145/2939672.2939785.

Summary

1. Supervised Learning Problems
   - Regression
   - Classification
2. Linear Approximations
3. Perceptron and Delta Rule
4. Learning
   - Simple Networks
   - Backpropagation
5. Classical Algorithms
   - K-Nearest Neighbours
   - Naïve Bayes Classifier
   - Decision Trees
   - Emsembles