EDAN95
Applied Machine Learning
Lecture 7: Linear Classification and Feed-Forward Networks

Pierre Nugues

Pierre.Nugues@cs.lth.se
http://cs.lth.se/pierre_nugues/

November 22, 2021

Content

Overview and practice of the major neural network architectures:

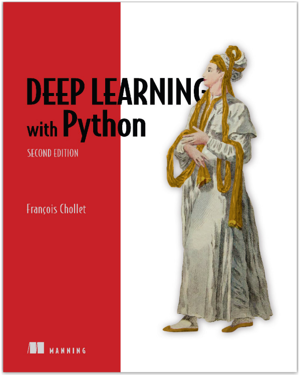
- Feed forward
- Convolutional
- Embeddings
- Recurrent
- LSTM

We will use:

- keras, `https://keras.io/`, an powerful API to design and train network, and
- scikit-learn, `http://scikit-learn.org/stable/`, a general purpose machine-learning toolkit.

The Book

Machine Learning with Python, second edition, by François Chollet (creator of Keras)
All the code examples, as well as explanations are available from github: `https://github.com/fchollet/deep-learning-with-python-notebooks`.



The book webpage: `https://www.manning.com/books/deep-learning-with-python-second-edition`

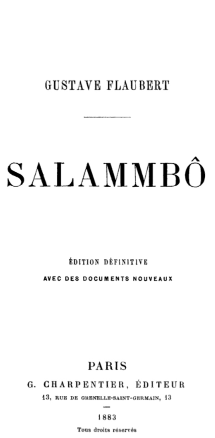
Some Definitions

1. Machine learning always starts with **data sets**: a collection of objects or observations.
2. Machine-learning algorithms can be classified along two main lines: **supervised** and **unsupervised** classification.
3. Supervised algorithms need a **training set**, where the objects are described in terms of attributes and belong to a known class or have a known output.
4. The performance of the resulting classifier is measured against a **test set**.
5. We can also use N-fold cross validation, where the test set is selected randomly from the training set N times, usually 10.
6. Unsupervised algorithms consider objects, where no class is provided.
7. Unsupervised algorithms learn regularities in data sets.

A Data Set: *Salammbô*

A corpus is a collection – a body – of texts.

French original	English translation



GUSTAVE FLAUBERT

SALAMMBÔ

ÉDITION DÉFINITIVE

AVEC DES DOCUMENTS NOUVEAUX

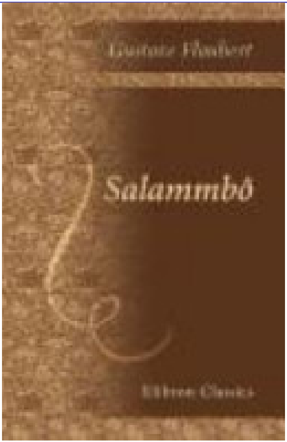
PARIS

G. CHARPENTIER, ÉDITEUR

13, RUE DE GRENELLE-SAINT-GERMAIN, 13

1883

Tous droits réservés



Supervised Learning

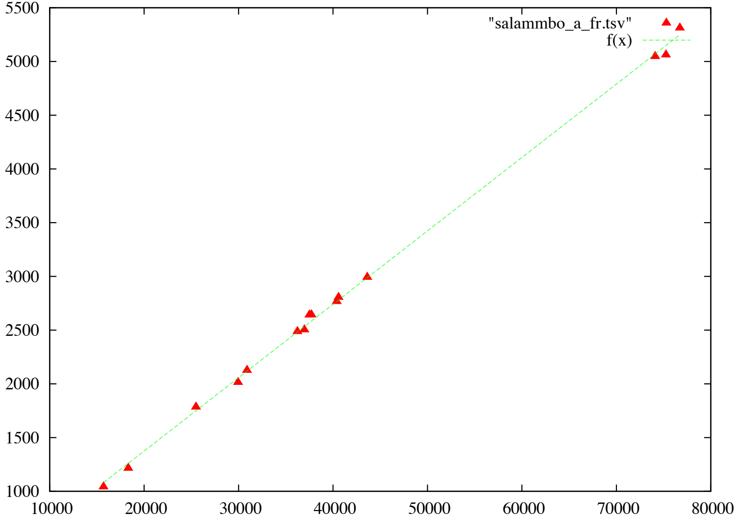
Letter counts from *Salammbô*

Chapter	French		English	
	# characters	# *A*	# characters	# *A*
Chapter 1	36,961	2,503	35,680	2,217
Chapter 2	43,621	2,992	42,514	2,761
Chapter 3	15,694	1,042	15,162	990
Chapter 4	36,231	2,487	35,298	2,274
Chapter 5	29,945	2,014	29,800	1,865
Chapter 6	40,588	2,805	40,255	2,606
Chapter 7	75,255	5,062	74,532	4,805
Chapter 8	37,709	2,643	37,464	2,396
Chapter 9	30,899	2,126	31,030	1,993
Chapter 10	25,486	1,784	24,843	1,627
Chapter 11	37,497	2,641	36,172	2,375
Chapter 12	40,398	2,766	39,552	2,560
Chapter 13	74,105	5,047	72,545	4,597
Chapter 14	76,725	5,312	75,352	4,871
Chapter 15	18,317	1,215	18,031	1,119

Data set: `https://github.com/pnugues/ilppp/tree/master/programs/ch04/salammbo`

Supervised Learning: Regression

Letter count from *Salammbô* in French



Models

We will assume that data sets are governed by functions or models.
For instance given the set:

$$\{(\mathbf{x}_i, y_i) | 0 < i \leqslant N\},$$

there exists a function such that:

$$f(\mathbf{x}_i) = y_i.$$

Supervised machine learning algorithms will produce hypothesized functions or models fitting the data.

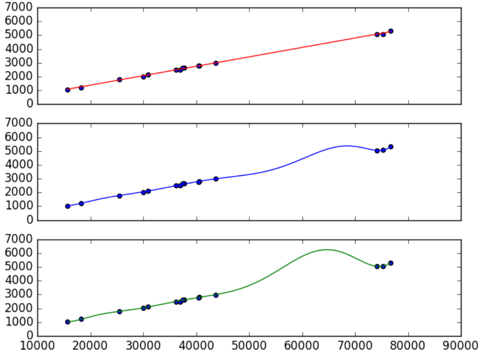
Notations

We will follow these notations:

- $\mathbf{x}$, the vector representing an observation (or sample, or example); in *Salammbô*, an observation is the number of letters in a chapter. We have 15 observations;
- y, the observed response (or target, or output); in programs, the variable names are `y` or `y_true`;
  in *Salammbô*, the number of *A*s in a chapter. We have 15 responses;
- $\hat{y}$, the value predicted by the model; in programs, the variable names are `y_pred` or `y_hat`;
- $\mathbf{w}$, also $\boldsymbol{\beta}$, the weights or parameters of the model, so that $\mathbf{w} \cdot \mathbf{x} = \hat{y}$
- $\mathbf{X}$, the matrix of all the observations
- $\mathbf{y}$, the vector of all the responses

Selecting a Model

Often, multiple models can fit a data set:
Three polynomials of degree: 1, a straight line, 8, and 9 to fit the *Salammbô* dataset.



A general rule in machine learning is to prefer the simplest hypotheses, here the lower polynomial degrees. Otherwise, the model can **overfit** the data.
In our case, the optimal model $\mathbf{w}$ has two parameters: (w_0, w_1).

Process and Algorithms

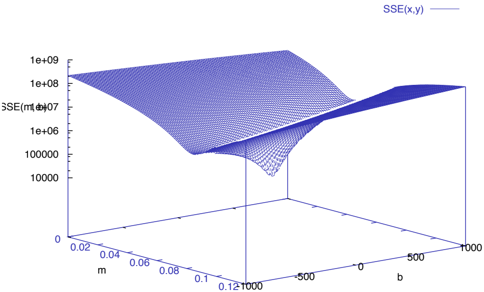
What are the optimal values of $\mathbf{w}$?

The model should minimize the difference between the predicted values and the observed values: This is called the **loss**

For *Salammbô*, the loss is the *mean of the squared errors* (MSE):

$$\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

Visualizing the Loss



To minimize this loss, the solver applies a *stochastic gradient descent* (SGD), or a variant of it, that finds a sequence of model parameters that will reduce the loss.

Keras provides a set of optimizers: sgd, rmsprop, adam, nadam, etc

Minimizing the Loss

The loss function is convex and has a unique minimum.
The loss reaches a minimum when the partial derivatives are zero:

$$
\begin{aligned}
\frac{\partial Loss}{\partial m} &= \sum_{i=1}^{q} \frac{\partial}{\partial m}(y_i - (mx_i + b))^2 &= -2\sum_{i=1}^{q} x_i(y_i - (mx_i + b)) &= 0 \\
\frac{\partial Loss}{\partial b} &= \sum_{i=1}^{q} \frac{\partial}{\partial b}(y_i - (mx_i + b))^2 &= -2\sum_{i=1}^{q} (y_i - (mx_i + b)) &= 0
\end{aligned}
$$

The Gradient Descent

The gradient descent is a numerical method to find the minimum of $f(x_0, x_1, x_2, ..., x_n) = y$, when there is no analytical solution.
Let us denote $\mathbf{x} = (x_0, x_1, x_2, ..., x_n)$
We derive successive approximations to find the minimum of f:

$$f(\mathbf{x}_1) > f(\mathbf{x}_2) > ... > f(\mathbf{x}_k) > f(\mathbf{x}_{k+1}) > .. > min$$

Points in the neighborhood of $\mathbf{x}$ are defined by $\mathbf{x} + \mathbf{v}$ with $||\mathbf{v}||$ small
Given $\mathbf{x}$, find $\mathbf{v}$ subject to $f(\mathbf{x}) > f(\mathbf{x} + \mathbf{v})$

The Gradient Descent (Cauchy, 1847)

Using a Taylor expansion: $f(\mathbf{x}+\mathbf{v}) = f(\mathbf{x}) + \mathbf{v}\cdot\nabla f(\mathbf{x}) + \ldots$

The gradient is a direction vector corresponding to the steepest slope:

$$\nabla f(x_0, x_1, x_2, \ldots, x_n) = (\frac{\partial f}{\partial x_0}, \frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n}).$$

$f(\mathbf{x}+\mathbf{v})$ reaches a minimum or a maximum when $\mathbf{v}$ and $\nabla f(\mathbf{x})$ are colinear:

- Steepest ascent: $\mathbf{v} = \alpha \nabla f(\mathbf{x})$,
- Steepest descent: $\mathbf{v} = -\alpha \nabla f(\mathbf{x})$,

where $\alpha > 0$.

We have then: $f(\mathbf{x} - \alpha \nabla f(\mathbf{x})) \approx f(\mathbf{x}) - \alpha ||\nabla f(\mathbf{x})||^2$.

The inequality:

$$f(\mathbf{x}) > f(\mathbf{x} - \alpha \nabla f(\mathbf{x}))$$

enables us to move one step down to the minimum.

We then use the iteration:

$$\mathbf{x}_{k+1} = \mathbf{x}_k - \alpha_k \nabla f(\mathbf{x}_k).$$

Computing the Gradient

Modern machine-learning platforms use an automatic differentiation algorithm.

1. For a video overview: `https://www.youtube.com/playlist?list=PLZHQObOWTQDNU6R1_67000Dx_ZCJB-3pi`, especially the two last lectures;
2. For a description of it in Tensorflow, see `https://www.tensorflow.org/guide/autodiff`
3. For a more elaborate description: `http://www.cs.toronto.edu/~rgrosse/courses/csc421_2019/slides/lec06.pdf`
4. For a description of the `tf.gradients` class: `https://www.tensorflow.org/api_docs/python/tf/gradients`
5. Something equivalent with PyTorch `https://pytorch.org/tutorials/beginner/blitz/autograd_tutorial.html`

Keras provides a set of optimizers (variants of gradient descent): sgd, rmsprop, adam, nadam, etc

The Matrices

$$\mathbf{X} = \begin{bmatrix} 1 & 36961 \\ 1 & 43621 \\ 1 & 15694 \\ 1 & 36231 \\ 1 & 29945 \\ 1 & 40588 \\ 1 & 75255 \\ 1 & 37709 \\ 1 & 30899 \\ 1 & 25486 \\ 1 & 37497 \\ 1 & 40398 \\ 1 & 74105 \\ 1 & 76725 \\ 1 & 18317 \end{bmatrix}; \mathbf{w} = \begin{bmatrix} 8.7253 \\ 0.0683 \end{bmatrix}; \hat{\mathbf{y}} = \begin{bmatrix} 2533.22 \\ 2988.11 \\ 1080.65 \\ 2483.36 \\ 2054.02 \\ 2780.95 \\ 5148.76 \\ 2584.31 \\ 2119.18 \\ 1749.46 \\ 2569.83 \\ 2767.97 \\ 5070.21 \\ 5249.16 \\ 1259.81 \end{bmatrix}; \mathbf{y} = \begin{bmatrix} 2503 \\ 2992 \\ 1042 \\ 2487 \\ 2014 \\ 2805 \\ 5062 \\ 2643 \\ 2126 \\ 1784 \\ 2641 \\ 2766 \\ 5047 \\ 5312 \\ 1215 \end{bmatrix}; \mathbf{se} = \begin{bmatrix} 913.26 \\ 15.14 \\ 1493.86 \\ 13.25 \\ 1601.31 \\ 578.40 \\ 7527.51 \\ 3444.53 \\ 46.57 \\ 1193.04 \\ 5065.18 \\ 38920 \\ 538.909 \\ 3948.29 \\ 2007.53 \end{bmatrix}.$$

Code Example

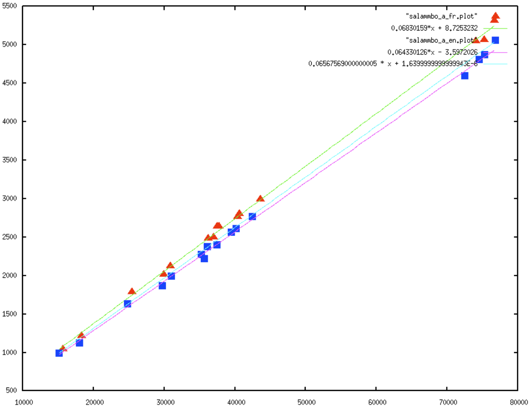
Jupyter Notebook: `1.1-datasetandregression.ipynb`

Classification Dataset

Data set for binary classification: *Salammbô* in French (1) and English (0)

	# char.	# A	class (y)	# char.	# A	class (y)
Chapter 1	36,961	2,503	1	35,680	2,217	0
Chapter 2	43,621	2,992	1	42,514	2,761	0
Chapter 3	15,694	1,042	1	15,162	990	0
Chapter 4	36,231	2,487	1	35,298	2,274	0
Chapter 5	29,945	2,014	1	29,800	1,865	0
Chapter 6	40,588	2,805	1	40,255	2,606	0
Chapter 7	75,255	5,062	1	74,532	4,805	0
Chapter 8	37,709	2,643	1	37,464	2,396	0
Chapter 9	30,899	2,126	1	31,030	1,993	0
Chapter 10	25,486	1,784	1	24,843	1,627	0
Chapter 11	37,497	2,641	1	36,172	2,375	0
Chapter 12	40,398	2,766	1	39,552	2,560	0
Chapter 13	74,105	5,047	1	72,545	4,597	0
Chapter 14	76,725	5,312	1	75,352	4,871	0
Chapter 15	18,317	1,215	1	[illegible]	[illegible]	0

Supervised Learning: Regression and Classification



Given the data set, $\{(\mathbf{x}_i, y_i) | 0 < i \leqslant N\}$ and a model f:

- Classification: $f(\mathbf{x}) = y$ is discrete,
- Regression: $f(\mathbf{x}) = y$ is continuous.

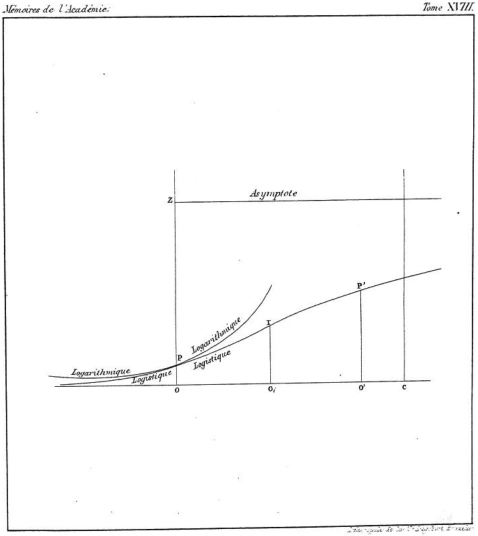
Berkson's Dataset (1944)

Binary classification with probabilities

Drug concentration	Number exposed	Survive Class 0	Die Class 1	Mortality rate	Expected mortality
40	462	352	110	.2359	.2206
60	500	301	199	.3980	.4339
80	467	169	298	.6380	.6085
100	515	145	370	.7184	.7291
120	561	102	459	.8182	.8081
140	469	69	400	.8529	.8601
160	550	55	495	.9000	.8952
180	542	43	499	.9207	.9195
200	479	29	450	.9395	.9366
250	497	21	476	.9577	.9624
300	453	11	442	.9757	.9756

Table: A data set. Adapted and simplified from the original article that described how to apply logistic regression to classification by Joseph Berkson, Application of the Logistic Function to Bio-Assay. *Journal of the American Statistical Association* (1944).

Classification with Probabilities: The Logistic Curve (Verhulst)



$$Logistic(x) = \frac{1}{1+e^{-x}}$$

$$\begin{aligned}\hat{y}(\mathbf{x}) &= Logistic(\mathbf{w}\cdot\mathbf{x})\\ &= \frac{1}{1+e^{-\mathbf{w}\cdot\mathbf{x}}}\end{aligned}$$

The logistic curve is also called a sigmoid

Multiple Classes: Types of Iris

Iris virginica

Iris setosa

Iris versicolor

Courtesy Wikipedia

Fisher's Iris Dataset (1936)

180 MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS

Table I

Iris setosa				*Iris versicolor*				*Iris virginica*			
Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width	Sepal length	Sepal width	Petal length	Petal width
5·1	3·5	1·4	0·2	7·0	3·2	4·7	1·4	6·3	3·3	6·0	2·5
4·9	3·0	1·4	0·2	6·4	3·2	4·5	1·5	5·8	2·7	5·1	1·9
4·7	3·2	1·3	0·2	6·9	3·1	4·9	1·5	7·1	3·0	5·9	2·1
4·6	3·1	1·5	0·2	5·5	2·3	4·0	1·3	6·3	2·9	5·6	1·8
5·0	3·6	1·4	0·2	6·5	2·8	4·6	1·5	6·5	3·0	5·8	2·2
5·4	3·9	1·7	0·4	5·7	2·8	4·5	1·3	7·6	3·0	6·6	2·1
4·6	3·4	1·4	0·3	6·3	3·3	4·7	1·6	4·9	2·5	4·5	1·7
5·0	3·4	1·5	0·2	4·9	2·4	3·3	1·0	7·3	2·9	6·3	1·8
4·4	2·9	1·4	0·2	6·6	2·9	4·6	1·3	6·7	2·5	5·8	1·8
4·9	3·1	1·5	0·1	5·2	2·7	3·9	1·4	7·2	3·6	6·1	2·5
5·4	3·7	1·5	0·2	5·0	2·0	3·5	1·0	6·5	3·2	5·1	2·0
4·8	3·4	1·6	0·2	5·9	3·0	4·2	1·5	6·4	2·7	5·3	1·9
4·8	3·0	1·4	0·1	6·0	2·2	4·0	1·0	6·8	3·0	5·5	2·1
4·3	3·0	1·1	0·1	6·1	2·9	4·7	1·4	5·7	2·5	5·0	2·0
5·8	4·0	1·2	0·2	5·6	2·9	3·6	1·3	5·8	2·8	5·1	2·4
5·7	4·4	1·5	0·4	6·7	3·1	4·4	1·4	6·4	3·2	5·3	2·3
5·4	3·9	1·3	0·4	5·6	3·0	4·5	1·5	6·5	3·0	5·5	1·8
5·1	3·5	1·4	0·3	5·8	2·7	4·1	1·0	7·7	3·8	6·7	2·2
5·7	3·8	1·7	0·3	6·2	2·2	4·5	1·5	7·7	2·6	6·9	2·3
5·1	3·8	1·5	0·3	5·6	2·5	3·9	1·1	6·0	2·2	5·0	1·5
5·4	3·4	1·7	0·2	5·9	3·2	4·8	1·8	6·9	3·2	5·7	2·3
5·1	3·7	1·5	0·4	6·1	2·8	4·0	1·3	5·6	2·8	4·9	2·0

Categorical Attributes. After Quinlan (1986)

Object	Attributes				Class
	Outlook	Temperature	Humidity	Windy	
1	Sunny	Hot	High	False	N
2	Sunny	Hot	High	True	N
3	Overcast	Hot	High	False	P
4	Rain	Mild	High	False	P
5	Rain	Cool	Normal	False	P
6	Rain	Cool	Normal	True	N
7	Overcast	Cool	Normal	True	P
8	Sunny	Mild	High	False	N
9	Sunny	Cool	Normal	False	P
10	Rain	Mild	Normal	False	P
11	Sunny	Mild	Normal	True	P
12	Overcast	Mild	High	True	P
13	Overcast	Hot	Normal	False	P
14	Rain	Mild	High	True	N

Linear Classification

We represent classification using a threshold function (a variant of the signum function):

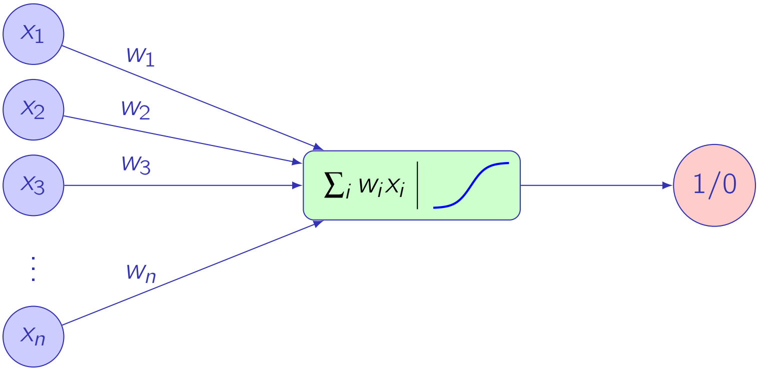
$$H(\mathbf{w} \cdot \mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

The classification function associates P with 1 and N with 0.
We want to find the separating hyperplane:

$$\begin{aligned} \hat{y}(\mathbf{x}) &= H(\mathbf{w} \cdot \mathbf{x}) \\ &= H(w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n), \end{aligned}$$

given a data set of q examples: $DS = \{(1, x_1^j, x_2^j, \ldots, x_n^j, y^j) | j : 1..q\}$.
We use $x_0 = 1$ to simplify the equations.
For a binary classifier, y has then two possible values $\{0, 1\}$ corresponding in our example to $\{\text{French}, \text{English}\}$.

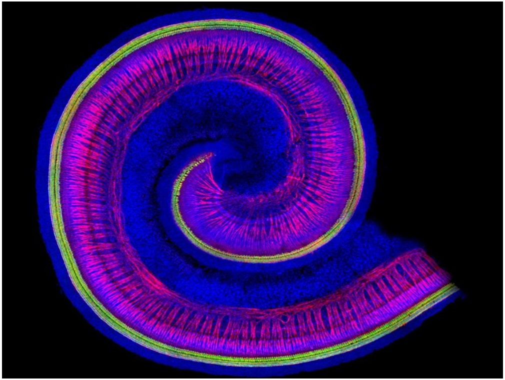
Logistic Regression as a Neural Network



Code Example

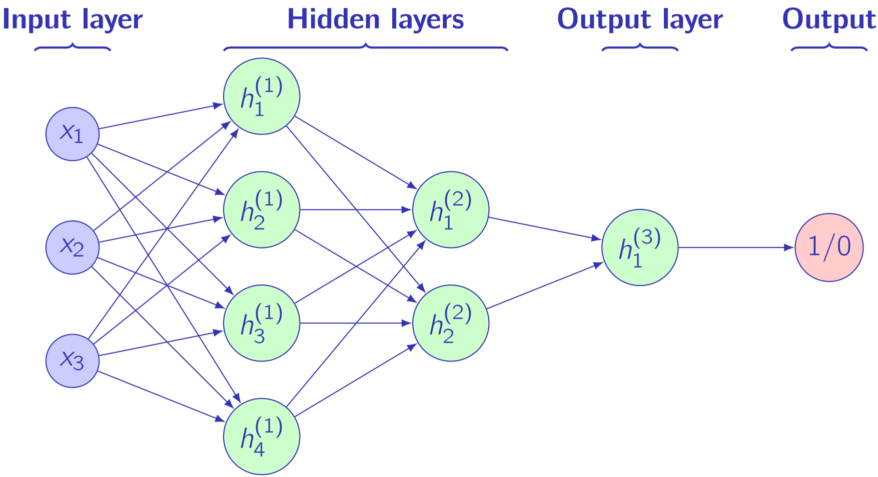
Jupyter Notebook: `1.2-salammboclassification.ipynb`, first part

Neural Networks



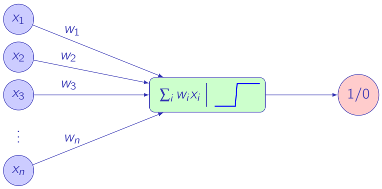
A photomicrograph showing the classic view of the snail-shaped cochlea with hair cells stained green and neurons showing reddish-purple. [Decibel Therapeutics (https://www.decibeltx.com)]. Source: https://www.genengnews.com/insights/targeting-the-inner-ear/

Neural Networks

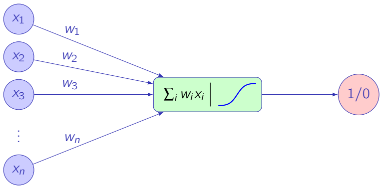


Activation Functions

The perceptron



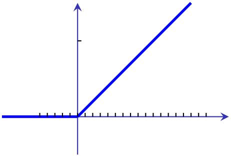
Logistic regression



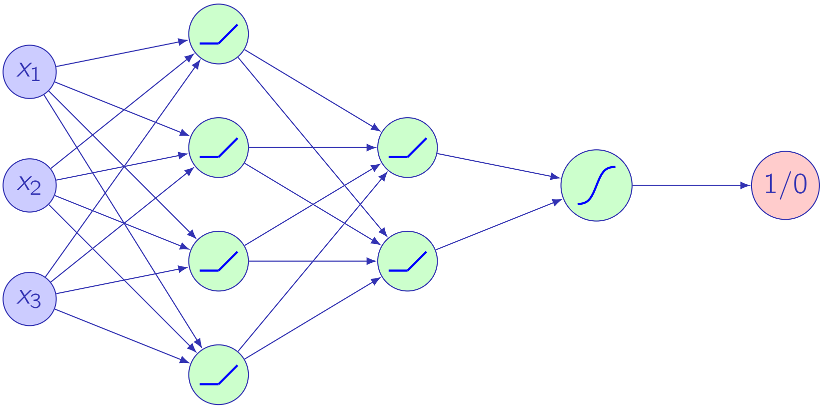
Activation Functions

Rectified linear unit (ReLU), where

$$reLU(x) = \max(0, x).$$



Neural Networks with Hidden Layers



Code Example

Jupyter Notebook: `1.2-salammboclassification.ipynb`, second part

Matrix Notation

- A feature vector (predictors): $\mathbf{x}$, and feature matrix: $\mathbf{X}$;
- The class: y and the class vector: $\mathbf{y}$;
- The predicted class (response): $\hat{y}$, and predicted class vector: $\hat{\mathbf{y}}$

$$\mathbf{X} = \begin{bmatrix} \mathit{Sunny} & \mathit{Hot} & \mathit{High} & \mathit{False} \\ \mathit{Sunny} & \mathit{Hot} & \mathit{High} & \mathit{True} \\ \mathit{Overcast} & \mathit{Hot} & \mathit{High} & \mathit{False} \\ \mathit{Rain} & \mathit{Mild} & \mathit{High} & \mathit{False} \\ \mathit{Rain} & \mathit{Cool} & \mathit{Normal} & \mathit{False} \\ \mathit{Rain} & \mathit{Cool} & \mathit{Normal} & \mathit{True} \\ \mathit{Overcast} & \mathit{Cool} & \mathit{Normal} & \mathit{True} \\ \mathit{Sunny} & \mathit{Mild} & \mathit{High} & \mathit{False} \\ \mathit{Sunny} & \mathit{Cool} & \mathit{Normal} & \mathit{False} \\ \mathit{Rain} & \mathit{Mild} & \mathit{Normal} & \mathit{False} \\ \mathit{Sunny} & \mathit{Mild} & \mathit{Normal} & \mathit{True} \\ \mathit{Overcast} & \mathit{Mild} & \mathit{High} & \mathit{True} \\ \mathit{Overcast} & \mathit{Hot} & \mathit{Normal} & \mathit{False} \\ \mathit{Rain} & \mathit{Mild} & \mathit{High} & \mathit{True} \end{bmatrix}; \mathbf{y} = \begin{bmatrix} \mathrm{N} \\ \mathrm{N} \\ \mathrm{P} \\ \mathrm{P} \\ \mathrm{P} \\ \mathrm{N} \\ \mathrm{P} \\ \mathrm{N} \\ \mathrm{P} \\ \mathrm{P} \\ \mathrm{P} \\ \mathrm{P} \\ \mathrm{P} \\ \mathrm{N} \end{bmatrix}$$

Converting Symbolic Attributes into Numerical Vectors

Linear classifiers are numerical systems.

Symbolic – nominal – attributes are mapped onto vectors of binary values.

This is called a one-hot encoding
A conversion of the weather data set.

Object	Attributes										Class
	Outlook			Temperature			Humidity		Windy		
	Sunny	Overcast	Rain	Hot	Mild	Cool	High	Normal	True	False	
1	1	0	0	1	0	0	1	0	0	1	*N*
2	1	0	0	1	0	0	1	0	1	0	*N*
3	0	1	0	1	0	0	1	0	0	1	*P*
4	0	0	1	0	1	0	1	0	0	1	*P*
5	0	0	1	0	0	1	0	1	0	1	*P*
6	0	0	1	0	0	1	0	1	1	0	*N*
7	0	1	0	0	0	1	0	1	1	0	*P*
8	1	0	0	0	1	0	1	0	0	1	*N*
9	1	0	0	0	0	1	0	1	0	1	*P*
10	0	0	1	0	1	0	0	1	0	1	*P*
11	1	0	0	0	1	0	0	1	1	0	*P*
12	0	1	0	0	1	0	1	0	1	0	*P*
13	0	1	0	1	0	0	0	1	0	1	*P*
14	0	0	1	0	1	0	1	0	1	0	*N*

Loss

The loss function is defined as $L(y, \hat{y})$ with $\hat{y} = h(\mathbf{x})$, where $\mathbf{x}$ is the vector of attributes, h the classifier, and y, the correct value.

Absolute value loss	$L_1(y, \hat{y})$	$=$	$\lvert y - \hat{y} \rvert$
Squared error loss	$L_2(y, \hat{y})$	$=$	$(y - \hat{y})^2$
0/1 loss	$L_{0/1}(y, \hat{y})$	$=$	0 if $y = \hat{y}$ else 1
Binary crossentropy			
Categorical crossentropy			

For Keras, see here: `https://keras.io/losses/` for the available losses

Empirical Loss

We compute the empirical loss of a classifier h on a set of examples E using the formula:

$$Loss(L, E, h) = \frac{1}{N}\sum_{E} L(y, h(x)).$$

For continuous functions:

$$Loss(L, E, h) = \frac{1}{N}\sum_{E} (y - h(x))^2.$$

Understanding the Loss: Entropy

Information theory models a text as a sequence of symbols.
Let $x_1, x_2, \ldots, x_N$ be a discrete set of N symbols representing the characters.
The information content of a symbol is defined as

$$I(x_i) = -\log_2 P(x_i) = \log_2 \frac{1}{P(x_i)},$$

where

$$P(x_i) = \frac{Count(x_i)}{\sum_{j=1}^{n} Count(x_j)}.$$

Entropy, the average information content, is defined as:

$$H(X) = -\sum_{x \in X} P(x) \log_2 P(x),$$

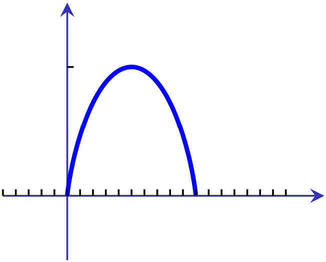
By convention: $0 \log_2 0 = 0$.

Understanding the Entropy

For a two-class set, we set:

$$x = \frac{p}{p+n} \text{ and } \frac{n}{p+n} = 1 - x.$$

$$I(x) = -x\log_2 x - (1-x)\log_2(1-x) \text{ with } x \in [0,1].$$



The entropy reaches a maximum when there are as many positive as negative examples in the data set. It is minimal when the set consists of either positive or negative examples.

Entropy of a Text

The entropy of the text is

$$
\begin{aligned}
H(X) &= -\sum_{x \in X} P(x) \log_2 P(x). \\
&= -P(A)\log_2 P(A) - P(B)\log_2 P(B) - \ldots \\
&\quad -P(Z)\log_2 P(Z) - P(\grave{A})\log_2 P(\grave{A}) - \ldots \\
&\quad -P(\ddot{Y})\log_2 P(\ddot{Y}) - P(blanks)\log_2 P(blanks).
\end{aligned}
$$

Entropy of Gustave Flaubert's *Salammbô* in French is $H(X) = 4.39$.

Cross-Entropy

Common losses in classification: binary or multinomial (categorical) crossentropy.

The cross entropy of M on P is defined as:

$$H(P,M) = -\sum_{x \in X} P(x) \log_2 M(x).$$

We have the inequality $H(P) \leq H(P,M)$.

The difference is called the Kullback-Leibler divergence.

	Entropy	Cross entropy	Difference
Salammbô, chapters 1-14, training set	4.39481	4.39481	0.0
Salammbô, chapter 15, test set	4.34937	4.36074	0.01137
Notre Dame de Paris, test set	4.43696	4.45507	0.01811
Nineteen Eighty-Four, test set	4.35922	4.82012	0.46090

Cross Entropy for Binary Cases

In practice, we use the mean and the natural logarithm:

$$H(P,M) = -\frac{1}{|X|}\sum_{x \in X} P(x) \log M(x),$$

where P is the truth, and M is the prediction of the model, a probability in the case of logistic regression.

In binary classification:

- $P(x) = 1$
- $M(x)$ is the predicted probability of being class 1.
- If the observation belongs to class 0, its predicted probability is $1 - M(x)$.

Example of Cross Entropy

Computing the cross-entropy of six observations:

Observations	1	2	3	4	5	6
Dose	140	300	140	160	140	250
Observed class (Truth)	0	1	1	1	1	1
Model prediction of being class 1	0.3487	0.9964	0.8557	0.9056	0.8557	0.9882
Model prediction of being class 0	0.6513					
$-P(x)\log M(x)$:	0.4287	0.0036	0.1559	0.0992	0.1559	0.0119

Mean = 0.14252826

Code Example

Jupyter Notebook: `1.3-logisticregression.ipynb`

Multiple Categories

We can generalize logistic regression to multiple categories.
We use then the *softmax* function:

$$P(y = i|\mathbf{x}) = \frac{e^{-\mathbf{w}_i \cdot \mathbf{x}}}{\sum_{j=1}^{C} e^{-\mathbf{w}_j \cdot \mathbf{x}}},$$

that defines the probability of an observation represented by $\mathbf{x}$ to belong to class i.
Again, we use stochastic gradient descent to compute the weights: $\mathbf{w}$.

Representing y

In Keras, the default representation of y and $\hat{y}$ are vectors (as opposed to sklearn)

y is an indicator vector (one-hot) and $\hat{y}$, a probability distribution

```
y[:5]
> array([2, 1, 0, 2, 0])
```

```
from keras.utils import to_categorical
Y_cat = to_categorical(y)
Y_cat[:5]
```

```
> array([[0., 0., 1.],
         [0., 1., 0.],
         [1., 0., 0.],
         [0., 0., 1.],
         [1., 0., 0.]], dtype=float32)
```

A complete example

The original categories:

```
y[121:126]
 [2 0 0 2 0]
```

The encoded categories:

```
Y_cat[121:126]
 [[0. 0. 1.]
 [1. 0. 0.]
 [1. 0. 0.]
 [0. 0. 1.]
 [1. 0. 0.]]
```

The predicted probabilities:

```
model.predict(X[121:126])
 [[9.4238410e-12 2.8314255e-03 9.9716860e-01]
 [9.9939132e-01 6.0863607e-04 2.5036247e-11]
 [9.9859804e-01 1.4019267e-03 3.5701425e-10]
 [1.2004078e-09 2.8088816e-02 9.7191113e-01]
 [9.9938595e-01 6.1400887e-04 2.7445022e-11]]
```

The predicted classes:

```
list(map(np.argmax, model.predict(X[121:126])))
 [2, 0, 0, 2, 0]
```

The loss, probability of the truth.

$$-\frac{1}{N}(\log(9.9716860\cdot 10^{-1}) + \log(9.9939132\cdot 10^{-1}) + \log(9.9859804\cdot 10^{-1}) + \ldots)$$

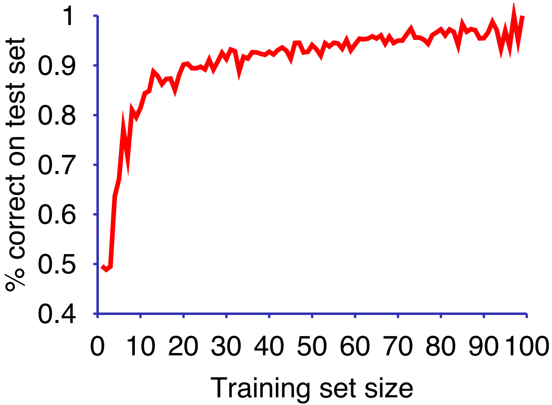
In the example, the prediction is also the truth. This is not always the case.

Code Example

Jupyter Notebook: `1.4-multiclass.ipynb`

Learning Curve

The classical evaluation technique uses a training set and a test set.
Generally, the larger the training set, the better the performance.
This can be visualized with a learning curve. From the textbook, Stuart Russell and Peter Norvig, *Artificial Intelligence*, 3rd ed., 2010, page 703.



Overfitting

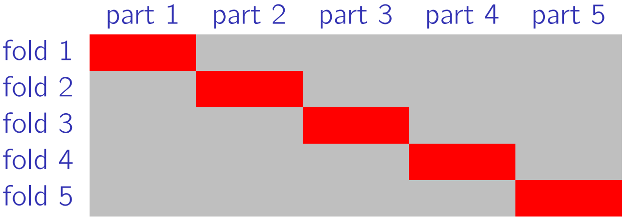
- When two classifiers have equal performances on a specific test set, the simplest one is supposed to be more general
- A small network is always preferable to a larger one.
- Complex classifiers may show an overfit to the training data and have poor performance when the data set changes.
- We assess the overfit by drawing the loss and accuracy curves for the training set and a separate validation set. See the examples in the Keras book.

Evaluation

- The standard evaluation procedure is to train the classifier on a training set and evaluate the performance on a test set.
- When we have only one set, we divide it in two subsets: the training set and the test set (or holdout data).
- The split can be 90–10 or 80–20
- This often optimizes the classifier for a specific test set and creates an overfit

Cross Validation

- A N-fold cross validation mitigates the overfit
- The set is partitioned into N subsets, $N = 5$ for example, one of them being the test set (red) and the rest the training set (gray).
- The process is repeated N times with a different test set: N folds



At the extreme, leave-one-out cross-validation

Model Selection

- Validation can apply to one classification method
- We can use it to select a classification method and its parametrization.
- Needs three sets: training set, development set, and test set.

Measuring Quality: The Confusion Matrix

A task in natural language processing: Identify the parts of speech (POS) of words.

Example: *The can rusted*

- The human: *The*/art (DT) *can*/noun (NN) *rusted*/verb (VBD)
- The POS tagger: *The*/art (DT) *can*/modal (MD) *rusted*/verb (VBD)

↓Correct	Tagger → DT	IN	JJ	NN	RB	RP	VB	VBD	VBG	VBN
DT	99.4	0.3	–	–	0.3	–	–	–	–	–
IN	0.4	97.5	–	–	1.5	0.5	–	–	–	–
JJ	–	0.1	93.9	1.8	0.9	–	0.1	0.1	0.4	1.5
NN	–	–	2.2	95.5	–	–	0.2	–	0.4	–
RB	0.2	2.4	2.2	0.6	93.2	1.2	–	–	–	–
RP	–	24.7	–	1.1	12.6	61.5	–	–	–	–
VB	–	–	0.3	1.4	–	–	96.0	–	–	0.2
VBD	–	–	0.3	–	–	–	–	94.6	–	4.8
VBG	–	–	2.5	4.4	–	–	–	–	93.0	–
VBN	–	–	4.6	–	–	–	–	4.3	–	90.6

After Franz (1996, p. 124)