Machine Learning

Andrea Asperti

Department of Computer Science, University of Bologna
Mura Anteo Zamboni 7, 40127, Bologna, ITALY
asperti@cs.unibo.it

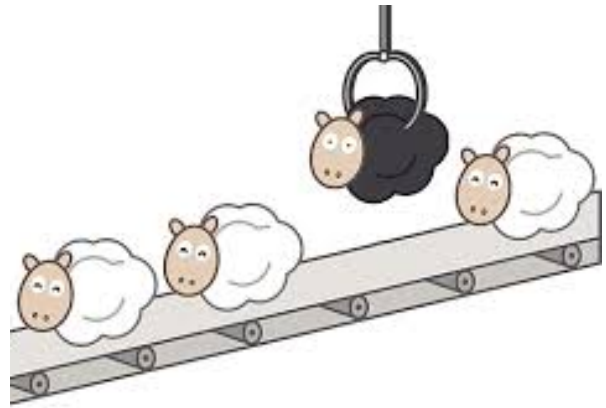
What Machine Learning is about

Target

Problems difficult to address by algorithmic means:

- classification problems

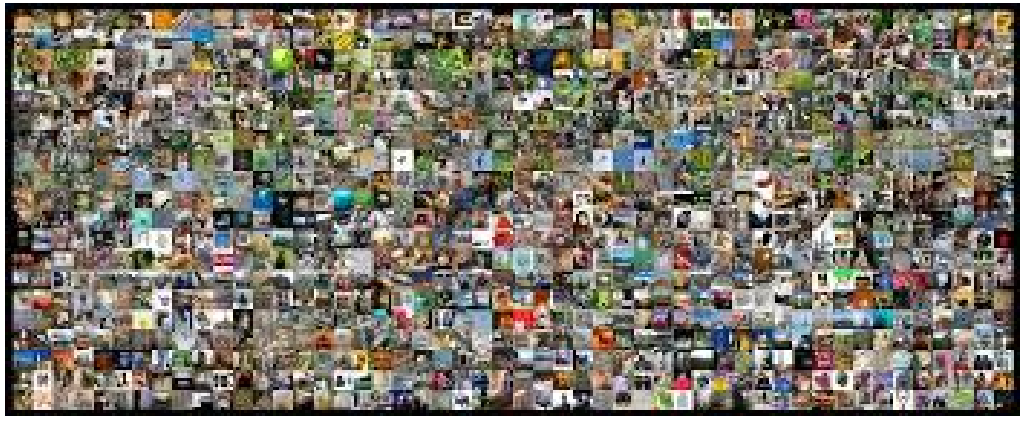
  spam detection, sentiment analysis, fraudulent transactions, ...
- image recognition
- speech/music recognition
- autoencoding
- data mining (e.g. clustering)
- ...



More and more problems addressed by Machine Learning techniques

Characteristics

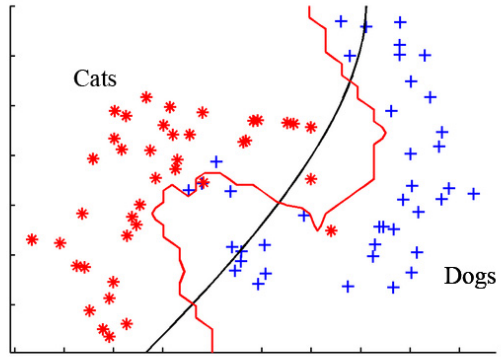
- low (metatheoretical) knowledge
- large number of input features
- big volume of training data
- adaptability



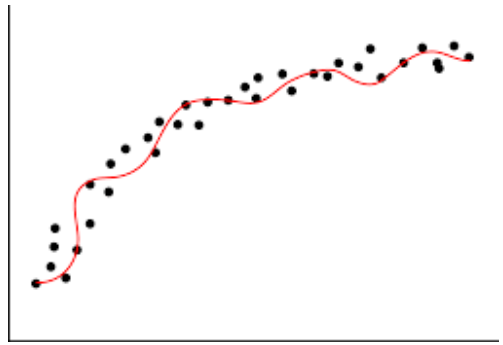
Imagenet database: $\approx$ 10 million images

The Machine Learning approach

- define a **model** for the task to be solved
  the model depends on a set of **parameters** Θ
- define a **performance metric**:
  some **error** measure to evaluate the model
- tune the parameters Θ to **minimize** the error on the **training set**



classification



regression

Machine learning is an optimization process

Why Learning?

Machine Learning problems are in fact **optimization problems**!
So, why talking about learning?

- The tuning of parameters is based on observations (**training set**). We learn from our past experience.

Why Learning?

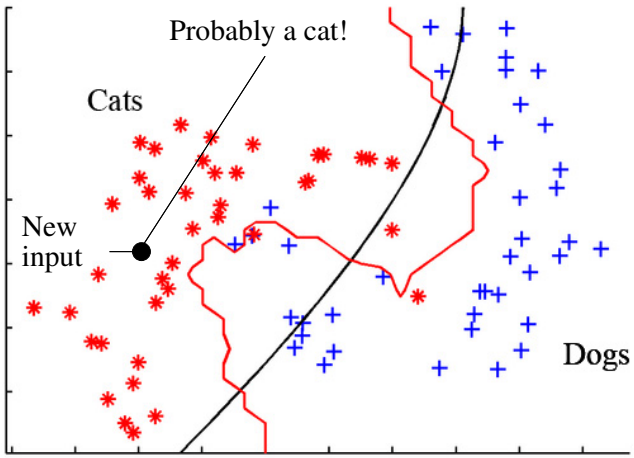
Machine Learning problems are in fact **optimization problems**!
So, why talking about learning?

- The tuning of parameters is based on observations (**training set**). We learn from our past experience.

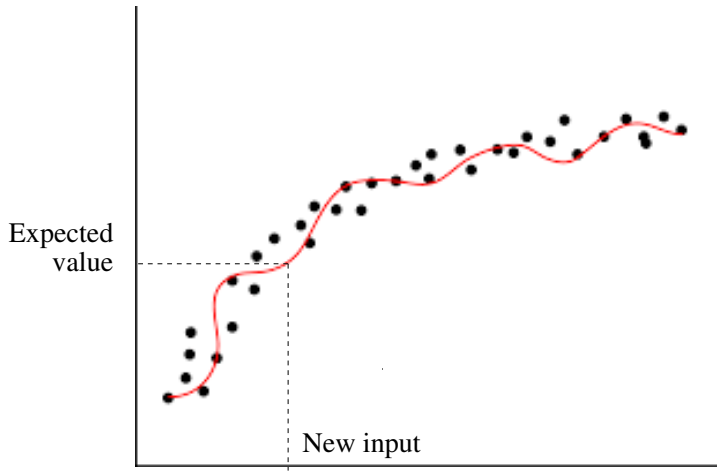
- the solution to the optimization problem is not given in an analytical form (often there is no closed form solution).

  We use **iterative** techniques (e.g. gradient descent) to progressively approximate the result, and this can be be understood as a form of learning process.

Goal: making predictions



classification

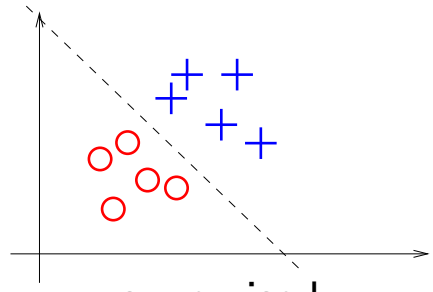


regression

Models can also be used to acquire knowledge on input data: finding clusters, making correlations, etc.

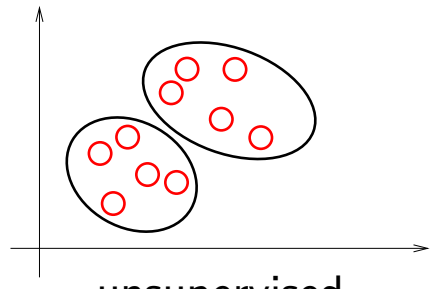
Different types of Learning Tasks

- **supervised learning**:
  inputs + outputs (labels)
  - classification
  - regression



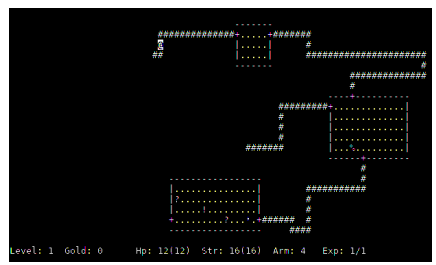
supervised

- **unsupervised learning**:
  just inputs
  - clustering
  - component analysis
  - autoencoding



unsupervised

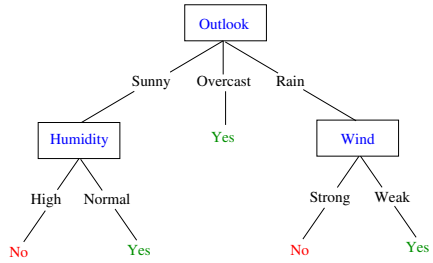
- **reinforcement learning**
  actions and rewards
  - learning long-term gains
  - "model-free" planning



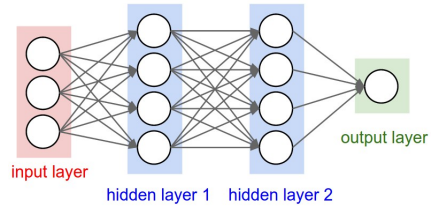
reinforcement

Many different techniques

- **Different ways to define the models**:
  - decision trees
  - mathematical expressions
  - neural networks
  - ...

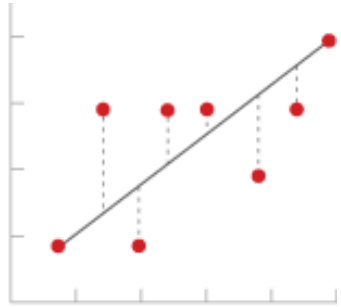


decision tree

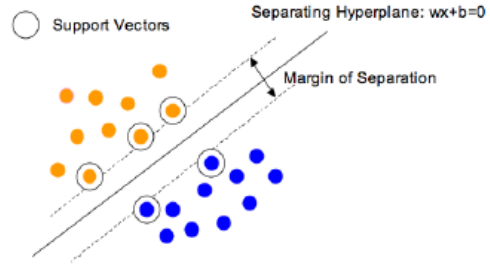


neural net

- **Different error (loss) functions**:
  - mean squared errors
  - logistic loss
  - cross entropy
  - cosine distance
  - maximum margin
  - ...



mean squared errors



maximum margin

Features

Features

Any information relative to a datum that describes some of its relevant properties is called a feature.

Features are the input of the learning process.

Learning is highly sensible to the choice of features.

Choosing good features is difficult (requires good domain knowledge).

Example of features

medical diagnosis	user profiling	weather forecasting
symptoms	demographic data	temperature
patient condition	personal interests	humidity
medical record	social communities	pressure
result of exams	life style	rain, wind, ...
...	...	...

Features in image processing

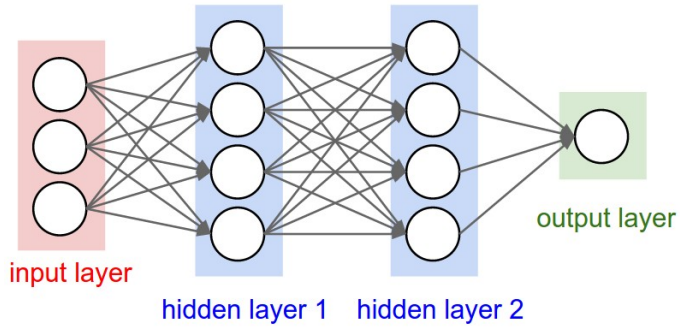
Picture taken from the OpenCv python-tutorial

Deep learning

Old approach: compute by hand good features, and apply a simple and well understood (e.g. linear) learning algorithm.

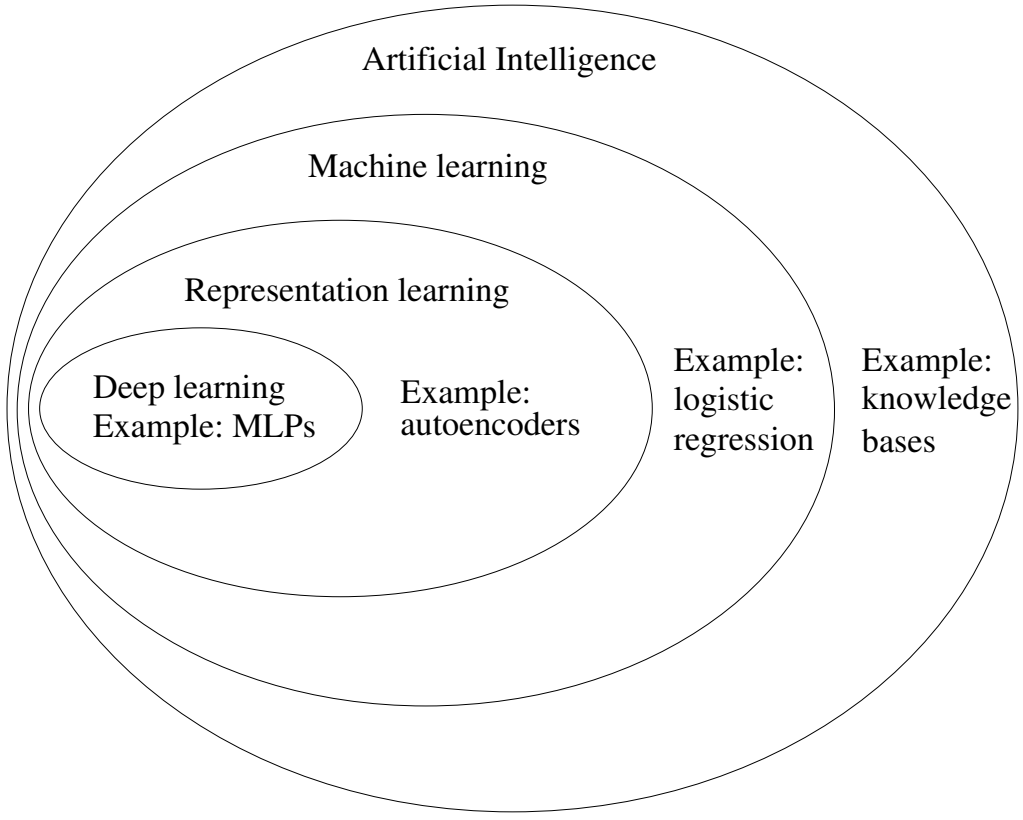
Modern (deep) approach: supply raw data and let to the machine the burden to synthesize good features (internal representations).

Deep learning is implemented through neural networks (NN)
A NN is deep when we have multiple hidden layers:



each hidden layer computes new features from previous ones.

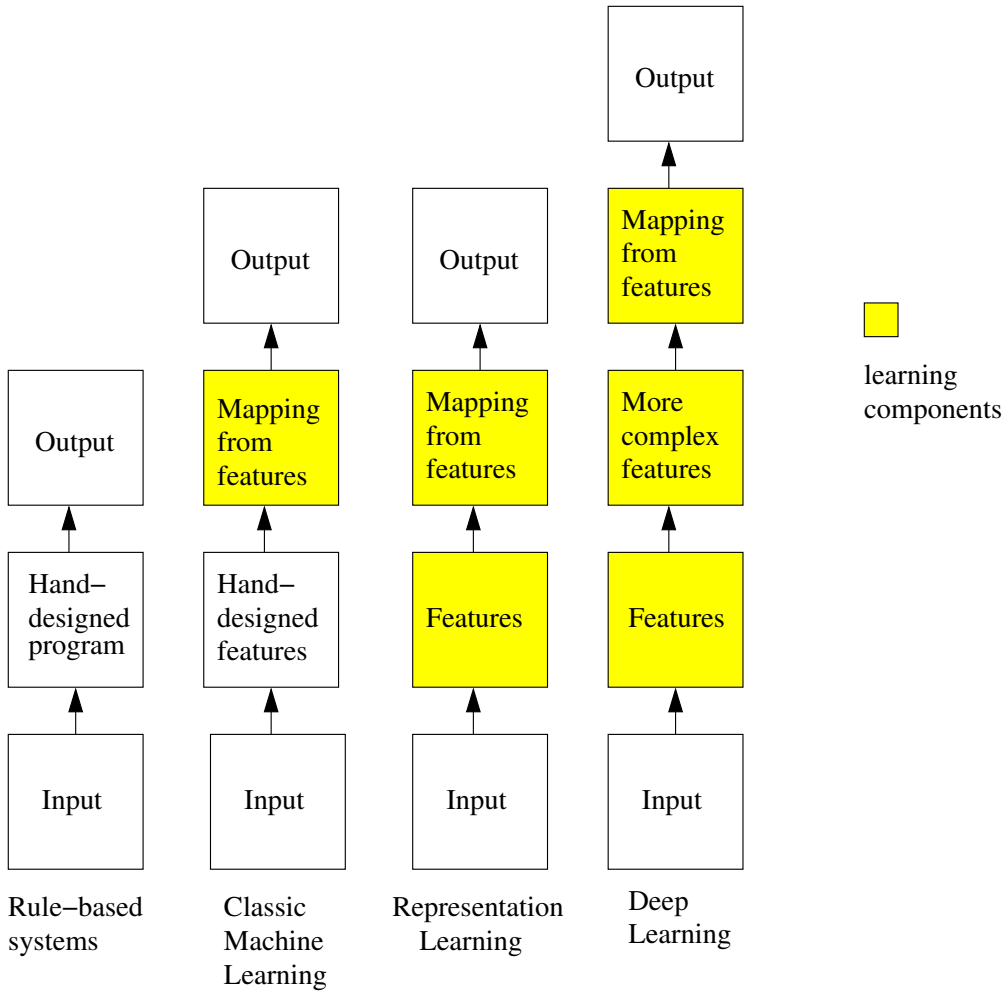
Relations between research areas



From "Deep Learning" by Y.Bengio, I.Goodfellow, A.Courville, MIT Press.

For a more historical perspective on the different fields consult my blog.

Components trained to learn



From "Deep Learning" by Y.Bengio, I.Goodfellow, A.Courville, MIT Press.

AI, machine learning, deep learning

- **Knowledge-based systems**: take an expert, ask him how he solves a problem and try to mimic his approach by means of logical rules

AI, machine learning, deep learning

- **Knowledge-based systems**: take an expert, ask him how he solves a problem and try to mimic his approach by means of logical rules
- **Traditional Machine-Learning**: take an expert, ask him what are the features of data relevant to solve a given problem, and let the machine learn the mapping

AI, machine learning, deep learning

- **Knowledge-based systems**: take an expert, ask him how he solves a problem and try to mimic his approach by means of logical rules
- **Traditional Machine-Learning**: take an expert, ask him what are the features of data relevant to solve a given problem, and let the machine learn the mapping
- **Deep-Learning**: get rid of the expert

Didactic Information

- structure of the course
- books, tutorials and blogs
- software
- examination
- office hours

Structure of the course

Part I: Machine Learning

- Decision Trees
- Probability basics
- Naif Bayes
- Maximum Likelihood estimation
- Logistic regression
- The gradient technique
- Generative vs Discriminative
- Multinomial regression
- Linear regression

Part II: Deep Learning

- Neural networks
- Backpropagation and Training
- Convolutions
- Convolutional Networks
- Autoencoders
- Generative Adversarial Networks
- Object detection and segmentation
- Recurrent Networks
- Attention & Transformers

Books

(Shallow) Machine Learning

- C.M.Bishop. Pattern Recognition and Machine Learning. Springer 2006.
- E. Alpaydin, Introduction to Machine Learning, Cambridge University Press, 2010.

Deep Learning

- Y.Bengio, I.Goodfellow and A.Courville. Deep Learning, MIT Press to appear.
- Dive into deep learning (D2L)

Online Tutorials and Blogs

Pssible to study on online material (fast updating):

- Tensorflow tutorials
- Deep Learning Tutorial. LISA lab. University of Montreal.
- Deep Mind blog
- Open AI blog
- Keras blog
- towardsdatascience
- Machine learning tutorial with Python
- a lot of interesting lessons and seminars on youtube
- a lot of material on github
- ...

Suggested software

Machine learning "legacy" techniques:
- Scikit-learn

Neural Networks and Deep Learning:
- TensorFlow, Google Brain
- Keras, F.Chollet
- PyTorch, Facebook
- MXNET, Apache
- ...

Image and signal processing:
- OpenCV
- Scipy

Assessment

- individual test (40%)
- individual project (60%)

The topic of the project will be given to you (on virtuale) 7-8 days before the scheluded date of the exam, and you need to submit your solution (a single python notebook) within the specified deadline. Typically, you have a week time to complete the task.

The topic is different for each session.

The individual test, once passed, does not need to be repeated.

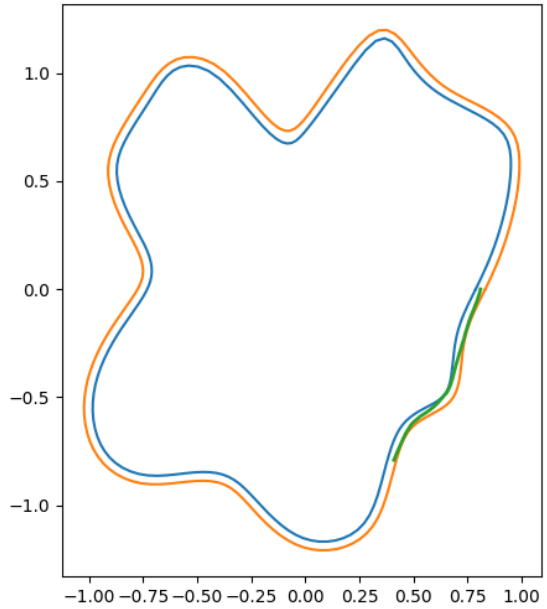
Extra credits: MicroRacer

Organize and participate in the MicroRacer annual Championship.

MicroRacer is a small didactic environment for testing DRL techniques in a racing setting, presented at the 8th International Conference on Machine Learning, Optimization, and Data Science. Here is the article.

- random track generation
- fast training (15-30 mn)
- baselines provided

- 1 point for participating
- 1 extra point for second place
- 2 extra points for first place



Office hours and links

Wednesday, 11-13

Prof.Andrea Asperti
andrea.asperti@unibo.it
Via Malaguti 1D

- homepage:https://www.unibo.it/sitoweb/andrea.asperti
- my own deeplearningblog
- deepfridays seminars

Decision Trees

Function Approximation

Train set: a set of training examples

$$\langle x^{(i)}, y^{(i)} \rangle$$

where

- $x^{(i)} \in X$ (set of inputs)
- $y^{(i)} \in Y$ (set of outputs)
- i is the instance of the training sample

Problem: "learn" the function mapping $x^{(i)}$ to $y^{(i)}$

Y discrete: classification problem (class prediction)
Y continuous: regression problem (value prediction).

Hypothesis Space

Machine learning techniques require a commitment to a given function space H, inside which we look for the function that provides the best approximation for the training set.
H reflects the way we are modeling data.

Example: Training set $= \{\langle 2, 3\rangle, \langle 3, 4\rangle\}$

Hypothesis Space

Machine learning techniques require a commitment to a given function space H, inside which we look for the function that provides the best approximation for the training set.
H reflects the way we are modeling data.

Example: Training set $= \{\langle 2, 3\rangle, \langle 3, 4\rangle\}$

$H =$ linear functions. We have an exact solution: $y = x + 1$

Hypothesis Space

Machine learning techniques require a commitment to a given function space H, inside which we look for the function that provides the best approximation for the training set.
H reflects the way we are modeling data.

Example: Training set $= \{\langle 2, 3\rangle, \langle 3, 4\rangle\}$

$H =$ linear functions. We have an exact solution: $y = x + 1$

Adding the instance $\langle 1, 0\rangle$, the model is not exact any more.

- we keep the model and content ourselves with approximate answers
- we change the model, for instance taking quadratic functions $y = -x^2 + 6x - 5$

Overfitting and underfitting

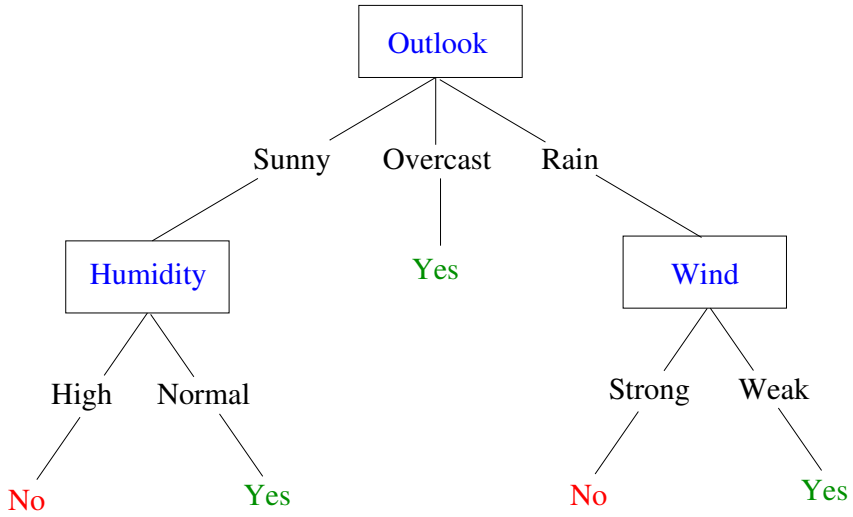
overfitting the model is too complex and specialized over the peculiarities of the instances of the training set

underfitting the model is too simple and does not allow to express the complexity of the observations.

Decision trees: an example

A good day to play tennis?

F : Outlook $\times$ Humidity $\times$ Wind $\times$ Temp $\rightarrow$ Play Tennis?



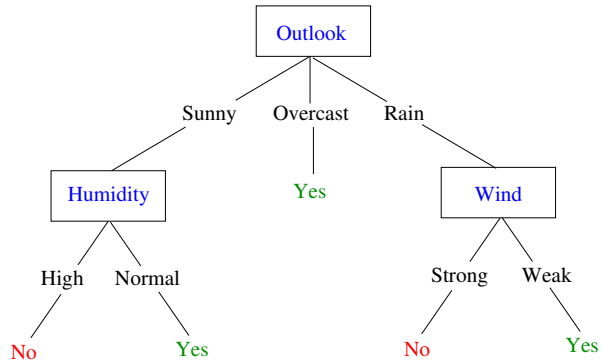
Every node tests an attribute (feature) X

Each branch corresponds to one of the possible discrete values of X

Every leaf predicts the answer Y (or a probability $P(Y|X)$)

Learn to build a decision tree

Problem configuration



- Input set X
  every instance $x \in X$ is a vector of features of the following kind:
  $\langle$ Humidity=high, Wind=weak, Outlook=rain, Temp=hot $\rangle$
- Target function $f : X \to Y$
  Y takes discrete values (booleans)
- Hypothesis Space $H = \{h \mid h : X \to Y\}$ (no restriction)
  - we try to model each $h \in H$ with a decision tree
  - each instance x defines a path in the tree leading to a leaf labelled with y

Play-tennis training set

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Input and output of the learning task

Input A set of training samples $\{\langle x^{(i)}, y^{(i)} \rangle\}$ relative to the target function f

Output The hypothesis $h \in H$ that best approximates f

But f is unknown ... so?

Input and output of the learning task

Input A set of training samples $\{\langle x^{(i)}, y^{(i)} \rangle\}$ relative to the target function f

Output The hypothesis $h \in H$ that best approximates f

But f is unknown ... so?

We try to approximate the training set

Expressiveness of the model

Let $X = X_1 \times X_2 \cdots \times X_n$ where $X_i = \{\text{True,False}\}$

Can we represent $Y = X_2 \wedge X_5$? and $Y = X_4 \vee X_1$?

Can we represent $Y = X_2 \wedge X_5 \vee (\neg X_3) \wedge X_4 \wedge X_1$?

Expressiveness of the model (2)

Important questions

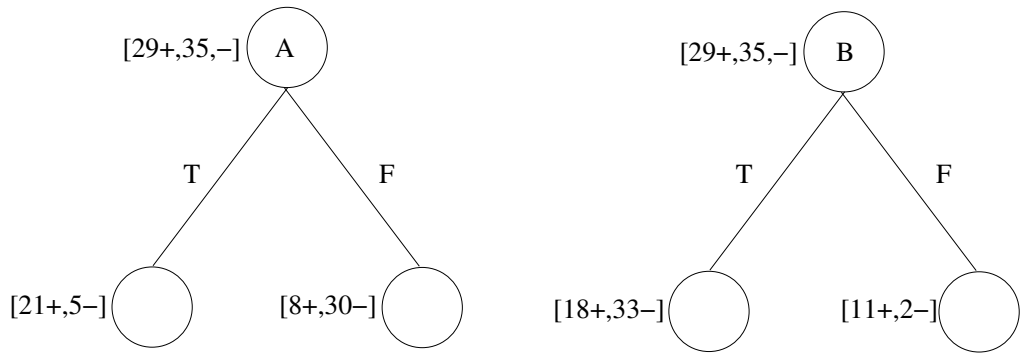
- do we have a decision tree for each h in the space hypothesis?
- if the tree exists, is it unique?
- if it is not unique, do we have a preference?

Top-down inductive construction

Main loop:

1. assign to the current node the "best" attribute X_i
2. create a child node for every possible value of X_i
3. for every child node, if all the examples in the training set associated with the node have a same label y, mark the node (leaf) with label y, otherwise iterate from point 1

What is the "best" attribute?



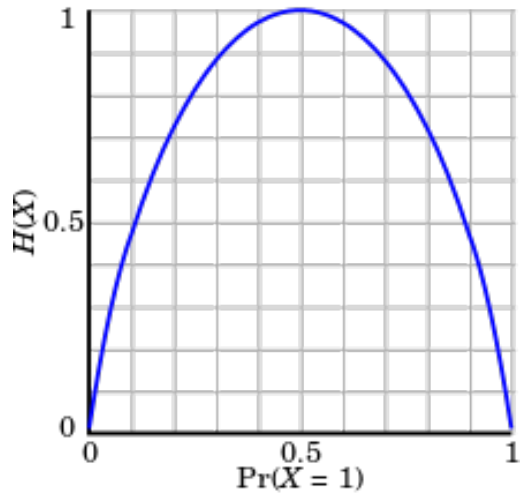
Entropy

The entropy $H(X)$ of a random variable X is

$$H(X) = -\sum_{i=1}^{n} P(X = i) log_2 P(X = i)$$

where n is the number of possible values of X.

Entropy measures the degree of impurity of the information. It is maximal when X is uniformly distributed over all values, and minimal (0) when it is concentrated on a single value.



Information Theory (Shannon)

Entropy is the average amount of information produced by a stochastic source of data.

Information is associated with the *probability* of each data (the "surprise" carried by the event):

- an event with probability 1 carries no information: $I(1) = 0$
- given two independent events with probabilities p_1 and p_2 their joint probability is $p_1 p_2$ but the information acquired is the sum of the informations of the two independent events, so

$$I(p_1 p_2) = I(p_1) + I(p_2)$$

It is hence natural to define

$$I(p) = -log(p)$$

Code Theory (Shannon)

Entropy also measures the average number of bits required to transmit outcomes produced by stochastic process X.

Suppose to have n events with the same probability. How many bits do you need to encode each possible outcome?

Code Theory (Shannon)

Entropy also measures the average number of bits required to transmit outcomes produced by stochastic process X.

Suppose to have n events with the same probability. How many bits do you need to encode each possible outcome?

$$log(n)$$

Code Theory (Shannon)

Entropy also measures the average number of bits required to transmit outcomes produced by stochastic process X.

Suppose to have n events with the same probability. How many bits do you need to encode each possible outcome?

$$log(n)$$

In this case,

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} P(X=i) log_2 P(X=i) \\ &= -\sum_{i=1}^{n} 1/n \, log_2(1/n) \\ &= log(n) \end{aligned}$$

Code Theory (Shannon)

Entropy also measures the average number of bits required to transmit outcomes produced by stochastic process X.

Suppose to have n events with the same probability. How many bits do you need to encode each possible outcome?

$$log(n)$$

In this case,

$$\begin{aligned} H(X) &= -\sum_{i=1}^{n} P(X=i) log_2 P(X=i) \\ &= -\sum_{i=1}^{n} 1/n \, log_2(1/n) \\ &= log(n) \end{aligned}$$

If events are not equiprobable we can do better!!!

Information gain

Entropy of X

$$H(X) = -\sum_{i=1}^{n} P(X = i) log_2 P(X = i)$$

Conditional Entropy of X given a specific $Y = v$

$$H(X|Y = v) = -\sum_{i=1}^{n} P(X = i|Y = v) log_2 P(X = i|Y = v)$$

Conditional Entropy of X given Y
(weighted average over all m possible values of Y)

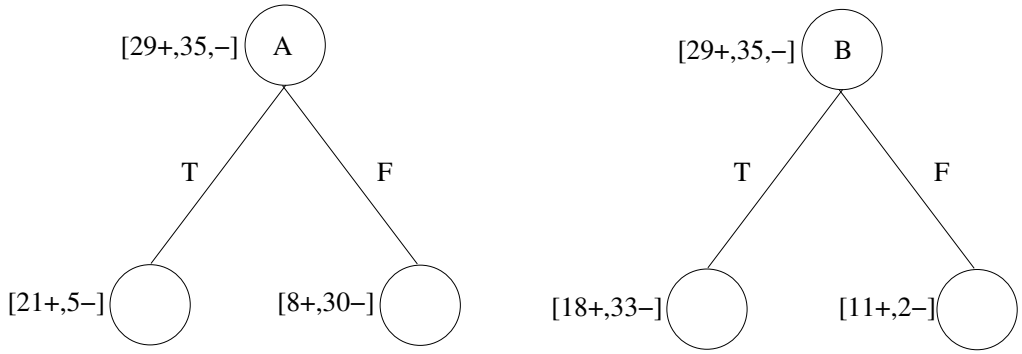
$$H(X|Y) = \sum_{v=1}^{m} P(Y = v) H(X|Y = v)$$

Information Gain between X and Y:

$$I(X, Y) = H(X) - H(X|Y) = H(Y) - H(Y|X)$$

Better A or B?

Let us measure the entropy reduction of the target variable Y due to some attribute X, that is the information gain $I(Y, X)$ between Y and X.



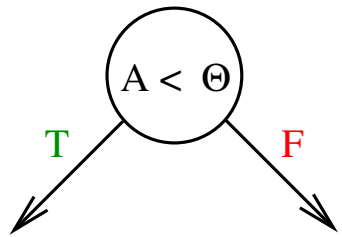
$H(Y) = -(29/64) \cdot log_2(29/64) - (35/64) \cdot log_2(35/64) = .994$
$H(Y|A = T) = -(21/26) \cdot log_2(21/26) - (5/26) \cdot log_2(5/26) = .706$
$H(Y|A = F) = -(8/38) \cdot log_2(8/38) - (30/38) \cdot log_2(30/38) = .742$
$H(Y|A) = .706 \cdot 26/64 + .742 \cdot 38/64 = .726$
$I(Y, A) = H(Y) - H(Y|A) = .994 - .726 = .288$

For B we get $H(Y|B) = .872$ and $I(Y, B) = .122$

So A is better!

The continuous case

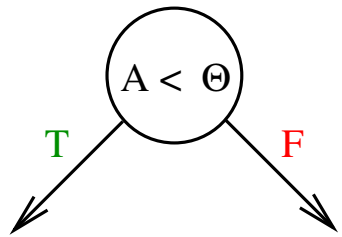
When attributes are continuous we take decisions based on thresholds:



- we compare thresholds with information gain
- how to choose candidate thresholds?

The continuous case

When attributes are continuous we take decisions based on thresholds:

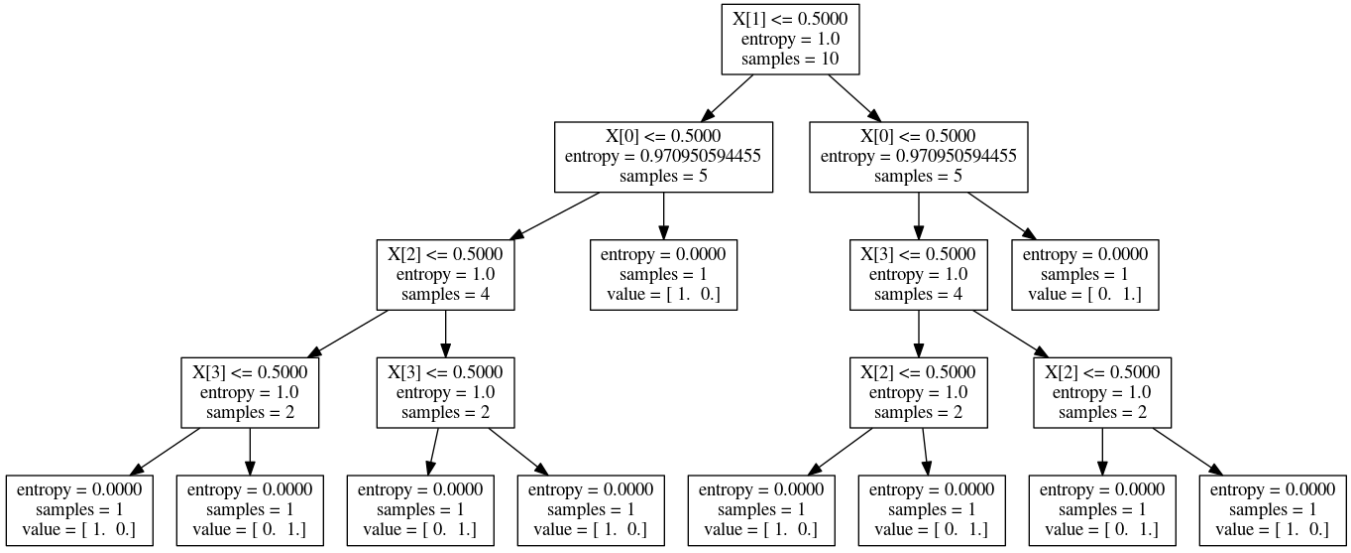


- we compare thresholds with information gain
- how to choose candidate thresholds?
  - sample at discrete intervals
  - order the test set w.r.t. the given attribute and choose thresholds at the average of two consecutive data

An example

```
from sklearn import tree
X = [[0,0,0,0], [1,0,0,0], [0,0,0,1], [0,0,1,0], [0,0,1,1],
     [0,1,0,0], [0,1,0,1], [1,1,0,1], [0,1,1,0], [0,1,1,1]]
Y = [0,0,1,1,0,0,1,1,1,0]
#label is 1 if X[2]==X[3], 0 otherwise

clf = tree.DecisionTreeClassifier(criterion='entropy',
                                  random_state=0)
clf = clf.fit(X, Y)

print(clf.predict([1,1,1,1]))   #> [1]
print(clf.feature_importances_) #> [0.17, 0.029, 0.4, 0.4]

tree.export_graphviz(clf, out_file='tree.dot')
#dot -T png tree.dot -o tree.png
```

Example: the decision tree



In this case, the information gain of individual features is not a good selection policy!

Still, importance is fair, since it is calculated ex post for a single decision tree by the amount that each attribute split point improves the performance measure, weighted by the number of observations the node is responsible for.

Next arguments

- No free lunch
- Inductive Bias

Function approximation problem

Problem Configuration:

- a set of instances X
  every instance $x \in X$ is a vector of features (attributes)
  $$x = \langle x_1, x_2, \ldots x_n \rangle$$
- Target function $f : X \to Y$, where Y takes discrete values
- Hypothesis Space: $H = \{h \mid h : X \to Y\}$

Input (of the training): A training set of $\{\langle x^{(i)}, y^{(i)} \rangle\}$ relative to the target function f

Output (of the training): The hypothesis $h \in H$ that better approximate (?!?) the target function f (not known)

Example: guessing the next element

$$2, 3, 5,$$

Example: guessing the next element

$$2, 3, 5, 9,$$

Example: guessing the next element

$$2, 3, 5, 9, 17, \ldots$$

Example: guessing the next element

$$2, 3, 5, 9, 17, \ldots$$

Not a single solution.
Why one should be better than others?

Example: guessing the next element

$$2, 3, 5, 9, 17, \ldots$$

Not a single solution.
Why one should be better than others?

To learn, we need an inductive bias!

A more complex example:
(just for fun)

1. 1
2. 11
3. 21
4. 1211
5. 111221
6. 312211
7. 13112221
8. ???

Example: guessing the next element

$$2, 3, 5, 9, 17, \ldots$$

Not a single solution.
Why one should be better than others?

To learn, we need an inductive bias!

A more complex example:
(just for fun)

1. 1
2. 11
3. 21
4. 1211
5. 111221
6. 312211
7. 13112221
8. ???

Exercise: what would a decision tree predict?

Learning: search for the best hypothesis

Learning can be understood as a search in the Space Hypothesis for the function that better fits the training set (according to some error measure).

Example: input described by 20 boolean features; boolean output.

- What is the dimension of the Space Hypothesis?
- How many training instances are need to uniquely determine the target function f?
- If we have m training instances, how many distinct functions satisfy them?
- Given a new instance, how many of the previous ones will satisfy it?

No Free Lunch (NFL)

Supposing that all hypothesis have the same probability, there is no reason to prefer one to the other.

Hume

Even after the observation of the frequent or constant conjunction of objects, we have no reason to draw any inference concerning any object beyond those of which we have had experience.

The choice of the hypothesis (or model) must be driven by inductive biases, otherwise no learning is possible.

The learning biases are the set of assumptions that the learner uses to predict outputs for new inputs.

An example: Occam's razor

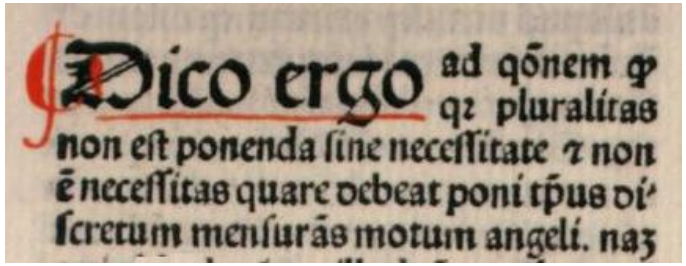
A well known example of inductive bias is the so called Occam's razor:

"*The simplest solution consistent with observation is to be preferred*" (lex parsimoniae).

Never explicitly stated in Occam's works

Similar concepts expressed by Duns Scoto and Thomas Aquinas:

"*Plurality is not to be posited without necessity*"



Occam's razor is e.g. used by decision trees.

Other examples of inductive biases

- **conditional independence** if hypothesis are expressed in a Bayesian framework (see later), we assume the conditional independence of the input random variables. Bias used by Naïve Bayes classifiers.
- **Maximum margin** when drawing a discrimination line between two classes, prefer the one that maximize the width of the margin between elements of the classes. Bias used by Support Vector Machines.
- **Features importance** compare features (e.g. via information gain) and give priority to the most useful ones. Bias used in Decision Tree learning and several algorithms based on feature selection.
- **Nearest neighbours** we assume that all data in a small neighbourhood of a given data belongs to a same class. Bias used in k-nearest neighbour algorithm.

No Free Lunch - discussion

The NFL theorem is based on the assumption that any possible extension of the target function beyond the training samples remains equiprobable.

Learning can only be done on a known training dataset.
We presume the training set is a significant sample of some target distribution we are trying to learn.

- discriminative approaches try to learn the probability of an output given the input, that is the conditional distribution $P(Y|X)$
- generative approaches try to learn the joint input-output distribution $P(X, Y)$

Overfitting

Overfitting

Let us consider the error of the hypothesis h

- on the training set, $error_{train}(h)$
- on the full data set $\mathcal{D}$, $error_{\mathcal{D}}(h)$

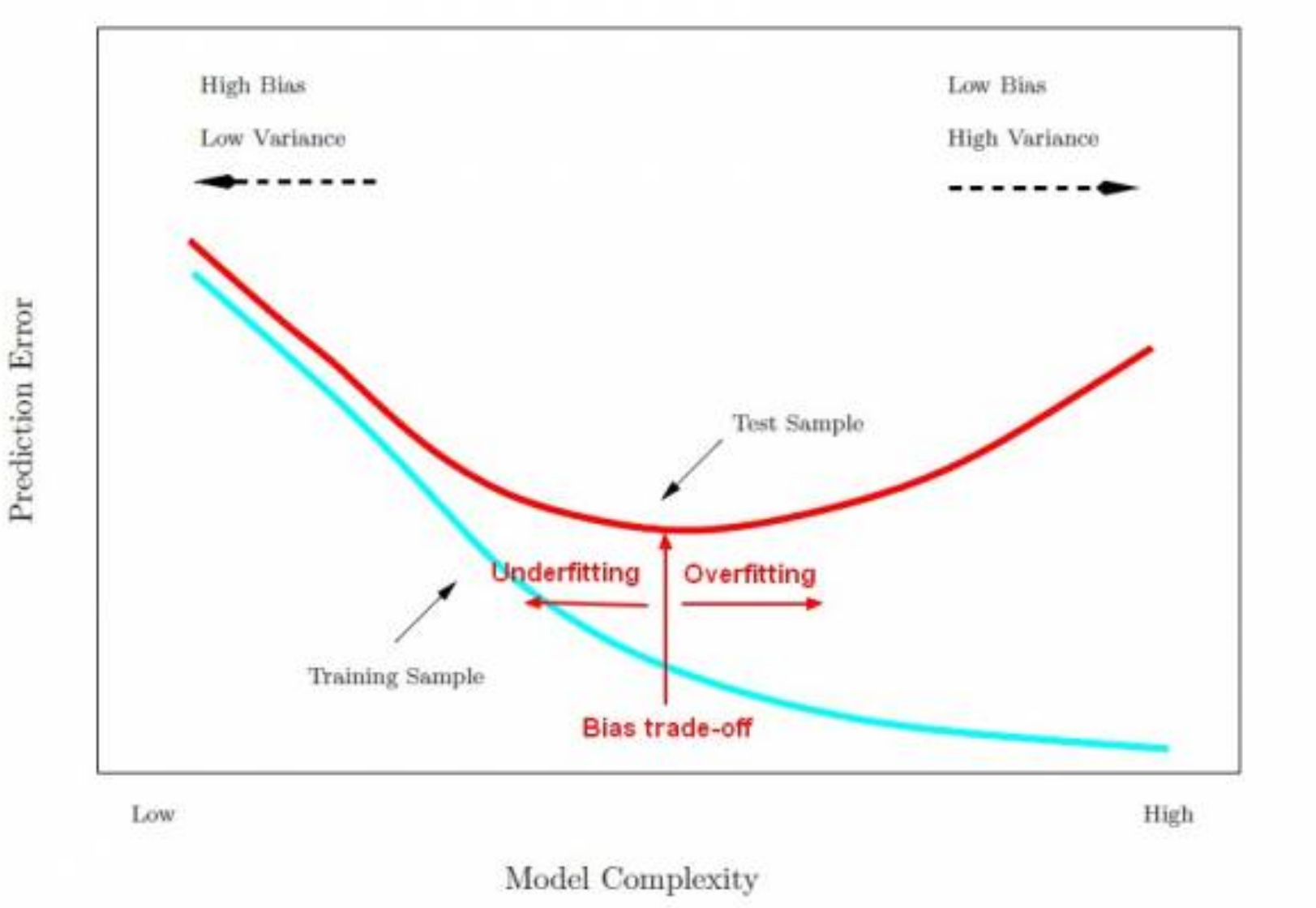
We say that h overfits the training set if there exists another hypothesis h' such that

$$error_{train}(h) < error_{train}(h')$$

but

$$error_{\mathcal{D}}(h) > error_{\mathcal{D}}(h')$$

Overfitting and model complexity



How to check and avoid overfitting

Problem: we do not know $\mathcal{D}$!

Divide the available data in two disjoint sets:

training set to be used to choose a candidate h

validation set to be used to asses the accuracy of h

Avoid overfitting with decision trees

- early stopping: terminate the construction of the tree as soon as the classification improvement is not statistically significative (e.g. the information gain is below some threshold, and/or we do not have sufficient data)
- post-pruning: develop the full tree and then proceed to backward prune it

reduce-error pruning

Build a full and exact decision tree for the training set.

Repeat the following operation until further pruning do not improve accuracy:

1. for any subtree, compute the impact of its removal on the classification accuracy on the validation set
2. greedily perform pruning of the subtree that optimizes accuracy

Variants: Gini's impurity

Gini's impurity measures the probability that a generic element get misclassified according to the current classification (an alternative to entropy).

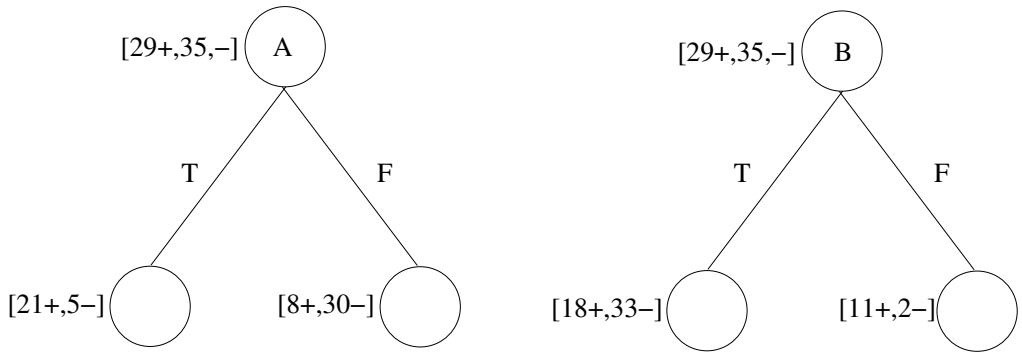
Given m categories, let f_i be the fraction of data with label i. This is equal to the probability that an input belongs to the category i. The probability of misclassifying it is hence $1 - f_i$, and its weighted average on all categories is just Gini's impurity, that is

$$I_G(f) = \sum_{i=1}^{m} f_i(1 - f_i) = \sum_{i=1}^{m} (f_i - f_i^2) = \sum_{i=1}^{m} f_i - \sum_{i=1}^{m} f_i^2 = 1 - \sum_{i=1}^{m} f_i^2$$

This metric is applied to every child node, and values are summed in a weighted way (similarly to the definition of Information-gain) to get a measure of the quality of an attribute.

Better A or B?

Let us evaluate the split using Gini's impurity.



For the attribute A:
$I_G(A = T) = 1 - (21/26)^2 - (5/26)^2 = .310$
$I_G(A = F) = 1 - (8/38)^2 - (30/38)^2 = .332$
$I_G(A) = .310 \cdot 26/64 + .332 \cdot 38/64 = .323$

For the attribute B:
$I_G(B = T) = 1 - (18/51)^2 - (33/51)^2 = .456$
$I_G(B = F) = 1 - (11/13)^2 - (2/13)^2 = .260$
$I_G(B) = .456 \cdot 51/64 + .260 \cdot 13/64 = .416$

Hence, A is better (lower impurity)

Positive aspects of decision trees

- easy to understand: simple logical rules, trees can be visualized
- little or no data preprocessing is required
- very low prediction cost
- can be used with both discrete and continuous features

Negative aspects of decision trees

- high risk of overfitting
- selection of attributes quite unstable
- easy to build strongly unbalanced trees, especially if a class is dominant It can be useful to pre-balance the dataset.

Decision Trees on Scikit-learn

Look at the following page

```
>>> from skelearn import tree
>>> X = [[0,0],[1,1]]
>>> Y [[0,1]]
>>> clf = tree.DecisionTreeClassifier()
>>> clf = clf.fit(X,Y)
```

- any method in the first part of course can be tested that easily
- you are **warmly** invited to play with them
- do not be too naif in your project

Random Forests

Decision Trees are normally used as components in Random Forests, operating as an ensemble.

Ensemble techniques exploits the principle that a large number of relatively uncorrelated models (e.g. trees) operating as a committee will typically outperform any of the individual constituent models.

Ensure differentiation by:

- Bagging: feeding different, randomly choosen sets of input data
- Feature Randomness: building trees from random subsets of features

Summary of main notions learned

- **Approximation problem**:
  - Features X, labels (categories) Y
  - Training set $\{\langle x^{(i)}, y^{(i)} \rangle\}$
  - Hypothesis Space $H = \{h \mid h : X \to Y\}$

- **Learning = search/optimization in H**
  - Different objectives are possible
    - minimize the error on the training set (0-1 loss)
    - among candidate solutions look for the "simpler" one (occam razor's)
    - ...
  - No learning is possible without an inductive bias (NFL theorem)

Probability basics

Probability basics

- **Events**
  - discrete and continuous random variable
- **Axioms of Probability Theory**
  - a reasonable theory of uncertainty
- **Independent events**
- **Conditional Probability**
- **Bayes Rule**
- **Joint Probability Distribution**
- **Expectation**
- **Independence, conditional Independence**

Random variable

A random variable X denotes an outcome about which we are uncertain, for instance the result of a randomized experiment

Examples

- $X = true$ if a (randomly drawn) student in this room is male
- $X =$ first name of the student
- $X = true$ if two randomly drawn students in this room have the same birthday

We define $P(X)$ (probability of X) as the fraction of times X is true, in repeated runs of the same experiment.

More formally

- the set Ω of possible outcomes of a random experiment is called sample space
  - e.g. the set of students in this room
- a random variable is a measurable function over Ω:
  - discrete: e.g. gender: $\Omega \rightarrow \{m, f\}$
  - continuous: e.g. height: $\Omega \rightarrow \mathcal{R}$
- an event is an arbitrary subset of Ω
  - $\{x \in \Omega | gender(x) = m\}$
  - $\{x \in \Omega | height(x) \leq 175cm\}$
- we are interested in probabilities of specific events
  - $P(gender = m)$
- and in their probabilities conditioned on other events
  - $P(gender = m | height \leq 175)$

Sample space

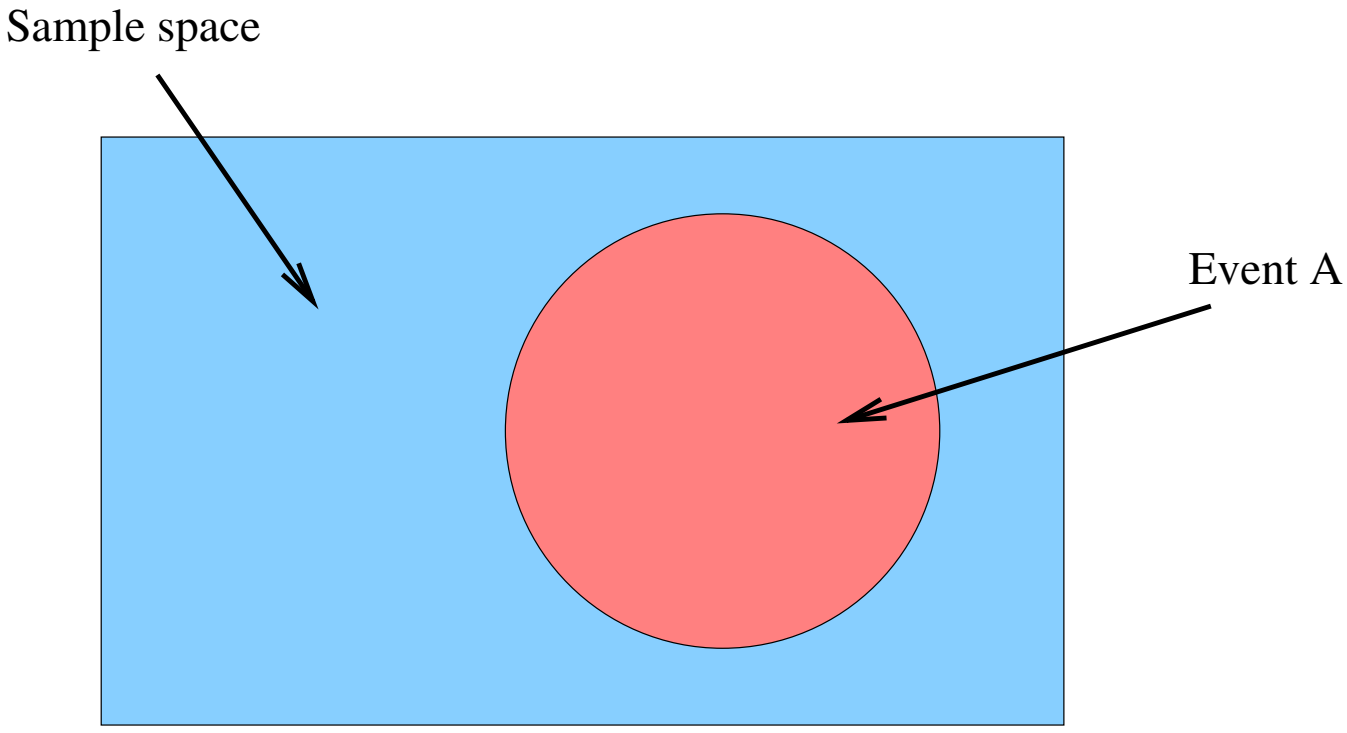
The definition of the sample space requires some caution.

Our probability intuition relies on the assumption that all samples in the space have the same probability.

Example: toss of a pair of dices.
The outcome is an integer number between 2 and 12.
However, not all of them have the same probability.

Better to consider the pairs of outcomes of each dice as elements of the sample space, and a random variable X expressing their sum.

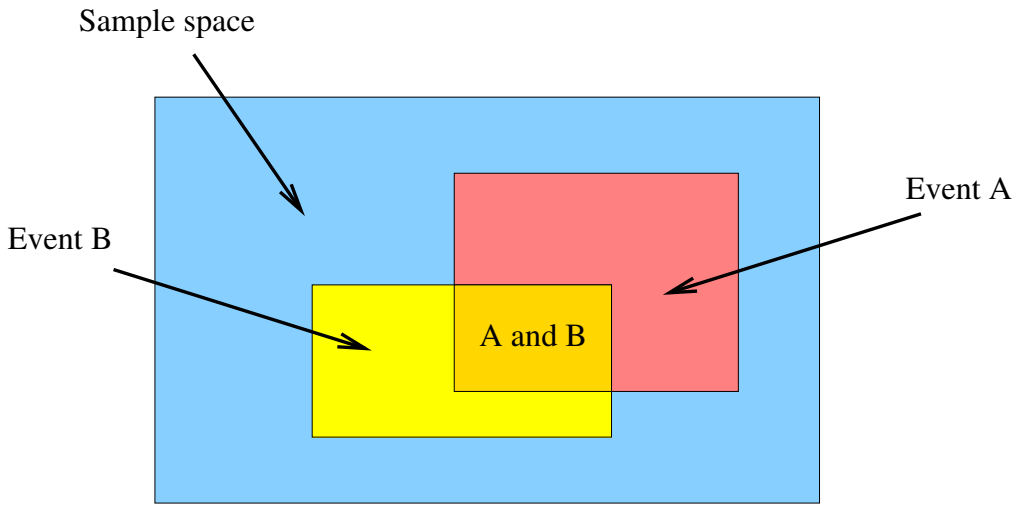
Graphical Visualization



The probability $P(A)$ of the event A is the ratio between the area of A and the area of the full sample space.

Probability Axioms

- $0 <= P(A) <= 1$
- $P(True) = 1$
- $P(False) = 0$
- $P(A \vee B) = P(A) + P(B) - P(A \wedge B)$



Derived theorems

$$P(\neg A) = 1 - P(A)$$

Proof: We know that

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

and in particular

$$P(A \vee \neg A) = P(A) + P(\neg A) - P(A \wedge \neg A)$$

but

$$P(A \vee \neg A) = P(\mathit{True}) = 1 \quad \mathit{and} \quad P(A \wedge \neg A) = P(\mathit{False}) = 0$$

so

$$1 = P(A) + P(\neg A) - 0$$

q.e.d.

Another interesting theorem

$$P(A) = P(A \wedge B) + P(A \wedge \neg B)$$

Proof: We know that

$$P(A \vee B) = P(A) + P(B) - P(A \wedge B)$$

Since

$$A = A \wedge (B \vee \neg B) = (A \wedge B) \vee (A \wedge \neg B)$$

we have:

$$\begin{aligned} P(A) = & \quad P(A \wedge B) + P(A \wedge \neg B) - P((A \wedge B) \wedge (A \wedge \neg B)) \\ = & \quad P(A \wedge B) + P(A \wedge \neg B) - P(\mathit{False}) \\ = & \quad P(A \wedge B) + P(A \wedge \neg B) \end{aligned}$$

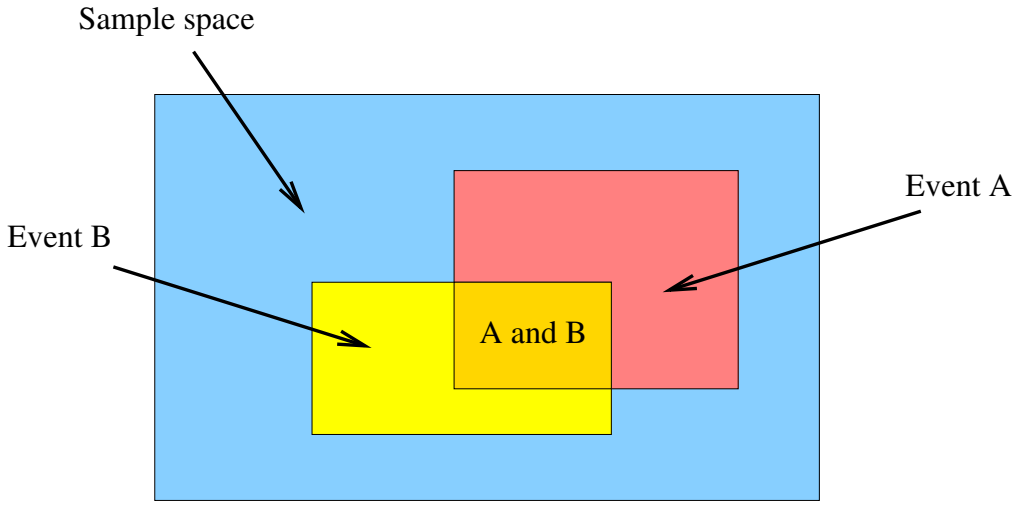
Multivalued discrete random variables

A is a k-valued discrete random variable if it may take exactly one of the values in the set $\{\nu_1, \nu_2, \ldots \nu_k\}$.

$P(A = \nu_i)$ is the event of all samples where A takes value ν_i.

$P(A = \nu_i \wedge A = \nu_j) = 0$ if $i \neq j$
$P(A = \nu_1 \vee A = \nu_2 \vee \ldots A = \nu_k) = 1$

Conditional Probability



Definition The conditional probability of the event A given the event B is defined as the quantity

$$P(A|B) = \frac{P(A \wedge B)}{P(B)}$$

Corollary: Chain rule

$$P(A \wedge B) = P(B) \cdot P(A|B) = P(A) \cdot P(B|A)$$

Independent events

Two events A and B are independent when

$$P(A|B) = P(A)$$

That is, the event B has no influence over A.
As a corollary,

$$P(A \wedge B) = P(A) \cdot P(B)$$

Moreover, since

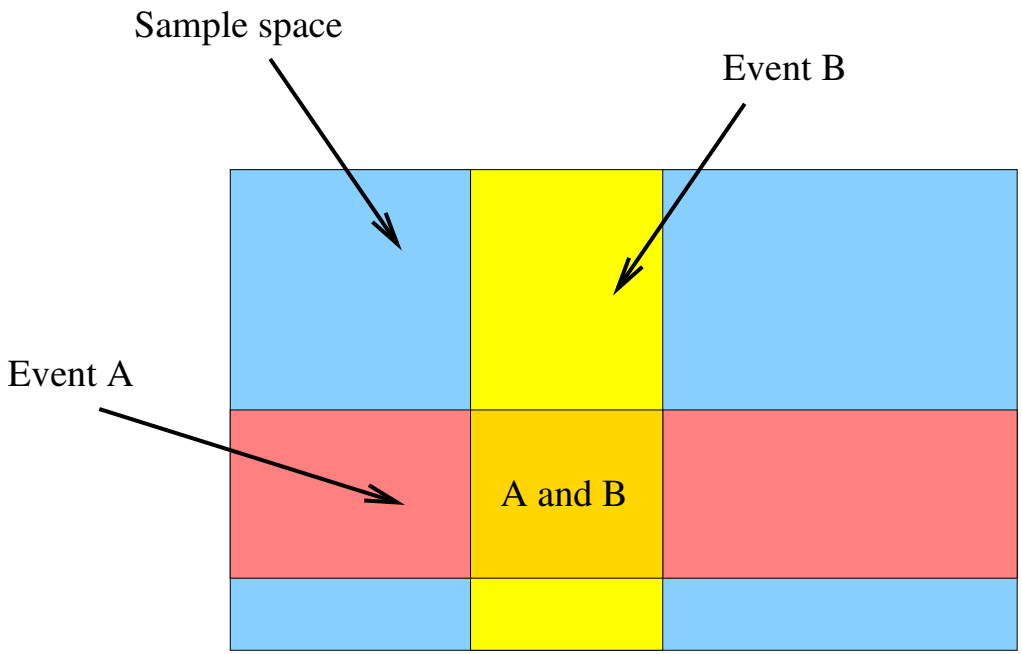
$$P(A \wedge B) = P(B|A) \cdot P(A)$$

we also have

$$P(B|A) = P(B)$$

Graphical intuition

Two “orthogonal” events:



Bayes' rule

$$P(A|B) = \frac{P(A) \cdot P(B|A)}{P(B)}$$

Example

- in a school 60% of students are boys and 40% are girls.
- girls wear in the same number skirts and trousers
- boys only wear trousers

If we see a student wearing trousers, what is the probability that is a girl?

Solution

The probability a priori that a student is a girl is

$$P(G) = 2/5$$

the probability that a student wears trousers is

$$P(T) = 1/5 + 3/5 = 4/5$$

the probability that a student wear trousers, given that the student is a girl, is

$$P(T|G) = 1/2$$

Hence, the probability that a student wearing trousers is a girl is

$$P(G|T) = \frac{P(G) \cdot P(T|G)}{P(T)} = \frac{2/5 \cdot 1/2}{4/5} = 1/4$$

Another formulation of Bayes rule

$$P(Y|X) = \frac{P(Y) \cdot P(X|Y)}{P(X)}$$

or more precisely, for every i, j,

$$P(Y = y_i|X = x_j) = \frac{P(Y = y_i) \cdot P(X = x_j|Y = y_i)}{P(X = x_j)}$$

But we know that

$$P(X = x_j) = \sum_i P(X = x_j, Y = y_i) = \sum_i P(Y = y_i) \cdot P(X = x_j|Y = y_i)$$

so

$$P(Y = y_i|X = x_j) = \frac{P(Y = y_i) \cdot P(X = x_j|Y = y_i)}{\sum_i P(Y = y_i) \cdot P(X = x_j|Y = y_i)}$$

Posterior, likelihood, prior and marginal

$$\underbrace{P(Y|X)}_{posterior} = \frac{\overbrace{P(X|Y)}^{likelihood} \cdot \overbrace{P(Y)}^{prior}}{\underbrace{P(X)}_{marginal\ likelihood}} = \frac{\overbrace{P(X|Y)}^{likelihood} \cdot \overbrace{P(Y)}^{prior}}{\underbrace{\sum_{Y} P(X|Y) \cdot P(Y)}_{marginal\ likelihood}}$$

"Marginal" because we marginalized (i.e. integrated) over Y

Next arguments

- The Joint Distribution
- Bayes classifiers
- Naïve Bayes

The Joint distribution

1. build a table with all possible combinations of values of random variables (features)

2. compute the probability for any different combination of values

gender	working hours	health	prob (params)
F	≤ 40	poor	0.25
F	≤ 40	rich	0.03
F	> 40	poor	0.04
F	> 40	rich	0.01
M	≤ 40	poor	0.33
M	≤ 40	rich	0.10
M	> 40	poor	0.13
M	> 40	rich	0.11

Given n (boolean) features, we need to compute $2^n - 1$ parameters.

Use of the joint distribution

Having the joint distribution we may compute the probability of any event expressible as a logical combination of the features

$$P(E) = \sum_{row \in E} P(row)$$

Example

Let us compute the probability $P(M, poor)$

gender	w. hours	wealth	prob.
F	≤ 40	poor	0.25
F	≤ 40	rich	0.03
F	> 40	poor	0.04
F	> 40	rich	0.01
M	≤ 40	poor	0.33
M	≤ 40	rich	0.10
M	> 40	poor	0.13
M	> 40	rich	0.11

$P(M, poor) = 0.33 + 0.13 = 0.46$

Inference with the Joint distribution

It is also easy to compute the probability of an event E_1 given another event E_2

$$P(E_1|E_2) = \frac{P(E_1 \wedge E_2)}{P(E_2)} = \frac{\sum_{row \in E_1 \wedge E_2} P(row)}{\sum_{row \in E_2} P(row)}$$

Example

$$P(M|poor) = \frac{P(M, poor)}{P(poor)}$$

We know that $P(M, poor) = 0.46$. Let us compute $P(poor)$:

gender	w. hours.	wealth	prob.
F	≤ 40	poor	0.25
F	≤ 40	rich	0.03
F	> 40	poor	0.04
F	> 40	rich	0.01
M	≤ 40	poor	0.33
M	≤ 40	rich	0.10
M	> 40	poor	0.13
M	> 40	rich	0.11

$P(poor) = .75$ and $P(M|poor) = 0.46/0.75 = 0.61$

Conditional Probability vs. learning

Instead of computing

$$f : X \rightarrow Y$$

we may compute the probability

$$p : P(Y|X)$$

We know that we can use the joint distribution, so we just need to compute it.

End of the story?

Complexity issues

Let us build the joint table relative to

$$P(Y = wealth | X_1 = gender, X_2 = orelav.)$$

X_1=gender	X_2=ore lav.	$P(rich \mid X_1, X_2)$	$P(poor \mid X_1, X_2)$
F	≤ 40	.09	.91
F	> 40	.21	.79
M	≤ 40	.23	.77
M	> 40	.38	.62

To fill the table we need to compute $4 = 2^2$ parameters.

If we have n random variable $X = X_1 \times X_2, \ldots X_n$ where each X_i is boolean, we need to compute 2^n parameters.

These parameters are probabilities: to get reasonable value we would need a huge amount of data.

Can we use Bayes rule?

We know that

$$P(Y = y_i|X = x_j) = \frac{P(Y = y_i) \cdot P(X = x_j|Y = y_i)}{\sum_i P(Y = y_i) \cdot P(X = x_j|Y = y_i)}$$

So, to compute $P(Y = y_i|X = x_j)$ it is enough to compute

$$P(Y) \quad \text{and} \quad P(X_1, X_2, \ldots X_n|Y)$$

- how many parameters for $P(Y)$?
- how many parameters for $P(X_1, X_2, \ldots X_n|Y)$?

Naïve Bayes

Naïve Bayes assumes that

$$P(X_1, X_2, \ldots X_n|Y) = \prod_i P(X_i|Y)$$

that is, given Y, X_i and X_j are independent from each other.

Conditional independence

Two events X_i and X_j are independent given Y if

$$P(X_i|X_j, Y) = P(X_i|Y)$$

Example 1 A box contains two coins: a regular coin and a fake two-headed coin (P(H)=1). Choose a coin at random, toss it twice and consider the following events:

A = First coin toss is H
B = Second coin toss is H
C = First coin is regular

A and B are NOT independent (prove it), but they are conditionally independent given C.

Example 2 For individuals, height and vocabulary are not independent, but they are if age is given.

Conditional independence in Naïve Bayes

$$\begin{aligned} P(X_1, X_2|Y) &= P(X_1|X_2, Y) \cdot P(X_2|Y) && \text{by the chain rule} \\ &= P(X_1|Y) \cdot P(X_2|Y) && \text{by cond. ind.} \end{aligned}$$

In general,

$$P(X_1, X_2, \ldots X_n|Y) = \prod_i P(X_i|Y)$$

How many parameters to describe $P(X_1, X_2, \ldots X_n|Y)$ (in the boolean case)?

- without conditional Independence
- in Naïve Bayes

Naïve Bayes in a nutshell

Bayes rule

$$P(Y = y_i | X_1, \ldots X_n) = \frac{P(Y = y_i) \cdot P(X_1, \ldots X_n | Y = y_i)}{P(X_1, \ldots X_n)}$$

Naïve Bayes

$$P(Y = y_i | X_1, \ldots X_n) = \frac{P(Y = y_i) \cdot \prod_j P(X_j | Y = y_i)}{P(X_1, \ldots X_n)}$$

Classification of a new sample $x^{new} = \langle x_1, \ldots x_n \rangle$

$$Y^{new} = arg\ max_{y_i} P(Y = y_i) \cdot \prod_j P(X_j = x_j | Y = y_i)$$

Learning algorithm

Discrete Random variables X_i, Y

- **Training**
  - for any possible value y_k of Y, estimate

$$\pi_k = P(Y = y_k)$$

  - for any possible value x_{ij} of X_i estimate

$$\theta_{ijk} = P(X_i = x_{ij} | Y = y_k)$$

- **Classification of** $a^{new} = \langle a_1, \ldots a_n \rangle$

$$\begin{aligned} Y^{new} &= arg\ max_{y_k}\ P(Y = y_k) \cdot \prod_i P(X_i = a_i | Y = y_k) \\ &= arg\ max_k\ \pi_k \cdot \prod_i \theta_{ijk} \end{aligned}$$

supposing that $a_i = x_{ij}$ (i.e. a_i is the j-th among the discrete values of the attribute X_i)

Estimating parameters

Maximum likelihood estimates (MLE's):

$$\pi_k = P(Y = y_k) = \frac{\#\mathcal{D}\{Y = y_k\}}{|\mathcal{D}|}$$

$$\theta_{ijk} = P(X = x_{ij}|Y = y_k) = \frac{\#\mathcal{D}\{X_i = x_{i,j} \wedge Y = y_k\}}{\#\mathcal{D}\{Y = y_k\}}$$

Example: a good day to play tennis?

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Rain	Cool	Normal	Strong	No
Overcast	Cool	Normal	Strong	Yes
Sunny	Mild	High	Weak	No
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes
Rain	Mild	High	Strong	No

Priors

$\pi_{Yes} = 9/14 = .64$

$\pi_{No} = 5/14 = .36$

Computing θ (Yes)

Outlook	Temp	Humidity	Wind	Play
Overcast	Hot	High	Weak	Yes
Rain	Mild	High	Weak	Yes
Rain	Cool	Normal	Weak	Yes
Overcast	Cool	Normal	Strong	Yes
Sunny	Cool	Normal	Weak	Yes
Rain	Mild	Normal	Weak	Yes
Sunny	Mild	Normal	Strong	Yes
Overcast	Mild	High	Strong	Yes
Overcast	Hot	Normal	Weak	Yes

Outlook

$\theta_{Sunny,Yes} = 2/9$

$\theta_{Overc.,Yes} = 4/9$

$\theta_{Rain,Yes} = 3/9$

Temp

$\theta_{Hot,Yes} = 2/9$

$\theta_{Mild,Yes} = 4/9$

$\theta_{Cool,Yes} = 3/9$

Humidity

$\theta_{High,Yes} = 3/9$

$\theta_{Normal,Yes} = 6/9$

Wind

$\theta_{Weak,Yes} = 6/9$

$\theta_{Strong,Yes} = 3/9$

Computing θ (No)

Outlook	Temp	Humidity	Wind	Play
Sunny	Hot	High	Weak	No
Sunny	Hot	High	Strong	No
Rain	Cool	Normal	Strong	No
Sunny	Mild	High	Weak	No
Rain	Mild	High	Strong	No

Outlook

$\theta_{Sunny,No} = 3/5$

$\theta_{Overc.,No} = 0$

$\theta_{Rain,No} = 2/5$

Temp

$\theta_{Hot,No} = 2/5$

$\theta_{Mild,No} = 2/5$

$\theta_{Cool,No} = 1/5$

Humidity

$\theta_{High,No} = 4/5$

$\theta_{Normal,No} = 1/5$

Wind

$\theta_{Weak,No} = 2/5$

$\theta_{Strong,No} = 3/5$

Prediction

New instance:

Outlook=Sunny,Temp.=Cool,Humidity=High,Wind=Strong

We need to compute

$$arg\ max_{y \in yes,no} p(y) \cdot p(sunny|y) \cdot p(cool|y) \cdot p(high|y) \cdot (strong|y)$$

that is, we need to compare

$$\pi_{yes} \cdot \theta_{sunny,yes} \cdot \theta_{cool,yes} \cdot \theta_{high,yes} \cdot \theta_{strong,yes}$$
$$= 9/14 \cdot 2/9 \cdot 3/9 \cdot 3/9 \cdot 3/9 = .0053$$

$$\pi_{no} \cdot \theta_{sunny,no} \cdot \theta_{cool,no} \cdot \theta_{high,no} \cdot \theta_{strong,no}$$
$$= 5/14 \cdot 3/5 \cdot 1/5 \cdot 4/5 \cdot 3/5 = .0205$$

NO!

Next arguments

- Generative techniques
- Limitations and cautions

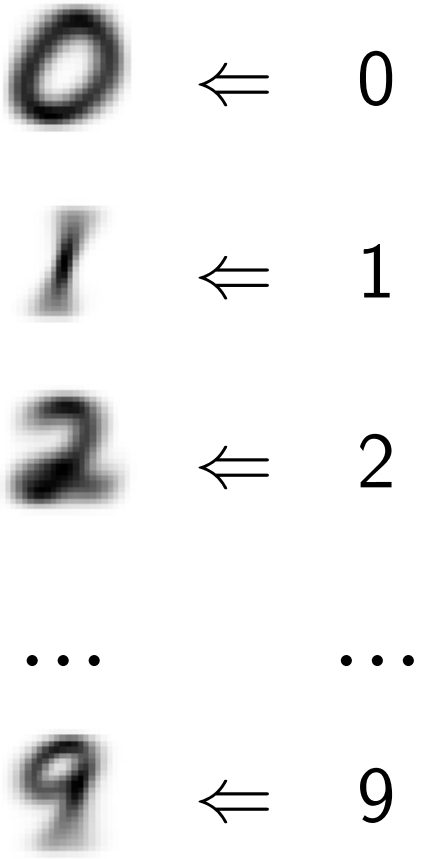
The generative nature of Naive Bayes

We are interested in computing

$$P(Y = y_i | X_1, \ldots X_n)$$

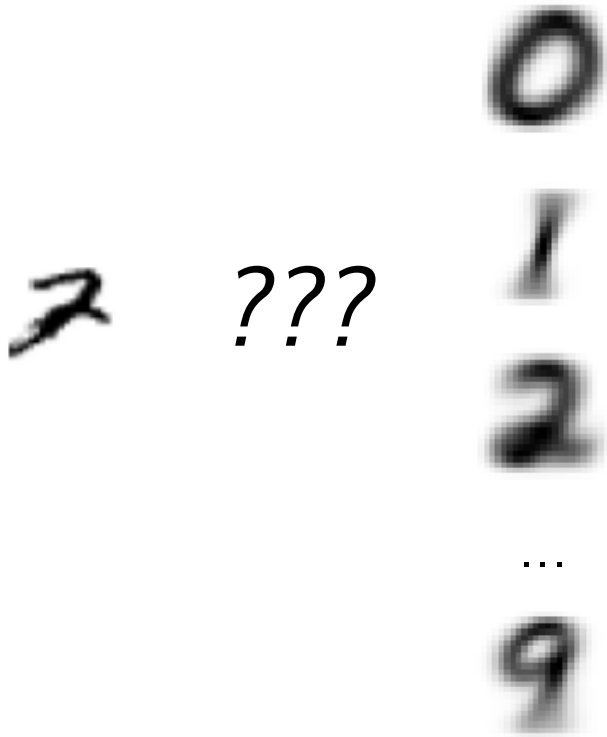
Bayes' rule allow to reverse the problem, asking instead to understand the **distribution of data, given the category**

$$P(X_1, \ldots X_n | Y = y_i)$$



Classification of new data

We classify a new data point by asking to which one of the input distributions it is the most likely to belong to:



Joint vs. Naïve

The distributions we would be interested in are the joint distributions

$$P(X_1, \ldots X_n | Y = y_i)$$

In the case of mnist, the distributions of images, reparted by categories, in a space with 784 dimensions.
Since this is unfeasible, we use the naive approach to assume that all features are independent, reducing the problem to the estimation of

$$P(X_j | Y = y_i)$$

for all possible feautures $j = 1, \ldots, n$.

Cautions (1)

In some cases, MLE for $P(X_i|Y)$ may be 0

e.g. $P(Play = No|Outlook = Overcast)$

- Why should be worry?
- How can we avoid the problem?

Cautions (2)

Naïve Bayes assumes events are independent from each other (given Y).

What if this is not the case?

Limitations

Suppose we have a set of random images, with pixels either 0 or 1.

Choose two pixels p_1 and p_2. We want to classify the image in category A if $p_1 == p_2$ and in category B otherwise.

It looks like a pretty simple classification. Let us try to use Naive Bayes.

What is $P(p_1 = 1|A)$?

What is $P(p_1 = 1|B)$?

What is $P(p_2 = 1|A)$?

What is $P(p_2 = 1|A)$?

So? ...

About
Maximum Likelihood Estimation

Back to parameters estimation

Maximum likelihood estimates (MLE's):

$$\pi_k = P(Y = y_k) = \frac{\#\mathcal{D}\{Y = y_k\}}{|\mathcal{D}|}$$

$$\theta_{ijk} = P(X_i = x_{ij}|Y = y_k) = \frac{\#\mathcal{D}\{X_i = x_{i,j} \wedge Y = y_k\}}{\#\mathcal{D}\{Y = y_k\}}$$

Back to parameters estimation

Maximum likelihood estimates (MLE's):

$$\pi_k = P(Y = y_k) = \frac{\#\mathcal{D}\{Y = y_k\}}{|\mathcal{D}|}$$

$$\theta_{ijk} = P(X_i = x_{ij}|Y = y_k) = \frac{\#\mathcal{D}\{X_i = x_{i,j} \wedge Y = y_k\}}{\#\mathcal{D}\{Y = y_k\}}$$

Why ?

Example

Suppose to toss a (possibly fake) coin 10 times.
We get 6 Heads (H) and 4 Tails (T).

One could "naturally" conclude that $P(H) = .6$ and $P(T) = .4$.

Example

Suppose to toss a (possibly fake) coin 10 times.
We get 6 Heads (H) and 4 Tails (T).

One could "naturally" conclude that $P(H) = .6$ and $P(T) = .4$.

Suppose that there are just two possibilities:

- the coin is fair, that is $P(H) = .5$ and $P(T) = .5$
- the coin is fake, with probabilities $P(H) = .7$ and $P(T) = .3$

Which is the most likely according to our observations, and why?

Independent ed equiprobable events

Bernoulli's Distribution (ex. flipping a coin)

two possible outcomes 0 and 1 with probabilities θ and $1 - \theta$.

Let X^n be the number of 0 in a sequence of n flips
X^n follows a binomial distribution

$$P(X^n = \alpha_0|\theta) = \binom{n}{\alpha_0} \cdot \theta^{\alpha_0} \cdot (1-\theta)^{\alpha_1}$$

where $\alpha_1 = n - \alpha_0$ is the number of 1 in the sequence (id est, $\alpha_0 + \alpha_1 = n$)

Back to our example

$n = 10$, $\alpha_0 = 6$ and $\alpha_1 = 4$.

If $\Theta = .5$ and $1 - \Theta = .5$

$$P(X^n = 6|\theta) = \binom{n}{\alpha_0} \cdot \theta^{\alpha_0} \cdot (1-\theta)^{\alpha_1} = \binom{10}{6} \cdot \left(\frac{1}{2}\right)^6 \cdot \left(\frac{1}{2}\right)^4 = .205$$

If $\Theta = .7$ and $1 - \Theta = .3$

$$P(X^n = 6|\theta) = \binom{n}{\alpha_0} \cdot \theta^{\alpha_0} \cdot (1-\theta)^{\alpha_1} = \binom{10}{6} \cdot \left(\frac{7}{10}\right)^6 \cdot \left(\frac{3}{10}\right)^4 = .200$$

So, $\Theta = .5$ is slightly more likely than $\Theta = .7$.

What is the most likely value for Θ?

Computing MLE

$$\hat{\theta} = arg\ max_\theta\ P(X^n = \alpha_0|\theta) =$$
$$= arg\ max_\theta\ \theta^{\alpha_0} \cdot (1-\theta)^{\alpha_1}$$

Equivalently we may look for θ maximizing the logarithm of the previous expression

$$ln(\theta^{\alpha_0} \cdot (1-\theta)^{\alpha_1}) = \alpha_0 ln(\theta) + \alpha_1 ln(1-\theta)$$

Deriving with respect to θ we get

$$\frac{\alpha_0}{\theta} - \frac{\alpha_1}{1-\theta} = \frac{\alpha_0 - \alpha_0\theta - \alpha_1\theta}{\theta \cdot (1-\theta)}$$

That is zero for

$$\theta = \frac{\alpha_0}{\alpha_0 + \alpha_1} = \frac{\alpha_0}{n}$$

Multivalued case

Discrete Distribution (ex. tossing a dice)
k possible outcomes $\{1, 2, \ldots, k\}$ with probabilities θ_i where $\sum_i \theta_i = 1$.

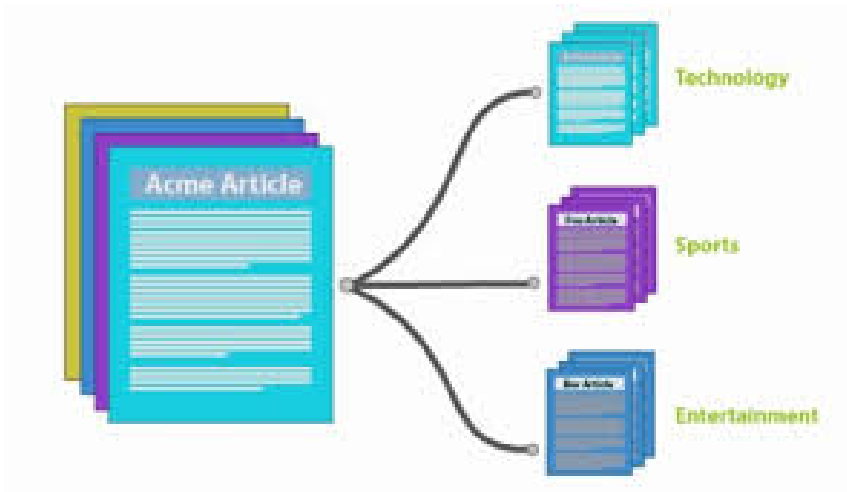
Sequence of n tosses: multinomial distribution

$$P(X^n = \overline{\alpha_i}|\theta) = c_{\overline{\alpha_i}} \prod_i \theta_i^{\alpha_i}$$

where α_i is the number of i in the sequence and $c_{\overline{\alpha_i}}$ is a combinatorial constant not depending on θ

$$\textbf{MLE}: \quad \theta_i = \frac{\alpha_i}{\sum_i \alpha_i} = \frac{\alpha_i}{n}$$

Document classification (bag of words approach)



Document classification

event X_i = i-th word in the document: a discrete random variable assuming as many values as words in the language

$$\theta_{i,word,\ell} = P(X_i = word | Y = \ell)$$

probability that in a document of the category "ℓ" the word "word" appears at position "i"

we assume that all event are independents (?) and have a distribution independent from the position (?)

$$\theta_{i,word,\ell} = \theta_{j,word,\ell} = \theta_{word,\ell}$$

Training and classification

Discrete random variables X_i, Y

- **Training**
  - for any possible value y_k of Y, estimate

$$\pi_k = P(Y = y_k)$$

  - for any possible value x_{ij} of the attribute X_i estimate

$$\theta_{ijk} = P(X_i = x_{ij} | Y = y_k)$$

- **Classification of** $a^{new} = \langle a_1, \ldots a_n \rangle$ (sequence of words)

$$\begin{aligned} Y^{new} &= arg\ max_{y_k}\ P(Y = y_k) \cdot \prod_i P(X_i = a_i | Y = y_k) \\ &= arg\ max_k\ \pi_k \cdot \prod_i \theta_{ijk} \end{aligned}$$

where $x_{ij} = a_i$

Estimating parameters

Maximum likelihood estimates (MLE's):

- $\pi_k = P(Y = y_k)$
  fraction of the documents in category y_k
- $\theta_{i,word,k} = \theta_{word,k} = P(X = word | Y = y_k)$
  frequency of the word "word" in documents of the category y_k

Log likelihood

Instead of

$$\begin{aligned} Y^{new} &= arg\ max_{y_k}\ P(Y = y_k) \cdot \prod_i P(X_i = w_j | Y = y_k) \\ &= arg\ max_k\ \pi_i \cdot \prod_i \theta_{ijk} \end{aligned}$$

We may compute

$$\begin{aligned} Y^{new} &= arg\ max_{y_k}\ log(P(Y = y_k) \cdot \prod_i P(X_i = w_j | Y = y_k)) \\ &= arg\ max_k\ log(\pi_k) + \sum_i log(\theta_{ijk}) \end{aligned}$$

Moreover, if $\theta_{ijk} = \theta_{i'jk} = \theta_{jk}$

$$\sum_i log(\theta_{ijk}) = \sum_j n_j \cdot log(\theta_{jk})$$

where n_j is the number of occurrences of the word w_j in the document to be classified.

Cosine similarity

Consider vectors where each word is a different dimension.

Training

For any category k build a "spectral" vector

$$s_k = \langle log(\theta_{jk}) \rangle_{j \in words}$$

θ_{jk} = frequency of the word j in documents of the category k.

Given a new document, compute a vector

$$d = \langle n_j \rangle_{j \in words}$$

and classify the document as

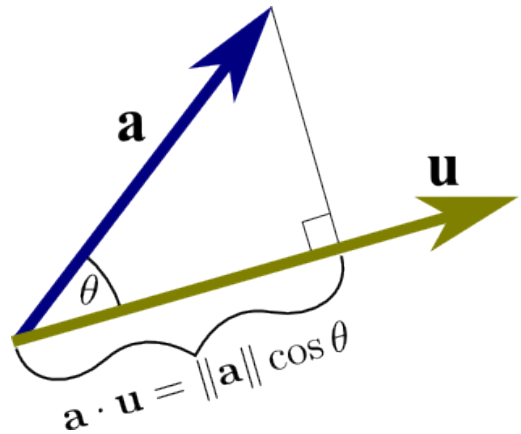
$$arg\ max_k \quad \underbrace{d \cdot s_k}_{\substack{cosine \\ similarity}} = \sum_j d_j \cdot s_{jk}$$

Dot product, geometrically and analytically

Geometric definition

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}|cos(\theta)$$

where θ is the angle between the two vectors



Analytic definition

given $\mathbf{a} = (a_1, a_2, \ldots, a_n)$
and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$

$$\mathbf{a} \cdot \mathbf{b} = \sum_{i=1}^{n} a_i b_i$$

Equivalence

By the cosine rule (a generalization of Pythagoras' theorem),

$$|\mathbf{a} - \mathbf{b}|^2 = |\mathbf{a}|^2 + |\mathbf{b}|^2 - 2|\mathbf{a}||\mathbf{b}|cos(\theta)$$

so

$$\mathbf{a} \cdot \mathbf{b} = \frac{|\mathbf{a}|^2 + |\mathbf{b}|^2 - |\mathbf{a} - \mathbf{b}|^2}{2}$$

Analytically (planar case):

let $\mathbf{a} = [a_1, a_2]$ and $\mathbf{b} = [b_1, b_2]$; we get

$$\mathbf{a} \cdot \mathbf{b} = \frac{a_1^2 + a_2^2 + b_1^2 + b_2^2 - (a_1 - b_1)^2 - (a_2 - b_2)^2}{2} = a_1 b_1 + a_2 b_2$$

Next Arguments

- The linear nature of Naïve Bayes

  (boolean case)

The linear nature of Naïve Bayes

X_i, Y booleans

Classification of $\vec{x} = \langle x_1, \ldots, x_n \rangle$:

$$\frac{P(Y=1|X_1 \ldots X_n = \vec{x})}{P(Y=0|X_1 \ldots X_n = \vec{x})} = \frac{P(Y=1)\prod_i P(X_i = x_i|Y=1)}{P(Y=0)\prod_i P(X_i = x_i|Y=0)} \geq 1$$

Passing to logarithms

$$log\frac{P(Y=1)}{P(Y=0)} + \sum_i log\frac{P(X_i = x_i|Y=1)}{P(X_i = x_i|Y=0)} \geq 0$$

Let $\theta_{ik} = P(X_i = 1|y = k)$ (hence $P(X_i = 0|y = k) = 1 - \theta_{ik}$), then we have the following linear discrimination test:

$$log\frac{P(Y=1)}{P(Y=0)} + \sum_i x_i \cdot log\frac{\theta_{i1}}{\theta_{i0}} + \sum_i (1-x_i) \cdot log\frac{1-\theta_{i1}}{1-\theta_{i0}} \geq 0$$

The case unfolding trick

Consider a function $f(x)$ where x is a boolean variable, then:

$$f(x) = x * f(1) + (1 - x) * f(0)$$

In general, given a function $f(x)$ with x ranging over a discrete domain $\{a_1, \ldots, a_n\}$ we can unfold f in the form

$$\begin{aligned} f(x) = \text{ case } x \text{ of:} \\ & [a_1 \Rightarrow f(a_1) \\ & |a_2 \Rightarrow f(a_2) \\ & \ldots \\ & |a_n \Rightarrow f(a_n) \\ &] \end{aligned}$$

A form of (finite) memoization.

Linear Classification

Classification algorithms based on a linear combination of the features values.

Every feature is evaluated independently from the others and contributes to the result in a linear way, with a suitable weight (that is a parameter of the model, to be estimated).

For instance, if the features are pixels of some image, we may use linear methods only up some normalization (in position and dimension) of the object to be recognized.

Gaussian Naïve Bayes

What if the features X_i are continuous?

For instance, X_i could be the height or age or annual income of some individual, or the color of a pixel in an image, etc.

To use Naïve Bayes we need to compute $P(X_i|Y)$, but if X_i is continuous, pointwise probabilities are null.

A common approach is to suppose $P(X_i|Y)$ has a Gaussian (or Normal) distribution.

Gaussian Distribution

Probability density (with integral = 1)

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

mean value

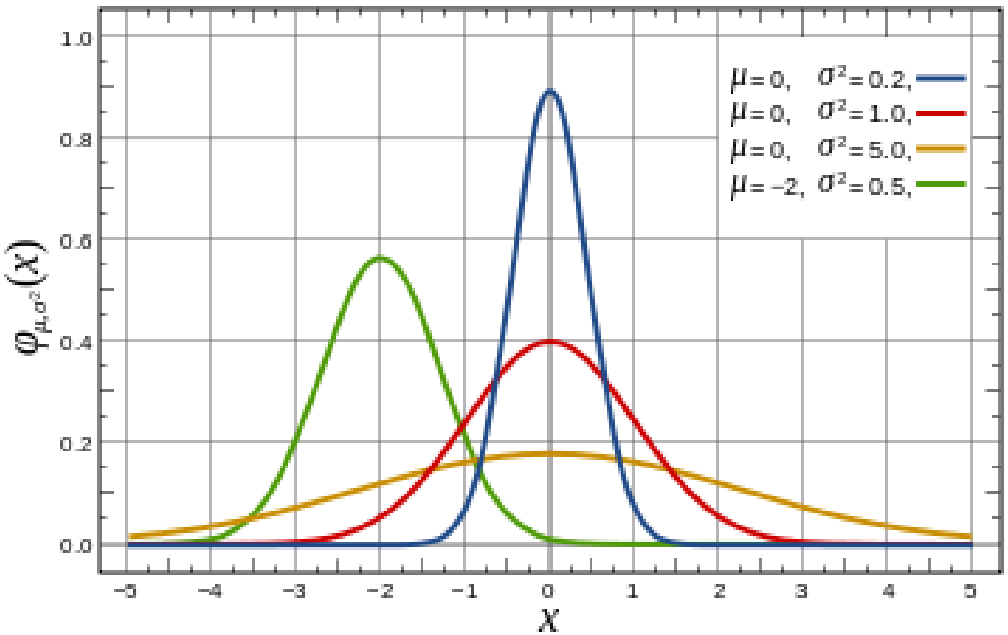
$$E[X] = \mu$$

variance

$$Var[X] = \sigma^2$$

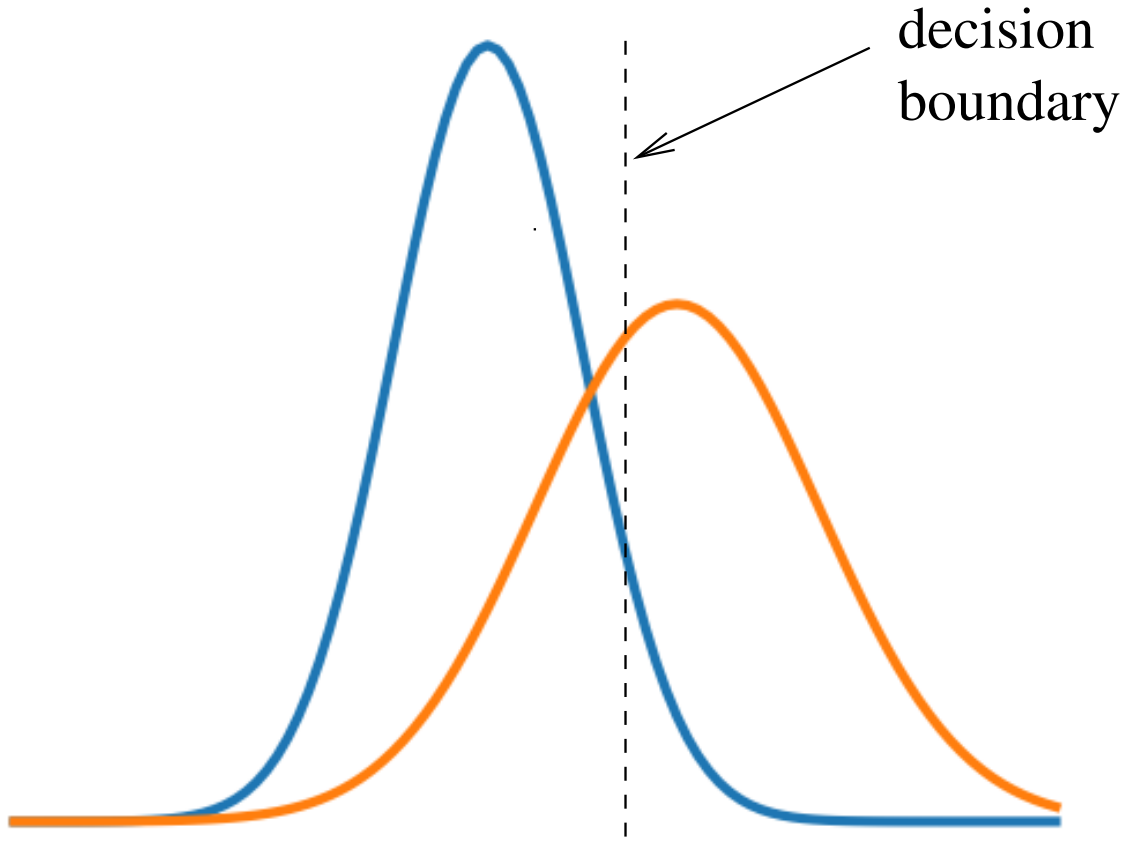
standard deviation

$$\sigma_X = \sigma$$

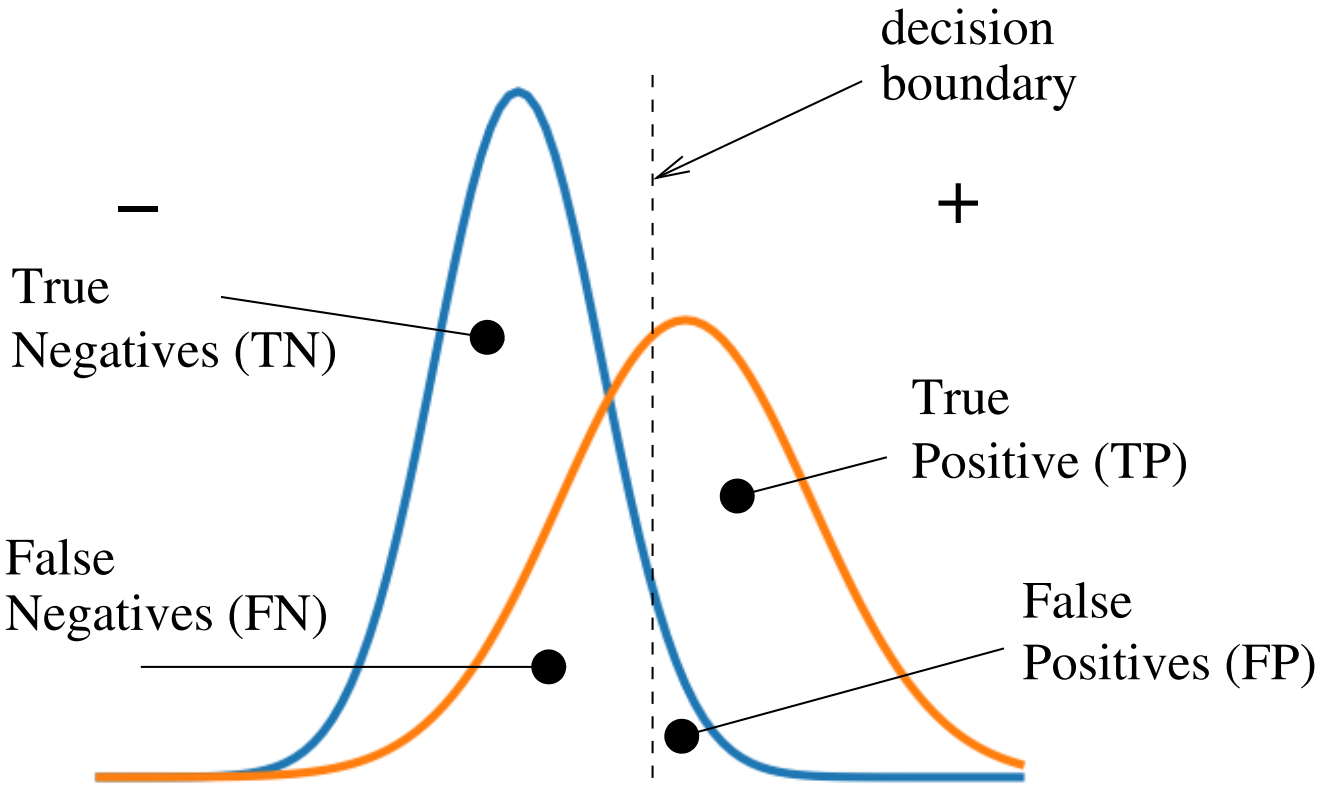


Example

Classify gender given height.



TP, FP, TN, FN


decision
boundary
–
+
True
Negatives (TN)
True
Positive (TP)
False
Negatives (FN)
False
Positives (FP)

Accuracy, Precision and Recall

$$Accuracy = \frac{TP + TN}{All}$$

How many samples have been correctly classified?

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

how many of the selected samples are relevant?

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

how many of relevant samples have been selected?

$$F1 = 2\frac{Precision \cdot Recall}{Precision + Recall}$$

armonic mean of precision and recall

Descriptive parameters of the model

We assume (inductive bias) that for every value y_k of Y the random variable $P(X_i|Y = y_k)$ has a Gaussian distribution

$$N(x|\mu_{ik}, \sigma_{ik}) = \frac{1}{\sigma_{ik}\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_{ik}^2}}$$

Learning: estimate the values of the parameters μ_{ik}, σ_{ik} and $\pi_k = P(Y = y_k)$.

Classification of $x^{new} = \langle a_1, \ldots a_n \rangle$

$$\begin{aligned} Y^{new} &= arg\ max_{y_k}\ P(Y = y_k) \cdot \prod_i P(X_i = a_i|Y = y_k) \\ &= arg\ max_k\ \pi_k \cdot \prod_i N(a_i|\mu_{ik}, \sigma_{ik}) \end{aligned}$$

Maximum Likelihood Estimates

μ_{ik} = **mean value of** X_i **for samples with label** $Y = y_k$

Formally:

$$\mu_{ik} = \frac{\sum_j X_i^j \delta(Y^j = y_k)}{\sum_j \delta(Y^j = y_k)}$$

where j ranges over samples in the training set

$$\delta(Y^j = y_k) = \begin{cases} 1 & \textit{se } Y^j = y_k \\ 0 & \textit{altrimenti} \end{cases}$$

σ_{ik}^2 = variance of X_i for samples with label $Y = y_k$

$$\sigma_{ik}^2 = \frac{\sum_j (X_i^j - \mu_{ij})^2 \delta(Y^j = y_k)}{\sum_j \delta(Y^j = y_k)}$$

Next arguments

- Logistic Regression
- The logistic function

Logistic Regression

Idea:

- Naïve Bayes allows us to compute $P(Y|X)$ after having learned $P(Y)$ and $P(X|Y)$
- Why not directly learn $P(Y|X)$?

The shape of $P(Y|X)$

Hypothesis:

- Y boolean random variable
- X_i continuous random variable
- X_i independent from each other Y
- $P(X_i|Y = k)$ have Gaussian distributions $N(\mu_{ik}, \sigma_i)$ (**Warning** not σ_{ik}!)
- Y has a Bernoulli distribution (π)

Then

$$P(Y = 1|X = \langle x_1 \dots x_n \rangle) = \frac{1}{1 + exp(w_0 + \sum_i w_i x_i)}$$

Proof

By hypothesis

$$P(X_i|Y=k) = N(X_i, \mu_{ik}, \sigma_i) = \frac{1}{\sigma_i\sqrt{2\pi}} e^{-\frac{(x-\mu_{ik})^2}{2\sigma_i^2}}$$

Hence:

$$\begin{aligned}
P(Y=1|X) &= \frac{P(Y=1)\cdot P(X|Y=1)}{P(Y=1)\cdot P(X|Y=1) + P(Y=0)\cdot P(X|Y=0)} \\
&= \frac{1}{1+\frac{P(Y=0)\cdot P(X|Y=0)}{P(Y=1)\cdot P(X|Y=1)}} \\
&= \frac{1}{1+exp(ln(\frac{P(Y=0)\cdot P(X|Y=0)}{P(Y=1)\cdot P(X|Y=1)}))} \\
&= \frac{1}{1+exp(ln\frac{1-\pi}{\pi} + \sum_i ln\frac{P(X_i|Y=0)}{P(X_i|Y=1)})} \\
&= \frac{1}{1+exp(ln\frac{1-\pi}{\pi} + \sum_i (\frac{\mu_{i0}-\mu_{i1}}{\sigma_i^2}X_i + \frac{\mu_{i1}^2-\mu_{i0}^2}{2\sigma_i^2})}
\end{aligned}$$

Very useful!

If

$$P(Y = 1|X = \langle x_1 \dots x_n \rangle) = \frac{1}{1 + exp(w_0 + \sum_i w_i x_i)}$$

then

$$P(Y = 0|X = \langle x_1 \dots x_n \rangle) = \frac{exp(w_0 + \sum_i w_i x_i)}{1 + exp(w_0 + \sum_i w_i x_i)}$$

Hence

$$\frac{P(Y = 0|X = \langle x_1 \dots x_n \rangle)}{P(Y = 1|X = \langle x_1 \dots x_n \rangle)} = exp(w_0 + \sum_i w_i x_i)$$

and in particular

$$ln \frac{P(Y = 0|X = \langle x_1 \dots x_n \rangle)}{P(Y = 1|X = \langle x_1 \dots x_n \rangle)} = w_0 + \sum_i w_i x_i$$

To classify $X = \langle x_1 \dots x_n \rangle$ it suffices to check if $w_0 + \sum_i w_i x_i > 0$

Logistic regression

Logistic regression *assumes*

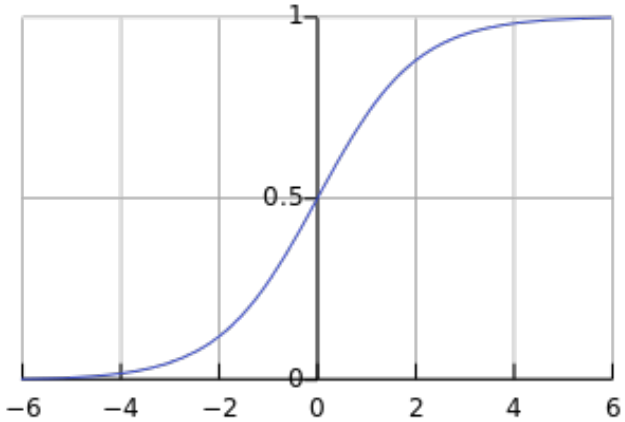
$$P(Y = 1|X = \langle x_1 \dots x_n \rangle) = \frac{1}{1 + exp(-w_0 - \sum_i w_i x_i)}$$

and directly tries to estimate the parameters w_i (the change in sign is ininfluent).

The function

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

is an important function called logistic function



Training for logistic regression (binary case)

We know that

$$P(Y = 1|x_i w_i) = \sigma(w_0 + \sum_i w_i x_i)$$

Given the (independent) samples $\langle x_i^\ell, y^\ell \rangle$, their probability is

$$\prod_\ell P(y^\ell|x_i^\ell w_i) = \prod_\ell P(y^\ell = 1|x_i^\ell w_i)^{y^\ell} \cdot P(y^\ell = 0|x_i^\ell w_i)^{(1-y^\ell)}$$

We want to find the values for the parameters w_i that maximize this probability (MLE).

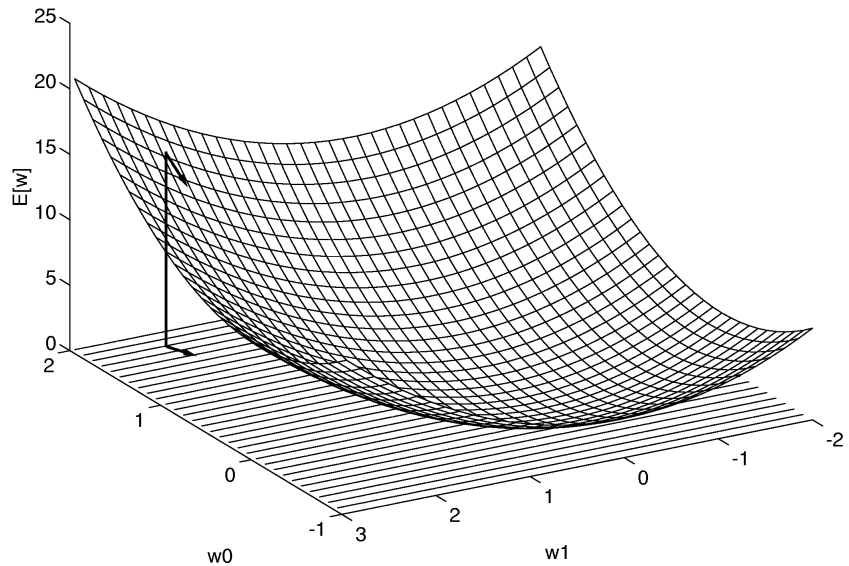
Analogously, we can work on the logarithm and maximize

$$\sum_\ell log\ P(y^\ell|x_i^\ell w_i) =$$
$$\sum_\ell (y^\ell \cdot log\ P(Y = 1|x_i^\ell w_i) + (1 - y^\ell) \cdot log\ P(Y = 0|x_i^\ell w_i))$$

Gradient

Unfortunately, there is no analytic solution for the previous optimization problem.

so, we use iterative optimization methods, like the gradient technique:



$$\nabla_w[E] = [\frac{\partial E}{\partial w_1}, \ldots, \frac{\partial E}{\partial w_n}]$$

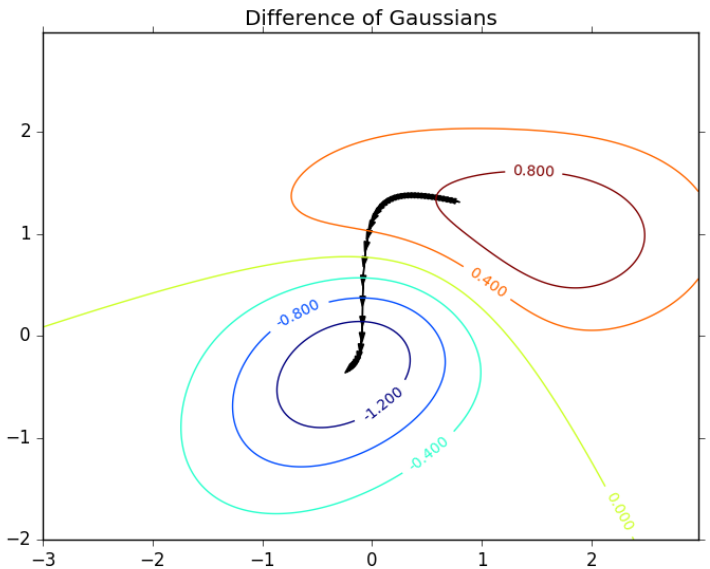
training: $\Delta w_i = \mu \cdot \frac{\partial E}{\partial w_i}$

$$w_i = w_i + \Delta w_i$$

The gradient technique

Iterative approach

The objective is to minimize some error function $\Theta(w)$ on all training examples, suitably adjusting the parameters.

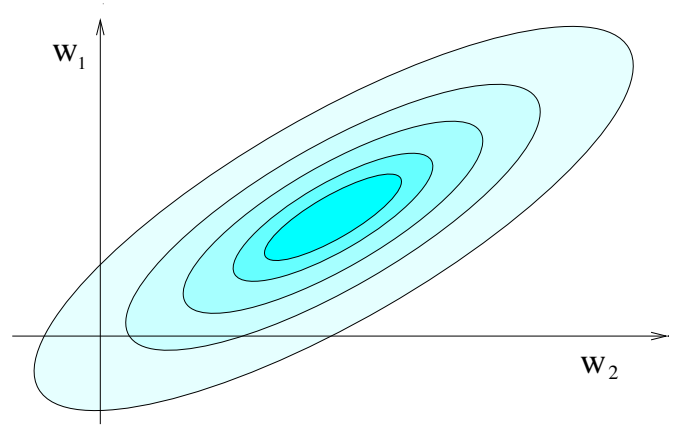


We can reach a minimal configuartion for $\Theta(w)$ by iteratively taking small steps in the direction opposite to the gradient (gradient descent).

This is a **general technique**.

The error surface

Image one "horizontal" axis for each parameter and one "vertical" axis for the error.

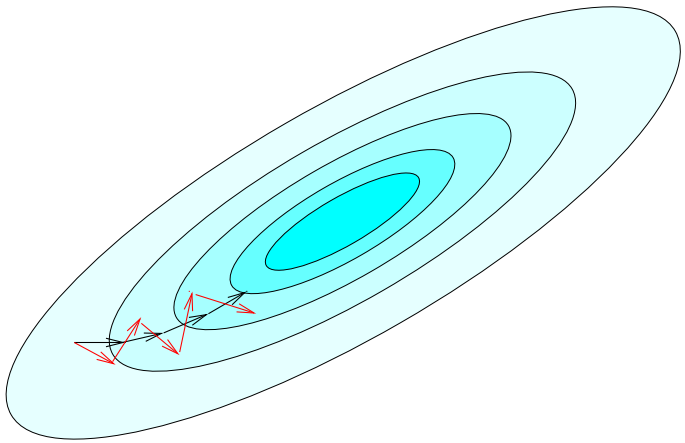


For a linear net with a squared error, the surface is a quadratic bowl:
- Vertical cross-sections are parabolas
- Horizontal cross-sections are ellipses

The error surface for logistic regression is convex too.

For multi-layer, non-linear nets the error surface can be much more complicated (with many local minima).

Online vs Batch learning

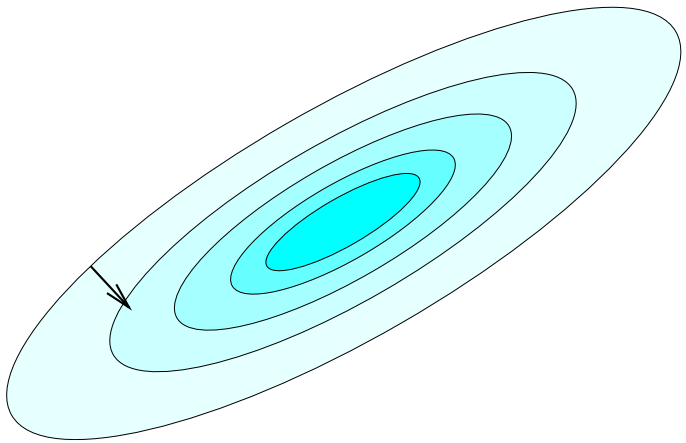


Full batch on all training samples: gradient points to the direction of steepest descent on the error surface (perpendicular to contour lines of the error surface)

Online (one sample at a time) gradient zig-zags around the direction of the steepest descent.

Mini-batch (random subset of training samples): a good compromise.

Backpropagation can be slow



The gradient does not necessarily points to the direction of the local minimum

Gradient for logistic regression

- probability that sample ℓ is in category $Y = 1$

$$P(Y = 1|x_i^\ell w_i) = \sigma(w_0 + \sum_i w_i x_i^\ell) = \alpha^\ell$$

- Log-likelihood $l(w)$

$$\sum_\ell log\, P(Y = y^\ell|x_i^\ell w_i) = \sum_\ell y^\ell log(\alpha^\ell) + (1 - y^\ell) log(1 - \alpha^\ell)$$

- gradient (proof on next slide)

$$\frac{\partial l(w)}{\partial w_i} = \sum_\ell x_i^\ell \cdot (y^\ell - \alpha^\ell)$$

Computing the gradient

$$\alpha^\ell = \sigma(z) = \frac{1}{1 + exp(-z))} \text{ where } z = w_0 + \sum_i w_i x_i^\ell$$

$$\frac{\partial log(\alpha^\ell)}{\partial z} = \frac{1}{\alpha^\ell}\frac{\partial \alpha^\ell}{\partial z} = \frac{exp(-z)}{1 + exp(-z)} = 1 - \alpha^\ell$$

$$log(1 - \alpha^\ell) = log\frac{exp(-z)}{1 + exp(-z)} = -z + log(\alpha^\ell)$$

$$\frac{\partial log(1 - \alpha^\ell)}{\partial z} = -1 + 1 - \alpha^\ell = -\alpha^\ell$$

remember that $l(w) = \sum_\ell y^\ell log(\alpha^\ell) + (1 - y^\ell) log(1 - \alpha^\ell)$

$$\frac{\partial l(w)}{\partial w_i} = \frac{\partial l(w)}{\partial z}\frac{\partial z}{\partial w_i} \quad = (\sum_\ell y^\ell(1 - \alpha^\ell) - (1 - y^\ell)\alpha^\ell)x_i^\ell$$
$$= (\sum_\ell (y^\ell - \alpha^\ell))x_i^\ell$$

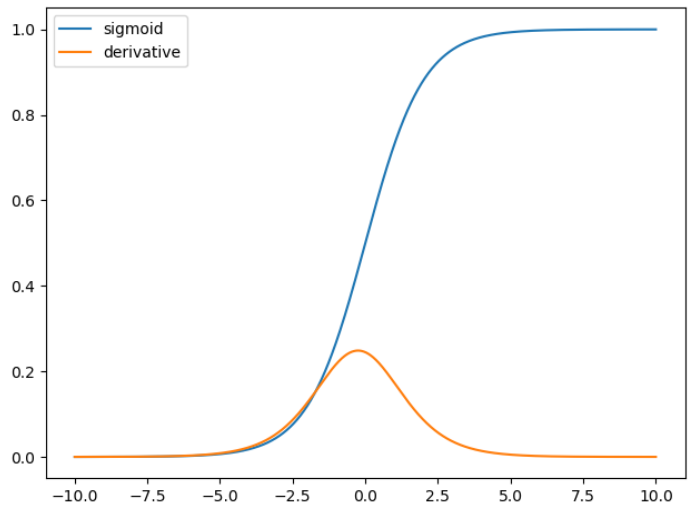
Observation: the derivative of the sigmoid

In the previous slide we proved that

$$\frac{\partial log(\alpha^{\ell})}{\partial z} = \frac{\partial \sigma(z)}{\partial z} = \frac{1}{\sigma(z)} \frac{\partial \sigma(z)}{\partial z} = 1 - \sigma(z)$$

Hence:

$$\frac{\partial \sigma(z)}{\partial z} = \sigma(z)(1 - \sigma(z))$$



The learning process

Iterate the following update operation until reaching the desired approximation (e.g. until accuracy on testing data is satisfactory, or increment is below a given threshold ϵ)

$$w_i \leftarrow w_i + \mu \sum_{\ell} x_i^{\ell} \cdot (y^{\ell} - P(Y = y^{\ell} | x_i w_i))$$

Frequently one add a regularizer (prior):

$$w_i \leftarrow w_i - \mu\lambda |w_i| + \mu \sum_{\ell} x_i^{\ell} \cdot (P(Y = y^{\ell} | x_i w_i) - y^{\ell})$$

- helps to keep parameters w_i close to 0
- tends to reduce overfitting

Log likelyhood vs square error

Let us compare a couple of classification models, according to their predictions:

correct label	1	0	1	0
prediction model 1	.6	.6	.6	.4
prediction model 2	.8	.8	.7	.25

Log likelyhood vs square error

Let us compare a couple of classification models, according to their predictions:

correct label	1	0	1	0
prediction model 1	.6	.6	.6	.4
prediction model 2	.8	.8	.7	.25

negative log likelihood: $-\sum_\ell logP(Y = y^\ell | x^\ell)$

model 1 : $-log(.6) - log(.4) - log(.6) - log(.6) \approx 3.53$

model 2 : $-log(.8) - log(.2) - log(.7) - log(.75) \approx 3.57$

Log likelyhood vs square error

Let us compare a couple of classification models, according to their predictions:

correct label	1	0	1	0
prediction model 1	.6	.6	.6	.4
prediction model 2	.8	.8	.7	.25

negative log likelihood: $-\sum_\ell logP(Y = y^\ell | x^\ell)$

model 1 : $-log(.6) - log(.4) - log(.6) - log(.6) \approx 3.53$

model 2 : $-log(.8) - log(.2) - log(.7) - log(.75) \approx 3.57$

square error: $\sum_\ell (y^\ell - pred^\ell)^2$

model 1 : $(1 - .6)^2 + (0 - .6)^2 + (1 - .6)^2 + (0 - .4)^2 = .84$

model 2 : $(1 - .8)^2 + (0 - .8)^2 + (1 - .7)^2 + (0 - .25)^2 = .8325$

Log likelyhood vs square error: the gradients

negative loglikelihood

$$\frac{\partial l(z)}{\partial z} = \sigma(z) - y^{\ell}$$

square error

$$\frac{\partial(\sigma(z) - y^{\ell})^2}{\partial z} = 2(\sigma(z) - y^{\ell})\frac{\sigma(z) - y^{\ell}}{\partial z}$$

$$= 2(\sigma(z) - y^{\ell})\sigma(z)(1 - \sigma(z))$$

The gradient for s.e. is nearly 0 if the prediction is entirely wrong!!

Generative vs discriminative classifiers

Generative vs discriminative methods

Classification: estimate $f : X \rightarrow Y$ or $P(Y|X)$.

Generative Classifiers (e.g., Naïve Bayes)

- assume a given distribution for $P(X|Y), P(X)$
- estimate parameters for $P(X|Y), P(X)$ from training data
- use Bayes' rule to infer $P(Y|X)$

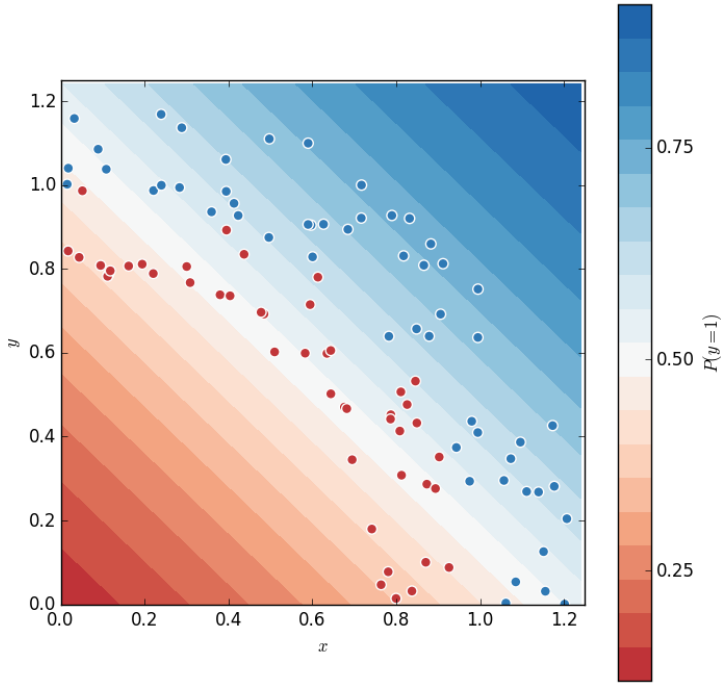
Discriminative Classifiers (e.g., Logistic regression)

- assume a given distribution for $P(Y|X)$ (or a given shape for the discrimination function)
- estimate the parameters for $P(Y|X)$ (or the discrimination function) from training data

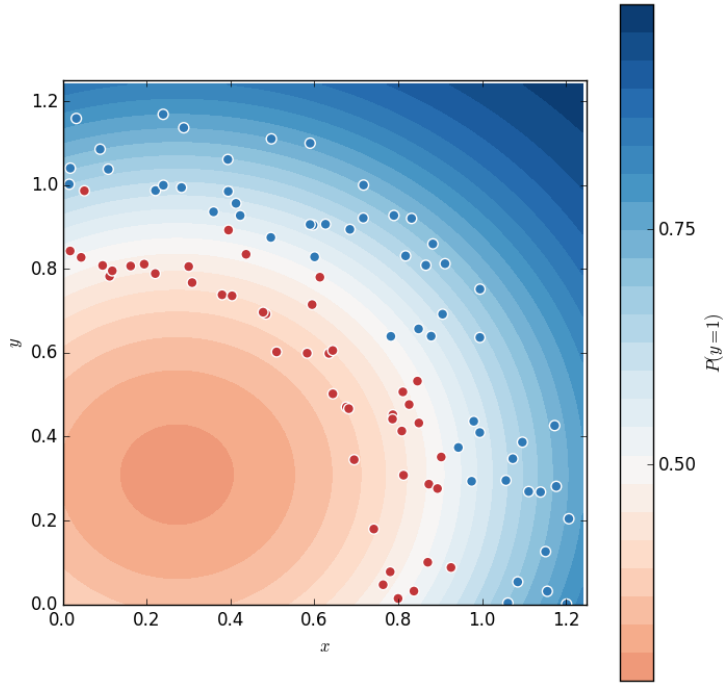
Naïve Bayes vs. Logistic regression

Example: points with random angle $0 \leq \theta \leq \pi$ and random distance $.75 \leq r \leq 1.25$ from origin.
Input features are the cartesian coordinates of the points (not mutually independent), and we try to discriminate points internal to the unary circle.

logistic regression



gaussian naïve bayes



Lo script in python (1)

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from matplotlib import pyplot as plt
from matplotlib.colors import ListedColormap

size = 100
# random training data
X = np.random.rand(size, 2)

def b(p): return p[1]>.5
y = map(b,X)

def toXY(p):
  theta = p[0]*np.pi/2
  len = (p[1]-.5)/2+1
  x,y=np.cos(theta)*len,np.sin(theta)*len
  return np.array([x,y],float)

for i in range(len(X)): X[i] = toXY(X[i])

xx, yy = np.mgrid[0:1.25:.01, 0:1.25:.01]
grid = np.c_[xx.ravel(), yy.ravel()]
```

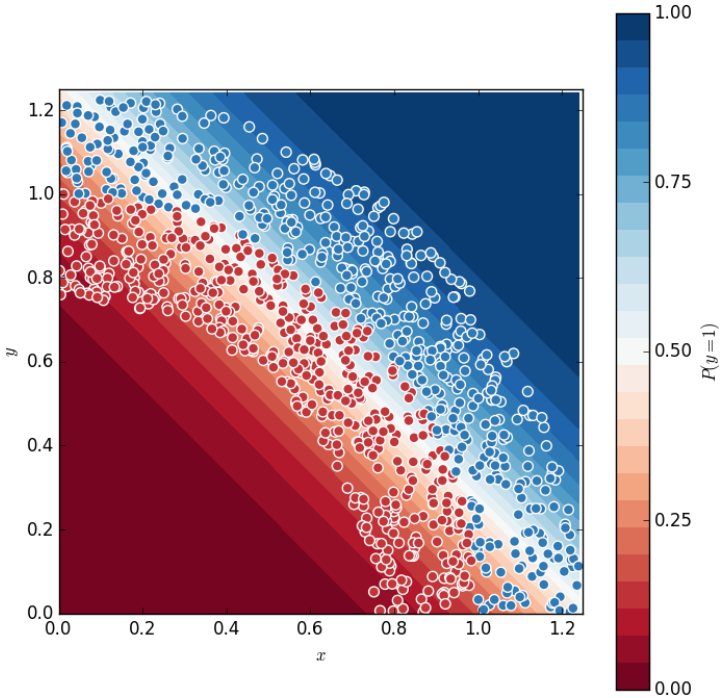
Lo script in python (2)

```
names = ["Logistic Regression", "Gaussian Naive Bayes"]
classifiers = [LogisticRegression(fit_intercept=True),GaussianNB()]

for name, clf in zip(names, classifiers):
  clf.fit(X,y)
  probs = clf.predict_proba(grid)[:, 1].reshape(xx.shape)
  #Now, plot the probability grid as a contour map
  f, ax = plt.subplots(figsize=(8, 8))
  contour = ax.contourf(xx, yy, probs, 25, cmap="RdBu",vmin=0, vmax=1)
  ax_c = f.colorbar(contour)
  ax_c.set_label("$P(y = 1)$")
  ax_c.set_ticks([0, .25, .5, .75, 1])
  #show the test set samples on top of it
  ax.scatter(X[:,0], X[:, 1], c=y, s=50,
             cmap="RdBu", vmin=-.2, vmax=1.2,
             edgecolor="white", linewidth=1)
  ax.set(aspect="equal",
         xlim=(0, 1.25), ylim=(0, 1.25),
         xlabel="$x$", ylabel="$y$")

plt.show()
```

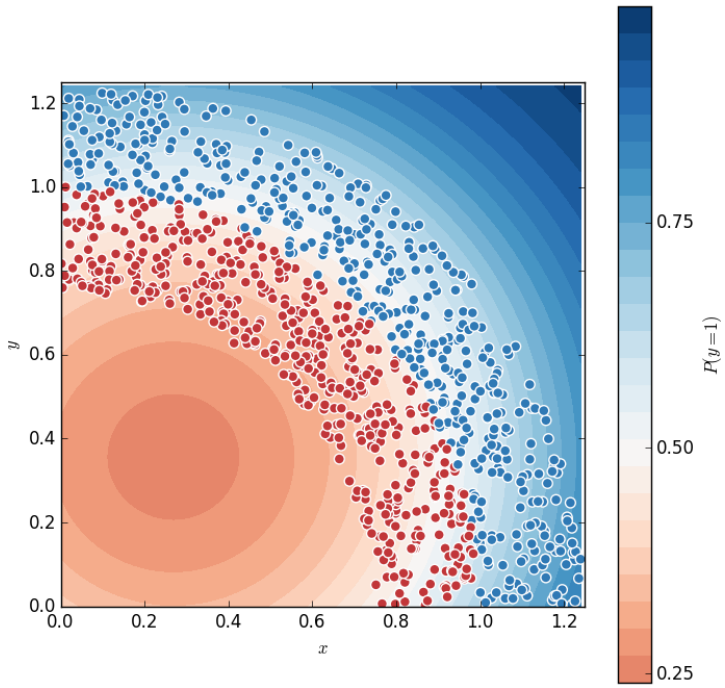
Naïve Bayes vs.Logistic regression

Same example with 1000 points instead of 100.

logistic regression



gaussian naïve bayes



Naïve Bayes vs.Logistic regression

Similar techniques.

Naïve Bayes is slightly more expressive.

Logistic regression is simpler and makes fewer assumptions on the model.

Cross entropy

Kullback-Leibler divergence

The Kullback-Leibler divergence $DKL(P\|Q)$ between two distributions Q and P, is a measure of the information loss due to approximating P with Q:

$$\begin{aligned} DKL(P\|Q) &= \sum_i P(i) log \frac{P(i)}{Q(i)} \\ &= \sum_i P(i)(logP(i) - logQ(i)) \\ &= -\underbrace{\mathcal{H}(P)}_{entropy} - \sum_i P(i) logQ(i) \end{aligned}$$

We call Cross-Entropy between P and Q the quantity

$$\mathcal{H}(P,Q) = -\sum_i P(i) logQ(i) = \mathcal{H}(P) + DKL(P\|Q)$$

Minimizing the cross entropy

Let P be the distribution of training data, and Q the distribution induced by the model.

We can take as our learning objective the minimization of the Kullback-Leibler divergence $DKL(P\|Q)$.

Since, given the training data, their entropy $\mathcal{H}(P)$ is constant, minimizing $DKL(P\|Q)$ is equivalent to minimizing the cross-entropy $\mathcal{H}(P,Q)$ between P and Q.

Cross entropy and log-likelihood

Let us consider the case of a binary classification.

Let $Q(y = 1|\mathbf{x})$ the probability that $\mathbf{x}$ is classified 1.
Hence, $Q(y = 0|\mathbf{x}) = 1 - Q(y = 1|\mathbf{x})$.

The real (observed) classification is $P(y = 1|\mathbf{x}) = y$ and similarly $P(y = 0|\mathbf{x}) = 1 - y$.

So we have

$$\begin{aligned}\mathcal{H}(P, Q) &= -\sum_i P(i) log Q(i) \\ &= -y\, log(Q(y = 1|\mathbf{x})) - (1 - y) log(1 - Q(y = 1|\mathbf{x})\end{aligned}$$

That is just the (negative) log-likelihood!

Multinomial Logistic Regression

Use of the multinomial model

Classification problems with multiple categories

- means of transport (bus, car, train, bicycle, etc.)
- favourite ice flavor (strawberry, lemon, cream, etc.)
- different characters in writing recognition
- ...

Multinomial regression allows us to associate a probability with the different categories, that is particularly useful when combining in cascade different learning techniques.

Binary case

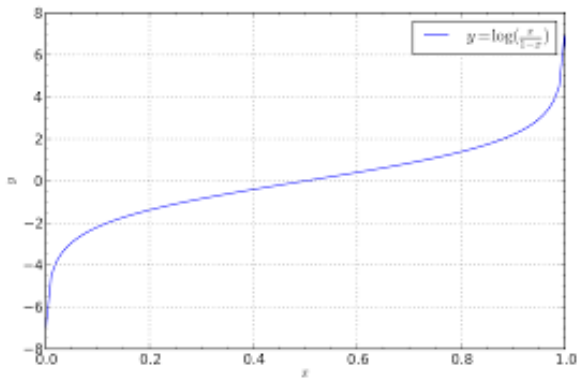
In the case of binary regression, we assume that the discrimination function between the two cases $Y = 0$ and $Y = 1$ is linear:

$$\underbrace{\underbrace{log \frac{P(Y = 0|\langle x_1 \dots x_n \rangle)}{P(Y = 1|\langle x_1 \dots x_n \rangle)}}_{odds}}_{log\ odds} = w_0 + \sum_i w_i x_i$$

This is sometimes expressed in terms of the logit function:

$$logit(P(Y = 0|\mathbf{x})) = w_0 + \sum_i w_i x_i$$

$$logit(p) = log \frac{p}{1 - p}$$



Logistic regression is also called binary logit.

n-ary case

In the case of multinomial regression with k categories $\{1, \ldots, k\}$ we have $k-1$ equations describing the log-odds of each category with respect to a reference category (e.g. category k):

$$log \frac{P(Y=1|\langle x_1 \ldots x_n \rangle)}{P(Y=k|\langle x_1 \ldots x_n \rangle)} = w_{1,0} + \sum_i w_{1,i} x_i$$

...

$$log \frac{P(Y=k-1|\langle x_1 \ldots x_n \rangle)}{P(Y=k|\langle x_1 \ldots x_n \rangle)} = w_{k-1,0} + \sum_i w_{k-1,i} x_i$$

This model relies on the **Independence of Irrelevant Alternatives** (IIA) assumption:
the odds between two categories A and B are not influenced by a third category C

Probability distributions

Let us use a vectorial notation (comprising the constant w_0)

$$log \frac{P(Y=j|\mathbf{x})}{P(Y=k|\mathbf{x})} = \mathbf{w}_j \mathbf{x}$$

Then, for every j

$$P(Y=j|\mathbf{x}) = P(Y=k|\mathbf{x}) \cdot e^{\mathbf{w}_j \mathbf{x}}$$

The sum of probabilities must be 1, that is

$$\sum_{j<k} P(Y=k|\mathbf{x}) \cdot e^{\mathbf{w}_j \mathbf{x}} + P(Y=k|\mathbf{x}) = 1$$

and hence

$$P(Y=k|\mathbf{x}) = \frac{1}{1+\sum_{j<k} e^{\mathbf{w}_j \mathbf{x}}} \quad \textit{and} \quad P(Y=j|\mathbf{x}) = \frac{e^{\mathbf{w}_j \mathbf{x}}}{1+\sum_{j<k} e^{\mathbf{w}_j \mathbf{x}}}$$

Comparison with the binary case

Binary case

$$P(Y = 1|\mathbf{x}) = \frac{1}{1 + e^{\mathbf{wx}}} \quad \textit{and} \quad P(Y = 0|\mathbf{x}) = \frac{e^{\mathbf{wx}}}{1 + e^{\mathbf{wx}}}$$

n-ary case

$$P(Y = k|\mathbf{x}) = \frac{1}{1 + \sum_{j<k} e^{\mathbf{w}_j \mathbf{x}}} \quad \textit{and} \quad P(Y = j|\mathbf{x}) = \frac{e^{\mathbf{w}_j \mathbf{x}}}{1 + \sum_{j<k} e^{\mathbf{w}_j \mathbf{x}}}$$

There exists a generalization of the logistic function?

softmax

The function

$$\text{softmax}(j, x_1, \ldots, x_k) = \frac{e^{x_j}}{\sum_{j=1}^{k} e^{x_j}}$$

generalizes the logistic function to the multinomial case.

Using softmax, we can express probabilities in the following way

$$P(Y = j|\mathbf{x}) = \text{softmax}(j, \mathbf{w}_1\mathbf{x}, \ldots, \mathbf{w}_{k-1}\mathbf{x}, 0)$$

It is easy to prove that for any c,

$$\text{softmax}(j, x_1 + c, \ldots, x_k + c) = \text{softmax}(j, x_1, \ldots, x_k)$$

hence, it is natural to assume that one of the argument (corresponding to the "reference category") is null, taking e.g. $c = -x_k$.

The same reasoning shows the irrelevance of the choice of the reference category for computing odds.

Training

The matrix of parameters $\mathbf{w_j}$ is computed via maximum likelihood estimation. The probability that a sample $\langle \mathbf{x}^\ell, y^\ell \rangle$ is correctly classified in the category y^ℓ is

$$P(Y = y^\ell | \mathbf{x}) = \prod_j P(Y = j | \mathbf{x})^{\delta_{jy^\ell}}$$

where δ_{ij} ìs the Kronecker δ: $\delta_{ij} := \begin{cases} 1 & \text{if } i = j \\ 0 & \text{if } i \neq j \end{cases}$

Supposing that data are independent, their likelihood is the product of individual probabilities. Passing to logarithms:

$$\sum_l \sum_j \delta_{jy^\ell} \cdot log(P(Y = j | \mathbf{x}))$$

that is usually maximized using a gradient technique.

About the loss function

$$\sum_{l} \underbrace{\sum_{j} \delta_{jy^{\ell}} \cdot log(P(Y = j|\mathbf{x}))}_{means\ \ that...}$$

for sample ℓ you only need to consider the category j corresponding to the true label y^{ℓ}, and the log of the probability $P(Y = j|\mathbf{x}))$ assigned by the model to the sample.

For multinomial regression, the loss function deriving from maximum loglikelihood is the categorical crossentropy between the observed categorical probability $\delta_{jy^{\ell}}$ of data (1 for the right category, 0 for others) and that syntesized by the model.

Gradient for multinomial logistic regression

- probability that sample ℓ is in category $Y = j$

$$P(Y = j|\mathbf{x}) = \text{softmax}(j, \mathbf{w}_1\mathbf{x}, \ldots, \mathbf{w}_{k-1}\mathbf{x}, 0) = \alpha_j^l$$

- Log-likelihood $l(w)$

$$\sum_l \sum_j \delta_{jy^\ell} \cdot log(P(Y = j|\mathbf{x}))$$

- gradient (feature i and category j)

$$\frac{\partial l(w)}{\partial w_{ij}} = \sum_\ell x_i^\ell \cdot (y^\ell - P(Y = j|\mathbf{x}))$$

Example: classification of mnist digits with MLR

Linear methods are powerful

One may have the erroneous feeling that linear methods are of no real interest, since too weak.

Usually, they becomes interesting when you have a large number of (independent) features at your disposal.

In a high dimensional space training points presumably have a very sparse distribution, hence it may becomes plausible to discriminate classes via hyperplanes.

Thinking in higher dimensions

The n-ball volume example.

The volume of a d-dimensional sphere of radius r grows as

$$Kr^d$$

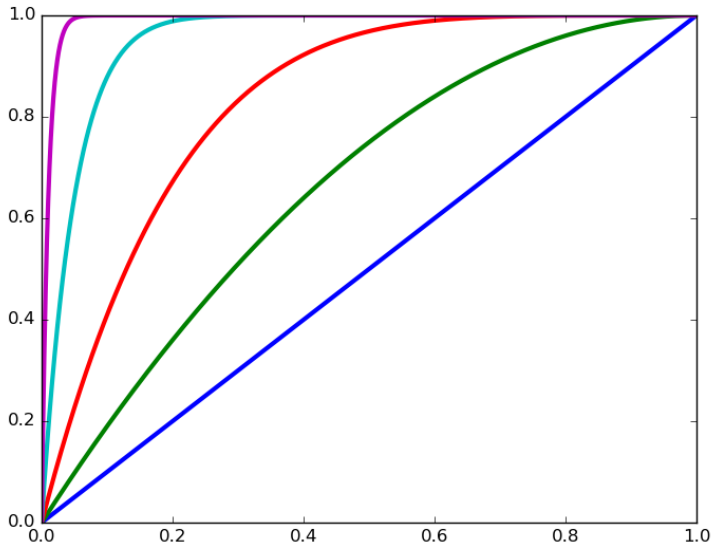
for some constant K.

The fraction of the volume of the sphere that lies between radius $r = 1$ and $r = 1 - \epsilon$ is

$$\frac{K - K(1-\epsilon)^d}{K} = 1 - (1-\epsilon)^d$$

The volume of a n-ball

The function $1 - (1 - \epsilon)^d$ for d=1,2,5,20,100



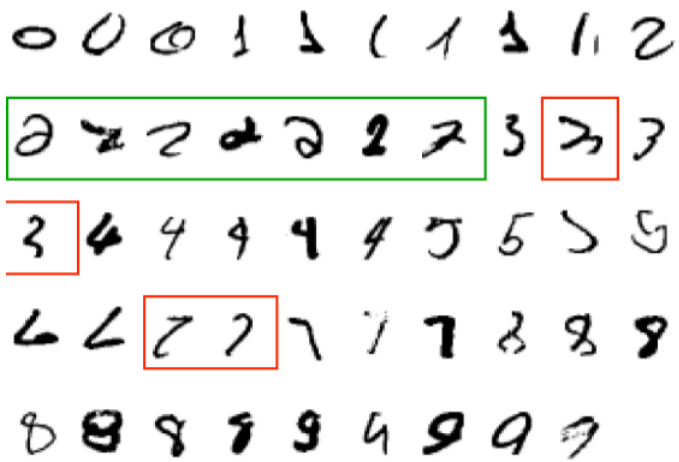
For a really high dimensional sphere, its volume is concentrated in a thin area close to the surface!!

The mnist case

In the case of images, each input digit is a possible feature.

Modified National Institute of Standards and Technology database

- grayscale images of handwritten digits, 28×28 pixels each
- 60,000 training images and 10,000 testing images



A tiny 28×28 image, already has 784 features.

Classification via multinomial regression

DEMO!

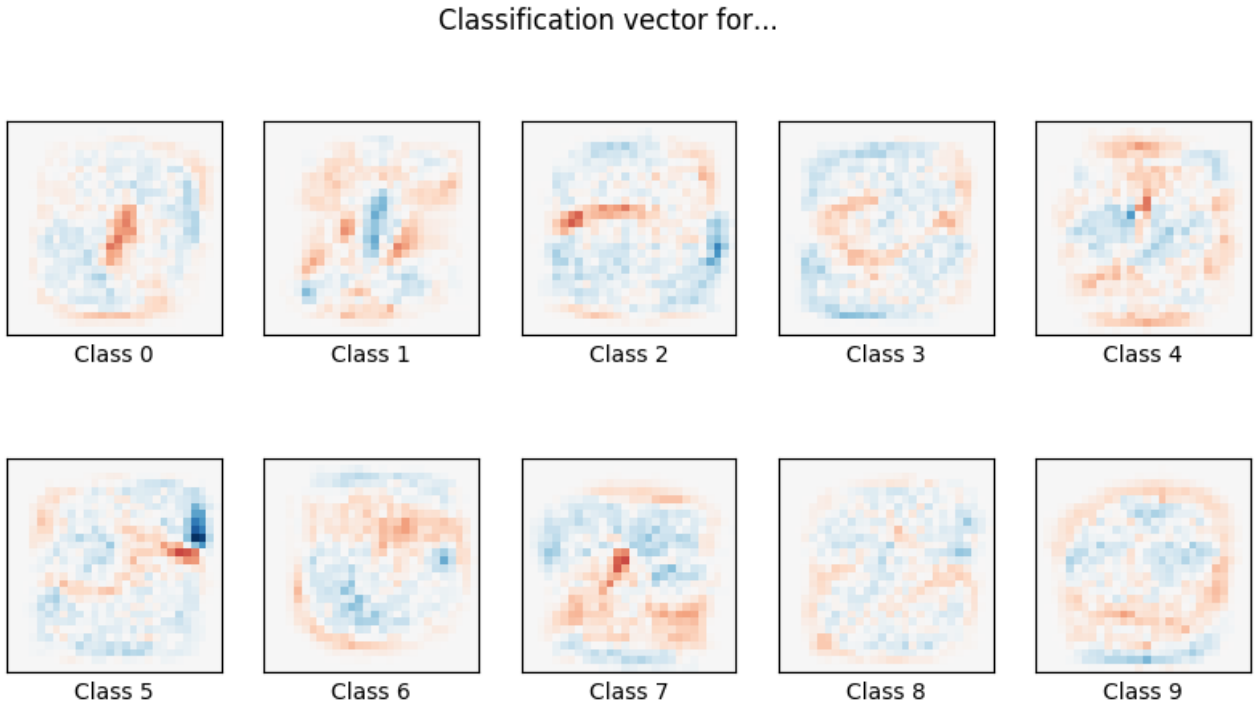
Classification via multinomial regression

DEMO!

```
Sparsity with L1 penalty: 25.18%
Test score with L1 penalty: 0.9257
Example run in 22.404 s
```

Weights as images

For each class, the coresponding weights have the same dimensionality of the input image (we have one weight for each feature).
We can draw them!



Linear Regression

Classification and regression

classification predicting a class:

Y discrete

regression predicting a numerical value:

Y continuous

Warning: the name logistic regression is misleading. Logistic regression is a classification technique.

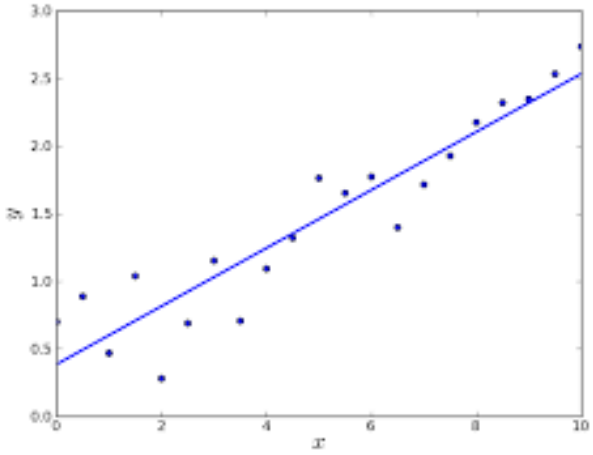
Regression

Learning $f : X \rightarrow Y$ where Y is a real number.

Probabilistic approach:

- choose a distribution $P(Y|X)$, characterized by some parameters θ
- optimize the parameters θ according to training data (MLE or MAP)

Parametric shape of $P(Y|X)$



Let us assume $Y = f(X)$ up to random, gaussian distributed, noise

$$Y = f(X) + \epsilon \text{ where } \epsilon \approx N(0, \sigma)$$

Hence, Y is a random variable with a distribution

$$p(Y|X) = N(f(X), \sigma)$$

with expected value $f(x)$ and standard deviation σ.

Estimating parameters

$$P(Y|X,W) = N[f(x),\sigma] = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(y-f(x))^2}{2\sigma^2}}$$

Log-likelihood:

$$l(W) = \sum_{\ell} ln \frac{1}{\sigma\sqrt{2\pi}} - \frac{(y^{\ell} - f(x^{\ell}))^2}{2\sigma^2}$$

Maximizing this quantity is the same as minimizing

$$err(W) = \sum_{\ell} (y^{\ell} - f(x^{\ell}))^2$$

But this is just the sum of squared errors.

The linear case

If $f(x) = w_0 + w_1(x)$, then

$$err(W) = \sum_{\ell} (y^{\ell} - w_0 - w_1 x^{\ell})^2$$

so

$$\frac{\partial err(W)}{\partial w_0} = -2 \sum_{\ell} (y^{\ell} - w_0 - w_1(x^{\ell}))$$

$$\frac{\partial err(W)}{\partial w_1} = -2 \sum_{\ell} (y^{\ell} - w_0 - w_1(x^{\ell}))x^{\ell}$$

We can use values for the gradient technique...

or solve the optimization problem analytically.

Analytic solution

Equating with 0 the partial derivatives, we get two equations in in two variables:

$$\begin{aligned} \textstyle\sum_\ell y^\ell &= N w_0 + w_1 \textstyle\sum_\ell x^\ell \\ \textstyle\sum_\ell y^\ell x^\ell &= w_0 \textstyle\sum_\ell x^\ell + w_1 \textstyle\sum_\ell (x^\ell)^2 \end{aligned}$$

Let

$$A = \begin{bmatrix} N & \sum_\ell x^\ell \\ \sum_\ell x^\ell & \sum_\ell (x^\ell)^2 \end{bmatrix} \quad W = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix} \quad Y = \begin{bmatrix} \sum_\ell y^\ell \\ \sum_\ell y^\ell x^\ell \end{bmatrix}$$

the previous equations are expressed by the system $AW = Y$, and hence

$$W = A^{-1} Y$$

Example

Let us look for a linear fitting for the points $(1, 2), (2, 3), (6, 4)$.

$$\sum_\ell y^\ell = N w_0 + w_1 \sum_\ell x^\ell$$
$$\sum_\ell y^\ell x^\ell = w_0 \sum_\ell x^\ell + w_1 \sum_\ell (x^\ell)^2$$

In our case,

$$\sum_\ell y^\ell = 9, N = 3, \sum_\ell x^\ell = 9$$
$$\sum_\ell y^\ell x^\ell = 32, \sum_\ell (x^\ell)^2 = 41$$

From which we get the equations

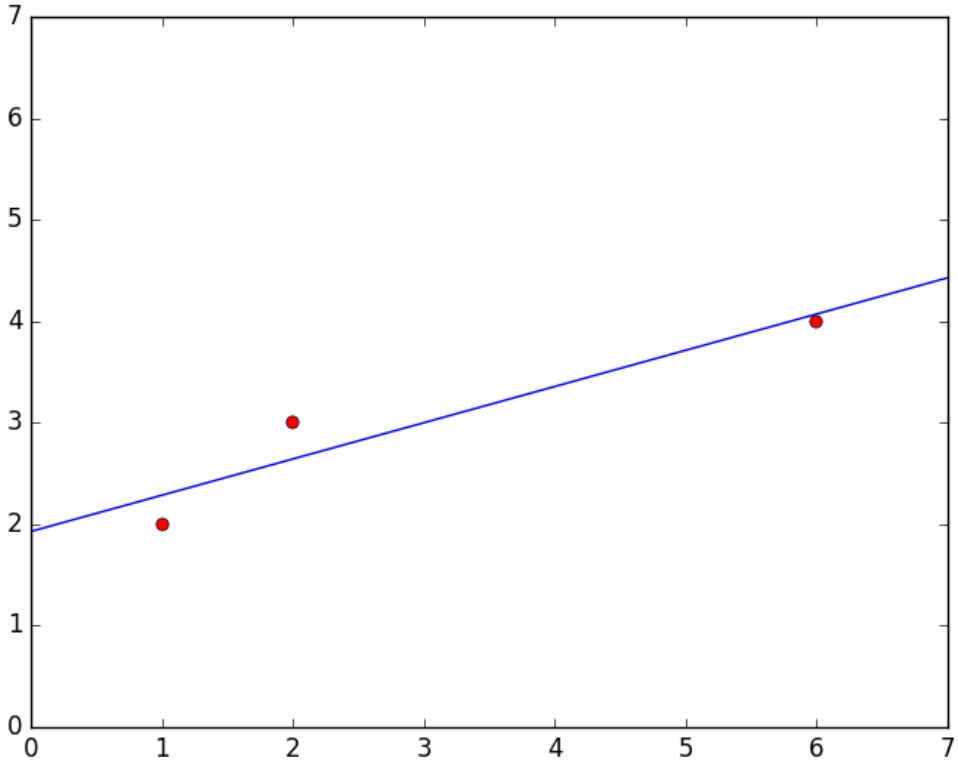
$$9 = 3w_0 + 9w_1$$
$$32 = 9w_0 + 41w_1$$

that has solutions $w_0 = 27/14$ and $w_1 = 5/14$

Example

Linear fitting for the points $(1, 2), (2, 3), (6, 4)$

$$y = \frac{27}{14} + \frac{5}{14}x$$



Regularization

Instead of minimizing

$$err(W) = \sum_{\ell} (y^{\ell} - f(x^{\ell}))^2$$

add a regularizer

$$err(W) = \lambda \sum_{i} w_i^2 + \sum_{\ell} (y^{\ell} - f(x^{\ell}))^2$$

(not solvable any longer by analytic means, even in the linear case)