INM431 Machine Learning

Artur S. d'Avila Garcez
a.garcez@city.ac.uk
http://www.staff.city.ac.uk/~aag/
@AvilaGarcez

Based on G. Hinton's slides and C. Bishop's book

Content

- Gaussian distribution
- Hypothesis evaluation (in the context of the coursework task)
- The Bayesian framework

Expectation and co-variance

The average value of function f(x) under a probability distribution p(x) is called the expectation of f(x), denoted:

$$\mathbb{E}[f] = \sum_x p(x) f(x)$$

Or in the case of continuous variables:

$$\mathbb{E}[f] = \int p(x) f(x) \, \mathrm{d}x$$

Given a finite number N of points drawn from the probability distribution or density then:

$$\mathbb{E}[f] \simeq \frac{1}{N} \sum_{n=1}^{N} f(x_n)$$

Co-variance

cov[x,y] express the extent to which random variables x and y vary together; it is a measure of linear dependence

If x and y are independent then cov[x,y] = 0, but the converse is not true, e.g. $y = x^2$

In the case of two vectors **x** and **y**, the covariance is a matrix

$$\begin{aligned} \mathrm{cov}[x,y] &= \mathbb{E}_{x,y}\left[\{x - \mathbb{E}[x]\}\{y - \mathbb{E}[y]\}\right] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y] \end{aligned}$$

$$\begin{aligned} \mathrm{cov}[\mathbf{x},\mathbf{y}] &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\{\mathbf{x} - \mathbb{E}[\mathbf{x}]\}\{\mathbf{y}^{\mathrm{T}} - \mathbb{E}[\mathbf{y}^{\mathrm{T}}]\}\right] \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}}[\mathbf{x}\mathbf{y}^{\mathrm{T}}] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^{\mathrm{T}}] \end{aligned}$$

Correlation = Normalised co-variance

The magnitude of the covariance is not necessarily easy to interpret…

Correlation:

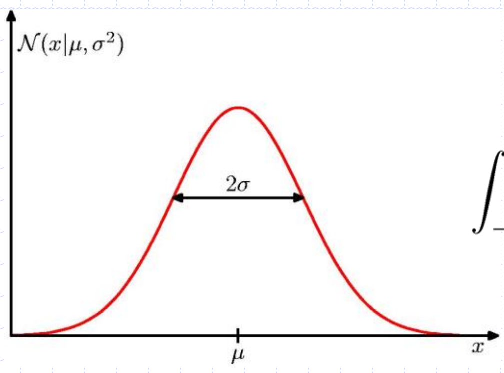
$$\rho_{XY} = E[(X - E[X])(Y - E[Y])]/(\sigma_X \sigma_Y)$$

The covariance of a variable with itself is called its variance

The correlation of a variable with itself is always 1

The Gaussian Distribution

$$\mathcal{N}\left(x|\mu,\sigma^2\right) = \frac{1}{(2\pi\sigma^2)^{1/2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$



$$\mathcal{N}(x|\mu,\sigma^2) > 0$$

$$\int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) \mathrm{d}x = 1$$

Gaussian Mean and Variance

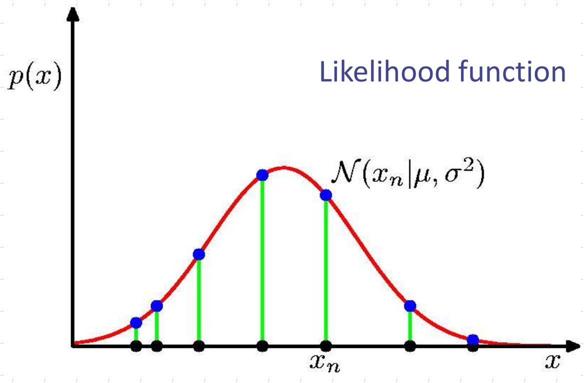
$$\mathbb{E}[x] = \int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) x \,\mathrm{d}x = \mu$$

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} \mathcal{N}\left(x|\mu,\sigma^2\right) x^2 \,\mathrm{d}x = \mu^2 + \sigma^2$$

$$\mathrm{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2$$

Precision parameter β = 1/var[x]

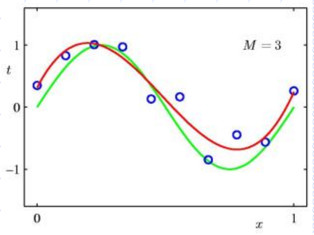
Gaussian Parameter Estimation

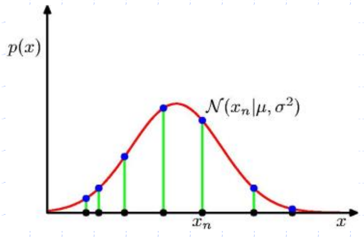


$$p(\mathbf{x}|\mu,\sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(x_n|\mu,\sigma^2\right)$$

So far…

Polynomial or Gaussian parameter estimation (choice is a bias)





$$y(x, \mathbf{w}) = w_0 + w_1 x + w_2 x^2 + \ldots + w_M x^M = \sum_{j=0}^{M} w_j x^j$$

$$p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}\left(x_n|\mu, \sigma^2\right)$$

© Artur Garcez

Coursework task

Choose a dataset

Classification or regression task?

Regression: predict UK Oil and Gas share price given various other share prices

Classification: predict "buy" or "sell" UK Oil and Gas shares

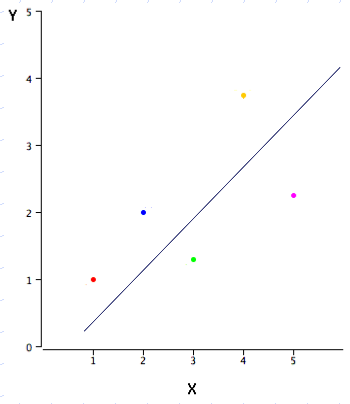
© Artur Garcez

Classification or regression task?

Example data:

X	Y
1.00	1.00
2.00	2.00
3.00	1.30
4.00	3.75
5.00	2.25

Generalization:
X = 6, Y = ?



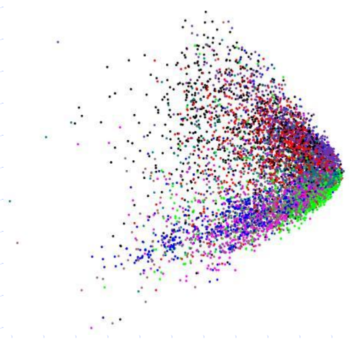
Regression: curve fitting, logistic regression, etc.
Classification: decision trees, naïve Bayes, etc.

Supervised or Unsupervised Learning?

Supervised: presence of target value (or label) allows calculation of error (or loss)
Obtaining good quality labels may not be an easy task

Unsupervised: no target value known (includes reinforcement learning); learn a joint distribution given the data instead

Example of unsupervised learning: Displaying the structure of a set of documents using Latent Semantic Analysis (a form of PCA)



Each document is converted to a vector of word counts. This vector is then mapped to two coordinates and displayed as a colored dot. The colors represent the hand-labeled classes.

When the documents are laid out in 2-D, the classes are not used. So we can judge how good the algorithm is by seeing if the classes are separated.

Semi-supervised learning

Data pre-processing (e.g. dimensionality reduction, which is unsupervised) followed by supervised learning (i.e. in the presence of labels, either a regression or classification task)

(Coursework) Include a description of the data (see example poster): continuous/discrete (binning), how many data points, dimensions, missing values, basic statistics (correlation, skewness), data normalization (re-scaling)…

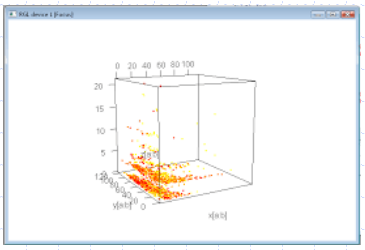
Data normalization

Rescaling: brings values into [0,1]
Scaling to unit length: dividing by magnitude
Standard score: assumes a normal distribution

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

$$x' = \frac{x}{||x||}$$

$$x' = \frac{x - \mu}{\sigma}$$



© Artur Garcez

Choose two methods for comparison

Popular methods:

Logistic Regression, Naïve Bayes,
Mixture of Gaussians, Hidden Markov Models,
Decision Trees, Random Forests
K-nearest neighbours (unsupervised)
Principal Component Analysis (unsupervised)

A note on sequence learning: time-series data can be rewarding to analyse but tricky to compare fairly with *sliding window* method

A note on combinations (e.g. PCA + regression) and variations (e.g. gradient boosting as variation of RFs): very good but only after the basics is done!

© Artur Garcez

Evaluation Methodology for Comparison: Generalisation

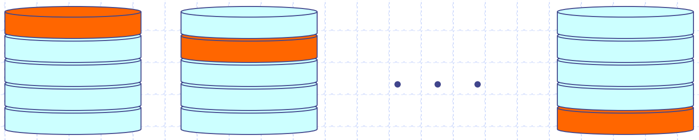
- We expect g (our approximation of f) to produce reasonable results for examples not seen during training, i.e. we expect the learning algorithm to *generalise* to unseen cases.
- We try to minimise a training set error (or maximise log likelihoods on the training data) as a proxy for minimising the model's generalisation error (assuming that training and test data come from the same distribution)

Estimating Generalisation Error

- Partition the set of examples into a training set and a validation/test set.
- The validation/test set is not seen by the model during training.
- The validation/test set error is an estimate of the generalisation error.
- If the above process is repeated for different validation/test sets then your confidence on the estimate will improve.
- Cross-validation or bootstrapping offer a systematic way of repeating this process.

n-fold Cross-Validation

- Divide the set of examples E into n subsets E_1, E_2, … , E_n
- For each E_i ($1 \leq i \leq n$), train a model with $E - E_i$ and test it (do not change parameters) with E_i
- Calculate the average validation/test set error



Note: Leaving m out = (|E| / m)-fold cross validation.

Bootstrapping

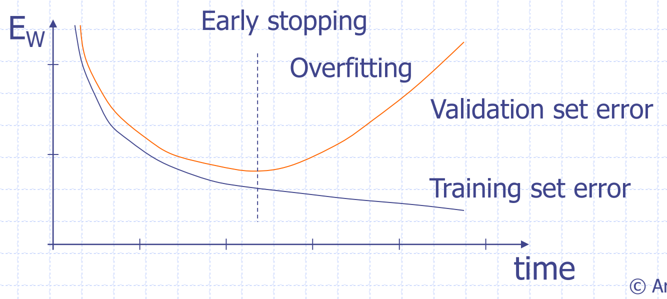
- Create k (pseudo) sets of examples E_1, E_2,… , E_k by randomly selecting |E| elements (with replacement) from E
- For each E_i ($1 \leq i \leq k$), train a model with $E - E_i$ and test it with E_i
- Calculate the average test set error; notice that the term is being used loosely, it is more precisely an averaged validation a.k.a. dev set (for development set) error

Model Selection

- Cross-validation can also be used for model selection…
- Split examples into training and test set (e.g. 70% vs. 30% or 90% vs. 10%)
- Apply cross-validation on training set
- Use averaged validation errors to choose each model's set of hyper-parameters
- Calculate test set error of chosen models for a comparison of the ML methods

Stopping criteria

- Training set error below 10%
- A fixed amount of time has passed
- No improvement over time
- Early stopping or a combination of the above



Example

Model selection vs ensembles and grid search

Polynomial curve fitting example: parameters **w** (the coefficients of the polynomial) and hyper-parameter M (the degree of polynomial)

Model selection: estimate **w** for a range of values for M (e.g. M=1,2,…,20) and select the model (value of M) with the highest accuracy

Ensemble method: average the result of a number of models (i.e. combining multiple values of M)

Example (cont.)

Grid search: if using regularization, there is also the regularization parameter λ

M\λ	**0.1**	**0.2**	**0.3**	**0.4**
1				
2				
3				

ML paradox: ensemble methods tend to achieve higher accuracy than their constituent (simpler) models

Evaluation metrics

Classification accuracy or mean-squared error
(RMSE, NMSR, etc.) Acc = 1 – NMSE

Average training time (given a machine)

Confusion matrix: true positives (TP), false positives (FP), true negatives (TN), false negatives (FN)

F measure and ROC curves…

n=165	Predicted: NO	Predicted: YES	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Other common terminology (not used by the ML expert):
sensitivity/specificity
precision/recall

© Artur Garcez

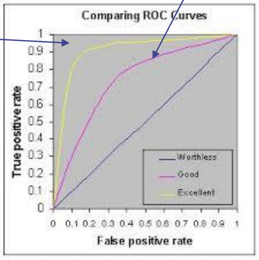
F measure and ROC curves

F1 score = 2TP/(2TP + FP + FN)
(better than accuracy e.g. for unbalanced datasets; ranges from 0 to 1 (best), used a lot in image analysis)

Receiver Operating Characteristic: false positive rate (FP/N) vs. true positive rate (TP/P)
To compare multiple confusion matrices (binary classifiers)

Each point given by a confusion matrix

Best classification is at top left



AUC (area under the ROC curve) = probability that a classifier will rank a randomly chosen positive instance higher than a randomly chosen negative one

© Artur Garcez

Are you a Bayesian or Frequentist?

If we only tossed a coin once and got *heads*…

Is p(*heads*)=0.5 or p(*heads*)=1?

Is it reasonable to give a single answer?

- If we don't have much data, we are unsure about p.
- Our computations will work much better if we take this uncertainty into account.

The Bayesian framework

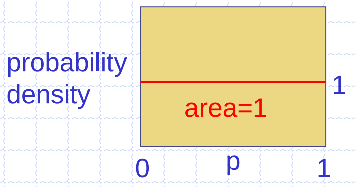
It assumes that a prior distribution always exists.

The prior may be very vague

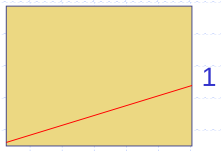
When we see some data, we combine our prior distribution with a likelihood function to get a posterior distribution (and keep applying Bayes' theorem iteratively)

The Bayesian framework (cont.)

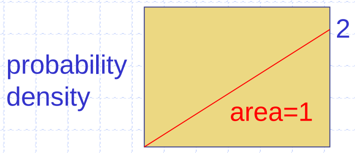
- Start with a prior distribution over p. In this case we used a uniform distribution



- Multiply the prior probability of each parameter value by the probability of observing *heads* given that value

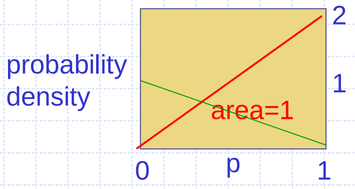


- Then scale up all of the probability densities so that their integral comes to 1. This gives the posterior distribution.

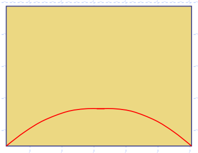


Let’s do it again:

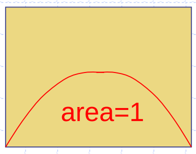
- Start with a prior distribution over p.
- Suppose you get tails now.



- Multiply the prior of each parameter value by the probability of observing tails given that value.

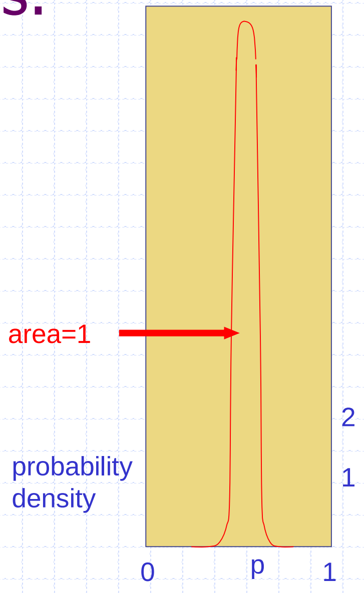


- Then re-normalize to get the posterior distribution.



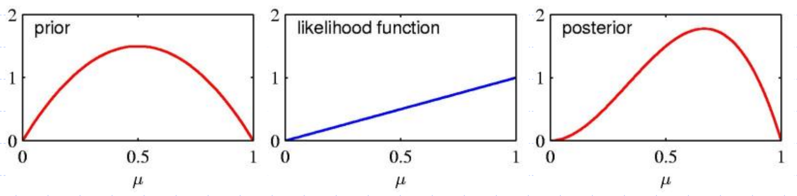
Let’s do it 100 times:

- After 53 heads and 47 tails we get a very sensible posterior distribution that has its peak at 0.53 (having assumed a uniform prior)



Bayesian approach

Prior x Likelihood *is proportional to* Posterior



One step of sequential Bayesian inference: the prior is given by a Beta distribution with parameters $a = 2$, $b = 2$ in this case, and the likelihood function corresponds to a single observation of *heads*. The posterior is a beta distribution with parameters $a = 3$, $b = 2$.
The parameters a and b of the Beta distribution are called *hyper-parameters* because they control the distribution of the parameter μ.

© Artur Garcez

Density estimation

We would like to model the probability distribution $p(x)$ of a random variable x given a finite set $D=\{x_1,\ldots,x_N\}$ of observations (data points).

The problem of density estimation is fundamentally ill-posed because there are infinitely many probability distributions that could have given rise to the finite data set observed…

A sampling assumption

- Assume that the training examples are drawn independently from the set of all possible examples.
- Assume that each time a training example is drawn, it comes from an identical distribution (i.i.d)
- Assume that the test examples are drawn in exactly the same way: i.i.d. and from the same distribution as the training data.
- These assumptions make it very unlikely that a strong regularity in the training data will be absent in the test data.

Some Loss Functions

- Squared difference between actual and target real-valued outputs (squared error).
- Number of classification errors:
  - Problematic for optimization because the derivative is not smooth.
- Negative log probability assigned to the correct answer:
  - This is usually the right function to use!
  - In some cases it is the same as squared error (e.g. regression with Gaussian output noise); in other cases it is very different…
- Cross-entropy error: use it with multi-class discrete classification tasks, i.e. one-hot target vectors

Loss function example

```
ML output     | Target
------------------------------
0.1  0.3  0.6  |  0  0  1
0.2  0.6  0.2  |  0  1  0
0.3  0.4  0.3  |  1  0  0
```

ML system gets the first two examples right but not the third example

Classification error = 1/3

Squared error for first example = (0.1 - 0)^2 + (0.3 - 0)^2 + (0.6 - 1)^2 = 0.26

MSE = (0.26 + 0.24 + 0.74)/3

Least squares = 1/2 (0.26 + 0.24 + 0.74)

Notice that the error values above are different for first and second examples, despite the same answer 0.6

Cross-entropy = $-\sum_{i=0}^{n} \ln(o_i) * t_i$ = - (ln(0.1)*0 + ln(0.3)*0 + ln(0.6)*1) = 0.51 for first example. Also, 0.51 for second example!

Mean cross entropy = 0.74

Maximizing Log Likelihood

Suppose that the probability of a coin landing heads (p(x=h)=*μ*) is not necessarily the same as that of it landing tails p(x=t)=1-*μ*

We can construct a likelihood function on the assumption that the observations are drawn independently (iid):

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}$$

We can estimate a value for *μ* by maximizing the log of the likelihood function; if we set the derivative of log p(*D/μ*) with respect to *μ* equal to zero, we obtain the maximum likelihood estimator, where m is the number of *heads*:

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N} x_n = \frac{m}{N}$$

Choosing a prior

Without a prior, suppose we flip a coin 3 times and observe 3 *heads.* Thus, *μML=1*

If we choose a prior to be proportional to powers of *μ* and (1 – *μ*) then the posterior distribution, which is proportional to the product of the prior by the likelihood function, will have the same functional form as the prior.

Bayes Theorem (revisited)

joint probability

conditional probability

$$p(D)\,p(W\mid D) = p(D,W) = p(W)\,p(D\mid W)$$

Prior probability of parameter vector W

Probability of observed data D given W

$$p(W\mid D) = \frac{p(W)\;p(D\mid W)}{p(D)}$$

Posterior probability of W given D

$$\int_W p(W)\,p(D\mid W)$$

Why do we maximize sums of log probabilities?

- Assuming that the errors on different training examples *c* are independent, we want to maximize the product of the probabilities of the training examples…

$$p(D\mid W) = \prod_c p(d_c\mid W)$$

- Because the log function is monotonic, it does not change where the maxima are. So we can maximize sums of log probabilities:

$$\log p(D\mid W) = \sum_c \log p(d_c\mid W)$$

- Recall log(xy) = log(x) + log(y)
  log(x/y) = log(x) – log(y)

Machine Learning trick 1:

- To avoid computing the posterior probabilities of all parameters W…
  - Start with a random W and then adjust it in the direction that improves p(W|D)
- It is easier to work in the log domain. To maximize p(W|D), minimize the *Cost* :

$$p(W \mid D) = p(W)\, p(D \mid W) / p(D)$$

$$Cost = -\log p(W \mid D) = -\log p(W) - \log p(D \mid W) + \log p(D)$$

Using log probabilities helps keep the values low by replacing the product of the probabilities by the sum of the log probabilities

Machine Learning trick 2:

- Completely ignore the prior over W:
  - This is equivalent to giving all possible W the same prior probability density
- Then all we need to do is maximize:

$$\log p(D \mid W) = \sum_c \log p(D_c \mid W)$$

- This is called maximum likelihood learning. It is widely used for fitting models in statistics…

So, Maximum Likelihood…

We want to determine the values of μ and σ that maximize the log likelihood:

$$\ln p\left(\mathbf{x}|\mu,\sigma^2\right) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi)$$

Maximizing w.r.t. μ:

$$\mu_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}x_n$$

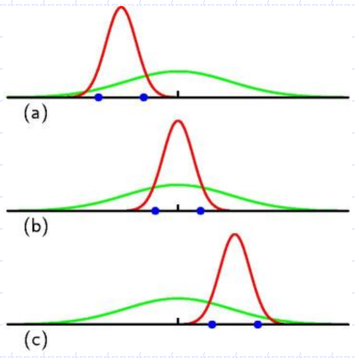
Maximizing w.r.t. σ^2:

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}(x_n-\mu_{\mathrm{ML}})^2$$

Properties of μ_{ML} and σ^2_{ML}

$$\mathbb{E}[\mu_{\mathrm{ML}}] = \mu$$

$$\mathbb{E}[\sigma^2_{\mathrm{ML}}] = \left(\frac{N-1}{N}\right)\sigma^2$$



On average, maximum likelihood will obtain the correct mean, but will underestimate the variance by a factor N-1/N.

Trading off the goodness of fit against the complexity of the model

- You can only expect a model to generalize well if it explains the data surprisingly well given the complexity of the model.
- If the model has as many degrees of freedom as the data, it can fit the data perfectly but so what?
- Learning theory seeks to measure the model complexity and how to control it to optimize generalization, c.f. Les Valiant's PAC learnability.
- The main principle is known as Ockham's razor (William of Ockham (c. 1287–1347) was an English friar, philosopher and theologian): Among competing hypotheses, under uncertainty, the one with the fewest assumptions should be selected.