Short course

A vademecum of statistical pattern recognition and machine learning

Classification

Massimo Piccardi
University of Technology, Sydney, Australia

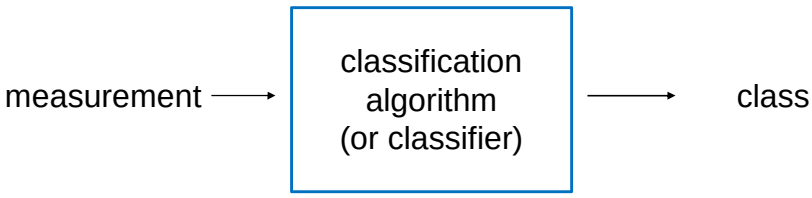
Agenda

- Classification
- Classification accuracy
- Binary classification test
- ROC curve
- Classification accuracy for multiple classes
- Maximum-a-posteriori decision rule
- Classification by Bayes' theorem
- Bayes risk
- Minimum risk decision rule
- Generative vs discriminative approaches
- Learning a probabilistic classifier

Classification

- Let us assume that we are given a set of **classes** (or **categories**)
- Assume that we are also given an object that can be categorised into one of the classes. The assignment of the object to a class is called **classification**
- Classification is performed by obtaining a **measurement** of the object and applying a classification algorithm

Classification



- The **class** takes *symbolic* values: “good”, “bad”, “Parisian”, “trustworthy”, “cat”, “dog”…
- The **measurement** can be multiple (e.g., age, income, occupation), and both numerical or symbolic

Classification

- Let us assume both the class and the measurement to be random variables, **y** and **x**, respectively
- We aim to provide a *prediction* (aka *assignment*), **ỹ**, that is as close as possible to the "ground-truth" class, y
- ỹ is another random variable that can be obtained from x through a deterministic algorithm, **f(x)**:

$$\tilde{y} = f(x)$$ *classifier*

Statistical classification

- In statistical classification, the fundamental quantity used for classification is:

$$p(y \mid x)$$

- Deriving p(y|x) is (properly) called **inference**. Often, it is confused/assimilated with classification itself, but it is only a part of it

Parameters

- Where are the parameters? Like any distribution, p(y|x) can be modelled through a chosen family and a set of parameters, as either a point-based or Bayesian estimate:

$$p(y \mid x, \theta^*)$$

$$p(y \mid x) \approx \int_\theta p(y \mid x, \theta) p(\theta \mid Y, X) d\theta$$

- In the following, we omit the explicit dependence on the parameters. We'll address parameters and learning later

Classification accuracy

- Before we address how to build a classification algorithm, let us discuss how we will measure its accuracy
- Let us adopt more prominent symbols for y and ỹ:
  - **T**, the "true class" of the given object (aka *ground truth*)
  - **A**, the class to which the classification algorithm assigns it
- We wish that classification will be as accurate as possible, i.e. A be the same as T most of the times (if not all)

Binary classification test

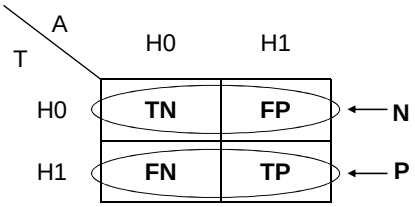
- Let us start with *binary classification*, the case where there are only two classes
- In addition, in a binary classification *test*, the two classes are asymmetric, of the type “it has the property/it hasn’t”
- Classes are often called the two hypotheses, **H1** and **H0**
- A binary classification test is often called a *detector*

Binary classification test

- Consistently with the idea of a test, based on the **true** labels, the samples can be divided into:
  - Positives (P) and Negatives (N)
- After classification, based on both the **true and assigned** labels, the same samples can be categorised as:
  - True Positives (TP), True Negatives (TN), False Positives (FP), False Negatives (FN)
- Obviously, TP and TN are desirable, FP and FN are not

Contingency table

- Using symbols P, N, TP, TN, FP, FN to mean the corresponding numbers, the *contingency table* for the binary classification test is:



- Please note that the following holds:
  - Total number of samples = **P + N** = **TP + TN + FP + FN**
  - **P = TP + FN** and **N = TN + FP**

Probabilities in a binary test

The contingency table can be used to estimate probabilities in a maximum-likelihood sense. For instance:

- **joint probability p(A = H1, T = H1)** is given by the number of true positives divided the total number of samples:
  - p(A = H1, T = H1) = TP/(P + N)
- **marginal probability p(T = H1)** can be obtained either directly as P/(P + N) or by marginalising A in the above:
  - p(T = H1) = p(A = H1, T = H1) + p(A = H0, T = H1) =
    = TP/(P + N) + FN/(P + N) = P/(P + N)
- **conditional probability p(A = H1 | T = H0)** is given by:
  - p(A = H1 | T = H0) = FP/N

Binary classification test

- Accuracy of a binary classification test is typically expressed by two numbers:
  - The rate of true H1 classified as H1 (**detection rate**, DR):
    p(A=H1 | T=H1) = TP/P = TP/(FN+TP)
  - The rate of true H0 classified as H1 (**false alarm rate**, FAR):
    p(A=H1 | T=H0) = FP/N = FP/(TN+FP)
- Similar rates take different names in different fields:
  - Signal Processing: the above
  - Information Theory: Recall ≡ DR;
    Precision ≡ TP/(TP+FP) = p(T=H1 | A=H1)
  - Medicine: Sensitivity ≡ DR; Specificity ≡ 1 - FAR
- FP ≡ *Type 1*, or *alpha*, error; FN ≡ *Type 2*, or *beta*, error

Binary classification test: example

Example:

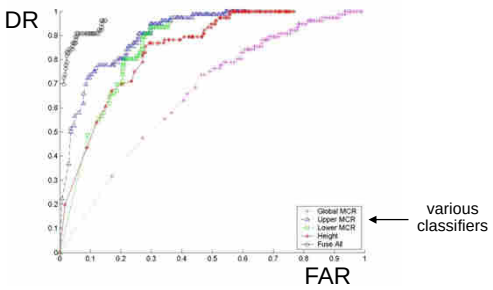
- 100 samples, values as in table
- Marginal probabilities of T:
  - p_{H0} = p(T=H0) = 88/100 = .88
  - p_{H1} = p(T=H1) = 12/100 = .12
- FAR and DR:
  - p_{FA} = FAR = 8/88 = 9%
  - p_D = DR = 10/12 = 83%

T \ A	H0	H1
H0	80	8
H1	2	10

- All the joint probabilities can be built from the above with Bayes' theorem, or derived directly from the table
- The **overall accuracy** is the number of correctly classified samples divided the total number of samples

ROC (receiver operating characteristic) curve

- Often the parameters of a classifier can be changed to achieve a better DR at the expense of a worse FAR, or vice versa
- The plot of (FAR, DR) values is a *ROC* curve. The area under it is known as the *area under the curve* (AUC)



Classification accuracy for multiple classes

- *Confusion* matrix (example with 4 classes):

T \ A	c_1	c_2	c_3	c_4
c_1	7	0	3	0
c_2	2	30	2	6
c_3	0	0	34	6
c_4	0	0	0	35

- The confusion matrix provides all information about accuracy, for each class and as a whole
  - what is the overall accuracy above? and the *overall error rate* (1 – accuracy)?
  - what is the detection rate/recall for class C3? and its false alarm rate?

Accuracy and error rate

- The overall **accuracy** and **error rate** (1 – accuracy) can be obtained as a weighted average over all the classes:

$$accuracy = \sum_{j=1}^{M} p\left(A = c_j \mid T = c_j\right) p\left(T = c_j\right) = \sum_{j=1}^{M} p\left(A = c_j, T = c_j\right)$$

$$error\ rate = \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} p\left(A = c_i \mid T = c_j\right) p\left(T = c_j\right) = \sum_{j=1}^{M} \sum_{i=1, i \neq j}^{M} p\left(A = c_i, T = c_j\right)$$

* See the Notes Page *

Empirical vs theoretical rates

- Alternatively the accuracy and error rate can be written as expectations of indicator functions. For instance, for the error rate let us define $\Delta(T,A)$ evaluating to 1 if $A \neq T$ and 0 otherwise:

$$error\ rate = \sum_{i=1}^{M} \sum_{j=1}^{M} \Delta(T, A)\, p\left(A = c_i, T = c_j\right) = E[\Delta(T, A)]$$

- If we use empirical distributions (i.e., estimates from a sample set) in the expectations, we talk about *empirical rates*. Using exact distributions returns the *theoretical rates*

* Notes Page *

Validation

- Most classifiers need a training stage, during which the parameters of the classifier are optimised based on a given set of samples (*training set*)
- If the accuracy is measured on the same set, the risk is that of obtaining an overly optimistic value
- The *test set* must be different from the training set; given a set of samples, this may be split into a training set (e.g., 66%) and a test set (e.g., 34%). No hard rule for the split. The measured accuracy is called **cross-validation accuracy**

Validation

- Repeating cross-validation several times and averaging the resulting accuracy may lead to more realistic estimates which depend less on the specific training and test sets
- In *N-fold cross validation*, a given data set is divided into N "slices": the data in N-1 are used for training and the last for testing; the procedure is repeated N times, and accuracy averaged

Minimum error rate as assignment criterion

- Based on a measurement, x, of an object, we want to *optimally assign* it to one of M classes, c_j, j=1..M
- We need to choose an assignment criterion: let us choose the **minimum error rate**
- We assume that we know p(y|x) (and p(x), and related marginals and conditionals) or that we can estimate them from a set of labelled samples
- We need a **decision rule**, to complete p(y|x) into a classifier f(x)

Maximum-a-posteriori decision rule

- It can be proven that the minimum error rate is obtained by applying this decision rule to each sample:

$$\tilde{y} = f(x) = \arg\max_{j=1\ldots M} p(y = c_j \mid x)$$

- The above criterion is called the **maximum-a-posteriori (MAP)** decision rule
- In the next slides, instead of $p(y = c_j \mid x)$ we will use short-hand notation $p(c_j \mid x)$

Example

- Let us say that we need to classify a piece of fruit as one of classes {apple, pear, avocado*} based on a measurement of size and weight
- For a given measurement x, we have come to know that the probabilities are:

  p(y = apple | x) = 0.2

  p(y = pear | x) = 0.5

  p(y = avocado | x) = 0.3
- According to the MAP rule, we'll classify this object as a pear

* avocados are classified as fruit despite their primary use in savoury food; in that, they share the fate of tomatoes

Classification by Bayes' theorem

- Instead of using $p(c_j \mid x)$ directly, applying Bayes' theorem allows us to write:

$$p(x, c_j) = p(x \mid c_j)p(c_j) = p(c_j \mid x)p(x)$$

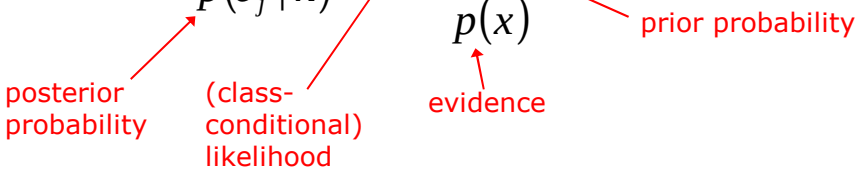
$$\rightarrow \quad p(c_j \mid x) = \frac{p(x \mid c_j)p(c_j)}{p(x)}$$

- Why is this theorem so fundamental? Because it can prove simpler to learn the probabilities on the right separately

Classification by Bayes' theorem

- This formula is so canonical that its quantities have names:

$$p(c_j \mid x) = \frac{p(x \mid c_j)p(c_j)}{p(x)}$$



- Note that the same class label is obtained if we maximise either **$p(c_j|x)$** (*posterior*) or **$p(c_j,x)$** (*joint*); p(x) (*evidence*) doesn't count in the assignment

Notes

- Why names *prior* and *posterior*?
  - **$p(c_j)$** is the probability of the class *before* (i.e. prior to) the measurement;
  - **$p(c_j \mid x)$** is the probability of the class *after* (i.e. a posteriori of) the measurement
- As expected, the $p(c_j|x)$ add up to 1 as shown by:

$$p(c_j \mid x) = \frac{p(x \mid c_j)p(c_j)}{p(x)} = \frac{p(x \mid c_j)p(c_j)}{\sum_j p(x \mid c_j)p(c_j)}$$

evidence of x

Example

- You hear an engine at night: is that a car (C) or a motorbike (M)?
- Let us assume that we measured the prior probabilities:
  p(C) = 0.90, p(M) = 0.10
- Let us also assume that we can model likelihoods p(noise | C), p(noise | M) from aural samples of cars and motorbikes
- First audio clip : p(noise | M) = 0.60, p(noise | C) = 0.20
  - p(M | noise) $\propto$ p(noise | M) * p (M) = 0.60 * 0.10 = 0.06
  - p(C | noise) $\propto$ p(noise | C) * p (C) = 0.20 * 0.90 = 0.18 $\rightarrow$ stick with car
- Second audio clip : p(noise | M) = 1.2; p(noise | C) = 0.10
  - p(M | noise) $\propto$ 1.2 * 0.1 = 0.12
  - p(C | noise) $\propto$ 0.10 * 0.90 = 0.09 $\rightarrow$ opt for bike

Bayes risk

- In many cases, the *cost*, or *loss*, associated with a misclassification varies with the combination of the true and assigned classes
- In such cases, errors are weighted by a loss function, $\boldsymbol{\Delta(y = c_i, \tilde{y} = c_j)}$ (or Δ_{ij} for short), that is the cost of misclassifying class c_i (true class) as c_j (assigned class)

Examples of loss functions

- In the simplest case, every type of error is given equal weight (e.g., 1). We can write the loss function (known as *zero-one loss*) as:

  $\Delta(T,A)$ (1 if $A \neq T$, 0 otherwise)

- Classifying a defective manufactured part as good may carry a higher cost than classifying a good part as defective (asymmetry: $\Delta_{ij} \neq \Delta_{ji}$)
- Classifying a colour *red* as *green* may have a higher cost than classifying *red* as *orange* because green is more different than orange from red
- In certain cases, classes may have a natural numerical value (e.g. , the intensity levels of a pixel). In this case, Δ_{ij} could be set to the square error (SE): $\Delta_{ij} = (i - j)^2$

Bayes risk

- The *overall loss*, also known as *Bayes' risk*, *integrated risk, expected loss* etc can be written as:

$$Bayes'\ risk = \sum_{y}\sum_{\tilde{y}} \Delta(y,\tilde{y})p(y,\tilde{y}) = E[\Delta(y,\tilde{y})]$$

- In other words, it is the expected value of cost function $\Delta(y,\tilde{y})$
- The overall loss is more commonly defined in terms of variables y and x, but this definition suits our review
- For the zero-one loss, it is the same as the error rate

* Notes Page *

Minimum risk decision rule

- It can be proven that the minimum expected loss is achieved by the following decision rule:

$$\tilde{y} = f(x) = \arg\min_{u} \left(\sum_{y} \Delta(y,u)\, p(y \mid x) \right)$$

- The rule says: for every class u (auxiliary variable), compute the *expected loss from this assignment* and choose the class causing the minimum expected loss
- This decision rule is called the **minimum risk decision rule**

Example

- A hypothetical loss function for our example:

$\Delta(y,u) =$

true / assigned	*apple*	*pear*	*avocado*
apple	0	2	2
pear	0.2	0	0.3
avocado	0.4	1	0

Example

- Expected loss for choosing u = apple:

  Δ(y = apple, u = apple) p(y = apple | x) = 0 * 0.2 +

  + Δ(y = pear, u = apple) p(y = pear | x) = 0.2 * 0.5 +

  + Δ(y = avocado, u = apple) p(y = avocado | x) = 0.4 * 0.3 =

  = 0.22

- Similarly, we compute the loss for u = pear as equal to 0.7; the loss for u = avocado is 0.55
- Based on the minimum risk decision rule, *we assign the piece of fruit to apple* with a large margin

Generative vs discriminative classifiers

- Classification by Bayes' theorem estimates p(x|y) and p(y) explicitly, and obtains p(y|x) via Bayes' theorem: this is known as a *generative* approach
- The reason for the *generative* name is that one can obtain (i.e. generate) samples of x by sampling either p(x|y) or p(x)
- As we have said many times, a model like p(y|x) alone does not contain any probabilistic information on x (and therefore cannot be sampled)

Discriminative approaches

- In *discriminative* probabilistic approaches, p(y|x) is estimated directly. A typical model is the *logistic regression classifier*
- One can also renounce probability altogether and build a *discriminant function*, f(x), without reference to p(y|x)
- Discriminative approaches in general can focus on class boundaries rather than entire densities: they may prove more accurate. As a most notable case, in a later lecture we'll present the *support vector machine*

Classification by regression

- In probabilistic approaches to classification, we model p(y|x)
- A simpler approach is to just put the classes in correspondence with regions of real numbers (or $\Re^n$) by building a **discriminant function**, f(x), that maps the measurement to the regions
- A simple example is a binary classifier for classes {A, B} that associates class A with $f(x) \geq 0$ and class B with $f(x) < 0$
- During training, the classes are given numerical values such as (+1, -1) or (+1, 0) or else; it is, in a way, classification by regression
- Examples of classifiers following this approach are linear classifiers (including the linear support vector machine), quadratic classifiers, AdaBoost and many others

Classification approaches: summary

generative $\tilde{y} = \arg\max_y \left(p(x \mid y) p(y) \right)$ (1)

discriminative $\tilde{y} = \arg\max_y \, p(y \mid x)$ (2)

discriminant $\tilde{y} = f(x)$ (3)

- If a loss function different from the zero-one loss needs to be used, the decision rule for (1) and (2) has to be as with the Bayes risk. For (3), the loss function has to be suitably accounted for inside f(x)

Learning

- How to learn a probabilistic classifier? Let us start by choosing to use Bayes' theorem to provide classification:

$$\tilde{y} = \arg\max_y \left(p(x \mid y) p(y) \right)$$

- Let us now make the parameters of the involved distributions explicit:

$$\tilde{y} = \arg\max_y \left(p(x \mid y, \theta_{xy}) p(y \mid \theta_y) \right)$$

- θ_{xy} represents the parameters of the likelihood of class y and θ_y represents the parameters of the prior

Learning

- Distribution $p(x|y,\theta_{xy})$ can be, for instance, a multivariate Gaussian. In that case, θ_{xy} represents its mean and covariance. Or it can be any other distribution in x
- There will be one likelihood *per class* (with M classes, M likelihoods, each with its own sets of parameters, θ_{xy})
- Distribution $p(y|\theta_y)$ is certainly a discrete distribution since the class variable, y, is discrete. Its parameters, θ_y, are the M probability values over the simplex
- For clarity, there is only one prior distribution overall
- θ notes the union set of all θ_{xy} and θ_y

Objective function: likelihood

- Given a training set of N pairs of measurements and class labels, $\{X,Y\} = \{x_i,y_i\}$, i = 1…N, we choose to train the classifier by maximising the following objective function:

$$\theta^* = \arg\max_{\theta}\left(p\left(X \mid Y,\theta_{xy}\right)p\left(Y \mid \theta_y\right)\right)$$

- This objective function is called the **(joint) likelihood of {x,y}** ($p(X,Y|\theta)$ for short) and is the same function used for the inference. In this way, we choose the parameters that maximise our inference function over the given training set

Maximum likelihood

- Further, we make it explicit that the pairs are i.i.d. given the parameters:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{N} \left(p\left(x_i \mid y_i, \theta_{xy}\right) p\left(y_i \mid \theta_y\right) \right)$$

- Let us then move to logarithmic scale:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \left(\ln p\left(x_i \mid y_i, \theta_{xy}\right) + \ln p\left(y_i \mid \theta_y\right) \right)$$

Maximum likelihood

- To maximise the objective function we differentiate it in each parameter and equate to 0
- Now, the founding point: when we differentiate it in θ_{xy} for class, say, l, all other terms cancel:

$$\frac{\partial}{\partial \theta_{xy=l}} \sum_{i=1}^{N} \left(\ln p\left(x_i \mid y_i, \theta_{xy}\right) + \ln p\left(y_i \mid \theta_y\right) \right) =$$

$$= \frac{\partial}{\partial \theta_{xy=l}} \sum_{i=1}^{N_l} \left(\ln p\left(x_i \mid y_i = l, \theta_{xy=l}\right) \right)$$

- This is equivalent to maximising the likelihood for this class alone, using only the samples from that class

Maximum likelihood

- Likewise, when we differentiate it in θ_y, all non-relevant terms cancel:

$$\frac{\partial}{\partial \theta_y} \sum_{i=1}^{N} \left(\ln p\left(x_i \mid y_i, \theta_{xy}\right) + \ln p\left(y_i \mid \theta_y\right) \right) =$$

$$= \frac{\partial}{\partial \theta_y} \sum_{i=1}^{N} \left(\ln p\left(y_i \mid \theta_y\right) \right)$$

- This is equivalent to maximising the likelihood of a common discrete distribution!

Discriminative training

- Conversely, let us now assume that we *do not want* to use Bayes' theorem to provide classification. We use directly:

$$\tilde{y} = \arg\max_{y} p\left(y \mid x, \theta\right)$$

where we have made the parameters explicit

- Given the same training set as before, it is now desirable to train θ by maximising this inference function over the training set:

$$\theta^* = \arg\max_{\theta} p\left(Y \mid X, \theta\right)$$

Objective function: conditional likelihood

- This objective function is known as the **conditional likelihood** and the corresponding maximisation as *maximum conditional likelihood estimation* (MCLE)
- By applying the i.i.d. assumption:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{N} p(y_i \mid x_i, \theta)$$

Maximum conditional likelihood

- Just to illustrate our point, let us assume that the distribution factorises into the same terms as before:

$$\theta^* = \arg\max_{\theta} \prod_{i=1}^{N} \left(\frac{p(x_i \mid y_i, \theta) p(y_i \mid \theta)}{p(x_i \mid \theta)} \right) =$$

$$= \arg\max_{\theta} \prod_{i=1}^{N} \left(\frac{p(x_i \mid y_i, \theta_{xy}) p(y_i \mid \theta_y)}{\sum_y p(x_i \mid y, \theta_{xy}) p(y \mid \theta_y)} \right)$$

the second equality coming from Bayes' theorem and marginalisation. It can be immediately seen that this distribution is significantly more complex *due to the denominator*

Maximum conditional likelihood

- Let us move to logarithmic scale:

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \begin{pmatrix} \ln p(x_i \mid y_i, \theta_{xy}) + \ln p(y_i \mid \theta_y) \\ -\ln \sum_y p(x_i \mid y, \theta_{xy}) p(y \mid \theta_y) \end{pmatrix}$$

- When we differentiate in, say, $\theta_{xy = l}$, these parameters appear in all N addenda!, not just those with the samples of class l
- In addition, the parameters of the other classes, $\theta_{xy \neq l}$, appear in the same equations, forcing us to a *simultaneous optimisation of all parameters*

Learning: summary

- Maximising the *joint likelihood* is as simple as maximising the likelihood of each class separately
- Maximising the *conditional likelihood* requires maximising in all the parameters *simultaneously*, over samples from all classes *at once*, and the expression is more complicated than the separate likelihoods
- However, it is paramount to say that the advantage of training p(y|x) directly is that we are not required to fully train likelihoods p(x) or p(x|y). Maximum conditional likelihood becomes a suitable objective for the *exponential family of distributions*. This is the topic of a following lecture

Possible further readings

- Bishop's PRML book, chapter 4.2 Probabilistic Generative Models and 4.3 Probabilistic Discriminative Models
- Andrew Ng, Michael Jordan, "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes", NIPS 2001, pp. 841-848
- Tony Jebara, "Machine Learning: Discriminative and Generative", Springer, 2003

