4 Classification

4.1 Practical Exercises

In this session, we take into account again the Iris dataset. This time we are more interested in the discrimination of the sample class, i.e., either Setosa, Versicolor or Virginica.

We do not have any metric over the space of the classes, i.e., it is not possible to order them. In this case one could consider regression techniques. Instead, we consider classification techniques. Let us start with discriminating between Setosa and non-Setosa basing on the sepal length and width.

```
load iris_dataset.mat;
x = zscore(irisInputs([1 2],:)');
t = irisTargets(1,:)';
gplotmatrix(x,[],t);
```

We will consider a set of methods which can be used when we want to discriminate between two classes, in the specific:

- Single perceptron;
- Logistic regression;
- Naive Bayes;
- K-nearest neighbour.

4.1.1 Perceptron

At first, let us perform a classification with a perceptron classifier:

- Hypothesis space: $y(\mathbf{x}_n) = sgn(\mathbf{w}^T\mathbf{x}_n) = sgn(w_0 + x_{n1}w_1 + x_{n2}w_2)$,
- Loss measure: Distance of misclassified points $L_P(\mathbf{w}) = -\sum_{n\in\mathcal{M}} \mathbf{w}^T\mathbf{x}_n C_n$,
- Optimization method: Online Gradient Descent,

where $sgn(\cdot)$ is the sign function.

In MATLAB the perceptron is classified as a simple neural network with a single layer, thus can be found in the *Neural Network Toolbox*.

```
net = perceptron;
net = train(net,x',t');
```

The recap of the perceptron provides you information about the evolution of the objective function reduction during the epochs and some performance measures. More specifically, to evaluate the performance of the chosen method we need to consider the *confusion matrix* which tells us the number of points which have been correctly classified and those which have been misclassified:

	Actual Class: 1	Actual Class: 0
Predicted Class: 1	tp	fp
Predicted Class: 0	fn	tn

By basing on this matrix we can evaluate:

- Accuracy: $Acc = \frac{tp+tn}{N}$ fraction of the samples correctly classified in the dataset;
- Precision: $Pre = \frac{tp}{tp+fp}$ fraction of samples correctly classified in the positive class among the ones classified in the positive class;
- Recall: $Rec = \frac{tp}{tp+fn}$ fraction of samples correctly classified in the positive class among the ones belonging to the positive class;
- F1 score: $F1 = \frac{2 \cdot Pre \cdot Rec}{Pre+Rec}$ harmonic mean of the precision and recall;

where tn is the number of true negatives, fp is the number of false positives, fn are the false negatives and tn are the true negatives. The higher these figures of merits the better the algorithm is performing.

In this case, we are able to correctly classify all the points we have and thus all the previous figures of merits are equal to 100% (or 1 equivalently).

If we want to implement the perceptron by hand we could do:

```
n_samples = size(x,1);
perc_t = t;
perc_t(perc_t == 0) = -1;
ind = randperm(n_samples);

perc_t = perc_t(ind);
x_perc = x(ind,:);
```

```
8
9  w = ones(1,3);
10 for jj = 1:10
11     for ii = 1:n_samples
12         if sign(w * [1 x_perc(ii,:)]') ~= perc_t(ii)
13             w = w + [1 x_perc(ii,:)] * perc_t(ii);
14         end
15     end
16 end
```

where we converted the output to be $\{-1, 1\}$ and we permuted the samples not to process all the samples from a single class consecutively. Remember that while the Neural Network toolbox is able to stop the optimization process by basing on the classification error, here we need to perform a large enough number of stochastic gradient updates s.t. it is able to correctly classify all the samples.

If we plot $x_2 = f(x_1) = -\frac{w_1 x_1 + w_0}{w_2}$ we are able to visualize the separation surface we learned in the input space:

```
1 figure();
2 gscatter(x(:,1),x(:,2),perc_t);
3 hold on;
4 s = -2:0.01:2.5;
5 plot(s, -(s * net.IW{1}(1) + net.b{1} ) / net.IW{1}(2),'k');
6 plot(s, -(s * w(2) + w(1) ) / w(3),'r');
```

4.1.2 Logistic regression

Let us change the methods for the classification task and consider Logistic regression with two classes:

- Hypothesis space: $y_n = y(x_n) = \sigma(w_0 + x_{n1}w_1 + x_{n2}w_2)$,
- Loss measure: MLE $L(\mathbf{w}) = -\sum_n [C_n \ln y_n + (1 - C_n) \ln(1 - y_n)]$,
- Optimization method: Gradient Descent,

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

In MATLAB :

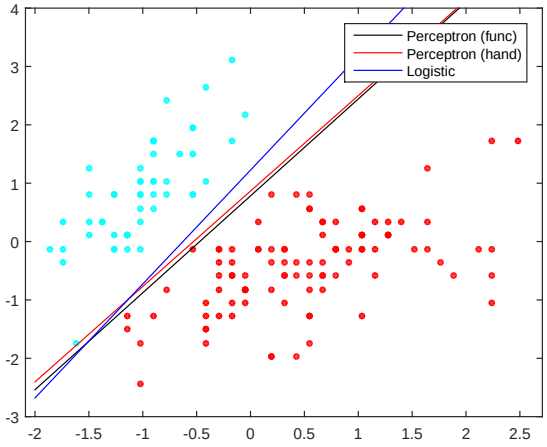
```
1 t = t+1;
2 [B, dev, stats] = mnrfit(x,t);
```

where we add one to the targets `t` since `mnrfit` considers categorical classes corresponding to positive integers. In the structure `stats` we have also information about the statistical significance of the learned parameters since if we perform the $logit(x) = \log(\frac{x}{1-x})$ transformation to the output we have:

$$logit(y_n) = w_0 + x_{n1}w_1 + x_{n2}w_2$$

thus we have the same statistical characterization of the parameters $\mathbf{w}$ as we had in the linear regression if we consider as output a specific transformation of the target.

In the classification case we have different methods for evaluating the goodness of fit of the model we trained. In the case of binary classification we identify a positive class, corresponding to sample that have $t = 1$, and a negative one, i.e., those samples having $t = 0$. Again we can consider as a performance measure the one we extract from the confusion matrix. We are able to find a separating hyperplane that correctly classifies all the given points.



With the function `mnrfit` it is even possible to handle multiple classes. In the case we want to discriminate among the three different iris classes we might do:

```
[t, ~] = find(irisTargets ~= 0);
[B_mul, dev_mul, stats_mul] = mnrfit(x,t);
```

Even in this case we can built a confusion matrix, which in this case is a 3×3 matrix. Here it is possible to define a generalization of the previously defined figure of merits, for instance the accuracy is:

$$Acc = \frac{\sum_{k \in C_1,\ldots,C_K} tc_k}{N}$$

where tc_k is the number of correctly classified points for class k.

4.1.3 Naive Bayes

Let us move to a generative model: Naive Bayes. Generative models have the purpose of modeling the joint pdf of the couple input/output $p(C_k, \mathbf{x})$, which allows us to generate also new data from what we learned, differently from discriminative models we are only interested in computing the probabilities that a given input is coming from a specific class $p(C_k|\mathbf{x})$, which is not sufficient to produce new samples.

In this case, the Naive Bayes method considers the naive assumption is that each input is conditionally (w.r.t. the class) independent from each other. If we consider the Bayes formula we have:

$$\begin{aligned}
p(C_k|\mathbf{x}) &= \frac{p(C_k)\, p(\mathbf{x}|C_k)}{p(\mathbf{x})} \\
&\propto p(x_1, \dots, x_M, C_k) \\
&= p(x_1|x_2, \dots, x_M, C_k)p(x_2, \dots, x_M, C_k) \\
&= p(x_1|x_2, \dots, x_M, C_k)p(x_2|x_3, \dots, x_M, C_k)p(x_3, \dots, x_n, C_k) \\
&= p(x_1|x_2, \dots, x_M, C_k)p(x_2|x_3, \dots, x_M, C_k) \dots p(x_{M-1}|x_M, C_k)p(x_M|C_k)p(C_k) \\
&= p(C_k) \prod_{j=1}^{M} p(x_j|C_k).
\end{aligned}$$

The decision function, which maximises the MAP probability, is the following:

$$y(\mathbf{x}) = \arg\max_k p(C_k) \prod_{j=1}^{M} p(x_j|C_k),$$

where as usual we do not consider the normalization factor $p(\mathbf{x})$.

In a specific case we have to define a prior distribution for the classes $P(C_k)\ \forall k$ and a distribution to compute the likelihood of the considered samples $p(x_j|C_k)\ \forall J,\ \forall k$. In the case of continuous variable one of the usual assumption is to use Gaussian distributions for each variable $p(x_j|C_k) = \mathcal{N}(x_j; \mu_{jk}, \sigma^2_{jk})$ and either a uniform prior $p(C_k) = \frac{1}{K}$ or a multinomial prior based on the samples proportions $P(C_k) = \frac{\sum_{i=1}^{N} I\{\mathbf{x}_n \in C_k\}}{N}$, where $I\{\cdot\}$ is the indicator function.

The complete model we consider is:

- Hypothesis space: $y_n = y(x_n) = \arg\max_k p(C_k)\mathcal{N}(x_j; \mu_{jk}, \sigma^2_{jk})$;
- Loss measure: Log likelihood;
- Optimization method: MLE.

In `MATLAB` we can train a naive Bayes classifier as follows:

```
nb_model = fitcnb(x,t);
nb_model.DistributionParameters
nb_model.Prior
```

where in `nb_model.DistributionParameters` we have the estimated values for μ_{jk} and σ_{jk} and in `nb_model.Prior` we have the prior for the classes C_k (which are equal to the uniform one in this case).

By considering the trained parameters we are able to generate new data:

```
param = nb_model.DistributionParameters;
prior = cumsum(nb_model.Prior);
n_dim = size(param,2);

n_gen = 1000;
gendata = zeros(n_gen,n_dim);
gentarget = zeros(n_gen,1);

for ii = 1:n_gen
    gentarget(ii) = find(prior > rand(),1);
    for jj = 1:n_dim
        mu = param{gentarget(ii),jj}(1);
        sigma = param{gentarget(ii),jj}(2);
        gendata(ii,jj) = normrnd(mu,sigma);
    end
end
```

where for each data we randomly pick a class, according to the class prior, and then sample from the corresponding Gaussian distribution for each dimension of the data `n_dim`.

4.1.4 K-nearest neighbour

At last we classify the iris dataset by considering a non-parametric method, for instance a K-NN classifier:

- A distance metric: Euclidean;
- How many neighbours: 3;
- A weight function: no weights;
- How to fit with local points: Majority voting (break ties with lowest index class).

In MATLAB we may use the function:

```
knn_model = fitcknn(x, t, 'NumNeighbors', 3);
```

which does not require any learning phase (being K-NN classifier a non parametric method). If we change the K we can notice how the decision boundaries becomes smoother or rougher as we increase or decrease the value of K, respectively.

4.2 Exercises

Exercise 4.1

Implement the learning algorithm for the logistic regression on the Iris dataset by relying on the batch gradient descend optimization.

In which situation the use of this optimization algorithm is a good idea?

Exercise 4.2

The abalone dataset is composed by the following elements:

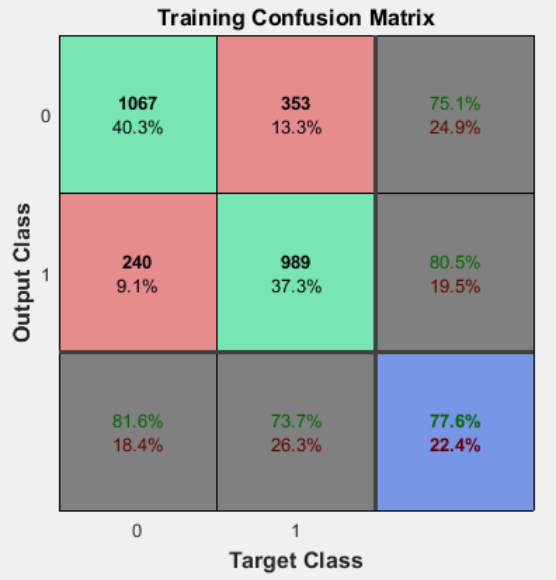
- an input matrix `abaloneInputs` (rows are features and columns are samples) containing as features Sex (M, F, and Infant), abalone Length, Diameter, Height, Whole weight, Shucked weight, Viscera weight, Shell weight
- a target vector `abaloneTargets` (a single feature where columns are samples)

Check if all the datapoints provided make sense.

Train a perceptron which uses the Length and Height of the shell to predict if an Abalone is Male or Female (remove the infants from the dataset analysed).

Exercise 4.3

Assume that training a perceptron on the Abalone setting provides you the following confusion matrix:



Compute the Accuracy and the F-score for this specific case.

Do you think that the perceptron converged at the end of the training procedure?

Exercise 4.4

Train a logistic regression which uses the Length and Height of the shell to predict if an Abalone is Male or Female (remove the infants from the dataset analysed).

Plot the separating hyperplane obtained by the logistic regression.

Exercise 4.5

Implement the K-NN classifier and apply it to the problem of imported car classification (`imports-85.mat` data). The task consists in understanding if a car is powered by gas (value 20) or diesel (11), stored in the 18-th column of the data matrix `X` basing on the curb-weight (column 7) and city-mpg (column 14).

How can we introduce some form of regularization in a K-NN classifier?

What are meaningful values for the parameter k of the K-NN classifier?

4.2.1 Questions

Exercise 4.6

Which of the following is an example of *qualitative variable*?

1. Height
2. Age
3. Speed
4. Colour

Provide a method to convert the qualitative ones into quantitative one, without introducing further structure over the data.

Exercise 4.7

Suppose we collect data for a group of workers with variables hours spent working x_1, number of completed projects x_2 and receive a bonus t. We fit a logistic regression and produce estimated coefficients: $w_0 = -6$, $w_1 = 0.05$ and $w_2 = 1$.

Estimate the probability that a worker who worked for 40h and completed 3.5 projects gets an bonus.

How many hours would that worker need to spend working to have a 50% chance of getting an bonus?

Do you think that values of z in $\sigma(z)$ lower than -6 make sense in this problem? Why?

* Exercise 4.8

Derive for logistic regression, the gradient descent update for a batch of K samples.

Do we have assurance of converge to the optimum?

Exercise 4.9

Tell if the following statement about the perceptron algorithm for classification are true or false.

1. Shuffling the initial data is fundamental for the perceptron optimization procedure;
2. We are guaranteed that the overall loss of the method is decreasing over time;
3. There exists a unique solution to the minimization of the perceptron loss;
4. The choice of a proper learning rate α might speed up the learning process.

Motivate your answer.

Exercise 4.10

You are working on a spam classification system using logistic regression. "Spam" is a positive class ($y = 1$) and "not spam" is the negative class ($y = 0$). You have trained your classifier and there are $N = 1000$ samples. The confusion matrix is:

	Actual Class: 1	Actual Class: 0
Predicted Class: 1	85	890
Predicted Class: 0	15	10

What is the classifier recall? What about the $F1$ score? What would you try to improve in such a system? Should we aim at solving a specific issue?

Exercise 4.11

Suppose you have trained a logistic regression classifier which is output is y_n. Currently, you predict 1 if $y_n > \tau$, and predict 0 if $y_n < \tau$, where currently the threshold is $\tau = 0.5$.

Suppose you decrease the threshold to $\tau = 0.3$. Which of the following are true? Check all that apply and provide a motivation.

1. The classifier is likely to now have higher precision.
2. The classifier is likely to have unchanged precision and recall, but higher accuracy.
3. The classifier is likely to now have higher recall.
4. The classifier is likely to have unchanged precision and recall, but lower accuracy.

Exercise 4.12

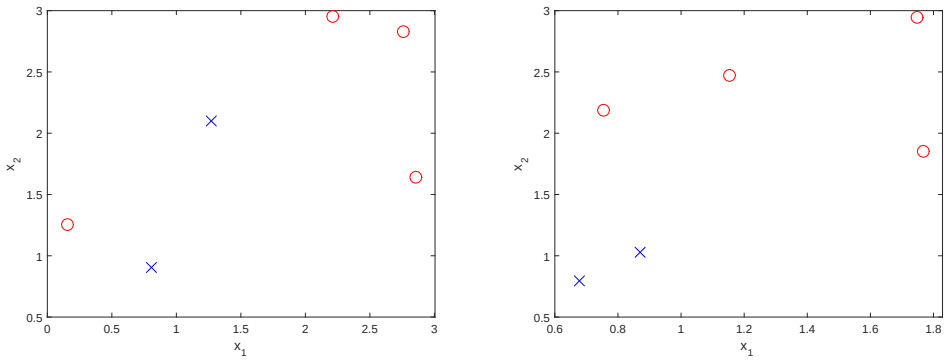
Which of the following is NOT a linear function in x:

1. $f(x) = a + b^2x$
2. $\delta_k(x) = \frac{x\mu}{\sigma^2} - \frac{\mu^2}{2\sigma^2} + \log(\pi)$
3. $logit(P(y = 1|x))$ where $P(y = 1|x)$ is a logistic regression
4. $P(y = 1|x)$ from logistic regression
5. $g(x) = \frac{x-1}{x+1}$

6. $h(x) = \frac{x^2-1}{x+1}$

Exercise 4.13

Consider the following datasets:



and consider the online stochastic gradient descend algorithm to train a perceptron. Does the learning procedure terminates? If so, how many steps we require to reach convergence? Provide motivations for your answers.

Exercise 4.14

Starting from the formula of the softmax classifier:

$$y_k(x) = \frac{\exp(w_k^T x)}{\sum_j \exp(w_j^T x)}$$

derive the formula for the sigmoid logistic regression for the two classes problem.

Exercise 4.15

Consider separately the following characteristics for an ML problem:

1. Large dataset (big data scenario);
2. Embedded system;
3. Prior information on data distribution;
4. Learning in a Real-time scenario.

Provide motivations for the use of either a parametric or non-parametric method.

Exercise 4.16

Consider a classification problem having more than two classes. Proposed a method to deal with multiple classes in each of the following methods:

1. Naive Bayes;
2. Perceptron;
3. Logistic regression;
4. K-NN.

Motivate your answers.