# Machine Learning: Lab assessment

February 2, 2017

## 1 Loading the data into your workspace

Each student has been assigned a different data file which can be downloaded from the following URL (do not forget to replace your student number when copying the address in the browser):

```
http://www.tsc.uc3m.es/~jarenas/MMC2017/YourNIA_data.mat
```

The necessary variables for the session

```
xtr_reg, str_reg, xval_reg, sval_reg, xtr_rl, xval_rl, ytr_rl, yval_rl
```

can now be loaded in your workspace using the following commands:

```
Matlab:
>>> load YourNIA_data

Python:
>>> import scipy.io
>>> data = scipy.io.loadmat('YourNIA_data.mat')
>>> xtr_gp = data['xtr_gp']
>>> str_gp = data['str_gp']
>>> ...
```

## 2 Regression question (50%)

After executing the previous command, you will find in your workspace a training set composed of 500 input-output data pairs, $\mathcal{D} = \{\mathbf{x}_i, s_i\}_{i=1}^{500}$.

The 500 input vectors have been arranged as the rows of variable `xtr_reg`, while their corresponding targets are available in vector `str_reg`. Furthermore, you can find 100 validation patterns as the rows of matrix `xval_reg` with corresponding labels stored in `sval_reg`. For this exercise, you should use all provided variables without any normalization process.

(a) Obtain the estimator $\hat{s} = s_0$, with $s_0$ a constant, that minimizes the average square error over the training data. Calculate the average square error of such estimator over the validation set. Save your results in variables `s0` and `E2val`.

(b) Calculate the validation error of a $k$-nn regression model, exploring the value of $k$ in the interval $k \in [1, 10]$. Save the validation average square errors in a vector `E2val_knn`, and the best value of $k$ (according to your validation set) as `bestK`.

(c) Let us assume next that the data have been generated according to the following model

$$s^{(i)} = [1\ ,\ x_1^{(i)}, \cdots, x_N^{(i)}\ , \exp(x_1^{(i)}), \cdots, \exp(x_N^{(i)})\ ]\mathbf{w} + \varepsilon^{(i)}, \quad 1 < i < 500$$

where $N$ is the dimension of the input data. Note that this model extends input vector $\mathbf{x}_i$ with the exponential value of each variable in the input pattern.

Assuming *a priori* distributions $\mathbf{w} \sim \mathcal{N}(\mathbf{0}, 2\mathbf{I})$ and $\varepsilon_i \sim \mathcal{N}(0, 0.5)$, and assuming also that noise samples $\{\varepsilon_i\}_{i=1}^{500}$ are independent of each other and of $\mathbf{w}$, obtain the *a posteriori* mean and covariance matrix of $\mathbf{w}$. Save your results in variables named `w_mean` and `w_cov`.

(d) Calculate the log-likelihood of the previous model using the training data only, and save the result as variable `logM`.

Do not forget to save variables

```
s0, E2val, E2val_knn, bestK, w_mean, w_cov, logM
```

# 3 Classification question (50%)

Each of the data matrices `xtr_rl` and `xval_rl` contains 500 10-dimensional data vectors. For this exercise, use directly the output of the `generatedata.p` function, without any normalization process.

It is known that the binary labels `ytr_rl` and `yval_rl` (with values in $\{0, 1\}$) have been generated according to a logistic model, so that

$$p\{y = 1|\mathbf{w}_e, \mathbf{x}\} = \frac{1}{1 + \exp(-\mathbf{w}_e^T \mathbf{x}_e)}$$

with $\mathbf{x}_e = (1, x_1, \ldots, x_{10})^T$ and $\mathbf{w}_e = (w_0, w_1, \ldots, w_{10})$. It is also suspected that variables $x_i$ for large $i$ are less relevant for the classification task, so we would like to fit a model with just the first $m$ variables $x_i$, (with $i \leq m$), for some $m$. In this exercise we will explore two procedures to do that.

(a) Normalize the training and validation data in such a way that all variables from the training data matrix have zero mean and unit variance. Save the result in variables `xtr_n` and `xval_n`. Use the normalized variables for the rest of the exam.

(b) As a preliminary exercise, adjust a logistic regression model using the available training data (`xtr_n` and `ytr_rl`). Use all variables, and determine the classification error rate measured over the validation data (`xval_n` and `yval_rl`).

To fit this model (and all models in the rest of the exercise), apply 500 iterations of a gradient descent algorithm to get the MAP estimates of weights, assuming a gaussian prior. Take a value of $C = 100$ and a learning step $\rho = 0.001$, initializing all weights to 0.

Save the resulting weight-vector in variable `w10` and the error rate in variable `e10`.

(c) Procedure 1: Obtain a new weight vector equal to `w10`, but setting to zero the all coefficients $w_n$, for $1 \leq n \leq 10$ such that

$$\sum_{i=n}^{10} w_i < 0.1 \cdot \sum_{i=i}^{10} w_i.$$

Find the new classification error rate when using this weight-vector, measured over the validation data. Save your result in variable `en`.

(d) Procedure 2: Train 10 different logistic regression models. Model $m$ should use only variables $x_1, x_2, \ldots, x_m$. Calculate the classification error rate $e_m$ (measured over the validation data) for each of the 10 cases. Then, select the model minimizing

$$E(m) = e_m + 0.005 \cdot m^2$$

and keep the best result. Save the minimum validation error in variable `eopt`, the optimal value of $m$ in `mopt`, and the corresponding weight-vector (only for this best case) in variable `wopt`.

Please, do not forget to save variables:

```
xtr_n, xval_n, w10, e10, en, eopt, mopt, wopt
```

# 4 Handing in your results

Save (at least) the variables mentioned in the exercises in a file called `results.mat`. The following matlab command performs this task for you:

```
save('results.mat', 'variable1', 'variable2', ...)
```

If you are using python, use instead:

```
>>> scipy.io.savemat('results.mat', {'variable1': variable1, 'variable2': ...
    variable2, ... })
```

Zip file `results.mat` together with your code in a file called `LabMMC.zip`, and upload the .zip file to Aula Global before the deadline expires.