UiT
THE ARCTIC UNIVERSITY OF NORWAY

2. Logistic Regression and Evaluation Metrics

FYS-2021 Exercises

Department of Physics and Technology
Faculty of Science and Technology

Logistic regression

Problem 1

Despite the term 'regression' being part of its name, logistic regression is considered as a classification algorithm. This naming can often lead to confusion, particularly when contrasted with linear regression. In Chapter 10.7 of the textbook (Alpaydin, 3rd ed.), the mathematical formulation for logistic regression is presented as:

$$y = \frac{1}{1 + \exp\left[-\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right)\right]}.$$

We can rewrite this expression using the sigmoid function, denoted by $\sigma(\cdot)$, as follows:

$$y = \sigma\left(\boldsymbol{w}^T\boldsymbol{x} + w_0\right).$$

In the formula, the term $\boldsymbol{w}^T\boldsymbol{x} + w_0$ can be viewed as a linear decision boundary hyperplane that splits the input space into two regions. The sigmoid function then maps each region into a label space ranging from $(0, 1)$, with the boundary located at the midpoint of 0.5.

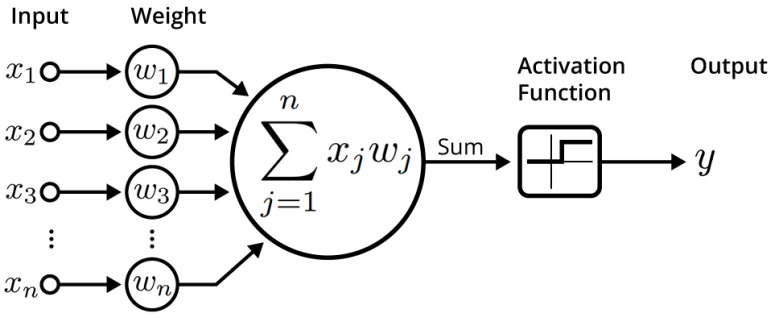
(1a) Generate five bivariate instances for each of two normal distributions, with each distribution representing a distinct class: $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$. Let $\boldsymbol{\mu}_1 = [0, 0]$, $\boldsymbol{\mu}_2 = [1, 1]$, and both $\boldsymbol{\Sigma}_1$ and $\boldsymbol{\Sigma}_2$ be identity matrices. Plot these instances on a Cartesian plane. Then, introduce an arbitrary linear decision boundary, represented by $\boldsymbol{w}^T\boldsymbol{x} + w_0$. The boundary should efficiently differentiate between the two classes.

(1b) Transform the Cartesian plane into a three-dimensional space by adding a z-axis, which represents the output of the sigmoid function, ranging from $(0, 1)$. Note that the linear decision boundary in the Cartesian plane corresponds to the midpoint of 0.5 on the z-axis. The instances sampled from the $N(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ distribution, which represents class 1, should yield a sigmoid output close to 0. Conversely, the instances from the $N(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ distribution, representing class 2, should have a sigmoid output close to 1.

(1c) Assume that the linear decision boundary in (1a) is ineffective, leading to an estimated accuracy of around 50%. Replicate the process in (1b) using this ineffective decision boundary.

(1d) Elaborate on the appropriate choice of the loss function for this binary classification task by discussing the differences and implications seen in the visualizations from (1b) and (1c).

(1e) The figure below depicts the schematic mechanism of a single neuron. Elaborate on the relevance between this neuron and logistic regression.



Problem 2

Note that the algorithms given in the book are very slow and inefficient. It is usually expected that the code is "vectorized". E.g. the nested loops:

```
for t = 0,...,N
  for j = 0,...,d
    o^t = o^t + w_j x_j^t
```

Should be written as a matrix multiplication:

Assume you have a weight vector $\boldsymbol{w} = [w_1, \ldots, w_d]^{\mathrm{T}}$ and a matrix $\mathbf{X}$ with the number of samples along the first (vertical) axis and the feature dimensions along the second (horisontal) axis:

$$\mathbf{X} = \begin{bmatrix} x_1^1 & \ldots & x_d^1 \\ & \vdots & \\ x_1^N & \ldots & x_d^N \end{bmatrix} = \begin{bmatrix} \boldsymbol{x}^1 \rightarrow \\ \vdots \\ \boldsymbol{x}^N \rightarrow \end{bmatrix}$$

then we can write the above for loop as a matrix multiplication like so:

```
o = np.dot(X,w)
```

The above operation will return the same output as the first pseudocode, but is ***much*** more efficient.

(2a) Implement your own, vectorised, version of the logistic discrimination algorithm. It might help if you first write it as pseudocode before you do the actual implementation!

It is possible to write the entire algorithm given in the books figure 10.6 as one single `for` loop!

Note that this is challenging! If you find this too challenging, you *may* iterate over the samples. However, do not iterate over the dimensions of the features!

In the file `tictac_end.csv`, the end configurations 958 different tic-tac-toe games are provided. Each row represents a different outcome, with the first element being 1 if player × won, and 0 otherwise. The rest of the row represents the board, such that, when reshaped to a 3×3 array, the original board is recovered.

(2b) Test your classifier using the tic-tac-toe end-game dataset. Report the confusion matrix and the classification accuracy.

(2c) OPTIONAL: From Alpaydin Chapter 10.8 (not in the syllabus). Implement your own logistic discrimination-by-regression algorithm, and test this in the same manner as above. Are there any notable differences between the two? Why/why not?

Problem 3

(3a) Show that for 2 vectors x and a in $\mathbb{R}^N$ we have the following relationship with the gradient of the scalar product:

$$\frac{\partial}{\partial a} x^T a = x = \frac{\partial}{\partial a} x^T a. \tag{1}$$

Evaluation metrics

Problem 4

Suppose you have a classifier that classifies to the positive class for

$$\hat{\mathrm{P}}(C_1|x) > \theta, \quad 0 < \theta < 1,$$

and that the number of false positives, f_p, classified can be modelled as

$$f_p = N \cos\left(\frac{\pi\theta}{2}\right)$$

where N is the total of negative training samples. The true positives can be modelled as

$$t_p = T \cos\left(\frac{\pi\theta}{2}\right)\left(2 - \cos\left(\frac{\pi\theta}{2}\right)\right)$$

where T is the number of positive training samples.

(4a) Find the true positive and false positive rates.

(4b) Plot the receiver operating characteristics (ROC) curve.

(4c) Calculate the AUC score. Hint: it might be useful to obtain an expression for the true positive rate as a function of the false positive rate.

Problem 5

Different classifier algorithms has probability outputs as given in `soft-classifications-#.csv` (the `#` is a number).

All these files are structured identically: The first column is the true label of the classified data point and the second column is the classified probability of that data point belonging to the positive class. In the file header you can find more info of number of classes and datapoints.

(5a) Implement a script that takes a dataset and makes two plots: The first plot should contain the ROC curve and the second plot should visualize the accuracy, tp-rate and fp-rate as a function of threshold. The script should also calculate the AUC-score and estimate the optimal ROC cut-off point.

(5b) What can you say about the classifiers from the plots? Which ones are good and which ones are bad, and why?