CS577 - Machine Learning – Assignment 3

Alexandros Angelakis csdp1363, angelakis@csd.uoc.gr

Exercise 1 - Logistic Regression (Theoretical)

The logistic regression model predicts probabilities $P(y = 1|x)$ and $P(y = 0|x)$ for binary classification based on the logistic function:

$$P(y = 1|x) = \frac{1}{1 + e^{-w^T x}}$$

$$P(y = 0|x) = \frac{e^{-w^T x}}{1 + e^{-w^T x}}$$

Since the data are linearly separable and we've centered our data around 0 (without loss of generality), the hyperplane $x = 0$ perfectly separates the two classes, meaning:
- All points in class 1 lie on one side of the decision boundary $(x > 0)$ such that $w^T x > 0$
- All points in class 0 lie on the other side of the decision boundary $(x < 0)$ such that $w^T x < 0$

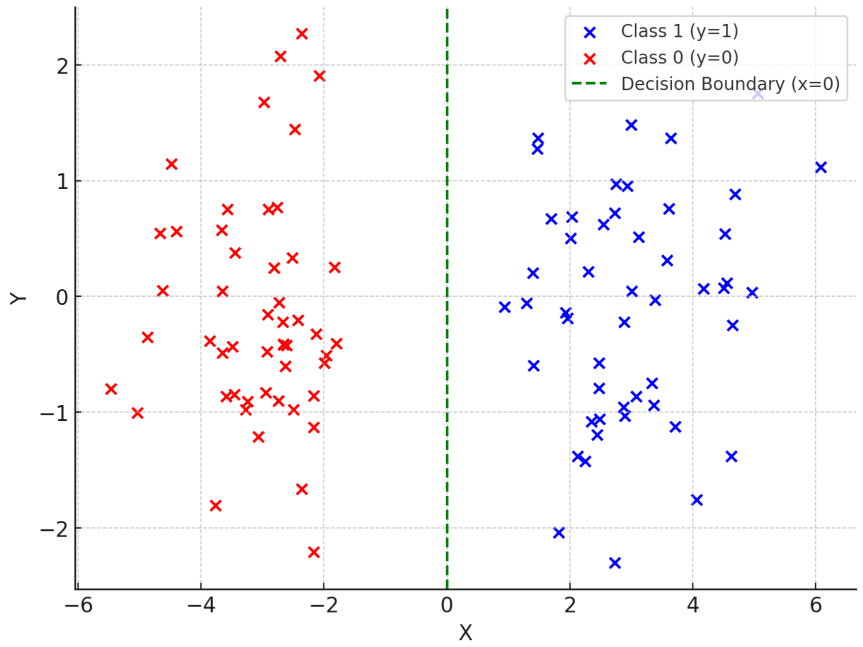

Figure 1: Plot that shows perfect separation between two classes at x = 0.

In order to train a logistic regression model, we need to use the maximum condition likelihood estimation (MCLE):

$$w^* = \arg\max_w \prod_{l=1}^{n} P(y^l|x^l, w)$$

which is equivalent to maximizing the log-likelihood:

$$\begin{aligned} w^* &= \arg\max_w \sum_{l=1}^n \ln P(y^l|x^l, w) \\ &= \arg\max_w \sum_{l=1}^n y^l \ln P(y^l = 1|x^l, w) + (1 - y^l) \ln P(y^l = 0|x^l, w) \end{aligned}$$

- For $y^l = 1$: since the data are linearly separable, we can find a w such that $w^T x^l > 0, \forall y^l = 1$. The probability $P(y^l = 1|x^l, w) = \frac{1}{1+e^{-w^T x}}$ approaches 1 as $w^T x^l \to \infty$ and the log-likelihood increases indefinitely: $\ln P(y^l = 1|x^l, w) \to \infty$. We can see this behaviour in Figure 2.

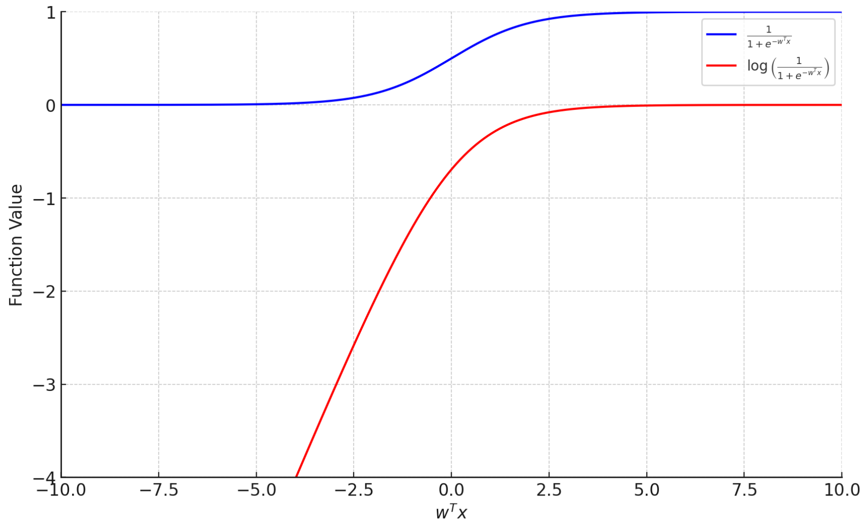

Figure 2: Plot with the probability $P(y^l = 1|x^l, w)$ and the log-likelihood $\ln P(y^l = 1|x^l, w)$.

- For $y^l = 0$: similarly, for all points in class 0 we can find a w such that $w^T x^l < 0$. The probability $P(y^l = 0|x^l, w) = \frac{e^{-w^T x}}{1+e^{-w^T x}}$ approaches 1 as $w^T x^l \to -\infty$, and the log-likelihood $\ln P(y^l = 0|x^l, w) \to 0$. This is also visible in Figure 3.

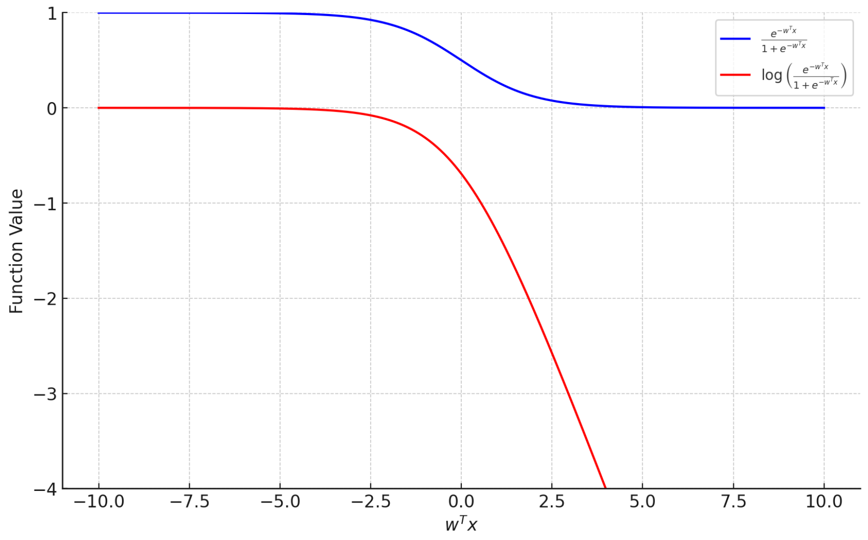

Figure 3: Plot with the probability $P(y^l = 0|x^l, w)$ and the log-likelihood $\ln P(y^l = 0|x^l, w)$.

Since the log-likelihood increases without bound as $|w| \to \infty$, the optimization algorithm will never converge, because it will never find a finite value for w. Instead, it will continue to increase the weights in an attempt to further separate the data, resulting in divergence to infinity. Essentially, the decision boundary becomes extremely sharp, converging to a step function.

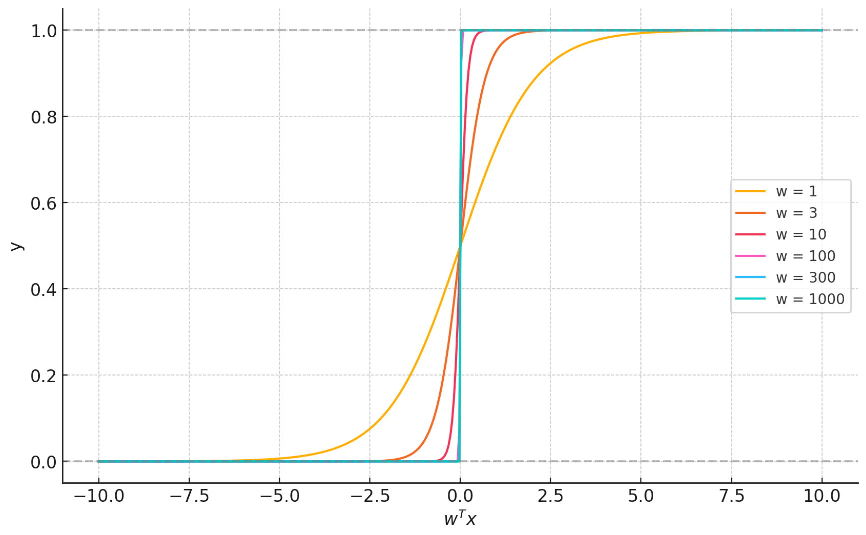

Figure 4: Decision Boundaries as w increases

Exercise 2 - Evaluation of Classifiers (Programming)

For this exercise, I applied all the classification algorithms—Trivial, NBC, and Logistic Regression—to all the given datasets (categorical, continuous, and mixed). For NBC, I used my own implementation, modifying it accordingly to process one-hot encoded data as well as mixed data. Running the "run_analysis" script will reproduce all results and plots. The plots will appear for each dataset, and they are also saved in the "figures/" folder.

In Figure 5 we see three plots showing the relationship between accuracy and sample size for each classifier across each dataset.

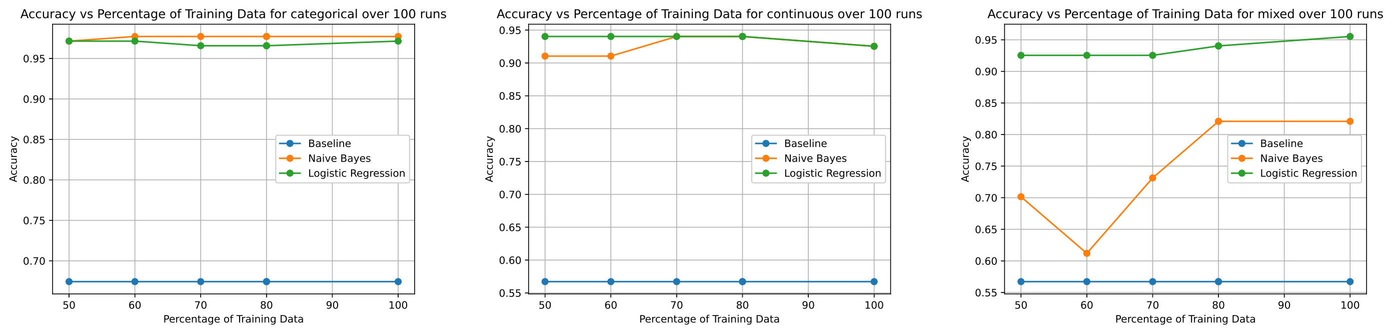

Figure 5: Mean Accuracy over 100 runs for each dataset

- For the categorical dataset, both Naïve Bayes and Logistic Regression perform very well, achieving an accuracy of approximately 96%, while the baseline accuracy is around 68%. There are minimal fluctuations across different percentages of training data.

- For the continuous dataset, Logistic Regression outperforms Naïve Bayes, achieving the highest accuracy across different training sizes, with a slight decrease in accuracy as the training percentage increases (from 94% to 92%). The baseline classifier remains flat and low, indicating poor performance relative to both NBC and LR.

- For the mixed dataset, Logistic Regression outperforms NBC across all training percentages, achieving consistently high accuracy that increases slightly as the training data percentage increases. NBC, on the other hand, shows more variability in its performance, particularly at lower training sizes, where its accuracy dips significantly (62%). However, its accuracy improves as the training percentage increases. The baseline classifier again performs poorly, showing no improvement with increased training size.

How does NBC compare to the trivial and LR classifiers?

NBC consistently outperforms the trivial classifier across all datasets, demonstrating its effectiveness in identifying non-trivial patterns within the data. In theory, the linear classifiers produced by Logistic Regression and Gaussian Naive Bayes converge to identical performance in the limit as the number of training examples approaches infinity, provided the Naive Bayes assumptions hold*. This behavior is observed in the left and middle figures, where both classifiers achieve nearly the same performance as the amount of training data increases. However, for the mixed data, where the structure is more complex and feature dependencies may play a role, the Naive Bayes bias causes it to perform less accurately than Logistic Regression. This is evident in the right figure, where the difference in accuracies between these two classifiers is clearly noticeable.

How does different samples size affect the results?

The performance of both NBC and LR generally improves or stabilizes as the percentage of training data increases. This is expected, as larger datasets typically provide more information, enabling the classifiers to generalize better. However, for the mixed dataset, NBC shows a more noticeable improvement with increased training data, indicating that it might struggle with smaller sizes on mixed data types, potentially due to increased data complexity. The baseline classifier remains unaffected by the percentage of training data, as its accuracy is determined by class distribution rather than learned patterns.

*GENERATIVE AND DISCRIMINATIVE CLASSIFIERS - Machine Learning - Tom M. Mitchell - Copyright ©2015.

Were the results expected? Why?

The results from these three plots are as expected. Logistic Regression is robust in handling both categorical and continuous data, which explains its strong performance across all datasets. Naïve Bayes, while effective, assumes feature independence, which can limit its performance on more complex datasets like the mixed dataset. Its lower accuracy on mixed data reflects this limitation. The baseline classifier's consistently low performance is also expected, as it does not learn from the data and simply predicts the majority class.

Exercise 3 - Regularization of Logistic Regression (Programming)

Using an existing implementation of the Logistic Regression, I trained four different classifiers, one with no penalty and three with Lasso penalty (with different λ values: 0.5, 10 and 100). In Figure 6 we can see the values of the weights of all the classifiers. The bigger the values of λ, the smaller the $C = \frac{1}{\lambda}$, corresponding to stronger regularization.

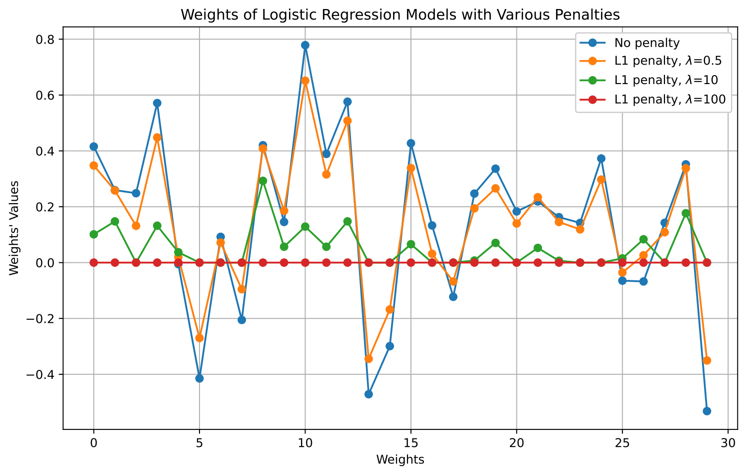

Figure 6: Weights and their values for each method

- Without any regularization (blue line), the weights vary widely with positive and negative values, reflecting each feature's individual contribution based on the data alone. This can result in large weight magnitudes if the model relies heavily on specific features, potentially causing overfitting.

- L1 Penalty with λ = 0.5 (orange line): With a small L1 penalty, the weights are reduced but still show some variation. This regularization lowers weight values slightly but allows the model to keep non-zero values for many features, helping to prevent overfitting while maintaining model expressiveness.

- L1 Penalty with λ = 10 (green line): With a larger penalty, more weights are pushed closer to zero. The stronger regularization makes the model more selective, keeping only the most important features while discarding weaker ones by setting their weights closer to zero, resulting in a sparser model.

- L1 Penalty with λ = 100 (red line): With a high penalty, all weights are driven to zero, as the strong regularization overwhelms the model's ability to assign non-zero values. This extreme level of regularization almost eliminates the influence of all features, resulting in a model that acts similarly to a baseline model with no features. This can lead to underfitting, as the model loses valuable information from the features.