**A2:** Classification with SVM, BP and MLR

Professors: SERGIO GÓMEZ JIMÉNEZ, JORDI DUCH GAVALDÀ

Students: SEBASTIAN BUZDUGAN, ERIKA LOZANDA MARTINEZ

**Main Concepts:**  
The task involves data classification using Support Vector Machines (SVM), Back-Propagation (BP), and Multiple Linear Regression (MLR) on different datasets. Key aspects include parameter selection via cross-validation, data preprocessing, normalization, and evaluation through error rates, confusion matrices, ROC curves, and AUC. The assignment emphasizes finding optimal parameters for each algorithm, handling both binary and multifeatured datasets, and includes comprehensive documentation of methods and analyses.

**3RD DATABASE: BANKNOTE AUTHENTICATION:**

<https://archive.ics.uci.edu/dataset/267/banknote+authentication>

Images were captured from authentic and counterfeit banknote-like samples. These images were digitized using an industrial camera typically employed in print inspection. The resulting images are 400x400 pixels in size. The combination of the objective lens and the distance to the subject yielded gray-scale images with a resolution of approximately 660 dpi. Features were extracted from these images using the Wavelet Transform tool.

**ATTRIBUTES:**

* variance of Wavelet Transformed image (continuous)
* skewness of Wavelet Transformed image (continuous)
* curtosis of Wavelet Transformed image (continuous)
* entropy of image (continuous)
* class (integer)

1. **Support Vector Machines (SVM)**

Support Vector Machines (SVM) are a supervised learning algorithm primarily used for classification tasks. They work by finding a hyperplane in a multi-dimensional space to distinctly classify data points. SVM is particularly effective due to its ability to use kernel functions, making it capable of handling both linear and non-linear data.

In my application, I utilized **Weka**, a popular data mining software, and the **LibSVM** package for SVM implementation. LibSVM in Weka requires categorical class labels, so I modified my dataset's numeric labels to categorical ('A' and 'B'). After this adjustment, I conducted the classification task in Weka's "Classify" tab, starting with the training set and then incorporating a test set for a comprehensive analysis.

A screenshot of a computer

Description automatically generated

The first database used is “ring-merged-weka-letter” – and convert 0/1 into A/B because this package does not allow those values. With this database we use the training set LibSVM and get the following output:

A screenshot of a computer

Description automatically generated

Then we need to use the custom test set in order to get the confusion Matrix. To compute the classification error using the formula :

A math equations with numbers

Description automatically generatedA math equations with numbers

Description automatically generated

A screenshot of a computer

Description automatically generated

Below we can see the plot result generated by Weka with this dataset in comparison to the same plot generated but using the “ring-merged-weka-letters”

A screenshot of a computer screen

Description automatically generatedA screen shot of a computer

Description automatically generated

1. **Multiple Linear Regression (MLR)**

For the MLR case, we are also using the Weka software but other libraries this time (LinearRegression). However in this case we can use the original CSV file.

Then we train the data using the default method:

A screenshot of a computer

Description automatically generated

Then we compare the results with the one for the cross-validation with 10 folds:

A screenshot of a computer

Description automatically generated

By creating a table to compare the values if they are bigger or smaller then 0.5 (threshold value) we can detect the approximation of the function. For 0.5 => 4667 (sum of errors) – which is very fast.

A screenshot of a computer

Description automatically generated

1. **Back-Propagation (BP)**

We have used TensorFlow, one of the most populat when it comes to BP algorithm and and Keras which is a library that provides the interface for artificial neural networks.

The classification error is a critical metric in evaluating the performance of our model. It measures the proportion of incorrect predictions out of the total number of instances. In a binary classification task, an error occurs when the predicted class does not match the actual class. For example, predicting that a client will subscribe ('yes') when they actually will not ('no'), or vice versa.

A screenshot of a computer

Description automatically generated

Figure 1 - Bank dataset

This matrix reveals that:

* 698 instances were correctly predicted as 'no' (True Negatives).
* 43 instances were incorrectly predicted as 'yes' (False Positives).
* 43 instances were incorrectly predicted as 'no' (False Negatives).
* 40 instances were correctly predicted as 'yes' (True Positives).

The classification error from the same fold is calculated to be approximately 10.44%. This error rate is a significant metric, considering the imbalance in the dataset; it reflects the proportion of misclassifications.

For the **banknote\_authentication** dataset, we've designed a neural network with the following specifications:

* **Learning Rate:** 0.05, which governs the speed at which the model learns. A moderate rate helps in converging to a global minimum without overshooting.
* **Momentum:** 0.1, to help accelerate the optimizer in the correct direction and prevent oscillations.
* **Neural Network Architecture:** The network consists of an input layer with 4 neurons (one for each feature), two hidden layers with 20 and 5 neurons respectively, and an output layer with a single neuron.

A screenshot of a computer

Description automatically generated

Figure 2 - Banknote dataset

This confusion matrix tells us that:

* 156 instances of genuine banknotes were correctly identified (True Negatives).
* 5 instances of genuine banknotes were incorrectly classified as forgeries (False Positives).
* 0 instances of forged banknotes were incorrectly classified as genuine (False Negatives).
* 114 instances of forged banknotes were correctly identified (True Positives).

From the given fold, the classification error is remarkably low, at approximately 1.82%. This low error rate indicates a high level of accuracy in the model's ability to classify banknotes as genuine or counterfeit.

A 4-fold cross-validation technique is also used for the ring dataset to validate the model's performance and ensure its ability to generalize to new data. This technique involves partitioning the dataset into four subsets, training on three subsets, and validating on the fourth. This process is repeated four times, with each subset used once as the validation set.

A screenshot of a computer

Description automatically generated

Figure 3 - Ring dataset

Interpreting the confusion matrix:

* 1233 True Negatives (TN): Correctly predicted negative class instances.
* 82 False Positives (FP): Negative class instances incorrectly predicted as positive.
* 7 False Negatives (FN): Positive class instances incorrectly predicted as negative.
* 1178 True Positives (TP): Correctly predicted positive class instances.

The classification error, derived from the confusion matrix, is approximately 3.56%, indicating the model's high accuracy in classifying the instances. This low error rate is particularly noteworthy in a dataset with a complex, non-linear decision boundary like that of the ring dataset.

The custom BP algorithm has been updated from its original form in the first assignment (A1), leading to enhanced functionality and now works. Here are the key improvements:

* **Enhanced Cross-Validation Methodology:** The addition of the 'trainingFolds' parameter to the BP algorithm is a major change. The 'folds' refer to the different segments that this new feature creates from the data and epochs. Training and validation sets are rotated using each fold in a methodical manner, which results in a more complete and reliable training procedure.
* **Modification to the "CostFunction" Method:** A new parameter has been added to the algorithm's "costFunction" method. This modification is essential for precisely computing the classification error, particularly in light of the 'forward' method's adoption of a threshold. The formula..... is used to calculate the error, where..... stands for various classification counts.
* **Normalization Procedures Are Removed:** The normalization and denormalization stages of data processing are eliminated in the current version of the BP algorithm. The'readDataAndNormalize' method has been modified due to the dataset being utilized already having a normalized format.
* **Threshold Application in the 'Forward' Method:** A notable change is the 'forward' method's addition of a new parameter. This update is necessary for testing since it applies a threshold to the final layer's output, turning it into binary form (0 or 1).

1. **Compute and compare**

Analyzing the data in the table below, it's evident that Multiple Linear Regression isn't the best fit for this scenario. Its predictions seem quite random, with errors nearing 50% in both cases. This is likely because linear regressions are designed to model linear relationships, and they falter when the relationships are nonlinear, leading to poor model performance.

|  |  |  |  |
| --- | --- | --- | --- |
|  | SVM | MLR | BP |
| ring-separable | 4.18% | 46.67% | 1.85% |
| ring-merged | 6.91% | 46.67% | 1.85% |

Neural networks typically excel when dealing with well-segregated and easily segmented data, which holds true in this case. Despite needing to modify our code and adjust the architecture and parameters, neural networks yielded quite impressive results. However, a notable downside is the longer execution time, probably due to the vast amount of data processed. This factor should be considered when opting for this method.

The most effective technique in this instance appears to be the Support Vector Machines (SVM). SVMs demonstrated lower classification error rates and overall better performance. This success can be attributed to the relative simplicity of creating hyperplanes in SVMs compared to the complexities involved in training a neural network. The nature of the data, which could be readily separated using lines in a plane, likely played a significant role in the effectiveness of SVMs in this context.

The BP algorithm demonstrates exceptional performance. This indicates the neural network's capability to capture complex, non-linear patterns in data. However, its performance on the bank dataset shows room for improvement.

|  |  |  |  |
| --- | --- | --- | --- |
| X | ring | bank | banknote |
| SVM | 4.18% | 9.22% | ? |
| MLR | 46.67% | 10.68% | 1.09% |
| BP | 1.85% | 11.04% | 0.36% |

**Performance Ratings**

* Green (Good): Error rates below 5% are considered excellent. BP's performance on the ring and banknote datasets falls into this category, showcasing high accuracy.
* Orange (Average): Error rates between 5% to 10% represent average performance. SVM's performance on the bank dataset and MLR on the banknote dataset are in this range, indicating a need for potential improvements or adjustments in model parameters.
* Red (Poor): Error rates above 10% are deemed below par. MLR's performance on the ring dataset is particularly concerning, indicating a poor fit of the model to the dataset's structure.

Based on these results, it seems that the BP algorithm performs better overall, especially for datasets with intricate boundaries or patterns. SVM remains a dependable technique across various datasets, whereas MLR's efficacy appears to be highly contingent on the dataset. The comparison will be completed and a complete picture of the capabilities of each method will be provided by the final error rate for SVM on the banknote authentication dataset, which is still to be determined (TBD).

GITHUB – REPOSITORY:

<https://github.com/sebastianbuzdugan/A2-NeuralNetworks>