Report

Homework 1

Goal: Provide solutions for two 10-class classification problems with medium and large input space

Andrea Massignan
1796802

Project Overview

The goal is to provide two different solution for a 10-class classification problem.

1 Dataset

The datasets used in this project consists in training sets and blind test sets. The training sets are composed by 50000 samples, each one with 100 features for Dataset 1 and 1000 features for Dataset 2. The blind test sets are composed by 10000 samples, each one with 100 features for Dataset 1 and 1000 features for Dataset 2. The labels are 10, one for each class.

- `X_train`: 50000 samples for each dataset.
- `n_features`: 100 for Dataset 1 and 1000 for Dataset 2.

2 Data Exploration

The datasets, that are in a csv file, the elements $V^{(i)}$ belonging to the X column are feature vectors that represent the input data.

The Y column contains the associated labels C_k(k=0,...9).

Thus, each line i in column X is a feature vector structured as:

$[V_1^{(i)}, V_2^{(i)}, ..., V_j^{(i)}, ... , V_d^{(i)}]$

Where d is the size of the feature vector with sizes described before.

And $i = 1, ..., n$, where n is the number of samples in the dataset as specified before.

These are loaded with an helper function called `load_data` that returns the training set in the X matrix and the labels in the Y vector.

Then the training set is split into `X_train` and `Y_train` with a test size of 0.2 and different random states.

Since the dataset is already vectorized, no further preprocessing is needed at the moment.

3 Metrics

The metrics used to evaluate the models are the accuracy and the confusion matrix.

3.1 Precision

Precision is a measure of the accuracy of the positive predictions made by a model. It is the ratio of true positive predictions to the total number of positive predictions made by the model (both true positives and false positives). Precision is calculated using the formula:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

Precision provides insight into the model's ability to avoid making false positive predictions. A high precision indicates that when the model predicts a positive class, it is likely to be correct.

3.2 Recall (Sensitivity or True Positive Rate)

Recall measures the ability of a model to correctly identify all relevant instances of the positive class. It is the ratio of true positive predictions to the total number of actual positive instances in the dataset (including both true positives and false negatives). Recall is calculated using the formula:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

Recall is especially important when the cost of false negatives (missing a positive instance) is high, as it focuses on minimizing false negatives.

3.3 F1-Score

The F1-score is the harmonic mean of precision and recall. It provides a balanced measure that considers both false positives and false negatives. The F1-score is calculated using the formula:

$$\text{F1-Score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

F1-score is useful when there is an uneven class distribution or when false positives and false negatives have different consequences.

3.4 Support

Support refers to the number of actual instances of each class in the dataset. It is the count of true positive and true negative instances for each class. Support is not directly involved in the calculation of precision, recall, or F1-score, but it provides context by showing how many instances belong to each class.

Classification

4 Data Loading and Splitting

4.1 Loading Dataset 1

The data loading process begins with dataset 1, where the feature matrix X and label vector Y are extracted using the `load_data` function. An informative message is printed to indicate the commencement of this operation. The resulting shape of matrix X is then presented, providing insights into the number of samples and features within dataset 1.

4.2 Loading Dataset 2

An analogous procedure is employed for dataset 2. The feature matrix $X2$ and label vector $Y2$ are obtained through the `load_data` function. A corresponding message is printed, and the shape of $X2$ is displayed to convey the dataset's sample and feature dimensions.

4.3 Splitting Data

The subsequent steps involve splitting the loaded data into training and testing sets:

```
X_train1, X_test1, Y_train1, Y_test1 = train_test_split(X, Y, ...)
  ...
X_train4, X_test4, Y_train4, Y_test4 = train_test_split(X2, Y2, ...)
```

Utilizing the `train_test_split` function, both dataset 1 and dataset 2 are partitioned into training and testing sets. Each dataset undergoes this process twice, resulting in four sets per dataset. The `test_size` parameter dictates the proportion allocated for testing (20%), while `random_state` ensures reproducibility.

The resulting variables (X_train1, X_test1, Y_train1, Y_test1, etc.) are now ready for subsequent machine learning tasks.

Additionally, the X_{train} is scaled with the `StandardScaler` function for dataset 1 and with the `MinMaxScaler` function for dataset 2. This is done to help the fitting process of the models.

First Evaluation with Binary Classification

After the data is loaded and splitted, the user is prompted to choose which of the dataset to use and then what type of model to use for the classification task. The user can choose between:

5 Bernoulli

BernoulliNB is a Naive Bayes classifier specifically tailored for multivariate Bernoulli models. Similar to MultinomialNB, this classifier is well-suited for handling discrete data.

The key distinction lies in their data types: MultinomialNB deals with occurrence counts, while BernoulliNB is optimized for binary or boolean features. BernoulliNB is preferred over other Naive Bayes classifiers when dealing with datasets where features are binary or boolean.

Additionally, BernoulliNB is a simpler model with fewer parameters, making it advantageous for scenarios with limited training data and leading to faster training and prediction times. We will use this model for the first evaluation and MultinomialNB for the multiclass evaluation.

5.1 Classification

Overall the model performs well, with an average accuracy of 90%.

Recall is 71%, meaning that the model captures a high percentage of actual positive instances.

Precision is 78%, meaning that the model has low false positive rate.

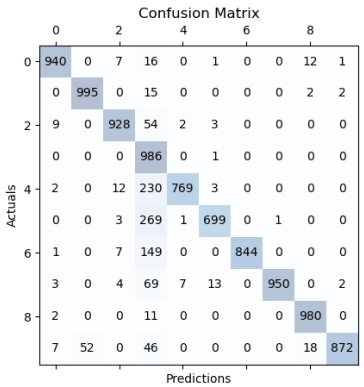


Figure 1: Confusion Matrix for BernoulliNB

6 SVM

Support Vector Machine is a supervised machine learning algorithm used for classification and regression tasks. It's particularly powerful in high-dimensional spaces and is effective in cases where the margin between different classes is clear.

The SVM model performs well in terms of precision and recall for most classes, but there are variations across different digits.
In this particular case we are using a linear kernel, which is a good choice for high-dimensional data, however, it may not be the best choice for this particular dataset.

- The accuracy is around 83% which is lower than the BernoulliNB model.
- Some classes, like class 5, have lower support (973), while others, like class 7, have higher support (1048).
- The F1-scores are generally high, however, class 2 stands out with a lower F1-score (0.22).
- Notably, class 5 has a recall of 1.00, on the other hand, classes 2 and 3 have lower recall values.
- For most classes, precision is relatively high, however, there are exceptions, such as class 5, where precision is notably lower (0.40).

The lower performance in certain classes, especially in terms of recall for classes 2 and 3, indicates areas where the model may benefit from further improvement.

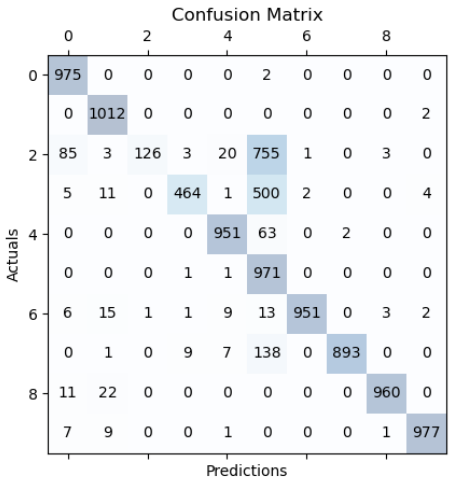


Figure 2: Confusion Matrix for SVM

7 Logistic Regression

Logistic regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable, although many more complex extensions exist. In regression analysis, logistic regression is estimating the parameters of a logistic model.

The model has a small problem of never converging and thus never stopping the training, however, it still performs well in terms of accuracy, precision and recall.

- Overall accuracy is 98%, indicating that the model correctly predicts the class labels for 98% of the instances in the dataset.
- High F1-scores (close to 1.0) indicate a good balance between precision and recall, this means that the model is robust enough.
- High recall values (close to 1.0) indicate that the model captures a high percentage of actual positive instances, which is fine.
- Precision also is close to 1.0, meaning that the model has low false positive rate.

The model performs very well across various metrics, indicating its effectiveness in classifying instances in the given dataset.

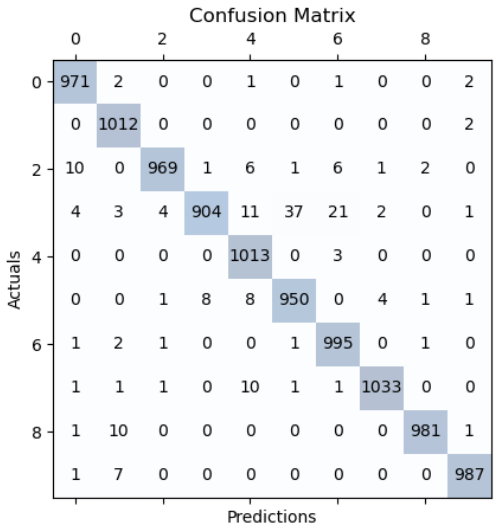


Figure 3: Confusion Matrix for Logistic Regression

Second Evaluation with Multiclass Classification

In this second batch we are using the same models as before, but with a different approach to the data.

8 Bernoulli to MultinomialNB

The Bernoulli model was almost fine, but still we want to improve the accuracy of the model, so we are going to use a MultinomialNB model with a multiclass approach.

- The accuracy is around 98% which is better than the BernoulliNB model.
- Even the F1-scores are generally high, with a minimum of 0.96, better than the BernoulliNB model (minimum of 0.71).
- Recall is also higher, with a minimum of 0.94, better than the BernoulliNB model (minimum of 0.72).
- Precision is higher too, with a minimum of 0.94, better than the BernoulliNB model (minimum of 0.56).

The MultinomialNB model appears to be better suited for the given dataset, providing higher accuracy and more balanced precision-recall trade-offs across all classes compared to the BernoulliNB model.

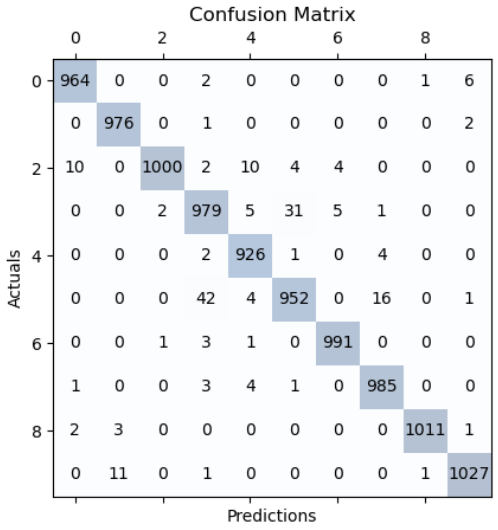


Figure 4: Confusion Matrix for MultinomialNB

9 SVM

The SVM was perfoming well, but I want to give a slight modification to the hyperparameters, so I'm going to use polynomial kernel instead of linear.

The model seems to performs better this way, in detail:

- The accuracy is around 99%, way better than the 83% of the linear kernel.
- The recall, precision and F1-scores are better too, with a minimum of 0.96, 0.97 and 0.97 respectively.
- Support values remain consistent, indicating that the number of instances for each class in the dataset is not significantly affected by the change in the kernel.

Overall the SVM with a poly kernel achieves a higher performance than the previous one (*linear*) and seems to be better suited for this type of dataset.

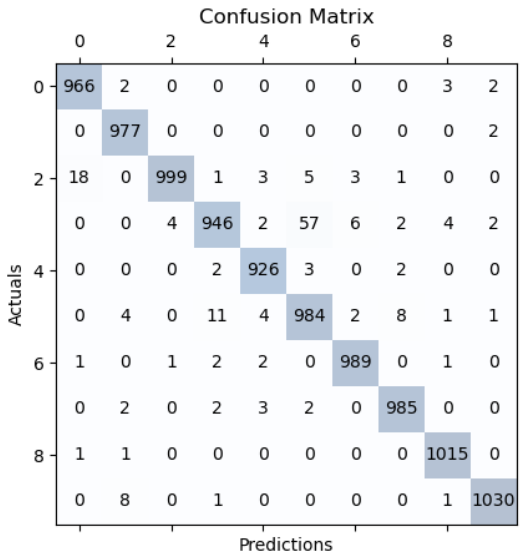


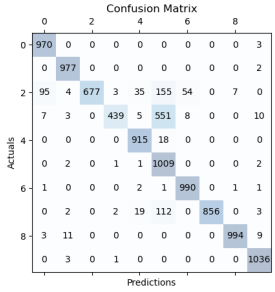
Figure 5: Confusion Matrix for SVM with poly kernel

10 Logistic Regression

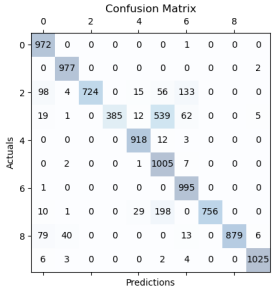
The Logistic Regression actually performed well even if it didn't converge, so I'm going to use the same model but with a different solver, to see if there could be any improvement.

- Tried the *newton-cg* solver, it converged but gave a lower performance: accuracy of 0.88, recall of 0.84, precision of 0.86 and f1-score of 0.86.
- Tried the *sag* solver, it gave a similar performance if slightly worse.
- Tried the *newton-cholesky* solver, it converged but the result was similar to the $newton-cg$ solver: accuracy of 0.86, recall of 0.83, precision of 0.94 and f1-score of 0.84.
- Tried the *liblinear* solver, it converged but the result was still worse than the *saga* solver: accuracy of 0.86, recall of 0.84, precision of 0.92 and f1-score of 0.85.
- Lastly the $lbfgs$ solver, it converged but the result was medium: accuracy of 0.89, recall of 0.90, precision of 0.95 and f1-score of 0.89.

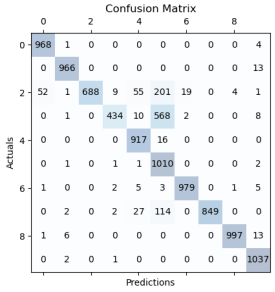
The Logistic Regression model with the *saga* solver still seems to be the best choice, by adding the *multiclass* option the results doesn't change much.



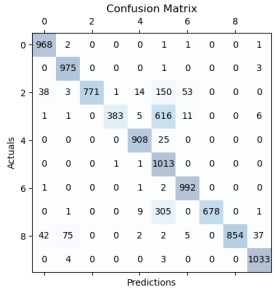
(a) $lbfgs$ solver



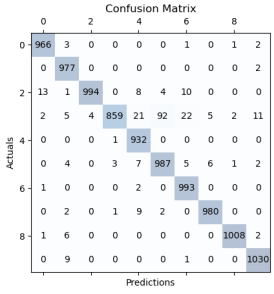
(b) *liblinear* solver



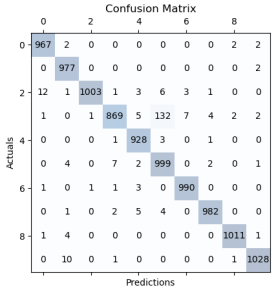
(c) *newton-cg* solver



(d) *newton-cholesky* solver



(e) *sag* solver



(f) *saga* solver

Blind Tests

We were provided with blind tests in the format of csv files that are structured as follows:

Index	X
0	$[V_1^{(0)}, V_2^{(0)}, ..., V_j^{(0)}, ... , V_d^{(0)}]$
...	...
i	$[V_1^{(i)}, V_2^{(i)}, ..., V_j^{(i)}, ... , V_d^{(i)}]$
...	...
N	$[V_1^{(N)}, V_2^{(N)}, ..., V_j^{(N)}, ... , V_d^{(N)}]$

The blind test structure consists of a two-column table with "Index" and "X." The "Index" column ranges from 0 to N, representing the index of each feature vector. The "X" column contains feature vectors denoted as arrays $[V_{ij}]$ with elements V_{ij}, where i is the index and j ranges from 1 to d, representing features. This format signifies N samples, each with a feature vector of length d.

11 Predictions

The predictions are made using the best model for each dataset, the results are the following:

- BernoulliNB: [3 8 8 ... 5 1 7]
- SVM with linear kernel: [5 8 8 ... 5 1 7]
- Logistic Regression: [3 8 8 ... 5 1 7]
- MultinomialNB: [3 8 8 ... 5 1 7]
- SVM with poly kernel: [3 8 8 ... 5 1 7]
- Logistic Regression with multiclass: [3 8 8 ... 5 1 7]

The predictions are saved in two csv files, one for each database.

Comparison with the second dataset

The second dataset is similar to the first one, but with many more labels, so we are going to use the same models as before to see how they perform.

- $BernoulliNB$: we have an improvement on accuracy to 0.96, the recall has a minimum of 0.88, the precision has a minimum of 0.85 and f1-score with a minimum of 0.91. So it performs better.
- SVM ($linear$): here the performance decreases, with an accuracy of 0.70, a recall with a minimum of 0.16, a precision with a minimum of 0.33 and an f1-score with a minimum of 0.27.
- *Logistic Regression* ($saga$): the performance is similar to the first dataset but slightly worse, with an accuracy of 0.94, a recall with a minimum of 0.67, a precision with a minimum of 0.83 and an f1-score with a minimum of 0.80.
- $MultinomialNB$: the performance is slightly worse here too, with an accuracy of 0.97, a recall with a minimum of 0.91, a precision with a minimum of 0.91 and an f1-score with a minimum of 0.91.
- SVM ($poly$): it's a bit worse, with an accuracy of 0.96, a recall with a minimum of 0.84, a precision with a minimum of 0.90 and an f1-score with a minimum of 0.91.
- *Logistic Regression* ($multiclass$, $saga$): the performance is similar to the first dataset, with an accuracy of 0.96, a recall with a minimum of 0.81, a precision with a minimum of 0.83 and an f1-score with a minimum of 0.88.

12 Different Loading Times

I have noticed that when working with the second dataset the loading times are a bit longer depending on the chosen model. When using BernoulliNB and MultinomialNB the loading is almost the same. SVM has a slight degradation on loading but we keep around 30-40 seconds of loading time. The Logistic Regression model has a huge degradation on loading, with a loading time of around 2-3 minutes.

13 Final Predictions

The predictions calculated with the second dataset are the following:

- BernoulliNB: [3 8 8 ... 5 1 7]
- SVM with linear kernel: [3 8 8 ... 5 1 7]
- Logistic Regression: [3 8 8 ... 5 1 7]
- MultinomialNB: [3 8 8 ... 5 1 7]
- SVM with poly kernel: [3 8 8 ... 5 1 7]
- Logistic Regression with multiclass: [3 8 8 ... 5 1 7]