Report TP1 Apprentissage Statistique Appliqué

AFANGNIBO Kokou Jérôme

HEDFI Houeida

KONDO Godson Leopold Junior

November 9, 2020

PART 1 : MNIST

1. Explanation of what happens when running clf.fit(X_train, y_train)

When we run clf.fit(X_train, y_train), the following three steps are made for each hyperparameter $k \in \{1, 2, 3, 4, 5\}$:

- The training sample, D, is partitioned into three sets, D^1, D^2 and D^3 in order to achieve a cross validation.

- The kNN algorithm is computed on D^{-i}, $i \in \{1, 2, 3\}$, and tested on D^i and an empirical risk, denoted by R_k^i, is calculated.

- The average risk, denoted $\bar{R}_k$, is calculated : $\bar{R}_k = \frac{1}{3} \sum_{i=1}^{3} R_k^i$.

The five average risks ($\bar{R}_k$, $k \in \{1, 2, 3, 4, 5\}$) are compared and the lowest average risk is returned with the value of k related. In our case, $\bar{R}_1 = 0.89149794$ and the best hyperparameter is $k = 1$.

2. Complexity for each of the three classifiers computed

If we note n the size of the training set and d the dimension space for those samples, we can say the following:

Training time complexity of the kNN algorithm is $\mathcal{O}(knd)$ while it is $\mathcal{O}(n^2)$ for the Support Vector Machines (LinearSVC()). Finally, the training time complexity for Logistic regression is $\mathcal{O}(nd)$.

3. Test accuracy and accuracy of a random guess

- Test accuracy is 0.875. This value is obtained on the test set by performing an 1-NN algorithm that is selected as the best one regarding to cross-validation accuracy on the training set.

- The accuracy of random guess = 0.1

4. LinearSVC() classifier, kernel and C

LinearSVC() classifier is a Support Vector Machines algorithm applied to classification problems. It uses a hinge or squared hinge loss functions and the kernel which only it supported is the linear one.

C is a strictly positive parameter; it is a regularization parameter that controls the strength of the penalty due to misclassification.

5. Outcome of np.logspace()

The command *np.logspace(-8, 8, 17, base=2)* divides the interval $[-8, 8]$ into 17 equidistant values. Each x value is modified according to the following formula: 2^x.
More generally, the command *np.logspace(-a, b, k, base=m)* divides the interval $[-a, b]$ into k equidistant values. Then, each x value is modified according to the following formula: m^x.

6. The warnings in SVC computation

- The warnings are convergence warnings. The SVC algorithm does not converge (with a maximum number of iterations of 5000) and the number of iterations must be increased.
- The parameter responsible for its appearence: ***max_iter***.

7. Changes with respect to the previous run of LinearSVC()

The change is that we have added a pipeline containing a scaler: ***MaxAbsScaler()***.
Machine learning (ML) pipelines consist of several steps to train a model. ML pipelines are used to help automate machine learning workflows. They operate by enabling a sequence of data to be transformed.

8. Explanation of what happens if we execute: pipe.fit(X_train, y_train) pipe.predict(X_test)

This command: *pipe.fit(X_train, y_train)* is used to train the model and to estimate parameters.
The second command *pipe.predict(X_test, y_test)* does not work because it contains an error. This command only takes as argument the covariates of the test sample.
The correct command would be : *pipe.predict(X_test)*. Executing this command allows us to output predictions for the target variable on the test sample.

9. Difference between StandardScaler() and MaxAbsScaler() and other scaling options available in sklearn

StandardScaler(): standardize features by removing the mean and scaling to unit variance. Centering and scaling happen independently on each feature by computing the relevant statistics on the samples in the training set. Whereas *MaxAbsScaler()* scales each feature by divided it by its maximum absolute value. It scales and translates each feature individually such that the maximal

absolute value of each feature in the training set will be 1.0. It does not shift/center the data, and thus does not destroy any sparsity.

Other scaling options available in sklearn are: ***MinMaxScaler, MaxAbsScaler, RobustScaler, PowerTransformer, QuantileTransformer, QuantileTransformer, Normalizer.***

10. Method to achieve test accuracy ≥ 0.9

We have used ***Ramdom Forest with a 3-fold cross validation*** algorithm and we have achieved a test accuracy equal to 0.935 (See Table **1**).
The random forest consists of n decision trees. Decision trees are built on bootstrapped training samples. Each node of a decision tree has a binary decision based whether or not $x_i \leq \gamma$ for a fixed γ (where x_i is a feature among 784 features). At each node, feature x_i and threshold γ are chosen to minimize resulting 'diversity' in the children nodes.
Each time a split in a tree is considered, a random sample of m features is chosen as split candidates from the full set of $d = 784$ features. The split is allowed to use only one of those m features. A majority vote between the trees gives the prediction for each individual.

The decision trees are built with a splitting procedure as described in the following:
Given some cell $\mathcal{C} \subset \mathbb{R}^m$ and the data $\mathcal{D} = \{(X_i, Y_i);\ X_i \in \mathcal{C}\}$. For each feature $j = 1, ..., m$ and for each value $v \in \mathbb{R}$ the data set is split into two:

$$\mathcal{D}_< = \{(X_i, Y_i) \in \mathcal{D};\ X_{i,j} < v\} \text{ and } \mathcal{D}_> = \{(X_i, Y_i) \in \mathcal{D};\ X_{i,j} \geq v\}$$

Then, parameters $p_<$ and $p_>$ are estimated as:

$$p_< = \frac{\#\{(X_i, Y_i) \in \mathcal{D}_<; Y_i = 1\}}{\#\mathcal{D}_<} \text{ and } p_> = \frac{\#\{(X_i, Y_i) \in \mathcal{D}_>; Y_i = 1\}}{\#\mathcal{D}_>}$$

The quality of the split is measured by **uncertainty**:

$$\frac{\#\mathcal{D}_<}{\#\mathcal{D}_< + \#\mathcal{D}_>} I(p_<, 1 - p_<) + \frac{\#\mathcal{D}_>}{\#\mathcal{D}_< + \#\mathcal{D}_>} I(p_>, 1 - p_>)$$

We used the Gini index so $I(p, 1 - p) = 2p(1 - p)$.
The split with the smaller uncertainty is the best.

The hyperparameter of the ***Random Forest*** method is the number of trees, denoted n. We have chosen values of n between 100 and 2000 with steps of 100. The best n of our model, chosen by cross validation is 1900.

11. The mistake in the visualization code

The error comes from the fact that the *bar* function takes scalars or sequence of scalars as argument but in the code it was given a vector. We therefore fixed the mistake by transforming the input into sequence of scalars. Therefore, the correct code line is:

axes[1, j].bar(np.arange(10), clf4.predict_proba(image.reshape(1, -1))[0])

12. Balanced accuracy score and its mathematical description.

The balanced accuracy score (BAS) in binary and multiclass classification problems helps to deal with imbalanced datasets. It is defined as the average of recall obtained on each class.

$$\text{BAS } = \frac{1}{10}\sum_{i=0}^{9} \text{Recall}_i$$

$$\text{with Recall}_i = \frac{\text{Number of pictures with original label } i \text{ and predicted label } i}{\text{Number of pictures with original label } i}$$

13. What is the confusion matrix? What are the conclusions that we can draw?

Confusion matrix is a matrix which measures the performance of machine learning classifiers. Each column corresponds to a real class and each row corresponds to an estimated class. The cell corresponding to the intersection of row R and column C contains the number of elements of the real class C which have been estimated as belonging to class R.
The balanced accuracy is 82,51%. On one hand, we can note that the number "6" is very well predicted because we observe a Recall of 100% for this number. The numbers "0" and "2" have also been well predicted because their Recall are higher than 90%. On the other hand, the number "5" is not so well recognised because there is a prediction error one time out of two.

PART 2 : Proposal of a loss function

Normally, for a classification problem, accuracy is reasonable as loss function. In our case, we need to improve the accuracy in order to take into account the fact that digits from $\{5, 6, 7, 8, 9\}$ should not be predicted to be from $\{0, 1, 2, 3, 4\}$.
The 0 - 1 Loss function and the accuracy are defined as follows :

$$\text{The 0-1 Loss function } (y_i, \hat{y}_i) = \mathbb{1}(\hat{y}_i \neq y_i)$$

$$\text{Accuracy } (y, \hat{y}) = 1 - \frac{1}{N} \sum_{i=1}^{N} \text{The 0-1 Loss function } (y_i, \hat{y}_i)$$

where N is the number of observations, y the true value and $\hat{y}$ the prediction.

Let's $A = \{5, 6, 7, 8, 9\}$ and $B = \{0, 1, 2, 3, 4\}$. Our new loss function and the new accuracy are :

$$\text{Our loss function } (y_i, \hat{y}_i) = \mathbb{1}(\hat{y}_i \neq y_i) + \alpha \times \mathbb{1}(y_i \in \text{A and } \hat{y}_i \in \text{B})$$

$$\text{Our accuracy } (y, \hat{y}) = 1 - \frac{1}{N + \alpha \times N_A} \sum_{i=1}^{N} \text{Our loss function } (y_i, \hat{y}_i)$$

where N_A is the number of digits in class A : {5, 6, 7, 8, 9} and α a number we can choose to give more or less importance to the new rule.

We chose in our new *accuracy score* $\alpha = 2$ in order to penalize more algorithms that predict digits of class A in class B.

Two algorithms were implemented: a ***SVM classifier*** and a ***random forest***. The pipeline we used is the same as the one we used before. Thus we used the *StandardScaler* of *Sklearn* to normalize the data. We present below the results of these 2 algorithms and the classifiers implemented above.

Table 1: Results

Algorithm	Test accuracy	Test Balanced_score	Our score on test
KNN	0.875	-	-
SVM Classifier	0.795	-	-
SVM classifier + Pipeline	0.840	0.825	0.907
Logistic Regression	0.84	-	-
Random Forest + Pipeline	0.935	-	0.962

Let us first note that all the hyper-parameters of our classifiers have been identified via a 3-fold cross validation. We can read in table **1** that without counting our Random Forest classifier, it is the k-nearest neighbor algorithm that obtains the best prediction score on the test sample (87.5%). With the new loss function that we have proposed, Random Forest is still more performant.