Machine Learning Project
Image Classification: Fashion-Mnist

Giovanni Dal Mas
giovanni.dalmas.1@studenti.unipd.it

1. Introduction

The Fashion MNIST dataset was proposed by Zalando research as an alternative to the classic MNIST. Considered too easy and overused, even Google research scientist and Deep Learning expert Ian Goodfellow called people to move away from MNIST dataset.[1]
Here is where Zalando's Fashion MNIST kicks in, with a slightly more challenging dataset for both ML and DL algorithms. The following project aims at addressing a clothing classification task on the Fashion-MNIST dataset. With this purpose in mind, I experimented with different ML algorithms to obtain a model with a satisfactory accuracy for the test set. Analysing the benchmark on Kaggle and GitHub, I decided to set the goal for my best model at 88% accuracy (which I was surprised to be surpassed by a couple of algorithms).

2. Dataset

The dataset consists of a training set of 60,000 examples and a test set of 10,000 examples. Each example is a greyscale image, 28 pixels in height and 28 pixels in width, for a total of 784 pixels. Each pixel has a single pixel-value associated with it, indicating the lightness or darkness of that pixel, with higher numbers meaning darker. This pixel-value is an integer between 0 and 255. The training and test data sets have 785 columns. The first column consists of the class labels and represents the article of clothing. The rest of the columns contain the pixel-values of the associated image. Each example is associated with a label from 10 classes.

0 = T-shirt/top; 1 = Trouser; 2 = Pullover; 3 = Dress; 4 = Coat; 5 = Sandal; 6 = Shirt; 7 = Sneaker; 8 = Bag; 9 = Ankle boot

2.1. Preprocessing

I loaded the data already flatten into a vector. I proceeded to normlize them to a range [0,1].The dataset does not present NaN values and it is well balanced among the classes, thus it does not require undersampling or oversampling. To improve the performance of the classic Neural Network, I also used one-hot encoding on the labels.

3. Methods and Experiments

3.1. Perceptron

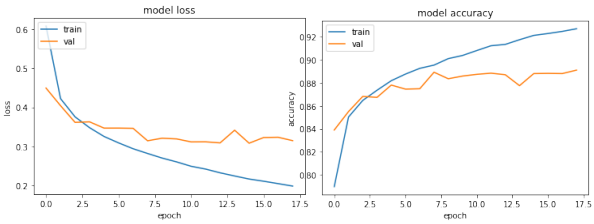
The first model I used is the simpliest one, the Perceptron. Due to its simplicity I wasn't expecting great results, but it got out with a decent 82.13% accuracy with no intercept.

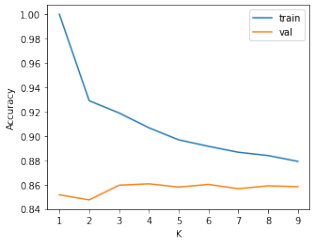
3.2. OVR and OVO on Perceptron

One-vs-rest (OVR for short) is a heuristic method for using binary classification algorithms for multi-class classification. It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident. Like one-vs-rest, one-vs-one splits a multi-class classification dataset into binary classification problems. Unlike the OVR that splits it into one binary dataset for each class, the one-vs-one approach splits the dataset into one dataset for each class versus every other class.[2]
The results I obtained using these strategies on the Perceptron were a 78.67% accuracy with OVR and a good 84.87% with OVO.

3.3. Logistic Regression

For the Logistic Regression classifier, I initially tried to fine tune the 'fit intercept'(True or False) and the C param-





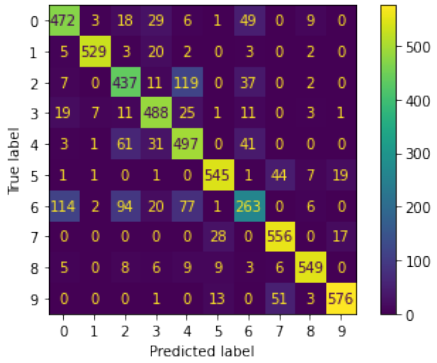
eter among 8 values from 0.01 to 20, but even by reducing drastically the possible Cs in the GridSearch, the algorithm was taking too long. This is why I decided to use a standard Logistic Regression, with default C=1.0, fit intercept = 'True' and max_iter= 20 000, that nonetheless achieved a good accuracy of 85.9%.

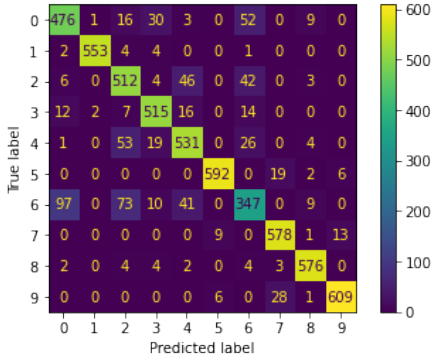
3.4. Classic NN

For the classic NN I first defined a simple Early Stopping to monitor the validation loss, with mode='min', since we are looking for decreasing patterns stop and patience=3 to make the algorithm stop after 3 epochs without improvement. I then proceeded to one-hot encode the labels. Experimenting a bit I observed that the performances do not improve with more than two hidden layers and using functions other than 'relu'. Moreover, I found a sweet spot using 64 units for both hidden layers. I tried to increase the batchsize to 32, but the optimal one turned out to be 16. I set the epoch to 200. The results were quite stunning with the model achieving 93.35% accuracy for the validation set.

3.5. K-NN

In pattern recognition, the k-nearest neighbors algorithm (k-NN) is a non-parametric method used for classification and regression. The input consists of the k closest training examples in the feature space. In k-NN classification, an object is classified by a plurality vote of its neighbors, with the object being assigned to the class most common among its k nearest neighbors (k is a positive integer, typically small). For my model I initially tried to find the best k in the range (1,100), but since it was taking way too much time, I restricted the range to (1,10). The optimal k were k=3 and k=4 that obtained the same accuracy of 86.07%.





3.6. Decision Tree

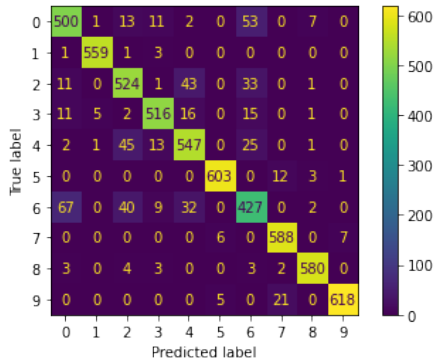
The decision tree is one of the most common algorithms for classification. To tune the hyperparameters I decided to use a Grid Search algorithm, namely GridSearchCV, that as default performs a 5-fold cross validation on the train validation set all together. The best parameter obtained for the max_depth was 10 and 'entropy' for the criterion.

3.7. Random Forrest

Random forest is an algorithm for classification, regression and other tasks, that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. The best parameters found by the GridSearch were: criterion='entropy', max depth=None, n estimators=20.
Above the confusion matrix, compared to the one from the Decision Tree algorithm, highlights how difficult it is for the models to correctly classify among T-shirt, Pullover, Coat and Shirt.

3.8. SVM

Support Vector Machines algorithm took very long to run (around 90 min) and with no surprise, since the core of an SVM is a quadratic programming problem (QP) with time complexity $O(n^2)$[3].
For the kernel function I chose the default Radial Basis



Function. To fine-tune the hyperparameters I opted for a grid search where I tuned the parameter C, that adds a penalty for each misclassified point. A low C makes the decision surface smooth, while a high C aims at classifying all training example correctly. The best C turned out to be 10. The model achieved a great 91.03% validation accuracy. From the confusion matrix we can see how the classifier struggled to correctly label the shirt, often misclassified as t-shirt, pullover or coat.

4. Conclusions

Model	Train Acc.	Val Acc.
Perceptron	-	0.8213
Percep OVO	0.8703	0.8487
Percep OVR	0.8014	0.7867
Log. Regression	0.8822	0.859
K-NN	0.9068	0.8607
Classic NN	0.9289	0.9335
Decision Tree	0.854	0.8187
Random Forest	0.9992	0.8815
SVM	0.9722	0.9103

As mentioned before the best model, based on validation accuracy, was the Classic NN that, evaluated on the test set, reached a good 88.84% accuracy.

In the table we can see all the results for the various models summarized. Overall just one methods performed under 80%. Nonetheless, the last three confusion matrices pointed out the main problem of this classification tasks, namely labeling correctly T-shirt from shirt and pullover from coat. The reasons can be found on the poor quality of the images, in some cases not easy to be distinguished even by human eye.

References

[1] GitHub official repository for Fashion-MNIST of Zalando Research: https://github.com/zalandoresearch/fashion-mnist

[2] Jason Brownlee.”One-vs-Rest and One-vs-One for Multi-Class Classification”, on April 13, 2020 in Ensemble Learning

[3] S.Surana on Kaggle: ”Computational Complexity of Machine Learning Models”