Machine Learning project report

Emma Roveroni
emma.roveroni@studenti.unipd.it

Marco Tesser
marco.tesser.1@studenti.unipd.it

1. Introduction

The machine learning project Fashion-MNIST, which is going to be explained in details in this paper, consists in an image classification challenge and its aim is to train a model that can eventually categorize precisely clothes' images, more specifically Zalando's article images. This project is designed to provide a valid replacement for the original MNIST dataset for benchmarking machine learning algorithms. The rationale behind this is that MNIST is too easy, overused and can not represent modern CV tasks. Therefore, our goal in this project is to observe how the accuracy changes when we use different types of machine learning algorithms.
In order do to this, we have defined several machine learning models, trained them with different parameters and then studied their accuracy to see which algorithm was the best. We can disclose that the models' accuracies are very similar among them and each of them has a very satisfying accuracy (typically around 0.85); SVM (support-vector machines) model has the highest one (0.9006).

2. Dataset

Fashion-MNIST is a dataset provided by Zalando Research Company. This dataset is a collection of Zalando's article images. It contains a training set of 60000 examples and a test set of 10000 examples. Each example is a 28x28 grayscale image, associated with a label from 10 classes. The classes are: t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot. In figure **1** we can see an example of how the data looks like.

We opted to do some preprocessing of the images before starting to train the models. We decided to scale each input image in order to get the dimensions in the range [0,1] instead of [0, 255]: to apply this principle we use the function **MinMaxScaler()**, available from the library *scikit-learn*, which transforms features by scaling each of them to a given range and preserving the shape of the original distribution (it doesn't meaningfully change the information embedded in the original data). We have done this because,if the features were not all in the same scale, one feature might be more influential than the others thus resulting in adding bias

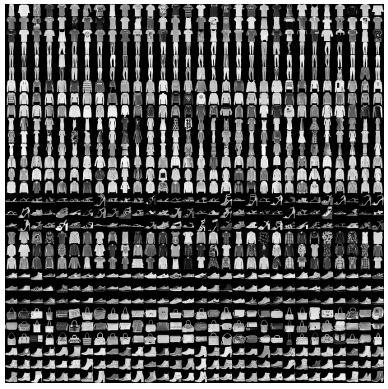

Figure 1. **Fashion-MNIST Dataset Images**

to the model. Also when the values of the features are closer to each other, the algorithm will train better and faster.

Then we decided to use PCA (principal component analysis), which is a method often used to reduce the dimensionality of large data sets by transforming a large set of variables into a smaller one that still contains most of the information. We chose PCA because, even though reducing the number of variables of a data set comes at the expense of accuracy, the dimensionality reduction is to trade a little of accuracy for a faster execution time. Finally, only for the Neural Network model we applied the one-hot encoding technique for the labels. This process allows to transform the label from an integer (that, in this case, is in the range [0,9]) to a vector which includes a '1' for the corresponding label position and the rest of the vector's elements are filled with '0'.

3. Method

The method we adopted to do this project is to try to train different types of machine learning algorithms and then observe how good they were through one metric: the accuracy. The model we decided to train are: logistic regression, k-NN, neural networks, decision tree, random forest and SVM. For each one of these we applied a grid search which does an exhaustive search over specified parameters' values for an estimator. The kind of parameters searched by

the grid search are different and specific for each model.

4. Experiments

1. **Logistic Regression** The parameters we decided to examine in this model are two: the parameter 'C', which is the inverse of regularization strength, and 'fit_intercept', which is a boolean that indicates if a constant (bias or intercept) should be added to the decision function. The result we obtained through the grid search is that the best parameters are: 'C'=0.5 and 'fit_intercept'=**TRUE**. The resultant accuracy is 0.8438, which is pretty high.

2. **k-NN** The only parameter we analyzed with k-NN is k, which is the number of data points that will be considered neighbors for the computation. We started to examine different ranges for k, using initially a for cycle. In this way we were able to observe how the accuracy changed with the increase of k through a plot like the one in figure **2**. The initial range used for the parameter

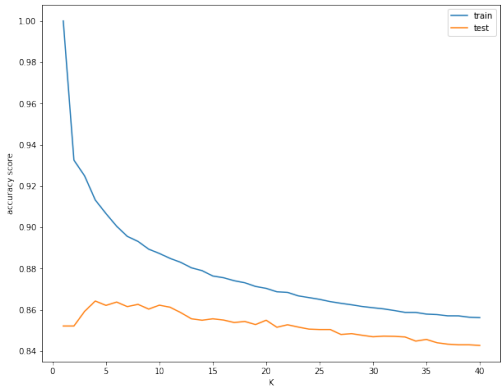

Figure 2. **k-NN accuracy score**

k was quite big, from 1 to 40, then, since we observed that after a few values the accuracy was decreasing, we chose to reduce k's range. So, we switched to the grid search method instead of using a for cycle since the former is more precise. In this way we obtained which k was giving us the best accuracy. For k equal to 8 the model had an accuracy of 0.8627.

3. **Neural Network** For what regards Neural Networks things have been done in a slight different way. Since it's impossible to implement a grid search with a non Sci-Kit learn based model (NN are built with Keras) we had to find the best hyperparameters using for cycles. As the very first thing though, we had to set our parameters for the NN. So we kept a variable for the totality of features of the dataset (after PCA) that will be used for implementing the input layer of the neural network, and in another we had the target data converted with one hot encoding to vectors of size of the number of classes. Then we defined a function called NN_Builder to create and compile the various models that were needed for the validation part. The hyperparameters that we wanted to test were: number of neurons, number of layers and batch size. The number of neurons in NN_builder was implemented as to decrease each layer after the first layer of a given number of neurons (depending on the model) to then finally have in the last hidden layer 20 neurons. So in every round of a nested for cycle what was tested was: a model of NN built with NN_Builder with a given number of neurons and a given number of layers compiled with a standard compiler and then fitted with a given batch_size. Histories, fitted models and val_accuracies' best results were kept for every model in separate lists to determine at the end which was the best model. A mechanism of EarlyStopping was also included to stop the computation after 3 consecutive training phases without a minimum improvement in accuracy. Finally out of all of these models we established that the best one was the one with 4 hidden layers, 150 neurons and batch size equal to 8, with an accuracy of 0.8725.

4. **Decision Tree** With the decision tree model the parameters to study were the following:

   - *max_depth*: the maximum depth of the tree;
   - *criterion*: the function to measure the quality of a split, which can be 'gini' (Gini impurity) or "entropy" (information gain).
   - *min_samples_split*: the minimum number of samples required to split an internal node.

   The best parameters' configuration we obtained through the grid search is the successive one: max_depth equal to 10, min_samples_split equal to 6 and criterion as entropy. Even though we tried really big ranges of values for every parameter, the best accuracy the model could reach is 0.7796, which compered to the accuracy of the previous (and following) models is the lowest. We can conclude that the decision tree classifier, with the Fashion-MNIST dataset, is surely not the best one to use.

5. **Random Forest** In this model the parameters we considered were pretty much the same as in the decision tree model. The only difference was that instead of observing the min_samples_split, whose value we left as the default one (2), we analyzed the *n_estimators*, which stands for the number of trees in the forest. The result we got from the grid search is the following: max_depth equal to None (which indicates that

the nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples), n_estimators equal to 20 and criterion as entropy. The accuracy with the best parameters found by the grid search is 0.8392, which is very alike to the other models.

6. **SVM** The parameters observed for the SVM with the grid search were two: the 'C' value and 'kernel', that specifies the type of kernel to be used in the algorithm. The kernel methods we considered were 'rbf' (radial basis function) and 'linear'. The best parameters found by the grid search were 10 for C and 'rbf' for the kernel. The accuracy of SVM with this parameters was the highest one: 0.9006. We plotted the following confusion matrix to observe better how SVM model actually behaved with Fashion-MNIST dataset.

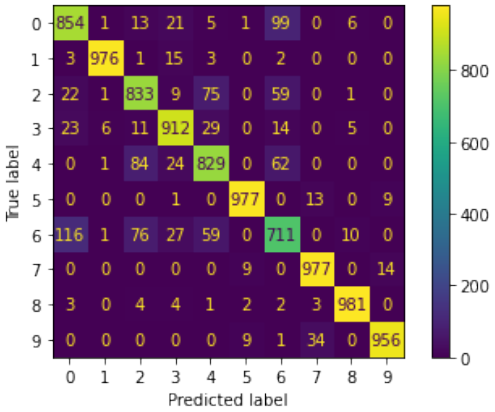

Figure 3. **Confusion matrix for SVM**

We can observe that the most misplaced prediction was that of label number 6, which corresponds to a shirt. This is understandable due to the fact that the items which were predicted instead of shirt were all upper body clothes that are actually similar to a shirt.