Machine Learning Project Report
"Fashion-MNIST Challenge"

Luca Dal Zotto
luca.dalzotto.1@studenti.unipd.it

Abstract

Computer vision is one of the hottest research fields in Machine Learning. The goal of this paper is to compare the effectiveness of some of the most famous classification algorithms for this kind of problem. In detail, we are going to face the "Fashion-MNIST" dataset, a collection of Zalando's article images, regarding 10 different classes (pieces of clothing). The analysis involves some simple algorithms, but also more sophisticated ones, up to a state-of-the-art approach: in details, Logistic Regression, KNN, Random Forest, SVM and CNN are considered. The comparison is carried out from different point of views, in order to highlight the strengths but also the limits of these algorithms. The experimental results are mostly in line with their theoretical characteristics, confirming the superiority of SVM and CNN in terms of accuracy.

1. Introduction

Zalando is one of the largest European online fashion platforms. Every product has a set of pictures demonstrating different aspects of the item. In [1], Xiao et al. explained step by step how this new dataset ("Fashion-MNIST") was generated starting from that collection of images.

Figure 1: Examples from the dataset.

The intention of the authors was to provide a more challenging alternative (compared to the original MNIST dataset) for benchmarking machine learning algorithms. Different models have been tested on this dataset, especially deep learning architectures. For example, I could mention a study from a research group of the university of Sydney, where a Random Forest Classifier, a SVM and a CNN were trained and compared in depth [3]. For what concerns CNN, the authors implemented two small networks (2 and 3 layers resp.) but still able to achieve a test set accuracy of 90.05 and 91.17%, outperforming both the SVM and the random forest models.

Another piece of work that inspired some ideas of this project was carried out by Greeshma and Sreekumar [4]. In this case, they focused only on CNN, exploring the impact of various hyper-parameter optimization methods and regularization techniques with deep neural network. In particular, they showed how by fine tuning their network the overall performance increased and the training time significantly decreased.

The aim of this project is to train different machine learning classifiers, paying attention to the hyper-parameters and focusing on the pros and cons of each method. In section 2, a more precise introduction to the dataset is given, making some data exploration and also some basic pre-processing operations. In section 3, the approach used for the analysis is discussed, presenting all the considered tools. Then, in section 4, the results are listed along with some comparisons. Finally, in section 5 some final remarks are made.

2. Data exploration and pre-processing

The dataset is freely available online at the link https://github.com/zalandoresearch/fashion-mnist but for this project, it has been provided through Kaggle, where it was already split in training, validation and test set, consisting respectively on 50,000, 10,000 and 10,000 samples. Each of them is a 28x28 gray scale image that has been flattened into a 784 dimensional vector. The value of a pixel channel is between 0 and 255, therefore it is convenient to normalize these numbers dividing by 255.

The label vector is one-dimensional and its values are the positive integers between 0 and 9, each one corresponding to a different piece of clothing: 'T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot'.

The different classes are almost equally represented in both the training and validation set (Table 1). Therefore, having a balanced dataset, no counter-measure such as

under-sampling or over-sampling is necessary. Moreover, the accuracy (fraction of correctly predicted samples) reflects well the performance of a model, so it can be selected as evaluation metric.

	Label	Training set frequency	Validation set frequency
T-shirt/top	0	4977	1023
Trouser	1	5012	988
Pullover	2	4992	1008
Dress	3	4979	1021
Coat	4	4950	1050
Sandal	5	5004	996
Shirt	6	5030	970
Sneaker	7	5045	955
Bag	8	5032	968
Ankle boot	9	4979	1024

Table 1: Training and Validation set class frequency.

3. Method

For the Python-based implementation, I used the scikit-learn library [5], which also allows to choose the most relevant hyper-parameters of the models. These are some values which allow to fine tune our model, adapting it to the particular problem at handle. Their values are chosen when the model is initialized, and do not change during the training phase (unlike the parameters).

In order to find the best combination of hyper-parameters, the most common strategy is Grid Search. Essentially, it consists on defining a list of values for each hyper-parameter, fitting a different model for each possible combination and evaluate them on a validation set. Since this strategy involves training multiple models, it is clear that the computational cost is very high, especially in the big data scenario. This is the reason why in this project, as preliminary phase, two subsets of the training and validation set (10 times smaller) were extracted and a first grid search was performed using them.

This procedure has two drawbacks: firstly, the extracted subsets could represent the original dataset not accurately. Therefore, I selected these subsets in a stratified fashion, so every class is equally represented on them. Secondly, of course, we have no guarantee that this procedure gives us the optimal combination of hyper-parameters for the original dataset. For this reason, this first grid search should be seen as a way to have a first idea, in broad terms, of where the optimal region of hyper-parameters for the original dataset should be located. Therefore, for the original dataset, instead of carrying out a complete, exhaustive (and computational prohibitive) grid search, we may focus only on few values close to this region. Anyway, since the size of these subsets is still large (6,000 samples), I do not expect to see big changes of the optimal values.

The first algorithm considered is Logistic Regression. The hyper-parameters considered in this case are:

- **reg_term:** inverse of regularization strength;
- **max_iter:** maximum number of iterations.

Secondly, I considered a non parametric model: k-Nearest Neighbors. In this case, the grid search was one-dimensional since it involved only the number of nearest neighbors to find at prediction time, k. The default euclidean distance has been used to compute the distances.

The hyper-parameter considered for Random Forest are:

- **n_estimators:** the number of trees in the forest;
- **max_depth:** the maximum depth of the tree.

Note that if n_estimators is set equal to 1, what we get is a decision tree. Then Support Vector Machine was considered. The grid search was conduced over:

- **C:** regularization parameter;
- **kernel:** kernel type to be used in the algorithm.

	hyper-parameter	values
LR	*reg_term*	[0.001, 0.01, 0.1, 1, 5, 10, 20, 40, 60, 80, 100]
	max_iter	[4, 8, 16, 32, 64, 80, 96, 114, 128, 150]
KNN	*n_neighbors*	[1, 2, 3, 5, 10, 20, 30, 50]
RF	*n_estimators*	[1, 20, 50, 75, 100, 130, 165, 200, 250, 500]
	depth	[10, 20, 50, 75, 100, 130, 165, 200]
SVM	*C*	[0.01, 0.1, 1, 10, 50, 100]
	kernel	['linear', 'poly', 'sigmoid', 'rbf']

Table 2: Hyper-parameters' values considered.

Lastly, I also considered a deep learning architecture in order to verify how much difference there is between shallow machine learning algorithms and *almost* state-of-the-art techniques. The CNN architecture used in this project consists on:

- 4 2D convolutional layers with a kernel of size 3x3. The first layer has 32 filters, the others 64. To avoid overfitting, dropout is implemented. Moreover, max pooling is applied to the last two layers to increase the spatial invariance. All activation functions are "relu";
- 2 dense layers. The first one with 128 units and hyperbolic tangent as activation, the second one with 10 units (as the number of classes) with the softmax activation function.

Adam algorithm is used as optimizer and sparse categorical cross-entropy as loss function. The batch size

is 64, like the maximum number of epochs. Indeed, the "EarlyStopping" utility is used to interrupt the training phase if no progress on the validation set is measured for five consecutive epochs [6]. Moreover, "ModelCheckpoint" saves the model with the best validation accuracy so far [7].

4. Experiments

In this section, the most relevant results of the analysis are reported. Since the workflow is exactly the same for the first four models, here I report all the steps of just the first one, Logistic Regression. First of all, I implemented manually the grid search for the reduced dataset. Then I looked at the performances of each combination.

In the case of Logistic Regression, it is possible to make the following observations: for values of reg_term smaller that 0.1 (stronger regularization), both training and validation accuracy are low. This is clearly an index of underfitting. On the other hand, for values larger that 50, the training accuracy is very high (close to 100%) while the validation accuracy is not that good. This means that our model is learning the peculiarities of the training set, loosing its generalization capacity (overfitting). Therefore, it seems that a good region for reg_term is between 5 and 40. For what concerns the number of iterations, the training curve is monotonically increasing, while the validation accuracy is higher between 32 and 128.

So, with the original dataset, I tried some values in these ranges, finding out that the optimal combination was with reg_term = 10 and max_iter = 128, leading to a validation accuracy of 0.851.

For the other models the reasoning was exactly the same, with the exception of CNN, which was implemented using Keras [8]. Since my computational resources were limited, I tried just a couple of different architectures, so the results obtained might be easily overcome. Remember that the aim of this part is just to make a comparison between a reasonably good deep learning model and the previous ones.

Figure 2: Learning curves (CNN model).

In Figure 2, it can be observed that after a certain number of epochs, the validation accuracy reaches a local maximum that is no longer overcome for five epochs, so, training is interrupted to avoid overfitting.

To summarise, the final results are reported in Figure 3.

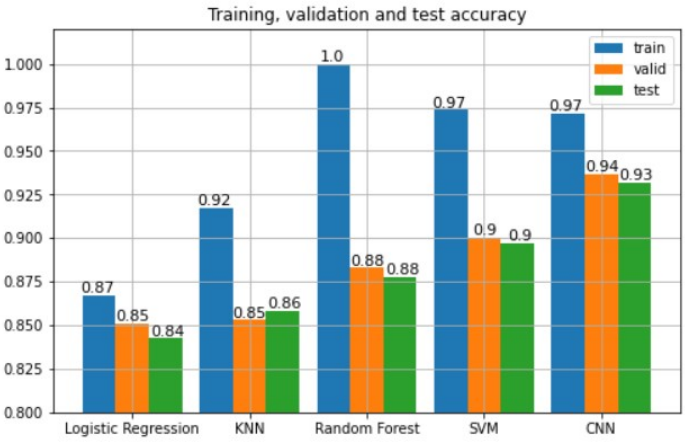

Figure 3: Model comparison.

We can see that CNN is much better that the other models, with a test accuracy of 93.2%. SVM also reaches a good result (slightly smaller than 90%). The performance of Random Forest is reasonable (87.7%) even though overfitting is still a problem, while KNN (85.8%) and Logistic Regression (84.2%) results are lower.

In order to have also a rough idea of the computational cost of these model, both the fitting and predicting running time were computed. I found out that SVM has the slowest training phase, followed by CNN. It is interesting to see that KNN has an extremely small fitting time, while the predicting time is huge. Remember that the training phase consists just in storing the training samples, while it is at fitting time that all pairwise distances are computed.

For the sake of completeness, the confusion matrix was computed for the best model (CNN). It should be highlighted that the most mis-classified category is 'Shirt' (only 794/970 correct predictions), which is confused mainly with 'T-shirt/top', 'Pullover' and 'Coat'.

5. Conclusions

In this project I faced an image recognition problem, testing the effectiveness of some famous Machine Learning tools. For technical details about the implementation and to consult additional plots and results, the reader is invited to consult the Jupyter notebook "LucaDalZotto_MLproject". To sum up, I found out that Logistic Regression performs poorly. KNN has too high computational requirements and a low accuracy too. Random Forest is not bad but SVM is significantly better. However, a tool that is very often applied for this kind of tasks like CNN outperforms all these models with a test accuracy of 93.2%.

References

[1] Han Xiao, Kashif Rasul, Roland Vollgraf, *Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms*, 2017.

[2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, *Imagenet classification with deep convolutional neural networks*, 2012.

[3] D. Bridgman, B. Lindsay, S. Shen, *Deep Learning vs. Linear Classifiers on the Fashion-MNIST Dataset*, 2019.

[4] Greeshma K. V., Sreekumar K., *Hyperparameter Optimization and Regularization on Fashion-MNIST Classification*, 2019.

[5] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, 2011.

[6] Aurélien Géron, *Hands-on Machine Learning with Scikit-Learn, Keras TensorFlow*, O'Reilly, 2019.

[7] Jojo Moolayil, *Learn Keras for Deep Neural Networks*, Apress, 2019.

[8] Chollet, François et al, *Keras*, `https://keras.io`, 2015.