Evaluating Machine Learning Models for Fashion Recognition

Mattia Brocco
mattia.brocco@studenti.unipd.it

1. Introduction

Image classification is the primary task of computer vision, which is precisely the field the task at hand falls in. Computer vision is evolving rapidly also in the fashion industry, to the point that deep learning algorithms have been already implemented in this industry, also for purposes similar to the one considered in this study, namely "low-level fashion recognition", i.e. computing and processing fashion images at the pixel level [1]. However, even in a business that in 2018 accounted for more than 3 trillion dollars [2], the use of manpower for recognition in fashion (and textile) is still widespread and this may cause issues in terms of efficiency and automation level, which do not always comply with the speed and consistency that the actual globalized world requires, and therefore inevitably increase the level of stress and fatigue on human resources [3].

This work, on the one hand, fits in a context in which algorithms for low-level clothing and accessories recognition are spurring (e.g. Convolutional Neural Networks are very close to the state of the art for this application [1]), and on the other hand, the data on which this study is built is widely used to assess the performance of state of the art models devoted at image classification.

The primary objective of this study is to find the family of algorithms, among those presented during the lectures of this course, that present the most favorable trade-off between classification performance and prediction time, with particular attention to multilayer perceptron. The secondary objective is to obtain an implementable algorithm that performs better than the baseline human performance, which is attested to be around 83.5% [4].

The assessment has brought to life as, among the families of models analyzed and the architectural choices that they have been subjected to, a Random Forest of 50 estimators with gradient descent reaches an accuracy of 88.52% on the test set, with a rate of more than 69.000 predictions per second.

2. Dataset

The dataset provided is known in the Machine Learning community as the replacement of the historical MNIST (modified National Institute of Standards and Technology) database. The latter is a large database of handwritten digits, used to train image processing systems and to benchmark them.

The Fashion MNIST, how the dataset at hand is also known as, is meant to provide a harder task than digits recognition in order to better reflect the actual problems that Computer Vision is facing [4]. However, the Fashion MNIST serves the same purpose as the MNIST database, thus, comparing supervised learning algorithms.

The data provided is in the form of a matrix of 60.000 rows and 784 columns (for training purposes), plus an additional 10.000 rows for testing. To speed up the repeated procedure of data import, data has been exported into feather file format, that can be easily read through Pandas in far less time than traditional file formats.

The possibility to work with a format that is already in a machine-readable format has speeded up the data cleaning operations. This because the original dataset is made of images, while here every images has been already broken down into 784 pixels (since the original images are 28x28 pixels), each corresponding to a number from 0 to 255 – that is the RGB codification for colors. To both training and testing dataset is associated an array with the corresponding lengths, that contain the labels for the supervised learning procedure. Labels go from 0 to 9, where each number stands for a category of clothes or accessories (e.g. trousers, coats, bags).

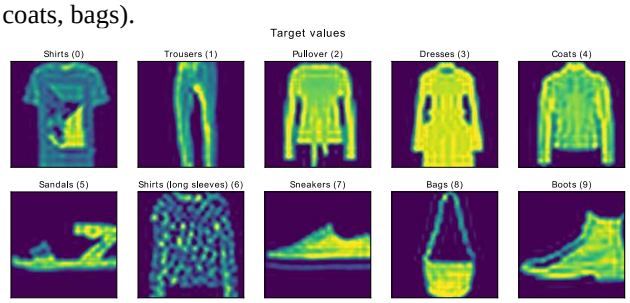

Figure 1: Target labels and significance

As a whole, the training dataset presents itself as a quite sparse matrix, as the distribution of values is highly left-skewed towards zero. However, data was perfectly clean, with no missing values and no other concerning values (e.g. values outside the [0, 255] interval).

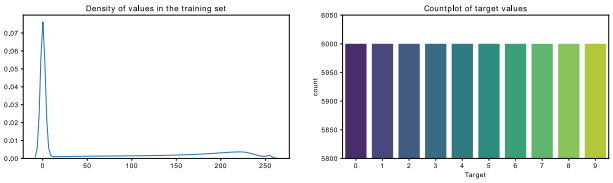

Figure 2: Density of values and labels distribution

The same stands for the training labels, as they are perfectly balanced, thus requiring no further techniques to re-balance the proportions between the target labels.

Finally, as no exclusion of data was necessary, the only preprocessing technique I've decided to apply is a min-max scaling to obtain all values comprised in the interval [0, 1].

3. Method

Throughout the study, the approach followed has been that of systematic evaluation of several algorithms that have been covered during the course, starting from the same extent of data preprocessing, in order to ensure a common starting point for all of them. The model used are:

- Perceptron
- Logistic Regression
- Support Vector Classifier (Kernel method)
- Decision Tree Classifier
- Random Forest (ensemble learning)
- Multi-layer Perceptron (neural network)

Accordingly, for all the algorithms considered, the procedure adopted involves the training of a basic model in order to assess if it is suitable for the purpose in the first place, second, the tuning of hyperparameters to estimate the extent to which it is possible to enhance the performance of the model used at first, and last, evaluate each new model that had its hyperparameters tuned on the basis of accuracy and precision on training, and prediction time.

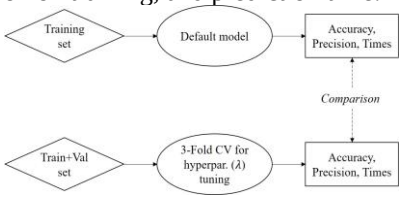

Figure 3: Process and data usage

The selection of the starting models and the order maintained tries to match both the order with which the models have been presented in class and my previous experience with supervised learning in order to obtain always more performing models.

What matters more than the order of the algorithms, however, are the metrics used to evaluate them and why these have been chosen for evaluation. In the first place, the metrics used are (1) accuracy, which represents the frequency of the algorithm being correct at prediction, (2) precision, defined as the frequency of being correct on predicted positive, and will be used as a proxy to represent the frequency of being correct on a given class, broadly speaking, (3) prediction time, expressed in predictions per seconds.

The reason for these choices comes from the need to address real-world problems in the most consistent way possible and, respectively, accuracy is a convenient score for the evaluation of multiclass classification problems, since it conveys information on the model's general performance (recall that it's defined as the frequency of being correct on the whole data given), considering that the confusion matrix of a multiclass problem may not be of immediate comprehension (in this case it is a (10x10) matrix). The second metric has been chosen for the particular scope of this task: out of the 10 classes, some are trivially identified since they are completely different from the others (e.g. sandals, bags, trousers), but what makes the difference here is being also able to discern between similar classes: in particular, between pullovers (2), coats (4) and long sleeves shirts (6). Third, the time a model takes to predict new data may come in handy for applications such as web platforms in which users frequently upload clothes images (e.g. second-hand resale platforms such as Vinted) without specifying the type of product they upload; therefore, even if real-time prediction may seldom be required, slower algorithms may impact the overall User Experience.

Another consideration should be made on how data has been used for the process. The training set of 60.000 images has been split into two parts (75% for training and 25%) for validation. The basic models have been trained on the training set, in order to find the parameters for this set of data, and a first feeling on the adequacy of the model is drawn by computing the accuracy on this training set. However, since the training times on the machine used are quite high, to reduce the overall time required for hyperparameter tuning, this hold-out approach has been replaced by 3-fold cross-validation that exploited both the training and the validation set (i.e. the 60.000 rows dataset), this way an estimate of the model's true performance is still somehow reliable (3-fold has been chosen again for hardware constraints).

Finally, the way which data is portioned is quite in accordance with the standard procedure used for machine learning studies, however, at a higher level, further concerns could arise on the data handling and the algorithms used: (1) dimensionality reduction techniques have been neglected in the study, and prior studies on the topic at hand has defined it as a good practice [5], and (2) lazy learners have been neglected as well, and prior experiments on the same set of data showed how the use of such learners (i.e. K-NN) yielded good performance, even though at the expense of high prediction time [6].

4. Experiments

The experimentation has started following Occam's razor, thus with a simple model, specifically the Perceptron. The choice comes also from possible exploitation of the "curse of dimensionality", according to which, as the dimensionality of feature space increases, data points become sparser and, considering the nature of the problem, classes may also be intrinsically linearly separable – which is crucial for such a linear classifier. In a multiclass problem, one of the first considerations is the strategy with which it is addressed with a binary classifier. In this sense, One-vs-One (training K(K-1)/2 binary classifiers, one for every pair of classes) and One-vs-Rest (training K binary classifiers, one for every class) have been tried out and the latter proves to have a better Train+Val accuracy (82.86%) – Train+Val as Scikit-Learn [7] allows the perceptron to have a validation subset kept during training. The next step is less trivial, as hyperparameter tuning is performed for regularization: in this case, the starting model was not overfitting, thus regularization has been performed merely to determine its impact on this specific situation. Unsurprisingly, the accuracy of the regularized perceptron has decreased (78%) as it has been prevented from becoming more flexible [8].

The second model tested is the Logistic Regression, and the process followed complies with the description provided in the third paragraph. Hence, a "default" model is trained and evaluated on the training set (87.67% train accuracy), and again, the scope of the hyperparameters selection falls within the scope of regularization. The circumstance is akin to what is seen for the perceptron, that is a non-overfitting model to which regularization is applied: accuracy in the Train+Val set does not increase as learning is discouraged to capture more variance, leaving a new accuracy almost equal to that of the initial model. However, if regularization has been tested to be self-defeating for non-overfitting models, the main drawback that logistic regression presents is the absurd fit time; prior evidence on the matter is provided [9], but probably the size of the training set is to blame since it also impacts the number of iterations to reach convergence (that has been increased w.r.t. default).

Next comes the turn of a Kernel method, specifically a One-vs-Rest Support Vector Classifier exploiting RBF (radial basis function) Kernel. The result of the first initialization of this model seems encouraging (91% train accuracy), but it may raise concerns about the overfitting of the model. However, by increasing the C parameter to reduce the margin width, the accuracy is skyrocketing (with 3-fold cross validation 97% training accuracy is reached with C = 10). Thus, no further hyperparameter tuning has been performed. However, the increased accuracy provided by the regularization may be further empirical evidence that data may be transformed into linearly separable [10]. The main concern of the SVC is the prediction time, even with fewer support vectors thanks to regularization, as it is around 70 pred./sec., even if it is the best performing model so far. This is probably to the high number of support vectors.

The fourth family of classification algorithms taken into consideration is that of the Decision Tree. The base model deployed shows a 100% accuracy on the training set, which is expected because if the splitting of data is not stopped, the model correctly classifies each sample (no impurity is left). Therefore, ex-post pruning is performed, i.e. the tree is allowed to grow to full depth, and then some branches are removed to prevent overfitting [11]. The technique adopted is cost complexity pruning, thus the hyperparameter to consider is α, which is the cost complexity parameter, and is inversely proportional to the tree's size. With cross-validation, it can be found $\alpha = 4.13e^{-4}$ and the resulting accuracy will be 85.62%.

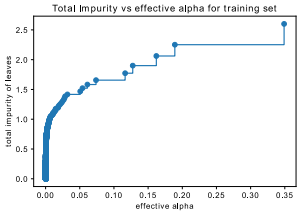

Figure 4: Impurity of leaves against α

Starting from the decision tree, it is possible to exploit ensemble learning, i.e. the combination of several weak learners. Here, a Random Forest using gradient boosting (sequential ensemble) with 100 estimators, each with the maximum depth of the tree found above is tested. As for the tree, overfitting is expected (99.72% train accuracy) and has been reduced by the reduction of the number of estimators exploited (from 100 to 50): Train+Val accuracy is dropped to 95.85%, but the learning curve shows only a very slight presence of overfitting. For the experimentation, the LightGBM [12] library (open-source library developed by Microsoft) is preferred to speed up training time, w.r.t. standard Sklearn Random Forest class.

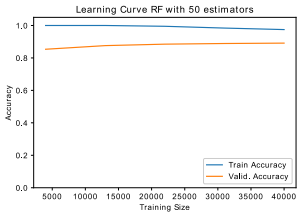

Figure 5: Random Forest's learning curve after CV

Lastly, a multilayer perceptron is subject to experimentation. In this case, the analysis process has been slightly different from the other models: the underlying assumption is the use of a feed-forward computation dense network, and given that, choices on the architecture of the network are done sequentially: first the number layers, then

the number of units per layer, and last the sequence of activation functions. The progressive choice is made on the basis of the validation accuracy and on the learning curve pattern.

At first, a series of layers from 2 to 21 have been trained and compared, and the choice has fallen on 5 layers as they provide the best validation accuracy and show a learning curve's regular pattern. Next, the choice of the number of units per layer is done under the assumption of a fixed number of units throughout the hidden layers of the network, and given the task, the number of units did not show a drastic impact on performance, but, since the behavior of the model tends to overfit as units increase (as shown below), 50 units per layer have been selected.

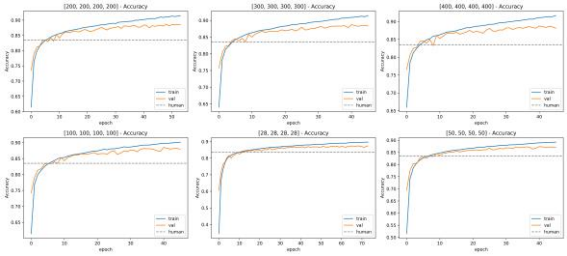

Figure 6: Learning curves w.r.t n° of units per layer

Third, the choice of activation function has been carried out by considering only rectified linear unit (ReLU) an exponential linear unit (ELU), with the additional assumption that ReLU activation in the input layer is the common practice [13]. The information conveyed by both validation accuracies and the learning curves provided evidence for the set of activation functions that follows: (1) ReLU (2) ELU (3) ReLU (4) ReLU (5) Softmax. This architecture shows a Train+Val accuracy of 88.86% and the learning curves show no evidence of overfitting.

5. Conclusions

Only 3 models have been also compared on the test set given their previous performance so far: SVC, Random Forest and MLP. The first is the one that exhibits the best performance in terms of both test accuracy (90.04%) and precision on most relevant classes (75% on coats, the "hardest" class). However, it shows a rate of prediction of ~65 pred./sec., which makes it hardly suitable for real world applications, considering the difference with the other two models' prediction rates. MLP shows a test accuracy of 86.47%, a precision on coats of 67% and a prediction time of more than 22.000 pred./sec.

Lastly, the model that shows the most addressing performance is the random forest, with a test accuracy of 88.52%, a precision on coats of 70% and a prediction time of more than 69.000 pred./sec.

References

[1] U. Raghava, Deep Learning in Fashion Industry, Medium, April 2021, https://medium.com/analytics-vidhya/deep-learning-in-fashion-industry-dcb897ac3c33

[2] X. Gu, F. Gao, M. Tan, P. Peng, Fashion analysis and understanding with artificial intelligence, Information Processing & Management, Volume 57, Issue 5, 2020

[3] J.J. Wen, W.K. Wong, Fundamentals of common computer vision techniques for fashion textile modeling, recognition, and retrieval, Editor(s): W.K. Wong, In The Textile Institute Book Series, Applications of Computer Vision in Fashion and Textiles, Woodhead Publishing, 2018, pp. 17-44

[4] Zalando Research, Fashion mNIST repository, Github www.github.com/zalandoresearch/fashion-mnist

[5] J.J. Wen, W.K. Wong, Fashion accessory segmentation, Editor(s): W.K. Wong, In The Textile Institute Book Series, Applications of Computer Vision in Fashion and Textiles, Woodhead Publishing, 2018, pp. 221-252

[6] G. Carbone, Fashion mNIST challenge – kNN, Kaggle, 2021 https://www.kaggle.com/gcarbone/gabriele-carbone-fashion-mnist-challenge-knn

[7] L. Buitinck et al., API design for machine learning software: experiences from the scikit-learn project, European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases, 2013

[8] P. Gupta, Regularization in Machine Learning, Medium, November 2017, https://towardsdatascience.com/regularization-in-machine-learning-76441ddcf99a

[9] R. Wang, N. Xiu, S. Zhou, An extended Newton-type algorithm for ℓ2-regularized sparse logistic regression and its efficiency for classifying large-scale datasets, Journal of Computational and Applied Mathematics, Volume 397, 2021

[10] M. Mazuecos, How to check for Linear Separability, Medium, May 2019, https://maurygreen.medium.com/how-to-check-for-linear-separability-13c177ae5a6e

[11] S. Kumar, 3 Techniques to Avoid Overfitting of Decision Trees, Medium, May 2021, https://towardsdatascience.com/3-techniques-to-avoid-overfitting-of-decision-trees-1e7d3d985a09

[12] Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., … Liu, T.-Y., Lightgbm: A highly efficient gradient boosting decision tree. Advances in Neural Information Processing Systems, 30, 2017, pp. 3146–3154.

[13] I. Goodfellow et al., Deep Learning (Adaptive Computation and Machine Learning series), MIT Press, 2016