ML Project: Image Classification

Bertipaglia Beatrice Sofia
beatricesofia.bertipaglia@studenti.unipd.it

Sartori Francesco
francesco.sartori.8@studenti.unipd.it

1. Introduction

Image Classification is one of the most fundamental tasks in *computer vision*, consisting of associating one or more labels to a given image. In this project we will focus on the *single-label classification*, i.e. all the models we will search for output a single prediction for each image.
We will work on the Fashion MNIST database in order to find the best model that classify this type of images. Our goal is not just to find the most accurate model, but to find a compromise between accuracy and computational cost in order to be more practical to a possible real world use. This means that we will not want to choose the model with the best score, but the one with a good accuracy and a reasonable running time.
To find the best model that classifies our images we will compare four different classifiers: *AdaBoost Classifier with base estimator Decision Tree, Ridge Classifier, Support Vector Classifier* and *Neural Network*. Also for each one we will try to find the best way to implement a image classification model, i.e. find the best parameters to use in the classifier.

2. Database

The Fashion MNIST database is a large database containing 70,000 images of Zalando's articles. Each element is a 28x28 grayscale image, represented as a vector of 784 numerical features, one for each image's pixel. The elements are divided in 10 different classes: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag and Ankle boot. Each class has approximately the same amount of images, around 10% of the database.
The database is, by default, splitted into a training set of 60,000 examples and a test set of 10,000 examples.
Having a balanced database it's not necessary to apply oversampling or undersampling techniques. Because the features belongs to a large interval of values, we transformed them by scaling each feature to a given range, i.e. the interval between zero and one, using the already implemented *MinMaxScaler*. We needed to scale the features because many models have difficulty to process large numbers, giving convergence errors.
Initially we tried to filter the features using *entropy* to see which pixels were more relevant in the classification problem, we obtained slightly worse results in favor of smaller running times, so we decided to preserve all our data without filtering.

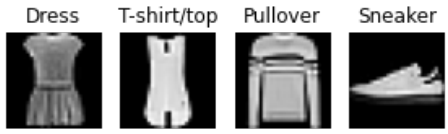


Figure 1. Example of some elements of the database.

3. Methods

Our approach for solving the classification problem consists in the implementation of very different, in terms of the theory and the algorithms behind them, types of classifiers:

- AdaBoost Classifier;
- Ridge Classifier;
- Support Vector Classifier;
- Neural Network.

We want to see which one fits better our image classification problem, i.e. finding the model with the best performance.
For each classifier we implemented a *GridSearch* to find the best values for the parameters that we considered more relevant. Then we trained the classifier with the best parameters on the entire training set in order to obtain the model with the best performance for the training and test accuracy score and for the running time. In the end we checked the distribution of the predictions over the 10 classes, we saw that all four models misclassifies the same one class more than the others.

3.1. AdaBoost Classifier

The first method we implemented use the AdaBoost Classifier, an *ensemble* algorithm that combines the outputs of *weak learners* (small decision trees) into a weighted sum

to improve performance. We chose AdaBoost because is an *adaptive* algorithm, i.e. subsequent weak learners are adapted in favor of those instances misclassified by previous classifiers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but as long as the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner. AdaBoost classifier is a classifier in the form

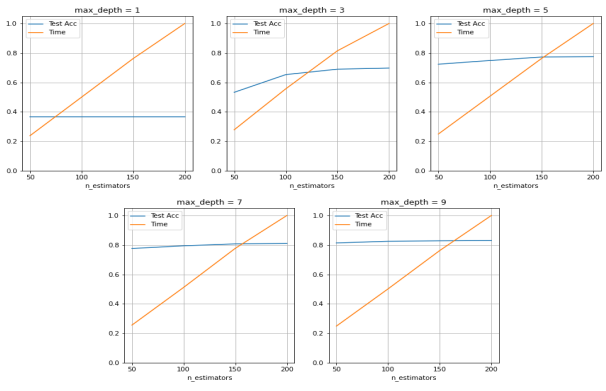
$$F_T(x) = \sum_{t=1}^{T} \alpha_t h_t(x)$$

where h_t is the output hypothesis on the training set, T is the number of estimators and to each estimator is assigned a weight computed as

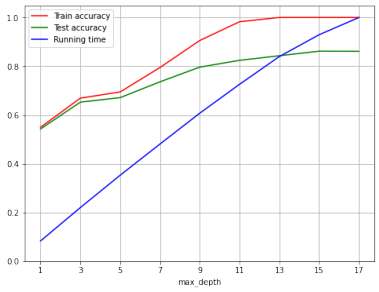
$$\alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t}$$

whit ϵ_t the weighted error rate of the weak classifier.

We saw that training an AdaBoost Classifier was quite time consuming, so to start we chose to train our model on a 5000 units training set in order to determine which choice of parameters improved the most the performance of the final model. The values we considered for the parameters were $[1, 3, 5, 7, 9]$ for the maximum depth of our base decision trees, and $[50, 100, 150, 200]$ for the number of estimators.
We saw that the number of estimators did not improve significantly the accuracy score on the test set, but the running time is basically linear on the number of estimators. Because of that we chose to search for the best value for the maximum depth using 50 estimators.
We then trained our classifier on the entire dataset to have



a realistic result, with maximum depth taking values from 1 to 17. The accuracy on the test set increased until the maximum depth reached 15, then started decreasing. We assume that the model started overfitting the data.



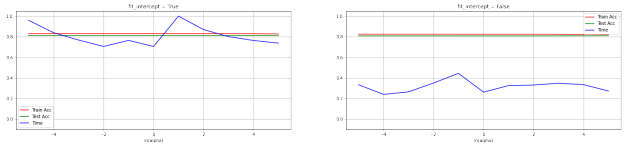
3.2. Ridge Classifier

Ridge Classifier is a classifier using *Ridge Regression*, a model that solves a regression problem where the loss function is the linear least squares function and regularization is given by the l2-norm. The loss function is given by

$$||y - Xw||_2^2 + \alpha||w||_2^2$$

where α is regularization parameter, w is the slope of the model, y is the given label and Xw is the predicted label.
For our multi-class classification, Ridge Classifier create a multi-output regression scenario, and then train 10 independent Ridge Regression models with a *one-versus-all* approach.

Unlike the AdaBoost Classifier, Ridge Classifier run way much faster so we could compute our tests on the entire dataset. The values we considered for the parameters were $[e^{-5}, e^{-4}, e^{-3}, e^{-2}, e^{-1}, 1, e, e^2, e^3, e^4, e^5]$ for α, and $[True, False]$ for the intercept.
We saw that the model with fit intercept is slightly better for the accuracy score, but is slower than the model without fit intercept. We also saw that changing the value for α didn't change much the accuracy score, that was to be expected because in the Ridge Regression the regularization constitutes a minor part of the loss function when the number of training element is large.



3.3. Support Vector Classifier

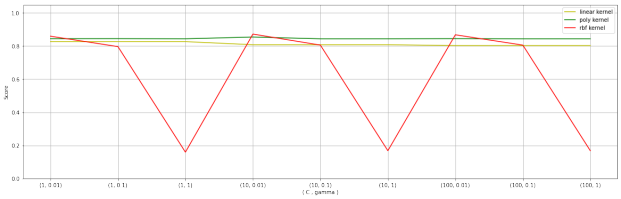
Support Vector Classifier tries, using a *one-vs-one* approach, to find a hyperplane that maximizes the separation of the data points to their potential classes in an 784-dimensional space. The data points with the minimum distance to the hyperplane are called *support vectors*. The SVC

has a kernel that converts the input data space into a higher-dimensional space. We considered three different kernel functions:

- Linear: $k(x_i, x_j) = x_i \cdot x_j$
- Polynomial: $k(x_i, x_j) = (\gamma + x_i \cdot x_j)^d$
- RBF: $k(x_i, x_j) = e^{-\gamma||x_i - x_j||_2^2}$

Every kernel function do takes two data points and calculates a distance score of those that helps to transform the data points to a higher-dimensional mapping, which reduces the computational effort and time and is especially useful for huge amounts of data.

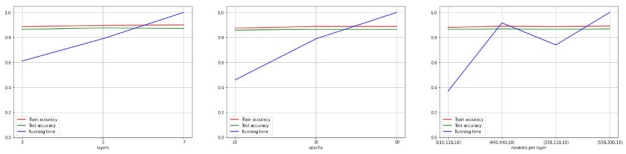
We used the already implemented *Grid Search* on all the tree different choices for the kernel on a 10000 training set for time issues, to find the best value for the regularization parameter from $[1, 10, 100]$, and for the kernel coefficient γ from $[0.01, 0.1, 1]$. For the polynomial kernel we only considered the case in which the degree $d = 2$.
We saw that the accuracy obtained with linear and polynomial kernels is quite stable for all the choices. The one obtained with rbf kernel changes a lot for different values of γ, but also gives the highest accuracy. Once we found the best values for the parameters, we implemented the classifier on the entire dataset.



3.4. Neural Network

A Neural Network is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. The concept of the artificial neural network was inspired by human biology and the way neurons of the human brain transmit a signal to other neurons and function together to understand inputs. Neural Networks are organized into *layers* of *nodes*. Each node multiply a *weight* to each of its incoming signals, and then adds the resulting products together. Finally, the result is passed through an *activation function*. Signals travel from the first layer to the last layer. Neural Networks, in order to learn, need to train on the same data possibly multiple times (*epochs*).

We started by implementing a Neural Network with 3, 5 and 7 layers, in order to determine what choice is the better one. We saw that the accuracies were similar but the Neural Network with 3 layers was more time efficient. Then we checked what choice of number of neurons per layer is the better one, working with a 3 layered Neural Network. Like we saw before, the accuracies were similar but the choice of 110 neurons for the first two layers is much faster. In the end we tried if more epochs would have improved the performance, implementing an *early stopping* to avoid overfitting. We saw that the early stopping decreased the accuracy score on the test set, indeed the model was trained on a 10% smaller set in order to leave a portion to validation. Due to early stopping we discovered that over 18 epochs could cause the classifier to overfit.



4. Conclusions

In the following table are displayed the performances of the best models we found.

	Train. Acc.	Test Acc.	R. Time
AdaBoost Class.	1.0	0.8618	1380s
Ridge Class.	0.8291	0.8122	0.44s
SVC	0.9707	0.9002	1659s
Neural Network	0.8987	0.8754	85s

Our results show that the the best accuracy score is obtained with a Support Vector Classifier model, although this is by far the slowest one. On the other hand, the Ridge Classifier is the fastest model but the less accurate. All things considered we think that a good compromise between time and accuracy is the better solution, so in this case we are inclined to choose the Neural Network model.
We also checked the distribution of the predictions over the 10 classes, all four models misclassifies the same one class more than the others, as it can be seen on the following figure.