Università degli Studi di Verona

DIPARTIMENTO DI INFORMATICA
Corso di Laurea in Ingegneria e scienze informatiche

Progetto di Teorie e Tecniche del Riconoscimento

Clothing Classification

Candidati:
Nicholas Merci
Matricola VR445512

Carlo Veronesi
Matricola VR450855

Anno Accademico 2019–2020

Contents

1 Motivation and Rationale

During the last few years, e-commerce has rapidly grown in terms of everyday usage.

Since the amount of people interested in online shopping is exponentially increasing, it is becoming much more important to have automatic systems that can guide the user during the choice of a product.

Among the various applications of machine learning, one of the most crucial is represented by product classification. Thus we decided to implement some of the techniques we learnt during the Pattern Recognition course in the context of clothing classification.

2 Objectives

We wanted to apply some of the methods learnt during the Pattern Recognition course to image classification, a task that is among the most important machine learning experiments.

In particular we implemented a KNN and SVM classificator for clothing products that could work in any kind of scene, in any type of image, from studio to amateur ones.

The final objective is to compare the two methods in order to find which one fits best for this particular task

3 Methodology

3.1 Dataset

The dataset we utilized is DeepFashion 2 [5], released in 2019. It contains 491K images with a total of 801K items of 13 clothing categories. Each item paired with annotations such as style, scale, view-point, occlusion, bounding box, landmark and masks.

The dataset is split into a training set (391K images), a validation set (34K images), and a test set (67K images). For computation and time reasons we decided to use the training set as our dataset.

As described in the Experiments section, we have applied some filtering to optimize the training phase.

3.2 Feature extraction

3.2.1 HOG features

Histogram of Oriented Gradient (HOG) feature descriptor consists in dividing the image into squared cells, computing horizontal and vertical gradients, then getting magnitude and direction and creating an histogram for each cell, normalizing it with a block-wise pattern.

This technique is mostly used for object detection and is not depending from colors.

The final results we are going to show have been obtained working with HOG features, which have given better accuracy than Neural features for this particular classification problem.

3.2.2 Bounding Box

Since the dataset is made up of images containing multiple items, we used the bounding box coordinates contained in the annotations to crop unwanted objects.

At this point, every training image contains a single dress, which occupies the whole image.

3.2.3 PCA

After the extraction of the Hog Features from our dataset, each sample was described by a feature vector of dimension [1 x 3780].

To reduce the dimensionality of our datas, and consequently also reducing computational time, we implemented the method of the Principal Component Analysis.

Once the procedure is completed, the feature vectors are reconstructed using only the K most important eigenvectors, where K is a parameter we can set to find a compromise between information loss and dimensionality.

For the implementation of PCA, we used the decomposition.pca method provided by scikit-learn. [1]

3.3 Model selection

In order to classify our datas, we used two different approaches: KNN and SVM. Each one has been tested with various combination of parameters, in order to find the best case in terms of accuracy, precision and recall.

3.3.1 KNN

The K-Nearest Neighbors is a non-parametric method which considers a range of the nearest K neighbors to classify the input data. The final classification is based on the most populated class in that neighborhood.

There are many different metrics to choose from, in order to calculate distances between the input and the other samples:

- Euclidean distance;
- Manhattan distance;
- Minkowski distance: equivalent to euclidean when p = 2, equivalent to manhattan when p = 1;
- Cosine distance.

For the implementation KNN, we used the KNeighborsClassifier method provided by the scikit-learn library. [4]

3.3.2 SVM

The Support-Vector Machine doesn't natively support the multi-class classification, so we adopted the One vs Rest strategy.

By doing this, we trained a different classificator for each of the classes, so that every classificator sees only a class as positive, while the others are considered as negative.

In order to implement this method, we built a model based on 13 SVC functions, provided by the scikit-learn library. [2]

SVM has been used with 3 different kernels, each of them with the best parameters values for our dataset.

The task of finding the best values was accomplished by a tuning phase. Through a cross-validation on the possible values for each kernel, we used the GridSearch method (from scikit-learn [3]) to find the best combination of them.

Once the best combination of values for the parameters were found, we started the training procedure based on those results.

4 Experiments and Results

4.0.1 Dataset filtering

Even if based on the training set instead of the entire dataset, the timing required by the training of the model was too high, so we decided to keep only one item from every image. Also, keeping the whole dataset didn't return better results than with a filtered dataset. The amount of the total set was decreased from 391K to 190K items.

In order to improve the dataset accuracy, we tried to filter the dataset by selecting items with minimum occlusion (1 in range 3), the aim was to train the model using images with a better quality. Given the limited amount of samples belonging to the categories "short sleeve outwear", "sling" and "sling dress", we decided to keep every occlusion value for them.

In order to keep a good balance for the entire dataset, we kept a maximum of 8K examples for each category.

Finally, we splitted the obtained dataset as 85% training and 15% test.

Table 1: Training set partitions

short sleeve top	long sleeve top	short sleeve outwear	long sleeve outwear	vest	sling
6765	6833	355	6251	6043	1365

shorts	trousers	skirt	short sleeve dress	long sleeve dress	vest dress	sling dress
4827	6890	2970	6771	668	6551	3458

Table 2: Test set partitions

short sleeve top	long sleeve top	short sleeve outwear	long sleeve outwear	vest	sling
1235	1167	79	1126	1039	253

shorts	trousers	skirt	short sleeve dress	long sleeve dress	vest dress	sling dress
853	1112	559	1229	668	1150	569

4.0.2 PCA

In order to reduce the number of features of our dataset, we executed PCA with 75% as the amount of components we wanted to maintain. In other attempts we used a slightly higher percentage (80%-85%), but the gain in the accuracy of the classifier was very marginal, while the dimensionality was increasing significantly. Once this compromise was found, the dimensionality of the feature vectors changed from 3780 to 393.

4.0.3 KNN

The K-Nearest Neighbors has been used with different K values (1, 3, 5, 7, 10, 15) for each of the following metrics:

- Euclidean distance
- Manhattan distance
- Cosine distance

In the instance of our classification problem, we can see that the cosine distance gives the best results overall.

Here is a summary of the results we obtained for accuracy, precision and recall.

Table 3: Accuracy

Metric \ K	1	3	5	7	10	15
Euclidean	30.78%	24.50%	21.04%	19.74%	17.69%	15.53%
Cosine	52.02%	47.85%	48.69%	48.91%	49.26%	49.66%
Manhattan	30.76%	24.08%	20.82%	19.23%	17.18%	14.91%

Table 4: Precision

Metric \ K	1	3	5	7	10	15
Euclidean	0.52	0.49	0.49	0.52	0.54	0.56
Cosine	0.51	0.51	0.51	0.52	0.53	0.54
Manhattan	0.50	0.47	0.48	0.47	0.51	0.53

Table 5: Recall

Metric \ K	1	3	5	7	10	15
Euclidean	0.31	0.24	0.20	0.20	0.18	0.16
Cosine	0.51	0.45	0.46	0.45	0.45	0.45
Manhattan	0.31	0.24	0.21	0.19	0.18	0.16

4.0.4 SVM

Prior to executing the SVM classifier, we applied a parameters tuning procedure. Here are the parameters we tuned for every kernel:

- Linear: C
- Polynomial: C - Degree
- RBF: C - Gamma

To perform the cross-validation, the GridSearch has been applied on a subset of the dataset, whose samples were normalized using the StandardScaler method (from sklearn [3]).

- Training Set: 3000 samples
- Test Set: 300 samples

The best combinations for each kernel resulted in the following:

- Linear: C = 10
- Polynomial: C = 10, Degree = 2
- RBF: C = 10, Gamma = 0.001

Here is a summary of the results we obtained for accuracy, precision and recall.

Table 6: Accuracy

Kernel \ Iterations	10	100	1000	5000
Linear	12.84%	15.43%	14.09%	15.99%
Polynomial	10.07%	11.05%	33.12%	50.33%
RBF	11.31%	17.36%	41.74%	56.71%

Table 7: Precision

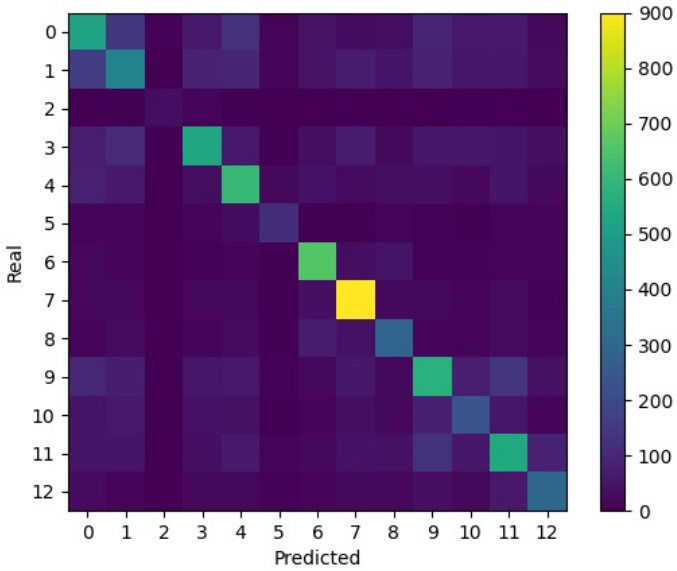
Kernel \ Iterations	10	100	1000	5000
Linear	0.12	0.26	0.14	0.15
Polynomial	0.10	0.17	0.37	0.54
RBF	0.41	0.49	0.51	0.62

Table 8: Recall

Kernel \ Iterations	10	100	1000	5000
Linear	0.15	0.19	0.18	0.20
Polynomial	1.00	0.19	0.30	0.47
RBF	0.12	0.14	0.38	0.54

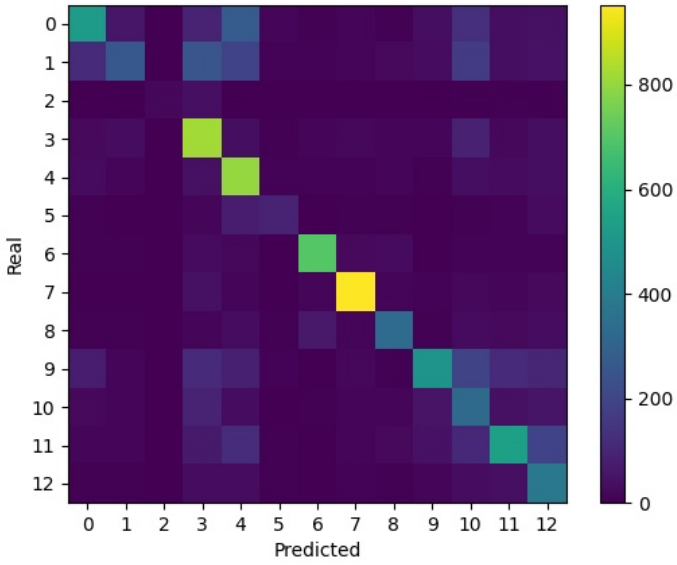
5 Confusion Matrices

Figure 1: Confusion Matrix of KNN classifier with K=1, using Cosine distance.



The other **Confusion Matrices with plots and actual data** and **Classification Reports** can be found in *Results/* folder

Figure 2: Confusion Matrix of SVM classifier with iter=5000, using RBF.



6 Qualitative Results

Figure 3: Some results obtained with KNN classifier with K=1, using cosine distance.

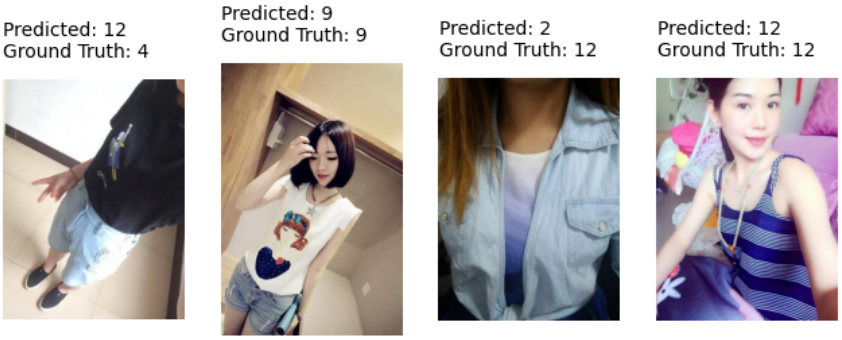
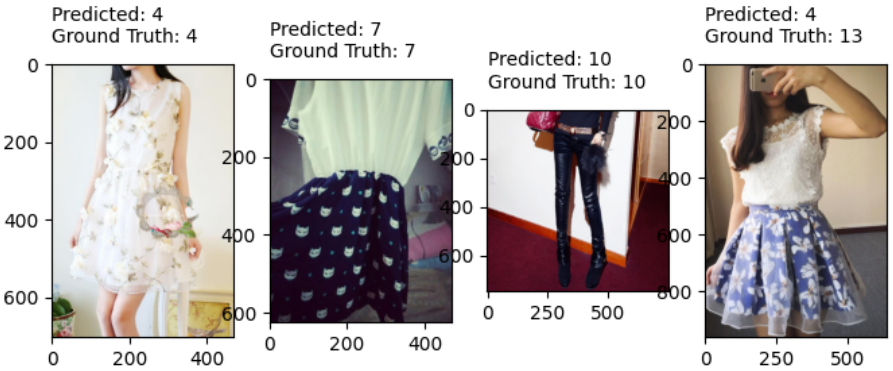


Figure 4: Some results obtained withSVM classifier with iter=5000, using RBF.



The other **Qualitative results** can be found in *Results/Qualitative*

7 Conclusions

The comparison between KNN and SVM results shows that the best results are obtained by the latter, with a very high number of iterations. In fact, KNN with cosine distance reaches approximately 52% of accuracy, while at 5000 iterations SVM with RBF kernel gets 56%. Of course, the training phase using this method is very heavy computationally (about 8 hours with our setup), while KNN takes less than a minute.

Moreover, we found large differences in performance changing distance function (for KNN) or kernel (for SVM), often about 20-30% of accuracy. This shows that, depending on the dataset in use, certain functions give way better results than others.

References

[1] sklearn.decomposition.pca.

[2] sklearn.decomposition.svc.

[3] sklearn.modelselection.gridsearchcv.

[4] sklearn.neighbors.kneighborsclassifier.

[5] Yuying Ge, Ruimao Zhang, Lingyun Wu, Xiaogang Wang, Xiaoou Tang, and Ping Luo. Deepfashion2: A versatile benchmark for detection, pose estimation, segmentation and re-identification of clothing images, 2019.