Clothes classification with SVM

March 14, 2020

1 Clothes Classification with Support Vector Machines

In this notebook we are going to explore the use of Support Vector Machines (SVM) for image classification. We are going to use a new version of the famous MNIST dataset (the original is a dataset of handwritten digits). The version we are going to use is called Fashion MNIST (https://pravarmahajan.github.io/fashion/) and is a dataset of small images of clothes and accessories.

The dataset labels are the following:

Label	Description
0	T-shirt/top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle boot

1.1 TODO: Insert your surname, name and ID number

Student name: Alessandro Valente
ID: 1234429

```
In [1]: #load the required packages

        %matplotlib inline

        import numpy as np
        import scipy as sp
        import pandas as pd
        import matplotlib.pyplot as plt

        import sklearn
        from sklearn.datasets import fetch_mldata
        from sklearn.neural_network import MLPClassifier
        import sklearn.metrics as skm
```

```
In [2]: # helper function to load Fashion MNIST dataset
        def load_mnist(path, kind='train'):
            import os
            import gzip
            import numpy as np
            labels_path = os.path.join(path, '%s-labels-idx1-ubyte.gz' % kind)
            images_path = os.path.join(path, '%s-images-idx3-ubyte.gz' % kind)
            with gzip.open(labels_path, 'rb') as lbpath:
                labels = np.frombuffer(lbpath.read(), dtype=np.uint8,offset=8)
            with gzip.open(images_path, 'rb') as imgpath:
                images = np.frombuffer(imgpath.read(), dtype=np.uint8,offset=16).reshape(len(la
            return images, labels
```

```
In [3]: #fix your ID ("numero di matricola") and the seed for random generator (as usual you c
        ID = 1234429
        np.random.seed(ID)
```

```
In [4]: #load the Fashion MNIST dataset from the 'data' folder and let's normalize the feature

        X, y = load_mnist('data', kind='train')
        # rescale the data
        X, y = X / 255., y # original pixel values are between 0 and 255
        print(X.shape, y.shape)
```

```
(60000, 784) (60000,)
```

Now split into training and test. Make sure that each label is present at least 10 times in training. If it is not, then keep adding permutations to the initial data until this happens.

```
In [5]: #random permute the data and split into training and test taking the first 500
        #data samples as training and the rests as test
        permutation = np.random.permutation(X.shape[0])

        X = X[permutation]
        y = y[permutation]

        m_training = 500

        X_train, X_test = X[:m_training], X[m_training:]
        y_train, y_test = y[:m_training], y[m_training:]

        labels, freqs = np.unique(y_train, return_counts=True)
        print("Labels in training dataset: ", labels)
        print("Frequencies in training dataset: ", freqs)
```