Cipia Vision – Home Assignment – MNIST multi-classifier using SVM and PCA&HOG features

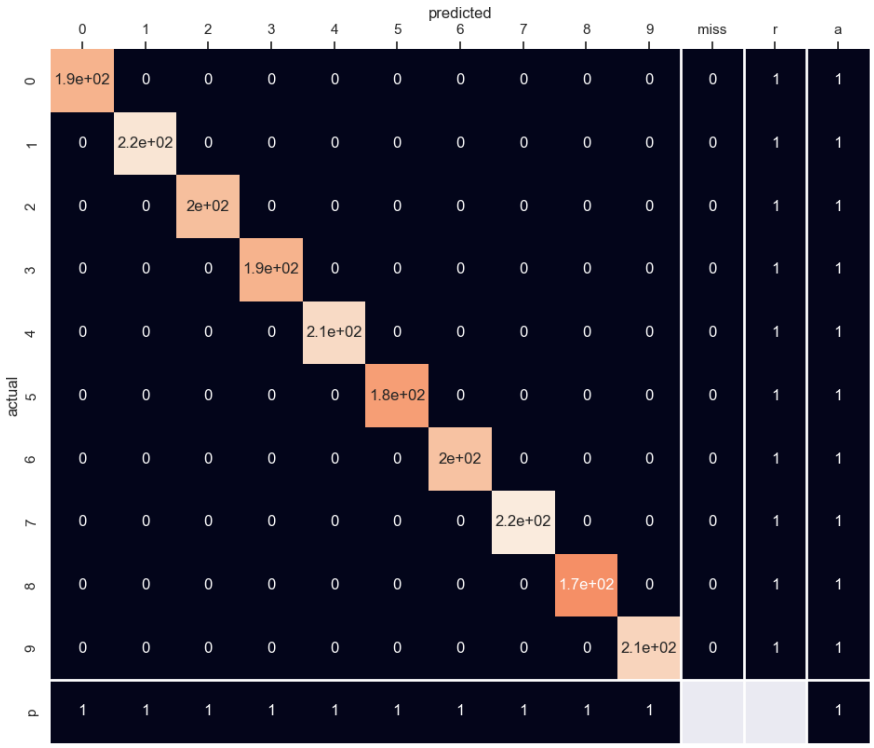
Name: Sagi Levi

GitHub link: https://github.com/sagilevi1/mnist_svm_from_scratch

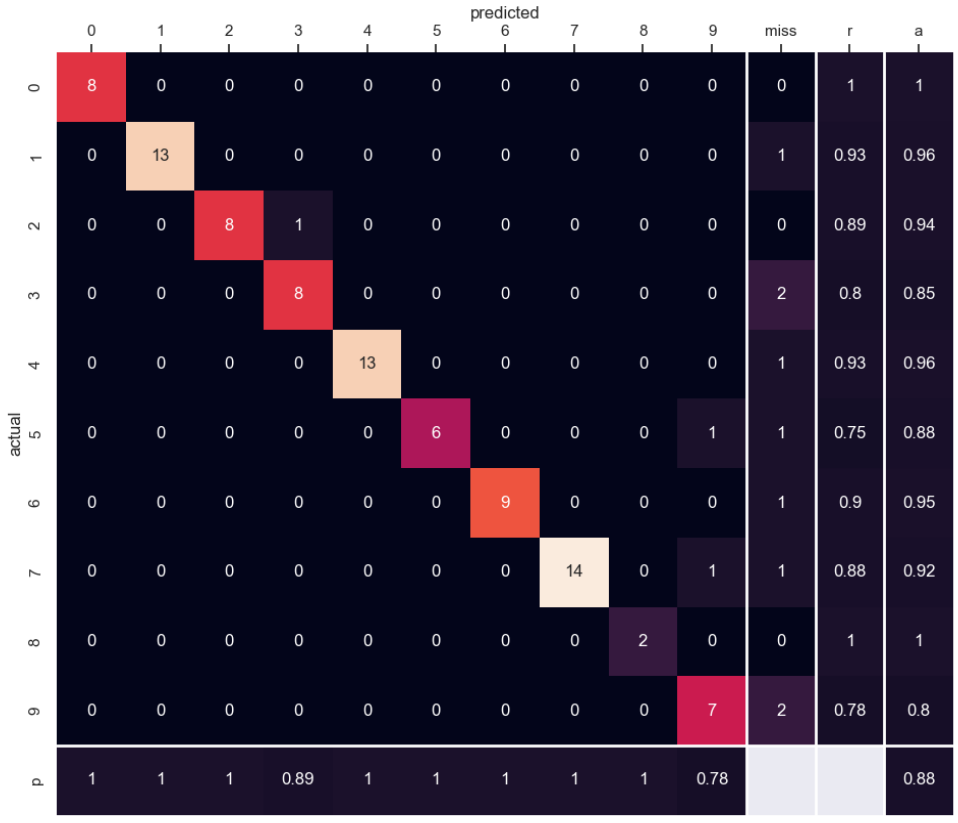
1. **Description**: Classify handwritten digits from the MNIST dataset by using linear SVM from scratch (one to all) and PCA, HOG features.
2. **Instructions:**
    a. pip install -r requirements.txt
    b. run the main python file.
3. **Solution architecture:**
    a. MNIST - Download the MNIST data set.
    b. HOG - Create HOG features for every image of the training and the testing set.
    c. PCA - Build a PCA transform from the HOG features of the training set. Make a dimensionality reduction to both training and testing HOG features by the PCA.
    d. SVM - Build a linear binary SVM classifier for every digit with the training set features.
    e. Predict for very digit the test and the train features and save results for the whole multi classifier.
    f. Build a confusion matrix.
4. **Hyperparameters:**
    a. Number of training (1 to 60k) and testing (1 to 10k) examples.
    b. HOG – quantize orientation, cell size, and block size
    c. PCA – maximum variance for the dimensionality reduction
    d. SVM – iterations, learning rate (power of every step gradient descent), lambda parameter (the power of the maximum margin constraints).
5. **Issues:**
    a. **Multiclass problems:**
        i. The algorithm loops on the digits and collects the results of every binary SVM classification (“first come first served”). For this reason, if 2 binary SVM predict true for the same example then the early digits will get it.
        ii. There are some examples that are rejected (predict negative) in all the binary classifications of all the digits. Eventually, the algorithm misses to class them.
    b. **Statistics:**
        i. True negative per digit: by definition, this parameter is the sum of all the correct predictions of the other digits. This makes the proportionality between TN and TP unbalanced and this influences the accuracy per digit. For that reason, I scale this parameter (TN) by the amount of the digits.
6. **Performance:**
    a. For 2k training examples the total accuracy of 100 testing examples was 0.88.
    b. For 20k training examples the total accuracy of 2000 testing examples was 0.97.
7. **Improvement Suggestions**:
    a. RBF kernel for the SVM.
    b. K-means instead the SVM.
    c. Neural network.

8. Appendices: Example of the confusion matrix:

a. 2000 train and 100 test examples:



Confusion Matrix Of The Training Set - 2000 Examples



Confusion Matrix Of The Testing Set - 100 Examples