PSM Fall 2016: Programming exercise: training a feature-based linear classifier for the digits dataset

Data. The digits dataset described in Part 4 of the PSM lecture notes, Section 4. Raw data are contained in the file `mfeat-pix.txt` in this exercise package. The Matlab script `basicDemo.m` contains routines for reading these data into Matlab, visualizing them, splitting them into training- and testing-data, and for generating correct labels. This script also includes a demo run for training a linear classifier on the basis of the ten class-prototype features f^k_i from the lecture notes.

Task.

1. Train a three-stage classification algorithm:
    a. In the first stage extracts features from input image vectors $\mathbf{x}_i$, resulting in a lower-dimensional feature vector $\mathbf{f}_i$.
    b. In the second stage, this feature vector $\mathbf{f}_i$ is linearly transformed to a 10-dimensional classification hypothesis vector $\mathbf{c}_i = W\mathbf{f}_i$.
    c. In the third stage, $\mathbf{c}_i$ is used to obtain a class decision by outputting that class label (from "class 1" to "class 10") which corresponds to the index of the maximal value in $\mathbf{c}_i$. (Note that "class 1" contains the "0"-images etc).
2. Use only half of the images from the provided dataset for training. The dataset contains 200 images per class. Use the first 100 per class for training, and the remaining 100 images for testing. The `basicDemo.m` script does this split for you.
3. Use linear regression to train W. This is pre-implemented in the script `basicDemo.m`.
4. It is up to you what features you extract from the raw image vectors. For instance, you can
    a. invent your own "hand-made" features,
    b. estimate PCA features on the basis of all training data (these PCA features represent geometric properties that are characteristic for all image classes simultaneously)
    c. estimate PCA features separately for each class.

    You can also mix different kinds of features.
5. Use cross-validation to optimize your feature repertoire (too many features will lead to overfitting). Use only the training images for this optimization!!
6. Test your final optimized three-stage classification algorithm on the test data and calculate the test misclassification rate.
7.
8. Write (with Latex or some text processor) a short documentation of the ideas behind your features, describe your cross-validation scheme, package this documentation together with your Matlab/Python code and submit this by email to x.he@jacobs-university.de and h.jaeger@jacobs-university.de

Comparison. For your orientation: the best published classifiers for this benchmark reach about 2% test misclassifications. The pre-implemented demo based on the prototype images has about 9%. Aim at a performance somewhere in this bracket.

Possible extensions. If you feel like it, you may improve the pre-implemented basic linear regression formula by using *ridge regression* (check out on Wikipedia).

Bonus points. Owen and I will award particularly insightful or carefully done (that includes a cleanly typeset documentation) submissions with a maximum of 5 bonus points – based on our subjective decision. Bonus points will be added undiluted to the final course grade percentage points.

Deadline. Email submissions are due by Friday, Dec 2, midnight. Late submissions will not be considered except in cases of an official medical excuse.