Machine Learning, Spring 2019: Miniproject 2

This is a programming project – where the programming part is the easy one; doing it in a machine-learning professional way is the tricky part. It will be graded and the grade counts toward the course grade. ***Join into groups of two*** *and submit a single solution per group, indicating the group members' names on the report sheet. You can use Python or Matlab.*

Please send your ***type-set*** *solutions by email to our two TA's Tianlin Liu (t.liu@jacobs-university.de) and Steven Abreu (s.abreu@jacobs-university.de)*

Deadline for submission is April 21, 23:59 hrs (email sending timestamp). Submissions arriving later (even a second after midnight) will be reviewed but not counted for the course grade.

Summary. The objective of this task is to train a classifier for the Digits dataset that we use in this course, implementing a full processing pipeline from feature extraction to (linear) classifier training, attempting to squeeze performance out of the classifier by careful cross-validation and regularization. – If you get a misclassification rate of below 5%, you're doing fine; the best results in the literature reach about 1.8% if my memory doesn't fail me.

Detailed task description. Use the digits data that you still can find at

http://minds.jacobs-university.de/sites/default/files/uploads/teaching/share/DigitsBasicRoutines.zip.

The classification task is to classify the 10 digit classes, using the original benchmark set-up. That is, from each of the 200 examples given in the dataset for each digit, take the first 100 as *training data* and the second 100 as *testing data*. Throughout the training and cross-validation process, never touch the testing data – pretend they don't exist. Only *after* you have a trained a classifier solely using the training data, you may test it *once* on the testing data.

Here are the steps you should carry out in order to train one classifier:

1. Decide on a set of k features that you use to trim the input pattern dimension from 240 to k. The choice of features is up to you (hand-made, K-means based, or PCA features). Apply these features, obtaining k-dimensional feature vectors $\mathbf{f}(x_i)$, one such feature vector for each of your 10×100 training patterns x_i. You will optimize the choice of features, and how many of them you use (that is, the choice of k), by cross-validation.
2. Let us number the classes as class 1 = the "1" patterns, …, class 9 = the "9" patterns, class 10 = the "0" patterns. For each training pattern x_i create a binary, 10-dimensional *target vector* z_i, which is zero everywhere except at the position corresponding to the class number of the pattern x_i, where it is set to 1.
3. Compute the linear regression weights W_{opt} to approximate z_i from $\mathbf{f}(x_i)$, (padded with a trailing 1 as bias, so $\mathbf{f}(x_i)$ actually has size $k + 1$). W_{opt} should be a matrix of size $10 \times (k + 1)$. Use ridge regression, that is, the formula (39) from the LNs. You

may choose to make do without ridge regression, using plain linear regression; then use formula (39) with α set to zero.

4. Set up a cross-validation scheme (number of folds – your choice), such that for every model choice (which features, number of features, setting of α) you can compute training and validation MSE's, and training and validation misclassification rates. The outcome of a training trial (one run of the cross-validation scheme based on one choice of model) is thus comprised of the four numbers MSE_{train}, $MISS_{train}$, $MSE_{validate}$, $MISS_{validate}$.
5. Optimize the $MSE_{validate}$ and $MISS_{validate}$ by experimenting with different model choices.

Deliverables:

1. A typeset report, target size 2 pages excluding figure(s), in which you document what you did. Discuss what you found. Give a graphic of the train/validation curves similar to Figure 20 in the lecture notes. And, of course, report the MSE and misclassification rate of you final cross-validation-optimized model on the test data.
2. Your code. It should be documented inline well enough to allow the TAs to quickly orient themselves within the code. The code should contain a *self-contained script* executing one complete feature-reduction and cross-validation trial that the TAs can run with a single click.

The grade will be based solely on the report, except when the TAs need to inspect the code in order to understand your report or to give helpful hints and then find your code under-documented, which will lead to a grade reduction.

Particularly well-done reports, which include also some extensions beyond the basic task outlined above (for example, a systematic exploration of different kinds of features), may be awarded with a max of 3 **bonus points**. Bonus points enter the final course grade calculation undiluted, that is, 3 bonus points will lift your final course grade by 3 percentage points.