# Machine Learning (lecture) Fall 2014: Exercise sheet 4

**Microproject**: classifying 10 digits by generalized linear regression

*Return your documented code and results by email by Sunday Oct 19, midnight*

**Task description, executive summary:** design, optimize and test a generalized linear discriminant classifier for the full digits dataset (digits 0, 1, ..., 9).

**Detail:**

- Training data: the first 100 of each of the 200 examples per digit in the by now familiar `mfeat` dataset. Use only these for optimizing your classification model (in a cross-validation scheme).
- Invent $L$ features $\phi_1(\mathbf{x}), ..., \phi_L(\mathbf{x})$ (where $\mathbf{x}$ is the raw picture vector) which you think may carry classification-relevant information. The choice of $L$ is in your hands.
- Implement a linear classifyer (of the type given in Equation (3.7) or (3.21) in the Section 3 handout) which takes $(1, \phi_1(\mathbf{x}), ..., \phi_L(\mathbf{x}))$ and returns a 10-dimensional output vector $\mathbf{y}(\mathbf{x})$.
- Compute regularized linear regresssion weights $\mathbf{W}$ of this linear classifier as in Equation (3.17).
- Find the best setting for the Tychonov regularizer coefficient by some version of cross-validation.
- Optionally (if you are ambitious) you may furthermore optimize your features (kind and number) with cross-validation.
- If you are done optimizing your classifier on the training data, test it on the remaining half of the data, that is, on the 10 x 100 = 1000 images that you did not touch so far. Report the nr of misclassifications.

It may be interesting for you that with some effort, 15 test misclassifications have been achieved (documented in an article by Duin and Tax, URL on the course homepage).

**Deliverables:**

1. A runnable Matlab or Python script `learn_test_digits` which trains the classifier weights $\mathbf{W}$ and tests the resulting classifier on the test data, returning the nr of misclassifications. This script need not do the cross-validation part. Instead, the script `learn_test_digits` will use the Tychonov regularizer and the features that you have optimized separately.
2. Any function definition files that are called from `learn_test_digits`.
3. You do *not* have to deliver the code that you used for the cross-validations or for experimenting with feature designs. However, please document this part of the project, explaining in plain English how you designed your features, and documenting (including a plot) the crossvalidation / regularization. Target size for the written documentation: 1 page.

Send your zipped files to h.jaeger@jacobs-university.de and the course TA Dzmitri Bahdanau d.bahdanau@jacobs-university.de by Sunday, October 19, midnight. Please name your zipfile as follows:

`<your last name>HW4.`

**Bonus** This is a relatively heavyweight exercise, and if you invest a lot of effort it should be accordingly rewarded. So, the five HWs with the best test recognition rates will receive 5, 4, 3, 2, 1 bonus points, respectively. Only solutions with a transparent written documentation are eligible however. A bonus point is added to the total course score at the end of the semester – it adds to the course score in "undiluted" form and thus is quite valuable. 3 bonus points roughly amount to an improvement of the final course grade by 1/3.