TECHNISCHE UNIVERSITÄT MÜNCHEN
Fakultät für Elektrotechnik und Informationstechnik
Lehrstuhl für Datenverarbeitung
PD Dr. Martin Kleinsteuber

Information Retrieval in High Dimensional Data

Lab #4, 09.11.2017

Logistic Regression

Note: Some of the routines in the following tasks are very memory-intensive. Try to minimize your memory consumption, e.g. by handing over unused variables to the garbage collector. If this does not suffice, consider working with a subset of the training data.

Task 1. Recall the data processing routines from the last lab course. The following excercises build on top of the extracted feature representations, but instead of the prebuilt classifier, we want to implement logistic regression by hand, i.e. by minimizing $L(\mathbf{w})$ from Section 3.1. in the lecture notes. To this end, make sure, that the variables `train`, `test`, `train_data_features` and `test_data_features` are loaded to your IPython shell.

a) Write a PYTHON function `logistic_gradient` that expects a training set matrix `X_train`, a ground truth label vector `y_train` and a current weight vector `w` as its input and returns the gradient `g` of the negative log-likelihood function of the logistic regression. Refer to the lecture notes for the mathematical definition.

b) Write a PYTHON function `logistic_hessian` that expects a training set matrix `X_train` and a current weight vector `w` as its input and returns the Hessian `H` of the negative log-likelihood function of the logistic regression. Refer to the lecture notes for the mathematical definition.

c) Write a PYTHON function `find_w` that expects a training set matrix `X_train`, a ground truth label vector `y_train`, and a maximum iteration number `max_it` that determines the optimal logistic regression weight vector `w_star` by performing Newton's method via calling `logistic_gradient` and `logistic_hessian` in each iteration. Make sure to include the affine offset w_0 in your model.

d) Write a function `classify_log` that expects a weight vector `w` and a test set matrix `X_test` and classifies the samples in `X_test` via logistic regression, returning a label vector `y_test`. Test your implementation on `train_data_features` and `test_data_features` with one iteration and with 10 iterations. What do you observe?

e) Logistic regression is prone to *overfitting*. To prevent this, regularizing parameters can be used. Adjust your implementation in such a way that instead of minimizing $L(\mathbf{w})$, it minimizes the term

$$L(\mathbf{w}) + \alpha\|\mathbf{w}\|^2, \tag{1}$$

where α is a non-negative regularization parameter. Test your implementation with $\alpha = 1$ with one iteration and with 10 iterations.

Helpful Python/Numpy functions

`np.diag(x)`	creates diagonal matrix with x on diagonal
`np.linalg.lstsq(A,b)`	returns the minimizer of $\|\mathbf{Ax} - \mathbf{b}\|$
`np.exp(X)`	exponential function