Exercises
Machine Learning
HS 2016

Machine Learning Laboratory
Dept. of Computer Science, ETH Zürich
Prof. Joachim M. Buhmann
Web https://ml2.inf.ethz.ch/courses/ml/

Series 5, Oct 24th, 2016
(Numerical Estimation and Classification)

Email questions 1, 2, 3 to: **Andreas Hess**
ahess@student.ethz.ch
Email questions 4, 5, 6 to: **Milica Petrovic**
milicap@student.ethz.ch

Please hand in solutions by Thursday Nov 17th.

Problem 1 (Efficient Leave-One-Out Cross Validation):

In this question, we propose a computationally efficient method to compute the error of leave-one-out cross-validation (LOOCV). Given the output vector $\boldsymbol{y} \in \mathbb{R}^n$ and the input matrix $\mathcal{X} \in \mathbb{R}^{d \times n}$, whose n columns correspond to the n data points $\boldsymbol{x}_i \in \mathbb{R}^d$, the objective of ridge regression is defined as:

$$f_\mu(\boldsymbol{w}) = \frac{1}{n}\|\mathcal{X}^T\boldsymbol{w} - \boldsymbol{y}\|^2 + \frac{\mu}{2}\|\boldsymbol{w}\|^2 \tag{1}$$

Let $\mathcal{X}_{(-i)} \in \mathbb{R}^{d \times (n-1)}$ denote the data matrix that is obtained by excluding column i form the full data matrix $\mathcal{X}$. To compute the LOOCV error, one has to compute the minimizer $\boldsymbol{w}^*_{(-i)}$, which is defined as

$$\boldsymbol{w}^*_{(-i)} = \arg\min_{\boldsymbol{w}} \left(\frac{1}{n-1}\|\mathcal{X}^T_{(-i)}\boldsymbol{w} - \boldsymbol{y}_{(-i)}\|^2 + \frac{\mu}{2}\|\boldsymbol{w}\|^2 \right) \tag{2}$$

for each $i \in \{1, .., n\}$ (remember: we have n folds and n hold-out datasets in LOOCV).
Then the cross-validation error is

$$\epsilon = \frac{1}{n}\sum_i^n \left(\langle \boldsymbol{x}_i, \boldsymbol{w}^*_{(-i)} \rangle - y_i \right)^2 \tag{3}$$

We want to prove that this error can be efficiently computed as

$$\epsilon = \frac{1}{n}\sum_i^n \left(\frac{y_i - \hat{y}_i}{1 - s_i} \right)^2 \tag{4}$$

where

$$\mathcal{A} = \left(\mathcal{X}\mathcal{X}^T + \frac{(n-1)\mu}{2}\mathcal{I} \right) \tag{5}$$

$$\hat{y}_i = \boldsymbol{x}_i^T \mathcal{A}^{-1} \mathcal{X}\boldsymbol{y} \tag{6}$$

$$s_i = \boldsymbol{x}_i^T \mathcal{A}^{-1} \boldsymbol{x}_i \tag{7}$$

To prove the above result, we follow three steps:

a. Prove that

$$\boldsymbol{w}^*_{(-i)} = \mathcal{A}^{-1}\mathcal{X}\boldsymbol{y} - \mathcal{A}^{-1}\boldsymbol{x}_i \left(\frac{1}{1 - \boldsymbol{x}_i^T\mathcal{A}^{-1}\boldsymbol{x}_i} \right) y_i + \frac{\mathcal{A}^{-1}\boldsymbol{x}_i\boldsymbol{x}_i^T\mathcal{A}^{-1}(\mathcal{X}\boldsymbol{y})}{1 - \boldsymbol{x}_i^T\mathcal{A}^{-1}\boldsymbol{x}_i} \tag{8}$$

Hint: Use Sherman-Morrison formula, i.e.

$$(\mathcal{M} - \boldsymbol{u}\boldsymbol{v}^T)^{-1} = \mathcal{M}^{-1} + \frac{\mathcal{M}^{-1}\boldsymbol{u}\boldsymbol{v}^T\mathcal{M}^{-1}}{1 - \boldsymbol{v}^T\mathcal{M}^{-1}\boldsymbol{u}} \tag{9}$$

where $\mathcal{M}$ is an invertible square matrix and $\boldsymbol{u}$, $\boldsymbol{v}$ are column vectors.

b. Derive that

$$\boldsymbol{x}_i^T \boldsymbol{w}_{(-i)}^* = \left(\frac{1}{1-s_i}\right)(\hat{y}_i - s_i y_i) \tag{10}$$

c. Show that

$$y_i - \boldsymbol{x}_i^T \boldsymbol{w}_{(-i)}^* = \left(\frac{1}{1-s_i}\right)(y_i - \hat{y}_i) \tag{11}$$

Problem 2 (Bootstrap):

In this question, we analyse the behaviour of a bootstrap variant, where the size of the bootstrap datasets is not assumed to be equal to the size of the original dataset. Assume we have a dataset $\mathcal{D}$, consisting of n data points (x_i, y_i).

a. Let $\mathcal{D}_c^{*1}$ be a bootstrap dataset we constructed from $\mathcal{D}$ by drawing $c \cdot n$ datapoint at random (with replacement), $c \in (0, 1]$. What is the expected fraction $u_{n,c}$ of data points that are present in $\mathcal{D}$ but not in $\mathcal{D}_c^{*1}$, i.e.,

$$u_{n,c} := \mathbf{E}\left[\frac{|\mathcal{D} - \mathcal{D}_c^{*1}|}{n}\right] \tag{12}$$

Derive an expression for $u_{n,c}$ that only depends on n and c.

b. The size n of a dataset is generally large and we therefore assume that $u_c := \lim_{n\to\infty} u_{n,c}$ is a good approximation of $u_{n,c}$. Derive a simplified expression for u_c and compute its value for $c = 1$.
Hint:

$$\lim_{n\to\infty}\left(\frac{1}{1-\frac{1}{n}}\right)^n = \lim_{n\to\infty}\frac{1}{\left(\frac{1}{1+\frac{1}{n}}\right)^n} \tag{13}$$

c. We now consider a binary classification task, i.e., let the x_i denote some feature vector and the y_i a binary class label (e.g. 1 or 0). Assume we train a binary classifier $\hat{f}_c^{*1}(x)$ on the bootstrap dataset $\mathcal{D}_c^{*1}$, and that the training error is equal to 0. How do we have to choose c, such that the test error we compute on the dataset $\mathcal{D}$ is exactly half as large as the actual generalization error?

Problem 3 (Practical Assignment (Numerical Estimation Methods)):

First of all, you can download the dataset and the code skeleton from the course webpage. Consider the ridge regression problem in problem 1. The question is how can we select the best regularizer μ, which minimizes the expected prediction? To address this issue, you have to follow three steps, where in each step you need to complete a brief implementation and also interpret the results.

a. In this experiment, we split the dataset into a test and training set. Then we do leave-one-out cross-validation (LOOCV) on a subset of the training set. Finally, we compare the error on the test set to the cross-validation error for different choices of the regularizer. Note that running LOOCV on the whole training set is expensive, hence we only use a random subset of the training set. To run the code, you have to fill in two blanks in the implementation of LOOCV. After running the code, explain whether the cross-validation provides a good approximation of the test error. Does it under- or overestimate the test error? Is this what you would have expected? If not, try to find a reason for the unexpected behaviour. **Hint**: change the random seed of `numpy`. Can the LOOCV error be used in order to choose the regularizer?

b. In this part, we accelerate LOOCV to run it on a larger training set. Use the result of Problem 1 to complete the implementation of the improved LOOCV in the python skeleton. Then report the execution time and the prediction error of the improved LOOCV, which is printed by the code.

c. Here we investigate the dependency of the best regularizer on the size of the training set. To this end, we run a model selection function to choose a regularizer which minimizes the LOOCV on the training set. The output of the code is a graph which shows the best regularizer for the different sample sizes. Firstly, fill in the missing lines of the model selection function. Secondly, explain how the best regularizer depends on the size of the training set.

Problem 4 (Decision Boundaries for Classification):

In this problem, you are working with social scientists to understand and predict voter preferences. Your collaborators have adminstered questionnaires to a sample of 1000 people, and used the answers to derive a two-dimensional representation of each repsondant, their 'questionnaire score'. A further question asked which of two political candidates the voters intended to vote for.

You can download this data from the course website - `decision_1.csv` under 'Homework' on piazza. Each row is a respondant, and the final column is their voting preference (1 or 0 corresponding to the candidates).

1. In your language of choice, visualise the data. Plot the respondents by their questionnaire scores and colour them by their voting preference. Do the groups look linearly separable?

2. You decide to use **logistic regression** to separate people into their voting preferences (binary classification).

   (a) Show how the posterior probability of class membership given data can be rewritten as a logistic function. Don't assume a specific probability distribution.

   (b) What is the argument of the function? What does it mean when its sign changes?

3. Now you assume the questionnaire scores are normally distributed given the underlying voting preference:

$$\begin{aligned} p(\mathbf{x}|v_0) &= \mathcal{N}(\mu_0, \Sigma_0) \\ p(\mathbf{x}|v_1) &= \mathcal{N}(\mu_1, \Sigma_1) \end{aligned} \tag{14}$$

   with equal prior probabilities, $p(v_0) = p(v_1) = 0.5$.

   (a) Assuming the covariance of each group is the same; $\Sigma_0 = \Sigma_1 = \Sigma$ (*does this seem like a reasonable assumption?*), what is the form of the decision boundary?

   (b) Relaxing this asssumption, show that when $\Sigma_0 \neq \Sigma_1$, the decision boundary is *quadratic* in $\mathbf{x}$.

   (c) Given the data (and its labels), what are the maximum-likelihood estimates of μ_0, μ_1 and Σ? *Hint: these have closed-form solutions for the multivariate normal distribution, you don't need to numerically optimise anything.* Estimate Σ for each class independently. Do you still feel justified in your assumption above?

   (d) Plot the decision boundary (and optionally colour misclassified voters) for the values of μ_0, μ_1 and Σ obtained above. For convenience, to get a linear decision boundary you can use $\Sigma = 1/2(\Sigma_0 + \Sigma_1)$. *Optional: find the quadratic decision boundary and visualise that.*

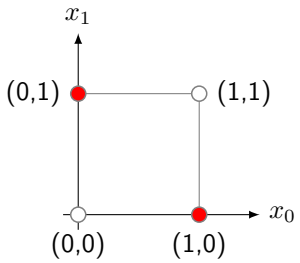
   (e) How many voters were misclassified?

4. One of your collaborators finds some misplaced questionnaires and sends you new data, see `decision_2.csv` under 'Homework' on piazza. Add this to your previous visualisation. What new modelling assumptions will you need to make? How would you go about classifying these new voters?

5. After all this analysis, you wonder if the normal assumption was incorrect. What other distribution could you use to model $p(\mathbf{x}|v)$, and why might it be appropriate?

Problem 5 (Perceptrons):

Recall the connectivity problem from lecture 8. As mentioned, this is equivalent to the XOR problem. Here is a diagram demonstrating the problem; points in red should be classified separately to those in white:



As evidenced by this illustration, there does not exist a linear decision boundary which separates the classes, so the perceptron algorithm will fail to solve the problem.

Construct a different set of features which admit a linear decision boundary for this problem, and provide the weights. That is, construct a set of functions $\phi(x_0, x_1)_i$ and $\mathbf{w}$ such that the sign of $\phi(x_0, x_1)^T\mathbf{w}$ separates the classes (feel free to assume homogeneous coordinates).

Hint: consider the truth table for the problem:

x_0	x_1	label
0	0	0
0	1	1
1	0	1
1	1	0

Problem 6 (Relationship between Fisher's discriminant and Least Squares):

The solution for Fisher's discriminant function for binary classification ($k = 1, 2$) was $\mathbf{w} \propto \mathbf{S}_W^{-1}(\bar{\mathbf{x}}_1 - \bar{\mathbf{x}}_1)$ with

$$\bar{\mathbf{x}}_k = \frac{1}{n_k}\sum_{n\in C_k}\mathbf{x}_n, \mathbf{S}_W = \sum_{k=1}^{2}\sum_{n\in C_k}(\mathbf{x}_n - \bar{\mathbf{x}}_k)(\mathbf{x}_n - \bar{\mathbf{x}}_k)^T$$

where n_k is the number of points in class C_k. Show that if you adopt the target coding scheme $t_1 = \frac{n}{n_1}$ for patterns in class 1 and $t_2 = -\frac{n}{n_2}$ for the others, with $n = n_1 + n_2$, then the solution of the least squares objective

$$E_{LS}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{n}(\mathbf{w}^T\mathbf{x}_i + w_0 - t_i)^2$$

is just Fisher's solution, that is, $\mathbf{w}_F = min_w E_{LS}(\mathbf{w})$