TKO3103 Exercise 2, linear and k-NN regression and n-fold cross validation

Technicalities

Contact info: Mon 10-11:00, Agora 452B/ptneva@utu.fi/by Moodle messages: Paavo Nevalainen.
Deadline: Mon Dec 14th 2015 at 23:55
Pair work or alone: you can form a different group (solo/pair) after Exercise 1 if you like.
Other details: Just as with Exercise 1.

Tasks

2.1) Use linear regression to predict the wine quality. Course material should help you to get acquainted with the tools. Report the regression weight vector $\boldsymbol{\beta}$ and two absolute error histograms (in-sample and out-of-sample). See details at the theory part.

2.2) Use k-NN regression (the average of *k* nearest samples) to predict the wine quality with *k*=1…8. Report the absolute error histograms (in-sample and out-of-sample) of *k*=1 and *k* with the smallest average error. If some of the concepts sound unfamiliar, attend lectures and study the slides.

scope	$k = 1$	$k = k_{best}$
in-sample	histogram plot	histogram plot
out-sample	histogram plot	histogram plot

You can save space by plotting some of the histograms on the same image using an outline representation. But separate bar histograms are also possible. Submit your report via Moodle **in pdf format**. Include your name (and the name of your partner).

The bare-bones theory

2.1) A short theory burst for a case with data matrix $\mathbf{D} \in \mathbb{R}^{n\times m}$ and the wine quality array $\mathbf{y} \in \mathbb{R}^n$, where *n* is the samples amount and *m* the attribute amount. First some definitions:

$\mathbf{x}_1^T = (1\ d_{11}\ d_{12}\ d_{13} \dots d_{1m})$ data line 1 values $d_{11}\ d_{12}\ d_{13} \dots d_{1m}$ with a sample value y_1
$g(\boldsymbol{\beta}, I) = \sum_{i\in I}\left(y_i - \mathbf{x}_i^T\boldsymbol{\beta}\right)^2$ objective function over $I \subset I_0 \subset \mathbb{N}_+,\ I_0 = [1, n]$.
$\boldsymbol{\beta}_I^* = argmin_{\boldsymbol{\beta}}\, g(\boldsymbol{\beta}, I)$ best weight vector for samples I

The in-sample and out-of-sample absolute prediction errors are defined below. Here is an algorithm:

$I_0 = I_1 \cup I_2 \cup \dots \cup I_N$ split data to e.g. *N=5* slots (division approximately even)
for *j= 1..N*:
  $J_j = I_0 \backslash I_j$ exclude each $\frac{1}{N}$th part of the data in turn
  $\mathbf{e}_{I_j} = \left\{y_i - \mathbf{x}_i^T\boldsymbol{\beta}_{J_j}^*\right\}_{i\in I_j}$ out-of-sample error for each part of the data
  $\mathbf{E}_{J_j} = \left\{y_i - \mathbf{x}_i^T\boldsymbol{\beta}_{J_j}^*\right\}_{i\in J_j}$ in-sample error for $\frac{N-1}{N}$ nth part of the data
end
$\boldsymbol{\beta}^* = \boldsymbol{\beta}_{I_0}^*$ report the weights acquired from the whole data
$\mathbf{e}_{I_0} = \cup\ (\mathbf{e}_{I_j})$ (*) out-of-sample error assembled, report as a histogram
$\mathbf{E}_{I_0} = \cup\ (\mathbf{e}_{J_j})$ (*) in-sample error assembled, report as a histogram
$e = mean(\mathbf{e}_{I_0})$ average error, report (same with *E*)
$\sigma^2 = var(\mathbf{e}_{I_0})$ variance, report (same with *E*)

Note that in-sample error vector error you get actually N-1 values per each sample. So, lengths of error vectors are: $\left|\mathbf{e}_{I_0}\right| = n$, $\left|\mathbf{E}_{I_0}\right| = (N-1)n$. But you can construct histograms anyways. The union operator in (*) is simply a vector concatenation.

2.2) You can find e.g. 8 nearest neighbors for all data points beforehand. E.g. Matlab *knnsearch* (http://se.mathworks.com/help/stats/knnsearch.html, experiment with simple data) and python (sklearn.neighbors, http://scikit-learn.org/stable/modules/neighbors.html , again, experiment first).[1]

Let us assume you have coded a function *y= lookup(I,* **x***,k)*= $mean_{i\in N_k(\mathbf{x})}\{y_i\}$, which returns the average of *y* values of *k* neighbors $N_k(\mathbf{x})$ of **x**. I is an index set referring to data (**D,y**):

for k=1…8:
  for *j= 1..N*:
    $\mathbf{e}_{I_j} = \{y_i - lookup(J_j, \mathbf{x}_i, k)\}_{i\in I_j}$ out-of-sample absolute error
    $\mathbf{E}_{J_j} = \{y_i - lookup(J_j, \mathbf{x}_i, k)\}_{i\in J_j}$ in-sample absolute error
  end
end

k_{best} is the one with the smallest absolute out-of-sample mean error *e*. The rest as in 1).

If things turn rough…

You can always do a partial submit. Do e.g. only 2.2), or do only in-sample error, or just one prediction over whole data and absolute errors from that. This results in less points, of course…

Points decline after the deadline…

Monday/Tue/Wed/Thu/Fri/Sat/Sun			
00:00 Tuesday…	exercise 1	exercise 2	…Monday 23:55
…	3	3	Nov 30th
Dec 1st	2	3	Dec 7th
Dec 8th	1	3	Dec 14th
Dec 15th	0	2	Dec 21st
Dec 22st	0	1	Dec 28th
Dec 29th	0	0	Jan 4th
Jan 5th	disqualified		…

[1] Aficionados: Do the preliminary neighbors search for each partitions I_j separatedly. Use e.g. $k' = 12$ with *knnsearch* (etc.) to make sure you find N_8 (relatively) easily.