Supervised Learning (COMP0078) – Coursework 1

Mark Herbster

Due : 14 November 2019.

Submission

You may work in groups of up to two.[1] You should produce a report about your results. You will not only be assessed on the **correctness/quality** of your answers but also on **clarity of presentation**. Additionally make sure that your code is *well commented*. Please submit on moodle i) your report (pdf) as well as a ii) zip file with your source code. Finally, please ensure that if you are working in a group 1) both of your names are on the coversheet and 2) both of you submit to moodle. Regarding the use of libraries, you should implement regression (with and without basis functions, kernels) using matrix algebra directly. Otherwise libraries are okay.

Questions please e-mail `sl-support@cs.ucl.ac.uk` .

1 PART I [50%]

1.1 Linear Regression

"Pluralitas non est ponenda sine neccesitate" – William of Ockham (ca. 1285-1349)

Linear regression overview: Given a set of data:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\} \tag{1}$$

where $\boldsymbol{x} = (x_1, \ldots, x_n)$ is a vector in $\Re^n$ and y is a real number. Linear regression finds a vector $\boldsymbol{w} \in \Re^n$ such that the sum of squared errors

$$\mathrm{SSE} = \sum_{t=1}^{m} (y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2 \tag{2}$$

is minimized. This is expressible in matrix form by defining X to be the $m \times n$ matrix

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\ x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \cdots & x_{m,n} \end{pmatrix}, \tag{3}$$

and defining $\boldsymbol{y}$ to be the column vector $\boldsymbol{y} = (y_1, \ldots, y_m)$. The vector $\boldsymbol{w}$ then minimizes

$$(X\boldsymbol{w} - \boldsymbol{y})^{\intercal}(X\boldsymbol{w} - \boldsymbol{y}).$$

In linear regression with basis functions we fit the data sequence with a linear combination of basis functions $(\phi_1, \phi_2, \ldots, \phi_k)$ where $\phi_i : \Re^n \to \Re$ which defines a feature map from $\boldsymbol{\phi} : \Re^n \to \Re^k$

$$\boldsymbol{\phi}(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \ldots, \phi_k(\boldsymbol{x})), \quad \boldsymbol{x} \in \Re^n .$$

[1] Affiliates students should work alone.

We use the basis functions to transform the data as follows

$$\{((\phi_1(\boldsymbol{x}_1),\ldots,\phi_k(\boldsymbol{x}_1)),y_1),\ldots,((\phi_1(\boldsymbol{x}_m),\ldots,\phi_k(\boldsymbol{x}_m)),y_m)\}, \tag{4}$$

and then applying linear regression above to this transformed dataset. In matrix notation we may denote this transformed dataset as

$$\Phi := \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_m) \end{pmatrix} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \ldots & \phi_k(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_m) & \ldots & \phi_k(\boldsymbol{x}_m) \end{pmatrix}, \quad (m \times k) \tag{5}$$

Linear regression on the transformed dataset thus finds a k-dimensional vector $\boldsymbol{w} = (w_1, \ldots, w_k)$ such that

$$(\Phi \boldsymbol{w} - y)^T(\Phi \boldsymbol{w} - y) = \sum_{t=1}^{m}(y_t - \sum_{i=1}^{k} w_i \phi_i(\boldsymbol{x}_t))^2 \tag{6}$$

is minimized.

A common basis ($n = 1$) used is the polynomial basis $\{\phi_1(x) = 1, \phi_2(x) = x, \phi_3(x) = x^2, \phi_4(x) = x^3, \ldots, \phi_k(x) = x^{k-1}\}$ of dimension k (order $k-1$) in the figure below we give a simple fit of four points produced by a linear ($k = 2$) and cubic ($k = 4$) polynomial.

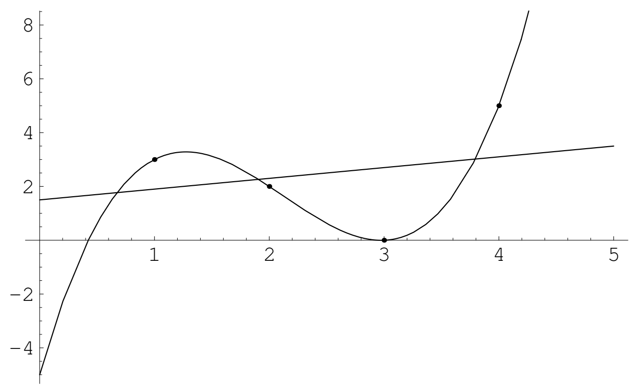
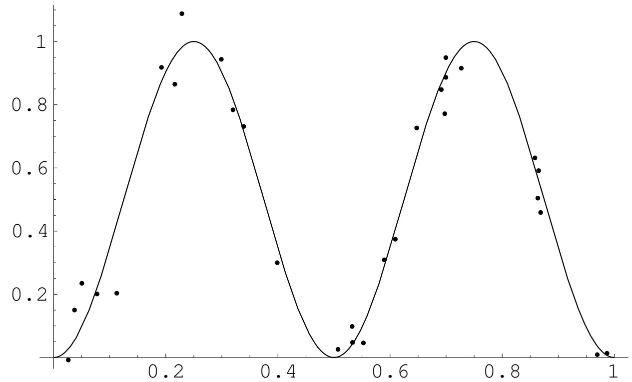


Figure 1: Data set $\{(1,3),(2,2),(3,0),(4,5)\}$ fitted with basis $\{1, x\}$ and basis $\{1, x, x^2, x^3\}$

1. **[5 pts]:** For each of the polynomial bases of dimension $k = 1, 2, 3, 4$ fit the data set of Figure 1 $\{(1,3),(2,2),(3,0),(4,5)\}$.
   (a) Produce a plot similar to Figure 1, superimposing the four different curves corresponding to each fit over the four data points.
   (b) Give the equations corresponding to the curves fitted for $k = 1, 2, 3$. The equation corresponding to $k = 4$ is $-5 + 15.17x - 8.5x^2 + 1.33x^3$.
   (c) For each fitted curve $k = 1, 2, 3, 4$ give the mean square error where $\text{MSE} = \frac{\text{SSE}}{m}$.
2. **[10 pts]:** In this part we will illustrate the phenomena of *overfitting*.
   (a) Define
   $$g_\sigma(x) := \sin^2(2\pi x) + \epsilon. \tag{7}$$
   where ϵ is a random variable distributed normally with mean 0 and variance σ^2 thus $g_\sigma(x)$ is a *random* function such that $\sin^2(2\pi x)$ is computed and then the normal noise is added on each "call" of the function. We then sample "x_i" uniformly at random from the interval $[0, 1]$ 30 times creating $(x_1, \ldots, x_{30})$ and apply $g_{0.07}$ to each x creating the data set
   $$S_{0.07,30} = \{(x_1, g_{0.07}(x_1)), \ldots, (x_{30}, g_{0.07}(x_{30}))\}. \tag{8}$$
      i. Plot the function $\sin^2(2\pi x)$ in the range $0 \le x \le 1$ with the points of the above data set superimposed. The plot should resemble



ii. Fit the data set with a polynomial bases of dimension $k = 2, 5, 10, 14, 18$ plot each of these 5 curves superimposed over a plot of data points.[2]

(b) Let the training error $\mathrm{te}_k(S)$ denote the MSE of the fitting of the data set S with polynomial basis of dimension k. Plot the natural log (ln) of the training error versus the polynomial dimension $k = 1, \ldots, 18$ (this should be a decreasing function).

(c) Generate a test set T of a thousand points,

$$T_{0.07,1000} = \{(x_1, g_{0.07}(x_1)), \ldots, (x_{1000}, g_{0.07}(x_{1000}))\}. \tag{9}$$

Define the test error $\mathrm{tse}_k(S, T)$ to be the MSE of the test set T on the polynomial of dimension k fitted from training set S. Plot the ln of the test error versus the polynomial dimension $k = 1, \ldots, 18$. Unlike the training error this is not a decreasing function. This is the phenomena of *overfitting*. Although the training error decreases with growing k the test error eventually increases since rather than fitting the function, in a loose sense, we begin to fit to the noise.

(d) For any given set of random numbers we will get slightly different training curves and test curves. It is instructive to see these curves smoothed out. For this part repeat items (c) and (d) but instead of plotting the results of a single "run" plot the average results of a 100 runs (note: plot the ln(avg) rather than the avg(ln)).

3. **[5 pts]:** Now use basis (for $k = 1, \ldots, 18$)

$$\sin(1\pi x), \sin(2\pi x), \sin(3\pi x), \ldots, \sin(k\pi x).$$

Repeat the experiments in 2 (b-d) with the above basis.

1.2 Filtered Boston housing and kernels

In this section we will use kernel methods to extend linear regression. Boston housing is a classic dataset where you are given 13 values and a goal is to predict the 14th which is the median house price. **Added 24 October 19:** `Instead of the original dataset we will instead use a modified dataset removing the ethically-suspect ''column B.'' Thus we will use 12 attributes to predict the 13th.` The unmodified dataset is described in more detail at `http://www.cs.toronto.edu/~delve/data/boston/bostonDetail.html`. There are 506 entries which we will split into train and test.

The **filtered** boston housing data set as a ".csv' file is located at

`http://www.cs.ucl.ac.uk/staff/M.Herbster/boston-filter`

4. **[10 pts]:** "Baseline versus full linear regression."
Rather than use all of our attributes for prediction it is often useful to see how well a baseline method works for a problem. In this exercise, we will compare the following:

[2]Depending on you implementation you may have numerical errors for large values of k this is 'ok' for the purposes of this exercise.

(a) Predicting with the mean y-value on the training set.

(b) Predicting with a single attribute and a bias term.

(c) Predicting with all the attributes

The training set should be 2/3, and the test set should be 1/3, of your data in (a)-(c). In the following average your results over 20 runs (each run based on a different (2/3,1/3) random split).

a. Naive Regression. Create a vector of ones that is the same length as the training set using the function `ones`. Do the same for the test set. By using these vectors we will be fitting the data with a constant function. Perform linear regression on the training set. Calculate the MSE on the training and test sets and note down the results.

b. Give a simple interpretation of the constant function in 'a.' above.

c. Linear Regression with single attributes. For each of the twelve attributes, perform a linear regression using only the single attribute but incorporating a bias term so that the inputs are augmented with an additional 1 entry, $(\boldsymbol{x}_i, 1)$, so that we learn a weight vector $\boldsymbol{w} \in \Re^2$.

d. Linear Regression using all attributes. Now we would like to perform linear regression using all of the data attributes at once.

   Perform linear regression on the training set using this regressor, and incorporate a bias term as above. Calculate the MSE on the training and test sets and note down the results. You should find that this method outperforms any of the individual regressors.

1.3 Kernelised ridge regression

For nonlinear regression, the dual version will prove important. Obviously, linear regression is not capable of achieving a good predictive performance on a nonlinear data set. Here, the dual formulation will prove extremely useful, in combination with the *kernel trick*.

For our exercises we will use a slight variation on the optimisation (as compared to the notes) that defines ridge regression for a training set with ℓ examples,

$$\boldsymbol{w}^* = \text{argmin}_{\boldsymbol{w}\in\Re^n} \frac{1}{\ell}\sum_{i=1}^{\ell}(\boldsymbol{x}_i^\intercal \boldsymbol{w} - y_i)^2 + \gamma \boldsymbol{w}^\intercal \boldsymbol{w}. \tag{10}$$

i.e, we have replace the sum of the square errors with the mean square error (MSE).[3] For a given kernel function K define the kernel matrix $\boldsymbol{K}$ for a training set of size ℓ elementwise via

$$K_{i,j} := K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

The dual optimisation formulation after kernelization is

$$\boldsymbol{\alpha}^* = \text{argmin}_{\boldsymbol{\alpha}\in\Re^\ell} \frac{1}{\ell}\sum_{i=1}^{\ell}(\sum_{j=1}^{\ell}\alpha_j K_{i,j} - y_i)^2 + \gamma \boldsymbol{\alpha}^\intercal \boldsymbol{K} \boldsymbol{\alpha}. \tag{11}$$

Now if we use $\boldsymbol{y} := (y_1, \ldots, y_\ell)^\intercal$ to denote a vector that contain the y-values of training set we may solve in the dual as follows

$$\boldsymbol{\alpha}^* = (\boldsymbol{K} + \gamma\ell\mathbf{I}_\ell)^{-1}\boldsymbol{y} \tag{12}$$

where $\mathbf{I}_\ell$ denotes the $\ell \times \ell$ identity matrix. The evaluation of the regression function on a test point can be reformulated as:

$$y_{\text{test}} = \sum_{i=1}^{\ell} \alpha_i^* K(\boldsymbol{x}_i, \boldsymbol{x}_{\text{test}}) \tag{13}$$

[3]The motivation is that we wish the regularisation parameter γ to 'scale' somewhat independently of training set size.

where the K is the kernel function.

5. **[20 pts]** "Kernel Ridge Regression"
In this exercise we will perform kernel ridge regression (KRR) on the data set using the Gaussian kernel,

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \exp\left(-\frac{\|\boldsymbol{x}_i - \boldsymbol{x}_j\|^2}{2\sigma^2}\right). \tag{14}$$

For this exercise, you will hold out 2/3 of data for training and report the test results on the remaining 1/3.

a. Create a vector of γ values $[2^{-40}, 2^{-39}, \ldots, 2^{-26}]$ and a vector of σ values $[2^7, 2^{7.5}, \ldots, 2^{12.5}, 2^{13}]$ (recall that σ is a parameter of the Gaussian kernel see equation (14)). Perform kernel ridge regression on the *training* set using five-fold cross-validation to choose among all pairing of the values of γ and σ. Choose the γ and σ values that perform the best to compute the predictor (by then retraining with those parameters on the training set) that you will use to report the test and training error.

b. Plot the "cross-validation error" (mean over folds of validation error) as a function of γ and σ.

c. Calculate the MSE on the training and test sets for the best γ and σ.

d. Repeat "exercise 4a,c,d" and "exercise 5c" over 20 random $(2/3, 1/3)$ splits of your data record the train/test error and the standard deviations (σ') of the train/test errors and summarise these results in the following type of table.

Method	**MSE train**	**MSE test**
Naive Regression	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 1)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 2)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 3)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 4)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 5)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 6)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 7)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 8)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 9)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 10)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 11)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (attribute 12)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Linear Regression (all attributes)	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$
Kernel Ridge Regression	$??.?? \pm \sigma'?$	$??.?? \pm \sigma'?$

2 PART II [50%]

2.1 Questions

Notation: We overload $[\cdot]$ as follows $[n] := \{1, 2, \ldots, n\}$ if n is a positive integer and $[\texttt{pred}] = 1$ if `pred` is a logical predicate which is true and $[\texttt{pred}] = 0$ otherwise.

6. **[20 pts]** *Bayes estimator.* In both of the following subquestions you will need to find the Bayes estimator with respect to the the probability mass function $p(x, y)$ over (X, Y) where X and Y are finite thus $\sum_{x \in X} \sum_{y \in Y} p(x, y) = 1$.

(a) For this subquestion $Y = [k]$ and let $\boldsymbol{c} \in [0, \infty)^k$ be a vector of k costs. Define $L_{\boldsymbol{c}} : [k] \times [k] \to [0, \infty)$ as,

$$L_{\boldsymbol{c}}(y, \hat{y}) := [y \neq \hat{y}] c_y$$

as the the imbalanced classification loss function, i.e., if we don't predict the correct outcome y we suffer c_y loss. Derive the Bayes estimator.

(b) For this subquestion $Y \subset \Re$. Let $L(y, \hat{y}) := |y - \hat{y}|$. Derive the Bayes estimator.

7. **[10 pts]** *Kernel modification* Consider the function $K_c(\boldsymbol{x}, \mathbf{z}) := c + \sum_{i=1}^{n} x_i z_i$ where $\boldsymbol{x}$, $\mathbf{z} \in \Re^n$.

   (a) For what values of $c \in \Re$ is K_c a positive semidefinite kernel? Give an argument supporting your claim. ("The closer your argument is to a proof the more likely it is to receive full credit.")

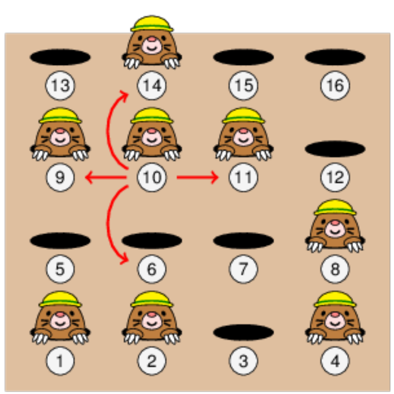
   (b) Suppose we use K_c as a kernel function with linear regression (least squares). Explain how c influences the solution.

8. **[15 pts]** Suppose we perform linear regression with a Gaussian kernel $K_\beta(\boldsymbol{x}, \mathbf{t}) = \exp(-\beta \|\boldsymbol{x} - \mathbf{t}\|^2)$ to train a classifier on a dataset $(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m) \in \Re^n \times \{-1, 1\}$. Thus obtaining a function $f : \Re^n \to \Re$ which is of the form $f(\mathbf{t}) = \sum_{i=1}^{m} \alpha_i K_\beta(\boldsymbol{x}_i, \mathbf{t})$. The corresponding classifier is then $\text{sign}(f(\mathbf{t}))$. This classifier depends on the parameter β selected for the kernel. In what scenario will chosen β enable the trained linear classifier to simulate a 1-Nearest Neighbor classifier trained on the same dataset?

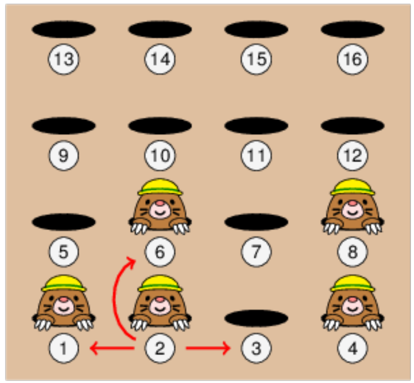
   Note β may be selected as a function of $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m$ and the test point $\mathbf{t}$, i.e., we may use a function so that $\beta = \hat{\beta}(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m, \mathbf{t})$ an exact function $\hat{\beta}(\cdot)$ need not be given just an argument that one exists.

   Give an argument supporting your reasoning. ("The closer your argument is to a proof the more likely it is to receive full credit.")

9. **[5pts]**: *Hint: "Think about vector spaces 'broadly'".*
   We consider the problem of generalized Whack-A-Mole! We'll play this game on a board with $n \times n$ holes, from which individual moles pop-up. Using a mallet you can then whack a mole and make it go hide underground. As you might know, however, whacking a mole will often have the effect of making other moles rise and come out from the ground. In this version of the game, every time you hit a hole with the mallet you'll cause the hole and the immediate four adjacent holes to change: any moles currently there will hide, while new moles pop-up from previously empty holes. Imagine, for example, that you find yourself in the following game configuration, and you prepare yourself to hit the hole at position 10 with the mallet.

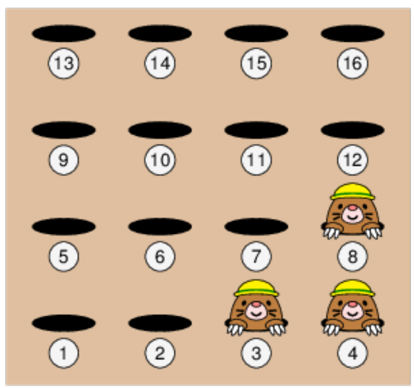


   You will cause the moles in positons 9, 10, 11, and 14 to hide, but you'll also make a mole appear at position 6. Thus ending up in the next configuration.



If you then hit the hole at position 2, then all of the moles at positions 1, 2, and 6 will hide; but a new one in position 3 will appear.

Example: Consider the 4×4 board with moles at positions 1,2,4,8,9,10,11, and 14. This is the same configuration of moles as given in the initial illustration above. After hitting holes at positions 10 then 2, you'll end up in the following configuration.



At this point, a single hit to position 4 will clear the board. Thus the hitting sequence $(10, 2, 4)$ is a solution.

Task: Design an algorithm for an $n \times n$ board which, given an initial board configuration, finds a sequence of holes that you can hit in order to empty the board if such a sequence exists. The algorithm must be polynomial in n and you must provide an argument that it is correct. No credit will be given for algorithms that are not polynomial in n. There is no need to implement your algorithm.