COS 424
Homework #1
Due Tuesday, February 23rd

See the course website for important information about collaboration and late policies, as well as where and when to turn in assignments.

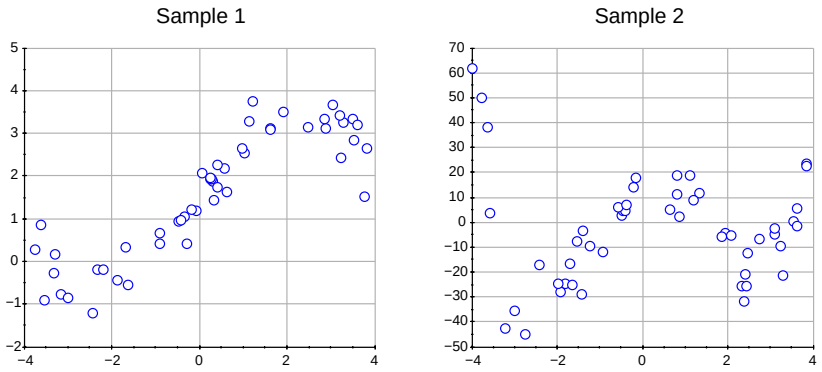
Data files

The questions make use of two data files available from
`http://www.cs.princeton.edu/courses/archive/spring10/cos424/w/hw1`.

```
hw1_sample2_train.txt
hw1_sample2_train.txt
```

Each data file contains $n = 50$ lines. Each line contains exactly 2 numbers representing the X and Y coordinates of points generated by adding noise to a secret function. The files `hw1_sample2_train.txt` and `hw1_sample2_train.txt` are only useful for the last question.

For verification, here is a graphical representation of both datasets.



Question 1

Implement linear least square curve fitting.

Provide plots showing the points and the fitted curve for both datasets using the following bases.

- Linear regression: $\Phi(x) = (1, x)$
- Cubic regression: $\Phi(x) = (1, x, x^2, x^3)$
- Cubic splines: $\Phi(x) = (1, x, x^2, x^3, [x+2]_+^3, [x]_+^3, [x-2]_+^3)$

The notation $[z]_+$ is a shorthand for $\max(0, z)$.

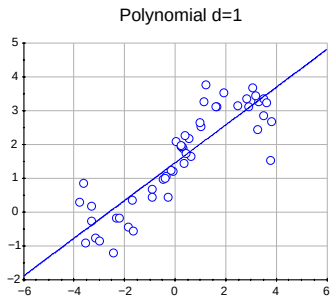
> We need only three main subroutines:
>
> - `phi(x)` computes returns the vector $\Phi(x)$ for the chosen basis.

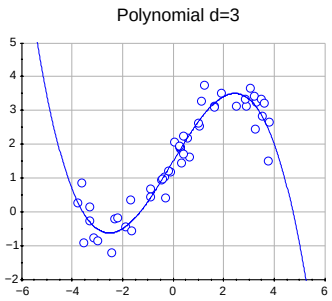
- `train(data)` computes the weights w for the given training points. Argument `data` is a matrix $n \times 2$ containing the training set. The following pseudo-code for the function `train` relies on a linear equation solver `solve`. Adding 10^{-6} to the diagonal of `xx` avoids numerical problems when the matrix is singular or near-singular.

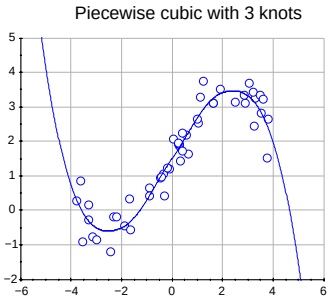
```
function train(data)
   n := data.nrows()
   d := phi(0).nrows()
   xx := new matrix(d,d)
   xy := new vector(d)
   for i := 1 to n
     phix := phi(data[i,0])
     xx := xx + phix * transpose(phix)
     xy := xy + phix * data[i,1]
   endfor
   for j := 1 to d
     xx[j,j] := xx[j,j] + 1e-6
   endfor
   return solve(xx, xy)
 endfunction
```

- `test(x, w)` returns `transpose(w)*phi(x)`, that is, the value in x of the curve with weights w.

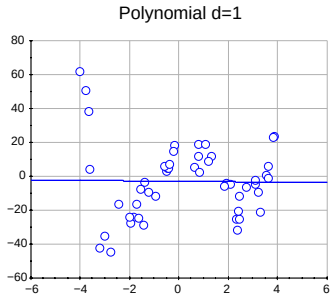
Plots for SAMPLE1

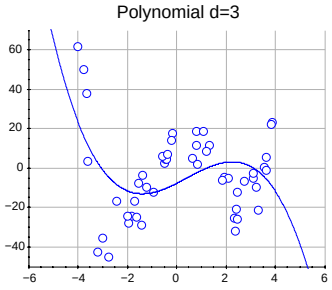


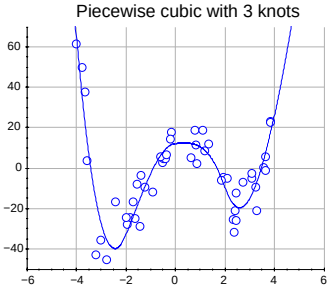




Plots for SAMPLE2







Question 2

Like polynomials, cubic splines often display strong variations outside of the domain of the input examples. Given a set of knots $r_1 < r_2 < \cdots < r_k$, we would like to construct a spline basis that always produces a piecewise function whose pieces are affine (polynomials of degree 1) when $x \leq r_1$ or $x \geq r_k$ and cubic (polynomials of degree 3) when $r_1 < x < r_k$.

Cubic splines can be written as

$$f(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \sum_{i=1}^{k} b_i [x - r_i]_+^3 \tag{1}$$

a) Write conditions on the parameters a_i and b_i expressing that the spline function has no cubic term when $x < r_1$ or $x > r_k$.

When $x < r_1$, $f(x) = a_3 x^3 + a_2 x^2 + a_1 x + a_0$.
When $x > r_k$, $f(x) = (a_3 + \sum_{i=1}^{k} b_i)\, x^3 + \ldots x^2 + \ldots x + \ldots$
The conditions are therefore $a_3 = 0$ and $\sum_{1=1}^{k} b_i = 0$.

b) Derive a spline basis that embodies these conditions.
Hint: write $c_i = \sum_{j=1}^{i} b_j$ and rewrite expression (1) with the c_i instead of the b_i.
The resulting splines should have the form

$$f(x) = a_0 + a_1 x + a_2 x^2 + \sum_{i=1}^{k-1} c_i \phi_i(x) \tag{2}$$

Since $b_i = c_i - c_{i-1}$ and $a_3 = 0$, we can rewrite (1) as:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \sum_{i=1}^{k} (c_i - c_{i-1})[x - r_i]_+^3$$

Reorganizing the sum and using the conditions $c_k = \sum_{i=1}^{k} b_i = 0$:

$$f(x) = a_0 + a_1 x + a_2 x^2 + \sum_{i=1}^{k-1} c_i ([x - r_i]_+^3 - [x - r_{i+1}]_+^3) \tag{3}$$

Therefore the basis is $\Phi(x) = (1, x, x^2, \ldots [x - r_i]_+^3 - [x - r_{i+1}]_+^3 \ldots)$ with $i = 1 \ldots k-1$.

c) Write further conditions on the coefficient a_i and c_i expressing that the spline function has no quadratic term when $x < r_1$ or $x > r_k$.

No quadratic terms when $x < r_1 \iff a_2 = 0$
Observe $(x-a)^3 - (x-b)^3 = 3(b-a)x^2 - 3(b^2 - a^2)x + (b^3 - a^3)$.
Then, when $x > r_k$, $\sum_{i=1}^{k-1} c_i ([x - r_i]_+^3 - [x - r_{i+1}]_+^3) = 3x^2 \sum_{i=1}^{k-1} c_i (r_{i+1} - r_i) + 3x \ldots + \ldots$.
The additional conditions are therefore $a_2 = 0$ and $\sum_{i=1}^{k-1} c_i (r_{i+1} - r_i) = 0$.

d) Derive a spline basis that embodies all these conditions, namely that the pieces of the resulting functions are affine when $x < r_1$ or $x > r_k$. Such splines are called *natural cubic splines*.

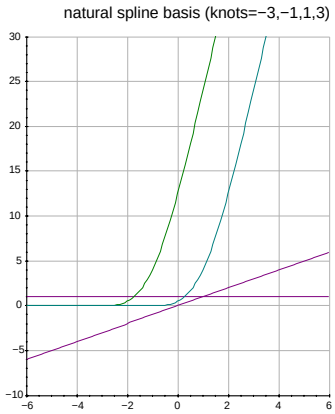
Same story as question (2b). Define $d_i = \sum_{j=1}^{i} c_i(r_{i+1} - r_i)$; recall $a_2 = 0$; and write (3) as

$$f(x) = a_0 + a_1 x + \sum_{i=1}^{k-1} \frac{d_i - d_{i-1}}{r_{i+1} - r_i}([x - r_i]_+^3 - [x - r_{i+1}]_+^3)$$

Reorganizing the sum and using the condition $d_{k-1} = 0$,

$$f(x) = a_0 + a_1 x + \sum_{i=1}^{k-2} d_i \left(\frac{[x - r_i]_+^3 - [x - r_{i+1}]_+^3}{r_{i+1} - r_i} - \frac{[x - r_{i+1}]_+^3 - [x - r_{i+2}]_+^3}{r_{i+2} - r_{i+1}} \right)$$

The basis is then $\Phi(x) = \left(1, x, \ldots \frac{[x-r_i]_+^3 - [x-r_{i+1}]_+^3}{r_{i+1}-r_i} - \frac{[x-r_{i+1}]_+^3 - [x-r_{i+2}]_+^3}{r_{i+2}-r_{i+1}}, \ldots\right)$ with $i = 1 \ldots k-2$.



This basis is plotted on the left with 4 knots
The non trivial basis functions are composed of four segments (flat, cubic, cubic, linear) with boundaries at the knots.
There are other possible basis for natural splines such as: $\left(1, x, \ldots \frac{[x-r_i]_+^3 - [x-r_{i+1}]_+^3}{r_{i+1}-r_i} - \frac{[x-r_{k-1}]_+^3 - [x-r_k]_+^3}{r_k - r_{k-1}} \ldots\right)$ or $\left(1, x, \ldots \frac{[x-r_i]_+^3 - [x-r_k]_+^3}{r_k - r_i} - \frac{[x-r_{k-1}]_+^3 - [x-r_k]_+^3}{r_k - r_{k-1}} \ldots\right)$.

e) Count the number of dimensions and the number of constraints (continuity, continuous first derivatives, continuous second derivatives) to confirm that the basis dimension should indeed be k.

Each segment of the fitted curve is a cubic polynomial. That would give $4(k+1)$ parameters. We have three constraints per knot stating that the function and its first two derivatives are continuous. Finally we have four additional constraints stating that the cubic and quadratic coefficients in the end pieces are zero. Therefore the final dimension is $4(k+1) - 3k - 4 = k$.

Question 3

We now consider natural splines with k evenly spaced knots such that $r_1 = -4$ and $r_k = 4$.

Use 5-fold cross-validation to determine the best k for each of the two samples.

Provide one plot for each dataset showing the average of the training and validation MSE obtained during k-fold cross-validation as a function of $k = 1 \ldots 8$.

First there is a problem with the question: there are no natural splines with $k < 2$. In fact the case $k = 2$ corresponds to a simple linear regression.

A typical code for computing the k-fold validation error contains

- A function `split_sample(data,nfolds,i)` to compute the i-th split of the data. This function returns two data matrices containing the training part and the testing part of the split. My

splitting function takes the data in the order of the data file and divides it in five segments of 10 examples. The testing part for the i-th fold is the i-th segment (10 examples). The training part is obtained by removing the i-th segment from the whole dataset (40 examples are left.)

- A function `mse_sample(data,w)` to compute the MSE of the curve w on sample *data*.

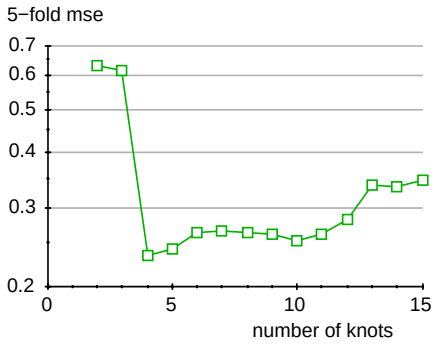
```
function mse_sample(data,w)
  err = 0
  for i=1 to data.nrows()
    err = err + square(data[i,1] - test(data[i,0], w))
  endfor
  return err / data.nrows()
endfunction
```

- A function `kfold_error(data,nfolds)` to perform the k-fold cross-validation.

```
function kfold_error(data,nfolds)
  err = 0
  for i=1 to nfolds
    trdata,tedata = split_sample(data,nfolds,i)
    err = err + mse_sample(tedata, train(trdata))
  endfor
  return err / nfolds
endfunction
```

The k-fold errors as a function of k look like the following plots. Depending on the details of the splitting function, there can be some differences...

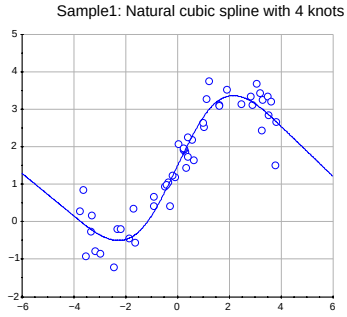
Sample1

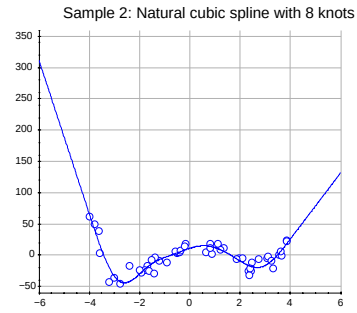


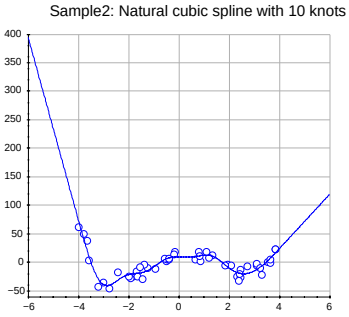
Sample2

Note that the best error for sample 2 is $k = 10$ which is greater than the maximal k specified in the question. Extra points for whoever pointed this out.

The following plots show curve fits for selected solutions.







Question 4

Download the two testing sets provided on the homework page and report the testing set MSE achieved by the models you have selected.

The code for this is trivial when you have programmed the k-fold validation.

The following tables give my results. The 5-fold MSE and the optimal k can vary depending on the details of your splitting function. But the test MSE should be the same because it is obtained by retraining on the whole dataset with the selected value of k.

Sample 1		
k	**5-fold MSE**	**test MSE**
2	0.63	0.73
3	0.61	0.70
4	**0.23**	**0.23**
5	0.24	0.24
6	0.26	0.24
7	0.27	0.26
8	0.26	0.27
9	0.26	0.27
10	0.25	0.28
11	0.26	0.30
12	0.28	0.29
13	0.34	0.30
14	0.33	0.32
15	0.35	0.33

Sample 2		
k	**5-fold mse**	**test mse**
2	530.53	520.29
3	557.35	537.47
4	551.26	516.63
5	214.57	173.92
6	183.30	138.38
7	135.94	120.08
8	116.75	115.46
9	111.45	115.16
10	**110.90**	**116.87**
11	118.76	113.58
12	162.39	110.68
13	258.07	110.20
14	385.17	133.13
15	332.35	154.03

Note how k-fold cross-validation underestimates the best k on sample2.
This probably happens because each fold is trained on 40 examples instead of 50.