GI07/M012
Homework #1
(Due 5 December 2016)

You should produce a report about your results (including all requested plots). You will not only be assessed on the correctness/quality of your code/experiments but also on clarity of presentation. This includes but is not limited to insuring that your code is *well commented* and that each plot has a descriptive title, and a legend if necessary. Please submit 1) the coursework to the department office on the due date at 12:00 noon with the following linked coversheet `http://www.cs.ucl.ac.uk/fileadmin/UCL-CS/students/documents/cwsheet.pdf` 2) on moodle please submit a zip file containing your coursework as well as your source code.

1 Gradient Descent

Aim: This set of exercises has a three fold aim i) to obtain a basic familiarity with matlab ii) a basic familiarity with gradient descent iii) using matlab to as an aid for mathematical reasoning (exercise 3).
Files for these exercises are available from `http://www.cs.ucl.ac.uk/staff/M.Herbster/GI07/` (e.g. `graddesc, fc,dfc`)
Recall gradient descent with fixed step size:

`Gradient descent method`
`given` $x_0 \in$ domain f and a step size $\lambda > 0$
`repeat`
$$x_{t+1} = x_t - \lambda \nabla_x f(x_t)$$
`until` stopping criteria is satisfied.

1. **[15 pts `basic`]:** In this first exercise we will produce a simple visualization of a gradient descent algorithm. Consider the function,
$$f(x, y) = (x-2)^2 + 2(y-3)^2$$

```
>> [X,Y] = meshgrid(linspace(0,5,15),linspace(0,5,15)) ;
>> mesh(X,Y,fcarg(X,Y)) ;
```

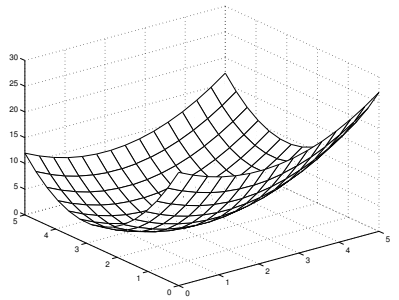


Figure 1

Algebraically, we see that $(2, 3)$ is the minima of this function. Numerically we may use the matlab function *graddesc.m* to calculate the minima.

```
>> graddesc('fc','dfc',[0,0],0.1,0.1)
ans =
    1.9550    2.9995
```

Suppose we start a gradient descent algorithm at (0,0) on the way to the minima we traverse a series of points,

$$\{(0, 0, f(0, 0)), (x_2, y_2, f(x_2, y_2)), (x_3, y_3, f(x_3, y_3)), \ldots, \sim (2.0, 3.0, 0.0))\} \quad (1)$$

Visualizing this path in three dimensions we have,

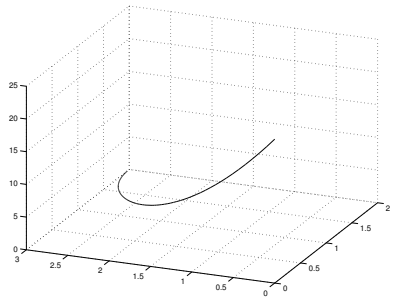


Figure 2

Projecting down to the xy plane we have,

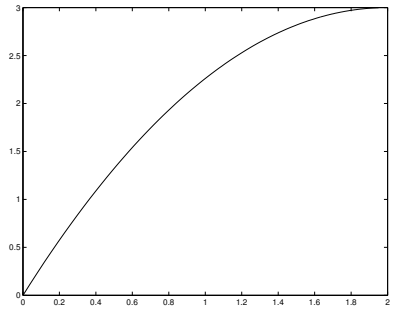


Figure 3

(a) Produce a plot similar to Figure 1. Show commands used.

(b) Modify the function *graddesc.m* to produce a sequence of points as in Equation 1.

i. Hand in modified code along with the following plots (subparts ii, iii) and their code. Ensure that the curves produced by the plots are as *smooth* as the figures.

ii. Produce a plot similar to Figure 2. (Hint: to do this it will be necessary to massage the sequence of points into a form usable by **plot3()**, then to produce the grid use the cmd **grid on**.

iii. Produce a plot similar to Figure 3.

2. **[15 pts `intermediate`]:** In this exercise we will use *gradient descent* to perform linear regression (linear least squares). Consider the matrix equation,

$$A\boldsymbol{x} = \boldsymbol{b}$$

if there is no exact solution then for any potential solution x we may define a column vector of error terms

$$\boldsymbol{e} = A\boldsymbol{x} - b.$$

The sum of the errors squared is then

$$\boldsymbol{e}^T\boldsymbol{e}$$

in matlab this is just $\boldsymbol{e}' * \boldsymbol{e}$. Thus the least squares solution is the $\boldsymbol{x}$ that minimizes

$$(A\boldsymbol{x} - \boldsymbol{b})^T(A\boldsymbol{x} - \boldsymbol{b}).$$

In Matlab

```
>> A\b
```

computes the least square solution directly.

(a) Give a matlab function to compute the least squares solution by gradient descent. The arguments should include the matrix A, the column vector $\boldsymbol{b}$, an initial guess, a step size, and a tolerance (i.e., a convergence criteria). For example: `mydescent(A,b,guess,step,tol)`. *Note: that standard gradient is a very inefficient method to compute the least squares solution to a set of equations.* Also observe in order to code `mydescent.m` you will need to determine the symbolic form of $\nabla_{\boldsymbol{x}}[(A\boldsymbol{x} - \boldsymbol{b})^T(A\boldsymbol{x} - \boldsymbol{b})]$ where $\nabla_{\boldsymbol{x}}$ is the gradient with respect to $\boldsymbol{x}$.

(b) Use the above function to give a least squares solution to the equations

$$\begin{aligned} x_1 - x_2 &= 1 \\ x_1 + x_2 &= 1 \\ x_1 + 2x_2 &= 3 \end{aligned}$$

(c) Visualize the above solution with a plot as in Figure 3.

`hint:` If you are unfamiliar with matrix differentiation. It may be valuable to work out the gradient in a more "explicit" form then convert the result back into a simple matrix form. Thus observe, if matrix A is $m \times n$ then

$$(A\boldsymbol{x} - \boldsymbol{b})^T (A\boldsymbol{x} - \boldsymbol{b}) = \sum_{i=1}^{m} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2$$

and

$$\nabla_{\boldsymbol{x}} \left[(A\boldsymbol{x} - \boldsymbol{b})^T (A\boldsymbol{x} - \boldsymbol{b}) \right] = \left(\sum_{i=1}^{m} \frac{\partial}{\partial_{x_1}} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2, \ldots, \sum_{i=1}^{m} \frac{\partial}{\partial_{x_n}} \left(\sum_{j=1}^{n} a_{ij} x_j - b_i \right)^2 \right) .$$

3. **[10 pts `hard`]:** This exercise explores the *convergence* of gradient descent in a single variable. Gradient descent can be defined as follows. Given $f(x)$, let f' denote the first derivative, let λ denote the step size and let x_0 denote the initial point, hence gradient descent can be defined as a sequence of iterates of $G_{f,\lambda}(x) = x - \lambda f'(x)$ which we will denote as follows

$$G^{(0)} = x_0,\ G^{(1)} = x_0 - \lambda f'(x_0),\ G^{(2)} = x_0 - \lambda f'(x_0) - \lambda f'(x_0 - \lambda f'(x_0)),\ \ldots \tag{2}$$

hence gradient descent on f with step size λ and starting point x_0 converges to x^* if $G^{(n)} \to x^*$ as $n \to \infty$. Gradient descent is said to converge with respect to an interval (x', x'') and a λ if for all $x \in (x', x'')$ we have that $G_{f,\lambda}(x)$ converges.

For the following exercises experimental arguments will receive a some credit however the best responses will include mathematical arguments. Notation: let $|x|$ denote the absolute value of x.

(a) Does there exist a starting point x_0 and step size λ such that gradient descent on $f(x) = |x-1|^3$ *nontrivially*[1] converges to 1. What is your evidence?

(b) Does there exist a starting point x_0 and step size λ such that gradient descent on $f(x) = \sqrt{|x-1|}$ *nontrivially* converges to 1. What is your evidence?

(c) For what values of $\lambda > 0$ does there exist an interval (x', x'') such that gradient descent on $f(x) = x^4 + 5x^2$ converges to 0. Why?

2 Linear Regression

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

Aim: This set of exercises is about to implementing linear regression with basis functions and visualising the phenomena of overfitting. Also included is a difficult challenge question.
Note: In this section it is **NOT** recommend to use gradient descent to find the solution to linear regression.

Linear regression overview: Given a set of data:

$$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\} \tag{3}$$

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

$$\text{SSE} = \sum_{t=1}^{m} (y_t - \boldsymbol{w} \cdot \boldsymbol{x}_t)^2 \tag{4}$$

[1] We say that gradient descent **non-trivially** converges from x_0 if there exists an ϵ such that gradient descent also converges with respect to the interval $(x_0 - \epsilon, x_0 + \epsilon)$.

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

$$X = \begin{pmatrix} x_{1,1} & x_{1,2} & \dots & x_{1,n} \\ x_{2,1} & x_{2,2} & \dots & x_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ x_{m,1} & x_{m,2} & \dots & x_{m,n} \end{pmatrix}, \tag{5}$$

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

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

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

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

We use the basis functions to transform the data as follows

$$\{((\phi_1(\boldsymbol{x}_1), \dots, \phi_k(\boldsymbol{x}_1)), y_1), \dots, ((\phi_1(\boldsymbol{x}_m), \dots, \phi_k(\boldsymbol{x}_m)), y_m)\}, \tag{6}$$

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

$$\Phi := \begin{pmatrix} \boldsymbol{\phi}(\boldsymbol{x}_1) \\ \vdots \\ \boldsymbol{\phi}(\boldsymbol{x}_m) \end{pmatrix} = \begin{pmatrix} \phi_1(\boldsymbol{x}_1) & \dots & \phi_k(\boldsymbol{x}_1) \\ \vdots & \ddots & \vdots \\ \phi_1(\boldsymbol{x}_m) & \dots & \phi_k(\boldsymbol{x}_m) \end{pmatrix}, \quad (m \times k) \tag{7}$$

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

$$(\Phi\boldsymbol{w} - y)^T (\Phi\boldsymbol{w} - y) = \sum_{t=1}^{m} (y_t - \sum_{i=1}^{k} w_i \phi_i(\boldsymbol{x}_t))^2 \tag{8}$$

is minimized.

A common basis ($n = 1$) used is the polynomial basis $\{\phi_1(x) = 1, \phi_2(x) = x, \phi_3(x) = x^2, \phi_4(x) = x^3, \dots, \phi_k(x) = x^{k-1}\}$ 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.

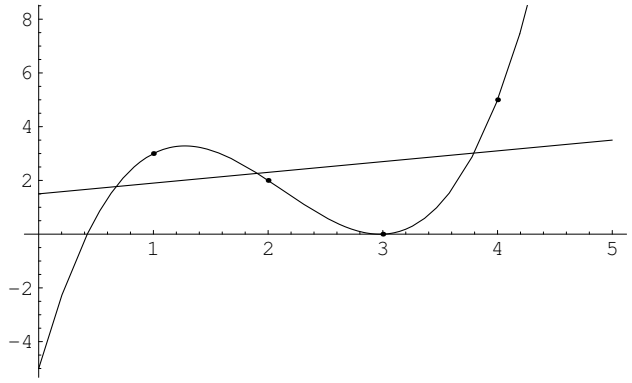


Figure 1: Data set $\{(1,3),(2,2),(3,0),(4,5)\}$ fitted with basis $\{1, x\}$ and basis $\{1, x, x^2, x^3\}$

1. **[15 pts `basic`]:** 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. (Use the matlab commands `hold on` and `hold off` to superimpose plots)

   (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 MSE $= \frac{\text{SSE}}{m}$.

2. **[10 pts `intermediate`]:** In this part we will illustrate the phenomena of *overfitting*. First we recall the normal distribution with mean μ and variance σ^2.

$$N_{\mu,\sigma}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(z-\mu)^2}{2\sigma^2}} \tag{9}$$

(a) The matlab function `randn` sample random numbers with the distribution $N_{0,1}$. Write a function that takes μ and σ as parameters and generates random numbers with a distribution $N_{\mu,\sigma}$.

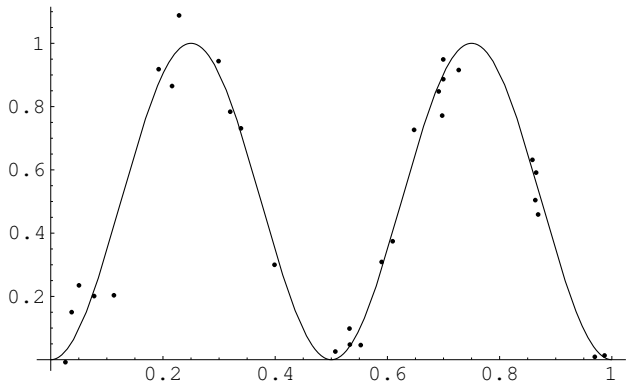
(b) Define

$$g_\sigma(x) = \sin^2(2\pi x) + N_{0,\sigma}. \tag{10}$$

where $g_\sigma(x)$ 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 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

$$S_{0.07,30} = \{(x_1, g_{0.07}(x_1)), \ldots, (x_{30}, g_{0.07}(x_{30})\}. \tag{11}$$

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.

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

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

$$T_{0.07,1000} = \{(x_1, g_{0.07}(x_1)), \ldots, (x_{1000}, g_{0.07}(x_{1000})\}. \tag{12}$$

Define the test error $\text{tse}_k(S, T)$ to be the MSE of the test set T on the polynomial of dimension k fitted from training set S. Plot the log 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.

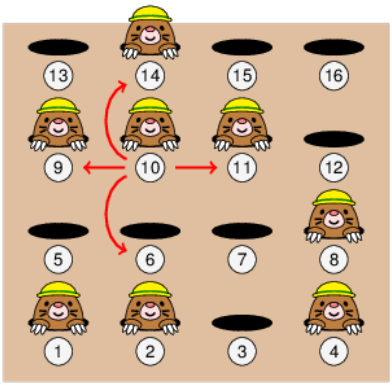
(e) For any given set of random numbers we will get slightly different training curves and test curves. It is instructive to see these curves smoothed out. For this part repeat items (c) and (d) but instead of plotting the results of a single "run" plot the average results of a 100 runs (note: plot the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

3. **[10 pts `intermediate`]:** In this part we will use as a basis

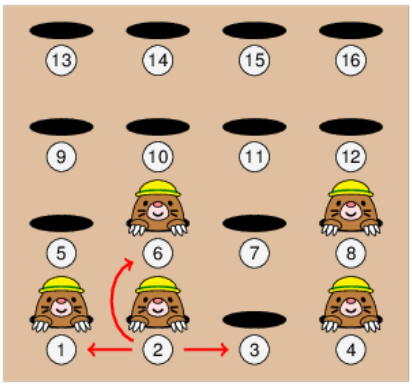
$$\sin(1\pi x), \sin(2\pi x), \sin(3\pi x), \ldots, \sin(k\pi x).$$

Repeat the experiments in 2 (c-e) with the above basis.

4. **[25 pts `challenge`]:** 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.

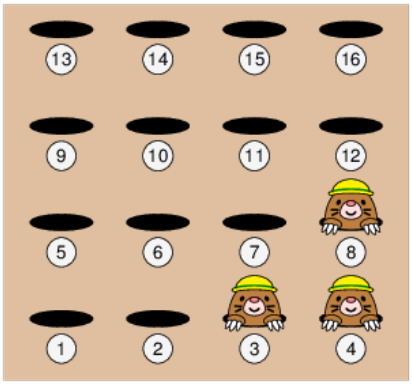


You will cause the moles in positons 9, 10, 11, and 14 to hide, but you'll also make a mole appear at position 6. Thus ending up in the next configuration.



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 should be polynomial in n and you should provide an argument that it is correct. No credit will be given for algorithms that are not polynomial in n.