University of Toronto Mississauga
Department of Mathematical and Computational Sciences
CSC 338 - Numerical Methods, Spring 2017

Assignment 2

Due date: Tuesday March 7, 11pm.
No late assignments will be accepted.

The material you hand in should be legible (either typed or *neatly* hand-written), well-organized and easy to mark, including the use of good English. In general, short, simple answers are worth more than long, complicated ones. Unless stated otherwise, all answers should be justified.

All computer problems are to be done in Python with NumPy and SciPy and should be properly commented and easy to understand. Hand in all the programs you are asked to write. They should be stored in a single file called `source.py`. We should be able to import this file as a Python module and execute your programs. For generating plots, you may find the SciPy functions `numpy.linspace` and `matplotlib.pyplot.plot` useful. Note that if `A` and `B` are two-dimensional numpy arrays, then `A*B` performs *pointwise* multiplication, *not* matrix multiplication. To perform matrix multiplication, you can use `numpy.dot(A,B)`.

You should hand in four files: the source code of all your Python programs, a pdf file of all the requested program output, answers to all the non-programming questions (scanned hand-writing is fine), and a scanned, signed copy of the cover sheet at the end of the assignment. Be sure to indicate clearly which question(s) each program and piece of output refers to. The four files should be submitted electronically as described on the course web page.

I don't know policy: If you do not know the answer to a question (or part), and you write "I don't know", you will receive 20% of the marks of that question (or part). If you just leave a question blank with no such statement, you get 0 marks for that question.

No more questions will be added

1. (16 points total) *LU factorization.*

   (a) (10 points) Write down the LU factorization of the following matrix:

   $$A = \begin{pmatrix} 1 & 2 & -3 \\ 2 & 3 & -5 \\ 3 & 2 & -3 \end{pmatrix}$$

   Do not use pivoting. You do not need to show the entire derivation, but do show each of the elimination matrices M_k and L_k used in the derivation.

   (b) (6 points) Use the LU factorizarion to solve the equations $Ax_1 = b_1$ and $Ax_2 = b_2$, where $b_1 = (2, 3, 4)^T$ and $b_2 = (5, 7, 3)^T$. You do not need to show the entire derivations, but do show the results of solving $Ly_i = b_i$ and $Ux_i = y_i$.

2. (15 points) Suppose A, B and C are non-singular $n \times n$ matrices, and b is an n-vector. How would you efficiently evaluate the following expression without computing any matrix inverses:

   $$((B^{-1})^2 + 3C)(A^{-1} + 5I)b$$

3. (10 points) Show that the number of multiplications in the LU factorization of a $n \times n$ matrix is $n^3/3 + O(n^2)$. Hint: $\sum_{i=1}^{n} i^k = n^{k+1}/(k+1) + O(n^k)$.

4. (8 points total) Which of the following matrices are orthogonal? (2 points each)

   (a) $\begin{pmatrix} -1 & 0 \\ 0 & -1 \end{pmatrix}$

   (b) $\begin{pmatrix} 0 & -1 \\ -1 & 0 \end{pmatrix}$

   (c) $\begin{pmatrix} -1 & 1 \\ 1 & 1 \end{pmatrix}$

   (d) $\frac{1}{\sqrt{3}} \begin{pmatrix} 1 & 2 \\ 2 & -1 \end{pmatrix}$

5. (7 points total) Consider the following linear least-squares problem:

   $$\begin{bmatrix} 1 & 2 \\ 0 & 3 \\ 0 & 0 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \approx \begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

   (a) (3 points) If we denote this problem $Ax \approx b$, what is the minimum value of $\|Ax - b\|$? Show your work.

   (b) (4 points) What is the solution vector, x, for this problem? Show your work.

6. (9 points total) We wish to annihilate all but the first entry of the vector $a = (1, 2, 2, 4)^T$ by using a Householder transformation, $H = I - 2vv^T/v^Tv$.

(a) (3 points) Without computing H, what is the value of Ha?

(b) (3 points) What is the value of v?

(c) (3 points) Use Python to compute H from v. Verify your answer in part (a) by using Python to compute Ha. If you do not get exactly the same answer as in part (a), explain why not. Hand in your computer code and the computed values of H and Ha. You may find the function `numpy.eye` useful.

7. (15 points total) Let H be the Householder matrix $I - 2vv^T/(v^Tv)$.

   (a) (3 points) Show that H is symmetric.

   (b) (5 points) Show that H is orthogonal.

   (c) (7 points) Let $v = a - \alpha e_1$, where a is a non-zero vector, $\alpha = \pm\|a\|$ and $e_1 = (1, 0...., 0)^T$. Show that $Ha = \alpha e_1$.

8. (15 points total) *Curve Fitting.*

   In this problem you will write a Python program to fit a curve to data using linear least-squares. First, download the file `data2.pickle.zip` from the course web site. Uncompress it (if your browser did not automatically do it). Start the Python interpreter and import the `pickle` module. You can then read the file `data2.pickle` with one of the following commands, the first for Python 2.x, and the second for Python 3.x:

```
with open('data2.pickle','rb') as f:
    data = pickle.load(f)

with open('data2.pickle','rb') as f:
    data = pickle.load(f,encoding='latin1')
```

   The variable `data` will now contain a 50×2 Numpy array. Each row of the array represents a 2-dimensional point, (t_i, y_i). The data is illustrated in the scatter plot in Figure 1 below.

   Your job is to fit a polynomial of degree 5 to this data, that is, to estimate the coefficients $x = (x_0, x_1, ..., x_5)$ in the function

$$y(t) = x_0 + x_1 t + x_2 t^2 + x_3 t^3 + x_4 t^4 + x_5 t^5$$

   You should formulate this as a least-squares problem, $min_x\|Ax - b\|^2$, and solve for x. Specifically, write a Python program `fitPoly()` that reads the data file and does the following:

   (a) (5 points) Constructs the matrix A and the vector b from the data.

   (b) (5 points) Solves the normal equations using Cholesky factorization and forward and back substitution. Prints out the estimated values of x_i. (Hand in these values.) Your program should use the methods `cholesky` and `solve_triangular` in `numpy.linalg`.

(c) (5 points) Plots the fitted polynomial (in green) using 1000 equally spaced values of t between 1 and 5. Plots the 100 data points (in blue) on top of the polynomial. (Hand in the plot.)

Optical Character Recognition, Part II. In the rest of this assignment, you will write and test some Python functions for use in optical character recognition in the next assignment. The strategy is to model each character as a multivariate normal distribution. Recall that in such a distribution, the probability density of an n-dimensional vector, x, is given by

$$\frac{exp[-(x-\mu)^T\Sigma^{-1}(x-\mu)/2]}{(2\pi)^{n/2}\,|\Sigma|^{1/2}} \tag{1}$$

Here, μ is the mean of the distribution, Σ is it's covariance matrix (which is symmetric and positive definite), and $|\Sigma|$ is the determinant of Σ. In optical character recognition, each character has a different μ and a different Σ, and one problem is to determine what these are. We shall do this using techniques from Artificial Intelligence, and specifically from the subfield of Machine Learning. The idea is to write and test a Python program that learns an estimate of μ and Σ from a sample of training data.

In addition to `numpy`, you will need to import the Python modules `scipy.linalg` and `matplotliob.pyplot`. Unless specified otherwise, do *not* use any statistical methods in `scipy.stats` or `numpy.random`. Also, whenever possible, *do not use loops.* Instead, use Numpy's vector and matrix operations, which are much faster and can be executed in parallel.

Finally, vectors are usually treated as columns, as in Equation (1) and in all of the questions above. However, it is sometimes convenient to treat them as row vectors, especially when storing them in a matrix, as in the questions below.

9. (24 points total) Suppose $x_1, x_2, ...x_m$ are vectors. Their mean is the vector $\hat{\mu} = (\hat{\mu}_1, ..., \hat{\mu}_n)$, where $\hat{\mu}_j = \sum_{i=1}^m x_{ij}/m$ and x_{ij} is the j^{th} component of vector x_i. The covariance matrix of the vectors is the matrix $\hat{\Sigma}$, where

$$\hat{\Sigma}_{kj} = \sum_{i=1}^{m}(x_{ik}-\hat{\mu}_k)(x_{ij}-\hat{\mu}_j)/m$$

We shall store the vectors in a $m \times n$ matrix, X, where the i^{th} row of the matrix is the vector x_i. (Note that x_i and $\hat{\mu}$ are treated in this question as row vectors.)

(a) (7 points) Prove that $\hat{\Sigma} = X_c^T X_c/m$, where X_c is a $m \times n$ matrix whose i^{th} row is the vector $x_i - \hat{\mu}$. Also prove that $\hat{\Sigma}$ is symmetric and positive semi-definite (This is the same as positive definite with $>$ replaced by $\geq$). (In standard parlance, X is called the data matrix, and X_c is called the centered data matrix, since the row vectors of X_c have a mean of 0.)

(b) (4 points) Write a Python function `fitNormal(X)` that fits a multivariate normal distribution to a data set stored in matrix `X`. The function should compute and return $\hat{\mu}$ and $\hat{\Sigma}$ as described above. Your program should use matrix multiplication and the method `numpy.sum`. Do *not* use the methods `numpy.mean` or `numpy.cov`.

(c) (5 points) Write a Python program `MVNinv(X,mu,Sigma)` that computes probabilities for a multivariate normal distribution with mean `mu` and covariance matrix `Sigma`. Specifically, the function should evaluate expression (1) for each row vector in matrix `X` and should return the probabilities as a vector. You should use the methods `det` and `inv` in `scipy.linalg` to compute the determinant and inverse of `Sigma`.

(d) (8 points) Write a Python program `MVNchol(X,mu,Sigma)` that does the same thing as `MVNinv` but by using Cholesky factorization and substitution instead of matrix inversion. You should use the methods `cholesky` and `solve_triangular` in `scipy.linalg`. For full marks, you should do substitution only once (i.e., use `solve_triangular` only once for each row of `X`). Explain your solution.

10. (9 points total) In this question, you will generate random 2-dimensional data sets to test your programs from Question 9. The data will come from a multivariate normal distribution with the following mean and covariance matrix:

$$\mu = (2, 3) \qquad\qquad \Sigma = \begin{pmatrix} 4 & 3 \\ 3 & 4 \end{pmatrix} \tag{2}$$

You can use the method `multivariate_normal` in `numpy.random` to generate the data sets. Your Python code for all the problems below should be written as a single program called `testOCR()`.

(a) (3 points) To illustrate the data, generate 1,000 random data points from the distribution. Use the function `plot` in `matplotlib.pyplot` to generate a scatter plot of the data. It should look similar to Figure 2 below but with many more points. Hand in the scatter plot.

(b) (3 points) Generate four random data sets containing 10, 100, 10,000 and 1,000,000 points, respectively. Use `fitNormal` to fit a multivariate normal distribution to each data set. Print out and hand in the estimated mean and covariance matrix for each data set. If you have done everything correctly, then as the data set gets larger, the estimated values, $\hat{\mu}$ and $\hat{\Sigma}$, should get increasingly close to the true values, μ and Σ, given in Equation (2). The estimate for the smallest data sample (10 data points) will probably not be very accurate.

(c) (3 points) If you have written the functions `MVNinv` and `MVNchol` correctly, then they should produce almost exactly the same answers. Test this by running them on a random data set of 100 points, using the mean and covariance matrix in Equation (2) as input to `MVNinv` and `MVNchol`. If p_1 and p_2 are the vectors of probabilities returned by `MVNinv` and `MVNchol`, respectively, then print out and hand in the 2-norm of $p_1 - p_2$. (Use the numpy function for computing vector norms.) You should get a very small number.

University of Toronto Mississauga
CSC 338 - Numerical Methods

Cover sheet for Assignment 2

Complete this page and hand it in with your assignment.

Name: Sayantan Chattopadhyay

(Underline your last name)

Student number: 1000818145

I declare that the solutions to Assignment 1 that I have handed in are solely my own work, and they are in accordance with the University of Toronto Code of Behavior on Academic Matters.

Signature: [illegible]

Figure 1: Data for curve fitting

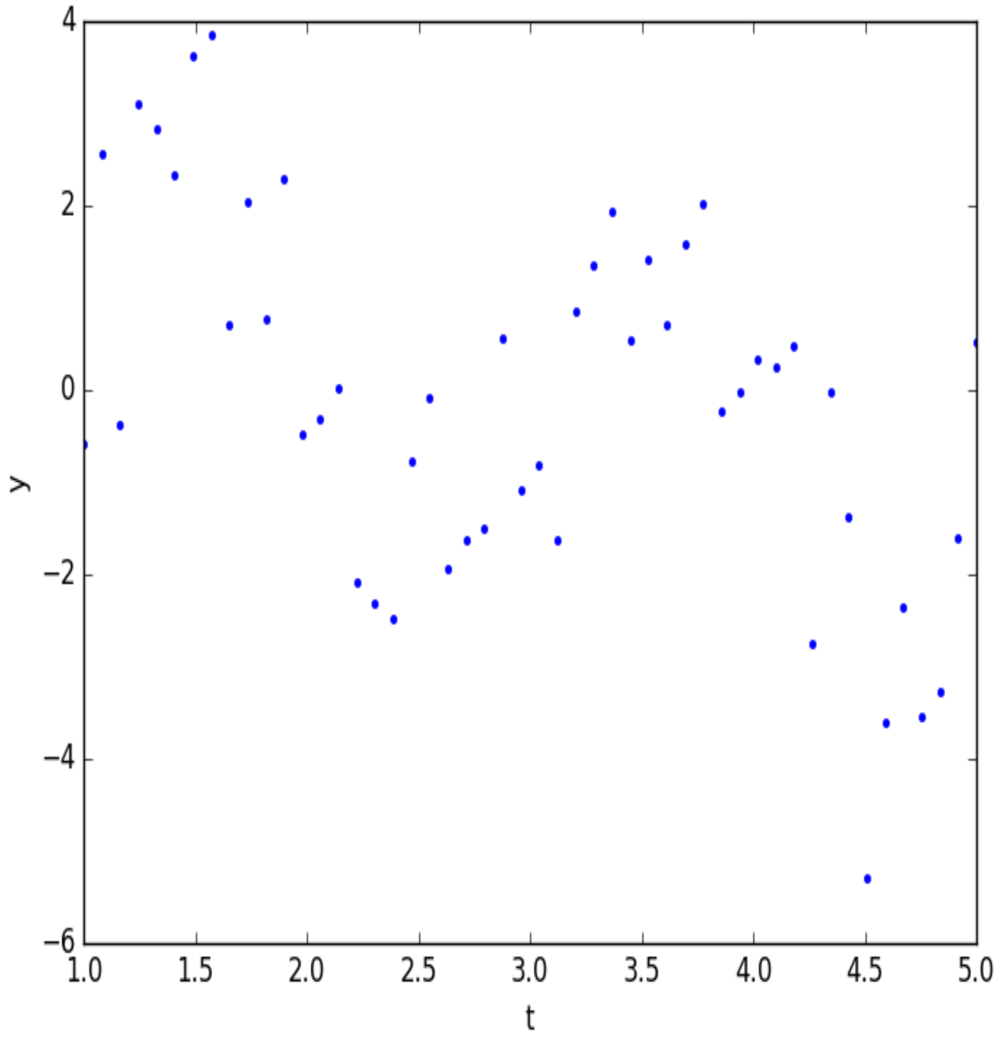
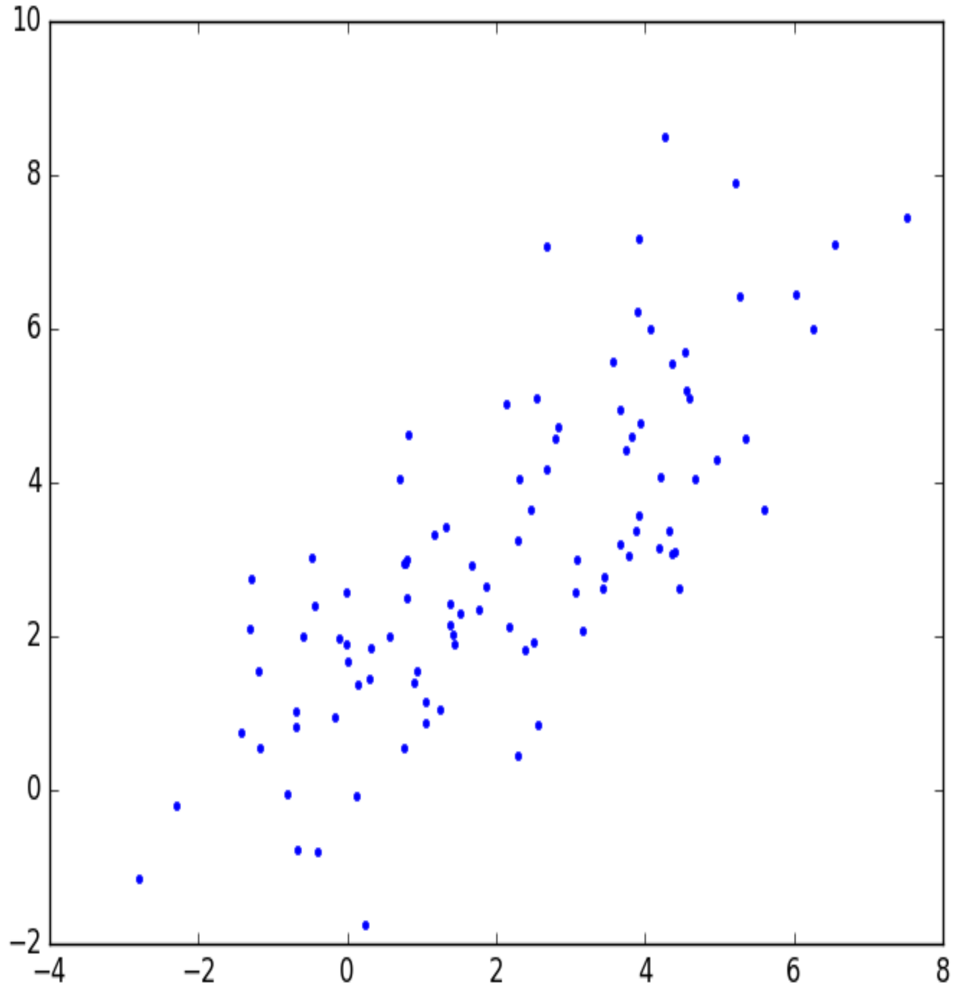


Figure 2: Sample of multivariate normal data


10
8
6
4
2
0
−2
−4
−2
0
2
4
6
8