MATH 728D: Machine Learning
Project 1: Regression with Linear Least Squares

This project will consider aspects of the *linear regression problem.* Such problems start with a set of m data pairs (t_i, y_i), and propose a model function $y(t)$ which depends linearly on n parameters x:

$$y(t) = x_1\,\phi_1(t) + x_2\,\phi_2(t) + ... + x_n\,\phi_n(t)$$

where

$$\Phi = \{\phi_1(t), \phi_2(t), ..., \phi_n(t)\}$$

is our set of *basis functions.* To define the model, we seek values for the unknown x. If we determine these values as the minimizers of the ℓ_2 norm of the residual, we are implementing a linear least squares algorithm, as we have discussed in class.

A polynomial model chooses the n basis functions $\{1, t, t^2, ..., t^{n-1}\}$, A trigonometric model the first n terms in the sequence $\{1, \sin(t), \cos(t), \sin(2t), ...\}$. Other options use exponentials, or a mix of polynomials and trigonometric functions.

For a given model, the *residual* vector r is defined by

$$r_i = y(t_i) - y_i$$

The error function that we will report is the *root-mean-square* (RMS) norm of the residual:

$$||r|| = \sqrt{\frac{1}{m}\sum_{i=1}^{m} r_i^2} = \sqrt{\frac{1}{m}\sum_{i=1}^{m} (y(t_i) - y_i)^2}$$

which can be evaluated in MATLAB by

```
rnorm = rms ( r );
```

or in Python by:

```
rnorm = np.sqrt ( ( r**2 ).mean() )
```

We use the RMS norm instead of the usual ℓ_2 norm simply because it is scaled to have a more uniform behavior as the vector size varies. When we refer to $||r||$, we will mean the RMS of the vector r.

The following datasets will be used, and will be referred to in the text by their 1-letter code abbreviation:

Code	file	m
F	filip.txt	82
M	mexico.txt	13
R	rising.txt	25
S	sinus.m / sinus.py	any

Each dataset is a text file, containing m lines, and line i contains the single pair of values t_i, y_i.

In Matlab, we can easily read dataset **F** by a command like:

```
data = load ( 'filip.txt' );
```

or in Python by:

```
data = np.loadtxt ( 'filip.txt' )
```

which in this case would create a 2×82 array.

The **S** dataset is unusual, because it is generated by calling a function which is given as input the value m, which can take any value. In Matlab, we create an **S** dataset of size 10 by a command like:

```
data = sinus ( 10 );
```

or in Python by:

```
from sinus import sinus
data = sinus ( 10 )
```

which creates a 2×10 array.

1 Create and Evaluate a Model

We wish to create a model $y(t)$ for the data in the **M** dataset that uses $n = 3$ polynomial basis functions.

At each data point, we would like the model to match the data. This can be expressed by a system of m equations involving n unknown coefficients x. These equations can be written compactly as an $m \times n$ linear system $A * x = b$. We know three methods for seeking an estimator x which minimizes the RMS error norm:

1. create and solve the normal equations `A' A x = A' b`;
2. factor `[Q,R] = qr(A)`, and solve `R x = Q' b`;
3. use the SVD pseudoinverse: `x = pinv ( A ) * b`;

Implement each method and fill in a table like the following:

Table 1:				
Method	x_1	x_2	x_3	$\|\|r\|\|$
1	.....	.....	.....	.....
2	.....	.....	.....	.....
3	.....	.....	.....	.....

In Matlab, solve $A * x = b$ by `x = A \ b`. In python, solve with `x=numpy.linalg.solve(A,b)`; get the QR factorization from `numpy.linalg.qr(a)` and the pseudoinverse from `numpy.linalg.pinv(a)`.

2 Vary the Basis Function Set

We wish to create a simple model of the data in the **R** dataset, using just $n = 3$ basis functions.

Using one of the methods discussed above (normal equations, QR, or SVD), set up the modeling problem and evaluate the RMS residual $||r||$ using each of the following basis function sets:

1. $\Phi_1 = \{1, t, t^2\}$;
2. $\Phi_2 = \{1, t, t^3\}$;
3. $\Phi_3 = \{1, t, \sin(t)\}$;

Table 2:	
Φ	$\|\|r\|\|$
$\{1, t, t^2\}$	.....
$\{1, t, t^3\}$	.....
$\{1, t, \sin(t)\}$	.....

3 Increase the Model Order

We wish to create a model $y(t)$ for the data in the **F** dataset, using polynomial basis functions. We haven't decided on the number n of such functions; as we increase n, we expect the RMS residual $||r||$ to decrease.

Use one of the three methods discussed above to set up the modeling problem for various values of n. You might find it best, when programming, to create a function that does this for a specific value of n, returning the value of the RMS residual $||r||$ as output. Then call that function in a loop to get your table.

Implement each method and fill in the table:

Table 3:	
n	$\|\|r\|\|_2$
1	.....
2	.....
3	.....
4	.....
5	.....
6	.....
7	.....
8	.....
9	.....
10	.....
11	.....
12	.....

4 Can a Model Predict Data It Hasn't Seen?

For this exercise, we need to create *two* samples of the **S** dataset of sizes $m1 = 10$ and $m2 = 20$. See the introduction for details on how to create these sets, which we can identify as **S1** and **S2**.

We are going to construct several least squares approximation models $y(t)$, of increasing order n, for the dataset **S1** in the usual way. We will evaluate each model function by computing the RMS residual $||r1||$, which measures the average error between $y(t)$ and the values $y1$ from the **S1** dataset. But we will **also** compute $||r2||$, the RMS residual which measure the average error between $y(t)$ and the values $y2$ from the **S2** dataset, which were not used in constructing $y(t)$.

As n increases, we expect $||r_1||$ to decrease. But will $y(t)$ do a reasonable job of predicting the values in **S2**, as indicated by the values of $||r_2||$? How does this change as n increases?

Table 4:		
n	$\|\|r_1\|\|$	$\|\|r_2\|\|$
1	.....	.....
2	.....	.....
3	.....	.....
4	.....	.....
5	.....	.....
6	.....	.....
7	.....	.....
8	.....	.....
9	.....	.....
10	.....	.....

It's relatively easy to evaluate $||r_1||$ because this is simply $||Ax - b||$. However, to compute $||r_2||$, you need to form and evaluate the polynomial $y(t)$ described by the coefficients x and compare it to the data in **S2**. You can use a single command to evaluate a polynomial at multiple points. In MATLAB, this is called `polyval()` and in python, `numpy.polyval()`. Be careful, since both functions assume that the coefficients are ordered so that the first value multiplies the highest power of t, not the lowest.

5 Increase the Data Count

In the previous example, we were given a dataset of fixed size m and used increasing values of n for our approximation order. As we did so, the approximation of the known data improved, but the approximating function became unreliable for "predicting" the values of the testing data we had kept aside.

Now we look at what happens if we hold n fixed, and consider using more and more data values m to inform our approximation.

Fix the model order $n = 10$. Using the dataset **S**, generate "training" samples of size $m1 = 10, 20, 40, 80, 160$. Also generate a single, fixed, "testing" dataset of size $m2 = 25$. Now for each training dataset, construct a polynomial model function, evaluate the RMS residual $||r_1||$, but also evaluate the model on the testing dataset, creating the RMS residual $||r_2||$. Report what happens to both residuals as $m1$ increases.

Table 5:

m1	$\|\|r_1\|\|$	$\|\|r_2\|\|$
10	.....	.....
20	.....	.....
40	.....	.....
80	.....	.....
160	.....	.....

6 What to Turn in:

The project should be completed and submitted by the beginning of class on Tuesday, March 19th. You should submit a report, roughly 3-4 pages in length, stored as a single PDF file, that includes:

1. **Team information:** The names of the team members. Teams can involve from 1 to 4 members. Each team member should participate in the project, and should be able to explain the process involved in getting the results of the exercises.
2. **Problem Description:** You should describe in your own words the problem of regression with linear least squares. This should explain the idea of a dataset, the construction of a model, the choices for the order of the model and the basis functions used, the measurement of error in the approximation, and ways of judging whether the order of an approximation is low, about right, or too high.
3. **Programming Description:** Explain in general terms how you computed your results. What language did you use; how did you set up and solve the linear system; how did you evaluate the model and residuals; did you try to plot the data or the models?
4. **Your tables:** You should turn in neatly formatted versions of Tables 1, 2, 3, 4 and 5. Three or four digits of accuracy should be enough for the table values.
5. **Comments on Results:** You should make at least a short comment about each exercise. Things to discuss include the behavior of the RMS error with increasing order; cases in which the approximation process seems to break down; observations about how to judge a good approximation.

You **do not** need to turn in your programs. However you should keep copies of them available, as they may be requested for review if there is an issue with your results.