CSCC11 Assignment 1: Least-Squares Regression

Written Part (3%): due Friday, September 18th, 11am

Question 1

Consider the following equations corresponding to measurements of two unknown quantities (x_1, x_2)

$$\begin{aligned} 5x_1 - 2x_2 &= 15 \\ -2x_1 + 5x_2 &= 10 \\ 3x_1 - x_2 &= 8 \end{aligned}$$

where x_1 and x_2 are scalars. Write these equations as a single matrix-vector equation and derive its least-squares solution. Compute the numerical solution using Matlab (i.e., find the values for x_1 and x_2 which best solve the equation in the least-squares sense) and provide it along with the Matlab commands you used to compute it.

What to submit Submit a paper printout of the Matlab code you used to setup the system and compute its numerical solution.

Question 2

In a vector space (e.g., $\mathbb{R}^n$), distance is traditionally computed using Euclidean distance

$$d^2(x, y) = (x - y)^T (x - y)$$

where $x \in \mathbb{R}^n$ and $y \in \mathbb{R}^n$. This measure of distance is circularly symmetric, meaning that all directions are considered equally important. However, sometimes certain dimensions (or directions) are considered "more important" than others. In these cases it makes sense to use a slightly different measure of distance, called the Mahalanobis distance. It is defined as

$$d_M^2(x, y) = (x - y)^T M (x - y)$$

where $M \in \mathbb{R}^{n \times n}$ is a square, invertible and positive definite matrix. Obviously, $d_M^2(x, y) = d^2(x, y)$ if M is the identity, meaning that the Mahalanobis distance is a generalization of the Euclidean distance metric. Now, consider the problem of computing the midpoint a set of points $\mathcal{D} = \{x_i\}_{i=1}^N$. Mathematically, the midpoint of the set is defined as the point $\bar{x}$ which minimizes $L(\bar{x}) = \sum_{i=1}^N d_M^2(x_i, \bar{x})$, i.e.,

$$\bar{x} = \arg\min_{\bar{x}} \sum_{i=1}^{N} d_M^2(x_i, \bar{x})$$

You are to derive an equation for the value of $\bar{x}$. To do this you must first compute the gradient of L with respect to $\bar{x}$, set the gradient equal to zero and solve for $\bar{x}$.

What to submit You should submit a written derivation of the gradient and the minimizer of $L(\bar{x})$. Clearly write out all steps and note any assumptions you make along the way.

Programming Part (3%): due Friday September 25th, 11am

Your goal in this assignment is to implement least-squares (LS) estimators for polynomial regression models. To get started, download the code and training data that is available on the course website. The code shows a couple of ways in which one might compute the least-squares estimate of a linear model for a function that maps a scalar input to a scalar output. The last section of the demo shows the preferred method, with matrix-vector operations and the pseudo-inverse. Your first task is to read through and run this code (cell by cell, reading the comments in each cell).

Your next task is to look at the training data which is in the file `a1TrainingData.mat`. Load the training data into Matlab. You will see that the training data comprises two vectors named $\mathbf{x}$ and $\mathbf{y}$, denoting the input and output for our regression problem. We'll denote these vectors here as

$$\mathbf{x} = [x_1, x_2, ..., x_N]^T \quad , \quad \mathbf{y} = [y_1, y_2, ..., y_N]^T \; . \tag{1}$$

In total we have N input-output pairs. To get a better feeling for the training data you may wish to plot it using the Matlab command `plot(x, y, '*b')`.

Your goal is to find a model function that predicts the output for a given input. We'll restrict our attention to polynomial functions. Each polynomial is expressed as a weighted sum of monomials, where the largest monomial degree is denoted K. I.E., K is the order of the polynomial. For this assignment you can assume that $1 \leq K \leq 12$. In summary,

$$y = f(x) = \sum_{k=0}^{K} w_k x^k \tag{2}$$

The goal is to select the model (i.e., find a reasonable value for K), and to find the LS estimate the parameters $\mathbf{w}_K = [w_0, w_1, ..., w_K]^T$ that provide the best predictions on future test data (which of course we don't know a priori). As a proxy for future data, we'll minimize the error in our predictions on the training data (using a squared loss function).

Write Regression Code

Write a function called `polynomialRegression` that takes three input arguments, namely,

- `K`, the degree of the polynomial,
- `x`, a vector of training inputs, and
- `y`, a vector of corresponding training outputs.

Your function should return a vector `w` of parameters (comprising the regression weights and the offset). Your code should use matrix-vector operations with the backslash operator to compute the solution.

Write a function called `evalPolynomial` that takes two input arguments, namely, a vector of input values `x`, and a vector of polynomial parameters `w` (assume that the parameters are ordered such that the jth element is the linear coefficient associated with the jth-order monomial, i.e., x^j). This function should output the predictions of the model for each value in the input vector.

Skeleton versions of both of these functions are provided in the A1 starter kit.

Fit Models

Write a script that uses these two functions to fit models to the training data. For each K between 1 amd 12 (i.e., the maximum degree of the polynomial)

1. Find the LS estimate for the polynomial coefficients.
2. Given the estimated parameters, compute the total amount of residual error in the training data.
3. Plot the fitted model for values of `x = [-2.1:0.1:2.1]`.

Finally, given the residual errors on the training data, for all models, plot the total error as a function of the polynomial degree K.

Test Models

Now it's time to consider some new, test data. This is provided in the file `a1TestData.mat and` which contains two arrays called `xTest` and `yTest`. With this data you can test to see how well your models generalize to new inputs which weren't used during training. On this new data, evaluate your models and compute the error in the model predictions (on the test dataset). Then, plot this error as a function of the degree of the polynomial models.

Analyze the Results

First, consider the plot of the error on the training data as a function of degree K. Explain which models seem to be "good" models and which ones seem to be bad. If you had to pick a value of K just from this plot, what would it be? Explain your reasoning in one or two sentences.

Now, consider the plot of error on the testing data as a function of model degree K. What are some qualitative differences between the behaviour you see in the performance of the models on the training data and on the test data? Can you say which models (i.e., which values of K) appear to overfit the training data? Can you guess the degree of the polynomial that was used to generate the training and test data? Explain the basis for your opinion.

What to submit Submit an electronic version of your solution by creating a tar file comprising the scripts and Matlab function files. This tar file should be named `A1.tar` and submitted it according to instructions on the course web site by the due date.

You also need to submit a printed version of this part of your assignment. It should include:

- a printed version of your Matlab scripts and function files,
- printouts of the two error plots (one with training data and one with test data) with their axes clearly labeled and a title indicating which is which,
- printout of the fit models for each value of K, and
- written answers to the questions described above.

Your paper copy should include a coversheet with the course number, assignment number, your name, student number and mathlab username.