Linear Regression

UCLA Math 156

Spring 2016

1 Overview

Linear regression is a fundamental tool for learning the relationship between a collection of independent variables and corresponding dependent variables. For simplicity, in this lab we consider modeling a function $\mathbb{R} \to \mathbb{R}$ by finding the least squared error fit to the data. For a collection of observations (x_n, t_n) and basis functions $\phi_j(x)$, the penalized least squares estimate is a linear combination $\sum_j w_j \phi_j(x)$ which minimizes

$$\min_w \sum_n \frac{1}{2} \left(\sum_j w_j \phi_j(x_n) - t_n \right)^2 + \frac{1}{2\mu^2} \sum_j w_j^2.$$

One can obtain this model from a probabilistic perspective by assuming a Gaussian noise model with a zero-mean Gaussian prior on the parameters. The nature of this problem makes it simple to solve for the minimum (also known as the maximum a posteriori, MAP, estimate) using standard linear algebra methods.

In this lab you are provided with code to work with two types of basis functions ϕ_j: Gaussian curves and hat functions. After writing a function which performs this least squares fit, you will apply the model to a small synthetic data set and study the effect of the prior (a.k.a. penalty) hyperparameter μ.

2 Provided Resources

- `lsefit.m` - Function to be completed which calculates a maximum likelihood fit to observations $(x_n, t_n) \in \mathbb{R}^2$ in a least squared error sense, using a provided collection of basis functions and the specified penalty μ.
- `simple.mat` - A small synthetic collection of observations $(x_n, t_n) \in \mathbb{R}^2$ with $x_n \in [0, 2\pi]$ sampled from a simple function (sin) with a small amount of noise (locations x in `x` and target values t in `y`).
- `test.mat` - Another collection of observations of the same function as in `simple.mat`, to be used for model selection. The contained variables are named `xt` and `yt`.
- `gauss_basis.m` - Function which generates a basis of Gaussian curve functions over the specified interval. The output is parameters for these basis functions, to be used by `func_gauss.m`. Each column of the output contains the mean and standard deviation for a Gaussian function in one dimension.
- `hat_basis.m` - Function which generates a basis of hat functions over the specified interval. The output is parameters for these basis functions, to be used by `func_hat.m`. Each columns of the output contains the start and end of the support interval for a hat function.
- `func_gauss.m` - Evaluates a Gaussian curve in one dimension with the given parameters, at the specified locations.
- `func_hat.m` - Evaluates a hat function in one dimension with the given parameters, at the specified locations.

3 Guide

1. Complete the function `function w = lsefit(x, t, params, phi, mu)` in `lsefit.m`. This function should find a MAP estimate (regularized least squares) for the coefficients w_j in the model mentioned previously. To do this, let the matrix Φ be such that $\Phi_{n,j} = \phi_j(x_n)$. Differentiating the energy and setting the gradient equal to zero, we arrive at (you should be able to derive this):

$$w = \left(\Phi^T\Phi + \frac{1}{\mu^2}I\right)^{-1}\left(\Phi^T\mathbf{t}\right).$$

   You will need to construct the design matrix Φ using the basis function and parameters passed as an argument to `lsefit.m`; use function handles to pass a function as argument `phi`, e.g., call `w = lsefit( x, y, params, @func_gauss, mu );` to use the Gaussian basis functions with parameters constructed using `gauss_basis.m`. Then, within `lsefit`, the values of the j^{th} basis function at the locations x are calculated using `phi(x, params(:,j))`.

2. (4) Train the model on the data in `simple.mat` using $M = 10$ hat functions and $\mu = 10^5$. Plot and turn in the learned model (the function fit to the data) on the interval $[0, 2\pi]$.

3. Do the same for other values of the hyperparameter such as $\mu = 10$ and $\mu = 1$.

4. (2) What, if anything, is the hyperparameter controlling? How does it impact the solution to the problem?

5. (2) Train the model with $M = 10$ Gaussian basis functions and $\mu = 10^5$, then plot and turn in the learned model evaluated (the function fit to the data) on the interval $[0, 2\pi]$.

6. As before, explore other values of the hyperparameter such as $\mu = 10$ and $\mu = 1$.

7. (2) What, if anything, is the hyperparameter controlling in this case? How does the dependence of the solution on the hyperparameter differ for this basis when compared to the hat basis?

8. (4) Fit the model with values of μ in the range $[1, 100]$ and using a Gaussian basis of ten elements on the data in `simple.mat`. For each model, calculate the squared error for the observations in `test.mat` (therefore testing the model). Generate and turn in a plot with μ on the x-axis and the total squared model error on the test data along the y-axis.

9. (2) What value of μ, when trained on the data in `simple.mat`, performs best on the data in `test.mat`? How do you know? Explain the shape of the plot you generated in the previous step.

10. (4) Repeat the process now fixing μ to be the optimal value you found and varying the number of basis elements from 1 to 100. Generate and turn in a plot with the number of basis elements on the x-axis and the error for the test data on the y-axis.