Introduction to Machine Learning

Homework 1

Due date: 09/17/2019

1 Problem 1 (10 points)

Cross-validation for Polynomial Fitting: Consider the problem of fitting a polynomial function. Assume we wish to find a one dimensional function that takes a scalar input and outputs a scalar $f : \mathbb{R} \rightarrow \mathbb{R}$. The function has the form

$$f(x;\boldsymbol{\theta}) \quad = \quad \theta_0 + \theta_1 x + \theta_2 x^2 + \ldots + \theta_d x^d$$

where d is the degree of the polynomial. Develop code that finds the θ which minimizes the risk

$$R_{emp}(\boldsymbol{\theta}) \quad = \quad \frac{1}{N}\sum_{i=1}^{N}\frac{1}{2}(y_i - f(x;\boldsymbol{\theta}))^2$$

on a data-set. To help you get started, download the Matlab code in "polyreg.m" (on the tutorial web page) to do polynomial curve fitting. Use your code on the dataset "problem1.mat". This should include a matrix x, corresponding to the scalar features $\{x_1, \ldots, x_N\}$, and a matrix y, corresponding to the scalar labels $\{y_1, \ldots, y_N\}$. Fit a polynomial model to this data for various choices for d, the degree of the polynomial.

Which value(s) of d seems somewhat more reasonable? Please justify your answer using some empirical measure.

It is easy to overfit the data when using polynomial regression. As a result, use cross-validation by randomly splitting the data-set into two halves to select the complexity of the model (in this case, the degree of the polynomial). Include a plot showing the training and testing risk across various choices of d, and plot your $f(x;\boldsymbol{\theta})$ overlaid on the data for the best choice of d according to cross-validation.

2 Problem 2 (10 points)

Regularized risk minimization: Modify the Matlab code for "polyreg.m" such that it learns a multivariate regression function $f : \mathbb{R}^{100} \rightarrow \mathbb{R}$, where the basis functions are of the form

$$f(x;\boldsymbol{\theta}) \quad = \quad \sum_{i=1}^{k} \theta_i x_i$$

The data-set is available in "problem2.mat". As before, the x variable contains $\{x_1, \ldots, x_N\}$ and the y variable contains their scalar labels $\{y_1, \ldots, y_N\}$.

Use an l_2 loss function to penalize the complexity of the model, e.g. minimize the risk

$$R_{\text{reg}}(\theta) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} \frac{1}{2}(y_i - f(x;\boldsymbol{\theta}))^2 + \frac{\lambda}{2N}\|\boldsymbol{\theta}\|^2$$

Use two-fold cross validation (as in Problem 1) to find the best value for λ. Include a plot showing training and testing risk across various choices of λ. A reasonable range for this data set would be from $\lambda = 0$ to $\lambda = 1000$. Also, mark the λ which minimizes the testing error on the data set.

What do you notice about the training and testing error?

3 Problem 3 (10 points)

Logistic Squashing Function. The logistic squashing function is given by $g(z) = 1/(1 + \exp(-z))$. Show that it satisfies the property $g(-z) = 1 - g(z)$. Also show that its inverse is given by $g^{-1}(y) = \ln(y/(1-y)$.

4 Problem 4 (20 points)

Logistic Regression: Implement a linear logistic regression algorithm for binary classification in Matlab using gradient descent. Your code should accept a dataset $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ where $\mathbf{x}_i \in \mathbb{R}^d$ and $y_i \in \{0, 1\}$ and find a parameter vector $\boldsymbol{\theta} \in \mathbb{R}^d$ for the classification function

$$f(\mathbf{x};\boldsymbol{\theta}) \quad = \quad \left(1 + \exp(-\boldsymbol{\theta}^\top \mathbf{x})\right)^{-1}$$

which minimizes the empirical risk with logistic loss

$$R_{emp}(\boldsymbol{\theta}) \quad = \quad \frac{1}{N} \sum_{i=1}^{N} (y_i - 1)\log(1 - f(\mathbf{x}_i;\boldsymbol{\theta})) - y_i \log(f(\mathbf{x}_i;\boldsymbol{\theta})).$$

Since you are using gradient descent, you will have to specify the step size η and the tolerance ϵ. Pick reasonable values for η and ϵ to then use your code to learn a classification function for the dataset in "dataset4.mat". Type "load dataset4" and you will have the variables X (input vectors) and Y (binary labels) in your Matlab environment which contain the dataset.

Show any derivations you need to make for this algorithm.

Use the whole data set as training. Show with figures the resulting linear decision boundary on the 2D X data. Show the binary classification error and the empirical risk you obtained throughout the run from random initialization until convergence. Note the number of iterations needed for your choice of η and ϵ.