COMP 652 - ECSE 608: Machine Learning - Assignment 1

Posted Thursday, September 14, 2017
Due Thursday, October 5, 2017

You should submit an archive of your code, as well as a pdf file with your answers (either typed or scanned), uploaded to MyCourses. If you cannot access MyCourses, email the assignment directly to Guillaume by the deadline (11:59pm EST on the day the assignment is due).

1. [45 points] **Regression**

   For this exercise, you will experiment with regression, regularization and cross-validation. You are provided with a data set in files `hw1-x.csv` (inputs) and `hw1-y.csv` (targets). You are allowed to use any programming language of your choice (Python, Matlab, R, etc). You can also use use any toolbox/package (e.g. `scickit-learn`) as long as you understand what is going on behind the scene (for example does the `fit_linear_model()` method you use fit a bias term or does it assume that the data is centered?).

   (a) [5 points] Load the data into memory. Make an appropriate $\mathbf{X}$ matrix and $\mathbf{y}$ vector. Split the data at random into one set $(\mathbf{X}_{train}, \mathbf{y}_{train})$ containing 80% of the instances, which will be used for training + validation, and a testing set $(\mathbf{X}_{test}, \mathbf{y}_{test})$ (containing remaining instances). Describe here any preprocessing of the data that you performed for this exercise (did you handle the bias term in linear regression directly or did you add a 1 to each input? Did you center the data? Why and how?, etc.).

   (b) [5 points] Run linear regression on the data using L_2 regularization, varying the regularization parameter $\lambda \in \{0, 0.1, 1, 10, \cdots, 10^5\}$. Plot on one graph the root-mean-square error (RMSE) for the training data and the testing data, as a function of λ (you should use a log scale for λ). Plot on another graph the L_2 norm of the weight vector you obtain. Plot on the third graph the actual values of the weights obtained (one curve per weight). Explain briefly what you see.

   (c) [5 points] Perform five-fold cross-validation on the training data to determine the best value of the regularization parameter λ (report the training and validation errors for each fold and briefly explain how you chose the best value of λ). Compare the best value of λ obtained to your results in question 1b and briefly comment.

   (d) [5 points] Suppose that the training data was sorted in increasing value of the target variable y, and you simply partitioned it by splitting it in k folds (without shuffling the data first). Explain what would happen if you tried to perform cross-validation with these folds.

   (e) [5 points] Re-format the data in the following way: take each of the input variables, and feed it through a set of Gaussian basis functions, defined as follows. For each variable, use 5 univariate basis functions with means evenly spaced between -1 and 1 and variance σ^2. You will experiment with σ^2 values of 0.1, 0.5, 1 and 5.

   (f) [5 points] Using no regularization and doing regression with this new set of basis functions, plot the training and testing error as a function of σ^2 (when using only basis functions of a given σ^2). Add constant lines showing the training and testing error you had obtained in question 1b. Explain how σ^2 influences overfitting and the bias-variance trade-off.

(g) [10 points] Suppose that instead of using a fixed set of evenly-spaced basis functions, you would like to adapt the placement of these functions. Derive a learning algorithm that computes both the placement of the basis function, μ_i, and the weight vector $\mathbf{w}$ from data (assuming that the width σ^2 is fixed). You should still allow for L_2 regularization of the weight vector. Note that your algorithm will need to be iterative.

(h) [5 points] Does your algorithm converge? If so, does it obtain a locally or globally optimal solution? Explain your answer.

2. [10 points] **Maximum likelihood**

Suppose that you are given a training set $\{(\mathbf{x}_i, y_i)\}_{i=1}^m \subset \mathbb{R}^n \times \mathbb{R}$ of m i.i.d. examples. In class we discussed that minimizing the mean squared error corresponds to an assumption that the labels of the data came from some target hypothesis $h_{\mathbf{w}}$, but then were observed after being perturbed by Gaussian noise, with the noise variables drawn i.i.d. from the same distribution.

Suppose now that the standard deviation with which we can observe the label of example i is σ_i. More precisely:

$$y_i = h_{\mathbf{w}}(\mathbf{x}_i) + \varepsilon_i \text{ with } \varepsilon_i \sim \mathcal{N}(0, \sigma_i)$$

Derive the maximum likelihood estimate of $\mathbf{w}$ in this case.

3. [10 points] **Bayesian analysis and biased coin**

Suppose you flip a coin with unknown bias θ (i.e. $P(x = H) = \theta$) three times and observe the outcome HHH. What is the maximum likelihood estimator for θ? Do you think this is a good estimator? Would you want to make predictions using this estimator?

Consider a Bayesian analysis of θ with a beta prior $p(\theta|\alpha, \beta) = \mathcal{B}(\theta; \alpha, \beta)$. What are the posterior mean and posterior mode of θ? Consider $(\alpha, \beta) = (50, 50)$. Plot the posterior density in this case. Is the maximum likelihood estimator a good summary of the distribution?

4. [25 points] **Multivariate Regression**

(a) [5 points] We consider the problem of learning a vector-valued function $f : \mathbb{R}^d \to \mathbb{R}^p$ from input-output training data $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^m$ where each $\mathbf{x}_i$ is a d-dimensional vector and each $\mathbf{y}_i$ is a p-dimensional vector. We choose our hypothesis class to be the set of linear functions from $\mathbb{R}^d$ to $\mathbb{R}^p$, that is function satisfying $f(\mathbf{x}) = \mathbf{W}^\top \mathbf{x}$ for some $d \times p$ regression matrix $\mathbf{W}$, and we want to minimize the squared error loss function

$$J(\mathbf{W}) = \sum_{i=1}^{m} \|\mathbf{W}^\top \mathbf{x}_i - \mathbf{y}_i\|_2^2 \tag{1}$$

over the training data.

Give an expression of $J(\mathbf{W})$ as a function of the data matrices $\mathbf{X} \in \mathbb{R}^{m \times d}$ and $\mathbf{Y} \in \mathbb{R}^{m \times p}$, and derive a closed-form solution (as a function of $\mathbf{X}$ and $\mathbf{Y}$) for the minimization problem $\mathbf{W}^* = \arg\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W})$.

(b) [5 points] Show that solving the problem from the previous question is equivalent to independently solving p independent classical linear regression problems (one for each component of the output vector), and give an example of a multivariate regression task where performing independent regressions for each output variables is not the best thing to do.

(c) [5 points] When the output variables are dependent, independently solving each of the regression problems may be suboptimal. The *low-rank regression* (or reduced-rank regression) model captures the dependencies between the output variables by constraining the regression matrix $\mathbf{W}$ to be of low rank.

Suppose that training data is generated with the model $\mathbf{y}_i = \mathbf{W}^\top \mathbf{x}_i + \boldsymbol{\xi}_i$ where the $\mathbf{x}_i$ are independently drawn from some distribution on $\mathbb{R}^d$, $\mathbf{W} \in \mathbb{R}^{d \times p}$ is of rank R, and the $\boldsymbol{\xi}_i$ are random noise terms i.i.d. from the multivariate normal distribution $\mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$.

Using the fact that $\mathbf{W}$ is of rank R, show that there exists a $(p - R)$-dimensional subspace of $\mathbb{R}^p$ containing only noise.

(d) [10 points] Propose an algorithm to minimize the squared error loss over the training data subject to a low rank constraint[1] on the regression matrix $\mathbf{W}$:

$$\min_{\mathbf{W} \in \mathbb{R}^{d \times p}} J(\mathbf{W}) \quad \text{s.t. } \operatorname{rank}(\mathbf{W}) \leq R.$$

5. [15 points] **Kernels**

In this problem, we consider constructing new kernels by combining existing kernels. Recall that for some function $K : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}$ to be a kernel, we need to find a feature mapping ϕ such that the kernel function can be written as a dot product between vectors in the feature space:

$$K(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z}) \quad \text{for all } \mathbf{x}, \mathbf{z} \in \mathbb{R}^n.$$

Mercer's theorem gives a necessary and sufficient condition for a function K to be a kernel: its corresponding kernel matrix has to be symmetric and positive semidefinite for any set of points $\mathbf{x}_1, \cdots, \mathbf{x}_m \in \mathbb{R}^n$.

Suppose that K_1 and K_2 are kernels over $\mathbb{R}^n \times \mathbb{R}^n$. For each of the cases below, prove that K is also a kernel. You can use either Mercer's theorem, or the definition of a kernel as needed.

(a) $K(\mathbf{x}, \mathbf{z}) = aK_1(\mathbf{x}, \mathbf{z}) + bK_2(\mathbf{x}, \mathbf{z})$, where $a, b > 0$ are real numbers

(b) $K(\mathbf{x}, \mathbf{z}) = K_1(\mathbf{x}, \mathbf{z})K_2(\mathbf{x}, \mathbf{z})$

(c) $K(\mathbf{x}, \mathbf{z}) = f(\mathbf{x})f(\mathbf{z})$ where $f : \mathbb{R}^n \to \mathbb{R}$ is a real-valued function

[1] There are different ways to do that. You could for example leverage the fact that $\operatorname{rank}(\mathbf{W}) \leq R$ if and only if there exists $\mathbf{A} \in \mathbb{R}^{d \times R}$ and $\mathbf{B} \in \mathbb{R}^{R \times p}$ such that $\mathbf{W} = \mathbf{AB}$.