CS4442A & CS9542: Artificial Intelligence II – Assignment #1

Due: 23:55, February 16, 2024

Instructions

Submission

Your submission should include: 1) a .pdf file containing all the answers to the questions. For the questions requiring to plot figures, you should also include all the figures in the .pdf file. 2) source code (e.g., .py files, .m files...).
Alternatively, you can use the IPython notebook and submit a .ipynb file that includes the code and solutions together. Make sure that your .ipynb file has displayed all the required figures and results.

Implementation

For Q2 and Q3 (linear and polynomial regression, regression with regularization), you are NOT allowed to use any existing toolbox in Python/Matlab for regression (e.g., fitlm function in Matlab) – this is the simplest machine learning model, and you should implement the regression algorithm by yourself to make sure that you have fully understood it (see Lectures 3 and 4). However, you may call functions for matrix operations (e.g., numpy for matrix inverse, matrix multiplication) and cross-validation.

1 Refreshing Mathematics [25 points]

Let $w \in \mathbb{R}^n$ is an n-dimensional column vector, and $f(w) \in \mathbb{R}$ is a function of w. In Lecture 2, we have defined the *gradient* $\nabla f(w) \in \mathbb{R}^n$ and *Hessian matrix* $H \in \mathbb{R}^{n\times n}$ of f with respect to w.

(a) [5 points] Let $a, b \in \mathbb{R}^n$ be two n-dimensional vectors, and $a \circ b \in \mathbb{R}^n$ be the *element-wise* product of two vectors.[1] Let $f(w) = a^\top(w \circ b)$. Compute $\nabla f(w)$ using the definition of gradient.

(b) [5 points] Let $f(w) = \mathrm{tr}(Bww^\top A)$, where $A, B \in \mathbb{R}^{n\times n}$ are squared matrices of size $n \times n$, and $\mathrm{tr}(A)$ is the *trace* of the squared matrix A. Using the definition of gradient, compute $\nabla f(w)$.

(c) [5 points] Let $f(w) = \mathrm{tr}(Bww^\top A)$. Compute the *Hessian matrix* H of f with respect to w using the definition.

(d) [5 points] If $A = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$ and $B = \begin{bmatrix} 2 & -1 \\ -1 & 3 \end{bmatrix}$. Is $f(w)$ a convex function? (Hint: you may use Python/Matlab/R for this question.)

(e) [5 points] In Lecture 5, we have define the sigmoid function: $\sigma(a) = \frac{1}{1+e^{-a}}$. Let $f(w) = \log(\sigma(w^\top x))$, where log is the natural logarithmic function. Compute $\nabla f(w)$ using the definition of gradient.

2 Linear and Polynomial Regression [50 points]

For this exercise, you will implement linear and polynomial regression in any programming language of your choice (e.g., Python/Matlab/R). The training data set consists of the features `hw1xtr.dat` and their desired outputs `hw1ytr.dat`. The test data set consists of the features `hw1xte.dat` and their desired outputs `hw1yte.dat`.

(a) [5 points] **Load** the training data `hw1xtr.dat` and `hw1ytr.dat` into the memory and **plot** it on one graph. **Load** the test data `hw1xte.dat` and `hw1yte.dat` into the memory and **plot** it on another graph.

[1] See `https://en.wikipedia.org/wiki/Hadamard_product_(matrices)` for the definition and some examples. Note that a vector is a special case of a matrix.

(b) [10 points] Add a column vector of 1's to the features, then use the linear regression formula discussed in Lecture 3 to obtain a 2-dimensional weight vector; **Plot** both the linear regression line and the training data on the same graph. (8 points)

Also **report** the average error on the training set using Eq. (1), rounded to four decimal places. (2 points)

$$err = \frac{1}{m}\sum_{i=1}^{m}(w^\top x_i - y_i)^2 \tag{1}$$

(c) [5 points] **Plot** both the regression line and the test data on the same graph. (3 points)

Also **report** the average error on the test set using Eq. (1), rounded to four decimal places. (2 points)

(d) [10 points] **Implement** the 2nd-order polynomial regression by adding new features x^2 to the inputs, **repeat (b) and (c)**. (8 points)

Compare the training error and test error. Is it a better fit than linear regression? Briefly explain. (2 points)

(e) [10 points] **Implement** the 3rd-order polynomial regression by adding new features x^2, x^3 to the inputs, **repeat (b) and (c)**. (8 points)

Compare the training error and test error. Is it a better fit than linear regression and 2nd-order polynomial regression? Briefly explain. (2 points)

(f) [10 points] **Implement** the 4th-order polynomial regression by adding new features x^2, x^3, x^4 to the inputs, **repeat (b) and (c)**. (6 points)

Compare the training error and test error. Compared with the previous results, which order is the best for fitting the data? Briefly explain. (4 points)

3 Regularization and Cross-Validation [25 points]

(a) [10 points] Using the training data to **implement ℓ_2-regularized for the 4th-order polynomial regression** (page 12 of Lecture 4, note that we do not penalize the bias term w_0), vary the regularization parameter $\lambda \in \{0.01; 0.1; 1; 10; 100; 1000\}$. **Plot** the training and test error (averaged over all instances) using Eq. (1) as a function of λ (you should use a $\log_{10}$ scale for λ). (6 points)

Which λ is the best for fitting the training data? Which λ is the best for fitting the test data? **Report** the corresponding errors (rounded to four decimal places) and briefly explain. (4 points)

(b) [10 points] **Plot** the value of each weight parameter (including the bias term w_0) as a function of λ. (10 points)

(c) [5 points] **Write a procedure** that performs five-fold cross-validation on your training data (page 7 of Lecture 4). Use it to determine the best value for λ. **Plot** the average error on the validation set as a function of λ and **report** the best value for λ. Is the same as the best λ in (a)? (3 points)

For the best fit, **plot** the test data and the ℓ_2-regularized 4th-order polynomial regression line obtained. (2 points)