EECS545 - Homework 1

January 17th, 2018

Instructions

- This homework is Due Monday the 29th at 5pm. No late submissions will be accepted.
- You will submit a write-up and your code for this homework.
- You are expected to use Python for the programming questions in this homework. Use of Python 3 is recommended.
- Submit your plots and answers as a pdf on Canvas under filename `uniqname_hw1.pdf`.
- Submit all your python code in a compressed zip file named `uniqname_hw1_code.zip`. Your zip file should contain `prob1.py,prob2.py,prob3.py`.
- We expect you to implement models and algorithms using numpy functions. You will not need any specific machine learning libraries for this homework.

1. (**25 points**) In this problem, you will implement and train linear regression models using different methods.

   We will be using the Boston housing dataset for this problem. The data is composed of 506 instances, each consisting 14 features relevant to house pricing and the price of a house. More details about the dataset can be found at `https://www.kaggle.com/c/boston-housing`.

   We will build a linear regression model to predict house prices given the features. Our objective function will be the mean-squared error (MSE) loss function. Use the starter code provided for this problem. Make sure your submitted python script has the same name as the starter code file `prob1.py`.

   (a) Normalize the features so that each feature has zero mean and unit standard deviation. (If you encounter division by zero for a feature, only subtract the mean for that feature).

   (b) Use **stochastic gradient descent (SGD)** to train your regression model. Implement the weight update rule and train the model until convergence (i.e., until the error does not change much. Choose an appropriate learning rate and number of epochs. Eg: Learning rate = 5e-4, Number of epochs = 500.). Plot the training error at the end of each epoch. Initialize the weight vector randomly from a uniform distribution (Eg: $U[-0.1, 0.1]$). Report the learned weight vector, the bias term and the train/test errors. Note: Make sure to append a '1' to your feature vectors so that a bias term is learned. This applies to subsequent problems as well.

   (c) Use **batch gradient descent** to train your model. Choose appropriate values for the learning rate and number of epochs (Eg: Learning rate = 5e-2, number of epochs = 500). Implement the weight update rule and train the model until convergence. Plot the training error (MSE) at the end of each epoch. Report the learned weight vector, the bias term and the train/test errors.

(d) Use the closed form solution to the regression problem to identify the weights and bias that minimize the MSE. Do they agree with the parameters you learned in the previous parts? Report the weight vector, bias and the train/test errors.

(e) You may have noticed that the test error is smaller than the training error in the previous experiments. This is due to some bias in the way the train and test split was chosen. To remove this bias consider the following exercise. Construct 100 random train/test splits and solve for the optimal w (using the closed form solution) for each split. Compute the training and test error in each case and report the mean training error and the mean test error. You should expect to see lower or similar mean training error. Feel free to use the starter (pseudo)code in canvas `prob1e.py` for this part.

2. (**25 points**) In this problem, we will explore the relationship between model capacity, training-set size and overfitting. We will use the same dataset and train/test splits that were used for the previous problem. Submit your code for this problem under filename `prob2.py`.

(a) We will consider polynomial features and observe how the performance varies with different polynomial orders. Consider polynomial features of order $\{0, 1, 2, 3, 4\}$. Solve for the parameter vector for each polynomial order using the closed form solution and plot training and test errors with respect to polynomial order. Plot the root mean-squared error (RMSE) instead of MSE for better axis scaling.

Note: Order 0 polynomial features corresponds to having a single constant feature (1), so that the model has a single parameter to be learned. Order 3 corresponds to deriving features x, x^2, x^3 for each individual feature. Although we could consider polynomial feature functions that depend on multiple features, we will only consider the simplistic setting where features are computed independently as above.

(b) In this part we will observe how performance varies with the size of the training set. Consider four cases where you train the model using only the first $20\%, 40\%, 60\%, 80\%, 100\%$ of the given training split. In each case, solve for the parameter vector and evaluate the training and test errors. Use the raw features for this part (order 1). Plot the training and test errors against training set size. Again, use RMSE as the error metric in your plot.

3. (**25 points**) Consider the following objective function for regularized least squares

$$E(w) = \frac{1}{2N}\sum_{i=1}^{N}(w^T\phi(x_n) - t_n)^2 + \frac{\lambda}{2}||w||^2 \tag{1}$$

where the training data instances are given by $D = \{(x_i, y_i)\}^N$, λ is a regularization parameter and w is the parameter vector.

(a) Derive a closed form solution for the weight vector to this regularized least squares objective function.

(b) In this part, we will search over different hyperparameter values to identify their best setting. For this purpose, split the training set so that the first 90% constitutes the new training set and the remaining 10% form the validation set. Fit a regularized linear regression model to the training set and estimate the loss on the validation set for each of the following values for the regularization parameter $\lambda \in \{0, 0.1, 0.2, 0.3, 0.4, 0.5\}$. Report the lowest RMSE error identified on the validation set and the corresponding value for λ. Also report the test error for this chosen λ. Submit your code under filename `prob3.py`.

4. (**25 points**) **Weighted Linear Regression**. Consider a linear regression problem in which we want to weigh different training examples differently. Specifically, suppose we want to minimize

$$E(w) = \frac{1}{2}\sum_{n=1}^{N} r_n(w^T x_n - t_n)^2 \tag{2}$$

The version of linear regression we saw in class corresponds to the case where all the weights r_i are one. In this problem, we will generalize some of those ideas to the weighted setting. In other words, we will allow the weights r_i to be different for each of the training examples.

(a) Show that $E(w)$ can also be written as

$$E(w) = (Xw - t)^T R(Xw - t) \tag{3}$$

for an appropriate definition of X, t and R, i.e., write the derivation as part of your submitted answer. State these three quantities clearly as part of your answer (i.e., specify the form, matrix, vector, scalar, their dimensions, and how to build them from the x_i's, t_i's and r_i's).

(b) Find the value of w that minimizes the above loss function by computing the derivative and setting it to zero. Express this optimal $w = w^*$ in terms of X, R and t.

(c) Suppose we have a training set $\{(x_i, t_i); i = 1, ..., N\}$ of N independent examples, but in which the t_i's were observed with different variances. Specifically, suppose that

$$p(t_i|x_i; w) = \frac{1}{\sqrt{2\pi}\sigma_i} exp\left(-\frac{(t_i - w^T x_i)^2}{2\sigma_i^2}\right) \tag{4}$$

In other words, t_i has mean $w^T x_i$ and variance σ_i^2 (where the σ_i's are fixed, known constants). Show that finding the maximum likelihood estimate of w reduces to solving a weighted linear regression problem. State clearly what the r_i's are in terms of the σ_i's.