EECS545 - Homework 2

January 29th, 2018

Instructions

- This homework is Due Monday, February the 12th 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_hw2.pdf`.
- Submit all your python code in a compressed zip file named `uniqname_hw2_code.zip`. Your zip file should contain `prob1.py`, `prob2.py`, `prob3.py`, `prob4.py`, `prob5.py` and `prob6.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 a locally weighted linear regression model and compare it with a linear regression model.

   We will be using synthetic data for this problem (generated by the function `data_generator` in `prob1.py`). Each instance is a real number $x_i \in \mathbb{R}$ sampled from the uniform distribution $U[0, 3]$, and its continuous label is sampled from a Gaussian distribution $\mathcal{N}(0.5x - 0.3 + sin(3x), 0.05^2)$. The size and noise scale σ are treated as parameters.

   You will use it to generate 100 instances for the training set and 30 instances for the test set.

   Our objective function will be the mean-squared error (MSE) loss function. Use the starter code provided and follow the instructions for this problem. You are allowed to use code from Homework 1. 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. Append a '1' to your feature vectors so that a bias term is learned. This applies to subsequent problems as well. (If you encounter division by zero for a feature, only subtract the mean for that feature). Reuse the closed form solution for the linear regression model from HW1. Plot the predicted labels for all test instances and report the test error.

   (b) Fit a locally weighted linear regression model to the training set with a Gaussian kernel with kernel width τ. Report the test errors and plot the predicted labels for $\tau \in \{0.2, 2.0\}$ for all test instances.

   Note: You're supposed to provide three errors and three plots in this problem.

2. (**15 points**) In this problem, we will plot marginal distributions and conditional distributions for multivariate Gaussians. Please follow the instructions to complete the functions in `prob2.py`. Submit your code for this problem under filename `prob2.py`. Feel free to use those recommended useful functions for `matplotlib` in starter code.

   (a) Complete the function `marginal_distribution`. This function receives the parameters of a multivariate Gaussian distribution over variables $X_1, X_2, ..., X_N$ as input and computes the marginal distribution of a subset of the variables $X_{i_1}, X_{i_2}, ..., X_{i_n}$, given the list of indices $\{i_1, i_2, ..., i_n\}$ as input.

   (b) Consider the multivariate Gaussian distribution defined by

   $$(X_1, X_2) \sim \mathcal{N}(\mu, \Sigma)$$

   where

   $$\Sigma = \begin{bmatrix} 1.0 & 0.5 \\ 0.5 & 1.0 \end{bmatrix}, \mu = [0, 0]^T$$

   Compute and plot the marginal distribution $P(X_1)$.

   (c) Complete the function `conditional_distribution`. This function receives the parameters of a multivariate Gaussian distribution over variables $X_1, X_2, ..., X_N$ as input and computes, given a set of selected variables and the values for these variables, the conditional distribution of the remaining variables.

   (d) Consider the multivariate Gaussian distribution defined by $(X_1, X_2, X_3, X_4) \sim \mathcal{N}(\mu, \Sigma)$ where

   $$\Sigma = \begin{bmatrix} 1.0 & 0.5 & 0.0 & 0.0 \\ 0.5 & 1.0 & 0.0 & 1.5 \\ 0.0 & 0.0 & 2.0 & 0.0 \\ 0.0 & 1.5 & 0.0 & 4.0 \end{bmatrix}, \mu = \begin{bmatrix} 0.5 \\ 0.0 \\ -0.5 \\ 0.0 \end{bmatrix}$$

   Compute and plot the conditional distribution $P(X_1, X_4; X_2 = 0.1, X_3 = -0.2)$.

   Note: Please report the covariance matrices and mean vectors, and plot the Gaussian distributions. If the Gaussian distribution is two-dimensional, please provide the *contour* plot for it.

3. (**25 points**) In this problem, we will implement a sequential Bayesian linear regression algorithm. Recall the example in slides; assume an observation y is generated by

   $$y = w_1 x + w_0 + \epsilon$$

   where

   $$w_1 = 0.5, w_0 = -0.3, \epsilon \sim \mathcal{N}(0, \beta^{-1}), \beta = 1.0$$

   and we want to learn the coefficients w_0 and w_1. We initialize the model with priors $P(w|m_0, S_0)$ where

   $$m_0 = [0, 0], S_0 = \alpha^{-1} I, \alpha = 2.0.$$

   Please follow the instructions to complete the functions in `prob3.py`. Feel free to use the recommended `matplotlib` functions for plotting in the starter code. A synthetic data generating function is provided in starter code. Please plot the initial distribution of w and the distributions after **1, 10, 20** instances. Notice that you should **not** do normalization in this

problem. (Please change the starter code and make sure the covariance matrix is initialized with $\alpha^{-1}I$ where $\alpha = 2.0$.)
Note: We slightly changed the starter code `prob3.py`, please make sure you're using the new version. There are four covariance matrices, mean vectors and **heatmaps** (by imshow()) required.

4. (**15 points**) We will fit a Gaussian process (GP) regression model to regress from 1-dimensional input features to labels in this problem. Please name your python script for this problem `prob4.py`. Consider the GP specified by the following mean and covariance functions.

$$y(x) \sim \mathcal{GP}(m(x), k(x, x'))$$
$$m(x) = 0, k(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

   (a) i. Given $x_1, x_2, ..., x_n \in \mathbb{R}$, write down the joint distribution over $y(x_1), y(x_2), ..., y(x_n)$.
   ii. Implement a function to plot sample functions from the above GP prior. Plot five sample functions in the range $x \in [-5, 5]$ at 100 equally spaced points (`numpy.linspace(-5, 5, 100)`) for the following values of the kernel parameter $\sigma \in \{0.3, 0.5, 1.0\}$.
   You may find the `numpy.random.multivariate_normal` function useful.

   (b) i. Suppose we observe training data with input/output pairs $D = \{(x_0^D, y_0^D), ..., (x_m^D, y_m^D)\}$. Consider an arbitrary set of points $x_0, x_1, ..., x_n \in \mathbb{R}$. Express the conditional distribution $p(y(x_0), y(x_1), ..., y(x_n)|D)$ using appropriately defined mean and covariance matrices.
   ii. Let $D = \{(-1.3, 2), (2.4, 5.2), (-2.5, -1.5), (-3.3, -0.8), (0.3, 0.3)\}$. For each of the three values of σ above, produce a plot clearly featuring the following.
   - The points in D
   - The posterior mean of the GP (thick curve)
   - Five sample functions from the posterior (thin curves)

5. (**25 points**) In this problem, you will implement the update rule to train a perceptron. You will find starter code in `prob5.py` on canvas. Please submit your code in a file `prob5.py`.

   You are given an array 'data', consisting of three features (an 'x' value, a 'y' value, and a bias term equal to 1). You have a 'target' array consisting of values -1 or 1. Your goal is to find a vector w such that for each element of the data array x_i and corresponding target value t_i:

$$w^T x_i > 0 \Leftrightarrow t_i > 0$$

   Do this by implementing the perceptron algorithm. Initialize your w vector to zeros, and iterate through the dataset 10 times, each time updating the w vector for each datum via the perceptron update rule.

   5Create a single plot, consisting of:

   - A scatter plot of x and y values from the dataset, with points colored based on their corresponding target value.
   - A line corresponding to the linear separator learned by your algorithm (points on this line x should be values such that $w^T x = 0$.

Do the same for the data array in `prob5b.py`. You will notice that this dataset is not seperable, so also propose a new method for learning a linear boundary between the datasets, implement this method, and make a scatter plot showing the results. This can be any adjustment to the update rule for the perceptron algorithm you feel might yield a good result.

6. (**25 points**) For this problem use the breast cancer wisconsin dataset. You can import this into your code via `load_breast_cancer` within `sklearn.datasets`. You will find starter code in `prob6.py` on canvas.

   Perform logistic regression to train a binary classifier using this dataset. Specifically, learn a parameter vector w such that the cross-entropy loss

   $$E(w) = -\left[\sum_i t_i \ln(\sigma(w^T x_i)) + (1 - t_i) \ln(1 - \sigma(w^T x_i))\right]$$

   is minimized. Initialize each element of w randomly between -1 and 1, and perform the minimization via stochastic gradient descent with a learning rate of $1e-2$. Use a test set that is $\frac{1}{3}$rd the size of the full dataset. Normalize the data as you did for the previous homework. Iterate through each example in the training set once in random order (that is, we only train for a single epoch). Make two plots, showing:

   - The training and test classification accuracy vs. the number of iterations of SGD that have been performed.
   - The average training and test error vs. the number of iterations of SGD that have been performed.

   Report your learned parameter vector w (this should be 30-dimensional). Report your final training and test cross-entropy and classification accuracy. Put your code in file `prob6.py`.