CMPS 242 Fourth Homework, Fall 2019

3 Problems. 13 pts, due 11:50pm Friday Nov. 8

This homework is to be done in groups of 2. Each group members should completely understand the group's solutions and *must* acknowledge all sources of inspiration, techniques, and/or helpful ideas (web, people, books, etc.) other than the instructor, TA, and class text. Each group should sign up on Canvas and electronically submit a single set of solutions for their group. See the Piazza post **New method for submitting homework** for more details.

Your solutions should be clearly numbered and in order. Although there are no explicit points for "neatness", the TA may deduct points for illegible or poorly organized solutions.

I *strongly* recommend that each student work on all of the problems individually before integrating their solutions into the group consensus (although the programming problems might be done as "pair programming"). Dividing up the problems so each student does just a subset of them is the wrong approach.

1. (3 pts) Prove that for an arbitrary number of examples N and number of features D, and set of N examples with D (real-valued) features, that the Least Squares cost function $J(\theta)$ is a convex function of the D-dimensional parameter vector θ.

   Recall that

   $$J(\theta) = \frac{1}{2}\sum_{i=1}^{N}(h_\theta(\mathbf{x}_i) - t_i)^2$$

   and the hypothesis $h_\theta(\mathbf{x})$ is $\theta \cdot \mathbf{x}$. You may either show that the Hessian H is positive semi-definite (easier, but requires more linear algebra knowledge), or that for any $\alpha \in [0, 1]$ and any parameter vectors v and w, we have $\alpha J(v) + (1-\alpha)J(w) \geq J(\alpha v + (1-\alpha)w)$ (a little longer, but simpler algebra). It may be helpful to prove it first for single examples and then use the property that the sum of convex functions is also a convex function. It may be helpful to know that for any real vector v, we have $v^\mathsf{T} v = \sum_i v_i^2 \geq 0$. (Hint: to gain intuition, you might want to examine the single example case with $D = 1$ and $D = 2$ first.)

2. (6 pts) Linear Regression

   The following table describes the training data for this problem:

x_1	x_2	x_3	t
3	9	2	19
6	9	1	19
7	7	7	10
8	6	4	11
1	0	8	-3

   Note: This is artificial & slightly cherry picked data. For each instance, the features x_i were generated randomly and then noise was added. The targets were generated by the

following function before the noise: $t = 0 * x_{1\ (\text{clean})} + 2 * x_{2\ (\text{clean})} - 1 * x_{3\ (\text{clean})} + 3$. Your computed weight vector should be somewhat similar to this, but with so few examples and so much relative noise you shouldn't expect to get really close.

(a) Save the data and run a linear regression algorithm on the full training set *without a bias component*. (Scikit learn has a `fit_intercept` parameter that can be set to false to do this.) Report the model (weights) and "Root mean squared error". Note that testing on the training set means we will may have relatively small error. Now add an x_0 feature that is always 1 to each example, and re-run the regression. Did the model or accuracy change? **We will use this augmented 4-feature training set for the rest of the problem.**

(b) Suppose you had an unlabeled instance $\mathbf{x} = [3, 3, 5]$. What $\hat{t}$ prediction for t would the second model from part (a) give?

(c) The "ridge parameter" is another name for the $L2$ regularization coefficient (often written λ). Run the linear regression (ridge regression) with small and large regularization parameters, say 0.0001 and 0.3. What are learned weights? Is the change qualitatively what you would expect?

(d) Stochastic Gradient Descent. Start with the weight vector $w = [1, 1, 1, 1]$ and step through stochastic gradient descent on the first two instances: $x = [1, 3, 9, 2], t = 19$ and $x = [1, 6, 9, 1], t = 19$ (note that these include the bias component). For your learning rate parameter, use $\eta = 0.01$, For your error function use squared error. Report the updated weights after each instance. Show your work for at least the first instance.

(e) Compute regression weights using eq. 3.34 from Bishop. In our notation that is: $\mathbf{w} = (X^\mathsf{T} X)^{-1} X^\mathsf{T} \mathbf{t}$, here X is the (5-row, 4-column) matrix whose rows are the unlabeled instances and whose columns are each of the features for the instance augmented by the fixed feature $x_0 = 1$ which will lead to the bias. How does this $\mathbf{w}$ compare to the weights from part a? If you like, you may use your favorite math software for the matrix arithmetic but it is quite feasible to do directly, except for the inverst. You should not compute the inverse by hand (Google "inverse matrix 4x4 calculator").

(f) If the examples are re-ordered (so the rows of X and elements of $\mathbf{t}$ are permuted the same way), what happens to the learned $\mathbf{w}$ vector and why?

3. (4 pts) Experiments on Linear Regression.

For the purpose to comparing the results, in this problem we use the sum-square-loss, Equation (3.26) from Bishop, as the loss function. In our notation, it will be

$$J(\theta) = \frac{1}{2} \sum_{n=1}^{N} (t_n - \theta \cdot \mathbf{x}_n)^2$$

Let N be the number of data points of the data set, and σ^2 be the variance of the noise. Generate a data set $D = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$ by these steps:

- Every feature vector $\mathbf{x}_n$ has two components: $\mathbf{x}_n = (x_{n,1}, x_{n,2})$. Each feature is generated by an uniform distribution over the interval $[-1, 1]$ independently, i.e. $\mathbf{x}_n \sim \text{Uniform}([-1, 1] \times [-1, 1])$.
- Let $\theta^* = (2, -1)$ be the true vector generating the labels. The label t_n of $\mathbf{x}_n$ is generated by the following formula:

$$t_n = \theta^* \cdot \mathbf{x}_n + \epsilon_n = 2x_{n,1} - x_{n,2} + \epsilon_n \quad , \text{ where } \epsilon_n \sim \mathcal{N}(0, \sigma^2)$$

(a) Closed form solution for Linear Regression
Generate two different data sets with $N = 10000$ examples and $\sigma^2 = 0.01$. Select the first k examples from the first data set as the training data set and use the second data set as the test data set. Use the closed form solution to minimize the MSE on the training data. Report the training error and test error for $k = 10, 100, 1000, 10000$. And explain the results you get.

(b) Stochastic Gradient Descent
Generate a training data set with $N = 10000, \sigma^2 = 1$. And use the Stochastic Gradient Descent algorithm:

- Given a data set $D = \{(\mathbf{x}_n, t_n)\}_{n=1}^N$, step size η and number of iterations T.
- $\theta_0 \leftarrow (0, \cdots, 0)$
- For $i = 1, \cdots, T$ Do
  - $n \sim \text{Uniform}(\{1, 2, \cdots, N\})$
  - $\theta_i \leftarrow \theta_{i-1} + \eta(t_n - \theta_{i-1} \cdot \mathbf{x}_n)\mathbf{x}_n$

Run the algorithm with $T = 10000$ and $\eta = 0.1, 0.01, 0.001$. Plot the MSE curves (say with a data point every 100 updates) for these three cases. Why would the different η's lead to different curves?

Recommend functions in Matlab / Python:

	Matlab	*Python*
Generate uniform samples	*rand*	*numpy.random.rand*
Generate normal distributions	*randn*	*numpy.random.randn*
Inverse	*inv*	*numpy.linalg.inv*
plot the results	*plot*	*matplotlib.pyplot.plot*

Additional problems (optional):

- Repeat 3b, but anneal the learning rate η to be something like $1/i$ in each iteration i.
- include regularization in problem 3 (see equation 3.27 from Bishop)