EECS545: Machine Learning, Winter 2022
Homework #1

Due 11:55pm on 1/25 (Tue).

Reminder: While you are encouraged to think about problems in small groups, all written solutions must be independently generated. Please typeset (with LaTeX) or hand-write your solutions legibly. Hand-writings that are very difficult to read may lead to some penalties. If you prefer hand-writing, please use scanners or mobile scanning apps that provide some correction for shading and projective transformations for better legibility; you should not just take photos and submit them.

While these questions require thought, please be as concise and clear in your answer as possible. Please address all questions to `piazza.com/umich/winter2022/eecs545wn2022/home` with a reference to the specific question as well as an optional summary in the subject line (e.g. Homework 1, Q2(c): Can we assume XXX conditions?).

Homeworks must be submitted via Gradescope (`https://gradescope.com/`) as pdf files. For any solutions that require programming, please use Numpy to implement the machine learning algorithms from scratch (no high-level libraries are allowed). Please submit code as well as the written solution including all derivations and figures you are asked to plot.

1 [42 points] Linear regression on a polynomial

The files `q1xTrain.npy`, `q1xTest.npy`, `q1yTrain.npy` and `q1yTest.npy` specify a linear regression problem for a polynomial. `q1xTrain.npy` represent the inputs ($\mathbf{x}^{(i)} \in \mathbb{R}$) and `q1yTrain.npy` represents the outputs ($y^{(i)} \in \mathbb{R}$) of the training set, with one training example per row.

(a) **[15 points]** You will compare the following two optimization methods, in finding the coefficients of a polynomial of degree one (i.e. slope and intercept) that minimize the training loss.

- Batch gradient descent
- Stochastic gradient descent

i. **[12 points]** Give the coefficients generated by each of the optimization methods. Report the hyperparameters used to generate the coefficients.

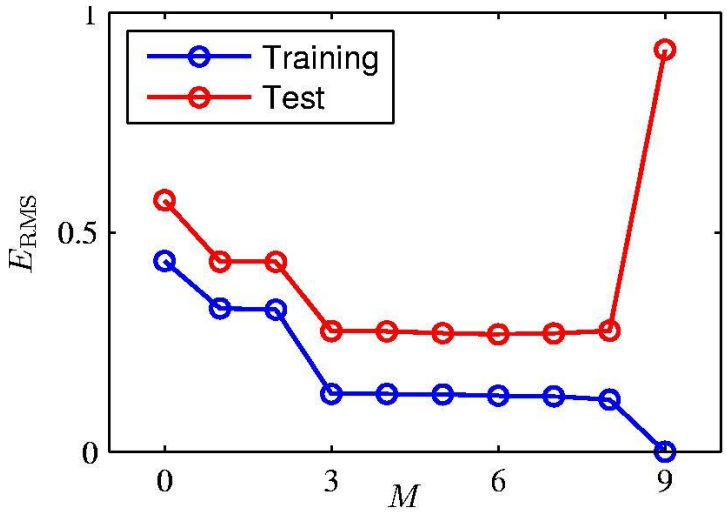
ii. **[3 points]** We compare two optimization methods in terms of the number of epochs required for convergence. We define an "epoch" as one pass through the entire training samples. Compare the number of epochs to converge for the methods. Which method converges faster? Report the hyperparameters used for comparison. [Hint: in this question, the training process can be viewed as convergent when the mean squared error $E_{MS} = \frac{1}{N}\sum_{i=1}^{N}(w_0 + w_1 x^{(i)} - y^{(i)})^2$ on the training dataset is consistently small enough (e.g. ≤ 0.2).]

(b) **[15 points]** Next, you will investigate the problem of over-fitting. Recall the figure from lecture that explored over-fitting as a function of the degree of the polynomial M, where the Root-Mean-Square (RMS) Error is defined as

$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N},$$

where

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=0}^{M} w_j\phi_j(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - y^{(i)}\right)^2:$$



i. **[10 points]** Regenerate the above chart with the provided data. To find the parameters, use the closed form solution of linear regression (assuming all the condition is met) that minimize the error for a M-degree polynomial (for $M = 0 \ldots 9$) for the training data in `q1xTrain.npy` and `q1yTrain.npy`. For the test curve, use the data in `q1xTest.npy` and `q1yTest.npy`. [Note: For different values of M, we assume the feature vector is $\phi(x^{(i)}) = (1, x^{(i)}, (x^{(i)})^2, \cdots, (x^{(i)})^M)$. The trend of your curve is not necessarily the same as the sample plot.]

ii. **[5 points]** Which degree polynomial would you say best fits the data? Was there evidence of under/over-fitting the data? Use your generated charts to justify your answer.

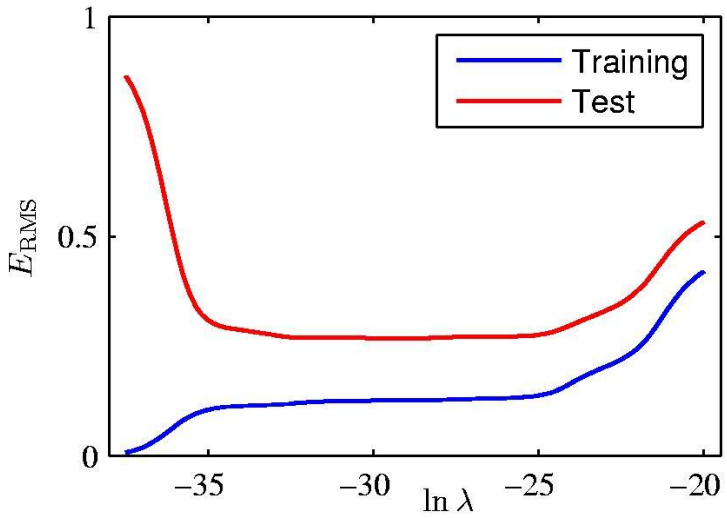
(c) **[12 points]** Finally, you will explore the role of regularization. Recall the image from lecture that explored the effect of the regularization factor λ:

i. **[10 points]** Derive the closed form solution of the ridge regression, find the coefficients that minimize the error for a ninth degree polynomial ($M = 9$) given regularization factor λ (for $\lambda \in \{0, 10^{-8}, 10^{-7}, 10^{-6}, 10^{-5}, \ldots, 10^{-1}, 10^{0}(= 1)\}$) for the training data specified in `q1xTrain.npy` and `q1yTrain.npy`. Now use these parameters to plot the E_{RMS} of both the training data and test data as a function of λ and regenerate the above chart, using `q1xTest.npy` and `q1yTest.npy` as the test data). Specifically, use the following regularized objective function:

$$\frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^T\phi(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2.$$

for optimizing the parameters $\mathbf{w}$, but please use the original (unregularized) E_{RMS} for plotting. [Note: the trend of your curve is not necessarily the same as the sample chart.]

ii. **[2 points]** Which λ value seemed to work the best?



2 [36 points] Locally weighted linear regression

Consider a linear regression problem in which we want to weight different training examples differently. Specifically, suppose we want to minimize

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N} r^{(i)}(\mathbf{w}^T\mathbf{x}^{(i)} - y^{(i)})^2.$$

where $r^{(i)}$ is the weight for the sample $(x^{(i)}, y^{(i)})$. In class, we worked out what happens for the case where all the weights (the $r^{(i)}$'s) are the same. In this problem, we will generalize some of those ideas to the weighted setting, and also implement the locally weighted linear regression algorithm. [Note: the weight $r^{(i)}$ can be different for each of the training example.]

(a) **[2 points]** Show that $E_D(w)$ can also be written as

$$E_D(\mathbf{w}) = (X\mathbf{w} - \mathbf{y})^T R(X\mathbf{w} - \mathbf{y})$$

for an appropriate diagonal matrix R, and where X is a matrix whose i-th row is $x^{(i)}$ and $\mathbf{y}$ is the vector whose i-th entry is $y^{(i)}$. State clearly what R is.

(b) **[8 points]** If all the $r^{(i)}$'s equal 1, then we saw in class that the normal equation is

$$X^T X\mathbf{w} = X^T\mathbf{y},$$

and that the value of $\mathbf{w}^*$ that minimizes $E_D(\mathbf{w})$ is given by $(X^TX)^{-1}X^T\mathbf{y}$. By finding the derivative $\nabla_{\mathbf{w}} E_D(\mathbf{w})$ and setting that to zero, generalize the normal equation and the closed form solution to this weighted setting, and give the new value of $\mathbf{w}^*$ that minimizes $E_D(w)$ in closed form as a function of X, R and $\mathbf{y}$.

(c) **[8 points]** Suppose we have a training set $\{(\mathbf{x}^{(i)}, y^{(i)}); i = 1\ldots, N\}$ of N independent examples, but in which the $y^{(i)}$'s were observed with differing variances. Specifically, suppose that

$$p(y^{(i)}|\mathbf{x}^{(i)}; \mathbf{w}) = \frac{1}{\sqrt{2\pi}\sigma^{(i)}} \exp\left(-\frac{(y^{(i)} - \mathbf{w}^T\mathbf{x}^{(i)})^2}{2(\sigma^{(i)})^2}\right)$$

I.e., $y^{(i)}$ has mean $\mathbf{w}^T\mathbf{x}^{(i)}$ and variance $(\sigma^{(i)})^2$ (where the $\sigma^{(i)}$'s are fixed, known, constants). Show that finding the maximum likelihood estimate of $\mathbf{w}$ reduces to solving a weighted linear regression problem. State clearly what the $r^{(i)}$'s are in terms of the $\sigma^{(i)}$'s.

(d) **[18 points]** The following items will use the files `q2x.npy` which contains the inputs ($\mathbf{x}^{(i)}$) and `q2y.npy` which contains the outputs ($y^{(i)}$) for a linear regression problem, with one training example per row.

i. **[4 points]** Implement (unweighted) linear regression ($y = \mathbf{w}^T\mathbf{x}$) on this dataset (using the closed form solution we learned in lectures, remember to include the intercept term.). Plot on the same figure the data (each data sample can be shown as a point $(x^{(i)}, y^{(i)})$ in the figure) and the straight line resulting from your fit.

ii. **[8 points]** Implement locally weighted linear regression on this dataset (using the weighted normal equations you derived in part (b)), and plot on the same figure the data and the curve resulting from your fit. When evaluating local regression at a query point x (which is real-valued in this problem), use weights

$$r^{(i)} = \exp\left(-\frac{(x - x^{(i)})^2}{2\tau^2}\right)$$

with a bandwidth parameter $\tau = 0.8$. (Again, remember to include the intercept term.)

iii. **[6 points]** Repeat (ii) four times with $\tau = 0.1$, 0.3, 2 and 10. Comment **briefly** on what happens to the fit when τ is too small or too large.

3 [22 points] Derivation and Proof

(a) **[8 points]** Consider the linear regression problem for 1D data, where we would like to learn a function $h(x) = w_1 x + w_0$ with parameters w_0 and w_1 to minimize the sum squared error $L = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - h(x^{(i)}))^2$ for N pairs of data samples $(x^{(i)}, y^{(i)})$. Derive the solution for w_0 and w_1 for this 1D case of linear regression. Show the derivation to get the solution $w_0 = \bar{Y} - w_1\bar{X}$ and $w_1 = \frac{\frac{1}{N}\sum_{i=1}^{N} x^{(i)}y^{(i)} - \bar{Y}\bar{X}}{\frac{1}{N}\sum_{i=1}^{N} x^{(i)2} - \bar{X}^2}$ where $\bar{X}$ is the mean of $\{x^{(1)}, x^{(2)}, \cdots, x^{(N)}\}$ and $\bar{Y}$ is the mean of $\{y^{(1)}, y^{(2)}, \cdots, y^{(N)}\}$.

(b) **[14 points]** Consider the definition and property of positive (semi-)definite matrix. Let $\mathbf{A}$ be a real, symmetric $d \times d$ matrix. $\mathbf{A}$ is positive semi-definite (PSD) if, for all $\mathbf{z} \in \mathcal{R}^d$, $\mathbf{z}^T\mathbf{A}\mathbf{z} \geq 0$. $\mathbf{A}$ is positive definite (PD) if, for all $\mathbf{z} \neq \mathbf{0}$, $\mathbf{z}^T\mathbf{A}\mathbf{z} > 0$. We write $\mathbf{A} \succeq 0$ when $\mathbf{A}$ is PSD, and $\mathbf{A} \succ 0$ when $\mathbf{A}$ is PD. The spectral theorem says that every real symmetric matrix $\mathbf{A}$ can be expressed via the spectral decomposition $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^T$ where $\mathbf{U}$ is a $d \times d$ matrix such that $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}$ and $\mathbf{\Lambda} = diag(\lambda_1, \lambda_2, \cdots, \lambda_d)$. Multiplying on the right by $\mathbf{U}$ we see that $\mathbf{A}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$. If we let u_i denote the i-th column of $\mathbf{U}$, we have $\mathbf{A}u_i = \lambda_i u_i$ for each i. Then λ_i are eigenvalues of $\mathbf{A}$, and the corresponding columns are eigenvectors associated to λ_i. The eigenvalues constitute the "spectrum" of $\mathbf{A}$, and the spectral decomposition is also called the eigenvalue decomposition of $\mathbf{A}$.

i. **[6 points]** Prove $\mathbf{A}$ is PD iff $\lambda_i > 0$ for each i.

ii. **[8 points]** Consider the linear regression problem where Φ and $\mathbf{y}$ are as defined in class and the closed form solution is $(\Phi^T\Phi)^{-1}\Phi^T\mathbf{y}$. We can get the eigenvalues of symmetric matrix $\Phi^T\Phi$ using spectral decomposition. We apply ridge regression, and the symmetric matrix in the solution is $\Phi^T\Phi + \beta\mathbf{I}$. Prove that the ridge regression has an effect of shifting all singular values by a constant β. For any $\beta > 0$, ridge regression makes the matrix $\Phi^T\Phi + \beta\mathbf{I}$ PD.

Code Submission Instructions

Your solution to Q1 and Q2 should be written in a *single* python program `q1.py` and `q2.py`, respectively. The program should be runnable with the following command line: `python3 q1.py` (or `python3 q2.py`) and produce outputs and plots for *all* subproblems. The program should read the data files (i.e., `*.npy`) from the same (current) working directory: for example, `X_train = np.load('q1xTrain.npy')`.

The program should print all necessary outputs (e.g. coefficients computed) into standard output (stdout). For plots, it is okay to save figures as multiple files (e.g. `q1-b.png`) as you want. There are no requirements on the filename or format, as long as it produces valid outputs. Be sure to include all outputs in your writeup.

Please upload your code files (`q1.py` and `q2.py`) to Gradescope. Additional python files are allowed. It is fine to include your outputs into your submission, but this is not required (we will re-run the code). However, please DO **NOT** include data files (`*.npy`) into your Gradescope submission.

Credits

Some questions adopted/adapted from Stanford CS229 materials and from Bishop PRML.