EECS545 WN24 Machine Learning
Homework #1

Due date: 11:55 PM, Tuesday Jan 30, 2024

Reminder: While you are encouraged to discuss problems in a small group (*up to 5 people*), you should write your solutions and code independently. Do not share your solution with anyone else in the class. If you worked in a group for the homework, you **should** include the names of **all** your collaborators in the homework submission. Your answers should be as concise and clear as possible. Please address all questions to `http://piazza.com/class#winter2024/eecs545` with a reference to the specific question in the subject line (e.g., Homework 1, Q2(c): can we assume XXX conditions?).

Submission Instruction: You should submit both **writeup** and **source code**. We may inspect your source code submission visually and rerun the code to check the results. Please be aware your points can be deducted if you don't follow the instructions listed below:

- Submit **writeup** to **Gradescope** (`https://www.gradescope.com/`)
  - Your writeup should contain your answers to **all** questions (typeset or hand-written), except for the implementation-only questions (marked with **(Autograder)**).
  - **For each subquestion, please select the associated pages from the pdf file in Gradescope. The graders are not required to look at pages other than the ones selected, and this may lead to some penalty when grading.**
  - Your writeup should be self-contained and self-explanatory. You should include the plots and figures in your writeup whenever asked, even when they are generated from the ipython notebook.
  - Please typeset (with LaTeX) or hand-write your solutions legibly. **Hand-writing that is difficult to read may lead to penalties**. If you prefer hand-writing, please use scanners or mobile scanning apps that provide shading corrections and projective transformations for better legibility; you **should *not*** just take photos and submit them without any image processing.
  - For all math derivations, you **should** provide detailed explanations as much as possible. If not, you may not be able to get a full credit if there are any mistakes or logical jumps.
- Submit **source code** to **Autograder** (`https://autograder.io/`)
  - Each ipython notebook file (i.e., `*.ipynb`) will walk you through the implementation task, producing the outputs and plots for *all* subproblems. You are required to write code on its matched python file (`*.py`). For instance, if you are working on `linear_regression.ipynb`, you are required to write code on `linear_regression.py`. Note that the outputs of your source code must match with your **writeup**.
  - We will read the data file in the `data` directory (i.e., `data/*.npy`) **from the same (current) working directory**: for example, `X_train = np.load('data/q3x.npy')`.
  - When you want to evaluate and test your implementation, please submit the `*.py` and `*.ipynb` files to Autograder for grading your implementations in the middle or after finish everything. You can partially test some of the files anytime, but please note that this also reduces your daily submission quota. You can submit your code up to **three** times per day.

- The autograder will give you information to help you troubleshoot your code. The following is the current list of error codes (will be updated accordingly if we encounter more common student implementation errors).

  0: Success

  1: The program exited with an unknown test failure.

  2: The program exited with an un-handled exception (usually a runtime error)

  3-5: Some other runtime error indicating an issue with the autograder code.

  40: NotImplementedError

  50: An uncategorized test failure different from the following error codes. Most likely indicates wrong outputs, but this may also include other assertion failures.

  51: There is an incorrect dtype on some output tensor.

  52: There is an incorrect shape on some output tensor.

  53: Some output tensors are not equal, or too different given the tolerance.

  54: Some output tensors are not equal with difference 1.0. Possibly due to rounding issues.

  55: The value of some input tensor was changed.

  60: The model's performance or converged loss is not good enough.

  61: The model might have diverged (e.g. encountered NaN or inf).

  62-63: The model has converged to a bad likelihood.

- Your program should run under an environment with following libraries:
  - Python 3.11 [1]
  - NumPy (for implementations of algorithms)
  - Matplotlib (for plots)

  Please do not use any other library unless otherwise instructed (no third-party libraries other than numpy and matplotlib are allowed). You should be able to load the data and execute your code on Autograder with the environment described above, but in your local working environment you might need to install them via `pip install numpy matplotlib`.

- For this assignment, you will need to submit the following files:
  - `linear_regression.py`
  - `linear_regression.ipynb`

  Do not change the filename for the code and ipython notebook file because the Autograder may not find your submission. Please do not submit any other files except `*.py` and `*.ipynb` to Autograder.

- When you are done, please upload your work to Autograder (sign in with your UMich account). To receive the full credit, your code must run and terminate without error (i.e., with exit code 0) in the Autograder. Keep all the cell outputs in your notebook files (`*.ipynb`). We strongly recommend you run **Kernel → Restart Kernel and Run All Cells** (in the Jupyter Lab menu) before submitting your code to Autograder.

Credits

Some problems were adopted Stanford CS229 and Bishop PRML.

[1] We recommend using Miniconda (`https://docs.conda.io/en/latest/miniconda.html`). You can create `eecs545` environment with Python 3.11 as `conda create --name eecs545 python=3.11`. It is fine to use other python distributions and versions as long as they are supported, but we will run your program with Python 3.11 on Autograder.

Changelog

- rev0 (2024/1/16 1 PM): Initial Release
- rev1 (2024/1/18 8 PM): Updates in Q3 (a) and (b) to match all with column-major notations (Special thanks to Ishan Kapnadak for the correction). Updated transpose notations and the text are marked as red below in 'before' ⇒ 'after' format.

  (a)
    - $E_D(\mathbf{w}) = (\mathbf{w}^\top X - \mathbf{y})^\top R(\mathbf{w}^\top X - \mathbf{y}) \quad \Rightarrow \quad E_D(\mathbf{w}) = (\mathbf{w}^\top X - \mathbf{y})R(\mathbf{w}^\top X - \mathbf{y})^\top$
    - X is a matrix whose i-th **row** is $x^{(i)}$ ⇒ X is a matrix whose i-th **column** is $x^{(i)}$

  (b)
    - $X^\top X\mathbf{w} = X^\top \mathbf{y} \quad \Rightarrow \quad XX^\top \mathbf{w} = X\mathbf{y}^\top$
    - $(X^\top X)^{-1}X^\top \mathbf{y} \quad \Rightarrow \quad (XX^\top)^{-1}X\mathbf{y}^\top$
    - $\nabla_{\mathbf{w}}(RX\mathbf{w}) = X^\top R^\top = X^\top R \quad \Rightarrow \quad \nabla_{\mathbf{w}}(RX^\top \mathbf{w}) = XR^\top = XR$

- rev2 (2024/1/19 11:30 AM): Additional updates in Q3 (a) and (b): we added $X, \mathbf{y}, \mathbf{w}$ with the dimensions ($X \in \mathbb{R}^{D\times N}, \mathbf{y} \in \mathbb{R}^{N\times 1}, \mathbf{w} \in \mathbb{R}^{D\times 1}$), and made the equations match with the dimensions. Changes in transpose notations from 'rev1' are highlighted in red and are presented below in a 'before' ⇒ 'after' format.

  (a)
    - $E_D(\mathbf{w}) = (\mathbf{w}^\top X - \mathbf{y})R(\mathbf{w}^\top X - \mathbf{y})^\top \quad \Rightarrow \quad E_D(\mathbf{w}) = (\mathbf{w}^\top X - \mathbf{y}^\top)R(\mathbf{w}^\top X - \mathbf{y}^\top)^\top$

  (b)
    - $XX^\top \mathbf{w} = X\mathbf{y}^\top \quad \Rightarrow \quad XX^\top \mathbf{w} = X\mathbf{y}$
    - $(XX^\top)^{-1}X\mathbf{y}^\top \quad \Rightarrow \quad (XX^\top)^{-1}X\mathbf{y}$

1 [31 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:

$$L = \frac{1}{2}\sum_{i=1}^{N}(y^{(i)} - h(x^{(i)}))^2 \tag{1}$$

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

$$w_0 = \overline{Y} - w_1\overline{X}, \tag{2}$$

$$w_1 = \frac{\frac{1}{N}\sum_{i=1}^{N} x^{(i)}y^{(i)} - \overline{Y}\,\overline{X}}{\frac{1}{N}\sum_{i=1}^{N}(x^{(i)})^2 - \overline{X}^2} \tag{3}$$

where $\overline{X}$ is the mean of $\{x^{(1)}, x^{(2)}, \cdots, x^{(N)}\}$ and $\overline{Y}$ is the mean of $\{y^{(1)}, y^{(2)}, \cdots, y^{(N)}\}$.

(b) [**14 points**] Recall 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 $\mathbf{z} \in \mathbb{R}^d$, $\mathbf{z}^\top \mathbf{A}\mathbf{z} \geq 0$. $\mathbf{A}$ is positive definite (PD) if, for all $\mathbf{z} \neq \mathbf{0}$, $\mathbf{z}^\top \mathbf{A}\mathbf{z} > 0$. We write $\mathbf{A} \succeq 0$ when $\mathbf{A}$ is PSD, and $\mathbf{A} \succ 0$ when $\mathbf{A}$ is PD.

It is known that every real symmetric matrix $\mathbf{A}$ can be factorized via the eigenvalue decomposition: $\mathbf{A} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ where $\mathbf{U}$ is a $d \times d$ matrix such that $\mathbf{U}\mathbf{U}^\top = \mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \lambda_2, \cdots, \lambda_d)$. Multiplying on the right by $\mathbf{U}$ we see that $\mathbf{A}\mathbf{U} = \mathbf{U}\mathbf{\Lambda}$. If we let $\mathbf{u}_i$ denote the i-th column of $\mathbf{U}$, we have $\mathbf{A}\mathbf{u}_i = \lambda_i\mathbf{u}_i$ for each i, λ_i are eigenvalues of $\mathbf{A}$, and the corresponding columns of $\mathbf{U}$ are eigenvectors associated to λ_i. The eigenvalues constitute the "spectrum" of $\mathbf{A}$.

i. [**6 points**] Prove $\mathbf{A}$ is PD *if and only if* $\lambda_i > 0$ for each i (Note: *if and only if* means you are asked to prove it for both directions).

ii. [**8 points**] Consider the linear regression problem where Φ and $\mathbf{y}$ are as defined in class; we saw that the closed form solution becomes $(\Phi^\top\Phi)^{-1}\Phi^\top\mathbf{y}$.

Now consider a ridge regression problem with the regularization term $\frac{1}{2}\beta\|\mathbf{w}\|_2^2$. We know that the symmetric matrix in the closed-form solution is $\Phi^\top\Phi + \beta\mathbf{I}$. Please (i) derive the eigenvalues and eigenvectors for $\Phi^\top\Phi + \beta\mathbf{I}$ with respect to the eigenvalues and eigenvectors of $\Phi^\top\Phi$ denoted as λ_i and $\mathbf{u}_i$, and then, (ii) for any $\beta > 0$, prove that the matrix $(\Phi^\top\Phi + \beta\mathbf{I})$ is PD.

(c) [**9 points**] In this sub-problem, we use logistic regression to predict the class label $y \in \{-1, +1\}$ instead of $y \in \{0, 1\}$ as in the ordinary logistic regression. Show that maximizing the log-likelihood of logistic regression, $\sum_{n=1}^{N} \log P(y^{(n)}|\mathbf{x}^{(n)})$, is equivalent to minimizing the following loss function:

$$\sum_{n=1}^{N} \log\left(1 + \exp(-y^{(n)} \cdot \mathbf{w}^\top\phi(\mathbf{x}^{(n)}))\right). \tag{4}$$

[Hint: You can expand the log-likelihood as follows: $\log P(y^{(n)}|\mathbf{x}^{(n)}) = \mathbb{I}(y^{(n)} = 1)\log P(y^{(n)} = 1|\mathbf{x}^{(n)}) + \mathbb{I}(y^{(n)} = -1)\log P(y^{(n)} = -1|\mathbf{x}^{(n)})$ then plug in the class posterior probability of the logistic regression model.]

2 [39 points] Linear regression on a polynomial

In this problem, you will implement linear regression on a polynomial. Please have a look at the accompanied starter code `linear_regression.py` and notebook `linear_regression.ipynb` for instructions first. Please note that all the sub-questions without (Autograder) need to be answered in your **writeup**.

Sample data The files `q2xTrain.npy`, `q2xTest.npy`, `q2yTrain.npy` and `q2yTest.npy` specify a linear regression problem for a polynomial. `q2xTrain.npy` represent the inputs ($\mathbf{x}^{(i)} \in \mathbb{R}$) and `q2yTrain.npy` represents the outputs ($y^{(i)} \in \mathbb{R}$) of the training set, with one training example per row.

2.1 GD and SGD

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 (GD)
- Stochastic gradient descent (SGD)

Here, as we seen in the class, the training objective is defined as:

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(\sum_{j=0}^{M} w_j\phi_j(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 = \frac{1}{2}\sum_{i=1}^{N}\left(\mathbf{w}^\top\phi(\mathbf{x}^{(i)}) - y^{(i)}\right)^2 \tag{5}$$

(a) [**12 points**] (Autograder) Implement the GD and SGD optimization methods. For all the implementation details (e.g., function signature, initialization of weight vectors, etc.), follow the instruction given in the code files. Your score for this question will be graded by the correctness of the implementation.

(b) [**3 points**] Please share the plot generated from the section 2(b) of your .ipynb file in your **write-up**, and then compare two optimization methods, GD and SGD. Which one takes less time and which one shows lower test objective $E(\mathbf{w}_{\text{test}})$?

2.2 Over-fitting study

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. To evaluate this behaviour, let's examine the Root-Mean-Square (RMS) Error is defined below. (Note: *we use the RMS error just for evaluation purpose, NOT as a training objective.*)

$$E_{RMS} = \sqrt{2E(\mathbf{w}^*)/N}, \tag{6}$$

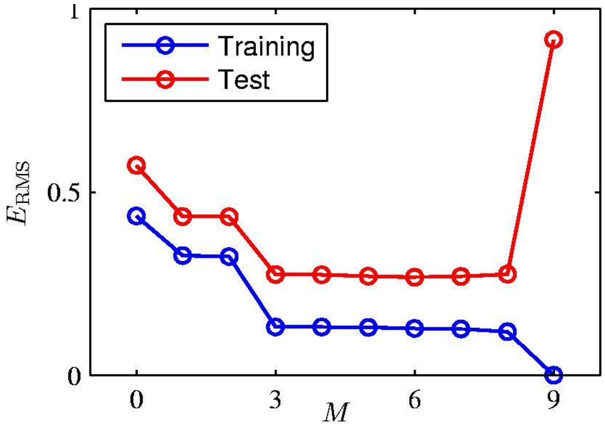

Figure 1: (Sample plot) Overfitting study: Train-Test accuracy.

(a) [**8 points**] (Autograder) In this subquestion, we will use the closed form solution of linear regression (assuming all the condition is met) instead of iterative optimization methods. Implement the closed form solution for the linear regression problem.

(b) [**2 points**] Regenerate the above plot with the provided data. The sample training data can be generated with M-degree polynomial features (for $M = 0 \dots 9$) from `q2xTrain.npy` and `q2yTrain.npy`: we assume the feature vector is $\phi(x^{(i)}) = (1, x^{(i)}, (x^{(i)})^2, \cdots, (x^{(i)})^M)$ for any values of M. For the test curve, use the data in `q2xTest.npy` and `q2yTest.npy`. Note that the trend of your curve is not necessarily the same as the sample plot. Attach your plot to the **write-up**.

(c) [**2 points**] In the **write-up**, please discuss: Which degree polynomial would you say best fits the data? Was there evidence of under/over-fitting the data? Use your generated plots to justify your answer.

2.3 Regularization (Ridge Regression)

Finally, you will explore the role of regularization. Recall the image from lecture that explored the effect of the regularization factor λ:

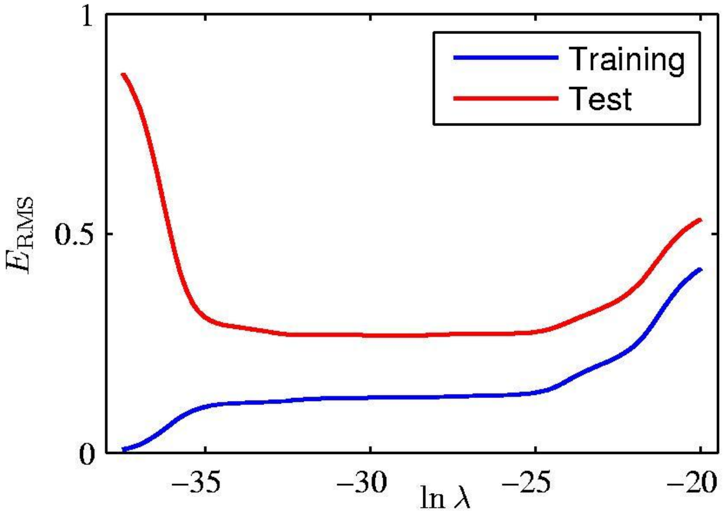

Figure 2: (Sample plot) effects of the regularization factor λ.

(a) **[8 points]** (Autograder) Implement the closed form solution of the ridge regression. Specifically, please use the following regularized objective function:

$$\frac{1}{2}\sum_{i=1}^{N}(\mathbf{w}^\top \phi(\mathbf{x}^{(i)}) - y^{(i)})^2 + \frac{\lambda}{2}||\mathbf{w}||_2^2. \tag{7}$$

for optimizing the parameters **w**.

(b) **[2 points]** For the sample data, our goal is to regenerate the above plot (Figure 2), but with $\lambda \in \{10^{-5}, 10^{-4}, \ldots, 10^{-1}, 10^{0}(=1)\})$. Please first compute the E_{RMS} over the training data specified in `q2xTrain.npy` and `q2yTrain.npy` with λ and then measure the test error with `q2xTest.npy` and `q2yTest.npy`. Please attach your (unregularized) E_{RMS} plot of both the training data and test data, obtained from 2(g), to the **write-up**. Note that the trend of your curve is not necessarily the same as the sample plot.

(c) **[2 points]** Discuss: Which λ value seemed to work the best for a ninth degree polynomial ($M = 9$)? Use your generated plots to justify your answer. Please provide your answer in your **write-up**.

3 [30 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

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

where $r^{(i)} \in \mathbb{R}$ is the "local" weight for the sample $(\mathbf{x}^{(i)}, y^{(i)})$. In class, we worked on its special case where all the weights (the $r^{(i)}$'s) are all equal. 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 data points in the training data.]

(a) **[3 points]** Show that $E_D(\mathbf{w})$ can also be written as

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

for an appropriate diagonal matrix R, and where $X \in \mathbb{R}^{D\times N}$ is a matrix whose i-th column is $x^{(i)} \in \mathbb{R}^{D\times 1}$ and $\mathbf{y} \in \mathbb{R}^{N\times 1}$ is the vector whose i-th entry is $y^{(i)}$. Here, please note that in locally weighted linear regression in this homework, we use raw data directly withing mapping to high-dimensional features (i.e., each input is represented as a D-dimensional input vector, not M-dimensional feature vector), so that's why we use notation for X instead of Φ. State clearly what the R matrix is.

(b) **[7 points]** As we already saw in the class, if all the $r^{(i)}$'s equal 1, then the normal equation for $\mathbf{w} \in \mathbb{R}^{D\times 1}$ becomes

$$XX^\top \mathbf{w} = X\mathbf{y},$$

and the value of $\mathbf{w}^*$ that minimizes $E_D(\mathbf{w})$ is given by $(XX^\top)^{-1}X\mathbf{y}$.[2] Now, by finding the derivative $\nabla_{\mathbf{w}} E_D(\mathbf{w})$ from (a) and setting that to zero, generalize the normal equation and the closed form solution to this locally-weighted setting, and give the new value of $\mathbf{w}^*$ that minimizes $E_D(w)$ in a closed form as a function of X, R and $\mathbf{y}$. (*hint*: $\nabla_{\mathbf{w}}(RX^\top \mathbf{w}) = XR^\top = XR$)

(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

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

I.e., $y^{(i)}$ is a Gaussian random variable with mean $\mathbf{w}^\top \mathbf{x}^{(i)}$ and variance $(\sigma^{(i)})^2$ (where the $\sigma^{(i)}$'s are fixed, known constants). Show that finding the maximum likelihood estimate (MLE) of $\mathbf{w}$ reduces to solving a weighted linear regression problem $E_D(\mathbf{w})$. State clearly what the $r^{(i)}$'s are in terms of the $\sigma^{(i)}$'s.

[2] We presented the row-major ($X \in \mathbb{R}^{N\times D}$) version of $\mathbf{w}^*$ as $(X^\top X)^{-1}X^\top \mathbf{y}$ in Lecture 2. For practice, we are using the column-major ($X \in \mathbb{R}^{D\times N}$) version in this problem.

(d) **[12 points, Programming Assignment]** In the following, you will use the files `q3x.npy` which contains the inputs $(\mathbf{x}^{(i)}) \in \mathbb{R}^{1\times D}$ and `q3y.npy` which contains the outputs (target) $(y^{(i)})$ for a linear regression problem, with one training example per row.[3]

i. **[8 points]** (Autograder) Implement the closed-form solution for locally weighted linear regression (see the accompanied code).

ii. **[2 points]** Use the implemented locally weighted linear regression solver on this dataset (using the weighted normal equations you derived in part (b)), and plot 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

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

with a bandwidth parameter $\tau \in \{0.1, 0.3, 0.8, 2, 10\}$. Please attach the plots generated in 3.(d).ii of your notebook to **write-up**.

iii. **[2 points]** In the **write-up**, discuss and comment **briefly** on what happens to the fit when τ is too small or too large.

[3] Please note that we are using row-major version for implementation, as described in the starter package.