UNIVERSITY OF MICHIGAN
Department of Electrical Engineering and Computer Science
EECS 445 — Introduction to Machine Learning
Fall 2022

Homework 2 (50 points)
Due: Wednesday, October 12th at 10 pm

Submission: Please upload your completed assignment to Gradescope. Your submission can be either handwritten or typed. Assign all pages (including pages containing code) to their respective questions. Incorrectly assigned pages will not be graded.

1 Soft-Margin SVM Dual Derivation [12 pts]

Consider the primal formulation of a soft-margin SVM with regularization and offset, where $C \geq 0$ is a regularization hyperparameter.

$$\begin{aligned} \underset{\bar{\theta},b,\bar{\xi}}{\text{minimize}} \quad & \frac{1}{2}\|\bar{\theta}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y^{(i)}(\bar{\theta}\cdot\bar{x}^{(i)} + b) \geq 1 - \xi_i, \\ & \xi_i \geq 0\ , \forall i = 1, ..., n. \end{aligned}$$

From this, we can obtain the dual formulation.

$$\begin{aligned} \underset{\bar{\alpha}}{\text{maximize}} \quad & \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y^{(i)}y^{(j)}\bar{x}^{(i)}\cdot\bar{x}^{(j)} \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y^{(i)} = 0, \\ & 0 \leq \alpha_i \leq C\ , \forall i = 1, ..., n. \end{aligned}$$

(a) (7 pt) Start with the primal form of a soft-margin SVM shown above and **derive** the dual formulation shown above.

(i) (3 pt) To start, state the Lagrangian of the primal. Hint: Remember to include separate terms corresponding to each of the constraints in the Lagrangian.

(ii) (4 pt) Next, swap the order of the optimization and find the optimal value of $\bar{\theta}$ as well as the constraints imposed by the optimal values of b and $\bar{\xi}$. You may assume that strong duality holds for this problem and that the duality gap is zero.

For the remaining problems, assume we trained a soft-margin SVM classifier with $C = 1$ on a dataset consisting of $n = 6$ points and learned the parameters listed in the table below.

i	$\bar{x}^{(i)}$	$y^{(i)}$	α_i
1	[-1.5 , -2.1]^T	1	0
2	[-1.9, 0]^T	1	0.322
3	[-3.7 , -2.5]^T	1	0
4	[0.5 , 0.75]^T	-1	0.497
5	[2 , 3.3]^T	-1	0
6	[1.6 , -2.5]^T	1	0.175

(b) (2 pt) Based on the above dataset, state which points are support vectors.

(c) (3 pt) Use the same dataset to find the optimal value of b and $\bar{\theta}$ and use them to **classify** the following points: [-4,1] and [2,5].

2 Linear Regression - Optimization Methods [28 pts]

Recall that for linear regression, the empirical risk with **squared loss** is:

$$R_N(\bar{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\frac{(y^{(i)} - \bar{\theta}\cdot\bar{x}^{(i)})^2}{2}$$

For this question, you are provided with skeleton code `q2_linear_regression.py`, and data files, `q2_train.csv` and `q2_validation.csv`, with which we will train and test our model. We are going to map the input dataset to a polynomial feature space and then train a linear regression model in the feature space. In each csv file, the first column specifies the output ($y^{(i)} \in \mathbb{R}$), and the second column specifies the input ($x^{(i)} \in \mathbb{R}$).

You may find it useful to generate 2D-plots for the training data and the output of regression functions.

2.1 Comparing Optimization Algorithms (18 pts)

(a) (2 pt) First, implement the function `generate_polynomial_features(X,M)` according to the specification in the skeleton code. This function transforms each example data point $x^{(i)}$ into an $M+1$ dimensional feature vector, $\phi(x^{(i)})$. In part 2.1 you will explore a solution based on a first degree polynomial. However, the function should be general enough to handle any $M \geq 0$ for latter parts. **Include a screenshot (or equivalent) of your implemented function as your solution to this question.**

(b) You will now implement and use three different optimization methods to find the coefficients θ_1 and θ_0 (slope and intercept, respectively) that minimize the **squared loss** for a first degree polynomial i.e., $\hat{y} = \theta_1 x + \theta_0$.

(i) **Include a screenshot (or equivalent) of your implemented functions as your solution to this question.** (6 pt)

(A) Implement `ls_gradient_descent(X,y)`.

(B) Implement `ls_stochastic_gradient_descent(X,y)`.

(C) Implement `closed_form_optimization(X,y)` (ignore `reg_param` for now).

Important note:

- For the family of gradient descent algorithms, check for convergence after each epoch (one pass through the entire training set).
- For SGD, do NOT shuffle the points after each epoch.
- The `prev_loss` and `new_loss` in the skeleton code refer to empirical risk for linear regression as defined above, which are used for convergence check. You should implement and use `calculate_empirical_risk` to avoid code duplication.
- Use the convergence criteria specified in the code: the algorithm should terminate when there is either marginal improvement in the loss ($<= 10^{-10}$) between two convergence checks, or after $1,000,000$ iterations — whichever happens first. One iteration here is defined as one update to $\bar{\theta}$.
- For the closed form solution, if you encounter numerical stability issues in the calculation of matrix inverse, consider using the pseudoinverse operation `np.linalg.pinv()`.

(ii) (4 pt) In function `q2_1`, make use of the functions you implemented above to find the coefficients θ_1 and θ_0 that minimize the squared loss for a first degree polynomial ($M = 1$). For GD and SGD, you need to specify a learning rate (or step size) η. Try different values of $\eta \in \{10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}\}$. Report your results in the following table. Here "# iterations" refers to the number of $\bar{\theta}$ updates. We provided a line for each algorithm for you to check your implementations.

Algorithm	η	θ_0	θ_1	# iterations	Runtime (s)
GD	10^{-4}	0.2970493	-0.10938172	459712	
GD	10^{-3}				
GD	10^{-2}				
GD	10^{-1}				
SGD	10^{-4}	0.30364222	-0.12053361	709100	
SGD	10^{-3}				
SGD	10^{-2}				
SGD	10^{-1}				
Closed form	-			-	

(c) (4 pt) Compare GD vs SGD: specifically, comment on runtime, number of iterations, and resulting coefficients at convergence.

(d) (2 pt) In this particular case, how does the runtime of the closed form solution compare to SGD? Which learning rate used in SGD produces the coefficients closest to the closed form solution?

2.2 Overfitting & Regularization (10 pt)

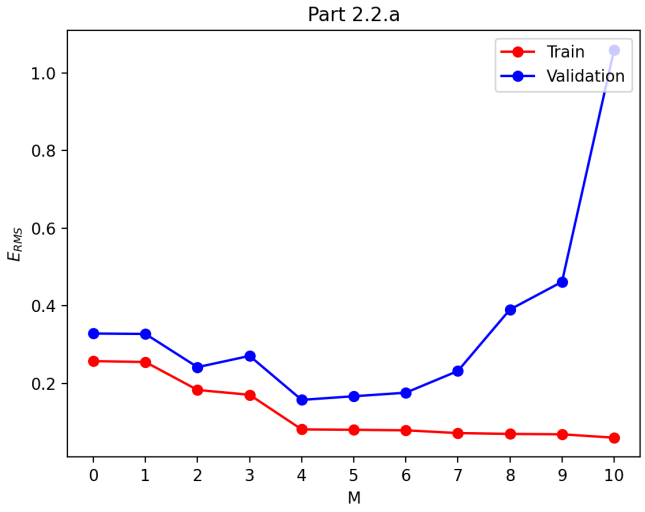
(a) Next, you will investigate the problem of overfitting. Here, we observe overfitting as we increase the degree of the polynomial, M. We evaluate solutions using Root-Mean-Square (RMS) Error, defined as

$$E_{RMS} = \sqrt{E(\bar{\theta})/N},$$

where

$$E(\bar{\theta}) = \sum_{i=1}^{N} (\bar{\theta} \cdot \phi(x^{(i)}) - y^{(i)})^2$$

(i) (2 pt) Implement the function `calculate_RMS_Error(X,y,theta)` according to the specification given in the skeleton code. In function `q2_2`, use the closed form solution from part 2.1 to find the θ that minimize the **squared loss** for an M^{th} degree polynomial (for $M = 0 \ldots 10$) for the training data. Then calculate the RMS Error, for each setting of M, on the training data and on the validation data (separately). Plot E_{rms} against M for both training data and validation data (in the same graph) and include it in your solution write-up. Your graph should look similar to the following:



(ii) (2 pt) Which degree polynomial would you say best fits the data? Is there any evidence of underfitting / overfitting? Use your generated chart to defend your answer.

(b) Now, you will explore the role of regularization.

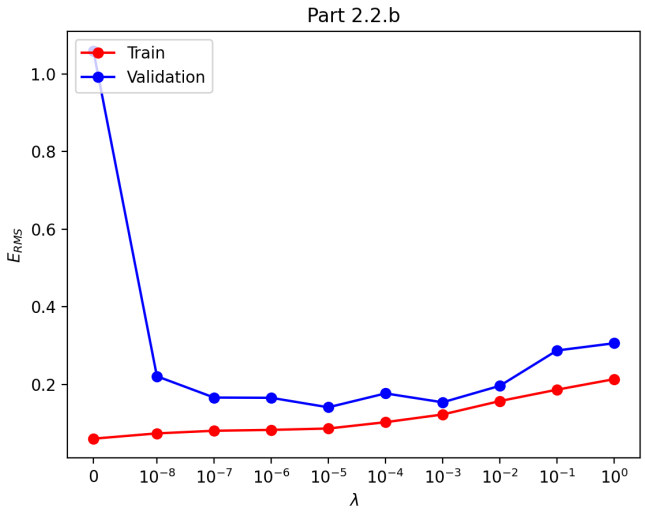
(i) (2 pt) Modify your implementation of `ls_closed_form_optimization(X,y,reg_param=0)` from part 2.1 to incorporate L2-regularization. Specifically, use the closed form solution associated with the following regularized objective function:

$$\frac{1}{N} \sum_{i=1}^{N} \frac{(\bar{\theta} \cdot \phi(x^{(i)}) - y^{(i)})^2}{2} + \frac{\lambda}{2} ||\bar{\theta}||_2^2$$

for optimizing the parameters $\bar{\theta}$. **Include a screenshot (or equivalent) of your implemented functions as your solution to this question.**

(ii) (2 pt) Use your function from part (b.i) to find the θ that minimize the objective function for a tenth degree polynomial ($M = 10$) given regularization factor λ (for $\lambda \in \{0, 10^{-8}, 10^{-7}, \ldots, 10^{-1}, 10^{0}\}$) for the training data specified in `q2_train.csv`. Now use these coefficients to calculate the RMS Error on both the training data and validation data as a function of λ and plot E_{RMS} against $\lambda \in \{0, 10^{-8}, 10^{-7}, \ldots, 10^{-1}, 10^{0}\}$ and include it in your solution write-up..

See the plot below. Depending on whether you choose to keep the factor $\frac{1}{n}$ in your calculations, you will get different plots. We will accept either version.



(iii) (2 pt) Which value of λ appears to work the best?

3 Decision Trees [10 pts]

(a) (2 pt) You are given a training dataset with d features and labeled examples $\bar{x}^{(1)}, \ldots, \bar{x}^{(n)}$. Based on these data, you build a decision tree D_1 by selecting which feature to split on at each iteration based on information gain. You then take one of the labeled examples, add a copy of it to the training set, and rerun your learning algorithm to create D_2. Will D_1 and D_2 necessarily be identical decision trees? **Provide justification.**

(b) (2 pt) Consider learning a decision tree with noise-free data (that is, the training set is generated from the underlying distribution, and both the features and labels are completely accurate). Suppose you start with a feature at the root that does not maximize information gain, can you still find a tree that fits the data exactly?**Provide justification.**

(c) The Univeristy of Michigan Museum of Natural History is hunting dinosaur fossils to expand their collection! You are tasked to predict the locations at which we will find a new dinosaur skeleton.

In an initial feasibility study, you collect data about 9 locations worldwide regarding humidity, temperature, altitude, and whether or not skeletons has been found:

Humidity	Temperature	Altitude	Dinosaur
high	high	low	NO
low	medium	medium	NO
medium	high	medium	YES
low	medium	high	YES
high	medium	medium	YES
medium	high	low	NO
high	medium	low	NO
high	low	high	YES
medium	low	low	YES

(i) (4 pt) Using entropy method to measure uncertainty, which features will the Decision Tree learning algorithm choose as the root? (Note that this decision tree will split into three branches, once for each feature value) **Provide justification through your work.**

(ii) (2 pt) **Construct** a decision tree for this dataset that achieves zero training error with the minimal number of leaf nodes.

REMEMBER to submit your completed assignment to Gradescope by 10:00pm ET on Wednesday, Oct. 12. Assign all pages (including pages containing code) to their respective questions. Incorrectly assigned pages will not be graded.