

HOMework 2

LINEAR REGRESSION, LOGISTIC REGRESSION AND MLE/MAP¹

CMU 10-701: INTRODUCTION TO MACHINE LEARNING (SPRING 2026)

<https://piazza.com/cmu/spring2026/10701>

OUT: Friday, Jan. 30, 2026

DUE: Tuesday, Feb. 10, 2026, 11:59pm

START HERE: Instructions

- **Collaboration policy:** Collaboration on solving the homework is allowed, after you have thought about the problems on your own. It is also OK to get clarification (but not solutions) from books or online resources, again after you have thought about the problems on your own. There are two requirements: first, cite your collaborators fully and completely (e.g., “Jane explained to me what is asked in Question 2.1”). Second, write your solution *independently*: close the book and all of your notes, and send collaborators out of the room, so that the solution comes from you only. See the Academic Integrity Section in our course syllabus for more information: <https://www.cs.cmu.edu/~10701-s26#Syllabus>
- **Late Submission Policy:** See the late homework policy here: <https://www.cs.cmu.edu/~10701-s26/#Syllabus>
- **Submitting your work:**
 - **Gradescope:** There will be two submission slots for this homework on Gradescope: Written and Programming.
For the written problems such as short answer, multiple choice, derivations, proofs, or plots, we will be using the written submission slot. Please use the provided template. The best way to format your homework is by using the Latex template released in the handout and writing your solutions in Latex. However submissions can be handwritten onto the template, but should be labeled and clearly legible. If your writing is not legible, you will not be awarded marks. Each derivation/proof should be completed in the boxes provided below the question, **you should not move or change the sizes of these boxes** as Gradescope is expecting your solved homework PDF to match the template on Gradescope. If you find you need more space than the box provides you should consider cutting your solution down to its relevant parts, if you see no way to do this, please add an additional page at the end of the homework and guide us there with a ‘See page xx for the rest of the solution’.
You are also required to upload your code, which you wrote to solve the final question of this homework, to the Programming submission slot. Your code may be run by TAs so please make sure it is in a workable state.
Regrade requests can be made after the homework grades are released, however this gives the TA the opportunity to regrade your entire paper, meaning if additional mistakes are found then points will be deducted.

For multiple choice or select all that apply questions, shade in the box or circle in the template document corresponding to the correct answer(s) for each of the questions. For L^AT_EX users, use ■ and ● for shaded boxes and circles, and don’t change anything else. If an answer box is included for showing work, **you must show your work!**

¹Compiled on Friday 30th January, 2026 at 19:17

1 Regularized Linear Regression [5 Points]

1. [5 Points] Consider the following linear regression model: for each data point in $\mathcal{D} = \{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^n$,

$$y^{(i)} = \mathbf{w}^T \mathbf{x}^{(i)} + \epsilon \text{ where } y^{(i)}, \epsilon \in \mathbb{R} \text{ and } \mathbf{w}, \mathbf{x}^{(i)} \in \mathbb{R}^{d+1}$$

In matrix notation, we can express this linear relationship for all data points as:

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\epsilon} \text{ where } \mathbf{y}, \boldsymbol{\epsilon} \in \mathbb{R}^n, \mathbf{X} \in \mathbb{R}^{n \times (d+1)}, \text{ and } \mathbf{w} \in \mathbb{R}^{d+1}$$

Assuming the residuals are normal and i.i.d. ($\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \mathbf{I})$), we can write:

$$\mathbf{y} | \mathbf{X}, \mathbf{w} \sim \mathcal{N}(\mathbf{X}\mathbf{w}, \sigma^2 \mathbf{I})$$

Now assume that we have a Gaussian prior on \mathbf{w} :

$$\mathbf{w} \sim \mathcal{N}\left(0, \frac{\sigma^2}{\lambda} \mathbf{I}\right)$$

for some fixed $\lambda > 0$. Recall that in ridge regression, the optimal parameter vector is given by:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2 + \lambda \|\mathbf{w}\|_2^2.$$

Show that the solution to the MAP estimate $\mathbf{w}_{\text{MAP}}^*$ in this setting is the same as the one obtained from ridge regression.

(Hint: Start by writing down the expression for the negative log posterior and show that minimizing it gives the same solution as minimizing the OLS with Ridge regression.)

2 Convexity of Logistic Regression [18 points]

Consider a binary classification problem where the goal is to predict a label $y \in \{0, 1\}$, given an input $\mathbf{x} \in \mathbb{R}^d$. A method that you can use for this task is *logistic regression*. In logistic regression, we model the log-odds as an affine function of the data and find weights to maximize the likelihood of our data under the resulting model.

Recall that an *affine function* f takes the form $f(x) = w^\top x + c$ with $c \in \mathbb{R}$ and $w, x \in \mathbb{R}^n$. In other words, it is a linear function composed with a translation.

2.1 Convex Optimization

Recall that the log-likelihood for a logistic regression model can be written as

$$\mathcal{L}(w) = \log P(y|\mathbf{X}, w) = \sum_{i=1}^n [y_i w^\top x_i - \log(1 + \exp(w^\top x_i))].$$

Our goal is to find the weight vector w that maximizes this likelihood. In this question, you will prove that \mathcal{L} is indeed a concave function (and hence, the negative conditional log likelihood is a convex function).

1. [3 points] A real-valued function $f : S \rightarrow \mathcal{R}$ defined on a convex set S , is said to be *convex* if

$$f(tx_1 + (1-t)x_2) \leq tf(x_1) + (1-t)f(x_2), \forall x_1, x_2 \in S, \forall t \in [0, 1].$$

Show that a linear combination of n convex functions, f_1, f_2, \dots, f_n , $\sum_{i=1}^n a_i f_i(x)$ is also a convex function $\forall a_i \in \mathbb{R}^+$.

2. **[1 point]** Show that a linear combination of n concave functions, f_1, f_2, \dots, f_n , $\sum_{i=1}^n a_i f_i(x)$ is also a concave function $\forall a_i \in \mathbb{R}^+$. Recall that if a function $f(x)$ is convex, then $-f(x)$ is concave. (You can use the result from part (1))

3. **[3 points]** Another property of twice differentiable convex functions is that the second derivative is non-negative. Using this property, show that $f(x) = \log(1 + \exp x)$ is a convex function. Note that this property is both sufficient and necessary. i.e. (if $f''(x)$ exists, then $f''(x) \geq 0 \iff f$ is convex)

4. **[3 points]** Let $f_i : \mathcal{S} \rightarrow \mathcal{R}$ for $i = 1, \dots, n$ be a set of convex functions. Is $f(x) = \max_i f_i(x)$ also convex? If yes, prove it. If not, provide a counterexample.

5. **[8 points]** Show that the log likelihood of *Logistic Regression* is a concave function. You may use the fact that if f and g are both convex, twice differentiable and g is non-decreasing, then $g \circ f$ is convex.

6. **[2 points]** Unfortunately, for this model, we cannot derive a closed-form solution with MLE. However, now that we have proven that \mathcal{L} is concave, are there other methods we can use to solve for w ? Is any method guaranteed to find the “best” solution, where “best” means achieving the lowest loss? In a few short words, explain why or why not.

3 Linear Regression for Time Series [30 Points]

In this problem you will explore fitting a linear regression to predict time series data using OLS and stochastic gradient descent.

3.1 Data Processing

In this problem, you will be using a temperature dataset which can be found in the file `temperature.csv`. This file contains timestamps (column labeled 'Date Time') at 30 minute intervals and corresponding temperature measurements (column labeled 'T (degC)') over eight years of data collection.

Your first task is to split this data into a train set and a test set. We want to use the first 6 years of data as our training set and the last 2 years as our test set. Since the dataset includes one measurement every 30 minutes (i.e. 2 measurements every hour), the training set should contain the first $2 * 24 * 365 * 6 = 105142$ samples and the test set should contain the last $2 * 24 * 365 * 2 = 35040$ samples (i.e. training and test comprises the entire dataset).

Our first task will be to train a model that predicts the temperature at a given time based on the measurements at the previous $D=10$ timesteps, so we use the value 10 for D below to set us up for this.

Concretely, we can write our training portion of the dataset as $X_{\text{train}} = \{x_1, x_2, \dots, x_T\}$ where each x_i is one temperature measurement and $T = 105142$ is the total number of training samples. In this setting, we will use the temperatures $x_1, x_2, x_3, \dots, x_D$ to predict the value at x_{D+1} ; temperatures x_2, x_3, \dots, x_{D+1} to predict the temperature at x_{D+2} ; and so on so that in general we use $x_i, x_{i+1}, \dots, x_{i+(D-1)}$ to predict the value of x_{i+D} .

You need to reformat the training portion of the dataset to follow this framework. You should create an `X_train` matrix that has D columns (i.e. the D consecutive timestamps) and $T - D$ rows. Note that data will be repeated in this matrix, since each row is shifted by only one timestep from the previous row and will therefore contain $D - 1$ of the same temperature values. You should also create a `y_train` vector that contains the target values we want to predict, i.e. $[x_{D+1}, x_{D+2}, \dots, x_T]$.

Similarly, we can define our test portion of our dataset as $X_{\text{test}} = \{x_{T+1}, x_{T+2}, \dots, x_{T+C}\}$ where C is the total number of the test samples. We will create an `X_test` matrix and a `y_test` vector following the same procedure as for the training dataset.

We have now created normalized datasets with 10 features and can use these 10 features to predict the target corresponding to each row.

3.2 Predicting Temperatures using Linear Regression

For this question, please submit all code you wrote in a single, self-contained file titled `time_series.py` on Gradescope. Your work in 3.2.1, 3.2.2 and 3.2.4 will be autograded on Gradescope. However, you should still fill in the answers again in the textboxes provided. All remaining questions will be manually graded.

1. **[4 points]** Fit an OLS linear regression model using `X_train` and `y_train` to find the OLS solution. Report the following values:
 - (a) the weights you learned (including the bias or intercept weight)
 - (b) the time taken to fit the model
 - (c) the MSE on `X_test`

Your OLS function will be autograded, but please also report the values in the textbox below. As a reminder, you should only use `numpy`, `time`, `copy` (optional), `math` (optional) in your implementations.

2. **[8 points]** Repeat the previous question for $D = \{50, 100, 500\}$. For these models, you should just report the weights on the first 10 features and the bias weight, along with the time required to fit each model and the MSE on `X_test`. Your OLS function will be autograded, but please also report the values in the textbox below.

3. [3 points] Using the times taken to fit the models for $D = 10, 50, 100$, and 500, estimate the time it would take to fit a model using features from an entire year's worth of data (i.e. $D = 2 * 24 * 365 = 17520$). Report how you estimated the time (i.e. what function did you fit?) and your time estimate in minutes.

4. [10 points] Based on our result from the previous part, we may conclude that using OLS will result in a very high computational cost. As an alternative, you will now learn the weights \mathbf{w} and bias b using **stochastic gradient descent**. We will use train and test datasets created for $D = 17520$, i.e. using the data from an entire year.

Your implementation of SGD should use a learning rate of $\eta = 1e-10$ and run for 20 epochs. Initialize your weights to be uniformly distributed, with each value set to $\frac{1}{D}$ and your bias to be 1. Additionally, while normally you would shuffle or permute the training data points in each epoch of SGD, for this assignment, you should loop through the training dataset in chronological order (i.e., the original ordering of the dataset) without randomly shuffling the data points.

As a reminder, you should only use `numpy` to implement SGD from scratch. After using SGD to train a LR model that uses a year's worth of data for 20 epochs, please report the following values:

- (a) Train MSE
- (b) Test MSE
- (c) Total training time
- (d) First 10 items in your final weight vector.

Your SGD function will be autograded, but please also report the values in the textbox below.

5. [5 points] Now we can examine the weights learned by SGD. What takeaways can you observe? Are any features particularly informative in predicting the targets? Please give a 2-3 sentence response describing your observations. **Hint:** consider which features have the largest weights; what observations do those features correspond to?

4 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment?
(b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

2. (a) Did you give any help whatsoever to anyone in solving this assignment?
(b) If you answered 'yes', give full details (e.g. "I pointed Joe Smith to section 2.3 since he didn't know how to proceed with Question 2")

3. (a) Did you find or come across code that implements any part of this assignment?
(b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).