Homework 2

Linear Regression, Logistic Regression, MLE/MAP, Naïve Bayes

UTK COSC 522: Machine Learning (Fall 2025)
OUT: Sunday, Sep 21st, 2025

DUE: Monday, Oct 13th, 2025, 11:59pm, **Bonus Point Deadline: Oct 8th, 2025, 11:59pm**

Homework Instructions

- **Collaboration policy: Taking Responsibility for Your Career**

  The skills you learn in this course are tools for your future, and this policy is designed to help you sharpen them. You are responsible for your own learning and career, and this framework ensures you get the most from every assignment.

  We encourage collaboration, but it must be done *right*. *First*, make a genuine effort to solve problems on your own. This independent effort is where the most critical learning occurs. *Afterward*, you may discuss strategies with peers or consult resources to clarify concepts—the goal is to deepen your own understanding, not to simply get an answer.

  Finally, like any professional, you must stand behind your own work. Your submitted solution must be written entirely by you, from scratch, without collaborators present. This proves you have truly mastered the material. To maintain academic and professional integrity, you must also cite any person or resource that contributed to your understanding along the way.

- **Late Submission Policy:** See the late homework policy here.

- **Submitting your work:**

  – **Canvas:** For this homework, you will submit a **single .zip file** to Canvas. Please name your file in the format **YourName_NetID_HW2.zip**. This zip file must contain exactly two files: your written solutions in a single PDF and your Python code.

  Your **single PDF file** must contain your answers to all questions, including written problems (proofs, short answers, plots) and the empirical questions from the programming section. You must use the provided homework template. Using the LaTeX version to typeset is strongly recommended, but you may also submit a scan of the template with legible handwriting; **illegible answers will not be graded**. Please complete all answers within the provided boxes.

  The second file in your zip archive must be your *single* completed programming file, named exactly as **time series.py**. Please ensure your code is runnable, as we may execute it to verify your results.

Regrade requests can be made after homework grades are released. Please be aware that a regrade request allows the TA to review your entire assignment, which may result in points being deducted if new errors are found.

1 Regularized Linear Regression [10 Points]

1. [**10 Points**] Consider the following linear regression model: for each data point in $\mathcal{D} = \{(\boldsymbol{x}^{(i)}, y^{(i)})\}_{i=1}^{n}$,

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

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

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

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

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

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

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

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

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

Show that the solution to the MAP estimate $\boldsymbol{w}^*_{\text{MAP}}$ 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.)

$$w^*_{MAP} = \operatorname{argmax}_w \mathcal{P}(w|y, X)$$

$$w^*_{MAP} = \operatorname{argmax}_w \log \mathcal{P}(y|X, w) + \log \mathcal{P}(w)$$

$$w^*_{MAP} = \operatorname{argmax}_w \log \mathcal{N}(y|Xw, \sigma^2 I) + \log \mathcal{N}(w|0, \tau^2 I)$$

$$w^*_{MAP} = \operatorname{argmax}_w \frac{-1}{2\sigma^2} \|y - Xw\|_2^2 + \frac{-1}{2\tau^2} \|w\|_2^2$$

$$w^*_{MAP} = \operatorname{argmin}_w \frac{1}{2\sigma^2} \|y - Xw\|_2^2 + \frac{1}{2\tau^2} \|w\|_2^2$$

$$w^*_{MAP} = \operatorname{argmin}_w \|y - Xw\|_2^2 + \frac{2\sigma^2}{2\tau^2} \|w\|_2^2$$

2 Convexity of Logistic Regression [28 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. Unfortunately, for this model, we cannot derive a closed-form solution with MLE. An alternative way to solve for w is to use gradient *ascent*, and update w step by step towards the optimal w. But we know gradient ascent will converge to the optimal solution w that maximizes the conditional log likelihood $\mathcal{L}$ when $\mathcal{L}$ is concave. 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. [**5 points**] A real-valued function $f : S \to \mathcal{R}$ defined on a convex set S, is said to be *convex* if

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

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

$$f(x) = \sum_{i=1}^{n} a_i f_i(x)$$

$$f(tx_1 + (1-t)x_2) = \sum_{i=1}^{n} a_i f_i(tx_1 + (1-t)x_2)$$

$$f_i(tx_1 + (1-t)x_2) \le tf_i(x_1) + (1-t)f_i(x_2)$$

$$a_i f_i(tx_1 + (1-t)x_2) \le a_i[tf_i(x_1) + (1-t)f_i(x_2)]$$

$$\sum_{i=1}^{n} a_i f_i(tx_1 + (1-t)x_2) \le \sum_{i=1}^{n} [ta_i f_i(x_1) + (1-t)a_i f_i(x_2)]$$

$$\sum_{i=1}^{n} a_i f_i(tx_1 + (1-t)x_2) \le t\sum_{i=1}^{n} a_i f_i(x_1) + (1-t)\sum_{i=1}^{n} a_i f_i(x_2)$$

$$f(tx_1 + (1-t)x_2) \le tf(x_1) + (1-t)f(x_2)$$

2. [**3 point**] Show that a linear combination of n concave functions, $f_1, f_2, ..., f_n$, $\sum_{i=1}^{n} a_i f_i(x)$ is also a concave function $\forall a_i \in R^+$. Recall that if a function $f(x)$ is convex, then $-f(x)$ is concave. (You can use the result from part (1))

If we are given the fact that a negated convex function is concave we can simply use the result from the previous part to say that if we negate each given concave function we can prove the above result. By transforming the concave functions in convex ones we save time and just have to prove the a linear combination of convex:w functions (with positive scalar coefficients) is also convex.

3. [**5 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)

$$0 \leq \frac{d^2}{d^2x} f(x)$$

$$0 \leq \frac{d^2}{d^2x} \log{(1 + \exp x)}$$

$$0 \leq \frac{d}{dx} \frac{\exp x}{1 + \exp x}$$

$$0 \leq \frac{\exp x}{(1 + \exp x)^2}$$

The above function always has a positive denominator, and the function exp(x) has a range of $[0,\infty)$, therefore it is always greater than 0.

4. **[5 points]** Let $f_i : \mathcal{S} \to \mathcal{R}$ for $i = 1, \ldots, n$ 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.

$$\text{let} f_1(x) = \max_i f_i(x)$$

$$f_i(tx_1 + (1-t)x_2) \leq tf_i(x_1) + (1-t)f_i(x_2)$$

$$\max_i f_i(tx_1 + (1-t)x_2) \leq \max_i (tf_i(x_1) + (1-t)f_i(x_2)$$

$$\max_i (tf_i(x_1) + (1-t)f_i(x_2)) \leq t \max_i f_i(x_1) + (1-t) \max_i f_i(x_2)$$

Given the following, and definitions for a,b,c

$$\max_i (a_i) \leq t \max_i (b_i) + (1-t) \max_i (c_i)$$

$$a_i = tf_i(x_1) + (1-t)f_i(x_2), b_i = f_i(x_1), c_i = f_i(x_2)$$

5. **[10 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.

Log-likelihood function: $L(w) = \sum_{i=1}^{n} [y_i w^T x_i - log(1 + \exp(w^T x_i))]$ The first portion of this summation ($y_i w^T x_i$) is linear, meaning it is both convex and concave, and therefore will not change the concave/convex nature of the rest of the function.
We're more interested in the second part $-log(1 + \exp(w^T x_i))$.
This is the composition of a log and an exp.
The first derivative of this function is the sigmoid function, meaning the second derivative is $\sigma(x)(1 - \sigma(x)$. We also know that the sigmoid function is $\geq$ 0 for all input. Which means that it is a convex function, and we also know that the negation of a convex function is concave.
Now returning back to the entire expression (ignoring the summation briefly), $y_i w^T x_i - log(1 + exp(w^T x_i$, we see that we are simply taking the difference of a linear function and a concave function, which results in a concave function.
We also know (from previously) that the summation of many concave functions still produces a concave function.
Therefore the log likelihood of logistic regression is a concave function.

3 Naïve Bayes [32 Points]

Let $X = (x_1, x_2, .., .x_d)$ denote a set of features and $y \in \{0, 1\}$ denote a binary label. Recall that naïve Bayes models the conditional label distribution $P(y \mid X)$ via the conditional distribution of features given the label $P(X \mid y)$:

$$P(y \mid X) \propto P(X \mid y)P(y)$$

1. **Multinomial Naïve Bayes** Suppose that each feature x_i takes values in the set $\{1, 2, ..., K\}$. Further, suppose that the label distribution is Bernoulli, and the feature distribution conditioned on the label is multinomial.

   (a) [**3 points**] What is the total number of parameters of the model under the naïve Bayes assumption? For full credit you must show your work.

   We would have 1 parameter for the prior, θ, which indicates the $P(y = 1)$.
   Then we would need to have a parameter indicating the "weight" of each feature given the label because we are predicting the probability of the features given a label. Therefore we would need 2 times the number of features (d) times the number of discrete values (k-1).
   Resulting in $2 \times d \times (K - 1)$ parameters.
   Totalling in $1 + 2 \times d \times (K - 1)$

   (b) [**3 points**] What is the total number of parameters of the model without the naïve Bayes assumption? For full credit you must show your work.

   The naive Bayes assumption is us assuming that each of the features is independent, therefore without the assumption we say that the features are not independent. We would therefore need to model $P(x_1, x_2, ..., x_d|y = 1)$, which requires $K^d - 1$ parameters. Each feature can be one of K values, with d total feature columns, and the last one is simply the remaining value to sum up to 1. We once again need a parameter for our prior, and each of the feature probabilities is needed for both labels, meaning our final count comes to $1 + 2 \times (K^d - 1)$

   (c) [**6 points**] Suppose we change the set of values that y takes, so that $y \in \{0, 1, ..., M-1\}$. How would your answers change in both cases (with and without the naïve Bayes assumption)? For full credit you must show your work.

   With the naive Bayes assumption: we would multiply the $d(K - 1)$ term by M instead of 2, and we would now need $M - 1$ priors. Resulting in $(M - 1) + Md(K - 1)$.
   Without the naive Bayes assumption: we would multiply the $(K^d - 1)$ term by M instead of 2, and we would again need $M - 1$ priors. Resulting in $(M - 1) + M(K^d - 1)$

2. **[10 points] Gaussian Naïve Bayes** Now suppose each feature is real-valued, with $x_i \in \mathbb{R}$, and $P(x_i \mid y = c) \sim \mathcal{N}(\mu_{i,c}, 1)$ for $i = 1, 2, ..., d$ and $c = 0, 1$. Again, suppose that the label distribution is Bernoulli with $P(y = 1) = \pi$. Under the naïve Bayes assumption, show that the decision boundary $\{(x_1, x_2, \ldots, x_d) : P(y = 0 \mid x_1, x_2, \ldots, x_d) = P(y = 1 \mid x_1, x_2, \ldots, x_d)\}$ is linear in $x_1, x_2, \ldots, x_d$.

$$\log P(y=1) + \sum_{i=1}^{d} \log(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x_i - \mu_{i,1})^2}{2}}) > \log P(y=0) + \sum_{i=1}^{d} \log(\frac{1}{\sqrt{2\pi}} e^{-\frac{(x_i - \mu_{i,0})^2}{2}})$$

$$\log(\pi) + \sum_{i=1}^{d} \log(\frac{1}{\sqrt{2\pi}}) + \log(e^{-\frac{(x_i - \mu_{i,1})^2}{2}}) > \log(1-\pi) + \sum_{i=1}^{d} \log(\frac{1}{\sqrt{2\pi}}) + \log(e^{-\frac{(x_i - \mu_{i,0})^2}{2}})$$

$$\log(\pi) + \sum_{i=1}^{d} \log(e^{-\frac{(x_i - \mu_{i,1})^2}{2}}) > \log(1-\pi) + \sum_{i=1}^{d} \log(e^{-\frac{(x_i - \mu_{i,0})^2}{2}})$$

$$\log(\pi) - \sum_{i=1}^{d} \frac{(x_i - \mu_{i,1})^2}{2} > \log(1-\pi) - \sum_{i=1}^{d} \frac{(x_i - \mu_{i,0})^2}{2}$$

$$\log(\frac{\pi}{1-\pi}) - \frac{1}{2}\sum_{i=1}^{d}[(x_i - \mu_{i,1})^2 - (x_i - \mu_{i,0})^2] > 0$$

$$\log(\frac{\pi}{1-\pi}) - \frac{1}{2}[\sum_{i=1}^{d} 2x_i(\mu_{i,0} - \mu_{i,1}) + \sum_{i=1}^{d}(\mu_{i,1}^2 - \mu_{i,0}^2)] > 0$$

$$\log(\frac{\pi}{1-\pi}) - \sum_{i=1}^{d} x_i(\mu_{i,0} - \mu_{i,1}) - \frac{1}{2}\sum_{i=1}^{d}(\mu_{i,1}^2 - \mu_{i,0}^2) > 0$$

$$w_0 = \log(\frac{\pi}{1-\pi}) - \sum_{i=1}^{d}(\mu_{i,1}^2 - \mu_{i,0}^2)$$

$$w_i = -(\mu_{i,0} - \mu_{i,1}) = \mu_{i,1} - \mu_{i,0}$$

We have shown that our decision boundary takes the form of $w_0 + w^T x = 0$, which is the equation of a hyperplane. This means we make a decision simply based on which side of the hyperplane the point lies, which is a linear decision. We only multiply the features (x) by w_i, $w_0 + w^T x$ determines which side of the hyperplane x lies on.

3. **MLE estimators can be biased** Given N independent observations drawn from a univariate Gaussian distribution $x^{(1)}, ..., x^{(N)} \sim \text{N}(\mu, \sigma^2)$, recall that the MLE of the mean and variance parameters are

$$\widehat{\mu} = \frac{1}{n}\sum_{i=1}^{n} x^{(i)} \text{ and } \widehat{\sigma}^2 = \frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \widehat{\mu})^2.$$

(a) [**6 points**] Prove that $\widehat{\mu}$ is an *unbiased* estimator of μ (by showing $\mathbb{E}[\widehat{\mu}] = \mu$) and that $\widehat{\sigma}^2$ is a *biased* estimator of σ^2 (by showing $\mathbb{E}[\widehat{\sigma}^2] \neq \sigma^2$).

Part 1:

$$\mathbb{E}[\widehat{\mu}]$$

$$= \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n} x^{(1)}]$$

$$= \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}[x^{(1)}]$$

$$= \frac{1}{n}\sum_{i=1}^{n} \mu$$

$$= \frac{n\mu}{n} = \mu$$

Part 2:

$$\mathbb{E}[\widehat{\sigma^2}]$$

$$= \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \widehat{\mu})^2]$$

At this point we cannot move the expected value inside as we did before, this is because $\widehat{\mu}$ is not a constant but instead a random variable which depends on x^i. We're attempting to subtract the sample mean from each x^i which is biasing the results.

(b) [**4 points**] Based on your response to the previous question, propose an unbiased estimator of σ^2.

We can instead use the true mean, which unlike $\widehat{\mu}$, the true mean is not a random variable. The proof looks much like above expect we can now see that the expected value $\mathbb{E}[(x^i - \mu)^2]$ is simply σ^2. Using $\tilde{\sigma}$ as our true σ.

$$\mathbb{E}[\tilde{\sigma^2}]$$

$$= \mathbb{E}[\frac{1}{n}\sum_{i=1}^{n}(x^{(i)} - \tilde{\mu})^2] = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}[(x^{(i)} - \tilde{\mu})^2]$$

$$= \frac{1}{n}\sum_{i=1}^{n}\sigma^2 = \frac{n\sigma^2}{n} = \sigma^2$$

4 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.

4.1 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 $X_{\text{train}} = \{x_1, x_2, ..., x_T\}$ 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 $x_1, x_2, x_3, ..., x_D$ to predict the value at x_{D+1}; temperatures $x_2, x_3, ..., x_{D+1}$ to predict the temperature at x_{D+2}; and so on so that in general we use $x_i, x_{i+1}, ..., x_{i+(D-1)}$ 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. $[x_{D+1}, x_{D+2}, ..., x_T]$.

Similarly, we can define our test portion of our dataset as $X_{\text{test}} = \{x_{T+1}, x_{T+2}, ..., x_{T+C}\}$ 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.

4.2 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 Canvas.

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`

   As a reminder, you are not permitted to use libraries other than `numpy` in your implementations.

   ```
   Weights: [0.04703033, 0.03691596, -0.01838865, -0.01131895, -0.0269746, -0.01710722, -0.03726656,
   -0.03752731, -0.06354903, -0.14689917, 1.31700047]
   Time: 0.00860238 Seconds
   MSE on X_test: 0.248276
   ```

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`.

   ```
   D = 50
   Weights: [ 1.56513152e-02 -4.10218693e-02 -2.13704521e-02 1.28165307e-02 1.22195478e-02 6.65382259e-03
   2.83376393e-03 1.50387463e-02 1.19685030e-02 1.09388051e-02 4.45944819e-04]
   Time: 0.1812548 Seconds
   MSE on X_test: 0.224512

   D = 100
   Weights: [ 1.39676989e-02 2.46391589e-03 -1.52371016e-02 1.46048026e-03 -2.07164421e-02 6.43107808e-03
   3.32987310e-03 9.83848058e-03 6.46355507e-03 6.80381936e-03 -2.42420759e-03]
   Time: 0.83357191 Seconds
   MSE on X_text: 0.2226265

   D = 500
   Weights: [ 8.47041325e-03 -2.93350619e-03 -2.12084992e-04 8.59965870e-03 -1.12654517e-02 1.92714010e-03
   5.97900957e-03 -5.78491785e-03 3.18520148e-03 -2.24934085e-03 -1.77437603e-04]
   Time: 19.9485123 Seconds
   MSE on X_test: 0.22023974
   ```

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.

I estimate that it would take around 404 minutes to train. This was estimated using a second order polynomial, with the equation $-0.0161 + 4.93E - 4x + 7.89E - 5x^2$. The time complexity of the training should be on the order of $\mathcal{O}(n^3)$, but I assume some optimizations are being made behind the scenes in numpy to utilize more than a single thread of execution, bringing the actual run-time complexity closer to $\mathcal{O}(n^2)$.

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.

   Again, please 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.

Train MSE: 14.32912868
Test MSE: 14.53604101
Total training time: 48.77699256 seconds
First 10 items in final weight vector: [1.00000497e+00 3.27979418e-04 3.25831333e-04 3.18415304e-04 3.06413297e-04 2.90506703e-04 2.71446570e-04 2.50401817e-04 2.28115344e-04 2.05639595e-04]

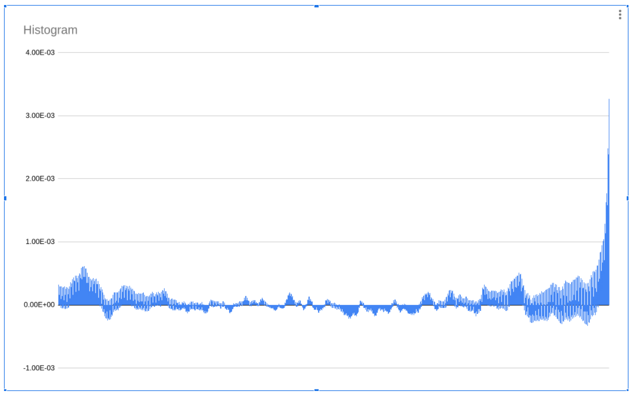

Figure 1: Weights excluding bias

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?

Looking at the plot of weights above this text box, and only observing the magnitude of each weight, we can see that the first quarter of the weights are of medium importance. The second and third quarter of weights hold little to nor importance, and finally the last quarter of weights holds very high importance, especially getting into the last 24 hours of measurements. We also observe that the weights shift from mostly positive to equally distributed around 0 coming into the last few weights. Finally, the weights at the very end of our array increase in magnitude drastically just before the final weight.

5 Collaboration Questions

1. (a) Did you receive any help whatsoever from anyone in solving this assignment? **Solution** No.

   (b) If you answered 'yes', give full details (e.g. "Jane Doe explained to me what is asked in Question 3.4")

   Solution

2. (a) Did you give any help whatsoever to anyone in solving this assignment? **Solution** No.

   (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")

   Solution

3. (a) Did you find or come across code that implements any part of this assignment? **Solution** No.

   (b) If you answered 'yes', give full details (book & page, URL & location within the page, etc.).

   Solution