Introduction to Machine Learning

ECE 580
Fall 2025

HW #4, Due 03/11/25 11:59pm

Submission Instructions

Submit your work to the corresponding assignment in Gradescope. Although Gradescope accepts multiple file formats, they strongly recommend submitting your assignment as a single PDF file.

It is your responsibility to ensure the uploaded file is: 1) the correct file, 2) complete (includes all pages), 3) legible, and 4) submitted on-time as determined by the Gradescope server system clock.

It is your responsibility to submit a multi-page PDF file and tag the pages that correspond to each question. Pages may be tagged after submission, even if the submission deadline has passed. If you are submitting close to the submission deadline, submit your assignment first then immediately return to tag pages.

When code is requested, submit a PDF print-out of your code. Submitting a URL for a cloud-based repository is insufficient.

Late Submissions

Late submissions will be accepted up to 5 days after the submission deadline, with the following point penalty applied if its late submission is not excused: [1]

1 day (0^+ to 24 hours) late: 2 point deduction ($\frac{1}{5}$ letter grade)
2 days (24^+ to 48 hours) late: 5 point deduction ($\frac{1}{2}$ letter grade)
3 days late: 10 point deduction (1 letter grade)
4 days late: 20 point deduction (2 letter grades)
5 days late: 30 point deduction (3 letter grades)
6 or more days late: score = 0 (not accepted for credit)

The late policy is designed to be minimally punitive for submissions up to 3 days late, yet encourages staying current with the coursework for our course by not allowing one assignment's late submission to overlap with the next assignment's submission.

A homework score will not drop below 0 as a result of applying the late penalty point deduction.

[1] One day = one 24-hour period or fraction thereof.

Gradient Descent and Convexity

1. (Logistic Regression) Consider the logistic regression setup: we are given a training dataset $\mathscr{D} = \{(x_1,y_1)...(x_n,y_n)\}$, where $x_i \in \mathbb{R}^d$ and $y_i \in \{0,1\}$. The probability of y conditioned on x, parameterized by w, is given by

$$Pr(y=1|x;w) = \frac{e^{x^\top w}}{1+e^{x^\top w}},$$

where $w \in \mathbb{R}^2$ is the parameter vector. The negative log-likelihood is thus

$$\ell(w) = \sum_{i=1}^{n} -y_i(w^\top x_i) + \log\left(1+\exp\left(w^\top x_i\right)\right).$$

Typo correction: in the previous version, was missing negative sign.

(5) (a) Write down the expression for $[\nabla\ell(w)]_i$, i.e. the i^{th} entry of the gradient.

(15) (b) Show that $\nabla^2\ell(w)$ has the form

$$\nabla^2\ell(w) = \sum_{i=1}^{n} c(x,w)(1-c(x,w))x_i x_i^\top,$$

for some scalar-valued function $c(x,w)$. State the expression for $c(x,w)$.

(15) (c) (Gradient Lipschitz) Recall that $\ell(w)$ satisfies the M-gradient-Lipschitz condition if, for all w,

$$\left\|\nabla^2\ell(w)\right\|_2 \leq M$$

where $\|A\|_2$ denotes the operator norm for matrix $A \in \mathbb{R}^{d\times d}$.

Using your answer in (b), prove that $\ell(w)$ is M gradient Lipschitz with $M = \frac{1}{4}\lambda_{max}$, where λ_{max} is the maximum eigenvalue of $\sum_{i=1}^n x_i x_i^\top$.
Hint: $\max_{p\in(0,1)} p(1-p) = 0.25$

(15) (d) (Gradient Lipschitz) In lecture, we saw two forms of the M gradient Lipschitz condition:

(A) For all u,v, $\|\nabla L(u) - \nabla L(v)\| \leq M\|u-v\|_2$.

(B) For all u, $\left\|\nabla^2 L(u)\right\|_2 \leq M$.

Prove that (B) implies (A). You may find the following facts useful:

1. $\frac{d}{dt}\nabla L(u+tx) = \nabla^2 L(u+tx)x$.
2. (Triangle Inequality) For any $y(\cdot) : \mathbb{R} \to \mathbb{R}^d$, $\left\|\int_0^1 y(t)dt\right\|_2 \leq \int_0^1 \|y(t)\|_2 dt$

(15) (e) (Convexity) Using your answer in (b), prove that $\ell(w)$ is convex via the second-order condition. I.e. show that $\nabla^2\ell(w)$ is positive semi-definite.

(10) (f) (Strong Convexity) Recall that $\ell(w)$ is m-strongly-convex if

$$\nabla^2\ell(w) - mI_{d\times d} \succeq 0.$$

Consider the L_2-regularized negative log likelihood for logistic regression, defined as

$$\ell_{reg,c}(w) = \sum_{i=1}^{n} -y_i(w^\top x_i) + \log\left(1+\exp\left(w^\top x_i\right)\right) + c\|w\|_2^2.$$

Show that $\ell_{reg,c}(w)$ is c-strongly-convex.

2. (Simple logistic regression in PyTorch.)
For this question, you will implement Gradient Descent in Pytorch to optimize the logistic regression negative log likelihood loss. Refer to `homework_4_starter_code.ipynb`.

We already defined for you the functions

1. `logistic_probability`$(w,x) = \frac{\exp(w^\top x)}{1+\exp(w^\top x)} = Pr(y=1|x;w)$
2. `logistic_neg_log_likelihood`$(w,X) = -\frac{1}{n}\sum_{i=1}^{n} y_i \log(p_i) + (1-y_i)\log(1-p_i)$,
where $p_i = \frac{\exp(w^\top x_i)}{1+\exp(w^\top x_i)} =$ `logistic_probability`(w,x_i).

Note that both functions operate on batch inputs.

(10) (a) For each choice of learning rate $\delta \in \{1, 0.1, 0.01\}$, perform 100 steps of gradient descent for `logistic_neg_log_likelihood`. Write your code in the 2(a) cell of the provided ipynb notebook.

(5) (b) On a single figure, for each learning rate δ, plot `logistic_neg_log_likelihood`(w_k^δ) against k, where $k \in \{1...100\}$ is the iteration number. What is the best δ out of the three choices?

(5) (c) Using the final-iteration w found for the best δ from part (b) above, visualize the decision boundary using the `visualize_boundary` function provided.

(5) (d) Submit a PDF print-out of your code

3. (Classification on a swiss-roll.)
For this question, you will implement and optimize a simple 2-layer neural network, for a more challenging classification problem.
We will be training a classifier for the data in `homework_4_data_swissroll.pth`. The data loading code has been provided.

(10) (a) As you did for question 2, implement gradient descent for the logistic regression on the data in `homework_4_data_swissroll.pth`. Use $\delta = 0.2$ and train for 200 iterations. Let w_{final} denote the parameter w found in the last iteration of gradient descent.

Visualize the decision boundary for w_{final} using `visualize_boundary`.

(10) (b) for each x_i, let the prediction be $\hat{y}_i$ be 1 if `logistic_probability`$(w,x_i) \geq 0.5$, and let $\hat{y}_i = 0$ otherwise. Let the classification accuracy be given by $\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}(\hat{y}_i = y_i)$.
On three separate figures, plot

1. logistic negative log likelihood against gradient descent iteration.
2. accuracy against iterations.
3. log(gradient norm) against iterations

Based on your plots, answer the questions:

1. Has the loss converged?
2. Is the classification accuracy converging to 100%?
3. Is the gradient norm converging to 0?

(20) (c) We will implement `MLP_probabillity` as a replacement for `logistic_probability`. MLP stands for "Multi-Layer-Preceptron". For this problem, the MLP will consist of one linear layer

(w_1), followed by a (coordinate-wise) ReLU activation, followed by another linear layer (w_2, b_2). Concretely, the prediction probability is defined by

$$\texttt{MLP\_probability}(w_1, w_2, b_2, x) = \frac{\exp(w_2^\top \texttt{relu}(w_1^\top x) + b_2)}{1 + \exp(w_2^\top \texttt{relu}(w_1^\top x) + b_2)} = Pr(y = 1 | x; w_1, w_2, b_2),$$

where $w_1 \in \mathbb{R}^{2\times 10}, w_2 \in \mathbb{R}^{10\times 1}, b_2 \in \mathbb{R}$.

Your implementation should handle a n-by-2 feature matrix X as input, and output a n-dimensional vector of probabilities. See the provided `logistic_probability` for an example.

(20) (d) Implement gradient descent on w_1, w_2, b_2, with respect to the negative log-likelihood (under `MLP_probability`). For full credit, ensure that your final classification accuracy is 100%.

(10) (e) On three separate figures, plot

1. MLP negative log likelihood, from question 3(d), against gradient descent iteration.
2. accuracy against iterations.
3. log(gradient norm) against iterations

Based on your plots, answer the questions:

1. Has the loss converged?
2. Is the classification accuracy converging to 100%?
3. Is the gradient norm converging to 0?

How does your answer to (e) differ from your answer to (b)?

(5) (f) Submit a PDF print-out of your code

Total weight = 175