CS 274A Homework 4

Probabilistic Learning: Theory and Algorithms, CS 274A, Winter 2025

Due: 6pm Thursday March 6th, submit via Gradescope

Instructions and Guidelines for Homeworks

- Please answer all of the questions and submit your solutions to Gradescope (either hand-written or typed are fine as long as the writing is legible).
- All problems are worth equal points (10 points) unless otherwise stated. All homeworks will get equal weight in computation of the final grade for the class (with lowest-scoring homework being dropped).
- The homeworks are intended to help you better understand the concepts we discuss in class. It is important that you solve the problems yourself to help you learn and reinforce the material from class. If you don't do the homeworks you may have difficulty in the exams later in the quarter.
- In problems that ask you to derive or prove a result you should submit a complete mathematical proof (i.e., each line must follow logically from the preceding one, without "hand-waving"). Be as clear as possible in explaining your notation and in stating your reasoning as you go from line to line.
- If you can't solve a problem, you can discuss the high-level concepts *verbally* with another student (e.g., what concepts from the lectures or notes or text are relevant to a problem). However, you should not discuss any of the details of a solution with another student. In particular, do not look at (or show to any other student) *any written material* directly related to the homeworks, including other students' solutions or drafts of solutions, solutions from previous versions of this class, etc. The work you submit should be your own original work.
- If you need to you can look up standard results/definition/identities from textbooks, class notes, textbooks, other reference material (e.g., from the Web). If you base any part of your solution on material that we did not discuss in class, or is not in the class notes, or is not a standard known result, then you should provide a reference in terms of where the result is from, e.g., "based on material in Section 2.2 in ....." or a URL (e.g., Wikipedia).
- Please read each problem carefully. If you believe there is a typo, or some information is missing, or the problem is unclear, please post a question on the Ed discussion board.

Required Reading for Homework 4:

1. For Problems 1,2,3: Class Notes on Regression (Noteset 5 on class Website), Sections 1 to 5 and 7 and 8 (you can skip Section 6 if you wish).
2. For Problems 4, 5, 6: Class Notes on Classification (NoteSet 6 on the class Website)

Also be aware that may need to look up certain terms (e.g., in a textbook or Wikipedia), such as *convexity* or *positive semi-definite* if you don't recall what these terms are or have not seen them before.

Optional Reading for Homework 4: The following reading may (in the Mathematics for Machine Learning (MML) text, `https://mml-book.github.io/book/mml-book.pdf`, where page numbers refer to the current (Feb 2025) online version) also provide additional helpful background for Problems 1 to 4, if you have time:

1. Chapter 7 on Continuous Optimization
   (a) Section 7.1, pages 225-233
   (b) Section 7.3 up to start of 7.3.1, pages 236-239
2. Chapter 8, sections 8.1 and 8.2
3. Chapter 9 on Linear Regression, Sections 9.1 and 9.2

Feel free to optionally read the other sections of the Chapters above. The notation will differ in places to the notation we are using in class, but will be broadly similar.

Problem 1: Normal Equations for Least Squares (MSE) Regression

Assume we have training data in the form $D = \{(\underline{x}_i, y_i)\}, i = 1, \ldots, N$, where each $\underline{x}_i$ is a d-dimensional real-valued vector (with one component set to the constant 1 to allow for an intercept term) and where each y_i is a real-valued scalar. Assume we wish to fit a linear model of the form $\underline{\theta}^T \underline{x}$ where $\underline{\theta}$ is a d-dimensional parameter vector, where by "fit" we mean here that we want to find $\hat{\underline{\theta}}$ that minimizes $MSE(\underline{\theta}) = \frac{1}{N}\sum_{i=1}^{N}\big(y_i - \underline{\theta}^T \underline{x}_i\big)^2$.

1. Prove that the solution to this problem can be written as the solution of a system of d linear equations (often referred to as the "normal equations") that can be written in the form $\mathbf{A}\underline{\theta} = \underline{b}$ where $\underline{\theta}$ has dimension $d \times 1$, $\mathbf{A}$ is a $d \times d$ matrix, and $\underline{b}$ is a $d \times 1$ vector. Starting from the definition of $MSE(\underline{\theta})$ above, carefully write out all steps in your proof, and clearly show how $\mathbf{A}$ and $\underline{b}$ are defined. If you need to assume as part of your solution that a particular matrix is full rank then assume so and state that you have assumed this.

2. Define the time complexity of minimizing $MSE(\underline{\theta})$ using the normal equations, given a dataset $D = \{(\underline{x}_i, y_i)\}, i = 1, \ldots, N$. Time complexity is defined as being "on the order of" (i.e., "big O") of some function of d and N, e.g., computing a sum of N terms has time complexity $O(N)$, multiplying a $1 \times N$ vector by a $N \times d$ matrix has time complexity $O(Nd)$, etc.

Problem 2: Estimating a Linear Model using Maximum Likelihood

Assume we have IID training data in the form $D = \{(x_i, y_i)\}, i = 1, \ldots, N$, where x_i and y_i are both one-dimensional and real-valued. Say we assume that y given x is a conditional Gaussian density with mean $E[y|x] = ax + b$ and with variance σ^2 (see example 8.4 in the MML text). Assume that a, b, and σ^2 are unknown.

Show from first principles that the maximum likelihood estimates for each of a, b, and σ^2 can be written as:

$$\hat{a} = \frac{\overline{xy} - \bar{x}\bar{y}}{\overline{x^2} - (\overline{x})^2}$$

$$\hat{b} = \overline{y} - \hat{a}\overline{x}$$

$$\hat{\sigma}^2 = \frac{1}{N}\sum_i \big(y_i - [\hat{a}x_i + \hat{b}]\big)^2$$

where terms such as $\bar{x}, \bar{y}$ represent empirical averages over the N datapoints, and terms like $\hat{a}$ represent maximum likelihood estimates.

Note that what is referred to likelihood above (and in the remaining problems in this homework) is actually the *conditional likelihood*, $P(D_y|D_x, \underline{\theta})$: see Sections 7 and 8 of NoteSet 5 on Regression.

Problem 3: L1 or Lasso Regression

Consider a squared error loss function $MSE(\underline{\theta}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - f(\underline{x}_i;\underline{\theta})^2$ with training data $D = \{(\underline{x}_i, y_i)\}, i = 1, \ldots, N$ and where f is some prediction model with unknown parameters $\underline{\theta} = (\underline{\theta}_1, \ldots, \underline{\theta}_p)$. A popular regularization method takes the form $r(\underline{\theta}) = \sum_{j=1}^{p} |\underline{\theta}_j|$, resulting in an optimization problem where we minimize $MSE(\underline{\theta}) + \lambda r(\underline{\theta})$ as a function of $\underline{\theta}$, rather than just minimizing $MSE(\underline{\theta})$. Here λ is the relative weight of the regularization term (this is known as L1 or Lasso regularization).

Clearly show how we can interpret L1 regularization in terms of a prior on $\underline{\theta}$ (by viewing this optimization problem from a Bayesian MAP perspective). Be sure to state clearly what distributional form this prior is, i.e., what name it has. (Note that you can assume here that the likelihood is Gaussian where the mean of y as a function of $\underline{x}$ is modeled by $f(\underline{x};\underline{\theta})$ and the variance term in the likelihood is known and fixed, as in Section 7 in NoteSet 5).

Problem 4: Convexity for Logistic Classifiers

Consider a classification problem where we have training data $D = \{(\underline{x}_i, y_i)\}, 1 \ldots, i, \ldots, N$ where $\underline{x}_i$ are real-valued d-dimensional vectors and $y_i \in \{0, 1\}$ are binary class labels. We will assume for convenience that the first component of $\underline{x}$ always takes value 1, allowing us to have an intercept (or bias) term in our model. Let $f(\underline{x};\underline{\theta})$ be a logistic regression model, where $\underline{\theta}$ is a d-dimensional parameter vector (set of weights), and our predictive model is

$$f(\underline{x};\underline{\theta}) \;=\; \frac{1}{1 + \exp(-\underline{\theta}^T \underline{x})}.$$

where $f(\underline{x};\underline{\theta})$ is our estimate of $p(y = 1|\underline{x})$.

Let the objective function (that we want to minimize) be the cross-entropy loss (also known as the log-loss), defined as

$$CE(\underline{\theta}) \;=\; -\frac{1}{N}\sum_{i=1}^{N} y_i \log f(x_i;\underline{\theta}) \;+\; (1 - y_i)\log\bigl(1 - f(x_i;\underline{\theta})\bigr).$$

1. Derive the equation for the gradient $\nabla_{\underline{\theta}}$ of $CE(\underline{\theta})$ for this optimization problem
2. Prove that $CE(\underline{\theta})$ has a single global minimum and no local minima by proving that it is a convex function of $\underline{\theta}$. Note that one way to prove this is to use the fact that a sufficient condition for a function like $CE(\underline{\theta})$ to be convex is that the Hessian matrix $\mathbf{H}_{\underline{\theta}}$ is positive semi-definite, where the Hessian matrix is the matrix of second partial derivatives of $CE(\underline{\theta})$ with respect to its parameters. There are also other ways to also prove convexity.
3. Iterative optimization methods for learning the parameters of a logistic model include first-order methods (which only use the gradient) and second-order methods (which use the Hessian). Define the computational (time) complexity as a function of d and N for computing (a) a gradient vector $\nabla_{\underline{\theta}}$, and (b) for computing a Hessian matrix $\mathbf{H}_{\underline{\theta}}$ and explain how you arrived at your solution.

Problem 5: Bayes Error Rate with Uniform and Exponential Densities

The Bayes error rate is defined as the optimal (minimum achievable) error rate for a classification problem. It can be expressed as a sum of integrals, where the sum is over different decision regions, such that within each decision region one of the classes is more likely than any other. See Section 6 in NoteSet 6 for full details.

Consider the classification problem with a 1-dimensional feature x, where $p(x|y = 1)$ is a uniform density $U(a, b)$, and where $p(x|y = 0)$ is an exponential density $\lambda \exp(-\lambda x)$ with $\lambda = 1$ (for $x \geq 0$, with $p(x|y = 0) = 0$ for $x < 0$. Given this setup answer the following problems.

1. Assume $a = 0, b > 0$. Derive a general equation (or set of equations) that defines the decision boundary (or boundaries) as a function of the parameters.
2. Let $a = 1, b = 3$, $\lambda = 1$. Assume $p(y = 1) = 0.5$. Generate a plot for each of $p(x|y = 1)p(y = 1)$ and $p(x|y = 0)p(y = 0)$ (put them on the same plot) and show clearly the location of the decision region or regions. You can let the x-axis range from 0 to 8.
3. Compute the Bayes error rate for part 2 of this problem to an accuracy of within 3 decimal places. Show clearly how you computed the Bayes error rate.
4. Now find the decision boundaries and compute the Bayes error rate, to an accuracy of 3 decimal places, for the case where $p(y = 1) = 0.4$ and $p(y = 2) = 0.6$.

Problem 6: Bayesian Estimation for Logistic Classifiers (with Python Notebook)

This problem will involve the use of a Python notebook which is running in Google Colab.

Before you start working on the solution to the problems below, please make sure to do the following first:

- Read Section 10.5 (up to end of Section 10.2.5.1) in Kevin Murphy's book: Probabilistic Machine Learning: An Introduction, March 2022, available online at `https://probml.github.io/pml-book/book1.html`. This will give you the relevant math background for the problem.
- In your browser open up the notebook for this problem: `https://colab.research.google.com/drive/1sLoNn3Il6_DrYcISE1KH-8IMfcGPzIHh#scrollTo=BlpNXwgrpBEb`. I recommend you make a local copy (e.g., in the menu at the top click "File: Save to Drive"). This notebook is running in Google Colab. If you are not familiar with Google Colab you may want to first read an introductory tutorial on Jupyter notebooks and Colab, or have someone show you how it works.
- Confirm that you can run all of the cells in the notebook in order and that various graphs are generated. Don't change any of the code (at least while doing the homework: feel free to experiment with the

code if you wish, e.g., change the number of data points in the simulations, etc, after you have finished the homework).

The notebook is going through the following steps:

- We generate data in a 2-dimensional space x_1, x_2 for a two-class problem, where the class label $y \in \{0, 1\}$. The data in each class is Gaussian, i.e., $p(x|y = 1)$ and $p(x|y = 0)$ are each Gaussians, with the same covariance structure and different means.
- We then investigate fitting a logistic classifier to this data of the form $p(y = 1|x) = \frac{1}{1+exp(-w_1x_1-w_2x_2)}$. Note that we don't fit an intercept term here (which means it is being set to 0 in effect): this restriction allows us visualize likelihoods, etc, in the two-dimensional (w_1, w_2) space.
- We define relatively weak Gaussian priors over (w_1, w_2), with mean zero and standard deviation equal to α in the priors. (Its defined as a joint prior, but the covariance is diagonal, so in effect its the product of two independent priors).
- The notebook then plots the surface (as contours) of the log-likelihood and the unnormalized posterior, $\log p(D|\theta)p(\theta)$,in the two-dimensional space defined by w_1, w_2.
- It then estimates (by finding the maximum of $\log p(D|\theta)p(\theta)$) the MAP values of w_1, w_2.
- Since the posterior cannot be computed in closed form for a logistic model, the notebook computes a Laplace approximation to the true posterior, in the form of a Gaussian density: see Section 4.6.8.2 in the Murphy book for details on the Laplace approximation, in particular Equation 4.214).
- Finally, we can approximate the posterior predictive for the weights, by sampling w_1, w_2 values from the approximate Laplace posterior, and then averaging over these samples to compute $p(y = 1|x)$ for any particular input x. (See Section 10.5.2 and 10.5.2.1 for details).

You should go through the notebook at this point and make sure you can see in the code where all of the steps above are happening. Note that even if you don't know Python, the syntax is fairly simple and with the comments you should be able to figure out how the steps above map to the different cells of code and the graphs generated in the notebook.

Answer the following questions and submit your answers (no need to submit any code or graphs). Note that it may be helpful to print out values of certain variables in the code, e.g., if you want to print a number w just add a line with `print(w)` in the code. Feel free to use equations as well as words in your answers.

1. The datapoints (if you use the provided random seed) are linearly separable (i.e., we can draw a linear decision boundary to perfectly separate the data from the two classes). Explain in 1 or 2 sentences what this implies for generating a maximum likelihood estimate for the logistic model. You can use the plot of the log-likelihood surface in the notebook to support your answer.
2. Why will adding a prior help with any issues that may come up with maximum likelihood in part 1 of this question?

3. What are the values of the MAP estimates for w_1, w_2?

4. Comment on the shape of the Laplace approximation relative to the shape of the unnormalized posterior (both are generated as plots by the notebook).

5. Comment on the shape of the posterior predictive for $p(y = 1|x)$ versus the shape $p(y = 1|x, w_{MAP}$ (both are shown as contour plots in the notebook)[1]. Explain as best you can (in 2 or 3 sentences or equations) what the specific differences are and why these differences exist.

6. What is the true optimal decision boundary for these two Gaussian classes i.e., the optimal values for w_1, w_2 if we had an infinite amount of data? (this requires solving for the decision boundary as in problems 1 and 2).

7. If the amount of training data were to approach infinity, could the logistic model perfectly match the optimal decision boundary from part 6? Explain your answer in 1 or 2 sentences.

[1] Technically $p(y = 1|x)$ should be written in full as $p(y = 1|x, D_x, D_y,$ prior params)