Homework #2

Due: February 27, 2026 at 11:59 PM

Classification and Bias-Variance Trade-offs

Introduction

This homework is about classification, bias-variance trade-offs, and uncertainty quantification.

The datasets that we will be working with relate to astronomical observations and loan applicants The first dataset, found at `data/planet-obs.csv`, contains information on whether a planet was observed (as a binary variable) at given points in time. This will be used in Problem 1. The second dataset, available at `data/hr.csv`, details different loan applicants and their measured debt to income ratio and credit score. You will work with this data in Problem 3.

As a general note, for classification problems we imagine that we have the input matrix $\mathbf{X} \in \mathbb{R}^{N \times D}$ (or perhaps they have been mapped to some basis $\mathbf{\Phi}$, without loss of generality) with outputs now "one-hot encoded." This means that if there are K output classes, rather than representing the output label y as an integer $1, 2, \ldots, K$, we represent $\mathbf{y}$ as a "one-hot" vector of length K. A "one-hot" vector is defined as having every component equal to 0 except for a single component which has value equal to 1. For example, if there are $K = 7$ classes and a particular data point belongs to class 3, then the target vector for this data point would be $\mathbf{y} = [0, 0, 1, 0, 0, 0, 0]$. We will define C_1 to be the one-hot vector for the 1st class, C_2 for the 2nd class, etc. Thus, in the previous example $\mathbf{y} = C_3$. If there are K total classes, then the set of possible labels is $\{C_1 \ldots C_K\} = \{C_k\}_{k=1}^K$. Throughout the assignment we will assume that each label $\mathbf{y} \in \{C_k\}_{k=1}^K$ unless otherwise specified. The most common exception is the case of binary classification ($K = 2$), in which case labels are the typical integers $y \in \{0, 1\}$.

Resources and Submission Instructions

In problems 1 and 3, you may use `numpy` or `scipy`, but not `scipy.optimize` or `sklearn`. Example code is given in the provided notebook. In past years, students have reported numerical stability issues for problem 3 and sometimes problem 1; we encourage students to not worry too much about this, as we are mindful of this in grading, but we recommend trying to run your code on **Google Colab** if you encounter issues (since this has resolved the issue for many students in the past).

Please type your solutions after the corresponding problems using this LaTeX template, and start each problem on a new page. Please submit the writeup PDF to the Gradescope assignment 'HW2'. Remember to assign pages for each question. **You must include any plots in your writeup PDF.** Please submit your LaTeXfile and code files to the Gradescope assignment 'HW2 - Supplemental.' The supplemental files will only be checked in special cases, e.g. honor code issues, etc. Your files should be named in the same way as we provide them in the repository, e.g. `hw2.pdf`, etc.

Problem 1 (Exploring Bias-Variance and Uncertainty, 30pts)

In this problem, we will explore the bias and variance of a few different model classes when it comes to logistic regression, use cross validation for model selection, and investigate two sources of predictive uncertainty in a synthetic (made-up) scenario.

We are using a powerful telescope in the northern hemisphere to gather measurements of some planet of interest. At certain times however, our telescope is unable to detect the planet due to its positioning around its star. The data in `data/planet-obs.csv` records the observation time in the "Time" column and whether the planet was detected in the "Observed" column (with the value 1 representing that it was observed). These observations were taken over a dark, clear week, which is representative of the region. Since telescope time is expensive, we would like to build a model to help us schedule and find times when we are likely to detect the planet.

1. Before we begin fitting any models, let us first derive the mathematical explanation for the bias-variance tradeoff. Let $\mathcal{D}$ denote the training dataset, while $\mathbf{x}$ is a fixed input (not in $\mathcal{D}$). Suppose the labels are generated via

   $$y = f(\mathbf{x}) + \varepsilon$$

   where $f(\mathbf{x})$ is the true regression function and ε is a noise term with $\mathbb{E}[\varepsilon|\mathbf{x}] = 0$ and $\text{Var}(\varepsilon|\mathbf{x}) = \sigma^2$. We'll denote the predictor learned from $\mathcal{D}$ as $\hat{f}_{\mathcal{D}}(\mathbf{x})$. Then, we have the prediction error as

   $$\text{MSE} = \mathbb{E}_{\mathcal{D},y|\mathbf{x}}[(y - \hat{f}_{\mathcal{D}}(\mathbf{x}))^2]$$

   We will see that this error decomposes into a noise term, a bias term, and a variance term.

   (a) Show that the expression for MSE above decomposes into

   $$\mathbb{E}_{y|\mathbf{x}}[(y - f(\mathbf{x}))^2] + \mathbb{E}_{\mathcal{D}}[(f(\mathbf{x}) - \hat{f}_{\mathcal{D}}(\mathbf{x}))^2] + \{\text{third term}\}$$

   Hint: Add and subtract $f(\mathbf{x})$.

   (b) Show that the third term from our final expression in part (b) simplifies to 0. Hint: Use Adam's Law/Law of Total Expectation.

   (c) Show that the first term from our final expression in part (b) represents the noise in our data, i.e. $\mathbb{E}_{y|\mathbf{x}}[(y - f(\mathbf{x}))^2] = \sigma^2$.

   (d) Define the average predictor $\bar{f}(\mathbf{x}) = \mathbb{E}_{\mathcal{D}}[\hat{f}_{\mathcal{D}}(\mathbf{x})]$. Now show that the the second term from our final expression in part (b) represents the squared bias plus variance, i.e. $\mathbb{E}_{\mathcal{D}}[(f(\mathbf{x}) - \hat{f}_{\mathcal{D}}(\mathbf{x}))^2] = (f(\mathbf{x}) - \bar{f}(\mathbf{x}))^2 + \mathbb{E}_{\mathcal{D}}[(\bar{f}(\mathbf{x}) - \hat{f}_{\mathcal{D}}(\mathbf{x}))^2]$.

2. Now let's explore the use of different bases to model our data. First, randomly split the data into 10 mini-datasets of size $N = 30$. We will use three different basis functions to model the data. We implemented basis 1 as $[1, t]$, where t is the data to help you get started. Then, choose two basis functions, which you'll use to transform the input data in the regression. (Hint: Examples of such basis functions were given in Lecture 2.) **Please write your chosen basis functions in the written section of your submission for this part of the question.**

   For each of these bases, fit a logistic regression model using $\text{sigmoid}(\mathbf{w}^\top \boldsymbol{\phi}(t))$ to each dataset by using gradient descent to minimize the negative log-likelihood. This means you will be running gradient descent 10 times for each basis, once for each dataset.

   Use the given starting values of $\mathbf{w}$ and a learning rate of $\eta = 0.001$, take 1,000 update steps for each gradient descent run, and make sure to average the gradient over the data points at each step. These parameters, while not perfect, will ensure your code runs reasonably quickly.

3. After consulting with a domain expert, we find that the probability of observing the planet is periodic as the planet revolves around its star—we are more likely to observe the planet when it is in front of its star than when it is behind it. In fact, the expert determines that observation follows the generating process $y \sim \text{Bern}(f(t))$, where $f(t) = 0.4 \times \cos(1.1t + 1) + 0.5$ for $t \in [0, 6]$ and $y \in \{0, 1\}$. Note that we, the modelers, do not usually see the true data distribution. Knowledge of the true $f(t)$ is only exposed in this problem to allow for verification of the true bias.

   Use the given code to plot the true process versus your learned models. Include your plots in your solution PDF.

   In no more than 5 sentences, explain how bias and variance are reflected in the 3 types of curves on the graphs. How do the fits of the individual and mean prediction functions change? Keeping in mind that none of the model classes match the true generating process exactly, discuss the extent to which each of the bases approximates the true process.

4. If we were to increase the size of each dataset drawn from $N = 30$ to a larger number, how would the bias and variance change for each basis? Why might this be the case? You may experiment with generating your own data that follows the true process and plotting the results, but this is **not** necessary. **Your response should not be longer than 5 sentences**.

5. Consider the test point $t = 0.1$. Using your models trained on basis $\boldsymbol{\phi}_3$, report the predicted probability of observation of the *first* model (the model trained on the first 30 data points). How can we interpret this probability as a measure of uncertainty? Then, compute the variance of the classification probability over your 10 models at the same point $t = 0.1$. How does this measurement capture another source of uncertainty, and how does this differ from the uncertainty represented by the classification probability? Repeat this process (reporting the first model's classification probability and the variance over the 10 models) for the point $t = 3.2$.

   Compare the uncertainties and their sources at times $t = 0.1$ and $t = 3.2$.

6. We now need to make some decisions about when to request time on the telescope. The justifications of your decisions will be sent to your funding agency, which will determine whether you will be allocated funds to use the telescope for your project. You seek out a team that has used the alternative telescope for observing this planet, and they provide you their observation file `data/planet-obs-alternate.csv`. Compare the observations from your telescope to theirs. What seems to be happening? What might be an appropriate model for this? Your funding agency asks you to refit your models on these new data. Do you think this is a reasonable ask, and if so, how will it help you make better decisions about when to request viewing time? If not, why do you think the additional modeling will not help? You do *not* need to do any modeling; answer this question in **no more than 10 lines**. We are looking for your reasoning; there may be more than one valid answer.

7. In part 1, we have selected 3 possible bases to use as features when training a model. This gave us 3 possible models. In this part, we will choose the right learning algorithm by using cross validation. Take your dataset split and train 10 models for each basis, using 9 of the chunks for training and 1 of the chunks for validation. Compare the average error across the 10 folds for the 3 bases. Which basis performs the best? Why do you think it performs the best?

Solution: Your solution here.

Problem 2 (Maximum likelihood in classification, 15pts)

Consider now a generative K-class model. We adopt class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$ for all $k \in \{1, \ldots, K\}$ (where π_k is a parameter of the prior). Let $p(\mathbf{x}|\mathbf{y} = C_k)$ denote the class-conditional density of features $\mathbf{x}$ (in this case for class C_k). Consider the data set $\mathcal{D} = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where as above $\mathbf{y}_i \in \{C_k\}_{k=1}^K$ is encoded as a one-hot target vector and the data are independent.

1. Write out the log-likelihood of the data set, $\ln p(\mathcal{D}; \boldsymbol{\pi})$.
2. Since the prior forms a distribution, it has the constraint that $\sum_k \pi_k - 1 = 0$. Using the hint on Lagrange multipliers below, give the expression for the maximum-likelihood estimator for the prior class-membership probabilities, i.e. $\hat{\pi}_k$. Make sure to write out the intermediary equation you need to solve to obtain this estimator. Briefly state why your final answer is intuitive.

For the remaining questions, let the class-conditional probabilities be Gaussian distributions with the same covariance matrix

$$p(\mathbf{x}|\mathbf{y} = C_k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \text{ for } k \in \{1, \ldots, K\}$$

and different means $\boldsymbol{\mu}_k$ for each class.

3. Derive the gradient of the log-likelihood with respect to vector $\boldsymbol{\mu}_k$. Write the expression in matrix form as a function of the variables defined throughout this exercise. Simplify as much as possible for full credit.
4. Derive the maximum-likelihood estimator $\hat{\mu}_k$ for vector $\boldsymbol{\mu}_k$. Briefly state why your final answer is intuitive.
5. Derive the gradient for the log-likelihood with respect to the covariance matrix $\boldsymbol{\Sigma}$ (i.e., looking to find an MLE for the covariance). Since you are differentiating with respect to a *matrix*, the resulting expression should be a matrix!
6. Derive the maximum likelihood estimator $\hat{\Sigma}$ of the covariance matrix.

Hint: Lagrange Multipliers. Lagrange Multipliers are a method for optimizing a function f with respect to an equality constraint, i.e.

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } g(\mathbf{x}) = 0.$$

This can be turned into an unconstrained problem by introducing a Lagrange multiplier λ and constructing the Lagrangian function,

$$L(\mathbf{x}, \lambda) = f(\mathbf{x}) + \lambda g(\mathbf{x}).$$

It can be shown that it is a necessary condition that the optimum is a critical point of this new function. We can find this point by solving two equations:

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = 0 \text{ and } \frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = 0$$

Cookbook formulas. Here are some formulas you might want to consider using to compute difficult gradients. You can use them in the homework without proof. If you are looking to hone your matrix calculus skills, try to find different ways to prove these formulas yourself (will not be part of the evaluation of this homework). In general, you can use any formula from the matrix cookbook, as long as you cite it. We opt for the following common notation: $\mathbf{X}^{-\top} := (\mathbf{X}^\top)^{-1}$

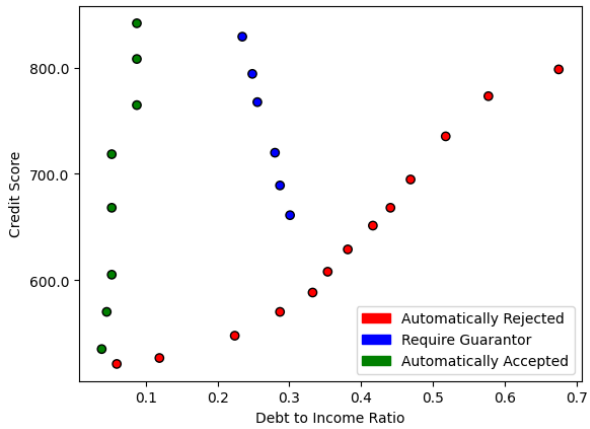
$$\frac{\partial \mathbf{a}^\top \mathbf{X}^{-1} \mathbf{b}}{\partial \mathbf{X}} = -\mathbf{X}^{-\top} \mathbf{a}\mathbf{b}^\top \mathbf{X}^{-\top}$$
$$\frac{\partial \ln |\det(\mathbf{X})|}{\partial \mathbf{X}} = \mathbf{X}^{-\top}$$

These formulas are from the matrix cookbook linked here: The Matrix Cookbook (University of Waterloo).

Solution: Your solution here.

Problem 3 (Classifying Loan Applicants, 30pts)

In this problem, you will code up three different classifiers to classify different types of loan applicants. The file `data/hr.csv` contains data on debt to income ratio measured in tenths of a percent and credit score. The data can be plotted on these two axes:



We've further transformed the raw data on debt to income ratio and credit score so that the default feature vector that you will be working with is defined as such:

$$\boldsymbol{x} = \left[\text{debt\_income\_ratio} \cdot \frac{200}{7} - 7.5, \frac{\text{credit\_score} - 500}{140} + 0.5\right]^\top$$

Please implement the following classifiers:

a) **A generative classifier with Gaussian class-conditional densities with a *shared covariance* matrix** across all classes. Feel free to re-use your Problem 2 results.

b) **Another generative classifier with Gaussian class-conditional densities , but now with a *separate covariance* matrix** learned for each class. (Note: The staff implementation can switch between the two Gaussian generative classifiers with just a few lines of code.)

c) **A multi-class logistic regression classifier** using the softmax activation function. In your implementation of gradient descent, **make sure to use L2 regularization** with regularization parameter $\lambda = 0.001$. Please also include a ***bias term, but do not regularize it.*** Limit the number of iterations of gradient descent to 200,000, and set the learning rate to be $\eta = 0.001$.

d) **Another multi-class logistic regression classifier** with the additional feature map:

$$\phi(\boldsymbol{x}) = [\ln(x_1 + 10), x_2^2]^\top$$

e) **A kNN classifier** in which you classify based on the $k = 1$ and $k = 5$ nearest neighbors and the following distance function:

$$\text{dist}(\boldsymbol{x}, \boldsymbol{x}') = (x_1 - x_1')^2/9 + (x_2 - x_2')^2$$

where nearest neighbors are those with the smallest distances from a given point.

Note 1: When there are more than two labels, no label may have the majority of neighbors. Use the label that has the most votes among the neighbors as the choice of label.

Note 2: The grid of points for which you are making predictions should be interpreted as our test space. Thus, it is not necessary to make a test point that happens to be on top of a training point ignore itself when selecting neighbors.

After implementing the above classifiers, complete the following exercises:

1. Plot the decision boundaries generated by each classifier for the dataset. Include them in your PDF. Identify the similarities and differences among the classifiers. What explains the differences—in particular, which aspects or properties of each model dictate the shape of its decision boundary?

2. Consider a loan applicant with Debt to Income Ratio 0.32 and Credit Score 350. To which class does each classifier assign this applicant? Report the classification probabilities of this applicant for models (c) and (d).

   Interpret how each model makes its classification decision. What else should we, the modelers, be aware of when making predictions on a point "far" from our training data? **Your response should no be longer than 5 sentences.**

3. Can you think of any ethical problem that might arise from using this classifier to make loan decisions? You may approach this from any angle you like. For instance, can you think of someone who might have a low credit score and high debt-to-income ratio that you believe should nonetheless be offered a loan? Are there other variables that should be accounted for to ensure fair decisions? Are credit scores and debt-to-income ratio good bases for loan decisions? More generally, is using a classifier trained on past decisions to determine loan eligibility problematic in any way?

Solution: Your solution here.

Problem 4 (Gradient Descent and Regularization, 15pts)

In this problem, you will analyze gradient descent for linear regression and study how *ridge regularization* modifies the optimization landscape. You will first derive gradient descent updates for ordinary least squares, then extend them to the ridge-regularized objective. Finally, you will implement these algorithms to empirically observe how regularization affects convergence and generalization.

Consider a regression dataset $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$, where each input $\mathbf{x}_i \in \mathbb{R}^D$ and response $y_i \in \mathbb{R}$. Let $\mathbf{X} \in \mathbb{R}^{N \times D}$ denote the design matrix and let $\mathbf{y} \in \mathbb{R}^N$ denote the vector of responses.

Detail: The predictors in this dataset are *highly correlated*. As a result, the ordinary least squares objective may be ill-conditioned, which can lead to unstable gradient descent updates and high-variance parameter estimates. In later parts of this problem, we will examine how ridge regularization modifies the optimization landscape in this setting.

Consider the ordinary least squares objective

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2.$$

1. Derive the gradient $\nabla_{\mathbf{w}}\mathcal{L}(\mathbf{w})$ with respect to the parameter vector $\mathbf{w}$. Your final expression should be written in matrix/vector form and simplified as much as possible.

2. Using your result, write the full-batch gradient descent update rule for $\mathbf{w}$ with step size $\eta > 0$.

   Now consider the ridge-regularized objective

   $$\mathcal{L}_{\text{ridge}}(\mathbf{w}) = \frac{1}{2}\|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \frac{\lambda}{2}\|\mathbf{w}\|_2^2,$$

   where $\lambda \geq 0$ is a regularization parameter. Important note: typically, we do not regularize the bias term, w_0, thus $\|\mathbf{w}\|_2^2$ should be replace with $\mathbf{w}'$ where $w_0' = 0$. However, for the sake of simplicity, we will omit that detail for the rest of the problem.

3. Derive the gradient $\nabla_{\mathbf{w}}\mathcal{L}_{\text{ridge}}(\mathbf{w})$. Clearly indicate how the regularization term modifies the gradient compared to the unregularized case.

4. Rewrite the update in a form that makes the effect of ridge regularization explicit. Briefly explain how this form illustrates the shrinkage effect of the regularization term on the parameters.

5. You will implement gradient descent for both the ordinary least squares and ridge-regularized objectives derived above and empirically study the effect of regularization. Specifically, you will

   - Implement gradient and loss functions for both regular and ridge regression
   - Write update step functions for stochastic gradient descent, stochastic gradient descent + momentum, and Adam optimizer.

   Try not to touch anything that is not in between "*sol*" and "*los*"!

6. Generate the contour plots for the trajectories of the three algorithms across both the regularized and non-regularized case. What do you notice about the speed of convergence as well as where the algorithms converge to? What do you notice about the geometry of the loss function?
   Note: the graphing code will pause when the first descent converges, so we see that some will not converge. Also, you'll see a subspace for the optimum in the unregularized case since we have a non-full rank data matrix $\mathbf{X}$*!*

Solution: Your solution here.

Name:
Collaborators and Resources: