Homework 2: Classification and Bias-Variance Trade-offs

Introduction

This homework is about classification and bias-variance trade-offs. In lecture we have primarily focused on binary classifiers trained to discriminate between two classes. In multiclass classification, we discriminate between three or more classes. Most of the material for Problem 1 and Problem 3, and all of the material for Problem 2 will be covered by the end of the Tuesday 2/8 lecture. The rest of the material will be covered by the end of the Thursday 2/10 lecture. We encourage you to read CS181 Textbook's Chapter 3 for more information on linear classification, gradient descent, classification in the discriminative setting (covers multiclass logistic regression and softmax), and classification in the generative setting. Read Chapter 2.8 for more information on the trade-offs between bias and variance.

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 $y \in \{0, 1\}$.

In problems 1 and 3, you may use `numpy` or `scipy`, but not `scipy.optimize` or `sklearn`. Example code given is in Python 3.

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 your plots in your writeup PDF.** The supplemental files will only be checked in special cases, e.g. honor code issues, etc.

Please submit your **LaTeX file and code files to the Gradescope assignment 'HW2 - Supplemental'**.

Problem 1 (Exploring Bias and Variance, 10 pts)

In this problem, we will explore the bias and variance of a few different model classes when it comes to logistic regression.

Consider the true data generating process $y \sim \text{Bern}(f(x))$, $f(x) = 0.4 \times \sin(1.2x) + 0.5$, where $x \in [-3, 3]$, and $y \in \{0, 1\}$. Recall that for a given x, bias and variance are defined in terms of expectations *over randomly drawn datasets* D from this underlying data distribution:

$$\text{Bias}[\hat{f}(x)] = \mathbb{E}_D[\hat{f}(x)] - f(x)$$
$$\text{Variance}[\hat{f}(x)] = \mathbb{E}_D[(\hat{f}(x) - \mathbb{E}_D[\hat{f}(x)])^2]$$

Here, $\hat{f}(x)$ is our estimator (learned through logistic regression on a given dataset D). We will directly explore the bias-variance trade-off by drawing multiple such datasets and fitting different logistic regression models to each. Remember that we, the modelers, do not usually see the true data distribution. Knowledge of the true $f(x)$ is only exposed in this problem to (1) make possible the simulation of drawing multiple datasets, and (2) to serve as a pedagogical tool in allowing verification of the true bias.

1. Consider the three bases $\phi_1(x) = [1, x]$, $\phi_2(x) = [1, x, x^2]$, $\phi_3(x) = [1, x, x^2, x^3, x^4, x^5]$. For each of these bases, generate 10 datasets of size $N = 30$ using the starter code provided, and fit a logistic regression model using $\text{sigmoid}(w^T \phi(x))$ 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. Note that the classes are represented with 0's and 1's.

   Use random starting values of w, $\eta = 0.001$, take 10,000 update steps for each gradient descent run, and make sure to average the gradient over the data points (for each step). These parameters, while not perfect, will ensure your code runs in a reasonable amount of time. The emphasis of this problem is on capturing the bias-variance trade-off, so don't worry about attaining perfect precision in the gradient descent as long as this trade-off is captured in the final models.

   Note: Overflow RuntimeWarnings due to `np.exp` should be safe to ignore, if any. Also, to reduce stress from randomness in students' solutions (due to randomized weight initialization differences), in line 109 of the `T2_P1.py` starter code, we call `np.random.seed(1738)` to set a deterministic random seed. Please do not change this! In addition, please do not change the randomized weight initialization code in lines $42 - 46$.

2. Create three plots, one for each basis. Starter code is available which you may modify. By default, each plot displays three types of functions: (1) the true data-generating distribution $f(x)$ (the probability that $y = 1$ for different x). (2) all 10 of the prediction functions learned from each randomly drawn dataset, and (3) the mean of the 10 prediction functions. Moreover, each plot also displays 1 of the randomly generated datasets and highlights the corresponding prediction function learned by this dataset.

3. How are bias and variance 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.

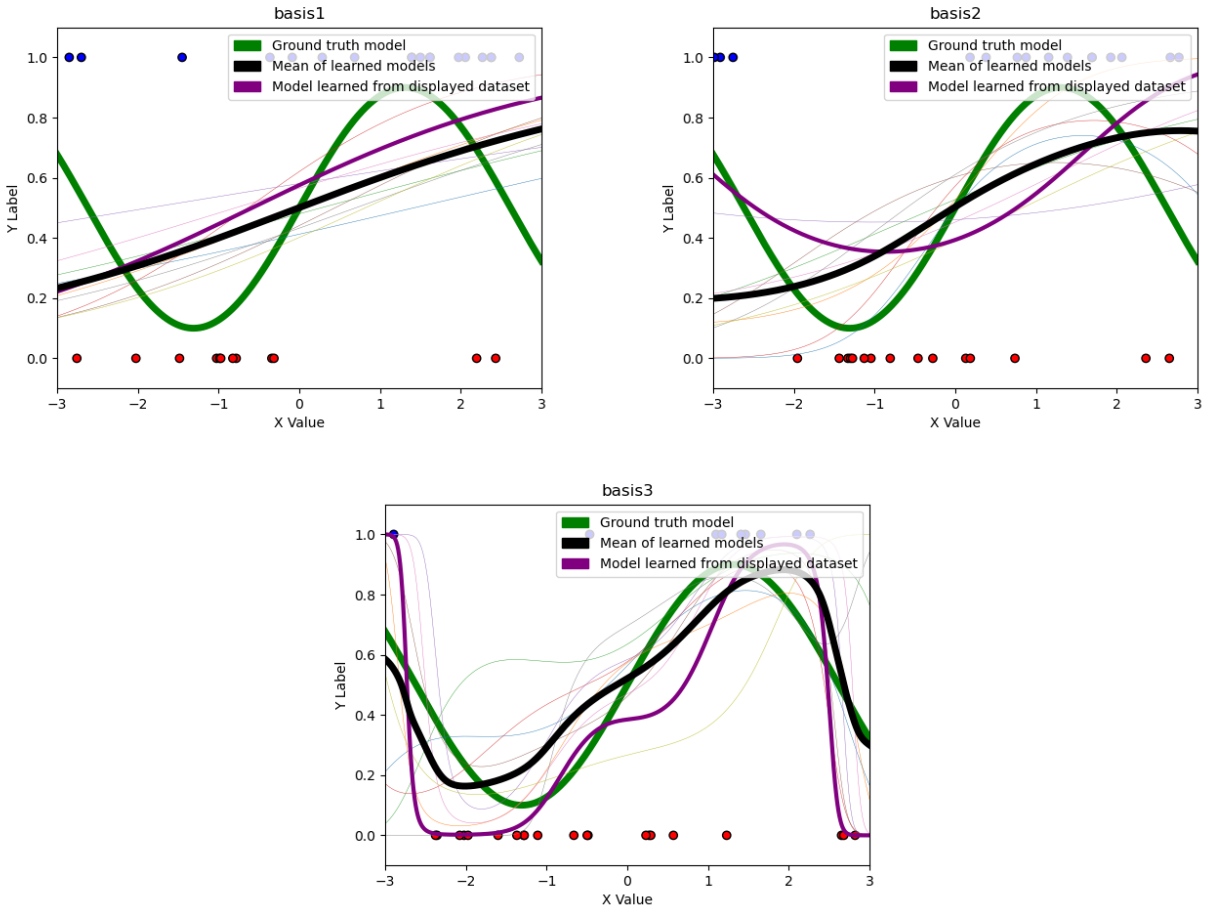
   Note: In this problem, we are not interested in whether the model is more biased for certain inputs x compared to other inputs x'. We are interested in the overall bias and variance of $\hat{f}(x)$ across the different basis choices. In other words, we want to investigate how the bias between $\hat{f}(x)$ and the ground truth as well as the variance of $\hat{f}(x)$ will be different over different basis choices.

4. If we were to increase the size of each dataset drawn from $N = 30$ to a larger number, how would the variance change? The bias? Why might this be the case?

Solution

Part 1 and 2

We want to generate three plots for each basis given in the question, where each plots shows the prediction function learned from a logistic regression on 10 datasets of size $N = 30$ using $\sigma(w^T\phi(x))$ and gradient descent to minimize the negative log likelihood with $\eta = 0.001$ and 10000 update steps. The code to generate these plots (including the gradient descent) is in `T2_P1.py` and the plots are shown below.



Part 3

As we can see from the plots above, it seems as though the bias is highest in basis one and lower in basis two and three, while variance is highest in basis three and lower in basis two and one.

The plot for basis one shows how it is highly biased (which makes sense since it is a simple model and the ground truth model is complex): despite changes in the training datasets, all of the individual prediction functions remain fairly similar and look almost linear, and the mean prediction function looks close to most of the individual prediction functions. Basis two is slightly less biased than bias one: the individual and mean prediction function seems to curve with the data generating function. However, its individual prediction functions also have higher variance, as they seem to vary more for each dataset. The mean prediction function for basis three is very close to our ground truth model, however the individual prediction functions

have very high variance and change significantly depending on the dataset.

It seems that basis three represents the true process the best. The mean of learned models for basis three almost directly overlays the ground truth model. Basis two curves with the ground truth model, but does not seem to match the amplitude of the ground truth model. Basis one looks almost linear, and so is not a good approximation of the true process.

Part 4

If we increased the size of each dataset, I think that variance would decrease overall. When there are few data points, changing some of the data can cause greater changes in the predictions than when there are a lot of data points. This causes less variance when the size of the training dataset increases.

If we increased the size of each dataset, I think that the bias here would increase overall. Bias stems from having too simple models that cannot accurately model the data. As the size of the dataset increases and the data more accurately reflects the truth, our models would increasingly not be able to reflect this truth due to their simplicity. And so bias would increase.

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 $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ 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, $\ln p(D; \boldsymbol{\pi})$.
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: $\mathbf{X}^{-\top} := (\mathbf{X}^\top)^{-1}$

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

Solution

Part 1

We want to write out the log-likelihood of the data set, $\ln p(D; \boldsymbol{\pi})$. First let us find the likelihood, $L(\boldsymbol{\pi}; D)$. We have the following general equation for our likelihood:

$$L(\boldsymbol{\pi}; D) = \prod_{i=1}^{N} p(\mathbf{x}_i | \mathbf{y}_i = C_k) p(\mathbf{y}_i = C_k; \boldsymbol{\pi})$$

Now subbing-in the information from the question and taking the log gives us the following log-likelihood, where $\mathbb{I}$ is an indicator variable.

$$\ell(\boldsymbol{\pi}; D) = \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}(\mathbf{y}_i = C_k)(\ln(p(\mathbf{x}_i | \mathbf{y}_i = C_k)) + \ln(\pi_k))$$

And so we have found our log-likelihood of the dataset.

Part 2

We want to write out the MLE for the prior class membership probabilities $\hat{\pi}_k$. We will do this by creating a new function L using Lagrange multipliers to account for the constraint $\sum_{k=1}^{K} \pi_k - 1 = 0$, take the derivative of L with respect to π_k, set to 0, and solve for π_k.

First let us create L as below. Lagrange multipliers allow us to optimize a function given an equality constraint of the form:

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

To use Lagrange multipliers to maximize our log-likelihood, we thus multiply our log-likelihood by -1 (minimizing the negative log-likelihood is equivalent to maximizing the log-likelihood). Subbing in our f, g thus gives us the following.

$$\begin{aligned} L(\boldsymbol{\pi}, \lambda) &= -\ell(\boldsymbol{\pi}; D) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \\ &= - \left[\sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}(\mathbf{y}_i = C_k)(\ln(p(\mathbf{x}_i | \mathbf{y}_i = C_k)) + \ln(\pi_k)) \right] + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right) \end{aligned}$$

Now let us take the derivative of L with respect to π_k.

$$\frac{\partial L(\boldsymbol{\pi}, \lambda)}{\partial \pi_k} = -\sum_{i=1}^{N} \frac{1}{\pi_k} + \lambda$$

Let N_k be the number of times the class C_k appears in our data D. Then setting the LHS to 0, and solving for π_k gives us the following expression for π_k.

$$\begin{aligned} \frac{\partial L(\boldsymbol{\pi}, \lambda)}{\partial \pi_k} &= -\sum_{i=1}^{N} \frac{1}{\pi_k} + \lambda \pi_k \\ 0 &= -\frac{N_k}{\pi_k} + \lambda \\ \pi_k &= \frac{N_k}{\lambda} \end{aligned}$$

Now let us take the derivative of L with respect to λ and set the LHS to 0.

$$\begin{aligned}\frac{\partial L(\boldsymbol{\pi}, \lambda)}{\partial \lambda} &= \sum_{k=1}^{K} \pi_k - 1 \\ 0 &= \sum_{k=1}^{K} \pi_k - 1 \\ \sum_{k=1}^{K} \pi_k &= 1\end{aligned}$$

Then subbing our equation for π_k into this equation gives us the following expression for λ.

$$\begin{aligned}\sum_{i=1}^{K} \frac{N_i}{\lambda} &= 1 \\ \frac{1}{\lambda} \sum_{i=1}^{K} N_i &= 1 \\ \lambda &= \sum_{i=1}^{K} N_i\end{aligned}$$

Finally, subbing this into our equation for π_k above gives us the final expression for π_k.

$$\begin{aligned}\pi_k &= \frac{N_k}{\lambda} \\ \pi_k &= \frac{N_k}{\sum_{i=1}^{K} N_i}\end{aligned}$$

And so our MLE for prior class-membership probabilities, $\hat{\pi}_k$ is $\hat{\pi}_k = N_k / \sum_{i=1}^{K} N_i$. This is intuitive because the prior probability of each class should be the proportion of each class in the data.

Part 3

We want to derive the gradient of the log-likelihood with respect to vector $\boldsymbol{\mu}_k$, where we know from the question that the class-conditional probabilities are Gaussian.

Let us first sub-in the multivariate Gaussian PDF into our likelihood function and simplify.

$$\begin{aligned}\ell(\boldsymbol{\pi}; D) &= \sum_{i=1}^{N} \sum_{k}^{K} \mathbb{I}(\mathbf{y}_i = C_k) \left[\ln\left((\det(2\pi\boldsymbol{\Sigma})^{-1/2}) \exp\left(-\frac{1}{2} (\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) \right) \right) \right] \\ &= -\frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{K} \mathbb{I}(\mathbf{y}_i = C_k) \left[\ln(\det(2\pi\boldsymbol{\Sigma})) + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_k) \right]\end{aligned}$$

Before we take the gradient, first note that by equation 86 from the Matrix Cookbook the following is true for a symmetric matrix W.

$$\frac{\partial}{\partial \mathbf{s}} (\mathbf{x} - \mathbf{s})^T \mathbf{W} (\mathbf{x} - \mathbf{s}) = -2\mathbf{W}(\mathbf{x} - \mathbf{s})$$

We can use this fact to take the derivative with respect to $\boldsymbol{\mu}_k$ since our covariance matrix is symmetric. Now let us take the gradient with respect to $\mathbf{u}_k$.

$$\begin{aligned}\nabla_{\boldsymbol{\mu}_k}\ell(\boldsymbol{\pi}; D) &= -\frac{1}{2}\sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)(-2\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)) \\ &= \sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\end{aligned}$$

And so we have found the gradient with respect to $\boldsymbol{\mu}_k$.

Part 4

Now we want to derive a MLE $\hat{\mu}_k$ for the vector $\boldsymbol{\mu}_k$. We will do this by setting the gradient with respect to $\boldsymbol{\mu}_k$ from Part 3 to 0 and then solving for $\boldsymbol{\mu}_k$.

Let us begin by setting the gradient to 0 and pulling out the inverse of the covariance matrix from the summation.

$$\begin{aligned}\nabla_{\boldsymbol{\mu}_k}\ell(\boldsymbol{\pi}; D) &= \sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) \\ 0 &= \boldsymbol{\Sigma}^{-1}\sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)\end{aligned}$$

Then multiplying both sides by the covariance matrix and expanding yields the following.

$$\begin{aligned}0 &= \sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)(\mathbf{x}_i - \boldsymbol{\mu}_k) \\ 0 &= \sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)\mathbf{x}_i - \mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\mu}_k\end{aligned}$$

Let N_k be the number of times the class C_k appears in our dataset. Then the expression can be simplified as follows.

$$\begin{aligned}0 &= \sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)\mathbf{x}_i - \mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\mu}_k \\ 0 &= -N_k\boldsymbol{\mu}_k + \sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)\mathbf{x}_i \\ \boldsymbol{\mu}_k &= \frac{\sum_{i=1}^{N}\mathbb{I}(\mathbf{y}_i = C_k)\mathbf{x}_i}{N_k}\end{aligned}$$

And so we have found the MLE for $\boldsymbol{\mu}_k$. This is intuitive because $\boldsymbol{\mu}_k$ should be the average of the features observed for the given class C_k.

Part 5

We want to derive the gradient for the log-likelihood with respect to the covariance matrix $\boldsymbol{\Sigma}$.

Let us start with the simplified log-likelihood after subbing in the multivariate Gaussian PDF from Part 3, and simplify a bit more.

$$\begin{aligned}\ell(\boldsymbol{\pi}; D) &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)\left[\ln(\det(2\pi\boldsymbol{\Sigma})) + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right] \\ &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)\ln(2\pi) + \mathbb{I}(\mathbf{y}_i = C_k)\ln(\det(\boldsymbol{\Sigma})) + \mathbb{I}(\mathbf{y}_i = C_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\end{aligned}$$

Now taking the gradient with respect to $\boldsymbol{\Sigma}$ gives the following.

$$\nabla_{\boldsymbol{\Sigma}}\ell(\boldsymbol{\pi}; D) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\nabla_{\boldsymbol{\Sigma}}\left[\mathbb{I}(\mathbf{y}_i = C_k)\ln(\det(\boldsymbol{\Sigma}))\right] + \nabla_{\boldsymbol{\Sigma}}\left[\mathbb{I}(\mathbf{y}_i = C_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right]$$

Using the Cookbook formulas given in the question, we can now simplify this expression. Note that we can use these formulas because $\boldsymbol{\Sigma}$ is symmetric. Also note that because the covariance matrix is symmetric, $\boldsymbol{\Sigma}^{-T} = \boldsymbol{\Sigma}^{-1}$.

$$\nabla_{\boldsymbol{\Sigma}}\ell(\boldsymbol{\pi}; D) = -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1} + \mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}$$

We can then simplify this further.

$$\nabla_{\boldsymbol{\Sigma}}\ell(\boldsymbol{\pi}; D) = -\frac{1}{2}N\boldsymbol{\Sigma}^{-1} - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}$$

And so we have found the gradient with respect to $\boldsymbol{\Sigma}$.

Part 6

We want to find the MLE of $\boldsymbol{\Sigma}$. We will do this by using the gradient with respect to $\boldsymbol{\Sigma}$ from Part 5, setting the LHS to 0, and solving for $\boldsymbol{\Sigma}$.

$$\begin{aligned}0 &= -\frac{1}{2}N\boldsymbol{\Sigma}^{-1} - \frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1} \\ \frac{1}{2}N\boldsymbol{\Sigma}^{-1} &= -\frac{1}{2}\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}\end{aligned}$$

Now we multiply by $\boldsymbol{\Sigma}$ from both the left and the right for the LHS and the RHS, multiply both sides by 2, and solve for $\boldsymbol{\Sigma}$.

$$\boldsymbol{\Sigma} = \frac{\sum_{i=1}^{N}\sum_{k=1}^{K}\mathbb{I}(\mathbf{y}_i = C_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}{N}$$

And so we have found our MLE for $\boldsymbol{\Sigma}$.

Problem 3 (Classifying Stars, 15pts)

You're tasked with classifying three different kinds of stars using their magnitudes and temperatures. See star.png for a plot of the data, adapted from http://astrosci.scimuze.com/stellar_data.htm and available as `data/hr.csv`, which you will find in the Github repository.

The CSV file has three columns: type, magnitude, and temperature. The first few lines look like this:

```
Type,Magnitude,Temperature
Dwarf,-5.8,-0.35
Dwarf,-4.1,-0.31
...
```

In this problem, you will code up 4 different classifiers for this task:

a) **A three-class generalization of logistic regression**, also known as softmax regression, in which you implement gradient descent on the negative log-likelihood. In Question 2 you will explore the effect of using different values for the learning rate η (`self.eta`) and regularization strength λ (`self.lam`). Make sure to include a bias term and to use L2 regularization. See CS181 Textbook's Chapter 3.6 for details on multi-class logistic regression and softmax. For your implementation, use the loss and gradient expressions provided there.

b) **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.

c) **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.)

d) **A kNN classifier** in which you classify based on the $k = 1, 3, 5$ nearest neighbors and the following distance function:

$$dist(star_1, star_2) = ((mag_1 - mag_2)/3)^2 + (temp_1 - temp_2)^2$$

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?

2. For logistic regression only, make a plot with "Number of Iterations" on the x-axis and "Negative Log-Likelihood Loss" on the y-axis for several configurations of the hyperparameters η and λ. Specifically, try the values 0.05, 0.01, and 0.001 for each hyperparameter. Limit the number of gradient descent iterations to 200,000. What are your final choices of learning rate (η) and regularization strength (λ), and why are they reasonable? How does altering these hyperparameters affect the ability to converge, the rate of convergence, and the final loss (a qualitative description is sufficient)? You only need to submit one plot for your final choices of hyperparameters.

   Note: The *likelihood* of the model is the probability of data given the model—it should not include the regularization term. The *objective* is the combination of the likelihood and the regularizer.

3. For both Gaussian generative models, report the negative log-likelihood loss. Which model has a lower loss, and why? For the separate covariance model, be sure to use the covariance matrix that matches the true class of each data point.

4. Consider a star with Magnitude 6 and Temperature 2. To what class does each classifier assign this star? Do the classifiers give any indication as to whether or not you should trust them?

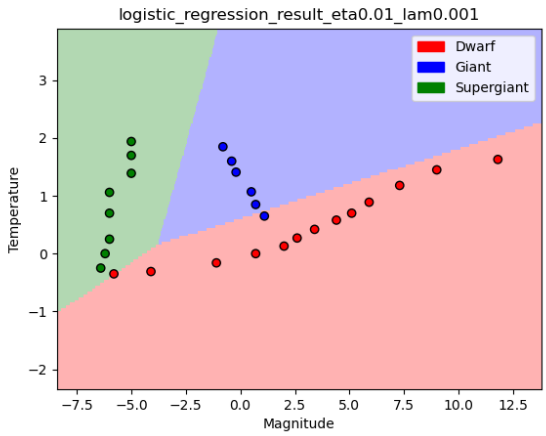
Problem 3 (cont.)

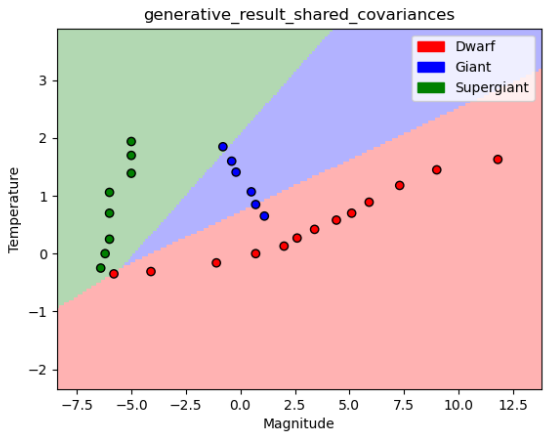
Implementation notes: Run the controller file, `T2_P3.py`, to test your code. Write the actual implementations in the `GaussianGenerativeModel`, `LogisticRegression`, and `KNNModel` classes, which are defined in the three `T2_P3_ModelName.py` files. These classes follow the same interface pattern as sklearn. Their code currently outputs nonsense predictions just to show the high-level interface, so you should replace their `predict()` implementations. You'll also need to modify the hyperparameter values in `T2_P3.py` for logistic regression.

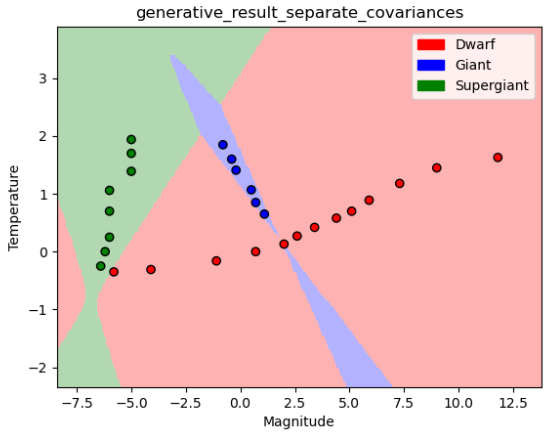
Solution

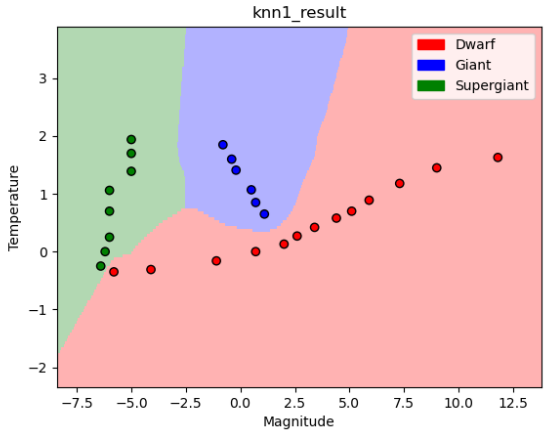
Part 1

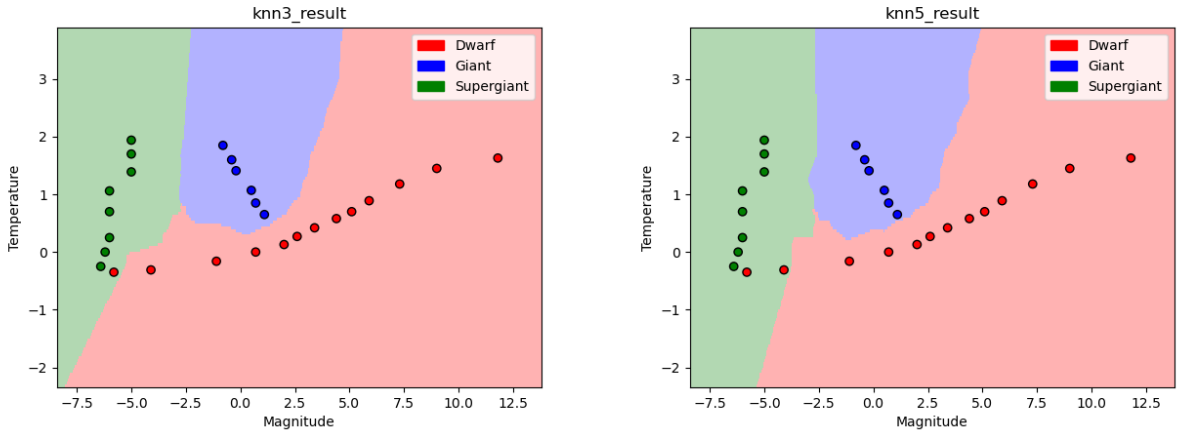
We want to plot the decision boundaries generated by each classifier for the dataset. The plots are shown below.











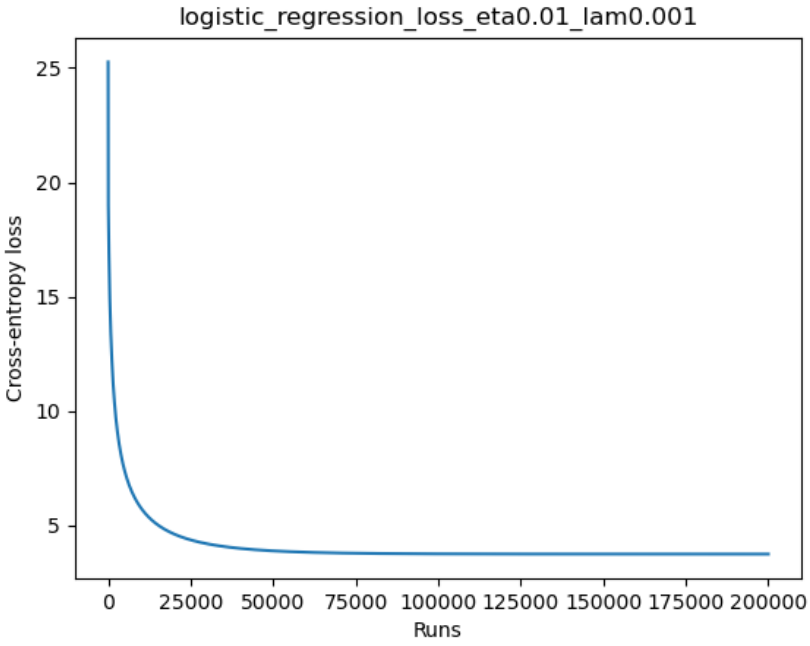
Let us identify some similarities and differences. As we can see, the KNN decision boundaries are all fairly similar across the choices of k, with the difference being that the green area expands as k increases. The logistic regression and generative (shared) also have somewhat similar decision boundaries, albeit the generative (shared) decision boundary for blue is more squished. The generative (separate) is the most unlike the others, and has almost (as another student mentioned in Ed) "hourglass" shaped decision boundaries.

The differences for KNN are explained by the increasing size of k and the geometric layout of the points: as we can see, the green points are more squished together than the red points, and so as k increases more green points are closer than red points in these areas.

The differences between the generative (shared) and generative (separate) can be explained by the effect of sharing versus having separate covariance matrices. When there are separate covariance matrices, the model effectively has more parameters and so can make more fine-grained (and perhaps more overfit) predictions for each class. When working with one shared covariance matrix, the predictions must be more general.

Part 2

For our logistic regression, we want to plot the negative log-likelihood loss against the number of iterations.



I chose $\eta = 0.01$ and $\lambda = 0.001$. This combination of hyperparameters yielded the lowest cross-entropy loss (around 3.59), converged relatively fast, and visually seemed to fit the data best. As η increases, the loss converges faster but often converges to a higher error, and increasing λ generally caused increases in error.

Part 3

The negative log-likelihood loss for the shared and separate covariance matrices are given in the table below.

Type	Negative log-likelihood loss
Shared	116.394
Separate	63.970

The separate covariance matrices has lower loss. I think that this is because when using separate covariance matrices, the model essentially has more parameters and so can give more fine-grained (and perhaps more over-fit) predictions for each class. This is shown in the plots, where we can see how the separate covariance predictions are more granular for each class than the shared covariance plot.

Part 4

We want to consider what class each classifier would assign a star with magnitude 6 and temperature 2. The class predictions for each classifier are given below.

Classifier	Star classification
Logistic	1
Gaussian generative (shared)	1
Gaussian generative (separate)	0
KNN (k=1)	0
KNN (k=3)	0
KNN (k=5)	0

I think that the KNN classifiers are all reasonably trustworthy for making predictions on new data that is close to the clusters of the other data, however I trust it less for data that is not close to these clusters. I think that the generative classifiers and logistic classsifiers inherently give some indication as to their trustworthiness since they are probabilistic (and so we can perhaps see the degree of certainty of each prediction for a class relative to other classes), but the separate covariance classifier seems to slightly overfit the data, perhaps decreasing its trustworthiness. The logistic and generative (shared) classifiers also seem fairly generalized, which makes me trust them more. Regarding the prediction for this new star, magnitude 6 and temperature 2 is somewhat in between two classes so it is worth understanding how sure we are about this prediction.

Name

Collaborators and Resources

Whom did you work with, and did you use any resources beyond cs181-textbook and your notes? James Kitch, Julian Schmitt, Bishop textbook (referred to in Ed by TFs), Ed, and the Matrix Cookbook.

Calibration

Approximately how long did this homework take you to complete (in hours)? 25