Aidan Tai
aidantai@college.harvard.edu
CS181-S23

Assignment #2
Due: 11:59pm EST, Feb 23th, 2023

Homework 2: Classification and Bias-Variance Trade-offs

Introduction

This homework is about classification, bias-variance trade-offs, and uncertainty quantification. 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. We encourage you to read CS181 Textbook's Chapter 3 for more information on linear classification, gradient descent, and classification in the discriminative setting. Read Chapter 2.8 for more information on the trade-offs between bias and variance.

The datasets that we will be working with relate to astronomical observations. 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 kinds of stars and their measured magnitude and temperature. 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 $y \in \{0, 1\}$.

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-Variance and Uncertainty)

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

We are currently managing a powerful telescope that is being used to monitor and gather measurements of some planet of interest. At certain times however, our telescope is unable to detect the planet at all. 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). Since it is expensive to use and maintain the telescope, we would like to build a model to help us schedule and find times when we are likely to detect the planet.

1. First split the data into 10 mini-datasets of size $N = 30$ (i.e. dataset 1 consists of the first 30 observations, dataset 2 consists of the next 30, etc. This has already been done for you). Consider the three bases $\boldsymbol{\phi}_1(t) = [1, t]$, $\boldsymbol{\phi}_2(t) = [1, t, t^2]$, and $\boldsymbol{\phi}_3(t) = [1, t, t^2, t^3, t^4, t^5]$. For each of these bases, fit a logistic regression model using 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 10,000 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.

2. 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 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.

3. 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**.

4. 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 $t = 3.2$. At which point in time would you be more confident in detecting the planet? There's no right answer—you should consider the two different types of uncertainty and their implications when translating from model output to decision making.

Problem 2 (Multi-class Logistic Regression and Softmax)

The objective of this problem is to generalize binary logistic regression into the more general case of three or more classes. You will use the results of this problem to implement a classifier in Problem 3.

Consider a K-class model with $K \geq 3$. Suppose we have a data set $\{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ with features $\{\mathbf{x}_n\}_{n=1}^N \in \mathbb{R}^d$ and one-hot encoded outputs $\{\mathbf{y}_n\}_{n=1}^N \in \mathbb{R}^K$ (see the introduction of this homework). For a K-dimensional vector $\mathbf{z} = [z_1, \ldots, z_K]^\top$, define the *softmax* function to be

$$\text{softmax}(\mathbf{z}) = \frac{1}{\sum_{i=1}^K \exp(z_i)} \begin{bmatrix} \exp(z_1) \\ \exp(z_2) \\ \vdots \\ \exp(z_K) \end{bmatrix}.$$

In other words, the softmax function is a function from $\mathbb{R}^K$ to $\mathbb{R}^K$ with k-th component of the output

$$\text{softmax}_k(\mathbf{z}) = \frac{\exp(z_k)}{\sum_{i=1}^K \exp(z_i)}.$$

We will use the shorthand notation $s_k(\mathbf{z})$ to abbreviate the above. Note that the softmax function is the general form of the sigmoid function in binary logistic regression. This means that the derivations in this problem will be very similar to what you have seen in class.

1. For $j, k \in \{1, \ldots, K\}$, show that the partial derivatives of the softmax function can be written in terms of the softmax function itself in the following form:

$$\frac{\partial s_k(\mathbf{z})}{\partial z_j} = s_k(\mathbf{z})(\delta_{jk} - s_j(\mathbf{z}))$$

   Here, δ_{jk} denotes the *Kronecker delta* function δ_{jk}:

$$\delta_{jk} = \begin{cases} 1 \text{ if } j = k, \\ 0 \text{ if } j \neq k. \end{cases}$$

   Since all of these partial derivatives are with respect to scalars, you can use the typical quotient rule from your univariate calculus class. It may help to consider the cases $j = k$ and $j \neq k$ separately.

   Using the answer above, find the logarithmic derivatives of softmax($\mathbf{z}$); that is, find $\dfrac{\partial \ln s_k(\mathbf{z})}{\partial z_j}$ for $j, k \in \{1, \ldots, K\}$.

Multi-class logistic regression has weights $\{\mathbf{w}_j\}_{j=1}^K \in \mathbb{R}^d$ for each class, which are often condensed into a single matrix $\mathbf{W} \in \mathbb{R}^{K \times d}$ (the j-th row of $\mathbf{W}$ corresponds to $\mathbf{w}_j$). We model the probabilities of class membership independently as

$$p(\mathbf{y}_n = C_k | \mathbf{x}_n, \mathbf{W}) = s_k(\mathbf{W}\mathbf{x}_n)$$

for $k \in \{1, \ldots, K\}$ and $n \in \{1, \ldots, N\}$. In addition, let y_{nk} denote the k-th component of the output vector $\mathbf{y}_n$.

2. Write out the negative log-likelihood of the data set, $\ell(\mathbf{W}) = -\ln p(\{\mathbf{y}_n\}_{n=1}^N | \{\mathbf{x}_n\}_{n=1}^N, \mathbf{W})$, in terms of s_k, $\mathbf{W}$, $\{\mathbf{x}_n\}_{n=1}^N$, and y_{nk}. You can start by noting that for a single observation $(\mathbf{x}_n, \mathbf{y}_n)$,

$$p(\mathbf{y}_n | \mathbf{x}_n, \mathbf{W}) = \prod_{k=1}^K p(\mathbf{y}_n = C_k | \mathbf{x}_n, \mathbf{W})^{y_{nk}}$$

   because $y_{nk} = 1$ if $\mathbf{y}_n$ belongs to class C_k and $y_{nk} = 0$ otherwise. The equation above simply lets us combine all possible class memberships of $\mathbf{y}_i$ into a single expression. This is also known as the "power trick" as we express the probability as a product of terms raised to a power that is either 0 or 1.

Problem 2 (cont.)

3. Consider the weight matrix $\mathbf{W}$ and the i-th feature vector $\mathbf{x}_i$. Denote their product as $\mathbf{z}_i = \mathbf{W}\mathbf{x}_i$. Compute the derivative of $\ell(\mathbf{W})$ with respect to the j-th coordinate of the K-dimensional vector $\mathbf{z}_i$ (denoted as z_{ij}). In particular, show that

$$\frac{\partial \ell}{\partial z_{ij}} = \sum_{k=1}^{K} y_{ik}(s_j(\mathbf{z}_i) - \delta_{kj})$$

   You may want to use your answer from part 1. Then, show that the above sum can be simplified:

$$\sum_{k=1}^{K} y_{ik}(s_j(\mathbf{z}_i) - \delta_{kj}) = s_j(\mathbf{z}_i) - y_{ij}.$$

   It will help to consider the case $k = j$ separately again and remove the delta function from the equation.

4. Conclude that the gradient of negative log-likelihood with respect to a single weight vector $\mathbf{w}_j$ is given by the *vector*

$$\frac{\partial \ell}{\partial \mathbf{w}_j} = \sum_{n=1}^{N} (s_j(\mathbf{W}\mathbf{x}_n) - y_{nj})\mathbf{x}_n$$

   for $j \in \{1, \ldots, K\}$. This can be done by using the chain rule:

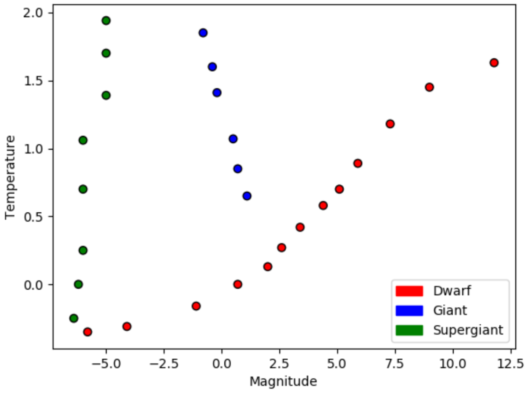
$$\frac{\partial \ell}{\partial \mathbf{w}_j} = \sum_{n=1}^{N} \sum_{k=1}^{K} \frac{\partial \ell}{\partial z_{nk}} \frac{\partial z_{nk}}{\partial \mathbf{w}_j}.$$

   We found the first derivative in part 3. What do we know about the second derivative when $k \neq j$? You can start by expressing z_{nk} in terms of the vectors $\{\mathbf{w}_i\}_{i=1}^{K}$ and $\{\mathbf{x}_i\}_{i=1}^{N}$.

   We can use this final expression to optimize the weights via gradient descent!

Problem 3 (Classifying Stars)

In this problem, you will code up three different classifiers to classify different types of stars. The file `data/hr.csv` contains data on magnitude and temperature. The data can be plotted on these two axes:



Please implement the following classifiers in the `SoftmaxRegression` and `KNNClassifier` classes:

a) **A multi-class logistic regression classifier** using the softmax activation function, which you investigated in Problem 2. In your implementation of gradient descent, **make sure to include a bias term and use L2 regularization** with regularization parameter $\lambda = 0.001$. Limit the number of iterations of gradient descent to 200,000, and set the learning rate to be $\eta = 0.001$.

b) **Another multi-class logistic regression classifier** with feature map $\phi(\mathbf{x}) = [\ln(x_1 + 10), x_2^2]^\top$, where x_1 and x_2 represent the values for magnitude and temperature, respectively.

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

$$dist(star_1, star_2) = (mag_1 - mag_2)^2/9 + (temp_1 - temp_2)^2$$

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—in particular, which aspects or properties of each model dictate the shape of its decision boundary?

2. Consider a star with Magnitude 3 and Temperature -2. To which class does each classifier assign this star? Report the classification probabilities of this star for each model.

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

Problem 4 (Impact Question: Understanding model-assisted decision making, uncertainty in classification and model interpretation in a high-stakes situation)

Prompt: A pharmaceutical drug company is conducting a clinical drug trial for a devastating disease for which conventional treatment is often ineffective. They want to estimate the effectiveness of a new drug on patients in order to obtain FDA approval and release the drug to the market. They approach you with the results of their clinical trial conducted on 100 patients with features and labels as follows:

Features = {*age, sex, height, blood pressure, drug administered?*}
Label = {*Did the patient get cured?*}

Since testing on a larger patient population is expensive for various reasons, they are interested in developing a machine learning model that will estimate the effectiveness of the drug. They provide you with covariate values of a single unseen patient for testing model performance.

1. You fit a logistic regression model on the 100 observations from the clinical trial and obtain the following coefficients which minimize the negative log likelihood. Let us call this model A:

   $p(y = 1|\mathbf{x}) = \sigma(0.2 + 0.8 * \textit{drug administered?} - 0.012 * \textit{age} + 0.45 * \textit{sex} + 0.001 * \textit{height} - 0.007 * \textit{blood pressure})$

   Say you have a female (coded as 0) patient who is age 50, 168cm tall, blood pressure 140. What is the change in classification probability of the patient getting cured when they are administered the drug versus not (please show your work)? Use your answer to formulate an interpretation of the value of w_1 – the coefficient for *drug administered?* – for stakeholders in the drug company. The drug company wants you to answer whether or not you see evidence for the efficacy of their drug – what would you say?

2. The drug company wants to know you how confident you are that you have the right model, so you decide to sample the existing dataset with replacement to train 100 bootstrapped models. Upon testing these 100 models on the unseen patient, you find that the predictive interval of the classification probability is around ±0.35 averaged over the bootstrapped models. The drug company is concerned and asks you to check if you can choose alternative models that are more confident in their predictions:

   (a) You first try adding some interaction terms and train this new model B on the original dataset:

   $p(y = 1|\mathbf{x}) = \sigma(-2 + 5 * \textit{drug administered?} + 2 * \textit{age} - 3.3 * \textit{sex} + 0.001 * \textit{height} + 0.2 * \textit{blood pressure} - 0.12 * \textit{age} * \textit{sex} - 0.34 * \textit{height} * \textit{sex})$

   Bootstrapping model B 100 times gives you a new predictive interval of ±0.1 (averaged over bootstrapped models). Why might this be happening – how would you explain this reduction in uncertainty in model B?

   (b) Encouraged by the success of adding interaction terms, you add all possible combinations of interaction terms, repeat the exercise of bootstrapping 100 times and call this model C. This gives you a predictive interval of ±0.39 averaged over the bootstrapped models. Why might this be happening – what is the source of this rise in uncertainty? What is a solution to reduce this comparatively large uncertainty, if you wished to keep all the new terms?

   (c) Which model (B or C, assuming you can reduce the predictive interval for model C) would you recommend to the drug company and why?

3. Assume that the drug company has picked model B as their final model of choice because it seems the most confident in its predictions.

   (a) Suppose that the confidence interval of your estimate of w_1 (coefficient for *drug administered*?) is ±6. What would you recommend to a critically ill patient who is desperately seeking access to this new (and very expensive, not-covered-by-insurance) drug and why?

   (b) The drug company has strong reasons to believe that the drug is indeed effective. What can the drug company do during the clinical trial to tighten the confidence interval of w_1?

Problem 4 (cont.)

3. (Continued...)

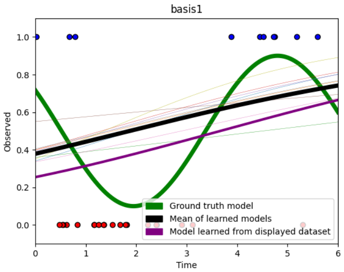
   (c) From prior clinical trials and modeling efforts, the drug company scientists strongly believe that the real value of w_1 is either 5 or 1. For now, you take this information as ground truth. In which scenario ($w_1 = 1$ or $w_1 = 5$) is it more costly to run the drug trial and demonstrate the effectiveness of the drug? Why?
   Hint: In which case do you need your confidence intervals for the estimates of w_1 to be tighter, to demonstrate drug effectiveness?

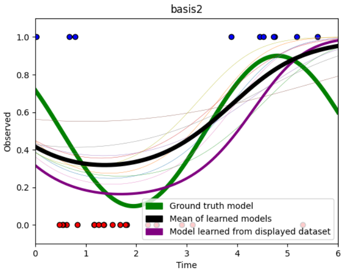
   (d) How do you think we should make use of domain knowledge provided by our partners – that the real value of w_1 is either 5 or 1 – during model building? For example, would it be a good idea for us to simply fix the parameter w_1 to be 5 or 1?

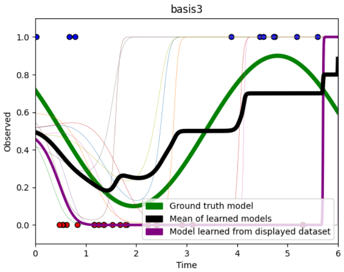
4. Let us revisit the data collection process during the clinical trial and consider data collected from two recruitment protocols. In the first recruitment protocol, the drug company publicly advertised the development of this drug to a hospital, encouraging interested patients to sign up for the trial. Among an audience of 100, 50 patients agreed to be administered the drug and 50 choose to opt out and are observed in the study without administering the drug. In the second recruitment protocol, the company was referred 100 patients who have the disease and administered the drug to each patient randomly based on a coin flip. When you train model B on the first dataset, it has a high value of w_1 and a small confidence interval of this estimate. In contrast, when you train model B on the second dataset, you get a smaller value for w_1 with an equally small confidence interval. Which model would you trust to predict the probability of being cured from the disease given a randomly selected new patient from a Boston hospital? Why? *Hint:* What confounding factor might we have here?

Solution 1

1. See code



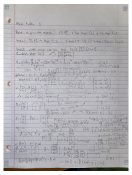




2. Basis1 has a very straight underfit curve that does not predict the training data very well, with a relatively good variance (low variance) at the cost of a problematic bias (high bias). Meanwhile, basis3 has the opposite problem: the model is highly overfit to the training data with a very low bias at the cost of a high variance and poor generalizability. Basis2 is much better than the other two, balancing bias and variance to best represent the model. Basis3 has a significant difference between each iteration of gradient descent–however, the mean of these iterations actually gets reasonably close to the ground truth model. Meanwhile, the iterations of basis1 are very similar because of the simplicity of the basis restricts its variance.

3. The total variance would decrease because model variance is dependent on the variance of the training data. Theoretically, adding more training data to the model would allow it to generalize the trend of the data better, which would involve a reduction in the model's variance error because more data means less uncertainty about predictions. Increasing training data should not have a significant impact on the bias because the bias is not directly dependent on the variance of the training set, but rather how well the model can capture the ground truth trend. While the certainty in the prediction can improve from more data, improvement in the true accuracy of the prediction is not guaranteed.

4. The basis3 first model's prediction for $t = 0.1$ is 0.51992796. Since the value is very close to 50%, it could be interpreted as having a low confidence/high uncertainty in how to classify this point. The variance across the 10 models for this point is var: 0.003429945554968907. This low variance indicates that the model has a high confidence/low uncertainty about this 50% prediction. In other words, the model is highly confident that it is uncertain about whether to classify the point as a 0 or a 1. The first uncertainty comes from variance in the data , while the second uncertainty (or rather, certainty) comes from low variance in the model. For the point $t = 3.2$, the first group of models have a very "strong" prediction near 0%. This indicates the first type of "certainty", in that the prediction is extremely confident in its prediction. However, the latter models suddenly start predicting close to 100%, showing the same extreme confidence, but now for the opposite classification. This creates a very high variance across the 10 models (var : 0.2499555618600618) which comes from high variance in the model (this is because it is overfit/overtuned for bias)



Problem 1 notes (unimportant) can be found in supplementals

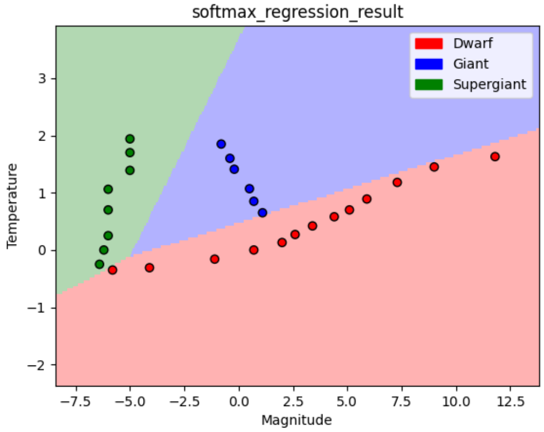
End Solution

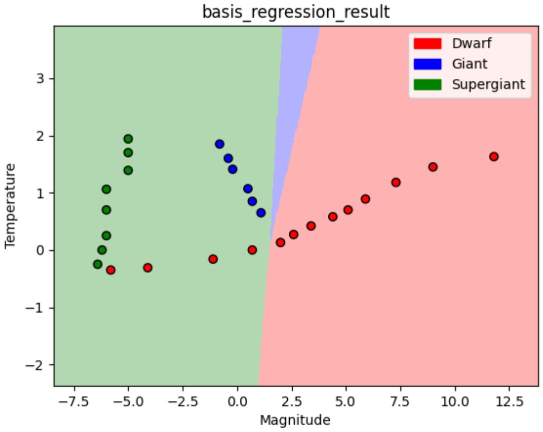
Solution 2

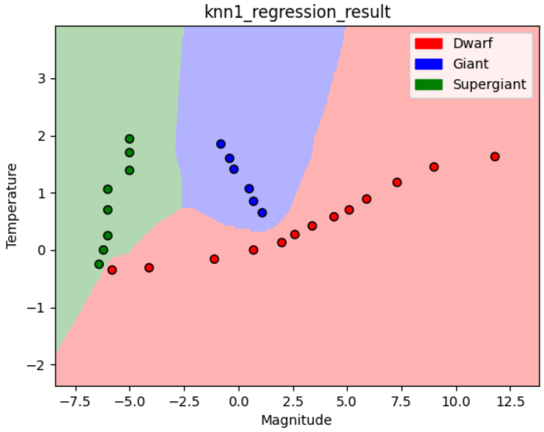
1.

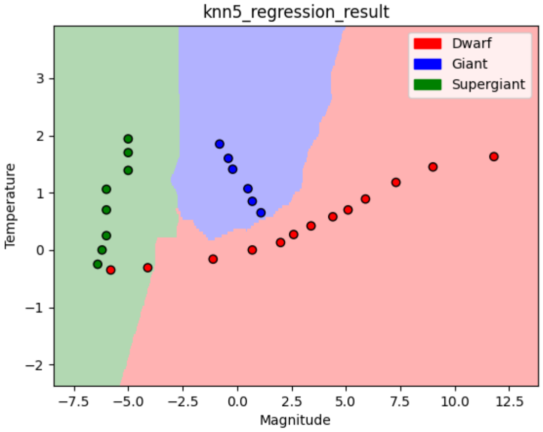
End Solution

Solution 3



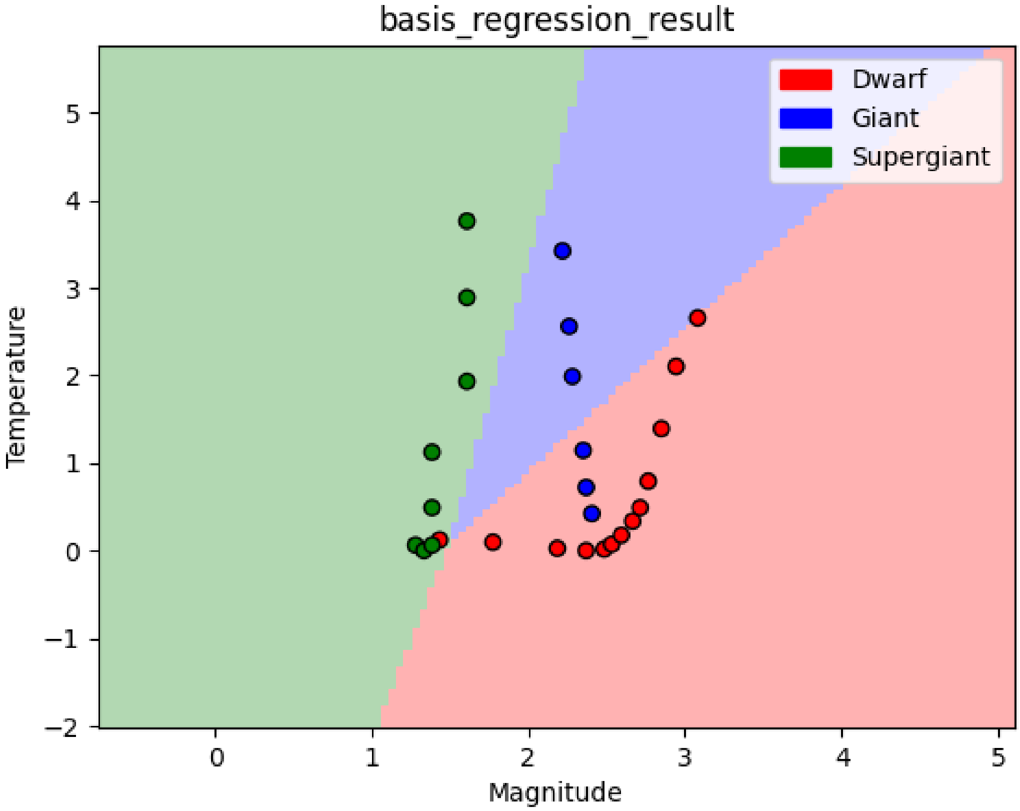






1. Since there are three classes for two dimensions, we can see that the softmax regressions divide up the 2D space with three intersecting 2D vectors, assigning a class to the three regions that result from these boundaries. By comparing the first softmax with the phi basis softmax, we can see that the basis model has a ”preference” toward the magnitude value in determining the boundary. We see this by the fact that the three vectors are extremely steep, which means that manipulating the Temperature feature of an observation to the point where its classification changes would require a significantly bigger numerical displacement compared to manipulating the Magnitude feature. We can understand this by considering the basis used for the basis_model: $[\ln(x_1 + 10), x_2^2]$.

   Below is a visualization of the basis model with the basis applied to the visualization's feature space:



We can see that the model's regression appears intuitive under this visualization, while the other visualization appears bizarre. Since the basis is "shrinking" the magnitude values and "expanding" the temperature values, running the basis_model's regression on new data without applying the same basis will "shock" the model because the temperature values will be appear "expanded" relative to the basis_model's "shrunk" feature space.

What's more important, however, is the fact that the actual regression is different as well; the basis_regression has worse accuracy, because the new "shape" of the feature space makes it more difficult to divide up the observations.

For the KNN regressions, we can see that the knn1_regression model predicts all of the training data with perfect accuracy because the lowest k values tend to overfit. The knn3 model provides a potentially more generalizable model with lower variance but higher bias. Interestingly enough, the knn1 model has smoother boundaries than knn3, which is a consequence of knn3's more complex "breakpoints" in which small shifts across the feature space will change the prediction because there is more likely to be a change in the set of k nearest neighbors as the feature vector changes. The increased complexity of this geometric calculation (since more neighbors are considered) is represented by an increase in the complexity of the boundary. However, this may be mistaken as a visual display of an overall increased in the complexity of the model and thus increased variance, but increasing k actually leads to more generalization and lower variance.

2. The classification probabilities are:

   (a) 0 with 100% (Probability per class: [1.00000000e+00 1.31778926e-28 2.56768493e-40])

(b) 1 with 96.45% (Probability per class: [0.03425432 0.96455651 0.00118917])

(c) 0 with 100% (Nearest Neighbor Class: [0])

(d) 0 with 100% (K Nearest Neighbor Classes: [0,0,0,0,0])

All of the models predict 0 with 100% probability except for the basis_regression, which has a different prediction with 96.45% probability. We can see how considering observations far away from most of the observations in our training data result in potentially surprising classifications. Even worse, the model provides a very "confident" prediction (as discussed in Problem 1, the model predicts with a very high probability) which can be dangerous considering that extrapolating observations far outside of the training data results in more unstable and uncertain predictions. It seems like the different classification for the basis_regression results from squaring the temperature in the basis, making it impossible for the model to use the sign of the temperatures to optimize the regression. Not only should we be aware of how models can be unstable in extrapolation, but we should also pay close attention to how this sense of "extrapolation" depends on the basis. In this case, observations with negative temperatures may have unstable or unreliable predictions despite being "close" to the training set.

End Solution

Solution 4

1.

End Solution

Name

Aidan Tai

Collaborators and Resources

None

Calibration

14