Assignment #2

Due: 11:59pm Mar 1st, 2019

Homework 2: Bayesian Methods and Multiclass Classification

Introduction

This homework is about Bayesian methods and multiclass classification. 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 first read the Bishop textbook coverage of these topic, particularly: Section 4.2 (Probabilistic Generative Models), Section 4.3 (Probabilistic Discriminative Models).

As usual, we imagine that we have the input matrix $\mathbf{X} \in \mathbb{R}^{n \times m}$ (or perhaps they have been mapped to some basis $\mathbf{\Phi}$, without loss of generality) but our outputs are now "one-hot coded". What that means is that, if there are c output classes, then rather than representing the output label y as an integer $1, 2, \ldots, c$, we represent $\mathbf{y}$ as a binary vector of length c. These vectors are zero in each component except for the one corresponding to the correct label, and that entry has a one. So, if there are 7 classes and a particular datum has label 3, then the target vector would be $C_3 = [0, 0, 1, 0, 0, 0, 0]$. If there are c classes, the set of possible outputs is $\{C_1 \ldots C_c\} = \{C_k\}_{k=1}^{c}$. Throughout the assignment we will assume that output $\mathbf{y} \in \{C_k\}_{k=1}^{c}$.

The problem set has three problems:

1. In the first problem, you will explore the properties of Bayesian estimation methods for the Bernoulli model.
2. In the second problem, you will dive into matrix algebra and the methods behind generative multiclass classifications. You will extend the discrete classifiers that we see in lecture to a Gaussian model.
3. Finally, in the third problem, you will implement logistic regression as well as a generative classifier from close to scratch.

Problem 1 (Bayesian Methods, 10 pts)

This question helps to build your understanding of the maximum-likelihood estimation (MLE) vs. maximum a posterior estimator (MAP) vs. a posterior predictive.

First consider the Beta-Bernoulli model (and see lecture 5.) Let θ be the probability that a coin comes up heads. Consider a prior $\theta \sim Beta(2, 2)$, and data $D = 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0$.

1. We are interested in seeing how θ, the probability that a coin comes up heads, changes as we get more data. Write down the expressions for:
   (a) the maximum likelihood estimate of θ
   (b) the MAP estimate of θ
   (c) the posterior predictive estimate of θ based on $P(X = 1|D)$, where X is a new flip and D is all the data you currently have.

   Notice in the case of the Beta-Bernoulli model, the posterior predictive can also be represented as a point estimate of θ, which is not always the case!

   Plot each of three different estimates after each additional sample. You can consider making a single plot with flips on the x-axis and three lines showing your guess for the probability of heads after each flip for each estimator.
2. Interpret the differences you see between the three different estimators.
3. Plot the posterior distribution (prior for 0 examples) on θ after 0, 4, 8, 12 and 16 examples. (Using whatever tools you like.) You may make separate plots for each or overlay all the plots to visualize how the posterior on θ changes.
4. Compute the marginal likelihood of the training data $p(D)$. Hint: Notice that the required integral looks like an unnormalized Beta distribution, and take advantage of the fact that integrating over a normalized Beta distribution is equal to 1.
5. Now consider an alternate model in which our prior over the coin is that it likely comes up heads or likely comes up tails, that is $\theta \sim \frac{1}{2}(Beta(10, 1) + Beta(1, 10))$. Compute the marginal likelihood for this model. Which of the two models has a higher marginal likelihood?

Solution

1.a)

$likelihood(\theta) = p(D|\theta) \propto \theta^D(1-\theta)^{N-D}$
$\rightarrow$ where D is the number of ones (heads) in the incoming data and N is the number of total observations (tosses)
$ln(p(D|\theta)) = Dln(\theta) + (N-D)ln(1-\theta)$
$\rightarrow$ take derivative w.r.t. θ
$\frac{\partial(ln(p(D|\theta)))}{\partial\theta} \propto \frac{D}{\theta} + \frac{N-D}{1-\theta}(-1)$
$\rightarrow$ set $\frac{\partial(ln(p(D|\theta)))}{\partial\theta} = 0$
$Dln(\theta) + (N-D)ln(1-\theta) = 0$
$\frac{D}{\theta} = \frac{N-D}{1-\theta}$
$D(1-\theta) = \theta(N-D)$
$D - D\theta = N\theta - D\theta$

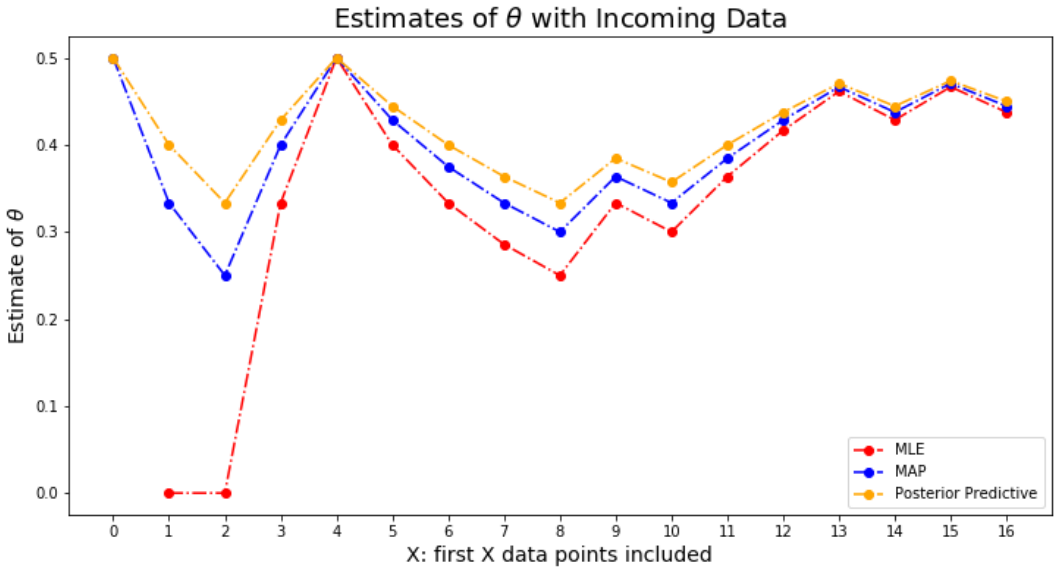
$D = N\theta$
$\theta = D/N$
$\rightarrow$ Thus, we see that $MLE_\theta = D/N$, which is the sample mean, or proportion, of ones (heads).

1.b)

$p(\theta|D) \propto p(D|\theta)p(\theta)$
$\rightarrow$ above we have that the posterior on θ is proportional to the likelihood on θ multiplied by the prior on θ, which is $\theta \sim Beta(\alpha, \beta$, where $\alpha = \beta = 2$
$p(\theta|D) \propto \theta^D(1-\theta)^{N-D} \cdot \theta^{\alpha-1}(1-\theta)^{\beta-1}$
$p(\theta|D) \propto \theta^{(D+\alpha)-1}(1-\theta)^{(N-D+\beta)-1}$
$\rightarrow$ Thus, $p(\theta|D) \sim Beta(\alpha_n, \beta_n)$ where $\alpha_n = \alpha + D$ and $\beta_n = \beta + N - D$
The MAP_θ is the mode of the distribution on the posterior of theta. For a distribution $Beta(\alpha, \beta)$ the mode is equal to $(\alpha - 1)/(\alpha + \beta - 2)$.
Thus, $MAP_\theta = \dfrac{\alpha_n - 1}{\alpha_n + \beta_n - 2}$

1.c)

$p(X|D) \displaystyle\int_0^1 p(X|\theta, D)p(\theta|D)d\theta$
$\rightarrow$ where $p(X|\theta, D) = p(X|\theta)$ is the sampling distribution of the sampling data i.e. $Bern(\theta) \sim Binom(1, \theta)$
$p(XD) = \displaystyle\int_0^1 \theta^X(1-\theta)^{1-X} \cdot \frac{1}{B(\alpha_n, \beta_n)}\theta^{\alpha_n-1}(1-\theta)^{\beta_n-1}d\theta$
$p(X|D) = \dfrac{1}{B(\alpha_n, \beta_n)} \displaystyle\int_0^1 \theta^{(X+\alpha_n)-1}(1-\theta)^{(1-X+\beta_n)-1}d\theta$
$p(X|D) = \dfrac{B(\alpha_n + X, \beta_n + 1 - X)}{B(\alpha_n, \beta_n)} \displaystyle\int_0^1 \frac{1}{B(\alpha_n + X, \beta_n + 1 - X)}\theta^{(X+\alpha_n)-1}(1-\theta)^{(1-X+\beta_n)-1}d\theta$
$\rightarrow$ here the integral sums to one as a proper beta distribution
$p(X|D) = \dfrac{B(\alpha_n + X, \beta_n + 1 - X)}{B(\alpha_n, \beta_n)}$
Thus, $p(X = 1|D) = \dfrac{B(\alpha_n + 1, \beta_n)}{B(\alpha_n, \beta_n)}$



2)

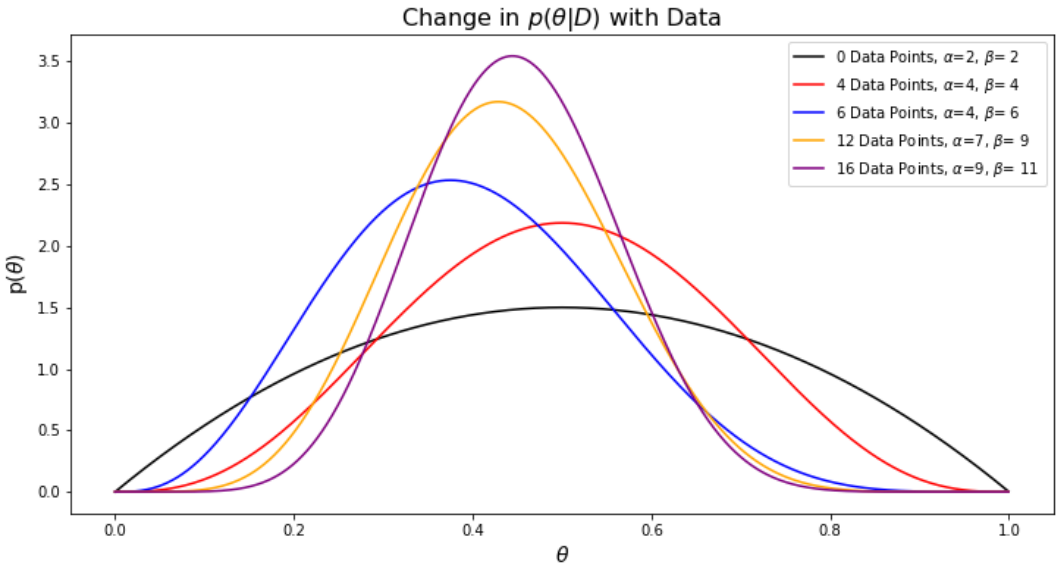
From the figure, "Estimates of θ with Incoming Data", we vary characteristic displays of the three estimators. With no data with which to draw an estimate from, the MLE estimate, unlike the MAP and Posterior Predictive (PP) estimate, can offer no prediction. The MLE estimate is based solely on observed data from which it draws a sample mean (proportion in this case). The MAP and PP estimates, on the other, are able to incorporate prior information. This is why the prior distribution is called a prior - it provides information when you may not have any data yet available. With no data, the MAP and PP estimates offer a estimate of 0.5. This makes sense as the prior on θ is a Beta(2,2) distribution which has a mode equal to its mean equal to $\alpha/(\alpha+\beta) = 1/2$.

As the data comes in one sample at a time, we see all three estimates begin to move. With initially no ones the MLE has no data available that the coin is fair at all, and with the no hits, the posterior and posterior predictive both fail as the data seemingly suggests a biased coin from the start. As further data come, however, we see all three samples move towards the sample mean. This again makes intuitive sense. For large values of α and β, the mode of the Beta distribution - and the MAP of θ - asymptotically approaches the mean of the Beta distribution, and we see that the posterior mean is a weighted average between the sample mean and the prior mean, where the weights are determined by counts (ones i.e. heads) from the data and what we can think of as pseudo counts from the prior (the α and β on the prior). We can essentially view the pseudo counts from the prior as describing a distribution on θ where the coin, on average, is far. With hyper parameters on this prior both equal to 2, the prior essentially provides us with 4 pseudo data points where we got two heads and two tails. Ultimately, as new observed data continues to come in it further and further counts outweighs the pseudo counts from the prior. And as the posterior predictive is a function of the posterior, the same story describes it as well - as seen in the plots.

3)

From the figure, "Change in $p(\theta|D)$ with Data", we see a manifestation of the the trend described in Part 2. Initially, the prior on θ tells us that θ comes from a Beta distribution centered at 0.5, but with small

magnitudes in the parameters α and β - i.e. few pseudo counts - the distribution is broad. However, as more data comes, the posterior is updated: the mean moves around slightly with the data, but after 16 observation the mean moves very close to zero and with large parameters in α and β - i.e. the counts plus pseudo counts - the distribution develops a peak, indicating a growing certainty (probability) in the value of θ being around 0.5.



4)

$$p(D) = \int_0^1 p(D|\theta)p(\theta)d\theta$$
$$p(D) = \int_0^1 \binom{N}{D}\theta^D(1-\theta)^{N-D}\frac{1}{B(\alpha,\beta)}\theta^{\alpha-1}(1-\theta)^{\beta-1}d\theta$$
$$p(D) = \int_0^1 \binom{N}{D}\frac{1}{B(\alpha,\beta)}\theta^{(D+\alpha)-1}(1-\theta)^{(N-D+\beta)-1}d\theta$$
$$p(D) = \binom{N}{D}\frac{B(\alpha+D,N-D+\beta)}{B(\alpha,\beta)}\int_0^1 \frac{1}{B(\alpha+D,N-D+\beta)}\theta^{(\alpha+D)-1}(1-\theta)^{(N-D+\beta)-1}d\theta$$
$\rightarrow$ here the integral sums to one as a proper beta distribution
$$p(D) = \binom{N}{D}\frac{B(\alpha+D,N-D+\beta)}{B(\alpha,\beta)}$$
$\rightarrow$ we observe $D = 7, N = 16$ and have $\alpha = \beta = 2$
$$p(D=7) = \binom{16}{7}\frac{B(2+7,16-7+2)}{B(2,2)} = 0.082559$$

5)

$$p(D) = \int_0^1 p(D|\theta)p(\theta)d\theta$$
$$p(D) = \int_0^1 p(D|\theta)\cdot(0.5p_1(\theta)+0.5p_2(\theta))d\theta$$

$\rightarrow$ where $p_i(\theta) \sim Beta(\alpha_i, \beta_i)$

$$p(D) = \int_0^1 0.5 p_1(D|\theta) p_1(\theta) d\theta + \int_0^1 0.5 p_2(D|\theta) p_2(\theta)) d\theta$$

$$p(D) = \sum_{i=1}^{2} 0.5 \binom{N}{D} \frac{B(\alpha_i + D, N - D + \beta_i)}{B(\alpha_i, \beta_i)}$$

$$p(D) = 0.5 \binom{N}{D} (\sum_{i=1}^{2} \frac{B(\alpha_i + D, N - D + \beta_i)}{B(\alpha_i, \beta_i)})$$

$$p(D) = 0.5 \binom{N}{D} (\frac{B(10 + D, N - D + 1)}{B(10, 1)} + \frac{B(1 + D, N - D + 10)}{B(1, 10)})$$

$$p(D = 7) = 0.5 \binom{16}{7} (\frac{B(10 + 7, 16 - 7 + 1)}{B(10, 1)} + \frac{B(1 + 7, 16 - 7 + 10)}{B(1, 10)})$$

$$p(D = 7) = 0.002154$$

From the two calculations of marginal likelihood, we see that the original model has the higher marginal likelihood.

Problem 2 (Return of matrix calculus, 10pts)

Consider now a generative c-class model. We adopt class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$ for all $k \in \{1, \ldots, c\}$ (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 $\mathbf{y}_i \in \{C_k\}_{k=1}^c$ is encoded as a one-hot target vector.

1. Write out the negated 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. Double-check your answer: the final result should be very intuitive!

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

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

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

3. Derive the gradient of the negative 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 for vector $\boldsymbol{\mu}_k$. Once again, your final answer should seem intuitive.
5. Derive the gradient for the negative 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 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

1)

$$p(D,\pi) = \prod_{n=1}^{N}\prod_{k=1}^{K} p(y_n = C_k|x_n)^{y_{n,k}} = \prod_{n=1}^{N}\prod_{k=1}^{K} (p(x_n|y_n = k)p(y_n = k))^{y_{n,k}}$$

$$\ln p(D,\pi) = \sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \cdot \ln p(y_n = C_k|x_n) = \sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \cdot \ln(p(x_n|y_n = k)p(y_n = k))$$

$$\ln p(D,\pi) = \sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \cdot (\ln p(x_n|y_n = k) + \ln p(y_n = k))$$

$$-\ln p(D,\pi) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \cdot (\ln p(x_n|y_n = k) + \ln(\pi_k))$$

$$-\ln p(D,\pi) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \ln p(x_n|y_n = k) - \sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \ln(\pi_k)$$

2)

Call $f(\pi) = -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \ln(\pi_k)$
$\max f(\pi) \quad s.t. \quad \sum_k \pi_k - 1 = 0$
$L(\pi,\lambda) = f(\pi) + \lambda(\sum_k \pi_k - 1)$

$$\frac{\partial L(\pi,\lambda)}{\partial \pi} = [-\sum_{n=1}^{N} \frac{y_{n,k}}{\pi_k} + \lambda]_{k=1}^{K} = [-\sum_{n:y_{n,k}=1}^{N} \frac{1}{\pi_k} + \lambda]_{k=1}^{K} = [-\frac{N_k}{\pi_k} + \lambda]_{k=1}^{K}$$

$\rightarrow$ where N_k is the number of observations in the kth class

$$\frac{\partial L(\pi,\lambda)}{\partial \lambda} = \sum_k \pi_k - 1$$

$\rightarrow$ set derivatives to 0

$$(1)\lambda = \frac{N_k}{\pi_k} \quad \forall k \in 1..K$$

$$(2)\sum_k \pi_k = 1$$

$\rightarrow$ substitute (1) into (2)

$$\sum_k \frac{N_k}{\lambda} = 1$$

$$\frac{1}{\lambda}\sum_k N_k = 1$$

$\rightarrow \sum_k N_k = N$

$$\frac{N}{\lambda} = 1 \quad \rightarrow \quad \lambda = N$$

$\rightarrow$ substitute back into (1)

$$N = \frac{N_k}{\pi_k} \quad \forall k \in 1..K$$

$$\pi_k = \frac{N_k}{N} \quad \forall k \in 1..K$$

3) & 4)

$$-\ln p(D,\pi) \propto -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k} \ln p(x_n|y_n = k)$$

$\rightarrow$ we have: $p(x_n|y_n = k) = (2\pi)^{-D/2}|det(\Sigma)|^{-1/2} exp(-\frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k)))$

$$\rightarrow \ln p(x_n|y_n = k) = -\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|det(\Sigma)| - \frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k)$$

$$-\ln p(D,\pi) \propto -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}(-\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|det(\Sigma)| - \frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k))$$

$$\frac{\partial(-\ln p(D,\pi))}{\partial\mu_k} = -\frac{\partial}{\partial\mu_k}(\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}(\frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k)))$$

$$\frac{\partial(-\ln p(D,\pi))}{\partial\mu_k} = -\sum_{n=1}^{N} y_{n,k}\frac{\partial}{\partial\mu_k}(\frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k))$$

$$\frac{\partial(-\ln p(D,\pi))}{\partial\mu_k} = \sum_{n=1}^{N} y_{n,k}(x_n - \mu_k)^T\Sigma^{-1}$$

$\rightarrow$ set $\frac{\partial(-\ln p(D,\pi))}{\partial\mu_k} = 0$

$$\sum_{n=1}^{N} y_{n,k}(x_n - \mu_k)^T\Sigma^{-1} = 0$$

$\rightarrow$ multiply on the right by Σ

$$\sum_{n=1}^{N} y_{n,k}(x_n - \mu_k)^T = 0$$

$$\sum_{n:y_{n,k}=1}^{N} (x_n - \mu_k) = 0$$

$$\sum_{n:y_{n,k}=1}^{N} (x_n) - N_k\mu_k = 0$$

$$\sum_{n:y_{n,k}=1}^{N} (x_n) = N_k\mu_k$$

$$\mu_k = \frac{1}{n_k}\sum_{n:y_{n,k}=1}^{N} x_n \quad \forall k \in 1..K$$

5) & 6)

$$-\ln p(D,\pi) \propto -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}(-\frac{D}{2}\ln 2\pi - \frac{1}{2}\ln|det(\Sigma)| - \frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k))$$

$$-\ln p(D,\pi) \propto -\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}(-\frac{1}{2}\ln|det(\Sigma)| - \frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k))$$

$$-\ln p(D,\pi) \propto \sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}(\frac{1}{2}\ln|det(\Sigma)|) + \sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}(\frac{1}{2}(x_n - \mu_k)^T\Sigma^{-1}(x_n - \mu_k))$$

$$\frac{\partial(-\ln p(D,\pi))}{\partial\Sigma} = N\Sigma^{-1} - \sum_{k=1}^{K}\sum_{n:y_{n,k}=1}^{N} \Sigma^{-1}(x_n-\mu_k)(x_n-\mu_k)^T\Sigma^{-1}$$

$\rightarrow$ set $\dfrac{\partial(-\ln p(D,\pi))}{\partial\Sigma} = 0$

$$N\Sigma^{-1} = \sum_{k=1}^{K}\sum_{n:y_{n,k}=1}^{N} \Sigma^{-1}(x_n-\mu_k)(x_n-\mu_k)^T\Sigma^{-1}$$

$\rightarrow$ multiply on the left and right by Σ

$$\Sigma N = \sum_{k=1}^{K}\sum_{n:y_{n,k}=1}^{N} (x_n-\mu_k)(x_n-\mu_k)^T$$

$$\Sigma = \frac{1}{N}\sum_{k=1}^{K}\sum_{n:y_{n,k}=1}^{N} (x_n-\mu_k)(x_n-\mu_k)^T$$

3. Classifying Stars [15pts]

You're tasked with classifying three different kinds of stars, based on their magnitudes and temperatures. The figure below is a plot of the data, adapted from http://astrosci.scimuze.com/stellar_data.htm and available as `hr.csv`, which you will find in the Github repository. The 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
Dwarf,-1.1,-0.16
...
```

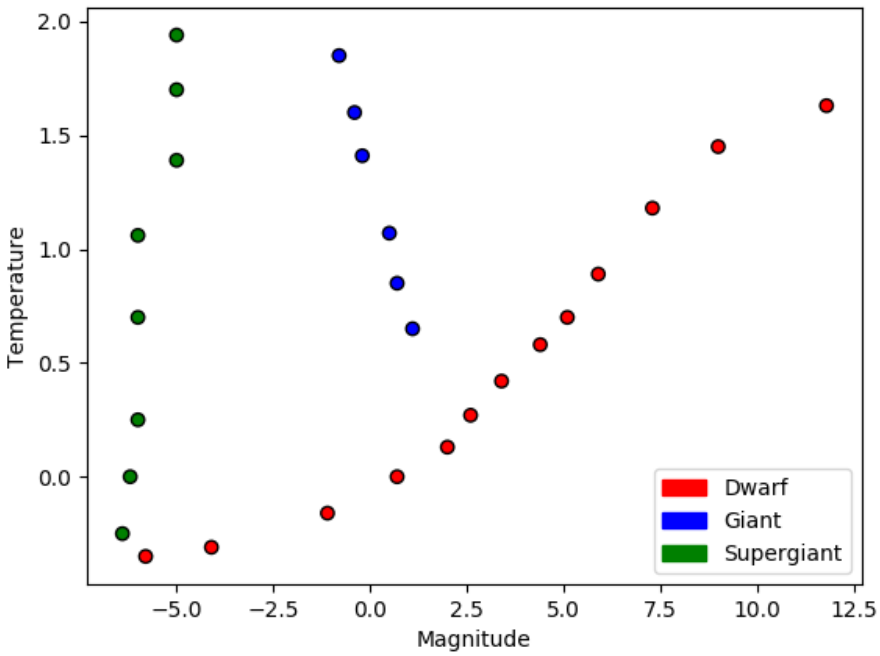


Figure 1: Magnitudes and temperatures of dwarf, giant, and supergiant stars. Adapted from http://astrosci.scimuze.com/stellar_data.htm

Problem 3 (Classifying Stars, 15pts)

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

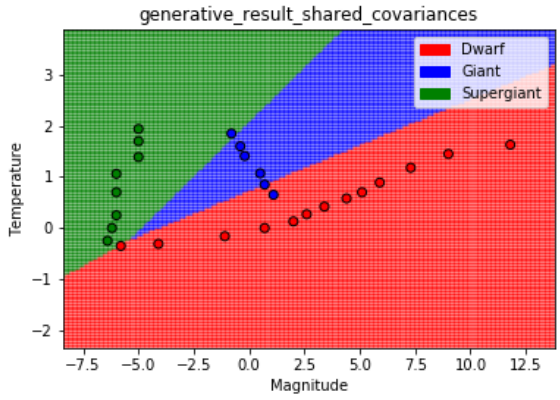
- The three-class generalization of logistic regression, also known as softmax regression, for these data. You will do this by implementing gradient descent on the negative log likelihood. You will need to find good values for the learning rate η and regularization strength λ. See the third practice problem in the section 3 notes for more information.
- A generative classifier with Gaussian class-conditional densities, as in Problem 2. In particular, make two implementations: one with a shared covariance matrix across all of the classes, and one with a separate covariance being learned for each class. (Note: the staff implementation can switch between these two with just a few lines of code.)

Implementation notes: you may use anything in `numpy` or `scipy`, except for `scipy.optimize`. The controller file is `problem3.py`, in which you will specify hyperparameters. The actual implementations you will write will be in `LogisticRegression.py` and `GaussianGenerativeModel.py`. These files include class interfaces for `GaussianGenerativeModel` and `LogisticRegression`. The classes you implement follow the same pattern as scikit-learn, so they should be familiar to you. The code currently outputs nonsense predictions just to show what the high-level interface should be, so you should completely remove the given `predict()` implementations and replace them with your implementations. You will also need to modify the hyperparameter values.

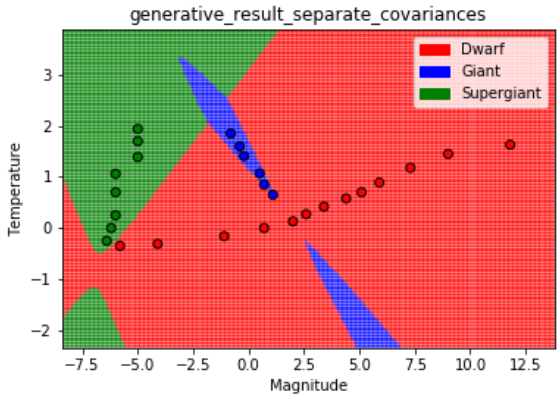
1. Plot the decision boundaries with the `visualize()` function. Include these plots in your assignment PDF. What are the similarities and differences between the classifiers? What explains the differences?
2. For logistic regression, plot negative log-likelihood loss with iterations on the x-axis and loss on the y-axis for several configurations of hyperparameters. Note which configuration yields the best final loss. Why are your final choices of learning rate (η) and regularization strength (λ) reasonable? How does altering these hyperparameters affect convergence? Focus both on the ability to converge and the rate at which it converges (a qualitative description is sufficient).
3. For both Gaussian generative models, report negative log likelihood. In the separate covariance matrix case, be sure to use the covariance matrix that matches the true class of each data point.
4. Finally, consider a star with magnitude 6 and temperature 2. To what class do each of the classifiers assign this star? Do the classifiers give any indication as to whether or not you should trust them?

Solution

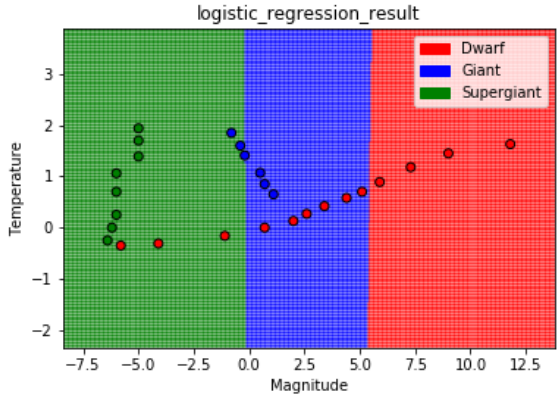
code and plots also available in attached IPython notebook and the pdf of the notebook



e.png



.png



.png

1)

From the plots of the decision boundaries, we see that the Gaussians generative model with a shared covariance and the softmax Linear Regression both have linear separable boundaries, whereas the Gaussian generative model with individual covaraince has quadratic boundaries. The gaussian generative model is linear in that the quadratic terms in the covariates drops out due to the shared covaraince matrix, whereas these quadratic terms do not drop out in the gaussian model with individual covariances matrices for the classes. The linear regression is linear separable too as its prediction is still linear in the covariates.

2)

I achieve best results with an $\eta = 0.1$ and a $\lambda = 10$. Larger values of η will allow for initial faster convergence (i.e. if you start from a very large loss), but these large values will hurt convergence towards a space with a local minima if the parameter causes a bouncing back and forth over the minima. Larger values of λ will push the weights on linear regression towards zero and help reach convergence as well - as with very large values of λ the weights will simply converge to 0. My determined values of the hyperparameters produced the lowest loss and "visually" most accurate boundaries over the other combinations of hyperparameters. With

enough iterations, all the combinations of hyperparameters converge, but seemingly doing so and getting stuck in local optima over the the space of the weights.

That being said, a comparison with the Logistic Regression model from sklearn reveals discrepancies with my model. My implementation fails to meet the same level of loss as sklearns implementation. Part of that may be due to the differing approaches - skleanr's ogistic regression uses the one versus rest approach to multiple logistic regression as opposed to my implementation of softmax logistic regression - but part may be do to an undetectable error (at least on my part) in my implementation.

3)

- implementation in code (IPython notebook and its pdf) -

Separate Covariance negative log-likelihood: 63.970359807289405

Shared Covariance negative log-likelihood: 116.3944649310099

4)

Test star type predictions for Separate Covariance Gaussian Model:
magnitude 6 and temperature 2: 0

Test star type predictions for Shared Covariance Gaussian Model:
magnitude 6 and temperature 2: 1

Test star type predictions for Linear Regression:
magnitude 6 and temperature 2: 0

The classifiers do not give an indication of whether one should trust them in the sense of an output. The individual covariance gaussian model appears to have the smallest negative log likelihood but this may be do to overfitting. Indeed, looking at the boundary plot of this latter model, one class - the giant class - is seemingly cut out from the others rather unnaturally. Perhaps the best indication of whether you should trust the classifiers is a visual inspection of the decision boundaries i.e. do they make sense. An inspection of the three models here appears to show that the gaussian model with the shared covariance matrix as the most reasonable model.

- Name: Meriton Ibrahimi
- Email: meritonibrahimi@college.harvard.edu
- Collaborators: N/A
- Approximately how long did this homework take you to complete (in hours): took me about continuous two days, but I have been a bit sick this week, so I'm not sure if it took me longer because I couldn't think clearly