Assignment #2 v1.0

Due: 11:59pm Feb 23rd, 2018

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:

- In the first problem, you will explore the properties of Bayesian estimation methods for the Bernoulli model as well as the special case of Bayesian linear regression with a simple prior.
- 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.
- 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) and posterior predictive estimator, first in the Beta-Bernoulli model and then in the linear regression setting.

First consider the Beta-Bernoulli model (and see lecture 5.)

1. Write down the expressions for the MLE, MAP and posterior predictive distributions, and for a prior $\theta \sim Beta(4, 2)$ on the parameter of the Bernoulli, and with data $D = 0, 0, 1, 1, 0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 0$, plot the three different estimates after each additional sample.
2. Plot the posterior distribution (prior for 0 examples) on θ after 0, 4, 8, 12 and 16 examples. (Using whatever tools you like.)
3. Interpret the differences you see between the three different estimators.

Second, consider the Bayesian Linear Regression model, with data $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^n$, $\mathbf{x}_i \in \mathbb{R}^m$, $y_i \in \mathbb{R}$, and generative model

$$y_i \sim \mathcal{N}(\mathbf{w}^\top \mathbf{x}_i, \beta^{-1})$$

for (known) precision β (which is just the reciprocal of the variance). Given this, the likelihood of the data is $p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I})$. Consider the special case of an isotropic (spherical) prior on weights, with

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$$

This prior makes sense when you have little prior information and do not know much about the relationship among features so you can simplify by assuming independence.

4. Using the method in lecture of taking logs, expanding and pushing terms that don't depend on $\mathbf{w}$ into a constant, and finally collecting terms and completing the square, confirm that the posterior on weights after data D is $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)$, where

$$\mathbf{S}_n = (\alpha\mathbf{I} + \beta\mathbf{X}^\top\mathbf{X})^{-1}$$
$$\mathbf{m}_n = \beta\mathbf{S}_n\mathbf{X}^\top\mathbf{y}$$

Solution:

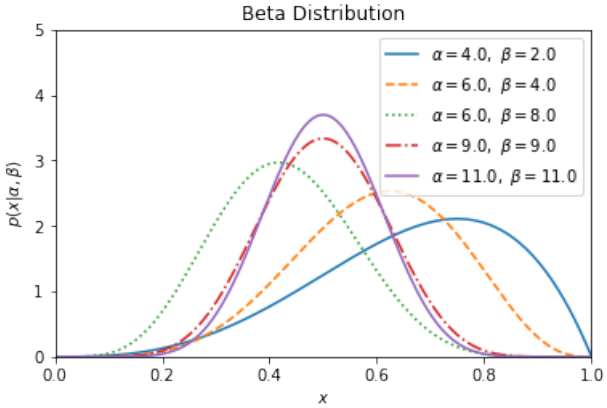
1. As derived in lecture, we showed that the addition of data points, $\theta \sim beta(\alpha, \beta)$ can be updated as $\theta \sim beta(\alpha + n_1, \beta + n_0)$. The optimal values for $\theta_{MLE}, \theta_{MAP}, \theta_{PP}$ can be calculated as follows:

$$\theta_{MLE} = \frac{n_1}{n_1 + n_0}$$

$$\theta_{MAP} = \frac{\alpha + n_1 - 1}{\alpha + n_1 + \beta + n_0 - 2}$$

$$\theta_{PP} = \frac{\alpha + n_1}{\alpha + n_1 + \beta + n_0}$$

2. The beta distribution can be updated as $\theta \sim beta(\alpha + n_1, \beta + n_0)$.



3. θ_{MLE} starts off as zero because we have no data points and the expression does not factor in the prior. θ_{MAP} and θ_{PP} exhibit very similar trend lines once plotted. This intuitively makes sense because θ_{MAP} represents mode and θ_{PP} represents mean so the values should converge after many data points. Furthermore, all three trend lines including θ_{MLE} should converge with more data as the experimental data overtakes the influence of the prior. There is no reason why θ_{MLE} is lower than θ_{MAP} or θ_{PP}, these results are just our specific to our limited dataset of 16.

4. Starting with expanding the posterior prior. In line (3), the multivariate normal and normal pdfs were applied to expand terms. Terms unrelated to $\mathbf{w}$ were dropped in line (5).

$$p(\mathbf{w}|D) = p(\mathbf{y}|\mathbf{X}, \mathbf{w})p(\mathbf{w}) \tag{1}$$

$$\ln p(\mathbf{w}|D) = \ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \ln p(\mathbf{w}) + C \tag{2}$$

$$= -\frac{\beta}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w}) - \frac{1}{2}\mathbf{w}^\top\alpha\mathbf{I}\mathbf{w} + C \tag{3}$$

$$= -\frac{\beta}{2}(\mathbf{y}^\top\mathbf{y} - \mathbf{y}^\top\mathbf{X}\mathbf{w} - \mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}) - \frac{1}{2}\mathbf{w}^\top\alpha\mathbf{I}\mathbf{w} + C \tag{4}$$

$$= -\frac{\beta}{2}(-2\mathbf{w}^\top\mathbf{X}^\top\mathbf{y} + \mathbf{w}^\top\mathbf{X}^\top\mathbf{X}\mathbf{w}) - \frac{1}{2}\mathbf{w}^\top\alpha\mathbf{I}\mathbf{w} \tag{5}$$

$$= -\frac{1}{2}(\mathbf{w}^\top(\alpha\mathbf{I} + \beta\mathbf{X}^\top\mathbf{X})\mathbf{w} - 2\mathbf{w}^\top(\beta\mathbf{X}^\top\mathbf{y})) \tag{6}$$

Starting with the multi-nomial PDF. Terms unrelated to $\mathbf{w}$ were dropped in line (3).

$$\ln p(\mathbf{w}|D) = -\frac{1}{2}(\mathbf{w} - \mathbf{m}_n)^\top \mathbf{S}_n^{-1}(\mathbf{w} - \mathbf{m}_n) + C \tag{1}$$

$$= -\frac{1}{2}(\mathbf{w}^\top \mathbf{S}_n^{-1}\mathbf{w} - \mathbf{w}^\top \mathbf{S}_n^{-1}\mathbf{m}_n - \mathbf{m}_n^\top \mathbf{S}_n^{-1}\mathbf{w} + \mathbf{m}_n^\top \mathbf{S}_n^{-1}\mathbf{m}_n) + C \tag{2}$$

$$= -\frac{1}{2}(\mathbf{w}^\top \mathbf{S}_n^{-1}\mathbf{w} - \mathbf{w}^\top \mathbf{S}_n^{-1}\mathbf{m}_n - \mathbf{m}_n^\top \mathbf{S}_n^{-1}\mathbf{w}) \tag{3}$$

$$= -\frac{1}{2}(\mathbf{w}^\top (\mathbf{S}_n^{-1})\mathbf{w} - 2\mathbf{w}^\top (\mathbf{S}_n^{-1}\mathbf{m}_n)) \tag{4}$$

By pattern making the two equations, we get the following. β can be pulled out of the $\mathbf{m}_n$ expression because it is a constant.

$$\mathbf{S}_n^{-1} = \alpha\mathbf{I} + \beta\mathbf{X}^\top\mathbf{X}$$

$$\mathbf{S}_n = (\alpha\mathbf{I} + \beta\mathbf{X}^\top\mathbf{X})^{-1}$$

$$\mathbf{S}_n^{-1}\mathbf{m}_n = \beta\mathbf{X}^\top\mathbf{y}$$

$$\mathbf{m}_n = \mathbf{S}_n\beta\mathbf{X}^\top\mathbf{y} = \beta\mathbf{S}_n\mathbf{X}^\top\mathbf{y}$$

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: $\mathbf{X}^{-\top} := (\mathbf{X}^\top)^{-1}$

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

Solutions:

1. We can write the negated log-likelihood of the dataset as the probability of each data point in a class using an indicator variable. We need the indicator variable to avoid double-counting because each data point belongs to one class. We can then multiply over all data points since they are independent.

$$
\begin{aligned}
-\ln p(D, \pi) &= -\ln(p(\mathbf{x}|\mathbf{y} = C_k)p(\mathbf{y} = C_k)) && (1)\\
&= -\ln(\Pi_{i=1}^{n}\Pi_{k=1}^{c}(p(\mathbf{x}_i|\mathbf{y}_i = C_k)\pi_k)^{y_{i,k}} && (2)\\
&= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k) && (3)
\end{aligned}
$$

2. We can utilize Lagrange Multipliers to help optimize the function. First we can address $\frac{\partial L(\pi_k, \lambda)}{\partial \pi_k} = 0$.

$$
\begin{aligned}
-\ln p(D, \pi) &= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k) + \lambda(\sum_{k=1}^{c} \pi_k - 1) && (1)\\
-\frac{\partial \ln p(D, \pi)}{\partial \pi_k} &= \frac{1}{\pi_k}\sum_{i=1}^{n} y_{i,k} + \lambda = 0 && (2)\\
\pi_k &= -\frac{\sum_{i=1}^{n} y_{i,k}}{\lambda} && (3)
\end{aligned}
$$

   Next, we can solve for the second constraint in the Lagrange Multiplier $\frac{\partial L(\pi_k, \lambda)}{\partial \lambda} = 0$. λ eventually simplifies to $-N$, where N is the total number of data points.

$$
\begin{aligned}
-\frac{\partial \ln p(D, \pi)}{\partial \lambda} &= \sum_{k=1}^{c} \pi_k - 1 = 0 && (4)\\
&= -\frac{1}{\lambda}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k} - 1 = 0 && (5)\\
\lambda &= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k} = -N && (6)
\end{aligned}
$$

   Plugging back in, we get that π_k is the ratio of the number of points in class k over the total number of data points.

$$\pi_k = \frac{\sum_{i=1}^{n} y_{i,k}}{N}$$

3. We can use the multivariate pdf to simplify the expression. Line (4) utilizes equation 86 of the Matrix

Cookbook to take the derivative of the expression since $\mathbf{\Sigma}^{-1}$ is a symmetric matrix.

$$
\begin{aligned}
-\ln p(D,\pi) &= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k) && (1)\\
&= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln \frac{1}{\sqrt{2\pi|\mathbf{\Sigma}|}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\} + \ln \pi_k) && (2)\\
&= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(-\frac{1}{2}\ln \mathbf{\Sigma} - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \ln \pi_k) && (3)\\
-\frac{\partial \ln p(D,\pi)}{\partial \pi_k} &= -\sum_{i=1}^{n} y_{i,k}(-\frac{1}{2}(-2\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)) && (4)\\
&= -\sum_{i=1}^{n} y_{i,k}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) && (5)
\end{aligned}
$$

4. We can find the MLE $\boldsymbol{\mu}_k$ by setting the expression equal to 0. The $\mathbf{\Sigma}^{-1}$ term can be multiplied out because it is a constant relative to the summation.

$$
\begin{aligned}
-\ln p(D,\pi) = -\sum_{i=1}^{n} y_{i,k}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) = 0 && (1)\\
-\sum_{i=1}^{n} y_{i,k}\mathbf{x} + \sum_{i=1}^{n} y_{i,k}\boldsymbol{\mu}_k = 0 && (2)\\
\boldsymbol{\mu}_k = \frac{\sum_{i=1}^{n} y_{i,k}x_i}{\sum_{i=1}^{n} y_{i,k}} && (3)
\end{aligned}
$$

We obtain the very intuitive result that $\boldsymbol{\mu}_k$ is the sum of all x_i in class k divided by the total number of entries in class k, or the average of x_i within class k.

5. We take the gradient of negative log likelihood but this time with respect to the the covariance matrix. Both Cookbook formulas described above are utilized.

$$
\begin{aligned}
-\ln p(D,\pi) &= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k) && (1)\\
&= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln \frac{1}{\sqrt{2\pi|\mathbf{\Sigma}|}} \exp\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\} + \ln \pi_k) && (2)\\
&= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(-\frac{1}{2}\ln \mathbf{\Sigma} - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \ln \pi_k) && (3)\\
-\frac{\partial \ln p(D,\pi)}{\partial \mathbf{\Sigma}} &= -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(-\frac{1}{2}\mathbf{\Sigma}^{-\top} - \frac{1}{2}(-\mathbf{\Sigma}^{-\top}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-\top})) && (4)\\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\mathbf{\Sigma}^{-\top} - \mathbf{\Sigma}^{-\top}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^\top \mathbf{\Sigma}^{-\top}) && (5)
\end{aligned}
$$

6. To solve for the MLE of the covariance matrix, we set the expression above to 0.

$$-\frac{\partial \ln p(D,\pi)}{\partial \boldsymbol{\Sigma}} = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\boldsymbol{\Sigma}^{-\top} - \boldsymbol{\Sigma}^{-\top}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^{\top}\boldsymbol{\Sigma}^{-\top}) = 0 \tag{1}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}\boldsymbol{\Sigma}^{-\top} = \sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}\boldsymbol{\Sigma}^{-\top}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^{\top}\boldsymbol{\Sigma}^{-\top}) \tag{2}$$

$$\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k} = \sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^{\top}\boldsymbol{\Sigma}^{-\top} \tag{3}$$

$$\boldsymbol{\Sigma}^{-\top} = \frac{\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}}{\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^{\top}} \tag{4}$$

$$\boldsymbol{\Sigma} = \left(\frac{\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}}{\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\mathbf{x}-\boldsymbol{\mu}_k)(\mathbf{x}-\boldsymbol{\mu}_k)^{\top}}\right)^{-\top} \tag{5}$$

3. Classifying Fruit [15pts]

You're tasked with classifying three different kinds of fruit, based on their heights and widths. Figure 1 is a plot of the data. Iain Murray collected these data and you can read more about this on his website at http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/. We have made a slightly simplified (collapsing the subcategories together) version of this available as `fruit.csv`, which you will find in the Github repository. The file has three columns: type (1=apple, 2=orange, 3=lemon), width, and height. The first few lines look like this:

```
fruit,width,height
1,8.4,7.3
1,8,6.8
1,7.4,7.2
1,7.1,7.8
...
```

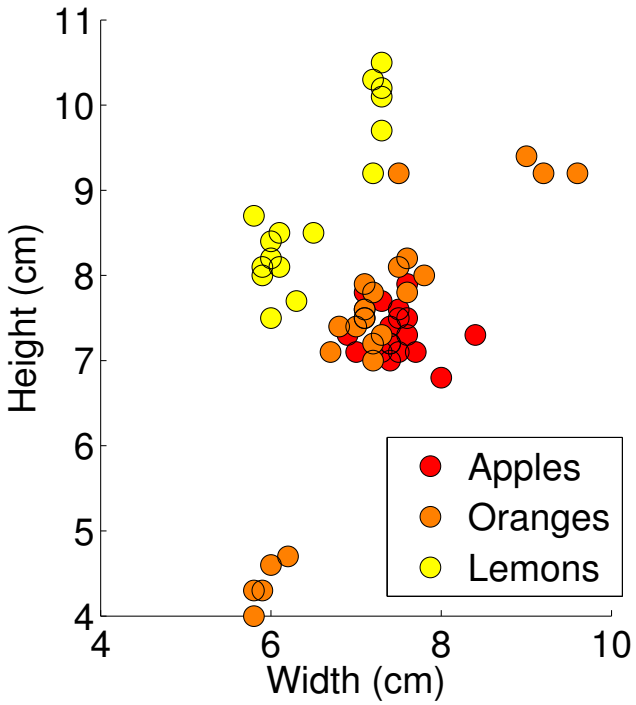


Figure 1: Heights and widths of apples, oranges, and lemons. These fruit were purchased and measured by Iain Murray: http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

Problem 3 (Classifying Fruit, 15pts)

You should implement the following:

- 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 information about multi-class logistic regression, softmax, and negative log likelihood.
- A generative classifier with Gaussian class-conditional densities, as in Problem 3. In particular, make two implementations of this, one with a shared covariance matrix across all of the classes, and one with a separate covariance being learned for each class. Note that the staff implementation can switch between these two by the addition of just a few lines of code. In the separate covariance matrix case, the MLE for the covariance matrix of each class is simply the covariance of the data points assigned to that class, without combining them as in the shared case.

You may use anything in `numpy` or `scipy`, except for `scipy.optimize`. That being said, if you happen to find a function in `numpy` or `scipy` that seems like it is doing too much for you, run it by a staff member on Piazza. In general, linear algebra and random variable functions are fine. 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`.

You will be given class interfaces for `GaussianGenerativeModel` and `LogisticRegression` in the distribution code, and the code will indicate certain lines that you should not change in your final submission. Naturally, don't change these. These classes will allow the final submissions to have consistency. There will also be a few hyperparameters that are set to irrelevant values at the moment. You may need to modify these to get your methods to work. The classes you implement follow the same pattern as scikit-learn, so they should be familiar to you. The distribution 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.

- The `visualize()` method for each classifier will save a plot that will show the decision boundaries. You should include these in this assignment.
- Which classifiers model the distributions well?
- What explains the differences?

In addition to comparing the decision boundaries of the three models visually:

- 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).
- 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.

Finally, consider a fruit with width 4cm and height 11cm. To what class do each of the classifiers assign this fruit? What do these results tell you about the classifiers' ability to generalize to new data? Repeat for a fruit of width 8.5cm and height 7cm.

Solution:

For logistic regression, I tuned the parameters such that I ended up selecting $\eta = 0.00075$ and $\lambda = 0.001$. As η increased, we can converge at a faster rate but if η is too large, we may converge to a different local optimum such as in the top right chart where loss was increasing. With a smaller η, we have more precision in our learning rate but at the expense of greater computational power. In the top right chart, we see that for small η, the rate of convergence is extremely slow. For λ, as λ increases, this regularization helps us converge at a faster rate. This is evident as the bottom left graph converges faster than the bottom right graph. However, for large λ, as larger weights are penalized, we can begin to lose accuracy through underfitting our model such as in the bottom right chart. I proceeded with the tuned parameters of the bottom right chart of $\eta = 0.00075$ and $\lambda = 0.001$.

Negated log-likelihood was calculated with L2 norms based on section 2 notes question 3, properties of softmax. The gradient was similarly adapted from section notes with an added regularization term.

$$\mathcal{L}(w_\ell) = -\sum_{i=1}^{N} \ln p(\mathbf{y}_i|\mathbf{x}_i; \{w_\ell\}) + \lambda \mathbf{w}^2$$

$$\frac{\partial}{\partial w_j}\mathcal{L}(w_\ell) = \sum_{i=1}^{N} [p(\mathbf{y}_i|\mathbf{x}_i; \{w_\ell\}) - y_{ik}]x_i + 2\lambda \mathbf{w}$$

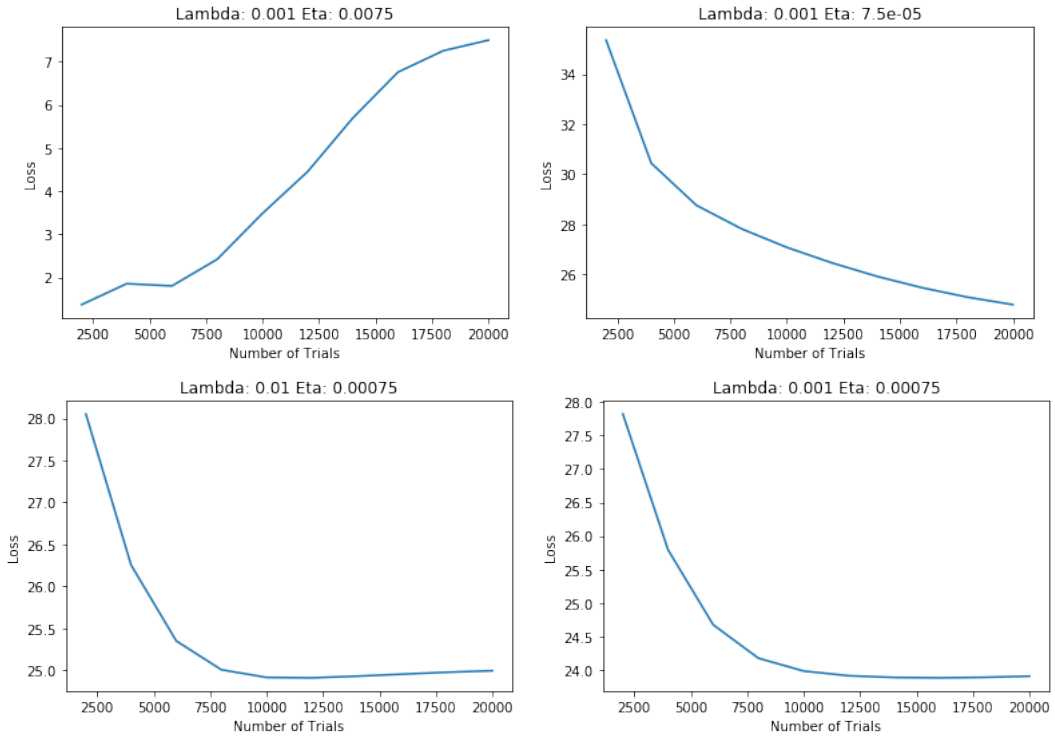


Figure 2: Hyperparameter tuning of λ and η lead to optimal performance with $\lambda = 0.001$ and $\eta = 0.00075$ shown in bottom right graph

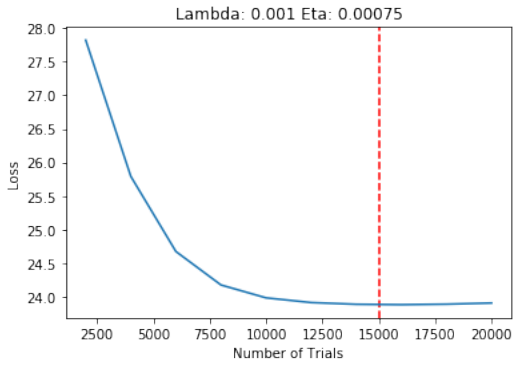


Figure 3: Number of trials run determined by when the gradient approached zero, signaling that the model has stopped improving

Logistic regression was run for 15,000 trials as determined by when the gradient of the norm approached < 0.05 with the tuned parameters of $\eta = 0.00075$ and $\lambda = 0.001$. The overall negated log-likelihood for logistic regression converged to 23.91. For Guassian distributions, I coded negative log-likelihood as calculated in question 2a and π_k as calculated in 2b.

$$-\ln p(D, \pi) = -\sum_{i=1}^{n}\sum_{k=1}^{c} y_{i,k}(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k)$$

$$\pi_k = -\frac{\sum_{i=1}^{n} y_{i,k}}{\lambda}$$

This yielded a negated log-likelihood of 147 for Guassian with separate covariances and 179 for Guassian with shared covariance.

We obtain the following visualizations for Guassian with individual covariances, Guassian with shared covariance, and logistic regression. Upon examination, guassian with separate covariances seems to be more accurate than guassian with shared covariances. We can examine this qualitatively with more accurate predictions in the graph and see this quantiatively through the lower negated log-likelihood, signaling a stronger fit. The logistic regression overall performs well but seems to be a weaker fit than the Guassian with shared covariance based on a higher number of misclassified apples (red entries). This may be the result of a poorer fit to the training data in the calculation of the weights.

The difference can be attributed to some of the analytical differences between generative and discriminative models. Logistic regression is a discriminative model, based on conditional probability $p(y|x)$, while Guassian is a generative model, based on joint probability $p(x, y)$. This means that the logistic regression learns directly how to map classifications from the features and focuses on learning the boundaries separating classes. In contrast, the Guassian takes a more holistic approach to the dataset and cares about the distribution of the datapoints in each class. Furthermore, the accuracy of the Guassian models hinges on the assumption that the data points follow a normal distribution, which may not always be true.

A fruit of width 4cm and height 11cm is classified as class 1 for Guassian with separate covariances and class 2 for Guassian with shared covariance and logistic regression. For a fruit of width 8.5cm and height 7cm, all three models classified this fruit as class 0. In general, discriminative models tend to be better at handling new data because the models are based on feature data. We observed that the discriminative and the Guassian model with shared covariance both yielded the same predictions. I hypothesize that Guassian model with separate covariance may have overfit to the difference classes and is less robust to outliers. This generates counterintutive predictions such as predicting both high height and low width as well as high width and low height to be oranges.

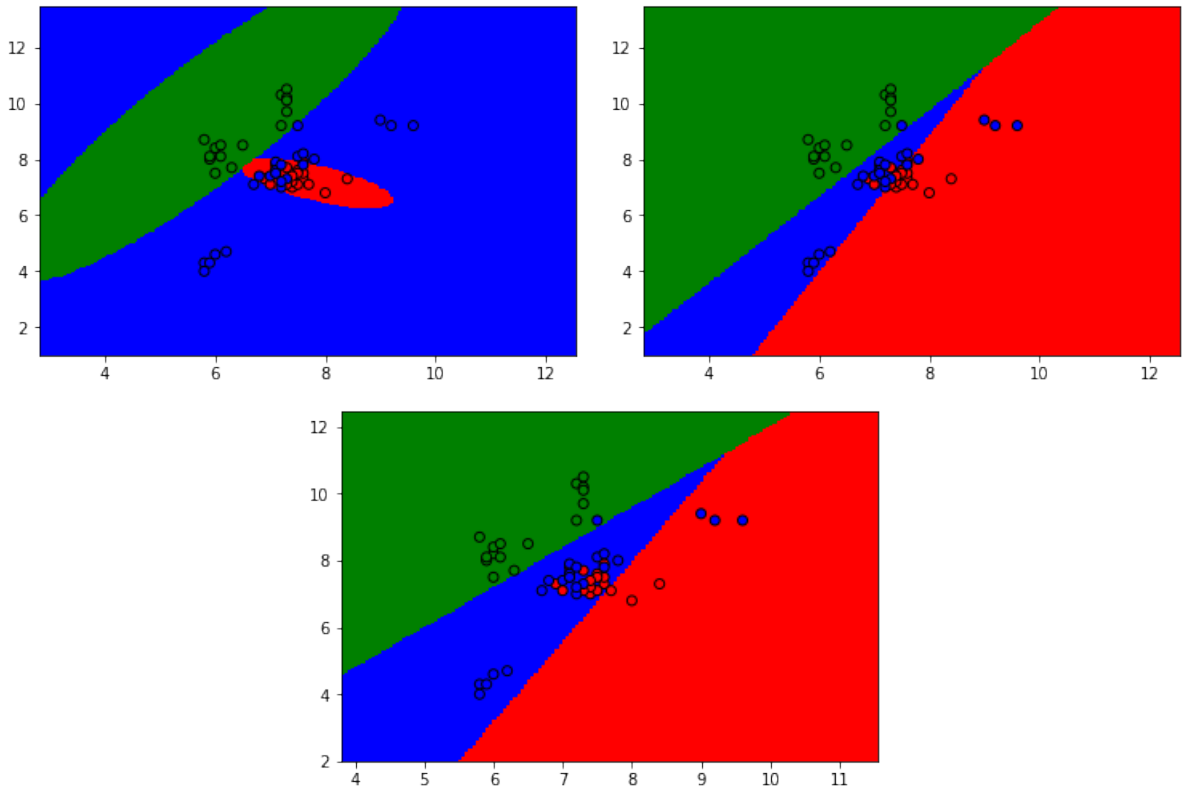


Figure 4: Guassian with individual covariances (Top Left), Guassian with shared covariance (Top Right), and logistic regression (Bottom)

Calibration [1pt]

Approximately how long did this homework take you to complete? 20 hours

Name, Email, and Collaborators

Name: Christine Zhang

Email: christinezhang@college.harvard.edu

Collaborators: David Yang, Emily Chen, Shira Li