Assignment #2 v1.1

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 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

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

1a. Write down the expressions for the MLE, MAP and posterior predictive distributions

Define num_0 and num_1 as the number of 0s of and 1s, respectively, that we have observed so far.

$$MLE = \frac{num_0}{num_0 + num_1}$$

$$MAP = \frac{\alpha + num_1 - 1}{\alpha + \beta + num_0 + num_1 - 2}$$

$$Postpred = \frac{\alpha + num_1}{\alpha + \beta + num_1 + num_0}$$

1b. 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.

Plot of estimate for the θ_{MLE}, θ_{MAP}, and posterior predictive, after each of 16 observations.

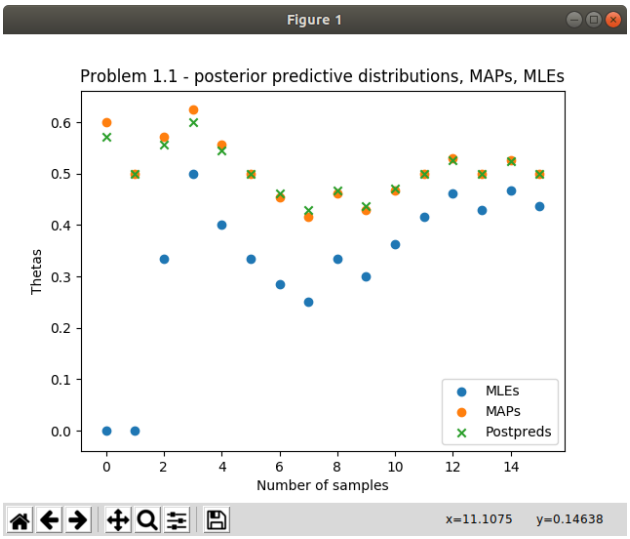


Figure 1

2. Plot the posterior distribution (prior for 0 examples) on θ after 0, 4, 8, 12 and 16 examples. (Using whatever tools you like.)

Posterior distribution on theta (with initial zero prior) on θ, after obtaining 0,4,8,12, and 16 samples

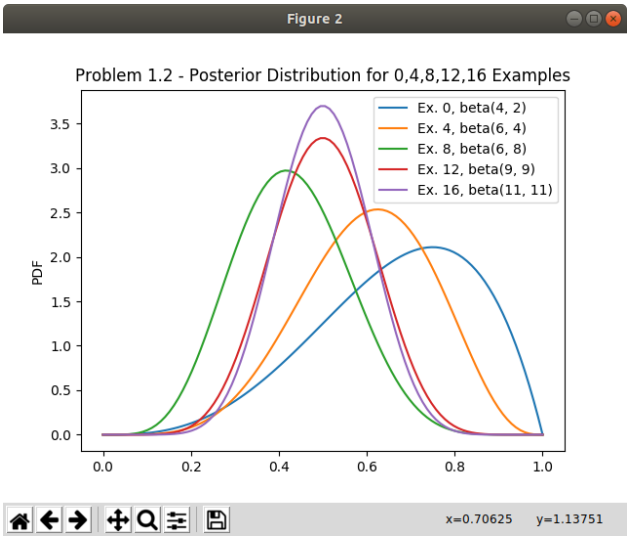


Figure 2: Problem 1, part 2.

3. Interpret the differences you see between the three different estimators.

   We have defined the posterior distribution as equal to $Bernoulli(\theta) \cdot Beta(\alpha, \beta)$. The Bernoulli distribution is the generative model, or *likelihood* of our data occurring (given this distribution) and the Beta distribution is our *prior*, where we weigh the new observations against our prior belief about the system.

   We started with an initial prior (in blue on Figure 2, which lends us to believe that the Bernoulli process we're observing is skewed toward producing ones, that is the θ parameter characterizing the distribution is fairly large.

   Note: In a Bernoulli distribution θ is the likelihood of seeing a one and $1 - \theta$ is the likelihood of seeing a zero.

   Our maximum likelihood estimates do not incorporate a prior belief. Thus $\theta_M LE$ exactly matches the initial two data points, zero. In contrast, our maximum posteriors (MAPs), incorporate prior belief. Thus, although we see two zeros, our MAP remains high. The posterior prediction incorporates our uncertainty about w into our prediction. It tends to estimate a value between the MAP and MLE, converging to the MAP prediction as the prior on the MAP updates. Over time, the MLE also converges to the MAP since the counts on average will equal the mean of the generative distribution.

   We can actually see this on the Figure 2. Initially our mean on the prior is skewed to the right, but as we get more samples, we update the prior so that it shifts to average around 0.5. Additionally, as we get observation data, we become more confident about the shape of our generative distribution, and thus the variance on the beta distribution goes down.

4. Given likelihood

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \beta^{-1}\mathbf{I})$$

   and prior

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

Confirm that the posterior on weights after data D is

$$\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)$$

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

Do so by [1] taking logs [2] expanding and pushing terms that don't depend on $\mathbf{w}$ into a constant [3] finally collecting terms and completing the square

At a high level, the posterior equals the likelihood times the prior.

After taking the natural log of the posterior, we get

$$\ln p(\mathbf{w}|D) \propto \ln p(\mathbf{y}|\mathbf{X}, \mathbf{w}) + \ln p(\mathbf{w})$$

We expand using the definition of the normal distribution, as applied to the likelihood and prior.

Note that

$$\mathcal{N}(\boldsymbol{x} \mid \boldsymbol{\mu}, \Sigma) = \frac{1}{\sqrt{(2\pi)^D|\boldsymbol{\Sigma}|}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{1}$$

And that the natural log is equal to

$$-\frac{1}{2}\left(D\ln 2\pi + \ln|\boldsymbol{\Sigma}| + (\mathbf{x}-\boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{2}$$

Plugging in and pushing terms that don't depend on w into a constant, we get

$$\begin{aligned}\ln p(\mathbf{w}|D) \propto & \\ &= -\frac{1}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T\beta(\mathbf{y}-\mathbf{X}\mathbf{w}) - \frac{1}{2}(\mathbf{w}-0)^T(\alpha^{-1}I)^{-1}(\mathbf{w}-0) \\ &= -\frac{\beta}{2}(\mathbf{y}-\mathbf{X}\mathbf{w})^T(\mathbf{y}-\mathbf{X}\mathbf{w}) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\end{aligned}$$

FOILing, and dropping the first term which doesn't depend on w ; and on the second line noting that $\mathbf{y}^T\mathbf{X}\mathbf{w} = \mathbf{w}^T\mathbf{X}^T\mathbf{y}$ as well as carrying out the transpose, we get

$$= -\frac{\beta}{2}\left(-\mathbf{y}^T\mathbf{X}\mathbf{w} - (\mathbf{X}\mathbf{w})^T\mathbf{y} + (\mathbf{X}\mathbf{w})^T\mathbf{X}\mathbf{w}\right) - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} \tag{3}$$

$$= -\frac{1}{2}\left(-2\beta(\mathbf{w}^T\mathbf{X}^T\mathbf{y}) + \beta(\mathbf{w}^T\mathbf{X}^T\mathbf{X}\mathbf{w}) - \alpha\mathbf{w}^T\mathbf{w}\right) \tag{4}$$

$$= -\frac{1}{2}\left(-2\mathbf{w}^T(\beta\mathbf{X}^T\mathbf{y}) - \mathbf{w}^T(\beta\mathbf{X}^T\mathbf{X} - \alpha)\mathbf{w}\right) \tag{5}$$

Now working backwards, we note that we would like to show that $\mathbf{w} \sim \mathcal{N}(\mathbf{w}|\mathbf{m}_n, \mathbf{S}_n)$. Doing as before (taking the log and expanding), we get that our target equation is... (on third line, dropping term without w, and combining two middle terms)

$$= -\frac{1}{2}\left((\mathbf{w}-\mathbf{m}_n)^T\mathbf{S}_n^{-1}(\mathbf{w}-\mathbf{m}_n)\right) \tag{6}$$

$$= -\frac{1}{2}(\mathbf{w}^T\mathbf{S}_n^{-1}\mathbf{w} - \mathbf{m}_n^T\mathbf{S}_n^{-1}\mathbf{w} - \mathbf{w}^T\mathbf{S}_n^{-1}\mathbf{m}_n + \mathbf{m}_n^T\mathbf{S}_n^{-1}\mathbf{m}_n) \tag{7}$$

$$= -\frac{1}{2}(\mathbf{w}^T\mathbf{S}_n^{-1}\mathbf{w} - 2\mathbf{w}^T\mathbf{S}_n^{-1}\mathbf{m}_n) \tag{8}$$

$$\tag{9}$$

Looking above at Equation 5 we see that it is of similar form to Equation (9) above, if we define

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

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

For the last equation, Shifting S_n to the other side we get

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

as we were asked to prove.

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

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

1. Write out the negated log-likelihood of the data set

The likelihood of the dataset occuring is the product of the probabilities of each datapoint occuring.

$$P(D; \boldsymbol{\pi}_k) = \prod_{i=1}^{n} P(x_i, y_i)$$

By definition, the joint probability

$$P(\mathbf{x}, \mathbf{y}) = p(\mathbf{y})p(\mathbf{x}|\mathbf{y})$$

We are given that:
the class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$
the class conditional $p(\mathbf{x}|\mathbf{y} = C_k)$

Furthermore, we know that there are c **classes. Thus we can write the likelihood**

$$p(D; \boldsymbol{\pi}) = \prod_{i=1}^{n} \prod_{k=1}^{c} \pi_k p(x_i|C_k)$$

To find the negative log likelihood (lnll), we take (as stated) the log of both sides

(Note that we use a mathematical notation trick to ensure that we can cleanly write the negative lnll. We use $\mathbb{I}\{y_i = k\}$ **to indicate that for the likelihood, we only want to evaluate the probability** $p(\mathbf{x}, y)$ **for x's true class, rather than all classes. To clean things up even further, we'll shorten it to** $\mathbb{I}_{ik}$ **which evaluates to 1 for the** ith **datapoint's true class k, and zero otherwise.)**

$$-\ln p(D; \boldsymbol{\pi}) = \sum_{i=1}^{n} \sum_{k=1}^{c} \mathbb{I}_{ik}[\ln(\pi_k) - \ln p(x_i|C_k)] \tag{10}$$

$$= -\sum_{i=1}^{n} \sum_{k=1}^{c} \mathbb{I}_{ik}[\ln(\pi_k) - \ln p(x_i|y_i = C_k)] \tag{11}$$

2.1 Using the hint on Lagrange multipliers below, give the expression for the maximum-likelihood estimator for the prior class-membership probabilities, i.e. $\hat{\pi}_k$.

To find the MLE for $\hat{\pi}_k$**, we take the derivative of the negative log likelihood with respect to** π_k**.**

$$\arg\max_{\boldsymbol{\pi}_k} -\ln L(D) = \arg\max_{\boldsymbol{\pi}_k} -\sum_{i=1}^{n} \sum_{k=1}^{c} \mathbb{I}_{ik}[\ln(\pi_k) - \ln p(x_i|y_i = C_k)] \tag{12}$$

$$= \arg\max_{\boldsymbol{\pi}_k} -\sum_{i=1}^{n} \sum_{k=1}^{c} \mathbb{I}_{ik} \ln(\pi_k) \tag{13}$$

(Note that the class conditional $p(x_i|C_k)$ **does *not* depend on** π_k**, which is a parameter of a completely separate distribution (the class prior), and so we have dropped the class conditional term.)**

To find the MLE for $\hat{\pi}_k$**, aka the** $\arg\max \boldsymbol{\pi}_k$ **of the likelihood function, we normally set the derivative (with respect to** $\boldsymbol{\pi}_k$**) equal to zero and solve for** $\boldsymbol{\pi}_k$**. However, note that as**

we have multiple $\boldsymbol{\pi}$ parameters (one for each class), we must include the constraint that $\sum_{k=1}^{c} \pi_k - 1 = 0$.

We will use the method of Lagrange multipliers to handle this extra constraint when solving for $\hat{\pi}_k$.

Substituting equation (13), and our constraint, into the following equation (where $g(\mathbf{x})$ is the constraint function)

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

we get

$$Lag(\boldsymbol{\pi}, \lambda) = -\sum_{i=1}^{n}\sum_{k=1}^{c} \mathbb{I}_{ik} \ln(\pi_k) + \lambda(\sum_{k=1}^{c} \pi_k - 1)$$

To find the optimum using the Lagrange multiplier method, we then solve the system of two equations

$$\frac{\partial}{\partial \boldsymbol{\pi}} Lag(\boldsymbol{\pi}_k, \lambda) = 0 \text{ and } \frac{\partial}{\partial \lambda} Lag(\boldsymbol{\pi}_k, \lambda) = 0.$$

Note that we can "remove" the summation over k because we are taking the derivative with respect to all individual pi_k's, as a vector, and the indicator allows us to only care about the kth class.

$$\frac{\partial}{\partial \pi_k} Lag(\boldsymbol{\pi}, \lambda) = -\sum_{i=1}^{n} \frac{1}{\pi_k} - \lambda = 0$$

The indicator $\mathbb{I}_{ik}$ says that we only care about the points in the k class, which we will denote as N_k. We may thus pull the π_k out of the summation, and get $N_k \pi_k$. Thus we solve and get

$$\pi_k = -\frac{N_k}{\lambda}$$

From the second derivative, we get

$$\frac{\partial}{\partial \lambda} Lag(\boldsymbol{\pi}, \lambda) = \sum_{k=1}^{c} \pi_k - 1 = 0$$

Combining the two equations, we may write that

$$\sum_{k=1}^{c} -\frac{\lambda}{N_k} = 1$$

Note that λ is a constant, while the sum across all classes of the number of points in each class is n. Simplifying we get that

$$-\lambda \sum_{k=1}^{c} \frac{1}{N_k} = 1$$
$$-\lambda \frac{1}{n} = 1$$
$$\lambda = -n$$

Thus we get that for each class, the prior on y_k, or our belief about the distribution of y_k, is simply the number of datapoints in that k class (N_k) over the total number of datapoints(n).

$$\pi_k = \frac{n}{N_k}$$

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!

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.

We will use the expression for negative log-likelihood (lnLL) in (11). We plug in the class-conditional distribution we were given

$$p(\mathbf{x}|\mathbf{y} = C_k) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$$

into the equation for the natural log of the normal distribution (2) to get our L_g equation

$$\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[-\frac{1}{2}(\ln|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)) + \ln\pi_k\right] \tag{14}$$

Dropping terms without μ_k we get

$$\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right] + const \tag{15}$$

Using the chain rule, we can say that

$$\frac{\partial L_g}{\partial \boldsymbol{\mu}_k} = \frac{\partial L_g}{\partial(\mathbf{x}_n - \boldsymbol{\mu}_k)}\frac{\partial(\mathbf{x}_n - \boldsymbol{\mu}_k)}{\partial \boldsymbol{\mu}_k} = -\frac{\partial L_g}{\partial(\mathbf{x}_n - \boldsymbol{\mu}_k)}$$

Using the formula

$$\frac{\partial \mathbf{a}^T\mathbf{X}\mathbf{a}}{\partial \mathbf{X}} = \frac{\partial \mathbf{a}^T\mathbf{X}^T\mathbf{a}}{\partial \mathbf{X}} = \mathbf{a}^T\mathbf{a}$$

we get

$$\frac{\partial L_g}{\partial \boldsymbol{\mu}_k} = \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-T})(\mathbf{x}_n - \boldsymbol{\mu}_k)\right]$$

For cleanliness the covariances can be pulled out of the summations so that we get

$$\frac{\partial L_g}{\partial \boldsymbol{\mu}_k} = \frac{1}{2}(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-T})\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}(\mathbf{x}_n - \boldsymbol{\mu}_k) \tag{16}$$

4. Derive the maximum-likelihood estimator for vector $\boldsymbol{\mu}_k$. Once again, your final answer should seem intuitive.

To get the MLE for μ_k We set the gradient in (16) equal to zero and solve for μ_k.

$$0 = \frac{1}{2}(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-T})\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)$$

$$0 = \sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)$$

Thinking carefully, we see that the sum of μ_k across all the points in that class, is equal to $N_k\mu_k$. Thus we can pull the μ_k out of the indicator and summation to get $N_k \cdot \mu_k$. Furthermore we see that since we are solving with respect to the kth class, we can indicate with a k subscript that that is the x we care about and remove the indicator for clarity.

Thus we get that

$$0 = N_k\mu_k - \sum_{i=1}^{n}\mathbf{x}_n$$

And therefore we end up with

$$\boldsymbol{\mu}_k = \frac{1}{N_k}\sum_{i=1}^{n}\mathbf{x}_n \tag{17}$$

In other words, that the mean of the distribution is the sum of the value of all the datapoints in the class, divided by the number of datapoints in the class.

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!

As before we start with our L_g as derived in (14).

$$\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[-\frac{1}{2}(\ln|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)) + \ln\pi_k\right] \tag{18}$$

Dropping terms without $\boldsymbol{\Sigma}$ we get

$$\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[-\frac{1}{2}(\ln|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k))\right] \tag{19}$$

Taking the partial derivative, we get

$$\frac{\partial L_g}{\partial\boldsymbol{\Sigma}} = \frac{\partial L_g}{\partial\boldsymbol{\Sigma}}\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[-\frac{1}{2}(\ln|\boldsymbol{\Sigma}| + (\mathbf{x}_i - \boldsymbol{\mu}_k)^T\Sigma^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k))\right] \tag{20}$$

Using the equations as given to us

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

We get

$$\frac{\partial L_g}{\partial\boldsymbol{\Sigma}} = \sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\left[-\frac{1}{2}(\boldsymbol{\Sigma}^{-T} + \boldsymbol{\Sigma}^{-T}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T(\mathbf{x}_i - \boldsymbol{\mu}_k)\boldsymbol{\Sigma}^{-T})\right] \tag{21}$$

6. Derive the maximum likelihood estimator of the covariance matrix.

Setting (21) equal to 0 and solving we get

$$\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\boldsymbol{\Sigma}^{-T} = \sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}\boldsymbol{\Sigma}^{-T}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T(\mathbf{x}_i - \boldsymbol{\mu}_k)\boldsymbol{\Sigma}^{-T} \tag{22}$$

Left and right multiplying (on both sides of the equation) by $\boldsymbol{\Sigma}^T$ we get

$$N\boldsymbol{\Sigma}^T = \sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T(\mathbf{x}_i - \boldsymbol{\mu}_k)$$

Taking the transpose of both sides, finally we get that

$$\boldsymbol{\Sigma} = \frac{1}{N}\sum_{i=1}^{n}\sum_{k=1}^{c}\mathbb{I}_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T(\mathbf{x}_i - \boldsymbol{\mu}_k)$$

Variance is a measure of how "spread out" a dataset is, relative to its mean. Thus it makes sense that the inside term looks like a squared error term, normalized by the total number of datapoints. Covariance measures the spread of multiple datasets and how they relate to each other. If increasing in one dimension often corresponds to an increase in another dimension, the two dimensions are related through a positive covariance matrix.

3. Classifying Fruit [15pts]

You're tasked with classifying three different kinds of fruit, based on their heights and widths. Figure 3 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
...
```

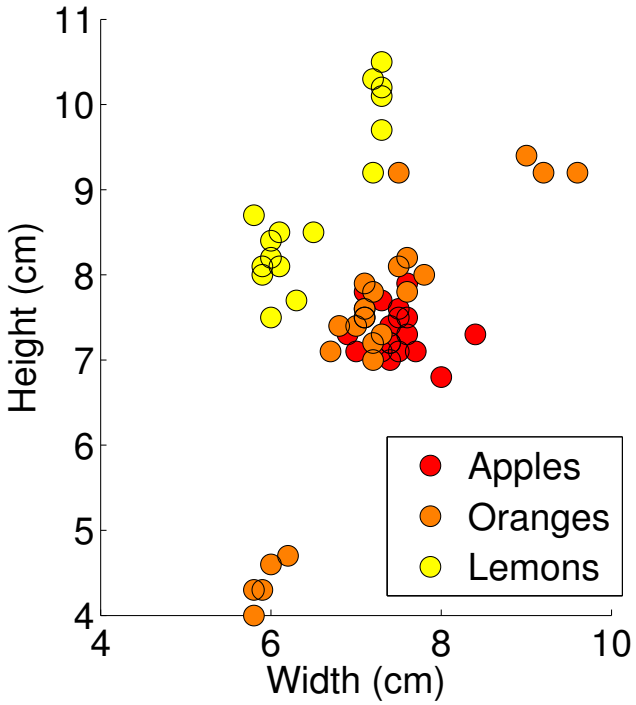


Figure 3: Heights and widths of apples, oranges, and lemons. These fruit were purchased and measured by Iain Murray: http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

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.

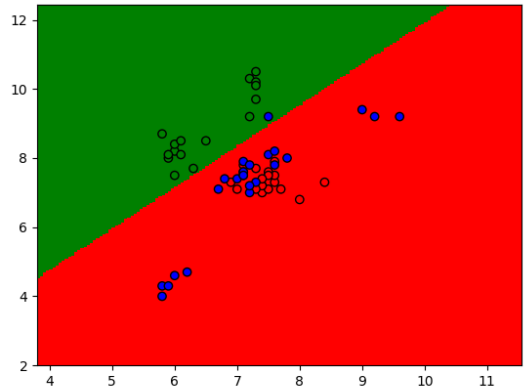
- The `visualize()` method for each classifier will save a plot that will show the decision boundaries. You should include these in this assignment.
- Which classifiers model the distributions well?
- What explains the differences?

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

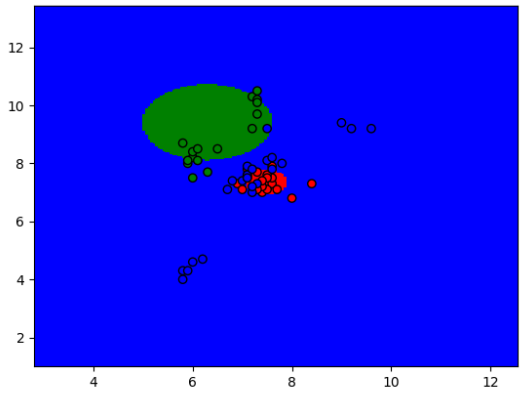
- For logistic regression, plot negative log-likelihood loss with iterations on the x-axis and loss on the y-axis for several configurations of hyperparameters. Note which configuration yields the best final loss. Why are your final choices of learning rate (η) and regularization strength (λ) reasonable? 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.

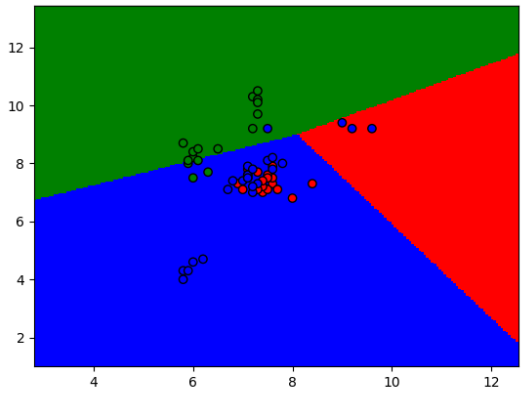
1. Decision boundaries for Logistic, Gaussian with and without shared covariance



(a)



(b)



(c)

Figure 4: Log, gen, gen with covar

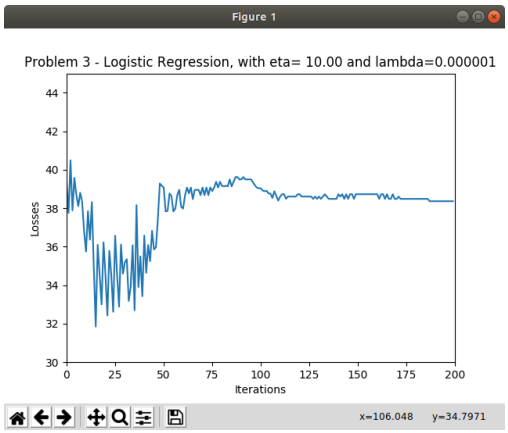
The logistic regression does a decent job of seperating the greens out, aka the lemons. However, it falters where we have the blue and red (aka oranes and apples) essentially intermised on top of each other. It is unable to distinguish between them and classifies them all as apples.

The generative model with shared covariances seems to struggle as well. There may be an error in my code, as it appears to have managed to incorrectly classify all reds (apples), calling them all blues (oranges), bug again, with the shared covariance Gaussian, we are only able to have linear decision boundaries and so the model has trouble distinguishing between oranges and apples. If we refer to the section 4 notes, we see that maximizing the posterior results in a linear term instead of a quadratic as in the separate covariance case.

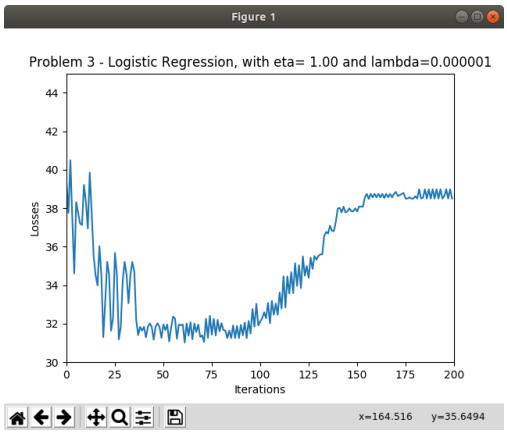
Most notably, the Generative model using multiple Gaussian distributions with separate covariances, is able to achieve a non-linear fit. This is because with a different covariance

per class, the log likelihood contains a quadratic term and we are able to a decision boundary as pictures in Fig 4 (b).

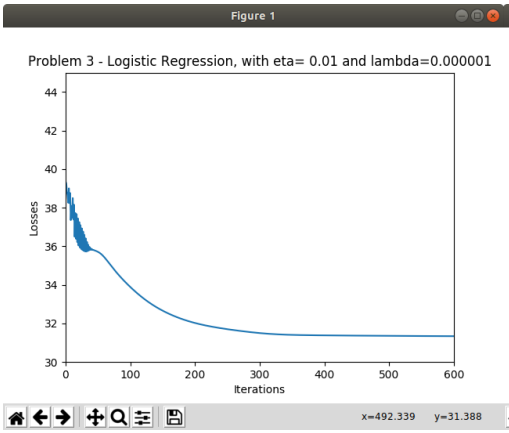
2. Logistic regression hyperparameters



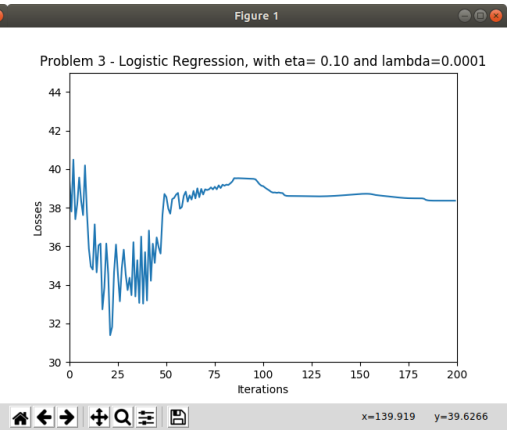
(a)



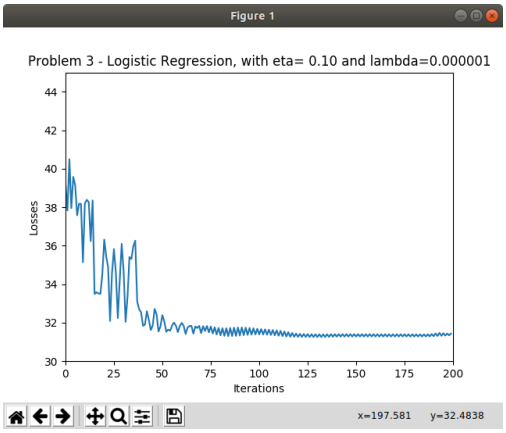
(b)



(c)



(d)



(e)

Figure 5: Problem 3: Hyperparameters

figure (e) Briefly, having η equal to 0.1 and λ kept extremely small (1 e-6) allowed for fast convergence and stable convergence.

figure (d) With λ values any greater than 1 e-6, the logistic regression model fails to converge. I hypothesize that, this is because the apples and oranges are so intermixed, that we must have relatively large weights and can't penalize the model for complexity too much.

figure (c) If η is too small, say 0.01, we see that it takes much longer to converge – at 200 samples, it has higher error rate than the $\eta = 0.1$ case.

figure (a) and (b) If η is too big, say 1.0 or 10.0, we see that the model fails to converge on a low error rate.

3. Gaussian Log Likelihood

   The final log likelihood achieved by the Gaussian with separate covariance:

   The final log likelihoods achieved by the Gaussian with shared covariance:

4. Fruit

   Test fruit predictions for Gaussian Model: width 4 cm and height 11 cm: 1 width 8.5 cm and height 7 cm: 1 Test fruit predictions for Shared Covariance Gaussian Model: width 4 cm and height 11 cm: 2 width 8.5 cm and height 7 cm: 1 Test fruit predictions for Linear Regression: width 4 cm and height 11 cm: 2 width 8.5 cm and height 7 cm: 1

   Manually looking on the chart by eye, we might expect that the 1st fruit should fall in the lemon range and the second fruit in the apple range.

   Gaussian (shared) thought 1st one was a lemon and the second one an orange.

   Logistic thought the 1st one was a lemon and the second one a orange.

   Guassian (separate) thought they were both oranges.

   We can see that for all models, more data would have been beneficial. The guassian (covariance separate) couldn't generalize to a massive outlier in the dataset (although, perhaps this fruit is quite unrealistic and so it is correct in assigning a label and then simply returning a high error). None of them ventured into apple territory, and in general seem to be struggling to make a reasonable decisions about what makes apples different from oranges.

   (0 = apple) (red) 1 = orange (blue) 2 = lemon (green)

Calibration [1pt]

Approximately how long did this homework take you to complete?

Between 70 to 90 hours, depending on if you count pre-readings

Name, Email, and Collaborators

Name: Nao Ouyang

Email: nouyang@g.harvard.edu

Collaborators: Eric Wilson Sharon

They are great