Shawn Pan
shawn_pan@g.harvard.edu
CS181-S17

Assignment #2
Due: 5:00pm Feb 24th, 2017

Collaborators: Chin Hui Chew

SOLUTION - Do Not Distribute
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 four 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 explore the properties of the softmax function, which is central to the method of multiclass logistic regression.
- In the third 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 fourth 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

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

4. Justify when you might use this prior in practice.
5. 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

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

6. Derive the special case of the MAP estimator for this problem as the isotropic prior becomes arbitrarily weak. What does the MAP estimator reduce to?
7. What did we observe in lecture about this estimator for the case where the prior is neither weak nor strong?

Solution

1. Let n_0 be the number of 0s in the data and n_1 be the number of 1s. As derived from the lecture 5 slides:

$$\begin{aligned}
\theta_{\text{MLE}} &= \frac{n_1}{n_0 + n_1} \\
\theta_{\text{MAP}} &= \frac{\alpha + n_1 - 1}{\alpha + \beta + n_0 + n_1 - 2} \\
&= \frac{3 + n_1}{4 + n_0 + n_1} \\
p(x = 1 \mid D) &= \frac{\alpha + n_1}{\alpha + \beta + n_1 + n_0} \\
&= \frac{4 + n_1}{6 + n_0 + n_1}
\end{aligned}$$

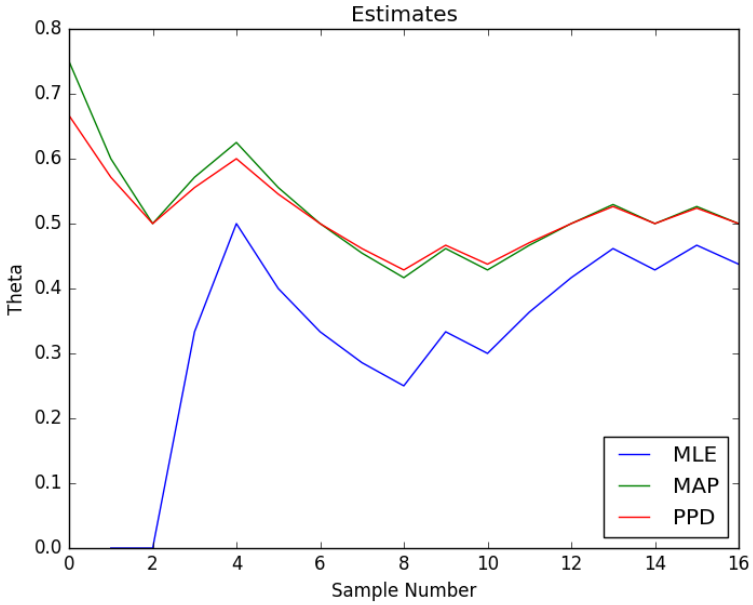

Figure 1: Problem 1.1

2. See plot.

3. The graph for MLE is lower than that of the MAP and PPD, because the MLE does not take the prior into account and the first two data points are 0. As more data points are added, the prior has less importance and the estimates approach each other. The MAP represents the mode of the posterior distribution, while the PPD represents the expected value. The MAP and PPD are very similar, other than the MAP being slightly further from 0.5 than the PPD, due to the distributions having a longer tail on one side.

4. You may want to use a isotropic zero-centered normal prior on weights for regularization. You want your linear regression model to have small magnitude weights to have lower variance, and adding a prior gives a preference for smaller magnitude weights.

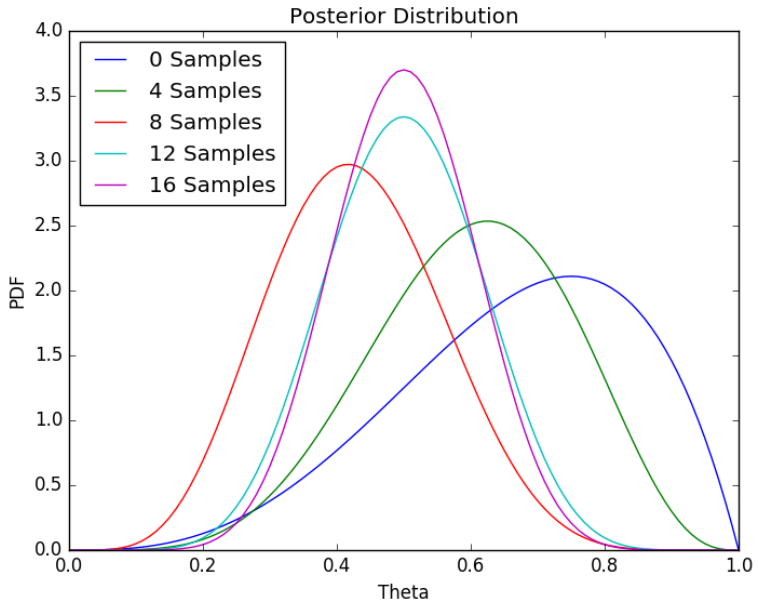

Figure 2: Problem 1.2

5.

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

We now expand out $\ln p(\mathbf{w}|\mathbf{X})$, so we can match terms to complete the square.

$$
\begin{aligned}
\ln p(\mathbf{w}|\mathbf{X}) &= -\frac{1}{2}(\mathbf{w}-\mathbf{m}_n)^T\mathbf{S}_n^{-1}(\mathbf{w}-\mathbf{m}_n) + \text{const} \\
&= \mathbf{w}^T\mathbf{S}_n^{-1}\mathbf{w} - \mathbf{w}^T\mathbf{S}_n^{-1}\mathbf{m}_n - \mathbf{m}_n^T\mathbf{S}_n^{-1}\mathbf{w} + \text{const}
\end{aligned}
$$

Matching terms, we have the following.

$$
\begin{aligned}
\mathbf{S}_n^{-1} &= \beta\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I} \\
\mathbf{S}_n &= (\beta\mathbf{X}^T\mathbf{X} + \alpha\mathbf{I})^{-1} \\
\mathbf{S}_n^{-1}\mathbf{m}_n &= \beta\mathbf{X}^T\mathbf{y} \\
\mathbf{m}_n &= \beta\mathbf{S}_n^{-1}\mathbf{X}^T\mathbf{y}
\end{aligned}
$$

6. With $\alpha = 0$ we have $\mathbf{S}_n = (\beta\mathbf{X}^\top\mathbf{X})^{-1}$ and $\mathbf{m}_n = \beta(\beta\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y}$ which is equivalent to ordinary least squares regression.

7. Linear regression with a normal prior is equivalent to ridge regression. When the prior is a balance between weak and strong, the regularization effect because a tradeoff between bias and variance. With cross-validation, one can find a good prior for the model.

End Solution

Problem 2 (Properties of Softmax, 8pts)

We have explored logistic regression, which is a discriminative probabilistic model over two classes. For each input $\mathbf{x}$, logistic regression outputs a probability of the class output y using the logistic sigmoid function.

The softmax transformation is an important generalization of the logistic sigmoid to the case of c classes. It takes as input a vector, and outputs a transformed vector of the same size,

$$\text{softmax}(\mathbf{z})_k = \frac{\exp(z_k)}{\sum_{\ell=1}^{c} \exp(z_\ell)}, \quad \text{for all } k$$

Multiclass logistic regression uses the softmax transformation over vectors of size c. Let $\{\mathbf{w}_\ell\} = \{\mathbf{w}_1 \dots \mathbf{w}_c\}$ denote the parameter vectors for each class. In particular, multiclass logistic regression defines the probability of class k as,

$$p(\mathbf{y} = C_k \,|\, \mathbf{x}; \{\mathbf{w}_\ell\}) = \text{softmax}([\mathbf{w}_1^\top \mathbf{x} \dots \mathbf{w}_c^\top \mathbf{x}]^\top)_k = \frac{\exp(\mathbf{w}_k^\top \mathbf{x})}{\sum_{\ell=1}^{c} \exp(\mathbf{w}_\ell^\top \mathbf{x})} \,.$$

As above, we are using $\mathbf{y} = C_k$ to indicate the output vector that represents class k.

Assuming data $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$, the negated log-likelihood can be written in the standard form, as

$$\mathcal{L}(\{\mathbf{w}_\ell\}) = -\sum_{i=1}^{n} \ln p(\mathbf{y}_i \,|\, \mathbf{x}_i; \{\mathbf{w}_\ell\})$$

Softmax is an important function in the context of machine learning, and you will see it again in other models, such as neural networks. In this problem, we aim to gain intuitions into the properties of softmax and multiclass logistic regression.

Show that:

1. The output of the softmax function is a vector with non-negative components that are at most 1.
2. The output of the softmax function defines a distribution, so that in addition, the components sum to 1.
3. Softmax preserves order. This means that if elements $z_k < z_\ell$, in $\mathbf{z}$, then $\text{softmax}(\mathbf{z})_k < \text{softmax}(\mathbf{z})_\ell$ for any k, ℓ.
4. Show that
$$\frac{\partial \text{softmax}(\mathbf{z})_k}{\partial z_j} = \text{softmax}(\mathbf{z})_k (I_{kj} - \text{softmax}(\mathbf{z})_j) \quad \text{for any } k, j$$
, where indicator $I_{kj} = 1$ if $k = j$ and $I_{kj} = 0$ otherwise.
5. Using your answer to the previous question, show that
$$\frac{\partial}{\partial \mathbf{w}_k} \mathcal{L}(\{\mathbf{w}_\ell\}) = \sum_{i=1}^{n} (p(\mathbf{y}_i = C_k | \mathbf{x}_i; \{\mathbf{w}_\ell\}) - y_{ik}) \mathbf{x}_i$$
By the way, this may be useful for Problem 3!

Solution

1. Note that $\exp(z) > 0$ for all z. Therefore, $\sum_{\ell=1}^{c} \exp(z_\ell) > 0$. Each component of the output of softmax is therefore a positive number divided by a positive number which is positive. The denominator is the sum of positive numbers including the term in the numerator, so the denominator is greater than the numerator. Therefore, each component is also at most 1.

2.
$$\begin{aligned}\sum_{k=1}^{c} \text{softmax}(\mathbf{z})_k &= \sum_{k=1}^{c} \frac{\exp(z_k)}{\sum_{\ell=1}^{c} \exp(z_\ell)} \\ &= \frac{\sum_{k=1}^{c} \exp(z_k)}{\sum_{\ell=1}^{c} \exp(z_\ell)} \\ &= 1\end{aligned}$$

3.
$$\begin{aligned} z_k &< z_l \\ \exp(z_k) &< \exp(z_l) \\ \frac{\exp(z_k)}{\sum_{m=1}^{c} \exp(z_m)} &< \frac{\exp(z_l)}{\sum_{m=1}^{c} \exp(z_m)} \\ \text{softmax}(\mathbf{z})_k &< \text{softmax}(\mathbf{z})_l \end{aligned}$$

4.
$$\begin{aligned}\frac{\partial \text{softmax}(\mathbf{z})_k}{\partial z_j} &= \frac{\partial}{\partial z_j}\left(\frac{\exp(z_k)}{\sum_{\ell=1}^{c} \exp(z_\ell)}\right) \\ &= \frac{\left(\sum_{\ell=1}^{c} \exp(z_\ell)\right) \frac{\partial}{\partial z_j} \exp(z_k) - \exp(z_k) \frac{\partial}{\partial z_j} \left(\sum_{\ell=1}^{c} \exp(z_\ell)\right)}{\left(\sum_{\ell=1}^{c} \exp(z_\ell)\right)^2} \\ &= \frac{\frac{\partial}{\partial z_j} \exp(z_k)}{\sum_{\ell=1}^{c} \exp(z_\ell)} - \left(\frac{\exp(z_k)}{\sum_{\ell=1}^{c} \exp(z_\ell)}\right)\left(\frac{\exp(z_j)}{\sum_{\ell=1}^{c} \exp(z_\ell)}\right) \\ &= \text{softmax}(\mathbf{z})_k I_{kj} - \text{softmax}(\mathbf{z})_k \text{softmax}(\mathbf{z})_j \\ &= \text{softmax}(\mathbf{z})_k (I_{kj} - \text{softmax}(\mathbf{z})_j)\end{aligned}$$

5. For convenience, we define $S = p(\mathbf{y} = C_k \,|\, \mathbf{x}; \{\mathbf{w}_\ell\}) = \text{softmax}([\mathbf{w}_1^\top \mathbf{x} \dots \mathbf{w}_c^\top \mathbf{x}]^\top)_k$.

$$\begin{aligned}
\frac{\partial}{\partial \mathbf{w}_k} \mathcal{L}(\{\mathbf{w}_\ell\}) &= \frac{\partial}{\partial \mathbf{w}_k} \left(-\sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik} \ln S \right) \\
&= -\sum_{i=1}^{n} y_{ik} \frac{\partial \ln S}{\partial \mathbf{w}_k} \\
&= -\sum_{i=1}^{n} y_{ik} \frac{\partial \ln S}{\partial S} \frac{\partial S}{\partial \mathbf{w}_k^T \mathbf{x}_i} \frac{\partial \mathbf{w}_k^T \mathbf{x}_i}{\partial \mathbf{w}_k} \\
&= -\sum_{i=1}^{n} y_{ik} \frac{1}{S} S(1-S)\mathbf{x}_i \\
&= \sum_{i=1}^{n} (S - y_{ik})\mathbf{x}_i \\
&= \sum_{i=1}^{n} (p(\mathbf{y}_i = C_k | \mathbf{x}_i; \{\mathbf{w}_\ell\}) - y_{ik})\mathbf{x}_i
\end{aligned}$$

End Solution

Problem 3 (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.

$$\begin{aligned}
-\ln p(D;\boldsymbol{\pi}) &= -\ln \prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{y}_i;\boldsymbol{\pi}) \\
&= -\ln \prod_{i=1}^{n} \sum_{k=1}^{c} y_{ik} p(\mathbf{x}_i|\mathbf{y}_i = C_k) p(\mathbf{y}_i = C_k;\boldsymbol{\pi}) \\
&= -\sum_{i=1}^{n} \ln \sum_{k=1}^{c} y_{ik} p(\mathbf{x}_i|\mathbf{y}_i = C_k) p(\mathbf{y}_i = C_k;\boldsymbol{\pi}) \\
&= -\sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik} \ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) p(\mathbf{y}_i = C_k;\boldsymbol{\pi})
\end{aligned}$$

Note that the $\sum_{k=1}^{c} y_{ik}$ term selects the correct class corresponding to y_i. The last step is valid, because y_i is a one-hot encoded vector and exactly one element in the sum in non-zero.

$$\begin{aligned}
&= -\sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik} \ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) - \sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik} \ln p(\mathbf{y}_i = C_k;\boldsymbol{\pi}) \\
&= -\sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik} \ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) - \sum_{i=1}^{n} \sum_{k=1}^{c} y_{ik} \ln \pi_k
\end{aligned}$$

2.

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

This answer makes sense, because it is simply the fraction of $\mathbf{y}$ of a particular class observed in the data.

3. For convenience, we lump off terms that are constant with respect to $\boldsymbol{\mu}_k$. Note that because the

covariance matrix is symmetric, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$.

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\mu}_k} &= \frac{\partial}{\partial \boldsymbol{\mu}_k}\left(-\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} \ln \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \text{const}_{\boldsymbol{\mu}_k}\right) \\
&= \frac{\partial}{\partial \boldsymbol{\mu}_k}\left(-\sum_{i=1}^{n} y_{ik}\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right) + \text{const}_{\boldsymbol{\mu}_k}\right) \\
&= -\sum_{i=1}^{n} y_{ik}\left(-\frac{1}{2}(\boldsymbol{\Sigma}^{-1} + \boldsymbol{\Sigma}^{-T})(\mathbf{x}_i - \boldsymbol{\mu}_k)(-1)\right) \\
&= -\sum_{i=1}^{n} y_{ik}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)
\end{aligned}$$

4. We find the MLE by setting the gradient to 0.

$$\begin{aligned}
0 &= -\sum_{i=1}^{n} y_{ik}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k) \\
\sum_{i=1}^{n} y_{ik}\mathbf{x}_i &= \sum_{i=1}^{n} y_{ik}\boldsymbol{\mu}_k \\
\hat{\boldsymbol{\mu}_k} &= \frac{\sum_{i=1}^{n} y_{ik}\mathbf{x}_i}{\sum_{i=1}^{n} y_{ik}}
\end{aligned}$$

This answer makes sense, because it is the mean $\mathbf{x}$ among the data points of the given class.

5. For convenience, we lump off terms that are constant with respect to $\boldsymbol{\Sigma}$. Note that because the covariance matrix is symmetric, $\boldsymbol{\Sigma} = \boldsymbol{\Sigma}^T$.

$$\begin{aligned}
\frac{\partial L}{\partial \boldsymbol{\Sigma}} &= \frac{\partial}{\partial \boldsymbol{\Sigma}}\left(-\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik} \ln \mathcal{N}(\mathbf{x}_i|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}) + \text{const}_{\boldsymbol{\Sigma}}\right) \\
&= \frac{\partial}{\partial \boldsymbol{\Sigma}}\left(-\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\left(-\frac{1}{2}\ln|\boldsymbol{\Sigma}| - \frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right) + \text{const}_{\boldsymbol{\Sigma}}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\left(\boldsymbol{\Sigma}^{-T} - \boldsymbol{\Sigma}^{-T}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-T}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\boldsymbol{\Sigma}^{-T}\left(I - (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-T}\right) \\
&= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\boldsymbol{\Sigma}^{-1}\left(I - (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}\right)
\end{aligned}$$

6. We find the MLE by setting the gradient to 0.

$$
\begin{aligned}
0 &= \frac{1}{2}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\boldsymbol{\Sigma}^{-1}\left(\mathbf{I} - (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}\right) \\
0 &= \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\left(\mathbf{I} - (\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}\right) \\
\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}\mathbf{I} &= \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1} \\
n\mathbf{I} &= \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1} \\
n\boldsymbol{\Sigma} &= \sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T \\
\hat{\boldsymbol{\Sigma}} &= \frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{c} y_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T
\end{aligned}
$$

End Solution

4. 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
...
```

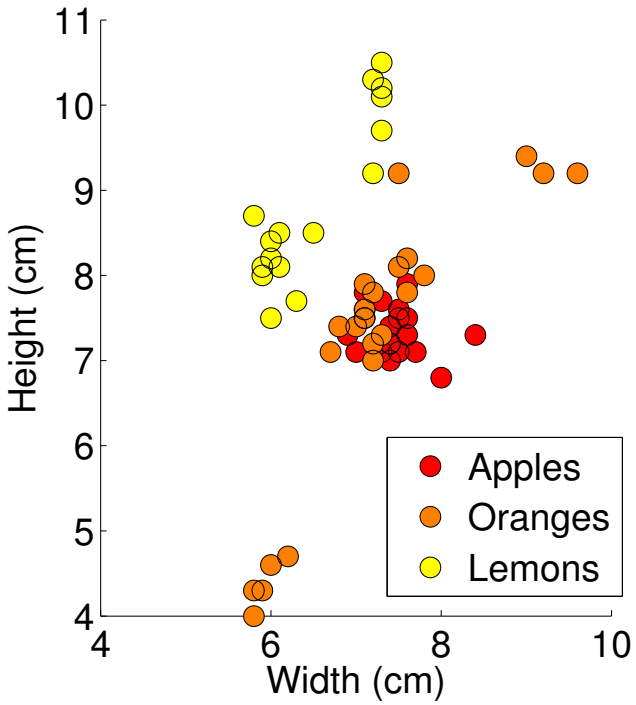

Figure 3: Heights and widths of apples, oranges, and lemons. These fruit were purchased and measured by Iain Murray: http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

Problem 4 (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.
- 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 `problem4.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, report negative log-likelihood loss for several configurations of hyperparameters. Why are your final choices of learning rate (η) and regularization strength (λ) reasonable? Plot loss during training for the best of these configurations, with iterations on the x-axis and loss on the y-axis (one way to do this is to add a method to the LogisticRegression Class that displays loss).
- For both Gaussian generative models, report likelihood. In the separate covariance matrix case, be sure to use the covariance matrix that matches the true class of each data point.

Table 1: Logistic regression losses

η	λ	iterations	loss
10^{-5}	0	140874	32.48
10^{-5}	0.01	132960	32.88
10^{-5}	0.1	105161	35.32
10^{-5}	1	89798	42.49
10^{-4}	0	14087	32.48
10^{-4}	0.01	13296	32.88
10^{-4}	0.1	10516	35.32
10^{-4}	1	8980	42.49

Solution

The generative model with separate covariances performs the best. All three classifiers separate out the green (lemons) well. Logistic regression and the generative model with shared covariances struggle to separate the red (apples) and blue (oranges). They produce linear decision boundaries and the data are not linearly separable. The generative model with separate covariances performs better separating the red and blue, because it can produce non-linear decision boundaries, e.g. a tight ellipse around the red region. However, there is still one region where red and blue points overlap and separation is impossible.

We select a learning rate $\eta = 10^{-4}$, because smaller step sizes take longer to converge during gradient descent, and a larger $\eta = 10^{-3}$ does not converge. We would need to do cross-validation on a testing set to select an appropriate regularization parameter λ, although 0, 0.01, and 0.1 all seem to give similiar decision boundaries. Note that loss increases with regularization because we penalize larger magnitude weights more, so if we only cared about the training data $\lambda = 0$ gives the least bias. Interestingly, higher regularization also converges slightly faster.

The generative model with separate covariances has a training loss (negative log likelihood) of 158.15, while the model with shared covariances has a loss of 208.41. The model with separate covariances has more degrees of freedom and can better fit the data.

End Solution

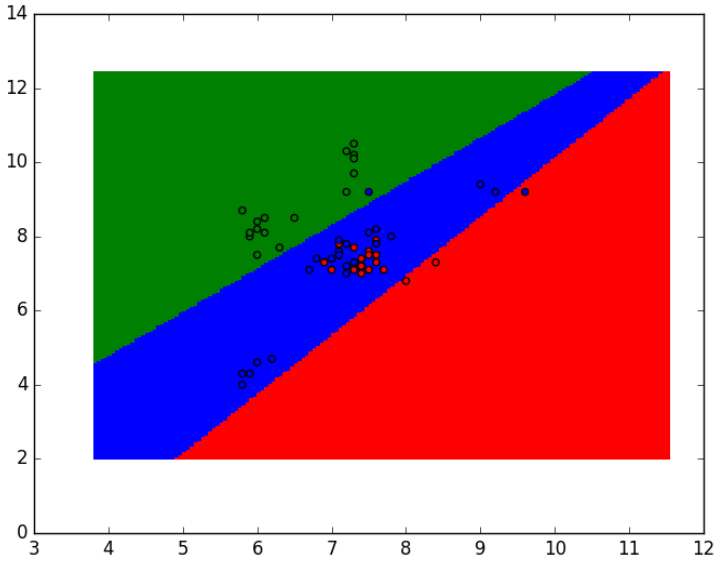

Figure 4: Logistic regression for $\eta = 10^{-4}$ and $\lambda = 0$.

Figure 5: Training loss for $\eta = 10^{-4}$ and $\lambda = 0$.

Calibration [1pt]

Approximately how long did this homework take you to complete? 22 hours

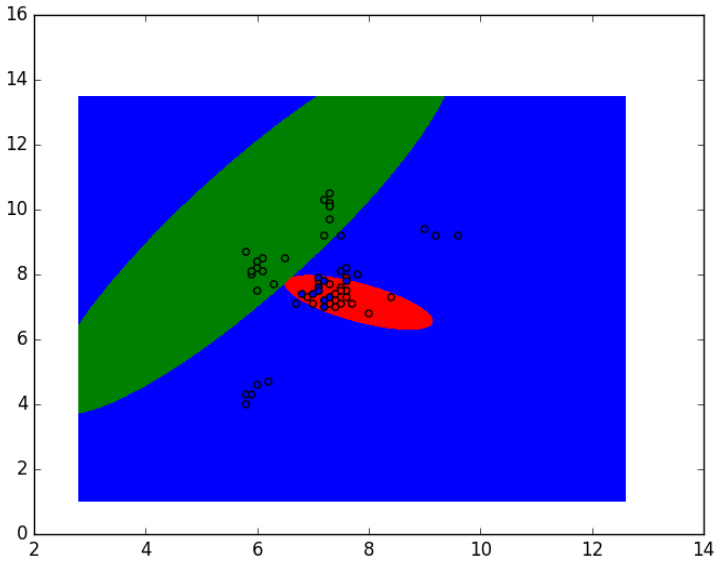

Figure 6: Generative model with separate covariances.

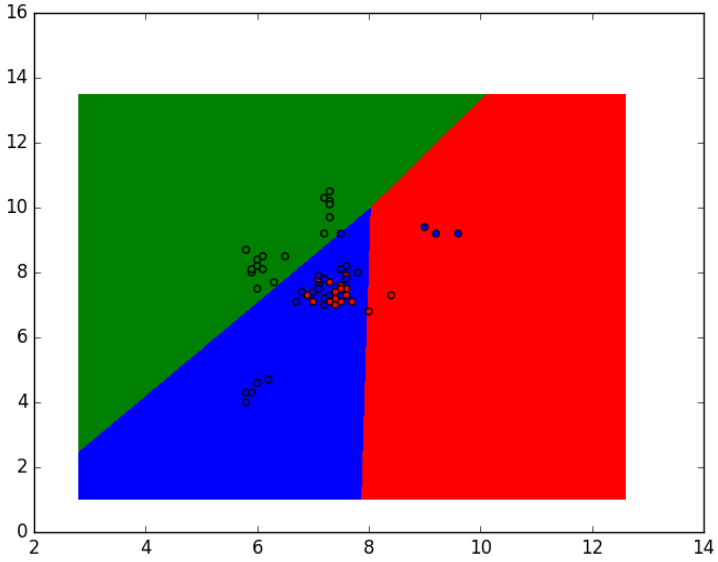

Figure 7: Generative model with shared covariances.