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

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

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

$$\begin{aligned}-\ln p(D; \boldsymbol{\pi}) &= -\sum_{i=1}^{n} \sum_{k=1}^{c} \ln(\pi_k) + \ln p(x_i|C_k) \\ &= -n \sum_{k=1}^{c} \ln(\pi_k) - \sum_{i=1}^{n} \sum_{k=1}^{c} \ln p(x_i|C_k)\end{aligned}$$

2. 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 likelihood function with respect to π_k.

   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 may drop the class conditional term.

$$\begin{aligned}\arg\max_{\boldsymbol{\pi}_k} - \ln L(D) &= \arg\max_{\boldsymbol{\pi}_k} -n \sum_{k=1}^{c} \ln(\pi_k) - \sum_{i=1}^{n} \sum_{k=1}^{c} \ln p(x_i|C_k) \\ &= \arg\max_{\boldsymbol{\pi}_k} -n \sum_{k=1}^{c} \ln(\pi_k)\end{aligned}$$

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

Substituing into the equation

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

we get

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

To find the optimum, we then solve the system of two equations $\frac{\partial}{\partial \boldsymbol{\pi}} Lag(\boldsymbol{\pi}_k, \lambda) = 0$ and $\frac{\partial}{\partial \lambda} Lag(\boldsymbol{\pi}_k, \lambda) = 0$.

$$\frac{\partial}{\partial \boldsymbol{\pi}} Lag(\boldsymbol{\pi}, \lambda) = n\pi_k + \lambda$$

Set equal to zero and we get that $\pi_i = \lambda/n$.

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

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.

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.

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
...
```

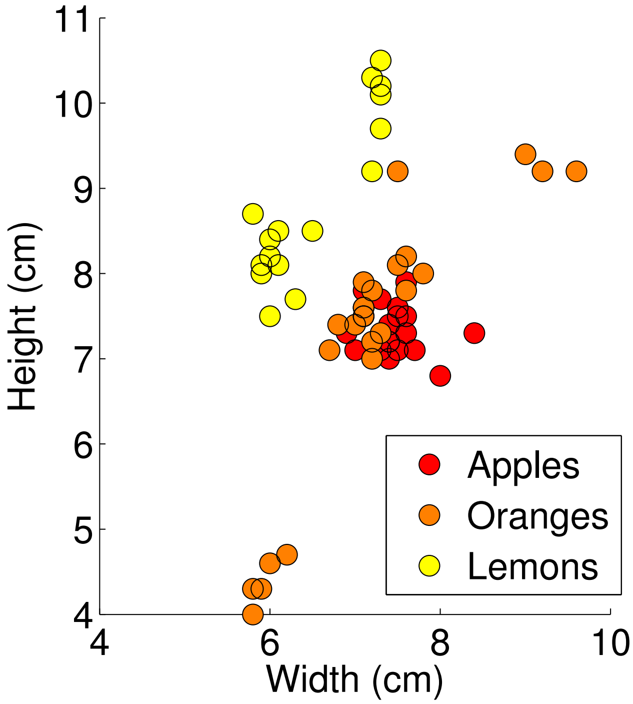

Figure 1: Heights and widths of apples, oranges, and lemons. These fruit were purchased and measured by Iain Murray: http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

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.

Calibration [1pt]

Approximately how long did this homework take you to complete?

Name, Email, and Collaborators

Name:

Email:

Collaborators: