Assignment #2

Due: 11:59pm Mar 1st, 2019

Homework 2: Bayesian Methods and Multiclass Classification

Introduction

This homework is about Bayesian methods and multiclass classification. In lecture we have primarily focused on binary classifiers trained to discriminate between two classes. In multiclass classification, we discriminate between three or more classes. We encourage you to first read the Bishop textbook coverage of these topic, particularly: Section 4.2 (Probabilistic Generative Models), Section 4.3 (Probabilistic Discriminative Models).

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

The problem set has three problems:

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

Problem 1 (Bayesian Methods, 10 pts)

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

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

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

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

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

Solution

Problem 2 (Return of matrix calculus, 10pts)

Consider now a generative c-class model. We adopt class prior $p(\mathbf{y} = C_k; \boldsymbol{\pi}) = \pi_k$ for all $k \in \{1, \ldots, c\}$ (where π_k is a parameter of the prior). Let $p(\mathbf{x}|\mathbf{y} = C_k)$ denote the class-conditional density of features $\mathbf{x}$ (in this case for class C_k). Consider the data set $D = \{(\mathbf{x}_i, \mathbf{y}_i)\}_{i=1}^n$ where as above $\mathbf{y}_i \in \{C_k\}_{k=1}^c$ is encoded as a one-hot target vector.

1. Write out the negated log-likelihood of the data set, $-\ln p(D; \boldsymbol{\pi})$.
2. Since the prior forms a distribution, it has the constraint that $\sum_k \pi_k - 1 = 0$. Using the hint on Lagrange multipliers below, give the expression for the maximum-likelihood estimator for the prior class-membership probabilities, i.e. $\hat{\pi}_k$. Make sure to write out the intermediary equation you need to solve to obtain this estimator. Double-check your answer: the final result should be very intuitive!

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

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

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

3. Derive the gradient of the negative log-likelihood with respect to vector $\boldsymbol{\mu}_k$. Write the expression in matrix form as a function of the variables defined throughout this exercise. Simplify as much as possible for full credit.
4. Derive the maximum-likelihood estimator for vector $\boldsymbol{\mu}_k$. Once again, your final answer should seem intuitive.
5. Derive the gradient for the negative log-likelihood with respect to the covariance matrix $\boldsymbol{\Sigma}$ (i.e., looking to find an MLE for the covariance). Since you are differentiating with respect to a *matrix*, the resulting expression should be a matrix!
6. Derive the maximum likelihood estimator of the covariance matrix.

Hint: Lagrange Multipliers. Lagrange Multipliers are a method for optimizing a function f with respect to an equality constraint, i.e.

$$\min_{\mathbf{x}} f(\mathbf{x}) \text{ s.t. } g(\mathbf{x}) = 0.$$

This can be turned into an unconstrained problem by introducing a Lagrange multiplier λ and constructing the Lagrangian function,

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

It can be shown that it is a necessary condition that the optimum is a critical point of this new function. We can find this point by solving two equations:

$$\frac{\partial L(\mathbf{x}, \lambda)}{\partial \mathbf{x}} = 0 \text{ and } \frac{\partial L(\mathbf{x}, \lambda)}{\partial \lambda} = 0$$

Cookbook formulas. Here are some formulas you might want to consider using to compute difficult gradients. You can use them in the homework without proof. If you are looking to hone your matrix calculus skills, try to find different ways to prove these formulas yourself (will not be part of the evaluation of this homework). In general, you can use any formula from the matrix cookbook, as long as you cite it. We opt for the following common notation: $\mathbf{X}^{-\top} := (\mathbf{X}^\top)^{-1}$

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

Solution

3. Classifying Stars [15pts]

You're tasked with classifying three different kinds of stars, based on their magnitudes and temperatures. The figure below is a plot of the data, adapted from http://astrosci.scimuze.com/stellar_data.htm and available as `hr.csv`, which you will find in the Github repository. The file has three columns: type, magnitude, and temperature. The first few lines look like this:

```
Type, Magnitude, Temperature
Dwarf,-5.8,-0.35
Dwarf,-4.1,-0.31
Dwarf,-1.1,-0.16
...
```

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

Problem 3 (Classifying Stars, 15pts)

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

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

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

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

Solution

- Name:
- Email:
- Collaborators:
- Approximately how long did this homework take you to complete (in hours):