Your Name
email@fas.harvard.edu
CS181-S17

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

Collaborators: John Doe, Fred Doe

SOLUTION - Do Not Distribute
Homework 2: Bayesian Methods and Multiclass Classification

Grading Instructions

In the solutions, you will see several highlighted checkpoints. These each have a label that corresponds to an entry in the Canvas quiz for this problem set. The highlighted statement should clearly indicate the criteria for being correct on that problem. If you satisfy the criteria for a problem being correct, mark "Yes" on the corresponding position on the Canvas quiz. Otherwise, mark "No". Your homework scores will be verified by course staff at a later date.

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?

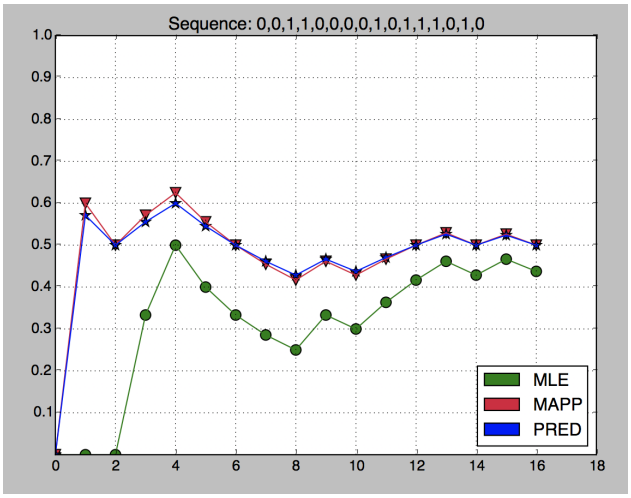
Solution

1. Let $n_1 = \#$ ones and $n_0 = \#$ zeros.

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

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

$$\boxed{\text{Posterior Predictive} = \frac{\alpha + n_1}{\alpha + \beta + n_1 + n_0}}$$



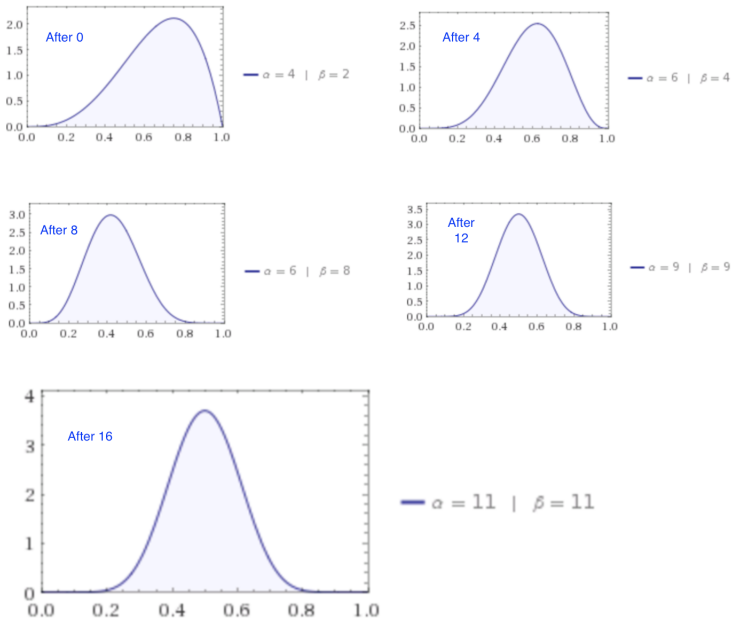
Check 1.1: You listed the correct expressions for the MLE, MAP, and posterior predictive, and your plot looks like the above.

	after 0	after 4	after 8	after 12	after 16
$n_1 + \alpha$	4	6	6	9	11
$n_0 + \beta$	2	4	8	9	11

2. We start with the prior $Beta(\alpha = 4, \beta = 2)$. Writing the posterior, the conjugate property gives us another Beta (with updated parameters):

$$p(\theta|D) \propto p(D|\theta)p(\theta) = \theta^{n_1+\alpha-1}(1-\theta)^{n_0+\beta-1} = Beta(\theta|n_1 + \alpha, n_0 + \beta)$$

Plot the (possibly unnormalized) Beta distributions. Notice the movement toward symmetry as we see more data points:



Check 1.2: Your plots should look identical to the above.

3. (See plot from question 1). The MAP and posterior predictive are both significantly higher than the MLE because they incorporate the prior $\theta \sim Beta(4, 2)$. This is especially evident at the beginning, where we see that the first two observations of 0 put the MLE at 0, while the MAP and posterior predictive are closer to 0.6. The MLE moves closer to the MAP and posterior predictive with more observations, as the prior becomes less influential on the estimators. The MAP and posterior predictive are close from the beginning and track each other even more closely with more observations as the constant terms that differ in their formulas are outweighed by more data.

Check 1.3: You gave similar explanations and did not contradict the above.

4. This prior makes sense when you do not know much about the relationship among features and choose to simplify by assuming independence.

Check 1.4: You gave a similar explanation for an isotropic (spherical) prior, highlighting that it is justified when you have little (prior) information about feature representations

5. Recall the generic setup to this problem.

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

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

Taking the logs of the PDFs we are left with:

$$= \text{const} - \frac{1}{2}\Big((\mathbf{w} - \mathbf{m}_0)^\top \mathbf{S}_0^{-1}(\mathbf{w} - \mathbf{m}_0) + \beta(\mathbf{y} - \mathbf{X}\mathbf{w})^\top(\mathbf{y} - \mathbf{X}\mathbf{w})\Big)$$

Let $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{S}_0 = \alpha^{-1}\mathbf{I}$

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

Expand and move non-$\mathbf{w}$ terms to const

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

Detour: recall that we already know the form of the Normal-Normal posterior:

$$\text{const} - \frac{1}{2}(\mathbf{w} - \mathbf{m}_n)^\top \mathbf{S}_n^{-1}(\mathbf{w} - \mathbf{m}_n)$$

Expand this term:

$$\text{const} - \frac{1}{2}\Big(\mathbf{w}^\top\mathbf{S}_n^{-1}\mathbf{w} - \mathbf{m}_n^\top\mathbf{S}_n^{-1}\mathbf{w} - \mathbf{w}^\top\mathbf{S}_n^{-1}\mathbf{m}_n + \mathbf{m}_n^\top\mathbf{S}_n^{-1}\mathbf{m}_n\Big)$$

Factor in the following way

$$\text{const} - \frac{1}{2}\Big(\mathbf{w}^\top\mathbf{S}_n^{-1}\mathbf{w} - 2\mathbf{w}^\top(\mathbf{S}_n^{-1}\mathbf{m}_n)\Big)$$

This term looks exactly like where we left off before the detour, where

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

and

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

This means that

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

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

Check 1.5: You reduced the expression for the posterior on weights to the form given in the problem statement. You should have followed approximately the same steps, and your steps should be correct.

6. In the case of a weak prior and spherical covariance matrix, $\mathbf{S}_0$ has large entries on the diagonal. The diagonal entries of $\mathbf{S}_0^{-1}$ are close to zero. The $\alpha\mathbf{I}$ term disappears (contributes a negligible amount to the matrix sum) as α goes to zero:

$$\mathbf{S}_n \approx \beta^{-1}(\mathbf{X}^\top\mathbf{X})^{-1}$$

And we get an interesting result for $\mathbf{m}_n$:

$$\boxed{\mathbf{m}_n \approx \beta^{-1}(\mathbf{X}^\top\mathbf{X})^{-1}\beta\mathbf{X}^\top\mathbf{y} = (\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = \theta_{\text{MLE}}}$$

Check 1.6: You found the above expression for the MAP and showed it was equivalent to the MLE (in this special case).

7. Suppose $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$. Posterior is $\mathbf{w} \sim \mathcal{N}(\mathbf{m}_n, \mathbf{S}_n)$, with

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

We see that

$$\boxed{\mathbf{w}_{\text{MAP}} = \mathbf{m}_n = \beta(\alpha\mathbf{I} + \beta\mathbf{X}^\top\mathbf{X})^{-1}\mathbf{X}^\top\mathbf{y} = (\mathbf{X}^\top\mathbf{X} + \frac{\alpha}{\beta}\mathbf{I})^{-1}\mathbf{X}^\top\mathbf{y}}$$

and we recover ridge regression.

Check 1.7: You found the above expression for the MAP and stated that this was equivalent to ridge regression.

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. The j^{th} component of the softmax function softmax($\mathbf{z}$) is:

$$\text{softmax}(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_i \exp(z_i)}.$$

As $\exp(x) > 0$ for all $x \in \mathbb{R}$, we have $\exp(z_j) > 0$ and $\sum_i \exp(z_i) > 0$. Thus the output of the softmax function is a vector with non-negative components. Since $\exp(z_j)$ appears in both the numerator and the denominator (as the $i = j$ term in the sum), the denominator must be at least as large as the numerator, and so the components are at most 1.

Check 2.1: You must both indicate that $\exp(z_j)$ is positive, and that $\sum_i \exp(z_i) \geq \exp(z_j)$. You will not get points if you missed either of these.

2. Summing over the components:

$$\sum_j \text{softmax}(\mathbf{z})_j = \sum_j \frac{\exp(z_j)}{\sum_i \exp(z_i)} = \frac{\sum_j \exp(z_j)}{\sum_i \exp(z_i)} = 1.$$

Check 2.2: You summed over the components to show that it equals 1.

3. If $z_j \geq z_k$, then $\exp(z_j) \geq \exp(z_k)$ as the exponential is a monotonically increasing function. Dividing by the positive constant $\sum_i \exp(z_i)$, this inequality implies that:

$$\sigma(\mathbf{z})_j = \frac{\exp(z_j)}{\sum_i \exp(z_i)} \geq \frac{\exp(z_k)}{\sum_i \exp(z_i)} = \text{softmax}(\mathbf{z})_k,$$

which shows that the softmax function preserves the order of the elements of $\mathbf{z}$.

Check 2.3: You either stated (or briefly showed via derivative) that exp is a monotonically increasing function. This means that it preserves order. You can either simply state that, or show it more formally as above.

4. To relate our notation to Bishop (4.106), note that $y_k = \text{softmax}(\mathbf{z})_k$ and $a_j = z_j$.

If $j \neq k$, then:

$$\begin{aligned}\frac{\partial \text{softmax}(\mathbf{z})_k}{\partial z_j} &= \frac{\partial}{\partial z_j}\frac{\exp(z_k)}{\sum_i \exp(z_i)} = -\frac{\exp(z_k)}{\left(\sum_i \exp(z_i)\right)^2}\exp(z_j) \\ &= -\frac{\exp(z_k)}{\sum_i \exp(z_i)}\frac{\exp(z_j)}{\sum_i \exp(z_i)} = -\text{softmax}(\mathbf{z})_k \text{softmax}(\mathbf{z})_j.\end{aligned}$$

If $j = k$ then:

$$\begin{aligned}\frac{\partial \text{softmax}(\mathbf{z})_k}{\partial z_j} &= \frac{\partial}{\partial z_j}\frac{\exp(z_k)}{\sum_i \exp(z_i)} = \frac{\exp(z_k)}{\sum_i \exp(z_i)} - \frac{\exp(z_j)^2}{\left(\sum_i \exp(z_i)\right)^2} \\ &= \left(1 - \frac{\exp(z_k)}{\sum_i \exp(z_i)}\right)\frac{\exp(z_k)}{\sum_i \exp(z_i)} = \text{softmax}(\mathbf{z})_k(1 - \text{softmax}(\mathbf{z})_j).\end{aligned}$$

Putting these results together: $\boxed{\frac{\partial \text{softmax}(\mathbf{z})_k}{\partial z_j} = \text{softmax}(\mathbf{z})_k(I_{kj} - \text{softmax}(\mathbf{z})_j)}$

Check 2.4: You broke the problem into the case where $j = k$ and $j \neq k$, and differentiated separately. Then, you combined them into a single term using the identity matrix, as above.

5. Write the negative log-likelihood. **NOTE:** this solution differentiates the loss with respect to $\mathbf{w}_j$ here (unlike $\mathbf{w}_k$ in the problem statement), to avoid confusion with the k used for indexing the summation over classes:

$$\mathcal{L}(\{\mathbf{w}_\ell\}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} \ln p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})$$

$$\frac{\partial}{\partial \mathbf{w}_j}\mathcal{L}(\{\mathbf{w}_\ell\}) = -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} \frac{\partial}{\partial \mathbf{w}_j} \ln p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} \left(\frac{1}{p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})}\right) \frac{\partial}{\partial \mathbf{w}_j} p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} \left(\frac{1}{p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})}\right) \frac{\partial}{\partial z_j} p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})\mathbf{x}_i$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} \left(\frac{1}{p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})}\right) \Big(p(\mathbf{y} = C_k|\mathbf{x}_i; \{\mathbf{w}_\ell\})\Big) \Big(I_{kj} - p(\mathbf{y} = C_j|\mathbf{x}_i; \{\mathbf{w}_\ell\})\Big)\mathbf{x}_i$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} \Big(I_{kj} - p(\mathbf{y} = C_j|\mathbf{x}_i; \{\mathbf{w}_\ell\})\Big)\mathbf{x}_i$$

$$= -\sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik} I_{kj}\mathbf{x}_i + \sum_{i=1}^{N} p(\mathbf{y} = C_j|\mathbf{x}_i; \{\mathbf{w}_\ell\})\mathbf{x}_i \Big(\sum_{k=1}^{C} y_{ik}\Big)$$

The I_{kj} in the first sum collapses the sum over k to the term where $j = k$. As y_{ik} form the components of a 1-of-C encoding, we have that $\sum_{k=1}^{C} y_{ik} = 1$. Using these facts:

$$\frac{\partial}{\partial \mathbf{w}_j}\mathcal{L}(\{\mathbf{w}_\ell\}) = -\sum_{i=1}^{N} y_{ij}\mathbf{x}_i + \sum_{i=1}^{N} p(\mathbf{y} = C_j|\mathbf{x}_i; \{\mathbf{w}_\ell\})\mathbf{x}_i$$

$$\boxed{= \sum_{i=1}^{N} (p(\mathbf{y} = C_j|\mathbf{x}_i; \{\mathbf{w}_\ell\}) - y_{ij})\mathbf{x}_i}$$

Check 2.5: You took the gradient correctly, and reduced it to the form shown in the problem statement. You should have used approximately the same steps, and your steps should be correct.

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. The log-likelihood is given by:

$$\boxed{\ln p(\{\mathbf{x}_i, \mathbf{y}_i\}|\{\pi_k\}) = \sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik}\big(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k\big)}$$

Note that the class prior π_k is part of the likelihood, as opposed to a *model prior*

Check 3.1: Your log likelihood should be the same as above (or an equivalent expression). It must include the class prior π_k.

2. Using the note at the end of the exercise, we know we need to maximize:

$$\ln p(\{\mathbf{x}_i, \mathbf{y}_i\}|\{\pi_k\}) = \sum_{i=1}^{N}\sum_{k=1}^{C} y_{ik}\big(\ln p(\mathbf{x}_i|\mathbf{y}_i = C_k) + \ln \pi_k\big) + \lambda\left(\left(\sum_{k=1}^{C}\pi_k\right) - 1\right)$$

Check 3.2: You added the Lagrange multiplier term (the term with λ above).

We take the derivative with respect to π_k and set it to 0:

$$\sum_{i=1}^{N}\frac{y_{ik}}{\pi_{ik}} + \lambda = 0$$

Rearrange to solve for π_k:

$$\pi_k = \frac{-1}{\lambda}\sum_{i=1}^{N} y_{ik}$$

Remember our original constraint:

$$\sum_{k=1}^{C}\pi_k = 1$$

Plug π_k into the original constraint:

$$\sum_{k=1}^{C}\left(\frac{-1}{\lambda}\sum_{i=1}^{N} y_{ik}\right) = 1$$

Solve for λ:

$$\lambda = -\sum_{k=1}^{C}\sum_{i=1}^{N} y_{ik} = -N$$

$$\boxed{\hat{\pi_k} = \frac{1}{N}\sum_{i=1}^{N} y_{ik}}$$

Note that the MLE for the class-prior simply reflects the proportion of classes in our dataset.

Check 3.3: You correctly solved for $\hat{\pi}_k$.

3. The log-likelihood can be written as such:

$$\ln p(\{\mathbf{x}_i, \mathbf{y}_i\}|\{\pi_k\}) = \sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)\right) + \text{constants w.r.t } \boldsymbol{\mu}_k$$

Using Equation 86 from the Matrix Cookbook, the gradient w.r.t $\boldsymbol{\mu}_k$ can be written as:

$$\boxed{\sum_{i=1}^{N} y_{ik}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}_k)}$$

Check 3.4: You correctly solved for the gradient.

4. We set the previous gradient equal to 0 to obtain:

$$\boxed{\hat{\mu}_k = \frac{1}{\sum_{i=1}^{N} y_{ik}}\sum_{i=1}^{N} y_{ik}\mathbf{x}_i}$$

Again, an intuitive answer. Our best guess for the average feature vector for a given class is the average of all feature vectors of that class in our dataset.

Check 3.5: You correctly solved for $\hat{\mu}_k$.

5. Starting with the log likelihood and using the two formulas in the cookbook mentioned above, the gradient w.r.t $\boldsymbol{\Sigma}$ can be written as:

$$\boxed{\sum_{i=1}^{N}\sum_{k=1}^{K} y_{ik}\left[-\frac{1}{2}\boldsymbol{\Sigma}^{-T} + \frac{1}{2}\boldsymbol{\Sigma}^{-T}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}^{-T}\right]}$$

Check 3.6: You correctly calculated the gradient as above, or an equivalent expression.

6. Setting it to 0 and multiply both sides by $\boldsymbol{\Sigma}^T$ from the left and right:

$$\sum_{i,k} y_{ik}\boldsymbol{\Sigma}^T = \sum_{k,i} y_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T$$

Taking the transpose (any matrix VV^T is symmetric), we have:

$$\boxed{\hat{\boldsymbol{\Sigma}} = \frac{1}{\sum_{i,k} y_{ik}}\sum_{k,i} y_{ik}(\mathbf{x}_i - \boldsymbol{\mu}_k)(\mathbf{x}_i - \boldsymbol{\mu}_k)^T}$$

Check 3.7: You correctly found the MLE for the covariance matrix. Note that this is the weighted average of the covariance matrix for each class separately.

End Solution

4. Classifying Fruit [15pts]

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

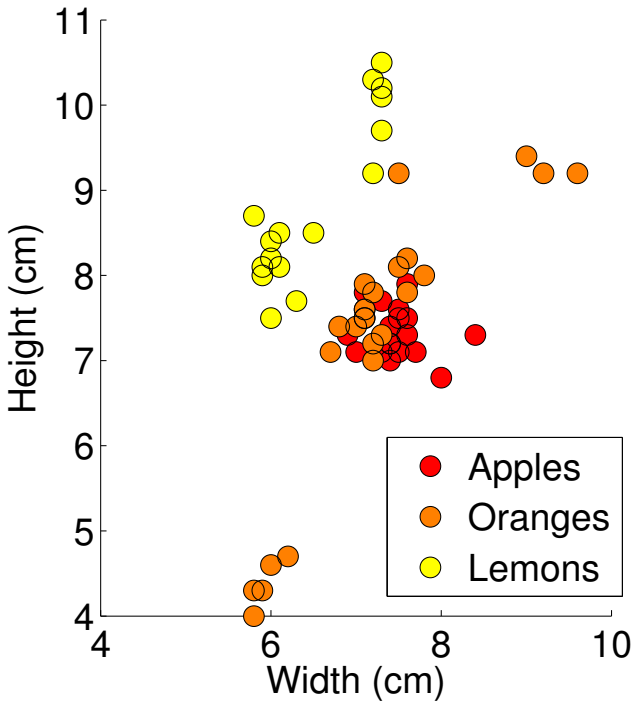


Figure 1: Heights and widths of apples, oranges, and lemons. These fruit were purchased and measured by Iain Murray: http://homepages.inf.ed.ac.uk/imurray2/teaching/oranges_and_lemons/.

Problem 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.

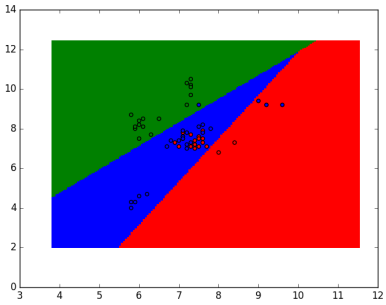
- The `visualize()` method for each classifier will save a plot that will show the decision boundaries. You should include these in this assignment.
- Which classifiers model the distributions well?
- What explains the differences?

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

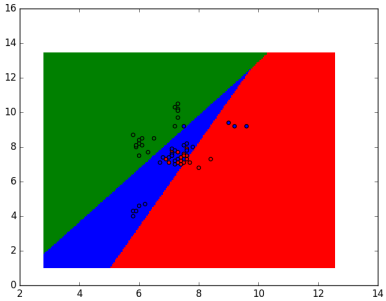
- For logistic regression, report negative log-likelihood loss for several configurations of hyperparameters. Why are your final choices of learning rate (η) and regularization strength (λ) reasonable? Plot loss during training for the best of these configurations, with iterations on the x-axis and loss 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.

Solution

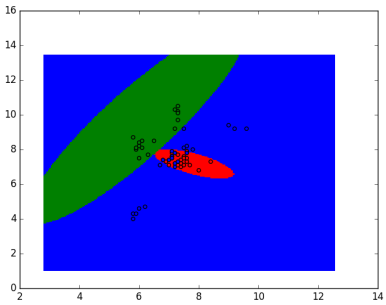
Your plots should look like the ones below. These are, in order, multiclass logistic regression classifier, the generative classifier with a shared covariance matrix, and the generative classifier with separate covariance matrices.



Check 4.1: Your plot for the logistic regression classifier matches the one above. The staff solution ran 500,000 epochs to achieve this solution.



Check 4.2: Your plot for the generative classifier with shared covariances matches the one above. It should look almost identical, if not exactly identical.



Check 4.3: Your plot for the generative classifier with separative covariances matches the one above. It should look almost identical, if not exactly identical.

Which classifiers model the distributions well?

This is a pretty open-ended question, so we will generally accept a broad range of answers. A few things that should be pretty clear: 1) the generative model with separate covariance matrices seems to model the distribution better than the generative model with a shared covariance matrix; 2) logistic regression and the generative model with shared covariance perform similarly.

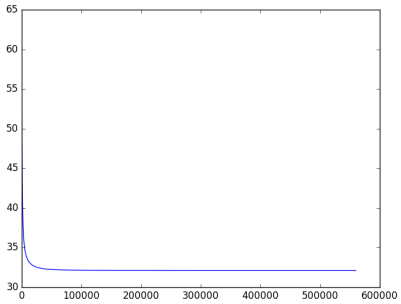
> **Check 4.4**: Your observations are consistent with the plots. As long as you present reasonable observations, and none of them contradict the examples above, you get the points.

What explains the differences?

Having a shared covariance matrix means that each of the three classes have approximately the same shape, which is elongated from the bottom left to the top-right of the plot. However, just from visual inspection, we can see that the three classes actually have fairly different shapes, with some being elongated from bottom left to top right, and the red ones being at a right angle to those. So, having separate covariance matrices for each one seems to lead to a better-looking fit. Logistic Regression generally performs well, but since it enforces linear decision boundaries, there is a limit to its ability to model this data.

> **Check 4.5**: You noted why the separate covariance case is different from the shared covariance case and that logistic regression is limited to linear decision boundaries. Also, you didn't say anything that directly contradicts the above.

Logistic Regression Quantitative Analysis



The above plot of loss during training came from using the hyperparameters $\eta = 0.0001$ and $\lambda = 0.1$. You could have chosen different final values for hyperparameters, but you should have seen that training with significantly larger values of η failed to converge and that smaller values of η caused longer training times but successfully converged.

> **Check 4.6**: You reported negative (log)likelihood loss for different configurations of hyperparameters and explained why your final choices of learning rate (η) and regularization strength (λ) were reasonable. If you reported something slightly different but that still quantifies the performance of the model, please check with us through Piazza to make sure your approach is okay.

Note that since we didn't explicitly require any cross-validation on testing data, there is not much to conclude about λ. By definition, loss on training data with regularization will be higher than without. Good job if you went the extra step to observe its effect on held-out data.

> **Check 4.7**: You plotted the loss during training for the best of these configurations.

Gaussian Generative Models Quantitative Analysis

For the separate covariance case, we report a negative log likelihood of 158.15, and for the shared covariance case, we report a negative log likelihood of 185.35. Note that the definition of likelihood takes the class-prior probabilities into account. We see that the whole data likelihood is higher for the separate covariance case, as expected. Allowing separate covariance matrices fits the data better, and more probability mass is given to the data points we observe, increasing the whole data likelihood.

> **Check 4.8**: You reported the whole data likelihood for the shared and separate covariance cases. This may or may not be negated and/or logged. You reported higher likelihood for the separate covariance model. If you reported something slightly different but that still quantifies the performance of the models, please check with us through Piazza to make sure your approach is okay.

End Solution

Calibration [1pt]

Approximately how long did this homework take you to complete?