

CS195-5: Introduction to Machine Learning
Fall 2006

Problem Set #2 : Nectarine

Out: September 28, 2006

Due: October 11, 2006, 11am EDT

Instructions

In order to identify problem sets to the automated submission system we will assign code names to each problem set. This one will be called “nectarine”.

How and what to submit? Please submit your solutions using the automated system in the following way.

1. Create a directory (say, `ps2`) in which you will work on your solutions. All files mentioned below are assumed to reside in that directory.
2. Typeset the written parts of your solution in \LaTeX . The final results should be a document in PDF (preferable) or in PostScript. please name this document `nectarinesolution.pdf` or `nectarinesolution.ps`, according to the format. **Important: please start the solution of every problem with `\newpage`.**
3. Create Matlab code (extension `.m`) and data (`.mat`) files needed.
4. Create a file named `README`, and include in it a short description of all Matlab files you are submitting.
5. **From the directory `ps2`**, run
`/course/cs195-5/bin/cs195-5handin nectarine`

You can resubmit your work as many times as needed, and each subsequent version will override the previous one.

Late submissions: there will be a penalty of 25 points for any solution submitted past the deadline until 11am on Friday, October 13. No submissions will be accepted past then.

What is the required level of detail? When asked to derive something, please clearly state the assumptions, if any, and strive for balance: justify any non-obvious steps, but try to avoid superfluous explanations. When asked to plot something, please include the figure as well as the code used to plot it; if multiple entities appear on a plot, make sure that they are clearly distinguishable (by color or style of lines and markers). When asked to provide a brief explanation or description, try to make your answers concise, but do not omit anything you believe is important.

1 Decision theory

We have seen in Lecture 7 that the minimal risk for a particular joint distribution $p(\mathbf{x}, y)$ is attained by the Bayes classifier

$$h^*(\mathbf{x}) = \operatorname{argmax}_c p(c | \mathbf{x}).$$

One may suspect that this bound is limited to *deterministic* classifiers. An attempt to “beat” this bound, then, could be based on the following, *randomized* classifier. Define, for any data point \mathbf{x} , a probability distribution $q(c | \mathbf{x})$. In other words, q is some probability distribution over class labels c conditioned on the input \mathbf{x} . The resulting randomized classifier (for which q serves as a parameter), given a data point \mathbf{x} , draws a random class label from q :

$$h_r(\mathbf{x}; q) = c_r, \quad c_r \sim q(c | \mathbf{x}).$$

To express the risk of this classifier we need to take the expectation over all possible outcomes of the random decision:

$$R(h_r; q) = \int_{\mathbf{x}} \sum_{c=1}^C \sum_{c'=1}^C L(c', c) q(c_r = c' | \mathbf{x}) p(\mathbf{x}, y = c) d\mathbf{x}.$$

Problem 1 [12 points]

Show that $R(h_r; q) \geq R(h^*)$, that is, that the risk of the randomized classifier h_r defined above is at least as high as the Bayes risk. Find the probability distribution $q(c_r | \mathbf{x})$ that minimizes the risk of h_r ; express the minimum risk attained in this way in terms of $R(h^*)$.

End of problem 1

Advice: Note that it is enough to show that the inequality holds for the conditional risk, i.e., that $R(h_r|\mathbf{x}) \geq R(h^|\mathbf{x})$ for any \mathbf{x} .*

We have discussed extensively the role that likelihood (class-conditional) plays in determining the optimal decision boundary. Below we will try to get some insight into the effect the class *priors* have on the decision boundary.

Assume that we model the class-conditional density for two classes, $y \in \{\pm 1\}$, in \mathbb{R}^d by Gaussians with equal covariances, and that we believe that the priors are uniform, i.e., $P_{+1} = P_{-1} = 1/2$. Furthermore, we are working under the 0/1 loss model. We have seen in class that the decision rule corresponding to optimal (Bayes) classifier is linear, given by

$$h^*(\mathbf{x}) = \text{sign}(\mathbf{w}^{*T} \mathbf{x} + w_0^*),$$

for certain \mathbf{w}^*, w_0^* .

Problem 2 [8 points]

Suppose now that our estimates for class conditionals remain exactly the same as before but the priors are estimated to be unequal, namely, $\hat{P}_{+1} = 0.7, \hat{P}_{-1} = 0.3$. How will it affect the decision boundary?

End of problem 2

Advice: The answer we are looking for is qualitative, but specific. You need to describe the new decision boundary in terms of \mathbf{w}^, w_0^* and whatever additional term(s) you may need.*

2 Estimation

In this section we will investigate various estimators for the parameter of the Bernoulli distribution. Recall that the probability mass function for a Bernoulli random variable X is

$$p(X; \theta) = \theta^X (1 - \theta)^{1-X}.$$

We will first consider the ML estimator of θ .

We will now consider the following experiment: a coin, distributed according to Bernoulli with parameter θ , is flipped three times. We will encode heads by 1 and tails by 0, so that an observed sequence {heads,tails,tails} will correspond to $x_1 = 1, x_2 = 0, x_3 = 0$.

In order to do the experiments in this section, you will need to simulate a series of 1,000 three-flips experiments. Create a data set of 1,000 3-dimensional binary vectors representing outcomes of coin flips, with probability of heads being 0.7. A quick way to do that in Matlab:

```
theta = 0.7;
r=unifrnd(0,1,3,1000);
coin=zeros(size(r));
coin(r<=theta)=1;
```

It is also useful to become familiar with Matlab's "vectorization" tricks. Many built-in function such as `mean`, `sum` or `prod` can operate on matrices, performing their respective operation along a specified dimension. That dimension is often taken as an optional argument, and often defaults to 1 (rows). Doing this is typically much faster than running a `for` loop. For instance, to calculate the average of each column in a 3×1000 matrix `Mat`, you can simply compute

```
meanM = mean(Mat);
```

and in `meanM` you will have a 1×1000 vector, in which the i -th element contains the average of the i -th column in `Mat`. Another useful operation is element-wise operator, e.g., in order to compute in `res(i)` the product of `a(i)` and `b(i)` you can simply compute

```
v = a.*b;
```

Other useful functions include `repmat` and `reshape`.

Problem 3 [5 points]

Show that the ML estimate of θ given a set of observations x_1, \dots, x_N is in fact $\frac{1}{N} \sum_{i=1}^N x_i$.

End of problem 3

Problem 4 [10 points]

Let θ^* denote the true value of the parameter of the underlying Bernoulli distribution. What is the bias of this ML estimator based on a set of three coin tosses ($N = 3$)? What is its variance? Explain the dependence of the answers on θ^* or lack of such dependence.

Now empirically evaluate the bias and variance of the ML estimator on the 1000-trial data set (i.e., $\theta^* = 0.7$ in this case). Report the results. Also,

report the mean squared error of the estimation of θ (averaged over 1,000 trials).

End of problem 4

Advice: In this case, there are only eight possible training sets of three observations. The expectation of any quantity may therefore be expressed as a sum of a eight terms (some of which are identical!).

We now move on to the Maximum A-Posteriori estimator based on a Beta prior. We will consider four priors:

$$p_1(\theta) = \text{Beta}(\theta; 1, 1) \quad (1)$$

$$p_2(\theta) = \text{Beta}(\theta; 4, 4) \quad (2)$$

$$p_3(\theta) = \text{Beta}(\theta; 20, 20) \quad (3)$$

$$p_4(\theta) = \text{Beta}(\theta; 20, 10) \quad (4)$$

Problem 5 [5 points]

Plot these four priors, and briefly explain in your own words what beliefs about θ they represent, qualitatively, and how they will differ in their effect on the MAP estimator.

End of problem 5

Advice: The Beta pdf is provided in Matlab: `betapdf`. A quick way to compute a values of the prior for a range of valid values of θ :

```
thetas = 0:.001:1;  
betas = betapdf(thetas,a,b);
```

Problem 6 [10 points]

Write down the formula for the bias and variance of the MAP estimator of θ based on a Beta prior with hyperparameters a, b . It is OK to express the answer in terms of Beta functions, without explicitly solving the sums.

Report an empirical estimate for the bias, variance and mean squared error of each of the four MAP estimators arising from the four priors in (1)-(4), on the same data set used in Problems 4-5. Compare the results for different priors to each other and to the results obtained with ML estimator, and briefly discuss the conclusions.

End of problem 6

Problem 7 [5 points]

Evaluate (empirically) and report bias, variance and the mean squared error of the estimator based on the expectation of posterior $p(\theta|X)$. What are advantages and disadvantages of using this estimator?

End of problem 7

3 Logistic regression and softmax

We have started our discussion on classification with a method that naïvely fits a function to the observed labels, i.e., models

$$\hat{y}(c|\mathbf{x}) = \mathbf{w}^T \mathbf{x},$$

where \mathbf{w} is found by minimizing the least-squares criterion on the training data. We will assume in this problem that \mathbf{x} has been augmented with the unit constant term.

This can be extended to non-linear functions in the standard way, by including in \mathbf{x} non-linear terms. For instance, when x is 1D we could fit a quadratic function to the labels by defining $\mathbf{x} = [1 \ x \ x^2]$. When the labels (in a two-category problem) are 0 and 1, the classification rule arising from this model is

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } f(\mathbf{x}; \mathbf{w}) \geq 0.5, \\ 0 & \text{otherwise.} \end{cases}$$

An alternative to the model above is the logistic regression model where we fit a logistic function to the posterior of one of the classes, that is,

$$\hat{p}(c|\mathbf{x}) = \sigma(\mathbf{w}^T \mathbf{x}).$$

The classifier corresponding to this model is the Bayes classifier, that relies on the logistic estimate of the posterior.

We will start with a “toy” data set. In order to fit the parameter vector of logistic regression you will need to fill in missing pieces in `logisticRegression.m` provided to you. Specifically, you will need to write `gradient.m` with a function computing the gradient of the error for logistic regression, and fill in the details of the call to `gradient()` inside `logisticRegression`.

Problem 8 [10 points]

Complete the code as stated above. Load `LogRegToy.mat` (available on the

Web and also under `/course/cs195-5/data/logreg`) which contains the following variables:

`toyXtrain`, `toyXtest` – the training and test examples (1D, i.e. one per column, without the unit terms).

`toyYtrain`, `toyYtest` – the labels, 0 or 1, for the training and test examples.

First, fit linear and quadratic functions directly to the labels in the training data using the least-squares criterion. Report the empirical errors of the two classifiers, and also the error obtained on the test data.

Now fit linear and quadratic logistic regression, using `logisticRegression.m`. Report the training and test errors of these two models.

Plot the two functions found by the least-squares procedure (linear and quadratic), and the functions found by logistic regression (linear and quadratic). Compare these plots, and the errors obtained with the four classifiers corresponding to these functions, and briefly explain the results of this comparison.

End of problem 8

Next we move to a more interesting case. We will classify part of the MNIST data set of handwritten digits (one of the most popular benchmarks in machine learning). You can see some details about the data set, as well as a compendium of results obtained by many classification algorithms, at <http://yann.lecun.com/exdb/mnist>.

Note that those results are reported for the full data set, containing 60,000 training and 10,000 testing examples of all ten digits.

The data set provided to you in `digitData.mat` (on the Web and under `/course/cs195-5/data/logreg`) contains only 10,000 examples from only four digits: 1,2,3 and 4. The digits are stored in columns of `digitX`. These are “vectorized” images (again, without the ones added – you will need to augment them before running regression). You can visualize them in the following way:

```
figure;  
imagesc(reshape(digitX(:,100),28,28));  
colormap gray; axis equal;
```

We will solve four binary classification problems: for each of the four digits, we will try to classify that digit versus all others. I.e., in the first problem examples corresponding to the digit 1 will be labeled as class $y = 1$ and all other examples as class $y = 0$.

Problem 9 [5 points]

Using 10-fold cross-validation, evaluate the error of the linear logistic regression model for each of the four tasks. Which digit appears to be most difficult to classify? Why do you think that is the case?

End of problem 9

A principled multi-class generalization of the logistic regression model is the softmax model. It requires that we maintain a separate parameter vector for each class. The model for class posterior for class c , $c = 1, \dots, C$ becomes

$$\hat{p}(c|\mathbf{x}) = \frac{\exp(\mathbf{w}_c^T \mathbf{x})}{\sum_{y=1}^C \exp(\mathbf{w}_y^T \mathbf{x})}.$$

Problem 10 [5 points]

Show that for the case of $C = 2$ the softmax model is equivalent to the logistic regression model. That is, show that for any two $d + 1$ -dimensional parameter vectors \mathbf{w}_1 and \mathbf{w}_2 in the softmax model, there exist a single $d + 1$ -dimensional parameter vector \mathbf{v} such that

$$\frac{\exp(\mathbf{w}_1^T \mathbf{x})}{\exp(\mathbf{w}_1^T \mathbf{x}) + \exp(\mathbf{w}_2^T \mathbf{x})} = \sigma(\mathbf{v}^T \mathbf{x}).$$

End of problem 10

4 Naïve Bayes and text classification

In this problem you will implement a Naïve Bayes (NB) classifier for e-mail messages based on word occurrence (modeled as a Bernoulli random variable). You will find it useful to recall the material in Lectures 9 and 10 before you work on this problem.

We will be using a public data set made available by SpamAssassin¹. The e-mail messages used for training and testing the classifier reside under `/course/cs195-5/data/spam` in the following subdirectories:

- `trainSpam` – training data for class $y = 1$ (SPAM);
- `trainHam1` – training set #1 for $y = 0$ (legitimate e-mail);
- `trainHam2` – training set #2 for $y = 0$ (legitimate e-mail);

¹<http://spamassassin.apache.org/publiccorpus>

`testSpam` – test examples from class $y = 1$;

`testHam` – test examples from class $y = 0$.

Each data file with extension `.eml` contains a single e-mail message (including the header). Feel free to browse through the files but note that the content, in particular that of the SPAM messages, may be offensive.

Your NB classifier will be using binary features based on occurrence of a set of keywords. The keywords are provided to you in the file `dictionary.mat` in the form of a cell array `dictionary`. Please stick with this set of keywords (and the features they define). The j -th binary feature is defined as follows:

$$\phi_j(\mathbf{x}) = \begin{cases} 1 & \text{if } \text{dictionary}\{j\} \text{ appears in } \mathbf{x} \text{ at least once,} \\ 0 & \text{otherwise.} \end{cases}$$

We also provide you with two Matlab functions that deal with reading in e-mail messages and extracting *counts* of the words in the dictionary (case insensitive) for each e-mail. These are `parseDirectory.m` and `parseEmail.m`. Please read the comments in those files. The code is available on the Web and also under `/course/cs195-5/matlab/spam`.

You will need to write (and turn in) the code for computing the binary features, estimating the parameters of Bernoulli distributions, and of course the NB classifier. Assume that the priors are equal for the two classes.

Problem 11 [10 points]

First, use only data from `trainSpam` and `trainHam1` for training (i.e. for estimating the parameters). Construct NB classifier using the maximum likelihood estimates for Bernoulli parameters. What is the classification error on the test data? Report both the total error and class-specific error on each class, i.e. the percentage of SPAM e-mails that got classified as legitimate and the percentage of legitimate e-mail that your classifier flagged as SPAM.

Now repeat the experiment, this time using the data from `trainHam2` instead of `trainHam1`. What are the test errors (total/SPAM/non-SPAM) now? Try to explain the difference between the performance of the two classifiers trained on different sets of non-SPAM examples.

End of problem 11

Advice: You may find the Matlab functions useful: `ismember` and/or `strmatch`.

Problem 12 [8 points]

In practice, the cost of misclassifying a valid e-mail is much higher than

making the opposite mistake. In other words, it is much worse to delete a legitimate e-mail by mistake than to let a few instances of SPAM slip through. To be concrete, let us say that the loss incurred by failing to detect a SPAM message is 1, and the loss incurred by mistakenly flagging a non-SPAM is 3.

Explain how you would change the NB classifier obtained in the previous problem to take this into account. Implement the change you propose, and report the new test errors (total/SPAM/non-SPAM), as well as the average loss on the test set.

End of problem 12

Advice: Think about the derivation of Bayes classifier under 0/1 loss, and how it would change under an asymmetric loss we are considering here.

Problem 13 [7 points]

Suppose now that you are limited to only ten features. That is, you need to pick out ten words from the given dictionary, and only the features corresponding to those words will be used in the Naïve Bayes classifier. Propose a way of selecting the ten features, list the features (words) selected using that method and report the test error of the NB classifier based on those features (use `trainSpam` and `trainHam2` for training).

End of problem 13

Problem 14 [10 points]

Extra credit Instead of the binary features we could use a more refined representation that encodes *relative frequencies* of the words. Let N_i be the number of times the i -th keyword appears in the e-mail \mathbf{x} , and let $N = \sum_i N_i$. Then, we can define

$$\phi'_j(\mathbf{x}) \triangleq N_i/N.$$

Propose a distribution model for such features, and a ML estimator for the parameter(s) of such a distribution. Build the Naïve Bayes classifier using these features and report its performance on the test set (under 0/1 loss). What can you say about the validity of the NB assumptions for the relative frequency features?

End of problem 14