CS 195-5: Machine Learning
Problem Set 2

Douglas Lanman
dlanman@brown.edu
11 October 2006

1 Decision Theory

Problem 1

Part 1: In this problem we will examine randomized classifiers. Define, for any point $\mathbf{x}$, a probabilty distribution $q(c|\mathbf{x})$ over the class labels c. We define a randomized decision rule given by

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

The risk $R(h_r; q)$ for this classifier is given by the expectation over all possible outcomes such that

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

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 $R(h^*)$.

Recall from class on 9/20/06 that the risk $R(h^*)$ of a C-way Bayesian classifier is given by

$$R(h^*) = \int_{\mathbf{x}} \sum_{c=1}^{C} L(h^*(\mathbf{x}), c) p(\mathbf{x}, y = c) d\mathbf{x}. \tag{2}$$

Equations 1 and 2 give the following form for the inequality $R(h_r; q) \geq R(h^*)$.

$$\int_{\mathbf{x}} \sum_{c=1}^{C} \sum_{c'=1}^{C} L(c', c) q(c_r = c'|\mathbf{x}) p(\mathbf{x}, y = c) d\mathbf{x} \geq \int_{\mathbf{x}} \sum_{c=1}^{C} L(h^*(\mathbf{x}), c) p(\mathbf{x}, y = c) d\mathbf{x}$$

Applying Bayes rule and grouping terms within the integrals provides the following inequality.

$$\int_{\mathbf{x}} \left[\sum_{c=1}^{C} \sum_{c'=1}^{C} L(c', c) q(c_r = c'|\mathbf{x}) p(y = c|\mathbf{x}) \right] p(\mathbf{x}) d\mathbf{x} \geq \int_{\mathbf{x}} \left[\sum_{c=1}^{C} L(h^*(\mathbf{x}), c) p(y = c|\mathbf{x}) \right] p(\mathbf{x}) d\mathbf{x}$$

Note that it is sufficient to show that this inequality holds for the conditional risks such that $R(h_r|\mathbf{x}) \geq R(h^*|\mathbf{x})$ for any $\mathbf{x}$. As a result, we can restate the previous inequality as follows.

$$\sum_{c=1}^{C} \sum_{c'=1}^{C} L(c', c) q(c_r = c'|\mathbf{x}) p(y = c|\mathbf{x}) \geq \sum_{c=1}^{C} L(h^*(\mathbf{x}), c) p(y = c|\mathbf{x}) \tag{3}$$

To prove this inequality, it is sufficient to show that each term in the summation satisfies the inequality independently such that

$$\sum_{c'=1}^{C} L(c', c) q(c_r = c'|\mathbf{x}) p(y = c|\mathbf{x}) \geq L(h^*(\mathbf{x}), c) p(y = c|\mathbf{x}),$$

holds for all c. Canceling the common factors of $p(y = c|\mathbf{x})$, we obtain the following result.

$$\sum_{c'=1}^{C} L(c', c)q(c_r = c'|\mathbf{x}) \geq L(h^*(\mathbf{x}), c)$$

Note that we can separate the summation on the right-hand side as follows.

$$L(h^*(\mathbf{x}), c)q(c_r = h^*(\mathbf{x})|\mathbf{x}) + \sum_{c' \neq h^*(\mathbf{x})}^{C} L(c', c)q(c_r = c'|\mathbf{x}) \geq L(h^*(\mathbf{x}), c)$$

$$\Rightarrow \sum_{c' \neq h^*(\mathbf{x})}^{C} L(c', c)q(c_r = c'|\mathbf{x}) \geq L(h^*(\mathbf{x}), c)\left(1 - q(c_r = h^*(\mathbf{x})|\mathbf{x})\right) \tag{4}$$

Since $q(c|\mathbf{x})$ must be a valid probability distribution, the following relationship must hold.

$$\sum_{c'=1}^{C} q(c_r = c'|\mathbf{x}) = 1 \quad \Rightarrow \quad 1 - q(c_r = h^*(\mathbf{x})|\mathbf{x}) = \sum_{c' \neq h^*(\mathbf{x})}^{C} q(c_r = c'|\mathbf{x}) \tag{5}$$

Substituting Equation 5 into 4 gives

$$\sum_{c' \neq h^*(\mathbf{x})}^{C} L(c', c)q(c_r = c'|\mathbf{x}) \geq \sum_{c' \neq h^*(\mathbf{x})}^{C} L(h^*(\mathbf{x}), c)q(c_r = c'|\mathbf{x}).$$

In order for this inequality to hold it is sufficient for

$$L(c', c) \geq L(h^*(\mathbf{x}), c), \tag{6}$$

for all $c' \neq h^*(\mathbf{x})$. In general, $L(h^*(\mathbf{x}), c)$ will be the minimum loss attributed to the decision of the optimum Bayes classifier. As a result, any valid loss function must assign a cost for misclassifying a sample that is greater than or equal to this value in order to minimize the conditional risk; this is precisely the requirement stated in Equation 6. As a result, the risk of the randomized classifier is at least as high as the Bayes risk such that $R(h_r; q) \geq R(h^*)$. (QED)

Part 2: Find the probability distribution $q(c|\mathbf{x})$ that minimizes the risk of h_r and express the corresponding minimum risk in terms of $R(h^*)$.

Since $R(h_r; q) \geq R(h^*)$, the probability distribution $q(c|\mathbf{x})$ that minimizes $R(h_r; q)$ could achieve the Bayes risk in the best case. From inspection of Equation 1, we find $q(c|\mathbf{x})$ must be given by

$$\boxed{q(c|\mathbf{x}) = \delta(c, h^*(\mathbf{x})) = \begin{cases} 1 & \text{for } c = h^*(\mathbf{x}) \\ 0 & \text{for } c \neq h^*(\mathbf{x}), \end{cases}},$$

where $\delta(c, h^*(\mathbf{x}))$ is the Kronecker Delta. Substituting this expression into Equation 1 gives

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

In conclusion, the minimum risk $R(h_r; q) = R(h^*)$ and it is achieved with the Bayes classifier where $q(c|\mathbf{x}) = \delta(c, h^*(\mathbf{x}))$. In other words, the randomized decision rule cannot perform better than the Bayes classifier.

Problem 2

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

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

for certain $\mathbf{w}^*$ and w_0^*. Suppose that the priors are instead given by $\hat{P}_{+1} = 0.7$ and $\hat{P}_{-1} = 0.3$. How will this affect the decision boundary?

Recall from class on 9/20/06 that, in the case of two classes, the Bayes classifier is

$$h^*(\mathbf{x}) = \underset{c=\pm 1}{\text{argmax}}\, \delta_c(\mathbf{x}) = \text{sign}\left(\delta_{+1}(\mathbf{x}) - \delta_{-1}(\mathbf{x})\right),$$

where $\delta_c(\mathbf{x}) \triangleq \log p_c(\mathbf{x}) + \log P_c$ is the *discriminant function*. Substituting for the discriminant function we obtain the following expression for the optimal classifier.

$$\begin{aligned} h^*(\mathbf{x}) &= \text{sign}\left(\log p_{+1}(\mathbf{x}) + \log P_{+1} - \log p_{-1}(\mathbf{x}) - \log P_{-1}\right) \\ &= \text{sign}\left(\log p_{+1}(\mathbf{x}) - \log p_{-1}(\mathbf{x}) + \log\left(\frac{P_{+1}}{P_{-1}}\right)\right) \end{aligned} \tag{7}$$

At this point, we recall the result of Problem 9 in Problem Set 1. In that problem we proved that the optimal decision rule was based on calculating a set of C linear discriminant functions

$$\delta_c(\mathbf{x}) = \log p_c(\mathbf{x}) = \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mu_c - \frac{1}{2}\mu_c^T\boldsymbol{\Sigma}^{-1}\mu_c, \tag{8}$$

when the class-conditionals were assumed to be multivariate Gaussians in $\mathbb{R}^d$ with equal covariance matrices and the prior probabilities were uniform such that $P_c = 1/C$. As we previously shown, under the assumption of uniform priors, the discriminant function $\delta_c(\mathbf{x}) = \log p_c(\mathbf{x})$. As a result, we can substitute Equation 8 into Equation 7 to obtain the solution for the Bayes classifier under arbitrary priors given by

$$\begin{aligned} h^*(\mathbf{x}) &= \text{sign}\left(\mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mu_{+1} - \frac{1}{2}\mu_{+1}^T\boldsymbol{\Sigma}^{-1}\mu_{+1} - \mathbf{x}^T\boldsymbol{\Sigma}^{-1}\mu_{-1} + \frac{1}{2}\mu_{-1}^T\boldsymbol{\Sigma}^{-1}\mu_{-1} + \log\left(\frac{P_{+1}}{P_{-1}}\right)\right) \\ &= \text{sign}\left(\left(\boldsymbol{\Sigma}^{-1}\mu_{+1} - \boldsymbol{\Sigma}^{-1}\mu_{-1}\right)^T\mathbf{x} + \left(\frac{1}{2}\mu_{-1}^T\boldsymbol{\Sigma}^{-1}\mu_{-1} - \frac{1}{2}\mu_{+1}^T\boldsymbol{\Sigma}^{-1}\mu_{+1}\right) + \log\left(\frac{P_{+1}}{P_{-1}}\right)\right) \\ &= \text{sign}\left(\mathbf{w}^{*T}\mathbf{x} + w_0^* + \log\left(\frac{P_{+1}}{P_{-1}}\right)\right), \end{aligned}$$

for $\mathbf{w}^* = \boldsymbol{\Sigma}^{-1}\mu_{+1} - \boldsymbol{\Sigma}^{-1}\mu_{-1}$ and $w_0^* = \frac{1}{2}\mu_{-1}^T\boldsymbol{\Sigma}^{-1}\mu_{-1} - \frac{1}{2}\mu_{+1}^T\boldsymbol{\Sigma}^{-1}$. In conclusion, we find that the optimal (Bayes) decision rule is given by the following expression.

$$\boxed{h^*(\mathbf{x}) = \text{sign}\left(\mathbf{w}^{*T}\mathbf{x} + w_0^* + \log\left(\frac{P_{+1}}{P_{-1}}\right)\right)}$$

For $\hat{P}_{+1} = 0.7$ and $\hat{P}_{-1} = 0.3$, $\log\left(\frac{P_{+1}}{P_{-1}}\right) = 0.8473$. As a result, we find that the decision boundary shifts towards class -1 (i.e., away from class $+1$). This is expected, since if the class-conditions are equal we'll select the class with the greater prior probability.

2 Estimation

Problem 3

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

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

Show that the ML estimator of θ, given a set of observations $\mathbf{X} = \{x_1, \ldots, x_N\}$, is $\hat{\theta}_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i$.

Assuming that the observations are independent and identically distributed (i.i.d.), then the likelihood $\mathcal{P}$ can be expressed as the following product.

$$\mathcal{P}(\mathbf{X};\theta) = \prod_{i=1}^{N} p(x_i|\theta) = \prod_{i=1}^{N} \theta^{x_i}(1-\theta)^{1-x_i}.$$

The log-likelihood ℓ is then given by

$$\begin{aligned}
\ell(\mathbf{X};\theta) = \log \mathcal{P}(\mathbf{X};\theta) &= \sum_{i=1}^{N} \log p(\mathbf{x}_i|\theta) = \sum_{i=1}^{N} \log\left(\theta^{x_i}(1-\theta)^{1-x_i}\right) \\
&= \sum_{i=1}^{N} \log\left(\theta^{x_i}\right) + \sum_{i=1}^{N} \log\left((1-\theta)^{1-x_i}\right) \\
&= \log\theta \sum_{i=1}^{N} x_i + \sum_{i=1}^{N} \log(1-\theta) - \log(1-\theta)\sum_{i=1}^{N} x_i \\
&= (\log\theta - \log(1-\theta))\sum_{i=1}^{N} x_i + N\log(1-\theta).
\end{aligned}$$

The corresponding ML estimate for θ is

$$\hat{\theta}_{ML} = \underset{\theta}{\operatorname{argmax}}\, \ell(\mathbf{X};\theta) = \underset{\theta}{\operatorname{argmax}} \left\{ (\log\theta - \log(1-\theta))\sum_{i=1}^{N} x_i + N\log(1-\theta) \right\}$$

The maximum (or minimum) of this function will necessarily be obtained where the derivative with respect to θ equals zero.

$$\begin{aligned}
&\frac{\partial}{\partial\theta}\left\{ (\log\theta - \log(1-\theta))\sum_{i=1}^{N} x_i + N\log(1-\theta) \right\} = 0 \\
&\Rightarrow \left(\frac{1}{\theta} + \frac{1}{1-\theta}\right)\sum_{i=1}^{N} x_i = \frac{1}{\theta(1-\theta)}\sum_{i=1}^{N} x_i = \frac{N}{1-\theta}
\end{aligned}$$

Simplifying this expression we prove the desired result.

$$\boxed{\hat{\theta}_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i} \tag{9}$$

(QED)

Problem 4

Part 1: Let θ^* denote the true value of the parameter of the underlying Bernoulli distribution. What are the bias and variance of the ML estimator based on a set of three coin tosses (i.e., N = 3)? Explain the dependence, or lack thereof, of the answers on θ^*.

In the following analysis let's define $\hat{\theta}_N$ as the ML estimate of the Bernoulli parameter using N trials. Recall from class on 9/22/06 that the bias and variance are given by

$$\text{bias}(\hat{\theta}_N) \triangleq E[\hat{\theta}_N - \theta^*] \quad \text{and} \quad \text{var}(\hat{\theta}_N) \triangleq E[(\hat{\theta}_N - \bar{\theta}_N)^2], \tag{10}$$

where $\bar{\theta}_N \triangleq E(\hat{\theta}_N)$ and the expectation is over all possible sample sequences of length N. As described in class on 9/25/06, the expectation for a discrete probability distribution is given by the following expression.

$$E[f(x)] = \sum_{v \in \mathcal{X}} f(v)p(x = v), \text{ where } \sum_{v \in \mathcal{X}} p(x = v) = 1 \tag{11}$$

From Problem 3 we know that the ML estimator is given by $\hat{\theta}_N = \frac{1}{N}\sum_{i=1}^{N} x_i$. Let $x_i = 1$ denote heads and $x_i = 0$ denote tails. Using this formulation, we find that the ML estimator reduces to $\hat{\theta}_N = m/N$, where m is the total number of heads in a sequence of length N. Combining Equations 10 and 11, we obtain

$$\text{bias}(\hat{\theta}_N) = E[\hat{\theta}_N] - \theta^* = \left\{ \sum_{m=0}^{N} \frac{m}{N} p(m \text{ heads}) \right\} - \theta^* \tag{12}$$

$$\text{var}(\hat{\theta}_N) = E[\hat{\theta}_N^2] - E[\theta_N]^2 = \left\{ \sum_{m=0}^{N} \left(\frac{m}{N}\right)^2 p(m \text{ heads}) \right\} - \left\{ \sum_{m=0}^{N} \frac{m}{N} p(m \text{ heads}) \right\}^2, \tag{13}$$

where $p(m \text{ heads})$ is the probability of obtaining m heads in N trials. In a sequence of three coin flips, there are only eight possible outcomes: $\{(000), (100), (010), (001), (110), (101), (011), (111)\}$. If the probability of heads is θ^*, then the probability of tails is $1 - \theta^*$. As a result, we obtain the following probabilities for the total number of heads in a set of three coin flips.

Number of Heads	Sequences	Probability
0	(000)	$(1-\theta^*)^3$
1	(100), (010), (001)	$3\theta^*(1-\theta^*)^2$
2	(110), (101), (011)	$3(\theta^*)^2(1-\theta^*)^1$
3	(111)	$(\theta^*)^3$

Using the probabilities tabulated above, we can substitute into Equation 12 to obtain the following expression for the bias.

$$\text{bias}(\hat{\theta}_3) = \left\{\theta^*(1-\theta^*)^2 + 2(\theta^*)^2(1-\theta^*)^1 + (\theta^*)^3\right\} - \theta^*$$

$$\Rightarrow \boxed{\text{bias}(\hat{\theta}_3) = 0}$$

Similarly, the variance is given by substitution into Equation 13.

$$\text{var}(\hat{\theta}_3) = \left\{ \frac{1}{3}\theta^*(1-\theta^*)^2 + \frac{4}{3}(\theta^*)^2(1-\theta^*)^1 + (\theta^*)^3 \right\} - \left\{\theta^*(1-\theta^*)^2 + 2(\theta^*)^2(1-\theta^*)^1 + (\theta^*)^3\right\}^2$$

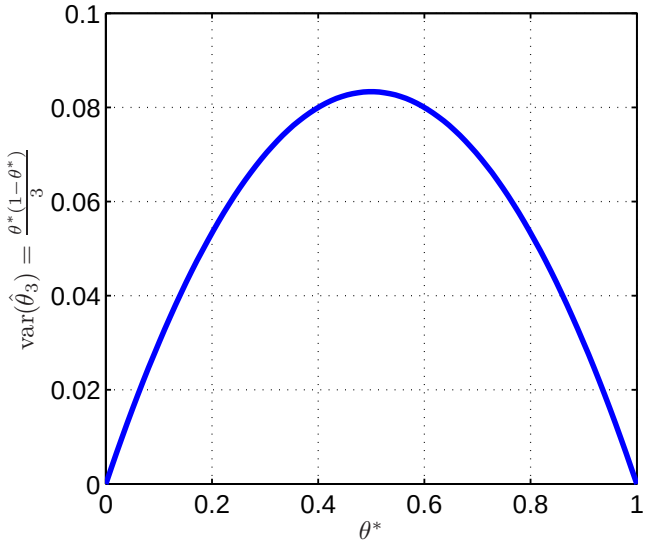


Figure 1: Dependence of $\mathrm{var}(\hat{\theta}_3)$ on θ^*.

$$\Rightarrow \boxed{\mathrm{var}(\hat{\theta}_3) = \frac{\theta^*(1-\theta^*)}{3}}$$

In conclusion, we find that the ML estimator is unbiased and, as a result, the bias is independent of θ^*. The variance, however, is a quadratic function of θ^*. In general, the variance will only be independent of θ^* when $\theta^* = \{0, 1\}$. In these cases the coin flips are deterministic (i.e., they always come up on the same side) and, as a result, the variance is zero. For all other values of θ^* the variance of the ML estimator will have the dependence on θ^* shown in Figure 1. (Note that these results are consistent with the statement made in class on 9/25/06; ML estimators tend to be unbiased, but have comparatively high variance.)

Part 2: Empirically evaluate the bias and variance of the ML estimator on a 1000-trial data set for $\theta^* = 0.7$. Also report the mean squared error for the estimation of θ (averaged over 1,000 trials).

A series of 1,000 three-flip Bernoulli trials were simulated using `prob4.m`. Note that the results reported here depend on the value of the random number seed specified on line 17. The empirical bias and variance were estimated using the following formulas.

$$\text{empirical-bias}(\hat{\theta}_3) = \left\{ \frac{1}{1000} \sum_{i=1}^{1000} (\hat{\theta}_3)_i \right\} - \theta^* \quad \text{and} \quad \text{empirical-var}(\hat{\theta}_3) = \frac{1}{1000} \sum_{i=1}^{1000} ((\hat{\theta}_3)_i - \bar{\theta}_3)^2$$

For the selected random number seed, we report the following results for $\theta^* = 0.7$.

	Predicted	Empirical
Bias	0	0.0143
Variance	0.07	0.0692
MSE	0.07	0.0694

First, note that the predictions are accurate. In addition, note that the mean squared error (MSE) satisfies the bias-variance decomposition described in class on 9/25/06. Specifically, the MSE is given by the following expression.

$$MSE(\hat{\theta}_3) = E[(\hat{\theta}_3 - \theta^*)^2] = \mathrm{bias}^2(\hat{\theta}_3) + \mathrm{var}(\hat{\theta}_3)$$

Problem 5

We now consider the maximum a posteriori (MAP) estimator for the Bernoulli distribution based on a Beta prior. We consider the following four priors: $\{p_1(\theta) = \text{Beta}(\theta; 1, 1), p_2(\theta) = \text{Beta}(\theta; 4, 4), p_3(\theta) = \text{Beta}(\theta; 20, 20), p_4(\theta) = \text{Beta}(\theta; 20, 10)\}$. Plot the priors and explain how they will effect the MAP estimator.

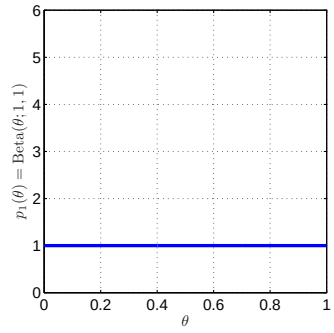
Recall that the Beta distribution, defined by hyperparameters a and b, is given by

$$\text{Beta}(\theta; a, b) \triangleq \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}.$$

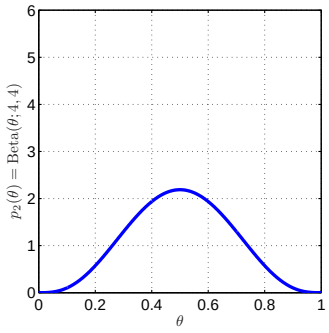
Using the Matlab function `betapdf`, the four priors were plotted in `prob5.m` and are shown in Figure 2. From class on 9/27/06 we know that the Beta distribution is conjugate for the Bernoulli; for a set of N consecutive coin flips X_N, if N_1 are heads and N_0 tails, then we have $p(\theta|X_N) \propto \theta^{N_1+a-1}(1-\theta)^{N_0+b-1}$. In other words, we can consider a and b as the effective number of prior heads and tails, respectively. In general, the default value $\theta_0 = a/(a+b)$ and $a+b$ indicates the confidence in this value. This gives the following prior believes under the four Beta distributions.

Beta Prior	Initial θ	Confidence
$p_1(\theta) = \text{Beta}(\theta; 1, 1)$	$1/2$	2
$p_2(\theta) = \text{Beta}(\theta; 4, 4)$	$1/2$	8
$p_3(\theta) = \text{Beta}(\theta; 20, 20)$	$1/2$	40
$p_4(\theta) = \text{Beta}(\theta; 20, 10)$	$2/3$	30

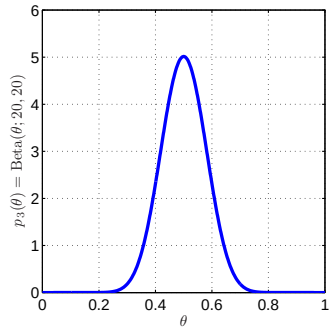
Note that the values in the table agree with the plots in Figure 2. Specifically, $\{p_1(\theta), p_2(\theta), p_3(\theta)\}$ are all consistent with a prior belief that $\theta = 1/2$, but represent increasing confidence in that belief. Alternatively, $p_4(\theta)$ represents a prior belief that $\theta = 2/3$ with slightly less confidence than $p_3(\theta)$. For all priors, increasing confidence means the MAP estimator will require more observations to overcome an incorrect prior.



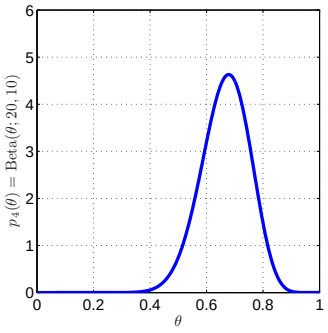
(a) $p_1(\theta) = \text{Beta}(\theta; 1, 1)$



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



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



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

Figure 2: Comparison of Beta priors for MAP estimation in Problems 5 and 6.

Problem 6

Part 1: Write down formulas for the bias and variance of the MAP estimator of θ based on a Beta prior.

Let's begin by deriving the MAP estimator for the Bernoulli distribution under a Beta prior. Recall from class on 9/27/06 that the maximum a posteriori (MAP) estimator satisfies the following condition.

$$\hat{\theta}_{MAP}(X) = \underset{\theta}{\operatorname{argmax}}\, p(\theta|X) = \underset{\theta}{\operatorname{argmax}}\, p(X|\theta)p(\theta) \tag{14}$$

In this problem, X represents a sample composed of N consecutive coin flips of which N_1 are heads and N_0 are tails. Let $p(X|\theta) \sim \text{Bin}(\theta; N_0, N_1)$ and $p(\theta) \sim \text{Beta}(\theta; a, b)$. Substituting into Equation 14, we obtain

$$\begin{aligned}\hat{\theta}_{MAP}(X) &= \underset{\theta}{\operatorname{argmax}} \left\{ \binom{N_0+N_1}{N_1} \left(\theta^{N_1}(1-\theta)^{N_0}\right) \left(\frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)}\theta^{a-1}(1-\theta)^{b-1}\right) \right\} \\ &= \underset{\theta}{\operatorname{argmax}} \left\{ \theta^{N_1+a-1}(1-\theta)^{N_0+b-1} \right\} \\ &= \underset{\theta}{\operatorname{argmax}} \left\{ (N_1+a-1)\log(\theta) + (N_0+b-1)\log(1-\theta) \right\},\end{aligned}$$

where the last expression is obtained by taking the logarithm of the previous equation. The maximum (or minimum) of this function will necessarily be obtained where the derivative with respect to θ equals zero.

$$\frac{\partial}{\partial\theta}\left\{ (N_1+a-1)\log(\theta) + (N_0+b-1)\log(1-\theta) \right\} = 0$$

$$\Rightarrow \frac{N_1+a-1}{\theta} = \frac{N_0+b-1}{1-\theta}$$

Simplifying this expression gives the solution for the MAP estimator for the Bernoulli distribution using a Beta prior.

$$\hat{\theta}_{MAP}(X) = \frac{N_1+a-1}{N+a+b-2} \tag{15}$$

Note that for $a = b = 1$ the MAP estimator is equivalent to the ML estimator given by Equation 9. As a result, we expect the bias, variance, and mean squared error under the Beta prior $p_1(\theta) = \text{Beta}(\theta; 1, 1)$ to be identical to Problem 4 – this observation is confirmed in the second part of this problem.

At this point we can obtain closed-form expressions for the bias and variance by substituting the MAP estimator into Equation 10 such that

$$\text{bias}(\hat{\theta}_N) = \left\{ \sum_{m=0}^{N} \left(\frac{m+a-1}{N+a+b-2} \right) p(m \text{ heads}) \right\} - \theta^* \tag{16}$$

$$\text{var}(\hat{\theta}_N) = \left\{ \sum_{m=0}^{N} \left(\frac{m+a-1}{N+a+b-2} \right)^2 p(m \text{ heads}) \right\} - \left\{ \sum_{m=0}^{N} \left(\frac{m+a-1}{N+a+b-2} \right) p(m \text{ heads}) \right\}^2, \tag{17}$$

where $p(m \text{ heads})$ is the probability of obtaining m heads in N trials. Using the probabilities from the table in Problem 4, we can simplify these expressions to obtain the following equations for the bias and variance of the MAP estimator on a set of three coin tosses. First, the bias is given by

$$\boxed{\text{bias}(\hat{\theta}_3) = \frac{(a-1) - (a+b-2)\,\theta^*}{a+b+1}}. \tag{18}$$

Similarly, the variance is given by

$$\boxed{\text{var}(\hat{\theta}_3) = \frac{3\,\theta^*(1-\theta^*)}{(a+b+1)^2}}. \tag{19}$$

Part 2: Report the empirical estimate for the bias, variance, and mean squared error of each of the four MAP estimators from Problem 5 on the data in Problem 4. Compare the results to the ML estimator.

A series of 1,000 three-flip Bernoulli trials were simulated using `prob6.m` (with the same random number seed as in `prob4.m`). The empirical bias and variance were estimated using the MAP estimator in Equation 15 and the procedure in Problem 4.2. For the selected random number seed, we report the following results for $\theta^* = 0.7$. (Note that the predicted values are shown in parentheses.)

Beta Prior	Bias	Variance	MSE
$p_1(\theta) = \text{Beta}(\theta; 1, 1)$	0.0143 (0)	6.9172e-2 (7.0000e-2)	6.9378e-2 (7.0000e-2)
$p_2(\theta) = \text{Beta}(\theta; 4, 4)$	-0.1286 (-0.1333)	7.6858e-3 (7.7778e-3)	2.4212e-2 (2.5556e-2)
$p_3(\theta) = \text{Beta}(\theta; 20, 20)$	-0.1843 (-0.1854)	3.7035e-4 (3.7478e-4)	3.4343e-2 (3.4735e-2)
$p_4(\theta) = \text{Beta}(\theta; 20, 10)$	-0.0180 (-0.0194)	6.4782e-4 (6.5557e-4)	9.7066e-4 (1.0302e-3)

First, note that the predictions are accurate. Once again, we note that the mean squared error (MSE) satisfies the bias-variance decomposition described in class on 9/25/06. In addition, we can confirm our previous claim that the MAP estimator under the Beta prior $p_1(\theta) = \text{Beta}(\theta; 1, 1)$ is equivalent to the ML estimator given by Equation 9; that is, the bias, variance, and mean squared error for $a = b = 1$ are identical to those reported in Problem 4.

Overall, we find that the MAP estimator is biased (except for the case $a = b = 1$). For large a and b, Equation 18 has the approximate form

$$\lim_{a \to \infty, b \to \infty} \text{bias}(\hat{\theta}_3) \approx \frac{a}{a+b} - \theta^*.$$

In Problem 5 we found that the default value $\theta_0 = a/(a+b)$; as a result, we find that if our initial belief (as expressed by the Beta prior) is accurate, then the MAP estimator will be nearly unbiased. This is verified by the empirical values, since the prior $p_4(\theta)$ leads to a negligible value for the estimated bias since it predicted that θ would be close to 2/3.

Finally, we note that the variance given by Equation 19 is monotonically decreasing in the "confidence" $a+b$. As a result, we find that the empirical variances decrease for increasing confidence values. We conclude that the MAP estimator has a better bias-variance trade-off for the sample data set. Under all four priors, the MAP estimator does no worse (in terms of mean squared error) than the ML estimator – in most cases it performs significantly better!

Problem 7

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

Recall from class on 9/27/06 that an alternative estimator for the Bernoulli parameter can be obtained by estimating the expectation according to the posterior.

$$\hat{\theta}_{Exp}(X) = E_{\theta \sim p(\theta|X)}[\theta|X] = \int_0^1 \theta p(\theta|X) d\theta$$

Applying Bayes rule to this equation provides the following expression for the "Exp" estimator in terms of the parametric model $p(X|\theta)$ and prior $p(\theta)$.

$$\hat{\theta}_{Exp}(X) = \int_0^1 \theta \left(\frac{p(X|\theta)p(\theta)}{p(X)} \right) d\theta = \frac{\int_0^1 \theta p(X|\theta)p(\theta)d\theta}{\int_0^1 p(X|\theta)p(\theta)d\theta} \tag{20}$$

As in Problem 6, X represents a sample composed of N consecutive coin flips of which N_1 are heads and N_0 are tails. Let $p(X|\theta) \sim \text{Bin}(\theta; N_0, N_1)$ and $p(\theta) \sim \text{Beta}(\theta; a, b)$. Substituting into Equation 20, we obtain

$$\hat{\theta}_{Exp}(X) = \frac{\int_0^1 \theta \left(\theta^{N_1+a-1}(1-\theta)^{N_0+b-1} \right) d\theta}{\int_0^1 \theta^{N_1+a-1}(1-\theta)^{N_0+b-1} d\theta}.$$

While not required for this problem, this integral can be solved in closed-form. Using the Mathematica notebook `prob7.nb`, I obtained the following solution for the "Exp" estimator.

$$\hat{\theta}_{Exp}(X) = \frac{N_1 + a}{N + a + b} \tag{21}$$

Note that, for the case of the Bernoulli distribution under a Beta prior, the MAP and "Exp" estimates are similar but not equivalent. Following the procedure for estimating the bias and variance in Problems 4 and 6, we obtained the following expressions with `prob7.nb`.

$$\text{bias}(\hat{\theta}_3) = \frac{a - (a+b)\,\theta^*}{a+b+3} \quad \text{and} \quad \text{var}(\hat{\theta}_3) = \frac{3\,\theta^*(1-\theta^*)}{(a+b+3)^2}$$

The empirical bias and variance were estimated with `prob7.m` using a numerical approximation to the "Exp" estimator in Equation 21 and the procedure in Problem 4.2. For the selected random number seed, we report the following results for $\theta^* = 0.7$. (Note that the predicted values are shown in parentheses.)

Beta Prior	Bias	Variance	MSE
$p_1(\theta) = \text{Beta}(\theta; 1, 1)$	-0.0713 (-0.0800)	2.4961e-2 (2.5200e-2)	3.0039e-2 (3.1600e-2)
$p_2(\theta) = \text{Beta}(\theta; 4, 4)$	-0.1415 (-0.1455)	5.1450e-3 (5.2066e-3)	2.5180e-2 (2.6364e-2)
$p_3(\theta) = \text{Beta}(\theta; 20, 20)$	-0.1850 (-0.1860)	3.3670e-4 (3.4072e-4)	3.4579e-2 (3.4954e-2)
$p_4(\theta) = \text{Beta}(\theta; 20, 10)$	-0.0290 (-0.0303)	5.7167e-4 (5.7851e-4)	1.4127e-3 (1.4968e-3)

First, note that the predictions are accurate. Once again, we note that the mean squared error (MSE) satisfies the bias-variance decomposition described in class on 9/25/06. Overall, we find that the "Exp" estimator has similar, but not identical, performance to the MAP estimator. This is expected, given the similarity of their closed-form solutions. From the experimental data it is apparent that the "Exp" estimator has greater bias than the MAP estimator, but has also has lower variance. In conclusion, we find that the "Exp" estimator is only useful when minimum variance is required; in all other cases, the MAP estimator is preferred due to its simple form which avoids the necessity of evaluating integrals numerically.

3 Logistic Regression and Softmax

Problem 8

Part 1: A naïve classification results from applying least-squares regression to the observed labels such that $\hat{y}(c|\mathbf{x}) = \hat{\mathbf{w}}^T\mathbf{x}$. When the class labels are 0 and 1, the corresponding classification rule is given by

$$h(\mathbf{x}) = \begin{cases} 1 & \text{if } f(\mathbf{x};\mathbf{w}) \geq 1/2, \\ 0 & \text{otherwise.} \end{cases} \tag{22}$$

Fit linear and quadratic functions directly to the labels (`toyYtrain`) and training examples (`toyXtrain`) in `LogRegToy.mat` using the least-squares criterion. Report the empirical errors of the two naïve classifiers and the error obtained on the test data (`toyXtest` and `toyYtest`).

Note that we have previously implemented polynomial least-squares regression for Problem 5 in Problem Set 1. From our previous discussion, recall that we can solve for the least squares polynomial regression coefficients $\hat{\mathbf{w}}$ by using the extended *design matrix* $\tilde{\mathbf{X}}$ such that

$$\hat{\mathbf{w}} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\mathbf{y}, \text{ with } \tilde{\mathbf{X}} = \begin{pmatrix} 1 & x_1 & x_1^2 & \dots & x_1^d \\ 1 & x_2 & x_2^2 & \dots & x_2^d \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_N & x_N^2 & \dots & x_N^d \end{pmatrix},$$

where d is the degree of the polynomial and $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ and $\mathbf{y}$ are the observed points and their associated labels, respectively. To prevent numerical round-off errors, this method was applied to the column-normalized design matrix $\tilde{\mathbf{X}}$ using `degexpand.m` within the main script `prob8.m` on lines 55-107. For this problem, the classification error was measured by the percent of incorrect classifications on the training/test data. Under the decision rule in Equation 22, this results in the following definition of the least-squares classification error.

$$\text{Error-LSQ} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(\hat{\mathbf{w}}^T\mathbf{x} \geq 1/2\right)\right)^2$$

Applying this definition, the empirical errors for linear and quadratic least-squares classifiers were estimated using `prob8.m` and are tabulated below.

Least-squares Regression Model	Training Error	Test Error
Linear	7.5%	14.444%
Quadratic	2.5%	6.667%

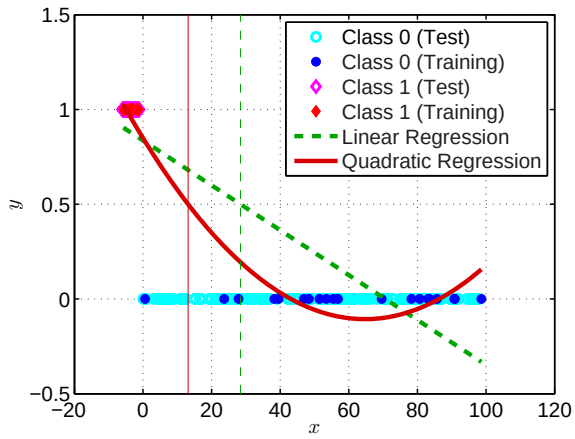
Part 2: An alternative to the naïve model above is the logistic regression model where the logistic function σ is fit to the posterior of a single class such that

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

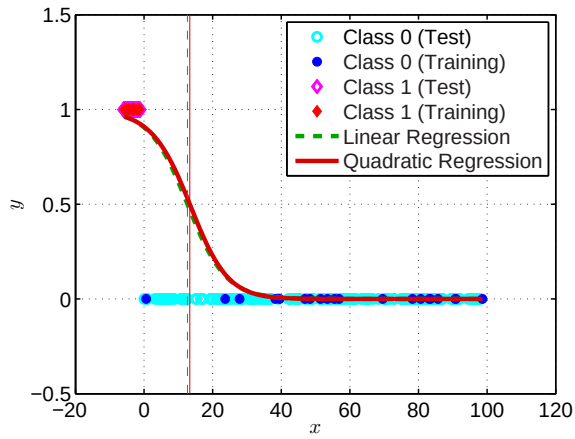
Apply linear and quadratic logistic regression to the examples by modifying `logisticRegression.m`. Report the training and test errors for these two models.

Polynomial logistic regression was implemented on lines 110-163 of `prob8.m`. Recall from class on 10/2/06 that the gradient of the log-probability is given by

$$\frac{\partial}{\partial \mathbf{w}}\log p(y_i|\mathbf{x}_i;\mathbf{w}) = \left(y_i - \sigma(w_0 + \mathbf{w}^T\mathbf{x}_i)\right)\begin{bmatrix} 1 \\ \mathbf{x}_i \end{bmatrix}.$$



(a) Least-squares Regression for `LogRegToy.mat`



(b) Logistic Regression for `LogRegToy.mat`

Figure 3: Comparison of least-squares and logistic polynomial regression.

Using this definition the gradient of the error function was implemented in `gradient.m` (with appropriate changes also made to `logisticRegression.m`). Recall from class on 9/29/06 that the corresponding decision boundary for the two-category logistic classifier is given by $p(y = 1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) = 1/2$. Under this decision rule the logistic classification error is defined as follows.

$$\text{Error-Logistic} = \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \left(\sigma(\mathbf{w}^T\mathbf{x}) \geq 1/2\right)\right)^2$$

Applying this definition, the empirical errors for linear and quadratic logistic classifiers were estimated using `prob8.m` and are tabulated below.

Logistic Regression Model	Training Error	Test Error
Linear	2.5%	6.667%
Quadratic	2.5%	6.667%

Part 3: Plot the functions found using least-squares and logistic regression (including the linear and quadratic cases). Compare these plots, the empirical errors, and briefly explain the results.

The linear and quadratic functions obtained using the least-squares criterion are shown in Figure 3(a). Similarly, the linear and quadratic functions obtained using logistic regresion are shown in Figure 3(b). Note that, in both figures, the decision boundaries are denoted by thin red/green lines – corresponding to the intersection of their respective classifier functions, shown by thick red/green lines, with $y = 1/2$.

From Figure 3(a) and the table in Part 1, it is apparent that the quadratic classifier achieves better performance on both the training and test data sets. As discussed in class on 9/29/06, there are several drawbacks of naïve classification using least-squared regression. Most significantly, the resulting classifiers are sensitive to outliers. As shown in the figure, the training data are not evenly distributed – the samples in Class 1 are more tightly clustered than those in Class 0 . As a result, we find that linear least squares regression biases heavily towards the right which results in greater out-of-sample test errors. Note, however, that the quadratic classifier will, for $x > 120$, begin to classify samples as in Class 1. This result may not be desirable and could lead to decreased generalization performance.

From Figure 3(b) and the table in Part 2, it is apparent that the linear and quadratic logistic classifiers achieve equivalent performance on both the training and test data sets. This is expected, since this problem involves a one-dimensional decision boundary (i.e., a single point). Unlike in higher dimensions, for this one-dimensional data set the added degree of freedom does not enable the quadratic classifier to achieve significantly-improved performance.

Problem 9

In this problem we will classify part of the MNIST data set of handwritten digits. The provided data set `digitData.mat` contains 10,000 examples from the four digits {1, 2, 3, 4}. We can classify each digit versus all others. Using 10-fold cross validation, evaluate the error of the linear logistic regression model for each of the four classification tasks. Which digit is most difficult to classify?

The Matlab script `prob9.m` was used to solve each of the four classification problems. The training data in `digitData.mat` was read and re-normalized to $[0, 1]$ on lines 14-47. On lines 50-103, 10-fold cross validation was implemented to evaluate the performance of the linear logistic classifier defined in `logisticRegression.m` and used in Problem 8. Recall that the k-fold cross validation score is defined as

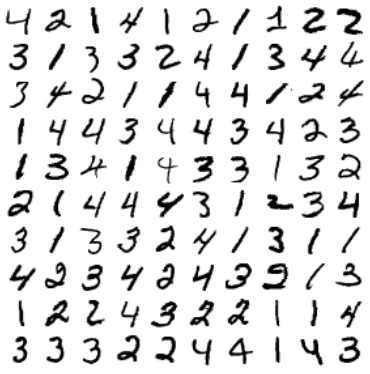
$$\hat{L}_k = \frac{1}{N} \sum_{i=1}^{k} \sum_{j \in \text{fold } i} \left(y_j - \left(\sigma(\hat{\mathbf{w}}_i^T \mathbf{x}_j) \geq 1/2 \right) \right)^2,$$

where $\hat{\mathbf{w}}_i$ is fit to all samples except those in the i^{th} fold. The resulting 10-fold cross-validation scores are tabulated below.

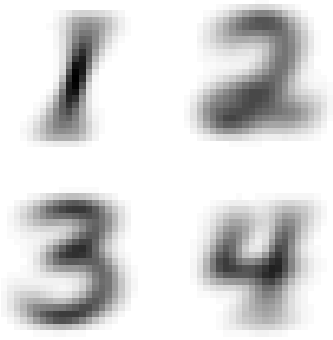
Classification Task	10-fold C.V. Error
1 vs. {2, 3, 4}	1.68%
2 vs. {1, 3, 4}	4.25%
3 vs. {1, 2, 4}	3.50%
4 vs. {1, 2, 3}	1.63%

Note that, due to the computational complexity of applying gradient descent in `logisticRegression.m`, the stopping criterion was chosen as the iteration when the change in error was 5.0 (0.01 was used in Problem 8). As a result, these estimates should be considered upper-bounds on the 10-fold cross-validation score.

From the table it is apparent that digit 2 is most difficult to classify, whereas digits 1 and 4 are the easiest. To better understand this result, examine the subset of training samples shown in Figure 4(a). Figure 4(b) shows the average across all members in each class and summarizes the general appearance of each digit. Recall that the decision rule under the linear logistic classifier is given by $\sigma(\mathbf{w}^T\mathbf{x}) \geq 1/2$. If we examine the argument of the logistic function, we find that $\mathbf{w}^T\mathbf{x}$ is simply a linear combination of the input values in $\mathbf{x}$. In other words, the parameters $\mathbf{w}$ of the logistic transform should have a strong positive weight for pixels that are unique to the class, while those shared by other classes should have a strong negative weight. From Figure 4(b) it is apparent that, on average, digit 2 does not possess many pixels which are not present in other classes. As a result the classification performance is diminished for the 2 vs. {1, 3, 4} task.



(a) Sample of the MNIST handwritten data set



(b) Class averages for `digitData.mat`

Figure 4: Samples and class averages obtained from a subset of the MNIST data set.

Problem 10

Recall that a principled multi-class generalization of the logistic regression model is the softmax model. Under this model the class posterior for class c becomes

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

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

$$\frac{\exp(\mathbf{w}_1^T\mathbf{x})}{\exp(\mathbf{w}_1^T\mathbf{x}) + \exp(\mathbf{w}_2^T\mathbf{x})} = \sigma(\mathbf{v}^T\mathbf{x}). \tag{23}$$

Recall from class on 9/29/06 that the logistic function σ is given by

$$\sigma(x) = \frac{1}{1+e^{-x}}. \tag{24}$$

Substituting Equation 24 into Equation 23 we obtain the following expression.

$$\frac{\exp(\mathbf{w}_1^T\mathbf{x})}{\exp(\mathbf{w}_1^T\mathbf{x}) + \exp(\mathbf{w}_2^T\mathbf{x})} = \frac{1}{1+\exp(-\mathbf{v}^T\mathbf{x})}$$

We can simplify this equation to obtain a closed form expression for $\mathbf{v}$ in terms of $\mathbf{w}_1$ and $\mathbf{w}_2$, as follows.

$$\begin{aligned}
\frac{1}{1+\exp(-\mathbf{v}^T\mathbf{x})} &= \frac{\exp(\mathbf{w}_1^T\mathbf{x})}{\exp(\mathbf{w}_1^T\mathbf{x}) + \exp(\mathbf{w}_2^T\mathbf{x})} \cdot \frac{\exp(-\mathbf{w}_1^T\mathbf{x})}{\exp(-\mathbf{w}_1^T\mathbf{x})} \\
&= \frac{\exp(\mathbf{w}_1^T\mathbf{x} - \mathbf{w}_1^T\mathbf{x})}{\exp(\mathbf{w}_1^T\mathbf{x} - \mathbf{w}_1^T\mathbf{x}) + \exp(\mathbf{w}_2^T\mathbf{x} - \mathbf{w}_1^T\mathbf{x})} \\
&= \frac{\exp(0)}{\exp(0) + \exp\left((\mathbf{w}_2^T - \mathbf{w}_1^T)\mathbf{x}\right)} \\
&= \frac{1}{1+\exp\left(-(\mathbf{w}_1 - \mathbf{w}_2)^T\mathbf{x}\right)}
\end{aligned}$$

In conclusion, we find that the softmax model for $C = 2$ is equivalent to logistic regression (i.e., Equation 23 is valid for all $\mathbf{w}_1$ and $\mathbf{w}_2$) with the following relationship between $\mathbf{v}$, $\mathbf{w}_1$, and $\mathbf{w}_2$.

$$\boxed{\mathbf{v} = \mathbf{w}_1 - \mathbf{w}_2}$$

(QED)

4 Naïve Bayes and Text Classification

Problem 11

Part 1: In this problem we will implement a Naïve Bayes (NB) spam filter for e-mail based on word occurrence (modeled as a Bernoulli random variable). The NB classifier will use binary features corresponding to the keywords provided in `dictionary.mat`. The j^{th} binary feature is defined as

$$\phi_j(\mathbf{x}) = \begin{cases} 1 & \text{if } \texttt{dictionary\{j\}} \text{ appears in } \mathbf{x} \text{ at least once,} \\ 0 & \text{otherwise.} \end{cases} \tag{25}$$

Write and turn in code for computing the binary features, estimating the parameters of the Bernoulli distributions (using a maximum likelihood procedure), and the NB classifier. Train the classifier using `trainSpam` and `trainHam1`. Report the classification errors on the test data sets.

The binary feature vectors defined by Equation 25 were computed on lines 18-52 of `estimateFeatures.m`, with email parsing handled by the provided routines `parseDirectory.m` and `parseEmail.m`. Each Bernoulli parameter $\hat{\theta}_{jc}$ (i.e., for feature j and class c) was determined on lines 54-57 of `estimateFeatures.m` using the ML estimator found in Problem 3. The NB classifier was implemented in `filterNB.m` using the decision rule derived on class on 9/27/06; a document with feature vector $\mathbf{x} = [\phi_1, \ldots, \phi_m]^T$ is classified as

$$\begin{aligned} \hat{y} = 1 \Leftrightarrow & \sum_{j=1}^{m} \phi_j \log(\theta_{j1}) + \sum_{j=1}^{m} (1-\phi_j) \log(1-\theta_{j1}) - \\ & \sum_{j=1}^{m} \phi_j \log(\theta_{j0}) - \sum_{j=1}^{m} (1-\phi_j) \log(1-\theta_{j0}) + \\ & \log(P_1) - \log(P_0) \geq 0, \end{aligned} \tag{26}$$

where "ham" is class 0 and SPAM is class 1 and P_1 and P_0 are the class priors (assumed to be equal). Using the Bernoulli parameters obtained with the `trainSpam` and `trainHam1` data sets, `prob11.m` was used to parse `testSpam` and `testHam` and estimate the empirical errors for the NB classifier. The results are tabulated below.

Total Error	"Ham" Error	SPAM Error
27.333%	40.0%	14.667%

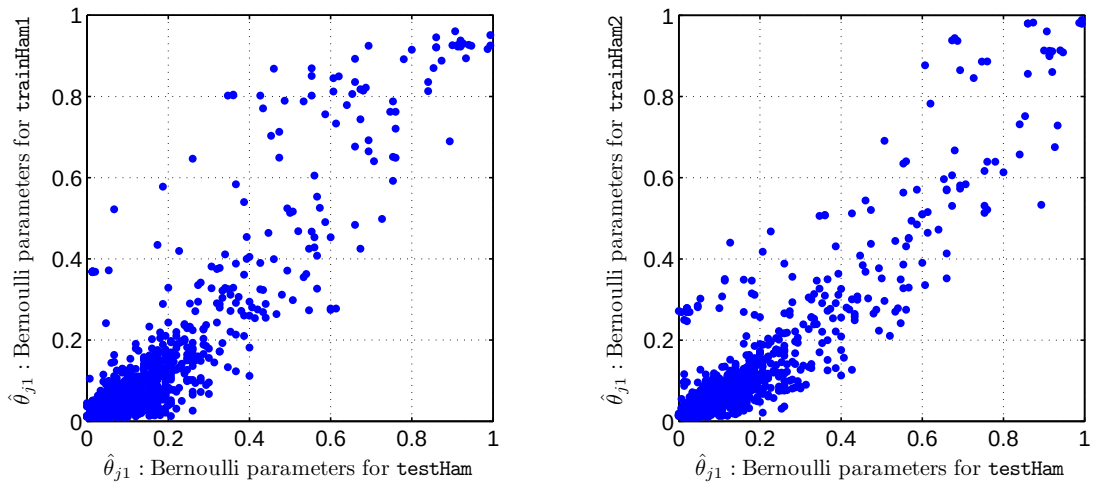
Note that the "ham" error corresponds to the percentage of legitimate emails that got classified as SPAM, whereas the SPAM errors are the percentage of SPAM emails that were classified as legitimate.

Part 2: Now repeat the experiment using `trainHam2` instead of `trainHam1`. Report the resulting classification errors. Compare the performance of the classifiers trained on different non-SPAM examples.

The procedure in the previous part was repeated using `trainHam2` instead of `trainHam1`. The results are tabulated below.

Total Error	"Ham" Error	SPAM Error
23.667%	29.333%	18.0%

Note that, overall, both NB classifiers achieved a total error rate less than 30%, which is significantly better than a random guess. Comparing these results, it also appears that the classifier trained on `trainHam2` performed better than that trained on `trainHam1`. Intuitively, one would expect that `trainHam2` must more-closely represent the variation within the test "ham" data set `testHam` than `trainHam1`. That is, the estimates of the Bernoulli parameters using `trainHam2` must more accurately model the parameters for



(a) Correlation between `testHam` and `trainHam1` (b) Correlation between `testHam` and `trainHam2`

Figure 5: Comparison of estimated Bernoulli parameters using various "ham" training data sets.

`testHam`. We can evaluate this hypothesis by estimating the Bernoulli parameters on both training sets and the test data and examining their similarity. This procedure was implemented using `compareBernoulli.m`. The resulting scatter plots are shown in Figure 5. Note that the scatter plot in Figure 5(b), comparing `testHam` with `trainHam2`, demonstrates that the Bernoulli parameters are *overestimated* using `trainHam2`. As a result, fewer emails are classified as spam and, for the limited test set of 150 emails, this resulted in a lower overall classification error, but an increased error rate for SPAM.

Problem 12

In practice, the cost of misclassifying a valid e-mail is much higher than the cost of misclassifying SPAM. To be concrete, assume that the loss incurred by failing to detect a SPAM message is 1, whereas the loss incurred by mistakenly flagging a non-SPAM is 3. Explain how to modify the NB classifier in Problem 11 to account for this asymmetric loss. Implement the necessary changes and report the new test errors as well as the average loss on the test set.

Recall from class on 9/20/06 that the risk $R(h)$, or the expected loss, of a two-way classifier $h(\mathbf{x})$ is given by

$$\begin{aligned} R(h) &= \int_{\mathbf{x}} \sum_{c=1}^{2} L(h(\mathbf{x}), c) p(\mathbf{x}, y = c) d\mathbf{x} \\ &= \int_{\mathbf{x}} \left[\sum_{c=1}^{2} L(h(\mathbf{x}), c) p(y = c|\mathbf{x}) \right] p(\mathbf{x}) d\mathbf{x}. \end{aligned}$$

To minimize the risk $R(h)$ it is sufficient to minimize the conditional risk $R(h|\mathbf{x})$ for any $\mathbf{x}$ such that

$$R(h|\mathbf{x}) = \sum_{c=1}^{2} L(h(\mathbf{x}), c) p(y = c|\mathbf{x}). \tag{27}$$

For the 0/1 loss model, Equation 27 leads to the NB classifier given in Equation 26. For an asymmetric loss function, however, we must define a *loss matrix* such that

$$L_{ij} = \begin{pmatrix} 0 & 3 \\ 1 & 0 \end{pmatrix}, \tag{28}$$

where L_{ij} is the loss incurred for classifying a class i e-mail as class j, where class 0 is legitimate e-mail and class 1 is SPAM. As described on page 42 in [2], the decision rule that minimizes the expected loss will assign each sample $\mathbf{x}$ such that

$$\hat{y} = 1 \Leftrightarrow L_{01} p(y = 0|\mathbf{x}) + L_{11} p(y = 1|\mathbf{x}) \leq L_{00} p(y = 0|\mathbf{x}) + L_{10} p(y = 1|\mathbf{x}).$$

Substituting for the loss matrix from Equation 28, applying Bayes rule, and taking the logarithm of the inequality gives the following form for the decision rule.

$$\hat{y} = 1 \Leftrightarrow \log \left(\frac{p(\mathbf{x}|y = 1) P_1}{p(\mathbf{x}|y = 0) P_0} \right) \geq \log(3) \tag{29}$$

Under the Naïve Bayes assumption the features are independent given their class such that

$$p(\mathbf{x}|y = c) = \prod_{j=1}^{m} p(\phi_j(\mathbf{x})|y = c). \tag{30}$$

Substituting Equation 30 into Equation 29 and applying the properties of the logarithm gives the following form for the decision rule.

$$\hat{y} = 1 \Leftrightarrow \sum_{j=1}^{m} \log \left(p(\phi_j(\mathbf{x})|y = 1) \right) - \sum_{j=1}^{m} \log \left(p(\phi_j(\mathbf{x})|y = 0) \right) + \log(P_1) - \log(P_0) \geq \log(3) \tag{31}$$

Under the Bernoulli distribution $\phi_j \sim \text{Bern}(\phi_j|\theta jy)$ and we have

$$p(\phi_j(\mathbf{x})|y = 0) = \theta_{j0}^{\phi_j} (1 - \theta_{j0})^{1 - \phi_j} \tag{32}$$

$$p(\phi_j(\mathbf{x})|y = 1) = \theta_{j1}^{\phi_j} (1 - \theta_{j1})^{1 - \phi_j}. \tag{33}$$

Substituting Equation 32 and Equation 33 into Equation 31 gives the following form for the Naïve Bayes classifier.

$$\begin{aligned}\hat{y} = 1 \Leftrightarrow &\sum_{j=1}^{m} \phi_j \log(\theta_{j1}) + \sum_{j=1}^{m} (1-\phi_j) \log(1-\theta_{j1}) - \\ &\sum_{j=1}^{m} \phi_j \log(\theta_{j0}) - \sum_{j=1}^{m} (1-\phi_j) \log(1-\theta_{j0}) + \\ &\log(P_1) - \log(P_0) \geq \log(3)\end{aligned}$$

Using the Bernoulli parameters obtained with `trainSpam` and `trainHam1`, `prob12.m` was used to parse `testSpam` and `testHam` and estimate the empirical errors and average loss for the modified NB classifier. The results are tabulated below.

Total Error	"Ham" Error	SPAM Error	Average Loss
26.667%	38.667%	14.667%	0.653

Similarly, the results using `trainSpam` and `trainHam2` are tabulated below.

Total Error	"Ham" Error	SPAM Error	Average Loss
24.0%	29.333%	18.667%	0.533

Note that, overall, the modifications made to the NB classifier had a negligible effect on the classification errors. Assuming that there isn't an error in our classifier, this can be attributed to putting too small a penalty on misclassifying "ham" e-mails. As discussed in [4], the cost of false positives vs. false negatives can range from 1 up to 1000; experimentally, it was found that increasing the penalty on misclassifying legitimate e-mails did lead to a reduced classification error for "ham" (while increasing the SPAM and total error rate).

Problem 13

Now suppose that the NB classifier is limited to ten features. Propose a method for selecting the ten keywords from the provided dictionary, list the keywords, and report the test error of the resulting classifier. (Use `trainSpam` and `trainHam2` for training.)

The original dictionary contains $1,420$ words. In general, we would like to select the ten most discriminative words. That is, we'd like some words that are highly-indicative of SPAM and some that are highly-indicative legitimate email. While mutual information metrics are typically used for this selection task, I propose a simpler selection procedure based on maximizing the difference in the Bernoulli parameters between classes [4]. More specifically, I select the ten words with the highest value of the "Bernoulli difference" Δ_j, where

$$\Delta_j \triangleq |\theta_{j1} - \theta_{j0}|.$$

In general, these keywords are outliers in the "Bernoulli space" shown in Figure 6.

This criterion was implemented using `redEstimateParams.m`; this Matlab routine parsed the training sets `trainSpam` and `trainHam2` and selected the following words.

Keyword	θ_{j0}	θ_{j0}
`jm@localhost`	0.887	0.232
`yyyy@localhost`	0.886	0.234
`imap`	0.899	0.380
`plain`	0.877	0.449
`drop`	0.912	0.498
`our`	0.122	0.625
`smtp`	0.275	0.724
`your`	0.329	0.770
`br`	0.083	0.506
`please`	0.113	0.525

Note that the reduced dictionary was saved as `redDictionary.mat` with Bernoulli parameters given in `redBernParams.mat`. Examining the keywords it is apparent that the selection criterion found five "ham" keywords, given by {`jm@localhost, yyyy@localhost, imap, plain, drop`}, and five "spam" words, given by {`our, smtp, your, br, please`}. A review of `trainHam2` finds that many SPAM emails contain HTML fragments; as a result, it is not surprising that `br`, the HTML clear tag, was selected as a top SPAM keyword.

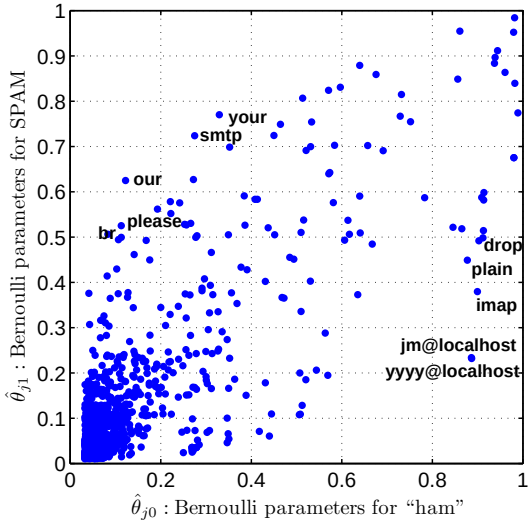


Figure 6: Keyword selection in the Bernoulli parameter space.

The reduced dictionary was used with the NB classifier developed in Problem 11 and evaluated on the test data given in `testSpam` and `testHam`. The results are tabulated below.

Total Error	"Ham" Error	SPAM Error
35.667%	26.0%	45.333%

Not surprisingly, the overall performance of the NB classifier is significantly degraded in comparison to the results for the complete dictionary found in Problem 11. More specifically, the reduction in the "ham" error rate is more than compensated for by the dramatic increase in SPAM classification errors. In general we find that, while a reduced dictionary may be more efficient, care must be taken to select enough keywords to maintain acceptable classification errors.

Problem 14: Extra Credit

Instead of binary features we could use a representation that encodes relative frequencies of words. Let N_j be the number of times the j^{th} keyword appears in the e-mail $\mathbf{x}$. Then we can define

$$\phi'_j(\mathbf{x}) \triangleq N_j/N, \tag{34}$$

where $N = \sum_{j=1}^{m} N_j$. Propose a distribution model for these features and ML estimator(s) for its parameter(s). Build a Naïve Bayes classifier using these features and report its performance on the test set using 0/1 loss. What can you say about the validity of the NB assumptions?

First, note that $\phi'_j(\mathbf{x})$ will be a continuous random variable on the interval $[0, 1]$. As a result, we propose to model $\phi'_j(\mathbf{x})$ using a Gaussian distribution. Recall from pages 78 in [2] that the Gaussian distribution $\mathcal{N}(x|\mu, \sigma^2)$ has the following form.

$$\mathcal{N}(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

From Problem 8 of Problem Set 1, the ML estimators for the mean μ and varance σ^2 are given by the following expressions.

$$\hat{\mu}_{ML} = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \text{and} \quad \hat{\sigma}^2_{ML} = \frac{1}{N}\sum_{i=1}^{N}(x_i - \hat{\mu}_{ML})^2$$

The Matlab routine `freqEstimateParams.m` was used to compute the relative frequency features defined by Equation 34 and estimate the parameters of the Gaussian distributions using the ML estimators.

Recall from class on 9/26/06 that the two-category Naïve Bayes classifier, under 0/1 loss, is given by

$$h_{NB}(\mathbf{x}) = 1 \Leftrightarrow \log\left(\frac{P_1}{P_0}\prod_{j=1}^{m}\frac{p(\phi_j(\mathbf{x})|y=1)}{p(\phi_j(\mathbf{x})|y=0)}\right) \geq 0, \tag{35}$$

where P_1 and P_1 are the class priors (assumed to be equal). For the Gaussian distribution $\phi_j \sim \mathcal{N}(\phi_j|\mu_{jy}, \sigma^2_{jy})$ and we have

$$p(\phi_j(\mathbf{x})|y=0) = \frac{1}{\sqrt{2\pi}\sigma_{j0}} \exp\left(-\frac{(\phi_j - \mu_{j0})^2}{2\sigma^2_{j0}}\right) \tag{36}$$

$$p(\phi_j(\mathbf{x})|y=1) = \frac{1}{\sqrt{2\pi}\sigma_{j1}} \exp\left(-\frac{(\phi_j - \mu_{j1})^2}{2\sigma^2_{j1}}\right). \tag{37}$$

Substituting Equations 36 and 37 into Equation 35, we obtain the following decision rule for the NB classifier.

$$h_{NB}(\mathbf{x}) = 1 \Leftrightarrow \frac{1}{2\sigma^2_{j0}}\sum_{i=1}^{m}(\phi_j - \mu_{j0})^2 - \frac{1}{2\sigma^2_{j1}}\sum_{i=1}^{m}(\phi_j - \mu_{j1})^2 + m\log\left(\frac{\sigma_{j0}}{\sigma_{j1}}\right) + \log(P_1) - \log(P_0) \geq 0$$

Using the Gaussian parameters obtained with the `trainSpam` and `trainHam1` data sets, `prob14.m` was used to parse `testSpam` and `testHam` and estimate the empirical errors for the NB classifier (implemented in `freqFilterNB.m`). The results are tabulated below.

Total Error	"Ham" Error	SPAM Error
22.333%	20.667%	24.0%

The results using `trainSpam` and `trainHam2` are tabulated below.

Total Error	"Ham" Error	SPAM Error
24.333%	9.333%	39.333%

In general, the overall error rate is comparable to that achieved with 0/1 loss in Problem 11 using word occurrence under a Bernoulli distribution. Note, however, that the error rate for legitimate email is significantly reduced. As a result, the relative word frequency feature under a Gaussian distribution does a better job at blocking spam *while* making fewer classification errors on legitimate emails. Further discussion of distribution selection and features is covered in [4].

Note that the Naïve Bayes assumption, namely that the features are independent given the class, is not satisfied for the relative word frequency feature in Equation 34. Due to their normalization, we must have $\sum_{j=1}^{m} \phi'_j(\mathbf{x}) = 1$. As a result, if a single value of $\phi'_j(\mathbf{x})$ is close to one, then the remaining values must be near zero – clearly violating the independence assumption. Although the NB assumption does not hold, we find that the results are still comparable to those previously seen in Problem 11 through 13. This illustrates that, even if the NB assumption doesn't hold, one can still proceed as if it does and obtain a simple classification scheme with some utility.

References

[1] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1996.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, 2006.

[3] Richard O. Duda, Peter E. Hart, and David G. Stork. *Pattern Classification (Second Edition)*. Wiley-Interscience, 2000.

[4] Vangelis Metsis, Ion Androutsopoulos, and Georgios Paliouras. Spam filtering with naive bayes – which naive bayes? Third Conference on Email and Anti-Spam (CEAS), 2006.