M5MS10 Machine Learning - Coursework 1

Jerome Wynne, CID 01540830

Question 1

We wish to build a classifier that predicts the personality type of previously unseen subjects on the basis of their D binary ratings. Write computer code to generate an artificial data set. The input parameters should be:

- *N, the number of training examples.*
- *D, the number of features (i.e. distinct item preferences).*
- *$p(C_j)$, the prior probability of being an introvert (a base rate for the population).*
- *p_{ij}, preference probabilities given class membership.*

Use your code to generate a training data set $\mathcal{D}$ and an independent testing data set $\mathcal{T}$, for some chosen parameters with $D > 2$ and all probabilities greater than 0. Comment on your choice of parameters, and describe some aspects of the simulated data sets you have generated and how these might affect subsequent learning.

Throughout this report we adopt the convention that $C_1 = 1$ and $C_2 = 0$. In code, y refers to a vector of observed classes.

Take $D := 3$, $N := 100$, and set the prior probability $P(C = C_1) := 0.5$ so that $P(C = C_2) = 0.5$. The preference probabilities given class membership, $p_{ij} = P(x_i = 1|C = C_j)$, are set according to Table 1.

The prior probability and preference probabilities conditional on class together determine the maximum accuracy that a classifier can achieve on a given query. To see why this is the case, consider predicting the class of a query with feature vector $\mathbf{x}$. The probability that a decision rule $d(\mathbf{x})$ misclassifies this query is $P(d(\mathbf{x}) \neq C|\mathbf{x})$. Assume that d is a probabilistic decision rule that returns 1 with probability $\delta(\mathbf{x})$. $d(\mathbf{x})$ is independent of C conditional on $\mathbf{x}$. Fix $P(C = 1|\mathbf{x}) = c(\mathbf{x})$. The probability of misclassification conditional on $\mathbf{x}$ is then:

$$\begin{aligned} P(d(\mathbf{x}) \neq C(\mathbf{x})) &= P(\{d(\mathbf{x}) = 1, C = 0\} \cup \{d(\mathbf{x}) = 0, C = 1\}|\mathbf{x}) && (1)\\ &= P(d(\mathbf{x}) = 1|\mathbf{x})P(C = 0|\mathbf{x}) + P(d(\mathbf{x}) = 0|\mathbf{x})P(C = 1|\mathbf{x}) && (2)\\ &= \delta(\mathbf{x})(1 - c(\mathbf{x})) + (1 - \delta(\mathbf{x}))c(\mathbf{x}) && (3)\end{aligned}$$

Where Line $1 \to 2$ exploits the disjoint events property of the probability measure and the conditional independence of $d(\mathbf{x})$ w.r.t. C given $\mathbf{x}$. The conditional misclassification probability can be minimized with respect to $\delta(\mathbf{x})$ by setting:

$$\delta(\mathbf{x}) = \begin{cases} 1 & \text{if } 0.5 < c(\mathbf{x}) \\ 0 & \text{if } c(\mathbf{x}) < 0.5 \\ q & \text{if } c(\mathbf{x}) = 0.5 \text{ where } 0 \leq q \leq 1 \end{cases} \tag{4}$$

Table 1: Specification of preference probabilities given class membership for datasets $\mathcal{D}$ and $\mathcal{T}$.

p_{ij}	$j = 1$	$j = 2$
$i = 1$	0.8	0.1
$i = 2$	0.8	0.6
$i = 3$	0.8	0.8

The misclassification probability for an optimal[1] classifier d^* is therefore:

$$P(d^*(\mathbf{x}) \neq C|\mathbf{x}) = \min\{c(\mathbf{x}), 1 - c(\mathbf{x})\} \tag{5}$$

This equation implies that the only quantity affecting the performance of an optimal classifier when given query $\mathbf{x}$ is $P(C = 1|\mathbf{x}) = c(\mathbf{x})$[2]. How does this probability relate to our choice of parameters for data generation? The odds ratio for the query is:

$$\frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \frac{p(C=1)}{p(C=0)} \frac{p(\mathbf{x}|C=1)}{p(\mathbf{x}|C=0)} \tag{6}$$

We assume that the features are independent conditional on the class, so we can factorise the right-hand side and substitute for the data generation parameters $p(C = 1), p_{ij}$:

$$\frac{p(C=1|\mathbf{x})}{p(C=0|\mathbf{x})} = \frac{p(C=1)}{p(C=0)} \prod_{i=1}^{D} \frac{p(x_i|C=1)}{p(x_i|C=0)} \tag{7}$$

$$= \frac{p(C=1)}{p(C=0)} \left(\prod_{i:x_i=1} \frac{p_{i1}}{p_{i2}} \right) \left(\prod_{i:x_i=0} \frac{1-p_{i1}}{1-p_{i2}} \right) \tag{8}$$

Denote the r.h.s. of Equation 8 by $u(\mathbf{x})$, apply the complement property of the probability measure to the denominator of the l.h.s., then rearrange to obtain:

$$P(C=1|\mathbf{x}) = c(\mathbf{x}) = \frac{u(\mathbf{x})}{1+u(\mathbf{x})} \tag{9}$$

Comparing Equation 5 and Equation 9, we can see that the query-conditional misclassification rate of the optimal classifier is only affected by u, which depends on our choice of parameters through the product of ratios u from Equation 5. For $u >> 1$ and $0 < u << 1$, the conditional misclassification probability is small[3]. This implies that a feature x_i is useful for prediction if $\frac{p_{i1}}{p_{i2}} >> 1$ or $0 < \frac{p_{i1}}{p_{i2}} << 1$. Features x_i for which $p_{i1} = p_{i2}$ do not affect the performance of the optimal classifier and can be omitted without loss of accuracy. Several weakly informative features e.g. x_1, x_2, x_3, for which $\prod_{i=1}^{3} \frac{p_{i1}}{p_{i2}} = a$ and $p_{i1} < p_{i2}$ for $i = 1, 2, 3$, are equivalent to a single informative feature $x'_1 : \frac{p_{1'1}}{p_{1'2}} = a$ in the sense that the maximum change in conditional misclassification probability due to variation in (x_1, x_2, x_3) is the same as that caused by variation in x'_1.

Based on the observations above, x_1 was made useful for prediction ($p_{11}/p_{12} = 0.8/0.1 = 8$), x_2 somewhat useful ($p_{21}/p_{22} = 0.8/0.6 = 1.333\ldots$), and x_3 redundant ($p_{31}/p_{32} = 0.8/0.8 = 1$). The base rate was balanced so that either class was equally likely - knowing that a query has been drawn from the data-generating distribution then provides no information as to that query's class.

The number of parameters that the linear classifier in Question 2 must learn scales with the dimensionality of the feature vector according to $2D$. D also interacts with the choice of p_{ij}: sampling many features that are independent and weakly indicative of class is likely to produce an odds ratio that favours the true class. $D = 3$ was chosen to make the case study easier to present and implement.

The sample size N was set based on the class prior and the variance of the linear classifier.. If $P(C = C_1) \approx 1$ so that $P(C = 0) \approx 0$, then - when sampling probabilistically - large N is required to obtain at least one observation with $C = 0$. If observations from both class are not obtained, then it is not possible to reason about how the feature distributions of either class differ. Given $P(C = C_1) = 0.5$, the probability of generating such a pathological sample for $N = 100$ is $\approx 10^{-30}$. Additionally, the linear classifier relies on parameter estimates to evaluate the odds ratio of Equation 8. The lienar classifier is therefore affected by the choice of parameter values in the same ways that the optimal classifier is, but suffers from the additional difficulty that it must estimate these parameters. An imprecise parameter estimate can degrade classifier accuracy. Setting N large meant that the classifier's fit would be similar under repeated sampling.

The code below defines a function that can generate a dataset of the type specified.

[1]in the sense that it has the greatest probability of classifying a query with known $\mathbf{x}$ correctly.

[2]NB in general, a classifier's performance depends on the distribution of $\mathbf{x}$.

[3]This is intuitive - the more a feature's class-conditional distribution changes with class, the more useful that feature is for prediction.

```r
# Data generation function
generateDataset <- function(N=100, D=3, PC1=0.5,
                            Pij=matrix(c(0.8, 0.1, 0.8, 0.6, 0.8, 0.8),
                            nrow=D, byrow=T)){
        # Sample class
        y <- rbinom(N, 1, prob=PC1)

        # Sample features according to conditional prob.
        X0 <- c()
        X1 <- c()
        X  <- matrix(rep(NA, D*N), nrow=N)
        for (k in 1:D) { # For each feature
          X1 <- rbinom(sum(y==1), 1, prob=Pij[k, 1]) # Sample feature for C = 1 obs.
          X2 <- rbinom(sum(y==0), 1, prob=Pij[k, 2]) # Sample feature for C = 0 obs.
          x  <- rep(NA, N) # tmp storage variable
          x[y==1] <- X1 # align with corresponding class
          x[y==0] <- X2
          X[ , k] <- x # include feature in observation matrix
        }

        return(list(X, y))
    }
```

```r
set.seed(69)

N_train <- 100
N_test  <- 100
D   <- 3
Pij <- matrix(c(0.8, 0.1, 0.8, 0.6, 0.8, 0.8), nrow=D, byrow=T)
PC1 <- 0.5

tmp <- generateDataset(N=N_train, D=D, Pij=Pij, PC1=PC1)
X.train <- tmp[[1]]
y.train <- tmp[[2]]

tmp <- generateDataset(N=N_test, D=D)
X.test <- tmp[[1]]
y.test <- tmp[[2]]
```

Question 2

Derive the generative linear classifier using a discriminant function of the form

$$g(\mathbf{x}) = \log \frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})} = \log \frac{p(\mathbf{x}|C_1)p(C_1)}{p(\mathbf{x}|C_2)p(C_2)} \tag{10}$$

and write computer code to train this classifier on $\mathcal{D}$. What form does your class conditional distribution of the data take? Discuss your choice of prior distribution and any assumptions or implications of this. What kind of learning algorithm does our assumptions result in?

The data-generating mechanism produces binary outcomes, which indicates that a multivariate Bernoulli likelihood model should be used. The coursework briefing states that the features are conditionally independent given the observed class, allowing us to make the Naive Bayes assumption. This pair of assumptions produces a multivariate Bernoulli Naive Bayes classifier that has parameters corresponding to the class probability, $\theta_1 = P(C = C_1)$, and the feature class-conditional probabilities, $\theta_{ij} = P(x_i = 1|C = C_j)$. We denote $(\theta_1, \theta_{11}, \theta_{12}, \ldots, \theta_{D1}, \theta_{D2})$ by θ

Using the above assumptions, the model's likelihood for a single observation can be written:

$$p(\mathbf{x}, C|\theta) = p(C|\theta_1) \prod_{i=1}^{D} p(x_i|\theta_{ij}, C = C_j) \tag{11}$$

The parameters take on values in $(0, 1)$ and we assume there is no reason to think that the parameters are correlated[4]. A suitable prior is therefore a product of Beta distributions:

$$p(\theta) = p(\theta_1) \prod_{i=1}^{D} \prod_{j=1}^{2} p(\theta_{ij}) \tag{12}$$

Where $p(\theta_1)$ and $p(\theta_{ij})$ are Beta density functions. These require two hyperparameters to be set (denoted α and β). In a commercial context it would be worthwhile to assess how desirable the items are to introverts and extroverts and select Beta parameters accordingly. In this exercise however, we simply take $\alpha = \beta = 1$ for all parameters so that the prior is a uniform distribution over $(0, 1)^{2D+1}$.

Inference is straightforward because the Beta distribution is conjugate with respect to the multivariate Bernoulli likelihood. It can be shown [1] that the posterior distribution is given by:

$$p(\theta_1|\mathcal{D}) = \text{Beta}(N_1 + 1, N_2 + 1) \tag{13}$$

$$p(\theta_{ij}|\mathcal{D}) = \text{Beta}(N_{ij} + 1, (N_j - N_{ij}) + 1) \tag{14}$$

where $N_1 = n(C = C_1)$, $N_2 = n(C = C_2) = N - N_1$, and $N_{ij} = n(x_i = 1, C = C_j)$. n is a function that receives a condition and returns the number of examples in the dataset $\mathcal{D}$ that satisfy it. Training the classifier therefore consists of evaluating counts N_j, N_{ij} on the training dataset.

The posterior mean for the parameter values is:

$$\bar{\theta}_1 = \frac{N_1 + 1}{N + 2} \tag{15}$$

$$\bar{\theta}_{ij} = \frac{N_{ij} + 1}{N_j + 2} \tag{16}$$

[4]In practice, there may be good reason to think that the parameters are correlated. For example we may be unsure how consumers feel about movie q and movie r, but feel confident that consumers will feel the same way about both objects (e.g. if they are similar items).

Where the $+1$ and $+2$ emerge as consequence of our choice of prior hyperparameters. By contrast, the maximum likelihood estimates for the parameters are:

$$\hat{\theta}_1 = \frac{N_1}{N} \tag{17}$$

$$\hat{\theta}_{ij} = \frac{N_{ij}}{N_j} \tag{18}$$

Expression pairs $(15, 16)$ and $(17, 18)$ produce similar parameter estimates. A crucial difference between them is that Equation 18 can take on a value of 0 if $N_{ij} = 0$ for any $i = 1, \ldots, D$. This is problematic since it means the conditional probability $p(C = C_j|\mathbf{x}) = 0$ and the discriminant function g is undefined[5]. The posterior mean values, $\bar{\theta}$, do not suffer from this problem since they must be in $(0, 1)$. This comparison highlights the importance of taking a Bayesian approach to this problem.

Our assumptions result in a learning algorithm that estimates the model parameters directly - it does not use a numerical optimization procedure, since properties of the posterior distribution can be obtained in closed form.

```
  N1 <- sum(y.train)
  N2 <- N_train - N1

  Nij <- matrix(rep(NA, 2*D), nrow=D) # Theta_ij = P(x_i = 1|C = C_j)
  Nij[, 1] <- colSums(X.train[y.train == 1, ])
  Nij[, 2] <- colSums(X.train[y.train == 0, ])

  theta1_bar  <- (N1 + 1)/(N_train + 2)
  thetaij_bar <- Nij
  thetaij_bar[ , 1] <- (thetaij_bar[ , 1] + 1)/(N1 + 2)
  thetaij_bar[ , 2] <- (thetaij_bar[ , 2] + 1)/(N2 + 2)

  print(theta1_bar) # Posterior mean for base rate (estimate)

## [1] 0.4901961

  print(PC1)        # Actual value

## [1] 0.5

  print(thetaij_bar) # Posterior mean for class-conditional prob. (estimate)

##           [,1]       [,2]
## [1,] 0.7843137 0.09433962
## [2,] 0.8039216 0.60377358
## [3,] 0.8627451 0.77358491

  print(Pij)         # Actual value

##      [,1] [,2]
## [1,]  0.8  0.1
## [2,]  0.8  0.6
## [3,]  0.8  0.8
```

The posterior expected value parameter estimates can be plugged into the discriminant function g:

$$g(\mathbf{x}) = \log\left(\frac{\bar{\theta}_1}{1-\bar{\theta}_1}\frac{\prod_{i=1}^{D}\bar{\theta}_{i1}^{x_i}(1-\bar{\theta}_{i1})^{1-x_i}}{\prod_{i=1}^{D}\bar{\theta}_{i2}^{x_i}(1-\bar{\theta}_{i2})^{1-x_i}}\right) \tag{19}$$

$$= \log\frac{\bar{\theta}_1}{1-\bar{\theta}_1} + \sum_{i:x_i=1}\log\frac{\bar{\theta}_{i1}}{\bar{\theta}_{i2}} + \sum_{i:x_i=0}\log\frac{1-\bar{\theta}_{i1}}{1-\bar{\theta}_{i2}} \tag{20}$$

[5]The same type of problem occurs if $\hat{\theta}_{ij} = 1$.

Which can be expressed in the canonical form of a linear classifier:

$$g(\mathbf{x}) = \mathbf{x}^T \left(\log \frac{\bar{\theta}_{\cdot 1}}{\bar{\theta}_{\cdot 2}} - \log \frac{\mathbf{1} - \bar{\theta}_{\cdot 1}}{\mathbf{1} - \bar{\theta}_{\cdot 2}} \right) + \mathbf{1}^T \log \frac{\mathbf{1} - \bar{\theta}_{\cdot 1}}{\mathbf{1} - \bar{\theta}_{\cdot 2}} + \log \frac{\bar{\theta}_1}{1 - \bar{\theta}_1} \tag{21}$$

where $\bar{\theta}_{\cdot 1} = (\bar{\theta}_{11}, \bar{\theta}_{21}, \ldots, \bar{\theta}_{D1})^T$, $\mathbf{1}$ is column vector of D ones, and division and $\log$ are element-wise.

Question 3

Write computer code to compute the predicted class as well as the classification error rate for an input data set with known labels, then estimate the classification error obtained on both $\mathcal{D}$ and $\mathcal{T}$. Comment on your results.

```
g <- function(x, theta1, Theta) {
  # Obtain numerator, p(x, C = C_1)
  num <- log(theta1) + sum(log(Theta[(x==1), 1])) + sum(log(1 - Theta[(x==0), 1]))
  # Obtain denominator, p(x, C = C_2)
  den <- log(1 - theta1) + sum(log(Theta[(x==1), 2])) + sum(log(1 - Theta[(x==0), 2]))
  return(as.numeric(num > den))
}

# Compute classification error rates
getErrorRate <- function(y_true, y_pred) mean(y_true != y_pred)

# Make predictions using rows of training and testing data sets
y_hat.train <- apply(X.train, MARGIN=1,  function(x) g(x, theta1_bar, thetaij_bar))
y_hat.test  <- apply(X.test, MARGIN=1, function(x) g(x, theta1_bar, thetaij_bar))

# Error rate with estimated parameters
print(getErrorRate(y.train, y_hat.train))

## [1] 0.14

print(getErrorRate(y.test, y_hat.test))

## [1] 0.19

# Error rate with true parameter values
print(getErrorRate(y.train, apply(X.train, MARGIN=1, function(x) g(x, PC1, Pij))))

## [1] 0.14

print(getErrorRate(y.test, apply(X.test, MARGIN=1, function(x) g(x, PC1, Pij))))

## [1] 0.19
```

We can see that the misclassification rate of the linear classifier is unaffected by whether the estimated or actual parameter values are used. This suggests that the misclassification rate is not attributable to imprecise parameter estimates. Instead, the errors must be due to the previously discussed limits on how predictive the features can be of class. Furthermore, the difference between training and testing error rate seems likely to be due to variability in the misclassification rate rather than variance in classifier's fit (i.e. it is unlikey that the classifier overfits because the variance of its parameter estimates is small).

Question 4

Generate a second independent testing data set, but now make sure that the distribution of each x_i in this new set is slightly different from the distribution of the data in $\mathcal{D}$. Assess whether *the performance of the classifier, as measured using the classification error rate, has changed. Comment on your results.*

```
perturbed_Pij <- Pij
perturbed_Pij[1, 1] <- 0.6 # => degrade class separation in x1
tmp <- generateDataset(N=N_test, D=D, PC1=PC1, Pij=perturbed_Pij)
X.testp <- tmp[[1]]
y.testp <- tmp[[2]]

# Evaluate misclassification rate
y_hat.testp  <- apply(X.testp, MARGIN=1, function(x) g(x, theta1_bar, thetaij_bar))
print(getErrorRate(y.testp, y_hat.testp)) # Decrease in misclassification rate

## [1] 0.25
```

The classifier's testing misclassification rate increased from ≈ 0.1 to ≈ 0.3 when $p_{11} : 0.8 \to 0.6$. The reason for this is clear: making the data-generating p_{ij} less consistent with how the classifier would classify a point[6] made it more likely that, after perturbation, a data point would be sampled on the incorrect side of the decision boundary.

The above example suggested that perturbing the class distribution of the data so that it was better-separated, while still being consistent with the feature's indicated class[7], would improve the classifier's misclassification rate. This is because data points of both classes become more likely to be sampled on the correct side of the classifier's decision boundary. The code below corroborates this speculation.

```
perturbed_Pij <- Pij
perturbed_Pij[1, 1] <- 0.99 # => improve class separation in x1
tmp <- generateDataset(N=N_test, D=D, PC1=PC1, Pij=perturbed_Pij)
X.testp <- tmp[[1]]
y.testp <- tmp[[2]]

y_hat.testp  <- apply(X.testp, MARGIN=1, function(x) g(x, theta1_bar, thetaij_bar))
print(getErrorRate(y.testp, y_hat.testp)) # decrease in misclassification rate

## [1] 0.05
```

Question 5

Plot the decision boundary when $p(C_1) = p(C_2) = 0.5$, $D = 3$, $p_{i1} = 0.8$, $p_{i2} = 0.5$ (for $i \in \{1, 2\}$), and $p_{31} = p_{32} = 0.5$. Comment on this boundary.

The decision boundary is the set of $\mathbf{x}$ values for which $p(C = C_1|\mathbf{x}) = p(C = C_2|\mathbf{x}) : g(\mathbf{x}) = 0$.

Noting that $\theta_{ij} = p(x_i = 1|C = C_j)$, the discriminant function is:

$$g(\mathbf{x}) = \log \frac{p(\mathbf{x}|C = C_1)p(C_1)}{p(\mathbf{x}|C = C_2)p(C_2)} \tag{22}$$

$$= \log \frac{\theta_1}{1 - \theta_1} + \sum_{j=1}^{D} \log \frac{p(x_j|C = C_1)}{p(x_j|C = C_2)} \tag{23}$$

$$= (x_1 + x_2) \log(1.6) + (2 - x_1 - x_2) \log(0.4) \tag{24}$$

Equating to zero and solving for x_2 in terms of x_1 enables us to plot the decision boundary, which is displayed in Figure 1 (shown overleaf). Only when $(x_1 = 1, x_2 = 1)$ is class $C = C_1 = 1$ predicted. The decision is unaffected by the value of x_3, as is shown by the absence of a component in x_3 for the boundary's surface normal vector. x_1 and x_2 are interchangable within the decision function.

[6]less consistent in the sense that the perturbation degraded class separation across the decision boundary.
[7]e.g. feature x_1 indicates that $C = C_1$ if $\frac{p_{11}}{p_{12}} > 1$ and $C = C_2$ if $\frac{p_{11}}{p_{12}} < 1$.

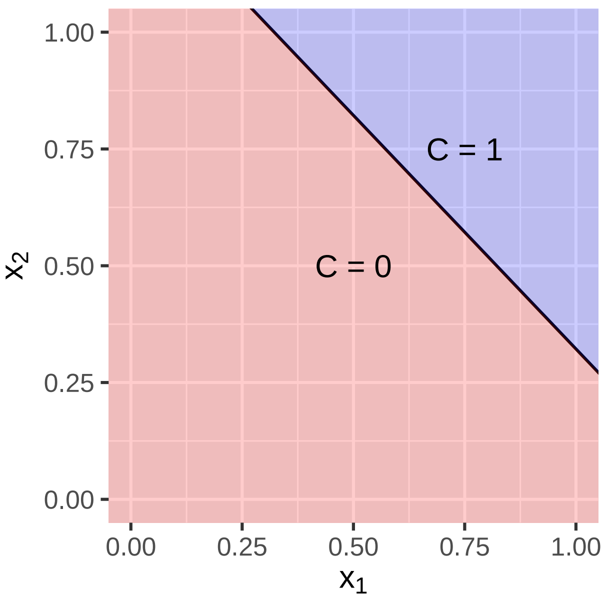

Figure 1: (Question 5) The decision boundary $0 = (x_1 + x_2)\log(1.6) + (2 - x_1 - x_2)\log(0.4)$. Note that the boundary's surface normal vector has no component in x_3, which is why the boundary is only presented in two dimensions.

References

1 K. Murphy, *Machine Learning: A Probabilistic Perspective*. The MIT Press, Cambridge, Massachussets, 2012, 0262018020, pp. 82-89.

Comment

The page limit (6 pages) was exceeded as a consequence of including the questions for clarity.