CMPUT 466/551 — Assignment 1

Instructor: R Greiner
Due Date: 12:30pm, Tuesday, 6 Oct 2009

The following exercises are intended to further your understanding of probability, Bayesian decision theory, linear regression and classifiers (LMS, logistic regression, LDA).
Relevant reading: [HTF Chapters 1,2,3,4]

Total points: Grad: 100; Undergrads: 80
% of course marks 20%

Everyone — both grads and undergrads – should do Questions 1–7 and 9. Only grads need to answer Question8 as well.

To submit your solutions: For Question 1–7 (and 8): submit in paper form to the course assignment drop box. Please ensure that your name, login, and student ID are written on your assignment.

To will submit your solution to Question 9: First zip your code `LMS.m`, `LR.m`, `LDA.m`, `9d.png`, and `9e.mat` into a single file, then email that file to `c466-submit@cs.ualberta.ca`. (Be sure to watch the newsgroup for possible further directions.)

Question 1 *[8 points] Using Bayes Rule — "Finger Exercises"*
Suppose that we have three coloured urns that contain fruit:

Urn (color)	#apples	#oranges	#limes
r (red)	3	4	3
b (blue)	1	1	0
g (green)	3	3	4

You select an urn at random, with probabilities $p(r) = 0.2$, $p(b) = 0.2$ and $p(g) = 0.6$, and draw a piece of fruit at random from that urn (with equal probability of selecting any item in that urn).

a [2]: What is the probability of selecting an apple?

b [3]: If you observed that the fruit is an orange, what is the probability that it came from the green box?

c [3]: *Paradox?*
Harry and Ron each practice casting spells for two sessions. In the first session, over 100 attempts, Harry's spells work correctly 60 times, while Ron are correct 90 times. In the second session, Harry is accurate just 1 time in 10, while Ron's are correct 300 times in 1000. What are their respective accuracies when considering *both sessions together*? Does this seem strange to you? (Note Harry is less accurate than Ron in both individual sessions — session1: $0.6<0.9$; and session2: $0.1<0.3$.)

Question 2 *[7 points] Probability Decision Boundary*
Assume you have learned the conditional probability distribution $P(\, y \,|\, \mathbf{x}\,)$ for a binary classification problem; let $p_A = P(\, y = A \,|\, \mathbf{x}\,)$. Consider the following loss matrix:

		true label y	
		A	B
predicted label $\hat{y}$	A	0	c_B
	B	c_A	0

(That is, c_B is the penalty for not saying B when you should — *i.e.*, for predicting the value is A when the correct label is B.)

a [5]: Show that the decision $\hat{y}$ that minimizes the expected loss is equivalent to setting a probability threshold θ and predicting $\hat{y} = A$ if $P(\, y = A \,|\, \mathbf{x}\,) > \theta$ and $\hat{y} = B$ otherwise. If possible, write θ as a function of c_A and c_B.

b [1]: What is the threshold for the loss matrix, if $c_B = 20$ and $c_A = 5$?

c [1]: Show a loss matrix where the threshold is 0.1.

Question 3 *[7 points] Reject Option*
Suppose any misclassification costs \$10, but a human can manually classify an object for only \$3. Here, it makes sense to allow the classifier to "reject" a test example (*i.e.*, hand it off to a person to make the call), rather than classifying it into one of the classes.

We can formulate this as the following loss matrix:

		true label y	
		A	B
decision	predict $\hat{y} = A$	0	c_B
	predict $\hat{y} = B$	c_A	0
	reject	c_r	c_r

where $c_A = c_B = 10$ and $c_r = 3$.

a [1]: Suppose $p_A = P(\, y = A \,|\, \mathbf{x}\,) = 0.2$. Which decision minimizes the expected loss?

b [1]: Which decision minimizes the expected loss if $P(\, y = B \,|\, \mathbf{x}\,) = 0.4$?

c [4]: Now consider general (strictly-positive) values of c_A, c_B and c_r, which might be different from one another. Show that, in general, there will be two thresholds θ_A and θ_B such that the optimal decision is to predict A if $P_A = P(\, y = A \,|\, \mathbf{x}\,) > \theta_A$, *reject* if $\theta_A > P(\, y = A \,|\, \mathbf{x}\,) > \theta_B$, and predict B if $P(\, y = A \,|\, \mathbf{x}\,) < \theta_B$. If possible, write θ_A and θ_B as functions of c_A, c_B and c_r.

d [1]: What are the values of these thresholds for the loss matrix shown above, with $c_A = c_B = 10$ and $c_r = 3$?

e [1]: What are the θ_A and θ_B values when $c_B = 20$, $c_A = 5$ and $c_r = 2$.

Question 4 *[3 points] Mean, Variance*
The uniform distribution for a continuous variable x is defined by

$$U(\, x \,|\, a, b\,) \quad = \quad \frac{1}{b-a}, \qquad a \le x \le b$$

Verify that this distribution is normalized, and find expressions for its mean and variance. You may use $\mathrm{var}[x] = E[x^2] - E[x]^2$.

Question 5 *[7 points]* For any loss function $L(\cdot,\cdot)$ [HTF Section 2.4], the "expected prediction error" (EPE) of the function $f(x)$ is

$$\mathrm{EPE}(f(x)) = \int\int L(y, f(x))\; p(x,y)\, dx\, dy$$

which means, for each specific x_0,

$$\mathrm{EPE}(f(x_0)) = \int L(y, f(x_0))\; p(y|x_0)\, dy$$

a [3]: When using squared loss $L(y, f(x)) = (y - f(x))^2$, prove that minimizing EPE leads to the conditional mean:

$$\hat{f}_{(L2)}(x_0) = \operatorname*{argmin}_{y_0} \mathrm{EPE}(y_0) = E[Y|X = x_0] \ .$$

b [4]: When using absolute loss $L(y, f(x)) = |y - f(x)|$, prove that minimizing EPE leads to the conditional median:

$$\hat{f}_{(L1)}(x_0) = \operatorname*{argmin}_{y_0} \mathrm{EPE}(y_0) = \mathrm{Median}(Y|X = x_0) \ .$$

Note c is a median for a probability density function (pdf) $p(z)$ when c divides the area under the pdf into two equal halves – *i.e.*, when $\int_{-\infty}^{c} p(z)\, dz = \int_{c}^{+\infty} p(z)\, dz$.

Question 6 *[13 points] MLE of Variance is Biased*
For each $n = 1..N$, let $\mathbf{x}_n \sim \mathcal{N}(\mu, \Sigma)$ denote an instance drawn (independently) from a Gaussian distriubtion with mean μ and convariance Σ. Recall $\mu_{ML} = \frac{1}{N}\sum_{m=1}^{N} \mathbf{x}_m$, and

$$E[\Sigma_{ML}] = E\left[\frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \mu_{ML})(\mathbf{x}_n - \mu_{ML})^T\right] \ .$$

Show that $E[\Sigma_{ML}] = \frac{N-1}{N}\Sigma$. You may want to prove, then use,

$$E\left[\mathbf{x}_n \mathbf{x}_m^T\right] = \mu\mu^T + \delta_{n,m}\mathbf{\Sigma} \tag{1}$$

where $\delta_{n,m} = 1$ if $m = n$ and $\ldots = 0$ otherwise.

Question 7 *[5 points] Weighted squared error*
In our derivation of the LMS algorithm, we used the squared error to approximate the 0/1 loss. Suppose that we have a general loss matrix, with the cost of a false positive being $L(1,0) = c_0$ and the cost of a false negative $L(0,1) = c_1$. Here, we seek the weight vector $\mathbf{w}$ that minimizes the following function:

$$J(\mathbf{w}) = \sum_{t=1}^{m} \frac{1}{2}\, c_{y_t}\, [\mathbf{w}\cdot\mathbf{x}_t - y_t]^2$$

(There is typically an additional factor of $1/m$, which we dropped to simplify the computation.) Compute the gradient based on this function, and show how the Batch LMS algorithm is modified to incorporate this change.

Question 8 *[20 points] Bias-Variance for Classifiers* **[ONLY GRADS!]**
In class, we explored the bias-variance tradeoff in the context of *regressors*. This question explores this tradeoff in the context of *classifiers* — here binary classifiers, $Y = \{0, 1\}$.

Following the lecture notes, we have an observation $\mathbf{x}$, with label t. (Note t could be stochastic — *i.e.*, different t's for same $\mathbf{x}$.) Given instance $\mathbf{x}$, we predict $y_d(\mathbf{x}) \in \{0, 1\}$, based on the function $y_d() = \text{Learning}(d)$, which depends on the training dataset $d \in D$. In class, we considered the loss function: $L_2(y, y') = (y - y')^2$; here, we instead consider

$$L(y, y') = L_{0/1}(y, y') \quad = \quad \begin{cases} 0 & \text{if } y = y' \\ 1 & \text{otherwise} \end{cases}$$

Using this function, define

$$y^*(\mathbf{x}) \quad = \quad \underset{y}{\text{argmin}}\, E_t\,[\,L(t, y)\,|\,\mathbf{x}\,] \quad = \quad \underset{y \in \{0,1\}}{\text{argmin}}\{P(\,t = y\,|\,\mathbf{x}\,)\}$$

to be the value $y \in \{0, 1\}$ that minimizes the loss. (Note a Bayes optimal classifier would return $y^*(\mathbf{x})$ for every $\mathbf{x}$.)

Let D be the distribution over datasets; then

$$\hat{y}_D(\mathbf{x}) \;=\; \underset{y}{\text{argmin}}\, E_{d \in D}\,[\,L(y_d(\mathbf{x}),\, y)\,]$$

is the value that minimizes this 0/1-loss, on average over the possible datasets.

Now define

$$\begin{array}{lll} \text{“Noise”:} & N(\mathbf{x}) \;=\; E_t\,[\,L(t,\; y^*(\mathbf{x}))\,|\,\mathbf{x}\,] \;=\; P(\,t \neq y^*(\mathbf{x})\,|\,\mathbf{x}\,) \\ \text{“Bias”:} & B(\mathbf{x}) \;=\; L(\,y^*(\mathbf{x}),\; \hat{y}_D(\mathbf{x})\,) \\ \text{“Variance”:} & V(\mathbf{x}) \;=\; E_{d \in D}\,[\,L(\hat{y}_D(\mathbf{x}),\; y_d(\mathbf{x}))\,] \end{array}$$

Show that

$$E_{D,t}\,[\,L(t, y_d(\mathbf{x}))\,|\,\mathbf{x}\,] \quad = \quad [2\,P_D(y_d(\mathbf{x}) = y^*(\mathbf{x})) \;-\; 1] \times N(\mathbf{x}) \;+\; B(\mathbf{x}) \;+\; (-1)^{B(\mathbf{x})} V(\mathbf{x})$$

[Hint: Prove that
1. *$L(t, y) = L(y^*, y) + c \times L(t, y^*)$ for some value of c that might depend on the values t, y, y^*;*
2. *$L(y^*, y) = L(y^*, \hat{y}_D) + c' \times L(\hat{y}_D, y)$, for some value of c' that might depend on y^*, y $\hat{y}_D$;*
3. *Then consider the distribution.]*

Question 9 *[30 points] Programming: Linear Threshold Unit*
In class, we discussed a number of ways to learn the parameters for a linear threshold unit. For this assignment, you will implement these learning algorithms in MATLAB, then evaluate them on some given datasets.

You should be familiar with `load`, `save`, `csvread`, `ops`, `slash`, `find`, `rand`, `randn`, `function`, `plot`, and `print`. Type `help <topic>` in MATLAB if you need to learn about

any of these. You ***may not*** use the built in regression functions or toolbox classification features for this assignment. Ensure that your algorithm names are the proper case, since MATLAB is case sensitive, and we will be testing your algorithms automatically.

Many of these algorithms involve setting parameters (*e.g.*, learning rates) and making decisions (batch or on-line); you should experiment here to find the appropriate settings. Your goal is to minimize the number of misclassifications – *i.e.*, 0/1 loss function.

Your algorithms will be tested with the datasets available from the course website; there will be a README (html) file that describes the process of loading the data into MATLAB. We will provide 5 datasets. Your algorithms should run correctly on all 5, and the last 2 will be used for parts **d** and **e**. You should ensure that your algorithms run in a reasonable time — one minute per algorithm per dataset (of under 1000 data points).

The training data will have the form

$$X = \begin{bmatrix} 1 & x_{1,1} & \dots & x_{1,n-1} \\ \vdots & \vdots & & \vdots \\ 1 & x_{t,1} & \dots & x_{t,n-1} \end{bmatrix} \qquad y = \begin{bmatrix} y_1 \\ \vdots \\ y_t \end{bmatrix}$$

where $n-1$ is the dimension of the (original) data, t is the training set size, and $y_i \in \{0, 1\}$. The far-left column of 1's is to allow the bias term to be the first element of your weight vector `w`. Each of your MATLAB functions should return a $n \times 1$ weight-value $w = \langle w_0, w_1, ..., w_{n-1} \rangle$. Given a new input $x = \langle x_1, ..., x_{n-1} \rangle$, we will compute $w_0 + \sum_{i=1}^{n-1} x_i \times w_i$, then return 1 if this is > 0 and 0 otherwise.

Each of your MATLAB functions should run autonomously, the same for each dataset. So if you want to the learning rate (or whatever) to depend on the input dataset, your code should compute this. Note this can involve some magic constants, if you wish. Your *program* can change these values from one dataset to another, but YOU may not!
(Of course, you should include name and id as a comment at the begining of each of your programs.)

Please view the `nfoldCV` subroutine provided as just a template... this code is too simplistic: Eg, you need to use BALANCED folds.

a [5]: Write a MATLAB function `LMS(X,y)` that takes a $t \times n$ matrix `X` and a $t \times 1$ vector of target values `y` and returns the weights `w` corresponding to the minimum mean-squared error linear function. Electronically submit this function in the file `LMS.m`.
(Given that the underlying regression process is seeking weights w such that $w \times x_i$ is close to $y_i \in \{0, 1\}$, the obvious threshold for distinguishing positive from negative would be 1/2 — *i.e.*, we would return "+" if $x \times w > 1/2$ and "−" otherwise. You may therefore want to shift your weights w to move the threshold to be 0. *Hint: this can be done in a very simple post-processing step.*)

b [10]: Write a MATLAB function `LR(X,y)` that returns the weights `w` derived using logistic regression to maximize conditional probability. Electronically submit this function in the file `LR.m`.

c [5]: Write a MATLAB function `LDA(X,y)` that returns the weights `w` derived using linear discriminant analysis. Electronically submit this function in the file `LDA.m`.

d [5]: Create a plot showing the 5-fold cross validation "learning curve" for each algorithm on the `LearningCurve` dataset. This will show how your estimate improves (or not) as you provide more and more data. Show the results for 10, 50, 100, 200, 500, and 1000 instances. Each point on your curves should be based on at least 10 different sub-datasets of the specified size, drawn randomly from the original dataset *WITH REPLACEMENT*. Hence, any sub-dataset may include an element more than once. Moreover, these different sub-datasets will probably *not* be disjoint. Also, for each dataset (or datasubset), you should compute the 5-fold CV score, wrt only that data(sub)set. Please include 95% confidence error-bars around each point. (Construct a t-confidence interval using the number of sub-datasets you chose. You may want to use Matlab's `errorbar` function.) Create a PNG file from your plot using the command "`print -dpng 9d`" and electronically submit the file `9d.png`. You should draw one graph for each of the 3 algorithms, named `9d-LMS.png`, `9d-LR.png`, `9d-LDA.png`. For each, let the x-axis range over the number of points used, and the y-axis be the 5-fold cross-validation error.

e [5]: Bake-off: Find the best weight vector `w` for the `CHALLENGE` dataset, using any algorithm you wish, with whatever parameters you think should work best. Use the command "`save 9e w`" to save the vector and electronically submit the file `9e.mat`. Your weight vector will be tested on unseen instances of the dataset and compared with the results with the rest of the class.