Final Exam

Advanced Methods for Data Analysis (36-402/36-608)

Due Thursday May 8, 2014 at 11:59pm

Instructions: you will submit this take-home final exam in three parts.

1. *Writeup.* This will be a complete writeup, in full data analysis report format, of what you have done. More details on the format to follow. And, as with your homework assignments, your submitted writeup must be in PDF format. You can produce this PDF using LaTeX, Word, or whatever you want, but at the end of the day, it must be a PDF. Any other type of file will not be accepted.

   Make sure that the extension here is .pdf.

2. *Code.* Also submit your R code for the homework as a plain text file. You should demarcate sections of the code that correspond to different tasks using clearly visible comments (e.g., as in `##### Cross-validation #####`).

   Make sure that the extension here is .txt.

3. *Data set.* Each of you will be emailed a slightly different data set, and for completeness you will submit this (Rdata) file as well.

   Make sure that the extension here is .Rdata.

You must not communicate at all with your classmates, your friends, or anybody other than the Professor and the TAs about this take-home final exam, during the week you have to complete it. Evidence of illegal communication will be taken very seriously.

You are of course allowed to consult your class notes, your homework assignments, or any of the other course materials in order to complete this exam. You may use the internet as a resource but you may not use it to communicate with anybody else or ask questions about your assignment. Everything you write must be in your own words, and you must explicitly cite any sources used other than the class materials.

The following describes the scientific problem and statistical questions that you must address, as well as the format of your writeup.

1 Cancer classification

This data set examines 14 types of cancer, numbered as: 1–breast, 2–prostate, 3–lung, 4–collerectal, 5–lymphoma, 6–bladder, 7–melanoma, 8–uterus, 9–leukemia, 10–renal, 11–pancreas, 12–ovary, 13–meso, 14–cns. You will predict the cancer type of a given patient given his or her gene expression measurements. This data is available on the website for the Elements of Statistical Learning textbook, and was originally taken from a paper by Ramaswamy et al. (2001), "Multiclass cancer diagnosis using tumor gene expression signatures". You have access to 16,063 gene expression measurements per patient, which is a lot! Note that the outcome here, cancer type, is discrete and unordered, taking values between 1 and 14, which makes this a classification problem with $K = 14$ classes.

You should have been emailed an R data file `cancerxxx.Rdata` (where `xxx` stands for a number between `001` and `100`). When loaded into your current R session, this gives you 4 objects: `xtrain`, `ytrain`, `xtest`, `ytest`. These are, in respective order: the gene expression measurements for the training set, the cancer types for the training set, the gene expression measurements for the test set, and the cancer types for the test set. There are 144 patients in the training set, and 54 in the test set. Therefore the dimensions are `xtrain`: $144 \times 16,063$, `ytrain`: 144×1, `xtest`: $54 \times 16,063$, `ytest`: 54×1. In other words, using the typical language and notation in data analysis, we have $p = 16,063$ predictor variables, $n_{\text{train}} = 144$ training observations, and $n_{\text{test}} = 54$ observations.

2 Statistical tasks, data analysis report

You will submit a data analysis report, just like you did with the last midterm take-home exam. As with this last exam, the data analysis report can be a *maximum of 5 pages*, and must abide by the section structure described below. You should clearly lay out what you have done, using figures to supplement your explanation. Your figures must be of proper (large) size with labeled, readable axes. You should not mindlessly paste raw R output into your writeup with 12 significant digits, etc. You can include R snippets at your discretion, if you think that will help your explanation. In general, you should take pride in making your report readable and clear. You will be graded both on statistical content and quality of presentation.

Section 1: Introduction

The introduction should be brief (1 or 2 paragraphs) but must properly describe the data set and motivate the problem. You cannot copy motivation text verbatim from this document or anywhere else for that matter. Everything you write must be in your own words (and as mentioned previously, this obviously holds for the entire report). You do not need to perform basic exploratory data analysis like you learned in 401, and in fact if you do so here, you will be wasting space. Examples of basic points to cover: where does the data set come from? How many genes (i.e., the expression levels of how many genes) have been measured? How many patients are there in the training set, and how many in the test set? What are the patient outcomes (types of cancer)? How many of each type of cancer is represented in the training set, test set? What is the ultimate statistical goal (cancer classification)? What about its nature makes the goal a statistically challenging problem?

Section 2: Variable screening

Between this section and the next, we will come up with a relatively naive predictive model based on variable screening and linear regression. The first thing you will do is to reduce the number of variables in consideration from 16,063 to ≈ 50. You will do this using a technique called *variable screening*. Suppose that you are given the gene expression measurements from gene j, across the n patients, which we will denote by $x_{1j}, x_{2j}, \ldots x_{nj} \in \mathbb{R}$. Recall that each of these patients falls into one of K cancer types, denoted by the class coding $y_1, \ldots y_n \in \{1, \ldots K\}$. Then the *between-class variation of gene j* is defined as

$$V_j = \sum_{k=1}^{K} n_k \cdot (\bar{x}_{kj} - \bar{x}_j)^2,$$

where n_k is the number of patients in class k (cancer type k), $\bar{x}_{kj}$ is the mean of patients in class k along gene j, i.e.,

$$\bar{x}_{kj} = \frac{1}{n_k} \sum_{i\,:\,y_i=k} x_{ij},$$

and $\bar{x}_j$ is the overall mean of patients along gene j, i.e.,

$$\bar{x}_j = \frac{1}{n}\sum_{i=1}^{n} x_{ij}.$$

On the training data, compute the between-class variance for each gene $j = 1, \dots p$. That is, to be perfectly clear, this should produce p separate between-class variance scores. Select the 50 genes that have the highest between-class variance scores, and save their identity along with the scores. Then, portray this information by producing a plot, with the numbers 1 through 50 on the x-axis, and the highest through the 50th highest between-class variance scores on the y-axis. We will refer to this smaller set of 50 variables as the *screened set* of variables.

In your screened set of variables, there will turn out to be pairs of variables that are perfectly correlated. You can visually spot this by looking at the trend of (ordered, decreasing) between-class variance scores in the plot: there will be occasional neighboring pairs of points with the exact same height. Identify the pairs of perfectly correlated variables (hint: the R `diff` command, applied to the ordered between-class variance scores). Further reduce the screened set by throwing out one of each pair of variables with perfect correlation. How many variables are left over? And what is the highest absolute correlation between pairs of variables in the new screened set?

Section 3: Linear regression of indicators

Here we will construct a linear regression based classifier using the screened set of variables from the last section (this is the final screened set, after throwing out variables with perfect correlation). Again, we will only use data from the training set. First consider predicting class 1 versus all other classes $2, \dots 14$, and (temporarily) recoding the class labels as $\tilde{y}_1^{(1)}, \dots \tilde{y}_n^{(1)}$, where

$$\tilde{y}_i^{(1)} = \begin{cases} 1 & \text{if } y_i = 1 \\ 0 & \text{if } y_i \neq 1 \end{cases}, \quad i = 1, \dots n.$$

We can now use a linear regression of $\tilde{y}^{(1)}$ on the screened set of variables, and let $\hat{\beta}^{(1)}$ denote the coefficients from this regression. Next consider predicting class 2 versus all others $1, 3, \dots K$. Similarly recode the class labels as $\tilde{y}_1^{(2)}, \dots \tilde{y}_n^{(2)}$ (where each is 1 or 0 depending on class 2 membership), fit a linear regression of $\tilde{y}^{(2)}$ on the screened set of variables, and let $\hat{\beta}^{(2)}$ denote the coefficients. After doing this K times, each time considering a different class versus all the rest, we should have K sets of coefficients $\hat{\beta}^{(1)}, \dots \hat{\beta}^{(K)}$. Now suppose that we are given a new observation and let x_0 denote the measurements of the screened set of variables for this observation. We predict the class of this new observation by looking at the fitted values from each of the K linear regressions, and choosing the highest:

$$\hat{y}_0 = \underset{k=1,\dots K}{\operatorname{argmax}} \; x_0^T \hat{\beta}^{(k)}.$$

This technique is called *linear regression of indicators*. It is not the most sophisticated technique, but it will give us a basic cancer classification rule. You will fit the linear regression of indicators model on the training set. What is the training error, i.e., how many training misclassifications are made? Also, why did we need to run this on the screened set of variables—why couldn't we just have used the full set?

As you (should) well know by now, training error is not at all a good measure of predictive performance. Still working entirely with the training data, use 4-fold cross-validation to estimate the test error of the linear regression of indicators model. How many cross-validation misclassifications are made? Is this substantially different from the number of training misclassifications?

An important note: for the cross-validation part, you should take care to "balance the folds" so that they contain a number of observations from each class that is proportional to the numbers on the full set. E.g., if the full training set has 8 observations from each of classes 1–4, and 16 from class 5, then each of the 4 folds should contain 2 observations from each of classes 1–4, and 4 from class 5. Probably the easiest way to do this programmatically is to work with the observations from each class separately when assigning tolds: for each class $k = 1, \dots K$, randomly divide the observations from this class into 4 folds and save their folds labels (e.g., $1, 4, 2, 4, 3, 1, 1, \dots$). Then just amalgamate the labels at the end and you will have assignments of all observations to folds in a way that is balanced. To verify that you've properly done this, report the class proportions within each fold.

If you cannot figure out how to make balanced folds, then just make unbalanced folds, i.e., just create folds in the way that you're used to doing. This is certainly better than nothing and you will still get substantial partial credit.

Section 4: Lasso multinomial regression

We will switch gears and consider a more sophisticated classification technique: *lasso multinomial regression*. Luckily there is readily available software for this technique in the `glmnet` R package. Consult the last section of the lecture notes on logistic regression for a recap of multinomial regression; in this model, with K classes, we fit $K-1$ coefficient vectors $\hat{\beta}^{(1)}, \dots \hat{\beta}^{(K-1)}$. In the lasso penalized version of multinomial regression, we fit these coefficients in such a way that they turn out to be sparse (i.e., each vector $\hat{\beta}^{(k)}$ has many zero components, for each $k = 1, \dots K-1$). Note that in our context, this corresponds to selecting out a handful of the 16,063 genes to make a classification rule (these are the ones with nonzero components in the coefficient vectors). As you learned in class, there is a tuning parameter λ associated with the lasso method that controls the number of zero components in the fitted coefficients, with larger λ meaning more zero components, and smaller λ many fewer zero components.

Use the `cv.glmnet` in the `glmnet` to fit a multinomial regression model with lasso penalty. As for the choice of λ, it is best if you don't prespecify a list of λ values and this function will choose 100 values for you, and compute the coefficient estimates at each of them. Note: you will want to set `alpha=1` and `family="multinomial"` as arguments in the function. The former instructs it to use the lasso penalty (`alpha=0` for the ridge penalty), and the latter instructs it to use multinomial regression for classification. Also, as suggested by its name, this function `cv.glmnet` will also perform cross-validation at each value of λ for us. For example, you can call:

```
las.cv = cv.glmnet(xtrain,ytrain,alpha=1,family="multinomial",foldid=bal.folds)
las.mod = las.cv$glmnet.fit
```

The first line runs 4-fold cross-validation for the lasso multinomial regression model, over a grid of 100 values of λ chosen internally by the function. Note that the last argument `foldid` here specifies the fold assignments that `cv.glmnet` should use, assuming that `bal.folds` were the balanced fold assignments you created previously in Section 3. The second line grabs the `glmnet` object that is the result of fitting the lasso multinomial regression model to the full training set (i.e., not to partial sets made up of 3 folds), at each of the 100 values of λ.

By inspecting the `las.cv` object (and reading relevant documentation if necessary), plot the CV error for lasso multinomial regression as a function of λ. Note that it's perfectly fine to use the default plotting function provided by the `glmnet` package for this (you shouldn't have to do this yourself!). Also note that the default of `cv.glmnet` is to consider multinomial deviance as the CV error measure, and not misclassification rate, and this is fine. What is the value $\lambda_{\min}$ selected by the usual rule? What is the value $\lambda_{1\text{se}}$ selected by the one standard error rule? And, for each rule, what are the number of cross-validation misclassifications incurred? (Hint: there are many ways to do this, but probably the easiest is just to rerun `cv.glmnet` with the option `type.measure="class"`). How does this compare to the performance of the linear regression of indicators method?

How many variables in total are used in the model chosen by the usual rule? And how many in the model chosen by the one standard error rule? To answer these questions, you will examine the fitted coefficients, at each of $\lambda_{\min}$, $\lambda_{1\text{se}}$, stored in the `las.mod` object. (Recall these are the coefficients when the models are fit to the full training set.) For the two values $\lambda_{\min}$, $\lambda_{1\text{se}}$, each one of the coefficient vectors $\hat{\beta}^{(1)}, \ldots \hat{\beta}^{(K-1)}$ vectors will have a select number of nonzero components, and you want to figure out the total number of components that have a nonzero entry in at least one coefficient vector. (Hint: the `unique` function in R.) Comment on whether or not this is a big reduction from the full set of 16,063 genes. What would this reduction mean practically to a cancer scientist?

Finally, consider the identities of the screened set of variables from Section 1 (after removing perfectly correlated variables), and the set of variables used in the model chosen by the one standard error rule. How many do they have in common? (Hint: the `%in%` operator in R.) What does this tell you about the effectiveness of variable screening as a tool for variable selection?

Section 5: Test set evaluations

Lastly, we will look at the test set. (All statistical inquiries up until now have used the training set exclusively.) Consider the linear regression of indicators model, and the lasso multinomial regression model chosen by the one standard error rule. Use each of these to predict the class (cancer type) of patients in the test set. What are the number of misclassifications committed by each model? Which is better in this regard, and is there a big difference? Are the misclassification rates (i.e., the proportions of observations misclassified) roughly consistent with the CV misclassification rates for these methods, that you computed previously? In the case that there are differences between the test error rates and the CV error rates, make an attempt to explain why.

Section 6: Discussion

The discussion should be brief (1 or 2 paragraphs). Summarize your analysis, in the context of the problem (cancer classification). Use this room to reflect on the results that you view as successes, and the results that raise suspicions in your mind. You can also provide ideas for what you might consider in the future, in this data setting.