STATISTICAL MODELLING Part IIC, Michaelmas 2024

Practical 8: Contingency tables and Cross Validation

Contingency tables

Fisman et al. conducted a study on dating behaviour using data from a Speed Dating experiment at Columbia University. Pairs of men and women students interacted for four minutes and then each filled out a form which recorded whether or not they wanted to receive the other's email address, and also various details about themselves including their chosen subject of study, what motivated them to sign up for the speed dating experiment, how often they go out and how often then go on dates. If both individuals wanted each others email addresses, it is considered a match. The original data are available at `http://andrewgelman.com/2008/01/21/the_speeddating_1/`. Download an edited version of the data from the course webpage with

```
> path <- "http://www.statslab.cam.ac.uk/~rds37/teaching/statistical_modelling/"
> SD_data <- read.csv(file.path(path, "SD_match.csv"), stringsAsFactors=T)
> SD_data[1:3, ]
  match subject_m   goal_m     date_m go_out_m subject_f   goal_f       date_f go_out_f
1     0      Econ      fun several/yr > 2/week       Law meet ppl almost never > 2/week
2     0      Econ      fun    1/month  2/month       Law meet ppl almost never > 2/week
3     1      Econ     date   > 2/week > 2/week       Law meet ppl almost never > 2/week
```

The first row records the meeting of a male Economics student and a female Law student, which did not result in a match (`match` is 0). The goals of the man and woman were to have fun and to meet people, respectively. We also have the frequencies with which the individuals go out and go out on dates. Let us first focus on the relationship between match and the subjects of the individuals.

```
> SD_subj <- table(SD_data[, c("subject_m", "subject_f", "match")])
> SD_subj
```

The `table` function converts the data into contingency table format (3-way in this case). The order in which the factors are given matters when we call `SD_subj` so some combinations will be easier to interpret; e.g., try

```
> table(SD_data[, c("match", "subject_m", "subject_f")])
```

In order to fit models to the data, we apply `as.data.frame` to the contingency table `SD_subj`. This produces a data frame where each each row gives the number of original observations (`Freq`) that fall into each of the possible categories given by each pair of subject types and match category.

```
> SD_subj_df <- as.data.frame( SD_subj )
> SD_subj_df
         subject_m       subject_f match Freq
1  Arts+Humanities Arts+Humanities     0  357
2             Econ Arts+Humanities     0  683
3              Law Arts+Humanities     0  126
4         Sciences Arts+Humanities     0  518
```

The `xtabs` function gives a contingency table for data frames (the other affects the output again):

```
> xtabs(Freq ~ subject_m + subject_f + match, data=SD_subj_df)
```

Since the numbers of Law students are fairly low, we could consider combining them with the Economics students. One way of doing this is as follows (the `vcd` package has a dedicated function to do this, but we will not use this here). Note that typically we wouldn't modify the actual data object `SD_data` but create a copy and then modify the copy.

```
> levels(SD_data$subject_m)
[1] "Arts+Humanities" "Econ"            "Law"             "Sciences"
> levels(SD_data$subject_m) <- c("Arts+Humanities", "Econ+Law", "Econ+Law", "Sciences")
> levels(SD_data$subject_f) <- c("Arts+Humanities", "Econ+Law", "Econ+Law", "Sciences")
```

(note that the `read.csv` function really does need the `StringsAsFactors` argument as otherwise this level merging would give an error). Now we create the contingency table again, and coerce it to a Data Frame, which will allow us to fit a GLM.

```
> SD_subj <- table(SD_data[, c("match", "subject_m", "subject_f")])
> SD_subj <- as.data.frame(xtabs(Freq ~ subject_m + subject_f + match, data=SD_subj))
```

Cross validation

Recall that during Practical 4 we studied a high dimensional model which only depended on a few of the parameters. Specifically, we examined models for the data generated via:

$$Y_i = 50(X_i - 0.1)(X_i - 0.7)(X_i - 1) + \epsilon_i \tag{1}$$

with $X_i \overset{iid}{\sim} U([0,1])$ and $\epsilon_i \overset{iid}{\sim} N(0,1)$. We avoided the problem of over-fitting a degree 10 polynomial by *regularising* the least squares minimisation problem. I.e., we added a term $\lambda\|\beta\|_2^2$ to the usual loss function $\|Y - X\beta\|_2^2$ and used the new minimiser over β as our estimate of the coefficients in the polynomial regression. For any particular $\lambda > 0$, we know that this minimiser has a closed form solution given by:

$$\hat{\beta}_\lambda = \underset{\beta \in \mathbb{R}^p}{\operatorname{argmin}} \|Y - X\beta\|_2^2 + \lambda\|\beta\|_2^2 \tag{2}$$

$$= (X^T X + \lambda I_p)^{-1} X^T Y. \tag{3}$$

where I_p is the identity matrix on p dimensions. Recall that we saw that the estimate, and thus the predictive performance, depend on this parameter λ chosen. If it is too big we *over-smooth* our estimated regression function, if it is too small we instead over-fit to the noise in the data. How should we choose λ in practice? We would like to select the λ that minimises the mean squared error $\|\tilde{Y} - \tilde{X}\hat{\beta}_\lambda\|_2^2$ for some new data point $(\tilde{X}, \tilde{Y})$. But how can we find this minimiser without being able to calculate this quantity to minimise?

The technique of **cross-validation** comes to the answer. Suppose that we consider the estimate $\hat{\beta}_{\lambda,-i}$ given the solution above with the i^{th} data point omitted from the design matrix and responses. We can use the performance of this estimator on this omitted data-point, i.e., $(Y_i - X_i\hat{\beta}_{\lambda,-i})^2$, as an unbiased estimate of the true mean squared error of $X\hat{\beta}_{\lambda,-i}$. Averaging these estimates over all $i \in \{1, \ldots, n\}$ leads to a (reasonable[1]) estimate of the mean squared error of $\hat{\beta}_\lambda$. This leads to the selection of λ via the minimisation of the **leave-one-out cross-validation error**:

$$\hat{\lambda}^{CV} = \underset{\lambda \in \mathbb{R}_{>0}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} (Y_i - X_i\hat{\beta}_{\lambda,-i})^2 \tag{4}$$

which we call the **leave-one-out cross validation** estimator of λ. In practice it can be computationally expensive to compute n estimators per λ, so instead we group the data into K **folds** of roughly equal numbers of datapoints. We then perform the same procedure: compute the estimator on the data excluding one of the folds and then compute its error on the fold not used in the estimation. Specifically, let I_k be disjoint subsets of $\{1, \ldots, n\}$ for $k = 1, \ldots, K$ and let $\hat{\beta}_{\lambda,-I_k}$ be the estimator computed with the data excluding the rows with indicies in I_k. Then we can select λ via:

$$\hat{\lambda}_K^{CV} = \underset{\lambda \in \mathbb{R}_{>0}}{\operatorname{argmin}} \sum_{k=1}^{K} \sum_{i \in I_k} (Y_i - X_i\hat{\beta}_{\lambda,-I_k})^2 \tag{5}$$

which we call the K**-fold cross validation** estimator of λ. We will fit cross validated regularised GLMs in the exercises using the package `glmnet`. This package can be installed and loaded using the following code:

[1] Note that these estimates are not quite comparable—one uses one more data point than the others. There is a lot of work required to fully justify that this estimate is reasonable but we omit this here.

```
install.packages("glmnet")
library(glmnet)
```

This package can handle generalised linear regression too (hence the GLM in the name), so can be applied in the logistic and Poisson regression examples we have exampled previously. It can also handle a more general class of regularisation penalties than just the ridge penalty $\lambda\|\beta\|_2^2$, called **elastic net penalties**:

$$\lambda\big(\alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2/2\big) \tag{6}$$

for $\alpha \in [0,1]$. When $\alpha = 0$ this just becomes the ridge penalty and when $\alpha = 1$ this becomes the **LASSO** penalty.

Exercises

1. Let us fit a simple independence model to the speed dating contingency table using a surrogate Poisson model.

   ```
   > mod1 <- glm(Freq ~., data=SD_subj, family=poisson)
   > mod1$dev
   [1] 63.78329
   ```

   The final line gives the deviance of the model. Which of the asymptotic results from lectures can be used for testing using this quantity and what is the limiting distribution? Explain and conduct the test at the 5% level.

2. Consider the following model to test the hypothesis that the joint distribution of subject pairs are the same for the matched and non-matched groups.

   ```
   > mod2 <- glm(Freq ~ subject_m*subject_f + match, data=SD_subj, family=poisson)
   > anova(mod1, mod2, test="LR")
   ```

   What does the `*` mean? (You may wish to look at the `?formula` documentation). Would you select this model over `mod1`? Does this model suggest rejecting the hypothesis above?

3. Download the data in the `prostate` dataset available here:

   ```
   website <- "https://web.stanford.edu/~hastie/ElemStatLearn/datasets/prostate.data"
   prostate <- read.table(website, header=T)
   ```

   This dataset contains a clinical indicator of prostate cancer, `lspa`, as well as several patient characteristics (age, weight) and levels of various antigens. There is also a column of `train` which helpfully splits this dataset into a set we will use for model building and a separate set we will use to evaluate this model. We will fit a ridge regression model to the training data using:

   ```
   train <- subset(prostate, train == TRUE)[,1:9]
   test <- subset(prostate, train == FALSE)[,1:9]
   X <- data.matrix(train[,1:8])
   Y <- train$lpsa
   prostate.ridge <- glmnet(X,Y,family = "gaussian",
                            alpha=0, lambda.min.ratio=1e-6,nlambda=1000)
   plot(prostate.ridge, xvar="lambda", label=TRUE)
   ```

   What does this plot show? Now let us select the cross validated ridge regressor using the `glmnet` package.

```
set.seed(2018) # for reproducible random partitions into V folds
prostate.cvridge <- cv.glmnet(X, Y, family = "gaussian", alpha=0,
                              lambda=prostate.ridge$lambda, nfolds=10)
prostate.cvridge$lambda.min # optimal regularisation parameter
plot(prostate.cvridge)

ind <- prostate.ridge$lambda==prostate.cvridge$lambda.min
round(prostate.ridge$a0[ind], 3) # intercept
round(prostate.ridge$beta[,ind], 3)
```

The GLM object (in `glmnet`) deliberately does not give us standard errors for these coefficients: why not?

4. Use the glm fitted in the previous question to compute the training and test errors for each value of λ. *Hint: the* `predict` *function works for these models too, and considers all* λ. Plot these errors against $\log(\lambda)$. How does the cross-validated choice of λ compare to the choice that minimises the test error?

5. Repeat this for the elastic net parameter $\alpha = 1$. What do you notice about the coefficient vs. $\log(\lambda)$ plot that did not occur for the ridge regression plot?