Pragmatic Programming Techniques

Tuesday, May 29, 2012

Predictive Analytics: Generalized Linear Regression

In the previous 2 posts, we have covered how to visualize input data to explore strong signals as well as how to prepare input data to a form that is situation for learning. In this and subsequent posts, I'll go through various machine learning techniques to build our predictive model.

1. Linear regression
2. Logistic regression
3. Linear and Logistic regression with regularization
4. Neural network
5. Support Vector Machine
6. Naive Bayes
7. Nearest Neighbor
8. Decision Tree
9. Random Forest
10. Gradient Boosted Trees

There are two general types of problems that we are interested in this discussion; Classification is about predicting a category (value that is discrete, finite with no ordering implied) while Regression is about predicting a numeric quantity (value is continuous, infinite with ordering).

For classification problem, we use the "iris" data set and predict its "species" from its "width" and "length" measures of sepals and petals. Here is how we setup our training and testing data.

```
> summary(iris)
  Sepal.Length       Sepal.Width        Petal.Length    Petal.Width
 Min.   :4.300000   Min.   :2.000000   Min.   :1.000   Min.   :0.100000
 1st Qu.:5.100000   1st Qu.:2.800000   1st Qu.:1.600   1st Qu.:0.300000
 Median :5.800000   Median :3.000000   Median :4.350   Median :1.300000
 Mean   :5.843333   Mean   :3.057333   Mean   :3.758   Mean   :1.199333
 3rd Qu.:6.400000   3rd Qu.:3.300000   3rd Qu.:5.100   3rd Qu.:1.800000
 Max.   :7.900000   Max.   :4.400000   Max.   :6.900   Max.   :2.500000
       Species
 setosa    :50
 versicolor:50
 virginica :50

> head(iris)
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
>
# Prepare training and testing data
> testidx <- which(1:length(iris[,1])%%5 == 0)
> iristrain <- iris[-testidx,]
> iristest <- iris[testidx,]
```

For regression problem, we use the "Prestige" data set (imported from the "car" R package) and predict the degree of "prestige" from a set of input variables such as "income", "education", "job types" ... etc. Here is how we setup our training and testing data.

```
> library(car)
> summary(Prestige)
   education            income             women
```

About Me

Ricky Ho

I am a software architect and consultant passionate in Distributed and parallel computing, Machine learning and Data mining, SaaS and Cloud computing.

View my complete profile

Popular Posts

MongoDb Architecture
NOSQL has become a very heated topic for large web-scale deployment where scalability and semi-structured data driven the DB requirement tow...

Designing algorithms for Map Reduce
Since the emerging of Hadoop implementation, I have been trying to morph existing algorithms from various areas into the map/reduce model. ...

NOSQL Patterns
Over the last couple years, we see an emerging data storage mechanism for storing large scale of data. These storage solution differs quite...

Couchbase Architecture
After receiving a lot of good feedback and comment on my last blog on MongoDb , I was encouraged to do another deep dive on another popular ...

Predictive Analytics: Overview and Data visualization
I plan to start a series of blog post on predictive analytics as there is an increasing demand on applying machine learning technique to ana...

Predictive Analytics: Generalized Linear Regression
In the previous 2 posts, we have covered how to visualize input data to explore strong signals as well as how to prepare input data to a fo...

BigTable Model with Cassandra and HBase
Recently in a number of "scalability discussion meeting", I've seen the following pattern coming up repeatedly ... To make you...

Blog Archive

```
 Min.   : 6.38000   Min.   :  611.000   Min.   : 0.00000
 1st Qu.: 8.44500   1st Qu.: 4106.000   1st Qu.: 3.59250
 Median :10.54000   Median : 5930.500   Median :13.60000
 Mean   :10.73804   Mean   : 6797.902   Mean   :28.97902
 3rd Qu.:12.64750   3rd Qu.: 8187.250   3rd Qu.:52.20250
 Max.   :15.97000   Max.   :25879.000   Max.   :97.51000
    prestige            census           type
 Min.   :14.80000   Min.   :1113.000   bc   :44
 1st Qu.:35.22500   1st Qu.:3120.500   prof:31
 Median :43.60000   Median :5135.000   wc   :23
 Mean   :46.83333   Mean   :5401.775   NA's: 4
 3rd Qu.:59.27500   3rd Qu.:8312.500
 Max.   :87.20000   Max.   :9517.000
> head(Prestige)
                    education income women prestige census type
gov.administrators      13.11  12351 11.16     68.8   1113 prof
general.managers        12.26  25879  4.02     69.1   1130 prof
accountants             12.77   9271 15.70     63.4   1171 prof
purchasing.officers     11.42   8865  9.11     56.8   1175 prof
chemists                14.62   8403 11.68     73.5   2111 prof
physicists              15.64  11030  5.13     77.6   2113 prof
> testidx <- which(1:nrow(Prestige)%%4==0)
> prestige_train <- Prestige[-testidx,]
> prestige_test <- Prestige[testidx,]
```

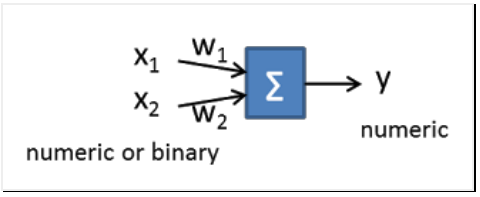
To measure the performance of our prediction, here we use some simple metrics to get through this basic ideas. For regression problem, we'll use the correlation between our prediction and the testing result. For classification problem, we'll use the contingency table to measure the count of true/false positive/negative in our prediction.

Now we have set the stage, lets get started.

Linear Regression

Linear regression has the longest history in statistics, well understood and is the most popular machine learning model. It is based on the assumption of a linear relationship exist between the input x1, x2 ... and output variable y (numeric).

$$y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + ...$$



The learning algorithm will learn the set of parameter such that the objective function: sum of square error $\sum(y_{actual} - y_{estimate})^2$ is minimized.

Here is the code examples in R.

```
> model <- lm(prestige~., data=prestige_train)
> summary(model)
Call:
lm(formula = prestige ~ ., data = prestige_train)

Residuals:
        Min          1Q      Median          3Q         Max
-13.9078951  -5.0335742   0.3158978   5.3830764  17.8851752

Coefficients:
                Estimate     Std. Error  t value     Pr(>|t|)
(Intercept) -20.7073113585  11.4213272697 -1.81304    0.0743733 .
education     4.2010288017   0.8290800388  5.06710 0.0000034862 ***
income        0.0011503739   0.0003510866  3.27661    0.0016769 **
women         0.0363017610   0.0400627159  0.90612    0.3681668
census        0.0018644881   0.0009913473  1.88076    0.0644172 .
typeprof     11.3129416488   7.3932217287  1.53018    0.1307520
typewc        1.9873305448   4.9579992452  0.40083    0.6898376
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 7.41604 on 66 degrees of freedom
  (4 observations deleted due to missingness)
Multiple R-squared: 0.820444,   Adjusted R-squared: 0.8041207
F-statistic: 50.26222 on 6 and 66 DF,  p-value: < 0.00000000000000022204

> # Use the model to predict the output of test data
> prediction <- predict(model, newdata=prestige_test)
```


- ► September (1)
- ► August (2)
- ► July (1)
- ► June (3)
- ▼ May (3)
  - Predictive Analytics: Generalized Linear Regressio...
  - Predictive Analytics: Data Preparation
  - Predictive Analytics: Overview and Data visualizat...
- ► April (3)
- ► March (1)
- ► February (1)
- ► January (1)

► 2011 (6)
► 2010 (18)
► 2009 (31)
► 2008 (22)
► 2007 (11)

Search This Blog

Search

Labels

machine learning data mining map reduce Architecture Design Cloud computing algorithm NOSQL Hadoop scalability Distributed system parallel processing big data predictive analytics Design patterns SOA performance REST ensemble method recommendation engine

Pages

- Home

```
> # Check for the correlation with actual result
> cor(prediction, prestige_test$prestige)
[1] 0.9376719009
```

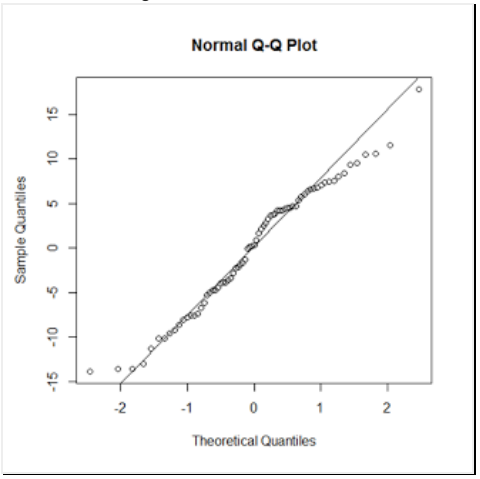
The coefficient column gives an estimation of Θi, there is an associated p-value that gives the confidence of each estimated Θi. For example, features not marked with at least one * can be safely ignored.

In the above model, education and income has a high influence to the prestige.

Linear regression has certain assumption about the underlying distribution of data which we need to validate. This include the residuals (error) has normal distribution with a zero mean and constant variation.

```
> # verify if the residuals are normally distributed
> rs <- residuals(model)
> qqnorm(rs)
> qqline(rs)
> shapiro.test(rs)
        Shapiro-Wilk normality test
data:  rs
W = 0.9744, p-value = 0.1424
```

Here is the plot of how the residual aligned with the normal distribution.

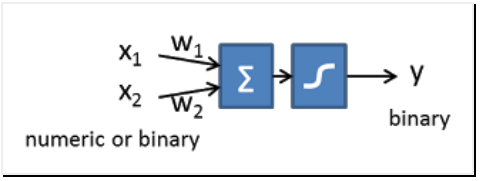


Logistic Regression

Logistic Regression is basically applying a transformation of the output of a linear regression such that it fits into the value of 0 to 1, and hence mimic the probability of a binary output. Logistic regression is the most popular machine learning technique applied in solving classification problem.

In a classification problem, the output is binary rather than numeric, we can imagine of doing a linear regression and then compress the numeric output into a 0..1 range using the logit function $1/(1+e^{-t})$

$y = 1/(1 + e^{-(\Theta_0 + \Theta_1 x_1 + \Theta_2 x_2 + \ldots)})$



The objective function in logistic regression is different. It is minimizing the entropy as follows ...

$\sum (y_{actual} * \log y_{estimate} + (1 - y_{actual}) * \log(1 - y_{estimate}))$

Here is the example R code to classify iris data.

```
> newcol = data.frame(isSetosa=(iristrain$Species == 'setosa'))
> traindata <- cbind(iristrain, newcol)
> head(traindata)
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species isSetosa
1          5.1         3.5          1.4         0.2  setosa     TRUE
2          4.9         3.0          1.4         0.2  setosa     TRUE
3          4.7         3.2          1.3         0.2  setosa     TRUE
4          4.6         3.1          1.5         0.2  setosa     TRUE
6          5.4         3.9          1.7         0.4  setosa     TRUE
7          4.6         3.4          1.4         0.3  setosa     TRUE
> formula <- isSetosa ~ Sepal.Length + Sepal.Width
                       + Petal.Length + Petal.Width
> logisticModel <- glm(formula, data=traindata,
                       family="binomial")
> # Predict the probability for test data
> prob <- predict(logisticModel, newdata=iristest, type='response')
> round(prob, 3)
  5  10  15  20  25  30  35  40  45  50  55  60  65  70  75
  1   1   1   1   1   1   1   1   1   1   0   0   0   0   0
 80  85  90  95 100 105 110 115 120 125 130 135 140 145 150
  0   0   0   0   0   0   0   0   0   0   0   0   0   0   0
```

Regression with Regularization

Notice our assumption of linearity between input and output variable above is not necessary true. One technique is to combine existing features by multiplying them together and hope we are lucky enough to hit some useful combination. Therefore we may end up having a large set of input features in our machine learning.

When we have a large size of input variable but moderate size of training data, we are subjected to the overfitting problem, which is our model fits too specific to the training data and not generalized enough for the data we haven't seen. Regularization is the technique of preventing the model to fit too specifically to the training data.

In linear regression, it is found that overfitting happens when Θ has a large value. So we can add a penalty that is proportional to the magnitude of Θ. In L2 regularization (also known as Ridge regression), Σ square(Θi) will be added to the cost function, while In L1 regularization (also known as Lasso regression), Σ ||Θi|| will be added to the cost function.

Both L1, L2 will shrink the magnitude of Θi, L2 tends to make dependent input variables having the same coefficient while L tends to pick of the coefficient of variable to be non-zero and other zero. In other words, L1 regression will penalize the coefficient of redundant variables that are linearly dependent and is frequently used to remove redundant features.

Combining L1 and L2, the general form of cost function becomes
Cost == Non-regularization-cost + λ (α.Σ ||Θi|| + (1- α).Σ square(Θi))

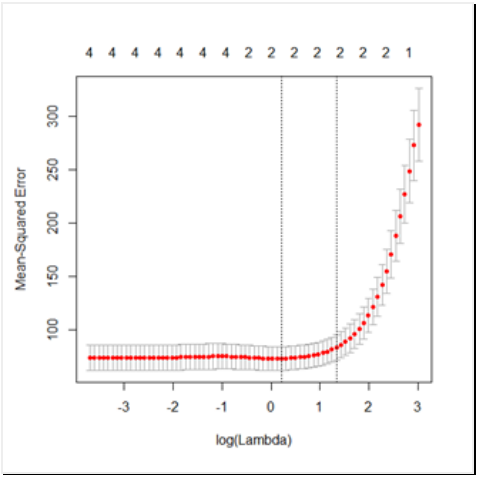
Notice there are 2 tunable parameters, lambda λ and alpha α. Lambda controls the degree of regularization (0 means no-regularization, infinity means ignoring all input variables because all coefficients of them will be zero). Alpha controls the degree of mix between L1 and L2. (0 means pure L2 and 1 means pure L1).

Glmnet is a popular regularization package. The alpha parameter need to be supplied based on the application need (for selecting a reduced set of variables, alpha=1 is preferred). The library provides a cross-validation test to automatically figure out what is the better lambda value.

Lets repeat the linear regression "prestige" data set and use regularization this time, we pick alpha = 0.7 to favor L1 regularization.

```
> library(glmnet)
> cv.fit <- cv.glmnet(as.matrix(prestige_train[,c(-4, -6)]),
                      as.vector(prestige_train[,4]),
                      nlambda=100, alpha=0.7,
                      family="gaussian")
> plot(cv.fit)
> coef(cv.fit)
5 x 1 sparse Matrix of class "dgCMatrix"
                      1
(Intercept) 6.3876684930151
education   3.2111461944976
income      0.0009473793366
women       0.0000000000000
census      0.0000000000000
> prediction <- predict(cv.fit,
                        newx=as.matrix(prestige_test[,c(-4, -6)]))
> cor(prediction, as.vector(prestige_test[,4]))
          [,1]
1 0.9291181193
```

Here is the cross-validation plot, which shows the best lambda with minimal mean square error.



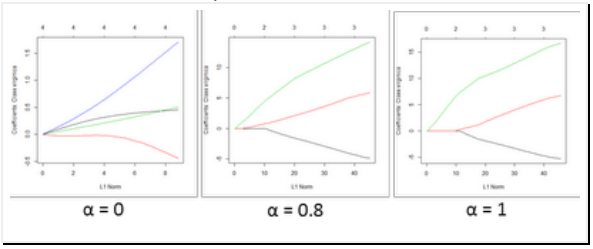
Also we can repeat the logistic regression "iris" data set and use regularization this time, we pick alpha = 0.8 to even favor L1 regularization more.

```
> library(glmnet)
> cv.fit <- cv.glmnet(as.matrix(iristrain[,-5]),
                      iristrain[,5], alpha=0.8,
                      family="multinomial")
> prediction <- predict(cv.fit,
                        newx=as.matrix(iristest[,-5]),
                        type="class")
> table(prediction, iristest$Species)
prediction    setosa versicolor virginica
  setosa         10          0         0
  versicolor      0         10         2
  virginica       0          0         8
```

Instead of picking lambda (the degree of regularization) based on cross validation, we can also based on the number of input variables that we want to retain. So we can plot the "regularization path" which shows how the coefficient of each input variables changes when the lambda changes and pick the right lambda that filter out the number of input variables for us.

```
> # try alpha = 0, Ridge regression
> fit <- glmnet(as.matrix(iristrain[,-5]),
                iristrain[,5], alpha=0,
                family="multinomial")
> plot(fit)
> # try alpha = 0.8, Elastic net
> fit <- glmnet(as.matrix(iristrain[,-5]),
                iristrain[,5], alpha=0.8,
                family="multinomial")
> plot(fit)
> # try alpha = 1, Lasso regression
> fit <- glmnet(as.matrix(iristrain[,-5]),
                iristrain[,5], alpha=1,
                family="multinomial")
> plot(fit)
```

Here is the output of these plots. Notice that as Alpha is closer to 1 (L1 regularization), it tends to have a stronger filter to the number of input variables.



In my next post, I will cover the other machine learning techniques.

Posted by Ricky Ho at 11:23 AM

+16 Recommend this on Google

Labels: data mining, linear regression, logistic regression, machine learning, predictive analytics, regularization

2 comments:

Bruce Lee said...

ricky your blog is excellent.
you should consider writing a book!

July 19, 2013 at 3:27 PM

Shameek Mukherjee said...

Hi Ricky,

Many thanks for this blog. I am new to R and trying to learn data mining using R by myself. I found this blog to be extremely useful. However, could you please explain a but about this line:

testidx <- which(1:length(iris[,1])%%5 == 0
[,1] is mentioning to first column in iris dataset but I didn't get he meaning of the whole line. I would be grateful if you can explain this code.
Many thanks in advance.

September 6, 2013 at 4:50 AM

Post a Comment

Newer Post Home Older Post

Subscribe to: Post Comments (Atom)