Bias Variance Trade-off, Ensemble Methods, Dimension Reduction

R Study Group Meet Up

Ryan Zhang

November 20, 2015

Bias and Variance Trade-off

Say we have build a regression function $\hat{y} = \hat{g}(x)$

Mean squared error in regression problem can be decomposed into three parts:

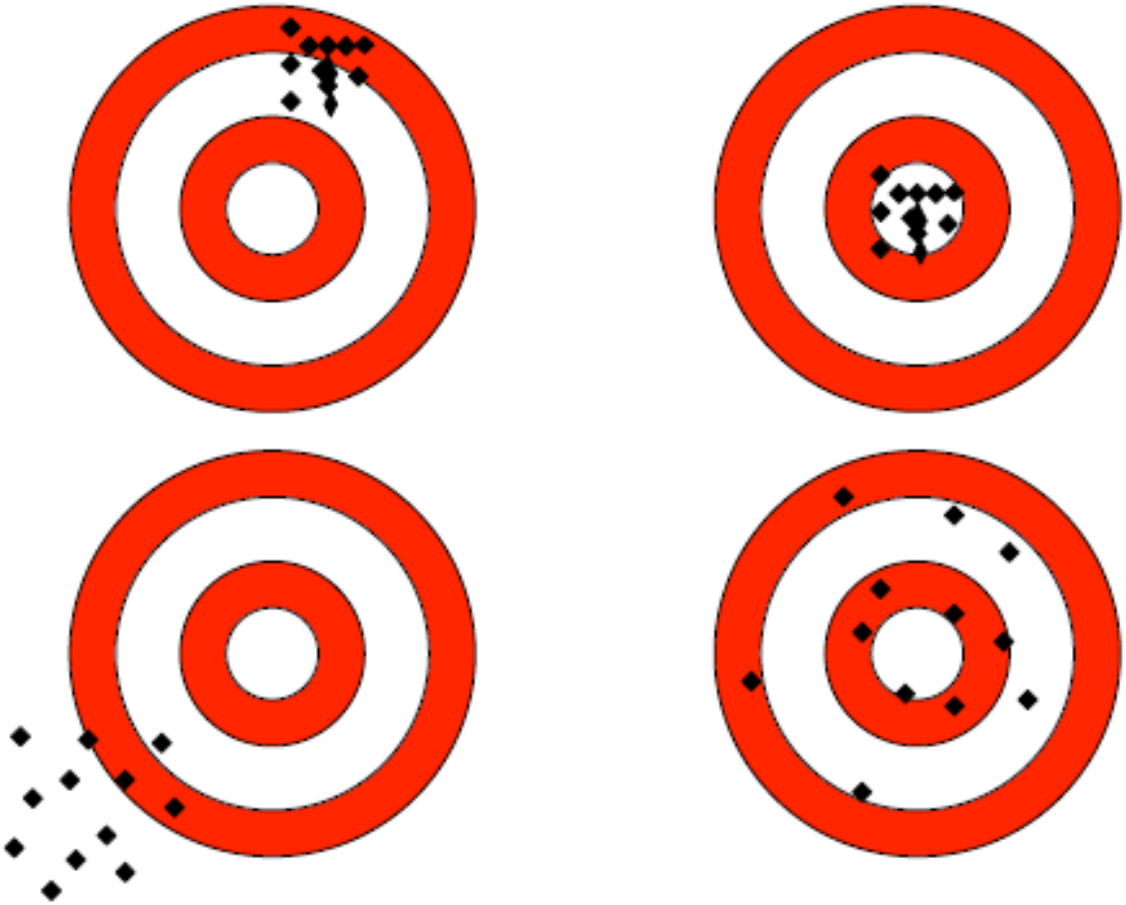
$$MSE = Var(\hat{g}(X)) + Bias(\hat{g}(X))^2 + Var(\epsilon)$$

See Prof. Wang's slides

- Variance: How $\hat{g}(X)$ would change if we use a different training data
- Bias: Limitation of the family of model(H) we choose

We want a model with low bias and variance at the same time, but generally, increase one will cause the other to decrease.

Interpretation of Bias and Variance



Model Complexity and Bias Variance Trade-off

Say if we have a first order multiple regression model, and a third order polynomial regression model both fitted using the same set of features

$$g_1(X) = \hat{\beta}_0 + \Sigma_{i=1}^{p} \hat{\beta}_i x_i$$

$$g_2(X) = \hat{\beta}_0 + \Sigma_{i=1}^{p} \Sigma_{d=1}^{3} \hat{\beta}_{i,d} x_i^d$$

Which model would you expect to have higher bias? _____

Which model would you expect to have higher variance? _____

Which model would you expect to be more generalizable? _____

Generally, as the model complexity goes up, bias will decrease whereas variance will increase

Why We Use CV to Pick Hyper Parameters?

Remember the λ in Lasso, cp in Decision Trees or $Kernel$ in Support Vector Machine?

What does the hyper parameter of a model control? _____

Tuning the value for hyper parameter is equalvent to balance between _____ and _____

Classifiers We Covered Now

1. Logistic Regression
2. Decision Trees
3. Support Vector Machine

In addition to add them to your resume(please do that)

What can we do about them?

Would the combination of different classifiers give better result?

Many heads are better than one

Example Data

Tired of red wine? Drinking white wine this time...

```
## 'data.frame':    4898 obs. of  12 variables:
##  $ fixed.acidity       : num  7 6.3 8.1 7.2 7.2 8.1 6.2 7 6.3 8.1 ...
##  $ volatile.acidity    : num  0.27 0.3 0.28 0.23 0.23 0.28 0.32 0.27 0.3 0.2
##  $ citric.acid         : num  0.36 0.34 0.4 0.32 0.32 0.4 0.16 0.36 0.34 0.4
##  $ residual.sugar      : num  20.7 1.6 6.9 8.5 8.5 6.9 7 20.7 1.6 1.5 ...
##  $ chlorides           : num  0.045 0.049 0.05 0.058 0.058 0.05 0.045 0.045
##  $ free.sulfur.dioxide : num  45 14 30 47 47 30 30 45 14 28 ...
##  $ total.sulfur.dioxide: num  170 132 97 186 186 97 136 170 132 129 ...
##  $ density             : num  1.001 0.994 0.995 0.996 0.996 ...
##  $ pH                  : num  3 3.3 3.26 3.19 3.19 3.26 3.18 3 3.3 3.22 ...
##  $ sulphates           : num  0.45 0.49 0.44 0.4 0.4 0.44 0.47 0.45 0.49 0.4
##  $ alcohol             : num  8.8 9.5 10.1 9.9 9.9 10.1 9.6 8.8 9.5 11 ...
##  $ quality             : Factor w/ 2 levels "0","1": 1 1 1 1 1 1 1 1 1 1 ...
```

Holdout 30% Data as Test Set

In this simple validation approach, we train classifiers on 70% of our data, and evaluate the performance of classifiers using the remaining 30% data

If we simply test our classifiers on the 70% training data, the measurement will be

---

```
library(caTools)
set.seed(0306)
split <- sample.split(wine$qualit,  SplitRatio = 0.7)
train <- wine[split,]
test <- wine[!split,]
c(nrow(train),nrow(test))
```

```
## [1] 3429 1469
```

Build Three Classifiers

Should be easy code now...

We will skip hyper parameter tuning for now

However you are welcome to try tuning them as an exercise

```
library(rpart);library(e1071);
rp <- rpart(quality~., data = train, control = rpart.control(cp = 0.01))
SVM <- svm(quality~., data = train, kernel ="polynomial",probability=T)
logReg <- glm(quality~., data = train, family = "binomial")
rp.pred <- predict(rp,test,type = "class")
SVM.pred <- predict(SVM, test)
logReg.pred <- predict(logReg, test, type = "response") >= 0.5
```

Test Set Accuracies

```
accuracy <- function(pred, true){
    t <- table(pred, true)
    return(round(sum(diag(t))/sum(t),4))}
c(accuracy(rp.pred, test$quality),accuracy(SVM.pred, test$quality),
accuracy(logReg.pred, test$quality))
```

```
## [1] 0.8135 0.8012 0.7890
```

Remember that accuracy is not always a good metrics, especially when the classes are not even.

The good wine here is 21.64%.

If we build a naive classifier which always predict a wine is bad, the accuracy would be about _____

What If We Let the Classifers Vote?

If two out of three classifiers says that the wine is good, we classify it as good wine

Why we always need multiple judges to make decisions?

To reduce _____

```
Ensemble.pred <- as.numeric(rp.pred) + as.numeric(SVM.pred) +
                 as.numeric(logReg.pred) - 2
Ensemble.pred <- ifelse(Ensemble.pred >= 2,1,0)
accuracy(Ensemble.pred, test$quality)
```

```
## [1] 0.8046
```

Again, the accuracy score may not make any sense.

Refer to my slides last time, we can look at precision, _____ , tpr, _____ , f1 , _____ , etc.

Many competitions hosted on Kaggle look at the AUC score, which is the area under the ROC curve.

Look at AUC for Three Classifiers

Notice that, in order to generate the ROC curve, we need a _____ output, rather than _____

```
library(ROCR)
rp.pred <- prediction(predict(rp,test,type = "prob")[,2], test$quality)
SVM.pred <- prediction(attr(predict(SVM, test, probability = T),
                            "probabilities")[,2], test$quality)
logReg.pred <- prediction(predict(logReg, test, type = "response"),
                          test$quality)
rp.auc <- unlist(slot(performance(rp.pred, "auc"),"y.values"))
SVM.auc <- unlist(slot(performance(SVM.pred, "auc"),"y.values"))
logReg.auc <- unlist(slot(performance(logReg.pred, "auc"),"y.values"))
c(rp.auc, SVM.auc, logReg.auc)
```

```
## [1] 0.8005412 0.7998213 0.7847374
```

What is the AUC for Ensemble?

What is we take the _____ of the probabilities predicted by three classifiers?

What would the AUC be if we use that as our final estimate?

```
Ensemble.prob <- (predict(rp,test,type = "prob")[,2] +
        attr(predict(SVM, test, probability = T), "probabilities")[,2] +
        predict(logReg, test, type = "response"))/3
Ensemble.pred <- prediction(Ensemble.prob, test$quality)
unlist(slot(performance(Ensemble.pred, "auc"),"y.values"))
```

```
## [1] 0.8345956
```

Ensemble of models different in nature will tend to decrease _____

Random Forest

Combine many (usually thousands of) decision trees models to form a forest

Remember we just said that the models should be different in nature but the tree building process is pretty deterministic

A bit review won't hurt...

At each node in the tree, we find the best split by searching through all features and find one feature that supports a cutoff point such that the _____ can be maximized

How can we build thousands of trees and the same time make them quite different from each other by introducing _____?

The Randomness in Random Forest

Introduce randomness in data:

Repeatedly draw random samples of the same size as the training set (with replace) from our training set, and fit decision trees on each of the _____ sample

Introduce randomness in tree building method:

At each node in the tree, only randomly selected m features can be considered to find a split

These changes will results `decorrelated` trees, thus by averaging these trees will reduce variance

Growing a Random Forest

```
library(randomForest)
set.seed(0306)
t0 <- Sys.time()
rf <- randomForest(quality~., data = train, ntree = 5000, importance = T)
print(Sys.time() - t0)
```

```
## Time difference of 1.199408 mins
```

```
rf.pred <- prediction(predict(rf, test, type = "prob")[,2], test$quality)
unlist(slot(performance(rf.pred, "auc"),"y.values"))
```

```
## [1] 0.9099935
```

Amazing improvement, are we over optimized?

Feature Importance From Random Forest

Decision Trees have very good interpretation, but Random Forest have none

But we gain an importance measure of features

Look at all the trees we built in the forest, scan through all the splits, aggregate the information gains/ decreases in Gini index resulted from split using a perticular feature

Repeat that for all the features

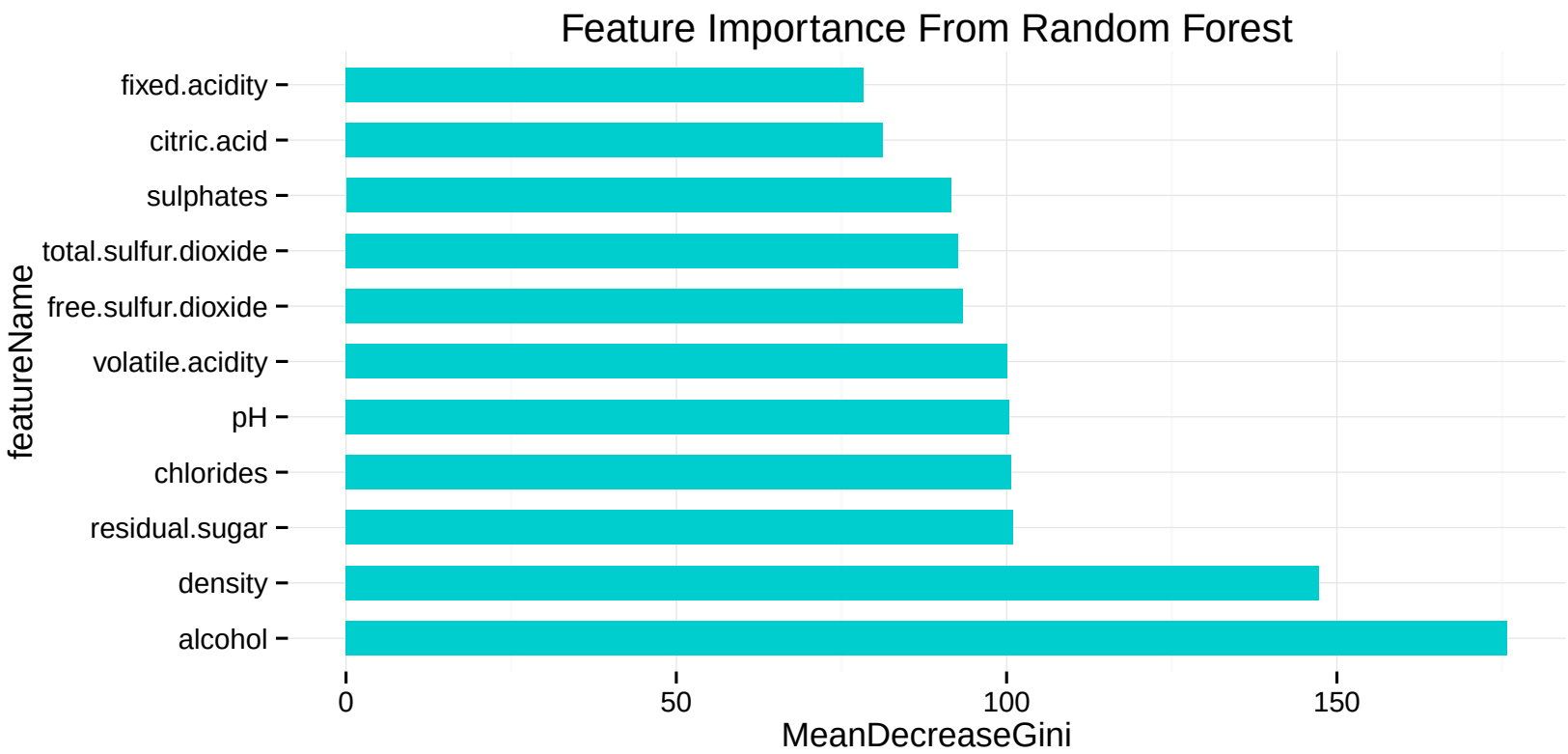
Feature Importance In Our Wine Case

We covered dplyr and ggplot, use them to plot the feature importance

Hope you are still familiar with their syntax. . . .

```
library(ggplot2);library(ggthemes);library(dplyr)
theme_set(theme_minimal(12))
df <- cbind.data.frame(featureName = names(wine)[-12],
            MeanDecreaseGini = rf$importance[,"MeanDecreaseGini"])
df <- arrange(df, desc(MeanDecreaseGini))
df$featureName <- factor(df$featureName,
                         levels = as.character(df$featureName))
ggplot(data = df, aes(x = featureName, y = MeanDecreaseGini)) +
    geom_bar(stat = "identity", fill = "cyan3", width = 0.618) +
    coord_flip() +
    ggtitle("Feature Importance From Random Forest")
```

Plotting Feature Importance



Tuning Hyper Parameters for Random Forest

Most important parameter is `mtry`

```
set.seed(1106)
t0 <- Sys.time()
rf.model <- tune.randomForest(x = train[,1:11], y = train[,12],
                    mtry = c(1:5), ntree = 5000,
    tunecontrol = tune.control(sampling = "cross", cross = 5))
print(Sys.time() - t0)
```

```
## Time difference of 13.83202 mins
```

```
print(rf.model$best.parameters)
```

```
##   mtry ntree
## 1    1  5000
```

What's the NEW Test Set AUC?

It should be higher

```
set.seed(0306)
t0 <- Sys.time()
rf <- randomForest(quality~., data = train, mtry = 1, ntree = 5000)
print(Sys.time() - t0)
```

```
## Time difference of 38.40468 secs
```

```
rf.pred <- prediction(predict(rf, test, type = "prob")[,2], test$quality)
unlist(slot(performance(rf.pred, "auc"),"y.values"))
```

```
## [1] 0.9138758
```

Boosting Trees

For a Random Forest model, we grow 5000 trees independently, we can ask 5000 computers, each grow a tree for us and then taking the average

Boosting is a method that we grown trees sequentially, each tree is grown using information from previously grown trees

The trees are simpler and they are fitted with special attention to the errors of the previous iteration. After the iteration the new model is aggregated into the model with a learning/shrinkage parameter

That is, each step we try to slowly improve on the places where it did not do well previously

Tuning a AdaBoost Model

It is time consuming, however it is right thing to do

```
library(ada);library(caret)
t0 <- Sys.time()
tuningParams <- expand.grid(iter = c(100,200),
                            nu = c(0.01,0.02,0.03),
                            maxdepth = 3:4)
trainControl <- trainControl(method = "cv", number = 5)
adaboost <- train(x = train[,1:11], y = train[,12], method = "ada",
                          trControl = trainControl,
                          tuneGrid = tuningParams)
print(Sys.time() - t0)
```

```
## Time difference of 5.831426 mins
```

Tuning a AdaBoost Model

```
adaboost$results
```

```
##       nu maxdepth iter  Accuracy     Kappa  AccuracySD    KappaSD
## 1   0.01        3  100 0.8136494 0.3016942 0.009827623 0.04413352
## 5   0.02        3  100 0.8139384 0.3124198 0.008198646 0.02591753
## 9   0.03        3  100 0.8165679 0.3294744 0.011800749 0.03217314
## 3   0.01        4  100 0.8162729 0.3290489 0.009277686 0.03665705
## 7   0.02        4  100 0.8194808 0.3562922 0.012933953 0.04442923
## 11  0.03        4  100 0.8203550 0.3665482 0.010294363 0.03369351
## 2   0.01        3  200 0.8151080 0.3196780 0.008293075 0.03581106
## 6   0.02        3  200 0.8171463 0.3417050 0.007088517 0.01449643
## 10  0.03        3  200 0.8159835 0.3405761 0.006875611 0.01803062
## 4   0.01        4  200 0.8186048 0.3532832 0.009619655 0.03498517
## 8   0.02        4  200 0.8223966 0.3763358 0.012524079 0.03992960
## 12  0.03        4  200 0.8291039 0.4043337 0.007694582 0.02212970
```

Test Set AUC?

Note these parameters are not well tuned, I guessed them

You should try and figure out them by cross-validation

I just don't have that much time this week...

```
set.seed(0306)
adaboost <- ada(x = train[,1:11], y = train[,12], loss = "ada",
                iter = 5000, nu = 0.05,  rpart.control(maxdepth = 3))
ada.pred <- prediction(predict(adaboost, test[,1:11], type = "prob")[,2],
                       test$quality)
unlist(slot(performance(ada.pred, "auc"),"y.values"))
```

```
## [1] 0.8954778
```

XGBoost

I am not very familiar with tuning the Boosting Trees, find out more else where if you are interested

There is a trending implementation of Boosted Models by Chinese(proudly) called XGBoost

I have tried it to fit models to datasets over 10GB in size and the training time is within minutes, super fast

Dimension Reduction

Thinking about feature selection, what we do is to _____ the dimension of the feature space by _____ features

Rather than doing that, we can **summarize** the high dimension features into lower dimension representations

It is like compress a huge blue-ray movie into smaller format in order to watch it on your phone. Of course, there will be loss in information during the process.

But the things got lost during the process are often the tiny details that does not affect you understanding what's going on in the movie

You may miss a freckle, but not the entire face

Why We Do This Compression?

1. Reduce highly correlated features into fewer ones
2. Otherwise we can't visualize data beyound 3d

I am not going into the details of the math behind this, it is just linear algebra and optimization which you have had enough already(ever heard about eigen vectors and eigen values?)

The gist is that, we want to find a way to summarize our high dimensional data set in to a lower dimensional representation and the same time maximizing the variations remained in the data set

Principal Component Analysis

There are many ways to do dimension reduction, PCA is just one of them

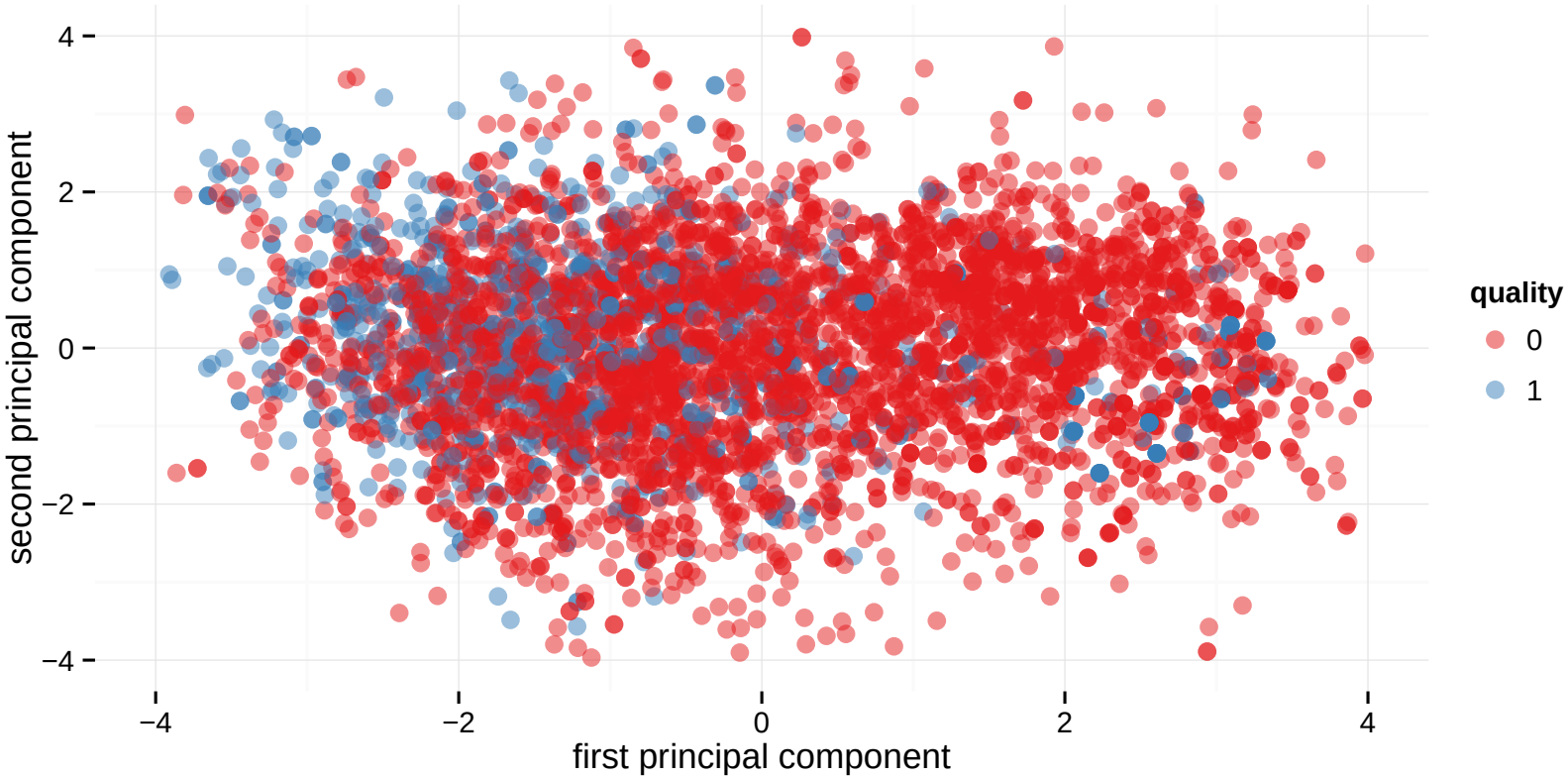
What PCA does is to find a set of orthonormal directions along which the original data are highly variable

Notice this process is _____ , since it does not make use of the information related to the labels

We are going to plot using the first two principal components

```
pcawine <- prcomp(wine[,1:11], scale = T)
df <- cbind.data.frame(pc1 = pcawine$x[,1], pc2 = pcawine$x[,2],
                       quality = wine$quality)
```

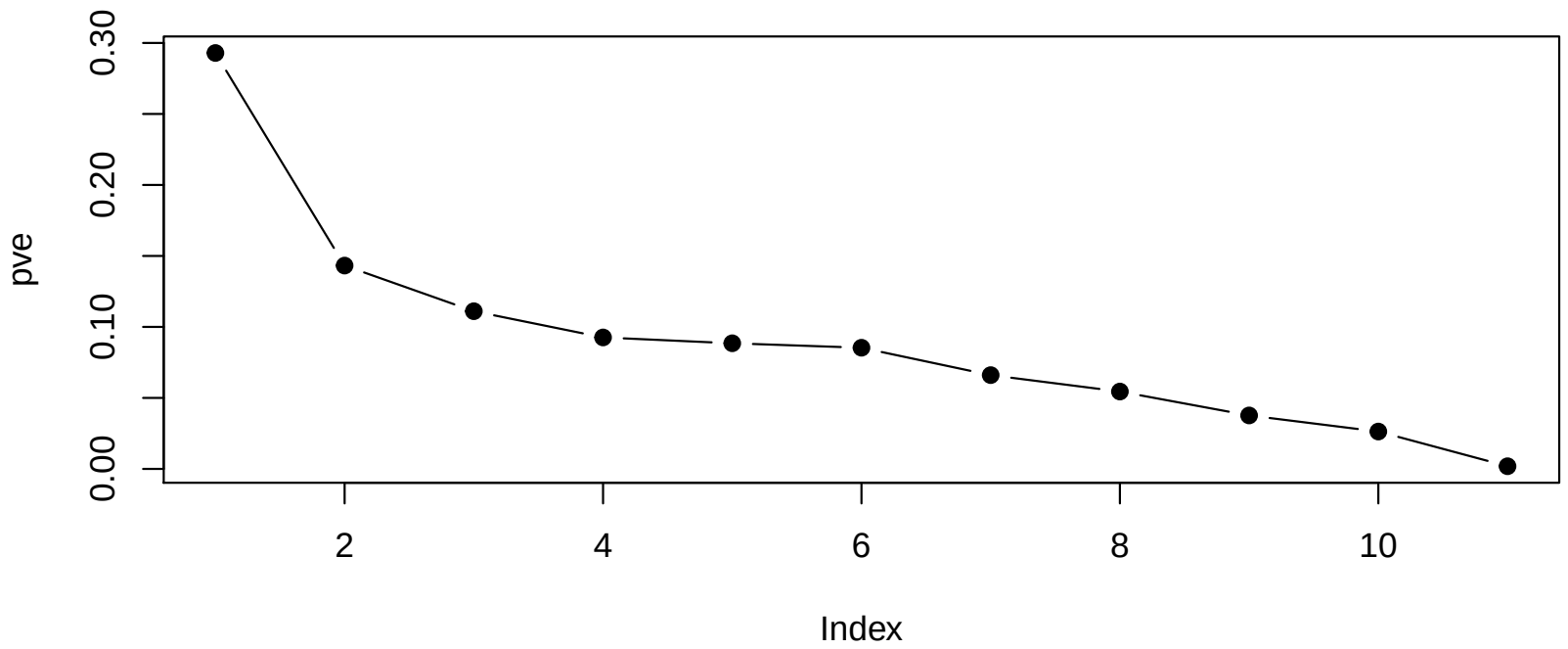
Plot with First Twon Principal Components



The Notion of Percentage of Variance Explained

These are just ratios of the variances along principal components and total variance within the data set

```
pve = pcawine$sdev^2/sum(pcawine$sdev^2)
plot(pve, pch = 19, type = "both")
```



Using Principal Components as Predictors

Let's use the first 2 principal components to build logistic regression model

```
pctrain <- cbind.data.frame( pcawine$x[split,1:2],quality = train$quality)
pctest <- cbind.data.frame(pcawine$x[!split,1:2], quality = test$quality)
pclogReg <- glm(quality~.,data = pctrain, family = "binomial")
pclogReg.pred <- prediction(predict(pclogReg, pctest, type = "response"),
                            test$quality)
unlist(slot(performance(pclogReg.pred, "auc"),"y.values"))
```

```
## [1] 0.6971515
```

Don't be superised to see such low score

We lost 60% variation in the data when using only two principal components

Caveat

Although it looks similar to feature selection in the sense that they both reduce the dimensionality of the problem

But it is not the thing you do when fighting _____ , since the process is _____

Do this when the data set is too large for you, or when the algorithm runs forever without give you a model

The Principal Components are hard to interpret, and there is a CUR decomposition method which does similar job as PCA but with some interpretation

Go read more about it if you are interested

Bad News or Good News?

NO MEET UP NEXT WEEK, ENJOY YOUR HOLIDAY~

The week after we will cover a Kaggle case, which will be the last meet up this semester