The class imbalance problem

In the Caravan data, there is only 6% of positive samples among 5822 and we found KNN, Logistic model, and LDA cannot beat the naive classifier that labels every case negative, in terms of the true positive rate. Problems of this type can create all sorts of difficulties in the task of obtaining predictive models.

- they require proper evaluation metrics as it is well known that the standard accuracy (or its complement error rate) is clearly inadequate for these domains. In effect, a naive classifer usually has high accuracy.
- Another problem with class imbalance is that it has a strong impact on the performance of the learning algorithms that tend to disregard the minority class given its lack of statistical support. This is particularly problematic in situations where this minority class is exactly the most relevant class, as is the case in our domain.

This lecture is partially based on the book: Data Mining with R: Learning with Case Studies by Luis Torgo.

One of the remedies is to use sampling methods that manipulate the training data to change the class distribution.
Under-sampling methods select a small part of the majority class examples and add them to the minority class cases, thereby building a dataset with a more balanced class distribution.
Over-sampling methods work the other way around, using some process to replicate the minority class examples. Many variants of these two general sampling approaches exist. A successful example is the **SMOTE** method (Chawla et al., 2002). The general idea of this method is to artificially generate new examples of the minority class using the nearest neighbors of these cases. Furthermore, the majority class examples are also under-sampled, leading to a more balanced dataset. We have implemented this sampling method in a function called SMOTE(). Our implementation uses a **mixed-mode distance function** so you can use SMOTE() on datasets with both continuous and nominal variables.

```
> library(DMwR)
> data(iris)
> data <- iris[,c(1,2,5)]
> data$Species <- factor(ifelse(data$Species ==
+                               'setosa','rare','common'))
> newData <- SMOTE(Species ~ .,data,perc.over=600)
> table(newData$Species)
> par(mfrow=c(1,2))
> plot(data[,1],data[,2],pch=19+as.integer(data[,3]),
+      main='Original Data')
> plot(newData[,1],newData[,2],
+      pch=19+as.integer(newData[,3]),main="SMOTE'd Data")
```

```
> library(ISLR)
> library(class)
> data(Caravan)
> attach(Caravan)
> standardized.X=scale(Caravan[,-86])
> test=1:1000
> train.X=standardized.X[-test,]
> test.X=standardized.X[test,]
> train.Y=Purchase[-test]
> test.Y=Purchase[test]
> knn.pred=knn(train.X,test.X,train.Y,k=5)
> table(knn.pred,test.Y)
> glm.fit=glm(Purchase~.,data=Caravan,family=binomial,
+             subset=-test)
> glm.probs=predict(glm.fit,Caravan[test,],
+                   type="response")
> glm.pred=rep("No",1000)
> glm.pred[glm.probs>.25]="Yes"
> table(glm.pred,test.Y)
```

```
> data = data.frame(standardized.X, Purchase=Caravan[,86])
> newdata = SMOTE(Purchase~.,data, perc.over=600)
> dim(newdata)
> train = !is.element(row.names(newdata),
+                         as.character(test))
> knn.SMOTE=knn(newdata[train,-86], data[test, -86],
+                 newdata[train,86], k=5)
> table(knn.SMOTE, data[test,86])
> glm.SMOTE=glm(Purchase~., newdata, subset=train,
+                 family=binomial)
> glm.SMOTE.probs=predict(glm.SMOTE, data[test,],
+                           type="response")
> glm.SMOTE.pred=rep("No", 1000)
> glm.SMOTE.pred[glm.SMOTE.probs > 0.5]="Yes"
> table(glm.SMOTE.pred, test.Y)
```

```
> library(MASS)
> lda.fit=lda(Purchase~., newdata, subset=train)
> lda.pred=predict(lda.fit, data[test,])
> table(lda.pred$post[,"Yes"] > 0.2, test.Y)
```

An overview of classification algorithm for imbalanced datasets

http://citeseerx.ist.psu.edu/viewdoc
/download?doi=10.1.1.413.3344&rep=rep1&type=pdf