1 Partial Least Squares Discriminant Analysis

- The plsda function is a wrapper for the plsr function in the pls package that does not require a formula interface and can take factor outcomes as arguments. The classes are broken down into dummy variables (one for each class). These 0/1 dummy variables are modeled by partial least squares.

- From this model, there are two approaches to computing the class predictions and probabilities:

  1. the softmax technique can be used on a per-sample basis to normalize the scores so that they are more "probability like" (i.e. they sum to one and are between zero and one). For a vector of model predictions for each class X, the softmax class probabilities are computed as. The predicted class is simply the class with the largest model prediction, or equivalently, the largest class probability. This is the default behavior for plsda.

  2. Bayes rule can be applied to the model predictions to form posterior probabilities. Here, the model predictions for the training set are used along with the training set outcomes to create conditional distributions for each class. When new samples are predicted, the raw model predictions are run through these conditional distributions to produce a posterior probability for each class (along with the prior). Bayes rule can be used by specifying probModel = "Bayes". An additional parameter, prior, can be used to set prior probabilities for the classes.

     The advantage to using Bayes rule is that the full training set is used to directly compute the class probabilities (unlike the softmax function which only uses the current sample's scores). This creates more realistic probability estimates but the disadvantage is that a separate Bayesian model must be created for each value of ncomp, which is more time consuming.

- For the sonar data set, we can fit two PLS models using each technique and predict the class probabilities for the test set.

```
plsFit <- plsda(training, trainClass, ncomp = 20)
```

plsFit Partial least squares classification, fitted with the kernel algorithm. The softmax function was used to compute class probabilities.

```
Call:
plsr(formula = y ~ x, ncomp = ncomp, data = tmpData)
plsBayesFit <- plsda(training, trainClass, ncomp = 20,
                     probMethod = "Bayes")
```

plsBayesFit Partial least squares classification, fitted with the kernel algorithm. Bayes rule was used to compute class probabilities.

```
Call:
plsr(formula = y ~ x, ncomp = ncomp, data = tmpData)
predict(plsFit, head(testing), type = "prob")
, , 20 comps

        M      R
4  0.6228 0.3772
6  0.5241 0.4759
12 0.3884 0.6116
16 0.1925 0.8075
17 0.1801 0.8199
19 0.1337 0.8663
predict(plsBayesFit, head(testing), type = "prob")
, , ncomp20

          M       R
```

```
4  0.950775 0.04922
6  0.585469 0.41453
12 0.076099 0.92390
16 0.002769 0.99723
17 0.003715 0.99628
19 0.023820 0.97618
```

- Similar to plsda, caret also contains a function splsda that allows for classification using sparse PLS. A dummy matrix is created for each class and used with the spls function in the spls package. The same approach to estimating class probabilities is used for plsda and splsda.