Lab Week 5

STAT5003

Dr. Justin Wishart

Semester 2, 2022

Contents

Preparation and assumed knowledge

- Classification content in Module 5.
- Listen to the Week 5 lecture pre-recording.
- Required `R` packages
  - `mlbench`
  - `PimaIndiansDiabetes2` dataset from `mlbench`

Aims

- Explore classification algorithms
  - Logistic regression
  - Linear Discriminant Analysis (LDA)
  - k-nearest neighbours (kNN)
  - Support Vector Machines (SVM)

This week, we will be learning how to do classification with the four algorithms - logistic regression, LDA, kNN and SVMs.

The dataset we will be using is called `PimaIndiansDiabetes` and it is included as part of the `mlbench` package. You will first need to install the `mlbench` package. To load the data, do the following:

```
library(caret)
```

```
## Loading required package: lattice
```

```
##
## Attaching package: 'caret'
```

```
## The following object is masked from 'package:purrr':
##
##     lift
```

```
data(PimaIndiansDiabetes2, package = "mlbench")
```

1 IDA Pima Indians Diabetes data

Inspect the `PimaIndiansDiabetes2` and verify its structure. It should have 9 columns, 8 of those being numeric features and a single class label variable. If any missing data is observed, discard it and use complete cases.

Solution

```
dim(PimaIndiansDiabetes2)
```

```
## [1] 768   9
```

```
head(PimaIndiansDiabetes2)
```

```
##   pregnant glucose pressure triceps insulin mass pedigree age diabetes
## 1        6     148       72      35      NA 33.6    0.627  50      pos
## 2        1      85       66      29      NA 26.6    0.351  31      neg
## 3        8     183       64      NA      NA 23.3    0.672  32      pos
## 4        1      89       66      23      94 28.1    0.167  21      neg
## 5        0     137       40      35     168 43.1    2.288  33      pos
## 6        5     116       74      NA      NA 25.6    0.201  30      neg
```

```
vapply(PimaIndiansDiabetes2, class, character(1))
```

```
##  pregnant   glucose  pressure   triceps   insulin      mass  pedigree       age
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##  diabetes
##  "factor"
```

```
vapply(PimaIndiansDiabetes2, anyNA, logical(1))
```

```
## pregnant  glucose pressure  triceps  insulin     mass pedigree      age
##    FALSE     TRUE     TRUE     TRUE     TRUE     TRUE    FALSE    FALSE
## diabetes
##    FALSE
```

```
complete.pimas <- PimaIndiansDiabetes2 %>% drop_na
```

Either directly above in the printed table or more explicitly with the `class` command applied to each element in the dataframe, we can see that the first 8 columns are numerically coded data and the last, 9th column, is a factor variable with the label `diabetes`.

2 Logistic Regression

Use `glm()` to perform logistic regression to classify observations as positive or negative for diabetes.

In particular, determine which of the features seem the most informative to explain the diabetes class.

Solution

```
logit.model <- glm(diabetes ~ ., data = PimaIndiansDiabetes2,
                   family = binomial(link = "logit"))
summary(logit.model)
```

```
##
## Call:
## glm(formula = diabetes ~ ., family = binomial(link = "logit"),
##     data = PimaIndiansDiabetes2)
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.7823  -0.6603  -0.3642   0.6409   2.5612
##
## Coefficients:
##               Estimate Std. Error z value Pr(>|z|)
## (Intercept) -1.004e+01  1.218e+00  -8.246  < 2e-16 ***
## pregnant     8.216e-02  5.543e-02   1.482  0.13825
## glucose      3.827e-02  5.768e-03   6.635 3.24e-11 ***
## pressure    -1.420e-03  1.183e-02  -0.120  0.90446
## triceps      1.122e-02  1.708e-02   0.657  0.51128
## insulin     -8.253e-04  1.306e-03  -0.632  0.52757
## mass         7.054e-02  2.734e-02   2.580  0.00989 **
## pedigree     1.141e+00  4.274e-01   2.669  0.00760 **
## age          3.395e-02  1.838e-02   1.847  0.06474 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 498.10  on 391  degrees of freedom
## Residual deviance: 344.02  on 383  degrees of freedom
##   (376 observations deleted due to missingness)
## AIC: 362.02
##
## Number of Fisher Scoring iterations: 5
```

The above prepares the session for classification analysis (not required for this subquestion but future ones), load the `e1071` and `caret` packages. After fitting the logistic regression with a call to `glm`, inspecting the fitted model reveals that the variable with the most significance is the `glucose` variable (ignoring the intercept). As an additional check, fit individual logistic regressions on each of the predictors individually,

```
predictors <- names(PimaIndiansDiabetes2)[-9]
sapply(predictors,
       function(x) summary(glm(as.formula(paste0("diabetes ~ ", x)),
                               family = binomial(link = "logit"),
                               data = PimaIndiansDiabetes2))$coefficients[2, 4])
```

```
##     pregnant      glucose     pressure      triceps      insulin         mass
## 2.147445e-09 2.985479e-33 5.718197e-06 8.023829e-09 7.166747e-08 4.309761e-16
##     pedigree          age
## 3.702926e-06 1.773155e-10
```

The above code extracts out the p-values for the predictors fitted on a simple logistic regression where there is only one predictor in each case. We can see that `glucose` has a p-value that is zero to 30 decimal places.

2.1 Logistic regression : compute the accuracy

Compute the accuracy of the logistic regression classifier across the entire training data set.

Solution

```
logit.decision <- ifelse(logit.model$fitted.values > 0.5, "pos", "neg")
complete.pima <- PimaIndiansDiabetes2 %>% drop_na
logit.accuracy <- mean(logit.decision == complete.pima$diabetes, na.rm = TRUE) * 100
logit.accuracy
```

```
## [1] 78.31633
```

Using the complete data in the dataset, the accuracy is 78.3163265%.

3 Classifier comparisons

Install and load the `caret` package. Use the `caret` package for this question.

3.1 Partition the data

Partition the `PimaIndiansDiabetes2` dataset into 75% training and 25% test.

Solution

```
set.seed(123)
inTrain <- createDataPartition(complete.pima$diabetes, p = .75)[[1]]
pimatrain <- complete.pima[inTrain, ]
pimatest  <- complete.pima[-inTrain, ]

nrow(pimatest)
```

```
## [1] 97
```

3.2 Train classifiers

Using the training dataset and all the given features, train three classifiers (logistic regression, LDA, kNN and SVM classifiers). This can be done using the `caret` package and selecting the appropriate `method` parameter argument in the `train` function. For SVM, consider using the choice `method = "svmLinearWeights"`. A full list of supported methods are given here. Compute the accuracy of each classifier on the training dataset.

```
# Logistic regression
logit.model <- train(diabetes ~ ., data = pimatrain, method = "glm",
                     family = binomial(link = "logit"),
                     trControl = trainControl(method = "repeatedcv", repeats = 5))
logit.model
```

```
## Generalized Linear Model
##
## 295 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 266, 266, 265, 265, 265, 266, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7833678  0.4891548
```

```r
# LDA
lda.model <- train(diabetes ~ ., data = pimatrain, method = "lda",
                   trControl = trainControl(method = "repeatedcv", repeats = 5))
lda.model
```

```
## Linear Discriminant Analysis
##
## 295 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 265, 265, 267, 266, 265, 265, ...
## Resampling results:
##
##   Accuracy   Kappa
##   0.7797192  0.4741093
```

```r
# knn
knn.model <- train(diabetes ~ ., data = pimatrain, method = "knn",
                   trControl = trainControl(method = "repeatedcv", repeats = 5))
knn.model
```

```
## k-Nearest Neighbors
##
## 295 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 265, 265, 265, 266, 265, 266, ...
## Resampling results across tuning parameters:
##
##   k  Accuracy   Kappa
##   5  0.7383399  0.3751215
##   7  0.7451938  0.3759583
##   9  0.7612841  0.4208203
##
## Accuracy was used to select the optimal model using the largest value.
## The final value used for the model was k = 9.
```

```r
# svm
svm.model <- train(diabetes ~ ., data = pimatrain, method = "svmLinearWeights",
                   trControl = trainControl(method = "repeatedcv", repeats = 5))

svm.model
```

```
## Linear Support Vector Machines with Class Weights
##
## 295 samples
##   8 predictor
##   2 classes: 'neg', 'pos'
##
## No pre-processing
```

```
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 266, 265, 265, 266, 266, 265, ...
## Resampling results across tuning parameters:
##
##   cost  weight  Accuracy   Kappa
##   0.25  1       0.7894614  0.4992388
##   0.25  2       0.7787406  0.5220056
##   0.25  3       0.7440903  0.4780240
##   0.50  1       0.7832742  0.4871205
##   0.50  2       0.7793612  0.5221001
##   0.50  3       0.7454220  0.4804341
##   1.00  1       0.7839409  0.4895880
##   1.00  2       0.7800279  0.5241532
##   1.00  3       0.7453530  0.4801408
##
## Accuracy was used to select the optimal model using the largest value.
## The final values used for the model were cost = 0.25 and weight = 1.
```

```
#
tr.control <- trainControl(method = "repeatedcv", repeats = 5)

# Or using a single call using lapply
class.methods <- c("glm", "lda", "knn", "svmLinearWeights")
fitted.models <- lapply(class.methods, function(m) {
  if (m != "glm")
    train(diabetes ~ ., data = pimatrain, method = m, trControl = tr.control)
  else
    train(diabetes ~ ., data = pimatrain, method = m, trControl = tr.control,
          family = binomial(link = "logit"))
})
names(fitted.models) <- class.methods
```

From the training data, it seems that the SVM, LDA and logistic models give the best accuracy at around 78~79% while the kNN lags behind slightly at 76%.

3.3 Assess on Test data

Using the trained classification models, classify the test set data and Compare their test set accuracies.

```
# Individual accuracies can be computed like the following in Logistic regression
logit.pred <- predict(logit.model, newdata = pimatest)
# use the helper function from caret that computes the classifier metrics
logit.confusion <- confusionMatrix(logit.pred, pimatest$diabetes, positive = "pos")


# Alternatively, all the accuracies can be computed
metrics <- lapply(fitted.models, function(mod) {
  predictions <- predict(mod, newdata = pimatest)
  confusionMatrix(predictions, pimatest$diabetes)
})
names(metrics) <- c("Logistic", "LDA", "kNN", "SVM")
overall <- vapply(metrics, "[[", numeric(7), what = "overall")
overall
```

```
##                 Logistic        LDA       kNN        SVM
## Accuracy       0.74226804 0.75257732 0.6804124 0.74226804
```

```
## Kappa          0.41240611 0.44038462 0.2595420 0.41240611
## AccuracyLower  0.64346836 0.65458083 0.5779888 0.64346836
## AccuracyUpper  0.82575179 0.83459178 0.7714775 0.82575179
## AccuracyNull   0.67010309 0.67010309 0.6701031 0.67010309
## AccuracyPValue 0.07813444 0.05013472 0.4618798 0.07813444
## McnemarPValue  1.00000000 1.00000000 0.7194375 1.00000000
```

Assessing the models on the test data instead of the training data we can see that all models have similar performance where again the three classification techniques of Logistic, LDA and SVM getting similar test accuracies around 75% while the kNN model has slightly worse accuracy at 68%.

4 Visualize the boundaries created by the classifiers

Consider only two features for predictors for ease of visualization. Construct models for `kNN`, logistic regression and SVM to classify the `diabetes` response based on the predictors `glucose` and `mass`.

4.1 Linear decision boundaries

In particular, plot the data of `mass` against `glucose` and colour the points by the `diabetes` labels. Then add to your plot the decision boundaries for a logistic regression and linear SVM using only the `mass` and `glucose` predictors. (Assume for logistic regression that the decision boundary is determined using a cutoff of 0.5 for the predicted probabilities).

Solution

$$\log(p/(1-p)) = \beta_0 + \beta_1 X_1 + \beta_2 X_2$$

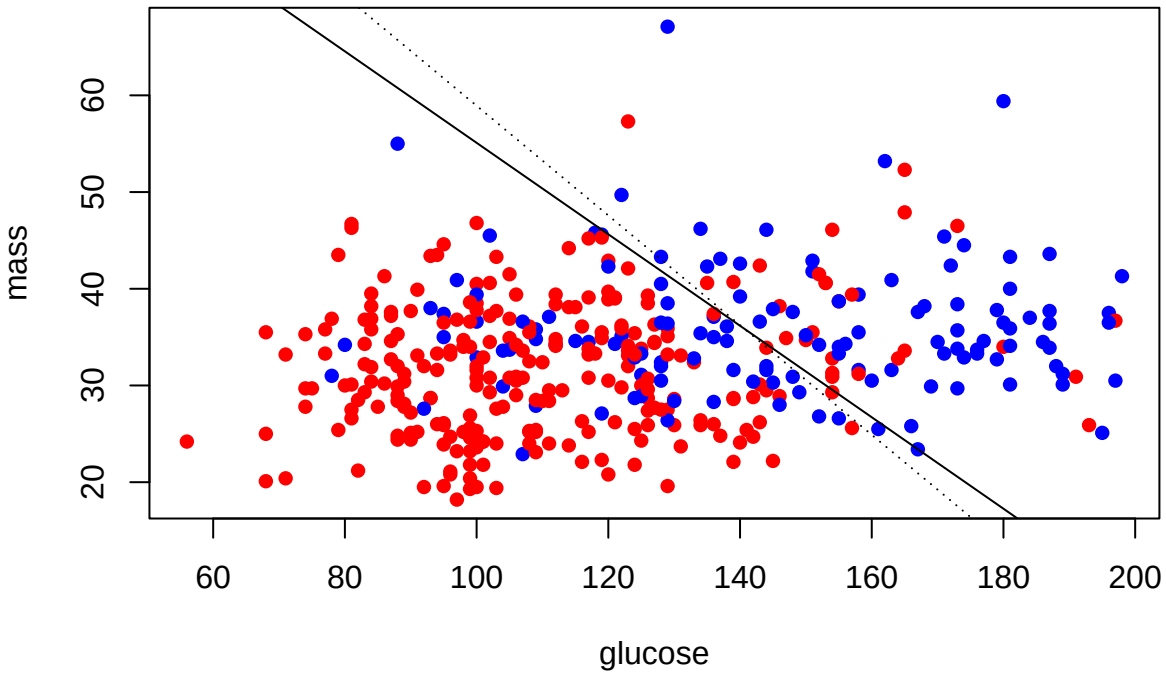
Re-arranging, when $p = 1/2$ we have, $\log(p/(1-p)) = 0$ and

$$X_2 = -\frac{\beta_0}{\beta_2} - \frac{\beta_1 X_1}{\beta_2}$$

```
logit.model2 <- glm(diabetes ~ glucose + mass, data = complete.pimas, family = binomial(link = "logit"))

logit.coefs <- coefficients(logit.model2)
logit.slope <- -logit.coefs["glucose"] / logit.coefs["mass"]
logit.intercept <- -logit.coefs["(Intercept)"] / logit.coefs["mass"]

# Using the same parameter search as earlier
svm.model2 <- train(diabetes ~ glucose + mass, data = pimatrain, method = "svmLinearWeights", scale = FA
                    trControl = trainControl(method = "repeatedcv", repeats = 5))
svm.coefs.linear <- coef(svm.model2$finalModel)
svm.linear.slope <- -svm.coefs.linear["glucose"] / svm.coefs.linear["mass"]
svm.linear.intercept <- -svm.coefs.linear["(Intercept)"] / svm.coefs.linear["mass"]
cols <- ifelse(complete.pimas$diabetes == "pos", "blue", "red")
plot(mass ~ glucose, data = complete.pimas, col = cols, pch = 16)
abline(logit.intercept, logit.slope, lty = "dotted")
abline(svm.linear.intercept, svm.linear.slope)
```

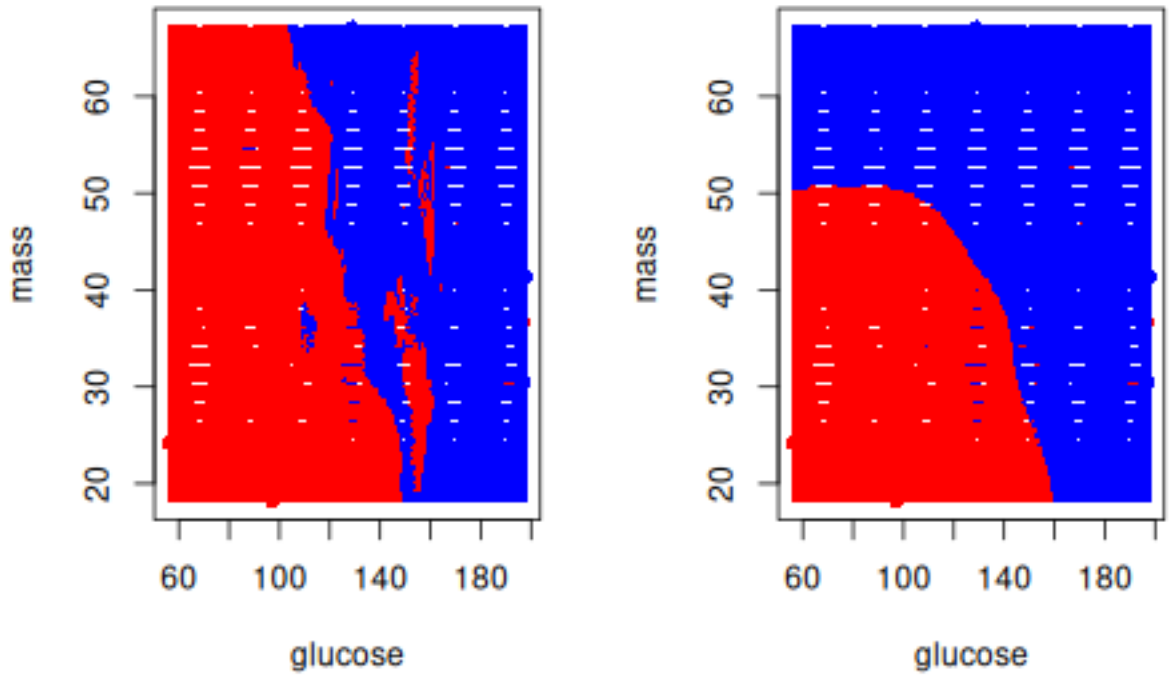


4.2 Nonlinear decision boundaries

Generate the regions for the `kNN` method and support vector machines using a radial kernel and comment on the differences between the generated boundaries.

Solution

```
viz.knn.model <- train(diabetes ~ glucose + mass, data = pimatrain, method = "knn",
                       prob = TRUE,
                       trControl = trainControl(method = "repeatedcv", repeats = 5))
# mapping decision boundary
relevant.dat <- complete.pimas %>% dplyr::select(glucose, mass)
grids <- lapply(relevant.dat, function(x) seq(from = min(x), to = max(x), length.out = 128))
viz.dat <- expand.grid(grids)
knn.predictions <- predict(viz.knn.model, viz.dat)
svm.radial.model <- train(diabetes ~ mass + glucose, data = pimatrain,
                          method = "svmRadialWeights",
                          trControl = trainControl(method = "repeatedcv", repeats = 5))
svm.predictions <- predict(svm.radial.model, viz.dat)
par(mfrow = c(1, 2))
plot(mass ~ glucose, data = complete.pimas, col = cols, pch = 16)
points(viz.dat, col = ifelse(knn.predictions == "neg", "red", "blue"), cex = 0.3, pch = 16)
plot(mass ~ glucose, data = complete.pimas, col = cols, pch = 16)
points(viz.dat, col = ifelse(svm.predictions == "neg", "red", "blue"), cex = 0.3, pch = 16)
```



Both generated decision regions are nonlinear and non-parametric. However, the `kNN` fits are much more volatile since they are more readily impacted by lone points in this case. The radial kernel can adapt non-linearly but also retains smoothness.