Lecture #14: More on Classification

CS 109A, STAT 121A, AC 209A: Data Science

Weiwei Pan, Pavlos Protopapas, **Kevin Rader**

Fall 2016

Harvard University

Announcements

- **Homework Grading**: we are returning to HW grading (regrading) now that the midterm is past
- **Extra Online OHs**: DCE students, we will be announcing extra Zoom hours starting next year.
- **Course Grades**: we will consider improvement throughout the term when determining final course grades.
- **Midterm II Details**: In-class or take-home? In-class or take-home? In-class or take-home?
- R^2 review.

Quiz Time

Time for Quiz...password is **rafisher**

Lecture Outline

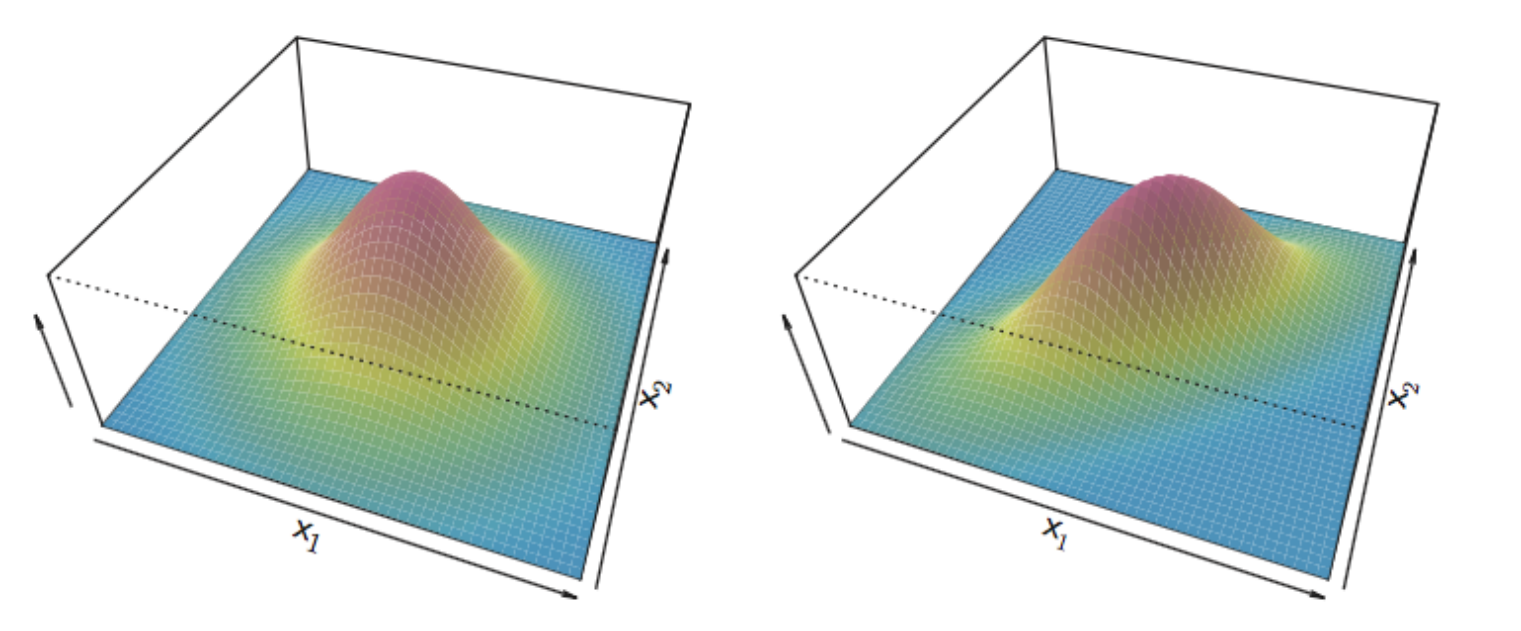
LDA for p $>$ 1

QDA

Comparison of Classification Methods

Recall the MVN distribution

Here is a visualization of the Multivariate Normal distribution with 2 variables:



MVN distribution

The joint PDF of the Multivariate Normal distribution, $\vec{X} \sim MVN(\vec{\mu}, \mathbf{\Sigma})$, is:

$$f(\vec{x}) = \frac{1}{2\pi^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x}-\vec{\mu})^T\mathbf{\Sigma}^{-1}(\vec{x}-\vec{\mu})\right)$$

where $\vec{x}$ is a p dimensional vector and $|\mathbf{\Sigma}|$ is the determinant of the $p \times p$ covariance matrix.

Let's do a quick dimension analysis sanity check...

What do $\vec{\mu}$ and $\mathbf{\Sigma}$ look like?

LDA for $p > 1$ (cont.)

Discriminant analysis in the multiple predictor case assumes the set of predictors for each class is then multivariate Normal: $\vec{X} \sim MVN(\vec{\mu}_k, \mathbf{\Sigma}_k)$.

Just like with LDA for one predictor, we make an extra assumption that the covariances are equal in each group, $\mathbf{\Sigma}_1^2 = \mathbf{\Sigma}_2^2 = ... = \mathbf{\Sigma}_K^2$ in order to simplify our lives.

Now plugging this assumed likelihood into the Bayes' formula (to get the posterior) results in:

$$P(Y = k|\vec{X} = \vec{x}) = \frac{\pi_k \frac{1}{2\pi^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_k)^T \mathbf{\Sigma}^{-1}(\vec{x} - \vec{\mu}_k)\right)}{\sum_{j=1}^{K} \frac{1}{2\pi^{p/2}|\mathbf{\Sigma}|^{1/2}} \exp\left(-\frac{1}{2}(\vec{x} - \vec{\mu}_j)^T \mathbf{\Sigma}^{-1}(\vec{x} - \vec{\mu}_j)\right)}$$

LDA for $p > 1$ (cont.)

Then doing the same steps as before (taking log and maximizing), we see that the classification will for an observation based on its predictors, $\vec{x}$, will be the one that maximizes (maximum of K of these $\delta_k(\vec{x})$):

$$\delta_k(\vec{x}) = \vec{x}^T \mathbf{\Sigma}^{-1} \vec{\mu}_k - \frac{1}{2} \vec{\mu}_k^T \mathbf{\Sigma}^{-1} \vec{\mu}_k + \log \pi_k$$

Note: this is just the vector-matrix version of the formula we saw in last lecture:

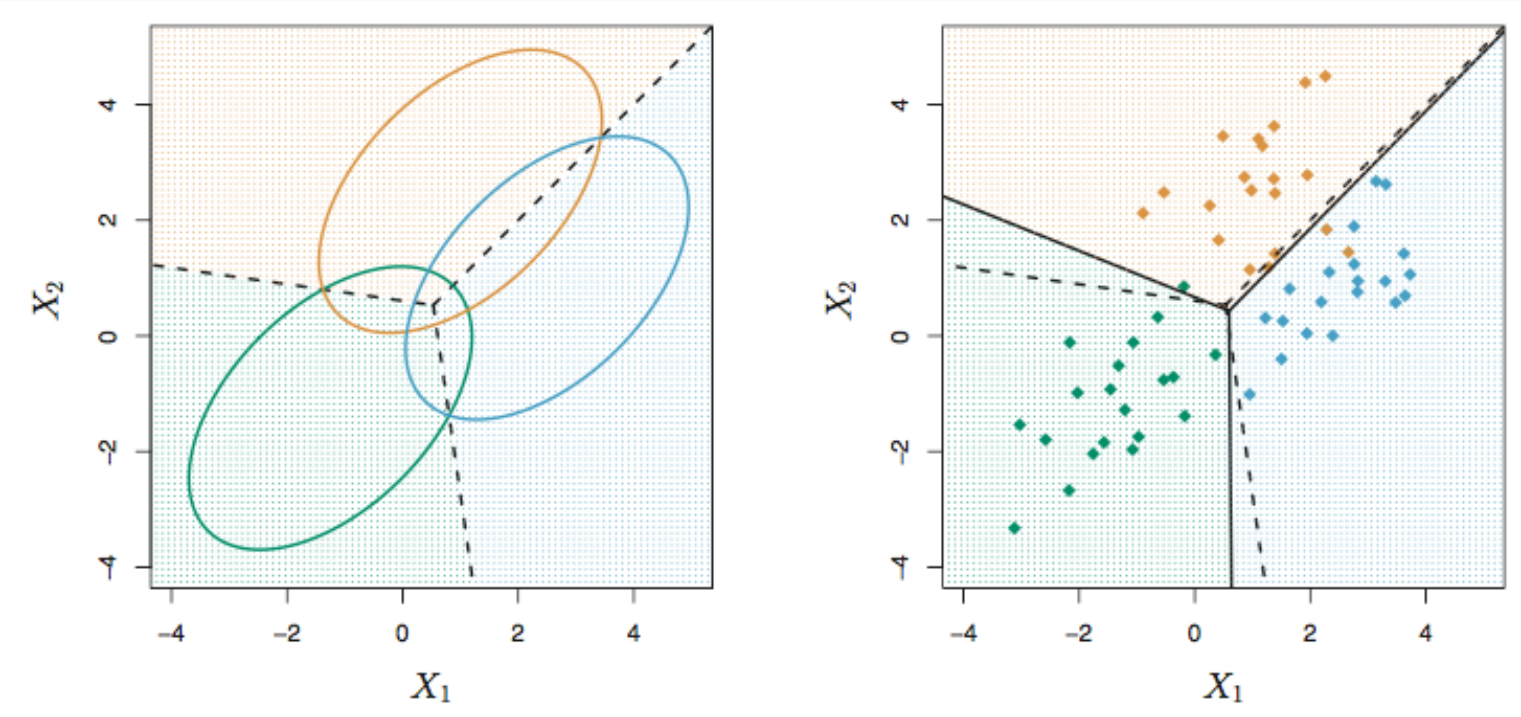
$$\delta_k(x) = x \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log \pi_k$$

What do we have to estimate now with the vector-matrix version? How many parameters are there?

There are pK means, pK variances, K prior proportions, and $\binom{p}{2} = \frac{p(p-1)}{2}$ covariances to estimate.

LDA when $p > 1$

The linear discriminant nature of LDA still holds when $p > 1$ (and when $K > 2$ for that matter as well). A picture can be very illustrative:



Lecture Outline

LDA for $p > 1$

QDA

Comparison of Classification Methods

Quadratic Discriminant Analysis (QDA)

A generalization to linear discriminant analysis is quadratic discriminant analysis (QDA).

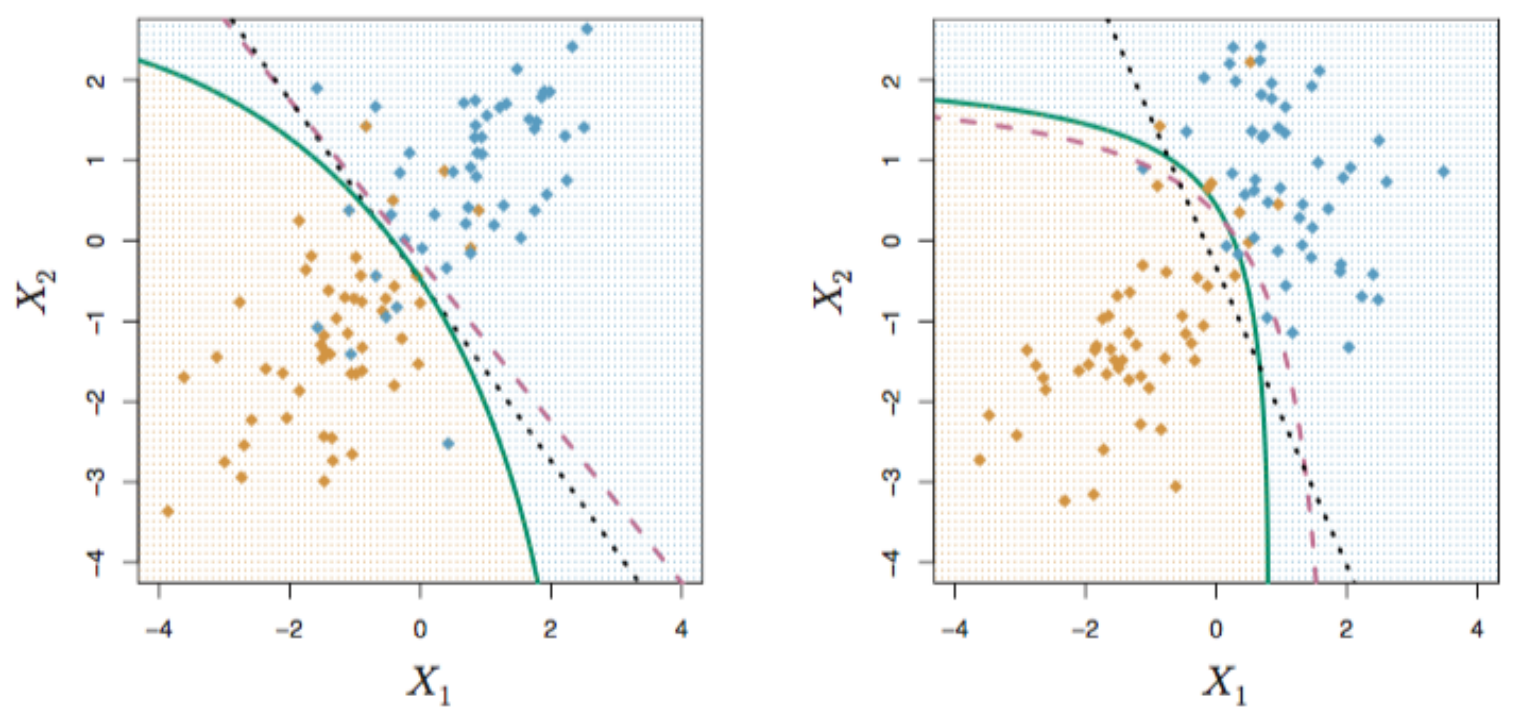
Why do you suppose the choice in name?

The implementation is just a slight variation on LDA. Instead of assuming the covariances of the MVN distributions within classes are equal, we instead allow them to be different.

This relaxation of an assumption completely changes the picture...

QDA in a picture

A picture can be very illustrative:



QDA (cont.)

When performing QDA, performing classification for an observation based on its predictors $\vec{x}$ is equivalent to maximizing the following over the K classes:

$$\delta_k(\vec{x}) = -\frac{1}{2}\vec{x}^T\mathbf{\Sigma}_k^{-1}\vec{x} + \vec{x}^T\mathbf{\Sigma}_k^{-1}\vec{\mu}_k - \frac{1}{2}\vec{\mu}_k^T\mathbf{\Sigma}_k^{-1}\vec{\mu}_k - \frac{1}{2}\log|\mathbf{\Sigma}_k| + \log\pi_k$$

Notice the 'quadratic form' of this expression. Hence the name QDA.

Now how many parameters are there to be estimated?

There are pK means, pK variances, K prior proportions, and $\binom{p}{2}K = \left(\frac{p(p-1)}{2}\right)K$ covariances to estimate. This could slow us down very much is K is large...

Discriminant Analysis in Python

LDA is already implemented in Python via the `sklearn.discriminant_analysis` package through the `LinearDiscriminantAnalysis` function.

QDA is in the same package and is the `QuadraticDiscriminantAnalysis` function.

It's very easy to use. Let's see how this works

Discriminant Analysis in Python

```
#read in the GSS data
gssdata = pd.read_csv("gsspartyid.csv")
print(gssdata[1:5])
```

```
  politicalparty  income  educ  abortion  republican
1     Republican     906     6         0           1
2       Democrat    1373     6         0           0
3       Democrat    1941     4         0           0
4       Democrat     355     7         0           0
```

```
LDA = da.LinearDiscriminantAnalysis()
X = gssdata[["income","educ","abortion"]]
model_LDA = LDA.fit(X,gssdata['republican'])
print("Specificity is",np.mean(model_LDA.predict(X[gssdata['republican']==0])))
print("Sensitivity is",1-np.mean(model_LDA.predict(X[gssdata['republican']==1])))
print("False positive rate is",np.mean(gssdata['republican'][model_LDA.predict(X)==1]))
print("False negative rate is",1-np.mean(gssdata['republican'][model_LDA.predict(X)==0]))
```

```
Specificity is 0.388387096774
Sensitivity is 0.374060150376
False positive rate is 0.5252365930599369
False negative rate is 0.7043090638930163
```

QDA vs LDA

So both QDA and LDA take a similar approach to solving this classification problem: they use Bayes' rule to flip the conditional probability statement and assume observations within each class are multivariate Normal (MVN) distributed.

QDA differs in that it does not assume a common covariance across classes for these MVNs. What advantage does this have? What disadvantage does this have?

QDA vs LDA (cont.)

So generally speaking, when should QDA be used over LDA? LDA over QDA?

The extra covariance parameters that need to be estimated in QDA not only slow us down, but also allow for another opportunity for overfitting. Thus if your training set is small, LDA should perform better for 'out-of-sample prediction', aka, predicting future observations (how do we mimic this process?)

Lecture Outline

LDA for $p > 1$

QDA

Comparison of Classification Methods

A Comparison of Methods

We have seen 4 major methods for doing classification:

1. k-NN
2. Logistic Regression
3. LDA
4. QDA

For a specific problem, which approach should be used?

Well of course, it depends on the nature of the data. So how should we decide?

Visualize the data!

Six Classification Models We'll Compare

Let's investigate which method will work the best (as measured by lowest overall classification error rate), by considering 6 different models for 4 different data sets (each data set as a pair of predictors...you can think of them as the first 2 PCA components). The 6 models to consider are:

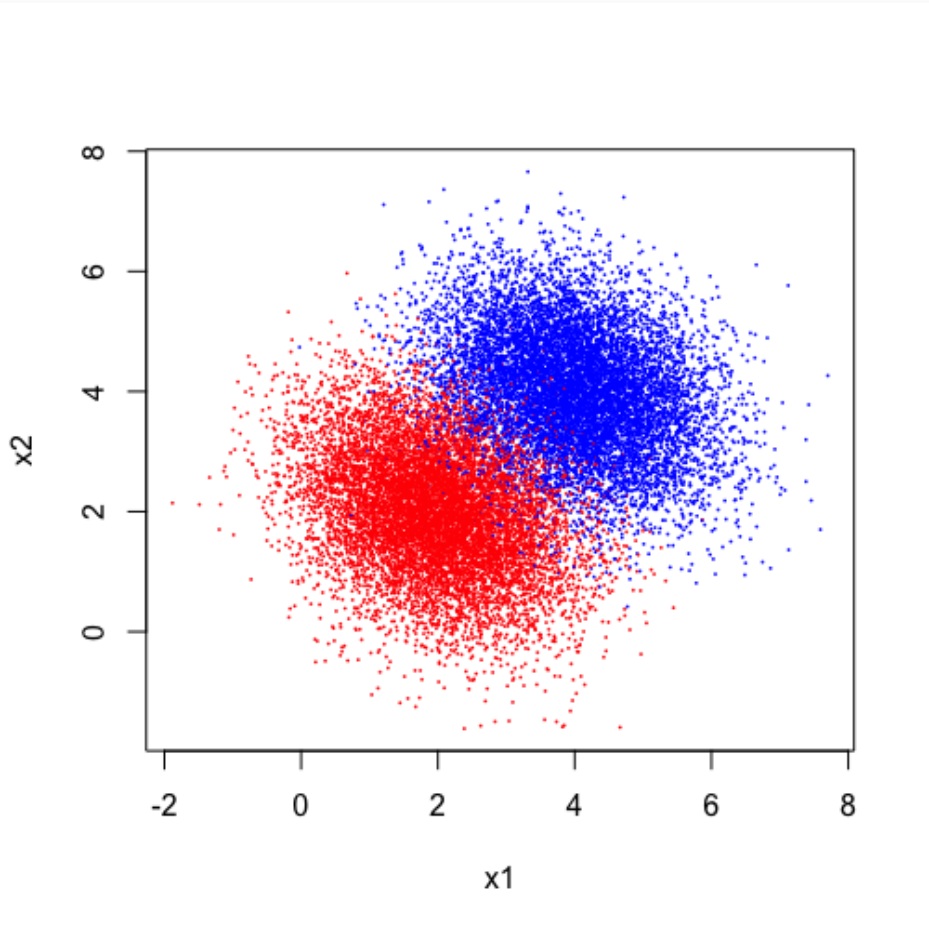
1. A logistic regression with only 'linear' main effects
2. A logistic regression with only 'linear' and 'quadratic' effects
3. LDA
4. QDA
5. k-NN where $k = 3$
6. k-NN where $k = 25$

What else will also be important to measure (besides error rate)?

Which method should perform better? #1

$n = 20,000,\ p = 2,\ K = 2,$

$\pi_1 = \pi_2 = 0.5$



method	misclass rate	run time (ms)
logit1	0.04410	417.95
logit2	0.04405	229.71
lda	0.04425	50.63
qda	0.04410	49.08
knn3	0.05225	1856.11
knn25	0.04500	2166.57

Notice anything fishy about out answers? What did Kevin do? What should he have done?

Easy to implement in Python

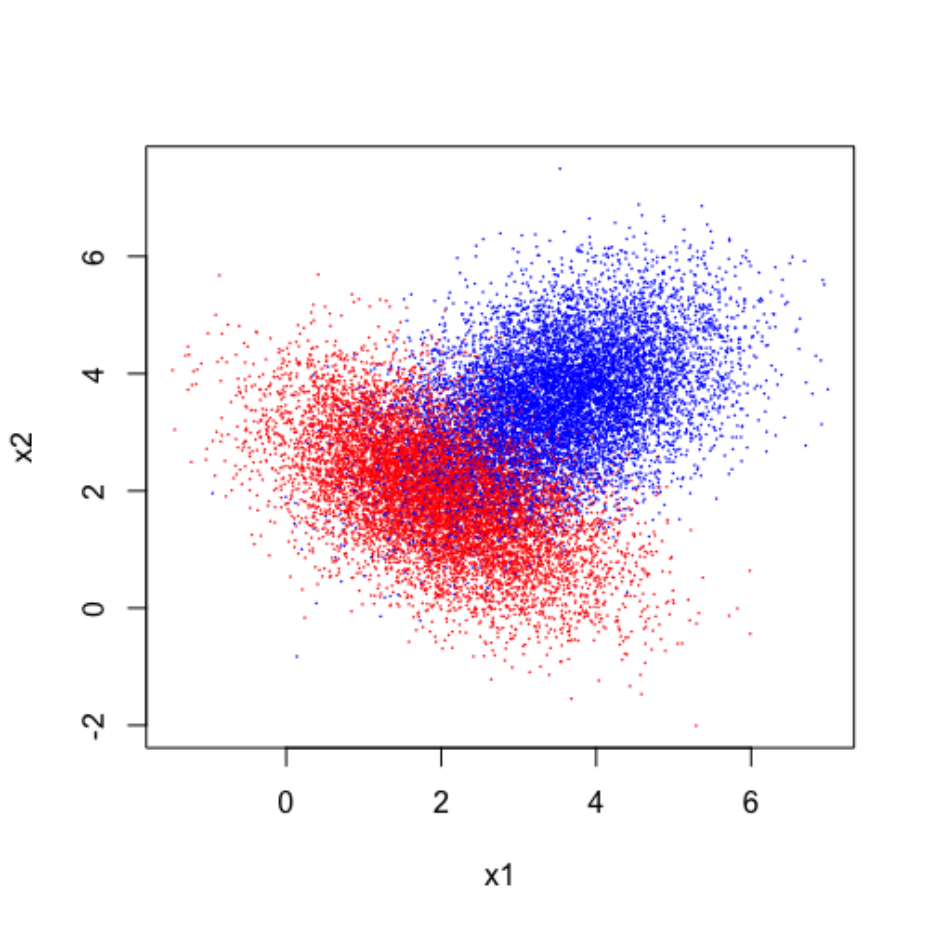
```
lda = da.LinearDiscriminantAnalysis()
qda = da.QuadraticDiscriminantAnalysis()
lda.fit(X,y)
qda.fit(X,y)
logit2 = sk.linear_model.LogisticRegression(C = 1000000)
logit1 = sk.linear_model.LogisticRegression(C = 1000000)
logit1.fit(X,y)
logit2.fit(X2,y)

print("Overall misclassification rate of Logit1 is",(1-logit1.score(X,y)))
print("Overall misclassification rate of Logit2 is",(1-logit2.score(X2,y)))
print("Overall misclassification rate of LDA is",(1-lda.score(X,y)))
print("Overall misclassification rate of QDA is",(1-qda.score(X,y)))
```

```
Overall misclassification rate of Logit1 is 0.0441
Overall misclassification rate of Logit2 is 0.0441
Overall misclassification rate of LDA is 0.04425
Overall misclassification rate of QDA is 0.0441
```

Which method should perform better? #2

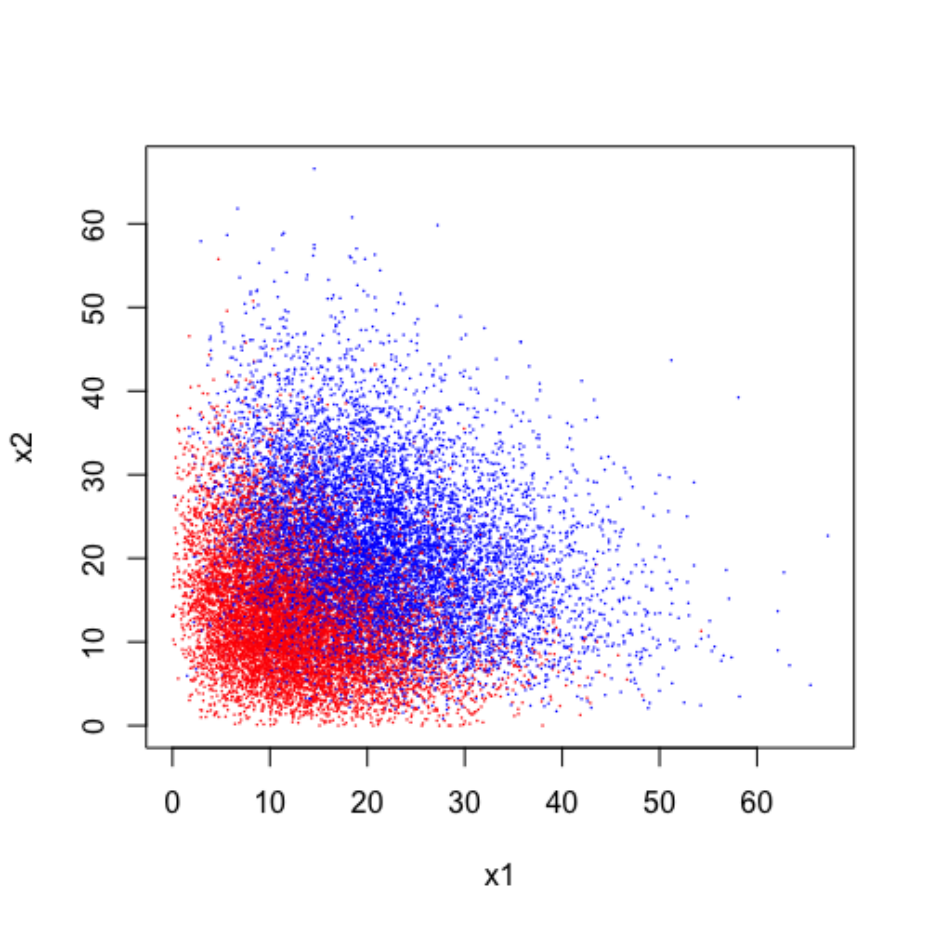
$$n = 20,000,\ p = 2,\ K = 2,$$
$$\pi_1 = \pi_2 = 0.5$$



method	misclass rate	run time (ms)
logit1	0.12230	169.53
logit2	0.11860	196.42
lda	0.12215	47.93
qda	0.11445	47.03
knn3	0.14380	1861.90
knn25	0.12015	2223.13

Which method should perform better? #3

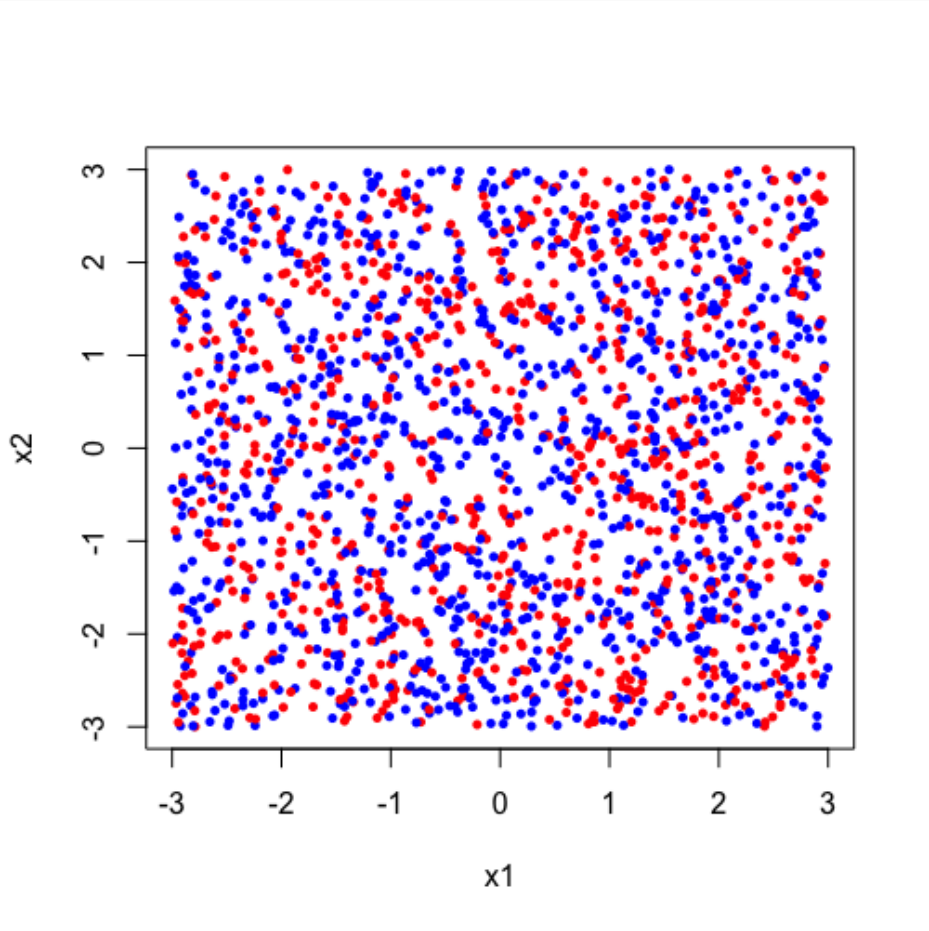
$$n = 20,000,\ p = 2,\ K = 2,$$
$$\pi_1 = \pi_2 = 0.5$$



method	misclass rate	run time (ms)
logit1	0.20260	1234.35
logit2	0.19535	192.99
lda	0.20320	49.08
qda	0.21450	60.61
knn3	0.23300	1869.44
knn25	0.20270	2166.77

Which method should perform better? #4

$$n = 20,000,\ p = 2,\ K = 2,$$
$$\pi_1 = \pi_2 = 0.5$$



method	misclass rate	run time (ms)
logit1	0.45690	1181.44
logit2	0.37880	147.95
lda	0.45770	51.06
qda	0.40705	44.04
knn3	0.34820	1835.42
knn25	0.30655	2126.38

Summary of Results

Generally speaking:

1. LDA outperforms Logistic Regression if the distribution of predictors is reasonably MVN (with constant covariance).
2. QDA outperforms LDA if the covariances are not the same in the groups.
3. k-NN outperforms the others if the decision boundary is extremely non-linear.
4. Of course, we can always adapt our models (logistic and LDA/QDA) to include polynomial terms, interaction terms, etc... to improve classification (watch out for overfitting!)
5. In order of computational speed (generally speaking, it depends on K, p, and n of course):

$$\text{LDA} > \text{QDA} > \text{Logistic} > k\text{-NN}$$