BIOS 635: Naive Bayes

Kevin Donovan

1/28/2021

Review

- Homework 3 due on 2/12 at 11PM through GitHub Classroom
- Article Evaluation 1 assigned, due on 2/9 through GitHub Classroom
- Last lecture: linear and quadratic discriminant analysis

Discriminant Analysis

Posterior Class Probability:

$$\Pr(Y = k|X = x) = \frac{f(x|k) * \Pr(Y = k)}{\sum_{l=1}^{K} f(x|l) * \Pr(Y = l)}$$

Prediction Rule: Given features x_0

$$\hat{y_0} = \underset{k=1,\ldots,K}{\operatorname{argmax}} \frac{f(x_0|k) * \Pr(Y = k)}{\sum_{l=1}^{K} f(x_0|l) * \Pr(Y = l)} = \underset{k=1,\ldots,K}{\operatorname{argmax}} f(x_0|k) * \Pr(Y = k)$$

LDA and QDA

$$\Pr(Y = k|X = x) = \frac{f(x|k) * \Pr(Y = k)}{\sum_{l=1}^{K} f(x|l) * \Pr(Y = l)}$$

- Linear discriminant analysis (LDA)
  - *Within classes, features are mutivariate normally distributed with **same** covariance structures* $\leftrightarrow$
  - $f(x|k) \sim \text{Multivariate Normal}(\mu_k, \Sigma)$ *for* $k = 1, \ldots, K$
- Quadratic discriminant analysis (QDA)
  - *Within classes, features are mutivariate normally distributed with **possibly different** covariance structures* $\leftrightarrow$
  - $f(x|k) \sim \text{Multivariate Normal}(\mu_k, \Sigma_k)$ *for* $k = 1, \ldots, K$

LDA vs logistic regression

For two-class prediction problem, can show for LDA, discriminant function $\rightarrow$

$$\log\left(\frac{\Pr(Y=1|X=x)}{\Pr(Y=0|X=x)}\right) = c_0 + c_1 x_1 + \ldots + c_p x_p$$

and from logistic regression:

$$\text{logit}(p) = \log\left(\frac{\Pr(Y=1|X=x)}{\Pr(Y=0|X=x)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

so the two have the same form.

LDA vs logistic regression

Difference = **how parameters are estimated**

- Logistic regression uses conditional likelihood based on $\Pr(Y = 1|X)$, denoted *discriminative learning*
- LDA uses **full likelihood** based on $f(x, y)$ with Bayes Rule, denoted *generative learning*
- However, in practice results often very similar

LDA vs logistic regression differences

- When classes are **well separated**, i.e. $\Pr(Y = 1|X)$ near 0 or 1, logsitic regression coefficients are **very unstable**. Not the case with LDA
- LDA makes assumptions on the distribution of $X|Y = k$ (Normality, same covariance)
  - *When assumptions hold, LDA can produce more stable decision boundaries, even with small n*
- When $K = 2$, due to a lack of assumptions on the distribution of $X|Y = k$, logisitic regression can be extended in many kinds of ways and is highly interpretable
  - *Hard to interpret/implement well for $K > 2$ classes. LDA is largely the same implementation-wise regardless of K*

Naive Bayes

Recall: Posterior class probability general form

$$\Pr(Y = k|X = x) = \frac{f(x|k) * \Pr(Y = k)}{\sum_{l=1}^{K} f(x|l) * \Pr(Y = l)}$$

Discriminant analysis: alter form of $f(x|k)$ to get new method

- With normal distribution but different Σ_k in each class $\rightarrow$ QDA
- Keeping normal distribution assumption, let's assume **features are independent** in each class
  - *This is **not** assuming features are marginally independent, i.e. independent in the entire population*
  - *Denoted conditional independence*
  - *For multivariate normal, this means Σ_k, **within-class feature correlations** are 0*
  - *In the likelihood, formulation is simplified as Σ_k are diagonal*
  - *Method denoted as Naive Bayes*

Naive Bayes

- Simple structure very useful when number of predictors p is large
- Ex. imagine $p = 1000$ and $n = 2000$. With LDA and QDA, estimating within-class correlations of features very difficult, even if equal correlations across classes is assumed (LDA)
  - *Instead, Naive Bayes assumes components of $X = (X_1, \ldots, X_p)$ are independent*
- Under independence, within-class feature distriubtion simplifies to:

$$\begin{aligned} f(x|k) &= f(x_1, x_2, \ldots, x_p|k) \\ &= f(x_1|k) * \ldots * f(x_p|k) \\ &= \prod_{j=1}^{p} f(x_j|k) \end{aligned}$$

- Multivariate densities go away, only need to estimate p *univariate* densities $f(x_j|k)$

Naive Bayes

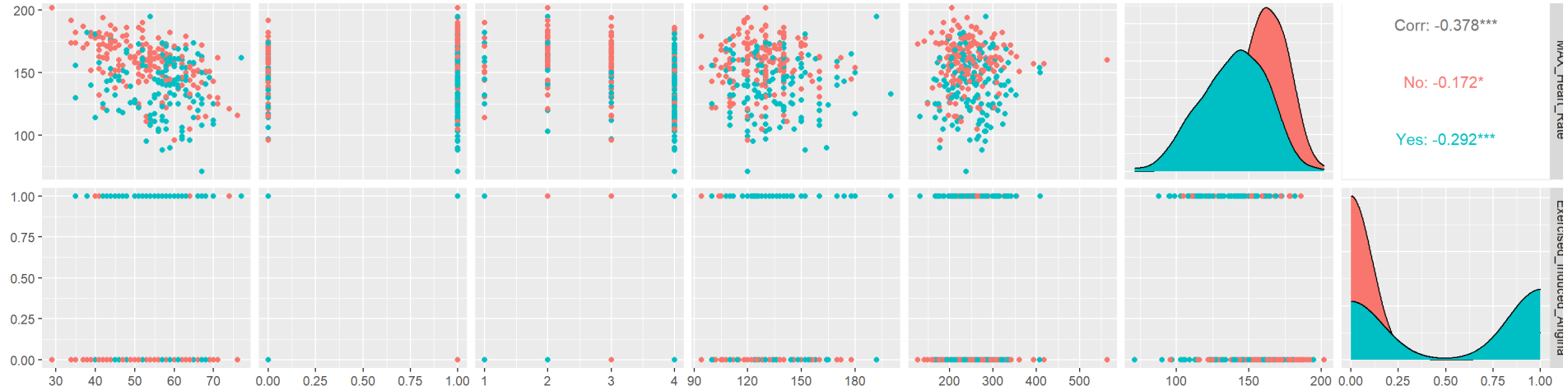
- Assume each feature is normally distributed in each class
  - *Discriminant function:*

$$\begin{aligned}\delta_k(x) &\propto \log\Big[\Pr(Y=k)f(x|k)\Big] \\ &= \Pr(Y=k)\prod_{j=1}^{p} f(x_j|k) \text{ by independent features within class} \\ &= -\frac{1}{2}\sum_{j=1}^{p}\Big[\frac{(x_j-\mu_{kj})^2}{\sigma_{kj}^2}+\log(\sigma_{kj}^2)\Big]+\log[\Pr(Y=k)] \text{ by independent normality within class}\end{aligned}$$

- **Notice**: Use of independence provides more flexibility on choice of distribution for each feature within the class
  - $\rightarrow$ *Naive Bayes can be used for mixed feature vectors (qualitative **and** quantitative). If quantitative, model $f(x_j|k)$ using probability mass function*
  - *Can also set other continuous densities for specific features (ex. variable is skewed within classes)*
- Distributional flexibility comes at cost of independence assumption

Naive Bayes

- Ex. Heart disease prediction
- **Recall**: Had non-normal/categorical features


Corr: -0.378***
No: -0.172*
Yes: -0.292***
MAX_Heart_Rate
Exercised_Induced_Angina
200
150
100
1.00
0.75
0.50
0.25
0.00
30 40 50 60 70
0.00 0.25 0.50 0.75 1.00
1 2 3 4
90 120 150 180
200 300 400 500
100 150 200
0.00 0.25 0.50 0.75 1.00

Naive Bayes

- Ex. Heart disease prediction
- **Recall**: Had non-normal/categorical features
- **Note**: Need R package `klaR` to do in `caret`

```
# Need to set categorical values to "factor" to get correct modeling of densities
heart_data <-
  heart_data %>%
  mutate(Sex=factor(Sex),
         Chest_Pain=factor(Chest_Pain),
         Exercised_Induced_Angina=factor(Exercised_Induced_Angina))

# Partition Data
set.seed(12)
train_test_indices <- createDataPartition(heart_data$heart_disease, p=0.6, list = FALSE)
heart_data_train <- heart_data[train_test_indices,]
heart_data_test <- heart_data[-train_test_indices,]

# Train
nb_fit <- train(heart_disease~Age+Sex+Chest_Pain+Resting_Blood_Pressure+Colestrol+
                MAX_Heart_Rate+Exercised_Induced_Angina,
                data = heart_data_train, method = "nb")

# Add in test set predictions
heart_data_test$estimated_prob_heart_disease <-
  predict(nb_fit, newdata=heart_data_test, type = "prob")$Yes

heart_data_test <-
  heart_data_test %>%
  mutate(pred_heart_disease =
           relevel(factor(ifelse(estimated_prob_heart_disease>0.5, "Yes", "No")),
                   ref = "No"))

# Compute confusion matrix
confusionMatrix(data = heart_data_test$pred_heart_disease,
                reference = heart_data_test$heart_disease,
                positive = "Yes")
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction No Yes
```

```
##        No  44  11
##        Yes 21  44
##
##               Accuracy : 0.7333
##                 95% CI : (0.6449, 0.8099)
##    No Information Rate : 0.5417
##    P-Value [Acc > NIR] : 1.243e-05
##
##                  Kappa : 0.4703
##
##  Mcnemar's Test P-Value : 0.1116
##
##             Sensitivity : 0.8000
##             Specificity : 0.6769
##          Pos Pred Value : 0.6769
##          Neg Pred Value : 0.8000
##              Prevalence : 0.4583
##          Detection Rate : 0.3667
##    Detection Prevalence : 0.5417
##       Balanced Accuracy : 0.7385
##
##        'Positive' Class : Yes
##
```

Naive Bayes

- Ex. Heart disease prediction
- Can use `NaiveBayes` function in `klaR` package to see estimated conditional densities/distributions for each feature

```
# Observe MLEs of parameters
nb_ests <- NaiveBayes(heart_disease~Age+Sex+Chest_Pain+Resting_Blood_Pressure+Colestrol+
              MAX_Heart_Rate+Exercised_Induced_Angina,
          data=heart_data_train,
          na.action = na.omit)

nb_ests$tables
```

```
## $Age
##         [,1]     [,2]
## No  52.62626 9.725399
## Yes 56.21429 8.269659
##
## $Sex
##          var
## grouping         0         1
##      No  0.4242424 0.5757576
##      Yes 0.1904762 0.8095238
##
## $Chest_Pain
##          var
## grouping          1          2          3          4
##      No  0.10101010 0.22222222 0.43434343 0.24242424
##      Yes 0.07142857 0.04761905 0.09523810 0.78571429
##
## $Resting_Blood_Pressure
##         [,1]     [,2]
## No  128.3030 15.82652
## Yes 135.1071 18.41753
##
## $Colestrol
##         [,1]     [,2]
## No  243.4747 56.21684
## Yes 245.4048 46.99478
##
## $MAX_Heart_Rate
##         [,1]     [,2]
## No  159.0303 19.78995
## Yes 136.9286 23.49611
```

```
##
## $Exercised_Induced_Angina
##         var
## grouping         0         1
##      No  0.8888889 0.1111111
##      Yes 0.4523810 0.5476190
```

Naive Bayes

Overview:

- Training Phase:
  - *For class k, estimate $\pi_k = \Pr(Y = k)$ using $\hat{\pi_k} = \frac{1}{n}\sum_{i=1}^{n} I(Y_i = k)$*
  - *For feature X_j*
    - If X_j is continuous, estimate $\mathrm{E}(X_j|Y = k) = \mu_{x_j|k}$ and $\mathrm{Var}(X_j|Y = k) = \sigma^2_{x_j|k}$ using within-class sample mean and variance
    - If X_j is categorical, estimate $\Pr(X_j = x_{jl}|Y = k) = \theta_{jlk}$ where l is a category X_j can take
    - Do so for each class k in sample
- Testing Phase:
  - *For subject with features $x_0 = (x_{01}, \ldots, x_{0p})$, predict outcome y_0 using*

$$\hat{y_0} = \underset{k=1,\ldots,K}{\operatorname{argmax}} \hat{\pi_k} \prod_{j=1}^{p} \hat{f}\left(x_{0j}|k\right)$$

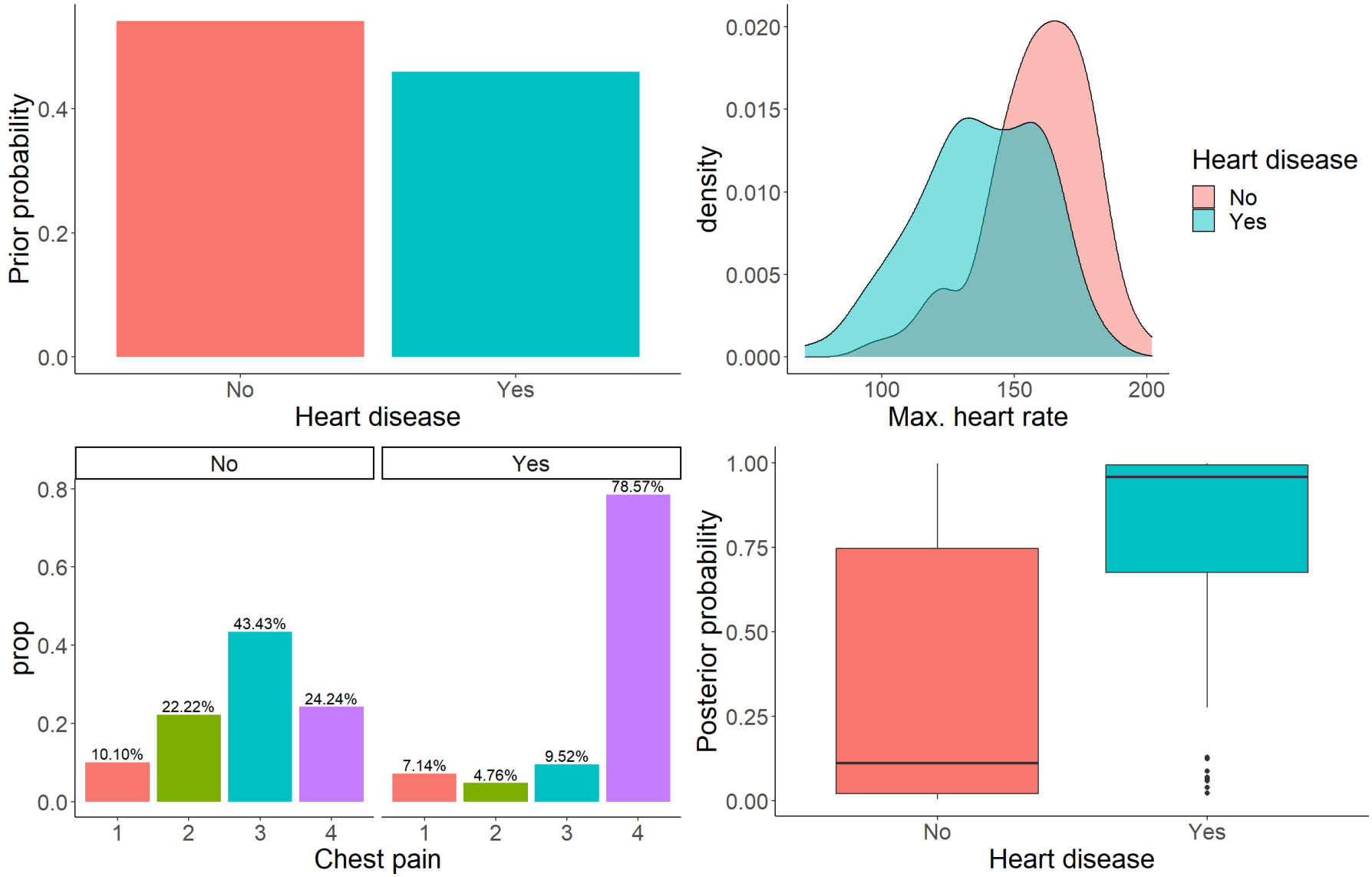
- Where $\hat{f}\left(x_{0j}|k\right)$ is obtained by plugging in
  - *conditional probabilities obtained in training if X_j is categorical*
  - *conditional means and variances obtained in training if X_j is continuous*

Naive Bayes

- Ex. Heart disease prediction

```
# Plot "prior probabilities
prior_probs <- data.frame("heart_disease"=unname(names(nb_ests$apriori)),
                          "prior_prob"=as.numeric(nb_ests$apriori))
prior_probs_plot <-
  ggplot(data=prior_probs, mapping=aes(x=heart_disease, y=prior_prob,
                                       fill=heart_disease))+
  geom_bar(stat="identity")+
  labs(x="Heart disease", y="Prior probability")+
  theme_classic()+
  theme(legend.position = "none",
        text=element_text(size=20))

# Plot density for max heart rate
max_heart_rate_density <-
  ggplot(data=heart_data_train, mapping=aes(x=MAX_Heart_Rate,
                                            fill=heart_disease))+
  geom_density(alpha=0.5)+
  labs(fill="Heart disease", x="Max. heart rate")+
  theme_classic()+
  theme(text=element_text(size=20))

# Plot distribution for Chest_Pain
chest_pain_density <-
  ggplot(data=heart_data_train, mapping=aes(x=Chest_Pain,
                                            group=heart_disease))+
  geom_bar(aes(y = ..prop.., fill = factor(..x..)), stat="count")+
  geom_text(aes( label = scales::percent(..prop..),
                 y= ..prop.. ), stat= "count", vjust = -0.25,
            size=4) +
  facet_grid(~heart_disease)+
  labs(fill="", x="Chest pain")+
  theme_classic()+
  theme(legend.position = "none",
        text=element_text(size=20))

post_probs <-
  ggplot(data=heart_data_test,
         mapping=aes(x=heart_disease, y=estimated_prob_heart_disease,
                     fill=heart_disease))+
  geom_boxplot()+
  labs(x="Heart disease", y="Posterior probability")+
  theme_classic()+
```

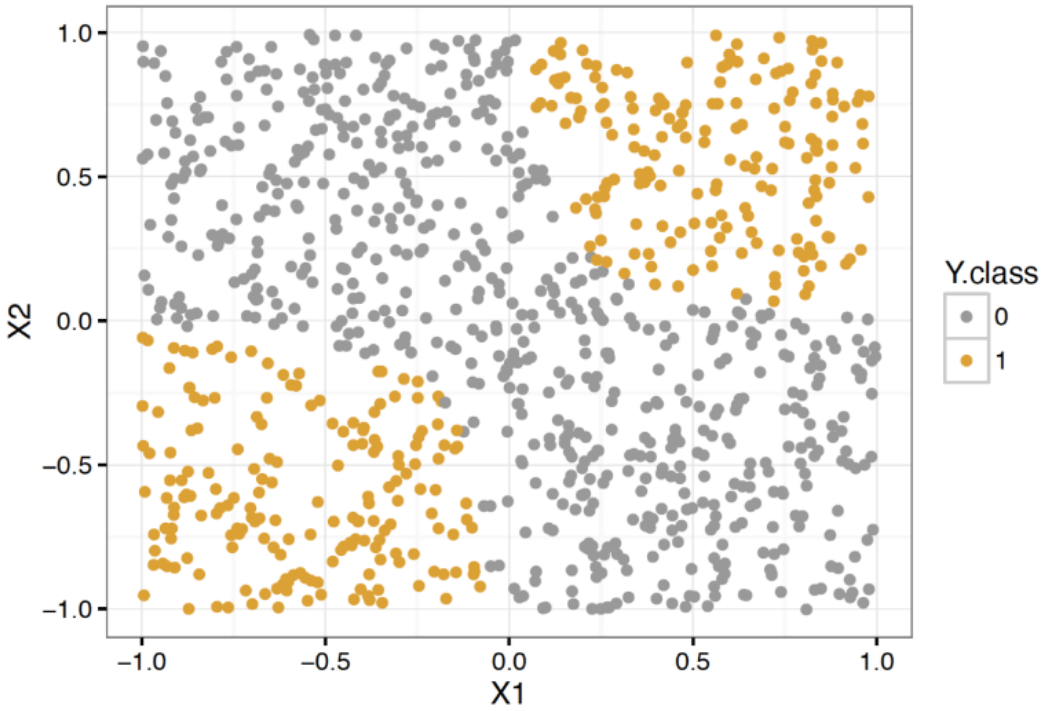
```
  theme(legend.position = "none",
        text=element_text(size=20))

# Create 4-panel plot
ggarrange(plotlist = list(prior_probs_plot, max_heart_rate_density,
                          chest_pain_density, post_probs),
          ncol=2, nrow=2)
```



Naive Bayes vs LDA vs QDA

- Naive Bayes scales well to problems where p is large
  - *Need large enough n to estimate univariate properties of each feature*
  - *With LDA have $K * p$ parameters for $\mathrm{E}(X_j|Y = k)$ and $0.5 * p * (p + 1)$ parameters for feature covariance matrix Σ*
  - *With QDA have $0.5 * K * p * (p + 1)$ parameters for the K covariance matrices Σ_k*
- However, LDA and QDA can capture and use **interactions in features** for prediction, Naive Bayes cannot

Feature interaction example



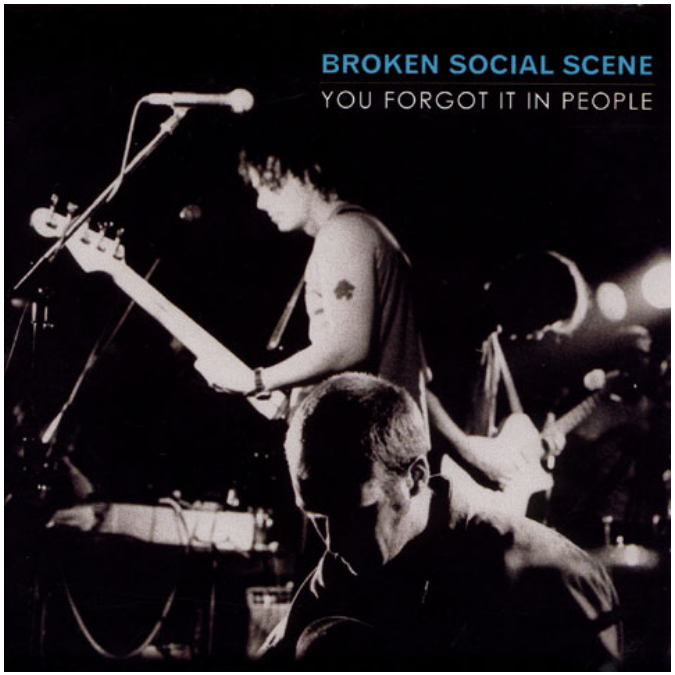
Songs of the session

Mexican Grand Prix by Mogwai

Your Hand in Mine by Explosion in the Sky

Pacific Theme by Broken Social Club

The Big Ship by Brian Eno




MOGWAI
HARDCORE WILL NEVER DIE,
BUT YOU WILL.