

CS545: Linear Models for Classification

Chuck Anderson

Department of Computer Science
Colorado State University

Fall, 2009

Linear Least Squares for Classification

- Indicator Variables
- Masking Problem
- Example

Linear Least
Squares for
Classification

- Indicator Variables
- Masking Problem
- Example

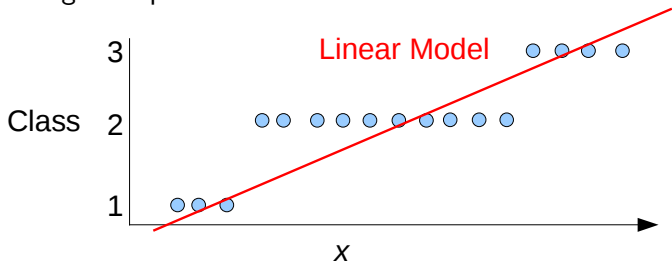
Generative Models for Classification

- QDA
- Fitting the Generative Distributions to Data
- Overfitting
- LDA
- Example

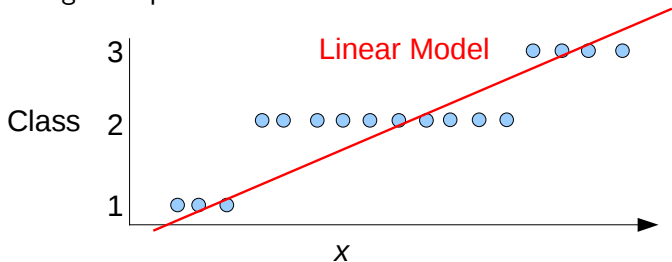
Generative Models
for Classification

- QDA
- Fitting the Generative
Distributions to Data
- Overfitting
- LDA
- Example

- To classify a sample as being a member of 1 of 3 different classes, we could use integers 1, 2, and 3 as target outputs.



- To classify a sample as being a member of 1 of 3 different classes, we could use integers 1, 2, and 3 as target outputs.



- Linear function of x seems to match data fairly well. Why is this not a good idea?

- We must convert the continuous y-axis value to discrete integers 1, 2, or 3. Without adding more parameters, we are forced to use the general solution of splitting at 1.5 and 2.5.

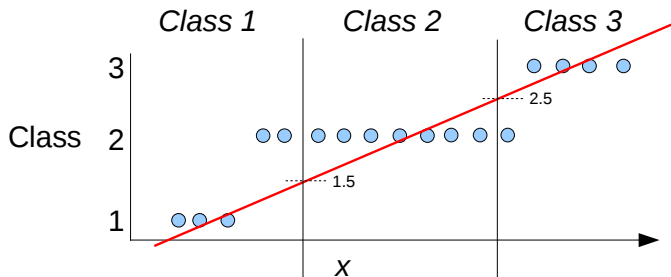
Linear Least
Squares for
Classification

Indicator Variables
Masking Problem
Example

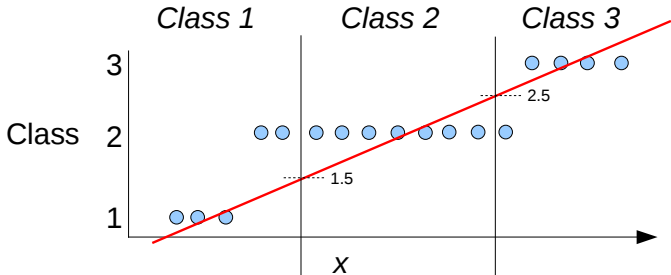
Generative Models
for Classification

QDA
Fitting the Generative
Distributions to Data
Overfitting
LDA
Example

- We must convert the continuous y-axis value to discrete integers 1, 2, or 3. Without adding more parameters, we are forced to use the general solution of splitting at 1.5 and 2.5.



- We must convert the continuous y-axis value to discrete integers 1, 2, or 3. Without adding more parameters, we are forced to use the general solution of splitting at 1.5 and 2.5.



- Rats! Boundaries are not where we want them.

Indicator Variables

- To allow flexibility, we need to decouple the modeling of the boundaries. Problem is due to using one value to represent all classes.

Indicator Variables

- To allow flexibility, we need to decouple the modeling of the boundaries. Problem is due to using one value to represent all classes.
- Instead, let's use three values, one for each class. Binary-valued variables are adequate. Class 1 = $(1, 0, 0)$, Class 2 = $(0, 1, 0)$ and Class 3 = $(0, 0, 1)$. Our linear model has three outputs now. How do we interpret the output for a new sample?

Indicator Variables

- To allow flexibility, we need to decouple the modeling of the boundaries. Problem is due to using one value to represent all classes.
- Instead, let's use three values, one for each class. Binary-valued variables are adequate. Class 1 = (1, 0, 0), Class 2 = (0, 1, 0) and Class 3 = (0, 0, 1). Our linear model has three outputs now. How do we interpret the output for a new sample?
- Let the output be $\mathbf{y} = (y_1, y_2, y_3)$. Convert these values to a class by picking the maximum value.

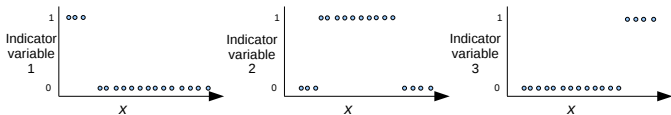
$$\text{class} = \underset{i}{\operatorname{argmax}} y_i$$

- To allow flexibility, we need to decouple the modeling of the boundaries. Problem is due to using one value to represent all classes.
- Instead, let's use three values, one for each class. Binary-valued variables are adequate. Class 1 = (1, 0, 0), Class 2 = (0, 1, 0) and Class 3 = (0, 0, 1). Our linear model has three outputs now. How do we interpret the output for a new sample?
- Let the output be $\mathbf{y} = (y_1, y_2, y_3)$. Convert these values to a class by picking the maximum value.

$$\text{class} = \underset{i}{\operatorname{argmax}} y_i$$

- Targets

- Can plot the three output components on three separate graphs. What linear functions will each one learn?



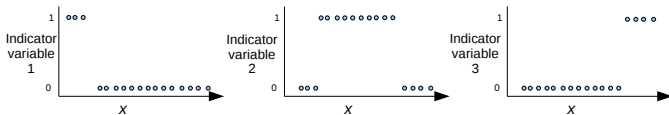
Linear Least Squares for Classification

Indicator Variables
Masking Problem
Example

Generative Models for Classification

QDA
Fitting the Generative Distributions to Data
Overfitting
LDA
Example

- Can plot the three output components on three separate graphs. What linear functions will each one learn?



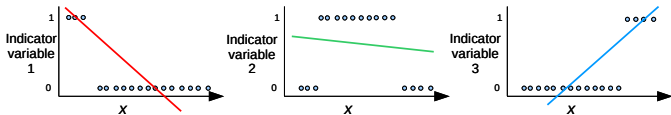
Linear Least Squares for Classification

Indicator Variables
Masking Problem
Example

Generative Models for Classification

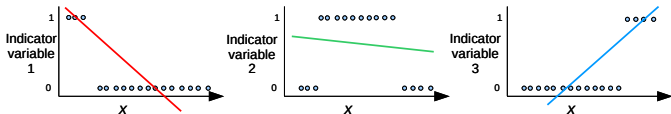
QDA
Fitting the Generative Distributions to Data
Overfitting
LDA
Example

- Can plot the three output components on three separate graphs. What linear functions will each one learn?



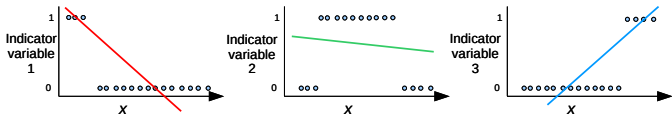
- Overlay them to see which one is the maximum for each x value.

- Can plot the three output components on three separate graphs. What linear functions will each one learn?

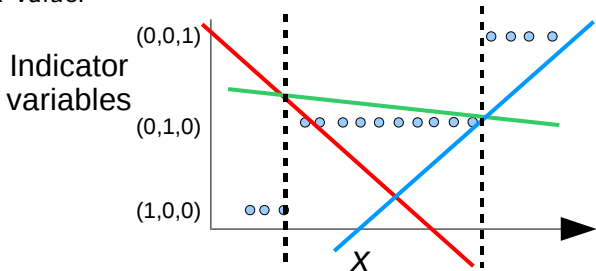


- Overlay them to see which one is the maximum for each x value.

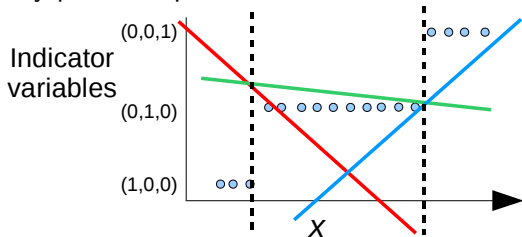
- Can plot the three output components on three separate graphs. What linear functions will each one learn?



- Overlay them to see which one is the maximum for each x value.



- See any potential problems?



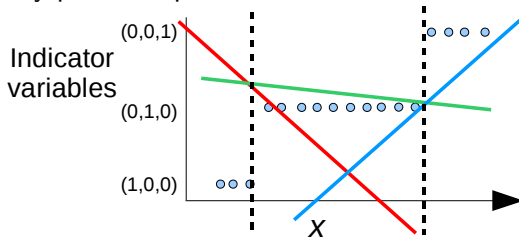
Linear Least Squares for Classification

Indicator Variables
Masking Problem
Example

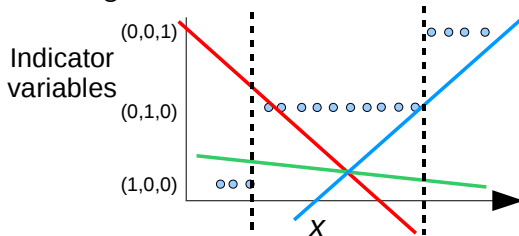
Generative Models for Classification

QDA
Fitting the Generative Distributions to Data
Overfitting
LDA
Example

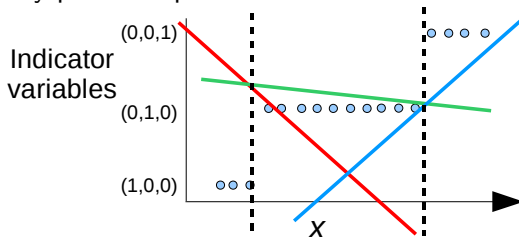
- See any potential problems?



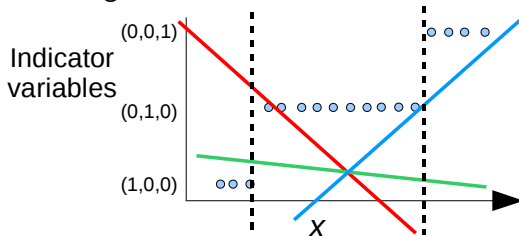
- What if the green line is too low?



- See any potential problems?

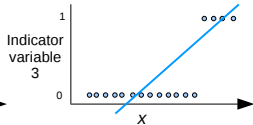
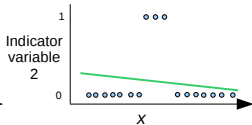
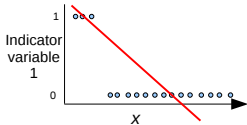


- What if the green line is too low?



- What could cause this?

• Too few samples from Class 2.



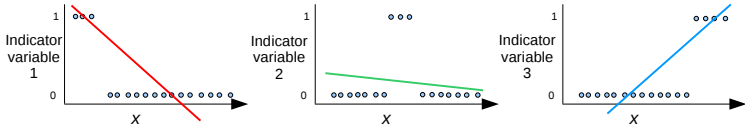
Linear Least Squares for Classification

Indicator Variables
Masking Problem
Example

Generative Models for Classification

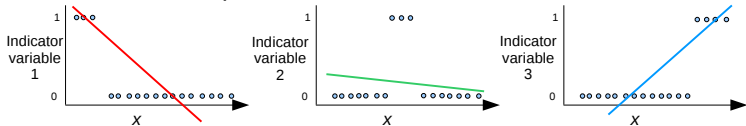
QDA
Fitting the Generative Distributions to Data
Overfitting
LDA
Example

- Too few samples from Class 2.



- There may be no values of x for which the second output, y_2 , of our linear model is larger than the other two. Class 2 has become **masked** by the other classes.

- Too few samples from Class 2.



- There may be no values of x for which the second output, y_2 , of our linear model is larger than the other two. Class 2 has become **masked** by the other classes.
- What other shape of function response would work better for this data? Hold that thought, while we try an example.

Application: Parkinsons Data Set from UCI ML Archive

- 147 samples from subjects with Parkinsons, 48 samples from healthy subjects

Application: Parkinsons Data Set from UCI ML Archive

- 147 samples from subjects with Parkinsons, 48 samples from healthy subjects
- Each sample composed of 21 numerical features extracted from voice recordings

Application: Parkinsons Data Set from UCI ML Archive

- 147 samples from subjects with Parkinsons, 48 samples from healthy subjects
- Each sample composed of 21 numerical features extracted from voice recordings
- from collaboration with the University of Oxford and the National Center for Voice and Speech in Denver.

Read and Prepare the Data

```
data <- read.table(" parkinsons.data",header=TRUE,sep=";")
### remove name and make numeric
data <- as.matrix(data[,-1])
### randomly rearrange data
data <- data[sample(nrow(data)),]
### status is 0 for healthy, 1 for Parkinsons
status <- data[,"status"]
### remove status column from data
data <- data[, -which(colnames(data)== "status" ) ]

dataHealthy <- data[status==0,]
dataParks <- data[status==1,]
nHealthy <- nrow(dataHealthy)
nParks <- nrow(dataParks)
```

Read and Prepare the Data

```

data <- read.table(" parkinsons.data",header=TRUE,sep=";")
### remove name and make numeric
data <- as.matrix(data[,-1])
### randomly rearrange data
data <- data[sample(nrow(data)),]
### status is 0 for healthy, 1 for Parkinsons
status <- data[,"status"]
### remove status column from data
data <- data[, -which(colnames(data)== "status" ) ]

dataHealthy <- data[status==0,]
dataParks <- data[status==1,]
nHealthy <- nrow(dataHealthy)
nParks <- nrow(dataParks)

```

• Look at the data

```

> data[1:2,]
  MDVP.Fo.Hz. MDVP.Fhi.Hz. MDVP.Flo.Hz. MDVP.Jitter... MDVP.Jitter.Abs.
[1,] 223.361 263.872 87.638 0.00352 2e-05
[2,] 136.926 159.866 131.276 0.00293 2e-05
  MDVP.RAP MDVP.PPQ Jitter.DDP MDVP.Shimmer MDVP.Shimmer.dB. Shimmer.APQ3
[1,] 0.00169 0.00188 0.00506 0.02536 0.225 0.01379
[2,] 0.00118 0.00153 0.00355 0.01259 0.112 0.00656
  Shimmer.APQ5 MDVP.APQ Shimmer.DDA NHR HNR RPDE DFA
[1,] 0.01478 0.01909 0.04137 0.01493 20.366 0.566849 0.574282
[2,] 0.00717 0.01140 0.01968 0.00581 25.703 0.460600 0.646846
  spread1 spread2 D2 PPE
[1,] -5.456811 0.345238 2.840556 0.232861
[2,] -6.547148 0.152813 2.041277 0.138512

```

Make Train and Test Partitions

- For small sample size or very uneven number of samples from each class, force equal sampling proportions of two classes when building train, test partitions.

Make Train and Test Partitions

- For small sample size or very uneven number of samples from each class, force equal sampling proportions of two classes when building train, test partitions.
- Two indicator variables is equivalent to using single variable. We will use values 1 and 2 to indicate Class 1 (healthy) and Class 2 (Parkinsons).

Make Train and Test Partitions

- For small sample size or very uneven number of samples from each class, force equal sampling proportions of two classes when building train, test partitions.
- Two indicator variables is equivalent to using single variable. We will use values 1 and 2 to indicate Class 1 (healthy) and Class 2 (Parkinsons).

```
trainf <- 0.8
Xtrain <- rbind(dataHealthy[1:floor(trainf*nHealthy)],
               dataParks[1: floor ( trainf *nParks),])
Ttrain <- matrix(c(rep(1,floor(trainf*nHealthy)),
                  rep(2, floor ( trainf *nParks))))
Xtest <- rbind(dataHealthy[-(1:floor(trainf*nHealthy))],
               dataParks[-(1:floor ( trainf *nParks),)])
Ttest <- matrix(c(rep(1,nHealthy-floor(trainf*nHealthy)),
                  rep(2,nParks-floor ( trainf *nParks))))
```

Calculate Linear Least Squares Solution

- Standardize

```
standardize <- makeStandardizeF(Xtrain)  
Xtrain1 <- cbind(1,standardize(Xtrain))
```

Calculate Linear Least Squares Solution

- Standardize

```
standardize <- makeStandardizeF(Xtrain)  
Xtrain1 <- cbind(1,standardize(Xtrain))
```

- Calculate w

```
w <- solve(t(Xtrain1)%*%Xtrain1,t(Xtrain1)%*%Ttrain)
```

Calculate Linear Least Squares Solution

- Standardize

```
standardize <- makeStandardizeF(Xtrain)  
Xtrain1 <- cbind(1,standardize(Xtrain))
```

- Calculate w

```
w <- solve(t(Xtrain1)%*%Xtrain1,t(Xtrain1)%*%Ttrain)
```

- Test it.

```
TtrainPredicted <- Xtrain1 %*% w  
TtestPredicted <- cbind(1,standardize(Xtest)) %*% w  
pCorrectTrain <- sum(apply(abs(cbind(TtrainPredicted-1, TtrainPredicted-2)),  
1, which.min) == Ttrain) / length(Ttrain) * 100.0  
pCorrectTest <- sum(apply(abs(cbind(TtestPredicted-1, TtestPredicted-2)),  
1, which.min) == Ttest) / length(Ttest) * 100.0  
cat(" Training data",pCorrectTrain," percent correct \n")  
cat(" Testing data",pCorrectTest," percent correct \n")
```

results in

```
Training data 93.54839 percent correct  
Testing data 80 percent correct
```

Linear Least
Squares for
ClassificationIndicator Variables
Masking Problem
ExampleGenerative Models
for ClassificationQDA
Fitting the Generative
Distributions to Data
Overfitting
LDA
Example

Calculate Linear Least Squares Solution

- Standardize

```
standardize <- makeStandardizeF(Xtrain)
Xtrain1 <- cbind(1,standardize(Xtrain))
```

- Calculate w

```
w <- solve(t(Xtrain1)%*%Xtrain1,t(Xtrain1)%*%Ttrain)
```

- Test it.

```
TtrainPredicted <- Xtrain1 %*% w
TtestPredicted <- cbind(1,standardize(Xtest)) %*% w
pCorrectTrain <- sum(apply(abs(cbind(TtrainPredicted-1, TtrainPredicted-2)),
  1, which.min) == Ttrain) / length(Ttrain) * 100.0
pCorrectTest <- sum(apply(abs(cbind(TtestPredicted-1, TtestPredicted-2)),
  1, which.min) == Ttest) / length(Ttest) * 100.0
cat(" Training data",pCorrectTrain," percent correct\n")
cat(" Testing data",pCorrectTest," percent correct\n")
```

results in

```
Training data 93.54839 percent correct
Testing data 80 percent correct
```

- Repeating it all, gives

```
Training data 90.32258 percent correct
Testing data 95 percent correct
```

Plot it

- What visualization would you use to check the results?

Linear Least
Squares for
Classification

Indicator Variables
Masking Problem
Example

Generative Models
for Classification

QDA
Fitting the Generative
Distributions to Data
Overfitting
LDA
Example

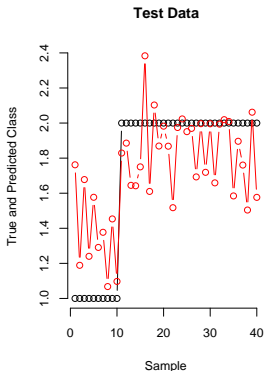
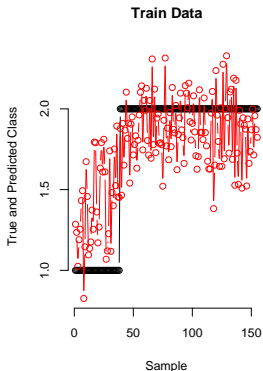
Plot it

- What visualization would you use to check the results?
- Let's plot the true class with the output of the model for each training sample, then each testing sample.

Plot it

- What visualization would you use to check the results?
- Let's plot the true class with the output of the model for each training sample, then each testing sample.

```
p <- par(mfrow=c(1,2),bty="n")
matplot(cbind(Ttrain, TtrainPredicted), type="b", pch=1, lty=1,
        xlab="Sample", ylab="True and Predicted Class", main="Train Data")
matplot(cbind(Ttest, TtestPredicted), type="b", pch=1, lty=1,
        xlab="Sample", ylab="True and Predicted Class", main="Test Data")
par(p)
```



Linear Least Squares for Classification

Indicator Variables

Masking Problem

Example

Linear Least
Squares for
Classification

Indicator Variables

Masking Problem

Example

Generative Models for Classification

QDA

Fitting the Generative Distributions to Data

Overfitting

LDA

Example

Generative Models
for Classification

QDA

Fitting the Generative
Distributions to Data

Overfitting

LDA

Example

Back to that Masking Problem

- What function shape were you thinking of that might fix the masking problem?

Back to that Masking Problem

- What function shape were you thinking of that might fix the masking problem?
- Radial basis function? Good choice! But, remember what a radial basis function resembles?

Back to that Masking Problem

- What function shape were you thinking of that might fix the masking problem?
- Radial basis function? Good choice! But, remember what a radial basis function resembles?
- Right again! A Gaussian distribution.

Back to that Masking Problem

- What function shape were you thinking of that might fix the masking problem?
- Radial basis function? Good choice! But, remember what a radial basis function resembles?
- Right again! A Gaussian distribution.
- This leads us into our discussion of generative models for classification. A very common generative model is the use of one Gaussian distribution to model data from each class. This is “generative”, because we are assuming that the Gaussian distribution is the underlying process that generates the data samples.

Using Generative Model to Classify

- So, let's say we come up with the generative distribution, such as a Gaussian distribution, for Class k , called $p(\mathbf{x}|\text{Class} = k)$, or $p(\mathbf{x}|C = k)$. How do we use it to classify?

Using Generative Model to Classify

- So, let's say we come up with the generative distribution, such as a Gaussian distribution, for Class k , called $p(\mathbf{x}|\text{Class} = k)$, or $p(\mathbf{x}|C = k)$. How do we use it to classify?
- Can just take the distribution with the highest value, $\text{argmax}_k p(\mathbf{x}|C = k)$. But we can do better than this...think Bayes' Rule.

Using Generative Model to Classify

- So, let's say we come up with the generative distribution, such as a Gaussian distribution, for Class k , called $p(\mathbf{x}|\text{Class} = k)$, or $p(\mathbf{x}|C = k)$. How do we use it to classify?
- Can just take the distribution with the highest value, $\text{argmax}_k p(\mathbf{x}|C = k)$. But we can do better than this...think Bayes' Rule.
- Ultimately we would like to know $p(C = k|\mathbf{x})$. How do we get this from $p(\mathbf{x}|C = k)$?

Using Generative Model to Classify

- So, let's say we come up with the generative distribution, such as a Gaussian distribution, for Class k , called $p(\mathbf{x}|Class = k)$, or $p(\mathbf{x}|C = k)$. How do we use it to classify?
- Can just take the distribution with the highest value, $\operatorname{argmax}_k p(\mathbf{x}|C = k)$. But we can do better than this...think Bayes' Rule.
- Ultimately we would like to know $p(C = k|\mathbf{x})$. How do we get this from $p(\mathbf{x}|C = k)$?
- Remember that

$$p(C = k, \mathbf{x}) = p(C = k|\mathbf{x})p(\mathbf{x}) = p(\mathbf{x}|C = k)p(C = k)$$

Bayes' Rule for Classification

$$p(C = k|\mathbf{x}) = \frac{p(\mathbf{x}|C = k)p(C = k)}{p(\mathbf{x})}$$

Bayes' Rule for Classification

$$p(C = k|\mathbf{x}) = \frac{p(\mathbf{x}|C = k)p(C = k)}{p(\mathbf{x})}$$

Bayes' Rule for Classification

$$p(C = k|\mathbf{x}) = \frac{p(\mathbf{x}|C = k)p(C = k)}{p(\mathbf{x})}$$

$$= \frac{p(\mathbf{x}|C = k)p(C = k)}{\sum_{k=1}^K p(\mathbf{x}, C = k)}$$

Bayes' Rule for Classification

$$p(C = k|\mathbf{x}) = \frac{p(\mathbf{x}|C = k)p(C = k)}{p(\mathbf{x})}$$

$$= \frac{p(\mathbf{x}|C = k)p(C = k)}{\sum_{k=1}^K p(\mathbf{x}, C = k)}$$

$$= \frac{p(\mathbf{x}|C = k)p(C = k)}{\sum_{k=1}^K p(\mathbf{x}|C = k)p(C = k)}$$

Simplify to just Two Classes

- For two classes, $k \in \{1, 2\}$. We will classify a sample \mathbf{x} as Class 2 if $p(C = 2|\mathbf{x}) > p(C = 1|\mathbf{x})$. Now expand and simplify...

Simplify to just Two Classes

- For two classes, $k \in \{1, 2\}$. We will classify a sample \mathbf{x} as Class 2 if $p(C = 2|\mathbf{x}) > p(C = 1|\mathbf{x})$. Now expand and simplify...

$$p(C = 2|\mathbf{x}) > p(C = 1|\mathbf{x})$$

$$\frac{p(\mathbf{x}|C = 2)p(C = 2)}{p(\mathbf{x})} > \frac{p(\mathbf{x}|C = 1)p(C = 1)}{p(\mathbf{x})}$$

$$p(\mathbf{x}|C = 2)p(C = 2) > p(\mathbf{x}|C = 1)p(C = 1)$$

Use our Gaussian Assumption

- Using our assumption that the generative distribution for each class is a Gaussian distribution,

$$p(\mathbf{x}|C = 2)p(C = 2) > p(\mathbf{x}|C = 1)p(C = 1)$$

$$\begin{aligned} \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_2|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2)} p(C = 2) \\ > \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_1|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1)} p(C = 1) \end{aligned}$$

$$\begin{aligned} |\Sigma_2|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2)} p(C = 2) \\ > |\Sigma_1|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1)} p(C = 1) \end{aligned}$$

Use our Gaussian Assumption

- Using our assumption that the generative distribution for each class is a Gaussian distribution,

$$p(\mathbf{x}|C = 2)p(C = 2) > p(\mathbf{x}|C = 1)p(C = 1)$$

$$\begin{aligned} \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_2|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2)} p(C = 2) \\ > \frac{1}{(2\pi)^{\frac{d}{2}}|\Sigma_1|^{\frac{1}{2}}} e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1)} p(C = 1) \end{aligned}$$

$$\begin{aligned} |\Sigma_2|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_2)^T\Sigma_2^{-1}(\mathbf{x}-\mu_2)} p(C = 2) \\ > |\Sigma_1|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\mu_1)^T\Sigma_1^{-1}(\mathbf{x}-\mu_1)} p(C = 1) \end{aligned}$$

- Hey, there are multiplications and exponentials here. Let's use logarithms.

$$\begin{aligned} & |\Sigma_2|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T \Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)} p(C=2) \\ & > |\Sigma_1|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)} p(C=1) \end{aligned}$$

$$\begin{aligned} & -\frac{1}{2} \ln |\Sigma_2| + -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T \Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \ln p(C=2) \\ & > -\frac{1}{2} \ln |\Sigma_1| + -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \ln p(C=1) \end{aligned}$$

- If we define each side of this last inequality as a discriminant function, $\delta_k(\mathbf{x})$ for Class k , then

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \ln P(C=k)$$

and the class of a new sample \mathbf{x} is $\operatorname{argmax}_k \delta_k(\mathbf{x})$.

$$\begin{aligned}
 & |\Sigma_2|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T \Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)} p(C=2) \\
 & > |\Sigma_1|^{-\frac{1}{2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)} p(C=1)
 \end{aligned}$$

$$\begin{aligned}
 & -\frac{1}{2} \ln |\Sigma_2| + -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)^T \Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2) + \ln p(C=2) \\
 & > -\frac{1}{2} \ln |\Sigma_1| + -\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)^T \Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1) + \ln p(C=1)
 \end{aligned}$$

- If we define each side of this last inequality as a discriminant function, $\delta_k(\mathbf{x})$ for Class k , then

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) + \ln P(C=k)$$

and the class of a new sample \mathbf{x} is $\operatorname{argmax}_k \delta_k(\mathbf{x})$.

- The boundary between Class 1 and Class 2 is the set of points \mathbf{x} for which $\delta_2(\mathbf{x}) = \delta_1(\mathbf{x})$. This equation is quadratic in \mathbf{x} , meaning that the boundary between Class 1 and 2 is quadratic. We have just defined Quadratic Discriminant Analysis, or QDA.

Fitting the Generative Distributions to Data

- Let's use the maximum likelihood solution for a Gaussian distribution.

Fitting the Generative Distributions to Data

- Let's use the maximum likelihood solution for a Gaussian distribution.
- Let X_k be the set of training samples from Class k , and N_k be the number of training samples from Class k .

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{\mathbf{x} \in X_k} \mathbf{x}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k - 1} \sum_{\mathbf{x} \in X_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T$$

Linear Least
Squares for
ClassificationIndicator Variables
Masking Problem
ExampleGenerative Models
for ClassificationQDA
Fitting the Generative
Distributions to DataOverfitting
LDA
Example

Fitting the Generative Distributions to Data

- Let's use the maximum likelihood solution for a Gaussian distribution.
- Let X_k be the set of training samples from Class k , and N_k be the number of training samples from Class k .

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{\mathbf{x} \in X_k} \mathbf{x}$$

$$\boldsymbol{\Sigma}_k = \frac{1}{N_k - 1} \sum_{\mathbf{x} \in X_k} (\mathbf{x} - \boldsymbol{\mu}_k)(\mathbf{x} - \boldsymbol{\mu}_k)^T$$

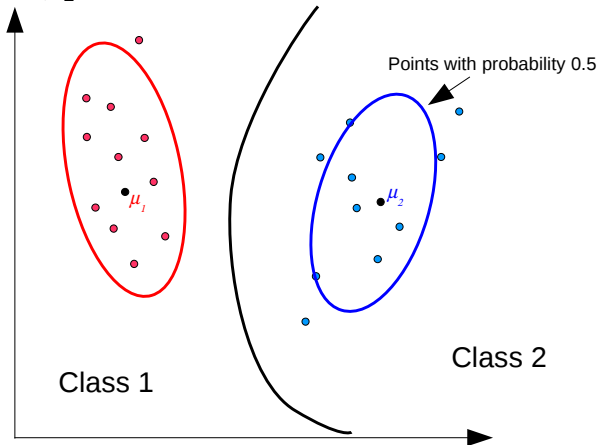
- What about $p(C = k)$, which is the a priori probability distribution of Class k ? If we have no prior belief that one class is more likely than any other,

$$p(C = k) = \frac{N_k}{N}$$

where N is the total number of samples from all classes.

Two Classes for Two-Dimensional Samples

- Given samples from two classes, calculate μ_1 and Σ_1 and μ_2 and Σ_2 .



Can QDA Overfit

- Assuming single Gaussian as model of data from each class does not seem to lead to an exceedingly complex model. But, how many parameters are there in the mean and covariance matrix, if data is d -dimensional?

Can QDA Overfit

- Assuming single Gaussian as model of data from each class does not seem to lead to an exceedingly complex model. But, how many parameters are there in the mean and covariance matrix, if data is d -dimensional?
- Mean has d components.

Can QDA Overfit

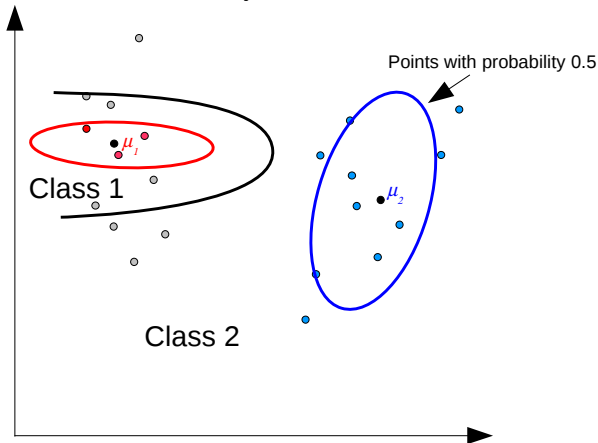
- Assuming single Gaussian as model of data from each class does not seem to lead to an exceedingly complex model. But, how many parameters are there in the mean and covariance matrix, if data is d -dimensional?
- Mean has d components.
- Covariance matrix has d^2 components. If $d = 100$, the covariance matrix has 100,000 parameters. Whoa!

- Assuming single Gaussian as model of data from each class does not seem to lead to an exceedingly complex model. But, how many parameters are there in the mean and covariance matrix, if data is d -dimensional?
- Mean has d components.
- Covariance matrix has d^2 components. If $d = 100$, the covariance matrix has 100,000 parameters. Whoa!
- Actually the covariance matrix is symmetric, so it only has $\frac{d^2}{2} + \frac{d}{2} = \frac{d(d+1)}{2}$ unique values. Still a lot. And we have one for each class, so total number of parameters, including mean, is $K(d + \frac{d(d+1)}{2})$.

- Assuming single Gaussian as model of data from each class does not seem to lead to an exceedingly complex model. But, how many parameters are there in the mean and covariance matrix, if data is d -dimensional?
- Mean has d components.
- Covariance matrix has d^2 components. If $d = 100$, the covariance matrix has 100,000 parameters. Whoa!
- Actually the covariance matrix is symmetric, so it only has $\frac{d^2}{2} + \frac{d}{2} = \frac{d(d+1)}{2}$ unique values. Still a lot. And we have one for each class, so total number of parameters, including mean, is $K(d + \frac{d(d+1)}{2})$.
- What if the data distribution is under-sampled?

Under-sampled

- Gaussian for Class 1 is far from correct. Class boundary will now lead to many errors.



How to Reduce Overfitting?

- Need to remove flexibility from the Gaussian model.
How?

How to Reduce Overfitting?

- Need to remove flexibility from the Gaussian model. How?
- Could restrict all covariance matrices to be diagonal. The ellipses would be parallel to the axes. Wouldn't work well if features are correlated.

How to Reduce Overfitting?

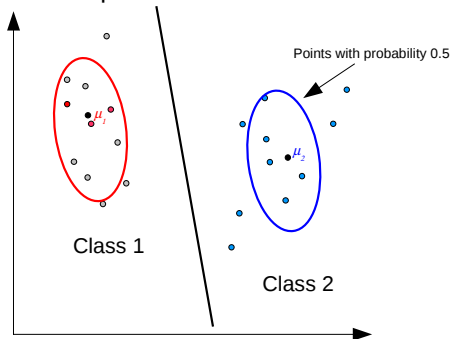
- Need to remove flexibility from the Gaussian model. How?
- Could restrict all covariance matrices to be diagonal. The ellipses would be parallel to the axes. Wouldn't work well if features are correlated.
- Could force all classes to have the same covariance matrix by averaging the covariance matrices from every class.

How to Reduce Overfitting?

- Need to remove flexibility from the Gaussian model. How?
- Could restrict all covariance matrices to be diagonal. The ellipses would be parallel to the axes. Wouldn't work well if features are correlated.
- Could force all classes to have the same covariance matrix by averaging the covariance matrices from every class.
- Seems like a bad idea, but at least we are using all of the data samples to come up with a covariance matrix.

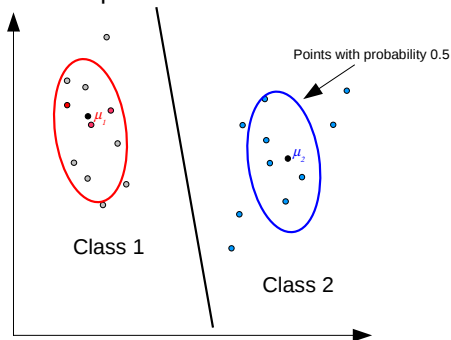
Averaging Covariance Matrices from Every Class

- Covariance matrix for each class is now the average of covariance matrix for each class, weighted by the fraction of samples from that class.



Averaging Covariance Matrices from Every Class

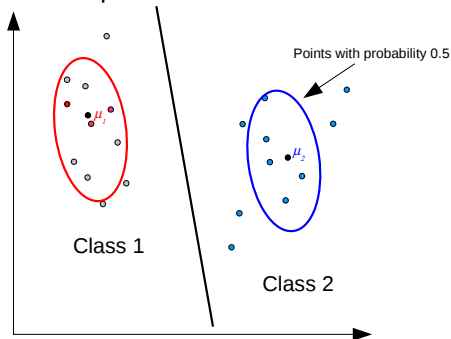
- Covariance matrix for each class is now the average of covariance matrix for each class, weighted by the fraction of samples from that class.



- Better result than using unique covariance matrices.

Averaging Covariance Matrices from Every Class

- Covariance matrix for each class is now the average of covariance matrix for each class, weighted by the fraction of samples from that class.



- Better result than using unique covariance matrices.
- Notice the boundary. It is now linear, not the quadratic curve we had before. Why?

- Remember our discriminant function.

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln P(C = k)$$

LDA

- Remember our discriminant function.

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln P(C = k)$$

- When we compare discriminant functions, $\delta_2(\mathbf{x}) > \delta_1(\mathbf{x})$, and use the same covariance matrix Σ for every class, we get

$$\begin{aligned} & -\frac{1}{2} \ln |\Sigma| + -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) + \ln p(C = 2) \\ & > -\frac{1}{2} \ln |\Sigma| + -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \ln p(C = 1) \end{aligned}$$

Linear Least Squares for Classification

Indicator Variables
Masking Problem
Example

Generative Models for Classification

QDA
Fitting the Generative Distributions to Data
Overfitting

LDA

Example

LDA

- Remember our discriminant function.

$$\delta_k(\mathbf{x}) = -\frac{1}{2} \ln |\Sigma_k| - \frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_k)^T \Sigma_k^{-1} (\mathbf{x} - \boldsymbol{\mu}_k) + \ln P(C = k)$$

- When we compare discriminant functions, $\delta_2(\mathbf{x}) > \delta_1(\mathbf{x})$, and use the same covariance matrix Σ for every class, we get

$$\begin{aligned} & -\frac{1}{2} \ln |\Sigma| + -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) + \ln p(C = 2) \\ & > -\frac{1}{2} \ln |\Sigma| + -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \ln p(C = 1) \end{aligned}$$

- which can be simplified to

$$\begin{aligned} & -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_2)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_2) + \ln p(C = 2) \\ & > -\frac{1}{2} (\mathbf{x} - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x} - \boldsymbol{\mu}_1) + \ln p(C = 1) \end{aligned}$$

and

$$\begin{aligned} & \mathbf{x}^T \Sigma^{-1} \boldsymbol{\mu}_1 - \frac{1}{2} \boldsymbol{\mu}_1^T \Sigma^{-1} \boldsymbol{\mu}_1 + \log P(C = 1) \\ & > \mathbf{x}^T \Sigma^{-1} \boldsymbol{\mu}_2 - \frac{1}{2} \boldsymbol{\mu}_2^T \Sigma^{-1} \boldsymbol{\mu}_2 + \log P(C = 2) \end{aligned}$$

- From previous slide

$$\mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \log P(C = 1)$$

$$> \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 - \frac{1}{2} \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 + \log P(C = 2)$$

Linear Least
Squares for
ClassificationIndicator Variables
Masking Problem
ExampleGenerative Models
for ClassificationQDA
Fitting the Generative
Distributions to Data
Overfitting**LDA**

Example

- From previous slide

$$\begin{aligned} & \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \log P(C = 1) \\ & > \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 - \frac{1}{2} \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 + \log P(C = 2) \end{aligned}$$

- So, our discriminant function has become

$$\delta_k(\mathbf{w}) = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log P(C = k)$$

- From previous slide

$$\begin{aligned} & \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \log P(C = 1) \\ & > \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 - \frac{1}{2} \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 + \log P(C = 2) \end{aligned}$$

- So, our discriminant function has become

$$\delta_k(\mathbf{w}) = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log P(C = k)$$

- This is linear in \mathbf{x} , hence and can be written as

$$\delta_k(\mathbf{w}) = \mathbf{x}^T \mathbf{w}_k + \text{constant}_k$$

- From previous slide

$$\begin{aligned} & \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 - \frac{1}{2} \boldsymbol{\mu}_1^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_1 + \log P(C = 1) \\ & > \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 - \frac{1}{2} \boldsymbol{\mu}_2^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_2 + \log P(C = 2) \end{aligned}$$

- So, our discriminant function has become

$$\delta_k(\mathbf{w}) = \mathbf{x}^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k - \frac{1}{2} \boldsymbol{\mu}_k^T \boldsymbol{\Sigma}^{-1} \boldsymbol{\mu}_k + \log P(C = k)$$

- This is linear in \mathbf{x} , hence and can be written as

$$\delta_k(\mathbf{w}) = \mathbf{x}^T \mathbf{w}_k + \text{constant}_k$$

- or, if we add a constant 1 to \mathbf{x} and include constant_k in \mathbf{w}

$$\delta_k(\mathbf{w}) = \mathbf{x}^T \mathbf{w}_k$$

- So, using Gaussian distributions as generative models and restricting the covariance matrices to all be the weighted average of class covariance matrices, results in a linear boundary. This approach is called Linear Discriminant Analysis (LDA).

- So, using Gaussian distributions as generative models and restricting the covariance matrices to all be the weighted average of class covariance matrices, results in a linear boundary. This approach is called Linear Discriminant Analysis (LDA).
- Both QDA and LDA are based on Gaussian distributions for modeling the data samples in each class.

Linear Least
Squares for
ClassificationIndicator Variables
Masking Problem
ExampleGenerative Models
for ClassificationQDA
Fitting the Generative
Distributions to Data
OverfittingLDA
Example

- So, using Gaussian distributions as generative models and restricting the covariance matrices to all be the weighted average of class covariance matrices, results in a linear boundary. This approach is called Linear Discriminant Analysis (LDA).
- Both QDA and LDA are based on Gaussian distributions for modeling the data samples in each class.
- QDA is more flexible, but LDA often works better in practice. When?

- So, using Gaussian distributions as generative models and restricting the covariance matrices to all be the weighted average of class covariance matrices, results in a linear boundary. This approach is called Linear Discriminant Analysis (LDA).
- Both QDA and LDA are based on Gaussian distributions for modeling the data samples in each class.
- QDA is more flexible, but LDA often works better in practice. When?
- Undersampled data

- So, using Gaussian distributions as generative models and restricting the covariance matrices to all be the weighted average of class covariance matrices, results in a linear boundary. This approach is called Linear Discriminant Analysis (LDA).
- Both QDA and LDA are based on Gaussian distributions for modeling the data samples in each class.
- QDA is more flexible, but LDA often works better in practice. When?
- Undersampled data
 - Small number of samples

- So, using Gaussian distributions as generative models and restricting the covariance matrices to all be the weighted average of class covariance matrices, results in a linear boundary. This approach is called Linear Discriminant Analysis (LDA).
- Both QDA and LDA are based on Gaussian distributions for modeling the data samples in each class.
- QDA is more flexible, but LDA often works better in practice. When?
- Undersampled data
 - Small number of samples
 - High dimensional data

- Calculate means and covariance matrices

```
Xtrains <- standardize(Xtrain)
Xtests <- standardize(Xtest)
mean1 <- colMeans(Xtrains[Ttrain==1,])
cov1 <- cov(Xtrains[Ttrain==1,])
mean2 <- colMeans(Xtrains[Ttrain==2,])
cov2 <- cov(Xtrains[Ttrain==2,])
```

- Calculate means and covariance matrices

```
Xtrains <- standardize(Xtrain)
Xtests <- standardize(Xtest)
mean1 <- colMeans(Xtrains[Ttrain==1,])
cov1 <- cov(Xtrains[Ttrain==1,])
mean2 <- colMeans(Xtrains[Ttrain==2,])
cov2 <- cov(Xtrains[Ttrain==2,])
```

- Form the QDA discriminant functions.

```
makeQDADiscF <- function(mean,sigma,prior) {
  sigmalnv <- solve(sigma)
  function(X) {
    diff <- X - matrix(mean,nrow(X),ncol(X),byrow=TRUE)
    -0.5 * log(det(sigma)) - 0.5 * rowSums(diff %**% sigmalnv * diff) + log(prior)
  }
}
qdadisc1 <- makeQDADiscF(mean1,cov1,nHealthy/(nHealthy+nParks))
qdadisc2 <- makeQDADiscF(mean2,cov2,nParks/(nHealthy+nParks))
```

- Calculate means and covariance matrices

```
Xtrains <- standardize(Xtrain)
Xtests <- standardize(Xtest)
mean1 <- colMeans(Xtrains[Ttrain==1,])
cov1 <- cov(Xtrains[Ttrain==1,])
mean2 <- colMeans(Xtrains[Ttrain==2,])
cov2 <- cov(Xtrains[Ttrain==2,])
```

- Form the QDA discriminant functions.

```
makeQDADiscF <- function(mean,sigma,prior) {
  sigmalnv <- solve(sigma)
  function(X) {
    diff <- X - matrix(mean,nrow(X),ncol(X),byrow=TRUE)
    -0.5 * log(det(sigma)) - 0.5 * rowSums(diff %*% sigmalnv * diff) + log(prior)
  }
}
qdadisc1 <- makeQDADiscF(mean1,cov1,nHealthy/(nHealthy+nParks))
qdadisc2 <- makeQDADiscF(mean2,cov2,nParks/(nHealthy+nParks))
```

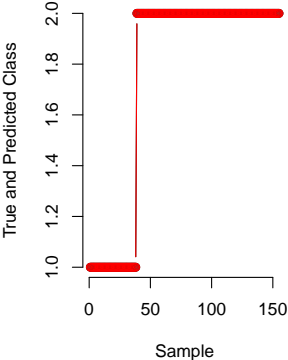
- Apply them to the train and test data.

```
TtrainPredicted <- apply(cbind(qdadisc1(Xtrains),qdadisc2(Xtrains)),1, which.max)
TtestPredicted <- apply(cbind(qdadisc1(Xtests),qdadisc2(Xtests)),1, which.max)
```

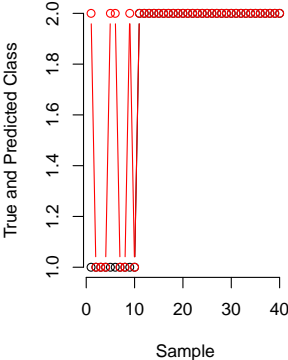
Results

Training data 100 percent correct
Testing data 90 percent correct

Train Data



Test Data



Linear Least Squares for Classification

Indicator Variables
Masking Problem
Example

Generative Models for Classification

QDA
Fitting the Generative Distributions to Data
Overfitting
LDA
Example

How about LDA?

- Same as QDA, but form one covariance matrix. Can reuse `makeQDADiscF`.

```
N <- nHealthy + nParks
covavg <- cov1 * nHealthy/N + cov2 * nParks/N
ldadisc1 <- makeQDADiscF(mean1,covavg,nHealthy/(nHealthy+nParks))
ldadisc2 <- makeQDADiscF(mean2,covavg,nParks/(nHealthy+nParks))
```

How about LDA?

- Same as QDA, but form one covariance matrix. Can reuse `makeQDADiscF`.

```
N <- nHealthy + nParks
covavg <- cov1 * nHealthy/N + cov2 * nParks/N
ldadisc1 <- makeQDADiscF(mean1,covavg,nHealthy/(nHealthy+nParks))
ldadisc2 <- makeQDADiscF(mean2,covavg,nParks/(nHealthy+nParks))
```

- Results are

```
Training data 100 percent correct
Testing data 82.5 percent correct
```

