Some announcements

- Reminder: Tutorial this Friday, 6-7 pm, Stewart Biology S3/3

- First mini-project released yesterday!

- First mini-project due Jan. 26\textsuperscript{th}
  But since the wrong date was announced in the slides, we’ll accept submissions until Monday the 29\textsuperscript{th} at \textbf{noon}
  We do recommend aiming for the original date since the second mini-project will start Jan. 26\textsuperscript{th}!

- To be submitted \textbf{individually}
  - You can discuss solutions, but write code and the report \textbf{by yourself}
  - Give credit to discussion partners, as per instructions!
Quiz

– To test yourself, there will be a quiz on MyCourses for most lectures
– Not graded, strictly optional
– One quiz on overfitting is now online
– Auto-gradeable items will give you feedback on how well you are doing
Today’s Quiz

1. What is meant by the term *overfitting*? What can cause overfitting? How can one avoid overfitting?

2. Which of the following increases the chances of overfitting (assuming everything else is held constant):
   a) Reducing the size of the training set.
   b) Increasing the size of the training set.
   c) Reducing the size of the test set.
   d) Increasing the size of the test set.
   e) Reducing the number of features.
   f) Increasing the number of features.
A quick look at evaluation functions

• We call \( L(Y, f_w(x)) \) the loss function.
  
  – Least-square / Mean squared-error (MSE) loss:

  \[
  L(Y, f_w(X)) = \sum_{i=1:n} (y_i - w^T x_i)^2
  \]

• Other loss functions?
A quick look at evaluation functions

- We call $L(Y, f_w(x))$ the loss function.
  - Least-square / Mean squared-error (MSE) loss:
    \[
    L(Y, f_w(X)) = \sum_{i=1:n} (y_i - w^T x_i)^2
    \]

- Other loss functions?
  - Absolute error loss:
    \[
    L(Y, f_w(X)) = \sum_{i=1:n} |y_i - w^T x_i|
    \]
  - 0-1 loss (for classification):
    \[
    L(Y, f_w(X)) = \sum_{i=1:n} 1(y_i \neq f_w(x_i))
    \]
A quick look at evaluation functions

• We call $L(Y, f_w(x))$ the loss function.
  
  – Least-square / Mean squared-error (MSE) loss:
    $$ L(Y, f_w(X)) = \sum_{i=1:n} (y_i - w^T x_i)^2 $$
  
  • Other loss functions?

  – Absolute error loss:
    $$ L(Y, f_w(X)) = \sum_{i=1:n} |y_i - w^T x_i| $$

  – 0-1 loss (for classification):
    $$ L(Y, f_w(X)) = \sum_{i=1:n} I(y_i \neq f_w(x_i)) $$

• Different loss functions make **different assumptions**.
  
  – Squared error loss assumes the data can be approximated by a global linear model with Gaussian noise.

  – Loss function **independent** of complexity penalty (L1 or L2)
A quick look at evaluation functions

- Assume data generated as

\[ p(y|\mathbf{x}; \mathbf{w}) = \mathcal{N}(\mathbf{x}^T \mathbf{w}, \sigma^2) \]

\[ = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left( -\frac{(y - \mathbf{x}^T \mathbf{w})^2}{2\sigma^2} \right) \]
A quick look at evaluation functions

• Assume data generated as

\[
p(y|x; w) = \mathcal{N}(x^T w, \sigma^2)
\]

\[
= \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(y - x^T w)^2}{2\sigma^2}\right)
\]

• Reasonable to try to maximize the probability of generating the labels \(y\)

\[
\arg\max_w p(y|x, w) = \arg\max_w \log p(y|x, w)
\]

\[
= \arg\max_w \text{const} - (y - x^T w)^2
\]
Next: Linear models for classification

**FIGURE 2.1.** A classification example in two dimensions. The classes are coded as a binary variable (BLUE = 0, ORANGE = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T \hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as ORANGE, while the blue region is classified as BLUE.
Classification problems

Given data set \( D = \langle x_i, y_i \rangle, i = 1:n \), with discrete \( y_i \), find a hypothesis which “best fits” the data.

- If \( y_i \in \{0, 1\} \) this is **binary** classification.
- If \( y_i \) can take more than two values, the problem is called **multi-class** classification.

### Fisher's Iris Data

<table>
<thead>
<tr>
<th>Sepal length</th>
<th>Sepal width</th>
<th>Petal length</th>
<th>Petal width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.7</td>
<td>3.2</td>
<td>1.3</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.1</td>
<td>1.5</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.6</td>
<td>1.4</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.7</td>
<td>0.4</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.4</td>
<td>1.4</td>
<td>0.3</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.0</td>
<td>3.4</td>
<td>1.5</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.4</td>
<td>2.9</td>
<td>1.4</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.9</td>
<td>3.1</td>
<td>1.5</td>
<td>0.1</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.7</td>
<td>1.5</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.4</td>
<td>1.6</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.8</td>
<td>3.0</td>
<td>1.4</td>
<td>0.1</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.3</td>
<td>3.0</td>
<td>1.1</td>
<td>0.1</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.8</td>
<td>4.0</td>
<td>1.2</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.7</td>
<td>4.4</td>
<td>1.5</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.9</td>
<td>1.3</td>
<td>0.4</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.3</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.7</td>
<td>3.6</td>
<td>1.7</td>
<td>0.3</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.6</td>
<td>1.5</td>
<td>0.3</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.4</td>
<td>3.4</td>
<td>1.7</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
<tr>
<td>5.1</td>
<td>3.7</td>
<td>1.5</td>
<td>0.4</td>
<td>I. setosa</td>
</tr>
<tr>
<td>4.6</td>
<td>3.6</td>
<td>1.0</td>
<td>0.2</td>
<td>I. setosa</td>
</tr>
</tbody>
</table>
Applications of classification

- Text classification (spam filtering, news filtering, building web directories, etc.)

- Image classification (face detection, object recognition, etc.)

- Prediction of cancer recurrence.

- Financial forecasting.

- Many, many more!
Simple example

- Given “nucleus size”, predict cancer recurrence.
- Univariate input: $X = \text{nucleus size}$.
- Binary output: $Y = \{\text{NoRecurrence} = 0; \text{Recurrence} = 1\}$
- Try: Minimize the least-square error.
Predicting a class from linear regression

- Here red line is: \[ Y' = X (X^T X)^{-1} X^T Y \]
Predicting a class from linear regression

• Here red line is: \( Y' = X (X^TX)^{-1} X^TY \)

• How to get a binary output?
  1. Threshold the output:
     \( \{ y \leq t \text{ for NoRecurrence}, \ y > t \text{ for Recurrence} \} \)
  2. Interpret output as probability:
     \( y = Pr (\text{Recurrence}) \)

• What if we have more than 2 classes?

• Can we find a better model?
Refresher: Bayes’ rule

• Say we know:
  Probability dog barks in a day where we have a visitor: 0.9,
  Probability dog barks in a day where we have no visitor: 0.2,
  Probability of a visitor on a given day: 0.1

• What is the probability we have a visitor if the dog barks?
  Y = “dog barks”
  X = “visitor”

• We are interested in \( p(X=\text{true} \mid Y=\text{true}) \)
Refresher: Bayes’ rule

- Bayes’ rule: \[ p(X = x, Y = y) = P(X = x, Y = y) \]
  \[ p(x, y) = p(x, y) \]
  \[ p(x|y)p(y) = p(y|x)p(x) \]
  \[ p(x|y) = \frac{p(y|x)p(x)}{p(y)} \]

- Note that \( p(y) \) doesn’t depend on \( x \), it is a normalization constant.

- Can be calculated as: \[ p(y) = \sum_{x' \in \mathcal{X}} p(y|x)p(x) \]
Refresher: Bayes’ rule

• Now we can fill in the numbers:

\[
p(x|y) = \frac{p(y|x)p(x)}{p(y|x)p(x) + p(y|\neg x)p(\neg x)}
\]

\[
= \frac{0.9 \cdot 0.1}{0.9 \cdot 0.1 + 0.2 \cdot 0.9} = \frac{1}{3}
\]
Refresher: Multivariate Normal

\[ p(x) = \frac{\exp\left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right)}{(2\pi)^{1/2} |\Sigma|^{1/2}} \]

- Here:

\[ \mu = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \]

Mean first dimension

Mean 2\textsuperscript{nd} dimension

\[ \Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \]

Variance 1\textsuperscript{st} dimension

Variance 2\textsuperscript{nd} dimension

Covariance 1\textsuperscript{st} \& 2\textsuperscript{nd} dim
Refresher: Multivariate Normal

\[ p(x) = \frac{\exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)}{(2\pi)^{1/2}|\Sigma|^{1/2}} \]

\[ \Sigma = \begin{pmatrix} 1 & 0.5 \\ 0.5 & 1 \end{pmatrix} \]

Var \( x_1 = \sigma_1^2 \)
Covariance(\( x_1, x_2 \))
Var \( x_2 = \sigma_2^2 \)

Covariance(\( x_1, x_2 \)) is related to the correlation coefficient \( \rho_{12} \):

\[ \text{Covariance}(x_1, x_2) = \rho_{12} \sigma_1 \sigma_2 \]
Refresher: Multivariate Normal

\[ p(x) = \frac{\exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right)}{(2\pi)^{1/2} |\Sigma|^{1/2}} \]

(a) Dependent (correlated)
\[
\begin{pmatrix}
4 & 1.4 \\
1.4 & 2
\end{pmatrix}
\]

(b) Independent, different var
\[
\begin{pmatrix}
4 & 0 \\
0 & 2
\end{pmatrix}
\]

(c) Independent, same var
\[
\begin{pmatrix}
2 & 0 \\
0 & 2
\end{pmatrix}
\]

Image: C. Bishop: Pattern Recognition & Machine Learning
Modeling for binary classification

- Two probabilistic approaches:
  1. Generative learning
  2. Discriminative learning
Modeling for binary classification

- Two probabilistic approaches:

1. **Generative learning**: Separately model $P(x|y)$ and $P(y)$. Use Bayes rule, to estimate $P(y|x)$:

$$P(y = 1 | x) = \frac{P(x | y = 1)P(y = 1)}{P(x)}$$

2. **Discriminative learning**: Directly estimate $P(y|x)$. 

Example: Given "nucleus size" predict cancer recurrence

Example: Solution by linear regression
- Univariate real input: nucleus size
- Output coding: nonrecurrence and recurrence
- Sum squared error minimized by the red line
Modeling for binary classification

• Today, we’ll focus on generative learning

• Two algorithms:
  – Linear discriminant analysis (continuous inputs)
  – Naïve Bayes (mainly: binary input)

• Monday, we’ll finish up and discuss discriminative learning
Generative learning: Linear discriminant analysis (LDA)

- Consider Bayes’ rule: 
  \[ P(y \mid x) = \frac{P(x \mid y)P(y)}{P(x)} \]

- Calculate probability of both classes, and choose whichever is higher

- Simplification:

  \[
  \frac{p(x \mid y_0)p(y_0)}{p(x)} \ \overset{?}{>} \ \frac{p(x \mid y_1)p(y_1)}{p(x)}
  \]

  \[
  \log(p(x \mid y_0)p(y_0)) \ \overset{?}{>} \ \log(p(x \mid y_1)p(y_1))
  \]
Generative learning: Linear discriminant analysis (LDA)

- We want to determine \( \log(p(x|y_0)p(y_0)) > \log(p(x|y_1)p(y_1)) \)

- Make explicit assumptions about \( P(x|y) \)
Generative learning: Linear discriminant analysis (LDA)

- We want to determine \( \log(p(x|y_0)p(y_0)) > \log(p(x|y_1)p(y_1)) \)

- Make explicit assumptions about \( P(x|y) \): \( P(x \mid y) = \frac{e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}}{(2\pi)^{1/2} |\Sigma|^{1/2}} \)
  
  - Multivariate Gaussian (dim.\(=m\)), with mean \( \mu \) and covariance matrix \( \Sigma \).
  
  - Notation: here \( x \) is a single instance, represented as an \( m \times 1 \) vector.

  - Key assumption of LDA: Both classes have the same covariance matrix, \( \Sigma \).

COMP-551: Applied Machine Learning
Generative learning: Linear discriminant analysis (LDA)

- We want to determine $\log(p(x|y_0)p(y_0)) > \log(p(x|y_1)p(y_1))$

- Make explicit assumptions about $P(x|y)$: $P(x | y) = \frac{e^{-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)}}{(2\pi)^{1/2} |\Sigma|^{1/2}}$
  
  - Multivariate Gaussian (dim. = $m$), with mean $\mu$ and covariance matrix $\Sigma$.
  
  - Notation: here $x$ is a single instance, represented as an $m*1$ vector.
  
  - Key assumption of LDA: Both classes have the same covariance matrix, $\Sigma$.

- Fill into equation, ignoring constant terms that don’t depend on the class
  $$\log(p(y_0)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + x^T \Sigma^{-1} \mu_0 > \log(p(y_1)) - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + x^T \Sigma^{-1} \mu_1$$

  Linear discriminant: $\delta_0(x) > \delta_1(x)$
Learning in LDA: 2 class case

• Estimate $p_1 = P(Y=1), \mu, \Sigma$, from the training data
• E.g. using maximum likelihood
Learning in LDA: 2 class case

- Estimate $p_1 = P(Y=1)$, $\mu$, $\Sigma$, from the training data
- E.g. using maximum likelihood

$$
\log L(p_1, \mu_0, \mu_1, \Sigma \mid D) = \sum_{i=1:n} \left[ \log P(y_i) + \log P(x_i \mid y_i; \mu_0, \mu_1, \Sigma) \right]
$$

$$
= \sum_{i=1:n} \left[ y_i \log p_1 + (1-y_i) \log(1-p_1) 
+ y_i \log N(x_i \mid y_i; \mu_1, \Sigma) 
+ (1-y_i) \log N(x_i \mid y_i; \mu_0, \Sigma) \right]
$$

(will have other form if params $P(x\mid y)$ have other form, e.g. Gaussian).
Learning in LDA: 2 class case

- Estimate \( p_1 = P(Y=1) \), \( \mu \), \( \Sigma \), from the training data

- E.g. using maximum likelihood

\[
\log L(p_1, \mu_0, \mu_1, \Sigma \mid D) = \sum_{i=1:n} \left[ \log P(y_i) + \log P(x_i \mid y_i; \mu_0, \mu_1, \Sigma) \right] \\
= \sum_{i=1:n} \left[ y_i \log p_1 + (1-y_i) \log(1-p_1) \\
+ y_i \log N(x_i \mid y_i; \mu_1, \Sigma) \\
+ (1-y_i) \log N(x_i \mid y_i; \mu_0, \Sigma) \right]
\]

(will have other form if params \( P(x \mid y) \) have other form, e.g. Gaussian).

- To estimate \( p_1 \), take derivative of \( \log L \) with respect to \( p_1 \), set to 0:

\[
\frac{\partial L}{\partial p_1} = \sum_{i=1:n} \left( y_i / p_1 - (1-y_i) / (1-p_1) \right) = 0
\]
Training a Naïve Bayes classifier

Solving for $p_1$ we get:

$$
\sum_{i=1}^{n} \frac{y_i}{p_1} - \frac{1 - y_i}{1 - p_1} = 0
$$
Training a Naïve Bayes classifier

Solving for $p_1$ we get:

\[
\sum_{i=1}^{n} \frac{y_i}{p_1} - \frac{1 - y_i}{1 - p_1} = 0
\]

\[
\sum_{i=1}^{n} \frac{y_i}{p_1} = \sum_{i=1}^{n} \frac{1 - y_i}{1 - p_1}
\]

\[
\frac{1}{p_1} \sum_{i=1}^{n} y_i = \frac{1}{1 - p_1} \sum_{i=1}^{n} (1 - y_i)
\]

\[
\frac{1}{p_1} \sum_{i=1}^{n} y_i = \frac{1}{1 - p_1} \left( n - \sum_{i=1}^{n} y_i \right)
\]

\[
\frac{1 - p_1}{p_1} = \frac{n - \sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} y_i}
\]

\[
\frac{1}{p_1} - 1 = \frac{n}{\sum_{i=1}^{n} y_i} - 1
\]

\[
p_1 = \frac{1}{n} \sum_{i=1}^{n} y_i
\]

= fraction of examples where $y=1$!
Learning in LDA: 2 class case

- Estimate $P(y)$, $\mu$, $\Sigma$, from the training data, then apply log-odds ratio.

E.g. maximum likelihood estimates:

- $P(y=0) = N_0 / (N_0 + N_1)$
- $P(y=1) = N_1 / (N_0 + N_1)$

where $N_1$, $N_0$, be # of training samples from classes 1 and 0, respectively.
Learning in LDA: 2 class case

- Estimate $P(y)$, $\mu$, $\Sigma$, from the training data, then apply log-odds ratio.

E.g. maximum likelihood estimates:

- $P(y=0) = N_0 / (N_0 + N_1)$  \hspace{1cm}  $P(y=1) = N_1 / (N_0 + N_1)$
  where $N_1$, $N_0$, be # of training samples from classes 1 and 0, respectively.

- Similar, we find the maximum likelihood estimate for the Gaussian distribution (as seen in probability course)

  - $\mu_0 = \sum_{i=1:n} I(y_i=0) x_i / N_0$  \hspace{1cm}  $\mu_1 = \sum_{i=1:n} I(y_i=1) x_i / N_1$
  where $I(x)$ is the indicator function: $I(x)=0$ if $x=0$, $I(x)=1$ if $x=1$. 
Learning in LDA: 2 class case

- Estimate $P(y)$, $\mu$, $\Sigma$, from the training data, then apply log-odds ratio.

E.g. maximum likelihood estimates:

- $P(y=0) = N_0 / (N_0 + N_1)$
  
  $P(y=1) = N_1 / (N_0 + N_1)$

  where $N_1$, $N_0$, be # of training samples from classes 1 and 0, respectively.

- Similar, we find the maximum likelihood estimate for the Gaussian distribution (as seen in probability course)

  $\mu_0 = \sum_{i=1:n} I(y_i=0) x_i / N_0$

  $\mu_1 = \sum_{i=1:n} I(y_i=1) x_i / N_1$

  where $I(x)$ is the indicator function: $I(x)=0$ if $x=0$, $I(x)=1$ if $x=1$.

- $\Sigma = \sum_{k=0:1} \sum_{i=1:n} I(y_i=k) (x_i - \mu_k)(x_i - \mu_k)^T / (N_0 + N_1)$
Learning in LDA: 2 class case

• Now ready to make a decision, decide class 0 if:

\[ \delta_0(x) > \delta_1(x) \]

\[ \log(p(y_0)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + x^T \Sigma^{-1} \mu_0 > \log(p(y_1)) - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + x^T \Sigma^{-1} \mu_1 \]

• Rearrange terms:

\[ \log(p(y_0)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + x^T \Sigma^{-1} \mu_0 - \log(p(y_1)) + \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 - x \Sigma^{-1} \mu_1 > 0 \]

\[ \log(p(y_0)) - \log(p(y_1)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + x^T \Sigma^{-1} (\mu_0 - \mu_1) > 0 \]
Learning in LDA: 2 class case

• Now ready to make a decision, decide class 0 if:

\[ \delta_0(x) > \delta_1(x) \]

\[
\log(p(y_0)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + x^T \Sigma^{-1} \mu_0 > \log(p(y_1)) - \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + x^T \Sigma^{-1} \mu_1
\]

• Rearrange terms:

\[
\log(p(y_0)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + x^T \Sigma^{-1} \mu_0 - \log(p(y_1)) + \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 - x \Sigma^{-1} \mu_1 > 0
\]

\[
\log(p(y_0)) - \log(p(y_1)) - \frac{1}{2} \mu_0^T \Sigma^{-1} \mu_0 + \frac{1}{2} \mu_1^T \Sigma^{-1} \mu_1 + x^T \Sigma^{-1} (\mu_0 - \mu_1) > 0
\]

This is a linear decision boundary! \[ W_0 \] \[ x^T w_1 \]
Learning in LDA: 2 class case

**FIGURE 2.1.** A classification example in two dimensions. The classes are coded as a binary variable (**BLUE** = 0, **ORANGE** = 1), and then fit by linear regression. The line is the decision boundary defined by $x^T \hat{\beta} = 0.5$. The orange shaded region denotes that part of input space classified as **ORANGE**, while the blue region is classified as **BLUE**.
More complex decision boundaries

- Want to accommodate more than 2 classes?
  - Consider the per-class linear discriminant function:
    \[
    \delta_y(x) = x^T \Sigma_y^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma_y^{-1} \mu_y + \log P(y)
    \]
  - Use the following decision rule:
    \[
    \text{Output} = \arg\max_y \delta_y(x) = \arg\max_y \left[ x^T \Sigma_y^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma_y^{-1} \mu_y + \log P(y) \right]
    \]
LDA decision boundaries – 3 class case

**FIGURE 4.5.** The left panel shows three Gaussian distributions, with the same covariance and different means. Included are the contours of constant density enclosing 95% of the probability in each case. The Bayes decision boundaries between each pair of classes are shown (broken straight lines), and the Bayes decision boundaries separating all three classes are the thicker solid lines (a subset of the former). On the right we see a sample of 30 drawn from each Gaussian distribution, and the fitted LDA decision boundaries.
More complex decision boundaries

• Want to accommodate more than 2 classes?
  – Consider the per-class linear discriminant function:
    \[
    \delta_y(x) = x^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log P(y)
    \]
  – Use the following decision rule:
    \[
    \text{Output} = \arg\max_y \delta_y(x) = \arg\max_y \left[ x^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log P(y) \right]
    \]

• Want more flexible (non-linear) decision boundaries?
  – Recall trick from linear regression of re-coding variables.
Using higher-order features

**FIGURE 4.1.** The left plot shows some data from three classes, with linear decision boundaries found by linear discriminant analysis. The right plot shows quadratic decision boundaries. These were obtained by finding linear boundaries in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$. Linear inequalities in this space are quadratic inequalities in the original space.
Using higher-order features

Use distance from the two + signs in original space (left) as features.

Chapter 3 has more details on linear regression. Note that we
thus if

4.2 Linear Regression of an Indicator Matrix

Here each of the response categories are coded via an indicat

Quadratic discriminant function:

Linear discriminant function:

\[ \delta_y(x) = x^T \Sigma^{-1} \mu_y - \frac{1}{2} \mu_y^T \Sigma^{-1} \mu_y + \log P(y) \]

Quadratic discriminant function:

\[ \delta_y(x) = -\frac{1}{2} |\Sigma_y| - \frac{1}{2} (x - \mu_y)^T \Sigma_y^{-1} (x - \mu_y) + \log P(y) \]

- QDA has more parameters to estimate, but greater flexibility to estimate the target function. Bias-variance trade-off.
FIGURE 4.6. Two methods for fitting quadratic boundaries. The left plot shows the quadratic decision boundaries for the data in Figure 4.1 (obtained using LDA in the five-dimensional space $X_1, X_2, X_1X_2, X_1^2, X_2^2$). The right plot shows the quadratic decision boundaries found by QDA. The differences are small, as is...
What you should know (LDA)

• Basic definition of linear classification problem.
• Linear discriminant analysis: definition, decision boundary.
• Quadratic discriminant analysis: basic idea, decision boundary.
• LDA vs QDA pros/cons.

• Worth reading further:
  – Under some conditions, linear regression for classification and LDA are the same (Hastie et al., p.109-110).