Statistical Methods for High Dimensional Biology

Supervised learning I: Classification

Keegan Korthauer

10 March 2021

with slide contributions from Sara Mostafavi, Gabriela Cohen-Freue, and Kevin Murphy

Announcements

- Project presentation dates posted today
- Project progress reports due Friday 19 March
- Next week's lectures (15 and 17 March) will both be **synchronous**
    - Guest lecturer Paul Pavlidis will talk about two topics that will be very relevant for many projects: Gene set enrichment analysis, and Gene networks and function prediction

Learning objectives

- Explain the purpose of **supervised learning** and how it differs from **unsupervised learning**
- Connect commonly used terms from statistics and machine learning
- Explain the goals of **classification**
- Understand the main ideas behind the mathematical frameworks such as **Naïve Bayes**, **Linear Discriminant Analysis**, and **K-Nearest Neighbors classification**

Supervised learning

A procedure or algorithm which uses a set of **inputs** (measured or preset variables) to predict the values of one or more **outputs** (variables which are influenced in some way by the inputs)

- This definition uses the language/terminology from the field of *machine learning*
- In statistical terminology:
    - we would say *predictor* or *independent* variables in place of **inputs**
    - we would say *response* or *dependent* variables in place of **outputs**

Examples in genomics

Machine learning vs classical statistics

Machine Learning

- Large number of variables (have no idea which are useful)
- Model complex, non-linear relationships
- Flexibility about defining a classifier: “loss/error minimization view” vs “generative” view
- Invent scalable algorithms that can solve parameters for very large models

Classical Statistics

- Handful of variables
- Typically assume linear relationships
- Typically think in terms of a “generative” model; has theoretical justification
- Thorough analysis/theory for models with less than a dozen parameters

ML vs Statistics terminology

Machine Learning	Classical Statistics
Labels / 'class' labels	Response / outcome
Examples	Data points
Features	Covariates / variables
Learning	Estimation / fitting
Weights / feature importance	Parameters
Generalization	Test set performance

Example: Predict phenotype from gene expression

1. Measure gene expression data relevant for the outcome you would like to predict (e.g., disease status or severity): **training data**

2. Formulate (i.e., write down) a **model** that relates the gene expression measurements (i.e. features/attributes) to the outcome

3. Fit/estimate model **parameters** based on data to fully specify the model

4. Apply the model to new data, where you don’t have (or at least don't *use*) information about outcome/response to make a prediction

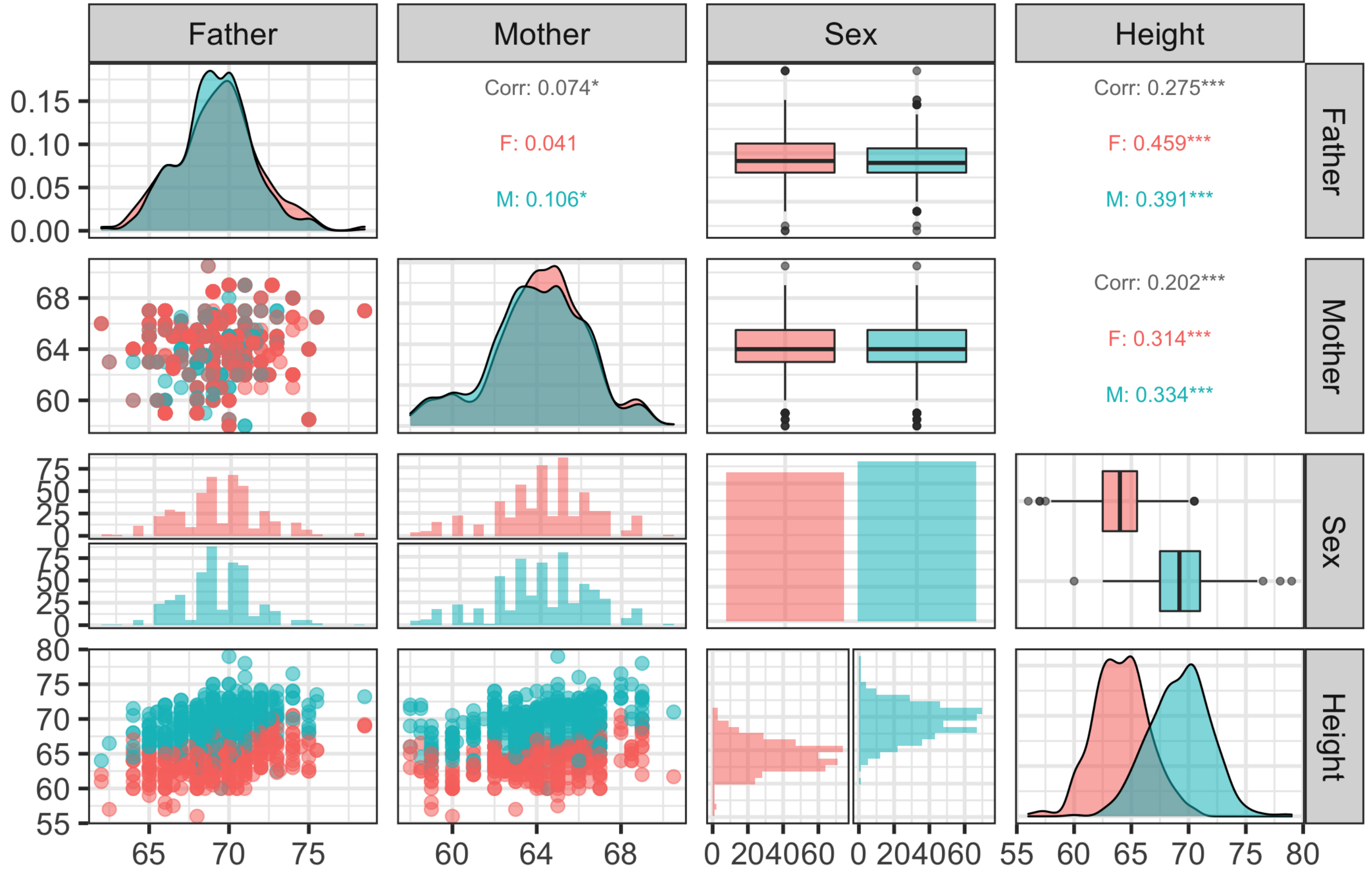
Galton's Height Data: predict the future (adult) height of a child

Data source

1. Gather training data

$\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_n, y_n)\}$

- **input**: $\boldsymbol{x}_i$, feature vector (Father height, mother height, sex)
- **output**: y_i, response (child height)



2. Formulate model relating input and output

- Write down a model that links the output variable to some function of the input variable(s)

$$y_i = f(\boldsymbol{x}_i) + \epsilon$$

- For example, let's say child height is linearly related to the additive effects of parental mean height and sex

$$y_i = \alpha + \beta_1 \left(\frac{x_{father,i} + x_{mother,i}}{2} \right) + \beta_2 x_{male,i} + \epsilon$$

- $x_{father,i}$ and $x_{mother,i}$ are the heights of the father and mother of child i, and $x_{male,i}$ is an indicator variable that the i^{th} child's sex is male
- α, β_1, and β_2 are model parameters

3. Fit the model to training data

- Let $\bar{x}_{parental,i}$ represent parental mean height
- How can we fit this model to minimize *error* on the training data?

$$y_i = \alpha + \beta_1 \bar{x}_{parental,i} + \beta_2 x_{male,i} + \epsilon_i, \text{ where } \epsilon_i \sim N(0, \sigma^2)$$

- e.g. find α, β_1, and β_2 such that the objective function (sum of squared errors) is minimized

$$\sum_{i=1}^{n} (y_i - \alpha - \beta_1 \bar{x}_{parental,i} - \beta_2 x_{male,i})^2$$

3. Fit the model to training data

- Let $\bar{x}_{parental,i}$ represent parental mean height
- How can we fit this model to minimize *error* on the training data?

$$y_i = \alpha + \beta_1 \bar{x}_{parental,i} + \beta_2 x_{male,i} + \epsilon_i, \text{ where } \epsilon_i \sim N(0, \sigma^2)$$

- e.g. find α, β_1, and β_2 such that the objective function (sum of squared errors) is minimized

$$\sum_{i=1}^{n} (y_i - \alpha - \beta_1 \bar{x}_{parental,i} - \beta_2 x_{male,i})^2$$

Linear regression!

Aside: generative model

(Mathematically equivalent) probabilistic formulation of linear regression:

$$y_i|\bar{x}_{parental,i}, x_{male,i} \ \sim \ N(\alpha + \beta_1\bar{x}_{parental,i} + \beta_2 x_{male,i},\ \sigma^2)$$

$$p(y_1, y_2, \ldots, y_n|\boldsymbol{x_1}, \boldsymbol{x_2}, \ldots, \boldsymbol{x_n}) = \prod_{i=1}^{n} f_N(y_i|\alpha + \beta_1\bar{x}_{parental,i} + \beta_2 x_{male,i},\ \sigma^2)$$

Where $f_N(y|\mu, \sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{y-\mu}{\sigma}\right)^2}$ is the normal probability density function

4. Apply model to predict on new data

- Suppose we collect a new dataset of parental mean heights $\bar{x}^*_{parental,i}$ and child's sex $x^*_{male,i}$ for an independent set of m children $i = 1, \ldots, m$

- We would like to predict each child's eventual adult height $\hat{y}^*_i$ based their sex and their parents' mean height

- **How?** Use the model parameters estimated from the training data $(\hat{\alpha}, \hat{\beta}_1, \hat{\beta}_2)$ and plug in our predictor variables $(\bar{x}^*_{parental,i}, x^*_{male,i})$

$$\hat{y}^*_i = \hat{\alpha} + \hat{\beta}_1 \bar{x}^*_{parental,i} + \hat{\beta}_2 x^*_{male,i}$$

Supervised learning

Regression

- continuous outcome

Classification

- binary outcome
- categorical outcome

The classification problem

Training data:

$$\{(\boldsymbol{x_1}, c_1), (\boldsymbol{x_2}, c_2), \ldots, (\boldsymbol{x_n}, c_n)\}$$

Instead of a *continuous* outcome/response (e.g. height), we now have discrete **class labels** $c_i \in \{1, \ldots, K\}$

Classifiers

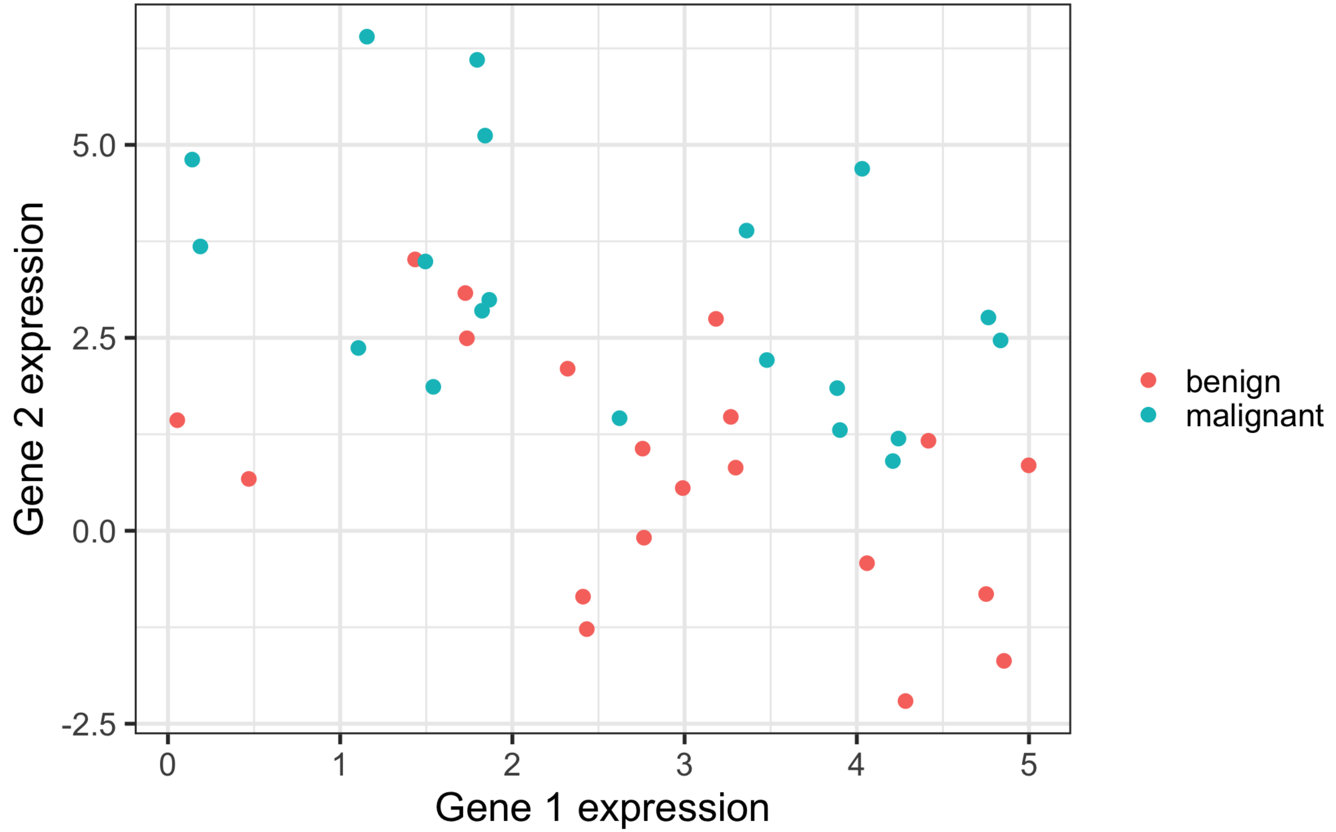
A **classifier** is a function f that maps input feature vectors $\boldsymbol{x_i} = \{x_{i1}, x_{i2}, \ldots, x_{ip}\}$ to output class labels $c_i \in \{1, \ldots, K\}$

- we assume that class i labels are *unordered* and *mutually exclusive*
- Let $\mathcal{X}$ be the feature space of all possible values of $\boldsymbol{x_i}$
  - this space could consist of continuous, discrete values or a mixture of the two (e.g. $\mathcal{X} = \{0, 1\}^p$ or $\mathcal{X} = \mathbb{R}^p$)
- **Goal**: to learn a function f that maps feature vectors to labels, based on labeled training set: $\{(\boldsymbol{x_1}, c_1), (\boldsymbol{x_2}, c_2), \ldots, (\boldsymbol{x_n}, c_n)\}$

$$f(\boldsymbol{x_i}) = c_i$$

Example classification task

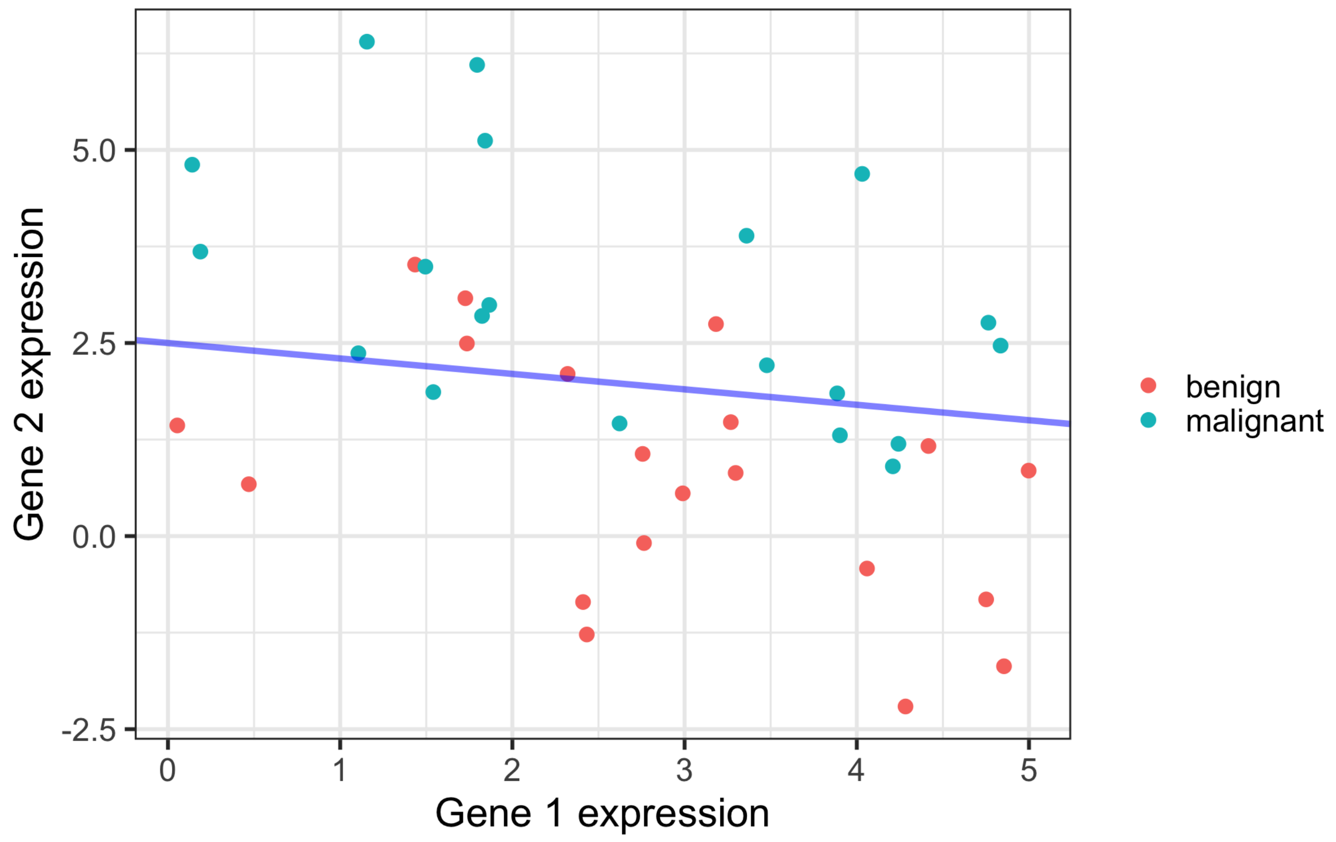
Partition the space of input data so that we minimize the number of "miss-classified" objects/points



Example classification task

Partition the space of input data so that we minimize the number of “miss-classified” objects/points

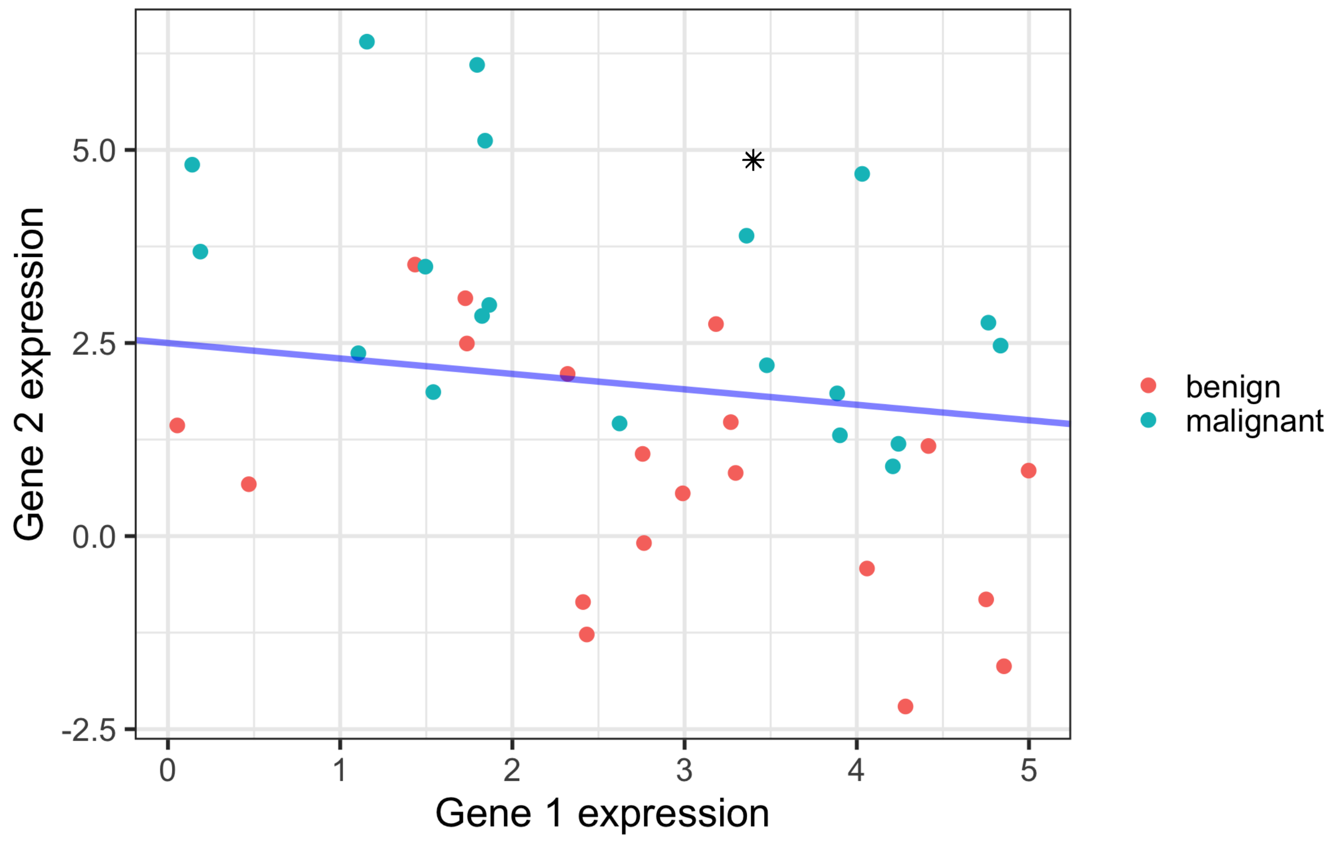
Our classification boundary might be more or less complex (e.g. nonlinear vs linear)



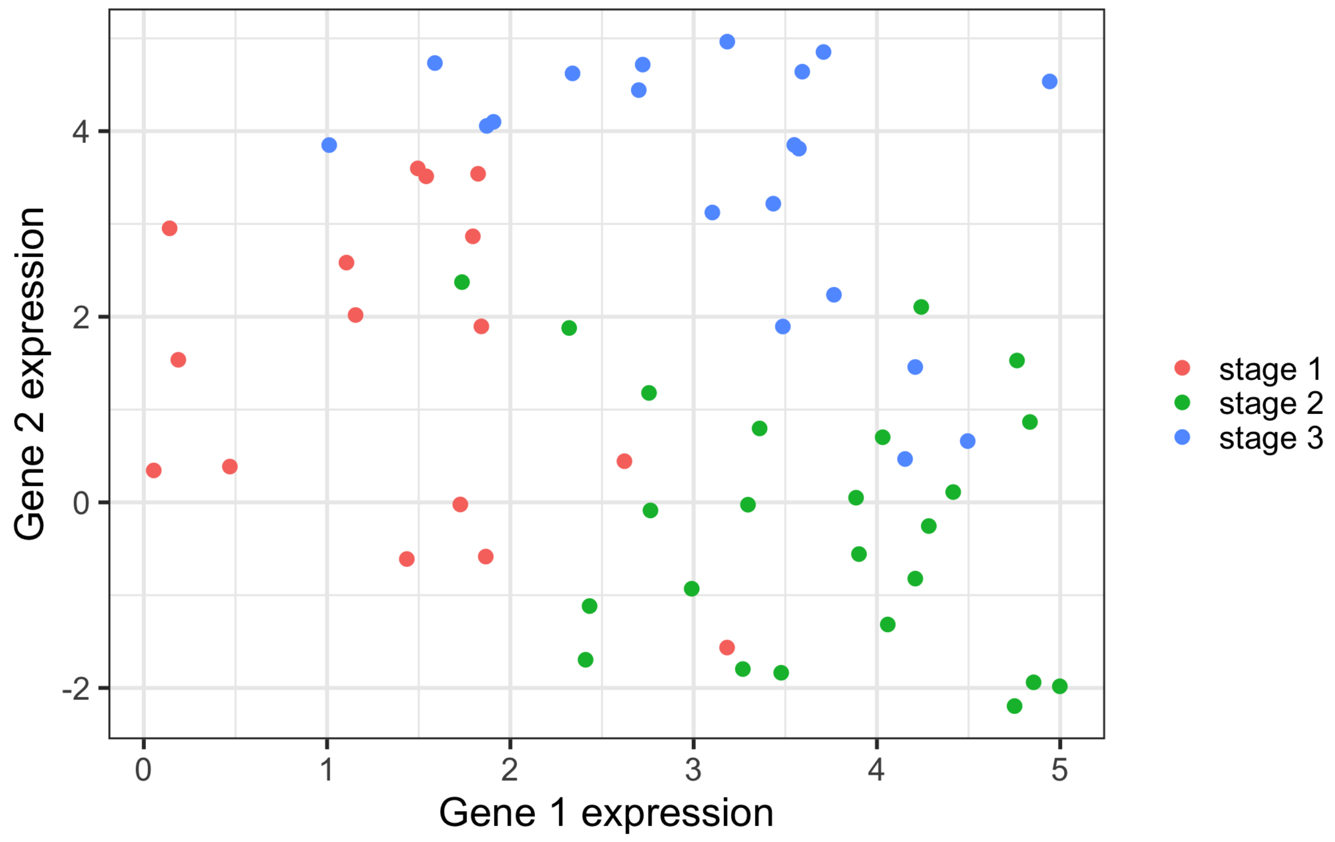
Example classification task

Partition the space of input data so that we minimize the number of "miss-classified" objects/points

Our classification boundary might be more or less complex (e.g. nonlinear vs linear)



More than 2 groups



Defining the cost of misclassification

To train a model, we need a **loss / error** function

The **cost** of a misclassification can be specified with a loss (or error) function

For example, if all misclassification errors are equally bad, the loss function for predicted class labels $\hat{c}_i$ given true class labels c_i would be

$$L(c_i, \hat{c}_i) = \begin{cases} 0 & \text{if } c_i = \hat{c}_i \\ 1 & \text{otherwise} \end{cases}$$

Minimizing the expected loss

Goal: predict the class that minimizes the *conditional expected loss*

- Expected conditional loss: $\rho(\hat{c}(\boldsymbol{x_i})) = \sum_{k=1}^{K} L(k, \hat{c}(\boldsymbol{x}_i))P(C = k|\boldsymbol{x_i})$
- Simplest case of two classes:
    - Expected loss of predicting class 1 when label was 2: $L(2, 1)p(2|\boldsymbol{x_i})$
    - Expected loss of predicting class 2 when label was 1: $L(1, 2)p(1|\boldsymbol{x_i})$
    - Predict class 1 if: $L(2, 1)p(2|\boldsymbol{x_i}) < L(1, 2)p(1|\boldsymbol{x_i})$
- But how to get $P(C = k|\boldsymbol{x_i})$?

Three main ways to solve classification problem

1. Learn a **generative model** (function) for the probability distribution of inputs for each class: $p(\boldsymbol{x_i}|C=k)$
   - Then use **Bayes rule** and the marginal distribution $p(C=k)$ (overall class prevalence) to predict $p(C=k|\boldsymbol{x_i})$
   - Recall Bayes rule: $P(A|B) = \frac{P(B|A)P(A)}{P(B)}$
2. Learn a **discriminative model** for conditional probability distribution of each class $p(C=k|\boldsymbol{x_i})$
   - Do not consider the distribution of the predictors $\boldsymbol{x_i}$
3. Learn a **non-parametric model**
   - e.g. a function that directly maps $\boldsymbol{x_i}$ to its predicted class c

Example classification methods

1. Generative
    - **Naïve Bayes**
    - **Linear/Quadratic discriminant analysis**
2. Discriminative
    - **Logistic regression**
    - Support vector machines
    - Decision trees (and Random Forest)
    - Neural networks
3. Non-parametric
    - **K-nearest neighbors**

Generative model solution

Learn the following for each value of $k \in \{1, \ldots, K\}$:

- **class-conditional density** $p(\boldsymbol{x_i} | C = k)$
- **class priors** (overall class prevalence) $p(C = k)$

Then apply Bayes rule to compute most likely class for each object/entity (*posterior*)

$$\begin{aligned} p(C = k | \boldsymbol{x_i}) &= \frac{p(\boldsymbol{x_i} | C = k) p(C = k)}{p(\boldsymbol{x_i})} \\ &= \frac{p(\boldsymbol{x_i} | C = k) p(C = k)}{\sum_{j=1}^{K} p(\boldsymbol{x_i} | C = j) p(C = j)} \end{aligned}$$

Naïve Bayes

- Most general of the **generative model** techniques
- Assumes features $\{x_{i1}, x_{i2}, \ldots, x_{ip}\}$ are **independent**
- Since features are independent, the conditional density of features given class can be written as the product of the individual feature conditional densities: $p(\boldsymbol{x_i}|C=k) = \prod_{m=1}^{p} p(x_{im}|C=k)$
- Useful only when the number of predictors is small (otherwise, hard to estimate all conditional distributions of features given class $p(x_{im}|C=k)$)

Gaussian Naïve Bayes

Assume features within each class are *independently* normally (or Guassian) distributed

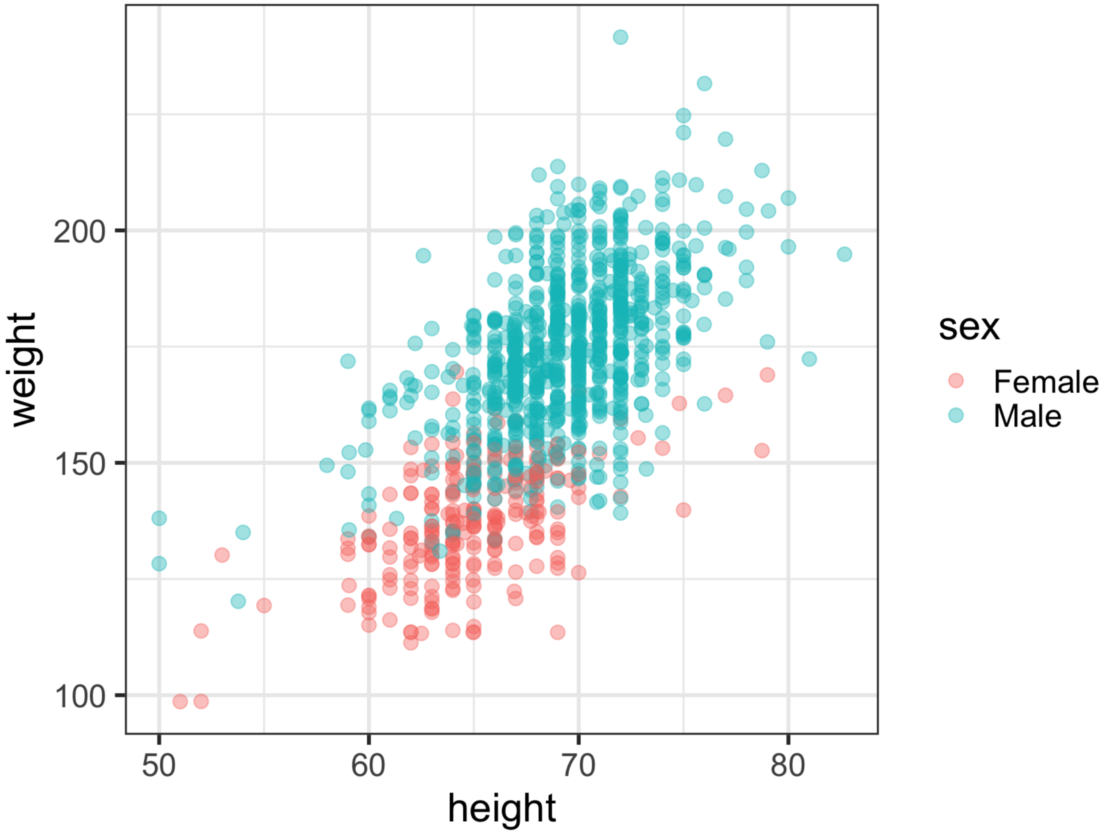
$$
\begin{aligned}
p(\boldsymbol{x_i}|C=k) &= \prod_{m=1}^{p} p(x_{im}|C=k) \\
&= \prod_{m=1}^{p} f_N(x_{im}|\mu_{mk}, \sigma_{mk})
\end{aligned}
$$

where $f_N(x|\mu, \sigma)$ is the Normal probability density function with mean μ and sd σ

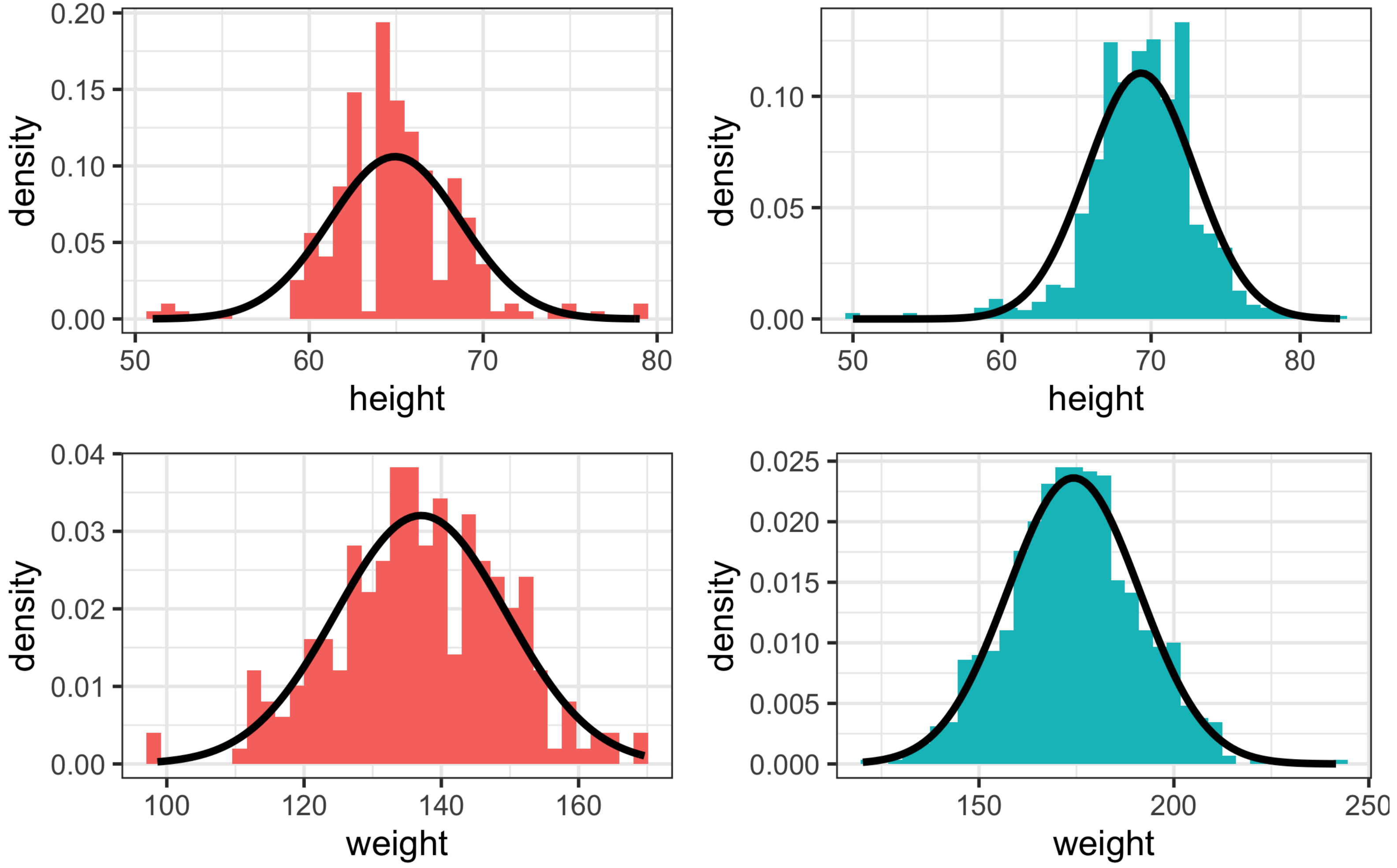
Estimate $\boldsymbol{\mu_k} = \{\mu_{1k}, \ldots, \mu_{pk}\}$, and $\boldsymbol{\sigma_k} = \{\sigma_{1k}, \ldots, \sigma_{pk}\}$ using maximum likelihood estimation (MLE) on training data

Example: Height data

Data/Example modified from: Intro to Data Science by Irizarry



Class-conditional densities of features



Class-conditional densities of features

```r
(mles <- heights %>%
  group_by(sex) %>%
  summarize(mean_height = mean(height),
            sd_height = sd(height),
            mean_weight = mean(weight),
            sd_weight = sd(weight)))
```

```
## # A tibble: 2 x 5
##   sex    mean_height sd_height mean_weight sd_weight
## * <fct>        <dbl>     <dbl>       <dbl>     <dbl>
## 1 Female        64.9      3.76        137.      12.5
## 2 Male          69.3      3.61        174.      16.9
```

e.g. $x_{i,weight}|C = Female \sim N(137, 12.5)$

Prediction using Gaussian Naïve Bayes

Goal: compute the **posterior probability** of class assignment for some new observation $\boldsymbol{x}^*$

$$\begin{aligned} p(C=k|\boldsymbol{x_i}) &= \frac{p(\boldsymbol{x_i}|C=k)p(C=k)}{\sum_{j=1}^{K} p(\boldsymbol{x_i}|C=j)p(C=j)} \\ &= \frac{p(C=k)\prod_{m=1}^{p} p(x_{im}|C=k)}{\sum_{j=1}^{k} p(C=j)\prod_{m=1}^{p} p(x_{im}|C=j)} \\ &= \frac{p(C=k)\prod_{m=1}^{p} f_N(x_{im}|\mu_{mk},\sigma_{mk})}{\sum_{j=1}^{K} p(C=j)\prod_{m=1}^{p} f_N(x_{im}|\mu_{mj},\sigma_{mj})} \end{aligned}$$

Prediction using Gaussian Naïve Bayes

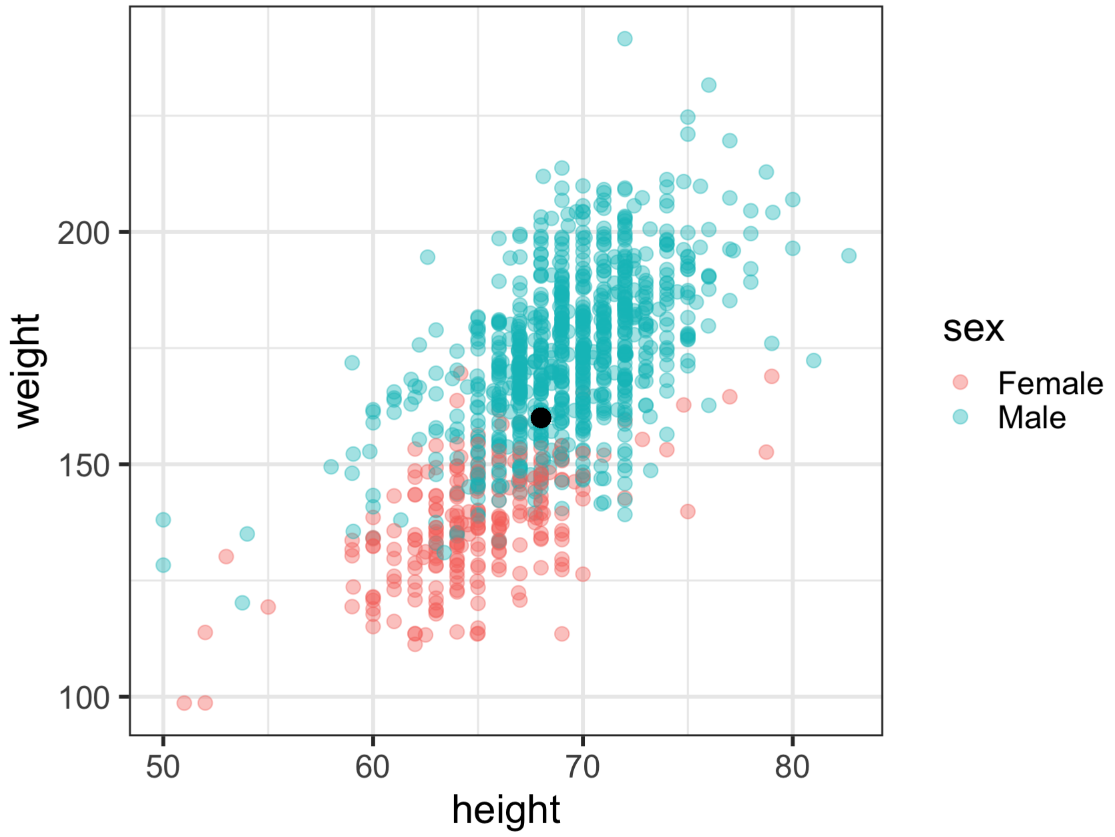
In our two-class example with heights,

$$p(C=k|\boldsymbol{x_i}) = \frac{p(C=k)\prod_{m=1}^{p} f_N(x_{im}|\mu_{mk},\sigma_{mk})}{p(C=1)\prod_{m=1}^{p} f_N(x_{im}|\mu_{m1},\sigma_{m1}) + p(C=2)\prod_{m=1}^{p} f_N(x_{im}|\mu_{m2},\sigma_{m2})}$$

- $p(C=1) =$ overall proportion in class "Female" in training data (0.227)
- $p(C=2) =$ overall proportion in class "Male" in training data (0.773)
- Male class height distribution: $x_{i,height}|\mu_{height,2},\sigma_{height,2} \sim N(69.3, 3.61)$
- Likewise we also have the female class height distribution, and the male and female class weight distributions (all Gaussian)

Prediction using Gaussian Naïve Bayes

For example, let's say we have a new observation: $\boldsymbol{x}^* = (x^*_{height} = 68, x^*_{weight} = 160)$



Prediction using Gaussian Naïve Bayes

For example, if $\boldsymbol{x}^* = (x^*_{height} = 68, x^*_{weight} = 160)$ we can compute $p(C = 2|\boldsymbol{x}^*)$ (predicted probability that this sample is male):

```r
new_height <- 68
new_weight <- 160
pM <- sum(heights == "Male")/nrow(heights)
mle_male <- mles %>% filter(sex == "Male")
mle_female <- mles %>% filter(sex == "Female")
num <- pM * (dnorm(new_height, mle_male$mean_height, mle_male$sd_height)*
             dnorm(new_weight, mle_male$mean_weight, mle_male$sd_weight))
den <- num +
  (1-pM) * (dnorm(new_height, mle_female$mean_height, mle_female$sd_height)*
            dnorm(new_weight, mle_female$mean_weight, mle_female$sd_weight))

num/den
```

```
## [1] 0.9288177
```

Prediction using Gaussian Naïve Bayes

Prediction/fitting on entire training dataset (classification rule assigns each observation to the class with posterior probability > 0.5)

```r
num <- pM * (dnorm(heights$height, mle_male$mean_height, mle_male$sd_height)*
             dnorm(heights$weight, mle_male$mean_weight, mle_male$sd_weight))
den <- num +
  (1-pM) * (dnorm(heights$height, mle_female$mean_height, mle_female$sd_height)*
            dnorm(heights$weight, mle_female$mean_weight, mle_female$sd_weight))
predict <- ifelse(num/den > 0.5, "Male", "Female")

# confusion matrix
table(predict, heights$sex)
```

```
##
## predict  Female Male
##    Female    190   50
##    Male       48  762
```

Linear Discriminant Analysis (LDA)

- Naïve Bayes is pretty naïve
    - assumes independence of features
- LDA relaxes this assumption
- LDA still assumes that features are normally distributed
    - In particular, that they are **Multivariate normal**: $\boldsymbol{x}_i | C = k \sim N(\boldsymbol{\mu}_k, \boldsymbol{\Sigma})$
    - $\boldsymbol{\Sigma}$ is the covariance matrix (defines relationship among features; in Naïve Bayes, can write as a diagonal matrix)
    - LDA assumes $\boldsymbol{\Sigma}$ is the *same for each class*

LDA classifier

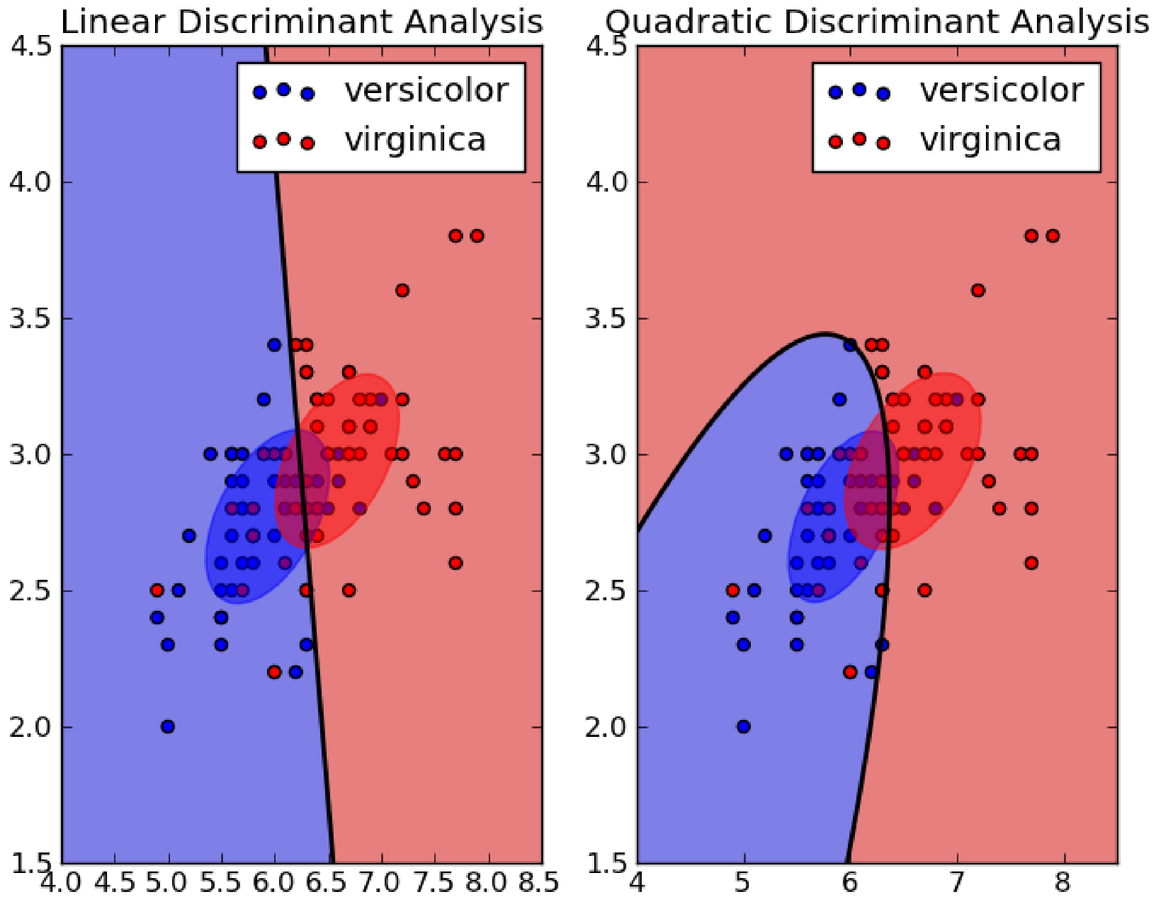
- $p(C = k)$ and μ_k estimated from MLE as in Naïve Bayes
- Pooled Sample covariance estimate $\hat{\boldsymbol{\Sigma}} = \frac{1}{n} \sum_{k=1}^{K} \sum_{i:c_i=k} (\boldsymbol{x}_i - \mu_k)(\boldsymbol{x}_i - \mu_k)^T$
- Predict class k for observation i that maximizes

$$p(C = k) f_{MVN}(\boldsymbol{x}_i | \hat{\boldsymbol{\mu}}_k, \hat{\boldsymbol{\Sigma}})$$

where $f_{MVN}(\boldsymbol{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the probability density function of the Multivariate Normal distribution

- Turns out that the decision rules for classifying into class k versus k' are **linear** combinations of the predictors
- If we instead let $\boldsymbol{\Sigma}$ be different for each class k, this is **Quadratic discriminant analysis**: decision boundaries are quadratic curves

LDA vs QDA


Linear Discriminant Analysis
Quadratic Discriminant Analysis
versicolor
virginica
4.5
4.0
3.5
3.0
2.5
2.0
1.5
4.0 4.5 5.0 5.5 6.0 6.5 7.0 7.5 8.0 8.5
4 5 6 7 8

LDA in height/weight example

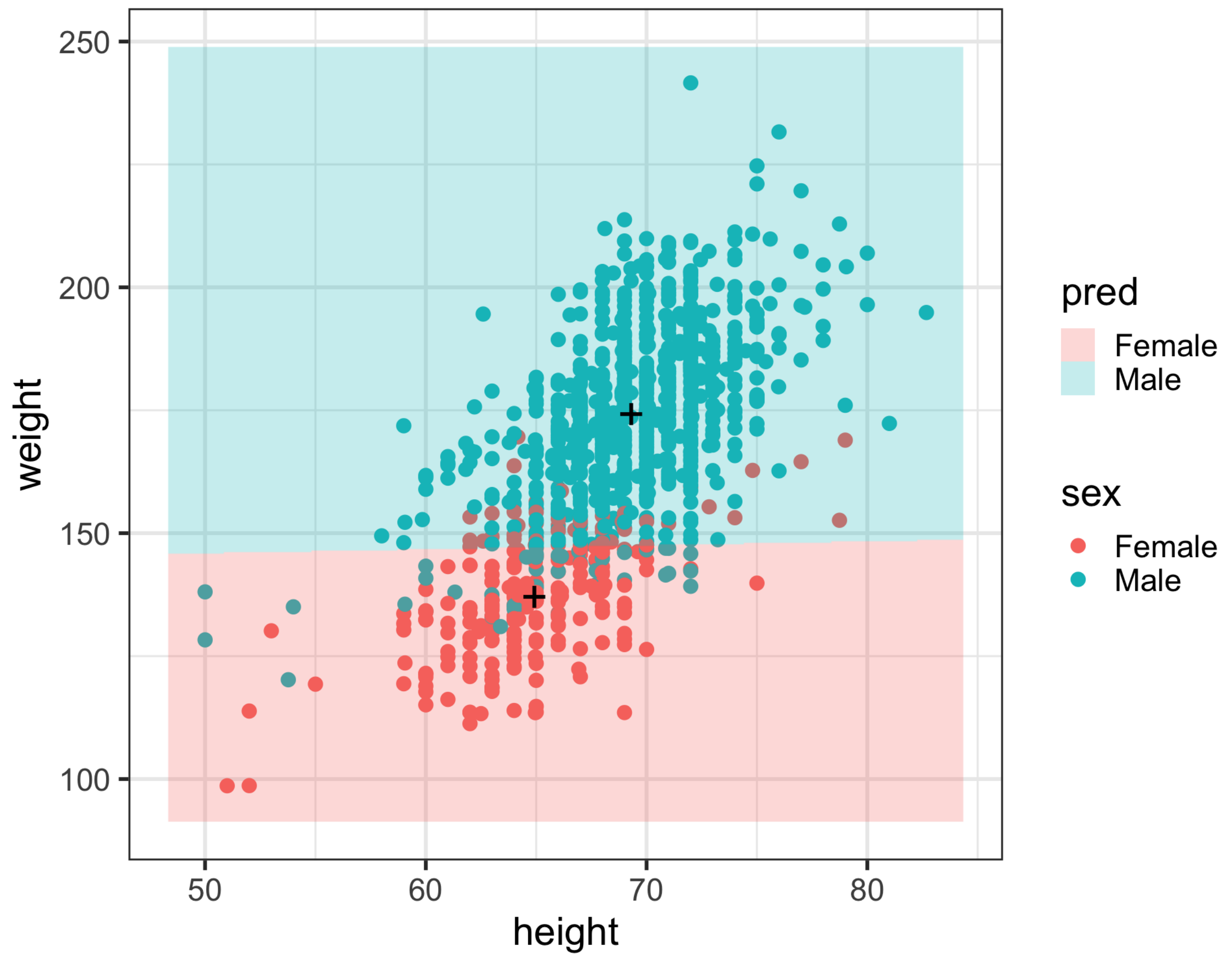
```
height_lda = lda(sex ~ weight + height,
                 data = heights)
height_lda
```

```
## Call:
## lda(sex ~ weight + height, data = heights)
##
## Prior probabilities of groups:
##    Female      Male
## 0.2266667 0.7733333
##
## Group means:
##           weight   height
## Female 137.0573 64.93942
## Male   174.2648 69.31475
##
## Coefficients of linear discriminants:
##                 LD1
## weight  0.063123732
## height -0.005069971
```

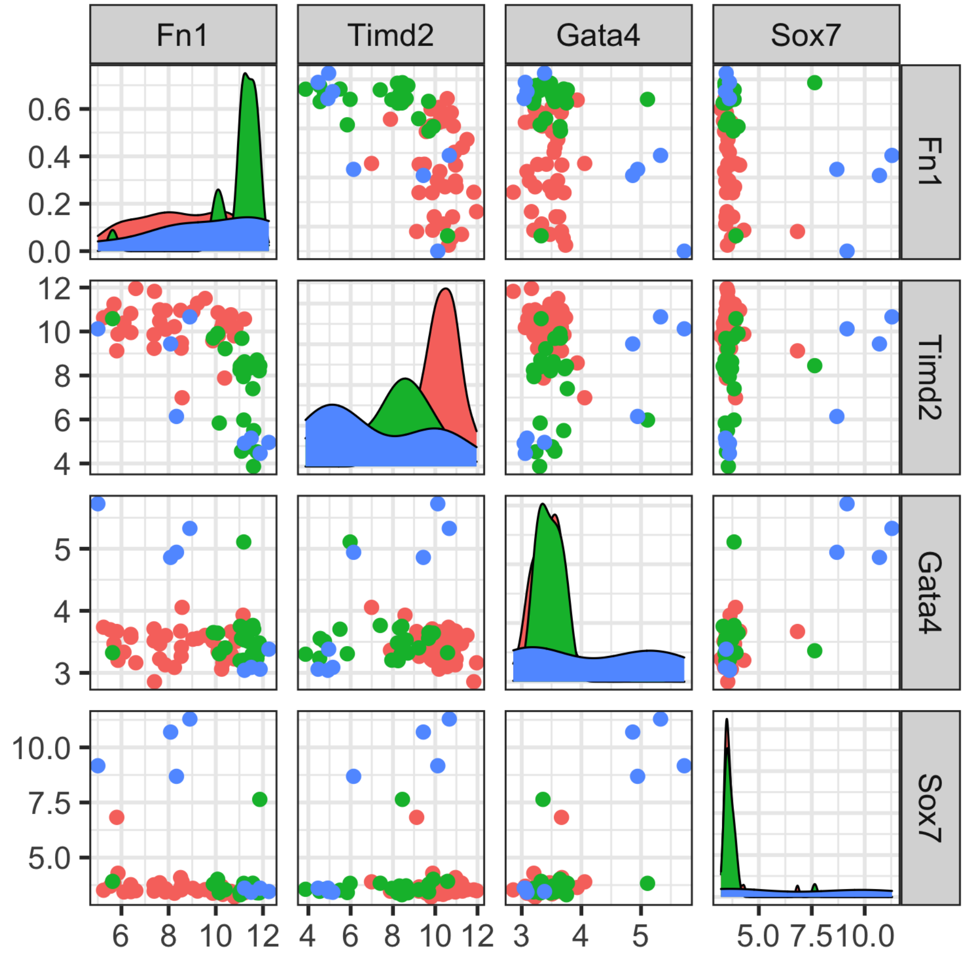
```
# confusion matrix
table(predict(height_lda)$class,
      heights$sex)
```

```
##
##          Female Male
##   Female    186   47
##   Male       52  765
```

Visualization of decision boundary



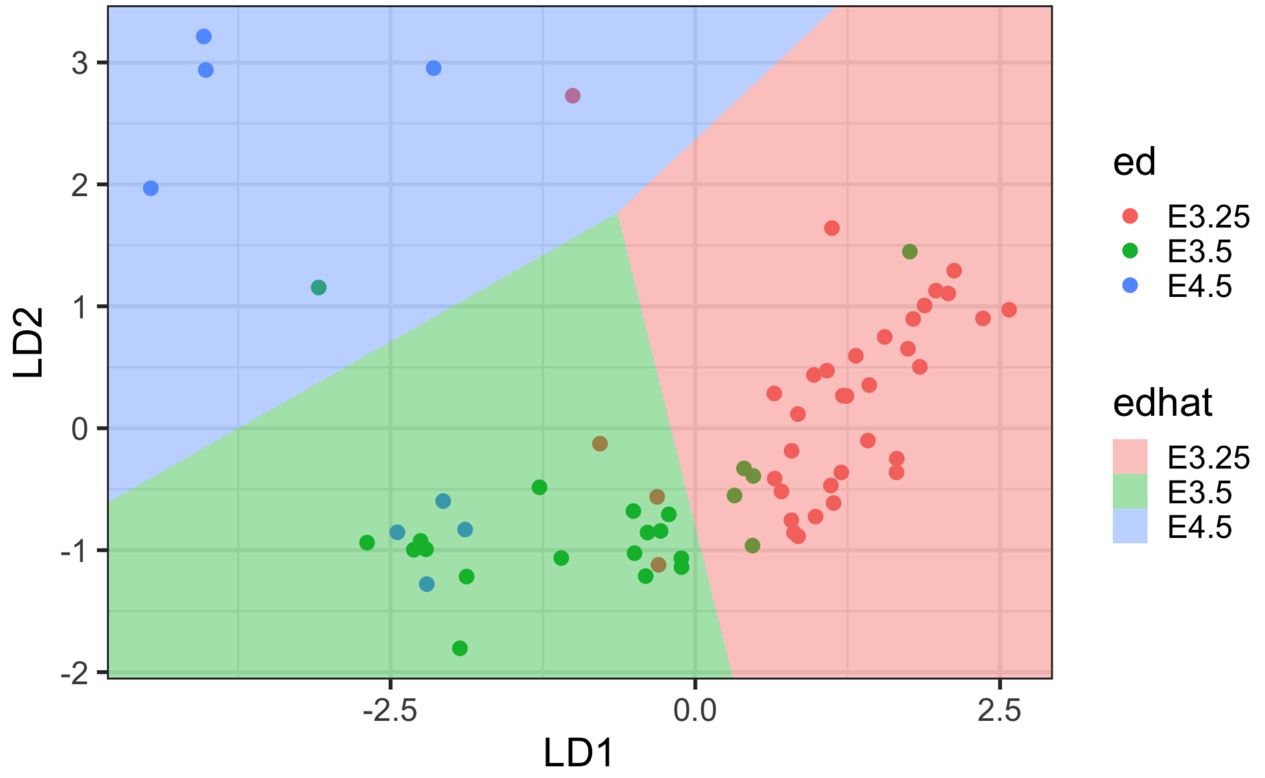
Generalizing LDA to more than 2 classes



- We'd like to build a classifier that predicts the embryonic cell state from gene expression
- Specifically, we'd like to classify observations into one of three developmental time points (embryonic days: E3.25, E3.5, E4.5, shown in colour)
- Suppose that we already know that four particular genes (Fn1, Timd2, Gata4 and Sox7) are relevant to the task

Data/example source: Modern Statistics for Modern Biology by Holmes and Huber

Generalizing LDA to more than 2 classes



Logistic regression

- Logistic regression can be used as a binary (two-class) classifier
- Logistic regression vs two-class LDA
    - Both logistic regression and LDA use MLE to estimate $p(C = k|\boldsymbol{x}_i)$
    - LDA assumes a parametric distribution for $p(\boldsymbol{x}_i|C = k)$; if assumption is reasonable, can be more powerful
    - Logistic more resistant to outliers, model mispecifications

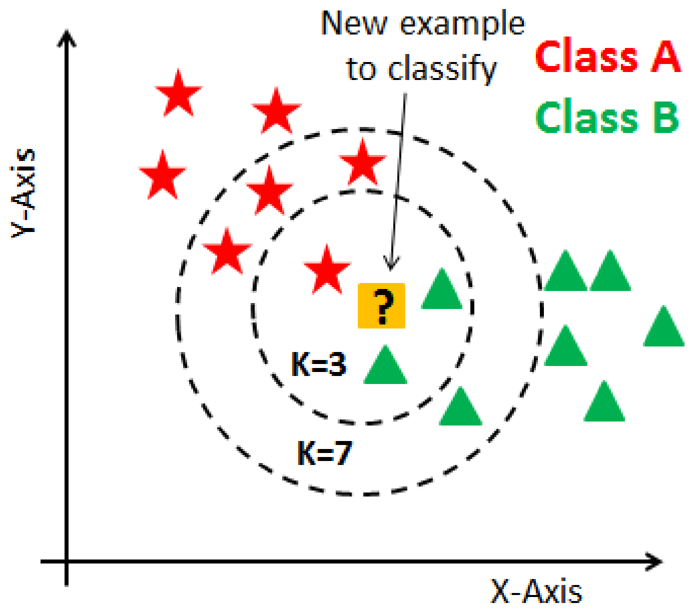
Non-parametric

Why should we bother thinking about a model for $p(C = k|\boldsymbol{x}_i)$ if our goal is just to partition the input space?

For example, if we want to assign a class to observation $\boldsymbol{x}_i$, why don't we just look at the class assigned to the point(s) **closest** to $\boldsymbol{x}_i$?

K-Nearest neighbour classifier

- One way to define "closest" is to use a fixed number of neighbours (K)
- Then count how many points of each class there are among these closest K neighbours to $\boldsymbol{x}_i$
- Use the majority class as the predicted class



Visualization of KNN classifier boundary

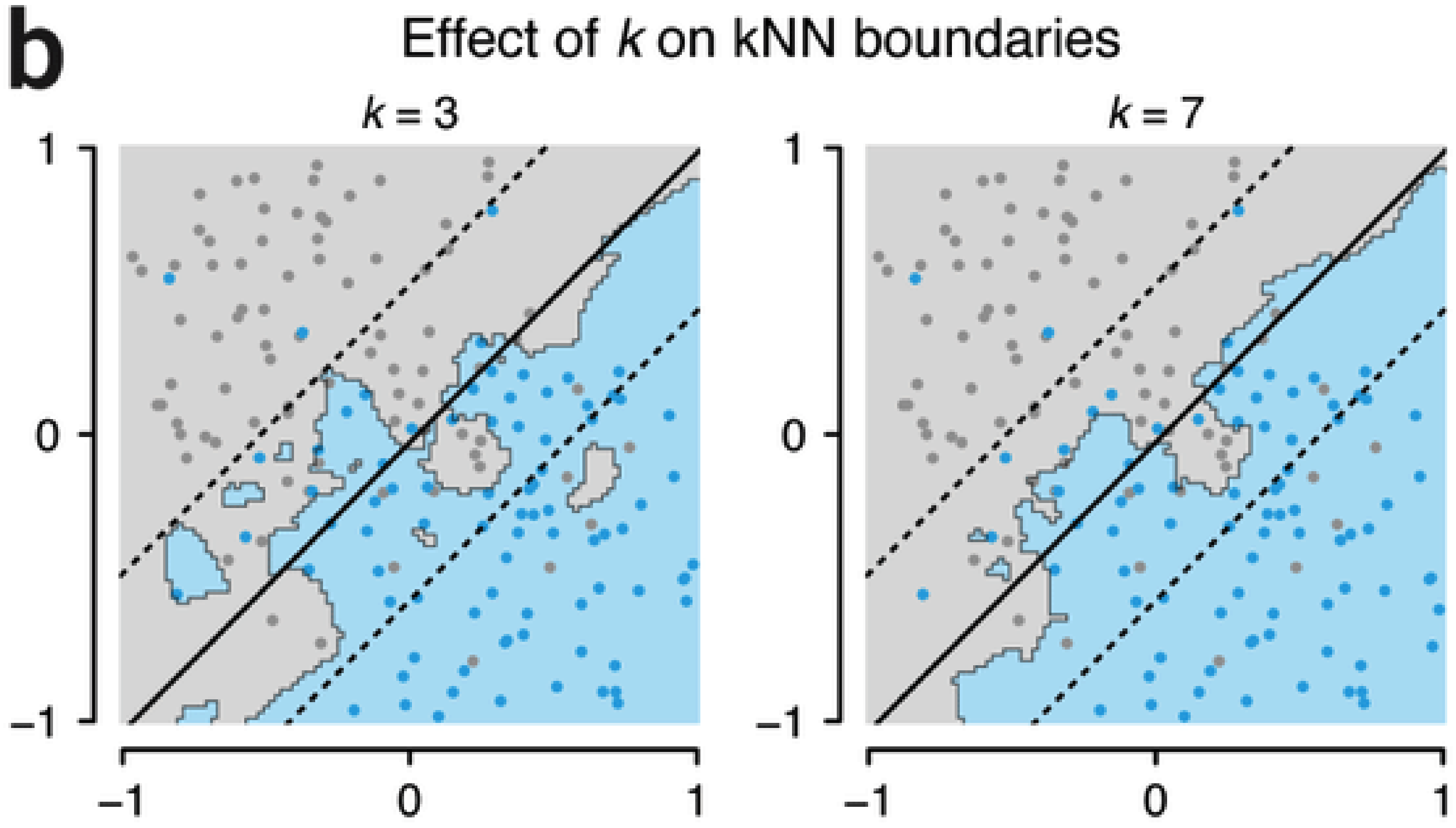


Fig 3b: Bzdok et al. (2018)

Additional Resources

- Conceptual overview: Chapter 12 of Modern Statistics for Modern Biology by Holmes and Huber
- More detailed overview + R implementation: Chapter 31 of Intro to Data Science by Irizarry
- Mathematical framework: Chapters 4 and 13 (plus other chapters that expand on methods we didn't cover: 12 and 15) in Elements of Statistical Learning by Hastie, Tibshirani and Friedman