STAT540
Lecture 17: March 11th 2016

Supervised Learning: Classification

Sara Mostafavi
Department of Statistics
Department of Medical Genetics
Center for Molecular Medicine and Therapeutics

** Many thanks to Drs. Gabriela Cohen-Freue and Kevin Murphy for lecture slides**

Biomarker studies

1. Identify molecular biomarkers of a disease
2. Based on the identified markers, create a molecular signature of the disease
3. Build a classifier to **predict** the outcome of samples

Goals of a supervised learning problem in Machine Learning

The New England
Journal of Medicine

VOLUME 347 DECEMBER 19, 2002 NUMBER 25

A GENE-EXPRESSION SIGNATURE AS A PREDICTOR OF SURVIVAL IN BREAST CANCER

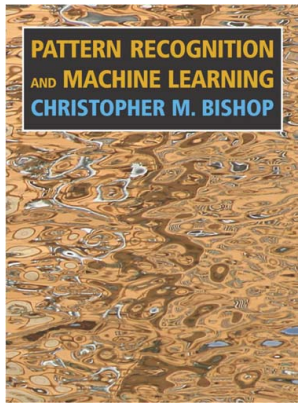
MARC J. VAN DE VIJVER, M.D., PH.D., YUDONG D. HE, PH.D., LAURA J. VAN 'T VEER, PH.D., HONGYUE DAI, PH.D., AUGUSTINUS A.M. HART, M.SC., DORIEN W. VOSKUIL, PH.D., GEORGE J. SCHREIBER, M.SC., JOHANNES L. PETERSE, M.D., CHRIS ROBERTS, PH.D., MATTHEW J. MARTON, PH.D., MARK PARRISH, DOUWE ATSMA, ANKE WITTEVEEN, ANNUSKA GLAS, PH.D., LEONIE DELAHAYE, TONY VAN DER VELDE, HARRY BARTELINK, M.D., PH.D., SJOERD RODENHUIS, M.D., PH.D., EMIEL T. RUTGERS, M.D., PH.D., STEPHEN H. FRIEND, M.D., PH.D., AND REN BERNARDS, PH.D.

Machine learning

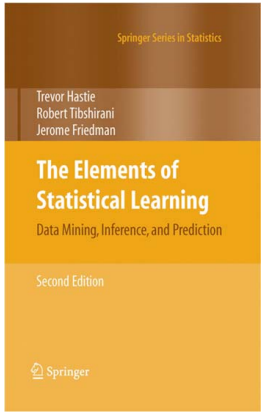
- Unsupervised learning:
  - Clustering algorithms
- Supervised learning:
  - Classification (categorical outcome)
  - Regression (continuous outcome)

Reading recommendation:

1) Pattern recognition & Machine Learning
by Christopher Bishop



2) The Elements of Statistical Learning
by Hastie, Tibshirani, and Friedman



How do we predict disease/severity based on expression data?

1. Gather some gene expression data relevant for outcome you would like to predict (e.g., disease status) → training data.

2. Build/train (i.e., write down) a model that relates the relevant features/attributes/genes to the outcome.
   - Fit model parameters based on data to fully specify the model.

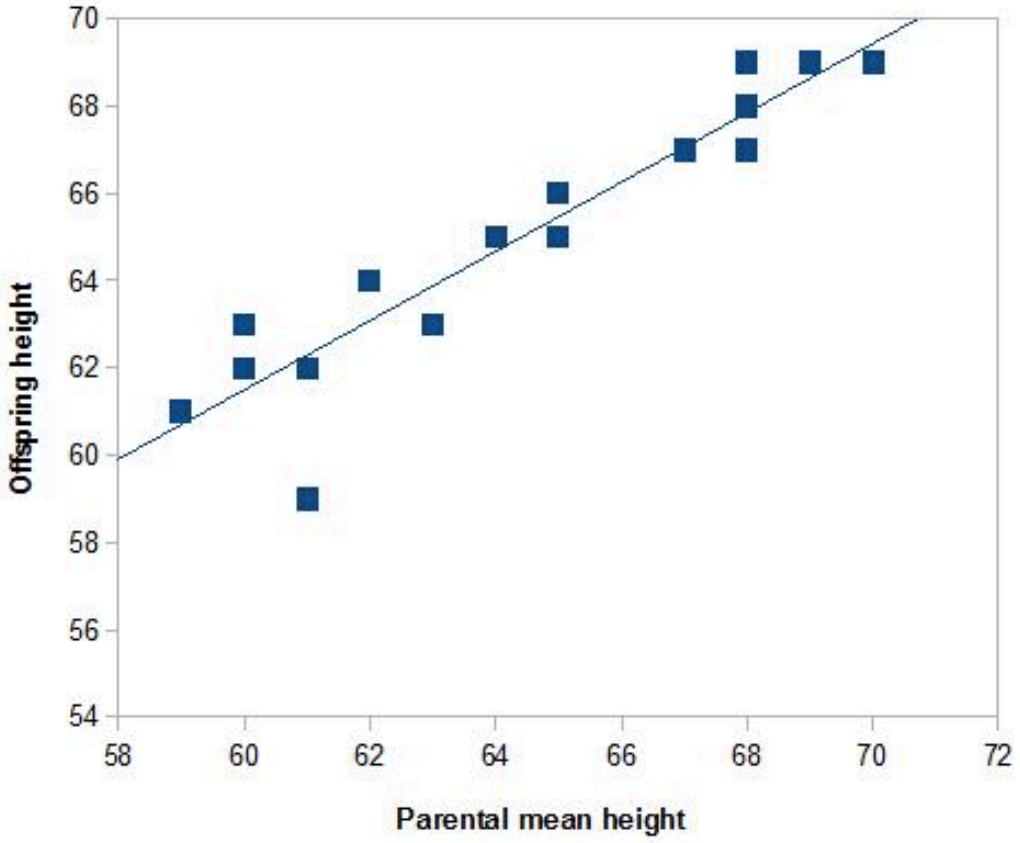
3. Apply the model to new data, where you don't have information about outcome (response) to make a prediction.

Supervised learning:
“toy” example

How would you predict the future height of a child (i.e., predict the eventual height of a child)?

How would you predict the future height of a child?

Use training data to estimate the slope of the line



1. Gather some training data:

Feature vector

Training data: $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$

Response/label/outcome

$$\vec{x}_i \subseteq \Re^p$$

$$y \subseteq \Re$$

2. Write down a model for relating the input data to response:

Feature vector

Training data: $\{(\vec{x}_1, y_1), (\vec{x}_2, y_2), \ldots, (\vec{x}_n, y_n)\}$

Response/label/outcome

Model: $y_i = f(\vec{x}_i; \theta) = \alpha^T \vec{x}_i + b = \sum_{j=1}^{p} \alpha_j x_{i,j} + b$

$\theta = \{\alpha_1, \ldots, \alpha_p, b\}$

Model parameter

2b. Fit (i.e., solve for) the model parameter using training data:

Model: $y = f(\vec{x};\theta) = \alpha^T \vec{x}$

Model parameter

Minimize the error on training data!

Objective function: $\underset{\alpha,b}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - (b + \alpha^T \vec{x}_i))^2$

Squared error

How do we fit the model parameters?

Differentiate the objective function wrt to *a*

Objective function:

$$\underset{\alpha}{\operatorname{argmin}} \sum_{i=1}(y_i - \alpha x_i)^2$$

Differentiate wrt *a:*

$$\frac{\partial f}{\partial \alpha} = \frac{1}{2}\sum_{i=1}^{n} x_i(y_i - \alpha x_i)$$

Set derivative to 0:

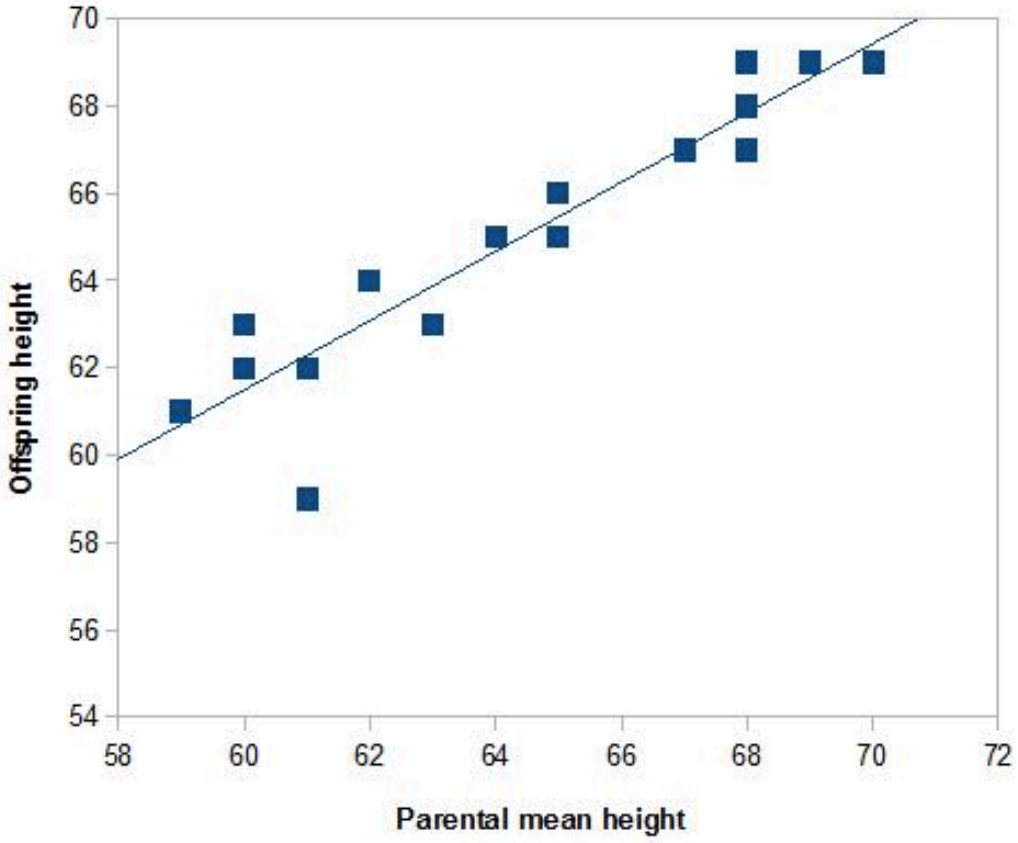
$$\frac{1}{2}\sum_{i=1}^{n} x_i(y_i - \alpha x_i) = 0$$

Solve for *a:*

$$\sum_{i=1}^{n} x_i y_i = \alpha \sum_{i=1}^{n} x_i^2$$

$$\alpha = \frac{\sum_{i=1}^{n} x_i^2}{\sum_{i=1}^{n} x_i y_i}$$

$$y_i = \alpha x_i + b$$


70
68
66
64
62
60
58
56
54
Offspring height
58
60
62
64
66
68
70
72
Parental mean height

How do you predict response/labels/class based on data?

1. Gather some data relevant for outcome you would like to predict (e.g., disease status) → training data.

2. Build/train (i.e., write down) a model that relates the relevant features/attributes to the outcome.
   - Fit model parameters based on data to fully specify the model.

3. Apply the model to new data, where you don't have information about outcome (response) to make a prediction.

Shift in focus compared to the application of "traditional statistics"

How we "viewed" the data before:

Gene expression levels = function ("design matrix")

y (response/outcome)

Big shift in emphasis!

outcome = function (" gene expression")

X

Supervised learning

- Regression
  - Continuous outcome
- Classification
  - Binary outcome
  - Categorical outcome

The classification problem setup

- The response represents class labels: it takes on an integer from the set $\{c_1,.., c_k\}$ when there are *k* classes.
- Example: benign tumor = class c_1 ; malignant tumor = class c_2

Feature vector

Input data: $\{(\vec{x}_1, c_1), (\vec{x}_2, c_2), ..., (\vec{x}_n, c_n)\}$

Class/labels

$$\vec{x}_i \subseteq \Re^p$$

$$c_i \subseteq \{c_1, .., c_k\}$$

The classification problem setup

- Given feature vectors for n objects (items), represented by matrix X, the goal is to predict which class each object belongs to.

- Example: give some gene expression data from a tumor, predict if the tumor is benign or malignant.

Classifiers: definitions

- A *classifier* is a function *f* that maps input feature vectors $\vec{x}_i$ to output class label $y_i \in \{c_1, .., c_k\}$ (we assume that class labels are unordered, and mutually exclusive)

- χ is the feature space consisting of $\vec{x}_i$'s , the feature space could consist of continuous, discrete values or a mixture of the two (i.e., $\chi = \{0,1\}^p$ or $\chi = \Re^p$)

- Goal: to learn a function *f* that maps feature vectors to labels, based on a labeled training set: $\{(\vec{x}_1, y_1), \ldots, (\vec{x}_n, y_n)\}$

  $f(\vec{x}_1) = y_1$

A loss function for making predictions

- To train a classifier and predict future outcomes, we minimize a loss function that quantifies the cost of misclassification.

- Simple rule: all errors are equally bad

$$L(a,b) = \begin{cases} 0 & \text{if } a = b \\ \\ 1 & \text{otherwise} \end{cases}$$

- In general, we predict the class that minimizes the *conditional expected* loss

$$\sum_{j=1}^{K} L(c_j, \hat{c}(\boldsymbol{X}))P(C = c_j|\boldsymbol{X})$$

- Simple case K=2, so let c_0=0 and c_1=1 (e.g, benign and malignant tumors)
- The expected loss for prediction 0 when it was a 1 is:

$$L(1,0)P(c_1|\boldsymbol{X})$$

- The expected loss for predicting 1 when it was a 0 is:

$$L(0,1)P(c_0|\boldsymbol{X})$$

- Predict class 1 if:

$$\frac{L(1,0)P(c_1|\boldsymbol{X})}{L(0,1)P(c_0|\boldsymbol{X})} > 1$$

Do we know $P(C = c_j \mid X)$?

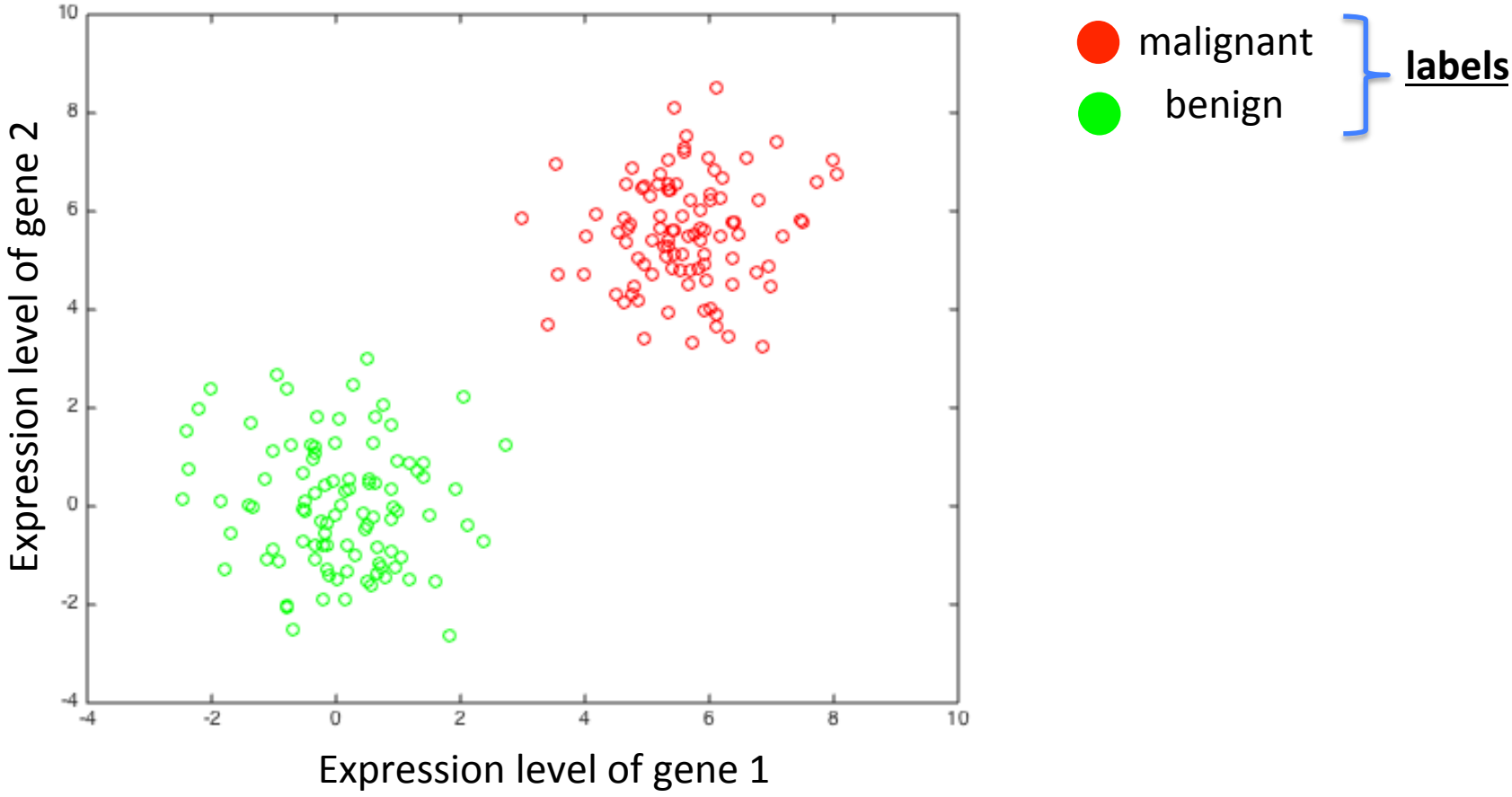
Three main approaches for estimating $P(C = c_j \mid X)$

- Learn a "generative" model (function) for the probability distribution for each class: $p(X|C=c_j)$ *(e.g.,* $f(X|C=c_j)$). Then at prediction time use Bayes rule and $p(c_j)$ to compute $p(c_j|X)$.

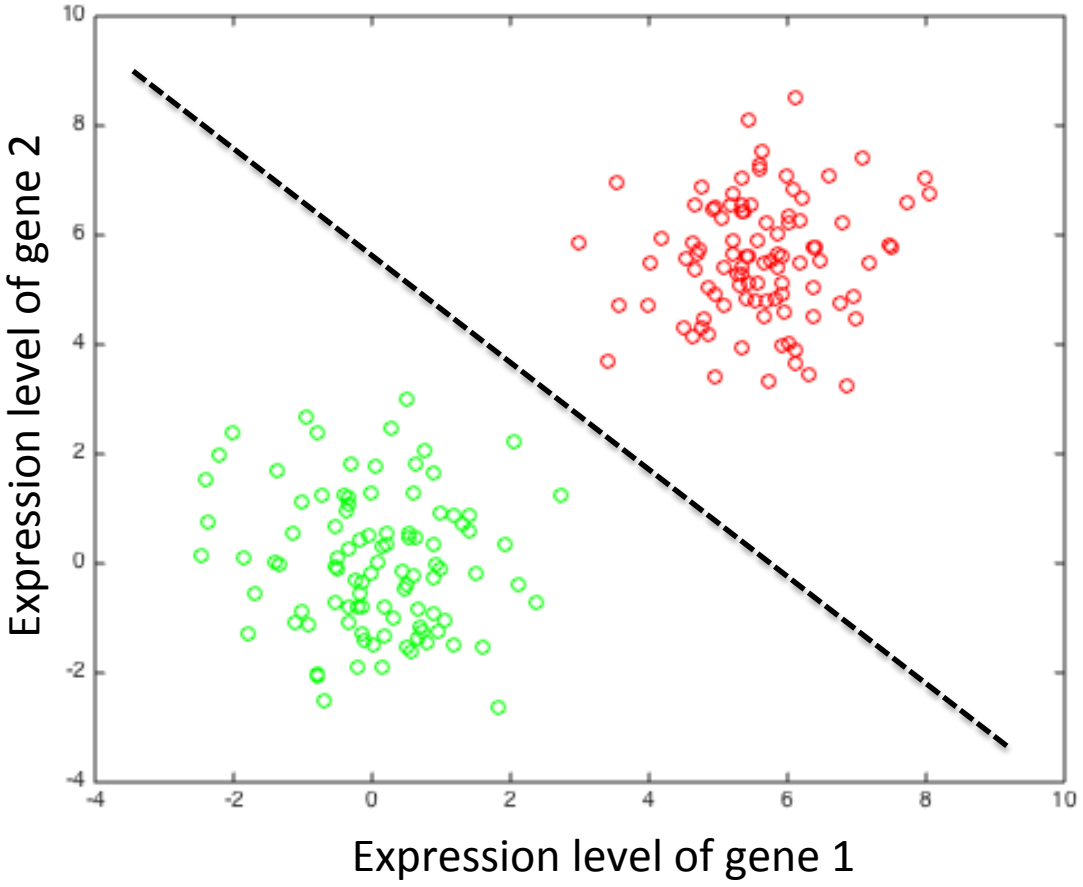
- Learn a "discriminative" for conditional probability distribution of each class: $p(C=c_j|X)$

- Non-parametric: e.g., learn a function that directly maps X_i to it's predicted class c_j.

- Discriminant classifier (e.g., logistic regression)
- Search for a line that best separates examples from the two classes



- Discriminant classifier (e.g., logistic regression)
- Search for a line that best separates examples from the two classes

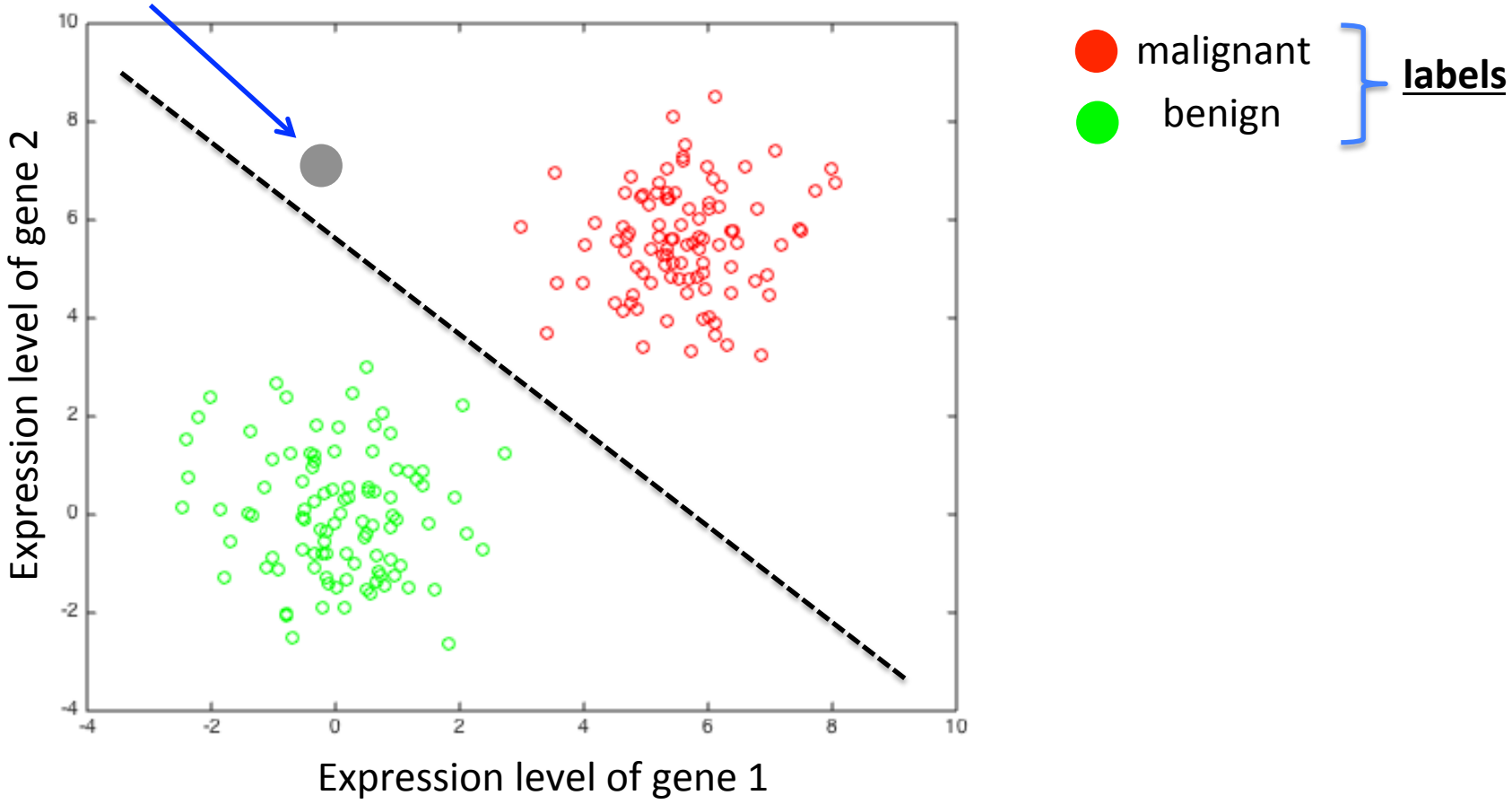


malignant
benign

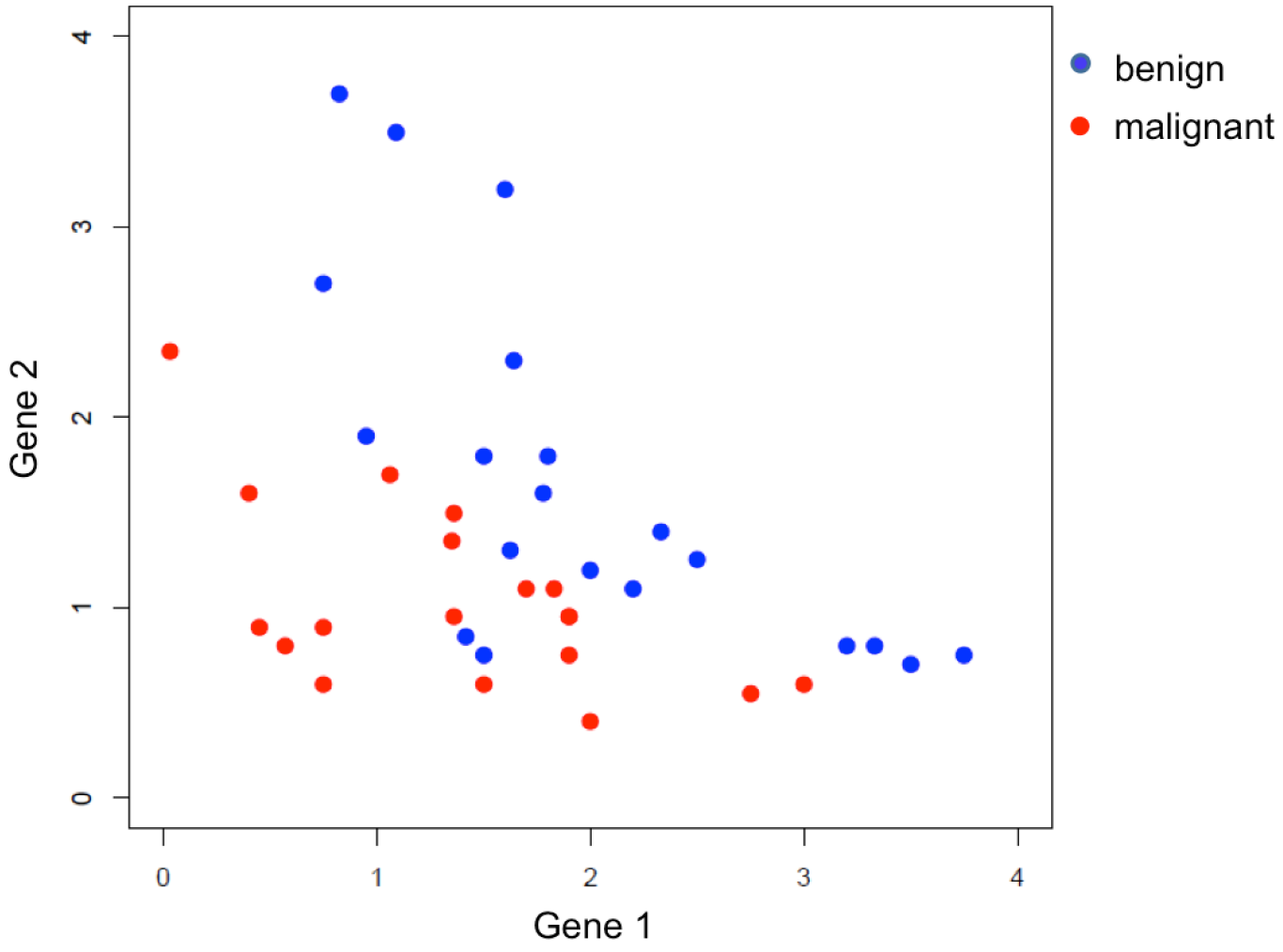
labels

- Discriminant classifier (e.g., logistic regression)
- Search for a line that best separates examples from the two classes

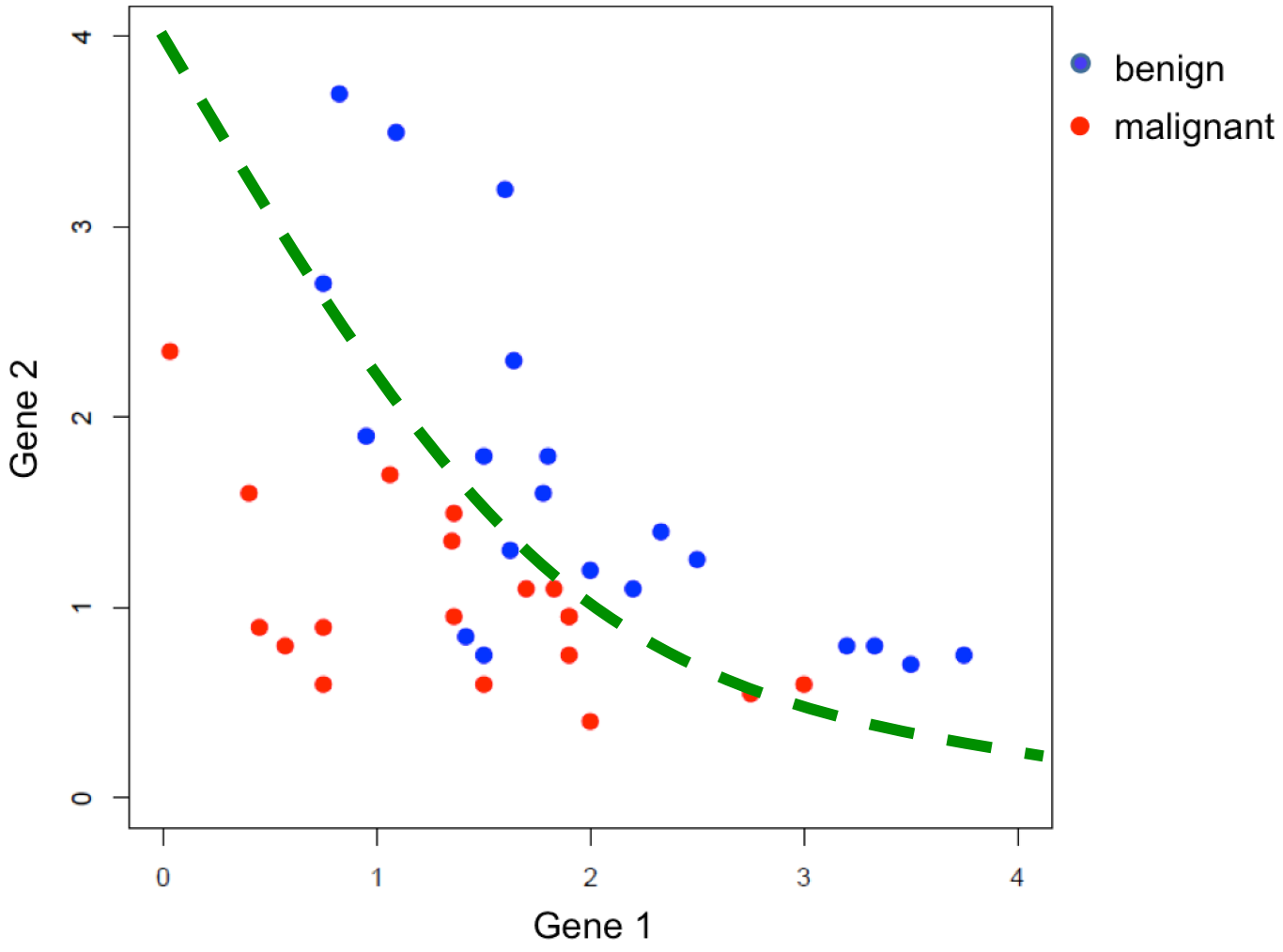
New example: what's your prediction?



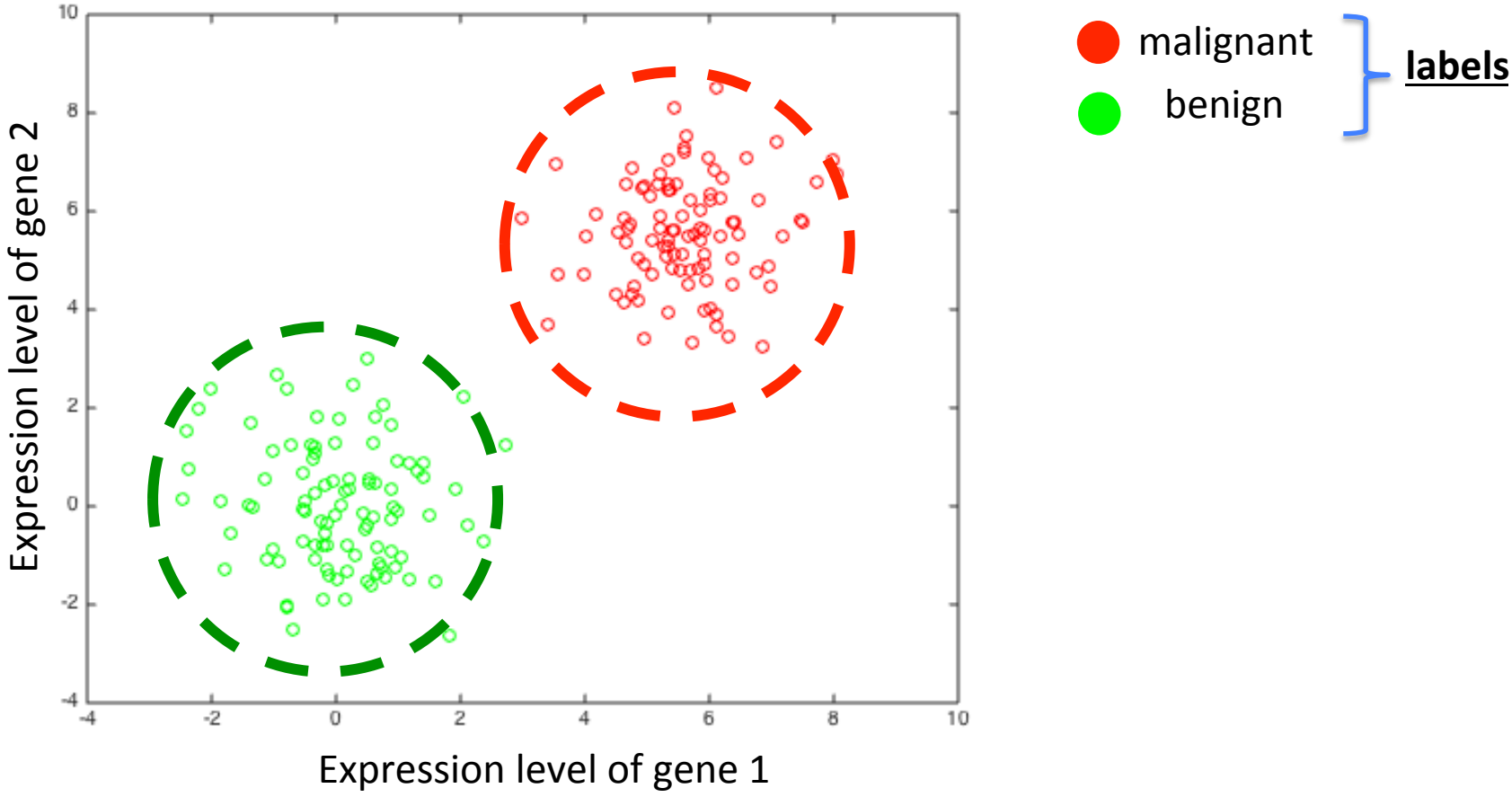
Classification task: partition the space of input data so that you are minimizing the number of “miss-classified” objects/points.



Classification task: partition the space of input data so that you are minimizing the number of “miss-classified” objects/points.



Generative classifier (e.g., Naïve Bayes)



The generative model for solving the classification problem

- Generative: learn class-conditional density p(x|c) for each value of c, and learn the class priors p(c): then one can apply the Bayes rule to compute most likely class for each object/entity.

$$p(\vec{x}_i \mid c_i = k) \qquad p(c_i = k)$$

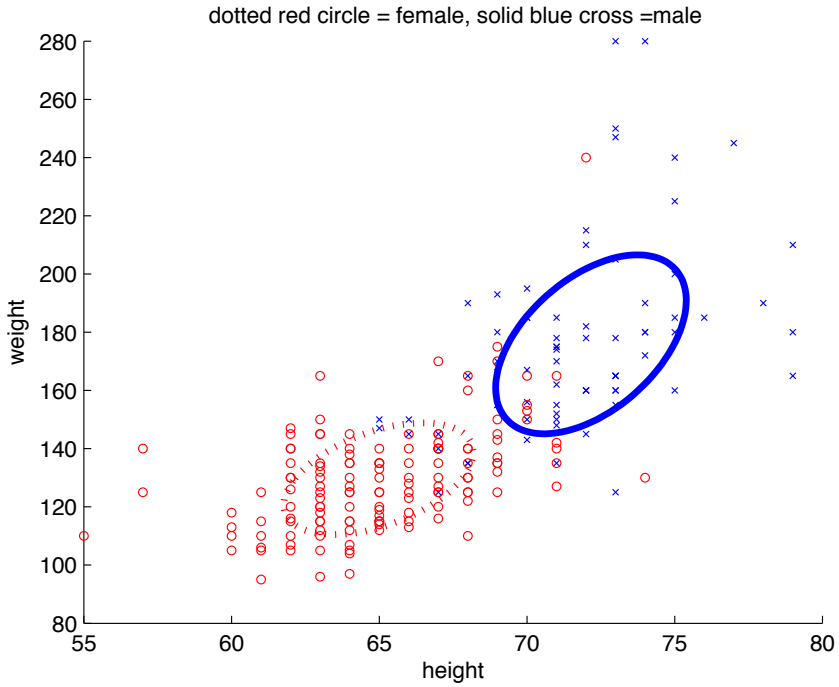
The generative model for solving the classification problem

- Generative: learn class-conditional density p(x|c) for each value of c, and learn the class priors p(c): then one can apply the Bayes rule to compute most likely class for each object/entity.

$$p(c_i = k \mid \vec{x}_i) = \frac{p(\vec{x}_i \mid c_i = k)p(c_i = k)}{p(\vec{x}_i)}$$

Working example

- Suppose $\vec{x} \in \Re^2$ the 2-d feature vector for each object represent the *height* and *weight* of an adult (in inches and pounds, respectively), and $y \in \{0,1\}$ represents male or female.



Naïve Bayes Model

- Assumes features are independent:

$$p(\vec{x}_i \mid c = k) = \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k)$$

Naïve Bayes: assumptions

- Assumes features are independent:

$$p(\vec{x}_i \mid c = k) = \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k)$$

$$= \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k, \theta_k)$$

Naïve Bayes: assumptions

- Assumes features are independent:

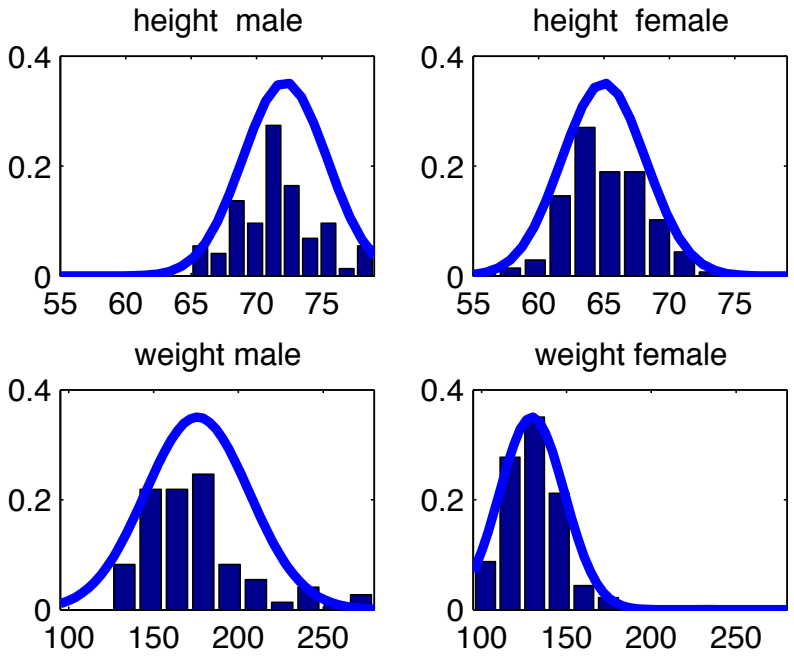
$$p(\vec{x}_i \mid c = k) = \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k)$$

$$= \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k, \theta_k)$$

$$= \prod_{j=1}^{p} N(x_j^{(i)} \mid \mu_{k,d}, \sigma_{k,d})$$

The univariate Gaussian Naïve Bayes model

$$p(\vec{x}_i \mid c = k) = \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k) = \prod_{j=1}^{p} N(x_j^{(i)} \mid \mu_{k,d}, \sigma_{k,d})$$

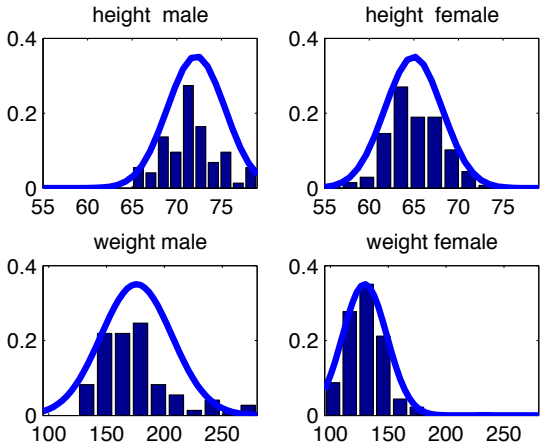


** Note we are assuming that the covariance matrices Σ_c and Σ_x diagonal **

The univariate Gaussian Naïve Bayes model: training time

- We can easily estimate all parameters using MLE (based on training data).

h	w	c
67	125	m
79	210	m
71	150	m
68	140	f
67	142	f
60	120	f
⋮	⋮	⋮



	h	w
m	$\mu = 71.66,\ \sigma = 3.13$	$\mu = 175.62,\ \sigma = 32.40$
f	$\mu = 65.07,\ \sigma = 3.19$	$\mu = 129.69,\ \sigma = 18.67$

The univariate Gaussian Naïve Bayes model: training time

- We can easily estimate all parameters using MLE (based on training data).

	h	w
m	$\mu = 71.66,\ \sigma = 3.13$	$\mu = 175.62,\ \sigma = 32.40$
f	$\mu = 65.07,\ \sigma = 3.19$	$\mu = 129.69,\ \sigma = 18.67$

$$p(\vec{x}_i \mid c = k) = \prod_{j=1}^{p} p(x_j^{(i)} \mid c = k) = \prod_{j=1}^{p} N(x_j^{(i)} \mid \mu_{k,d}, \sigma_{k,d})$$

$$= N(x_1^{(i)} \mid 71.66, 3.13) \times N(x_2^{(i)} \mid 175.62, 32.40)$$

The Gaussian Naïve Bayes model: testing time

Goal: to compute the *posteriori* probability of class assignment for some unknown example x_t

$$p(c = k \mid \vec{x}_t) = \frac{p(\vec{x}_t \mid c = k)\, p(c = k)}{p(\vec{x}_t)}$$

The Gaussian Naïve Bayes model: testing time

Goal: to compute the *posteriori* probability of class assignment for some unknown example x_j

$$p(c = k \mid \vec{x}_t) \propto p(\vec{x}_t \mid c = k)\, p(c = k)$$

$$= p(c = k) \prod_{j=1}^{p} N(x_j^{(t)} \mid \mu_{k,j}, \sigma_{k,j})$$

The Gaussian Naïve Bayes model: testing time

Goal: to compute the *posteriori* probability of class assignment for some unknown example x_j

$$p(c = k \mid \vec{x}_t) = p(c = k) \times N(x_1^{(t)} \mid \mu_{c,1}, \sigma_{c,1}) \times N(x_2^{(t)} \mid \mu_{c,2}, \sigma_{c,2})$$

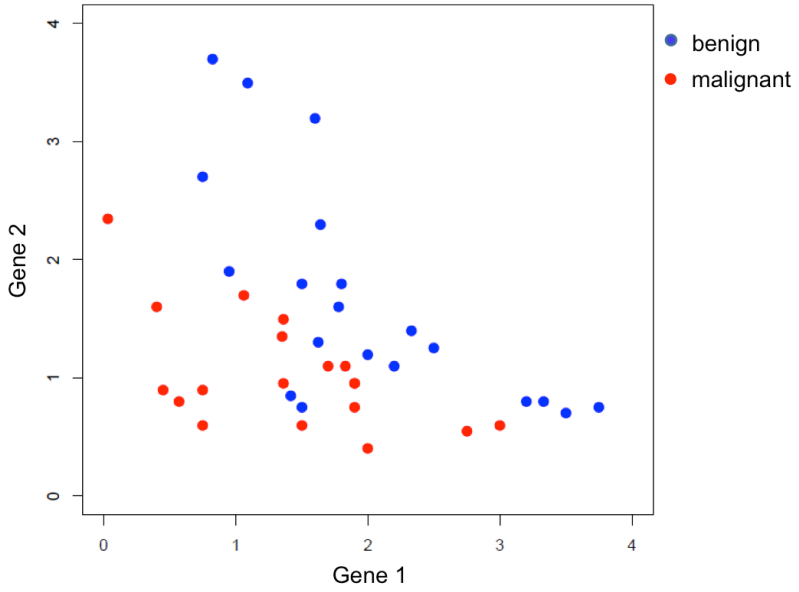
Recall that we computed all model parameters using MLE

	h	w
m	$\mu = 71.66,\ \sigma = 3.13$	$\mu = 175.62,\ \sigma = 32.40$
f	$\mu = 65.07,\ \sigma = 3.19$	$\mu = 129.69,\ \sigma = 18.67$

Linear Discriminant Analysis

(by Fisher in 1936)

- Assume that gene 1 and gene 2 are normally distributed in each class

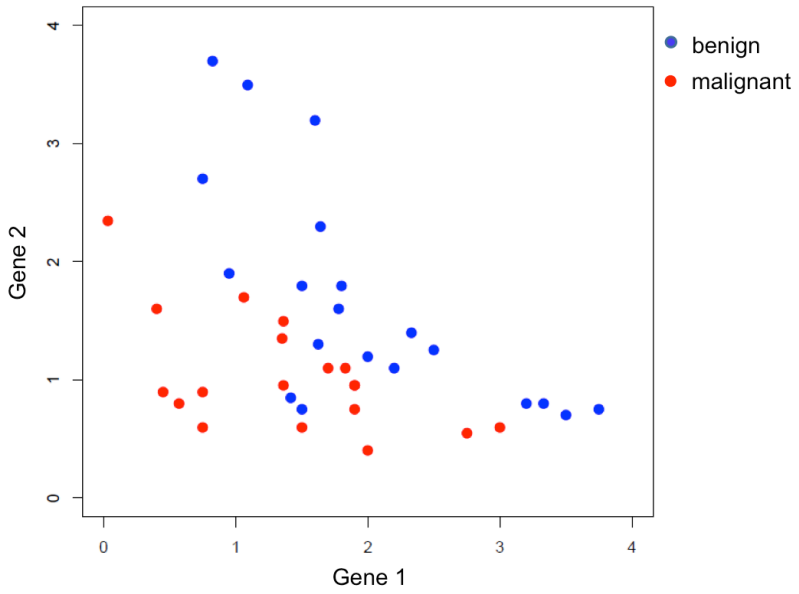


Linear Discriminant Analysis

(by Fisher in 1936)

- Assume that gene 1 and gene 2 are normally distributed in each class

$$p(\vec{x}_i \mid c_k) \sim N(\vec{x}_i \mid \mu_k, \Sigma_k)$$



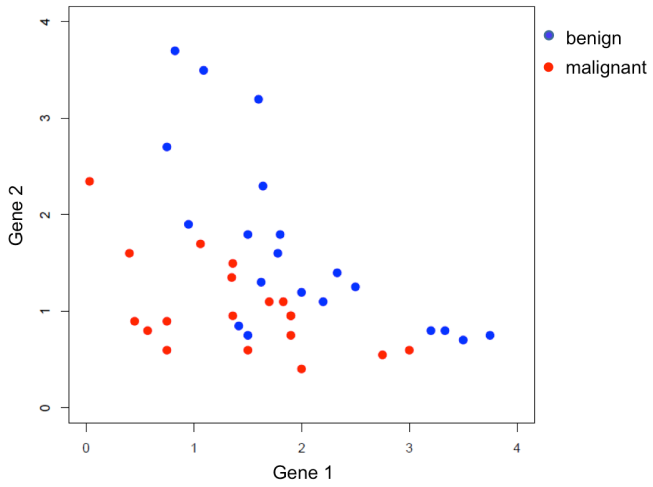
Linear Disciminant Analysis

(by Fisher in 1936)

- Assume that gene 1 and gene 2 are normally distributed in each class

$$p(c=0 \mid \vec{x}_i) \sim N(\vec{x}_i \mid \mu_0, \Sigma\ ) \frac{p(c=0)}{p(\vec{x}_i)}$$

$$p(c=1 \mid \vec{x}_i) \sim N(\vec{x}_i \mid \mu_1, \Sigma) \frac{p(c=1)}{p(\vec{x}_i)}$$



Linear Disciminant Analysis

(by Fisher in 1936)

- Assume that gene 1 and gene 2 are normally distributed in each class

$$p(c=0 \mid \vec{x}_i) \sim N(\vec{x}_i \mid \mu_0, \Sigma\ ) \frac{p(c=0)}{p(\vec{x}_i)}$$

$$p(c=1 \mid \vec{x}_i) \sim N(\vec{x}_i \mid \mu_1, \Sigma\ ) \frac{p(c=1)}{p(\vec{x}_i)}$$

$$\frac{p(c=0 \mid \vec{x}_i)}{p(c=1 \mid \vec{x}_i)} > 1 \Leftrightarrow \log\left(\frac{p(c=0 \mid \vec{x}_i)}{p(c=1 \mid \vec{x}_i)}\right) > 0$$

Linear Disciminant Analysis

(by Fisher in 1936)

- Assume that gene 1 and gene 2 are normally distributed in each class

$$p(c=0 \mid \vec{x}_i) \sim N(\vec{x}_i \mid \mu_0, \Sigma\ ) \frac{p(c=0)}{p(\vec{x}_i)}$$

$$p(c=1 \mid \vec{x}_i) \sim N(\vec{x}_i \mid \mu_1, \Sigma\ ) \frac{p(c=1)}{p(\vec{x}_i)}$$

$$\frac{p(c=0 \mid \vec{x}_i)}{p(c=1 \mid \vec{x}_i)} > 1 \Leftrightarrow \log\left(\frac{p(c=0 \mid \vec{x}_i)}{p(c=1 \mid \vec{x}_i)}\right) > 0 \Leftrightarrow \log\left(\frac{N(\vec{x}_i \mid \mu_0, \Sigma)}{N(\vec{x}_i \mid \mu_1, \Sigma)} \frac{p(c=0)}{p(c=1)}\right) > 0$$

Linear Discriminant Analysis

$$\log\left(\frac{N(\vec{x}_i \mid \mu_0, \Sigma)p(c=0)}{N(\vec{x}_i \mid \mu_1, \Sigma)p(c=1)}\right) > 0 \quad \Leftrightarrow \vec{a}^T \vec{x}_i + b > 0$$

where

$$\vec{a} = \Sigma^{-1}(\mu_0 - \mu_1)$$

$$b = (\mu_0 - \mu_1)^T \Sigma^{-1}(\mu_0 - \mu_1) - \log\left(\frac{p(c=0)}{p(c=1)}\right)$$

- The optimal LDA, classifies a point $\vec{x}_i$ in class 1 ("malignant") if:

$$\Leftrightarrow \vec{a}^T \vec{x}_i + b > 0$$

- We need to estimate $\vec{a}$ and *b*, but this is very easy now:

$$\vec{a} = \Sigma^{-1}(\mu_0 - \mu_1)$$

$$b = (\mu_0 - \mu_1)^T \Sigma^{-1}(\mu_0 - \mu_1) - \log(\frac{p(c=0)}{p(c=1)})$$

- Based on the data:

$$\vec{a} = (-2.77, -2.37) \quad b = 7.72$$

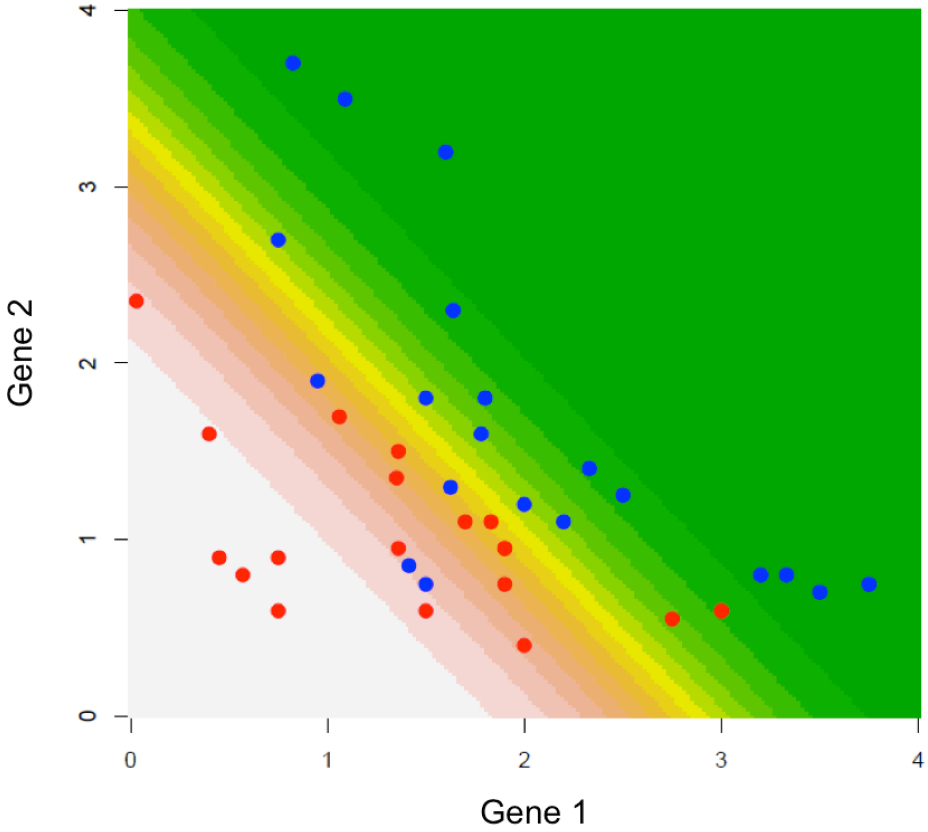
- The optimal classifier classifies a point x in class 0 if

$$-2.77\text{gene1} - 2.37\text{gene2} + 7.72 > 0$$

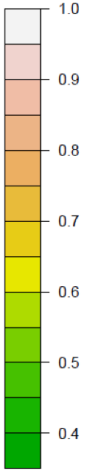
- And

$$\hat{P}(c = 1 | \text{gene1, gene2}) = \frac{\exp(-2.77\text{gene1} - 2.37\text{gene2} + 7.72)}{1 + \exp(-2.77\text{gene1} - 2.37\text{gene2} + 7.72)}$$

Linear Discriminant Analysis (LDA)



Posterior for class 1



$$\vec{a}^T \vec{x}_i + b > 0$$

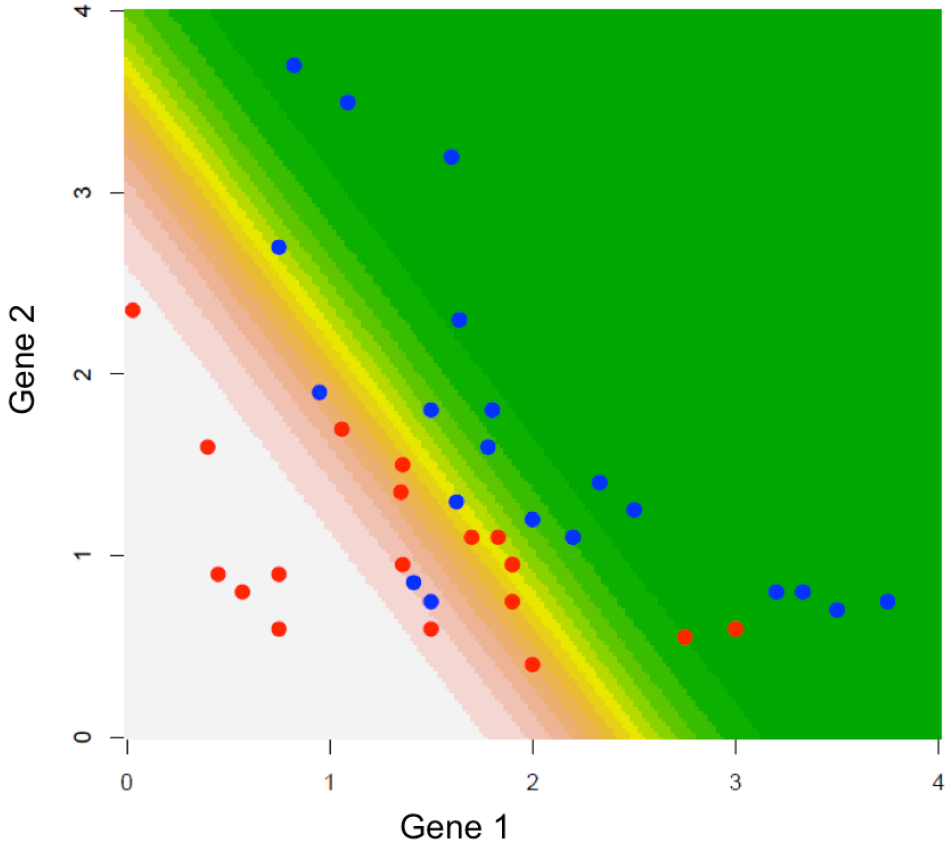
Supervised learning: the discriminant model

$$\log\left(\frac{p(c=0\mid\vec{x}_i)}{p(c=1\mid\vec{x}_i)}\right)=\log\left(\frac{p(c=0\mid\vec{x}_i)}{1=p(c=0\mid\vec{x}_i)}\right)=\vec{a}^T\vec{x}_i+b$$

We could simply start with this formula, and estimate ***a*** and *b* from the training data through maximum likelihood (→ **logistic regression**)

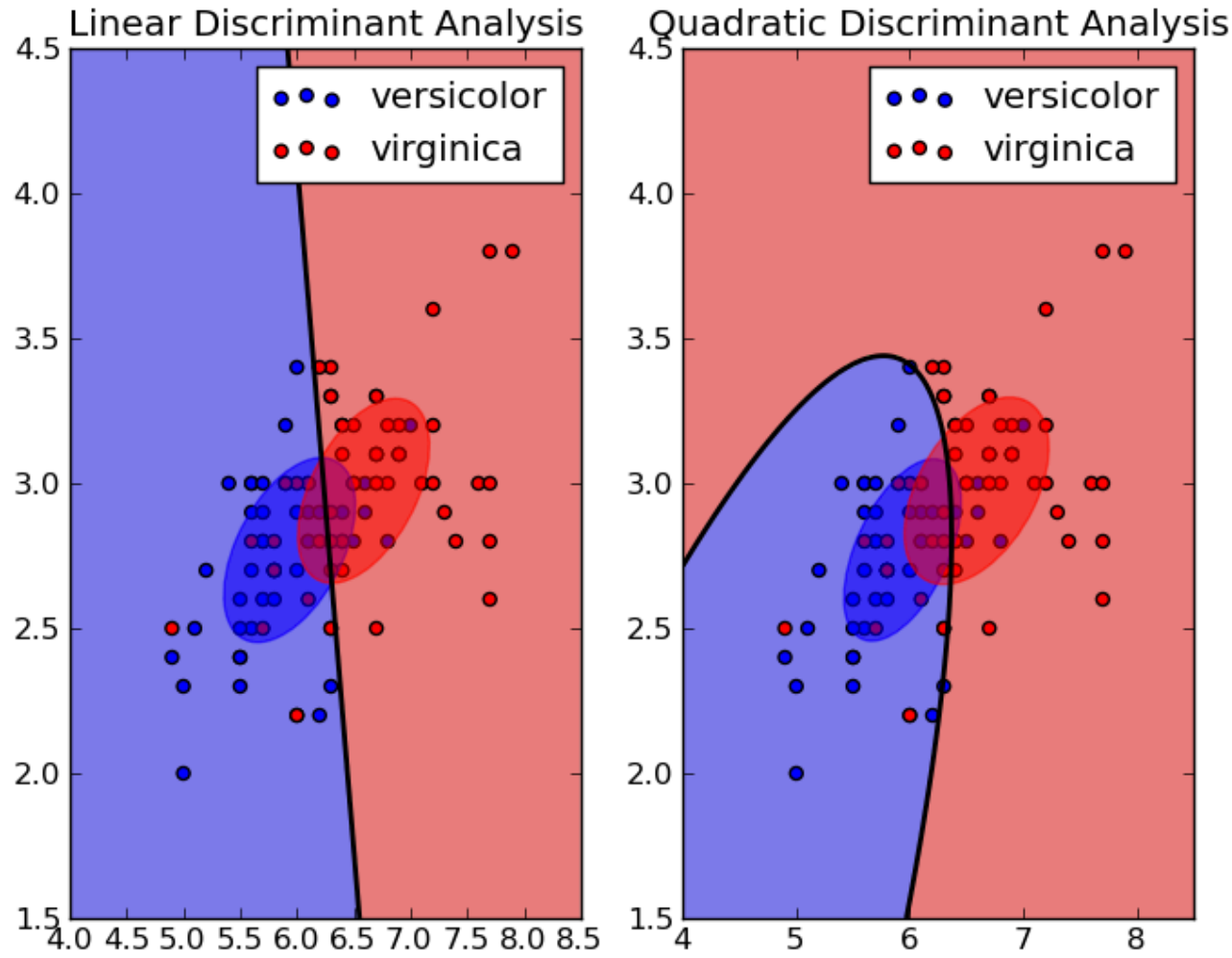
$$\vec{a}=(-3.88,2.65),\quad b=9.53$$

Classification with Logistic Regression



LDA vs. logistic regression

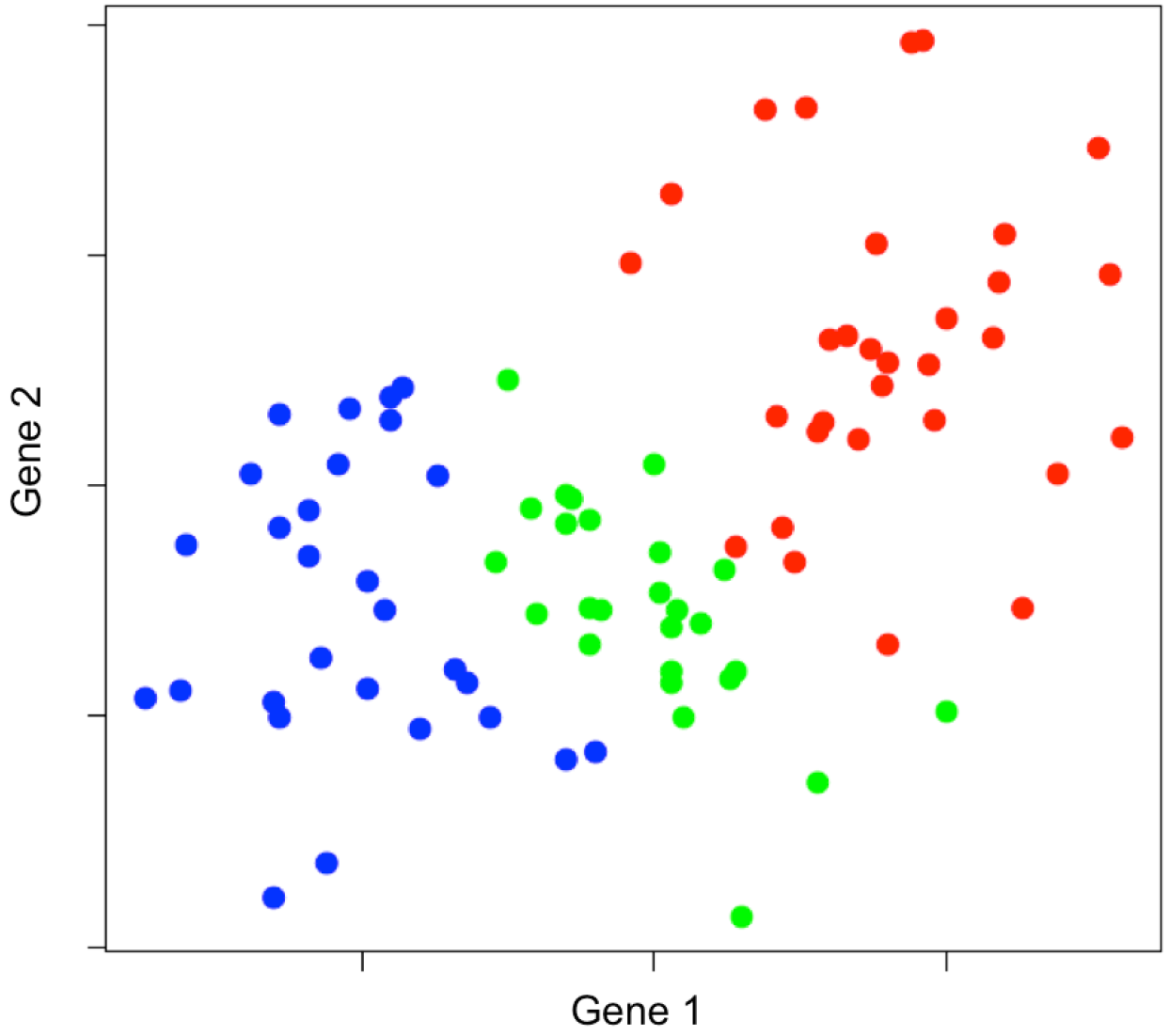
- Both use MLE estimates for the parameters.
- Both estimate $P(c_j|\boldsymbol{X})$. However, LDA assumes a distribution for $f(\boldsymbol{X}|c_j)$. If the assumption is reasonable, then it uses more information to predict.
- However, logistic is more resistant to outliers and model misspecifications



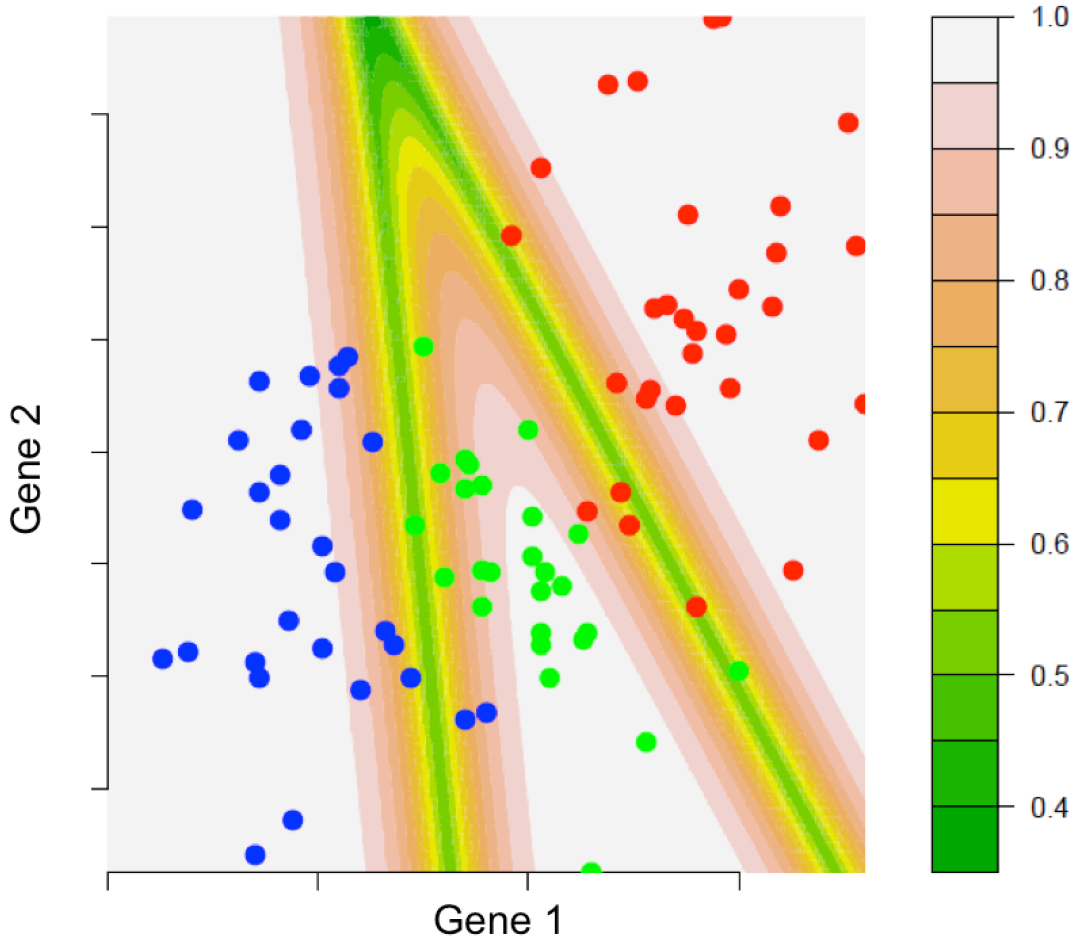
Linear discriminant analysis, Sigma is constrained to be diagonal. But in Quadratic discriminant analysis, you don't have the diagonal assumption (i.e., a more complex model of the data).

(Based on artificial data)

More than 2 groups



LDA for 3 groups



Colors correspond to maximum posterior

No model, Aka Non-parametric

- Why should we bother thinking about a model for $P(c_k|x_i)$ if our goal is *just* to partition the input space?

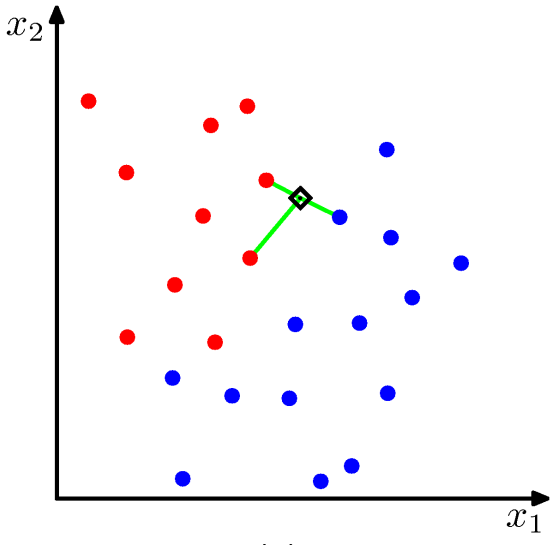
- *e.g.,* if we want to assign a class to x_t why don't we just look at the class assigned to the point closest to x_t?

k-nearest neighbor classifier

One way to define ”near” is to use a fixed predetermined number of neighbours (K)

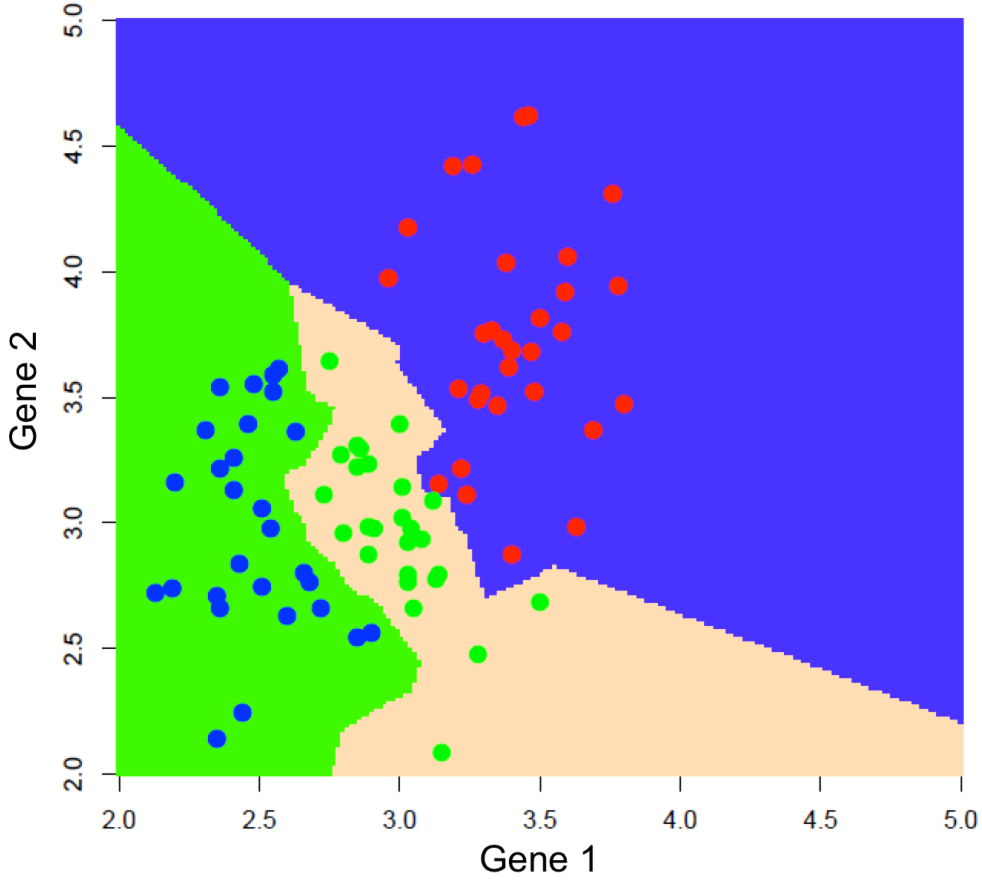
–> count how many points of each class there are among the closest K neighbours to $\boldsymbol{x}$

–> use the majority class as the predicted class

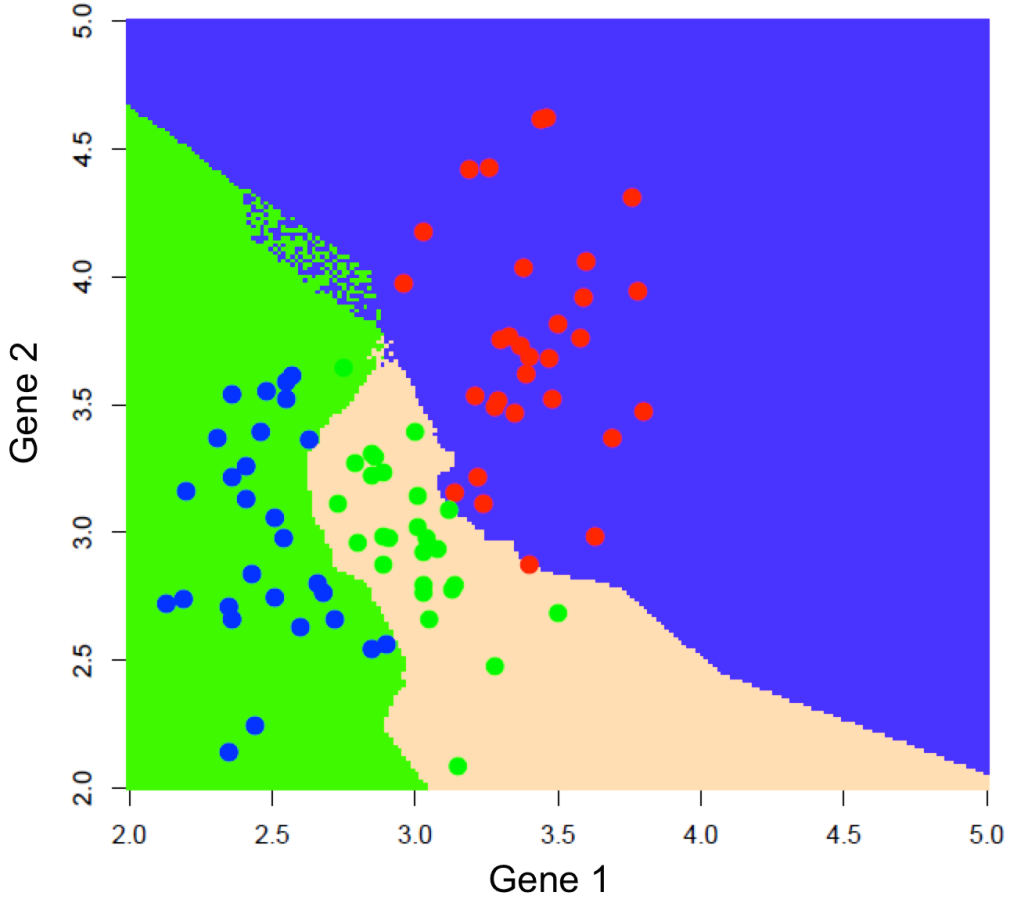


Bishop figure 2.27

1-NN



5-NN



Some example classification methods

Linear/Quadratic discriminant analysis
Naïve Bayes

Logistic regression
Support vector machines
Neural networks & “Deep Nets”
Decision trees

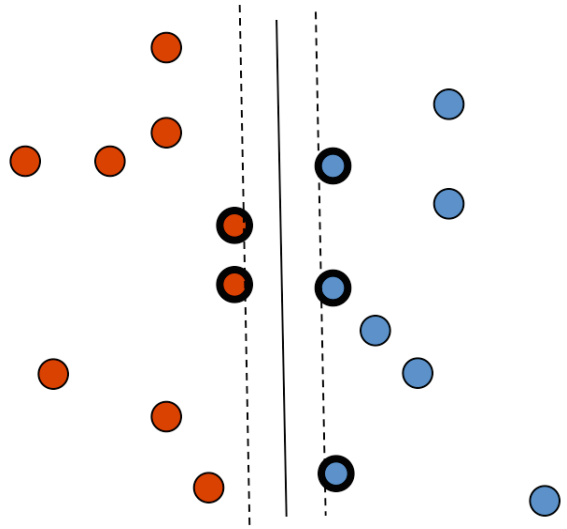
K-nearest neighbors

Support vector machines (discriminant approach)

- Support Vector Machine (SVM) will find **maximum margin separating hyperplane**.

In the solution, only some samples are support vectors, i.e., they support the decision boundary.

Intuitively: samples far from the decision boundary are ”easy” and don’t matter. Only samples near the boundary are used to define it.



Conclusions and summary

- There are many methods for performing classification.
- Which one is better? We need a criteria to evaluate them on data of interest (coming up next week)
- Some work better with smaller sample sizes; some work better with larger number of covariates/features

Analogy to biology

- Affymetrix chip
  - U133 chip
  - Exon array 1.0
- Illumina chip
- NanoString
- Illumina True-Seq