STAT540-Lecture 14: Classification

Dr. Gabriela Cohen Freue
Department of Statistics, UBC

February 27th, 2019

When biology speaks, we listen.

[...]Without understanding the genetics of a tumor, you can't treat it. Knowing this, we developed an advanced genomics platform for tumor gene expression profiling and built microarray assay tests that can determine whether an individual patient is at high or low risk for breast or colon cancers recurrence

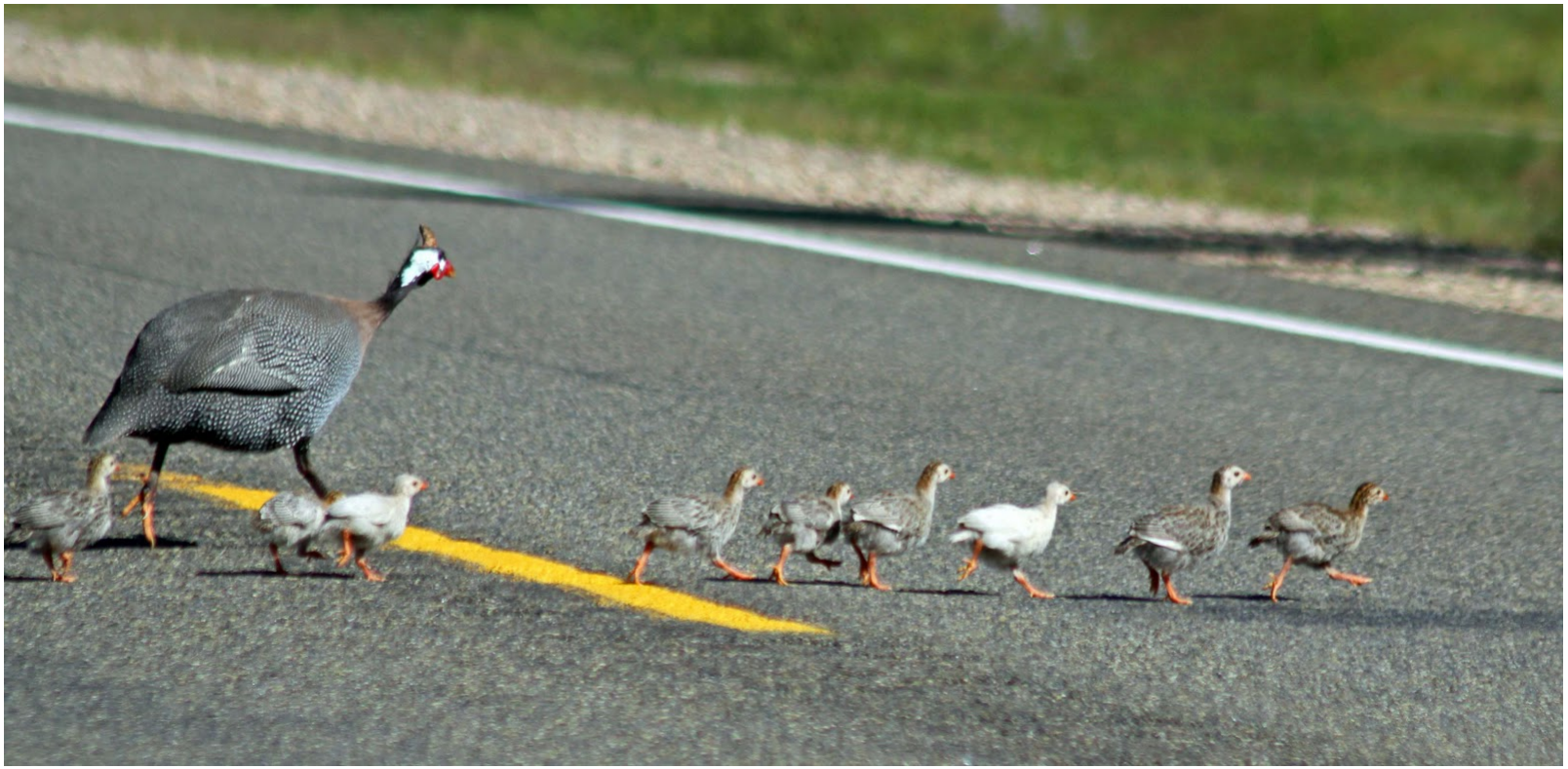
Biomarkers Studies

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

Goals of a supervised learning problem in Machine Learning

Machine Learning	Traditional Statistics
Prediction	Causality
Large number of variables (e.g., genes)	Handful number of variables
Complex non-linear relationships	Often assume linear relationships
Willing to use *ad hoc* methods	Rely on theoretical justifications
Heavy use of computing	Designed for hand-calculation

Where have we been so far in STAT540?



We have been cautiously crossing to the road!

Machine Learning: *Supervised vs. Unsupervised*

Supervised: prior knowledge of 'groups' or 'covariates' is used. For example, linear models:

- Build a linear model that best characterize and discriminate known classes (e.g., use a set of genes to distinguish benign from aggressive tumors)
- Classes in the data are known and used to estimate model parameters

Machine Learning: *Supervised vs. Unsupervised*

Unsupervised: information about groups in the data is not used. For example, cluster analysis:

- Identify cluster of samples that are similar to each other and distinct from those in other clusters. Only distances of attributes were used to cluster samples
- Clusters can be later interpreted based on additional information about the objects (e.g., cluster 3 contains most of the pre-natal samples)
- However, the information of the design was *not* used to form the clusters

A new goal... prediction

Have we learned the machinery we need?

Notation

Let y be the response (also denoted *target*) we want to predict:

- **Classification**: y takes values from a finite set of labels (e.g., $\{$benign, malignant$\}$)
- **Regression**: y is a real number (e.g., tumor size)

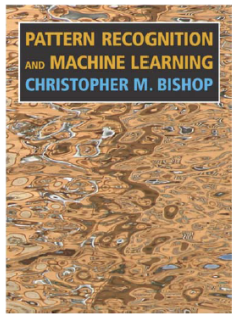
We want to predict y based on the values of p variables called covariates ($\boldsymbol{X} \in \mathbb{R}^p$), also known as explanatory variables, inputs, features, or predictors.

A "Supervised Learning Machine"

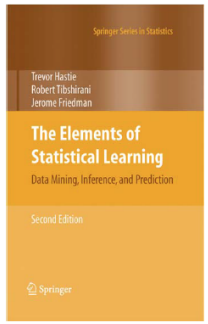
- We have a **training set** of n cases in which we know the values of both y and $\boldsymbol{X}$.
- Based on the training data, we will produce a model or a **rule** to predict y. These can be parametric or non-parametric.
- We want to predict y on new samples of a **test set** for which only $\boldsymbol{X}$ is known.

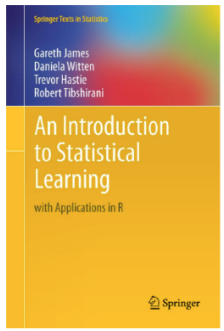
Reading recommendation:

1) Pattern Recognition & Machine Learning
by Christopher Bishop



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





https://statweb.stanford.edu/~tibs/ElemStatLearn/
http://www-bcf.usc.edu/~gareth/ISL/

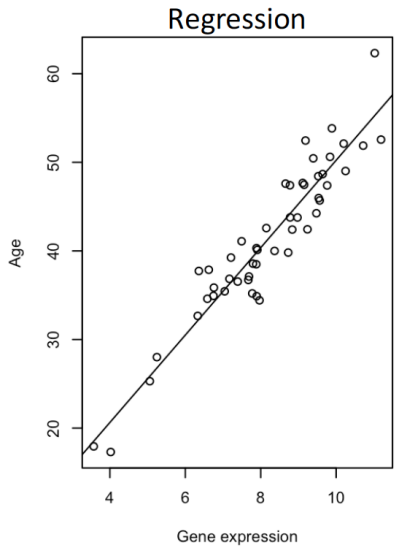
Stanford ONLINE Lagunita | **HumanitiesSciences:** StatLearning-SP Statistical Learning

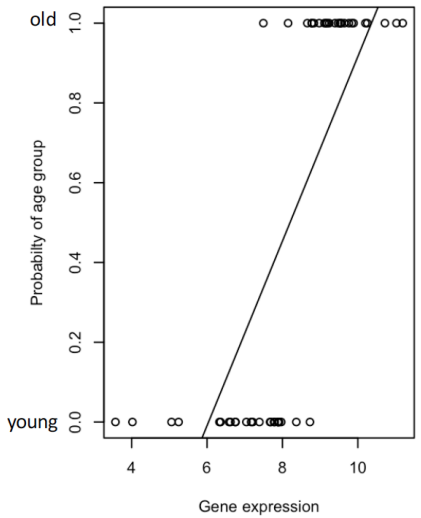
Classification

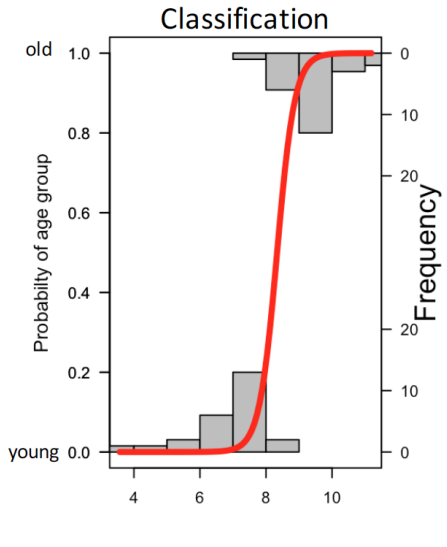
- The response y represents a class label, i.e., given M classes: $y \in \{c_1, c_2, \ldots, c_M\}$
  - e.g., c_1: benign tumor and c_2: malignant tumor

Classification

- The response y represents a class label, i.e., given M classes: $y \in \{c_1, c_2, \ldots, c_M\}$
  - e.g., c_1: benign tumor and c_2: malignant tumor







- Probabilities can be greater than 1 or less than 0
- Difficult to use for more than 2 categories (implies ordering of groups)
- Poor decision boundary

- Probabilities are between 0 and 1.
- Determine decision boundary
- stat::glm(..., family = "binomial")

Classification (cont.)

- A **classifier** is a function that maps an input feature vectors $\boldsymbol{X}$ (vector of size p into an output *class* $c(\boldsymbol{X})$
  - e.g., expression of p genes to predict if a tumor is malignant

Classification (cont.)

- A **classifier** is a function that maps an input feature vectors $\boldsymbol{X}$ (vector of size p into an output *class* $c(\boldsymbol{X})$
  - e.g., expression of p genes to predict if a tumor is malignant
- Notation: to distinguish a continuous from a discrete response, I will denote the response

$$c(\boldsymbol{X}) \in C = \{c_1, c_2, \ldots, c_M\}$$

A Loss Function to Classify

The prediction of the class depends on how costly a missclassification is

- e.g., we may more likely classify tumors as ”malignant” as the cost of classifying it as ”benign” when it was not is too high

A Loss Function to Classify

The prediction of the class depends on how costly a missclassification is

- e.g., we may more likely classify tumors as ”malignant” as the cost of classifying it as ”benign” when it was not is too high

The cost of missclassification can be modeled using a loss function:

- e.g., all errors are equally bad

A Loss Function to Classify

The prediction of the class depends on how costly a missclassification is

- e.g., we may more likely classify tumors as ”malignant” as the cost of classifying it as ”benign” when it was not is too high

The cost of missclassification can be modeled using a loss function:

- e.g., all errors are equally bad

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

Minimizing the Expected Loss

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

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

A simple case: M=2 (two classes: $c_1 = 1$ and $c_2 = 0$):

- The expected loss of predicting a class 0 when it was a 1 is $L(1,0)P(c_1|\boldsymbol{X})$
- The expected loss of predicting a class 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 $P(C = c_j|\boldsymbol{X})$?

Classification Approaches

Three main ways to solve classification problems:

Classification Approaches

Three main ways to solve classification problems:

1. Learn a "generative" model for the probability distribution of the inputs for each class, i.e., $f(\boldsymbol{X}|c_j)$. Use this function and the class probability $P(c_j)$ to find $P(c_j|\boldsymbol{X})$

Classification Approaches

Three main ways to solve classification problems:

1. Learn a ”generative” model for the probability distribution of the inputs for each class, i.e., $f(\boldsymbol{X}|c_j)$. Use this function and the class probability $P(c_j)$ to find $P(c_j|\boldsymbol{X})$

2. Learn a ”discriminative” model for the conditional probability distribution of classes given inputs, i.e., $P(c_j|\boldsymbol{X})$

Classification Approaches

Three main ways to solve classification problems:

1. Learn a ”generative” model for the probability distribution of the inputs for each class, i.e., $f(\boldsymbol{X}|c_j)$. Use this function and the class probability $P(c_j)$ to find $P(c_j|\boldsymbol{X})$

2. Learn a ”discriminative” model for the conditional probability distribution of classes given inputs, i.e., $P(c_j|\boldsymbol{X})$

3. Learn some function that directly maps an input vector $\boldsymbol{X}$ to a class c_j

Generative Models

We learn models from the training data for the probability or probability density of the inputs for items in each of the possible classes:

$$f(\boldsymbol{X}|c_j) \quad \text{for } j = 1, \ldots, M$$

Using Bayes' theorem:

$$P(c = c_j|\boldsymbol{X}) = \frac{f(\boldsymbol{X}|c = c_j)p_j}{f(\boldsymbol{X})} = \frac{f_j(\boldsymbol{X})p_j}{f(\boldsymbol{X})}$$

For example, we can assume:

$$X|c = c_j \sim \mathcal{N}(\mu_j, \mathbf{\Sigma})$$

Recall the binary case (2 classes) with equal error loss.
Predict 1 if:

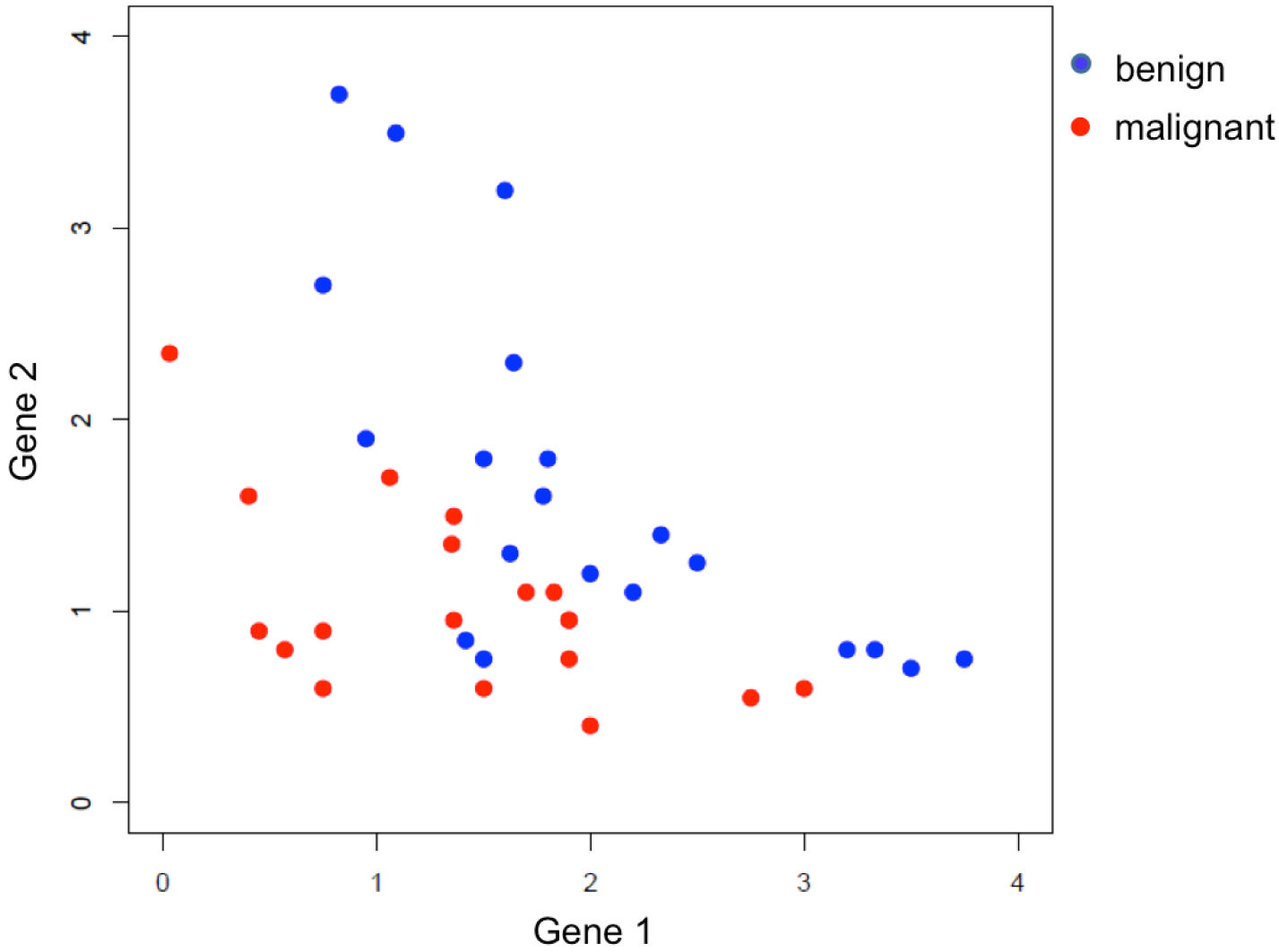
$$\frac{P(c_1|\boldsymbol{X})}{P(c_0|\boldsymbol{X})} > 1 \iff \frac{f_1(\boldsymbol{X})p_1}{f_0(\boldsymbol{X})p_0} > 1 \iff \log\left(\frac{f_1(\boldsymbol{X})p_1}{f_0(\boldsymbol{X})p_0}\right) > 0$$

$$\iff \boldsymbol{a}'\boldsymbol{X} + b > 0$$

where,

$$\begin{aligned} \boldsymbol{a} &= \mathbf{\Sigma}^{-1}(\mu_1 - \mu_0) \\ b &= (\mu_1 - \mu_0)'\mathbf{\Sigma}^{-1}(\mu_1 + \mu_0) - \log\left(\frac{p_0}{p_1}\right) \end{aligned}$$

(Based on artificial data)



Example

- Assume that gene 1 and gene 2 are normally distributed in each class
- The optimal classifier, classifies a point with x =(gene 1,gene 2) in class 1 (”malignant”) if

$$\boldsymbol{a}'\boldsymbol{X} + b > 0$$

- We can estimate this line using estimates of

$$\mu_0,\ \mu_1,\ \boldsymbol{\Sigma},\ p_0,\ p_1$$

- How?

Example: estimates

- Based on the data:

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

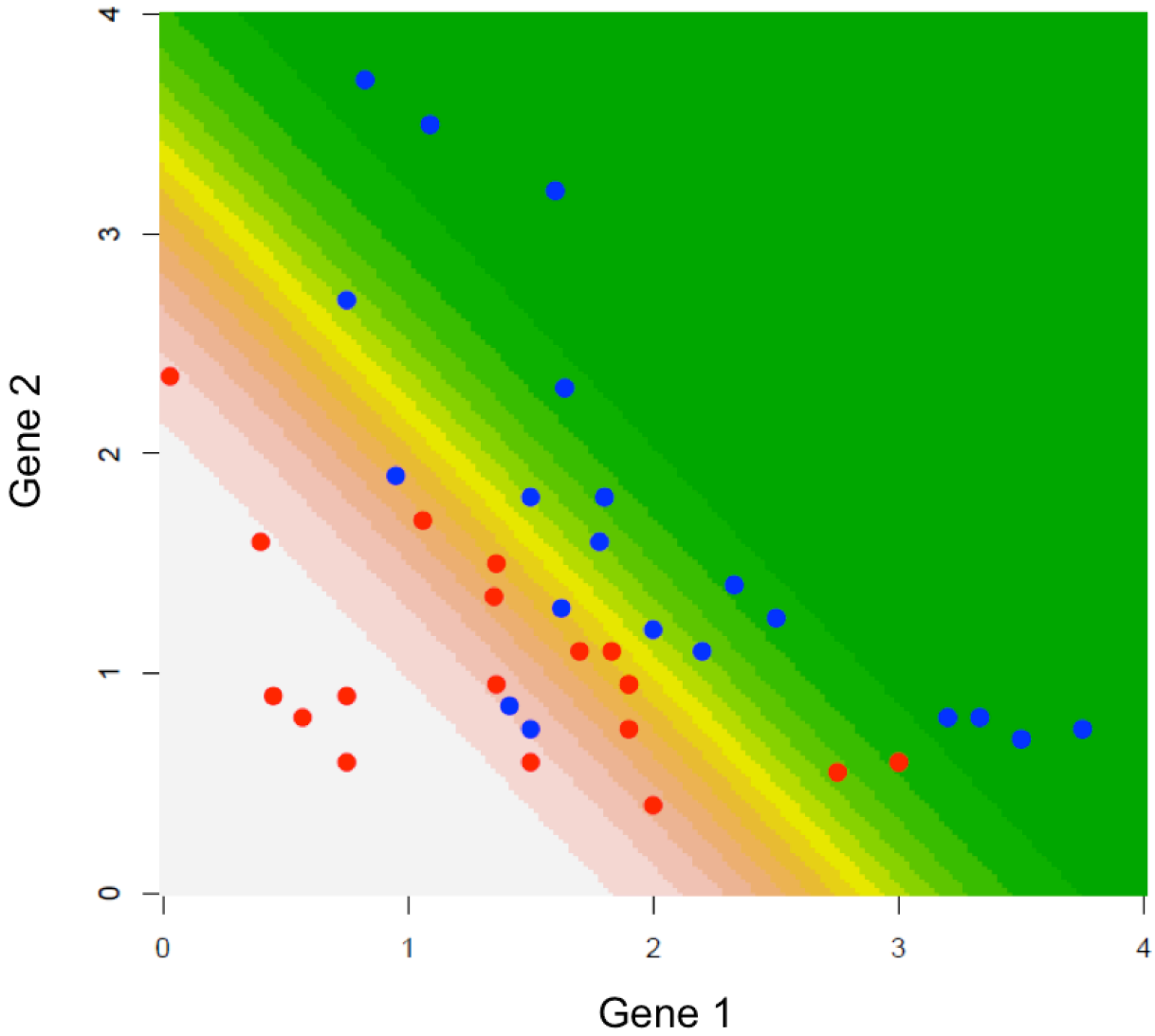
- The optimal classifier classifies a point x in class 1 if

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

- And

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

Linear Discriminant Analysis (LDA)



2) Discriminative Models

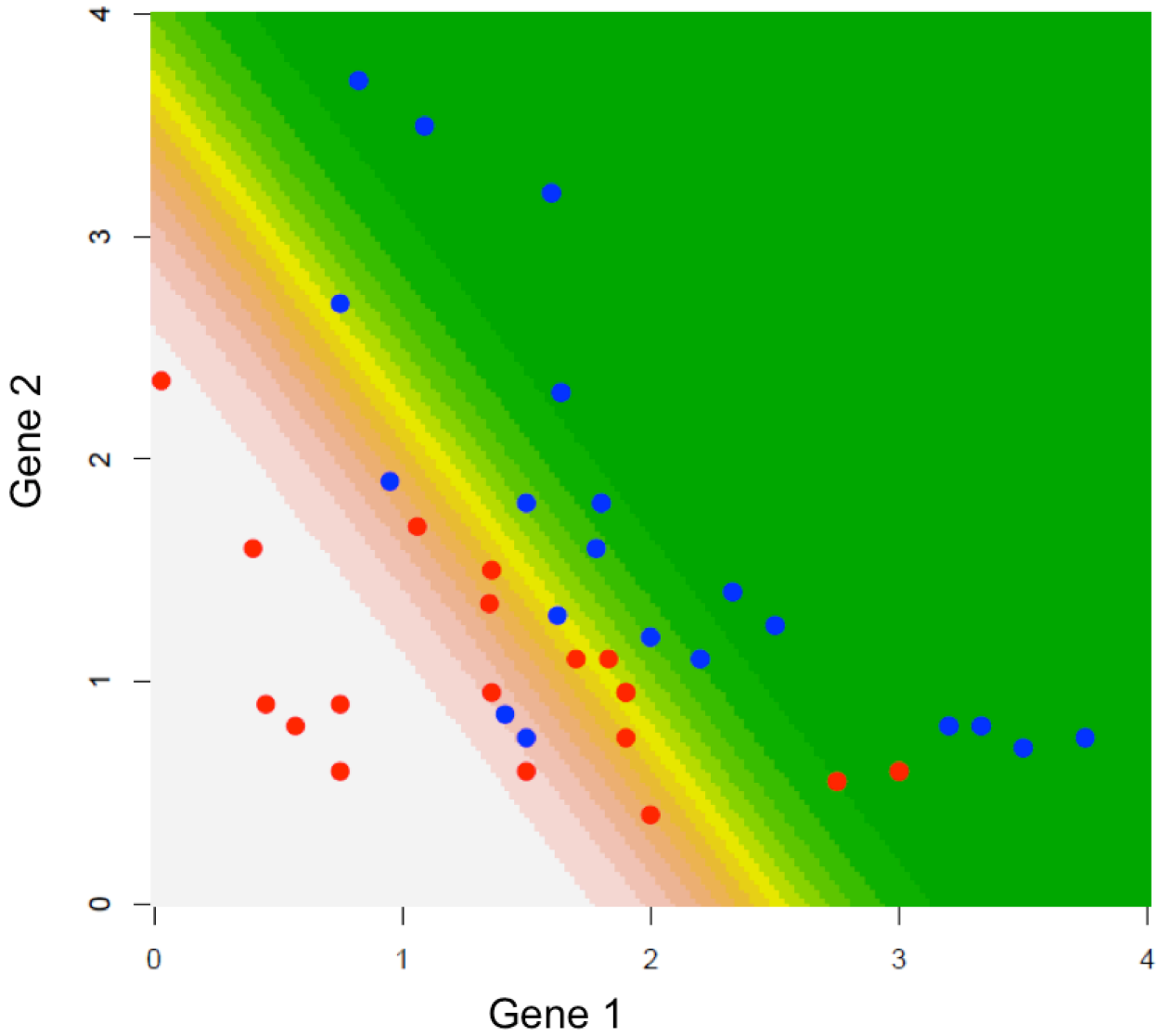
Generative Models: assuming normality

$$\log\left(\frac{P(c=1|\boldsymbol{X})}{P(c=0|\boldsymbol{X})}\right) = \log\left(\frac{P(c=1|\boldsymbol{X})}{1-P(c=1|\boldsymbol{X})}\right) = \boldsymbol{a}'\boldsymbol{X} + b$$

We could simply start with this formula, and estimate $\boldsymbol{a}$ and b from the training data through maximum likelihood (logistic regression)

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

Classification with Logistic Regression



LDA *vs* Logistic

- Both use MLE estimates for the parameters.

LDA *vs* Logistic

- 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.

LDA *vs* Logistic

- 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

3) No Model

Why would one bother thinking about a model for $P(c_j|\boldsymbol{X})$??

If we want to choose the class with the highest probability given $\boldsymbol{x}$, then:

$->$ look for the proportion of points of class c "near" $\boldsymbol{x}$

$->$ and choose the class with the highest proportion.

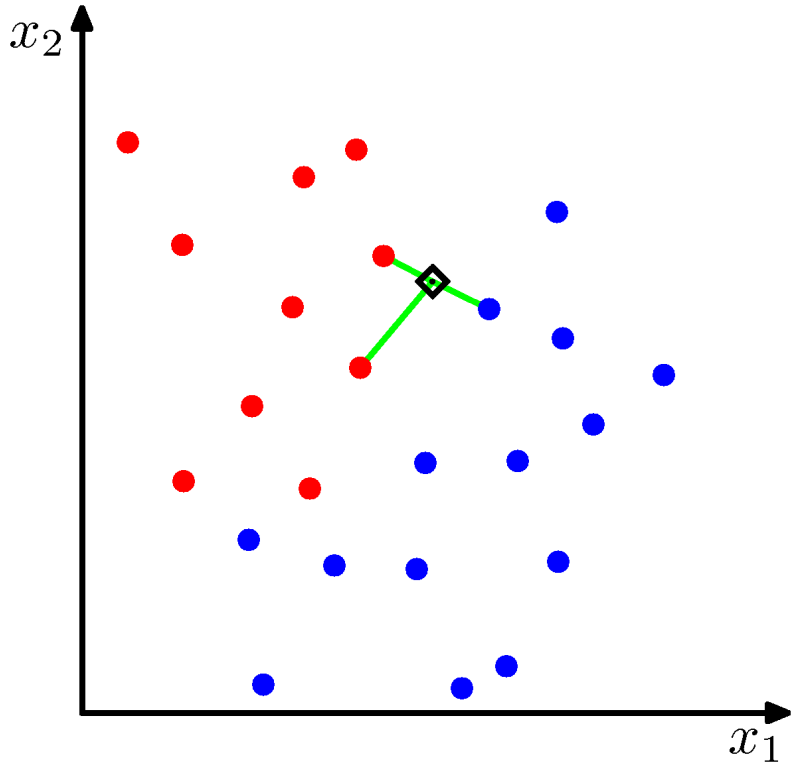
Nearest neighbours
(peer pressure! - majority votes!)

$K-$Nearest Neighbours

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

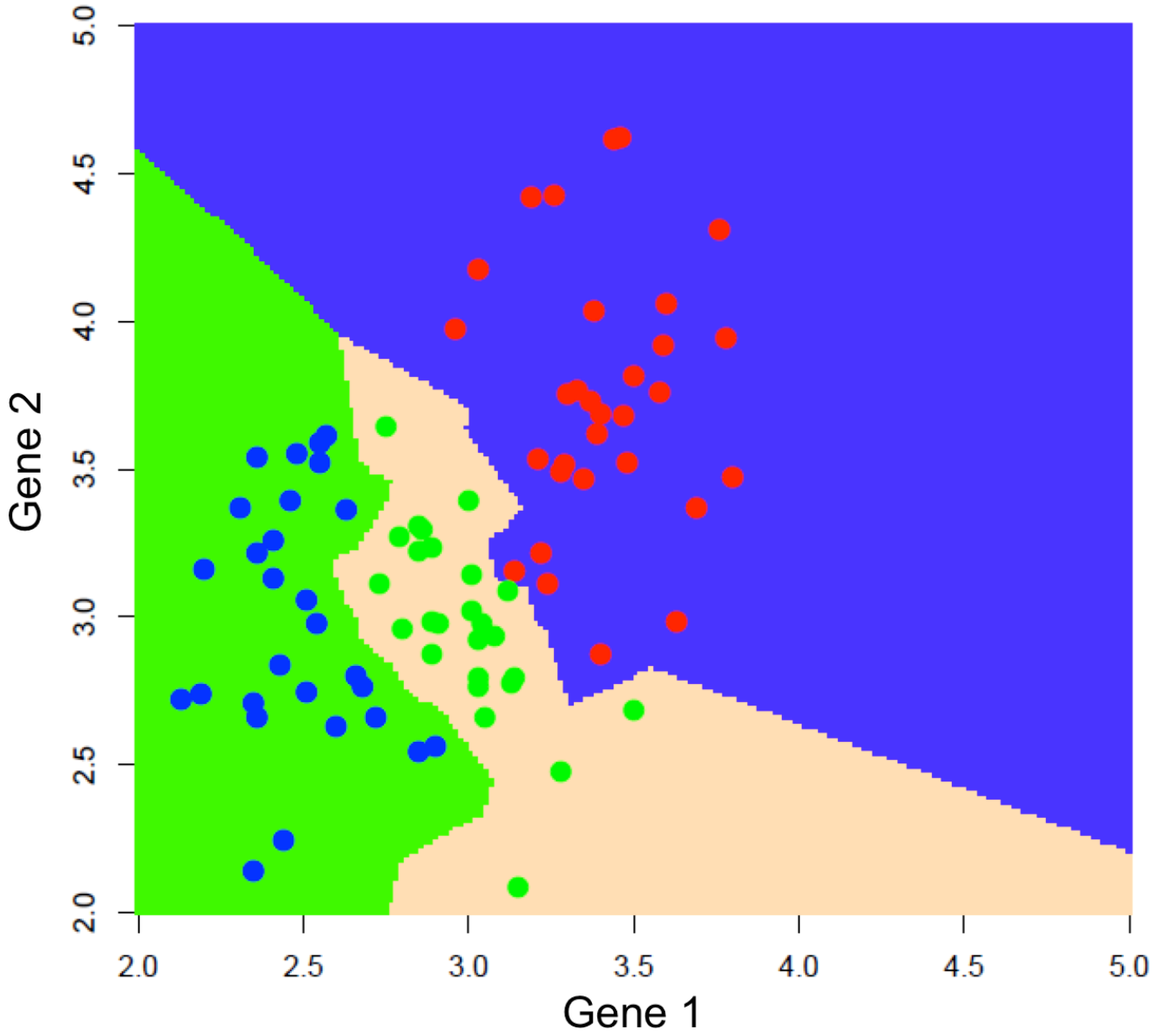
–> count how many points of each class there are among the closest K neighbours to x

–> use the majority class as the predicted class

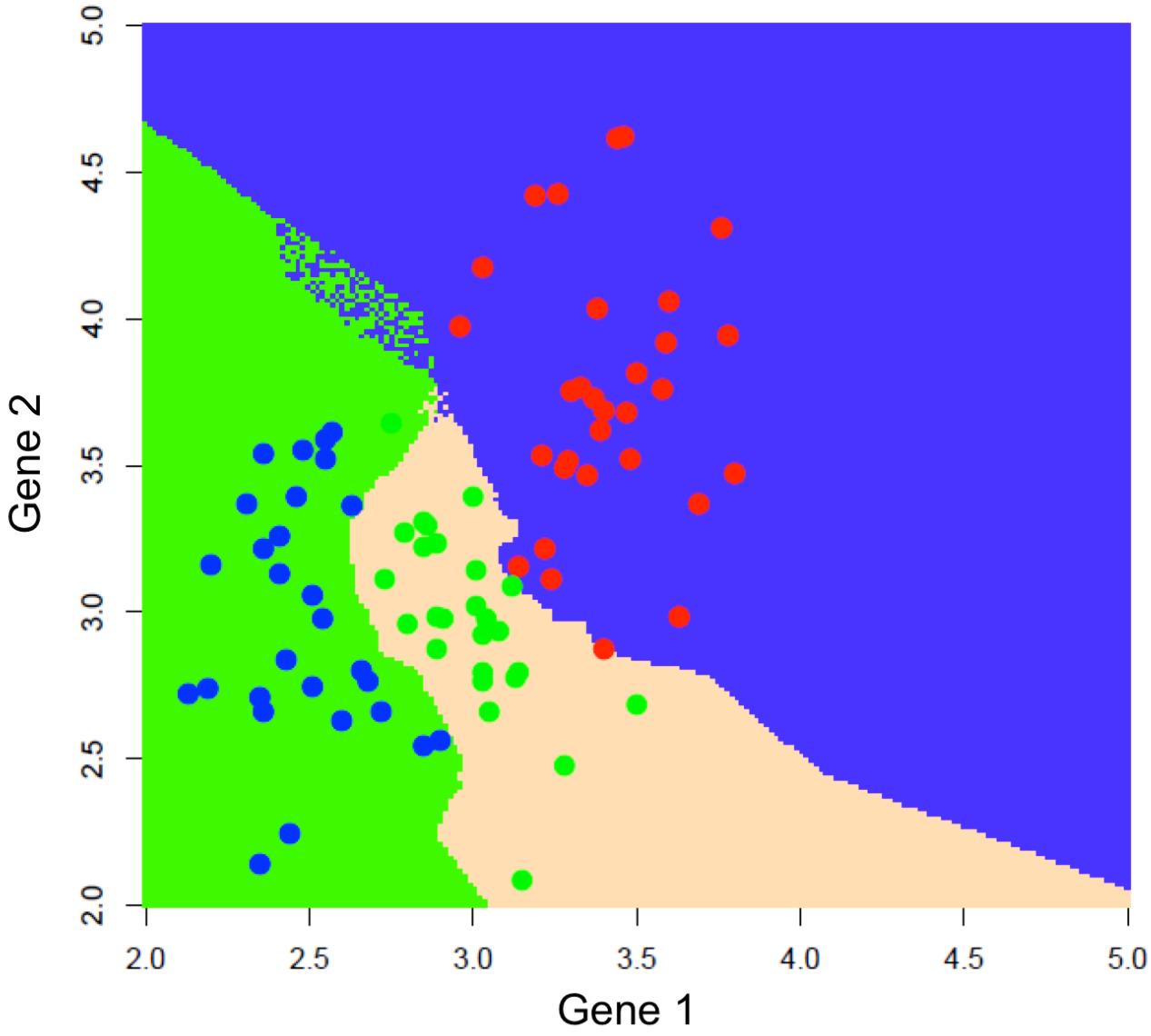


Bishop figure 2.27

1-NN



5-NN

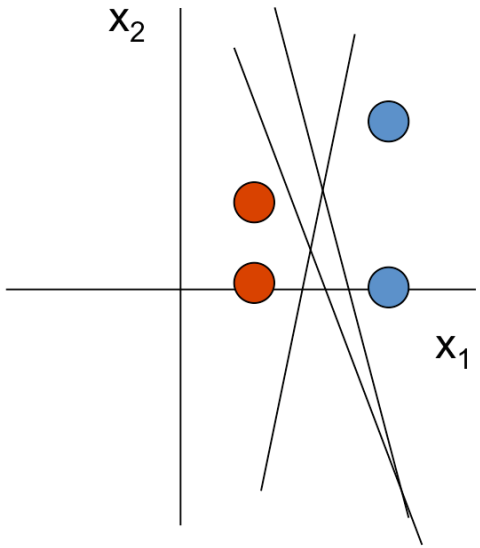


Properties of KNN

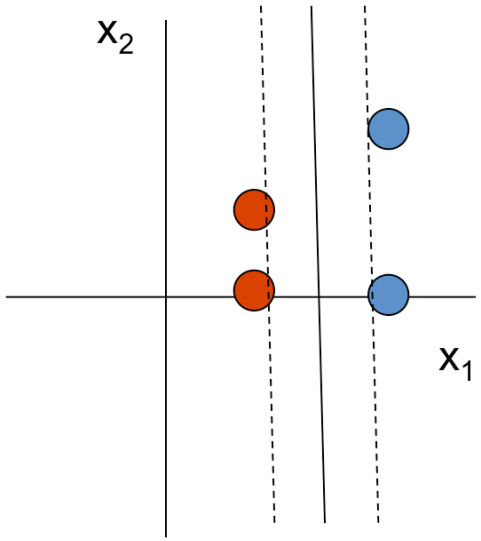
- There is no training step and the estimation of $P(c_j|\boldsymbol{x})$ is simple.
- Choosing K is not trivial: there is a trade-off between variance and bias (decision surface is smoother as K increases).
- Variants which use weighted votes might do somewhat better, but in practice KNN is beaten by other methods.

Support Vector Machine

- Motivation: maximum margin classifier.
- Artificial Neural Network (ANN) finds ”some” solution.
- Support Vector Machine (SVM) will find **maximum margin separating hyperplane**.



KNN: multiple possible solutions

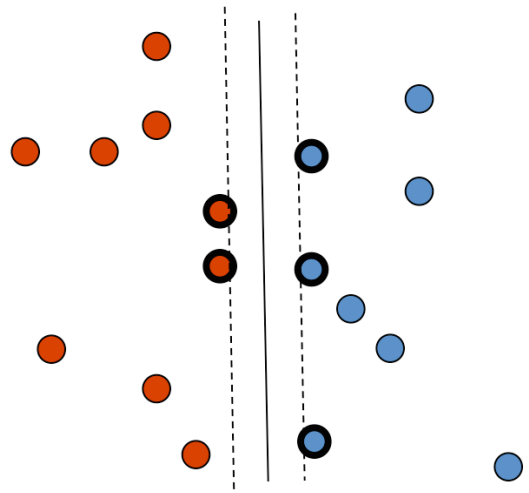


SVM: Only one optimal solution

Support Vector Machine

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.



Paul Pavlidis, STAT540, 2012

What is a support vector machine?

1. A subset of training examples (**support vectors**)
2. A vector of weights for the support vectors ($\vec{\alpha}$)
3. A similarity function: K($\mathbf{x}_1$,$\mathbf{x}_2$) (**kernel** function)

Predicting class labels for example x_i:

$$f(\vec{x}_i) = \text{sign}(\sum_j \alpha_j y_j K(\vec{x}_i, \vec{x}_j))$$

$y_i = \{-1, 1\}$ We are switching to -1,1 class labels for mathematical convenience

(Thanks to P. Domingos for slides)

The SVM kernel "trick"

Instead of working with the original data vectors, we work with a function of the data called **kernel**: $K(\boldsymbol{x}, \boldsymbol{y})$

Examples of kernels:

- Linear: $K(\boldsymbol{x}, \boldsymbol{y}) = \boldsymbol{x}'\boldsymbol{y}$
- Radial Basis: $K(\boldsymbol{x}, \boldsymbol{y}) = \exp\left(-\gamma \|\boldsymbol{x} - \boldsymbol{y}\|^2\right)$
- Polinomial: $K(\boldsymbol{x}, \boldsymbol{y}) = (1 + \boldsymbol{x}'\boldsymbol{y}^d),\ d \geq 0$

Advantages of using kernel

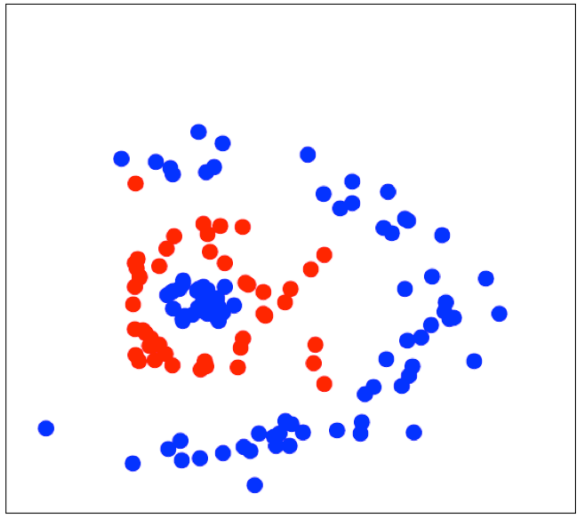
- Can ”project” data into a high dimensional ”feature space”. Allows solving problems that are not linearly separable in the input space.

- Can solve problems where explicit representation is a problem, but where you can define a valid distance between examples.

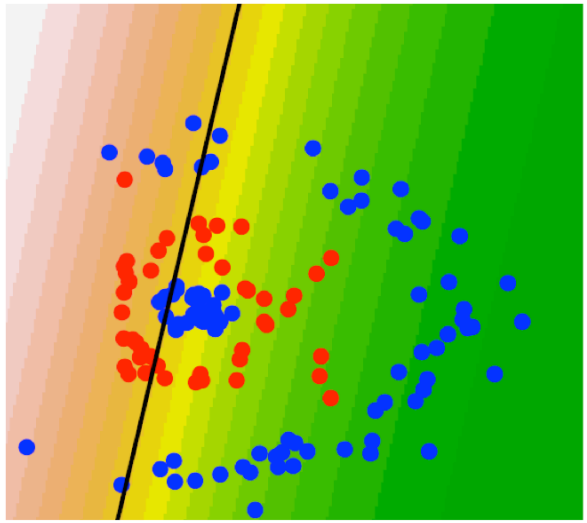
  Example: distances between text strings of different lengths.

Other aspects of SVMs

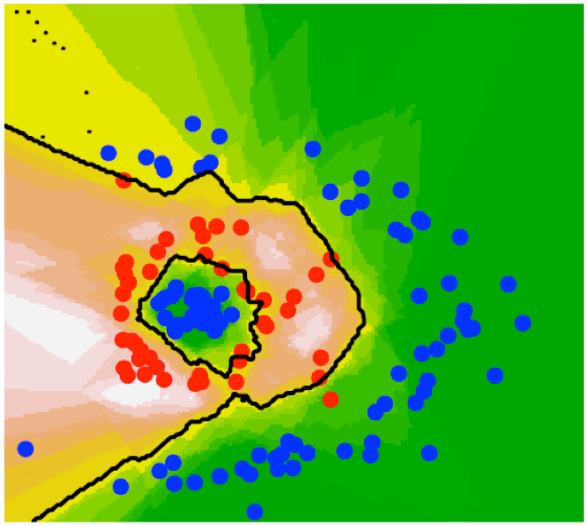
- Can make margin ”soft”, so it tolerates misclassified training examples. This is used in practice to solve unseparable problems without overfitting.
- Because solution is sparse (few support vectors) classification is quick.



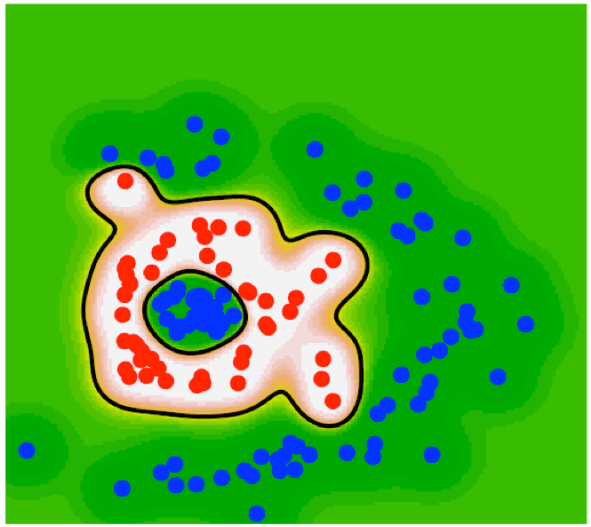
LDA



kNN 15



SVM



Conclusions

- There are many methods and algorithms to perform classification
- Which one is better? We need a criteria to evaluate them
- All methods were illustrated based on 2 genes. Some of the methods will work well with large number of covariates (e.g., kNN or SVM). However, other methods require to have more observations than covariates ($n > p$) (e.g., LDA)