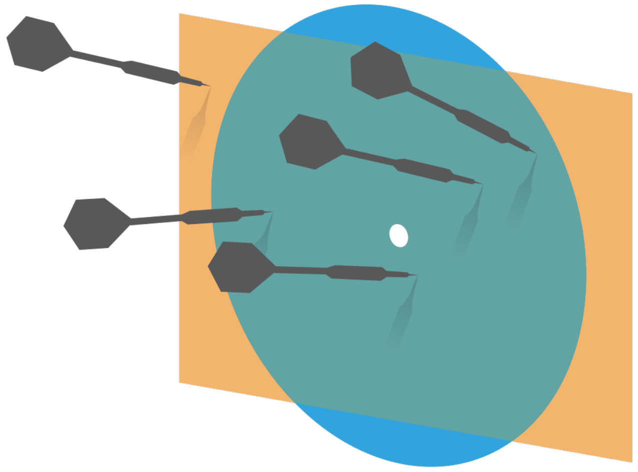
Probability and Statistics for Computer Science



"…many problems are naturally classification problems"---Prof. Forsyth

Credit: wikipedia

Hongye Liu, Teaching Assistant Prof, CS361, UIUC, 10.29.2020

Last time

- Review of Covariance matrix
- Dimension Reduction
- Principal Component Analysis
- Examples of PCA

Objectives

- Demo of Principal Component Analysis
- Introduction to classification

$\downarrow$

Dimension Reduction

Q. Which are true?

A . PCA allows us to project data to the direction along which the data has the biggest variance

B. PCA allows us to compress data

C. PCA uses linear transformation to show patterns of data

D. PCA allows us to visualize data in lower dimensions

[✓ E.] All of the above

unsupervised

Demo of the PCA by solving diagonalization of covariance matrix

Mean centering

Rotate the data to eigenvectors

← diagonalize the covariant

Project the data

(choose a few important PCs)

Notebook 18

Diagonalization example

- For

$$A = \begin{bmatrix} 5 & 3 \\ 3 & 5 \end{bmatrix}$$

λ_i ? $|A - \lambda I| = 0$

$$\begin{vmatrix} 5-\lambda & 3 \\ 3 & 5-\lambda \end{vmatrix} = 0 \Rightarrow \begin{cases} \lambda_1 = 8 \\ \lambda_2 = 2 \end{cases}$$

eigenvectors?

$\lambda_1 = 8$

$A v_1 = 8 v_1$

$(A - 8I) v_1 = 0$

$$\begin{bmatrix} -3 & 3 \\ 3 & -3 \end{bmatrix} v_1 = 0 \Rightarrow v_1 = \begin{bmatrix} 1 \\ 1 \end{bmatrix}$$

$v_1' = \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

$u_1' = \frac{1}{\sqrt{2}} \begin{bmatrix} -1 \\ -1 \end{bmatrix}$

$\Rightarrow u_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ 1 \end{bmatrix}$

$\uparrow$ normalized eigenvector

$U = [u_1 \quad u_2]$

$= ?$

$\lambda_2 = 2 \quad u_2 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 \\ -1 \end{bmatrix}$

$\Lambda = U^T A U$

$$\Lambda = ? \begin{bmatrix} 8 & 0 \\ 0 & 2 \end{bmatrix}$$

Q. Which of these is NOT true?

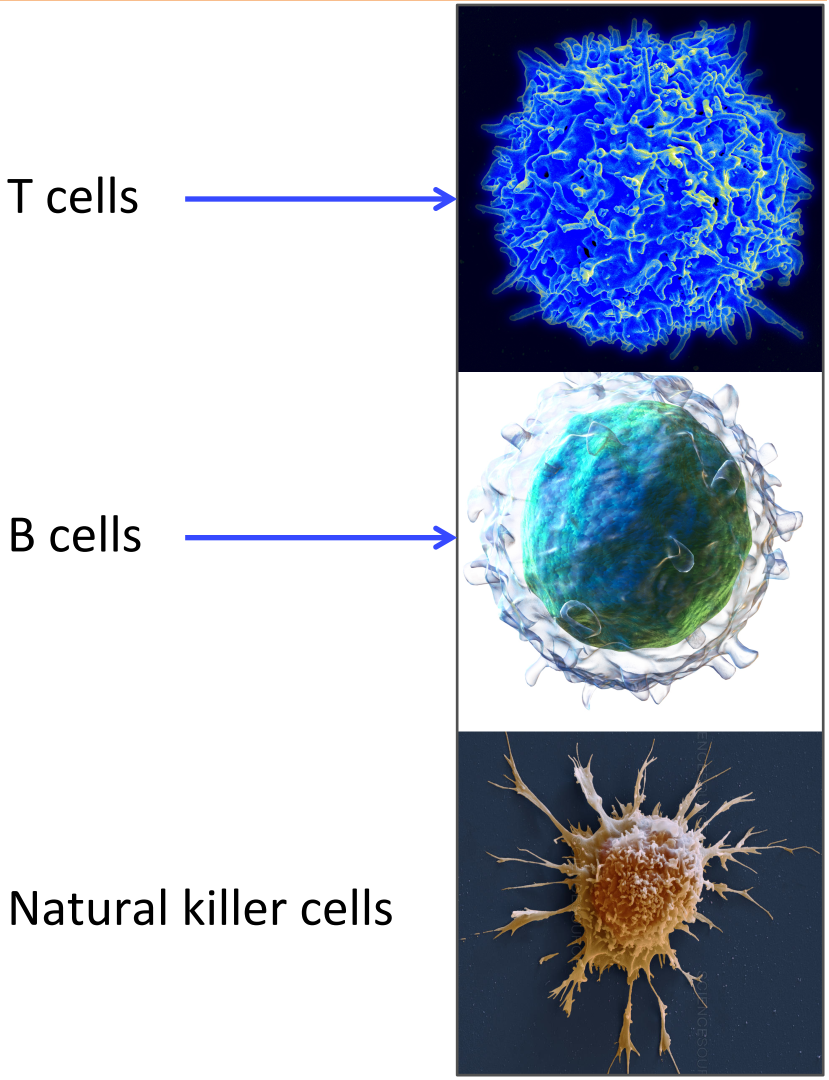
A. The eigenvectors of covariance can have opposite signs and it won’t affect the reconstruction

B. The PCA analysis in some statistical program returns standard deviation instead of variance

[✓ C.] It doesn’t matter how you store the data in matrix

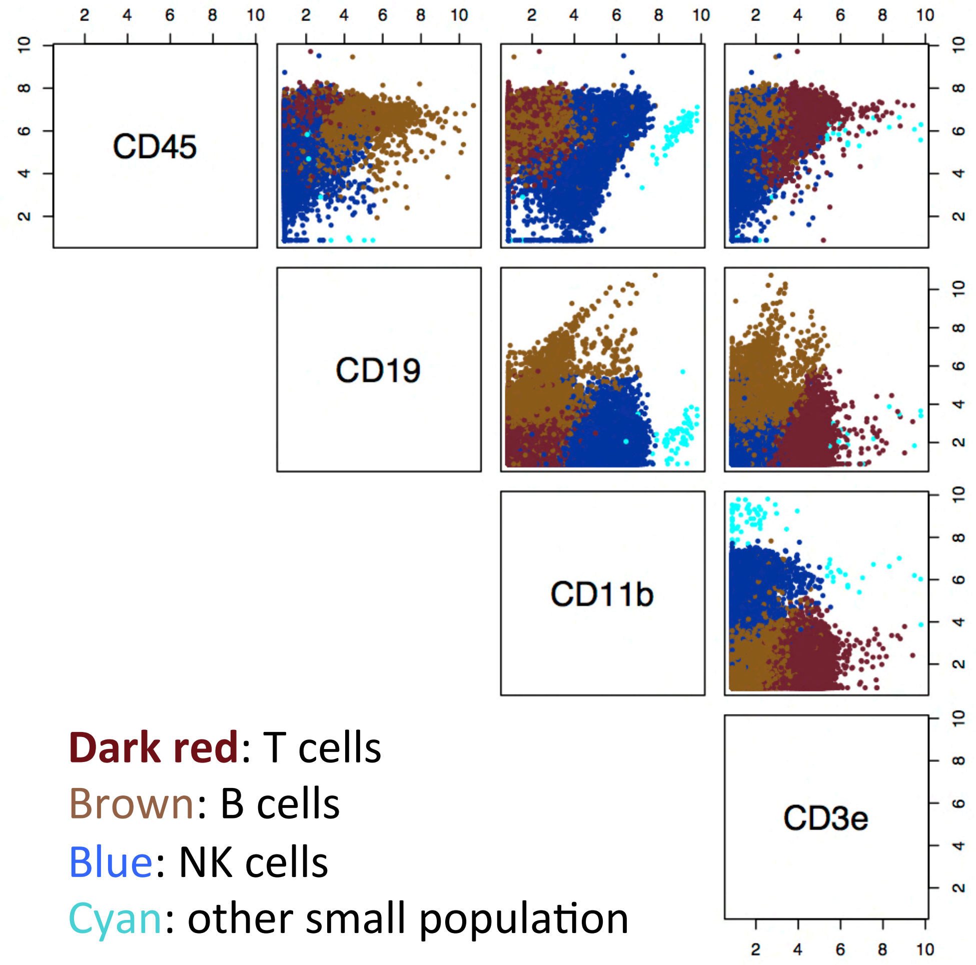
Demo: PCA of Immune Cell Data

- There are 38816 white blood immune cells from a mouse sample
- Each immune cell has 40+ features/ components
- Four features are used as illustration.
- There are at least 3 cell types involved



Scatter matrix of Immune Cells

- There are 38816 white blood immune cells from a mouse sample
- Each immune cell has 40+ features/ components
- Four features are used as illustration.
- There are at least 3 cell types involved

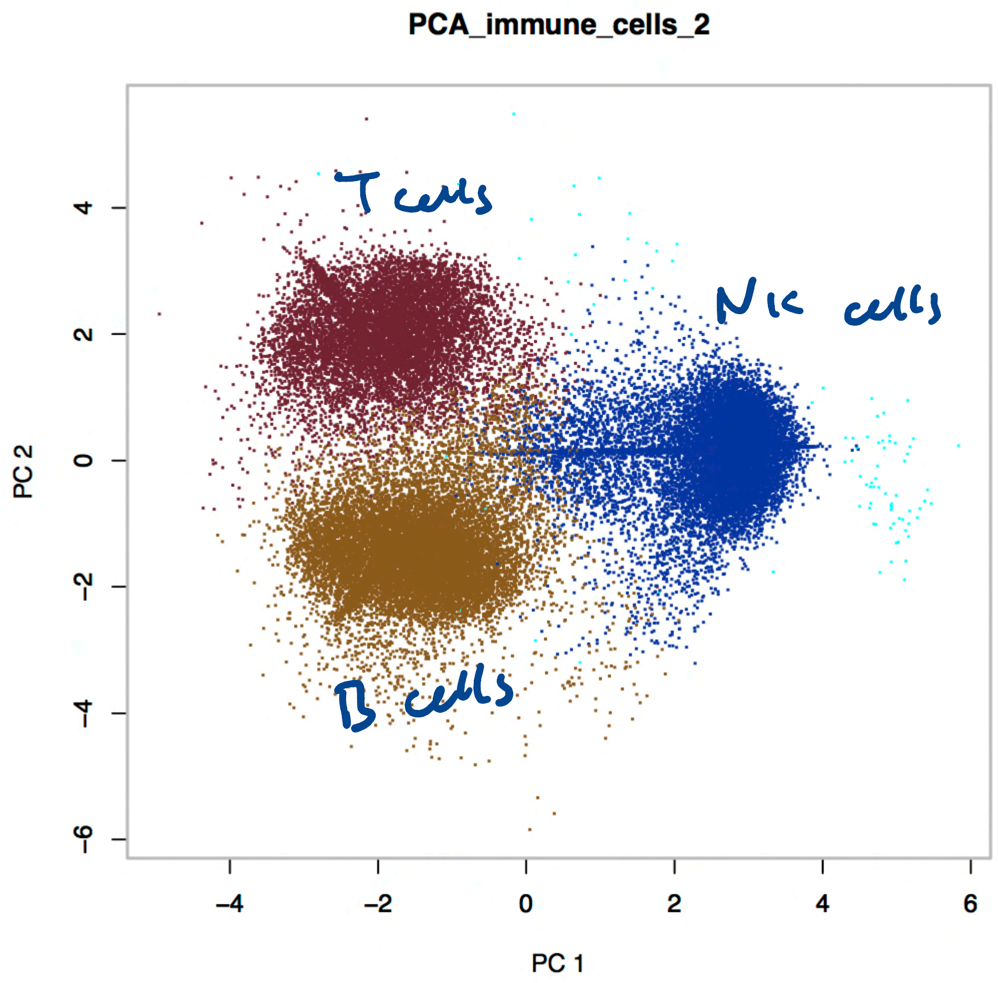


Dark red: T cells
Brown: B cells
Blue: NK cells
Cyan: other small population

PCA of Immune Cells

```
> res1
$values   Eigenvalues
[1] 4.7642829 2.1486896 1.3730662
0.4968255

            Eigenvectors
$vectors
          [,1]       [,2]       [,3]       [,4]
[1,]  0.2476698  0.00801294 -0.6822740
0.6878210
[2,]  0.3389872 -0.72010997 -0.3691532
-0.4798492
[3,] -0.8298232  0.01550840 -0.5156117
-0.2128324
[4,]  0.3676152  0.69364033 -0.3638306
-0.5013477
```



More features used

- There are 38816 white blood immune cells from a mouse sample

$N = 38{,}816$

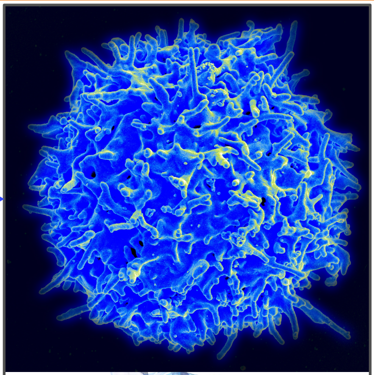
- Each immune cell has **42 features**/components

$d = 42$

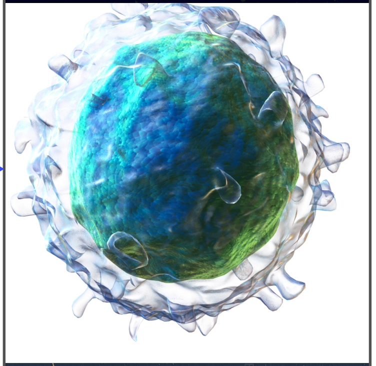
- There are at least 3 cell types involved

Curated labels

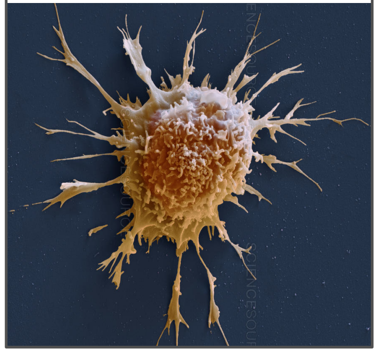
T cells



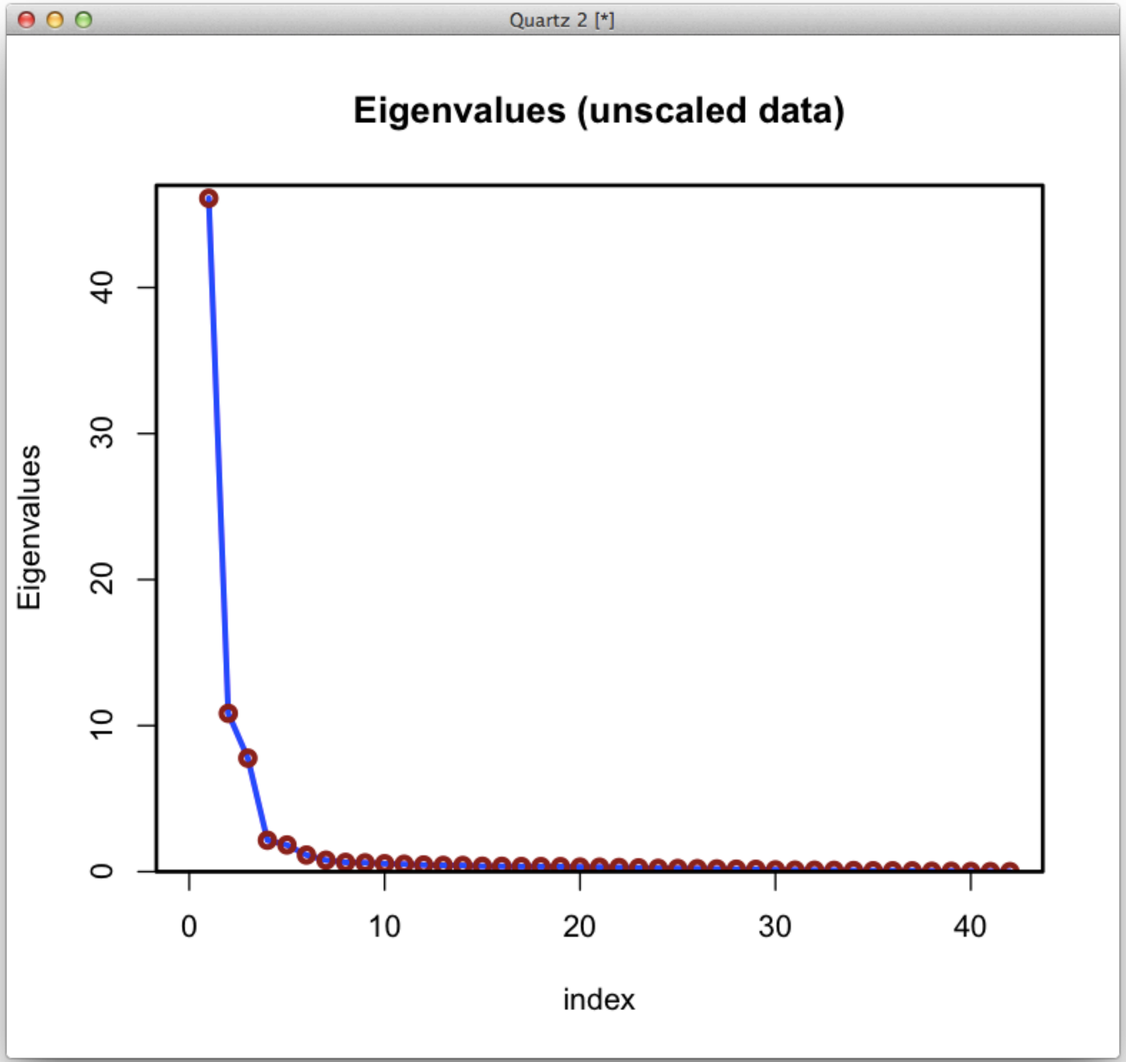
B cells



Natural killer cells

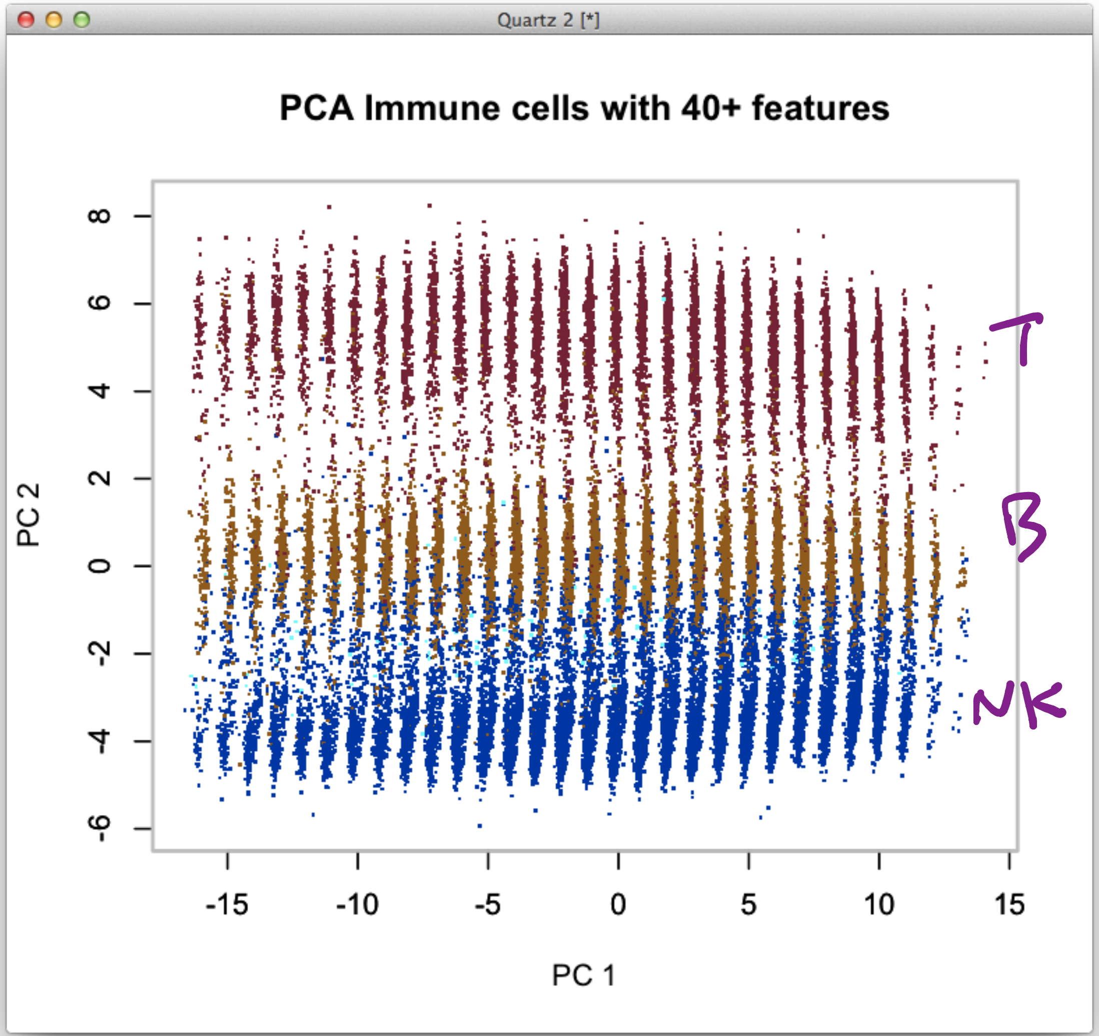


Eigenvalues of the covariance matrix

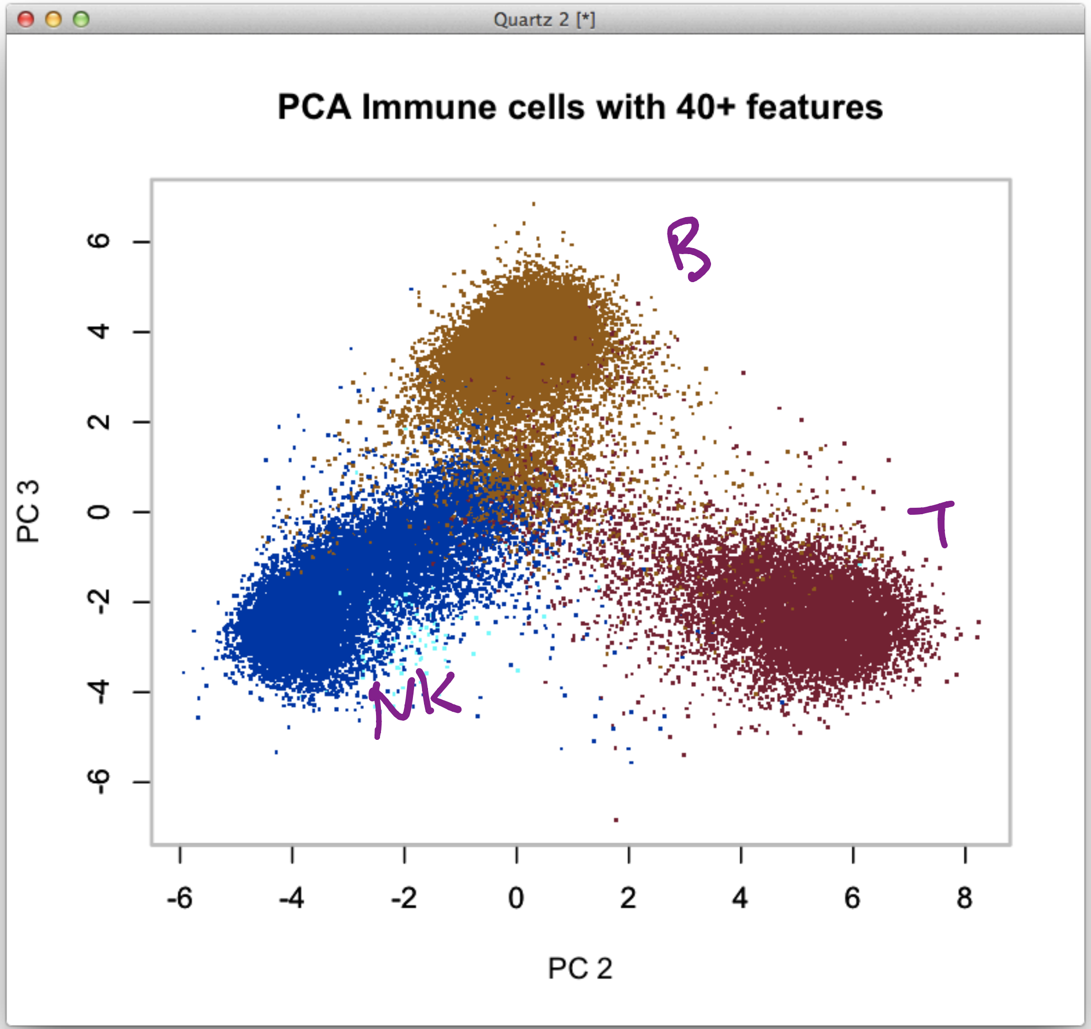


Large variance doesn't mean important pattern

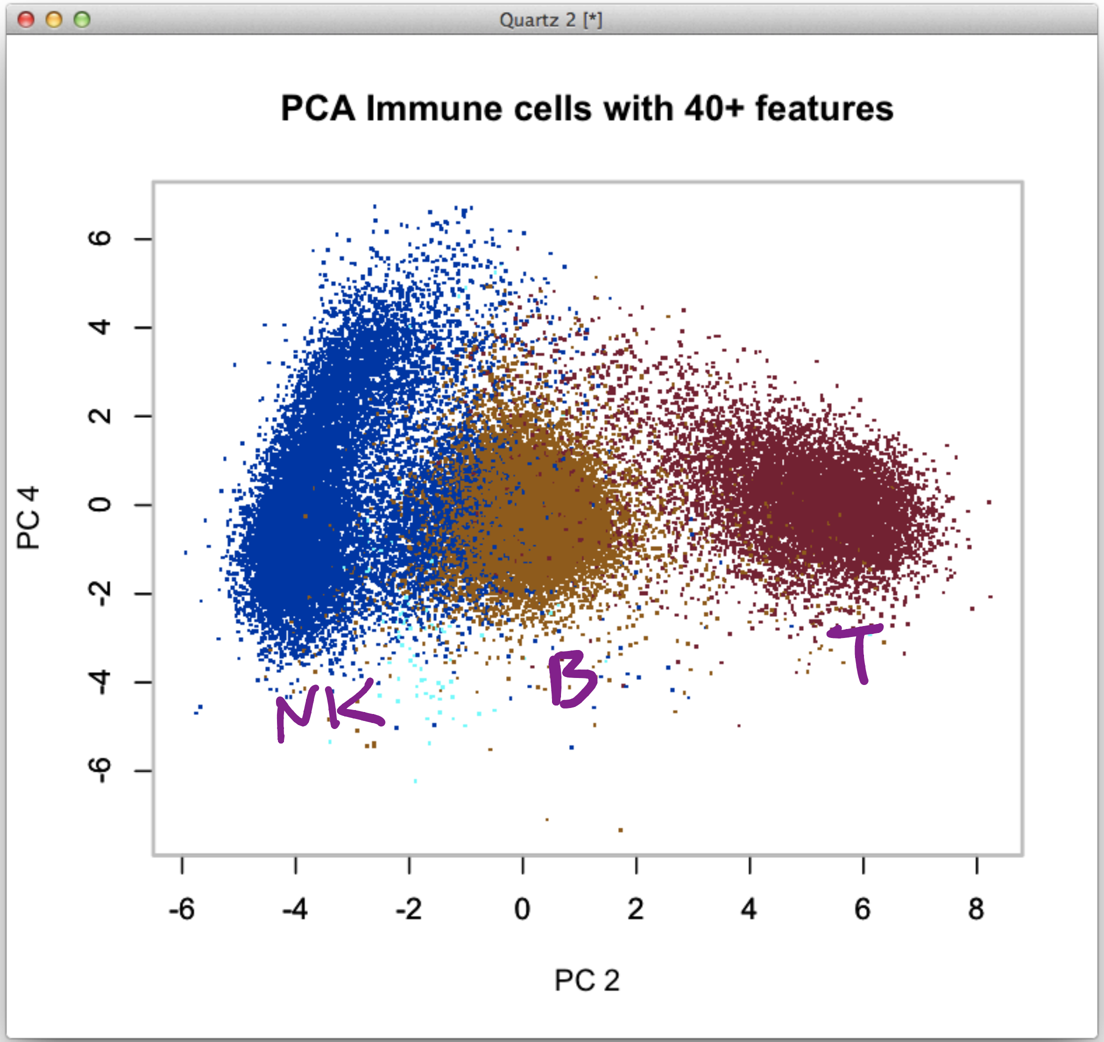
Principal component 1 is just cell length



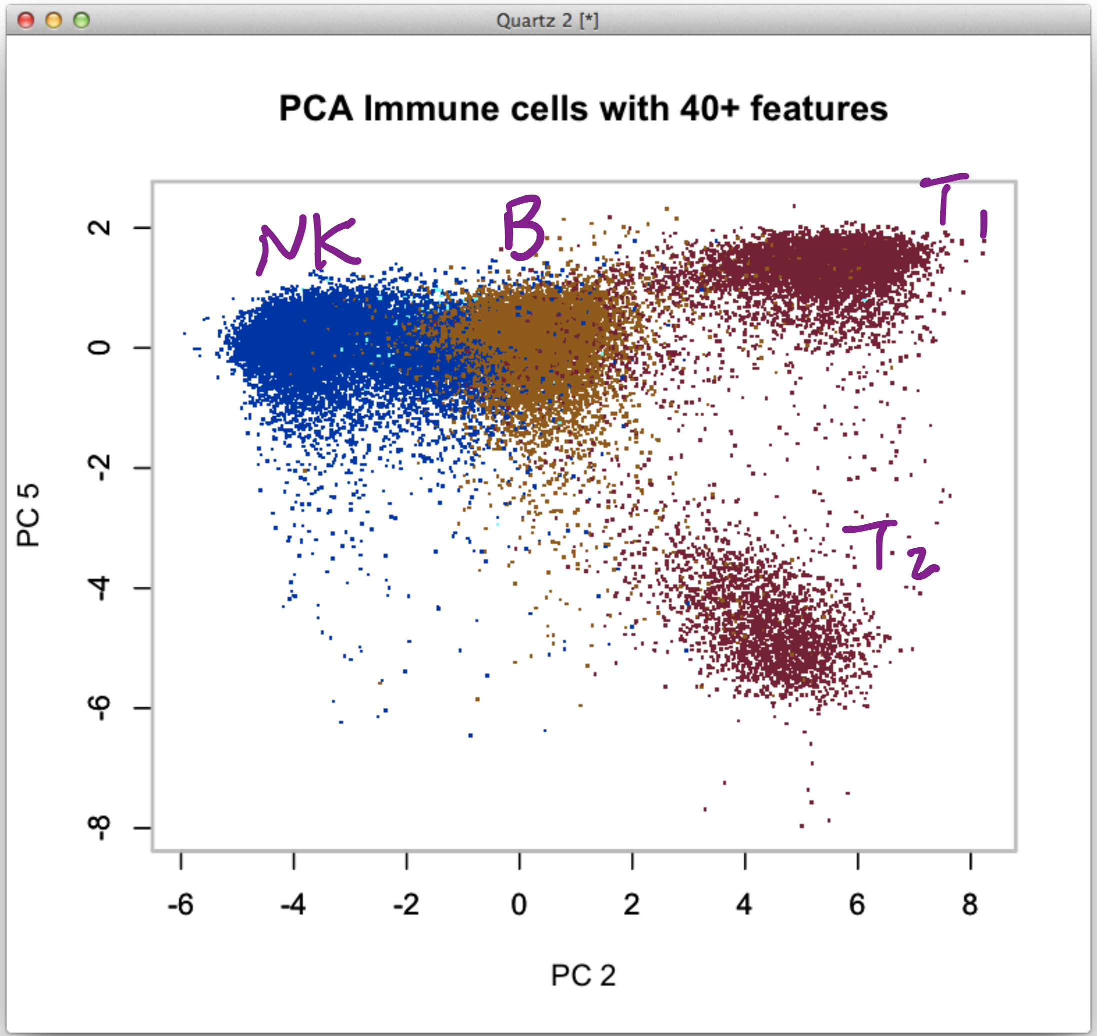
Principal component 2 and 3 show different cell types



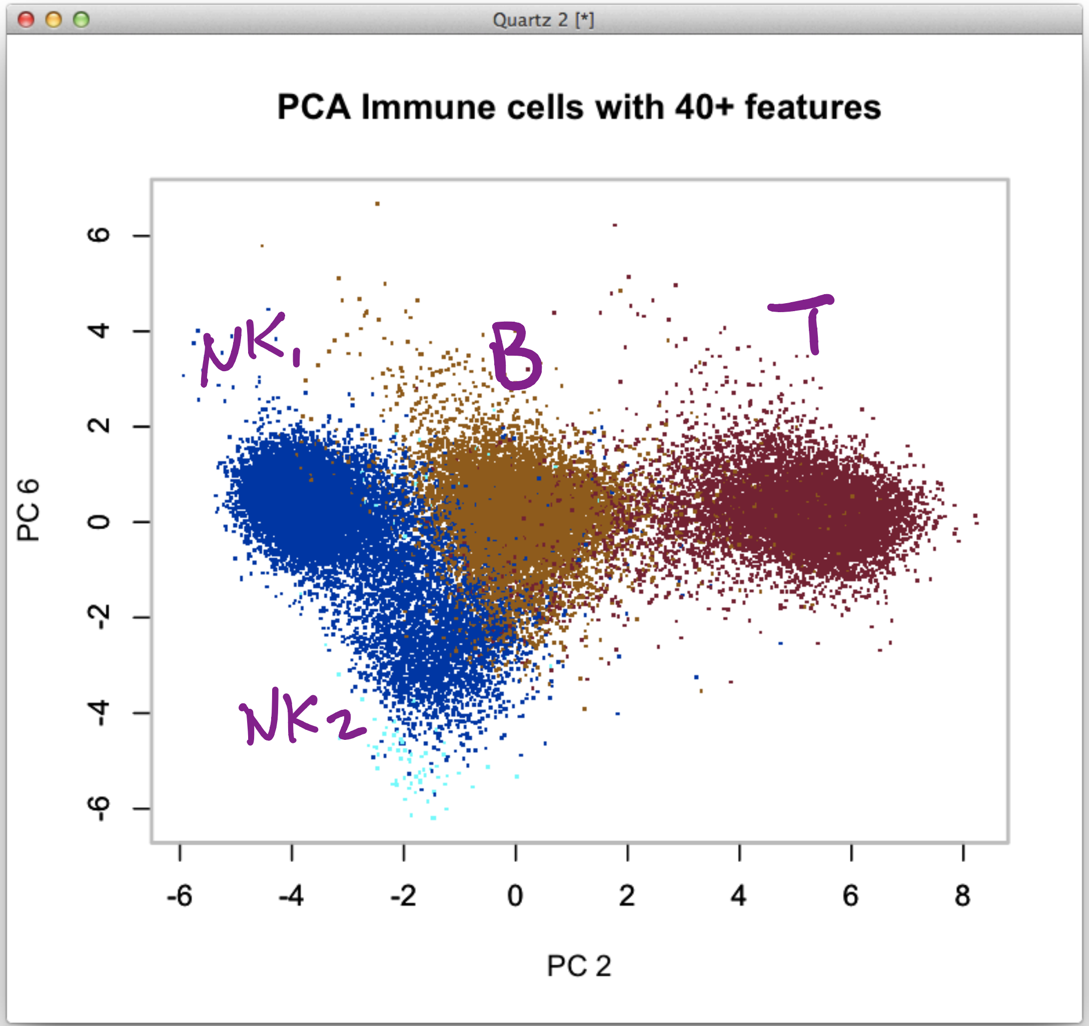
Principal component 4 is not very informative



Principal component 5 is interesting



Principal component 6 is interesting

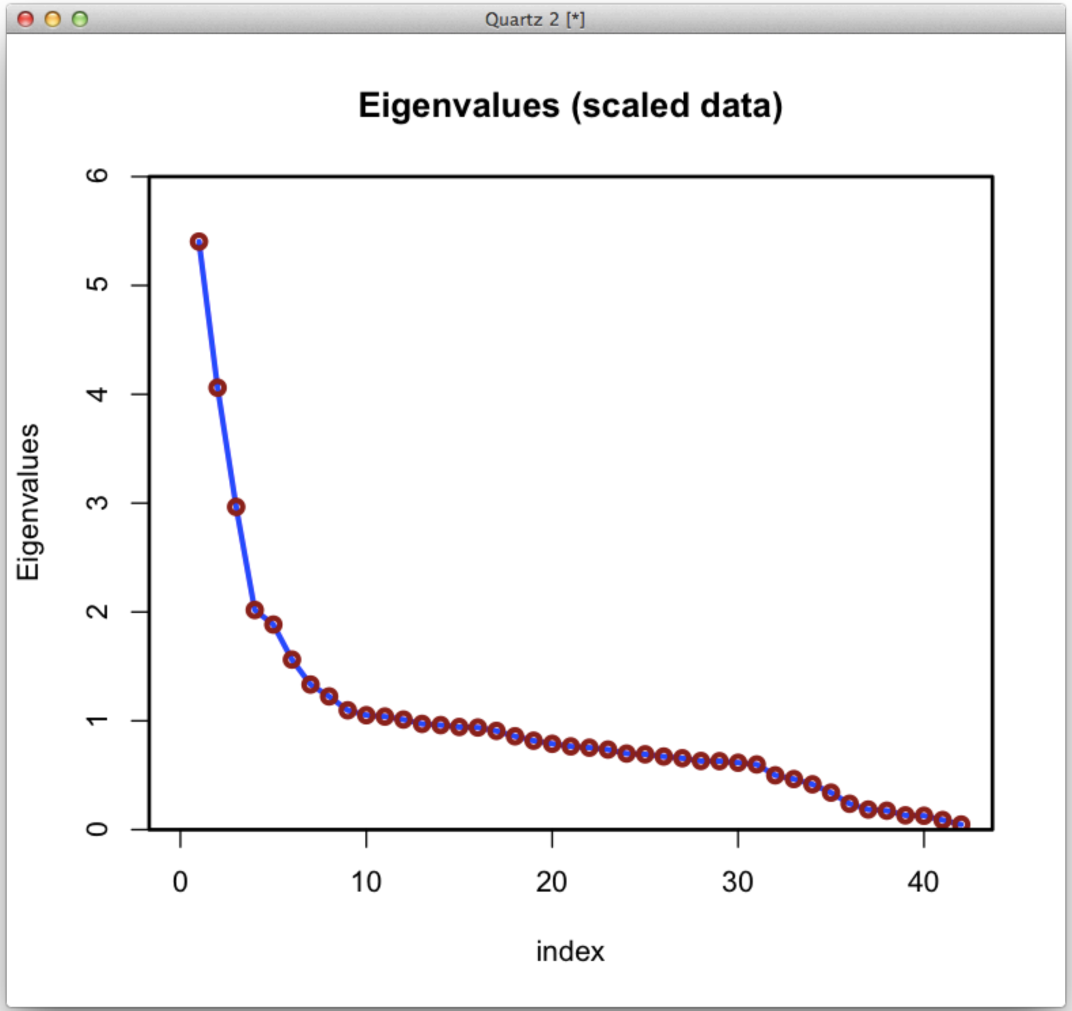


Scaling the data or not in PCA

- Sometimes we need to scale the data for **each** feature have very different value range.
- After scaling the eigenvalues may change significantly.
- Data needs to be investigated case by case

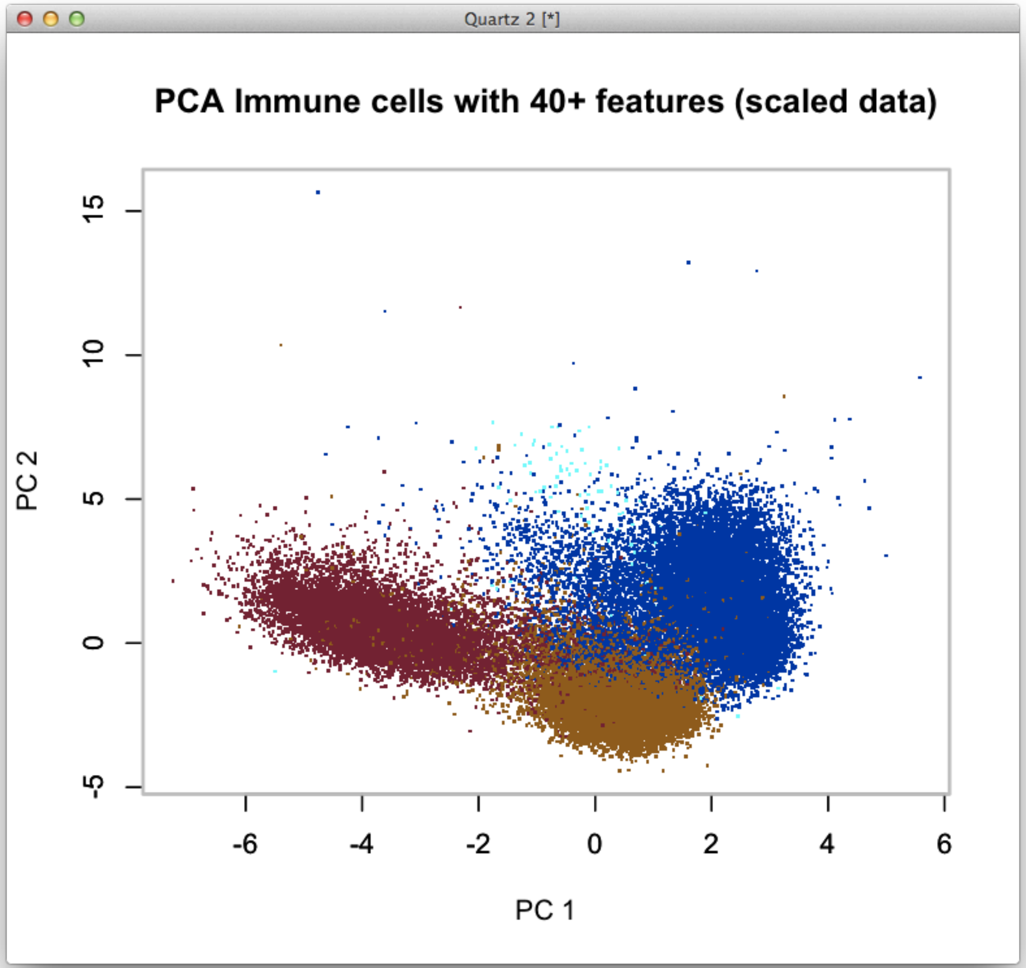
Eigenvalues of the covariance matrix (scaled data)

Eigenvalues do not drop off very quickly



Principal component 1 & 2 (scaled data)

Even the first 2 PCs don't separate the different types of cell very well



$$\underline{\text{covmat}(\{x\})}$$

$$U = [u_1 \; u_2 \; \cdots \; u_d]$$

$$r = \underline{\boxed{U^T} m}$$

$$\text{covmat}(\{r\}) = \Lambda$$

m = matrix $d \times N$

d rows

feature 1

Q. Which of these are true?

✓ A. Feature selection should be conducted with domain knowledge

✓ B. Important feature may not show big variance

✗ C. Scaling doesn’t change eigenvalues of covariance matrix

[✓ D. A & B]

Q. Which of these are true?

A. Feature selection should be conducted with domain knowledge

B. Important feature may not show big variance

C. Scaling doesn’t change eigenvalues of covariance matrix

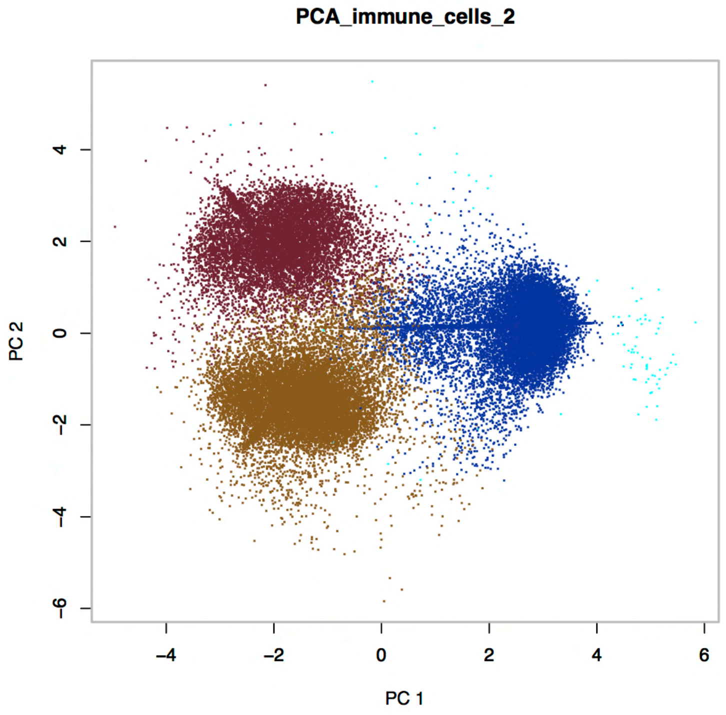
[✓ D. A & B]

Learning to classify

- Given a set of feature vectors x_i, where each has a class label y_i, we want to train a classifier that maps unlabeled data with the same features to its label.

$\overbrace{\hspace{6em}}$

CD45	CD19	CD11b	CD3e	Type
6.59564671	1.297765164	7.073280884	1.155202366	1
6.742586812	4.692018952	3.145976639	1.572686963	4
6.300680301	1.20613983	6.393630905	1.424572629	2
5.455310882	0.958837541	6.149306002	1.493503124	1
5.725565772	1.719787885	5.998232014	1.310208305	1
5.552847151	0.881373587	6.02155471	0.881373587	3
5	0	7	1.2	?



Binary classifiers

- A binary classifier maps each feature vector to one of two classes.
- For example, you can train the classifier to:
  - Predict a gain or loss of an investment
  - Predict if a gene is beneficial to survival or not
  - …

Multiclass classifiers

- A multiclass classifier maps each feature vector to one of three or more classes.
- For example, you can train the classifier to:
  - Predict the cell type given cells' measurement
  - Predict if an image is showing tree, or flower or car, etc
  - …

Given our knowledge of probability and statistics, can you think of any classifiers?

Bayesian Inference

probability

$P(\theta | D)$

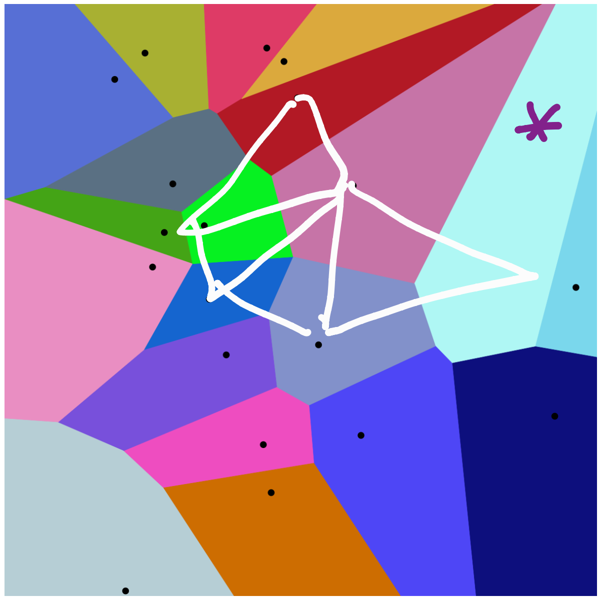
Given our knowledge of probability and statistics, can you think of any classifiers?

- We will cover classifiers such as nearest neighbor, decision tree, random forest, Naïve Bayesian and support vector machine.

Nearest neighbors classifier

- Given an unlabeled feature vector
  - Calculate the distance from **x**
  - Find the closest labeled $\mathbf{x}_i$
  - Assign the same label to **x**

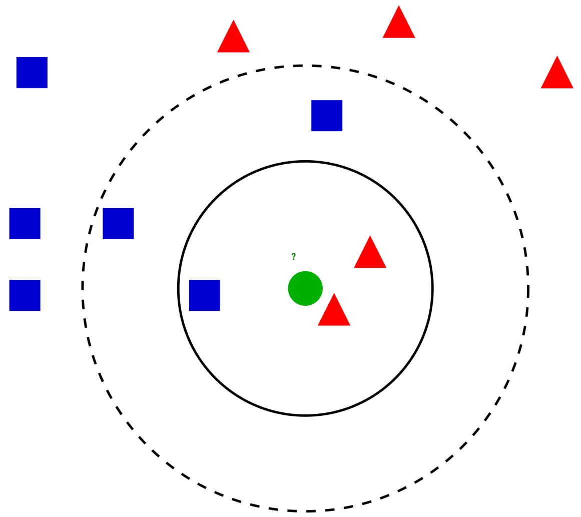
- Practical issues
  - We need a distance metric
  - We should first standardize the data
  - Classification may be less effective for very high dimensions



Source: wikipedia

Variants of nearest neighbors classifier

- In k-nearest neighbors, the classifier:
  - Looks at the k nearest labeled feature vectors $\mathbf{x}_i$
  - Assigns a label to $\mathbf{x}$ based on a majority vote



- In (k, ℓ)-nearest neighbors, the classifier:
  - Looks at the k nearest labeled feature vectors
  - Assigns a label to $\mathbf{x}$ if at least ℓ of them agree on the classification

How do we know if our classifier is good?

- We want the classifier to avoid some mistakes on unlabeled data that we will see in run time.

- **Problem 1**: some mistakes may be more costly than others

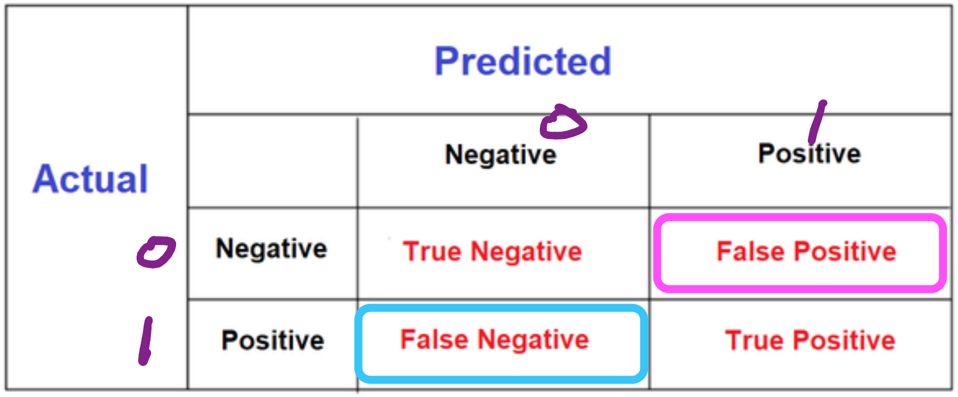
  We can tabulate the types of error and define a loss function

- **Problem 2**: It's hard to know the true labels of the run-time data

  We must separate the labeled data into a training set and test/validation set

Performance of a binary classifier

- A binary classifier can make two types of errors
  - False positive (**FP**)
  - False negative (**FN**)



		Predicted	
Actual		0 Negative	1 Positive
	0 Negative	True Negative	False Positive
	1 Positive	False Negative	True Positive

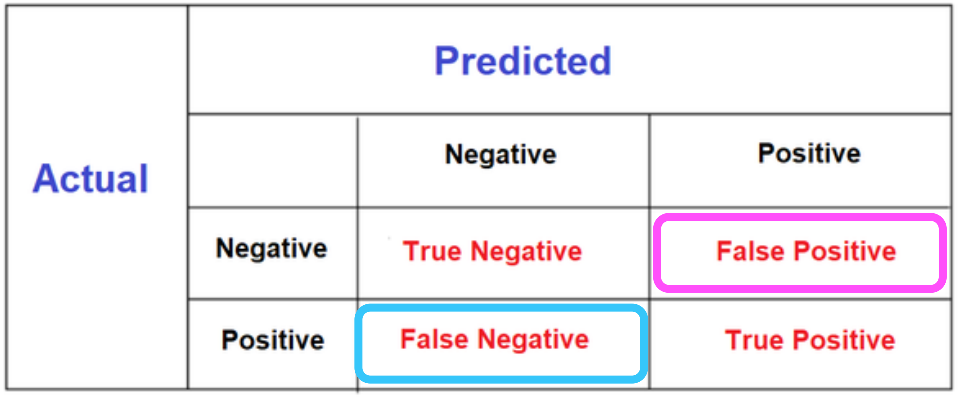
- Sometimes one type of error is more costly
  - Drug effect test
  - Crime detection

- We can tabulate the performance in a class confusion matrix

TP	FP
15	**3**
7	25
FN	TN

Performance of a binary classifier

- A loss function assigns costs to mistakes
- The 0-1 loss function treats FPs and FNs the same
  - Assigns loss 1 to every mistake
  - Assigns loss 0 to every correct decision



		Predicted	
Actual		Negative	Positive
	Negative	True Negative	False Positive
	Positive	False Negative	True Positive

- Under the 0-1 loss function
  - accuracy= $\dfrac{TP+TN}{TP+TN+FP+FN}$
- The baseline is 50% which we get by random decision.

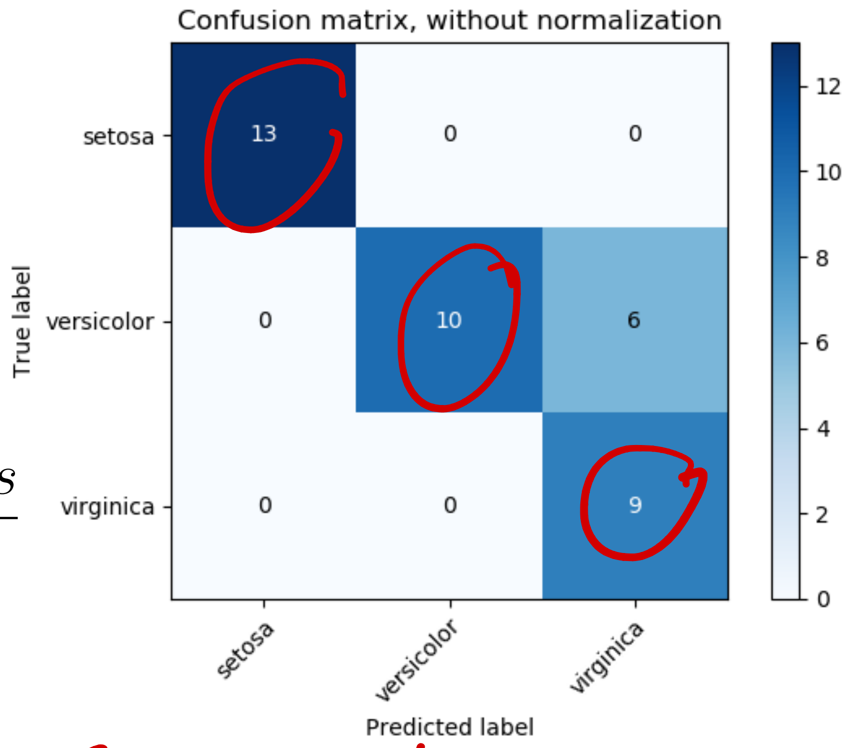
Performance of a multiclass classifier

- Assuming there are **c** classes:
- The class confusion matrix is c × c
- Under the 0-1 loss function

accuracy= $\dfrac{\textit{sum of diagonal terms}}{\textit{sum of all terms}}$

ie. in the right example, accuracy = 32/38=84%

- The baseline accuracy is 1/c.



$$\frac{c}{c^2} = \frac{1}{c}$$

Source: scikit-learn

Training set vs. validation/test set

- We expect a classifier to perform worse on run-time data
  - Sometimes it will perform much worse: an **overfitting** in training
  - An extreme case is: the classifier correctly labeled 100% when the input is in the training set, but otherwise makes a random guess
- To protect against overfitting, we separate training set from validation/test set
  - **Training set** for training the classifier
  - **Validation/test set** is for evaluating the performance
- It's common to reserve at least 10% of the data for testing

Cross-validation

- If we don't want to "waste" labeled data on validation, we can use **cross-validation** to see if our training method is sound.

n labeled data pts.

$(n-1) \to$ training 1 data pt $\to$ testing

- Split the labeled data into training and validation sets in multiple ways

testing

$$\boxed{\frac{N}{K}} \qquad N - \frac{N}{K}$$

training

- For each split (called a **fold**)
  - Train a classifier on the training set
  - Evaluate its accuracy on the validation set

$\sqcup \sqcup \sqcup \ldots$

k

- Average the accuracy to evaluate the training methodology

How many trained models I can have for the leave one out cross-validation?

If I have a data set that has 50 labeled data entries, how many leave-one-out validations I can have?

$\boxed{\text{A.}}$ 50

B. 49

C. 50*49

$\binom{50}{1} \rightarrow$ testing.

How many trained models I can have for the leave one out cross-validation?

If I have a data set that has 50 labeled data entries, how many leave-one-out validations I can have?

[✓ A. 50]

B. 49

C. 50*49

How many trained models can I have with this cross-validation?

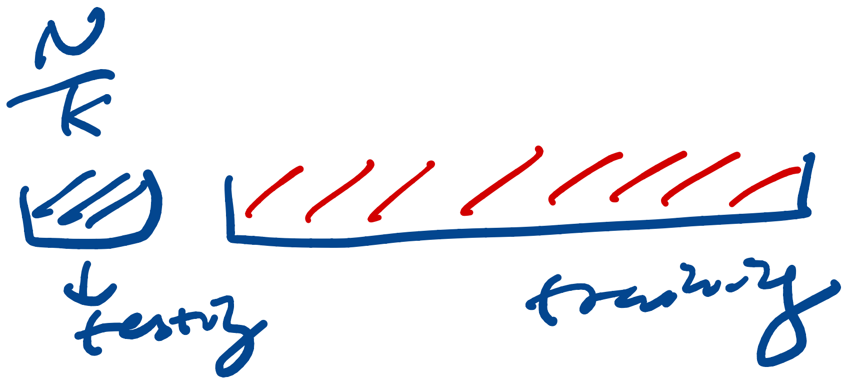
If I have a data set that has 51 labeled data entries, I divide them into three folds (17,17,17). How many trained models can I have? $\frac{N}{K}$

51

$K=3$

$\frac{N}{K}=17$

$\binom{51}{17}$



***The common practice of using fold is to divide the samples into equal sized k groups and reserve one of the group as the test data set.**

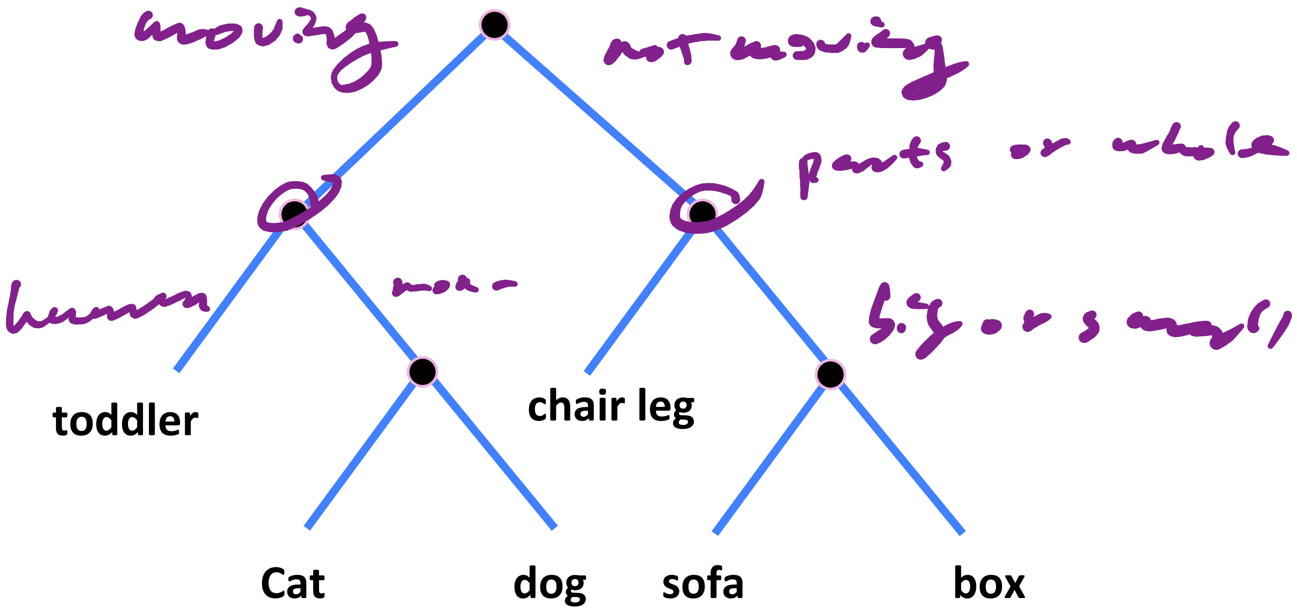
How many trained models can I have with this cross-validation?

If I have a data set that has 51 labeled data entries, I divide them into three folds (17,17,17). How many trained models can I have?

$$\boxed{\binom{51}{17}}$$

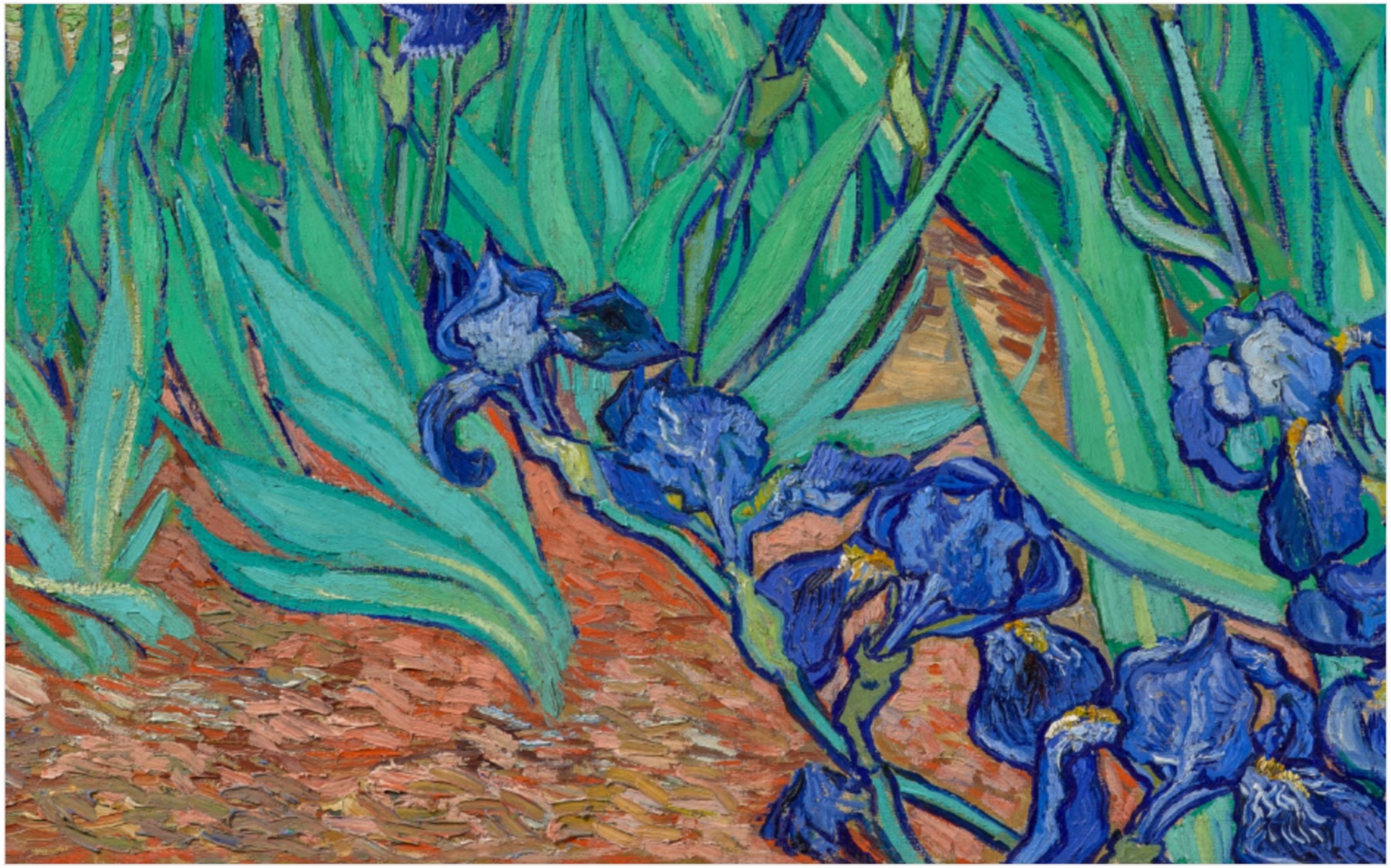
Decision tree: object classification

- The object classification **decision tree** can classify objects into multiple classes using sequence of simple tests. It will naturally grow into a tree.



Iris example: which type is this?

Irises



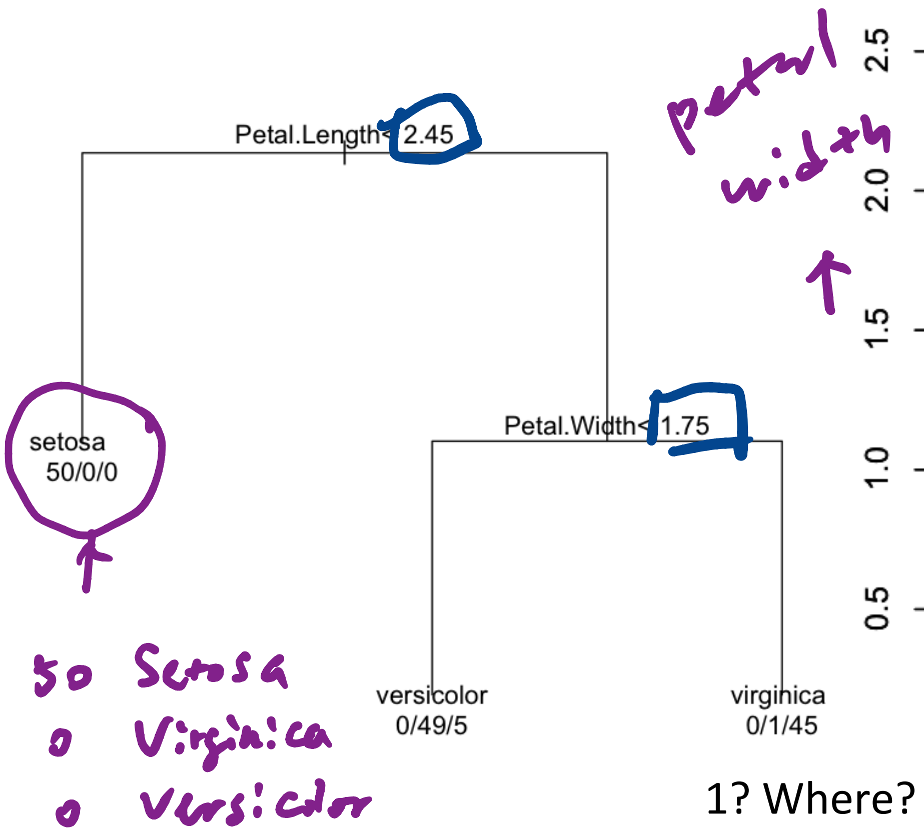
NO COPYRIGHT - UNITED STATES

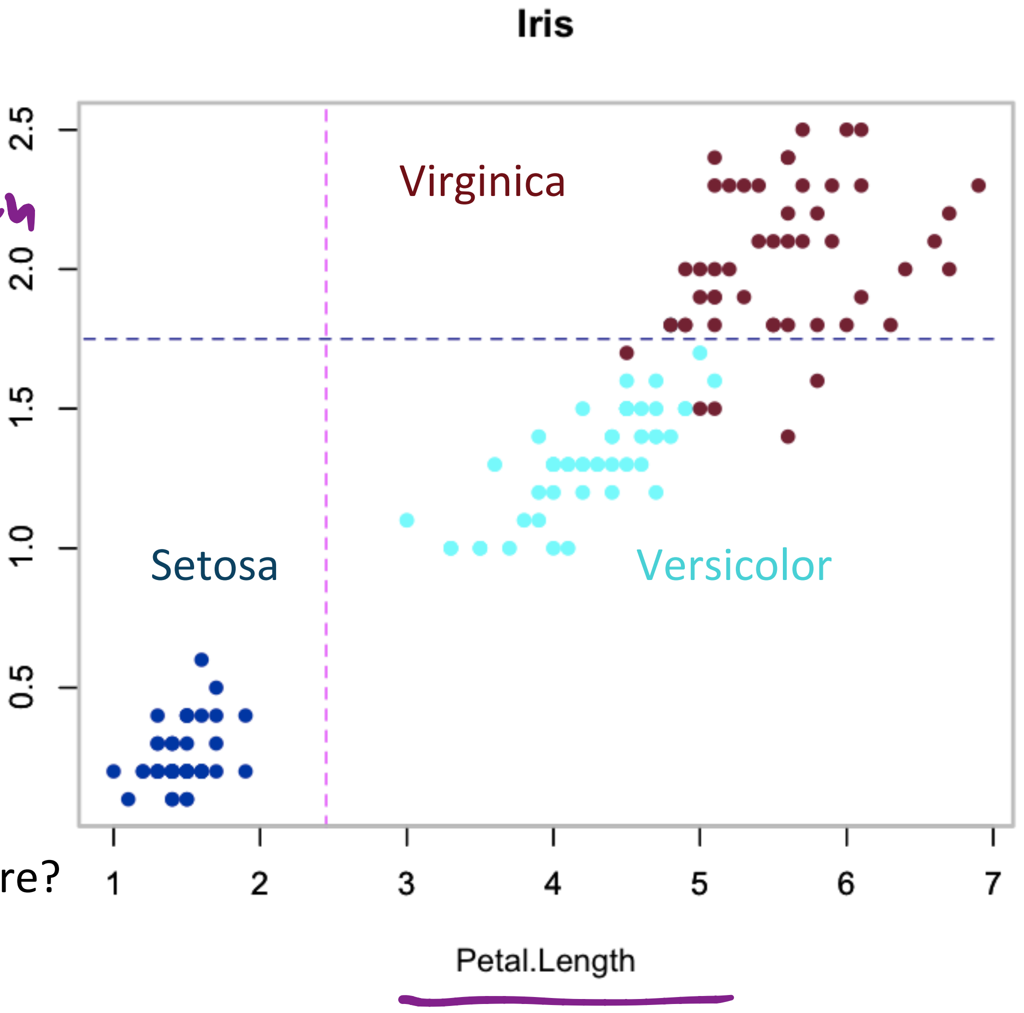
DOWNLOAD

This image is available for download, without charge, under the Getty's Open Content Program.

Training a decision tree: example

- The "Iris" data set





1? Where?

Q: What is accuracy of this decision tree given the confusion matrix ?

$$\begin{bmatrix} 50 & 0 & 0 \\ 0 & 49 & 5 \\ 0 & 1 & 45 \end{bmatrix}$$

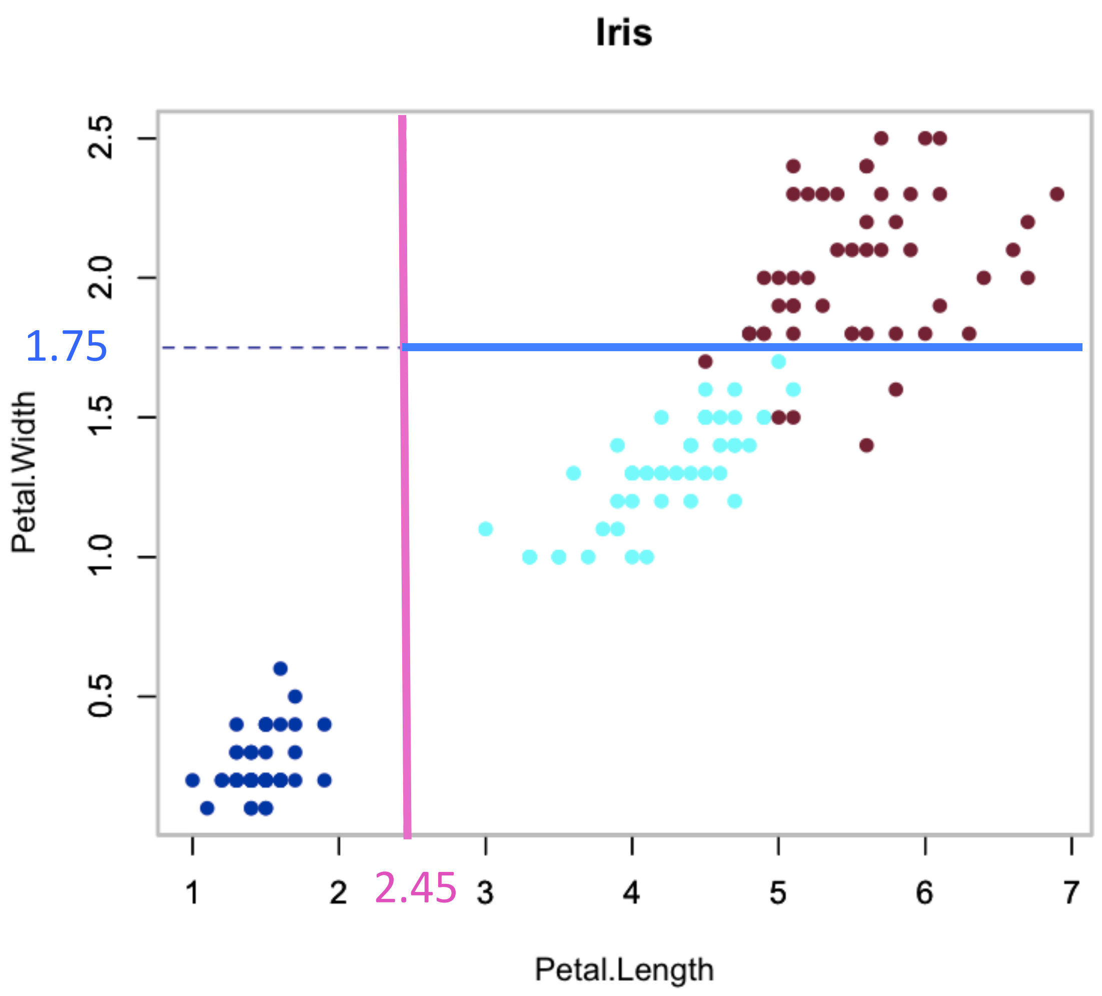
A. 6/150
[✓ B. 144/150]
C. 145/150

Q: What is accuracy of this decision tree given the confusion matrix ?

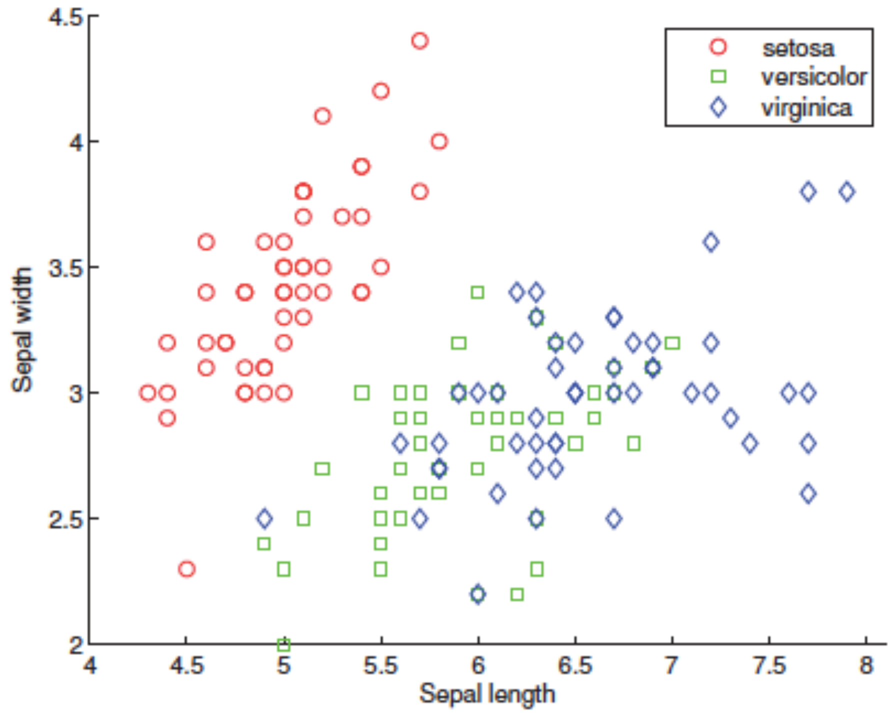
$$\begin{bmatrix} 50 & 0 & 0 \\ 0 & 49 & 5 \\ 0 & 1 & 45 \end{bmatrix}$$

A. 6/150
[✓ B. 144/150]
C. 145/150

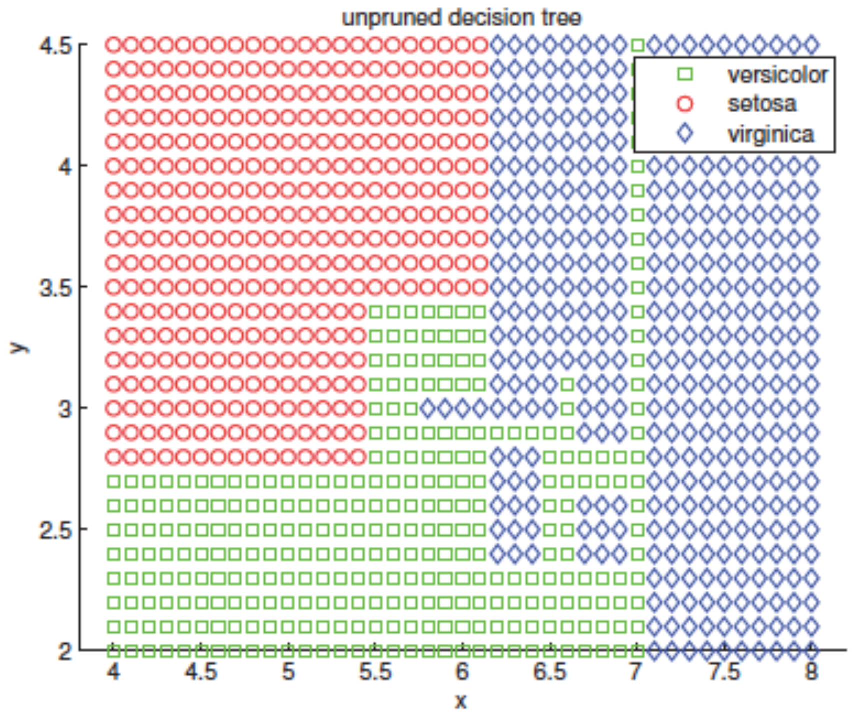
Decision Boundary



Another Decision Boundary



(a)



(b)

Credit: Kelvin Murphy, “Machine Learning: A Probabilistic Perspective”, 2012

Training a decision tree

- **Choose a dimension/feature and a split**

Training a decision tree

- Choose a dimension/feature and a split
- **Split the training Data into left- and right-child subsets D_l and D_r**

Training a decision tree

- Choose a dimension/feature and a split
- Split the training Data into left- and right-child subsets D_l and D_r
- **Repeat the two steps above recursively on each child**

Training a decision tree

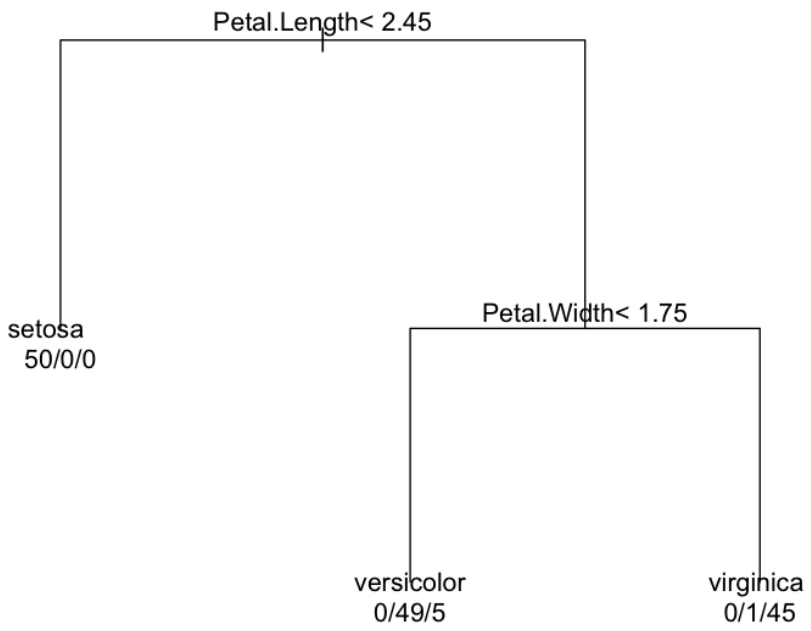
- Choose a dimension/feature and a split
- Split the training Data into left- and right-child subsets D_l and D_r
- Repeat the two steps above recursively on each child
- **Stop the recursion based on some conditions**

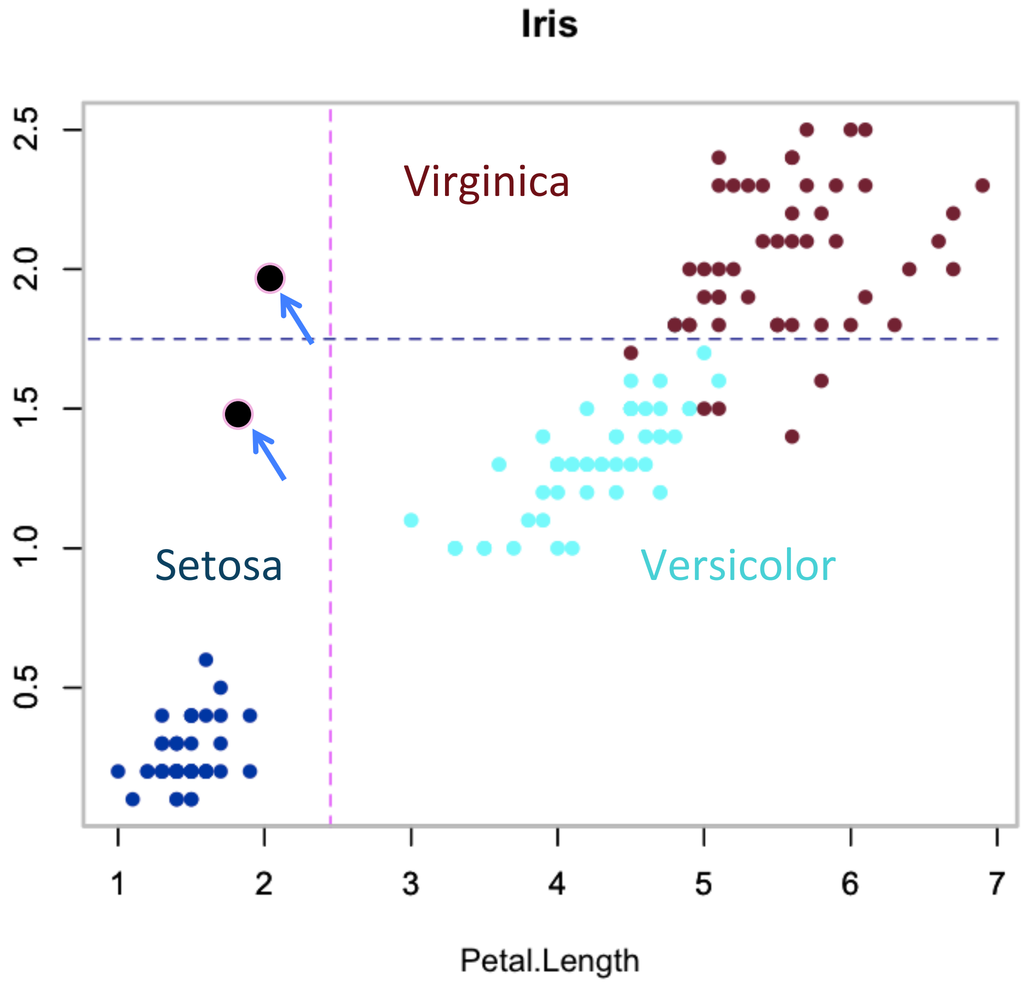
Training a decision tree

- Choose a dimension/feature and a split
- Split the training Data into left- and right-child subsets D_l and D_r
- Repeat the two steps above recursively on each child
- Stop the recursion based on some conditions
- **Label the leaves with class labels**

Classifying with a decision tree: example

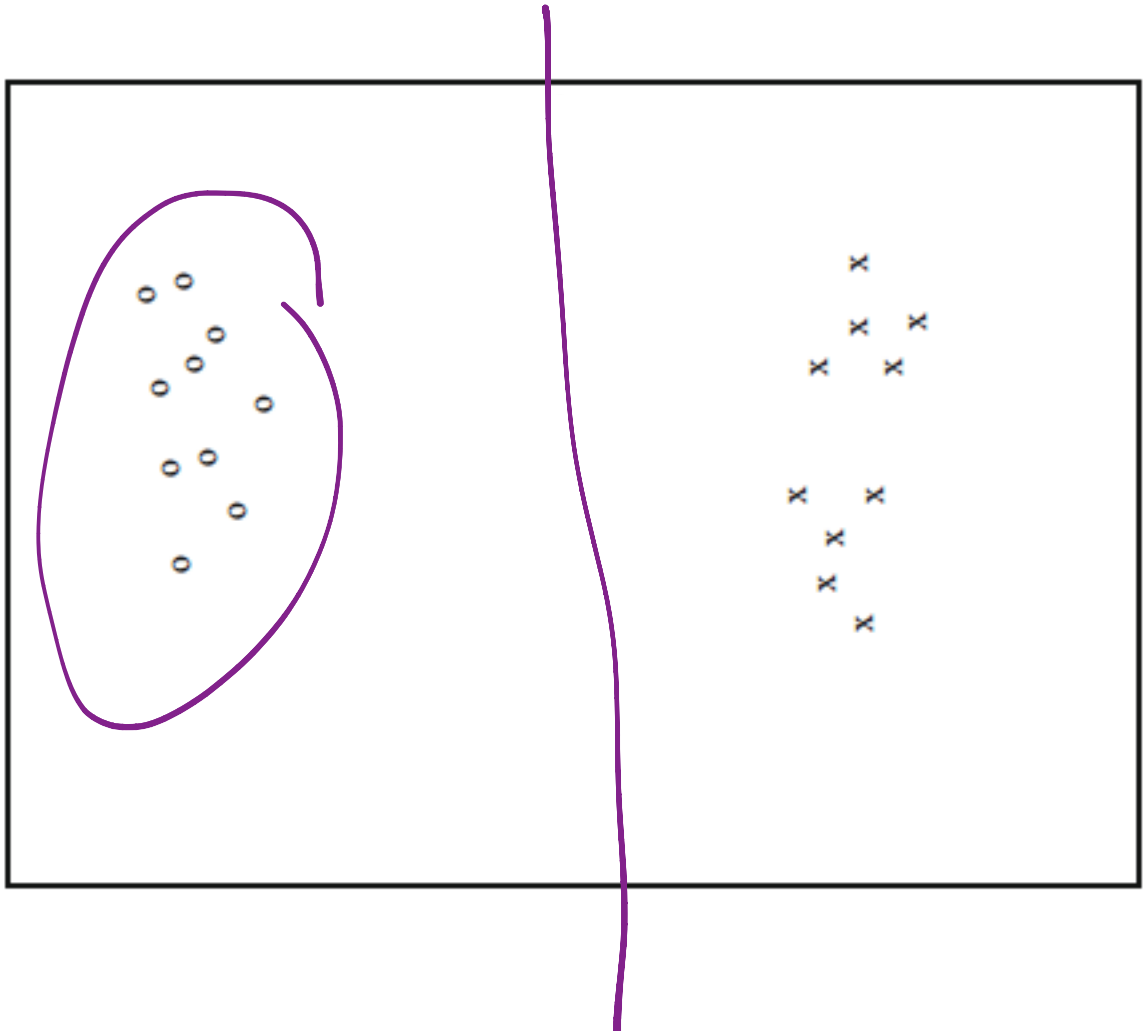
- The "Iris" data set





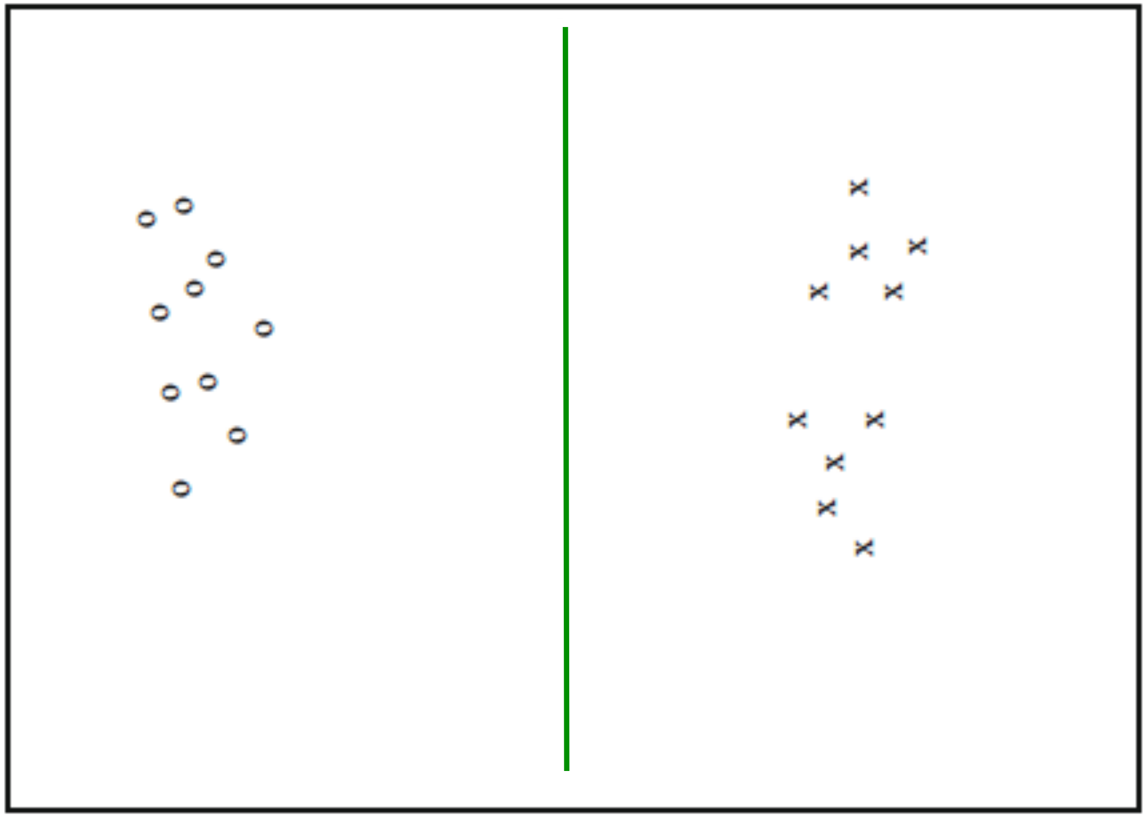
Choosing a split

- An informative split makes the subsets more concentrated and reduces ~~uncertainty~~ about class labels



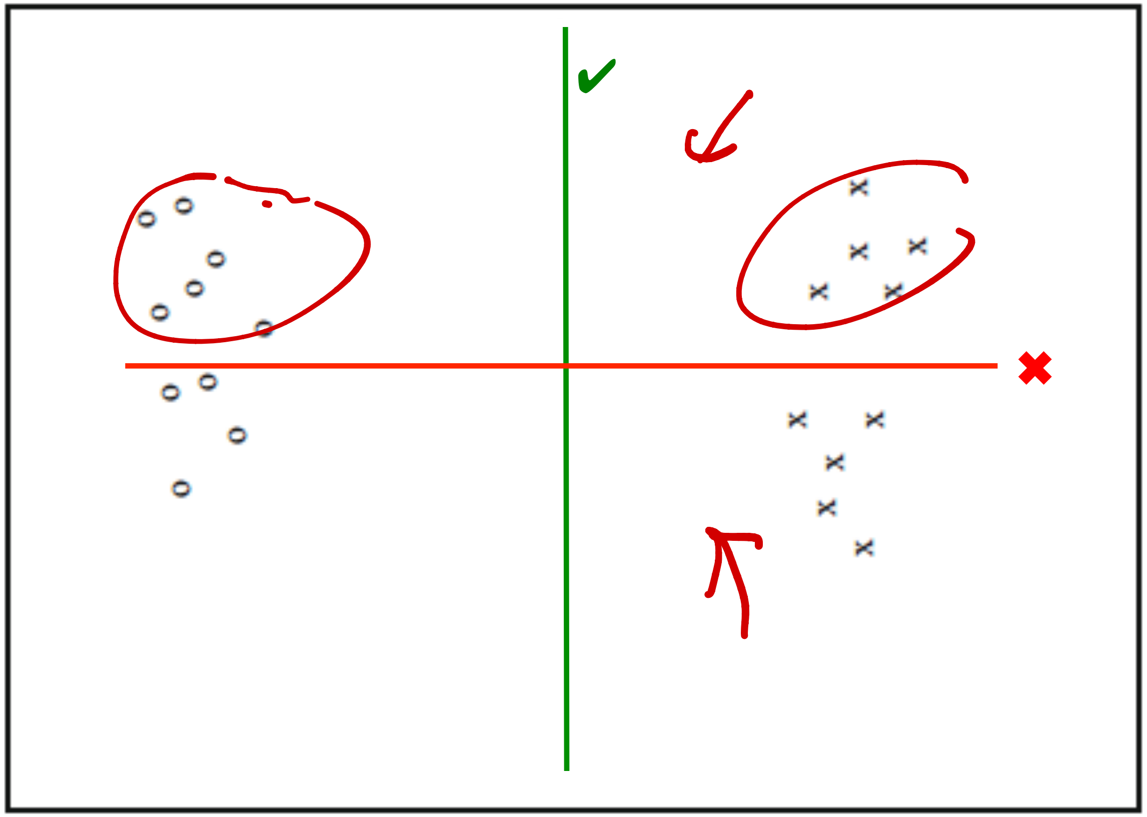
Choosing a split

- An informative split makes the subsets more concentrated and reduces uncertainty about class labels

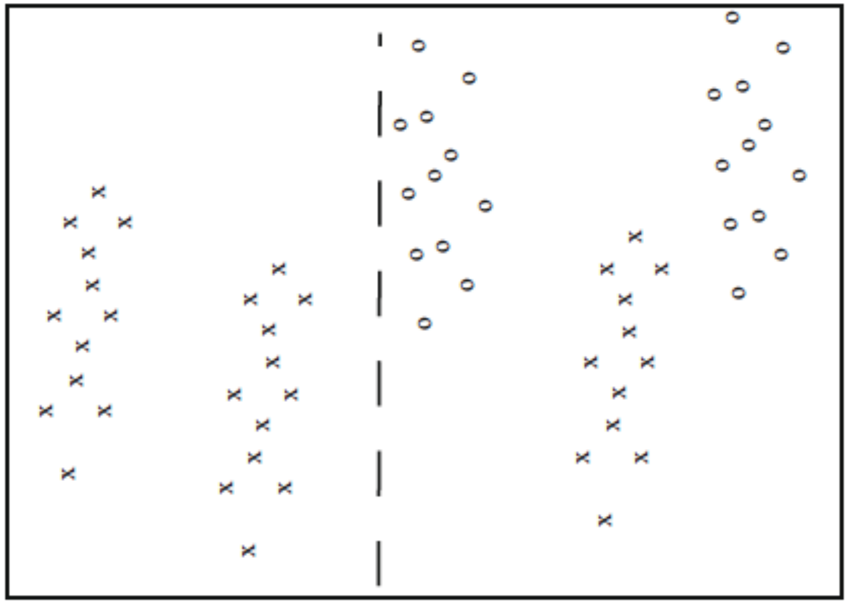


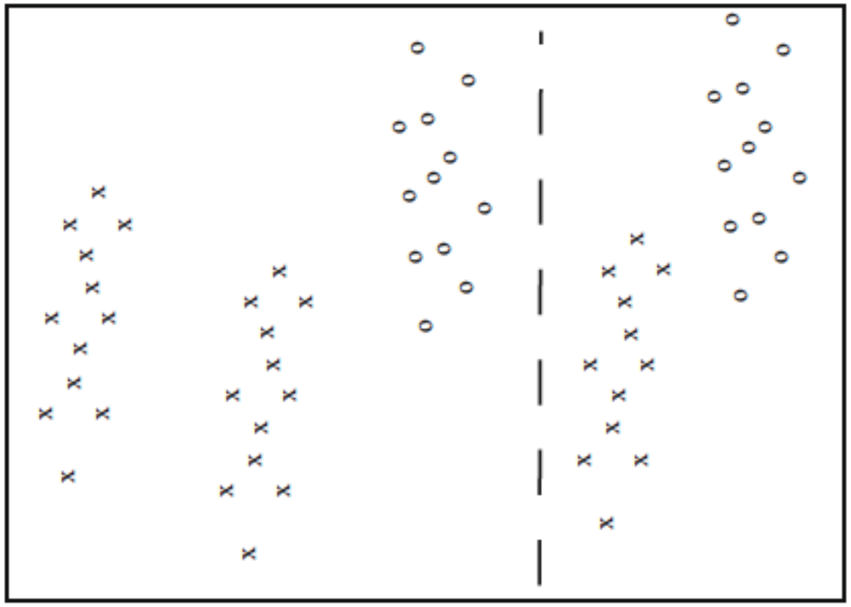
Choosing a split

- An informative split makes the subsets more concentrated and reduces uncertainty about class labels



Which is more informative?





Quantifying uncertainty using entropy

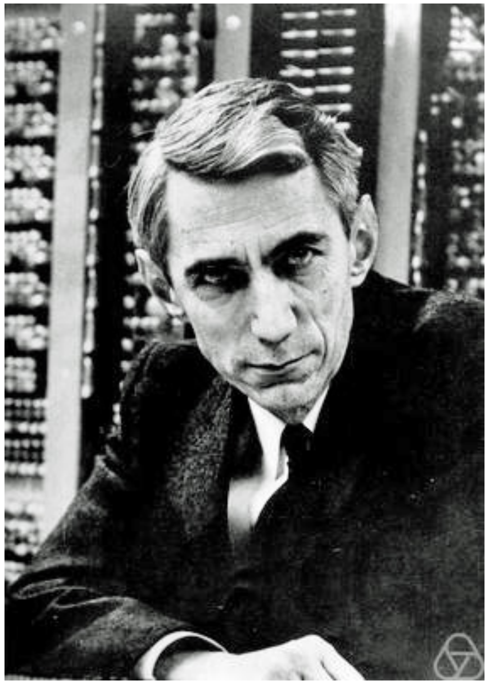
- We can measure uncertainty as the number of bits of information needed to distinguish between classes in a dataset (first introduced by Claude Shannon)
  - We need $\log_2 2 = 1$ bit to distinguish 2 equal classes
  - We need $\log_2 4 = 2$ bit to distinguish 4 equal classes

0
1

0 0
0 1
1 0
1 1

N

$\log_2 N$


Claude Shannon (1916-2001)

Quantifying uncertainty using entropy

- **Entropy** (Shannon entropy) is the measure of uncertainty for a general distribution
  - If class ***i*** contains a fraction ***P*(*i*)** of the data, we need $log_2 \frac{1}{P(i)}$ bits for that class
  - The entropy H(D) of a dataset is defined as the **weighted mean** of entropy for every class:

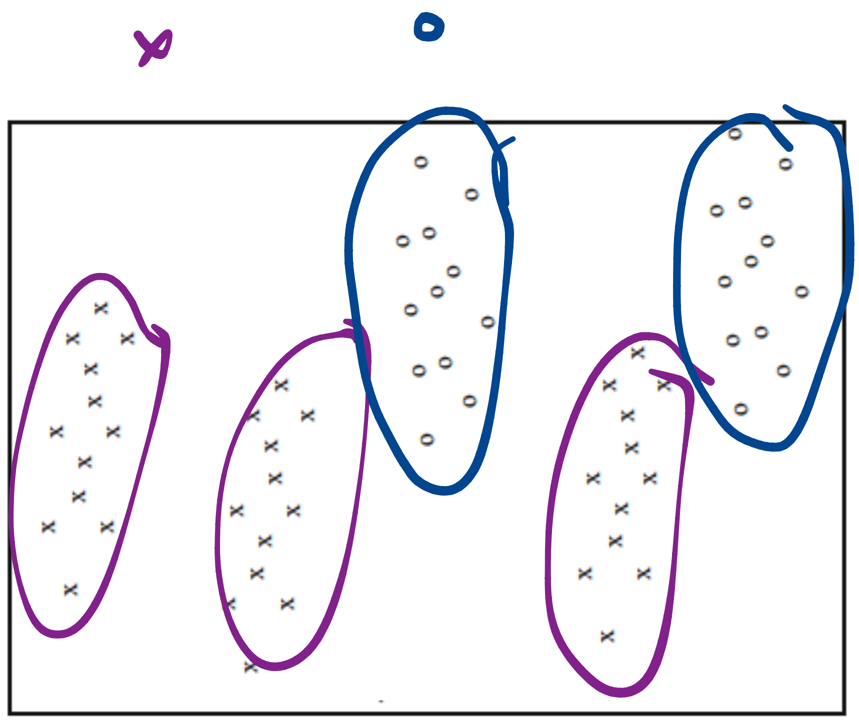
$$H(D) = \sum_{i=1}^{c} P(i) log_2 \frac{1}{P(i)}$$

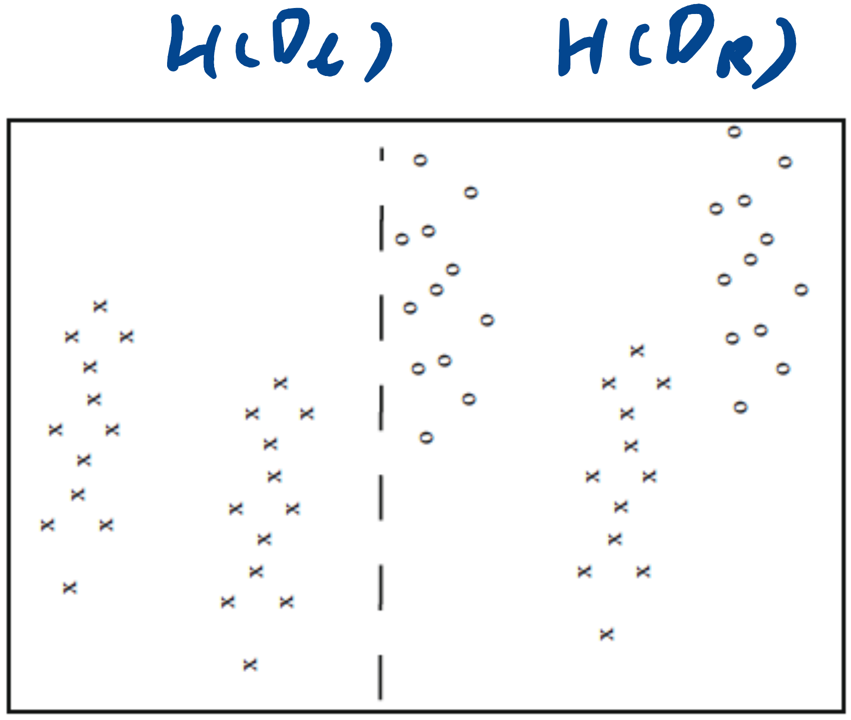
$\log_2 N$

$N = \frac{1}{P(i)}$

$c \rightarrow$ # of classes

Entropy: before the split





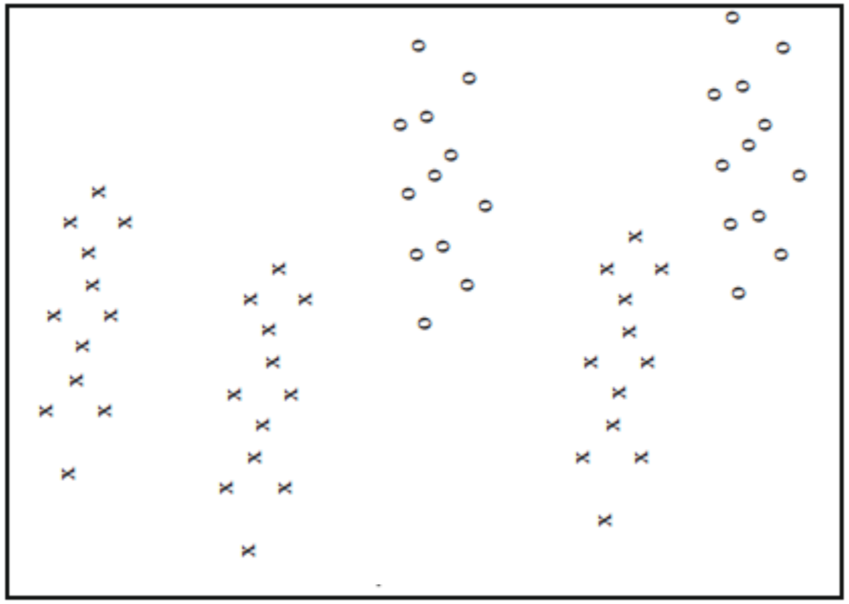
$$\underline{H(D)} = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}$$

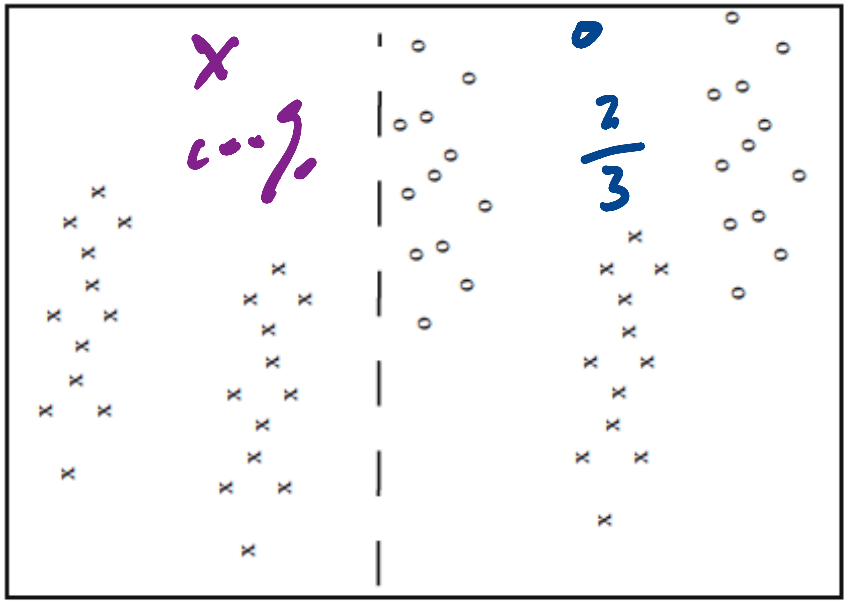
$$\underline{= 0.971\ bits}$$

$\frac{3}{5} \cdot \log_2 \frac{1}{p}$

$\log_2 \frac{1}{p} = -\log_2 p$

Entropy: examples



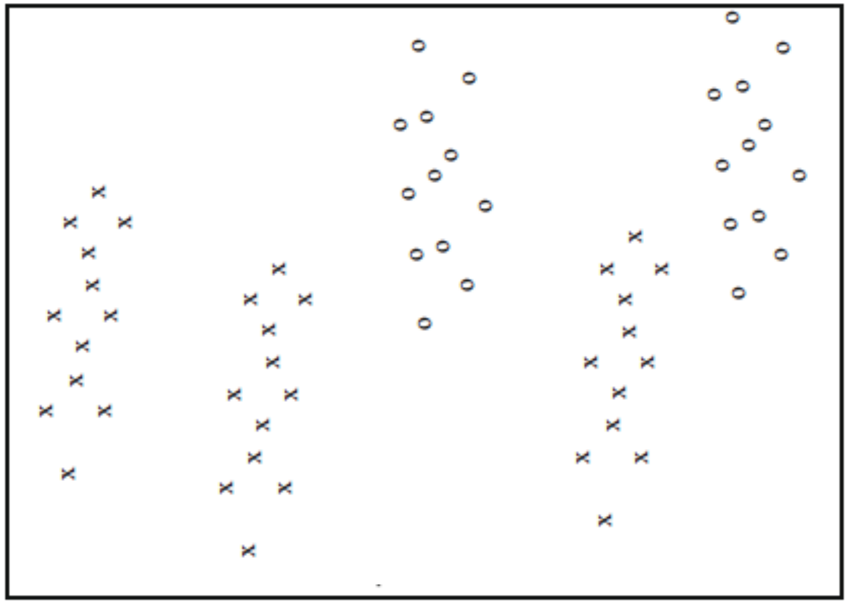


$$H(D) = -\frac{3}{5} log_2 \frac{3}{5} - \frac{2}{5} log_2 \frac{2}{5}$$

$$= 0.971 \; bits$$

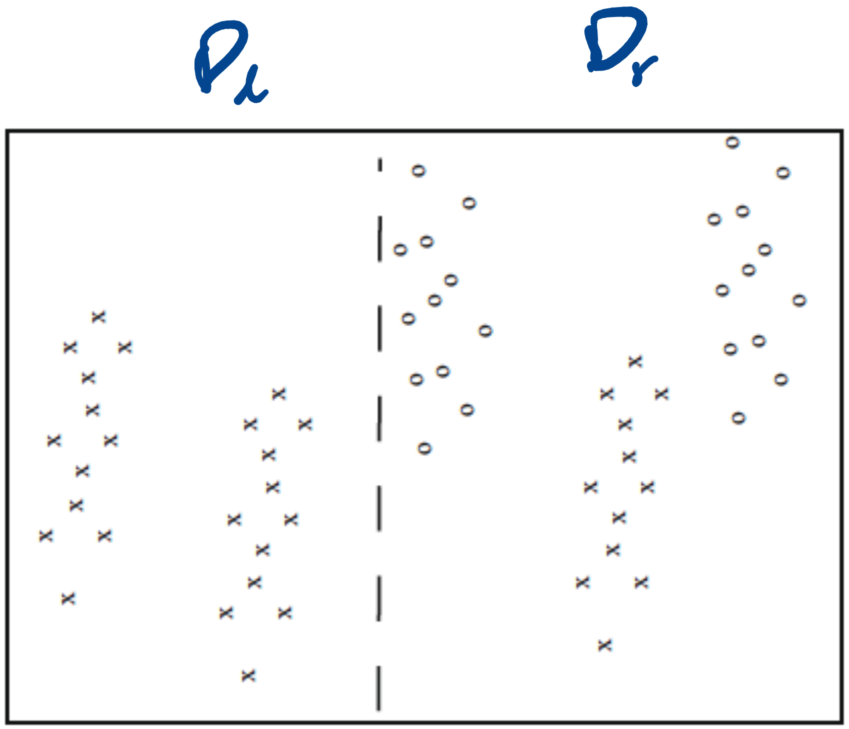
$$\underline{H(D_l)} = -1 \; log_2 1 = 0 \; bits$$

Entropy: examples



$$H(D) = -\frac{3}{5}log_2\frac{3}{5} - \frac{2}{5}log_2\frac{2}{5}$$

$$= 0.971\ bits$$



$$\underline{H(D_l)} = -1\ log_2 1 = \underline{0\ bits}$$

$$\underline{H(D_r)} = -\frac{1}{3}log_2\frac{1}{3} - \frac{2}{3}log_2\frac{2}{3}$$

$$\underline{= 0.918\ bits}$$

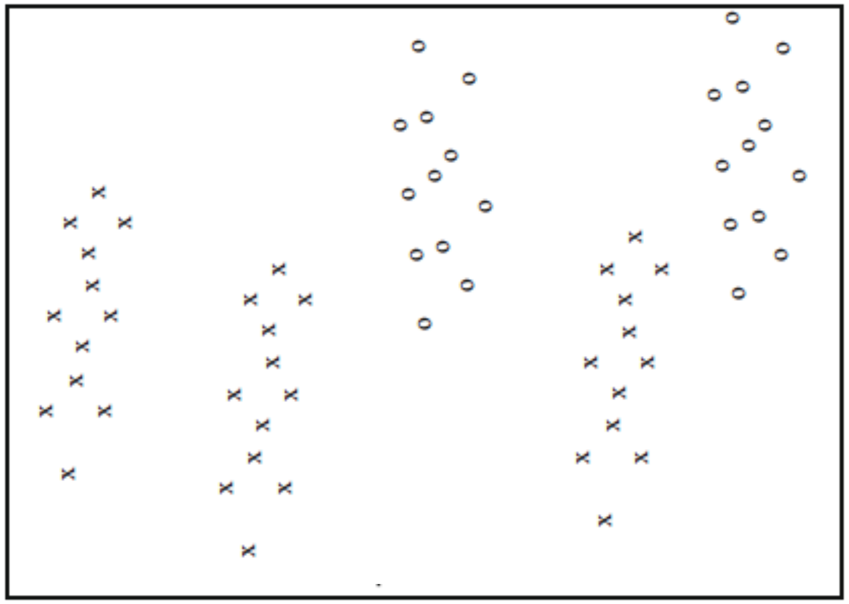
Information gain of a split

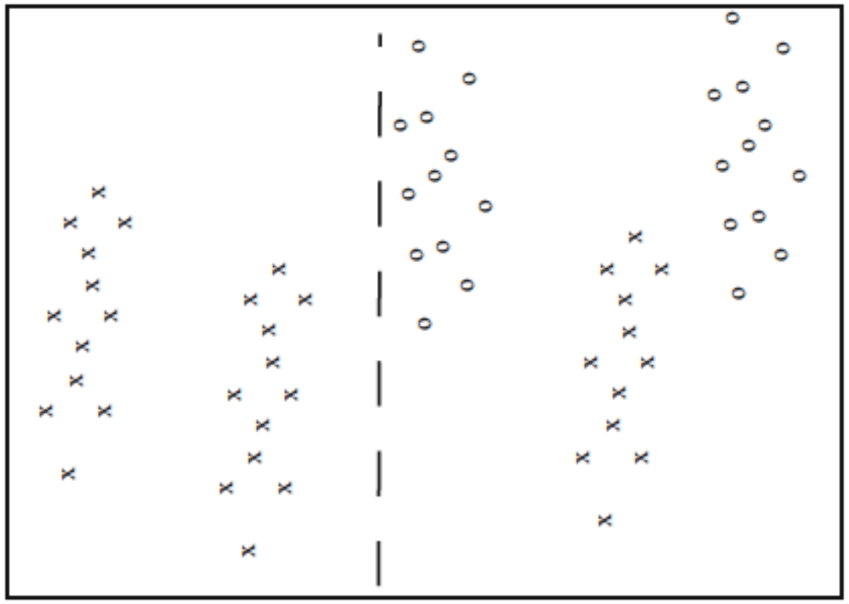
- The information gain of a split is the amount of entropy that was reduced on average after the split

$$I = \boxed{H(D)} - \left(\boxed{\frac{N_{Dl}}{N_D}} \underline{H(D_l)} + \frac{\boxed{N_{Dr}}}{\boxed{N_D}} \underline{H(D_r)}\right)$$

- where
  - N_D is the number of items in the dataset *D*
  - N_{Dl} is the number of items in the left-child dataset D_l
  - N_{Dr} is the number of items in the left-child dataset D_r

Information gain: examples



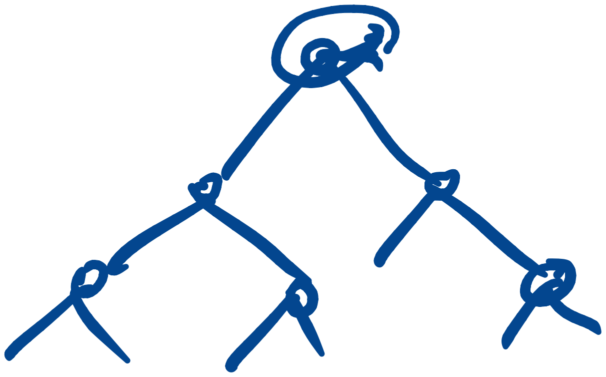


$$\boxed{I} = \underline{H(D)} - (\frac{N_{Dl}}{N_D}H(D_l) + \frac{N_{Dr}}{N_D}H(D_r))$$
$$= 0.971 - (\frac{24}{60} \times 0 + \frac{36}{60} \times 0.918)$$
$$= \underline{0.420\ bits}$$

uncertainty is reduced !

Q. Is the splitting method global optimum?

A. Yes
B. No



Decision for the lowest entropy is made at each node locally for the data at that point and feature

Q. Is the splitting method global optimum?

A. Yes

[✓ B. No]

Not necessarily globa

Assignments

- Read Chapter 11 of the textbook
- Next time: Decision tree, Random forest classifier
- Prepare for midterm2 exam
  - Lec 11-Lec 18, Chapter 6-10

Additional References

- Robert V. Hogg, Elliot A. Tanis and Dale L. Zimmerman. “Probability and Statistical Inference”
- Morris H. Degroot and Mark J. Schervish "Probability and Statistics”

See you next time

See You!