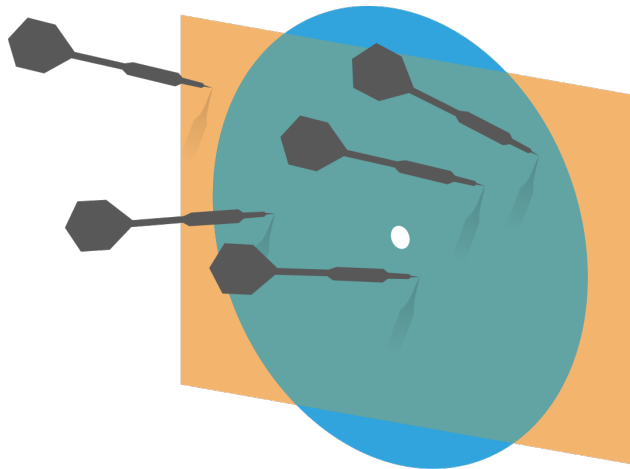


# Probability and Statistics for Computer Science



“...many problems are naturally  
classification problems” ---Prof.  
Forsyth

Credit: wikipedia

# Last time

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

# Objectives

- ✱ Demo of Principal Component Analysis
- ✱ Introduction to classification

# Reconstructing the data

- ✱ Given the projected data  $\mathbf{p}_{d \times n}$  and  $\text{mean}(\{\mathbf{x}\})$ , we can approximately reconstruct the original data

$$\hat{\mathbf{D}} = \mathbf{U}\mathbf{p} + \text{mean}(\{\mathbf{x}\})$$

- ✱ Each reconstructed data item  $\hat{\mathbf{D}}_i$  is a linear combination of the columns of  $\mathbf{U}$  weighted by  $\mathbf{p}_i$
- ✱ The columns of  $\mathbf{U}$  are the normalized eigenvectors of the  $\text{Covmat}(\{\mathbf{x}\})$  and are called the **principal components** of the data  $\{\mathbf{x}\}$

# End-to-end mean square error

- ✱ Each  $\mathbf{x}_i$  becomes  $\mathbf{r}_i$  by translation and rotation
- ✱ Each  $\mathbf{p}_i$  becomes  $\hat{\mathbf{x}}_i$  by the opposite rotation and translation

- ✱ Therefore the end to end mean square error is:

$$\frac{1}{N-1} \sum_i \|\hat{\mathbf{x}}_i - \mathbf{x}_i\|^2 = \frac{1}{N-1} \sum_i \|\mathbf{r}_i - \mathbf{p}_i\|^2 = \sum_{j=s+1}^d \lambda_j$$

- ✱  $\lambda_{s+1}, \dots, \lambda_d$  are the smallest d-s eigenvalues of the  $\text{Covmat}(\{\mathbf{x}\})$

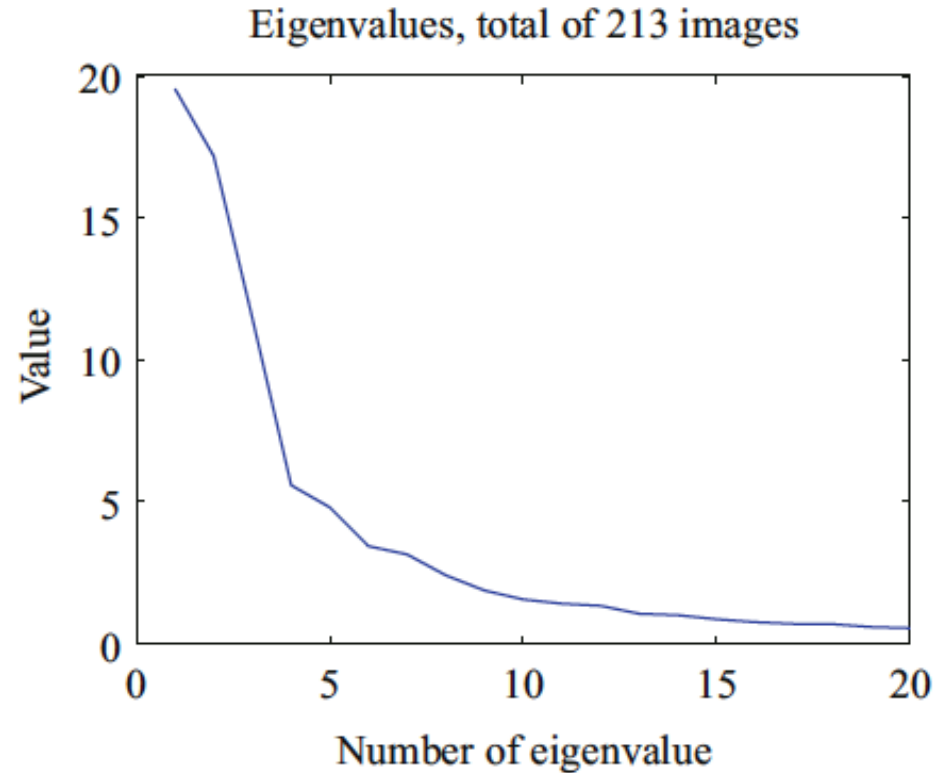
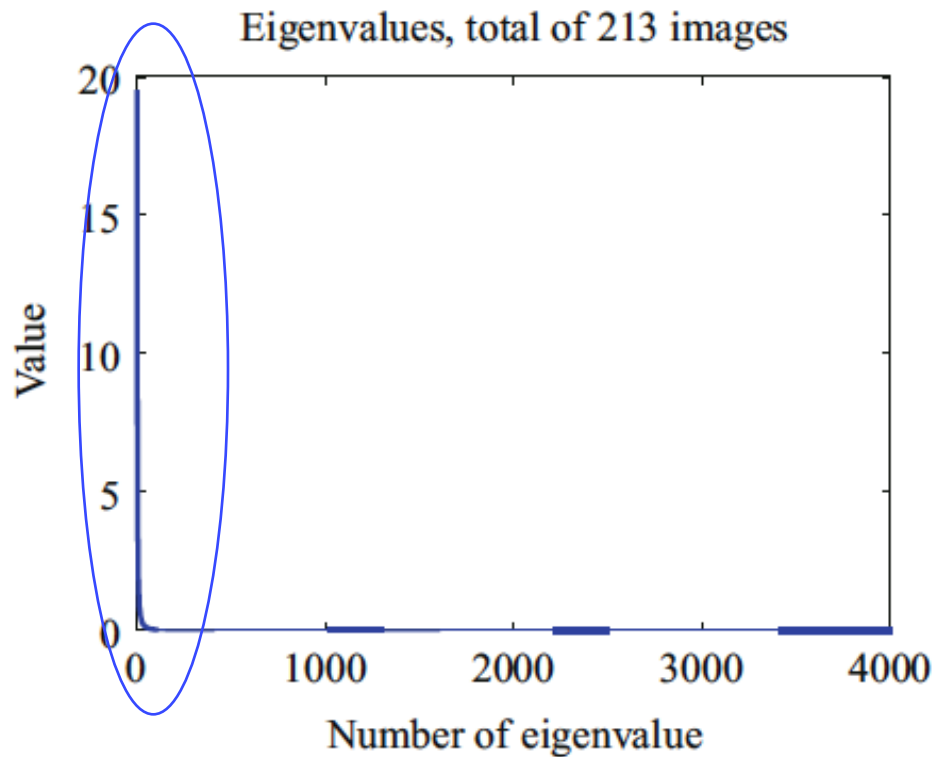
# PCA: Human face data

- ✱ The dataset consists of 213 images
- ✱ Each image is grayscale and has 64 by 64 resolution
- ✱ We can treat each image as a vector with dimension  $d = 4096$

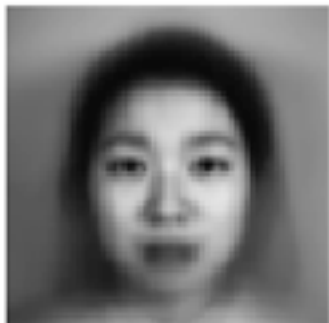


Credit: Prof. Forsyth

# How quickly the eigenvalues decrease?

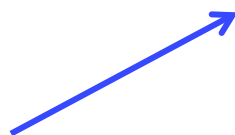


# What do the principal components of the images look like?



Mean image

The first 16  
principal  
components  
arranged into  
images





# Reconstruction of the image



**The original**

1<sup>st</sup> row show the reconstructions using  
some number of principal components  
2<sup>nd</sup> row show the corresponding errors

Mean

1

5

10

20

50

100



Credit: Prof. Forsyth

# 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

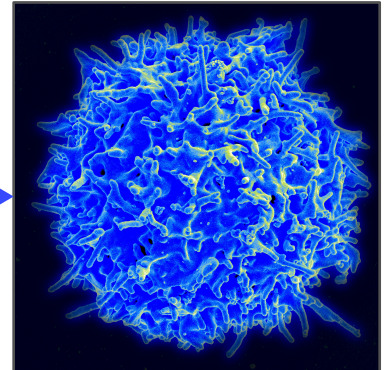
## Q. Which of these is NOT true?

- 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

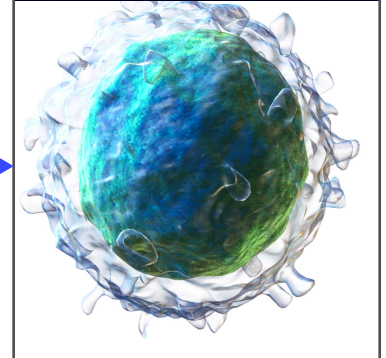
# Demo: PCA of Immune Cell Data

- ✱ 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

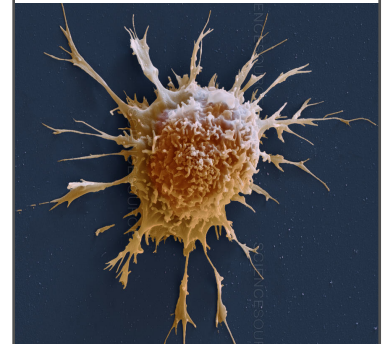
T cells



B cells

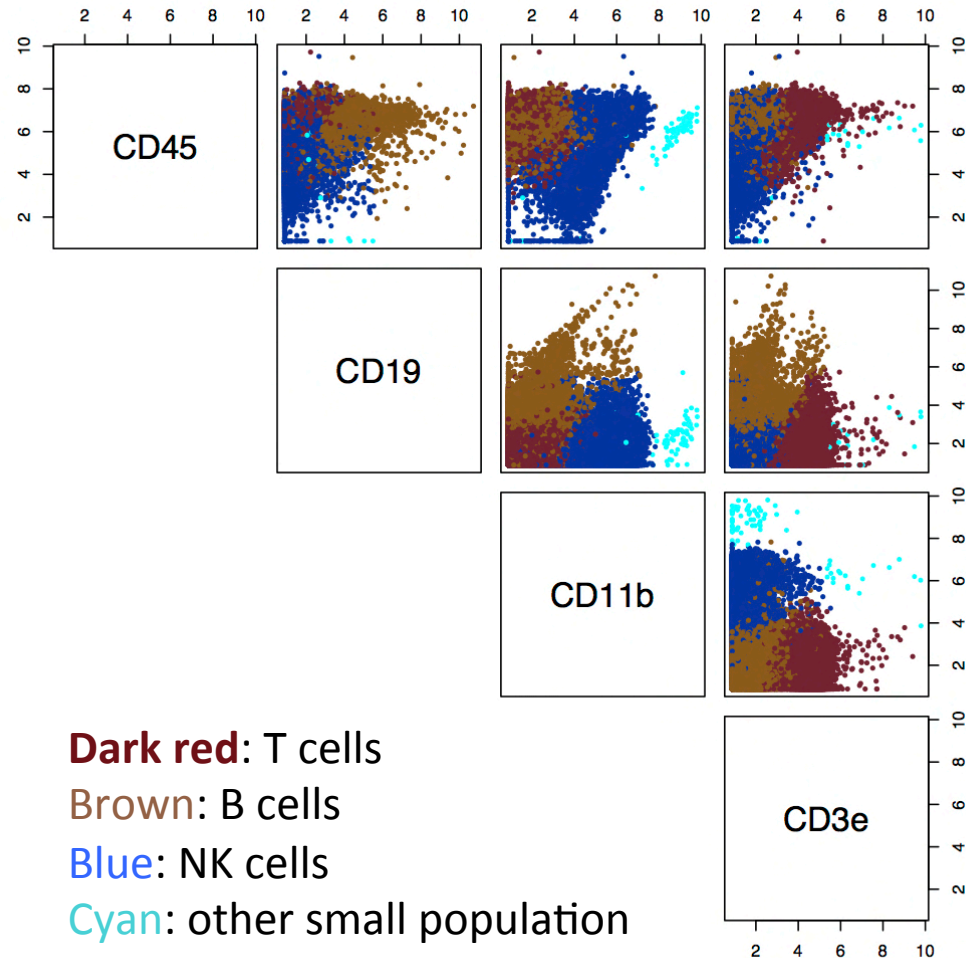


Natural killer cells



# Scatter matrix of Immune Cells

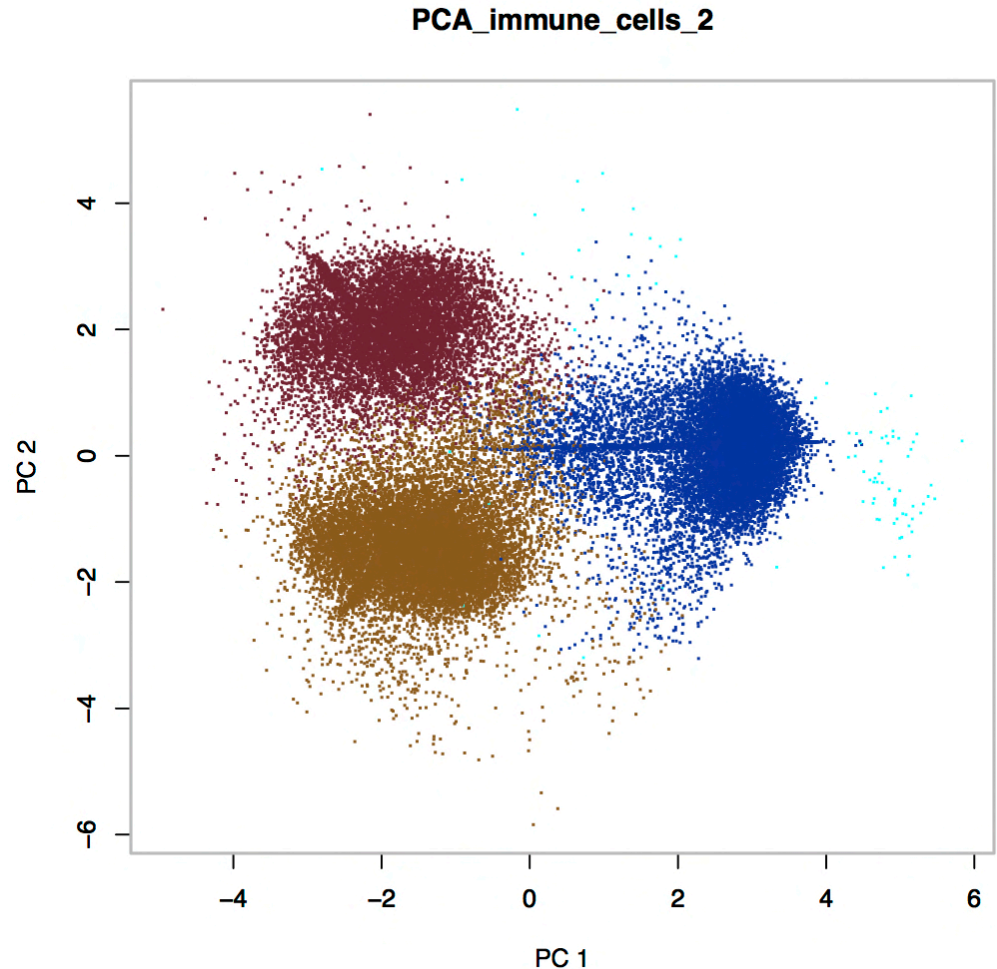
- ✱ There are 38816 white blood immune cells from a mouse sample
- ✱ Each immune cell has 40+ features/components
- ✱ Four features are used as illustration.
- ✱ There are at least 3 cell types involved



# PCA of Immune Cells

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

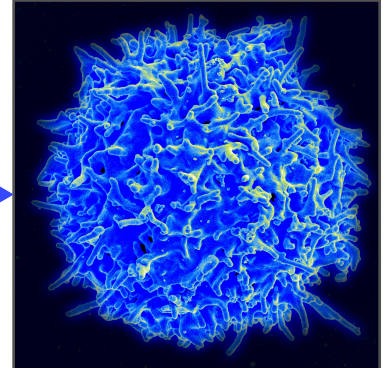
Eigenvectors
$vector
      [,1]  [,2]  [,3]  [,4]
[1,] 0.2476698 0.00801294 -0.6822740
0.6878210
[2,] 0.3389872 -0.72010997 -0.3691532
-0.4798492
[3,] -0.8298232 0.01550840 -0.5156117
-0.2128324
[4,] 0.3676152 0.69364033 -0.3638306
-0.5013477
```



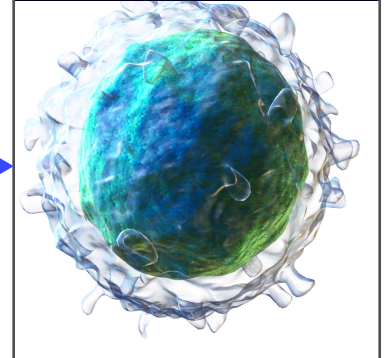
# More features used

- ✱ There are 38816 white blood immune cells from a mouse sample
- ✱ Each immune cell has **42 features/components**
- ✱ There are at least 3 cell types involved

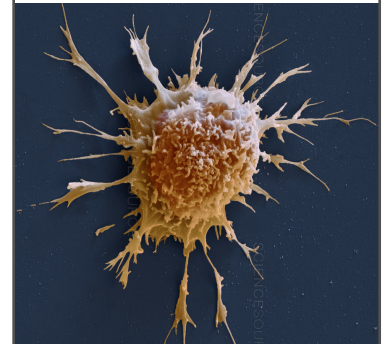
T cells



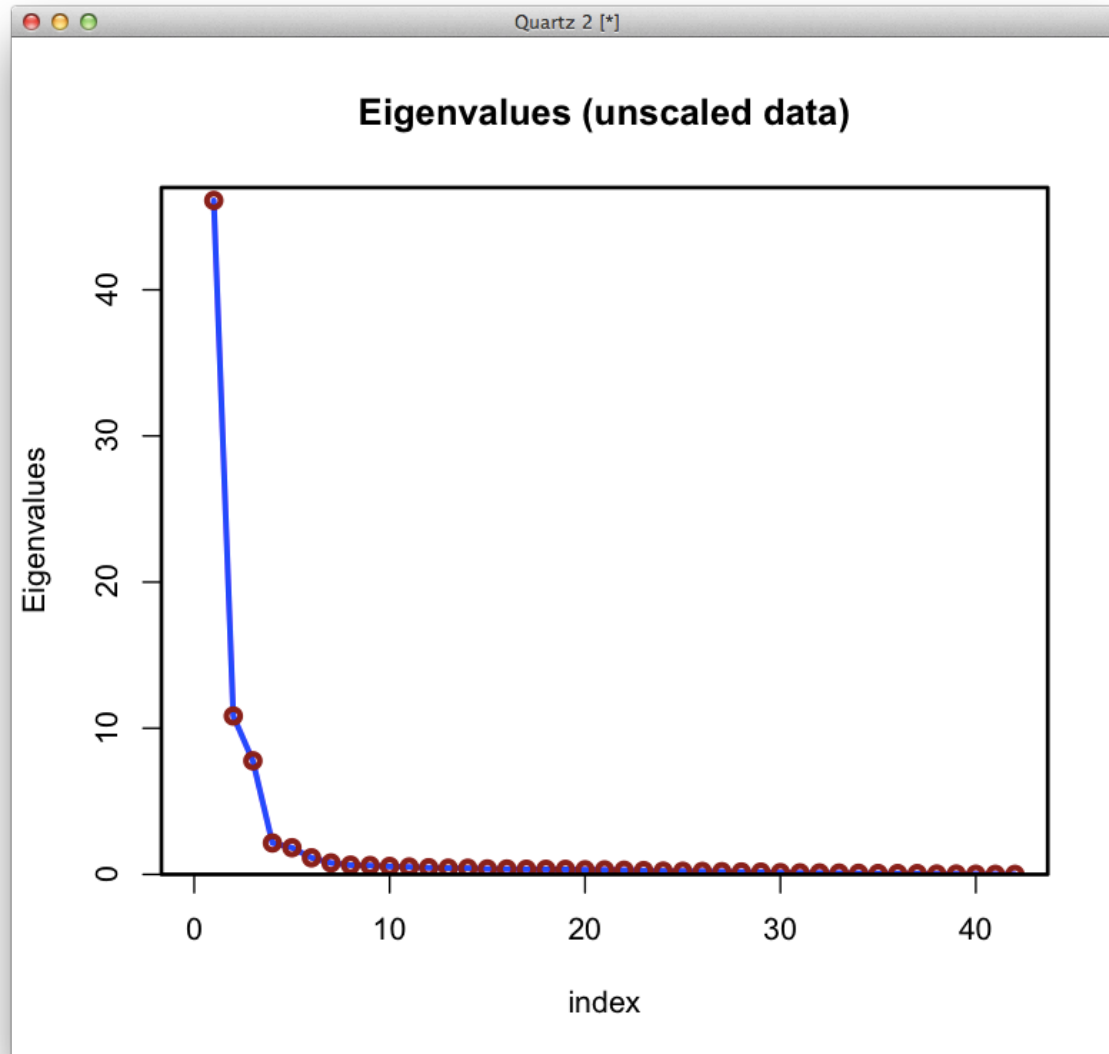
B cells



Natural killer cells



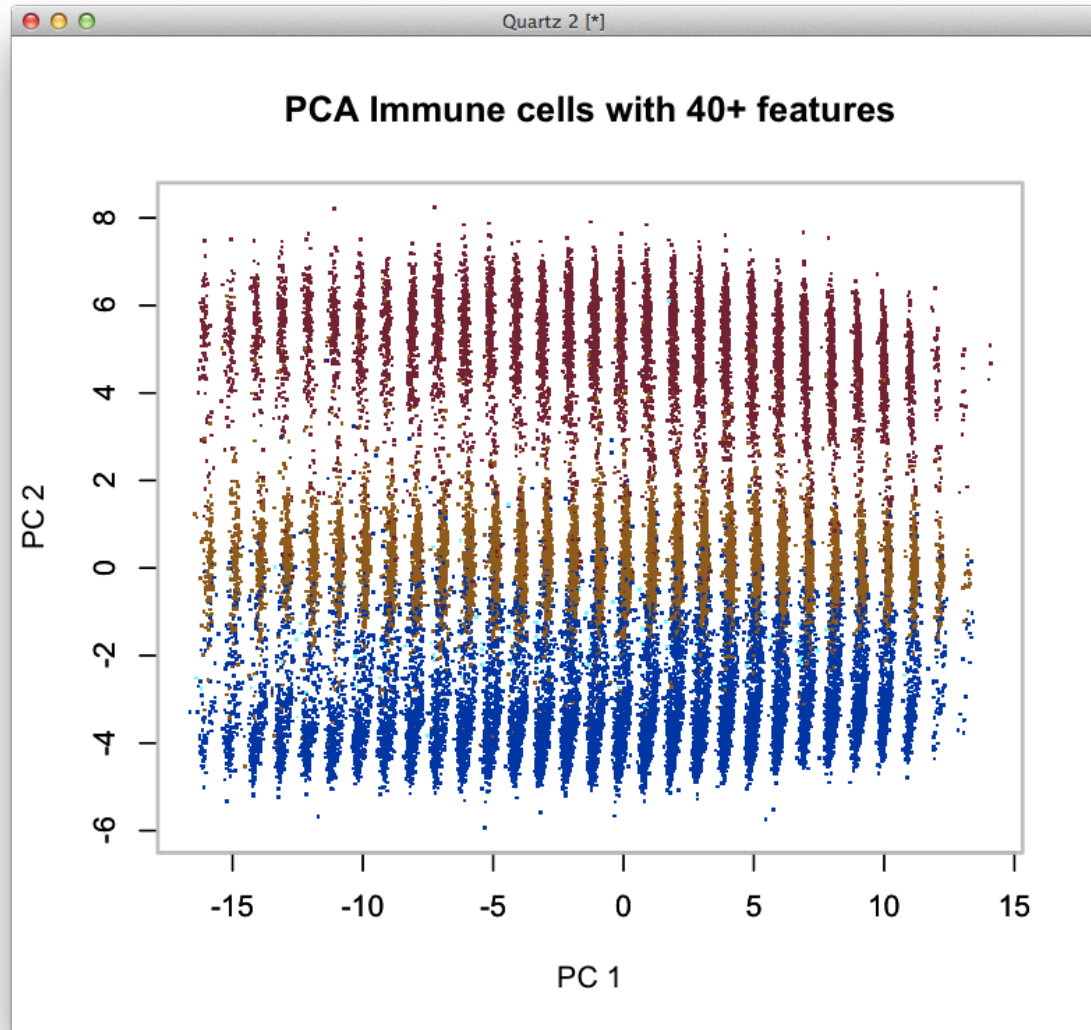
# Eigenvalues of the covariance matrix



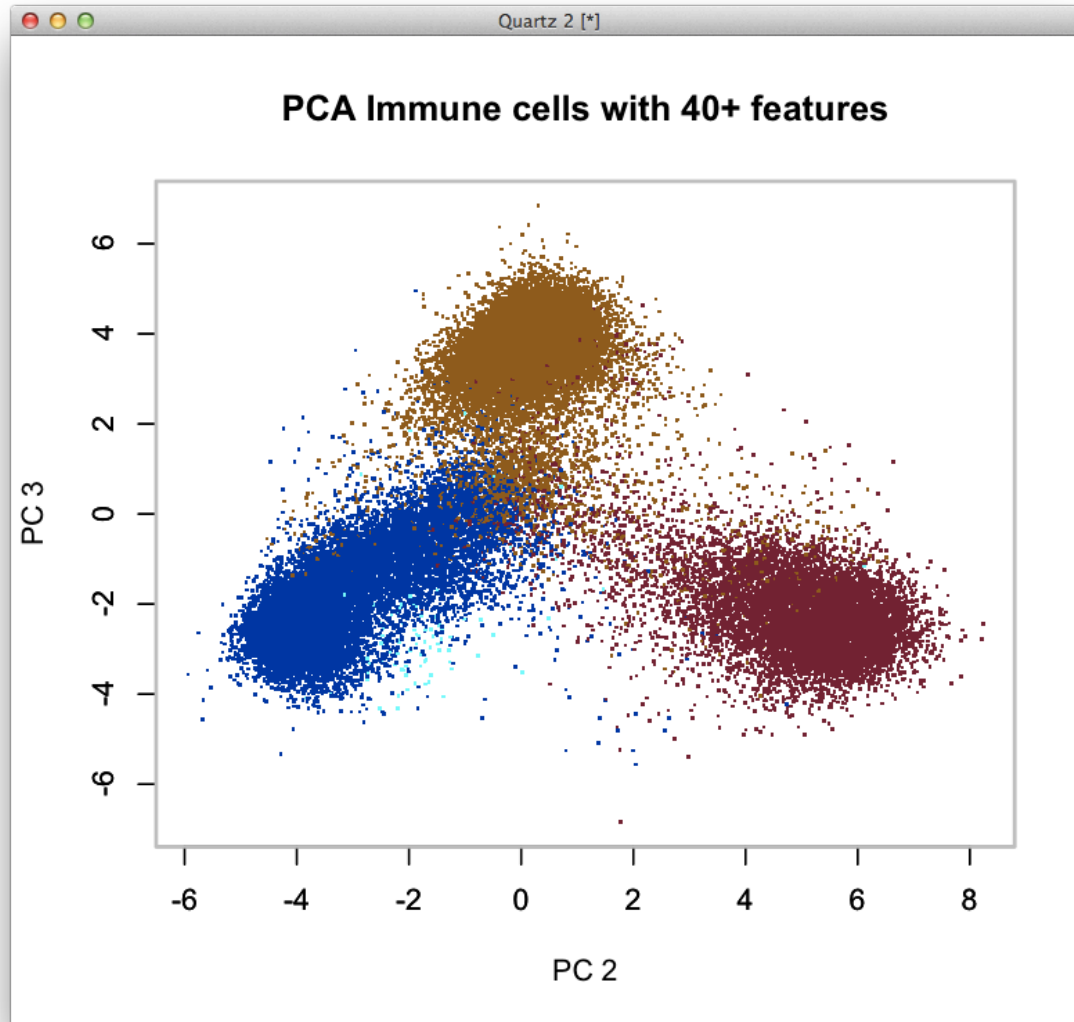


# Large variance doesn't mean important pattern

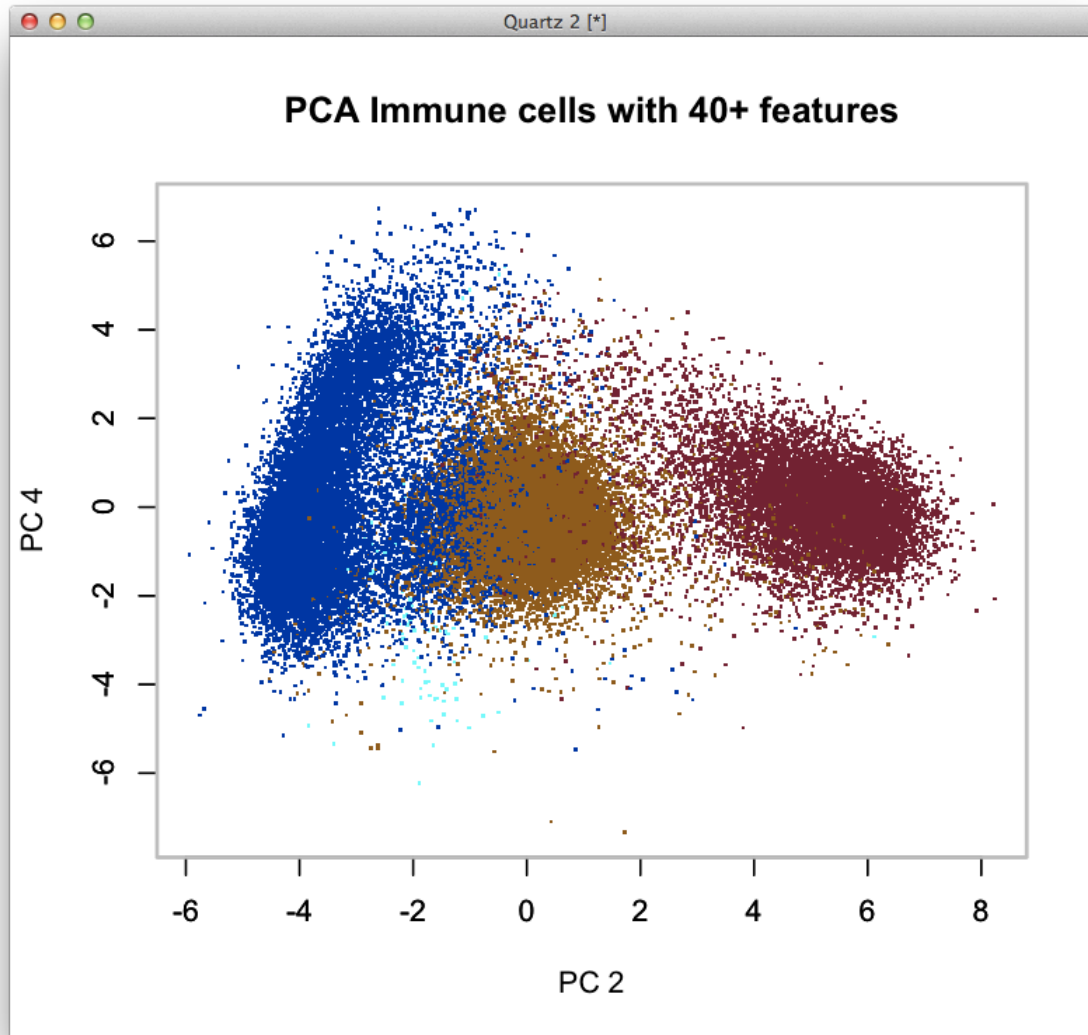
Principal component 1 is just cell length



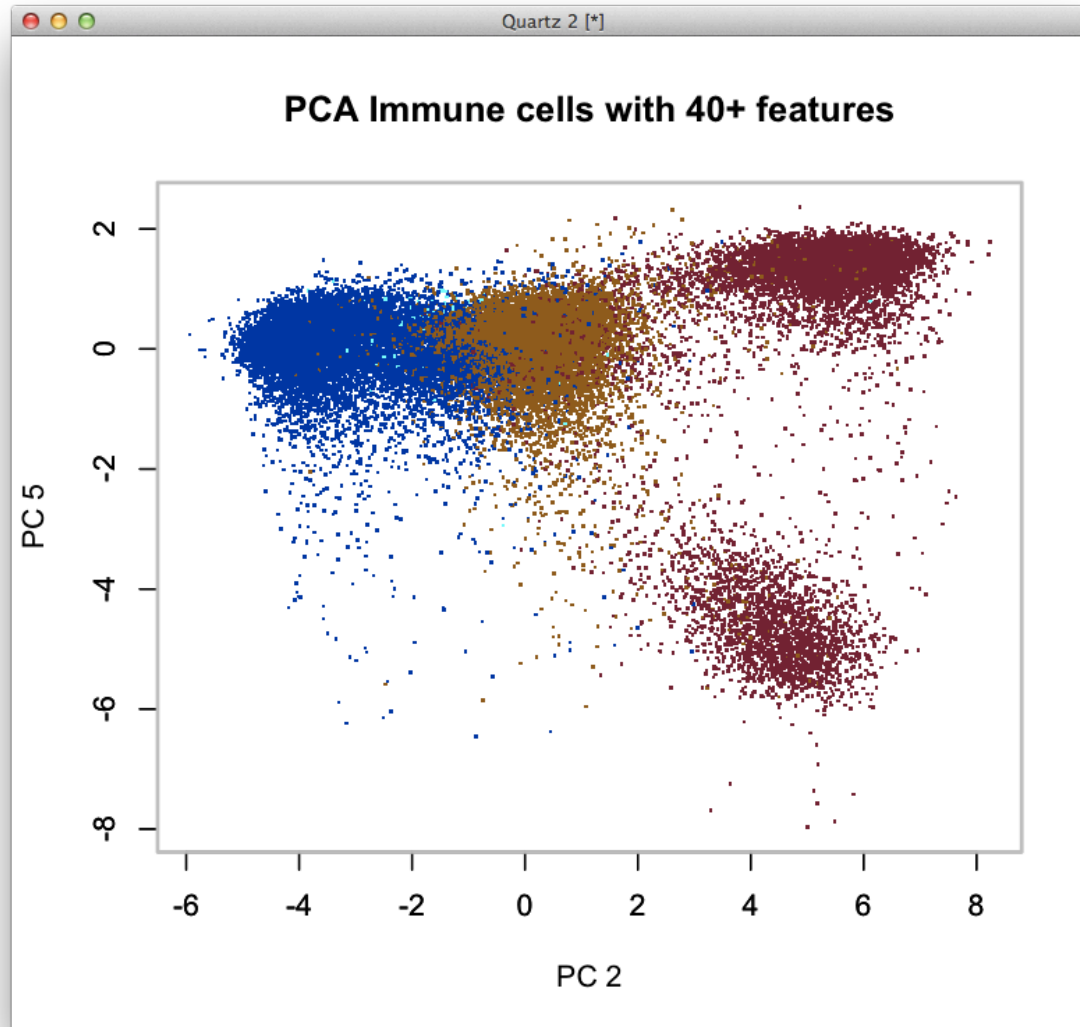
# Principal component 2 and 3 show different cell types



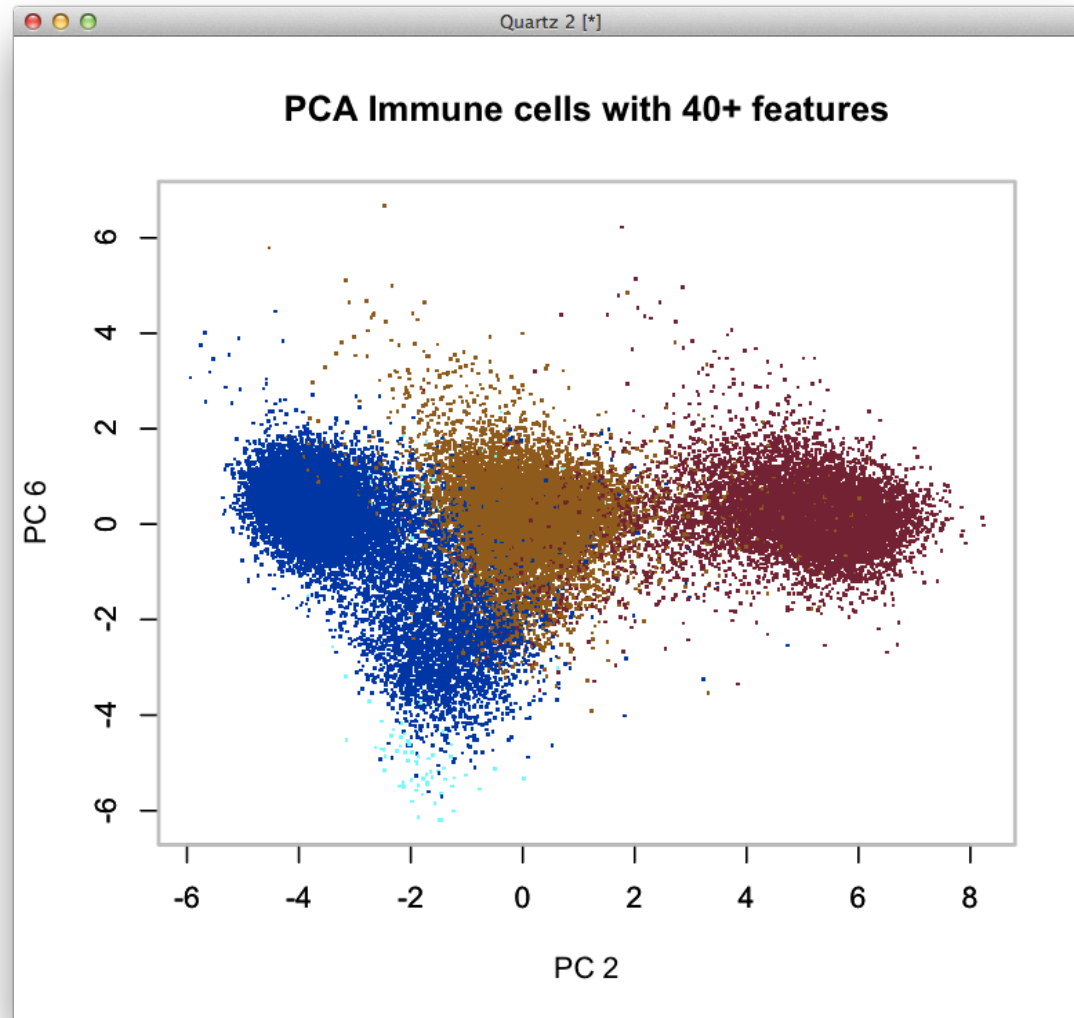
# Principal component 4 is not very informative



# Principal component 5 is interesting



# Principal component 6 is interesting

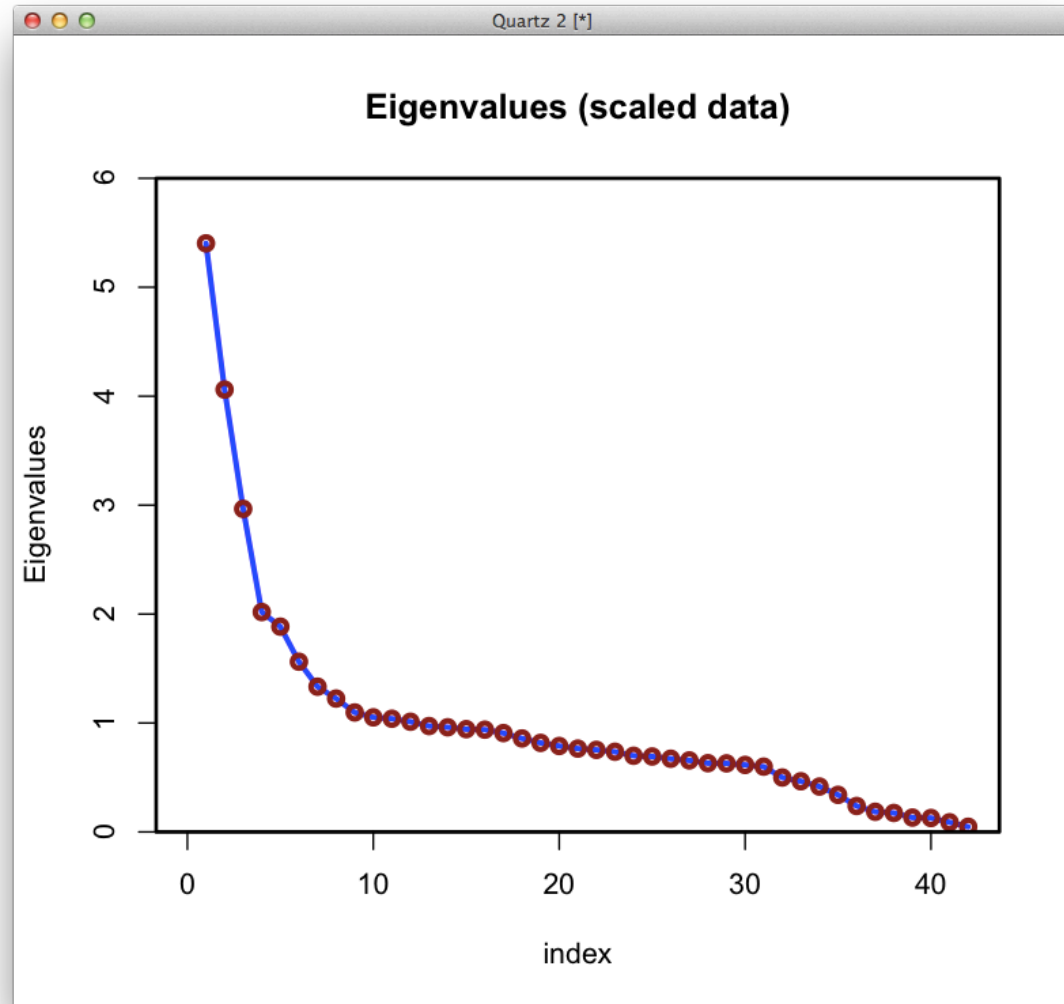


# Scaling the data or not in PCA

- ✱ 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

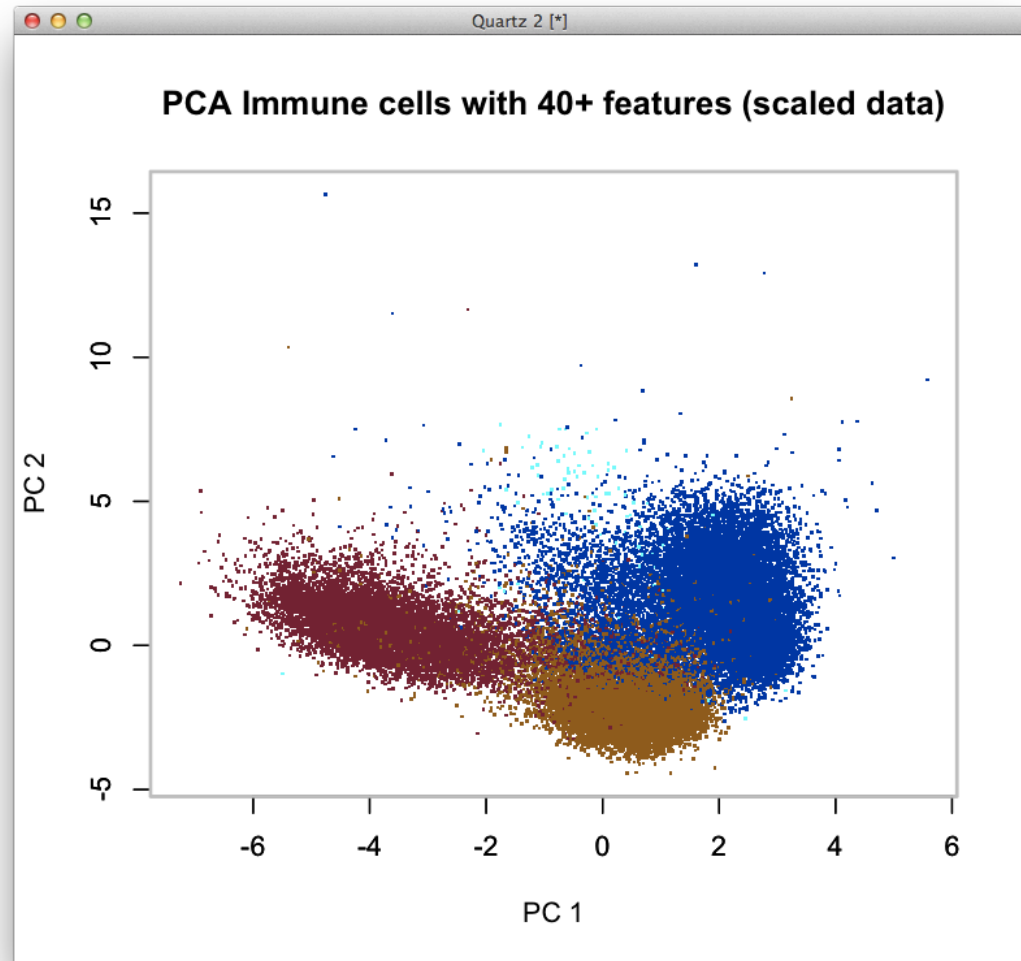
# Eigenvalues of the covariance matrix (scaled data)

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






# Q. Which of these are true?

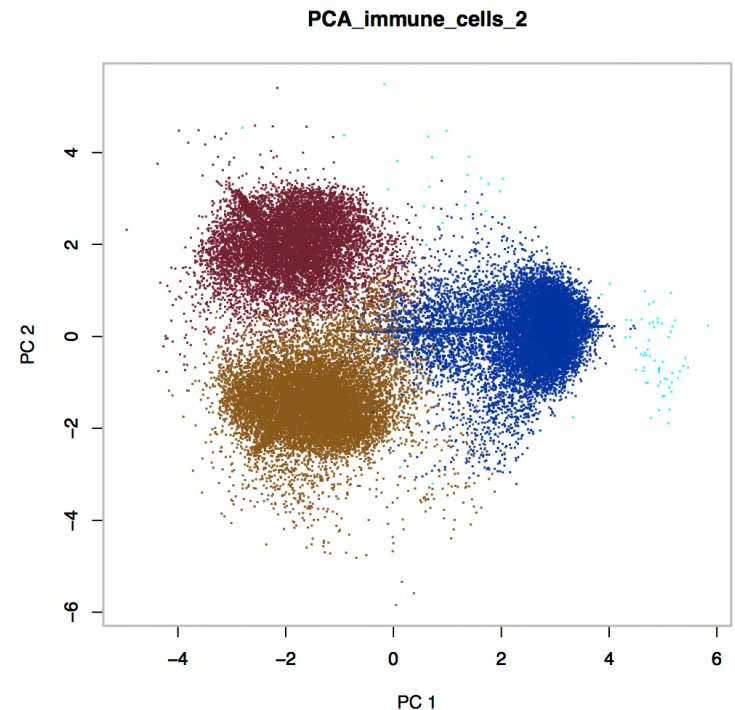
- A. Feature selection should be conducted with domain knowledge
- B. Important feature may not show big variance
- C. Scaling doesn't change eigenvalues of covariance matrix
- D. A & B

# 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.



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



# 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
  - ✱ ...

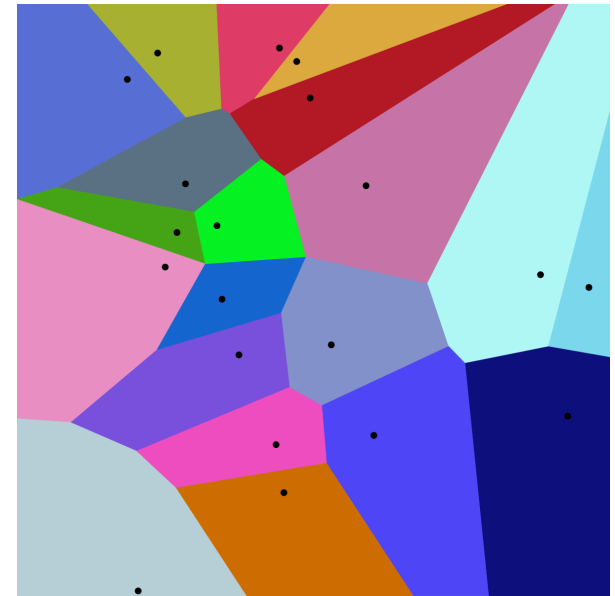
Given our knowledge of probability and statistics, can you think of any classifiers?

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  $\mathbf{x}$
  - \* Find the closest labeled  $\mathbf{x}_i$
  - \* Assign the same label to  $\mathbf{x}$
- \* Practical issues
  - \* We need a distance metric
  - \* We should first standardize the data
  - \* Classification may be less effective for very high dimensions



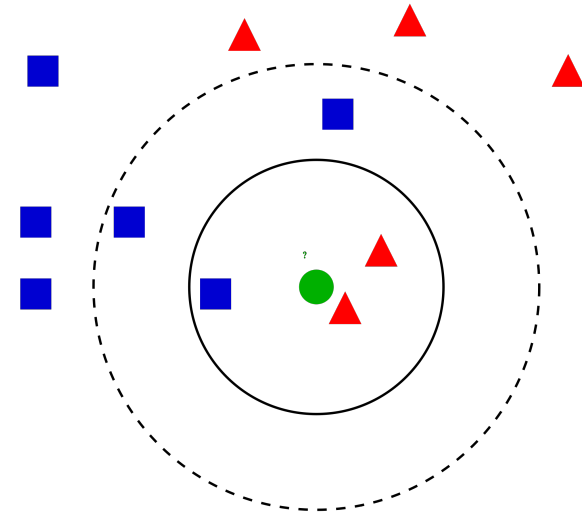
Source: wikipedia

# 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  $(k, \ell)$ -nearest neighbors, the classifier:

- ✱ Looks at the k nearest labeled feature vectors

- ✱ Assigns a label to  $\mathbf{x}$  if at least  $\ell$  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)
- ✱ Sometimes one type of error is more costly
  - ✱ Drug effect test
  - ✱ Crime detection
- ✱ We can tabulate the performance in a class confusion matrix

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

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

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

# Performance of a multiclass classifier

✱ Assuming there are  $c$  classes:

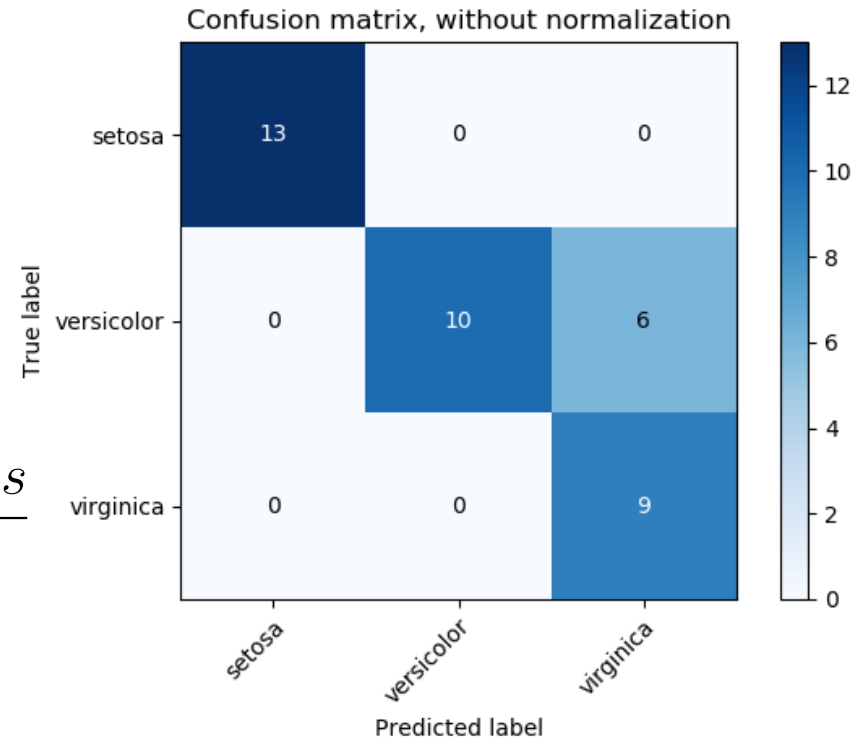
✱ The class confusion matrix is  $c \times c$

✱ Under the 0-1 loss function

$$\text{accuracy} = \frac{\text{sum of diagonal terms}}{\text{sum of all terms}}$$

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

✱ The baseline accuracy is  $1/c$ .



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.
- ✱ Split the labeled data into training and validation sets in multiple ways
- ✱ For each split (called a **fold**)
  - ✱ Train a classifier on the training set
  - ✱ Evaluate its accuracy on the validation set
- ✱ 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?

A. 50

B. 49

C.  $50 \times 49$

# 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?

**\*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.**



# Assignments

- ✱ Read Chapter 11 of the textbook
- ✱ Next time: Decision tree, Random forest classifier
- ✱ Prepare for midterm2 exam (4/15)
  - ✱ Lec 11-Lec 17, 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!*

