# Probability and Statistics for Computer Science



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

Credit: wikipedia

Hongye Liu, Teaching Assistant Prof, CS361, UIUC, 10.29.2021

#### Last time

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

## Objectives

#### # Principal Component Analysis (II)

#### **\*** Introduction to classification

## Step 2: Rotate so that the new data has diagonalized covariance matrix



## Step 3: Drop component(s)



## PCA an example

Step 1.

⋇

畿

$$\frac{1}{N-1}\sum_{i}\|r_{i}-p_{i}\|^{2} = \frac{1}{N-1}\sum_{i}\sum_{j=s+1}^{d}(r_{i}^{(j)})^{2}$$

$$\frac{1}{N-1}\sum_{i}\|r_{i}-p_{i}\|^{2} = \frac{1}{N-1}\sum_{i}\sum_{j=s+1}^{d}(r_{i}^{(j)})^{2} = \sum_{j=s+1}^{d}\sum_{i}\frac{1}{N-1}(r_{i}^{(j)})^{2}$$

$$\frac{1}{N-1} \sum_{i} ||r_{i} - p_{i}||^{2} = \frac{1}{N-1} \sum_{i} \sum_{j=s+1}^{d} (r_{i}^{(j)})^{2} = \sum_{j=s+1}^{d} \sum_{i} \frac{1}{N-1} (r_{i}^{(j)})^{2}$$
$$= \sum_{j=s+1}^{d} var(r_{i}^{(j)})$$

$$\frac{1}{N-1} \sum_{i} ||r_{i} - p_{i}||^{2} = \frac{1}{N-1} \sum_{i} \sum_{j=s+1}^{d} (r_{i}^{(j)})^{2} = \sum_{j=s+1}^{d} \sum_{i} \frac{1}{N-1} (r_{i}^{(j)})^{2}$$
$$= \sum_{j=s+1}^{d} var(r_{i}^{(j)})$$
$$= \sum_{j=s+1}^{d} \lambda_{j}$$

## Examples: 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

N

- There are 38816 white blood immune cells from a mouse sample
- Each immune cell has
   40+
   features/components
- Four features are used for the 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

\$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





#### New coordinates in PCA

> head(new\_coord\_t)

PC1 PC2 PC3 PC4

1 3.6739228 0.1127233 -1.32744266 0.61005994

2 -0.9255199 -2.1016573 -0.80762548 -0.29104900

3 3.1150230 0.3526459 -0.83994064 0.46074556

4 3.1801414 0.5679807 -0.07097689 0.01539266

5 2.7972723 -0.1073053 -0.39168826 -0.03981390

6 3.3012610 0.1979659 0.17965423 0.45373049





## What is the percentage of variance that PC1 covers?

54%

16%

C. 25%

Given the eigenvalues: 4.7642829 2.1486896 1.3730662 0.4968255, what is the percentage that PC1 covers?

$$\sum Var(f_i) = \sum \lambda_i$$

## PCA an example

Step 1.

⊯

⊯

$$D = \begin{bmatrix} 3 & -4 & 7 & 1 & -4 & -3 \\ 7 & -6 & 8 & -1 & -1 & -7 \end{bmatrix} \Rightarrow mean(D) = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

$$m = \begin{bmatrix} 3 & -4 & 7 & 1 & -4 & -3 \\ 7 & -6 & 8 & -1 & -1 & -7 \end{bmatrix}$$

$$Av = \lambda v$$

$$A - \lambda 1 \begin{bmatrix} z \\ -\lambda 2 \end{bmatrix}$$
Step 2.
$$Covmat(m) = \begin{bmatrix} 20 & 25 \\ 25 & 40 \end{bmatrix} \Rightarrow \lambda_1 \simeq 57; \quad \lambda_2 \simeq 3$$

$$\Rightarrow U = \begin{bmatrix} 0.5606288 & -0.8280672 \\ 0.8280672 & 0.5606288 \end{bmatrix} \quad U^T = \begin{bmatrix} 0.5606288 & 0.8280672 \\ -0.8280672 & 0.5606288 \end{bmatrix}$$

$$\Rightarrow r = U^T m = \begin{bmatrix} 7.478 & -7.211 & 10.549 & -0.267 & -3.071 & -7.478 \\ 1.440 & -0.052 & -1.311 & -1.389 & 2.752 & -1.440 \end{bmatrix}$$
Step 3. 
$$\Rightarrow p = \begin{bmatrix} 7.478 & -7.211 & 10.549 & -0.267 & -3.071 & -7.478 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

## More features used

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

42 × 38816



## 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 X A & B

### Reconstructing the data

 ${}^{*}$  Given the projected data  ${m p}_{d imes n}$  and mean({x}), we can approximately reconstruct the original data \_ – – –

 $m = D - mean({x})$ 

$$\widehat{D} = Up + mean(\{x\})$$

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

#### End-to-end mean square error

- \* Each  $oldsymbol{x}_i$  becomes  $oldsymbol{r}_i$  by translation and rotation
- \* Each  $oldsymbol{p}_i$  becomes  $\widehat{oldsymbol{x}}_i$  by the opposite rotation and translation
- \*\* Therefore the end to end mean square error is:  $\frac{1}{N-1}\sum_{i}\|\widehat{x}_{i}-x_{i}\|^{2} = \frac{1}{N-1}\sum_{i}\|r_{i}-p_{i}\|^{2} = \sum_{j=s+1}^{d}\lambda_{j}$ \*\*  $\lambda_{s+1}, ..., \lambda_{d}$  are the smallest d-s eigenvalues of the Covmat({x})

### PCA: Human face data

ing axi

- The dataset consists of 213 images of faces
- \* Each image is grayscale and has 64 by 64 resolution
- \* We can treat each image as a vector with dimension  $d = 4096 = 64 \times 64$ 
  - n = 2/3



#### 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



 $1^{st}$  row show the reconstructions using some number of principal components  $2^{nd}$  row show the corresponding errors



## 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?

## |u=)

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

### Learning to classify

Given a set of feature vectors x<sub>i</sub>, where each has a class label y<sub>i</sub>, we want to train a classifier that maps unlabeled data with the same features to its label.



## **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 x
  - ✤ Find the closest labeled x<sub>i</sub>
  - Assign the same label to x
- # Practical issues
  - We need a distance metric



\* We should first standardize the data

Source: wikipedia

Classification may be less effective for very high dimensions

#### Variants of nearest neighbors classifier

- In k-nearest neighbors, the classifier:
  - Looks at the k nearest labeled feature vectors x<sub>i</sub>
  - \* Assigns a label to 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 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)
- Sometimes one type of error is more costly
  - # Drug effect test
  - # Crime detection
- We can tabulate the performance in a class confusion matrix





## 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



- \* Under the 0-1 loss function \* 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 × c

\* Under the 0-1 loss function \* accuracy=  $\frac{sum \ of \ diagonal \ terms}{sum \ of \ all \ terms}$ 

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 calit (called a fold)
- 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\*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, 5). 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.

 $\binom{\diamond}{17}$ 

### 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

Ð



#### Training a decision tree: example

#### \* The "Iris" data set

Iris



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



A. 6/150B. 144/150C. 145/150

## Assignments

#### Read Chapter 11 of the textbook

- \*\* Next time: Decision tree, Random forest classifier
- # Prepare for midterm2 exam (11/12)

\* 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!

