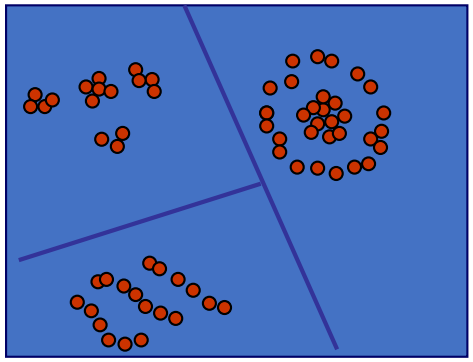
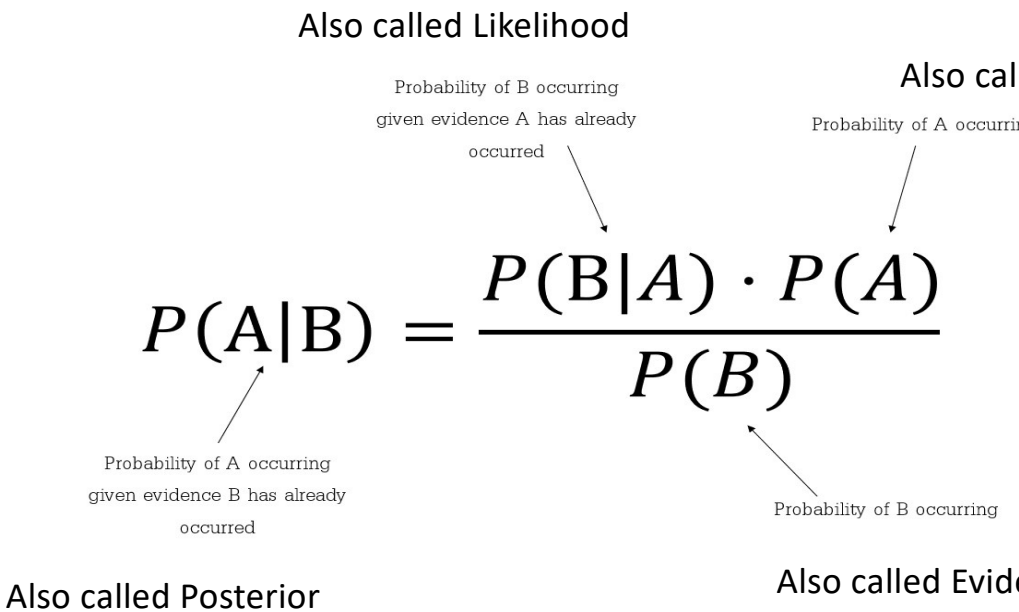
Naïve Bayes Classifier
and
Fisher's Linear Discriminant Function



These slides describe two classifiers: The naïve Bayes (NB) classifier and the Fisher's Linear Discriminant Function. The NB is a simple classifier that is fast and easy to train. It is highly popular for text classification tasks. The FLD is also a linear classifier. It is also used as a dimensionality reduction method in many situations.

Bayes Classifier

- A probabilistic framework for solving classification problems. Bayes classifier represents an example of generative approach. As the name implies, it is based on Bayes Theorem.



Bayes Theorem relates conditional probabilities with evidence and priors

Bayesian Classifier Scenario

- We assume there are *d* features: $x_1, \cdots, x_d$
- These features are measured on objects from two classes: $c_1 \text{ and } c_2$
- According to Bayes' theorem, the probability that the observation vector $\mathbf{x} = [x_1, \cdots, x_d]^T$ belongs to class c_j is given by the following relationship:

$$P(c_j|\mathbf{x}) = \frac{P(x_1,\cdots,x_d|c_j)P(c_j)}{P(x_1,\cdots,x_d)}, j = 1, 2$$

We classify vector **X** as belonging to class C_1 if the posteriori probability (the result of above expression with j=1 is better than the result with j=2. In case of a tie, we randomly pick one of the two options.

Naïve Bayes (NB) Classifier

- A Bayes classifier is termed as the NB classifier when we assume that features are independent of each other. Such an assumption rarely holds true, but the assumption helps simplify math and the classifier design. The NB classifier is particularly popular for text classification.
- With independent features, the expression for posteriori computation from the previous slide can be written as follows:

$$P(c_j|\mathbf{x}) = \frac{P(c_j)\prod_{i=1}^{d} P(x_i|c_j)}{P(x_1,\cdots,x_d)}, j = 1, 2$$

Independence assumption makes it easy to compute conditional probabilities. Say we have 10 discrete attributes. Without independence assumption, we need to calculate 1024 conditional probabilities per class. With independence assumption, we just need 10 conditional probabilities per class.

NB Classifier

- If you look at the denominator in the posteriori computation expression, you will find it is constant for a given input. Thus, we can ignore it and the classification rule for a given observation vector can be expressed as:
- Assign $\mathbf{x} \rightarrow c_1$ if $P(c_1) \prod_{i=1}^{d} P(x_i|c_1) \geq P(c_2) \prod_{i=1}^{d} P(x_i|c_2)$
- Otherwise assign $\mathbf{x} \rightarrow c_2$

NB Classifier for C Classes

- Many times, the number of classes is more than two. An example would be digit classification.
- Assuming there are C classes, the NB classification rule becomes

$$\mathbf{x} \rightarrow c_j \text{ where } P(c_j)\prod_{i=1}^{d} P(x_i|c_j) > P(c_k)\prod_{i=1}^{d} P(x_i|c_k), k = 1,...,C \text{ and } k \neq j$$

Again, a tie is resolved randomly.

An Example of NB Classification

- A company like Netflix divides its customers into two categories: Conservatives (C) and Liberals (L). 90% of its customers are conservatives. The company uses this classification of customers to serve the appropriate ads to maximize its revenue.
- The company's database shows the following preferences for four different show categories by its two types of customers

	Show Category S1	**Show Category S2**	**Show Category S3**	**Show Category S4**
Liberals	95% like this category	5% like this category	2% like this category	20% like this category
Conservatives	3% like this	82% like this	34% like this	92% like this

Example Contd.

- A new customer just signed by the company likes S1 and S3 categories of shows but dislikes S2 and S4 show categories. Assuming that likes/dislikes of different show categories are independent of each other, how should the company classify this new customer?
- This type of classification problems are suitable for an NB classifier.
- Let's see what we have here. We know the following priors (a-priori probabilities)
- Prob.(C) = 0.90 and Prob.(L) = 0.10

If we were forced to classify the new customer without knowing his/her likings, we would classify the customer as Conservative because Prob.(C) > Prob.(L). By finding out the customer's likings, Bayes rule tells us how to convert these priors into posterior probabilities.

Example Contd.

- From the company's database, we know:
- Prob.(S1=1/C) = 0.03; Prob.(S2=1/C) = 0.82; Prob.(S3=1/C) = 0.34; Prob.(S4=1/C) = 0.92
- Prob.(S1=1/L) = 0.95; Prob.(S2=1/L) = 0.05; Prob.(S3=1/L) = 0.02; Prob.(S4=1/L) = 0.20

[S1=1 implies liking S1, S1=0 means not liking S1; same for other shows]

Example Contd.

- The new customer likes S1 and S3 but doesn't like S2 and S4. Using this information, let us calculate
- Prob.(C)*Prob.(S1=1/C)*Prob.(S2=0/C)*Prob.(S3=1/C)*Prob.(S4=0/C)

= 0.9 * 0.03 * 0.18 * 0.34 * 0.08 => 0.00013219

- Let's do the similar calculation for class Liberal
- Prob.(L)*Prob.(S1=1/L)*Prob.(S2=0/L)*Prob.(S3=1/L)*Prob.(S4=0/L)

= 0.1 * 0.95 * 0.95 * 0.02 * 0.8 => 0.0014

- The above numbers suggest the new customer is "Liberal"
- If we were asked to also estimate the probability of the new customer being liberal, it will be computed as 0.0014/(0.0014 + 0.00013219) => 0.9161

How to Estimate Probabilities from Data?

- In the simple example given earlier, the conditional probabilities were known. In general, these must be estimated from the training data.
- For discrete attributes, we simply count how many times an attribute value occurs for a class in the training data
- For numerical attributes, we have two options
  - Discretize attributes, or
  - Assume an underlying distribution and estimate the distribution parameters

Example

F1	F2	F3	F4	F5
0	0	1	1	1
1	0	1	1	0
1	1	0	0	1
1	1	0	0	0
0	1	0	0	1
0	0	0	1	0

Introvert

F1	F2	F3	F4	F5
1	0	0	1	1
1	1	0	0	1
1	1	1	1	0
1	1	0	1	0
1	1	0	1	1
1	0	1	1	0
1	0	1	0	0

Outgoing

We have data from 6 persons, labelled as introverts, and from 7 persons labelled as outgoing. The data is based on five traits that are present (1) or absent (0). This data is our training set. Now, we receive a data vector [1 0 1 1 0] that represents the traits of a new person. Our goal is to classify this new person as introvert or outgoing.

To estimate conditional probabilities, we simply count and obtain the following estimates. This type of estimation is known as *maximum likelihood estimation*.

p(F1=1/Int) = 1/2
p(F2=1/Int) = 1/2
p(F3=1/Int) = 1/3
p(F4=1/Int) = 1/2
p(F5=1/Int) = 1/2

p(F1=1/Out) = 1
p(F2=1/Out) = 4/7
p(F3=1/Out) = 3/7
p(F4=1/Out) = 5/7
p(F5=1/Out) = 3/7

p(Int)=6/13
p(Out)=7/13

Using these conditional probabilities, you can now apply NB classifier. The result in this case will classify the new person as Outgoing

Another Example

Outlook			Temperature			Humidity			Windy			Play	
	Yes	*No*		*Yes*	*No*		*Yes*	*No*		*Yes*	*No*	*Yes*	*No*
Sunny	2	3	Hot	2	2	High	3	4	False	6	2	9	5
Overcast	4	0	Mild	4	2	Normal	6	1	True	3	3		
Rainy	3	2	Cool	3	1								
Sunny	2/9	3/5	Hot	2/9	2/5	High	3/9	4/5	False	6/9	2/5	9/	5/
Overcast	4/9	0/5	Mild	4/9	2/5	Normal	6/9	1/5	True	3/9	3/5	14	14
Rainy	3/9	2/5	Cool	3/9	1/5								

Outlook	Temp.	Humidity	Windy	Play
Sunny	Cool	High	True	?

Likelihood of the two classes

For "yes" = $2/9 \times 3/9 \times 3/9 \times 3/9 \times 9/14 = 0.0053$

For "no" = $3/5 \times 1/5 \times 4/5 \times 3/5 \times 5/14 = 0.0206$

Conversion into a probability by normalization:

P("yes") = 0.0053 / (0.0053 + 0.0206) = 0.205

P("no") = 0.0206 / (0.0053 + 0.0206) = 0.795

In this example, the table on left summarizes different weather conditions under which a game was either played or not played.

The top of the second table lists a particular combination of weather conditions. The goal is to apply NB classifier to figure out play/no-play.

Laplace Smoothing

- Specially, important in text mining where new words may appear in test documents
- The idea of Laplace smoothing is to modify the formula for calculating probabilities to avoid zero probabilities and division by zero
- Conditional Probability without smoothing

$$P_r(word_i \mid class_j) = \frac{count\_of\_docs(word_i, class_j)}{count\_of\_docs(class_j)}$$

- Probability with smoothing

$$P_r(word_i \mid class_j) = \frac{count\_of\_docs(word_i, class_j) + 1}{count\_of\_docs(class_j) + 2}$$

Example

- There are four documents in our training set. Each document is characterized by six words (features). There are two classes: Topic is China or not.

docID	Chinese	Beijing	Shanghai	Macao	Tokyo	Japan	Topic = China?
1	1	1	0	0	0	0	yes
2	1	0	1	0	0	0	yes
3	1	0	0	1	0	0	yes
4	1	0	0	0	1	1	no
5	1	0	0	0	1	1	?

- The document # 5 needs to be classified.

Estimate Probabilities

c = about China; $\bar{c}$ = not about China.

Parameter estimates:

$$\begin{aligned}
\hat{P}(Chinese|c) &= (3+1)/(3+2) = 4/5 \\
\hat{P}(Japan|c) = \hat{P}(Tokyo|c) &= (0+1)/(3+2) = 1/5 \\
\hat{P}(Beijing|c) = \hat{P}(Macao|c) = \hat{P}(Shanghai|c) &= (1+1)/(3+2) = 2/5
\end{aligned}$$

$$\begin{aligned}
\hat{P}(Chinese|\bar{c}) &= (1+1)/(1+2) = 2/3 \\
\hat{P}(Japan|\bar{c}) = \hat{P}(Tokyo|\bar{c}) &= (1+1)/(1+2) = 2/3 \\
\hat{P}(Beijing|\bar{c}) = \hat{P}(Macao|\bar{c}) = \hat{P}(Shanghai|\bar{c}) &= (0+1)/(1+2) = 1/3
\end{aligned}$$

Furthermore, $\hat{P}(c) = \frac{3}{4}$ and $\hat{P}(\bar{c}) = \frac{1}{4}$.

Classifying Doc# 5

Naive Bayes prediction for document 5:

$$\begin{aligned}\hat{P}(c|(1,0,0,0,1,1)) \propto \hat{P}(c) \cdot \hat{P}(\mathit{Chinese}|c) \cdot (1 - \hat{P}(\mathit{Beijing}|c)) \\ \cdot (1 - \hat{P}(\mathit{Shanghai}|c) \cdot (1 - \hat{P}(\mathit{Macao}|c)) \\ \cdot \hat{P}(\mathit{Tokyo}|c) \cdot \hat{P}(\mathit{Japan}|c) \\ = 3/4 \cdot 4/5 \cdot 3/5 \cdot 3/5 \cdot 3/5 \cdot 1/5 \cdot 1/5 \approx 0.005\end{aligned}$$

Analogously

$$\hat{P}(\bar{c}|(1,0,0,0,1,1)) \propto 1/4 \cdot (2/3)^6 \approx 0.022$$

Hence document 5 is classified as not being about China.

Taking Word Frequencies into Consideration

- The previous model for calculations ignores how many times a word occurs in a document. This works well for short documents, e.g. tweets
- A better approach for text classification may be to make use of multinomial Naïve Bayes model considering word or term frequencies
- This is done by using the following formula for probability estimation (V is the size of the vocabulary)

$$P_r(word_i \mid class_j) = \frac{count(word_i, class_j) + 1}{\sum_{k=1}^{V} count(word_k, class_j) + V}$$

Multinomial Example

This was considered earlier using the binary model, known as the Bernoulli model. We now use the multinomial model.

Representation in the Multinomial model:

docID	Chinese	Beijing	Shanghai	Macao	Tokyo	Japan	Topic = China?
1	2	1	0	0	0	0	yes
2	2	0	1	0	0	0	yes
3	1	0	0	1	0	0	yes
4	1	0	0	0	1	1	no
5	3	0	0	0	1	1	?

Multinomial Example: Probability Estimation

$$\begin{aligned}
\hat{P}(\textit{Chinese}|c) &= (5+1)/(8+6) = 3/7 \\
\hat{P}(\textit{Japan}|c) = \hat{P}(\textit{Tokyo}|c) &= (0+1)/(8+6) = 1/14 \\
\hat{P}(\textit{Beijing}|c) = \hat{P}(\textit{Macao}|c) = \hat{P}(\textit{Shanghai}|c) &= (1+1)/(8+6) = 1/7 \\
\\
\hat{P}(\textit{Chinese}|\bar{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textit{Japan}|\bar{c}) = \hat{P}(\textit{Tokyo}|\bar{c}) &= (1+1)/(3+6) = 2/9 \\
\hat{P}(\textit{Beijing}|\bar{c}) = \hat{P}(\textit{Macao}|\bar{c}) = \hat{P}(\textit{Shanghai}|\bar{c}) &= (0+1)/(3+6) = 1/9
\end{aligned}$$

Furthermore, $\hat{P}(c) = \frac{3}{4}$ and $\hat{P}(\bar{c}) = \frac{1}{4}$.

Plugging in values, the probability that doc5 comes from “China” class is about 0.0003 and “Not China” class is 0.0001. This is different from earlier answer because it takes word frequencies into account too.

What Happens with Continuous Attributes?

- One easy solution for estimating conditional densities is to transform the continuous attribute to a discrete attribute. For example, attribute *age* could be binned into four or more bins
- Another solution is to model the attribute variation through some probability distribution. The most common choice for this is the Gaussian or Normal distribution.

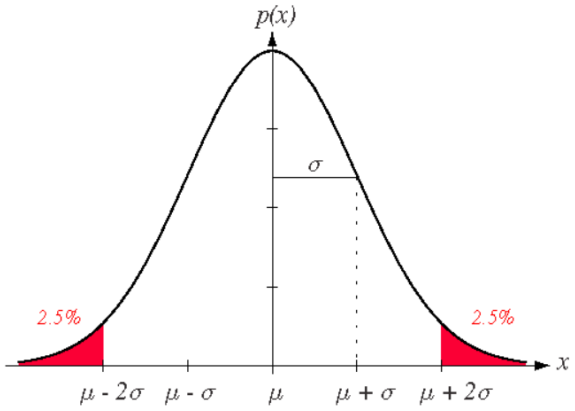
Normal Distribution

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{-\frac{(A_i - \mu_{ij})^2}{2\sigma_{ij}^2}}$$

We use this for continuous attributes.

  - One for each attribute-class pair
  - We estimate mean and variance for each pair using the training data.



Example

Temperature	Humidity	Play
85	85	No
80	90	No
65	70	No
72	95	No
71	80	No
83	78	Yes
70	96	Yes
68	80	Yes
64	65	Yes
69	70	Yes
75	80	Yes
75	70	Yes
72	90	Yes
81	75	Yes

The data on the left has two continuous attributes and two classes. Using this data we want to build a NB classifier and answer the question whether play or not when temperature is 78 and humidity 82?

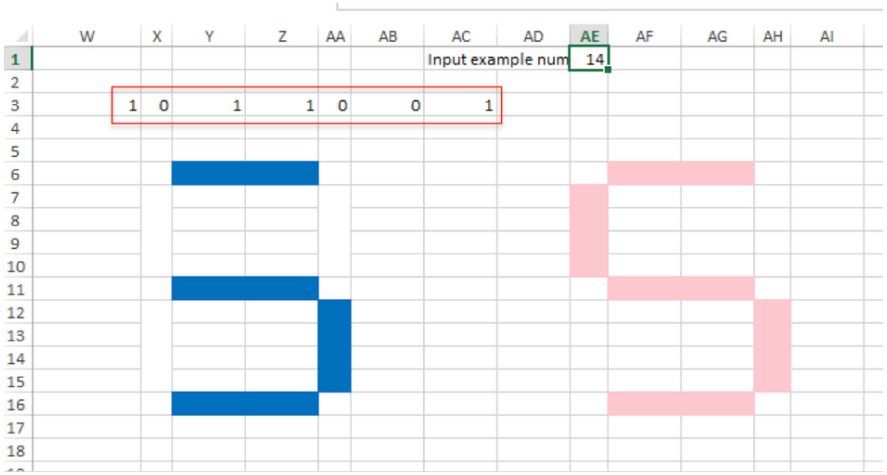
First, we fit normal distribution to the attributes.

Pr(No)	0.36		
Pr(Yes)	0.64		
Mean(Temp/No)	74.6	STD(Temp/No)	8.812869378
Mean(Hum/No)	84	STD(Hum/No)	9.617692031
Mean(Temp/Yes)	73	STD(Temp/Yes)	6.164414003
Mean(Hum/Yes)	78.22	STD(Hum/Yes)	9.884050002

See the Excel worksheet posted on Moodle for Excel formulas used here

Should we play with temperature 78 and humidity 82?				
Prob(No/Temp=78)	0.04202157	Prob(No)	0.060920477	
Prob(No/Hum=82)	0.040592804			
Prob(Yes/Temp=78)	0.046575577	Prob(Yes)	0.112337845	Play
Prob(Yes/Hum=82)	0.037519183			

https://iksinc.online/2017/07/17/faulty-led-display-digit-recognition-illustration-of-naive-bayes-classifier-using-excel/



Displayed faulty input

Predicted digit display

This link gives a complete example of NB classifier from data generation to model building and classification.

Naïve Bayes (Summary)

- Simple, easy to understand, build, and implement
- Fast training and classification
- Linear classifier, widely used in text mining and other applications
- Robust to isolated noise points
- Handles missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes. In that case, the result is a Gaussian classifier which may or may not be linear depending upon the covariance matrices.

Linear Discriminant Function (LDF)/LDA

- Consider a linear function, *g(x).* Such a function can be used as a binary (two-category) classifier with the following rule:

  Decide c_1 *if* $g(x) > 0$ *and* c_2 *if* $g(x) < 0$

  If $g(x) = 0 \Rightarrow$ *x can be assigned to either class*

- One way to express a linear function is as W^tX where W is called the *weight vector*
- How do we find such a function or the weight vector using training examples?
- One answer -> Use Fisher's criterion

Linear Discriminant Functions

- Consider two points, X_1 and X_2, that lie on the hyperplane specified by *g(X).* Then

  $$W^t X_1 + w_0 = W^t X_2 + w_0$$

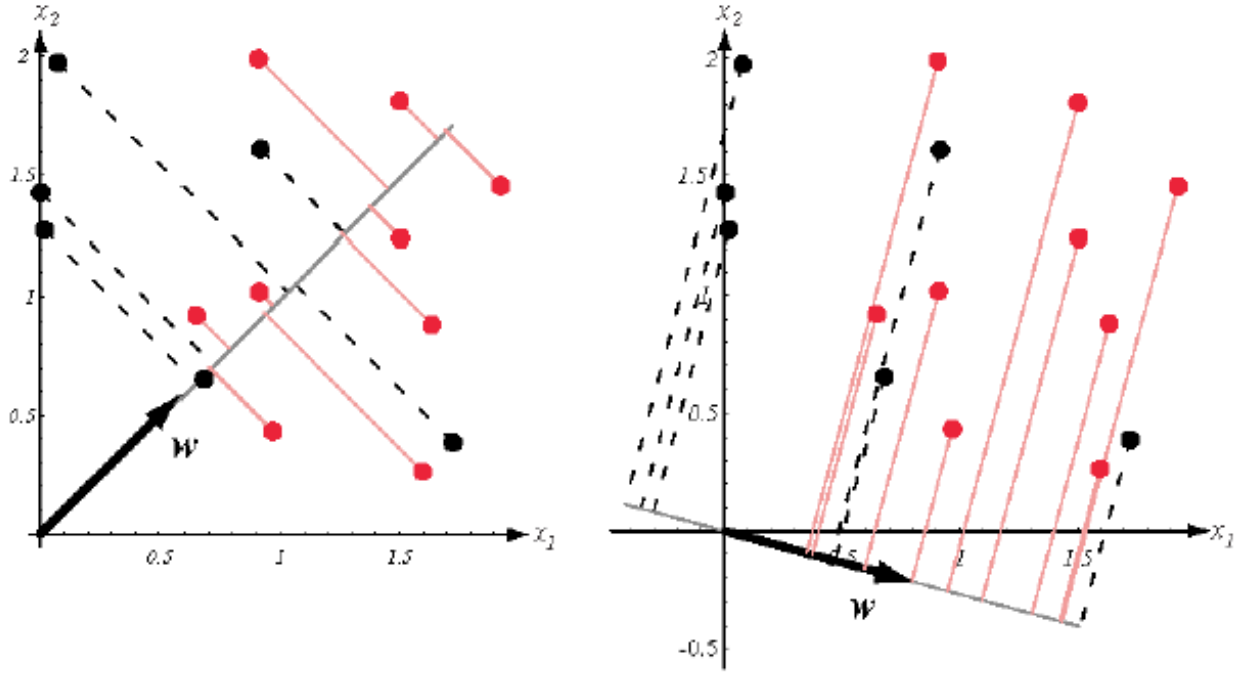
  and

  $$W^t(X_1 - X_2) = 0$$

- The highlighted equation shows that the weight vector is normal to any vector lying in the the hyperplane

Instead of thinking about the hyperplane, consider two points lying on a line. You will get a similar equation for the perpendicular bisector of the line.

Fisher's Linear Discriminant Function (FLDF)



Black and red circles represent training data from two classes.

Rotate the thick black line around origin to find the best projection where red and black dots are best separated. How do we do it mathematically?

Fisher's Criterion Function

$y_i = \mathbf{w}^t \mathbf{x}_i$ Defines the projection using W

$$J(\mathbf{w}) = \frac{|\tilde{m}_1 - \tilde{m}_2|^2}{(\tilde{s}_1^2 + \tilde{s}_2^2)}$$

This tells us how to find the best projection.

$\tilde{m}_1, \tilde{m}_2$ Projected means of training data points from two classes

$\tilde{s}_1^2, \tilde{s}_2^2$ Projected variances of training data points from two classes

We want to maximize J(W). Why?

Fisher's LDF

- To compute the optimal $\mathbf{w}$, we define the *scatter matrices* $\mathbf{S}_i$

$$\mathbf{S}_i = \sum_{\mathbf{x} \in \mathcal{D}_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T \text{ where } \mathbf{m}_i = \frac{1}{\#\mathcal{D}_i} \sum_{\mathbf{x} \in \mathcal{D}_i} \mathbf{x},$$

the *within-class scatter matrix* $\mathbf{S}_\mathbf{W}$

$$\mathbf{S}_\mathbf{W} = \mathbf{S}_1 + \mathbf{S}_2,$$

and the *between-class scatter matrix* $\mathbf{S}_\mathbf{B}$

$$\mathbf{S}_\mathbf{B} = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T.$$

- Then, the criterion function becomes

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S}_\mathbf{B} \mathbf{w}}{\mathbf{w}^T \mathbf{S}_\mathbf{W} \mathbf{w}}$$

and the optimal $\mathbf{w}$ can be computed as

$$\mathbf{w} = \mathbf{S}_\mathbf{W}^{-1}(\mathbf{m}_1 - \mathbf{m}_2).$$

FLD Example

Example:

2. A company hires its customer service representatives using three skill tests. Each skill test score lies in the interval of 0-10. A record of past year's hires provides the following information:

Employee #	Skill 1 Score	Skill 2 Score	Skill 3 Score	Employee Performance
1	8	9	6	Good
2	6	7	5	Good
3	5	4	7	Bad
4	3	7	2	Bad
5	9	6	3	Good
6	4	5	5	Bad
7	2	6	4	Bad
8	4	3	4	Bad
9	7	8	2	Good
10	9	4	4	Good

A prospective hire has the following score:

Skill 1 = 5 Skill 2 = 5 Skill 3 = 6

It is decided that the prospective hire should not be offered an employment if the test scores predict a bad performance. Answer whether this person should be hired or not?

```
G = np.array([[8,9,6],[6,7,5],[9,6,3],[7,8,2],[9,4,4]])# Define the input vectors for "Good"
Gmean = np.mean(G,0)# compute mean vector
print(Gmean)
```

```
[7.8 6.8 4. ]
```

```
B = np.array([[5,4,7],[3,7,2],[4,5,5],[2,6,4],[4,3,4]])# "Bad" employees
Bmean = np.mean(B,0)
print(Bmean)
```

```
[3.6 5.  4.4]
```

```
S1 = 4*np.cov(G.T)# Calculate the scatter matrix
print(S1)
```

```
[[ 6.8 -5.2 -1. ]
 [-5.2 14.8  3. ]
 [-1.   3.  10. ]]
```

```
S2 = 4*np.cov(B.T)
print(S2)
```

```
[[ 5.2 -5.   5.8]
 [-5.  10.  -7. ]
 [ 5.8 -7.  13.2]]
```

```
S = S1+S2 ##Total scatter matrix
print(S)
```

```
S = S1+S2 ##Total scatter matrix
print(S)
```

```
[[ 12.  -10.2   4.8]
 [-10.2  24.8  -4. ]
 [  4.8  -4.   23.2]]
```

```
from numpy.linalg import inv
S_inv = inv(S)
print(S_inv)
```

```
[[ 0.13580391  0.05279105 -0.01899546]
 [ 0.05279105  0.06199744 -0.00023307]
 [-0.01899546 -0.00023307  0.04699336]]
```

```
W = np.matmul(S_inv,(Gmean-Bmean))## Weight vector for projection
print(W)
```

```
[ 0.67299849  0.33341102 -0.09899779]
```

```
Gmean_proj = np.matmul(W.T,Gmean)
Bmean_proj = np.matmul(W.T,Bmean)
zcut = 0.5*(Gmean_proj+Bmean_proj)
print(Gmean_proj,Bmean_proj,zcut)
```

```
7.1205920055937515 3.6542594103251362 5.387425707959444
```

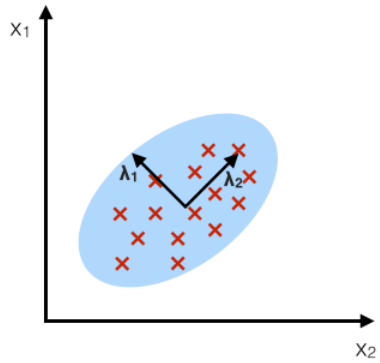
```
test_emp_proj = np.matmul(W.T,[5,5,6])
print(test_emp_proj)
```

```
4.438060832070853
```

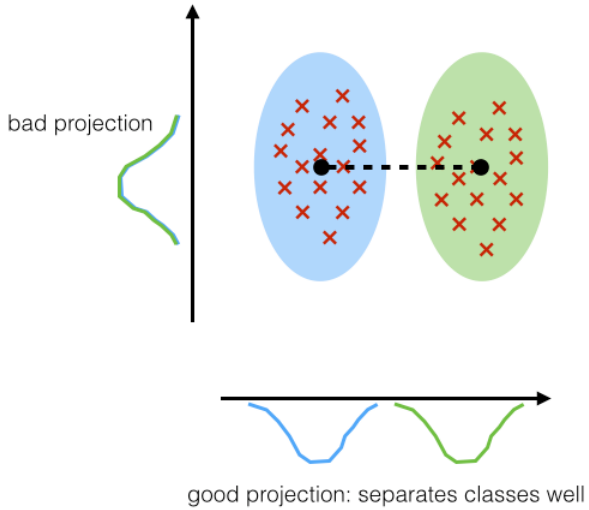
```
The candidate shouldn't be hired
```

Difference between PCA & FLDF

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



LDA also does dimensionality reduction but it uses class labels also.

Fisher's Discriminant for Multiple Classes

- For *k* classes, we can find *k-1* projections when dimensionality *d > k.* The projections are defined by the following equation:

$$\mathbf{y}_i = \mathbf{W}^t \mathbf{x}_i \text{ for } i = 1, \cdots, N$$

- The mapped vector $\mathbf{y}_i$ is of *k-1* dimensions. The matrix $\mathbf{W}^t$ is of size ?
- The weight vectors for projection are given by the eigenvectors of the following matrix

$$\mathbf{S}_w^{-1}\mathbf{S}_B$$

where

$$\mathbf{S}_W = \sum_{i=1}^{k} \mathbf{S}_k \text{ and}$$

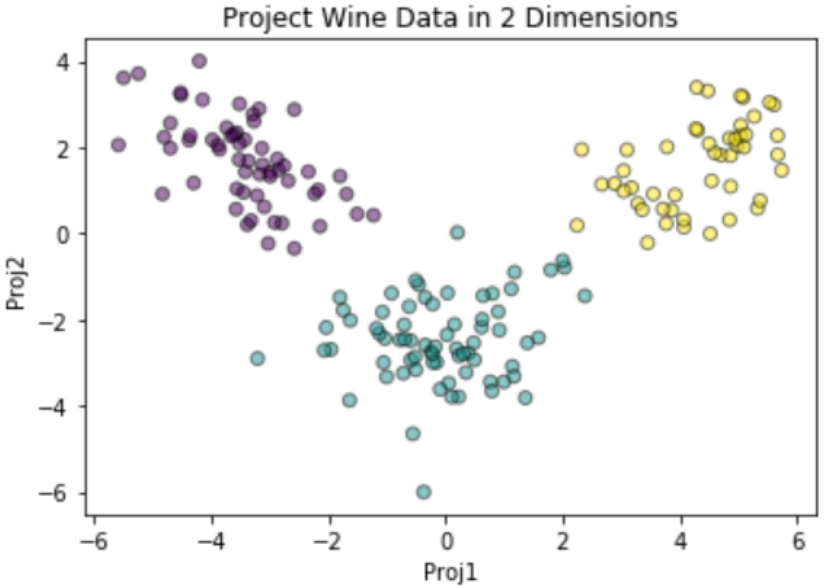
$$\mathbf{S}_k = \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^t \text{ and,}$$

$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^t$$

$\mathbf{m}_k$: mean vector of class *k* training examples

m : mean vector of all training examples

Example



Left panel : FLD

Right panel: PCA

https://iksinc.online/2018/11/12/dimensionality-reduction-via-linear-discriminant-analysis/

FLD Summary

- FLD is commonly referred as LDA.
- FLD can be used for M classes by defining M-1 projections
- Underlying assumption is that features are independent and are normally distributed. However, the method performs well even when these assumptions are not met
- PCA finds the axes with maximum variance for the whole data set whereas LDA tries to find the axes for best class separability. In practice, often a LDA is done followed by a PCA for dimensionality reduction.