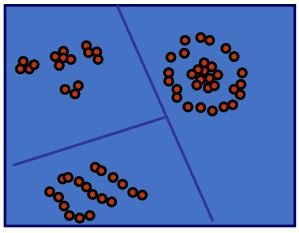
Building Classification Models : Part 2



Bayes Classifier

- A probabilistic framework for solving classification problems. An example of generative approach.
- Conditional Probability:

$$P(C \mid A) = \frac{P(A,C)}{P(A)}$$

$$P(A \mid C) = \frac{P(A,C)}{P(C)}$$

- Bayes theorem:

$$P(C \mid A) = \frac{P(A \mid C)P(C)}{P(A)}$$

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (x_1, x_2,…,x_n)
  - Goal is to predict class c
  - Specifically, we want to find the value of c that maximizes P(c| x_1, x_2,…,x_n)

Bayesian Classifiers

- Approach:
  - compute the posterior probability $P(c|x_1, x_2, \ldots, x_n)$ for all values of C using the Bayes theorem

$$P(c|x_1, x_2, \ldots, x_n) = \frac{P(x_1, x_2, \ldots, x_n|c)P(c)}{P(x_1, x_2, \ldots, x_n)}$$

  - Choose value of C that maximizes $P(c|x_1, x_2, \ldots, x_n)$
  - Equivalent to choosing value of C that maximizes $P(x_1, x_2, \ldots, x_n|c)P(c)$
  - How to estimate $P(x_1, x_2, \ldots, x_n|c)$?

Naïve Bayes Classifier

- Assume independence among attributes x_i when class is given:
  - $P(x_1, x_2, \ldots, x_n | c) = P(x_1|c_j)\, P(x_2|c_j) \ldots P(x_n|c_j)$
  - Can estimate $P(x_i|c_j)$ for all x_i and c_j.
  - New point is classified to c_j if $P(c_j) \prod P(x_i|c_j)$ is maximal.

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.

How to Estimate Probabilities from 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
  - Assume an underlying distribution and estimate the distribution parameters

Example

	Spam	Not Spam
Word1	80/100	20/50
Word2	70/100	10/50
Word3	40/100	30/50
Word4	60/100	40/50

- Test document(D) with Word1, Word2, and Word4
- Pr(Spam/D) = 0.67*0.80*0.70*0.60 = 0.225
- Pr(Not Spam/D) = 0.33*0.40*0.20*0.80 = 0.0211
- D is considered spam

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

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

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	?

Estimate Probabilities

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

Parameter estimates:

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

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

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

Classifying Doc# 5

Naive Bayes prediction for document 5:

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

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 taking into account 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

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

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

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.

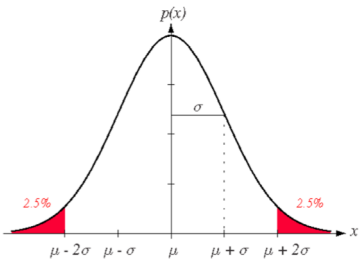
Normal Distribution

- Normal distribution:

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

We use this for continuous attributes.

- One for each (A_i,c_i) pair
- We will estimate mean and variance for each pair using the training data



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
- Handle missing values by ignoring the instance during probability estimate calculations
- Robust to irrelevant attributes
- Independence assumption may not hold for some attributes
  - Use other techniques such as Bayesian Belief Networks (BBN)

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 *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

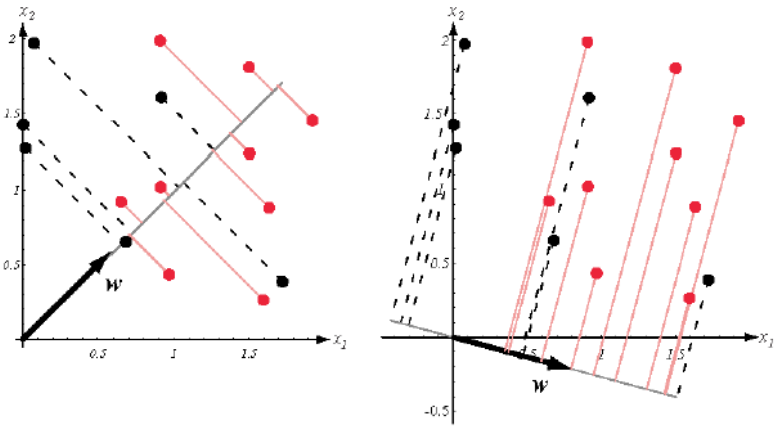
$$W^t X_1 + w_0 = W^t X_2 + w_0$$

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

Fisher's Linear Discriminant Function (FLDF)



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

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

$\tilde{m}_1, \tilde{m}_2$ 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 w, we define the *scatter matrices* S_i

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

the *within-class scatter matrix* $\mathrm{S_W}$

$$\mathrm{S_W} = \mathrm{S}_1 + \mathrm{S}_2,$$

and the *between-class scatter matrix* $\mathrm{S_B}$

$$\mathrm{S_B} = (\mathrm{m}_1 - \mathrm{m}_2)(\mathrm{m}_1 - \mathrm{m}_2)^T.$$

- Then, the criterion function becomes

$$J(\mathrm{w}) = \frac{\mathrm{w}^T \mathrm{S_B} \mathrm{w}}{\mathrm{w}^T \mathrm{S_W} \mathrm{w}}$$

and the optimal w can be computed as

$$\mathrm{w} = \mathrm{S_W^{-1}}(\mathrm{m}_1 - \mathrm{m}_2).$$

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
```

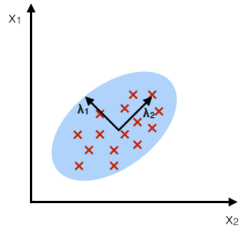
```
test_emp_proj = np.matmul(W.T,[5,5,6])
print(test_emp_proj)
```

```
4.438060832070853
```

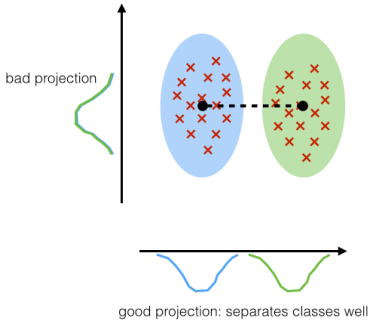
```
The candidate shouldn't be hired
```

Difference between PCA & FLDF

PCA:
component axes that maximize the variance



LDA:
maximizing the component axes for class-separation



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

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

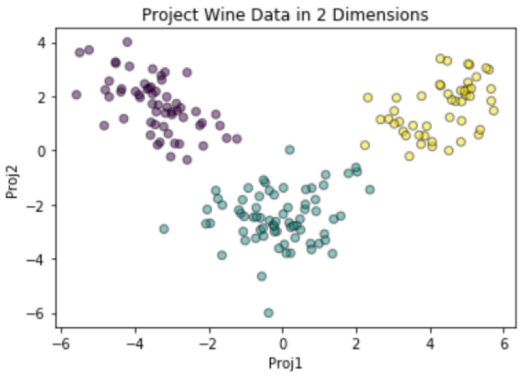
$$\mathbf{S}_k = \sum_{n \in C_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^t \text{ and,}$$

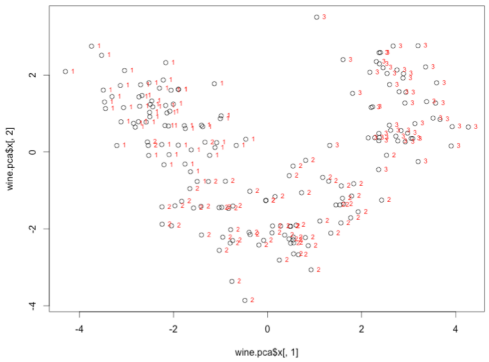
$$\mathbf{S}_B = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^t$$

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

m : mean vector of all training examples

Example





Left panel : FLD

Right panel: PCA

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.