Predictive Model: Generative Classifiers

Dr Liew How Hui

Jan 2022

Revision

Given Y = output $j \in \{1, ...K\}$, X = inputs x.

A **discriminative** Learning Model:

$$\mathbb{P}(Y = j | X = x) = \text{some estimate}$$

- kNN, wkNN: nonparametric
  $h_D(\mathrm{x}) = \mathrm{argmax}_{j \in \{1,...,K\}} \mathbb{P}(Y = j | X = \mathrm{X})$
- LR, multinomial LR, ANN: parametric
  $h_D(\mathrm{x}) = I(\mathbb{P}(Y = 1 | X = \mathrm{x}) > 0.5)$ (cut-off=0.5)
- Decision trees (to be introduced): nonparametric
  $h_D(\mathrm{x})$ =if-else statements.

Outline

1. Discriminative vs Generative Models

2. Naïve Bayes Classifiers

3. Bayesian Network

4. Discriminant Analysis Models

Discriminative & Generative Learning Models

Many supervised learning models can be viewed as approximations to the probability $\mathbb{P}(X, Y)$ over the product space of inputs and output.

- *Discriminative learning*: $\mathbb{P}(X, Y)$ is estimated from $\mathbb{P}(Y|X)$ directly (discriminative boundary of (X, Y)).
- *Generative learning*: $\mathbb{P}(X, Y)$ is estimated from $P(X|Y)P(Y)$. Examples:
  - Naive Bayes
  - Discriminative Analysis, e.g. LDA, QDA

Discriminative & Generative Models (cont)

The probabilistic framework that underlie the discriminative models is the **Maximum Likelihood Estimation (MLE)**

$$\begin{aligned}\hat{\theta} &= \text{argmax}_\theta\, P(D;\theta) \\ &= \underset{\theta}{\text{argmax}}\, P((\mathrm{x}_1, y_1), \cdots, (\mathrm{x}_n, y_n); \theta)\end{aligned}$$

The probabilistic framework that underlie the generative models is the **Maximum a Posteriori (MAP)**:

$$\begin{aligned}\hat{\theta} &= \text{argmax}_\theta\, P(\theta|D) \\ &= \underset{\theta}{\text{argmax}}\, P((\mathrm{x}_y, y_1), \cdots, (\mathrm{x}_n, y_n) \mid \theta) P(\theta).\end{aligned}$$

Generative Models

How to obtain a generative model?

Answer: Bayes Theorem $\mathbb{P}(B|A) = \frac{\mathbb{P}(A|B)\mathbb{P}(B)}{\mathbb{P}(A)}$.

Discriminative-Generative link:

$$\mathbb{P}(Y = y|\mathrm{X} = \mathrm{x}) = \frac{\mathbb{P}(\mathrm{X} = \mathrm{x}|Y = y)\mathbb{P}(Y = y)}{\mathbb{P}(\mathrm{X} = \mathrm{x})}. \tag{1}$$

Note that '$\mathbb{P}$' is regarded as "probability density" function f when x and y are continuous.

Generative Models (cont)

When the response variable Y is categorical and has K distinct values 1, ..., K, then (1) becomes

$$\mathbb{P}(Y=j|\mathrm{X}=\mathrm{x}) = \frac{\mathbb{P}(\mathrm{X}=\mathrm{x}|Y=j)\mathbb{P}(Y=j)}{\sum_{k=1}^{K}\mathbb{P}(\mathrm{X}=\mathrm{x}|Y=k)\mathbb{P}(Y=k)}, \tag{2}$$

where $j \in \{1, \cdots, K\}$;

- $\mathbb{P}(Y=j)$ is called the **prior probability**, i.e. the probability that a randomly chosen observation comes from the jth class of response variable Y;

Generative Models (cont)

- $\mathbb{P}(Y = k|\mathrm{X} = \mathrm{x})$, called the *posterior probability*, is the probability that the new observation belongs to the kth class, given the predictor value for that observation;
- $\mathbb{P}(\mathrm{X} = \mathrm{x}) = \sum_{k=1}^{K} \mathbb{P}(\mathrm{X} = \mathrm{x}|Y = k)\mathbb{P}(Y = k)$ is regarded as a constant w.r.t the response class j;
- $\mathbb{P}(\mathrm{X} = \mathrm{x}|Y = j)$ is called the **likelihood function**, a density function of X for an observation that comes from the jth class. It is relatively large if there is a high probability that an observation in the jth class has $\mathrm{X} \approx \mathrm{x}$.

Generative Models (cont)

The generative classifier derived from (2) is

$$\begin{aligned} h_D(\mathrm{x}) &= \operatorname*{argmax}_{j\in\{1,\dots,K\}} \mathbb{P}(Y = j|\mathrm{X} = \mathrm{x}) \\ &= \operatorname*{argmax}_{j\in\{1,\dots,K\}} \frac{\mathbb{P}(\mathrm{X} = \mathrm{x}|Y = j)\mathbb{P}(Y = j)}{\mathbb{P}(\mathrm{X} = \mathrm{x})} \\ &= \operatorname*{argmax}_{j\in\{1,\dots,K\}} \mathbb{P}(\mathrm{X} = \mathrm{x}|Y = j)\mathbb{P}(Y = j) \\ &= \operatorname*{argmax}_{j\in\{1,\dots,K\}} [\ln \mathbb{P}(\mathrm{X} = \mathrm{x}|Y = j) + \ln \mathbb{P}(Y = j)] \end{aligned} \tag{3}$$

Outline

1 Discriminative vs Generative Models

2 Naïve Bayes Classifiers

3 Bayesian Network

4 Discriminant Analysis Models

Naïve Bayes Classifiers

A **naïve Bayes classifier (NB)** (`https://en.wikipedia.org/wiki/Naive_Bayes_classifier`) is a simple probabilistic classifier (2) based on applying Bayes' theorem with **strong (naive) independence assumptions** on the likelihood function:

$$
\begin{aligned}
&\mathbb{P}(\mathrm{X} = \mathrm{x} | Y = j) \\
&= \mathbb{P}(X_1 = x_1, X_2 = x_2, ..., X_p = x_p | Y = j) \\
&= \mathbb{P}(X_1 = x_1 | Y = j) \times \cdots \times \mathbb{P}(X_p = x_p | Y = j) \\
&= \prod_{i=1}^{p} \mathbb{P}(X_i = x_i | Y = j).
\end{aligned}
$$

Naïve Bayes Classifiers (cont)

Therefore, the generative classifier (3) becomes

$$\begin{aligned} h_D(\mathrm{x}) &= \operatorname*{argmax}_{j\in\{1,\dots,K\}} \mathbb{P}(Y=j)\prod_{i=1}^{p}\mathbb{P}(X_i=x_i|Y=j) \\ &= \operatorname*{argmax}_{j\in\{1,\dots,K\}} \ln\mathbb{P}(Y=j)+\left[\sum_{i=1}^{p}\ln\mathbb{P}(X_i=x_i|Y=j)\right]. \end{aligned} \tag{4}$$

Naïve Bayes Classifiers (cont)

The estimate of the prior distribution $\mathbb{P}(Y = j)$ using MLE is as follows:

$$\mathbb{P}\widehat{(Y = j)} = \frac{\#\{i \ : \ y_i = j\}}{n}. \tag{5}$$

However, it is possible to choose

$$\mathbb{P}(Y = j) = \frac{1}{K}$$

if we know the outcome should be uniformly distributed.

Naïve Bayes Classifiers (cont)

The features X_i can be categorical or numeric:

- One X_i is categorical — Categorical NB
- All X_i are binary — Bernoulli NB
- All X_i are integral — Multinomial NB & Complement NB(?)
- One X_i is numeric — Gaussian NB

Categorical NB

The feature X_i

- is categorical;
- takes on M_i possible values: $\{c_1^{(i)}, \cdots, c_{M_i}^{(i)}\} =: \mathscr{C}_i$

The conditional distribution is assumed to be

$$\mathbb{P}(X_i = c | Y = j) = \frac{n_{X_i=c \ \& \ Y=j}}{n_{Y=j}}, \quad c \in \mathscr{C}_i. \tag{6}$$

Categorical NB (cont)

Final Exam May 2019, Q3(d) Example

Table Q3(d) shows a data set containing 7 observations with 3 categorical predictors, X_1, X_2 and X_3.

Observation	X_1	X_2	X_3	Y
1	C	No	0	Positive
2	A	Yes	1	Positive
3	B	Yes	0	Negative
4	B	Yes	0	Negative
5	A	No	1	Positive
6	C	No	1	Negative
7	B	Yes	1	Positive

Table Q3(d)

Without Laplace smoothing, predict the response, Y for an observation with $X_1 = \text{B}$, $X_2 = \text{Yes}$ and $X_3 = 1$ using Naïve Bayes approach. (5 marks)

Categorical NB (cont)

Solution: Let '+' denote 'Positive' and '-' denote 'Negative'.

prior, $\mathbb{P}(Y)$	$\mathbb{P}(X_1\|Y)$	$\mathbb{P}(X_2\|Y)$	$\mathbb{P}(X_3\|Y)$	prop, Π	$\hat{Y}$
$\mathbb{P}(+) = \frac{4}{7}$	$\mathbb{P}(\text{B}\|+) = \frac{1}{4}$	$\mathbb{P}(\text{Yes}\|+) = \frac{1}{2}$	$\mathbb{P}(1\|+) = \frac{3}{4}$	0.0536	
$\mathbb{P}(-) = \frac{3}{7}$	$\mathbb{P}(\text{B}\|-) = \frac{2}{3}$	$\mathbb{P}(\text{Yes}\|-) = \frac{2}{3}$	$\mathbb{P}(1\|-) = \frac{1}{3}$	0.0635	$\checkmark, -$

Since $\mathbb{P}(Y = -|X) > \mathbb{P}(Y = +|X)$, Y has higher probability to be "Negative".

Categorical NB (cont)

Consider the following case given in `https://machinelearningmastery.com/naive-bayes-tutorial-for-machine-learning/`

Weather	Car	Y
sunny	working	go-out
rainy	broken	go-out
sunny	working	go-out
sunny	working	go-out
sunny	working	go-out
rainy	broken	stay-home
rainy	broken	stay-home
sunny	working	stay-home
sunny	broken	stay-home
rainy	broken	stay-home

Construct the categorical Naïve Bayes model for the above data.

Categorical NB (cont)

Solution: Let X_1=Weather, X_2=Car. The categorical Naïve Bayes model:

$$\begin{aligned} &\mathbb{P}(Y=j|\mathrm{X}=\mathrm{x}) \\ \propto&\mathbb{P}(Y=j)\times\mathbb{P}(X_1=x_1|Y=j)\times\mathbb{P}(X_2=x_2|Y=j) \end{aligned}$$

where Prior, $\mathbb{P}(Y)=\begin{cases}0.5, & Y=out\\ 0.5, & Y=stay\end{cases}$

$$\mathbb{P}(X_1|Y=out)=\begin{cases}\frac{4}{5}, & X_1=sunny\\ \frac{1}{5}, & X_1=rainy\end{cases},$$

$$\mathbb{P}(X_1|Y=stay)=\begin{cases}\frac{2}{5}, & X_1=sunny\\ \frac{3}{5}, & X_1=rainy\end{cases}$$

Categorical NB (cont)

Solution (cont):

$$\mathbb{P}(X_2|Y = out) = \begin{cases} \frac{4}{5}, & X_2 = working \\ \frac{1}{5}, & X_2 = broken \end{cases},$$

$$\mathbb{P}(X_2|Y = stay) = \begin{cases} \frac{1}{5}, & X_2 = working \\ \frac{4}{5}, & X_2 = broken \end{cases}$$

Categorical NB (cont)

Implementation in Python: `CategoricalNB` The probability of category c in feature X_i given class j is estimated with "Laplace smoothing":

$$\mathbb{P}(X_i = c | Y = j;\ \alpha) = \frac{n_{X_i=c\ \&\ Y=j} + \alpha}{n_{Y=j} + \alpha d_i} \tag{7}$$

where α is a **smoothing parameter** and d_i is the number of available categories of feature X_i defined above. It has the following form:

```
from sklearn.naive_bayes import CategoricalNB
CategoricalNB(alpha=1.0, fit_prior=True,
  class_prior=None)
```

Multinomial NB

The Naïve Bayes algorithm for multinomially distributed data is called a *multinomial Naïve Bayes classifier*:

Application: **text classification**

$$\begin{aligned} & h_D(\texttt{document}) \\ =& \operatorname*{argmax}_{j=1,\cdots,K} \mathbb{P}(\texttt{document}|Y=j)\mathbb{P}(Y=j) \\ =& \operatorname*{argmax}_{j=1,\cdots,K} \mathbb{P}(wc_1, wc_2, \cdots, wc_p|Y=j)\mathbb{P}(Y=j) \end{aligned}$$

where wc_i is the number of times the word X_i, $i=1,\cdots,p$, occurred in the `document`, p is the size of the vocabulary.

Multinomial NB (cont)

Possible entries of "classes" for `document` are "scientific", "economic", "management", etc. A naïve estimate for $\mathbb{P}(Y=j)$ is

$$\begin{aligned}&\mathbb{P}(Y=j)\\ \approx&\frac{\text{number of documents of class } j}{\text{number of documents, } n};\\ &\mathbb{P}(X_i=wc_i|Y=j)\\ \approx&\frac{\begin{array}{l}\text{total number of the occurrences of}\\ \text{the word } X_i \text{ in documents of class } j\end{array}}{\text{total number of words } X_1,\cdots,X_p \text{ in documents of class } j} =: \theta_{ji}.\end{aligned}$$

Multinomial NB (cont)

A more robust estimate of the parameters $\theta_j := (\theta_{j1}, \ldots, \theta_{jp})$ is given by a smoothed version of MLE:

$$\mathbb{P}(X_i = wc_i | Y = j) \approx \frac{N_{ji} + \alpha}{N_j + \alpha d_i}$$

where $N_{ji} = \sum_{y_i=j} wc_i$ is the number of times feature i appears in a sample of class j in the training set D, and $N_j = \sum_{i=1}^{n} N_{ji}$ is the total count of all features for class j.

For the smoothing priors $\alpha \geq 0$,

$\alpha < 1$ is called *Lidstone smoothing*,

$\alpha = 1$ is called *Laplace smoothing*.

Multinomial NB (cont)

The conditional probability is

$$\mathbb{P}(\mathrm{X}=\mathrm{x}|Y=j)=\frac{(\sum_{i=1}^{p} wc_i)!}{wc_1!\times\cdots\times wc_p!}\prod_{i=1}^{p}\theta_{ji}^{wc_i}. \tag{8}$$

According to `https://scikit-learn.org/stable/modules/naive_bayes.html`, the Multinomial Naïve Bayes classifiers is implemented as `MultinomialNB`.

```
from sklearn.naive_bayes import MultinomialNB
MultinomialNB(alpha=1.0, fit_prior=True,
  class_prior=None)
```

Complement Multinomial NB

A *complement Naïve Bayes* (CNB) algorithm is an adaptation of the standard MNB algorithm that is particularly suited for imbalanced data sets.

The procedure for calculating the weights is as follows:

$$\widehat{\theta}_{ji} = \frac{\alpha_i + \sum_{k:y_j \neq j} d_{ij}}{\alpha + \sum_{k:y_j \neq j} \sum_s d_{sj}} \Rightarrow w'_{ji} = \ln \widehat{\theta}_{ji} \Rightarrow w_{ji} = \frac{w'_{ji}}{\sum_k |w'_{jk}|}$$

where the summations are over all documents k not in class j, d_{ij} is either the count or tf-idf value (term frequency-inverse document frequency, see `https://en.wikipedia.org/wiki/Tf-idf`) of term i in document j.

Complement Multinomial NB (cont)

In Python's Sklearn, CNB is implemented as `ComplementNB` and has the form:

```
from sklearn.naive_bayes import *
ComplementNB(alpha=1.0, fit_prior=True,
  class_prior=None, norm=False)
```

Bernoulli NB

Bernoulli Naïve Bayes is used when the data is distributed according to multivariate Bernoulli distributions i.e., x_i is a binary value.

The conditional probability is

$$\mathbb{P}(\mathrm{X} = \mathrm{x}|Y = j) = \prod_{i=1}^{p} \theta_{ji}^{x_i}(1 - \theta_{ji}^{x_i})^{1-x_i} \tag{9}$$

It is implemented in Python as `BernoulliNB`:

```
from sklearn.naive_bayes import *
BernoulliNB(alpha=1.0, binarize=0.0,
  fit_prior=True, class_prior=None)
```

Example

Real-world application is complex. There are a lot of information on the Internet, so I will not create a “fake” example, but just use the a Python example.

```
# https://jakevdp.github.io/PythonDataScienceHandbook/05.05-naive-bay
from sklearn.datasets import fetch_20newsgroups
# Downloads https://ndownloader.figshare.com/files/5975967 and
# put in under ~/scikit_learn_data/
data = fetch_20newsgroups()
data.target_names

categories = [’talk.religion.misc’, ’soc.religion.christian’,
              ’sci.space’, ’comp.graphics’]
train = fetch_20newsgroups(subset=’train’, categories=categories)
test  = fetch_20newsgroups(subset=’test’,  categories=categories)

from sklearn.feature_extraction.text import TfidfVectorizer,CountVect
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline
```

Example (cont)

```

# https://machinelearningmastery.com/prepare-text-data-machine-learni
# https://scikit-learn.org/stable/modules/feature_extraction.html
vectoriser = CountVectorizer()
vectoriser.fit(train.data)
print(vectoriser.vocabulary_)
feature_table = vectoriser.transform(train.data)
model = make_pipeline(TfidfVectorizer(), MultinomialNB())
model.fit(train.data, train.target)
labels = model.predict(test.data)

from sklearn.metrics import confusion_matrix
import seaborn as sns, matplotlib.pylab as plt
mat = confusion_matrix(test.target, labels)
sns.heatmap(mat.T, square=True, annot=True, fmt='d', cbar=False,
            xticklabels=train.target_names,
            yticklabels=train.target_names)
plt.xlabel('true label')
plt.ylabel('predicted label')
```

Gaussian Naïve Bayes

For continuous inputs X_i in (4), it is assume that X_i is Gaussian:

$$\mathbb{P}(X_i = x_i | Y = j) = \frac{1}{\sigma_j\sqrt{2\pi}} \exp(-\frac{(x_i - \mu_j)^2}{2\sigma_j^2}). \tag{10}$$

It is defined by a mean and standard deviation specific to X_i and j. The estimations of the mean and standard deviation are

$$\mu_j = \mathbb{E}[X_i | Y = j], \quad \sigma_j^2 = \mathbb{E}[(X_i - \mu_j)^2 | Y = j]. \tag{11}$$

Gaussian Naïve Bayes (cont)

The maximum likelihood estimator for (11) are

$$\begin{aligned}\widehat{\mu}_j &= \frac{1}{\sum_{i=1}^n I(y_i = j)} \sum_{k=1}^n X_{ki} I(Y = j),\\ s_j &= \frac{1}{(\sum_{i=1}^n I(y_i = j)) - 1} \sum_{k=1}^n (X_{ki} - \widehat{\mu}_j)^2 I(y_i = j).\end{aligned} \tag{12}$$

Gaussian Naïve Bayes (cont)

Example 8.3.2

The table below shows the data collected for predicting whether a customer will default on the credit card or not:

customer	balance	student	Default
1	500	No	N
2	1980	Yes	Y
3	60	No	N
4	2810	Yes	Y
5	1400	No	N
6	300	No	N
7	2000	Yes	Y
8	940	No	N
9	1630	No	Y
10	2170	Yes	Y

Gaussian Naïve Bayes (cont)

(a) Compute the probability density of customer with balance 2080, given Default=Y.
(b) Compute the probability of customer who is a student, given Default=Y.
(c) Calculate the “probability density” of default for a student customer with balance 2080 by using the Naïve Bayes assumption.

Gaussian Naïve Bayes (cont)

Note: This question just asks for specific answer without the full model, so we don't need to write the full model.

(a) **Solution**:

$$\begin{aligned}&\mathbb{P}(\texttt{balance} = 2080 \mid \texttt{Default} = Y)\\ =&\frac{1}{s_Y\sqrt{2\pi}}\exp(-\frac{(2080-\mu_Y)^2}{2s_Y^2}) = 0.0009162\end{aligned}$$

where $\mu_Y = \frac{1980+2810+2000+1630+2170}{5} = 2118$;
$s_Y = 433.7857$

Gaussian Naïve Bayes (cont)

(b) **Solution**: $\mathbb{P}(\texttt{student} = \mathit{Yes} \mid \texttt{Default} = Y) = \dfrac{4}{5}$

(c) **Solution**:

$\mathbb{P}(\texttt{student} = \mathit{Yes} \mid \texttt{Default} = N) = \frac{0}{5} = 0$

$$
\begin{aligned}
&\mathbb{P}(\texttt{Default} = Y \mid \texttt{balance} = 2080,\ \texttt{student} = \mathit{Yes}) \\
=&\frac{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = \mathit{Yes} \mid \texttt{Default} = Y)\mathbb{P}(\texttt{Default} = \texttt{Y})}{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = \mathit{Yes}) =: \mathbb{P}(\ldots)} \\
=&\frac{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = \mathit{Yes} \mid \texttt{Default} = Y)\mathbb{P}(\texttt{Default} = \texttt{Y})}{\mathbb{P}(\ldots \mid \texttt{Default} = Y)\mathbb{P}(\texttt{Default} = \texttt{Y}) + \mathbb{P}(\ldots \mid \texttt{Default} = N)\mathbb{P}(\texttt{Default} = \texttt{N})} \\
=&\frac{0.0009162 \times \frac{4}{5} \times \frac{5}{10}}{0.0009162 \times \frac{4}{5} \times \frac{5}{10} + \mathbb{P}(\texttt{balance} = 2080 | \texttt{Default} = N) \times 0 \times \frac{5}{10}} = 1
\end{aligned}
$$

Gaussian Naïve Bayes (cont)

Remark on Example 8.3.2: Let x_1=balance, x_2=student, y=Default. The Naive Bayes Model is

$$h_D(x_1, x_2) = \operatorname*{argmax}_j \mathbb{P}(x_1|y = j)\mathbb{P}(x_2|y = j)\mathbb{P}(y = j).$$

where the prior $\mathbb{P}(y) = \begin{cases} 0.5 & y = N \\ 0.5 & y = Y \end{cases}$

$$\mathbb{P}(x_2|y = N) = \begin{cases} 1 & x_2 = No \\ 0 & x_2 = Yes \end{cases}$$

$$\mathbb{P}(x_2|y = Y) = \begin{cases} 1/5 & x_2 = No \\ 4/5 & x_2 = Yes \end{cases}$$

Gaussian Naïve Bayes (cont)

Remark on Example 8.3.2 (cont):

$$\mathbb{P}(x_1|y=N) = \frac{1}{\sqrt{2\pi}(533.6666)} \exp\left\{-\frac{(x_1-640)^2}{2(533.6666)^2}\right\}$$

$$\mathbb{P}(x_1|y=Y) = \frac{1}{\sqrt{2\pi}(433.7857)} \exp\left\{-\frac{(x_1-2118)^2}{2(433.7857)^2}\right\}$$

Gaussian Naïve Bayes (cont)

Final Exam Jan 2019, Q3(c)

A more efficient marketing strategy can be achieved by targeting the customers who have higher probability to complete a purchase. Hence, you have been asked to predict whether a customer will buy the product based on their demographic data such as age, race, gender and income. Table Q3(c) shows the data collected from previous records.

Gaussian Naïve Bayes (cont)

Final Exam Jan 2019, Q3(c) cont

Cust.	Age	Race	Gender	Income	Result
1	52	Indian	Male	RM 11,500	Not Buy
2	22	Chinese	Female	RM 6,500	Buy
3	30	Chinese	Male	RM 8,000	Buy
4	26	Malay	Male	RM 8,500	Buy
5	27	Indian	Female	RM 6,500	Buy
6	32	Chinese	Female	RM 9,500	Not Buy
7	33	Indian	Male	RM 4,000	Not Buy
8	50	Malay	Female	RM 10,000	Buy
9	31	Chinese	Female	RM 5,500	Buy
10	27	Malay	Male	RM 9,200	Not Buy

Table Q3(c)

Gaussian Naïve Bayes (cont)

Final Exam Jan 2019, Q3(c) cont

(i) State an assumption used in Naïve Bayes approach. (1 mark)

(ii) Using Naïve Bayes approach without Laplace smoothing, predict whether a Malay female customer, aged 29, with income RM7,800, will buy the product. (9 marks)

Gaussian Naïve Bayes (cont)

Gaussian Naïve Bayes (Classifier) is available in Python as `GaussianNB` of the form:

```
from sklearn.naive_bayes import GaussianNB
GaussianNB(priors=None, var_smoothing=1e-09)
```

All the above mentioned naïve Bayes models are available in R except the complement NB. R provides unified functions such as `naivebayes::naive_bayes`, `e1071::naiveBayes`, `bnlearn::naive.bayes` (which can only handle categorical data), `klaR::NaiveBayes`.

```
naive_bayes(formula, data, prior = NULL, laplace = 0,
  usekernel = FALSE, usepoisson = FALSE,
  subset, na.action = stats::na.pass, ...)
```

Gaussian Naïve Bayes (cont)

Example 8.3.5: The Iris dataset (`https://archive.ics.uci.edu/ml/datasets/Iris`) consists four attributes: sepal length, sepal width, petal length and petal width, all in cm; and a label of 3 classes (Iris Setosa, Iris Versicolour, Iris Virginica).

```
# https://www.python-course.eu/naive_bayes_classifier_scikit.php
from sklearn import datasets, metrics, naive_bayes
dataset = datasets.load_iris()
model = naive_bayes.GaussianNB()
model.fit(dataset.data, dataset.target)
print(model)
expected  = dataset.target
predicted = model.predict(dataset.data)
# summarise the fit of the model
print(metrics.confusion_matrix(predicted, expected))
print(metrics.classification_report(predicted, expected))
```

Laplace Smoothing

An issue faced by a Naïve Bayes classifier with “discrete” data is the numerator in (6) being zero, i.e. $n_{X=c,Y=j} = 0$. In this case, the posterior probability will become zero regardless of the value of other density functions and the Naïve Bayes classifier will fail.

Laplace Smoothing (cont)

Example 8.4.1: By using the data from Example 8.3.2, perform the following tasks.

(a) Compute the probability density of customer with balance 2080, given Default=N.

(b) Compute the probability of customer who is a student, given Default=N.

(c) Calculate the “probability density” of non-default for a student customer with balance 2080 by using the Naïve Bayes assumption.

Laplace Smoothing (cont)

(a) Solution: $\mathbb{P}(\texttt{balance} = 2080 \mid \texttt{Default} = N) = \frac{1}{s_N\sqrt{2\pi}}\exp(-\frac{(2080-\mu_N)^2}{2s_N^2}) = 1.9616 \times 10^{-5}$

where $\mu_N = \frac{500+60+1400+300+940}{5} = 640; \quad s_N = 533.6666$

(b) Solution:
$\mathbb{P}(\texttt{student} = \mathit{Yes} \mid \texttt{Default} = N) = \frac{0}{5} = 0$

(c) Solution:

$$\mathbb{P}(\texttt{Default} = N \mid \texttt{balance} = 2080,\ \texttt{student} = \mathit{Yes})$$
$$= \frac{1.9616 \times 10^{-5} \times 0 \times \frac{5}{10}}{\mathbb{P}(\texttt{balance} = 2080,\ \texttt{student} = \mathit{Yes})} = 0.$$

Laplace Smoothing (cont)

This situation is normally happened to categorical variable. To avoid the stated problem, *Laplace smoothing* or `https://en.wikipedia.org/wiki/Additive_smoothing` is applied to the Naïve Bayes classifier. Laplace smoothing modified the density function of categorical variable by adding α (by default, $\alpha = 1$) to each variable per class:

$$\mathbb{P}(X = x_i | Y = k) = \frac{n_{X=x_i;Y=k} + \alpha}{n_{Y=k} + d\alpha} \tag{13}$$

where d is the number of classes in the categorical variable X.

Laplace Smoothing (cont)

Example 8.4.2: Redo Example 8.4.1 by applying Laplace smoothing.

Solution: The "continuous" variable is the same:

$$\mathbb{P}(\texttt{balance} = 2080 \mid \texttt{Default} = N) = 1.9616 \times 10^{-5}$$

The categorical variable needs the Laplace smoothing ($\alpha = 1$, $d = 2$ for $\texttt{student}$=Yes or No)

$$\mathbb{P}(\texttt{student} = Yes \mid \texttt{Default} = N) = \frac{0+1}{5+2}$$

Laplace Smoothing (cont)

Example 8.4.2 cont.

Therefore,

$$\begin{aligned}
&\mathbb{P}(\texttt{Default} = N \mid \texttt{balance} = 2080,\ \texttt{student} = Yes) \\
=&\frac{\mathbb{P}(2080, Yes \mid N)\mathbb{P}(N)}{\mathbb{P}(2080, Yes \mid N)\mathbb{P}(N) + \mathbb{P}(2080, Yes \mid Y)\mathbb{P}(Y)} \\
=&\frac{1.9616 \times 10^{-5} \times \frac{1}{7} \times \frac{5}{10}}{1.9616 \times 10^{-5} \times \frac{1}{7} \times \frac{5}{10} + 0.000916154 \times \frac{5}{7} \times \frac{5}{10}} \\
=&\frac{1.401151e-06}{1.401151e-06 + 0.0003271979} = 0.004264014
\end{aligned}$$

Outline

1 Discriminative vs Generative Models

2 Naïve Bayes Classifiers

3 Bayesian Network

4 Discriminant Analysis Models

Bayesian Network

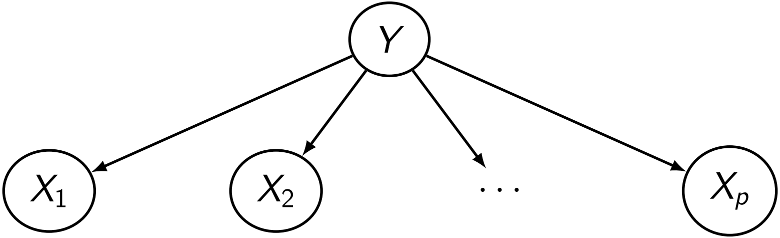
Things not coming out in Final Assessment:

> *Computations related to Bayesian Network, Ensemble Methods, SVM*

A *Bayesian network* (or *belief network*) $B = \langle N, A, \Theta \rangle$ is a directed acyclic graph (DAG) $\langle N, A \rangle$ with a conditional probability distribution for each node, collectively represented by Θ. Each node $n \in N$ represents a random variable X_i (which represents a feature) and each arc $a \in A$ represents a probabilistic dependency between features.

Bayesian Network (cont)

The Naïve Bayes classifier is a special class of Bayesian networks with the following DAG:



Bayesian Network (cont)

The statistical meaning of this Bayesian network can be described by the conditional independence set $Q = \{(X_i, pa(X_i), nde(X_i)|i = 1, \cdots, p\}$ where $pa(x)$ is the set of all parents of node x, $de(x)$ is the set of all descendents of node x, and $nde(x) = N \setminus \{x\} \setminus pa(x) \setminus de(x)$ is the set of all non-descendants of node x. The joint probability distribution of $X_1, \cdots, X_p$ can be factored based on the probability decomposition of the Bayesian network:

$$\mathbb{P}(X_1, \cdots, X_p) = \prod_{i=1}^{p} \mathbb{P}(X_i|pa(X_i)). \tag{14}$$

Bayesian Network (cont)

This means that the “Probability Distribution Table” (on the left of (14)) can be factored into a product of much smaller Conditional Probability Tables (on the right of (14)).

Bayesian networks explicitly express conditional independencies in probability distributions and allows computation of probabilities distributions using the chain rule:

$$\mathbb{P}(A|B,C) = \mathbb{P}(A|C), \quad \mathbb{P}(A,B|C) = \mathbb{P}(A|C)\mathbb{P}(B|C)$$

where A and B are **conditionally independent** given C.

Outline

1. Discriminative vs Generative Models

2. Naïve Bayes Classifiers

3. Bayesian Network

4. Discriminant Analysis Models

Discriminant Analysis (DA) Theory (cont)

Naive Bayes is a “parametric” & generative model.

Discriminant analysis is another “parametric” & generative model.

The difference: the “assumptions” on the likelihood functions.

Discriminant analysis assumes that $\mathbb{P}(\mathrm{X} = \mathrm{x} | Y = k)$ is approximated by multivariate Gaussian distribution:

$$f_j(\mathrm{x}) = \frac{1}{(2\pi)^{p/2}\sqrt{|\mathrm{C}_j|}} \exp\left\{-\frac{1}{2}(\mathrm{x} - \boldsymbol{\mu}_j)^T \mathrm{C}_j^{-1}(\mathrm{x} - \boldsymbol{\mu}_j)\right\} \tag{15}$$

DA Theory (cont)

DA models are generative predictive models (3) which assumes that the predictors $\mathrm{X} = (X_1, X_2, \cdots, X_p)$ to be numeric and are drawn from a *multivariate Gaussian* (or a *multivariate normal*) distribution, $\mathit{Normal}(\boldsymbol{\mu}, \mathrm{C})$. Therefore, the "$\mathbb{P}$" in (3) should be regarded as "probability density" because the predictors are numeric.

DA Theory (cont)

Two important DA models are the *linear discriminant analysis (LDA)* models and *quadratic discriminant analysis (QDA)* models.

In the QDA models, $\mathbb{P}(\mathsf{X} = \mathsf{x} | Y = j)$ is assumed to have the following form:

$$\mathbb{P}(\mathsf{X} = \mathsf{x} | Y = j) = \frac{1}{(2\pi)^{p/2}\sqrt{|\mathsf{C}_j|}} \exp\left\{-\frac{1}{2}(\mathsf{x} - \boldsymbol{\mu}_j)^T \mathsf{C}_j^{-1} (\mathsf{x} - \boldsymbol{\mu}_j)\right\} \tag{16}$$

where $\boldsymbol{\mu}$ and C_j are the class-specific mean vector and the class-specific covariance matrix respectively.

QDA Theory

By substituting (16) into (3), we have

$$\begin{aligned} h_D(\mathrm{x}) &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \Bigg[\ln \mathbb{P}(Y=j) + \ln \frac{1}{(2\pi)^{p/2}\sqrt{|\mathrm{C}_j|}} \times \\ &\qquad \exp\left\{ -\frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_j)^T \mathrm{C}_j^{-1}(\mathrm{x}-\boldsymbol{\mu}_j) \right\} \Bigg] \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \Bigg[\ln \mathbb{P}(Y=j) - \frac{1}{2}\ln|\mathrm{C}_j| - \\ &\qquad \frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_j)^T \mathrm{C}_j^{-1}(\mathrm{x}-\boldsymbol{\mu}_j) \Bigg] \end{aligned} \tag{17}$$

QDA Theory (cont)

From here, we can obtain the “discriminant functions” for QDA:

$$\delta_j(\mathsf{x}) = \ln \mathbb{P}(Y = j) - \frac{1}{2} \ln |\mathsf{C}_j| - \frac{1}{2}(\mathsf{x} - \boldsymbol{\mu}_j)^T \mathsf{C}_j^{-1}(\mathsf{x} - \boldsymbol{\mu}_j) \tag{18}$$

where $j = 1, \cdots, K$.

Note that the “boundary” between classes is a quadratic surface according to (18).

QDA Theory (cont)

Parameter Estimation

The prior distribution

$$\mathbb{P}(\widehat{Y = j}) = \frac{\#\{i \ : \ y_i = j\}}{n}$$

The class-specific covariance matrix

$$\widehat{\mathrm{C}}_j = \frac{1}{n_j - 1} \sum_{i=1}^{n} (\mathrm{x}_i - \widehat{\boldsymbol{\mu}}_j)(\mathrm{x}_i - \widehat{\boldsymbol{\mu}}_j)^T I(y_i = j). \qquad (19)$$

LDA Theory

In the LDA models, $\mathbb{P}(\mathrm{X} = \mathrm{x} | Y = j)$ is assumed to be (15) with a common covariance matrix C:

$$\mathrm{C}_1 = \mathrm{C}_2 = \cdots = \mathrm{C}_K =: \mathrm{C}.$$

In such a case, the generative model (17) becomes (let $\pi_j = \mathbb{P}(Y = j)$)

$$\begin{aligned} h_D(\mathrm{x}) &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \left[\ln \pi_j - \frac{1}{2}\ln|\mathrm{C}| - \frac{1}{2}(\mathrm{x}-\boldsymbol{\mu}_j)^T\mathrm{C}^{-1}(\mathrm{x}-\boldsymbol{\mu}_j)\right] \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \left[\ln \pi_j - \frac{1}{2}\left(\mathrm{x}^T\mathrm{C}^{-1}\mathrm{x} - \boldsymbol{\mu}_j^T\mathrm{C}^{-1}\mathrm{x} - \mathrm{x}^T\mathrm{C}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^T\mathrm{C}^{-1}\boldsymbol{\mu}_j\right)\right] \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \left[\ln \pi_j - \frac{1}{2}\left(-2\boldsymbol{\mu}_j^T\mathrm{C}^{-1}\mathrm{x} + \boldsymbol{\mu}_j^T\mathrm{C}^{-1}\boldsymbol{\mu}_j\right)\right] \\ &= \underset{j\in\{1,\dots,K\}}{\operatorname{argmax}} \left[\ln \pi_j + \boldsymbol{\mu}_j^T\mathrm{C}^{-1}\mathrm{x} - \frac{1}{2}\boldsymbol{\mu}_j^T\mathrm{C}^{-1}\boldsymbol{\mu}_j\right]. \end{aligned} \tag{20}$$

LDA Theory (cont)

From here, we obtain the *discriminant function* for LDA:

$$\delta_j(\mathrm{X}) = \ln \mathbb{P}(Y = j) - \frac{1}{2}\boldsymbol{\mu}_j^T \mathrm{C}^{-1}\boldsymbol{\mu}_j + \boldsymbol{\mu}_j^T \mathrm{C}^{-1}\mathrm{x}. \tag{21}$$

Note that the “boundary” between classes is a linear space according to (21). The name of LDA is from the fact that the discriminant functions, $\delta_j(\mathrm{x})$ are linear functions of x.

LDA Theory (cont)

The estimation of the parameters in the discriminant function (21) using the **standard moment estimators** is as follows.

- The prior probability π_j is approximated by the fraction of training samples of class j:

$$\widehat{\pi}_j = \frac{\sum_{i=1}^{n} I(y_i = j)}{n}. \tag{22}$$

- The centre of each class $\boldsymbol{\mu}_j$ has an estimation:

$$\widehat{\boldsymbol{\mu}}_j = \sum_{i=1}^{n} \mathrm{x}_i I(y_i = j) \Big/ \sum_{i=1}^{n} I(y_i = j). \tag{23}$$

LDA Theory (cont)

- The common covariance matrix C is an unbiased estimate of its covariance matrix of the vectors of deviations $(\mathrm{x}_1 - \widehat{\boldsymbol{\mu}}_{y_1})$, $(\mathrm{x}_2 - \widehat{\boldsymbol{\mu}}_{y_2})$, $\cdots$, $(\mathrm{x}_n - \widehat{\boldsymbol{\mu}}_{y_n})$:

$$\widehat{\mathrm{C}} = \frac{1}{n-K}\sum_{j=1}^{K}\sum_{i=1}^{n}(\mathrm{x}_i - \widehat{\boldsymbol{\mu}}_j)(\mathrm{x}_i - \widehat{\boldsymbol{\mu}}_j)^T I(y_i = j) \quad (24)$$

LDA Theory (cont)

LDA method has the following advantages:

- When the classes are well-separated, it will be more stable as compared to other models.
- If *n* is small and the distribution of the predictors X is approximately normal in each of the classes, LDA is more stable than logistic regression.
- When we have more than two response classes, LDA is more popular.

LDA Examples

Final Exam May 2019, Q4

(a) State the advantages of using linear discriminant analysis in classification as compared to logistic regression. (3 marks)

Solution:

- LDA is more stable when the classes are well-separated.
- LDA is more stable when n is small and the distribution of the predictors X is approximately normal in each of the classes.
- LDA is better when there are more than two response classes.

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

- (b)
  - (i) State two assumptions made in linear discriminant analysis. (2 marks)
  - (ii) Bayes' theorem states that posterior probability to estimate probability of a new observation belongs to the jth class can be written as

$$\mathbb{P}(Y = j|X = x) = \frac{\pi_j \mathbb{P}(x|Y = j)}{\sum_{i=1}^{K} \pi_i \mathbb{P}(x|Y = i)}.$$

where $\pi_j = \mathbb{P}(Y = j)$. Classification is done by assigning an observation to the class which posterior probability, $\mathbb{P}(Y = j|X = x)$ is the largest. (continue next)

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

(b) (ii) (cont) Linear discriminant analysis involves assigning the observation to the class for which discriminant function, $\delta_j(X)$ is the largest. For linear discriminant analysis with one predictor, the discriminant function is

$$\delta_j(x) = x \cdot \frac{\mu_j}{\sigma^2} - \frac{\mu_j^2}{2\sigma^2} + \ln(\pi_j).$$

With assumptions stated in Q4(b)(i), show how discriminant function, $\delta_j(x)$ can be equivalent to posterior probability, $\mathbb{P}(Y = j | X = x)$ in linear discriminant analysis with one predictor.

(10 marks)

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

(c) A teacher is preparing an extra class for the students who are predicted to fail in their final exam. The teacher would like to predict the performance of the current students in final exam (fail/pass). You are to build a model for the teacher by using 500 previous students' record. Below shows some information and analysis of the previous record:

- (I) The coursework consisted of Assignment, Quiz and Test.
- (II) Average mark for students who passed in Assignment was 73.9; whereas average mark for students who failed in Assignment was 51.4.

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

(c)
- (III) Average mark for students who passed in Quiz was 68.2; whereas average mark for students who failed in Quiz was 42.3.
- (IV) Average mark for students who passed in Test was 63.7; whereas average mark for students who failed in Test was 35.6.
- (V) There were 380 students passed the final exam.
- (VI) The inverse of the group covariance matrix for the collected data is

$$C^{-1} = \begin{bmatrix} 0.0022 & 0.0132 & 0.0095 \\ 0.0132 & 0.0074 & 0.0108 \\ 0.0095 & 0.0108 & 0.0180 \end{bmatrix}$$

where x_1 = Assignment; x_2 = Quiz; x_3 = Test.

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

(c) Using linear discriminant analysis, predict the final exam performance (pass/fail) of a current student who scored 55.7 marks in Assignment, 49.8 marks in Quiz and 52.6 marks in Test. (10 marks)

Solution: Let pass $= 1$ and fail $= 0$.

Prior probability (given in item (V) and using (5)):

$$\widehat{\pi}_1 = \frac{380}{500} = 0.76; \quad \widehat{\pi}_0 = 1 - 0.76 = 0.24$$

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

(c) **Solution** (cont):
Mean vector (given in items (II), (III) and (IV) and calculate using (23)):

$$\widehat{\mu}_1 = [73.9,\ 68.2,\ 63.7]; \quad \widehat{\mu}_0 = [51.4,\ 42.3,\ 35.6]$$

Discriminant function, $\delta_j(X)$:

$$\widehat{\delta}_j(X) = \widehat{\mu}_j C^{-1} \mathrm{x}^T - \frac{1}{2}\widehat{\mu}_j C^{-1} \widehat{\mu}_j^T + \ln \pi_j,$$

$$\widehat{\delta}_1(X) = \ln(0.76) - 435.8066 + [1.6680,\ 2.1681,\ 2.5852]\mathrm{x}^T$$
$$\widehat{\delta}_0(X) = \ln(0.24) - 166.5589 + [1.0096,\ 1.3760,\ 1.5859]\mathrm{x}^T$$

LDA Examples (cont)

Final Exam May 2019, Q4 cont.

(c) **Solution** (cont): For a new observation, $\mathrm{x}^* = [55.7,\ 49.8,\ 52.6]$,

$$\widehat{\delta}_1(\mathrm{x}^*) = 118.6826$$
$$\widehat{\delta}_0(\mathrm{x}^*) = 123.4746$$

Since $\delta_1(\mathrm{x}^*) < \delta_0(\mathrm{x}^*)$, the new observation should be assigned to class 0, the student will "more likely to" fail in final exam.

LDA Examples (cont)

Example 8.8.3:

Table below shows the data collected:

Customer	Balance	Default
1	500	N
2	1980	Y
3	60	N
4	2810	Y
5	1400	N
6	300	N
7	2000	Y
8	940	N
9	1630	Y
10	2170	Y

LDA Examples (cont)

Example 8.8.3 cont.

Use the data and the predictive model LDA to predict if a customer with balance 1500 will default in his credit card?

Are you able to obtain these?

$$\delta_1(1500) = 3.2565, \quad \delta_2(1500) = 2.5003.$$

LDA Implementations

LDA and QDA are implemented:

- in R's `MASS` package as `lda` and `qda` respectively.
- in Python's `sklearn.discriminant_analysis` as `LinearDiscriminantAnalysis` and `QuadraticDiscriminantAnalysis`.

LDA Implementations (cont)

Example 8.8.4:

Redo Example 8.8.3 using R. Are you able to find the discriminant functions?

The "boundary" according to Example 8.8.3 is

$$-10.17773 + 0.008956171x = -1.559164 + 0.002706303x$$
$$\Rightarrow x = 1379$$

LDA Implementations (cont)

Example 8.8.4 cont.:

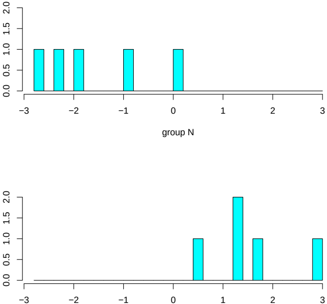
```
library(MASS)  # for lda

d.f = data.frame(
    balance = c(500 , 1980, 60  , 2810, 1400, 300 , 2000, 940 , 1630,
    default = c("N", "Y", "N", "Y", "N", "N", "Y", "N", "Y", "Y")
)
m = lda(default ~ ., data = d.f)
print(m); plot(m)

deltaN = log(m$prior[1]) - 0.5*m$scaling*m$means[1]**2
deltaY = log(m$prior[2]) - 0.5*m$scaling*m$means[2]**2
cat("deltaN(x) =", deltaN, "+", m$scaling*m$means[1], "x\n")
cat("deltaY(x) =", deltaY, "+", m$scaling*m$means[2], "x\n")
cat("boundary=",-(deltaN - deltaY)/(m$scaling*(m$means[1]-m$means[2])
```

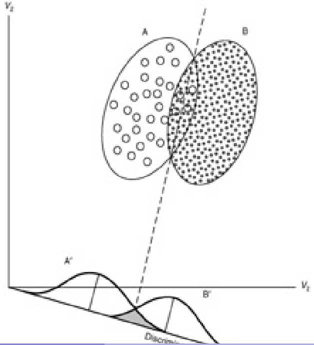
The "discriminant functions" and "boundary" according to MASS::lda are (?) $\delta_N(x) = -421.8348 + 1.316068x, \quad \delta_Y(x) = -4613.02 + 4.355361x \Rightarrow x = 1379$

LDA Implementations (cont)

Example 8.8.4 cont.:

The graph on the left below shows the set of histograms of the defaults on the Fisher linear discriminant. It can be regarded as the projection of the high-dimensional data as histogram.





LDA Implementations (cont)

Example 8.8.5: On the “flame” data.

```
d.f = read.table("flame.txt", header=FALSE)
d.f$V3 = factor(d.f$V3)
m = MASS::lda(V3 ~ ., d.f)

### m$scaling is not enough to obtain ‘‘discriminant functions’’
#d1 = log(m$prior[1])-0.5*sum(m$scaling*(m$means[,1]**2))
#d2 = log(m$prior[2])-0.5*sum(m$scaling*(m$means[,2]**2))
#coeff = m$scaling*(m$means[,1]-m$means[,2])
## Equation: d1 - d2 + coeff1 x1 + coeff2 x2 = 0
## => x2 = (d2-d1)/coeff2 + coeff1/coeff2
#a = (d2-d1)/coeff[2]; b = -coeff[1]/coeff[2]
#plot(d.f$V1, d.f$V2, pch=as.integer(d.f[,3]),
#       xlab="x1",ylab="x2",main="Flame Data")
#abline(a,b)
```

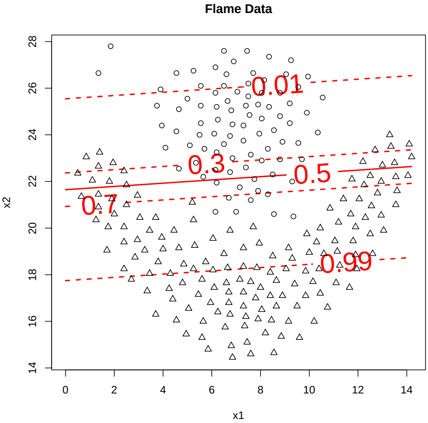
LDA Implementations (cont)

Example 8.8.5 cont.:

```r
g.x1 = seq(0,max(d.f[,1]),by=0.1)
g.x2 = seq(min(d.f[,2]),max(d.f[,2]),by=0.1)
d.grid = expand.grid(V1=g.x1, V2=g.x2)
prob = predict(m, newdata=d.grid)
prob = matrix(prob$posterior[,2], length(g.x1), length(g.x2))
contour(g.x1, g.x2, prob, levels=c(0.01,0.3,0.5,0.7,0.99),
  col="red", labcex=2.5, xlab="x1",ylab="x2",
  main="Flame Data",lty=c(2,2,1,2,2),lw=2)
points(d.f$V1, d.f$V2, pch=as.integer(d.f[,3]),
  xlab="x1",ylab="x2",main="Flame Data")
```

LDA Implementations (cont)

Example 8.8.5 cont.:


Flame Data
x1
x2