

Classification: probabilistic classifiers

生醫光電所 吳育德

Outlines

- You have learned, for a model and based on observed data, we can
 1. Find parameters that minimize a loss function,
 2. Find parameters that maximize a likelihood function
 3. Treat parameters as random variables and using Bayes' rule and make predictions.
- You will learn probabilistic classifiers
 1. Bayes classifier and Naïve Bayes classifier
 2. Logistic regression
- And non-probabilistic Classifiers
 3. K-nearest neighbors
 4. Support vector machines and other kernel methods

The General Classification Problem

- Given a set of N training samples, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, each is D -dimensional.

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \cdots & \mathbf{x}_{1,D} \\ \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \cdots & \mathbf{x}_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{N,1} & \mathbf{x}_{N,2} & \cdots & \mathbf{x}_{N,D} \end{bmatrix} \text{ or } \begin{bmatrix} \mathbf{x}_1^{(1)} & \mathbf{x}_2^{(1)} & \cdots & \mathbf{x}_D^{(1)} \\ \mathbf{x}_1^{(2)} & \mathbf{x}_2^{(2)} & \cdots & \mathbf{x}_D^{(2)} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_1^{(N)} & \mathbf{x}_2^{(N)} & \cdots & \mathbf{x}_D^{(N)} \end{bmatrix}$$

- Each \mathbf{x}_i is labelled by $t_i \in \{1, 2, \dots, C\}$, $\mathbf{t} = \begin{bmatrix} t_1 \\ \vdots \\ t_N \end{bmatrix}$
- Predict the class t_{new} for an unseen new sample \mathbf{x}_{new} based on \mathbf{X} and \mathbf{t} .
- In classification the response variable is an integer rather than a real value.

Probabilistic Classifiers

- The output is the probability of a new object belonging to a particular class:

$$p(T_{new} = c | \mathbf{x}_{new}, \mathbf{X}, t)$$

- As a probability it must satisfy:

$$0 \leq p(T_{new} = c | \mathbf{x}_{new}, \mathbf{X}, t) \leq 1$$

$$\sum_{c=1}^C p(T_{new} = c | \mathbf{x}_{new}, \mathbf{X}, t) = 1$$

- In many applications, the probability is useful, as it provides a level of **confidence** in the output

We need to know

- What is the multivariate Gaussian distribution $N(\mathbf{x}; \mu, \Sigma)$?
- How to compute the mean μ , and the covariance Σ
 - Using maximizing the likelihood function
 - Empirical mean and covariance
- How to make prediction $p(t_{new} = c | \mathbf{x}_{new}, \mathbf{X}_c, \mathbf{t}_c), c = 1, 2, 3$
 - Using the Bayes' rule

$$p(A|B) = \frac{p(A \cap B)}{p(B)} = \frac{p(B|A)p(A)}{p(B)}$$

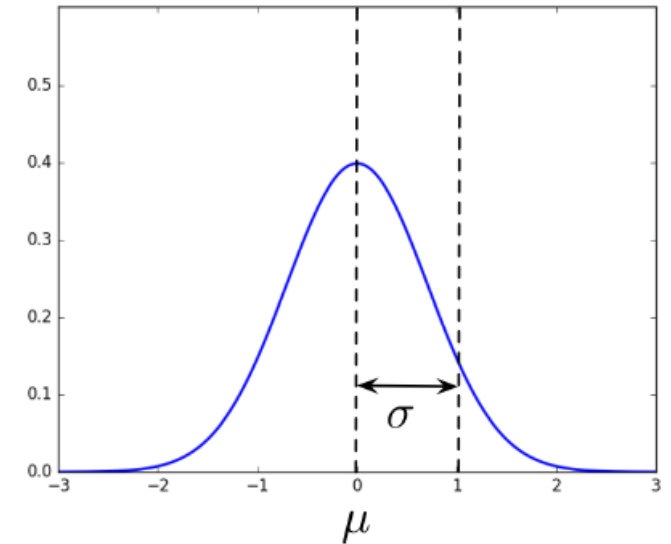
$$\rightarrow p(t_{new} = c | \mathbf{x}_{new}, \mathbf{X}, \mathbf{t}) = \frac{p(\mathbf{x}_{new} | t_{new} = c, \mathbf{X}_c, \mathbf{t}_c) p(t_{new} = c | \mathbf{X}_c, \mathbf{t}_c)}{p(\mathbf{x}_{new} | \mathbf{X}, \mathbf{t})}$$

Gaussian Distribution

Gaussian Distribution

- Recall the **Gaussian**, or **normal**, distribution:

$$N(x; \mu, \sigma^2) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

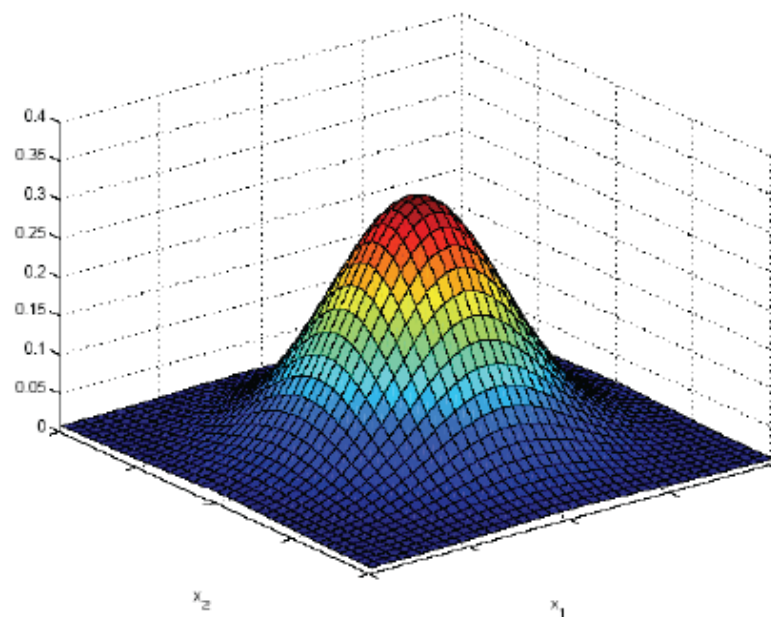


- In machine learning, we use Gaussians a lot because they make the calculations easy.

Multivariate Gaussian Distribution

- Multivariate Gaussian Distribution $\mathbf{x} \sim N(\mu, \Sigma)$, or $N(\mathbf{x}; \mu, \Sigma)$ is

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x} - \mu)^T \Sigma^{-1} (\mathbf{x} - \mu)\right]$$



Multivariate parameters

- Mean: $E[\mathbf{x}] = [\mu_1, \dots, \mu_d]^T$

- Covariance

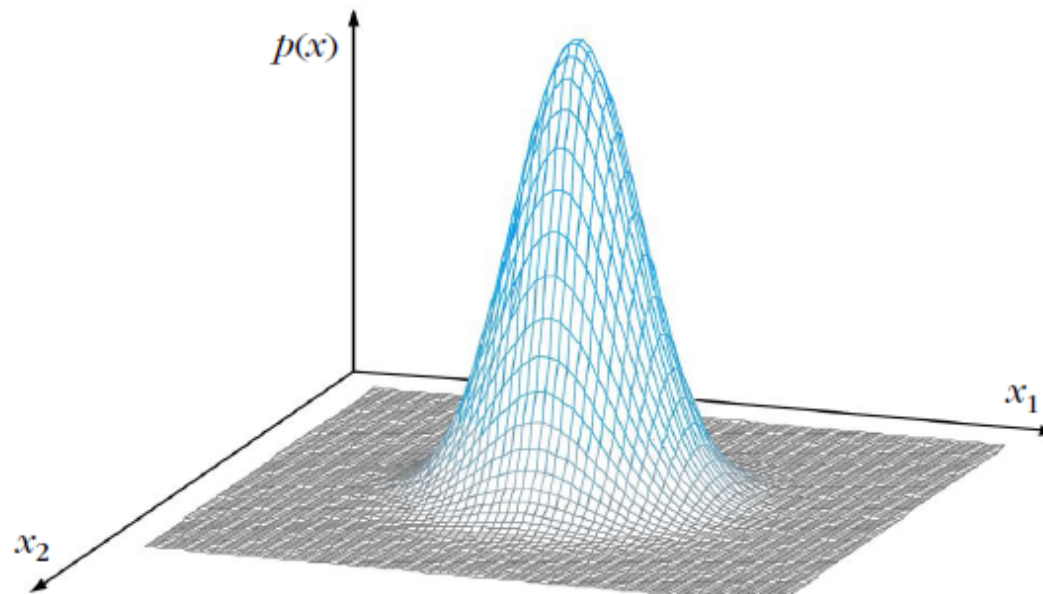
$$\Sigma = \text{Cov}(\mathbf{x}) = E[(\mathbf{x} - \mu)(\mathbf{x} - \mu)^T] = \begin{bmatrix} \sigma_1^2 & \sigma_{12} & \cdots & \sigma_{1d} \\ \sigma_{21} & \sigma_2^2 & \cdots & \sigma_{2d} \\ \vdots & \vdots & \ddots & \vdots \\ \sigma_{d1} & \sigma_{d2} & \cdots & \sigma_d^2 \end{bmatrix}$$

- For Gaussians - all you need to know is mean and covariance

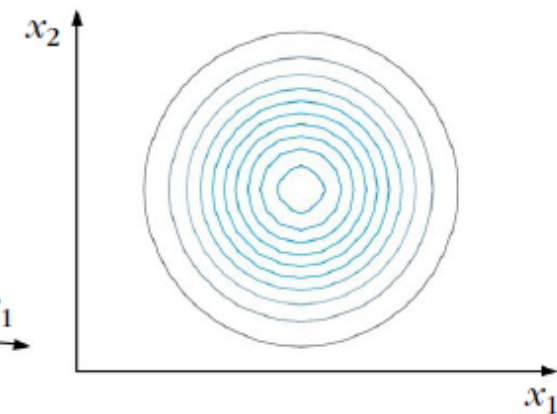
2D Gaussian pdf, diagonal Σ with $\sigma_1^2 = \sigma_2^2$

$$\Sigma = \begin{bmatrix} 3 & 0 \\ 0 & 3 \end{bmatrix} \quad (x - \mu)^T \Sigma^{-1} (x - \mu) = [x_1 \ x_2] \begin{bmatrix} \frac{1}{\sigma_1^2} & 0 \\ 0 & \frac{1}{\sigma_2^2} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = C$$

$$\frac{x_1^2}{\sigma_1^2} + \frac{x_2^2}{\sigma_2^2} = C$$

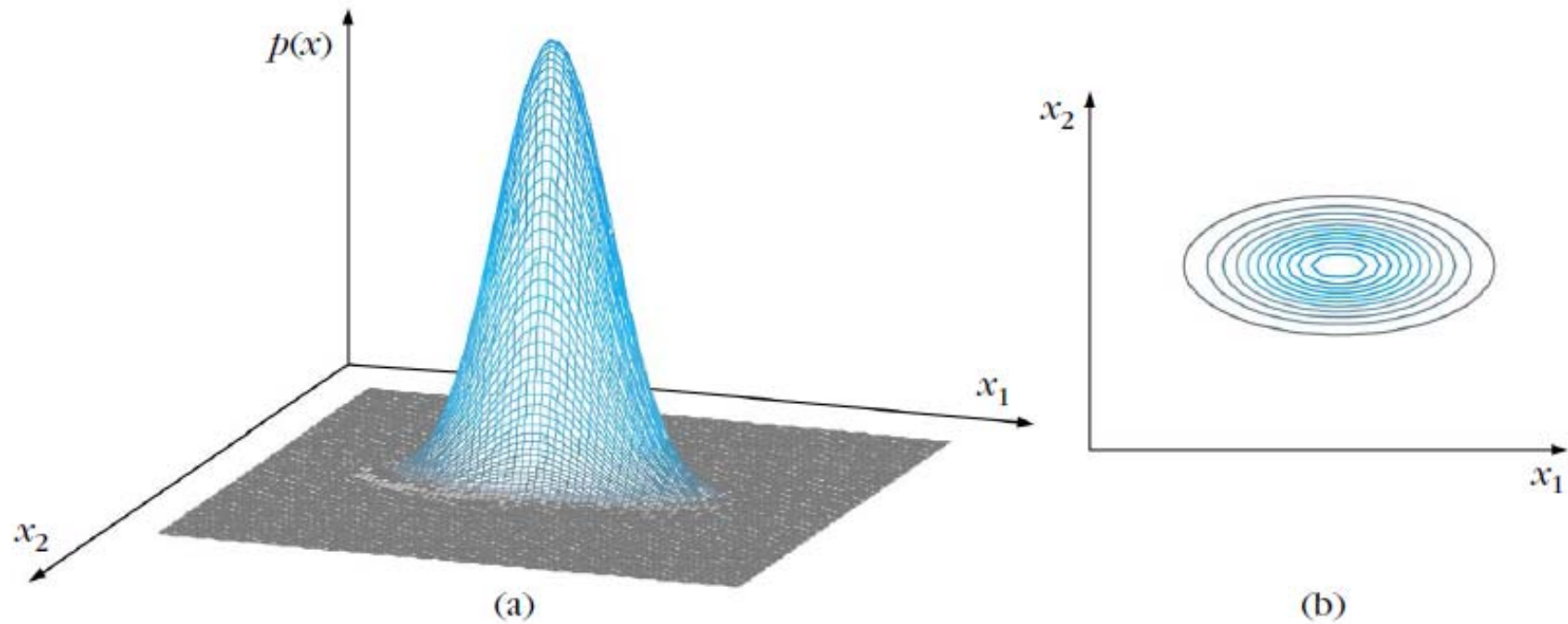


(a)

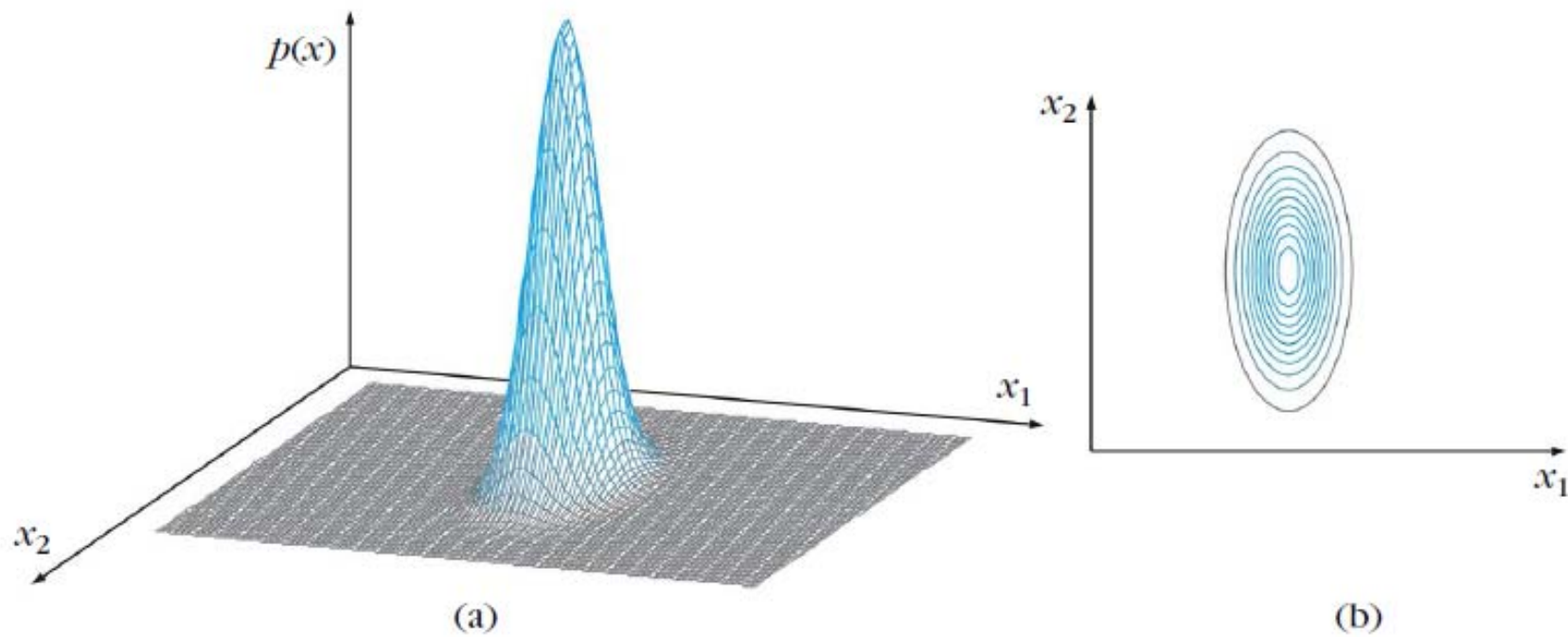


(b)

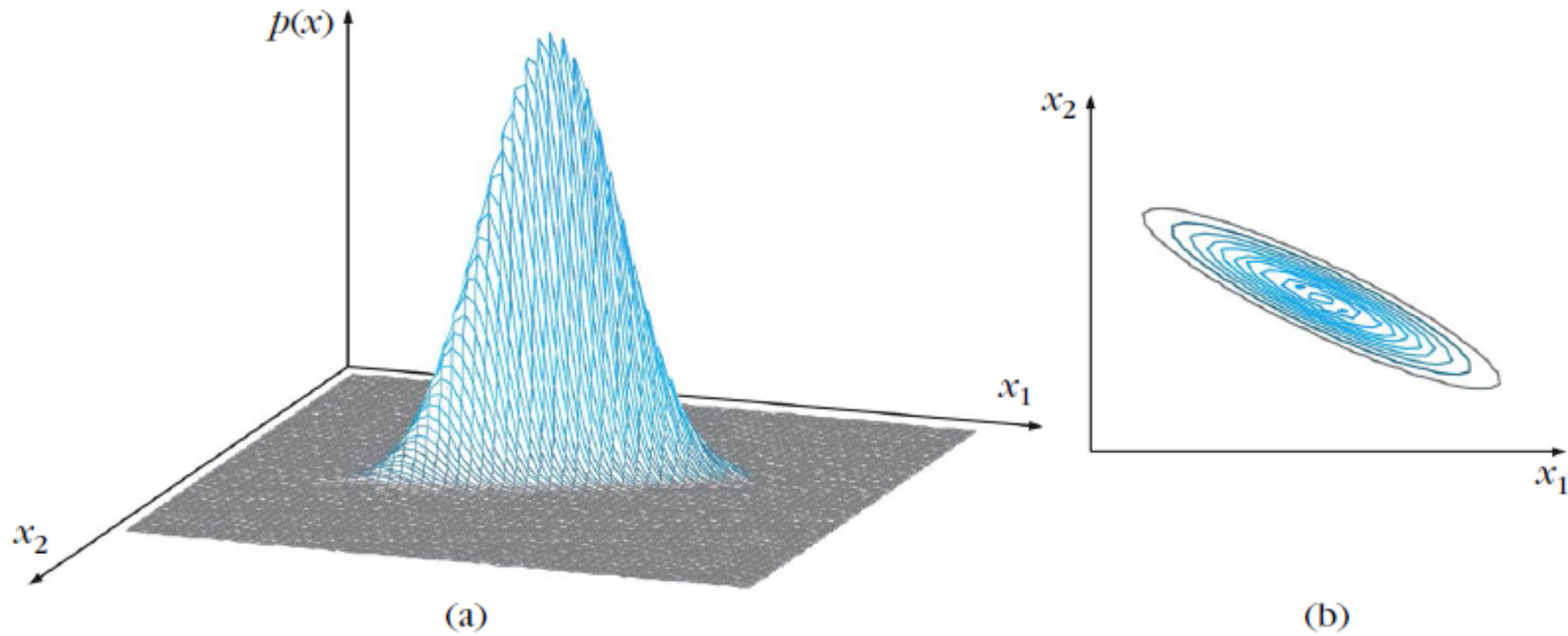
2D Gaussian pdf, diagonal Σ with $\sigma_1^2 = 15 \gg \sigma_2^2 = 3$



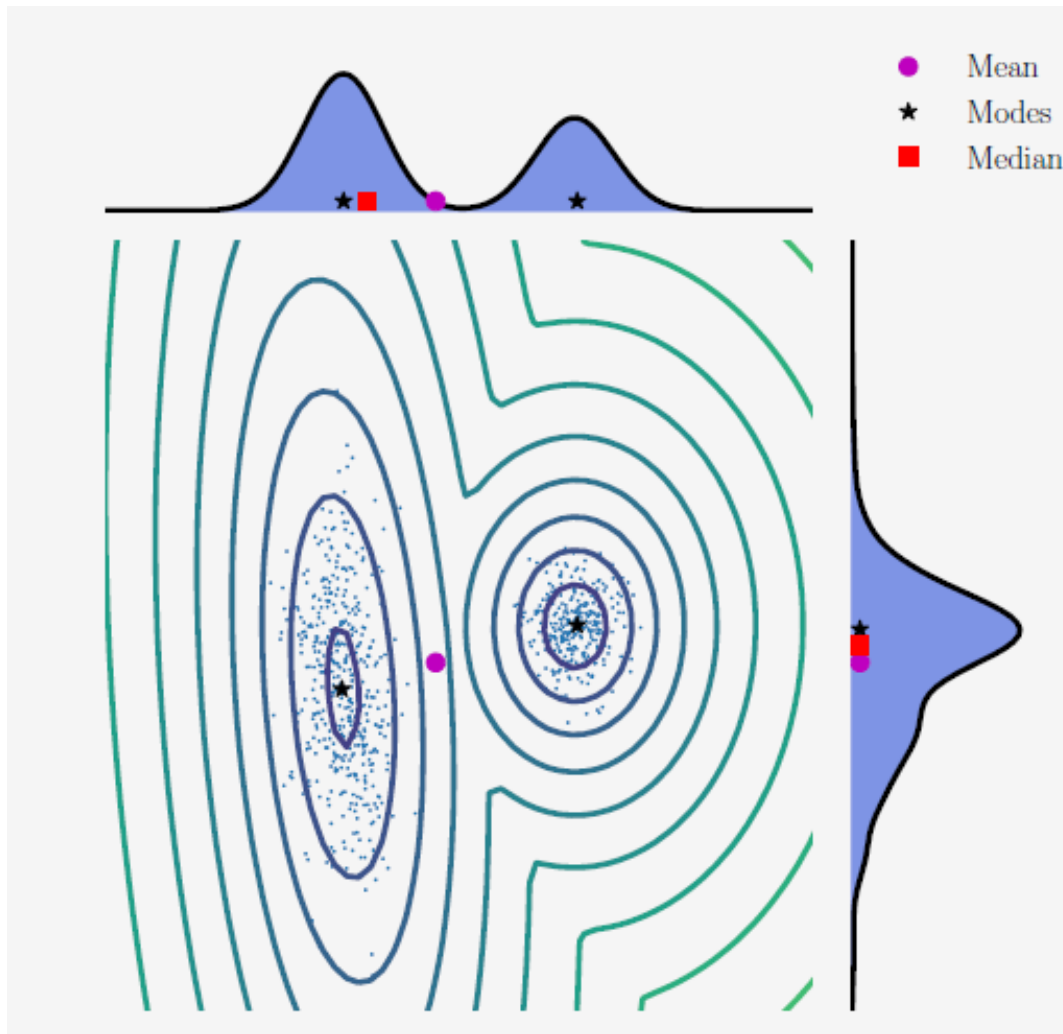
2D Gaussian pdf, diagonal Σ with $\sigma_1^2 = 3 \ll \sigma_2^2 = 15$



2D Gaussian pdf, non-diagonal Σ



Mean, mode, median for a 2D Dataset and its marginal densities



M. P. Deisenroth, A. A. Faisal, C. S. Ong, Mathematics For Machine Learning, Cambridge University Press, 2019

Maximum Likelihood

Maximum Likelihood

- Let $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$ be independent random samples drawn from pdf $p(\mathbf{x}; \boldsymbol{\theta})$. We form the joint pdf $p(X; \boldsymbol{\theta})$, where $X = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$

$$p(X; \boldsymbol{\theta}) \equiv p(\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N; \boldsymbol{\theta}) = \prod_{k=1}^N p(\mathbf{x}_k; \boldsymbol{\theta})$$

It is known as the **likelihood function** of $\boldsymbol{\theta}$ with respect to X

- Using the monotonicity of log, we define the *log-likelihood function*

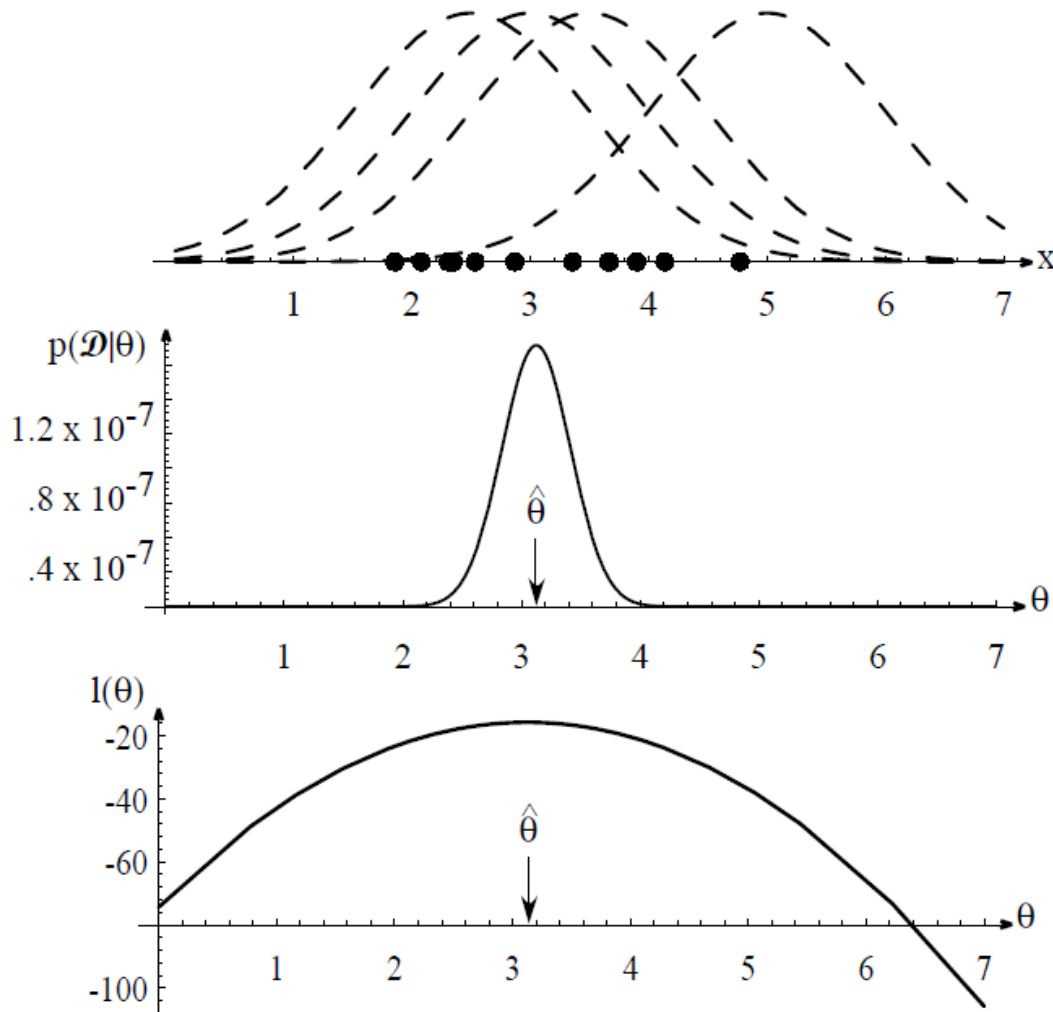
$$l(\boldsymbol{\theta}) \equiv \log \prod_{k=1}^N p(\mathbf{x}_k; \boldsymbol{\theta}) = \sum_{k=1}^N \log p(\mathbf{x}_k; \boldsymbol{\theta})$$

$$\hat{\theta}_{ML} = \arg \max_{\boldsymbol{\theta}} l(\boldsymbol{\theta}) \Rightarrow \frac{\partial l(\boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = \sum_{k=1}^N \frac{1}{p(\mathbf{x}_k; \boldsymbol{\theta})} \frac{\partial p(\mathbf{x}_k; \boldsymbol{\theta})}{\partial \boldsymbol{\theta}} = 0$$

- The ML estimate corresponds to the peak of the log-likelihood function.

Maximum Likelihood

Assume 1D training points are drawn from a Gaussian of a particular variance, but **unknown mean**. Four of the infinite number of candidate source distributions are shown in dashed lines.



The likelihood $p(D;\theta)$ as a function of the mean. If we had a very large number of training points, this likelihood would be very narrow.


The value that maximizes the likelihood is marked $\hat{\theta}$;

$\hat{\theta}$ also maximizes the logarithm of the likelihood — i.e., the log-likelihood $l(\theta)$, shown at the bottom.

Maximum Likelihood

Assume that N data points, x_1, x_2, \dots, x_N , have been generated by a 1D Gaussian pdf with unknown mean and variance. Derive the $\hat{\mu}_{ML}$ and $\hat{\sigma}_{ML}^2$.

$$\begin{aligned} L(\mu, \sigma^2) &\equiv \log \prod_{k=1}^N p(x_k; \mu, \sigma^2) = \sum_{k=1}^N \log p(x_k; \mu, \sigma^2) = \sum_{k=1}^N \log \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x_k - \mu)^2}{2\sigma^2}} \\ &= -\frac{N}{2} (\log(2\pi) + \log \sigma^2) - \frac{1}{2\sigma^2} \sum_{k=1}^N (x_k - \mu)^2 \end{aligned}$$

$$\frac{\partial L(\mu, \sigma^2)}{\partial \mu} = -\frac{1}{\sigma^2} \sum_{k=1}^N (x_k - \mu) = 0 \Rightarrow \hat{\mu}_{ML} = \frac{1}{N} \sum_{k=1}^N x_k$$


$$\frac{\partial L(\mu, \sigma^2)}{\partial \sigma^2} = -\frac{N}{2\sigma^2} + \frac{1}{2\sigma^4} \sum_{k=1}^N (x_k - \mu)^2 = 0 \Rightarrow \hat{\sigma}_{ML}^2 = \frac{1}{N} \sum_{k=1}^N (x_k - \mu)^2$$

Maximum Likelihood

Assume that N data points, $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, are vectors generated from a Gaussian pdf with unknown mean and covariance matrix. Derive the ML estimate of the variance.

$$\begin{aligned} l(\boldsymbol{\mu}, \Sigma) &\equiv \log \prod_{k=1}^N p(\mathbf{x}_k; \boldsymbol{\mu}) = \sum_{k=1}^N \log p(\mathbf{x}_k; \boldsymbol{\mu}) \\ &= \sum_{k=1}^N \log \frac{1}{(2\pi)^{d/2} |\Sigma|^{1/2}} \exp\left[-\frac{1}{2} (\mathbf{x}_k - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_k - \boldsymbol{\mu})\right] \\ &= -\frac{N}{2} (\log(2\pi^d |\Sigma|)) - \frac{1}{2} \sum_{k=1}^N (\mathbf{x}_k - \boldsymbol{\mu})^T \Sigma^{-1} (\mathbf{x}_k - \boldsymbol{\mu}) \end{aligned}$$

$$\frac{\partial L(\boldsymbol{\mu}, \Sigma)}{\partial \boldsymbol{\mu}} = 0 \Rightarrow \hat{\boldsymbol{\mu}}_{ML} = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k$$

$$\frac{\partial L(\boldsymbol{\mu}, \Sigma)}{\partial \Sigma} = 0 \Rightarrow \hat{\Sigma}_{ML} = \frac{1}{N} \sum_{k=1}^N (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{ML})(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_{ML})^T$$

Maximum Likelihood: An example with three classes (plotcc.m)

Now we have

$$N(\mu_1, \Sigma_1), N(\mu_2, \Sigma_2), N(\mu_3, \Sigma_3)$$

We can compute

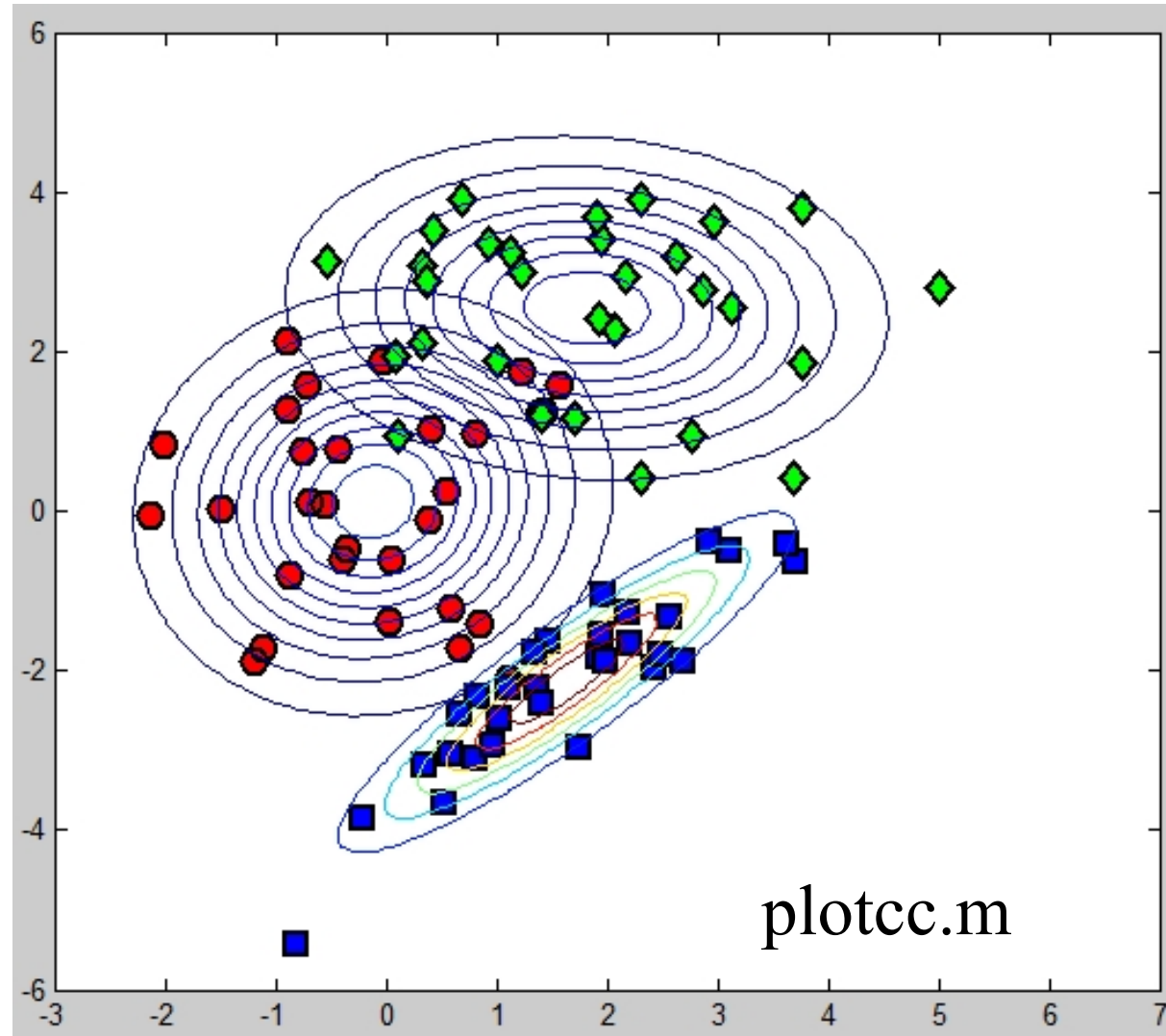
$$p(\mathbf{x}_{new} | t_{new} = \mathbf{1}, \mathbf{X}_1, \mathbf{t}_1)$$

$$p(\mathbf{x}_{new} | t_{new} = \mathbf{2}, \mathbf{X}_2, \mathbf{t}_2)$$

$$p(\mathbf{x}_{new} | t_{new} = \mathbf{3}, \mathbf{X}_3, \mathbf{t}_3)$$

And

$$\sum_{c=1}^3 p(\mathbf{x}_{new} | t_{new} = c, \mathbf{X}_c, \mathbf{t}_c) = \mathbf{1}$$



Fit class-conditional Gaussians for each class (plotcc.m)

```
class_var = [];  
for c = 1:length(cl)  
    pos = find(t==cl(c));  
    % Find the means  
    class_mean(c,:) = mean(X(pos,:));  
    class_var(:,:,c) = cov(X(pos,:),1);  
end  
%% Plot the contours  
for c = 1:length(cl)  
    pos = find(t==cl(c));  
    plot(X(pos,1),X(pos,2),col{c},...  
         'markersize',10,'linewidth',2,'markerfacecolor',fcol{c});  
end  
xlim([-3 7]),ylim([-6 6])  
[Xv,Yv] = meshgrid(-3:0.1:7,-6:0.1:6);  
for c = 1:length(cl)  
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)];  
    tempc = class_var(:,:,c);  
    const = -log(2*pi) - log(det(tempc));  
    Probs = exp(const - 0.5*diag(temp*inv(tempc)*temp'));  
    contour(Xv,Yv,reshape(Probs,size(Xv)));  
end
```

```
>> class_mean
```

```
class_mean =
```

-0.1141	0.1117
1.8161	2.5445
1.6356	-2.1388

```
>> class_var
```

```
class_var(:,:,1) =
```

0.9896	0.0886
0.0886	1.5076

```
class_var(:,:,2) =
```

1.7073	-0.1028
-0.1028	1.0659

```
class_var(:,:,3) =
```

1.1158	1.0470
1.0470	1.1917

Bayes' Rule

Joint, and Conditional Probability

- $p(A \cap B)$: the joint probability of the events A and B .
- A conditional probability measure $p(A|B)$ is defined by

$$p(A|B) = \frac{p(A \cap B)}{p(B)}$$

- Similarly,

$$p(B|A) = \frac{p(B \cap A)}{p(A)}$$

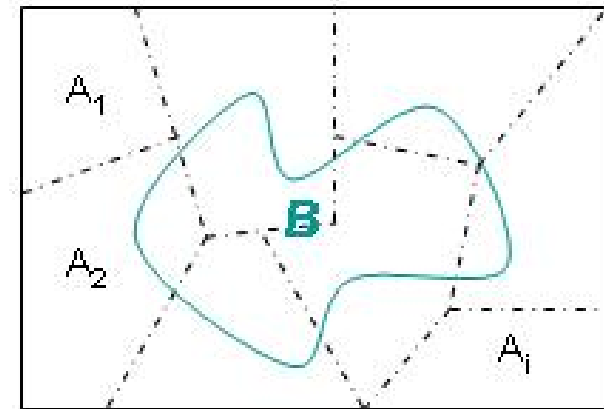
- Therefore

$$p(A|B) = \frac{p(B|A)p(A)}{p(B)}$$

Total Probability

- Let A_1, A_2, \dots, A_n be n mutually exclusive events, i.e., $A_i \cap A_j = \emptyset, i \neq j$, and $\bigcup_{i=1}^n A_i = \Omega$ (sample space)
- Let B be any event so that

$$B = B \cap \bigcup_{i=1}^n A_i = \bigcup_{i=1}^n (B \cap A_i)$$



$$\Rightarrow p(B) = \sum_{i=1}^n p(B \cap A_i) = \sum_{i=1}^n p(B|A_i)p(A_i)$$

Bayes' Theorem

- Now

$$p(A_i|B) = \frac{p(B|A_i)p(A_i)}{p(B)} = \frac{p(B|A_i)p(A_i)}{\sum_{i=1}^n p(B|A_i)p(A_i)}$$

- Therefore

$$p(t_{new} = c | \mathbf{x}_{new}, X, t) = \frac{p(\mathbf{x}_{new} | t_{new} = c, X_c, t_c) p(t_{new} = c | X_c, t_c)}{\sum_{c'=1}^3 p(\mathbf{x}_{new} | t_{new} = c', X_{c'}, t_{c'}) p(t_{new} = c' | X_{c'}, t_{c'})}$$
$$= \frac{\text{likelihood} \times \text{prior}}{\text{Evidence}}$$

Example

Assume a certain class is given a midterm exam.

S : the event that a student studied, $P(S)=0.7$

$P(\text{a student pass the exam} \mid S) = P(A|S)=0.9$

$P(\text{a student pass the exam} \mid S^c) = P(A|S^c)=0.05$

Given that a student did not pass the exam, what is the probability that she or he studied?

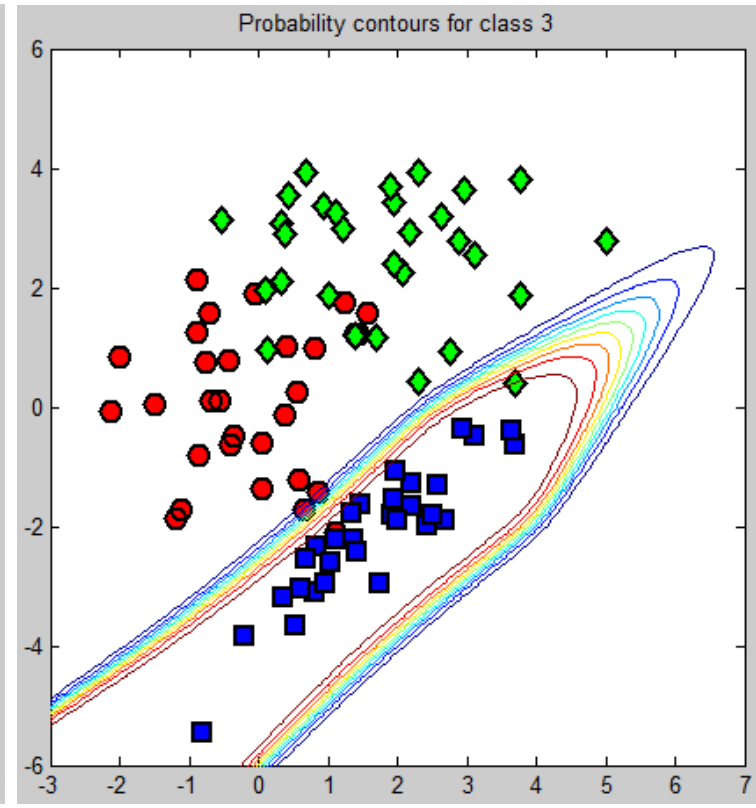
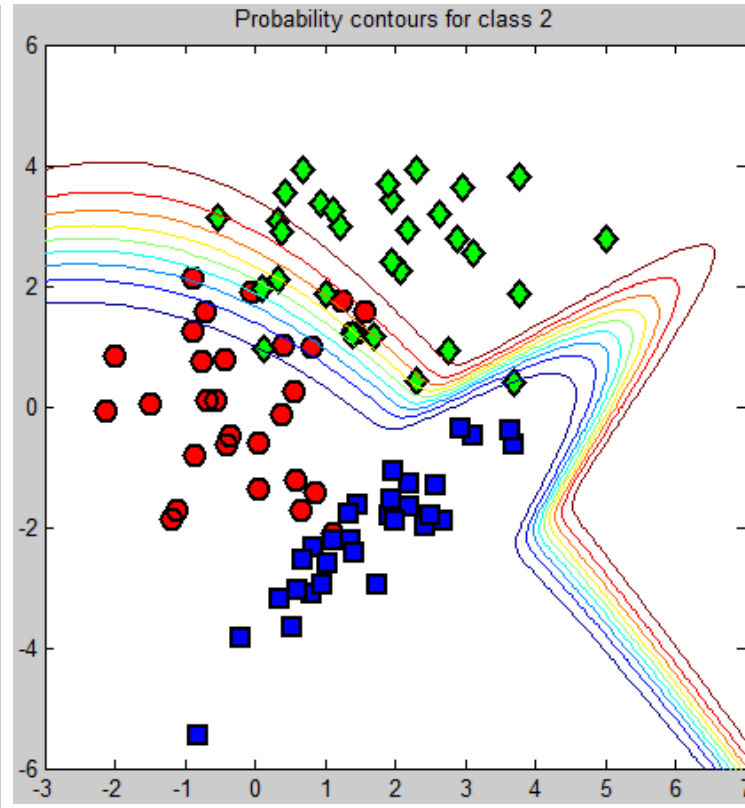
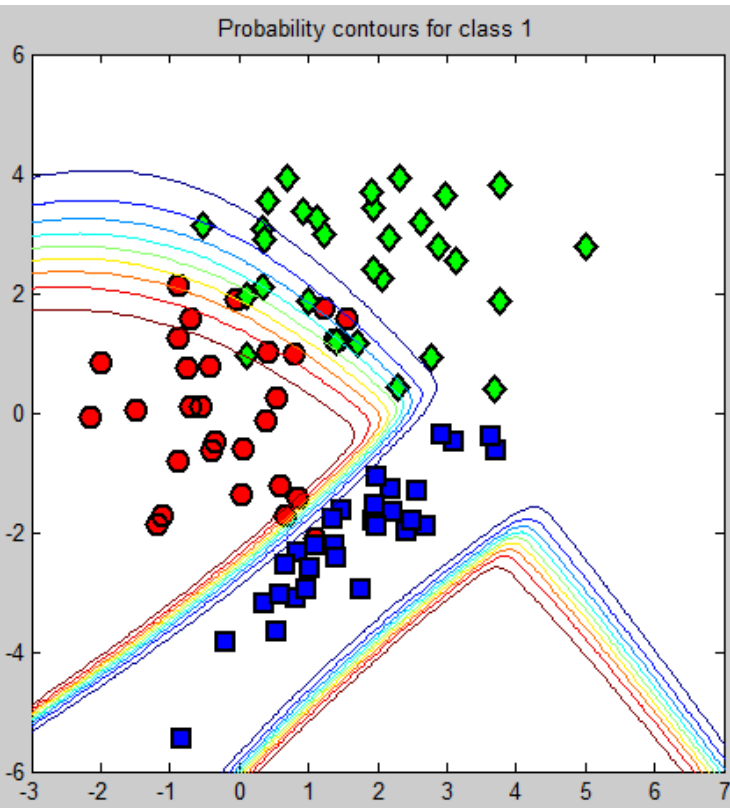
$$\begin{aligned} P(S|A^c) &= \frac{P(A^c \cap S) = P(A^c|S)P(S)}{P(A^c) = P(A^c \cap S) + P(A^c \cap S^c)} \\ &= \frac{P(A^c|S)P(S)}{P(A^c|S)P(S) + P(A^c|S^c)P(S^c)} \\ &= \frac{(1-0.9)*0.7}{(1-0.9)*0.7 + (1-0.05)*(1-0.7)} = \frac{0.07}{0.335} = 19.7\% \end{aligned}$$

Making prediction

- Since we can use the known $N(\mu_1, \Sigma_1)$, $N(\mu_2, \Sigma_2)$, and $N(\mu_3, \Sigma_3)$ to compute $p(\mathbf{x}_{new} | t_{new} = 1, X_1, t_1)$, $p(\mathbf{x}_{new} | t_{new} = 2, X_2, t_2)$ and $p(\mathbf{x}_{new} | t_{new} = 3, X_3, t_3)$
- If we assume the prior $p(t_{new} = c | X_c, t_c) = \frac{N_c}{N_1 + N_2 + N_3}$
- Therefore

$$\begin{aligned} p(t_{new} = c | \mathbf{x}_{new}, X, t) &= \frac{p(\mathbf{x}_{new} | t_{new} = c, X_c, t_c) p(t_{new} = c | X_c, t_c)}{\sum_{c'=1}^3 p(\mathbf{x}_{new} | t_{new} = c', X_{c'}, t_{c'}) p(t_{new} = c' | X_{c'}, t_{c'})} \\ &= \frac{N(\mathbf{x}_{new}; \mu_c, \Sigma_c) \frac{N_c}{N_1 + N_2 + N_3}}{\sum_{c'=1}^3 N(\mathbf{x}_{new}; \mu_{c'}, \Sigma_{c'}) \frac{N_{c'}}{N_1 + N_2 + N_3}} \end{aligned}$$

Making prediction (bayesclass.m)



```
%% bayesclass.m: Compute the predictive probabilities
```

```
[Xv,Yv] = meshgrid(-3:0.1:7,-6:0.1:6);
```

```
Probs = [];
```

```
for c = 1:length(cl)
```

```
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)];
```

```
    tempc = class_var(:, :, c);
```

```
    const = -log(2*pi) - log(det(tempc));
```

```
    Probs(:, :, c) = reshape(exp(const - 0.5*diag(temp*inv(tempc)*temp')), size(Xv));
```

```
end
```

```
Probs = Probs./repmat(sum(Probs,3),[1,1,3]);
```

```
%% Plot the predictive contours
```

```
figure(2);
```

```
for i = 1:3
```

```
    subplot(1,3,i);
```

```
    hold off
```

```
    for c = 1:length(cl)
```

```
        pos = find(t==cl(c));
```

```
        plot(X(pos,1),X(pos,2),col{c}, ...
```

```
            'markersize',10,'linewidth',2,'markerfacecolor',fcol{c});
```

```
        hold on
```

```
    end
```

```
    xlim([-3 7]),ylim([-6 6])
```

```
    contour(Xv,Yv,Probs(:, :, i));
```

```
    ti = sprintf('Probability contours for class %g',i);title(ti);
```

```
end
```

$$p(t_{new} = c | X_c, t_c) = 1/3$$

$$p(t_{new} = c | x_{new}, X, t) = \frac{N(x_{new}; \mu_c, \Sigma_c) \frac{N_c}{N_1 + N_2 + N_3}}{\sum_{c'=1}^3 N(x_{new}; \mu_{c'}, \Sigma_{c'}) \frac{N_{c'}}{N_1 + N_2 + N_3}}$$

$$\mathbf{x}_{new} = \begin{bmatrix} 2 \\ 0 \end{bmatrix}, t_{new} = p(t_{new} = \mathbf{c} | \mathbf{x}_{new}, \mathbf{X}_c, \mathbf{t}_c) = ?$$

c	$p(\mathbf{x}_{new} t_{new} = c, \mathbf{X}_c, \mathbf{t}_c)$	$p(t_{new} = c \mathbf{X}_c, \mathbf{t}_c)$	$\frac{p(\mathbf{x}_{new} t_{new} = c, \mathbf{X}_c, \mathbf{t}_c)}{\sum_{c'=1}^3 p(\mathbf{x}_{new} t_{new} = c', \mathbf{X}_{c'}, \mathbf{t}_{c'})}$
1	0.0109	0.333	0.7072
2	0.0042	0.333	0.2741
3	0.0003	0.333	0.0187

bayesclass_x_new.m

```

%% bayesclass_x_new.m: Repeat without Naive assumption
class_var = [];
for c = 1:length(cl)
    pos = find(t==cl(c));
    % Find the means
    class_mean(c,:) = mean(X(pos,:));
    class_var(:, :, c) = cov(X(pos,:), 1);
end
%% Compute the predictive probabilities
Xv=2;
Yv=0 ;
Probs = [];
for c = 1:length(cl)
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)];
    tempc = class_var(:, :, c);
    const = -log(2*pi) - log(det(tempc));
    Probs(:, :, c) = reshape(exp(const -
0.5*diag(temp*inv(tempc)*temp')), size(Xv));
end
[Probs(:, :, 1) Probs(:, :, 2) Probs(:, :, 3) sum(Probs, 3)] % 分子 與 分母

Probs = Probs ./ repmat(sum(Probs, 3), [1, 1, 3])

```

The naive-Bayes assumption

The naive-Bayes assumption

- Fitting a 2-D Gaussian requires 5 parameters: 2 for μ_c , and 3 for Σ_c .
→ feasible for 30 training points in each class.
- But fitting a D-dimensional Gaussian requires $D + D + D(D - 1)/2$ parameters.
→ For 10 dimensions, 30 data points are not sufficient to fit 65 parameters.
- Naive Bayes assumption: the class-conditional distributions can be factorized into a product of univariate distributions.
→ $p(\mathbf{x}_i | t_i = c, \mathbf{X}_c, \mathbf{t}_c) = \prod_{d=1}^D p(x_{id} | t_i = c, \mathbf{X}_c, \mathbf{t}_c)$
→ cannot model any within-class dependencies
→ 10-dimensional Gaussian requires 20 parameters instead of 65

Maximum Likelihood: An example with three classes

Now we have

$$N(\mu_1, \sigma_1^2 I), N(\mu_2, \sigma_2^2 I), N(\mu_3, \sigma_3^2 I)$$

We can compute

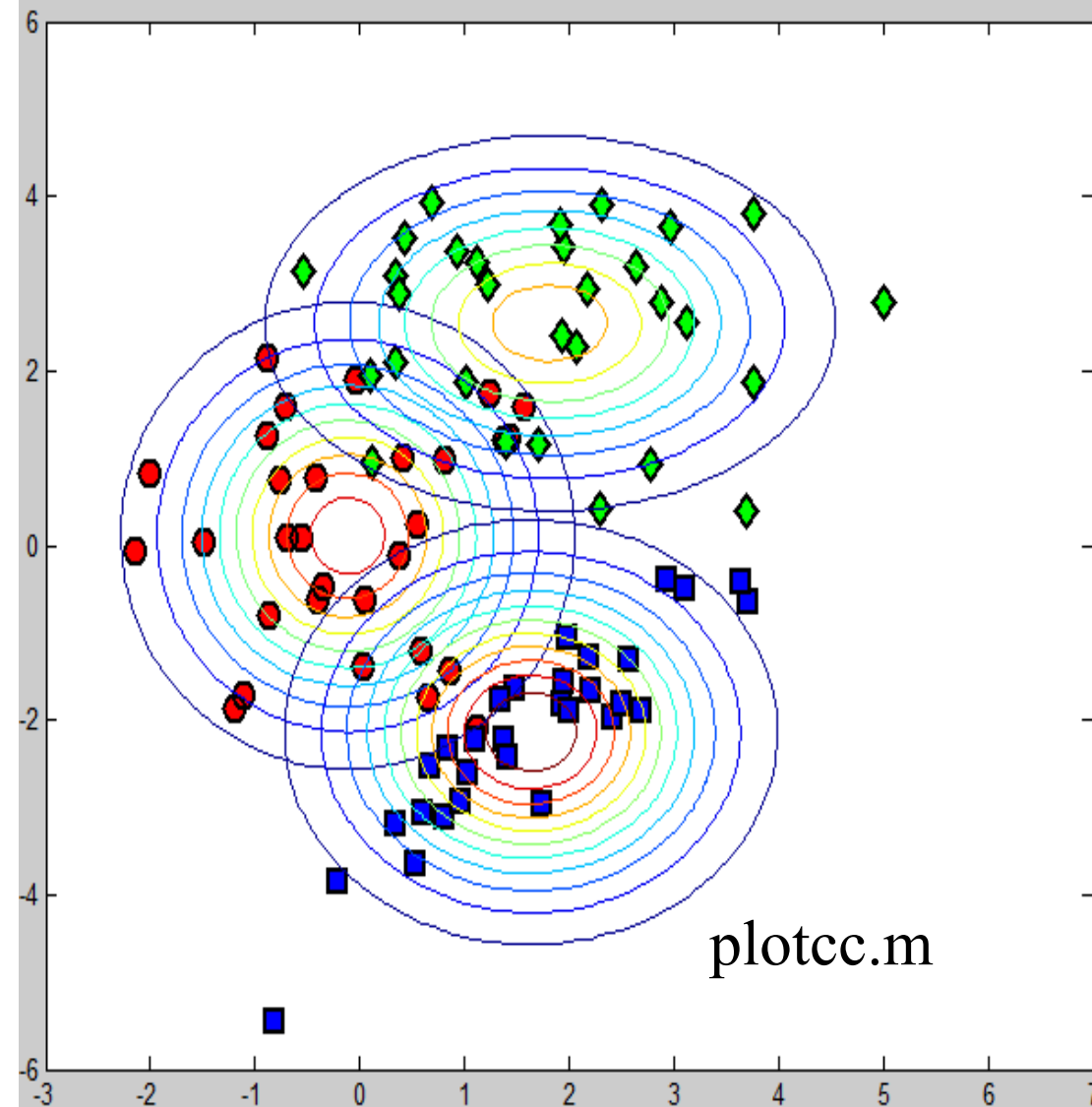
$$p(\mathbf{x}_{new} | t_{new} = 1, X_1, t_1)$$

$$p(\mathbf{x}_{new} | t_{new} = 2, X_2, t_2)$$

$$p(\mathbf{x}_{new} | t_{new} = 3, X_3, t_3)$$

And

$$\sum_{c=1}^3 p(\mathbf{x}_{new} | t_{new} = c, X_c, t_c) = 1$$




```

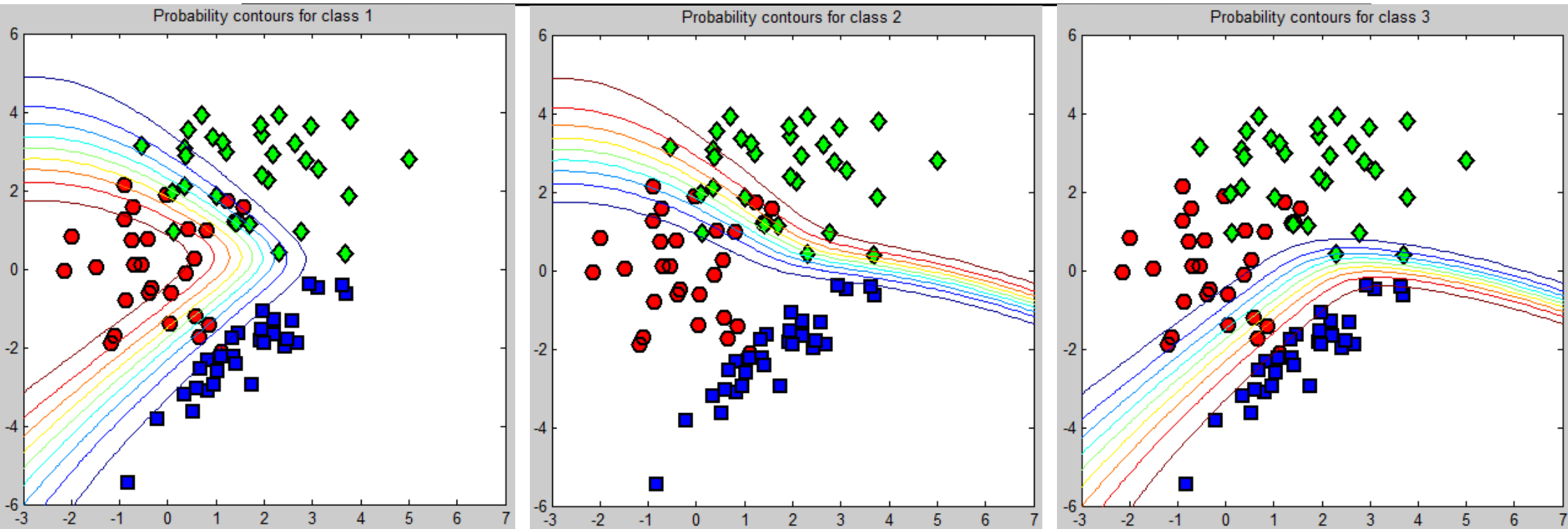
%% Fit class-conditional Gaussians for each class
% Using the Naive (independence) assumption
for c = 1:length(cl)
    pos = find(t==cl(c));
    % Find the means
    class_mean(c,:) = mean(X(pos,:));
    class_var(c,:) = var(X(pos,:),1);
end

%% Plot the contours
[Xv,Yv] = meshgrid(-3:0.1:7,-6:0.1:6);
for c = 1:length(cl)
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)];
    tempc = diag(class_var(c,:));
    const = -log(2*pi) - log(det(tempc));
    Probs = exp(const - 0.5*diag(temp*inv(tempc)*temp')));
    contour(Xv,Yv,reshape(Probs,size(Xv)));
end

```



Making prediction (bayesclass_naive.m)



Although the class-conditional distribution $p(\mathbf{x}_i | t_i = 3, \mathbf{X}_3, \mathbf{t}_3)$ for class 3 is not particularly appropriate, the classification contours are still reasonable

```

%% Compute the predictive probabilities
[Xv,Yv] = meshgrid(-3:0.1:7,-6:0.1:6);
Probs = [];
for c = 1:length(cl)
    temp = [Xv(:)-class_mean(c,1) Yv(:)-class_mean(c,2)];
    tempc = diag(class_var(c,:));
    const = -log(2*pi) - log(det(tempc));
    Probs(:, :, c) = reshape(exp(const - 0.5*diag(temp*inv(tempc)*temp')), size(Xv));
end
Probs = Probs./repmat(sum(Probs,3),[1,1,3]);
%% Plot the predictive contours
for i = 1:3
    subplot(1,3,i);
    for c = 1:length(cl)
        pos = find(t==cl(c));
        plot(X(pos,1),X(pos,2),col{c},...
            'markersize',10,'linewidth',2,'markerfacecolor',fcol{c});
        hold on
    end
    xlim([-3 7]), ylim([-6 6])
    contour(Xv,Yv,Probs(:, :, i));
    ti = sprintf('Probability contours for class %g',i);
    title(ti);
end

```

Summary

- We assume the class-conditional distribution $p(\mathbf{x}_i | t_i = c, \mathbf{X}_c, \mathbf{t}_c)$ is a multivariate Gaussian $N(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$
- $\hat{\boldsymbol{\mu}}_c = \frac{1}{N} \sum_{k=1}^N \mathbf{x}_k$ and $\hat{\boldsymbol{\Sigma}}_c = \frac{1}{N} \sum_{k=1}^N (\mathbf{x}_k - \hat{\boldsymbol{\mu}}_c)(\mathbf{x}_k - \hat{\boldsymbol{\mu}}_c)^T$ are the empirical mean and covariance computed from maximizing likelihood function.

- We make prediction using Bayes' rule

$$p(t_{\text{new}} = c | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{N(\mathbf{x}_{\text{new}}; \boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c) \frac{N_c}{N_1 + N_2 + N_3}}{\sum_{c'=1}^3 N(\mathbf{x}_{\text{new}}; \boldsymbol{\mu}_{c'}, \boldsymbol{\Sigma}_{c'}) \frac{N_{c'}}{N_1 + N_2 + N_3}}$$

- Use Naïve assumption to reduce the number of estimated parameters

Remark: Discriminant Analysis

Assume $p(\mathbf{x}_i | t_i = c, \mathbf{X}_c, \mathbf{t}_c)$ are Gaussian densities,

1. **the same** $\Sigma_c = \Sigma$ in each class, this leads to **linear discriminant analysis**.
2. **different** Σ_c in each class, we get **quadratic discriminant analysis**.
3. Σ_c are **diagonal**, i.e., conditional independence in each class, we get **naïve Bayes**.

Naïve Bayes for text classification

Example: Classifying Text into Spam/Not-spam

- Example: classify text into spam/not-spam (yes $\rightarrow \omega_1$; no $\rightarrow \omega_2$)
- Use bag-of-words features, get binary vector \mathbf{x} for each email
- Example:
It was the best of times,
it was the worst of times,
it was the age of wisdom,
it was the age of foolishness
- Now we can make a list of all of the words in our model vocabulary
“it”, “was”, “the”, “best”, “of”, “times”, “worst”, “age”, “wisdom”, “foolishness”
- The simplest scoring method is to mark the presence of words
(document) as a Boolean value, 0 for absent, 1 for present.
- | | |
|-------------------------------|--------------------------------|
| It was the best of times, | [1, 1, 1, 1, 1, 1, 0, 0, 0, 0] |
| it was the worst of times, | [1, 1, 1, 0, 1, 1, 1, 0, 0, 0] |
| it was the age of wisdom, | [1, 1, 1, 0, 1, 0, 0, 1, 1, 0] |
| it was the age of foolishness | [1, 1, 1, 0, 1, 0, 0, 1, 0, 1] |

Bayes Classifier

- Given binary features $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$ we want to compute class probabilities using Bayes Rule:

$$p(\omega_c|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_c)p(\omega_c)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_c)p(\omega_c)}{\sum_{c=1}^C p(\mathbf{x}|\omega_c)p(\omega_c)}$$

- To compute $p(\omega_c|\mathbf{x})$ we need: $p(\mathbf{x}|\omega_c)$ and $p(\omega_c)$
- Assume the training data are $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N$, each is D -dimensional

$$\mathbf{X} = \begin{bmatrix} \mathbf{x}_{1,1} & \mathbf{x}_{1,2} & \cdots & \mathbf{x}_{1,D} \\ \mathbf{x}_{2,1} & \mathbf{x}_{2,2} & \cdots & \mathbf{x}_{2,D} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{x}_{N,1} & \mathbf{x}_{N,2} & \cdots & \mathbf{x}_{N,D} \end{bmatrix}$$

and $\omega_c, c = 1, \dots, C$ are classes

Naïve Bayes

- Assume two classes: spam/non-spam and a dictionary of D words with binary features $\mathbf{x} = [x_1, x_2, \dots, x_D]^T$ saying whether each word appears in the e-mail.
- If we define a joint distribution $p(\omega_c, x_1, x_2, \dots, x_D)$, this gives enough information to determine $p(\omega_c)$ and $p(\mathbf{x}|\omega_c)$.
- Problem: specifying a joint distribution over $D + 1$ binary variables requires 2^{D+1} entries. This is computationally prohibitive and would require an absurd amount of data to fit.
- We'd like to impose **structure** on the distribution such that:
 - it can be **compactly** represented
 - **learning** and **inference** are both tractable

Naïve Bayes

- Naïve Bayes makes the assumption that the word features x_i are **conditionally independent** given the class ω_c .
 - x_i and x_j are independent under the conditional distribution $p(\mathbf{x}|\omega_c)$.
 - Mathematically,
$$\begin{aligned} p(\omega_c, x_1, x_2, \dots, x_D) &= p(\omega_c)p(x_1, x_2, \dots, x_D|\omega_c) \\ &= p(\omega_c)p(x_1|\omega_c) \cdots p(x_D|\omega_c) \end{aligned}$$
- Compact representation of the joint distribution
 - Prior probability of class: $p(\omega_c) = \theta_{\omega_c}$
 - Conditional probability of word feature given class: $p(x_j|\omega_c) = \theta_{j\omega_c}$
 - $2D + 1$ parameters in total

Naïve Bayes: **Learning**

- The parameters can be learned efficiently because the log-likelihood decomposes into independent terms for each feature.

$$L(\theta) = \sum_{i=1}^N \log p(\omega_{i,c}, \mathbf{x}_i) = \sum_{i=1}^N \log \{ p(\omega_{i,c}) \prod_{j=1}^D p(x_{i,j} | \omega_{i,c}) \}$$

$$= \sum_{i=1}^N \log p(\omega_{i,c}) + \sum_{j=1}^D \sum_{i=1}^N \log p(x_{i,j} | \omega_{i,c})$$

$$= \text{Bernoulli log-likelihood of labels} + \sum_{j=1}^D \text{Bernoulli log-likelihood for feature } x_j$$

- Each of these log-likelihood terms depends on different sets of parameters, so they can be optimized independently
- We want to maximize $\sum_{i=1}^N \log p(x_{i,j} | \omega_{i,c})$

Naïve Bayes: Making Predictions (Inference)

- We predict the category by performing inference in the model.
- Apply Bayes' Rule:

$$p(\omega_c|\mathbf{x}) = \frac{p(\mathbf{x}|\omega_c)p(\omega_c)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|\omega_c)p(\omega_c)}{\sum_{c=1}^C p(\omega_c)p(\mathbf{x}|\omega_c)} = \frac{p(\omega_c) \prod_{j=1}^D p(x_j|\omega_c)}{\sum_{c=1}^C p(\omega_c) \prod_{j=1}^D p(x_j|\omega_c)}$$

- We need not compute the denominator if we're simply trying to determine the mostly likely ω_c .
- Shorthand notation: $p(\omega_c|\mathbf{x}) \propto p(\omega_c) \prod_{j=1}^D p(x_j|\omega_c)$
- Given the value of \mathbf{x} , its class is decided by the likelihood ratio

$$l_{12} = \frac{p(\mathbf{x}|\omega_1)p(\omega_1)}{p(\mathbf{x}|\omega_2)p(\omega_2)} \begin{matrix} \omega_1 \\ > \\ < \\ \omega_2 \end{matrix} 1$$

Example: The spam/not-spam case

- Consider the binary bag-of-words vector $\mathbf{x} = [x_1, x_2, \dots, x_d]^T$, $x_j \in \{0,1\}$, $j = 1, 2, \dots, D$. Two classes: yes $\rightarrow \omega_1$; no $\rightarrow \omega_2$, $p(\omega_c)$ are assumed given
- $N_{a,b}$ is the counts for $x_j = a \in \{0,1\}$, and $\omega_c = b \in \{\omega_1, \omega_2\}$
- Let $p(x_j = 1|\omega_1) = p_j = \frac{N_{1,\omega_1}}{N_{0,\omega_1} + N_{1,\omega_1}}$ and $p(x_j = 1|\omega_2) = q_j = \frac{N_{1,\omega_2}}{N_{0,\omega_2} + N_{1,\omega_2}}$
- ω_1 : it was the best of times,
it was the worst of times,
 $\mathbf{x}_1 = [1, 1, 1, 1, 1, 1, 0, 0, 0, 0]$
 $\mathbf{x}_2 = [1, 1, 1, 0, 1, 1, 1, 0, 0, 0]$
- $p_1 = p_2 = p_3 = p_5 = p_6 = \frac{2}{12}, p_4 = p_7 = \frac{1}{12}, p_8 = p_9 = p_{10} = \frac{0}{12}$
- ω_2 : it was the age of wisdom,
it was the age of foolishness
 $\mathbf{x}_3 = [1, 1, 1, 0, 1, 0, 0, 1, 1, 0]$
 $\mathbf{x}_4 = [1, 1, 1, 0, 1, 0, 0, 1, 0, 1]$
- $q_1 = q_2 = q_3 = q_5 = q_8 = \frac{2}{12}, q_9 = q_{10} = \frac{1}{12}, q_4 = q_6 = q_7 = \frac{0}{12}$

Example: The discrete features case

- $p(x_j = 1|\omega_1) = p_j, p(x_j = 1|\omega_2) = q_j$
- However, adopting statistical independence among features, we can write

$$p(\mathbf{x}|\omega_1) = \prod_{j=1}^D p_j^{x_j} (1 - p_j)^{1-x_j},$$

e. g. $p(\mathbf{x}|\omega_1) = [1 \ 1 \ 1 \ 1 \ 1 \ 1 \ 0 \ 0 \ 0 \ 0]^T = p_1 p_2 p_3 p_4 p_5 p_6 (1 - p_7)(1 - p_8)(1 - p_9)(1 - p_{10})$

$$p(\mathbf{x}|\omega_2) = \prod_{j=1}^D q_j^{x_j} (1 - q_j)^{1-x_j}$$

- Hence, the number of required estimates is $2D$, that is, the p_j 's and q_j 's.

Example: learning p_j and q_j given $\mathbf{x}_1, \dots, \mathbf{x}_N$

- $p(x_j = 1|\omega_1) = p_j, p(x_j = 1|\omega_2) = q_j,$
- $p(x_{i,j}|\omega_1) = p_j^{x_{i,j}}(1 - p_j)^{1-x_{i,j}}, p(x_{i,j}|\omega_2) = q_j^{x_{i,j}}(1 - q_j)^{1-x_{i,j}}$
- Log-likelihood:

$$\begin{aligned}\log \prod_{i=1}^N p(x_{i,j}|\omega_1) &= \sum_{i=1}^N \log p(x_{i,j}|\omega_1) = \sum_{i=1}^{N_{1,\omega_1}} x_{i,j} \log p_j + \sum_{i=1}^{N_{0,\omega_1}} (1 - x_{i,j}) \log(1 - p_j) \\ &= N_{1,\omega_1} \log p_j + N_{0,\omega_1} \log(1 - p_j) \\ \log \prod_{i=1}^N p(x_{i,j}|\omega_2) &= \sum_{i=1}^N \log p(x_{i,j}|\omega_2) = \sum_{i=1}^{N_{1,\omega_2}} x_{i,j} \log q_j + \sum_{i=1}^{N_{0,\omega_2}} (1 - x_{i,j}) \log(1 - q_j) \\ &= N_{1,\omega_2} \log q_j + N_{0,\omega_2} \log(1 - q_j)\end{aligned}$$

- Obtain **maximum likelihood** estimates by setting derivatives to zero:

$$p_j = \frac{N_{1,\omega_1}}{N_{0,\omega_1} + N_{1,\omega_1}}, \quad q_j = \frac{N_{1,\omega_2}}{N_{0,\omega_2} + N_{1,\omega_2}}$$

Example: Making Predictions (Inference)

- Given the value of \mathbf{x} , its class is decided by the likelihood ratio

$$l_{12} = \frac{p(\mathbf{x}|\omega_1)p(\omega_1)}{p(\mathbf{x}|\omega_2)p(\omega_2)} \begin{matrix} \omega_1 \\ > \\ \omega_2 \\ < \end{matrix} 1$$

- The number of values that \mathbf{x} can take, for all possible combinations of x_j , amounts to 2^D .
- If we do not adopt independence assumption, one must have enough training data to obtain probability estimates for each one of these values (probabilities add to one, thus $2^D - 1$ estimates are required).

Example: Making Predictions (Inference)

- The *linear* discriminant function for decision boundary:

$$\frac{p(\mathbf{x}|\omega_1)p(\omega_1)}{p(\mathbf{x}|\omega_2)p(\omega_2)} = \frac{\prod_{j=1}^D p_j^{x_j} (1-p_j)^{1-x_j} p(\omega_1)}{\prod_{j=1}^D q_j^{x_j} (1-q_j)^{1-x_j} p(\omega_2)} = \begin{matrix} \omega_1 \\ > \\ < \\ \omega_2 \end{matrix} 1 = 1$$

$$\log \prod_{j=1}^D p_j^{x_j} (1-p_j)^{1-x_j} p(\omega_1) = \log \prod_{j=1}^D q_j^{x_j} (1-q_j)^{1-x_j} p(\omega_2)$$

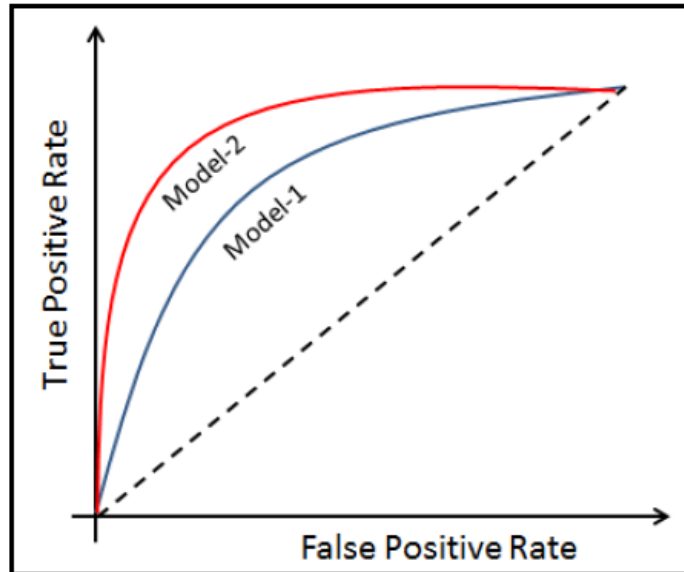
$$\sum_{j=1}^D (x_j \log p_j + (1-x_j) \log(1-p_j) + \log p(\omega_1)) = \sum_{i=1}^D (x_j \log q_j + (1-x_j) \log(1-q_j) + \log p(\omega_2))$$

$$0 = \sum_{j=1}^D \left(x_j \log \frac{p_j}{q_j} + (1-x_j) \log \frac{(1-p_j)}{(1-q_j)} + \log \frac{p(\omega_1)}{p(\omega_2)} \right) \equiv g(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + \mathbf{w}_0$$

$$\text{where } \mathbf{w}^T = \left[\log \frac{p_1(1-q_1)}{q_1(1-p_1)}, \dots, \log \frac{p_D(1-q_D)}{q_D(1-p_D)} \right], \mathbf{w}_0 = \sum_{j=1}^D \log \frac{(1-p_j)}{(1-q_j)} + \log \frac{p(\omega_1)}{p(\omega_2)}$$

ROC curve and AUC

- AUC-ROC curve is to measure and assess the performance of classification models.
- ROC (Receiver Operating Characteristics) is a pictorial visualization of model performance.
- It plots a 2D probability plot between the FP rate (or 1-specificity) and the TP rate (or sensitivity).
- We can also represent the area covered by a model with a single number using AUC:



AUC	0.5	No Discrimination
	0.6 – 0.7	Poor
	0.7 – 0.8	Acceptable(fair)
	0.8 – 0.9	Excellent(good)
	>0.9	Outstanding

ROC curve and AUC

舉例來說，用血壓值來檢測一個人是否有高血壓，測出的血壓值是連續的實數（從0~200都有可能），以收縮壓140／舒張壓90為閾值，閾值以上便診斷為有高血壓，閾值未滿者診斷為無高血壓。二元分類模型的個案預測有四種結局：

- 1.真陽性（TP）：診斷為有，實際上也有高血壓。
- 2.偽陽性（FP）：診斷為有，實際卻沒有高血壓。
- 3.真陰性（TN）：診斷為沒有，實際上也沒有高血壓。
- 4.偽陰性（FN）：診斷為沒有，實際卻有高血壓。

這四種結局可以畫成2 × 2的混淆矩陣：

		真實值		總數
		<i>p</i>	<i>n</i>	
預測輸出	<i>p'</i>	真陽性 (TP)	偽陽性 (FP)	<i>P'</i>
	<i>n'</i>	偽陰性 (FN)	真陰性 (TN)	<i>N'</i>
總數		<i>P</i>	<i>N</i>	

ROC空間將偽陽性率（FPR）定義為 X 軸，真陽性率（TPR）定義為 Y 軸。

- TPR：在所有實際為陽性的樣本中，被正確地判斷為陽性之比率。

$$TPR = TP / (TP + FN)$$

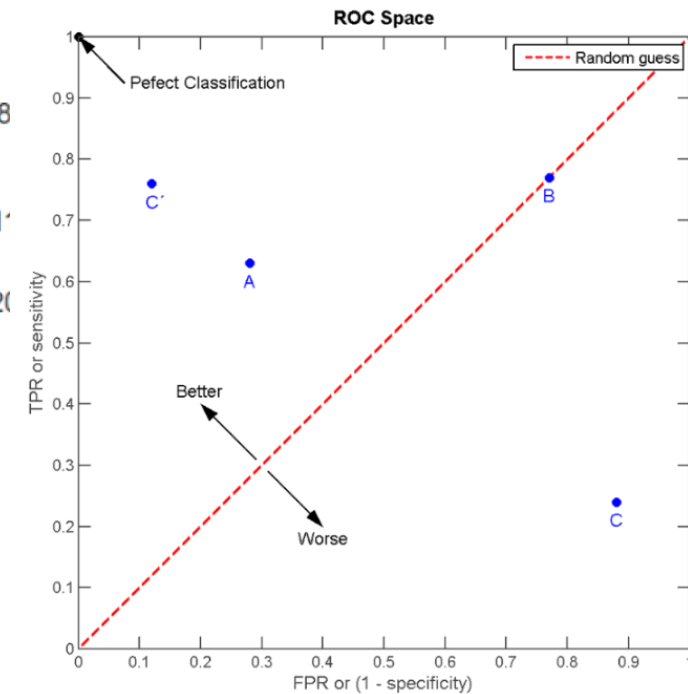
- FPR：在所有實際為陰性的樣本中，被錯誤地判斷為陽性之比率。

$$FPR = FP / (FP + TN)$$

ROC curve and AUC

給定一個二元分類模型和它的閾值，就能從所有樣本的（陽性／陰性）真實值和預測值計算出一個 $(X=FPR, Y=TPR)$ 座標點。在這條線的以上的點代表了一個好的分類結果（勝過隨機分類），而在這條線以下的點代表了差的分類結果（劣於隨機分類）。

A			B			C			C'		
TP=63	FP=28	91	TP=77	FP=77	154	TP=24	FP=88	112	TP=76	FP=12	8
FN=37	TN=72	109	FN=23	TN=23	46	FN=76	TN=12	88	FN=24	TN=88	112
100	100	200	100	100	200	100	100	200	100	100	200
TPR = 0.63			TPR = 0.77			TPR = 0.24			TPR = 0.76		
FPR = 0.28			FPR = 0.77			FPR = 0.88			FPR = 0.12		
ACC = 0.675			ACC = 0.500			ACC = 0.180			ACC = 0.820		

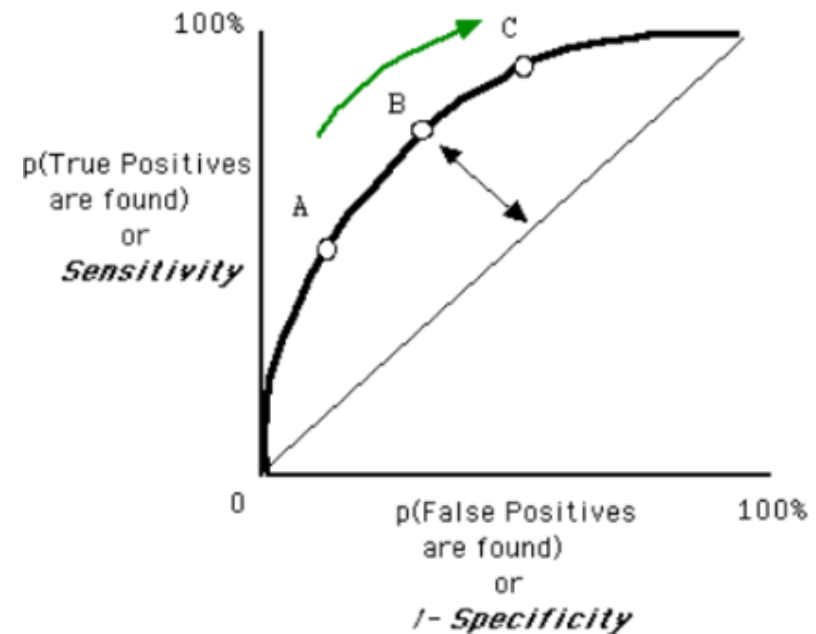
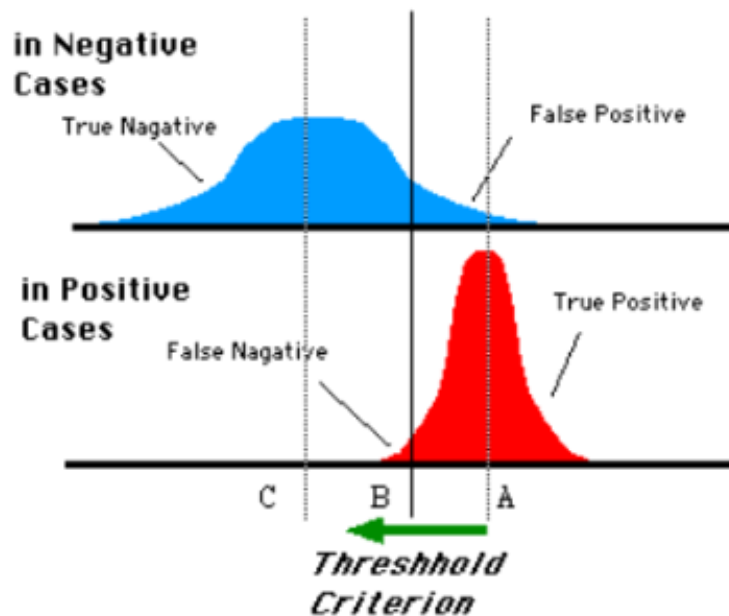


- 上述ROC空間裡的單點，是給定分類模型且給定閾值後得出的。
- 但同一個二元分類模型的閾值可能設定為高或低，每種閾值的設定會得出不同的FPR和TPR。

ROC curve and AUC

- 將同一模型每個閾值的 (FPR, TPR) 座標都畫在ROC空間裡，就成為特定模型的**ROC**曲線
- 例如下圖，人體的血液蛋白濃度是呈常態分布的連續變數，病人的分布是紅色，平均值為 A g/dL，健康人的分布是藍色，平均值是 C g/dL。健康檢查會測量血液樣本中的某種蛋白質濃度，達到某個值（閾值，threshold）以上診斷為有疾病徵兆。
- 研究者可以調整閾值的高低（將下圖的B垂直線往左或右移動），便會得出不同的偽陽性率與真陽性率，總之即得出不同的預測準確率。

Distributions of the Observed signal strength



<https://zh.wikipedia.org/wiki/ROC%E6%9B%B2%E7%BA%BF>

General Case: Multinomial Distribution

Multinomial coefficient:

The number of ways in which N distinct objects can be divided into k classes of sizes n_1, n_2, \dots, n_k , where $n_1 + n_2 + \dots + n_k = N$

Using the multiplication principle, the total number of division is

$$\binom{N}{n_1} \times \binom{N - n_1}{n_2} \times \binom{N - n_1 - n_2}{n_3} \times \dots \times \binom{N - n_1 - n_2 - \dots - n_{k-1}}{n_k} = \frac{N!}{n_1! \times \dots \times n_k!}$$

Example : In a class of 25 students, the instructor wants

project1 : a group of 10

project2 : a group of 8

project3 : a group of 7

How many possible ways can the instructor assign the students to the three projects?

sol :

$$\binom{25}{10} \times \binom{15}{8} \times \binom{7}{7} = \frac{25!}{10! \times 8! \times 7!} = 956112300$$

General Case: Multinomial Distribution

- Imagine you were building a machine that produces random documents of N words and you wanted to define a distribution over these documents
- One way of representing a document would be with a vector of word counts.
- Assuming D possible words in the vocabulary, the vector would be of length D and the j th element would hold the number of times the j th word appears in the document.
- Example: : it was the best of times and worst of times . ($N=10$)
it was the age of wisdom and age of foolishness. ($N=10$)
- Now we can make a list of all of the words in our model vocabulary
“it”, “was”, “the”, “best”, “of”, “times”, “and”, “worst”, “age”, “wisdom”, “foolishness” ($D=11$)

General Case: Multinomial Distribution

- Example: : it was the best of times and worst of times . ($N=10$)
it was the age of wisdom and age of foolishness. ($N=10$)
- Now we can make a list of all of the words in our model vocabulary
“it”, “was”, “the”, “best”, “of”, “times”, “and”, “worst”, “age”, “wisdom”, “foolishness” ($D=11$)

$$y = \begin{bmatrix} \text{it} \\ \text{was} \\ \text{the} \\ \text{best} \\ \text{of} \\ \text{times} \\ \text{and} \\ \text{worst} \\ \text{age} \\ \text{wisdom} \\ \text{foolishness} \end{bmatrix}, y_1 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 2 \\ 2 \\ 1 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, y_2 = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \\ 2 \\ 0 \\ 1 \\ 0 \\ 2 \\ 1 \\ 1 \end{bmatrix}$$

General Case: Multinomial Distribution

- The multinomial distribution allows us to define a distribution over such vectors.
- Let Y be a random variable that represents a document. An instance of Y is a vector of word counts $\mathbf{y} = \begin{bmatrix} y_1 \\ \vdots \\ y_D \end{bmatrix}$, the multinomial distribution of \mathbf{y} is

$$P(Y = \mathbf{y}) = p(\mathbf{y}) = \frac{(\sum_{j=1}^D y_j)!}{y_1! \times \cdots y_D!} \prod_{j=1}^D q_j^{y_j}$$

q_j 's represent the probabilities of the individual words, $\sum_{j=1}^D q_j = 1$

Classifying Text

- The most common way of encoding a document as a vector of numerical values is to use the bag-of-words model.
- If the total number of unique words in all documents is D , each document is represented as an D -dimensional vector.
- The vector for the i th document, \mathbf{x}_i , is made up of the counts of the number of times each word appears. x_{ij} is the number of times the j th word appears in document i .
- Given that the number of unique words is likely to be large, we will make the naive-Bayes assumption

$$p(\mathbf{x}_i | t_i = c, \dots) = \prod_{j=1}^D p(x_{ij} | t_i = c, \dots)$$

Classifying Text

- The bag-of-words model assumes the ordering of the words is not important. For example, \mathbf{x}_i is identical for the following two documents
 1. The quick brown fox jumps over the lazy dog.
 2. Dog quick lazy the jumps fox brown the over.

- We use multinomials for the class-conditional distributions.

$$p(\mathbf{x}_i|\mathbf{q}) = \left(\frac{s_i!}{\prod_{j=1}^D x_{ij}!} \right) \prod_{j=1}^D q_j^{x_{ij}}$$

where $s_i = \sum_{j=1}^D x_{ij}$ and $\mathbf{q} = \begin{bmatrix} q_1 \\ \vdots \\ q_D \end{bmatrix}$ are a set of parameters, each of which is a probability $\sum_{j=1}^D q_j = 1$.

- The multinomial distribution **automatically makes the Naive Bayes** assumption through the product over j .

Classifying Text

- There will be one multinomial, and hence one \mathbf{q}_c , for each class c .
- Given the set of training objects \mathbf{x}_i corresponding to class c , i.e., $\mathbf{x}_1, \dots, \mathbf{x}_{N_c}$, using maximum likelihood (homework 9.2) to obtain

$$q_{cj} = \frac{\sum_{i=1}^{N_c} x_{ij}}{\sum_{j'=1}^D \sum_{i=1}^{N_c} x_{ij'}}, c = 1, \dots, C$$

- Assume the prior distribution $P(T_{\text{new}} = c | \mathbf{X}, \mathbf{t}) = \frac{1}{C}$, we make predictions

$$p(T_{\text{new}} = c | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{p(\mathbf{x}_{\text{new}} | T_{\text{new}} = c, \mathbf{X}, \mathbf{t}) p(T_{\text{new}} = c | \mathbf{X}, \mathbf{t})}{\sum_{c'=1}^C p(\mathbf{x}_{\text{new}} | T_{\text{new}} = c', \mathbf{X}, \mathbf{t}) p(T_{\text{new}} = c' | \mathbf{X}, \mathbf{t})}$$

$$\text{where } p(\mathbf{x}_{\text{new}} | T_{\text{new}} = c, \mathbf{X}, \mathbf{t}) = \left(\frac{s_{\text{new}}!}{\prod_{j=1}^D x_{\text{new},j}!} \right) \prod_{j=1}^D q_{cj}^{x_{\text{new},j}}, s_{\text{new}} = \sum_{j=1}^D x_{\text{new},j}$$

Smoothing

- It is quite feasible that a particular word (say j th word) will never appear in documents from one class (say c) :
 - not many religious newsgroup posts are likely to mention 'baseball'.
$$\Rightarrow q_{cj} = 0, \Rightarrow \prod_{j=1}^D q_{cj}^{x_{ij}} = 0,$$
$$\Rightarrow p(\mathbf{x}_{new} | T_{new} = c, \mathbf{X}, \mathbf{t}) = 0, \Rightarrow p(T_{new} = c | \mathbf{x}_{new}, \mathbf{X}, \mathbf{t}) = 0$$

\Rightarrow A document including a word that doesn't appear in any of the training documents will have probability 0 of belonging to all classes.

- This can be overcome by placing a prior density on \mathbf{q} that encodes the belief that all probabilities > 0 , such as the Dirichlet density

$$p(\mathbf{q}_c | \boldsymbol{\alpha}) = \left(\frac{\Gamma(\sum_{j=1}^D \alpha_j)}{\prod_{j=1}^D \Gamma(\alpha_j)} \right) \prod_{j=1}^D q_{cj}^{\alpha_j - 1}$$

Smoothing

- Assume that $\alpha_j = \alpha$, i.e. the parameter used to define the Dirichlet is the same for each word
- We can estimate \mathbf{q}_c with the MAP estimate rather than the maximum likelihood, that is:

$$\max_{\mathbf{q}_c} p(\mathbf{x}_i | \mathbf{q}_c) p(\mathbf{q}_c | \alpha) = \max_{\mathbf{q}} \left(\frac{s_i!}{\prod_{j=1}^D x_{ij}!} \right) \prod_{j=1}^D q_{cj}^{x_{ij}} \left(\frac{\Gamma(\sum_{j=1}^D \alpha)}{\prod_{j=1}^D \Gamma(\alpha)} \right) \prod_{j=1}^D q_{cj}^{\alpha-1}$$

$$\Rightarrow q_{cj} = \frac{\alpha-1 + \sum_{i=1}^{N_c} x_{ij}}{D(\alpha-1) + \sum_{j'=1}^D \sum_{i=1}^{N_c} x_{ij'}} > 0 \text{ if } \alpha > 1, \text{ and } \rightarrow \frac{1}{D} \text{ if } \alpha \uparrow$$

- This technique is often referred to as **smoothing** and also be considered as another example of regularization

Newsgroup Example

- The 20 newsgroup data were split into training and test sets $\approx 11,000$ and $\approx 7,000$ documents, respectively.

- Setting $\alpha = 2 \Rightarrow q_{cj} = \frac{\alpha - 1 + \sum_{i=1}^{N_c} x_{ij}}{D(\alpha - 1) + \sum_{j'=1}^D \sum_{i=1}^{N_c} x_{ij'}}$, using

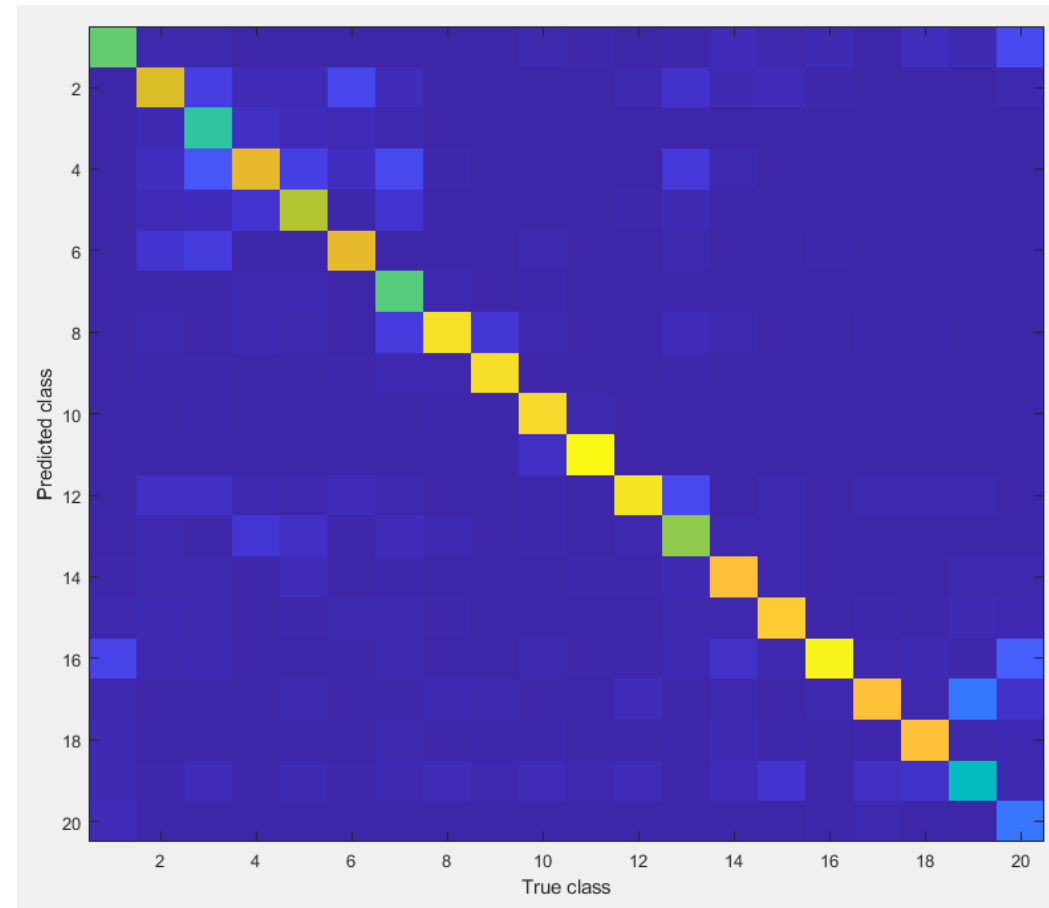
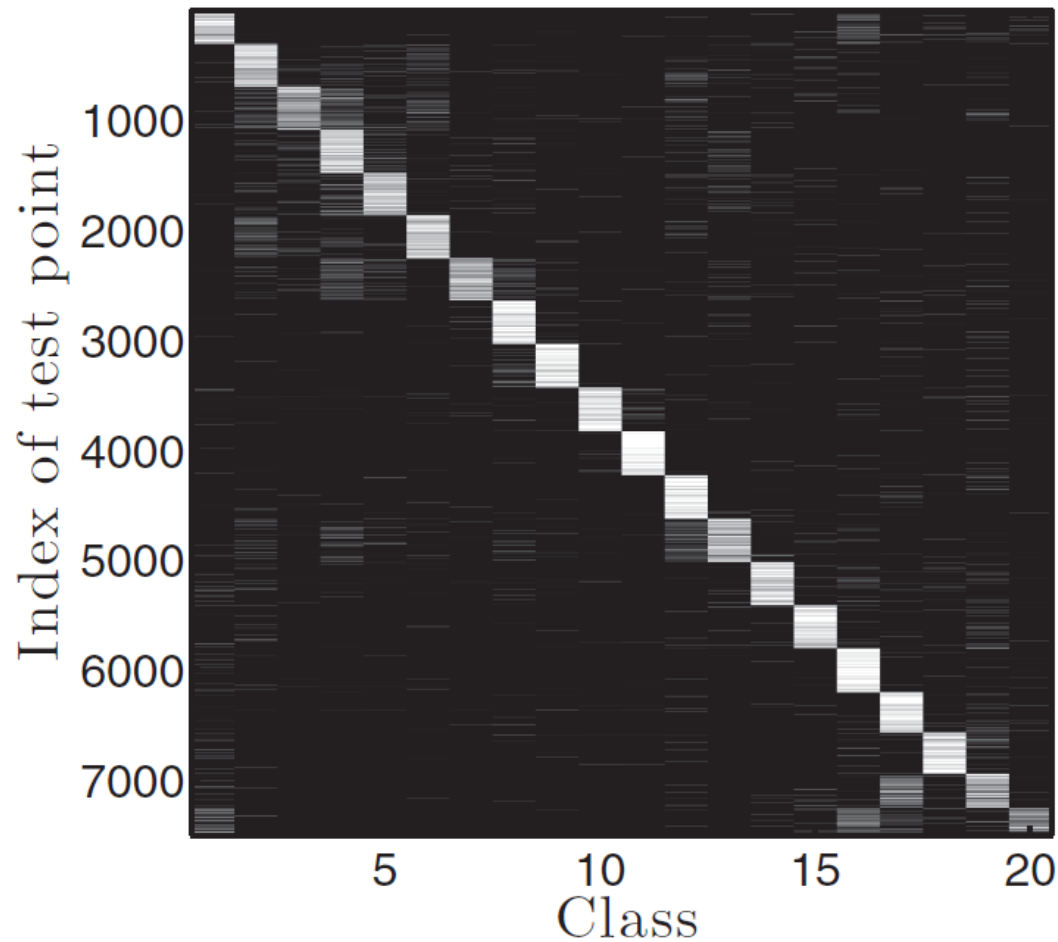
$$p(\mathbf{x}_{\text{new}} | T_{\text{new}} = c, \mathbf{X}, \mathbf{t}) = \left(\frac{s_{\text{new}}!}{\prod_{j=1}^D x_{\text{new},j}!} \right) \prod_{j=1}^D q_{cj}^{x_{\text{new},j}} \text{ and the prior } P(T_{\text{new}} = c | \mathbf{X}, \mathbf{t}) = \frac{1}{20},$$

we compute the classification probabilities

$$p(T_{\text{new}} = c | \mathbf{x}_{\text{new}}, \mathbf{X}, \mathbf{t}) = \frac{p(\mathbf{x}_{\text{new}} | T_{\text{new}} = c, \mathbf{X}, \mathbf{t}) p(T_{\text{new}} = c | \mathbf{X}, \mathbf{t})}{\sum_{c'=1}^C p(\mathbf{x}_{\text{new}} | T_{\text{new}} = c', \mathbf{X}, \mathbf{t}) p(T_{\text{new}} = c' | \mathbf{X}, \mathbf{t})}$$

- For each of the 7,000 \mathbf{x}_{new} vectors, we have a set of 20 probabilities.
- Assign each \mathbf{x}_{new} to the class for which it has the highest probability and the overall classification rate 78%.

Graphical representation of the predictive probabilities for the Bayesian classifier on the 20 newsgroups data. Each row corresponds to one test point and the test points are ordered by true class. The whiter the colour, the higher the probability



MATLAB Script: newspred.m

```
% newspred.m % From A First Course in Machine Learning, Chapter 5.
% Naive Bayesian classifier on the 20 newsgroup data
clear all; close all;
load ../data/newsgroups.mat %% Load the data
%% Compute the class conditional q parameters
alpha = 2; % Smoothing parameter
M = size(X,2); % Vocabulary size
q = zeros(20,M);
for c = 1:20
    pos = find(t==c);
    q(c,:) = (alpha - 1 + sum(X(pos,:),1))./(M*(alpha-1) + sum(sum(X(pos,:))));
end
%% Compute the test probabilities
% Do this with logs for numerical stability.
% Note: this takes quite a long time!
Nt = size(Xt,1);
testP = zeros(Nt,20);
for c = 1:20
    fprintf('\nClass %g',c);
    testP(:,c) = sum(Xt.*log(repmat(q(c,:),Nt,1)),2);
end
```

```

%% Normalise
C = 20;
prior = repmat(1/C,1,C); % Prior class probabilities
testP = testP + repmat(log(prior),Nt,1);
testP = exp(testP - repmat(max(testP,[],2),1,20));
testP = testP./repmat(sum(testP,2),1,20);
%% Visualise the probabilities
imagesc(testP);
%% Make the confusion matrix % Assign to max probability
assignments = (testP == repmat(max(testP,[],2),1,C));
[r,c] = find(assignments);
[r I] = sort(r);
c = c(I);
confusion = zeros(C);
for predicted = 1:C
    for true = 1:C
        confusion(predicted,true) = sum(testt==true & c==predicted);
    end
end
imagesc(confusion)
xlabel('True class');
ylabel('Predicted class');

```

Summary

- Naïve Bayes is an amazingly cheap learning algorithm!
- Training time: estimate parameters using **maximum likelihood**
 - Compute co-occurrence counts of each feature with the labels.
 - Requires only **one pass** through the data!
- Test time: apply Bayes' Rule
 - Cheap because of the model structure. (For more general models, Bayesian inference can be very expensive and/or complicated.)
- We covered the Bernoulli and multinomial cases.
- Unfortunately, it's usually less accurate compared to discriminative models.
 - The problem is the “naïve” **independence assumption**.

Homework #9-1

Students decide to make a text classifier. To begin with they attempt to classify documents as either sport or politics. They decide to represent each document as a (row) vector of attributes describing the presence or absence of words.

$x = (\text{goal football golf defense offence wicket office strategy})$

Training data from sport documents and from politics documents is represented below using a matrix in which each row represents the 8 attributes.

```
xP=[1 0 1 1 1 0 1 1; % Politics
    0 0 0 1 0 0 1 1;
    1 0 0 1 1 0 1 0;
    0 1 0 0 1 1 0 1;
    0 0 0 1 1 0 1 1;
    0 0 0 1 1 0 0 1]
```

```
xS=[1 1 0 0 0 0 0 0; % Sport
    0 0 1 0 0 0 0 0;
    1 1 0 1 0 0 0 0;
    1 1 0 1 0 0 0 1;
    1 1 0 1 1 0 0 0;
    0 0 0 1 0 1 0 0;
    1 1 1 1 1 0 1 0]
```

Using a maximum likelihood naive Bayes classifier, what are the respective probabilities that the document $x = (1\ 0\ 0\ 1\ 1\ 1\ 1\ 0)$ is about politics and sport ?

Homework #9-2

Compute the maximum likelihood estimates of $q_{cj} = \frac{\sum_{i=1}^{N_c} x_{ij}}{\sum_{j'=1}^D \sum_{i=1}^{N_c} x_{ij'}}$, $c = 1, \dots, C$ for class c of a Bayesian classifier with multinomial class-conditionals and a set of N_c , M -dimensional objects belonging to class c , $\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_{N_c}$.

Hint: The multinomials for the class-conditional distributions is

$$p(\mathbf{x}_i | \mathbf{q}_c) = \left(\frac{s_i!}{\prod_{j=1}^D x_{ij}!} \right) \prod_{j=1}^D q_{cj}^{x_{ij}}$$

where $s_i = \sum_{j=1}^D x_{ij}$ and $\mathbf{q}_c = \begin{bmatrix} q_{c1} \\ \vdots \\ q_{cD} \end{bmatrix}$ are a set of parameters, each of which is a probability $\sum_{j=1}^D q_{cj} = 1$.

Homework #9-2

Maximize the log-likelihood $\log \prod_{i=1}^{N_c} p(\mathbf{x}_i | \mathbf{q}_c)$ with the constrain $\sum_{j=1}^D q_{cj} = 1$.

Using the Lagrangian multiplier method to solve for q_{cj}

Homework #9-3

Study Homework 9-3.jpynb by yourself.

Deadline of Homework #9-1, 9-2: 2022/11/28 3:30pm