EE 610, ML Fundamentals

Logistic Regression

Dr. W. David Pan
Dept. of ECE
UAH

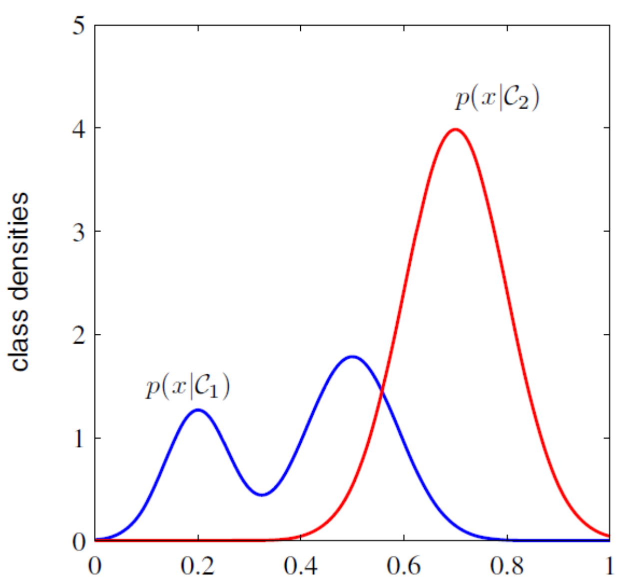
Topics

- Generative and discriminative models for interference and decision
- Logistic sigmoid function and its inverse (logit function)
- Parametric form of the posterior class probabilities
- Maximum likelihood solution for multivariate Gaussian distributions
- Linear algebra and matrix calculus
- Generalized linear models and link functions
- Logistic regression using maximum likelihood solution
- Newton-Raphson iterative optimization
- Implementations

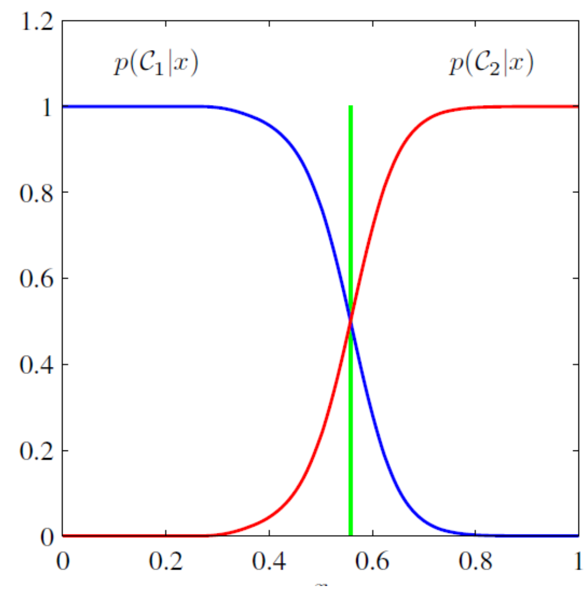
- Logistic regression, despite its name, is a linear model for classification rather than regression.
- Meaning of “Regression”:
  - A return to a former or less developed state.
  - In statistics, regression is the technique that allows one "to go back" from messy, hard to interpret data, to a clearer and more meaningful model.
- The most significant difference between **regression** versus **classification** is that regression helps predict a continuous quantity, while classification predicts discrete class labels.
- Logistic regression is also known in the literature as logit regression, maximum-entropy classification (MaxEnt), or the log-linear classifier.
- In this model, the probabilities describing the possible outcomes of a single trial are modeled using a logistic function.

Inference and Decision

- We can break the classification problem down into two separate stages:
  - **Inference** stage, where we use training data to learn a model for $p(C_k|\mathbf{x})$, and
  - The subsequent **decision** stage, where we use these posterior probabilities to make optimal class assignments.
- Approaches that model the distribution of inputs as well as outputs are known as **generative models**, because by sampling from them it is possible to generate synthetic data points in the input space.



$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})}$$



- **Discriminative Models**
  - First solve the inference problem of determining the posterior class probabilities $p(C_k|\mathbf{x})$, and then subsequently use decision theory to assign each new $\mathbf{x}$ to one of the classes. Approaches that model the posterior probabilities *directly* are called *discriminative models*.
- **Discriminant Function**
  - Find a function $f(\mathbf{x})$, called a discriminant function, which maps each input $\mathbf{x}$ directly onto a class label.
  - For instance, in the case of two-class problems, f(·) might be binary valued such that $f = 0$ represents class C_1 and $f = 1$ represents class C_2.
  - In this case, we no longer have access to the posterior probabilities $p(C_k|\mathbf{x})$.

Probabilistic Generative Models

- Here we adopt a generative approach, where we model the class-conditional densities, and class priors, then use these to compute posterior probabilities through Bayes' theorem (using the two-class problem as an example):

$$
\begin{aligned}
p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\
&= \frac{1}{1+\exp(-a)} = \sigma(a)
\end{aligned}
$$

where

$$
a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}
$$

$\sigma(a)$ is the *logistic sigmoid* function defined by

$$
\sigma(a) = \frac{1}{1+\exp(-a)}
$$

The logistic sigmoid function is sometimes called ***logistic*** function.

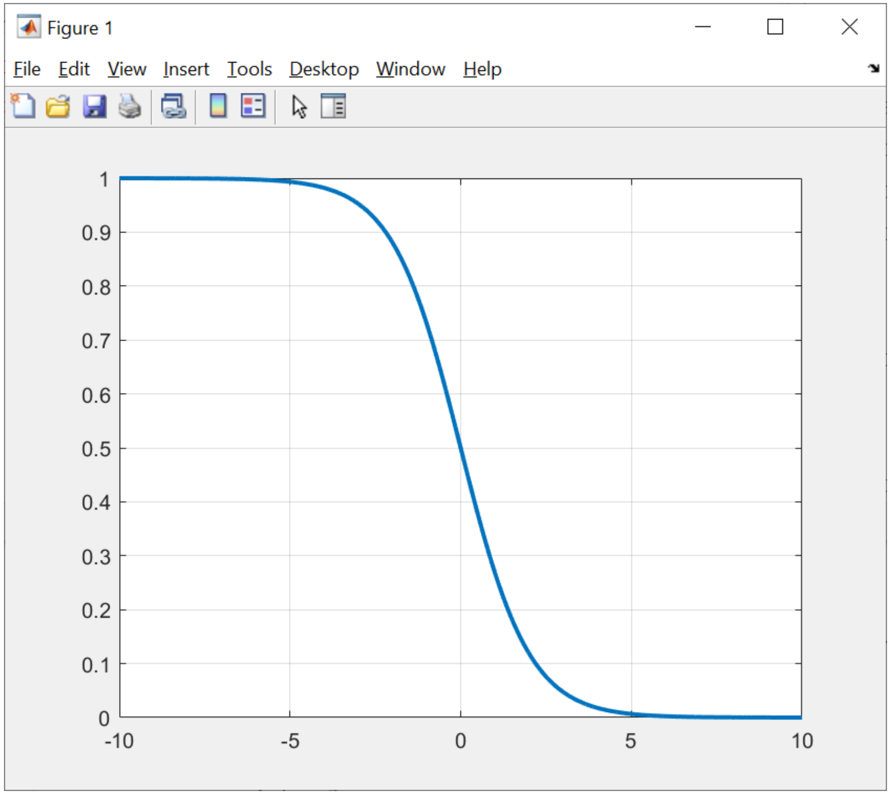
Sigmoid Function

- The term “sigmoid” means S-shaped.
- This type of function is sometimes also called a “squashing function” because it maps the whole real axis into a finite interval.
- It satisfies the following symmetry property

$$\sigma(-a) = 1 - \sigma(a)$$

- The inverse of the logistic sigmoid is given by the ***logit*** function:

$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$



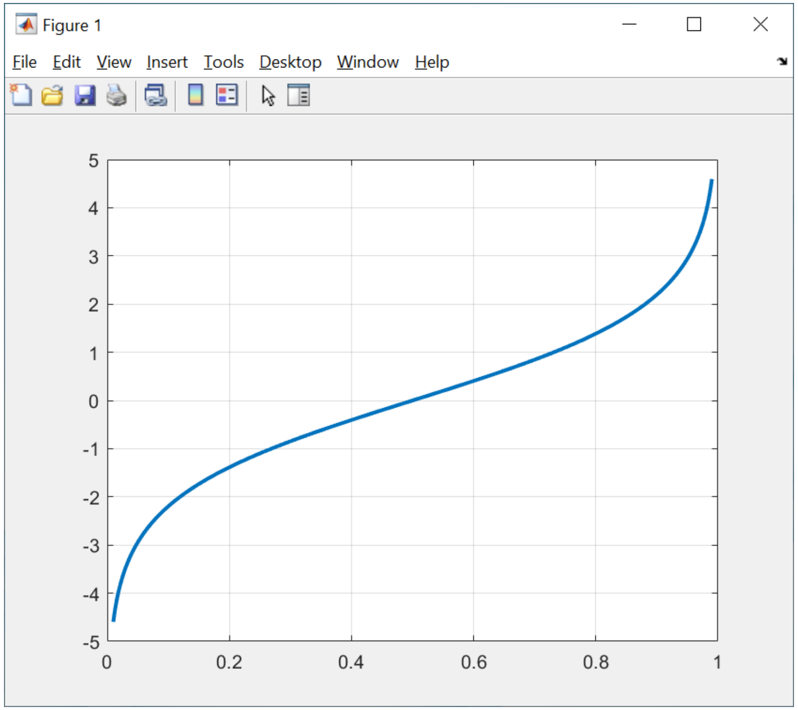

```
>> a = -10: 0.01: 10;
>> s = 1./(1 + exp(a));
>> plot(a, s); grid
```

logit function

The sigmoid function:

$$\sigma(a) = p(C_1|\mathbf{x}) = \frac{1}{1+e^{-a}}$$

where $a(\mathbf{x}) = \ln\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}$



The **logit** function (also called the **log odd** function) represents the log of the ratio of probabilities:

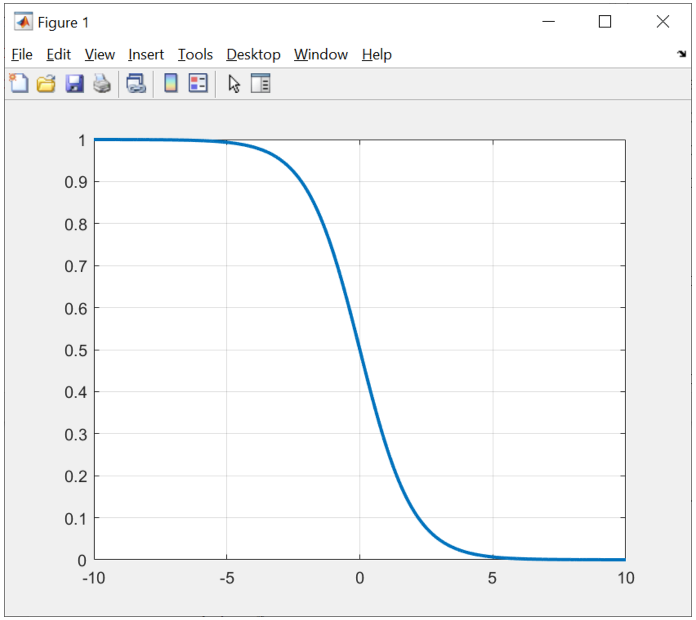
$$a(\mathbf{x}) = \ln\left(\frac{\sigma}{1-\sigma}\right) = \ln\frac{p(C_1|\mathbf{x})}{1-p(C_1|\mathbf{x})} = \ln\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}$$

```
>> s = 0.01: 0.001: 0.99;
>> a = log(s./(1-s));
>> plot(s,a); grid
```

The **logistic** (sigmoid) function and the **logit** function are inverse of each other.

Two Functions Side by Side

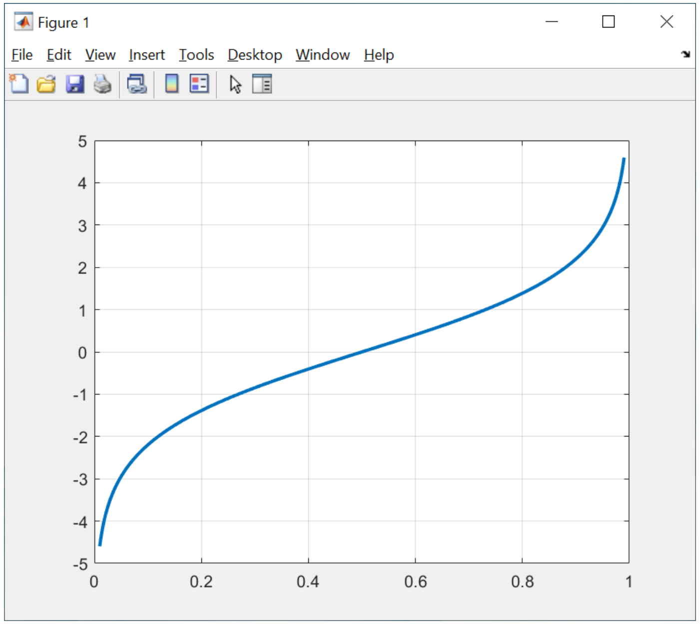
Logistic (sigmoid) function



$$\sigma(a) = p(C_1|\mathbf{x}) = \frac{1}{1+e^{-a}}$$

Logistic function outputs probability

logit function



$$a = \ln\left(\frac{\sigma}{1-\sigma}\right)$$

Logit function outputs log likelihood ratio

The **logistic** (sigmoid) function and the **logit** function are inverse of each other.

Softmax Function

Multiclass ($K > 2$) generalization of the logistic sigmoid to a *normalized exponential*:

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$

In contrast to the two-class case:

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} = \frac{1}{1+\exp(-a)} = \sigma(a)$$

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}$$

The *softmax* function represents a smoothed version of the "max" function because, if $a_k \gg a_j$ for all $j \neq k$, then $p(C_k|\mathbf{x}) \approx 1$, and $p(C_j|\mathbf{x}) \approx \mathbf{0}$.

Parametric Form for $p(C_k|\mathbf{x})$

- Assume that the class-conditional densities are Gaussian.
- We consider first two classes, and assume that all classes share the same covariance matrix.
- Thus the density for class C_k is given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

$$\sigma(a) = p(C_1|\mathbf{x}) = \sigma\left(\ln\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}\right) = \sigma\left(\ln\frac{p(C_1|\mathbf{x})\boldsymbol{p}(\mathbf{x})}{p(C_2|\mathbf{x})\boldsymbol{p}(\mathbf{x})}\right) = \sigma\left(\ln\frac{p(\mathbf{x}|C_1)\boldsymbol{p}(C_1)}{p(\mathbf{x}|C_2)\boldsymbol{p}(C_2)}\right)$$

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$ where

$$\begin{aligned} \mathbf{w} &= \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)} \end{aligned}$$

Similar to Two Classes for LDA

Decision functions (with a common covariance matrix $\boldsymbol{C}$, where $\boldsymbol{C}^{\mathrm{T}} = \boldsymbol{C}$):

$$d_1(\mathbf{x}) = \ln P(\omega_1) - \frac{1}{2}\ln|\boldsymbol{C}| - \frac{1}{2}[(\mathbf{x} - \mathbf{m_1})^{\mathrm{T}}\mathrm{C}^{-1}(\mathbf{x} - \mathbf{m_1})]$$

$$d_2(\mathbf{x}) = \ln P(\omega_2) - \frac{1}{2}\ln|\boldsymbol{C}| - \frac{1}{2}[(\mathbf{x} - \mathbf{m_2})^{\mathrm{T}}\mathrm{C}^{-1}(\mathbf{x} - \mathbf{m_2})]$$

Decision Boundary (assuming equal class probabilities): $d_1(\mathbf{x}) = d_2(\mathbf{x})$

$$(\mathbf{x} - \mathbf{m_1})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m_1}) = (\mathbf{x} - \mathbf{m_2})^{\mathrm{T}}\mathbf{C}^{-1}(\mathbf{x} - \mathbf{m_2})$$

$$\begin{aligned} &\mathbf{x}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{m_1} - \mathbf{m_1^{\mathrm{T}}}\mathbf{C}^{-1}\mathbf{x} + \mathbf{m_1^{\mathrm{T}}}\mathbf{C}^{-1}\mathbf{m_1} \\ = &\mathbf{x}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x} - \mathbf{x}^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{m_2} - \mathbf{m_2^{\mathrm{T}}}\mathbf{C}^{-1}\mathbf{x} + \mathbf{m_2^{\mathrm{T}}}\mathbf{C}^{-1}\mathbf{m_2} \end{aligned}$$

Cancellation
due to the assumption of same covariance (LDA); otherwise quadratic function of $\mathbf{x}$, thus QDA results.

$$(\mathbf{m_1} - \mathbf{m_2})^{\mathrm{T}}\mathbf{C}^{-1}\mathbf{x} = \frac{1}{2}(\mathbf{m_1^{\mathrm{T}}}\mathbf{C}^{-1}\mathbf{m_1} - \mathbf{m_2^{\mathrm{T}}}\mathbf{C}^{-1}\mathbf{m_2})$$

Maximum Likelihood Solution

- Once we have specified a parametric functional form for the class-conditional densities, we can then determine the values of the parameters, together with the prior class probabilities $p(C_k)$, using maximum likelihood.
- This requires a data set comprising observations of **x** along with their corresponding class labels.
- Consider first the case of two classes, each having a Gaussian class-conditional density with a shared covariance matrix, and suppose we have a data set $\{\mathbf{x}_n, t_n\}$, where $n = 1, \ldots, N$. Here $t_n = 1$ denotes class C_1 and $t_n = 0$ denotes class C_2.
- We denote the prior class probability $p(C_1) = \pi$, so that $p(C_2) = 1 - \pi$.
- For a data point $\mathbf{x}_n$ from class C_1, we have $t_n = 1$ and hence

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}).$$

- Similarly, for a data point $\mathbf{x}_n$ from class C_2, we have $t_n = 0$ and hence

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}).$$

The likelihood function is given by

$$p(\mathbf{t}|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \boldsymbol{\Sigma}) = \prod_{n=1}^{N} [\pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma})]^{t_n} [(1-\pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \boldsymbol{\Sigma})]^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$

It is convenient to maximize the log of the likelihood function.

- Consider first the maximization with respect to π.
  - The terms in the log likelihood function that depend on π are

$$\sum_{n=1}^{N} \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\}$$

  - Setting the derivative with respect to π equal to zero, we obtain

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}$$

  - Thus the maximum likelihood estimate for π is the fraction of points in class C_1 as expected. This can be generalized to the multiclass case, where the maximum likelihood estimate of the prior probability associated with class C_k is given by the fraction of the training set points assigned to that class.

.

Maximum Likelihood Estimate of the Means

- We can pick out of the log likelihood function those terms that depend on $\boldsymbol{\mu}_1$

$$\sum_{n=1}^{N} t_n \ln \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}) = -\frac{1}{2}\sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1) + \text{const.}$$

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}).$$

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\boldsymbol{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)^{\mathrm{T}} \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu}_k)\right\}$$

- Setting the derivative with respect to $\boldsymbol{\mu}_1$ to zero, we can obtain

$$\boldsymbol{\mu}_1 = \frac{1}{N_1}\sum_{n=1}^{N} t_n \mathbf{x}_n$$

  which is simply the mean of all the input vectors x_n assigned to class C_1.

- By a similar argument, we have

$$\boldsymbol{\mu}_2 = \frac{1}{N_2}\sum_{n=1}^{N} (1 - t_n) \mathbf{x}_n$$

  which is simply the mean of all the input vectors x_n assigned to class C_2.

Matrix Calculus

For a scalar α given by a quadratic form: $\alpha = \mathbf{x}^T \mathbf{A} \mathbf{x}$

where $\mathbf{x}$ *is* $n \times 1$, $\mathbf{A}$ *is* $n \times n$, *and* $\mathbf{A}$ *does not depend on* $\mathbf{x}$, *then*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T \left(\mathbf{A} + \mathbf{A}^T\right)$$

Proof: *By definition*

$$\alpha = \sum_{j=1}^{n} \sum_{i=1}^{n} a_{ij} x_i x_j$$

Differentiating with respect to the k*th element of* $\mathbf{x}$ *we have*

$$\frac{\partial \alpha}{\partial x_k} = \sum_{j=1}^{n} a_{kj} x_j + \sum_{i=1}^{n} a_{ik} x_i$$

for all $k = 1, 2, \ldots, n$, *and consequently,*

$$\frac{\partial \alpha}{\partial \mathbf{x}} = \mathbf{x}^T \mathbf{A}^T + \mathbf{x}^T \mathbf{A} = \mathbf{x}^T \left(\mathbf{A}^T + \mathbf{A}\right)$$

For the special case $\mathbf{A}^{\mathrm{T}} = \mathbf{A}$, then $\frac{\partial [\mathbf{x}^{\mathrm{T}}(\mathbf{A}+\mathbf{A}^{\mathrm{T}})]}{\partial x} = 2\mathbf{x}^{\mathrm{T}}\mathbf{A}$

Maximum Likelihood Solution for the Shared Covariance Matrix

$$-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N} t_n(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1)$$

$$-\frac{1}{2}\sum_{n=1}^{N}(1-t_n)\ln|\boldsymbol{\Sigma}| - \frac{1}{2}\sum_{n=1}^{N}(1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_2) = -\frac{N}{2}\ln|\boldsymbol{\Sigma}| - \frac{N}{2}\mathrm{Tr}\left\{\boldsymbol{\Sigma}^{-1}\mathbf{S}\right\}$$

where we defined

$$\begin{aligned}
\mathbf{S} &= \frac{N_1}{N}\mathbf{S}_1 + \frac{N_2}{N}\mathbf{S}_2 \\
\mathbf{S}_1 &= \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^{\mathrm{T}} \\
\mathbf{S}_2 &= \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^{\mathrm{T}}.
\end{aligned}$$

Setting to zero the derivative of the above expression with respect to $\boldsymbol{\Sigma}^{-1}$, we can show that $\boldsymbol{\Sigma} = \mathbf{S}$, where $\mathbf{S}$ represents a weighted average of the covariance matrices associated with each of the two classes separately.
Details of the proof are as follows.

Linear Algebra and Calculus Formulas

- The trace is invariant under cyclic permutations of matrix products: $\mathrm{tr}\,[ABC] = \mathrm{tr}\,[CAB] = \mathrm{tr}\,[BCA]$
- Since $x^T Ax$ is scalar, we can take its trace and obtain the same value: $x^T Ax = \mathrm{tr}\,\left[x^T Ax\right] = \mathrm{tr}\,\left[xx^T A\right]$
- $\frac{\partial}{\partial A}\mathrm{tr}\,[AB] = B^T$
- $\frac{\partial}{\partial A}\log|A| = (A^{-1})^T = (A^T)^{-1}$
- The determinant of the inverse of an invertible matrix is the inverse of the determinant: $|A| = \frac{1}{|A^{-1}|}$

$$\frac{\partial}{\partial A}x^T Ax = \frac{\partial}{\partial A}\mathrm{tr}\,\left[xx^T A\right] = [xx^T]^T = \left(x^T\right)^T x^T = xx^T$$

Cyclic Property

$$\operatorname{tr}(\mathbf{AB}) = \sum_{i=1}^{m} (\mathbf{AB})_{ii} = \sum_{i=1}^{m} \sum_{j=1}^{n} a_{ij} b_{ji} = \sum_{j=1}^{n} \sum_{i=1}^{m} b_{ji} a_{ij} = \sum_{j=1}^{n} (\mathbf{BA})_{jj} = \operatorname{tr}(\mathbf{BA}).$$

$$\operatorname{tr}[ABC] = \operatorname{tr}[CAB] = \operatorname{tr}[BCA]$$

```
>> A = rand(2,2); B = rand(2,2); C = rand(2,2);
>> trace(A*B*C); trace(B*C*A); trace(C*A*B)
```

Derivative of the Trace of Matrix Product

$$\frac{\partial}{\partial A}\mathrm{tr}\,[AB] = B^T$$

$$\begin{aligned}
\mathrm{tr}AB &= \mathrm{tr}\begin{bmatrix} \longleftarrow \vec{a_1} \longrightarrow \\ \longleftarrow \vec{a_2} \longrightarrow \\ \vdots \\ \longleftarrow \vec{a_n} \longrightarrow \end{bmatrix} \begin{bmatrix} \uparrow & \uparrow & & \uparrow \\ \vec{b_1} & \vec{b_2} & \cdots & \vec{b_n} \\ \downarrow & \downarrow & & \downarrow \end{bmatrix} \\
&= \mathrm{tr}\begin{bmatrix} \vec{a_1}^T\vec{b_1} & \vec{a_1}^T\vec{b_2} & \cdots & \vec{a_1}^T\vec{b_n} \\ \vec{a_2}^T\vec{b_1} & \vec{a_2}^T\vec{b_2} & \cdots & \vec{a_2}^T\vec{b_n} \\ \vdots & & \ddots & \vdots \\ \vec{a_n}^T\vec{b_1} & \vec{a_n}^T\vec{b_2} & \cdots & \vec{a_n}^T\vec{b_n} \end{bmatrix} \\
&= \sum_{i=1}^{m} a_{1i}b_{i1} + \sum_{i=1}^{m} a_{2i}b_{i2} + \ldots + \sum_{i=1}^{m} a_{ni}b_{in} \\
\Rightarrow \quad \frac{\partial \mathrm{tr}AB}{\partial a_{ij}} &= b_{ji} \\
\Rightarrow \quad \nabla_A \mathrm{tr}AB &= B^T
\end{aligned}$$

Determinant and Adjoint Matrix

```
>> A = magic(2)
A =
    1    3
    4    2
>> det(A)
ans =
  -10
>> X = adjoint(A)
ans =
   2.0000   -3.0000
  -4.0000    1.0000

>> A*X
ans =
 -10.0000   -0.0000
       0  -10.0000
```

The adjugate or classical adjoint of a square matrix is the ***transpose*** of its cofactor matrix.

The (j,i)-th cofactor of A is defined as follows.

$$a_{ji}' = (-1)^{i+j}\det(A_{ij})$$

A_{ij} is the submatrix of A obtained from A by removing the i-th row and j-th column.

cofactor of A:

```
   2.0000   -4.0000
  -3.0000    1.0000
```

X = adjoint(A) returns the Classical Adjoint (Adjugate) Matrix X of A, such that A*X = det(A)*eye(n) = X*A, where n is the number of rows in A.

- The (i, j) **minor** of A, denoted M_{ij} is the determinant of the $(n-1) \times (n-1)$ matrix that remains after removing the ith row and jth column from A.
- The **cofactor** matrix of A, denoted C, is an $n \times\ n$ matrix such that $C_{ij}\ =\ (-1)^{i+j} M_{ij}$.
- The **adjugate** matrix of A, denoted $\mathrm{adj}(A)$, is simply the transpose of C.
- If A is invertible, then $A^{-1} = \frac{1}{|A|}\mathrm{adj}(A)$, so $(A^{-1})_{ij}^{\mathrm{T}} = \frac{1}{|A|} C_{ij}$.

```
>> X = adjoint(A)
X =
   2.0000   -3.0000
  -4.0000    1.0000
>> X/det(A)
ans =
  -0.2000    0.3000
   0.4000   -0.1000
>> inv(A)
ans =
  -0.2000    0.3000
   0.4000   -0.1000
```

```
A =
     1     3
     4     2
```

```
>> inv(A)'
ans =
  -0.2000    0.4000
   0.3000   -0.1000
>> C = X'
C =
   2.0000   -4.0000
  -3.0000    1.0000
>> C/det(A)
ans =
  -0.2000    0.4000
   0.3000   -0.1000
```

Cofactor Expansion of the Determinant

$|A| = \Sigma_{k=1}^{n} A_{ik} C_{ik}$, thus

The derivative of a scalar function $|A|$, of the matrix A of independent variables, with respect to (each of the elements of) the matrix A is:

$$\frac{\partial |A|}{\partial A_{ij}} = \sum_{k=1}^{n} \frac{\partial A_{ik}}{\partial A_{ij}} C_{ik} + A_{ik} \frac{\partial C_{ik}}{\partial A_{ij}} = C_{ij} + 0 = C_{ij}$$

For any k, the elements of A which affect C_{ik} are those which do not lie on row *i* or column k. Hence, $\frac{\partial C_{ik}}{\partial A_{ij}} = 0$ for all k!

```
>> A
A =
[ a, b]
[ c, d]
>> det(A)
ans =
a*d - b*c
```

```
>> [diff(det(A),a), diff(det(A),b); diff(det(A),c), diff(det(A),d)]
ans =
[  d, -c]
[ -b,  a]
```

```
>> adjoint(A)
ans =
[  d, -b]
[ -c,  a]
```

Cofactor matrix is the transpose:

```
[  d, -c]
[ -b,  a]
```

Differentiation of the Log Determinant

$$\frac{\partial \ln|A|}{\partial A_{ij}} = \frac{1}{|A|}\frac{\partial |A|}{\partial A_{ij}} = \frac{1}{|A|}C_{ij} = (A^{-1})^{\mathrm{T}}_{ij}$$

since $(A^{-1})^{\mathrm{T}}_{ij} = \frac{1}{|A|}C_{ij}$

$$\frac{\partial}{\partial A}\log|A| = (A^{-1})^T = (A^T)^{-1}$$

```
>> f = log(det(A))
f =
log(a*d - b*c)
>> [diff(f,a), diff(f,b); diff(f,c), diff(f,d)]
ans =
[  d/(a*d - b*c), -c/(a*d - b*c)]
[ -b/(a*d - b*c),  a/(a*d - b*c)]
```

```
>> inv(A).'
ans =
[  d/(a*d - b*c), -c/(a*d - b*c)]
[ -b/(a*d - b*c),  a/(a*d - b*c)]
```

```
>> inv(A.')
ans =
[  d/(a*d - b*c), -c/(a*d - b*c)]
[ -b/(a*d - b*c),  a/(a*d - b*c)]
```

$$
\begin{aligned}
l(\mu, \mathbf{\Sigma}|\mathbf{x^{(i)}}) &= \log \prod_{i=1}^{m} f_{\mathbf{X^{(i)}}}(\mathbf{x^{(i)}}|\mu, \mathbf{\Sigma}) \\
&= \log \prod_{i=1}^{m} \frac{1}{(2\pi)^{p/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{x^{(i)}}-\mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{x^{(i)}}-\mu)\right) \\
&= \sum_{i=1}^{m}\left(-\frac{p}{2}\log(2\pi) - \frac{1}{2}\log|\Sigma| - \frac{1}{2}(\mathbf{x^{(i)}}-\mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{x^{(i)}}-\mu)\right)
\end{aligned}
$$

$$
l(\mu, \Sigma;) = -\frac{mp}{2}\log(2\pi) - \frac{m}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x^{(i)}}-\mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{x^{(i)}}-\mu)
$$

$$
\frac{\partial}{\partial A}x^T Ax = \frac{\partial}{\partial A}\mathrm{tr}\left[xx^T A\right] = [xx^T]^T = \left(x^T\right)^T x^T = xx^T
$$

$$
\begin{aligned}
l(\mu, \mathbf{\Sigma}|\mathbf{x^{(i)}}) &= \mathrm{C} - \frac{m}{2}\log|\Sigma| - \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x^{(i)}}-\mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}(\mathbf{x^{(i)}}-\mu) \\
&= \mathrm{C} + \frac{m}{2}\log|\Sigma^{-1}| - \frac{1}{2}\sum_{i=1}^{m}\mathrm{tr}\left[(\mathbf{x^{(i)}}-\mu)(\mathbf{x^{(i)}}-\mu)^{\mathbf{T}}\mathbf{\Sigma^{-1}}\right]
\end{aligned}
$$

$$
\frac{\partial}{\partial \Sigma^{-1}} l(\mu, \mathbf{\Sigma}|\mathbf{x^{(i)}}) = \frac{m}{2}\Sigma - \frac{1}{2}\sum_{i=1}^{m}(\mathbf{x^{(i)}}-\mu)(\mathbf{x^{(i)}}-\mu)^T \quad \text{Since } \Sigma^T = \Sigma
$$

$$
\begin{aligned}
0 &= m\Sigma - \sum_{i=1}^{m}(\mathbf{x^{(i)}}-\mu)(\mathbf{x^{(i)}}-\mu)^T \\
\hat{\Sigma} &= \frac{1}{m}\sum_{i=1}^{m}(\mathbf{x^{(i)}}-\hat{\mu})(\mathbf{x^{(i)}}-\hat{\mu})^T
\end{aligned}
$$

Generalized Linear Models and Link Function

- So far we have considered classification models that work directly with the original input vector $\mathbf{x}$.
- We can also make a fixed nonlinear transformation of the inputs using a vector of **basis functions** $\phi(\mathbf{x})$.
- The resulting decision boundaries will be linear in the feature space Φ, and these correspond to nonlinear decision boundaries in the original $\mathbf{x}$ space.
- We begin our treatment of generalized linear models by considering the problem of two-class classification.
- Extension of logistic sigmoid function representation of the posterior probability from $\sigma(a) = p(C_1|\mathbf{x}) = \frac{1}{1+e^{-a}}$, where $\mathrm{a}(\mathbf{x}) = \ln\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}$ is the *logic* function, to **Logistic Regression** as follows:
$p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^{\mathrm{T}}\phi)$, and $p(C_2|\phi) = 1 - p(C_1|\phi)$.
- $\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \sigma^{-1}[y(\phi)] = \mathrm{a}[p(C_1|\phi(\mathbf{x}))]$. The inverse of the sigmoid – the *logit* function is called the **link function**, which converts the probability of the response variables to a generalized linear combination of explanatory variables (input vector $\mathbf{x}$).
- We have seen an example of logistic regression previously, when we fitted Gaussian class conditional densities.

Parametric Form for $p(C_k|\mathbf{x})$

- Assume that the class-conditional densities are Gaussian.
- We consider first two classes, and assume that all classes share the same covariance matrix.
- Thus the density for class C_k is given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{(2\pi)^{D/2}} \frac{1}{|\mathbf{\Sigma}|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^{\mathrm{T}}\mathbf{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\right\}$$

$$\sigma(a) = p(C_1|\mathbf{x}) = \sigma\left(\ln\frac{p(C_1|\mathbf{x})}{p(C_2|\mathbf{x})}\right) = \sigma\left(\ln\frac{p(C_1|\mathbf{x})\boldsymbol{p}(\mathbf{x})}{p(C_2|\mathbf{x})\boldsymbol{p}(\mathbf{x})}\right) = \sigma\left(\ln\frac{p(\mathbf{x}|C_1)\boldsymbol{p}(C_1)}{p(\mathbf{x}|C_2)\boldsymbol{p}(C_2)}\right)$$

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0)$$

where the weight and bias are based on the means and covariance matrix estimated by the MLE method – too many parameters to estimate!

$$\mathbf{w} = \mathbf{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\mathbf{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

Logistic Regression

- In a two-class classification problem, the posterior probability of class C_1 can be written as a logistic sigmoid acting on a linear function of the feature vector ϕ so that
  $p(C_1|\phi) = y(\phi) = \sigma(\mathbf{w}^{\mathrm{T}}\phi)$
- Note that this is a model for classification rather than regression.
- For an M-dimensional feature space Φ, this model has M adjustable parameters.
- By contrast, when we previously fitted Gaussian class conditional densities using maximum likelihood, we would have used $2M$ parameters for the means, and $\frac{M(M+1)}{2}$ parameters for the (shared) covariance matrix. Together with the class prior $p(C_1)$, this gives a total of $\frac{M(M+5)}{2} + 1$ parameters, which grows quadratically with M.
- For large values of M, there is a clear advantage in working with the logistic regression model **directly**.
- To determine the parameters of the logistic regression model, we can use
  - Maximum likelihood
  - Iterative reweighted least squares

Maximum Likelihood

- For a data set $\{\Phi_n, t_n\}$, where $t_n \in \{0, 1\}$ and $\phi_n = \phi(\mathbf{x}_n)$, with $n = 1, \ldots, N$, the likelihood function can be written as

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n} \{1 - y_n\}^{1-t_n}$$

where $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ and $y_n = p(C_1|\phi_n) = \sigma(a_n)$, and $a_n = \mathbf{w}^{\mathrm{T}}\phi_n$.

- we define an *error function* by taking the negative logarithm of the likelihood, which gives the ***cross-entropy*** error function as

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\}$$

- Taking the gradient of the error function with respect to $\mathbf{w}$, by making use of the derivative of the logistic sigmoid function $\sigma(a_n) = \frac{1}{1+e^{-a_n}}$, as $\frac{d\sigma(a_n)}{da_n} = \sigma(1 - \sigma)$, we obtain:

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n)\phi_n$$

$$y_n = p(C_1|\phi_n) = \sigma(a_n), \quad \frac{dy_n}{da_n} = \sigma(1-\sigma) = y_n(1-y_n), \quad a_n = \mathbf{w}^{\mathrm{T}}\phi_n$$

$$\frac{d(t_n \ln y_n)}{d\mathbf{w}} = \frac{d(t_n \ln y_n)}{da_n}\frac{da_n}{d\mathbf{w}} = \frac{t_n y_n(1-y_n)\frac{da_n}{d\mathbf{w}}}{y_n} = t_n(1-y_n)\phi_n$$

$$\frac{d[(1-t_n)\ln(1-y_n)]}{d\mathbf{w}} = \frac{-(1-t_n)y_n(1-y_n)\frac{da_n}{d\mathbf{w}}}{(1-y_n)} = -(1-t_n)y_n\phi_n$$

$$\nabla E(\mathbf{w}) = -\sum_{n=1}^{N}(t_n\phi_n - y_n\phi_n) = \sum_{n=1}^{N}(y_n - t_n)\phi_n$$

- The contribution to the gradient of the log likelihood from data point n is given by the "error" $(y_n - t_n)$ between the target value and the prediction of the model, times the basis function vector ϕ_n.
- While there is no closed-form solution to the minimization problem, the error function can be minimized by an efficient iterative technique based on the ***Newton-Raphson*** iterative optimization scheme.

Newton' Method

- Newton's method is an iterative method, $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$, for finding the roots of a differentiable function F, which are solutions to the equation $f(x) = 0$.
- Newton's method can be applied to the derivative f' of a twice-differentiable function f to find the roots of the derivative (solutions to $f'(x) = 0$), which are known as the critical points of f. These solutions may be minima, maxima, or saddle points.
- The second-order Taylor expansion of f around x_k is

$$f(x_k + t) \approx f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2.$$

- If the second derivative is positive, the quadratic approximation is a convex function of t, and its minimum can be found by:

$$0 = \frac{\mathrm{d}}{\mathrm{d}t}\left(f(x_k) + f'(x_k)t + \frac{1}{2}f''(x_k)t^2\right) = f'(x_k) + f''(x_k)t,$$

$$x_{k+1} = x_k + t = x_k - \frac{f'(x_k)}{f''(x_k)}$$

The Newton-Raphson Method for $E(\mathbf{w})$

$$\mathbf{w}^{(\text{new})} = \mathbf{w}^{(\text{old})} - \mathbf{H}^{-1}\nabla E(\mathbf{w})$$

where $\mathbf{H}$ is the Hessian matrix whose elements comprise the second derivatives of $E(\mathbf{w})$ with respect to the components of $\mathbf{w}$.

$$\nabla E(\mathbf{w}) = \sum_{n=1}^{N}(y_n - t_n)\phi_n = \Phi^{\mathrm{T}}(\mathbf{y} - \mathbf{t})$$

$\frac{dy_n}{da_n} = \sigma(1-\sigma) = y_n(1-y_n)$, and $a_n = \mathbf{w}^{\mathrm{T}}\phi_n$

$$\nabla\nabla E(\mathbf{w}) = \frac{d[\sum_{n=1}^{N}(y_n - t_n)\phi_n^{\mathrm{T}}]}{\mathrm{d}\mathbf{w}} = \frac{d[\sum_{n=1}^{N}(y_n - t_n)\phi_n^{\mathrm{T}}]}{\mathrm{d}a_n}\frac{da_n}{\mathrm{d}\boldsymbol{w}}$$

$$\mathbf{H} = \nabla\nabla E(\mathbf{w}) = \sum_{n=1}^{N} y_n(1-y_n)\phi_n\phi_n^{\mathrm{T}} = \Phi^{\mathrm{T}}\mathbf{R}\Phi$$

Where $\mathbf{R}$ is the $N \times N$ diagonal matrix with elements

$R_{nn} = y_n(1-y_n)$ where $y_n = p(C_1|\phi_n) = \sigma(\mathbf{w}^{\mathrm{T}}\phi_n)$

$\mathbf{R}$ is a weighing matrix, which is not constant but depends on the parameter vector $\mathbf{w}$.

Iterative Reweighted Least Squares

$$\begin{aligned} \mathbf{w}^{(\text{new})} &= \mathbf{w}^{(\text{old})} - (\Phi^{\mathrm{T}}\mathbf{R}\Phi)^{-1}\Phi^{\mathrm{T}}(\mathbf{y}-\mathbf{t}) \\ &= (\Phi^{\mathrm{T}}\mathbf{R}\Phi)^{-1}\left\{\Phi^{\mathrm{T}}\mathbf{R}\Phi\mathbf{w}^{(\text{old})} - \Phi^{\mathrm{T}}(\mathbf{y}-\mathbf{t})\right\} \\ &= (\Phi^{\mathrm{T}}\mathbf{R}\Phi)^{-1}\Phi^{\mathrm{T}}\mathbf{R}\mathbf{z} \end{aligned}$$

where $\mathbf{z}$ is an N-dimensional vector with elements

$$\mathbf{z} = \Phi\mathbf{w}^{(\text{old})} - \mathbf{R}^{-1}(\mathbf{y}-\mathbf{t})$$

- The update formula takes the form of a set of **normal equations** for a weighted least-squares problem.
- Because the weighing matrix $\mathbf{R}$ depends on the parameter vector $\mathbf{w}$, we must apply the normal equations iteratively, each time using the new weight vector $\mathbf{w}$ to compute a revised weighing matrix $\mathbf{R}$.
- For this reason, the algorithm is known as *iterative reweighted least squares* (IRLS).

Extension to Multiclass Problem

Previously,

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_j p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

where $a_k = \ln p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)$

- In multiclass classification, the posterior probabilities are given by a softmax transformation of linear functions of the feature variables, so that

$$p(\mathcal{C}_k|\phi) = y_k(\phi) = \frac{\exp(a_k)}{\sum_j \exp(a_j)}$$

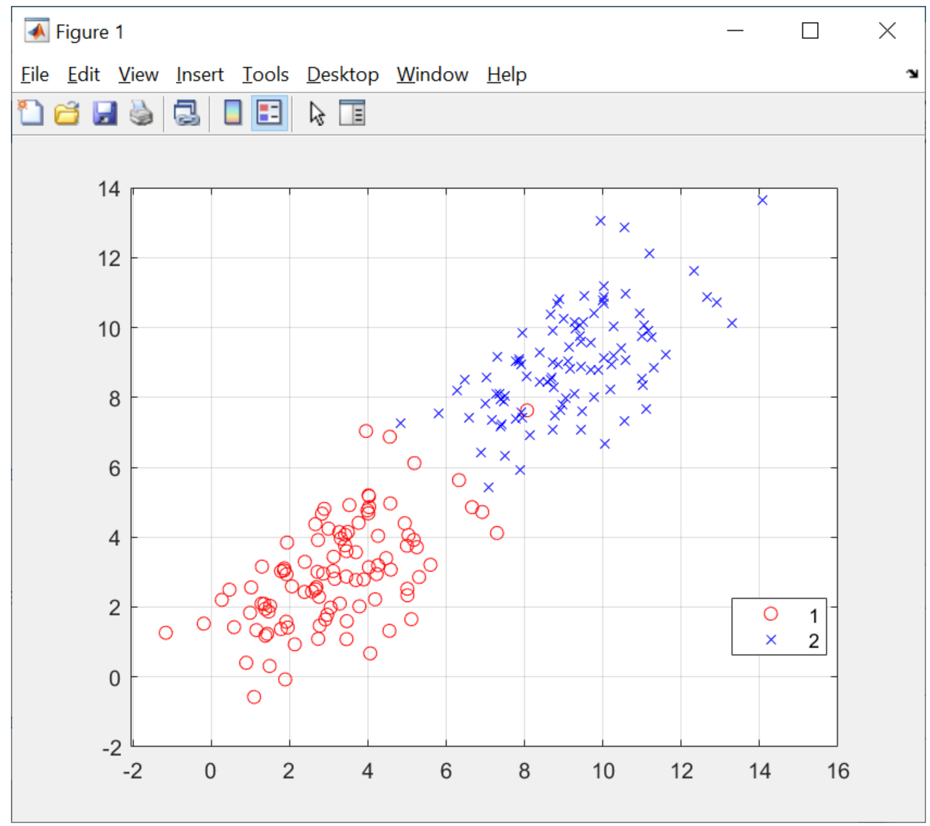
where the "activations" are given by $a_k = \mathbf{w}_k^{\mathrm{T}}\phi.$

- We can use the maximum likelihood to determine the parameters $\{\mathbf{w}_k\}$ of this model directly.
- Similarly, we can appeal to the Newton-Raphson update to obtain the corresponding IRLS algorithm for the multiclass problem.

‘logistic_regression_demo.m’

```
m1 = [3, 3]';    % Mean vector
cov1 = [2 1; 1 2]; % Covariance matrix
rng default
r1 = mvnrnd(m1,cov1,N);

m2 = [9, 9]';
cov2 = [2,1; 1,2];  % Same as cov1
r2 = mvnrnd(m2,cov2,N);
```



Mnrfit( ) Function

```
% logistic regression
B = mnrfit(data, label);

% Maximum likelihood estimates
mu1 = mean(data_C1);
mu2 = mean(data_C2);
Sigma1 = cov(data_C1);
Sigma2 = cov(data_C2);
Sigma = 1/2*(Sigma1 + Sigma2);

% Slopes and intercepts based on theoretical results
inv(Sigma)*(mu1-mu2)'
-0.5*mu1*inv(Sigma)*mu1' + 0.5*mu2*inv(Sigma)*mu2'
```

```
>> B
B =
  19.0582
  -1.2745
  -1.7747
```

$$\mathbf{w} = \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2)$$

$$w_0 = -\frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 + \ln\frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}$$

mnrval ( ) function and Decision Boundary

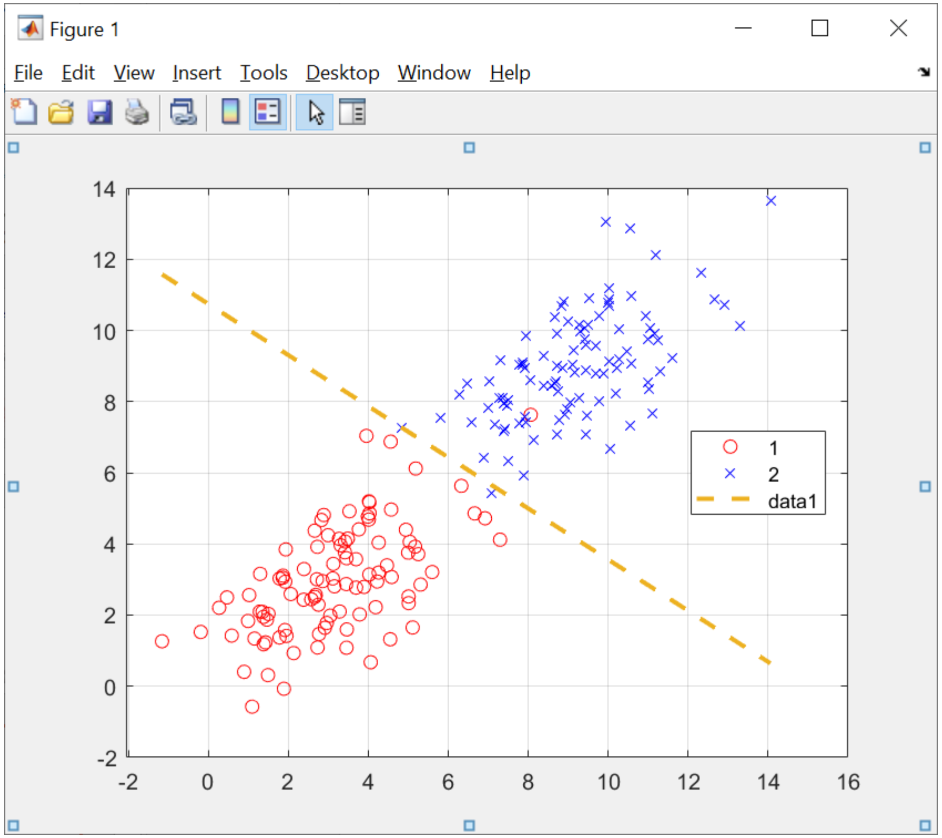
```
>> B
B =
   19.0582
   -1.2745
   -1.7747
```

```
>> x = mean(data)
x =
   6.1741    5.9980
```

```
>> prob = mnrval(B, x)
prob =
   0.6329    0.3671
```

```
>> B(2)*x(1) + B(3)*x(2) + B(1)
ans =
   0.5447
```

```
>> log(prob(1)/prob(2))
ans =
   0.5447
```




```
% Boundary: B(2)*x1 + B(3)*x2 + B(1)  = 0
% log(P(C1|X)/P(C2|X)) >= 0, if P(C1|X)>=P(C2|X)

x1 = min(data(:,1)): 0.01: max(data(:,1));
x2 =  -(B(2)*x1 + B(1))/B(3);
plot(x1,x2)
```

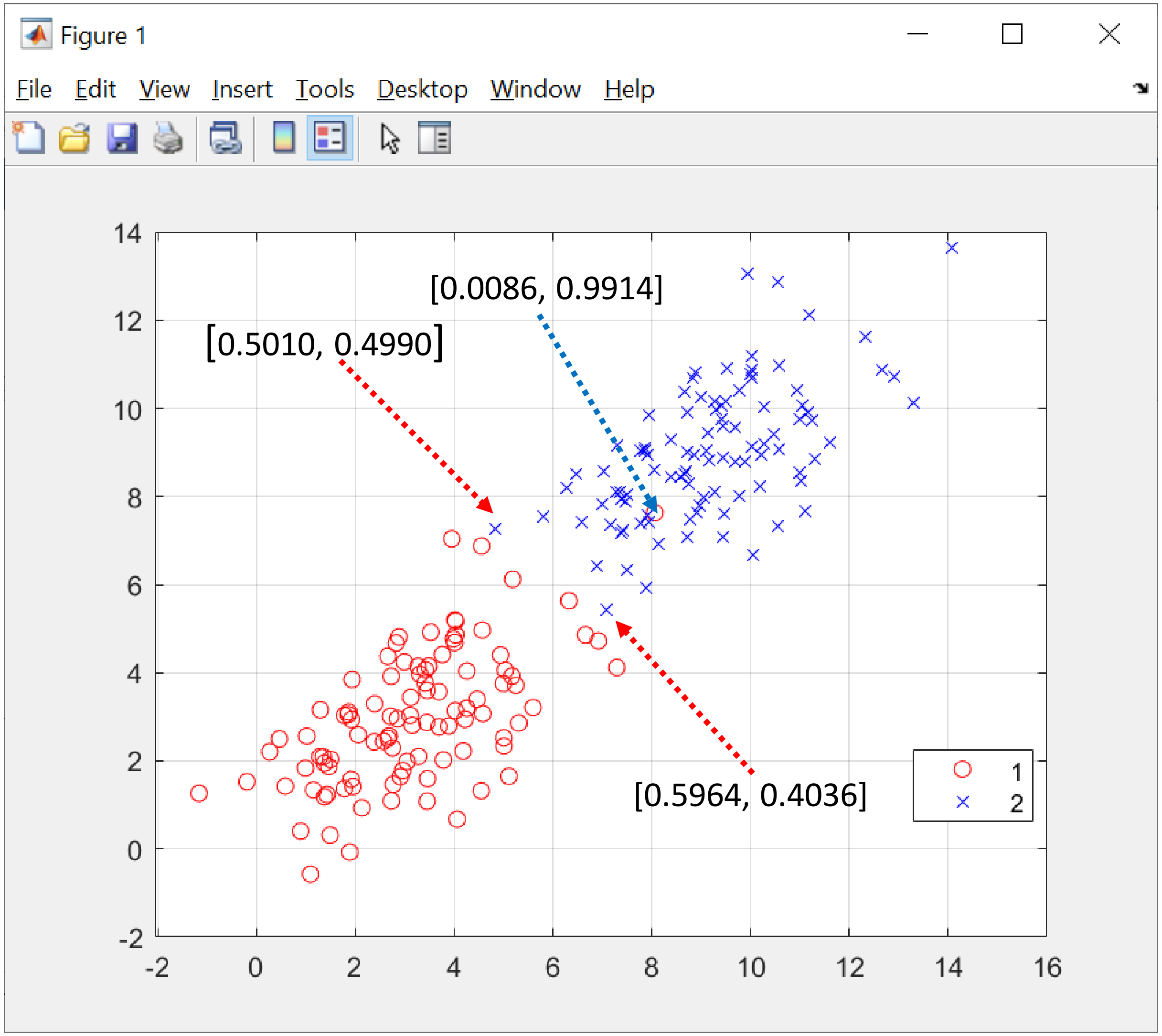
Classification Error Performance

```
P = mnrval(B, data);

label_pred = (P(:,1)>=0.5)
+ 2*(P(:,2)>=0.5);

find(label ~= label_pred)

length(find(label ~=
label_pred))/length(label)
```



sklearn

```
import numpy as np
infile = r"C:\...\logistic_regress.csv“

dataset = np.loadtxt(infile, delimiter=',')
X = dataset[:, 0:2]
y = dataset[:,2]  # labels

from sklearn.linear_model import LogisticRegression as LG

clf = LG().fit(X, y)
clf.intercept_
clf.coef_

clf.score(X,y)

y_pred = clf.predict(X)
num_errors = np.sum(y != y_pred)
num_errors/np.size(y)
```

Summary

- Logistic Regression fits a model that can predict the probability of multi-class responses belonging to one of the multiple classes.
- Logistic regression is a classification method, which provides the posterior class probabilities.
- Because of its simplicity, logistic regression is commonly used as a starting point for binary classification problems.
- Logistic regression can be used to fit a generalized linear model, which involves fitting the response data onto a linear combination of some fixed basis function on the input data, instead of fitting the responses directly onto a linear combination of the input data.
- Best used ...
  - When data can be clearly separated by a single, linear boundary.
  - As a baseline for evaluating more complex classification methods.