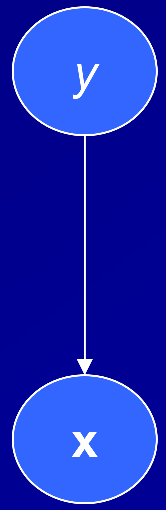
Linear Discriminant Analysis

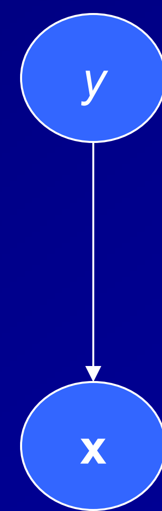
- Learn P(**x**,*y*). This is sometimes called the <u>generative</u> approach, because we can think of P(**x**,*y*) as a model of how the data is generated.
  - For example, if we factor the joint distribution into the form

    $P(\mathbf{x},y) = P(y)\,P(\mathbf{x} \mid y)$
  - we can think of P(*y*) as "generating" a value for *y* according to P(*y*). Then we can think of P(**x** | *y*) as generating a value for **x** given the previously-generated value for *y*.
  - This can be described as a Bayesian network



Linear Discriminant Analysis (2)

- P(y) is a discrete multinomial distribution
  - example: P(y = 0) = 0.31, P(y = 1) = 0.69 will generate 31% negative examples and 69% positive examples
- For LDA, we assume that P($\mathbf{x}$ | y) is a multivariate normal distribution with mean μ_k and covariance matrix Σ



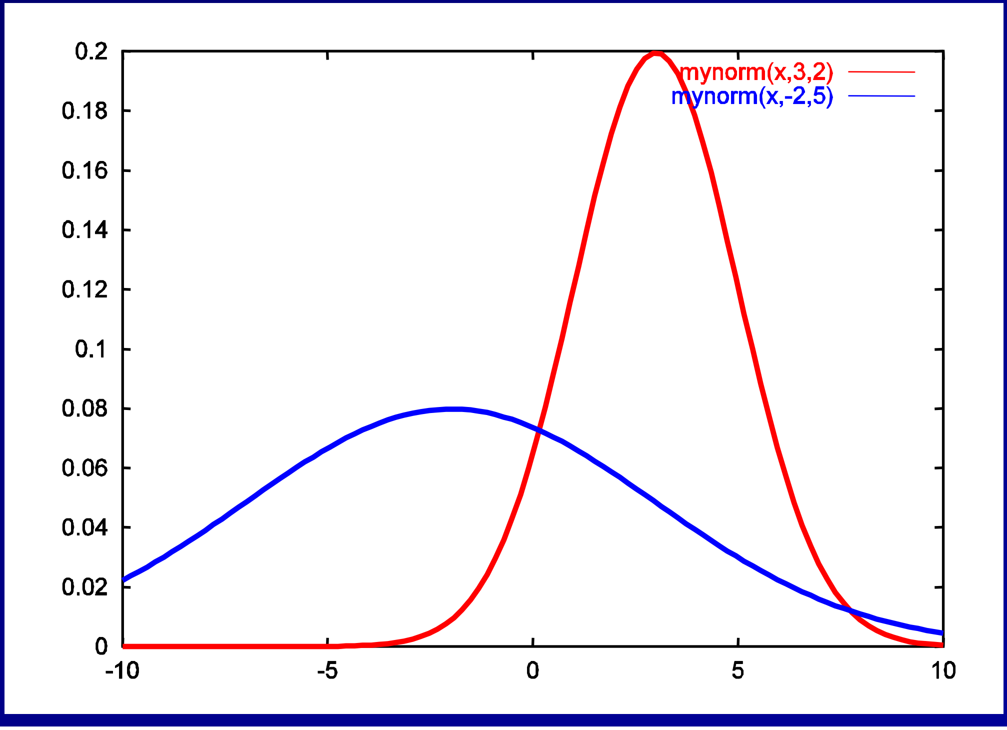
$$P(\mathbf{x}|y=k) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{x}-\mu_k]^T\Sigma^{-1}[\mathbf{x}-\mu_k]\right)$$

Multivariate Normal Distributions: A tutorial

- Recall that the univariate normal (Gaussian) distribution has the formula

$$p(x) = \frac{1}{(2\pi)^{1/2}\sigma} \exp\left[-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right]$$

- where μ is the mean and σ^2 is the variance
- Graphically, it looks like this:



The Multivariate Gaussian

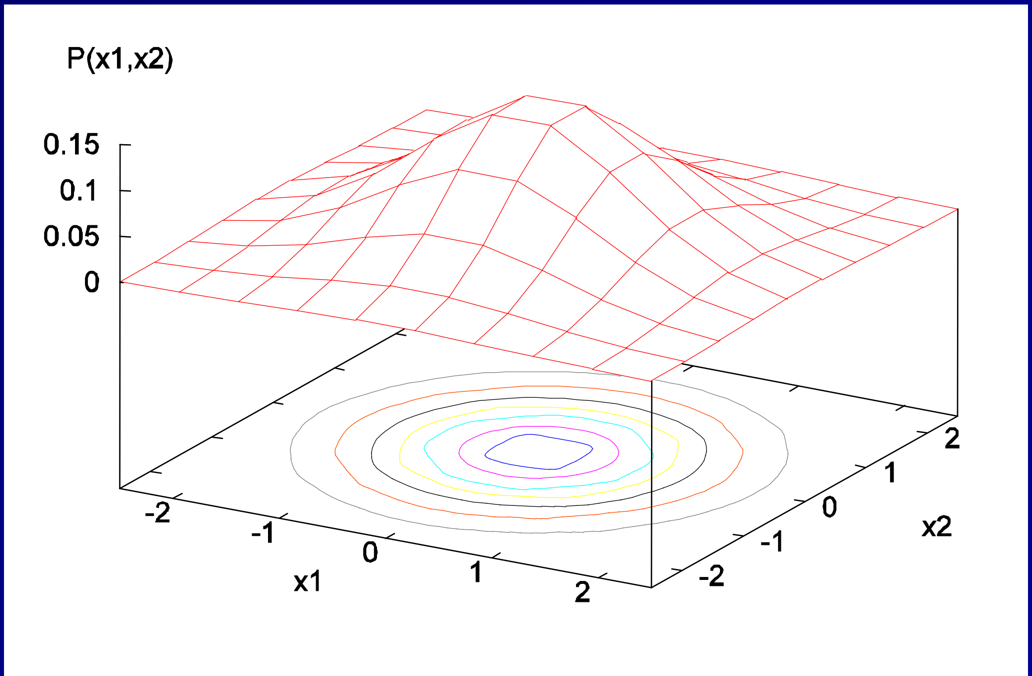
- A 2-dimensional Gaussian is defined by a mean vector $\mu = (\mu_1,\mu_2)$ and a covariance matrix

$$\Sigma = \begin{bmatrix} \sigma^2_{1,1} & \sigma^2_{1,2} \\ \sigma^2_{1,2} & \sigma^2_{2,2} \end{bmatrix}$$

- where $\sigma^2_{i,j} = E[(x_i - \mu_i)(x_j - \mu_j)]$ is the variance (if *i* = *j*) or co-variance (if $i \neq j$). Σ is symmetrical and positive-definite.

The Multivariate Gaussian (2)

- If Σ is the identity matrix $\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$ and $\mu = (0, 0)$, we get the standard normal distribution:

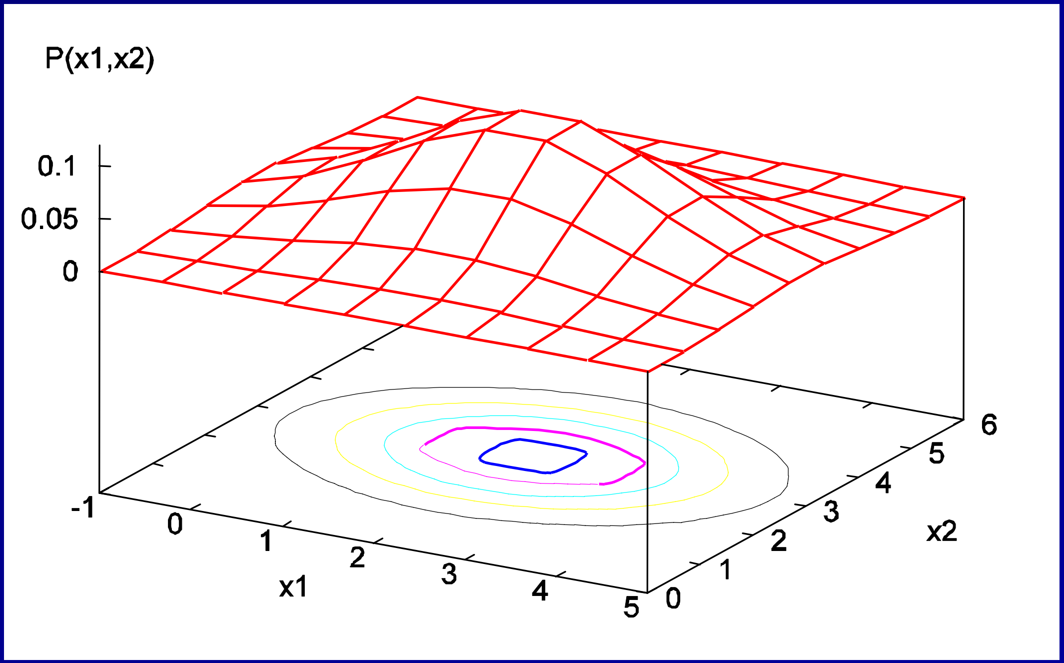


The Multivariate Gaussian (3)

- If Σ is a diagonal matrix, then x_1, and x_2 are independent random variables, and lines of equal probability are ellipses parallel to the coordinate axes. For example, when

$$\Sigma = \begin{bmatrix} 2 & 0 \\ 0 & 1 \end{bmatrix}$$ and

$$\mu = (2, 3)$$ we obtain

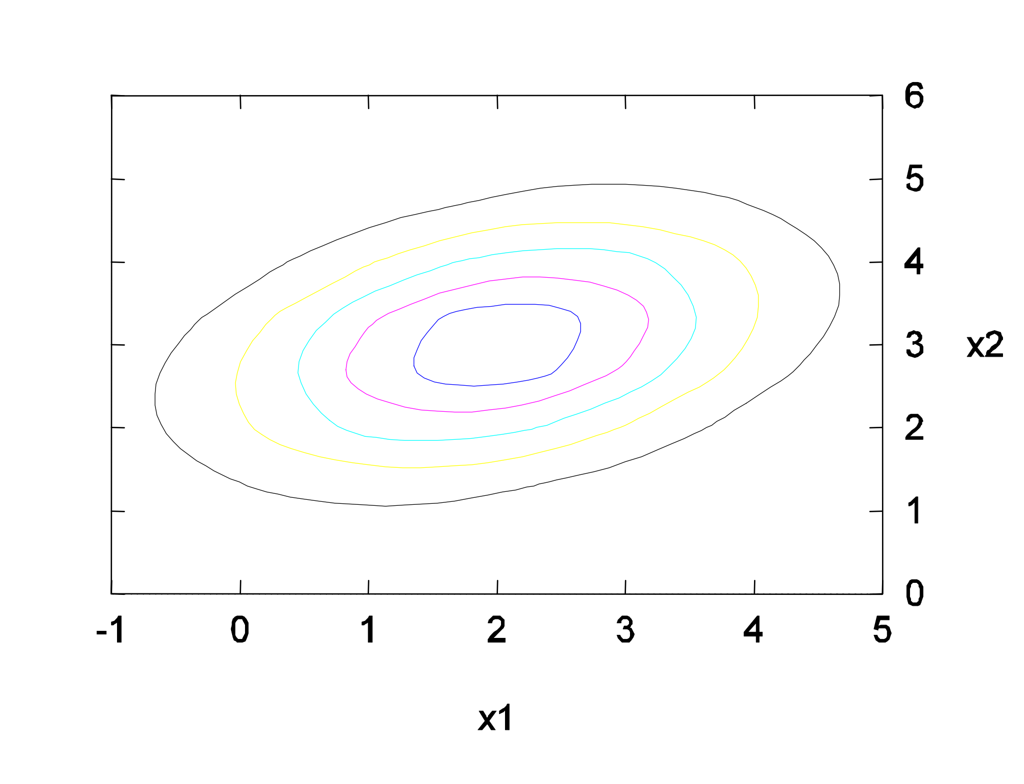


The Multivariate Gaussian (4)

- Finally, if Σ is an arbitrary matrix, then x_1 and x_2 are dependent, and lines of equal probability are ellipses tilted relative to the coordinate axes. For example, when

$$\Sigma = \begin{bmatrix} 2 & 0.5 \\ 0.5 & 1 \end{bmatrix} \text{ and}$$

$$\mu = (2, 3) \text{ we obtain}$$



Estimating a Multivariate Gaussian

- Given a set of N data points $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, we can compute the maximum likelihood estimate for the multivariate Gaussian distribution as follows:

$$\begin{aligned} \widehat{\mu} &= \frac{1}{N}\sum_i \mathbf{x}_i \\ \widehat{\Sigma} &= \frac{1}{N}\sum_i (\mathbf{x}_i - \widehat{\mu}) \cdot (\mathbf{x}_i - \widehat{\mu})^T \end{aligned}$$

- Note that the dot product in the second equation is an <u>outer product</u>. The outer product of two vectors is a matrix:

$$\mathbf{x} \cdot \mathbf{y}^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cdot [y_1 \; y_2 \; y_3] = \begin{bmatrix} x_1y_1 & x_1y_2 & x_1y_3 \\ x_2y_1 & x_2y_2 & x_2y_3 \\ x_3y_1 & x_3y_2 & x_3y_3 \end{bmatrix}$$

- For comparison, the usual dot product is written as $\mathbf{x}^T \cdot \mathbf{y}$

The LDA Model

- Linear discriminant analysis assumes that the joint distribution has the form

$$P(\mathbf{x}, y) = P(y)\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{x} - \mu_y]^T \Sigma^{-1} [\mathbf{x} - \mu_y]\right)$$

where each μ_y is the mean of a multivariate Gaussian for examples belonging to class *y* and Σ is a single covariance matrix <u>shared by all classes</u>.

Fitting the LDA Model

- It is easy to learn the LDA model in a single pass through the data:
  - Let $\widehat{\pi}_k$ be our estimate of P(*y* = *k*)
  - Let N_k be the number of training examples belonging to class *k*.

$$\widehat{\pi}_k = \frac{N_k}{N}$$

$$\widehat{\mu}_k = \frac{1}{N_k} \sum_{\{i : y_i = k\}} \mathbf{x}_i$$

$$\widehat{\Sigma} = \frac{1}{N} \sum_i (\mathbf{x}_i - \widehat{\mu}_{y_i}) \cdot (\mathbf{x}_i - \widehat{\mu}_{y_i})^T$$

- Note that each $\mathbf{x}_i$ is subtracted from its corresponding $\widehat{\mu}_{y_i}$ prior to taking the outer product. This gives us the "pooled" estimate of Σ

LDA learns an LTU

- Consider the 2-class case with a 0/1 loss function. Recall that

$$\begin{aligned} P(y=0|\mathbf{x}) &= \frac{P(\mathbf{x}, y=0)}{P(\mathbf{x}, y=0) + P(\mathbf{x}, y=1)} \\ P(y=1|\mathbf{x}) &= \frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0) + P(\mathbf{x}, y=1)} \end{aligned}$$

- Also recall from our derivation of the Logistic Regression classifier that we should classify into class ŷ = 1 if

$$\log \frac{P(y=1|\mathbf{x})}{P(y=0|\mathbf{x})} > 0$$

- Hence, for LDA, we should classify into ŷ = 1 if

$$\log \frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} > 0$$

because the denominators cancel

LDA learns an LTU (2)

$$P(\mathbf{x}, y) = P(y)\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{x}-\mu_y]^T\Sigma^{-1}[\mathbf{x}-\mu_y]\right)$$

$$\frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \frac{P(y=1)\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{x}-\mu_1]^T\Sigma^{-1}[\mathbf{x}-\mu_1]\right)}{P(y=0)\frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{x}-\mu_0]^T\Sigma^{-1}[\mathbf{x}-\mu_0]\right)}$$

$$\frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \frac{P(y=1)\ \exp\left(-\frac{1}{2}[\mathbf{x}-\mu_1]^T\Sigma^{-1}[\mathbf{x}-\mu_1]\right)}{P(y=0)\ \exp\left(-\frac{1}{2}[\mathbf{x}-\mu_0]^T\Sigma^{-1}[\mathbf{x}-\mu_0]\right)}$$

$$\log\frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \log\frac{P(y=1)}{P(y=0)} - \frac{1}{2}\left([\mathbf{x}-\mu_1]^T\Sigma^{-1}[\mathbf{x}-\mu_1] - [\mathbf{x}-\mu_0]^T\Sigma^{-1}[\mathbf{x}-\mu_0]\right)$$

LDA learns an LTU (3)

- Let's focus on the term in brackets:

$$\left([\mathbf{x}-\mu_1]^T\Sigma^{-1}[\mathbf{x}-\mu_1] - [\mathbf{x}-\mu_0]^T\Sigma^{-1}[\mathbf{x}-\mu_0]\right)$$

- Expand the quadratic forms as follows:

$$[\mathbf{x}-\mu_1]^T\Sigma^{-1}[\mathbf{x}-\mu_1] \;=\; \mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\mu_1 - \mu_1^T\Sigma^{-1}\mathbf{x} + \mu_1^T\Sigma^{-1}\mu_1$$

$$[\mathbf{x}-\mu_0]^T\Sigma^{-1}[\mathbf{x}-\mu_0] \;=\; \mathbf{x}^T\Sigma^{-1}\mathbf{x} - \mathbf{x}^T\Sigma^{-1}\mu_0 - \mu_0^T\Sigma^{-1}\mathbf{x} + \mu_0^T\Sigma^{-1}\mu_0$$

- Subtract the lower from the upper line and collect similar terms. Note that the quadratic terms cancel! This leaves only terms linear in **x**.

$$\mathbf{x}^T\Sigma^{-1}(\mu_0-\mu_1)+(\mu_0-\mu_1)\Sigma^{-1}\mathbf{x}+\mu_1^T\Sigma^{-1}\mu_1-\mu_0^T\Sigma^{-1}\mu_0$$

LDA learns an LTU (4)

$$\mathbf{x}^T\Sigma^{-1}(\mu_0-\mu_1)+(\mu_0-\mu_1)\Sigma^{-1}\mathbf{x}+\mu_1^T\Sigma^{-1}\mu_1-\mu_0^T\Sigma^{-1}\mu_0$$

- Note that since Σ^{-1} is symmetric $\mathbf{a}^T\Sigma^{-1}\mathbf{b} = \mathbf{b}^T\Sigma^{-1}\mathbf{a}$ for any two vectors **a** and **b**. Hence, the first two terms can be combined to give

$$2\mathbf{x}^T\Sigma^{-1}(\mu_0 - \mu_1) + \mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0.$$

- Now plug this back in…

$$\log\frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \log\frac{P(y=1)}{P(y=0)} - \frac{1}{2}\left[2\mathbf{x}^T\Sigma^{-1}(\mu_0 - \mu_1) + \mu_1^T\Sigma^{-1}\mu_1 - \mu_0^T\Sigma^{-1}\mu_0\right]$$

$$\log\frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \log\frac{P(y=1)}{P(y=0)} + \mathbf{x}^T\Sigma^{-1}(\mu_1 - \mu_0) - \frac{1}{2}\mu_1^T\Sigma^{-1}\mu_1 + \frac{1}{2}\mu_0^T\Sigma^{-1}\mu_0$$

LDA learns an LTU (5)

$$\log \frac{P(\mathbf{x}, y=1)}{P(\mathbf{x}, y=0)} = \log \frac{P(y=1)}{P(y=0)} + \mathbf{x}^T \Sigma^{-1}(\mu_1 - \mu_0) - \frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_0^T \Sigma^{-1} \mu_0$$

Let

$$\begin{aligned} \mathbf{w} &= \Sigma^{-1}(\mu_1 - \mu_0) \\ c &= \log \frac{P(y=1)}{P(y=0)} - \frac{1}{2}\mu_1^T \Sigma^{-1} \mu_1 + \frac{1}{2}\mu_0^T \Sigma^{-1} \mu_0 \end{aligned}$$

Then we will classify into class $\hat{y} = 1$ if

$$\mathbf{w} \cdot \mathbf{x} + c > 0.$$

This is an LTU.

Two Geometric Views of LDA
View 1: Mahalanobis Distance

- The quantity $D_M(\mathbf{x},\mathbf{u})^2 = (\mathbf{x}-\mathbf{u})^T\Sigma^{-1}(\mathbf{x}-\mathbf{u})$ is known as the (squared) Mahalanobis distance between **x** and **u**. We can think of the matrix Σ^{-1} as a linear distortion of the coordinate system that converts the standard Euclidean distance into the Mahalanobis distance
- Note that

$$\log P(\mathbf{x}|y=k) \;\propto\; \log \pi_k - \frac{1}{2}[(\mathbf{x}-\mu_k)^T\Sigma^{-1}(\mathbf{x}-\mu_k)]$$

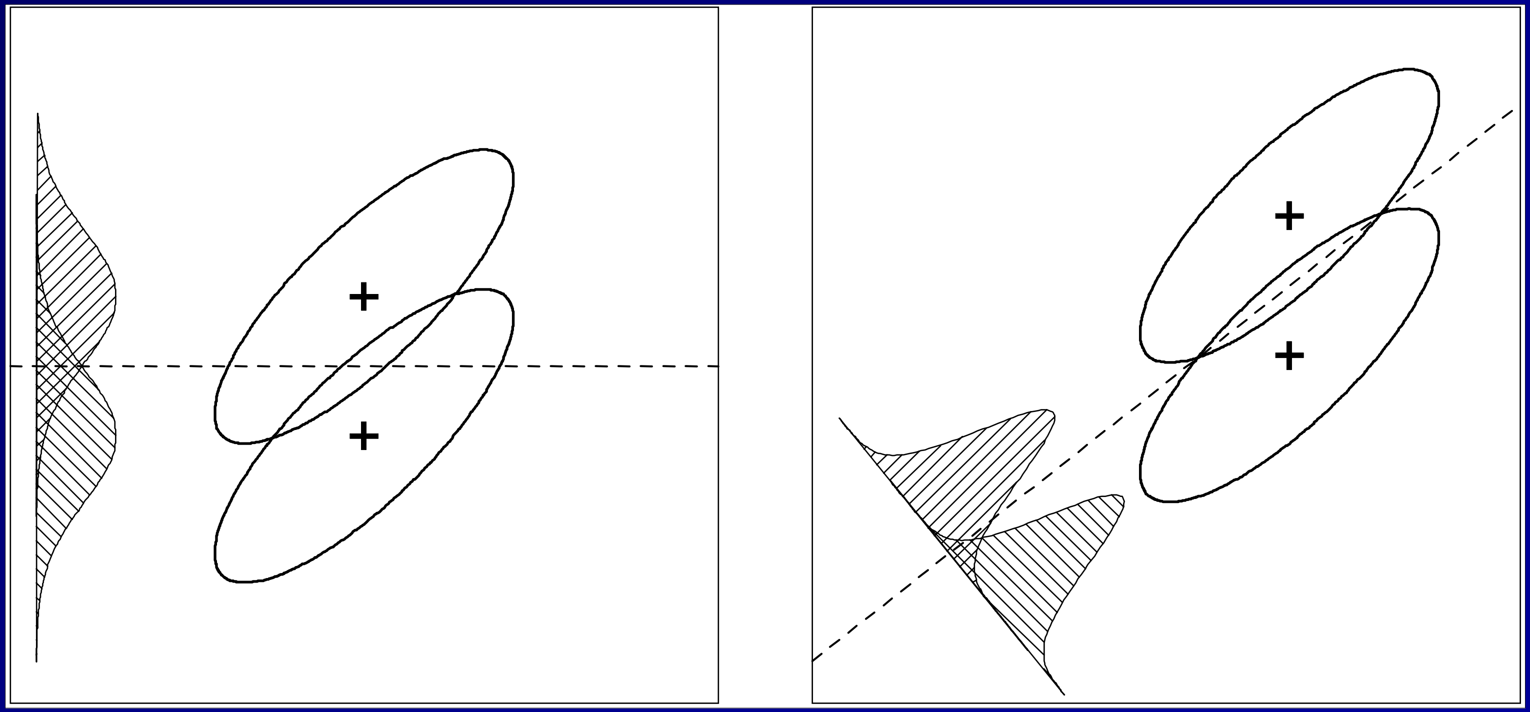
$$\log P(\mathbf{x}|y=k) \;\propto\; \log \pi_k - \frac{1}{2}D_M(\mathbf{x},\mu_k)^2$$

- Therefore, we can view LDA as computing
  - $D_M(\mathbf{x},\mu_0)^2$ and $D_M(\mathbf{x},\mu_1)^2$

  and then classifying **x** according to which mean μ_0 or μ_1 is closest in Mahalanobis distance (corrected by log π_k)

View 2: Most Informative Low-Dimensional Projection

- LDA can also be viewed as finding a hyperplane of dimension K – 1 such that **x** and the {μ_k} are projected down into this hyperplane and then **x** is classified to the nearest μ_k using Euclidean distance inside this hyperplane



Generalizations of LDA

- General Gaussian Classifier
  - Instead of assuming that all classes share the same Σ, we can allow each class *k* to have its own Σ_k. In this case, the resulting classifier will be a quadratic threshold unit (instead of an LTU)
- Naïve Gaussian Classifier
  - Allow each class to have its own Σ_k, but require that each Σ_k be diagonal. This means that *within* each class, any pair of features x_{j1} and x_{j2} will be assumed to be statistically independent. The resulting classifier is still a quadratic threshold unit (but with a restricted form)

Summary of Linear Discriminant Analysis

- Learns the joint probability distribution P(**x**, *y*).
- Direct Computation. The maximum likelihood estimate of P(**x**,*y*) can be computed from the data without search. However, inverting the Σ matrix requires $O(n^3)$ time.
- Eager. The classifier is constructed from the training examples. The examples can then be discarded.
- Batch. Only a batch algorithm is available. An online algorithm could be constructed if there is an online algorithm for incrementally updated Σ^{-1}. [This is easy for the case where Σ is diagonal.]