

Machine Learning (CS 567) Lecture 5

Fall 2008

Time: T-Th 5:00pm - 6:20pm

Location: GFS 118

Instructor: Sofus A. Macskassy (macskass@usc.edu)

Office: SAL 216

Office hours: by appointment

Teaching assistant: Cheol Han (cheolhan@usc.edu)

Office: SAL 229

Office hours: M 2-3pm, W 11-12

Class web page:

<http://www-scf.usc.edu/~csci567/index.html>

Administrative: Regrading

- This regrading policy holds for quizzes, homeworks, midterm and final exam.
- Grading complaint policy: if students have problems at assignment grading, feel free to talk to the instructor/TA for it ***within one week*** of getting the graded work back. After one week ***no regrading*** will be done.
- Even if students only request re-grading one of the answers, all this assignment will be checked and graded again. (take the risk of possible losing total points.)

Lecture 5 Outline

- Logistic Regression (from last week)
- Linear Discriminant Analysis
- Off-the-shelf Classifiers

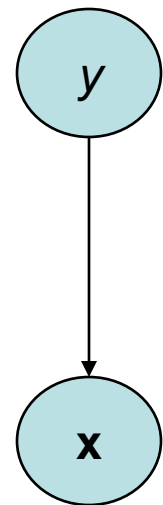
Linear Threshold Units

$$h(\mathbf{x}) = \begin{cases} +1 & \text{if } w_1x_1 + \dots + w_nx_n \geq w_0 \\ -1 & \text{otherwise} \end{cases}$$

- We assume that each feature x_j and each weight w_j is a real number (we will relax this later)
- We will study three different algorithms for learning linear threshold units:
 - Perceptron: classifier
 - Logistic Regression: conditional distribution
 - Linear Discriminant Analysis: joint distribution

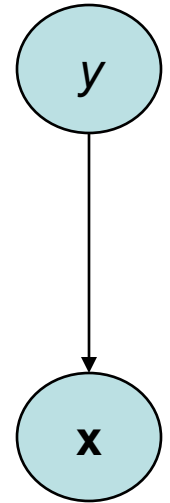
Linear Discriminant Analysis

- Learn $P(\mathbf{x}, y)$. This is sometimes called the generative approach, because we can think of $P(\mathbf{x}, y)$ as a model of how the data is generated.
 - For example, if we factor the joint distribution into the form
$$P(\mathbf{x}, y) = P(y) P(\mathbf{x} | y)$$
 - we can think of $P(y)$ as “generating” a value for y according to $P(y)$. Then we can think of $P(\mathbf{x} | y)$ as generating a value for \mathbf{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 $P(\mathbf{x} \mid y)$ is a multivariate normal distribution with mean μ_k and covariance matrix Σ



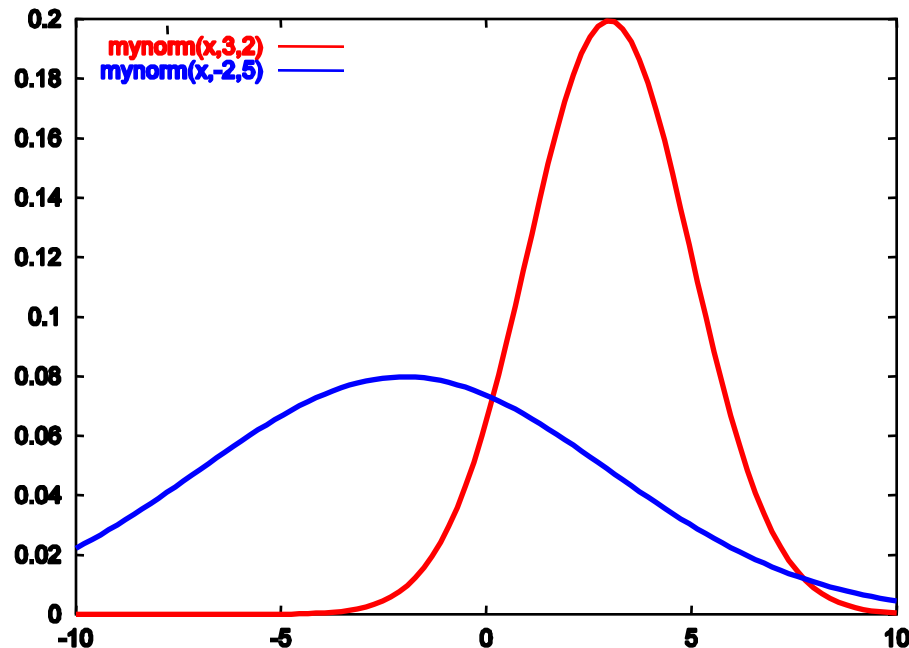
$$P(\mathbf{x}|y = k) = \frac{1}{(2\pi)^{n/2} |\Sigma|^{1/2}} \exp \left(-\frac{1}{2} [\mathbf{x} - \mu_k]^T \Sigma^{-1} [\mathbf{x} - \mu_k] \right)$$

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

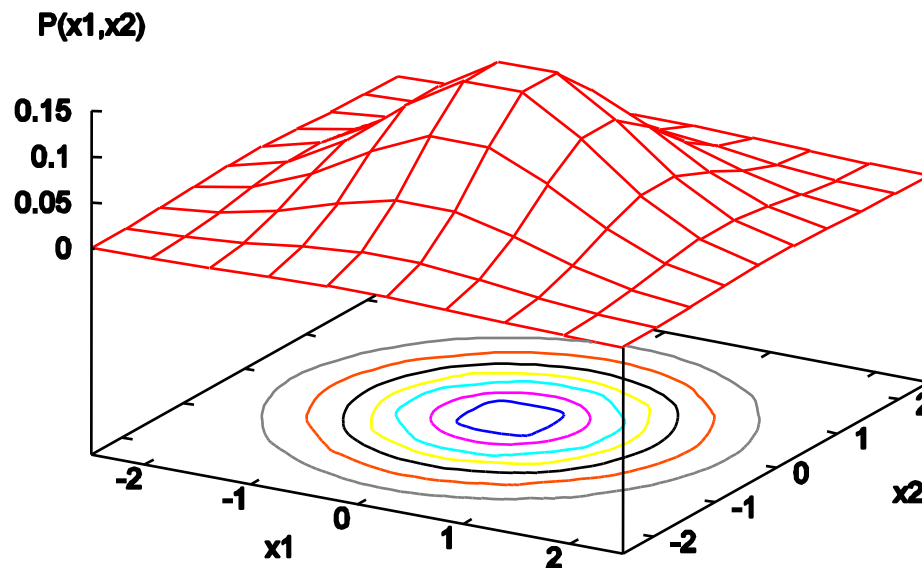
$$\Sigma = \begin{bmatrix} \sigma_{1,1}^2 & \sigma_{1,2}^2 \\ \sigma_{1,2}^2 & \sigma_{2,2}^2 \end{bmatrix}$$

- where $\sigma_{i,j}^2 = E[(x_i - \mu_i)(x_j - \mu_j)]$ is the variance (if $i = j$) or co-variance (if $i \neq j$).

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:

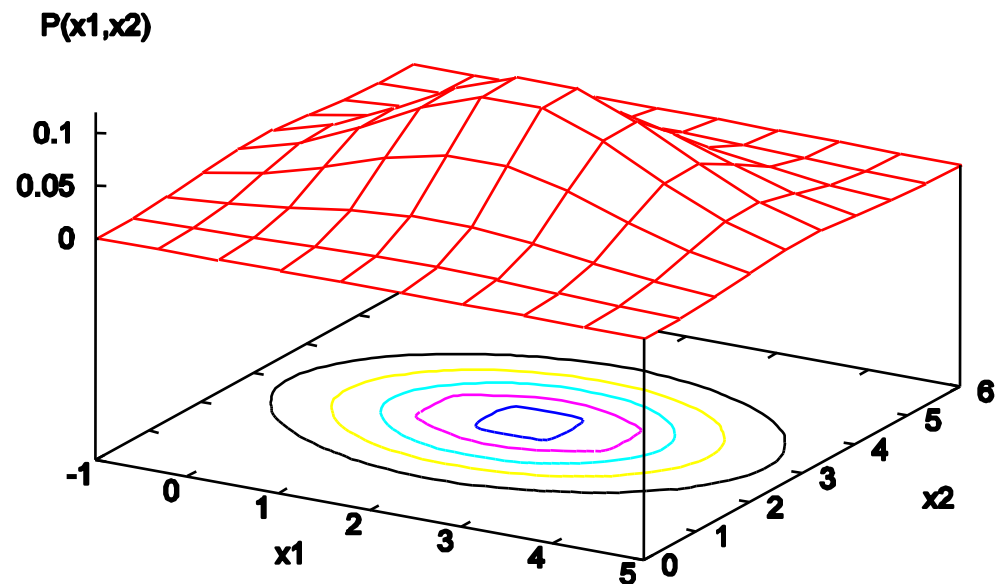


The Multivariate Gaussian (3)

- 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

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

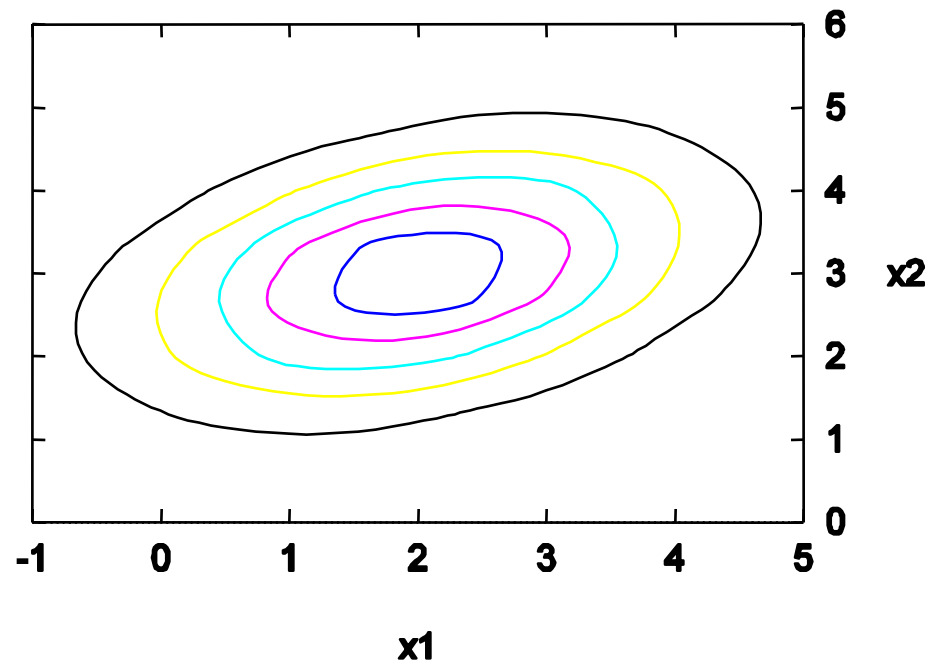
$$\mu = (2, 3) \quad \text{we obtain}$$



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 $\{\mathbf{x}_1, \dots, \mathbf{x}_N\}$, we can compute the maximum likelihood estimate for the multivariate Gaussian distribution as follows:

$$\hat{\boldsymbol{\mu}} = \frac{1}{N} \sum_i \mathbf{x}_i$$

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

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

$$\mathbf{x} \cdot \mathbf{y}^T = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \cdot [y_1 \ y_2 \ y_3] = \begin{bmatrix} x_1 y_1 & x_1 y_2 & x_1 y_3 \\ x_2 y_1 & x_2 y_2 & x_2 y_3 \\ x_3 y_1 & x_3 y_2 & x_3 y_3 \end{bmatrix}$$

- 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

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

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

Fitting the LDA Model

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

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

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

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

- Note that each \mathbf{x}_i is subtracted from its corresponding $\hat{\mu}_{y_i}$ 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

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

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

- Also recall from our derivation of the Logistic Regression classifier that we should classify into class $\hat{y} = 1$ if

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

- Hence, for LDA, we should classify into $\hat{y} = 1$ 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 \mathbf{x} .

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

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 \mathbf{a} and \mathbf{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

$$\mathbf{w} = \Sigma^{-1} (\mu_1 - \mu_0)$$

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

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 \mathbf{x} and \mathbf{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

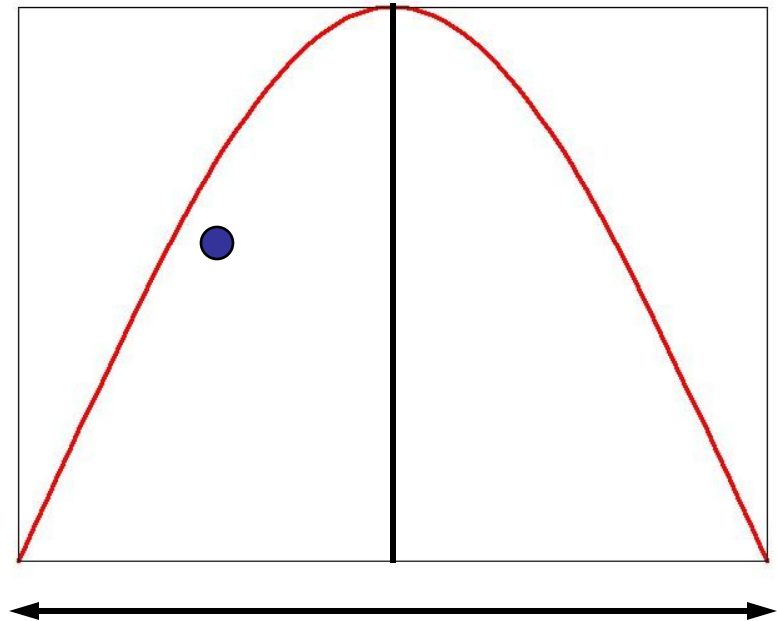
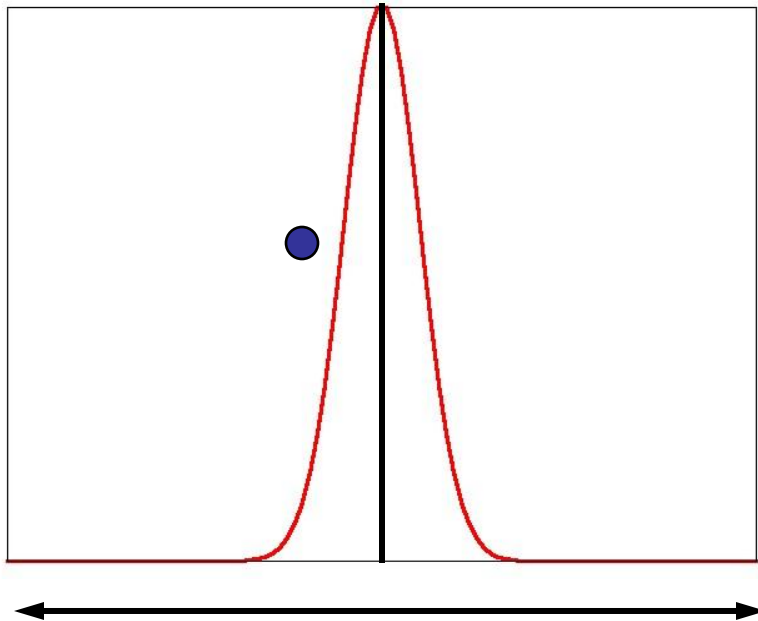
$$\log P(\mathbf{x}|y = k) \propto \log \pi_k - \frac{1}{2} [(\mathbf{x} - \mu_k)^T \Sigma^{-1} (\mathbf{x} - \mu_k)]$$

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

- 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 \mathbf{x} according to which mean μ_0 or μ_1 is closest in Mahalanobis distance (corrected by $\log \pi_k$)

Mahalanobis Distance

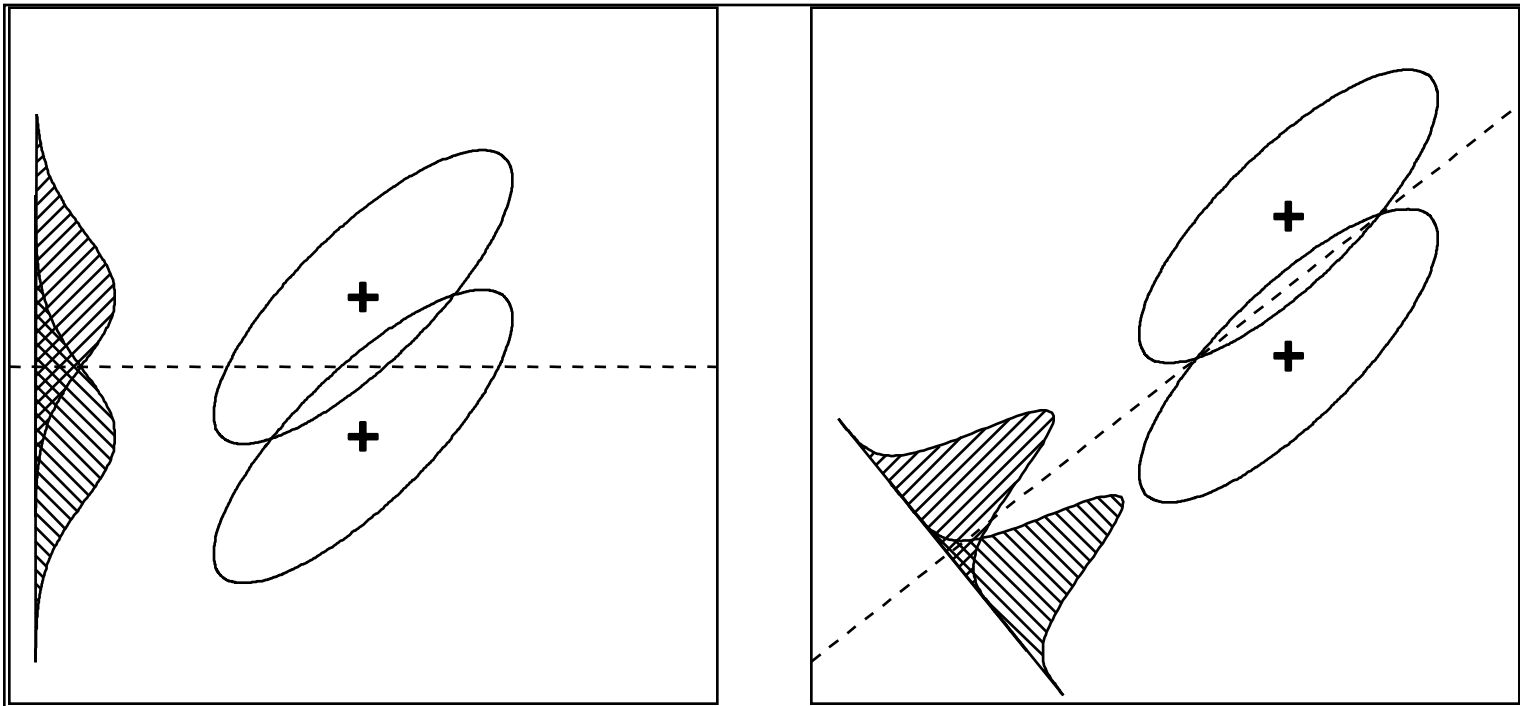
- Which point is closer to the vertical line?



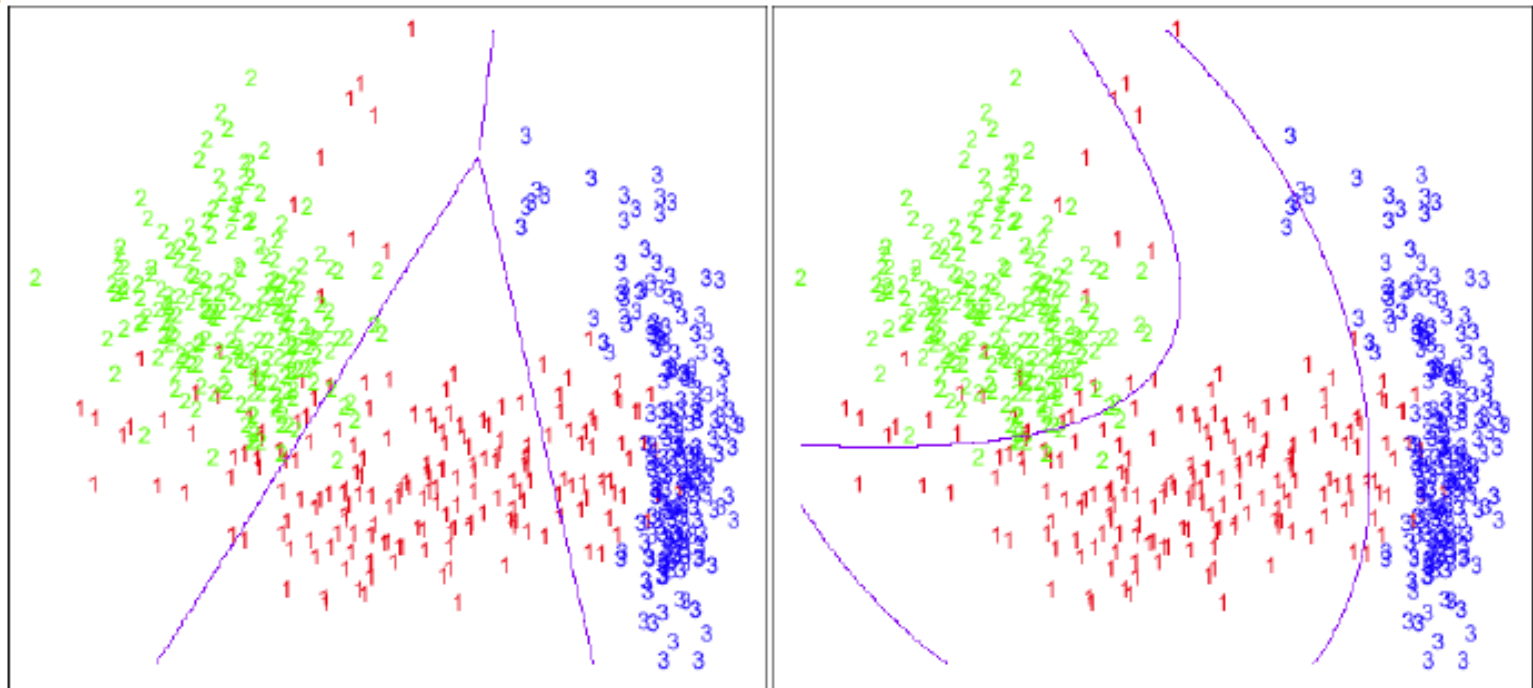
- Weak point of mahalanobis: if only few points observed, then the variance is not meaningful.

View 2: Most Informative Low-Dimensional Projection

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



Nonlinear boundaries with LDA



Left: Linear decision boundary found by LDA

Right: Quadratic decision boundary found by LDA. This was obtained by finding linear decision boundaries in the five dimensional space $X_1, X_2, X_1 * X_2, X_1^2, X_2^2$. ***Linear inequalities in this space are quadratic inequalities in the original space.***

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_{j_1} and x_{j_2} will be assumed to be statistically independent. The resulting classifier is still a quadratic threshold unit (but with a restricted form)

Quadratic Discriminant Analysis (QDA)

Quadratic Discriminant Analysis (QDA) is a generalization of LDA where different classes are allowed to have different covariance matrices. i.e.,

$\Sigma_k \neq \Sigma_l$ for classes $k \neq l$.

$$\hat{\Sigma}_k = \frac{1}{N_k} \sum_{i, y_i=k} (\mathbf{x}_i - \hat{\mu}_k) \cdot (\mathbf{x}_i - \hat{\mu}_k)^T$$

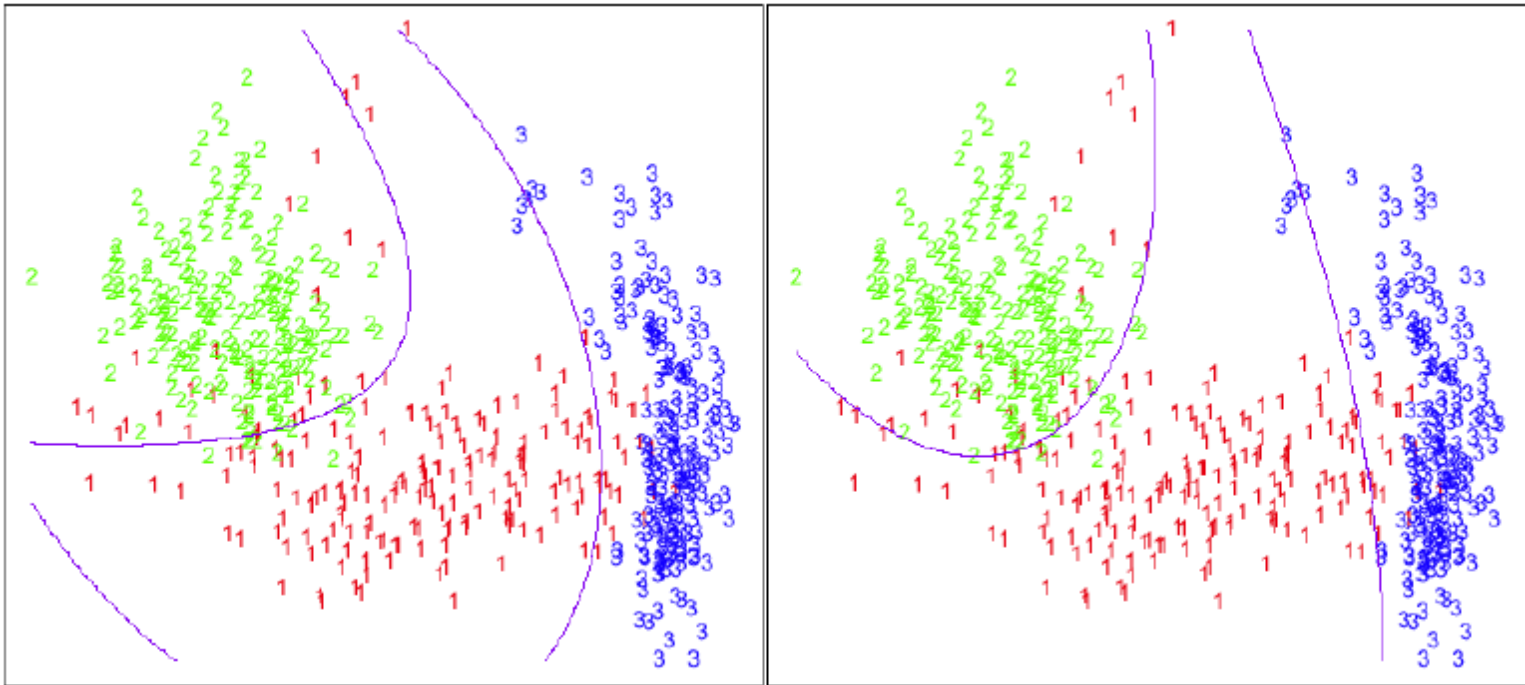
$$P(\mathbf{x}, y = k) = P(y = k) \frac{1}{(2\pi)^{n/2} |\Sigma_k|^{1/2}} \exp\left(-\frac{1}{2}[\mathbf{x} - \mu_k]^T \Sigma_k^{-1} [\mathbf{x} - \mu_k]\right)$$

We can solve for

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

in a similar way as before. Note, however, that the Σ_0 and Σ_1 determinants do not cancel.

LDA vs. QDA



Left: Quadratic fit by LDA in transformed space.

Right: Quadratic fit by QDA.

The differences in the fit are very small.

Generalizations of LDA

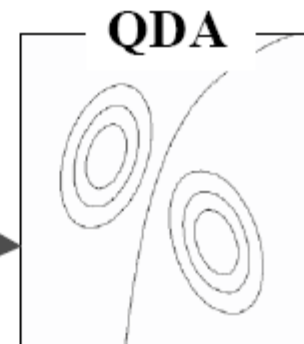
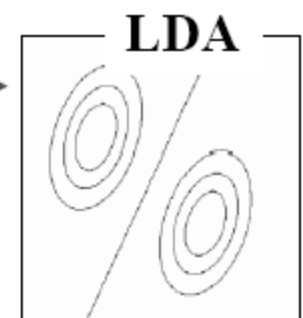
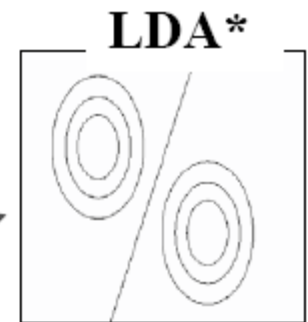
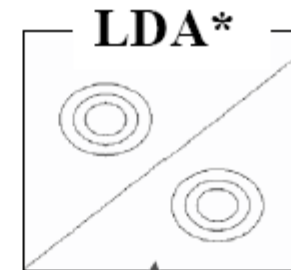
- General Gaussian Classifier
 - Instead of assuming that all classes share the same Σ , we can allow each class k to have its own Σ_k . In this case, the resulting classifier will be a quadratic threshold unit (instead of an LTU)
- Naïve Gaussian Classifier
 - Allow each class to have its own Σ_k , but require that each Σ_k be diagonal. This means that *within* each class, any pair of features x_{j_1} and x_{j_2} will be assumed to be statistically independent. The resulting classifier is still a quadratic threshold unit (but with a restricted form)

Different Discriminant Functions

Here we compare the Discriminant Functions created by various Gaussian Density assumptions

Covariance	Covariance Matrix Type	Covariance Matrix	Number of parameters
Shared	Hyperspheric	$\Sigma_k = \Sigma = \sigma^2 \mathbf{I}$	1
Shared	Axis-aligned	$\Sigma_k = \Sigma = \text{diag}(d_{ii})$	p
Shared	Hyper ellipsoidal	$\Sigma_k = \Sigma$	$p(p+1)/2$
Different	Hyper ellipsoidal	Σ_k	$K^*(p(p+1)/2)$

Here, p is the dimensionality of the input data and K is the number of classes.



Summary of Linear Discriminant Analysis

- Learns the joint probability distribution $P(\mathbf{x}, y)$.
- Direct Computation. The maximum likelihood estimate of $P(\mathbf{x}, y)$ 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.]

Comparing Perceptron, Logistic Regression, and LDA

- How should we choose among these three algorithms?
- There is a big debate within the machine learning community!

Issues in the Debate

- Statistical Efficiency. If the generative model $P(\mathbf{x}, y)$ is correct, then LDA usually gives the highest accuracy, particularly when the amount of training data is small. If the model is correct, LDA requires 30% less data than Logistic Regression in theory
- Computational Efficiency. Generative models typically are the easiest to learn. In our example, LDA can be computed directly from the data without using gradient descent.

Issues in the Debate

- Robustness to changing loss functions. Both generative and conditional probability models allow the loss function to be changed at run time without re-learning. Perceptron requires re-training the classifier when the loss function changes.
- Robustness to model assumptions. The generative model usually performs poorly when the assumptions are violated. For example, if $P(\mathbf{x} | y)$ is very non-Gaussian, then LDA won't work well. Logistic Regression is more robust to model assumptions, and Perceptron is even more robust.
- Robustness to missing values and noise. In many applications, some of the features x_{ij} may be missing or corrupted in some of the training examples. Generative models typically provide better ways of handling this than non-generative models.