syde675 pattern recognition

Bayesian Decision

J.Zelek 2022

Multivariate Gaussian/NormalDensity

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)} \tag{1}$$

The constant term $\frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}}$ ensures that $f_k(x)$ integrates to 1. For additional intuition, plug in $p = 1$ and see that it reduces to the familiar *univariate* (or 1D) Gaussian density, parameterized by a mean μ and variance σ^2. The equation (1) will come down to:

$$f(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{2}$$

Multivariate Gaussian/NormalDensity

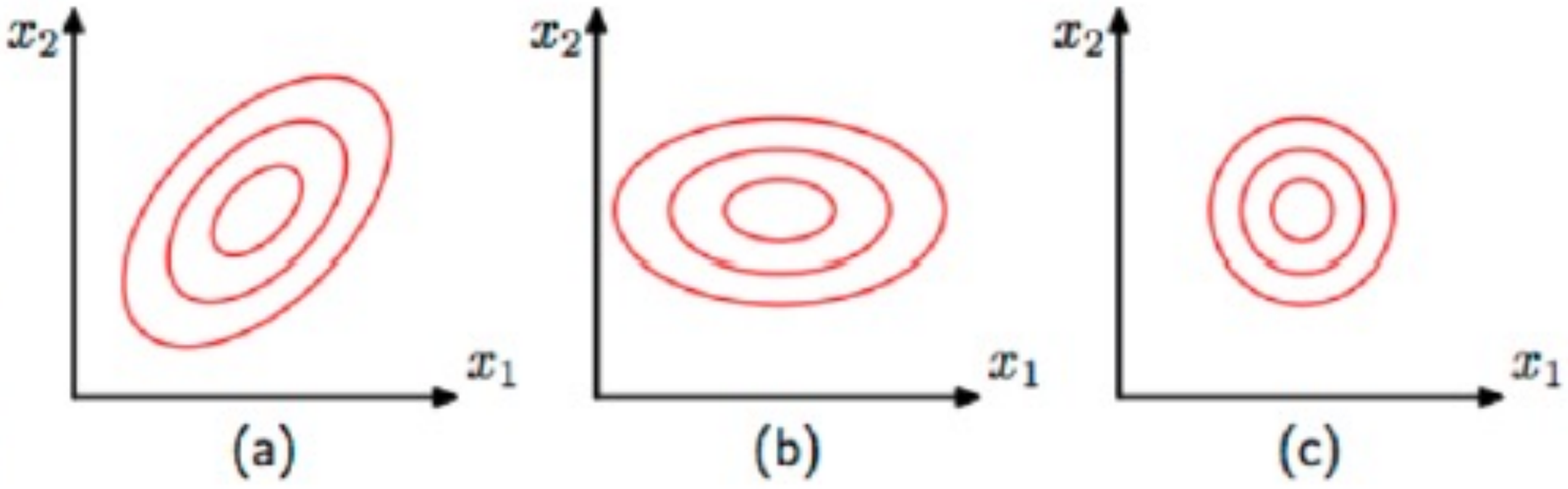
In particular, we often use multivariate normals as *class conditional densities*, e.g., to describe the height and weight of a pet given that it is a dog. We say "X conditioned on class k is normally distributed" and write:

$$X|G = k \sim \mathcal{N}(\mu_k, \Sigma_k) \tag{3}$$

In this expression, μ_k and Σ_k represent the *mean vector* and *covariance matrix* of the density for the kth class, respectively, and p is the number of dimensions of the feature vector x, as usual. The covariance matrix is sometimes called the variance-covariance matrix since the variances of the individual variables are on the diagonal and covariances are off-diagonal.

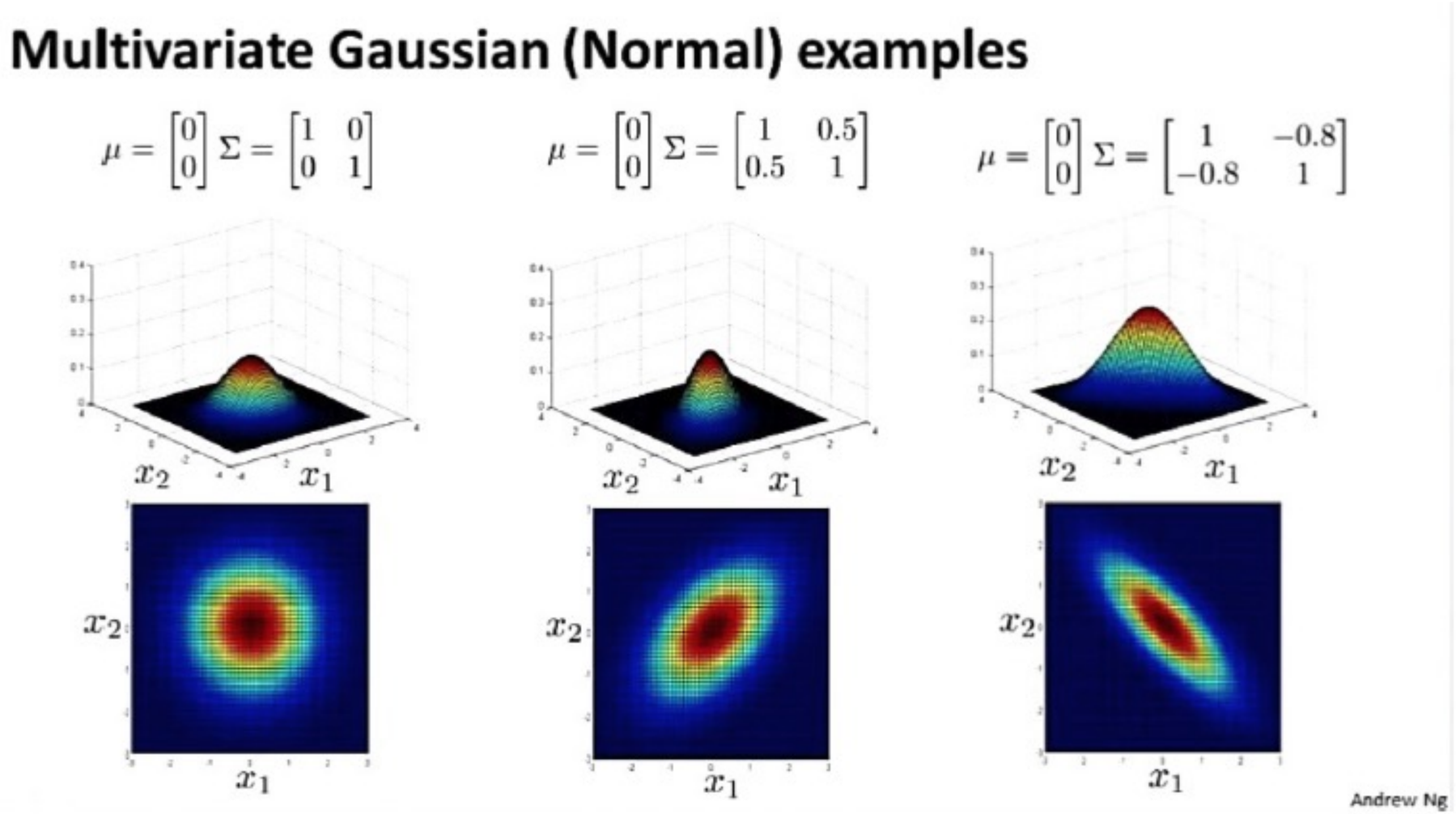
Multivariate Gaussian/NormalDensity

Figure 2.8 Contours of constant probability density for a Gaussian distribution in two dimensions in which the covariance matrix is (a) of general form, (b) diagonal, in which the elliptical contours are aligned with the coordinate axes, and (c) proportional to the identity matrix, in which the contours are concentric circles.



The level sets of a bivariate Gaussian are ellipses

Multivariate Gaussian/NormalDensity



Bivariate Gaussian as surface plots & heatmaps. If the covariance matrix is represented by an identity matrix, the contour plots of pdf will be concentric circles. In any other case, looking @ covariance matrices doesn't make any sense, you can't predict it based on the values. Eigenvalues & eigenvectors should be used instead.

Multivariate Gaussian/NormalDensity

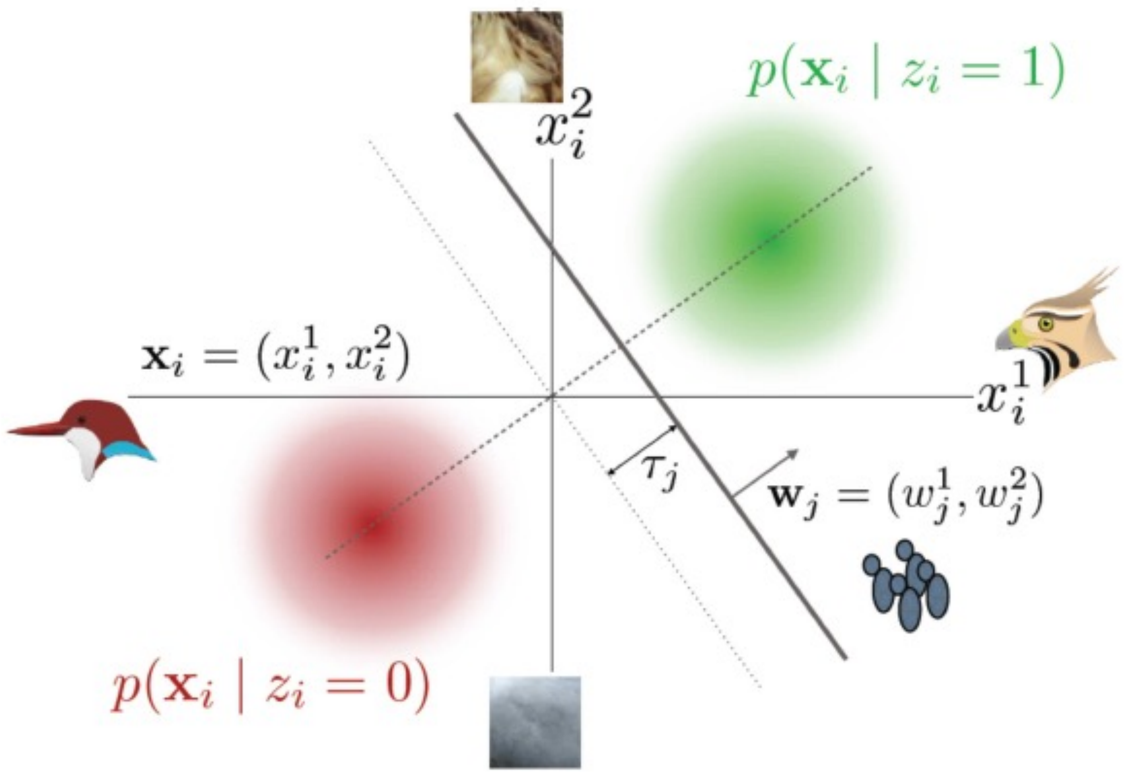


Figure shows an example of bivariate Gaussians used to model features of two birds (sandpiper vs. hawk). In this example, we have two class conditional densities with different means but equal covariance matrices, and the features capture characteristics of the beak and color of the plumage. The figure also foreshadows how we can use these densities to define a decision boundary.

Multivariate Gaussian/NormalDensity

$$f_k(x) = \frac{1}{(2\pi)^{p/2}|\Sigma_k|^{1/2}} e^{-\frac{1}{2}(x-\mu_k)^\top \Sigma_k^{-1}(x-\mu_k)} \tag{1}$$

Look at the exponent in Equation 1. Assuming $\mu_k = 0$, which we refer to as the *mean zero* or *centered* case, $(x - \mu_k)^\top \Sigma_k^{-1}(x - \mu_k)$ has the form $f(x) = x^\top Qx$, which is known in linear algebra as a *quadratic form*. Looking at the dimensions of these terms, the exponent resolves to a scalar. The entries in the matrix Q control the shape of the function $f(x)$, as illustrated in Figure 1. An ellipse in that figure represents the locus of values x such that $f(x) = x^\top Qx =$ some constant. In the case of Equation 1, the covariance matrix Σ_k^{-1} is *positive definite*, a property that we will discuss later. People often think of covariance matrices as ellipses: perfect circles mean no correlation.

Linear Discriminant Analysis

Linear Discriminant Analysis[1] is a supervised classification method that assumes the class conditional densities are Gaussian and that all of the covariance matrices are equal, i.e., $\Sigma_k = \Sigma$. It can also be used for dimensionality reduction, which takes in very large feature vectors and collapses them into a small number of meaningful dimensions. To see how it works, let π_k denote the prior probability of class k (which means $\sum_{k=1}^{K} \pi_k = 1$) and apply Bayes' theorem to express the posterior probability as follows:

$$Pr(G = k|X = x) = \frac{f_k(x)\pi_k}{\sum_{l=1}^{K} f_l(x)\pi_l}$$

Linear Discriminant Analysis

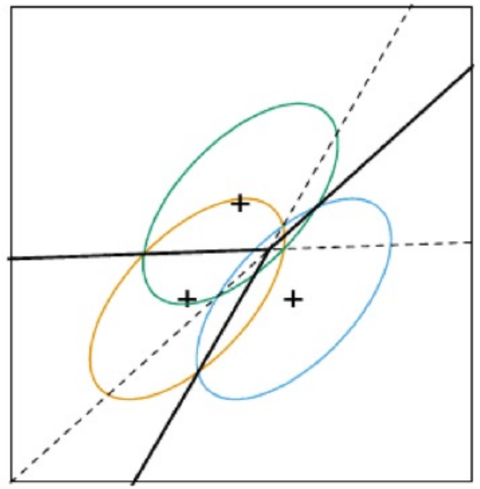
Recall that $f_k(x)$ is the class-conditional density and the denominator is the normalization factor or marginal likelihood. Bayes' rule allows us to take the class conditional density and turn it around so we can predict the class given our data. Assume for now that π_k, μ_k and Σ_k are all known. Now look at the log-odds we get by comparing two classes k and l:

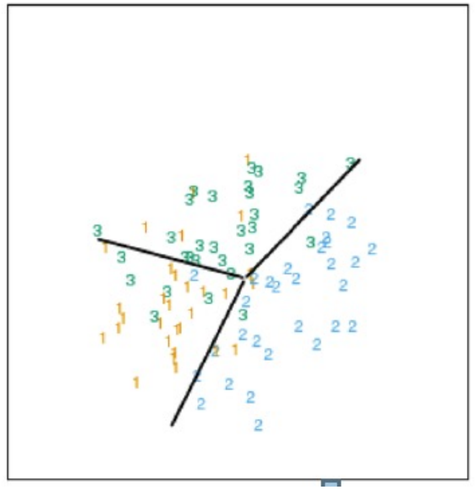
$$\log \frac{Pr(G = k|X = x)}{Pr(G = l|X = x)} = \log \frac{f_k(x)}{f_l(x)} + \log \frac{\pi_k}{\pi_l}$$

Notice how the log-odds on the priors appears as a constant offset or shift. Assuming Gaussian class conditional densities, we get

$$\log \frac{Pr(G = k|X = x)}{Pr(G = l|X = x)} = \log \frac{\pi_k}{\pi_l} - \frac{1}{2}(\mu_k + \mu_l)^\top \Sigma^{-1}(\mu_k - \mu_l) + x^\top \Sigma^{-1}(\mu_k - \mu_l)$$

Linear Discriminant Analysis





Note that the normalizations constants canceled out and, since all the Σ_ks are equal, we don't have any terms quadratic in x. In other words, x only appears linearly in this expression, which has the form $\alpha_{k0} + \alpha_k^\top x$.

The locus where $Pr(G = k|X = x) = Pr(G = l|X = x)$ is a line (plane, hyperplane, etc.) and this is the *decision boundary*: on one side, class k is more likely than class l, vice versa on the other side.

Figure [redacted] shows the decision boundaries between three classes in 2 dimensions. If the covariances weren't equal, the decision boundaries wouldn't be straight.

Linear Discriminant Function

Formally speaking, the *linear discriminant functions* have the form

$$\delta_k(x) = x^\top \Sigma^{-1} \mu_k - \frac{1}{2} \mu_k^\top \Sigma^{-1} \mu_k + \log \pi_k$$

with decision rule:

$$G(x) = \arg\max_k \delta_k(x)$$

Assuming we've already calculated the mean and covariance and we've estimated the prior, to use this as a classifier we take an observation x, plug this into discriminant function for each class and see which one yields the maximum discriminant function value.

If the covariance matrices are spherical (i.e., $\Sigma \propto I$) and the priors are equal (e.g., sandpipers just as likely as hawks at a campsite), the decision boundary is simply the perpendicular bisector of the line joining the two means. Changing the prior (or class balance) has the effect of shifting the decision boundary along that line. In the case of non-spherical covariance matrices, as in Fig. [redacted] the decision boundary still bisects the line joining the means but is no longer perpendicular.

Linear Discriminant Function – Parameter Estimation

We've been assuming that the priors, mean and covariance were estimated for us thus far. Here's how you estimate them.

Priors:

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

where N_k is the number of class k observations and N is the total number of observations.

Sample mean:

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{g_i = k} x_i$$

Here the sum is over the feature vectors belonging to class k.

Sample covariance:

$$\hat{\Sigma} = \frac{1}{N-K} \sum_{k=1}^{K} \sum_{g_i = k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^\top$$

Covariance matrices are also known as *centered second moment* matrices. The centering operation refers to subtracting the class mean. The second moments refer to the sum of the outer products of the feature vectors with themselves. You can think of this as the vector counterpart to squaring for scalars. Each term in that sum is a $p \times p$ matrix containing all the possible products between pairs of entries in a feature vector. The outer summation is over all K classes; we can do this *pooling* operation because we assumed that all classes have same covariance matrix.

Linear Discriminant Function – compare w Logistic Regression

The log posterior odds in Linear Discriminant Analysis has the form

$$\alpha_{k0} + \alpha_k^\top x$$

which looks an awful lot like

$$\beta_{k0} + \beta_k^\top x$$

from logistic regression. The difference lies in how the coefficients are estimated. The bottom line is that the results of using the two approaches are often very similar. LR is a discriminative classifier. LR has the advantage of making fewer assumptions about how the data are distributed. Linear Discriminant Analysis is a generative classifier, and it has the advantage of offering low dimensional projections of the data, useful for visualization. Linear Discriminant Analysis assumes a multivariate Gaussian distribution, so it can generate additional data as needed.

Singular Value Decomposition

First, a recap of *eigenvectors* and *eigenvalues*. Given a matrix $\mathbf{A}$, a vector v and a scalar λ if it holds that $\mathbf{A}v = \lambda v$ then v is an *eigenvector* of $\mathbf{A}$ and λ is the corresponding *eigenvalue*. In other words, if we multiply $\mathbf{A}$ by the vector v we get back the same vector, scaled by some constant.

Given $\mathbf{X} \in \mathbb{R}^{N \times p}$ we write its SVD as $\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top$. Where:

- $\mathbf{U} \in \mathbb{R}^{N \times p}$: Matrix of the left singular vectors.
- $\mathbf{V} \in \mathbb{R}^{p \times p}$: Matrix of the right singular vectors.
- $\mathbf{D} = diag\{d_1, d_2, \ldots, d_p\} \in \mathbb{R}^{p \times p}$, where for all the singular values it holds that $d_1 \geq d_2 \geq \ldots \geq d_p \geq 0$.

See Figure 1 for a visualization of this decomposition.

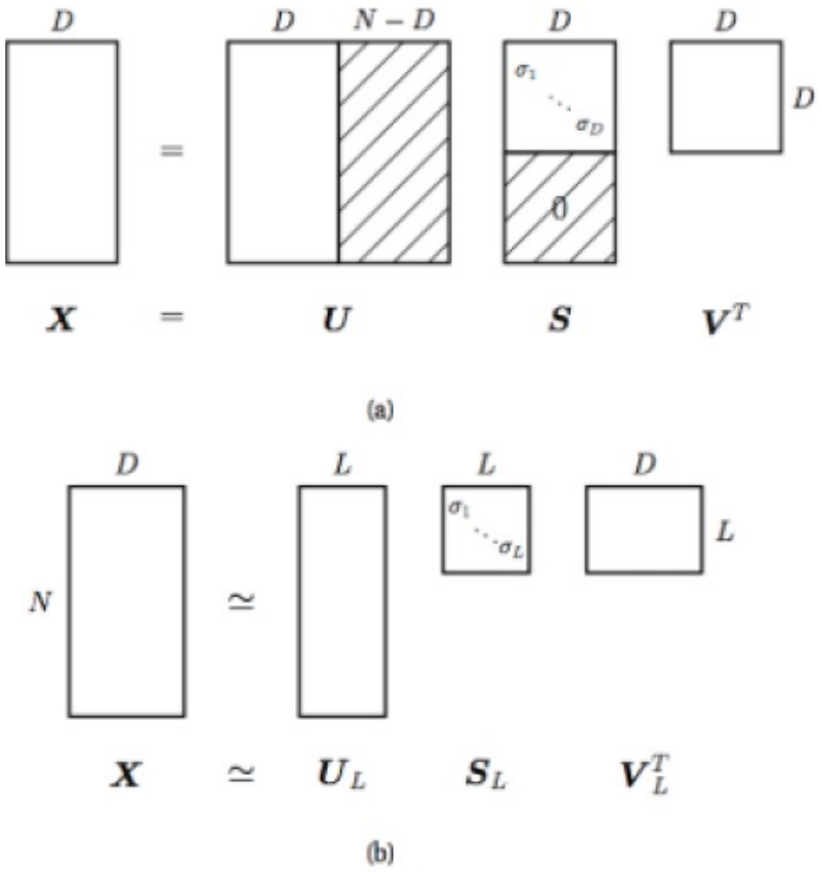


Figure 12.8 (a) SVD decomposition of non-square matrices $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$. The shaded parts of $\mathbf{S}$, and all the off-diagonal terms, are zero. The shaded entries in $\mathbf{U}$ and $\mathbf{S}$ are not computed in the economy-sized version, since they are not needed. (b) Truncated SVD approximation of rank L.

Singular Value Decomposition

One thing that's very important to notice is that, if any single values from $\mathbf{D}$ equals to 0, some of our data (observation) is redundant. That way, we can say that the SVD can help us identify and minimize the existence of redundant data in our observations.

$\mathbf{U}$ and $\mathbf{V}$ both have orthogonal columns, so $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ and $\mathbf{V}^\top\mathbf{V} = \mathbf{I}$ where $\mathbf{I}$ is the identity matrix of size $p \times p$.

The SVD can help us extract useful information out of the original matrix $\mathbf{X}$. Consider the matrix $\mathbf{X}^\top\mathbf{X} \in \mathbb{R}^{p\times p}$:

$$\mathbf{X}^\top\mathbf{X} = (\mathbf{U}\mathbf{D}\mathbf{V}^\top)^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top = \mathbf{V}\mathbf{D}\mathbf{U}^\top\mathbf{U}\mathbf{D}\mathbf{V}^\top$$

D is diagonal, hence $\mathbf{D}^\top = \mathbf{D}$. Since $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$ we get that:

$$\mathbf{X}^\top\mathbf{X} = \mathbf{V}\mathbf{D}^2\mathbf{V}^\top$$

In other words, $\mathbf{V}\mathbf{D}^2\mathbf{V}^\top$ is the eigendecomposition of $\mathbf{X}^\top\mathbf{X}$. Notice that, by construction, the eigenvalues (i.e., entries of the diagonal matrix $\mathbf{D}^2$) are non-negative. This means the matrix $\mathbf{X}^\top\mathbf{X}$ is *positive semi-definite*.

We can also write the SVD as a sum of outer products:

$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^\top = \sum_{j=1}^{p} d_j u_j v_j^\top$$

Where u_j and v_j are the jth columns of $\mathbf{U}$ and $\mathbf{V}$, respectively. The sum in the above equation is a sum of rank 1 matrices weighted by the scalars d_j.

Singular Value Decomposition

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	2	2	2	0	0
John	1	1	1	0	0
Jack	5	5	5	0	0
Jill	0	0	0	2	2
Jenny	0	0	0	3	3
Jane	0	0	0	1	1

- we use economy-size decomposition to reduce extra zeros in singular value matrix, along with columns in $\mathbf{U}$,$\mathbf{V}$ that multiply those zeros, which reduces original dimension.
- $\mathbf{U}$ connects people to concepts: Joe likes SciFi ($\mathbf{U}_{1,1} = .18$) but not romance ($\mathbf{U}_{1,2} = 0$).
- $\mathbf{V}$ relates movies to concepts: *The Matrix* (the movie) has $\mathbf{V}_{1,1} = .58$ and $\mathbf{V}_{1,2} = 0$, which suggests that its genre is SciFi rather than romance.
- Σ is the strength of each concept. In this movie case, the strength of the SciFi concept is 12.4, while the strength of the romance concept is 9.5, meaning the SciFi concept is stronger because SciFi has more data supported information on the movies of that genre and people who like it.

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 2 & 2 & 2 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 2 & 2 \\ 0 & 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} .18 & 0 \\ .36 & 0 \\ .18 & 0 \\ .90 & 0 \\ 0 & .53 \\ 0 & .80 \\ 0 & .27 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 9.64 & 0 \\ 0 & 5.29 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}}_{V^T}$$

Given a vector of ratings from a new user Quincy: q = [4; 0; 0; 0; 0]^T. (he only rated one movie, The Matrix) we can get his genre preferences by calculating qV = [2:32; 0]. We can also try to predict his rating for the movies he hasn't seen by calculating qVV^T = [1:35; 1:35; 1:35; 0; 0].

Singular Value Decomposition

We can also use SVD to approximate pictures as seen in Figure 4. The approximation is done by keeping only the leading entries (singular values) in $\mathbf{D}$, or equivalently, setting the singular values below some threshold to zero, and reconstructing the matrix. The more singular values we truncate, the coarser the approximation. In Figure 4 the top right image was reconstructed using only 2 singular values, the bottom left image used 5 and the bottom image used 20 (out of the original 200 nonzero singular values).

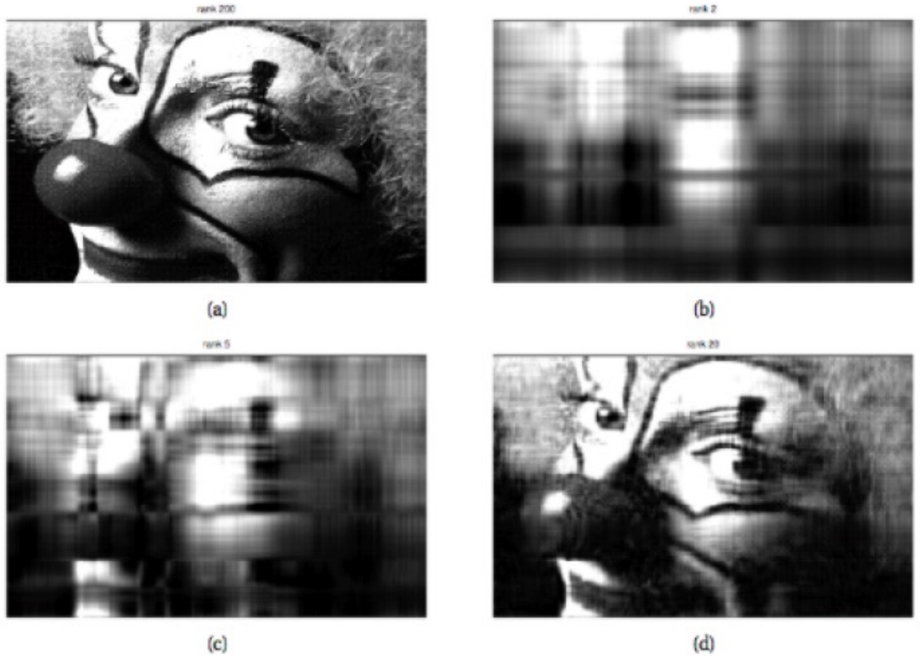


Figure 12.9 Low rank approximations to an image. Top left: The original image is of size 200×320, so has rank 200. Subsequent images have ranks 2, 5, and 20. Figure generated by `svdImageDemo`.

Fisher's Linear Discriminant

In many applications we need to work with high dimensional data, which can be difficult to visualize. Figure 6 shows an example in which coffee bean odor is characterized by a 60 dimensional feature vector obtained using a gas sensor array. Is it possible to find a lower dimensional representation of this data that preserves the separation between the classes? Fisher LDA provides one method for doing this.

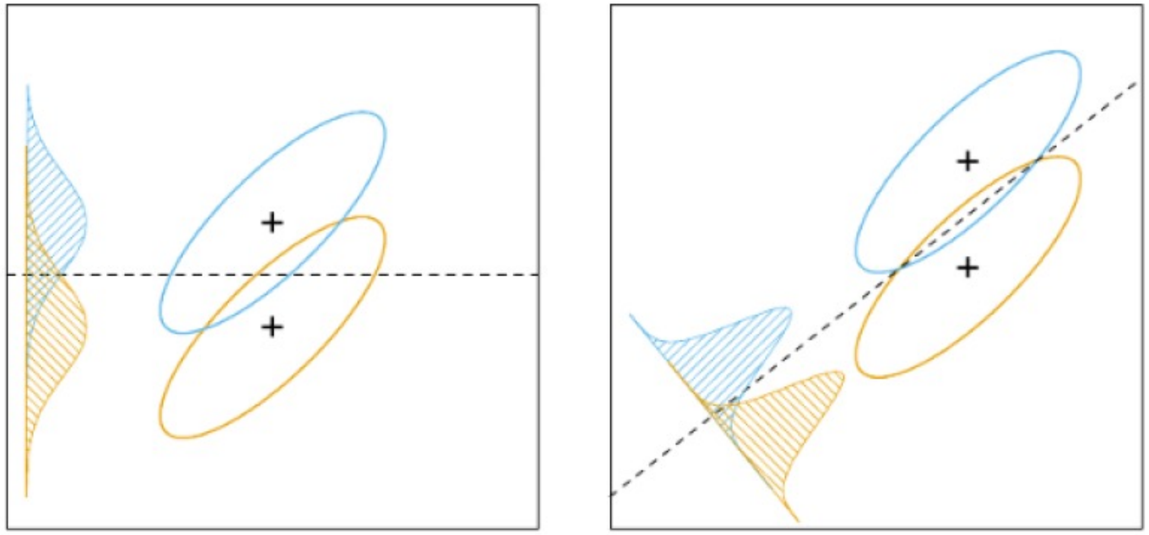
However, it is important to note that dimensionality reduction lowers data fidelity (unless a dimension is a linear variation of another).

Fisher's Linear Discriminant

Consider the two class densities depicted in Fig. 5. Suppose we wanted to project this down to one dimension. What is the best projection direction? Fisher posed the problem as follows:

> Find the linear combination $Z = a^\top X$ such that the *between-class variance* is maximized relative to the *within-class variance*.

This means that we are trying to find a projection that minimizes the overlap between the two densities. On our example, the image on the left shows the projection along the x-axis, i.e., using the vector $a = [1, 0]^\top$. That projection does keep the means separated, but there's still a troubling amount of overlap between the projected densities. The example on the right shows a preferable projection direction, since the projected densities have considerably less overlap.



Fisher's Linear Discriminant

- $\mathbf{M} \in \mathbb{R}^{K \times p}$: matrix of class centroids (means). It contains the K sample means $\hat{\mu}_k$ (transposed) in its rows. The kth row is a vector of the kth class with the length of p feature means.
- $\mathbf{W} \in \mathbb{R}^{p \times p}$: the within-class scatter matrix. This is the same (up to a constant) as the pooled sample covariance $\hat{\Sigma}$ (pooling means ignoring the class).
- $\mathbf{B} \in \mathbb{R}^{p \times p}$: the between-class scatter matrix. It is equal (up to a constant) to the covariance matrix of the class centroids (i.e., rows of $\mathbf{M}$).

To compute $\mathbf{B}$ we take the mean of the class means, which we'll denote $\hat{\mu}$ (mean of the individuals mean vectors). $\mathbf{B}$ is then given by

$$\mathbf{B} = \sum_{k=1}^{K} (\hat{\mu}_k - \hat{\mu})(\hat{\mu}_k - \hat{\mu})^\top$$

In the 2-class case, one can use $(\hat{\mu}_1 - \hat{\mu}_2)(\hat{\mu}_1 - \hat{\mu}_2)^\top$ instead of $\mathbf{B}$ without affecting the solution for a.

To answer the problem posed by Fisher, we maximize the following *generalized Rayleigh quotient*, which is a ratio of two quadratic forms:

$$\max_a \frac{a^\top \mathbf{B} a}{a^\top \mathbf{W} a}; a \in \mathbb{R}^{p \times 1}$$

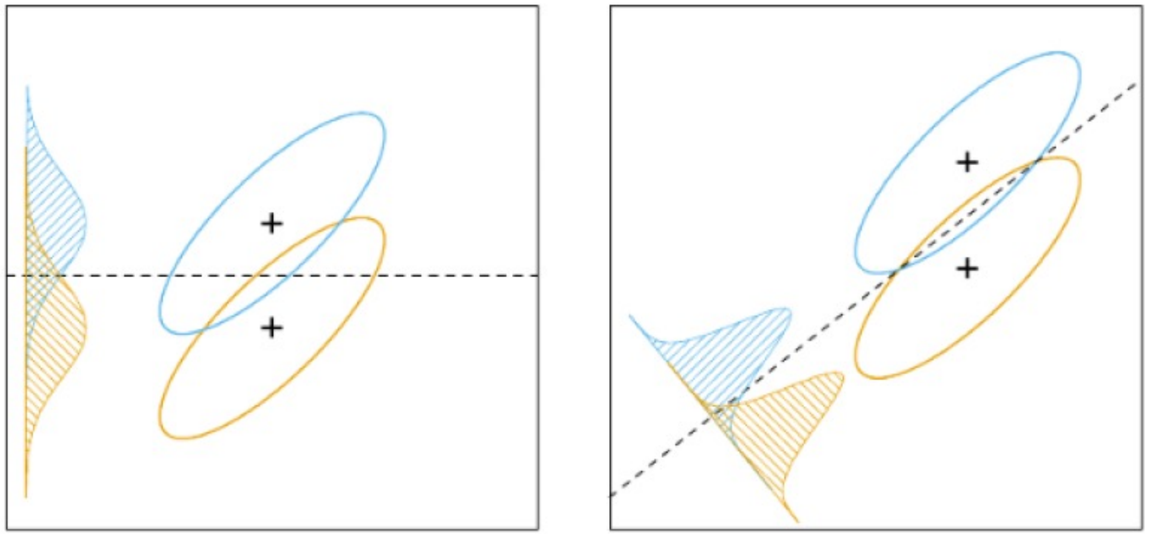
Equivalently we can solve the constrained maximization problem

$$\max_a a^\top \mathbf{B} a \quad \text{subject to} \quad a^\top \mathbf{W} a = 1$$

Fisher's Linear Discriminant

This is a generalized eigenvalue problem with a equal to the leading eigenvector[1] of $\mathbf{W}^{-1}\mathbf{B}$. The vector a is the best projection direction of the data from p dimensions to 1 dimension. Since we are looking for direction, the magnitude of a is insignificant without loss of generality. In Figure 5, a is the direction of the vector on the right. We call the estimated a vector the 1st *discriminant coordinate* or *canonical variate* for the data, and we'll denote it a_1 here. In this example we would stop with a_1, but in general, when $p > 2$, we can solve for the next best projection direction a_2 by maximizing the Rayleigh quotient subject to the additional constraint that a_2 be orthogonal to a_1 in $\mathbf{W}$, i.e., $a_2^\top \mathbf{W} a_1 = 0$. For the next best projection, a_3, we assume that it is orthogonal to both a_1 and a_2 in $\mathbf{W}$, and so on.

To summarize, Fisher's LDA takes into account both the separation of the means and the spread (or "scatter") of the data when selecting the best projection direction.



PCA vs LDA

Like Fisher's linear discriminant, PCA is a dimensionality reduction method that finds successive projection directions, but unlike LDA it is unsupervised. PCA just computes one covariance matrix for all the data, while in Fisher's Linear Discriminant there are two covariance matrices (one of them is within-class and the other is between-class). The first *principal axis* is the axis along which the data exhibits the greatest variance. (We'll cover how to compute this later in the semester.) In the example shown in Figure 5, that would give us roughly the opposite of what Fisher LDA gives us, i.e., almost completely overlapping class densities. This isn't meant as a criticism of PCA, since it's not the right tool to use when you have labelled classes, it's just intended as a point of comparison. We'll still find PCA highly useful for dimensionality reduction in other problem settings.

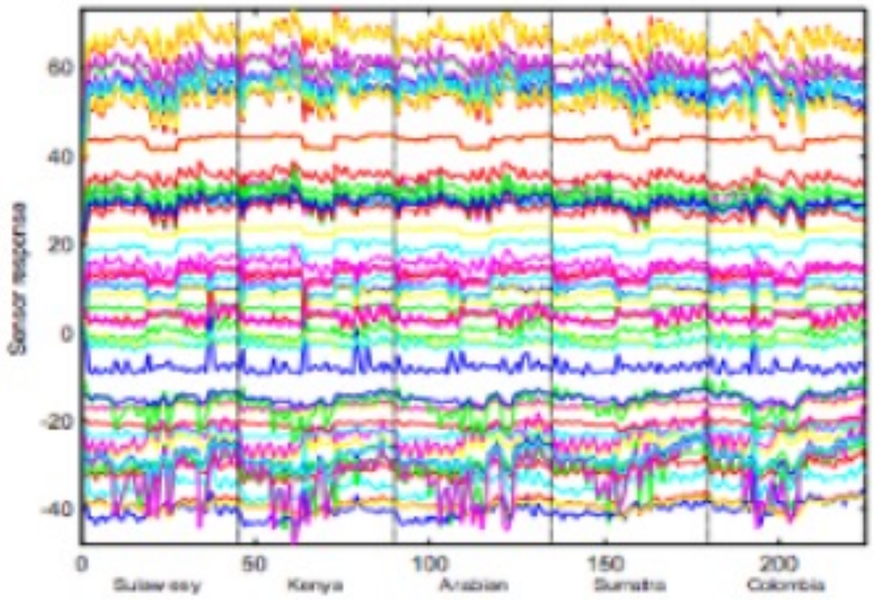
PCA vs LDA

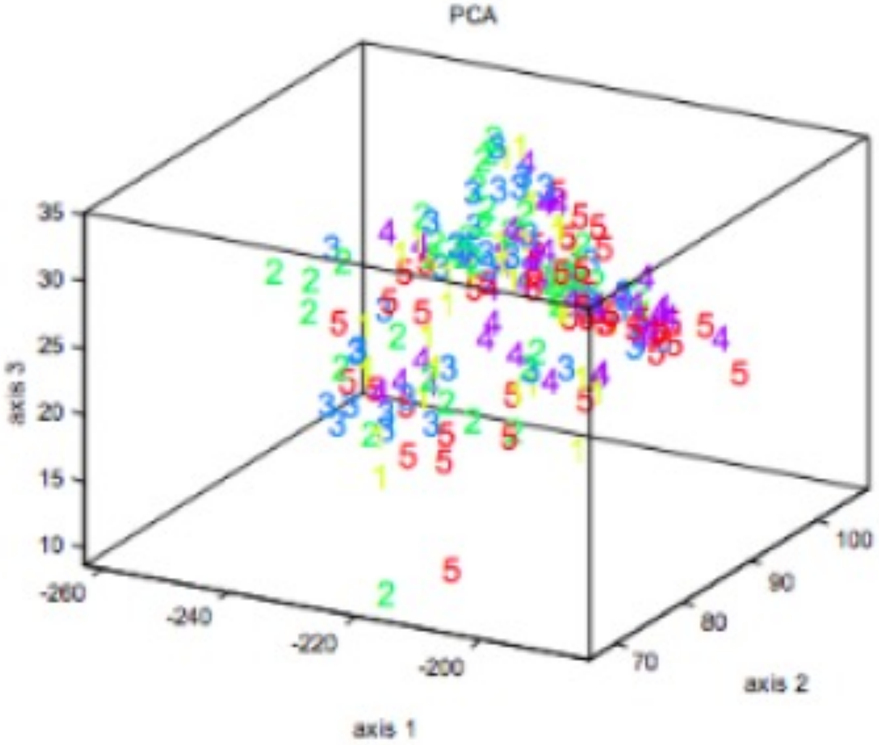
This example illustrates the performance of PCA and LDA on an odor recognition problem

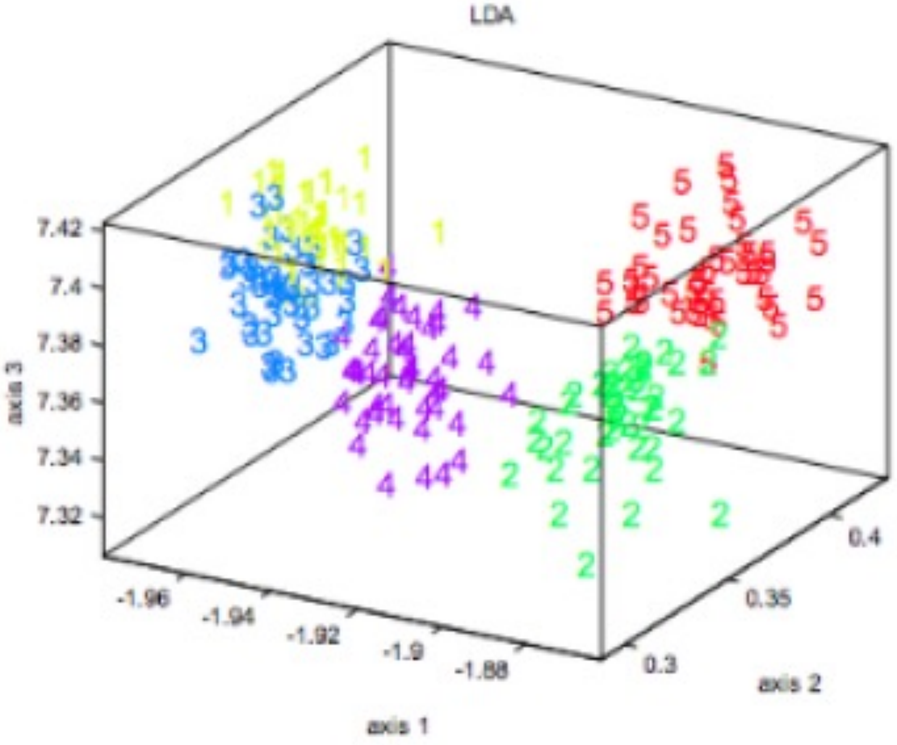
- Five types of coffee beans were presented to an array of gas sensors
- For each coffee type, 45 "sniffs" were performed and the response of the gas sensor array was processed in order to obtain a 60-dimensional feature vector

Results

- From the 3D scatter plots it is clear that LDA outperforms PCA in terms of class discrimination
- This is one example where the discriminatory information is not aligned with the direction of maximum variance

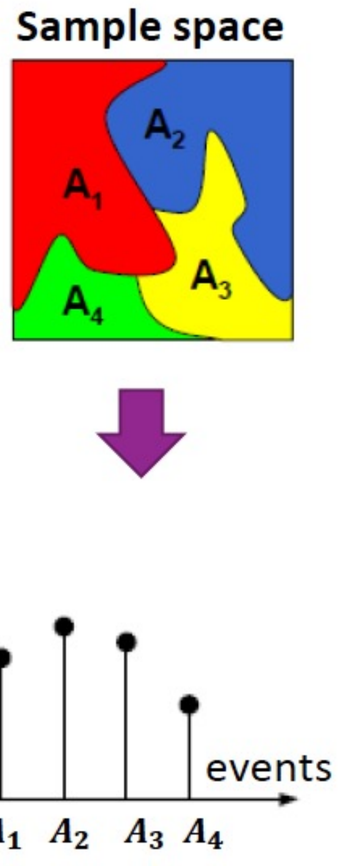






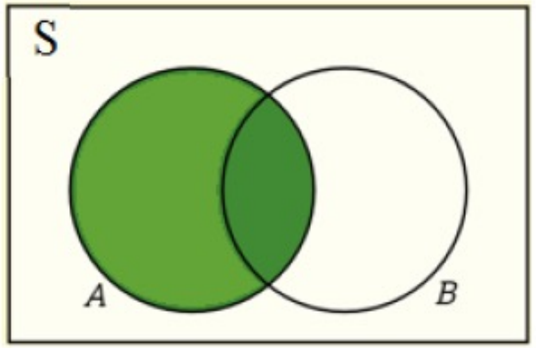
Probability Theory

- Probability theory is the **study of uncertainty**
- Probabilities are numbers assigned to events that indicate ***"how likely"*** it is that the event will occur when a random experiment is performed

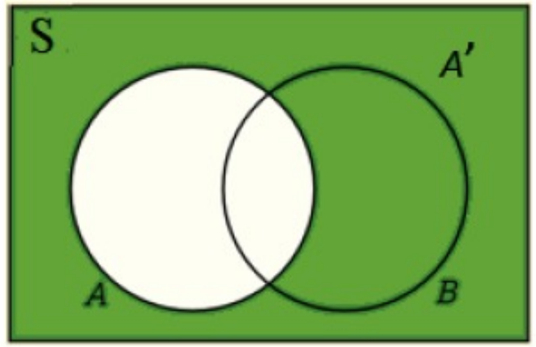


- Terminology:
  - **Sample space (S):** The **set** of all the outcomes of a random experiment.
  - **Event space (E):** (or set of events) : A set whose elements (called events) are subsets of S
  - **Probability measure (P)**: A function $P: E \rightarrow \mathbb{R}$ that satisfies the following axioms:
    - Axiom I: $0 \leq P(A_i) \leq 1 \quad \forall\ A_i \in E$
    - Axiom II: $P(S) = 1$
    - Axiom III: If $A_i \cap A_j = \emptyset$ (A_i and A_j are disjoint events) then $P(A_i \cup A_j) = P(A_i) + P(A_j)$

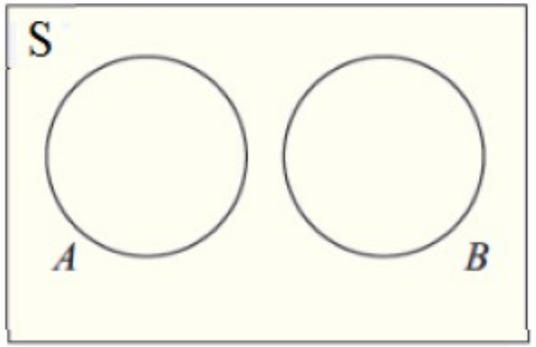
Basic Definitions: Venn Diagrams



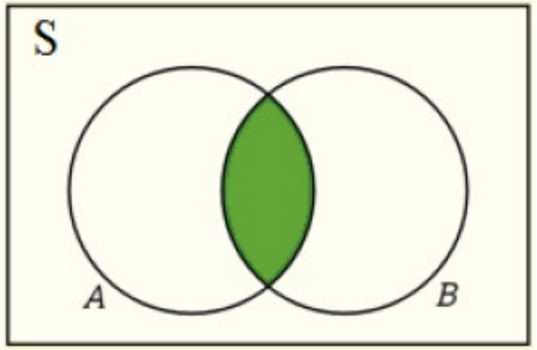
Set A



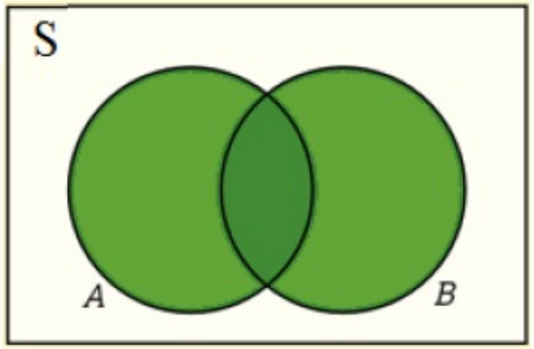
A' the complement of A



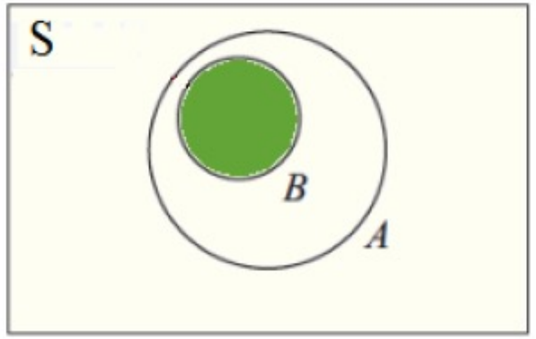
A and B are disjoint sets
$P(A \cap B) = 0$



A and B $\quad A \cap B$
A intersect B



A or B $\quad A \cup B$
A union B



B is subset of A $\quad B \subset A$

$P(A \cup B) = P(A \text{ or } B) = P(\text{event A occurs or event B occurs or both occur})$
$P(A \cap B) = P(A \text{ and } B) = P(\text{both event A occurs and event B occurs})$

More Properties of Probability

- $P(\emptyset) = 0$
- $P(S) = 1$
- $0 \leq P(E) \leq 1 \quad \forall E \subseteq S$
- $P(\bar{E}) = 1 - P(E)$
- $If\,(A \subseteq B) \quad \Rightarrow \quad P(A) \leq P(B)$

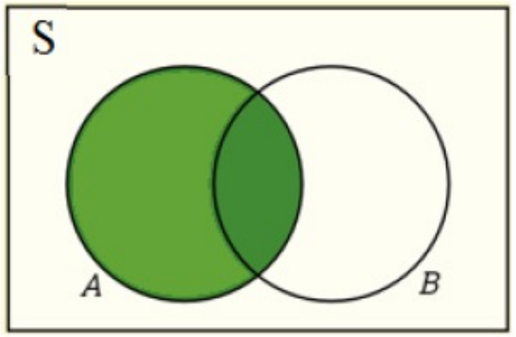
Conditional Probability

- If A and B are two events, the probability of event A when we already know that event B has occurred is defined by:
  - Read as "**Probability of A given B**"

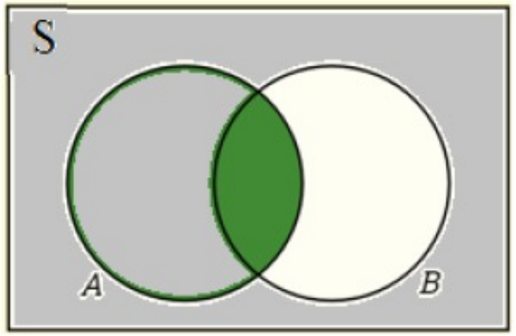
$$P(A)$$

$$\boldsymbol{P(A|B)} = \frac{\mathrm{P}(\mathrm{A} \cap B)}{P(B)}$$

Probability of A given B



B has occurred



The original sample space S becomes B

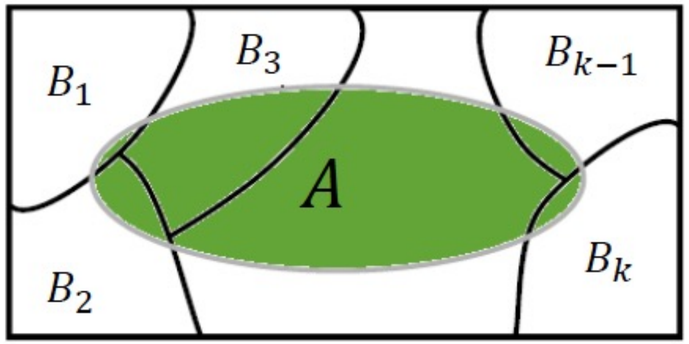
- P(B) simply re-normalizes the probability of events that occur jointly with B

- We also have:

$$P(A \cap B) = P(B)P(A|B)$$

Total Probability Theorem

- $If\ B_1, B_2, \ldots, B_k$ are a set of disjoint events such that $\bigcup_{i=1}^{k} B_i = S$ then $\sum_i^k P(\mathrm{B}_i) = 1$
- An event A can be represented as:



$$A = A \cap S = A \cap (B_1 \cup B_2 \cup \cdots \cup B_k)$$
$$= (A \cap B_1) \cup (A \cap B_2) \cup \cdots \cup (A \cap B_k)$$

- Since $B_1, B_2, \ldots, B_k$ are a set of disjoint events:

$$P(A) = P\big((A \cap B_1) \cup (A \cap B_2) \cup \cdots \cup (A \cap B_k)\big) = P(A \cap B_1) + P(A \cap B_2) + \cdots + P(A \cap B_k)$$

- and, therefore:

$$\boxed{P(A) = P(A|B_1)P(B_1) + P(A|B_2)P(B_2) + \cdots + P(A|B_k)P(B_k) = \sum_{i=1}^{k} P(A|B_i)P(B_i)}$$

Total Probability Theorem

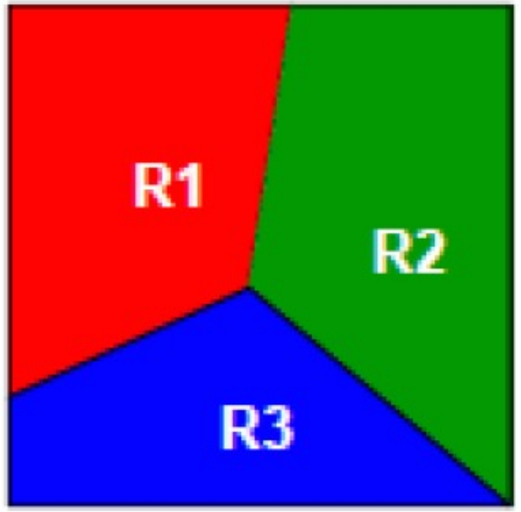
recall: $if\ A\ and\ B$ disjoints $\Rightarrow P(A \cup B) = P(A) + P(B)$

recall: $P(A \cap B) = P(A|B)\,P(B)$

Introduction: Classifier

❑ Classifier:

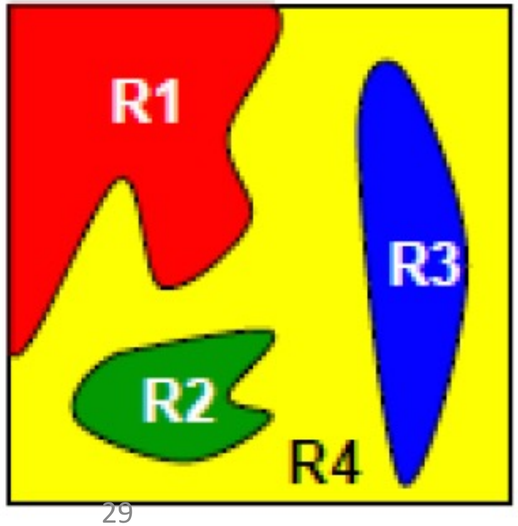
A classifier is an hypothesis (rule) that assigns a class label to any object in a particular object representation format.



❑ Decision boundaries:

borders between decision regions are called

- The classification of feature vector ***x*** consists of determining x belongs to which decision region
- We have:
  - Linear decision boundary
    - Piecewise linear decision boundary
  - Non-linear decision boundary



❑ Classification accuracy/ Classification error

Introduction: Classification

- For the purpose of pattern classification, suppose we have K disjoint classes $C_1, C_2 \dots, C_K$, and we are going to classify the sample $\boldsymbol{x}$.
  - $\boldsymbol{x}$ shows the feature vector of the sample

- In classification, the goal is to find a mapping from inputs $\boldsymbol{X}$ to output $\boldsymbol{T}$ given the training samples (a set of labeled samples, i.e. input-output pairs)

- Training set $D = \{(\boldsymbol{x}_1, t_1), (\boldsymbol{x}_2, t_2), \dots, (\boldsymbol{x}_m, t_m)\}$ where $t_i \in \{C_1, C_2, \dots, C_K\}$
  - m: #samples
  - K: #class
    - $K = 2$ is called binary classification. Often assume $t \in \{-1,1\}$ or $t \in \{0,1\}$
    - $K > 2$ is called multi-class classification
  - $\boldsymbol{x_i} = [x_1, x_2, \dots, x_n]^T$ where n shows the dimension of each sample, i.e. #features

Bayesian Decision Theory (Bayes Classifier)

- Given a classification task of K classes, $C_1;\ C_2; \dots;\ C_K$, and an input vector $\boldsymbol{x}$, we can form K conditional probabilities:

$$p(C_k|\boldsymbol{x}) \quad \forall\, k = 1;\ 2; \dots\ ; K$$

- What can be a decision rule?
  - Choose the class that is most 'probable' given the observed feature vector $\boldsymbol{x}$
  - More formally: Evaluate the posterior probability of each class $P(C_k|x)$ and choose the class with largest $P(C_k|x)$, i.e.:

$$\text{Class of sample } \boldsymbol{x} = C^* = \underset{C_k}{argmax}\ P(C_k|\boldsymbol{x}) \qquad \forall\, k = 1;\ 2; \dots\ ; K$$

Bayes Theorem & Statistical Pattern Recognition

- Each term in the **Bayes Theorem** has a special name, which **you should be familiar with.** ⚠

$$P(C_k|\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_k)P(C_k)}{\sum_{i=1}^{N_C} P(\boldsymbol{x}|C_k)P(C_k)} = \frac{P(\boldsymbol{x}|C_k)P(C_k)}{p(\boldsymbol{x})}$$

- $\boldsymbol{P(C_k)}$: **Prior probability** (Probability of class C_k)
- $\boldsymbol{P(x|C_k)}$: **Likelihood** (Conditional probability of observation $\boldsymbol{x}$ given class C_k)
- $\boldsymbol{P(C_k|x)}$: **Posterior Probability** (Probability of class C_k given the observation $\boldsymbol{x}$)
- $p(\boldsymbol{x})$: A **normalization factor** which is constant and doesn't affect the decision. Why?

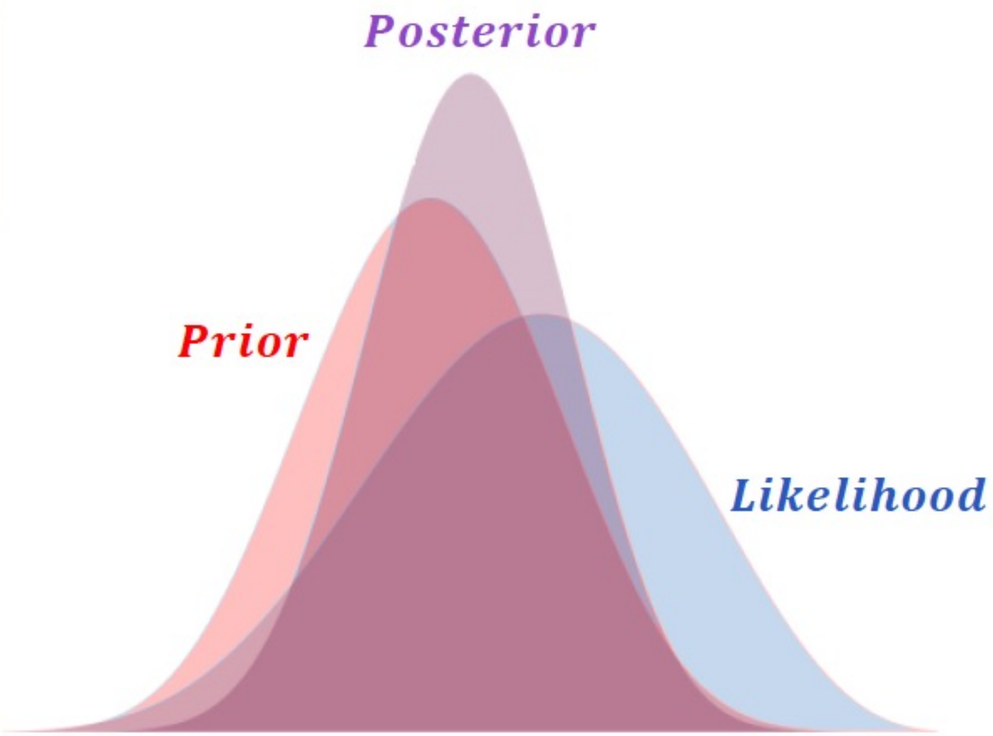
$$\boldsymbol{Posterior} = \frac{\boldsymbol{Likelihood} \times \boldsymbol{Prior}}{\textbf{normalization factor}}$$

Bayes Theorem

Bayes Theorem & Statistical Pattern Recognition

Bayes Theorem

$$Posterior = \frac{Likelihood \times Prior}{\text{normalization factor}}$$



- Generally, it can be said that
  - Likelihood = P(observation | unknown)
  - Posterior Probability = $P(\text{unknown} \mid \text{observation})$

recall: $P(A \cap B) = P(A|B)\,P(B)$

recall: $P(A) = \sum_{i=1}^{k} P(A|B_i)P(B_i)$

Bayes Theorem

- $If\ B_1, B_2, \ldots, B_k$ are a set of disjoint events such that $\bigcup_{i=1}^{k} B_i = S$ and event A occurs, what is the probability of B_j?

Thomas Bayes (1702-1761)

$$P(B_j|A) = \frac{P(A \cap B_j)}{P(A)} = \frac{P(A|B_j)P(B_j)}{\sum_{i=1}^{k} P(A|B_i)P(B_i)}$$

Bayes Theorem

- This is known as **Bayes Theorem** or **Bayes Rule**, and is (one of) the most useful relations in probability and statistics, especially in **Statistical Pattern Recognition**.

Bayesian. Decision Theory (Bayes Classifier)

- Given a classification task of K classes, $C_1;\ C_2; \dots;\ C_K$, and an input vector $\boldsymbol{x}$, we can form K conditional probabilities:

$$p(C_k|\boldsymbol{x}) \quad \forall\, k = 1;\ 2; \dots\ ; K$$

- What can be a decision rule?
  - Choose the class that is most 'probable' given the observed feature vector $\boldsymbol{x}$
  - More formally: Evaluate the posterior probability of each class $P(C_k|x)$ and choose the class with largest $P(C_k|x)$, i.e.:

$$\text{Class of sample } \boldsymbol{x} = C^* = \underset{C_k}{argmax}\ P(C_k|\boldsymbol{x}) \qquad \forall\, k = 1;\ 2; \dots\ ; K$$

Bayesian. Decision Theory & Likelihood Ratio Test

- Without loss of generality, for a binary (2-class) classification, the decision rule will be:
  - If $p(C_1|\boldsymbol{x}) > p(C_2|\boldsymbol{x})$ then $\boldsymbol{x}$ is classified to C_1
  - If $p(C_1|\boldsymbol{x}) < p(C_2|\boldsymbol{x})$ then $\boldsymbol{x}$ is classified to C_2
  - If $p(C_1|\boldsymbol{x}) == p(C_2|\boldsymbol{x})$ then $\boldsymbol{x}$ is classified to either C_1 *or* C_2
    - Which one is better C_1 or C_2?

In other words

$$p(C_1|\boldsymbol{x}) \underset{C_1}{\overset{C_2}{\lessgtr}} p(C_2|\boldsymbol{x})$$

- Applying Bayes rule:

$$\frac{P(\boldsymbol{x}|C_1)P(C_1)}{P(\boldsymbol{x})} \underset{C_1}{\overset{C_2}{\lessgtr}} \frac{P(\boldsymbol{x}|C_2)P(C_2)}{P(\boldsymbol{x})}$$

$P(\boldsymbol{x})$ dose not affect on the decision rule

$$P(\boldsymbol{x}|C_1)P(C_1) \underset{C_1}{\overset{C_2}{\lessgtr}} P(\boldsymbol{x}|C_2)P(C_2)$$

- By rearranging the expression, the **minimization of Bayes error** leads to:

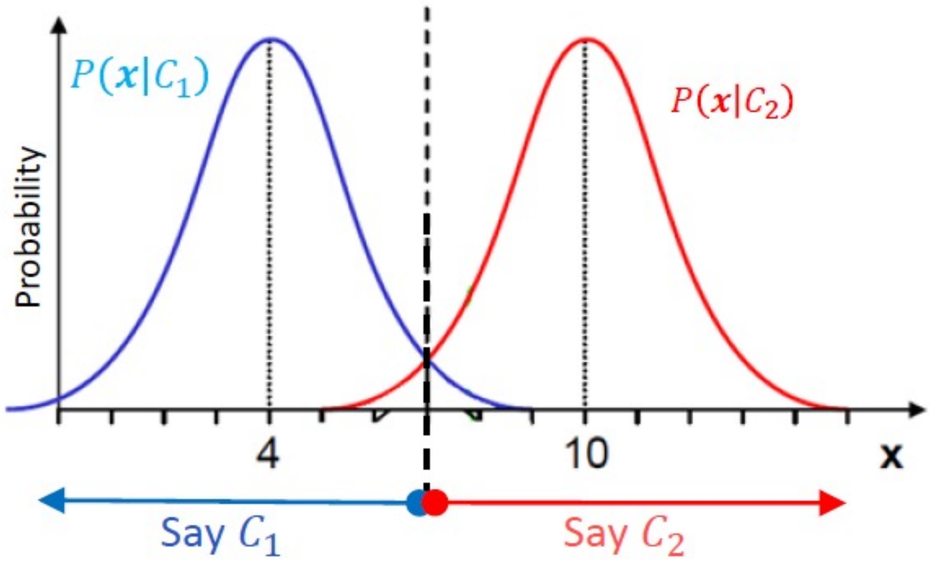
$$\Lambda(\boldsymbol{x}) = \underbrace{\frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)}}_{\text{Likelihood ratio}} \underset{C_1}{\overset{C_2}{\lessgtr}} \frac{P(C_2)}{P(C_1)}$$

Likelihood Ratio Test (LRT)

Likelihood Ratio Test: example

- Consider a binary classification problem with the following likelihood functions:

$$P(\boldsymbol{x}|C_1) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-4)^2}{2}\right) \qquad P(\boldsymbol{x}|C_2) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-10)^2}{2}\right)$$



- Derive a decision rule by assuming equal class priors

$$\Lambda(\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} \underset{C_1}{\overset{C_2}{\lessgtr}} \frac{P(C_2)}{P(C_1)} \quad \Rightarrow \quad \Lambda(\boldsymbol{x}) = \frac{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-4)^2}{2}\right)}{\frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-10)^2}{2}\right)} \underset{C_1}{\overset{C_2}{\lessgtr}} \frac{0.5}{0.5}$$

$$\Rightarrow \quad \exp\left(-\frac{(x-4)^2}{2}\right) \underset{C_1}{\overset{C_2}{\lessgtr}} \exp\left(-\frac{(x-10)^2}{2}\right) \quad \Rightarrow \quad (x-4)^2 \underset{C_1}{\overset{C_2}{\gtrless}} (x-10)^2 \quad \Rightarrow \quad (x-4)^2 - (x-10)^2 \underset{C_1}{\overset{C_2}{\gtrless}} 0$$

$$\Rightarrow \quad x \underset{C_1}{\overset{C_2}{\gtrless}} 7$$

How would change the decision rule if $P(C_1) = 2P(C_2)$? (Exercise)

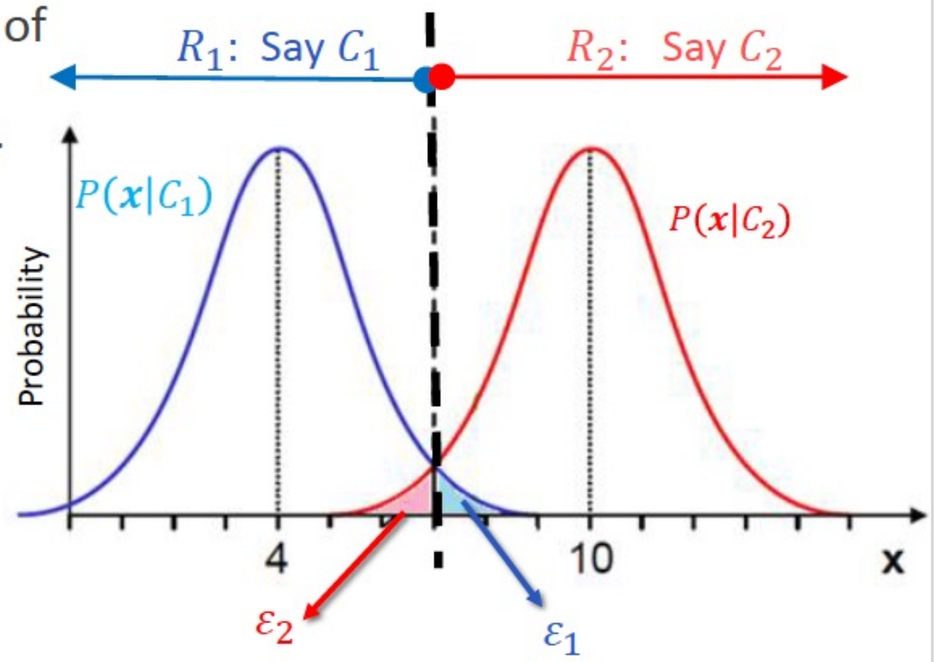
The probability of error

- The performance of a decision rule can be measured by its probability of error $P(error)$
  - Based on **Total Probability Theorem**, the error can be produced by each class. Therefore, we have:

$$P(error) = \sum_{i=1}^{K} \boxed{P(error|C_i)}\, P(C_i)$$

class conditional probability of error

$$P(error|C_i) = P(choose\ C_k|C_i) = P(x \in R_k|C_i) = \int_{R_k} P(\boldsymbol{x}|C_i)dx$$



- So, for our binary problem:

$$P(error) = P(C_1)\underbrace{\int_{R_2} P(\boldsymbol{x}|C_1)dx}_{\varepsilon_1} + P(C_2)\underbrace{\int_{R_1} P(\boldsymbol{x}|C_2)dx}_{\varepsilon_2}$$

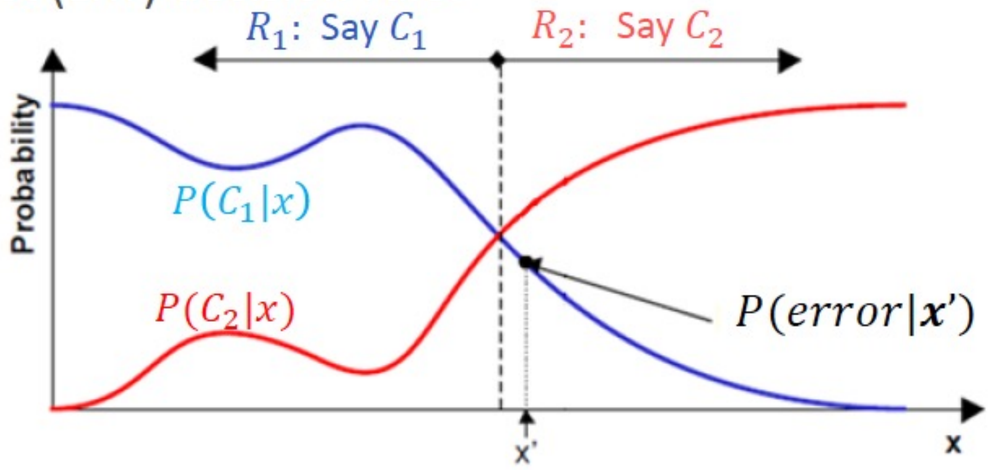
  - With equal priors $P(error) = \frac{\varepsilon_1+\varepsilon_2}{2}$ (why?)

Minimizing the probability of error

- The best decision rule is the one that minimizes the probability of error.
- For this purpose it is convenient to minimize $P(error)$ over all sample space, i.e.:
  - Based on Total Probability Theorem

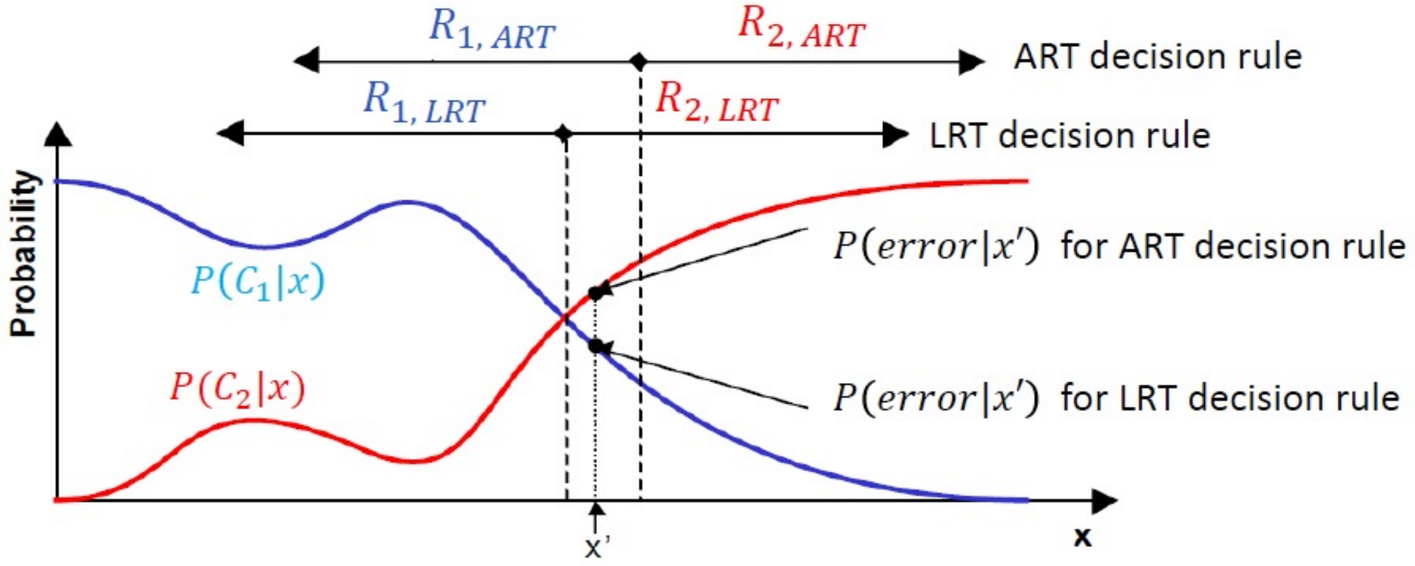
$$P(error) = \int_{-\infty}^{+\infty} P(error|x)\, P(x)\, dx$$

- At each point x', $P(error|\boldsymbol{x}')$ is equal to $P(C_i|\boldsymbol{x}')$ where we choose the other class C_k
  - For Likelihood Ratio Test (LRT) decision rule:



Minimizing the probability of error & LRT

- How good is the Likelihood Ratio Test (LRT) decision rule?
  - Based on the following figures, consider LRT decision rule versus **Another** decision rule called **ART**



  - From the figure it is clear that, for any value of x', the LRT decision rule will always have a lower $P(error|x')$

- Therefore:
  - For any given problem, the **minimum probability of error** is achieved by **LRT decision rule**.
  - This probability of error is called the **Bayes Error Rate** and is the **BEST any classifier can do**

Bayes Risk

- The classification probability of error is not always the best criteria to find the best classifier. Why?
  - This is because the classification probability of error assigns the same importance to the all errors

- In some applications some wrong decisions may have more serious implications than others such as:
  - Intrusion detection system, patient classification, and spam filtering
  - For example, misclassifying a cancer tissue as a healthy patient is a much more serious problem than the other way around

- This concept can be formalized by assigning a penalty term λ_{ik} to weight each error
  - λ_{ik} represents the cost of choosing class C_i when class C_k is the true class

- We define the **Bayes Risk** as **the expected value of the cost** and try to minimize the Bayes risk:

$$\Re = E[loss] = \sum_{i=1}^{K}\sum_{k=1}^{K} \lambda_{ik}\ P(choose\ C_i\ \ and\ \ x \in C_k) = \sum_{i=1}^{K}\sum_{k=1}^{K} \lambda_{ik}\ P(C_k)\, P(x \in R_i | C_k)$$

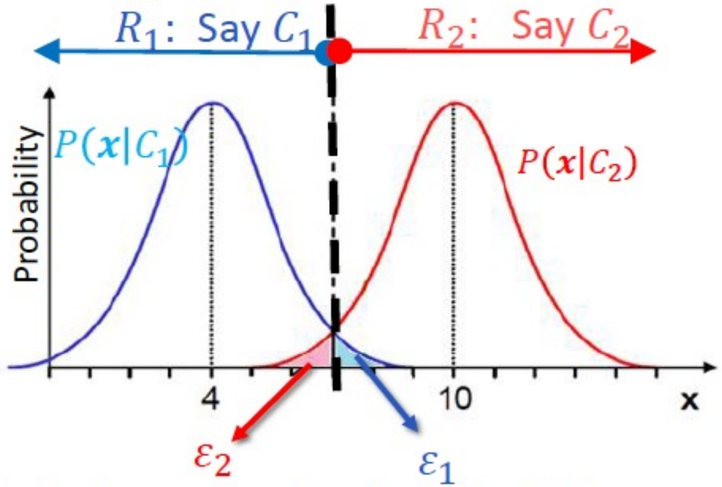
Bayes Risk

Minimizing the Bayes Risk

$$\Re = E[loss] = \sum_{i=1}^{K}\sum_{k=1}^{K} \lambda_{ik}\ P(choose\ C_i\ \ and\ \ x \in C_k) = \sum_{i=1}^{K}\sum_{k=1}^{K} \lambda_{ik}\ P(C_k)\,P(x \in R_i|C_k)$$

Bayes Risk

- When the minimization of the Bayes Risk is equivalent to minimizing the classification error probability (Bayes error)?



- What is the decision rule that minimizes the Bayes Risk?
  - Remind that $P(x \in R_i|C_k) = \int_{R_i} P(\boldsymbol{x}|C_k)dx$
  - For a binary classification problem, we have:

$$\Re = \int_{R_1}[\lambda_{11}P(C_1)P(\boldsymbol{x}|C_1) + \lambda_{12}P(C_2)P(\boldsymbol{x}|C_2)]dx\ +\ \int_{R_2}[\lambda_{21}P(C_1)P(\boldsymbol{x}|C_1) + \lambda_{22}P(C_2)P(\boldsymbol{x}|C_2)]dx$$

  - The optimal decision rule will be achieved by minimizing the expected risk $\Re$

Minimizing the Bayes Risk

- Let us give a intuitive solution to minimize the Bayes Risk
  - Let

$$r_1 = \lambda_{11} P(C_1) P(\boldsymbol{x}|C_1) + \lambda_{12} P(C_2) P(\boldsymbol{x}|C_2)$$
$$r_2 = \lambda_{21} P(C_1) P(\boldsymbol{x}|C_1) + \lambda_{22} P(C_2) P(\boldsymbol{x}|C_2)$$

  - What can be a reasonable decision rule to determine the class of a sample $\boldsymbol{x}$?
  - We assign $\boldsymbol{x}$ to C_1 if $r_1 < r_2$, generally:

$$r_1 \underset{C_2}{\overset{C_1}{\lessgtr}} r_2 \quad \Rightarrow \quad (\lambda_{21} - \lambda_{11}) P(C_1) P(\boldsymbol{x}|C_1) \underset{C_2}{\overset{C_1}{\gtrless}} (\lambda_{12} - \lambda_{22}) P(C_2) P(\boldsymbol{x}|C_2)$$

  - In other words, **minimization of Bayes Risk** leads to:

$$\frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} \frac{(\lambda_{12} - \lambda_{22})}{(\lambda_{21} - \lambda_{11})} \frac{P(C_2)}{P(C_1)}$$

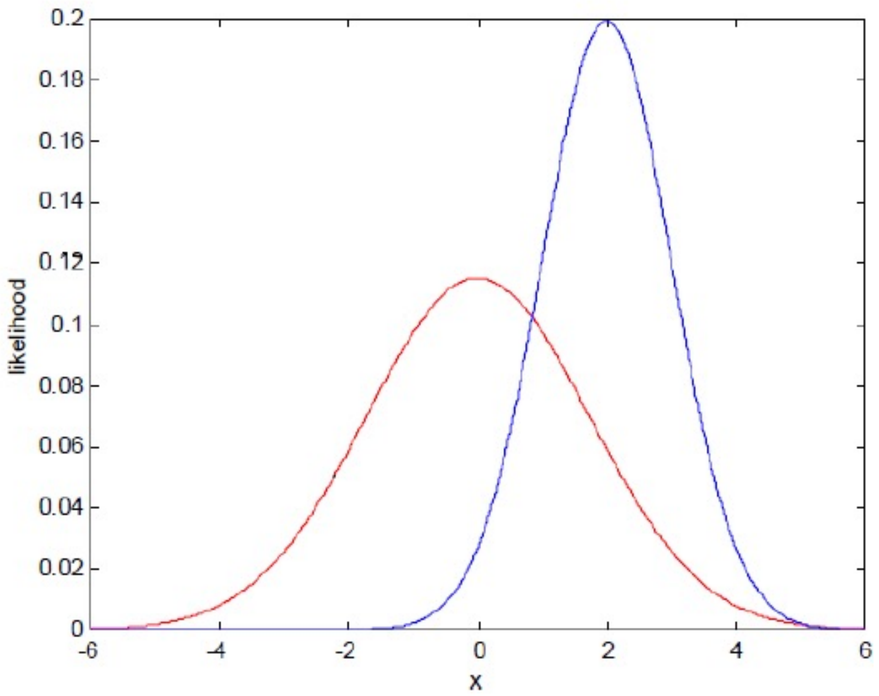
This is also a likelihood ratio test

When it is equivalent to minimizing the Bayes error?

Likelihood Ratio Test: Exercise

- Consider a binary classification problem with the following likelihood functions:

$$P(\boldsymbol{x}|C_2) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{(x-2)^2}{2}\right) \qquad P(\boldsymbol{x}|C_1) = \frac{1}{\sqrt{2\pi}\sqrt{3}} \exp\left(-\frac{(x)^2}{3}\right)$$



  - Sketch the two densities
  - What is the likelihood ratio?
  - Assume $P(C_1) = P(C_2) = 0.5$, determine a decision rule that minimize the Bayes error
  - Assume $P(C_1) = P(C_2) = 0.5$ and $\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ \sqrt{3} & 0 \end{bmatrix}$, determine a decision rule that minimize the Bayes risk

Variations of the Likelihood Ratio Test

- **Bayes** decision rule: The LRT decision rule that minimizes the Bayes Risk:

$$\Lambda(\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} \frac{(\lambda_{12}-\lambda_{22})}{(\lambda_{21}-\lambda_{11})} \frac{P(C_2)}{P(C_1)}$$

Bayes decision rule

- **Maximum A Posteriori (MAP)** decision rule: The LRT decision rule that minimizes the Bayes error (Bayes risk with zero-one cost function $\begin{bmatrix} \lambda_{11} & \lambda_{12} \\ \lambda_{21} & \lambda_{22} \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ called :

$$\Lambda(\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} \frac{P(C_2)}{P(C_1)}$$

Maximum A Posteriori (MAP) decision rule

- **Maximum Likelihood (ML)** decision rule: The LRT decision rule with equal priors and zero-one cost function

.

$$\Lambda(\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} 1$$

Maximum Likelihood (ML) decision rule

Variations of the Likelihood Ratio Test

- **Neyman-Pearson** decision rule (1933, J. Neyman, E. Pearson): it fixes one class error probabilities, $\varepsilon_1 < \alpha$, and seeks to minimize the other, i.e. ε_2

$$\Lambda(\boldsymbol{x}) = \frac{P(\boldsymbol{x}|C_1)}{P(\boldsymbol{x}|C_2)} \underset{C_2}{\overset{C_1}{\gtrless}} \eta \qquad \text{Let } \varepsilon_1 < \alpha \text{ and find } \eta \text{ which minimizes } \varepsilon_2$$

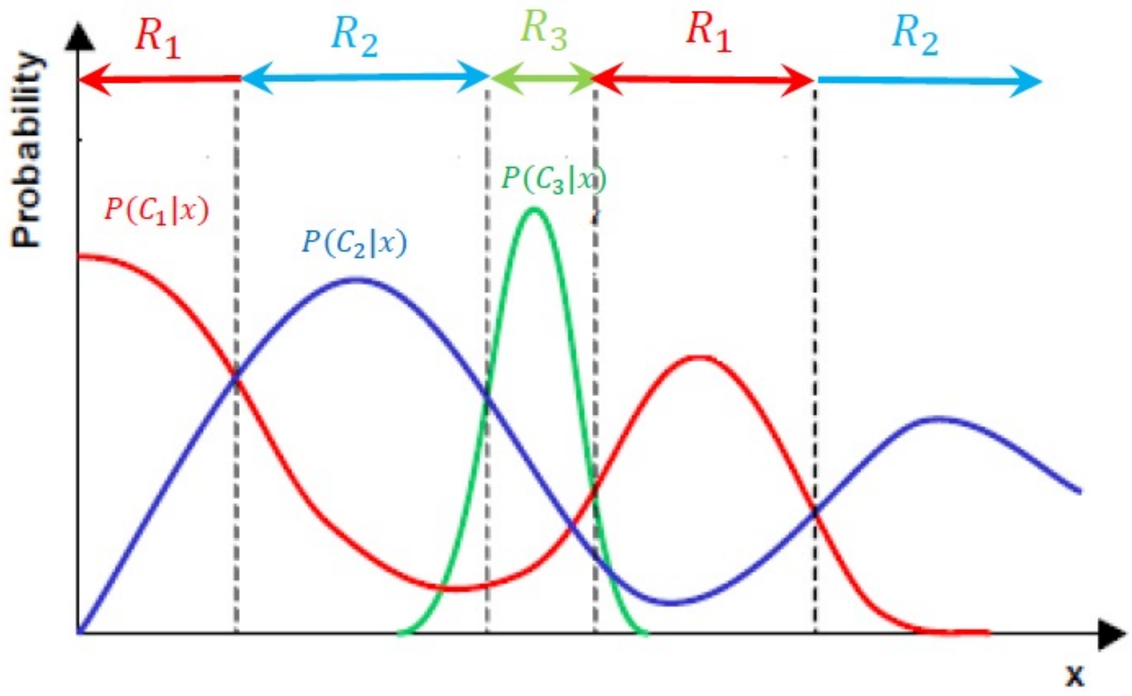
Neyman-Pearson decision rule

  - It is commonly used in Detection and Estimation Theory
  - For example:
    - Consider detection problem in a cancer detection. It can be a binary classification.
    - Due to our sensitivity, we must not miss more than 1% of patients. It means that we are interested in $\varepsilon_1 < 0.01$. Therefore, we try to minimize ε_2 while $\varepsilon_1 < 0.01$

- The Neyman-Pearson Criterion is very useful when **it does not require knowledge of priors and cost function**

Minimize Bayes Error for Multi class problems

- The decision rule that minimizes P[error] generalizes very easily to multi-class problems
- P[error] is minimized by the MAP Criterion, i.e., select the class with maximum posterior



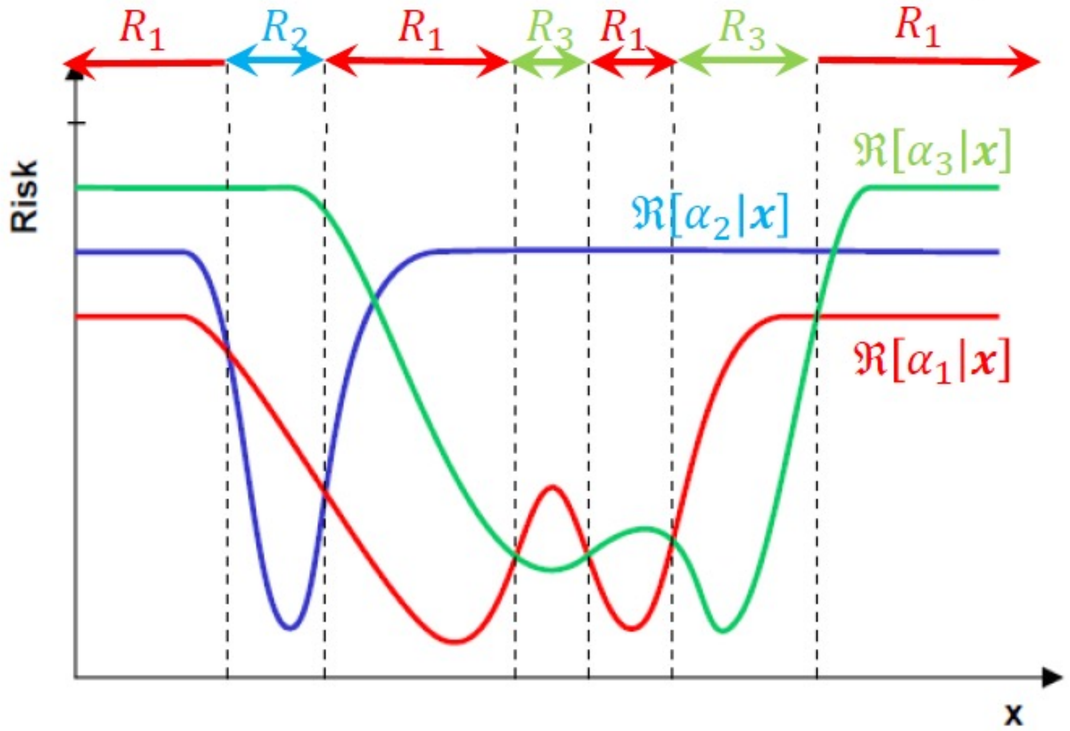
Minimize Bayes Error for Multi class problems

- Let the risk of assigning $\boldsymbol{x}$ to class C_i as:
  - $\Re risk$[assigning $\boldsymbol{x}$ to class $C_i|\boldsymbol{x}$]$= \Re[\alpha_i|\boldsymbol{x}]$

$$\Re[\alpha_i|\boldsymbol{x}] = \sum_{k=1}^{K} \lambda_{ik} P(C_k|\boldsymbol{x})$$

- The Bayes risk over all samples will be:

$$\Re = E[loss] = \int \boldsymbol{P}(\boldsymbol{x})\, \Re[\alpha|\boldsymbol{x}] d\boldsymbol{x}$$



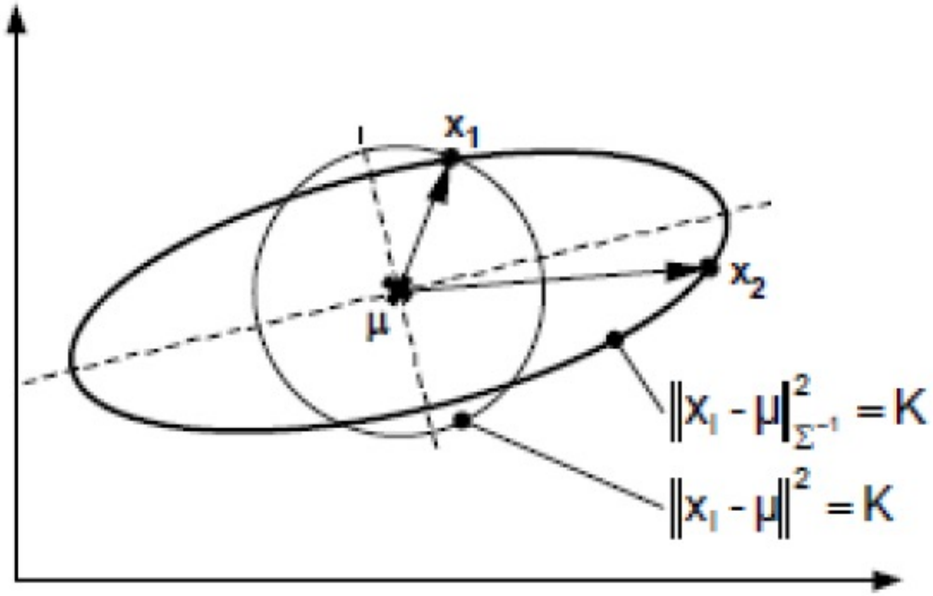
- In order to minimize $\Re[X]$, we will have to minimize the conditional risk $\Re[\alpha|\boldsymbol{x}]$ at each sample x in the feature space
  - It is equivalent to choosing C_i such that $\Re(\alpha i|x)$ is minimum

Euclidean & Mahalanobis Distance

Very important distance in statistical pattern recognition

EUCLIDEAN DISTANCE

$$\|\boldsymbol{x_k} - \boldsymbol{\mu}\|^2 = (\boldsymbol{x_k} - \boldsymbol{\mu})^T(\boldsymbol{x_k} - \boldsymbol{\mu}) = k$$



MAHALANOBIS DISTANCE

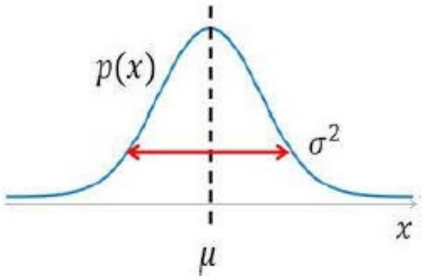
$$\|\boldsymbol{x_k} - \boldsymbol{\mu}\|^2_{\Sigma^{-1}} = (\boldsymbol{x_k} - \boldsymbol{\mu})^T\Sigma^{-1}(\boldsymbol{x_k} - \boldsymbol{\mu}) = k$$

- Mahalanobis distance uses Σ^{-1} norm
- Σ^{-1} shows a stretching factor on the space
- For $\Sigma^{-1} = I$ Mahalanobis distance becomes the Euclidean distance

Gaussian (normal) Distribution

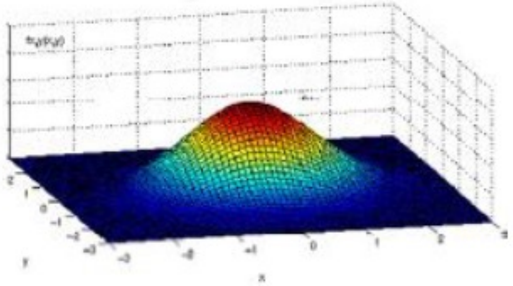
- 1-dim Gaussian distribution with mean μ and variance σ^2 is given by:

$$P(x) = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x-\mu)^2}{|2\sigma^2}\right)$$



- d-dim Gaussian distribution with mean vector $\boldsymbol{\mu}$ and covariance matrix Σ is given by:

$$P(\boldsymbol{x}) = \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{(2\pi)^{d/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu})^T \Sigma^{-1} (\boldsymbol{x}-\boldsymbol{\mu})\right)$$



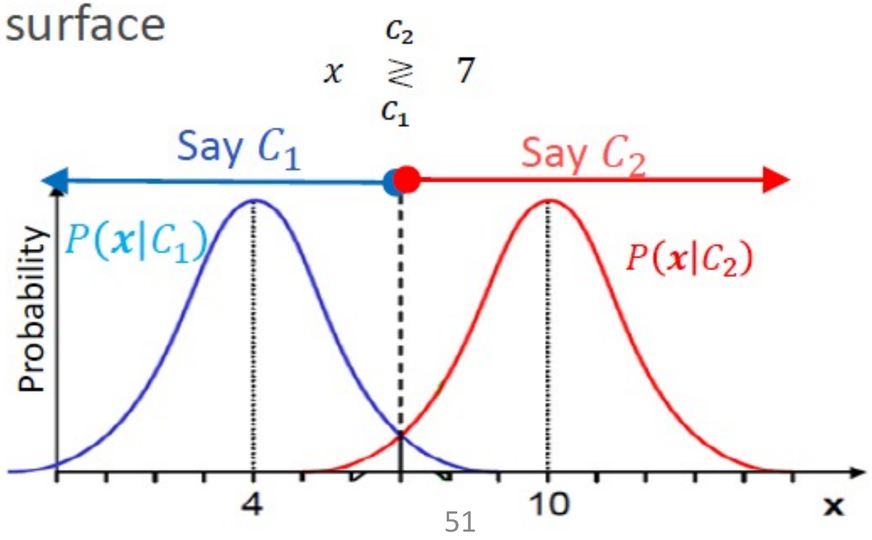
Decision Boundary & Discriminant Function

- A classifier portions the feature space into K regions for K classes.
- Decision boundaries: borders between decision regions are called decision boundaries
- For a binary classification problem
  - The decision surface is computed by:

$$g_2(\boldsymbol{x}) = g_1(\boldsymbol{x}) \quad \Rightarrow \quad g_2(\boldsymbol{x}) - g_1(\boldsymbol{x}) = 0 \quad \text{(solve it to find the decision boundary)}$$

  - $g_2(\boldsymbol{x}) - g_1(\boldsymbol{x}) = 0$: The samples located exactly on the decision surface
  - $g_2(\boldsymbol{x}) - g_1(\boldsymbol{x}) > 0$: The samples assigned to class C_2
  - $g_2(\boldsymbol{x}) - g_1(\boldsymbol{x}) < 0$: The samples assigned to class C_1

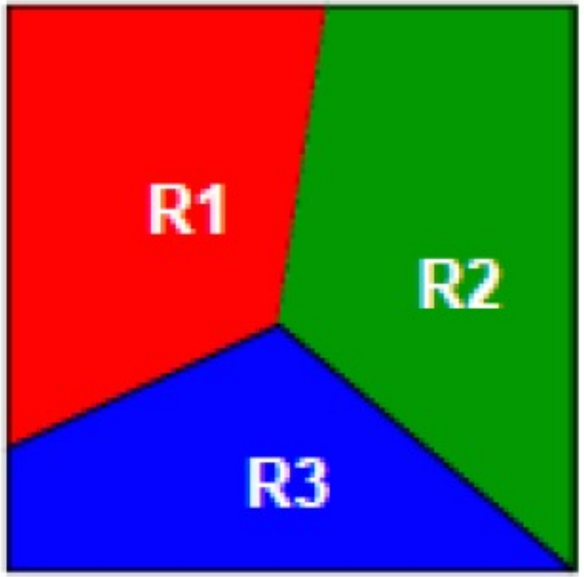
For ML decision rule, the decision boundary is calculated by solving: $P(\boldsymbol{x}|C_1) = P(\boldsymbol{x}|C_2)$



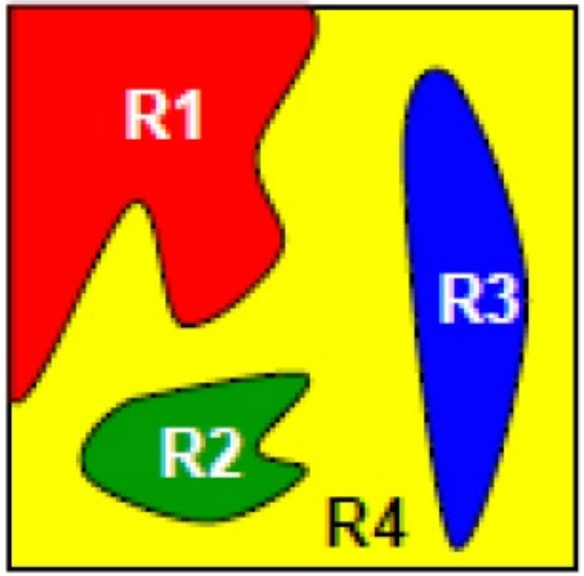
Decision Boundary & Discriminant Function

- The decision surfaces, for a K-class problems:

$$g_{ki}(\boldsymbol{x}) = g_k(\boldsymbol{x}) - g_i(\boldsymbol{x}) \quad \forall\, k; i \in \{1, 2, \ldots, K\} \;\; and \;\; k \neq i$$



Linear decision boundary
(Piecewise linear decision boundary)



Non-linear decision boundary

Bayes Classifiers for normally distributed classes

- MAP discriminant function is $g_k(\boldsymbol{x}) = P(C_k|\boldsymbol{x})$
  - Using Bayes rule:

$$g_k(\boldsymbol{x}) = P(C_k|\boldsymbol{x}) = \frac{P(x|C_k)P(C_k)}{P(x)} = \frac{1}{(2\pi)^{d/2}|\Sigma_\mathrm{k}|^{1/2}} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma_\mathrm{k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right) P(C_k)\frac{1}{P(\boldsymbol{x})}$$

  - Eliminating the constant terms:

$$g_k(\boldsymbol{x}) = |\Sigma_\mathrm{k}|^{-1/2} \exp\left(-\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma_\mathrm{k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k)\right) P(C_k)$$

  - It is preferable to work with the following discriminant function
    - By taking log which is a monotonically increasing function

$$\boxed{g_k(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma_\mathrm{k}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log(|\Sigma_\mathrm{k}|) + \log\big(P(C_k)\big)}$$

General discriminant function for a MAP rule

    - This expression is called a quadratic discriminant function

Bayes Classifiers for normally distributed classes

- General discriminant function for a MAP rule becomes:

$$g_k(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T\Sigma_{\mathrm{k}}^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log(|\Sigma_{\mathrm{k}}|) + \log\big(P(C_k)\big)$$

- We consider the Bayes classifiers for Normally distributed classes in:
  - Case 1: $\Sigma_k = \sigma^2 I$
    - Σ diagonal and same for all classes with the same variances at each direction
  - Case 2: $\Sigma_k = \Sigma$
    - Σ diagonal and same for all classes
    - Σ non-diagonal but same for all classes
  - Case 3: $\Sigma_{\mathrm{k}} = \sigma_k^2 I$
    - Σ diagonal with different variances
  - Case 4: $\Sigma_k \neq \Sigma_i$
    - General case

When a covariance matrix is diagonal?

Case 1 $\Sigma_k = \sigma^2 I$

❑ $\Sigma_k = \sigma^2 I = \sigma^2 \begin{bmatrix} 1 & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & 1 \end{bmatrix}$

$$g_k(x) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log(|\Sigma_k|) + \log\big(P(C_k)\big)$$

- It means that the features are independent (diagonal Σ) with the same variances for all classes
- In this case:

$$g_k(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T(\sigma^2 I)^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log(|\sigma^2 I|) + \log\big(P(C_k)\big)$$

- Eliminating the sesond term which is constant:

$$g_k(\boldsymbol{x}) = -\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T(\boldsymbol{x}-\boldsymbol{\mu}_k) + \log\big(P(C_k)\big) = -\frac{1}{2\sigma^2}\big(\boldsymbol{x}^T\boldsymbol{x} - 2\boldsymbol{\mu}_k^T\boldsymbol{x} + \boldsymbol{\mu}_k^T\boldsymbol{\mu}_k\big) + \log\big(P(C_k)\big)$$

- Eliminating the term $\boldsymbol{x}^T\boldsymbol{x}$ which is constant for all classes:

$$g_k(\boldsymbol{x}) = -\frac{1}{2\sigma^2}\big(-2\boldsymbol{\mu}_k^T\boldsymbol{x} + \boldsymbol{\mu}_k^T\boldsymbol{\mu}_k\big) + \log\big(P(C_k)\big) = \boldsymbol{w}_k^T\boldsymbol{x} + w_{k0} \quad where \begin{cases} \boldsymbol{w}_k = \dfrac{\boldsymbol{\mu}_k}{\sigma^2} \\ -\dfrac{1}{2\sigma^2}\boldsymbol{\mu}_k^T\boldsymbol{\mu}_k + \log\big(P(C_k)\big) \end{cases}$$

- This discriminant function is linear
- Therefore, the decision boundaries $g_k(\boldsymbol{x}) = g_i(\boldsymbol{x})$, will be hyper-planes

Case 1 $\Sigma_k = \sigma^2 I$

$$g_k(\boldsymbol{x}) = -\frac{1}{2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T \Sigma_k^{-1}(\boldsymbol{x}-\boldsymbol{\mu}_k) - \frac{1}{2}\log(|\Sigma_k|) + \log\big(P(C_k)\big)$$

- If we assume equal priors:

$$g_k(\boldsymbol{x}) = -\frac{1}{2\sigma^2}(\boldsymbol{x}-\boldsymbol{\mu}_k)^T(\boldsymbol{x}-\boldsymbol{\mu}_k)$$

  - This is called **minimum distance classifier** or **nearest mean classifier**
  - For unit variance ($\sigma^2 = 1$), the distance becomes the Euclidean distance

Generative approach

- <u>Inference stage</u>
  - Determine class conditional densities $p(\boldsymbol{x}|y = k)$ and priors $p(y = k)$
  - Use the Bayes theorem to find $p(y = k|\boldsymbol{x})$

It models the distribution of input features as well as labels (and so it can generate synthetic data points)

- <u>Decision stage</u>: After learning the model (inference stage), make optimal class assignment for new input
  - if $p(y = i|\boldsymbol{x}) > p(y = j|\boldsymbol{x}) \quad \forall j \neq i$ then decide $y = i$

Generative approach: Bayes theorem

- Bayes' theorem

Posterior Likelihood Prior

$$p(y=k|\boldsymbol{x}) = \frac{p(\boldsymbol{x}|y=k)p(y=k)}{p(\boldsymbol{x})}$$

- Posterior probability: $p(y=k|\boldsymbol{x})$
- Likelihood or class conditional probability: $p(\boldsymbol{x}|y=k)$
- Prior probability: $p(y=k)$

$p(\boldsymbol{x})$: pdf of feature vector $\boldsymbol{x}$ ($p(\boldsymbol{x}) = \sum_{k=1}^{K} p(\boldsymbol{x}|y=k)p(y=k)$)
$p(\boldsymbol{x}|y=k)$: pdf of feature vector $\boldsymbol{x}$ for samples of class $y=k$
$p(y=k)$: probability of the label be k

Generative approach: Bayesian decision rule

- If $P(y = 1|\boldsymbol{x}) > P(y = 2|\boldsymbol{x})$ decide $\mathcal{C}_1$
  otherwise decide $\mathcal{C}_2$

Equivalent

- If $\frac{p(x|y=1)P(y=1)}{p(x)} > \frac{p(x|y=2)P(y=2)}{p(x)}$ decide $\mathcal{C}_1$
  otherwise decide $\mathcal{C}_2$

Equivalent

- If $p(\boldsymbol{x}|y = 1)P(y = 1) > p(\boldsymbol{x}|y = 2)P(y = 2)$ decide $\mathcal{C}_1$
  otherwise decide $\mathcal{C}_2$

In general:

If $p(\boldsymbol{x}|y = i)P(y = i) > p(\boldsymbol{x}|y = j)P(y = j) \;\; \forall j \neq i$ then decide $\mathcal{C}_i$

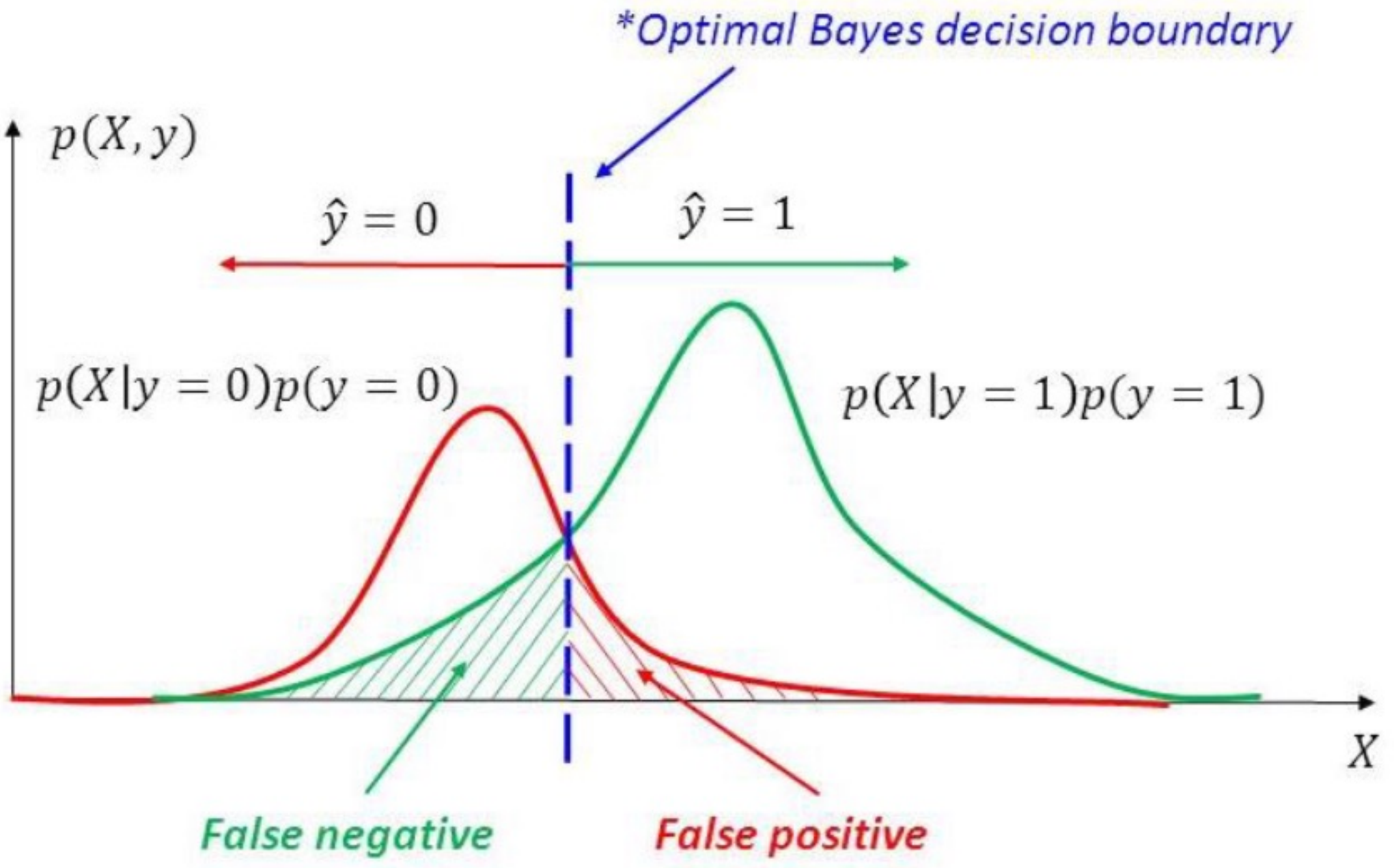
Discriminative approach

- <u>Inference stage</u>
  - Determine the posterior class probabilities $P(y = k|\boldsymbol{x})$ directly

- <u>Decision stage</u>: After learning the model (inference stage), make optimal class assignment for new input
  - if $P(y = i|\boldsymbol{x}) > P(y = j|\boldsymbol{x}) \quad \forall j \neq i$ then decide $\mathcal{C}_i$

How to find probabilities?

- We use Maximum Likelihood to find parameters:
  - **Generative models:** Maximum data likelihood
    - Finds $\widehat{\boldsymbol{\theta}}$ that maximizes $p(\boldsymbol{X}, \boldsymbol{Y} | \boldsymbol{\theta})$
    - It can compute $p(\boldsymbol{X} | \boldsymbol{\theta})$
  - **Discriminative models:** Maximum conditional data likelihood
    - Finds $\widehat{\boldsymbol{\theta}}$ that maximizes $p(\boldsymbol{Y} | \boldsymbol{X}, \boldsymbol{\theta})$ that is required for classification
    - Doesn't waste effort on finding $p(\boldsymbol{X} | \boldsymbol{\theta})$


*Optimal Bayes decision boundary
$p(X, y)$
$\hat{y} = 0$
$\hat{y} = 1$
$p(X|y = 0)p(y = 0)$
$p(X|y = 1)p(y = 1)$
X
False negative
False positive

Naïve Bayes Review

Consider the example of distinguishing cis-males and cis-females using a 2D feature vector with height and weight as shown in Figure 1. We assume Gaussian class conditional densities

$$p(x|y=c,\theta_c) = \mathcal{N}(x|\mu_c, \Sigma_c)$$

where the mean μ_c and covariance matrix Σ_c depend on the class (cis-male or cis-female). The scatter plot in Figure 1 shows the raw data points along with covariance ellipses.

Although the covariance ellipses clearly show a correlation between the two variables, Naive Bayes assumes the features are independent given the class, i.e., the joint density is just a product of the individual densities:

$$p(x|y=c,\theta_c) = \prod_{d=1}^{p} p(x_d|y=c,\theta_c) = \prod_{d=1}^{p} \mathcal{N}(x_d|\mu_{cd}, \sigma_{cd})$$

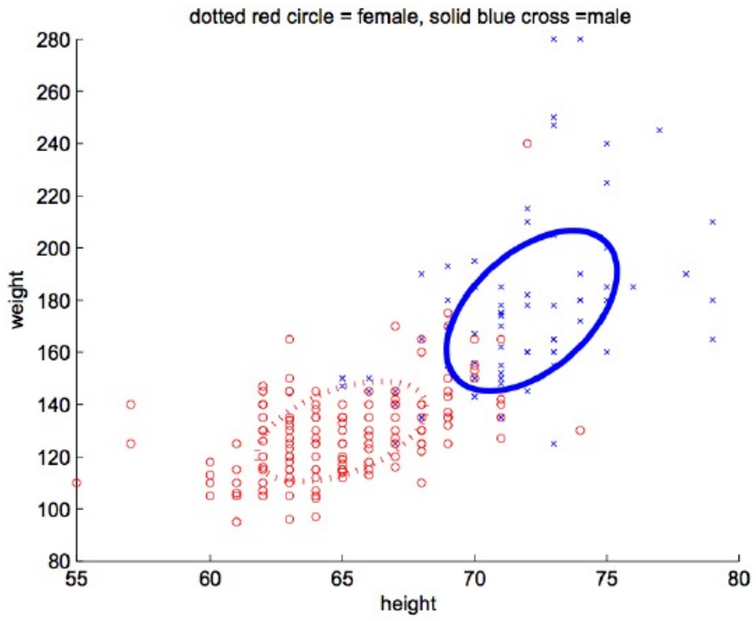


Figure 1: Scatter plot of weight and height for men and women with covariance ellipses superimposed.

Naïve Bayes Review

Therefore we need only fit a univariate Gaussian to each feature variable. Figure 2 illustrates the histograms and Gaussian fits for the individual features from each class. Thus, the training procedure for our Naive Bayes classifier consists of estimating the mean and variance of each variable per class.

For testing, we apply Bayes' rule:

$$p(y = m|x) = \frac{p(x|y = m)p(y = m)}{p(x|y = m)p(y = m) + p(x|y = f)p(y = f)}$$

The posterior probability of the class being cis-male given the features equals the class-conditional likelihood times the prior probability over the normalization term which sums over all the possible cases.

For simplicity, we'll assume equal priors for cis-male and cis-female, i.e.. $p(y = m) = p(y = f) = 0.5$, which tells us that

$$p(y = m|x) = \frac{p(x|y = m)}{p(x|y = m) + p(x|y = f)}$$

which is, in turn, equal to

$$p(y = m|x) = \frac{p(x_h|y = m)p(x_w|y = m)}{p(x_h|y = m)p(x_w|y = m) + p(x_h|y = f)p(x_w|y = f)}$$

by the Naive Bayes independence assumption. During testing you can just plug x_h and x_w into this expression (using the fitted normal densities for each $p(\cdot)$) and you'll get the probability of cis-male. The probability of cis-female would be 1 minus that.

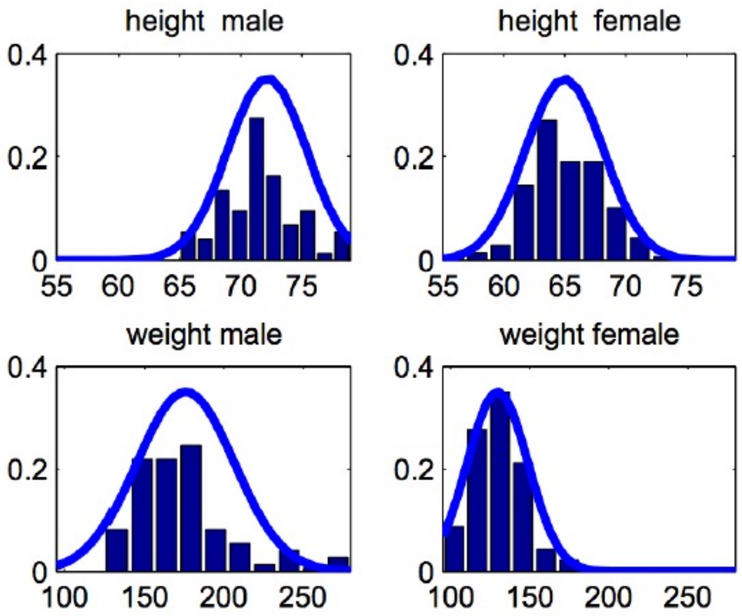


Figure 2: Histograms and univariate Gaussian fits to each feature variable.

Naïve Bayes with Discrete Features

Consider the word-by-document matrix shown in Figure 3. Each row contains a bit vector indicating presence or absence or words in a document. And x represents a "document". If we denote a row of this matrix by the vector x, then $x_d = 1$ means that word d appears in the document at least once. If the documents represent abstracts of scientific papers, then the classes c could represent subjects (biology, chemistry, physics, etc.).

The binary outcome is a Bernoulli random variable. Under the Naive Bayes assumption, the class-conditional densities become a product of Bernoullis with parameter θ_{cd}. So the class-conditional density – the probability of observing a set of words in a document given that the topic is physics, for example – is the product of the probabilities of each individual word being present in that topic.

There are a few problems, though. If you're dealing with scientific papers, the word "abstract" probably appears in every paper. You could argue that it's harmless but some words might throw you off (some word that has nothing to do with a scientific topic that happens to be there, like stop words). For example, if you found a movie has the word "excellent" appeared the most in its review, they probably also came from "the least excellent movie". You can also run into numerical underflow problems.

Recall that Naive Bayes fights the curse of dimensionality because the parameter estimation problem scales linearly with the dimensionality. If you're modeling joint densities without an independence assumption, you have to deal with the covariances (at least) for all possible variables and the probability densities get a lot more complicated.

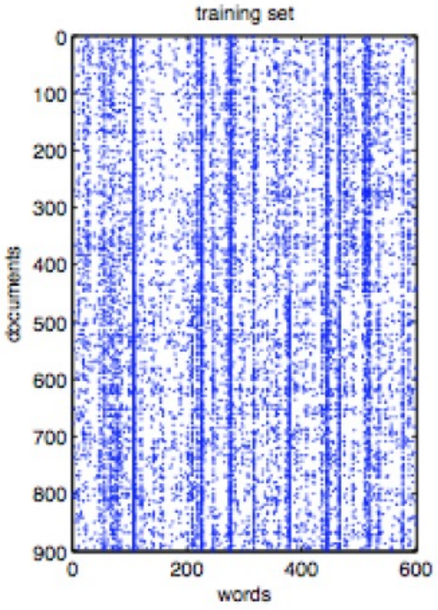
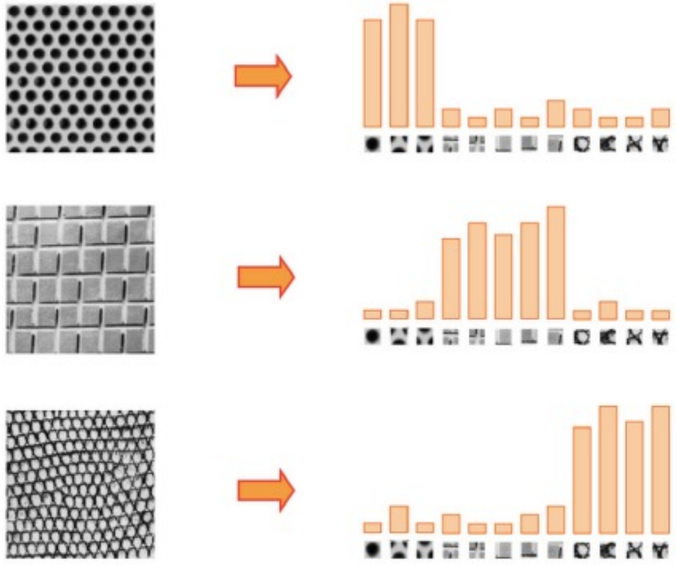


Figure 3: Document-by-Word Matrix. Each row is like a bag-of-words vector (i.e., histogram) thresholded to 0 or 1.

Bag of Words

Origin 1: Texture recognition



Julesz, 1981; Cula & Dana, 2001; Leung & Malik 2001; Mori, Belongie & Malik, 2001; Schmid 2001; Varma & Zisserman, 2002, 2003; Lazebnik, Schmid & Ponce, 2003

Figure 4: Bag-of-words descriptions of texture images using *textons*. Here, the "words" are vector-quantized image patches that constitute a "vocabulary" for images of common textures. [Fergus]

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary Salton & McGill (1983)

Figure 5: Word cloud visualization of bag-of-words descriptors for presidential speeches. [Fergus]

Bag of Words

If we keep the word counts we can apply Naive Bayes with Multinoulli instead of Bernoulli. Figures 4 and 5 show examples of bag-of-words descriptions for texture images and text, respectively.

We use the term "bag" to convey the idea that we're not paying attention to the order of the words (texton matters, not the ordering). Naive Bayes gives us a simple means of training classifiers and estimating per class probabilities for unknown test examples. In BoW, E,F are normalized histogram vectors that containing word frequency. But what if you just want to compare two things? How do we compare them? In the case of classification, there's a well-defined training step per class, requiring collection of labeled examples and estimating parameters to characterize each class. With pairwise comparison, there is no training step, one needs only a measure of similarity or dissimilarity.

Pair wise Similarity - Chi-Squared Test

The chi-squared test measures the dissimilarity between an empirical histogram and a reference pdf through the following expression

$$\chi^2 = \sum_{i=1}^{K} \frac{(F_i - E_i)^2}{E_i}$$

where F_i is the empirical count in bin i, E_i is the theoretical count in bin i, and K is the number of bins. We model the error as a χ^2 random variable with $K-1$ degrees of freedom., i.e., the pdf of the sum of $\mathcal{N}(0,1)$-distributed random variables. This is illustrated for the case of a Gaussian pdf in Fig. 6. The χ^2 test simply adds up the squared error between the measured and theoretical bin counts, normalized by the theoretical bin count. As such, relative differences between the counts play an important role.

Pair wise Similarity - Chi-Squared Test

χ^2 "distance" is a measure of dissimilarity between a pair of histograms. So there's no theoretical reference and we instead have two empirical histograms.

$$\chi^2 = \frac{1}{2}\sum_{i=1}^{K}\frac{(F_i - G_i)^2}{F_i + G_i}$$

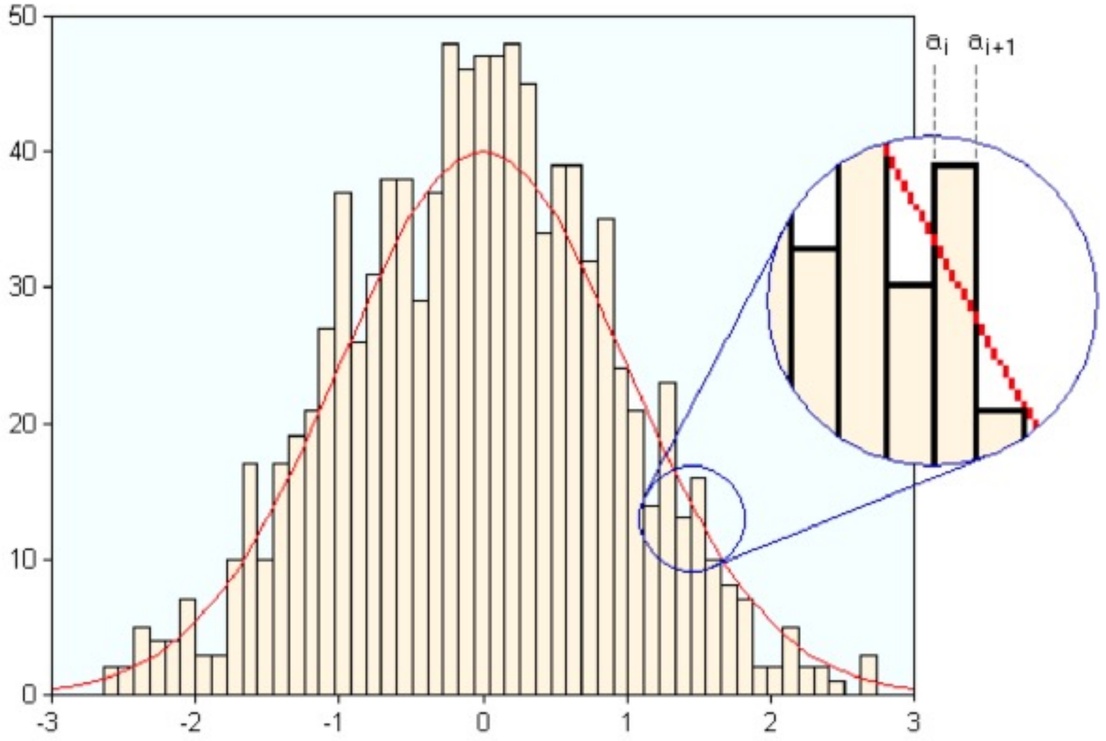
where F_i, G_i are the two empirical histograms. Notice that the χ^2 distance doesn't require any sort of ordering at all provided that the word positions in the two histograms correspond with one another. Another note is that χ^2lies between 0 and 1.

For contrast, consider the L_2^2 or Euclidean distance between F and G:

$$L_2^2 = \sum_{i=1}^{K}(F_i - G_i)^2$$

This looks like the expression for the χ^2 distance without the per-bin normalization term. As described above for the χ^2 test, the presence of the normalization term emphasizes differences with larger relative significance. Specifically, two bin counts that are quite large with a difference of Δ and two bins counts are are rather small with a difference of Δ are counted in the same manner in the L_2 case, whereas the latter gets more weight in the χ^2 case.

Pair wise Similarity - Chi-Squared Test



http://www.vias.org/tmdatanaleng/cc_test_chi-square.html

Figure 6: Illustration of the χ^2 test of fit between an empirical histogram and a reference pdf.

Cluster Analysis

Cluster analysis is an unsupervised approach. The aim of clustering is to partition data into groups such that points in the same group are similar to one another and points in different groups are dissimilar from one another.

There is no "correct" clustering. Nonetheless some solutions are better than others. It all depends on the measure of dissimilarity. "It's the affinities, stupid," that is, defining how two points like each other makes the difference–the affinity function.

An important distinction for clustering is central ($N \times k$ data structure to maintain distances between data points to clusters, as in k-means and k-medoids) vs. pairwise ($N \times N$ data structure to maintain distances between all pairs, as in spectral clustering or graph theoretic).

With central methods, like k-means, you have N data points and $k \ll N$ that are some points of reference or prototypes. You want to associate all data points with these prototypes to break data into k groups. The problem is to figure out where the prototype should be and what data points should belong to each group.

Cluster Analysis

It's kind of easy to see how you might cluster two clumps of normally distributed data points using something like K-means. What do you do if one of the groups is instead an annulus, as in Fig. 7? A human observer will recognize two groups, but central clustering methods won't see this since neither group possesses central, prototype points. Instead, the group membership is propagated from neighbor to neighbor, which we can capture using graph theoretic or pairwise clustering approaches.

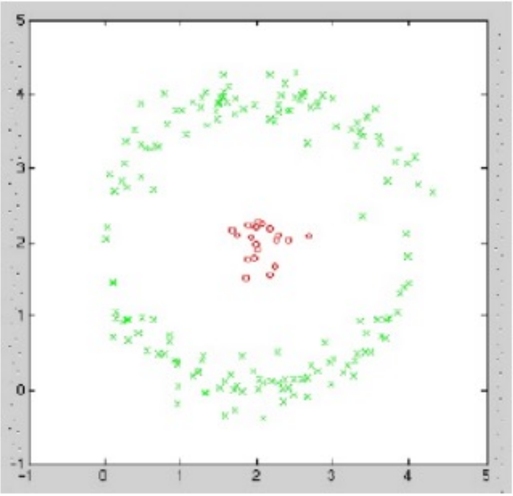


Figure 7: Example of datapoints arranged in two groups (a clump and an annulus) that aren't well suited to K-means clustering.

Cluster Analysis

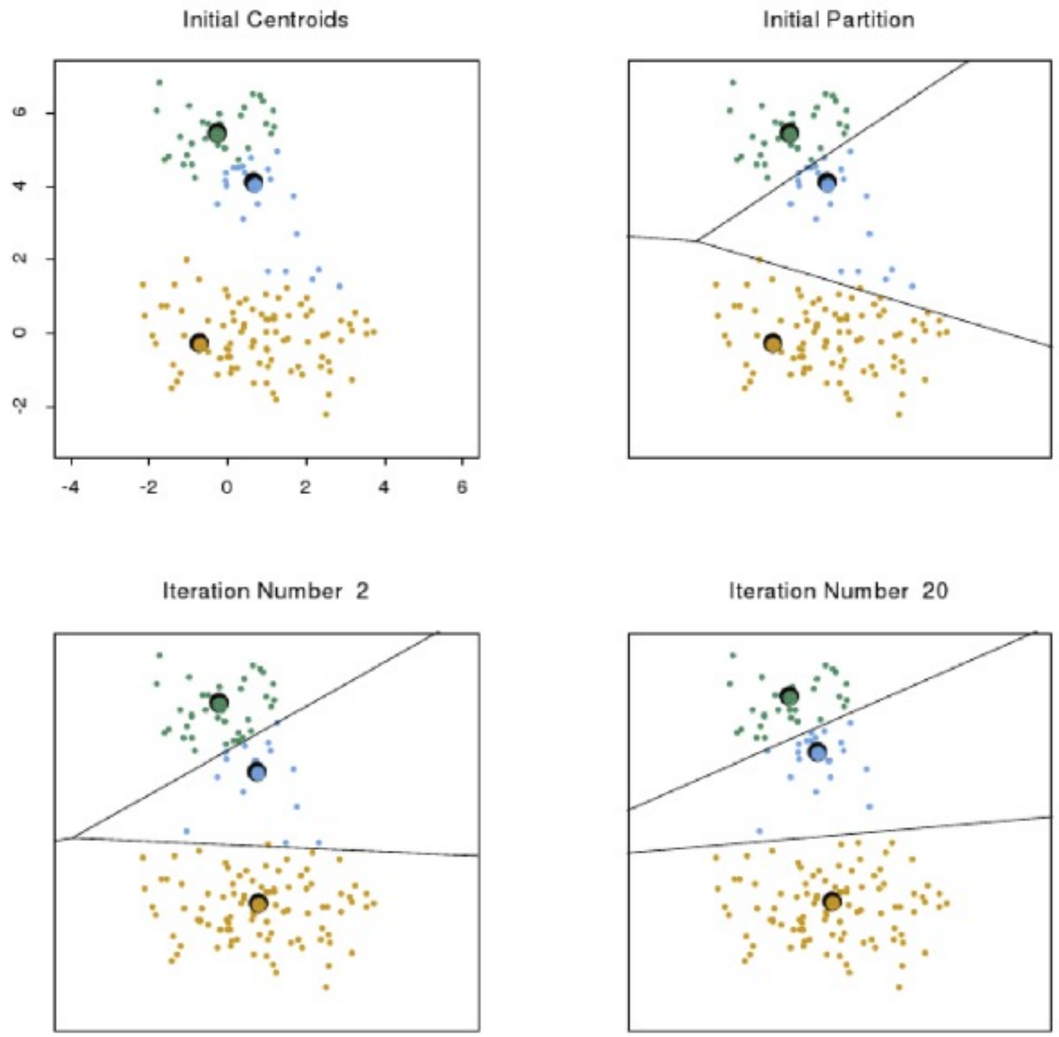


FIGURE 14.6

Figure 8: K-means clustering

Kernel Density Estimation

KERNEL:

1. A device for local weighted averaging, related to the use of kernels in convolution.
2. A generalization of an inner product for non-linear machine learning.

Suppose we have a random sample $x_1, \ldots, x_N$ drawn from pdf (probability density function) $f_X(x)$ (assuming $X \in \mathbb{R}$) and we want to estimate f_X at a new point x_0. Before appealing to kernels, a very simple approach for computing this estimate $\hat{f}_X(x_0)$ would be to use a simple average of the samples within a small neighborhood around x_0 of width λ. This is called *neighbor-based density esimation*:

$$\hat{f}_X(x_0) = \frac{\#x_i \in \mathcal{N}(x_0)}{N\lambda}$$

In analogy to the k-NN based smoothing approach from last lecture, this method will produce a jumpy estimate of the density: if we imagine sliding the neighborhood along the horizontal axis, samples will pop in and out of the averaging interval discontinuously, causing corresponding fluctuations in $\hat{f}_X$.

Parzen Estimate

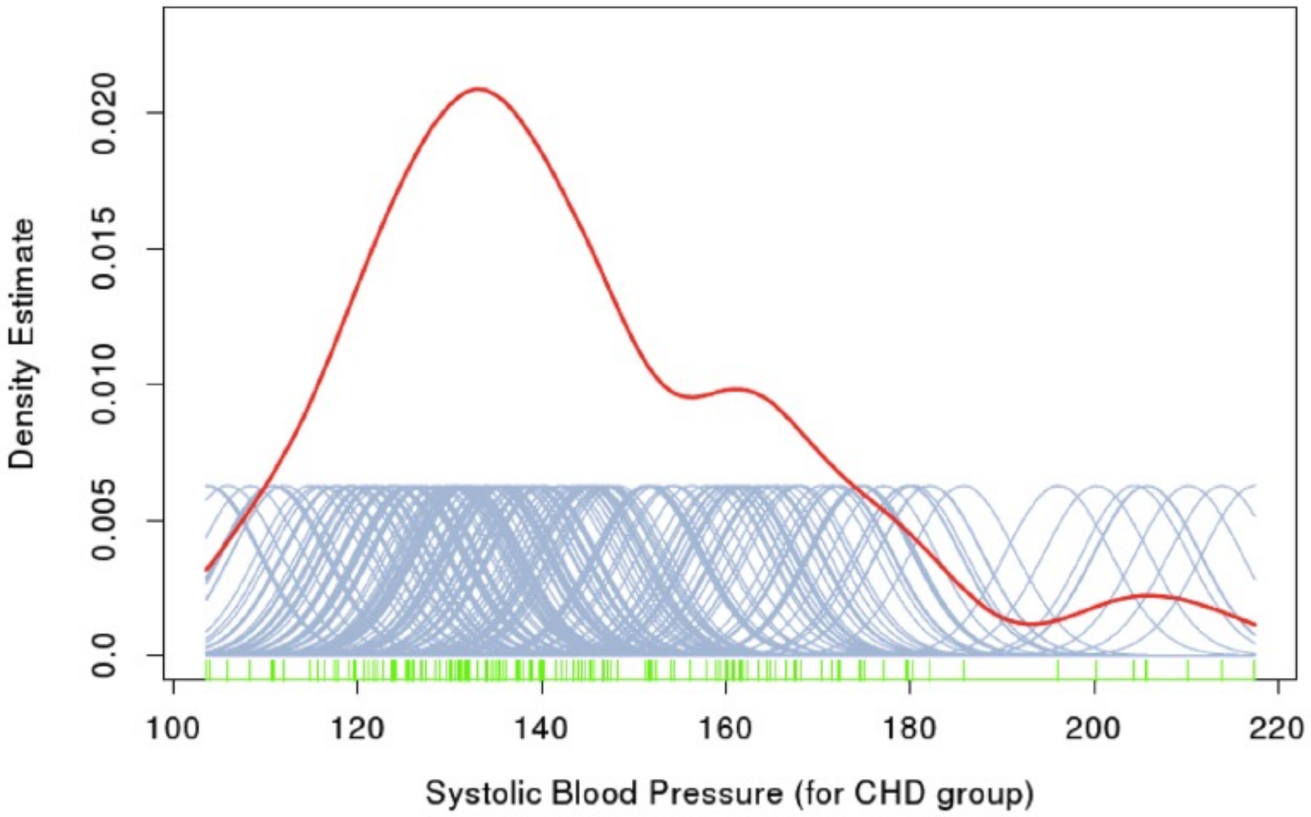


Figure 1: [HTF Figure 6.13] Each green tick represents a single recorded observation. Each tick, which we can think of as a unit impulse function, is replaced with a kernel function – a Gaussian in this case – represented by the blue curves. The kernel density estimate, shown in red, is simply the sum of all these kernels. The red figure adds up all Gaussians and thus there's a bugger bump where the Gaussians are more crowded. As a side note, in order to choose the right width of the Gaussians we need to use cross-validation. (CHD: Coronary Heart Disease)

Parzen Estimate

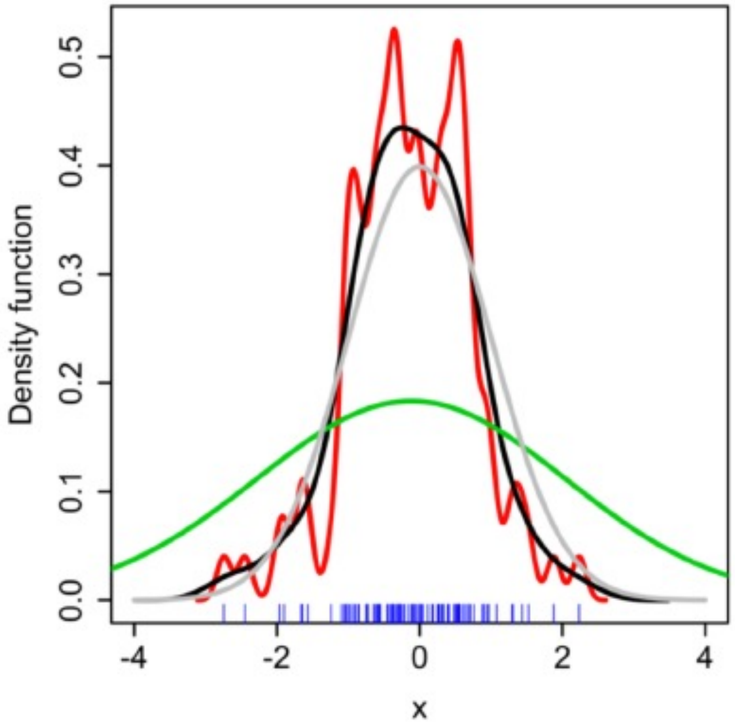


Figure 2: Kernel density estimate (KDE) with different bandwidths of a random sample of 100 points from a standard normal distribution. Grey: true density (standard normal). Red: KDE with $\lambda = 0.05$. Green: KDE with $\lambda = 2$. Black: KDE with $\lambda = 0.337$. [Wikipedia]

Parzen Estimate

The kernel smoothing counterpart to the above method is expressed as follows:

$$\hat{f}_X(x_0) = \frac{1}{N\lambda} \sum_{i=1}^{N} K_\lambda(x_0, x_i)$$

Where $K_\lambda(x_0, x) = \phi(|x - x_0|/\lambda)$ is the kernel of width λ centered around x_0 at a point x. This is also known as a *Parzen estimate* and choices of kernels such as a Gaussian or triangle are known as *Parzen windows*. An example of kernel density estimation using a Gaussian is shown in Figure 1.

The Parzen Estimate is closely related to convolution:

$$\hat{f}_X(x) = \frac{1}{N} \sum_{i=1}^{N} \phi_\lambda(x - x_i) = (\hat{F} * \phi_\lambda)(x)$$

where $\phi_\lambda(\cdot)$ is a Gaussian density and $\hat{F}$ is an impulsive distribution with mass of $1/N$ at each x_i. This is equivalent to adding iid Gaussian noise to each observation x_i.[1] As in the case of bin width selection for histograms, the choice of kernel width can dramatically impact the appearance of the density estimate; see Figure 2. When applying KDE on supervised learning problems, one can use cross validation to choose the best kernel width λ.

KDE based Classification

Suppose we have a J class problem. Fit a KDE $\hat{f}_j(x)$, $j = 1, \ldots, J$ separately for each class and estimate the priors $\hat{\pi}_j$ using the sample proportions. Bayes' rule tells us that the resulting posterior is:

$$\hat{Pr}(G = j | X = x_0) = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_{k=1}^{J} \hat{\pi}_k \hat{f}_k(x_0)}$$

The classifier returns the class $j \in G$ with the highest posterior probability.

In Figure 3 we see how based on the KDE in a 2 class scenario we get a threshold that separates the two classes. The fluctuating, wiggly behavior around 190-200 mm Hg arises from the sparsity of the data in that section.

Very few points fall inside the kernel there and the expression for the posterior becomes unstable; as if it were "hallucinating" data. One way to address this, which we won't be covering, is to use a kernel with adaptive width.

KDE based Classification

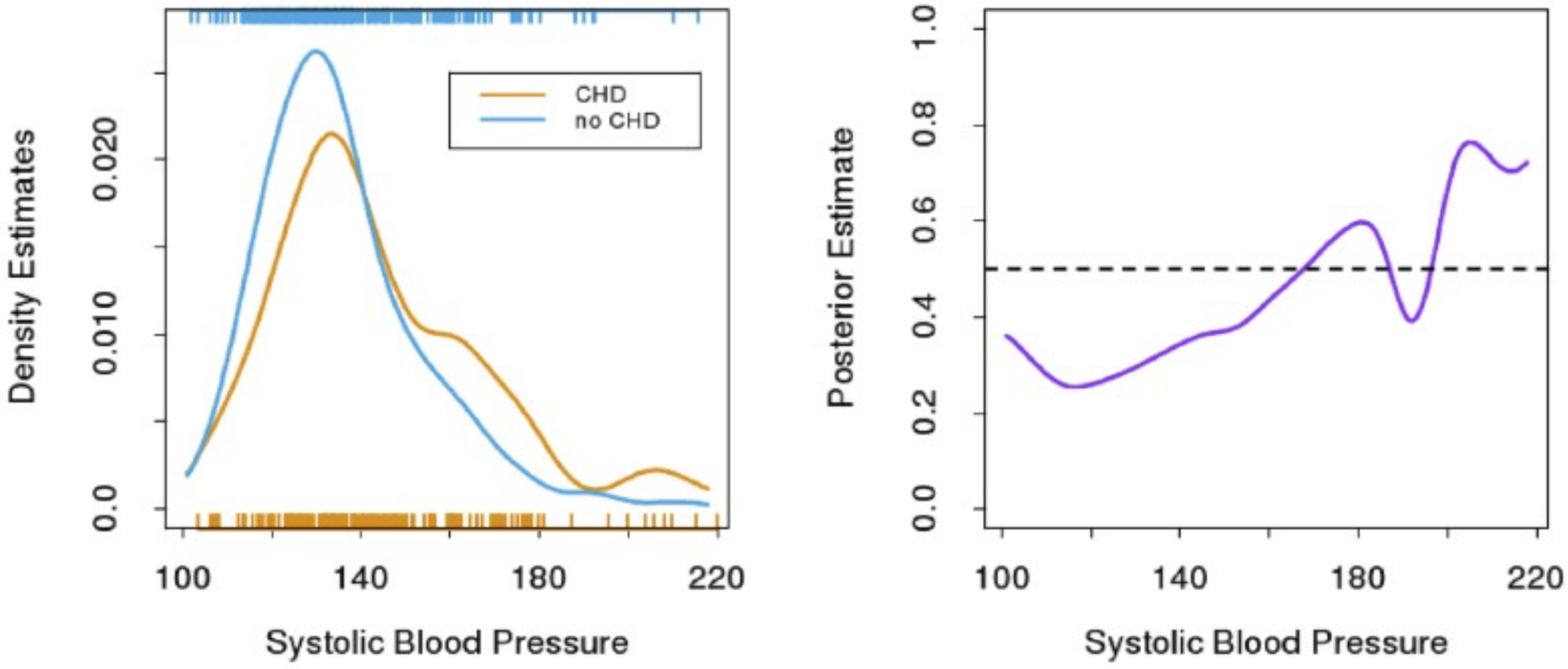


Figure 3: [HTF Figure 6.14] The left graph shows the KDEs for 2 classes and the right graph shows the classification result (where the dashed line represents the decision threshold).

Curse of Dimensionality

All the examples we have seen in this lecture so far were 1 dimensional. In the blood pressure example, we saw that sparsity created a problem, i.e., there weren't enough data points to provide a reliable KDE in a certain subinterval. Similarly, if we were histogramming, we'd have some bins with few or no point counts. When we scale up to higher dimensions empty bins become much more common. When we divide a 1D bin into 10 bins we get 10 bins, but when we divide a 3D bin with the same granularity along each dimension we get 1000 bins. This problem is one manifestation of the *curse of dimensionality* (Bellman, 1961).

Naïve Bayes

A very popular and surprisingly effective way to deal with the curse of dimensionality when dealing with a high dimensional classification problem (where $p \gg 1$) is *Naive Bayes*.
Naive Bayes assumes that given a class $G = j$, the features x_k are independent (hence the "naive" in the name) and ignores the joint probabilities.

$$f_j(X) = \prod_{k=1}^{p} f_{jk}(X_k)$$

This is a very simple case of a generative model. Obviously, this assumption is not very realistic, but it tends to do reasonable things at the class boundary. The consequences of this simplification are as follows:

- Individual class conditional marginal density f_{j_k} can each be estimated using a 1D kernel density estimate.
- If some component X_j is discrete, no problem, we can swap in a histogram for its density estimate.

Naïve Bayes

Naive Bayes handles curse of dimensionality by putting each feature in its own "silo." With its focus on the behavior near the decision boundary, Naive Bayes often performs better than far more sophisticated alternatives.

[insert example here from Murphy – gender classification from height & weight]

The log-odds transformation in this case (using class J as the base) is given by:

$$\log \frac{Pr(G=l|X)}{Pr(G=J|X)} = \log \frac{\pi_l f_l(X)}{\pi_J f_J(X)} = \log \frac{\pi_l \prod_{k=1}^{p} f_{lk}(X_k)}{\pi_J \prod_{k=1}^{p} f_{Jk}(X_k)}$$

$$\Downarrow$$

$$\log \frac{Pr(G=l|X)}{Pr(G=J|X)} = \log \frac{\pi_l}{\pi_J} + \sum_{k=1}^{p} \log \frac{f_{lk}(X_k)}{f_{Jk}(X_k)}$$

If we let $\log \frac{f_{lk}(X_k)}{f_{Jk}(X_k)} = g_{lk}(x_k)$ and $\frac{\pi_l}{\pi_J} = \alpha_l$ we get:

$$\log \frac{Pr(G=l|X)}{Pr(G=J|X)} = \alpha_l + \sum_{k=1}^{p} g_{lk}(x_k)$$

Incidentally this has the same form as the "generalized additive model" but it is fit differently. This is analogous to Logistic Regression and Linear Discriminant Analysis having the same functional form but different modeling assumptions and fitting procedures. Note that g_{l_k} in the above equation is a nonlinear function.