Clustering and Classification

Prof. Gustavo Nonato

NYU / CUSP - GX 5006

February 22, 2017

Learning Strategies

Clustering learns a model which groups observations that are "similar" to each other. The similarity criterion is predefined and application dependent. Input data is typically not annotated - unsupervised task.

Learning Strategies

Clustering learns a model which groups observations that are "similar" to each other. The similarity criterion is predefined and application dependent. Input data is typically not annotated - unsupervised task.

Classification learns a model which maps a vector of attributes into one of several classes, relying on several input-output examples. Input data is typically annotated (labelled) - supervised task.

Learning Strategies

Clustering learns a model which groups observations that are "similar" to each other. The similarity criterion is predefined and application dependent. Input data is typically not annotated - unsupervised task.

Classification learns a model which maps a vector of attributes into one of several classes, relying on several input-output examples. Input data is typically annotated (labelled) - supervised task.

Situations were data is partially annotated are also common (semi-supervised tasks, more often in classification).

Clustering

Introduction

There are many clustering techniques:

Introduction

There are many clustering techniques:

- Hierarchical
  - Single Link
  - ⋮
- Partitional
  - K-means
  - Mixture Resolving
  - Spectral Clustering
  - Density-based
  - ⋮

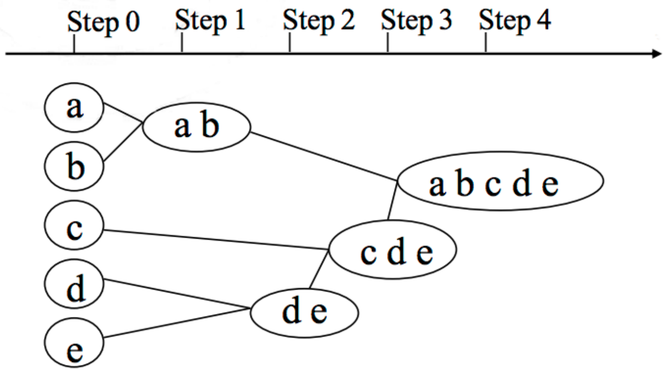
Introduction

There are many clustering techniques:

- Hierarchical
  - Single Link
  - $\vdots$
- Partitional
  - K-means
  - Mixture Resolving
  - Spectral Clustering (later in this course)
  - Density-based
  - $\vdots$

Hierarchical Clustering

Hierarchical Clustering



Hierarchical Clustering: Agglomerative Method

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster
3. Update distances between clusters

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster
3. Update distances between clusters
4. Repeat steps 1-3 until a single cluster (or the desired number of clusters) is obtained

Hierarchical Clustering: Agglomerative Method

1. Start with n clusters (one for each instance)
2. Find the most similar pair of clusters C_i and C_j and merge them into a single cluster
3. Update distances between clusters
4. Repeat steps 1-3 until a single cluster (or the desired number of clusters) is obtained

Step 3 can assume different forms:

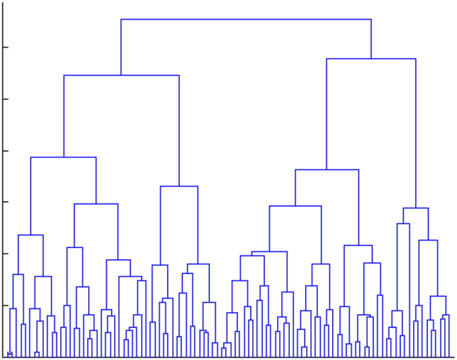
$$d(C_a, C_b) = \min_{i \in C_a, j \in C_b}]\{d(i,j)\} \quad \text{Single Link}$$

$$d(C_a, C_b) = \max_{i \in C_a, j \in C_b} \{d(i,j)\} \quad \text{Complete Link}$$

$$d(C_a, C_b) = \frac{1}{n_a n_b} \sum_{i \in C_a, j \in C_b} \{d(i,j)\} \quad \text{Average Link}$$

Hierarchical Clustering

Dendogram



K-means

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

K-means

Suppose a data set $\{\mathbf{x}_1, \dots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \, \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \boxed{r_{ij}} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

$r_{ij} = 1$ if $\mathbf{x}_i \in cluster_j$, 0 otherwise, and

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \boxed{r_{ij}} \|\mathbf{x}_i - \boxed{\boldsymbol{\mu}_j}\|^2$$

$r_{ij} = 1$ if $\mathbf{x}_i \in \ cluster_j$, 0 otherwise, and
$\boldsymbol{\mu}_j$ is a "prototype" associate to $cluster_j$.

K-means

Suppose a data set $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, $\mathbf{x}_i \in \mathbb{R}^d$ and $k \geq 2$ the number of cluster.

The idea is to partition the data set into k clusters such that inter-cluster distances are smaller than distances between points in different clusters.

Mathematically this idea can be state as:

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} \boxed{r_{ij}} \|\mathbf{x}_i - \boxed{\boldsymbol{\mu}_j}\|^2$$

$r_{ij} = 1$ if $\mathbf{x}_i \in \ cluster_j$, 0 otherwise, and
$\boldsymbol{\mu}_j$ is a “prototype” associate to $cluster_j$.

The goal is to find $\{r_{ij}\}$ and $\{\boldsymbol{\mu}_j\}$ so as to minimize J.

K-means

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

K-means

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

If the $\{\boldsymbol{\mu}_j\}$ are fixed then the minimum of J is reached when

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_s \|\mathbf{x}_i - \boldsymbol{\mu}_s\|^2 \\ 0 & \text{otherwise} \end{cases}$$

K-means

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \| \mathbf{x}_i - \boldsymbol{\mu}_j \|^2$$

If the $\{\boldsymbol{\mu}_j\}$ are fixed then the minimum of J is reached when

$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_s \|\mathbf{x}_i - \boldsymbol{\mu}_s\|^2 \\ 0 & \text{otherwise} \end{cases}$$

If $\{r_{ij}\}$ is fixed then the minimum can be obtained by setting the derivative of J w.r.t. $\boldsymbol{\mu}_j$ to zero, resulting in

$$\boldsymbol{\mu}_j = \frac{\sum_i r_{ij} \mathbf{x}_j}{\sum_i r_{ij}}$$

K-means

$$J = \sum_{i=1}^{n} \sum_{j=1}^{k} r_{ij} \|\mathbf{x}_i - \boldsymbol{\mu}_j\|^2$$

If the $\{\boldsymbol{\mu}_j\}$ are fixed then the minimum of J is reached when

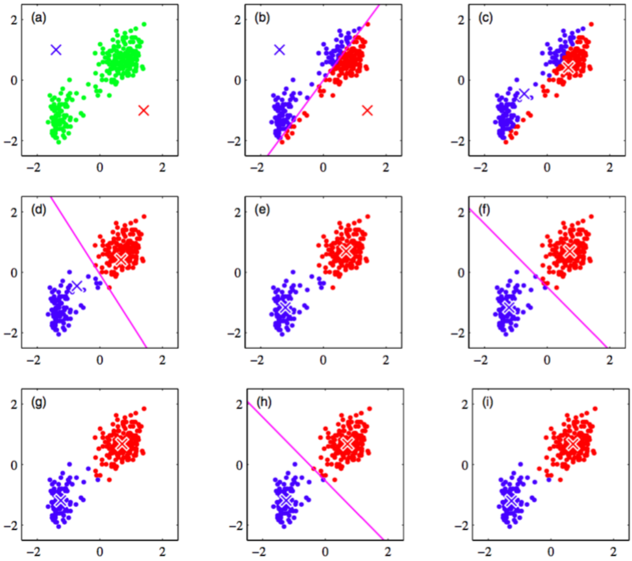
$$r_{ij} = \begin{cases} 1 & \text{if } j = \arg\min_s \|\mathbf{x}_i - \boldsymbol{\mu}_s\|^2 \\ 0 & \text{otherwise} \end{cases}$$

If $\{r_{ij}\}$ is fixed then the minimum can be obtained by setting the derivative of J w.r.t. $\boldsymbol{\mu}_j$ to zero, resulting in

$$\boldsymbol{\mu}_j = \frac{\sum_i r_{ij}\mathbf{x}_j}{\sum_i r_{ij}}$$

$\boldsymbol{\mu}_j$ is simply the average of the $\mathbf{x}_i \in cluster_j$.

K-means



(figure extracted from Bishop's book)

K-means

- K-means always converges to a local minimum

K-means

- K-means always converges to a local minimum
- There are versions able to handle dynamic data

K-means

- K-means always converges to a local minimum
- There are versions able to handle dynamic data
- The algorithm can be adapted to deal with “distances” (dissimilarities) other than Euclidean distance (see chapter 9 of Bishop’s book).

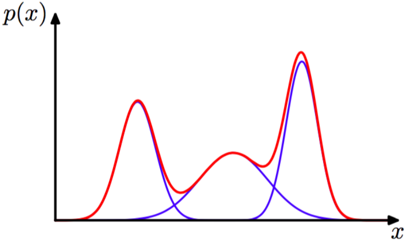
Mixture Resolving

Mixture Resolving

A Gaussian mixture distribution is given by

$$p(\mathbf{x}) = \sum_{i=1}^{k} c_i N(\mathbf{x}|\boldsymbol{\mu}_i, \sigma_i)$$

where $c_i \geq 0$ and $\sum_i c_i = 1$.



Mixture Resolving

Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a mixture solving algorithm aims to find parameters c_i, $\boldsymbol{\mu}_i$, Σ_i and a "responsibility" (membership) function γ_{ij} so as to maximize the likelihood

$$p(\mathbf{X}|\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} c_j N(\mathbf{x}_i | \boldsymbol{\mu}_j, \Sigma_j) \right)$$

Mixture Resolving

Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a mixture solving algorithm aims to find parameters c_i, $\boldsymbol{\mu}_i$, Σ_i and a "responsibility" (membership) function γ_{ij} so as to maximize the likelihood

$$p(\mathbf{X}|\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} c_j N(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j) \right)$$

the γ_{ij} should be such that $N(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j) > N(\mathbf{x}_i|\boldsymbol{\mu}_s, \Sigma_s)$, $j \neq s$.

Mixture Resolving

Given a data set $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$, a mixture solving algorithm aims to find parameters c_i, $\boldsymbol{\mu}_i$, Σ_i and a "responsibility" (membership) function γ_{ij} so as to maximize the likelihood

$$p(\mathbf{X}|\boldsymbol{c}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{i=1}^{n} \left(\sum_{j=1}^{k} c_j N(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j) \right)$$

the γ_{ij} should be such that $N(\mathbf{x}_i|\boldsymbol{\mu}_j, \Sigma_j) > N(\mathbf{x}_i|\boldsymbol{\mu}_s, \Sigma_s)$, $j \neq s$.

Such optimization can be accomplish via an Expectation Maximization strategy (chapter 9 of Bishop's book).

Mixture Resolving

The optimization works as follows:

Mixture Resolving

The optimization works as follows:

1. (E-step) Fixing c_i, $\boldsymbol{\mu}_i$, Σ_i we can compute the probability of a Gaussian with parameters $\boldsymbol{\mu}_j, \Sigma_j$ generates to point $\mathbf{x}_i$ as:

$$\gamma_{ij} = \frac{c_j N(\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{s=1}^{k} c_s N(\boldsymbol{\mu}_s, \Sigma_s)}$$

The optimization works as follows:

Mixture Resolving

1. (E-step) Fixing c_i, $\boldsymbol{\mu}_i$, Σ_i we can compute the probability of a Gaussian with parameters $\boldsymbol{\mu}_j, \Sigma_j$ generates to point $\mathbf{x}_i$ as:

$$\gamma_{ij} = \frac{c_j N(\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{s=1}^{k} c_s N(\boldsymbol{\mu}_s, \Sigma_s)}$$

2. (M-step) Fixing γ_{ij} the parameters can be obtained by setting to zero the derivative of the likelihood, resulting in:

$$\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} \mathbf{x}_i$$

$$\hat{\Sigma}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^\top$$

$$\hat{c}_j = \frac{\sum_{i=1}^{n} \gamma_{ij}}{n}$$

The optimization works as follows:

Mixture Resolving

1. (E-step) Fixing c_i, $\boldsymbol{\mu}_i$, Σ_i we can compute the probability of a Gaussian with parameters $\boldsymbol{\mu}_j, \Sigma_j$ generates to point $\mathbf{x}_i$ as:

$$\gamma_{ij} = \frac{c_j N(\boldsymbol{\mu}_j, \Sigma_j)}{\sum_{s=1}^{k} c_s N(\boldsymbol{\mu}_s, \Sigma_s)}$$

2. (M-step) Fixing γ_{ij} the parameters can be obtained by setting to zero the derivative of the likelihood, resulting in:

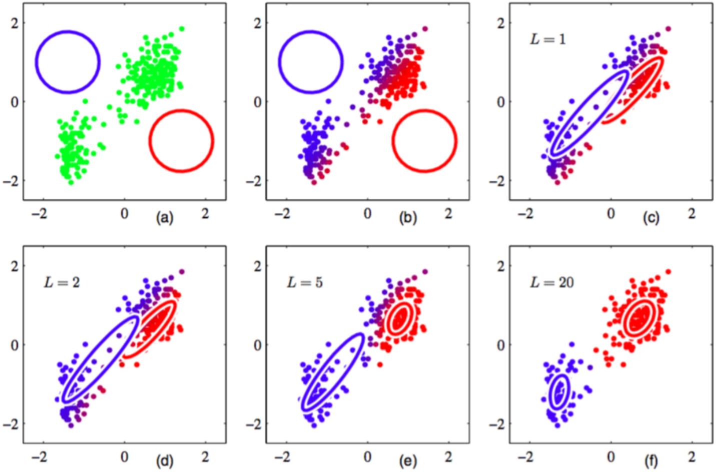
$$\hat{\boldsymbol{\mu}}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} \mathbf{x}_i$$

$$\hat{\Sigma}_j = \frac{1}{N_j} \sum_{i=1}^{n} \gamma_{ij} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_j)^\top$$

$$\hat{c}_j = \frac{\sum_{i=1}^{n} \gamma_{ij}}{n}$$

Steps E and M are repeated until convergence.

Mixture Resolving



(figure extracted from Bishop's book)

Mixture Resolving

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

Mixture Resolving

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)

Mixture Resolving

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)
- $\gamma_{ij} \rightarrow r_{ij}$ when $\epsilon \rightarrow 0$.

Mixture Resolving

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)
- $\gamma_{ij} \to r_{ij}$ when $\epsilon \to 0$.

Therefore, K-means tends to generate spherically shaped clusters !!

Mixture Resolving

It can be shown that the K-means algorithm is a particular case of the Mixture Resolving when

- variances $\Sigma_i = \epsilon \mathbf{I}$ for all Gaussians (spherically shaped Gaussians)
- $\gamma_{ij} \rightarrow r_{ij}$ when $\epsilon \rightarrow 0$.

Therefore, K-means tends to generate spherically shaped clusters !!
The convergence of Mixture Resolving is slower.
K-means is typically used to set initial conditions !!

Classification

Introduction

Lots of classification techniques:

Introduction

Lots of classification techniques:

- Logistic Regression
- Bayesian Classifier
- SVM
- Neural Networks
- $\vdots$

Introduction

Lots of classification techniques:

- Logistic Regression
- Bayesian Classifier
- SVM (next class)
- Neural Networks
- ⋮

Bayes Classifier

Bayes Classifier

Given a sample $\mathbf{x}$ and k classes, the classification problem accounts for learning a model, based on observed data, that assigns $\mathbf{x}$ to a particular class (or computes the probability of $\mathbf{x}$ belongs to each class).

Bayes Classifier

Given a sample $\mathbf{x}$ and k classes, the classification problem accounts for learning a model, based on observed data, that assigns $\mathbf{x}$ to a particular class (or computes the probability of $\mathbf{x}$ belongs to each class).

From Bayes' rule

Bayes Classifier

Given a sample $\mathbf{x}$ and k classes, the classification problem accounts for learning a model, based on observed data, that assigns $\mathbf{x}$ to a particular class (or computes the probability of $\mathbf{x}$ belongs to each class).

From Bayes' rule

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}|y) \;\; p(y)}{\sum_y p(\mathbf{x}|y)p(y)}$$

Bayes Classifier

Given a sample $\mathbf{x}$ and k classes, the classification problem accounts for learning a model, based on observed data, that assigns $\mathbf{x}$ to a particular class (or computes the probability of $\mathbf{x}$ belongs to each class).

From Bayes' rule

$$p(y|\mathbf{x}) = \frac{\boxed{p(\mathbf{x}|y)}\,\boxed{p(y)}}{\sum_y p(\mathbf{x}|y)p(y)}$$

all the trick is in playing with those distributions.

Bayes Classifier

Given a sample $\mathbf{x}$ and k classes, the classification problem accounts for learning a model, based on observed data, that assigns $\mathbf{x}$ to a particular class (or computes the probability of $\mathbf{x}$ belongs to each class).

From Bayes' rule

$$p(y|\mathbf{x}) = \frac{\boxed{p(\mathbf{x}|y)}\,\boxed{p(y)}}{\sum_y p(\mathbf{x}|y)p(y)}$$

all the trick is in playing with those distributions.
$\mathbf{x}$ should be assigned to the class y that satisfies

$$\arg\max_y \log(p(y|\mathbf{x}))$$

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

- $p(y)$ comes from a density distribution or it is known for each class

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

- $p(y)$ comes from a density distribution or it is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, where the covariance matrix is the same for all classes

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

- $p(y)$ comes from a density distribution or it is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, where the covariance matrix is the same for all classes

For two classes, that is $y = \{1, 2\}$, the decision boundary is given by:

$$0 = \log \frac{p(y = 1|\mathbf{x})}{p(y = 2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y = 1)p(y = 1)}{p(\mathbf{x}|y = 2)p(y = 2)}$$

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

- $p(y)$ comes from a density distribution or it is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, where the covariance matrix is the same for all classes

For two classes, that is $y = \{1, 2\}$, the decision boundary is given by:

$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

$$= \boldsymbol{\mu}_1^\top \Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p(y=1)) - \log(p(y=2))$$

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

- $p(y)$ comes from a density distribution or it is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, where the covariance matrix is the same for all classes

For two classes, that is $y = \{1, 2\}$, the decision boundary is given by:

$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

$$= \boldsymbol{\mu}_1^\top \Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p(y=1)) - \log(p(y=2))$$

$$\mathbf{w}^\top \mathbf{x} + w_o$$

Bayes Classifier

Given $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)\}$, $\mathbf{x}_i \in \mathbb{R}^d$, $y_i \in \{1, \ldots, k\}$, lets assume that

- $p(y)$ comes from a density distribution or it is known for each class
- $p(\mathbf{x}|y) = N(\boldsymbol{\mu}_y, \Sigma)$, where the covariance matrix is the same for all classes

For two classes, that is $y = \{1, 2\}$, the decision boundary is given by:

$$0 = \log \frac{p(y=1|\mathbf{x})}{p(y=2|\mathbf{x})} = \log \frac{p(\mathbf{x}|y=1)p(y=1)}{p(\mathbf{x}|y=2)p(y=2)}$$

$$= \boldsymbol{\mu}_1^\top \Sigma^{-1}\mathbf{x} - \boldsymbol{\mu}_2^\top \Sigma^{-1}\mathbf{x} - \frac{1}{2}\boldsymbol{\mu}_1^\top \Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^\top \Sigma^{-1}\boldsymbol{\mu}_2 + \log(p(y=1)) - \log(p(y=2))$$

$$\mathbf{w}^\top \mathbf{x} + w_o$$

Thus, assuming a single covariance matrix for all Gaussians, the decision boundary is linear

Bayes Classifier

In order to make the decision based on

$$p(y|\mathbf{x})$$

the parameters $p(y), \boldsymbol{\mu}_y$, and Σ must be estimated.

Bayes Classifier

In order to make the decision based on

$$p(y|\mathbf{x})$$

the parameters $p(y), \boldsymbol{\mu}_y$, and Σ must be estimated.

The *Linear Discriminant Analysis* (LDA) algorithm simply estimates the parameters from the training data as (approximates the MLE solution):

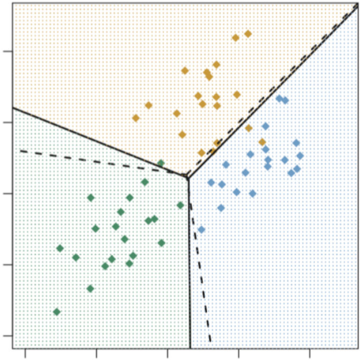
$$\hat{p}(y) = \frac{n_y}{n}$$

$$\hat{\boldsymbol{\mu}}_y = \frac{1}{n_y} \sum_{\mathbf{x}_i \in y} \mathbf{x}_i$$

$$\hat{\Sigma} = \frac{1}{n-k} \sum_{y} \sum_{\mathbf{x}_i \in y} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_y)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_y)^\top$$

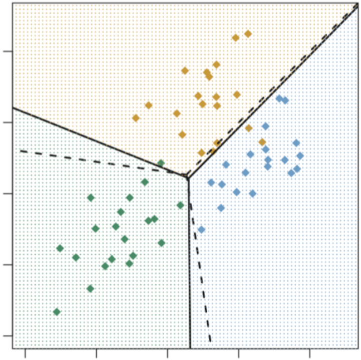
Bayes Classifier

Since the covariance is constant for all classes, LDA has a linear decision boundary



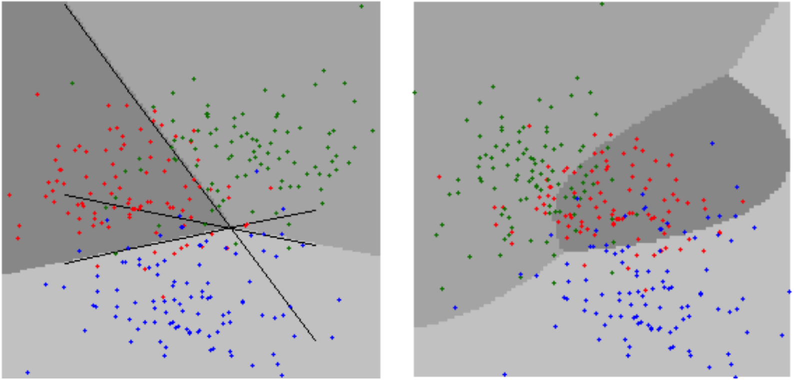
Bayes Classifier

Since the covariance is constant for all classes, LDA has a linear decision boundary



This is one of the main limitations when using a single covariance matrix. Non-linear boundaries can be obtained solving the MLE problem with distinct covariance matrices for each class.

Bayes Classifier



Naive Bayes Classifier

Naive Bayes Classifier

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Naive Bayes Classifier

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

the attributes of each data instances are independent (notice x_j is not bold, it is a scalar representing the j-th attribute of $\mathbf{x}$).

Naive Bayes Classifier

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

the attributes of each data instances are independent (notice x_j is not bold, it is a scalar representing the j-th attribute of $\mathbf{x}$).

Therefore,

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)\, p(y) = p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier

Naive Bayes assume a joint density distribution for

$$p(\mathbf{x}, y) = p(\mathbf{x}|y)p(y)$$

Recalling that $\mathbf{x} \in \mathbb{R}^d$, the Naive Bayes classifier assumes that

$$p(\mathbf{x}|y) \sim \prod_{j=1}^{d} p(x_j|y)$$

the attributes of each data instances are independent (notice x_j is not bold, it is a scalar representing the j-th attribute of $\mathbf{x}$).

Therefore,

$$p(y|\mathbf{x}) \propto p(\mathbf{x}|y)\, p(y) = p(y) \prod_{j=1}^{d} p(x_j|y)$$

The class y to be assigned to $\mathbf{x}$ is the one satisfying

$$\arg\min_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier: Discrete Case

If we have SUFFICIENT training data given as a table *attribute* $\times$ *class*, the Naive Bayes can be directly applied

Naive Bayes Classifier: Discrete Case

If we have SUFFICIENT training data given as a table *attribute* $\times$ *class*, the Naive Bayes can be directly applied

x_i	age	income	job status	gender	class
1	20-30	medium	manager	male	ok
2	40-50	high	engineer	female	rich
3	20-30	medium	student	male	ok
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
n	60-70	medium	retired	male	poor

Naive Bayes Classifier: Discrete Case

$p(poor) = \frac{\#poor}{n}$, $p(ok) = \frac{\#ok}{n}$, $p(rich) = \frac{\#rich}{n}$
$p(20-30|poor) = \frac{\#20-30 \in poor}{\#poor}$
$p(30-40|ok) = \frac{\#30-40 \in ok}{\#ok}$
$\vdots$

Naive Bayes Classifier: Discrete Case

$p(poor) = \frac{\#poor}{n}, p(ok) = \frac{\#ok}{n}, p(rich) = \frac{\#rich}{n}$
$p(20-30|poor) = \frac{\#20-30 \in poor}{\#poor}$
$p(30-40|ok) = \frac{\#30-40 \in ok}{\#ok}$
$\vdots$
$p(ok|(40-50, high, student, male)) = p(ok) \cdot (p(40-50|ok) \cdot p(high|ok) \cdot p(student|ok) \cdot p(male|ok))$
$p(poor|(40-50, high, student, male)) = \cdots$
$p(rich|(40-50, high, student, male)) = \cdots$

Naive Bayes Classifier: Discrete Case

$p(poor) = \frac{\#poor}{n}, p(ok) = \frac{\#ok}{n}, p(rich) = \frac{\#rich}{n}$
$p(20-30|poor) = \frac{\#20-30 \in poor}{\#poor}$
$p(30-40|ok) = \frac{\#30-40 \in ok}{\#ok}$
$\vdots$
$p(ok|(40-50, high, student, male)) = p(ok) \cdot (p(40-50|ok) \cdot p(high|ok) \cdot p(student|ok) \cdot p(male|ok))$
$p(poor|(40-50, high, student, male)) = \cdots$
$p(rich|(40-50, high, student, male)) = \cdots$

The class assigned to an instance, say $(40-50, high, student, male)$, is the one resulting in largest probability.

Naive Bayes Classifier: Discrete Case

$p(poor) = \frac{\#poor}{n}$, $p(ok) = \frac{\#ok}{n}$,$p(rich) = \frac{\#rich}{n}$
$p(20-30|poor) = \frac{\#20-30 \in poor}{\#poor}$
$p(30-40|ok) = \frac{\#30-40 \in ok}{\#ok}$
$\vdots$
$p(ok|(40-50, high, student, male)) = p(ok) \cdot (p(40-50|ok) \cdot p(high|ok) \cdot p(student|ok) \cdot p(male|ok))$
$p(poor|(40-50, high, student, male)) = \cdots$
$p(rich|(40-50, high, student, male)) = \cdots$

The class assigned to an instance, say $(40-50, high, student, male)$, is the one resulting in largest probability.

$$\arg\min_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

Laplace correction (add 1 in each count) makes possible to deal with zero conditionals.

Naive Bayes Classifier: Gaussian

$$\arg\min_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier: Gaussian

$$\arg\min_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

$p(y) = \frac{n_y}{n}$ (Bernoulli)

Naive Bayes Classifier: Gaussian

$$\arg\min_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

$p(y) = \frac{n_y}{n}$ (Bernoulli)

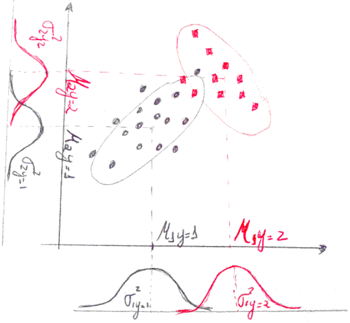
$p(x_j|y) = N(\mu_{jy}, \sigma_{jy}^2)$

Naive Bayes Classifier: Gaussian

$$\arg\min_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

$p(y) = \frac{n_y}{n}$ (Bernoulli)

$p(x_j|y) = N(\mu_{jy}, \sigma^2_{jy})$
(μ_{jy} is the mean of a Gaussian for attribute j in class y)

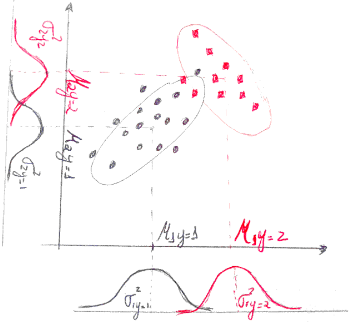


Naive Bayes Classifier: Gaussian

$$\arg\min_y p(y) \prod_{j=1}^{d} p(x_j|y)$$

$p(y) = \frac{n_y}{n}$ (Bernoulli)

$p(x_j|y) = N(\mu_{jy}, \sigma^2_{jy})$
(μ_{jy} is the mean of a Gaussian for attribute j in class y)



The ser of parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ can be obtained by maximizing the likelihood function,

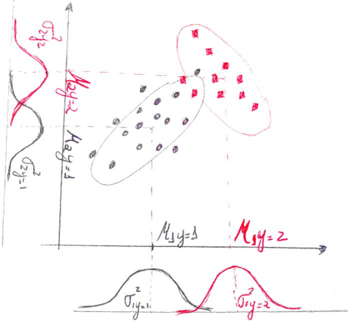
$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{y} p(y) \prod_{\mathbf{x} \in y} \prod_{j=1}^{d} p(x_j|y)$$

Naive Bayes Classifier: Gaussian

$$\arg\min_{y} p(y) \prod_{j=1}^{d} p(x_j|y)$$

$p(y) = \frac{n_y}{n}$ (Bernoulli)

$p(x_j|y) = N(\mu_{jy}, \sigma^2_{jy})$
(μ_{jy} is the mean of a Gaussian for attribute j in class y)



The ser of parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}$ can be obtained by maximizing the likelihood function,

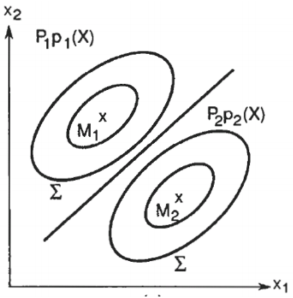
$$L(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \prod_{y} p(y) \prod_{\mathbf{x} \in y} \prod_{j=1}^{d} p(x_j|y)$$

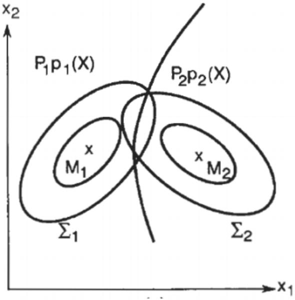
which results in:

$$\hat{\mu}_{jy} = \frac{1}{n_y} \sum_{\mathbf{x} \in y} x_j \quad \hat{\sigma}^2_{iy} = \frac{1}{n_y} \sum_{\mathbf{x} \in y} (x_j - \hat{\mu}_{jy})^2$$

Naive Bayes Classifier: Gaussian

Decision Boundary





Logistic Regression

Logistic Regression

Bayes and Naive Bayes classifier assume distributions for $p(\mathbf{x}, y)$, or equivalentrly to $p(\mathbf{x}|y)$ and $p(y)$ (*generative models*).

Logistic Regression

Bayes and Naive Bayes classifier assume distributions for $p(\mathbf{x}, y)$, or equivalentrly to $p(\mathbf{x}|y)$ and $p(y)$ (*generative models*).

Logistic regression, assumes a density distribution to $p(y|\mathbf{x})$ (*discriminative model*).

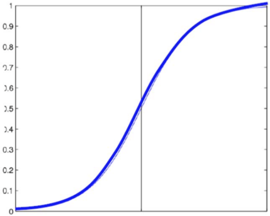
Logistic Regression

Bayes and Naive Bayes classifier assume distributions for $p(\mathbf{x}, y)$, or equivalentrly to $p(\mathbf{x}|y)$ and $p(y)$ (*generative models*).

Logistic regression, assumes a density distribution to $p(y|\mathbf{x})$ (*discriminative model*).

Considering two classes $y \in \{0, 1\}$, the logistic regression assumption is:

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta_1}))}$$



$\beta_0 + \mathbf{x}^\top \boldsymbol{\beta_1} \geq 0$ means $\mathbf{x}$ to class 1 and to class 0 otherwise (linear decision boundary).

Logistic Regression

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta_1}))}$$

Parameters β_0 and $\boldsymbol{\beta}_1$ can be obtained by MLE.

Logistic Regression

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta_1}))}$$

Parameters β_0 and $\boldsymbol{\beta}_1$ can be obtained by MLE.

The likelihood function is given by:

$$L(\beta_0, \boldsymbol{\beta}_1) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i)$$

Logistic Regression

$$p(y|\mathbf{x}) = \frac{1}{1 + \exp(-(\beta_0 + \mathbf{x}^\top \boldsymbol{\beta_1}))}$$

Parameters β_0 and $\boldsymbol{\beta}_1$ can be obtained by MLE.

The likelihood function is given by:

$$L(\beta_0, \boldsymbol{\beta}_1) = \prod_{i=1}^{n} p(y_i|\mathbf{x}_i)$$

The maximization has no analytical formula and a gradient descent is typically applied to numerical approximation of the parameters.

Recap

Summary of the Lecture:

Recap

Summary of the Lecture:

- K-means is a particular case of Bayesian clustering (mixture resolving)
  - spherically shaped clusters
  - computationally efficient

Recap

Summary of the Lecture:

- K-means is a particular case of Bayesian clustering (mixture resolving)
  - spherically shaped clusters
  - computationally efficient
- Bayesian classifiers are generative models (distribution to $p(\mathbf{x}, y)$)
  - a single covariance matrix for all classes result in linear decision boundaries

Recap

Summary of the Lecture:

- K-means is a particular case of Bayesian clustering (mixture resolving)
  - spherically shaped clusters
  - computationally efficient
- Bayesian classifiers are generative models (distribution to $p(\mathbf{x}, y)$)
  - a single covariance matrix for all classes result in linear decision boundaries
- Naive Bayes assume an independent Gaussian distribution to each attribute and each class.
  - A single variance results in linear decision boundaries

Recap

Summary of the Lecture:

- K-means is a particular case of Bayesian clustering (mixture resolving)
  - spherically shaped clusters
  - computationally efficient
- Bayesian classifiers are generative models (distribution to $p(\mathbf{x}, y)$)
  - a single covariance matrix for all classes result in linear decision boundaries
- Naive Bayes assume an independent Gaussian distribution to each attribute and each class.
  - A single variance results in linear decision boundaries
- Logistic regression is a discriminative model (distribution to $p(y|\mathbf{x})$
  - No analytical solution, paramters are aproximated numerically.
  - Linear decision boundary