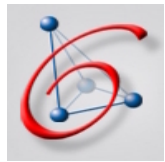
Master 2 Informatique

Probabilistic Learning and Data Analysis

Faicel Chamroukhi
Maître de Conférences
UTLN, LSIS UMR CNRS 7296

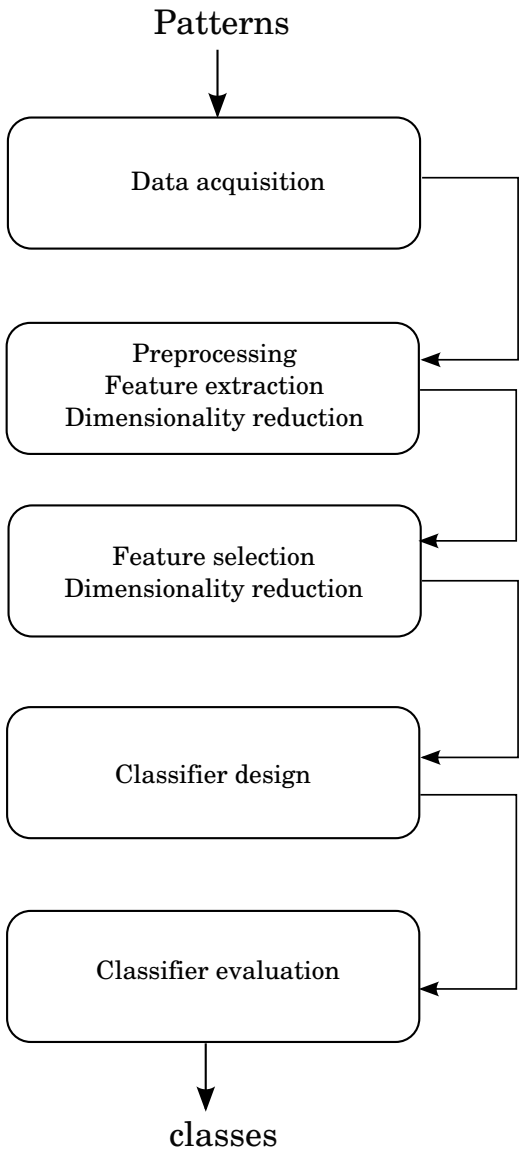


email: chamroukhi@univ-tln.fr
web: chamroukhi.univ-tln.fr

Overview

Introduction on pattern recognition

Pattern recognition system



- Sensors
-> *Rough data*

- Denoising
- Data standardization
- Feature extraction
-> *Features*

- Feature selection
- Optimize the reprezentation space:
linear/nonlinear projection
-> *"Optimal features"*

- Chose the learning approach
- Learn the classes' parameters (if generative)
learn the decision boundaries (eg, discriminative)
- Model selection
-> *Decision rule*

- *"Score (e.g, classification error rate)"*

Pattern recognition system

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.

Pattern recognition system

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.
- **Data Preprocessing :** *denoising*, *standardization*,... **feature extraction** for data *representation*.

Pattern recognition system

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.
- **Data Preprocessing :** *denoising*, *standardization*,... **feature extraction** for data *representation*.
- **Feature selection :** optimization of the representation space by applying for example, *linear* or *nonlinear dimensionality reduction* techniques, in a *supervised* or *unsupervised* context.

Pattern recognition system

- **Data acquisition :** the sensors part in which the rough data (e.g, speech signals, images ...) are acquired (e.g., measured) through dedicated sensors.
- **Data Preprocessing :** *denoising*, *standardization*,... **feature extraction** for data *representation*.
- **Feature selection :** optimization of the representation space by applying for example, *linear* or *nonlinear dimensionality reduction* techniques, in a *supervised* or *unsupervised* context.
- **Classifier design :** Given a training set of (selected) features (observations) $\rightarrow$ design of a *decision rule* with respect to a chosen *optimality criterion*

Pattern recognition system

- **Classifier evaluation** : Once the classifier is designed, its performance has to be assesses for example by computing the *classification error rate* on new data examples, in order to evaluate its generalization capabilities.

Pattern recognition system

- **Classifier evaluation** : Once the classifier is designed, its performance has to be assesses for example by computing the *classification error rate* on new data examples, in order to evaluate its generalization capabilities.

- these stages are not independent. They are highly interrelated and, depending on the results, one may go back to redesign earlier stages in order to improve the overall performance.

- There are some methods that combine stages, for example, a statistical learning at two stages (learning for feature extraction and learning for classification).

- Some stages can also be removed, for example when the classifier is directly built without feature extraction nor feature selection.

Data representation

Speech signal representation :

- Linear Predictive Coding (LPC)
- Mel Frequency Cepstrum Coding (MFCC)
- ...

Image representation :

- Histogram
- Local Binary Patterns (LBP)
- ...

Data Classification

- We talk about "direct" classification, e.g., K-NN

Data Classification

- We talk about "direct" classification, e.g., K-NN
- Very often, the classification task involves a learning problem (e.g., Neural Networks, Support Vector Machines, Gaussian Discriminant Analysis, Gaussian Mixtures, Hidden Markov Models,...)

  $\Rightarrow$ we learn a decision rule, or a functional mapping between possibly labeled features and the considered classes.

Data Classification

- We talk about "direct" classification, e.g., K-NN
- Very often, the classification task involves a learning problem (e.g., Neural Networks, Support Vector Machines, Gaussian Discriminant Analysis, Gaussian Mixtures, Hidden Markov Models,...)

  $\Rightarrow$ we learn a decision rule, or a functional mapping between possibly labeled features and the considered classes.
- The main questions concerning the classifier design : the type of the approach (generative, discriminative,..), linear/non-linear separation, and the type of the criterion.

Data Classification

- We talk about "direct" classification, e.g., K-NN
- Very often, the classification task involves a learning problem (e.g., Neural Networks, Support Vector Machines, Gaussian Discriminant Analysis, Gaussian Mixtures, Hidden Markov Models,...)

  $\Rightarrow$ we learn a decision rule, or a functional mapping between possibly labeled features and the considered classes.
- The main questions concerning the classifier design : the type of the approach (generative, discriminative,..), linear/non-linear separation, and the type of the criterion.

  $\Rightarrow$ In this course, we focus on probabilistic classifiers in a *maximum likelihood estimation (MLE)* framework.
- Probabilistic approaches can easily address problems related to missing information and allows for the integration of prior knowledge (Bayesian approaches), such as experts information.

Machine learning context

- The paradigm for automatically (without human intervention) learning from raw data is known as *machine learning*
- acquisition of knowledge from rough data for analysis, interpretation, prediction.
- *automatically* extracting useful information, possibly unknown, from rough data :
  - features
  - simplified models
  - classes,...

Machine learning context

- Statistical learning = machine learning + Statistics
- distinguished by the fact that the data are assumed to be realizations of random variables $\Rightarrow$ define probability densities over the data $\Rightarrow$ statistical (probabilistic) models.
- take benefit from the asymptotic properties of the estimators, e.g., consistency (e.g., Maximum likelihood)
- To make accurate decisions and predictions for future data, there is an important need to understand the *processes generating the data*.

Machine learning context

- Statistical learning = machine learning + Statistics
- distinguished by the fact that the data are assumed to be realizations of random variables $\Rightarrow$ define probability densities over the data $\Rightarrow$ statistical (probabilistic) models.
- take benefit from the asymptotic properties of the estimators, e.g., consistency (e.g., Maximum likelihood)
- To make accurate decisions and predictions for future data, there is an important need to understand the *processes generating the data*.
- $\Rightarrow$ This therefore leads us to *generative* learning

Machine learning context

Supervised/Unsupervised :

- *Supervised learning* : both the input (observation) and the output (target : class in classification) are available

- The objective is to predict the class of new data given predefined learned classes : *classification (discrimination)* problem

Machine learning context

Supervised/Unsupervised :

- *Supervised learning* : both the input (observation) and the output (target : class in classification) are available

- The objective is to predict the class of new data given predefined learned classes : *classification (discrimination)* problem

- In several application domains, we are confronted with the problem of missing information (class label missing, unknown, hidden).
- $\Rightarrow$ an *unsupervised* learning problem
- The objective is to discover possible classes (exploratory analysis)
- main models : latent data models : e.g., mixture models, HMMs
  $\Rightarrow$ a *clustering/segmentation* problem.

Machine learning context

Static/Dynamic :
Two contexts for the classification problem :

- Static context : The classification rules are taken from *static* modeling techniques because the data are assumed to be independent
- this hypothesis may be restrictive regarding some real phenomena
- *dynamical* framework : building decision rules from sequential data (or time series).

Objectives

- machine learning concepts for data analysis
- overview of some statistical learning approaches from the literature, with a particular focus on generative learning (see how they work).
- How can we define an accurate discrimination rule by considering both homogeneous and dispersed data ? (classification (discrimination)
- When expert information is missing, how can we automatically search for possible classes ? unsupervised learning for segmentation, clustering..
- How can we model the underlying (dynamical) behavior from sequential data ? (Sequential modeling)

Discriminative learning

- Two main approaches are generally used in the statistical learning literature : the *discriminative* approach and the *generative* approach
- Discriminative approaches (especially used in supervised learning (classification, regression)) learn a direct map from the inputs $\mathbf{x}$ to the output y, or they directly learn a model of the conditional distribution $p(y|\mathbf{x})$.
- From the conditional distribution $p(y|\mathbf{x})$, we can make predictions of y for any new value of $\mathbf{x}$ by using the Maximum A Posteriori (MAP) classification rule :

$$\hat{y} = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}).$$

Generative learning

- Generative classifiers learn a model of the joint distribution $p(\mathbf{x}, y)$

  $\Rightarrow$ model the class conditional density $p(\mathbf{x}|y)$ together with the prior probability $p(y)$.
- The required posterior class probability is then computed using Bayes' theorem

$$p(y|\mathbf{x}) = \frac{p(y)p(\mathbf{x}|y)}{\sum_{y'} p(y')p(\mathbf{x}|y')}.$$

- the outputs y are not always available (i.e., they may be missing or hidden)

  $\Rightarrow$ generative approaches are more suitable for unsupervised learning.

$\Rightarrow$ In this course we focus on probabilistic models for data modeling, discrimination and clustering.

Classification (discrimination)

1. Introduction on pattern recognition

2. Classification
   - K-nearest neighbors (KNN)
   - Multi-class logistic regression
   - Neural Network
   - Support Vector Machines (SVMs)
   - Gaussian Discriminant Analysis
   - Mixture Discriminant Analysis

Data Classification

Notations :

- Given a training data set comprising n labeled observations $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$ where $\mathbf{x}$ denotes the observation (or the input) which is assumed to be continuous-valued in $\mathcal{X} = \mathbb{R}^d$

- y denotes the target variable (or the output) representing the class label which is a discrete-valued variable in $\mathcal{Y} = \{1, \ldots, K\}$

- K being the number of classes.

- In classification, the aim is to predict the value of the class label y for a new observation $\mathbf{x}$.

K-NN

- a direct supervised classification approach

K-NN

- a direct supervised classification approach
- Does not need "learning" but only storing the data

K-NN

- a direct supervised classification approach
- Does not need "learning" but only storing the data
- It's Son very simple : its principle is as follows : the class of a new data point is the one of its nearest neighbors (the majority among the K nearest neighbors) in the sense of a chosen distance (e.g, Euclidean distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \tag{1}$$

K-NN

- a direct supervised classification approach
- Does not need "learning" but only storing the data
- It's Son very simple : its principle is as follows : the class of a new data point is the one of its nearest neighbors (the majority among the K nearest neighbors) in the sense of a chosen distance (e.g, Euclidean distance)

$$d(\mathbf{x}_i, \mathbf{x}_j) = \sqrt{\sum_{k=1}^{d} (x_{ik} - x_{jk})^2} \tag{1}$$

- $\Rightarrow$ As it needs computing, for each test data point, the distances with all the data points from the labeled training set, it may be computationally expensive for large data sets.

K-NN

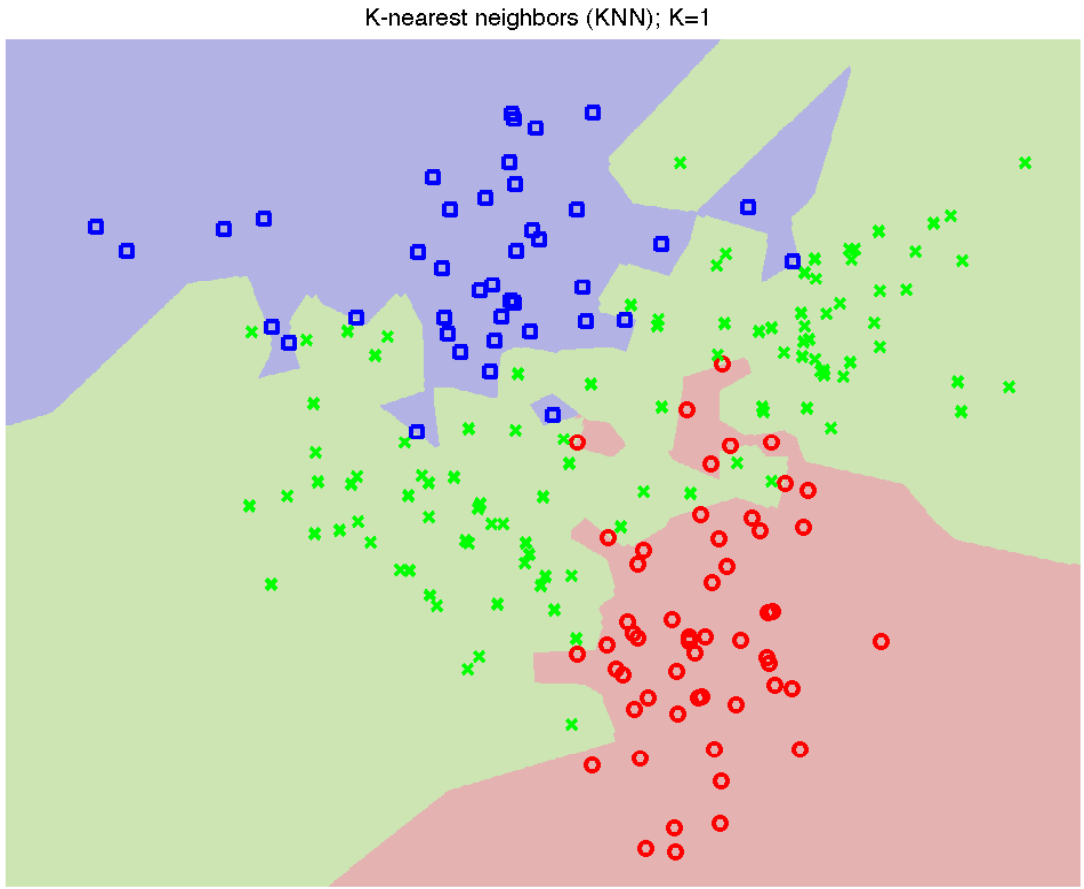
Algorithm 1 K-NN algorithm.

Inputs : Labeled data set : $\mathbf{X}^{\text{train}} = (\mathbf{x}_1^{\text{train}}, ..., \mathbf{x}_n^{\text{train}})$ and $\mathbf{y}^{\text{train}} = (y_1^{\text{train}}, ..., y_n^{\text{train}})$; Test data set $\mathbf{X}^{\text{test}} = (\mathbf{x}_1^{\text{test}}, ..., \mathbf{x}_m^{\text{test}})$; number of NN : K

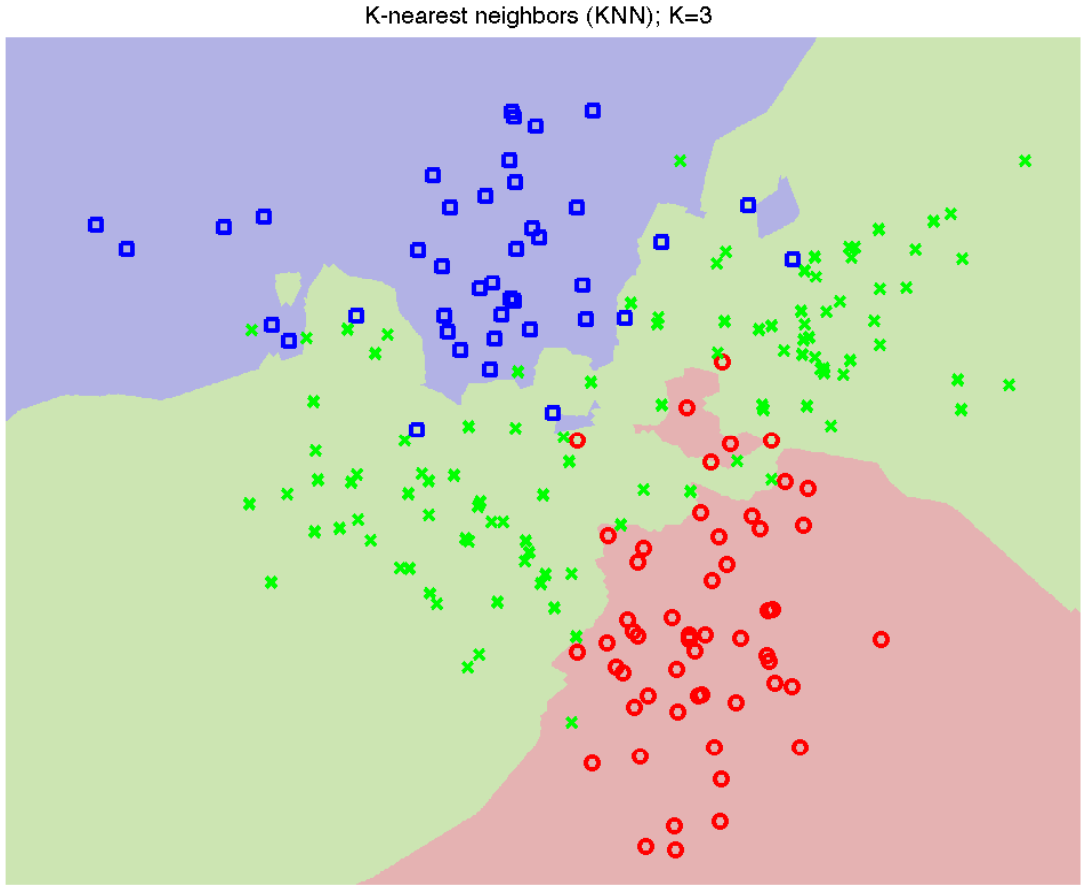
for $i = 1, \ldots, m$ **do**
  for $j = 1, \ldots, n$ **do**
    compute the Euclidean distances d_{ij} between $\mathbf{x}_i^{\text{test}}$and $\mathbf{x}_j^{\text{train}}$
    $\mathbf{d}_j \leftarrow \|\mathbf{x}_i - \mathbf{x}_j\|^2$
  end for
  The class y_i^{test} for the ith example is the one of its nearest neighbors :
  Sort the distance vector $\mathbf{d}_j$ in an increasing order for $j = 1, \ldots, n$
  Get at the same time the indexes of the elements in the new order
  Get the classes of the first K elements
  $\Rightarrow$ the class y_i^{test} is the majority class
end for

Output : Classes of the test data $\mathbf{y}^{\text{test}} = (y_1^{\text{test}}, ..., y_m^{\text{test}})$

K-NN



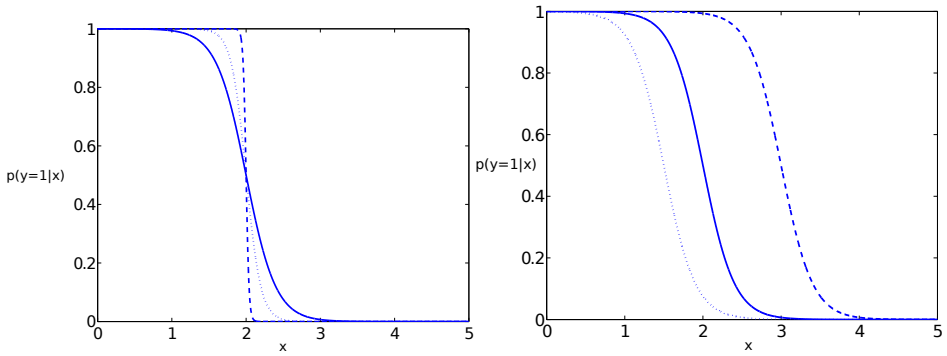
K-NN



Multi-class logistic regression

- a probabilistic supervised discriminative approach
- directly models the classes' posterior probabilities via :

$$p(y = k|\mathbf{x}) = \pi_k(\mathbf{x}; \mathbf{w}) = \frac{\exp(w_k^T \mathbf{x})}{\sum_{h=1}^{K} \exp(w_h^T \mathbf{x})}$$



- a logistic transformation of a linear function in $\mathbf{x}$
- ensures that the posterior probabilities are constrained to sum to one and remain in $[0, 1]$.
- The model parameter : $\mathbf{w} = (w_1, \ldots, w_K)^T$

Parameter estimation for Multi-class logistic regression

- The maximum likelihood is used to fit the model.
- The conditional log-likelihood of $\mathbf{w}$ for the given class labels $\mathbf{y} = (y_1, \ldots, y_n)$ conditionally on the inputs $\mathbf{X} = (\mathbf{x}_1, \ldots, \mathbf{x}_n)$:

$$\begin{aligned}
\mathcal{L}(\mathbf{w}) = \mathcal{L}(\mathbf{w}; \mathbf{X}, \mathbf{y}) &= \log \prod_{i=1}^{n} p(y_i|\mathbf{x}_i; \mathbf{w}) \\
&= \log \prod_{i=1}^{n} \prod_{k=1}^{K} p(y_i = k|\mathbf{x}_i; \mathbf{w})^{y_{ik}} \\
&= \sum_{i=1}^{n} \sum_{k=1}^{K} y_{ik} \log \pi_k(\mathbf{x}_i; \mathbf{w})
\end{aligned}$$

  where y_{ik} is an indicator binary variable such that $y_{ik} = 1$ if and only $y_i = k$ (i.e, $\mathbf{x}_i$ belongs to the class k).
- This log-likelihood is convex but can not be maximized in a closed form.
- The Newton-Raphson (NR) algorithm is generally used

Newton-Raphson for Multi-class logistic regression

- The Newton-Raphson algorithm is an iterative numerical optimization algorithm
- starts from an initial arbitrary solution $\mathbf{w}^{(0)}$, and updates the estimation of $\mathbf{w}$
- A single NR update is given by :

$$\mathbf{w}^{(l+1)} = \mathbf{w}^{(l)} - \left[\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T}\right]^{-1} \frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} \tag{2}$$

  where the Hessian and the gradient of $\mathcal{L}(\mathbf{w})$ (which are respectively the second and first derivative of $\mathcal{L}(\mathbf{w})$) are evaluated at $\mathbf{w} = \mathbf{w}^{(l)}$.
- NR can be stopped when the relative variation of $\mathcal{L}(\mathbf{w})$ is below a prefixed threshold.

IRLS I

The gradient component $\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_h}$ $(h = 1, \ldots, K-1)$ is given by

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_h} = \sum_{i=1}^{n} \big(y_{ih} - \pi_h(\mathbf{x}_i; \mathbf{w})\big)\mathbf{x}_i$$

which can be formulated in a matrix form as

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial w_h} = \mathbf{X}^T(\mathbf{y}_h - \mathbf{p}_h)$$

where $\mathbf{X}$ is the $n \times (d+1)$ matrix whose rows are the input vectors $\mathbf{x}_i$, $\mathbf{y}_h$ is the $n \times 1$ column vector whose elements are the indicator variables y_{ih} for the hth logistic component :

$$\mathbf{y}_h = (y_{1h}, \ldots, y_{nh})^T$$

IRLS II

and $\mathbf{p}_h$ is the $n \times 1$ column vector of logistic probabilities corresponding to the ith input

$$\mathbf{p}_h = (\pi_h(\mathbf{x}_1; \mathbf{w}), \ldots, \pi_h(\mathbf{x}_n; \mathbf{w}))^T.$$

Thus, the matrix formulation of the gradient of $\mathcal{L}(\mathbf{w})$ for all the logistic components is

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \mathbf{X}^{*T}(\mathbf{Y} - \mathbf{P}) \quad (3)$$

where $\mathbf{Y} = (\mathbf{y}_1^T, \ldots, \mathbf{y}_{K-1}^T)^T$ and $\mathbf{P} = (\mathbf{p}_1^T, \ldots, \mathbf{p}_{K-1}^T)^T$ are $n \times (K-1)$ column vectors and $\mathbf{X}^*$ is the $(n \times (K-1))$ by $(d+1)$ matrix of $K-1$ copies of $\mathbf{X}$ such that $\mathbf{X}^* = (\mathbf{X}^T, \ldots, \mathbf{X}^T)^T$.
The Hessian matrix is composed of $(K-1) \times (K-1)$ block matrices where each block matrix is of dimension $(d+1) \times (d+1)$ and is given by :

$$\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial w_h \partial w_k^T} = -\sum_{i=1}^{n} \pi_h(\mathbf{x}_i; \mathbf{w}) \left(\delta_{hk} - \pi_k(\mathbf{x}_i; \mathbf{w})\right) \mathbf{x}_i \mathbf{x}_i^T$$

which can be formulated in a matrix form as

$$\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial w_h \partial w_k^T} = -\mathbf{X}^T \mathbf{W}_{hk} \mathbf{X}$$

where $\mathbf{W}_{hk}$ is the $n \times n$ diagonal matrix whose diagonal elements are $\pi_h(\mathbf{x}_i; \mathbf{w}) \left(\delta_{hk} - \pi_k(\mathbf{x}_i; \mathbf{w})\right)$ for $i = 1, \ldots, n$. For all the logistic components $(h, k = 1, \ldots, K-1)$, the Hessian takes the following form :

$$\frac{\partial^2 \mathcal{L}(\mathbf{w})}{\partial \mathbf{w} \partial \mathbf{w}^T} = -\mathbf{X}^{*T} \mathbf{W} \mathbf{X}^* \quad (4)$$

where $\mathbf{W}$ is the $(n \times (K-1))$ by $(n \times (K-1))$ matrix composed of $(K-1)) \times (K-1))$ block matrices, each block is $\mathbf{W}_{hk}$ $(h, k = 1, \ldots, K-1)$. It can be shown that the Hessian matrix for the multi-class logistic regression model is positive semi definite and therefore the optimized log-likelihood is concave.
The NR algorithm (2) in this case can therefore be reformulated from the Equations (3) and (4) as

$$\begin{aligned}
\mathbf{w}^{(l+1)} &= \mathbf{w}^{(l)} + (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*})^{-1}\mathbf{X}^{*T}(\mathbf{Y} - \mathbf{P}^{(l)}) \\
&= (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*})^{-1}\left[\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*}\mathbf{w}^{(l)} + \mathbf{X}^{*T}(\mathbf{Y} - \mathbf{P}^{(l)})\right] \\
&= (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X})^{-1}\mathbf{X}^{*T}\left[\mathbf{W}^{(l)}\mathbf{X}^{*}\mathbf{w}^{(l)} + (\mathbf{Y} - \mathbf{P}^{(l)})\right] \\
&= (\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{X}^{*})^{-1}\mathbf{X}^{*T}\mathbf{W}^{(l)}\mathbf{Y}^{*}
\end{aligned}$$

where $\mathbf{Y}^{*} = \mathbf{X}^{*}\mathbf{w}^{(l)} + (\mathbf{W}^{(l)})^{-1}(\mathbf{Y} - \mathbf{P}^{(l)})$ which yields in the Iteratively Reweighted Least Squares (IRLS) algorithm.

Neural Network

notes vues en cours

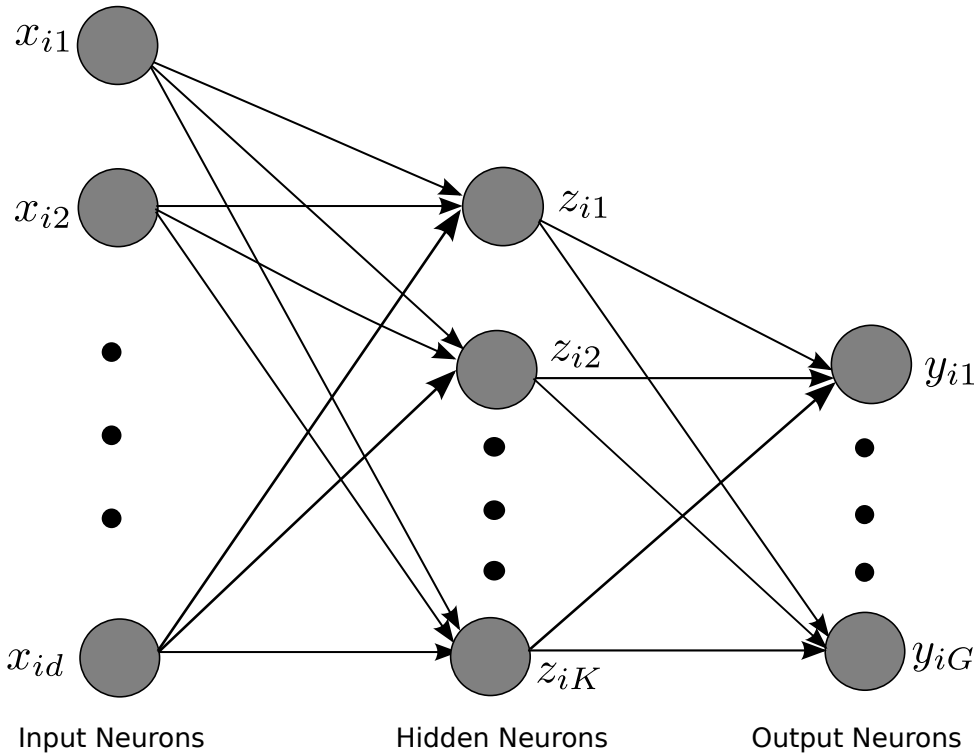


Figure : Graphical representation of Multi-Layer Perceptron (MLP).

Support Vector Machines (SVMs)

1. Introduction on pattern recognition

2. Classification
   - K-nearest neighbors (KNN)
   - Multi-class logistic regression
   - Neural Network
   - Support Vector Machines (SVMs)
   - Gaussian Discriminant Analysis
   - Mixture Discriminant Analysis

Support Vector Machines (SVMs) I

Support vector machines (SVMs) (Guyon and Vapnik, 1992; Vapnik, 1999; Schölkopf, 1997; Schölkopf et al., 1998; Cristianini and Shawe-Taylor, 2000), including support vector classifier (SVC) and support vector regressor (SVR), originally developed by Vapnik, are among the most robust and accurate methods in statistical learning. They are by definition supervised.

Support vector Classifiers (SVM) (Vapnik, 1999; Schölkopf et al., 1998; Schölkopf, 1997) is a supervised learning method that was originally designed as a binary classification algorithm. Consider a two-class linearly separable learning task, the aim of SVC is to find a discriminant function as defined by the hyperplane that separates the positive data points and the negative data points with maximum margin (Cristianini and Shawe-Taylor, 2000; Smola et al., 1999) which has been proved able to offer the best generalization ability (Guyon and Vapnik, 1992; Vapnik, 1998, 1999).

Support Vector Machines (SVMs) II

Suppose we are given a set of inputs $(\mathbf{x}_1, \ldots, \mathbf{x}_n)$ where each $\mathbf{x}_i$ is in $\mathcal{X}$ (we assume that $\mathcal{X} = \mathbb{R}^d$). Any hyperplane in the input space can be written as :

$$\begin{aligned} f_{\mathbf{w}\in\mathbb{R}^d, b\in\mathbb{R}} : \mathcal{X} &\rightarrow \{+1, -1\} \\ \mathbf{x} &\rightarrow \text{sign}(\mathbf{w}^T\mathbf{x} + b) \end{aligned} \tag{5}$$

The margin can be defined as the amount of space, or separation, between the two classes as defined by a hyperplane. Geometrically, the margin corresponds to the shortest distance between the closest data points to any point on the hyperplane. The corresponding optimal hyperplane can be defined as

$$\mathbf{w}^T\mathbf{x} + b = 0 \tag{6}$$

where $\mathbf{w}$ is a d dimensional weight vector and b is a scalar called the bias (also called the intercept), in the optimal hyperplane. The desired

Support Vector Machines (SVMs) III

directionally geometrical distance from the data point $\mathbf{x}$ to the optimal hyperplane is :

$$\frac{\mathbf{w}^T\mathbf{x} + b}{\|\mathbf{w}\|} \tag{7}$$

Consequently, SVC aims to find the parameters $\mathbf{w}$ and b for an optimal hyperplane in order to maximize the margin of separation that is determined by the shortest geometrical distances r^* from the two classes, respectively, thus SVC is also called *maximal margin classifier*. Now if we scale the margin to be equal to 1 (Cristianini and Shawe-Taylor, 2000), that is, given a training set $((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n))$ with $(\mathbf{x}_i, y_i) \in \mathcal{X} \times \{+1, -1\}$ we have

$$\begin{cases} \mathbf{w}^T\mathbf{x}_i + b \geq 1 \text{ for } y_i = +1 \\ \mathbf{w}^T\mathbf{x}_i + b \leq -1 \text{ for } y_i = -1 \end{cases} \tag{8}$$

Support Vector Machines (SVMs) IV

The particular data points (x_i, y_i) for which the equalities of the first or second parts in Equation (8) are satisfied, that is

$$\mathbf{w}^T\mathbf{x}^* + b = \pm 1 \tag{9}$$

are called *support vectors*, which are the data points that lie on the optimal hyperplane and which we denote by $\mathbf{x}^*$. Then, from (7) and (9), the corresponding geometrical distance from the support vector $\mathbf{x}^*$ to the optimal hyperplane is :

$$r^* = \frac{\mathbf{w}^T\mathbf{x}^* + b}{\|\mathbf{w}\|} = \begin{cases} \frac{1}{\|\mathbf{w}\|} \text{ if } y^* = +1 \\ \frac{-1}{\|\mathbf{w}\|} \text{ if } y^* = -1 \end{cases} \tag{10}$$

where y^* is the label of the support vector $\mathbf{x}^*$. Figure 3 illustrates a geometric construction of the corresponding optimal hyperplane for a two-dimensional input space.

Support Vector Machines (SVMs) V

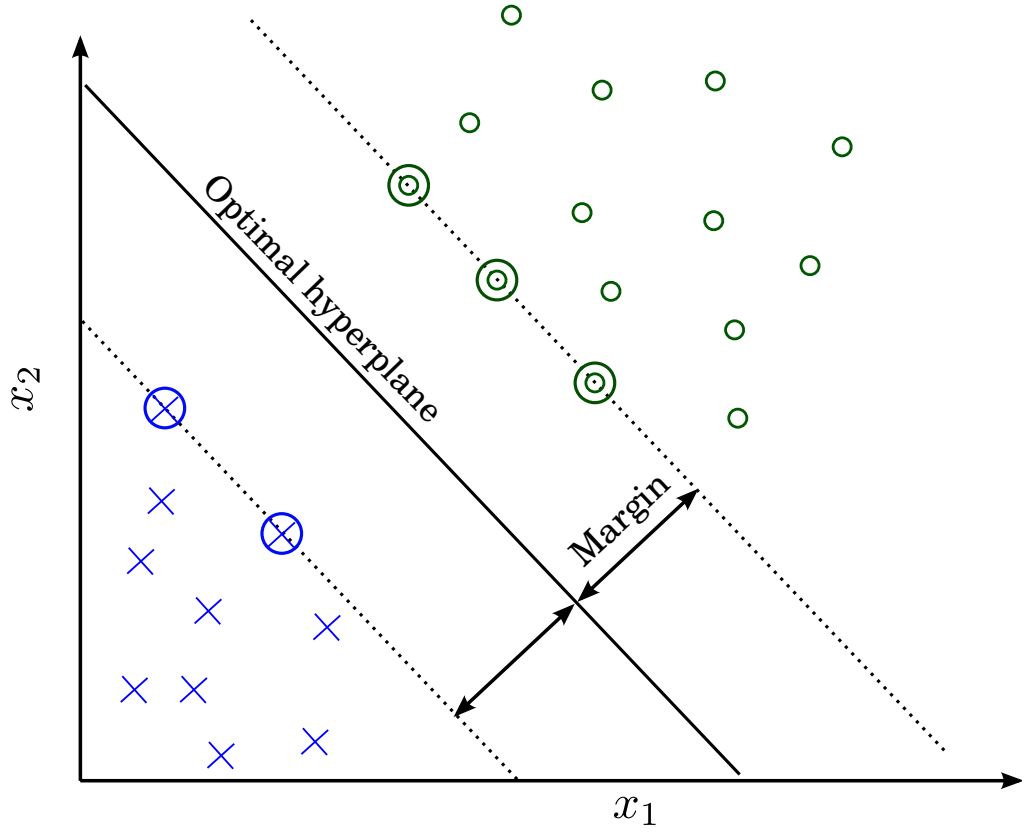


Figure : Illustration of the optimal hyperplane in SVC for a linearly separable case.

Support Vector Machines (SVMs) VI

As it can be seen on Figure 3, the total margin of separation is $\rho = 2r^* = \frac{2}{\|\mathbf{w}\|}$. To ensure that the maximum margin hyperplane can be found, SVC attempts to maximize ρ with respect to $\mathbf{w}$ and b subject to a linear constraints, that is :

$$\begin{aligned} \max_{\mathbf{w}\in\mathbb{R}^d} \quad & \frac{1}{\|\mathbf{w}\|} \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}^* + b) \geq 1 \ \forall i = 1, \ldots, n \end{aligned} \tag{11}$$

or equivalently,

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^d} \quad & \frac{1}{2}\|\mathbf{w}\|^2 \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 \ \forall i = 1, \ldots, n. \end{aligned} \tag{12}$$

Support Vector Machines (SVMs) VII

The constrained optimization problem given in Equation (12) is known as the *primal problem* and it is generally solved by using the Lagrange multipliers. The constructed Lagrange function is given by :

$$L_{primal}(\mathbf{w}, b, \alpha) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{i=1}^{n} \alpha_i [y_i(\mathbf{w}^T \mathbf{x} + b) - 1] \tag{13}$$

where α_i is the Lagrange multiplier with respect to the ith inequality. Differentiating $L(\mathbf{w}, b, \alpha)$ with respect to $\mathbf{w}$ and b, and setting the results equal to zero, we get the following two conditions of optimality :

$$\begin{cases} \mathbf{w} = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \\ \sum_{i=1}^{n} \alpha_i y_i = 0 \end{cases} \tag{14}$$

Support Vector Machines (SVMs) VIII

Substituting Equation (14) into the Lagrange function Equation (13), we can get the corresponding *dual problem* :

$$\begin{aligned} \max_{(\alpha_1,\ldots,\alpha_n)\in\mathbb{R}^n} \quad & L_{dual}(\alpha_1,\ldots,\alpha_n) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \ \forall i = 1,\ldots,n \\ \text{and} \quad & \alpha_i \geq 0 \ \forall i = 1,\ldots,n. \end{aligned} \tag{15}$$

The dual problem in Equation (15) is a typical convex quadratic programming optimization problem. In many cases, it can efficiently converge to the global optimum by adopting some appropriate optimization techniques, such as the sequential minimal optimization (SMO) algorithm (Cristianini and Shawe-Taylor, 2000; Schölkopf and Smola, 2001). Once we have obtained the α_i values for $i = 1,\ldots,n$, we can solve for the weight

Support Vector Machines (SVMs) IX

vector $\mathbf{w}$ and the bias b. According to the KKT complementarity conditions (see Chapter 6 in (Schölkopf and Smola, 2001)) only the Lagrange multipliers α_i's that are non-zero correspond to constraints in (12) which are precisely met. Formally, for all $i = 1, \ldots, n$, we have :

$$\alpha_i[y_i(\mathbf{w}^T\mathbf{x} + b) - 1] = 0 \ \forall i = 1, \ldots, n. \tag{16}$$

which gives rise to two cases :

i. $\alpha_i = 0$, or

ii. $y_i(\mathbf{w}^T\mathbf{x} + b) - 1 = 0$, which gives $y_i(\mathbf{w}^T\mathbf{x}_i + b) = 1$

Support Vector Machines (SVMs) X

The data points $\mathbf{x}_i$ for which $\alpha_i > 0$ are called *Support Vectors*. According to (16), they lie exactly on the margin. The other points, which represent the vast majority of the data points, will have $\alpha_i = 0$
After determining the optimal Lagrange multipliers α_i^* , we can compute the optimal weight vector $\mathbf{w}^*$ by Equation (14) :

$$\mathbf{w}^* = \sum_{i=1}^{n} \alpha_i^* y_i \mathbf{x}_i \tag{17}$$

Then, if we take a positive support vector $\mathbf{x}^*$, the corresponding optimal bias b^* can be written as

$$b^* = 1 - \mathbf{w}^{*T} \mathbf{x}^* \text{ for } y^* = +1. \tag{18}$$

Maximal margin SVC, represents the original starting point of the SVM algorithms and assumes that the dataset is perfectly linearly separable. However, in many real-world problems, especially in many complex

Support Vector Machines (SVMs) XI

nonlinear classification cases the classes cannot be completely linearly separated, and therefore the hard margin SVC will not perform well the classification task. To solve these inseparable problems, two approaches are generally adopted . The first one is to relax the rigid inequalities in Equation (12) and thus lead to so-called soft margin optimization. Another method is to apply the kernel trick to linearize those nonlinear problems from the input space to the future space . In this section, we first give an overview of soft margin SVC.

Here we consider the case where the classes overlap so that a perfect separation is not possible, as shown in Figure **??**. The points of the opposite classes that are mixed together in the data represent the training error that exists even for the maximum margin hyperplane. The soft margin idea aims to extend the SVC to handle such a set of points with overlapping classes so that the hyperplane allows a few of such noisy data

Support Vector Machines (SVMs) XII

to exist. In particular, a slack variable $\xi_i \geq 0$ is introduced in (8) to account for the amount of a violation of classification by the classifier as follows :

$$y_i(\mathbf{w}^T\mathbf{x} + b) \geq 1 - \xi_i \tag{19}$$

which correspond to a relaxed separation constraints. Here $\xi_i \geq 0$ is the new slack variable for point $\mathbf{x}_i$. First note that if $\xi_i = 0$, then the point is treated the same way as before for the maximum margin SVM in the linear separable case. Now if $0 < \xi_i < 1$, then the point is still correctly classified, since it will remains on the right side of the hyperplane. Finally, if $\xi_i \geq 1$ then the point is misclassified, since in this case it appears on the wrong side of the hyperplane.

Support Vector Machines (SVMs) XIII

The goal of classification now is to find the hyperplane (given by $\mathbf{w}$ and b) with the maximum margin, that also minimizes the sum of the slack variables. The new objective function becomes :

$$\begin{aligned} \min_{\mathbf{w}\in\mathbb{R}^d,(\xi_1,\ldots\xi_n)\in\mathbb{R}^n} \quad & \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}+b) \geq 1-\xi_i \ \forall i = 1,\ldots,n \\ \text{and} \quad & \xi_i \geq 0 \ \forall i = 1,\ldots,n \end{aligned} \tag{20}$$

where C is a fixed regularization parameter that controls the trade-off between complexity of the model and the number of inseparable points.

Support Vector Machines (SVMs) XIV

This new primal problem is solved using Lagrange multipliers and the new dual problem of the soft margin is formulated as (Schölkopf and Smola, 2001) :

$$\begin{aligned} \max_{(\alpha_1,\ldots,\alpha_n)\in\mathbb{R}^n} \quad & L_{dual}(\alpha_1,\ldots,\alpha_n) = \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j \\ \text{subject to} \quad & \sum_{i=1}^{n}\alpha_i y_i = 0 \ \forall i = 1,\ldots,n \\ \text{and} \quad & 0 \leq \alpha_i \leq C \ \forall i = 1,\ldots,n. \end{aligned} \tag{21}$$

It can be seen that the slack variables ξ_i's do not appear in the dual problem for this linearly inseparable case (21) which is exactly the same as the dual problem in the linearly separable case (15). The only difference is the constraint on the α_i that is $\alpha_i \geq 0$ is replaced with$0 \leq \alpha_i \leq C$. Otherwise, the two cases are similar, including the computations of the optimal values $\mathbf{w}$ and b and the definition of the support vectors.

Linear Discriminant Analysis

- generative approach that consists in modeling each conditional-class density by a multivariate Gaussian :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

Linear Discriminant Analysis

- generative approach that consists in modeling each conditional-class density by a multivariate Gaussian :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

- $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is the mean vector
- $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$ is the covariance matrix
- $\boldsymbol{\Psi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for $k = 1, \ldots, K$.

Linear Discriminant Analysis

- generative approach that consists in modeling each conditional-class density by a multivariate Gaussian :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \frac{1}{(2\pi)^{\frac{d}{2}}|\boldsymbol{\Sigma}_k|^{\frac{1}{2}}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_k)^T\boldsymbol{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k)\Big)$$

- $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is the mean vector
- $\boldsymbol{\Sigma}_k \in \mathbb{R}^{d\times d}$ is the covariance matrix
- $\boldsymbol{\Psi}_k = (\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ for $k = 1, \ldots, K$.

- Linear Discriminant Analysis (LDA) arises when we assume that all the classes have a common covariance matrix $\boldsymbol{\Sigma}_k = \boldsymbol{\Sigma}\ \forall k = 1, \ldots, K$.

Linear Discriminant Analysis

- The term “linear" in LDA is due to the fact that the decision boundaries between each pair of classes k and h are linear.

Linear Discriminant Analysis

- The term “linear" in LDA is due to the fact that the decision boundaries between each pair of classes k and h are linear.

- The decision boundary between classes k and h, which is the set of inputs $\mathbf{x}$ verifying $p(y = k|\mathbf{x}) = p(y = h|\mathbf{x})$, or by equivalence :

$$\log \frac{p(y = g|\mathbf{x}; \boldsymbol{\Psi}_k)}{p(y = h|\mathbf{x}; \boldsymbol{\Psi}_h)} = 0 \Leftrightarrow \log \frac{\pi_k}{\pi_h} + \log \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_h, \boldsymbol{\Sigma})} =$$

Linear Discriminant Analysis

- The term "linear" in LDA is due to the fact that the decision boundaries between each pair of classes k and h are linear.

- The decision boundary between classes k and h, which is the set of inputs $\mathbf{x}$ verifying $p(y = k|\mathbf{x}) = p(y = h|\mathbf{x})$, or by equivalence :

$$\log \frac{p(y = g|\mathbf{x}; \boldsymbol{\Psi}_k)}{p(y = h|\mathbf{x}; \boldsymbol{\Psi}_h)} = 0 \Leftrightarrow \log \frac{\pi_k}{\pi_h} + \log \frac{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_k, \boldsymbol{\Sigma})}{\mathcal{N}(\mathbf{x}; \boldsymbol{\mu}_h, \boldsymbol{\Sigma})} =$$

$$\Leftrightarrow \log \tfrac{\pi_k}{\pi_h} - \tfrac{1}{2}(\boldsymbol{\mu}_k + \boldsymbol{\mu}_h)^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_h) + \mathbf{x}^T \boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_k - \boldsymbol{\mu}_h) = 0,$$

$\Rightarrow$ a linear function in $\mathbf{x}$ and therefore the classes will be separated by hyperplanes in the input space.

Linear Discriminant Analysis : Parameter Estimation

- Each of the class prior probabilities π_k is calculated with the proportion of the class g in the training data set :

$$\pi_k = \frac{\sum_{i|y_i=k}}{n} = \frac{n_k}{n}.$$

Linear Discriminant Analysis : Parameter Estimation

- Each of the class prior probabilities π_k is calculated with the proportion of the class g in the training data set :

$$\pi_k = \frac{\sum_{i|y_i=k}}{n} = \frac{n_k}{n}.$$

- The parameters $\boldsymbol{\Psi}_k$ are estimated by maximum likelihood

Linear Discriminant Analysis : Parameter Estimation

- Each of the class prior probabilities π_k is calculated with the proportion of the class g in the training data set :

$$\pi_k = \frac{\sum_{i|y_i=k}}{n} = \frac{n_k}{n}.$$

- The parameters $\boldsymbol{\Psi}_k$ are estimated by maximum likelihood
- the log-likelihood of $\boldsymbol{\Psi}_k$ given an i.i.d sample :

$$\mathcal{L}(\boldsymbol{\Psi}_k) = \log \prod_{i|y_i=k} \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}) = \sum_{i|y_i=k} \log \mathcal{N}(\mathbf{x}_i; \boldsymbol{\mu}_k, \boldsymbol{\Sigma}).$$

- $\Rightarrow$ The problem is solved in a closed form

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i|y_i=k} \mathbf{x}_i,$$

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

Illustration

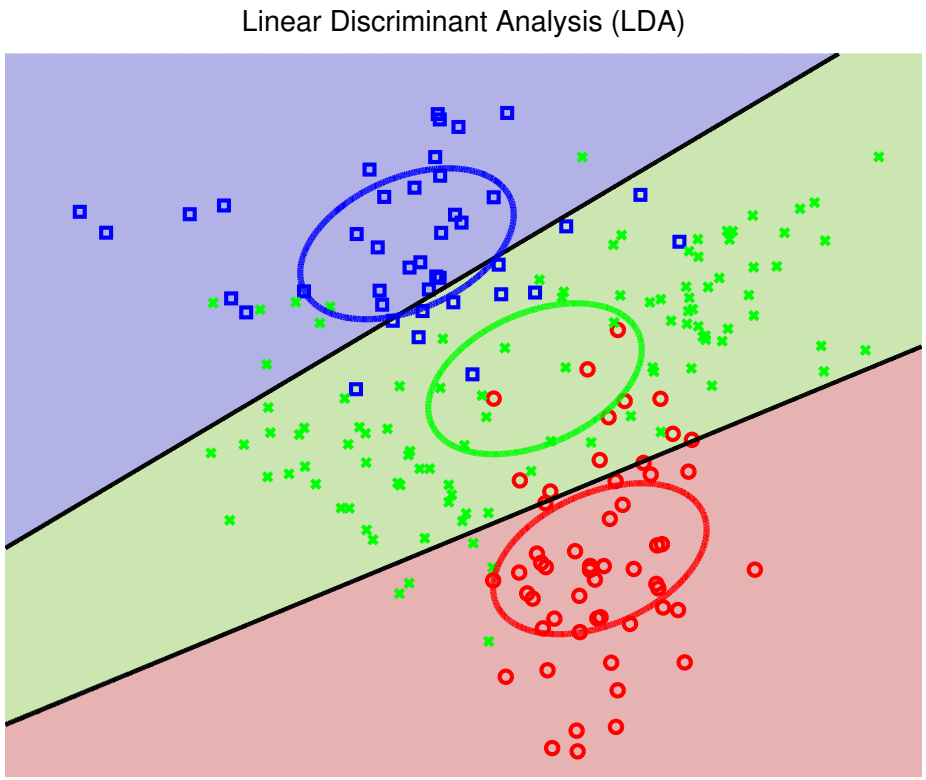


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that LDA provides linear separation.

Quadratic Discriminant Analysis

- Quadratic Discriminant Analysis (QDA) is an extension of LDA that considers a different covariance matrix for each class.

Quadratic Discriminant Analysis

- Quadratic Discriminant Analysis (QDA) is an extension of LDA that considers a different covariance matrix for each class.
- The decision functions are quadratic :

$$\log \frac{p(y=k|\mathbf{x})}{p(y=h|\mathbf{x})} = \log \frac{\pi_k}{\pi_h} - \frac{1}{2} \log \frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_h|} - \frac{1}{2}\{(\mathbf{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_h)^T \mathbf{\Sigma}_h^{-1}(\mathbf{x}-\boldsymbol{\mu}_h)\} = 0.$$

$\Rightarrow$ This function is quadratic in $\mathbf{x}$, we then get quadratic discriminant functions in the input space.

Quadratic Discriminant Analysis

- Quadratic Discriminant Analysis (QDA) is an extension of LDA that considers a different covariance matrix for each class.
- The decision functions are quadratic :

$$\log \frac{p(y=k|\mathbf{x})}{p(y=h|\mathbf{x})} = \log \frac{\pi_k}{\pi_h} - \frac{1}{2} \log \frac{|\mathbf{\Sigma}_k|}{|\mathbf{\Sigma}_h|} - \frac{1}{2}\{(\mathbf{x}-\boldsymbol{\mu}_k)^T \mathbf{\Sigma}_k^{-1}(\mathbf{x}-\boldsymbol{\mu}_k) - (\mathbf{x}-\boldsymbol{\mu}_h)^T \mathbf{\Sigma}_h^{-1}(\mathbf{x}-\boldsymbol{\mu}_h)\} = 0.$$

  $\Rightarrow$ This function is quadratic in $\mathbf{x}$, we then get quadratic discriminant functions in the input space.
- The parameters $\mathbf{\Psi}_k$ for QDA are estimated similarly as for LDA, except that separate covariance matrix must be estimated for each class :

$$\hat{\boldsymbol{\mu}}_k = \frac{1}{n_k} \sum_{i|y_i=k} \mathbf{x}_i$$

$$\hat{\mathbf{\Sigma}}_k = \frac{1}{n_k} \sum_{i|y_i=k} (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T.$$

Illustration

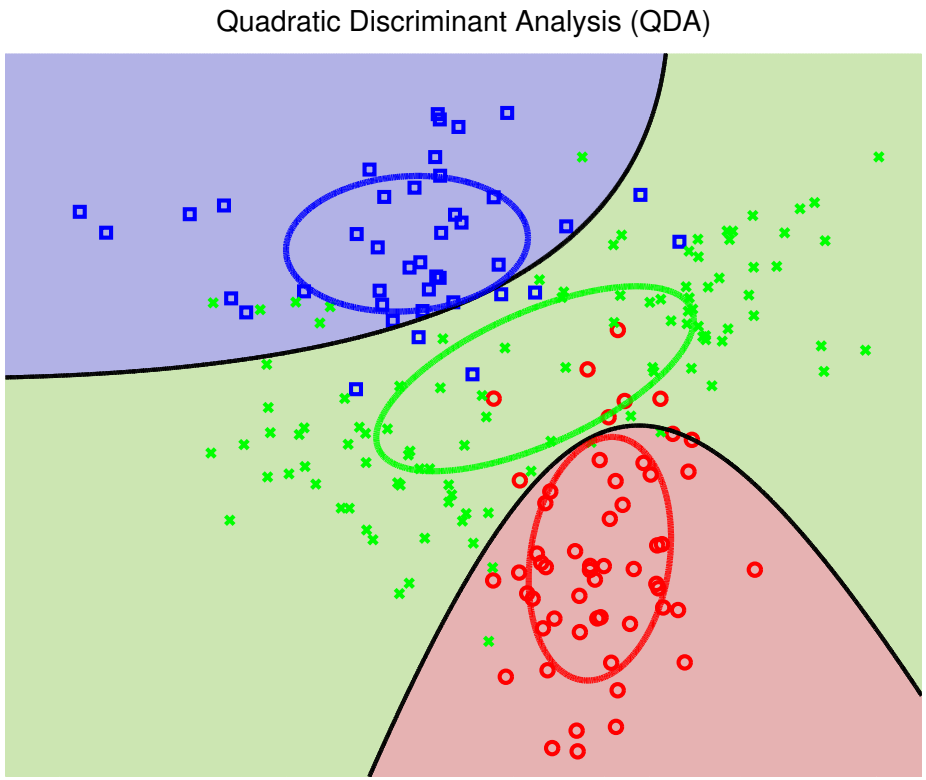


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that QDA provides quadratic boundaries in the plan.

Mixture Discriminant Analysis

- for Gaussian discriminant analysis, in both LDA and QDA, each class density is modeled by a single Gaussian.

- This may be limited for modeling non homogeneous classes where the classes are dispersed.

  $\Rightarrow$ In Mixture Discriminant Analysis (MDA) each class density is modeled by a Gaussian mixture density

- with MDA, we can therefore capture many specific properties of real data such as multimodality, unobserved heterogeneity, heteroskedasticity, etc.

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

where R_k is the number of mixture components for class k

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

  where R_k is the number of mixture components for class k

- $$\boldsymbol{\Psi}_k = (\pi_{k1},\ldots,\pi_{kR_k},\boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kR_k},\ldots,\boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kR_k})$$

  is the parameter vector of the mixture density of class k

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

  where R_k is the number of mixture components for class k

- $$\boldsymbol{\Psi}_k = (\pi_{k1},\ldots,\pi_{kR_k},\boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kR_k},\ldots,\boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kR_k})$$

  is the parameter vector of the mixture density of class k
- the π_{kr}'s $(r = 1,\ldots,R_k)$ are the non-negative mixing proportions satisfying $\sum_{r=1}^{R_k}\pi_{kr} = 1\ \forall k$.

Mixture Discriminant Analysis (MDA)

- Each class g is modeled by a Gaussian mixture density :

$$p(\mathbf{x}|y=k;\boldsymbol{\Psi}_k) = \sum_{r=1}^{R_k} \pi_{kr}\mathcal{N}(\mathbf{x};\boldsymbol{\mu}_{kr},\boldsymbol{\Sigma}_{kr})$$

  where R_k is the number of mixture components for class k

- $$\boldsymbol{\Psi}_k = (\pi_{k1},\ldots,\pi_{kR_k},\boldsymbol{\mu}_{k1},\ldots,\boldsymbol{\mu}_{kR_k},\ldots,\boldsymbol{\Sigma}_{k1},\ldots,\boldsymbol{\Sigma}_{kR_k})$$

  is the parameter vector of the mixture density of class k
- the π_{kr}'s $(r = 1,\ldots,R_k)$ are the non-negative mixing proportions satisfying $\sum_{r=1}^{R_k}\pi_{kr} = 1\ \forall k$.
- we can allow a different covariance matrix for each mixture component as well as a common covariance matrix

Illustration

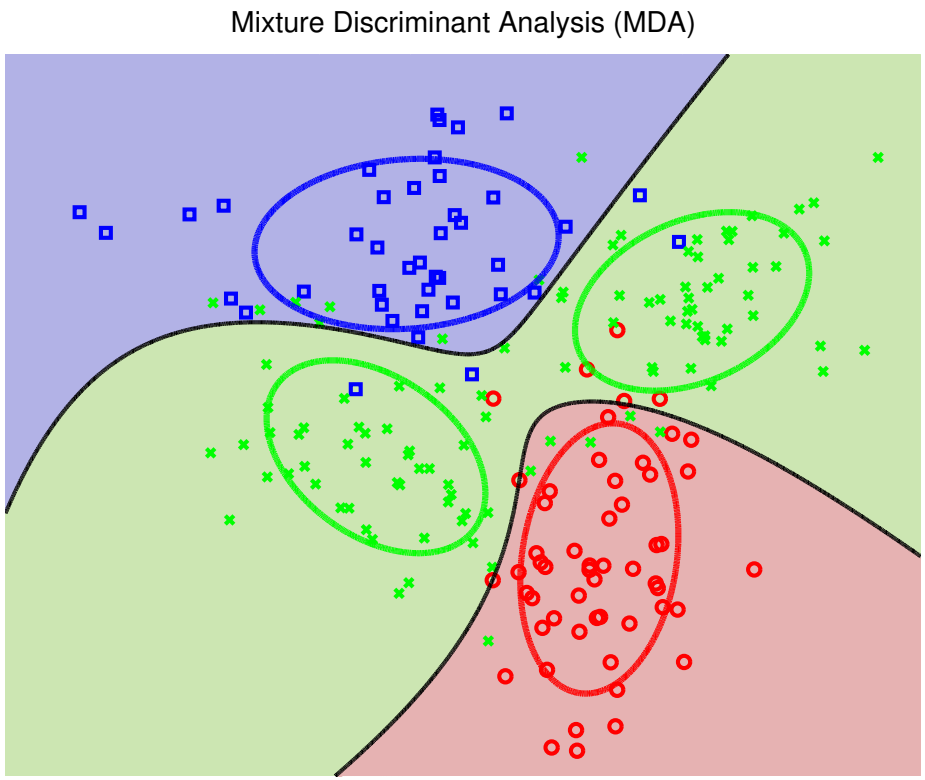


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that MDA provides more non-linear decision boundaries.

Illustrations of Logistic Regression, LDA, QDA and MDA

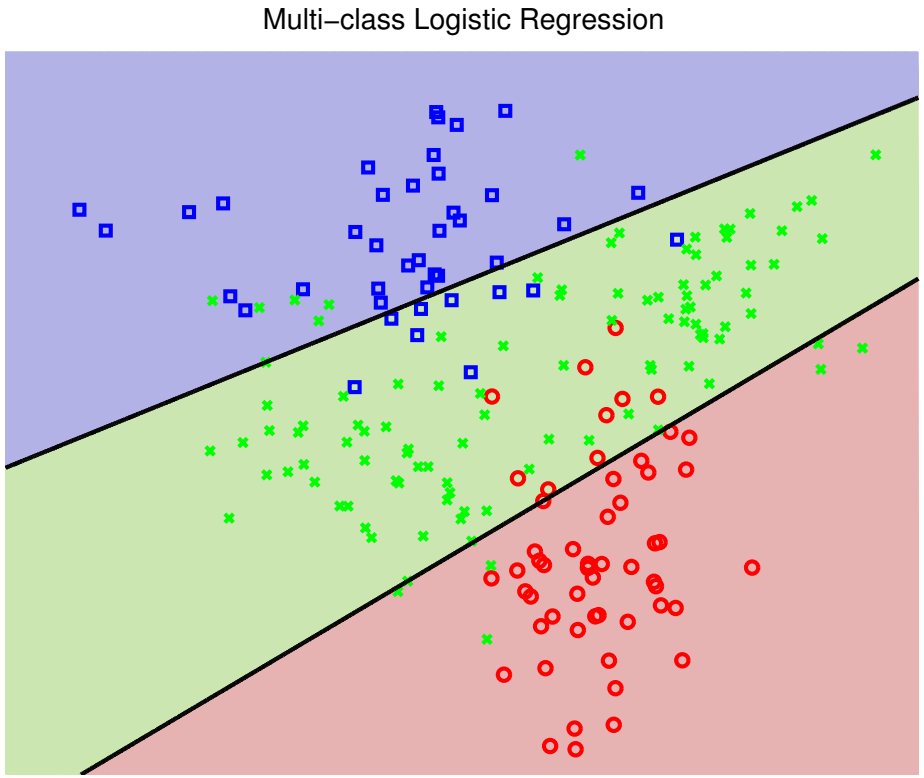


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Illustrations of Logistic Regression, LDA, QDA and MDA

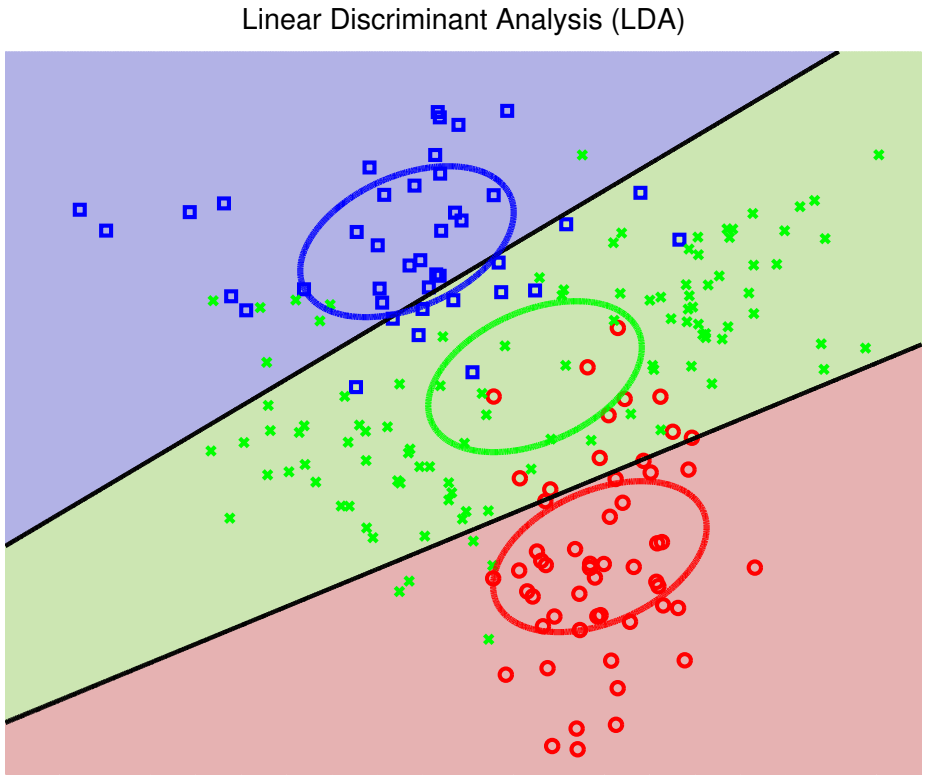


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Illustrations of Logistic Regression, LDA, QDA and MDA

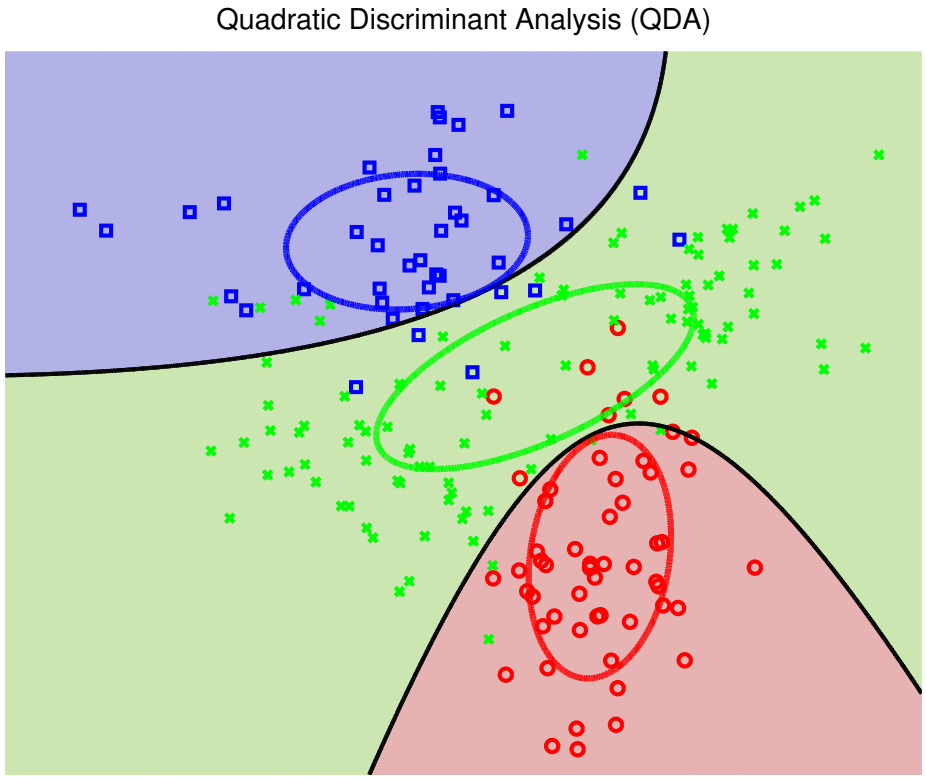


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Illustrations of Logistic Regression, LDA, QDA and MDA

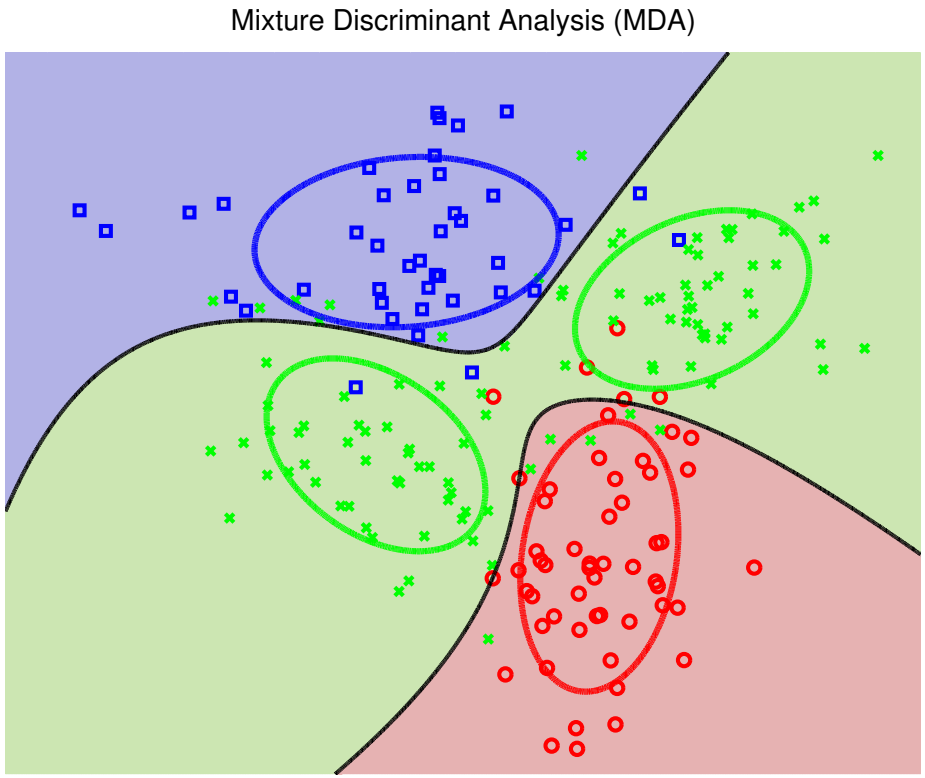


Figure : A three-classes classification example of a synthetic data set in which one of the classes occurs into two sub-classes, with training data points denoted in blue (□), green (×), and red (○). Ellipses denote the contours of the class probability density functions, lines denote the decision boundaries, and the background colors denote the respective classes of the decision regions. We see that both LDA and Logistic regression provide linear separation, while QDA and MDA provide non linear separation. MDA can further deal the problem of heterogeneous classes.

Bibliography I

Cristianini, N. and Shawe-Taylor, J. (2000). An Introduction to Support Vector Machines and other kernel-base learning methods. Cambridge University Press.

Guyon, B. B. I. and Vapnik, V. (1992). A training algorithm for optimal margin classifiers. In Haussler, I. D., editor, Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory (COLT), pages 144–152. ACM Press.

Schölkopf, B. (1997). Support Vector Learning. PhD thesis, Technischen Universität Berlin. Published by : R. Oldenbourg Verlag, Munich.

Schölkopf, B., Burges, C., and Smola, A. (1998). Introduction to support vector learning. In Schölkopf, B., Burges, C., and Smola, A., editors, Advances in Kernel Methods – Support Vector Learning, pages 1–22. MIT Press.

Schölkopf, B. and Smola, A. J. (2001). Learning with Kernels : Support Vector Machines, Regularization, Optimization, and Beyond (Adaptive Computation and Machine Learning). The MIT Press, 1st edition.

Smola, A. J., Bartlett, P., Schölkopf, B., and (Eds.), D. S. (1999). Advances in large margin classifiers.

Vapnik, V. N. (1998). Statistical Learning Theory. John Wiley & Sons.

Vapnik, V. N. (1999). The Nature of Statistical Learning Theory (Information Science and Statistics). Springer.