Università degli Studi di Padova

Department of Industrial Engineering

Machine Learning
Lesson #14

Academic year 2025-2026

Prof. Pierantonio Facco

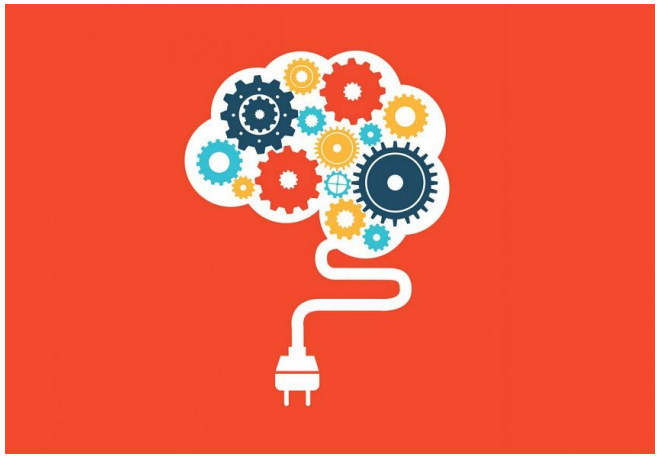
CAPE-Lab, Computer-Aided Process Engineering Laboratory

Email: pierantonio.facco@unipd.it

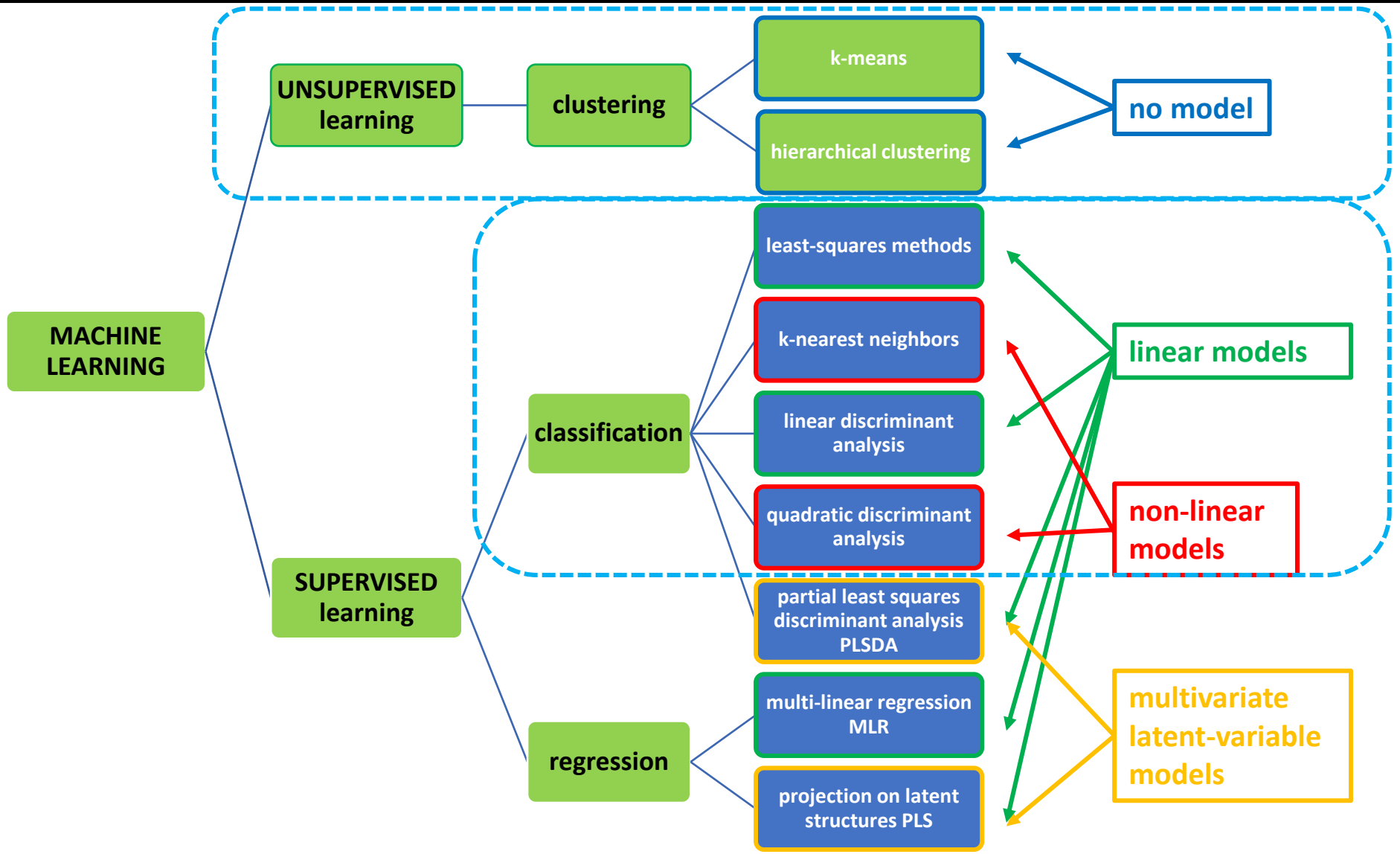
URL: https://research.dii.unipd.it/capelab/

Warm-up

- Please, connect to Kahoot!



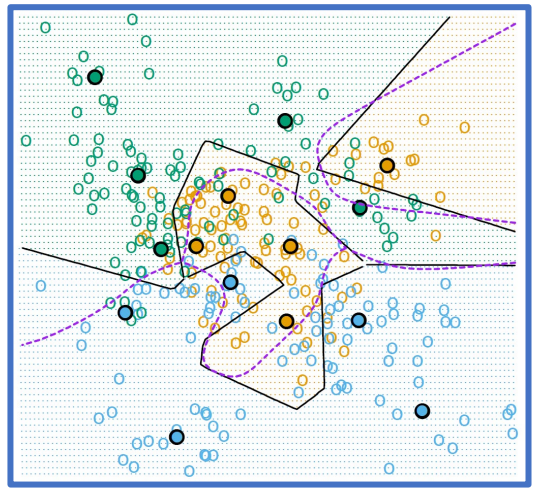
Recap and contextualization



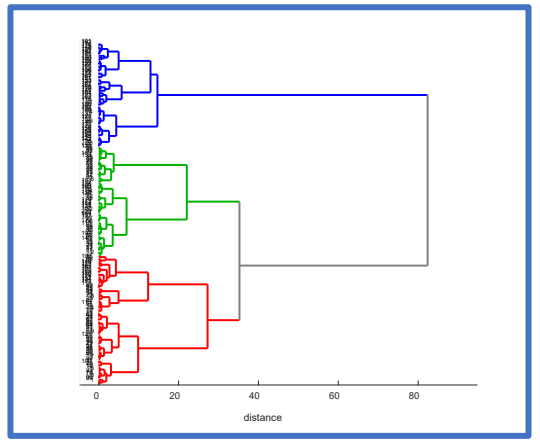
Recap on unsupervised learning

- **Unsupervised learning** is used to assess information on observations grouping based on an unsupervised **description** of the patterns which are present into the variables (also called features)
  - **non-structured models** for data segmentation
  - a measurement of data **similarity** is needed
- Methodologies:
  - ***K*-means**:
    - the **number of clusters** is selected a-priori and **prototypes** are found
    - the dissimilarity of the observation from the cluster prototypes is minimized within each cluster
  - **hierarchical clustering**:
    - a **hierarchy** of similarity is found
    - a **dissimilarity measurement** and a **linkage rule** are needed
      - agglomerative paradigm
      - divisive paradigm
    - the number of clusters is not selected a-priori
    - a **dendrogram** is used to describe data

***K*-means**



hierarchical clustering



Supervised learning

Linear models

- Given a vector of V inputs: $\mathbf{x}^{\mathrm{T}} = [x_1, x_2, \ldots, x_V]$, the output y (here a 1-element vector $\mathbf{y}$ is considered for the sake of simplicity) can be predicted by the model:

$$\hat{y} = \beta_0 + \sum_{v=1}^{V} \beta_v x_v$$

  - β_0 is the intercept (a.k.a. bias in machine learning)
  - a Matlab® command for regression coefficient estimation is: **`regress`**
    - suggestion: include the constant variable 1 in the $\mathbf{x}$ dataset to comprise β_0 in a $[(1+V)\times 1]$ vector of coefficients $\boldsymbol{\beta}$
- This model is a hyperplane in the $(V+1)$-dimensional input–output space
  - if the constant is included in $\mathbf{x}$, it is an affine set cutting the y-axis at the point $(\mathbf{0}, \beta_0)$
  - if not, the hyperplane includes the origin and is a subspace
- The linear model can be re-written in the vector form as the inner product:

$$y = \mathbf{x}^{\mathrm{T}}\boldsymbol{\beta}$$

  - this case can be extended to the case of a $[1 \times M]$ $\mathbf{y}$ vector
    - in this case the coefficients matrix **B** is a $[V \times M]$ matrix
- Viewed as a function over the V-dimensional input space:
  - $f(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\beta}$ is linear
  - the gradient $f'(\mathbf{x}) = \boldsymbol{\beta}$ is a vector in input space that points in the steepest uphill direction

Least-squares methods

- The linear model can be fitted to a set of N training data $\mathbf{X}$ and $\mathbf{y}$ through the **least squares method** ⟹ the coefficients $\boldsymbol{\beta}$ are estimated in such a way as to minimize the residual sum of squares:

$$RSS = \sum_{n=1}^{N} \left(y_n - \mathbf{x}_n^{\mathrm{T}}\boldsymbol{\beta}\right)^2$$

  - RSS is a quadratic function of the parameters, hence its minimum always exists
  - the minimum may not be unique

- The solution for a $[N \times M]$ $\mathbf{Y}$ matrix and a $[N \times V]$ $\mathbf{X}$ matrix of V predictors for N observations is found differentiating RSS:

$$RSS = (\mathbf{Y} - \mathbf{XB})^{\mathrm{T}}(\mathbf{Y} - \mathbf{XB})$$

⟹

$$\frac{\partial RSS}{\partial \mathbf{B}} = \mathbf{X}^{\mathrm{T}}(\mathbf{Y} - \mathbf{XB}) = 0$$

- If $\mathbf{X}^{\mathrm{T}}\mathbf{X}$ is nonsingular, then a unique solution exists and is:

$$\mathbf{B} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$$

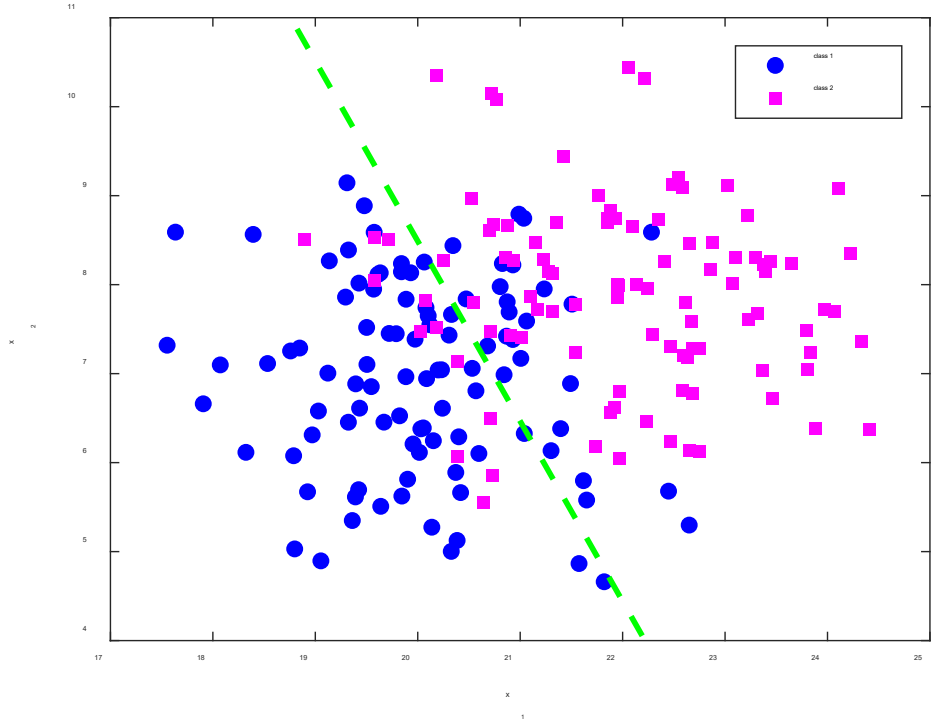
  - the estimated value of the n-th observation is:

$$\hat{y}_{n,m} = \mathbf{x}_n^{\mathrm{T}}\boldsymbol{\beta}_m$$

Least-squares method for classification

- Let's look at an example of the **linear model for classification**

- The two classes are separated by a **linear decision boundary**
  - several misclassifications are present on both sides of the decision boundary
  - is a linear decision boundary the best separation?
    - the optimal decision boundary may be nonlinear and disjoint and much more difficult to obtain
  - the region of overlap is inevitable
    - future data to be predicted will be plagued by this overlap as well



Linear classification models

- **Linear classification methods** are characterized by linear decision boundaries
- **Linear decision boundaries** can be found in several different ways:
  - fit linear regression models to the M class indicator variables
  - the fitted linear model for the m-th indicator response variable is:

$$\hat{y}_{n,m} = \mathbf{x}_n^{\mathrm{T}}\boldsymbol{\beta}_m$$

  - the decision boundary between class m_1 and m_2 is the set of points for which $\hat{y}_{n,m_1} = \hat{y}_{n,m_2}$ that is the set of x variables that satisfies:

$$\mathbf{x}_n^{\mathrm{T}}\boldsymbol{\beta}_{m_1} - \mathbf{x}_n^{\mathrm{T}}\boldsymbol{\beta}_{m_2} = 0$$

    - this is an **affine set/hyperplane**
    - the same is true for any pair of classes

  - the input space is divided into regions of constant classification, with piecewise hyperplanar decision boundaries

Regression models for classification

- This **regression approach** is a member of a class of methods which:
  - model discriminant functions for each class
  - then classify a new observation $\mathbf{x}_{\mathrm{NEW}}$ to the class with the largest value for its discriminant function
- The most direct approach is to explicitly **model the boundaries** between the classes as **linear**
  - for a two-class problem in a V-dimensional input space, this amounts to modeling the decision boundary as a hyperplane (i.e., a normal vector and a cut-point)
  - different methods are used to find "separating hyperplanes"
- The variable set $[x_1, x_2, \ldots, x_V]$ can be also expanded by including their squares and cross-products, thereby adding $V(V+1)/2$ additional variables to obtain **quadratic decision boundaries**

Classes in the responses, i.e., Y variables

- What is, in your opinion, the best way to express the responses in the $\mathbf{Y}$ matrix?

Coded responses for classification problems

- Each of the response categories are **coded** through an **indicator (i.e., dummy) variable**
  - if the response has M classes, for each observation n there will be M indicators $y_{n,m}$ for $m = 1, 2, \dots, M$, with:
    - $y_{n,m} = 1$ if the sample n belongs to the m-th class
    - $y_{n,m} = 0$ otherwise
  - these class indicators are collected together in a vector $\mathbf{y}_n = \left[y_{n,1} y_{n,2} \ \dots y_{n,M}\right]$ for a single observation n
  - the N training observations form an $[N \times M]$ $\mathbf{Y}$ indicator response matrix
    - $\mathbf{Y}$ is a matrix of 0's and 1's, with each row having a single 1

Classification through linear regression

- A **linear regression model** to each of the columns of $\mathbf{Y}$ is fitted simultaneously:

$$\mathbf{Y} = \mathbf{X}\mathbf{B}$$

  - one coefficient vector is present for each response column $\mathbf{y}_m$
  - hence a $[(V+1) \times M]$ $\mathbf{B}$ coefficient matrix is adopted, where:

$$\mathbf{B} = \left(\mathbf{X}^{\mathrm{T}}\mathbf{X}\right)^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{Y}$$

  - $\mathbf{X}$ is the model matrix with $V+1$ columns corresponding to the V inputs, and a leading column of 1's for the intercept

- A **new observation** with input $\mathbf{x}_{\mathrm{NEW}}$ is classified as follows:

1. compute the fitted output $\hat{\mathbf{y}}_{\mathrm{NEW}}^{\mathrm{T}} = \left[1\ \mathbf{x}_{\mathrm{NEW}}^{\mathrm{T}}\right]\mathbf{B}$, a vector of M elements
2. identify the largest component and classify accordingly:

$$\hat{y}_{\mathrm{NEW}} = \max_{M}(\hat{\boldsymbol{y}}_{\mathrm{NEW}})$$

Characterization of the classification performance

- Confusion matrix

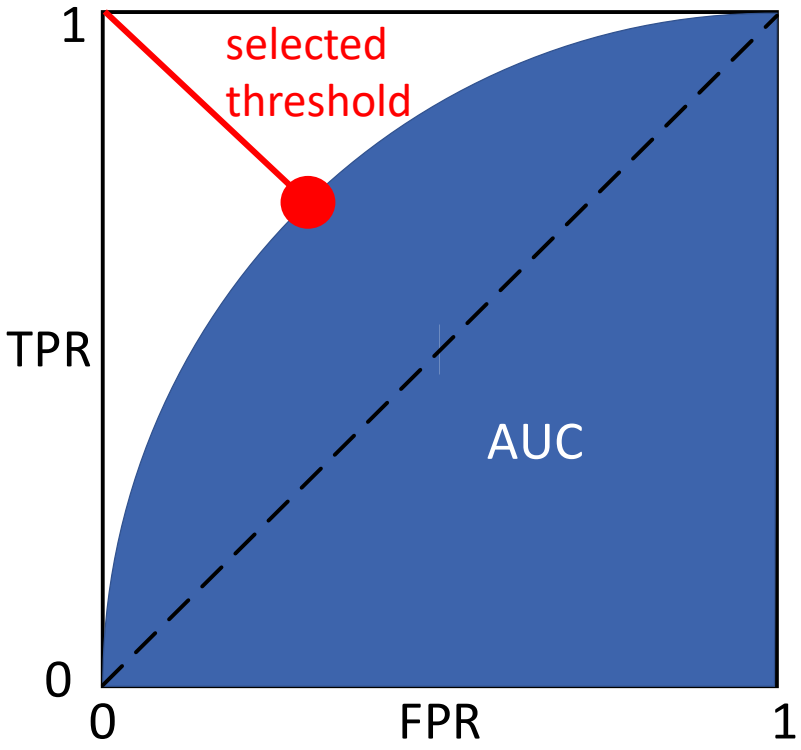
real \ predicted	negative	positive
negative	TN true negative	FP false positive
positive	FN false negative	TP true positive

- Accuracy (% correct classifications) $A = \frac{TP+TN}{N}$
- Precision $P = \frac{TP}{TP+FP}$
- Recall (true positive rate/sensitivity) $R = TPR = \frac{TP}{TP+FN}$
- False positive rate (fall out) $FPR = \frac{FP}{FP+TN}$
- Specificity (selectivity) $TNR = \frac{TN}{FP+TN}$
- F1 score $F1 = 2\frac{P \cdot R}{P+R}$

Receiver operating curve and area under the curve

- **Receiver operating curve** (**ROC curve**) illustrates the diagnostic ability of a binary classifier system as its discrimination threshold is varied
  - recall/true positive rate/sensitivity on y axis
$$R = TPR = \frac{TP}{TP + FN}$$
  - false positive rate on x axis FP
$$FPR = \frac{FP}{FP + TN}$$
  - diagonal black dashed line indicates a random classifier

- **Area Under the Curve** (**AUC**) is an index of classification performance (ratio between the area under the ROC curve and the total area)
  - $AUC \in [0,1]$
  - $AUC = 0.5$ random classifier
  - $AUC \sim 1$ very good classifier
    - high TPR
    - low FPR



k-Nearest Neighbor

Nearest-neighbor methods

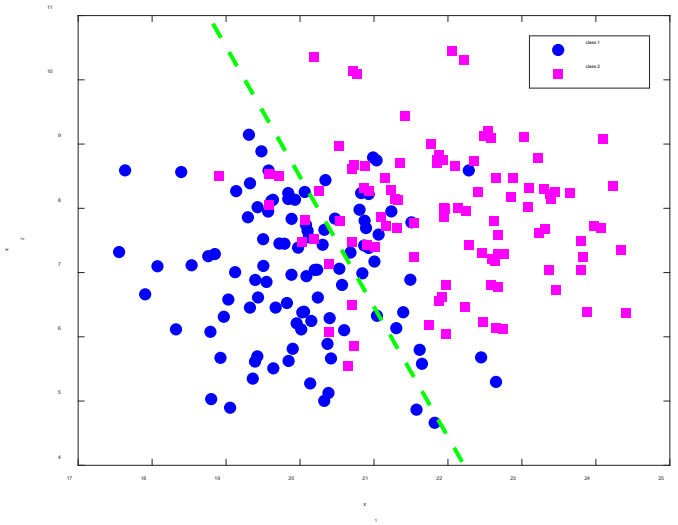
- ***K*-nearest neighbor** uses the K observations in the training set that are the closest in the input space to estimate $\hat{y}$:

$$\hat{y} = \frac{1}{K} \sum_{\mathbf{x}_n \in N_K(\mathbf{x})} y_k$$

  - $N_K(\mathbf{x})$ is the neighborhood in the input space defined by the K closest points in the training samples
  - find the K observations which are the closest to the new observation in the input space and average their responses y_k
- **Closeness** implies a distance metric (and identifies a measurement of similarity)
  - Euclidean distance can be used to this purpose
- In **classification** the *K*-nearest neighbor responses are averaged for the binary coded response (namely, y's that are zeros or ones)
  - thus $\hat{y}$ is the proportion of one class in the predictors' neighborhood
    - if $\hat{y} \geq 0.5$, the observation is attributed to the class of the 1s, to the class of 0s otherwise
- The **decision boundaries** that separate the regions of the classes are **irregular** and respond to local clusters where one class dominates

Nearest-neighbor methods vs. linear boundary

linear regression method



***K*-nearest neighbors**

Nearest neighbor for classification

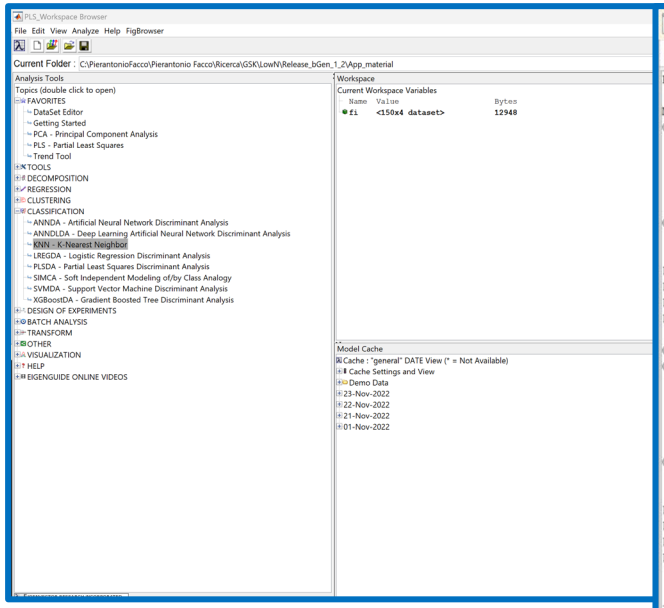
- Much fewer training observations are usually misclassified with nearest neighbor
  - for *K*-nearest neighbor the error of classification on the training data e should be approximately an increasing function of K and is $e = 0$ for $K = 1$
  - independent test set would give a more satisfactory means for comparing the different methods
- It appears that *K*NN has a single parameter (i.e., the number of neighbors K) compared to the V parameters in least-squares fits
  - however, the effective number of parameters of *K*NN is N/K and is generally $> V$
  - note that if the neighborhoods were non-overlapping, there would be N/K neighborhoods, and we would fit one parameter (a mean) in each neighborhood
- Sum-of-squared errors cannot be used as a criterion for selecting K on the training set
  - it would always pick $k = 1$ in *K*NN!
- It seems that *K*NN is more appropriate for *scenarios of mixtures of Gaussian populations*
  - for two classes of multi-normally distributed data the decision boundaries would be unnecessarily noisy

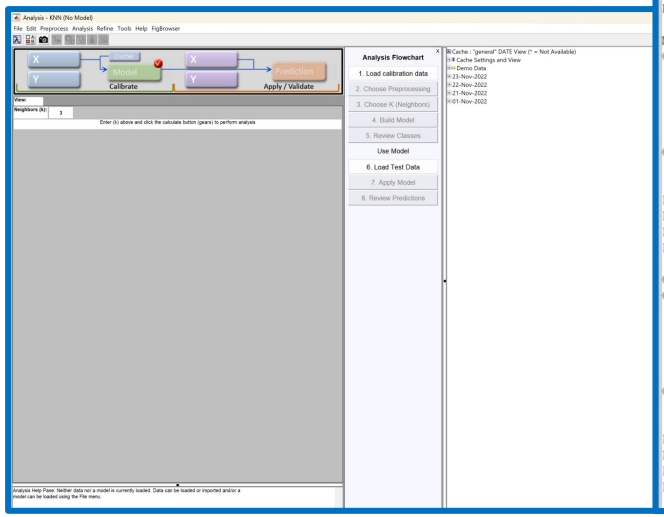
Least squares vs. nearest neighbor

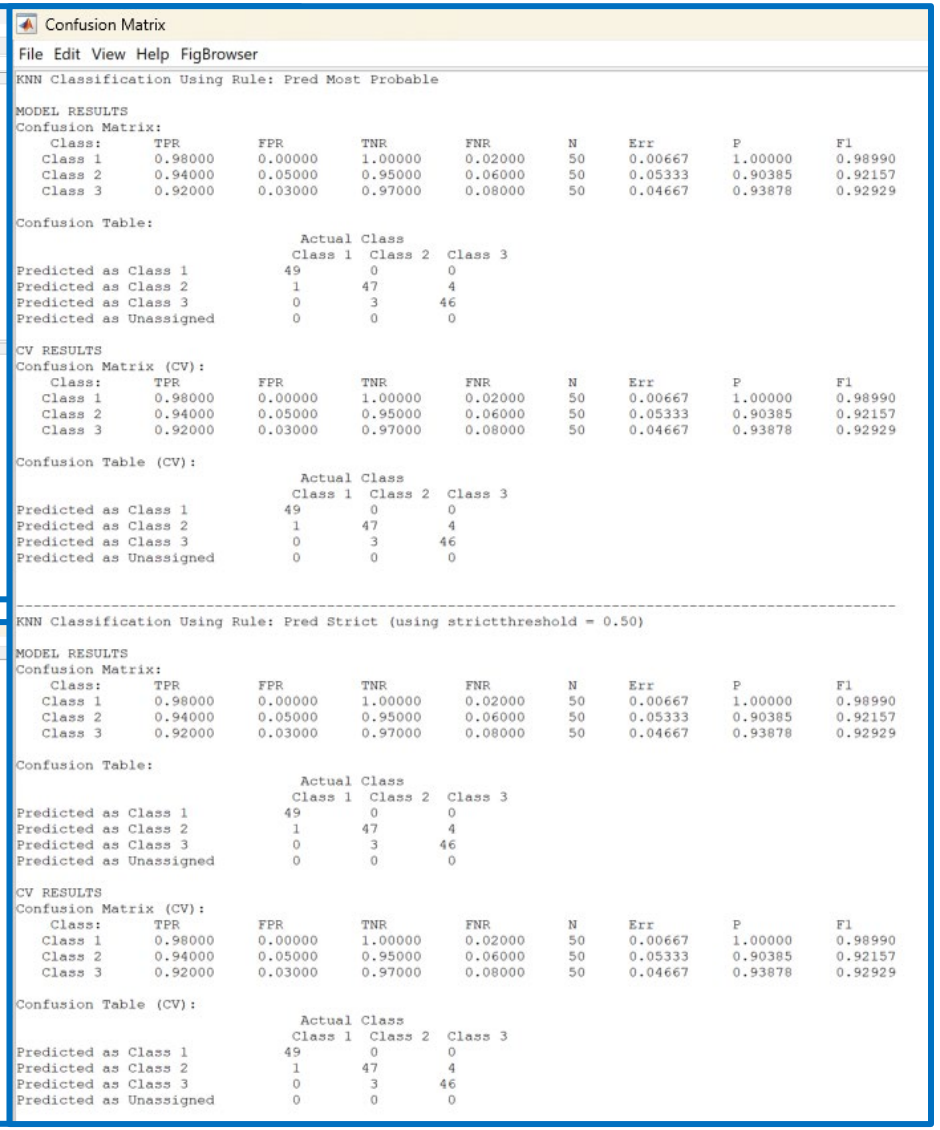
- Least squares rely heavily on the assumption that a **linear decision boundary** is appropriate
  - the linear decision boundary from least squares is very smooth, and apparently stable to fit
  - the method has low variance and potentially high bias
- *K*NN do not rely on any stringent **assumptions** about the underlying data and can adapt to any situation
  - any sub-region of the decision boundary depends on a handful of input points and their positions
  - it is wiggly and unstable
  - the method has high variance and low bias
- Each method has its own situations for which it works best:
  - linear regression is more appropriate for scenarios of bivariate Gaussian distributions of independent components for two classes
  - *K*NN is more suitable for scenarios of mixtures of Gaussian distributions

Example of *k*-NN: Fisher's iris

- In PLS_Toolbox®:
  - **Browse**
  - select **Classification**
  - select **KNN**
  - **File**
  - **Load data**
  - **X-Block**
  - Select **fi**
  - Click OK
  - Click **Perform KnnClassification Analysis** button







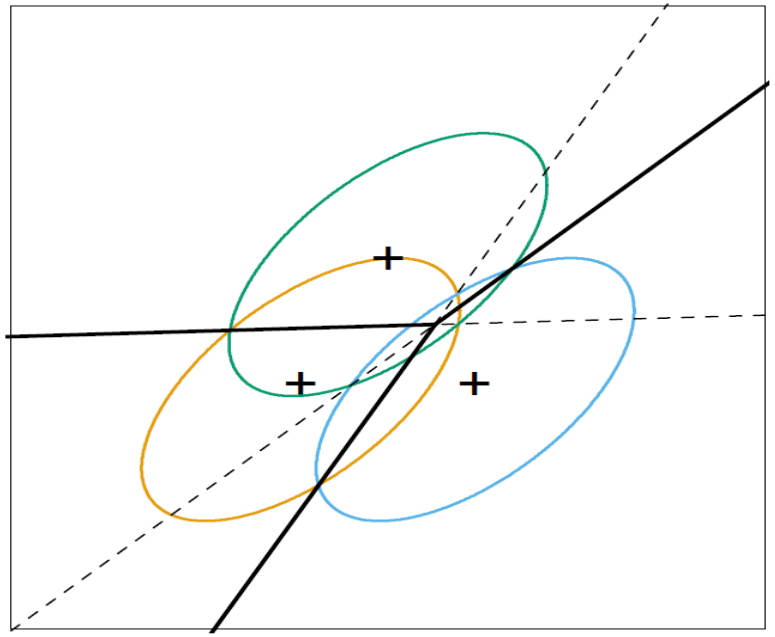
Linear and quadratic discriminant analysis

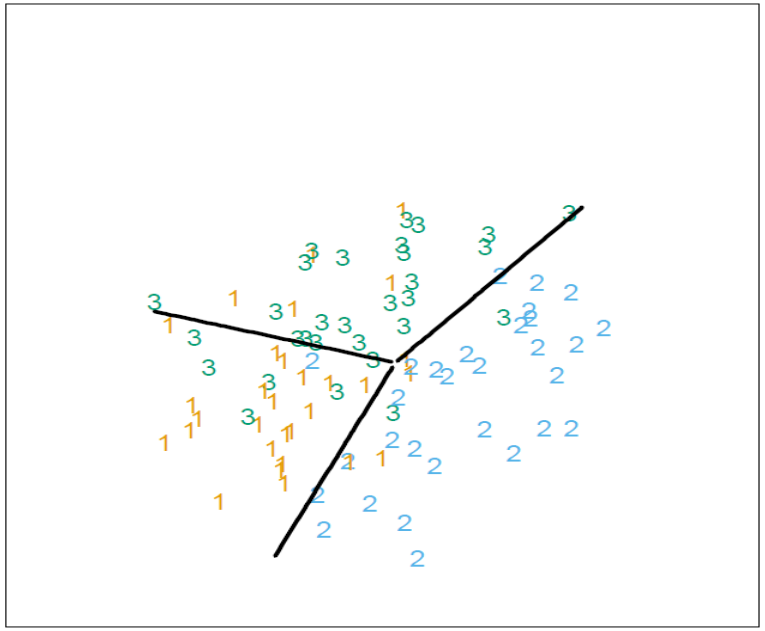
Linear Discriminant Analysis LDA

- Suppose to have a multivariate Gaussian distribution (for $m = 1, 2, \dots, M$):

$$f_m(x) = \frac{1}{(2\pi)^{V/2}|\Sigma_m|^{1/2}} e^{-(\mathbf{x}-\boldsymbol{\mu}_m)^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}(\mathbf{x}-\boldsymbol{\mu}_m)}$$

- **Linear discriminant analysis LDA** arises in the special case when **the classes have the same covariance matrix** $\Sigma_m = \Sigma$
- Notice that the decision boundaries are not the perpendicular bisectors of the line segments joining the centroids
  - this would be the case if the covariance were spherical





Linear discriminant functions

- The linear discriminant functions (in $\mathbf{x}$) which defines the likelihood of attributing a sample to a class is:

$$\delta_m(\mathbf{x}) = \mathbf{x}^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_m - \frac{1}{2}\boldsymbol{\mu}_m^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_m + \log\pi_m$$

- where the sample parameters are calculated using the training data:
  - $\pi_m = \frac{N_m}{N}$ and N_m is the number of class-m observations
  - $\boldsymbol{\mu}_m = \sum_m \frac{\mathbf{x}_i}{N_m}$
  - $\boldsymbol{\Sigma} = \sum_m \sum_m \frac{(\mathbf{x}_i - \boldsymbol{\mu}_m)(\mathbf{x}_i - \boldsymbol{\mu}_m)^T}{N - m}$
- The **decision rule** is:

this is equivalent to a SVD of $\mathbf{S}_{\mathrm{W}}^{-1}\mathbf{S}_{\mathrm{B}}$, ratio between the scatter $\mathbf{S}_{\mathrm{B}}$ between classes and the one within each class $\mathbf{S}_{\mathrm{W}}$

$$G(\mathbf{x}) = \max_m\left(\delta_m(\mathbf{x})\right)$$

- The **LDA rule** classifies to class 2 if:

$$\mathbf{x}^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}_2 - \boldsymbol{\mu}_1) > \frac{1}{2}\boldsymbol{\mu}_2^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_2 - \frac{1}{2}\boldsymbol{\mu}_1^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu}_1 + \log\frac{N_1}{N} - \log\frac{N_2}{N}$$

  - class 1 otherwise
- With two classes there is a simple correspondence between linear discriminant analysis and classification by linear least squares

Quadratic discriminant analysis QDA

- The derivation of the LDA direction via least squares **does not use a Gaussian assumption** for the features
  - its applicability extends beyond the realm of Gaussian data
  - the derivation of the intercept or cut-point requires Gaussian data
  - it makes sense to choose the cut-point that empirically minimizes training error for a given dataset
    - this works well in practice, but it is not mentioned in the literature
- *With more than two classes, LDA is not the same as linear regression of the class indicator matrix*
  - it avoids the masking problems associated with that approach
- If the $\boldsymbol{\Sigma}_m$ are not assumed to be equal, then the quadratic terms in $\mathbf{x}$ remain and the quadratic discriminant functions of **quadratic discriminant analysis QDA** are:

$$\delta_m(\mathbf{x}) = -\frac{1}{2}\log|\boldsymbol{\Sigma}_m| - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_m)^{\mathrm{T}}\boldsymbol{\Sigma}_m^{-1}(\mathbf{x}-\boldsymbol{\mu}_m) + \log\pi_m$$

  - the decision boundary between each pair of classes is described by a quadratic equation

Practical implementation of LDA and QDA

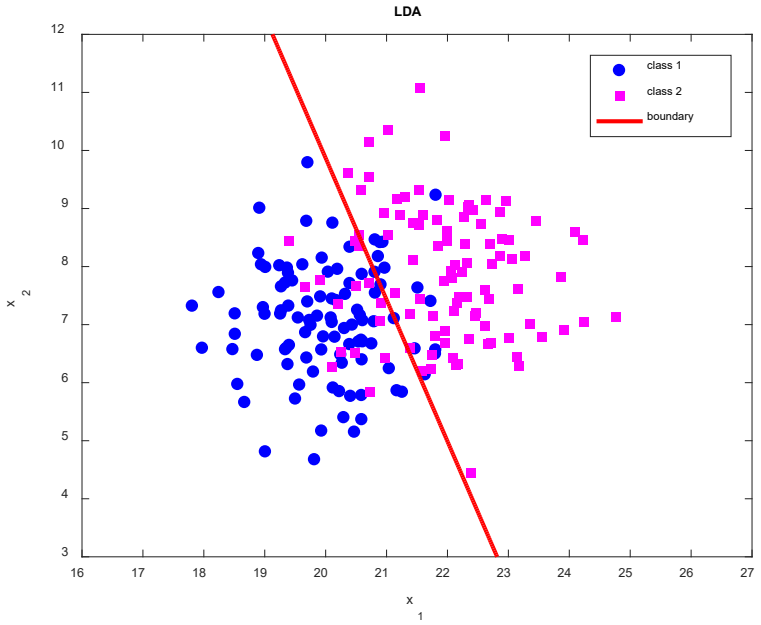
- Matlab® command for both LDA and QDA: **`fitcdiscr`**
  - example of LDA:

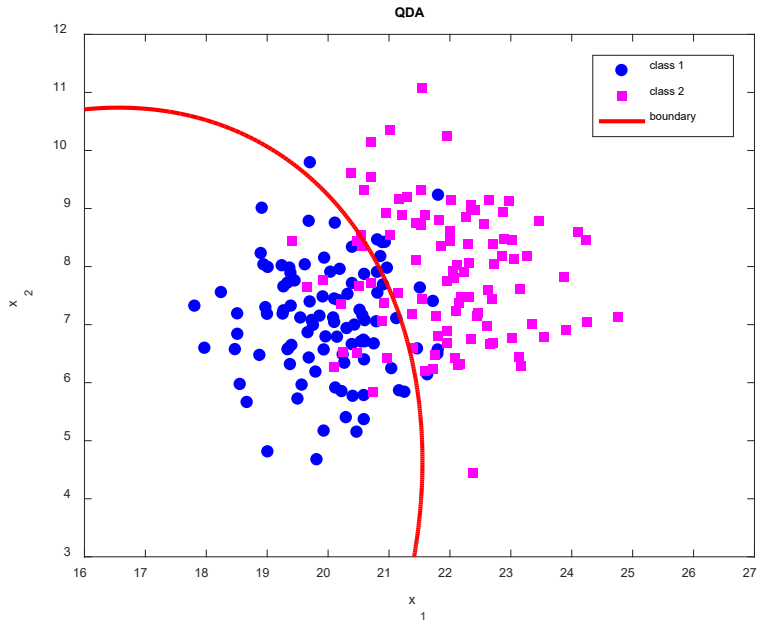
```
lda_model=fitcdiscr(X,Y,'DiscrimType','linear','FillCoeffs','on')
```

  - example of QDA:

```
qda_model=fitcdiscr(X,Y,'DiscrimType','quadratic','FillCoeffs','on')
```

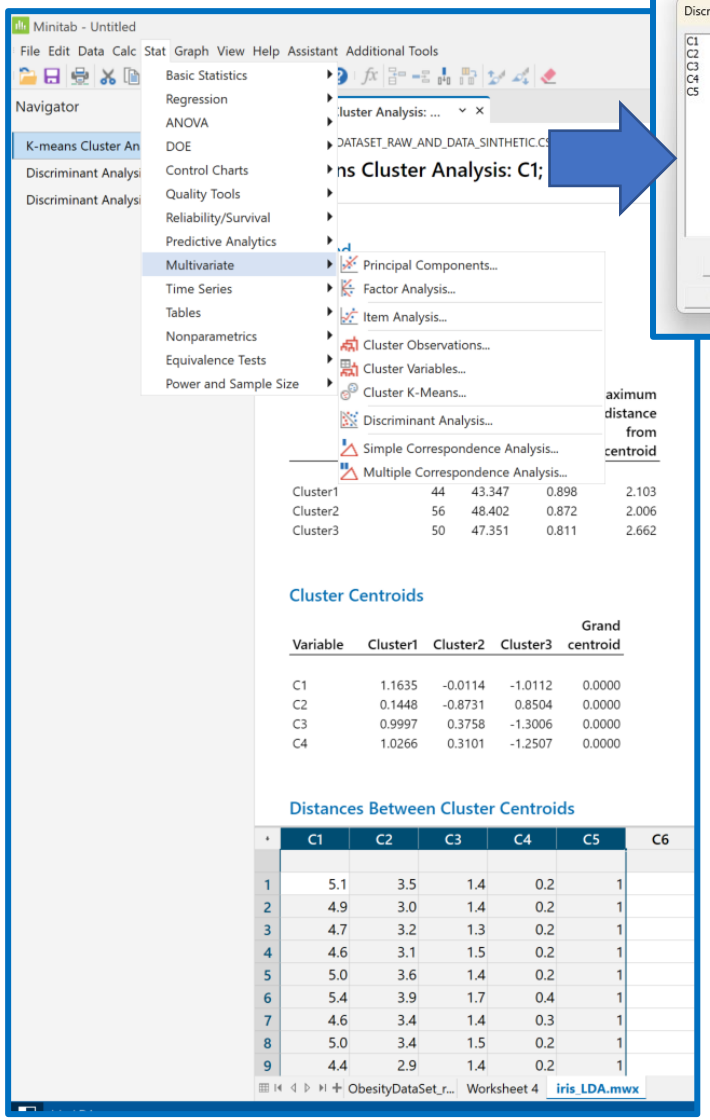
    - pay attention that in the model object the coefficients of the discriminant boundary can be found in the folder **`.coeff`**

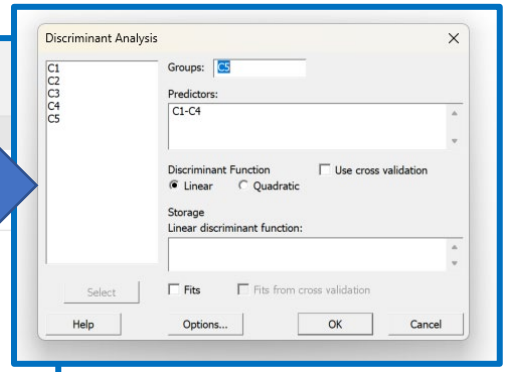




Example of LDA and QDA: Fisher's iris

- In Minitab®:
  - **Stat**
  - **Multivariate**
  - **Discriminant analysis**
  - select **Groups**
  - select **Predictors**
  - select **Linear** or **Quadratic**
  - click OK





Squared Distance Between Groups

	1	2	3
1	0.000	89.864	179.385
2	89.864	0.000	17.201
3	179.385	17.201	0.000

Linear Discriminant Function for Groups

	1	2	3
Constant	-85.21	-71.75	-103.27
C1	23.54	15.70	12.45
C2	23.59	7.07	3.69
C3	-16.43	5.21	12.77
C4	-17.40	6.43	21.08

Summary of Classification

	True Group		
Put into Group	1	2	3
1	50	0	0
2	0	48	1
3	0	2	49
Total N	50	50	50
N correct	50	48	49
Proportion	1.000	0.960	0.9

Correct Classifications

N	Correct	Proportion
150	147	0.980

Summary of Misclassified Observations

Observation	True Group	Pred Group	Group	Squared Distance	Probability
71**	2	3	1	130.862	0.000
			2	8.670	0.253
			3	6.507	0.747
84**	2	3	1	149.030	0.000
			2	8.439	0.143
			3	4.864	0.857
134**	3	2	1	133.067	0.000
			2	5.253	0.729
			3	7.236	0.271

Today homework

- Practice with *k*NN, LDA and QDA on the Fisher iris and the ovarian cancer dataset:
  - consider Matlab® commands:
    - `knn` or `fitcknn`
    - `fitdisrc`
- Compare the results with those you obtained with *k*-means and hierarchical clustering

Notice

- **Lesson #16** is a flipped one, its video is available on Moodle
  - complete the following activities:
    1. **attend the video lesson**
    2. **self-assess your learning**
    3. **re-read the following papers** available in the “Suggested reading” in Moodle:
       - Geladi, P., Kowalski, B.R. (1986). Partial least-squares regression: a tutorial. *Anal. Chim. Acta,* **185**, 1–17
       - Wise, B.M., Gallagher, N.B. (1996). The process chemometrics approach to process monitoring and fault detection. *J. Process Control,* **6**, 329–348
    4. **prepare questions** and anything you need to discuss with the teacher and your mates in the next lecture
  - the following lecture will be:
    - 1/2 Q&A: questions (of the students) and answers (of the teacher)
    - 1/2 we will start the DoE part

… per sempre a fianco a me!