Lecture01: An Introduction to Machine Learning

- Course and textbook objective: Democratizing machine learning with commonly accessible soft- and hardware and regain control over our destiny rather than letting robots or artificial intelligence (AI) agents rule us.
- The job of robots and AI agents is just to make our life easier.
- However, we are running the risk of becoming too dependent on them.

Useful Websites

- https://www.r-bloggers.com/in-depth-introduction-to-machine-learning-in-15-hours-of-expert-videos/
- https://cran.r-project.org/web/views/MachineLearning.html
- www.kaggle.com

Historical Background

- AI objective: Perform predictions of outcomes y or future actions by using specific AI rules (i.e., functions $f(x)$) based on observable information x.

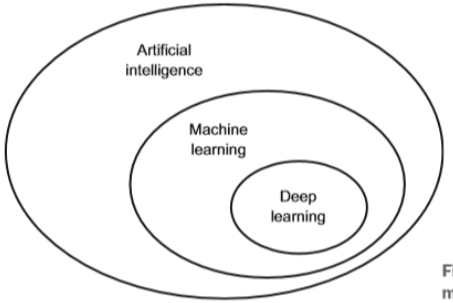


Figure 1.1 Artificial intelligence, machine learning, and deep learning

Artificial Intelligence

- In the 1950 computer pioneers start asking the question “Can computers think?”.
- Ultimately its definition was reduced to automating intellectual tasks normally performed by humans.
- The tasks were hardcoded as “rules” and lead to symbolic AI and expert systems in the 1980.
- AI able to solve well-defined logical problems but it cannot hand complex and fuzzy problems.

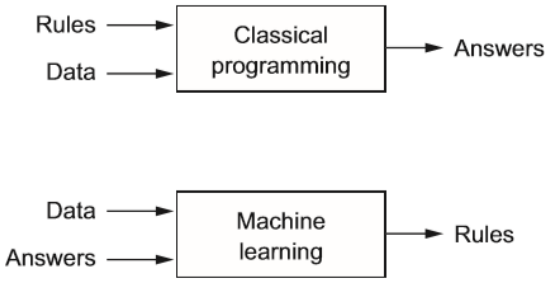


Figure 1.2 Machine learning: a new programming paradigm

Machine Learning

- ML started in the 1990s asking the question “can a computer discover hidden rules by processing data?”
- In ML humans input data and answers, whereas the computer finds rules connecting data with answers.
- This is achieved by training the “machine”.
- Once rules are established with some confidence, they can be used to perform predictions of answers based on input data.
- ML is tightly related to statistics but applies it on large and complex data sets for which standard statistical models would be too rigid.

- It is applied in big data analysis which are characterized by [a] volume, [b] variety, [c] velocity, and veracity (that is, certainty or the lack thereof).
- Criticism of ML: it is data drive and ideas are "proven" empirically rather than being rooted in theory.

Deep Learning

- It is a specialized subfield of ML
- With the increase in affordable computing power and an increasing availability of data deep learning network became prominent in the early 2010.
- It is based on successive layers of rules with increasingly more meaningful representations of the outcome y.
- "Deep" does not refer to a deeper understanding of the task.
- Deep learning is based on the mathematical framework of neural networks but does not have any relationship to neurobiology (i.e., neurobiology studies how our brains work).
- One can think of neural network as a series of filters (the layers) that distill the input information x into a purified version that is closely related to the output y.
- Building blocks of understanding neural networks are logistic regression and gradient descent optimizers.
- A sequence of layers weight feature input vectors $\mathbf{x}_i$ that will generate an output label y_i: $y_i = f_{NN}(\mathbf{x}_i) = f_3\big(f_2(f_1(\mathbf{x}_i))\big)$.
- Deep learning involves more than two non-output layers. In $y_i = f_{NN}(\mathbf{x}_i) = f_3\big(f_2(f_1(\mathbf{x}_i))\big)$ the function f_3 is the output layer.

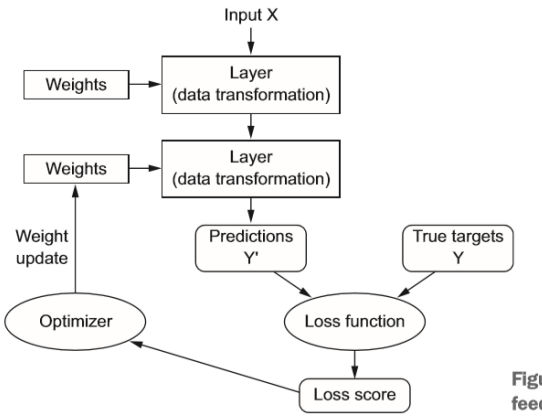


Figure 1.9 The loss score is used as a feedback signal to adjust the weights.

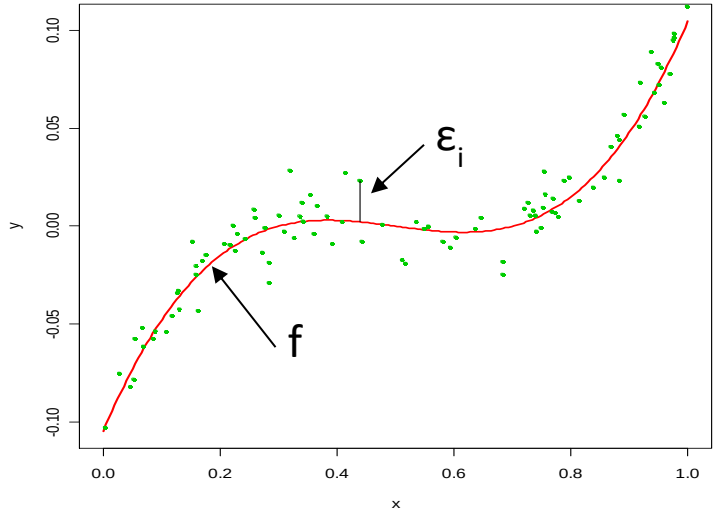
- The objective of specifying the layer functions $f_l(\quad)$ is to minimize the prediction error, which is measured by a loss score functions.

Fundamental Setting of Supervised ML

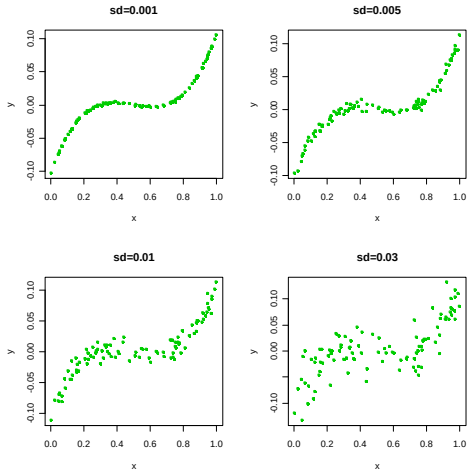
Some of the figures in this presentation are taken from Chapter 2 of "An Introduction to Statistical Learning, with applications in R (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani "

- In supervised learning both the outcome y_i and the features $\mathbf{x}_i$ of object i are observable.
- The outcome and the features are connected by a *true* function $f(\quad)$ plus an unknown random error ε_i:

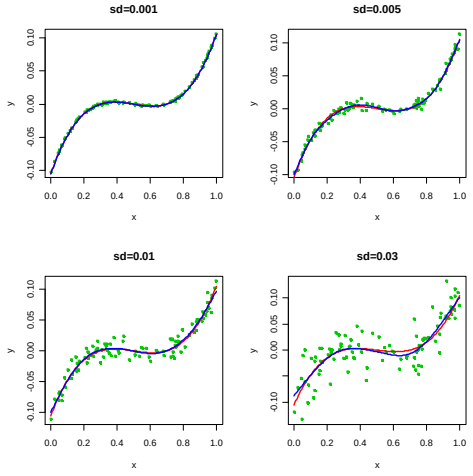
$$y_i = f(\mathbf{x}_i) + \varepsilon_i$$



- As the variance of the error increases the exact function form of $f(\quad)$ becomes more and more uncertain.



- There are different estimates $\hat{f}(\quad)$ for true but unknown $f(\quad)$



- In supervised learning a training sample with $(y_i, \mathbf{x}_i)$ is used to obtain an "accurate" estimate $\hat{f}(\quad)$ of $f(\quad)$.
- At test sample $(y_0, \mathbf{x}_0)$ is usually withheld to evaluate the accuracy of the estimator $\hat{f}(\quad)$
- Since many different estimators $\hat{f}(\quad)$ of varying degrees of flexibility can be specified there is usually a trade-off between prediction accuracy and model generality.
- Overfitting leads to estimators $\hat{f}(\quad)$ well adapted a particular given sample and its specific random error ε but they do not generalized to another sample which will have a different random error ε.
  → There is no *free lunch* in statistics: no one estimation model dominates all others over all possible data sets.

Evaluating the Model Fit

- Mean-Square-Error for training and test data sets: $MSE = \frac{1}{n} \cdot \sum_{i=1}^{n} \left(y_i - \hat{f}(\mathbf{x}_i) \right)^2$
- Other goodness of fit measures, such as the "confusion" matrix and ROC (receiver operating characteristics) are used for classification problems.
- Medium use of estimated parameters:

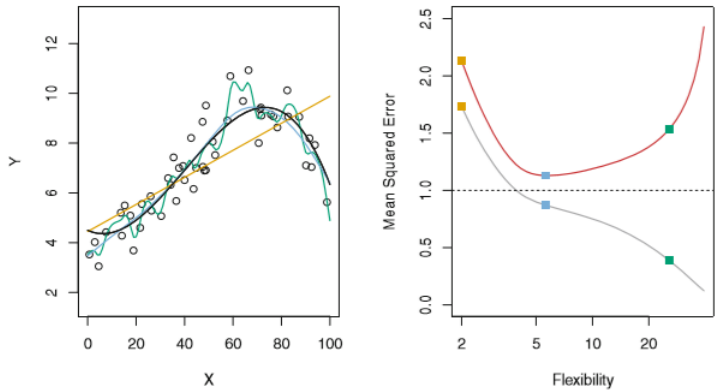


FIGURE 2.9. Left: *Data simulated from f, shown in black. Three estimates of f are shown: the linear regression line (orange curve), and two smoothing spline fits (blue and green curves).* Right: *Training MSE (grey curve), test MSE (red curve), and minimum possible test MSE over all methods (dashed line). Squares represent the training and test MSEs for the three fits shown in the left-hand panel.*

- Black line is the data generating process (DGP) with a random error. Right dashed line is the MSE associated with the disturbances.
- Trade-off between Variance and Bias
- Decomposition of expected squared prediction error $E\left(y_0 - \hat{f}(x_0)\right)^2 = Var\left(\hat{f}(x_0)\right) + \left[Bias\left(\hat{f}(x_0)\right)\right]^2 + Var(\varepsilon)$

$E\left(y_0 - \hat{f}(x_0)\right)^2$	Expected MSE over all possible test samples (y_0, x_0)
$Var\left(\hat{f}(x_0)\right)$	Variation of the *estimated* function $\hat{f}$ based on the use of different training samples. Simpler models are more stable.

$[Bias(\hat{f}(x_0))]^2$	This variation error is introduced by mis-specifying the true model of the DGP. Under-fitting a simple model to more complex data variations is a certain cause. A proper selected model will have no bias.
$Var(\varepsilon)$	Irreducible error variation. Overfitting model try to capture this random variation.

- General rule: As the model becomes more flexible the bias will decrease and the variance will increase.
- Low use of paramters:

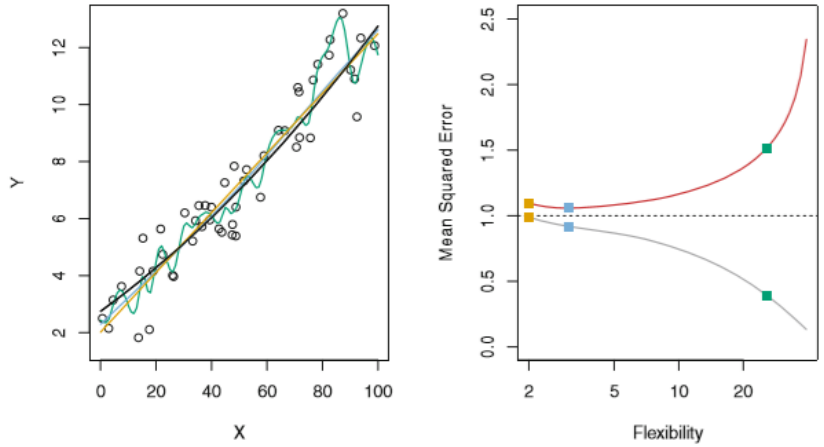


FIGURE 2.10. *Details are as in Figure 2.9, using a different true f that is much closer to linear. In this setting, linear regression provides a very good fit to the data.*

- High use of estimated parameters

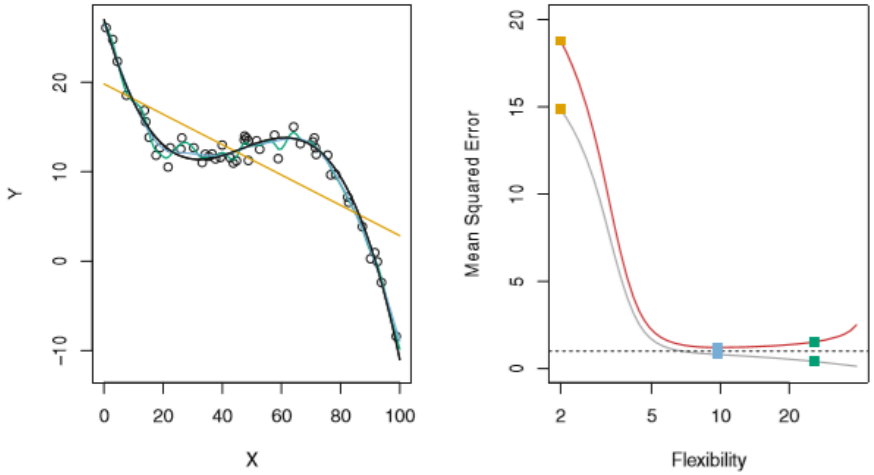


FIGURE 2.11. *Details are as in Figure 2.9, using a different f that is far from linear. In this setting, linear regression provides a very poor fit to the data.*

- Effect of the model specification on variance and bias in dependence of the model complexity:

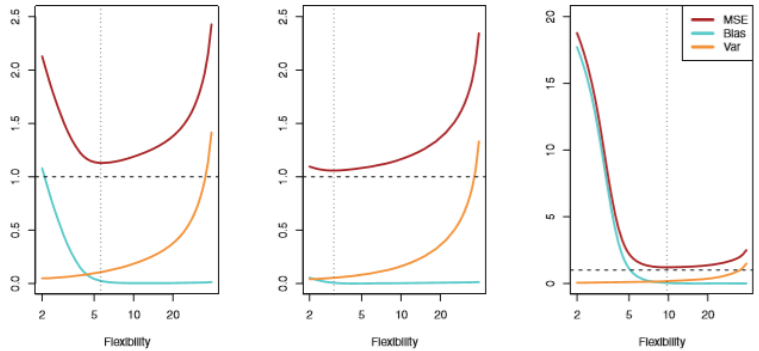


FIGURE 2.12. *Squared bias (blue curve), variance (orange curve), Var(ϵ) (dashed line), and test MSE (red curve) for the three data sets in Figures 2.9–2.11. The vertical dotted line indicates the flexibility level corresponding to the smallest test MSE.*

Classification: Bayesian k-Nearest Neighbor Class Assignment

- In the classification setting the outcome variable is discrete, that is,

$$y_i = k \text{ if object } i \text{ belongs to the } k^{th} \text{ class}$$

- The error rate, which we aim to minimize, becomes $\frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$, where the indicator function $I(\quad) = 1$ if the condition is true and otherwise it is 0.
- In a Bayesian classifier we assign an observation i to the class k if its conditional probability $\Pr(Y = k|X = \mathbf{x}_0)$ is larger than for any other class.
  For just two classes the probability has to be $\Pr(Y = k|X = \mathbf{x}_0) > 0.5$.

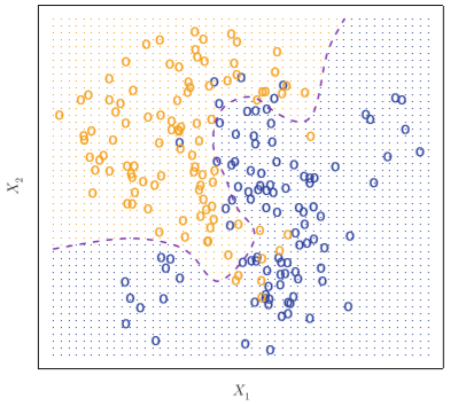


FIGURE 2.13. *A simulated data set consisting of 100 observations in each of two groups, indicated in blue and in orange. The purple dashed line represents the Bayes decision boundary. The orange background grid indicates the region in which a test observation will be assigned to the orange class, and the blue background grid indicates the region in which a test observation will be assigned to the blue class.*

- The Bayesian class membership probabilities in K-nearest neighborhood (kNN) is calculated as:

- Determine those K training points which are the closest to the prediction point $\mathbf{x}_0$.
  Closeness is measured in terms of the Euclidian distance between two objects i and j with their feature vectors $\mathbf{x}_i = (x_{i1}, x_{i2}, \dots, x_{iP})^T$ and $\mathbf{x}_j = \left(x_{j1}, x_{j2}, \dots, x_{jP}\right)^T$ by

$$d_{ij} = \sqrt{\sum_{p=1}^{P} \left(x_{ip} - x_{jp}\right)^2} \ .$$

- Subsequently estimate the Bayesian group probabilities by $\Pr(Y = k|X = \mathbf{x}_0) = \frac{1}{K}\sum_{i\in\mathcal{N}_0} I(y_i = k)$, where $\mathcal{N}_0$ is the neighborhood around $\mathbf{x}_0$ with K training points.
- Assign the observation to class j with the highest probability.

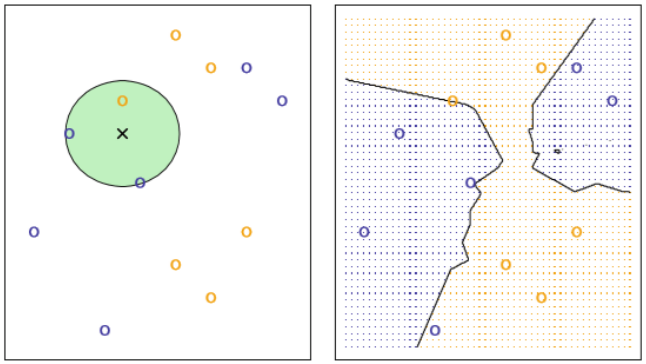


FIGURE 2.14. *The KNN approach, using $K = 3$, is illustrated in a simple situation with six blue observations and six orange observations.* Left: *a test observation at which a predicted class label is desired is shown as a black cross. The three closest points to the test observation are identified, and it is predicted that the test observation belongs to the most commonly-occurring class, in this case blue.* Right: *The KNN decision boundary for this example is shown in black. The blue grid indicates the region in which a test observation will be assigned to the blue class, and the orange grid indicates the region in which it will be assigned to the orange class.*

- The user sets the hyper-parameter K to the desired number of nearest neighbors This hyper-parameter K determines flexibility of kNN algorithm:

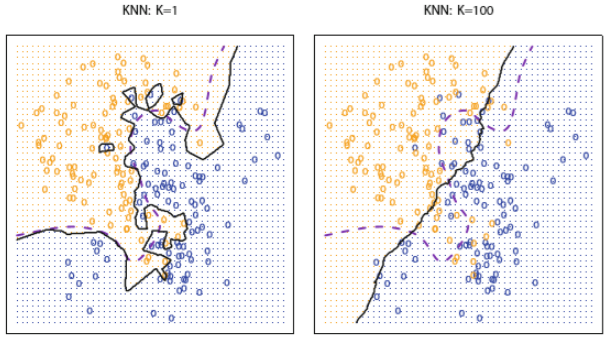


FIGURE 2.16. *A comparison of the KNN decision boundaries (solid black curves) obtained using $K = 1$ and $K = 100$ on the data from Figure 2.13. With $K = 1$, the decision boundary is overly flexible, while with $K = 100$ it is not sufficiently flexible. The Bayes decision boundary is shown as a purple dashed line.*

- As well as the test and training error rates:

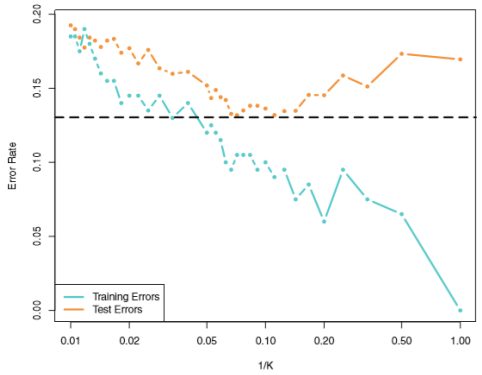


FIGURE 2.17. *The KNN training error rate (blue, 200 observations) and test error rate (orange, 5,000 observations) on the data from Figure 2.13, as the level of flexibility (assessed using $1/K$) increases, or equivalently as the number of neighbors K decreases. The black dashed line indicates the Bayes error rate. The jumpiness of the curves is due to the small size of the training data set.*

- The function **`class::knn( )`** performs kNN model calibrations (see script **`kNN.R`**).
- Advantages of the kNN model:
  - Simple and effective estimation which gives by default the probability of the predicted group membership.
  - In the two-group scenario other decision rules than the maximum predicted group probability can be used.
  - Makes no assumptions about the underlying distribution of the features.
  - Virtually no training time required.
- Disadvantages of the kNN model:
  - As the number of features to classify the data increases the decision space becomes more sparsely populated which increases the overall uncertainty.
  - Features that are not relevant for the classification task are not eliminated (ML concept of regularization) thus there is the potential of overfitting the data.
  - The performance of the algorithm depends on how ***comparable*** the features are scaled.
  - For each test data point $\mathbf{x}_0$ the distances need to be calculated in order to identify the k nearest neighborhood $\mathcal{N}_0$.
  - Euclidian distances for nominal scaled features and missing feature information require additional processing.
  - The algorithm's prediction performance depends on the setting of the hyper-parameter K.

Fundamental Setting of Unsupervised ML

- In unsupervised learning, only the object features $\mathbf{x}_i$ but not the outcome y_i are known to the investigator.
- A function $f(\quad)$ is sought that allows us to guess what y could be.

- An example is by grouping objects together into clusters. With respect to features $\mathbf{x}_i$ the clusters are supposed to be internally as homogenous as possible, but as heterogeneous (distinct) from each other.
- Again, fuzziness between the cluster delimitations determines the success of the classification rule.

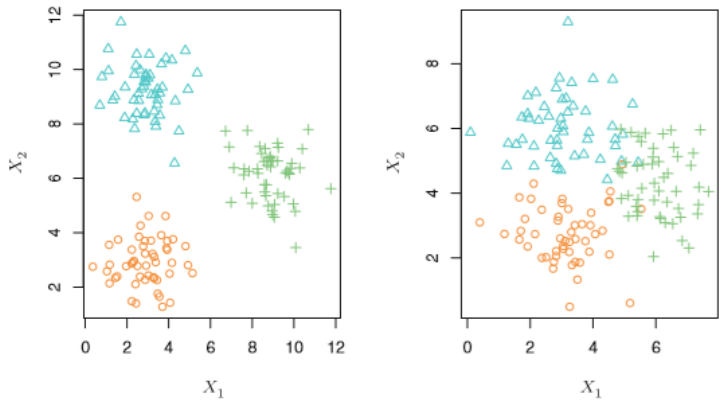


FIGURE 2.8. *A clustering data set involving three groups. Each group is shown using a different colored symbol.* Left: *The three groups are well-separated. In this setting, a clustering approach should successfully identify the three groups.* Right: *There is some overlap among the groups. Now the clustering task is more challenging.*

- Example: Identifying a multivariate mixture distribution
  - The distribution of points with multiple centers and ellipsoidal standard distances can be modeled by a mixture of multi-normal distributions:

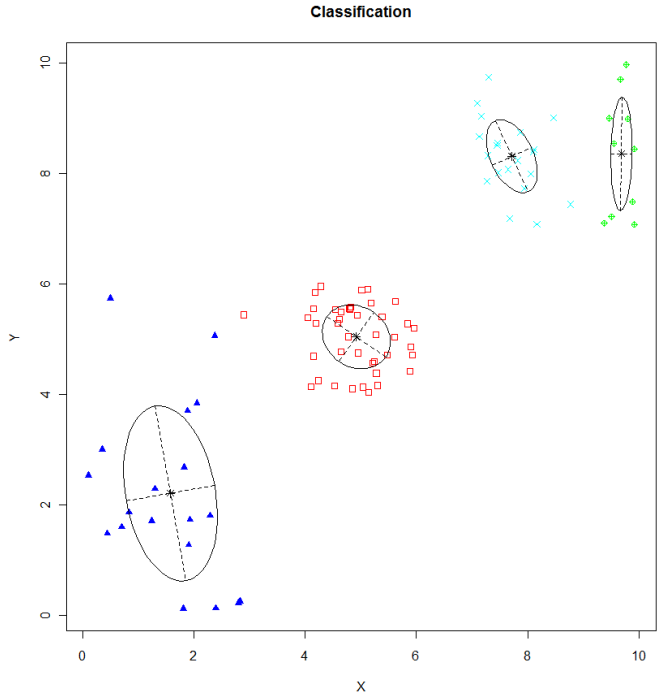
$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \sim \sum_{p=1}^{P} \pi_p \cdot \mathcal{N}\left(\begin{bmatrix} \mu_{x,p} \\ \mu_{y,p} \end{bmatrix}, \begin{bmatrix} \sigma_{x,p}^2 & \sigma_{xy,p} \\ \sigma_{xy,p} & \sigma_{y,p}^2 \end{bmatrix} \right) \text{with } \pi_p > 0 \text{ and } \sum_{p=1}^{P} \pi_p = 1$$

  - Unknowns are:
    [a] the number of clusters P,
    [b] the proportion of points per cluster π_p,

[c] the P cluster centers $\begin{bmatrix}\mu_{x,p}\\ \mu_{y,p}\end{bmatrix}$ and

[d] their P ellipsoidal covariance matrices $\begin{bmatrix}\sigma_{x,p}^2 & \sigma_{xy,p}\\ \sigma_{xy,p} & \sigma_{y,p}\end{bmatrix}$.

- See the sample script **`findClusters.r`**:




```
> p.Mclust$parameters

$pro
[1] 0.2144090 0.4522577 0.1109484 0.2223849

$mean
      [,1]     [,2]     [,3]     [,4]
X 1.573950 4.919038 9.688369 7.715275
Y 2.208468 5.042728 8.355194 8.312360

$variance$sigma
, , 1
           X          Y
X  0.7124123 -0.3320267
Y -0.3320267  2.5172177

, , 2
            X           Y
X  0.37565944 -0.05669909
Y -0.05669909  0.33411790

, , 3
           X          Y
X 0.03605865 0.01643843
Y 0.01643843 1.05226865

, , 4
           X          Y
X  0.2100294 -0.1332837
Y -0.1332837  0.4367538
```