Statistical Learning

Javier Nogales
www.est.uc3m.es/nogales
@fjnogales

MS in Statistics for Data Science

2019

uc3m | Universidad Carlos III de Madrid

Objectives

- Become familiar with different analytical tools, based on data, to make business decisions
- Develop skills for the main statistical and machine learning tools in supervised learning
- Use these models to make practical predictions/classifications and perform analytical inferences
- Handle the R language for these tools

Organization

- Subject organized in two main topics: statistical-based models and machine-learning tools (roughly 70% + 30%)

- 7 weeks

- Practical course: 50% basic concepts + 50% computer labs (using R)

- Evaluation: 50% midterm-project (5th week) + 50% final project (8th week)

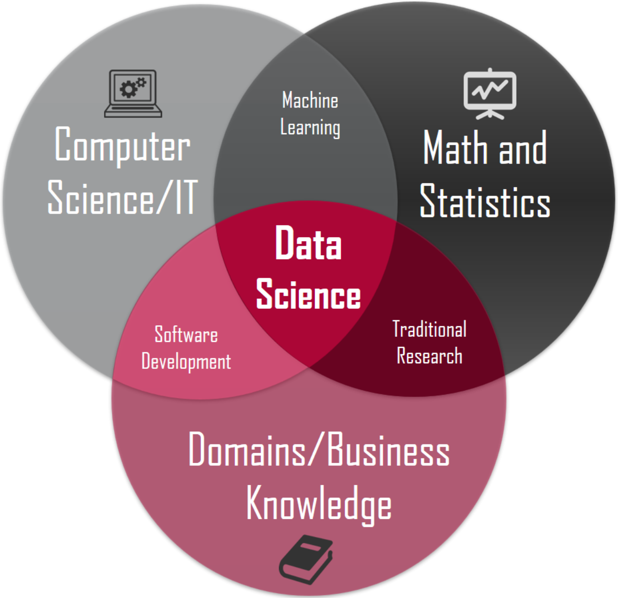
  Real-data project:

  - First part (deadline 5th week): prepare the input and statistical-learning tools (5 points)
  - Second part (final report upload on 8th week): machine-learning tools and prepare the output (5 points)
  - See more details in Aula Global

Statistical Learning

1. Introduction: data science and supervised learning
2. Probabilistic Learning
3. Machine Learning

uc3m | Universidad Carlos III de Madrid


Computer Science/IT
Machine Learning
Math and Statistics
Data Science
Software Development
Traditional Research
Domains/Business Knowledge

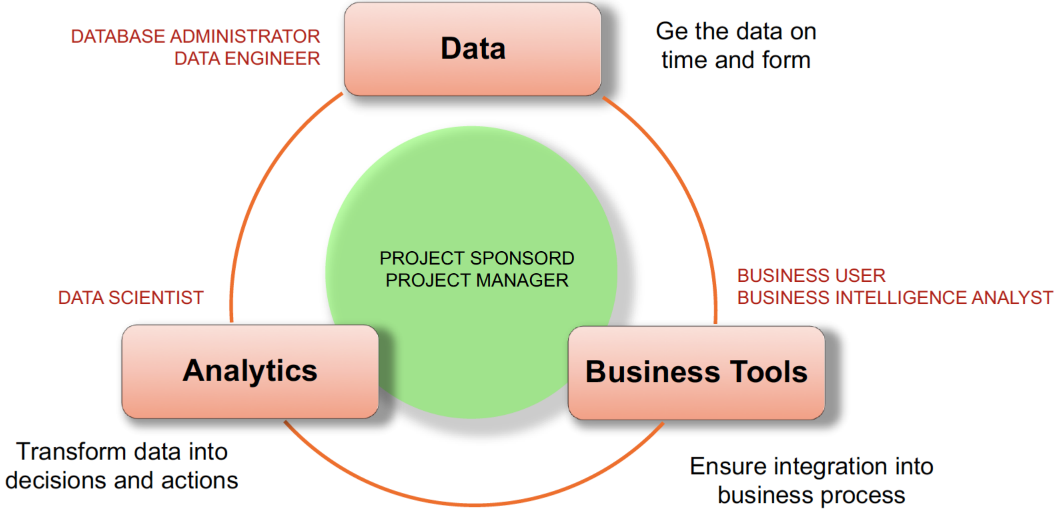
What is Data Science?

Obtain **valuable** information from data

Specifically: collection, processing and analysis of data to guide or make optimal decisions

Data science is a mathematical/statistical data-driven solution to a business problem

Different roles



DATABASE ADMINISTRATOR
DATA ENGINEER

Data

Ge the data on time and form

PROJECT SPONSORD
PROJECT MANAGER

DATA SCIENTIST

Analytics

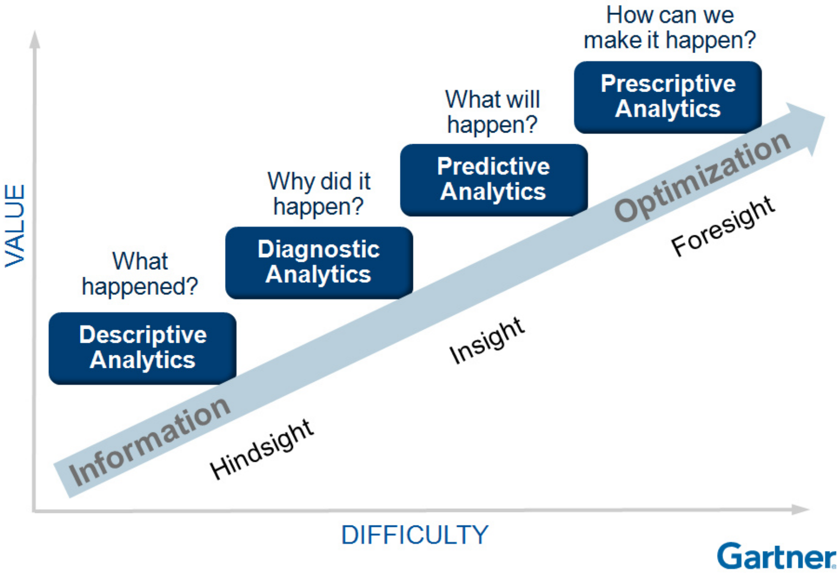
Transform data into decisions and actions

BUSINESS USER
BUSINESS INTELLIGENCE ANALYST

Business Tools

Ensure integration into business process

Analytic Value Escalator



What is a Data Scientist?

- Traditional analysts: degrees in business, economics, or similar. More interested in models from structured datasets, not too much emphasis in computing

- Data scientists: degrees in mathematics, computer science, statistics, or similar. More interested in complicated data (unstructured), with more emphasis in computing

- Era of Big Data: Analytics and Data Science are practically the same

What is a Data Scientist?

- Skills in Statistics and Machine Learning and Data Mining
- Also with Computer Science and Domain knowledge
- And also able to communicate and collaborate efficiently

What is a Data Scientist?

- Skills in Statistics and Machine Learning and Data Mining
- Also with Computer Science and Domain knowledge
- And also able to communicate and collaborate efficiently

Data Science: Steps

1. Prepare the input: collect, clean, transform, filter, aggregate, merge, verify, etc.
2. Prepare a model: build, estimate, validate, predict, etc.
3. Prepare the output: communicate the results, interpret, publish, etc.

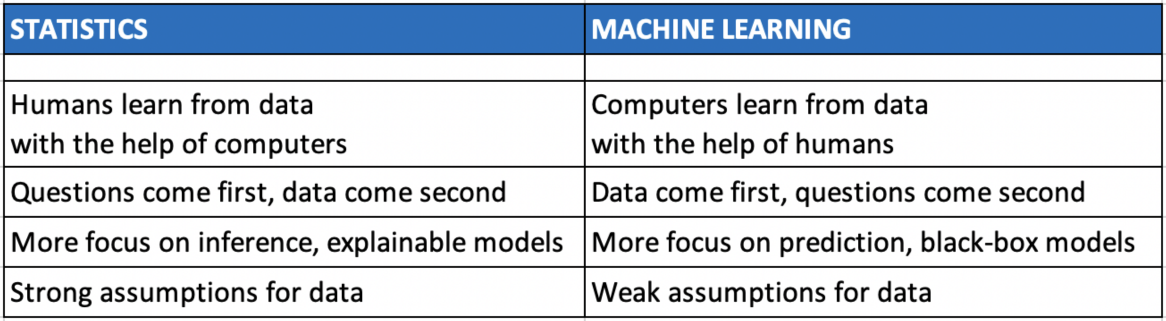
Big-data modelling focuses on second step: analytical tools

Data Science: Analytical Tools

Analytical tools come from Statistics, Machine Learning and Data Mining

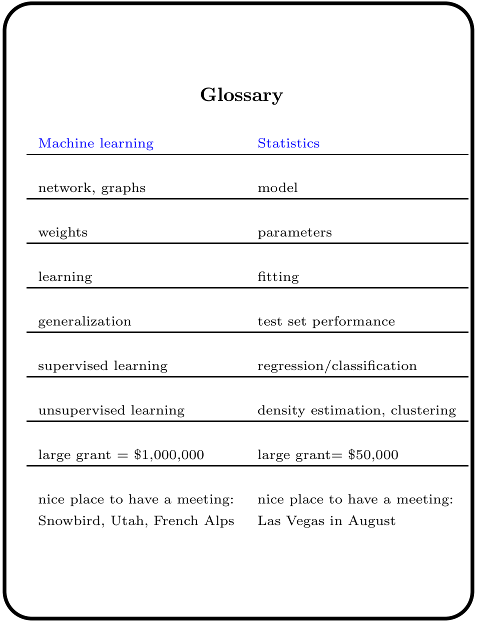
- Machine learning is somehow a subfield of Artificial Intelligence and Computer Science
  - More emphasis on large scale applications and prediction accuracy
- Statistics is somehow a subfield of Applied Mathematics
  - More emphasis on models and their interpretability, and the associated precision and uncertainty (explanation, inference)
- Data Mining is somehow a subfield of Computer Science
  - More emphasis on unstructured data and searching for unknown interesting patterns (unsupervised learning, pattern recognition)

Statistics vs Machine Learning



STATISTICS	MACHINE LEARNING
Humans learn from data with the help of computers	Computers learn from data with the help of humans
Questions come first, data come second	Data come first, questions come second
More focus on inference, explainable models	More focus on prediction, black-box models
Strong assumptions for data	Weak assumptions for data

Statistics vs Machine Learning (by R. Tibshirani)



Glossary

Machine learning	Statistics
network, graphs	model
weights	parameters
learning	fitting
generalization	test set performance
supervised learning	regression/classification
unsupervised learning	density estimation, clustering
large grant = $1,000,000	large grant= $50,000
nice place to have a meeting: Snowbird, Utah, French Alps	nice place to have a meeting: Las Vegas in August

Data Science: Programming Tools

- Programming tools preferred in statistics: R, Matlab
- Programming tools preferred in machine learning: Python, Matlab
- Other: SAS, Stata, SPSS, c++, Java, ...

Python vs R for Data Science

- Two most popular programming languages for data science: Python and R
- Both are free and open source
- R is more focused on data analysis
- Python is more general focused, hence a powerful and versatile language
- R advantages: data modeling, number of available packages, reports and visualization
- Python advantages: computing power, better integration in software engineering, specialized in deep learning, Keras and TensorFlow

Machine/Statistical Learning

- Supervised learning: predict or estimate an output (*response*) from various inputs (*predictors*)

  Statistical tools: better understanding about the relationship between the response and the predictors (inference)

  Machine learning tools: better (prediction accuracy)

  - Most widely used tools: regression, classification, text analytics, recommendation systems, time series,

- Unsupervised learning: tools for understanding data, with no target attribute (no labels). Usually organize into some natural groups

  Difficult to know how well your are doing; useful as a pre-processing step for supervised learning

  - Most widely used tools: PCA, clustering/segmentation, association rules

- Reinforcement learning: learn to make decisions based on a reward signal

  Hence, learn a set of actions or policies in order to maximize a expected reward

  - Related with optimization (dynamic programming under uncertainty)

- Others: semi-supervised learning, optimization, simulation, . . .

Supervised Learning

- Take inputs $x_1, x_2, \ldots, x_k$ and map them to an output y
  - Statistical learning uses probability assumptions to find this map or function
  - Machine learning does not use probabilities, just the data
- To know whether the map is working correctly, we need some kind of metric to measure performance
  - Loss function: count number of times the model is working wrong (classification), or measure how close the prediction is to y (regression)
- The objective is to minimize the loss

A good reference to start. . .

- G. James, D. Witten, T. Hastie and R. Tibshirani.
  An Introduction to Statistical Learning with Applications in R

Supervised Learning

Some notation:

- Output y: target, dependent variable, response, ...
- Input $x_1, x_2, \ldots, x_k$: predictors, features, regressors, covariates, independent variables, ...
- Data organized by rows: observations, samples, examples, instances, ...

One framework, but mainly two categories:

- Regression
- Classification

Supervised Learning Framework

- Usual framework in Machine Learning / Statistics:

$$\text{Data} = \text{Model} + \text{Noise}$$

- When we focus on one variable predicted by others: supervised learning

$$y = g(x_1, \dots, x_k) + \text{Noise}$$

  - Statistical (linear) approximation: $y = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p + \epsilon$
  - Machine learning approximation: $y = \text{map}(x_1, \dots, x_p) + \epsilon$

- Note the variables in the true model, g, may be different in the approximations

- Two main categories:
  - Classification: y is categorical
  - Regression: y is numeric (usually continuous)

Linear regression: a brief review

- The target, y, is a continuous variable that depends linearly on predictors

- Although the linear assumption may seem simplistic, it is extremely useful in practice

- Linear regression can deal with interactions and non-linearities

- Estimation is based on least squares: the loss function is the sum of residual squares (training prediction error)

- But what happens if the input comes from Big Data?

High-dimensional regression: an advance

- In high dimension, or p/n large, the dataset may contain redundant information about the response because some predictors may be correlated (collinearity)

- In other words: more information (more variables) is not necessary better, because this new information will come with much more noise in high dimension

Hence, how to improve linear models when the dimension is high?

- Statistical Learning: improve interpretability while attaining good predictive performance, replace least squares with some alternative fitting tools

- Machine Learning: focus on prediction accuracy, deals better with large scale applications

Today, much overlap between both views

To explain or to predict?

In predictive modeling, there are three sources of uncertainty:

- The error in the coefficients when the linear approximation is true (estimation error)
- The error in the linear approximation when the true model is non-linear, or contains other variables (model bias)
- The noise in the DGP: Data $=$ Model $+$ Noise (irreducible error)

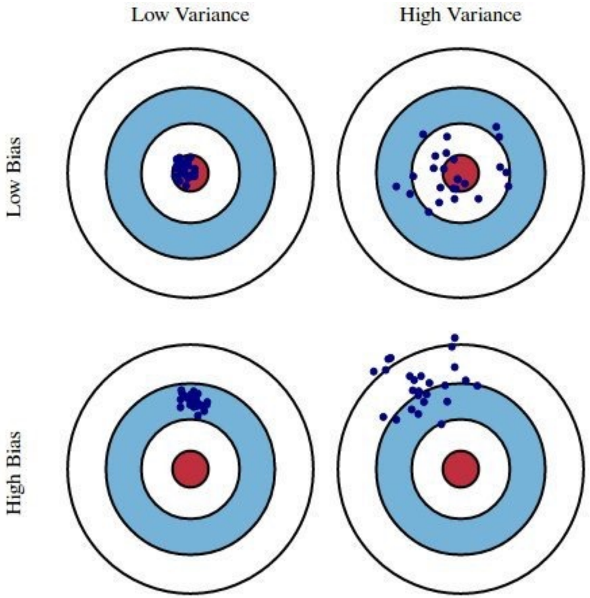
$$\boxed{(\text{Prediction Error})^2 = \sigma^2 + \text{Bias}^2 + \text{Var}}$$

To explain or to predict?

$$\boxed{(\text{Prediction Error})^2 = \sigma^2 + \text{Bias}^2 + \text{Var}}$$

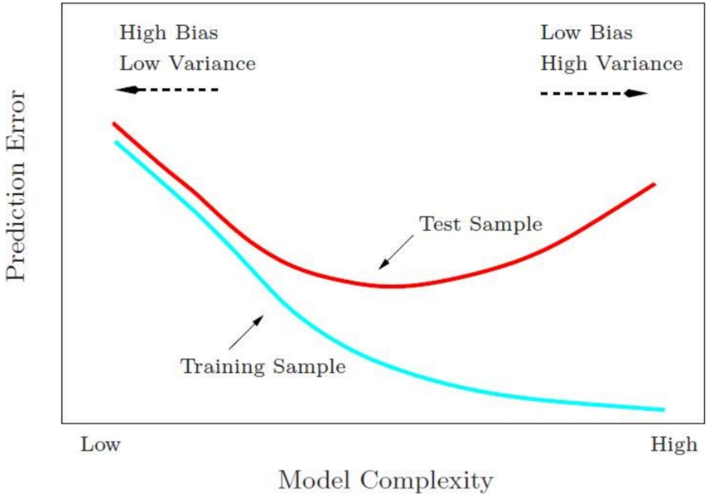
- Statistics:
  - Focus on minimizing Bias (by assuming knowledge about population, DGP)
  - Hence, able to obtain formulas for Var that provides explanation (inference, effects of predictors on response)
  - The Var can be large in practice
- Machine Learning:
  - Focus on minimizing $\text{Bias}^2 + \text{Var}$
  - No assumptions needed (discover knowledge), hence no formulas and no explanation
  - But good prediction performance

Predictions: Bias vs Variance



Overfitting

When a model fits or predict very well the training data but bad the testing data (p is large compared with n)



Supervised Learning: Classification

- Traditional machine-learning classification: process to predict a categorical/qualitative variable (*response/target*) from various inputs (*predictors*)

  Example: predict whether tomorrow will rain or not

  Focus is on prediction, hence better prediction performance

- Probabilistic learning: the same but first predict the probability of each category, then predict the category

  Example: tomorrow there will be 70-percent chance of rain

  Focus is on explanation + inference, hence better understanding (about relationship between the response and the predictors)

Classification

- We have a data matrix $X \in \mathbb{R}^{n \times p}$ for the predictors or features. The observations in X come from different known sub-populations (categories)

- We know a sample of well-classified elements that will allow to classify new observations

- We will use *labels* y to identify the categories

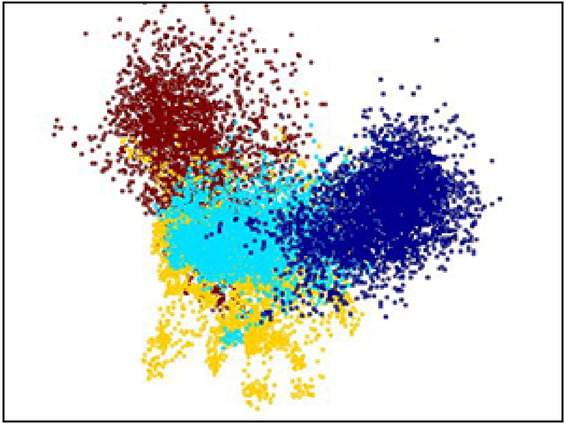
- Main question: if we have a new observation, can we predict to which category it will belong?

Classification

- Many applications:
  - Credit scoring, credit risk
  - Stock trading
  - Electronic fraud detection
  - Spam filtering or text classification
  - Diagnosing medical conditions
  - DNA sequence data to detect disease causing
  - Pattern recognition: voice recognition, text classification, image recognition

Classification

- Four groups, two variables, thousands of observations



- For a new observation, can we predict to which group it will belong?

Classification

- Statistical Classification:
  - Logistic regression
  - Bayes classifiers
    - LDA
    - QDA
    - Naïve Bayes
    - Shrinkage classification
- Machine-Learning Classification:
  - Nearest Neighbors
  - Neural Networks
  - SVMs
  - Decision trees, Random Forests, Gradient Boosting, . . .

Statistical Learning

1. Introduction: data science and supervised learning
2. Probabilistic Learning
3. Machine Learning

Probabilistic Learning

Mainly two families:

- Bayes Classifiers
- Logistic Regression

Bayesian Learning

Classification: Bayes Classifiers

- Probabilistic learning: instead of focusing on the conditional mean of of the response given the predictors, $y|x_1, \ldots, x_p$, focus on the conditional probability: $p(y|x_1, \ldots, x_p)$
- In logistic regression, this conditional probability is modelled directly
- In Bayes classifiers, the conditional probability is modelled in a different and less direct way:
    - First, we model the predictors X separately for each given class y: $p(x|y)$
    - Then, we apply the Bayes formula to get $p(y|x)$
- This idea is more stable than logistic regression when the classes are well-separated
- It is popular when we have more than two classes
- Bayes classifiers perform well if variables are somehow Gaussian
  Note this implies the predictors should be numeric, but not mandatory

A review: Bayes' Theorem

- $P(A|B)$ probability of A given B is true/known, also called posterior probability
- In practice, A is an unknown event and B is known data/information
- $P(A)$ is the prior probability (what we know about A independently of B) before collecting data
- $P(B)$ is called the marginal distribution (independent of all events)
- $P(B|A)$ is the conditional probability of B knowing A, also known as the likelihood (of the data given A is true)
- Bayes' Theorem:

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

- In many cases, we have several mutually disjoint events: $A_1, \ldots, A_K$. In that case,

$$P(B) = \sum_k P(B \cap A_k) = \sum_k P(B|A_k)P(A_k)$$

A review: Bayes' Theorem

Example: drug test

- 0.4% of a given population use some type of drug

- Conventional drug test: produce 99% true positives for drug users and 99% true negatives for non-drug users

- If one individual is tested positive, what is the probability of having used the drug?

- Bayes' rule:

$$P(\text{user}|+) = \frac{P(+|\text{user})P(\text{user})}{P(+|\text{user})P(\text{user}) + P(+|!\text{user})P(!\text{user})} = \frac{0.99 \times 0.004}{0.99 \times 0.004 + 0.01 \times 0.996} = 28.4\%$$

- Very small... why?

Bayes Classifiers

- Imagine we have G groups, and the output can take G distinct values, coded as $g = 1, \ldots, G$
- Let $\pi_g = P(y \in g)$ denote the prior probability (known), for $g = 1, \ldots, G$, that an observation belongs to group g
- Note previous probabilities are independent of predictors X
- First, model the (multivariate) distribution of predictors for each g: $f_g(x)$

Bayes Classifiers

- Then, apply Bayes Theorem to obtain the posterior probabilities given a new observation x:

$$p_g(x) = P(y \in g \mid X = x) = \frac{f_g(x)\pi_g}{\sum_k f_k(x)\pi_k}$$

- Finally, apply Bayes' rule: assign new observation x to that group with largest $p_g(x)$, i.e. $\max_g \ f_g(x)\pi_g$

- Note that $p_1(x) + \cdots + p_G(x) = 1$

- The posterior probabilities are the same as in logistic regression but estimated in a different way

Bayes Classifiers

- Bayes classifiers are optimal in the sense they minimize the classification error rate

- This is just in theory: we need to know exactly, without any error, $f_g(x)$ and π_g for each g

- In practice, we just approximate or estimate $f_g(x)$ and π_g

- For instance, π_g can be estimated using the proportion of training observations that belong to class g: $\hat{\pi}_g = n_g/n$

- But without previous knowledge, $\hat{\pi}_g = 1/G$ is a good choice

- Next, we will see some useful approximations for $f_g(x)$ in practice

Bayes Classifiers

- If each $f_g(x)$ is assumed to be multivariate normal, $f_g \sim \mathcal{N}_p(\mu_g, \Sigma_g)$, then

- Bayes rule: $\max_g \; f_g(x)\pi_g \equiv$

$$\max_g \quad \pi_g \; (2\pi)^{-p/2} \; \det(\Sigma_g)^{-1/2} \; \exp(-0.5(x-\mu_g)^T \Sigma_g^{-1}(x-\mu_g))$$

  which is equivalent (taking logs) to:

$$\min_g \quad (x-\mu_g)^T \Sigma_g^{-1}(x-\mu_g) + \log\det(\Sigma_g) - 2\log\pi_g$$

- This is called Quadratic Discriminant Analysis (QDA)

- We can estimate parameters using the training set, but may be too complex in high dimension (p/n large): we need to estimate G vector of means and G covariance matrices

- In practice, it is an unbiased estimator in high variance

Classification: QDA

- Assuming the data matrix X is grouped in G submatrices, each with n_g observations, parameters are estimated as follows: $\hat{\mu}_g = \bar{x}_g$, $\hat{\Sigma}_g = S_g$, and $\hat{\pi}_g = \frac{n_g}{n}$

- If we do not have prior knowledge about π_g, then assume $\pi_1 = \cdots = \pi_G$, and focus on

$$\min_{g} \quad (x - \bar{x}_g)^T S_g^{-1} (x - \bar{x}_g) + \log\det(S_g)$$

- Note QDA may be too complex in high dimension (p large): from n observations, we need to estimate G vector of means and G covariance matrices

- Moreover, some S_g could be singular (or close to)

Classification: QDA

- For these reasons, QDA is relatively unstable in practice (except for very large samples and low dimension)

- Moreover, it is very sensitive to deviations from Gaussian distribution

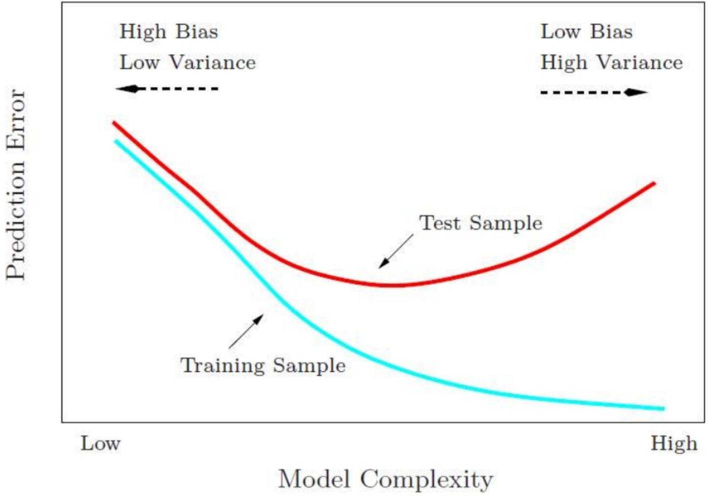
- Hence, convenient to impose restrictions in higher dimension

- Linear discriminant analysis (LDA): simplified version, much less parameters

  less variance but higher bias than QDA

Classification: simple models

When we simplify: we introduce some bias to reduce variance (less prediction error)



Classification: LDA

- Idea: introduce some bias to reduce variance (less prediction error)

- The most common restriction is to assume all covariance matrices, Σ_g, are equal. Then, the only covariance matrix to be estimated is $\hat{\Sigma} = S_w = \sum_{g=1}^{G} (\frac{n_g - 1}{n - G}) S_g$

- Bayes rule: $\max_g \;\; f_g(x)\pi_g \equiv$

$$\min_g \quad (x - \bar{x}_g)^T S_w^{-1} (x - \bar{x}_g) - 2 \log \hat{\pi}_g$$

- This is indeed a linear rule (in x), equivalent to

$$\min_g \quad 2\bar{x}_g^T S_w^{-1} x - \bar{x}_g^T S_w^{-1} \bar{x}_g - 2 \log \hat{\pi}_g$$

- This is called Linear Discriminant Analysis (LDA)

Classification: LDA

- It is linear because the terms $x^T S_w^{-1} x$ are constant for all groups

- Similar to the linear rule in logistic regression:
  - If we have 2 groups (0 and 1), $\text{logit}(p) = \log \frac{p}{1-p} = \beta_0 + \beta^T x$
  - In LDA, $\text{logit}(p) = (\mu_1 - \mu_0)^T \Sigma^{-1}\ x + \log \frac{\pi_0}{\pi_1} = \alpha_0 + \alpha^T x$
  - Both approaches give linear logit, but they estimate parameters in different ways

Classification: LDA

- If we do not have prior knowledge about π_g, then assume $\pi_1 = \cdots = \pi_G$, and focus on

$$\min_g \quad (x - \bar{x}_g)^T S_w^{-1} (x - \bar{x}_g)$$

- I.e., classify x in the population whose mean is closest (using Mahalanobis distance)

LDA

Some recommendations:

- LDA can be sensitive to outliers: hence, try to identify and remove them previously
- LDA assumes all groups have the same covariance matrix: hence, standardize the variables previously
- LDA assumes predictor distribution is multivariate gaussian: hence, transform the predictors previously to make them somehow symmetric
- For binary classification, better logistic regression (unless very well-separated groups)
- For multi-class classification, better LDA (unless very-unbalanced groups)

Classification: Example

- Fisher's Iris data set: best known data set in classification

- Data collected by Edgar Anderson in the 30's to quantify the morphologic variation of Iris flowers of three related species

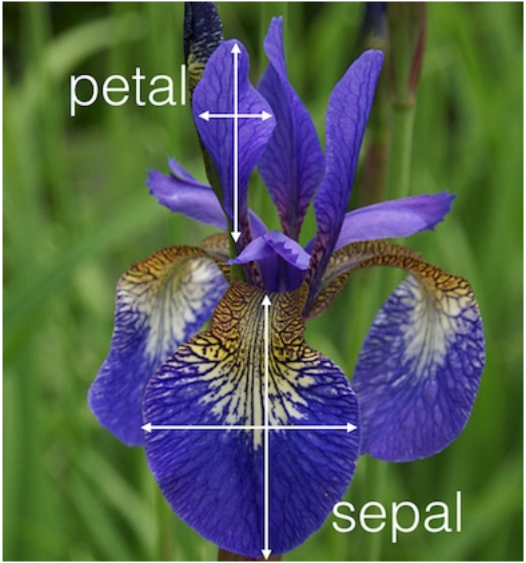
- It consists of 150 Iris flowers of three different species: setosa, versicolor, virginica

- Four types of measurements for each flower, the length and width of sepals and petals in centimeters, respectively

- Popularized by Ronald Fisher in 1936

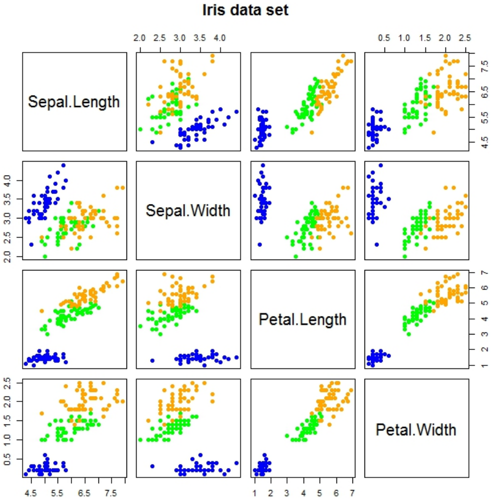
- Based on the combination of the four features, Fisher developed a linear discriminant model to distinguish the species from each other

Iris Flower



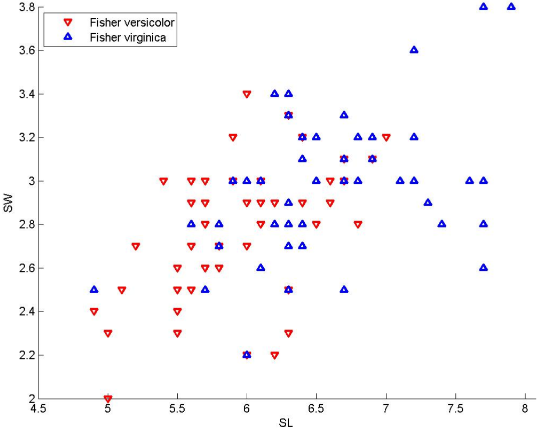
Iris Scatter Plot

In blue, setosa Iris; in green, versicolor Iris; in orange, virginica Iris



Classification: Example

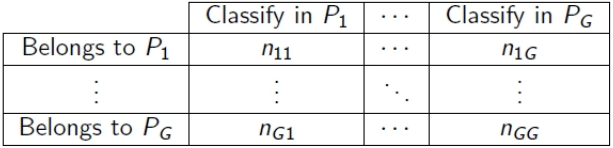
- Iris dataset: focus first on two species (versicolor and virginica)



- And consider just two features (length and width of sepals)

Classification: performance measures

- Confusion matrix:



	Classify in P_1	$\cdots$	Classify in P_G
Belongs to P_1	n_{11}	$\cdots$	n_{1G}
$\vdots$	$\vdots$	$\ddots$	$\vdots$
Belongs to P_G	n_{G1}	$\cdots$	n_{GG}

where n_{ij} is the amount of observations coming from group P_i and classified in group P_j

- The error is then: $\frac{n_{12}+\cdots+n_{G,G-1}}{n}$, the sum of off-diagonal elements

- The accuracy is the sum of the diagonal terms

- The confusion matrix should be computed using cross-validation or testing sets (not training ones)

Performance measures

- The in-sample or training error may be too optimistic, due to overfitting

- We need more realistic errors:

- Use cross-validation: exclude from the sample one observation, estimate parameters to build the classification rule, classify the excluded observation, and finally check the error. Hence, n classifications are performed

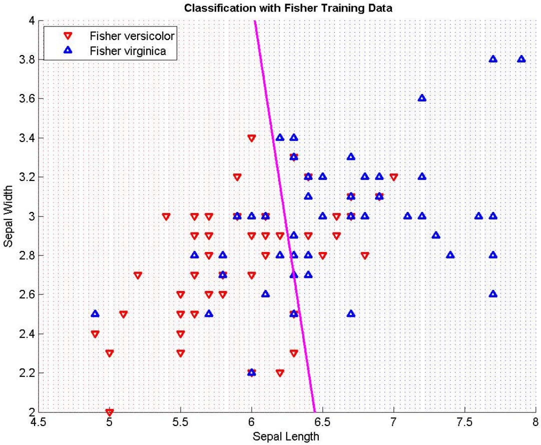
- Use a training and a validation set: let's say, 80% of the data to estimate (train or learn) the model and the other 20% to validate. Repeat many times

- These are known as out-of-sample or testing errors

- Finally, note from confusion matrix other performance measures can be obtained: Accuracy, Kappa, Sensitivity (TPs), Specificity (TNs), etc.

Example

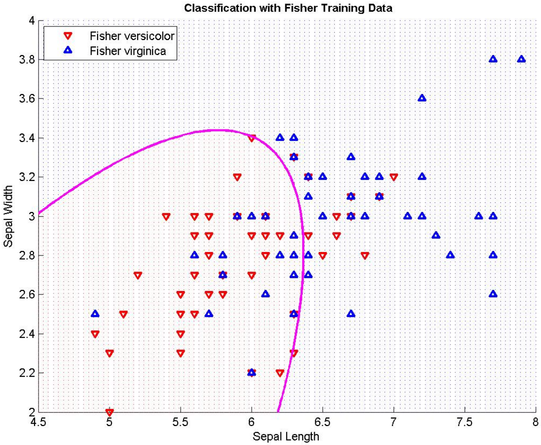
- LDA classifier:



- In-sample error $= 25\%$

Example

- QDA classifier:



- In-sample error $= 29\%$

Example

- Iris dataset: consider now the three species and the four features
- In-sample error for LDA: 2%
- CV error for LDA: 2%
- Out-of-sample error for LDA: 2.3%

- In-sample error for QDA: 2.7%
- CV error for QDA: 2.7%
- Out-of-sample error for QDA: 2.7%

Naïve Bayes

- Another restriction, successful in high dimension, is to assume variables (features) are independent
- This is called Naïve Bayes Classification
- It assumes: $f_g(x) = f_g(x_1) \times f_g(x_2) \times \cdots \times f_g(x_p)$
- Hence, there are many versions depending on distribution f_g:
  Gaussian, kernel, multinomial, bernoulli, etc.
- It is easy to build and particularly useful for very large data sets
- Widely used in text classification (high dimensional training data sets): spam filters, sentimental analysis, news articles classification, etc.

Naïve Bayes

- Gaussian Naïve Bayes: useful for continuous predictors

  - Linear naïve Bayes:
    Impose the pooled covariance matrix is diagonal:

    $$\hat{\Sigma} = \mathrm{diag}(S_w)$$

  - Quadratic naïve Bayes:
    Impose each covariance matrix is diagonal:

    $$\hat{\Sigma}_g = \mathrm{diag}(S_g), \quad g = 1, \ldots, G$$

Naïve Bayes

- Multinomial Naïve Bayes: useful for discrete counts
  - number of times outcome number x_i is observed over the n trials
  - used in text classification word count vectors
- Bernoulli Naïve Bayes: useful for binary predictors
  - used in text classification word occurrence vectors

Naïve Bayes

General comments:

- Easy and fast to predict and understand
- Perform well in multi-class prediction, especially when we have a large number of categorical predictors
- But depends heavily on independence of predictors
- Posterior probabilities not reliable
- Mainly used in: text classification (spam filters), sentiment analysis (is text negative, positive or neutral?), and recommendation systems (would a user like a given product?)

Shrinkage Methods in Classification

- If the dimension is large, then better to use LDA than QDA

- And better to use Naïve Bayes than LDA

- Shrinkage classification is a way to combine above ideas:
  - $\hat{\Sigma}_g(\lambda) = (1-\lambda)S_g + \lambda S_w$, where $0 \leq \lambda \leq 1$ (combine LDA and QDA)
  - $\hat{\Sigma}_g(\alpha, \lambda) = (1-\alpha)\hat{\Sigma}_g(\lambda) + \alpha\nu_\lambda I$, where $0 \leq \alpha \leq 1$ and $\nu_\lambda = \mathrm{tr}(\hat{\Sigma}_g(\lambda))/p$ (and now combine with naive)

- Parameters α and λ need to be optimized to improve performance

Bayes classifiers: advanced tools and packages

- Factor-Based Linear Discriminant Analysis: package: HiDimDA
- Linear Discriminant Analysis with Stepwise Feature Selection: packages: klaR, MASS
- Penalized Discriminant Analysis: package: mda, penalizedLDA, klaR, rlda, sda, sparsediscrim
- Sparse Linear Discriminant Analysis: packages: sparseLDA, sparsediscrim
- Naive Bayes: packages: naivebayes, klaR,

Logistic Regression

Logistic Regression: a review

- Extension of classical multiple regression where the response was a continuous variable

- Now the response is a qualitative variable, usually binary, or the associated proportions

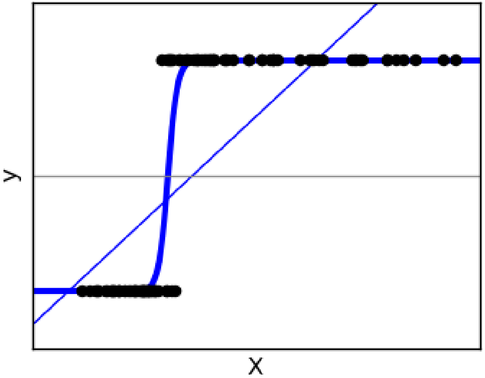
- Hence, logistic regression can be viewed as a particular case of
  - GLM (logit link)
  - Classification (linear classifier)

- The predictors can be either continuous or categorical

Logistic Regression

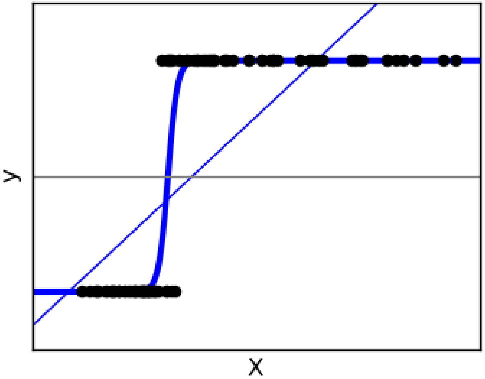
- Imagine we have just two groups: binary regression.
  Then, the possible y labels are 0 or 1 (first and second group, respectively).

- Instead of modeling y as a continuous variable and then $E(y|x_1, \ldots, x_p) = \beta_0 + \beta^T x$



Logistic Regression

- Better to consider y is binary and then model
$$E(y|x_1,\ldots,x_p) = p = P(y=1|X=x) = F(\beta_0 + \beta^T x)$$
where F is a continuous function between 0 and 1



Logistic Regression

- The quantity $\frac{p}{1-p}$ is called the odds

- A small value indicates very low probability that $y = 1$, whereas a large value indicates very high probability

- Usually, we impose that the log-odds or $\text{logit}(p)$ is linear:

$$\text{logit}(p) = \log \frac{p}{1-p} = \beta_0 + \beta^T x$$

- In this case, a negative value indicates most likely $y = 0$, whereas a positive value indicates most likely $y = 1$ (more symmetry)

- If we increase an x-variable by one unit, the logit is increased by the corresponding β

Logistic Regression

- Hence, $p = F(\beta_0 + \beta^T x) = \frac{\exp(\beta_0+\beta^T x)}{1+\exp(\beta_0+\beta^T x)} = \frac{1}{1+\exp(-\beta_0-\beta^T x)}$
- Note $P(y = 0|X = x) = 1 - p = \frac{1}{1+\exp(\beta_0+\beta^T x)}$
- Need optimization to estimate the β's (based on the MLE of y given x)
- Non-linear LS could be used, but MLE has better statistical properties

Logistic Regression

- MLE: max $\prod_{i=1}^{n} p_i(x_i|\beta)^{y_i}(1 - p_i(x_i|\beta))^{1-y_i}$

- The multivariate distribution of X is not so important: explanatory variables can be non-Gaussian, contain categorical variables, etc.

- What is important is that the logit be linear

- Taking logs: max $\sum_{i=1}^{n}(\beta_0 + \beta^T x_i)y_i - \sum_{i=1}^{n} \log(1 + \exp(\beta_0 + \beta^T x_i))$

- Use a convex optimization solver, problem is concave and differentiable

- After the estimation, for a given observation x, we can predict the associated probability:

$$\hat{p}(x) = P(y = 1) = \frac{\exp(\hat{\beta}_0 + \hat{\beta}^T x)}{1 + \exp(\hat{\beta}_0 + \hat{\beta}^T x)} = \frac{1}{1 + \exp(-\hat{\beta}_0 - \hat{\beta}^T x)}$$

Logistic Regression

- Note logistic regression is also a linear classifier: the decision boundary separating the two predicted classes is the solution of $\hat{\beta}_0 + \hat{\beta}^T x = 0$

- That means, we are minimizing the mis-classification rate: we say $y = 1$ when $p \geq 0.5$ and $y = 0$ otherwise. I.e. we say $y = 1$ when $\hat{\beta}_0 + \hat{\beta}^T x \geq 0$

- Moreover, the distance from a observation to the decision boundary is

$$\frac{\hat{\beta}_0 + \hat{\beta}^T x}{||\hat{\beta}||}$$

- Hence, class probabilities go towards the extremes (0 and 1) more rapidly when $\hat{\beta}$ is larger

- If mis-classification errors are asymmetric, instead of classifying $y = 1$ with rule $\hat{p} > 0.5$, change the 0.5 to be more conservative or risky

Logistic Regression

- General case: G groups

- Hence, labels $y = \{0, 1, \ldots, G-1\}$

- First group is the control group, $y = 0$

- For the first group:

$$p_0 = P(y = 0|X = x) = \frac{1}{1 + \exp(\beta_{0,1} + \beta_1^T x) + \cdots + \exp(\beta_{0,G-1} + \beta_{G-1}^T x)}$$

Logistic Regression

- For the rest of the groups:

$$p_g = P(y = g|X = x) =$$

$$\frac{\exp(\beta_{0,g} + \beta_g^T x)}{1 + \exp(\beta_{0,1} + \beta_1^T x) + \cdots + \exp(\beta_{0,G-1} + \beta_{G-1}^T x)}, \quad g = 1, \ldots, G-1$$

- We need to impose $p_0 + \ldots + p_{G-1} = 1$

- Estimation is performed through optimization of the MLE

- Note in this case there are $G-1$ linear classifiers

Logistic Regression

- Take care: if we include explanatory variables that are not significant, then we may introduce a large bias in the estimation

- Hence, make a good variable selection before estimation, or estimate through shrinkage or regularization: add a penalty term of the form $\rho||\beta||$ in MLE

- The performance of the logistic regression model for $G \geq 3$ is not so good. Better performance attained by other classification tools. But for $G = 2$ the performance is good, as it allows for general predictors

Bayes Rule and Cost-Sensitive Learning

Probabilistic Learning: Bayes rule

- Bayes rule is optimal in the sense it minimizes the overall error rate, i.e. the number of elements off the diagonal in the confusion matrix (if the underlying assumptions are true, or in the training set)
- Hence, we are assuming all the classification errors are equally important
  But what happens if we are not interested in minimizing such overall error?
- Unbalanced classes and/or cost-sensitive learning

Probabilistic Learning: unbalanced classes

How to deal with unbalanced classes?

- Adjust prior probabilities, π_g: instead of using the proportions found in the training set, be more conservative (shrink towards 0.5)
- Change threshold in Bayes rule (0.5 in binary problems) for posterior probabilities
- Use other performance measures (rather than accuracy and error rate)
- Cost-sensitive learning
- Sub-sampling (for instance sampling more from minority class)

Probabilistic Learning: unbalanced classes

Performance measures, especially useful when 2 classes are unbalanced:

	Predicted	
Actual	yes	no
yes	TP	FN
no	FP	TN

- kappa: adjusts accuracy by the possibility of a correct prediction obtained by chance. The greater the better
- sensitivity: true positive rate, TP/(TP+FN). The greater the better
- specificity: true negative rate, TN/(TN+FP). The greater the better
- precision: positive predictive value, TP/(TP+FP). The greater the better
- recall: TP/(TP+FN) = sensitivity, but used with different interpretation. The greater the better
- F1 = 2*precision*recall/(precision+recall). The greater the better. Good measure to compare models

Probabilistic Learning: cost-sensitive learning

- Cost matrix:

	Predicted	
Actual	yes	no
yes	$-c_{11}$	c_{12}
no	c_{21}	$-c_{22}$

- Bayes rule is optimal when $c_{11} = c_{22} = 0$ and $c_{21} = c_{12}$

- In many applications, the losses from classification errors are unbalanced: $c_{11} \neq c_{22}$ and $c_{21} \neq c_{12}$

- The ROC curve (Receiver Operating Characteristic) can also help: tradeoff between true positives and false positives. The closer the curve to the top-left position, the better

- Cost-sensitive learning implies domain knowledge for the specific application, as we can reduce one specific error rate but at the cost of increasing the others

- Only a few packages in Caret, mainly for some SVMs (e1071) and trees (C5.0 and CART)

Probabilistic Learning: final remarks

- Logistic regression performs well for binary classification, and it allows categorical predictors (features). Recommendable to regularize to avoid overfitting
- For more than two groups, LDA performs better if predictors are somehow Gaussian
- In Bayes classifiers, feature selection should be done (to choose the important features) or shrinkage
- Validate conveniently the classification tools (using testing sets) and consider always misclassification losses
- Try as many classifications tools as you can: SVMs, RFss, kNN, etc.
- SVM is indeed similar to logistic regression with shrinkage

Statistical Learning

1. Introduction: data science and supervised learning

2. Probabilistic Learning

3. Machine Learning

uc3m | Universidad Carlos III de Madrid

Machine Learning

- Nearest Neighbors
- SVMs
- Decision trees, Random Forests, Gradient Boosting
- Neural Networks

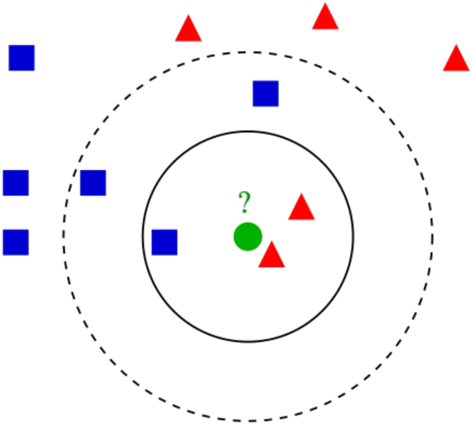
k-Nearest Neighbors

k-Nearest Neighbors

- Simple method with good performance for highly non-linear data and moderate p/n

- The k-NN algorithm is as follows:
  - Define a distance between observations (Euclidean, Mahalanobis, Manhattan, ...)
  - Compute the distance between the new observation to classify, x, and all the observations in the sample
  - Select the k closest observations to x and compute the proportion of them that belongs to each group
  - Then, classify x with the largest proportion (mode)

k-NN for classification



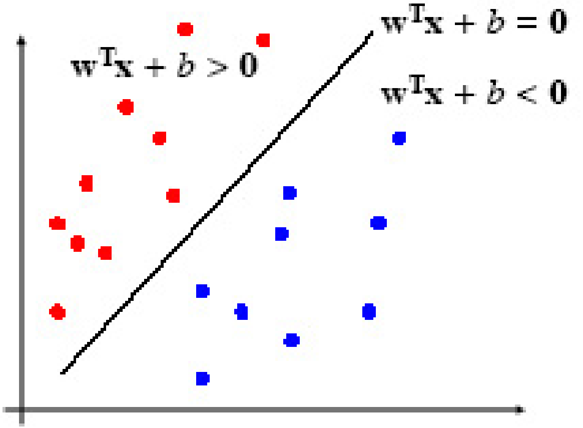
Classification: k-NN

- This method is a data-mining tool: no assumptions needed
- But it cannot tell us which predictors are important
- Well-suited for highly non-linear data
- But bad performance if the dimension, p, is high
- The key element in k-NN is the selection of k (neighbors)
- Usually, cross-validation is used to select k with less prediction error (bias-variance tradeoff)
- Preferable to try odd numbers for k

Support Vector Machines

Support Vector Machines

- We start again with a linear classifier, but changing notation (standard in ML)


$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b > 0$
$\mathbf{w}^T\mathbf{x} + b < 0$

Support Vector Machines

- But what is the best classifier? Now, not assumptions on data . . .

- Idea: try to maximize the gap between the two classes

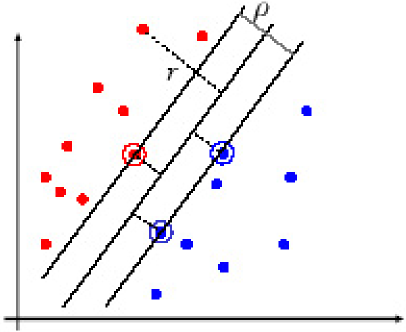
- Distance from a given point x to the linear classifier:

$$r = \frac{w^T x + b}{\|w\|}$$

- The support vectors are those points closest to the classifier

Support Vector Machines

- The margin ρ associated to the classifier is the width between the two classes



- Objective: maximize the margin

Support Vector Machines

- We assume (not relevant) that all the points are at least one unit distant from the classifier:

$$\begin{aligned} w^T x_i + b &\geq 1 \quad \text{if } y_i = 1 \\ w^T x_i + b &\leq -1 \quad \text{if } y_i = -1 \end{aligned}$$

- For the support vectors, the inequalities become equalities, hence,

$$\rho = \frac{2}{\|w\|}$$

- Moreover, maximize $\frac{2}{\|w\|}$ is equivalent to minimize $\|w\|^2$

Support Vector Machines

- So, we obtain the following optimization framework:

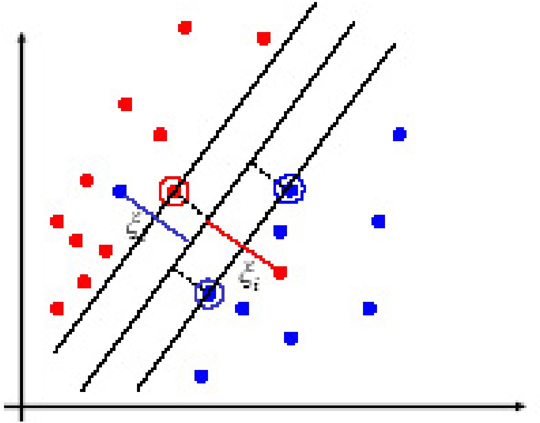
$$\begin{aligned} \text{minimize}_{w,b} \quad & \frac{1}{2} w^T w \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 \quad i = 1, \ldots, n \end{aligned}$$

- But what happen if the sample is not totally separable by a linear classifier?

- That is, it is almost separable, but with some noise, or with classification errors, etc.

Support Vector Machines

- In this case, we can add slack variables ζ_i to allow for classification errors



Support Vector Machines

- Optimization framework: (C controls for the overfitting)

$$\begin{aligned} \text{minimize}_{w,b} \quad & \frac{1}{2} w^T w + C \sum_i \zeta_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - \zeta_i \quad i = 1, \ldots n \\ & \zeta \geq 0 \end{aligned}$$

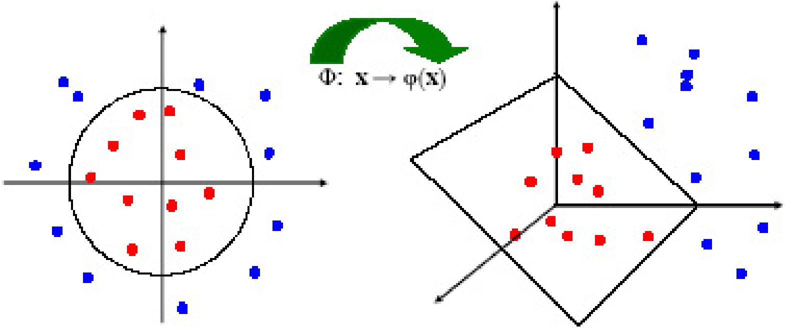
- In practice, the following dual problem is solved:

$$\begin{aligned} \text{maximize}_{\alpha} \quad & \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j (x_i^T x_j) \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0, \quad i = 1, \ldots n, \\ & 0 \leq \alpha_i \leq C \quad i = 1, \ldots n \end{aligned}$$

Support Vector Machines

- But what happen if there is more noise or the classification is nonlinear?

  In this case, the original sample is transformed into a higher space (using kernels). In this space, it is possible to find a linear classifier



For instance, the boundary $w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2 + w_5 x_1 x_2 + b = 0$ is non-linear in (x_1, x_2) but linear in $(x_1, x_2, x_1^2, x_2^2, x_1 x_2)$

Support Vector Machines

- Mercer's Theorem: every symmetric and semidefinite positive function, K, is a kernel: $K(x_i, x_j) = \phi(x_i)^T \phi(x_j)$

- So, the problem to solve is the following:

$$\begin{aligned} \text{maximize}_\alpha \quad & \sum_i \alpha_i - \frac{1}{2} \sum_i \sum_j \alpha_i \alpha_j y_i y_j K(x_i, x_j) \\ \text{subject to} \quad & \sum_i \alpha_i y_i = 0, \quad i = 1, \ldots n, \\ & 0 \leq \alpha_i \leq C \quad i = 1, \ldots n \end{aligned}$$

  And it is still a quadratic program!!

- Kernels examples: lineal, polinomial, gaussian (radial-basis), two-layer perceptron, etc.

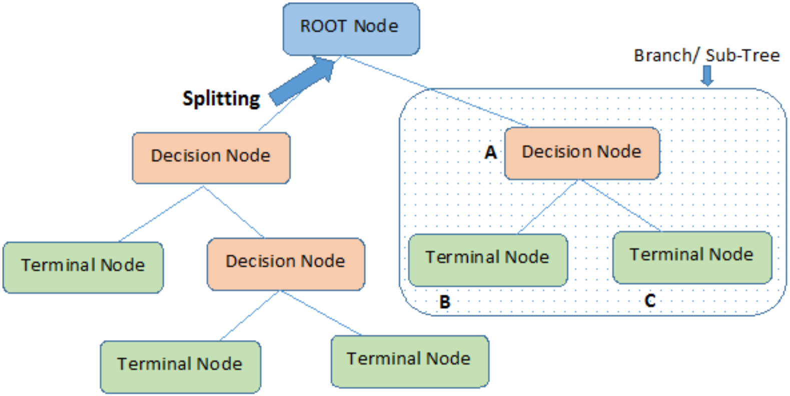
Support Vector Machines: final remarks

- The kernel tricks help introducing nonlinearity into the model in a cheap way
- ▶ https://www.youtube.com/watch?v=3liCbRZPrZA
- SVMs are the state-of-the-art classifiers in some applications like pattern recognition, in particular hand-written-character recognition
- Finding the best model requires testing of various combinations of kernels and model parameters
- Not influenced by outliers and not very prone to over-fitting
- But difficult interpretation
- The loss function of linear SVM is similar to that of Logistic Regression with a ridge-penalty term
- SVMs are expensive to train for large problems but are able to deal with non-linear data in high-dimension

Decision Trees and Random Forests

Decision Trees

- The best visual introduction I know: ▸ Link
  http://www.r2d3.us/visual-intro-to-machine-learning-part-1/



Note:- A is parent node of B and C.

Ensemble Models

Combine several models to form a powerful team,
or how create a strong team from multiple weak learners

Ensemble methods improve the predictive performance of a given model or fitting technique
Also called meta-learning methods

Mainly used in classification, but also in regression, time series, . . .

Main advantages:

- Reduce variance: results are less dependent on features of a single model and training set
- Reduce bias: combination of multiple models may produce more reliable predictions than a single model

Ensemble Models

- If we have N different models to estimate, predict, classify, etc., how can we select the best one?

- Can we select the winner among these N models (under some performance measure)?

- Maybe, but a better idea is to combine these N models
  - hint: averaging may help reducing variance, like the average $\text{Var}(\bar{x}) = \sigma^2/n$
  - when is it true?

- General principle: construct a combination of some model/methods, instead of using a single (maybe the best) method

- In practice, models in the combination will be correlated because the training set is the same or similar for all the models

Bagging

- Used in regression or classification, to reduce previous correlations

- DTs are the basis for more advanced tools: Random Forests

- DTs are fast and have small bias, but the variance is large

- Bagging = Bootstrap Aggregation

- Proposed by Leo Breiman: general-purpose procedure for reducing the variance of a statistical learning method

- Ideally we should use different training sets for each prediction model $\hat{y}$

- But we do not generally have access to multiple training sets, so we can use the bootstrap to reuse many times the (single) training set

- The reduction in the variance is related to the degree of independence (correlations) between the single models

Bagging

- Bagging uses the bootstrap to reuse the (single) training set

- For instance, if $\hat{y}$ is our prediction model for a training set (x, y), then the bagging prediction for B bootstrap replicates is

$$\hat{y}^{\text{Bag}} = \frac{1}{B} \sum_{b=1}^{B} \hat{y}_b^*(x_b^*, y_b^*)$$

  where $\hat{y}_b^*$ denotes the prediction of the b-th decision tree

- Note bagging uses equal weights and focuses on changing the training set

- For classification: use majority vote on the classification results, instead of average (as in regression)

- Out-of-bag error estimation: Bagging automatically yields an estimate of the testing (out-of-sample) error using the out-of-bag observations

Out-of-bag observations

- How many values are left out of a bootstrap resample?

- Given a sample $x_1, \ldots, x_n$ and assuming all x_i are different, the probability that a particular value x_i is left out of a bootstrap resample is

$$P(x_{u_j} \neq x_i,\ 1 \leq j \leq n) = \left(1 - \frac{1}{n}\right)^n$$

- This is because $P(x_{u_j} = x_i) = 1/n$

- When n is large, previous probability converges to $e^{-1} \simeq 0.37$

- 37% of the original sample values are usually left out

- Out-of-bag error: we can use the out-of-bag observations (as a testing set) to compute the out-of-bag error

Random Forest

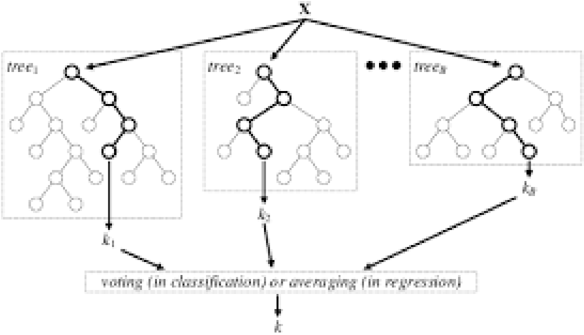
- Most-known bagging-learning tool
- Random Forest improves the predictive performance of DTs by averaging them
- It also reduces overfitting: for each node of the tree, consider only a random subset of predictors, $m < p$, for instance $m = \sqrt{p}$
- Approximately $2/3$ of the total training data is used for growing each tree
- And the remaining $1/3$ are left out and not used in the construction of each tree
- In classification, the forest chooses the group having the most votes over all the trees in the forest

Random Forest: steps

1. Draw B bootstrap samples of size n: a decision tree for each bootstrap sample

2. Grow each tree, by selecting a random set of m (out of p) features at each node, and choosing the best feature to split on

3. Choose the most popular vote over all the trees to produce the final prediction



Random Forest: final comments

- Advantages: more accurate, reduce variance and overfitting, easy to implement and parallelize, can easily handle categorical features and also non-linear relationships

- Disadvantages: less interpretation and understanding, computationally expensive, variance is not reduced enough if predictors are too correlated

Boosting

- Boosting = boost the performance of weak learners to attain the performance of stronger learners
- Ensemble proposed by Schapire and Freund to reduce mainly the bias (but also the variance) in a sequential way
- Weak learner: accuracy only a bit better than random guessing
- Boosting learns slowly (sequentially) in order to give more influence to observations that are incorrectly classified by previous learners
- The two most important types of boosting algorithms in classification are the Ada Boost (Adaptive Boosting) algorithm (Freund, Schapire, 1997) and the Arcing algorithm (Breiman, 1996)

Boosting: steps

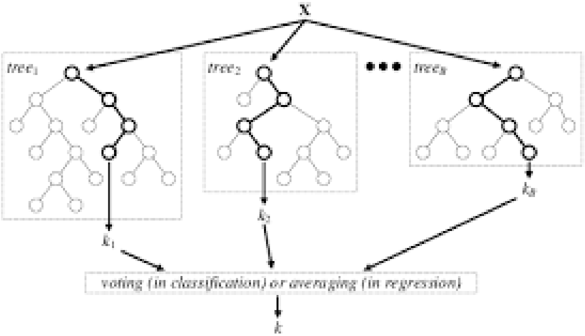
1. Assign same weights to all observations in the training set, and train a weak learner

2. Change the observation weights giving more influence to observations that were incorrectly classified by previous learners, and repeat the process

3. At the end, choose a weighted majority vote over all the weak learners to make final prediction

$$C(x) = \text{sign}(\sum_{i=1}^{M} \alpha_m C_m(x)),$$

where $\alpha_m = \log(1 - \text{error}_m)/\text{error}_m$ is a measure of individual accuracy



Gradient Boosting

- Instead of using the error of each weak learner, define a loss function and use the gradient descent to minimize the loss by adding weak learners
- I.e. instead of up-weighting observations that were misclassified before, identify large residuals from previous iterations
- The residuals are defined by a loss function and they are improved by the gradient algorithm
- The loss function is problem-dependent and must be differentiable
- It may overfit quicky. To avoid overfitting:
    - Tree Constraints (number of trees, tree depth, number of nodes, etc.)
    - Shrinkage (the learning rate)
    - Random sampling: stochastic gradient boosting
    - Penalized Learning: penalized values in terminal nodes, like XGboost

Gradient Boosting

- Advantages:

  Can handle any type of predictors

  Good predictive power

  Robustness to outliers if robust loss functions are used

- Disadvantages:

  Scalability, it is a sequential approach, hence difficult to parallelize

  Can overfit quickly, to avoid it we need to tune many hyper-parameters

Differences between Bagging and Boosting

- In boosting, each model is built on top of the previous ones whereas in bagging each model is built independently

- In bagging, the weak learners are aggregated equally whereas in boosting they are aggregated with different weights

- Bagging is a method of reducing variance while boosting focuses more on bias (but reduces also the variance)

- Bagging solves overfitting naturally while boosting needs advanced tools to deal with it

- Hence, Gradient Boosting have many hyper-parameters to tune, while Random Forest is practically tuning-free

- In general, Gradient Boosting $\succ$ Boosting $\succ$ Random Forests $\succ$ Bagging $\succ$ Decision Trees

Neural Networks

Neural Networks

The new hype: Artificial Intelligence and Deep Learning

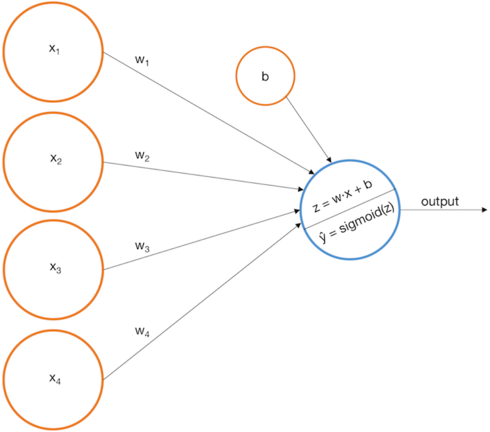
Incredible success in applications like image recognition (diagnosis in medical imaging), natural language processing (virtual personal assistants, language translator), automated trading, autonomous cars, drug discoveries, recommendation systems, . . .

Easy to understand if you know Logistic Regression:

- Artificial Neural Networks are like many logistic regressions interconnected through a network
- Each logistic regression is a single layer perceptron (one node in the network)
- The logistic function is now called sigmoid (a type of activation function)

From $p = \frac{1}{1+\exp(-\beta_0-\beta^T x)}$ to sigmoid $= \frac{1}{1+\exp(-b-w^T x)}$

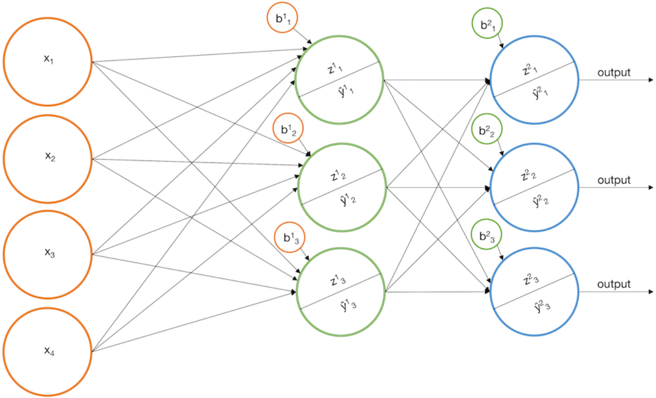
Single Layer Perceptron



Example with 4 predictors (plus constant 1) and one output (yes or no, binary classification)

Multi-Layer Perceptron

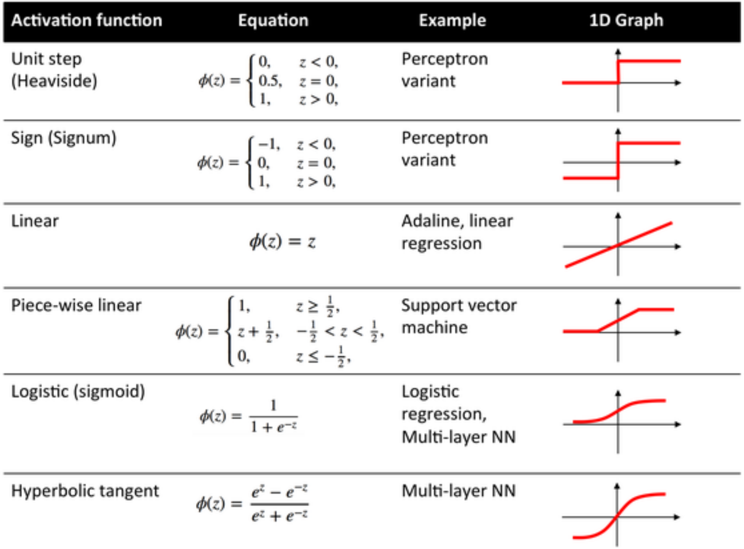
How to incorporate non-linearities and interactions? Add more layers and perceptrons (nodes)



Neural network: many layers (in the graph an input layer, a hidden layer, and an output layer with the scores of 3 classes or groups)

Alternative activation functions

The activation function, like the sigmoid, must be non-linear, otherwise the NN would be just a single layer perceptron. Its value is the input for next neurons



Activation function	Equation	Example	1D Graph
Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer NN	

Deep Learning

- Can we add just more hidden layers to improve performance?
  Yes! This is called deep learning = deep neural networks

- Take care: the Universal Approximation Theorem says you can approximate any continuous function using a NN with just a single hidden layer

- Hence, in theory no need of more than one hidden layer. But DL fits a problem faster in practice

- DL estimates the weights in the NN by minimizing a differentiable loss function through the (mini-batch) gradient descent method, computing the derivatives using back-propagation (chain's rule). In practice, millions of parameters to estimate in a highly non-linear function

- In general, larger networks tend to predict better than smaller networks, but overfitting should be avoided through regularization

- Use TensorFlow in practice (an open-source DL flexible framework) to select network topology, optimization algorithm, activation function, etc.

Classification: final remarks

No-Free-Lunch-Theorem: there is no best algorithm that works best in all cases

- Logistic classification is the most widely used technique in practice, performs well for binary classification if problem is roughly linear. It allows general predictors (features), and provides explanation, but needs some feature engineering (and regularization). The performance worsens with more than two groups. Good choice to make cost-sensitive learning

- LDA performs well for more than two groups but predictors need to be somehow Gaussian (linear problem). Suffers from collinearity unless regularization is used

- Naive Bayes is quite simple and fast, but depends on the independence condition (high bias, low variance). Performs well in text applications, allows general features, but suffers from collinearity

- For non-linear data, k-NN performs well if dimension p/n is low

Classification: final remarks

No-Free-Lunch-Theorem: there is no best algorithm that works best in all cases

- SVM is especially useful when non-linear kernels are used (non-linear problems). Especially popular in text classification problems. Not suffer from collinearity but hard to interpret. Scales bad with n and not provide probabilities

- Random Forests: good performance in practice, good for distributing computing, deals well with interactions and non-linearities. Provides variable importance and scales well with n, but provides probabilities not as reliable as LDA or LR

- Gradient Boosting: tends to perform better than RF but harder to tune parameters. Prone to overfitting. Not good for parallel computing, but performs well in practice when n is large

- Neural Networks: nowadays, one of the most successful approaches, can handle non-linear data and high dimension, but computationally expensive and difficult to avoid overfitting. A completely black-box tool but many open source implementations. Difficult to tune lots of hyper-parameters, non-linear optimization problem. Provides probabilities (scores) not as reliable as LDA or LR. Performs very well in practice when n is huge

“It is much more interesting to live with uncertainty than to live with answers that might be wrong.”

Richard Feynman, Nobel laureate in Physics

The end. . .

Thanks for your attention!!

Javier Nogales
www.est.uc3m.es/Nogales
@fjnogales