MACHINE LEARNING MODELS

XGBOOST

Gradient Boosting

Definition

*Gradient boostin*g is one of the most powerful techniques for building predictive models. It is an automatic learning technique used for regression analysis and statistical classification problems, which produces a predictive model in the form of a set of weak predictive models, typically decision trees. It builds the model in a staggered manner as other boosting methods do, and generalizes them allowing arbitrary optimization of a differentiable loss function.

This algorithm is an efficient algorithm for converting relatively poor hypotheses into very good hypotheses. A **weak hypothesis** or weak learner is defined as one whose performance is at least slightly better than random chance. *Hypothesis boosting* was the idea of filtering observations, leaving those observations that the weak learner can handle and focusing on developing new weak learns to handle the remaining difficult observations. The idea is to use the weak learning method several times to get a succession of hypotheses, each one refocused on the examples that the previous ones found difficult and misclassified.

Gradient boosting involves three elements:

1. A **loss function** to be optimized.
2. A **weak learner** to make predictions.
3. An **additive model** to add weak learners to minimize the loss function.

Loss Function

A **loss function** or **cost function** is a function that maps an event or values of
one or more variables onto a real number intuitively representing some *"cost"* associated with the event. An optimization problem seeks to minimize a loss function.

The *loss function* used depends on the type of problem being solved.

It must be *differentiable* (a **differentiable function** of one real variable is a function whose derivative exists at each point in its domain), but many standard loss functions are supported and you can define your own. For example, regression may use a squared error and classification may use logarithmic loss.

A benefit of the gradient boosting framework is that a new boosting algorithm does not have to be derived for each loss function that may want to be used, instead, it is a generic enough framework that any differentiable loss function can be used.

Weak Learner

Decision trees are used as the weak learner in gradient boosting. Specifically regression trees are used that output real values for splits and whose output can be added together, allowing subsequent models outputs to be added and “correct” the residuals in the predictions.

Trees are constructed in a greedy manner, choosing the best split points based on purity scores like Gini or to minimize the loss. Initially, very short decision trees were used that only had a single split, called a decision stump. Larger trees can be used generally with 4-to-8 levels. It is common to constrain the weak learners in specific ways, such as a maximum number of layers, nodes, splits or leaf nodes. This is to ensure that the learners remain weak, but can still be constructed in a greedy manner.

Additive Model

Trees are added one at a time, and existing trees in the model are not changed. A gradient descent procedure is used to minimize the loss when adding trees. Traditionally, gradient descent is used to minimize a set of parameters, such as the coefficients in a regression equation or weights in a neural network. After calculating error or loss, the weights are updated to minimize that error.
Instead of parameters, we have **weak learner sub-models** or more specifically decision trees. After calculating the loss, to perform the gradient descent procedure, we must **add a tree to the model** that reduces the loss (i.e. follow the gradient). We do this by parameterizing the tree, then modify the parameters of the tree and move in the right direction by reducing the residual loss.

Generally this approach is called **functional gradient descent** or **gradient descent with functions**. One way to produce a weighted combination of classifiers which optimizes the cost is by gradient descent in function space.

A fixed number of trees are added or training stops once loss reaches an acceptable level or no longer improves on an external validation dataset.

Improvements to Basic Gradient Boosting

Gradient boosting is a greedy algorithm and can overfit a training dataset quickly. It can benefit from regularization methods that penalize various parts of the algorithm and generally improve the performance of the algorithm by reducing overfitting. In this this section we will look at 4 enhancements to basic gradient boosting:

1. *Tree Constraints*
2. *Shrinkage*
3. *Random sampling*
4. *Penalized Learning*

Tree Constraints

It is important that the weak learners have skill but remain weak. There are a number of ways that the trees can be constrained. A good general heuristic is that the more constrained tree creation is, the more trees you will need in the model, and the reverse, where less constrained individual trees, the fewer trees that will be required.

Below are some constraints that can be imposed on the construction of decision trees:

- **Number of trees**, generally adding more trees to the model can be very slow to overfit. The advice is to keep adding trees until no further improvement is observed.
- **Tree depth**, deeper trees are more complex trees and shorter trees are preferred. Generally, better results are seen with 4-8 levels.
- **Number of nodes or number of leaves**, like depth, this can constrain the size of the tree, but is not constrained to a symmetrical structure if other constraints are used.
- **Number of observations per split** imposes a minimum constraint on the amount of training data at a training node before a split can be considered
- **Minimum improvement to loss** is a constraint on the improvement of any split added to a tree.

Weighted Updates

The predictions of each tree are added together sequentially. The contribution of each tree to this sum can be weighted to slow down the learning by the algorithm. This weighting is called a **shrinkage** or a **learning rate**. Each update is simply scaled by the value of the “learning rate parameter v”.

The effect is that learning is slowed down, in turn require more trees to be added to the model, in turn taking longer to train, providing a configuration trade-off between the number of trees and learning rate. Decreasing the value of v [the learning rate] increases the best value for M [the number of trees]. It is common to have small values in the range of 0.1 to 0.3, as well as values less than 0.1.

Similar to a learning rate in stochastic optimization, shrinkage reduces the influence of each individual tree and leaves space for future trees to improve the model.

Stochastic Gradient Boosting

A big insight into bagging ensembles and random forest was allowing trees to be greedily created from subsamples of the training dataset. This same benefit can be used to reduce the correlation between the trees in the sequence in gradient boosting models. This variation of boosting is called **stochastic gradient boosting**.

At each iteration a subsample of the training data is drawn at random (without replacement) from the full training dataset. The randomly selected subsample is then used, instead of the full sample, to fit the base learner.

A few variants of stochastic boosting that can be used:

- Subsample rows before creating each tree.
- Subsample columns before creating each tree
- Subsample columns before considering each split.

Generally, aggressive sub-sampling such as selecting only 50% of the data has shown to be beneficial.

Penalized Gradient Boosting

Additional constraints can be imposed on the parameterized trees in addition to their structure. Classical decision trees like CART are not used as weak learners, instead a modified form called a regression tree is used that has numeric values in the leaf nodes (also called terminal nodes). The values in the leaves of the trees can be called weights in some literature.

As such, the leaf weight values of the trees can be regularized using popular regularization functions, such as:

- L1 regularization of weights.
- L2 regularization of weights.

(See functions here)

The additional regularization term helps to smooth the final learnt weights to avoid overfitting. Intuitively, the regularized objective will tend to select a model employing simple and predictive functions.

Definition

XGBoost stands for e**X**treme **G**radient **B**oosting. XGBoost is an implementation of ***gradient boosted decision trees*** designed for speed and performance that is dominative competitive machine learning. The library is laser focused on computational speed and model performance, as such there are few frills. Nevertheless, it does offer a number of advanced features.

The implementation of the model supports the features of the scikit-learn and R implementations, with new additions like regularization. Three main forms of gradient boosting are supported:

- **Gradient Boosting:** algorithm also called gradient boosting machine including the learning rate.
- **Stochastic Gradient Boosting:** with sub-sampling at the row, column and column per split levels.
- **Regularized Gradient Boosting:** with both L1 and L2 regularization.

The library provides a system for use in a range of computing environments, not least:

- **Parallelization** of tree construction using all of your CPU cores during training.
- **Distributed Computing** for training very large models using a cluster of machines.
- **Out-of-Core Computing** for very large datasets that don't fit into memory.
- **Cache Optimization** of data structures and algorithm to make best use of hardware.

The implementation of the algorithm was engineered for efficiency of compute time and memory resources. A design goal was to make the best use of available resources to train the model. Some key algorithm implementation features include:

- **Sparse Aware** implementation with automatic handling of missing data values.
- **Block Structure** to support the parallelization of tree construction.
- **Continued Training** so that you can further boost an already fitted model on new data.

Why Use XGBoost?

The two reasons to use XGBoost are also the two goals of the project:

1. *Execution Speed.*
2. *Model Performance.*

XGBoost Execution Speed

Generally, XGBoost is fast. Really fast when compared to other implementations of gradient boosting.

XGBoost Model Performance

XGBoost dominates structured or tabular datasets on classification and regression predictive modeling problems.The evidence is that it is the go-to algorithm for competition winners on the Kaggle competitive data science platform.

What Algorithm Does XGBoost Use?

The XGBoost library implements the **gradient boosting decision tree algorithm**. This algorithm goes by lots of different names such as *gradient boosting*, *multiple additive regression trees*, *stochastic gradient boosting* or *gradient boosting machines*.

Boosting is an ensemble technique where new models are added to correct the errors made by existing models. Models are added sequentially until no further improvements can be made. **Gradient boosting** is an approach where new models are created that predict the residuals or errors of prior models and then added together to make the final prediction. It is called gradient boosting because it uses a gradient descent algorithm to minimize the loss when adding new models.
This approach supports both regression and classification predictive modeling problems.

LightGBM

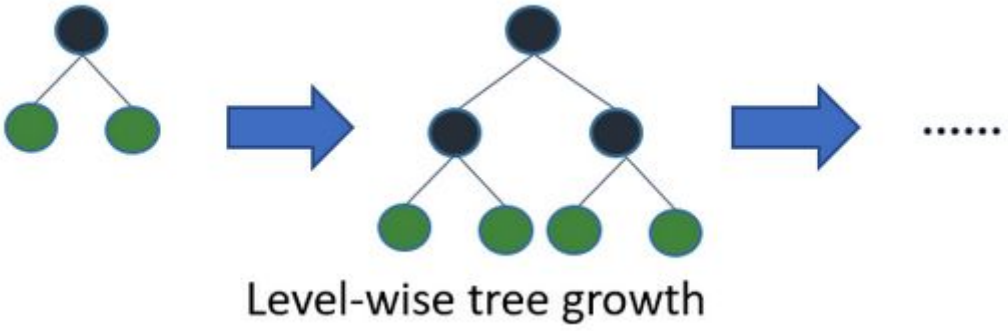
Light GBM is a gradient boosting framework that uses tree based learning algorithm; is a fast, distributed, high-performance gradient boosting framework based on decision tree algorithm, used for ranking, classification and many other machine learning tasks.

LightGBM is a great implementation that is similar to XGBoost but varies in a few specific ways, especially in how it creates the trees.**Light GBM grows tree vertically** while other algorithm grows trees horizontally meaning that Light GBM grows tree **leaf-wise** while other algorithm grows level-wise (like XGBOOST). That is, since it is based on decision tree algorithms, it splits the tree leaf wise with the best fit whereas other boosting algorithms split the tree depth wise or level wise rather than leaf-wise.

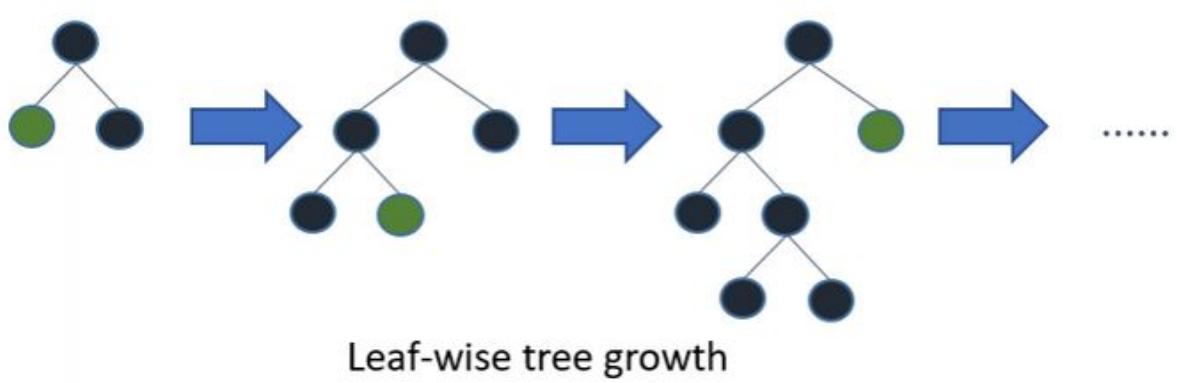
Compared with depth-wise growth, the leaf-wise algorithm can converge much faster. This is because, when growing on the same leaf in Light GBM, the leaf-wise algorithm can reduce more loss than the level-wise algorithm and hence results in much better accuracy which can rarely be achieved by any of the existing boosting algorithms. Also, it is surprisingly very fast, hence the word ‘Light’.

However, leaf-wise splits lead to increase in complexity and may lead to overfitting if not used with the appropriate parameters. It can be overcome by specifying another parameter *max-depth* which specifies the depth to which splitting will occur.

Before is a diagrammatic representation by the makers of the Light GBM to explain the difference clearly.



Level-wise tree growth in XGBOOST.



Leaf wise tree growth in Light GBM.

Advantages

1. **Faster training speed and higher efficiency**: Light GBM use histogram based algorithm i.e it buckets continuous feature values into discrete bins which fasten the training procedure.
2. **Lower memory usage:** Replaces continuous values to discrete bins which result in lower memory usage.
3. **Better accuracy than any other boosting algorithm:** It produces much more complex trees by following leaf wise split approach rather than a level-wise approach which is the main factor in achieving higher accuracy. However, it can sometimes lead to overfitting which can be avoided by setting the max_depth parameter.
4. **Compatibility with Large Datasets:** It is capable of performing equally good with large datasets with a significant reduction in training time as compared to XGBOOST.
5. **Parallel learning supported.**

Why Light GBM is gaining extreme popularity?

The size of data is increasing day by day and it is becoming difficult for traditional data science algorithms to give faster results. Light GBM is prefixed as ‘Light’ because of its **high speed.** Light GBM can **handle the large size** of data and **takes lower memory to run**. Another reason of why Light GBM is popular is

because it **focuses on accuracy of results**. LGBM also **supports GPU learning** and thus data scientists are widely using LGBM for data science application development.

Why don't more people use it then?

XGBoost has been around longer and is already installed on many machines. LightGBM is rather new and didn't have a Python wrapper at first. The current version is easier to install and use so no obstacles here. Many of the more advanced users on Kaggle and similar sites already use LightGBM and for each new competition, it gets more and more coverage. Still, the starter scripts are often based around XGBoost as people just reuse their old code and adjust a few parameters. I'm sure this will increase once there are a few more tutorials and guides on how to use it (most of the non-ScikitLearn guides currently focus on XGBoost or neural networks).

Can we use Light GBM everywhere?

No, it is not advisable to use LGBM on small datasets. Light GBM is **sensitive to overfitting** and can easily overfit small data. There is no threshold on the number of rows but my experience suggests me to use it only for data with 10,000+ rows.

Parameters

Control Parameters

- **max_depth:** It describes the maximum depth of tree. This parameter is used to handle model overfitting. Any time you feel that your model is overfitted, my first advice will be to lower max_depth.
- **min_data_in_leaf:** It is the minimum number of the records a leaf may have. The default value is 20, optimum value. It is also used to deal overfitting
- **feature_fraction:** Used when your boosting is random forest. 0.8 feature fraction means LightGBM will select 80% of parameters randomly in each iteration for building trees.
- **bagging_fraction:** specifies the fraction of data to be used for each iteration and is generally used to speed up the training and avoid overfitting.
- **early_stopping_round:** This parameter can help you speed up your analysis. Model will stop training if one metric of one validation data doesn’t improve in last early_stopping_round rounds. This will reduce excessive iterations.
- **lambda:** lambda specifies regularization. Typical value ranges from 0 to 1.

- **min_gain_to_split:** This parameter will describe the minimum gain to make a split. It can used to control number of useful splits in tree.
- **max_cat_group:** When the number of category is large, finding the split point on it is easily overfitting. So LightGBM merges them into 'max_cat_group' groups, and finds the split points on the group boundaries. Default:64

Core Parameters

- **task:** It specifies the task you want to perform on data. It may be either train or predict.
- **application:** This is the most important parameter and specifies the application of your model, whether it is a regression problem or classification problem. LightGBM will by default consider model as a regression model.
    - *regression*: for regression
    - *binary*: for binary classification
    - *multiclass*: for multiclass classification problem
- **boosting:** defines the type of algorithm you want to run, default=gdbt
    - *gbdt*: traditional Gradient Boosting Decision Tree
    - *rf*: random forest
    - *dart*: Dropouts meet Multiple Additive Regression Trees
    - *goss*: Gradient-based One-Side Sampling
- **num_boost_round:** Number of boosting iterations, typically 100+
- **learning_rate:** This determines the impact of each tree on the final outcome. GBM works by starting with an initial estimate which is updated using the output of each tree. The learning parameter controls the magnitude of this change in the estimates. Typical values: 0.1, 0.001, 0.003…
- **num_leaves:** number of leaves in full tree, default: 31
- **device:** default: cpu, can also pass gpu

Metric parameter

- **metric:** again one of the important parameter as it specifies loss for model building. Below are few general losses for regression and classification.
    - mae: mean absolute error
    - mse: mean squared error
    - binary_logloss: loss for binary classification
    - multi_logloss: loss for multi classification

IO parameters

- **max_bin:** it denotes the maximum number of bin that feature value will bucket in.
- **categorical_feature:** It denotes the index of categorical features. If categorical_features=0,1,2 then column 0, column 1 and column 2 are categorical variables.
- **ignore_column:** same as categorical_features just instead of considering specific columns as categorical, it will completely ignore them.
- **save_binary:** If you are really dealing with the memory size of your data file then specify this parameter as 'True'. Specifying parameter true will save the dataset to binary file, this binary file will speed your data reading time for the next time.

Knowing and using above parameters will definitely help you implement the model. Remember I said that implementation of LightGBM is easy but parameter tuning is difficult. So let's first start with implementation and then I will give idea about the parameter tuning.

Multiple Layer Perception

Definition

The field of artificial neural networks is often just called **neural networks** or **multi-layer perceptrons** after perhaps the most useful type of neural network. A perceptron is a single neuron model that was a precursor to larger neural networks.

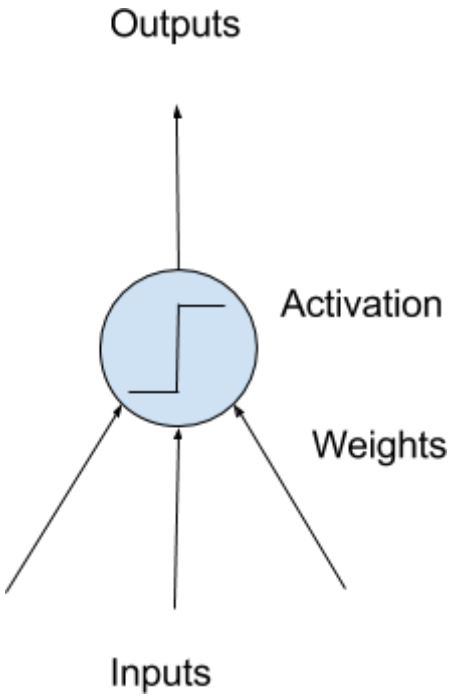
It is a field that investigates how simple models of biological brains can be used to solve difficult computational tasks like the predictive modeling tasks we see in machine learning. The goal is not to create realistic models of the brain, but instead to develop robust algorithms and data structures that we can use to model difficult problems.

The power of neural networks come from their ability to learn the representation in your training data and how to best relate it to the output variable that you want to predict. In this sense neural networks learn a mapping. Mathematically, they are capable of learning any mapping function and have been proven to be a universal approximation algorithm.

The predictive capability of neural networks comes from the hierarchical or multi-layered structure of the networks. The data structure can pick out (learn to represent) features at different scales or resolutions and combine them into higher-order features. For example from lines, to collections of lines to shapes.

Neurons

The building block for neural networks are **artificial neurons**. These are simple computational units that have weighted input signals and produce an output signal using an *activation function*.



Neuron Weights

You may be familiar with linear regression, in which case the weights on the inputs are very much like the coefficients used in a regression equation. Like linear regression, each neuron also has a bias which can be thought of as an input that always has the value 1.0 and it too must be weighted.

For example, a neuron may have two inputs in which case it requires three weights. One for each input and one for the bias. Weights are often initialized to small random values, such as values in the range 0 to 0.3, although more complex initialization schemes can be used.

Like linear regression, larger weights indicate increased complexity and fragility. It is desirable to keep weights in the network small and regularization techniques can be used.

Activation

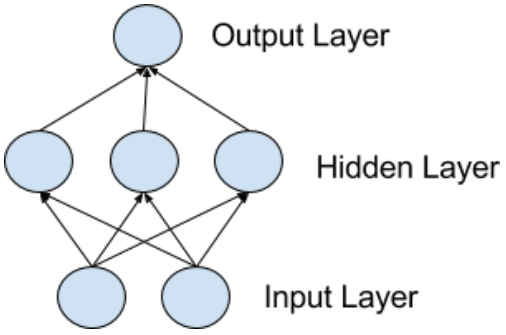
The weighted inputs are summed and passed through an activation function, sometimes called a **transfer function**. An *activation function* is a simple mapping of summed weighted input to the output of the neuron. It is called an activation function because it governs the threshold at which the neuron is activated and strength of the output signal.

Historically simple step activation functions were used where if the summed input was above a threshold, for example 0.5, then the neuron would output a value of 1.0, otherwise it would output a 0.0.

Traditionally non-linear activation functions are used. This allows the network to combine the inputs in more complex ways and in turn provide a richer capability in the functions they can model. Non-linear functions like the logistic also called the *sigmoid function* were used that output a value between 0 and 1 with an s-shaped distribution, and the hyperbolic tangent function also called tanh that outputs the same distribution over the range -1 to +1. More recently the rectifier activation function has been shown to provide better results.

Networks of Neurons

Neurons are arranged into networks of neurons. A row of neurons is called a **layer** and one network can have multiple layers. The architecture of the neurons in the network is often called the **network topology**.



Input or Visible Layers

The bottom layer that takes input from your dataset is called the **visible layer**, because it is the exposed part of the network. Often a neural network is drawn with a visible layer with one neuron per input value or column in your dataset. These *are not neurons as described above*, but simply pass the input value though to the next layer.

Hidden Layers

Layers after the input layer are called **hidden layers** because that are not directly exposed to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value.

Given increases in computing power and efficient libraries, very deep neural networks can be constructed. *Deep learning* can refer to having many hidden layers in your neural network. They are deep because they would have been unimaginably slow to train historically, but may take seconds or minutes to train using modern techniques and hardware.

Output Layer

The final hidden layer is called the **output layer** and it is responsible for outputting a value or vector of values that correspond to the format required for the problem.

The choice of activation function in he output layer is strongly constrained by the type of problem that you are modeling. For example:

- A regression problem may have a single output neuron and the neuron may have no activation function.
- A binary classification problem may have a single output neuron and use a sigmoid activation function to output a value between 0 and 1 to represent the probability of predicting a value for the class 1. This can be turned into a crisp class value by using a threshold of 0.5 and snap values less than the threshold to 0 otherwise to 1.
- A multi-class classification problem may have multiple neurons in the output layer, one for each class (e.g. three neurons for the three classes in the famous iris flowers classification problem). In this case a softmax activation function may be used to output a probability of the network predicting each of the class values. Selecting the output with the highest probability can be used to produce a crisp class classification value.

Training Networks

Once configured, the neural network needs to be trained on your dataset.

Data Preparation

You must first prepare your data for training on a neural network. Data must be numerical, for example real values. If you have categorical data, such as a sex attribute with the values “male” and “female”, you can convert it to a real-valued representation called a one hot encoding. This is where one new column is added for each class value (two columns in the case of sex of male and female) and a 0 or 1 is added for each row depending on the class value for that row.
This same one hot encoding can be used on the output variable in classification problems with more than one class. This would create a binary vector from a single column that would be easy to directly compare to the output of the neuron in the network’s output layer, that as described above, would output one value for each class.

Neural networks require the input to be scaled in a consistent way. You can rescale it to the range between 0 and 1 called **normalization**. Another popular technique is to standardize it so that the distribution of each column has the mean of zero and the standard deviation of 1.

Scaling also applies to image pixel data. Data such as words can be converted to integers, such as the popularity rank of the word in the dataset and other encoding techniques.

Stochastic Gradient Descent

The classical and still preferred training algorithm for neural networks is called **stochastic gradient descent**.This is where one row of data is exposed to the network at a time as input. The network processes the input upward activating neurons as it goes to finally produce an output value. This is called a *forward pass* on the network. It is the type of pass that is also used after the network is trained in order to make predictions on new data.

The output of the network is compared to the expected output and an error is calculated. This error is then propagated back through the network, one layer at a time, and the weights are updated according to the amount that they contributed to the error. This clever bit of math is called the backpropagation algorithm.

The process is repeated for all of the examples in your training data. One of updating the network for the entire training dataset is called an *epoch*. A network may be trained for tens, hundreds or many thousands of epochs.

Weight Updates

The weights in the network can be updated from the errors calculated for each training example and this is called **online learning**. It can result in fast but also chaotic changes to the network. Alternatively, the errors can be saved up across all of the training examples and the network can be updated at the end. This is called **batch learning** and is often more stable.

Typically, because datasets are so large and because of computational efficiencies, the size of the batch, the number of examples the network is shown before an update is often reduced to a small number, such as tens or hundreds of examples. The amount that weights are updated is controlled by a configuration parameters called the **learning rate**. It is also called the *step size* and controls the step or change made to network weight for a given error. Often small weight sizes are used such as 0.1 or 0.01 or smaller.

The update equation can be complemented with additional configuration terms that you can set.

- *Momentum* is a term that incorporates the properties from the previous weight update to allow the weights to continue to change in the same direction even when there is less error being calculated.
- *Learning Rate Decay* is used to decrease the learning rate over epochs to allow the network to make large changes to the weights at the beginning and smaller fine tuning changes later in the training schedule.

Prediction

Once a neural network has been trained it can be used to make predictions. You can make predictions on test or validation data in order to estimate the skill of the model on unseen data. You can also deploy it operationally and use it to make predictions continuously.

The network topology and the final set of weights is all that you need to save from the model. Predictions are made by providing the input to the network and performing a forward-pass allowing it to generate an output that you can use as a prediction.

Disadvantages

The disadvantages of Multi-layer Perceptron (MLP) include:

- MLP with hidden layers have a non-convex loss function where there exists more than one local minimum. Therefore different random weight initializations can lead to different validation accuracy.
- MLP requires tuning a number of hyperparameters such as the number of hidden neurons, layers, and iterations.
- MLP is sensitive to feature scaling.

ARIMA

Series temporales

Definición

Una **serie temporal** es una colección ordenada de mediciones tomadas en intervalos regulares; por ejemplo, los precios diarios de las acciones o los datos de ventas semanales. Los intervalos pueden representar cualquier unidad de tiempo, pero debe utilizarse un mismo intervalo para todas las mediciones. Además, si algún intervalo no tiene ninguna medición, debe definirse en el valor perdido. De esta forma, el número de intervalos con mediciones (incluidos los que tienen valores perdidos) define la duración del período histórico de los datos.

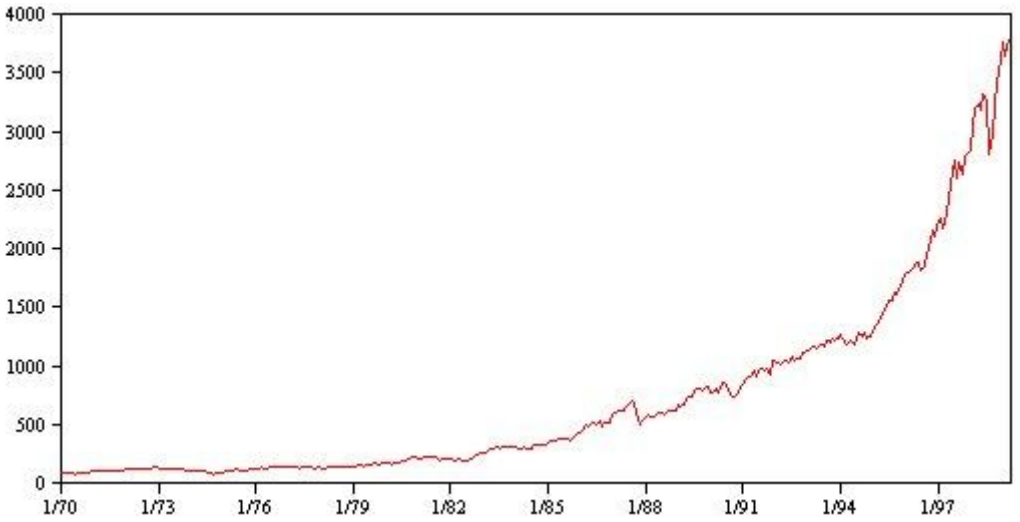
Características de las series temporales

Estudiar el comportamiento pasado de una serie le ayudará a identificar los patrones y realizar mejores previsiones. Cuando se representan, muchas series temporales muestran una o varias de estas características:

- Tendencias
- Ciclos estacionales y no estacionales
- Pulsos y pasos
- Valores atípicos

Tendencias

Una **tendencia** es un cambio gradual ascendente o descendente en el nivel de la serie o la trayectoria que siguen los valores de la serie de aumentar o disminuir a lo largo del tiempo.



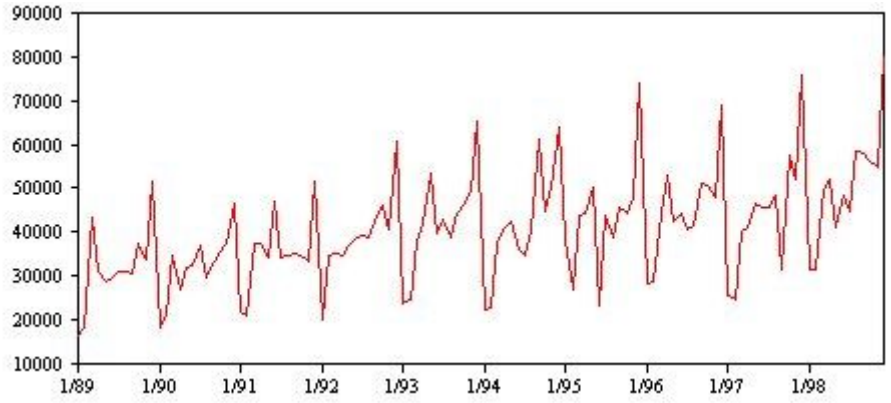
Las tendencias pueden ser **locales** o **globales**, pero una misma serie puede mostrar ambas. Históricamente, los gráficos de series del índice del mercado de valores muestran una tendencia global ascendente. Han aparecido tendencias descendentes locales en épocas de recesión y tendencias ascendentes locales en épocas de prosperidad.

Las tendencias también pueden ser **lineales** o **no lineales**. Las tendencias lineales son incrementos aditivos positivos o negativos en el nivel de la serie, comparables al efecto del interés simple sobre el principal. Las tendencias no lineales suelen ser multiplicativas, con incrementos proporcionales a los valores de series anteriores.

Las tendencias lineales globales son adecuadas y hacen previsiones correctas tanto con los modelos ARIMA como con los de suavizado exponencial. **Al generar modelos ARIMA, suelen diferenciarse las series que muestran tendencias para eliminar el efecto de éstas**.

Ciclos estacionales

Un **ciclo estacional** es un patrón repetitivo y predecible de los valores de las series.

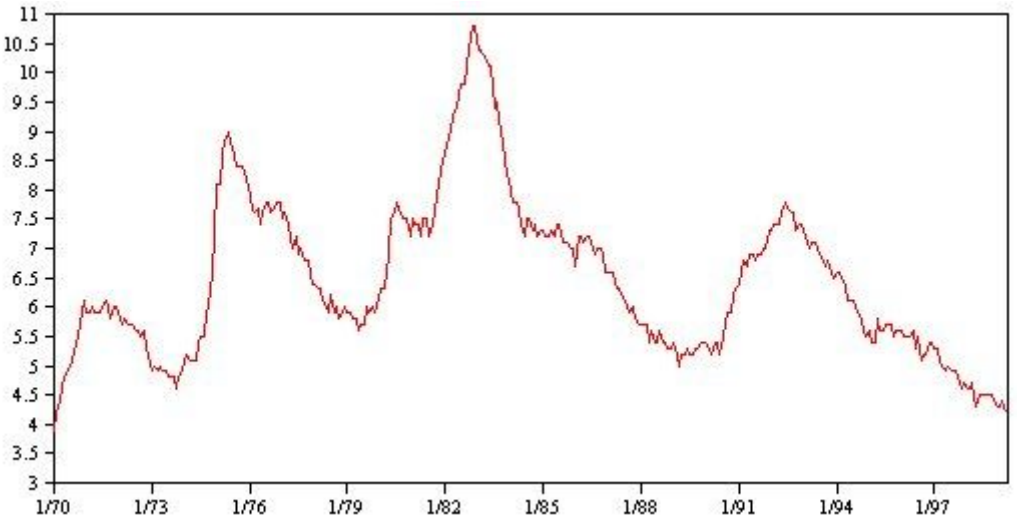


Los ciclos estacionales están ligados al intervalo de la serie. Por ejemplo, los datos mensuales suelen mostrar un comportamiento cíclico a lo largo de trimestres y años. Una serie mensual puede mostrar un ciclo trimestral significativo con un mínimo en el primer trimestre o un ciclo anual con un pico en cada mes de diciembre. Se dice que las series con un ciclo estacional muestran **estacionalidad**.

Los patrones estacionales resultan útiles para obtener buenos ajustes y previsiones. Hay modelos ARIMA y de suavizado exponencial que capturan la estacionalidad.

Ciclos no estacionales

Un **ciclo no estacional** es un patrón repetitivo y posiblemente impredecible de los valores de las series.



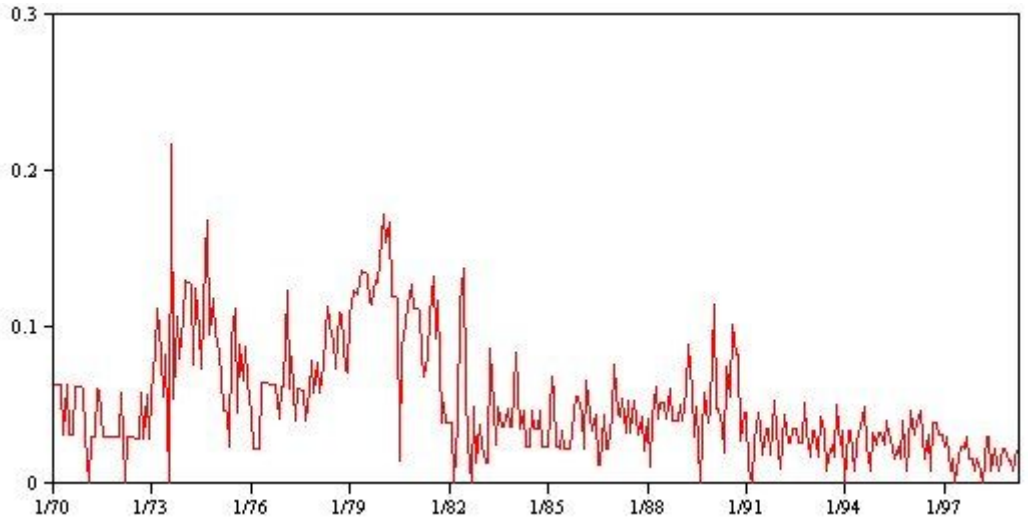
Algunas series, como la tasa de desempleo, muestran un claro comportamiento cíclico; no obstante, la periodicidad del ciclo varía a lo largo del tiempo, por lo que resulta difícil predecir cuándo se van a producir máximos o mínimos. Otras series pueden tener ciclos predecibles, pero no se ajustan exactamente al calendario gregoriano o tienen ciclos que se prolongan más de un año. Por ejemplo, las mareas siguen el calendario lunar, los viajes y el comercio internacionales relacionados con los Juegos Olímpicos aumentan cada cuatro años, y hay muchas festividades religiosas cuyas fechas gregorianas cambian de un año a otro.

Los patrones cíclicos no estacionales son difíciles de modelar y suelen aumentar la incertidumbre de las previsiones. El mercado de valores, por ejemplo, proporciona numerosos ejemplos de series que han desafiado el trabajo de los que hacen las previsiones. No obstante, los patrones no estacionales se deben tener en cuenta cuando existen. En muchos casos, aún así es posible identificar un modelo que se ajuste a los datos históricos razonablemente bien, lo que le ofrece una oportunidad excelente para minimizar la incertidumbre en las previsiones.

Pulsos y pasos

Muchas series experimentan cambios bruscos de nivel. Normalmente son de dos tipos:

- Un cambio repentino y *temporal*, o **pulso**, en el nivel de la serie.
- Un cambio repentino y *permanente*, o **paso**, en el nivel de la serie.



Cuando se observan pasos o pulsos, es importante encontrar una explicación convincente. Los modelos de series temporales están diseñados para explicar cambios graduales y no repentinos. Por tanto, suelen subestimar los pulsos y pueden quedar inutilizados por los pasos, lo que da como resultado modelos poco ajustados y previsiones imprecisas. (Es posible que algunos casos de estacionalidad parezcan presentar cambios repentinos de nivel, pero que el nivel sea constante de un período estacional a otro.)

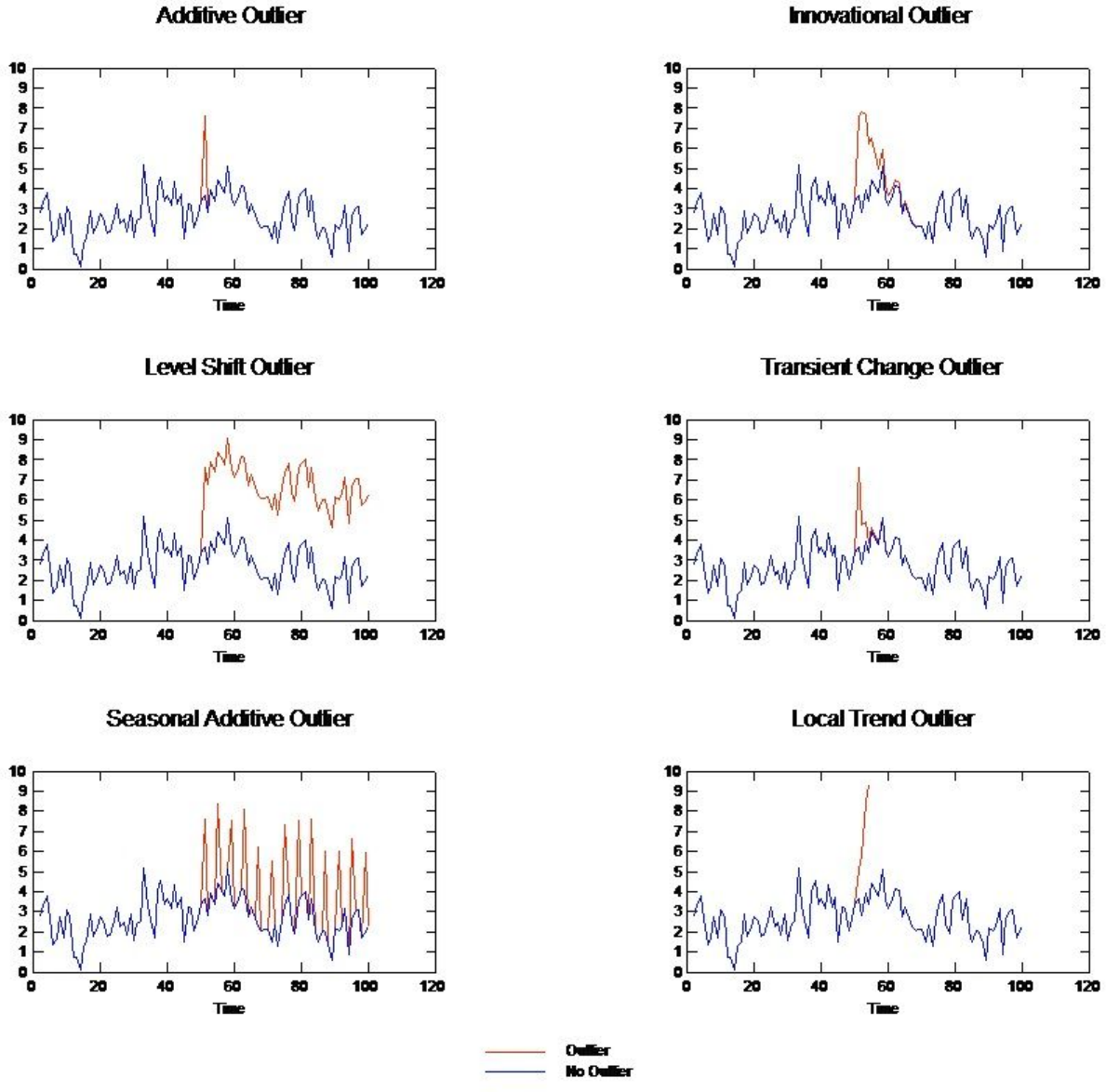
Si puede explicarse una alteración, se puede modelar mediante una **intervención** o un **evento**. Por ejemplo, en agosto de 1973, la Organización de Países Exportadores de Petróleo (OPEP) impuso un embargo sobre el petróleo que cambió drásticamente la tasa de inflación, aunque recuperó sus niveles normales en los meses siguientes. Si especifica una **intervención por puntos** para el mes del embargo, puede mejorar el ajuste del modelo, lo que mejorará las previsiones indirectamente. Por ejemplo, puede que un comercio minorista descubra que sus ventas se incrementaron mucho más de lo normal un día que todos los artículos se rebajaron un 50%. Si se especifica una promoción de rebajas del 50% como **evento** recurrente, puede mejorar el ajuste del modelo y estimar la repercusión que tendría esa misma promoción en el futuro.

Valores atípicos

Los desplazamientos en el nivel de una serie temporal que no se pueden explicar se denominan **valores atípicos**. Estas observaciones no coinciden con el

resto de las series y pueden influir considerablemente en el análisis y, por lo tanto, afectar a la capacidad de previsión del modelo de serie temporal.

En la siguiente figura se muestran los distintos tipos de valores atípicos que se producen normalmente en las series temporales. Las líneas azules representan una serie sin valores atípicos. Las líneas rojas sugieren un patrón que podría estar presente si la serie contuviera valores atípicos. Estos valores atípicos se clasifican todos como **deterministas** porque afectan únicamente al nivel de la media de la serie.



- **Valor atípico aditivo.** Un valor atípico aditivo aparece como un valor inesperadamente alto o bajo que se produce para una única observación. Las siguientes observaciones no se ven afectadas por un

valor atípico aditivo. Los valores atípicos aditivos consecutivos se denominan normalmente **parches de valores atípicos aditivos**.

- **Valor atípico innovador.** Un valor atípico innovador se caracteriza por un impacto inicial con efectos que se extienden sobre las siguientes observaciones. La influencia de los valores atípicos puede aumentar mientras avanza el tiempo.

- **Valor atípico de cambio de nivel.** En el cambio de nivel, todas las observaciones que aparecen después del valor atípico se desplazan a un nuevo nivel. A diferencia de los valores atípicos aditivos, un valor atípico de cambio de nivel afecta a diversas observaciones y tiene un efecto permanente.

- **Valor atípico de cambio transitorio.** Los valores atípicos de cambio transitorio son similares a los valores atípicos de cambio de nivel, pero su efecto se reduce exponencialmente en las siguientes observaciones. Finalmente, las series vuelven a su nivel normal.

- **Valor atípico aditivo estacional.** Un valor atípico aditivo estacional aparece como un valor inesperadamente alto o bajo que se produce repetidamente en intervalos regulares.

- **Valor atípico de tendencia local.** Un valor atípico de tendencia local produce un cambio general en la serie causado por un patrón en los valores atípicos después de la aparición del valor atípico inicial.

La detección de valores atípicos en una serie temporal implica determinar la ubicación, tipo y magnitud de todos los valores atípicos presentes. Tsay (1988) propuso un procedimiento iterativo para detectar el cambio del nivel de la media con el fin de identificar los valores atípicos deterministas. Este proceso implica la comparación de un modelo de serie temporal que supone que no hay presentes valores atípicos con otro modelo que incorpore valores atípicos. Las diferencias entre modelos permiten calcular el efecto de tratar cualquier punto como un valor atípico.

Componentes de las series temporales

El estudio descriptivo de series temporales se basa en la idea de descomponer la variación de una serie en varias componentes básicas. Este enfoque no siempre resulta ser el más adecuado, pero es interesante cuando en la serie se observa cierta tendencia o cierta periodicidad. Hay que resaltar que esta descomposición no es en general única.

Este enfoque descriptivo consiste en encontrar componentes que correspondan a una tendencia a largo plazo, un comportamiento estacional y una parte aleatoria. Las componentes o fuentes de variación que se consideran habitualmente son las siguientes:

1. **Tendencia secular o regular**: Se puede definir como un cambio a largo plazo que se produce en relación al nivel medio, o el cambio a largo plazo de la media. La tendencia se identifica con un movimiento suave de la serie a largo plazo. La notaremos t_t.
2. **Efecto Estacional (Variación estacional):** Muchas series temporales presentan cierta periodicidad o dicho de otro modo, variación de cierto periodo (anual, mensual ...). Por ejemplo, el paro laboral aumenta en general en invierno y disminuye en verano. Estos tipos de efectos son fáciles de entender y se pueden medir explícitamente o incluso se pueden eliminar del conjunto de los datos, desestacionalizando la serie original. La notaremos e_t.
3. **Variación cíclica:** Representa la pauta de comportamiento de la serie de carácter periódico, con periodos de duración diferente, desconocida y superiores a un año: por ejemplo los ciclos económicos de prosperidad, recesion y recuperación. Resulta difícil separar esta componente de la tendencia secular, ya que para ello es necesario disponer de una serie larga y con un número de ciclos completo. La notaremos c_t.
4. **Componente Aleatoria (Variación aleatoria, residual, irregular o accidental)**: Una vez identificados los componentes anteriores y después de haberlos eliminado, persisten unos valores que son aleatorios. Se pretende estudiar qué tipo de comportamiento aleatorio presentan estos residuos, utilizando algún tipo de modelo probabilístico que los describa. La notaremos r_t.

De estas cuatro componentes, se le suele dar más importancia a la primera, a la segunda y a la cuarta.

Es necesario aislar de alguna manera la componente aleatoria y estudiar qué modelo probabilístico es el más adecuado. Conocido éste, podremos conocer el

comportamiento de la serie a largo plazo. Este aislamiento de la componente aleatoria se suele abordar de dos maneras:

1. **Enfoque descriptivo**: Se estima t_t y e_t y se obtiene r_t como

$$r_t = Y_t - t_t - e_t$$

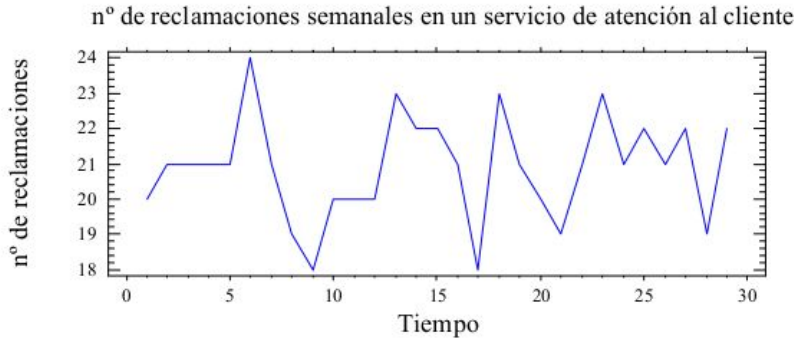
2. **Enfoque de Box-Jenkins**: Se elimina de Y_t la tendencia y la parte estacional (mediante transformaciones o filtros) y queda sólo la parte probabilística. A esta última parte se le ajustan modelos paramétricos.

*Nota: La **desestacionalización** es el proceso que se lleva a cabo para eliminar las variaciones estacionales y eliminarlas del comportamiento global de la serie, para poder observar mejor el movimiento de ésta. Por ejemplo, si se observa que las ventas trimestrales de un supermercado al pasar del tercer trimestre al cuarto*
aumentan en 30 millones. ¿ Qué ha ocurrido?, ¿ se debe este aumento a la eficacia publicitaria y del personal de la empresa o a que en el cuarto trimestre están las fiestas de Navidad y el consumo familiar aumenta?. Si observamos las ventas en el segundo y cuarto trimestres son estacionalmente altas y en el primero y tercero son estacionalmente bajas. Luego si se quiere analizar la evolución real de las ventas del supermercado hay que desestacionalizar la serie, con lo que se podrán comparar los distintos trimestres.

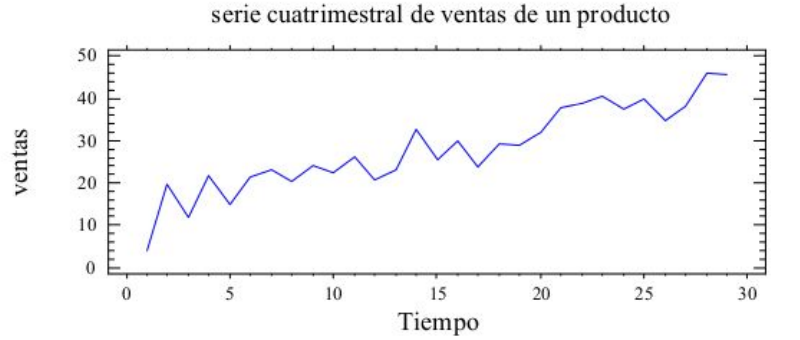
Análisis descriptivo de series temporales

La primera herramienta descriptiva básica es el **gráfico temporal**. Un gráfico temporal se construye situando los valores de la serie en el eje de ordenadas y los instantes temporales en el eje de abscisas. Construir este gráfico es de gran utilidad para observar el comportamiento de la serie temporal. Se presentan las siguientes series temporales:

Ejemplo 1



Ejemplo 2



Tipos de esquemas para series temporales

Las *cuatro componentes* enumeradas determinan conjuntamente los valores de la variable analizada en cada instante, sin que pueda valorarse con precisión el influjo individual de cada una de ellas.

Dos son los esquemas generalmente más admitidos sobre la forma en que la serie temporal se descompone en sus cuatro componentes: el **aditivo** y el **multiplicativo**.

Esquema (modelo) aditivo

Supone que las observaciones se generan como suma de las cuatro componentes,
es decir:

$$Y_t = t_t + c_t + e_t + r_t$$

En este caso cada componente se expresa en el mismo tipo de unidad que las observaciones. La variación residual, en este modelo, es independiente de las demás componentes, es decir la magnitud de dichos residuos no depende del valor que tome cualquier otra componente de la serie, (análogamente la variación estacional y la cíclica son independientes de las demás componentes).

Esquema (modelo) multiplicativo

Supone que las observaciones se generan como producto de las cuatro componentes, es decir:

$$Y_t = t_t \times c_t \times e_t \times r_t$$

En este modelo (multiplicativo puro) la *tendencia secular* se expresa en el mismo tipo de unidad que las observaciones, y el resto de las componentes en tanto por uno. Aquí no se cumple la hipótesis de independencia del esquema aditivo.

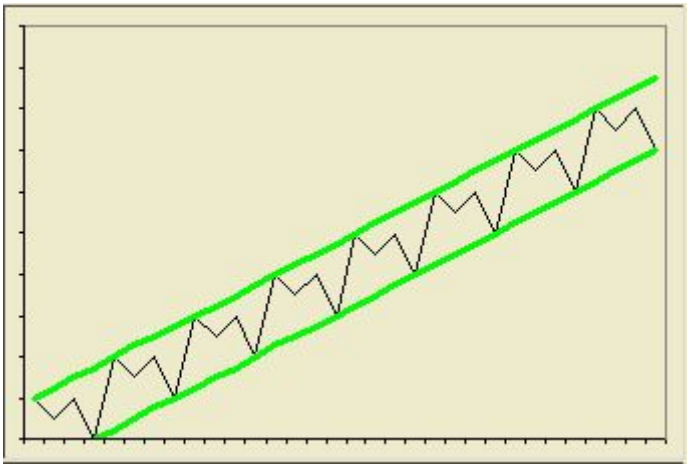
Otro tipo de modelo multiplicativo que si la cumple llamado **modelo multiplicativo mixto** es el siguiente:

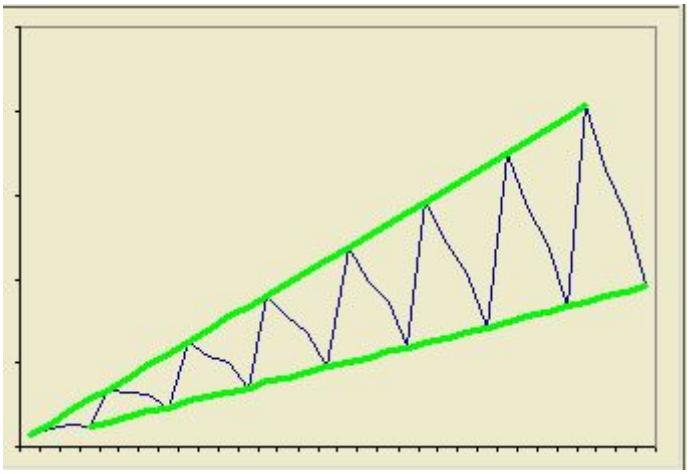
$$Y_t = t_t \times c_t \times e_t + r_t$$

Existen otros modelos que combinan esquemas aditivos y multiplicativos, tratando de resolver las carencias o inconvenientes de los modelos más sencillos. Señalaremos que generalmente el modelo multiplicativo es el que mejor se adapta a la descripción de variables económicas.

Distinción de los distintos esquemas

Para diferenciar un esquema de otro, vamos a observar las siguientes imágenes:





En la primera imagen, la tendencia se ha agregado a la estacionalidad. Por ello, la banda que ocupa la serie es siempre del mismo grosor.

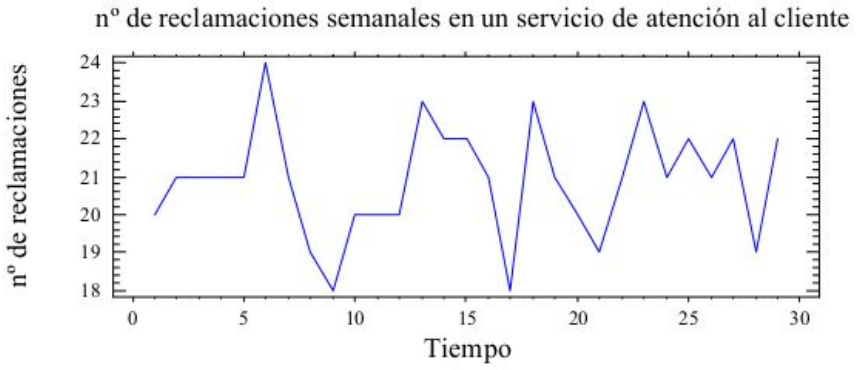
En la segunda imagen, la tendencia se multiplica por la estacionalidad. Por eso, el efecto de ésta es superior cuanto mayor sea la tendencia. Esta observación distingue el primer modelo de los dos últimos. La distinción entre el modelo multiplicativo y el mixto tiene que ver con el comportamiento de la componente irregular.

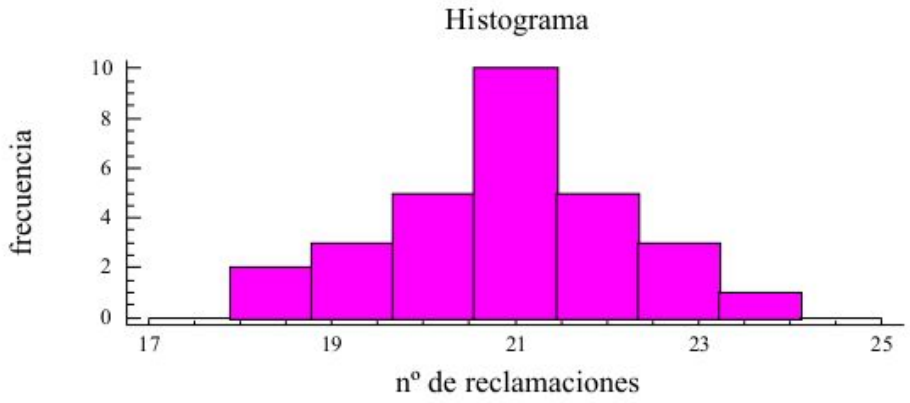
Clasificación descriptiva de las series temporales

Las series temporales se pueden clasificar en:

- **Estacionarias**: Una serie es estacionaria cuando es estable, es decir, cuando la media y la variabilidad son constantes a lo largo del tiempo. Esto se refleja gráficamente en que los valores de la serie tienden a oscilar alrededor de una media constante y la variabilidad con respecto a esa media también permanece constante en el tiempo. Es una serie básicamente estable a lo largo del tiempo, sin que se aprecien aumentos o disminuciones sistemáticos de sus valores. Para este tipo de series tiene sentido conceptos como la media y la varianza. Sin embargo, también es posible aplicar los mismos métodos a series no estacionarias si se transforman previamente en estacionarias. En el *Ejemplo 1* se presenta una serie estacionaria discreta. La serie es estable alrededor de un valor central. Si representamos un histograma de esta serie, podemos describir adecuadamente la información: en promedio, se reciben unas 21 reclamaciones semanales. Este número es bastante estable y la distribución de la variable es aproximadamente simétrica. La mejor predicción para el próximo valor de la serie es la media, aunque lo ideal sería aplicar los modelos de Inferencia para series estacionarias que se presentarán más adelante.

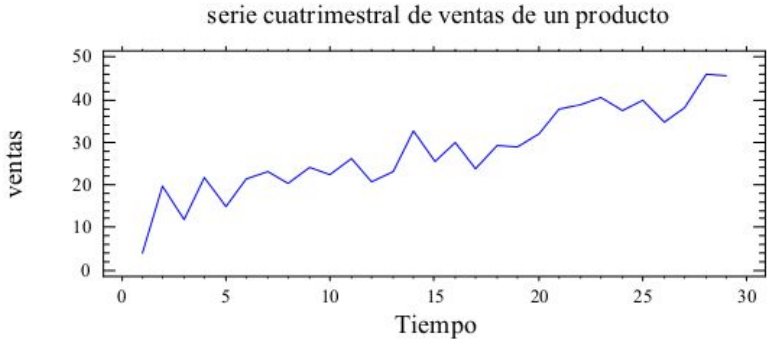
Ejemplo 1



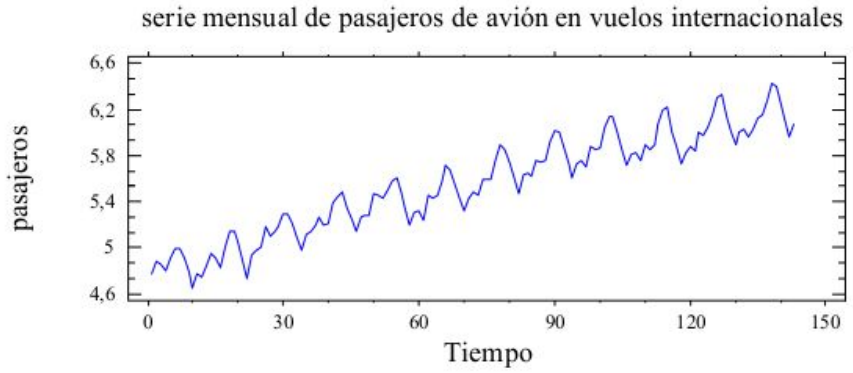


- **No Estacionarias**: Son series en las cuales la media y/o variabilidad cambian en el tiempo. Los cambios en la media determinan una tendencia a crecer o decrecer a largo plazo, por lo que la serie no oscila alrededor de un valor constante. Por ejemplo, la serie del Ejemplo 2 presenta una fuerte tendencia creciente aunque existen importantes oscilaciones con relación a esa tendencia de crecimiento lineal. La serie del ejemplo 3 presenta además de de una tendencia creciente, una pauta estacional debido a que los pasajeros transportados en los meses de verano es mayor que en el resto del año. La serie del Ejemplo 4 muestra cambios de nivel y oscilaciones erráticas sin una pauta clara. La segunda mitad de la serie presenta una tendencia decreciente, estabilizándose en la parte final de la serie.

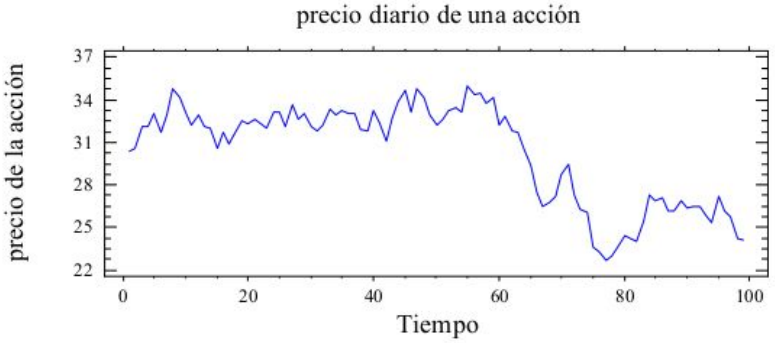
Ejemplo 2



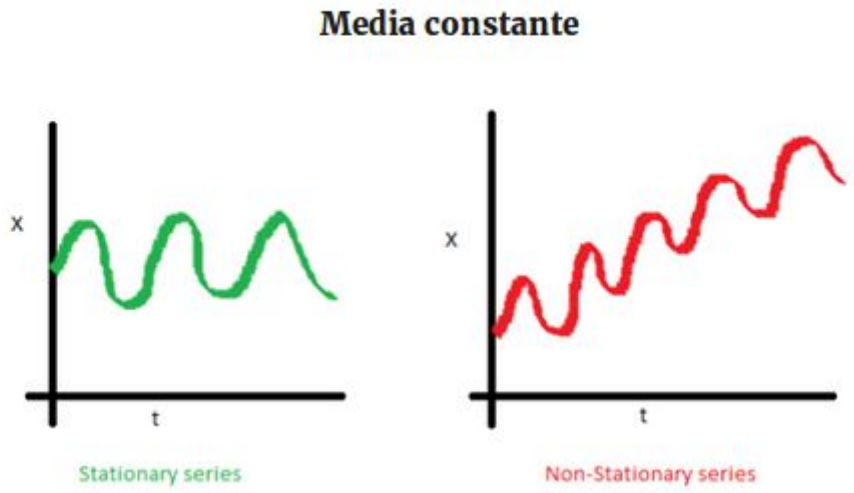
Ejemplo 3



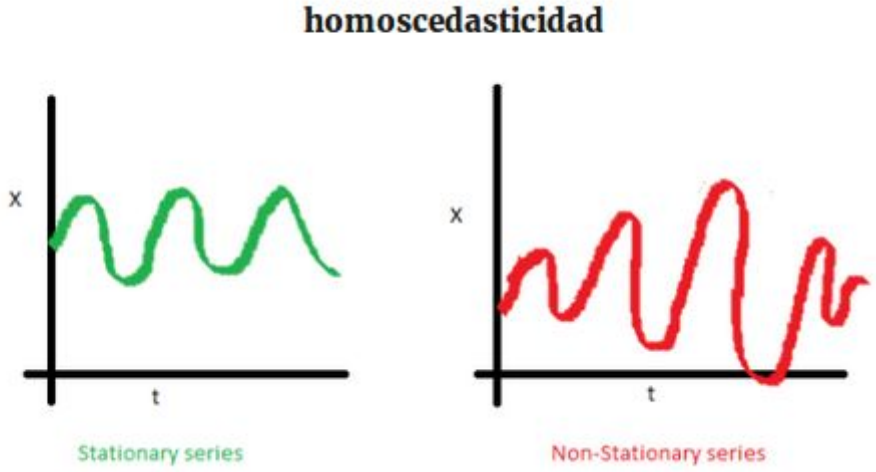
Ejemplo 4



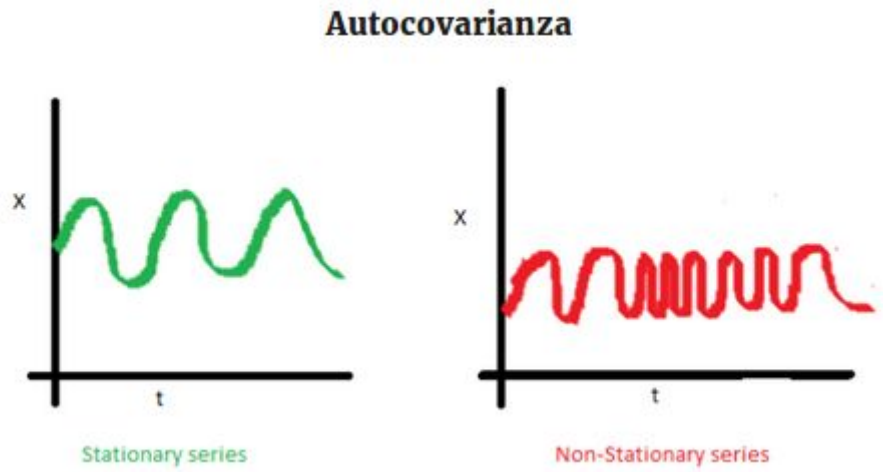
La diferencia entre los dos tipos de series temporales se puede visualizar mejor en las siguientes imágenes:



La serie de la izquierda tiene una media constante, en cambio la figura de la derecha muestra tendencia, y su media se incrementa con el paso del tiempo.



La serie de la derecha no es estacionaria, su varianza se incrementa.



En la serie de la derecha, la autocovarianza (covarianza) no es constante.

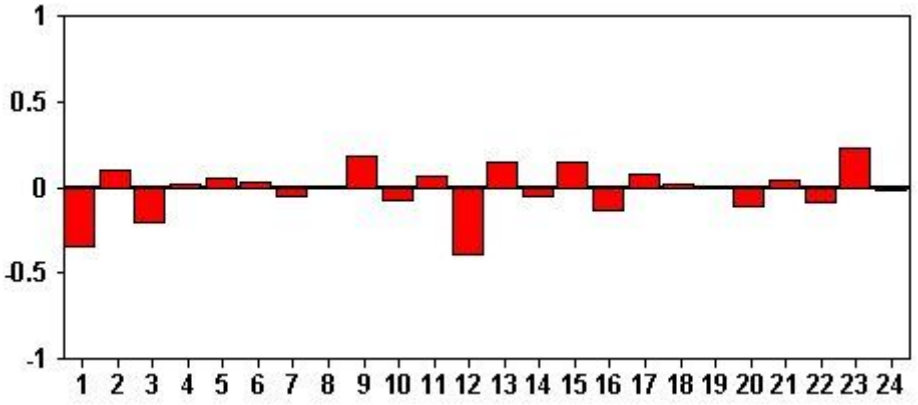
Funciones de autocorrelación y autocorrelación parcial

La autocorrelación y la autocorrelación parcial son medidas de asociación entre valores de series actuales y pasadas e indican cuáles son los valores de series pasadas más útiles para predecir valores futuros. Con estos datos podrá determinar el orden de los procesos en un modelo ARIMA. Más concretamente,

- La **autocorrelación simple (FAC)** mide la relación lineal entre las observaciones de una serie de dato Y_t, distanciados en un lapso de tiempo k. El lapso de tiempo k se conoce como *retardo* o *retraso*. Este retardo denota el periodo de tiempo entre los valores de la serie, para el cual se mide el tipo y grado de correlación de la variable considerada.
- La **autocorrelación parcial (FACP)**, es una medida asociada a la autocorrelación simple. Es la estimación de la autocorrelación simple, para el mismo retardo k, con la eliminación del efecto producido por las autocorrelaciones para retardos menores a k, las cuales están presentes en la estimación de la autocorrelación simple. La autocorrelación parcial no

considera las autocorrelaciones acumuladas para el retardo k para el que se estima.

La diferencia entre los dos tipos de autocorrelación FAC y FACP es que la autocorrelación simple brinda para un retardo k tanto la relación entre las observaciones con una diferencia de k retardos de tiempo, como la relación para retardos menores, mientras que la auto correlación parcial brinda solo la relación para la diferencia estricta en k retrasos de tiempo.



El eje *x* del gráfico de FAS indica el retardo en el que se calcula la autocorrelación; el eje *y* indica el valor de la correlación (entre -1 y 1). Por ejemplo, un trazo de unión en el retardo 1 de un gráfico de FAS indica que existe una fuerte correlación entre el valor de cada serie y el valor anterior, un trazo de unión en el retardo 2 indica que existe una fuerte correlación entre el valor de cada serie y el valor que aparece dos puntos anteriores, etc.

- Una correlación positiva indica que los valores grandes actuales se corresponden con valores grandes en el retardo especificado; una correlación negativa indica que los valores grandes actuales se corresponden con valores pequeños en el retardo especificado.
- El valor absoluto de una correlación es una medida de la fuerza de la asociación, con valores absolutos mayores que indican relaciones más fuertes.

Para saber los tipos de modelo AR y MA (explicados más adelante), es preciso observar las funciones de autocorrelación parcial y autocorrelación simple respectivamente.

Nota: Ver vídeo y documento (a partir de la página 7) para comprender mejor estos conceptos.

Estimación de la tendencia

Para estimar la tendencia supondremos que tenemos una serie no estacionaria sin componente estacional, es decir, que la serie se puede descomponer en

$$Y_t = t_t + r_t$$

Las series de los Ejemplos 2 y 4 son de este tipo. Para estimar t_t debemos realizar alguna hipótesis sobre su forma. Vamos a analizar varios casos.

Tendencia determinista

En este caso supondremos que la tendencia es una función determinística. La función más sencilla posible es una recta, es decir,

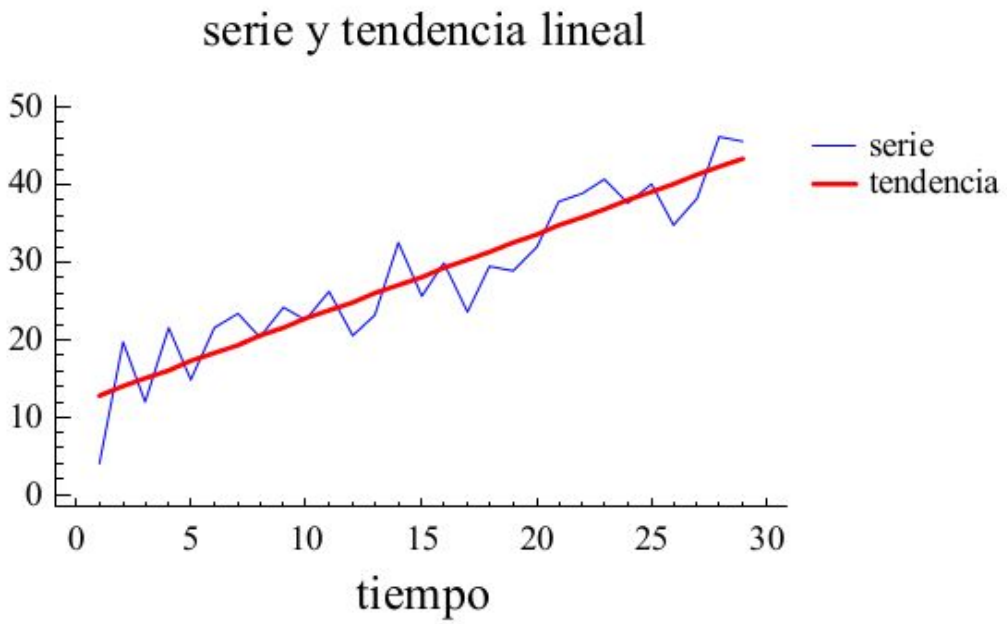
$$t_t = a + bt$$

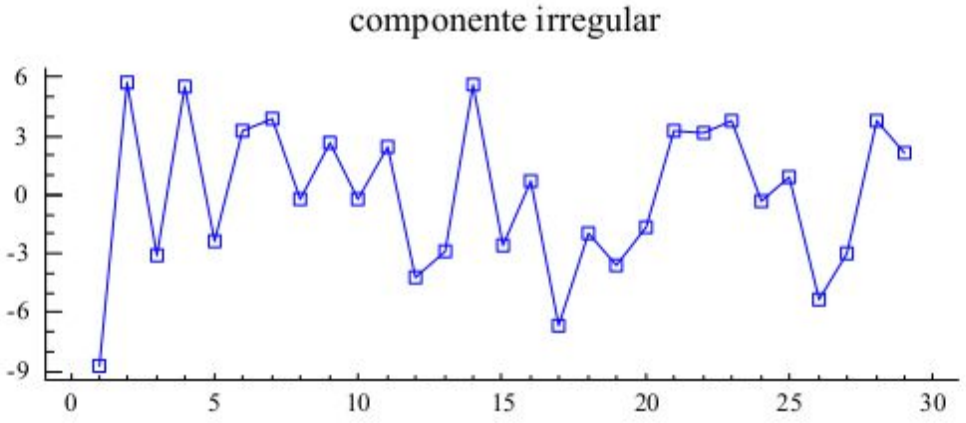
donde a y b son dos constantes a determinar. La forma de estimar estas constantes es mediante un modelo de regresión lineal entre las variables Y_t y el tiempo $t = 1, 2, 3, ...$ De esta forma, si estimamos los parámetros $\widehat{a}$ y $\widehat{b}$, entonces la componente irregular será

$$r_t = Y_t - \widehat{a} - \widehat{b}t$$

Ahora r_t sería una serie estacionaria que tendríamos que modelizar.

El Ejemplo 2 presenta una serie sin estacionalidad que presenta una tendencia que se podría expresar de forma lineal. La tendencia de la serie y la componente irregular serían, en este ejemplo,





Lo que nos queda es una serie estacionaria. En algunos casos, como en el Ejemplo 4, no es posible ajustar la tendencia mediante una recta. En estos casos, lo mejor sería ajustar la tendencia a un polinomio o a la curva que mejor se pueda ajustar. Para ello, tendríamos que ajustar una regresión no lineal. Otra opción es describir la tendencia de manera evolutiva o *diferenciar la serie*.

Tendencia evolutiva (medias móviles)

Medias móviles

Pretendemos obtener una trayectoria que refleje el movimiento a largo plazo de la serie eliminando o reduciendo en lo posible las fluctuaciones periódicas que van teniendo lugar en torno a la misma. Un modo de reducir la variabilidad de la serie, se obtiene mediante el cálculo de promedios que aglutinen y compensen valores altos y bajos. El método consiste pues, en ir agrupando sistemáticamente un número fijo k de valores de la serie y determinar para cada grupo su media.

Para obtener las medias móviles de orden k tomaremos los k primeros elementos observados en la serie y calculamos su media:

$$y_1, y_2, y_3, \cdots, y_k \quad \Rightarrow \quad y'_1 = \frac{\sum_{i=1}^{k} y_i}{k}$$

y esta media *y'1* la hacemos corresponder al periodo mediano (o medio) de los periodos $1, 2, 3, ..., k$ (notemos que si k es impar, la mediana es el instante que está en el centro, pero si k es par, será un instante comprendido entre los dos centrales).

Para obtener la segunda media móvil, consideramos los elementos:

$$y_2, y_3, \cdots, y_{k+1}$$

Y calculamos su media

$$y'_2 = \frac{\sum_{i=2}^{k+1} y_i}{k}$$

De manera análoga asignamos esta media al instante mediano correspondiente a los periodos de las observaciones que intervienen en dicha suma. Así sucesivamente se continúa el proces o hasta que intervenga en la media la última observación de la serie. Obtendremos así las $n-k+1$ medias móviles que representan el nuevo movimiento suavizado de la serie.

$$y'_1, y'_2, y'_3, \ldots, y'_{n-k+1}$$

donde de forma general:

$$y'_j = \frac{\sum_{i=j}^{k+j-1} y_i}{k}$$

Aclaraciones:

- Notemos que las medias obtenidas por un movimiento de orden k par, no estarán asignadas a los instantes registrados en la serie original. Por consiguiente, en este caso es preciso centralizar la serie de medias móviles efectuando un nuevo movimiento de orden dos (cálculo de medias móviles con $k=2$) sobre los valores y' de las medias móviles de orden k .
- Es evidente que cuanto mayor sea k más suavizadas serán las series obtenidas, es decir, reflejarán menos fluctuaciones, pero también perderemos más información, ya que si k es par, perdemos en total k observaciones; y si k es impar, perdemos en total $k-1$ observaciones.
- Hay que señalar también que este método, aunque fácil de calcular, es poco manejable si lo comparamos con las ventajas de una ecuación o función matemática.

Ejemplo: Los siguientes datos corresponden al número de accidentes de tráfico durante 10 meses registrados en una determinada zona considerada como conflictiva:

mes	1	2	3	4	5	6	7	8	9	10
nº accidentes	20	25	15	22	30	42	48	51	45	49

Si representamos la serie

vemos que su tendencia es ascendente; vamos a obtenerla mediante medias móviles de orden 3. Tomemos por tanto, los tres primeros valores de la serie y calculemos su media:

$$20,25,15 \quad \Rightarrow \quad y'_1 = \frac{20+25+15}{3} = 20$$

Quitemos ahora el primer valor y añadamos el cuarto:

$$25,15,22 \quad \Rightarrow \quad y'_2 = \frac{25+15+22}{3} = 20.67$$

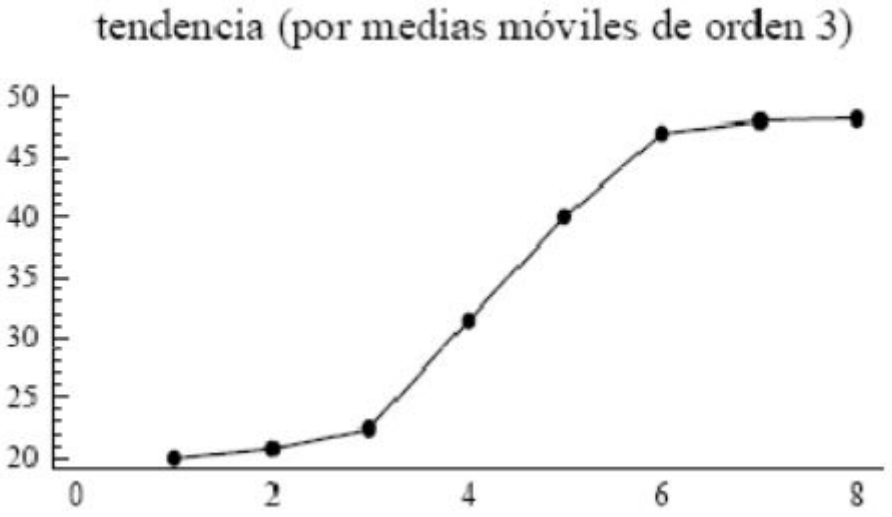
Quitando el segundo valor y añadiendo el quinto:

$$15,22,30 \quad \Rightarrow \quad y'_3 = \frac{15+22+30}{3} = 22.33$$

Repitiendo esta operación aparecen todas las medias móviles de orden 3. Asignaremos cada una de ellas al periodo mediano correspondiente:

mes	1	2	3	4	5	6	7	8	9	10
y'_i		20.00	20.67	22.33	31.33	40.00	47.00	48.00	48.33	

Observe que como k es impar, hemos perdido $k-1$ observaciones $(3-1=2)$
Representemos la nueva serie de medias móviles:



De forma similar a la obtención de las medias móviles de orden 3, calculemos las de orden 4. Son las siguientes:

20.5, 23, 27.25, 35.5, 42.75, 46.5, 48.25

que corresponderán a los periodos medianos:

2.5, 3.5, 4.5, ... , 12.5

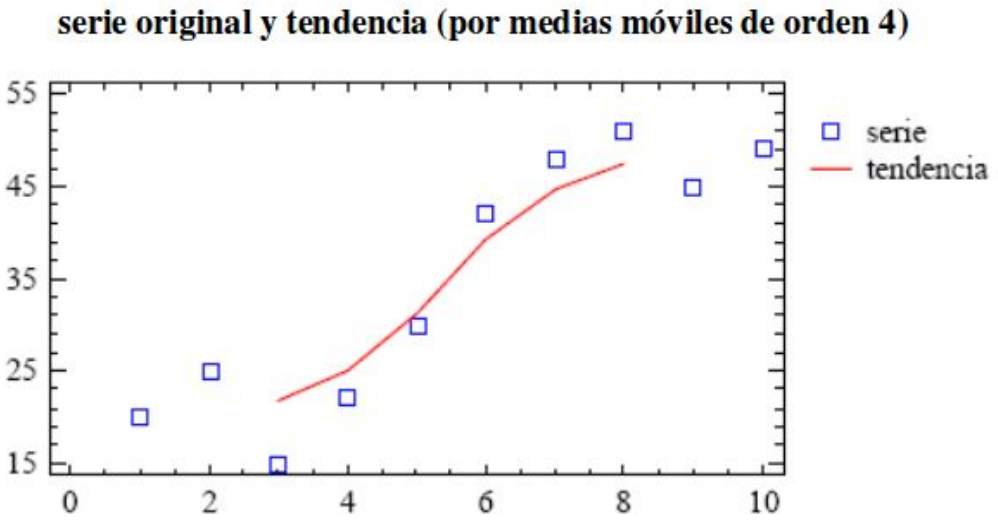
Nota: 2.5=Me(1,2,3,4); 3.5=Me(2,3,4,5), etc.

Volvemos a tomar en las medias móviles de orden 4, medias móviles de orden 2, ya que es preciso centrar la serie para que los valores obtenidos vengan referidos también a los periodos originales:

mes	1	2	3	4	5	6	7	8	9	10
y'_i			21.750	25.125	31.375	39.125	44.625	47.375		

Nota: 3=Me(2.5,3.5); 4=Me(3.5,4.5), etc.

Hemos perdido $k = 4$ observaciones. Si representamos estas medias móviles, vemos que la serie se suaviza aún más:



Definición

Se supone que la tendencia es una función que evoluciona lentamente y que puede aproximarse en intervalos muy cortos (por ejemplo de 3 ó 5 datos) por una función temporal simple. En general se supone una recta, pero ahora sus coeficientes van cambiando suavemente en el tiempo.

Suponemos que la representación de la tendencia por una recta es válida para tres períodos consecutivos de tiempo, $t-1$, t , $t+1$, y representamos las tendencias en los tres periodos consecutivos de la siguiente manera:

$$T_{t-1} = T_t - crecimiento$$
$$T_t$$
$$T_{t+1} = T_t + crecimiento$$

Si hacemos la media de tres observaciones consecutivas

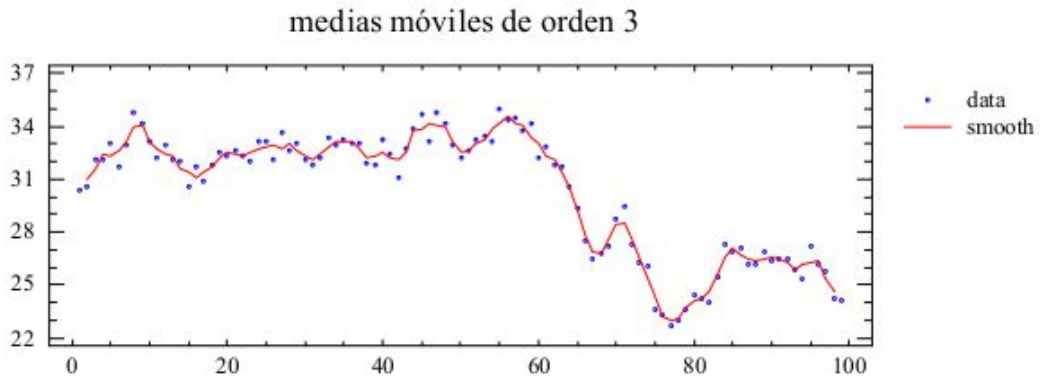
$$m_t = \frac{x_{t-1} + x_t + x_{t+1}}{3}$$

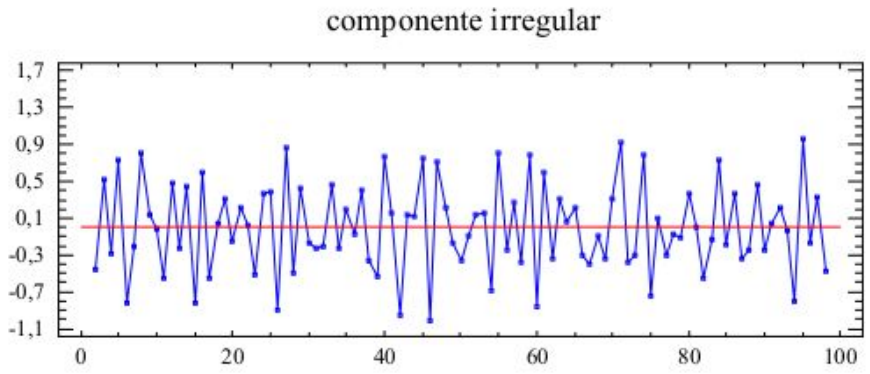
entonces

$$m_t = T_t + \frac{I_{t-1} + I_t + I_{t+1}}{3}$$

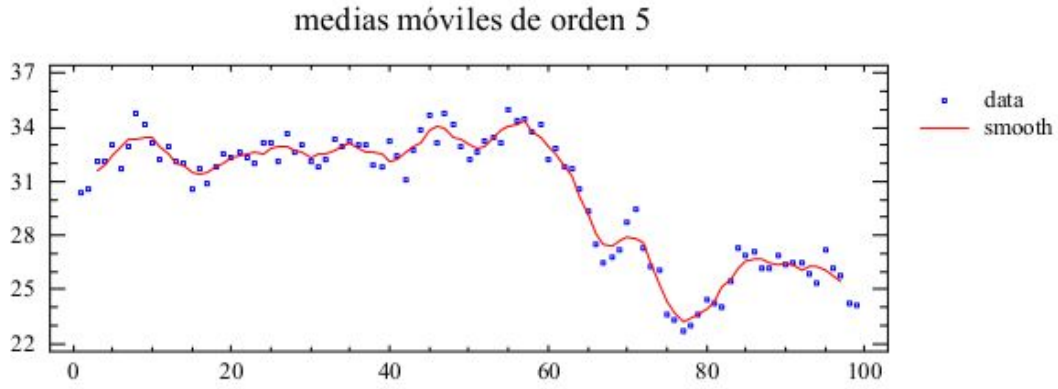
y como la componente irregular tiene media cero, la media de los tres valores del componente irregular se puede suponer que es despreciable frente a la tendencia, y m_t representa la tendencia en ese instante. Esta operación se denomina **media móvil de orden tres**. Se observa que realizando esta operación se pierde la primera observación y la última. Si calculamos las medias móviles de orden 5, perderemos las dos primeras observaciones y las dos últimas.

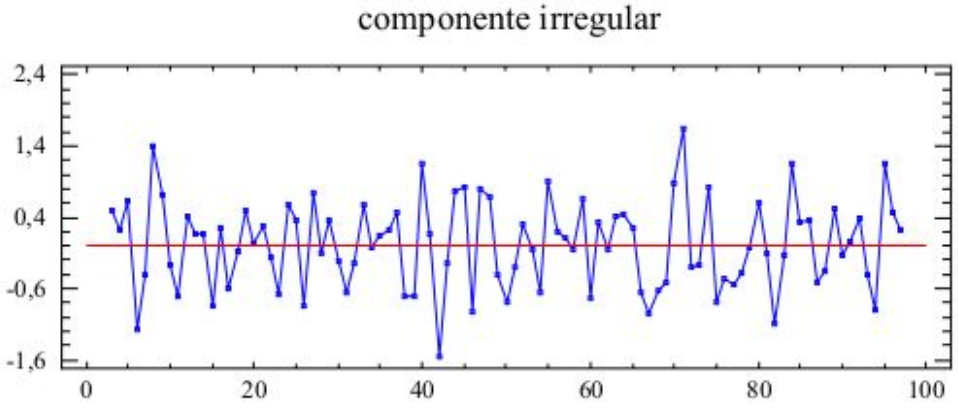
Aplicando este método a la serie del Ejemplo 4, las medias móviles de orden con la correspondiente componente irregular son:





mientras que para el ajuste a una media móvil de orden 5 queda:





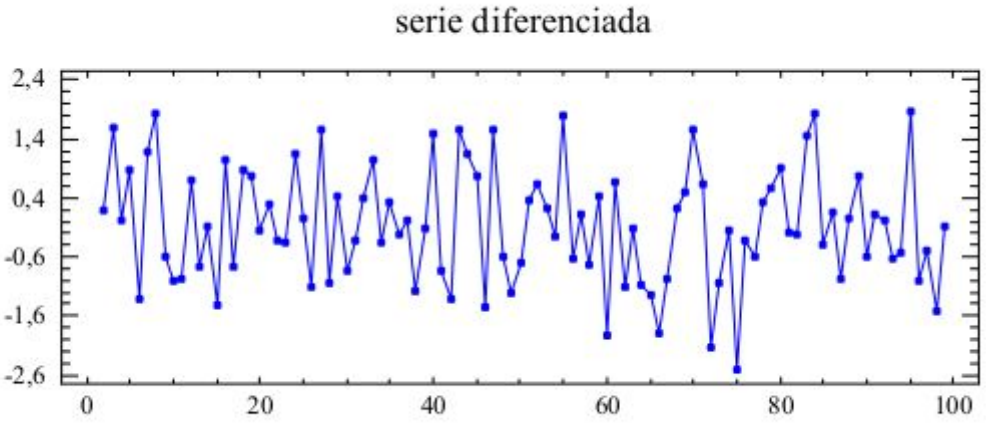
Diferencia de la serie

Un tercer método más general para eliminar la tendencia consiste en suponer que la tendencia evoluciona lentamente en el tiempo, de manera que en el instante t la tendencia debe estar próxima a la tendencia en el instante t − 1. De esta forma, si restamos a cada valor de la serie el valor anterior, la serie resultante estará aproximadamente libre de tendencia. Esta operación se denomina **diferenciación de la serie** y consiste en pasar de la serie original x_t a la serie y_t mediante:

$$y_t = x_t - x_{t-1}$$

De este modo, la serie diferenciada resulta ser estacionaria.

La



serie del Ejemplo 4 diferenciada queda como

Estimación de la estacionalidad

Vamos a eliminar de la serie la componente estacional, es decir, se desestacionaliza la serie mediante los últimos 6 años de la serie del Ejemplo 3. Esta serie presenta una estacionalidad mensual, de modo que se colocan los datos en una tabla de doble entrada:

	90	91	92	93	94	95	medias	coef. est.
Enero	5.49	5.65	5.75	5.83	5.89	6.03	5.77	-0.14
Febrero	5.45	5.62	5.71	5.76	5.83	5.97	5.72	-0.19
Marzo	5.59	5.76	5.87	5.89	6.01	6.04	5.86	-0.05
Abril	5.59	5.75	5.85	5.85	5.98	6.13	5.86	-0.05
Mayo	5.60	5.76	5.87	5.89	6.04	6.16	5.89	-0.02
Junio	5.75	5.92	6.05	6.08	6.16	6.28	6.04	0.13
Julio	5.90	6.02	6.14	6.20	6.31	6.43	6.17	0.26
Agosto	5.85	6.00	6.15	6.22	6.33	6.41	6.16	0.25
Septiembre	5.74	5.87	6.00	6.00	6.14	6.23	6.00	0.09
Octubre	5.61	5.72	5.85	5.88	6.01	6.13	5.87	-0.04
Noviembre	5.47	5.60	5.72	5.74	5.89	5.97	5.73	-0.18
Diciembre	5.63	5.72	5.82	5.82	6.00	6.07	5.84	-0.07

En este ejemplo hay efecto estacional mensual y, por tanto, existen 12 coeficientes estacionales, uno para cada mes del año. Para estimarlos, se calcula primero la media de las observaciones para cada mes, $M_1, ..., M_{12}$, y el coeficiente estacional resulta ser:

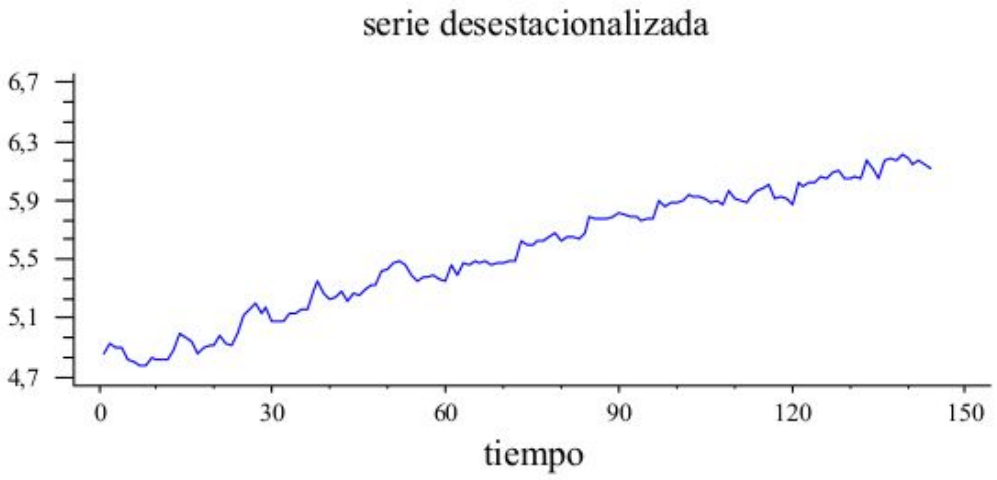
$$S_i = M_i - M \text{ para } i = 1, ..., 12$$

donde M es la media total de las observaciones. Así se puede observar que los meses con observaciones más pequeñas (por debajo de la media general) tienen coeficientes estacionales negativos, mientras que los meses con observaciones mayores tienen coeficientes estacionales positivos.

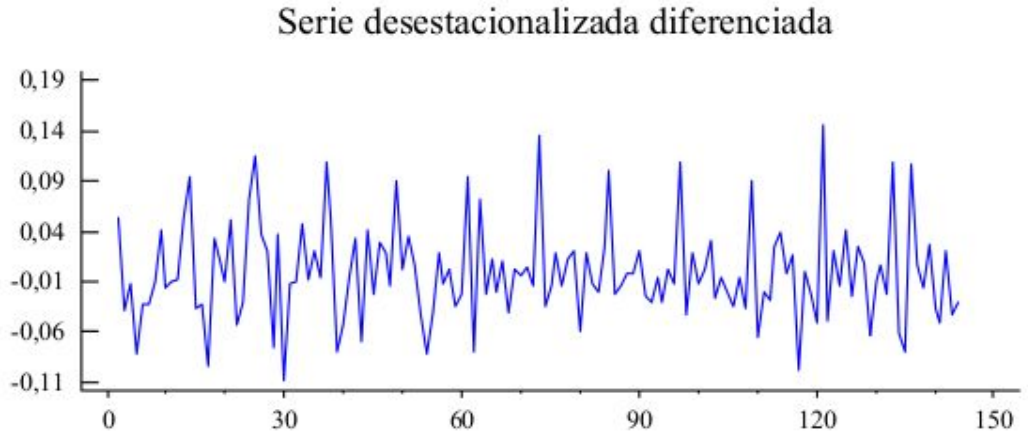
En el ejemplo anterior, la media total de las observaciones es $M = 5,91$, y así $S_1 = M_1 - M = 5,77 - 5,91 = -0,14$, . . .

Evidentemente, la suma de los coeficientes estacionales tiene que ser cero. Las temporadas más bajas son las correspondientes a los meses de febrero y noviembre, y las más altas las de los meses de julio y agosto.

Se denomina *serie desestacionalizada* a una serie donde se ha eliminado el efecto de cada mes y que se obtiene restando al valor de cada mes el coeficiente estacional de dicho mes. Teniendo una serie desestacionalizada, podemos observar el comportamiento general de una serie cronológica sin tener en cuenta la componente estacional. En nuestro ejemplo, con todos los datos, la serie desestacionalizada es:



Comprobamos que esta serie ya no tiene estacionalidad, pero todavía tiene tendencia. Para eliminar la tendencia procedemos a aplicar los procedimientos de la sección anterior a la serie desestacionalizada. Por ejemplo, diferenciando esta serie obtenemos:



Sin embargo esta serie vuelve a ser estacional, esto es, el problema es que al diferenciar ha aparecido otra vez un efecto estacional. Para explicar esto, vamos a analizar algunas observaciones de la serie $x_1, x_2, ..., x_{13}, x_{14} ... x_1$.

Al desestacionalizar la serie, hemos realizado la siguiente transformación:

$$y_1 = x_1 - E_1$$
$$y_2 = x_2 - E_2$$
$$...$$

$$y_{13} = x_{13} - E_1$$
$$y_{14} = x_{14} - E_2$$
$$\ldots$$

Ahora, al diferenciar la serie obtenemos:

$$z_2 = y_2 - y_1 = x_2 - x_1 - E_2 + E_1 = x_2 - x_1 - (E_2 - E_1)$$

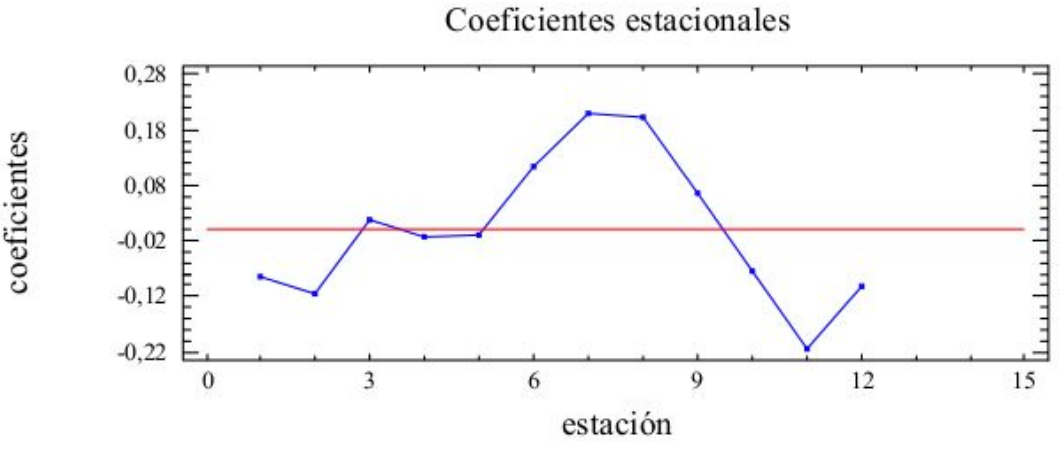
.....

$$z_{14} = y_{14} - y_{13} = x_{14} - x_{13} - E_2 + E_1 = x_{14} - x_{13} - (E_2 - E_1)$$

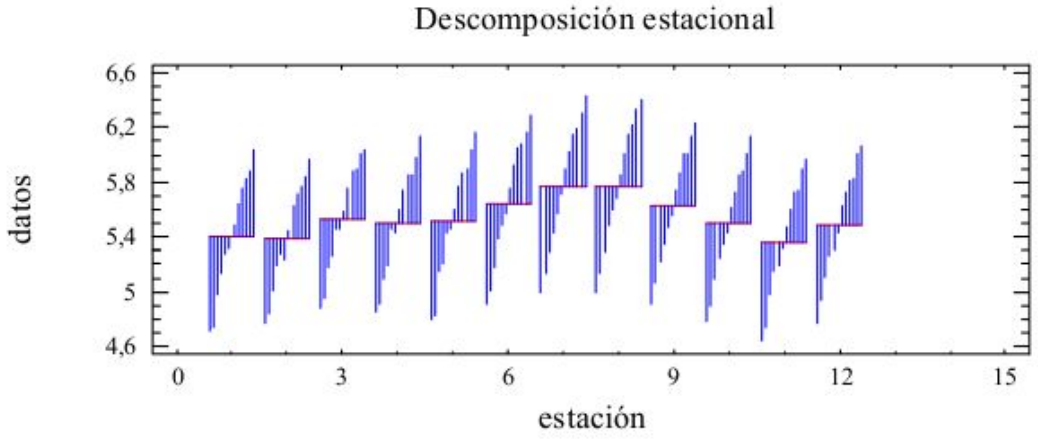
es decir, aparece en la serie un coeficiente estacional: $(E_2 - E_1)$.

Para solucionar este problema, se puede eliminar la tendencia de la serie desestacionalizada mediante los métodos descritos en las secciones anteriores. Otra solución, que es la más apropiada, es eliminar primero la tendencia de la serie y en segundo lugar desestacionalizar la serie.

Para comprobar que la serie tiene componente estacional, se puede comprobar a partir del gráfico temporal de la serie. El siguiente gráfico muestra los coeficientes estacionales de la serie del Ejemplo 3. Como se puede comprobar éstos son mayores para los meses de julio y agosto, y son menores para los meses de diciembre y febrero, lo cual indica que hay estacionalidad.

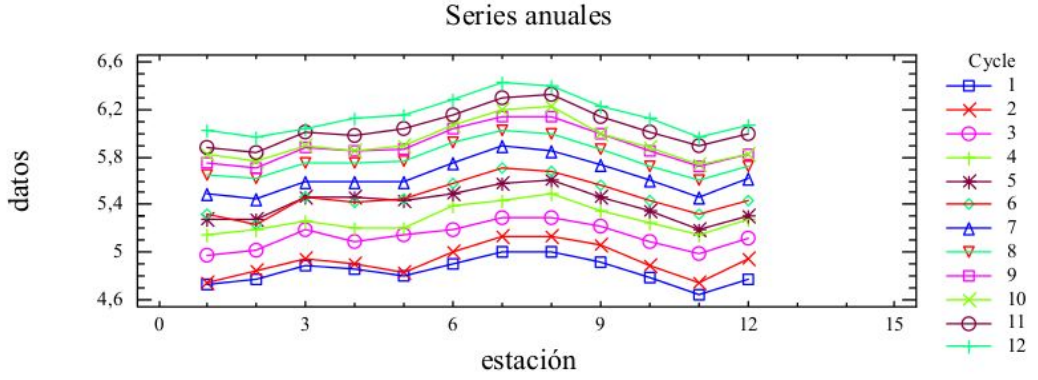


El siguiente gráfico muestra la descomposición estacional de la serie. Las líneas horizontales para cada mes muestran la media de la serie para cada uno de los meses (estaciones). Las líneas verticales que salen de cada línea horizontal indican, en cada mes, cómo varía la serie en los diferentes años. Podemos ver que en cada mes, el patrón durante los 12 años es el mismo, lo cual indica claramente que la serie tiene una tendencia creciente.



El siguiente gráfico muestra la serie año a año. La variación de la serie en los distintos meses es muy parecida todos los años, lo cual indica que hay estacionalidad. Además, las gráficas de los doce años están en orden creciente: la del primer año está por debajo de la del segundo año, la del segundo año por debajo de la del tercer año, y así sucesivamente.Esto indica que la serie tiene una tendencia creciente.

A veces, una simple inspección del gráfico temporal es suficiente para



observar las principales características de la serie, pero en caso de duda, algunos de los gráficos anteriores pueden ser de gran utilidad.

Las transformaciones suelen ser útiles para estabilizar una serie antes de estimar modelos y se aplican sobre las componentes de las series temporales. **Esto es especialmente importante para modelos ARIMA, que necesitan que las series sean estacionarias antes de estimar los modelos**.

Aunque la mayoría de las series interesantes no son estacionarias, ARIMA es eficaz siempre y cuando la serie se pueda convertir en estacionaria mediante la aplicación de transformaciones tales como el logaritmo natural, la diferenciación o la diferenciación estacional.

Nota: La estacionalidad o variación estacional de una serie temporal es la variación periódica y predecible de la misma con un periodo inferior o igual a un año.

Definition

An *ARIMA model* is a class of statistical models for analyzing and forecasting time series data. It explicitly caters to a suite of standard structures in time series data, and as such provides a simple yet powerful method for making skillful time series forecasts.

Box y Jenkins han desarrollado modelos estadísticos para series temporales que tienen en cuenta la dependencia existente entre los datos, esto es, cada observación en un momento dado es modelada en función de los valores anteriores. Los análisis se basan en un modelo explícito. Los modelos se conocen con el nombre genérico de *ARIMA*

ARIMA is an acronym that stands for **AutoRegressive Integrated Moving Average**. It is a generalization of the simpler AutoRegressive Moving Average and adds the notion of integration.
This acronym is descriptive, capturing the key aspects of the model itself. Briefly, they are:

- **AR**: *Autoregression*. A model that uses the dependent relationship between an observation and some number of lagged observations.
- **I**: *Integrated*. The use of differencing of raw observations (e.g. subtracting an observation from an observation at the previous time step) in order to make the time series stationary.
- **MA**: *Moving Average*. A model that uses the dependency between an observation and a residual error from a moving average model applied to lagged observations.

Each of these components are explicitly specified in the model as a parameter. A standard notation is used of *ARIMA(p,d,q)* where the parameters are substituted with integer values to quickly indicate the specific ARIMA model being used.
The parameters of the ARIMA model are defined as follows:

- **Autorregresivo (p)**. Es el número de órdenes autorregresivos del modelo. Los órdenes autorregresivos especifican los valores previos de la serie utilizados para predecir los valores actuales. Por ejemplo, un orden autorregresivo igual a 2 especifica que se van a utilizar los valores de la serie correspondientes a dos períodos de tiempo del pasado para predecir el valor actual.
- **Diferencia (d)**. Especifica el orden de diferenciación aplicado a la serie antes de estimar los modelos. La diferenciación es necesaria si hay tendencias (las series con tendencias suelen ser no estacionarias y el modelado de ARIMA asume la estacionariedad) y se utiliza para eliminar su efecto. El orden de diferenciación se corresponde con el grado de la tendencia de la serie (la diferenciación de primer orden representa las tendencias lineales, la diferenciación de segundo orden representa las tendencias cuadráticas, etc.).
- **Media móvil (q)**. Es el número de órdenes de media móvil presentes en el modelo. Los órdenes de media móvil especifican el modo en que se utilizan las desviaciones de la media de la serie para los valores previos con el fin de predecir los valores actuales. Por ejemplo, los órdenes de media móvil de 1 y 2 especifican que las desviaciones del valor medio de la serie de cada uno de los dos últimos períodos de tiempo se tienen en cuenta al predecir los valores actuales de la serie.

A linear regression model is constructed including the specified number and type of terms, and the data is prepared by a degree of differencing in order to make it stationary, i.e. to remove trend and seasonal structures that negatively affect the regression model.

A value of 0 can be used for a parameter, which indicates to not use that element of the model. This way, the ARIMA model can be configured to perform the function of an ARMA model, and even a simple AR, I, or MA model.

Adopting an ARIMA model for a time series assumes that the underlying process that generated the observations is an ARIMA process. This may seem obvious, but helps to motivate the need to confirm the assumptions of the model in the raw observations and in the residual errors of forecasts from the model.

Fases

El modelo ARIMA permite describir un valor como una función lineal de datos anteriores y errores debidos al azar, además, puede incluir un componente cíclico o estacional. Es decir, debe contener todos los elementos necesarios para describir el fenómeno. Box y Jenkins recomiendan como mínimo 50 observaciones en la serie temporal.

La metodología de Box y Jenkins se resume en cuatro fases:

- La **primera fase** consiste en **identificar el posible modelo ARIMA** que sigue la serie, lo que requiere:

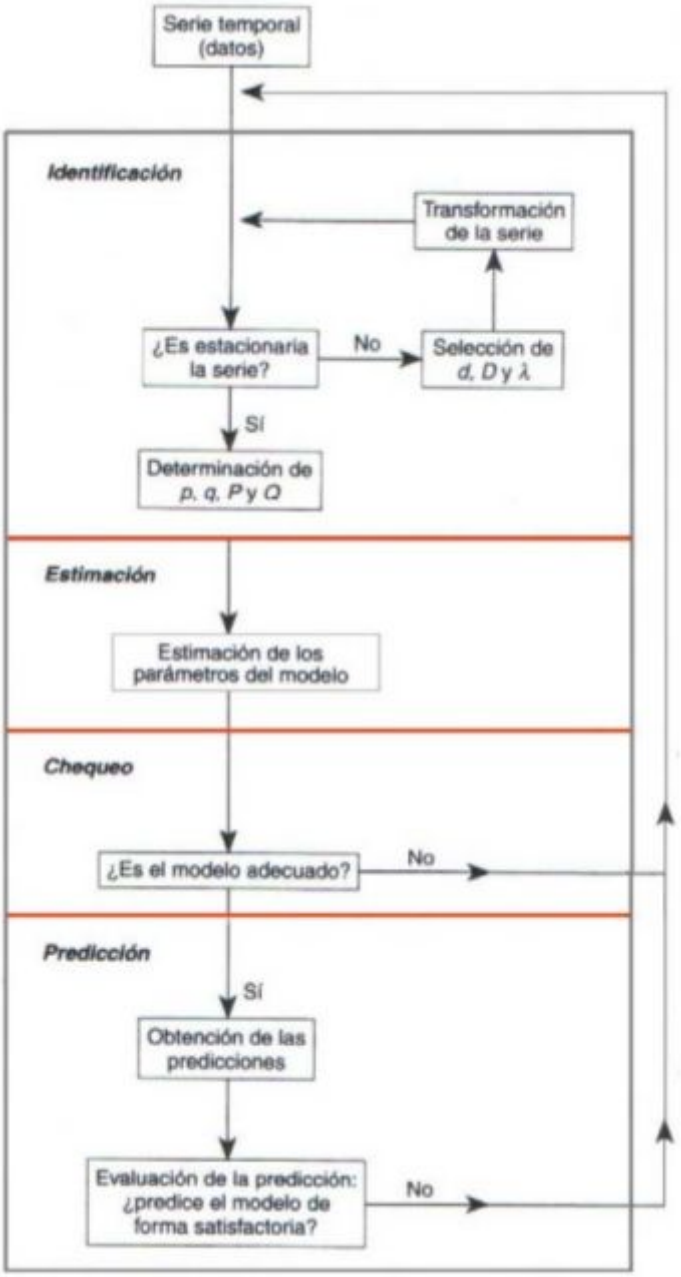
  - ❖ Decidir qué transformaciones aplicar para convertir la serie observada en una serie estacionaria.
  - ❖ Determinar un modelo **ARMA** para la serie estacionaria, es decir, los órdenes p y q de su estructura autorregresiva y de media móvil.

Por lo tanto, para realizar la primera fase, hay que determinar si la serie de tiempo es estacionaria y si hay alguna estacionalidad significativa que necesite ser modelada.

- La **segunda fase**: seleccionado provisionalmente un modelo para la serie estacionaria, se pasa a la segunda etapa de estimación, donde los **parámetros AR y MA del modelo se estiman** por máxima verosimilitud y se obtienen sus errores estándar y los residuos del modelo.

- La **tercera fase** es el diagnóstico, donde se comprueba que los residuos no tienen estructura de dependencia y siguen un proceso de ruido blanco. Si los residuos muestran estructura se modifica el modelo para incorporarla y se repiten las etapas anteriores hasta obtener un modelo adecuado.

- La **cuarta fase** es la predicción, una vez que se ha obtenido un modelo adecuado se realizan predicciones con el mismo.



Pasos a seguir para el análisis de datos

1. **Recogida de datos**: Es conveniente disponer de 50 o más datos, y en el caso de series mensuales, trabajar entre seis y diez años completos.

2. **Representación gráfica**: Es de gran utilidad disponer de un gráfico de la serie para decidir sobre la estacionariedad. En ocasiones, se utilizan medias y desviaciones típicas por subperiodo para juzgar sobre la estacionariedad de la serie.

3. **Transformación previa de la serie**: Cuando la serie no es estacionaria en varianza se requiere una transformación logarítmica. No obstante, la transformación logarítmica es muy frecuente incluso en series con dispersión relativamente constante en el tiempo. Una práctica habitual es ensayar con la serie original y en logaritmos y comprobar resultados.

4. **Eliminación de la tendencia**: La observación del gráfico de la serie indica la existencia o no de tendencia. Una tendencia lineal será corregida tomando primeras diferencias, que será el caso más frecuente. Una tendencia no lineal suele llevar en la práctica al uso de dos diferencias como mucho.

5. **Identificación del modelo**: Consiste en determinar el tipo de modelo más adecuado, esto es, el orden de los procesos autorregresivos y de medias móviles de las componentes regular y estacional. Técnicamente esta decisión se toma en base a las funciones de autocorrelación (FAC) y autocorrelación parcial (FAC parcial), tanto en la parte regular como estacional. Es habitual terminar eligiendo entre los procesos más simples AR(1), AR(2), MA(1), MA(2) y ARMA(1,1), tanto en la parte regular como estacional. En caso de duda pueden seleccionarse varios modelos alternativos que serán estimados y contrastados posteriormente, para definir finalmente el modelo adoptado.

6. **Estimación de los coeficientes del modelo**: Decidido el modelo, se procede a la estimación de sus parámetros, dado que se trata de un procedimiento iterativo de cálculo, pueden sugerirse valores iniciales.

7. **Contraste de validez del modelo:** Se utilizan distintos procedimientos para valorar el modelo o modelos inicialmente seleccionados: contraste de significación de parámetros, covarianzas entre estimadores, coeficiente de correlación, suma de cuadrados de errores, etc.

8. **Análisis detallado de los errores**: Se tendrán en cuenta las diferencias históricas entre valores reales y estimados por el modelo para su valoración final. Hay que verificar un comportamiento no sistemático de los mismos, así como analizar la posible existencia de errores especialmente significativos.

9. **Selección del modelo**: En base a los resultados de pasos anteriores, se decide sobre el modelo adoptado.

10. **Predicción**: El modelo seleccionado se utilizará como fórmula inicial de predicción.

Identificación práctica del modelo

Identificar un modelo significa utilizar los datos recogidos, así como cualquier información de cómo se genera la serie temporal objeto de estudio, para sugerir un conjunto reducido de posibles modelos, que tengan muchas posibilidades de ajustarse a los datos. Ante una serie temporal empírica, se deben encontrar los valores **(p, d, q**) más apropiados.

- Si la serie temporal presenta una tendencia, lo primero que debe de hacerse es convertirla en estacionaria mediante una diferenciación de orden **d**. Una vez diferenciada la serie, una buena estrategia consiste en comparar los correlogramas de la función de autocorrelación (ACF) y la función de autocorrelación parcial (ACFP), proceso que suele ofrecer una orientación para la formulación del modelo orientativo.

- Los procesos autorregresivos presentan función de autocorrelación parcial (ACFP) con un número finito de valores distinto de cero. Un proceso AR(p) tiene los primeros **p** términos de la función de autocorrelación parcial distintos de cero y los demás son nulos. Esta afirmación es muy fuerte, y en la práctica se considera que una muestra dada proviene de un proceso autorregresivo de orden p si los términos de la función de autocorrelación parcial son casi cero a partir del que ocupa el lugar p.

- Los procesos de medias móviles presentan función de autocorrelación con un número finito de valores distintos de cero. Un proceso MA(q) tiene los primeros q términos de la función de autocorrelación distintos de cero y los demás son nulos**.**

Las dos propiedades descritas son muy importantes con vistas a la identificación de un proceso mediante el análisis de las funciones de autocorrelación y autocorrelación parcial.

El resumen de los pasos de identificación de un modelo de series temporales:

1. **Decidir si** X_t **necesita ser transformada** para eliminar la no estacionariedad en media p en la no estacionariedad en varianza (*heteroscedasticidad*). Puede ser conveniente utilizar logaritmos de la serie o aplicar la transformación de Box-Cox.
2. **Determinación del grado d de diferenciación adecuado**. En general, la falta de estacionariedad se manifiesta en que los coeficientes de la función de autocorrelación estimada tienden a

decrecer muy lentamente. La pregunta es, ¿cuán lentamente ha de ser el decrecimiento de los coeficientes de la función de autocorrelación parcial (ACFP) para que el proceso sea estacionario?. En general, solo ocasionalmente los datos económicos del correlograma dejarán de decrecer tras las primeras diferencias, y en este caso serían necesarias segundas diferencias. Una diferenciación superflua solo sirve para alterar el esquema de autocorrelación evidente en una serie estacionaria y complicarlo innecesariamente.

3. **Decidir los valores de (p, q)**, y si existe una componente estacional, decidir los órdenes de los operadores estacionales (P, Q). Para este apartado se utilizan las funciones de autocorrelación (ACF) y autocorrelación parcial (ACFP) según el siguiente cuadro:

Proceso	Función de autocorrelación (ACF)	Función de autocorrelación parcial (ACFP)
MA(q)	Solo los **q** primeros coeficientes son significativos. El resto se anulan bruscamente (coef. 0 para **retardo >q**)	Decrecimiento rápido exponencial atenuado u ondas sinusoidales.
AR(p)	Decrecimiento rápido exponencial atenuado u ondas sinusoidales.	Solo los **p** primeros coeficientes son significativos. El resto se anulan bruscamente (coef. 0 para retardo >q)
ARIMA(p, d, q)	Comportamiento irregular en los retardos (1, … , q) con **q** picos. Decrecimiento para retardos posteriores a **q**.	Decrece (aproximadamente con exponenciales atenuados y ondas sinusoidales). No cero pronto.

Detección práctica de la estacionariedad

Para detectar rápidamente la estacionariedad se pueden calcular la sucesión de medias y varianzas por años, si se obtienen variaciones significativas crecientes y decrecientes a lo largo de los años, indica que no hay estacionariedad. Este resultado conduce a tomar logaritmos y diferenciar la serie original con el objetivo de atenuar la falta de estacionariedad en media y varianza.

Otro método, si los coeficientes de la ACF no decaen rápidamente hay un indicio claro de falta de estacionariedad en media, lo que llevaría a tomar primeras diferencias en la serie original. Si hay duda sobre diferenciar o no, o sobre cuántas veces hay que diferenciar, se calcula la varianza de la serie original y de la serie sometida a diferentes diferenciaciones, tomando como diferenciación adecuada

aquella para la que la varianza es mínima. El método es tanto más adecuado cuanto mayor sea la diferencia entre las varianzas anteriores. La *sobrediferenciación* suele evitarse observando si en la parte de medias móviles alguna raíz es próxima a la unidad.

La estacionariedad, así como la estacionalidad, también puede detectarse a través de las funciones de autocorrelación (ACF) y autocorrelación parcial estimadas (ACFP).

Modelos autorregresivos AR(p)

Un *modelo autorregresivo AR* describe una clase particular de proceso en que las observaciones en un momento dado son predecibles a partir de las observaciones previas del proceso más un término de error. El caso más simple es el **ARIMA(1,0,0)** o **AR(1)** o de primer orden, cuya expresión matemática es:

$$AR(1) \equiv X_t = \phi_1 X_{t-1} + a_t$$

El proceso autorregresivo de orden p, representado por **ARIMA(p,0,0)** o simplemente por **AR(p)**:

$$AR(p) \equiv X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + a_t$$

que puede ponerse, mediante el operador de cambio retroactivo B, en la forma:

$$(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p) X_t = a_t$$
$$B^k(X_t) = X_{t-k}$$

- Un proceso autorregresivo AR(p) es **estacionario** si las raíces del polinomio en B dado por: $(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p)$ caen fuera del círculo unidad. Esta condición es equivalente a que las raíces de la ecuación: $x^p - \phi_1 x^{p-1} - \phi_2 x^{p-2} - ... - \phi_{p-1} x - \phi_p = 0$ sean todas inferiores a uno en módulo.
- Un proceso autorregresivo siempre es **invertible**.

Modelo de medias móviles MA(q)

Un modelo de *medias móviles MA* describe una serie temporal estacionaria. En este modelo el valor actual puede predecirse a partir de la componente aleatoria

de este momento y, en menor medida, de los impulsos aleatorias anteriores. El modelo **ARIMA(0,0,1)**, también denotado por **MA(1)**, viene dado por la expresión:

$$X_t = a_t - v_1 a_{t-1}$$

El proceso de medias móviles de orden q, representado por **ARIMA(0,0,q)** o también por **Ma(q)**, viene dado por la expresión:

$$X_t = a_t - v_1 a_{t-1} - v_2 a_{t-2} - ... - v_q a_{t-q}$$

que puede ponerse, mediante el operador de cambio retroactivo B, en la forma:

$$X_t = (1 - v_1 B - v_2 B^2 - ... - v_q B^q) a_t$$

- Un proceso de medias móviles es siempre estacionario.
- Un proceso de medias móviles Ma(q) es **invertible** si las raíces del polinomio en B definido por $(1 - v_1 B - v_2 B^2 - ... - v_q B^q)$ caen fuera del círculo unidad. Esta condición es equivalente a que las raíces de la ecuación $x^q - \phi_1 x^{q-1} - \phi_2 x^{q-2} - ... - \phi_{q-1} x - \phi_q = 0$ sean todas inferiores a uno en módulo.

Modelos ARMA(p,q)

Una extensión natural de los modelos **AR(p)** y **MA(q)** es un tipo de modelos que incluyen tanto términos autorregresivos como de medias móviles y se definen como **ARIMA(p, 0, q)**. Se representan por la ecuación:

$$X_t = \phi_1 X_{t-1} + \phi_2 X_{t-2} + ... + \phi_p X_{t-p} + a_t - v_1 a_{t-1} - v_2 a_{t-2} - ... - v_q a_{t-q}$$

que puede ponerse de la forma:

$$X_t - \phi_1 X_{t-1} - \phi_2 X_{t-2} - ... - \phi_p X_{t-p} = a_t - v_1 a_{t-1} - v_2 a_{t-2} - ... - v_q a_{t-q}$$

es decir,

$$(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p) X_t = (1 - v_1 B - v_2 B^2 - ... - v_q B^q) a_t$$

El proceso ARMA(p, q) es **estacionario** si lo es su componente autorregresiva, y es invertible si lo es su componente de medias móviles.

- Un modelo ARMA(p, q) es **estacionario** si las raíces del polinomio definido por $(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p)$ caen fuera del círculo unidad. Esta condición es equivalente a que las raíces de la ecuación: $x^p - \phi_1 x^{p-1} - \phi_2 x^{p-2} - ... - \phi_{p-1} x - \phi_p = 0$ sean todas inferiores a uno en módulo.

- Un modelo ARMA(p, q) es **invertible** si las raíces del polinomio en B definido mediante $(1 - v_1 B - v_2 B^2 - ... - v_q B^q)$ caen fuera del círculo unidad. Esta condición es equivalente a que las raíces de la ecuación:

  $x^q - \phi_1 x^{q-1} - \phi_2 x^{q-2} - ... - \phi_{q-1} x - \phi_q = 0$ sean todas inferiores a uno en módulo.

Modelos ARIMA(p,q)

Un modelo **ARIMA(0, d, 0)** es una serie temporal que se convierte en ruido blanco (proceso puramente aleatorio) después de ser diferenciada d veces. El modelo (0, d, 0) se expresa mediante: $(1 - B)^d X_t = a_t$

El modelo general ARIMA(p, d, q) denominado **proceso autorregresivo integrado de medias móviles de orden p, d, q**, toma la expresión:

$$(1 - \phi_1 B - \phi_2 B^2 - ... - \phi_p B^p)\ (1 - B)^d X_t = \\ = (1 - v_1 B - v_2 B^2 - ... - v_q B^q) a_t$$

Un modelo ARIMA(p, d, q) permite describir una serie de observaciones después de que hayan sido diferenciadas **d** veces, a fin de extraer las posibles fuentes de no estacionariedad. Esta fórmula se puede aplicar a cualquier modelo. Si hay alguna componente p, d, q, igual a cero, se elimina el término correspondiente de la fórmula general.

Los modelos cíclicos o estacionales son aquellos que se caracterizan por oscilaciones cíclicas, también denominadas **variaciones estacionales**. Las variaciones cíclicas a veces se superponen a una tendencia secular.

Las series con tendencia secular y variaciones cíclicas pueden representarse mediante los modelos **ARIMA(p, d, q)(P, D, Q)S**. El primer paréntesis **(p, d, q)** se refiere a la tendencia secular o parte regular de la serie y el segundo paréntesis **(P, D, Q)** se refiere a las variaciones estacionales, o parte cíclica de la serie temporal (with *p* = non-seasonal AR order, *d* = non-seasonal differencing, *q* = non-seasonal

MA order, *P* = seasonal AR order, *D* = seasonal differencing, *Q* = seasonal MA order, and *S* = time span of repeating seasonal pattern). (Ver enlace)

Ejemplo en Python

- Crear un modelo ARIMA para series temporales
- Configurar los parámetros de ARIMA (Parameter Tuning)
- Introducción a la autocorrelación y a la autocorrelación parcial
- Cómo realizar una búsqueda exhaustiva de los hiperparámetros del modelo ARIMA

K-Nearest Neighbors (KNN)

Introducción

El algoritmo de los **k vecinos más cercanos** (**k-NN Nearest Neighbour**) es un sistema de clasificación supervisado basado en criterios de vecindad. Recordemos que los sistemas de **clasificación supervisados** son aquellos en los que, a partir de un conjunto de ejemplos clasificados (**conjunto de entrenamiento**), intentamos asignar una clasificación a un segundo conjunto de ejemplos. En particular, la idea básica sobre la que se fundamenta este paradigma es que un nuevo caso se va a clasificar en la clase más frecuente a la que pertenezcan sus k vecinos más cercanos. El paradigma se fundamenta, por tanto, en una idea muy simple e intuitiva, lo que unido a su fácil implementación hace que sea un paradigma clasificatorio muy extendido.

El algoritmo K-NN básico

La notación a utilizar es la siguiente:

		X_1	$\ldots$	X_j	$\ldots$	X_n	C
$(\mathbf{x}_1, c_1)$	1	x_{11}	$\ldots$	x_{1j}	$\ldots$	x_{1n}	c_1
	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
$(\mathbf{x}_i, c_i)$	i	x_{i1}	$\ldots$	x_{ij}	$\ldots$	x_{in}	c_i
	$\vdots$	$\vdots$		$\vdots$		$\vdots$	$\vdots$
$(\mathbf{x}_N, c_N)$	N	x_{N1}	$\ldots$	x_{Nj}	$\ldots$	x_{Nn}	c_N
$\mathbf{x}$	$N+1$	$x_{N+1,1}$	$\ldots$	$x_{N+1,j}$	$\ldots$	$X_{N+1,n}$	?

- D indica un fichero de N casos, cada uno de los cuales está caracterizado por n variables predictoras, $X_1, ..., X_n$ y una variable a predecir, la clase C.
- Los N casos se denotan por:

  $((x_1, c_1), ..., (x_N, c_N))$ para todo $i = 1, ..., N$ donde
  $(x_i = (x_{i,1} ... x_{i,n})$ para todo $i = 1, ..., N$
  $c_i \in \{c^1, ..., c^m\}$ para todo $i = 1, ..., N$

  $c^1, ..., c^m$ denotan los m posibles valores de la variable clase C.
- El nuevo caso que se pretende clasificar se denota por $x = (x_1, ..., x_n)$.

En la siguiente figura se presenta un pseudocódigo para el clasificador K-NN básico:

COMIENZO
  Entrada: $D = \{(\mathbf{x}_1, c_1), \ldots, (\mathbf{x}_N, c_N)\}$
    $\mathbf{x} = (x_1, \ldots, x_n)$ nuevo caso a clasificar
  PARA todo objeto ya clasificado (x_i, c_i)
    calcular $d_i = d(\mathbf{x}_i, \mathbf{x})$
  Ordenar $d_i (i = 1, \ldots, N)$ en orden ascendente
  Quedarnos con los K casos $D_{\mathbf{x}}^K$ ya clasificados más cercanos a $\mathbf{x}$
  Asignar a $\mathbf{x}$ la clase más frecuente en $D_{\mathbf{x}}^K$
FIN

Tal y como puede observarse en el mismo, se calculan las distancias de todos los casos ya clasificados al nuevo caso, x, que se pretende clasificar. Una vez seleccionados los K casos ya clasificados, D_x^k más cercanos al nuevo caso, x, a éste se le asignará la clase (valor de la variable C) más frecuente de entre los K objetos, D_x^k.

La siguiente figura muestra de manera gráfica un ejemplo de lo anterior:

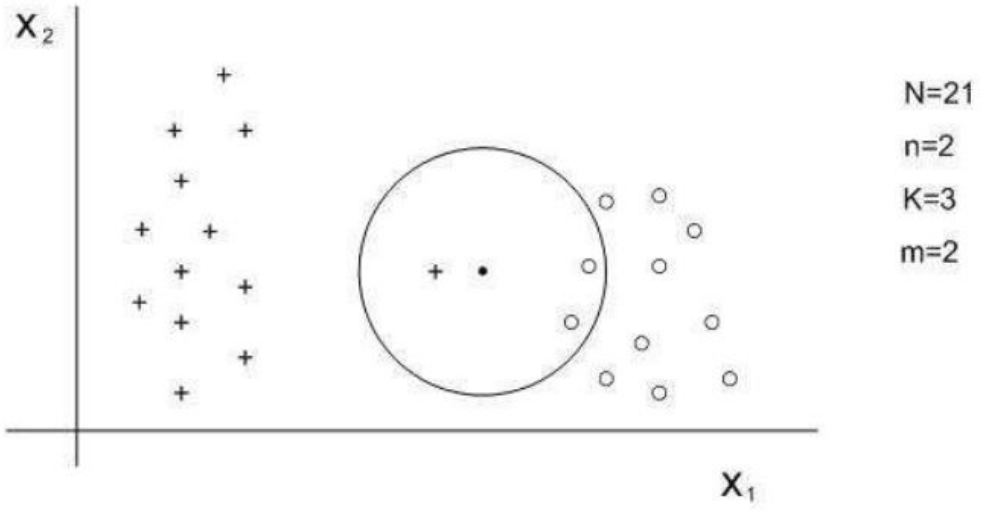
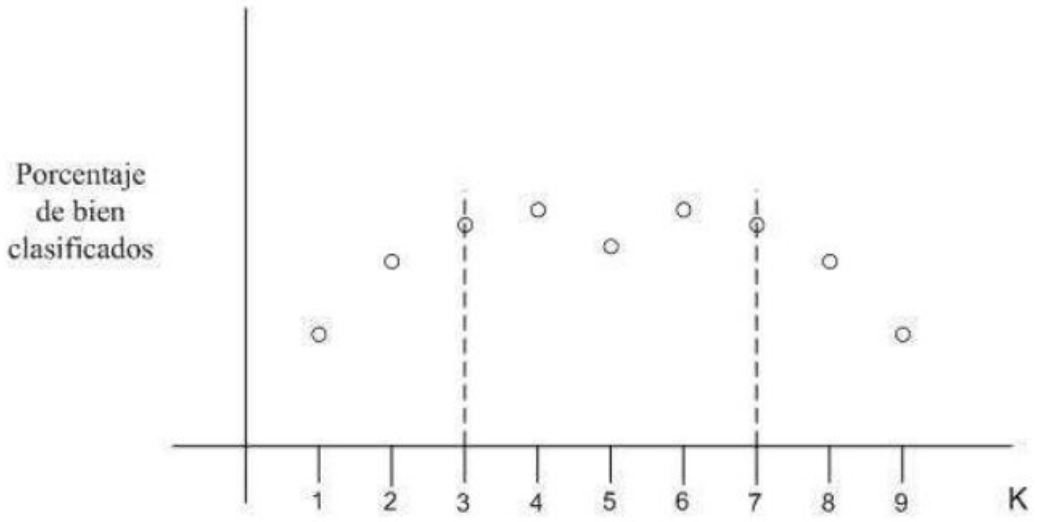


Figura 3: Ejemplo de aplicación del algoritmo K-NN básico

Tal y como se puede apreciar en la figura, tenemos 24 casos ya clasificados en dos posibles valores ($m = 2$). Las variables predictoras son X_1 y X_2, y se ha seleccionado $K = 3$. De los 3 casos ya clasificados que se encuentran más cercanos al nuevo caso a clasificar, x (representado por •), dos de ellos pertenecen a la clase °, por lo que el clasificador 3-NN predice la clase ° para el nuevo caso. Nótese que el caso más cercano a x pertenece a la clase +. Es decir, que si hubiésemos utilizado un clasificador 1-NN, x se hubiese asignado a +.

En caso de que se produzca un *empate* entre dos o más clases, conviene tener una *regla heurística* para su ruptura. Ejemplos de reglas heurísticas para la ruptura de empates pueden ser: seleccionar la clase que contenta al vecino más próximo, seleccionar la clase con distancia menor, etc.

Otra cuestión importante es la determinación del valor de K. Se constata empíricamente que el porcentaje de casos bien clasificados es no monótono con respecto de K (como puede verse en la figura siguiente):



De esta manera, una buena elección de valores para K serían aquellos comprendidos entre 3 y 7.

Variantes sobre el algoritmo básico

En este apartado vamos a introducir algunas variantes sobre el algoritmo básico.

K-NN con rechazo

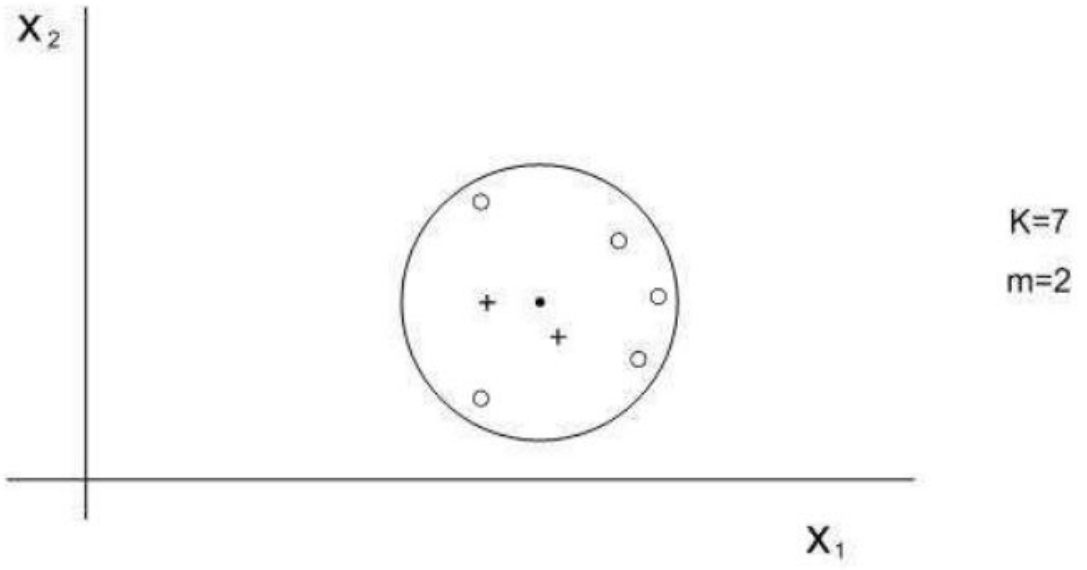
La idea subyacente al K-NN con rechazo es que para poder clasificar un caso se debe de tener ciertas garantías. Es por ello por lo que puede ocurrir que un caso quede sin clasificar si no existen ciertas garantías de que la clase a asignar sea la correcta.

Dos ejemplos utilizados para llevar a cabo clasificaciones con garantías son los siguientes:

- el número de votos obtenidos por la clase deberá superar un *umbral prefijado*. Si suponemos que trabajamos con $K = 10$, y $m = 2$, dicho umbral puede establecerse en 6.
- establecimiento de algún tipo de *mayoría absoluta* para la clase a asignar. Así, si suponemos que $K = 20$, $m = 4$, podemos convenir en que la asignación del nuevo caso a una clase sólo se llevará a cabo en el caso de que la diferencia entre las frecuencias mayor y segunda mayor supere 3.

K-NN con distancia media

En el K-NN con distancia media la idea es asignar un nuevo caso a la clase cuya distancia media sea menor. Un ejemplo de esto se visualiza en la siguiente figura:



En este ejemplo, a pesar de que 5 de los 7 casos más cercanos al mismo pertenecen a la clase °, el nuevo caso se clasifica como +, ya que la distancia media a los dos casos + es menor que la distancia media a los cinco casos °.

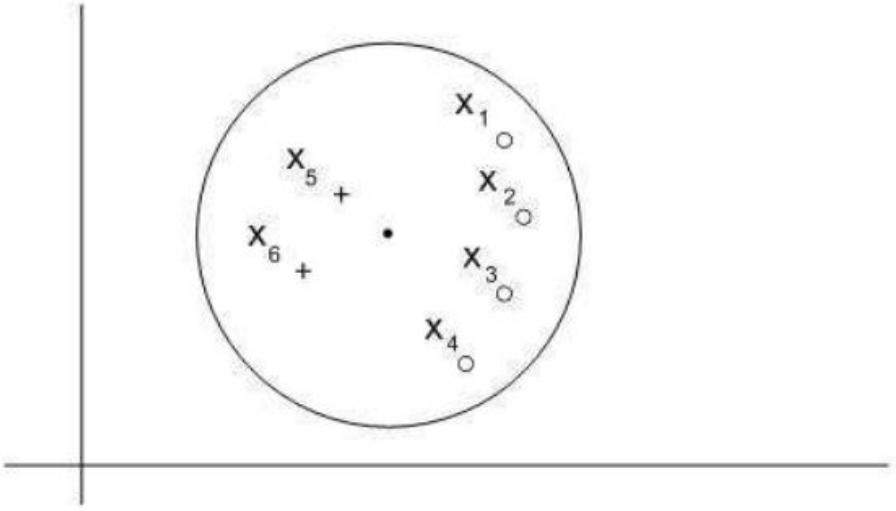
K-NN con distancia mínima

En el K-NN con distancia mínima se comienza seleccionando un caso por clase, normalmente el caso más cercano al baricentro de todos los elementos de dicha clase. En este paso se reduce la dimensión del fichero de casos a almacenar de N a m. A continuación se asigna el nuevo caso a la clase cuyo representante esté más cercano.

El procedimiento anterior puede verse como un 1-NN aplicado a un conjunto de m casos (uno por cada clase). El coste computacional de este procedimiento es inferior al K-NN genérico, si bien su efectividad está condicionada a la homogeneidad dentro de las clases (cuanto mayor sea dicha homogeneidad, se espera que el procedimiento sea más efectivo).

K-NN con pesado de casos seleccionados

La idea en el K-NN con el que se efectúa un pesado de los casos seleccionados es que los K casos seleccionados no se contabilicen de igual forma, sino que se tenga en cuenta la distancia de cada caso seleccionado al nuevo caso que pretendemos seleccionar. Como ejemplo podemos convenir en pesar cada caso seleccionado de manera inversamente proporcional a la distancia del mismo al nuevo caso. Esto se puede apreciar en las siguientes figuras:



	$d(\mathbf{x}_i, \mathbf{x})$	w_i
$\mathbf{x}_1$	2	0,5
$\mathbf{x}_2$	2	0,5
$\mathbf{x}_3$	2	0,5
$\mathbf{x}_4$	2	0,5
$\mathbf{x}_5$	0,7	1/0,7
$\mathbf{x}_6$	0,8	1/0,8

En la tabla se puede consultar el peso w_i que se asignaría a cada uno de los casos seleccionados provenientes de la gráfica. Como resulta que, de los 6 casos seleccionados, los pesos relativos a los casos tipo ° suman 2, que es una cantidad inferior al peso de los dos casos tipo + $(\frac{1}{0,7} + \frac{1}{0,8})$, el K-NN con pesado de casos seleccionados clasificaría el nuevo caso como perteneciente a la clase +.

K-NN con pesado de casos seleccionados

En todas las aproximaciones presentadas hasta el momento, la distancia entre el nuevo caso que se pretende clasificar, x, y cada uno de los casos x_r, $r = 1, ..., N$ ya clasificados pertenecientes al fichero de casos D, da el mismo peso a todas y cada una de las n variables, $X_1, ..., X_n$. Es decir, la distancia $d(x, x_r)$ entre x y x_r se calcula por ejemplo por medio de la distancia euclídea de la siguiente manera:

$$d(x, x_r) = \sum_{j=1}^{n} (x, x_{rj})^2$$

Esta manera de calcular la distancia, es decir, otorgando la misma importancia a todas las variables, puede resultar peligrosa para el paradigma K-NN en el caso de que algunas de las variables sean irrelevantes para la variable clase C. Este es el motivo por el que resulta interesante el utilizar distancias entre casos que ponderen cada variable de una manera adecuada. Es decir, la distancia entre x y x_r se calcularía:

$$d(x, x_r) = \sum_{j=1}^{n} w_j (x, x_{rj})^2$$

con lo cual la variable X_j tiene asociado un peso w_j que habrá que determinar. Para clasificar la idea supongamos los datos de la siguiente figura:

X_1	X_2	C
0	0	1
0	0	1
0	0	1
1	0	1
1	0	1
1	1	1
0	1	0
0	1	0
0	1	0
1	1	0
1	1	0
1	0	0

En esta figura se puede observar que la variable X_1 es irrelevante para la variable C, ya que las 6 veces en las que X_1 toma el valor 0, en tres de ellas C

toma el valor 1, y en las otras tres el valor 0, mientras que de las 6 veces en las que X_1 toma el valor 1, en tres de ellas C toma el valor 0, y las tres restantes toma el valor 1. Esta situación es totalmente opuesta a la situación con la variable X_2. De las 6 veces en que X_2 toma el valor 0, en 5 casos C vale 1, valiendo 0 en el caso restante, mientras que de las 6 veces en las que X_2 toma el valor 1, en 5 casos C vale 0 valiendo 1 en el caso restante.

Ante la situación anterior parece intuitivo que en el cálculo de las distancias se deba de pesar más la aportación de la variable X_2 que la aportación de la variable X_1. Una manera de calcular la ponderación w_i de cada variable X_i es a partir de la **medida de información mínima** $I(X_i, C)$ entre dicha variable X_i y la variable clase C. Es la medida de la cantidad de información que una variable aleatoria contiene sobre la otra. La información mutua de dos variables aleatorias se define de la siguiente manera:

$$I(X_i, C) = \sum_{x_i} \sum_{c} p_{X_i C}(x_i, c) \cdot log_2 \frac{p_{X_i C}(x_i, c)}{p_{X_i}(x_i) \cdot p_C(c)}$$

para todo $i = 1, ..., n$.

Esta medida de información mutua entre dos variables se interpreta como la reducción en la incertidumbre sobre una de las variables cuando se conoce el valor de la otra variable. Cuanto mayor sea la medida de información mutua entre dos variables mayor será la dependencia existente entre las mismas.

Reducción del fichero de casos inicial

En este apartado veremos dos aproximaciones distintas, con las que se trata de paliar la fuerte dependencia del costo computacional del clasificador K-NN con respecto del número de casos, N, almacenados. Los distintos métodos de reducción del fichero de casos inicial los podemos clasificar en *técnicas de edición* y en *técnicas de condensación*.

En las técnicas de edición, el subconjunto del fichero inicial se obtiene por medio de la eliminación de algunos de los casos, mientras que las técnicas de condensación tratan de obtener el subconjunto del fichero inicial por medio de la selección de unos cuantos casos.

Edición de Wilson

La idea en la edición propuesta por Wilson es el someter a prueba a cada uno de los elementos del fichero de casos inicial. Para ello, para cada caso se compara su clase verdadera con la que propone un clasificador K-NN obtenido con todos los casos excepto el mismo. En el caso de que ambas clases no coincidan, el caso es eliminado. En cierto modo, el método tiene una analogía con la validación *leave-one-out*.

Se puede también considerar una edición de Wilson repetitiva, y parar el procedimiento cuando en 2 selecciones sucesivas no se produzcan cambios.

Condensación de Hart

El condensado de Hart efectúa una selección de casos que pueden considerarse raros. Para ello, para cada caso, y siguiendo el orden en el que se encuentran almacenados los casos en el fichero, se construye un clasificador K-NN con tan sólo los casos anteriores al caso en cuestión. Si el caso tiene un valor de la clase distinto al que le asignaría el clasificador K-NN, el caso es seleccionado. Si por el contrario la clase verdadera del caso coincide con la propuesta del clasificador K-NN, el caso no se selecciona. Es claro que el método de condensación de Hart es dependiente del orden en el que se encuentren almacenados los casos en el fichero.

Método de regresión basado en KNN

Un método de aproximación simple no paramétrica es el basado en la regla del vecino más cercano, que consiste en estimar el valor de un dato desconocido a partir de las características del dato más próximo, según una medida de similitud o distancia. El método del vecino más cercano se puede extender utilizando no uno, sino un conjunto de datos más cercanos para predecir el valor de los nuevos (los **k-vecinos** mencionados anteriormente). Al considerar más de un vecino, se brinda inmunidad ante ruido y se suaviza la curva de estimación.

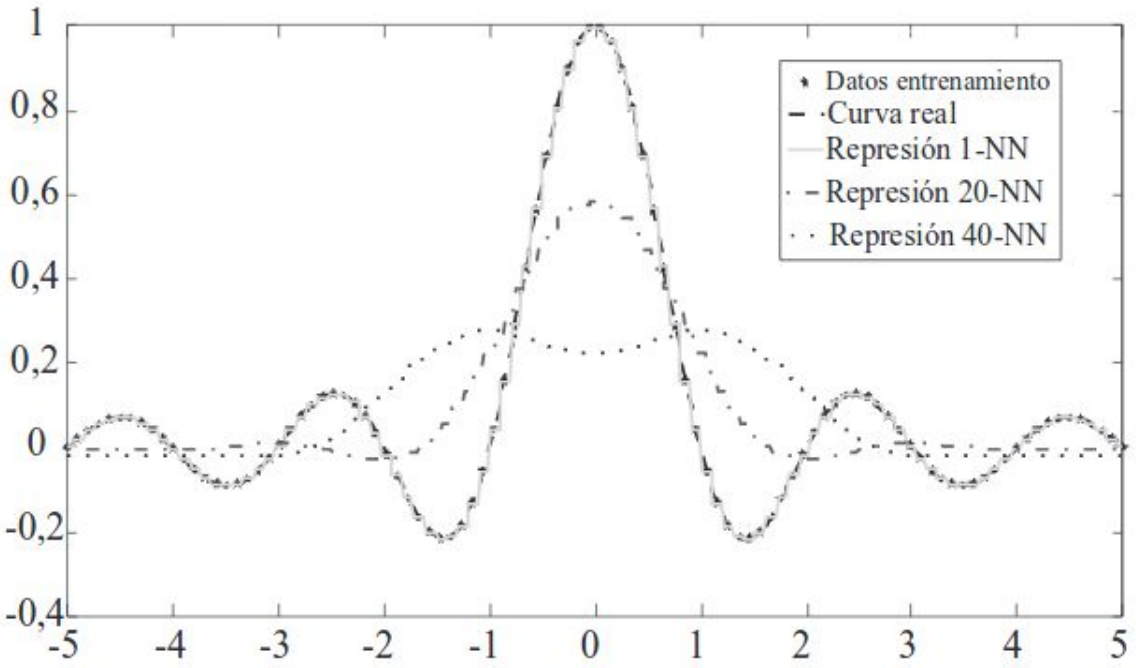
El método de los k-vecinos más cercanos se adapta fácilmente a la regresión de funciones con valores continuos. El algoritmo asume que todos los datos pertenecen a R^p, y mediante una medida de distancia en ese espacio se determinan los k datos más cercanos al nuevo dato x_q para aproximar una función $f : R^p \rightarrow R$ a partir de los k valores ya seleccionados. Esta función corresponde al promedio de los k valores más cercanos; si se considera el promedio aritmético (todos los datos dentro del grupo tienen igual relevancia), la

función aproximación tiene la siguiente forma:

$$\widehat{f}(x_q) \leftarrow \ : \frac{1}{k} \sum_{i=1}^{k} f(x_i)$$

Todos los datos deben estar normalizados, para evitar que las características en el conjunto de entrada con valores más altos dominen el cálculo de la distancia.

Como ejemplo de análisis, se presenta la construcción de la curva de regresión de la función $sinc(x)$ con 100 datos linealmente espaciados en el intervalo [-5,5]:



La medida de distancia utilizada es la *euclídea clásica*. En la línea continua de forma escalonada se muestra la curva de regresión utilizando un vecino más cercano (1-NN). Cuando se aplica el método de k-NN con promedio aritmético como se muestra en la curva a trazos para $k = 20$, se observa una curva más suavizada, sin embargo la aparente mejor regresión se descompensa con la tendencia lineal de la curva a medida que k aumenta, como se observa en la curva para $k = 40$. Lo anterior se debe a que el valor calculado ante un nuevo dato corresponde al valor medio de los k datos más cercanos.

Debido a que el método de k-NN se basa en la distancia, su desempeño se mejora al considerar un promedio ponderado, que da mayor importancia a los datos más cercanos al nuevo ejemplar, tal como se presenta en la siguiente fórmula:

$$\widehat{f}(x_q) \leftarrow \frac{\sum_{i=1}^{k} w_i \cdot f(x_i)}{\sum_{i=1}^{k} w_i}$$

$$w_i \equiv KER(d(x_q, x_i))$$

El peso w_i que se muestra en la fórmula está definido como el resultado de una función denominada *KER*, que es una función *Kernel* que determina la ponderación de cada punto basado en la distancia al punto de referencia dada por la función *d*. La función *Kernel* corresponde al peso que se le da a los datos para obtener un promedio ponderado, por lo tanto debe variar inversamente con la distancia para que los puntos más cercanos tengan mayor peso (mayor importancia)

Con la ponderación por distancia, se soluciona el problema de linealización que se presenta al considerar valores grande de *k* en la predicción del nuevo valor, ya que los datos más lejanos tendrán poco efecto sobre el valor a predecir. Utilizando una ponderación igual al cuadrado del inverso de la distancia, se obtiene:

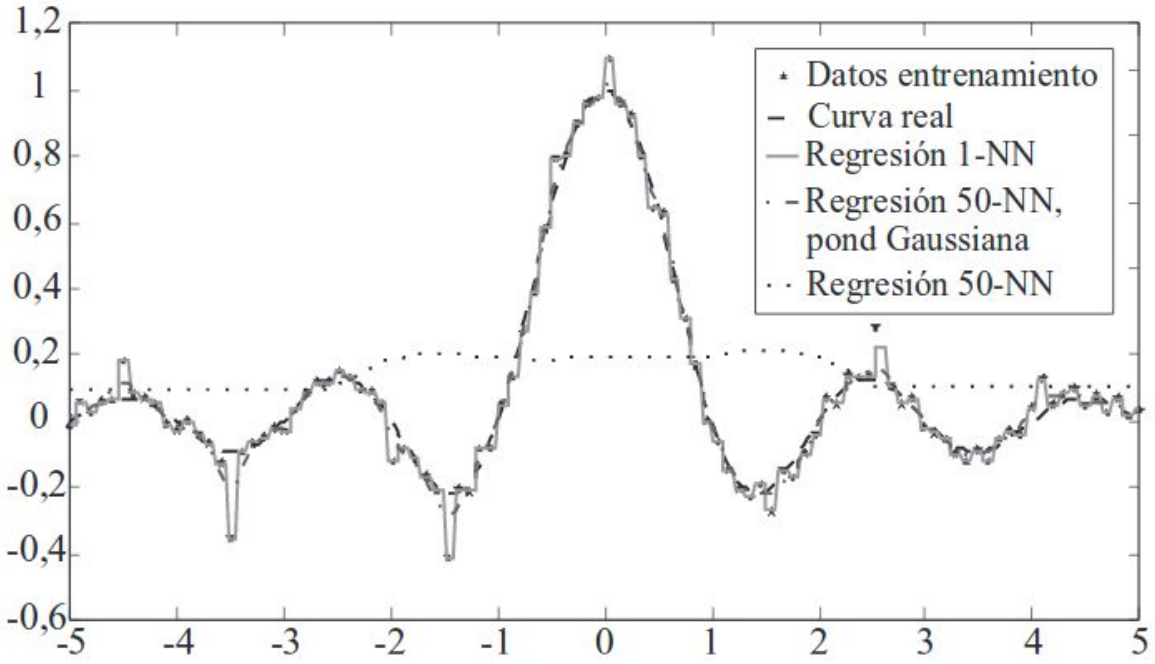
$$w_i \equiv d(x_q, x_i)^{-2}$$

Para el caso $x_i = x_q$, que resulta en un denominado cero para w, se asigna el valor de $f(x_i)$ a $\widehat{f}(x_q)$.

Sin embargo, pese a que se usa un *Kernel*, se tiene un sobreentrenamiento, ya que la curva siempre pasa por los valores de todos los datos, haciendo la curva vulnerable al ruido. Como solución a este problema se consideran funciones *Kernel* que a distancia cero la ponderación sea infinita. Un ejemplo de este tipo de ponderaciones y la más utilizada es el *Kernel* Gaussiano:

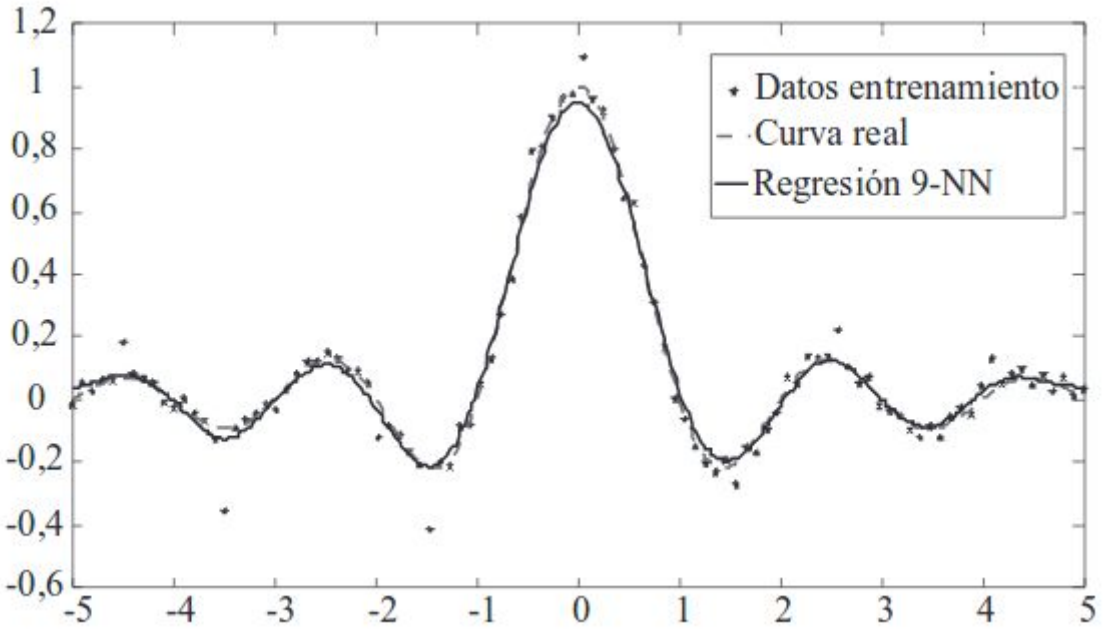
$$w_i \equiv e^{-d(x_q, x_i)^2}$$

La aplicación del *Kernel* Gaussiano para la regresión empleando 50-NN se presenta en la siguiente figura:



Los datos tienen ruido de distribución normal con desviación estándar del 3% de la amplitud de la señal, y la característica de que cada 5 datos el ruido posee mayor intensidad (desviación estándar del 10%). La aplicación del *Kernel* Gaussiano mejora la curva de regresión, atenuando los cambios bruscos de la ponderación según el cuadrado del inverso de la distancia. Aunque la curva de regresión usando *Kernel* Gaussiano tiene un alto error con los datos de entrenamiento, su comportamiento es fiel a la curva real, no se afecta por el ruido en los datos y conserva su tendencia.

El valor k debe ser elegido de tal manera que la curva de regresión sea lo más similar posible a la original, lo cual se logra aplicando validación cruzada en la escogencia de *k*. Con validación cruzada, se obtiene $k = 9$ y la curva resultante se muestra en la figura siguiente:



donde se observa que esta curva es más similar a la curva real que la obtenida con $k = 50$ mostrada en la anterior figura a ésta.

Ejemplo en Python

- Implementar k-vecinos más cercanos desde cero

Bibliography

- https://machinelearningmastery.com/gentle-introduction-gradient-boosting-algorithm-machine-learning/ (A Gentle Introduction to the Gradient Boosting Algorithm for Machine Learning)
- https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/ (A Gentle Introduction to XGBoost for Applied Machine Learning)
- https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/ (How to Develop Your First XGBoost Model in Python with scikit-learn)
- https://lightgbm.readthedocs.io/en/latest/ (LightGBM)
- https://medium.com/@pushkarmandot/https-medium-com-pushkarmandot-what-is-lightgbm-how-to-implement-it-how-to-fine-tune-the-parameters-60347819b7fc (LightGBM)
- https://www.analyticsvidhya.com/blog/2017/06/which-algorithm-takes-the-crown-light-gbm-vs-xgboost/ (Comparison between XGBoost and LightGBM)

- https://machinelearningmastery.com/neural-networks-crash-course/ (Multi-Layer Perceptrons)
- https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/ (Neural networks in Python)
- https://www.springboard.com/blog/beginners-guide-neural-network-in-python-scikit-learn-0-18/
- http://benalexkeen.com/feature-scaling-with-scikit-learn/ -> Feature Scaling with scikit-learn
- https://machinelearningmastery.com/arima-for-time-series-forecasting-with-python/ -> How to Create an ARIMA Model for Time Series Forecasting with Python
- https://www.ibm.com/support/knowledgecenter/es/SS3RA7_sub/modeler_mainhelp_client_ddita/components/dt/timeseries_data.html (IBM series temporales)
- https://drive.google.com/open?id=13HkVDNrJKlLrRAoqKfcMBLU2vvnaioD6 (Documento en el que se explican los conceptos básicos de las series temporales)
- http://halweb.uc3m.es/esp/Personal/personas/jmmarin/esp/EDescrip/tema7.pdf (Series temporales)
- http://www.ugr.es/~fabad/desestacionalizacion.pdf (Desestacionalización de una serie temporal)
- http://www2.ulpgc.es/hege/almacen/download/5/5672/Tema4.pdf (Series temporales -> Para mirar la desestacionalización)
- https://estrategiastrading.com/series-estacionarias/ (Series estacionarias: Por qué son importantes para trabajar con modelos)
- http://www.fuenterrebollo.com/Economicas2013/series-temporales-ejercicios.pdf (Ejercicio resuelto medias móviles)
- http://metodos.upct.es/Asignaturas/Diplomatura/Introduccion_estadistica/2008_2009/material_didactico/apuntes/TEMA5SERIESTEMPORALES.pdf (Series temporales
- http://www5.uva.es/estadmed/datos/series/series2.htm (Esquemas de las series temporales)
- http://www.est.uc3m.es/esp/nueva_docencia/comp_col_get/lade/Econometria_II_NOdocencia/Documentaci%C3%B3n%20y%20apuntes/TEMA%203_Procesos%20estoc%C3%A1sticos.%20La%20FAC%20y%20el%20correlograma.pdf (Proceso de ruido blanco)
- http://www.uam.es/personal_pdi/economicas/rarce/pdf/Box-Jenkins.PDF (Box-Jenkins)
- https://machinelearningmastery.com/model-residual-errors-correct-time-series-forecasts-python/ (How to Model Residual Errors to Correct Time Series Forecasts with Python)

- http://ucanalytics.com/blogs/step-by-step-graphic-guide-to-forecasting-through-arima-modeling-in-r-manufacturing-case-study-example/ (Modelo ARIMA -> Mejor enlace para ver una visión global del modelo)
- https://onlinecourses.science.psu.edu/stat510/node/67 (Seasonal ARIMA models)
- https://machinelearningmastery.com/gentle-introduction-autocorrelation-partial-autocorrelation/ (Introducción a la autocorrelación y a la autocorrelación parcial)
- http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t9knn.pdf (Clasificadores K-NN)
- http://www.cs.us.es/~fsancho/?e=77 (Clasificación supervisada y no supervisada)
- http://opac.pucv.cl/pucv_txt/txt-5500/UCE5803_01.pdf (Selección de variables de entrada para procesos autorregresivos, mediante el cálculo de la información mutua usando los k-vecinos más cercanos)
- http://www.scielo.org.co/pdf/rfiua/n45/n45a09.pdf (Estrategia de regresión basada en el método de los k vecinos más cercanos para la estimación de la distancia de falla en sistemas radiales)
- https://machinelearningmastery.com/k-nearest-neighbors-for-machine-learning/ (K-Nearest Neighbors for Machine Learnig)