MachineLearning Models

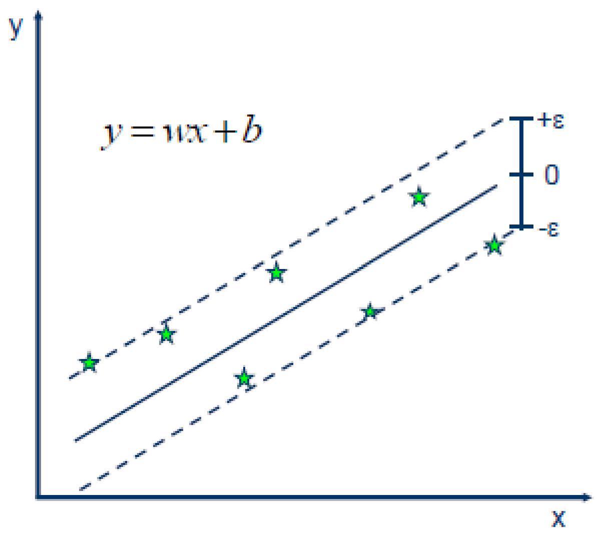
- Support Vector Regression
- Decision Tree Regression
- Random Forest
- Naive Bayes Classification
- Gradient Boosted Trees
- XG Boost
- Deep Learning
- Autoregressive Model

Support Vector Regression

SVR is an regression algorithm , so we can use **SVR** for **working** with continuous Values instead of Classification which is SVM. Kernel: The function used to map a lower dimensional data into a higher dimensional data.In simple regression we minimise the error rate. While in SVR we try to fit the error within a certain threshold.

Support Vector Machine can also be used as a regression method, maintaining all the main features that characterise the algorithm (maximal margin). The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences.

First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities. In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration. However, the main idea is always the same: to minimize error, individualising the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated.

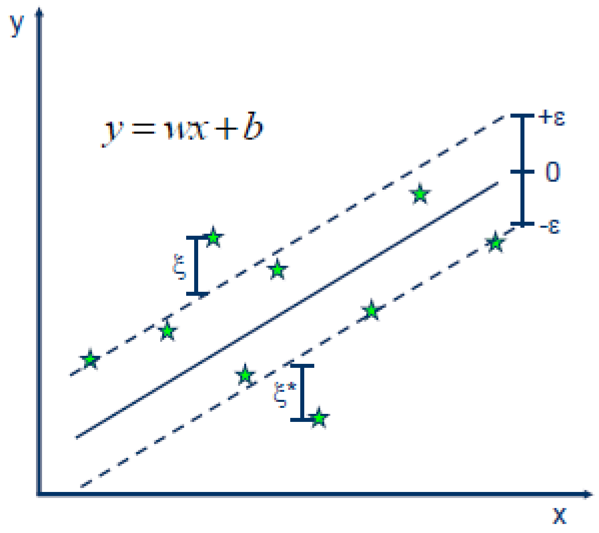


- Solution:

$$\min \frac{1}{2}\|w\|^2$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon$$
$$wx_i + b - y_i \le \varepsilon$$



- Minimize:

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N}\left(\xi_i + \xi_i^*\right)$$

- Constraints:

$$y_i - wx_i - b \le \varepsilon + \xi_i$$
$$wx_i + b - y_i \le \varepsilon + \xi_i^*$$
$$\xi_i, \xi_i^* \ge 0$$

Linear SVR

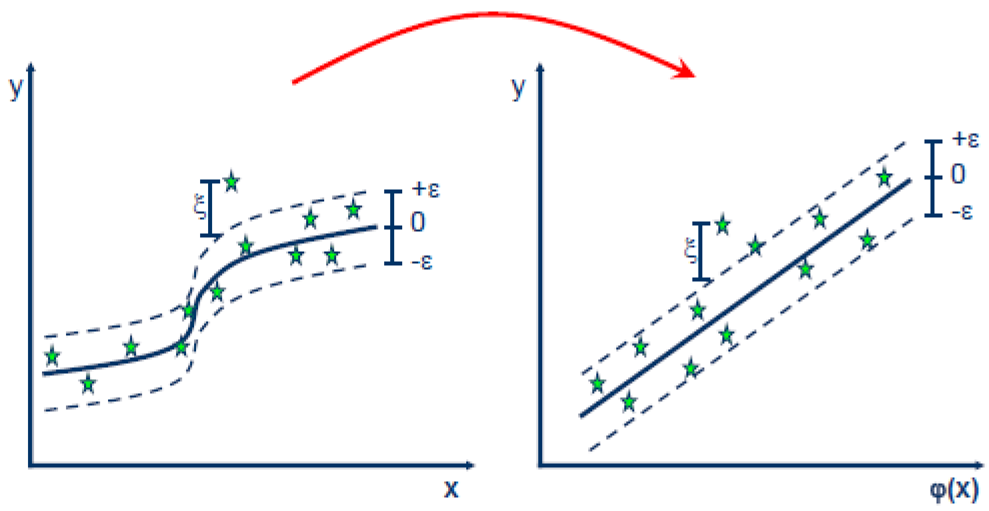
$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot\langle x_i, x\rangle + b$$

Non-linear SVR

The kernel functions transform the data into a higher dimensional feature space to make it possible to perform the linear separation.

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot\langle \varphi(x_i), \varphi(x)\rangle + b$$

$$y = \sum_{i=1}^{N}\left(\alpha_i - \alpha_i^*\right)\cdot K(x_i, x) + b$$



Kernel functions

Polynomial

$$k(\mathrm{x}_i,\mathrm{x}_j)=(\mathrm{x}_i.\mathrm{x}_j)^d$$

Gaussian Radial Basis function

$$k(\mathrm{x}_i,\mathrm{x}_j)=\exp\left(-\frac{\left\|\mathrm{x}_i-\mathrm{x}_j\right\|^2}{2\sigma^2}\right)$$

Decision Tree Regression

Decision Tree Regression Decision trees (DT) can be used both in classification and regression problems. Quinlan proposed ID3 algorithm as the first decision tree algorithm (Quinlan, 1986). Decision tree algorithms classify both categorical (classification) and numerical (regression) samples in a form of tree structure with a root node, internal nodes and leaf nodes. Internal nodes contain one of the possible input variables (features) available at that point in the tree. The selection of input variable is chosen using information gain or impurity for classification problems and standard deviation reduction for regression problems. The leaves represent labels/predictions. Random forest and gradient boosting algorithms are both decision tree based algorithms. In this study, decision tree method is applied for regression problems where variance reduction is employed for selection of variables in the internal nodes. Firstly, variance of root node is calculated using Equation 6, then variance of features is calculated using Equation 7 to construct the tree.

$$\sigma^2=\frac{\sum_{i=1}^{n}(x_i-\mu)^2}{n} \qquad (6)$$

In Equation 6, n is the total number of samples and μ is the mean of the samples in the training set. After calculating variance of the root node, variance of input variables is calculated as follows:

$$\sigma_X^2 = \sum_{c \varepsilon X} P(c)\sigma_c^2 \qquad (7)$$

In Equation 7, X is the input variable and c’s are the distinct values of this feature. For example, X : Brand and c : Samsung, Apple or Nokia. P(c) is the probability of c being in the attribute X and 2 c is the variance of the value c. Input variable that has the minimum variance or largest variance reduction is selected as the best node as shown in Equation 8:

$$vr_X = \sigma^2 - \sigma_X^2 \qquad (8)$$

Finally leaves are representing the average values of instances with **Bootstrap Method**

The bootstrap method is a statistical technique for estimating quantities about a population by averaging estimates from multiple small data samples.

This process continues recursively, until variance of leaves is smaller than a threshold or all input variables are used. Once a tree has been constructed, new instance is tested by asking questions to the nodes in the tree. When reaching a leaf, value of that leaf is taken as prediction.

Random Forest

Random Forest is one of the most effective machine learning models for predictive analytics.

random forest model is a type of additive model that makes predictions by combining decisions from a sequence of base models. More formally we can write this class of models as:

g(x)=f0(x)+f1(x)+f2(x)+...

where the final model g is the sum of simple base models fi. Here, each base classifier is a simple decision tree. This broad technique of using multiple models to obtain better predictive performance is called model ensembling. In random forests, all the base models are constructed independently using a different subsample of the data.

WHY CHOSE RANDOM FORESTS?

Different kinds of models have different advantages. The random forest model is very good at handling tabular data with numerical features, or categorical features with fewer than hundreds of categories. Unlike linear models, random forests are able to capture non-linear interaction between the features and the target.

One important note is that tree based models are not designed to work with very sparse features. When dealing with sparse input data (e.g. categorical features with large dimension), we can either pre-process the sparse features to generate numerical statistics, or switch to a linear model, which is better suited for such scenarios.

- Random forest algorithm can use both for classification and the regression kind of problems.
- Random forest algorithm is a supervised classification algorithm.
- Random Forest creates forest with number of trees.
- Advantages of Random Forest are:

1. Same forest classifier can use for both classification and the regression task
2. Random forest classifier will handle the missing values
3. Random forest won’t overfit the model
4. Can model the random forest classifier for categorical values also

Overview of Random Forest

Random forest (RF) is a type of meta learner that uses number of decision trees for both classification and regression problems (Breiman, 2001). The features and samples are drawn randomly for every tree in the forest and these trees are trained independently. Each tree is constructed with bootstrap sampling method. Bootstrapping relies on sampling with replacement. Given a dataset D with N samples, a training data set of size N is created by sampling from D with replacement. The remaining samples in D that are not in the training set are separated as the test set. This kind of sampling is called bootstrap sampling. The probability of an example not being chosen in the dataset that has N samples is :

$$Pr = 1 - \frac{1}{N} \qquad (9)$$

The probability of being in the test set for a sample is:

$$Pr = \left(1 - \frac{1}{N}\right)^N \approx \exp^{-1} = 0.368 \qquad (10)$$

Every tree has a different test set and this set consists of totally %63.2 of data. Samples in the test set are called out-of-bag data. On the other hand, every tree has different features which are selected randomly. While selecting nodes in the tree, only a subset of the features are selected and the best one is chosen as separator node from this subset. Then this process continues recursively until a certain error rate is reached. Each tree is grown independently to reach the specified error rate. For instance, stock feature is chosen as the best separator node among the other randomly selected features, and likewise price feature is chosen as second best node for the first tree in Figure 3. This tree is constructed with two nodes such as stock and price, whereas TREE N has four nodes and some of them are different than the TREE 1.

Example :

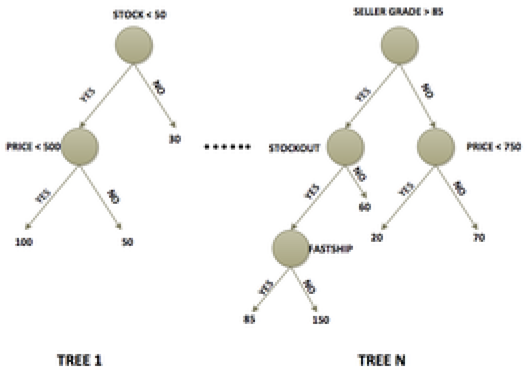


Figure 3: Random Forest

Due to bootstrapping sampling method, there is no need to use cross-validation or separate datasets for training and testing. This process is done internally. In this project, minimum root mean squared error was achieved by using random forest with 20 trees in the first level.

Naive Bayes Classification

Naive Bayes Classification is a supervised machine learning technique. It is simple but one of the most effective techniques of classification. There are some assumptions made in naive bayes. Even if these assumptions are violated a little, but it still performs quite well. Assumptions made in Naive Bayes:

- All the samples are i.i.d, i.e., all random variables are independent of each other and are drawn from the similar distribution.
- All features are conditionally independent.

Let us delve into the theorem on which Naive Bayes classifier is built, i.e., Bayes theorem. The Bayes theorem states that,

Posterior Probability

The posterior probability, in the context of a classification problem, can be interpreted as: “What is the probability that a particular object belongs to class i given its observed feature values?”

For Example

Here,

cj represent jth class of classes{1,2,3 ..n}

xi represent features vector of ith sample of samples{1,2,3 ..m}

Posterior Probability simply means**,** “given the feature vector xi what is the probability of sample i belonging to class cj?”

Objective Function of Naive Bayes: Maximize the posterior probability given the training data to formulate a decision rule for new data. The decision rule for above problem can be formulated as,

If a sample belongs to class j, then P(cj | xi) will be maximum.

In other words, for each test example i, we find:

Class-Conditional Probability

Under the naive assumption made above, the class-conditional probabilities or (likelihoods) of the samples can be directly estimated from the training data instead of evaluating all possibilities of x. Thus, given a d-dimensional feature vector x, the class conditional probability can be calculated as follows:

Here, P(xcj) simply means: “How likely is it to observe this particular pattern x given that it belongs to class cj?” The “individual” likelihoods for every feature in the feature vector can be estimated via the maximum-likelihood estimate, which is simply a frequency in the case of categorical data:

Where,

Nxi, cj: Number of times feature xi appears in samples from class cj.

Ncj: Total count of all features in class cj.

Prior Probability

In the context of pattern classification, the prior probabilities are also called class priors, which describe “the general probability of encountering a particular class.” Here,

P(cj) = Probability that new document belongs to class j.

If the prior has a uniform distribution, then it will be same for all the classes, and posterior probability will depend only on evidence and class-conditional probability. As evidence is common for all the classes, therefore, the posterior probability will depend only on class-conditional probability.

Prior knowledge can be obtained from the experts or can be estimated through the training data available. For the later method, it is necessary that the training data is i.i.d. and representative sample of the entire population. The maximum likelihood estimate for the prior:

Ncj: Count of samples from class cj.

Nc : Total number of samples

Evidence

The evidence P(x) can be understood as the probability of encountering a particular pattern x independent from the class label.

We know that,

So,

Although the evidence term is required to accurately calculate the posterior probabilities, it can be removed from the decision rule, as its only contribution to the classification is of normalization.

Additive Smoothing

If during testing time, we come across a feature that we didn't come across during training time then the individual conditional probability of that particular feature will become zero, thus making class-conditional probabilities equal to zero. So we have to modify the formula for calculating individual conditional probability. In order to avoid the problem of zero probabilities, an additional smoothing term can be added to the multinomial Bayes model. The most common variants of additive smoothing are the so-called Lidstone smoothing (<1) and Laplace smoothing (=1).

Nxi, cj: Number of times feature xi appears in samples from class cj.

Ncj: Total count of all features in class cj.

: Parameter for additive smoothing

d: Dimensionality of feature vector x

Gradient Boosted Trees

Gradient Boosted Trees (GBT) are ensemble learning of decision trees (Friedman, 2001). GBT are said to be the combination of gradient descent and boosting algorithms. Boosting methods aim at improving the performance of classification task by converting weak learners to strong ones. There are multiple boosting algorithms in literature (Oza and Russell, 2001), (Grabner and Bischof, 2006), (Grabner et al., 2006), (Tutz and Binder, 2006). Adaboost is the first boosting algorithm proposed by Freund and Schapire (Freund et al., 1999). It works by weighting each sample in the dataset. Initially all samples are weighted equally likely and after each training iteration, misclassified samples are re-weighted more heavily. Boosting algorithms consist of many weak learners and use weighted summation of them. A weak learner can be defined as a learner that performs better than random guessing and it is used to compensate the shortcomings of existing weak learners. Gradient boosted trees uses gradient descent algorithm for the shortcomings of weak learners instead of using re-weighting mechanism. This algorithm is used to minimize the loss function (also called error function) by moving in the opposite direction of the gradient and finds a local minimum. In literature, there are several different loss functions such as Gaussian L2, Laplace L1, Binomial Loss functions etc (Natekin and Knoll, 2013). Squared-error loss function, commonly used in many regression problems, is used.

Let L(yi ,F(xi)) be the loss function where yi is actual output and F(xi) is the model we want to fit in. Our aim is to minimize

$$J = \sum_{i=1}^{N} (y_i - F(x_i))^2$$

function.By using gradient descent algorithm,

$$F(x_i) = F(x_i) - \alpha \frac{\delta J}{\delta F(x_i)} \qquad (11)$$

In Equation 11, is learning rate that accelerates or decelerates the learning process. If the learning rate is very large then optimal or minimal point may be skipped. If the learning rate is too small, more iterations are required to find the minimum value of the loss function. While trees are constructed in parallel or independently in random forest ensemble learning, they are constructed sequential in gradient boosting. Once all trees have been trained, they are combined to give the final output.

Evaluation Method

We used Root Mean Squared Error (RMSE) to evaluate model performances. It is square root of the summation of differences between actual and predicted values. RMSE is frequently used in regression analysis. RMSE can be calculated as shown in Equation 12. ˆi's are predicted and 's are actual values.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (\hat{\gamma}_i - \gamma)^2}{n}} \qquad (12)$$

XG Boost

- XGBoost stands for e**X**treme **G**radient **B**oosting.
- XGBoost is an algorithm implementation of gradient boosted decision trees designed for speed and performance.
- XGBoost is a software library that can be downloaded and installed on machine and then accessed from a variety of interfaces.

Deep Learning

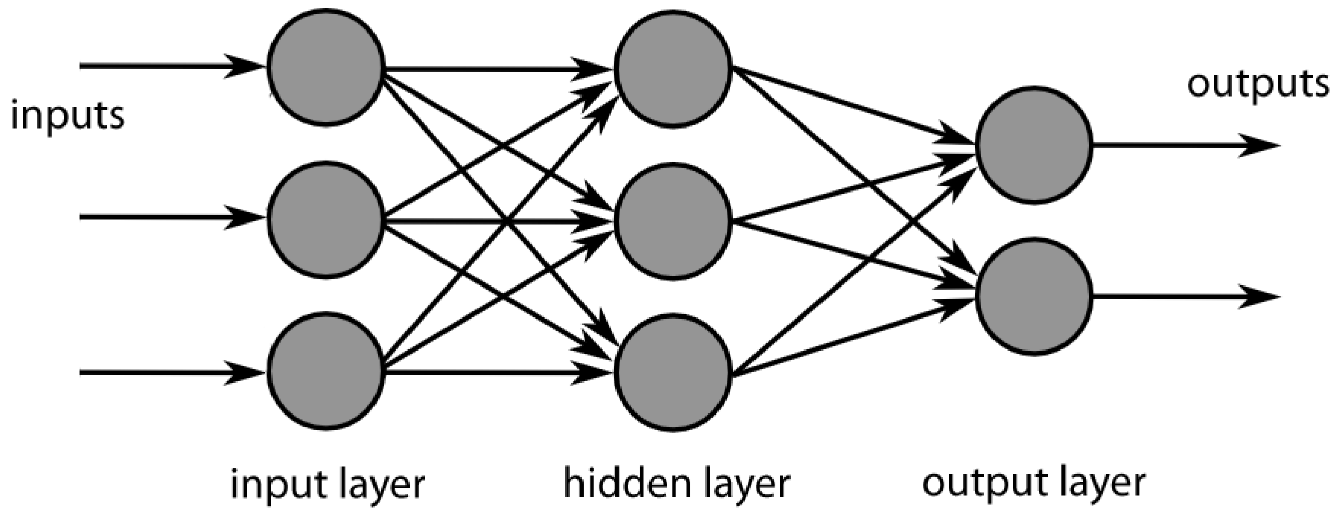
It is a sub-category of machine learning. Similar to machine learning, deep learning also has supervised, unsupervised, and reinforcement learning in it.

Deep learning is one of the only methods by which we can overcome the challenges of feature extraction. This is because deep learning models are capable of learning to focus on the right features by themselves, requiring little guidance from the programmer. Basically, deep learning mimics the way our brain functions i.e. it learns from experience. As you know, our brain is made up of billions of neurons that allows us to do amazing things. Even the brain of a one year old kid can solve complex problems which are very difficult to solve even using super-computers.

For example:

- Recognize the face of their parents and different objects as well.
- Discriminate different voices and can even recognize a particular person based on his/her voice.
- Draw inference from facial gestures of other persons and many more.

The idea of AI was inspired by the human brain. So, let's try to connect the dots here, deep learning was inspired by artificial neural networks and artificial neural networks commonly known as ANN were inspired by human biological neural networks. Deep learning is one of the ways of executing machine learning.



As we can see from the above figure, it is a shallow neural network which can be called as *Shallow Learning Network*. A neural network will always have:

- Input layer: It can be pixels of an image or a time series data
- Hidden layer: Commonly known as weights which are learned while the neural network is trained
- Output layer: Final layer mainly gives you a prediction of the input you fed into your network.

So, the neural network is an approximation function in which the network tries to learn the parameters (weights) in hidden layers which when multiplied with the input gives you a predicted output close to the desired output.

Deep Learning is nothing but stacking multiple such hidden layers between the input and the output layer, hence the name Deep learning

Deep Learning Methods

There are various methods designed to apply deep learning. Each proposed method has a specific use case like the kind of data you have, whether it is supervised or unsupervised learning you would like to apply, what type of task you would want to solve with the data. So depending on these factors, we choose one of the methods that can best solve your problem.

Some of the deep learning methods are:

- Convolutional Neural Network,
- Recurrent Neural Network,
- Long short-term memory,
- Auto Encoders
    - Denoising autoencoder
    - Stacked autoencoder
    - Variational autoencoder
    - Sparse autoencoder
- Generative Adversarial network

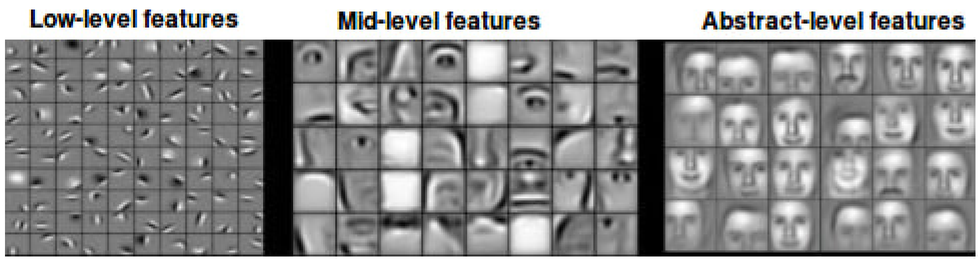
Deep learning can be used for:

- Classification: whether it is a task of predicting a cat or a dog and even multi-class classification,
- Feature extraction for visualizing the data in a lower dimensional space,
- Data preprocessing,
- Time series prediction: stock prediction, weather prediction, etc.,
- Regression tasks like localization,
- Object recognition which is both a classification & a regression task,
- Robotics.

Comparison between Deep Learning & Machine Learning!

- **Functioning**: Deep learning is a subset of machine learning that takes data as an input and makes intuitive and intelligent decisions using an artificial neural network stacked layer-wise. On the other hand, machine learning being a super-set of deep learning takes data as an input, parses that data, tries to make sense of it (decisions) based on what it has learned while being trained.

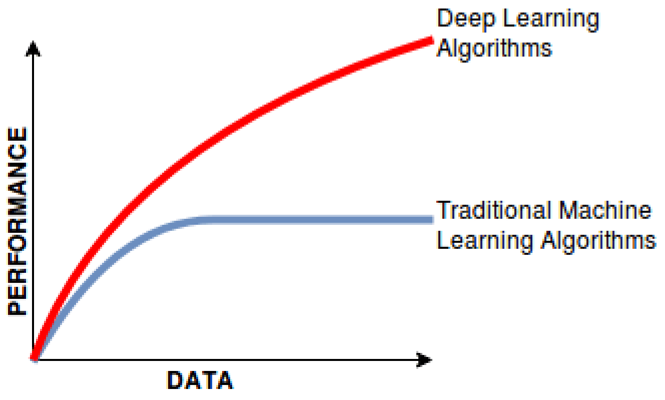
- **Feature Extractor**: Deep learning is considered to be a suitable method for extracting meaningful features from the raw data. It does not depend on hand-crafted features like local binary patterns, a histogram of gradients, etc., and most importantly it performs a hierarchical feature extraction. It learns features layer-wise which means that in initial layers it learns low-level features and as it moves up the hierarchy it starts to learn a more abstract representation of the data (as shown in the figure below). On the other hand, machine learning is not a good method for extracting meaningful features from the data. It relies on hand-crafted features as an input to perform well.



Layer-wise features learned by deep learning network

Here features mean pixel values, shape, textures, position, color and orientation. The performance of most of the traditional machine learning algorithm depends on how accurately the features are identified and extracted. Using traditional feature extractors does not solve the significant problem since even a slight variation in the data changes the features extracted from a conventional feature extractor like local binary pattern (LBP), a histogram of oriented gradients (HOG), etc. Whereas, a deep learning network tries to learn all of these features through a combination of layers at different levels and finally combines them to form a bigger picture as an abstract representation.

- **Data Dependency:** Machine learning algorithms often work well even if the dataset is small, but deep learning is *Data Hungry* the more data you have, the better it is likely to perform. It is often said that with more data the network depth (number of layers) also increase hence more computation.



Autoregressive Model

An autoregressive (AR) model **predicts future behavior based on past behavior**. It is used for forecasting when there is some correlation between values in a time series and the values that precede and succeed them. it will *only* use past data to model the behavior.The process is basically a linear regression of the data in the current series against one or more past values in the same series.

In an AR model, the value of the outcome variable (Y) at some point *t* in time is — like "regular" linear regression — directly related to the predic tor variable (X). Where simple linear regression and AR models differ is that Y is dependent on X **and previous values** for Y.

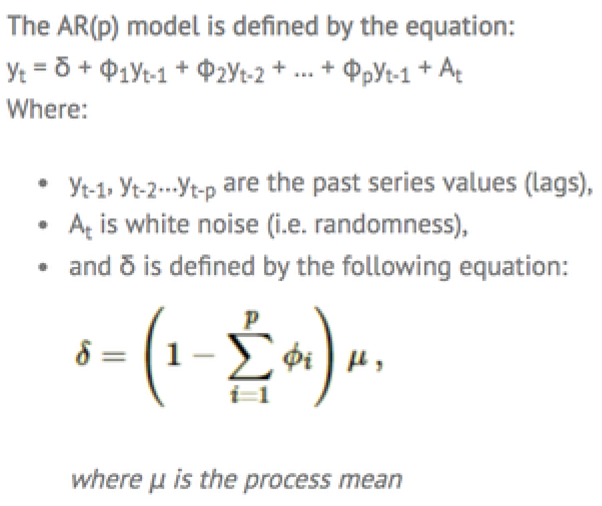
The AutoRegression process is an example of a stochastic process, which have degrees of uncertainty or randomness built in. The randomness means that you might be able to predict future trends pretty well with past data, but we are never going to get 100 percent accuracy. Usually, the process gets "close enough" for it to be useful in most scenarios.

An AR(p) model is an autoregressive model where specific lagged values of yt are used as predictor variables. Lags are where results from one time period affect following periods.

The value for "p" is called the *order*.

For example, an AR(1) would be a "first order autoregressive process."

The outcome variable in a first order AR process at some point in time *t* is related only to time periods that are one period apart (i.e. the value of the variable at t – 1). A second or third order AR process would be related to data two or three periods apart.



The AR(p) model is defined by the equation:

$y_t = \delta + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \ldots + \Phi_p y_{t-1} + A_t$

Where:

- $y_{t-1}, y_{t-2} \ldots y_{t-p}$ are the past series values (lags),
- A_t is white noise (i.e. randomness),
- and δ is defined by the following equation:

$$\delta = \left(1 - \sum_{i=1}^{p} \phi_i\right) \mu \,,$$

where μ is the process mean

Moving Average Model

MA(*q*) Model

The moving average (MA) model captures serial autocorrelation in a time series *Yt* by expressing the conditional mean of *Yt* as a function of past innovations,
*t*1,*t*2,…,*tq.*

An MA model that depends on ***q*** past innovations is called an MA model of degree ***q***, denoted by MA(***q***).

The form of the MA(***q***) model in Econometrics Toolbox™ is

yt=c + t + 1*t*1 + …+ *q tq,*

where *t* is an uncorrelated innovation process with mean zero. For an MA process, the unconditional mean of *Yt* is = *c*.

In lag operator polynomial notation, ***Li Yt=Yti***. Define the degree *q* MA lag operator polynomial (*L*) = (1 + 1 *L* + … +*qLq*). We can write the MA(*q*) model as

Yt = + (*L*) *t*.

ARMA model

ARMA model and its parameters Autoregressive–moving-average (ARMA) models provide a parsimonious description of a stationary stochastic process in terms of two polynomials, one for the autoregression and the second for the moving average. The model consists of two parts, an autoregressive (AR) part and a moving average (MA) part. The AR part involves regressing the variable on its own past values. The MA part involves modeling the error term as a linear combination of error terms occurring contemporaneously and at various times in the past.

The notation ARMA(p, q) refers to the model with p autoregressive terms and q moving-average terms

$$X_n = c + e_n + \sum_{i=1}^{p} a_i X_{n-i} + \sum_{i=1}^{q} b_i e_{n-i}$$

```
x[n] = c + e[n] + np.sum(a * x[np.arange((n-1),(n-p-1),-1)]) +
                  np.sum(b * e[np.arange((n-1),(n-q-1),-1)])
```

where ai and bi are the parameters of the model, c is a constant, and et is white noise.

ARMA models in general   after choosing p and q, fitted by least squares regression to find the values of the parameters which minimize the error term.

ARIMA

ARIMA stands for Autoregressive Integrated Moving Average models. Univariate (single vector) ARIMA is a forecasting technique that projects the future values of a series based entirely on its own inertia. **Its main application is in the area of short term forecasting requiring at least 40 historical data points. It works best when your data exhibits a stable or consistent pattern over time with a minimum amount of outliers.** Sometimes called Box-Jenkins , ARIMA is usually superior to exponential smoothing techniques when the data is reasonably long and the correlation between past observations is stable. If the data is short or highly volatile, then some smoothing method may perform better. If you do not have at least 38 data points, we should go for some other method than ARIMA.

When applying ARIMA methodology is to check for stationarity. "Stationarity" implies that the series remains at a fairly constant level over time. If a trend exists, as in most economic or business applications, then your data is NOT stationary. The data should also show a constant variance in its fluctuations over time. This is easily seen with a series that is heavily seasonal and growing at a faster rate. In such a case, the ups and downs in the seasonality will become more dramatic over time. Without these stationarity conditions being met, many of the calculations associated with the process cannot be computed.

If a graphical plot of the data indicates nonstationarity, then we should "difference" the series. Differencing is an excellent way of transforming a non-stationary series to a stationary one. This is done by subtracting the observation in the current period from the previous one. If this transformation is done only once to a series, you say that the data has been "first differenced". This process essentially eliminates the trend if your series is growing at a fairly constant rate. If it is growing at an increasing rate, you can apply the same procedure and difference the data again. Your data would then be "second differenced".

Autocorrelations:

"Autocorrelations" are numerical values that indicate how a data series is related to itself over time. More precisely, it measures how strongly data values at a specified number of periods apart are correlated to each other over time. The number of periods apart is usually called the "lag".

For example,

an autocorrelation at lag 1 measures how values 1 period apart are correlated to one another throughout the series. An autocorrelation at lag 2 measures how the data two periods apart are correlated throughout the series.Autocorrelations may range from +1 to -1. A value close to +1 indicates a high positive correlation while a value close to -1 implies a high negative correlation. These measures are most often evaluated through graphical plots called "correlagrams". A correlagram plots the auto- correlation values for a given series at different lags. This is referred to as the "autocorrelation function" and is very important in the ARIMA method.

ARIMA methodology attempts to describe the movements in a stationary time series as a function of what are called "autoregressive and moving average" parameters. These are referred to as AR parameters (autoregessive) and MA parameters (moving averages).

An AR model with only 1 parameter may be written as...

X(t) = A(1) * X(t-1) + E(t)

where X(t) = time series under investigation

A(1) = the autoregressive parameter of order 1

X(t-1) = the time series lagged 1 period

E(t) = the error term of the model

This simply means that any given value X(t) can be explained by some function of its previous value, X(t-1), plus some unexplainable random error, E(t). If the estimated value of A(1) was .30, then the current value of the series would be related to 30% of its value 1 period ago. Of course, the series could be related to more than just one past value. For example,

X(t) = A(1) * X(t-1) + A(2) * X(t-2) + E(t)

This indicates that the current value of the series is a combination of the two immediately preceding values, X(t-1) and X(t-2), plus some random error E(t). So this model is an autoregressive model of order 2.

A second type of Box-Jenkins model is called a "moving average" model. Although these models look very similar to the AR model, the concept behind them is quite different. Moving average parameters relate what happens in period t only to the random errors that occurred in past time periods, i.e. E(t-1), E(t-2), etc. rather than to X(t-1), X(t-2), (Xt-3) as in the autoregressive approaches. A moving average model with one MA term may be written as follows

X(t) = -B(1) * E(t-1) + E(t)

The term B(1) is called an MA of order 1. The negative sign in front of the parameter is used for convention only and is usually printed out auto-matically by most computer programs. The above model simply says that any given value of X(t) is directly related only to the random error in the previous period, E(t-1), and to the current error term, E(t). As in the case of autoregressive models, the moving average models can be extended to higher order structures covering different combinations and moving average lengths.

ARIMA methodology also allows models to be built that incorporate both autoregressive and moving average parameters together. These models are often referred to as "mixed models". Although this makes for a more complicated forecasting tool, the structure may indeed simulate the series better and produce a more accurate forecast. Pure models imply that the structure consists only of AR or MA parameters - not both.

The models developed by this approach are usually called ARIMA models because they use a combination of autoregressive (AR), integration (I) - referring to the reverse process of differencing to produce the forecast, and moving average (MA) operations. An ARIMA model is usually stated as ARIMA(p,d,q). This represents the order of the autoregressive components (p), the number of differencing operators (d), and the highest order of the moving average term. For example, ARIMA(2,1,1) means that you have a second order autoregressive model with a first order moving average component whose series has been differenced once to induce stationarity.