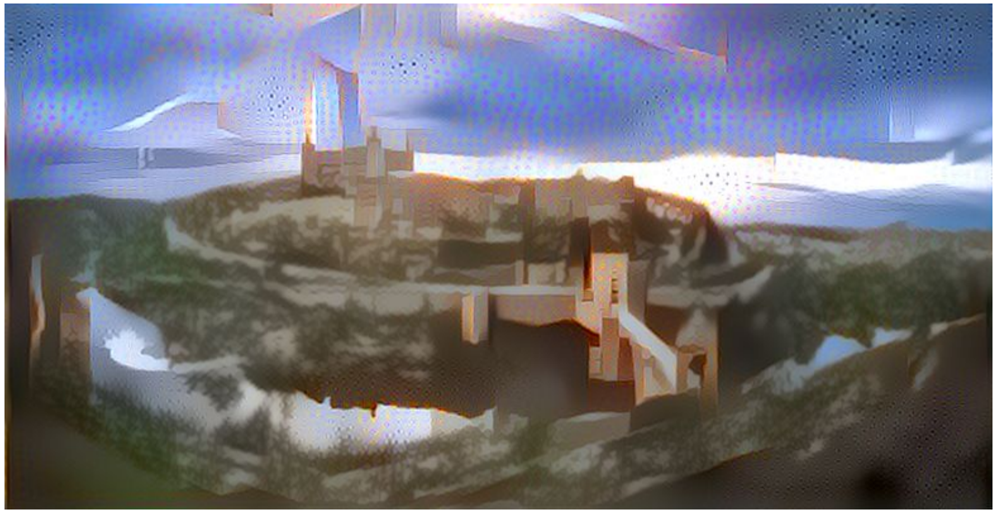
CLASSIC MACHINE LEARNING VS DEEP LEARNING



Afi Escuela de Finanzas

Analytical Index

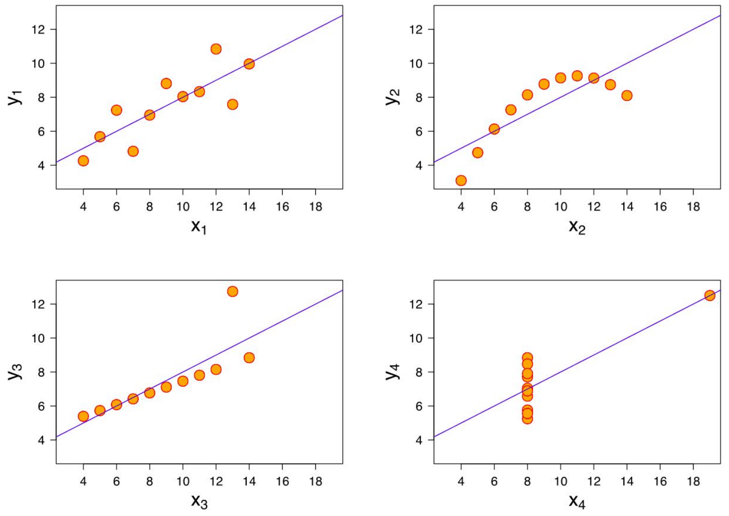
Classic ML Algorithms Review

Afi Escuela de Finanzas

K-Nearest Neighbors

Pros:

- It is one of the simplest algorithms to understand.
- Good interpretability
- Feature importance is generated at the time model building. With the help of hyperparameter lamba, you can handle features selection hence we can achieve dimensionality reduction
- It works in most of the cases. Even when it doesn't fit the data exactly, we can use it to find the nature of the relationship between the two variables.



Cons:

- The algorithm assumes data is normally distributed, but most of the times it is not.
- Before building model multicollinearity should be avoided.
- Prone to outliers
- For sure we can get rid of those three with some data treatment!
- By its definition, linear regression only models relationships between dependent and independent variables that are linear. It assumes there is a straight-line relationship between them which is incorrect sometimes.
- If we have a number of parameters higher than the number of samples available then the model starts to model the noise rather than the relationship between the variables.
- Look at the Anscombe quartet

Linear Regression is, by definition, a linear approach to modeling the relationship between a dependent variable and one or more independent variables. In another words is the affine space of dimension 1 (i.e: a straight line) in the direction of the dependent variable that minimizes the average distance (l2) to the data sample.

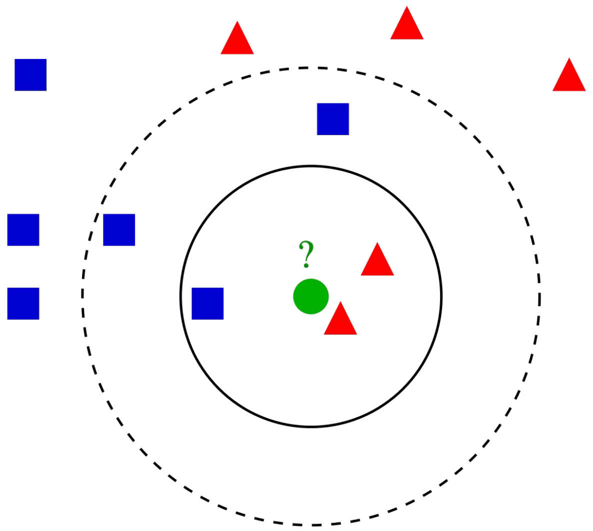
K-Nearest Neighbors

Pros:

- K-NN is pretty intuitive and simple:
- K-NN has no assumptions: K-NN is a non-parametric algorithm which means there are no assumptions to be met to implement K-NN.
- No Training Step: K-NN does not explicitly build any model, it simply tags the new data entry based learning from historical data. New data entry would be tagged with majority class in the nearest neighbor.
- It constantly evolves
- Very easy to implement for multi-class problem
- One single Hyper Parameter
- Variety of distance criteria to be choose from: Euclidean Distance , Hamming Distance , Manhattan Distance , Minkowski Distance ...

Cons:

- K-NN is a slow algorithm: K-NN might be very easy to implement but as dataset grows efficiency or speed of algorithm declines very fast.
- Curse of Dimensionality: KNN works well with small number of input variables but as the numbers of variables grow K-NN algorithm struggles to predict the output of new data point.
- K-NN needs homogeneous features: If you decide to build k-NN using a common distance, like Euclidean or Manhattan distances, it is completely necessary that features have the same scale
- Imbalanced data causes problems: k-NN doesn't perform well on imbalanced data. If we consider two classes,
- Outlier sensitivity: K-NN algorithm is very sensitive to outliers as it simply chose the neighbors based on distance criteria.
- Missing Value treatment: K-NN inherently has no capability of dealing with missing value problem.



K- Nearest Neighbors or also known as K-NN belong to the family of supervised machine learning algorithms which means we use labeled (Target Variable) dataset to predict the class of new data point. The K-NN algorithm is a robust classifier which is often used as a benchmark for more complex classifiers such as Neural Network or Support vector machine (SVM).

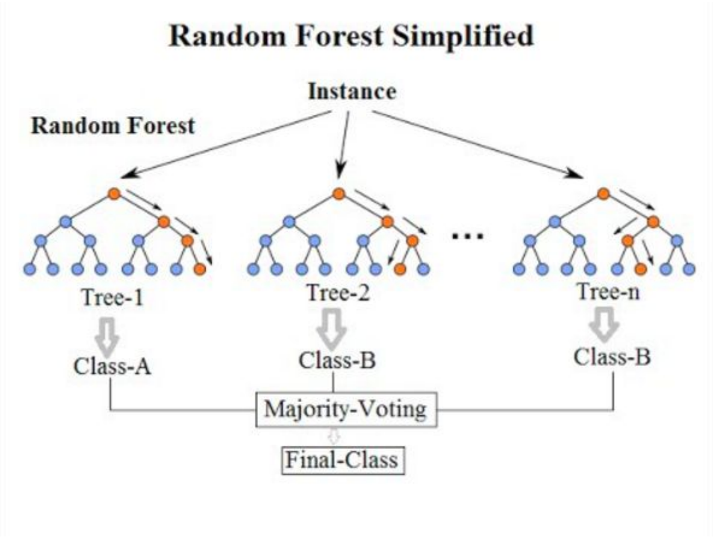
Random Forests

Pros:

- RF can handle large data sets with higher dimensionality. It can handle thousands of input variables and identity most significant variables so it is even considered as one of the dimensionality reduction method.
- It can output importance of variable, which can be a very handy feature.
- It has an effective method for estimating missing data and maintains accuracy when large proportion of the data are missing.
- It has methods for balancing errors in data sets where classes are imbalanced.
- Random forest involves sampling of the input data with replacement called as bootstrap sampling.

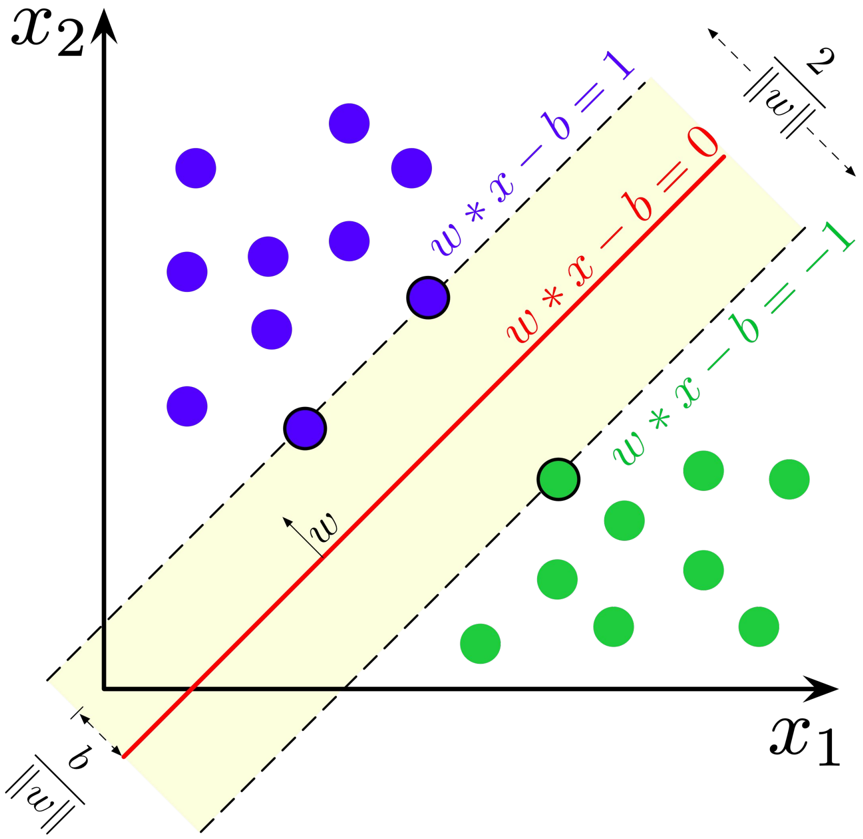
Cons:

- RF surely does a good job at classification but not as for regression problem as it does not gives precise continuous nature prediction. In case of regression, it doesn't predict beyond the range in the training data, and that they may over fit data sets that are particularly noisy.
- Random forest can feel like a black box approach for a statistical modelers we have very little control on what the model does.



A random forest consists of multiple random decision trees. Two types of randomnesses are built into the trees. First, each tree is built on a random sample from the original data. Second, at each tree node, a subset of features are randomly selected to generate the best split.

Supported Vector Machines



The main idea behind a SVM is to find a hypersurface that helps us separate the data. With that in mind, we can use different kernels, which will define the kind of surface we are building.

When to use a SVM?
SVMs are good for image analysis tasks, such as image classification and handwritten digit recognition. Also SVM are very effective in text-mining tasks, particularly due to its effectiveness in dealing with high-dimensional data. For example, it is used for detecting spam, text category assignment, and sentiment analysis.
Another application of SVM is in Gene Expression data classification, again, because of its power in high dimensional data classification.
SVM can also be used for other types of machine learning problems, such as regression, outlier detection, and clustering.

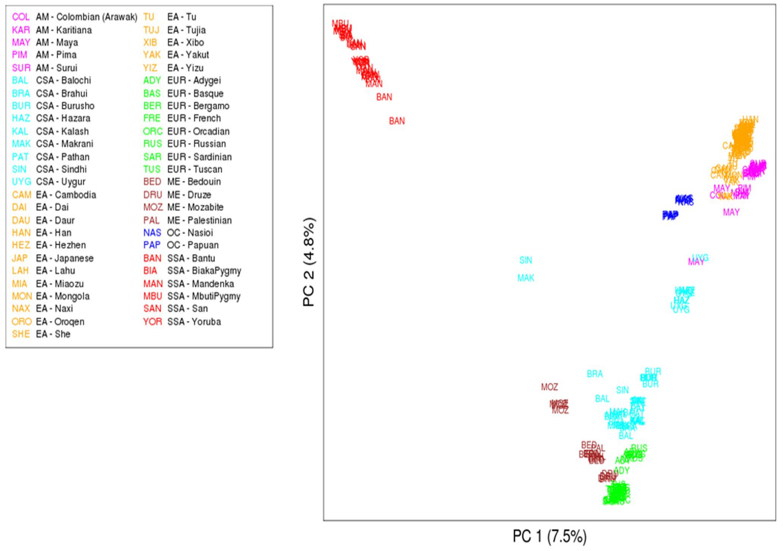
Pros:
- They're accurate in high dimensional spaces;
- They use a subset of training points in the decision function (called support vectors), so it's also memory efficient.
- Capable of learning complex functions/hypersurfaces to separate the data

Cons:
- The algorithm is prone for over-fitting. Be careful with that
- SVMs do not directly provide probability estimates, which are desirable in most classification problems.
- SVMs are not very efficient computationally, if your dataset is very big, such as when you have more than one thousand rows.

Principal Component Analysis

Pros:

- Removes Correlated Features: After implementing the PCA on your dataset, all the Principal Components are independent of one another. There is no correlation among them.
- Improves Algorithm Performance: PCA is a very common way to speed up your Machine Learning algorithm by getting rid of useless variables which don’t contribute in any decision making. The training time of the algorithms reduces significantly with less number of features.
- Reduces Overfitting: PCA helps in overcoming the overfitting issue by reducing the number of features.
- Improves Visualization: It is very hard to visualize and understand the data in high dimensions. PCA transforms a high dimensional data to low dimensional data.



Cons:

- Independent variables become less interpretable: After implementing PCA on the dataset, your original features will turn into Principal Components: linear combination of your original features, which are not as readable and interpretable as original features.
- Data standardization is must before PCA: You must standardize your data before implementing PCA, otherwise PCA will not be able to find the optimal Principal Components.
- Beware of categorical features: All the categorical features are required to be converted into numerical features before PCA can be applied.
- Information Loss: Although Principal Components try to cover maximum variance among the features in a dataset, if we don’t select the number of Principal Components with care, it may miss some info.

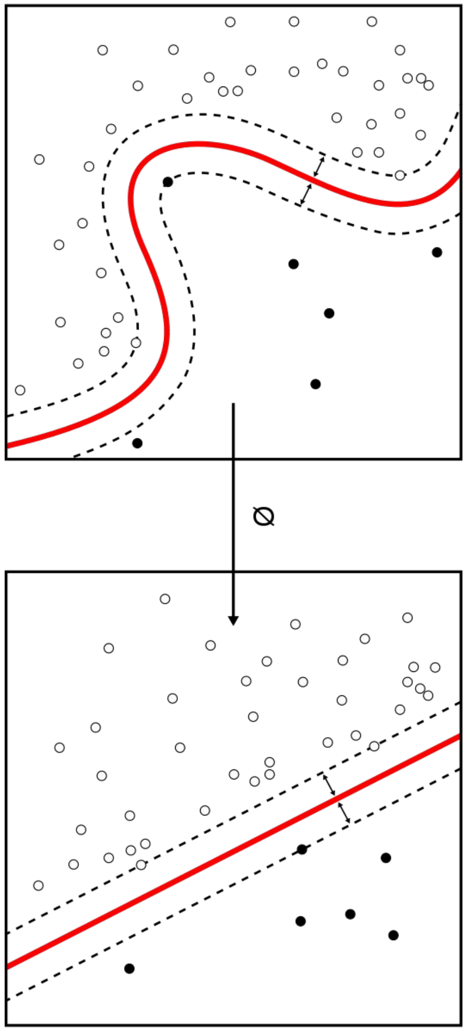
Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables (entities each of which takes on various numerical values) into a set of values of linearly uncorrelated variables called principal components. This transformation is defined in such a way that the first principal component has the largest possible variance (that is, accounts for as much of the variability in the data as possible), and each succeeding component in turn has the highest variance possible under the constraint that it is orthogonal to the preceding components. The resulting vectors (each being a linear combination of the variables and containing n observations) are an uncorrelated orthogonal basis set. PCA is sensitive to the relative scaling of the original variables.

Some keys on Learning with a Machine

Afi Escuela de Finanzas

Feature Engineering

- A feature is an individual measurable property or characteristic of a phenomenon being observed. The concept of a “feature” is related to that of an explanatory variable, which is used in Machine learning Modelling. Feature vectors combine all the features for a single row into a numerical vector.
- Part of the art of choosing features is to pick a minimum set of independent variables that explain the problem. If two variables are highly correlated, either they need to be combined into a single feature, or one should be dropped. Sometimes people perform principal component analysis to convert correlated variables into a set of linearly uncorrelated variables.
- But creating new features with the amount of information the model needs is not easy. It can be considered an art. Requires some business knowledge in order to understand what combinations can affect the model results.
- It also requires the knowledge of the model, because some models will not be able to understand some features, while others can get a high benefit from them.
- It also requires feature selection, due to the model performances, its business value or its explainability. On these terms some simple Classic ML algorithms can get higher benefit than more sophisticated DL ones.
- Feature engineering is strictly (in terms of performance improvement) needed on classic ML algorithms, while it is not on DL. The DL layers do learn the features they consider most important for the outcome of the model, both linear and non-linears, depending on the model architecture.
- What about adding Feature Engineering to DL models? The basic theory would say no, but if you find some features you can add to the classifier/regressor, it would probably help. Keep in mind that these additions should be made after the model backbone does its part.



Time and Hardware

Machine learning algorithms may be more desirable if you need quicker results. They are faster to train and require less computational power. The number of features and observations will be the key factors that affect training time. Engineers applying machine learning should expect to spend a majority of their time developing and evaluating features to improve model accuracy.

Deep learning models will take time to train. Pretrained networks and public datasets can shorten training through transfer learning, but sometimes these can be complicated to implement. In general, deep learning algorithms can take anywhere from a minute to a few weeks to train depending on your hardware and computing power. Engineers applying deep learning should expect to spend a majority of their time training models and making modifications to the architecture of their deep neural network.

Machine learning algorithms require less computational power. For example, desktop CPUs are sufficient for training these models.

For deep learning models, specialized hardware is typically required due to the higher memory and compute requirements. Specialized hardware is also appropriate because the operations performed within a deep neural network, such as convolutions, lend themselves well to the parallel architecture of the GPU.

Deep learning models take significant computing power. They should be considered if GPUs are available, or if there is time to run trainings on a CPU (which will take significantly longer). Usually, a Deep Learning algorithm takes a long time to train due to large number of parameters. Popular ResNet algorithm takes about two weeks to train completely from scratch. At test time, Deep Learning algorithm takes much less time to run. Whereas, if you compare it with k-nearest neighbors, test time increases on increasing the size of data. Although this is not applicable on all machine learning algorithms, as some of them have small testing times too.

Data and Complexity

In traditional Machine learning techniques, most of the applied features need to be identified by an domain expert in order to reduce the complexity of the data and make patterns more visible to learning algorithms to work. The biggest advantage Deep Learning algorithms as discussed before are that they try to learn high-level features from data in an incremental manner. This eliminates the need of domain expertise and hard core feature extraction.

- **Problem solving approach**. Deep Learning techniques tend to solve the problem end to end, where as Machine learning techniques need the problem statements to break down to different parts to be solved first and then their results to be combine at final stage. For example for a multiple object detection problem, Deep Learning techniques like Yolo net take the image as input and provide the location and name of objects at output. But in usual Machine Learning algorithms like SVM, a bounding box object detection algorithm is required first to identify all possible objects to have the HOG as input to the learning algorithm in order to recognize relevant objects.

- **Interpretability**. This is the main issue why many sectors using other Machine Learning techniques over Deep Learning. DL algorithms are hard to interpret. Indeed mathematically you can find out which nodes of a deep neural network were activated, but we don’t know what there neurons were supposed to model and what these layers of neurons were doing collectively. So we fail to interpret the results. Which is not in case of Machine Learning algorithms like decision trees, logistic regression etc.

So... Then what?

Both

In general, classical (non-deep) machine learning algorithms train and predict much faster than deep learning algorithms; one or more CPUs will often be sufficient to train a classical model.

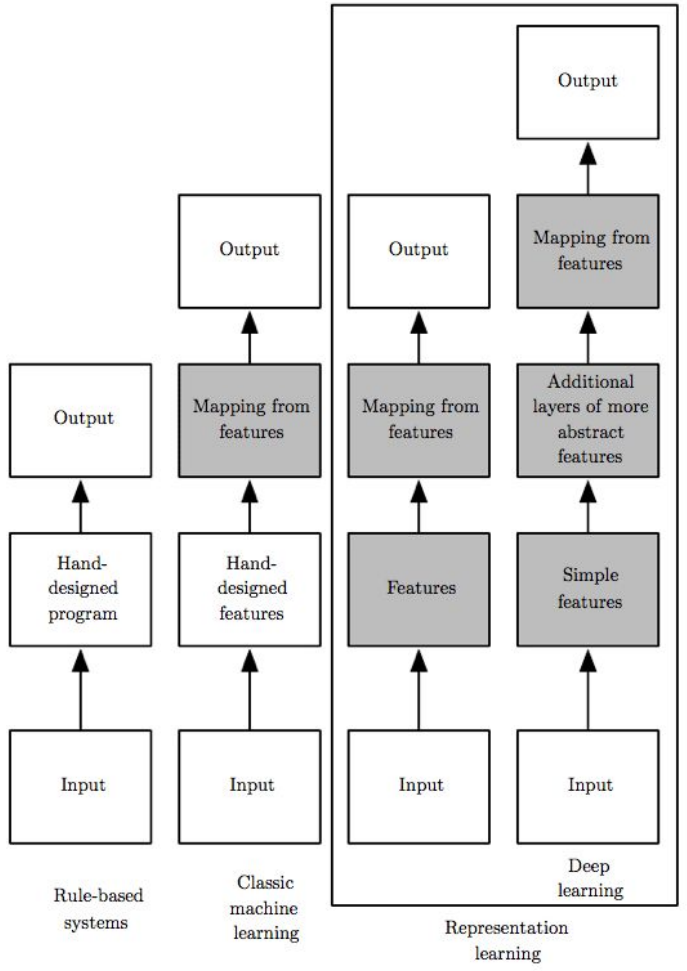
Deep learning models often need hardware accelerators such as GPUs, TPUs, or FPGAs for training, and
also for deployment at scale; without them, the models would take months to train.

For many problems, some classical machine learning algorithms will produce a “good enough” model. For other problems, classical machine learning algorithms have not worked terribly well in the past.

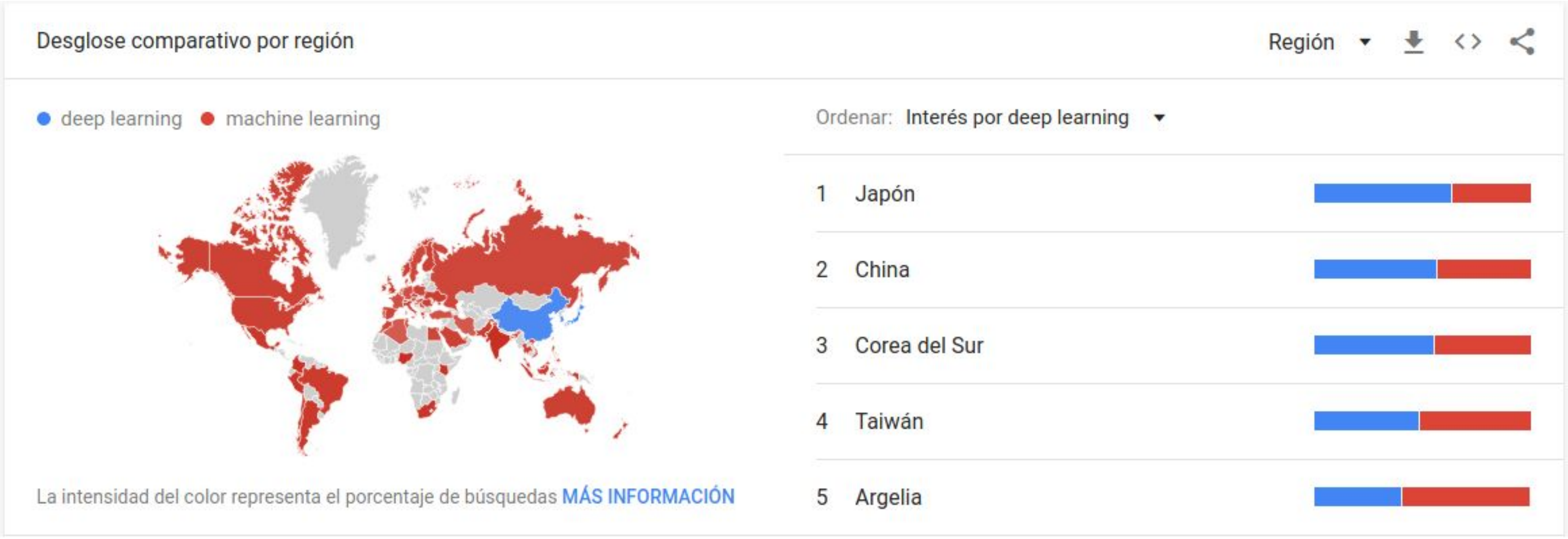
One area that is usually attacked with deep learning is natural language processing, which encompasses language translation, automatic summarization, co-reference resolution, discourse analysis, morphological segmentation, named entity recognition, natural language generation, natural language understanding, part-of-speech tagging, sentiment analysis, and speech recognition.

Another prime area for deep learning is image classification, which includes image classification with localization, object detection, object segmentation, image style transfer, image colorization, image reconstruction, image super-resolution, and image synthesis.

In addition, deep learning has been used successfully to predict how molecules will interact in order to help pharmaceutical companies design new drugs, to search for subatomic particles, and to automatically parse microscope images used to construct a three-dimensional map of the human brain.



Conclusion


Desglose comparativo por región

Región

● deep learning ● machine learning

Ordenar: Interés por deep learning

1	Japón
2	China
3	Corea del Sur
4	Taiwán
5	Argelia

La intensidad del color representa el porcentaje de búsquedas MÁS INFORMACIÓN

Some takeouts

- Deep Learning outperform other techniques if the data size is large. But with small data size, traditional Machine Learning algorithms are preferable… Unless we can apply Transfer Learning
- Deep Learning techniques need to have high end infrastructure to train in reasonable time.
- When there is lack of domain understanding for feature introspection, Deep Learning techniques outshines others as you have to worry less about feature engineering.
- Deep Learning really shines when it comes to complex problems such as image classification, natural language processing, and speech recognition.

Afi Escuela de Finanzas