10 Best Machine Learning Algorithms

Originally published at https://www.unite.ai/ten-best-machine-learning-algorithms/

Though we're living through a time of extraordinary innovation in GPU-accelerated machine learning, the latest research papers frequently (and prominently) feature algorithms that are decades, in certain cases 70 years old.

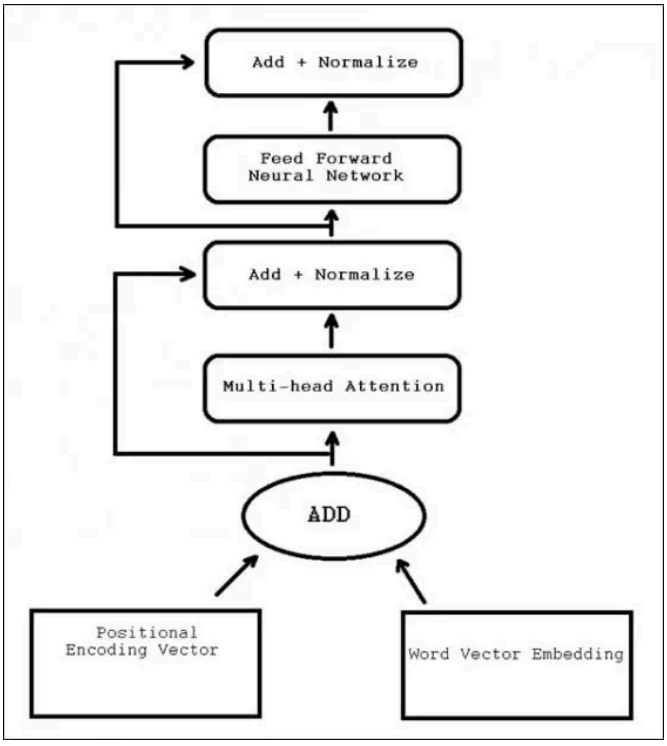
Some might contend that many of these older methods fall into the camp of 'statistical analysis' rather than machine learning, and prefer to date the advent of the sector back only so far as 1957, with the invention of the Perceptron.

Given the extent to which these older algorithms support and are enmeshed in the latest trends and headline-grabbing developments in machine learning, it's a contestable stance. So let's take a look at some of the 'classic' building blocks underpinning the latest innovations, as well as some newer entries that are making an early bid for the AI hall of fame.

1: Transformers

In 2017 Google Research led a research collaboration culminating in the paper *Attention Is All You Need*. The work outlined a novel architecture that promoted attention mechanisms from 'piping' in encoder/decoder and recurrent network models to a central transformational technology in their own right.

The approach was dubbed *Transformer*, and has since become a revolutionary methodology in Natural Language Processing (NLP), powering, amongst many other examples, the autoregressive language model and AI poster-child GPT-3.

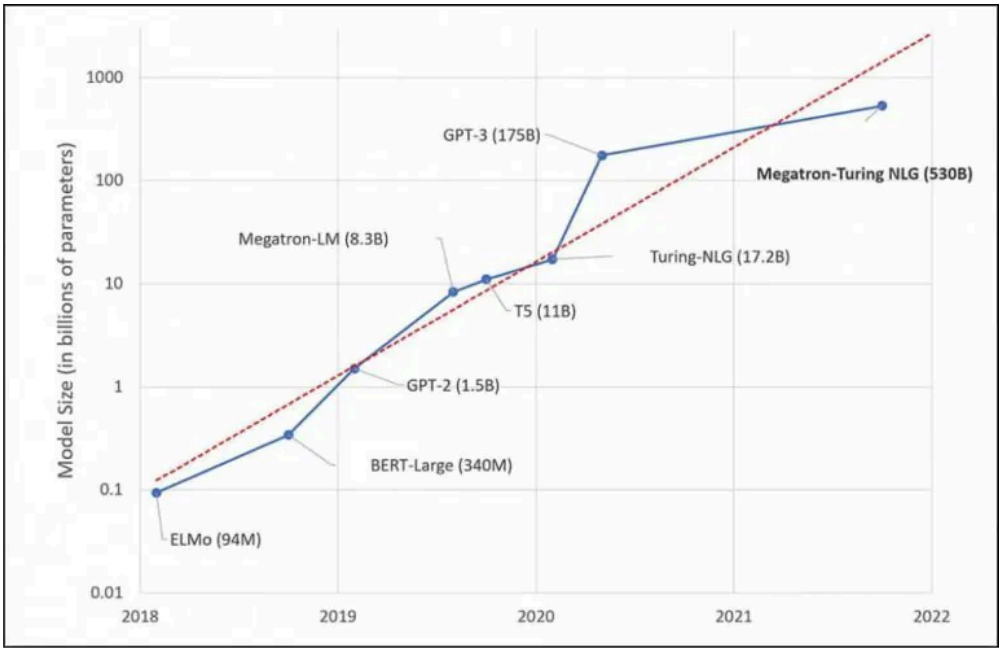


Transformers elegantly solved the problem of sequence transduction, also called 'transformation', which is occupied with the processing of input sequences into output sequences. A transformer also receives and manages data in a continuous manner, rather than in sequential batches, allowing a 'persistence of memory' which RNN architectures are not designed to obtain. For a more detailed overview of transformers, take a look at our reference article.

In contrast to the Recurrent Neural Networks (RNNs) that had begun to dominate ML research in the CUDA era, Transformer architecture could also be easily parallelized, opening the way to productively address a far larger corpus of data than RNNs.

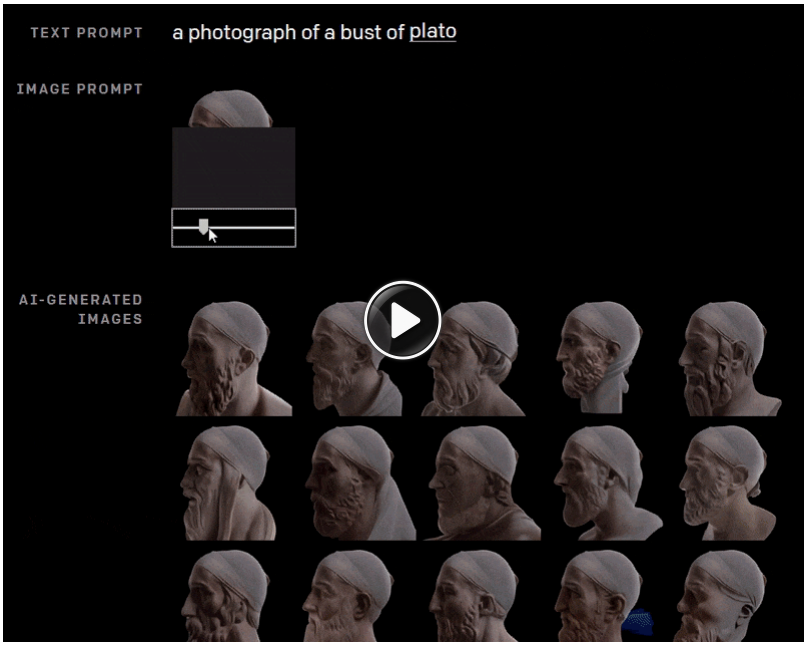
Popular Usage

Transformers captured the public imagination in 2020 with the release of OpenAI's GPT-3, which boasted a then record-breaking 175 billion parameters. This apparently staggering achievement was eventually overshadowed by later projects, such as the 2021 release of Microsoft's Megatron-Turing NLG 530B, which (as the name suggests) features over 530 billion parameters.



A timeline of hyperscale Transformer NLP projects. Source: Microsoft

Transformer architecture has also crossed over from NLP to computer vision, powering a new generation of image synthesis frameworks such as OpenAI's CLIP and DALL-E, which use text>image domain mapping to finish incomplete images and synthesize novel images from trained domains, among a growing number of related applications.



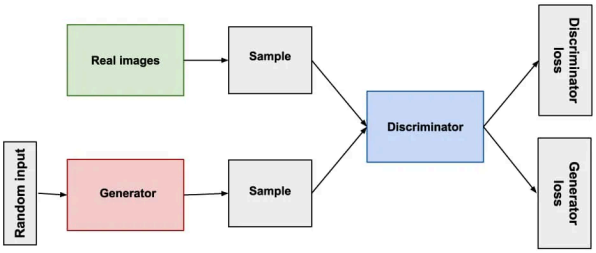
CLICK TO VIEW ANIMATED GIF

DALL-E attempts to complete a partial image of a bust of Plato. Source: https://openai.com/blog/dall-e/

2: Generative Adversarial Networks (GANs)

Though transformers have gained extraordinary media coverage through the release and adoption of GPT-3, the Generative Adversarial Network (GAN) has become a recognizable brand in its own right, and may eventually join *deepfake* as a verb.

First proposed in 2014 and primarily used for image synthesis, a Generative Adversarial Network architecture is composed of a *Generator* and a *Discriminator*. The Generator cycles through thousands of images in a dataset, iteratively attempting to reconstruct them. For each attempt, the Discriminator grades the Generator's work, and sends the Generator back to do better, but without any insight into the way that the previous reconstruction erred.



Source: https://developers.google.com/machine-learning/gan/gan_structure

This forces the Generator to explore a multiplicity of avenues, instead of following the potential blind alleys that would have resulted if the Discriminator had told it where it was going wrong (see #8 below). By the time the training is over, the Generator has a detailed and comprehensive map of relationships between points in the dataset.

CLICK TO VIEW ANIMATED GIF

From the paper Improving GAN Equilibrium by Raising Spatial Awareness*: a novel framework cycles through the sometimes-mysterious latent space of a GAN, providing responsive instrumentality for an image synthesis architecture.* Source: https://genforce.github.io/eqgan/

By analogy, this is the difference between learning a single humdrum commute to central London, or painstakingly acquiring The Knowledge.

The result is a high-level collection of features in the latent space of the trained model. The semantic indicator for a high level feature could be 'person', whilst a descent through specificity related to the feature may unearth other learned characteristics, such as 'male' and 'female'. At lower levels the sub-features can break down to, 'blonde', 'Caucasian', et al.

Entanglement is a notable issue in the latent space of GANs and encoder/decoder frameworks: is the smile on a GAN-generated female face an entangled feature of her 'identity' in the latent space, or is it a parallel branch?

GAN-generated faces from thispersondoesnotexist. Source: https://this-person-does-not-exist.com/en

The past couple of years have brought forth a growing number of new research initiatives in this respect, perhaps paving the way for feature-level, Photoshop-style editing for the latent space of a GAN, but at the moment, many transformations are effectively 'all or nothing' packages. Notably, NVIDIA's EditGAN release of late 2021 achieves a high level of interpretability in the latent space by using semantic segmentation masks.

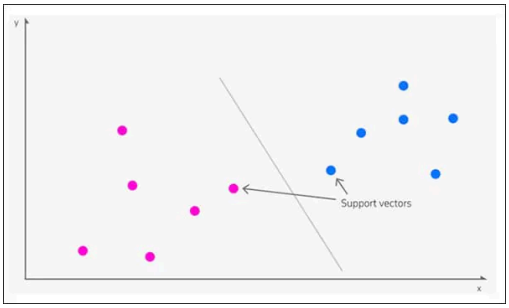
Popular Usage

Beside their (actually fairly limited) involvement in popular deepfake videos, image/video-centric GANs have proliferated over the last four years, enthralling researchers and the public alike. Keeping up with the dizzying rate and frequency of new releases is a challenge, though the GitHub repository Awesome GAN Applications aims to provide a comprehensive list.

Generative Adversarial Networks can in theory derive features from any well-framed domain, including text.

3: SVM

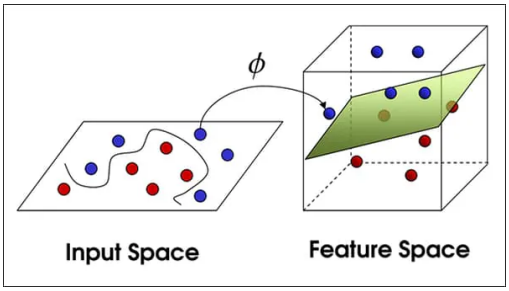
Originated in 1963, Support Vector Machine (SVM) is a core algorithm that crops up frequently in new research. Under SVM, vectors map the relative disposition of data points in a dataset, while *support* vectors delineate the boundaries between different groups, features, or traits.



Support vectors define the boundaries between groups. Source: https://www.kdnuggets.com/2016/07/support-vector-machines-simple-explanation.html

The derived boundary is called a *hyperplane*.

At low feature levels, the SVM is *two-dimensional* (image above), but where there's a higher recognized number of groups or types, it becomes *three-dimensional*.



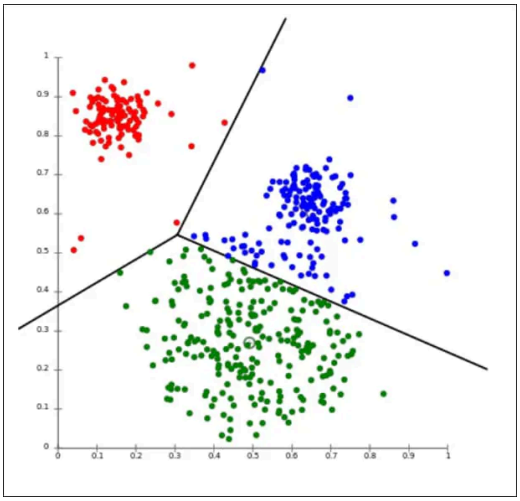
A deeper array of points and groups necessitates a three-dimensional SVM. Source: https://cml.rhul.ac.uk/svm.html

Popular Usage

Since support Vector Machines can effectively and agnostically address high-dimensional data of many kinds, they crop up widely across a variety of machine learning sectors, including deepfake detection, image classification, hate speech classification, DNA analysis and population structure prediction, among many others.

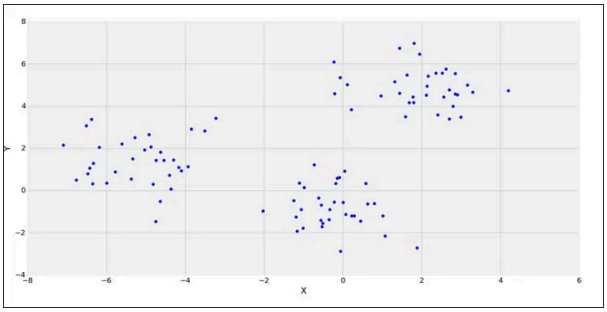
4: K-Means Clustering

Clustering in general is an unsupervised learning approach that seeks to categorize data points through density estimation, creating a map of the distribution of the data being studied.



K-Means clustering divines segments, groups and communities in data. Source: https://aws.amazon.com/blogs/machine-learning/k-means-clustering-with-amazon-sagemaker/

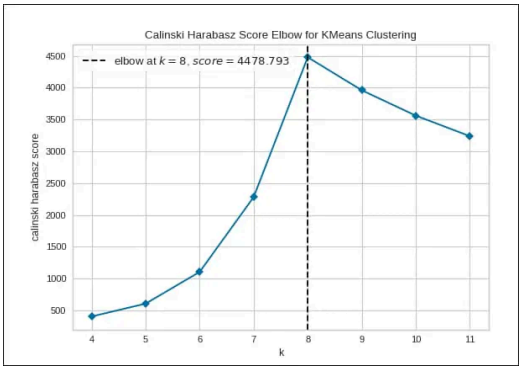
K-Means Clustering has become the most popular implementation of this approach, shepherding data points into distinctive 'K Groups', which may indicate demographic sectors, online communities, or any other possible secret aggregation waiting to be discovered in raw statistical data.



Clusters form in K-Means analysis. Source: https://www.geeksforgeeks.org/ml-determine-the-optimal-value-of-k-in-k-means-clustering/

The K value itself is the determinant factor in the utility of the process, and in establishing an optimal value for a cluster. Initially, the K value is randomly assigned, and its features and vector characteristics compared to its neighbors. Those neighbors that most closely resemble the data point with the randomly assigned value get assigned to its cluster iteratively until the data has yielded all the groupings that the process permits.

The plot for the squared error, or 'cost' of differing values among the clusters will reveal an elbow point for the data:



The 'elbow point' in a cluster graph. Source: https://www.scikit-yb.org/en/latest/api/cluster/elbow.html

The elbow point is similar in concept to the way that loss flattens out to diminishing returns at the end of a training session for a dataset. It represents the point at which no further distinctions between groups is going to become apparent, indicating the moment to move on to subsequent phases in the data pipeline, or else to report findings.

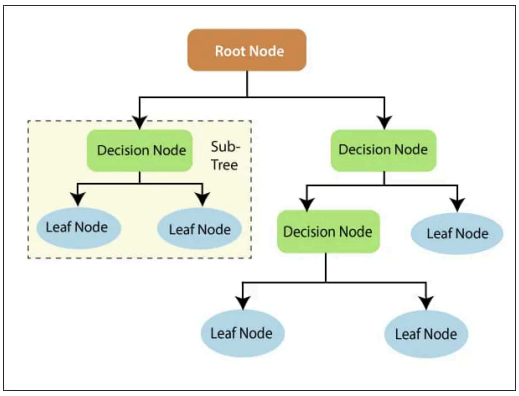
Popular Usage

K-Means Clustering, for obvious reasons, is a primary technology in customer analysis, since it offers a clear and explainable methodology to translate large quantities of commercial records into demographic insights and ‘leads’.

Outside of this application, K-Means Clustering is also employed for landslide prediction, medical image segmentation, image synthesis with GANs, document classification, and city planning, among many other potential and actual uses.

5: Random Forest

Random Forest is an ensemble learning method that averages the result from an array of decision trees to establish an overall prediction for the outcome.



Source: https://www.tutorialandexample.com/wp-content/uploads/2019/10/Decision-Trees-Root-Node.png

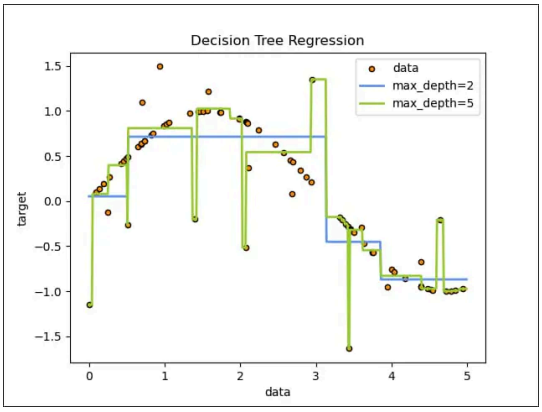
If you’ve researched it even as little as watching the *Back to the Future* trilogy, a decision tree itself is fairly easy to conceptualize: a number of paths lie before you, and each path branches out to a new outcome which in turn contains further possible paths.

In reinforcement learning, you might retreat from a path and start again from an earlier stance, whereas decision trees commit to their journeys.

Thus the Random Forest algorithm is essentially spread-betting for decisions. The algorithm is called ‘random’ because it makes *ad hoc* selections and observations in order to understand the *median* sum of the results from the decision tree array.

Since it takes into account a multiplicity of factors, a Random Forest approach can be more difficult to convert into meaningful graphs than a decision tree, but is likely to be notably more productive.

Decision trees are subject to overfitting, where the results obtained are data-specific and not likely to generalize. Random Forest’s arbitrary selection of data points combats this tendency, drilling through to meaningful and useful representative trends in the data.



Decision tree regression. Source: https://scikit-learn.org/stable/auto_examples/tree/plot_tree_regression.html

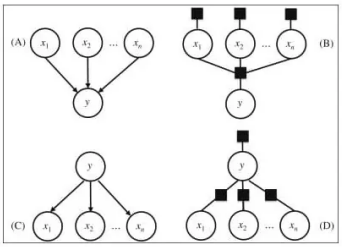
Popular Usage

As with many of the algorithms in this list, Random Forest typically operates as an 'early' sorter and filter of data, and as such consistently crops up in new research papers. Some examples of Random Forest usage include Magnetic Resonance Image Synthesis, Bitcoin price prediction, census segmentation, text classification and credit card fraud detection.

Since Random Forest is a low-level algorithm in machine learning architectures, it can also contribute to the performance of other low-level methods, as well as visualization algorithms, including Inductive Clustering, Feature Transformations, classification of text documents using sparse features, and displaying Pipelines.

6: Naive Bayes

Coupled with density estimation (see *4*, above), a naive Bayes classifier is a powerful but relatively lightweight algorithm capable of estimating probabilities based on the calculated features of data.



Feature relationships in a naive Bayes classifier. Source: https://www.sciencedirect.com/topics/computer-science/naive-bayes-model

The term 'naïve' refers to the assumption in Bayes' theorem that features are unrelated, known as *conditional independence*. If you adopt this standpoint, walking and talking like a duck aren't enough to establish that we're dealing with a duck, and no 'obvious' assumptions are prematurely adopted.

This level of academic and investigative rigor would be overkill where 'common sense' is available, but is a valuable standard when traversing the many ambiguities and potentially unrelated correlations that may exist in a machine learning dataset.

In an original Bayesian network, features are subject to scoring functions, including minimal description length and Bayesian scoring, which can impose restrictions on the data in terms of the estimated connections found between the data points, and the direction in which these connections flow.

A naive Bayes classifier, conversely, operates by assuming that the features of a given object are independent, subsequently using Bayes' theorem to calculate the probability of a given object, based on its features.

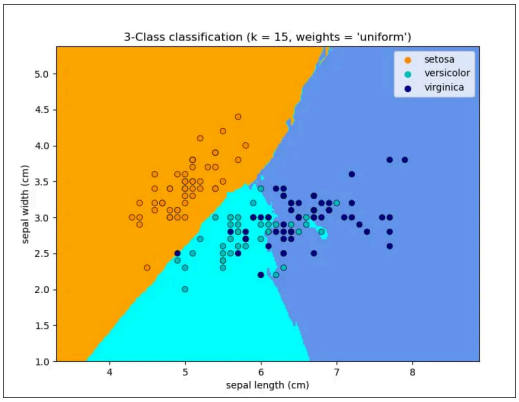
Popular Usage

Naive Bayes filters are well-represented in disease prediction and document categorization, spam filtering, sentiment classification, recommender systems, and fraud detection, among other applications.

7: K- Nearest Neighbors (KNN)

First proposed by the US Air Force School of Aviation Medicine in 1951, and having to accommodate itself to the state-of-the-art of mid-20th century computing hardware, K-Nearest Neighbors (KNN) is a lean algorithm that still features prominently across academic papers and private sector machine learning research initiatives.

KNN has been called 'the lazy learner', since it exhaustively scans a dataset in order to evaluate the relationships between data points, rather than requiring the training of a full-fledged machine learning model.



A KNN grouping. Source: https://scikit-learn.org/stable/modules/neighbors.html

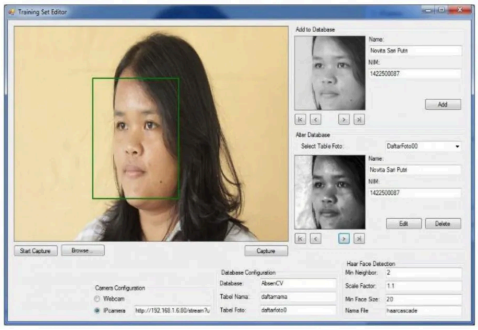
Though KNN is architecturally slender, its systematic approach does place a notable demand on read/write operations, and its use in very large datasets can be problematic without adjunct technologies such as Principal Component Analysis (PCA), which can transform complex and high volume datasets into representative groupings that KNN can traverse with less effort.

A recent study evaluated the effectiveness and economy of a number of algorithms tasked to predict whether an employee will leave a company, finding that the septuagenarian KNN remained superior to more modern contenders in terms of accuracy and predictive effectiveness.

Popular Usage

For all its popular simplicity of concept and execution, KNN is not stuck in the 1950s – it's been adapted into a more DNN-focused approach in a 2018 proposal by Pennsylvania State University, and remains a central early-stage process (or post-processing analytical tool) in many far more complex machine learning frameworks.

In various configurations, KNN has been used or for online signature verification, image classification, text mining, crop prediction, and facial recognition, besides other applications and incorporations.

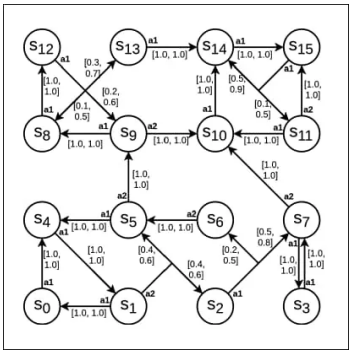


A KNN-based facial recognition system in training. Source: https://pdfs.semanticscholar.org/6f3d/d4c5ffeb3ce74bf573428616869444490f513.pdf

8: Markov Decision Process (MDP)

A mathematical framework introduced by American mathematician Richard Bellman in 1957, The Markov Decision Process (MDP) is one of the most basic blocks of reinforcement learning architectures. A conceptual algorithm in its own right, it has been adapted into a great number of other algorithms, and recurs frequently in the current crop of AI/ML research.

MDP explores a data environment by using its evaluation of its current state (i.e. 'where' it is in the data) to decide which node of the data to explore next.



Source:
https://www.sciencedirect.com/science/article/abs/pii/S0888613X18304420

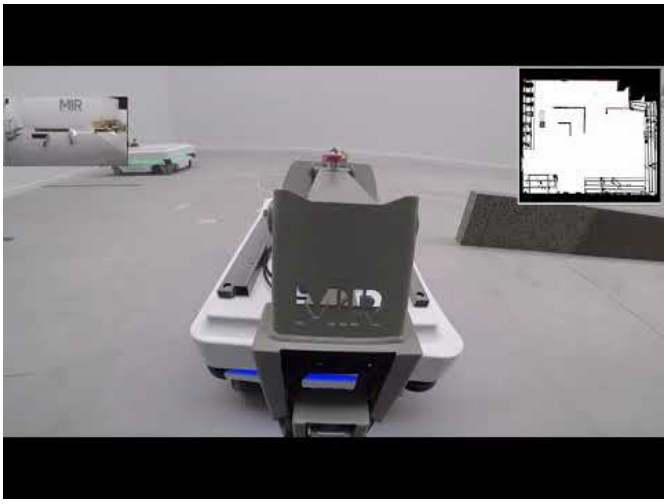
A basic Markov Decision Process will prioritize near-term advantage over more desirable long-term objectives. For this reason, it is usually embedded into the context of a more comprehensive policy architecture in reinforcement learning, and is often subject to limiting factors such as discounted reward, and other modifying environmental variables that will prevent it from rushing to an immediate goal without consideration of the broader desired outcome.

Popular Usage

MDP's low-level concept is widespread in both research and active deployments of machine learning. It's been proposed for IoT security defense systems, fish harvesting, and market forecasting.

Besides its obvious applicability to chess and other strictly sequential games, MDP is also a natural contender for the procedural training of robotics systems, as we can see in the video below.

Global Planner using a Markov Decision Process - Mobile Industrial Robotics



9: Term Frequency-Inverse Document Frequency

Term Frequency (TF) divides the number of times a word appears in a document by the total number of words in that document. Thus the word *seal* appearing once in a thousand-word article has a term frequency of 0.001. By itself, TF is largely useless as an indicator of term importance, due to the fact that meaningless articles (such as *a*, *and*, *the*, and *it*) predominate.

To obtain a meaningful value for a term, Inverse Document Frequency (IDF) calculates the TF of a word across multiple documents in a dataset, assigning low rating to very high-frequency stopwords, such as articles. The resulting feature vectors are normalized to whole values, with each word assigned an appropriate weight.

Total Number of Documents	100,000,000
Term of Interest	Number of Documents Containing that Term
a	100,000,000
boat	1,000,000
mobile	100,000
mobilegeddon	1,000

TF-IDF weights the relevance of terms based on frequency across a number of documents, with rarer occurrence an indicator of salience. Source: https://moz.com/blog/inverse-document-frequency-and-the-importance-of-uniqueness

Though this approach prevents semantically important words from being lost as outliers, inverting the frequency weight does not automatically mean that a low-frequency term is *not* an outlier, because some things are rare *and* worthless. Therefore a low-frequency term will need to prove its value in the wider architectural context by featuring (even at a low frequency per document) in a number of documents in the dataset.

Despite its age, TF-IDF is a powerful and popular method for initial filtering passes in Natural Language Processing frameworks.

Popular Usage

Because TF-IDF has played at least some part in the development of Google's largely occult PageRank algorithm over the last twenty years, it has become very widely adopted as a manipulative SEO tactic, in spite of John Mueller's 2019 disavowal of its importance to search results.

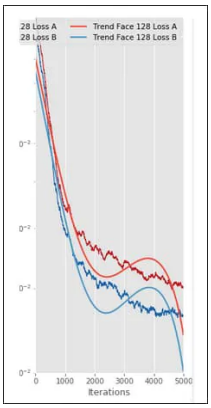
Due to the secrecy around PageRank, there is no clear evidence that TF-IDF is *not* currently an effective tactic for rising in Google's rankings. Incendiary discussion among IT professionals lately indicates a popular understanding, correct or not, that term abuse may still result in improved SEO placement (though additional accusations of monopoly abuse and excessive advertising blur the confines of this theory).

10: Stochastic Gradient Descent

Stochastic Gradient Descent (SGD) is an increasingly popular method for optimizing the training of machine learning models.

Gradient Descent itself is a method of optimizing and subsequently quantifying the improvement that a model is making during training.

In this sense, 'gradient' indicates a slope downwards (rather than a color-based gradation, see image below), where the highest point of the 'hill', on the left, represents the beginning of the training process. At this stage the model has not yet seen the entirety of the data even once, and has not learned enough about relationships between the data to produce effective transformations.



A gradient descent on a FaceSwap training session. We can see that the training has plateaued for some time in the second half, but has eventually recovered its way down the gradient towards an acceptable convergence.

The lowest point, on the right, represents convergence (the point at which the model is as effective as it is ever going to get under the imposed constraints and settings).

The gradient acts as a record and predictor for the disparity between the error rate (how accurately the model has currently mapped the data relationships) and the weights (the settings that influence the way in which the model will learn).

This record of progress can be used to inform a learning rate schedule, an automatic process that tells the architecture to become more granular and precise as the early vague details transform into clear relationships and mappings. In effect, gradient loss provides a just-in-time map of where the training should go next, and how it should proceed.

The innovation of Stochastic Gradient Descent is that it updates the model's parameters on each training example per iteration, which generally speeds up the journey to convergence. Due to the advent of hyperscale datasets in recent years, SGD has grown in popularity lately as one possible method to address the ensuing logistic issues.

On the other hand, SGD has negative implications for feature scaling, and may require more iterations to achieve the same result, requiring additional planning and additional parameters, compared to regular Gradient Descent.

Popular Usage

Due to its configurability, and in spite of its shortcomings, SGD has become the most popular optimization algorithm for fitting neural networks. One configuration of SGD that is becoming dominant in new AI/ML research papers is the choice of the Adaptive Moment Estimation (ADAM, introduced in 2015) optimizer.

ADAM adapts the learning rate for each parameter dynamically ('adaptive learning rate'), as well as incorporating results from previous updates into the subsequent configuration ('momentum'). Additionally, it can be configured to use later innovations, such as Nesterov Momentum.

However, some maintain that the use of momentum can also speed ADAM (and similar algorithms) to a sub-optimal conclusion. As with most of the bleeding edge of the machine learning research sector, SGD is a work in progress.

First published 10th February 2022. Amended 10th February 20.05 EET – formatting.

Writer on machine learning, domain specialist in human image synthesis. Former head of research content at Metaphysic.ai, until its dissolution into DNEG's Brahma.ai.
Portfolio site: martinanderson.ai
Contact: martin@martinanderson.ai

Discover More