Glossary of Machine Learning Terms

A constantly updated machine learning glossary

Posted by Josh (https://twitter.com/semanti_ca) on 08-10-2018

A B C D E F G H I J K L M N O P Q R S T V X Y Z

A

Accuracy

Accuracy is the measure of how good a classification model is. It is given by the number of correctly classified examples divided by the number of classified examples.

Activation Function

Activation function is any non-linear function applied to the weighted sum of the inputs of a *neuron* in a *neural network*. The presence of activation functions makes neural networks capable of approximating virtually any function.

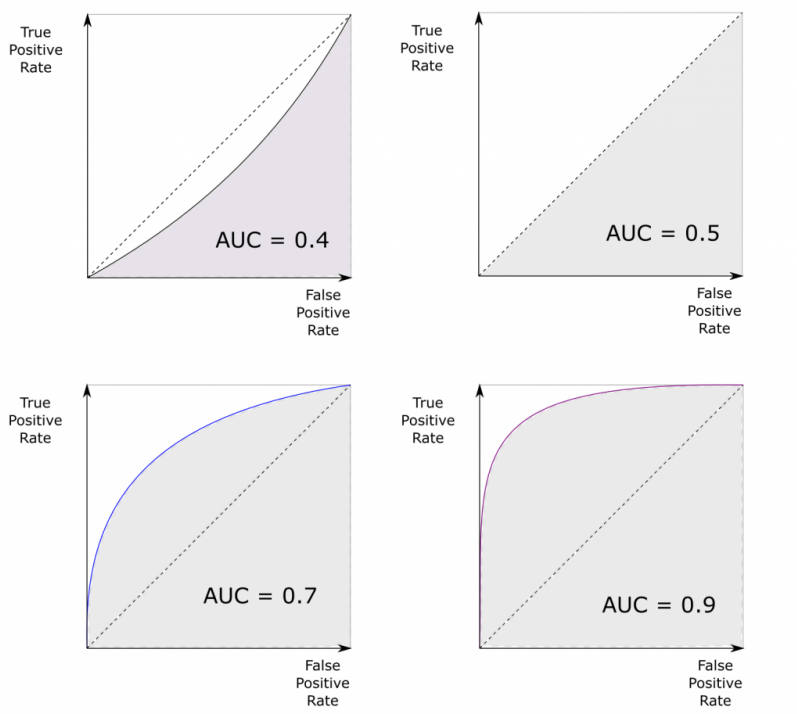
Active Learning

Active learning is a kind of *machine learning* in which the algorithm decides on the data it learns from. Active learning is valuable when *labeled examples* are expensive to obtain. Instead of blindly seeking a diverse range of labeled examples, an active learning algorithm selectively seeks the particular range of examples it needs for learning.

Area under the ROC Curve (AUC)

The ROC curve (ROC stands for "Receiver Operating Characteristic" and comes from radar engineering) is a commonly used method for and evaluating the performance of *classification models*. ROC curves use a combination the false positive rate (i.e. examples that were predicted positive, but actually negative) and true positive rate (i.e. examples that were

correctly predicted positive) to build up a summary picture of the classification performance. ROC curves are widely used because they are relatively simple to understand and capture more than one aspect of the classification.



Area under the ROC Curve. Source: deparkes.co.uk (https://deparkes.co.uk/2018/02/16/the-roc-curve/).

The higher the *area under the ROC curve* (AUC) is, the better is the classifier. An AUC greater than 0.5 is better than a random classifier. A perfect classifier would have an AUC of 1.

Autoencoder

An *autoencoder* is a *neural network* whose goal is to predict the input itself, typically through a *bottleneck* layer somewhere in the network. By introducing a bottleneck, the network is forced to learn a lower-dimensional representation of the input, effectively compressing the input into a good representation. Autoencoders are related to *principal component analysis* and other *dimensionality reduction* and *representation learning* techniques, but can learn more complex mappings due to their nonlinear nature.

B

Backpropagation

Backpropagation is the primary algorithm for performing gradient descent on *neural networks*. First, the output values of each node are calculated in a forward pass. Then, the partial derivative of the error with respect to each parameter is calculated in a backward pass

through the computation graph.

Bag of Words

Bag of words is a method of feature engineering for text documents. According to the bag of word approach, in the feature vector, each dimension represents the presence or absence of a specific token in the text document. Therefore, such an approach to representing a document as a feature vector ignores the order of words in the document. However, in practice, bag of words often works well in document classification.

Bagging

Bootstrap aggregating, also called *bagging*, is a machine learning ensemble meta-algorithm designed to improve the stability and accuracy of machine learning algorithms used in statistical classification and regression. It also reduces variance and helps to avoid overfitting.

Bagging consists of training different *base learners* on different subsets of the *training set* randomly, by drawing (with replacement) random training sets from the given sample. To obtain the final prediction, the predictions of the base learners are averaged.

Baseline

A *baseline* is an algorithm or a heuristic that can be used as a *model*, often obtained without using machine learning or by using the most simplistic feature engineering method. A baseline helps analysts quantify the minimal expected performance on a particular problem.

Base Learner

Base learner is a learning algorithm used to build *models* that are then combined by an *ensemble learning algorithm*. A popular example of a base learner is *decision tree* which is used as a base learner in the **random forest** and **gradient boosting** ensemble learning algorithms.

Batch

Batch, or minibatch, is the set of examples used in one iteration of model training using *gradient descent*.

Batch Normalization

Batch normalization is a technique that normalizes layer inputs per mini-batch. The technique consists in providing any layer in a *neural network* with inputs that have a zero mean and a unit variance (also see *standardization*). In practice, it results in speeding up training, allows for the usage of higher *learner rates*, and often has a *regularization* effect. *Batch normalization* has been found to be very effective for *convolutional* and general *feedforward neural networks*.

Bayes' Theorem

Bayes' theorem, named after 18th-century British mathematician Thomas Bayes, is a mathematical formula for determining conditional probability. The theorem provides a way to revise existing predictions or theories given new or additional evidence:

$$P(A\mid B)={\frac {P(B\mid A)\,P(A)}{P(B)}},$$

where \(A\) and \(B\) are events and \(P(B)\neq 0\).

- \(P(A\mid B)\) is a conditional probability: the likelihood of event \(A\) occurring given that \(B\) is true.
- \(P(B\mid A)\) is also a conditional probability: the likelihood of event \(B\) occurring given that \(A\) is true.
- \(P(A)\) and \(P(B)\) are the probabilities of observing \(A\) and \(B\) independently of each other.

Bias

The *bias* is an error from erroneous assumptions in the learning algorithm. High bias can cause an algorithm to miss the relevant relations between *features* and *target* outputs (underfitting).

Bias-Variance Tradeoff

The *bias–variance tradeoff* is the property of a set of *predictive models* whereby models with a lower *bias* in parameter estimation have a higher *variance* of the parameter estimates across samples, and vice versa. The bias-variance problem is the conflict in trying to simultaneously minimize these two sources of error that prevent *supervised learning* algorithms from generalizing beyond their *training set*.

Bigram

A *bigram* is a sequence of two tokens or two characters extracted from a string. For example, in the word "cat" there are two character bigrams: "ca" and "at". In the sentence "I like to swim" there are three-word bigrams: "I like", "like to", "to swim". Bigrams are often used as

features together with (or in lieu of) words in *bag of words*.

Binary Variable

A *binary variable* is a variable (a *feature* or the *target*) that can take values either "Yes" or "No" (True or False, one or zero, etc).

Binary Classification

A *binary classification* problem is a *classification* problem where the label is only one out of two classes. For example, spam detection or determining if a patient has a certain disease or not are examples of a binary classification problem.

Binning

Binning (also called **bucketing**) is the process of converting a continuous feature into multiple binary features called **bins** or **buckets**, typically based on value range. For example, instead of representing age as a single integer-valued feature, the analyst could chop ranges of gates into discrete bins: all ages between 0 and 5 years-old could be put into one bin, 6 to 10 years-old could be the second bin, 11 to 15 years-old could be the third bin, 16 to 25 would be the fourth been and so on.

Boosting

Boosting is an *ensemble learning* technique that iteratively combines a set of simple and not very accurate classifiers (referred to as *weak classifiers*) into a classifier with high accuracy (called a *strong classifier*) by giving a higher weight to the examples that the model is currently classifying incorrectly.

Multiple boosting algorithms exist. The most widely used ones are **AdaBoost** and **gradient boosting**.

Bootstrapping

Bootstrapping can have one of the following meanings:

1. *Bootstrapping* is the process of dividing the dataset into multiple subsets (with replacement). Each subset is of the same size as that of the dataset. The subsets are called bootstrap samples. Bootstrapping can be used to estimate a quantity of a *population*. This is done by repeatedly taking small samples, calculating the statistic, and taking the average of the calculated statistics.

2. *Bootstrapping* a model means training a model on a small set of labeled data, and then manually reviewing unlabeled examples for errors, and then adding those to the training set in an iterative process.

Bottleneck

A *bottleneck* layer is a layer that contains few nodes compared to the previous layers. It can be used to obtain a representation of the input with reduced dimensionality. An example of this is the use of autoencoders with bottleneck layers for nonlinear *dimensionality reduction* or *representation learning*.

Bucketing

See *binning*.

C

Categorical Feature

A *categorical feature* is a *feature* defined by a categorical variable.

Categorical Variable

A *categorical variable* is a variable that can take on one of a limited and usually fixed number of possible values. For example, the variable "street_light_color" can have three possible values: "red", "yellow", and "green".

Centroid

A *centroid* is the center of a cluster as determined by a *k-means* or **k-median** algorithm. For instance, if k is 4, then the k-means or k-median algorithm finds 4 centroids.

Class

A *class* is a group to which an *example* can belong. A labeled example consists of a *feature vector* and a reference to the class it belongs to. A specific class attached to a feature vector is called *label*. For example, in a binary classification model that detects spam, there are two classes, "spam" and "not spam". In a multi-class classification model that identifies plant species, the classes would be "trees", "flowers", "mushrooms", and so on.

Classification

Classification is a problem of assigning a *label* to an *example*.

Classification Algorithm

A *classification algorithm* is a *machine learning algorithm* that uses *labeled examples* to create a model that can be used for *classification*.

Classification Threshold

Classification threshold is a real number that defines a criterion that is applied to a *model*'s predicted score in order to separate the positive class from the negative class. For instance, a classification threshold is used when mapping *logistic regression* results to *binary classification*: consider a logistic *regression model* that determines the probability of a given email message being spam. If the classification threshold is 0.9, then logistic regression values above 0.9 are classified as spam and those below 0.9 are classified as not spam.

Classification Model

See *Model*, *Classification*

Cluster

A *cluster* is a group of *examples* (usually, this group is smaller than the complete dataset) that have some resemblance according to a *similarity metric*.

Clustering

Clustering is a problem of assigning *examples* to one or more *clusters*.

Clustering Algorithm

A *clustering algorithm* is an unsupervised learning method used to discover the inherent groupings in the data. For example, customers can be grouped on the basis of their purchasing behavior which is further used to segment the customers.

Example of clustering algorithms: *k-Means*, *hierarchical clustering*, **DBSCAN**.

Confusion Matrix

The *confusion matrix* is a table that summarizes how successful the classification model was at predicting *examples* belonging to various *classes*. One axis of the confusion matrix is the *label* that the model predicted, and the other axis is the actual label. In a *binary classification* problem, there are two classes. Let say, the classifier has to predict labels "spam" or "not spam":

	spam (predicted)	**not spam (predicted)**
spam (actual)	23	1
not spam (actual)	12	556

The preceding confusion matrix shows that of the 24 samples that actually was spam, the model correctly classified \(23\) as spam (\(23\) *true positives*), and incorrectly classified 1 as not spam (\(1\) *false negative*). Similarly, of \(568\) samples that actually were not spam, \(556\) were correctly classified (\(556\) *true negatives*) and \(12\) were incorrectly classified (\(6\) *false positives*).

The confusion matrix for a multi-class classification problem can help you determine mistake patterns. For example, a confusion matrix could reveal that a model trained to recognize different species of animals tends to mistakenly predict "cat" instead of "panther", or "mouse" instead of "rat".

Confusion matrices can be used to calculate different performance metrics, such as *precision* and *recall*.

Continuous Variable

A *continuous variable* is a variable that has an infinite number of possible values. In other words, any value is possible for the variable. Examples of a continuous variable are time, distance, or weight.

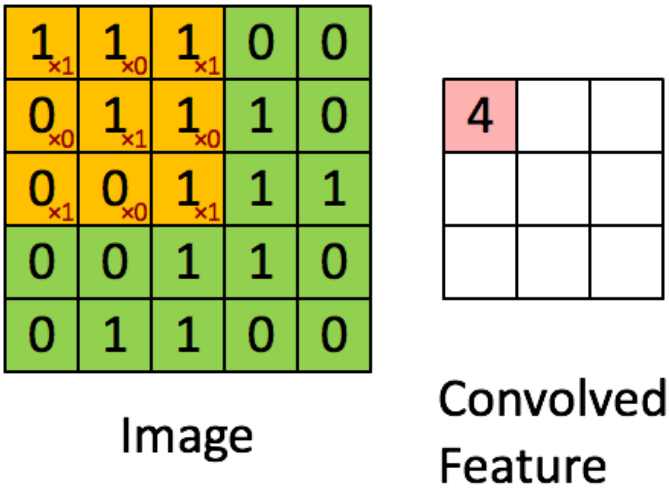
Convergence

Convergence often refers to a state reached during iterative *training* in which *training loss* and *validation loss* change very little or not at all with each *iteration* after a certain number of iterations. A model reaches convergence when additional training using the same data will not improve the model.

Convolution

A *convolution* mixes the *convolutional filter* and the input matrix in order to train weights.

Convolution is an operation of applying a *convolution filter* to the input matrix which is usually has a much higher dimensionality than that of the convolution filter. The application of the convolution filter to the input consists of multiplying the matrix of the filter by some patch of the input and then summing the values of the obtained matrix. The patches are selected by using a sliding window approach. See an illustration below:



Calculating convolution by sliding image patches over the entire input. One input patch (yellow) of the original input is multiplied by the kernel (red numbers in the yellow patch), and its sum is written to one feature map pixel (red cell in the convolved feature). Source: stanford.edu (http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution).

Convolutional Filter

A *convolutional filter* (or convolutional kernel) is usually a square matrix whose values have to be learned by a neural network training algorithm. Convolutional filters are parts of convolutional neural networks where they are used by multiplying their values by the values of the corresponding patches of the input (or the previous layer) and then summing the obtained values. See *convolution* for an illustration.

Convolutional Neural Network (CNN)

A *convolutional neural network* is a class of *feedforward neural networks*, most commonly applied to analyze images. It uses *convolutional* layers that filter inputs for useful information. These convolutional layers have parameters that are learned so that these filters are adjusted automatically to extract the most useful information for the task at hand.

Usually, multiple convolutions are used as well as multiple convolutional layers. This enables the neural network to filter images for different types of patterns and for more and more abstract information after each layer.

Convolutional networks usually also use *pooling* to detect the pattern even if it appears at some unusual place. Pooling also reduces the memory consumption and thus allows for the usage of more convolutional layers.

Cosine Similarity

Cosine similarity is a *similarity metric* that measures the similarity between two *examples*. It is given by the cosine of the angle between two feature vectors.

Cost Function

See *loss function*.

Cross-Validation

Cross-validation is a method for evaluating a statistical *model* computed by a learning algorithm that has *hyperparameters*. Divide the training data into several parts, and in turn, use one part to test the model fitted to the remaining parts. *Cross-validation* can be used for model selection or *hyperparameter tuning*.

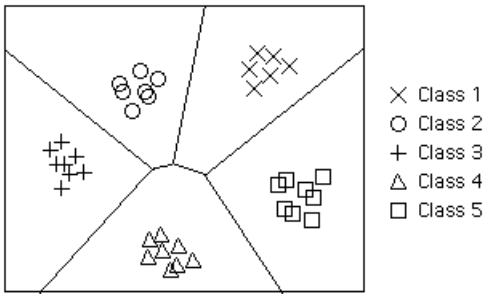
D

Dataset

A *dataset* is a collection of *examples*.

Decision Boundary

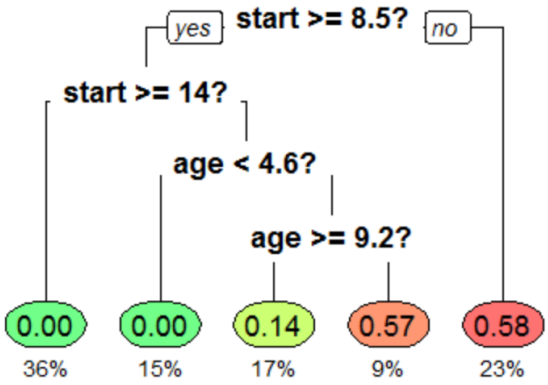
In a *classification* problem with two or more classes, a *decision boundary* is a hypersurface that partitions the underlying vector space into two or more regions, one for each *class*. How well the classifier works depends upon how well the decision boundary separates examples of different classes from one another. In the figure below, the lines separating examples of each class from one another are decision boundaries.



A decision boundary. Source: princeton.edu (https://www.cs.princeton.edu/courses/archive/fall08/cos436/Duda/PR_simp/bndrys.htm).

Decision Tree

A *decision tree* is a decision support tool that uses a tree-like graph or model of decisions and their possible consequences. A decision tree is also a way of visually representing an algorithm. The example below is a decision tree that estimates the probability of kyphosis after surgery, given the age of the patient and the vertebra at which surgery was started:



A decision tree. Source: Wikipedia (https://en.wikipedia.org/wiki/Decision_tree_learning).

A decision tree can be learned based on a *dataset* using a decision tree learning algorithm. Examples of such algorithms are ID3, C4.5, CART.

Dimensionality Reduction

Dimensionality reduction (also known as dimension reduction) is a process of converting a *dataset* having vast dimensionality (that is, each example in this dataset is a vector of very high dimensionality) into a dataset with lesser dimensionality ensuring that it conveys similar information concisely.

Typical algorithms used for dimensionality reduction are **Principal Component Analysis** (PCA), **UMAP**, and various *embedding* techniques such as **word2vec**.

Dimensionality reduction is helpful in training a *model* using a bigger dataset. Also, in many cases, the *accuracy* of the model increases after the original dataset is transformed into a dimensionality-reduced dataset.

Dropout

Dropout is a regularization technique for *neural networks*. It prevents *neurons* from co-adapting by randomly setting a fraction of them to \(0\) at each training iteration.

Dummy Variable

A *dummy variable* is usually a binary variable derived from some other variable or a combination of variables. An example of dummy variable one that takes value 0 or 1. 0 means value is true (i.e. $age < 25$) and 1 means value is false (i.e. $age >= 25$)

E

Early Stopping

Early stopping is a *regularization* method that involves ending model training before *training loss* finishes decreasing. In early stopping, the engineer ends model training when the *validation loss* starts to increase, that is when generalization performance worsens.

Embedding

An *embedding* maps an input representation, such as a word or sentence, into a vector. Most frequently embeddings refer to word embeddings such as **word2vec** or **GloVe**. However, sentences, paragraphs or images can be embedded too. For example, by mapping images and their textual descriptions into a common embedding space and minimizing the distance between them, one can match labels with images. Embeddings may be learned using *neural networks*.

Ensemble Learning

Ensemble learning is a problem of learning a *strong classifier* by combining multiple *weak classifiers*.

Ensemble Learning Algorithm

An *ensemble learning algorithm* combines multiple *weak classifiers* to build a *strong classifier* (the one with a higher accuracy than that of each individual weak classifier).

Epoch

An *epoch* is one pass through the *training set* by a machine learning algorithm.

Evaluation Metric

An *evaluation metric* is a formula or a technique used to measure the quality of the *model*. Examples include *area under the ROC curve* (AUC), *F-score*, *log loss*.

Example

An *example* (also called *instance*) is a member of a *dataset*. Typically, an example is a vector of *features*. Each feature represents some specific property of the example. For instance, if the dataset contains examples of stars, then one feature can represent the percentage of hydrogen in the star, another feature could represent star's diameter, a third feature could represent some property of star's magnetic field, and so on. All examples represented as vectors have vectors of the same dimensionality and every dimension represents the same feature.

Example, Labeled

A *labeled example* is a pair that contains the vector of features and a label. A label is typically the quantity the *model* tries to predict. If our vector of features represents the parameters of a star, then the label could be its age. The machine learning problem would be to predict the age of a star given the vector of its features.

Also, see *unlabeled example*.

Example, Unlabeled

An *unlabeled example* is an *example* that only contains features.

Also, see *labeled example*.

Exploding Gradient Problem

The *exploding gradient problem* is the opposite of the *vanishing gradient problem*. In deep neural networks (neural networks with many layers) gradients may explode during backpropagation, resulting in number overflows. A common technique to deal with exploding gradients is to perform *gradient clipping*.

F

F1 Score

F1 score (also **F-score** or **F-measure**) *evaluation metric* combines both *precision* and *recall* as a measure of the effectiveness of classification:

$$F_{1} = 2 \cdot \frac {\mathrm {precision} \cdot \mathrm {recall} }{\mathrm {precision} +\mathrm {recall}}$$

False Negative (FN)

A *false negative* is an *example* in which the model mistakenly predicted the negative class. For example, the model inferred that a particular email message was not spam (the negative *class*), but that email message actually was spam.

False Positive (FP)

A *false positive* is an example in which the model mistakenly predicted the positive class. For example, the model inferred that a particular email message was spam (the positive *class*), but that email message was actually not spam.

Feature

A *feature* is an attribute of an example, usually a part of a *feature vector*. It can be numerical or categorical. If an example is a person, it can have the following features: height (numerical), weight (numerical), race (categorical), etc.

Feature Selection

Feature selection is a process of removing from the dataset *features* that seem irrelevant for modeling. This is a combinatorial optimization problem. The direct approach (the "wrapper" method) retrains and re-evaluates a given model for many different feature sets. An approximation (the "filter" method) instead optimizes simple criteria which tend to improve performance. The two simplest optimization methods are **forward selection** (keep adding the best feature) and **backward elimination** (keep removing the worst feature).

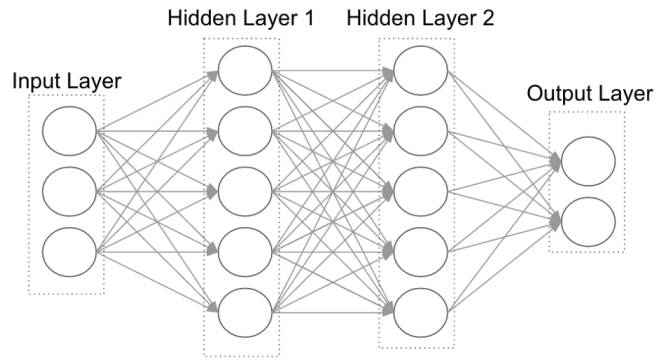
Feature Vector

A *feature vector* is a vector in which each dimension represent a certain *feature* of an *example*.

Feedforward Neural Network

A *feedforward neural network* is a *neural network* wherein connections between the *units* do not form a cycle. As such, it is different from *recurrent neural networks*.

In such a network, the information moves in only one direction, forward, from the input nodes, through the hidden nodes (if any) and to the output nodes. There are no cycles or loops in the network. See example below:



Feedforward neural network with 2 hidden layers

An example of a feedforward neural network with two hidden layers. Source: Medium (https://medium.com/notes-on-learning/crash-course-in-deeplearning-cdfed577468e).

Few-Shot Learning

Few-shot learning is a machine learning approach, usually employed in *classification*, designed to learn effective classifiers from only a small number of *training examples*.

One-shot learning tries to learn from one example only.

G

Gated Recurrent Unit (GRU)

The *Gated Recurrent Unit* (or, simply, GRU) is a simplified version of an *LSTM* unit with fewer parameters. Just like an LSTM cell, it uses a gating mechanism to allow RNNs to efficiently learn long-range dependencies by reducing the *vanishing gradient problem*. The GRU consists of a reset and update gate that determine which the extent to which the unit should keep the old value and to which the new value has to replace it at the current time step.

Gradient Clipping

Gradient clipping means capping gradient values before applying them in gradient descent algorithm during *backpropagation*. Gradient clipping helps ensure numerical stability and prevents *exploding gradient problem*.

Gradient Descent

Gradient descent is an iterative optimization algorithm for differentiable functions. It is designed to find the minimum of a function. In many *machine learning algorithms*, gradient descent, or its variant is used to find the minimum of the *loss function* given the training dataset.

Also see: *stochastic gradient descent*.

Grid Search

Grid search is a way of *hyperparameter tuning*. The process consists of training the *model* on all possible combinations of hyperparameter values and then selecting the best combination. The best combination of hyperparameters is the one that performs the best on the *validation set*.

H

Hidden Layer

A *hidden layer* in *neural network* is a layer in between input layers and output layers, where *neurons* take in a set of weighted inputs and produce an output through an *activation function*.

Hierarchical Clustering Algorithm

Hierarchical clustering algorithms is a category of *clustering algorithms* that create a tree of clusters. Hierarchical clustering algorithms are well-suited to hierarchical data, such as botanical taxonomies. There are two types of hierarchical clustering algorithms:

- An *agglomerative clustering algorithm* first assigns every example to its own cluster and iteratively merges the closest clusters to create a hierarchical tree;
- A *divisive clustering algorithm* first groups all examples into one cluster and then iteratively divides the cluster into a hierarchical tree.

See also: *partition clustering algorithm*.

Hinge Loss

The *hinge loss* is a *loss function* used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for *support vector machines* (SVMs). For an intended output $y \in \{-1,1\}$ and a classifier score $\hat{y}$, the hinge loss of the prediction $\hat{y}$ is defined as

$$loss(\hat{y})=\max(0,1-y \cdot \hat{y})$$

Note that y should be the "raw" output of the classifier's decision function, not the predicted class label. For instance, in linear SVMs, $\hat{y}=\vec{w} \cdot \vec{x} + b$, where $(\vec{w}, b)$ are the parameters of the hyperplane and $\vec{x}$ is the example to

classify.

Holdout Set

Holdout set is a part of the *dataset* that contains *examples* intentionally not used ("held out") during training. The *validation set* and *test set* are examples of holdout data. Holdout data helps to evaluate *model*'s ability to generalize to examples other than the examples it was trained on. The loss on the holdout set provides a better estimate of the loss on an unseen data set than does the loss on the training set.

Hyperparameter

Hyperparameter is a parameter of a *machine learning algorithm* whose value is not optimized during training. Depending on the algorithm, a hyperparameter could be the number of training *iterations*, the size of the *minibatch*, a *regularization* parameter, the value of the *learning rate*, and many others.

Since hyperparameters are not optimized by the machine learning algorithm itself, it is often optimized using *cross-validation* and techniques like *grid search*, *random search*, *Bayesian optimization*, *evolutionary optimization*, and others.

Hyperparameter Tuning

Hyperparameter tuning is the procedure that aims at finding the best values of *hyperparameters*, usually by trying different values and checking their performance on the *validation set*.

One frequent way of hyperparameter tuning is *grid search*.

Hyperplane

A *hyperplane* is a boundary that separates a space into two subspaces. For example, a line is a hyperplane in two dimensions and a plane is a hyperplane in three dimensions. In machine learning, a hyperplane is usually a boundary separating a high-dimensional space. For instance, the **Support Vector Machine** (SVM) machine learning algorithm uses hyperplanes to separate positive classes from negative classes, often in a very high-dimensional space.

I

Input Layer

The *input layer* in *neural network* is a layer whose *neurons* take as input the *features* of the input *instance*.

Instance

See *example*.

Instance-Based Learning

Instance-based learning (sometimes called memory-based learning) is a family of *learning algorithms* that, instead of performing explicit generalization, compares new problem instances with instances seen in training, which have been stored in memory.

An example of an instance-based learning algorithm is *k Nearest Neighbours*.

Iteration

In machine learning, *iteration* refers to the number of times the *machine learning algorithm*'s parameters are updated while training a model on a dataset. For example, each iteration of training a *neural network* takes a certain number of *training examples* and updates the *parameters* by using *gradient descent* or some other weight update rule.

K

K-Means

k-means is a *partitioning clustering algorithm* for clustering data into exactly \(k\) clusters. It works as follows. First, define \(k\) initial cluster *centroids*. Second, assign each example to the closest centroid. Third, recompute the new position for each centroid as the average of the examples assigned to it. Iterate back to step two.

K Nearest Neighbors

K Nearest Neighbors (also often abbreviated as kNN) is an *instance-based learning algorithm* that can be used for both classification and regression. When used in the *classification* context, the algorithm predicts the class of an *unlabeled example* as the majority class among \(k\) its closest neighbors in the vector space. In the *regression* context, the label of an unlabeled example is calculated as an average of the labels of the \(k\) its closest neighbors. The distance from one example to another is usually given by a *similarity metric*.

Kernel

Kernel methods can be thought of as *instance-based learners*: rather than learning some fixed set of *parameters* corresponding to the *features* of their inputs, they instead "remember" the i-th training example $(\vec{x}^{i},y^{i})$ and learn for it a corresponding weight w^{i}. Prediction for unlabeled inputs, i.e., those not in the training set, is treated by the application of a similarity function k, called a *kernel*, between the unlabeled input $\vec{x'}$ and each of the training inputs $\vec{x}^{i}$. For instance, a kernelized binary classifier typically computes a weighted sum of similarities

$$\hat{y}=\operatorname{sgn} \sum_{i=1}^{n} w^{i}y^{i}k(\vec{x}^{i},\vec{x'}),$$

where

- $\hat{y} \in \{-1,+1\}$ is the kernelized binary classifier's predicted label for the unlabeled input $\vec{x'}$ whose hidden true label y is of interest;
- $k\colon \mathcal{X}\times \mathcal{X} \to \vec{R}$ is the kernel function that measures similarity between any pair of inputs $\vec{x}, \vec{x'} \in \mathcal{X}$;
- the sum ranges over the n labeled examples $\{(\vec{x}^{i},y^{i})\}$ ($i=1..n$) in the classifier's training set, with $y^{i} \in \{-1,+1\}$;
- the $w^{i} \in \mathbb{R}$ are the weights for the training examples, as determined by the learning algorithm;
- the sign function sgn determines whether the predicted classification $\hat{y}$ comes out positive or negative.

L

Label

A *label* is a *class* (in classification) or a *target* (in regression) assigned to a *labeled example*.

Labeled Example

See *Example*, *Labeled*.

Linear Regression

Linear regression is a popular *regression algorithm* that learns a model which is a linear combination of features.

Learning Algorithm

A *learning algorithm*, or a *machine learning algorithm*, is any algorithm that can produce a *model* by analyzing a *dataset*.

Learning Rate

Learning rate is a scalar used to update *model parameters* via *gradient descent*. During each iteration, the gradient descent algorithm multiplies the gradient by the learning rate. The resulting product is called the **gradient step**.

Learning rate is one of the *hyperparameters* of *neural network* learning algorithms.

Log Loss

Log loss is *loss function* used in the binary *logistic regression*:

$$logloss(y) = -{(y\log(p) + (1 - y)\log(1 - p))},$$

where \(p\) is the probability predicted by the model that the label is \(y\) and \(y\) is the true label.

Logistic Regression

Logistic regression is *regression algorithm* that produces a model that generates a probability for each possible discrete *label* value in classification problems by applying a *sigmoid function* to a linear prediction.

The term *logistic regression* is usually employed in *binary classification* problems. If the label can have more than two discrete values as in *multi-class classification* problems, then the term multi-class logistic regression or multinomial regression is used.

Long Short-Term Memory Unit (LSTM)

Long short-term memory (LSTM) *units* in *recurrent neural networks* help reducing the *vanishing gradient problem* during the *backpropagation*. LSTM unit is a *neuron* that has a memory cell and three gates: "input", "output" and "forget". The purpose of the memory cell is to retain information previously used by the RNN or forget if needed. Neural networks with LSTM units, also called LSTM networks, are explicitly designed to avoid the long-term dependency problem in RNNs and have been shown to be able to learn complex sequences better than simple RNNs.

The structure of a memory cell is: an input gate, that determines how much of information from the previous layer gets stored in the cell; the output gate, that determines how of the next layer gets to know about the state of the current cell; and the forget gate, which determines what to forget about the current state of the memory cell.

Loss Function

In the *classification* context, a function $l(\hat{y}, y) : Y \times Y \rightarrow \mathbb {R}_{\ge 0}$, describes the cost of assigning the *label* $\hat{y}$ to a sample of *class* y.

In the *regression* context, a function $l(\hat{y}, y) : \mathbb{R} \times \mathbb {R} \rightarrow \mathbb {R}_{\ge 0}$, describes the cost of assigning the value $\hat{y}$ to the regression function evaluated at $\vec{x}$, where it takes value y.

The simplest loss function for classification is the $0-1$ loss, having value 0 if and only if $y = \hat{y}$, and value 1 otherwise. The typical loss function for regression estimation is the squared error. The expected value of the loss is called *risk*. Synonym: *cost function*, term less frequently used in the classification literature.

LSTM Network

An *LSTM network* is a *recurrent neural network* that is using *LSTM units*.

M

Machine Learning

Machine learning is a subfield of computer science, mathematics, and statistics that focuses on the design of systems that can learn from and make decisions and predictions based on data. Machine learning enables computers to act and make decisions based on examples rather than being explicitly programmed to carry out a certain task. *Machine learning algorithms* are designed to learn and improve over time when exposed to new data.

Machine Learning Algorithm

See *learning algorithm*.

Metric Learning

Metric learning (or similarity metric learning) is the task of learning a similarity function over *examples*. A metric or similarity function has to obey four axioms: non-negativity, identity of indiscernibles, symmetry, and subadditivity/triangle inequality. In practice, metric learning algorithms ignore the condition of identity of indiscernibles and learn a pseudo-metric.

Minibatch

See *batch*.

Model

A *model*, also known as a *statistical model*, is the result of a *machine learning algorithm* applied to the training data. Model is often a parametrized mathematical formula, where parameters are learning by the machine learning algorithm. Given an input example, a model can produce the classification label or regression value directly, or it can produce a probability for each possible value (label).

Model, Classification

A *classification model* is the model that is used to classify *examples*, that is to produce a categorical label given a *feature vector*.

Model, Regression

A *regression model* is the model that is used to produce a real-valued label given a *feature vector*.

Multi-Class Classification

Multi-class classification is a *classification* problem that distinguishes among more than two classes. For example, the problem of assigning a label such as "history", "technology", "finance", etc to a text document would be multi-class. Conversely, the problem of dividing emails into only two categories (spam and not spam) would be a *binary classification* problem.

Multinomial Regression

Multinomial regression is a variant of the *logistic regression* algorithm used for *multi-class classification* problems.

N

Neuron

A *neuron*, also called *unit* is a node in a *neural network*, typically taking in multiple input values and generating one output value. The neuron calculates the output value by applying an *activation function* (nonlinear transformation) to a weighted sum of input values.

Neural Network

A *model* that is composed of layers consisting of simple connected *units* or *neurons*.

Normalization

Normalization is the process of converting an actual range of values into a standard range of values, typically in the interval $[-1,+1]$ or $[0,1]$.

For example, suppose the natural range of a certain *feature* is 350 to $1,450$. By subtracting 350 from every value of the feature, and dividing the result by $1,100$, one can normalize those values into the range $[0,1]$.

O

One-Hot Encoding

One-hot encoding refers to a way of transforming a *categorical feature* into a vector of several binary features where all components are 0, except for one component with a value of 1. For example, if our *example* has a categorical feature "weather" and this feature has three possible values: "sun", "rain", "clouds", then to transform this feature into something a *machine learning algorithm* that can only work with numerical values, one can transform this feature into a vector of three numerical values:

$$sun = [1,0,0]$$ $$rain = [0,1,0]$$ $$clouds = [0,0,1]$$

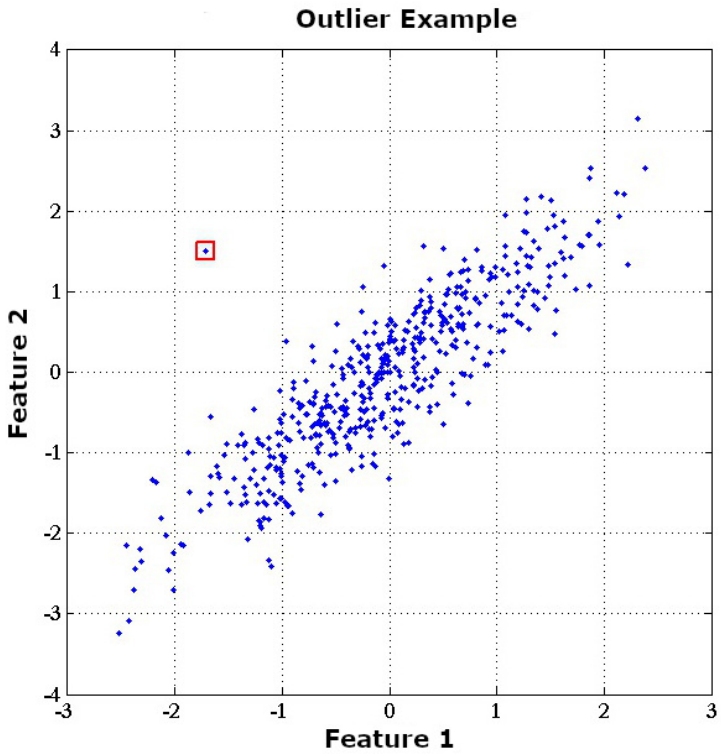
One-Shot Learning

One-shot learning is the problem of training a *model* with only a single *example* per *class*.

One way to build a system capable of one-shot learning is to use *representation learning*, to learn representations or *features* of data that can be used to accurately classify a single example.

Outlier

An *outlier* is an *example* that appears far away and diverges from an overall pattern in the *dataset*:



An outlier.

Output Layer

The *output layer* in *neural network* is a layer whose *neurons* produce the output (for the *classification* or *regression* task).

Overfitting

Overfitting occurs when the machine learning algorithm learns a model that fits the training data too well by incorporating details and noise specific to the training data. Overfitting is characterized by the nearly perfect prediction by the model of the labels of the *training examples* and poor prediction of the labels of examples from the validation set.

The problem of overfitting is usually solved by *regularization* or *early stopping*,

P

Parameter

A *parameter* of a *model* is the quantity a machine learning algorithm modifies in order to minimize the *loss function*. For example, in *neural networks*, parameters are weights applied to inputs of *neurons*.

Partitioning Clustering Algorithm

A *partitioning clustering algorithm* is used to put *unlabeled examples*, within a dataset, into multiple groups based on their similarity. The similarity is given by a *similarity metric*, chosen by the analyst. The algorithms also require the analyst to specify the number of clusters to be generated.

Examples of a partition clustering algorithm include *k-means*, **k-medoids**, **Fuzzy c-means**.

Pooling

Pooling means reducing a matrix created by an earlier convolution to a smaller matrix. Pooling usually involves taking either the maximum or average value across the pooled area.

Population

Population is the complete set of examples, from which the *dataset* was obtained. Usually, the dataset size is assumed to be much smaller than the size of the population. In many cases, population size is infinite.

Precision

Precision can be measured as of the total positive predictions made by the *model*, how many those positive predictions were correct. It can be computed as follows:

$$precision = \frac{TP}{(TP + FP)},$$ where \(TP\) is the number of *true positives*, \(FP\) is the number of *false positives*.

See also: *recall* and *confusion matrix*.

Predictive Model

See *model*.

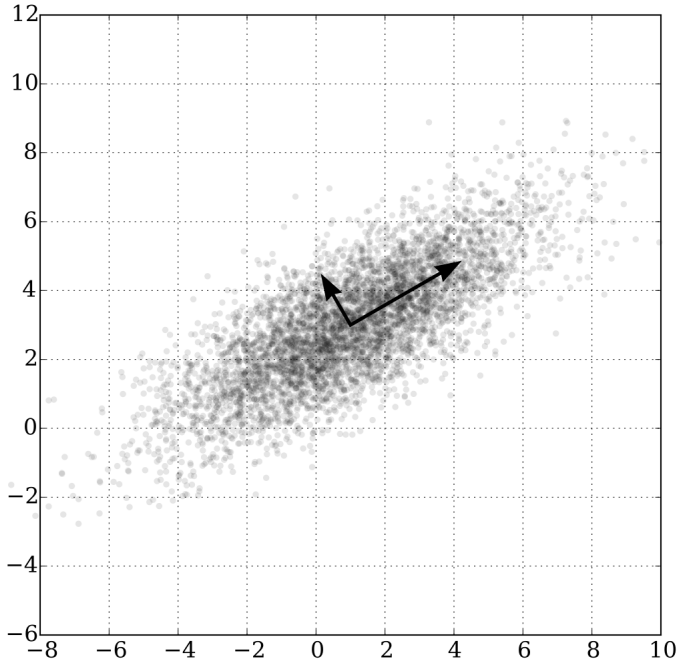
Principal Component Analysis (PCA)

Principal component analysis, or PCA, is a linear transformation which projects \(n\) examples, each consisting of \(m\) features on a hyperplane in such a way that the projection error is minimal.

The principal components are random variables of maximal variance constructed from linear combinations of the input features. Equivalently, they are the projections onto the principal component axes, which are lines that minimize the average squared distance to each example

in the data set. To ensure uniqueness, all of the principal component axes must be orthogonal.

Below, for a two-dimensional dataset, its two principal components are shown as arrows:



Principal components. Source: Wikipedia.

Basically, any dataset can be represented by its principal component vectors and the projections of examples on those vectors. By keeping only several biggest principal components and the projections of the examples on them, and by discarding the rest of information, one can reconstruct the dataset from a much smaller amount of information (with some small loss in accuracy of reconstruction because of the discarded information).

PCA is often used for not just for dimensionality reduction, but also for visualization (by reducing the dataset to two or three dimensions/principal components) and clustering (by doing it in the reduced space).

R

Random Search

Random search is a *hyperparameter tuning* technique in which the combinations of different hyperparameter values are first generated and then they are sampled randomly and used to train a model.

Recall

Recall is described as the measure of how many of the positively *labeled examples* were correctly classified by the *model*:

$$recall = \frac{TP}{(TP + FN)},$$

where \(TP\) is the number of *true positives* and \(FN\) is the number of *false negatives*.

See also: *precision* and *confusion matrix*.

Recurrent Neural Network (RNN)

A *recurrent neural network* (or, RNN) is a *neural network* that contains an internal memory. An RNN has an internal loop that allows information to persist in the network. *Neurons* receive information not just from the previous layer, but also from themselves from the previous pass. This means that the order of inputs to the RNN matter as RNNs have a state.

RNNs are particularly sensitive to the *vanishing* and *exploding gradient problems*, where depending on the *activation functions* used, the information can get lost over time. *Long short-term memory units* (LSTM) addresses this problem. RNNs are commonly used with sequential data, like in natural language processing.

Common examples of sequential data includes texts, audio and video.

Regression Algorithm

A *regression algorithm* is a machine learning algorithm that produces a *regression model*.

Regression Model

A *regression model* type of *model* that outputs continuous (typically, floating-point).

Reinforcement Learning

Reinforcement learning is a subfield of machine learning where the machine perceives its environment's **state** as a vector of features. The machine can execute **actions** in every state and different actions bring different **rewards** and also moves the machine to another state. The goal of the reinforcement learning is to learn a **policy**, that is the prescription of the optimal action to execute in each state. The action is optimal if it maximizes the average reward.

The biggest difference with *classification* is that the optimal action in each state depends on the probability of getting into other states and the optimal actions to take in those states. In classification, the optimal prediction for an *example* is independent of the future examples and predictions.

Representation Learning

Representation learning, or *feature learning*, is a set of techniques that allows a system to automatically discover the representations needed for feature detection or classification from raw data. This replaces manual feature engineering and allows a machine to both learn the features and use them to perform a specific task.

Popular representation learning techniques include *k-means* clustering (additional features are added to *feature vectors* indicating which cluster the example belongs to), *principal component analysis* and *autoencoders*.

Regression Model

See *Model*, *Regression*.

Regularization

Regularization is a technique to make the fitted function smoother. This helps to prevent overfitting. The most widely used regularization techniques are **L1**, **L2**, **dropout**, and **weight decay**.

Risk

The *risk* is the expected value of the *loss function*. Because the probability distribution of the data is usually unknown, **empirical risk** (the average value of the loss obtained by applying the loss function to all training examples) is often used instead. Many machine learning algorithms learn such values of the parameters that minimize the empirical risk.

S

Semi-Supervised Learning

Semi-supervised learning is a problem of learning a *model* by using both *labeled* and *unlabeled examples*.

Semi-supervised learning techniques take advantage of a small amount of labeled data and a large amount of unlabeled data to produce a better model than a purely supervised learning or a purely unsupervised learning technique.

Similarity Metric

A *similarity metric* is a function that takes two *feature vectors* as input and returns a real number that indicates how these two feature vectors are "similar". Usually, the more two vectors are similar the higher is the real number. Most often similarity metrics are used in clustering. The most widely used similarity metric is *cosine similarity* which returns the cosine of the angle between two feature vectors. The similarity metric can be handcrafted for a problem in hand or can be learned (see *metric learning*).

Standardization

Standardization (or **z-score normalization**) is the process where the *features* are rescaled so that they'll have the properties of a standard normal distribution with $\mu=0$ and $\sigma=1$, where μ is the mean (the average value of the feature, averaged over all *examples* in the dataset) and σ is the standard deviation from the mean.

Standard scores (or z-scores) of features are calculated as follows:

$$f' = \frac{f - \mu}{\sigma}$$

Stochastic Gradient Descent (SGD)

Stochastic Gradient Descent, or SGD, is a type of *gradient descent* algorithm where the gradient of the function to be optimized is computed by taking a sample of data. The update to the coefficients is performed for each training instance, rather than at the end of the batch of instances.

The learning can be much faster with stochastic gradient descent for very large training datasets and often one only need a small number of passes through the dataset (one pass through the dataset is called *epoch*) to reach a good or good enough set of coefficients.

Strong Classifier

A *strong classifier* is a *classifier* produced by an *ensemble learning algorithm* by combining multiple *weak classifiers* to reach a much higher classification *accuracy* than that of each individual weak classifier.

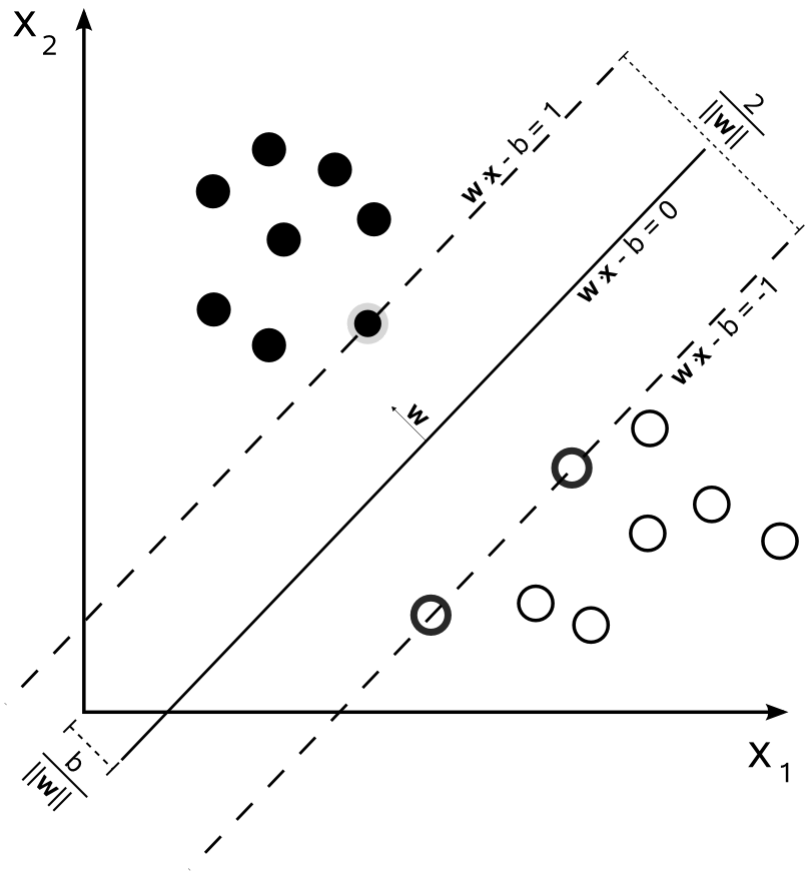
Supervised Learning

Supervised learning is a problem of learning a *model* (either regression or classification) by using *labeled examples*.

Support Vector Machine (SVM)

Support vector machine, or SVM, *is a* classification algorithm *that seeks to maximize the margin between positive and negative classes. SVM is often used together with* kernels, *functions that map input examples (given as multidimensional vectors) to a higher dimensional space. For example, consider a classification problem in which example consists of a hundred features. In order to maximize the margin between positive and negative classes, SVM, using a kernel function, could internally map those features into a million-dimension space. SVM uses a loss function called* hinge loss*.

For two-dimensional *feature vectors*, the problem and the solution can be visualized as a plot below (taken from Wikipedia (https://en.wikipedia.org/wiki/Support_vector_machine)):



On the above illustration, the dark circles are positive examples, the white circles are negative examples, and the line given by \(wx - b = 0\) is the *decision boundary*.

T

Target

See *label*.

Tokenization

In Natural Language Processing, *tokenization* is the process of splitting a text string into units called tokens. A token may be a word or a group of words.

Topic Modeling

Topic modeling is the problem of finding topics in a collection of documents. Usually, the analyst decides how many topics a collection of documents is likely to contain and the algorithm groups together words characterizing each topic. The algorithms frequently used for topic modeling are **Latent Dirichlet Allocation** (LDA) and **Latent Semantic Indexing** (LSI).

Training

Training is the process of building a *model* by applying a *machine learning algorithm* to the *training data*.

Training Example

A *training example* is a member of the *training set*.

Training Loss

See *risk*.

Training Set

The *training set* is a holdout set used for final *model* assessment (following the *hyperparameter tuning* procedure).

Transfer Learning

Transfer learning is using a model trained to solve one problem to help to solve another problem. For example, a *neural network* trained to distinguish between different kinds of jungle animals can be reused to train another neural network that would distinguish between different kinds of domestic animals.

Transfer learning might involve transferring knowledge from the solution of a simpler task to a more complex one or involve transferring knowledge from a task where there is more data to one where there is fewer data.

True Positive (TP)

In *binary classification*, a *true positive* is a positive *labeled example* whose label was predicted by the model correctly.

True Negative (TN)

In *binary classification*, a *true negative* is a negative *labeled example* whose label was predicted by the model correctly.

U

Underfitting

Underfitting is a situation in which the *model* trained on the *training data* doesn't predict well *training examples*.

Unlabeled Example

See *Example, Unlabeled*.

Unsupervised Learning

Unsupervised learning is a problem, given an *unlabeled dataset*, automatically find hidden (or latent) structure in this dataset.

Examples of an unsupervised learning problem include *clustering*, *topic modeling*, and *dimensionality reduction*.

Unsupervised Learning Algorithm

An *unsupervised learning algorithm* is an algorithm that solves an *unsupervised learning* problem.

V

Validation Loss

Validation loss is the average loss computed by applying the *loss function* the outputs of the model applied to the examples from the validation set.

Validation Example

A *validation example* is a member of the *validation set*.

Validation Set

The *validation set* is a holdout set used for *hyperparameter tuning*.

Vanishing Gradient Problem

The *vanishing gradient problem* happens in very deep neural networks, typically recurrent neural networks, that use activation functions whose gradients tend to be small. Because these small gradients are multiplied during backpropagation, they tend to "vanish" throughout the layers, preventing the network from learning long-term dependencies. Common ways to counter this problem is to use activation functions like *ReLU* or *LSTM* that do not suffer from small gradients. The opposite of this problem is called the *exploding gradient problem*.

Variance

The *variance* is an error from sensitivity to small fluctuations in the *training set*. High variance can cause an algorithm to model the random noise in the training data, rather than the intended outputs (overfitting).

W

Weak Classifier

In *ensemble learning*, a *weak classifier* is usually one of a collection of low-accuracy classifiers, which, when combined by an *ensemble learning algorithm*, can produce a *strong classifier*.

This glossary is constantly updated. Didn't find the term you looked for? Let us know (https://www.semanti.ca/contacts/) and we will add this term to the list.

Read our previous post "Recommended IDE for Data Scientists and Machine Learning Engineers (https://semanti.ca/blog/?recommended-ide-for-data-scientists-and-machine-learning-engineers)" or subscribe (https://semanti.ca/blog/?feed) to our RSS feed.

Found a mistyping or an inconsistency in the text? Let us know (https://www.semanti.ca/contacts/) and we will improve it.

Like it? Share it!

(https://twitter.com/home?status=https://www.semanti.ca/blog/?glossary-of-machine-learning-terms)
(https://www.facebook.com/sharer/sharer.php?u=https://www.semanti.ca/blog/?glossary-of-machine-learning-terms)
(https://plus.google.com/share?url=https://www.semanti.ca/blog/?glossary-of-machine-learning-terms)
(https://www.linkedin.com/shareArticle?mini=true&url=https://www.semanti.ca/blog/?glossary-of-machine-learning-terms)
(http://www.reddit.com/submit?url=https://www.semanti.ca/blog/?glossary-of-machine-learning-terms)


(/)

Home (/) · Blog (/blog/) · Pricing (/pricing/) · About (/about/) · Contact (/contacts/)

+1 646 9050250

human@semanti.ca (mailto:human@semanti.ca)

semanti.ca

We build AI-powered APIs for automated extraction of data from web pages. No programming required.

(https://www.linkedin.com/company/semanti-

(https://plus.google.com/u/0/b/semantica_plus)