Machine Learning Glossary

This glossary defines general machine learning terms, plus terms specific to TensorFlow.

Jnfortunately, as of April 2019 we no longer update non-English versions of Machine Learning Crash Cour
: see the English version (the version you are currently reading) for the most up-to-date content.

ɔu Know?

ın **filter the glossary** by choosing a topic from the Glossary dropdown in the top navigation bar.

A

A/B testing

A statistical way of comparing two (or more) techniques, typically an incumbent against a new rival. A/B testing aims to determine not only which technique performs better but also to understand whether the difference is statistically significant. A/B testing usually considers only two techniques using one measurement, but it can be applied to any finite number of techniques and measures.

accuracy

The fraction of **predictions** (#prediction) that a **classification model** (#classification_model) got right. In **multi-class classification** (#multi-class), accuracy is defined as follows:

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Number Of Examples}}$$

In **binary classification** (#binary_classification), accuracy has the following definition:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Number Of Examples}}$$

See **true positive** (#TP) and **true negative** (#TN).

action

RL

In reinforcement learning, the mechanism by which the **agent** (#agent) transitions between **states** (#state) of the **environment** (#environment). The agent chooses the action by using a **policy** (#policy).

activation function

A function (for example, **ReLU** (#ReLU) or **sigmoid** (#sigmoid_function)) that takes in the weighted sum of all of the inputs from the previous layer and then generates and passes an output value (typically nonlinear) to the next layer.

active learning

A **training** (#training) approach in which the algorithm *chooses* some of the data it learns from. Active learning is particularly valuable when **labeled examples** (#labeled_example) are scarce or 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.

AdaGrad

A sophisticated gradient descent algorithm that rescales the gradients of each parameter, effectively giving each parameter an independent **learning rate** (#learning_rate). For a full explanation, see this paper (http://www.jmlr.org/papers/volume12/duchi11a/duchi11a.pdf).

agent RL

In reinforcement learning, the entity that uses a **policy** (#policy) to maximize expected **return** (#return) gained from transitioning between **states** (#state) of the **environment** (#environment).

agglomerative clustering

See **hierarchical clustering** (#hierarchical_clustering).

AR

Abbreviation for **augmented reality** (#augmented_reality).

area under the PR curve

See **PR AUC (Area under the PR Curve)** (#PR_AUC).

area under the ROC curve

See **AUC (Area under the ROC curve)** (#AUC).

artificial general intelligence

A non-human mechanism that demonstrates a *broad range* of problem solving, creativity, and adaptability. For example, a program demonstrating artificial general intelligence could translate text, compose symphonies, *and* excel at games that have not yet been invented.

artificial intelligence

A non-human program or model that can solve sophisticated tasks. For example, a program or model that translates text or a program or model that identifies diseases from radiologic images both exhibit artificial intelligence.

Formally, **machine learning** (#machine_learning) is a sub-field of artificial intelligence. However, in recent years, some organizations have begun using the terms *artificial intelligence* and *machine learning* interchangeably.

attribute

Synonym for **feature** (#feature). In fairness, attributes often refer to characteristics pertaining to individuals.

AUC (Area under the ROC Curve)

An evaluation metric that considers all possible **classification thresholds** (#classification_threshold).

The Area Under the **ROC curve** (#ROC) is the probability that a classifier will be more confident that a randomly chosen positive example is actually positive than that a randomly chosen negative example is positive.

augmented reality

A technology that superimposes a computer-generated image on a user's view of the real world, thus providing a composite view.

automation bias

When a human decision maker favors recommendations made by an automated decision-making system over information made without automation, even when the automated decision-making system makes errors.

average precision

A metric for summarizing the performance of a ranked sequence of results. Average precision is calculated by taking the average of the **precision** (#precision) values for each relevant result (each result in the ranked list where the recall increases relative to the previous result).

See also **Area under the PR Curve** (#area_under_the_pr_curve).

B

backpropagation

The primary algorithm for performing **gradient descent** (#gradient_descent) on **neural networks** (#neural_network). First, the output values of each node are calculated (and cached) in a forward pass. Then, the **partial derivative** (#partial_derivative) of the error with respect to each parameter is calculated in a backward pass through the graph.

bag of words

A representation of the words in a phrase or passage, irrespective of order. For example, bag of words represents the following three phrases identically:

- the dog jumps
- jumps the dog
- dog jumps the

Each word is mapped to an index in a **sparse vector** (#sparse_vector), where the vector has an index for every word in the vocabulary. For example, the phrase *the dog jumps* is mapped into a feature vector with non-zero values at the three indices corresponding to the words *the*, *dog*, and *jumps*. The non-zero value can be any of the following:

- A 1 to indicate the presence of a word.
- A count of the number of times a word appears in the bag. For example, if the phrase were *the maroon dog is a dog with maroon fur*, then both *maroon* and *dog* would be represented as 2, while the other words would be represented as 1.

- Some other value, such as the logarithm of the count of the number of times a word appears in the bag.

baseline

A **model** (#model) used as a reference point for comparing how well another model (typically, a more complex one) is performing. For example, a **logistic regression model** (#logistic_regression) might serve as a good baseline for a **deep model** (#deep_model).

For a particular problem, the baseline helps model developers quantify the minimal expected performance that a new model must achieve for the new model to be useful.

batch

The set of examples used in one **iteration** (#iteration) (that is, one **gradient** (#gradient) update) of **model training** (#model_training).

See also **batch size** (#batch_size).

batch normalization

Normalizing (#normalization) the input or output of the **activation functions** (#activation_function) in a **hidden layer** (#hidden_layer). Batch normalization can provide the following benefits:

- Make **neural networks** (#neural_network) more stable by protecting against **outlier** (#outliers) weights.
- Enable higher **learning rates** (#learning_rate).
- Reduce **overfitting** (#overfitting).

batch size

The number of examples in a **batch** (#batch). For example, the batch size of **SGD** (#SGD) is 1, while the batch size of a **mini-batch** (#mini-batch) is usually between 10 and 1000. Batch size is usually fixed during **training** (#training) and **inference** (#inference); however, **TensorFlow** (#TensorFlow) does permit dynamic batch sizes.

Bayesian neural network

A probabilistic **neural network** (#neural_network) that accounts for uncertainty in **weights** (#weight) and outputs. A standard neural network regression model typically **predicts** (#prediction) a scalar value; for example, a model predicts a house price of 853,000. By contrast, a Bayesian neural network predicts a distribution of values; for example, a model predicts a house price of 853,000 with a standard deviation of 67,200. A Bayesian neural network relies on Bayes' Theorem (https://betterexplained.com/articles/an-intuitive-and-short-explanation-of-bayes-theorem/) to calculate uncertainties in weights and predictions. A Bayesian neural network can be useful when it is important to quantify uncertainty, such as in models related to pharmaceuticals. Bayesian neural networks can also help prevent **overfitting** (#overfitting).

Bellman equation RL

In reinforcement learning, the following identity satisfied by the optimal **Q-function** (#q-function):

$$Q(s,a) = r(s,a) + \gamma \mathbb{E}_{s'|s,a} \max_{a'} Q(s',a'))$$

Reinforcement learning (#reinforcement_learning) algorithms apply this identity to create **Q-learning** (#q-learning) via the following update rule:

$$Q(s,a) \leftarrow Q(s,a) + \alpha \left[r(s,a) + \gamma \max_{a_1} Q(s',a') - Q(s,a) \right]$$

Beyond reinforcement learning, the Bellman equation has applications to dynamic programming. See the Wikipedia entry for Bellman Equation (https://wikipedia.org/wiki/Bellman_equation).

bias (ethics/fairness)

1. Stereotyping, prejudice or favoritism towards some things, people, or groups over others. These biases can affect collection and interpretation of data, the design of a system, and how users interact with a system. Forms of this type of bias include:

- **automation bias** (#automation_bias)
- **confirmation bias** (#confirmation_bias)
- **experimenter's bias** (#confirmation_bias)
- **group attribution bias** (#group_attribution_bias)
- **implicit bias** (#implicit_bias)
- **in-group bias** (#in-group_bias)
- **out-group homogeneity bias** (#out-group_homogeneity_bias)

2. Systematic error introduced by a sampling or reporting procedure. Forms of this type of bias include:

- **coverage bias** (#selection_bias)
- **non-response bias** (#selection_bias)
- **participation bias** (#participation_bias)
- **reporting bias** (#reporting_bias)
- **sampling bias** (#selection_bias)
- **selection bias** (#selection_bias)

Not to be confused with the bias term (#bias) in machine learning models or **prediction bias** (#prediction_bias).

bias (math)

An intercept or offset from an origin. Bias (also known as the **bias term**) is referred to as *b* or w_0 in machine learning models. For example, bias is the *b* in the following formula:

$$y' = b + w_1x_1 + w_2x_2 + \ldots w_nx_n$$

Not to be confused with **bias in ethics and fairness** (#bias_ethics) or **prediction bias** (#prediction_bias).

bigram

An **N-gram** (#N-gram) in which N=2.

binary classification

A type of **classification** (#classification_model) task that outputs one of two mutually exclusive **classes** (#class). For example, a machine learning model that evaluates email messages and outputs either "spam" or "not spam" is a **binary classifier** (#binary_classification).

binning

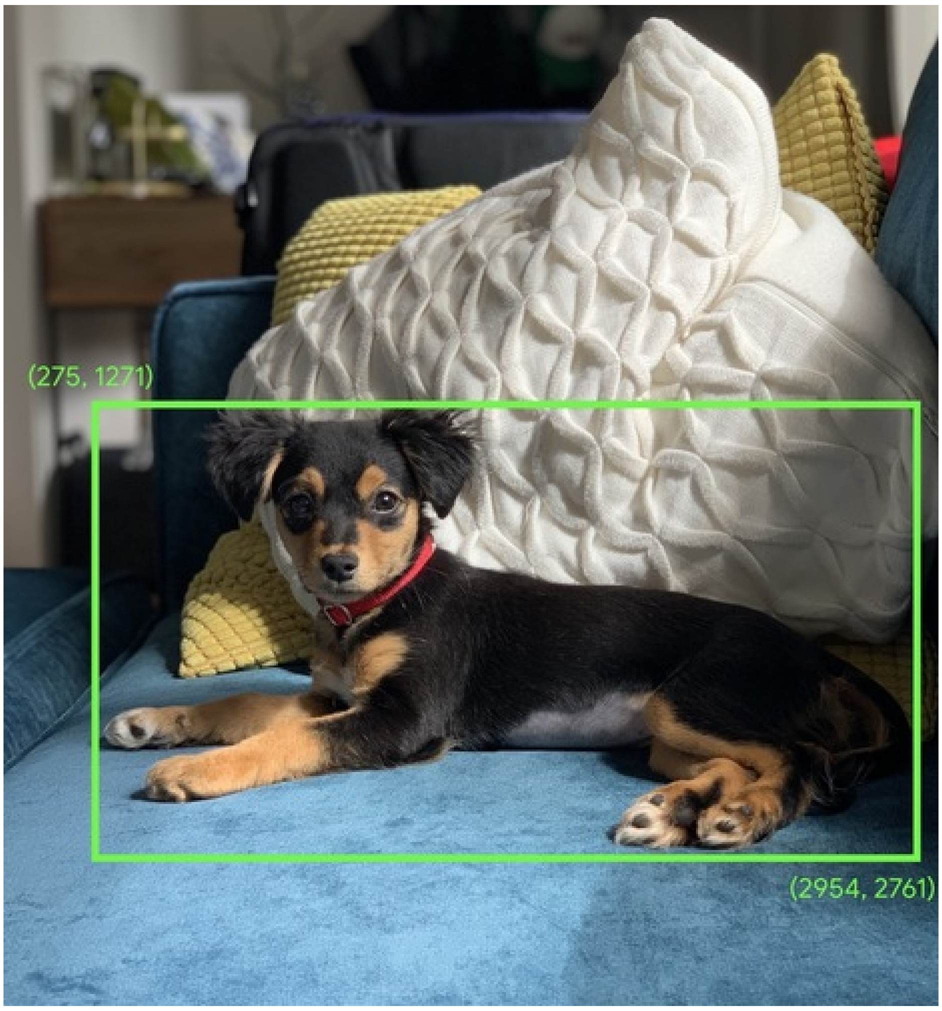
See **bucketing** (#bucketing).

boosting

A machine 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 (a "strong" classifier) by **upweighting** (#upweighting) the examples that the model is currently misclassfying.

bounding box

In an image, the (*x*, *y*) coordinates of a rectangle around an area of interest, such as the dog in the image below.



broadcasting

Expanding the shape of an operand in a matrix math operation to **dimensions** (#dimensions) compatible for that operation. For instance, linear algebra requires that the two operands in a matrix addition operation must have the same dimensions. Consequently, you can't add a matrix of shape (m, n) to a vector of length n. Broadcasting enables this operation by virtually expanding the vector of length n to a matrix of shape (m,n) by replicating the same values down each column.

For example, given the following definitions, linear algebra prohibits A+B because A and B have different dimensions:

```
A = [[7, 10, 4],
     [13, 5, 9]]
B = [2]
```

However, broadcasting enables the operation A+B by virtually expanding B to:

```
 [[2, 2, 2],
  [2, 2, 2]]
```

Thus, A+B is now a valid operation:

```
[[7, 10, 4],  +  [[2, 2, 2],  =  [[ 9, 12, 6],
 [13, 5, 9]]      [2, 2, 2]]      [15, 7, 11]]
```

See the following description of broadcasting in NumPy (https://docs.scipy.org/doc/numpy-1.15.0/user/basics.broadcasting.html) for more details.

bucketing

Converting a (usually **continuous** (#continuous_feature)) feature into multiple binary features called buckets or bins, typically based on value range. For example, instead of representing temperature as a single continuous floating-point feature, you could chop ranges of temperatures into discrete bins. Given temperature data sensitive to a tenth of a degree, all temperatures between 0.0 and 15.0 degrees could be put into one bin, 15.1 to 30.0 degrees could be a second bin, and 30.1 to 50.0 degrees could be a third bin.

C

calibration layer

A post-prediction adjustment, typically to account for **prediction bias** (#prediction_bias). The adjusted predictions and probabilities should match the distribution of an observed set of labels.

candidate generation

The initial set of recommendations chosen by a recommendation system. For example, consider a bookstore that offers 100,000 titles. The candidate generation phase creates a much smaller list of suitable books for a particular user, say 500. But even 500 books is way too many to recommend to a user. Subsequent, more expensive, phases of a recommendation system (such as **scoring** (#scoring) and **re-ranking** (#re-ranking)) whittle down those 500 to a much smaller, more useful set of recommendations.

candidate sampling

A training-time optimization in which a probability is calculated for all the positive labels, using, for example, **softmax** (#softmax), but only for a random sample of negative labels. For example, if we have an example labeled *beagle* and *dog* candidate sampling computes the predicted probabilities and corresponding loss terms for the *beagle* and *dog* class outputs in addition to a random subset of the remaining classes (*cat*, *lollipop*, *fence*). The idea is that the **negative classes** (#negative_class) can learn from less frequent negative reinforcement as long as **positive classes** (#positive_class) always get proper positive reinforcement, and this is indeed observed empirically. The motivation for candidate sampling is a computational efficiency win from not computing predictions for all negatives.

categorical data

Features (#feature) having a discrete set of possible values. For example, consider a categorical feature named `house style`, which has a discrete set of three possible values: `Tudor, ranch, colonial`. By representing `house style` as categorical data, the model can learn the separate impacts of `Tudor`, `ranch`, and `colonial` on house price.

Sometimes, values in the discrete set are mutually exclusive, and only one value can be applied to a given example. For example, a `car maker` categorical feature would probably permit only a single value (`Toyota`) per example. Other times, more than one value may be applicable. A single car could be painted more than one different color, so a `car color` categorical feature would likely permit a single example to have multiple values (for example, `red` and `white`).

Categorical features are sometimes called **discrete features** (#discrete_feature).

Contrast with **numerical data** (#numerical_data).

centroid

The center of a cluster as determined by a **k-means** (#k-means) or **k-median** (#k-median) algorithm. For instance, if k is 3, then the k-means or k-median algorithm finds 3 centroids.

centroid-based clustering

A category of **clustering** (#clustering) algorithms that organizes data into nonhierarchical clusters. **k-means** (#k-means) is the most widely used centroid-based clustering algorithm.

Contrast with **hierarchical clustering** (#hierarchical_clustering) algorithms.

checkpoint

Data that captures the state of the variables of a model at a particular time. Checkpoints enable exporting model **weights** (#weight), as well as performing training across multiple sessions. Checkpoints also enable training to continue past errors (for example, job preemption). Note that the **graph** (#graph) itself is not included in a checkpoint.

class

One of a set of enumerated target values for a label. For example, in a **binary classification** (#binary_classification) model that detects spam, the two classes are *spam* and *not spam*. In a **multi-class classification** (#multi-class) model that identifies dog breeds, the classes would be *poodle*, *beagle*, *pug*, and so on.

classification model

A type of machine learning model for distinguishing among two or more discrete classes. For example, a natural language processing classification model could determine whether an input sentence was in French, Spanish, or Italian. Compare with **regression model** (#regression_model).

classification threshold

A scalar-value criterion that is applied to a model's predicted score in order to separate the **positive class** (#positive_class) from the **negative class** (#negative_class). Used when mapping **logistic regression** (#logistic_regression) results to **binary classification** (#binary_classification). For example, 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*.

class-imbalanced dataset

A **binary classification** (#binary_classification) problem in which the **labels** (#label) for the two classes have significantly different frequencies. For example, a disease dataset in which 0.0001 of examples have positive labels and 0.9999 have negative labels is a class-imbalanced problem, but a football game predictor in which 0.51 of examples label one team winning and 0.49 label the other team winning is *not* a class-imbalanced problem.

clipping

A technique for handling **outliers** (#outliers). Specifically, reducing feature values that are greater than a set maximum value down to that maximum value. Also, increasing feature values that are less than a specific minimum value up to that minimum value.

For example, suppose that only a few feature values fall outside the range 40–60. In this case, you could do the following:

- Clip all values over 60 to be exactly 60.
- Clip all values under 40 to be exactly 40.

In addition to bringing *input values* within a designated range, clipping can also used to force *gradient values* within a designated range during training.

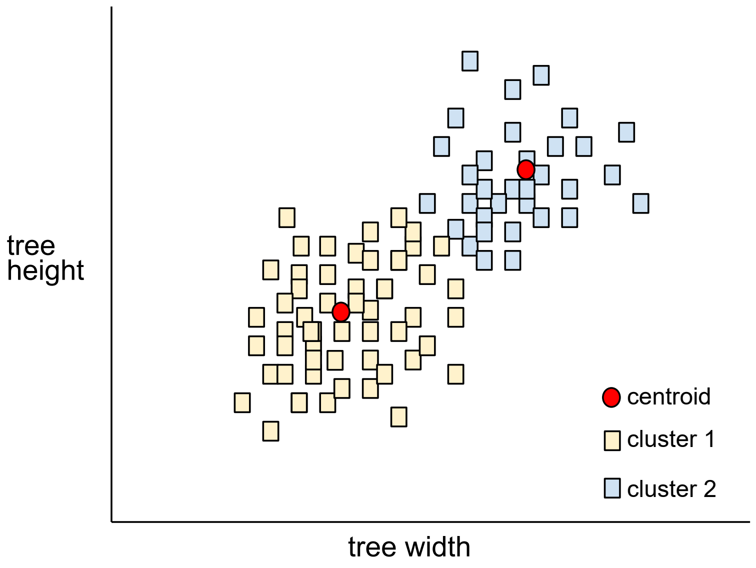
Cloud TPU

A specialized hardware accelerator designed to speed up machine learning workloads on Google Cloud Platform.

clustering

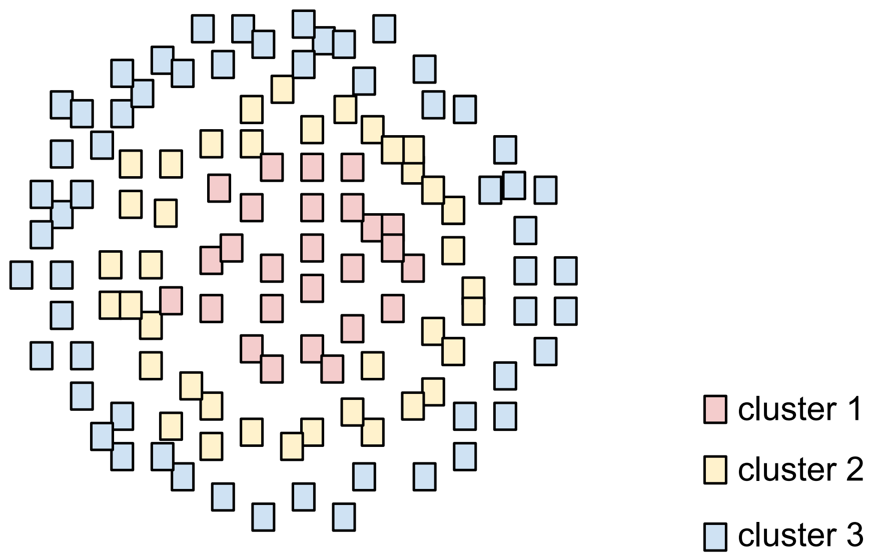
Grouping related **examples** (#example), particularly during **unsupervised learning** (#unsupervised_machine_learning). Once all the examples are grouped, a human can optionally supply meaning to each cluster.

Many clustering algorithms exist. For example, the **k-means** (#k-means) algorithm clusters examples based on their proximity to a **centroid** (#centroid), as in the following diagram:



A human researcher could then review the clusters and, for example, label cluster 1 as "dwarf trees" and cluster 2 as "full-size trees."

As another example, consider a clustering algorithm based on an example's distance from a center point, illustrated as follows:



co-adaptation

When **neurons** (#neuron) predict patterns in training data by relying almost exclusively on outputs of specific other neurons instead of relying on the network's behavior as a whole. When the patterns that cause co-adaption are not present in validation data, then co-adaptation causes overfitting. **Dropout regularization** (#dropout_regularization) reduces co-adaptation because dropout ensures neurons cannot rely solely on specific other neurons.

collaborative filtering

Making **predictions** (#prediction) about the interests of one user based on the interests of many other users. Collaborative filtering is often used in **recommendation systems** (#recommendation_system).

confirmation bias

The tendency to search for, interpret, favor, and recall information in a way that confirms one's preexisting beliefs or hypotheses. Machine learning developers may inadvertently collect or label data in ways that influence an outcome supporting their existing beliefs. Confirmation bias is a form of **implicit bias** (#implicit_bias).

Experimenter's bias is a form of confirmation bias in which an experimenter continues training models until a preexisting hypothesis is confirmed.

confusion matrix

An NxN table that summarizes how successful a **classification model's** (#classification_model) predictions were; that is, the correlation between the label and the model's classification. One axis of a confusion matrix is the **label** (#label) that the model predicted, and the other axis is the actual label. N represents the number of **classes** (#class). In a **binary classification** (#binary_classification) problem, N=2. For example, here is a sample confusion matrix for a binary classification problem:

	Tumor (predicted)	**Non-Tumor (predicted)**
Tumor (actual)	18	1
Non-Tumor (actual)	6	452

The preceding confusion matrix shows that of the 19 samples that actually had tumors, the model correctly classified 18 as having tumors (18 **true positives** (#TP)), and incorrectly classified 1 as not having a tumor (1 **false negative** (#FN)). Similarly, of 458 samples that actually did not have tumors, 452 were correctly classified (452 **true negatives** (#TN)) and 6 were incorrectly classified (6 **false positives** (#FP)).

The confusion matrix for a **multi-class classification** (#multi-class) problem can help you determine mistake patterns. For example, a confusion matrix could reveal that a model trained to recognize handwritten digits tends to mistakenly predict 9 instead of 4, or 1 instead of 7.

Confusion matrices contain sufficient information to calculate a variety of performance metrics, including **precision** (#precision) and **recall** (#recall).

continuous feature

A floating-point feature with an infinite range of possible values. Contrast with **discrete feature** (#discrete_feature).

convenience sampling

Using a dataset not gathered scientifically in order to run quick experiments. Later on, it's essential to switch to a scientifically gathered dataset.

convergence

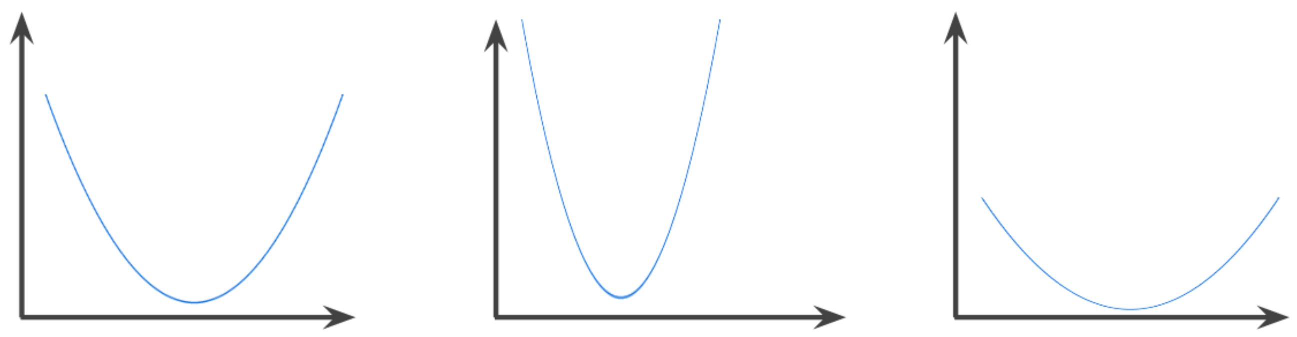
Informally, often refers to a state reached during **training** (#training) in which training **loss** (#loss) and **validation** (#validation) loss change very little or not at all with each iteration after a certain number of iterations. In other words, a model reaches convergence when additional training on the current data will not improve the model. In **deep learning** (#deep_model), loss values sometimes stay constant or nearly so for many iterations before finally descending, temporarily producing a false sense of convergence.

See also **early stopping** (#early_stopping).

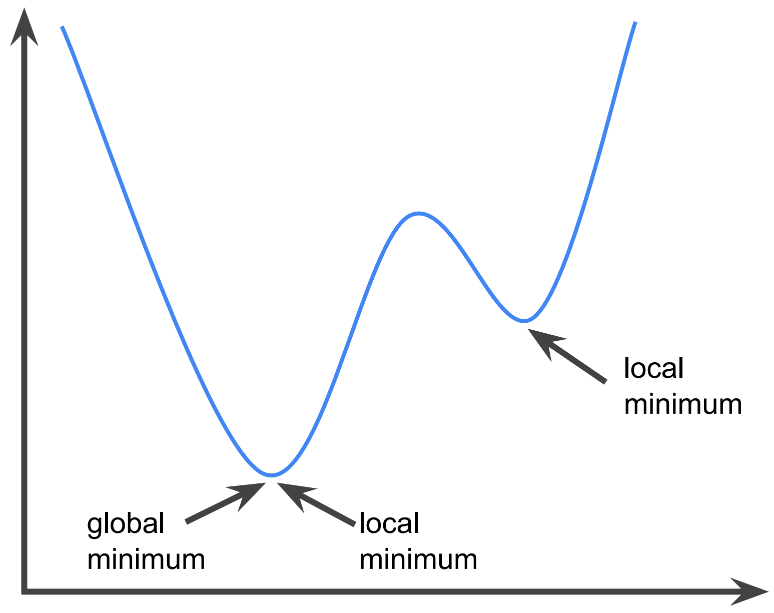
See also Boyd and Vandenberghe, Convex Optimization (https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf).

convex function

A function in which the region above the graph of the function is a **convex set** (#convex_set). The prototypical convex function is shaped something like the letter **U**. For example, the following are all convex functions:



By contrast, the following function is not convex. Notice how the region above the graph is not a convex set:



A **strictly convex function** has exactly one local minimum point, which is also the global minimum point. The classic U-shaped functions are strictly convex functions. However, some convex functions (for example, straight lines) are not U-shaped.

A lot of the common **loss functions** (#loss), including the following, are convex functions:

- **L_2 loss** (#L2_loss)
- **Log Loss** (#Log_Loss)
- **L_1 regularization** (#L1_regularization)
- **L_2 regularization** (#L2_regularization)

Many variations of **gradient descent** (#gradient_descent) are guaranteed to find a point close to the minimum of a strictly convex function. Similarly, many variations of **stochastic gradient descent** (#SGD) have a high probability (though, not a guarantee) of finding a point close to the minimum of a strictly convex function.

The sum of two convex functions (for example, L_2 loss + L_1 regularization) is a convex function.

Deep models (#deep_model) are never convex functions. Remarkably, algorithms designed for **convex optimization** (#convex_optimization) tend to find reasonably good solutions on deep networks anyway, even though those solutions are not guaranteed to be a global minimum.

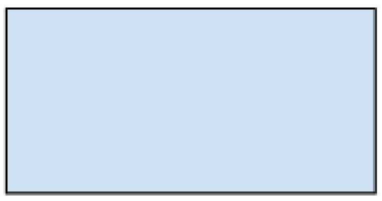
convex optimization

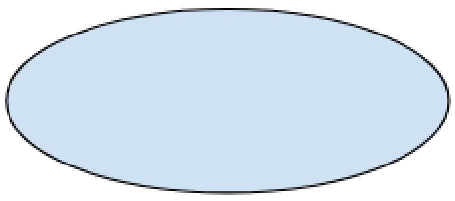
The process of using mathematical techniques such as **gradient descent** (#gradient_descent) to find the minimum of a **convex function** (#convex_function). A great deal of research in machine learning has focused on formulating various problems as convex optimization problems and in solving those problems more efficiently.

For complete details, see Boyd and Vandenberghe, Convex Optimization (https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf).

convex set

A subset of Euclidean space such that a line drawn between any two points in the subset remains completely within the subset. For instance, the following two shapes are convex sets:





By contrast, the following two shapes are not convex sets:

## convolution

In mathematics, casually speaking, a mixture of two functions. In machine learning, a convolution mixes the convolutional filter and the input matrix in order to train **weights** (#weight).

The term "convolution" in machine learning is often a shorthand way of referring to either **convolutional operation** (#convolutional_operation) or **convolutional layer** (#convolutional_layer).

Without convolutions, a machine learning algorithm would have to learn a separate weight for every cell in a large **tensor** (#tensor). For example, a machine learning algorithm training on 2K x 2K images would be forced to find 4M separate weights. Thanks to convolutions, a machine learning algorithm only has to find weights for every cell in the **convolutional filter** (#convolutional_filter), dramatically reducing the memory needed to train the model. When the convolutional filter is applied, it is simply replicated across cells such that each is multiplied by the filter.

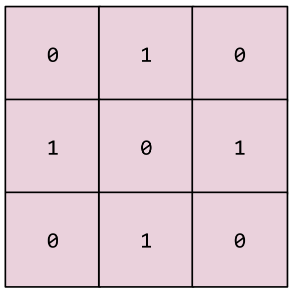
convolutional filter

One of the two actors in a **convolutional operation** (#convolutional_operation). (The other actor is a slice of an input matrix.) A convolutional filter is a matrix having the same **rank** (#rank) as the input matrix, but a smaller shape. For example, given a 28x28 input matrix, the filter could be any 2D matrix smaller than 28x28.

In photographic manipulation, all the cells in a convolutional filter are typically set to a constant pattern of ones and zeroes. In machine learning, convolutional filters are typically seeded with random numbers and then the network **trains** (#training) the ideal values.

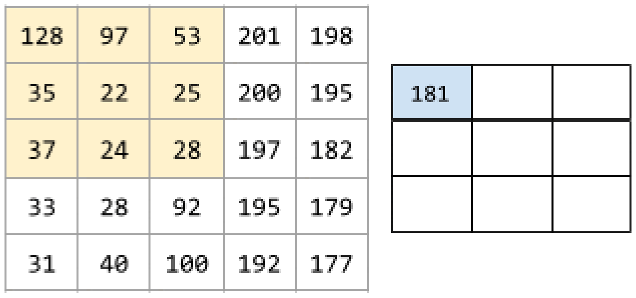
## convolutional layer

A layer of a **deep neural network** (#deep_model) in which a **convolutional filter** (#convolutional_filter) passes along an input matrix. For example, consider the following 3x3 **convolutional filter** (#convolutional_filter):



0	1	0
1	0	1
0	1	0

The following animation shows a convolutional layer consisting of 9 convolutional operations involving the 5x5 input matrix. Notice that each convolutional operation works on a different 3x3 slice of the input matrix. The resulting 3x3 matrix (on the right) consists of the results of the 9 convolutional operations:



128	97	53	201	198
35	22	25	200	195
37	24	28	197	182
33	28	92	195	179
31	40	100	192	177

181		

convolutional neural network

A **neural network** (#neural_network) in which at least one layer is a **convolutional layer** (#convolutional_layer). A typical convolutional neural network consists of some combination of the following layers:

- **convolutional layers** (#convolutional_layer)

- **pooling layers** (#pooling)
- **dense layers** (#dense_layer)

Convolutional neural networks have had great success in certain kinds of problems, such as image recognition.

convolutional operation

The following two-step mathematical operation:

1. Element-wise multiplication of the **convolutional filter** (#convolutional_filter) and a slice of an input matrix. (The slice of the input matrix has the same rank and size as the convolutional filter.)
2. Summation of all the values in the resulting product matrix.

For example, consider the following 5x5 input matrix:

128	97	53	201	198
35	22	25	200	195
37	24	28	197	182
33	28	92	195	179
31	40	100	192	177

Now imagine the following 2x2 convolutional filter:

1	0
0	1

Each convolutional operation involves a single 2x2 slice of the input matrix. For instance, suppose we use the 2x2 slice at the top-left of the input matrix. So, the convolution operation on this slice looks as follows:

$$\begin{bmatrix} 128 & 97 \\ 35 & 22 \end{bmatrix} \times \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \Rightarrow \begin{bmatrix} 128 & 0 \\ 0 & 22 \end{bmatrix} = \boxed{128+22=150}$$

A **convolutional layer** (#convolutional_layer) consists of a series of convolutional operations, each acting on a different slice of the input matrix.

cost

Synonym for **loss** (#loss).

counterfactual fairness

A **fairness metric** (#fairness_metric) that checks whether a classifier produces the same result for one individual as it does for another individual who is identical to the first, except with respect to one or more **sensitive attributes** (#sensitive_attribute). Evaluating a classifier for counterfactual fairness is one method for surfacing potential sources of bias in a model.

See "When Worlds Collide: Integrating Different Counterfactual Assumptions in Fairness" (https://papers.nips.cc/paper/7220-when-worlds-collide-integrating-different-counterfactual-assumptions-in-fairness.pdf) for a more detailed discussion of counterfactual fairness.

coverage bias

See **selection bias** (#selection_bias).

crash blossom

A sentence or phrase with an ambiguous meaning. Crash blossoms present a significant problem in **natural language understanding** (#natural_language_understanding). For example, the headline *Red Tape Holds Up Skyscraper* is a crash blossom because an NLU model could interpret the headline literally or figuratively.

critic RL

Synonym for **Deep Q-Network** (#deep_q-network).

cross-entropy

A generalization of **Log Loss** (#Log_Loss) to **multi-class classification problems** (#multi-class). Cross-entropy quantifies the difference between two probability distributions. See also **perplexity** (#perplexity).

cross-validation

A mechanism for estimating how well a model will generalize to new data by testing the model against one or more non-overlapping data subsets withheld from the **training set** (#training_set).

custom Estimator

An **Estimator** (#Estimators) that you write yourself by following these directions (https://www.tensorflow.org/guide/custom_estimators).

Contrast with **premade Estimators** (#premade_Estimator).

D

data analysis

Obtaining an understanding of data by considering samples, measurement, and visualization. Data analysis can be particularly useful when a dataset is first received, before one builds the first model. It is also crucial in understanding experiments and debugging problems with the system.

data augmentation

Artificially boosting the range and number of **training** (#training) examples by transforming existing examples to create additional examples. For example, suppose images are one of your features, but your dataset doesn't contain enough image examples for the model to learn useful associations. Ideally, you'd add enough **labeled** (#label) images to your dataset to enable your model to train properly. If that's not possible, data augmentation can rotate, stretch, and reflect each image to produce many variants of the original picture, possibly yielding enough labeled data to enable excellent training.

DataFrame

A popular datatype for representing datasets in **pandas** (#pandas). A DataFrame is analogous to a table. Each column of the DataFrame has a name (a header), and each row is identified by a number.

data set or dataset

A collection of **examples** (#example).

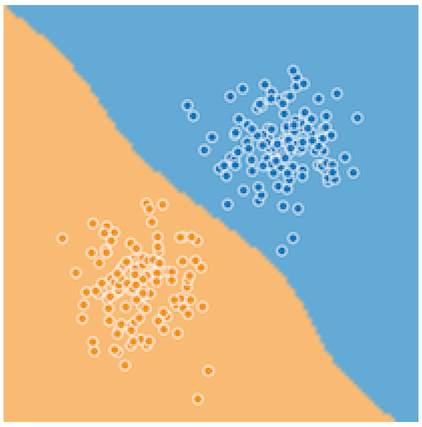
Dataset API (tf.data)

A high-level **TensorFlow** (#TensorFlow) API for reading data and transforming it into a form that a machine learning algorithm requires. A `tf.data.Dataset` object represents a sequence of elements, in which each element contains one or more **Tensors** (#tensor). A `tf.data.Iterator` object provides access to the elements of a `Dataset`.

For details about the Dataset API, see Importing Data (https://www.tensorflow.org/programmers_guide/datasets) in the TensorFlow Programmer's Guide.

decision boundary

The separator between classes learned by a model in a **binary class** (#binary_classification) or **multi-class classification problems** (#multi-class). For example, in the following image representing a binary classification problem, the decision boundary is the frontier between the orange class and the blue class:

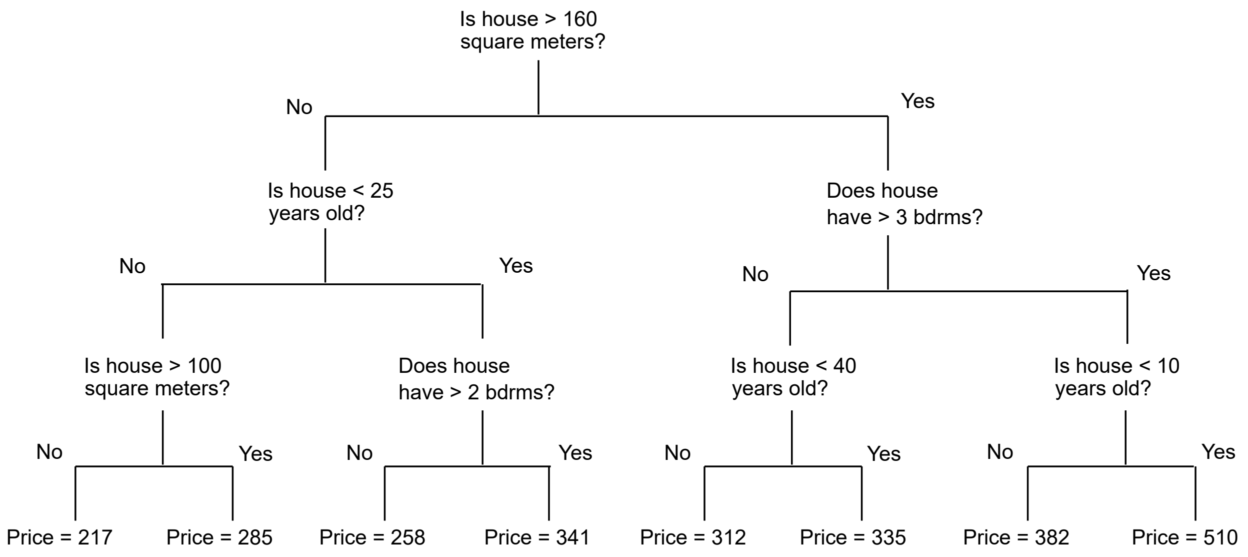


decision threshold

Synonym for **classification threshold** (#classification_threshold).

decision tree

A model represented as a sequence of branching statements. For example, the following over-simplified decision tree branches a few times to predict the price of a house (in thousands of USD). According to this decision tree, a house larger than 160 square meters, having more than three bedrooms, and built less than 10 years ago would have a predicted price of 510 thousand USD.



Machine learning can generate deep decision trees.

deep model

A type of **neural network** (#neural_network) containing multiple **hidden layers** (#hidden_layer).

Contrast with **wide model** (#wide_model).

deep neural network

Synonym for **deep model** (#deep_model).

Deep Q-Network (DQN) RL

In **Q-learning** (#q-learning), a deep **neural network** (#neural_network) that predicts **Q-functions** (#q-function).

Critic is a synonym for Deep Q-Network.

demographic parity

A **fairness metric** (#fairness_metric) that is satisfied if the results of a model's classification are not dependent on a given **sensitive attribute** (#sensitive_attribute).

For example, if both Lilliputians and Brobdingnagians apply to Glubbdubdrib University, demographic parity is achieved if the percentage of Lilliputians admitted is the same as the percentage of Brobdingnagians admitted, irrespective of whether one group is on average more qualified than the other.

Contrast with **equalized odds** (#equalized_odds) and **equality of opportunity** (#equality_of_opportunity), which permit classification results in aggregate to depend on sensitive attributes, but do not permit classification results for certain specified ground-truth labels to depend on sensitive attributes. See "Attacking discrimination with smarter machine learning" (http://research.google.com/bigpicture/attacking-discrimination-in-ml/) for a visualization exploring the tradeoffs when optimizing for demographic parity.

dense feature

A **feature** (#feature) in which most values are non-zero, typically a **Tensor** (#tensor) of floating-point values. Contrast with **sparse feature** (#sparse_features).

dense layer

Synonym for **fully connected layer** (#fully_connected_layer).

depth

The number of **layers** (#layer) (including any **embedding** (#embeddings) layers) in a **neural network** (#neural_network) that learn weights. For example, a neural network with 5 **hidden layers** (#hidden_layer) and 1 output layer has a depth of 6.

depthwise separable convolutional neural network (sepIN

A **convolutional neural network** (#convolutional_neural_network) architecture based on Inception (https://github.com/tensorflow/tpu/tree/master/models/experimental/inception), but where Inception modules are replaced with depthwise separable convolutions. Also known as Xception.

A depthwise separable convolution (also abbreviated as separable convolution) factors a standard 3-D convolution into two separate convolution operations that are more computationally efficient: first, a depthwise convolution, with a depth of 1 ($n \times n \times 1$), and then second, a pointwise convolution, with length and width of 1 ($1 \times 1 \times n$).

To learn more, see Xception: Deep Learning with Depthwise Separable Convolutions (https://arxiv.org/pdf/1610.02357.pdf).

device

A category of hardware that can run a TensorFlow session, including CPUs, GPUs, and **TPUs** (#TPU).

dimension reduction

Decreasing the number of dimensions used to represent a particular feature in a feature vector, typically by converting to an **embedding** (#embeddings).

dimensions

Overloaded term having any of the following definitions:

- The number of levels of coordinates in a **Tensor** (#tensor). For example:
    - A scalar has zero dimensions; for example, `["Hello"]`.
    - A vector has one dimension; for example, `[3, 5, 7, 11]`.
    - A matrix has two dimensions; for example, `[[2, 4, 18], [5, 7, 14]]`.

  You can uniquely specify a particular cell in a one-dimensional vector with one coordinate; you need two coordinates to uniquely specify a particular cell in a two-dimensional matrix.
- The number of entries in a **feature vector** (#feature_vector).
- The number of elements in an **embedding** (#embeddings) layer.

discrete feature

A **feature** (#feature) with a finite set of possible values. For example, a feature whose values may only be *animal*, *vegetable*, or *mineral* is a discrete (or categorical) feature. Contrast with **continuous feature** (#continuous_feature).

discriminative model

A **model** (#model) that predicts labels from a set of one or more features. More formally, discriminative models define the conditional probability of an output given the features and weights; that is:

```
p(output | features, weights)
```

For example, a model that predicts whether an email is spam from features and weights is a discriminative model.

The vast majority of supervised learning models, including classification and regression models, are discriminative models.

Contrast with **generative model** (#generative_model).

discriminator

A system that determines whether examples are real or fake.

The subsystem within a **generative adversarial network** (#generative_adversarial_network) that determines whether the examples created by the **generator** (#generator) are real or fake.

disparate impact

Making decisions about people that impact different population subgroups disproportionately. This usually refers to situations where an algorithmic decision-making process harms or benefits some subgroups more than others.

For example, suppose an algorithm that determines a Lilliputian's eligibility for a miniature-home loan is more likely to classify them as “ineligible” if their mailing address contains a certain postal code. If Big-Endian Lilliputians are more likely to have mailing addresses with this postal code than Little-Endian Lilliputians, then this algorithm may result in disparate impact.

Contrast with **disparate treatment** (#disparate_treatment), which focuses on disparities that result when subgroup characteristics are explicit inputs to an algorithmic decision-making process.

disparate treatment

Factoring subjects' **sensitive attributes** (#sensitive_attribute) into an algorithmic decision-making process such that different subgroups of people are treated differently.

For example, consider an algorithm that determines Lilliputians’ eligibility for a miniature-home loan based on the data they provide in their loan application. If the algorithm uses a Lilliputian’s affiliation as Big-Endian or Little-Endian as an input, it is enacting disparate treatment along that dimension.

Contrast with **disparate impact** (#disparate_impact), which focuses on disparities in the societal impacts of algorithmic decisions on subgroups, irrespective of whether those subgroups are inputs to the model.

ıg: Because sensitive attributes are almost always correlated with other features the data may have, expli
ing sensitive attribute information does not guarantee that subgroups will be treated equally. For example
ing sensitive demographic attributes from a training data set that still includes postal code as a feature m
ss disparate treatment of subgroups, but there still might be disparate impact upon these groups because
code might serve as a **proxy** (#proxy_sensitive_attributes) for other demographic information.

divisive clustering

See **hierarchical clustering** (#hierarchical_clustering).

downsampling

Overloaded term that can mean either of the following:

- Reducing the amount of information in a feature in order to train a model more efficiently. For example, before training an image recognition model, downsampling high-resolution images to a lower-resolution format.

- Training on a disproportionately low percentage of over-represented class examples in order to improve model training on under-represented classes. For example, in a **class-imbalanced dataset** (#class_imbalanced_data_set), models tend to learn a lot about the **majority class** (#majority_class) and not enough about the **minority class** (#minority_class). Downsampling helps balance the amount of training on the majority and minority classes.

DQN

RL

Abbreviation for **Deep Q-Network** (#deep_q-network).

dropout regularization

A form of **regularization** (#regularization) useful in training **neural networks** (#neural_network). Dropout regularization works by removing a random selection of a fixed number of the units in a network layer for a single gradient step. The more units dropped out, the stronger the regularization. This is analogous to training the network to emulate an exponentially large ensemble of smaller networks. For full details, see Dropout: A Simple Way to Prevent Neural Networks from Overfitting (http://jmlr.org/papers/volume15/srivastava14a.old/srivastava14a.pdf).

dynamic model

A **model** (#model) that is trained online in a continuously updating fashion. That is, data is continuously entering the model.

E

eager execution

A TensorFlow programming environment in which **operations** (#Operation) run immediately. By contrast, operations called in **graph execution** (#graph_execution) don't run until they are explicitly evaluated. Eager execution is an imperative interface (https://wikipedia.org/wiki/Imperative_programming), much like the code in most programming languages. Eager execution programs are generally far easier to debug than graph execution programs.

early stopping

A method for **regularization** (#regularization) that involves ending model training *before* training loss finishes decreasing. In early stopping, you end model training when the loss on a **validation dataset** (#validation_set) starts to increase, that is, when **generalization** (#generalization) performance worsens.

embeddings

A categorical feature represented as a continuous-valued feature. Typically, an embedding is a translation of a high-dimensional vector into a low-dimensional space. For example, you can represent the words in an English sentence in either of the following two ways:

- As a million-element (high-dimensional) **sparse vector** (#sparse_features) in which all elements are integers. Each cell in the vector represents a separate English word; the value in a cell represents the number of times that word appears in a sentence. Since a single English sentence is unlikely to contain more than 50 words, nearly every cell in the vector will contain a 0. The few cells that aren't 0 will contain a low integer (usually 1) representing the number of times that word appeared in the sentence.

- As a several-hundred-element (low-dimensional) **dense vector** (#dense_feature) in which each element holds a floating-point value between 0 and 1. This is an embedding.

In TensorFlow, embeddings are trained by **backpropagating** (#backpropagation) **loss** (#loss) just like any other parameter in a **neural network** (#neural_network).

embedding space

The d-dimensional vector space that features from a higher-dimensional vector space are mapped to. Ideally, the embedding space contains a structure that yields meaningful mathematical results; for example, in an ideal embedding space, addition and subtraction of embeddings can solve word analogy tasks.

The dot product (https://wikipedia.org/wiki/Dot_product) of two embeddings is a measure of their similarity.

empirical risk minimization (ERM)

Choosing the function that minimizes loss on the training set. Contrast with **structural risk minimization** (#SRM).

ensemble

A merger of the predictions of multiple **models** (#model). You can create an ensemble via one or more of the following:

- different initializations

- different **hyperparameters** (#hyperparameter)
- different overall structure

Deep and wide models (https://www.tensorflow.org/tutorials/wide_and_deep) are a kind of ensemble.

environment RL

In reinforcement learning, the world that contains the **agent** (#agent) and allows the agent to observe that world's **state** (#state). For example, the represented world can be a game like chess, or a physical world like a maze. When the agent applies an **action** (#action) to the environment, then the environment transitions between states.

episode RL

In reinforcement learning, each of the repeated attempts by the **agent** (#agent) to learn an **environment** (#environment).

epoch

A full training pass over the entire dataset such that each example has been seen once. Thus, an epoch represents `N`/**batch size** (#batch_size) training **iterations** (#iteration), where `N` is the total number of examples.

epsilon greedy policy RL

In reinforcement learning, a **policy** (#policy) that either follows a **random policy** (#random_policy) with epsilon probability or a **greedy policy** (#greedy_policy) otherwise. For example, if epsilon is 0.9, then the policy follows a random policy 90% of the time and a greedy policy 10% of the time.

Over successive episodes, the algorithm reduces epsilon’s value in order to shift from following a random policy to following a greedy policy. By shifting the policy, the agent first randomly explores the environment and then greedily exploits the results of random exploration.

equality of opportunity

A **fairness metric** (#fairness_metric) that checks whether, for a preferred **label** (#label) (one that confers an advantage or benefit to a person) and a given **attribute** (#attribute), a classifier predicts that preferred label equally well for all values of that attribute. In other words, equality of opportunity measures whether the people who should qualify for an opportunity are equally likely to do so regardless of their group membership.

For example, suppose Glubbdubdrib University admits both Lilliputians and Brobdingnagians to a rigorous mathematics program. Lilliputians’ secondary schools offer a robust curriculum of math classes, and the vast majority of students are qualified for the university program. Brobdingnagians’ secondary schools don’t offer math classes at all, and as a result, far fewer of their students are qualified. Equality of opportunity is satisfied for the preferred label of "admitted" with respect to nationality (Lilliputian or Brobdingnagian) if qualified students are equally likely to be admitted irrespective of whether they're a Lilliputian or a Brobdingnagian.

For example, let's say 100 Lilliputians and 100 Brobdingnagians apply to Glubbdubdrib University, and admissions decisions are made as follows:

Table 1. Lilliputian applicants (90% are qualified)

	Qualified	Unqualified
Admitted	45	3
Rejected	45	7
Total	90	10

Percentage of qualified students admitted: 45/90 = 50%
Percentage of unqualified students rejected: 7/10 = 70%
Total percentage of Lilliputian students admitted: (45+3)/100 = 48%

Table 2. Brobdingnagian applicants (10% are qualified):

	Qualified	Unqualified
Admitted	5	9
Rejected	5	81
Total	10	90

Percentage of qualified students admitted: 5/10 = 50%
Percentage of unqualified students rejected: 81/90 = 90%
Total percentage of Brobdingnagian students admitted: (5+9)/100 = 14%

The preceding examples satisfy equality of opportunity for acceptance of qualified students because qualified Lilliputians and Brobdingnagians both have a 50% chance of being admitted.

While equality of opportunity is satisfied, the following two fairness metrics are not satisfied:

demographic parity (#demographic_parity): Lilliputians and Brobdingnagians are admitted to the universi different rates; 48% of Lilliputians students are admitted, but only 14% of Brobdingnagian students are admitted.

equalized odds (#equalized_odds): While qualified Lilliputian and Brobdingnagian students both have the chance of being admitted, the additional constraint that unqualified Lilliputians and Brobdingnagians bot the same chance of being rejected is not satisfied. Unqualified Lilliputians have a 70% rejection rate, wher unqualified Brobdingnagians have a 90% rejection rate.

See "Equality of Opportunity in Supervised Learning" (https://arxiv.org/pdf/1610.02413.pdf) for a more detailed discussion of equality of opportunity. Also see "Attacking discrimination with smarter machine learning" (http://research.google.com/bigpicture/attacking-discrimination-in-ml/) for a visualization exploring the tradeoffs when optimizing for equality of opportunity.

equalized odds

A **fairness metric** (#fairness_metric) that checks if, for any particular label and attribute, a classifier predicts that label equally well for all values of that attribute.

For example, suppose Glubbdubdrib University admits both Lilliputians and Brobdingnagians to a rigorous mathematics program. Lilliputians' secondary schools offer a robust curriculum of math classes, and the vast majority of students are qualified for the university program. Brobdingnagians' secondary schools don't offer math classes at all, and as a result, far fewer of their students are qualified. Equalized odds is satisfied provided that no matter whether an applicant is a Lilliputian or a Brobdingnagian, if they are qualified, they are equally as likely to get admitted to the program, and if they are not qualified, they are equally as likely to get rejected.

Let's say 100 Lilliputians and 100 Brobdingnagians apply to Glubbdubdrib University, and admissions decisions are made as follows:

Table 3. Lilliputian applicants (90% are qualified)

	Qualified	**Unqualified**
Admitted	45	2
Rejected	45	8
Total	90	10

Percentage of qualified students admitted: 45/90 = 50%
Percentage of unqualified students rejected: 8/10 = 80%
Total percentage of Lilliputian students admitted: (45+2)/100 = 47%

Table 4. Brobdingnagian applicants (10% are qualified):

	Qualified	**Unqualified**
Admitted	5	18
Rejected	5	72

Total	10	90

Percentage of qualified students admitted: 5/10 = 50%
Percentage of unqualified students rejected: 72/90 = 80%
Total percentage of Brobdingnagian students admitted: (5+18)/100 = 23%

Equalized odds is satisfied because qualified Lilliputian and Brobdingnagian students both have a 50% chance of being admitted, and unqualified Lilliputian and Brobdingnagian have an 80% chance of being rejected.

Vhile equalized odds is satisfied here, **demographic parity** (#demographic_parity) is *not satisfied*. Lilliputi
obdingnagian students are admitted to Glubbdubdrib University at different rates; 47% of Lilliputian stude
mitted, and 23% of Brobdingnagian students are admitted.

Equalized odds is formally defined in "Equality of Opportunity in Supervised Learning" (https://arxiv.org/pdf/1610.02413.pdf) as follows: "predictor Ŷ satisfies equalized odds with respect to protected attribute A and outcome Y if Ŷ and A are independent, conditional on Y."

Contrast equalized odds with the more relaxed **equality of opportunity** (#equality_of_opportunity) metric.

Estimator

An instance of the `tf.Estimator` class, which encapsulates logic that builds a TensorFlow graph and runs a TensorFlow session. You may create your own **custom Estimators** (#custom_estimator) (as described here (https://www.tensorflow.org/extend/estimators)) or instantiate **premade Estimators** (#premade_Estimator) created by others.

example

One row of a dataset. An example contains one or more **features** (#feature) and possibly a **label** (#label). See also **labeled example** (#labeled_example) and **unlabeled example** (#unlabeled_example).

experience replay

RL

In reinforcement learning, a **DQN** (#deep_q-network) technique used to reduce temporal correlations in training data. The **agent** (#agent) stores state transitions in a **replay buffer** (#replay_buffer), and then samples transitions from the replay buffer to create training data.

experimenter's bias

See **confirmation bias** (#confirmation_bias).

exploding gradient problem

The tendency for **gradients** (#gradient) in a **deep neural networks** (#deep_neural_network) (especially **recurrent neural networks** (#recurrent_neural_network)) to become surprisingly steep (high). Steep gradients result in very large updates to the weights of each node in a deep neural network.

Models suffering from the exploding gradient problem become difficult or impossible to train. **Gradient clipping** (#gradient_clipping) can mitigate this problem.

Compare to **vanishing gradient problem** (#vanishing_gradient_problem).

F

fairness constraint

Applying a constraint to an algorithm to ensure one or more definitions of fairness are satisfied. Examples of fairness constraints include:

- **Post-processing** (#post-processing) your model's output.
- Altering the **loss function** (#loss) to incorporate a penalty for violating a **fairness metric** (#fairness_metric).
- Directly adding a mathematical constraint to an optimization problem.

fairness metric

A mathematical definition of “fairness” that is measurable. Some commonly used fairness metrics include:

- **equalized odds** (#equalized_odds)
- **predictive parity** (#predictive_parity)
- **counterfactual fairness** (#counterfactual_fairness)
- **demographic parity** (#demographic_parity)

Many fairness metrics are mutually exclusive; see **incompatibility of fairness metrics** (#incompatibility_of_fairness_metrics).

false negative (FN)

An example in which the model mistakenly predicted the **negative class** (#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)

An example in which the model mistakenly predicted the **positive class** (#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.

false positive rate (FPR)

The x-axis in an **ROC curve** (#ROC). The false positive rate is defined as follows:

$$\text{False Positive Rate} = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

feature

An input variable used in making **predictions** (#prediction).

Feature column (tf.feature_column)

A function that specifies how a model should interpret a particular feature. A list that collects the output returned by calls to such functions is a required parameter to all **Estimators** (#Estimators) constructors.

The `tf.feature_column` functions enable models to easily experiment with different representations of input features. For details, see the Feature Columns chapter (https://www.tensorflow.org/guide/feature_columns) in the TensorFlow Programmers Guide.

"Feature column" is Google-specific terminology. A feature column is referred to as a "namespace" in the VW (https://wikipedia.org/wiki/Vowpal_Wabbit) system (at Yahoo/Microsoft), or a field (https://www.csie.ntu.edu.tw/~cjlin/libffm/).

feature cross

A **synthetic feature** (#synthetic_feature) formed by crossing (taking a Cartesian product (https://wikipedia.org/wiki/Cartesian_product) of) individual binary features obtained from **categorical data** (#categorical_data) or from **continuous features** (#continuous_feature) via **bucketing** (#bucketing). Feature crosses help represent nonlinear relationships.

feature engineering

The process of determining which **features** (#feature) might be useful in training a model, and then converting raw data from log files and other sources into said features. In TensorFlow, feature engineering often means converting raw log file entries to **tf.Example** (#tf.Example) protocol buffers. See also tf.Transform (https://github.com/tensorflow/transform).

Feature engineering is sometimes called **feature extraction**.

feature extraction

Overloaded term having either of the following definitions:

- Retrieving intermediate feature representations calculated by an **unsupervised** (#unsupervised_machine_learning) or pretrained model (for example, **hidden layer** (#hidden_layer) values in a **neural network** (#neural_network)) for use in another model as input.
- Synonym for **feature engineering** (#feature_engineering).

feature set

The group of **features** (#feature) your machine learning model trains on. For example, postal code, property size, and property condition might comprise a simple feature set for a model that predicts housing prices.

feature spec

Describes the information required to extract **features** (#feature) data from the **tf.Example** (#tf.Example) protocol buffer. Because the tf.Example protocol buffer is just a container for data, you must specify the following:

- the data to extract (that is, the keys for the features)
- the data type (for example, float or int)
- The length (fixed or variable)

The **Estimator API** (#Estimators) provides facilities for producing a feature spec from a list of **FeatureColumns** (#feature_columns).

feature vector

The list of feature values representing an **example** (#example) passed into a model.

federated learning

A distributed machine learning approach that **trains** (#training) machine learning **models** (#model) using decentralized **examples** (#example) residing on devices such as smartphones. In federated learning, a subset of devices downloads the current model from a central coordinating server. The devices use the examples stored on the devices to make improvements to the model. The devices then upload the model improvements (but not the training examples) to the coordinating server, where they are aggregated with other updates to yield an improved global model. After the aggregation, the model updates computed by devices are no longer needed, and can be discarded.

Since the training examples are never uploaded, federated learning follows the privacy principles of focused data collection and data minimization.

For more information about federated learning, see this tutorial (https://federated.withgoogle.com).

feedback loop

In machine learning, a situation in which a model's predictions influence the training data for the same model or another model. For example, a model that recommends movies will influence the movies that people see, which will then influence subsequent movie recommendation models.

feedforward neural network (FFN)

A neural network without cyclic or recursive connections. For example, traditional **deep neural networks** (#deep_neural_network) are feedforward neural networks. Contrast with

recurrent neural networks (#recurrent_neural_network), which are cyclic.

few-shot learning

A machine learning approach, often used for object classification, designed to learn effective classifiers from only a small number of training examples.

See also **one-shot learning** (#one-shot_learning).

fine tuning

Perform a secondary optimization to adjust the parameters of an already trained **model** (#model) to fit a new problem. Fine tuning often refers to refitting the weights of a trained **unsupervised** (#unsupervised_machine_learning) model to a **supervised** (#supervised_machine_learning) model.

forget gate

The portion of a **Long Short-Term Memory** (#Long_Short-Term_Memory) cell that regulates the flow of information through the cell. Forget gates maintain context by deciding which information to discard from the cell state.

full softmax

See **softmax** (#softmax). Contrast with **candidate sampling** (#candidate_sampling).

fully connected layer

A **hidden layer** (#hidden_layer) in which each **node** (#node) is connected to *every* node in the subsequent hidden layer.

A fully connected layer is also known as a **dense layer** (#dense_layer).

G

GAN

Abbreviation for **generative adversarial network** (#generative_adversarial_network).

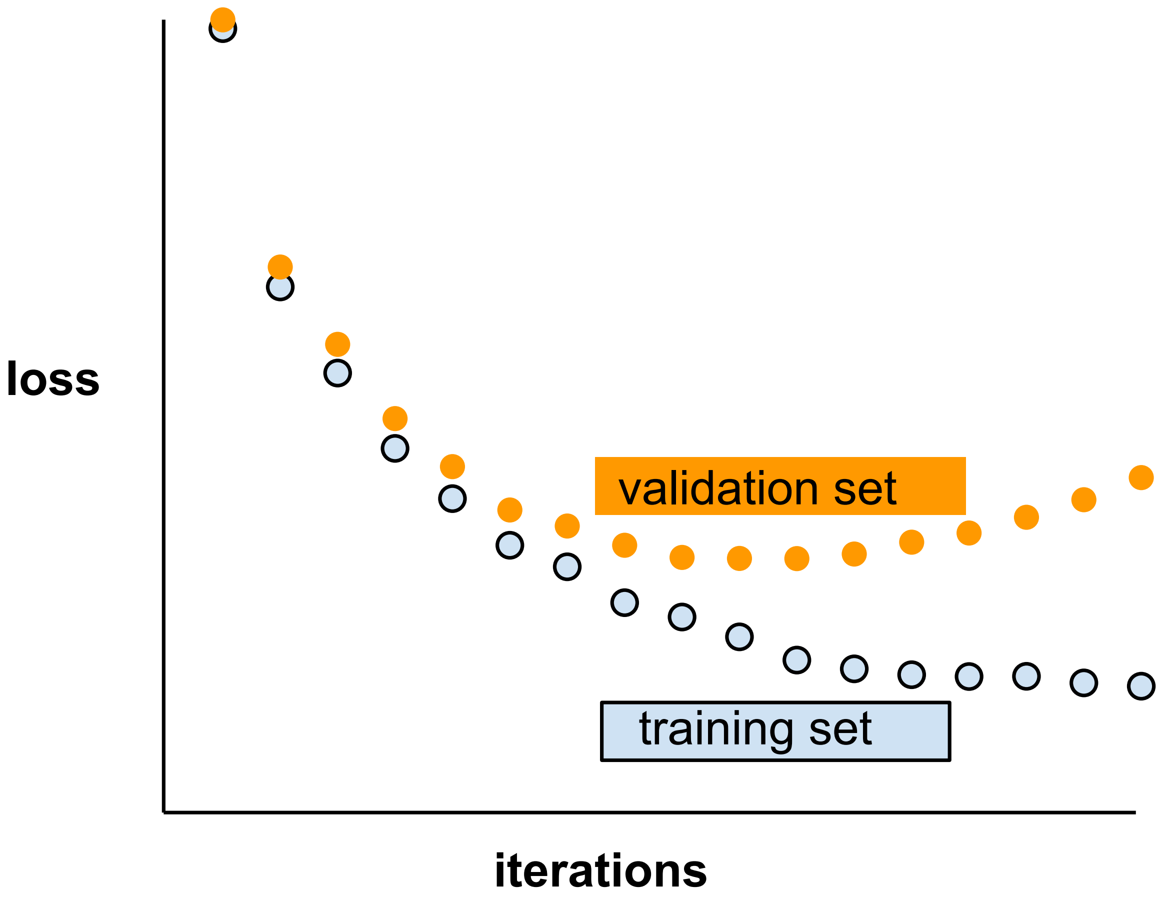
generalization

Refers to your model's ability to make correct predictions on new, previously unseen data as opposed to the data used to train the model.

generalization curve

A **loss curve** (#loss_curve) showing both the **training set** (#training_set) and the **validation set** (#validation_set). A generalization curve can help you detect possible **overfitting** (#overfitting).

For example, the following generalization curve suggests overfitting because loss for the validation set ultimately becomes significantly higher than for the training set.



generalized linear model

A generalization of **least squares regression** (#least_squares_regression) models, which are based on Gaussian noise (https://wikipedia.org/wiki/Gaussian_noise), to other types of models based on other types of noise, such as Poisson noise (https://wikipedia.org/wiki/Shot_noise) or categorical noise. Examples of generalized linear models include:

- **logistic regression** (#logistic_regression)
- multi-class regression
- least squares regression

The parameters of a generalized linear model can be found through **convex optimization** (#convex_optimization).

Generalized linear models exhibit the following properties:

- The average prediction of the optimal least squares regression model is equal to the average label on the training data.
- The average probability predicted by the optimal logistic regression model is equal to the average label on the training data.

The power of a generalized linear model is limited by its features. Unlike a deep model, a generalized linear model cannot "learn new features."

## generative adversarial network (GAN)

A system to create new data in which a **generator** (#generator) creates data and a **discriminator** (#discriminator) determines whether that created data is valid or invalid.

generative model

Practically speaking, a model that does either of the following:

- Creates (generates) new examples from the training dataset. For example, a generative model could create poetry after training on a dataset of poems. The **generator** (#generator) part of a **generative adversarial network** (#generative_adversarial_network) falls into this category.
- Determines the probability that a new example comes from the training set, or was created from the same mechanism that created the training set. For example, after training on a dataset consisting of English sentences, a generative model could determine the probability that new input is a valid English sentence.

A generative model can theoretically discern the distribution of examples or particular features in a dataset. That is:

```
p(examples)
```

Unsupervised learning models are generative.

Contrast with **discriminative models** (#discriminative_model).

generator

The subsystem within a **generative adversarial network** (#generative_adversarial_network) that creates new **examples** (#example).

Contrast with **discriminative model** (#discriminative_model).

gradient

The vector of **partial derivatives** (#partial_derivative) with respect to all of the independent variables. In machine learning, the gradient is the vector of partial derivatives of the model function. The gradient points in the direction of steepest ascent.

gradient clipping

A commonly used mechanism to mitigate the **exploding gradient problem** (#exploding_gradient_problem) by artificially limiting (clipping) the maximum value of gradients when using **gradient descent** (#gradient_descent) to train a model.

gradient descent

A technique to minimize **loss** (#loss) by computing the gradients of loss with respect to the model's parameters, conditioned on training data. Informally, gradient descent iteratively adjusts parameters, gradually finding the best combination of **weights** (#weight) and bias to minimize loss.

## graph

In TensorFlow, a computation specification. Nodes in the graph represent operations. Edges are directed and represent passing the result of an operation (a **Tensor** (#tensor)) as an operand to another operation. Use **TensorBoard** (#TensorBoard) to visualize a graph.

graph execution

A TensorFlow programming environment in which the program first constructs a **graph** (#graph) and then executes all or part of that graph. Graph execution is the default execution mode in TensorFlow 1.x.

Contrast with **eager execution** (#eager_execution).

greedy policy RL

In reinforcement learning, a **policy** (#policy) that always chooses the action with the highest expected **return** (#return).

ground truth

The correct answer. Reality. Since reality is often subjective, expert **raters** (#rater) typically are the proxy for ground truth.

group attribution bias

Assuming that what is true for an individual is also true for everyone in that group. The effects of group attribution bias can be exacerbated if a **convenience sampling** (#convenience_sampling) is used for data collection. In a non-representative sample, attributions may be made that do not reflect reality.

See also **out-group homogeneity bias** (#out-group_homogeneity_bias) and **in-group bias** (#in-group_bias).

H

hashing

In machine learning, a mechanism for bucketing **categorical data** (#categorical_data), particularly when the number of categories is large, but the number of categories actually appearing in the dataset is comparatively small.

For example, Earth is home to about 60,000 tree species. You could represent each of the 60,000 tree species in 60,000 separate categorical buckets. Alternatively, if only 200 of those tree species actually appear in a dataset, you could use hashing to divide tree species into perhaps 500 buckets.

A single bucket could contain multiple tree species. For example, hashing could place *baobab* and *red maple*—two genetically dissimilar species—into the same bucket.

Regardless, hashing is still a good way to map large categorical sets into the desired number of buckets. Hashing turns a categorical feature having a large number of possible values into a much smaller number of values by grouping values in a deterministic way.

For more information on hashing, see the Feature Columns chapter (https://www.tensorflow.org/guide/feature_columns) in the TensorFlow Programmers Guide.

heuristic

A quick solution to a problem, which may or may not be the best solution. For example, "With a heuristic, we achieved 86% accuracy. When we switched to a deep neural network, accuracy went up to 98%."

hidden layer

A synthetic layer in a **neural network** (#neural_network) between the **input layer** (#input_layer) (that is, the features) and the **output layer** (#output_layer) (the prediction). Hidden layers typically contain an **activation function** (#activation_function) (such as **ReLU** (#ReLU)) for training. A **deep neural network** (#deep_neural_network) contains more than one hidden layer.

hierarchical clustering

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

- **Agglomerative clustering** first assigns every example to its own cluster, and iteratively merges the closest clusters to create a hierarchical tree.

- **Divisive clustering** first groups all examples into one cluster and then iteratively divides the cluster into a hierarchical tree.

Contrast with **centroid-based clustering** (#centroid_based_clustering).

hinge loss

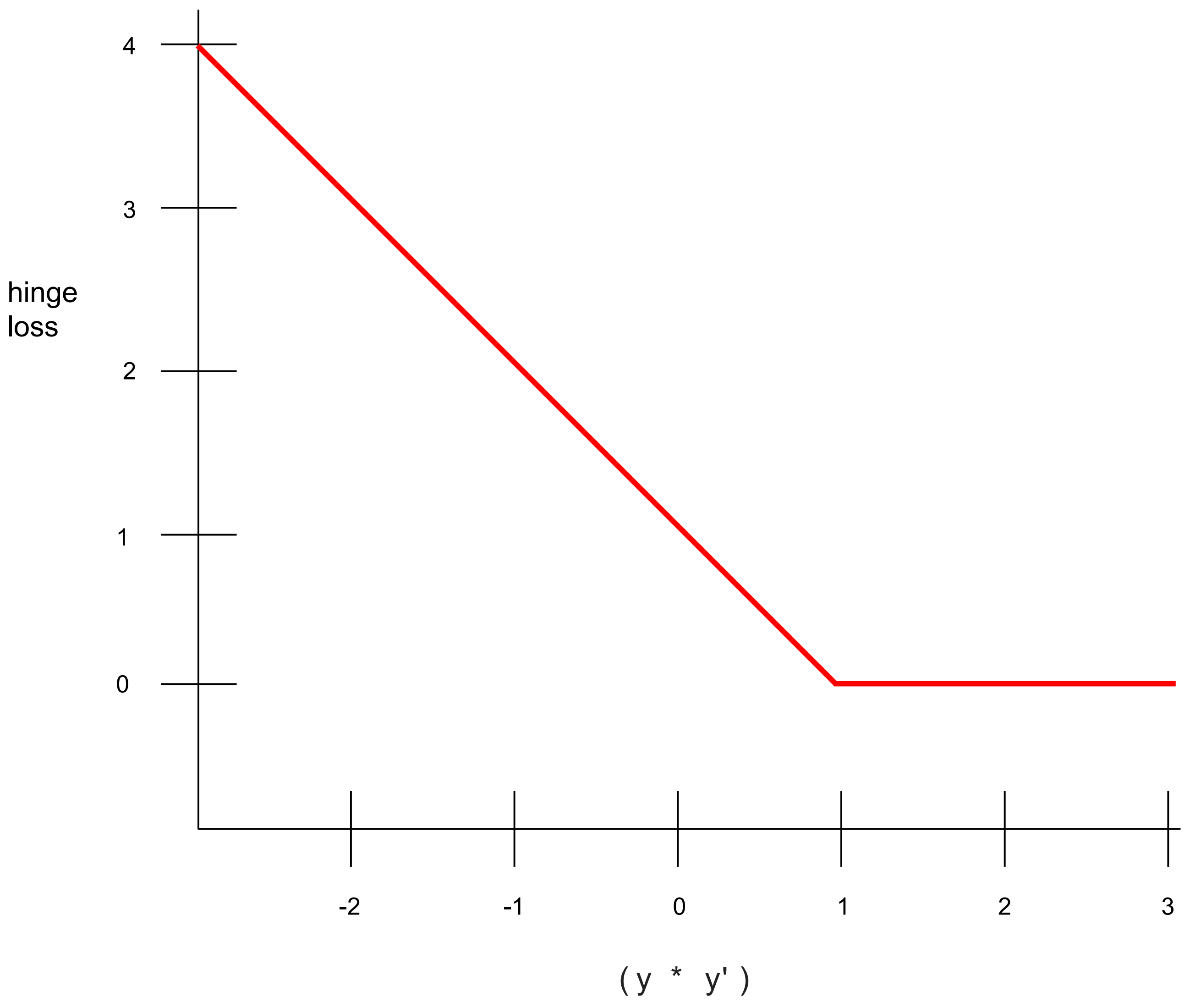
A family of **loss** (#loss) functions for **classification** (#classification_model) designed to find the **decision boundary** (#decision_boundary) as distant as possible from each training example, thus maximizing the margin between examples and the boundary. **KSVMs** (#KSVMs) use hinge loss (or a related function, such as squared hinge loss). For binary classification, the hinge loss function is defined as follows:

$$\text{loss} = \max(0, 1 - (y * y'))$$

where *y* is the true label, either -1 or +1, and *y'* is the raw output of the classifier model:

$$y' = b + w_1x_1 + w_2x_2 + \ldots w_nx_n$$

Consequently, a plot of hinge loss vs. (y * y') looks as follows:



holdout data

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

hyperparameter

The "knobs" that you tweak during successive runs of training a model. For example, **learning rate** (#learning_rate) is a hyperparameter.

Contrast with **parameter** (#parameter).

hyperplane

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. More typically in machine learning, a hyperplane is the boundary separating a high-dimensional space. **Kernel Support Vector Machines** (#KSVMs) use hyperplanes to separate positive classes from negative classes, often in a very high-dimensional space.

I

i.i.d.

Abbreviation for **independently and identically distributed** (#iid).

image recognition

A process that classifies object(s), pattern(s), or concept(s) in an image. Image recognition is also known as **image classification**.

For more information, see ML Practicum: Image Classification (/machine-learning/practica/image-classification).

imbalanced dataset

Synonym for **class-imbalanced dataset** (#class_imbalanced_data_set).

implicit bias

Automatically making an association or assumption based on one's mental models and memories. Implicit bias can affect the following:

- How data is collected and classified.
- How machine learning systems are designed and developed.

For example, when building a classifier to identify wedding photos, an engineer may use the presence of a white dress in a photo as a feature. However, white dresses have been customary only during certain eras and in certain cultures.

See also **confirmation bias** (#confirmation_bias).

incompatibility of fairness metrics

The idea that some notions of fairness are mutually incompatible and cannot be satisfied simultaneously. As a result, there is no single universal **metric** (#fairness_metric) for quantifying fairness that can be applied to all ML problems.

While this may seem discouraging, incompatibility of fairness metrics doesn't imply that fairness efforts are fruitless. Instead, it suggests that fairness must be defined contextually for a given ML problem, with the goal of preventing harms specific to its use cases.

See "On the (im)possibility of fairness" (https://arxiv.org/pdf/1609.07236.pdf) for a more detailed discussion of this topic.

independently and identically distributed (i.i.d)

Data drawn from a distribution that doesn't change, and where each value drawn doesn't depend on values that have been drawn previously. An i.i.d. is the ideal gas (https://wikipedia.org/wiki/Ideal_gas) of machine learning—a useful mathematical construct but almost never exactly found in the real world. For example, the distribution of visitors to a web page may be i.i.d. over a brief window of time; that is, the distribution doesn't change during that brief window and one person's visit is generally independent of another's visit. However, if you expand that window of time, seasonal differences in the web page's visitors may appear.

individual fairness

A fairness metric that checks whether similar individuals are classified similarly. For example, Brobdingnagian Academy might want to satisfy individual fairness by ensuring that two students with identical grades and standardized test scores are equally likely to gain admission.

Note that individual fairness relies entirely on how you define "similarity" (in this case, grades and test scores), and you can run the risk of introducing new fairness problems if your similarity metric misses important information (such as the rigor of a student's curriculum).

See "Fairness Through Awareness" (https://arxiv.org/pdf/1104.3913.pdf) for a more detailed discussion of individual fairness.

inference

In machine learning, often refers to the process of making predictions by applying the trained model to **unlabeled examples** (#unlabeled_example). In statistics, inference refers to the process of fitting the parameters of a distribution conditioned on some observed data. (See the Wikipedia article on statistical inference (https://wikipedia.org/wiki/Statistical_inference) .)

in-group bias

Showing partiality to one's own group or own characteristics. If testers or raters consist of the machine learning developer's friends, family, or colleagues, then in-group bias may invalidate product testing or the dataset.

In-group bias is a form of **group attribution bias** (#group_attribution_bias). See also **out-group homogeneity bias** (#out-group_homogeneity_bias).

input function

In TensorFlow, a function that returns input data to the training, evaluation, or prediction method of an **Estimator** (#Estimators). For example, the training input function returns a **batch** (#batch) of features and labels from the **training set** (#training_set).

input layer

The first layer (the one that receives the input data) in a **neural network** (#neural_network).

instance

Synonym for **example** (#example).

interpretability

The degree to which a model's predictions can be readily explained. Deep models are often non-interpretable; that is, a deep model's different layers can be hard to decipher. By contrast, linear regression models and **wide models** (#wide_model) are typically far more interpretable.

inter-rater agreement

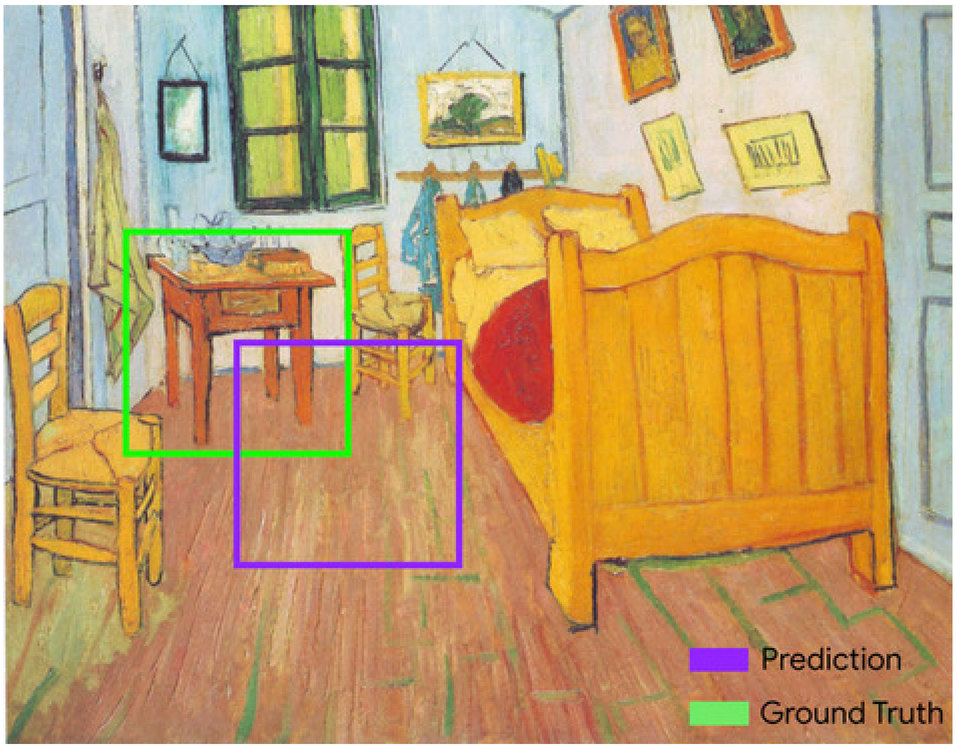
A measurement of how often human raters agree when doing a task. If raters disagree, the task instructions may need to be improved. Also sometimes called **inter-annotator agreement** or **inter-rater reliability**. See also Cohen's kappa (https://wikipedia.org/wiki/Cohen%27s_kappa), which is one of the most popular inter-rater agreement measurements.

intersection over union (IoU)

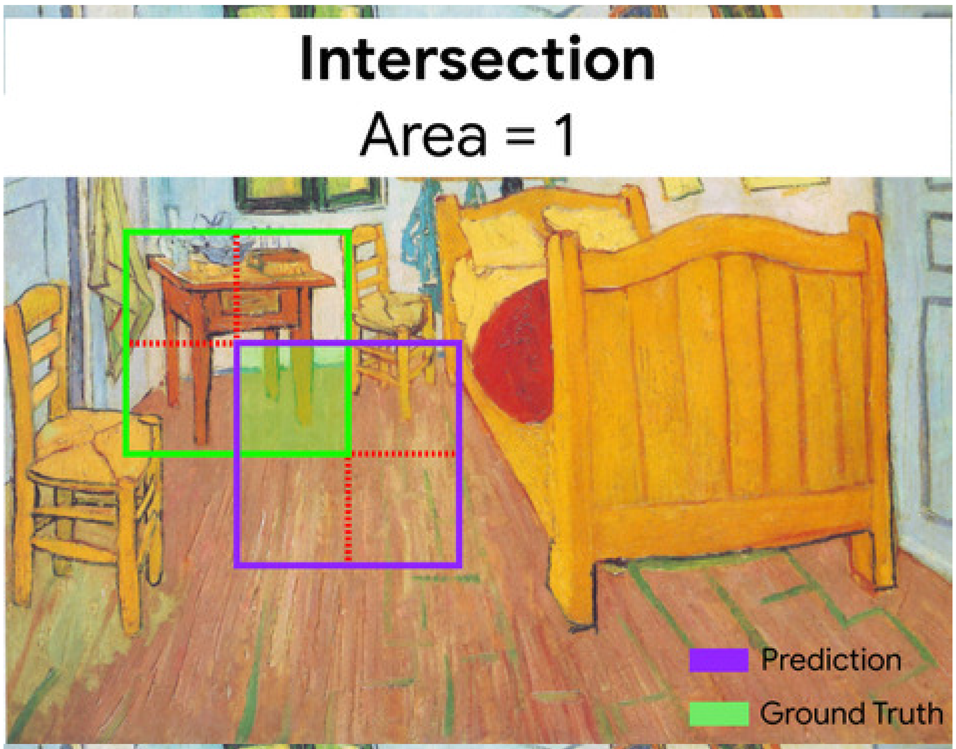
The intersection of two sets divided by their union. In machine-learning image-detection tasks, IoU is used to measure the accuracy of the model's predicted **bounding box** (#bounding_box) with respect to the **ground-truth** (#ground_truth) bounding box. In this case, the IoU for the two boxes is the ratio between the overlapping area and the total area, and its value ranges from 0 (no overlap of predicted bounding box and ground-truth bounding box) to 1 (predicted bounding box and ground-truth bounding box have the exact same coordinates).

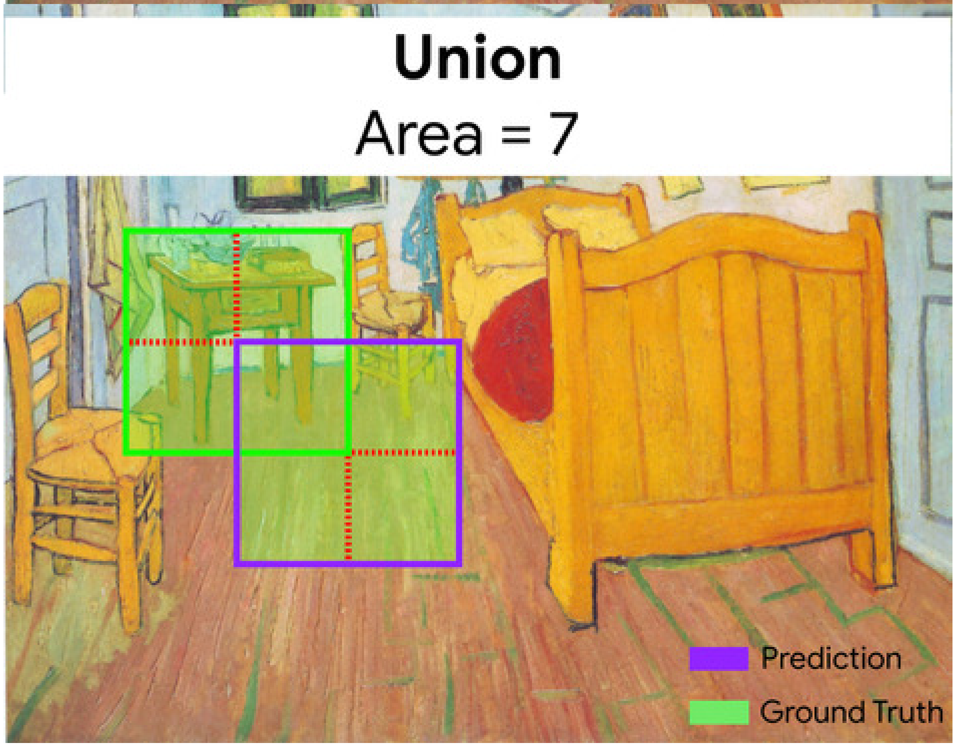
For example, in the image below:

- The predicted bounding box (the coordinates delimiting where the model predicts the night table in the painting is located) is outlined in purple.
- The ground-truth bounding box (the coordinates delimiting where the night table in the painting is actually located) is outlined in green.



Here, the intersection of the bounding boxes for prediction and ground truth (below left) is 1, and the union of the bounding boxes for prediction and ground truth (below right) is 7, so the IoU is $\frac{1}{7}$.





IoU

Abbreviation for **intersection over union** (#intersection_over_union).

item matrix

In **recommendation systems** (#recommendation_system), a matrix of **embeddings** (#embeddings) generated by **matrix factorization** (#matrix_factorization) that holds latent signals about each **item** (#items). Each row of the item matrix holds the value of a single latent feature for all items. For example, consider a movie recommendation system. Each column in the item matrix represents a single movie. The latent signals might represent genres, or might be harder-to-interpret signals that involve complex interactions among genre, stars, movie age, or other factors.

The item matrix has the same number of columns as the target matrix that is being factorized. For example, given a movie recommendation system that evaluates 10,000 movie titles, the item matrix will have 10,000 columns.

items

In a **recommendation system** (#recommendation_system), the entities that a system recommends. For example, videos are the items that a video store recommends, while books are the items that a bookstore recommends.

iteration

A single update of a model's weights during training. An iteration consists of computing the gradients of the parameters with respect to the loss on a single **batch** (#batch) of data.

K

Keras

A popular Python machine learning API. Keras (https://keras.io) runs on several deep learning frameworks, including TensorFlow, where it is made available as tf.keras (https://www.tensorflow.org/api_docs/python/tf/keras).

keypoints

The coordinates of particular features in an image. For example, for an **image recognition** (#image_recognition) model that distinguishes flower species, keypoints might be the center of each petal, the stem, the stamen, and so on.

Kernel Support Vector Machines (KSVMs)

A classification algorithm that seeks to maximize the margin between **positive** (#positive_class) and **negative classes** (#negative_class) by mapping input data vectors to a higher dimensional space. For example, consider a classification problem in which the input dataset has a hundred features. To maximize the margin between positive and negative classes, a KSVM could internally map those features into a million-dimension space. KSVMs uses a loss function called **hinge loss** (#hinge-loss).

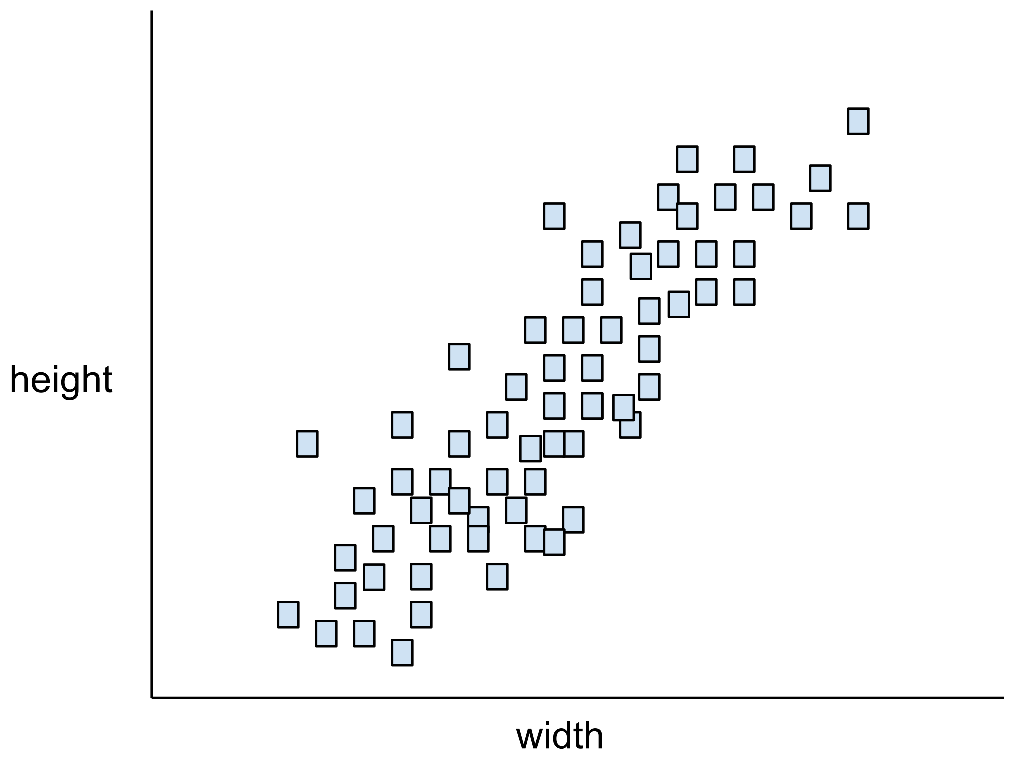
k-means

A popular **clustering** (#clustering) algorithm that groups examples in unsupervised learning. The k-means algorithm basically does the following:

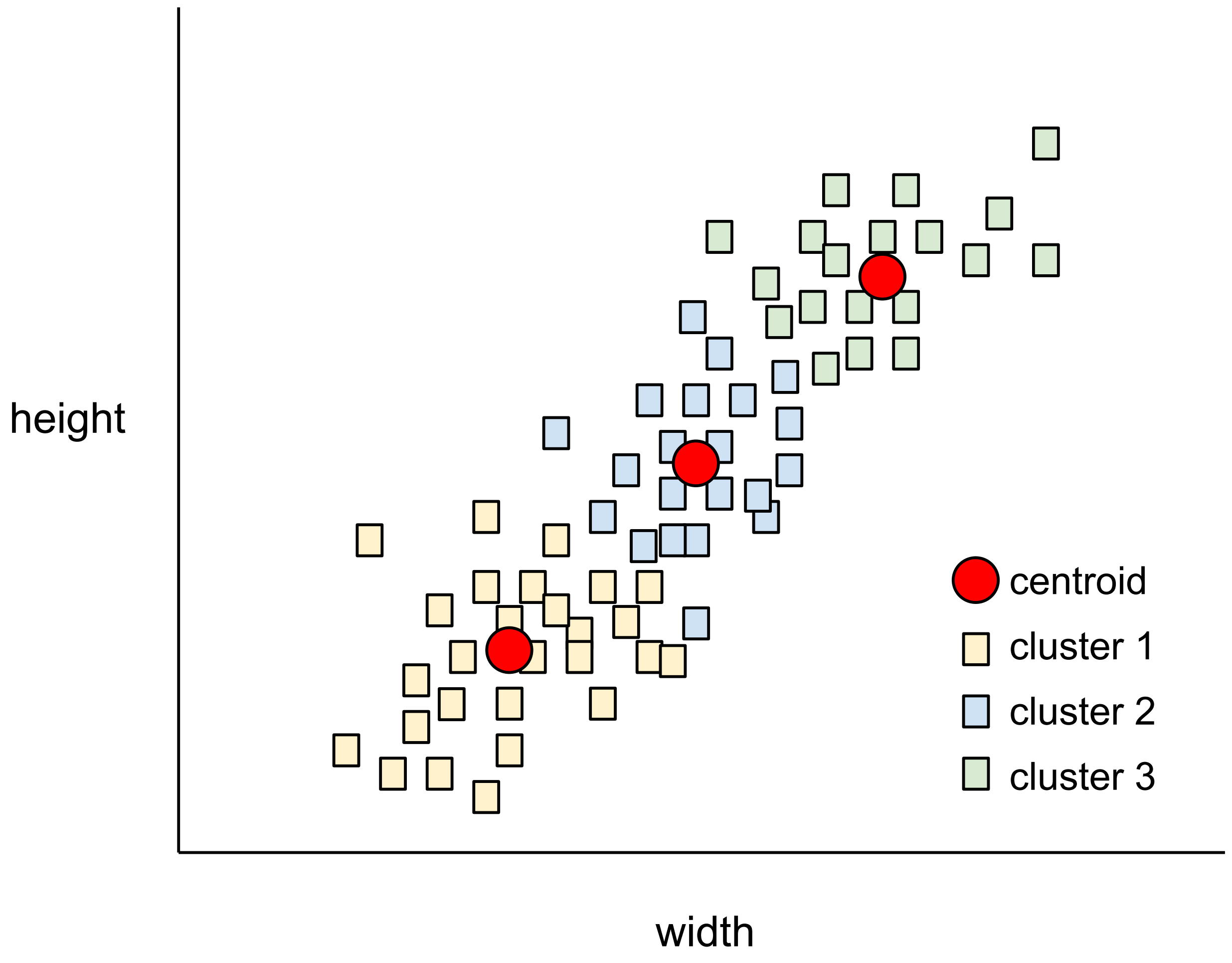
- Iteratively determines the best k center points (known as **centroids** (#centroid)).
- Assigns each example to the closest centroid. Those examples nearest the same centroid belong to the same group.

The k-means algorithm picks centroid locations to minimize the cumulative *square* of the distances from each example to its closest centroid.

For example, consider the following plot of dog height to dog width:



If k=3, the k-means algorithm will determine three centroids. Each example is assigned to its closest centroid, yielding three groups:



Imagine that a manufacturer wants to determine the ideal sizes for small, medium, and large sweaters for dogs. The three centroids identify the mean height and mean width of each dog in that cluster. So, the manufacturer should probably base sweater sizes on those three centroids. Note that the centroid of a cluster is typically *not* an example in the cluster.

The preceding illustrations shows k-means for examples with only two features (height and width). Note that k-means can group examples across many features.

k-median

A clustering algorithm closely related to **k-means** (#k-means). The practical difference between the two is as follows:

- In k-means, centroids are determined by minimizing the sum of the *squares* of the distance between a centroid candidate and each of its examples.

- In k-median, centroids are determined by minimizing the sum of the distance between a centroid candidate and each of its examples.

Note that the definitions of distance are also different:

- k-means relies on the Euclidean distance (https://wikipedia.org/wiki/Euclidean_distance) from the centroid to an example. (In two dimensions, the Euclidean distance means using the Pythagorean theorem to calculate the hypotenuse.) For example, the k-means distance between (2,2) and (5,-2) would be:

$$\text{Euclidean distance} = \sqrt{(2-5)^2 + (2--2)^2} = 5$$

- k-median relies on the Manhattan distance (https://wikipedia.org/wiki/Taxicab_geometry) from the centroid to an example. This distance is the sum of the absolute deltas in each dimension. For example, the k-median distance between (2,2) and (5,-2) would be:

$$\text{Manhattan distance} = |2-5| + |2--2| = 7$$

L

L_1 loss

Loss (#loss) function based on the absolute value of the difference between the values that a model is predicting and the actual values of the **labels** (#label). L_1 loss is less sensitive to outliers than **L_2 loss** (#squared_loss).

L_1 regularization

A type of **regularization** (#regularization) that penalizes weights in proportion to the sum of the absolute values of the weights. In models relying on **sparse features** (#sparse_features), L_1 regularization helps drive the weights of irrelevant or barely relevant features to exactly 0, which removes those features from the model. Contrast with **L_2 regularization** (#L2_regularization).

L_2 loss

See **squared loss** (#squared_loss).

L_2 regularization

A type of **regularization** (#regularization) that penalizes weights in proportion to the sum of the *squares* of the weights. L_2 regularization helps drive outlier weights (those with high positive or low negative values) closer to 0 but not quite to 0. (Contrast with **L1 regularization** (#L1_regularization).) L_2 regularization always improves generalization in linear models.

label

In supervised learning, the "answer" or "result" portion of an **example** (#example). Each example in a labeled dataset consists of one or more features and a label. For instance, in a housing dataset, the features might include the number of bedrooms, the number of bathrooms, and the age of the house, while the label might be the house's price. In a spam detection dataset, the features might include the subject line, the sender, and the email message itself, while the label would probably be either "spam" or "not spam."

labeled example

An example that contains **features** (#feature) and a **label** (#label). In supervised training, models learn from labeled examples.

lambda

Synonym for **regularization rate** (#regularization_rate).

(This is an overloaded term. Here we're focusing on the term's definition within **regularization** (#regularization).)

landmarks

Synonym for **keypoints** (#keypoints).

layer

A set of **neurons** (#neuron) in a **neural network** (#neural_network) that process a set of input features, or the output of those neurons.

Also, an abstraction in TensorFlow. Layers are Python functions that take **Tensors** (#tensor) and configuration options as input and produce other tensors as output. Once the necessary Tensors have been composed, the user can convert the result into an **Estimator** (#Estimators) via a **model function** (#model_function).

Layers API (tf.layers)

A TensorFlow API for constructing a **deep** (#deep_model) neural network as a composition of layers. The Layers API enables you to build different types of **layers** (#layer), such as:

- `tf.layers.Dense` for a **fully-connected layer** (#fully_connected_layer).
- `tf.layers.Conv2D` for a convolutional layer.

When writing a **custom Estimator** (#custom_estimator), you compose Layers objects to define the characteristics of all the **hidden layers** (#hidden_layer).

The Layers API follows the **Keras** (#Keras) layers API conventions. That is, aside from a different prefix, all functions in the Layers API have the same names and signatures as their counterparts in the Keras layers API.

learning rate

A scalar used to train a model via gradient descent. During each iteration, the **gradient descent** (#gradient_descent) algorithm multiplies the learning rate by the gradient. The resulting product is called the **gradient step**.

Learning rate is a key **hyperparameter** (#hyperparameter).

least squares regression

A linear regression model trained by minimizing **L_2 Loss** (#L2_loss).

linear model

A **model** (#model) that assigns one **weight** (#weight) per **feature** (#feature) to make **predictions** (#prediction). (Linear models also incorporate a **bias** (#bias).) By contrast, the relationship of weights to features in **deep models** (#deep_model) is not one-to-one.

A linear model uses the following formula:

$$y' = b + w_1x_1 + w_2x_2 + \ldots w_nx_n$$

where:

- y' is the raw prediction. (In certain kinds of linear models, this raw prediction will be further modified. For example, see logistic regression (#logistic_regression).)
- b is the **bias** (#bias).
- w is a **weight** (#weight), so w_1 is the weight of the first feature, w_2 is the weight of the second feature, and so on.
- x is a **feature** (#feature), so x_1 is the value of the first feature, x_2 is the value of the second feature, and so on.

For example, suppose a linear model for three features learns the following bias and weights:

- b = 7
- w_1 = -2.5
- w_2 = -1.2
- w_3 = 1.4

Therefore, given three features (x_1, x_2, and x_3), the linear model uses the following equation to generate each prediction:

$$y' = 7 + (-2.5)(x_1) + (-1.2)(x_2) + (1.4)(x_3)$$

Suppose a particular example contains the following values:

- x_1 = 4
- x_2 = -10
- x_3 = 5

Plugging those values into the formula yields a prediction for this example:

$$y' = 7 + (-2.5)(4) + (-1.2)(-10) + (1.4)(5)$$

$$y' = 16$$

Linear models tend to be easier to analyze and train than deep models. However, deep models can model complex relationships *between* features.

Linear regression (#linear_regression) and **logistic regression** (#logistic_regression) are two types of linear models. Linear models include not only models that use the linear equation but also a broader set of models that use the linear equation as part of the formula. For example, logistic regression post-processes the raw prediction (y') to calculate the prediction.

linear regression

Using the raw output (y') of a **linear model** (#linear_model) as the actual prediction in a **regression model** (#regression_model). The goal of a regression problem is to make a real-valued prediction. For example, if the raw output (y') of a linear model is 8.37, then the prediction is 8.37.

Contrast linear regression with **logistic regression** (#logistic_regression). Also, contrast regression with **classification** (#classification_model).

logistic regression

A **classification model** (#classification_model) that uses a **sigmoid function** (#sigmoid_function) to convert a **linear model's** (#linear_model) raw prediction (y') into a value between 0 and 1. You can interpret the value between 0 and 1 in either of the following two ways:

- As a probability that the example belongs to the **positive class** (#positive_class) in a binary classification problem.
- As a value to be compared against a **classification threshold** (#classification_threshold). If the value is equal to or above the classification threshold, the system classifies the example as the positive class. Conversely, if the value is below the given threshold, the system classifies the example as the **negative class** (#negative_class). For example, suppose the classification threshold is 0.82:

- Imagine an example that produces a raw prediction (y') of 2.6. The sigmoid of 2.6 is 0.93. Since 0.93 is greater than 0.82, the system classifies this example as the positive class.
- Imagine a different example that produces a raw prediction of 1.3. The sigmoid of 1.3 is 0.79. Since 0.79 is less than 0.82, the system classifies that example as the negative class.

Although logistic regression is often used in **binary classification** (#binary_classification) problems, logistic regression can also be used in **multi-class classification** (#multi-class) problems (where it becomes called **multi-class logistic regression** or **multinomial regression**).

logits

The vector of raw (non-normalized) predictions that a classification model generates, which is ordinarily then passed to a normalization function. If the model is solving a **multi-class classification** (#multi-class) problem, logits typically become an input to the **softmax** (#softmax) function. The softmax function then generates a vector of (normalized) probabilities with one value for each possible class.

In addition, logits sometimes refer to the element-wise inverse of the **sigmoid function** (#sigmoid_function). For more information, see tf.nn.sigmoid_cross_entropy_with_logits (https://www.tensorflow.org/api_docs/python/tf/nn/sigmoid_cross_entropy_with_logits).

Log Loss

The **loss** (#loss) function used in binary **logistic regression** (#logistic_regression).

log-odds

The logarithm of the odds of some event.

If the event refers to a binary probability, then **odds** refers to the ratio of the probability of success (p) to the probability of failure (1-p). For example, suppose that a given event has a 90% probability of success and a 10% probability of failure. In this case, odds is calculated as follows:

$$\text{odds} = \frac{\text{p}}{\text{(1-p)}} = \frac{.9}{.1} = 9$$

The log-odds is simply the logarithm of the odds. By convention, "logarithm" refers to natural logarithm, but logarithm could actually be any base greater than 1. Sticking to convention, the log-odds of our example is therefore:

$$\text{log-odds} = ln(9)\ = 2.2$$

The log-odds are the inverse of the **sigmoid function** (#sigmoid_function).

Long Short-Term Memory (LSTM)

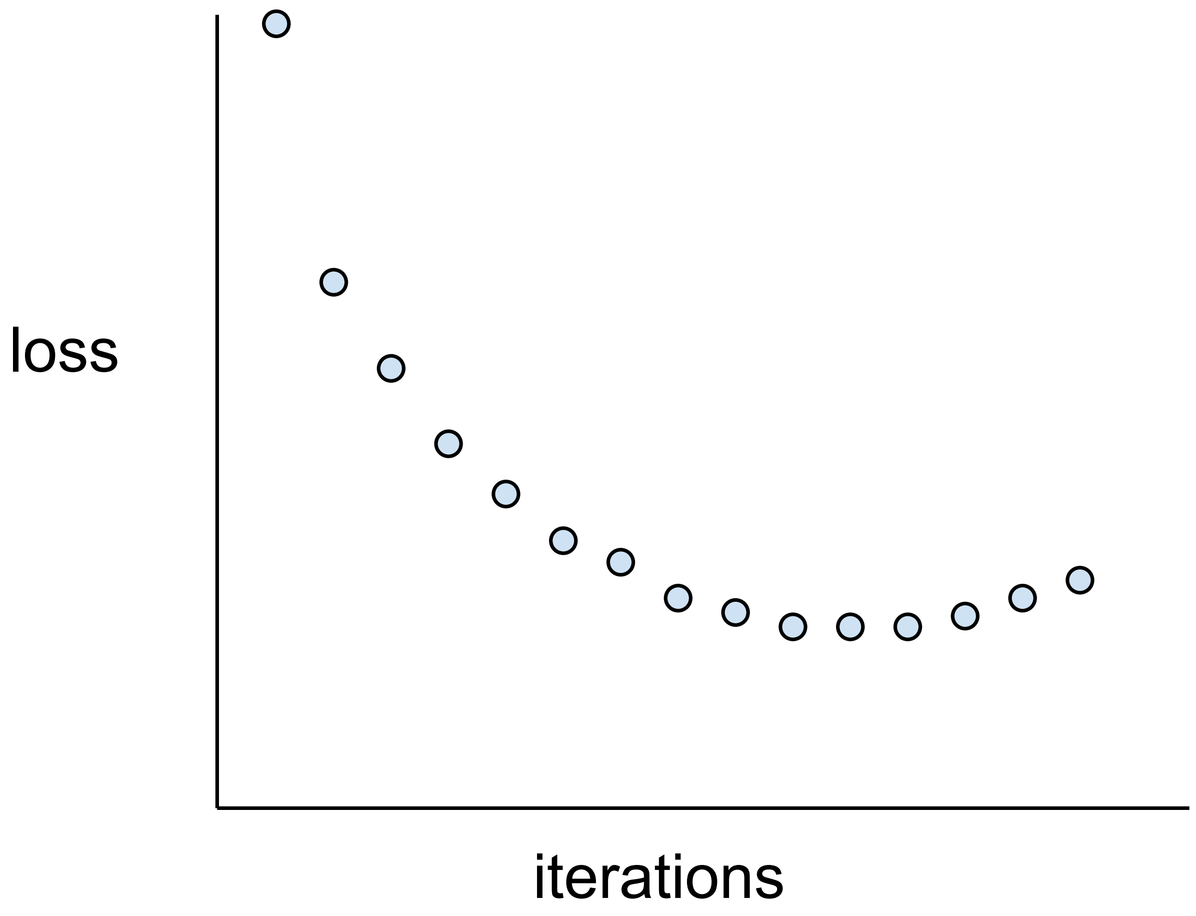
A type of cell in a **recurrent neural network** (#recurrent_neural_network) used to process sequences of data in applications such as handwriting recognition, machine translation, and image captioning. LSTMs address the **vanishing gradient problem** (#vanishing_gradient_problem) that occurs when training RNNs due to long data sequences by maintaining history in an internal memory state based on new input and context from previous cells in the RNN.

loss

A measure of how far a model's **predictions** (#prediction) are from its **label** (#label). Or, to phrase it more pessimistically, a measure of how bad the model is. To determine this value, a model must define a loss function. For example, linear regression models typically use **mean squared error** (#MSE) for a loss function, while logistic regression models use **Log Loss** (#Log_Loss).

loss curve

A graph of **loss** (#loss) as a function of training **iterations** (#iteration). For example:



The loss curve can help you determine when your model is **converging** (#convergence), **overfitting** (#overfitting), or **underfitting** (#underfitting).

loss surface

A graph of weight(s) vs. loss. **Gradient descent** (#gradient_descent) aims to find the weight(s) for which the loss surface is at a local minimum.

LSTM

Abbreviation for **Long Short-Term Memory** (#Long_Short-Term_Memory).

M

machine learning

A program or system that builds (trains) a predictive model from input data. The system uses the learned model to make useful predictions from new (never-before-seen) data drawn from the same distribution as the one used to train the model. Machine learning also refers to the field of study concerned with these programs or systems.

majority class

The more common label in a **class-imbalanced dataset** (#class_imbalanced_data_set). For example, given a dataset containing 99% non-spam labels and 1% spam labels, the non-spam labels are the majority class.

Markov decision process (MDP) RL

A graph representing the decision-making model where decisions (or **actions** (#action)) are taken to navigate a sequence of **states** (#state) under the assumption that the **Markov property** (#Markov_property) holds. In reinforcement learning, these transitions between states return a numerical **reward** (#reward).

Markov property

RL

A property of certain **environments** (#environment), where state transitions are entirely determined by information implicit in the current **state** (#state) and the agent's **action** (#action).

matplotlib

An open-source Python 2D plotting library. matplotlib (https://matplotlib.org/) helps you visualize different aspects of machine learning.

matrix factorization

In math, a mechanism for finding the matrices whose dot product approximates a target matrix.

In **recommendation systems** (#recommendation_system), the target matrix often holds users' ratings on **items** (#items). For example, the target matrix for a movie recommendation system might look something like the following, where the positive integers are user ratings and 0 means that the user didn't rate the movie:

	Casablanca	The Philadelphia Story	Black Panther	Wonder Woman	Pulp Fiction
User 1	5.0	3.0	0.0	2.0	0.0

User 2	4.0	0.0	0.0	1.0	5.0
User 3	3.0	1.0	4.0	5.0	0.0

The movie recommendation system aims to predict user ratings for unrated movies. For example, will User 1 like *Black Panther*?

One approach for recommendation systems is to use matrix factorization to generate the following two matrices:

- A **user matrix** (#user_matrix), shaped as the number of users X the number of embedding dimensions.
- An **item matrix** (#item_matrix), shaped as the number of embedding dimensions X the number of items.

For example, using matrix factorization on our three users and five items could yield the following user matrix and item matrix:

```
User Matrix                 Item Matrix

1.1   2.3           0.9   0.2   1.4    2.0   1.2
0.6   2.0           1.7   1.2   1.2   -0.1   2.1
2.5   0.5
```

The dot product of the user matrix and item matrix yields a recommendation matrix that contains not only the original user ratings but also predictions for the movies that each user hasn't seen. For example, consider User 1's rating of *Casablanca*, which was 5.0. The dot product corresponding to that cell in the recommendation matrix should hopefully be around 5.0, and it is:

```
(1.1 * 0.9) + (2.3 * 1.7) = 4.9
```

More importantly, will User 1 like *Black Panther*? Taking the dot product corresponding to the first row and the third column yields a predicted rating of 4.3:

```
(1.1 * 1.4) + (2.3 * 1.2) = 4.3
```

Matrix factorization typically yields a user matrix and item matrix that, together, are significantly more compact than the target matrix.

Mean Absolute Error (MAE)

An error metric calculated by taking an average of absolute errors. In the context of evaluating a model’s accuracy, MAE is the average absolute difference between the expected and predicted values across all training examples. Specifically, for n examples, for each value y and its prediction $\hat{y}$, MAE is defined as follows:

$$\text{MAE} = \frac{1}{n}\sum_{i=0}^{n}|y_i - \hat{y}_i|$$

Mean Squared Error (MSE)

The average squared loss per example. MSE is calculated by dividing the **squared loss** (#squared_loss) by the number of **examples** (#example). The values that **TensorFlow Playground** (#TensorFlow_Playground) displays for "Training loss" and "Test loss" are MSE.

metric

A number that you care about. May or may not be directly optimized in a machine-learning system. A metric that your system tries to optimize is called an **objective** (#objective).

Metrics API (tf.metrics)

A TensorFlow API for evaluating models. For example, `tf.metrics.accuracy` determines how often a model's predictions match labels. When writing a **custom Estimator** (#custom_estimator), you invoke Metrics API functions to specify how your model should be evaluated.

mini-batch

A small, randomly selected subset of the entire batch of **examples** (#example) run together in a single iteration of training or inference. The **batch size** (#batch_size) of a mini-batch is usually between 10 and 1,000. It is much more efficient to calculate the loss on a mini-batch than on the full training data.

mini-batch stochastic gradient descent (SGD)

A **gradient descent** (#gradient_descent) algorithm that uses **mini-batches** (#mini-batch). In other words, mini-batch SGD estimates the gradient based on a small subset of the training data. **Vanilla SGD** (#SGD) uses a mini-batch of size 1.

minimax loss

A loss function for **generative adversarial networks** (#generative_adversarial_network), based on the **cross-entropy** (#cross-entropy) between the distribution of generated data and real data.

Minimax loss is used in the first paper (https://arxiv.org/pdf/1406.2661.pdf) to describe generative adversarial networks.

minority class

The less common label in a **class-imbalanced dataset** (#class_imbalanced_data_set). For example, given a dataset containing 99% non-spam labels and 1% spam labels, the spam labels are the minority class.

ML

Abbreviation for **machine learning** (#machine_learning).

MNIST

A public-domain dataset compiled by LeCun, Cortes, and Burges containing 60,000 images, each image showing how a human manually wrote a particular digit from 0–9. Each image is stored as a 28x28 array of integers, where each integer is a grayscale value between 0 and 255, inclusive.

MNIST is a canonical dataset for machine learning, often used to test new machine learning approaches. For details, see The MNIST Database of Handwritten Digits (http://yann.lecun.com/exdb/mnist/).

model

The representation of what a machine learning system has learned from the training data. Within TensorFlow, model is an overloaded term, which can have either of the following two related meanings:

- The **TensorFlow** (#TensorFlow) graph that expresses the structure of how a prediction will be computed.

- The particular weights and biases of that TensorFlow graph, which are determined by **training** (#model_training).

model capacity

The complexity of problems that a model can learn. The more complex the problems that a model can learn, the higher the model's capacity. A model's capacity typically increases with the number of model parameters. For a formal definition of classifier capacity, see VC dimension (https://wikipedia.org/wiki/VC_dimension).

model function

The function within an **Estimator** (#Estimators) that implements machine learning training, evaluation, and inference. For example, the training portion of a model function might handle tasks such as defining the topology of a deep neural network and identifying its **optimizer** (#optimizer) function. When using **premade Estimators** (#premade_Estimator), someone has already written the model function for you. When using **custom Estimators** (#custom_estimator), you must write the model function yourself.

For details about writing a model function, see the Creating Custom Estimators chapter (https://www.tensorflow.org/guide/custom_estimators) in the TensorFlow Programmers Guide.

model training

The process of determining the best **model** (#model).

Momentum

A sophisticated gradient descent algorithm in which a learning step depends not only on the derivative in the current step, but also on the derivatives of the step(s) that immediately preceded it. Momentum involves computing an exponentially weighted moving average of the gradients over time, analogous to momentum in physics. Momentum sometimes prevents learning from getting stuck in local minima.

multi-class classification

Classification problems that distinguish among more than two classes. For example, there are approximately 128 species of maple trees, so a model that categorized maple tree species would be multi-class. Conversely, a model that divided emails into only two categories (*spam* and *not spam*) would be a **binary classification model** (#binary_classification).

multi-class logistic regression

Using **logistic regression** (#logistic_regression) in **multi-class classification** (#multi-class) problems.

multinomial classification

Synonym for **multi-class classification** (#multi-class).

N

NaN trap

When one number in your model becomes a NaN (https://wikipedia.org/wiki/NaN) during training, which causes many or all other numbers in your model to eventually become a NaN.

NaN is an abbreviation for "Not a Number."

natural language understanding

Determining a user's intentions based on what the user typed or said. For example, a search engine uses natural language understanding to determine what the user is searching for based on what the user typed or said.

negative class

In **binary classification** (#binary_classification), one class is termed positive and the other is termed negative. The positive class is the thing we're looking for and the negative class is the other possibility. For example, the negative class in a medical test might be "not tumor." The negative class in an email classifier might be "not spam." See also **positive class** (#positive_class).

neural network

A model that, taking inspiration from the brain, is composed of layers (at least one of which is **hidden** (#hidden_layer)) consisting of simple connected units or **neurons** (#neuron) followed by nonlinearities.

neuron

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

N-gram

An ordered sequence of N words. For example, *truly madly* is a 2-gram. Because order is relevant, *madly truly* is a different 2-gram than *truly madly*.

N	Name(s) for this kind of N-gram	Examples
2	bigram or 2-gram	*to go, go to, eat lunch, eat dinner*
3	trigram or 3-gram	*ate too much, three blind mice, the bell tolls*
4	4-gram	*walk in the park, dust in the wind, the boy ate lentils*

Many **natural language understanding** (#natural_language_understanding) models rely on N-grams to predict the next word that the user will type or say. For example, suppose a user typed *three blind*. An NLU model based on trigrams would likely predict that the user will next type *mice*.

Contrast N-grams with **bag of words** (#bag_of_words), which are unordered sets of words.

NLU

Abbreviation for **natural language understanding** (#natural_language_understanding).

node (neural network)

A **neuron** (#neuron) in a **hidden layer** (#hidden_layer).

node (TensorFlow graph)

An operation in a TensorFlow **graph** (#graph).

noise

Broadly speaking, anything that obscures the signal in a dataset. Noise can be introduced into data in a variety of ways. For example:

- Human raters make mistakes in labeling.
- Humans and instruments mis-record or omit feature values.

non-response bias

See **selection bias** (#selection_bias).

normalization

The process of converting an actual range of values into a standard range of values, typically -1 to +1 or 0 to 1. For example, suppose the natural range of a certain feature is 800 to 6,000. Through subtraction and division, you can normalize those values into the range -1 to +1.

See also **scaling** (#scaling).

numerical data

Features (#feature) represented as integers or real-valued numbers. For example, in a real estate model, you would probably represent the size of a house (in square feet or square meters) as numerical data. Representing a feature as numerical data indicates that the feature's values have a *mathematical* relationship to each other and possibly to the label. For example, representing the size of a house as numerical data indicates that a 200 square-meter house is twice as large as a 100 square-meter house. Furthermore, the number of square meters in a house probably has some mathematical relationship to the price of the house.

Not all integer data should be represented as numerical data. For example, postal codes in some parts of the world are integers; however, integer postal codes should not be represented as numerical data in models. That's because a postal code of `20000` is not twice (or half) as potent as a postal code of 10000. Furthermore, although different postal codes *do* correlate to different real estate values, we can't assume that real estate values at postal code 20000 are twice as valuable as real estate values at postal code 10000. Postal codes should be represented as **categorical data** (#categorical_data) instead.

Numerical features are sometimes called **continuous features** (#continuous_feature).

NumPy

An open-source math library (http://www.numpy.org/) that provides efficient array operations in Python. **pandas** (#pandas) is built on NumPy.

O

objective

A metric that your algorithm is trying to optimize.

objective function

The mathematical formula or metric that a model aims to optimize. For example, the objective function for **linear regression** (#linear_regression) is usually **squared loss** (#squared_loss). Therefore, when training a linear regression model, the goal is to minimize squared loss.

In some cases, the goal is to maximize the objective function. For example, if the objective function is accuracy, the goal is to maximize accuracy.

See also **loss** (#loss).

offline inference

Generating a group of **predictions** (#prediction), storing those predictions, and then retrieving those predictions on demand. Contrast with **online inference** (#online_inference).

one-hot encoding

A sparse vector in which:

- One element is set to 1.
- All other elements are set to 0.

One-hot encoding is commonly used to represent strings or identifiers that have a finite set of possible values. For example, suppose a given botany dataset chronicles 15,000 different species, each denoted with a unique string identifier. As part of feature engineering, you'll probably encode those string identifiers as one-hot vectors in which the vector has a size of 15,000.

one-shot learning

A machine learning approach, often used for object classification, designed to learn effective classifiers from a single training example.

See also **few-shot learning** (#few-shot_learning).

one-vs.-all

Given a classification problem with N possible solutions, a one-vs.-all solution consists of N separate **binary classifiers** (#binary_classification)—one binary classifier for each possible outcome. For example, given a model that classifies examples as animal, vegetable, or mineral, a one-vs.-all solution would provide the following three separate binary classifiers:

- animal vs. not animal
- vegetable vs. not vegetable
- mineral vs. not mineral

online inference

Generating **predictions** (#prediction) on demand. Contrast with **offline inference** (#offline_inference).

Operation (op)

A node in the TensorFlow graph. In TensorFlow, any procedure that creates, manipulates, or destroys a **Tensor** (#tensor) is an operation. For example, a matrix multiply is an operation that takes two Tensors as input and generates one Tensor as output.

optimizer

A specific implementation of the **gradient descent** (#gradient_descent) algorithm. TensorFlow's base class for optimizers is tf.train.Optimizer (https://www.tensorflow.org/api_docs/python/tf/train/Optimizer). Popular optimizers include:

- AdaGrad (https://www.tensorflow.org/api_docs/python/tf/train/AdagradOptimizer), which stands for ADAptive GRADient descent.
- Adam (https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer), which stands for ADAptive with Momentum.

Different optimizers may leverage one or more of the following concepts to enhance the effectiveness of gradient descent on a given **training set** (#training_set):

- momentum (https://www.tensorflow.org/api_docs/python/tf/train/MomentumOptimizer) (Momentum)
- update frequency
- sparsity/regularization (Ftrl (https://www.tensorflow.org/api_docs/python/tf/train/FtrlOptimizer))
- more complex math (Proximal (https://www.tensorflow.org/api_docs/python/tf/train/ProximalGradientDescentOptimizer), and

others)

You might even imagine an NN-driven optimizer (https://arxiv.org/abs/1606.04474).

out-group homogeneity bias

The tendency to see out-group members as more alike than in-group members when comparing attitudes, values, personality traits, and other characteristics. **In-group** refers to people you interact with regularly; **out-group** refers to people you do not interact with regularly. If you create a dataset by asking people to provide attributes about out-groups, those attributes may be less nuanced and more stereotyped than attributes that participants list for people in their in-group.

For example, Lilliputians might describe the houses of other Lilliputians in great detail, citing small differences in architectural styles, windows, doors, and sizes. However, the same Lilliputians might simply declare that Brobdingnagians all live in identical houses.

Out-group homogeneity bias is a form of **group attribution bias** (#group_attribution_bias).

See also **in-group bias** (#in-group_bias).

outliers

Values distant from most other values. In machine learning, any of the following are outliers:

- **Weights** (#weight) with high absolute values.
- Predicted values relatively far away from the actual values.
- Input data whose values are more than roughly 3 standard deviations from the mean.

Outliers often cause problems in model training. **Clipping** (#clipping) is one way of managing outliers.

output layer

The "final" layer of a neural network. The layer containing the answer(s).

overfitting

Creating a model that matches the **training data** (#training_set) so closely that the model fails to make correct predictions on new data.

P

pandas

A column-oriented data analysis API. Many machine learning frameworks, including TensorFlow, support pandas data structures as input. See the pandas documentation (http://pandas.pydata.org/) for details.

parameter

A variable of a model that the machine learning system trains on its own. For example, **weights** (#weight) are parameters whose values the machine learning system gradually learns through successive training iterations. Contrast with **hyperparameter** (#hyperparameter).

Parameter Server (PS)

A job that keeps track of a model's **parameters** (#parameter) in a distributed setting.

See the TensorFlow Architecture chapter (https://www.tensorflow.org/guide/extend/architecture) in the TensorFlow Programmers Guide for details.

parameter update

The operation of adjusting a model's **parameters** (#parameter) during training, typically within a single iteration of **gradient descent** (#gradient_descent).

partial derivative

A derivative in which all but one of the variables is considered a constant. For example, the partial derivative of *f(x, y)* with respect to *x* is the derivative of *f* considered as a function of *x* alone (that is, keeping *y* constant). The partial derivative of *f* with respect to *x* focuses only on how *x* is changing and ignores all other variables in the equation.

participation bias

Synonym for non-response bias. See **selection bias** (#selection_bias).

partitioning strategy

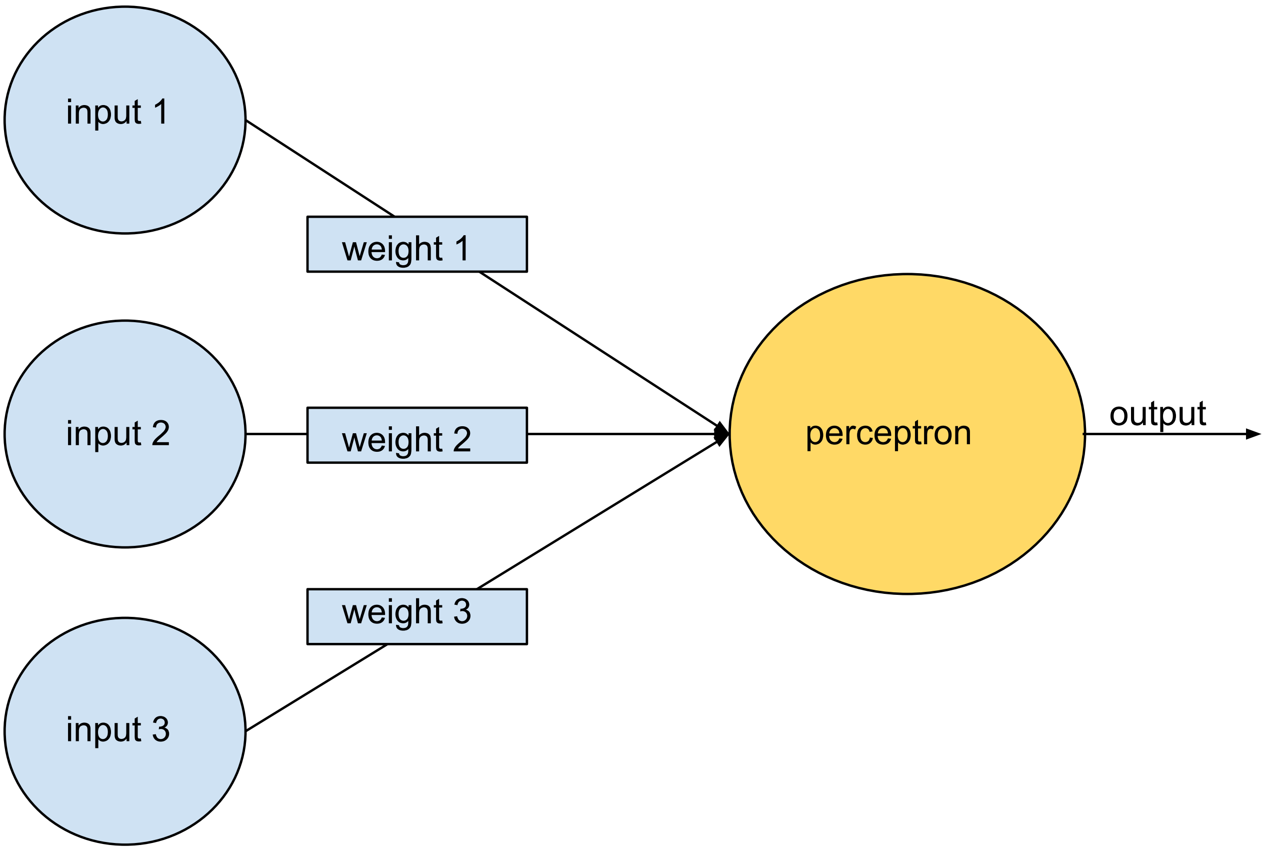
The algorithm by which variables are divided across **parameter servers** (#Parameter_Server).

perceptron

A system (either hardware or software) that takes in one or more input values, runs a function on the weighted sum of the inputs, and computes a single output value. In machine learning, the function is typically nonlinear, such as **ReLU** (#ReLU), **sigmoid** (#sigmoid_function), or tanh. For example, the following perceptron relies on the sigmoid function to process three input values:

$$f(x_1, x_2, x_3) = \text{sigmoid}(w_1 x_1 + w_2 x_2 + w_3 x_3)$$

In the following illustration, the perceptron takes three inputs, each of which is itself modified by a weight before entering the perceptron:



Perceptrons are the (**nodes** (#node)) in **deep neural networks** (#deep_model). That is, a deep neural network consists of multiple connected perceptrons, plus a **backpropagation** (#backpropagation) algorithm to introduce feedback.

performance

Overloaded term with the following meanings:

- The traditional meaning within software engineering. Namely: How fast (or efficiently) does this piece of software run?
- The meaning within machine learning. Here, performance answers the following question: How correct is this **model** (#model)? That is, how good are the model's predictions?

perplexity

One measure of how well a **model** (#model) is accomplishing its task. For example, suppose your task is to read the first few letters of a word a user is typing on a smartphone keyboard, and to offer a list of possible completion words. Perplexity, P, for this task is approximately the number of guesses you need to offer in order for your list to contain the actual word the user is trying to type.

Perplexity is related to **cross-entropy** (#cross-entropy) as follows:

$$P = 2^{-\text{cross entropy}}$$

pipeline

The infrastructure surrounding a machine learning algorithm. A pipeline includes gathering the data, putting the data into training data files, training one or more models, and exporting the models to production.

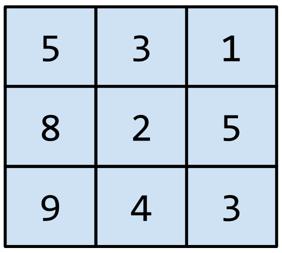
policy

RL

In reinforcement learning, an **agent's** (#agent) probabilistic mapping from **states** (#state) to **actions** (#action).

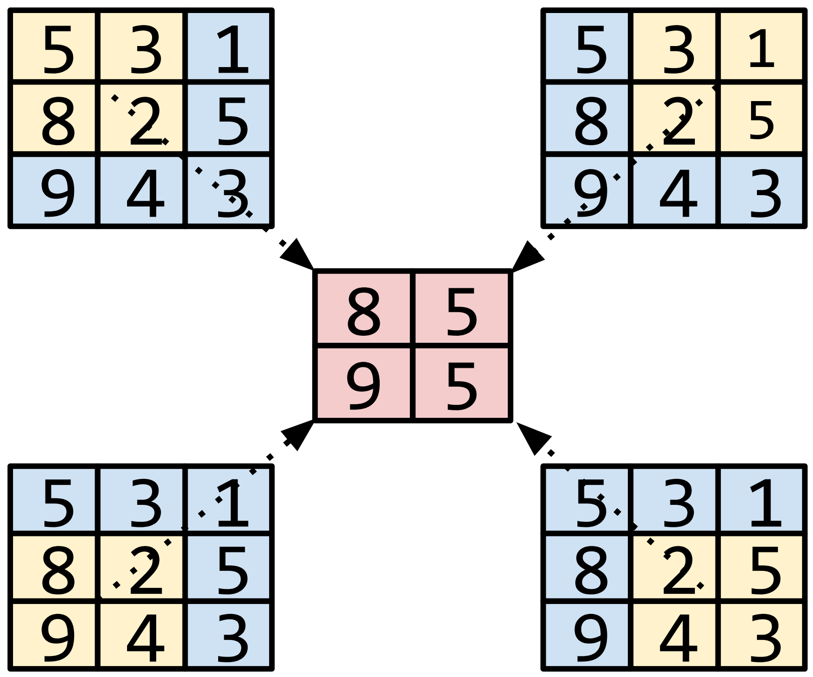
pooling

Reducing a matrix (or matrices) created by an earlier **convolutional layer** (#convolutional_layer) to a smaller matrix. Pooling usually involves taking either the maximum or average value across the pooled area. For example, suppose we have the following 3x3 matrix:



5	3	1
8	2	5
9	4	3

A pooling operation, just like a convolutional operation, divides that matrix into slices and then slides that convolutional operation by **strides** (#stride). For example, suppose the pooling operation divides the convolutional matrix into 2x2 slices with a 1x1 stride. As the following diagram illustrates, four pooling operations take place. Imagine that each pooling operation picks the maximum value of the four in that slice:



Pooling helps enforce **translational invariance** (#translational_invariance) in the input matrix.

Pooling for vision applications is known more formally as **spatial pooling**. Time-series applications usually refer to pooling as **temporal pooling**. Less formally, pooling is often called **subsampling** or **downsampling**.

positive class

In **binary classification** (#binary_classification), the two possible classes are labeled as positive and negative. The positive outcome is the thing we're testing for. (Admittedly, we're simultaneously testing for both outcomes, but play along.) For example, the positive class in a medical test might be "tumor." The positive class in an email classifier might be "spam."

Contrast with **negative class** (#negative_class).

post-processing

Processing the output of a model *after* the model has been run. Post-processing can be used to enforce fairness constraints without modifying models themselves.

For example, one might apply post-processing to a binary classifier by setting a classification threshold such that **equality of opportunity** (#equality_of_opportunity) is maintained for some attribute by checking that the **true positive rate** (#TP_rate) is the same for all values of that attribute.

PR AUC (area under the PR curve)

Area under the interpolated **precision-recall curve** (#precision-recall_curve), obtained by plotting (recall, precision) points for different values of the **classification threshold** (#classification_threshold). Depending on how it's calculated, PR AUC may be equivalent to the **average precision** (#average_precision) of the model.

precision

A metric for **classification models** (#classification_model). Precision identifies the frequency with which a model was correct when predicting the **positive class** (#positive_class). That is:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

precision-recall curve

A curve of **precision** (#precision) vs. **recall** (#recall) at different **classification thresholds** (#classification_threshold).

prediction

A model's output when provided with an input **example** (#example).

prediction bias

A value indicating how far apart the average of **predictions** (#prediction) is from the average of **labels** (#label) in the dataset.

Not to be confused with the **bias term** (#bias) in machine learning models or with **bias in ethics and fairness** (#bias_ethics).

predictive parity

A **fairness metric** (#fairness_metric) that checks whether, for a given classifier, the **precision** (#precision) rates are equivalent for subgroups under consideration.

For example, a model that predicts college acceptance would satisfy predictive parity for nationality if its precision rate is the same for Lilliputians and Brobdingnagians.

Predictive parity is sometime also called *predictive rate parity*.

See "Fairness Definitions Explained" (http://fairware.cs.umass.edu/papers/Verma.pdf) (section 3.2.1) for a more detailed discussion of predictive parity.

predictive rate parity

Another name for **predictive parity** (#predictive_parity).

premade Estimator

An **Estimator** (#Estimators) that someone has already built. TensorFlow provides several premade Estimators, including `DNNClassifier`, `DNNRegressor`, and `LinearClassifier`. To learn more about premade Estimators, see the Premade Estimators chapter (https://www.tensorflow.org/guide/premade_estimators) in the TensorFlow Programmers Guide.

Contrast with **custom estimators** (#custom_estimator).

preprocessing

Processing data before it's used to train a model. Preprocessing could be as simple as removing words from an English text corpus that don't occur in the English dictionary, or could be as complex as re-expressing data points in a way that eliminates as many attributes that are correlated with **sensitive attributes** (#sensitive_attribute) as possible. Preprocessing can help satisfy **fairness constraints** (#fairness_constraint).

pre-trained model

Models or model components (such as **embeddings** (#embeddings)) that have been already been trained. Sometimes, you'll feed pre-trained embeddings into a **neural network** (#neural_network). Other times, your model will train the embeddings itself rather than rely on the pre-trained embeddings.

prior belief

What you believe about the data before you begin training on it. For example, **L_2 regularization** (#L2_regularization) relies on a prior belief that **weights** (#weight) should be small and normally distributed around zero.

proxy (sensitive attributes)

An attribute used as a stand-in for a **sensitive attribute** (#sensitive_attribute). For example, an individual's postal code might be used as a proxy for their income, race, or ethnicity.

proxy labels

Data used to approximate labels not directly available in a dataset.

For example, suppose you want *is it raining?* to be a Boolean label for your dataset, but the dataset doesn't contain rain data. If photographs are available, you might establish pictures of people carrying umbrellas as a proxy label for *is it raining?* However, proxy labels may distort results. For example, in some places, it may be more common to carry umbrellas to protect against sun than the rain.

Q

Q-function

RL

In reinforcement learning, the function that predicts the expected **return** (#return) from taking an **action** (#action) in a **state** (#state) and then following a given **policy** (#policy).

Q-function is also known as **state-action value function**.

Q-learning RL

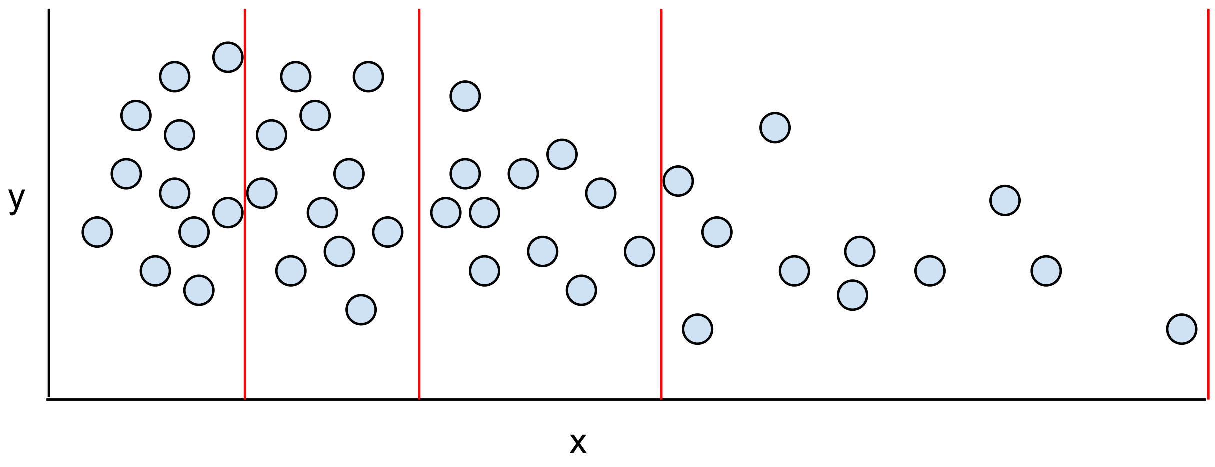
In reinforcement learning, an algorithm that allows an **agent** (#agent) to learn the optimal **Q-function** (#q-function) of a **Markov decision process** (#markov_decision_process) by applying the **Bellman equation** (#bellman_equation). The Markov decision process models an **environment** (#environment).

quantile

Each bucket in **quantile bucketing** (#quantile_bucketing).

quantile bucketing

Distributing a feature's values into **buckets** (#bucketing) so that each bucket contains the same (or almost the same) number of examples. For example, the following figure divides 44 points into 4 buckets, each of which contains 11 points. In order for each bucket in the figure to contain the same number of points, some buckets span a different width of x-values.



quantization

An algorithm that implements **quantile bucketing** (#quantile_bucketing) on a particular **feature** (#feature) in a **dataset** (#data_set).

queue

A TensorFlow **Operation** (#Operation) that implements a queue data structure. Typically used in I/O.

R

random forest

An ensemble approach to finding the **decision tree** (#decision_tree) that best fits the training data by creating many decision trees and then determining the "average" one. The "random" part of the term refers to building each of the decision trees from a random selection of features; the "forest" refers to the set of decision trees.

random policy RL

In reinforcement learning, a **policy** (#policy) that chooses an **action** (#action) at random.

rank (ordinality)

The ordinal position of a class in a machine learning problem that categorizes classes from highest to lowest. For example, a behavior ranking system could rank a dog's rewards from highest (a steak) to lowest (wilted kale).

rank (Tensor)

The number of dimensions in a **Tensor** (#tensor). For instance, a scalar has rank 0, a vector has rank 1, and a matrix has rank 2.

Not to be confused with **rank (ordinality)** (#rank_ordinality).

rater

A human who provides **labels** (#label) in **examples** (#example). Sometimes called an "annotator."

recall

A metric for **classification models** (#classification_model) that answers the following question: Out of all the possible positive labels, how many did the model correctly identify? That is:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

recommendation system

A system that selects for each user a relatively small set of desirable **items** (#items) from a large corpus. For example, a video recommendation system might recommend two videos from a corpus of 100,000 videos, selecting *Casablanca* and *The Philadelphia Story* for one user, and *Wonder Woman* and *Black Panther* for another. A video recommendation system might base its recommendations on factors such as:

- Movies that similar users have rated or watched.
- Genre, directors, actors, target demographic...

Rectified Linear Unit (ReLU)

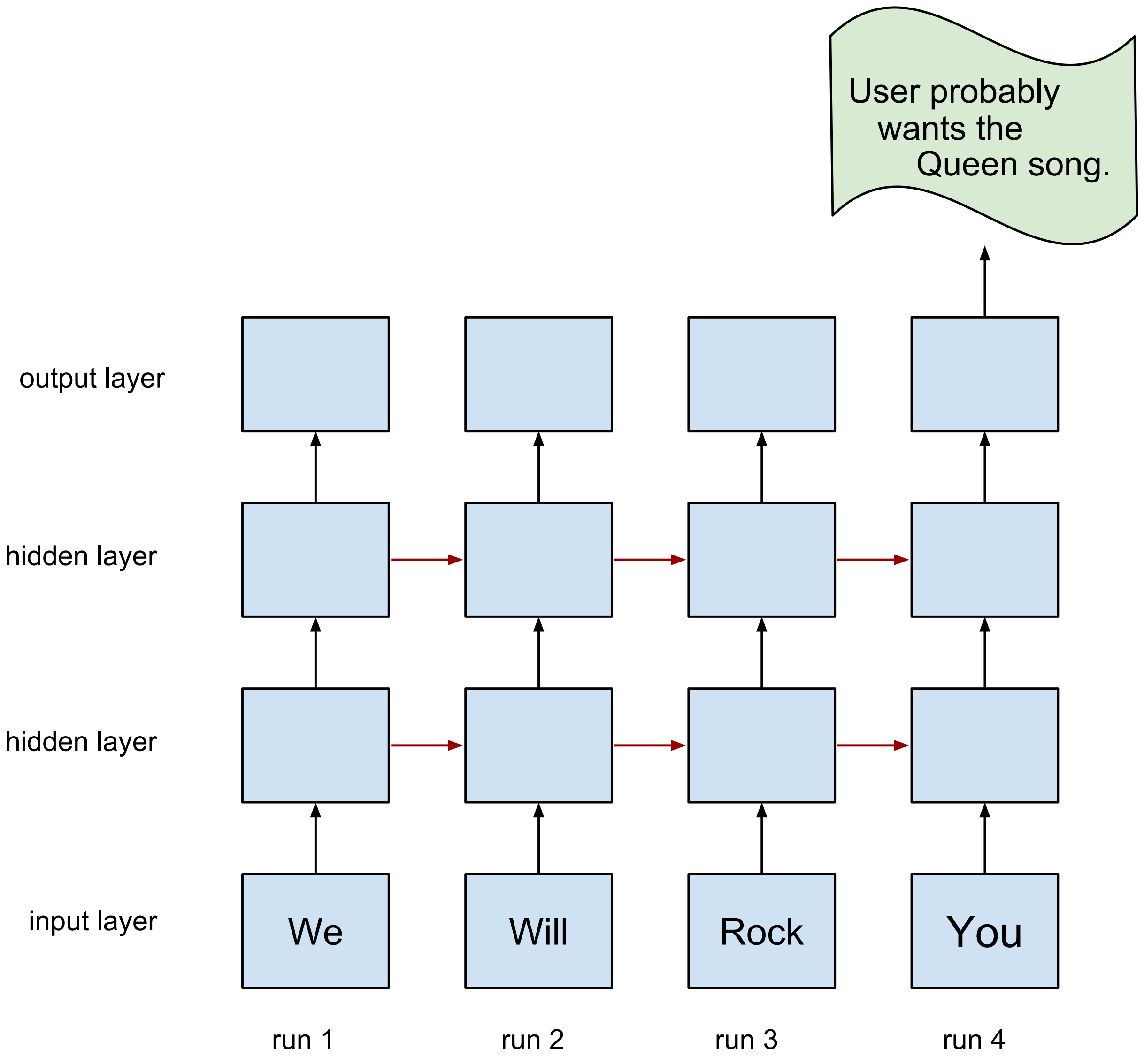
An **activation function** (#activation_function) with the following rules:

- If input is negative or zero, output is 0.
- If input is positive, output is equal to input.

recurrent neural network

A **neural network** (#neural_network) that is intentionally run multiple times, where parts of each run feed into the next run. Specifically, hidden layers from the previous run provide part of the input to the same hidden layer in the next run. Recurrent neural networks are particularly useful for evaluating sequences, so that the hidden layers can learn from previous runs of the neural network on earlier parts of the sequence.

For example, the following figure shows a recurrent neural network that runs four times. Notice that the values learned in the hidden layers from the first run become part of the input to the same hidden layers in the second run. Similarly, the values learned in the hidden layer on the second run become part of the input to the same hidden layer in the third run. In this way, the recurrent neural network gradually trains and predicts the meaning of the entire sequence rather than just the meaning of individual words.



regression model

A type of model that outputs continuous (typically, floating-point) values. Compare with **classification models** (#classification_model), which output discrete values, such as "day lily" or "tiger lily."

regularization

The penalty on a model's complexity. Regularization helps prevent **overfitting** (#overfitting). Different kinds of regularization include:

- **L_1 regularization** (#L1_regularization)
- **L_2 regularization** (#L2_regularization)
- **dropout regularization** (#dropout_regularization)
- **early stopping** (#early_stopping) (this is not a formal regularization method, but can effectively limit overfitting)

regularization rate

A scalar value, represented as lambda, specifying the relative importance of the regularization function. The following simplified **loss** (#loss) equation shows the regularization rate's influence:

$$\text{minimize}(\text{loss function} + \lambda(\text{regularization function}))$$

Raising the regularization rate reduces **overfitting** (#overfitting) but may make the model less **accurate** (#accuracy).

reinforcement learning (RL)

RL

A family of algorithms that learn an optimal **policy** (#policy), whose goal is to maximize **return** (#return) when interacting with an **environment** (#environment). For example, the ultimate reward of most games is victory. Reinforcement learning systems can become expert at playing complex games by evaluating sequences of previous game moves that ultimately led to wins and sequences that ultimately led to losses.

replay buffer

RL

In **DQN** (#deep_q-network)-like algorithms, the memory used by the agent to store state transitions for use in **experience replay** (#experience_replay).

reporting bias

The fact that the frequency with which people write about actions, outcomes, or properties is not a reflection of their real-world frequencies or the degree to which a property is characteristic of a class of individuals. Reporting bias can influence the composition of data that machine learning systems learn from.

For example, in books, the word *laughed* is more prevalent than *breathed*. A machine learning model that estimates the relative frequency of laughing and breathing from a book corpus would probably determine that laughing is more common than breathing.

representation

The process of mapping data to useful **features** (#feature).

re-ranking

The final stage of a **recommendation system** (#recommendation_system), during which scored items may be re-graded according to some other (typically, non-ML) algorithm. Re-ranking evaluates the list of items generated by the **scoring** (#scoring) phase, taking actions such as:

- Eliminating items that the user has already purchased.
- Boosting the score of fresher items.

return RL

In reinforcement learning, given a certain policy and a certain state, the return is the sum of all **rewards** (#reward) that the **agent** (#agent) expects to receive when following the **policy** (#policy) from the **state** (#state) to the end of the **episode** (#episode). The agent accounts for the delayed nature of expected rewards by discounting rewards according to the state transitions required to obtain the reward.

Therefore, if the discount factor is γ, and $r_0, \ldots, r_N$ denote the rewards until the end of the episode, then the return calculation is as follows:

$$\text{Return} = r_0 + \gamma r_1 + \gamma^2 r_2 + \ldots + \gamma^{N-1} r_{N-1}$$

reward RL

In reinforcement learning, the numerical result of taking an **action** (#action) in a **state** (#state), as defined by the **environment** (#environment).

ridge regularization

Synonym for **L$_2$ regularization** (#L2_regularization). The term **ridge regularization** is more frequently used in pure statistics contexts, whereas **L$_2$ regularization** is used more often in machine learning.

RNN

Abbreviation for **recurrent neural networks** (#recurrent_neural_network).

ROC (receiver operating characteristic) Curve

A curve of **true positive rate** (#TP_rate) vs. **false positive rate** (#FP_rate) at different **classification thresholds** (#classification_threshold). See also **AUC** (#AUC).

root directory

The directory you specify for hosting subdirectories of the TensorFlow checkpoint and events files of multiple models.

Root Mean Squared Error (RMSE)

The square root of the **Mean Squared Error** (#MSE).

rotational invariance

In an image classification problem, an algorithm's ability to successfully classify images even when the orientation of the image changes. For example, the algorithm can still identify a tennis racket whether it is pointing up, sideways, or down. Note that rotational invariance is not always desirable; for example, an upside-down 9 should not be classified as a 9.

See also **translational invariance** (#translational_invariance) and **size invariance** (#size_invariance).

S

sampling bias

See **selection bias** (#selection_bias).

SavedModel

The recommended format for saving and recovering TensorFlow models. SavedModel is a language-neutral, recoverable serialization format, which enables higher-level systems and tools to produce, consume, and transform TensorFlow models.

See the Saving and Restoring chapter (https://www.tensorflow.org/guide/saved_model) in the TensorFlow Programmer's Guide for complete details.

Saver

A TensorFlow object (https://www.tensorflow.org/api_docs/python/tf/train/Saver) responsible for saving model checkpoints.

scalar

A single number or a single string that can be represented as a **tensor** (#tensor) of **rank** (#rank) 0. For example, the following lines of code each create one scalar in TensorFlow:

```
breed = tf.Variable("poodle", tf.string)
temperature = tf.Variable(27, tf.int16)
precision = tf.Variable(0.982375101275, tf.float64)
```

scaling

A commonly used practice in **feature engineering** (#feature_engineering) to tame a feature's range of values to match the range of other features in the dataset. For example, suppose that you want all floating-point features in the dataset to have a range of 0 to 1. Given a particular feature's range of 0 to 500, you could scale that feature by dividing each value by 500.

See also **normalization** (#normalization).

scikit-learn

A popular open-source machine learning platform. See www.scikit-learn.org (http://www.scikit-learn.org/).

scoring

The part of a **recommendation system** (#recommendation_system) that provides a value or ranking for each item produced by the **candidate generation** (#candidate_generation) phase.

selection bias

Errors in conclusions drawn from sampled data due to a selection process that generates systematic differences between samples observed in the data and those not observed. The following forms of selection bias exist:

- **coverage bias**: The population represented in the dataset does not match the population that the machine learning model is making predictions about.
- **sampling bias**: Data is not collected randomly from the target group.
- **non-response bias** (also called **participation bias**): Users from certain groups opt-out of surveys at different rates than users from other groups.

For example, suppose you are creating a machine learning model that predicts people's enjoyment of a movie. To collect training data, you hand out a survey to everyone in the front row of a theater showing the movie. Offhand, this may sound like a reasonable way to gather a dataset; however, this form of data collection may introduce the following forms of selection bias:

- coverage bias: By sampling from a population who chose to see the movie, your model's predictions may not generalize to people who did not already express that level of interest in the movie.
- sampling bias: Rather than randomly sampling from the intended population (all the people at the movie), you sampled only the people in the front row. It is possible that the people sitting in the front row were more interested in the movie than those in other rows.
- non-response bias: In general, people with strong opinions tend to respond to optional surveys more frequently than people with mild opinions. Since the movie survey is optional, the responses are more likely to form a bimodal distribution (https://wikipedia.org/wiki/Multimodal_distribution) than a normal (bell-shaped) distribution.

semi-supervised learning

Training a model on data where some of the training examples have labels but others don’t. One technique for semi-supervised learning is to infer labels for the unlabeled examples, and then to train on the inferred labels to create a new model. Semi-supervised learning can be useful if labels are expensive to obtain but unlabeled examples are plentiful.

sensitive attribute

A human attribute that may be given special consideration for legal, ethical, social, or personal reasons.

sentiment analysis

Using statistical or machine learning algorithms to determine a group's overall attitude—positive or negative—toward a service, product, organization, or topic. For example, using **natural language understanding** (#natural_language_understanding), an algorithm could perform sentiment analysis on the textual feedback from a university course to determine the degree to which students generally liked or disliked the course.

sequence model

A model whose inputs have a sequential dependence. For example, predicting the next video watched from a sequence of previously watched videos.

serving

A synonym for **inferring** (#inference).

session (tf.session)

An object that encapsulates the state of the TensorFlow runtime and runs all or part of a **graph** (#graph). When using the low-level TensorFlow APIs, you instantiate and manage one or more `tf.session` objects directly. When using the Estimators API, Estimators instantiate session objects for you.

shape (Tensor)

The number of elements in each **dimension** (#dimensions) of a tensor. The shape is represented as a list of integers. For example, the following two-dimensional tensor has a shape of [3,4]:

```
[[5, 7, 6, 4],
 [2, 9, 4, 8],
 [3, 6, 5, 1]]
```

TensorFlow uses row-major (C-style) format to represent the order of dimensions, which is why the shape in TensorFlow is [3,4] rather than [4,3]. In other words, in a two-dimensional TensorFlow Tensor, the shape is [*number of rows*, *number of columns*].

sigmoid function

A function that maps logistic or multinomial regression output (log odds) to probabilities, returning a value between 0 and 1. The sigmoid function has the following formula:

$$y = \frac{1}{1 + e^{-\sigma}}$$

where σ in **logistic regression** (#logistic_regression) problems is simply:

$$\sigma = b + w_1x_1 + w_2x_2 + \ldots w_nx_n$$

In other words, the sigmoid function converts σ into a probability between 0 and 1.

In some **neural networks** (#neural_network), the sigmoid function acts as the **activation function** (#activation_function).

similarity measure

In **clustering** (#clustering) algorithms, the metric used to determine how alike (how similar) any two examples are.

size invariance

In an image classification problem, an algorithm's ability to successfully classify images even when the size of the image changes. For example, the algorithm can still identify a cat whether it consumes 2M pixels or 200K pixels. Note that even the best image classification algorithms still have practical limits on size invariance. For example, an algorithm (or human) is unlikely to correctly classify a cat image consuming only 20 pixels.

See also **translational invariance** (#translational_invariance) and **rotational invariance** (#rotational_invariance).

sketching

In **unsupervised machine learning** (#unsupervised_machine_learning), a category of algorithms that perform a preliminary similarity analysis on examples. Sketching algorithms use a locality-sensitive hash function (https://wikipedia.org/wiki/Locality-sensitive_hashing) to identify points that are likely to be similar, and then group them into buckets.

Sketching decreases the computation required for similarity calculations on large datasets. Instead of calculating similarity for every single pair of examples in the dataset, we calculate similarity only for each pair of points within each bucket.

softmax

A function that provides probabilities for each possible class in a **multi-class classification model** (#multi-class). The probabilities add up to exactly 1.0. For example, softmax might determine that the probability of a particular image being a dog at 0.9, a cat at 0.08, and a horse at 0.02. (Also called **full softmax**.)

Contrast with **candidate sampling** (#candidate_sampling).

sparse feature

Feature (#feature) vector whose values are predominately zero or empty. For example, a vector containing a single 1 value and a million 0 values is sparse. As another example, words in a search query could also be a sparse feature—there are many possible words in a given language, but only a few of them occur in a given query.

Contrast with **dense feature** (#dense_feature).

sparse representation

A **representation** (#representation) of a tensor that only stores nonzero elements.

For example, the English language consists of about a million words. Consider two ways to represent a count of the words used in one English sentence:

- A **dense representation** of this sentence must set an integer for all one million cells, placing a 0 in most of them, and a low integer into a few of them.

- A sparse representation of this sentence stores only those cells symbolizing a word actually in the sentence. So, if the sentence contained only 20 unique words, then the sparse representation for the sentence would store an integer in only 20 cells.

For example, consider two ways to represent the sentence, "Dogs wag tails." As the following tables show, the dense representation consumes about a million cells; the sparse representation consumes only 3 cells:

Dense Representation

Cell Number	Word	Occurrence
0	a	0
1	aardvark	0
2	aargh	0
3	aarti	0
… 140,391 more words with an occurrence of 0		
140395	dogs	1
… 633,062 words with an occurrence of 0		
773458	tails	1
… 189,135 words with an occurrence of 0		
962594	wag	1
… many more words with an occurrence of 0		

Sparse Representation

Cell Number	Word	Occurrence
140395	dogs	1
773458	tails	1
962594	wag	1

sparse vector

A vector whose values are mostly zeroes. See also **sparse feature** (#sparse_features).

sparsity

The number of elements set to zero (or null) in a vector or matrix divided by the total number of entries in that vector or matrix. For example, consider a 10x10 matrix in which 98 cells contain zero. The calculation of sparsity is as follows:

$$\text{sparsity} = \frac{98}{100} = 0.98$$

Feature sparsity refers to the sparsity of a feature vector; **model sparsity** refers to the sparsity of the model weights.

spatial pooling

See **pooling** (#pooling).

squared hinge loss

The square of the **hinge loss** (#hinge-loss). Squared hinge loss penalizes outliers more harshly than regular hinge loss.

squared loss

The **loss** (#loss) function used in **linear regression** (#linear_regression). (Also known as **L_2 Loss**.) This function calculates the squares of the difference between a model's predicted value for a labeled **example** (#example) and the actual value of the **label** (#label). Due to squaring, this loss function amplifies the influence of bad predictions. That is, squared loss reacts more strongly to outliers than **L_1 loss** (#L1_loss).

state RL

In reinforcement learning, the parameter values that describe the current configuration of the environment, which the **agent** (#agent) uses to choose an **action** (#action).

state-action value function RL

Synonym for **Q-function** (#q-function).

static model

A model that is trained offline.

stationarity

A property of data in a dataset, in which the data distribution stays constant across one or more dimensions. Most commonly, that dimension is time, meaning that data exhibiting stationarity doesn't change over time. For example, data that exhibits stationarity doesn't change from September to December.

step

A forward and backward evaluation of one **batch** (#batch).

step size

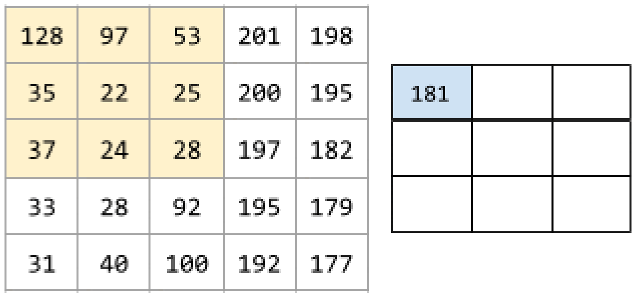
Synonym for **learning rate** (#learning_rate).

stochastic gradient descent (SGD)

A **gradient descent** (#gradient_descent) algorithm in which the batch size is one. In other words, SGD relies on a single example chosen uniformly at random from a dataset to calculate an estimate of the gradient at each step.

stride

In a convolutional operation or pooling, the delta in each dimension of the next series of input slices. For example, the following animation demonstrates a (1,1) stride during a convolutional operation. Therefore, the next input slice starts one position to the right of the previous input slice. When the operation reaches the right edge, the next slice is all the way over to the left but one position down.



128	97	53	201	198
35	22	25	200	195
37	24	28	197	182
33	28	92	195	179
31	40	100	192	177

181		

The preceding example demonstrates a two-dimensional stride. If the input matrix is three-dimensional, the stride would also be three-dimensional.

structural risk minimization (SRM)

An algorithm that balances two goals:

- The desire to build the most predictive model (for example, lowest loss).
- The desire to keep the model as simple as possible (for example, strong regularization).

For example, a function that minimizes loss+regularization on the training set is a structural risk minimization algorithm.

For more information, see http://www.svms.org/srm/ (http://www.svms.org/srm/).

Contrast with **empirical risk minimization** (#ERM).

subsampling

See **pooling** (#pooling).

summary

In TensorFlow, a value or set of values calculated at a particular **step** (#step), usually used for tracking model metrics during training.

supervised machine learning

Training a **model** (#model) from input data and its corresponding **labels** (#label). Supervised machine learning is analogous to a student learning a subject by studying a set of questions and their corresponding answers. After mastering the mapping between questions and answers, the student can then provide answers to new (never-before-seen) questions on the same topic. Compare with **unsupervised machine learning** (#unsupervised_machine_learning).

synthetic feature

A **feature** (#feature) not present among the input features, but created from one or more of them. Kinds of synthetic features include:

- **Bucketing** (#bucketing) a continuous feature into range bins.
- Multiplying (or dividing) one feature value by other feature value(s) or by itself.
- Creating a **feature cross** (#feature_cross).

Features created by **normalizing** (#normalization) or **scaling** (#scaling) alone are not considered synthetic features.

T

tabular Q-learning

RL

In reinforcement learning, implementing **Q-learning** (#q-learning) by using a table to store the **Q-functions** (#q-function) for every combination of **state** (#state) and **action** (#action).

target

Synonym for **label** (#label).

target network

RL

In **Deep Q-learning** (#q-learning), a neural network that is a stable approximation of the main neural network, where the main neural network implements either a **Q-function** (#q-function) or a **policy** (#policy). Then, you can train the main network on the Q-values predicted by the target network. Therefore, you prevent the feedback loop that occurs when the main network trains on Q-values predicted by itself. By avoiding this feedback, training stability increases.

temporal data

Data recorded at different points in time. For example, winter coat sales recorded for each day of the year would be temporal data.

Tensor

The primary data structure in TensorFlow programs. Tensors are N-dimensional (where N could be very large) data structures, most commonly scalars, vectors, or matrices. The elements of a Tensor can hold integer, floating-point, or string values.

TensorBoard

The dashboard that displays the summaries saved during the execution of one or more TensorFlow programs.

TensorFlow

A large-scale, distributed, machine learning platform. The term also refers to the base API layer in the TensorFlow stack, which supports general computation on dataflow graphs.

Although TensorFlow is primarily used for machine learning, you may also use TensorFlow for non-ML tasks that require numerical computation using dataflow graphs.

TensorFlow Playground

A program that visualizes how different **hyperparameters** (#hyperparameter) influence model (primarily neural network) training. Go to http://playground.tensorflow.org (http://playground.tensorflow.org) to experiment with TensorFlow Playground.

TensorFlow Serving

A platform to deploy trained models in production.

Tensor Processing Unit (TPU)

An application-specific integrated circuit (ASIC) that optimizes the performance of machine learning workloads. These ASICs are deployed as multiple **TPU chips** (#TPU_chip) on a **TPU device** (#TPU_device).

Tensor rank

See **rank (Tensor)** (#rank_Tensor).

Tensor shape

The number of elements a **Tensor** (#tensor) contains in various dimensions. For example, a [5, 10] Tensor has a shape of 5 in one dimension and 10 in another.

Tensor size

The total number of scalars a **Tensor** (#tensor) contains. For example, a [5, 10] Tensor has a size of 50.

termination condition

RL

In reinforcement learning, the conditions that determine when an **episode** (#episode) ends, such as when the agent reaches a certain state or exceeds a threshold number of state transitions. For example, in tic-tac-toe (https://wikipedia.org/wiki/Tic-tac-toe) (also known as noughts and crosses), an episode terminates either when a player marks three consecutive spaces or when all spaces are marked.

test set

The subset of the dataset that you use to test your **model** (#model) after the model has gone through initial vetting by the validation set.

Contrast with **training set** (#training_set) and **validation set** (#validation_set).

tf.Example

A standard protocol buffer (https://developers.google.com/protocol-buffers/) for describing input data for machine learning model training or inference.

tf.keras

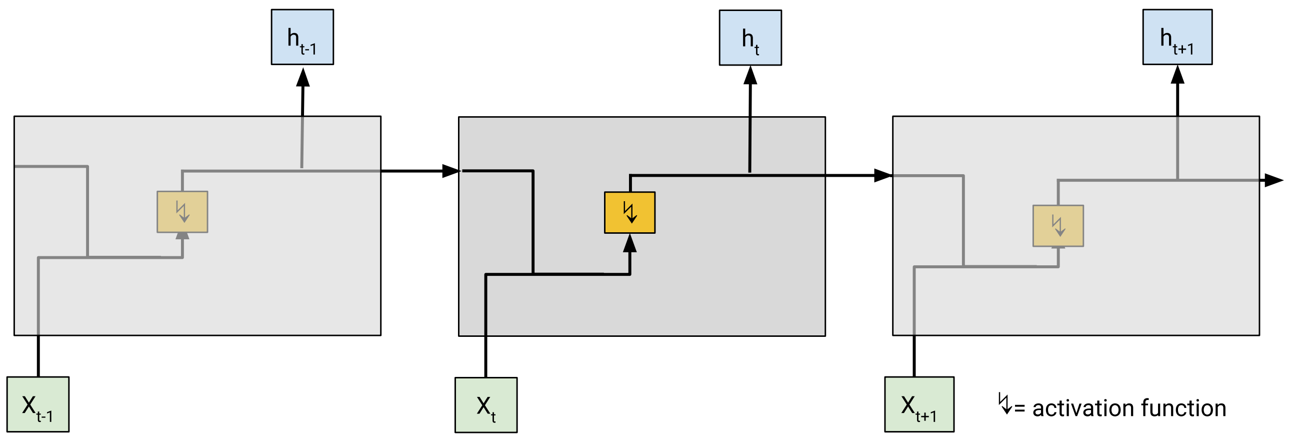
An implementation of **Keras** (#Keras) integrated into **TensorFlow** (#TensorFlow).

time series analysis

A subfield of machine learning and statistics that analyzes **temporal data** (#temporal_data). Many types of machine learning problems require time series analysis, including classification, clustering, forecasting, and anomaly detection. For example, you could use time series analysis to forecast the future sales of winter coats by month based on historical sales data.

timestep

One "unrolled" cell within a **recurrent neural network** (#recurrent_neural_network). For example, the following figure shows three timesteps (labeled with the subscripts t-1, t, and t+1):



tower

A component of a **deep neural network** (#deep_neural_network) that is itself a deep neural network without an output layer. Typically, each tower reads from an independent data source. Towers are independent until their output is combined in a final layer.

TPU

Abbreviation for **Tensor Processing Unit** (#TPU).

TPU chip

A programmable linear algebra accelerator with on-chip high bandwidth memory that is optimized for machine learning workloads. Multiple TPU chips are deployed on a **TPU device** (#TPU_device).

TPU device

A printed circuit board (PCB) with multiple **TPU chips** (#TPU_chip), high bandwidth network interfaces, and system cooling hardware.

TPU master

The central coordination process running on a host machine that sends and receives data, results, programs, performance, and system health information to the **TPU workers** (#TPU_worker). The TPU master also manages the setup and shutdown of **TPU devices** (#TPU_device).

TPU node

A TPU resource on Google Cloud Platform with a specific **TPU type** (#TPU_type). The TPU node connects to your VPC Network (https://cloud.google.com/vpc/docs/) from a peer VPC network (https://cloud.google.com/vpc/docs/vpc-peering). TPU nodes are a resource defined in the Cloud TPU API (https://cloud.google.com/tpu/docs/reference/rest/v1/projects.locations.nodes).

TPU Pod

A specific configuration of **TPU devices** (#TPU_device) in a Google data center. All of the devices in a TPU pod are connected to one another over a dedicated high-speed network. A TPU Pod is the largest configuration of **TPU devices** (#TPU_device) available for a specific TPU version.

TPU resource

A TPU entity on Google Cloud Platform that you create, manage, or consume. For example, **TPU nodes** (#TPU_node) and **TPU types** (#TPU_type) are TPU resources.

TPU slice

A TPU slice is a fractional portion of the **TPU devices** (#TPU_device) in a **TPU Pod** (#TPU_Pod). All of the devices in a TPU slice are connected to one another over a dedicated high-speed network.

TPU type

A configuration of one or more **TPU devices** (#TPU_device) with a specific TPU hardware version. You select a TPU type when you create a **TPU node** (#TPU_node) on Google Cloud Platform. For example, a `v2-8` TPU type is a single TPU v2 device with 8 cores. A `v3-2048` TPU type has 256 networked TPU v3 devices and a total of 2048 cores. TPU types are a resource defined in the Cloud TPU API (https://cloud.google.com/tpu/docs/reference/rest/v1/projects.locations.acceleratorTypes).

TPU worker

A process that runs on a host machine and executes machine learning programs on **TPU devices** (#TPU_device).

training

The process of determining the ideal **parameters** (#parameter) comprising a model.

training set

The subset of the dataset used to train a model.

Contrast with **validation set** (#validation_set) and **test set** (#test_set).

trajectory

RL

In reinforcement learning, a sequence of tuples (https://wikipedia.org/wiki/Tuple) that represent a sequence of **state** (#state) transitions of the **agent** (#agent), where each tuple corresponds to the state, **action** (#action), **reward** (#reward), and next state for a given state transition.

transfer learning

Transferring information from one machine learning task to another. For example, in multi-task learning, a single model solves multiple tasks, such as a **deep model** (#deep_model) that has different output nodes for different tasks. 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 less data.

Most machine learning systems solve a *single* task. Transfer learning is a baby step towards artificial intelligence in which a single program can solve *multiple* tasks.

translational invariance

In an image classification problem, an algorithm's ability to successfully classify images even when the position of objects within the image changes. For example, the algorithm can still identify a dog, whether it is in the center of the frame or at the left end of the frame.

See also **size invariance** (#size_invariance) and **rotational invariance** (#rotational_invariance).

trigram

An **N-gram** (#N-gram) in which N=3.

true negative (TN)

An example in which the model *correctly* predicted the **negative class** (#negative_class). For example, the model inferred that a particular email message was not spam, and that email message really was not spam.

true positive (TP)

An example in which the model *correctly* predicted the **positive class** (#positive_class). For example, the model inferred that a particular email message was spam, and that email message really was spam.

true positive rate (TPR)

Synonym for **recall** (#recall). That is:

$$\text{True Positive Rate} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

True positive rate is the y-axis in an **ROC curve** (#ROC).

U

unawareness (to a sensitive attribute)

A situation in which **sensitive attributes** (#sensitive_attribute) are present, but not included in the training data. Because sensitive attributes are often correlated with other attributes of one's data, a model trained with unawareness about a sensitive attribute could still have **disparate impact** (#disparate_impact) with respect to that attribute, or violate other **fairness constraints** (#fairness_constraint).

underfitting

Producing a model with poor predictive ability because the model hasn't captured the complexity of the training data. Many problems can cause underfitting, including:

- Training on the wrong set of features.
- Training for too few epochs or at too low a learning rate.
- Training with too high a regularization rate.
- Providing too few hidden layers in a deep neural network.

unlabeled example

An example that contains **features** (#feature) but no **label** (#label). Unlabeled examples are the input to **inference** (#inference). In **semi-supervised** (#semi-supervised_learning) and **unsupervised** (#unsupervised_machine_learning) learning, unlabeled examples are used during training.

unsupervised machine learning

Training a **model** (#model) to find patterns in a dataset, typically an unlabeled dataset.

The most common use of unsupervised machine learning is to cluster data into groups of similar examples. For example, an unsupervised machine learning algorithm can cluster songs together based on various properties of the music. The resulting clusters can become an input to other machine learning algorithms (for example, to a music recommendation service). Clustering can be helpful in domains where true labels are hard to obtain. For example, in domains such as anti-abuse and fraud, clusters can help humans better understand the data.

Another example of unsupervised machine learning is principal component analysis (PCA) (https://wikipedia.org/wiki/Principal_component_analysis). For example, applying PCA on a dataset containing the contents of millions of shopping carts might reveal that shopping carts containing lemons frequently also contain antacids.

Compare with **supervised machine learning** (#supervised_machine_learning).

upweighting

Applying a weight to the **downsampled** (#downsampling) class equal to the factor by which you downsampled.

user matrix

In **recommendation systems** (#recommendation_system), an **embedding** (#embeddings) generated by **matrix factorization** (#matrix_factorization) that holds latent signals about user preferences. Each row of the user matrix holds information about the relative strength of various latent signals for a single user. For example, consider a movie recommendation system. In this system, the latent signals in the user matrix might represent each user's interest in particular genres, or might be harder-to-interpret signals that involve complex interactions across multiple factors.

The user matrix has a column for each latent feature and a row for each user. That is, the user matrix has the same number of rows as the target matrix that is being factorized. For example, given a movie recommendation system for 1,000,000 users, the user matrix will have 1,000,000 rows.

V

validation

A process used, as part of **training** (#training), to evaluate the quality of a **machine learning** (#machine_learning) model using the **validation set** (#validation_set). Because the validation set is disjoint from the training set, validation helps ensure that the model's performance generalizes beyond the training set.

Contrast with **test set** (#test_set).

validation set

A subset of the dataset—disjoint from the training set—used in **validation** (#validation).

Contrast with **training set** (#training_set) and **test set** (#test_set).

vanishing gradient problem

The tendency for the gradients of early **hidden layers** (#hidden_layer) of some **deep neural networks** (#deep_neural_network) to become surprisingly flat (low). Increasingly lower gradients result in increasingly smaller changes to the weights on nodes in a deep neural network, leading to little or no learning. Models suffering from the vanishing gradient problem become difficult or impossible to train. **Long Short-Term Memory** (#Long_Short-Term_Memory) cells address this issue.

Compare to **exploding gradient problem** (#exploding_gradient_problem).

W

Wasserstein loss

One of the loss functions commonly used in **generative adversarial networks** (#generative_adversarial_network), based on the earth-mover's distance (https://wikipedia.org/wiki/Earth_mover%27s_distance) between the distribution of generated data and real data.

Wasserstein Loss is the default loss function in TF-GAN (https://github.com/tensorflow/tensorflow/tree/master/tensorflow/contrib/gan).

weight

A coefficient for a **feature** (#feature) in a linear model, or an edge in a deep network. The goal of training a linear model is to determine the ideal weight for each feature. If a weight is 0, then its corresponding feature does not contribute to the model.

Weighted Alternating Least Squares (WALS)

An algorithm for minimizing the objective function during **matrix factorization** (#matrix_factorization) in **recommendation systems** (#recommendation_system), which allows a downweighting of the missing examples. WALS minimizes the weighted squared error between the original matrix and the reconstruction by alternating between fixing the row factorization and column factorization. Each of these optimizations can be solved by least squares **convex optimization** (#convex_optimization). For details, see the Recommendation Systems course (/machine-learning/recommendation/collaborative/matrix)

wide model

A linear model that typically has many **sparse input features** (#sparse_features). We refer to it as "wide" since such a model is a special type of **neural network** (#neural_network) with a large number of inputs that connect directly to the output node. Wide models are often easier to debug and inspect than deep models. Although wide models cannot express nonlinearities through **hidden layers** (#hidden_layer), they can use transformations such as **feature crossing** (#feature_cross) and **bucketization** (#bucketing) to model nonlinearities in different ways.

Contrast with **deep model** (#deep_model).

width

The number of **neurons** (#neuron) in a particular **layer** (#layer) of a **neural network** (#neural_network).