A

accuracy

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

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

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

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

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

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.

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).

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.

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 (https://en.wikipedia.org/wiki/Partial_derivative) of the error with respect to each parameter is calculated in a backward pass through the graph.

baseline

A simple **model** (#model) or heuristic used as reference point for comparing how well a model is performing. A baseline helps model developers quantify the minimal, expected performance on a particular problem.

batch

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

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 and inference; however, TensorFlow does permit dynamic batch sizes.

bias

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 **prediction bias** (#prediction_bias).

binary classification

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

binning

See **bucketing** (#bucketing).

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 sampling

A training-time optimization in which a probability is calculated for all the positive labels, using, for example, 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.

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_classification) model that identifies dog breeds, the classes would be *poodle*, *beagle*, *pug*, and so on.

class-imbalanced data set

A **binary classification** (#binary_classification) problem in which the **labels** (#label) for the two classes have significantly different frequencies. For example, a disease data set 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.

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*.

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 that the model predicted, and the other axis is the actual label. N represents the number of classes. 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

	Tumor (predicted)	Non-Tumor (predicted)
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), and incorrectly classified 1 as not having a tumor (1 false negative). Similarly, of 458 samples that actually did not have tumors, 452 were correctly classified (452 true negatives) and 6 were incorrectly classified (6 false positives).

The confusion matrix of a multi-class confusion matrix 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. The confusion matrix contains sufficient information to calculate a variety of performance metrics, including precision and recall.

continuous feature

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

convergence

Informally, often refers to a state reached during training in which training **loss** (#loss) and 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, 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 Convex Optimization by Boyd and Vandenberghe (http://stanford.edu/~boyd/cvxbook/).

convex function

A function typically shaped approximately like the letter U or a bowl. However, in degenerate cases (https://en.wikipedia.org/wiki/Degeneracy_(mathematics)), a convex function is shaped like a line. For example, the following are all convex functions:

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

Convex functions are popular loss functions. That's because when a minimum value exists (as is often the case), many variations of **gradient descent** (#gradient_descent) are guaranteed to find a point close to the minimum point of the 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.

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

Deep models are usually *not* convex functions. Remarkably, algorithms designed for convex optimization tend to work reasonably well on deep networks anyway, even though they rarely find a minimum.

cost

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

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).

D

data set

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

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:

deep model

A type of **neural network** (#neural_network) containing multiple **hidden layers** (#hidden_layer). Deep models rely on trainable nonlinearities.

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

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).

derived feature

Synonym for **synthetic feature** (#synthetic_feature).

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).

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://www.jmlr.org/papers/volume15/srivastava14a.old/source/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

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 data set** (#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.

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

empirical risk minimization (ERM)

Choosing the model 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.

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 Estimators (as described here (https://www.tensorflow.org/extend/estimators)) or instantiate **pre-made Estimators** (#pre-made_Estimator) created by others.

example

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

F

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 (FP rate)

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

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

feature

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

feature columns (FeatureColumns)

A set of related features, such as the set of all possible countries in which users might live. An example may have one or more features present in a feature column.

Feature columns in TensorFlow also encapsulate metadata such as:

- the feature's data type
- whether a feature is fixed length or should be converted to an embedding

A feature column can contain a single feature.

"Feature column" is Google-specific terminology. A feature column is referred to as a "namespace" in the VW (https://en.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 (multiplying or taking a Cartesian product of) individual features. 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 set

The group of **feature** (#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).

full softmax

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

G

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.

generalized linear model

A generalization of **least squares regression** (https://developers.google.com/machine-learning/glossary/least_squares_regression) models, which are based on Gaussian noise (https://en.wikipedia.org/wiki/Gaussian_noise), to other types of models based on other types of noise, such as Poisson noise (https://en.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 (https://en.wikipedia.org/wiki/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."

gradient

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

gradient clipping

Capping **gradient** (#gradient) values before applying them. Gradient clipping helps ensure numerical stability and prevents exploding gradients

(http://www.cs.toronto.edu/~rgrosse/courses/csc321_2017/readings/L15%20Exploding%20and%20Vanishing%20Gradients.pdf)

.

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 (https://www.tensorflow.org/api_docs/python/tf/Tensor)) as an operand to another operation. Use **TensorBoard** (#TensorBoard) to visualize a graph.

H

heuristic

A practical and nonoptimal solution to a problem, which is sufficient for making progress or for learning from.

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). A neural network contains one or more hidden layers.

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:

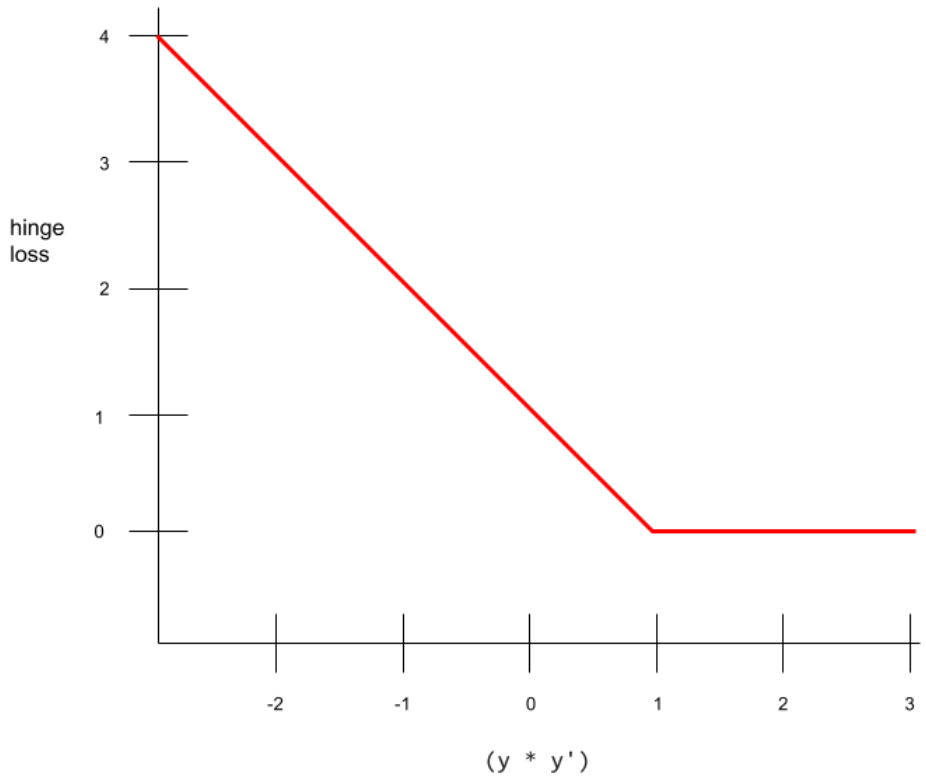
$$loss = max(0, 1 - (y' * y))$$

where *y'* is the raw output of the classifier model:

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

and *y* is the true label, either -1 or +1.

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 data set** (#validation_set) and **test data set** (#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 data set 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).

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://en.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.

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://en.wikipedia.org/wiki/Statistical_inference).)

input layer

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

instance

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

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://en.wikipedia.org/wiki/Cohen%27s_kappa), which is one of the most popular inter-rater agreement measurements.

K

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 data set consists of a hundred features. In order to maximize the margin between positive and negative classes, KSVMs could internally map those features into a million-dimension space. KSVMs uses a loss function called hinge loss (#hinge-loss).

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 data set consists of one or more features and a label. For instance, in a housing data set, 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).)

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.

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 regression

A type of **regression model** (#regression_model) that outputs a continuous value from a linear combination of input features.

logistic regression

A model that generates a probability for each possible discrete label value in classification problems by applying a **sigmoid function** (#sigmoid_function) to a linear prediction. Although logistic regression is often used in **binary classification** (#binary_classification) problems, it can also be used in **multi-class** (#multi-class) classification problems (where it becomes called **multi-class logistic regression** or **multinomial regression**).

Log Loss

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

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).

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.

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).

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.

ML

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

model

The representation of what an ML system has learned from the training data. This 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 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 in 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 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).

N

NaN trap

When one number in your model becomes a NaN (https://en.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."

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.

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).

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.

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 data set 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-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). Different optimizers (subclasses of `tf.train.Optimizer`) account for concepts such as:

- momentum (https://www.tensorflow.org/api_docs/python/tf/train/MomentumOptimizer) (Momentum)
- update frequency (AdaGrad (https://www.tensorflow.org/api_docs/python/tf/train/AdagradOptimizer) = ADAptive GRADient descent; Adam (https://www.tensorflow.org/api_docs/python/tf/train/AdamOptimizer) = ADAptive with Momentum; RMSProp)
- 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).

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.

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 ML frameworks, including TensorFlow, support pandas data structures as input. See pandas documentation (http://pandas.pydata.org/).

parameter

A variable of a model that the ML system trains on its own. For example, **weights** (#weight) are parameters whose values the ML 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.

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.

partitioning strategy

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

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 ML. 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:

$$P = 2^{-crossentropy}$$

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.

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).

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:

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

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 data set.

pre-made Estimator

An **Estimator** (#Estimator) that someone has already built. TensorFlow provides several pre-made Estimators, including `DNNClassifier`, `DNNRegressor`, and `LinearClassifier`. You may build your own pre-made Estimators by following these instructions (https://www.tensorflow.org/extend/estimators).

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.

Q

queue

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

R

rank

Overloaded term in ML that can mean either of the following:

- 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.
- The ordinal position of a class in an ML 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).

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:

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

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.

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).

representation

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

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).

S

Saver

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

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 data set. For example, suppose that

you want all floating-point features in the data set 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 ML platform. See www.scikit-learn.org (http://www.scikit-learn.org/).

sequence model

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

session

Maintains state (for example, variables) within a TensorFlow program.

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:

$$\sigma = b + w_1 x_1 + w_2 x_2 + \ldots w_n x_n$$

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).

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).

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).

static model

A model that is trained offline.

stationarity

A property of data in a data set, 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 data set to calculate an estimate of the gradient at each step.

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 model 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).

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) that is not present among the input features, but is derived from one or more of them. Kinds of synthetic features include the following:

- Multiplying one feature by itself or by other feature(s). (These are termed **feature crosses** (#feature_cross).)
- Dividing one feature by a second feature.
- **Bucketing** (#bucketing) a continuous feature into range bins.

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

T

target

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

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.

Tensor Processing Unit (TPU)

An ASIC (application-specific integrated circuit) that optimizes the performance of TensorFlow programs.

Tensor rank

See **rank** (#rank).

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.

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** (#hyperparameters) 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.

test set

The subset of the data set 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.

training

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

training set

The subset of the data set used to train a model.

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

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 (TP rate)

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

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

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

U

unlabeled example

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

unsupervised machine learning

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

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://en.wikipedia.org/wiki/Principal_component_analysis). For example, applying PCA on a data set 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).

V

validation set

A subset of the data set—disjunct from the training set—that you use to adjust **hyperparameters** (#hyperparameter).

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

W

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.

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).