# Accuracy

The fraction of predictions that a **Classification Model** got right. In Multi-Class Classification, Accuracy is defined as follows:

For **Binary Classification**, Accuracy can also be calculated in terms of positives and negatives:

# Activation Function

A function, such as **ReLU** or **Sigmoid**, 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.

Any mathematical function can serve as an Activation Function. Let’s use the (sigma) symbol to represent a given Activation Function. Thus, the value of a node in the network is given by the following formula:

# AUC (Area Under the ROC Curve)

An evaluation metric that considers all possible **Classification Thresholds**.

The Area Under the ROC curve 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.

# Backpropagation

The primary algorithm for performing **Gradient Descent** on **Neural Networks**.

First, the output values of each node are calculated (and cached) in a forward pass. Then, the partial derivative of the error with respect to each parameter is calculated in a backward pass through the graph.

# Bag of Words Representation

**Bag of Words** **Representation** represents the words in a phrase, independent 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, 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

If we pick the sample phrase John likes to watch movies. Mary likes movies too., we could derive from it a list of words to be represented in using Bag of Words:

"John", "likes", "to", "watch", "movies", "Mary", "likes", "movies", "too"

Now, we can use the Bag of Words Representation to map each word to, in this case, its frequency in the sample phrase:

{"John":1, "likes":2, "to":1, "watch":1, "movies":2, "Mary":1, "too":1}

# Batch

In **Gradient Descent**, a batch is the total number of examples you use to calculate the gradient in a single iteration.

# Bias

An intercept or offset from an origin. Bias (or Bias Term) is referred to as b or w0 in Machine Learning models.

For example, Bias is the b in the following formula:

# Binary Classification

A type of classification task that outputs only one of two possible classes.

For example, a Machine Learning model that evaluates email messages and outputs either "spam" or "not spam" is a binary classifier.

# Bucketing

Bucketing consists in the conversion of a (usually continuous) 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 with a single decimal point, temperatures between 0.0 and 15.0 degrees could be put into one bin, 15.1 to 30.0 degrees in a second bin, 30.1 to 50.0 degrees in a third bin and so on.

# Calibration Layer

A post-prediction adjustment, typically to account for **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 are interested in determining whether an input image is a beagle or a bloodhound, in the context of a dog image classifier model, we don't have to provide probabilities for every example that is not of a dog.

The idea is that the negative classes can learn from less frequent negative reinforcement as long as positive classes always get proper positive reinforcement, something that is observed empirically. The motivation for Candidate Sampling is a computational efficiency win from not computing predictions for all negatives.

# Class-Imbalanced Dataset

A **Binary Classification** problem in which the labels for the two classes have significantly different frequencies.

For example, a disease dataset in which 0.01% of examples have positive labels and 99.99% have negative labels is a class-imbalanced problem, but a football game predictor in which 51% of examples label one team winning and 49% label the other team winning is not a class-imbalanced problem.

# Classification Matrix

An NxN table that summarizes how successful a Classification Model's 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.

The following is the Classification Matrix for a Classification Model that predicted whether a patient had a tumor (“Tumor”) or did not have a tumor (“Non-tumor”):

|  |  |  |
| --- | --- | --- |
|  | Tumor (predicted) | Non-tumor (predicted) |
| Tumor (positive class) | 18 | 1 |
| Non-tumor (negative class) | 6 | 452 |

The preceding Confusion Matrix shows that, of the 19 (18 + 1) 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 (6 + 452) samples that actually did not have tumors, 452 were correctly classified (452 true negatives) and 6 were incorrectly classified (6 false positives).

Confusion Matrices contain sufficient information to calculate a variety of performance metrics, including **Precision** and **Recall**.

# Classification Models

A classification model that predicts discrete values.

For example, classification models make predictions about whether a given email is spam or whether a given image is an image of a dog or a cat.

# Classification Threshold

A scalar-value criterion that is applied to a model's predicted score in order to separate the positive class from the negative class. The Classification Threshold is used when mapping **Logistic Regression** results to 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.

The Classification Threshold is also known as the **Decision Threshold**.

# Clipping

The process of converting values are higher or lower than a given limit to the value of that limit.

For example, given a rooms\_per\_person feature that contains the majority of its values in the range [1.0, 4.0], but still contains a handful of values above the upper bound of 4.0, you could clip those values to the bound, that is, values that were previously bigger than 4.0 are now simply converted to that upper bound value. This way, you can ensure the feature won’t contain any values bigger than 4.0.

# Clustering

**Clustering** is the most well-known unsupervised learning technique.

This technique finds structure in unlabeled data by identifying similar groups or clusters.

# Collaborative Filtering

**Collaborative Filtering** is responsible for making predictions about the interests of one user based on the interests of many other users.

Collaborative filtering is often used in recommendation systems.

# Convergence

A Machine Learning model reaches convergence when additional training on the current data will not improve the model, that is, it has reached a state during training in which training loss and validation loss change very little or not at all with each iteration after a certain number of iterations.

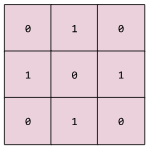
# Convex Optimization

The process of using mathematical techniques such as **Gradient Descent** to find the minimum of a convex function.

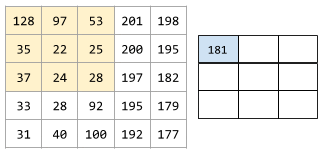
# Convolutional Layer

A **Convolutional Layer** is layer of a **Deep Neural Network** in which a convolutional filter passes along an input matrix.

For example, consider the following 3x3 convolutional filter:

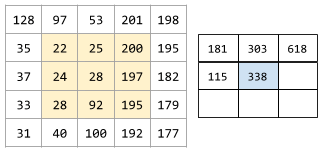


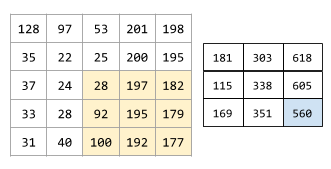
Then, for each cell highlighted by the 3x3 convolutional filter in the following 5x5 input matrix, each cell is multiplied by the corresponding value in the filter cell (that is, it obtains a product for each filter-matrix cell pair). For example, in this particular instance:



The value 181 is obtained by performing element-wise multiplication, that is, it multiplies each filter-tile pair of cells, with the final result being the sum of all the products:

At the end, after the filter passes over all the possible 3x3 slices of the input we obtain the final output feature map, also known as a convoluted feature (the 3x3 matrix shown on the right):





# Convolutional Neural Network (CNN)

A **Convolutional Neural Network** is a specific type of **Neural Network** in which at least one layer is a **Convolutional Layer**. A typical Convolutional Neural Network consists of some combination of the following layers:

* convolutional layers
* pooling layers
* dense layers

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

# Data Augmentation

The process of artificially increasing the range and number of training examples by randomly transforming certain examples to create new versions of these.

For example, if you have a small dataset of images to train your model with, Data Augmentation can apply various transformations to existing images, such as rotation, stretching, reflection, etc. so that you end up with multiple variants of the original images and, in result, increase the size of your dataset.

# Definition

Machine Learning systems learn how to combine input to produce useful predictions on never-before-seen data.

# Dense Representation

A representation of a tensor (vector) that stores all elements, no matter the length of possible values.

For example, given a feature street\_name that has the possible values {'Charleston Road', 'North Shoreline Boulevard', 'Shorebird Way', 'Rengstorff Avenue'}, the tensor must represent all these values with the corresponding weight.

If a street is located on North Shoreline Boulevard, then the tensor could be stored as [0, 1, 0, 0]. However, if the street was located, for example, at the corner between North Shoreline Boulevard and Shorebird Way, the vector could hold two elements with a weight of 1: [0, 1, 1, 0].

What’s most important in Dense Representation is that it represents the weight for all possible values, be no matter the weight. This contrasts with **Sparse Representation** which holds only the nonzero values (as a tensor).

# Dropout Regularization

A form of **Regularization** useful in training **Neural Networks**.

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:

* 0.0 = No Dropout Regularization
* 1.0 = Drop out everything (the model learns nothing)
* Values between 0.0 and 1.0 = More useful

This is analogous to training the network to emulate an exponentially large ensemble of smaller networks.

# Dynamic Model

A Machine Learning model that is trained online in a continuously updating fashion, that is, data is continuously entering the model.

# Early Stopping

A method for **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 starts to increase, that is, when generalization performance worsens.

# Embedding

**Embeddings** represent categorical features as continuous-valued features. Typically, an Embedding is a translation of a high-dimensional vector into a low-dimensional space.

Ideally, an Embedding captures some of the semantics of the input by placing semantically similar inputs close together in the embedding space. An Embedding can be learned and reused across models.

# Empirical Risk Minimization

**Training** a model simply means learning (determining) good values for all the weights and the bias from labeled examples.

In supervised learning, a Machine Learning algorithm builds a model by examining many examples and attempting to find a model that minimizes loss. This process is called **empirical risk minimization**.

# Example

Examplesare a particular instance of data x.

Examples can be separated into **labeled** and **unlabeled examples**.

A labeled example contains both the features and the correspondent label, such as {features, label} or (x, y)*.* These examples should be used to train the model.

Anunlabeled example contains only the features of a given example, such as {features, ?} or (x, ?)*.* These examples are used to make the model predict the label of unlabeled examples.

# F1 Score

In statistical analysis of **Binary Classification**, the F1 Score (also known as **F-score** or **F-measure**) is a measure of a test's accuracy.

F1 Score considers both the Precision and the Recall of the test to compute the score.

The F1 Score is the harmonic average of the Precision and Recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

The mathematical formula for calculating the F1 score is the following:

# False Negative (FN)

An example in which the model mistakenly predicted the **Negative Class**.

For example, the model inferred that a particular email message was not spam (the negative class), but that email message actually was spam.

# False Positive (FP)

An example in which the model mistakenly predicted the **Positive Class**.

For example, the model inferred that a particular email message was spam (the positive class), but that email message was actually not spam.

# Feature Cross

A Feature Cross is a **Synthetic Feature** formed by crossing (taking a Cartesian product of) individual binary features obtained from categorical data or from continuous features via **Bucketing**.

Feature Crosses help represent nonlinear relationships.

# Feature Engineering

The process of determining which features might be useful in training a model, and then converting raw input data into said features.

In TensorFlow, Feature Engineering often means converting raw log file entries to *tf.Example* protocol buffers.

Feature Engineering is sometimes called **Feature Extraction**.

# Features

Features are input variables describing our data, the *x* variable in simple linear regression.

Machine learning can use just a single feature but could also use as many as needed, which is denoted by {x1, x2, ... xn}.

Features can be, for example, the address of an email’s sender, specific words in the text of a message, etc.

# Fully Connected Layers

We call a pair of layers in a **Neural Network** fully connected layers when every node in the first layer is connected to every node in the second layer.

# Generalization

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

# Gradient

The gradient of f(x, y) is a two-dimensional vector that tells you in which *(x, y)* direction to move for the maximum increase in height. Thus, the negative of the gradient moves you in the direction of maximum decrease in height. In other words, the negative of the gradient vector points into the valley.

In a more abstract definition, the gradient is the vector of partial derivatives with respect to all of the independent variables.

# Gradient Clipping

Caps gradient before applying them.

Gradient Clipping ensures the magnitude of the gradients do not become too large during training, which can cause gradient descent to fail.

# Gradient Descent

Gradient Descent is an algorithm to minimize loss by computing the gradients of loss with respect to the model's parameters, conditioned on training data.

In other words, gradient descent iteratively adjusts parameters, gradually finding the best combination of weights and bias to minimize loss.

# Gradient Step

The resulting product of multiplying the learning rate (a scalar) by the gradient, used by the **Gradient Descent** algorithm.

# Hidden Layer

A synthetic layer in a **Neural Network** between the **Input Layer** (the features) and the **Output Layer** (the prediction).

A Neural Network contains one or more hidden layers.

# Hyperparameter

The elements tweaked during successive runs of training of a Machine Learning model.

# Independently and Identically Distributed (i.i.d)

Data drawn from a distribution that doesn't change, where each value drawn doesn't depend on values that have been drawn previously. An i.i.d. is the ideal fuel 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.

# Input Layer

The first layer in a **Neural Network**.

This layer is responsible for receiving the input data (the features).

# K-Means

K-Means is the most popular clustering algorithm and, as the purpose implies, is used to separate data so that data similar to one another are in the same group, while data different from one another are in different groups.

This algorithm addresses two core aspects:

* The number of clusters (groups) we expect to separate the data into (the “K” part)
* The average distance of data points to each cluster center, known as the centroid (the “Means” part)

It is an iterative approach:

1. Place k random centroids for the initial clusters
2. Assign data samples to the nearest centroid
3. Update centroids based on the above-assigned data samples

Steps 2 and 3 are repeated until convergence is achieved, that is, when points don’t move between clusters and the centroids are stabilized.

# K-Nearest Neighbors Classifier

K-Nearest Neighbors (KNN) is a classification algorithm. The central idea is that data points with similar attributes tend to fall into similar categories.

In essence, the classifier finds the k nearest neighbors to the example in your dataset and classifies the unknown example with the class that’s prevalent in the k neighbors.

In the case of ties (where k is even), a common practice is to then classify the unknown example with the class of the closest known example.

Keep in mind that a low k often leads to overfitting and a large k often leads to underfitting.

# L1 Regularization

A type of **Regularization** that penalizes weights in proportion to the sum of the absolute values of the weights.

In models that rely on sparse features, L1 Regularization helps drive the weights of irrelevant or barely relevant features to exactly 0, which removes those features from the model.

# L2 Regularization

A type of **Regularization** that penalizes weights in proportion to the sum of the squares of the weights.

L2 regularization helps drive outlier weights (those with high positive or low negative values) closer to 0 but not quite to 0.

L2 regularization always improves generalization in linear models.

# Label

A label is what we are predicting, the *y* variable in simple linear regression.

A label can be the result of a binary decision such as whether an email is considered to be spam or more unpredictable results such as the future price of a product.

# Lambda

Synonym for **Regularization Rate**.

# Latent Dimension

When learning Embeddings, the individual dimensions are not learned with names. Sometimes, we can look at the Embeddings and assign semantic meanings to the dimensions, and other times we cannot. Often, each such dimension is called a **Latent Dimension**, as it represents a feature that is not explicit in the data but rather inferred from it.

# Learning Rate

A scalar used to train a model via **Gradient Descent**.

For each iteration, the Gradient Descent algorithm multiplies the learning rate by the Gradient, with the resulting product being called the **Gradient Step**.

The Learning Rate is considered an **Hyperparameter**.

# Linear Regression

Linear Regression is a method for finding the straight line or hyperplane that best fits a set of points.

Linear Regression is used when the output is continuous, such as weight, height, number of hours, etc.

A Linear Regression equation is of the form

, where:

* y is the value we’re trying to predict
* m is the slope of the graphical line represented by the equation
* x is the value of the input feature
* b is the value where the line intercepts the y-axis

By convention, in Machine Learning this equation is written in the form

, where:

* y’ is the predicted label (the output)
* b is the bias (the y-intercept), sometimes referred to as w0
* w1 is the weight of feature 1; weight is the same concept as the slope in the traditional equation
* x1 is a feature (a known input)

The equation for models that use multiple features would look something like

To infer/predict an output (y’), just substitute the x1 by actual values.

# Log Loss

The loss function used in binary Logistic Regression.

Log Loss is mathematically defined as:

Where:

* is the dataset containing many labeled (x, y) examples
* y is the label in the labeled example. Given that this is Logistic Regression, every value of y must be either 0 or 1
* y’ is the predicted value (in the range of 0 to 1), given the set of features in x

# Logistic Regression

A model that generates a probability for each possible discrete label value in classification problems by applying a **Sigmoid Function** to a linear prediction.

Although Logistic Regression is often used in binary classification problems, it can also be used in multi-class classification problems. At this point, it’s called **Multi-Class Logistic Regression** or **Multinomial Regression**.

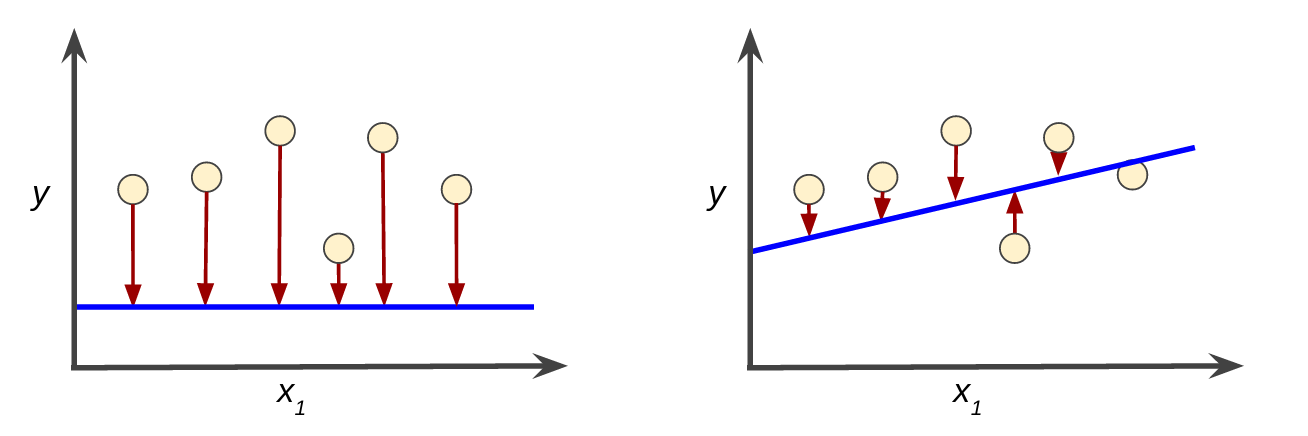
Usually, Logistic Regression is used when the output is categorical in nature, such as yes/no, true/false, red/green/blue, etc.

# Loss

Loss is the penalty for a bad prediction, that is, loss is a number indicating how bad the model's prediction was on a single example.

If the model's prediction is perfect, the loss is zero; otherwise, the loss is greater.

The goal of training a model is to find a set of weights and biases that have low loss, on average, across all examples.



# Machine Learning (ML)

A program or system that builds (trains) a predictive model from input data.

The model is fed **Examples**, pairs consisting of **Features** (variables that describe the data) and **Labels** (the “answer” of the example), which are used to learn and improve (train) what the model knows regarding the data it is being trained on. Later, the model can be used to make predictions on the labels of new data, that is, data that it has not seen before.

It can be used in simpler cases like judging whether an email is spam or not, the type of a flower or even to predict next week’s weather.

# Mean Square Error (MSE)

The Mean Square Error (MSE) is the average squared loss per example, over the whole dataset.

To calculate the MSE, sum all the squared losses for individual examples and then divide that by the number of examples:

, where:

* (x, y) is an example in which
  + x is the set of features the model uses to make predictions
  + y is the example’s label
* prediction(x) is a function of the weights and bias applied to the set of features *x*
* D is a dataset containing many labeled examples, which are (x, y) pairs
* N is the number of examples in D

Although MSE is commonly-used in Machine Learning, it is neither the only practical loss function nor the best loss function for all circumstances.

# Min-Max Normalization

**Min-Max Normalization** is one of the most common ways to normalize data.

For every feature, the minimum value of that feature gets transformed into 0, the maximum value gets transformed into 1, and every other value gets transformed into a floating point between 0 and 1.

Its mathematical formula is:

However, Min-Max Normalization has one fairly significant downside: it does not handle outliers very well. For example, if you have 99 values between 0 and 40 and one of those is 100, then all the values will be normalized to the 0 to 0.4 scale.

On the other hand, Min-Max Normalization guarantees all features will have the exact same scale.

# Mini-batch

A mini-batch is typically between 10 and 1,000 examples, chosen at random.

This smaller batch is, like a regular **Batch**, run together in a single iteration of a model’s training.

# Mini-batch Stochastic Gradient Descent (Mini-batch SGD)

A variant of **SGD** that uses a mini-batch, that is, a random small subset of the training data instead of a single example.

# Model

A model defines the relationship between features and label.

For example, a spam detection model might associate certain features strongly with "spam". A Machine Learning model covers both training and inference.

Inference means applying the trained model to unlabeled examples, that is, using the trained model to make useful predictions (y').

Training means creating or making the model learn, that is, feeding the model with labeled examples so it can gradually learn the relationships between features and label.

# Multi-Class Classification

Multi-Class Classification is a type of classification that distinguishes 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.

# Multi-Hot Encoding

A sparse vector that can contain multiple elements set to 1, with the rest being set to 0.

Multi-hot Encoding is commonly used to represent strings or identifiers that can have multiple values.

# NaN Trap

When one number in the model becomes a NaN (Not a Number) during training, which causes many or all other numbers in the model to eventually become a NaN.

# Negative Class

In **Binary Classification**, one class is defined positive and the other is termed negative. The **Positive Class** is the thing we're looking for, 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".

# Neural Network (NN)

A Neural Network is a model that is composed of layers (at least one of which is hidden), consisting of simple connected units or **Neurons** followed by nonlinearities.

A Neural Network is composed of the following elements:

* A set of nodes, equivalent to Neurons, organized in layers
* A set of **Weights** representing the connections between each Neural Network layer and the layer beneath it. The layer beneath may be another Neural Network layer, or some other kind of layer
* A set of **Biases**, one for each node
* An **Activation Function** that transforms the output of each node in a layer. Different layers may have different Activation Functions

# Neuron

A node in a Neural Network, typically taking in multiple input values and generating one output value.

The neuron calculates the output value by applying an **Activation Function** (nonlinear transformation) to a weighted sum of input values.

# Normalization

Normalization is the process of converting a given 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, these values can be normalized into the range -1 to +1.

# Offline Inference

Generating a group of predictions, storing those predictions, and then retrieving those predictions on demand.

# One-Hot Encoding

A sparse vector in which only a single element is set to 1, with the rest being set to 0.

One-hot Encoding is commonly used to represent strings or identifiers that have a finite set of possible values.

# One vs. All

Given a classification problem with N possible solutions, a One vs. All solution consists of N separate binary classifiers: 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 on demand.

# Out-of-Vocabulary (OOV)

A “catch-all” way of grouping all other possibilities that are not present in the vocabulary of a feature.

For example, given a street\_name feature that has {'Charleston Road', 'North Shoreline Boulevard', 'Shorebird Way', 'Rengstorff Avenue'} as its possible values, we could map the street names to integers to construct its vocabulary as such:

* map Charleston Road to 0
* map North Shoreline Boulevard to 1
* map Shorebird Way to 2
* map Rengstorff Avenue to 3
* map everything else (OOV) to 4

Here, class 4 will be our Out-of-Vocabulary class to cover all other street names that aren’t specified.

# Output Layer

The final layer of a **Neural Network**.

This layer contains the model’s answer, that is, the output.

# Overfitting

A Machine Learning model that matches the training data so closely that the model fails to make correct predictions on new data.

# Periods

Periods is a convenience variable that controls the granularity of reporting.

The following formula applies when talking about periods:

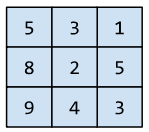
For example, if periods is set to 7, steps is set to 70 and the model is being trained with batch *size*s of 1, then the model will output the loss value every 10 steps, provided a print() call is included in the code, of course.

Note that modifying periods does not alter what the model learns.

# Pooling

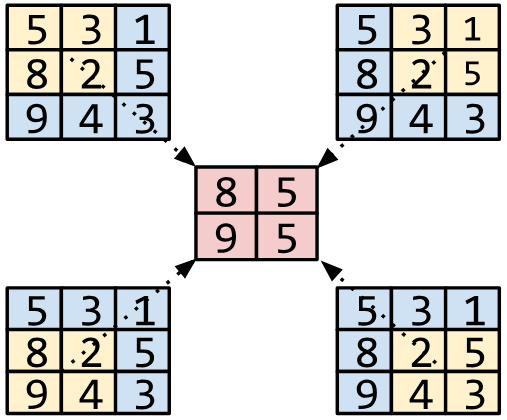
**Pooling** is the process of reducing a matrix (or matrices) created by an earlier convolutional layer to a smaller matrix. Pooling usually involves taking either the maximum or average value across the pooled area.

For example, suppose we have the following 3x3 matrix:



A pooling operation, just like a convolutional operation, divides that matrix into slices and then slides that convolutional operation by strides.

Suppose the pooling operation divides the convolutional matrix into 2x2 slices with a 1x1 stride. As the following figure illustrates, four pooling operations take place where each pooling operation picks the maximum value of the four in that slice:



# Positive Class

In **Binary Classification**, one class is defined positive and the other is termed negative. The Positive Class is the thing we're looking for, 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".

# Precision

A metric for **Classification Models**.

Precision identifies the frequency with which a model was correct when predicting the **Positive Class**. Its mathematical formula is:

# Prediction Bias

A value that indicates how far apart the average of predictions is from the average of labels in the dataset.

# Recall

A metric for **Classification Models** for calculating, out of all the possible positive labels, how many did the model correctly identify. Its mathematical formula is:

# Regression Models

A regression model predicts continuous values.

For example, regression models make predictions about the price of a house or the probability raining tomorrow.

# Regularization

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

* **L1 Regularization**
* **L2 Regularization**
* Dropout Regularization
* 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 equation shows the Regularization Rate's influence:

Raising the Regularization Rate reduces overfitting but may make the model less accurate.

# ReLU (Rectified Linear Unit)

An activation function with the following rules:

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

It is mathematically defined as:

# ROC Curve (Receiver Operating Characteristic)

A curve of true positive rate vs. false positive rate at different classification thresholds.

# Scaling

Scaling is used to make a range of values match the range of other features in the dataset.

For example, suppose that you want all floating-point features in the data set to have a range of 0 to 1. Given a feature's range of 0 to 500, you could scale that feature by dividing each value by 500.

# 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 mathematical formula:

With corresponding to the **Linear Regression** formula:

In other words, the Sigmoid Function converts into a probability between 0 and 1.

In some neural networks, the Sigmoid Function acts as the activation function.

# Softmax

A function that provides probabilities for each possible class in a **Multi-Class Classification** model. The computed probabilities add up to exactly 1.0.

For example, Softmax might determine that the probability of a particular image being a dog is 0.9, a cat 0.08, and a horse 0.02.

Softmax can also be called **Full Softmax**.

# Sparse Representation

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

For example, the English language consists of about a million words. Then, there would be two ways to represent a count of the words used in a single English sentence:

* A **Dense Representation** of the sentence must set an integer for all cells, setting a 0 in most of them, and a low integer in just a few
* A Sparse Representation of the sentence stores only cells that symbolize a word is in a sentence (cells with nonzero values). So, if the sentence contained only 20 unique words, then the Sparse Representation would make use of just 20 cells

|  |  |  |
| --- | --- | --- |
| Dense Representation | | |
| Cell Number | **Word** | **Present** |
| 0 | a | 0 |
| 1 | aardvark | 0 |
| 2 | aargh | 0 |
| 3 | aarti | 0 |
| **… 140,391 more words not present (0)** | | |
| 140395 | dogs | 1 |
| **… 633,062 words not present (0)** | | |
| 773458 | tails | 1 |
| **… 189,136 words not present (0)** | | |
| 962594 | wag | 1 |
| **… many more words not present (0)** | | |

|  |  |  |
| --- | --- | --- |
| Sparse Representation | | |
| Cell Number | **Word** | **Present** |
| 140395 | dogs | 1 |
| 773458 | tails | 1 |
| 962594 | wag | 1 |

# Squared Loss

The Squared Loss, also known as **L2 loss**, is used to calculate the loss for a single example of the model.

The squared loss is calculated as the square of the difference between the label and the prediction, that is

(observation – prediction(x))2, which is equivalent to

# Static Model

A Machine Learning 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. 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 Size

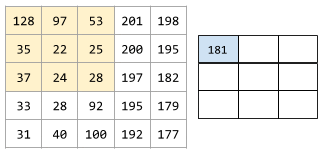
A synonym for **Learning Rate**.

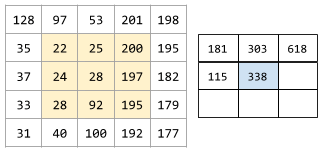
# Stride

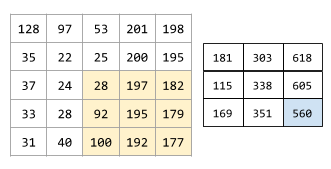
In a **convolutional operation** or **pooling**, the stride is the delta (difference, step) in each dimension of the next series of input slices.

For example, the following images are a part of a convolutional operation that uses a stride value of (1,1), that is, 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.







# Stochastic Gradient Descent (SGD)

**Stochastic Gradient Descent** (SGD) uses a single (random) example, that is, a batch size of one, per iteration. Given enough iterations, SGD works, but is very noisy.

# 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.

# Synthetic Feature

A Synthetic Feature is a feature not present among the input features, instead it’s created from one or more of them.

Kinds of synthetic features include:

* **Bucketing** a continuous feature into range bins
* Multiplying/dividing one feature value by other feature value(s) or by itself
* Creating a **Feature Cross**

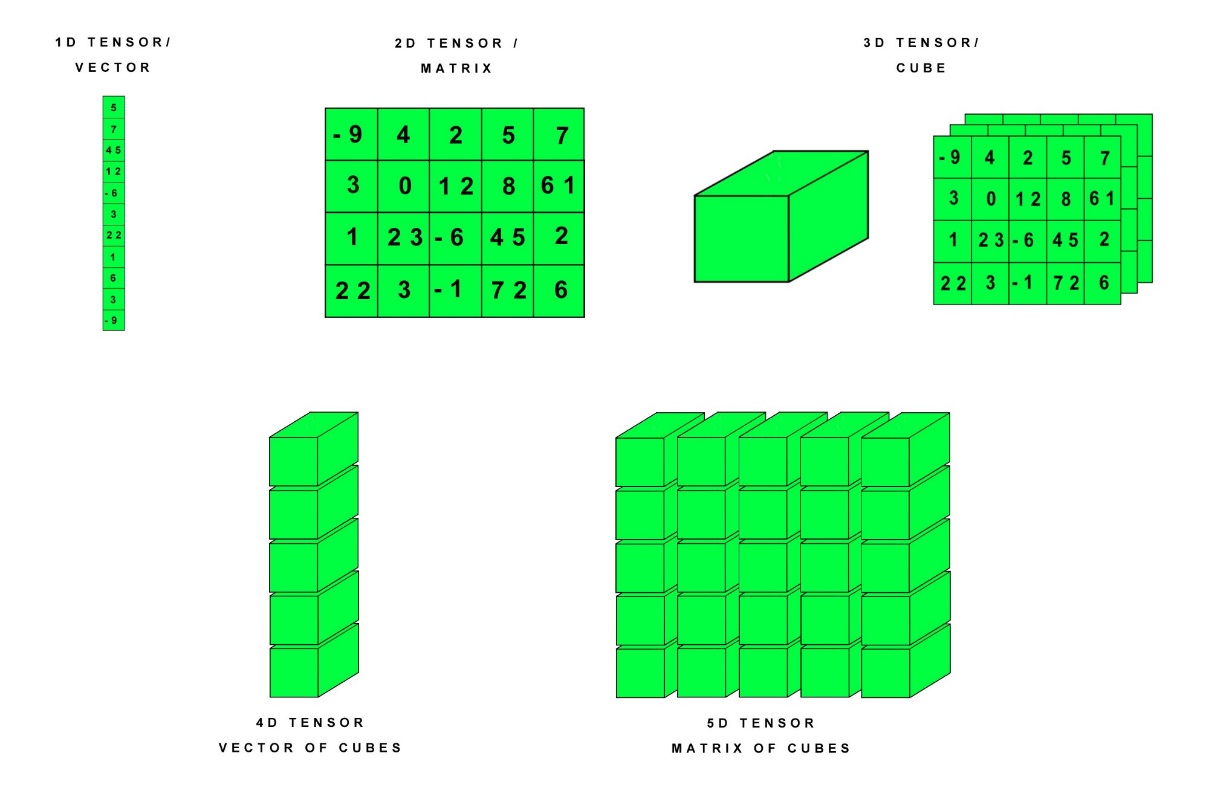
Features created by **Normalizing** or **Scaling** alone are not considered synthetic features.

# Tensor

At its core, a Tensor is a data container, most of the time a container for numbers (integers or floating-points).

Tensors can be of different dimensions:

* 0D Tensor- Scalar (a single number)
* 1D Tensor- Vector
* 2D Tensor- Matrix
* 3D Tensor- Cube
* 4D Tensor- Vector of Cubes
* 5D Tensor- Matrix of Cubes



# Test Set

The subset of the dataset used to test the model after the model has gone through initial vetting by the **Validation Set**.

# Training Set

The subset of the dataset used to train a model.

# Translational Invariance

In an image classification problem, the **Translational Invariance** is 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.

# True Negative (TN)

An example in which the model correctly predicted the **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**.

For example, the model inferred that a particular email message was spam, and that email message really was spam.

# 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**

# Vocabulary

A mapping of all the possible feature values to integers.

# Validation Set

A subset of the dataset (distinct from the **Training Set**) used to adjust **Hyperparameters**.

# Weight

A coefficient for a 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.

# Z-Score Normalization

**Z-Score Normalization** is a strategy for normalizing data while also avoiding issues with outliers.

Its mathematical formula is:

Where:

* value: the value to be normalized
* : the mean value of the feature which value belongs to
* : the standard deviation value of the feature which value belongs to

If a value is equal to the mean of all the values of the feature, it will be normalized to 0; if it is below the mean then it will be a negative number; if it is above the mean then it will be a positive number.

However, Z-Score Normalization does not normalize different features with the same scale.