APPENDIX 2

The Words of Machine Learning

One of the first hurdles for machine learning rookies is learning the vocabulary. So many words in ML sound familiar, and yet subtly foreign. You'll probably ask yourself questions like: "Am I supposed to already know what a *feature* is?", or "What does *numerically stable* mean, exactly?"

This appendix gives you quick definitions of common terms and expressions of ML. Most entries also refer you to the section in the book where a term is first introduced, in case you need something more than a quick reminder. These definitions have no pretense to be Wikipedia-worthy. They're here just to trigger your memory and point you at additional information.

Mgosoft PDF Split Merge

You won't find *all* the terms from the book in here—only those that I thought you might want to review. If you can't find the word you're looking for, then look it up in the book's index.

In the definitions, terms written in *italic* have their own entry in this appendix.

Activation function

The function that follows the *weighted sum* in each *layer* of a *neural network*—in this book, typically a *sigmoid*, a *softmax*, or a *ReLU*. I introduce the name "activation function" in Chapter 8, The Perceptron, on page 99—but a detailed discussion of activation functions comes much later, in Understanding Activation Functions, on page 229.

Accuracy

One of the performance metrics that you can use to evaluate a *classifier*. For example, if a program correctly classifies one fourth of the test examples, you can say that it has 25% accuracy.

Adam

A variant of *gradient descent*. See Gradient Descent on Steroids, on page 240.

Artificial neural network

A more technically precise name for a *neural network*. There was a time where people felt the need to specify that they're talking about an "artificial" neural network, as opposed to a biological one.

Asymptotic

A curve is asymptotic when it approaches a value without ever quite reaching it. If I say, for example, "this *loss* asymptotically approaches a value of 10," that means that the loss gets closer and closer to 10, but it never quite reaches it.

Backpropagation

The fundamental algorithm for training *neural networks*. Backpropagation—or "backprop," for friends,—calculates the gradients of the *loss* with respect to the *weights*. Chapter 11, Training the Network, on page 129 is dedicated to backpropagation.

Batch gradient descent

The "plain vanilla" version of *gradient descent*, where the gradient is calculated based on all the training examples, taken together as a single batch. By contrast, see *mini-batch gradient descent* and *stochastic gradient descent*.

Mgosoft PDF Split Merge

Batch normalization

One of the most effective techniques for improving neural networks. We briefly talk about it in One Last Trick: Batch Normalization, on page 244.

Bias

In many ML systems, including the ones in this book, the bias is one of the learnable *parameters* of the *model*. To see its mathematical meaning, check out Adding a Bias, on page 26 and Adding More Dimensions, on page 46. However, the term "bias" also has a second meaning in machine learning: you can say that a system has "high bias" when it *underfits* the training data, as I explain in Bias and Variance, on page 217.

Bias-variance trade-off

Bias and *variance* are alternative terms for *underfitting* and *overfitting*. The "trade-off" between them is a pragmatic concern: in many *supervised learning* systems, reducing overfitting can cause the system to underfit the training data, and vice versa. This balancing act is explored in Chapter 17, Defeating Overfitting, on page 211.

Bias column

In some machine learning models, the *bias* is a special case among *parameters*: other parameters are used in the *weighted sum*, but the bias

stands on its own. In Bye Bye, Bias, on page 59, I introduce a trick to get rid of this special case: by adding a column full of 1s to the inputs, we turn the bias into a *weight* like any other. This ad hoc column is called the "bias column."

Binary classifier

A *classifier* with an boolean output—for example, one that classifies the state of a hardware component as either "working" or "broken." If the classifier has more than two classes, then it's a *multiclass classifier*.

Broadcasting

A feature of some numerical libraries such as NumPy. Broadcasting allows you to treat arrays like you would treat individual values. For example, you can subtract two NumPy arrays: each element in the result is the subtraction of the matching elements in the two original arrays. You can see examples of broadcasting throughout this book, starting with Implementing Prediction, on page 21 and Upgrading the Loss, on page 56.

Categorical label

See *label*.

Chain rule

Mgosoft PDF Split Merge

The chain rule is the foundation of the *backpropagation* algorithm. It says that the *derivative* of a composite function is the product of the derivatives of the component functions. See From the Chain Rule to Backpropagation, on page 131.

Classes

The word "class" has different meanings in machine learning and programming. In ML, the classes are the categories that you use to organize data. For example, a *classifier* that recognizes digits has 10 classes, one for each digit; a classifier that recognizes dog breeds has one class per breed; and so on. By contrast, a class in programming is something else entirely—see Creating and Using Objects, on page 290.

Classifier

A system that assigns data to one of a limited set of *classes*. For example, you might label a movie review as "positive," "negative," or "neutral." The machine that reads the review and applies that label is a classifier. I introduce the concept in Chapter 5, A Discerning Machine, on page 63.

CNN

See *convolutional neural network*.

Computational graph

A notation to visualize the flow of data and operations in a system. Computational graphs are used often to represent neural networks, especially in the context of frameworks such as TensorFlow. I use computational graphs for the diagrams in Chapter 11, Training the Network, on page 129.

Computer vision

The scientific field that studies how computers can recognize images and videos. Today, most computer vision problems are solved with machine learning.

Convergence

In general, an algorithm converges when it finds a result, as opposed to running forever. In machine learning, this word is usually associated with algorithms like *gradient descent*. If GD converges, this means that it manages to minimize the *loss*; if it doesn't, that means that it oscillates indefinitely around the minimum loss, or even that it steps further and further from the minimum.

Convex

A mathematical function is convex when you can pick any two points on it, join them with a line, and the line won't cross any other point in the function. Intuitively, that means that the function has a single "cavity" (like, say, the exponential function) as opposed to multiple ones (like the sinus function).

Mgosoft PDF Split Merge

Convolution

A mathematical operation that serves as the basis for *convolutional neural networks*. We broadly show what convolutions look like in Convolutions, on page 252.

Convolutional Neural Network

A *neural network* architecture that is often used to learn data with a bi-dimensional or three-dimensional shape—in particular, images. Check out Chapter 19, Beyond Vanilla Networks, on page 247 for the details.

Cross-entropy loss

One of the formulae that we use to calculate the *loss*. It's a generalization of the *log loss*, and it's used for *multiclass classification* problems. See Writing the Classification Functions, on page 124.

Dead neuron

The phenomenon of dead neurons happens when an *activation function* in a neural network gets *saturated*, and its *gradient* gets close to 0. When

that happens, the *gradient descent* algorithm gets stuck, and the *nodes* that are downhill of the activation function stop learning. We describe this phenomenon in Dead Neurons, on page 141 and The Sigmoid and Its Consequences, on page 231.

Decision boundary

A *classifier* does its job by partitioning the space of the *examples* into areas. The decision boundary is the border between those areas. It's called "decision boundary" because data points on different sides of the boundaries are assigned to different *classes*. I know, that's a pretty abstract definition—but you can find concrete examples in Tracing a Boundary, on page 149.

Deep learning

To put it simply, "deep learning" is a short name for "machine learning done with *neural networks* that have many *layers*." More broadly, the term indicates a large set of techniques that are used today in ML, such as *convolutional neural networks*, *recurrent neural networks*, and many more. For a better definition, read Chapter 20, Into the Deep, on page 261.

Dense layer

The same as a *fully connected layer*.

Mgosoft PDF Split Merge

Dev set

Another name for the *validation set*.

Derivative

See the definition for *gradient*, and also the one for *partial derivative*.

Differentiable

A function is differentiable when you can calculate its *derivative* at any point. Intuitively, that happens when the function is smooth, and it doesn't have sudden jumps or holes. I use this term in When Gradient Descent Fails, on page 42.

Dropout

A bizarre (but powerful) *regularization* technique that randomly turns off *nodes* in a training *neural network*. See Advanced Regularization, on page 242.

Early stopping

A technique to counter *overfitting*, described in A Regularization Toolbox, on page 224. Early stopping is about finding the best moment to stop the *training phase*—after the system has learned enough to be useful, but before it starts overfitting the *training set*.

Epoch

In *mini-batch gradient descent* (or *stochastic gradient descent*), a training iteration processes only a fraction of the entire training set. By contrast, an "epoch" is a pass through the entire training set. For example, if you have 10,000 training examples, and you process them in batches of 10, it will take 1,000 iterations to complete an epoch.

Examples

See *supervised learning*.

Exploding gradient

A counterpart to the *vanishing gradient* problem. In the case of the exploding gradient, the *gradient* grows quickly as it *backpropagates* through a *neural network*. See Don't Play with Explosives, on page 234.

Feature

A common name to indicate the *input variables* in a machine learning system. For example, the pizza forecasting problem from Adding More Dimensions, on page 46 has three features: "Reservations," "Temperature," and "Tourists." In an image recognition problem, such as the one in Chapter 7, The Final Challenge, on page 87, you can treat each pixel as a separate feature.

Mgosoft PDF Split Merge

Feature engineering

Also known as "feature extractions." The process by which a human extracts basic *features* from data, that are then processed by a computer. One of the selling points of *deep learning* is that it often makes feature engineering unnecessary, as I explain in Unreasonable Effectiveness, on page 264.

Feature scaling

ML systems tend to work better if their input *features* span a similar range—for example, from 0 to 1. On the other hand, if some features can take much larger values than others, then you might want to rescale them to the same range. That's what "feature scaling" means. Read Preparing Data, on page 178 for more details, or check out the entry on *standardization* for an broader data preparation technique.

Forward propagation

The process by which data move from the input to the output of a supervised learning system. I use this term for the first time in Confidence and Doubt, on page 67—but it really starts making sense when I introduce *neural networks*, because it parallels the process of *backpropagation*.

Fully connected layer

A *layer* in a neural network where each *node* is connected to all the nodes in a neighboring layer. Also called a *dense layer*.

Fully connected network

A traditional *neural network*, made up entirely of *fully connected layers*. You usually say that a neural network is "fully connected" to tell it apart from other architectures, like *convolutional neural networks*.

Generative Adversarial Networks

One of the most striking ideas in deep learning, GANs are systems that generate realistic artificial data, usually images. See The Path of Image Generation, on page 269.

Global minimum

The minimum value over an entire function. For example, imagine a function that takes a date and returns the temperature in Chicago on that day. The global minimum of that function is on January 20, 1985, when the thermometers dropped to −27 °F—the lowest temperature ever recorded in Chicago. By contrast, a *local minimum* is a value that's lower than the ones around it. If the temperature drops a bit one one day, and then raises again on the following day, that's a local minimum.

Mgosoft PDF Split Merge

Glorot initialization

Another name for *Xavier initialization*.

Gradient

Intuitively, imagine the gradient as an arrow pointing "uphill" on a curve. The idea of a gradient is tightly connected to the concept of the *derivative*, that measures how a function changes as its input changes. If the derivative of the curve is zero at a specific point, that means the curve is flat at that point. If the derivative is positive, the curve is sloped upwards. If the derivative is negative, the curve is sloped downwards. The larger the derivative, the steeper the slope.

Gradient descent

One of the fundamental algorithms of machine learning. Gradient descent iteratively calculates the *gradient* of a system's *loss*, and changes the weights to step in the direction where the loss diminishes. GD is the subject of Chapter 3, Walking the Gradient, on page 31, and some of its variations are described in Chapter 13, Batchin' Up, on page 159 and Gradient Descent on Steroids, on page 240.

Ground truth

Observations collected from real-world data, as opposed to predictions. You often hear the term "ground truth" to refer to the *labels* in a *supervised learning* dataset, as in: "The system would have predicted 300 millimeters of rain for last year, but the ground truth was actually 400."

Hidden layer

Any layer in a *neural network* that is "inside" the network—that is, neither the *input layer* nor the *output layer*. See Chaining Perceptrons, on page 113.

Hyperparameters

In machine learning, there are two important types of parameters. On one side, there are the parameters of the *model function*, that the system learns during the *training phase*. On the other, there are parameters that you configure yourself before training—such as the *learning rate*, the number of training iterations, and the number of nodes in the *hidden layers* of a *neural network*. To avoid confusion between the two kinds of parameters, people use different names: the learnable parameters are often called *weights*, and the parameters that you configure yourself are called "hyperparameters."

Mgosoft PDF Split Merge

Identity function

The function that outputs the same value as its input. It can be used as an *activation function* in some corner cases, as we mention in Picking the Right Function, on page 237.

Input layer

The first *layer* in a *neural network*. It takes the values of the *input variables*. See Chaining Perceptrons, on page 113.

Input variable

See *supervised learning*.

L1/L2

Two common *regularization* techniques, explained in L1 and L2 Regularization, on page 220. They can be wrapped up like this: keep the *weights* in a *neural network* small, so that the network generates a smoother function.

Label

In *supervised learning*, each example is composed of an input and an expected output. The label is the expected output. A label can be either numerical or categorical. For example, in a dataset that tracks the effect

of temperatures on precipitation, the temperature would be the input variable, and the precipitation wold be a numerical label. On the other hand, a database of dog pictures might have dog images as the inputs, and the dog's breeds as their categorical labels.

Layers

The main units of organization of the *nodes* in a *neural network*. See the diagram in Here's the Plan, on page 118 for an example of a three-layered network. Other entities such as *dropouts* are sometimes called "layers" in the context of some machine learning libraries.

Leaky ReLU

See *ReLU*.

Learning rate

One of the most important *hyperparameters* of the *gradient descent* algorithm. At each step of GD, the gradient of the weights is multiplied by the learning rate—hence, the bigger the learning rate, the bigger the step. You meet this hyperparameter very early in the book, in Closer and Closer, on page 24. Later on, in Hands On: Tweaking the Learning Rate, on page 30 and Tuning the Learning Rate, on page 185, I discuss the trade-offs involved in making the learning rate smaller or larger.

Mgosoft PDF Split Merge

Learning rate decay

A variant of plain vanilla *gradient descent* where the *learning rate* progressively decreases during training. See Gradient Descent on Steroids, on page 240.

Linear

Intuitively, a function is linear when it can be plotted as a straight shape. Also check out *nonlinear*.

Linear regression

A tecnique that predicts an input from an output by approximating the relation between input and output with a line—or with a higher-dimensional *linear* shape. In this book, linear regression is the first concrete example of *supervised learning*, in Chapter 2, Your First Learning Program, on page 15. Later on, in Chapter 4, Hyperspace!, on page 45, I introduce higher-dimensional *multiple linear regression*.

Linearly separable

A dataset is linearly separable if it can be partitioned with a "straight" shape. For example, imagine a plane populated with two different kinds of data: circles and triangles. If the data are linearly separable, then you

can trace a straight line that has all the circles on one side, and all the triangles on the other. Check out Tracing a Boundary, on page 149 for visual examples.

Local gradient

In the context of the *chain rule*, the local gradient is the gradient of an operation's output with respect to its input. I use this concept to explain *backpropagation* in From the Chain Rule to Backpropagation, on page 131.

Local minimum

See *global minimum*. I talk about both kinds of minima there.

Log loss

One of the formulae that we use to calculate the *loss*. It works well for *binary classification* problems. See Smoothing It Out, on page 69.

Logistic function

A more technically accurate name for the *sigmoid*.

Logistic regression

Logistic regression is a statistics technique to model a binary variable. I don't really use the name "logistic regression" in the main text of this book, except in passing—but this technique is the theoretical underpinning for the binary classifier in Chapter 5, A Discerning Machine, on page 63.

Mgosoft PDF Split Merge

Logits

The inputs of a *softmax*.

Loss

A function that measures the error in a machine learning system—in other words, a number that measures how bad the system's forecast is. The *training phase* of a machine learning system is all about minimizing the loss, by tweaking the *weights* with techniques such as *gradient descent*. In this book, we use different formulae to calculate the loss, including the *mean squared error* and the *log loss*.

Matrix

A bi-dimensional array, like a grid.

Matrix multiplication

A frequent operation in ML. It operates on two *matrices* of shapes `(a, b)` and `(b, c)`, and results in a third matrix of shape `(a, c)`. Check out the details in Multiplying Matrices, on page 49.

Matrix transpose

An operation that flips a *matrix* over its diagonal. For an example, see Transposing Matrices, on page 51.

Mean squared error

One of the common formulae used to calculate the *loss*. I introduce it in How Wrong Are We?, on page 22.

Mini-batch gradient descent

A variation of *gradient descent* where each step calculates the gradient on a subset of the training examples, rather than all of them. For example, you might have 10,000 training examples, but only feed them to GD in batches of 50. The exact number of examples for each mini-batch is a *hyperparameter* of the system. I introduce batch GD in Batch by Batch, on page 162.

Model function

Supervised learning works by approximating data with a function. That function is also called the "model." For example, in Chapter 2, Your First Learning Program, on page 15, we use a line as our model.

Model selection

Mgosoft PDF Split Merge

The process of choosing an algorithm, and hence a *model function*, for a ML system. For example, you might compare the results of *linear regression*, a *neural network*, and some other algorithm, and pick the algorithm that gives the most accurate predictions.

Momentum

In the context of machine learning, momentum (or "gradient descent with momentum") is a variant of plain vanilla *gradient descent*. See Gradient Descent on Steroids, on page 240.

Multilayer perceptron

An alternative (and somewhat old-fashioned) name for a *neural network*.

Multiclass classifier

Also called *multinomial classifiers*, multiclass classifiers assign data to many classes, as opposed to *binary classifiers*. For example, if you're deciding which brand a car belongs to, then you're doing multiclass classification—unless you only have one or two brands, that is. Chapter 7, The Final Challenge, on page 87 is all about multiclass classification.

Multinomial classifier

Another name for a *multiclass classifier*.

Multiple linear regression

Linear regression on a function that takes more than one input variable. Chapter 4, Hyperspace!, on page 45 is dedicated to multiple linear regression.

Neural network

Neural networks are a popular architecture for *supervised learning*, and the main characters in this book. We introduce them in Chaining Perceptrons, on page 113, and use them throughout the rest of the book. Neural networks are usually defined as "shallow" when they consist of three *layers*, and "deep" when they have more.

Neuron

Sometimes, people use the word "neuron" to indicate a *node* in a neural network. More precisely, the neuron is the component that includes a *node*, its preceding nodes, and the operations in between—a *weighted sum* and an *activation function*. Check out Chaining Perceptrons, on page 113 for a picture.

Node

The values inside a neural network, that are arranged in *layers*. See Chaining Perceptrons, on page 113 for a picture.

Mgosoft PDF Split Merge

Nonlinear

A function is nonlinear when it cannot be plotted as a straight shape. When we talk about nonlinearity in *neural networks*, that's usually in the context of *activation functions*.

Normalization

In general, you normalize something when you scale its value to be between 0 and 1. For example, the *softmax* function normalizes a bunch of variables so that each variable is in that range, and their sum equals 1. However, sometimes the term "normalization" is used in place of the term "standardization," that is more specific. For example, the technique known as *batch normalization* should arguably be called batch *standardization*.

Numerical stability

Technically, a computation is "numerically unstable" when it tends to amplify small fluctuations in its inputs. In concrete, that means that the computation tends to generate values that are very large or very small, and those values can cause an overflow or an underflow. For example, the *softmax* that we code in this book is numerically unstable because it

can easily generate huge values that are too large for Python to handle. For more, see Numerical Stability, on page 124.

One-hot encoding

An encoding technique that converts a set of *categorical* values to a set of arrays, each containg a single 1 and a bunch of 0s. For example, the array `[1, 2, 3]` could be one-hot encoded as `[[1, 0, 0], [0, 1, 0], [0, 0, 1]]`. One-hot encoding is useful to encode the labels in certain machine learning problems, for reasons I discuss in One-Hot Encoding, on page 89.

Outlier

In a dataset, an outlier is a data point that's significantly different from its siblings. For example, imagine a dataset that shows a programmer's income and the number of hours she worked. On most months, there is a strict correlation between the two—except on one lucky month, when the programmer happened to win a few thousands bucks at the lottery while she was on vacation. That month would be an outlier.

Output layer

The last *layer* in a *neural network*. See Chaining Perceptrons, on page 113.

Overfitting

Mgosoft PDF Split Merge

A machine learning system is "overfitting" when it's more accurate on the training data than it is on new, unknown data. For example, a face-recognition system might become good at recognizing faces in the pictures that it's been trained on, and then fail when confronted with new pictures of the same people. Overfitting happens because the system "knows" the training data, so it ends up memorizing the accidental details of that data instead of generalizing out of it. To use a metaphor, the system is acting like a student who memorizes a concept by rote learning, without really understanding it. Chapter 17, Defeating Overfitting, on page 211 is entirely dedicated to overfitting.

Partial derivative

Take a function with more than one variable, such as `a * b`. The partial derivative is the derivative of that function with respect to only one variable—either `a` or `b`. For more details, and an intuitive description of this concept, see Partial Derivatives, on page 38.

Perceptron

Historically, the perceptron was one of the first working ML systems. It's also the base for *multilayer perceptrons*—that is, *neural networks*. Chapter 8, The Perceptron, on page 99 is dedicated to the perceptron.

Prediction phase

The prediction phase (as opposed to the *training phase*) is the phase of *supervised learning* where you get your money back—the time when you ask the system to make an inference from *unlabeled data*. For example, during a speech recognizer's prediction phase, the system could take a sound and "predict" the vocal command in it. I talk about the two phases of supervised learning in Supervised Learning, on page 6.

Recurrent Neural Networks

RNNs are one of the most important types of *neural network* architectures today, especially in the field of natural language processing. I don't talk about RNNs in this book, but I mention their basic principles at the end of the book, in The Path of Language, on page 268.

Recursion

See *recursion*.

Reinforcement learning

A style of machine learning where the algorithm learns by trial and error. I talk briefly about reinforcement learning in Programming vs. Machine Learning, on page 4.

Mgosoft PDF Split Merge

Regularization

A family of approaches to reduce *overfitting*. More precisely, "regularization" is any change that you make to a system like a *neural network* to reduce the distance between its performance on the *training set* and its performance on the *validation set*. For the details, read Regularizing the Model, on page 217 and A Regularization Toolbox, on page 224.

ReLU

One of the most popular *activation functions*, described in Enter the ReLU, on page 235. Also comes in a few variants, like the *Leaky ReLU* and the *softplus*.

RMSprop

A variant of *gradient descent*. See Gradient Descent on Steroids, on page 240.

Saturation

An *activation function* saturates when it receives an input that's outside its "ideal" range, which pushes its *gradient* close to zero. The best way to understand this concept is to see a concrete example of it. Read Dead Neurons, on page 141 for an example of a saturating *sigmoid*.

Scalar

A single number you might represent in your code with an integer or floating point value. By contrast, a value that's composed of multiple scalars is called a "vector"—but in programming, it's more likely to be called an "array."

Sigmoid

The sigmoid is a function that takes any number and converts it to the range from 0 to 1, excluded. It's often used as an *activation function*. I introduce it in Invasion of the Sigmoids, on page 66.

Softmax

The softmax is a function that takes an array of numerical inputs (called *logits*) and returns an array of the same size, where the inputs have been rescaled to the range from 0 to 1, excluded. It's often used as the last *activation function* in a *neural network*. I introduce it in Enter the Softmax, on page 116.

Softplus

See *ReLU*.

Standard deviation

Mgosoft PDF Split Merge

A measure of how "spread out" a variable is. A variable with a low standard deviation rarely strays too far from its average value. One example from Chapter 15, Let's Do Development, on page 177 is the height of humans compared to the height of plants. The second has higher standard deviation than the first, because plants have wildly different heights, while humans don't: nobody is hundreds of times taller than anyone else.

Standard normal distribution

A distribution of data with a mean of 0 and a standard deviation of 1. I briefly mention it in Weight Initialization Done Right, on page 143.

Standardization

A technique to rescale data before feeding it to a machine learning system. In the strictest sense, you standardize a bunch of input *features* when you rescale them so that their mean is zero, and their *standard deviation* is 1. However, the term is used often to mean slightly different techniques. For an example, read Preparing Data, on page 178.

Statistical noise

Here is an example to explain what statistical noise is: every time you weight yourself, you're likely to get a different weight. Some of these differences might be meaningful, and they depend on the fact that you're

gaining or losing weight. Other differences might be due to irrelevant factors: you weighted yourself at different times during the day, your scale doesn't have perfect accuracy, and so on. These irrelevant fluctuations in real-world data are called "statistical noise." We talk about this concept in our discussion of *overfitting*, in The Causes of Overfitting, on page 212.

Stochastic gradient descent

For most people, "stochastic gradient descent" means "*mini-batch gradient descent* with a batch size of 1." In chapters such as Chapter 13, Batchin' Up, on page 159, I refer to that meaning. However, some ML practitioners (and some libraries, including Keras) use "stochastic gradient descent" as a synonym to "mini-batch gradient descent," regardless of the batch size. We'll have to live with the confusion, and hope that one of those conflicting meanings fades with time.

Strictly increasing/decreasing

Intuitively, a function is strictly increasing if it always points "upward," and strictly decreasing if it always points "downward." More formally, as the input grows, the value of a strictly increasing function always grows, and the value of a strictly decreasing function always drops. For example, if I say "the amount of food we consume increases with the number of dogs we own," that means that once we get a new dog, we'll need the same or more food—never less food. If we say that "the amount of food *strictly* increases with the number of dogs," that means we'll need more food—never the same or less food.

Mgosoft PDF Split Merge

Supervised learning

A style of machine learning where a system takes a set of *examples* in the form of *input variables* and *labels* and learns the relation between the input variable and the label. For example, the input variable could be the number of reservations at a restaurant, and the label could be the number of pizzas sold; or, the input variable could be an X-ray scan, and the label could be a boolean value that tells whether the scan shows pneumonia. This style of ML is called "supervised" because somebody, supposedly a human, took care of preparing and labeling the examples.

Swish

See *ReLU*.

Tensor

A multidimensional array. A tensor with one dimension is usually called a "vector," and a tensor with two dimensions is usually called a *matrix*—so

people tend to reserve the name "tensor" for arrays with three or more dimensions.

Test set

The subset of *examples* you use to test the system. In general, you shouldn't use the same examples to test and train the system, for reasons I explain in Training vs. Testing, on page 79.

Training set

The subset of examples you use to train the system. In general, you shouldn't reuse them to test the system as well, for reasons I explain in Training vs. Testing, on page 79.

Training phase

The training phase (as opposed to the *prediction phase*) is the phase of *supervised learning* where the system goes through the *examples* and learns the relation between the *input variables* and the *label*. For example, during a speech recognizer's training phase, the system could sort through a large number of sound clips, each labeled with the vocal command in the clip. I talk about the two phases of machine learning in Supervised Learning, on page 6.

Mgosoft PDF Split Merge

Translation invariance

A property of *convolutional neural networks* that allows them to identify shapes wherever they are inside an image. See CNNs' Secret Sauce, on page 253.

Tuple

A Python type that is similar to a list, but immutable. See Collections, on page 280.

Underfitting

Simply put, a machine learning system is "underfitting" when it's not powerful enough to make accurate predictions—either because of its architecture, or because its hyperparameters haven't been tuned correctly. For example, imagine trying to predict fluctuations in ice cream sales with linear regression. Ice cream sales are mostly periodical—higher in summer and lower in winter—so it's hard to approximate them with a straight line. If you try, then your predictions aren't likely to be very good in most cases. That's a case of underfitting. Contrast that concept with the concept of *overfitting*.

Unsupervised learning

A style of machine learning where the machine finds patterns in unlabeled data. I talk about it in What About Unsupervised Learning?, on page 11.

Validation set

The set of examples that is used to choose a system's model and hyperparameters during the development process. (Also see *model selection*.) The validation set should be different from the *test set*, that is only used at the very end of the process. The reason for that difference is that you don't want your system to *overfit* the test data. For a more detailed explanation, see A Testing Conundrum, on page 173.

Vanishing gradient

A problem that plagued deep *neural networks* for years. The vanishing gradient slows down training to a crawl as you increase the number of *layers* in a network. It was partially solved by switching from *sigmoid*-like *activation functions* to other functions such as the *ReLU*. For the nitty gritties, see The Vanishing Gradient, on page 233.

Variance

In machine learning, you can say that a system has "high variance" when it *overfits* the training data, as I mention in Bias and Variance, on page 217.

Mgosoft PDF Split Merge

Weight

One of the learnable *parameters* in a *perceptron*, or a *neural network*.

Weighted sum

An operation that consists in adding together a number of elements, each multiplied by a *weight*: `element1 * weight1 + element2 * weight2 + ...`

Xavier initialization

Xavier (or "Glorot") initialization is a method to initialize the *weights* in a *neural network*. It helps reduce problems such as *dead neurons* and *vanishing* or *exploding gradients*. For the details, see Better Weight Initialization, on page 239.