Contents

Machine Learning concepts 4
- Learning Algorithm 4
- Predictive Model (Model) 4
- Model, Classification 4
- Model, Regression 4
- Representation Learning 4
- Supervised Learning 4
- Unsupervised Learning 4
- Semi-Supervised Learning 5
- Parameter 5
- Population 5

Algorithms 5
- Linear Regression 5
- Principal Component Analysis (PCA) 5
- K-Means 6
- Support Vector Machine (SVM) 7
- Transfer Learning 7
- Decision Tree 7
- Dimensionality Reduction 8
- Instance based learning 8
  - Instance-Based Learning 8
  - K Nearest Neighbors 8
  - Kernel 9

Training 9
- Basics 9
  - Training 9
  - Training Example 9
  - Training Set 9
  - Iteration 9
  - Convergence 9
- Data 10

Standardization 10
Holdout Set 10
Normalization 10
One-Hot Encoding 10
Outlier 11
Embedding 11
Regression 12
Regression Algorithm 12
Regression Model 12
Classification 12
Class 12
Hyperplane 12
Decision Boundary 12
False Negative (FN) 13
False Positive (FP) 13
True Negative (TN) 13
True Positive (TP) 13
Precision 13
Recall 14
F1 Score 14
Few-Shot Learning 14
Hinge Loss 14
Log Loss 14
Ensemble 15
Ensemble Learning 15
Strong Classifier 15
Weak Classifier 15
Boosting 15
Evaluation 15
Validation Example 15
Validation Loss 15
Validation Set 16
Variance 16

Cost Function 16
Cross-Validation 16
Overfitting 16
Regularization 16
Underfitting 16
Evaluation Metrics 17
Evaluation Metric 17
Regression metrics 17
Mean Absolute Error. 17
Mean Squared Error. 17
R^2 17
Classification metrics 17
Accuracy. 17
Logarithmic Loss. 17
Area Under ROC Curve. 17
Confusion Matrix. 17
Hyperparameter 18
Hyperparameter 18
Hyperparameter Tuning 18
Grid Search 18
Random Search 18

Source

This Glossary is a subset from the original source for the use in book learn machinelearning coding basics in a weekend

The glossary is adapted from the source Glossary of Machine Learning Terms (reproduced and adapted with permission of semantic.ca).

Where other sources are referenced, they are noted in the relevant sections.

Machine Learning concepts

Learning Algorithm

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

Predictive Model (Model)

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

Model, Classification

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

Model, Regression

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

Representation Learning

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

Supervised Learning

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

Unsupervised Learning

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

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

Semi-Supervised Learning

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

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

Parameter

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

Population

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

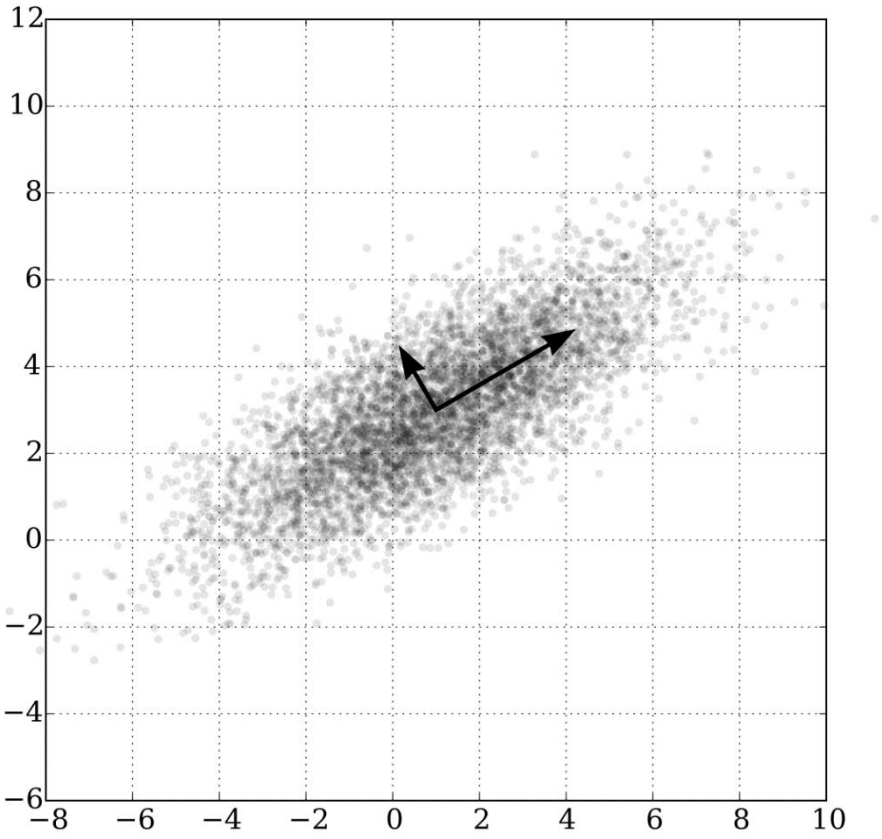
Algorithms

Linear Regression

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

Principal Component Analysis (PCA)

Principal component analysis, or PCA, is a linear transformation which projects n examples, each consisting of m features on a hyperplane in such a way that the projection error is minimal.
The principal components are random variables of maximal variance constructed from linear combinations of the input features. Equivalently, they are the projections onto the principal component axes, which are lines that minimize the average squared distance to each example in the data set. To ensure uniqueness, all of the principal component axes must be orthogonal.
Below, for a two-dimensional dataset, its two principal components are shown as arrows:



Principal components. Source: Wikipedia.

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

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

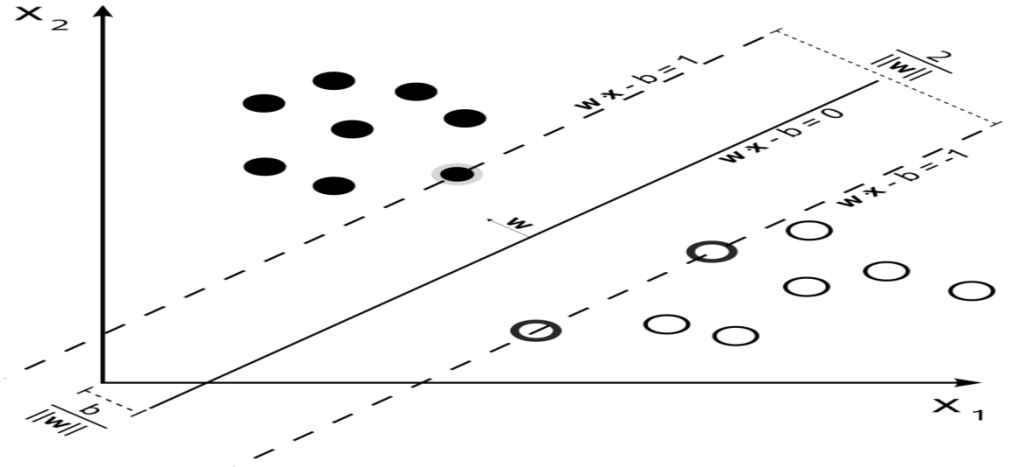
K-Means

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

Support Vector Machine (SVM)

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

For two-dimensional *feature vectors*, the problem and the solution can be visualized as a plot below (taken from Wikipedia):



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

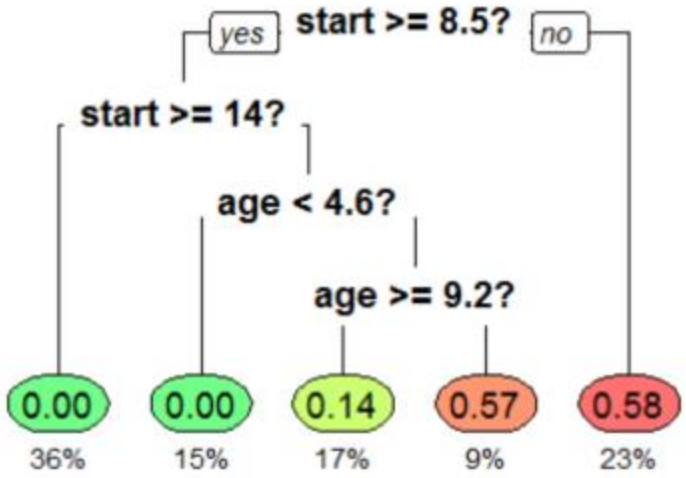
Transfer Learning

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

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

Decision Tree

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



A decision tree. Source: Wikipedia.

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

Dimensionality Reduction

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

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

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

Instance based learning

Instance-Based Learning

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

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

K Nearest Neighbors

K Nearest Neighbors (also often abbreviated as kNN) is an *instance-based learning algorithm* that can be used for both classification and regression. When used in the *classification* context, the algorithm predicts the class of an *unlabeled example* as the majority class among k its closest

neighbors in the vector space. In the *regression* context, the label of an unlabeled example is calculated as an average of the labels of the k its closest neighbors. The distance from one example to another is usually given by a *similarity metric*.

Kernel

Kernel methods can be thought of as *instance-based learners*: rather than learning some fixed set of *parameters* corresponding to the *features* of their inputs, they instead "remember" the i-th training example (x→i,yi) and learn for it a corresponding weight wi. Prediction for unlabeled inputs, i.e., those not in the training set, is treated by the application of a similarity function k, called a *kernel*, between the unlabeled input x'→ and each of the training inputs →x→i.

Training

Basics

Training

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

Training Example

A *training example* is a member of the *training set*. The *training set* is a holdout set used for final *model* assessment

Training Set

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

Iteration

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

Convergence

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

Data

Standardization

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

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

Holdout Set

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

Normalization

Normalization is the process of converting an actual range of values into a standard range of values, typically in the interval [−1,+1][−1,+1] or [0,1][0,1].
For example, suppose the natural range of a certain *feature* is 350 to 1,450. By subtracting 350 from every value of the feature, and dividing the result by 1,100, one can normalize those values into the range [0,1][0,1].

One-Hot Encoding

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

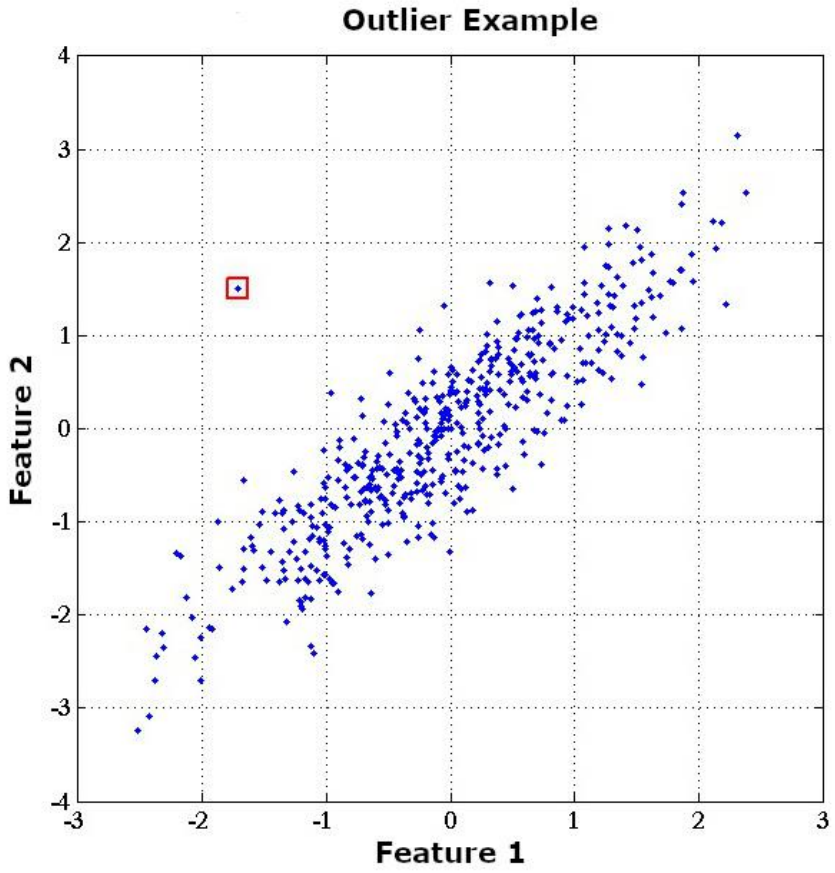
$$sun = [1, 0, 0]$$

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

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

Outlier

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



An outlier.

Embedding

An *embedding* maps an input representation, such as a word or sentence, into a vector. Most frequently embeddings refer to word embeddings such as **word2vec** or **GloVe**. However, sentences, paragraphs or images can be embedded too. For example, by mapping images and

their textual descriptions into a common embedding space and minimizing the distance between them, one can match labels with images. Embeddings may be learned using *neural networks*.

Regression

Regression Algorithm

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

Regression Model

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

Classification

Class

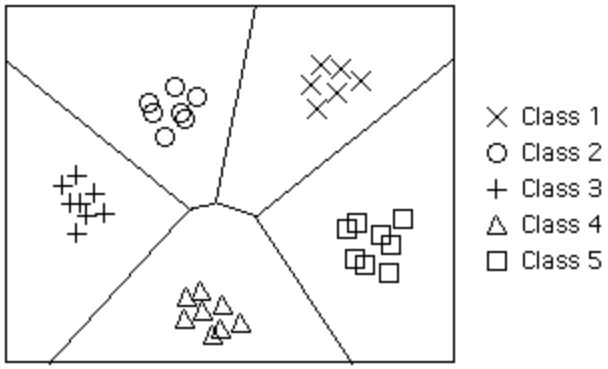
A *class* is a group to which an *example* can belong. A labeled example consists of a *feature vector* and a reference to the class it belongs to. A specific class attached to a feature vector is called *label*. For example, in a binary classification model that detects spam, there are two classes,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

Hyperplane

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

Decision Boundary

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



A decision boundary. Source: princeton.edu.

False Negative (FN)

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

False Positive (FP)

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

True Negative (TN)

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

True Positive (TP)

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

Precision

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

$$precision = \frac{TP}{(TP + FP)},$$

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

Recall

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

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

where TPTP is the number of *true positives* and FNFN is the number of *false negatives*.
See also: *precision* and *confusion matrix*.

F1 Score

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

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}}$$

Few-Shot Learning

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

Hinge Loss

The *hinge loss* is a *loss function* used for training classifiers. The hinge loss is used for "maximum-margin" classification, most notably for *support vector machines* (SVMs).

Log Loss

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

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

where pp is the probability predicted by the model that the label is y and y is the true label.

Ensemble

Ensemble Learning

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

Strong Classifier

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

Weak Classifier

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

Boosting

Boosting is an *ensemble learning* technique that iteratively combines a set of simple and not very accurate classifiers (referred to as *weak classifiers*) into a classifier with high *accuracy* (called a *strong classifier*) by giving a higher weight to the examples that the model is currently classifying incorrectly.
Multiple boosting algorithms exist. The most widely used ones are **AdaBoost** and **gradient boosting**.

Evaluation

Validation Example

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

Validation Loss

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

Validation Set

The *validation set* is a holdout set used for *hyperparameter tuning*. *Validation loss* is the average loss computed by applying the *loss function* the outputs of the model applied to the examples from the validation set.

Variance

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

Cost Function

A cost function(also called a loss function) represents a "cost" associated with the event. An optimization problem seeks to minimize a loss function.

Cross-Validation

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

Overfitting

Overfitting occurs when the machine learning algorithm learns a model that fits the training data too well by incorporating details and noise specific to the training data. Overfitting is characterized by the nearly perfect prediction by the model of the labels of the *training examples* and poor prediction of the labels of examples from the validation set.
The problem of overfitting is usually solved by *regularization* or *early stopping*,

Regularization

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

Underfitting

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

Evaluation Metrics

Evaluation Metric

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

Regression metrics

Mean Absolute Error.

Mean Squared Error.

R^2

Classification metrics

Accuracy

Classification accuracy is the number of correct predictions made as a ratio of all predictions made. This is the most common evaluation metric for classification problems, it is also the most misused. It is really only suitable when there are an equal number of observations in each class (which is rarely the case) and that all predictions and prediction errors are equally important, which is often not the case.

Source: Machine Learning Mastery

Logarithmic Loss.

Logarithmic loss (or logloss) is a performance metric for evaluating the predictions of probabilities of membership to a given class. The scalar probability between 0 and 1 can be seen as a measure of confidence for a prediction by an algorithm. Predictions that are correct or incorrect are rewarded or punished proportionally to the confidence of the prediction.

Source: Machine Learning Mastery

Area Under ROC Curve.

The AUC represents a model's ability to discriminate between positive and negative classes. An area of 1.0 represents a model that made all predictions perfectly. An area of 0.5 represents a model as good as random. Learn more about ROC here.

Confusion Matrix.

The confusion matrix is a handy presentation of the accuracy of a model with two or more classes.

The table presents predictions on the x-axis and accuracy outcomes on the y-axis. The cells of the table are the number of predictions made by a machine learning algorithm.

For example, a machine learning algorithm can predict 0 or 1 and each prediction may actually have been a 0 or 1. Predictions for 0 that were actually 0 appear in the cell for prediction=0 and actual=0, whereas predictions for 0 that were actually 1 appear in the cell for prediction = 0 and actual=1. And so on.

You can learn more about the Confusion Matrix on the Wikipedia article.

Source: Source: Machine Learning Mastery

Hyperparameter

Hyperparameter

Hyperparameter is a parameter of a *machine learning algorithm* whose value is not optimized during training. Depending on the algorithm, a hyperparameter could be the number of training *iterations*, the size of the *minibatch*, a *regularization* parameter, the value of the *learning rate*, and many others. Since hyperparameters are not optimized by the machine learning algorithm itself, it is often optimized using *cross-validation* and techniques like *grid search*, *random search*, *Bayesian optimization*, *evolutionary optimization*, and others.

Hyperparameter Tuning

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

Grid Search

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

Random Search

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