

# Fundamentals of Machine Learning

## 1 Four Branches of Machine Learning

### 1.1 Supervised Learning

It consists of:

- learning to map input data to known targets, given a set of examples (often annotated by humans);
- classification and regression;
- exotic variants:
  - *Sequence generation* - Given a picture, predict a caption describing it. Sequence generation can sometimes be reformulated as a series of classification problems (such as repeatedly predicting a word or token in a sequence).
  - *Syntax tree prediction* - Given a sentence, predict its decomposition into a syntax tree.
  - *Object detection* - Given a picture, draw a bounding box around certain objects inside the picture. This can also be expressed as a classification problem (given many candidate bounding boxes, classify the contents of each one) or as a joint classification and regression problem, where the bounding-box coordinates are predicted via vector regression.
  - *Image segmentation* - Given a picture, draw a pixel-level mask on a specific object.

### 1.2 Unsupervised Learning

It consists of finding interesting transformations of the input data without the help of any targets, for the purposes of data visualization, data compression, or data denoising, or to better understand the correlations present in the data at hand.

Example: *Dimensionality reduction* and *clustering*.

### 1.3 Self-supervised Learning

This is a specific instance of supervised learning. Self-supervised learning is supervised learning without human-annotated labels. There are still labels involved (because the learning has to be supervised by something), but they are generated from the input data, typically using a heuristic algorithm.

Example: *Autoencoders*.

### 1.4 Reinforcement Learning

An *agent* receives information about its environment and learns to choose actions that will maximize some reward.

Example: A neural network that “looks” at a video-game screen and outputs game actions in order to maximize its score can be trained via reinforcement learning.

## Classification and Regression Glossary

- Sample or input - One data point that goes into your model.
- Prediction or output - What comes out of your model.
- Target - The truth. What your model should ideally have predicted, according to an external source of data.
- Prediction error or loss value - A measure of the distance between your model's prediction and the target.
- Classes - A set of possible labels to choose from in a classification problem. For example, when classifying cat and dog pictures, "dog" and "cat" are the two classes.
- Label - A specific instance of a class annotation in a classification problem. For instance, if picture #1234 is annotated as containing the class "dog", then "dog" is a label of picture #1234.
- Ground-truth or annotations - All targets for a dataset, typically collected by humans.
- Binary classification - A classification task where each input sample should be categorized into two exclusive categories.
- Multiclass classification - A classification task where each input sample should be categorized into more than two categories: for instance, classifying hand-written digits.
- Multilabel classification - A classification task where each input sample can be assigned multiple labels. For instance, a given image may contain both a cat and a dog and should be annotated both with the "cat" label and the "dog" label. The number of labels per image is usually variable.
- Scalar regression - A task where the target is a continuous scalar value. Predicting house prices is a good example: the different target prices form a continuous space.
- Vector regression - A task where the target is a set of continuous values: for example, a continuous vector. If you are doing regression against multiple values (such as the coordinates of a bounding box in an image), then you are doing vector regression.
- Mini-batch or batch - A small set of samples (typically between 8 and 128) that are processed simultaneously by the model. The number of samples is often a power of 2, to facilitate memory allocation on GPU. When training, a mini-batch is used to compute a single gradient-descent update applied to the weights of the model.

## 2 Evaluating Machine-Learning Models

### 2.1 Training, Validation, and Test Sets

#### Why We Need Validation Sets?

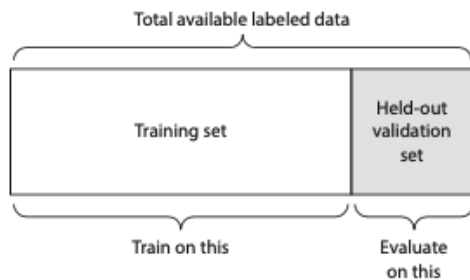
Developing a model always involves tuning its configuration: for example, choosing the number of layers or the size of the layers (called the hyperparameters of the model, to distinguish them from the parameters, which are the network's weights).

Tuning the configuration of the model based on its performance on the validation set can quickly result in overfitting to the validation set. Central to this phenomenon is the notion of *information leaks*. Every time we tune a hyperparameter of your model based on the model's performance on the validation set, some information about the validation data leaks into the model.

We will end up with a model that performs artificially well on the validation data. We care about performance on completely new data, not the validation data, so we need to use a completely different, never-before-seen dataset to evaluate the model: the test dataset. Our model shouldn't have had access to any information about the test set, even indirectly.

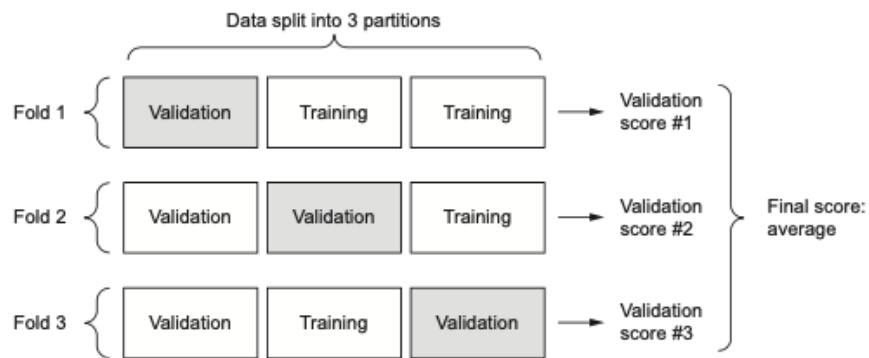
#### Three Evaluation Methods

- Simple Hold-Out Validation



**Figure 4.1** Simple hold-out validation split

- K-Fold Validation



**Figure 4.2** Three-fold validation

- Iterated K-Fold Validation Shuffling (for little data available): applying K-fold validation multiple times, shuffling the data every time before splitting it K ways. The final score is the average of the scores obtained at each run of K-fold validation. Note that we end up training and evaluating  $P \times K$  models (where P is the number of iterations we use), which can be very expensive.

## Data Shuffling: Link


### Random Shuffling

Let's take a first simple example where we want to shuffle the following dataset.

name	phone	country
赵	(86) 156 1xxx xxxx	China
Martin	(33) 06 10 xx xx xx	France
钱	(86) 156 2xxx xxxx	China
John	(44) 020 xxxx xxxx	UK
孙	(86) 156 3xxx xxxx	China
李	(86) 156 4xxx xxxx	China
陈	(86) 156 5xxx xxxx	China
Éric	(33) 06 20 xx xx xx	France
Marc	(33) 06 30 xx xx xx	France
Anne	(33) 07 40 xx xx xx	France
Sophie	(33) 07 50 xx xx xx	France
Lily	(44) 0113 xxxx xxxx	UK
Lucy	(44) 0121 xxxx xxxx	UK
Gary	(44) 029 xxxx xxxx	UK

By applying a random shuffling algorithm, we can get:

name	phone	country
赵	(86) 156 1xxx xxxx	China
Martin	(33) 06 10 xx xx xx	France
钱	(86) 156 2xxx xxxx	China
John	(44) 020 xxxx xxxx	UK
孙	(86) 156 3xxx xxxx	China
李	(86) 156 4xxx xxxx	China
陈	(86) 156 5xxx xxxx	China
Éric	(33) 06 20 xx xx xx	France
Marc	(33) 06 30 xx xx xx	France
Anne	(33) 07 40 xx xx xx	France
Sophie	(33) 07 50 xx xx xx	France
Lily	(44) 0113 xxxx xxxx	UK
Lucy	(44) 0121 xxxx xxxx	UK
Gary	(44) 029 xxxx xxxx	UK



name	phone	country
赵	(33) 06 30 xx xx xx	UK
Martin	(44) 029 xxxx xxxx	China
钱	(86) 156 3xxx xxxx	France
John	(33) 07 40 xx xx xx	China
孙	(86) 156 1xxx xxxx	UK
李	(86) 156 4xxx xxxx	UK
陈	(33) 07 50 xx xx xx	UK
Éric	(33) 06 10 xx xx xx	France
Marc	(86) 156 5xxx xxxx	France
Anne	(44) 0113 xxxx xxxx	France
Sophie	(86) 156 2xxx xxxx	France
Lily	(33) 06 20 xx xx xx	China
Lucy	(44) 0121 xxxx xxxx	China
Gary	(44) 020 xxxx xxxx	China

## Things to Keep in Mind

- Data representativeness - We want both our training set and test set to be representative of the data at hand.
- The arrow of time - If we are trying to predict the future given the past, we should not randomly shuffle our data before splitting it.
- Redundancy in our data - If some data points in our data appear twice, then shuffling the data and splitting it into a training set and a validation set will result in redundancy between the training and validation sets.

## 3 Data Preprocessing, Feature Engineering

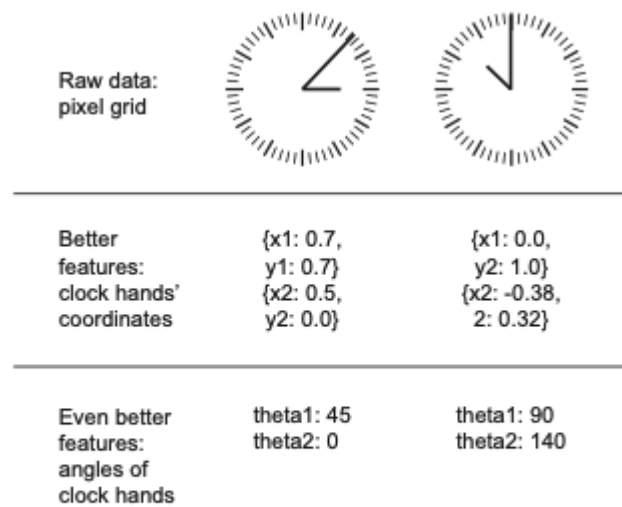
### Data Preprocessing

- Vectorization - All inputs and targets in a neural network must be tensors of floating-point data.
- Normalization - The data should take small values and be homogeneous.
- Handling missing values.

### Feature Engineering

Feature engineering is the process of using our own knowledge about the data and about the machine-learning algorithm at hand (in this case, a neural network) to make the algorithm work better by applying hardcoded (nonlearned) transformations to the data before it goes into the model.

- Good features still allow us to solve problems more elegantly while using fewer resources.
- Good features let you solve a problem with far less data.



**Figure 4.3** Feature engineering for reading the time on a clock

## 4 Overfitting and Underfitting

The fundamental issue in machine learning is the tension between optimization and generalization. Optimization refers to the process of adjusting a model to get the best performance possible on the training data (the learning in machine learning), whereas generalization refers to how well the trained model performs on data it has never seen before.

### 4.1 Fighting Overfitting: Regularization

- Reducing the network's size.
- Adding weight regularization.
- Adding dropout.

## 5 Summary: The Universal Workflow of Machine Learning

### 5.1 Defining the Problem and Assembling a Dataset

Defining the Problem:

- What will our input data be? What are we trying to predict?
- What type of problem are we facing? Binary classification? Multiclass classification? Scalar regression? Vector regression? Multiclass, multilabel classification? Clustering, generation, or reinforcement learning?

## Hypotheses We Make:

- Outputs can be predicted given inputs.
- Available data is sufficiently informative to learn the relationship between inputs and outputs.

## 5.2 Choosing a Measure of Success

Accuracy? Precision and recall?

- Balanced-classification problems: accuracy and area under the receiver operating characteristic curve (ROC AUC).
- Class-imbalanced problems: precision and recall.
- Ranking problems or multilabel classification: mean average precision.

## 5.3 Deciding on an Evaluation Protocol

- Simple Hold-Out Validation - The way to go when you have plenty of data.
- K-Fold Validation - The right choice when you have too few samples for hold-out validation to be reliable.
- Iterated K-Fold Validation Shuffling - For performing highly accurate model evaluation when little data is available.

## 5.4 Preparing Data

- Vectorization
- Normalization
- Feature engineering

## 5.5 Developing a Model that Does Better Than a Baseline

The goal at this stage is to achieve *statistical power*: that is, to develop a small model that is capable of beating a dumb baseline.

Three key choices to build our first working model:

- Last-layer activation - This establishes useful constraints on the network's output.
- Loss function - This should match the type of problem we are trying to solve.
- Optimization configuration - What optimizer will we use? What will its learning rate be? In most cases, it's safe to go with *rmsprop* and its default learning rate.



Problem type	Last-layer activation	Loss function
Binary classification	sigmoid	binary_crossentropy
Multiclass, single-label classification	softmax	categorical_crossentropy
Multiclass, multilabel classification	sigmoid	binary_crossentropy
Regression to arbitrary values	None	mse
Regression to values between 0 and 1	sigmoid	mse or binary_crossentropy

## 5.6 Scaling up: Developing a Model that Overfits

Once we have obtained a model that has statistical power, the question becomes, is our model sufficiently powerful? Does it have enough layers and parameters to properly model the problem at hand?

To figure out how big a model we will need, we must develop a model that overfits.

- Add layers.
- Make the layers bigger.
- Train for more epochs.

When we see that the model's performance on the validation data begins to degrade, we have achieved overfitting.

## 5.7 Regularizing the Model and Tuning the Hyperparameters

- Add dropout.
- Try different architectures: add or remove layers.
- Add L1 and/or L2 regularization.
- Try different hyperparameters to find the optimal configuration.
- Optionally, iterate on feature engineering: add new features, or remove features that don't seem to be informative.