

Types of machine learning

Ibrahim A. Hameed, PhD, Professor

03.02.2021

Four branches of machine learning

- **Supervised learning:**
 - This is by far the most common case.
 - It consists of learning to map input data to known targets, given a set of examples.
 - Used for classification and regression
- **Unsupervised learning**
 - This branch of machine learning 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**.
 - Unsupervised learning is the bread and butter of data analytics, and it's often a **necessary step in better understanding a dataset** before attempting to solve a supervised-learning problem.
 - ***Dimensionality reduction* and *clustering* are well-known categories of unsupervised learning.**

Four branches of machine learning

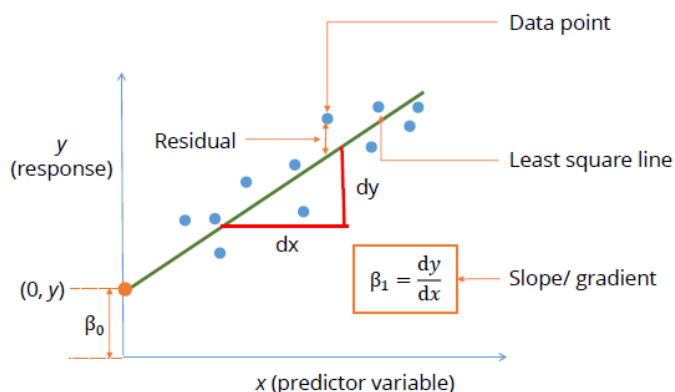
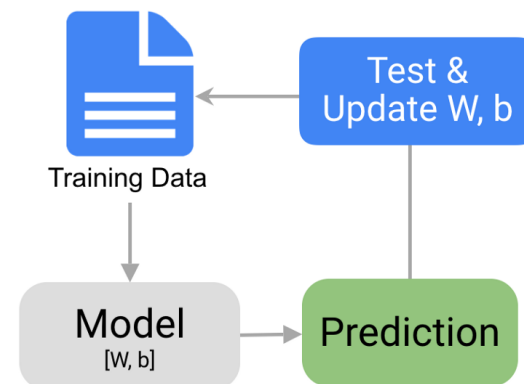
- **Self-supervised learning**
 - Self-supervised learning is supervised learning without human-annotated labels — you can think of it as supervised learning without any humans in the loop.
 - There are still labels involved, but they're generated from the input data, typically using a heuristic algorithm.
 - For example, predicting the next frame in a video, given past frames, or the next word in a text, given previous words.
- **Reinforcement learning**
 - In reinforcement learning, an *agent* receives information about its environment and learns to choose actions that will maximize some reward.
 - For instance, 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.

Types of machine learning problems

- **Binary classification** — is supervised learning since training datasets is labelled. Classification task where each input sample has only two classes (1-0 or Yes - No). Some examples include:
 - Credit Card Fraud Detection
 - Medical Diagnosis
 - Spam Detection
- **Multiclass classification** — A classification task where each input sample should be categorized into more than two categories/classes. As an example:
 - classify a set of images of fruits which may be oranges, apples, or pears.
 - Classifying handwritten 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. For example: Predicting house prices.

Model development: universal workflow

- Define your problem
- Collect data
- Choose a measure of success
- Set an evaluation protocol
- Prepare your data
- Develop a benchmark/baseline model
- Develop a better model & tuning its hyperparameters
- Document your results ... deploy your model



$$y = \beta_0 + \beta_1 x + u$$

The equation is annotated to show its components: y is the 'Actual value', $\beta_0 + \beta_1 x$ is the 'Predicted value', and u is the 'Residual'.

How to appropriately define your problem?

- The following questions must be answered:
 - What is the main objective? What are we trying to predict?
 - What are the target features?
 - What is the input data? Is it available?
 - What kind of problem are we facing? Binary classification? clustering? regression?
 - Can our outputs be predicted given the inputs?
 - Is the available data sufficient informative to learn the relationship between the inputs and the outputs?

Data collection

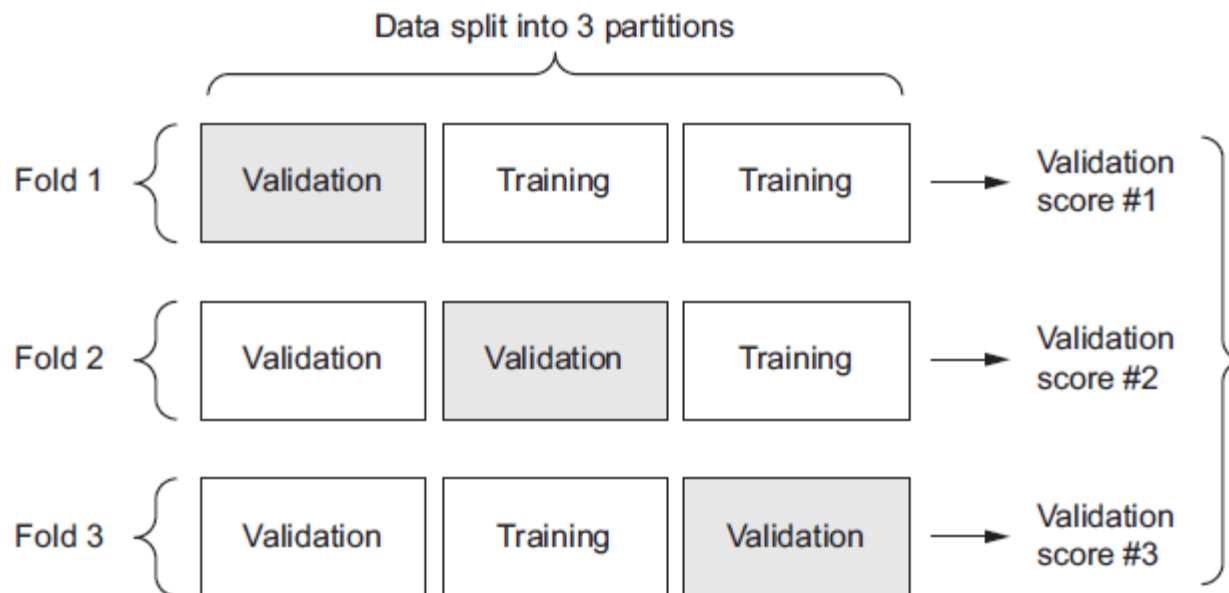
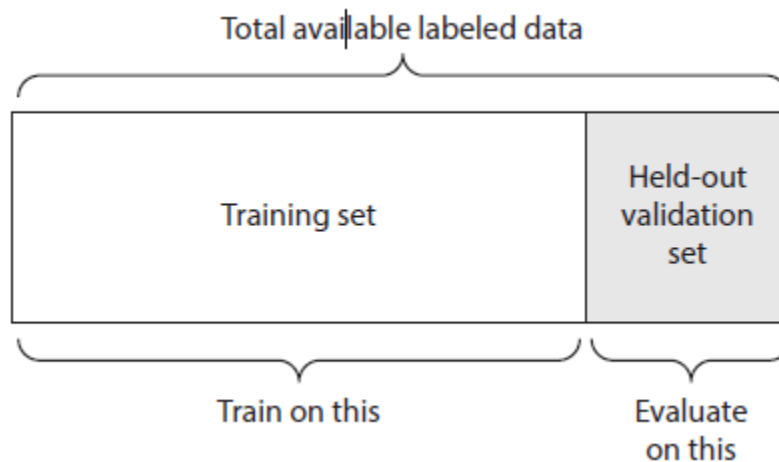
- Data collection is the process of gathering and measuring information on targeted variables in an established system, to enable one to answer questions and evaluate outcome.
- Data sources: sensors, surveys, interviews, books, images, etc.
- The more and better data that we get, the better our model will perform.

Choose a measure of success/evaluation metrics

- If you cannot measure it, you cannot improve it.
- What is considered success?
 - Precision
 - Accuracy
 - Recall
 - Fairness
 - biasness
- Regression problems: mean squared error (MSE)
- Classification problems: precision, accuracy, recall

Setting an evaluation protocol

- To measure the progress towards achieving the goal
- Data splitting:
 - Hold out validation: split the data into three parts/portions/containers to avoid information leak (training, testing, and validation).
 - K-fold validation: split the data into k partitions/folds, for each k in K , train the model with $K-1$ partitions and evaluate on k then take the average accuracy.
 - Iterated k-fold validation with shuffling: we use it when having little data available. Apply K-fold cross-validation several times and shuffling the data each time before splitting it into K partitions.
- Notes:
 - In classification problems, both training and testing data should be representative of the data, shuffle before splitting and that all classes are equally represented in training and testing sets (balanced).
 - Do not shuffle if you are trying to use the past to predict the future (time series)
 - Check if there are duplicates and redundant data to avoid inaccurate learning.



Data preparation

- Data preprocessing aims at making the raw data at hand more amenable to neural networks. This includes vectorization, normalization, handling missing values, and feature extraction.
- Dealing with missing data (eliminate features with many missing values with the risk of losing relevant information, imputing the missing values with estimates such as the mean value of the rest of the samples)
- Data vectorization: Handling categorical data (convert categorical string values into integers: red=0, green=1, blue=2)
- Feature scaling: learning algorithms perform much better when dealing with features that are on the same scale.
 - Normalization: rescaling the features to a range of [0,1] (min-max scaling)
 - Standardization: centering the features at mean of 0 and a standard deviation of 1 (normal distribution).
- Selecting meaningful features: PCA, selectKBest, autoencoder
- Data splitting.

$$X_{changed} = \frac{X - X_{min}}{X_{max} - X_{min}}$$

$$z = \frac{x - \mu}{\sigma}$$

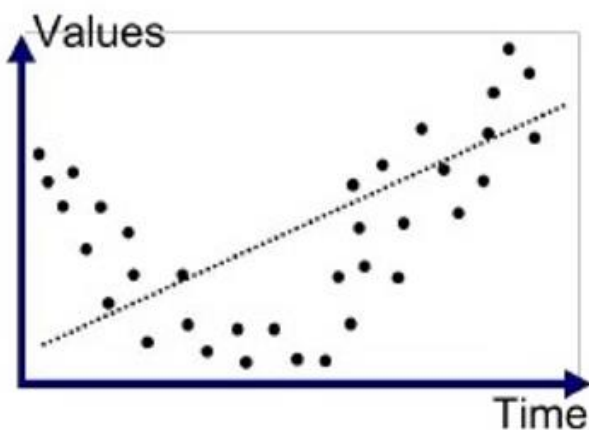
μ = Mean
 σ = Standard Deviation

Develop a benchmark/baseline model

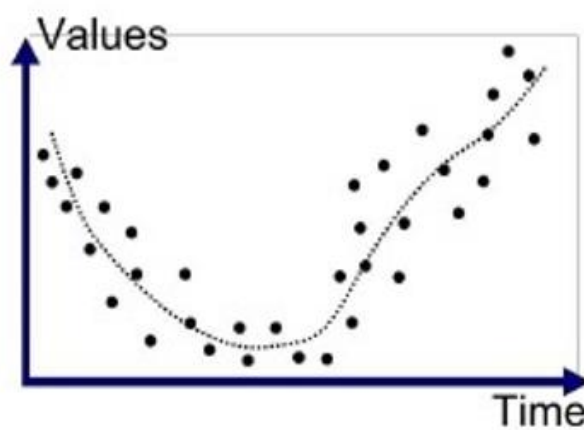
- Select a model to train using $[X, Y]$
- The goal is to achieve a model that can generalize — perform well on never-before-seen data.
- Split the data into 3 sets; training, validation and test sets.
- In training we adjust weights (model parameters).
- Validation set is used as the test set for the training set.
- From validation error and accuracy we can discover overfitting during the training and use techniques such as early stopping to stop training and do not overfit the data set.
- In testing we measure the generalization power of your model (avoid overfitting and min generalization) (test set is used to evaluate the final model).
- A model overfit the data when it has high model capacity or little data.

Overfitting and underfitting

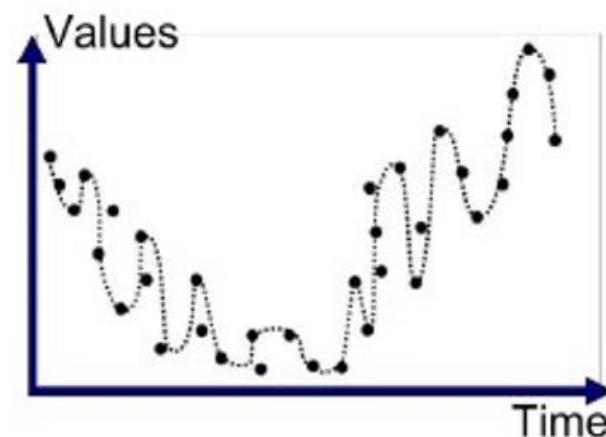
- Optimization is the process of adjusting a model to get the best performance possible on training data (the learning process).
- Generalization is how well the model performs on unseen data. The goal is to obtain the best generalization ability.
- To improve generalization: more data, adding weight regularization (L1 and L2), early stopping, model (size) reduction to reduce model capacity, dropout, etc.



Underfitted



Good Fit/Robust



Overfitted





(5) Developing a model that does better than a baseline

- Your goal at this stage is to achieve *statistical power*: develop a small model that is capable of beating a dumb RANDOM baseline.
- Three key choices to build your first working model:
 - Choose a proper last-layer activation that matches your problem (i.e., regression, classification, etc.).
 - Choose a proper loss function that matches the problem you are trying to solve.
 - Optimize configuration—What optimizer will you use? What will its learning rate be? etc.

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

Feature engineering for reading the time on a clock

How many inputs?

Raw data: pixel grid			CNN	
Better features: clock hands' coordinates	$\{x1: 0.7, y1: 0.7\}$ $\{x2: 0.5, y2: 0.0\}$	$\{x1: 0.0, y1: 1.0\}$ $\{x2: -0.38, y2: 0.32\}$	FFNN	
Even better features: angles of clock hands	theta1: 45 theta2: 0	theta1: 90 theta2: 140	Simple formula	



The place of feature engineering in the machine learning workflow

- Good features make the subsequent modeling step easy and the resulting model more capable of completing the desired task.
- Bad features may require a much more complicated model to achieve the same level of performance.

