Note of Hands-on ML with Scikit-Learn & TensorFlow by Lei Fu

Chapter 1. The Machine Learning Landscape

> Early specialized applications, e.g. Optical Character Recognition (OCR).
> Became main stream in 1990s: *spam filter*.

What Is Machine Learning?

Machine Learning is the science (and art) of programming computers so they can *learn* from data.

Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.

- Arthur Samuel, 1959

A computer program is said to learn from experience **E** with respect to some task **T** and some performance measure **P**, if its performance on T, as measured by P, improves with experience E.

- Tom Mitchell, 1997

Training set, training instance (sample), accuracy

Why Use Machine Learning

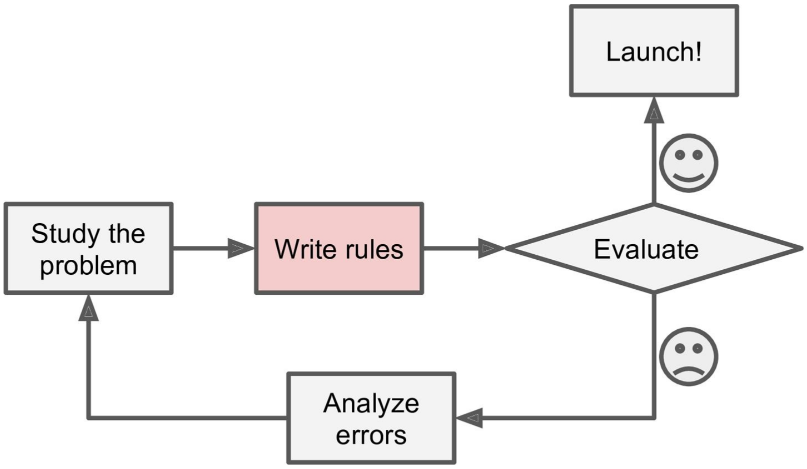

Figure 1-1. The traditional approach

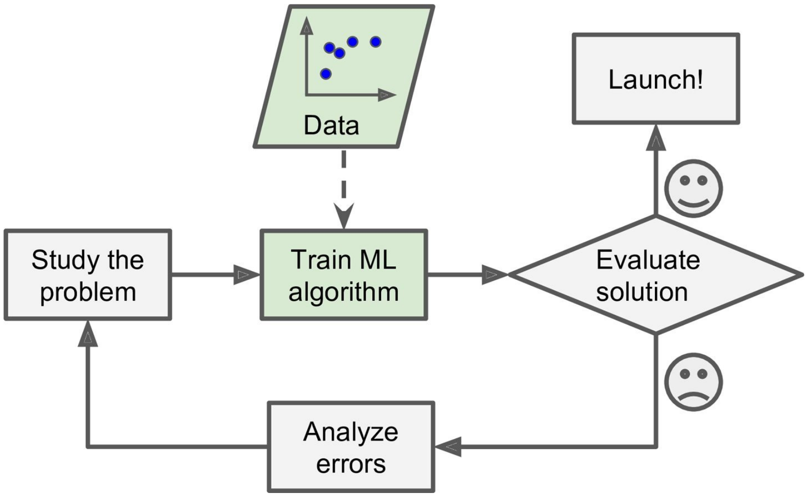

Figure 1-2. Machine Learning approach

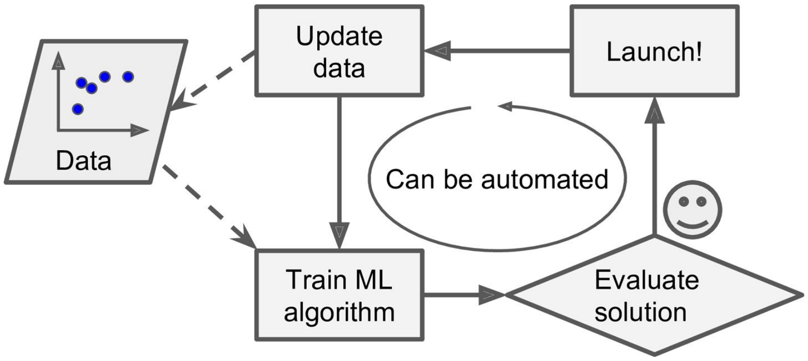

Figure 1-3. Automatically adapting to change

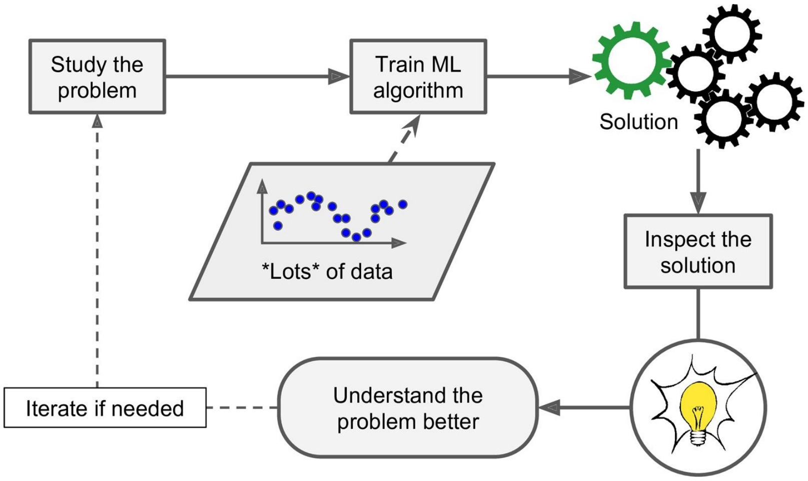

Figure 1-4. Machine learning can help humans learning

Machine Learning is great for:

1. Problems for which existing solutions require a lot of hand-tuning or long lists of rules: one Machine Learning algorithm can often simplify code and perform better (shorter, easier to maintain, accurate).

2. Complex problems for traditional approaches or have no known algorithm (speech recognition).
3. Fluctuating environments: a Machine Learning system can adapt to new data.
4. Help humans learning. Getting insights about complex problems and large amounts of data.

Types of Machine Learning Systems

Amount and type of human supervision:

1. supervised
2. unsupervised
3. semisupervised
4. reinforcement learning

Whether or not they can learn incrementally on the fly:

1. online
2. batch learning

Whether they work by simply comparing new data points to known data points, or instead detect patterns in the training data and build a predictive model:

1. instance-based
2. model-based

Supervised/Unsupervised Learning

Supervised Learning

Label: desired solution

Classification/regression, predictors.
Logistic Regression: regression algorithm sued for classification.
Attribute: a data type; Feature: attribute + its value

Supervised learning algorithms:

1. k-Nearest Neighbors
2. Linear Regression
3. Logistic Regression
4. Support Vector Machines (SVM)
5. Decision Tress and Random Forests
6. Neural networks

Unsupervised Learning
Training data is unlabeled.

Unsupervised learning algorithms:

Clustering

1. k-Means
2. Hierarchical Cluster Analysis (HCA)

3. Expectation Maximization

Visualization and dimensionality reduction

1. Principal Component Analysis (PCA)
2. Kernel PCA
3. Locally-Linear Embedding (LLE)
4. t-distributed Stochastic Neighbor Embedding (t-SNE)

Association rule learning

- Apriori
- Eclat

Visualization, dimensionality reduction, feature extraction. Reduce dimension of training data before feed to ML algorithm, benefit: run faster, less disk and memory usage, sometimes better performance.

Anomaly detection

Association rule learning
Goal: dig into large amounts of data and discover interesting relations between attributes.

Semisupervised Learning
Algorithm deals with partially labeled training data (usually a lot of unlabeled data and a little bit of labeled data).

Reinforcement Learning
The *learning system*, called an *agent* in this context, can observe the environment, select and perform actions, and get rewards in return (or penalties in the form of negative rewards). It must then learn by itself what is the best strategy, called a *policy*, to get the most reward over time. A policy defines what action the agent should choose when it is in a given situation.

Batch and Online Learning

Batch learning (*offline Learning*)
The system is incapable of learning incrementally: it must be trained using all the available data. Taking a lot of time and computing resources, so done *offline*.

(Time) Not able to adapt to rapidly changing data; (computing resource) expensive; not able to learn autonomously and limited resources.

Online learning (*incremental learning*)
Train the system incrementally by feeding it data instances sequentially, either individually or by small groups called mini-batches. (Each learning step is fast and cheap, so the system can learn about new data on the fly, as it arrives.)

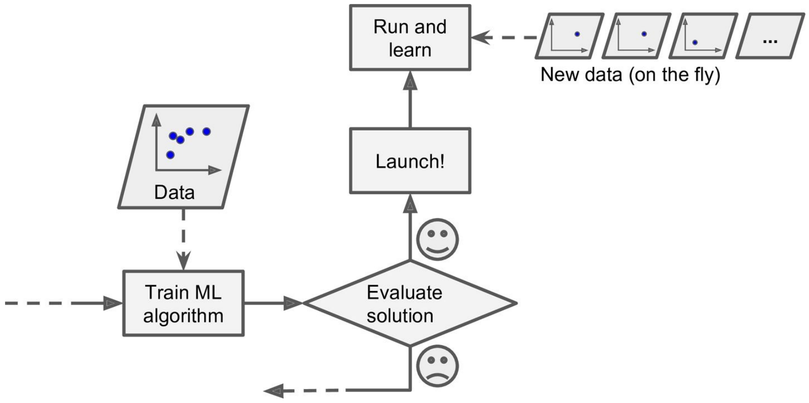

Figure 1-13. Online learning

Ideal for system:

- receive data as a continuous flow
- need to adapt to change rapidly or autonomously
- limited resources (discard old data, save space)
- *out-of-core learning* (huge data that cannot fit in memory)

WARNING:
This whole process is usually done offline (i.e., not on the live system), so online learning can be a confusing name. Think of it as incremental learning.

Learning rate: how fast to adapt to changing data.

Challenge: if bad data is fed to the system, its performance will gradually decline. To reduce this risk you need to monitor your system closely and promptly switch learning off (and possibly revert to a previously working state) if you detect a drop in performance. You may also want to monitor the input data and react to abnormal data (anomaly detection).

Supervised/Unsupervised Learning

One more way to categorize Machine Learning systems is by how they *generalize*.

Instance-based learning
The system learns the examples by heart, then generalizes to new cases using a *similarity measure*.

Model-based learning
Another way to generalize from a set of examples is to build a model of these examples, then use that model to make *predictions*.

Utility function (or fitness function) measures how good your model is.
Cost function measures how bad it is.

Linear Regression algorithm: you feed it your training examples and it finds the parameters that make the linear model fit best to your data. This is called *training the model*.

In summary:

- Study the data.
- Select a model.
- Train it on the training data (i.e., the learning algorithm searches for the model parameter values that minimize a cost function).
- Finally, you applied the model to make predictions on new cases (this is called inference), hoping that this model will generalize well.

Main Challenges of Machine Learning

Bad data or bad algorithm.

Insufficient Quantity of Training Data

Very different Machine Learning algorithms, including fairly simple ones, performed almost identically well on a complex problem of natural language disambiguation once they were given enough data (Michele Banko and Eric Brill, 2001)

The idea that data matters more than algorithms for complex problems was further popularized by Peter Norvig et al in 2009 (“The Unreasonable Effectiveness of Data”). It should be noted, however, that small- and medium-sized datasets are still very common, and it is not always easy or cheap to get extra training data, so don’t abandon algorithms just yet.

Nonrepresentative Training Data

In order to generalize well, it is crucial that your training data be representative of the new cases you want to generalize to.

Difficulties:

- If the sample is too small, you will have *sampling noise* (i.e., nonrepresentative data as a result of chance).
- Even very large samples can be nonrepresentative if the sampling method is flawed. This is called *sampling bias*.

Poor-Quality Data

Errors, outliers, and noise require a significant mount of time to clean.

- Obvious outliers, simply discard them or fix the errors manually.
- If some instances are missing a few features, decide ignore this attribute, ignore these instances, fill in the missing values, or train one model with the feature and another without it.

Irrelevant Features

A critical part of the success of a Machine Learning project is coming up with a good set of features to train on. This process, called *feature engineering*, involves:

- *Feature selection*: selecting the most useful features to train on among existing features.
- *Feature extraction*: combining existing features to produce a more useful one (dimensionality reduction).

- Creating new features by gathering new data.

Overfitting the Training Data

Overfitting: the model performs well on the training data, but it does not generalize well.

WARNING:
Overfitting happens when the model is too complex relative to the amount and noisiness of the training data.

Solutions:

- Simplify the model by selecting one with fewer parameters, by reducing the number of attributes in the training data or by constraining the model (*regularization*)
- Gather more training data
- Reduce the noise in the training data (e.g., fix data errors and remove outliers)

Hyperparameter: controls the amount of regularization to apply during learning.

Underfitting the Training Data

Underfitting: the model is too simple to learn the underlying structure of the data.

Solutions:

- Selecting a more powerful model, with more parameters
- Feeding better features to the learning algorithm (feature engineering)
- Reducing the constraints on the model (e.g., reducing the regularization hyperparameter)

Stepping Back

Big picture:

- Machine Learning is about making machines get better at some task by learning from data, instead of having to explicitly code rules.
- There are many different types of ML systems: supervised or not, batch or online, instance-based or model-based, and so on.
- In a ML project you gather data in a training set, and you feed the training set to a learning algorithm. If the algorithm is model-based, it tunes some parameters to fit the model to the training set (i.e., to make good predictions on the training set itself), and then hopefully it will be able to make good predictions on new cases as well. If the algorithm is instance-based, it just learns the examples by heart and uses a similarity measure to generalize to new instances.
- The system will not perform well if your training set is too small, or if the data is not representative, noisy, or polluted with irrelevant features (garbage in, garbage out). Lastly, your model needs to be neither too simple (in which case it will underfit) nor too complex (in which case it will overfit).
- Evaluate the trained model, and fine-tune it if necessary.

Testing and Validating

Split data into *training set* and *test set*.

Generalization error (out-of-sample error): the error rate on new cases.

Validation set: to determine hyperparameter.

Cross-validation (purpose: avoid “wasting” too much training data)
the training set is split into complementary subsets, and each model is trained against a different combination of these subsets and validated against the remaining parts. Once the model type and hyperparameters have been selected, a final model is trained using these hyperparameters on the full training set, and the generalized error is measured on the test set.

No Free Lunch (NFL) theorem
There is no model that is a *priori* guaranteed to work better. (David Wolpert, 1996)

In other words, if you make absolutely no assumption about the data, then there is no reason to prefer one model over any other. The only way to know for sure which model is best is to evaluate them all. Since this is not possible, in practice you make some reasonable assumptions about the data and you evaluate only a few reasonable models.