Hands-On Machine Learning Chapter 1 Questions

by Fabian Leuk (12215478)

universität innsbruck

Read through the definition of machine learning. You want to estimate the weight of a person based on his or her height. Now try to describe the terms task, performance measure, training set, training instance, and model using this concrete example.

Machine Learning definition

The job a machine learning system is supposed to do is called the **task**.

The metric by which the machine learning system's ability of completing a task is measured is called a **performance measure**.

The examples that the system uses to learn are called **training set**.

Each training example is called a **training instance** or sample.

The part of a machine learning system that learns and makes predictions is called a **model**.

Task: Estimate weight of a person based on height

Performance measure: MSE, RMSE, MAE

Training set: n examples of height - weight combinations.

Training instance: {"height":185,"weight":95}

Model: Linear regression model

universität innsbruck

Read through the definition of machine learning. You want to estimate the weight of a person based on his or her height. Now try to describe the terms task, performance measure, training set, training instance, and model using this concrete example.

Screenshots from book

> 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

Your spam filter is a machine learning program that, given examples of spam emails (flagged by users) and examples of regular emails (nonspam, also called "ham"), can learn to flag spam. The examples that the system uses to learn are called the *training set*. Each training example is called a *training instance* (or *sample*). The part of a machine learning system that learns and makes predictions is called a *model*. Neural networks and random forests are examples of models.

In this case, the task *T* is to flag spam for new emails, the experience *E* is the *training data*, and the performance measure *P* needs to be defined; for example, you can use the ratio of correctly classified emails. This particular performance measure is called *accuracy*, and it is often used in classification tasks.

What is the difference between the traditional programming and the machine learning approach? What motivates the machine learning approach? When will I use one approach and when will I use the other, think about an example. What other strengths does a machine learning approach have? Try using the terms “fluctuating environments” and “data mining”.

Machine Learning vs Traditional Programming

Traditional programming refers to implementing an algorithm to solve a task. It tends to be suitable for tasks that do not underlie fluctuation or an overwhelming complexity.

Machine Learning in contrast is about making machines get better at some task by learning from data, instead of having to explicitly code rules. The performance of ML systems tends to outperform traditional programming approaches for a number of problems such as speech recognition. As Machine Learning systems learn by their experience, they are suitable for tasks being exposed to environmental fluctuations.

An example is the detection of money laundering in bank transactions. A traditional programming approach would decide based on the origin, target and size of the transaction if a transaction should be reviewed by a bank agent. However, origins and targets might be volatile over time as legislation in countries changes and launderers might find ways to circumvent programmed approaches such as splitting the money up in smaller transactions, so that a ML approach might be more suitable as it learns by agents detecting transactions.

What is the difference between the traditional programming and the machine learning approach? What motivates the machine learning approach? When will I use one approach and when will I use the other, think about an example. What other strengths does a machine learning approach have? Try using the terms “fluctuating environments” and “data mining”.

Machine Learning vs Traditional Programming

Machine Learning is great for

- problems for which existing solutions require a lot of fine-tuning or long lists of rules.

  => Detection of Spam-Mails requires the knowledge of which word combinations increase the probability of a mail being spam

- complex problems for which using a traditional approach yields no good solution

  => Speech Recognition

- fluctuating environments

  => Spam-Mail authors might change their spam mails, so that word combinations in spam mails change

- getting insights about complex problems and large amounts of data (data mining)

  => find hidden correlations in data e.g., for customer segmentation

What is the difference between the traditional programming and the machine learning approach? What motivates the machine learning approach? When will I use one approach and when will I use the other, think about an example. What other strengths does a machine learning approach have? Try using the terms “fluctuating environments” and “data mining”.

Screenshots from book

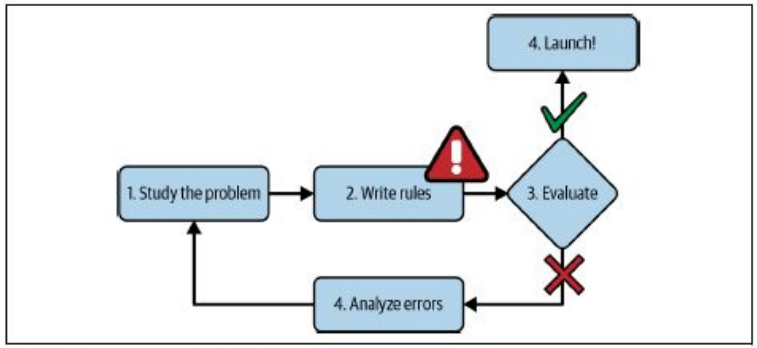


Figure 1-1. The traditional approach

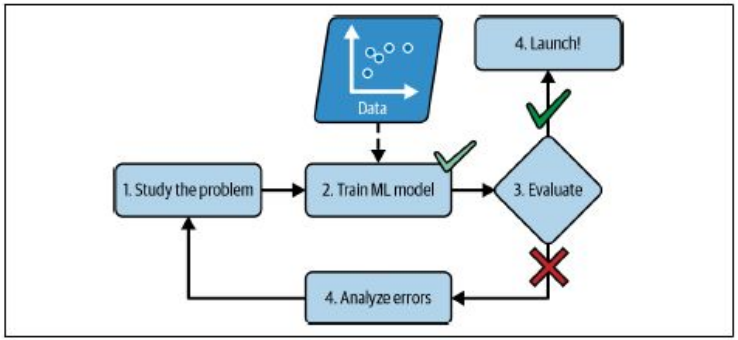


Figure 1-2. The machine learning approach

To summarize, machine learning is great for:

- Problems for which existing solutions require a lot of fine-tuning or long lists of rules (a machine learning model can often simplify code and perform better than the traditional approach)
- Complex problems for which using a traditional approach yields no good solution (the best machine learning techniques can perhaps find a solution)
- Fluctuating environments (a machine learning system can easily be retrained on new data, always keeping it up to date)
- Getting insights about complex problems and large amounts of data

Try to summarize all five training supervisions (supervised learning, unsupervised learning, self-supervised learning, semi-supervised learning, and reinforcement learning).

Summarize all five training supervisions

In **supervised learning**, the training set includes the solutions, called labels => Classification, Regression

In **unsupervised learning**, training data is unlabeled => Clustering, Visualization, Dimensionality Reduction, Anomaly Detection, Association Rule Learning

In **semi-supervised learning**, parts of the data is labeled, while other parts aren't. => Mix of supervised and unsupervised problems

In **self-supervised learning**, a fully labeled data set is generated from a fully unlabeled one. => Transfer learning

A machine learning system (agent) that observes the environment, selects and performs actions, receives rewards or penalties uses **reinforcement learning**. It must find out by itself what the best policy is to get rewards or prevent penalties. A policy defines what action the agent should choose in a given situation.

Try to summarize all five training supervisions (supervised learning, unsupervised learning, self-supervised learning, semi-supervised learning, and reinforcement learning).

Screenshots from book

In *supervised learning*, the training set you feed to the algorithm includes the desired solutions, called *labels* (Figure 1-5).

In *unsupervised learning*, as you might guess, the training data is unlabeled (Figure 1-7). The system tries to learn without a teacher.

Since labeling data is usually time-consuming and costly, you will often have plenty of unlabeled instances, and few labeled instances. Some algorithms can deal with data that's partially labeled. This is called *semi-supervised learning* (Figure 1-11).

Another approach to machine learning involves actually generating a fully labeled dataset from a fully unlabeled one. Again, once the whole dataset is labeled, any supervised learning algorithm can be used. This approach is called *self-supervised learning*.

Reinforcement learning is a very different beast. 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, as shown in Figure 1-13). 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.

universität innsbruck

What is the difference between batch and online learning? What is the disadvantage of batch learning? Do you already know of models that definitely do batch learning? What does the term “out-of-core learning” mean? What are challenges in online learning?

Batch Learning

Batch learning refers to a machine learning system being trained offline with a set of training data. Once launched into production, it *does not learn incrementally*.

The major disadvantage is that the system’s performance decays over time given environmental dynamism. (model rot, data drift) => retrain model and launch again

Thus, batch models are especially suitable for models that do not tend to be exposed to model rot or data drift such as speech recognition.

Online Learning

In **online learning**, the system is *trained incrementally* by data instances sequentially, making it extremely suitable for systems that need to adapt extremely fast.

It is also suitable for huge data sets that do not fit into one machine’s main memory. (“**out-of-core learning**”)

One challenge is to find the right learning rate, which determines how fast new data can be included and how fast old data is forgotten by the system.

Another challenge is that the quality of new data determines the quality of the new system. A high learning rate again increases the exposition of the system to this effect => anomaly detection algorithm to disable learning temporarily

universität innsbruck

What is the difference between batch and online learning? What is the disadvantage of batch learning? Do you already know of models that definitely do batch learning? What does the term “out-of-core learning” mean? What are challenges in online learning?

Screenshots from book

In *batch learning*, the system is incapable of learning incrementally: it must be trained using all the available data. This will generally take a lot of time and computing resources, so it is typically done offline. First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned. This is called *offline learning*.

Unfortunately, a model's performance tends to decay slowly over time, simply because the world continues to evolve while the model remains unchanged. This phenomenon is often called *model rot* or *data drift*. The solution is to regularly retrain the model on up-to-date data. How often you need to do that depends on the use case: if the model classifies pictures of cats and dogs, its performance will decay very slowly, but if the model deals with fast-evolving systems, for example making predictions on the financial market, then it is likely to decay quite fast.

Even a model trained to classify pictures of cats and dogs may need to be retrained regularly, not because cats and dogs will mutate overnight, but because cameras keep changing, along with image formats, sharpness, brightness, and size ratios. Moreover, people may love different breeds next year, or they may decide to dress their pets with tiny hats—who knows?

If you want a batch learning system to know about new data (such as a new type of spam), you need to train a new version of the system from scratch on the full dataset (not just the new data, but also the old data), then replace the old model with the new one. Fortunately, the whole process of training, evaluating, and launching a machine learning system can be automated fairly easily (as we saw in Figure 1-3), so even a batch learning system can adapt to change. Simply update the data and train a new version of the system from scratch as often as needed.

In *online learning*, you train the system incrementally by feeding it data instances sequentially, either individually or in 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 (see Figure 1-14).

Online learning is useful for systems that need to adapt to change extremely rapidly (e.g., to detect new patterns in the stock market). It is also a good option if you have limited computing resources; for example, if the model is trained on a mobile device.

Additionally, online learning algorithms can be used to train models on huge datasets that cannot fit in one machine's main memory (this is called *out-of-core* learning). The algorithm loads part of the data, runs a training step on that data, and repeats the process until it has run on all of the data (see Figure 1-15).

One important parameter of online learning systems is how fast they should adapt to changing data: this is called the *learning rate*. If you set a high learning rate, then your system will rapidly adapt to new data, but it will also tend to quickly forget the old data (and you don't want a spam filter to flag only the latest kinds of spam it was shown). Conversely, if you set a low learning rate, the system will have more inertia; that is, it will learn more slowly, but it will also be less sensitive to noise in the new data or to sequences of nonrepresentative data points (outliers).

universität innsbruck

Briefly name all the challenges we encounter in Machine Learning
The data are the basis for our models. If the data is not representative, we will only be able to make limited predictions with our model. What do we mean by sampling noise and sampling bias?
What are features? Use a simple ML model and describe what is meant by feature selection and feature extraction?
What is and what favors overfitting in Machine Learning and how can we prevent it?
What is and what favors underfitting in Machine Learning and how can we prevent it?

Main Challenges of Machine Learning

The two main challenges of machine learning are bad model and bad data. Bad data can mean that training data sets were insufficiently large, non representative or low-quality (errors, outliers, noise) data.

Sampling noise is a flaw in data that arises due to small data sets. Small data sets might result in non representative data as an outcome of chance. **Sampling bias** refers to a data flaw related to the collection of data.

In Machine Learning, a **feature** is input data the model is trained on.

Feature selection is the process of deciding on the most relevant features the model should be trained on. **Feature extraction** is the process of combining existing feature to produce a more useful one. For example when predicting airline delays, the lunch served on the flight is most likely not a determinant of flight delay, so it can be excluded from the training data, which is feature selection. The weekday however might relate to flight delay, but in a way that only a “busy weekday” (Monday, Friday) is a determinant for flight delay, so that the categorical variable weekday can be transformed to a binary (busy weekday: TRUE or FALSE). This would be feature extraction.

Overfitting means that a complex model performs well on the training data, but does not generalize well. Overfitting occurs when the model is too complex relative to the amount and noisiness of training data. Countermeasures are simplifying the model by selecting fewer parameters, reducing the number of features, constraining the model (regularization), gathering more training data or reduce the noise in the training data.

Underfitting means that a simple model is not able to learn the underlying structure of the data. One factor resulting to underfitting is not enough features included in the data. Countermeasures included selecting a more powerful and complex model, feed more or better features or reduce constraints on the model.

Briefly name all the challenges we encounter in Machine Learning
The data are the basis for our models. If the data is not representative, we will only be able to make limited predictions with our model. What do we mean by sampling noise and sampling bias?
What are features? Use a simple ML model and describe what is meant by feature selection and feature extraction?
What is and what favors overfitting in Machine Learning and how can we prevent it?
What is and what favors underfitting in Machine Learning and how can we prevent it?

Screenshots from the book

It is crucial to use a training set that is representative of the cases you want to generalize to. This is often harder than it sounds: if the sample is too small, you will have *sampling noise* (i.e., nonrepresentative data as a result of chance), but even very large samples can be nonrepresentative if the sampling method is flawed. This is called *sampling bias*.

- *Feature selection* (selecting the most useful features to train on among existing features)
- *Feature extraction* (combining existing features to produce a more useful one—as we saw earlier, dimensionality reduction algorithms can help)
- Creating new features by gathering new data

Say you are visiting a foreign country and the taxi driver rips you off. You might be tempted to say that *all* taxi drivers in that country are thieves. Overgeneralizing is something that we humans do all too often, and unfortunately machines can fall into the same trap if we are not careful. In machine learning this is called *overfitting*: it

Underfitting the Training Data

As you might guess, *underfitting* is the opposite of overfitting: it occurs when your model is too simple to learn the underlying structure of the data. For example, a linear model of life satisfaction is prone to underfit; reality is just more complex than the model, so its predictions are bound to be inaccurate, even on the training examples.

Here are the main options for fixing this problem:

- Select a more powerful model, with more parameters.
- Feed better features to the learning algorithm (feature engineering).
- Reduce the constraints on the model (for example by reducing the regularization hyperparameter).

Overfitting happens when the model is too complex relative to the amount and noisiness of the training data. Here are possible solutions:

- Simplify the model by selecting one with fewer parameters (e.g., a linear model rather than a high-degree polynomial model), by reducing the number of attributes in the training data, or by constraining the model.
- Gather more training data.
- Reduce the noise in the training data (e.g., fix data errors and remove outliers).

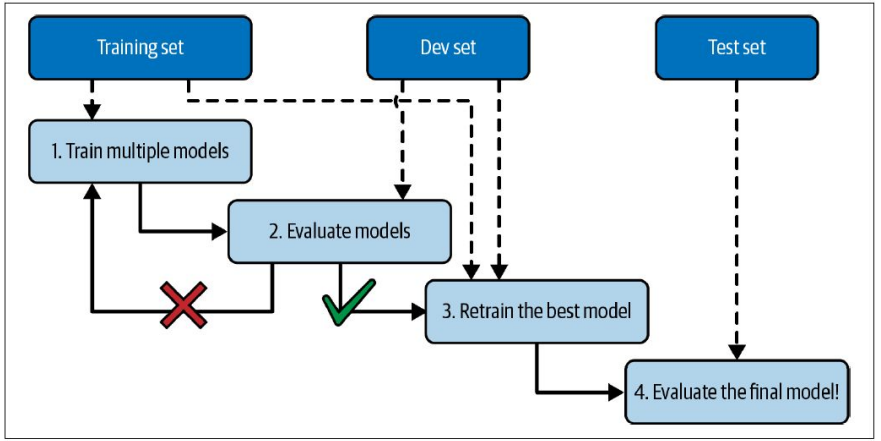
universität innsbruck

What is the motivation of holdout validation? Using Figure 1-25, try to explain what is the test set, the training set, and the validation set?

Testing and Validating

When training a ML model, training data is used. Afterwards, it's generalization is judged based on the measured generalization error on a test data set.

For hyperparameter tuning, multiple hyperparameters should be tested. In order to avoid overfitting to test data, the training data is subdivided into training set and dev set. The dev set is now used to judge the different models. Finally, the full training set (dev set + training set) is used to train the model given the best hyperparameters and the test set is used to report the generalization error. This process is called **holdout validation.**



universität innsbruck

What is the motivation of holdout validation? Using Figure 1-25, try to explain what is the test set, the training set, and the validation set?

Screenshots from the book

Evaluating a model is simple enough: just use a test set. But suppose you are hesitating between two types of models (say, a linear model and a polynomial model): how can you decide between them? One option is to train both and compare how well they generalize using the test set.

Now suppose that the linear model generalizes better, but you want to apply some regularization to avoid overfitting. The question is, how do you choose the value of the regularization hyperparameter? One option is to train 100 different models using 100 different values for this hyperparameter. Suppose you find the best hyperparameter value that produces a model with the lowest generalization error—say, just 5% error. You launch this model into production, but unfortunately it does not perform as well as expected and produces 15% errors. What just happened?

The problem is that you measured the generalization error multiple times on the test set, and you adapted the model and hyperparameters to produce the best model *for that particular set*. This means the model is unlikely to perform as well on new data.

A common solution to this problem is called *holdout validation* (Figure 1-25): you simply hold out part of the training set to evaluate several candidate models and select the best one. The new held-out set is called the *validation set* (or the *development set*, or *dev set*). More specifically, you train multiple models with various hyperparameters on the reduced training set (i.e., the full training set minus the validation set), and you select the model that performs best on the validation set. After this holdout validation process, you train the best model on the full training set (including the validation set), and this gives you the final model. Lastly, you evaluate this final model on the test set to get an estimate of the generalization error.

universität innsbruck