Machine Learning - 1*

Anca Ralescu
EECS Department
University of Cincinnati
Cincinnati, OH, USA
Anca.Ralescu@uc.edu

1 The ML landscape – Terminology & Notation

In an attempt to define the field of ML we look at three definitions:

Definition 1 *(due to Arthur Samuel - 1959) ML is the field of study which gives computers the ability to learn without being* explicitly *programmed.*

Obviously, something is a bit amiss with the above definition because the only way we can give a computer *any ability* is by programming them. Therefore, a more concrete definition specifies that....

Definition 2 *Machine Learning is the science (and art) of programming the computers such they can learn from data.*

Definition **??** introduces a concrete notion, that of learning from data, but with this one must define what exactly means to learn from the data. This brings us to the third definition below, due to Tom Mitchell (professor of Machine Learning at CMU, and department head of the first Department of Machine Learning at a US university).

Definition 3 *A computer program is said to learn from experience E (data), with respect to some task T, and some performance measure P, if its performance on T, as measured by P improves with experience E.*

Why use ML?

Consider the task of designing a spam filter using traditional programming techniques. By the way, programming a spam filter, based on the words in the subject line of a message, was one of the first successful ML applications. Figure **??** shows the steps in such an approach:

Iterate between the steps below until the algorithm is good to launch:

*Based on J. Geron's book.

1. Identify clues to a message being a spam: these may be words such as 'credit cards', 'amazing' in the subject line
2. Write an algorithm to detect these words
3. Test the performance

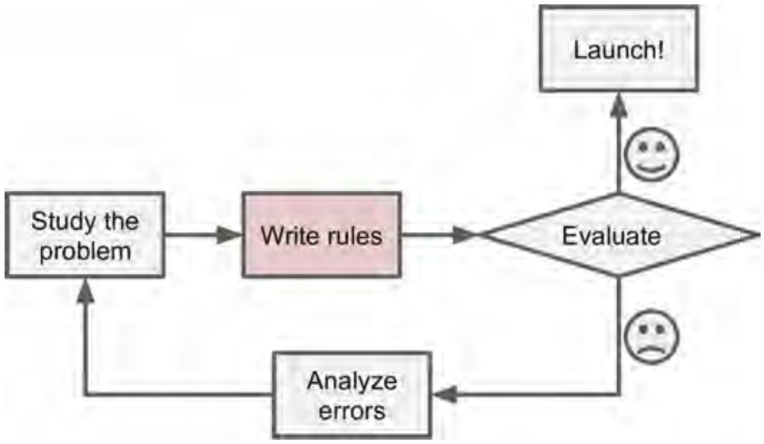

Figure 1: Steps in a traditional (non ML) spam filter algorithm.

1. Traditional approach: requires a long and rather complex list of words.
2. ML based algorithm:
   - automatically learn the patterns which can distinguish 'ham' from 'spam'.
   - much shorter and usually much more accurate program.

For the particular "spam" problem, as in many other cases, including for example, all fraudulent activities, the perpetrators are never idle. There is always a "tug-of-war" like game between those attackers and those defenders. That means that those producing spam messages are active. Upon noticing that messages with a particular text in the subject line have been rejected (by the user, for example), they will (subtly) change those programs as well. In that case, the defenders must update their list of terms indicative of spam.... but this may not happen before they have already suffered some casualties.

A spam filter based on Machine Learning techniques **automatically learns** which words and phrases are good predictors of spam by detecting unusually frequent patterns of words in the 'spam' examples compared to the 'ham' examples as shown in Figure **??**.

Advantages of the ML approach:

- Much shorter program

- Much easier to to maintain
- Most likely more accurate.

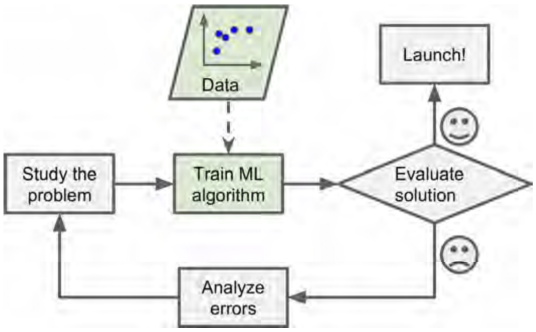

Figure 2: Machine Learning approach.

The most important aspect of the ML based approach is adaptability, as illustrated in Figure **??**.

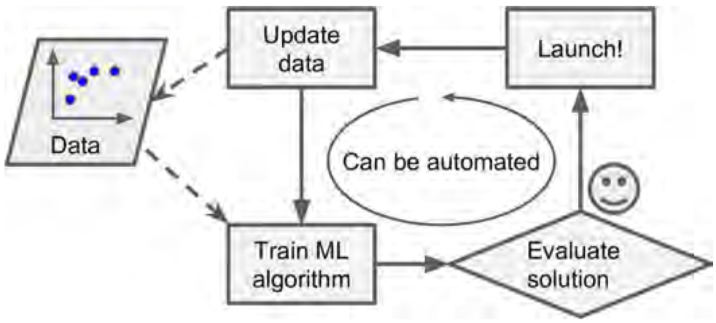

Figure 3: Automatically adapting to change.

ML is very important for attacking problems where there are no known algorithms, e.g.,speech recognition.

Moreover, recently ML approaches have been used to help human users to better understand a problem, and eventually find a solutions (e.g., conjecture of an equation, which then is subject to proof) (see Figure **??**).

For instance, once the spam filter has been trained on enough spam, it can easily be inspected to reveal the list of words and combinations of words that it believes are the best predictors of spam.

Data Mining:

Applying ML techniques to dig into large amounts of data can help discover patterns that were not immediately apparent. Sometimes this will reveal unsuspected correla-

tions or new trends, and thereby lead to a better understanding of the problem.

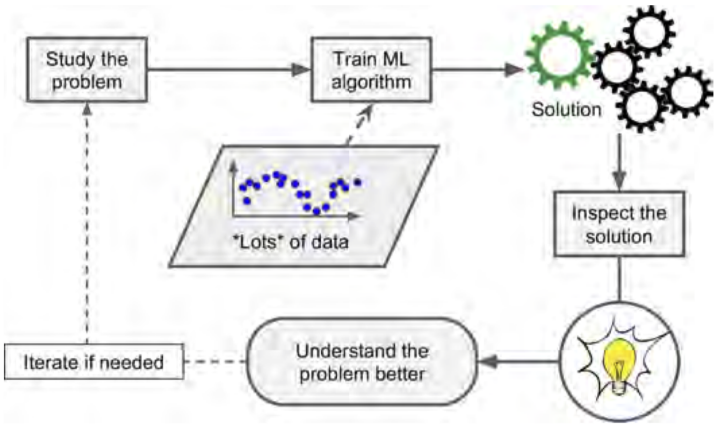

Figure 4: Machine Learning can help humans learn.

To summarize, Machine Learning is great for:

- Problems for which existing solutions require a lot of hand-tuning or long lists of rules $\Longrightarrow$ one ML algorithm can often simplify code and perform better.
- Complex problems for which there is no good solution at all using a traditional approach $\Longrightarrow$ ML techniques can find a solution.
- Fluctuating environments $\Longrightarrow$ an ML system can adapt to new data.
- Getting insights about complex problems and large amounts of data.

Types of Machine Learning Systems

ML system can be classified according to various aspects:

1. **Training mode**: with human supervision
   - supervised,
   - unsupervised,
   - semisupervised, and
   - Reinforcement Learning
2. **Incremental/ online versus batch learning**
3. **How they work:**
   - by simply comparing new data points to known data points (instance-based),
   - instead detect patterns in the training data and build a predictive model (model-based learning)

Supervised / Unsupervised Learning

Based on the amount of supervision (usually, this is feedback from the user) during training. Typical of supervised ML are classification tasks. The training examples presented to the system have a label, e.g., 'spam' or 'ham', as illustrated in Figure **??**. Notice that in the classification task, we want to predict a class. Therefore, the

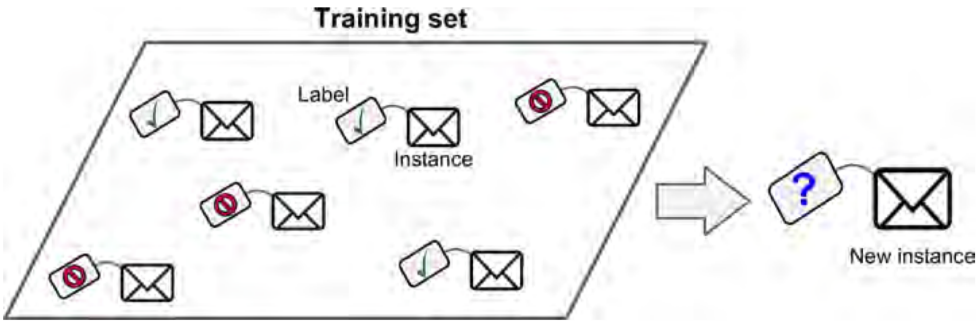

Figure 5: A labeled training set for supervised learning (e.g., spam classification).

domain of the answer is discrete, or categorical. The domain may be continuous, i.e., a subset of the real line ($\mathbb{R}$). In this case, the ML problem is called a regression problem[1] [2]. Typical examples of regression include predicting the price of a house, car, etc.

[1]The term regression has a slightly negative connotation. However, it was introduced in statistics as *regression to the mean*, by Francis Galton who observed that the children of tall people are usually shorter than their parents.

[2]https://en.wikipedia.org/wiki/Francis_Galton#Correlation_and_regression+

based on the values of various *attributes*[3]. The attributes/features are usually referred to *predictors*. In a price predicting problem, the price is the target or the predicted variable. Figure **??** illustrates an instance of regression

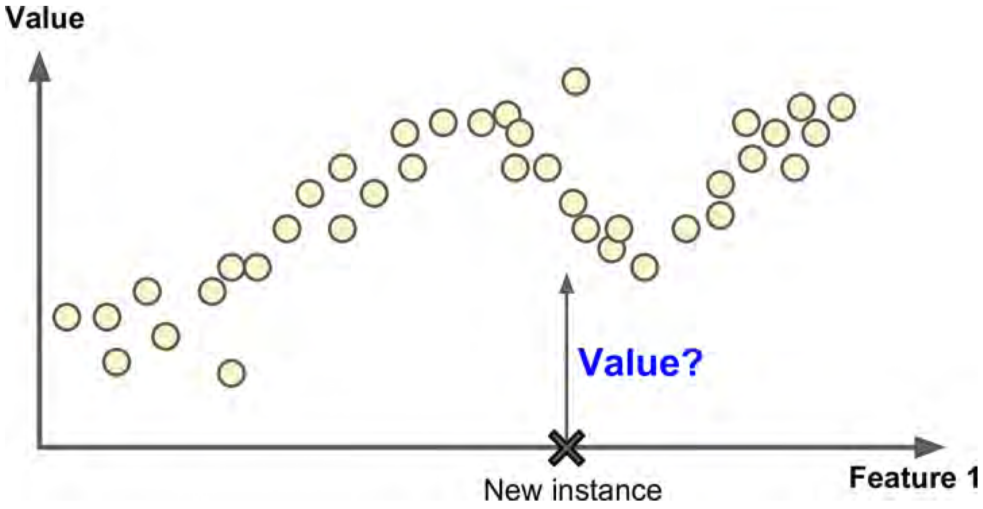

Figure 6: Regression.

Classification vs regression. Regression is a more difficult problem than classification in the sense that if we solve a regression problem, then the curve obtained can be used to divide the input space into two regions - above and below the regression curve, and therefore, examples fall into two categories corresponding to these two regions.

Moreover, regression can be used for classification in another way, by fitting a curve that predicts class probability (value in the continuous interval $[0, 1]$) – this is called logistic regression. Then a threshold on the probability of an example, is used to decide the class for that example.

Most important supervised learning algorithms include:

- k-Nearest Neighbors
- Linear Regression
- Logistic Regression
- Support Vector Machines (SVMs)
- Decision Trees and Random Forests
- Neural networks (Some NN architectures can be unsupervised, or semisupervised)

[3]Attribute values are called *features*. For example, height of a person is an attribute, values such as 5ft5, or 6ft are features. However, in recent years people tend to use the terms attributes and features interchangeably.

Unsupervised learning

Training data are not labeled, as illustrated in Fig. **??**.

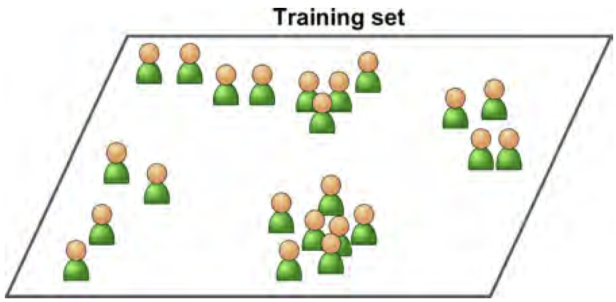

Figure 7: An unlabeled training set for unsupervised learning.

Most important unsupervised learning algorithms include:

1. Clustering (see Fig **??**)
    - k-Means
    - Hierarchical Cluster Analysis (HCA)
    - Expectation Maximization
2. Visualization and dimensionality reduction: feed a lot of complex and unlabeled data, and they output a 2D or 3D representation of your data that can easily be plotted.
    - Principal Component Analysis (PCA)
    - Kernel PCA
    - Locally-Linear Embedding (LLE)
    - t-distributed Stochastic Neighbor Embedding (t-SNE): See Figure **??**
3. Association rule learning
    - Apriori
    - Eclat

Related unsupervised tasks:

- Dimensionality reduction: simplify the data without losing too much information.
    - *Feature extraction:* merge several correlated features into one (car's mileage is very correlated with its age, $\longrightarrow$ merge them into one feature
- Anomaly detection: e.g.,

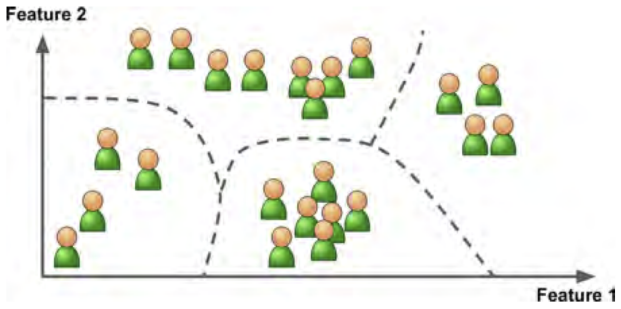

Figure 8: Clustering

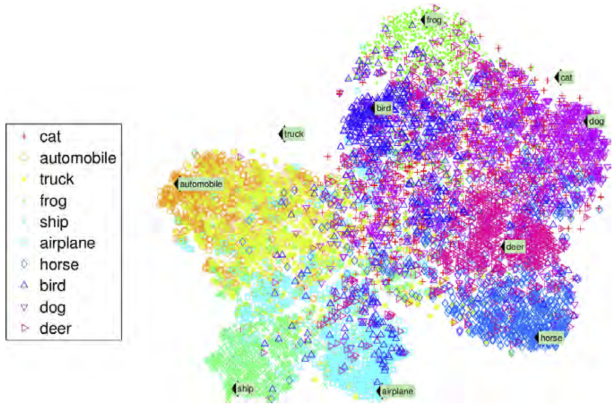

Figure 9: Example of a t-SNE visualization highlighting semantic clusters.

  - detecting unusual credit card transactions to prevent fraud, catching manufacturing defects, or automatically removing outliers from a dataset before feeding it to another learning algorithm.
  - The system is trained with normal instances,
  - A new instance is analyzed as a normal one or anomaly (see Figure 1-10).

- Association rule learning: dig into large amounts of data and discover interesting relations between attributes. For example, suppose you own a supermarket.

Semisupervised Learning

partially labeled data. Labeled data is quite expensive and hard to come by. Then there are various options to deal with the lack of labeled data. One of these is *data augmentation*. Another way is to use semisupervised learning by using labeled and unlabeled training examples (Figure **??**).

Example: Google Photos, are good examples of this:

Figure 10: Anomaly detection.

- Upload all family photos to the service
- Unsupervised par (clustering)t: automatically recognizes that the same person A shows up in photos 1, 5, and 11 while another person B shows up in photos 2, 5, and 7.
- The system needs the user to tell it who these people are: ideally, just one label per person $\longrightarrow$ the system is then able to name everyone in every photo (In practice it often creates a few clusters per person, and sometimes mixes up two people who look alike, so a few labels per person and manually clean up some clusters may be needed).

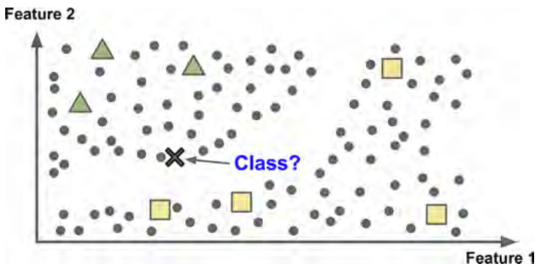

Figure 11: Semisupervised learning.

Reinforcement Learning

- Very different from the above systems
- The learning system – an agent
  1. observes the environment,
  2. selects and perform actions, and
  3. gets rewards in return (or penalties in the form of negative rewards, as in Figure 1-12).

- Objective is 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.

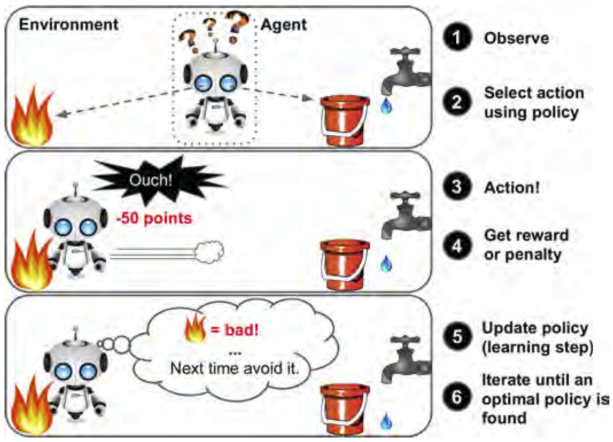

Figure 12: Reinforcement Learning.

Example of reinforcement learning:

- Robots implement Reinforcement Learning algorithms to learn how to walk.
- DeepMind's AlphaGo program is also a good example of Reinforcement Learning (it beat the world champion Lee Sedol at the game of Go.
- It learned its winning policy by analyzing millions of games, and then playing many games against itself. Note that learning was turned off during the games against the champion; AlphaGo was just applying the policy it had learned.

Batch and Online Learning

Another criterion used to classify ML systems is whether or not the system can learn incrementally from a stream of incoming data.

Batch learning or offline learning

- System must be trained using all the available data.
- Typically done offline because it takes a lot of time and computing resources.
- First the system is trained, and then it is launched into production and runs without learning anymore; it just applies what it has learned.

- To know about new data (such as a new type of spam), a new version of the system must be trained from scratch on the full dataset (old data + new data)
- Process can be automated, as shown in Figure **??**.

Pros and Cons of Batch Learning and its update:

- Easy
- time consuming
- requires a lot of computing resources

Online or incremental learning (Figure ?? and Figure ??)

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

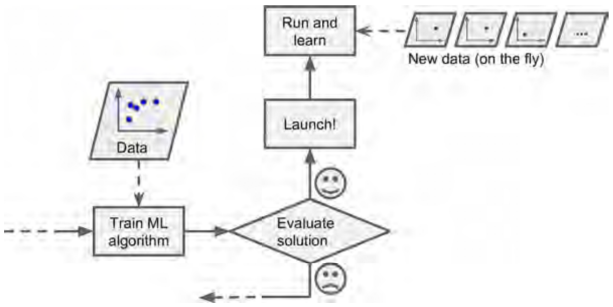

Figure 13: Online learning.

Online learning algorithms can also be used to train systems 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 1-14).

Online learning systems must adapt to changing data.
The ability of adaptation is determined by the learning rate, usually denote by λ or $\eta \in [0, 1]$:

- Large $\lambda \Longrightarrow$ rapid adaptation to new data; quick forgetting the old data (you don't want a spam filter to flag only the latest kinds of spam it was shown).

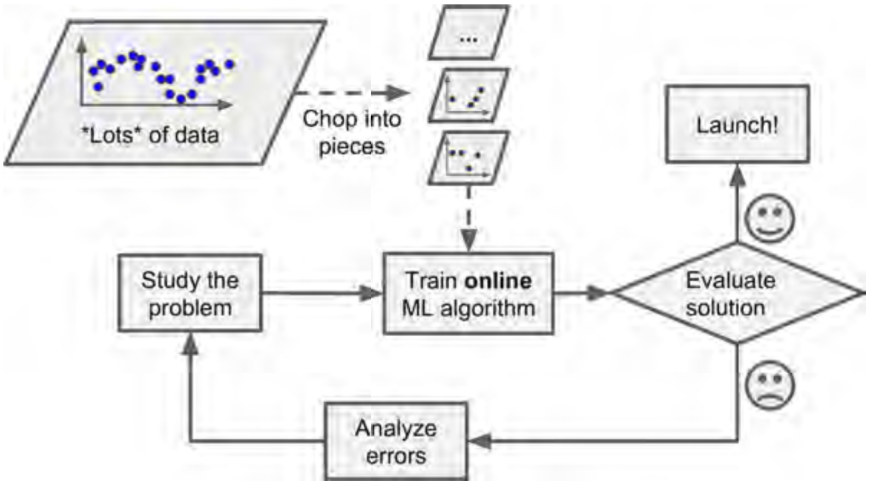

Figure 14: Using online learning to handle huge datasets.

- Small $\lambda \Longrightarrow$ the system will have more inertia: it learn more slowly; less sensitive to noise in the new data or to sequences of non-representative data points.

Challenge: if bad data (e.g., malfunctioning sensor on a robot, or from someone spamming a search engine to try to rank high in search) is fed to the system, the system's performance will gradually decline.

Possible solution:

- Monitor the data (use *anomaly detection*) and stop learning if needed
- Possibly reverting to a previous state of the ML system

Instance-Based Learning (Figure ??)

- (Most) trivial form of learning is simply to learn by heart.
- It can recognize only those instances which are *identical* to those used to train

Replace *identity* by *similarity* but this brings up the issue of how to measure similarity (e.g, words in common of two texts).

Model-Based Learning (Figure ??)

- Build a model from the training examples
- Use the model to make *predictions*

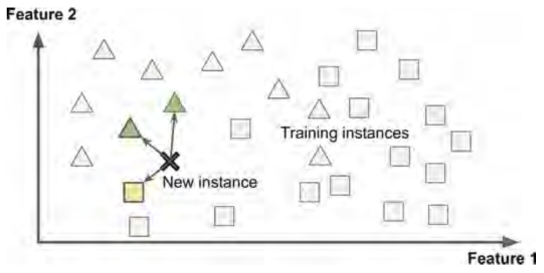

Figure 15: Instance-based learning.

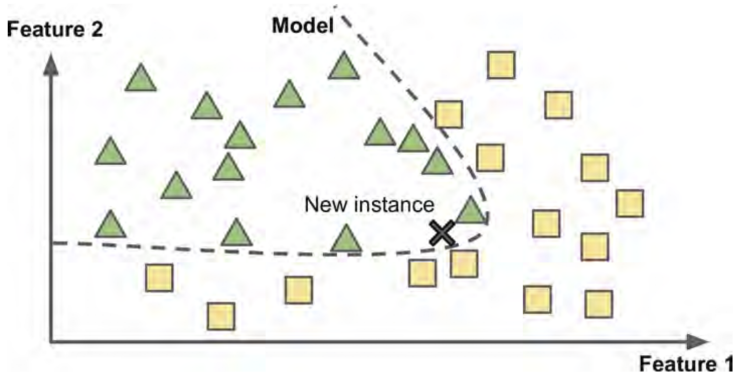

Figure 16: Model-based learning.

Example 1 *Consider the data shown in Table* **??***:*

Figure **??** *shows data from some random countries. A trend can be noticed from Figure* **??***: it looks like life satisfaction goes up more or less linearly as the country's GDP per capita increases*[4].

$\Longrightarrow$ **Model selection:** *model life satisfaction as a linear function of GDP per capita*

$$life\_satisfaction = \theta_0 + \theta_1 \times GDP\_per\_capita, \tag{1}$$

where θ_0 and θ_1 are the parameters of the model.

Usually, we collect the parameters as a vector $\theta = (\theta_0, \theta_1)$. Different values of these parameters produce different linear functions, as shown in Figure **??**.

To decide the model, i.e. the values of the parameters θ_0, θ_1, we must define a criterion with respect to which these values will be selected.

This is done via a fitness functions*, which evaluates how* good *the model is; or a* cost functions *which evaluates how* bad *the model is.*

Here we have then (linear, in this case) regression: the best parameter values are $\theta_0 = 4.85$ and $\theta_1 = 4.91 \times 10^{-5}$, which yields the linear function shown in Figure **??**.

[4]Ocham's razor: Prefer simple explanations to more complex.

Table 1: Does money bring happiness?

Country	GDP per capita (USD)	Life Satisfaction
Hungary	12,240	4.9
Korea	27,195	5.8
France	37,675	6.5
Australia	50,962	7.3
US	55,805	7.2

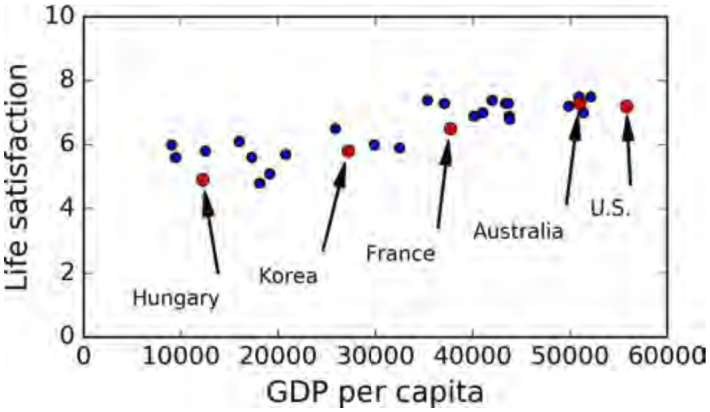

Figure 17: A few random countries.

Summary

- Study the data.
- Select a model.
- Train it on the training data (i.e., the learning algorithm searched for the model parameter values that minimize a cost function).
- Apply 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 vs "bad"algorithms

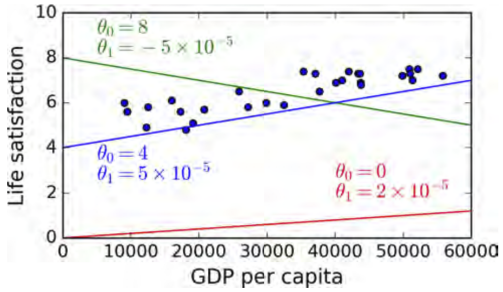

Figure 18: A few possible linear models.

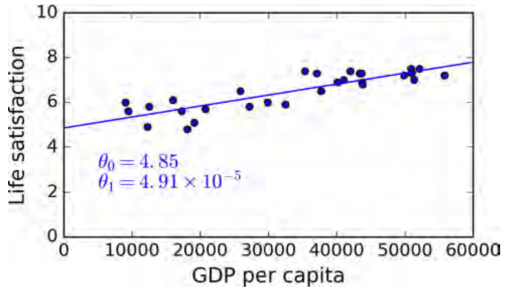

Figure 19: The linear model that fits the training data best.

Bad data

Insufficient Quantity of Training Data

- A lot of training data is needed, from thousands to millions of examples
- 2001 Microsoft paper (Michele Banko and Eric Brill): 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 (as you can see in Figure **??**).

Non-representative Training Data

- Representative training data $\Longrightarrow$ good generalization (regardless of the type of training – instance-based learning or model-based learning).
- For example, the set of countries used earlier for training the linear model was not perfectly representative; a few countries were missing. Figure **??** shows what the data looks like when you add the missing countries.

Sampling the training set is very important:

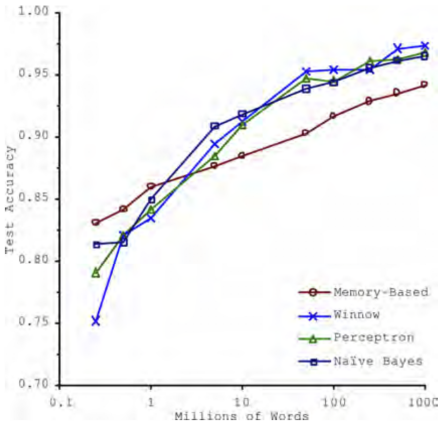

Figure 20: The importance of data versus algorithms.

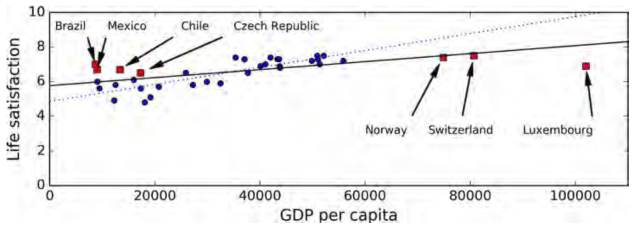

Figure 21: A more representative training sample.

- Too small training data set $\Longrightarrow$ Sample noise: non-representative data as a result of chance
- Sample bias: flawed sampling method even from a large data set.

Poor-Quality Data

- If some instances are clearly outliers, it may help to simply discard them or try to fix the errors manually.
- If some instances are missing a few features (e.g., 5% of your customers did not specify their age), must decide whether ignore this attribute altogether, ignore these instances, fill in the missing values (e.g., with the median age), or train one model with the feature and one model without it, and so on.

Irrelevant Features

- Prevent "garbage in, garbage out"

- An ML r system will only be capable of learning if the training data contains enough relevant features and not too many irrelevant ones.
- Good feature set $\Longrightarrow$ critical part of the success of a ML project: feature engineering
- 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 algorithms can help).
- Creating new features by gathering new data.

Overfitting the Training Data

The model performs well on training data (good modeling power) bad on test data (bad generalization power). See example of overfitting in Figure **??**, where the GDP data is fitted by a polynomial of degree > 2. Such a polynomial has more parameters, that is, it is more complex.

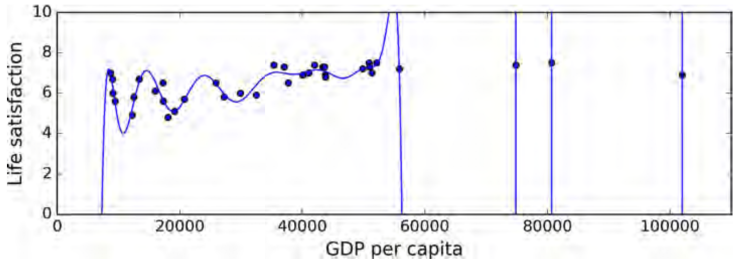

Figure 22: Overfitting the training data

Overfitting is reduced by regularization. The linear model with θ_0 and θ_1 has two degrees of freedom. Either one could be 0:

- If $\theta_0 = 0$ the line always passes through the origin: all we can do is to change its slope
- $\theta_1 =$, the model line is horizontal: all it could do is move the line up or down to get as close as possible to the training instances, so it would end up around the mean. θ_0 will end up being the mean of the training data.

Regularization is illustrated in Figure **??**. It forces the model to have a smaller slope (smaller θ_1). It fits the training data less, but the test data more.

Important Note 1 *Need to remember:*

- *The modeling power (prediction on the training data) and generalization power (prediction on the test data) are two very important concepts in ML*

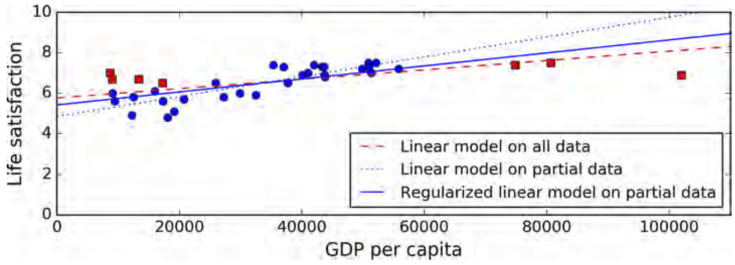

Figure 23: Regularization reduces the risk of overfitting.

- *We start with a good model (which works well on the training data). Recall the notion of Version Space, i.e., the collection of all hypotheses consistent with the examples of the concept*
- *Principle underlying ML: give up modeling power in order to gain generalization power.*