95-828: Machine Learning for Problem Solving **Spring 2019**

Lecture 1: Machine Learning: Basic Concepts

Instructor: Leman Akoglu

Learning objectives for this lecture are:

- Understand basic machine learning concepts and workflow
- Distinguish between different types of machine learning tasks
- Take a concrete task and cast it as a learning problem, with a formal notion of input space, features, and output space
- Review state of the art applications of machine learning in the real world
- Get familiar with the fundamental concept of learning: generalization (vs. overfitting)

1.1 What is Machine Learning?

Machine learning is the systematic study of algorithms and systems that improve their knowledge or performance with experience (i.e., more data).

Machine learning stands on finding and exploiting the patterns in the data. Often humans cannot pin down the data generating mechanism mathematically, but they have data (**examples** or **observations**) on it. The machine learning system figures out what the humans want based on those examples.

For instance, a bank may wish to predict whether it is safe to approve an incoming loan application based on the applications they have received in the past. This prediction could be based on many factors: the applicant's age, salary, current debt, credit score, requested loan amount, etc. The desired outcome in this case is binary: whether to approve or not.

In formal terms, we can list the components of learning as follows:

- **Input example** (a.k.a. **observation**, **instance**): i (loan application)
- **Features** (a.k.a. **attributes, variables, covariates, predictors**): $\mathbf{x}_i$ (factors like age, salary, etc. derived from example i)
- **Output** (a.k.a. **response, target, label**): y_i (good/bad customer)
- (Unknown) Target function: $f : \mathcal{X} \to \mathcal{Y}$ (ideal credit approval formula)
- **Training data**: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_i, y_i), \ldots, (\mathbf{x}_n, y_n)$ (historical examples)
- Hypothesis set $\mathcal{H} = \{h\}$ (family of machine learning models)
- Learning algorithm A (optimization or search algorithm)

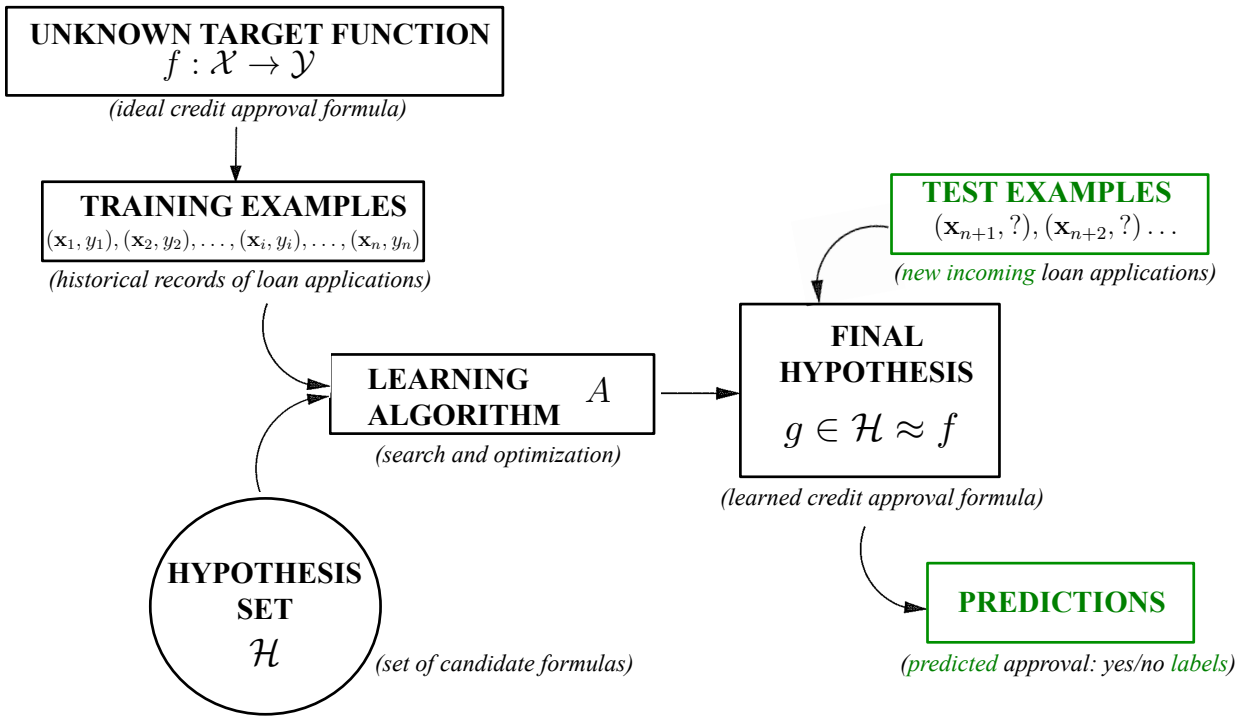


Figure 1.1: Basic framework of the learning problem

- Hypothesis function $g \in \mathcal{H}$ (**classifier or regressor**): $g : \mathcal{X} \to \mathcal{Y}$ (loan approval formula to be used)

- **Test data**: $(\mathbf{x}_{n+1}, ?), (\mathbf{x}_{n+2}, ?) \ldots$

There is the input $\mathbf{x}$; application information used to make a loan decision. There is the ideal but unknown target function f. Given a **training dataset** $\mathcal{D}$ of input-output examples where $y_i = f(\mathbf{x}_i)$, for $i = 1, \ldots, n$, inputs corresponding to previous/historical applications, and the correct loan decision for them in hindsight. Finally there is a learning algorithm A that used $\mathcal{D}$ to pick a formula (model) g that approximates f. A chooses g from a family of candidate formulas, which we call the hypothesis set $\mathcal{H}$. For instance, $\mathcal{H}$ could be the family of all linear formulas, as we will introduce in the next lecture.

When a new customer applies for a loan, the bank will base its decision on g (not on f, which remains unknown). The decision will be good to the extent g replicates f faithfully. To achieve that, A chooses g that best matches f on the training examples, with the hope that it will continue to match f on newcoming **test examples** $\mathbf{x}_{n+1}, \mathbf{x}_{n+2}, \ldots$. The extent to which this hope is justified is called **generalization**, which we will revisit shortly.

1.1.1 Canonical Learning Tasks

The basic premise of learning is to use a set of examples to uncover an underlying process. It is a broad premise, and hard to fit into a single framework. As a result, different learning paradigms exist, varying in the nature of the input and output data.

In this course we will focus on two fundamental paradigms: Supervised and Unsupervised learning. There are other complex settings, including reinforcement learning, semi-supervised learning, active learning, structured prediction, and so on, which you may learn at more advanced machine learning courses.

Briefly put, reinforcement learning is about an agent learning to interact with an environment, with some ultimate goal. The environment could be a Go board, and the goal to win the game (in 2016, AlphaGo beat the world champion in Go). For self-driving cars, the environment could be roads, sensed by cameras and laser sensors, and the goal would be to get from A to B quickly and safely. Semi-supervised learning aims to learn from only a small set of labeled examples combined with lots of unlabeled examples. Active learning carefully selects unlabeled examples to query an oracle for labels, with a goal to learn quickly from as few queries as possible.

- **Supervised Learning**: (*learning with a teacher*) This is the learning paradigm we discussed thus far, which is illustrated in Figure 1.1.

  The input data to supervised learning is of the (features, output) form

  $$\mathcal{D} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{n} \overset{\text{i.i.d}}{\sim} P(\mathbf{x}, y) \ .$$

  Generally, we assume these examples are drawn **i.i.d.** from some *unknown* joint distribution $P(\mathbf{x}, y)$, also called the **data generating distribution**. We say they are drawn **i.i.d.**, which stands for **independent and identically distributed**. In other words, $(\mathbf{x}_i, y_i) \in \mathcal{D}$ pairs are independent and all come from the same distribution P.

  We will focus on two types of supervised learning problems:

  1. **Classification** refers to the setting where the output is **discrete** (categorical or nominal).
     - **Binary classification**: Only two output values (yes/no) are of interest, i.e. $y \in \{0, 1\}$; e.g., spam/'ham' emails, approved/declined applications, positive/negative sentiment, etc.
     - **Multi-class classification**: More than two output values are of interest, i.e. $y \in \{1, \ldots, C\}, C > 2$; e.g., face or activity recognition, predicting movie genres (comedy, drama, action, etc.), news categories (politics, entertainment, travel, religion, etc.), potential diseases for a patient with certain symptoms and medical history

2. **Regression** refers to the setting where the output is **continuous** (or real). e.g., predicting tomorrow's value of a stock, temperature, someone's salary or age.

- **Unsupervised Learning**: (*learning without a teacher*) In this case the input data contains only the features $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^n \overset{\text{i.i.d}}{\sim} P(\mathbf{x})$. The goal is to find structure and patterns in the data, such as whether the data clusters in several groups, whether data points lie on a low-dimensional subspace, or find ways to represent the distribution compactly.

  We will learn three types of unsupervised learning problems:

  1. **Clustering**: grouping input examples into clusters based on similarity. e.g., clustering books of similar topics
  2. **Density estimation**: estimating the probability of any $\mathbf{x}$. e.g., population density in a given location.
  3. **Dimensionality reduction**: identifying latent factors (or dimensions) that capture the underlying structure in the data. e.g., topic representation of documents

1.1.2 What does it mean to learn?

In order to develop learning machines, we must know what learning actually means, and how to determine success (or failure).

Consider the scenario where Bob has just begun taking a course on machine learning. At the end of the course, he will be expected to have **"learned"** all about this topic. A common way of gauging whether or not he has learned is for the teacher, Alice, to give him an exam. He has done well at learning if he answers the questions on the exam correctly.

What makes a reasonable exam? It would be unfair to give an exam on History of Pottery, since Bob has *no prior experience* with it and his performance would *not* be *representative* of his learning. On the other hand, if the exam only asks questions that Alice answered in class, that is also a bad exam: we would expect Bob give correct answers, but this would not demonstrate that he has learned—he would simply be recalling his past experience.

In the former case, we are expecting it to generalize beyond its experience, which is unfair. In the latter case, we are not expecting it to generalize at all. From this scenario, we can derive two rules of thumb for the notion of learning:

- There should be a strong relationship between the data that our algorithm sees at training time and the data it sees at test time. Formally, train and test data should come from the same **data generating distribution**, which is simply a probability distribution over input/output pairs.
- The performance of the learning algorithm should be measured on **unseen test** data.

1.1.3 Generalization vs. Overfitting

Generalization is perhaps the most central concept in machine learning. What is desired is that Bob (the ML system) **observes specific examples** from the course, and then has to answer **related, but new** questions on the exam. This tests whether Bob (the system) has the ability to **generalize.**

Consider the following regression prediction function g:

$$g(\mathbf{x}) = \begin{cases} y_i, & \text{if } \mathbf{x} = \mathbf{x}_i \text{ for } i = 1, \ldots, n \\ \text{any random value}, & \text{otherwise} \end{cases}$$

It would perform perfectly on training examples with zero error, as it simply predicts their exact output. However, it would perform extremely poorly on unseen test examples. This is because this predictor has *not* **learned to generalize**, but rather *memorized* (**overfit**) to the training dataset.

Consider the following classification scenario of predicting the (binary) output label of whether or not to play outside, based on the 3 (nominal) weather features. The training data contains the following 8 examples:

Class	Outlook	Temperature	Windy?
Play	Sunny	Low	Yes
No play	Sunny	High	Yes
No play	Sunny	High	No
Play	Overcast	Low	Yes
Play	Overcast	High	No
Play	Overcast	Low	No
No play	Rainy	Low	Yes
Play	Rainy	Low	No

The task is to assign a label to the following test example:

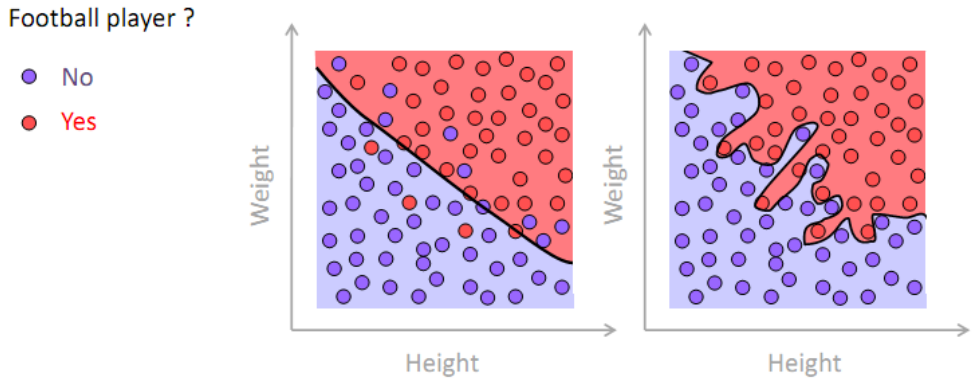
Class	Outlook	Temperature	Windy?
???	Sunny	Low	No

As you can notice, the test example is not seen in the training data. The goal of learning (classification in this case) is to generalize beyond the training data. One way to achieve this objective is to produce probabilistic predictions, that is $P(y|\mathbf{x}_{test}, \mathcal{D})$ which is a probability distribution over class labels. We will learn various classifiers throughout the course.

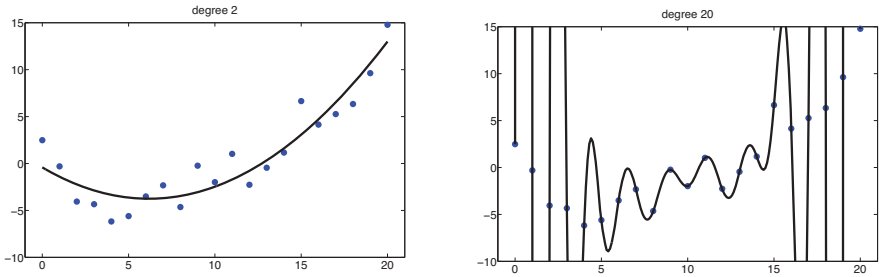
In general, learning very complex or *highly flexible* classifiers or regressors run the risk of overfitting to the training data. For instance, see the examples in Figure 1.2 for (a) binary classification, and (b) polynomial regression. On the left is shown the "just right" model, and on the right we show a model that overfit. Overfitting occurs due to trying to model every minor variation in the input, which is not a good idea since such variation is more

likely to be noise than true signal. As you may notice, the overfit models are unlikely to perform well for unseen test examples.

Later in the course, we will learn about **model selection**, where we have to choose between models with *different degrees of flexibility*. We will discuss the best practices such as **regularization** and **cross validation** for learning the "just right" models from data.



(a) Binary classification



(b) Polynomial regression: poly-fits of degrees (left) 2 and (right) 20 onto 21 training points. (Based on Figure 1.18 from Kevin Murphy, Machine Learning, 2012)

Figure 1.2: Models shown on the left are more likely to **generalize** to unseen test examples than models on the right that **overfit** to training data.

Different ML algorithms have different strengths and weaknesses, and can perform well on different datasets. When presented with a data analysis/ML problem, understanding the trade-offs in terms of *accuracy-speed-complexity-interpretability* among different ML models is necessary in order for you to determine which one to apply, and be able to diagnose its success or failure. When approaching a new problem it is always helpful to have many tools in your toolbox.