Fundamentals of Machine Learning

Daniel Yukimura

yukimura@impa.br

August 22, 2018

What is Machine Learning?

- Machine Learning (ML) studies systems that are trained from data rather than being explicitly programmed.
- More formally what is under study is the process of inductive inference which can be roughly described as:
  1. Observe a phenomenon.
  2. Construct a model of that phenomenon.
  3. Make predictions using this model.

What is Machine Learning?

- Machine Learning (ML) studies systems that are trained from data rather than being explicitly programmed.
- More formally what is under study is the process of inductive inference which can be roughly described as:
  1. Observe a phenomenon.
  2. Construct a model of that phenomenon.
  3. Make predictions using this model.

What is Machine Learning?

- Machine Learning (ML) studies systems that are trained from data rather than being explicitly programmed.
- More formally what is under study is the process of inductive inference which can be roughly described as:
  1. Observe a phenomenon.
  2. Construct a model of that phenomenon.
  3. Make predictions using this model.

What is Machine Learning?

- Machine Learning (ML) studies systems that are trained from data rather than being explicitly programmed.

- More formally what is under study is the process of inductive inference which can be roughly described as:
  1. Observe a phenomenon.
  2. Construct a model of that phenomenon.
  3. Make predictions using this model.

What is Machine Learning?

- Machine Learning (ML) studies systems that are trained from data rather than being explicitly programmed.
- More formally what is under study is the process of inductive inference which can be roughly described as:
  1. Observe a phenomenon.
  2. Construct a model of that phenomenon.
  3. Make predictions using this model.

What is Machine Learning?

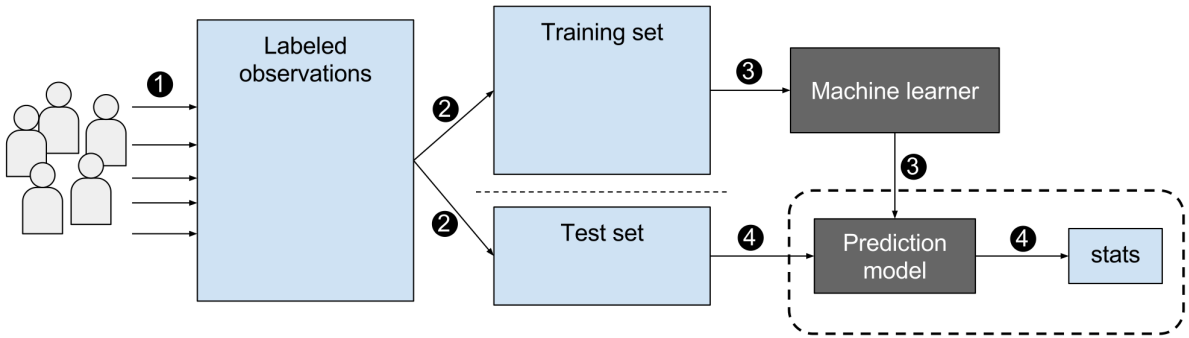
- Machine Learning (ML) studies systems that are trained from data rather than being explicitly programmed.

- More formally what is under study is the process of inductive inference which can be roughly described as:
  1. Observe a phenomenon.
  2. **Construct a model of that phenomenon.**
  3. Make predictions using this model.

  The goal of Machine Learning is to *automate* this process.
  (and the goal of Learning Theory is to *formalize* it)

What is Machine Learning?

Machine Learning represents a set of methods that automatically extract **features** from data in order to solve **prediction** tasks like:



- **Forecasting** (e.g. Energy Demand, Finances, Earthquakes)
- **Classification** (e.g. Cancer Diagnosis, Credit Risk Assessment)
- **Detecting Anomalies** (e.g. Security, Frauds, Epidemics, Virus Mutations)
- **Decision Making** (e.g. Robotics, Trading, AI)
- etc... Recommendation Systems, Self-driving cars, Machine Translation, Virtual Assistants,...

Learning from data:

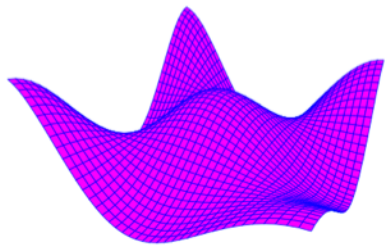
- How do we transform this concept of **learning** into an **explicit/practical** set of steps?
- What are the factors involved?

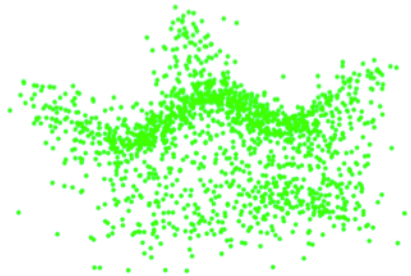
- **Statistical Inference** (or a version of it) is how you do it -

Remark: For tasks like Reinforcement Learning different frameworks might work better.

How do we perceive data?

1 - Observing the phenomenon





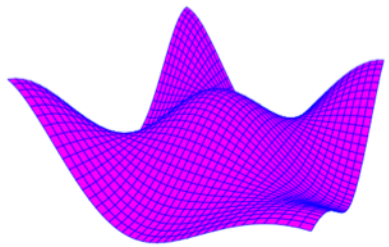
- In Statistics we call the set where our data lives in as **Population**.
- This set is represented as a probability space (ref.)

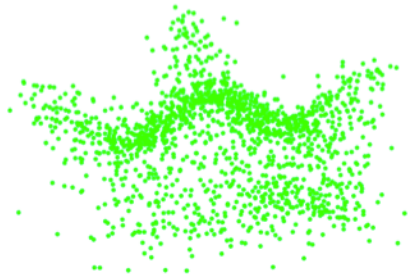
$$(\mathcal{Z}, \mathcal{Z}, p).$$

- Our data set $\mathcal{D} = \{Z_i\}_{i=1}^{n} \subseteq S$ is a collection of i.i.d. random variables $Z_i \sim p$.

How do we perceive data?

1 - Observing the phenomenon





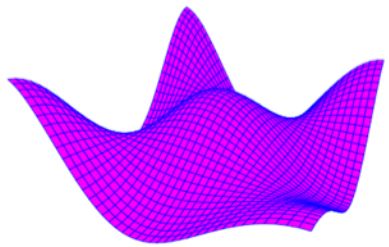
- In Statistics we call the set where our data lives in as **Population**.
- This set is represented as a probability space (ref.)

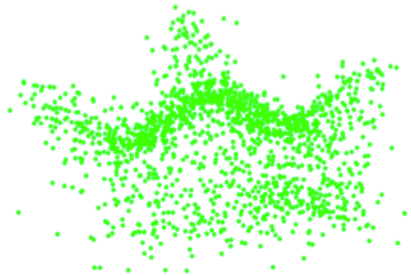
$$(\mathcal{Z}, \mathfrak{Z}, p).$$

- Our data set $\mathcal{D} = \{Z_i\}_{i=1}^{n} \subseteq S$ is a collection of i.i.d. random variables $Z_i \sim p$.

How do we perceive data?

1 - Observing the phenomenon





- In Statistics we call the set where our data lives in as **Population**.
- This set is represented as a probability space (ref.)

$$(\mathcal{Z}, \mathfrak{Z}, p).$$

- Our data set $\mathcal{D} = \{Z_i\}_{i=1}^{n} \subseteq S$ is a collection of i.i.d. random variables $Z_i \sim p$.

What kind of problem?

1 - Observing the phenomenon

When designing learning algorithms, is common to distinguish between two main types:

- **Supervised Learning:** We are interested in the underlying predictive relationship between our labelled data set $\{(X_i, Y_i)\}_{i=1}^n$.
- **Unsupervised Learning:** Find *interesting structure* in unlabelled data. Which can mean estimating the density $p(Z)$ itself.

What kind of problem?

1 - Observing the phenomenon

When designing learning algorithms, is common to distinguish between two main types:

- **Supervised Learning:** We are interested in the underlying predictive relationship between our labelled data set $\{(X_i, Y_i)\}_{i=1}^n$.
- **Unsupervised Learning:** Find *interesting structure* in unlabelled data. Which can mean estimating the density $p(Z)$ itself.

Supervised Learning

2 - Constructing (learning) the model

We now focus on the **supervised** type:

Observation #	fixed acidity	density	...	res. sugar	...	pH	Quality
1	7.4	0.9978	...	1.9	...	3.51	5
2	7.8	0.9968	...	2.6	...	3.2	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
232	5.2	0.9927	...	1.6	...	3.54	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	
1599	6	0.99549	...	3.6	...	3.39	6

Table: Wine quality data set [link]

By observing a collection of examples $\{(X_i, Y_i)\}_{i=1}^n$ we want to construct a **predictor** $\hat{y} : \mathcal{X} \to \mathcal{Y}$ that makes good predictions of Y given X (*on samples that are likely to appear in practice*).

Supervised Learning

2 - Constructing (learning) the model

We now focus on the **supervised** type:

Observation #	fixed acidity $X(1)$	density $X(2)$	... ...	res. sugar $X(j)$	... ...	pH $X(d)$	Quality Y
1	7.4	0.9978	...	1.9	...	3.51	5
2	7.8	0.9968	...	2.6	...	3.2	5
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
232	5.2	0.9927	...	1.6	...	3.54	7
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
1599	6	0.99549	...	3.6	...	3.39	6

Table: Wine quality data set [link]

By observing a collection of examples $\{(X_i, Y_i)\}_{i=1}^n$ we want to construct a **predictor** $\hat{y} : \mathcal{X} \to \mathcal{Y}$ that makes good predictions of Y given X (*on samples that are likely to appear in practice*).

How do we deal with uncertainty?

2 - Constructing (learning) the model

To obtain **generalization** and deal with **noise sensitivity** is often necessary to model the problem in the probabilistic setting as

$$\hat{y}(x) = \text{“best guess”} = \operatorname*{argmax}_{y \in \mathcal{Y}} p(y \mid x). \tag{1}$$

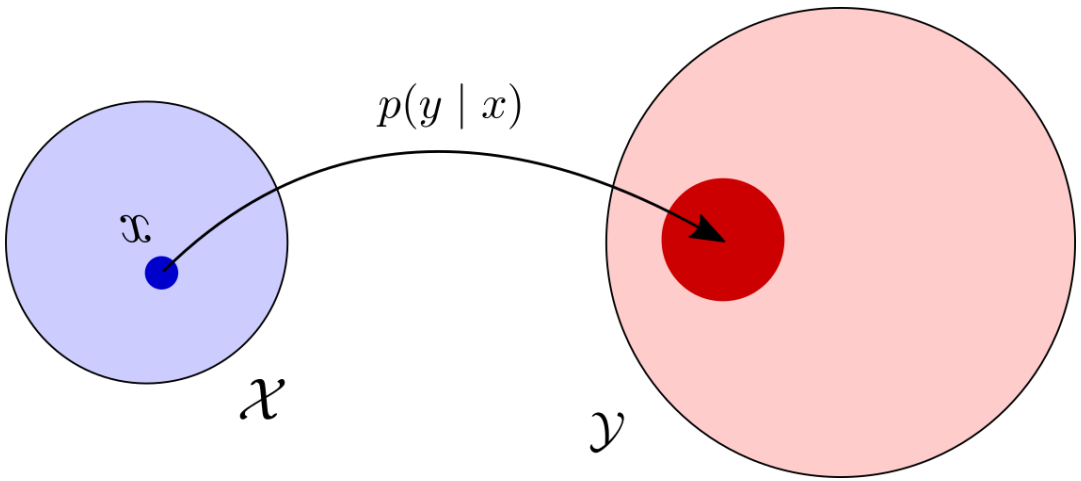


Figure: For each point $x \in \mathcal{X}$ there is a distribution of likely outputs $p(y \mid x)$.

Supervised Learning

2 - Constructing (learning) the model

The *Naive* objective is usually unfeasible - *No Free Lunch* [link]) - How do we build a **learning algorithm** that given a set of data points can learn a **consistent estimator**?

$$\hat{y} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \rightarrow \mathcal{F}. \tag{2}$$

Where $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ is the **hypothesis class**, a set of possible predictors.

The choice of $\mathcal{F}$ is the first step into solving this question.

Supervised Learning

2 - Constructing (learning) the model

The *Naive* objective is usually unfeasible - *No Free Lunch* [link]) - How do we build a **learning algorithm** that given a set of data points can learn a **consistent estimator**?

$$\hat{y} : \bigcup_{n=1}^{\infty} (\mathcal{X} \times \mathcal{Y})^n \to \mathcal{F}. \tag{3}$$

Where $\mathcal{F} \subseteq \mathcal{Y}^{\mathcal{X}}$ is the **hypothesis class**, a set of possible predictors.

The choice of $\mathcal{F}$ is the first step into solving this question.

Supervised Learning

Example 1: Linear Regression

Salary as a function of the number of years of experience

YearsExperience	Salary
1.1	39343
1.3	46205
1.5	37731
2	43525
2.2	39891
2.9	56642
3	60150
3.2	54445
3.2	64445
3.7	57189
3.9	63218

Goal: Finding the best $\hat{y}(\cdot) \in \mathcal{F}$ that can predict well new data.

$$\hat{y}(x) = ax + b \qquad \leftrightarrow \qquad \mathcal{F} = \{\text{“all linear models”}\} \approx \mathbb{R}^2$$

In this case this is the same as finding the best parameters $(a, b) \in \mathbb{R}^2$, such that the line fits well the data.

from this tutorial

Supervised Learning

Parameter estimation

Parameters are real values that control the behavior of a model.

$$\mathcal{F} = \{f_\theta : \mathcal{X} \to \mathcal{Y} \mid \theta \in \Theta\}$$

And therefore the problem translates into finding the best parameter $\hat{\theta}$ given a data set.

Question:

How do we find a good model?

Empirical Risk Minimization

Loss function

- We choose a **loss function** $\ell : \mathcal{F} \times \mathcal{Z} \to \mathbb{R}^+$ (or $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$).
- Which should quantify the loss of considering hypothesis $f \in \mathcal{F}$ and observing example $z \in \mathcal{Z}$.
- Usually $\mathcal{Y}$ possess a metric and we take the loss $\ell(\hat{y}, y)$ to be the distance between y and $\hat{y}(x)$.

Empirical Risk Minimization

Risk Function

Def: We define the (Population) **risk** associated to $f \in \mathcal{F}$.

$$R(f) = \mathbb{E}_{(X,Y)\sim p}[\ell(f(X), Y)]. \tag{4}$$

From this analogy, our ideal estimator is therefore, the one of minimum risk

$$f^* = \operatorname*{argmin}_{f\in\mathcal{F}} R(f). \tag{5}$$

Remark: In a parametric setting we define the risk on the parameter $\theta \in \Theta$

$$R(\theta) = \mathbb{E}_{(X,Y)\sim p}[\ell(\hat{y}(X, \theta), Y)]. \tag{6}$$

Empirical Risk Minimization

Regression setting ($\mathcal{Y} = \mathbb{R}$)

Using the **squared error loss** $\ell(\hat{y}, y) = (\hat{y} - y)^2$, the associated risk becomes

$$R(f) = \mathbb{E}(f(X) - Y)^2. \tag{7}$$

The minimizer in this case is the conditional expectation

$$f^*(x) = \mathbb{E}[Y \mid X = x]. \tag{8}$$

Empirical Risk Minimization

Empirical Risk

Since the distribution p is unknown we can't actually compute the risk function in practice.

- **Def:** We define the **empirical risk** as

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f, Z_i) \tag{9}$$

  where the $Z_i = (X_i, Y_i)$ are the examples in our data set.

- We choose an estimator $\hat{y}_n : \mathcal{Z}^n \in \mathcal{F}$ that minimizes the empirical risk

$$\hat{y}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_n(f) \tag{10}$$

  We call this strategy **empirical risk minimization (ERM)**.

Empirical Risk Minimization

Empirical Risk

Since the distribution p is unknown we can't actually compute the risk function in practice.

- **Def:** We define the **empirical risk** as

$$R_n(f) = \frac{1}{n} \sum_{i=1}^{n} \ell(f, Z_i) \tag{9}$$

  where the $Z_i = (X_i, Y_i)$ are the examples in our data set.

- We choose an estimator $\hat{y}_n : \mathcal{Z}^n \in \mathcal{F}$ that minimizes the empirical risk

$$\hat{y}_n = \operatorname*{argmin}_{f \in \mathcal{F}} R_n(f) \tag{10}$$

  We call this strategy **empirical risk minimization** (**ERM**).

Linear Regression

- We assume $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ and a ypothesis class consisting only of linear models

$$\mathcal{F} = \left\{ \hat{y} \in \mathcal{Y}^{\mathcal{X}} : \exists \theta \in \mathbb{R}^{d+1} \text{ s.t. } \hat{y}(x) = \hat{y}(x, \theta) = \theta_0 + \sum_{i=1}^{d} \theta_i x_i, \forall x \in \mathcal{X} \right\} \tag{11}$$

- Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ we are interested in finding a parameter vector $\hat{\theta}$ that best approximates the relation of y and x by a linear model.

Linear Regression

- We assume $\mathcal{X} = \mathbb{R}^d$, $\mathcal{Y} = \mathbb{R}$ and a ypothesis class consisting only of linear models

$$\mathcal{F} = \left\{ \hat{y} \in \mathcal{Y}^{\mathcal{X}} : \exists \theta \in \mathbb{R}^{d+1} \text{ s.t. } \hat{y}(x) = \hat{y}(x, \theta) = \theta_0 + \sum_{i=1}^{d} \theta_i x_i, \forall x \in \mathcal{X} \right\} \tag{11}$$

- Given a dataset $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ with $x_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ we are interested in finding a parameter vector $\hat{\theta}$ that best approximates the relation of y and x by a linear model.

Linear Regression

- We see that ERM is equivalent to the method of **least squares**:

$$\begin{aligned} R_n(\theta) &= \sum_{i=1}^{n} (y_i - \hat{y}(x_i, \theta))^2 \\ &= \sum_{i=1}^{n} (y_i - \theta_0 - \sum_{j=1}^{d} \theta_j x_{ij})^2. \end{aligned} \tag{12}$$

- We can rewrite this sum using matrix notation as

$$R_n(\theta) = (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta). \tag{13}$$

Linear Regression

- We see that ERM is equivalent to the method of **least squares**:

$$\begin{aligned} R_n(\theta) &= \sum_{i=1}^{n} (y_i - \hat{y}(x_i, \theta))^2 \\ &= \sum_{i=1}^{n} (y_i - \theta_0 - \sum_{j=1}^{d} \theta_j x_{ij})^2. \end{aligned} \tag{12}$$

- We can rewrite this sum using matrix notation as

$$R_n(\theta) = (\mathbf{y} - \mathbf{X}\theta)^T (\mathbf{y} - \mathbf{X}\theta). \tag{13}$$

Linear Regression

- We can minimize this quadratic equation by differentiating it with respect to θ

$$\frac{\partial}{\partial \theta} R_n(\theta) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\theta) \tag{14}$$

$$\frac{\partial^2}{\partial^2 \theta} R_n(\theta) = -2\mathbf{X}^T\mathbf{X}. \tag{15}$$

- When $\mathbf{X}^T\mathbf{X}$ is positive definite we obtain the unique solution

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{16}$$

Linear Regression

- We can minimize this quadratic equation by differentiating it with respect to θ

$$\frac{\partial}{\partial \theta} R_n(\theta) = -2\mathbf{X}^T(\mathbf{y} - \mathbf{X}\theta) \tag{14}$$

$$\frac{\partial^2}{\partial^2 \theta} R_n(\theta) = -2\mathbf{X}^T\mathbf{X}. \tag{15}$$

- When $\mathbf{X}^T\mathbf{X}$ is positive definite we obtain the unique solution

$$\hat{\theta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}. \tag{16}$$

Generalization

- We emphasize that training learning models differ from a pure traditional optimization task. In most cases the performance measure its defined over a **test set** of examples, that are not experienced during training, this is a way of determining if our model is generalizing for instances outside of the data set.
- One way of *measuring* generalization is by the **excess expected risk**

$$R(f_{\mathcal{D}}) - R(f^*)$$

or even, since $R(f_{\mathcal{D}})$ is random,

$$\mathbb{E}_{\mathcal{D}}[R(f_{\mathcal{D}}) - R(f^*)]$$

Generalization

- We emphasize that training learning models differ from a pure traditional optimization task. In most cases the performance measure its defined over a **test set** of examples, that are not experienced during training, this is a way of determining if our model is generalizing for instances outside of the data set.
- One way of *measuring* generalization is by the **excess expected risk**

$$R(f_{\mathcal{D}}) - R(f^*)$$

or even, since $R(f_{\mathcal{D}})$ is random,

$$\mathbb{E}_{\mathcal{D}}[R(f_{\mathcal{D}}) - R(f^*)]$$

Generalization

- **Consistency** can written as

$$\lim_{n \to \infty} \mathbb{E}_{\mathcal{D}}[R(f_{\mathcal{D}}) - R(f^*)] = 0 \tag{17}$$

- **Learning rates:** For all $\varepsilon > 0$ if $n \geq n(\varepsilon)$, then

$$\mathbb{E}_{\mathcal{D}}[R(f_{\mathcal{D}}) - R(f^*)] \leq \varepsilon$$

$n(\varepsilon)$ is called **sample complexity**.

Generalization

- **Consistency** can written as

$$\lim_{n \to \infty} \mathbb{E}_{\mathcal{D}}[R(f_{\mathcal{D}}) - R(f^*)] = 0 \tag{17}$$

- **Learning rates:** For all $\varepsilon > 0$ if $n \geq n(\varepsilon)$, then

$$\mathbb{E}_{\mathcal{D}}[R(f_{\mathcal{D}}) - R(f^*)] \leq \varepsilon$$

$n(\varepsilon)$ is called **sample complexity**.

Generalization

How to design a good learning algorithm?

- **Fitting:** An estimator should *fit* data well.
- **Stability:** An estimator should be stable, it should not change much if data change slightly

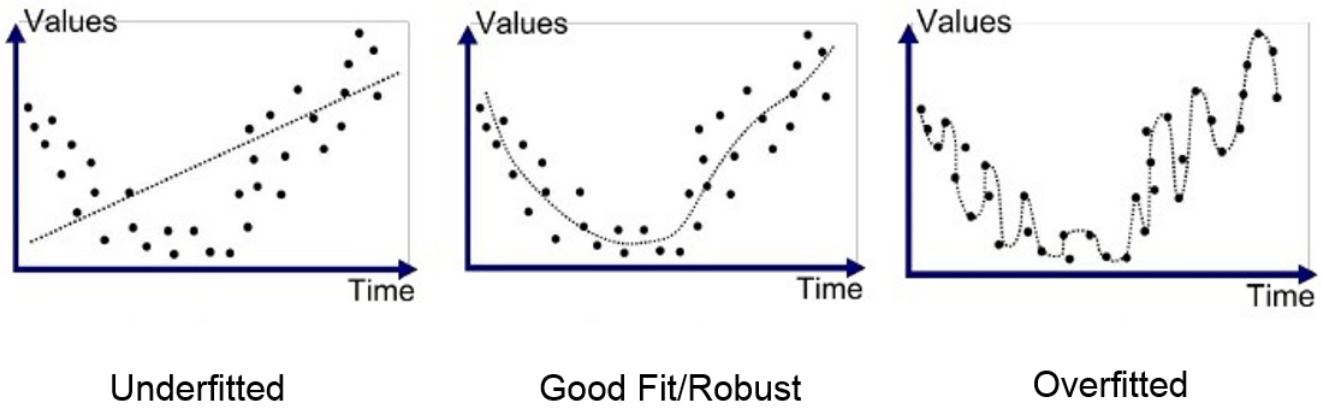
Generalization

How to design a good learning algorithm?

- **Fitting:** An estimator should *fit* data well.
- **Stability:** An estimator should be stable, it should not change much if data change slightly

Overfitting

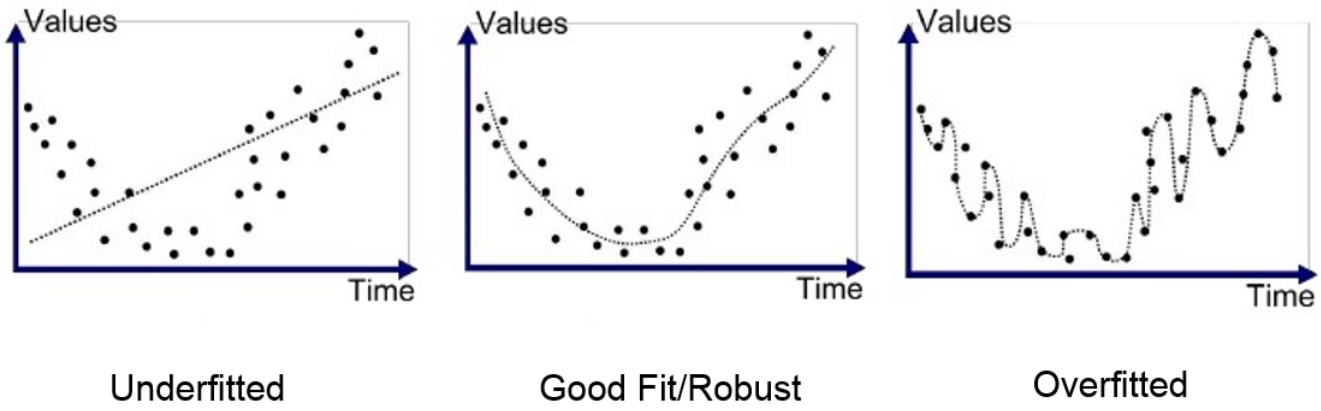
How to design a good learning algorithm?



- We say an algorithm **overfits** when it fits the data, but fails to capture the structure for generalization.
- We say the algorithm **underfits** when it is stable while disregarding the data.

Overfitting

How to design a good learning algorithm?



- We say an algorithm **overfits** when it fits the data, but fails to capture the structure for generalization.
- We say the algorithm **underfits** when it is stable while disregarding the data.

Capacity and Likelihood

- How do we decide between two hypothesis h and $\tilde{h}$?
  The principle of **Occam's razor** says we always prefer the *simplest* model that fits well the data.
- The **likelihood** of seeing the sample $\mathcal{D}$ of size n assuming the hypothesis h is

$$p(\mathcal{D} \mid h) = \left[\frac{1}{\text{size}(h)}\right]^n \tag{18}$$

  assuming that the samples are independent.