00 Introduction

October 15, 2021

Information: *Some basics concepts of Machine Learning.*

Written by: *Zihao Xu*

Last updated date: *Oct.15.2021*

1 Artificial Intelligence, Machine Learning, Deep Learning

1.1 Artificial Intelligence:

- A.I. is a very **broad** field.
- Turing test:
  - 'A human tries to tell the difference between a human correspondent and a machine one. When they can't, according to Turing, the machine should be judged intelligent.'
- In Oxford Dictionary
  - 'the theory and development of computers systems **able to perform tasks that normally require human intelligence**, such as visual perception, speech recognition, decision-making, and translation between languages'

1.2 Machine Learning

1.2.1 Definition

- A huge part in the concept of AI (Models, Algorithms, Theory) as well as useful tools for some AI applications (Computer Vision, Robotics).
- **Train an algorithm to reproduce answers from data.**
- Usually can be divided into **supervised learning** and **unsupervised learning**.

1.2.2 Dataset

- A dataset is a collection of many **examples**, which are sometimes called **data points**
- One common way of describing a dataset is with a **design matrix**
  - Containing a different example in each row
  - Each column corresponds to a different feature

1.2.3 Supervised Learning

- To estimate a **function between input and output** given a dataset containing **input-output examples**.
  - Examples: Regression, Classification, Filtering, …

1.2.4 Unsupervised Learning

- To **model the input data directly**, or in other words, understand the input data directly given a dataset containing many features
  - Examples: Clustering, Dimensionality Reduction, Density Estimation, Generative models, …
  - Motivation: Labeling data is always expensive (supervised) while gathering raw data is cheap (unsupervised).
  - Yann LeCun, Head of Facebook AI, 2016
    - 'AI systems today do not possess "common sense", which humans and animals acquire by observing the world, acting in it, and understanding the physical constraints of it. Some of us see unsupervised learning as the key towards machines with common sense.'

1.3 Common machine learning tasks

1.3.1 Classification

- Specify which of k categories some input belongs to
- Typically, produce a function $f : \mathbb{R}^n \to \{1, \cdots, k\}$.
  - Assigns a input $\mathbf{x}$ to a category identified by numeric code y
- In some cases. produce a probability distribution over classes
- One common example is object recognition

1.3.2 Regression

- Predict a numerical value given some input
- Output a function $f : \mathbb{R}^n \to \mathbb{R}$
- Similar to classification except that the output format is different

1.3.3 Transcription

- Observe a relatively unstructured representation of some kind of data and transcribe the information into discrete textual form
- One example is to produce a text in the form of a sequence of characters given a photograph containing an image of the text

1.3.4 Machine translation

- Given a sequence of symbols in some language, convert it into a sequence of symbols in another language

1.3.5 Structured output

- Any tasks where the output is a vector (or other data structure containing multiple values) with important relationships between the different elements
- One example is pixel-wise segmentation of images, where every pixel in an image is assigned to a specific category
- The program must output several values that are all tightly interrelated

1.3.6 Anomaly detection

- Sift through a set of events or objects and flags some of them as being unusual or atypical
- One example is credit card

1.3.7 Synthesis and sampling

- Generate new examples that are similar to those in the given data

1.3.8 Imputation of missing values

- Given a new example $\mathbf{x} \in \mathbb{R}^n$ but with some entries x_i of $\mathbf{x}$ missing, provide a prediction of the missing values

1.3.9 Denoising

- Given as input a corrupted example $\tilde{\mathbf{x}} \in \mathbb{R}^n$ obtained by an unknown corruption process from a clean example $\mathbf{x} \in \mathbb{R}^n$, predict the clean example $\mathbf{x}$ from its corrupted version

1.3.10 Density estimation

- Learn a function $p_{\text{model}} : \mathbb{R}^n \to \mathbb{R}$ where p_{model} can be interpreted as a probability density function on the space that the examples were drawn from

1.4 Deep Learning

1.4.1 Definition

- A particular successful **Machine Learning method based on deep sequences of neural networks**.
- Generally speaking, deep models are **sequential transformations learned from data**. If you use deep neural networks to define sequential transformations from input to output, then you do Deep Learning.

1.4.2 Successful Applications

- Any machine learning task:
  - Regression, Classification, Clustering, Dimensionality reduction, Density estimation, Filtering, …
- Natural language processing
- Reinforcement learning
  - self-driving cars, playing Atari games, AlphaGo
- Design of engineering systems

1.4.3 Why does Deep Learning work so well?

- **Basic theory**
  - The universal approximation theorem (Cybenko, 1989)
- **Deterministic Reason**

  - Advanced learning algorithms (back-propagation, stochastic gradient descent, etc.), advanced software (tensorflow, pytorch) and advanced hardware (GPU) allow larger model sizes and more complicated computation
  - The amount of available training data has increased
- **Popular in various applications**
  - Instead of handcrafting features in classical Machine Learning, Deep Learning let the deep model do all the feature engineering automatically.
  - Deep Learning can automatically learn **a hierarchy of representations of high-dimensional data**.

1.4.4 Problems of applying Deep Learning in engineering

- Many approaches are **black boxes** with no guarantees on convergence and performance.
- Deep Models would make errors if data are perturbed.

2 General Supervised Learning Process

2.1 Problem Definition

2.1.1 Mathematical Representation

$$y = f_{gt}(x)$$

- x: Input of the function
  - Called **features**, attributes, covariates or variables
  - Can be numeric, categorical, discrete or nominal
  - Examples:
    - [GRE scores, GPA, major]
    - An image
    - A sentence
    - A d-dimentional vector of numbers
- y: Output of the function
  - Called output, response, target, or **label**
  - Can be numeric, categorical, discrete or nominal
    - The problem is known as Regression when output is numeric
    - The problem is known as Classification when the output is categorical
- f_{gt}: Ground truth of the function to be estimated
  - Sometimes we know nothing about this function, such as its structure, type or basis functions.
  - It stands for the true relationships between the input and output.

2.1.2 Goal

- Approximate f and make use of it, so that the ability to inferring corresponding y from x is obtained.

2.2 Machine Learning Approach

2.2.1 Mathematical Representation

$$\hat{y} = \hat{f}_\theta(x)$$

- x: Input of the function.
- $\hat{y}$: Output of the estimated function. Usually called **predictions**.
- $\hat{f}_\theta$: Estimations of the ground truth function. Consists of basic functions we're familiar with and can be tuned with the parameter θ.

2.2.2 Training set

- Given a set of **input-output** pairs which is called **Training Set** and denoted by $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$, tune the parameter θ to achieve the "best" or at least a "good"answer, which means trying to make the predictions $\hat{y}$ as close as possible to the ground truth y
- Use such a tuned model to **predict** the value of output y when some unseen input x occur

2.2.3 Common approximations

- Support vector machines
- Radial basis functions
- Gaussian mixture functions
- Neural networks
  - Very high capacity order
  - Easy to train with modern computational tools
  - Shallow neural networks: Use one easy-to-train layer
  - Deep neural networks: Train a hierarchical stack of layers.
- …

2.2.4 Performance Measure

- Measure the "distance" between ground truth y and prediction $\hat{y} = \hat{f}_\theta(x)$
- The function to measure this "distance" is usually called **loss function** and is denoted by $l(\theta)$
- Largely depends on specific tasks and the output type. The required performance should be guaranteed when the loss is minimized.
  - Regression: Mean Square Error
  - Classification: Cross-Entropy Loss
  - Localization: Intersection over Union
  - …

2.2.5 Optimization

- Select θ which minimizes the **loss function computed on training set**
- Mathematically speaking: $\theta^* = \underset{\theta}{\operatorname{argmin}}\{l(\theta)\}$
- Modern optimization algorithms
  - Gradient descent
  - Expectation maximization

2.2.6 Validation Set

- Some times we are unsure about some hyper-parameters which are set manually like the model structure (or function basis)
- First train different models on the same training set with different hyper-parameters, then test the performance of these models and choose the one with best performance on **another set of input-output pairs** which is usually called **Validation Set**
- Do not have any intersection with the training set

2.2.7 Test Set

- Usually we're interested in how well the trained model performs on data that it **has not seen before**, since this determines how well it will work when deployed in the real world
- Evaluate the performance of the trained model on a **test set** of data that is separate from the data used for training and validation
- Do not have any intersection with training set and validation set

2.2.8 Summary

1. Collect a dataset which contains **input-output pairs**
2. If necessary, divide the data set to three parts for training, validation and testing.
3. Determine the function basis (model structure)
4. Determine the loss function needs to be optimized
5. Train the model by finding out θ^* which minimized the loss function computed on **train set**
6. Train the model under different hyper-parameters set manually and select the best hyper-parameters based on its performance on **validation set**
7. Test such a well specific model on **test set**
8. Use this model for **prediction** in real problems

2.3 No Free Lunch Theorem

2.3.1 Brief summary

- All models are approximations
- All models make assumptions
- Assumptions are never perfect

2.3.2 For supervised learning

- While a general-purpose algorithm has average performance on all types of problem, a highly specialized algorithm can reach a incredibly high performance on a specific type of problem, sacrificing its performance on all the other type of problems.
- Always check the assumptions made implicitly or explicitly when selecting the model and algorithm
- Do not expect an algorithm performs well in every situation

3 A single-hidden-layer neural network

3.1 General definition

$$\hat{\mathbf{y}} = \mathbf{A}^{(2)}\sigma(\mathbf{A}^{(1)}\mathbf{X} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}$$

- $\mathbf{A}$: A matrix of multiplicative weights
- $\mathbf{b}$: A column vector of additive offsets, sometimes called *bias*
- σ: A point-wise activation function
- $\mathbf{X} = \{\mathbf{x}_1, \cdots, \mathbf{x}_n\}$ is usually called the **input layer**
- $\mathbf{H} = \sigma(\mathbf{A}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$ is usually called a **hidden layer**
- $\hat{\mathbf{y}} = \{\hat{y}_1, \cdots, \hat{y}_m\}$ is usually called the **output layer**

3.2 Why does this structure work?

- Universal Approximation Theorem (Cybenko,1989)
  - Fix a continuous function $\sigma : \mathbb{R} \to \mathbb{R}$ (activation function) and positive integers d, D. The function σ is not a polynomial if and only if, for every continuous function $f : \mathbb{R}^d \to \mathbb{R}^D$ (target function), every compact subset K of $\mathbb{R}^d$, and every $\epsilon > 0$ there exists a continuous function $f_\epsilon : \mathbb{R}^d \to \mathbb{R}^D$ (the layer output) with representation

    $$f_\epsilon = W_2 \circ \sigma \circ W_1$$

    where W_2, W_1 are composable affine maps and $\circ$ denotes component-wise composition, such that the approximation bound

    $$\sup_{x \in K} ||f(x) - f_\epsilon(x)|| < \epsilon$$

    holds for any ϵ arbitrarily small (distance from f to f_ϵ can be infinitely small).
- The point-wise activation function provides the network with the ability to approximate non-linearities. Otherwise, the **combination of linear transformation would always be a linear transformation**.

3.3 Common activation functions

- Logistic sigmoid function:
  - $\sigma_i(z) = \frac{1}{1+e^{-z_i}}$
- Rectified linear unit (ReLU):
  - $\sigma_i(z) = \begin{cases} 0 & if\ z_i \leq 0 \\ z_i & if\ z_i > 0 \end{cases}$
- Leaky ReLU
  - $\sigma_i(z) = \begin{cases} \alpha z_i & if\ z_i \leq 0 \\ z_i & if\ z_i > 0 \end{cases}$
- Softmax
  - $\sigma_i(z) = \frac{e^{z_i}}{\Sigma_j e^{z_j}}$

3.4 Deep Neural Networks

For many years, the **shallow learning** consists of 1 or 2 layers of network was thought to be enough since the universal approximation theorem claimed that a single-hidden-layer network can

learn any function. But over the past decade, there has been overwhelming empirical evidence that **deep learning** typically consists of 10 to 100 layers **dramatically outperforms shallow learning** on a wide range of real applications