Probability for ML Cheatsheet

Compiled by Tu Nguyen Ngoc. Material based on Kelvin Murphy's book (`@probml`). Please share comments, suggestions, and errors at `github.com/nguyentuss/Probability-for-Machine-Learning-Cheatsheet`.

Last Updated February 22, 2025

Introduction

Supervised Learning

The task T is to learn a mapping f from $x \in X$ to $y \in Y$. The x are also called the **features**. The output y is called the **label**. The experience E is given in the form of a set of N input-output pair $\mathcal{D} = \{(x_n, y_n)\}, n = 1 \to N$ is called **training set**. (N is called the **sample size**. The performance P depends on the type of output we want to predict.

Classification

In classification problem, the output space is a set of C labels called **classes**, $Y = \{1, 2, ..., C\}$. The problem predicting the class label given a input is called **pattern recognition**. The goal of supervised learning in classification problem is want to predict the label. A common way to measure the perform on this task is called **misclassification rate**.

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \mathbb{I}\left(y_n \neq f(x_n; \boldsymbol{\theta})\right)$$

Where $\mathbb{I}(e)$ is indicator function, which return 1 if the conditional is true, return 0 otherwise. We can also use the notation **loss function** $l(y, \hat{y})$.

$$\mathcal{L}(\boldsymbol{\theta}) \triangleq \frac{1}{N} \sum_{n=1}^{N} \ell\left(y_n, f(x_n; \boldsymbol{\theta})\right)$$

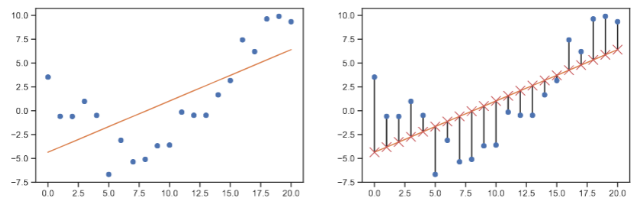
Regression

Similarly to the classification problem, but now the output in regression are a real-value $y \in \mathbb{R}$ instead the discrete value $y \in \{1, ..., C\}$; this is known as **regression**. So we need to use a different loss function. For regression, the most common choice is to use quadratic loss, or ℓ_2 loss (L2 normalization)

$$\ell_2(y, \hat{y}) = (y - \hat{y})^2$$

This penalizes large residuals $y - \hat{y}$. The empirical risk when use quadratic risk is equal to the **Mean squared error** or **MSE**.

$$MSE(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} (y_n - f(x_n; \boldsymbol{\theta}))^2$$



An example of the regression model in 1d data, we can fix the data using the **linear regression** model.

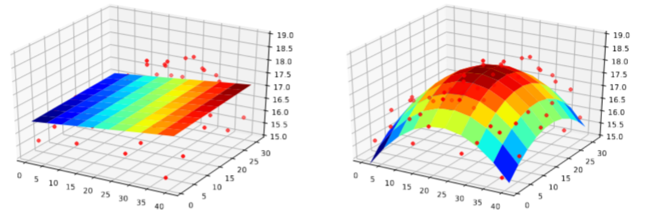
$$f(x; \boldsymbol{\theta}) = b + wx$$

Where w is the **slope**, b is the **bias**, and θ are the parameters of the model, we can minimize the sum square error.

$$\hat{\boldsymbol{\theta}} = \arg\min_{\boldsymbol{\theta}} MSE(\boldsymbol{\theta})$$

If we have multiple input features, we can write

$$f(\mathbf{x}; \boldsymbol{\theta}) = b + w_1 x_1 + ... + w_D x_D = b + \mathbf{w}^{\mathbf{T}} \mathbf{x}$$



(a) (b)

We can improve the fit by using a **Polynomial regression** model with degree $\mathcal{D}$. This now have the form

$$f(x; \mathbf{w}) = \mathbf{w}^{\mathbf{T}} \phi(x)$$

Where $\phi(x)$ are the feature vector derived from the input

$$\phi(x) = [1, x, x^2, ..., x^D]$$

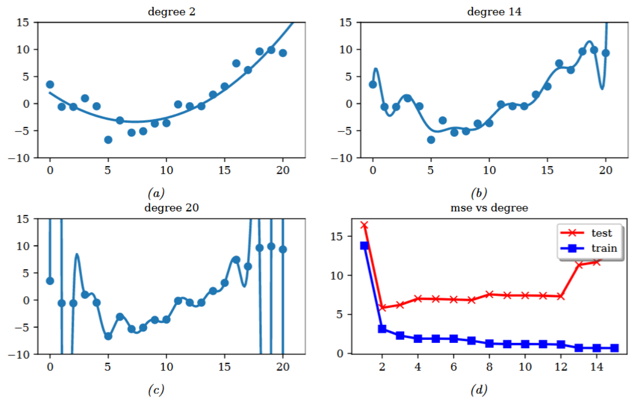
Overfitting

Empirical risk (training loss function)

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{train}}) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{train}}} \ell(y, f(x; \boldsymbol{\theta}))$$

The difference $\mathcal{L}(\boldsymbol{\theta}; p^*) - \mathcal{L}(\boldsymbol{\theta}; D_{train})$ called **generalization gap**. If a model has a large generalization gap (i.e., low empirical risk but high population risk), it is a sign that it is overfitting. In practice we don't know p^*. However, we can partition the data we do have into two subsets, known as the training set and the **test set**. Then we can approximate the population risk using the **test risk**:

$$\mathcal{L}(\boldsymbol{\theta}; \mathcal{D}_{\text{test}}) = \frac{1}{|\mathcal{D}_{\text{test}}|} \sum_{(\mathbf{x}, \mathbf{y}) \in \mathcal{D}_{\text{test}}} \ell(y, f(x; \boldsymbol{\theta}))$$

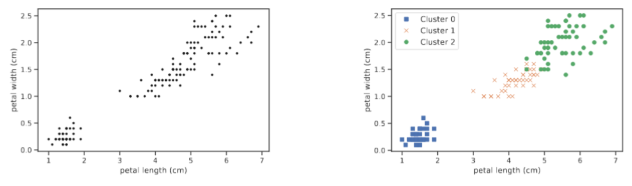


We can make the training loss function to zero if we increase the degree $\mathcal{D}$, but it will increase the testing loss function. The purpose about the prediction accuarcy on new data, A model that fit the training data but which is too much complex. It will call the **overfitting**. If **D** is too small, the model will be **underfitting**.

Unsupervised learning

In supervised learning, we assume that each input x in training set have a output targets y, and our goal is to learn the input-output mapping. Although this is useful, and difficult, supervised learning is just a to find a mathematical function to fit the data points. So go back to the unsupervised learning, we will opposed to just learning a mapping. We just get $\mathcal{D} = \{(x_n : n = 1 : N)\}$ without any ouputs y_n.This is called **unsupervised learning**.
From a probabilistic perspective, we can view the task of unsupervised learning as fitting an unconditional model of the form $p(x)$, which can generate new data x, whereas supervised learning involves fitting a conditional model, $p(y|x)$, which specifies (a distribution over) outputs given inputs.
Unsupervised learning avoids the needs of collect large labeled datasets for training, which can be often time comsuming and expensive and does not rely on manually labeled data or predefined categories, unlike supervised learning, which learns from labeled examples. Instead, unsupervised learning finds patterns, structures, or groupings in the data based on inherent similarities or relationships.

Clustering



(a) (b)

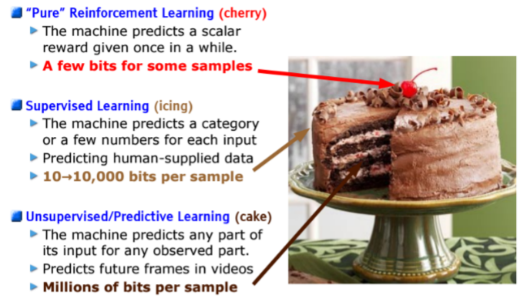
A simple example of unsupervised learning is the problem of finding clusters in data. The goal is to partition the input into regions that contain "similar" points.

Self-supervised learning

Self-supervised learning that automatically generating **labels** from **unlabeled data**. Try to learn to predict a color image from a grayscale image, or to mask out words in a sentence and then try to predict them given the surrounding context. The hope is that the resulting predictor $\hat{x}_1 = f(x_2; \boldsymbol{\theta})$. Where x_2 is the observed input and $\hat{x}_1$ is the predict output, will learn useful features from the data, that can be used in standard.

Reinforcement learning

The system or agent has to learn how to interact with its environment. For example, creating a bot playing Mario, a bot will interact and integration with the world, run left or right or the bot will jump if they see a block stone.(`Click to see the detail`)



- "Pure" Reinforcement Learning (cherry)
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- Supervised Learning (icing)
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**