AI Notes

Huu Duc Nguyen M.Sc.

29 February 2022

Contents

Abbreviations

AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CS	Computer Science
CV	Computer Vision
RL	Reinforcement Learning
NLP	Natural Language Processing
prob.	probability
params.	parameters
algor.	algorithms
PDF	Probability Density Function
MLE	Maximum Likelihood Estimation
MAP	Maximum A Posteriori
MoG	Mixture of Gaussians
NAG	Nestorov Accelerated Gradient
RMSprop	Root mean squared prop
Adam	Adaptive moment estimation
PCA	Principal Component Analysis
K-L	Kullback–Leibler
ReLU	Rectified Linear Unit

1 Introduction

Artificial Intelligence (AI) is the study field that leverages the ability of machines to mimic the problem-solving skill of human. It lies in the core of countless novel applications in real life, self-driving cars, virtual assistant, face recognition, etc. Machine Learning (ML) is a sub-field of Computer Science (CS) and AI, that "gives computers the ability to learn without being explicitly programmed" (Wikipedia). As a great amount of collected data and powerful computational hardware arise, Deep Learning (DL) is then a subset of ML (Fig. 3.1). Advanced applications which relate to Natural Language Processing (NLP), Computer Vision (CV), robotic learning, etc., are with in this DL subset.

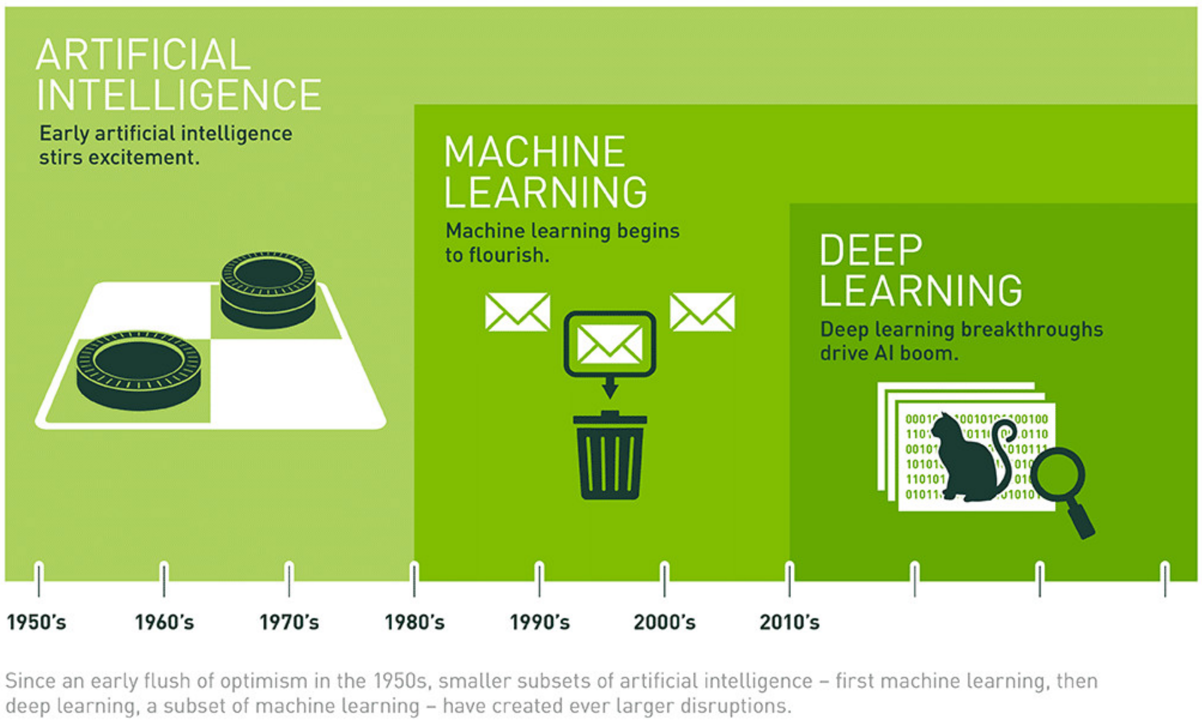
ARTIFICIAL INTELLIGENCE
Early artificial intelligence stirs excitement.
MACHINE LEARNING
Machine learning begins to flourish.
DEEP LEARNING
Deep learning breakthroughs drive AI boom.
1950's
1960's
1970's
1980's
1990's
2000's
2010's
Since an early flush of optimism in the 1950s, smaller subsets of artificial intelligence – first machine learning, then deep learning, a subset of machine learning – have created ever larger disruptions.

Figure 1.1: The relation between AI, ML and DL (src).

[TODO: The structure of the notes]
It's important to understand that there are more to AI and ML than just neural networks. In the end, to create a meaningful and working neural network, I believe one should have strong background in the basics of ML as well. Henceforth, I keep record of what I have learnt in the whole AI field. The structure of the notes is as follows:

- Chap. 2 introduces common ideas in ML.
- Chap. 3 introduces the mathematics background on probabilities, matrix.

- [**TODO: chapter 3**] explain basic concepts, the branching of different classes in ML. Later chapters presents each smaller branches.

2 Overview of Machine Learning

A machine learning algorithm is an algorithm that has the ability to *learn* from the data. A computer program is said to **learn**, if its performance at tasks in T, measured by P, improves with experience E (in which the experience is equivalent to the data). [GBC16]

2.1 Task T

A *task* is usually described by how the ML model process a single *data point*. This section presents some common ML tasks. [Vu18]

2.1.1 Classification

The task is to specify a label for the given data point. The labels are usually members of a list.

E.g., in the problem of digit classification, the data point is images of hand-written numbers. The data set comes with their labels as well. The task is then, given a unseen image, the model would be able to tell which number is in that image. In this problem, there are 10 possible labels, i.e., $0, 1, \ldots, 9$.

2.1.2 Regression

If the desired output is a real value, instead of a label in a list, then it's a regression problem. E.g.:

- with an image as the input data, the model predicts the age of the person
- given a feature vector, the model generates an image

2.1.3 Clustering

This is the task of grouping relevant data points based on some relationship between them.

E.g., find the pattern in customer shopping behaviors.

2.1.4 Others

Some worth-mentioning tasks:

- Recommendation System
- Machine Translation
- Completion
- Ranking
- Information Retrieval
- Denoising

2.2 Performance P

Usually, the dataset is divided into *training set* and *test set*. The model uses the training set to tune update the model parameters (params.) and the test set to examine the performance.

Online training is the approach when new data will continuously arise and introduce for the model to learn. E.g. in Reinforcement Learning (RL). *Offline training* is the opposite, the model learns from the a fixed training set.

2.3 Experience E

2.3.1 Supervised Learning

Supervised Learning is the approach that predict the outputs of new data points based on pairs of known inputs and outputs. This is the most common type of ML algorithms (algor.).

2.3.2 Unsupervised Learning

On the opposite, with unsupervised Learning, there is no known output, just inputs. Unsupervised Learning algor. will carry on some tasks based on the characteristics of the dataset, e.g. clustering, dimension reduction.

2.4 Model Parameters and Loss Function

Each ML model is described by a set of model params.. E.g., in the problem of finding a line passing through points in the 2D plane, the model params. are a, b in the line equation $y = ax + b$. The of training aim is to find the model params. that leads to the best performance. For classification problems, it means having the least number of incorrect classified data points. For regression problems, it means having the smallest difference with the actual output. It is then equivalent to having a optimization problem, in which we try to minimize a loss/cost function.

3 Probabilities

3.1 Definitions

3.1.1 Basic Definitions

- If x is discrete: $\sum\limits_x p(x) = 1$ with $\forall\ 0 \leq p(x) \leq 1$
- If x is continuous: $\int p(x)\,dx = 1 \Rightarrow \exists$ a **Probability Density Function (PDF)**
  $p(x)$ can take any positive value, as long as $\int p(x)\,dx = 1$
  [**TODO: Add image**]
  NOTE: : theoretically $p(x) = 0, \forall x$
- Common types

Joint probability:	$p(x_i, y_i)$	$(= p(X = x_i, Y = y_i))$
Marginal probability:	$p(x_i)$	$(= p(X = x_i))$
Conditional probability:	$p(y_i \vert x_i)$	$(= p(Y = y_i \vert X = x_i))$

- Sum rule: $\sum$ joint probability (prob.) = marginal prob.
  $\Rightarrow$ Marginalization
  - discrete variable: $p(x) = \sum\limits_y p(x, y)$
  - continuous variable: $p(x) = \int p(x, y) dy$
- Product rule: Product of marginal prob. and conditional prob. = joint prob.

3.1.2 Independence and Variability

- Independence. E.g.: x, y are independent, then

$$\begin{cases} p(x|y) = p(x) \\ p(y|x) = p(y) \end{cases} \iff p(x, y) = p(x).p(y)$$

- Variability
  - variance:
    $var\left[f\right] = \mathbb{E}\left[\left(f(x) - \mathbb{E}\left[f(x)\right]\right)^2\right] = \mathbb{E}\left[f(x)^2\right] - \mathbb{E}\left[f(x)\right]^2$
  - covariance:
    $cov\left[x, y\right] = \mathbb{E}_{x,y}\left[xy\right] - \mathbb{E}\left[x\right].\mathbb{E}\left[y\right] = \mathbb{E}_{x,y}\left[xy^T\right] - \mathbb{E}\left[x\right].\mathbb{E}\left[y^T\right]$

- covariance matrix

3.1.3 Bayes Rule

$$
\begin{aligned}
&p(x_i|y_i).p(y_i) = p(y_i|x_i).p(x_i) = p(x_i, y_i) \\
\Rightarrow\ &p(y_i|x_i) = \frac{p(x_i|y_i).p(y_i)}{p(x_i)} = \frac{p(x_i|y_i).p(y_i)}{\sum\limits_y p(x_i|y_i).p(y_i)}
\end{aligned}
$$

$\Rightarrow$ the ***Bayes equation***:

$$\textbf{posterior} = \frac{\textbf{likelihood} \times \textbf{prior}}{\textbf{normalization factor}}$$

3.1.4 Expectation

For variable x:	$\mathbb{E}[x] = \sum\limits_x x.p(x)$	$\left(= \int x.p(x)dx\right)$
For function $f(\cdot)$:	$\mathbb{E}[f(x)] = \sum\limits_x f(x).p(x)$	$\left(= \int f(x).p(x)dx\right)$

3.2 Types of Probability Distributions

Reference source: machinelearningcoban.com.

3.2.1 Bernoulli Distribution

Bernoulli Distribution is a distribution to describe binary discrete variables. It's the case that the variable can only take value in 2 classes $x \in \{0, 1\}$. E.g., the probability of throwing a coin. The Bernoulli distribution is defined with parameter $\lambda \in [0, 1]$:

$$p(x) = \text{Bern}_x[\lambda] = \begin{cases} p(x = 1) = \lambda \\ p(x = 0) = 1 - \lambda \end{cases} \tag{3.1}$$

In short form, the above equation can be combined into one:

$$p(x) = \lambda^x (1 - \lambda)^{(1-x)} \Rightarrow \begin{cases} p(0) = \lambda^0 (1 - \lambda)^1 = 1 - \lambda \\ p(1) = \lambda^1 (1 - \lambda)^0 = \lambda \end{cases} \tag{3.2}$$

3.2.2 Categorical Distribution

Categorical Distribution is the generalization of *Bernoulli Distribution*, in case there are K classes for the discrete variable $x \in \{1, 2, \dots, K\}$. Accordingly, there will be K parameters to describe this PDF: $\lambda = [\lambda_1, \lambda_2, \dots, \lambda_K]$, with $\lambda_k \geq 0$ and $\sum \lambda_k = 1$. Each λ_k represents the probability to take the output k: $p(x = k) = \lambda_k$. In short: $p(x) = \text{Cat}_x[\lambda]$.

Another common way to represent the output is the one-hot vector, $\mathbf{x} \in \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_K\}$ with $\mathbf{e}_k$ is the k-unit vector, which has all 0-element, except the k-element equal to 1. E.g., given 3 classes: $\mathbf{e}_1 = [1, 0, 0]^T, \mathbf{e}_2 = [0, 1, 0]^T, \mathbf{e}_3 = [0, 0, 1]^T$. We will then have:

$$p(\mathbf{x} = \mathbf{e}_k) = \prod_{j=1}^{K} \lambda_j^{x_j} = \lambda_k \tag{3.3}$$

because for $\mathbf{x} = \mathbf{e}_k$, only $x_k = 1$, while $x_j = 0, \forall j \neq k$.

3.2.3 Univariate Normal Distribution

Univariate Normal Distribution is also known as the Gaussian distribution. For single dimension data (in 1D): $x \in (-\infty, \infty)$, the mean $\mu \in \mathbb{R}$, and the variance σ^2 with $\sigma \in \mathbb{R}$.

$$p(x) = \text{Norm}_x \left[\mu, \sigma^2\right] = \mathcal{N}(\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}}.\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) \tag{3.4}$$

NOTE:

- **Marginals prob. of Gaussian are again Gaussian.**
- When estimating the params. of a Gaussian, beware the underestimation problem.

$$\begin{aligned}
\mathbb{E}\left[\mu_{ML}\right] &= \mu \\
\mathbb{E}\left[\sigma^2_{ML}\right] &= \left(\frac{N-1}{N}\right)\sigma^2 \\
\Rightarrow \widetilde{\sigma}^2 &= \left(\frac{N}{N-1}\right)\sigma^2_{ML} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \hat{\mu})^2
\end{aligned}$$

3.2.4 Multivariate Normal Distribution

Multivariate Normal Distribution is the extension of *Univariate Normal Distribution* to multi-dimensional data: $\mathbf{x}, \boldsymbol{\mu} \in \mathbb{R}^D, \sigma^2 \Rightarrow \Sigma \in \mathbb{S}^D_{++}$ ($\mathbb{S}^D_{++}$ is the set of positive definite

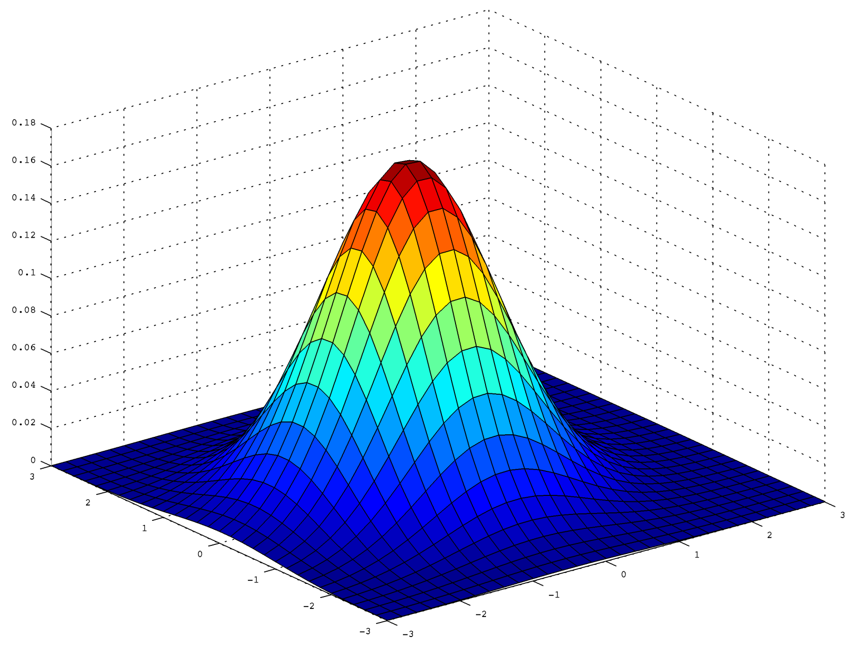

Figure 3.1: Bivariate Gaussian distribution (src).

symmetric matrix)

$$p(x) = \mathrm{Norm}_x[\boldsymbol{\mu}, \Sigma] = \mathcal{N}(\boldsymbol{\mu}, \Sigma) = \frac{1}{2\pi^{D/2}|\Sigma|^{\frac{1}{2}}}.\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right) \tag{3.5}$$

3.2.5 Beta Distribution

This distribution describes the parameter for another distributions. E.g., Dirichlet PDF describes Categorical Distribution (Subsec. 3.2.2)

3.3 Parameter Estimation

Many of ML problems are boiled down to finding *statistical models*. Those models could predict the prob. for the classification problem, prob. of events that will happen, etc. It all end up with finding the suitable set of params. for these *statistical models*.

3.3.1 Maximum Likelihood Estimation

Maximum Likelihood Estimation (MLE) finds the parameters that maximize the prob. of the existing data.

$$\theta = \underset{\theta}{\operatorname{argmax}}\, p(x_1, x_2, \dots, x_N|\theta) \tag{3.6}$$

Assuming independent variables:
$$\theta = \underset{\theta}{\operatorname{argmax}} \prod_{n=1}^{N} p(x_n|\theta) \tag{3.7}$$

Maximum log-likelihood:
$$\theta = \underset{\theta}{\operatorname{argmax}} \sum_{n=1}^{N} [\log p(x_n|\theta)] \tag{3.8}$$

Minimum negative log-likelihood:
$$\theta = \underset{\theta}{\operatorname{argmin}} \sum_{n=1}^{N} [-\log p(x_n|\theta)] \tag{3.9}$$

3.3.2 Maximum A Posteriori

Sometimes, we have prior knowledge of the PDF. E.g., we know that the prob. of getting head when flipping a coin is around 50%. Maximum A Posteriori (MAP) takes advantage of the prior knowledge $p(\theta)$ on the parameters θ by applying Bayes rule (Subsec. 3.1.3)

$$\theta = \underset{\theta}{\operatorname{argmax}} \prod_{n=1}^{N} p(x_n|\theta)p(\theta) \tag{3.10}$$

MLE suffers when there is not enough data $\Rightarrow$ use MAP

3.4 Naive Bayes Classifier

Naive implies having the independence assumption on the variables.

$$\begin{aligned} c &= \underset{c\in\mathbb{C}}{\operatorname{argmax}}\, p(c|x) \\ &= \underset{c\in\mathbb{C}}{\operatorname{argmax}}\, p(x|c)\, p(c) \end{aligned}$$

If x is:

- continuous variable $\Rightarrow$ Gaussian Naive Bayes
- feature vector $\Rightarrow$ Multinomial Naive Bayes
- binary vector $\Rightarrow$ Bernoulli Naive Bayes

Minimize the expected loss: $\mathbb{E}[L] = \sum_k \sum_j \int_{R_j} L_{kj}\, p(x, C_k)\, dx$ by choosing region R_j such that $\mathbb{E}[L] = \sum_k L_{kj}\, p(C_k|x)$

3.5 Views on the Decision Problem

3.5.1 Generative Methods

First determine the class-conditional densities and separately infer the prior class prob.
$\Rightarrow$ Bayes theorem $\Rightarrow$ class membership

$$p(x|C_k)\, p(C_k) \Rightarrow y_k(x)$$

E.g., Mixture of Gaussians

3.5.2 Discriminative Methods

First solve the inference problem of determined the posterior class prob.

3.6 Unknown Notes

E.g., 2 class $C_1,\ C_2$, 2 decisions $\alpha_1,\ \alpha_2$.

The loss: $L(\alpha_j|C_k) = L_{kj}$.

The expected loss is equal to the $Risk(R)$.

$$\mathbb{E}_{\alpha_1}[L] = R(\alpha_1|x) = L_{11}\, p(C_1|x) + L_{21}\, p(C_2|x)$$
$$\mathbb{E}_{\alpha_2}[L] = R(\alpha_2|x) = L_{12}\, p(C_1|x) + L_{22}\, p(C_2|x)$$

Choose α_1 if $R(\alpha_1|x) < R(\alpha_2|x)$

3.7 Probability Density Estimation

3.7.1 Histogram

This is ***non-parametric*** prob. density estimation. All other approaches are ***parametric***. The prob. of a bin:

$$p_i = \frac{n_i}{N.\Delta_i} \tag{3.11}$$

in which n_i is the number of data points in that bin, N is the total number of data point, Δ_i is the width of the bin, often $\Delta_i = \Delta$.

Notes:

- Δ serves as the **smoothing factor**
- With D as the dimensions of the data points. The number of bins grow exponentially with $\mathcal{O}(k^D)$

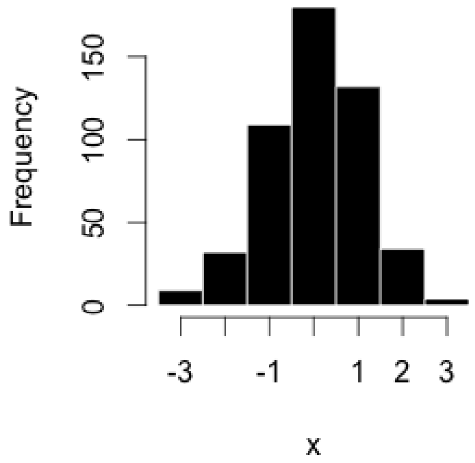

Figure 3.2: Example of a histogram, $\Delta = 1$ (src).

3.7.2 Parametric Probability Density Estimation

In the other hands, one could find the prob. from $p = \int_{\mathcal{R}} p(y)dy \approx p(x)V$ where the region $\mathcal{R}$ is sufficiently small.

$\Rightarrow p(x) \approx \dfrac{K}{N.V}$, where K is the number of data points in the region, V is the volume of the region.

3.7.3 Kernel Methods

The kernel methods fix V and determine K. The volume V is the space restricted within a parzen window $k(u)$ that satisfies $k(u) \geq 0$.

A hyper-space cube: $$k(u) = \begin{cases} 1 \;\; if \;\; |u_i| \leq \frac{1}{2}h, \;\; i = 1, 2, \ldots, D \\ 0 \;\; else \end{cases} \tag{3.12}$$

The number of points inside: $$K = \sum_{n=1}^{N} k(x - x_n) \tag{3.13}$$

The region *volume*: $$V = \int k(u)du = h^D \tag{3.14}$$

The probability: $$\Rightarrow p(x) \approx \frac{K}{N.V} = \frac{1}{N.h^D} \sum_{n=1}^{N} k(x - x_n) \tag{3.15}$$

The ***symmetric Gaussian kernel is a better substitution*** for the asymmetric parzen window.

A Gaussian kernel: $$k(u) = \frac{1}{\sqrt{2\pi h^2}} exp\left(\frac{-u^2}{2h^2}\right) \tag{3.16}$$

The region *volume*: $$V = \int k(u)du = 1 \tag{3.17}$$

The probability: $$\Rightarrow p(x) \approx \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi)^{D/2} h} exp\left(\frac{-||x - x_n||^2}{2h^2}\right) \tag{3.18}$$

For Kernel methods, h is the ***smoothing factor***.

Generalization: $k(u) \geq 0$, $\int k(u)du = 1$.

Size of the hypersphere is proportional to h^2.

3.7.4 K-Nearest Neighbor

When you fix K and determine V, it leads to K-Nearest Neighbor.

[**TODO: Add image**]

$$p(x) \approx \frac{K}{NV} \tag{3.19}$$

Here, K is the ***smoothing factor***.

- **Too much bias $\Rightarrow$ too smooth**
- **Too much variance $\Rightarrow$ NOT smooth enough**

$\Rightarrow$ combine parametric methods to a mixture model

Mixture distribution = multi parametric model

3.7.5 Mixture of Gaussians

Mixture of Gaussians (MoG), as **Generative Model**, is defined from the prob. sum of elemental Gaussians: $p(x|\theta) = \sum_{j=1}^{M} p(x|\theta_j)p(j)$, where $p(x|\theta_j)$ is a **mixture component**, $p(j) = \pi_j$ is the **weight of the component**

$$p(x|\theta_j) = \frac{1}{\sqrt{2\pi}\sigma_j} exp\left[\frac{-(x-\mu_j)^2}{2\sigma_j^2}\right], \quad p(j) = \pi_j, \quad \sum \pi_j = 1 \tag{3.20}$$

$$p(x|\theta_j) = \frac{1}{(2\pi)^{\frac{D}{2}}|\Sigma_j|^{\frac{1}{2}}} exp\left[-\frac{1}{2}(x-\mu_j)^T\Sigma_j^{-1}(x-\mu_j)\right] \tag{3.21}$$

3.7.6 K-Means Clustering

There are 3 steps:

- Pick K centroids
- Assign sample to the centroid
- Adjust centroids

Step 2 and 3 are repeated until there is no change.

This leads to a local optimum, depends on initialization. It's sensitive to ***outliers***, detects ***spherical clusters only***.

[TODO: Add images]

Application: e.g., image compression.

3.7.7 EM Clustering

It's short for Expectation-Maximization. Assuming N data points and K Gaussians.

- **E-Step:** Fix the Gaussians, find $\gamma_j(x)$, which represent the **responsibility of component j for x**.

$$\gamma_j(x_n) = \frac{\pi_j \mathcal{N}\left(x_n | \mu_j, \Sigma_j\right)}{\sum_{k=1}^{K} \pi_k \mathcal{N}\left(x_n | \mu_k, \Sigma_k\right)} \quad \forall j = 1, 2, \ldots, K, \quad n = 1, 2, \ldots, N \tag{3.22}$$

- **M-Step:** Fix $\gamma_j(x)$, update the Gaussians.

$$\hat{N}_j = \sum_{n=1}^{N} \gamma_j(x_n) \tag{3.23}$$

$$\hat{\mu}_j = \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n) x_n \tag{3.24}$$

$$\hat{\pi}_j = \frac{\hat{N}_j}{N} \tag{3.25}$$

$$\hat{\Sigma}_j = \frac{1}{\hat{N}_j} \sum_{n=1}^{N} \gamma_j(x_n) \left(x_n - \hat{\mu}_j\right) \left(x_n - \hat{\mu}_j\right)^T \tag{3.26}$$

Notes:

- Regularization with $\Sigma + \sigma_{min} I$
- Initialization μ_j with K-Means
- Hard-assignment: each data point to 1 class $\Rightarrow$ K-Means
- Soft-assignment: each data point $\Rightarrow$ prob. to fall into many classes $\Rightarrow$ EM Clustering

EM **needs more iteration**, because there are **more params.**.

4 Linear Regression

This is the simplest regression problem in ML.

Problem statement: Given data points $\mathbf{x}_i \in \mathbb{R}^D$ and their labels $y_i \in \mathbb{R}$, find the "line" that fit these data points. The line is represented via parameters $\mathbf{w}$. For each data point $\mathbf{x}$ and its label y

$$\begin{aligned}
\mathbf{w} &= [w_0, w_1, \dots, w_n]^T \\
\bar{\mathbf{x}} &= [1, x_0, \dots, x_n] \qquad \text{(x bar)} \\
y &\approx \hat{y} = \bar{\mathbf{x}}.\mathbf{w} \qquad \text{(y hat)} \\
\Rightarrow \frac{1}{2}e^2 &= \frac{1}{2}\left(y - \bar{\mathbf{x}}.\mathbf{w}\right)^2
\end{aligned}$$

The loss function for all points

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \bar{\mathbf{x}}_i.\mathbf{w}\right)^2 \tag{4.1}$$

The average loss:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2N}\sum_{i=1}^{N}\left(y_i - \bar{\mathbf{x}}_i.\mathbf{w}\right)^2 \tag{4.2}$$

We need to find the weights that minimize the loss function

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}}\,\mathcal{L}(\mathbf{w}) \tag{4.3}$$

We now can write the loss function using matrix form:

$$\mathcal{L}(\mathbf{w}) = \frac{1}{2}||\mathbf{y} - \overline{\mathbf{X}}.\mathbf{w}||_2^2 \tag{4.4}$$

with $\overline{\mathbf{X}} = \begin{bmatrix} \bar{\mathbf{x}}_1 \\ \bar{\mathbf{x}}_2 \\ \vdots \\ \bar{\mathbf{x}}_n \end{bmatrix}$ and $\mathbf{y} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$

Solution:

$$\frac{\partial \mathcal{L}(\mathbf{w})}{\partial \mathbf{w}} = \overline{\mathbf{X}}^T \left(\overline{\mathbf{X}}\mathbf{w} - \mathbf{y}\right) = 0 \tag{4.5}$$

$$\iff \overline{\mathbf{X}}^T\overline{\mathbf{X}}\mathbf{w} = \overline{\mathbf{X}}^T\mathbf{y} \tag{4.6}$$

$$\iff \mathbf{w} = \left(\overline{\mathbf{X}}^T\overline{\mathbf{X}}\right)^\dagger \overline{\mathbf{X}}^T\mathbf{y} \tag{4.7}$$

in which, $A^\dagger$ (A dagger) is the pseudo inverse of a matrix, because it's might not inverse-able. $\mathbf{A}^\dagger = \left(\mathbf{A}^\mathbf{T}\mathbf{A}\right)^{-\mathbf{1}}\mathbf{A}^\mathbf{T}$

NOTE: Sensitive to outliers $\Rightarrow$ pre-processing

Multi-variable:

$$\begin{aligned} \mathcal{L}(\mathbf{w}) &= \frac{1}{2N}||\mathbf{y} - \overline{\mathbf{X}}.\mathbf{w}||_2^2 \\ \Rightarrow \nabla_\mathbf{w}\mathcal{L}(\mathbf{w}) &= \frac{1}{N}\overline{\mathbf{X}}^T(\overline{\mathbf{X}}\mathbf{w} - \mathbf{y}) \end{aligned}$$

Linear Discriminant Functions:

$$\begin{aligned} y(\mathbf{x}) &= \widetilde{\mathbf{W}}^T\widetilde{\mathbf{x}} \\ E(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\sum_{k=1}^{K}\left(\mathbf{w}_k^T\mathbf{x}_n - t_{kn}\right)^2 \\ \widetilde{\mathbf{W}} &= \left(\widetilde{\mathbf{X}}^T\widetilde{\mathbf{X}}\right)^{-1}\widetilde{\mathbf{X}}^T T \\ &= \widetilde{\mathbf{X}}^\dagger T \end{aligned}$$

Generalized Discriminant:

$$y_k(x) = \sum_{j=1}^{M} w_{kj}\phi_j(x) + w_{k0} = \sum_{j=0}^{M} w_{kj}\phi_j(x), \quad \phi_0(x) = 1 \tag{4.8}$$

5 Logistics Regression

Though named as "regression", logistics regression is used for classification problem. For the classification problem with 2 classes, instead of a hard threshold (from linear classification), we could have a soft one, and find the prob. of the input belongs to either class.

$$f(x) = \theta(\omega^T x) \tag{5.1}$$

The sigmoid function:

$$f(s) = \frac{1}{1+e^{-s}} \triangleq \sigma(s) \tag{5.2}$$

$$s = \ln\left(\frac{\sigma}{1-\sigma}\right) \tag{5.3}$$

$$\sigma'(s) = \sigma(s)\,(1-\sigma(s)) \tag{5.4}$$

The cross entropy error function:

$$J(w,x,y) = -\left(y_i log\, z_i + (1-y_i)\, log(1-z_i)\right) \tag{5.5}$$

with $z_i = f(w^T x_i)$

$$\Rightarrow \frac{\partial J}{\partial w} = (z_i - y_i)x_i \tag{5.6}$$

$$\Rightarrow w \quad = w + \eta(y_i - z_i)x_i \tag{5.7}$$

NOTE: Require less params., only D dimensions, compared to Gaussians with $\frac{M(M+5)}{2} + 1$ params.

6 Softmax Regression

Multinomial Logistics Regression, Maximum Entropy Classifier

$$a_i = \frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} \tag{6.1}$$

$$\text{so that} \begin{cases} a_i > 0 \\ \sum a_i = 1 \\ z_m > z_n \iff a_m > a_n \quad \text{(order)} \end{cases}$$

When z_i is too big

$$\frac{\exp(z_i)}{\sum_{j=1}^{C} \exp(z_j)} = \frac{\exp(z_i - c)}{\sum_{j=1}^{C} \exp(z_j - c)} \tag{6.2}$$

with $c = \underset{i}{max}\ z_i$

$$J(w, x, y) = -\sum_{i=1}^{N} \sum_{j=1}^{C} y_{ij} \log(a_{ij}) \tag{6.3}$$

$$J(w, x, y) = -\sum_{n=1}^{N} \sum_{k=0}^{1} [\mathbb{I}(t_n = k) \ln p(y_n = k | x_n; w)] \tag{6.4}$$

$$\Rightarrow E(w) = -\sum_{n=1}^{N} \sum_{k=1}^{K} \left[\mathbb{I}(t_n = k) \ln \frac{\exp\left(w_k^T x\right)}{\sum_{j=1}^{K} \exp(w_j^T x)} \right] \tag{6.5}$$

$$\frac{\partial J_i(w)}{\partial W} = x_i e_i^T = x_i (a_{ij} - y_{ij})^T \tag{6.6}$$

$$W = W + \eta x_i (y_i - a_i)^T \tag{6.7}$$

7 Error Functions

As mention in Sec. 2.4, the loss function is the metric to assess the performance of the ML model with a certain task. Different loss functions fit with different tasks. Sometimes, the design of the loss function is what makes the differences.

[TODO: Add graph and explanation]

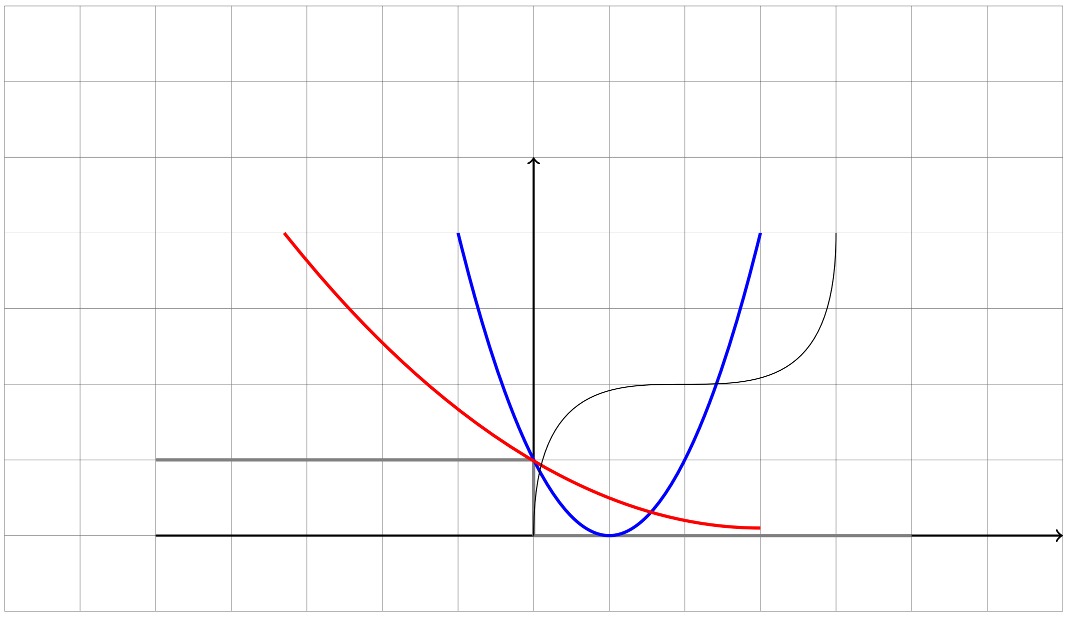

Figure 7.1: Different error functions: Ideal Miss-classification Error (gray line)

7.1 Ideal Miss-classification Error

Gradient $= 0 \Rightarrow$ can't use gradient descent.
It simply counts incorrectly classified points.

7.2 Squared Error - L_2 Loss

- Leads to closed form solutions
- Sensitive to outliers
- Penalize "too correct" data points

7.3 Cross Entropy Error

- Concave function ⇒ unique minimum exists
- Robust to outliers, error increases only roughly linear
- No closed-form solution, requires iterative method

7.4 Squared Error on Sigmoid / Tanh

- No penalty for "too correct" points
- Zero gradient for confidently incorrect classifications

⇒ **Do NOT** use L_2 loss with sigmoid outputs, instead, use cross-entropy.

7.5 Hinge Error

- Robust to outliers
- Zero error for points outside margin ⇒ sparsity
- Not differentiable around $z_n = 1$

NOTE: Want the correct class to have a score that is higher than incorrect class by a fixed margin Δ.

$$L_i = \sum_{j \neq y_i} \max(0, s_j - s_{y_i} + \Delta) \tag{7.1}$$

in which, s_j is other classes score, s_{y_i} is real class score.

7.6 L_1, L_0 Loss

Median, no wrong points

$$L_1 = \sum |t - y| \tag{7.2}$$

$$L_2 = \sum (t - y)^2 \tag{7.3}$$

7.7 Average Loss

Mathematically, dividing the loss by the amount of data N (in each batch or epoch) doesn't have any effect on the result. However, it's usually advisable to take the average to have more meaningful judgment and avoiding overflow when there are numerous data points.

8 Neural Network

8.1 Gradient Descent

NOTE: Just use ADAM??

8.1.1 Vanilla Gradient Descent

$$\theta_{t+1} = \theta_t - \eta \nabla_\theta f(\theta_t) \tag{8.1}$$

Check derivative:

$$f'(x) \approx \frac{f(x+\varepsilon) - f(x-\varepsilon)}{2\varepsilon} \qquad \text{(numerical gradient)} \tag{8.2}$$

8.1.2 Momentum

- Init: $v_{dW_0} = 0, v_{db_0} = 0$
- Calculate dW, db
- Update W, b

$$\Rightarrow \begin{cases} v_{dW} & = \beta v_{dW} + (1-\beta) dW \\ v_{db} & = \beta v_{db} + (1-\beta) db \end{cases} \Rightarrow \begin{cases} W & = W - \alpha v_{dW} \\ b & = b - \alpha v_{db} \end{cases} \tag{8.3}$$

  The above formulas are to calculate the moving average of v_{dW} and v_{db}.
- Tips: Choose $\beta_1 = \mathbf{0.9}$, implying taking average of the last 10 steps.
- Reference source: DeepLearning.AI.

8.1.3 Nesterov Accelerated Gradient

Nestorov Accelerated Gradient (NAG):

$$v_t = \gamma v_{t-1} + \eta \nabla_t J\left(\theta - \gamma v_{t-1}\right) \tag{8.4}$$

8.1.4 RMSprop

- Init $s_{dW_0} = 0, s_{db_0} = 0$
- Calculate dW, db
- Update W, b

$$\begin{cases} s_{dW} &= \beta s_{dW} + (1-\beta) dW^2 \\ s_{db} &= \beta s_{db} + (1-\beta) db^2 \end{cases} \Rightarrow \begin{cases} W &= W - \alpha \frac{dW}{\sqrt{s_{dW}} + \varepsilon} \\ b &= b - \alpha \frac{db}{\sqrt{s_{db}} + \varepsilon} \end{cases} \tag{8.5}$$

- Tips: choose $\boldsymbol{\beta_2 = 0.999}$, $\boldsymbol{\varepsilon = 10^{-7}}$
- Reference source: DeepLearning.AI.

8.1.5 Adam

Adam is basically the combination of Momentum and RMSprop.

- Init $v_{dW_0}, s_{dW_0}, v_{db_0}, s_{db_0} = 0$
- Calculate dW, db
- Update W, b

$$\begin{cases} v_{dW} &= \beta_1 v_{dW} + (1-\beta_1) dW \\ v_{db} &= \beta_1 v_{db} + (1-\beta_1) db \\ s_{dW} &= \beta_2 s_{dW} + (1-\beta_2) dW^2 \\ s_{db} &= \beta_2 s_{db} + (1-\beta_2) db^2 \end{cases} \Rightarrow \begin{cases} v_{dW}^{cor.} &= \frac{v_{dW}}{1-\beta_1^t} \\ v_{db}^{cor.} &= \frac{v_{db}}{1-\beta_1^t} \\ s_{dW}^{cor.} &= \frac{s_{dW}}{1-\beta_2^t} \\ s_{db}^{cor.} &= \frac{s_{db}}{1-\beta_2^t} \end{cases} \Rightarrow \begin{cases} W &= W - \alpha \frac{v_{dW}^{cor.}}{\sqrt{s_{dW}^{cor.}} + \varepsilon} \\ b &= b - \alpha \frac{v_{db}^{cor.}}{\sqrt{s_{db}^{cor.}} + \varepsilon} \end{cases} \tag{8.6}$$

- Tips: choose $\boldsymbol{\beta_1 = 0.9}$, $\boldsymbol{\beta_2 = 0.999}$, $\boldsymbol{\varepsilon = 10^{-7}}$
- Reference source: DeepLearning.AI.

8.2 Tips and Tricks

- Shuffling
- Data Augmentation: reshape, rescale, crops, zooming, change color (color Principal Component Analysis (PCA))
- Normalizing the inputs

  Convergence is the fastest if
  - The mean of each input variable = 0
  - Scale ⇒ same covariance

  Mean cancellation ⇒ Kullback–Leibler (K-L) expansion ⇒ covariance equalization (if possible)
- Leaky Rectified Linear Unit (ReLU) is better a bit than ReLU, ELU
- Weights initialization: Xavier-Glorot:

$$W \sim U\left(0, \sqrt{\frac{6}{n_{in} + n_{out}}}\right)$$

- Batch Norm(alization): Normalize after each layer

  ⇒ learn the moving average
- Drop out

  NOTE: When in inferencing (after training), must multiply the activation output with the prob. that the weights are set to 0

Bibliography

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep learning.* MIT press, 2016.

[Vu18] H. T. Vu. *Deep learning.* 2018.