
Statistical Machine Learning

1. Information

This is a summary for the lecture Statistical Machine Learning held by Professor Jan Peters Institute of Intelligent Autonomous Systems of the department of computer science at Technical University of Darmstadt in Germany. Please use this summary only for private causes.

Topics Covered:

- A short Linear Algebra Refresher
- A short Optimization Refresher

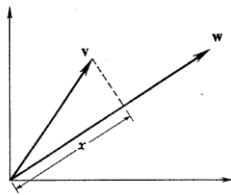
Linear Algebra Refresher

1. Vectors

- Multiplication by scalar (increases/decreases the length (magnitude) of a vector):

$$c\mathbf{v} = c \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} = \begin{bmatrix} cv_1 \\ \vdots \\ cv_n \end{bmatrix}$$

- Addition of vectors: $\mathbf{a} + \mathbf{b} = \begin{bmatrix} a_1 + b_1 \\ \vdots \\ a_n + b_n \end{bmatrix}$
- Positive recombination (important for vector spaces): $\mathbf{u} = c_1\mathbf{v}_1 + \dots + c_n\mathbf{v}_n$
- Inner product: $\mathbf{v}\mathbf{w} = \mathbf{v}^T\mathbf{w} = v_1w_1 + \dots + v_nw_n$
- Length of a vector (Frobenius norm):
 - $\|\mathbf{v}\| = (\mathbf{v}\mathbf{v})^{\frac{1}{2}}$
 - $\|c\mathbf{v}\| = |c|\|\mathbf{v}\|$
 - Triangle inequality: $\|\mathbf{v}_1 + \mathbf{v}_2\| \leq \|\mathbf{v}_1\| + \|\mathbf{v}_2\|$
- Angle between vectors: $\cos(\theta) = \frac{\mathbf{v}\mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|}$
- Projection of a vector (x is not a vector!): $x = \|\mathbf{v}\| \cos(\theta) = \|\mathbf{v}\| \frac{\mathbf{v}\mathbf{w}}{\|\mathbf{v}\|\|\mathbf{w}\|} = \frac{\mathbf{v}\mathbf{w}}{\|\mathbf{w}\|}$



- Vector transpose: $\mathbf{v} = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}, \mathbf{v}^T = [v_1, \dots, v_n]$
- Inner product: $\mathbf{v}^T\mathbf{v}$
- Outer product: $\mathbf{w}\mathbf{v}^T$

2. Matrices

- Scalar multiplication is done like for vectors.
- Addition of matrices is only defined if they have the same dimensions.
- For a transpose of a matrix the rows and columns are flipped. Here are the properties of transposes:
 - $(\mathbf{M}^T)^T = \mathbf{M}$
 - $(\mathbf{MN})^T = \mathbf{N}^T\mathbf{M}^T$
 - $(\mathbf{M} + \mathbf{N})^T = \mathbf{M}^T + \mathbf{N}^T$
 - If \mathbf{M} is square and $\mathbf{M} = \mathbf{M}^T$ is satisfied, then \mathbf{M} is symmetric.
- Multiplication of Vector by a Matrix: $\mathbf{u} = \mathbf{W}\mathbf{v}, \mathbf{W} \in \mathbb{R}^{M \times N}, \mathbf{v} \in \mathbb{R}^{N \times 1}, \mathbf{u} \in \mathbb{R}^{M \times 1}$
- Multiplication of a Matrix by a Matrix: $\mathbf{C} = \mathbf{AB}, \mathbf{A} \in \mathbb{R}^{M \times N}, \mathbf{B} \in \mathbb{R}^{N \times K}, \mathbf{C} \in \mathbb{R}^{M \times K}$
- Verifying the right dimensions is an important sanity checker when working with matrices
- Matrix Inverses

-
- Square matrices:

Operations and Linear Transformations

Wrap-Up

Optimization Refresher

1. Objectives

Make you remember or acquire key concepts in optimization!

- Covered Topics:
 - Unconstrained Optimization
 - Lagrangian Optimization
 - Numerical Methods (Gradient Descent)

2. Motivation

“All learning problems are essentially optimization problems on data.” was said by Christopher G. Atkeson, Professor at CMU. This does not mean that optimization is all you need to understand to be able to solve machine learning problems. It rather means that it is difficult to apply machine learning without some grasp of optimization. Some examples are:

- Image deblurring
- Robot trajectory planning
- Machine Learning (ML)

Virtually all ML algorithms can be expressed as minimizing a loss function over observed data. Given inputs $x_i \in X$, desired outputs $y_i \in Y$, a hypothesis function $h_\theta: X \rightarrow Y$ defined by parameters $\theta \in \mathbb{R}^n$ and a loss function $J: Y \times X \rightarrow \mathbb{R}_+$, ML algorithms solve the optimization problem:

$$\min_{\theta} \sum_{i=1}^m J(h_{\theta}(x_i), y_i)$$

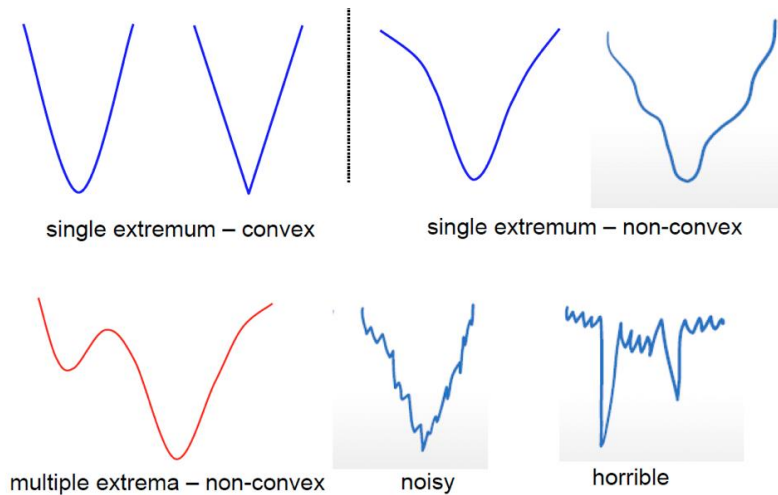
So, in any learning system we have

1. Parameters θ to enable learning
2. Dataset \mathcal{D} to learn from
3. A cost function $J(\theta, \mathcal{D})$ to measure our performance
4. Some assumptions on the data, with equality and inequality constraints,

How can we solve such problems in general? ML tells us how to come up with data-based cost functions such that optimization can solve them! E.g. in regression we are trying to minimize the mean squared error between the prediction and reference value. Other optimization problems would be classification or Clustering.

Good ML tells us how to come up with data-based cost functions such that optimization can solve them efficiently!

Most cost functions are difficult. Ideally, they are convex. The next figure highlights this.



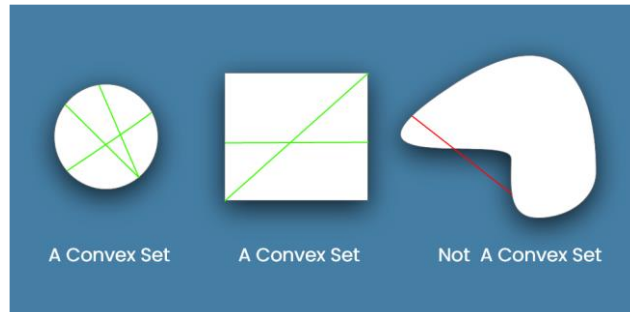
3. Convexity

3.1. Convex Sets

A set $C \subseteq \mathbb{R}^n$ is convex if $\forall \mathbf{x}, \mathbf{y} \in C$ and $\forall \alpha \in [0,1]$:

$$\alpha \mathbf{x} + (1 - \alpha) \mathbf{y} \in C$$

This is the equation of the line segment between \mathbf{x} and \mathbf{y} . I.e. for a given α , the point $\alpha \mathbf{x} + (1 - \alpha) \mathbf{y}$ lies in the line segment between \mathbf{x} and \mathbf{y} . To simplify things, think of convex sets as shapes where any line joining 2 points in this set is never outside the set.



Example of Convex Sets

Convex Functions

Unconstrained & Constrained Optimization

Numerical Optimization

Wrap-Up

Introduction to Machine Learning

1. Introduction

1.1. Why Should You Learn Machine Learning?

We generate much more data than we can analyze manually (Era of big data). No human being can deal with this data avalanche. Machine Learning (ML) is inherently data driven. Data is at the core of ML. The goal of ML is to design general-purpose methodologies to extract valuable patterns from data, ideally without much domain-specific expertise.

It's amazing what modern ML methods can do! For example:

- Image Generation (DALL-E, Midjourney): Generating photorealistic images from text prompts
- Text Generation (ChatGPT): Generate human like responses
- Intelligent Robots (ANYmal, Boston Dynamics): Perform complex behaviors such as juggling

1.2. What is Machine Learning?

ML is a subfield of artificial intelligence that gives computers the ability to learn without explicitly being programmed. Artificial Intelligence (AI) systems are used to perform complex tasks in a way that is like how humans solve problems. More formally, According to T. Mitchell (1997): "A computer program is said to learn from experience E with respect to some class of tasks T , and performance measure P , if its performance at tasks in T , as measured by P , improves with experience E ." The Goal is to develop **a machine / an algorithm** that **learns** to perform a **task** from **experience**!

A classic example is the recognition of handwritten digits. These digits are given to us as small digital images. We have to build a "machine" to decide which digit it is. The obvious challenge is that there are many different ways in which people handwrite.

The goal of ML is to develop a machine/an algorithm that learns to perform a task from past experience. To put it more abstractly: Our task is to learn a mapping from input to output.

$$f: I \rightarrow O$$

Or to put it differently, we want to predict the output from the input.

$$y = f(x; \theta)$$

Where $x \in I$ is the Input such as images, text, sensor measurements, $y \in O$ is the output and $\theta \in \Theta$ are the parameters that need to be learned!

1.3. Types of Machine Learning

- **Supervised Learning:** Data with labels (input/output pairs)
 - Regression
 - Classification
- **Semi-supervised learning:** Data with and without labels
 - Clustering
 - Dimensionality reduction

-
- Density estimation
 - **Reinforcement learning:** Learning by doing.
 - **Unsupervised Learning:** Data without label.
-

2. Wrap Up

You know now:

- What Machine Learning is and what it is not.
- Some examples of Machine Learning applications.
- The different types of learning problems.

Statistics

1. Random Variables

1.1. General Definition

A random variable is a random number determined by chance. Its value is unknown and/or could change e.g., the temperature outside the room at the current time. More formally, it is **drawn according to a probability distribution**. Typical random variables in statistical learning are input data, output data or noise.

A probability distribution describes the probability (mass/density) that the random variable will be equal to a certain value. The probability distribution can be given by the physics of an experiment (e.g., throwing dice).

An important concept is the data generating model, e.g., what is the data generating model for:

- i. throwing dice
- ii. regression
- iii. classification
- iv. visual perception?

1.2. Discrete/Continuous Random Variable

Let χ denote the set of possible values that a random variable X can take, i.e., the *sample space* or *state space*.

Discrete random variable i.e., χ is finite (or countably finite). We define the **probability mass function** (pmf) as the probability that X would be *equal* to a sample x .

$$p(x) = P(X = x) \text{ with } 0 \leq p(x) \leq 1 \text{ and } \sum_{x \in \chi} p(x) = 1$$

Continuous random variable i.e., χ is infinite and uncountable. We define the **probability density function** (pdf) or density as the *relative likelihood* that X would be *close* to a sample x .

$$p(x), \text{ with } p(x) \geq 0 \text{ and } \int_{\chi} p(x) dx = 1$$

1.3. Cumulative distribution function

Let $\chi = \mathbb{R}$ then the Cumulative distribution function (cdf) is an increasing differentiable function F , mapping \mathbb{R} to $[0,1]$ with $F(-\infty) = 0$ and $F(+\infty) = 1$.

$$F_x(x) = P(X \leq x) = \int_{-\infty}^x p(x') dx'$$

$$p(x) = \frac{dF_x}{dx}(x)$$

This is a good way to sample ANY random variable from a uniform distribution!

2. Basic Rules of Probability

- Joint distribution: $p(x, y)$
- Marginal distribution: $p(y) = \int p(x, y) dx$
- Conditional distribution: $p(y|x) = \frac{p(x, y)}{p(x)}$, if $p(x) > 0$
- Chain rule of probabilities:

$$\begin{aligned} p(x_1, \dots, x_n) &= p(x_1|x_2, \dots, x_n)p(x_2, \dots, x_n) \\ &= p(x_1|x_2, \dots, x_n)p(x_2|x_3, \dots, x_n)p(x_3, \dots, x_n) \dots p(x_{n-1}|x_n)p(x_n) \end{aligned}$$

- Bayes Rule: $p(x|y) = \frac{p(x|y)p(y)}{p(x)} \leftrightarrow \text{posterior} = \frac{\text{likelihood} * \text{prior}}{\text{marginal}}$

3. Expectations, Variance and Moments

3.1. Expectation

The average value of some function $f(x)$ under a probability distribution $p(\cdot)$:

$$\mathbb{E}_{x \sim p(\cdot)}[f(X)] = \mathbb{E}_X[f] = \mathbb{E}[f] = \begin{cases} \sum_x p(x)f(x) & \text{discrete case} \\ \int p(x)f(x)dx & \text{continuous case} \end{cases}$$

We note $\mu = \mathbb{E}[X]$. **Expectation rules:** $X \rightarrow \mathbb{E}[X]$ is a linear function i.e., $\mathbb{E}[\alpha X + Y] = \alpha \mathbb{E}[X] + \mathbb{E}[Y]$.
In General: $\mathbb{E}[g(X)] \neq g(\mathbb{E}[X])$.

3.2. Conditional Expectation

$$\mathbb{E}_{x \sim p(\cdot|y)}[f(X)] = \mathbb{E}_X[f|y] = \mathbb{E}[f|y] = \begin{cases} \sum_x p(x|y)f(x) & \text{discrete case} \\ \int p(x|y)f(x)dx & \text{continuous case} \end{cases}$$

3.3. Approximate Expectation (Monte Carlo Sampling)

$$\mathbb{E}[f] = \int p(x)f(x)dx \approx \frac{1}{N} \sum_{n=1}^N f(x_n)$$

When there is no analytical solution, we can use this to approximate integrals by sampling! This is also known as Monte-Carlo sampling.

3.4. Variance and Covariance

Variances give a measure of dispersion, e.g. the expected spread of the variable in relation to its mean:

$$\begin{aligned} \text{var}[X] &= \mathbb{E}[(X - \mathbb{E}[X])^2] = \mathbb{E}[X^2] - \mathbb{E}[X]^2 \\ \text{std}[X] &= \sqrt{\text{var}[X]} = \sigma \end{aligned}$$

Covariances give a measure of correlation - how much two variables change together.

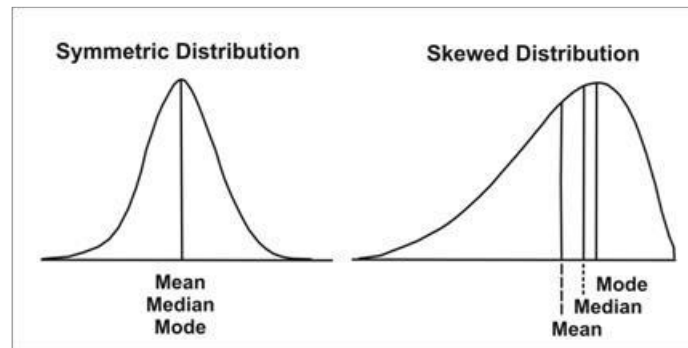
$$\text{cov}[X, Y] = \mathbb{E}_{X,Y}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])] = \mathbb{E}_{X,Y}[XY] - \mathbb{E}[X]\mathbb{E}[Y]$$

Note the very **important rule**:

$$\mathbb{E}[XX^T] = \mathbb{E}[X]\mathbb{E}[X]^T + \text{cov}[X, X] = \mu\mu^T + \Sigma$$

3.5. Moments of Random Variables

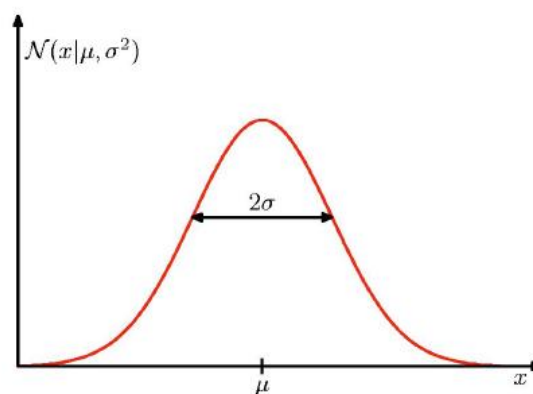
- Definition of a Moment: $m_n = \mathbb{E}[X^n]$
- Central Moment (deviation from the mean) Variance: $cm_n = \mathbb{E}[(x - \mu)^n]$
- Variance: cm_2
- skewness (measure of asymmetry): cm_3
- kurtosis (measure of heavy tailed-ness and light tailed-ness): cm_4



4. The Gaussian Distribution

The gaussian distribution is given as:

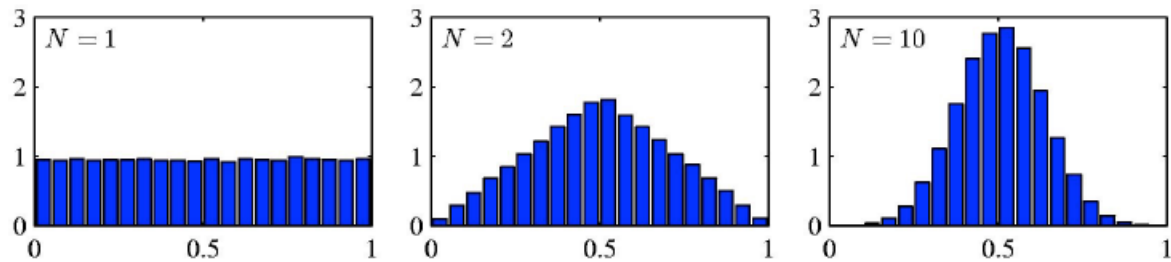
$$p(x) = N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$



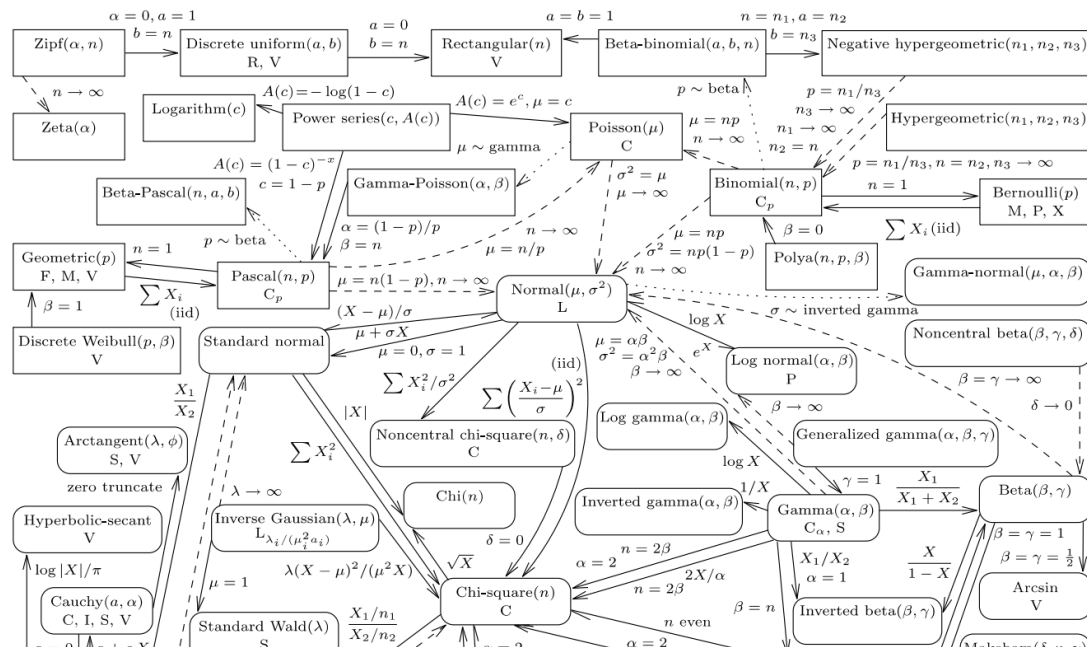
4.1. Central Limit Theorem

The distribution of the sum of N i.i.d. (independent and identically distributed) random variables becomes increasingly Gaussian as N grows.

Application: What would be the “shape” of the mean of samples drawn from ANY random variable? Gaussian, if we draw enough samples. This is why Gaussians are SO important! Let’s look at an example where we have N uniform $[0,1]$ random variables:



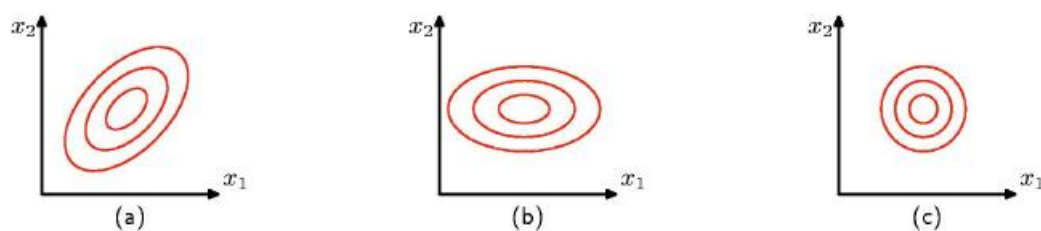
Gaussians are often a **good** model of data. Working with Gaussians leads to **analytic solutions for complex operations**. The distribution landscape is very big though and we do not always use Gaussian distributions for everything!



4.2. Multivariate Gaussian Distribution

The multivariate Gaussian distribution is given as

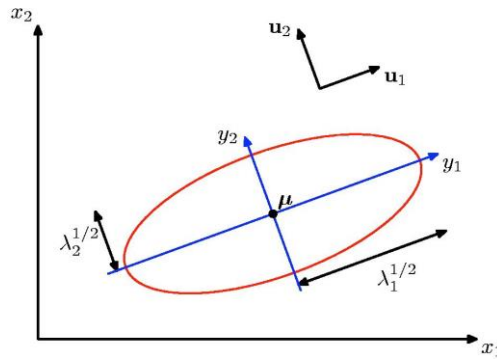
$$p(\mathbf{x}) = N(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{(2\pi)^{\frac{D}{2}}} \frac{1}{\sqrt{|\boldsymbol{\Sigma}|}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})}$$



To clear some confusion, for a chosen vector \mathbf{x} , $N(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is a real number indicating the relative likelihood of \mathbf{X} to be close to \mathbf{x} . The mean $\boldsymbol{\mu}$ is just a specific vector amongst all the possible vectors. The covariance matrix $\boldsymbol{\Sigma}$ tells us how two dimensions of a vector are related to each other.

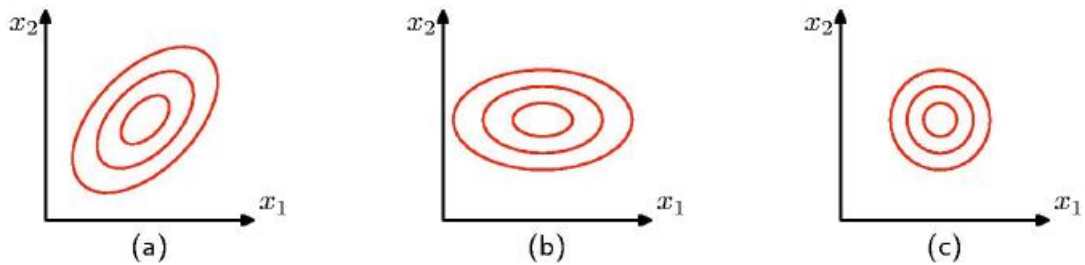
The **geometry** of the Multivariate Gaussian is given as

- $\Delta^2 = (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1} (\mathbf{x} - \boldsymbol{\mu})$
- $\boldsymbol{\Sigma}^{-1} = \sum_{i=1}^D \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T$
- $\Delta^2 = \sum_{i=1}^D \frac{y_i^2}{\lambda_i}$
- $y_i = \mathbf{u}_i^T (\mathbf{x} - \boldsymbol{\mu})$



where Δ is the **Mahalanobis distance**. The moments of the Multivariate Gaussian are given as:

$$\text{var}[\mathbf{X}] = \text{cov}[\mathbf{X}, \mathbf{X}] = \boldsymbol{\Sigma}$$



4.2.1. Partitioned Gaussian Distributions

We partition \mathbf{x} into two disjoint subsets

$$p(\mathbf{x}) = N(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma})$$

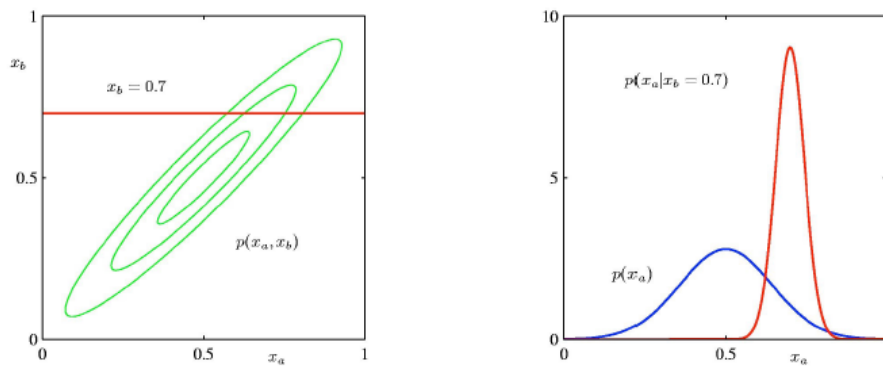
$$\mathbf{x} = \begin{pmatrix} x_a \\ x_b \end{pmatrix}, \boldsymbol{\mu} = \begin{pmatrix} \mu_a \\ \mu_b \end{pmatrix}, \boldsymbol{\Sigma} = \begin{pmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{pmatrix}, \boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}, \boldsymbol{\Lambda} = \begin{pmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{pmatrix}$$

where $\boldsymbol{\Lambda}$ is the precision matrix.

4.2.2. Partitioned Conditionals and Marginals

If the joint distribution $p(x_a, x_b)$ is Gaussian, then the

- a) conditional distributions $p(x_a|x_b)$ and $p(x_b|x_a)$ are also Gaussians.



- b) the marginal distributions $p(x_a)$ and $p(x_b)$ are also Gaussians.

4.2.3. Manipulating Gaussians

- a) Converting Marginal $p(\mathbf{x})$ and Conditional $p(\mathbf{y}|\mathbf{x})$ to Joint Distribution $p(\mathbf{x}, \mathbf{y})$:

$$\underbrace{\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})}_{p(\mathbf{x})} \underbrace{\mathcal{N}(\mathbf{y}|\mathbf{b} + \mathbf{F}\mathbf{x}, \mathbf{B})}_{p(\mathbf{y}|\mathbf{x})} = \underbrace{\mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{a} \\ \mathbf{b} + \mathbf{F}\mathbf{a} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{A}^T\mathbf{F}^T \\ \mathbf{F}\mathbf{A} & \mathbf{B} + \mathbf{F}\mathbf{A}^T\mathbf{F}^T \end{bmatrix}\right)}_{p(\mathbf{x}, \mathbf{y})}$$

- b) Converting Joint Distribution $p(\mathbf{x}, \mathbf{y})$ to Marginal $p(\mathbf{x})$ and Conditional $p(\mathbf{y}|\mathbf{x})$:

$$\underbrace{\mathcal{N}\left(\begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \middle| \begin{bmatrix} \mathbf{a} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix}\right)}_{p(\mathbf{x}, \mathbf{y})} = \underbrace{\mathcal{N}(\mathbf{x}|\mathbf{a}, \mathbf{A})}_{p(\mathbf{x})} \underbrace{\mathcal{N}(\mathbf{y}|\mathbf{b} + \mathbf{C}^T\mathbf{A}^{-1}(\mathbf{x} - \mathbf{a}), \mathbf{B} - \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C})}_{p(\mathbf{y}|\mathbf{x})}$$

4.3. Why are Gaussian distributions important?

- Central Limit Theorem.
- they are linked with many common distributions.
- they ease computations.

5. Exponential Family

All distributions from this family are uni-modal:

$$p(\mathbf{x}|\boldsymbol{\eta}) = h(\mathbf{x})g(\boldsymbol{\eta})e^{\boldsymbol{\eta}^T u(\mathbf{x})}$$

where $\boldsymbol{\eta}$ is the natural parameters and $g(\boldsymbol{\eta}) \int h(\mathbf{x})e^{\boldsymbol{\eta}^T u(\mathbf{x})} d\mathbf{x} = 1$, hence g can be interpreted as a normalization coefficient. The **exponential family** is a large class of distributions that are all analytically appealing, because taking the log of them decomposes them into simple terms.

An example is the Bernoulli Distribution:

$$p(x|\mu) = \mu^x (1 - \mu)^{1-x} = e^{x \ln \mu + (1-x) \ln(1-\mu)} = (1 - \mu) e^{\ln(\frac{\mu}{1-\mu})x} = \frac{1}{1 + e^\eta} e^{\eta x} = h(x) g(\eta) e^{\eta u(x)}$$

where $\eta = \ln\left(\frac{\mu}{1-\mu}\right)$, $h(x) = 1$, $g(\eta) = \frac{1}{1+e^\eta}$ and $u(x) = x$.

6. Information and Entropy

6.1. Core Questions

- How can we represent information compactly, i.e., using as few bits as possible?
 - Compressing text with GZIP, pictures in JPEG, movies in MPEG or sound in MP3.
- How can we transmit or store data reliably?
 - ECC memory
 - Error Correction on CDs
 - Communication with space probes
- Machine Learning Questions:
 - How can we measure complexity?
 - How can we measure “distances” between probability distributions?
 - How can we reconstruct data?

6.2. What is information

All letters in the English alphabet have a very different probability p_i of occurring. N bits can encode 2^N characters. The alphabet can be encoded in:

$$\lceil \log_2 27 \rceil \approx \lceil 4.75 \rceil = 5 \text{ bits}$$

We define the information in a single character as:

$$h(p_i) = -\log_2 p_i$$

Events with a low probability correspond to high information content. The *average* information in a character in an English text is:

$$H(p) = \mathbb{E}(h(.)) = -\sum_i p_i \log_2(p_i) \approx 4.1$$

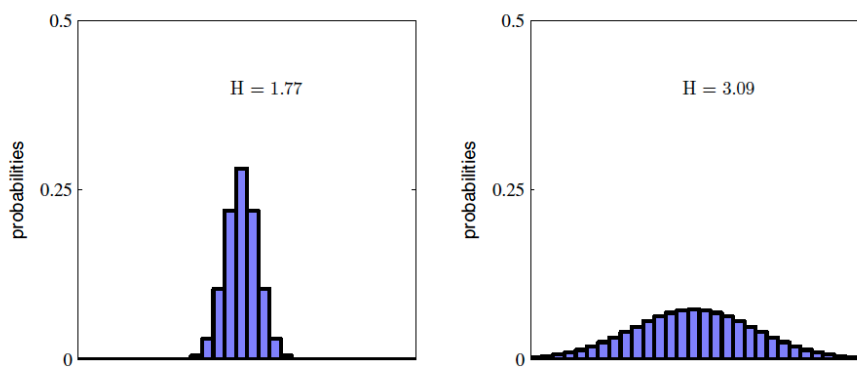
This quantity is called the **entropy**. On average, with the right encoding, we can represent each letter with 4.1 bits instead of 4.7.

Claude Shannon considered information as a **message** sent by a sender to a receiver, i.e., he wanted to solve the problem of **how to best encode information** that a sender wished to transmit to a receiver. Shannon gave information a mathematical value based on **probability** defined in terms of the concept of information **entropy** more commonly known as **Shannon entropy**. Information is defined as the measure of the increase of uncertainty for a receiver. Entropy quantifies the amount of uncertainty involved in the value of a random variable or the outcome of a random process. E.g., identifying the outcome of a **fair coin** flip provides less information (**lower entropy**) than specifying the outcome from a roll of a dice. Indeed:

$$-\ln\left(\frac{1}{2}\right) < -\ln\left(\frac{1}{6}\right)$$

6.3. Entropy of Distributions - Kullback-Leibler Divergence

What's the difference between these distributions?



The Kullback-Leibler Divergence - [KL Divergence](#) - is a similarity measure between two distributions, and is defined as:

$$KL(p||q) = - \int p(x) \ln q(x) dx - \left(- \int p(x) \ln p(x) dx \right) = - \int p(x) \ln \frac{q(x)}{p(x)} dx$$

It represents the average additional amount of extra bits required to specify a symbol X , given that its underlying probability distribution is the estimated $q(x)$ and not the true one $p(x)$.

Some very important properties

- **It is not a distance:** $KL(p||q) \neq KL(q||p)$
- **It is non-negative:** $KL(p||q) \geq 0$ with equality if $\forall x, p(x) = q(x)$

7. Wrap-Up

You know now:

- What random variables are (both continuous and discrete)
- What probability distributions are
- What expectation and variance are
- What a Gaussian distribution is and why it is so important
- What information and entropy are
- How to measure the similarity between two probability distributions

Bayesian Decision Theory

1. Objectives

Make you understand how to make an optimal decision! Covered Topics:

- Classification from a Bayesian point of view
- Bayesian Optimal Decisions
- Risk-based Classification

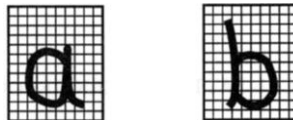
2. Bayesian Decision Theory

All statistical methods in machine learning share the fundamental assumption that the data generation process is governed by the rules of probability. The data is understood to be a set of random samples from some underlying probability distribution. Keep in mind for future lectures: Even if we do not explicitly mention the existence of an underlying probability distribution the basic assumption about how the data is generated is always there!

Bayesian Decision Theory is the statistical approach to pattern classification. It leverages probability to make classifications, and measures the risk (i.e. cost) of assigning an input to a given class.

2.1. Example

Let's delve into Bayesian Decision Theory using an example of handwritten character recognition. We have two classes of handwritten characters, denoted as C_1 and C_2 which represent the characters "a" and "b" respectively.



The **goal** is to classify a **new observed letter** such that the **probability of misclassification** is minimized!

2.2. First concept: Class Priors

The **a priori** probability of a data point belonging to a particular class is called the **class prior**. What can we tell about the probability **before** ("prior" meaning "before") seeing **new** data, meaning looking at **past occurrences** of the outcomes? What is the probability in the past?

Consider the following sequence of past data:

abaaa babaa aabba aaaaa

So, what is the probability that C_1 and C_2 occurs in the **next** observed letter? Based on the experience (i.e., the events that occurred in the past), the prior probabilities $p(a)$ and $p(b)$ are:

$$C_1 = a, p(C_1) = 0.75$$

$$C_2 = b, p(C_2) = 0.25$$

$$\sum_k p(C_k) = 1$$

Given the prior probabilities, we might predict that the next letter will likely be “a”. However, this prediction is based solely on past data and does not take into account the current context or situation.

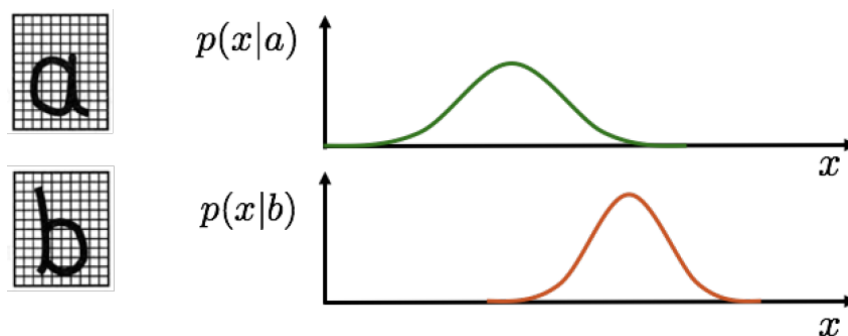
For instance, the past data might have been collected from individuals who predominantly write with their left hand. If the new letter we want to classify is written by a right-handed individual, the prior probabilities might not hold true.

The prior probability is akin to predicting a patient’s disease based only on their past medical history, without considering their current symptoms. It provides a baseline expectation based on historical data but does not account for present circumstances.

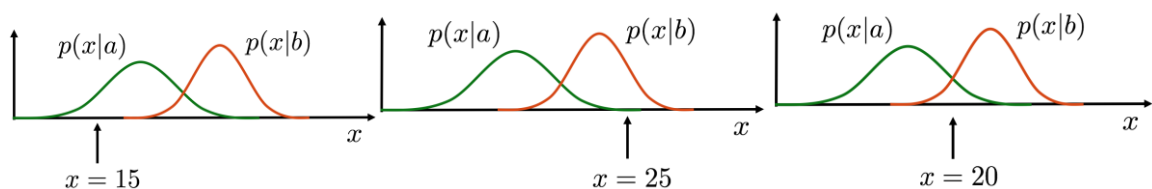
This is where the concept of class conditional probabilities, also known as likelihood, comes into play.

2.3. Second concept: Class conditional probabilities

This deals with the probability (likelihood) of making an observation x given that it comes from some class C_k . Here, x is often a feature vector, which measures/describes certain properties of the input data, e.g., number of black pixels, aspect ratio etc.



Assume we have a **new data points** $x = \{15, 25, 20\}$. How do we decide which class the new data points belong to?



- $p(15|b) < p(15|a) \rightarrow \text{class "a"}$
- $p(25|b) > p(25|a) \rightarrow \text{class "b"}$
- $p(25|b) = p(25|a) \rightarrow$ Assuming the previous priors $p(a)$ and $p(b)$, we should decide for class “a” since $p(a) > p(b)$

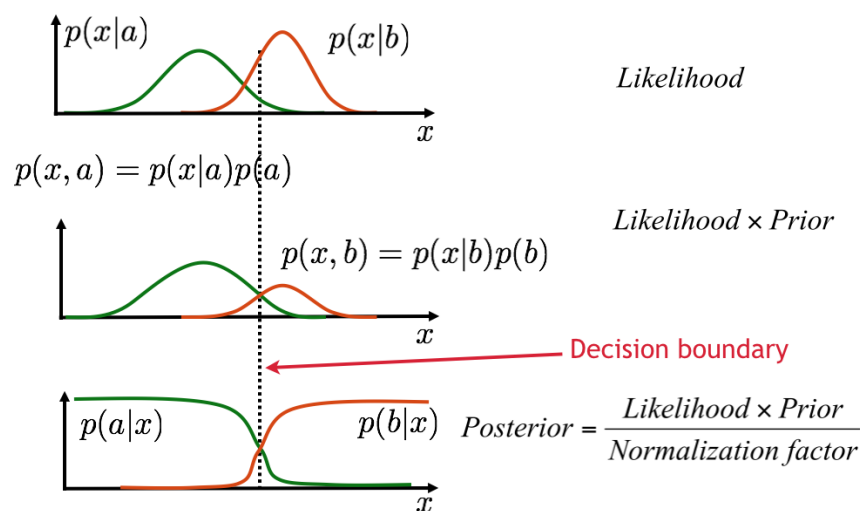
How can we formalize the incorporation of the **prior** and **likelihood** (class conditional probabilities)?

2.4. Third Concept: Class posterior probabilities

We want to find the **a posteriori probability** (posterior), i.e., the probability of class C_k given the observation x . For that we use the **Bayes' Theorem**:

$$p(C_k|x) = \frac{p(x|C_k)p(C_k)}{(x)} = \frac{p(x|C_k)p(C_k)}{\sum_j p(x|C_j)p(C_j)}$$

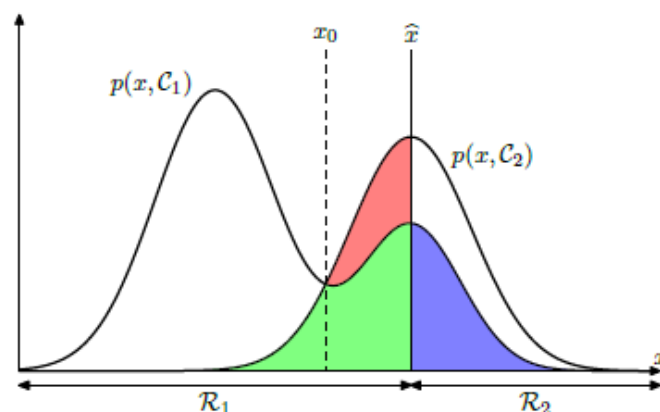
$$\text{Posterior} = \frac{\text{Likelihood} \times \text{Prior}}{\text{Normalization Factor}}$$



2.5. Bayesian Decision Theory

Why is it called Bayesian Decision Theory? To some extent, because it involves applying Bayes' rule. But this is not the whole story. The real reason is that it is built on so-called Bayesian Probabilities. Probability is not just interpreted as a frequency of a certain event happening. Rather, it is seen as a *degree of belief* in an outcome. Only this allows us to assert a *prior belief* in a data point coming from a certain class. Even though this might seem easy to accept to you now, this interpretation was quite contentious in statistics for a long time.

The goal of Bayesian decision theory is to minimize the *misclassification rate*, the probability of making a wrong decision:



$$\begin{aligned}
 p(\text{error}) &= p(x \in \mathcal{R}_1, \mathcal{C}_2) + p(x \in \mathcal{R}_2, \mathcal{C}_1) \\
 &= \int_{\mathcal{R}_1} p(x, \mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x, \mathcal{C}_1) dx \\
 &= \int_{\mathcal{R}_1} p(x|\mathcal{C}_2) p(\mathcal{C}_2) dx + \int_{\mathcal{R}_2} p(x|\mathcal{C}_1) p(\mathcal{C}_1) dx
 \end{aligned}$$

The *optimal decision rule* is to decide for \mathcal{C}_1 if

$$\begin{aligned}
 &p(\mathcal{C}_1|x) > p(\mathcal{C}_2|x) \\
 \Leftrightarrow &\frac{p(x|\mathcal{C}_1)p(\mathcal{C}_1)}{p(x)} > \frac{p(x|\mathcal{C}_2)p(\mathcal{C}_2)}{p(x)} \\
 \Leftrightarrow &p(x|\mathcal{C}_1)p(\mathcal{C}_1) > p(x|\mathcal{C}_2)p(\mathcal{C}_2) \\
 \Leftrightarrow &\frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} > \frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)} \quad (\text{Likelihood Ratio Test})
 \end{aligned}$$

A classifier obeying this rule is called *Bayes Optimal Classifier*. The *decision boundary* is the point where $\frac{p(x|\mathcal{C}_1)}{p(x|\mathcal{C}_2)} = \frac{p(\mathcal{C}_2)}{p(\mathcal{C}_1)}$. This line (or curve) can then be drawn into some graph and is the point where the classifier “switches” to the other class. This is most of the time only used for understanding what the classifier does than for real application (however, understanding what happens is important).

Recap Likelihood- Ratio Test

The likelihood ratio test is a statistical method used to compare two statistical models. It assesses the ratio of the likelihood of the data under one model to the likelihood of the data under another model. This test helps determine which model is more likely to be a better fit for the data.

Generalization to more than two classes

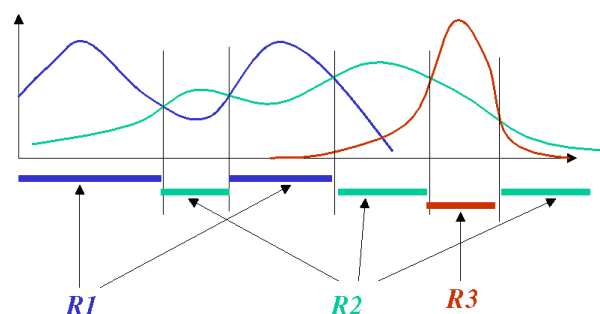
Decide for class k whenever it has the greatest posterior probability of all classes

$$p(\mathcal{C}_k|x) > p(\mathcal{C}_j|x) \quad \forall j \neq k$$

Which again results in the *Likelihood-Ratio Test*:

$$\begin{aligned}
 p(x|\mathcal{C}_k) p(\mathcal{C}_k) &> p(x|\mathcal{C}_j) p(\mathcal{C}_j) \quad \forall j \neq k \\
 \frac{p(x|\mathcal{C}_k)}{p(x|\mathcal{C}_j)} &> \frac{p(\mathcal{C}_j)}{p(\mathcal{C}_k)} \quad \forall j \neq k
 \end{aligned}$$

This yields more decision regions ($\mathcal{R}_1, \mathcal{R}_2, \mathcal{R}_3, \dots$) and multiple decision boundaries.

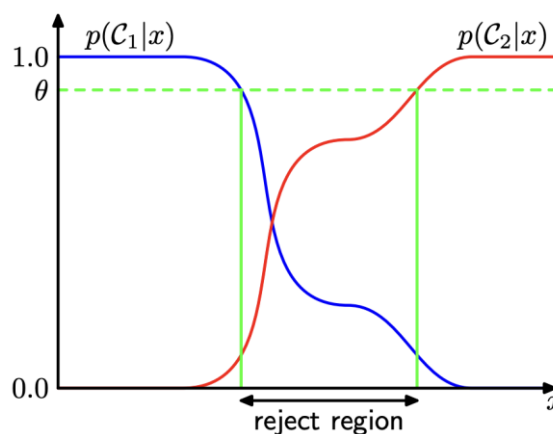


High Dimensional Features

So far, we have only considered one-dimensional features, i.e., $x \in \mathbb{R}$. We can use more features and generalize to an arbitrary D -dimensional feature space, i.e., $\mathbf{x} \in \mathbb{R}^D$. For example, in the salmon vs. sea-bass classification task $\mathbf{x} = [x_1 \ x_2]^T \in \mathbb{R}^2$, where x_1 is the width and x_2 is the lightness. The decision boundary we devised still applies to $\mathbf{x} \in \mathbb{R}^D$. We just need to use *multivariate class-conditional densities* $p(\mathbf{x}|C_k)$.

Reject Option

Classification errors arise from decision regions where the largest posterior probability $p(C_k|\mathbf{x})$ is significantly less than 1. Relatively high uncertainty about class memberships. For some applications, it may be better to reject an automatic decision entirely by introducing a decision threshold θ or use a dummy class "don't know" to which the system assigns all ambiguous cases. What are further reasons why we might want the machine learning model to abstain from making predictions? Insufficient training data, Out-of-distribution test data, Adversarial attacks, Biased outputs, Output not aligned with certain values, High risk and many more.



2.6. Risk Minimization

So far, we have tried to *minimize the misclassification rate*. There are many cases when *not every misclassification is equally bad*.

- Smoke detector
 - If there is a fire, we need to be very sure that we classify it as such
 - If there is no fire, it is ok to occasionally have a false alarm
- Medical diagnosis
 - If the patient is sick, we need to be very sure that we report them as sick
 - If they are healthy, it is ok to classify them as sick and order further testing that may help clarifying this up

Decision with Loss Functions

Differentiate between the possible decisions α_i and the possible true classes C_j . The loss may be asymmetric as in the medical diagnosis example:

$$\text{loss}(\text{decision} = \text{healthy} | \text{patient} = \text{sick}) \gg \text{loss}(\text{decision} = \text{sick} | \text{patient} = \text{healthy})$$

Expected loss of making a decision α_i is

$$R(\alpha_i|x) = \mathbb{E}_{C_k \sim p(C_k|x)} [\lambda(\alpha_i|C_k)] = \sum_j \lambda(\alpha_i|C_j) p(C_j|x)$$

with the loss function $\lambda(\alpha_i|C_j)$. The expected loss of a decision is also called the *risk of making a decision*. So, instead of minimizing the misclassification rate (recap)

$$\begin{aligned} p(\text{error}) &= p(x \in \mathcal{R}_1, C_2) + p(x \in \mathcal{R}_2, C_1) \\ &= \int_{\mathcal{R}_1} p(x, C_2) dx + \int_{\mathcal{R}_2} p(x, C_1) dx \\ &= \int_{\mathcal{R}_1} p(x|C_2) p(C_2) dx + \int_{\mathcal{R}_2} p(x|C_1) p(C_1) dx \end{aligned}$$

We minimize the overall risk

$$R(\alpha_i|x) = \mathbb{E}_{C_k \sim p(C_k|x)} [\lambda(\alpha_i|C_k)] = \sum_j \lambda(\alpha_i|C_j) p(C_j|x)$$

3. Wrap-Up

Now, you know:

- The definition of class priors, class conditional probabilities and class posteriors
- How to use Bayes Theorem for classification
- How to calculate the probability of misclassification
- How to obtain optimal decisions using Bayes optimal classifier
- How to generalize decision making using multi-dimensional features and more than 2 classes
- The value of risk minimization and how it relates to misclassification

4. Questions

How do we incorporate prior knowledge on the class distribution?

How can we decide on classifying a query based on simple and general loss functions?

What does “Bayes optimal” mean?

How can we deal with 2 or more classes?

How can we deal with high dimensional feature vectors?

What are the equations for misclassification rate and risk?

Probability Density Estimation

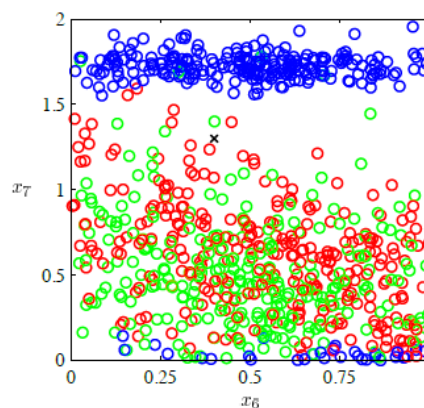
1. Objectives

Understanding on how to estimate $p(x)$. Covering topics:

- Density Estimation
- Maximum Likelihood Estimation
- Non-Parametric Models
- Mixture Models

2. Probability Density Estimation

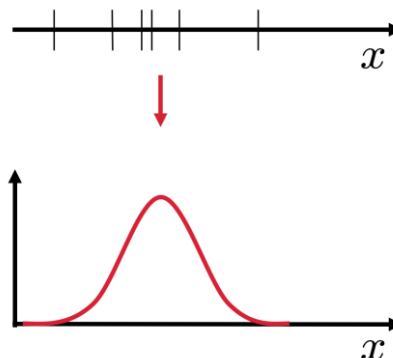
Consider the following training data:



How do we get the probability distributions from this so that we can classify with them? So far, we have seen *Optimal Bayes classification*, based on probability distributions $p(x|C_k)p(C_k)$. The prior $p(C_k)$ is easy to deal with. We can “just count” the number of occurrences of each class in the training data. We need to estimate learn the class-conditional probability density $p(x|C_k)$.

- In Supervised training the input data points, and their true labels (classes) are known
- Goal: Estimate the density separately for each class C_k

Remember that the relationship between the outcomes of a random variable x and its probability $p(X = x)$ is referred to as the probability density, or simply the “density.” E.g., given training data (x_1, x_2, x_3, \dots) estimate $p(x)$.



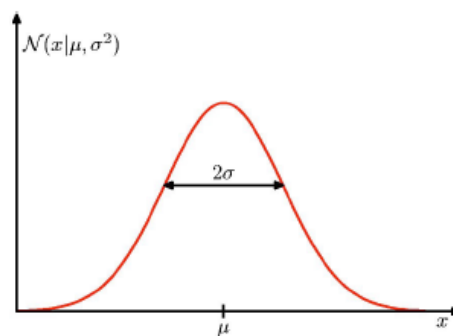
There are 3 main types of Probability Density Estimation models:

- *Parametric probability density estimation*: Involves selecting a common distribution and estimating the distribution parameters from data samples.
- *Non-parametric probability density estimation*: Involves fitting a model to the arbitrary distribution of the data, e.g., kernel density estimation – every known data point in the dataset is used as a parameter.
- *Mixture density models*: Are flexible models that combine parametric and non-parametric estimations.

3. Parametric Density Models

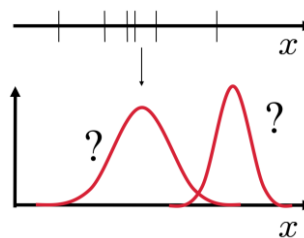
The simple case for parametric density models is the Gaussian Distribution, which is governed by 2 parameters: mean and variance. If we know these parameters, we can fully describe $p(x)$.

$$p(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp \left\{ -\frac{(x - \mu)^2}{2\sigma^2} \right\}$$



- Notation for parametric density models: $x \sim p(x|\theta)$
- Gaussian distribution: $\theta = (\mu, \sigma) \rightarrow x \sim p(x|\mu, \sigma)$

Learning means to estimate the parameters θ given the training data $D = \{x_1, x_2, x_3, \dots\}$:



The *Likelihood* of θ is defined as the probability that the data D was generated from the probability density function with parameters θ :

$$L(\theta) = p(D|\theta)$$

3.1. Maximum Likelihood Method

Consider a set of points $D = \{x_1, x_2, x_3, \dots\}$, we are interested in the likelihood of all data $p(D|\theta)$. The Likelihood can be further written as (by using i.i.d. assumption, see recap below):

$$L(\theta) = p(D|\theta) = p(x_1, \dots, x_N|\theta) = p(x_1|\theta) \cdot \dots \cdot p(x_N|\theta) = \prod_{n=1}^N p_n(x_n|\theta)$$

This leads us to the likelihood estimation

$$\hat{\theta}_{ML} = \operatorname{argmax}_{\theta} p(D|\theta)$$

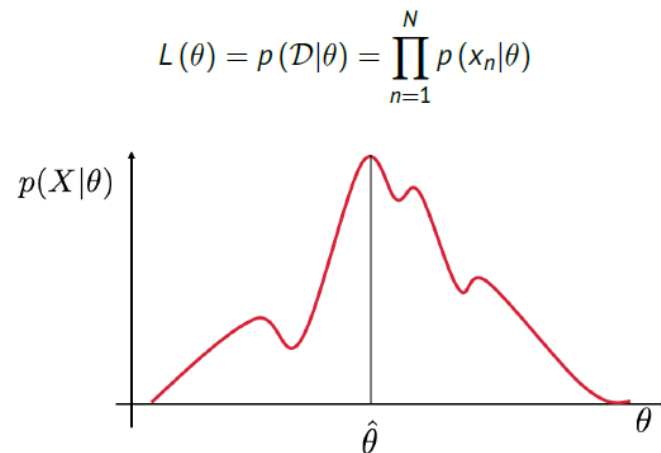
that seeks for the parameter $\hat{\theta}_{ML}$ which best explains the data D . $\hat{\theta}_{ML}$ is a random variable and an estimate based on the available dataset. Consequently, we are interested in its mean value (the most probable value) and its variance. It is more convenient and numerically stable to maximize the log-likelihood w.r.t. θ :

$$LL(\theta) = \log L(\theta) = \log p(D|\theta) = \log \prod_{n=1}^N p_n(x_n|\theta) = \sum_{n=1}^N \log p(x_n|\theta)$$

Because the logarithm is monotonically increasing, it holds

$$\hat{\theta}_{ML} = \operatorname{argmax}_{\theta} \sum_{n=1}^N \log p(x_n|\theta) = \operatorname{argmax}_{\theta} LL(\theta)$$

Maximizing a sum of terms is always easier than maximizing a product; cf., the difficulty of expressing the derivative of a long product of terms.



Recap: Independent and identically distributed (i.i.d.) data

The random variables x_1 and x_2 are independent if:

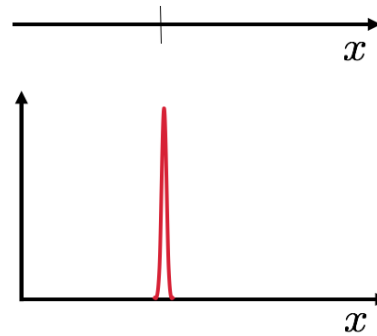
$$P(x_1 \leq \alpha, x_2 \leq \beta) = P(x_1 \leq \alpha)P(x_2 \leq \beta) \quad \forall \alpha, \beta \in \mathbb{R}$$

The random variables x_1 and x_2 are identically distributed if:

$$P(x_1 \leq \alpha) = P(x_2 \leq \alpha) \quad \forall \alpha \in \mathbb{R}$$

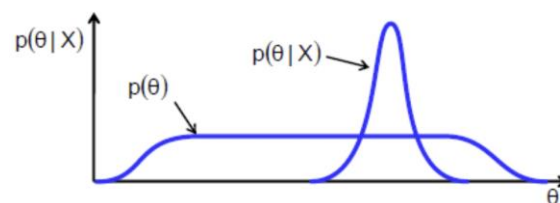
3.2. Likelihood Estimation: Degenerate case

If we try to estimate a single data point $N = 1, D = \{x_1\}$, the probability density of the Gaussian becomes a Dirac δ .



3.3. Bayesian Estimation

The MLE provides point estimates about the model parameters. The Bayesian treatment represents our uncertainty about the parameters θ using a prior over θ and updating our posterior estimation via Bayes rule!



Bayesian estimation/learning of parametric distributions assumes that the *parameters are random variables* too. This allows us to use *prior knowledge* about the parameters. Formalize this as a conditional probability $p(x|D)$:

$$p(x|D) = \int p(x, \theta|D) d\theta$$

$$p(x, \theta|D) = p(x|\theta, D) p(\theta|D)$$

The parametric density $p(x|\theta)$ can be fully determined with the parameters θ , i.e., θ is a sufficient statistic. Hence, we have $p(x|\theta, D) = p(x|\theta)$:

$$p(x|D) = \int p(x|\theta) p(\theta|D) d\theta$$

The probability $p(\theta|D)$ makes it explicit how the parameter estimation depends on the training data and can be calculated using the Bayes theorem

If $p(\theta|D)$ is small in most places, but large for a single $\hat{\theta}$ then we can approximate:

$$p(x|D) \approx p(x|\hat{\theta})$$

this is sometimes also referred to as the **Bayes point**. The more uncertain we are about estimating $\hat{\theta}$, the more the density is averaged across multiple θ .

The problem is in general that it is intractable to integrate out the parameters θ (or only possible to do so numerically).

Example with closed form solution

For Gaussian data distribution, the variance is known and fixed, We estimate the distribution of the mean

$$p(\mu|\mathcal{D}) = \frac{p(\mathcal{D}|\mu) p(\mu)}{p(\mathcal{D})}$$

with prior

$$p(\mu) = \mathcal{N}(\mu_0, \sigma_0^2)$$

The sample mean,

Conjugate Priors

Conjugate Priors are prior distributions for the parameters that do not “change” the type of the parametric model. This means that both the prior and the posterior lie in the same distribution family. For example, as we saw that a Gaussian prior on the mean is conjugate to the Gaussian model. This works here because the product of two Gaussians is a Gaussian, and the marginal of a Gaussian is a Gaussian. Generally, it is not as easy!

Table 3.3.1. *Natural conjugate priors for some common exponential families*

$f(x \theta)$	$\pi(\theta)$	$\pi(\theta x)$
Normal $\mathcal{N}(\theta, \sigma^2)$	Normal $\mathcal{N}(\mu, \tau^2)$	$\mathcal{N}(\varrho(\sigma^2\mu + \tau^2x), \varrho\sigma^2\tau^2)$ $\varrho^{-1} = \sigma^2 + \tau^2$
Poisson $\mathcal{P}(\theta)$	Gamma $\mathcal{G}(\alpha, \beta)$	$\mathcal{G}(\alpha + x, \beta + 1)$
Gamma $\mathcal{G}(\nu, \theta)$	Gamma $\mathcal{G}(\alpha, \beta)$	$\mathcal{G}(\alpha + \nu, \beta + x)$
Binomial $\mathcal{B}(n, \theta)$	Beta $\mathcal{Be}(\alpha, \beta)$	$\mathcal{Be}(\alpha + x, \beta + n - x)$
Negative Binomial $\mathcal{Neg}(m, \theta)$	Beta $\mathcal{Be}(\alpha, \beta)$	$\mathcal{Be}(\alpha + m, \beta + x)$
Multinomial $\mathcal{M}_k(\theta_1, \dots, \theta_k)$	Dirichlet $\mathcal{D}(\alpha_1, \dots, \alpha_k)$	$\mathcal{D}(\alpha_1 + x_1, \dots, \alpha_k + x_k)$
Normal $\mathcal{N}(\mu, 1/\theta)$	Gamma $\mathcal{Ga}(\alpha, \beta)$	$\mathcal{G}(\alpha + 0.5, \beta + (\mu - x)^2/2)$

4. Non-Parametric Models

4.1. Why use non-parametric representations?

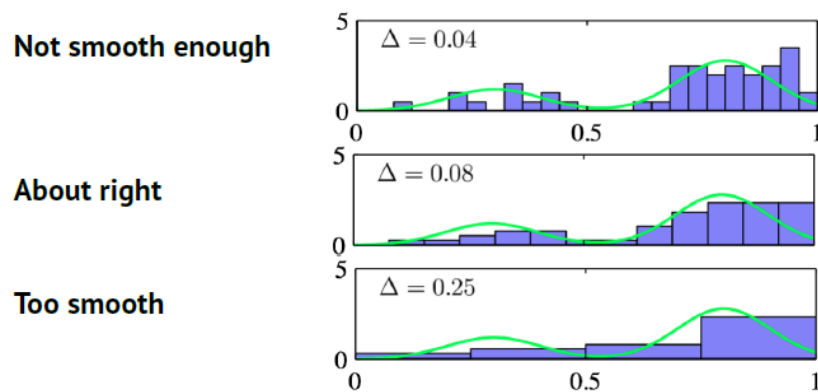
Often, we do not know what functional form the class-conditional density takes (or we do not know what function family we need). Probability density is estimated directly from the data (i.e., without an explicit parametric model):

- Histograms
- Kernel density estimation (Parzen windows)
- K-nearest neighbors.

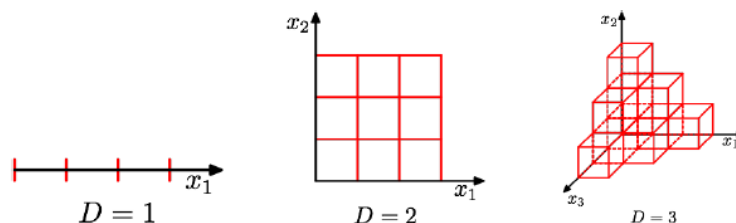
Every data point is a parameter, so non-parametric models have an uncertain and possibly infinite number of parameters.

4.2. Histograms

The feature space is discretized into bins as seen in the next figure:



In the infinitesimal bin-width limit, any probability density can be approximated arbitrarily well. Assuming some locality metric, e.g., Euclidean distance. In High-dimensional feature spaces there is an exponential increase in the number of bins. Hence it requires exponentially much data. This is commonly known as the **Curse of dimensionality** as illustrated in the next figure:



The estimated histogram density has discontinuities due to the bin edges. How to choose the size of the bins? The bin width controls the **smoothness** – the value of the smoothing parameter should not be too large or too small to obtain good results. We will see that this is a general issue that we must keep in mind.

4.3. Kernel Density Estimation

A data point x is sampled from probability density $p(x)$. The probability that x falls in region R is

$$P(\mathbf{x} \in R) = \int_R p(\mathbf{x}) d\mathbf{x}$$

- If R is sufficiently small, with Volume V , then $p(x)$ is almost constant:

$$P(\mathbf{x} \in R) = \int_R p(\mathbf{x}) d\mathbf{x} \approx p(\mathbf{x}) V$$

- If R is sufficiently large, then:

$$P(\mathbf{x} \in R) = \frac{K}{N} \implies p(\mathbf{x}) \approx \frac{K}{NV}$$

where N is the number of total points and K is the number of points falling in the region R .

K-nearest Neighbors

5. Mixture models

6. Wrap-Up

Now you know:

- The parametric, non-parametric, and mixture models.
- More about the likelihood function and how to derive the maximum likelihood estimators for the Gaussian distribution
- What Bayesian estimation is
- Different non-parametric models (histogram, kernel density estimation and k-nearest neighbours)

7. Questions

What are parametric methods, and how to obtain their parameters?

How many parameters have non-parametric methods?

What are mixture models?

Should gradient methods be used for training mixture models?

What is the biggest problem of mixture models?

Clustering

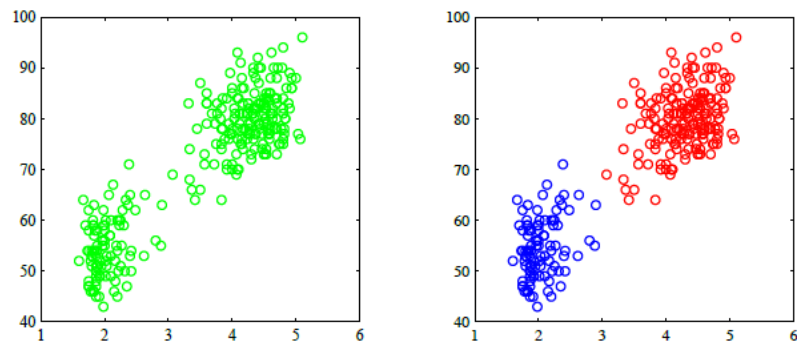
1. Objectives

Make you understand how to find meaningful groups of data points. Covered Topics:

- Taxonomy of clustering algorithms
- K-Means Clustering
- EM Clustering
- Mean Shift Clustering

2. Introduction

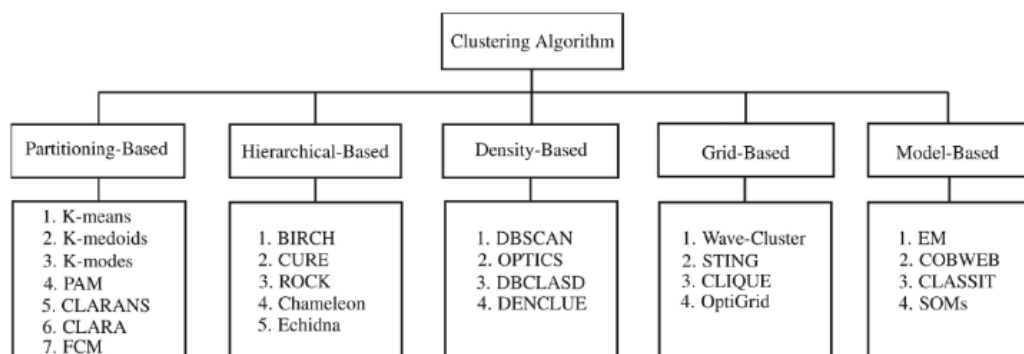
Clustering is a type of unsupervised learning. All data points $\{x_i\}_{i=1}^n$ are unlabeled! The objective is to find groups of similar \mathbf{x} values in the data space and associate these with a discrete set of clusters.



Clustering is often applied in an exploratory way to obtain a better understanding of the data! Found clusters might not correspond to any interpretable classes. Number of clusters is typically unknown and left to the user to decide.

2.1. Overview of Clustering Approaches

There is a multitude of clustering algorithms. Here is a possible categorization with example algorithms:

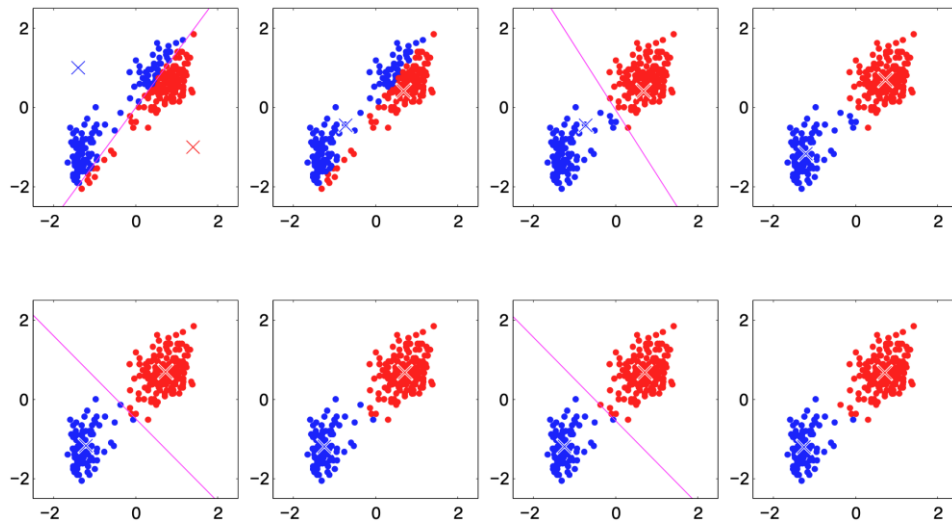


- Partitioning-Based: Divide data points into a number of partitions, where each partition represents a cluster. Each group must contain at least one data point. Each data point must belong to exactly one group

- Hierarchical-Based: Data points are organized in a hierarchical manner depending on the medium of proximity. Distinguish between agglomerative clustering (bottom-up), divisive clustering (top-down).
- Density-Based: Data points are separated based on their regions of density. A cluster grows in any direction that density leads to. Clusters of arbitrary shape can be discovered
- Grid-based: A grid is imposed on the data space which is then used for clustering. Fast processing time - only need to go once through the dataset. Clustering quality depends on imposed grid structure.
- Model-Based. Optimizes the fit between the given data and some (predefined) mathematical model. Assumption: Data is generated by a mixture of underlying probability distributions.

3. K-Means Clustering

The K-Means Algorithm (see below) is guaranteed to converge after finite iterations. However only to local optimum and the result depends on initialization too. Here is an example:



K-Means optimizes the following objective function:

$$J = \sum_{n=1}^N \sum_{k=1}^K r_{nk} \|\mathbf{x}_n - \mu_k\|^2$$

where r_{nk} is an indicator variable that checks whether μ_k is the nearest cluster centre to point x_n :

$$r_{nk} = \begin{cases} 1 & , \text{ if } k = \arg \min_j \|\mathbf{x}_n - \mu_j\|^2 \\ 0 & , \text{ otherwise} \end{cases}$$

- Strengths
 - Simple and fast to compute
 - Converges to local minimum of within-cluster squared error
- Weaknesses
 - How to set k ?
 - Sensitive to initial centers
 - Sensitive to outliers
 - Only detects spherical clusters

Algorithm 1: K-Means Clustering

Input: data points $\{x_i\}_{i=1}^n$

Output: clustered datapoints

1: initialize k arbitrary (cluster means)

2: **repeat**

3: Assign each sample to the closest centroid

5: Adjust the centroids to be the means of the samples assigned to them

4: **until** convergence (no change)

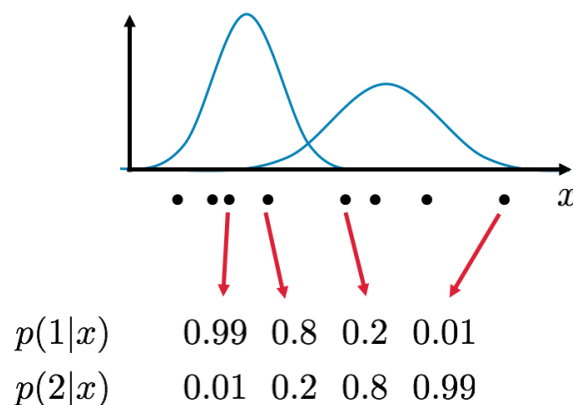
5: **return** clustered datapoints

4. EM-Clustering

EM is model-based clustering algorithm. Data is generated by underlying probability distributions and it's clustered using a method where assignments are “soft”, or flexible. This method involves two key phases: the Expectation step and the Maximization step.

4.1. Expectation Step

Probabilistic assignment of data points to probability distributions:



If we are just given the data points, we don't know which mixture component θ_j generated which data point x :

$$p(x|\theta) = \sum_{j=1}^2 \pi_j p(x|\theta_j)$$

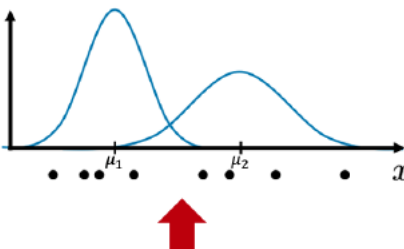
However, we can evaluate the posterior probability that an observed x was generated from, e.g., the first mixture component:

$$\begin{aligned} p(j=1|x, \theta) &= \frac{p(j=1, x|\theta)}{p(x|\theta)} \\ &= \frac{p(x|\theta_1)p(j=1)}{\sum_{j=1}^2 p(x|\theta_j)p(j)} = \gamma_{j=1}(x) \end{aligned}$$

where $\gamma_j(x)$ is called the “responsibility” of component j for x .

4.2. Maximization step

After having calculated the probabilistic assignments, the model parameters can be re-estimated. Using maximum likelihood estimation:

$$\mu_j = \frac{\sum_{n=1}^N p(z_j|x_n) x_n}{\sum_{n=1}^N p(z_j|x_n)}$$


$p(1 x)$	0.99	0.8	0.2	0.01
$p(2 x)$	0.01	0.2	0.8	0.99

Algorithm 2: EM Algorithm

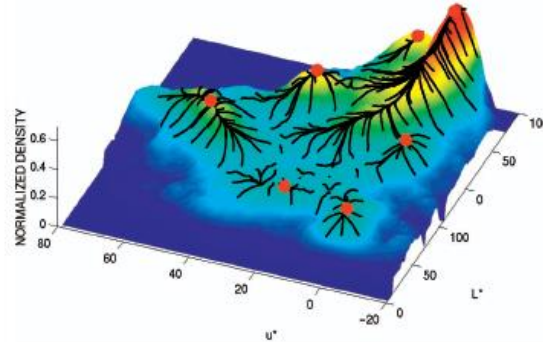
Input: data points $\{x_i\}_{i=1}^n$

Output: clustered datapoints

- 1: initialize the means μ_k , covariances Σ_k and mixing coefficients π_k
- 2: **repeat**
- 3: E-step: Assign each sample to the closest centroid
- 5: Adjust the centroids to be the means of the samples assigned to them
- 4: **until** convergence (no change)
- 5: **return** clustered datapoints

4.3. Mean Shift Clustering

Mean shift clustering is an agglomerative clustering method for finding the modes (maxima) in a cloud of data points where the points are most dense using kernel density estimation and local search. The search path starts at different points and “climbs up the hills” (mean shift clustering is a hill climbing algorithm).



Paths that converge at the same point get the same label. The grand scheme is to start with a kernel density estimate

More precise: The mean shift procedure tries to find the modes of a kernel density estimate through local search.

The black lines indicate search paths starting at different points. Paths converging at the same point get assigned the same label! By starting with kernel density estimate

$$\hat{f}(\mathbf{x}) = \frac{1}{Nh^d} \sum_{i=1}^N k\left(\left\|\frac{\mathbf{x} - \mathbf{x}_i}{h}\right\|^2\right)$$

We can derive the mean shift procedure by taking the gradient of the kernel density estimate.

Algorithm 3: Mean Shift Clustering

Input: data points $\{\mathbf{x}_i\}_{i=1}^n$

Output: clustered datapoints

- 1: initialize the means μ_k , covariances Σ_k and mixing coefficients π_k
- 2: **repeat**
- 3: E-step: Assign each sample to the closest centroid
- 5: Adjust the centroids to be the means of the samples assigned to them
- 4: **until** convergence (no change)
- 5: **return** clustered datapoints

5. Wrap-Up

You know now:

- A taxonomy of clustering algorithms
- Exemplary clustering algorithms:

-
- K-Means
 - Expectation Maximization
 - Mean Shift
-

6. Questions

How can clustering algorithms be categorized?

How do you find clusters using the k-means algorithm?

How are the expectation and maximization steps defined?

What are the difficulties when using the EM algorithm?

How does mean-shift clustering work?

Do the clustering algorithms converge to local or global solutions?

How does density estimation relate to clustering?

Evaluation

1. Objectives

- Make you understand how to evaluate the performance of estimators
- Covered Topics:
 - Underfitting & Overfitting
 - Bias & Variance
 - Cross-Validation

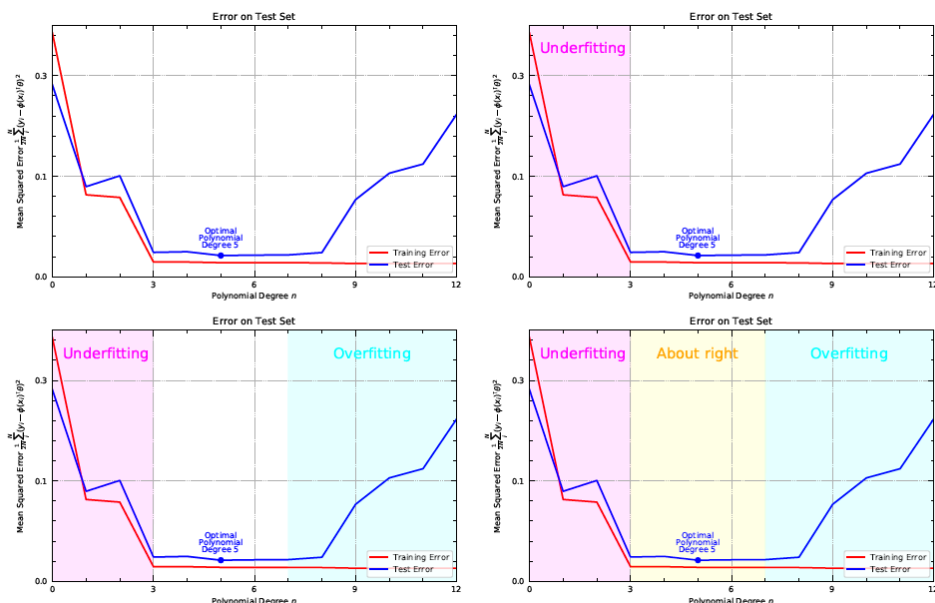
2. Introduction

Evaluation is crucial in the machine learning cycle! So far, we have seen Classification using the Bayes classifier $p(C_k|x) \propto p(x|C_k)p(C_k)$, probability density estimation to estimate the class-conditional densities $p(x|C_k)$ and clustering to find groups of similar data points and associate these with a discrete set of clusters. But how well are we carrying out each tasks? We need a way of performance evaluation:

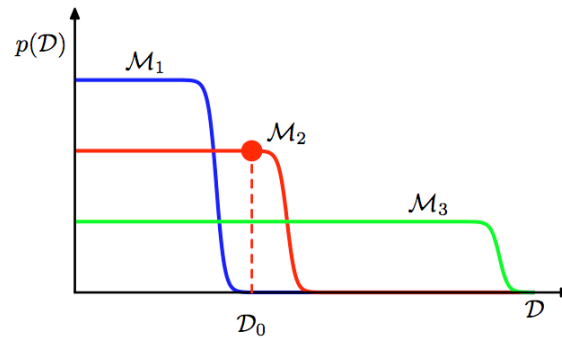
- For density estimation (or really, parameter estimation)
- For the classifier as a whole
- For the clustering algorithm and its hyperparameters

3. Underfitting & Overfitting

Overfitting is everywhere! A small training error does not mean we have a good model. We need to rethink the *model selection*. Meaning we question the number of parameters a.k.a. degree of polynomial n , we also question if our *model class* is sufficiently rich to prevent underfitting or if it is too rich which can then lead to overfitting.

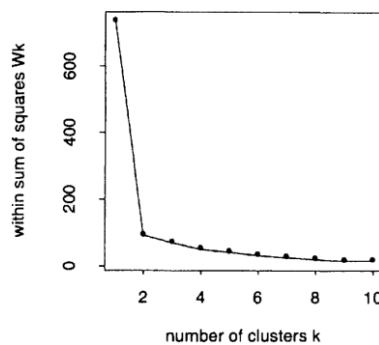


According to Occam's Razor we always choose the simplest model or the model with the smallest model complexity that fits the data.

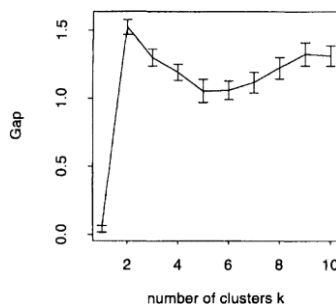


In many clustering algorithms, the number of clusters needs to be predefined as a hyperparameter. If the dataset is complex, the choice might not be trivial. How can we find the optimal number of clusters?

- **Elbow method:** Plot the total within-cluster sum of square (WSS) as a function of the number of clusters. The "elbow point" represents the optimal number of clusters.



- **Gap statistics:** Compare the WSS with that of an appropriate null reference distribution. The gap is the largest for the optimal number of clusters



4. Bias and Variance

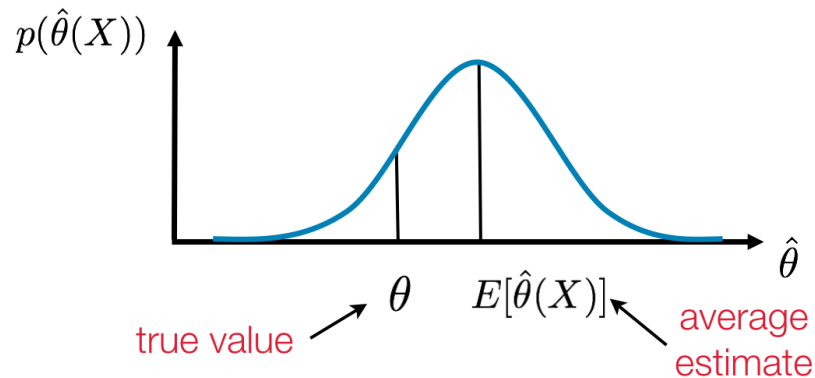
As we saw before, maximum likelihood is just one possible way to estimate a parameter. How can we assess how good an estimator is? Assume we have an estimator $\hat{\theta}$ that estimates the parameter θ from the data set \mathbf{X} . The Bias of an estimator is the expected deviation from the true parameter:

$$\text{bias}(\hat{\theta}) = \mathbb{E}_{\mathbf{X}} [\hat{\theta}(\mathbf{X}) - \theta]$$

The variance of an estimator is the expected squared error between the estimator and the mean estimator

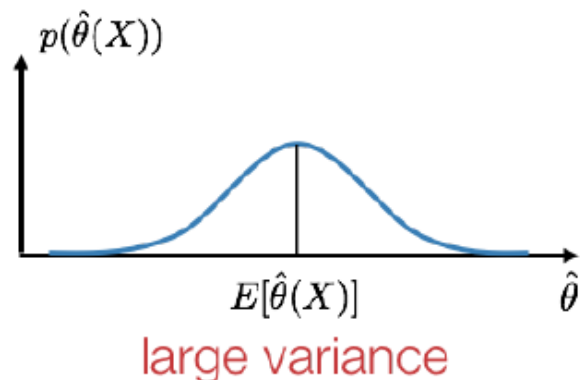
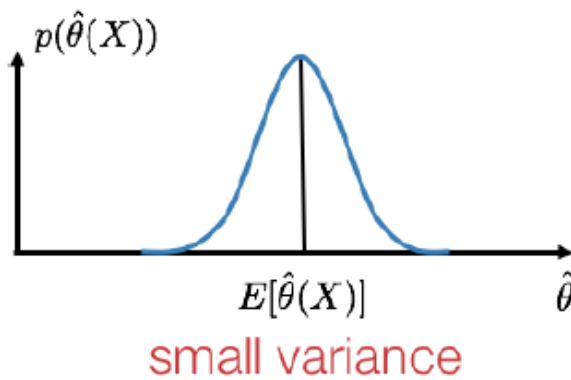
$$\text{var}(\hat{\theta}) = \mathbb{E}_{\mathbf{X}} \left[\left\{ \hat{\theta}(\mathbf{X}) - \mathbb{E}_{\mathbf{X}} [\hat{\theta}(\mathbf{X})] \right\}^2 \right]$$

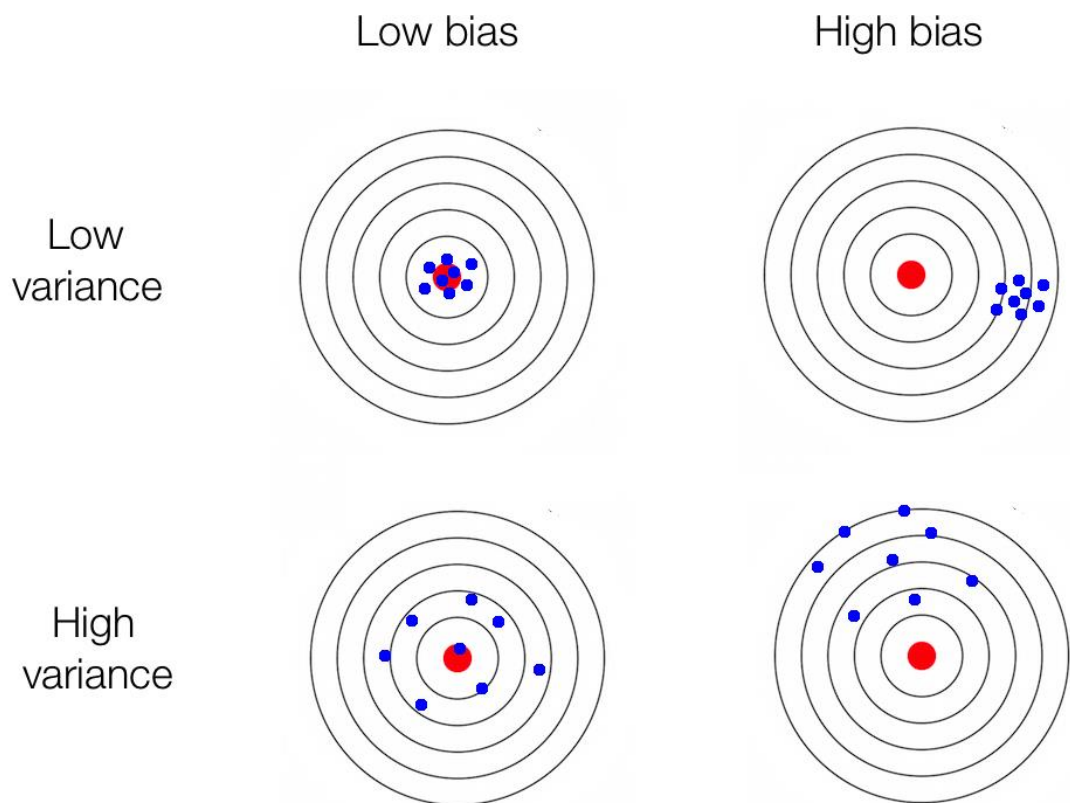
The estimate $\hat{\theta}(\mathbf{X})$ is a random variable, because we assumed that \mathbf{X} is a random sample from a true underlying distribution.



An estimator is biased if the expected value of the estimator $E_{\mathbf{X}}[\hat{\theta}(\mathbf{X})]$ differs from the true value of the parameter θ . Otherwise, it is called unbiased, i.e., $E_{\mathbf{X}}[\hat{\theta}(\mathbf{X})] = \theta$.

Ideally, we want an unbiased estimator with small variance, but in practice this is not that easy as we will see shortly (Bias-Variance-Trade-off).





An estimator with **zero bias** and **minimum variance** is called a **Minimum Variance Unbiased Estimator (MVUE)**. A Minimum Variance Unbiased Estimator which is **linear in the features** is called a **Best Linear Unbiased Estimator (BLUE)**.

Recall Maximum Likelihood Estimation. Consider a set of points $D = \{x_1, \dots, x_N\}$, we are interested in likelihood of all data $p(D|\theta)$. The assumption is that the data is i.i.d. (independent and identically distributed). We maximize the log-likelihood w.r.t θ :

$$\log p(D|\theta) = \log \prod_{n=1}^N p(x_n|\theta) = \sum_{n=1}^N \log p(x_n|\theta)$$

This leads us to the Bias-Variance Trade-off. Typically, you cannot minimize both, bias, and variance, together! Our learning algorithm will only generalize well if we find the right trade-off between bias and variance. Simple enough to prevent overfitting to the particular training data set that we have. Yet expressive enough to be able to represent the important properties of the data. To ensure that our learning algorithm works well, we have to evaluate it on test data. But what if we don't have any?

5. Cross-Validation

The goal is to find a good model (e.g., good set of features). For that we split the dataset into a training set, validation set and test set:

1. Training set: Fit parameters
2. Validation set: Choose model class or single parameters
3. Test set: Estimate prediction error of trained model

The Error/Loss L needs to be estimated on an independent set! How do we select the best model or how does validation work? One method is Cross Validation. We partition data into K sets D_k , use $K - 1$ for *Training* and 1 for *Validation*. After that we compute:

$$\theta_k(\mathcal{M}_j) = \operatorname{argmin}_{\theta \in \mathcal{M}_j} \sum_{\kappa \neq k} \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_\kappa} L_{f_\theta}(\mathbf{x}_i, y_i)$$

$$L_k(\mathcal{M}_j) = \sum_{(\mathbf{x}_i, y_i) \in \mathcal{D}_k} L_{f_{\theta_k}}(\mathbf{x}_i, y_i)$$

Exhaustive Cross Validation: Try all partitioning possibilities

⇒ **Computationally expensive**

Bootstrap: Randomly sample non-overlapping training / validation sets

Another model selection method is K-fold Cross Validation. This is the cheapest reasonable approach

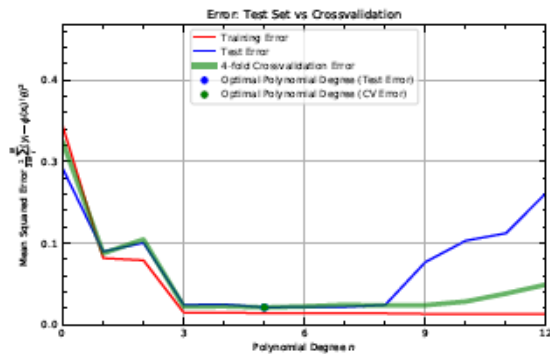
Training Set	Training Set	Validation Set
Training Set	Validation Set	Training Set
Validation Set	Training Set	Training Set

Compute the validation loss and choose model with smallest average validation loss:

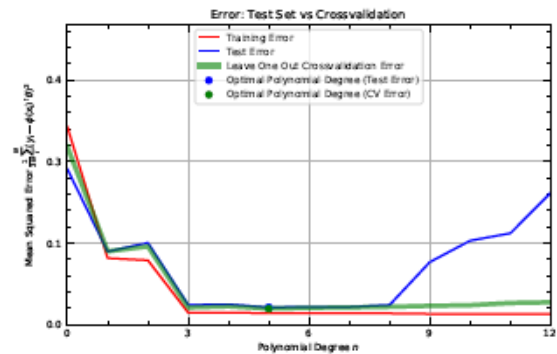
$$\mathcal{M}^* = \operatorname{argmin}_{\mathcal{M}} \frac{1}{K} \sum_{k=1}^K L_k(\mathcal{M})$$

A popular used variant is Leave-one-out cross validation (LOOCV) with $K = N - 1$, where the validation set size is 1. Here is a comparison:

4-fold Cross-Validation



Leave-one-out Cross-Validation



6. Wrap-Up

You know now:

- The meaning of under- and overfitting
- How to find the optimal number of clusters
- How to compute the bias and variance of an estimator
- The definition of MVUE and BLUE
- The bias and variance of Gaussian MLE
- The Bias-Variance trade-off
- How to mimic test data evaluation using cross-validation
- Different variants of cross-validation

7. Questions

What does under- and overfitting mean?

What is Occam's Razor?

How can you prevent under- and overfitting in clustering?

How can you calculate the bias and variance of an estimator?

What is the difference between MVUE and BLUE?

Are maximum likelihood estimators always unbiased?

What is the bias-variance trade-off?

What is leave-one-out cross-validation? What do we need it for?

Questions

1. Introduction to Machine Learning

1.1. Self-Test Questions

What are some of Machine Learning applications?

It's amazing what modern ML methods can do! For example:

- Image Generation (DALL-E, Midjourney): Generating photorealistic images from text prompts
- Text Generation (ChatGPT): Generate human like responses
- Intelligent Robots (ANYmal, Boston Dynamics): Perform complex behaviors such as juggling

When can we benefit from using Machine Learning methods?

We generate much more data than we can analyze manually (Era of big data). No human being can deal with this data avalanche. Machine Learning (ML) is inherently data driven. Data is at the core of ML. The goal of ML is to design general-purpose methodologies to extract valuable patterns from data, ideally without much domain-specific expertise.

What are the different types of learning?

- **Supervised Learning:** Data with labels (input/output pairs)
 - Regression
 - Classification
- **Semi-supervised learning:** Data with and without labels
 - Clustering
 - Dimensionality reduction
 - Density estimation
- **Reinforcement learning:** Learning by doing.
- **Unsupervised Learning:** Data without label.

2. Statistics & Probabilities Refresher

2.1. Self-Test Questions

What is a random variable?

A random variable is a random number determined by chance. Its value is unknown and/or could change e.g., the temperature outside the room at the current time. More formally, it is drawn according to a probability distribution.

- Discrete random variable: Finite/Countable sample/state space
- Continuous random variable: Infinite/Uncountable sample/state space

What is a distribution?

A probability distribution describes the probability (mass/density) that the random variable will be equal to a certain value. The probability distribution can be given by the physics of an experiment (e.g., throwing dice).

Probability mass function (pmf) is defined as the probability that X (discrete random variable) would be equal to a sample x :

$$p(x) = P(X = x) \text{ with } 0 \leq p(x) \leq 1 \text{ and } \sum_{x \in \mathcal{X}} p(x) = 1$$

Probability density function (pdf) or density is defined as the **relative likelihood** that X (continuous random variable) would be **close** to a sample x :

$$p(x), \text{ with } p(x) \geq 0 \text{ and } \int_{\mathcal{X}} p(x) dx = 1$$

What is a Gaussian distribution?

The gaussian distribution is defined as:

$$p(x) = N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{1}{2\sigma^2}(x-\mu)^2}$$

The distribution of the sum of N i.i.d. (independent and identically distributed) random variables becomes increasingly Gaussian as N grows.

What is an expectation?

An expectation is the average value of some function $f(x)$ under a probability distribution $p(\cdot)$:

$$\mathbb{E}_{x \sim p(\cdot)}[f(X)] = \mathbb{E}_X[f] = \mathbb{E}[f] = \begin{cases} \sum p(x)f(x) & \text{discrete case} \\ \int p(x)f(x)dx & \text{continuous case} \end{cases}$$

We note $\mu = \mathbb{E}[X]$.

What is a joint distribution?

A joint distribution is the probability distribution that encompasses two or more random variables.

What is a conditional distribution?

What is a distribution with a lot of information?

How to measure the difference between distributions?