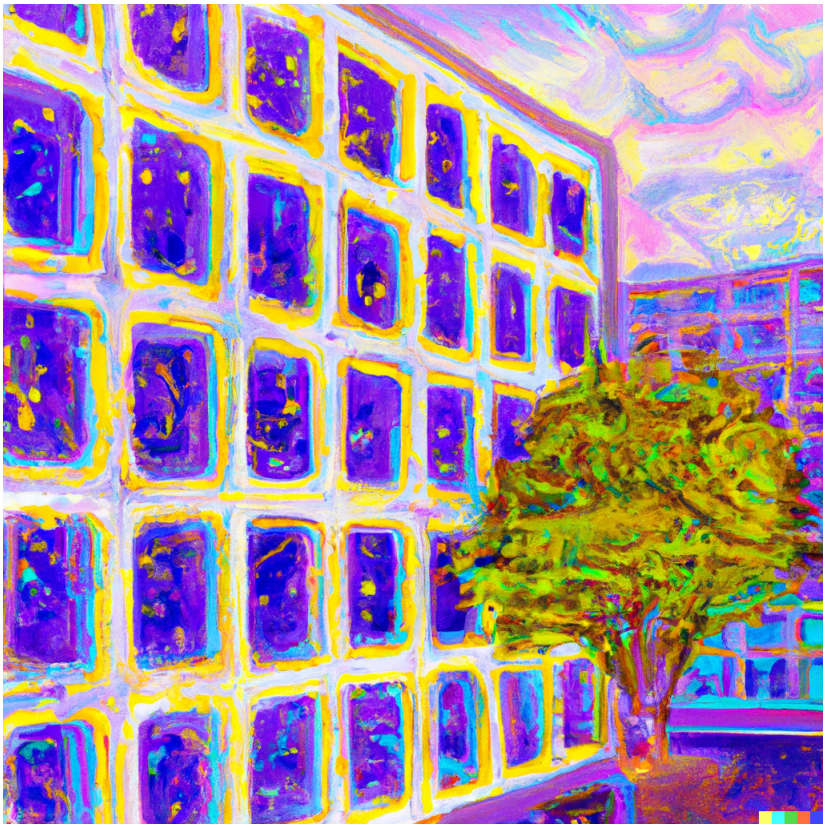
Advanced Machine Learning
Summary

Tim Launer

December 30, 2022



Contents

1 Machine Learning Fundamentals

1.1 Anomaly Detection with PCA and EM

We will introduce a method for anomaly detection based on principal component analysis (PCA), Gaussian mixture models (GMM) and the expectation maximization algorithm. Anomaly or outlier detection is relevant in a variety of use cases, such as detecting fraudulent transactions, detecting sick cells in tissue or identifying suspicious behaviour.

Problem Formalization - Anomaly Detection
Given a set $X = \{x_1, \dots, x_n\}$ from a normal class $N \subseteq \mathbb{R}^D$, compute a function $\phi : \mathbb{R}^D \to \{0, 1\}$ such that $\phi(x) = 1$ if and only if $x \notin N$.

Strategy
An anomaly is an unlikely event, thus we fit a model of a parametric family of distributions $\{p(\cdot|\theta) : \theta \in \Theta\}$ onto a projection in a low dimensional space.
It has been observed that linear projections of high dimensional distributions onto low dimensional spaces resemble Gaussian distributions. Thus, we will:

- project $X = \{x_1, \dots, x_n\}$ onto $\mathbb{R}^d$ with $\pi : \mathbb{R}^D \to \mathbb{R}^d$, where $d << D$.
- fit a GMM p_θ to projected data
- assign each point $x \in \mathbb{R}^D$ the anomaly score $-\log(p_\theta(\pi(x))$

In general, we must keep in mind that very often mathematically tractable solutions do not reflect reality. Thus, there will always be a trade-off between fidelity and tractability of our solutions. This is a recurring theme throughout the lecture.

1.1.1 Dimensionality Reduction with PCA

Principal component analysis tries to find, given $X = \{x_1, \dots, x_n\}$, a linear projection $\pi : \mathbb{R}^D \to \mathbb{R}^d$ with $d << D$ such that $\pi(X)$ has "sufficiently large" variance.

First, we want to deal with the simple case of $d = 1$. Thus, we want to find a projection of the data onto the real line $\pi : \mathbb{R}^D \to \mathbb{R}$. This means that

$$\pi(x) = u_1^T x$$

for some $u_1 \in \mathbb{R}^D$ with $||u_1|| = 1$. We can see that the mean of the projection data is $u_1^T \bar{X}$, where $\bar{X} = \frac{1}{n} \sum_{x \in X} x$ and that its variance is

$$\frac{1}{n} \sum_{x_i} \left(u_1^T \bar{X} - u_1^T x_i\right)^2 = u_1^T \left(\frac{1}{n} \sum_{x_i} (\bar{X} - x_i)(\bar{X} - x_i)^T \right) u_1 := u_1^T S u_1.$$

S is the **covariance matrix** of the data-set. We now want to compute u_1 to maximize the variance of the projection. This can be solved using the Lagrange method for optimization with constraints:
Finding the $\max_{u_1 \in \mathbb{R}^D} u_1^T S u_1$ s.t. $||u_1|| = 1$ is equivalent to optimizing

$$\begin{aligned} Ł(u_1) &= f(x) + \lambda g(x) = u_1^T S u_1 + \lambda(1 - u_1^T u_1) \\ &= u_1^T (S - \lambda \mathcal{I}) u_1 + \lambda \end{aligned}$$

Differentiating w.r.t u_1 and setting equal zero we can easily find:

$$0 \doteq \frac{\partial Ł}{\partial u_1} = (S - \lambda \mathcal{I}) u_1 \iff S u_1 = \lambda u_1$$

Thus, the optimal u_1^* is an eigenvalue of S.
The intuitive follow up question is then: which eigenvalue exactly?
Note:

$$u_1^{T*} S u_1^* = u_1^{T*}(\lambda u_1^*) = \lambda ||u_1^*|| = \lambda$$

Since we want to maximize this expression, λ must be the maximal eigenvalue of S.

Now we can generalize this result to the case $d \geq 1$. We proceed as follows:

- compute u_1^* from X as above.
- let $X_1 = x - proj_{u_1^*} x : x \in X$.
- compute u_2^* from X_1 from X_1 as above.
- let $X_2 = x - proj_{u_1^*} x : x \in X_1$.
- $\cdots$
- compute u_d^* from X_{d-1} as above.
- Define $\pi(x) = (x^T u_1^*, x^T u_2^*, \ldots, x^T u_d^*)$

Note that the definition of a linear projection subtraction is $x - proj_u(x) = x - u(u^T x)$.

We have now found a way to compute a linear projection of the data into a low dimensional space using PCA. Now, we must compute a Gaussian Mixture model on the projected data.

1.1.2 Gaussian Mixture Models with Maximum Likelihood

First, we introduce the model family.

Definition - GMM:
A GMM with K components consists of all distributions whose PDF is of the form

$$p(x) = \sum_{k \leq K} \pi_k \cdot \mathcal{N}(x | \mu_k, \Sigma_K)$$

where $\pi_1, \ldots, pi_k \in \mathbb{R}^+$ s.t. $\pi_1 + \cdots + \pi_k = 1$ are weights assigned to each Gaussian distribution.

Furthermore, let Z be a random variable with range in $1 \ldots, K$ and whos pdf is $Pr(Z = k) = \pi_k$.
The parameters we are concerned with are $\theta = \{\pi_k, \mu_k, \Sigma_k\}$.

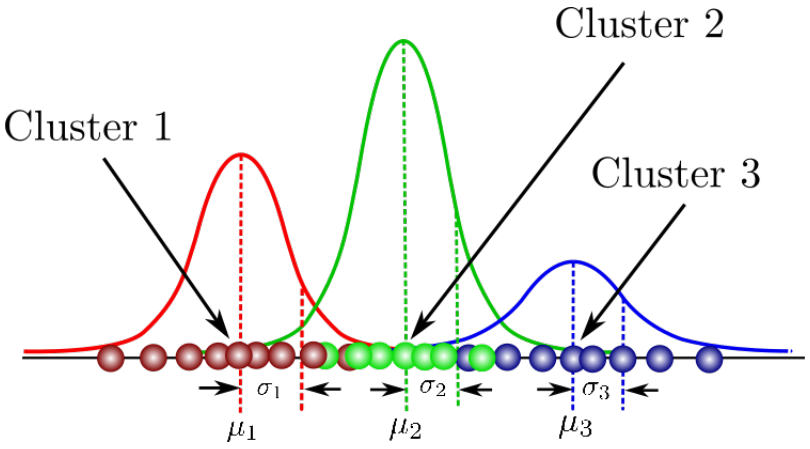


Figure 1: Schematic of a Gaussian Mixture Model in 1 dimension.

Maximum Likelihood Fitting:
Given a dataset $X = \{x_1, \dots, x_n\}$ and a GMM with K components, we want to choose the parameters θ such that the model p_θ maximizes the log-likelihood of the data under the model:

$$\log p_\theta(X) = \sum_{i \leq n} \log p_\theta(x_i) = \sum_{i \leq n} \log \left(\sum_k \pi_k \cdot \mathcal{N}\left(x_i | \mu_k, \Sigma_k\right) \right)$$

If we the $z_i \in \{1, \dots, K\}$ denoting the Gaussian where x_i comes from, then the above expression for the log-likelihood would be much more mathematically tractable:

$$\log p_\theta(X, Z) = \log \left(\Pi_i p_\theta(x_i, z_i) \right) = \sum_i \log \pi_{z_i} + \log \mathcal{N}\left(x_i | \mu_{z_i}, \Sigma_{z_i}\right)$$

We will now look at a popular trick to decompose the intractable expression for $\log p_\theta(X)$ which we want to maximize w.r.t θ into more tractable terms. This will intuitively lead us directly to the expectation maximization (EM) algorithm.

Note that in the following, all expectations are taken over $Z \sim q$, where q can be any distribution for the variable $Z = \{z_1 \dots, z_n\}$. Recall, that the random variable set Z specifies the cluster each data-point comes from, such that $z_i = k$ tells us that a data-point x_i came from the Gaussian cluster k with parameters μ_k, Σ_k.

$$\log p_\theta(X) = \mathbb{E}\left[\log p_\theta(X)\right] = \mathbb{E}\left[\log \left(\frac{p_\theta(X, Z)}{p_\theta(Z|X)} \frac{q(Z)}{q(Z)} \right)\right] = \mathbb{E}\left[\log p_\theta(X, Z)\right] - \mathbb{E}\left[\log q(Z)\right] + \mathbb{E}\left[\log \left(\frac{q(Z)}{p_\theta(Z|X)} \right)\right]$$

The first equality holds because q is not a distribution over X, the second by the rules for conditional probabilities, the third by properties of the logarithm. We now define the functions

$$M(q, \theta) := \mathbb{E}\left[\log p_\theta(X, Z)\right] - \mathbb{E}\left[\log q(Z)\right] = \mathbb{E}\left[\log \left(\frac{p_\theta(X, Z)}{q(Z)} \right)\right]$$

and

$$E(q, \theta) := \mathbb{E}\left[\log \left(\frac{q(Z)}{p_\theta(Z|X)} \right)\right]$$

Note that these functions are both probability distributions over the data X, taking on different shapes depending on the arguments. These have the properties (proven in exercises):

- 1) $E(q.\theta) \geq 0$ (by Jensen's Inequality)
- 2) $E(q^*, \theta) = 0$ for $q^* = \arg\min_q E(q, \theta) = p_\theta(Z|X)$
- 3) $\max_\theta M(q^*, \theta) \geq \log p_\theta(X)$

The properties and the expression we found directly lead us to an optimization procedure recall, we want to maximize $\log p_\theta(X)$): First, we choose an initial θ_0. By observation 1) and 2), we know that M is always less than $\log p_\theta(X)$, except when q is optimal, in which case $M(q^*, \theta) = \log p_\theta(X)$. This means that $M(q, \theta)(X)$ is an evidence lower bound on X. By observation 3), we also know that for q^*, if we find $\theta^* = \max_\theta M(q^*, \theta)$, we will get $p_{\theta^*}(X) \geq p_\theta(X)$. We can thus maximize $p_\theta(X)$ iteratively by first finding q for a given θ s.t. E is zero and then finding θ^* for a given q^*, which will yield a larger $p_{\theta^*}(X)$. Note here the greater or equal sign... convergence to a maximum $p_\theta(X)$ is guaranteed but the speed is not. We could get stuck.

Here is the algorithm more formally:

- Initialize θ_0
- For $t = 1, 2, \dots$ do:
  Expectation Step: assign $q^* = \arg\min_q E(q, \theta_{t-1})$
  Maximization Step: compute $\theta_t = \arg\max_\theta M(q^*, \theta)$

There is another intuitive way of interpreting the algorithm: at each step, we first assign each point to its most likely cluster and then reassign the parameters of all the Gaussian clusters to better fit the assigned points.

- The expectation step assigns a distribution to Z. This is equivalent to assigning each point to a Gaussian distribution in the GMM (cluster).
- The maximization step calculates the parameters $\theta = \{\mu_k, \Sigma_k\}$ to make each Gaussian a better fit for the assigned points.

Designing the Anomaly detector
Two interesting things were mentioned. First, we define the anomaly detector as the function $\phi : \mathbb{R}^D \to \{0, 1\}$ which should output 1 if the anomaly score $-\log p_\theta(\pi(x)) > \tau$ for some threshold τ, and $\phi(x) = 0$ otherwise. When training, we want to use a validation metric that gives us both high precision $p = \frac{|C \cap A|}{|C|}$ and high recall $r = \frac{|C \cap A|}{|A|}$, where A is the set of anomalies and C the set of points our classifier marks as anomalies. Thus, we introduced the F_1 score:

$$F_1 = \frac{2}{\frac{1}{p} + \frac{1}{r}}$$

1.2 Representation, Measurements and Data Types

I will write down here a thoughts from the introductory lecture by Prof. Buhmann that stand out to me.

In general, there are two ways of thinking:

- The deductive way: we start from axioms and deduce observable consequences, testing the axiomatic theory.
- The inductive way: we start from observations (data) and then try to identify general laws (or axioms).

Machine learning is obviously about inductive thinking.
Note the difficulty of defining intelligence (when thinking about what artificial intelligence should be). Prof. Buhmann considers the ability to do counterfactual reasoning and planning to be intelligence.

Algorithm run-time, memory etc have been studied widely, but their robustness and generalizations have not been studied. Learning algorithms are methods (statistical) to explore reality. A lot of work still needs to be done on their generalization and robustness. It is very difficult to talk about correctness of algorithms that process inputs with noise and compute random variables as outputs.

One of the essential questions of machine learning algorithms is validation - how can we know that our algorithms operating on huge data-sets are actually doing the correct thing? Note that verification is guaranteeing performance on seen data, validation aims to quantify how well an algorithm will do on unseen data.
Furthermore, we want to design algorithms that will perform well (whatever that means?) on problems that humans cannot solve - unsupervised learning.

When developing machine learning algorithms, you have to take a very puristic position about what exactly you want to guarantee with your methods. Indeed, in many high-complexity fields, many scientific results are wrong (50% of papers in psychology).
In machine learning pipelines, often preprocessing of the data makes the most significant difference. When we finally analyse the result of an algorithm pipeline in a high complexity problem where "correctness" is not defined, figuring out the sensitivity of each component in the pipeline is important for validation and improvement. It is important to put a lot of attention onto the weakest link of a pipeline.

A model where the signal does not fit into a humans memory cannot be understood. The simpler the signal for the model (an equation to describe the theory), the harder it often is to predict from it. For example, in weather forecasting, Navier Stokes equations are still used for prediction. However, much more data enters now into our boundary conditions, which enables more precise estimation of the solutions. For equations

which have no analytical solutions, data on boundary conditions are very important. The struggle is that in high complexity fields, without tools the scientific method is hard.

Representations
A first though - tuples of numbers are not neccessarily vectors. Vector spaces require certain operations (addition and scaling) and sometimes these operations make no sense for certain tuples.
First, some definitions:
Measurement Space is the mathematical space in which the data are represented (numerical, boolean, categorical spaces).
Features are derived quantities or indirect observations which often significantly compress the information content of measurements (e.g. edges in images).
Note, The selection of a specific feature space predetermines the metric to compare data; this choice is the first significant design decision in a machine learning system.

The Learning Problem
Here is the basic learning problem:

- Representation of objects (data representation)
- Definition / modeling of structure and hypothesis classes
- Optimization - search for preferred structures / models
- Validation - are the stuctures indeed in the data or are they explained by fluctuations or method biases?

We are looking for a function $f \in \mathcal{C}$ out of the hypothesis class (or solution space) so that $f : \mathcal{X} \to \mathcal{Y}$.
The **Loss Function** $\mathcal{Q}(y, f(x))$ measures the deviation between dependent variables y and our prediction $f(x)$ - it is essentially a compact parameterization of a posterior. In choosing a loss function we implicitly make assumptions about the probabilistic law of the posterior.
The **Expected Risk** for a random variable X is:

$$R(f) = \mathbb{E}_X\left[R(f, X)\right] = \int_{\mathcal{X}} R(f, X)P(X)dX = \int_{\mathcal{Y}}\int_{\mathcal{X}} \mathcal{Q}(y, f(x))P(X, Y)dXdY$$

with

$$R(f, X) = \int_{\mathcal{Y}} \mathcal{Q}(y, f(x))P(X|Y)dY$$

The **Empirical Risk** is an approximation of the expected risk of our approximation. We split our data into training, validation and test data. Validation data is used as a guide to select an estimator (one with good generalization). Test data is used to evaluate a predictor (never go back to changing the estimator after using test data!).
Using the defintion of risk, we can now specify the training, validation and test error:

$$\hat{R}(\hat{f}, Z) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{Q}(Y_i, \hat{f}(X_i)),$$

where $Z = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$ for train, validation and test data respectively. Note that the empirical risk (above on test data) is not the expected risk but only an approximation we use.

Measurements and Objects
We are given an object space $\mathcal{O}$. A measurement X will map an object set into a domain $\mathbf{K}$ (for example into $\mathbb{R}$) $X : \mathcal{O} \to \mathbf{K}$. For more examples see lecture 2B slides.
If you use polyadic data $\mathcal{O}_1 \times \mathcal{O}_2 \times \mathcal{O}_3$ such as {test persons}×{traits}×{behaviours} and you measure them as vectors, you might loose a lot of information.
If you map certain types of data into a specific representation, information about it might not be able to

be easily learn-able. For example, sometimes data easily separable in 20 dimensions might be a mess when projected into 2 dimensions. Furthermore, different scales in data can introduce problems in algorithms, thus one should normalize data to get rid of various scales.
Invariances in data are important: if the measurements are invariant under a set of transformation then the mathematical definition of structure should obey the same invariances. Otherwise, our structure search procedure breaks the symmetry in an a priori (not data-dependent) way. In simpler terms, if the data has a certain scale with invariances (Kelvin, ratio scale), our solution structure (e.g. estimator) must obey the same invariances.

scale type	transformation invariances
nominal	$\mathcal{T} = \{f : \mathbb{R} \to \mathbb{R} \mid f \text{ bijective}\}$
ordinal	$\mathcal{T} = \{f : \mathbb{R} \to \mathbb{R} \mid f(x_1) < f(x_2),\ \forall x_1 < x_2\}$
interval	$\mathcal{T} = \{f : \mathbb{R} \to \mathbb{R} \mid f(x) = ax + c,\ a \in \mathbb{R}^+,\ c \in \mathbb{R}\}$
ratio	$\mathcal{T} = \{f : \mathbb{R} \to \mathbb{R} \mid f(x) = ax,\ a \in \mathbb{R}^+\}$
absolute	$\mathcal{T} = \{f : \mathbb{R} \to \mathbb{R} \mid f \text{ is identity map}\}$

Figure 2: Formal characterisation of different scale types and their invariance properties.

1.3 Density Estimation

Definition (from Wikipedia)
In statistics, probability density estimation or simply density estimation is the construction of an estimate, based on observed data, of an unobservable underlying probability density function.

Simply, as outlined in the figure below, the inference problem is as follows:
We are given a dataset $X = \{x_1, \dots, x_n\}$ following some parametric distribution $p_\theta(X)$ parameterized by an unknown parameter θ. We want to build an estimator that finds an approximation $\hat{\theta}$ for the true parameter θ explaining the data. Our goal is to minimize the expected difference between the estimation $\hat{\theta}$ and the ground truth θ. However, as we will see shortly, this is impossible. Therefore, we instead try to find asymptotically efficient, asymptotically normal and consistent approximations. We will see shortly what this means.

Recap: Parametric vs. Non-Parametric Inference
Recall our fundamental problem: "given $X = \{x_1, \dots, x_n\} \sim F$, find the distribution F". We now want to construct a statistical model $\mathcal{H}$, which is a set of distributions (or densities or regression functions), to solve this. A **Parametric model** is a set $\mathcal{H}$ of such functions which can be characterized by finitely many parameters

$$\mathcal{H} = \{p(x; \theta) \,:\, \theta \in \Theta\},$$

where Θ is the model parameter space. An example would be the statistical model set of Gaussian distributions characterized by parameters $\mu \in \mathbb{R}^d$ and $\Sigma \in \mathbb{R}^{d \times d}$, thus $\Theta = \mathbb{R}^d \times \mathbb{R}^{d \times d}$. We implicitly assume that the distribution F lies in the parametric family $\mathcal{H}$.
In contrast, a **Non-Parametric Model** is a set $\mathcal{H}$ which cannot be characterized by finitely many parameters, such as for example $\mathcal{H} = \{\text{all CDFs}\}$. Once we have chosen a model, we must choose an estimator $p \in \mathcal{H}$ to best explain our data. In the following, we will concern ourselves with parametric inference. Note that when working with statistical models, choosing the correct model is key.

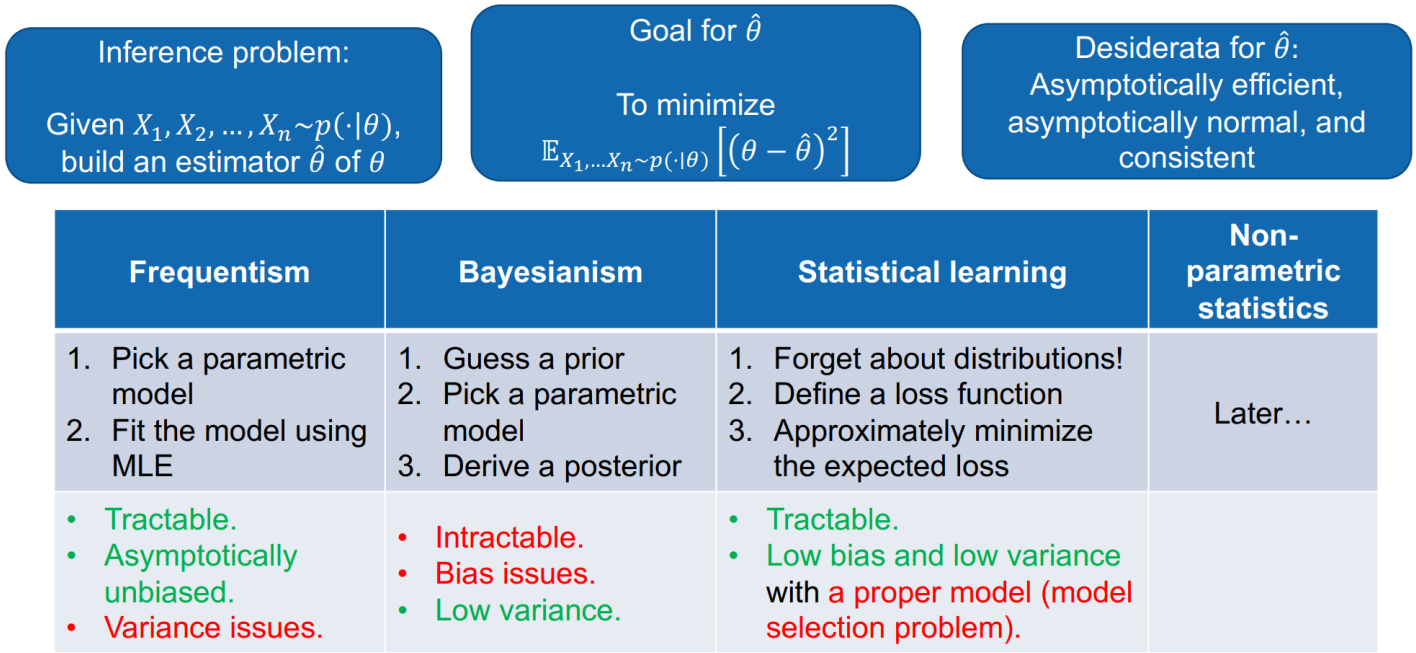


Figure 3: Statistical Inference overview and standard approaches.

1.3.1 The Rao-Cramer Bound and Optimal Estimators

The **Rao-Cramer Bound** shows that in general, even unbiased estimators $\hat{\theta}$, s.t. $\mathbb{E}\left[\theta - \hat{\theta}\right] = 0$, are not optimal and thus cannot retrieve the original θ (never have variance 0):

$$\mathbb{E}_X\left[(\theta - \hat{\theta})^2\right] \geq \frac{1}{I_n(\theta)} = \left(\mathbb{E}\left[\left(\frac{\partial}{\partial\theta}\log p(X|\theta\right)^2\right]\right)^{-1},$$

where $I_n(\theta)$ is the **Fisher Information**.

Proof.
First, we need to define the score:

$$\Lambda := \frac{\partial}{\partial\theta}\log p(X|\theta) = \frac{\frac{\partial}{\partial\theta}p(X|\theta)}{p(X|\theta)}.$$

The score has the following properties:

- 1) $\mathbb{E}_X[\Lambda] = \int p(X|\theta)\frac{\frac{\partial}{\partial\theta}p(X|\theta)}{p(X|\theta)} = \frac{\partial}{\partial\theta}1 = 0$
- 2) $\mathbb{E}_X\left[\Lambda\hat{\theta}\right] = \int p(X|\theta)\frac{\frac{\partial}{\partial\theta}p(X|\theta)}{p(X|\theta)}\hat{\theta}(X) = \frac{\partial}{\partial\theta}\mathbb{E}_X\left[\hat{\theta}\right] = \frac{\partial}{\partial\theta}\text{bias}(\hat{\theta} + 1$

Let $\hat{\theta}$ be unbiased. First, note:

$$\text{Cov}(\Lambda, \hat{\theta})^2 = \mathbb{E}\left[(\Lambda - \mathbb{E}[\Lambda])(\hat{\theta} - \theta)\right]^2 = (\mathbb{E}\left[\Lambda\hat{\theta} - \Lambda\theta\right])^2 = (\mathbb{E}\left[\Lambda\hat{\theta}\right] - \theta\mathbb{E}[\Lambda])^2 = 1$$

Furthermore:

$$\text{Cov}(\Lambda, \hat{\theta})^2 = \mathbb{E}\left[(\Lambda - \mathbb{E}[\Lambda])(\hat{\theta} - \mathbb{E}\left[\hat{\theta}\right])\right]^2 \leq \mathbb{E}\left[\Lambda^2\right]\mathbb{E}\left[(\hat{\theta} - \theta)^2\right]$$

Thus, we have shown that $1 \leq \mathbb{E}\left[\Lambda^2\right] \mathbb{E}\left[(\hat{\theta}-\theta)^2\right]$ which implies:

$$\mathbb{E}\left[(\hat{\theta}-\theta)^2\right] \geq \frac{1}{\mathbb{E}\left[\Lambda^2\right]} = \frac{1}{I_n(\theta)}$$

We will not prove the last equality.
We have shown that there is no hope to recover θ exactly. So, what can we aim for instead??

- **Asymptotic efficiency** for unbiased estimators: $\lim_{n\to\infty} \mathbb{E}\left[(\hat{\theta}-\theta)^2\right] I_n(\theta) = 1$.
- **Consistency** $\lim_{n\to\infty} Pr(|\hat{\theta}-\theta| > \epsilon) = 0$ for all $\epsilon > 0$.
- **Asymptotic Normality** (tractable confidence intervals) $\hat{\theta}_n \sim \mathcal{N}\left(\hat{\theta}_n|\theta,\, se^2\right)$

These are our desiderata for the estimator $\hat{\theta}$. We will now explore the popular approaches.

1.3.2 Frequentism

Frequentism follows two steps: first, we choose a parametric model $\mathcal{H} = \{p(x;\theta) : \theta \in \Theta\}$. Then, we choose the estimator $p_\theta \in \mathcal{H}$ which maximizes the likelihood of the data.
We define the likelihood function as $\mathcal{L}_n(\theta) = \prod_{i=1}^n p_\theta(X_i)$. Then, the **Maximum Likelihood Estimator (MLE)** is the $\hat{\theta}_n$ which maximizes the likelihood function $\hat{\theta}_n = \arg\max_\theta \mathcal{L}_n(\theta)$. It can be shown that MLE is unbiased and fulfills the desiderata. However, the frequentist approach often suffers from unnecessarily large variance of the estimator.
Recall the **Bias-Variance Decomposition**:

$$\begin{aligned}\mathbb{E}\left[(\hat{\theta}-\theta)^2\right] &= \mathbb{E}\left[(\theta - \mathbb{E}\left[\hat{\theta}_n\right] + \mathbb{E}\left[\hat{\theta}_n\right] - \hat{\theta}_n)^2\right] = \left(\theta - \mathbb{E}\left[\hat{\theta}_n\right]\right)^2 + \mathbb{E}\left[(\mathbb{E}\left[\hat{\theta}_n\right] - \hat{\theta}_n)^2\right] \\ &= \text{bias}(\hat{\theta}_n)^2 + \text{var}(\hat{\theta}_n)\end{aligned}$$

Thus, we can trade bias for variance! This leads us to Bayesian estimation.

1.3.3 Bayesian Inference

Bayesianism follows the following approach:

- 1) Impose a prior on the parameter θ by choosing a distribution (the prior) $q(\theta)$ over Θ.
- 2) Choose a parametric model $\mathcal{H}$ to define the likelihood $p(X|\theta) \in \mathcal{H}$.
- 3) Update the distribution $q(\theta)$ to find the posterior $p(\theta|X) = \frac{p(X|\theta)q(\theta)}{p(X)} = \frac{p(X|\theta)q(\theta)}{\int q(\theta)p(X|\theta)d\theta}$.
- 4) Estimate and event of interest ϕ - $p(\phi|X) = \int p(\theta|X)p(\phi|\theta)d\theta$.

Note that the integral in step 3) and 4) is often intractable. This means that the likelihood and prior need to be chosen carefully for this approach to work. It is useful to introduce **Conjugate Priors**:
Let $\mathcal{H} = \{p(x;\theta) : \theta \in \Theta\}$ and $\mathcal{Q} = \{q(\,\cdot\,;\omega) : \omega \in \Omega\}$ be two parametric families of distributions. $\mathcal{Q}$ is a conjugate prior for $\mathcal{H}$ if the posterior induced by any two $p \in \mathcal{H}$ and $q \in \mathcal{Q}$ is also in $\mathcal{Q}$.
Conjugate priors are very attractive for Bayesian Inference because we get tractable posteriors.
Note that the Bayesian approach might not fulfill the desiderata.

1.3.4 Statistical Learning

Statistical learning is yet another approach to density estimation. It provides mathematically tractable solutions with low bias and low variance... for a proper model.
In statistical learning, we forget about distributions. First, we must define a model (not a probability distribution) which is a set of functions $\mathcal{H}$ for the task. For example, this could be $\mathcal{H} \subseteq \{f|f : \mathbb{R}^d \to [0,1]\}$

for a classification task.
Second, we must define a loss function $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ which takes the ground truth and the prediction and returns a loss score. The strategy is then to choose the function $f \in \mathcal{H}$ which minimizes the expected (approximated by the empirical) risk:

$$\min_{f \in \mathcal{H}} \mathbb{E}_{X,Y} \left[\mathcal{L}(Y, f(X)) \right] \approx \min_{f \in \mathcal{H}} \frac{1}{n} \sum_{i \leq n} \mathcal{L}(y_i, f(x_i))$$

In statistical learning, we are given the freedom to choose the model and the loss function. However, with that freedom comes danger - choosing a good model is difficult. In general, statistical learning does not fulfill the desiderata, it strongly depends on the model and loss function.

When choosing the model, it is important to make sure that the functions are mathematically tractable (such as e.g. $\mathcal{H} \subseteq L_2$). When choosing the loss function, there are many options used commonly in the literature (see slide 52, Lecture 3). One common example is the cross entropy $\mathcal{L} = -\log p_{f(x)(y)}$. When $f(x) \in [0, 1]$, f induces a Bernoulli distribution and we find the empirical loss $\frac{1}{n} \sum_{i \leq n} \left(-y_i \log(\sigma(w^T x)) - (1 - y_i) \log(1 - \sigma(w^T x)) \right)$.

When dealing with a range of variables (features), the model selection becomes the appropriate selection of a subset of "meaningful" variables. Furthermore, a good loss function must be selected. There are a number of different approaches for variable (model) selection, including the Akaike information criterion, cross validation or the **Bayesian Information Criterion** (BIC) (for loss function derivation). When imposing a Gaussian prior on the vector of weights (coefficients). It can then be shown that the log likelihood of your data becomes:

$$\log p(X, Y) \approx \text{const} + \log(P(X, Y|w*) - \frac{|S|}{2} \log(n),$$

where $S \subseteq \{1, \ldots, d\}$ is the set of variables used and $\log(P(X, Y|w*)$ is the maximized likelihood function. The BIC is then given by:

$$\text{BIC} = -\log p(X, Y|w*) + \frac{|S|}{2} \log(n).$$

We want to select a model with a high BIC (in Lecture, low BIC in Wikipedia???). One way to do this is by forward selection, where we train models with more and more variables and find an optimal cutoff.

1.4 Regression and the Bias-Variance Trade-off

We talk about regression problems and continue to think about optimal estimators and the bias-variance tradeoff. Remember to look at Steins Paradox...

1.4.1 The Regression Problem

The regression problem objective is to find, given $(X, Y) = \{(X_1, Y_1), \ldots, (X_n, Y_n)\}$, the (noisy) functional relationship $Y(X) = f(X) + \epsilon(X)$. The optimal solution of the regression problem is the minimum of the expected conditional risk

$$f^*(X) = \arg\min_f \mathbb{E}_{Y|X} \left[(Y - f(X))^2 \right] = \mathbb{E} \left[Y|X = x \right].$$

For a derivation, see lecture 4 Notes Thursday. If we assume a Gaussian noise distribution $\epsilon \sim \mathcal{N}\left(\epsilon|Y - f^*(X), \sigma^2\right)$, we find

$$f^*(X) = \mathbb{E}_{Y|X} \left[\left(\frac{Y - f(X)}{\sigma} \right)^2 \right].$$

Note that since we have limited observations, we always make implicit smoothness assumptions, i.e. we always bet on a simple relationship. In general, the statistical learning approach and the MLE often leads to the same solution.

Recap - Linear Regression
We choose as our statistical model the linear functions $f : \mathbb{R}^d \to \mathbb{R}$ s.t., given vector of inputs $X^T = \{X_1, \ldots, X_d\}$:

$$Y = \beta_0 + \sum_{j \leq d} X_j \beta,$$

where β_0 is referred to as the bias in machine learning or the intercept in statistics. Equivalently, we can assume a constant coordinate X_0 s.t. $X \in \mathbb{R}^{d+1}$, which makes the model simply $Y = X^T\beta$ (often also $Y = w^T X$).
To solve the linear regression model, we introduce the **Residual Sum of Squares (RSS)** estimator:
We seek to minimize, given the data $D = \{(x_i, y_i) | i = 1, \ldots, n\} \subset \mathbb{R}^{d+1} \times \mathbb{R}$, the sum:

$$RSS(\beta) = \sum_{i=1}^{n} (y_i - x_i^T \beta)^2 = (Y - X\beta)^T (Y - X\beta).$$

The minimum condition is $0 \doteq \nabla_\beta RSS(\beta) = X^T(Y - X\beta)$, giving the solution for non-singular $X^T X$: $\hat{\beta} = (X^T X)^{-1} X^T Y$. We can make predictions directly with $y = x^T \hat{\beta} = x^T (X^T X)^{-1} X^T Y$.
If we go back to the statistical Gaussian noise assumption, we find that the predictions should follow the normal distribution $\beta \sim \mathcal{N}\left(\beta | \hat{\beta}, (X^T X)^{-1} \sigma^2\right)$.

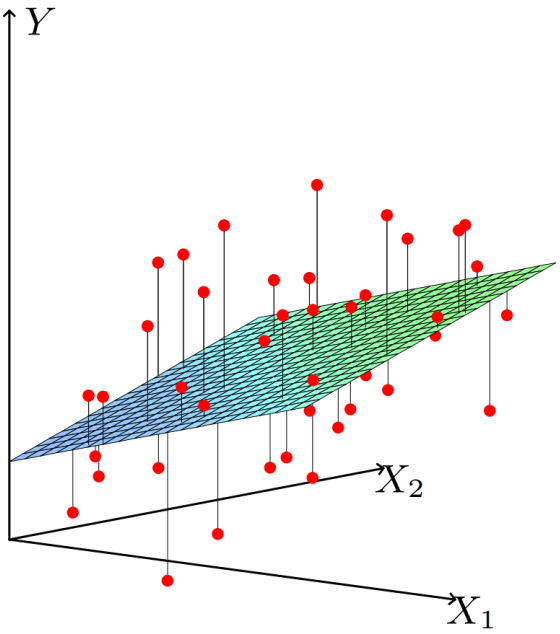


Figure 4: Linear Least Squares Fitting - We seek the linear function of X that minimizes the sum of squared residuals from Y.

Optimality of the Least Squares Estimate
We showed that the least squares estimate of the parameter β has the smallest variance of all linear unbiased estimates!
Consider the problem of estimating the value of of a linear combination $\theta = a^T\beta$, e.g. the $f(a)$ at a new location $a \in \mathbb{R}^{d+1}$. Look at slides 7 and 8 of Lecture 4 and repeat the calculations to show that the least squares estimate $\hat{\theta} = a^T \hat{\beta} = a^T (X^T X)^{-1} X^T Y$ is unbiased (i.e. $\mathbb{E}\left[a^T \hat{\beta}\right] = a^T \beta$ and that its variance is:

$$\mathbb{V}\left[a^T \hat{\beta}\right] = \sigma^2 a^T (X^T X)^{-1} a.$$

Furthermore, any alternative estimator $\tilde{\theta} = c^T Y = a^T \hat{\beta} + a^T DY$ is also unbiased if $a^T DX = 0$.
Gauss Markov Theorem
For any linear estimate $\tilde{\theta} = c^T Y$ that is unbiased for $a^T \beta$ will have larger variance than $\hat{\beta}$:

$$\mathbb{V}\left[a^T \hat{\beta}\right] \leq \mathbb{V}\left[c^T Y\right].$$

Proof.
Let $c^T Y = a^T((X^T X)^{-1} X^T + D)Y$ be any unbiased estimate of the function $f(a) = a^T \beta$. It follows that $a^T DX = 0$ (see slides).

$$\begin{aligned}\mathbb{V}\left[c^T Y\right] &= \mathbb{E}\left[(c^T Y)^2\right] - \mathbb{E}\left[c^T Y\right]^2 = c^T(\mathbb{E}\left[[]\, YY^T\right] - \mathbb{E}\left[Y\right]\mathbb{E}\left[Y\right]^T)c = \sigma^2 c^T c \\ &= \cdots \text{(see slides, do yourself)} \\ &= \mathbb{V}\left[a^T \hat{\beta}\right] + \sigma^2 a^T DD^T a \geq \mathbb{V}\left[a^T \hat{\beta}\right].\end{aligned}$$

The question becomes, is this now the best we can do at all?
We found that $\hat{f}(x) = x^T(X^T X)^{-1} X^T Y$ is the best unbiased estimate we can find. However, over-fitting can be a problem. But now we must remember the bias-variance tradeoff.

1.4.2 The Bias-Variance Tradeoff

Remember that the mean squared error of our prediction can be split into different kinds of errors:

$$\text{MSE} = \text{bias}^2 + \text{variance} + \text{noise variance}.$$

More technically, for a dataset $D = \{(x_i, y_i) | i = 1, \ldots, n\} \subset \mathbb{R}^{d+1} \times \mathbb{R}$ we can see:

$$\mathbb{E}_D\left[\mathbb{E}_{Y|X=x}\left[(\hat{f}(x) - Y)^2\right]\right] = \mathbb{E}_D\left[\hat{f}(x) - \mathbb{E}\left[Y|X=x\right]\right]^2 + \mathbb{E}_D\left[\left(\hat{f}(x) - \mathbb{E}_D\left[\hat{f}(x)\right]\right)^2\right] + \mathbb{E}\left[(Y - \mathbb{E}\left[Y|X=x\right])^2\right].$$

In general, we are facing two main issues:

- We only have limited data to construct an estimate.
- We don't know the hypothesis (model) class $\mathcal{C}$ - complexity is unknown.

If we choose a very complex hypothesis class $\mathcal{C}$, we might quickly run into overfitting. However, using a $\mathcal{C}$ that is too simple will not allow us to captures functional relationship - we under-fit. Usually, minimizing both the bias and variance is impossible. The optimal tradeoff between bias and variance is achieved when we avoid both underfitting (large bias) and overfitting (large variance). For small datasets and a large $\mathcal{C}$ we usually have large variance and small bias. For large datasets and a small $\mathcal{C}$ we usually have small variance and large bias.
Outlook: Ensemble methods seem to avoid the bias/variance tradeoff since they lower variance while keeping the bias fixed. Note: The Rao-Cramer inequality defines a lower bound for variance reduction by ensemble averaging (no free lunch).

Regularization
There are some tricks we can employ to try to find a good tradeoff:

- Regularization - adding a complexity term to the loss function $\arg\min_\theta \sum_D \mathcal{L}(f(x_i, \theta), y_i) + R(\theta)$, which is often equivalent to choosing priors and using maximum a posteriori (MAP) estimators in a Bayesian Framework.

- Model selection based on generalization error estimate (e.g. by cross-validation).
- Ensembles of classifiers (later...).

We will now focus on regularization of linear regression. There are two very common approaches - Ridge and LASSO (Least absolute shrinkage and selection operator) are outlined in the figure below.

Ridge Regression

Cost function: $RSS(\beta;\lambda) = (\mathbf{y}-\mathbf{X}\beta)^{\mathsf{T}}(\mathbf{y}-\mathbf{X}\beta)+\lambda\beta^{\mathsf{T}}\beta.$

Bayesian view: $Y|(X,\beta)\sim\mathcal{N}\left(x^{\mathsf{T}}\beta,\sigma^2\mathbf{I}\right)$, prior on β: $\beta\sim\mathcal{N}(0,\sigma^2/\lambda\mathbb{I})$.

Solution: $\hat{\beta}^{\text{ridge}} = (\mathbf{X}^{\mathsf{T}}\mathbf{X}+\lambda\mathbf{I})^{-1}\mathbf{X}^{\mathsf{T}}\mathbf{y}$

Tikhonov regularization $R(\beta)=\lambda\beta^{\mathsf{T}}\beta$ is also called weight decay in Neural Networks Literature.

LASSO

Cost function: $RSS(\beta;\lambda) = (\mathbf{y}-\mathbf{X}\beta)^{\mathsf{T}}(\mathbf{y}-\mathbf{X}\beta)+\lambda\|\beta\|_1.$

Bayesian view: $Y|(X,\beta)\sim\mathcal{N}\left(x^{\mathsf{T}}\beta,\sigma^2\mathbf{I}\right)$, prior on β_i: Laplace: $p(\beta_i)=\frac{\lambda}{4\sigma^2}\exp(-|\beta|\frac{\lambda}{2\sigma^2})$.

Solution: By efficient optimization techniques (e.g. LARS). Note: $\|\beta\|_1=\sum_{j=0}^{d}|\beta_j|$ is not differentiable.

Figure 5: Equivalence of common regularization methods and Bayesian MAP estimation. For model selection, the complexity parameters λ or s are chosen by estimates of the generalization error, e.g. cross-validation.

If we look at the ridge regression estimate $X\hat{\beta}_{ridge} = X(X^TX+\lambda\mathcal{I})^1X^TY$ and take the singular value decomposition $X=UDV^T$, we find:

$$X\hat{\beta}_{ridge} = UD(D^2+\lambda\mathcal{I})^1U^TY = \sum_{j=1}^{d} u_j\frac{d_j^2}{d_j^2+\lambda}u_j^Ty,$$

where d_j are the singular values. Note that the shrinkage factor is small for small singular values and approaches 1 for large ones - built in model selection.
Equivalently, LASSO can be defined as the least squares loss function subject to the constraint $\sum_{j=1}^{d}|\beta_j|<s$ for some shrinkage factor s.
LASSO estimates are known to be sparse with few coefficients non-vanishing. This is because the least squares error (LSE) surface often hits the corners of the constraint surface (see fig 3.12 of Hastie et al.). This is illustrated in the figure below, along with the generalized version of ridge regression.

Generalized Ridge Regression

$$\hat{\beta} = \arg\min_{\beta}\left\{\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{d}x_{i,j}\beta_j\right)^2+\lambda\sum_{j=1}^{d}|\beta_j|^q\right\}.$$

This cost function models the shrinkage of the coefficients!

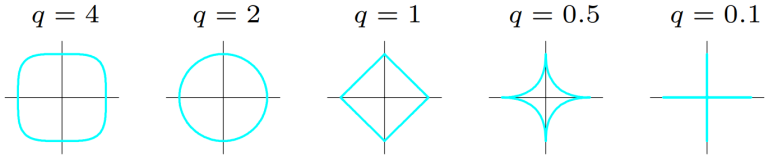


Figure 6: General Ridge Regression Illustration - the least squares error (LSE) surface often hits the corners of the constraint surface for $q=1$, which leads to sparse solution. However, other choices will not give nice sparse kernels. For $q<1$, cost functions may be non-convex.

Non Linear Regression
To do non linear regression with the same methods, we can transform X non linearly $\phi(X)$ and then do linear regression in the new feature space. E.g. use transformations $h_m(X) : \mathbb{R}^d \to \mathbb{R}$, $1 \leq m \leq M$, then the model

$$f(X) = \sum_{m=1}^{M} \beta_m h_m(X)$$

will be linear in β and but nonlinear in X. In the lecture, examples such as smoothing wavelet and cubic splines transformations are presented.

2 Gaussian Processes

2.1 Kernel Methods - A Summary

3 Support Vector Machines (SVMs)

3.1 Hard Margin and Soft Margin SVMs

3.2 Structured Support Vector Machines

4 Ensemble Methods

5 Deep Learning

6 Non-parametric Bayesian Methods

7 PAC Learning