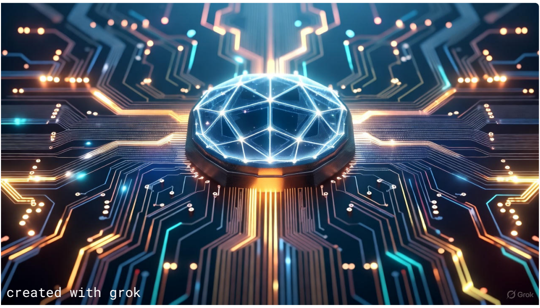
Advanced Machine Learning

Silvan Stadelmann - 3. November 2025 - v0.0.1

github.com/silvasta/summary-aml



Contents

Representation

1 Learning objectives

Estimation of Dependences Based on Empirical Data

What is the learning problem?

$$y = f_\theta(x) + \eta \quad \text{with} \quad \nu \sim \mathsf{P}(\eta|0, \sigma^2)$$

2 Expected risk

- Conditional expected risk

- Total expected risk

3 Empirical risk

- Test and Train Data

Test data cannot be used before the final estimator has been selected!

Training error $\hat{R}\hat{(}f_n, \mathcal{Z}^{\text{train}})$ for Empirical Risk Minimizer (ERM) $\hat{f}_n$

4 Empirical test error and expected risk

Distinguish

5 Comparing algorithm performance on test data

6 Data

6.1 Feature space

- Measurement space $\mathcal{X}$

+ numerical $\mathcal{X} \subset \mathbb{R}^d$

+ boolean $\mathcal{X} = \mathbb{B}$

+ categorial $\mathcal{X} = \{1, ..., k\}$

Features are derived quantities or indirect observations which often significantly compress the information content of measurements.

Remark 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.

Taxonomy of Data

6.2 Example of Data

- monadic data

- dyadic data

- pairwise data

- polyadic data

7 Mathematical Spaces

- Topological spaces

- Metric space

- Euclidean vector spaces

- Probability Spaces

Regression

8 Linear Regression

- Statistical model

$$Y = X^\top \beta. \quad Y \in \mathbb{R}. X.\beta \in \mathbb{R}^{d+1}$$

- Residual Sum of Squares (RSS)

$$RSS(\beta) \;=\; (\mathbf{y} - \mathbf{X}\beta)^\top (\mathbf{y} - \mathbf{X}\beta)$$
$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}$$

9 Gauss Markov Theorem

10 Bias/Variance Dilemma

- Tradeoff, split Error

- Identify error components

11 Bayesian Maximum A Posteriori (MAP) estimates

11.1 Ridge Regression

- Cost function

- Bayesian view

- Solution

Tikhonov regularization

11.2 LASSO

- Cost function

- Bayesian view

- Solution

11.3 Ridge vs. LASSO Estimation

12 Remarks on Shrinkage Methods

- Generalized Ridge Regression

Idea behind shrinkage When white noise is added to the data then all Fourier coefficients are increased by a constant on average. ▯ Shrink all coefficients by the estimated noise amount to derive a robust predictor.

13 Model averaging is common practice

- Previous: Gaussian process motivated by Bayesian linear regression.

- Seldom: take MAP estimator in Bayesian setting.

- Bayesian approach: average models with different parameters (weighted according to prior).

- Cross validation: Take average over models trained on different folds.

- Winners of most Machine Learning competitions (e.g. on Kaggle): ensembles (weighted averages of models).

-

14 Combining Regressors - Bias

TODO: formula

-

15 Combining Regressors - Variance

TODO: formula

16 Ensemble Learning

The idea of classifier ensembles Boosting is an approach to machine learning based on the idea of creating a highly accurate prediction rule by combining many relatively weak and inaccurate rules.

- Computational advantage

- Statistical advantage

17 Induction Principles for Classifier Selection

I) Empirical Risk Minimization (ERM) Principle

II) Bayesian inference by model averaging

18 Motivation for Ensemble Methods

- Train several sufficiently diverse predictors

- Bagging

- Arcing

- Boosting

19 Weak Learners Used for Bagging or Boosting

Combining Classifiers

Bagging Classifiers

Classifier selection: First compare, then bag!

Bagging: The Mechanism

Decision Trees

Random Forests

The Idea of Boosting

AdaBoost

Data Reweighting

Boosted Classifier

Comparison of ensemble methods

20 Loss functions for classification

21 Learning Objectives

- To motivate, understand, and design Gaussian processes.

- To be able to analytically derive procedures for making predictions with Gaussian processes.

- To analytically compute conditionals, marginals, and posteriors of Gaussians.

- To formulate and understand kernels.

- To be able to use kernel engineering to design new kernels.

- To be able to make a formal connection between Gaussian processes and Bayesian linear regression.

22 Gaussian Processes

22.1 Bayesian linear regression

multiple linear regression model

$$Y = X^T\beta + \epsilon \quad \text{Gaussian Noise } \epsilon \sim \mathcal{N}(\epsilon|0, \sigma^2)$$

$$p(Y|X, \beta, \sigma) = \mathcal{N}(Y|X^T\beta, \sigma^2) \propto e^{-\frac{1}{2\sigma^2}(Y - X^T\beta)^2}$$

Bayesian linear regression extends multiple linear regression by defining a prior over the regression coefficients, for example (ridge regression)

- Model inversion

22.2 Moments of Bayesian linear regression

Setting

Expected Value

Covariance

23 Gaussian processes

Moments of joint Gaussian:

$Y \sim \mathcal{N}(Y|0, k_{i,j} + \sigma^2 \text{if } i = j)$

with $k_{i,j}$ kernel function

Gaussian Processes as "kernelized linear regression"

- Kernel functions specify the similarity between any two data points.

23.1 Recall

Kernel properties:

- Symmetry

- Positive semi-definit

23.2 Gram matrix

Must be positive semi-definit

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

23.3 Examples of kernel functions

Linear kernel: k(x, x0) = xT x0

Polynomial kernel: k(x, x0) = (xT x0 + 1)p , for p ∈ N

Gaussian (RBF) kernel: k(x, x0) = exp −kx − x0 k22 /h2

Sigmoid (tanh) kernel: k(x, x0) = tanh κxT x0 − b

Different kernels have different **invariance properties**!

For example, invariance to **rotation** or **translation.**

23.4 Kernel engineering by composition

Addition: Multiplication: Scaling: Composition:

23.5 Prediction by Gaussian processes

Predictive density $p(y_{n+1}|x_{n+1}, X, y)$

Reminder: Conditional Gaussian Distributions

23.6 Prediction by Gaussian processes

23.7 Kernel validation

Goal: Validate hyperparameters of kernels by random splits D

24 Controller Optimization for Robust Control

Machine Learning in Control Systems

Machine learning techniques are becoming more and more important for enabling computers to control complex and stochastic systems and predict the outcomes of such systems.

24.1 Gaussian processes for Control

A Fundamental problem when designing controllers for dynamic systems is the estimation of the controller parameters. Besides pure statistical performance, robustness arises as an important design issue.

The classical approach selects a model of the system to design an initial controller; parameters are then tuned manually to achieve best performance.

An alternative approach uses methods from machine learning to optimize statistical performance, e.g., Bayesian optimization.

Safety-critical system failures may happen because these methods evaluate different controller parameters.

24.2 Safe optimization

Overcome safety-critical system failures by using a specialized optimization algorithm for automatic controller parameter tuning. This algorithm models the underlying performance measure as a GP and only explores new controller parameters whose performance lies above a safe performance threshold with high probability.

Support Vector Machines

TODO: check short scrips

Neural Networks

Transformer

Diffusion

Lecture 8, 03.11.25

Diffusion (to produce images)

- goal find $p^\star$

- Start with gaussian distribution

- generative process, from noise to image

- corruptive process, from image sample to noise

Training

corruptive process

x(i) image

$e_t(i) \rightarrow x_t(i)$

up to the middle / noise

reverse process

from noise to image

Diffusion process for MNIST

24.3 sgrok

Diffusion Models for Image Generation

Diffusion models are a class of generative models aimed at producing images by learning to approximate the true data distribution $p^\star$. The core idea involves two complementary processes: a **corruptive process** that gradually adds noise to an image, transforming it into pure Gaussian noise, and a **reverse process** that denoises the noise back to a realistic image.

Corruptive Process (Forward Diffusion) Starting from a clean image sample $x_0(i)$ (e.g., a digit from the MNIST dataset), the corruptive process applies a sequence of noise additions over T timesteps:

$$x_t(i) = \sqrt{\alpha_t} x_{t-1}(i) + \sqrt{1-\alpha_t}\epsilon_t(i), \quad \epsilon_t(i) \sim \mathcal{N}(0, I),$$

where α_t controls the noise level, and $\epsilon_t(i)$ is Gaussian noise. This continues until $x_T(i)$ approximates a standard Gaussian distribution (pure noise) at timestep T, effectively destroying the image structure.

Training During training, the model learns to reverse this corruption. A neural network (e.g., a U-Net) is trained to predict the noise $\epsilon_t(i)$ added at each timestep t, given the noisy image $x_t(i)$. The objective is to minimize the difference between predicted and actual noise, often using a mean-squared error loss. This allows the model to capture the data distribution by simulating the forward process on training samples (like MNIST digits) and optimizing the reverse denoising steps.

Generative Process (Reverse Diffusion) For generation, start with a sample from a Gaussian distribution (pure noise) $x_T \sim \mathcal{N}(0, I)$. The trained model iteratively denoises it via the reverse process:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \hat{\epsilon}_t(x_t) \right) + \sigma_t z, \quad z \sim \mathcal{N}(0, I$$

where $\hat{\epsilon}_t$ is the model's noise prediction, and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. Over T steps, this transforms noise into a high-fidelity image, such as a synthetic MNIST digit.

This framework, as in Denoising Diffusion Probabilistic Models (DDPM), excels in tasks like image synthesis due to its stable training and high-quality outputs. For implementation, engineers can use libraries like Diffusers (Hugging Face) with pre-trained models for MNIST or extend to advanced variants like Stable Diffusion for conditional generation.

24.4 Encoder

Autoencoder

24.4.1 Clip

- encode image and text

- match them somehow

- produce like diagonal matrix

24.5 Pipeline

Goal: text to image

Idea:

- produce text and image embeddings that are semantically relatable

- text -> produce embedding -> generative model -> image

24.5.1 details

- cross attention

24.6 Unet Architecture

prompt -> encoder -> embeding-text

image -> encoder ->et -> corrupted image xt xt -> unet (only image input), now?

add text with cross-attention inbetween UNet steps

24.6.1 Multi-headed X Attention Mechanism

embeding-text (Batch,MaxTokens,DimEmbeddings)

embeding-image (Batch,Chanels, Heigth, Width)

wK,wV to Et, gives K(B,F,M,Dk),V(B,F,M,Dv)

wQ to Ei, gives Q

- calculate similarities

use K,Q to create P(B,F,M,H,W)

- P = similarity of token m in text and pixel (h,w) of image

create S, softmax of P (along dimension M)

Create A(B,F,Dv,H,W) from S and V

use wO(Do,H,W) to create output Ao(B,Do,H,W)

25 Graph

Toxic, non toxic? structure matters -> graph

G(V,E)

V=1,...,N

$Ec = V \times V$

Exercises

26 Problem 1 - Regression

- Linear Regression

- Ridge Regression

- Noisy Regression

E1.2.c - An Engineer's rule of thumb is to choose K as min $\sqrt{n}, 10$

- Overfitting

- Cross Validation

- Generative vs. Discriminative Modeling