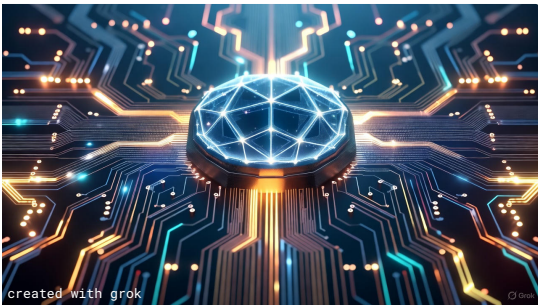


# Advanced Machine Learning

Silvan Stadelmann - 18. November 2025 - v0.0.1

[github.com/silvasta/summary-aml](https://github.com/silvasta/summary-aml)



## Contents

|  |          |
|--|----------|
| <b>Representation</b>                                | <b>3</b> |
| 1 Learning objectives                                | 3        |
| 2 Expected risk                                      | 3        |
| 3 Empirical risk                                     | 3        |
| 4 Empirical test error and expected risk             | 3        |
| 5 Comparing algorithm performance on test data       | 3        |
| 6 Data   | 3        |
| 6.1 Feature space . . . . .                          | 3        |
| 6.2 Example of Data . . . . .                        | 3        |
| 7 Mathematical Spaces                                | 4        |
| <b>Regression</b>                                    | <b>4</b> |
| 8 Linear Regression                                  | 4        |
| 9 Gauss Markov Theorem                               | 4        |
| 10 Bias/Variance Dilemma                             | 4        |
| 11 Bayesian Maximum A Posteriori (MAP) estimates     | 4        |
| 11.1 Ridge Regression . . . . .                      | 4        |
| 11.2 LASSO . . . . .                                 | 4        |
| 11.3 Ridge vs. LASSO Estimation . . . . .            | 4        |
| 12 Remarks on Shrinkage Methods                      | 4        |
| 13 Learning Objectives                               | 4        |
| 14 Gaussian Processes                                | 4        |
| 14.1 Bayesian linear regression . . . . .            | 4        |
| 14.2 Moments of Bayesian linear regression . . . . . | 5        |
| 15 Gaussian processes                                | 5        |
| 15.1 Recall . . . . .                                | 5        |
| 15.2 Gram matrix . . . . .                           | 5        |
| 15.3 Examples of kernel functions . . . . .          | 5        |

|           |  |          |
|-----------|--|----------|
| 15.4      | Kernel engineering by composition . . . . .          | 5        |
| 15.5      | Prediction by Gaussian processes . . . . .           | 5        |
| 15.6      | Prediction by Gaussian processes . . . . .           | 5        |
| 15.7      | Kernel validation . . . . .                          | 5        |
| <b>16</b> | <b>Controller Optimization for Robust Control</b>    | <b>5</b> |
| 16.1      | Gaussian processes for Control . . . . .             | 5        |
| 16.2      | Safe optimization . . . . .                          | 6        |
| <b>17</b> | <b>Model averaging is common practice</b>            | <b>6</b> |
| <b>18</b> | <b>Combining Regressors - Bias</b>                   | <b>6</b> |
| <b>19</b> | <b>Combining Regressors - Variance</b>               | <b>6</b> |
| <b>20</b> | <b>Ensemble Learning</b>                             | <b>6</b> |
| <b>21</b> | <b>Induction Principles for Classifier Selection</b> | <b>6</b> |
| <b>22</b> | <b>Motivation for Ensemble Methods</b>               | <b>6</b> |
| <b>23</b> | <b>Weak Learners Used for Bagging or Boosting</b>    | <b>6</b> |
| <b>24</b> | <b>Loss functions for classification</b>             | <b>7</b> |
|           | <b>Support Vector Machines</b>                       | <b>7</b> |
| <b>25</b> | <b>SVM and convex optimization</b>                   | <b>7</b> |
| 25.1      | Learning objectives . . . . .                        | 7        |
| 25.2      | Instability of the perceptron . . . . .              | 7        |
| 25.3      | Maximum-margin criterion . . . . .                   | 7        |
| 25.4      | Lagrange Multipliers . . . . .                       | 7        |
| 25.5      | Convex optimization . . . . .                        | 7        |
| 25.5.1    | The dual . . . . .                                   | 7        |
| 25.5.2    | Weak duality and strong duality . . . . .            | 7        |
| 25.5.3    | Slater's condition . . . . .                         | 7        |
| 25.5.4    | Consequences of strong duality . . . . .             | 7        |
| 25.5.5    | Convex optimization via the dual . . . . .           | 7        |
| 25.6      | SVMs . . . . .                                       | 7        |
| 25.7      | Soft-margin SVMs . . . . .                           | 7        |
| <b>26</b> | <b>Kernelization</b>                                 | <b>7</b> |
| 26.1      | Kernels . . . . .                                    | 7        |
| <b>27</b> | <b>SVM summary</b>                                   | <b>7</b> |
| <b>28</b> | <b>Structural SVMs</b>                               | <b>8</b> |
|           | <b>Neural Networks</b>                               | <b>8</b> |
|           | <b>Transformer</b>                                   | <b>8</b> |
| <b>29</b> | <b>Tokenization and embeddings</b>                   | <b>8</b> |
| <b>30</b> | <b>Self-attention</b>                                | <b>8</b> |
| 30.1      | Multi-headed attention . . . . .                     | 9        |
| 30.2      | Cross-attention . . . . .                            | 9        |
| 30.3      | Masked self-attention . . . . .                      | 9        |
| <b>31</b> | <b>Positional encodings</b>                          | <b>9</b> |
| <b>32</b> | <b>Broadcasting</b>                                  | <b>9</b> |
| <b>33</b> | <b>Residual Blocks</b>                               | <b>9</b> |

|   |           |
|---|-----------|
| <b>34 Transformer Architecture</b>            | <b>9</b>  |
| <b>Diffusion</b>                              | <b>9</b>  |
| 34.1 sgrok . . . . .                          | 9         |
| 34.2 Encoder . . . . .                        | 10        |
| 34.2.1 Clip . . . . .                         | 10        |
| 34.3 Pipeline . . . . .                       | 10        |
| 34.3.1 details . . . . .                      | 10        |
| 34.4 Unet Architecture . . . . .              | 10        |
| 34.4.1 Multi-headed X Attention Mechanism . . | 11        |
| <b>35 Graph</b>                               | <b>11</b> |
| <b>Exercises</b>                              | <b>11</b> |
| <b>36 Problem 1 - Regression</b>              | <b>11</b> |

**Representation**

**1 Learning objectives**

Estimation of Dependences Based on Empirical Data

What is the learning problem?

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

**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)

**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  $\mathcal{X} = \{1, \dots, k\}$

**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

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

- Residual Sum of Squares (RSS)

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

## 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.  $\Rightarrow$  Shrink all coefficients by the estimated noise amount to derive a robust predictor.

## 13 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.

## 14 Gaussian Processes

### 14.1 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

## 14.2 Moments of Bayesian linear regression

Setting

Expected Value

Covariance

## 15 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.

#### 15.1 Recall

Kernel properties:

- Symmetry

- Positive semi-definit

#### 15.2 Gram matrix

Must be positive semi-definit

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

#### 15.3 Examples of kernel functions

Linear kernel:  $k(x, x_0) = x^T x_0$

Polynomial kernel:  $k(x, x_0) = (x^T x_0 + 1)^p$ , for  $p \in \mathbb{N}$

Gaussian (RBF) kernel:  $k(x, x_0) = \exp -\frac{\|x - x_0\|^2}{2\sigma^2}$

Sigmoid (tanh) kernel:  $k(x, x_0) = \tanh \kappa x^T x_0 - b$

Different kernels have different **invariance properties!**

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

#### 15.4 Kernel engineering by composition

Addition: Multiplication: Scaling: Composition:

#### 15.5 Prediction by Gaussian processes

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

Reminder: Conditional Gaussian Distributions

#### 15.6 Prediction by Gaussian processes

#### 15.7 Kernel validation

Goal: Validate hyperparameters of kernels by random splits D

## 16 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.

#### 16.1 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.

## 16.2 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.

## 17 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).
- 

## 18 Combining Regressors - Bias

TODO: formula

-

## 19 Combining Regressors - Variance

TODO: formula

## 20 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

## 21 Induction Principles for Classifier Selection

- I) Empirical Risk Minimization (ERM) Principle
- II) Bayesian inference by model averaging

## 22 Motivation for Ensemble Methods

- Train several sufficiently diverse predictors
- Bagging
- Arcing
- Boosting

## 23 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

## **24 Loss functions for classification**

### **Support Vector Machines**

## **25 SVM and convex optimization**

### **25.1 Learning objectives**

- Lagrange multipliers

- Basics of convex optimization and duality

### **25.2 Instability of the perceptron**

### **25.3 Maximum-margin criterion**

### **25.4 Lagrange Multipliers**

### **25.5 Convex optimization**

#### **25.5.1 The dual**

#### **25.5.2 Weak duality and strong duality**

#### **25.5.3 Slater's condition**

#### **25.5.4 Consequences of strong duality**

- The primal optimal solution is the minimizer of the Lagrangian:

- Complementary slackness:

#### **25.5.5 Convex optimization via the dual**

### **25.6 SVMs**

- Formula separation band

- Projection

- Normalization trick

- Formalization of primal

Why do we compute the dual?

- SVM calculation can become intractable in high dimensions or when working with transformations (more on this later).

- The dual does not depend on the dimensionality of the dataset in the primal!

Complementary slackness

What about the intercept?

What if the dataset is not linearly separable?

- Soft-margin SVMs

- Kernels

### **25.7 Soft-margin SVMs**

- Formulation

- The role of C

dual ...

## **26 Kernelization**

Transformations and kernels

Polynomial transformations

SVMs and kernels

### **26.1 Kernels**

no need for  $\phi$  only  $\phi^T(x)\phi(x')$  for some cases of  $x$  and  $x'$

## **27 SVM summary**

Support Vector Machine (SVM): Idea

- margin

- kernel

Nonlinear Transformation in Kernel Space

SVM Lagrangian for functional margin formulation

Non-separable case: Soft Margin SVM

Learning the Soft Margin SVM

## 28 Structural SVMs

From margins to score functions

Extensions to the SVM

Multiclass SVMs (linear discriminant function)

Structural SVMs

..more examples

## Neural Networks

- Activation functions

- Forward computation

- Gradient descent

- Chain rule

A **computation DAG** is a directed acyclic graph where vertices are annotated with variables and edges are annotated with differentiable vector fields.

- generalization chain rule

- Backward computation

## Transformer

### 29 Tokenization and embeddings

- Topic classification

Strategy for NLP

We create for each word in the sentence an embedding vector. Then we aggregate these embeddings to create an embedding for the sentence. We then apply a function on the embedding to map the sentence to a distribution of topics.

- Tokenization

The first step is to transform the sentence into a sequence of tokens.

A popular method is the WordPiece tokenization. It produces a decomposition of a training corpus of text into  $S$  tokens, where  $S$  is predefined in advance.

1. Set  $T$  = set of characters occurring in text

2. While size of  $T > S$  do:

1. Find the most common pair  $a, b$  of tokens in  $T$  occurring in the text.

2. Remove  $a$  and  $b$  from  $T$  and put instead a new token  $ab$  in  $T$ .

### 30 Self-attention

- Sentiment classification for movies

- Attention

- Relational matrices

These maps can also be used to infer meanings from words

Every word finds its meaning through other words in the sentence!

Attention as information retrieval

An attention map for location of objects

- Query

- Keys



- Values

Abstraction of an attention mechanism

### 30.1 Multi-headed attention

### 30.2 Cross-attention

Consider the problem of translating from English to German:

- A different attention mechanism is needed

Abstraction of a cross-attention mechanism

### 30.3 Masked self-attention

Consider the task of generating text for answering questions.

## 31 Positional encodings

- Consider the sentences the dog chased the cat and the cat chased the dog.

- They are permutations of each other, so the attention maps will also just be permutations of each other.

- Note that attention mechanisms do not pay attention to the position of a token in the sentence.

- Binary positional encoding

- Positional encodings

Why adding and not concatenating?

## 32 Broadcasting

- Broadcasting in PyTorch

- Examples of broadcasting

Broadcasting formalized

Implementing the masked-attention mechanism

## 33 Residual Blocks

- The degradation problem

- Residual blocks

## 34 Transformer Architecture

BERT: Bidirectional Encoder Representations from Transformers

## Diffusion

Lecture 8, 03.11.25

Diffusion (to produce images)

- goal find  $p^*$

- 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

### 34.1 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  $p^*$ . 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  $\alpha_t$  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.

## 34.2 Encoder

Autoencoder

### 34.2.1 Clip

- encode image and text
- match them somehow
- produce like diagonal matrix

## 34.3 Pipeline

Goal: text to image

Idea:

- produce text and image embeddings that are semantically relatable
- text -> produce embedding -> generative model -> image

### 34.3.1 details

- cross attention

## 34.4 Unet Architecture

prompt -> encoder -> embedding-text

image -> encoder ->  $\epsilon_t$  -> corrupted image  $x_t$

$x_t$  -> unet (only image input), now?

add text with cross-attention inbetween UNet steps

**34.4.1 Multi-headed X Attention Mechanism**

embedding-text (Batch,MaxTokens,DimEmbeddings)

embedding-image (Batch,Chanel, Height, Width)

$w_K, w_V$  to  $E_t$ , gives  $K(B,F,M,D_k), V(B,F,M,D_v)$

$w_Q$  to  $E_i$ , 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  $A(B,F,D_v,H,W)$  from  $S$  and  $V$

use  $w_O(D_o,H,W)$  to create output  $A_o(B,D_o,H,W)$

**35 Graph**

Toxic, non toxic? structure matters -> graph

$G(V,E)$

$V=1,...,N$

$$Ec = V \times V$$

**Exercises**

**36 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