Algorithmic Foundations of Datascience

A summary of the Lecture

Felix F.

Summer term 2019

Abstract

This summary is based on the lecture 'Algorithmic Foundations of Datascience' held by Prof. Martin Grohe in the summer term of 2019 at RWTH Aachen University.

Since this summary is made by student(s), there are definitely mistakes in the script. Should you find one feel free to contact me or push a correction to the repository.

Contents

1 Machine Learning Basics

1.1 Definitions

All the following definitions are specific to 'Machine Learning' and might change throughout the summary.

Data is viewed as a collection of data items (or dataset). A **data item** is represented by a so-called feature vector. The **feature vector** basically a list of features and attributes in form of a vector

1.1.1 Domains

The **Domain** $\mathbb{D}$ is a set containing the range of all possible values for a single feature. The **Instance Space** $\mathbb{X}$ is the Cartesian product of the domains of all features in the feature vector. The number of features is the dimension of the instance space (or the number of entries in the feature vector).

◀ $\mathbb{D}$

◀ $\mathbb{X}$

> *Example:* The domain space can be something like $\{true, false\}$ or $\mathbb{N}$, aso. The instance space of the domain spaces is then (informally) $\mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_l$

Assuming n is the number of data items, and l the number of features the whole dataset can be viewed as an $(n \times l)$-matrix over the instance space.

1.2 Forms of Learning

Unsupervised learning tries to detect patterns in data with, hence the name, no feedback, supplied the Learning algorithm. This technique is mostly used for clustering.

Reinforcement Learning tries to find actions that maximise reward. The feedback to the learning algorithm is provided in form of a reward (or punishment)

In **Supervised learning** one tries to approximate a function by training on input-output pairs. **Classification** has a finite-valued function and tries to predict values for future inputs. **Regression** has a numerical function and tries to predict the expected values for future inputs. Supervised learning can be differentiated into two different set-ups. First, **Batch Learning**, in which all examples are supplied at once and the learner has to come up with a hypothesis. Where as in **Online Learning**, examples are given 'on-the-fly' once at a time and the initial hypothesis gets improved over time.

Semi-supervised Learning is similar to supervised learning, but takes only a few and possibly faulty examples and tries to make the best of the possibly small and faulty dataset.

1.3 Hypothesis Space

$\mathcal{H}$ ▶ In Supervised learning, the unknown target function h is choosen from the **hypothesis space** $\mathcal{H}$, hence $h \in \mathcal{H}$. The Hypothesis space contains all linear functions, all polynomials or all polynomials of a pre-scribed degree, and all functions that can be described by a decision tree.

A learning problem is **Realisable** if the target function h is in the Hypothesis Space.

> *Example:* Assuming a target function $h(x) := a \cdot \sin(x) + bx + c$. Then, if $\mathcal{H}$ only contains polynomials, the learning problem is **not realizable**. But, assuming $\mathcal{H}$ contains linear functions or polynomials in x and $\sin(x)$, the learning problem **is realisable**.

1.4 Validation

The goal of the learning algorithm is to produce a hypothesis that generalizes well, that is, approximates the target function well ion the all data points (and not only those in the training set). To evaluate how well a hypothesis generalizes we can evaluate it against some test set. A **test set** is generates by splitting the examples into a test set and training set.

The empirical observation that simpler hypothesis tend to generalise better is picked up by **Occam's Razor** which states, that the simplest hypothesis which is consistent with the data should be choosen.

Overfitting is the phenomenon, which occurs when the leaner tries to match the training examples as exactly as possible which leads to complex hypotheses that generalize badly. Thus, to avoid overfitting, it is often better to choose a simple hypothesis even if it doesn't match the data exactly.

1.5 Nearest Neighbour Learning

The underlying idea of this simple learning algorithm is to predict the value of a function at point x by looking at known values of points close to the given one and assume the value of x is similar.

1.5.1 Metric

d ▶ A **metric** d on $\mathbb{X}$ is a function $d : \mathbb{X}^2 \longrightarrow \mathbb{R}$ such that for all $x, y, z \in \mathbb{X}$

Nonnegativity	**Symmetry**	**Triangle Inequality**
$d(x,y) \geq 0$ and $d(x,y) = 0 \iff x = y$;	$d(x,y) = d(y,x)$	$d(x,z) \leq d(x,y) + d(y,z)$

1.5.2 Metric Spaces

Suppose $\mathbb{X}$ is the instance space, and d be some metric on $\mathbb{X}$. Then we have a **Metric Space** $(\mathbb{X}, d)$.

◀ $(\mathbb{X}, d)$

1.5.2.1 Common distance metrics

The distance between points x and y is denoted as $d(x, y)$.

◀ $d(x, y)$

Euclidean distance with $\mathbb{X} = \mathbb{R}^\ell$:

$$d(x, y) = \sqrt{\sum_{i=1}^{\ell} (x_i - y_i)^2} \tag{1.5.1}$$

Manhattan distance with $\mathbb{X} = \mathbb{R}^\ell$:

$$d(x, y) = \sum_{i=1}^{\ell} |x_i - y_i| \tag{1.5.2}$$

Hamming distance with $\mathbb{X} = \mathbb{D}_1 \times \mathbb{D}_2 \times \ldots \times \mathbb{D}_\ell$

$$d(x, y) = |\{\, i \mid x_i \neq y_i \}| \tag{1.5.3}$$

1.5.3 Nearest Neighbour Classifier

Problem description: 'Learn' a function f, that associates a class $f(x) \in \mathbb{Y}$ with every $x \in \mathbb{X}$. Basically: $f : \mathbb{X} \to \mathbb{Y}, x \mapsto y$. Function f is provided with a dataset to learn from: $(x_1, y_1), \ldots, (x_m, y_m)$ with m entries where each entry corresponds to $(x_i, f(x_i))$. x_i is called data item and y_i is called class x_i.

k-Nearest Neighbour Classifier uses $x \in \mathbb{X}$ as input.

1. Find the k nearest neighbours $x_{i_1}, \ldots, x_{i_k}$ of x in $\{x_1, \ldots, x_m\}$

2. Take a 'majority vote' of the class that appears the most among $y_{i_1}, \ldots, y_{i_k}$

Choosing an appropriate parameter k: When choosing $k = 1$ the classifier tends to overhit, while with larger k one might start to oversimplify. Obviously, the optimal value for k is difficult to determine and depends heavily on the application.

1.5.4 Learning Decision Trees

A decision tree is a rooted tree with labelled notes and edges

- Every internal node of the tree is labelled by an (input) feature

- Every edge is labeled by a value or range of values for the feature labelling its source node.

- Every leaf is labeled with an output value

Semantics: The function value for an input vector x is the value at the leaf of the unique path in the tree whose edges are labelled by the feature values in x.

Note: Computing a smallest decision tree for a given sets of examples is NP-Complete

1.5.4.1 Greedy Algorithm for Building Decision Trees

Input: Set $\mathcal{A}$ of features, set $\mathcal{S}$ of examples
Objective: Compute decision tree t

```
if $\mathcal{S} = \emptyset$ then
    create leaf $t$ with arbitrary value
else if all samples in $\mathcal{S}$ have the same $y$-value then
    create leaf $t$ with that $y$-value
else
    choose feature $A \in \mathcal{A}$ that discriminates best between examples in $\mathcal{S}$
    create new node $t$ with feature $A$
    partition examples in S according to their $A$-value into parts $S_1, ..., S_m$
    recursively call algorithm on $\mathcal{A} \backslash \{A\}$ and the $S_i$ and attach resulting trees $t_i$ as children to $t$
end if
return $t$
```

1.6 The Perceptron

The Perceptron algorithm attempts to find a linear classifier for a Boolean classification problem.

1.6.1 Scalar product

$\langle \mathbf{x}, \mathbf{y} \rangle$ ▶ Two vectors, $\mathbf{x} = (x_1, \ldots, x_l)^T$, $\mathbf{y} = (y_1, \ldots, y_l)^T \in \mathbb{R}^\ell$

$$\langle \mathbf{x}, \mathbf{y} \rangle = \sum_{i=1}^{\ell} x_i y_i$$

1.6.2 Euclidean norm

$\|\mathbf{x}\|$ ▶ Vector $\mathbf{x} \in \mathbb{R}^\ell$

$$\|\mathbf{x}\| = \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} = \sqrt{\sum_{i=1}^{\ell} x_i^2}$$

1.6.3 Cauchy-Schwarz Inequality

$$|\langle \mathbf{x}, \mathbf{y} \rangle| \leq \|\mathbf{x}\| \cdot \|\mathbf{y}\| \tag{1.6.1}$$

1.6.4 Hyperplanes

A (affine) **Hyperplane** in $\mathbb{R}^\ell$ is an affine subspace of dimension $\ell - 1$. It can be described as the set of all solutions $\mathbf{x} \in \mathbb{R}^\ell$ to the equation $a_1 x_1 + \ldots + a_\ell x_\ell = b$ where $a_1, \ldots, a_\ell, b \in \mathbb{R}$. In short: $\langle \mathbf{a}, \mathbf{x} \rangle - b = 0$, or more formally: $P = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle = b\}$.

A Hyperplane $P = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle = b\}$ is **homogeneous** if $b = 0$, or equivalently, if $0 \in P$. That's, the Hyperplane goes through the origin.

1.6.5 Halfspaces

A **Halfspace** in $\mathbb{R}^\ell$ is the set of all points on one side of a Hyperplane. A Halfspace can be described as the set of solutions $\mathbf{x} \in \mathbb{R}^\ell$ to an inequality $\langle \mathbf{a}, \mathbf{x} \rangle - b \geq 0$ where $\mathbf{a} \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$. Or more formally: $H = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle - b \geq 0\}$

A Halfspace is a **homogeneous Halfspace** as follows: $H = \{x \in \mathbb{R}^\ell \mid \langle \mathbf{a}, \mathbf{x} \rangle \geq 0\}$. So basically a halfspace with $b = 0$.

1.6.6 Linear Classification

The **Boolean Classification** is a classification problem with the target space $\mathbb{D} = \{+1, -1\}$. The goal is to learn an unknown target function $f : \mathbb{R}^\ell \to \{+1, -1\}$.

The input for the learning function is a sequence of data items (so called training sequence):

$$S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in \mathbb{R}^\ell \times \{+1, -1\} \tag{1.6.2}$$

The hypothesis space ($\mathcal{H}$) consists of **linear separators**, which are functions $h : \mathbb{R}^\ell \to \mathbb{R}$ in the form of:

$$h(\mathbf{x}) = \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b) = \begin{cases} +1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b > 0 \\ 0 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b = 0 \\ -1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b < 0 \end{cases} \tag{1.6.3}$$

with $w \in \mathbb{R}^\ell$, the weight vector and $b \in \mathbb{R}^\ell$ the bias.

Because of the mismatching target function range of $\{+1, 0, -1\}$ and the target space of the Boolean Classification $\{+1, -1\}$ the linear separator function is redefined as follows:

$$f(\mathbf{x}) = \left\{ \begin{array}{ll} +1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b \geq 0 \\ -1 & \text{if } \langle \mathbf{w}, \mathbf{x} \rangle - b < 0 \end{array} \right. \tag{1.6.4}$$

A hypothesis h is a **consistent hypothesis** if for a training sequence $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ the following applies: $h(\mathbf{x}_i) = y_i$ for all $i \in [m]$.

1.6.6.1 Non-bias Linear Classification

A **homogeneous linear separator** is a function, such that $\mathbf{x} \mapsto \langle \mathbf{w}, \mathbf{x} \rangle$. The removal of the bias (b) implies, the linear separator goes through the origin of the coordinate system.

Theorem Properties of non-bias Linear Classification (1.6.5)

Let $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ with $\mathbf{x}_i = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell$ and $y_i \in \{-1, 1\}$ and $\mathbf{w} = (w_1, \ldots, w_\ell) \in \mathbb{R}^\ell$ and $b \in \mathbb{R}$. Then the following are equivalent.

1. $\mathbf{x} \mapsto \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b)$ is a linear separator consistent with S
2. $\mathbf{x}' \mapsto \operatorname{sgn}(\langle \mathbf{w}', \mathbf{x}' \rangle)$ is a homogeneous linear separator consistent with $S' = ((\mathbf{x}'_1, y_1), \ldots, (\mathbf{x}'_m, y_m))$, where $\mathbf{x}'_i := (x_{i_1}, \ldots, x_{i_\ell}, 1)$ and $\mathbf{w}' = (w_1, \ldots, w_\ell, -b)$

1.6.6.2 Normalized Linear Classification

Intuition: as the name suggests informally, we simply divide all $\mathbf{x}$ in the training sequence by the length of the largest $\mathbf{x}$ vector. Therefore, the largest vector then has a length of at most 1.

Suppose we have a training sequence $S = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m))$ with $\mathbf{x}_i = (x_{i_1}, \ldots, x_{i_\ell}) \in \mathbb{R}^\ell$ and $y_i \in \{-1, 1\}$. We can apply the following transformation to all datapoints $\mathbf{x}_i$

$$\mathbf{x}_i \mapsto \hat{\mathbf{x}}_\mathbf{i} := \frac{\mathbf{x}'_i}{\max_{1 \le j \le m} \|\mathbf{x}'_j\|} \tag{1.6.6}$$

Resulting in the **normalized training sequence** $\hat{S} = ((\hat{\mathbf{x}}_1, y_1), \ldots, (\hat{\mathbf{x}}_m, y_m))$. Additionally $\hat{S}$ has following properties:

1. $\hat{S}$ has a homogeneous linear separator if and only if S has a linear separator
2. $0 < \|\hat{\mathbf{x}}_\mathrm{i}\| \le 1$ for all $i \in [m]$

1.6.6.3 Linear Seperator Margin

The **margin of a linear seperator** is the orthogonal distance of the nearest point to the linear separator function h. Formally: the margin of h with respect to the target sequence S is

$$\min_{(x,y) \in S} |\langle \mathbf{w}, \mathbf{x} \rangle| \tag{1.6.7}$$

Theorem (1.6.8)

Let S be a normalized sequence of examples such that there is a homogeneous linear separator consistent with S of margin γ. Then the perceptron algorithm applied to S finds a linear separator after at most $\frac{1}{\gamma^2}$updates of w.

Proof See Page 1.46 in the slidedeck for Kap1

1.6.6.4 The Perceptron Algorithm

Input: Normalized training sequence S
Objective: Compute weight vector $\mathbf{w}$ such that the hypothesis $x \mapsto \operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle)$ is consistent with S

```
$w \leftarrow 0$
repeat
    for all $(x, y) \in S$ do
        if $\operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle) \neq y$ then
            $w \leftarrow w + yx$
        end if
    end for
until $\operatorname{sgn}(\langle \mathbf{x}, \mathbf{w} \rangle) = y$ for all $(x, y) \in S$
```

1.6.6.5 Linear Separation for Nonlinear Problems

Suppose the instance space is $\mathbb{R}^2$ and the hypothesis space consists of the quadratic separators if the form:

$$(x_1, x_2) \mapsto \operatorname{sgn}(a_1 x_1^2 + a_2 x_1 x_2 + a_3 x_2^2 + a_4 x_1 + a_5 x_2 + a_6) \tag{1.6.9}$$

To learn the parameters $a_1, \ldots, a_6$, we define a transformation $\tau : \mathbb{R}^2 \to \mathbb{R}^5$ by

$$\tau(x_1, x_2) = (x_1^2, x_1 x_2, x_2^2, x_1, x_2) \tag{1.6.10}$$

Then quadratic separators for a training sequence

$$S = ((\mathrm{x}_1, y_1), \ldots, (\mathrm{x}_m, y_m)) \in \mathbb{R}^2 \times \{+1, -1\} \tag{1.6.11}$$

correspond to linear separator for the transformed training sequence

$$\tau(S) := ((\tau(\mathrm{x}_1), y_1), \ldots, (\tau(\mathrm{x}_m), y_m)) \in \mathbb{R}^5 \times \{+1, -1\} \tag{1.6.12}$$

1.7 Neural Networks

TODO: Add this section. Maybe later, but not that important, since the section on the slides is very broad

1.8 k-Means Clustering

This is the only example of unsupervised learning covered by the lecture.
Task: Put a collection of data points into k clusters whereby the intra-cluster distances are to be minimized and the inter-cluster distances are to be maximized. The number of clusters k is fixed in advance.
Again, we take $\mathbb{R}^\ell$ as the instance space and use the Euclidean metric for distance measurements.

1.8.1 Centroid Based Clustering

In centroid based clustering each cluster in represented by its centroid z (a.k.a. "centre of a mass" or "mean"), i.e. it is represented by the point that minimizes the squared distance to all points in the cluster. Each data point x will be associated with the cluster whose centroid is closest to x.

Instance: Points $x_1, ..., x_n \in \mathbb{R}^\ell$, number $k \in \mathbb{N}$
Problem: Find points $z^1, ..., z^k \in \mathbb{R}^\ell$ and a partition $C^1, ..., C^k$ of $\{x_1, ..., x_n\}$ that minimizes

$$\sum_{j=1}^{k} \sum_{x \in C^j} \|x - z^j\|^2 \tag{1.8.1}$$

THEOREM (1.8.2)

(1) The Centroid Clustering problem is NP-hard, even if either the dimension ℓ or the cluster number k are fixed to be 2.
(2) If both k and ℓ are fixed, the problem can be solved in polynomial time.

PROOF was omitted ∎

1.8.2 k-Means Clustering Algorithm

Input: $x_1, \dots, x_n \in \mathbb{R}^\ell$, $k \in \mathbb{N}$
Objective: c.f. 1.8.1

```
Choose initial "centroids" $z^1, ..., z^k$ (for example randomly)
repeat
    $C^j \leftarrow \emptyset \ \forall j \in [k]$
    for $i \leftarrow 1$ to $n$ do
        $j \leftarrow \arg\min_j \|x_i - z^j\|$          (if there is a tie, choose the smallest $j$)
        add $x_i$ to $C^j$
    end for
    $z^j \leftarrow \frac{\sum_{x \in C^j} x}{|C^j|} \forall j \in [k]$
until $C^1, ..., C^k$ no longer change
return $C^1, ..., C^k$
```

THEOREM (1.8.3)

The k-Means algorithm always halts in a finite number of steps.

PROOF See Page 1.56 in the slidedeck for Kap1 ∎

Remarks on the algorithm:

- The number of steps the k-Means algortihm takes until it halts may be exponential in the number n of input points - however in practice the algorithm usually converges quickly

- The k-Means algorithm does not necessarily conpute an optimal solution for the Centroid Clustering problem

- The clustering returned by the k-Means algorithm may depend on the choice of the inituial centroids $z^1, \dots, z^k$

2 Information and Compression

2.1 Probability Theory

Recall that a probability space consists of a set Ω, the **sample space**, a set $\mathcal{E} \subseteq 2^{\Omega}$, the **event space**, and a **probability distribution** $\mathcal{P}$ that associates an probability $\mathcal{P}(E)$ with each event $E \in \mathcal{E}$. ◀ Ω ◀ $\mathcal{P}$

For an event $A(\omega)$ depending on an element $\omega \in \Omega$ of the sample space we sometimes write $\Pr_{w \sim \mathcal{P}}(A(\omega))$ to denote the probability $\mathcal{P}(A(\omega))$

A **random variable** on a probability space $(\Omega, \mathcal{P})$ is a mapping $X : \Omega \to \mathbb{R}$. By $X < x$, for $x \in \mathbb{R}$, we done the event $\{\omega \in \Omega \mid X(\omega) \leq x\}$. We usually denote the probability of this event by $\Pr(X \leq x)$, without reference to the probability distribution $\mathcal{P}$.

The **cumulative distribution function** F_X of X is defined by $F_X : \mathbb{R} \to [0, 1]$. ◀ F_X

For random variables X with a finite or countably infinite range, we can define the **probability mass function** $f_X : \mathbb{R} \to [0, 1]$ by $f_X(x) := \Pr(X = x)$. Then have

$$F_X(x) = \sum_{y \leq x} f_x(y)$$

This gives us a **probability distribution** $\mathcal{P}_x$ the probability distribution of X ◀ $\mathcal{P}_x$

$$\mathcal{P}_X(A) := \sum_{x \in A} \Pr(X = x)$$

For continuous random variables (on an arbitrary probability space) there exists a **propability density function**

$$F_X(x) = \int_{-\infty}^{x} f_x(y)dy$$

2.1.1 Random Vectors and Joint Distributions

Let (X, Y) be a pair of random variables with a finite range over the same probability space $(\Omega, \mathcal{P})$. The **joint distribution of** X, Y is the probability distribution $\mathcal{P}_{(X,Y)}$ on $\mathbb{R}^2$ defined by

$$\begin{aligned}\mathcal{P}_{(X,Y)}(\{(x, y)\}) : &= \Pr(X = x, Y = y) \\ &= \mathcal{P}(\{\omega \in \Omega | X(\omega) = x \text{ and } Y(\omega) = y\})\end{aligned}$$

Conversely, from the joint distribution we can retrieve the **marginal distribution** of X

$$\Pr(X = x) = \sum_{y} \Pr((X, Y) = (x, y))$$

A tuple $\mathbf{X} = (X_1, \ldots, X_k)$ of random variables is sometimes called a **random vector**

2.1.2 Independence

Let (Ω, P) be a probability space and $k \geq 1$. Events $E_1, \ldots, E_k \subseteq$ are **independent** if $\mathcal{P}(E_1, \ldots, E_k) = \mathcal{P}(E_1) \cdots \mathcal{P}(E_k)$. Here $E_1, \ldots, E_k$ denotes the event $E_1 \cap \ldots \cap E_k$.

Random variables $X_1, \ldots, X_k : \Omega \to \mathbb{R}$ are independent if for all $A_1, \ldots, A_k \subseteq \mathbb{R}$ the events $X_i \in A_i$ are independent, that is,

$$\Pr(X_1 \in A_1, \ldots, X_k \in A_k) = \Pr(X_1 \in A_1) \cdots \Pr(X_k \in A_k)$$

We also say that the random vector $\mathbf{X} = (X_1, \ldots, X_k)$ is independent.

2.1.3 Expectation and Variance

Let X be a random variable. The expectation of x is $\mathrm{E}(X) = \sum_{x \in \mathbb{R}} x \cdot \Pr(X = x)$

The variance of X is $\mathrm{Var}(X) = E((X - E(X))^2)$

1. For $\alpha, \beta \in \mathbb{R}, \mathrm{E}(\alpha X + \beta Y) = \alpha \mathrm{E}(X) + \beta \mathrm{E}(Y)$ linearity of expectation
2. X and Y are independent then $\mathrm{E}(X \cdot Y) = \mathrm{E}(X)\mathrm{E}(Y)$
3. $\mathrm{Var}(X) = \mathrm{E}\left(X^2\right) - \mathrm{E}(X)^2$
4. X and Y are independent then $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$

2.1.4 Markov's and Chebyshev's Inequalities

Markov's Inequality. Let X be a nonnegative random variable. Then for all $a \geq 0$,

$$\Pr(X \geq a) \leq \frac{E(X)}{a} \tag{2.1.1}$$

Chebyshev's Inequality. Let X be a random variable. Then for all $b > 0$,

$$\Pr(|X - \mathrm{E}(X)| \geq b) \leq \frac{\mathrm{Var}(X)}{b^2} \tag{2.1.2}$$

Proof of Markov's Inequality See Slide 2.8-a ▮

Proof of Chebyshev's Inequalit See Slide 2.8-a ▮

2.1.5 k-Wise Independence

A sequence $X_1, \dots, X_n$ of random variables is **k-wise independent**, for some $k \geq n$, if for all $i_1 < \dots < i_k \in [n]$ the sequence $X_{i_1}, \dots, X_{i_k}$ is independent. Instead 2-wise independent, we usually say **pairwise independent**

Let $X_1, \dots, X_n$ be a pairwise independent sequence of random variables and $X := \sum_{i=1}^{n} X_i$. Then

$$Var(X) = \sum_{i=1}^{n} \mathrm{Var}(X_i) \tag{2.1.3}$$

PROOF See Slide 2.9-a

2.1.6 The Weak Law of Large Numbers

TODO

2.2 Digression: Concentration Inequalities

2.2.1 Sum of Random Variables

The sum of a rancom process is follows:

$$X = \sum_{i=1}^{n} X_i$$

Obviously, the expected value is $\mu := E(X)$. We show, that with high probability, X is close to its expected value, that is:

$$\Pr(|X - \mu| \geq \text{something big}) \leq \text{something small}$$

These inequalities are called **concentration inequalities** or **tail bounds**.

2.2.2 Chernoff Bounds

Let $X_1, \dots, X_n$ be a sequence of independent $\{0,1\}$-valued random variables. Let $X := \sum_{i=1}^{n} X_i$ and $\mu := E(X)$. Then for $0 \leq c \leq 1$:

$$\Pr(X \geq (1+c)\mu) \leq e^{-\frac{\mu c^2}{3}} \text{ and } \Pr(X \leq (1-c)\mu) \leq e^{-\frac{\mu c^2}{2}}$$

2.2.3 Hoeffding Bounds

Let $X_1, \dots, X_n$ be a sequence of independent identically distributed $\{0,1\}$-valued random variables. Let $X := \sum_{i=1}^{n} X_i$ and $\mu := E(X)$. Then for $0 \leq d \leq 1$:

$$\Pr(X \geq \mu + dn) \leq e^{-2nd^2} \text{ and } \Pr(X \leq \mu - dn) \leq e^{-2nd^2}$$

2.2.4 General Random variable concentration

THEOREM General random variable Concentration (2.2.1)

Let $X_1, \dots, X_n$ be a sequence of independent random variable swit $E(X_i) = 0$ and $\mathrm{Var}(X_i) \leq \sigma^2$, and $X := \sum_{i=1}^{n} X_i$. Let $a \in \mathbb{R}$ such that $0 \leq a \leq \sqrt{2}n\sigma^2$ and suppose that $E(X_i^k) \leq \sigma^2 k!$ for $3 \leq k \leq \lceil \frac{a^2}{4n\sigma^2} \rceil$. Then $\Pr(|X| \geq a) \leq 3e^{-\frac{a^2}{12n\sigma^2}}$.

2.3 Entropy

$I(A)$ ▶ The **information content** $I(A)$ of a event A in a finite probability space $(\Omega, \mathcal{P})$ is:

$$I(A) = \log(\frac{1}{\mathcal{P}(A)})$$

PROOF Proof and Lemma for the corresponding base of the logarithm on Slide $2.20 - (a - c)$ ■

The **entropy** of a probability distribution $\mathcal{P}$ on a finite sample space Ω is defined to be

$$H(\mathcal{P}) := \sum_{w \in \Omega} \mathcal{P}(\{w\}) * \log \frac{1}{\mathcal{P}(\{w\})}$$

where we take $0 \log(1/0)$ to be 0 (or only sum over all w with $\mathcal{P}(w) > 0$).

The **entropy** of a random variable X with finite range is defined to be

$$H(X) := \sum_{x \in rg(X)} \Pr(X = x) * \log \frac{1}{\Pr(X = x)}$$

2.3.1 Lower and Upper Bounds

Let $\mathcal{P}$ be a probability distribution on a sample space Ω of size $|\Omega| = n$ Then

$$0 \leq H(\mathcal{P}) \leq \log n$$

Furthermore,

$$H(\mathcal{P}) = 0 \iff \exists \omega \in \Omega : \mathcal{P}(\{\omega\}) = 1$$

and

$$H(\mathcal{P}) = \log(n) \iff \mathcal{P} \text{ is the uniform distribution, that is, } \mathcal{P}(\{\omega\}) = 1/n \text{ for all } \omega \in \Omega$$

Let $f : D \to \mathbb{R}$, where $D \subseteq \mathbb{R}$. Then f is convex if for all $\lambda \in [0, 1]$ and all $x, y \in D$ it holds that $\lambda x + (1 - \lambda)y \in D$ and

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y)$$

If $f(\lambda x + (1 - \lambda)y < \lambda f(x) + (1 - \lambda)f(y)$ for all $\lambda \in (0, 1)$ and all distinct $x, y \in D$ then f is strictly convex.

2.3.1.1 Jensen's Inequality

Let X be a random variable, and let $f : D \to \mathbb{R}$ be a convex functionwith $\mathrm{rg}(X) \subseteq D$. Then $E(X) \in D$ and

$$f(\mathrm{E}(X)) \leq \mathrm{E}(f(X))$$

f is strictly convex, then equality holds if and only if X

PROOF Proof on page 2.24-(a-c)

2.3.2 Log Sum Inequality

For all $i \in [n]$, let $p_i \in \mathbb{R}_{\geq 0}$,$q_i \in \mathbb{R}_{>0}$, and let $p := \sum_i p_i$ and $q := \sum_i q_i$. Then

$$\sum_{i=1}^{n} p_i \log(\frac{p_i}{q_i}) \geq p \log(\frac{p}{q})$$

Moreover, equality holds if and only if $p_i/q_i = p_j/q_j$ for all i, j

PROOF Proof on page 2.25-(a-c)

2.3.3 Information Gain & Entropy

TODO - Slides are not very understandable – come up with something better

2.4 Compression

Out intuition behind the entropy of a probability distribution $\mathcal{P}$ is that it measures the average 'Information' of a sample from $\mathcal{P}$, that is, of an elementary event $\{w\}$.

- Information is the average number of bit we need to transfer (or store) many conservative samples from a distribution, using the best possible coding scheme for this specific distribution.
- Suppose we have a string where the symbols are independently sampled from our distribution. Then the information content of the distribution should measure how well we can compress the string.

Strings are compressed over a finite **source alphabet** Σ. We assume $\Sigma \geq 2$. Compressed strings are encoded in the **target alphabet** $\{0, 1\}$. A **compression scheme** over Σ is a pair $\Gamma = (\mathrm{com}_\Gamma, \mathrm{dec}_\Gamma)$ consisting of ◀ Σ

- **compression mapping** $\mathrm{com}_\Gamma : \Sigma \to \{0, 1\}^*$ ◀ com_Γ
- **decompression mapping** $\mathrm{dec}_\Gamma : \{0, 1\}^* \to \Sigma^*$. ◀ dec_Γ

The goal is to have a compression scheme, that is ideally **lossless**, that is $dec(com(\vec{x})) = x$ for all $x \in \Sigma^*$, and guarantees a good compression.

ρ_Γ ▶

The **compression rate** of a scheme Γ $\vec{X}$ is $\frac{|com(\vec{x})|}{|\vec{x}|}$. Intuitively, the compression rate if Γ is the maximum of the compression rates Γ of all strings in Γ^*. As this maximum does not necessarily exist, we define a compression rate for strings of length n separately for each n: the compression rate of Γ is the function $\rho_\Gamma : \mathbb{N} \to \mathbb{R}$ defined by

$$\rho_\Gamma(n) := \max_{x \in \Sigma^n} \frac{|\operatorname{com}(x)|}{|x|}$$