Summary of

Algorithmic Foundations of Data Science

Prof. Dr. Martin Grohe

Summer Semester 2023

RWTH Aachen University

1 Understanding Machine Learning Algorithms

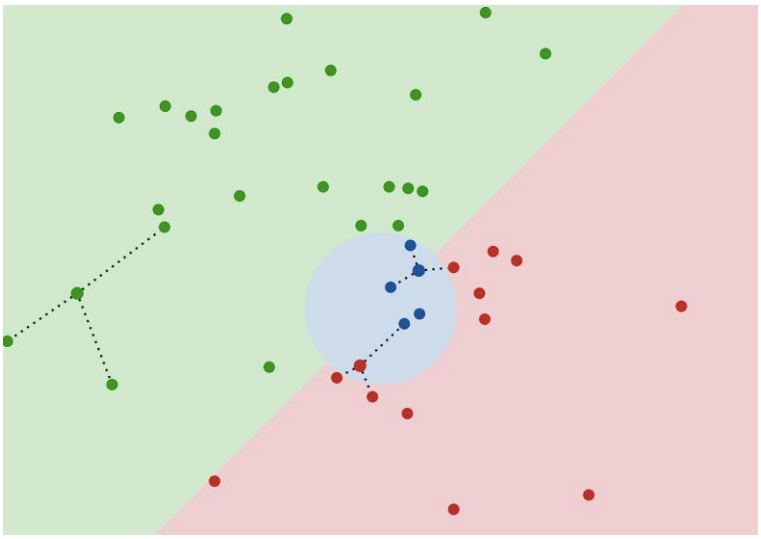
1.1 Nearest Neighbour Algorithm

Classifier k-Nearest Neighbour

▷ Based on labelled examples $(x_1, y_1), \ldots (x_m, y_m)$
▷ k is some parameter

Input: $x \in \mathbb{X}$

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", that is, return the class y that appears most often among $y_{i_k}, \ldots, y_{i_k}$ (break ties arbitrarily).



- Search of nearest neighbour in n data points requires time $O(n)$
- This can be improved using a k-d-tree and binary search

⇒ Data structure choice is important

1.2 Decision Trees

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

1. if $S = \emptyset$ then
2. create leaf t with arbitrary value
   ▷ e.g., majority value for parent node
3. else if all examples in S have the same y-value then
4. create leaf t with that y-value
5. else
6. choose feature $A \in \mathcal{A}$ that discriminates best between examples in S
7. create new node t with feature A
8. partition examples in S according to their A-value into parts $S_1, \ldots, S_m$
9. recursively call algorithm on $\mathcal{A} \setminus \{A\}$ and the S_i and attach resulting trees t_i as children to t
10. return t

There are different heuristics for choosing the "best" feature, most often information gain is used.

Theorem 1.3

(1) *Every Boolean function $f : \{0,1\}^n \to \{0,1\}$ is represented by a decision tree height n with $2^{n+1} - 1$ nodes.*

(2) *There is a Boolean function $f : \{0,1\}^n \to \{0,1\}$ such that every decision tree representing f has height at least n and at least $2^{n+1} - 1$ nodes.*

Theorem 1.4

Computing a smallest decision tree for a given set of examples is NP*-hard.*

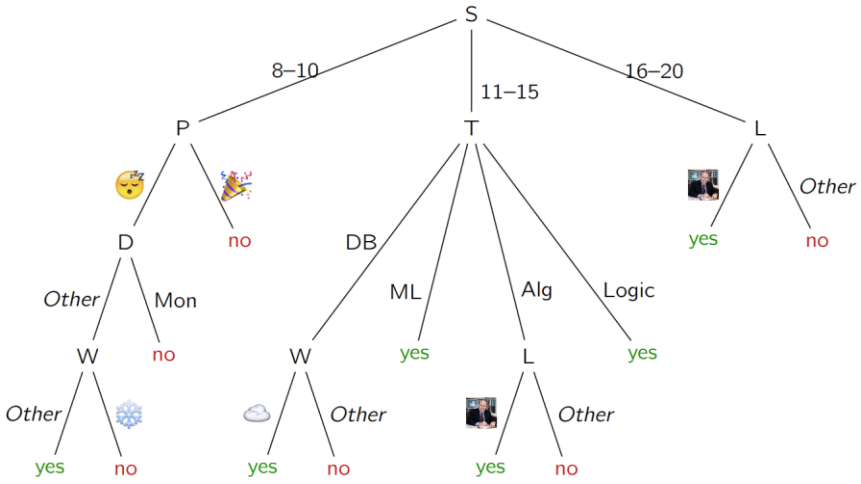
Lemma 1.5

A binary tree has k leaves if and only if it has $2k - 1$ nodes.

Example

Feature	Values
Weather	Sunny (☀️), Cloudy (☁️), Rainy (☔), Snow (❄️)
Day of the week	Mon, Tue, Wed, Thu, Fri
Party the night before	yes (🥳), no (😴)
Topic	DB, ML, Alg, Logic
Start time	8–10, 11–15, 16-20
Lecturer	Codd (👴), Karp (🧔), Rabin (👨), Valiant (👴)

Example	W	D	P	T	S	L	Output
x_1	☀️	Mon	😴	DB	8–10	👴	y_1 = no
x_2	☁️	Wed	😴	ML	11–15	👨	y_2 = yes
x_3	☔	Thu	😴	Alg	11–15	🧔	y_3 = no
x_4	☁️	Wed	😴	DB	11–15	👴	y_4 = yes
x_5	☔	Thu	🥳	Logic	8–10	🧔👨	y_5 = no
x_6	❄️	Tue	🥳	Logic	16–20	🧔👨	y_6 = yes
x_7	❄️	Tue	😴	Alg	16–20	👨	y_7 = no
x_8	☔	Wed	😴	Logic	11–15	🧔👨	y_8 = yes
x_9	☀️	Thu	😴	DB	11–15	👴	y_9 = no
x_{10}	☁️	Thu	😴	DB	8–10	👴	y_{10} = yes
x_{11}	☁️	Wed	😴	ML	16–20	👨	y_{11} = no
x_{12}	☔	Thu	🥳	DB	16–20	👴	y_{12} = no
x_{13}	☔	Tue	😴	Alg	11–15	🧔	y_{13} = yes
x_{14}	❄️	Tue	😴	Alg	8–10	🧔	y_{14} = no
x_{15}	❄️	Mon	😴	Alg	8–10	🧔👨	y_{15} = no
x_{16}	☔	Tue	😴	Alg	8–10	🧔	y_{16} = yes

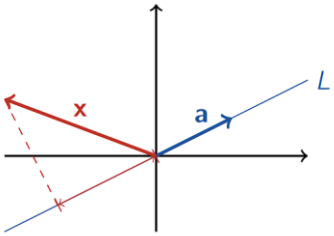


1.3 Linear Classifiers

1.3.1 Background

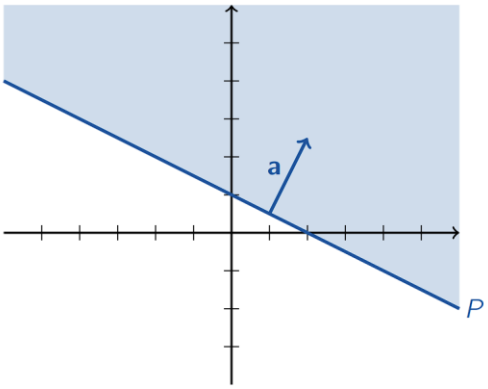
Scalar Product

- Scalar product $\langle x, y \rangle = \sum_{i=1}^{l} x_i \, y_i$
- Euclidean norm $\|x\| = \sqrt{\langle x, x \rangle}$
- Cauchy-Schwarz Inequality $|\langle x, y \rangle| \leq \|x\| * \|y\|$
- Let $a, x \in \mathbb{R}^l$ and L be the line through a. Then the length of the projection of x into L is $\frac{|\langle a,x \rangle|}{\|a\|}$



Hyperplane and Hyperspace

- A hyperplane in $\mathbb{R}^l$ is the set of all solutions to a linear equation $P = \{(x_1, \dots, x_l) \in \mathbb{R}^l \mid \langle a, x\rangle - b = 0\}$
- A halfspace in $\mathbb{R}^l$ is the set of all points on one side of a hyperplane $H = \{(x_1, \dots, x_l) \in \mathbb{R}^l \mid \langle a, x\rangle - b \geq 0\}$
- These are homogenous if $b = 0$



Hyperplane $P = \{\mathbf{x} \mid \langle \mathbf{a}, \mathbf{x}\rangle = b\}$ for $\mathbf{a} = \begin{pmatrix}1\\2\end{pmatrix}$ and $b = 1$.

Halfspace $H = \{\mathbf{x} \mid \langle \mathbf{a}, \mathbf{x}\rangle \geq b\}$

1.3.2 Linear Classification

- Goal: Learn unknown target function $f: \mathbb{R} \to \{+1, -1\}$
- Input: Training sequence $S = \big((x_1, y_1), \dots, (x_m, y_m)\big) \in \mathbb{R}^l \times \{+1, -1\}$
- Hypothesis space consist of linear separators with $w \in \mathbb{R}^l$ (called weight vector) and $b \in \mathbb{R}^l$ (called bias)

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

- A hypothesis h is consistent with a training sequence S if $h(x_i) = y_i$ for all $(x_i, y_i) \in S$
- A homogeneous linear separator is a function $x \mapsto sgn(\langle w, x\rangle)$

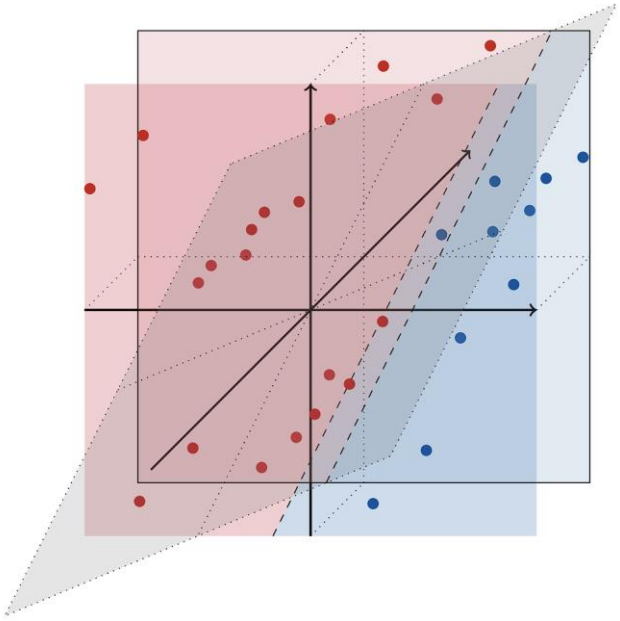
Lemma 1.6

Let $S = \big((\mathbf{x}_1, y_1), \dots, (\mathbf{x}_m, y_m)\big) \in \big(\mathbb{R}^\ell \times \{-1, 1\}\big)^m$ with $\mathbf{x}_i = (x_{i1}, \dots, x_{i\ell}) \in \mathbb{R}^\ell$ and $\mathbf{w} = (w_1, \dots, w_\ell) \in \mathbb{R}^\ell$, $b \in \mathbb{R}$. Then the following are equivalent.

(i) $\mathbf{x} \mapsto \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x}\rangle - b)$ *is a linear separator consistent with S.*

(ii) $\widehat{\mathbf{x}} \mapsto \operatorname{sgn}(\langle \widehat{\mathbf{w}}, \widehat{\mathbf{x}}\rangle)$ *is a homogeneous linear separator consistent with $\widehat{S} = \big((\widehat{\mathbf{x}}_1, y_1), \dots, (\widehat{\mathbf{x}}_m, y_m)\big)$, where $\widehat{\mathbf{x}}_i := (x_{i1}, \dots, x_{i\ell}, 1)$ and $\widehat{\mathbf{w}} = (w_1, \dots, w_\ell, -b)$.*

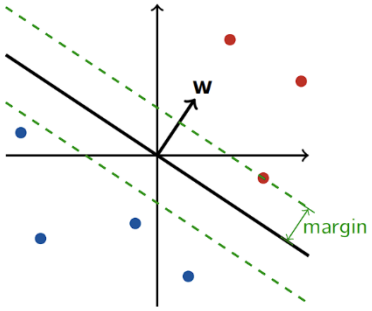
Example



Margin

Given a homogeneous linear separator h and a sequence S of examples:

The margin of h is $\min_{(x,y)\in S} \frac{|\langle w,x\rangle|}{\|w\|}$.



1.3.3 Empirical Risk Minimisation (ERM)

Instance: Training data $S = \big((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)\big) \in \big(\mathbb{R}^\ell \times \{-1, 1\}\big)^m$

Problem: Compute weight vector $\mathbf{w} \in \mathbb{R}^\ell$ such that the homogeneous linear separator $\mathbf{x} \mapsto \mathrm{sgn}(\langle \mathbf{w}, \mathbf{x}\rangle)$ minimises the training error, that is, the number of i such that $\mathrm{sgn}(\langle \mathbf{w}, \mathbf{x}_i\rangle) \neq y_i$

Observation 1.9
For all $\mathbf{w}, \mathbf{x} \in \mathbb{R}^\ell$ *and* $y \in \{-1, 1\}$,

$$\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle) = y \quad \iff \quad y \langle \mathbf{w}, \mathbf{x} \rangle > 0.$$

Lemma 1.10
The following are equivalent:

(i) *There is a* $\mathbf{w} \in \mathbb{R}^\ell$ *such that the homogeneous linear separator* $\mathbf{x} \mapsto \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$ *is consistent with* S*, that is,* $\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x}_i \rangle) = y_i$ *for all* $i \in [m]$.

(ii) *There is a* $\mathbf{w} \in \mathbb{R}^\ell$ *such that* $y_i \cdot \langle \mathbf{w}, \mathbf{x}_i \rangle > 0$ *for all* $i \in [m]$.

(iii) *There is a* $\mathbf{w} \in \mathbb{R}^\ell$ *such that* $y_i \cdot \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1$ *for all* $i \in [m]$.

(iv) *There is a* $\mathbf{w} \in \mathbb{R}^\ell$ *such that* $A_S \mathbf{w} \geq \mathbf{1}$*, where* $A_S = (a_{ij})_{i \in [m], j \in [\ell]} \in \mathbb{R}^{m \times \ell}$ *is the matrix with entries* $a_{ij} = y_i \cdot x_{ij}$ *and* $\mathbf{1} = \underbrace{(1, \ldots, 1)}_{m \text{ times}}^\top$.

1.3.4 Perceptron Algorithm

Input: Training sequence S such that there is homogeneous linear separator consistent with S.
Objective: Compute $\mathbf{w} \in \mathbb{R}^\ell$ such that $\mathbf{x} \mapsto \operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$ is consistent with S.

1. $\mathbf{w} \leftarrow \mathbf{0}$
2. repeat
3. for all $(\mathbf{x}, y) \in S$ do
4. if $\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle) \neq y$ then
5. $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$
6. until $\operatorname{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle) = y$ for all $(\mathbf{x}, y) \in S$

Theorem 1.13
Suppose that there is a homogeneous linear separator consistent with S *of margin* γ*. Let*

$$\lambda := \max_{(\mathbf{x}, y) \in S} \|\mathbf{x}\|.$$

Then the perceptron algorithm applied to S *finds a linear separator after at most* $(\lambda/\gamma)^2$ *updates of* $\mathbf{w}$.

Good	Always finds consistent separators if they exist
	Does so efficiently
Bad	Separators found are not optimal in any sense (→ SVMs)

1.3.5 Support Vector Machines

SVM

Instance $S \in \left(\mathbb{R}^\ell \times \{-1, 1\}\right)^m$
Problem Compute $\mathbf{w} \in \mathbb{R}^\ell$
minimising $\|\mathbf{w}\|^2$
subject to $y \langle \mathbf{w}, \mathbf{x} \rangle \geq 1$ for all $(\mathbf{x}, y) \in S$

This is new compared to ERM

Theorem 1.15

Suppose that there is a homogeneous linear separator consistent with S. Let $\mathbf{w}^* \in \mathbb{R}^\ell$ be an optimal solution to SVM. Then $h^* : \mathbf{x} \mapsto \text{sgn}(\langle \mathbf{w}^*, \mathbf{x} \rangle)$ is a homogeneous linear separator consistent with S.

Furthermore, for every homogeneous linear separator $h : \mathbf{x} \mapsto \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle)$ consistent with S, the margin of h w.r.t. S is smaller than or equal to the margin of h^* w.r.t. S.

1.3.6 Logistic Regression

- Methods considered so far all assume that there is a linear separator consistent with S
- In practice, this is often not given
- Logistic regression finds a separator that separates the data points reasonably well

Sigmoid Function

$$sig: \mathbb{R} \to [0,1], \qquad sig(z) := \frac{1}{1+e^{-z}}$$

Hypotheses

Logistic regression returns a function of the form

$$f: \mathbb{R}^l \to [0,1], \qquad f(x) = sig(\langle w, x \rangle - b)$$

or with $b = 0$ in the homogeneous case. It expresses the confidence that the correct label for x is 1.

$$\begin{array}{lll} \langle w,x \rangle \to \infty & f(x) \to 1 & \text{High confidence (label is } +1) \\ \langle w,x \rangle \to 0 & f(x) \to \frac{1}{2} & \text{Low confidence (can't decide)} \\ \langle w,x \rangle \to -\infty & f(x) \to 0 & \text{High confidence (label is } -1) \end{array}$$

Training

The goal is to maximize the confidence of giving the correct answer

⇒ Maximize $f(yx) = \begin{cases} f(x), & y = 1 \\ 1 - f(x), & y = -1 \end{cases} = sig(y \langle w,x \rangle)$ for all $(x,y) \in S$

⇒ Maximize $\prod_{(x,y)\in S} sig(y \langle w,x \rangle)$

⇒ Minimize $\sum_{(x,y)\in S} \ln(1 + exp(-y \langle w,x \rangle))$ computationally more efficient

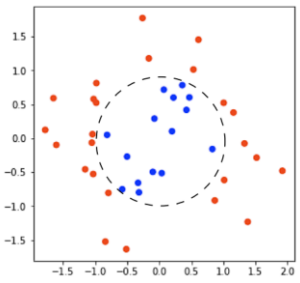
Logistic Regression

Instance $S \in \left(\mathbb{R}^\ell \times \{-1,1\}\right)^m$

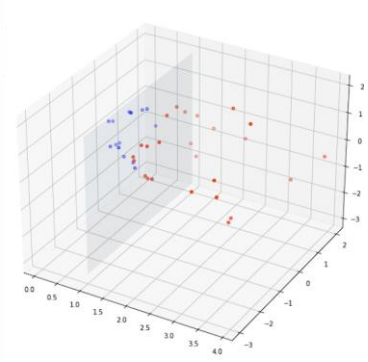
Problem Compute $\mathbf{w} \in \mathbb{R}^\ell$ minimising $\sum_{(x,y)\in S} \ln\left(1 + \exp(-y \langle \mathbf{w}, \mathbf{x} \rangle)\right)$.

1.3.7 Linear Separation for Nonlinear Problems

Mapping the data to a higher dimensional space may transform nonlinear functions to linear function in the new higher-dimensional space.



$$(x,\ y) \mapsto (x^2+y^2,\ x,\ y) \longrightarrow$$



1.3.8 K-Means Clustering

Centroid Clustering

Instance Points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^\ell$, number $k \in \mathbb{N}$

Problem Find points $\mathbf{z}^1, \ldots, \mathbf{z}^k \in \mathbb{R}^\ell$ and a partition $C^1, \ldots, C^k$ of $\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ that minimise

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

Algorithm K-MEANS

Input: $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^\ell$, $k \in \mathbb{N}$

1. Choose initial "centroids" $\mathbf{z}^1, \ldots, \mathbf{z}^k$ (for example, randomly)
2. repeat
3. $\quad C^j \leftarrow \emptyset$ for all $j \in [k]$
4. $\quad$ for $i \leftarrow 1$ to n do
5. $\quad\quad j \leftarrow \arg\min_j \|\mathbf{x}_i - \mathbf{z}^j\|$
   (If there is a tie, choose the smallest j)
6. $\quad\quad$ add $\mathbf{x}_i$ to C^j
7. $\quad \mathbf{z}^j \leftarrow \frac{\sum_{\mathbf{x} \in C^j} \mathbf{x}}{|C^j|}$ for all $j \in [k]$
8. until $C^1, \ldots, C^k$ no longer change
9. return $C^1, \ldots, C^k$

Theorem 1.18

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

Theorem 1.19

(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.*

Remarks

- Time complexity is exponential, usually converges quickly in practice though
- Does not necessarily compute an optimal solution
- Solution depends on the initial centroids

2 Information and Compression

2.1 Background from Probability Theory

- Probability space $(\Omega, \mathcal{P})$
- Sample space Ω
- Probability distribution (of X) $\mathcal{P}_X(A) = \sum_{\omega \in A} \Pr(X = \omega)$
- Event $A \subseteq \Omega$
- Random variable $X: \Omega \mapsto \mathbb{R}$
  $X = x$ refers to the event $\{\omega \in \Omega \mid X(\omega) = x\}$
  $X \in A$ refers to the event $\{\omega \in \Omega \mid X(\omega) \in A\}$
- Cumulative distribution function $F_X(x) = \Pr(X \leq x)$
- Events are independent if $\mathcal{P}(E_1, \dots, E_k) = \mathcal{P}(E_1) * \dots * \mathcal{P}(E_k)$

Meaning

The sample space Ω consists of all outcomes of the experiment. Every outcome (elementary event) $\omega \in \Omega$ represents all possible aspects of the outcome, e.g. w_2 = "Dice on table, position (2.25, 0.5), 6 facing up, colour red, …". A random variable X maps the outcomes to something we are interested in (like the dice number facing up). An event A can consist of multiple elementary events (like dice roll outcome is even).

2.1.1 Expectation and Variance

- Expectation of X $E(X) = \sum_{x \in \mathbb{R}} x * \Pr(X = x)$
- Variance of X $Var(X) = E\left(\left(X - E(X)\right)^2\right)$

Properties

- $E(\alpha X + \beta Y) = \alpha E(X) + \beta E(Y)$ (linearity)
- $Var(X) = E(X^2) - E(X)^2$
- If X and Y are independent:
  - $E(X * Y) = E(X) * E(X)$
  - $Var(X + Y) = Var(X) + Var(Y)$

Lemma 2.1

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

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

2.1.2 Markov's and Chebyshev's Inequalities

Let X be a random variable (nonnegative for Markov's). Then for all $a, b > 0$:

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

Markov

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

Chebyshev

2.1.3 Weak Law of Large Numbers

Theorem 2.2

Let $X_1, \dots, X_n$ be a pairwise independent sequence of random variables of variance $\mathrm{Var}(X_i) \leq \sigma^2$, and let $X := \sum_{i=1}^{n} X_i$. Then for all $c > 0$,

$$\Pr\left(|X - \mathrm{E}(X)| \geq cn\right) \leq \frac{\sigma^2}{c^2 n}.$$

Corollary 2.3 (Weak Law of Large Numbers)

Let $X_1, X_2, \ldots$ be a sequence of independent identically distributed random variables and $\mu := \mathsf{E}(X_i)$. Then for all $\varepsilon > 0$,

$$\lim_{n\to\infty} \Pr\left(\left|\frac{\sum_{i=1}^{n} X_i}{n} - \mu\right| \geq \varepsilon\right) = 0.$$

2.2 Concentration Inequalities

General Form

- $X = \sum_{i=1}^{n} X_i$
- $\mu = E(X)$

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

2.2.1 Chernoff Bounds

Let $X_1, \ldots, X_n$ be a sequence of independent $\{0, 1\}$-valued random variables. Then for $0 \leq c \leq 1$:

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

$$\Pr(X \leq (1 - c)\mu) \leq e^{-\frac{\mu c^2}{2}}$$

From the above it follows that

$$\Pr(|X - \mu| \geq c\mu) \leq 2e^{-\frac{\mu c^2}{3}}$$

2.2.2 Hoeffding Bounds

Let $X_1, \ldots, X_n$ be a sequence of independent identically distributed $\{0, 1\}$-valued random variables. For $0 \leq d \leq 1$:

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

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

From the above it follows that

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

2.2.3 Concentration on More General Random Variables

Theorem 2.8

Let $X_1, \ldots, X_n$ be a sequence of independent random variables with $\mathsf{E}(X_i) = 0$ and $\mathsf{Var}(X_i) \leq \sigma^2$, 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 $3 \leq k \leq \left\lceil \frac{a^2}{4n\sigma^2} \right\rceil$. Then

$$\Pr(|X| \geq a) \leq 3e^{-\frac{a^2}{12n\sigma^2}}.$$

2.3 Entropy

In the following, $\log(\ldots)$ always refers to $\log_2(\ldots)$, so that information content is measured in bits.

2.3.1 Information Content

We define the information content of an event A to be

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

2.3.2 Entropy

Of a probability distribution $\mathcal{P}$ on a finite sample space Ω:

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

Of a random variable X with finite range (rg):

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

Lemma 2.13

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

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

$$\exists \omega \in \Omega\colon\ \mathcal{P}(\{\omega\}) = 1 \qquad \forall \omega \in \Omega\colon\ \mathcal{P}(\{\omega\}) = 1/n$$

Lemma 2.14 (Jensen's Inequality)

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

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

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

Corollary 2.15 (Log Sum Inequality)

For all $i \in [n]$, let $p_i \in$, $q_i \in$, and let $p := \sum_i p_i$ and $q := \sum_i q_i$. Then

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

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

2.3.3 Decision Tree Learning Revisited

Let S be a set of labelled examples $(\vec{x}, y)$ that consist of a feature vector $\vec{x}$ and a target value y. For each attribute A and value x in the domain $\mathbb{D}_A$ of A we let $S_{A=x}$ be the set of all examples in S with A-value x.

We define the following probability distributions:

$$\mathcal{P}(\{y\}) = \frac{|\{(\vec{x}, y') \in S \mid y' = y\}|}{|S|}$$

$$\mathcal{P}_{A=x}(\{y\}) = \frac{|\{(\vec{x}, y') \in S_{A=x} \mid y' = y\}|}{|S_{A=x}|}$$

Information Gain

To build a decision tree, choose the attribute A that maximizes the information gain $G(S, A)$

$$G(S, A) = H(\mathcal{P}) - \sum_{x \in \mathbb{D}_A} \frac{|S_{A=x}|}{|S|} * H(\mathcal{P}_{A=x})$$

2.4 Compression

Notation

- Source alphabet Σ
- Target alphabet $\{0, 1\}$
- Compression scheme $\Gamma = (com_\Gamma, dec_\Gamma)$
- Compression mapping $com_\Gamma : \Sigma^* \mapsto \{0, 1\}^*$

- Decompression mapping $dec_\Gamma : \{0,1\}^* \mapsto \Sigma^*$
- Probability distribution (cannot be defined on Σ^*)
  - On Σ $\mathcal{P}(\{x_i\})$
  - On Σ^n $\mathcal{P}^n(\{x_1 \dots x_n\}) = \prod_{i=1}^{n} \mathcal{P}(\{x_i\})$

2.4.1 Compression and Loss Rate

The compression rate of Γ for strings of length n is

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

The loss rate of Γ for strings of length n is the probability that a compressed string is not decompressed correctly:

$$\lambda_{\Gamma,\mathcal{P}}(n) = \Pr_{x \sim \mathcal{P}^n}\left(x \neq dec\big(com(x)\big)\right)$$

Goal

Ideally, we would like to have a compression scheme that

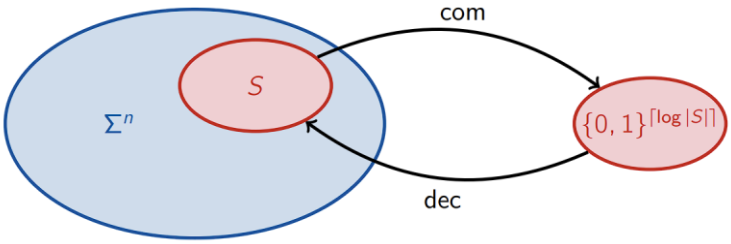
- **guarantees** a good compression rate
- is lossless: $dec\big(com(x)\big) = x$ for all $x \in \Sigma^*$

Observation 2.16

Let $n \in \mathbb{N}$. There is no lossless compression scheme Γ such that $\rho_\Gamma(n) < \log|\Sigma|$.

We need to compromise either on the losslessness or on the compression rate. The goal is to make information loss/bad compression rate unlikely.

2.4.2 Lossy Compression



Idea

- The most likely strings in Σ^n fall into a relatively small set S
- Compress strings in S losslessly
- Consider strings in $\Sigma^n \backslash S$ as losses

Constructing an Optimal Compression Scheme

Choose an $\varepsilon > 0$ and a set $S_\varepsilon(n) \subseteq \Sigma^n$ consisting of the most common symbols with

$$\mathcal{P}^n\big(S_\varepsilon(n)\big) \geq 1 - \varepsilon$$

Define com_ε and dec_ε to be lossless on $S_\varepsilon(n)$. Then for every $n \in \mathbb{N}$ it holds that

- Loss rate $\lambda_\varepsilon(n) \leq \varepsilon$
- Compression rate $\rho_\varepsilon(n) = \frac{\lceil \log |S_\varepsilon(n)| \rceil}{n}$

Lemma 2.17

$$\lim_{n \to \infty} \rho_\varepsilon(n) = H(\mathcal{P})$$

2.4.3 Shannon's Source Coding Theorem

Theorem 2.18 (Source Coding Theorem)

(1) *For every $\varepsilon > 0$ there is a compression scheme Γ_ε over Σ such that $\lambda_{\Gamma_\varepsilon,\mathcal{P}}(n) \leq \varepsilon$ for all n and $\lim_{n\to\infty} \rho_{\Gamma_\varepsilon}(n) = H(\mathcal{P})$.*

(2) *There is no compression scheme Γ such that, for some $\alpha, \beta > 0$, it holds that $\lambda_{\Gamma,\mathcal{P}}(n) \leq 1 - \alpha$ and $\rho_\Gamma(n) \leq H(\mathcal{P}) - \beta$ for infinitely many $n \in \mathbb{N}$.*

→ The optimal compression rate is the entropy of the probability distribution $\mathcal{P}$.

3 Statistical Learning Theory

3.1 The PAC Learning Framework

This mathematical framework allows us to formulate exactly what we expect of a good learning algorithm, as opposed to experimental evaluation.

3.1.1 Formal Framework

- Instance space $\mathbb{X}$
- Probability distribution (generates data) $\mathcal{D}$
- Target function $f^* : \mathbb{X} \to \{0,1\}$
- Hypotheses $h : \mathbb{X} \to \{0,1\}$
- Hypotheses class $\mathcal{H}$ with $h \in \mathcal{H}$
- Training sequence $T = \big((x_1, y_1), \dots, (x_m, y_m)\big) \in (\mathbb{X} \times \{0,1\})^m$ with $y_i = f^*(x_i)$
- Generalisation error (or risk) $err_D(h) = \Pr_{x \sim \mathcal{D}}\big(h(x) \neq f^*(x)\big)$
- Training error (or empirical risk) $err_T(h) = \frac{1}{m}|\{i \in [m] \mid h(x_i) \neq y_i\}|$

Our algorithm receives a training sequence T and produces a hypothesis h_T from some hypothesis class $\mathcal{H}$.
If $f^* \in \mathcal{H}$ then the learning problem is called realisable.
If $err_T(h) = 0$ then h is consistent with T.
Our goal is to minimize the generalisation error $err_D(h)$.

3.1.2 Probably Approximately Correct Learning

An algorithm is a PAC learning algorithm if
for all $\varepsilon, \delta > 0$
there is a number of training instances $m = m(\varepsilon, \delta)$ such that
for every probability distribution $\mathcal{D}$ on $\mathbb{X}$

$$\Pr_{T \sim \mathcal{D}^m}(err_{\mathcal{D}}(h_T) \leq \varepsilon) > 1 - \delta$$

3.1.3 Empirical Risk Minimisation

An ERM algorithm returns a hypothesis h_T such that

$$err_T(h_T) = \min_{h \in \mathcal{H}} err_T(h)$$

To counteract overfitting, we can add a regularisation term ("complexity cost") to the function we minimise:

$$h_T = \arg\min_{h \in \mathcal{H}} \Big(err_T(h) + \rho\big(cost(h)\big)\Big)$$

3.2 Sample Size Bounds

A learning algorithm only sees the training error, but it actually wants to minimise the generalisation error. This section covers bounds of the form:

> "If we see sufficiently many examples then the generalisation error is close to the training error."

3.2.1 Bound: Simple Sample Size Bound

Theorem 3.4 (Simple Sample Size Bound)

Let $\mathcal{H}$ be a finite hypothesis class. Let $\varepsilon, \delta > 0$ and

$$m \geq \frac{1}{\varepsilon} \ln\left(\frac{|\mathcal{H}|}{\delta}\right).$$

Then for any data generating distribution $\mathcal{D}$,

$$\Pr_{T \sim \mathcal{D}^m}\Big(\forall h \in \mathcal{H} : \text{if } h \text{ is consistent with } T, \text{ then } \mathrm{err}_{\mathcal{D}}(h) \leq \varepsilon\Big) > 1 - \delta.$$

An ERM algorithm that for $\varepsilon, \delta > 0$ sees $m \geq \frac{1}{\varepsilon} \ln\left(\frac{|\mathcal{H}|}{\delta}\right)$ examples is a PAC learning algorithm.

3.2.2 Example (Simple Sample Size Bound)

- $\mathcal{H} = \{f : \mathbb{R}^{10} \to \{0,1\}, \boldsymbol{x} \mapsto \langle \boldsymbol{w}, \boldsymbol{x} \rangle - b\}$ with $w_1, \ldots, w_{10}$ and b in 64-bit floating-point numbers
- $|\mathcal{H}| = 2^{11*64} = 2^{704}$
- Let $\varepsilon = 0.1, \delta = 0.05$

If $m \geq \frac{1}{\varepsilon} \ln\left(\frac{|\mathcal{H}|}{\delta}\right) \approx 4910$ and the returned hypothesis is consistent with T,
then with probability larger than 95% the generalisation error is at most 10%.

3.2.3 Bound: Uniform Convergence

Theorem 3.6 (Uniform Convergence)

Let $\mathcal{H}$ be a finite hypothesis class. Let $\varepsilon, \delta > 0$ and

$$m \geq \frac{1}{2\varepsilon^2} \ln\left(\frac{2|\mathcal{H}|}{\delta}\right).$$

Then for any data generating distribution $\mathcal{D}$,

$$\Pr_{T \sim \mathcal{D}^m}\left(\forall h \in \mathcal{H} : \left|\operatorname{err}_T(h) - \operatorname{err}_{\mathcal{D}}(h)\right| \leq \varepsilon\right) > 1 - \delta.$$

3.2.4 Bound: Agnostic PAC Learning

Theorem 3.7 (Agnostic PAC Learning Sample Size Bound)

Consider an ERM algorithm with a finite hypothesis class $\mathcal{H}$ that given T produces a hypothesis h_T.
Let $\varepsilon, \delta > 0$ and

$$m \geq \frac{2}{\varepsilon^2} \ln\left(\frac{2|\mathcal{H}|}{\delta}\right).$$

Let $\mathcal{D}$ be a data generating distribution and $h^* = \arg\min_{h \in \mathcal{H}} \operatorname{err}_{\mathcal{D}}(h)$.
Then

$$\Pr_{T \sim \mathcal{D}^m}\left(\left|\operatorname{err}_{\mathcal{D}}(h_T) - \operatorname{err}_{\mathcal{D}}(h^*)\right| \leq \varepsilon\right) > 1 - \delta.$$

3.2.5 Bound: Description Scheme

Let $\mathcal{H}$ be an arbitrary, possibly infinite, class of hypotheses.

- Assume that we have some scheme Δ for describing the hypotheses $h \in \mathcal{H}$ by strings in Σ^* for some finite alphabet Σ (of size $|\Sigma| \geq 2$).
- For every $h \in \mathcal{H}$, we let $|h|_\Delta$ be the length of the shortest description of h.

Now we can measure the "simplicity" of a hypothesis h as the description length $|h|_\Delta$.

Theorem 3.8

Let $n \in \mathbb{N}$, $\varepsilon, \delta > 0$, and

$$m \geq \frac{1}{\varepsilon}\left(n \ln|\Sigma| + \ln\left(\frac{2}{\delta}\right)\right).$$

Then for any data generating distribution $\mathcal{D}$,

$$\Pr_{T \sim \mathcal{D}^m}\left(\forall h \in \mathcal{H} : \left(|h|_\Delta \leq n \wedge \operatorname{err}_T(h) = 0 \implies \operatorname{err}_{\mathcal{D}}(h) \leq \varepsilon\right)\right) > 1 - \delta.$$

3.2.6 Occam's Razor

Choose the simplest hypothesis consistent with the data.

4 Multiplicative Weight Updates

Notation

- Round $t \geq 1$
- Expert $i \in I$ with $|I| = n$
- Event $j^{(t)} \in J$
- Weight for expert i $w_i^{(t)}$
- Loss matrix $L \in \mathbb{R}^{I \times J}$ where $0 \leq L_{ij} \leq 1$ is the loss for expert i and event j
- Expert probability distribution $\mathcal{D}^{(t)}(\{i\}) = p_i^{(t)} = \frac{w_i^{(t)}}{\sum_{i' \in I} w_{i'}^{(t)}}$
- Expected loss $L^{(t)} = \sum_{i \in I} p_i^{(t)} L_{ij^{(t)}}$

In each round we follow the advice of expert i with probability $p_i^{(t)}$. Thus if event j happens our loss is L_{ij}.

Our goal is to minimize the expected loss.

4.1 The MWU Algorithm

Let some constant $0 < \alpha < 1$.

For all $i \in I$: $w_i^{(1)} = 1$

For $t \geq 1$ and $i \in I$: $w_i^{(t+1)} = (1-\alpha)^{L_{ij^{(t)}}} w_i^{(t)}$

Theorem 4.3

For every $t \geq 1$ and every $i \in I$,

$$\sum_{s=1}^{t} L^{(s)} \leq \frac{\ln n}{\alpha} + (1+\alpha) \sum_{s=1}^{t} L_{ij^{(s)}}$$

In the long run, the algorithm guarantees the losses to be a bit more than twice the losses of the best expert.

4.2 Boosting Weak Learning Algorithms

Here we consider Boolean classification problems again.

4.2.1 Weak and Strong Learner

Strong learner

(⇔ PAC learning algorithm)

For all $\varepsilon, \delta > 0$
There is an $m = m(\varepsilon, \delta)$ such that
For every probability distribution $\mathcal{D}$ on $\mathbb{X}$:

$$\Pr_{T \sim \mathcal{D}^m}(err_{\mathcal{D}}(h_T) \leq \varepsilon) > 1 - \delta$$

Weak learner

(γ-weak learner)

Let $0 \leq \gamma < 1/2$.

For all $\delta > 0$
There is an $m = m(\delta)$ such that
For every probability distribution $\mathcal{D}$ on $\mathbb{X}$:

$$\Pr_{T \sim \mathcal{D}^m}(err_{\mathcal{D}}(h_T) \leq \gamma) > 1 - \delta$$

4.2.2 Boosting

Suppose we have a γ-weak learner $\mathcal{L}$ for our classification problem.

Input: Training sequence $T = ((x_1, y_1), \ldots, (x_n, y_n))$
Error parameter $\varepsilon > 0$

Output: Hypothesis h with $err_T(h) < \varepsilon$

Our goal is to reduce the error of a weak learner to turn it into a strong learner.

- Instance set $X = \{x_1, \dots, x_n\}$
- Probability distribution $\mathcal{D}^{(t)}$ on X
- Confidence parameter $\delta_0 = \frac{1}{10}$
- Number of examples $m_0 = m(\delta_0) \leq n$
- Set of experts $I = [n]$
- Set of events $J = \{\text{hypotheses generated by } \mathcal{L} \text{ on } m_0 \text{ input examples from X}\}$
- Loss matrix $L_{ij} = \begin{cases} 1, & j(x_i) = y_i \\ 0, & \text{otherwise} \end{cases}$
- Update parameter $\alpha = \frac{1}{2} - \gamma$

We call a hypothesis good if $err_{\mathcal{D}}(h) < \gamma$. The weak learner $\mathcal{L}$ generates a good hypothesis with probability at least $1 - \delta_0 = \frac{9}{10}$.

1. Use the MWU algorithm to adapt the weights of $\mathcal{D}^{(t)}$.
2. Re-run $\mathcal{L}$ on $\mathcal{D}$ until it returns a good hypothesis.
3. Repeat for $t^* = \frac{2}{\alpha^2} \ln \frac{1}{\varepsilon}$ rounds
4. Return the following hypothesis

$$h(x) = \begin{cases} 1, & \left|\left\{s \leq t \mid j^{(s)}(x) = 1\right\}\right| \geq \frac{t}{2} \\ 0, & \text{otherwise} \end{cases}$$

Theorem 4.6

$$\text{err}_{\mathcal{T}}(h) < \varepsilon.$$

4.3 Bandit Learning

Notation

- Action $a \in [n]$
- Rounds $1 \leq s \leq t$
- Action probability distribution $\mathcal{D}^{(s)}$ on $[n]$
- Reward of single action $0 \leq q_a^{(s)} \leq 1$
- Payoff matrix $Q = \left(q_a^{(s)}\right) \in \mathbb{R}^{n \times t}$
- Action sequence $\boldsymbol{a} = \left(a^{(1)}, \dots, a^{(t)}\right) \in [n]^t$
- Maximal single-action reward $q_{max}^{(t)} = \max_{a \in [n]} \sum_{s=1}^{t} q_a^{(s)}$
- Reward of action sequence $q(\boldsymbol{a}) = \sum_{s=1}^{t} q_{a^{(s)}}^{(s)}$
- Regret $r(\boldsymbol{a}) = q_{max}^{(t)} - q(\vec{a})$

4.3.1 Multiplicative Weights Update Algorithm

Parameter γ

0 ⟵ γ ⟶ 1

Exploitation — Exploration

Exp3 stands for exponential-weight algorithm for exploration and exploitation.

The main difference to the expert advice scenario is that we cannot observe the reward/loss that the other actions would have given.

Algorithm EXP3

Parameter: γ, where $0 < \gamma \leq 1$.
Initialisation: $w_a^{(1)} := 1$ for all $a \in [n]$

1. for $s = 1, 2, \ldots, t$ do
2. $\quad \mathcal{D}^{(s)}$ probability distribution defined by

$$\mathcal{D}^{(s)}(\{a\}) := p_a^{(s)} := (1-\gamma)\frac{w_a^{(s)}}{\sum_{a'=1}^{n} w_{a'}^{(s)}} + \frac{\gamma}{n}$$

3. $\quad$ action $a^{(s)}$ drawn randomly from $\mathcal{D}^{(s)}$
4. $\quad$ reward $q^{(s)} \leftarrow q_{a^{(s)}}^{(s)}$
5. $\quad$ weights are updated as follows:

$$w_a^{(s+1)} \leftarrow \begin{cases} w_a^{(s)} \cdot \exp\left(\frac{\gamma q^{(s)}}{n p_a^{(s)}}\right) & \text{if } a = a^{(s)}, \\ w_a^{(s)} & \text{otherwise} \end{cases}$$

Corollary 4.8

Set $\gamma^* := \min\left\{1, \sqrt{\frac{n \cdot \ln n}{(e-1) \cdot q_{\max}^{(t)}}}\right\}$.

Then for every payoff matrix, the expected regret of the Exp3 algorithm with parameter γ^* *satisfies*

$$r(Exp3) \leq 2.63 \cdot \sqrt{q_{\max}^{(t)} \cdot n \cdot \ln n}.$$

5 High-Dimensional Data

5.1 The Strange Geometry of High-Dimensional Spaces

5.1.1 The Volume is Near the Surface

Let $X \subseteq \mathbb{R}^l$. By $Vol(X)$ we denote the volume of X.

Theorem 5.2

The volume of high-dimensional objects is concentrated near the surface.
(Formal description omitted)

Think of it as dividing an n-dimensional sphere into cubes of equal size. The higher n is, the more cubes there are near the surface of the sphere compared to near the center.

This causes problems e.g. when finding the k-nearest neighbours in high-dimensional space.

5.1.2 High-Dimensional Unit Balls

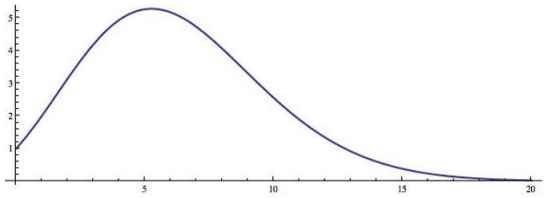
The l-dimensional unit ball is the set

$$B^l = \{\boldsymbol{x} \in \mathbb{R}^l \mid \|\boldsymbol{x}\| \leq 1\}$$

Theorem 5.3 (Volume of the Unit Ball)

$$\lim_{\ell \to \infty} \text{vol}(B^\ell) = 0.$$

Plot of $vol(B^l)$ for values $l = 0, \dots, 20$



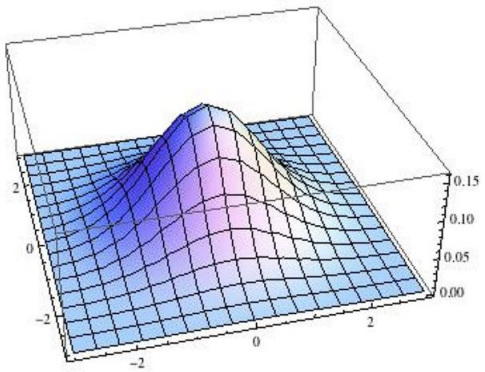
5.2 Dimension Reduction by Random Projections

5.2.1 Spherical Gaussian Distribution

An l-dimensional spherical Gauss distribution with mean $\boldsymbol{\mu} \in \mathbb{R}^l$ and variance σ^2 is

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

Example with $l = 2,\ \mu = 0,\ \sigma^2 = 1$



Lemma 5.7

A spherical Gaussian distribution is obtained by choosing the coordinates independently from a 1-dimensional normal distribution.

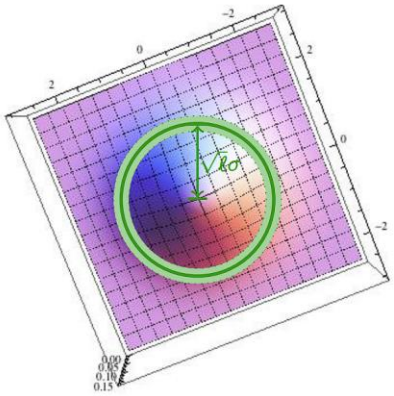
5.2.2 Gaussian Annulus Theorem

Theorem 5.9

Let $b \leq \sqrt{\ell}$, and let $\mathbf{x} \in \mathbb{R}^\ell$ be drawn from an ℓ-dimensional spherical Gaussian distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$. Then

$$\Pr\left(\sqrt{\ell} - b < \|\mathbf{x}\| < \sqrt{\ell} + b\right) \geq 1 - 3e^{-cb^2},$$

for a constant $c > 0$ not depending on ℓ and b.



5.2.3 The Reduction Mapping

Let dimensions $k, l \in \mathbb{N}$ with $k \leq l$.

Draw $\boldsymbol{u_1}, \ldots, \boldsymbol{u_k} \in \mathbb{R}^l$ independently from a l-dimensional Gaussian with mean $\mu = 0$ and variance $\sigma^2 = 1$ and let

$$U = \frac{1}{\sqrt{k}} \begin{pmatrix} \boldsymbol{u_1} \\ \vdots \\ \boldsymbol{u_k} \end{pmatrix} \in \mathbb{R}^{k \times l}$$

The mapping $\boldsymbol{x} \mapsto U\boldsymbol{x}$ is the "random projection" that we use for dimensionality reduction.

5.2.4 The Johnson-Lindenstrauss Lemma

Corollary 5.13 (Johnson-Lindenstrauss Lemma)

Let $0 < \varepsilon < 1$ and $k, \ell, n \in \mathbb{N}$ such that $k \geq \frac{3}{c\varepsilon^2} \ln n$, where c is the constant from the Gaussian Annulus Theorem. Then for every set $X \subseteq \mathbb{R}^\ell$ of size $|X| = n$,

$$\Pr\left(\forall \mathbf{x}, \mathbf{y} \in X : (1-\varepsilon)\|\mathbf{x}-\mathbf{y}\| \leq \|U\mathbf{x} - U\mathbf{y}\| \leq (1+\varepsilon)\|\mathbf{x}-\mathbf{y}\|\right) \geq 1 - \frac{3}{2n}.$$

Here the distance between x and y and the distance between the projections of x and y are compared. It says that the difference between the two distances is likely to be less than ε.

5.3 Background from Linear Algebra

Let a matrix $A \in \mathbb{C}^{n \times n}$. We denote the $n \times n$ identity matrix as I.

5.3.1 Eigenvalues and Eigenvectors

Eigenvalue $\lambda \in \mathbb{C}$

Eigenvector $\boldsymbol{u} \in \mathbb{C}^n \backslash \{\mathbf{0}\}$

$$A\boldsymbol{u} = \lambda \boldsymbol{u}$$

The eigenvalues λ_i of matrix A satisfy the equation

$$\det(A - xI) = \prod_{i=1}^{n} (\lambda_i - x)$$

To calculate the eigenvalues, solve this equation for $x \in \mathbb{C}$

$$\det(A - xI) = 0$$

To calculate the eigenvector for the eigenvalue λ_i, solve this equation for $\boldsymbol{x} \in \mathbb{C}^n$

$$(A - \lambda_i I) * \boldsymbol{x} = \mathbf{0}$$

5.3.2 Spectrum

Geometric multiplicity of λ_i = Dimension of the eigenspace $\{\boldsymbol{u_1}, \ldots, \boldsymbol{u_k}, \mathbf{0}\}$ with $\boldsymbol{u_i}$ eigenvectors of λ_i

Algebraic multiplicity of λ_i = Number of times the factor $(\lambda_i - x)$ appears in $\det(A - xI)$

Spectrum of A $\{\lambda_1, \ldots, \lambda_n\}$

Spectral radius of A $\rho(A) = \max_{i\in[n]}|\lambda_i|$

A matrix A is diagonalisable if there are a non-singular ($\det \neq 0$) matrix U and a diagonal matrix Λ such that

$$U^{-1}AU = \Lambda$$

Theorem 5.18

$A \in \mathbb{C}^{n\times n}$ is diagonalisable if and only if $\mathbb{C}^n$ has a basis consisting of eigenvectors of A.

5.3.3 Spectral Decomposition

A basis $\boldsymbol{u_1}, \dots, \boldsymbol{u_n} \in \mathbb{R}^n$ is called orthonormal if $\forall i \in [n]: \|\boldsymbol{u_i}\| = 1$ and $\forall i, j \in [n]: \langle \boldsymbol{u_i}, \boldsymbol{u_j} \rangle = 0$ with $i \neq j$.

A matrix $U \in \mathbb{R}^{n\times n}$ is called orthogonal if $U^\top U = I$.

Corollary 5.20 (Spectral Decomposition)

Let $A \in \mathbb{R}^{n\times n}$ be a symmetric matrix. Then there is an orthogonal matrix $U \in \mathbb{R}^n$ such that

$$A = U\Lambda U^\top,$$

where Λ is the diagonal matrix whose diagonal entries form the spectrum of A.

5.4 Power Iteration

The power iteration algorithm finds the greatest (in absolute value) eigenvalue and a corresponding eigenvector.

Let $A \in \mathbb{C}^{n\times n}$ be a diagonalisable matrix with spectrum $\lambda_1, \dots, \lambda_n$ where $\lambda_1 \geq 0$ and $\lambda_1 \geq |\lambda_2| \geq \cdots \geq |\lambda_n|$.

Let $\mathbf{x} \in \mathbb{C}^n$ and for $k \geq 0$,

$$\mathbf{x}_k := A^k\mathbf{x}.$$

For all $k \in \mathbb{N}$, let $\hat{\mathbf{x}}_k := \frac{\mathbf{x}_k}{\|\mathbf{x}_k\|}$.

Theorem 5.26

Assume $\mathbf{x}$ is not orthogonal to $\mathbf{u}_1$. Then for $k \to \infty$ the sequence $(\hat{\mathbf{x}}_k)$ converges to an eigenvector of A associated with λ_1.

Algorithm Power-Iteration

Input: Matrix $A \in \mathbb{C}^{n\times n}$, vector $\mathbf{x} \in \mathbb{C}^n$
Output: Vector $\mathbf{v} \in \mathbb{C}^n$

1. $\mathbf{v}_0 \leftarrow \mathbf{x}/\|\mathbf{x}\|$
2. $k \leftarrow 0$
3. repeat
4. $\quad k \leftarrow k + 1$
5. $\quad \mathbf{v}_k \leftarrow \frac{A\mathbf{v}_{k-1}}{\|A\mathbf{v}_{k-1}\|}$
6. until the sequence converges within some tolerance
7. return $\mathbf{v}_k$

The algorithm may not converge if A does not satisfy the above assumptions or x is orthogonal to $\boldsymbol{u_1}$.

5.5 Principal Component Analysis

PCA reduces an l-dimensional data set to a new k-dimensional data set, for some $k \leq l$.

Let a data matrix $A \in \mathbb{R}^{n\times l}$ whose rows are the data points $\boldsymbol{a_1}, \dots, \boldsymbol{a_n} \in \mathbb{R}^l$.

Centering the Data

Subtract the mean from each data point for all $i \in [n]$

$$\boldsymbol{a_i'} = \boldsymbol{a_i} - \frac{1}{n}\sum_{j=1}^{n} \boldsymbol{a_j}$$

If centred, the variance of the data projected to some line $span(\boldsymbol{x})$ is

$$\|Ax\|^2 = \sum_{i=1}^{n} \langle a_i, x \rangle^2$$

Dimension Reduction via PCA

A PCA-transformation of A is an orthogonal matrix $U \in \mathbb{R}^{l \times l}$ with columns $u_1, \ldots, u_l$ where for all $j \in [l]$

$$u_j = \underset{\substack{x \in \mathbb{R}^l \\ \|x\|=1 \\ x \perp u_1, \ldots, u_{j-1}}}{\arg\max} \sum_{i=1}^{n} \langle a_i, x \rangle^2$$

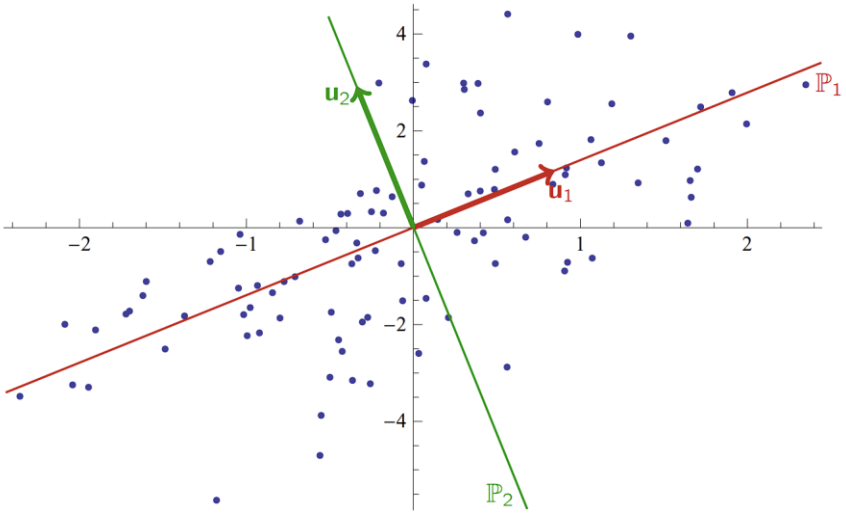
A best-fit k-dimensional subspace for our data is a subspace $\mathbb{X} \subseteq \mathbb{R}^l$ such that the projection of $a_1, \ldots, a_n$ into $\mathbb{X}$ has maximum variance.

Theorem 5.29

Let $A \in \mathbb{R}^{n \times \ell}$, and let U be a PCA-transformation of A with columns $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$. Then for every $k \in [\ell]$,

$$\mathbb{U}_k := \operatorname{span}(\mathbf{u}_1, \ldots, \mathbf{u}_k)$$

is a best-fit k-dimensional subspace for A.



PCA via the Covariance Matrix

The covariance matrix of $A \in \mathbb{R}^{n \times l}$ is

$$C = A^\top A \in \mathbb{R}^{l \times l}$$

Lemma 5.30

C is symmetric and positive semi-definite, that is, all its eigenvalues are nonnegative real numbers.

Theorem 5.31

Let $A \in \mathbb{R}^{n \times \ell}$ be a data matrix and $C = A^\top A$ the corresponding covariance matrix.

Let $\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_\ell$ be the eigenvalues of C. Let $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$ be an orthonormal basis of $\mathbb{R}^\ell$ such that $\mathbf{u}_j$ is an eigenvector of C associated with λ_j, and let $U \in \mathbb{R}^{\ell \times \ell}$ be the matrix with columns $\mathbf{u}_1, \ldots, \mathbf{u}_\ell$.

Then U is a PCA-transformation of A.

Basically $A \longrightarrow C \longrightarrow \lambda_1, \ldots, \lambda_l \longrightarrow u_1, \ldots, u_l \longrightarrow U$

5.6 Spectral Clustering

Setting

We have

- Set of data points X
- Similarity measure $s: X \times X \to \mathbb{R}_{\geq 0}$ with $\forall x, y \in X: s(x,y) = s(y,x)$
- Similarity matrix $S \in \mathbb{R}^{n \times n}$ with $S_{ij} = s(x_i, x_j)$
- Partition matrix $U \in \mathbb{R}^{n \times k}$ with exactly k distinct rows

Goal: Partition X into k clusters $C^1, \dots, C^k$ in a way that minimizes the similarity between points in distinct clusters.

We simplify $s(x_i, x_j)$ to $s(i,j)$ and $x_i \in C^1$ to $i \in C^1$ because the actual data points are not important.

Based on Minimum Cuts

Choose a partition $C^1, \dots, C^k$ that minimizes

$$mincut(C^1, \dots, C^k) = \sum_{p=1}^{k} \sum_{i \in C^p, j \notin C^p} S_{ij}$$

Problem: This method often chooses all clusters but one of size 1.

Based on Balanced Cuts

Choose a partition $C^1, \dots, C^k$ that minimizes

$$balcut(C^1, \dots, C^k) = \sum_{p=1}^{k} \frac{1}{|C^p|} \sum_{i \in C^p, j \notin C^p} S_{ij}$$

Problem: Minimizing the $balcut$ function is computationally hard.

The Laplacian

The trace of a matrix is

$$trace(A) = \sum_{i=1}^{n} a_{ii}$$

Definition 5.33

The Laplacian of S is the matrix

$$L = D - S \in \mathbb{R}^{n \times n},$$

where D is the diagonal matrix with entries $D_{ii} = \sum_{j=1}^{n} S_{ij}$.

Lemma 5.36

Let $C^1, \dots, C^k$ be a partition of $[n]$, and let $U \in \mathbb{R}^{n \times k}$ be the matrix with entries

$$U_{ip} := \begin{cases} \frac{1}{\sqrt{|C^p|}} & \text{if } i \in C^p, \\ 0 & \text{otherwise} \end{cases}$$

Then

$$\text{balcut}(C^1, \dots, C^k) = \text{trace}(U^\top L U).$$

Lemma 5.38

Let $U \in \mathbb{R}^{n \times k}$ be an orthonormal partition matrix, and let $C^1, \dots, C^k$ be the partition of $[n]$ associated with U. Then

$$\text{balcut}(C^1, \dots, C^k) = \text{trace}(U^\top L U).$$

Corollary 5.39

Finding a partition $C^1, \dots, C^k$ that minimises $\text{balcut}(C^1, \dots, C^k)$ is equivalent to finding an orthonormal partition matrix $U \in \mathbb{R}^{n \times k}$ that minimises $\text{trace}(U^\top L U)$.

Theorem 5.40

Let $U \in \mathbb{R}^{n \times k}$ be an orthonormal matrix whose columns are eigenvectors to the k smallest eigenvalues of L. Then

$$\text{trace}(U^\top L U) \leq \text{trace}(V^\top L V)$$

for all orthonormal matrices $V \in \mathbb{R}^{n \times k}$.

Spectral Clustering Algorithm

Algorithm Spectral Clustering

Input: Similarity matrix $S \in \mathbb{R}^{n \times n}$, number k of clusters

1. Compute Laplacian L of S
2. Compute orthonormal matrix $U \in \mathbb{R}^{n \times k}$ whose columns are eigenvectors to the k smallest eigenvalues of L
3. let $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^k$ be the rows of U
4. cluster $\mathbf{x}_1, \ldots, \mathbf{x}_n$ using k-means, yielding clusters $C^1, \ldots, C^k$
5. return $C^1, \ldots, C^k$

6 The Monte Carlo Method

6.1 Estimation through Sampling

6.1.1 Basic Monte Carlo

Goal

Estimate a quantity μ. (usually $\mu = E(f)$)

Method

- Express μ as the expected value of a random variable f defined on a (potentially very large) sample space Ω.
- Draw m independent samples $\omega_1, \ldots, \omega_m \in \Omega$, where m depends on the desired error bound ε and confidence bound δ.
- Return $\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} f(\omega_i)$ as the estimate.

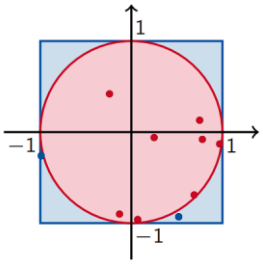
Example: Estimating π

$\Omega = \left\{\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \in \mathbb{R}^2 \mid x_1, x_2 \in [-1, 1]\right\}$

$f(\boldsymbol{\omega}) = \begin{cases} 4, & \|\boldsymbol{\omega}\| \leq 1 \\ 0, & \text{otherwise} \end{cases}$

For samples $\boldsymbol{\omega_1}, \ldots, \boldsymbol{\omega_m} \in \Omega$:

$\hat{\mu} = \frac{4}{m}|\{i \in [m] \mid \|\boldsymbol{\omega_i}\| \leq 1\}| \approx \pi$



Lemma 6.1

Let $\varepsilon, \delta \in \mathbb{R}_{>0}$, and let $b := \max\{|f(\omega)| \mid \omega \in \Omega\}$. If $m \geq \frac{b \ln\left(\frac{2}{\delta}\right)}{\varepsilon^2}$ then

$$\Pr\left(|\hat{\mu} - \mu| \leq \varepsilon\right) \geq 1 - \delta.$$

6.1.2 Rejection Sampling

We want to sample from a probability space $(\Omega, \mathcal{P})$.

-	Target distribution	$\mathcal{P}(\omega) = \frac{\mathcal{D}(\omega)}{Z}$	(normalized $\mathcal{D}(\omega)$)
-	Density function	$\mathcal{D}(\omega)$	
-	Normalizing constant	$Z = \sum_{\omega \in \Omega} \mathcal{D}(\omega)$	(unknown)
-	Upper bound	$\Delta(\omega)$ with $\forall \omega \in \Omega : \mathcal{D}(\omega) \leq \Delta(\omega)$	
-	Proposal distribution	$\Pi(\omega) = \frac{\Delta(\omega)}{\sum_{\omega' \in \Omega} \Delta(\omega')}$	(normalized $\Delta(\omega)$)

Rejection Sampling

1. Sample a point $\omega \in \Omega$ from distribution Π
2. With probability $\frac{\mathcal{D}(\omega)}{\Delta(\omega)}$, accept and return ω; otherwise reject (and repeat).

$\Pr(\text{accepts eventually}) := a = \frac{Z}{\sum_{\omega \in \Omega} \Delta(\omega)} = \frac{\sum_{\omega \in \Omega} \mathcal{D}(\omega)}{\sum_{\omega \in \Omega} \Delta(\omega)}$

$\Pr(\text{accepts in at most } k \text{ iterations}) = 1 - e^{-ak}$

If it accepts, then it returns $\omega \in \Omega$ with probability $\mathcal{P}(\omega)$.

Example

We want to sample points uniformly from $X \subseteq \mathbb{R}^l$. We know a cuboid $Q \subseteq \mathbb{R}^l$ such that $X \subseteq Q$.

- $\Omega = Q$
- $\mathcal{P}(x) = \text{"uniform distribution on } X\text{"} = \frac{\mathcal{D}(x)}{vol(X)}$
- $Z = \int_Q \mathcal{D}(x)\, dx = vol(X)$
- $\mathcal{D}(x) = \begin{cases} 1, & x \in X \\ 0, & x \notin X \end{cases}$
- $\Delta(x) = \begin{cases} 1, & x \in Q \\ 0, & x \notin Q \end{cases}$
- $\Pi(x) = \frac{\Delta(x)}{vol(Q)}$

Observe that $\mathcal{D}(x) \leq \Delta(x)$ for all $x \in \Omega$.

The acceptance probability is $a = \frac{vol(X)}{vol(Q)}$. Since we know $vol(Q)$, this gives us an estimator for $Z = vol(X)$.

6.2 Random Walks and Markov Chains

6.2.1 Formal Framework

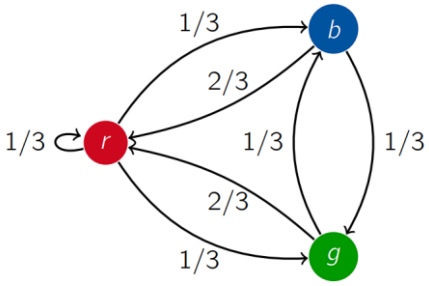
- Markov chain Q with state space $[n]$ for some $n \in \mathbb{N}$
- Transition matrix $Q = (q_{ij}) \in \mathbb{R}^{n \times n}$ where
  - $q_{ij} \geq 0$ is the probability of a transition from state i to state j
  - Q is a stochastic matrix, that is $\sum_{j \in [n]} q_{ij} = 1$ for all $i \in [n]$
- Graph G_Q is defined by $V(G_Q) = [n]$ and $E(G_Q) = \{(i,j) \mid q_{ij} > 0\}$

A Markov chain is a stochastic process that generates a sequence of states by randomly choosing a new state, based only on the current state.

We call the Markov chain connected if for all $i, j \in [n]$ there is a path from i to j in G_Q (G_Q is strongly connected).

Example

$$Q = \begin{pmatrix} \frac{1}{3} & \frac{1}{3} & \frac{1}{3} \\ \frac{2}{3} & 0 & \frac{1}{3} \\ \frac{2}{3} & \frac{1}{3} & 0 \end{pmatrix}$$



6.2.2 Fundamental Theorem of Markov Chains

- Probability vectors $\boldsymbol{p} = (p_1, \dots, p_n) \in \mathbb{R}^{1 \times n}$ where $\sum_{i=1}^{n} p_i = 1$
- Initial probability distribution $\boldsymbol{p_0} = (p_{01}, \dots, p_{0n})$
- Probability distribution after t steps $\boldsymbol{p_t} = (p_{t1}, \dots, p_{tn}) = \boldsymbol{p_0} Q^t$
- Average probability distribution $\boldsymbol{a_t} = (a_{t1}, \dots, a_{tn}) = \frac{1}{t} \sum_{s=1}^{t} \boldsymbol{p_s}$
- Stationary distribution $\boldsymbol{\pi}$

Theorem 6.9

Let $Q \in \mathbb{R}^{n \times n}$ be the transition matrix of a connected Markov chain. Then there is a unique probability vector $\boldsymbol{\pi} \in \mathbb{R}^{1 \times n}$ such that $\boldsymbol{\pi} Q = \boldsymbol{\pi}$. Moreover, for every initial distribution $\mathbf{p}_0 \in \mathbb{R}^{1 \times n}$ the average probability distributions $\mathbf{a}_t$ satisfy

$$\lim_{t \to \infty} \mathbf{a}_t = \boldsymbol{\pi}.$$

Lemma 6.11

Let $Q = (q_{ij}) \in \mathbb{R}^{n \times n}$ be the transition matrix of a connected Markov chain, and let $\boldsymbol{\pi} = (\pi_1, \dots, \pi_n) \in \mathbb{R}^{1 \times n}$ be a probability vector satisfying

$$\pi_i q_{ij} = \pi_j q_{ji}$$

for all $i, j \in [n]$. Then $\boldsymbol{\pi}$ is the stationary distribution of the Markov chain.

Example (continued)

Initial distribution: $\boldsymbol{p_0} = \left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$

Stationary distribution: $\boldsymbol{\pi} = \left(\frac{1}{2}, \frac{1}{4}, \frac{1}{4}\right)$

We can check that $\boldsymbol{\pi Q} = \boldsymbol{\pi}$ or that $\pi_i q_{ij} = \pi_j q_{ji}$ for all $i, j \in [n]$.
For example:

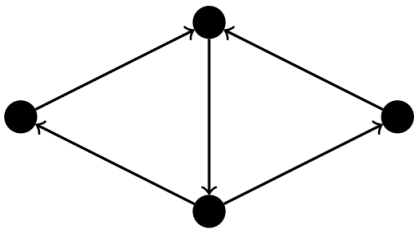
$$\pi_1 q_{13} = \frac{1}{2} * \frac{1}{3} = \frac{1}{6} = \frac{1}{4} * \frac{2}{3} = \pi_3 q_{31}$$

t	$\mathbf{p}_t$	$\mathbf{a}_t$
0	(0.33, 0.33, 0.33)	
1	(0.56, 0.22, 0.22)	(0.56, 0.22, 0.22)
2	(0.48, 0.26, 0.26)	(0.52, 0.24, 0.24)
3	(0.51, 0.25, 0.25)	(0.51, 0.24, 0.24)
4	(0.50, 0.25, 0.25)	(0.51, 0.24, 0.24)
5	(0.50, 0.25, 0.25)	(0.51, 0.25, 0.25)
6	(0.50, 0.25, 0.25)	(0.51, 0.25, 0.25)
7	(0.50, 0.25, 0.25)	(0.51, 0.25, 0.25)
8	(0.50, 0.25, 0.25)	(0.51, 0.25, 0.25)
9	(0.50, 0.25, 0.25)	(0.50, 0.25, 0.25)
10	(0.50, 0.25, 0.25)	(0.50, 0.25, 0.25)

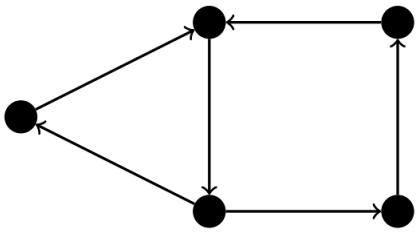
6.2.3 Aperiodic and Ergodic Markov Chains

A Markov chain $\mathcal{Q}$ with graph G_Q is

- Aperiodic if the greatest common divisor of the length of all cycles in G_Q is 1
- Ergodic if it is connected and aperiodic



connected, but periodic



ergodic

Theorem 6.15

Let $\mathcal{Q}$ be an ergodic Markov chain. Let $\mathbf{p}_0, \mathbf{p}_1, \mathbf{p}_2, \ldots$ be the sequence of probability distributions reached by $\mathcal{Q}$ from an arbitrary initial distribution $\mathbf{p}_0$, and let $\boldsymbol{\pi}$ be the stationary distribution. Then

$$\lim_{t\to\infty} \mathbf{p}_t = \boldsymbol{\pi}.$$

Theorem 6.9	Connected	$\Rightarrow \lim_{t\to\infty} \boldsymbol{a_t} = \boldsymbol{\pi}$	(not guaranteed to converge)
Theorem 6.15	Connected, aperiodic	$\Rightarrow \lim_{t\to\infty} \boldsymbol{a_t} = \boldsymbol{\pi}$ and $\lim_{t\to\infty} \boldsymbol{p_t} = \boldsymbol{\pi}$	(guaranteed to converge)

To make a non-ergodic Markov chain ergodic, calculate

$$\alpha Q + (1 - \alpha) I$$

Where $\alpha \in [0, 1]$ and I is the identity matrix. This preserves the stationary distribution.

6.2.4 The Stationary Distribution as Eigenvector

- $\boldsymbol{\pi} = \boldsymbol{\pi Q} \Rightarrow \boldsymbol{\pi}^\top$ is an eigenvector of $Q^\top$ with eigenvalue 1 ("left eigenvector")
- Spectral radius $\rho(Q) = 1$
- Iterating the Markov chain is essentially just the Power Iteration algorithm

6.3 Markov Chain Monte Carlo

6.3.1 Idea

Estimate $\mu = E(f)$ by taking samples $\omega_1, \ldots \omega_m$ and return $\hat{\mu} = \frac{1}{m} \sum_{i=1}^{m} f(\omega_i)$ as an estimator. The difficulty is to sample from the probability space $(\Omega, \mathcal{P})$, e.g. can be very large.

Set up an ergodic Markov chain with state space Ω and stationary distribution $\mathcal{P}$. Simulate until the distribution is close to $\mathcal{P}$ and return the current state as a sample from $(\Omega, \mathcal{P})$.

Notation

Let $\mathcal{Q}$ be a Markov chain with state space Ω. We assume $\mathcal{Q}$ to be ergodic.

- $\mathcal{P}_0$ Initial probability distribution on Ω
- $\mathcal{Q}_{\mathcal{P}_0,t}$ Distribution after t steps
- $\mathcal{Q}_{\omega_0,t}$ Distribution after t steps, if $\mathcal{P}_0(\omega_0) = 1$ for some $\omega_0 \in \Omega$
- $\mathcal{Q}_\infty$ Stationary distribution
- $\mathcal{Q}(\omega, \omega')$ Transition probability

For all $t \in \mathbb{N}, \omega_t \in \Omega$:

$$\mathcal{Q}_{\mathcal{P}_0,t}(\omega_t) = \sum_{\omega_0,\dots,\omega_{t-1}\in\Omega} \mathcal{P}_0(\omega_0) \prod_{i=1}^{t} \mathcal{Q}(\omega_{i-1}, \omega_i)$$

6.3.2 Total Variation Distance

Let two probability distributions $\mathcal{P}, \mathcal{P}'$ over the same state space Ω.

Lemma 6.18

$$\text{dist}_{\text{TV}}(\mathcal{P}, \mathcal{P}') = \frac{1}{2} \sum_{\omega\in\Omega} |\mathcal{P}(\omega) - \mathcal{P}'(\omega)|.$$

6.3.3 Mixing Time

For all $\omega_0 \in \Omega$: $\lim_{t\to\infty} dist_{TV}(\mathcal{Q}_{\omega_0,t}, \mathcal{Q}_\infty) = 0$

Definition 6.19

The mixing time of $\mathcal{Q}$ is the function $T_{\mathcal{Q}} : (0,1) \to \mathbb{N}$ defined by

$$T_{\mathcal{Q}}(\varepsilon) = \min_{t\in\mathbb{N}} \{t \in \mathbb{N} \mid \forall \omega_0 \in \Omega : \text{dist}_{\text{TV}}(\mathcal{Q}_{\omega_0,t}, \mathcal{Q}_\infty) \le \varepsilon\}.$$

6.3.4 Approximation with a Markov Chain

Let $b \in \mathbb{R}$ and f be a random variable on $(\Omega, \mathcal{P})$ with $|f(\omega)| \le b$ for all $\omega \in \Omega$.
We want to estimate

$$\mu = E(f) = \sum_{\omega\in\Omega} f(\omega)\mathcal{P}(\omega)$$

The error (difference to μ) is bounded by the following.

Lemma 6.20

For all $t \in \mathbb{N}$ and all $\omega_0 \in \Omega$, we have

$$\left| \sum_{\omega\in\Omega} f(\omega)\mathcal{Q}_{\omega_0,t}(\omega) - \mu \right| \le 2b \cdot \text{dist}_{\text{TV}}(\mathcal{Q}_{\omega_0,t}, \mathcal{P}).$$

6.3.5 MCMC Theorem

Generate m samples by performing an independent run of $\mathcal{Q}$ with t steps for each sample ω_i.

Let

- $\varepsilon, \delta > 0$
- $m \in \mathbb{N}$ with $m \ge \frac{4b}{\varepsilon^2} \ln\left(\frac{2}{\delta}\right)$
- $t \in \mathbb{N}$ with $t \ge T_{\mathcal{Q}}\left(\frac{\varepsilon}{4b}\right)$
- $\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} f(\omega_i)$

Theorem 6.21 (MCMC Theorem)

Then

$$\Pr\left(|\hat{\mu} - \mu| \le \varepsilon\right) \ge 1 - \delta.$$

6.3.6 Metropolis-Hastings Sampling

We want to construct a Markov chain $\mathcal{Q}$ with $\mathcal{Q}_\infty = \mathcal{P} = \frac{\mathcal{D}}{Z}$ for an unknown normalizing constant Z.
You have to provide a connected undirected graph G of maximum degree Δ with vertex set $V(G) = \Omega$.

Then set transition probabilities

$$\mathcal{Q}(\omega, \omega') = \begin{cases} \frac{1}{\Delta+1} * \min\left(1, \frac{\mathcal{D}(\omega')}{\mathcal{D}(\omega)}\right), & \omega\omega' \in E(G) \\ 1 - \sum_{\omega'' \in N_G(\omega)} \mathcal{Q}(\omega, \omega''), & \omega = \omega' \\ 0, & \text{otherwise} \end{cases}$$

Theorem 6.22
$\mathcal{Q}$ is an ergodic Markov chain with stationary distribution $\mathcal{P}$.

The mixing time $T_{\mathcal{Q}}(\varepsilon)$ is polynomially bounded in $\frac{1}{\varepsilon}$ and the size of the input graph.

6.4 Coupling of Markov Chains

6.4.1 Definition

We view a coupling $\mathcal{C}$ as a pair $(\mathcal{Q}_1, \mathcal{Q}_2)$ of Markov chains running simultaneously.

Definition 6.23
Let $\mathcal{Q}$ be a Markov chain with state space Ω. A coupling of $\mathcal{Q}$ is a Markov chain $\mathcal{C}$ with state space $\Omega \times \Omega$ such that for all $\omega_1, \omega_2, \omega_1', \omega_2' \in \Omega$ it holds that

$$\sum_{\omega' \in \Omega} \mathcal{C}((\omega_1, \omega_2), (\omega_1', \omega')) = \mathcal{Q}(\omega_1, \omega_1'),$$
$$\sum_{\omega' \in \Omega} \mathcal{C}((\omega_1, \omega_2), (\omega', \omega_2')) = \mathcal{Q}(\omega_2, \omega_2').$$

Lemma 6.24
Let $\mathcal{C}$ be a coupling of a Markov chain $\mathcal{Q}$ with state space Ω. Suppose that for some $\varepsilon \in (0,1)$ and $t \in \mathbb{N}$ it holds that

$$\mathcal{C}_{(\omega_{01}, \omega_{02}), t}\left(\{(\omega, \omega) \mid \omega \in \Omega\}\right) \geq 1 - \varepsilon$$

for all $\omega_{01}, \omega_{02} \in \Omega$.
Then $T_{\mathcal{Q}}(\varepsilon) \leq t$.

Interpretation: If the two chains are likely in the same state, then they must be close to the stationary distribution.

6.4.2 Example: Shuffling Cards

Shuffle n cards by repeating:

- Pick random card $i \in [n]$
- Put the card on top of the deck

Question: When is the deck shuffled well?

We let the state space $\Omega = S_n$ be the set of all permutations of $[n]$.
For the previous permutation $\pi \in S_n$, we define the next permutation $\pi_{i \to 1}$ as

$$\pi_{i \to 1}(j) = \begin{cases} \pi(i), & j = 1 \\ \pi(j-1), & 1 < j \leq i \\ \pi(j), & i < j \leq n \end{cases}$$

We define an ergodic Markov chain where $\mathcal{Q}_\infty$ is the uniform distribution on S_n:

$$\mathcal{Q}(\pi, \pi') = \begin{cases} \frac{1}{n}, & \exists i \in [n]: \pi' = \pi_{i \to 1} \\ 0, & \text{otherwise} \end{cases}$$

We define a coupling $\mathcal{C}$ of $\mathcal{Q}$:

$$\mathcal{C}\big((\pi_1,\pi_2),(\pi_1',\pi_2')\big) = \begin{cases} \frac{1}{n}, & \exists i \in [n]\text{: Card with value } i \text{ was moved to the top in both chains} \\ 0, & \text{otherwise} \end{cases}$$

Lemma 6.27

Let $t \geq n \ln\left(\frac{n}{\varepsilon}\right)$. Then for all $\pi_{01}, \pi_{02} \in S_n$,

$$\mathcal{C}_{(\pi_{01},\pi_{02}),t}\Big(\{(\pi,\pi) \mid \pi \in S_n\}\Big) \geq 1-\varepsilon.$$

Corollary 6.28

For all $\varepsilon \in (0,1)$,

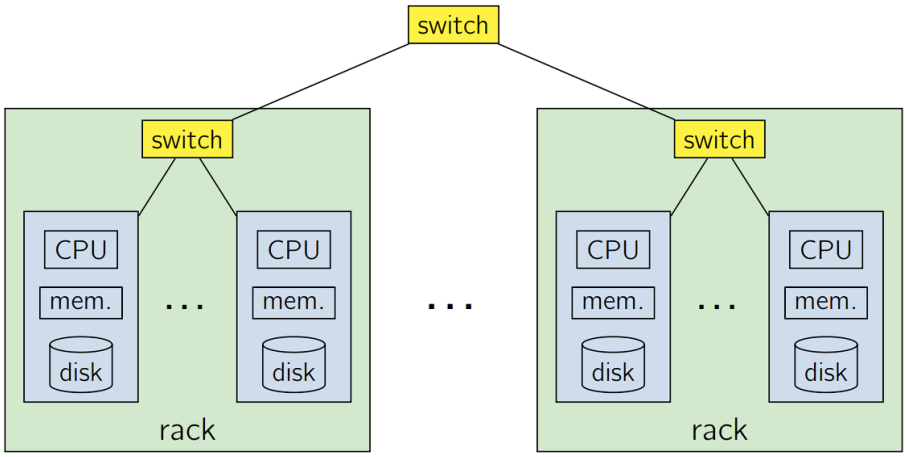
$$T_{\mathcal{Q}}(\varepsilon) \leq n \ln\left(\frac{n}{\varepsilon}\right).$$

7 Algorithms for Massively Parallel Systems

7.1 Computing Clusters and Map-Reduce Environment

7.1.1 Cluster Architecture

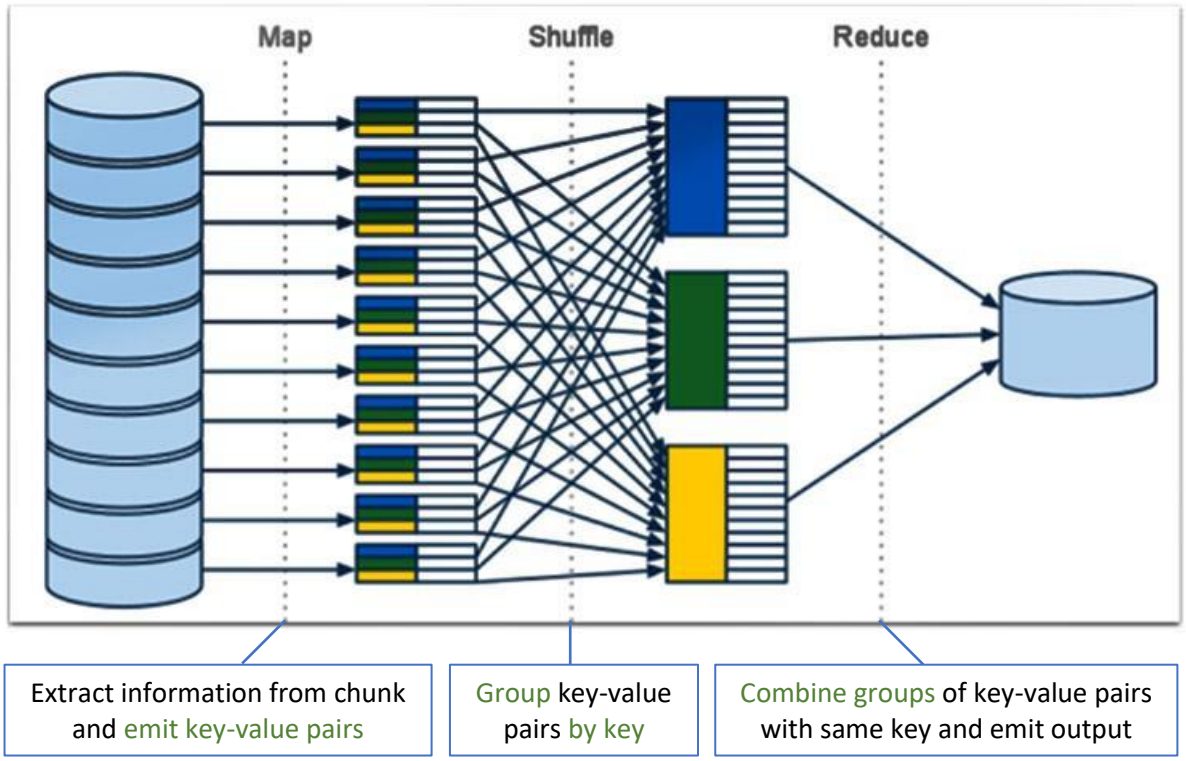
Tasks on massive amounts of data can be carried out in parallel on computing clusters.



Distributed File Systems

- Files are split into chunks that are redundantly stored in multiple racks
- A master node handles
  - Where file chunks are stored
  - Task scheduling
  - Failure detection
- Use the Map-Reduce programming model to distribute computation

7.1.2 Map-Reduce Programming Model



The user has to provide a MAP and a REDUCE function, the system takes care of the rest.

Example: Counting Words

	INPUT	OUTPUT
KEY	Document name	Word
VALUE	Text of document	Count

Map function

```
1. function MAP(key, value)
2.     for all words w in value do
3.         emit (w, 1)
```

Reduce function

```
1. function REDUCE(key, values)
2.     count ← 0
3.     for all v in values do
4.         count ← count + v
5.     emit (key, count)
```

7.2 Map-Reduce Algorithms

7.2.1 Recap: Relational Algebra

- Relation — $\mathcal{R} \subseteq U_1 \times \ldots \times U_k$ — $\{(a73b, \text{Marc}, 19), (6n45, \text{Anna}, 40), \ldots\}$
- Attributes — $A_1, \ldots, A_k$ — "ID, name, age"
- Relation name — R — "Employees"
- Relation schema — $R(A_1, \ldots, A_k)$

Tuples $t \in \mathcal{R}$ of schema $R(A_1, \ldots, A_k)$ are stored as key-value pairs (R, t).

7.2.2 Relational Algebra in Map-Reduce

Projection

Input	Relation $\mathcal{R}$ of schema $R(A, B, C)$	
Output	Relation $Q = \pi_{A,C}(\mathcal{R})$ of schema $Q(A, C)$	
MAP function	On input $\big(R, (a, b, c)\big)$,	emit $\big((a, c), 1\big)$
REDUCE function	On input $\big((a, c), \text{values}\big)$,	emit $\big(Q, (a, c)\big)$

Intersection

Input	Relations $\mathcal{R}, \mathcal{S}$ of the same schema with names R and S	
Output	Relation $Q = \mathcal{R} \cap \mathcal{S}$ with name Q	
MAP function	On input (R, t),	emit (t, R)
	On input (S, t),	emit (t, S)
REDUCE function	On input (t, values),	
	If $R \in \text{values}$ and $S \in \text{values}$	emit (Q, t)

Join

Input	Relation $\mathcal{R}$ of schema $R(A, B)$ and relation $\mathcal{S}$ of schema $S(B, C)$	
Output	Relation $Q = \mathcal{R} \bowtie \mathcal{S}$ with name Q	
MAP function	On input $\big(R, (a, b)\big)$,	emit $\big(b, (R, a)\big)$
	On input $\big(S, (b, c)\big)$,	emit $\big(b, (S, c)\big)$
REDUCE function	On input (b, values),	
	For all $(R, a), (S, c) \in \text{values}$	emit $\big(Q, (a, b, c)\big)$

Group and Average

Input	Relation $\mathcal{R}$ of schema $R(A, B, C)$	
Output	Relation Q of schema $Q(A, C^*)$ where C^* is the average over C for that group	
MAP function	On input $\big(R, (a, b, c)\big)$,	emit (a, c)
REDUCE function	On input (a, values),	
	Compute average c^* of $c \in \text{values}$ and	emit $\big(Q, (a, c^*)\big)$

7.2.3 Matrix-Vector Multiplication

Compute $\boldsymbol{Av}$ for a matrix $A = (a_{ij}) \in \mathbb{R}^{m \times n}$ and vector $\boldsymbol{v} = (v_i) \in \mathbb{R}^n$.

MAP function	On input $\big((i, j), a_{ij}\big)$,	emit $\big(i, a_{ij} v_j\big)$
REDUCE function	On input (i, values),	emit $(i, \sum_{v \in \text{values}} v)$

If $\boldsymbol{v}$ does not fit in memory, split $\boldsymbol{v}$ into segments and A into vertical stripes.

7.2.4 Matrix-Matrix Multiplication

Algorithm in Two Rounds

First Map-Reduce Round

MAP function: On input $(A, (i, j, v))$, emit $(j, (A, i, v))$.
On input $(B, (j, k, w))$, emit $(j, (B, k, w))$.

REDUCE function: On input $(j, values)$, emit $((i, k), vw)$ for all $(A, i, v), (B, k, w) \in values$ such that $vw \neq 0$.

Second Map-Reduce Round

MAP function: The identity function: on input $((i, k), x)$, emit $((i, k), x)$.

REDUCE function: On input $((i, k), values)$, compute the sum x^* of all $x \in values$ and emit $(C, (i, k, x^*))$.

Algorithm in One Round

MAP function: On input $(A, (i, j, v))$, emit all key-value pairs $((i, k), (A, j, v))$ for $k \in [n]$.

On input $(B, (j, k, w))$, emit all key-value pairs $((i, k), (B, j, w))$ for $i \in [\ell]$.

REDUCE function: On input $((i, k), values)$, compute the sum x of all vw for $(A, j, v), (B, j, w) \in values$ and emit $(C, (i, k, x))$.

7.3 Cost Measures

Wall-Clock Time

Total time for the MR-process to finish.

→ Heavily system dependent
→ Too complicated to analyse

Number of Rounds

Number of MR-rounds in the process.

→ Reasonable
→ Not sufficient alone, view in connection with other measures

Communication Cost

Sum of input sizes to all tasks.

→ Reasonable
→ Minimized by using only one node, which is pointless

Replication Rate

Number of key-value pairs produced by all map tasks divided by the input size.

→ Only relevant for single-round MR-processes

Maximum Load

Maximum input length for single reducer or reduce task.

→ Measures load balancing

7.4 Analysis of Matrix Multiplication

Let input matrices $A = (a_{ij}) \in \mathbb{R}^{l \times m}, B = (b_{jk}) \in \mathbb{R}^{m \times n}$ and $p, q \in [0, 1]$ with

$$\Pr(a_{ij} \neq 0) = p \text{ independently for all } i, j$$
$$\Pr(b_{jk} \neq 0) = q \text{ independently for all } j, k$$

Communication Cost

Two-round: Expected communication cost

$$2p\ell m + 2qmn + 2pq\ell mn.$$

Single-round: Expected communication cost

$$p\ell m + qmn + (p+q)\ell mn.$$

Communication cost of two-round algorithm is likely to be much lower.

Maximum Load

Two-round: Expected load of reducer in the first round is $p\ell + qn$; with high probability the maximum load is below $(1+\varepsilon)(p\ell + qn)$.
Expected load of reducer in the second round is pqm; with high probability the maximum load is below $(1+\varepsilon)pqm$.

Single-round: Expected load of reducer is $pm + qm$; with high probability the maximum load is below $(1+\varepsilon)(pm + qm)$.

Maximum load of both algorithms is comparable.

7.5 Multiway Joins in Map-Reduce

We want to compute the natural join $Q = \mathcal{R}_1 \bowtie \cdots \bowtie \mathcal{R}_l$.

Let

- $R_i(A_{i1}, \ldots, A_{ik_i})$ be the schema of $\mathcal{R}_i$,
- $A_1, \ldots, A_k$ be the list of all attributes of all the relations, that is,

$$\{A_1, \ldots, A_k\} = \{A_{ij} \mid i \in [\ell], j \in [k_i]\}$$

- V_i be the set of values of attribute A_i

7.5.1 The Hypercube Algorithm

Parameters

- s = number of reducers
- $s_1, \ldots, s_k \in \mathbb{N}$ such that $\prod_{i=1}^{k} s_i = s$
  s_i is called the share of attribute A_i
- $h_1, \ldots, h_k$ independently chosen hash functions $h_i : V_i \to [s_i]$ (for simplicity assumed to be truly random)

MAP function: On input $(R_i, (a_1, \ldots, a_{k_i}))$, emit all pairs

$$\Big((p_1, \ldots, p_k), (R_i, (a_1, \ldots, a_{k_i}))\Big)$$

such that

- $p_j \in [s_j]$ for all $j \in [k]$,
- $p_j = h_j(a_{j'})$ for all $j \in [k], j' \in [k_i]$ such that $A_{ij'} = A_j$.

REDUCE function: On input $(\overline{p}, \mathit{values})$, compute

$$\mathcal{Q}(\overline{p}) := \mathcal{R}_1(\overline{p}) \bowtie \ldots \bowtie \mathcal{R}_\ell(\overline{p}),$$

where

$$\mathcal{R}_i(\overline{p}) := \{t \mid (R_i, t) \in \mathit{values}\},$$

and emit all pairs (Q, t) for $t \in \mathcal{Q}(\overline{p})$.

8 Streaming Algorithms

8.1 Basics

Data items arrive in a stream. We need to compute on-the-fly because data volume is too large for memory.

Formal Model

- Universe $\mathbb{U}$ with $N = |\mathbb{U}|$
- Input stream $a_1, \dots, a_n$ where $a_i \in \mathbb{U}$

The length of the stream n is not known by the algorithm.

8.1.1 Simple Sampling Algorithm

We want to pick an element a_i uniformly at random from the stream.

Algorithm SIMPLESAMPLE

Input: Stream $a_1, \dots, a_n$ ▷ Assume $n \geq 1$

1. $i \leftarrow 0$
2. while not end of stream do
3. $\quad i \leftarrow i + 1$
4. $\quad sample \leftarrow a_i$ with probability $1/i$
   ▷ otherwise *sample* keeps its current value
5. return *sample*

Example

Let stream $a_1, \dots, a_n = 1, 2, 3, 4, 5, 6$.

$$\Pr(\text{sample} = a_3 = 3)$$
$$= \Pr(a_3 \text{ is chosen}) * \Pr(\text{no } a_i \text{ with } i > 3 \text{ is chosen})$$
$$= \frac{1}{3} * \left(1 - \frac{1}{4}\right) * \left(1 - \frac{1}{5}\right) * \left(1 - \frac{1}{6}\right)$$
$$= \frac{1}{3} * \frac{3}{4} * \frac{4}{5} * \frac{5}{6} = \frac{1}{6} = \frac{1}{n}$$

8.1.2 Reservoir Sampling

We want to pick elements $a_{i_1}, \dots, a_{i_k}$ uniformly at random from the stream.

Algorithm RESERVOIRSAMPLE

Input: Stream $a_1, \dots, a_n$, $k \leq n$

1. for $i = 1, \dots, k$ do
2. $\quad sample[i] \leftarrow a_i$
   ▷ variable i has value k now
3. while not end of stream do
4. $\quad i \leftarrow i + 1$
5. $\quad replace \leftarrow \begin{cases} \text{true} & \text{with probability } \frac{k}{i}, \\ \text{false} & \text{otherwise} \end{cases}$
6. $\quad$ if *replace* then
7. $\quad\quad$ choose j uniformly at random from $[k]$
8. $\quad\quad sample[j] \leftarrow a_i$
9. return *sample*

The algorithm needs space $O(\log n + k * \log N)$.

8.2 Hashing

- Universe $\mathbb{U}$ with $N = |\mathbb{U}|$
- Hash function $h : \mathbb{U} \to \mathbb{T}$
- Family of hash functions $\mathcal{H} \subseteq \mathbb{T}^{\mathbb{U}}$

A hash function that is truly random (has a uniform distribution) would be ideal, but this requires too much memory. Instead, we define a family $\mathcal{H} \subseteq \mathbb{T}^{\mathbb{U}}$ and draw a hash function uniformly at random from that subset.

8.2.1 Universal Hashing

Definition 8.5

A family $\mathcal{H}$ of hash functions from $\mathbb{U}$ to $\mathbb{T}$ is universal if for all distinct $x, x' \in \mathbb{U}$,

$$\Pr_{h\in\mathcal{H}}\left(h(x) = h(x')\right) \leq \frac{1}{|\mathbb{T}|}.$$

Interpretation: A function from a universal family likely has a low number of collisions. The values $h(x)$ are evenly distributed on $\mathbb{T}$.

Collisions

For a function $h : \mathbb{U} \to \mathbb{T}$ and a set $S \subseteq \mathbb{U}$, the number of collisions is

$$coll(h,S) = \left|\left\{\{x,x'\} \in \binom{S}{2} \mid h(x) = h(x')\right\}\right|$$

Where $\binom{S}{2}$ denotes the set of 2-element subsets of S.

Lemma 8.8

Let $\mathcal{H}$ be a universal family of hash functions from $\mathbb{U}$ to $\{0,\ldots,2^k-1\}$. Then for every $\delta > 0$ and every set $S \subseteq \mathbb{U}$ of cardinality $|S| = n$,

$$\mathrm{E}_{h\in\mathcal{H}}\left(\mathrm{coll}(h,S)\right) = \frac{n(n-1)}{2^{k+1}}$$

and $\Pr_{h\in\mathcal{H}}\left(\mathrm{coll}(h,S) \geq \frac{n^2}{\delta 2^{k+1}}\right) \leq \delta.$

8.2.2 Strongly k-Universal Families

Definition 8.10

Let $k \geq 2$, and let $\mathcal{H}$ be a family of hash functions from $\mathbb{U}$ to $\mathbb{T}$.

(1) $\mathcal{H}$ is k-universal if for all distinct $x_1,\ldots,x_k \in \mathbb{U}$,

$$\Pr_{h\in\mathcal{H}}\left(h(x_1) = h(x_2) = \ldots = h(x_k)\right) \leq \frac{1}{|\mathbb{T}|^{k-1}}$$

(2) $\mathcal{H}$ is strongly k-universal if or all distinct $x_1,\ldots,x_k \in \mathbb{U}$ and all $y_1,\ldots,y_k \in \mathbb{T}$,

$$\Pr_{h\in\mathcal{H}}\left(h(x_1) = y_1 \wedge \ldots \wedge h(x_k) = y_k\right) = \frac{1}{|\mathbb{T}|^k}$$

Implications

- $\mathcal{H}$ is 2-universal $\Leftrightarrow$ $\mathcal{H}$ is universal
- $\mathcal{H}$ is strongly k-universal $\Rightarrow$ $\mathcal{H}$ is k-universal

Alternative Definition of Strongly k-Universal Families

Lemma 8.12

Let $2 \leq k \leq |\mathbb{U}|$, and let $\mathcal{H}$ be a family of hash functions from $\mathbb{U}$ to $\mathbb{T}$. Then $\mathcal{H}$ is strongly k-universal if and only if it has the following two properties.

k-Independence: For all distinct $x_1,\ldots,x_k \in \mathbb{U}$ and all $y_1,\ldots,y_k \in \mathbb{T}$,

$$\Pr_{h\in\mathcal{H}}\left(\bigwedge_{i=1}^{k} h(x_i) = y_i\right) = \prod_{i=1}^{k} \Pr_{h\in\mathcal{H}}\left(h(x_i) = y_i\right).$$

That is, the indicator random variables for the events $h(x_i) = y_i$ are independent.

Uniformity: For all $x \in \mathbb{U}$ and $y \in \mathbb{T}$,

$$\Pr_{h\in\mathcal{H}}\left(h(x) = y\right) = \frac{1}{|\mathbb{T}|}$$

Construction of Strongly k-Universal Families

It remains to construct a strongly k-universal family mapping $\mathbb{U}$ to $\mathbb{T}:=\{0,\dots,M-1\}$ for an $M \ll N$.

- We choose a prime power $q \geq N$ and define the mappings $f_{\mathbf{a}} : \mathbb{U} \to \{0,\dots,q-1\}$ for $\mathbf{a} \in \mathbb{F}_q^k$ as on the previous slide.
- We define functions $h_{\mathbf{a}} : \mathbb{U} \to \{0,1\dots,M-1\}$ by

$$h_{\mathbf{a}}(x):=f_{\mathbf{a}}(x) \mod M.$$

- We let $\mathcal{H}_{q,M}^k:=\{h_{\mathbf{a}} \mid \mathbf{a} \in \mathbb{F}_q^k\}$.

Theorem 8.14

If M divides q, then the family $\mathcal{H}_{q,M}^k$ is strongly k-universal.

8.3 Counting Distinct Elements

Given a data stream, we want to count the number of distinct elements in $a_1, \dots, a_n$.

Theorem 8.17

The problem cannot be solved (exactly) in space $< \min\{N, n\}$.

8.3.1 Estimating the Size of a Set

Let $S \subseteq \mathbb{U}$ be random subset with $1 \leq m = |S| \leq N = |\mathbb{U}|$.

Use $\frac{N}{\min S}$ as an estimator for m. Then

$$E_{S\sim\binom{\mathbb{U}}{m}}(\min S) \approx \frac{N}{m} \quad \Rightarrow \quad \frac{N}{E_{S\sim\binom{\mathbb{U}}{m}}(\min S)} \approx m$$

Note

We get a more precise estimate if we compute the t-th smallest element and return $\frac{tM}{x}$ as our estimator (for some constant t depending on the approximation error).

8.3.2 The Approximate Counting Algorithm

Let $\varepsilon > 0, M \geq 12N^2$ and $\mathcal{H}$ be a strongly 2-universal family of hash functions from $\mathbb{U}$ to $[M]$. A hash function is used, because the values need to be uniformly distributed from 0 to M for the estimator to work.

Algorithm COUNT(ε)

1. $h \sim \mathcal{H}$
2. $t \leftarrow \lceil \frac{24}{\varepsilon^2} \rceil$
3. $(x_1,\dots,x_t) \leftarrow (M+1,\dots,M+1)$
4. while not end of stream do
5. $\quad a \leftarrow$ next stream element
6. $\quad$ INSERT$((x_1,\dots,x_t), h(a))$
7. if $x_t \leq M$ then
8. $\quad$ return $\frac{tM}{x_t}$
9. else
10. $\quad i \leftarrow \max\{j \leq t \mid x_j < M+1\}$
11. $\quad$ return i

INSERT$((x_1,\dots,x_t), y)$

1. $i = 1$
2. while $y > x_i$ do
3. $\quad i \leftarrow i+1$
4. if $i \leq t$ and $y < x_i$ then
5. $\quad$ for $j = i+1$ to t do
6. $\quad\quad x_j \leftarrow x_{j-1}$
7. $\quad x_i \leftarrow y$
8. return $(x_1,\dots,x_t)$

The algorithm maintains a sorted list $x_1, \dots, x_t$ of the $t = \left\lceil \frac{24}{\varepsilon^2} \right\rceil$ smallest stream elements.

It returns $\frac{tM}{x_t}$ as the estimator.

Space

Algorithm COUNT(ε) needs memory space

$$O(t \log M) = O\left(\frac{1}{\varepsilon^2} \log N\right).$$

Approximation Guarantee

Let m be the number of distinct elements in the input stream, and let m^* be the estimator returned by $\textsc{Count}(\varepsilon)$. Then

$$\Pr\Big((1-\varepsilon)m \le m^* \le (1+\varepsilon)m\Big) \ge \frac{3}{4}.$$

8.3.3 The Median Trick

For some $k \ge 1$,

- Run $2k-1$ copies of $\textsc{Count}(\varepsilon)$ in parallel with hash functions $h_1, \dots, h_{2k-1}$ drawn independently from $\mathcal{H}$.
- Let $m^{(1)}, \dots, m^{(2k-1)}$ be the resulting estimators for the number d of distinct elements in the input stream.
  Return the median m^* of $m^{(1)}, \dots, m^{(2k-1)}$.

Call this version of the algorithm $\textsc{MCount}(\varepsilon, k)$.

Approximation Guarantee

For every $\delta > 0$ there exists a $k = O(\ln(1/\delta))$ such that the estimator m^* returned by $\textsc{MCount}(\varepsilon, k)$ satisfies

$$\Pr\Big((1-\varepsilon)m \le m^* \le (1+\varepsilon)m\Big) \ge 1-\delta.$$

As before, m denotes the number of distinct elements in the input stream.

8.4 Frequency Moments

8.4.1 Frequencies

Let $\boldsymbol{a} = a_1, \dots, a_n$ be a data stream of elements from $\mathbb{U}$.

The frequency of $u \in \mathbb{U}$ in $\boldsymbol{a}$ is

$$f_u(\boldsymbol{a}) = |\{\, i \in [n] \mid a_i = u \,\}|$$

The pth frequency moment for $p \ge 0$ of $\boldsymbol{a}$ is

$$F_p(\boldsymbol{a}) = \sum_{\substack{u \in \mathbb{U} \\ f_u(\boldsymbol{a}) > 0}} \big(f_u(\boldsymbol{a})\big)^p$$

Interpretation

- $F_0 = m$ — Number of distinct elements in $\boldsymbol{a}$
- $F_1 = n$ — Length of $\boldsymbol{a}$
- $\sqrt[p]{F_p} = \|\boldsymbol{f}\|_p$ — The ℓ_p-norm of $\boldsymbol{f} = (f_u)_{u \in \mathbb{U}}$ (Euclidean norm for $p = 2$)
- $\sqrt[p]{F_p} \xrightarrow[p \to \infty]{} \max\{f_u \mid u \in \mathbb{U}\}$

8.4.2 AMS Estimator for F_k

Let $k \in \mathbb{N}_{\ge 2}$.

Estimator A_k

- Pick an index $i \in [n]$ uniformly at random.
- Let $r := |\{j \ge i \mid a_j = a_i\}|$ (the number of occurrences of a_i in the "rest of the stream" starting at position i).
- Let $A_k := n(r^k - (r-1)^k)$.

Then

$$E(A_k) = F_k$$

Algorithm AMS-Estimator

1. $i = 0$
2. while not end of stream do
3. $\quad i \leftarrow i + 1$
4. $\quad$ with probability $1/i$ do
5. $\quad\quad a \leftarrow a_i$
6. $\quad\quad r \leftarrow 0$
7. $\quad$ if $a_i = a$ then
8. $\quad\quad r \leftarrow r + 1$
9. return $i(r^k - (r-1)^k)$

Problem: The variance of the estimator A_k is very high.

8.4.3 Tug-of-War Estimator for F_2

Let $\mathcal{H}$ be a strongly 4-universal family of hash functions from $\mathbb{U}$ to $\{-1, 1\}$.

Algorithm Tug-of-War

1. draw h uniformly at random from $\mathcal{H}$
2. $x \leftarrow 0$
3. while not end of stream do
4. $\quad a \leftarrow$ next element from stream
5. $\quad x \leftarrow x + h(a)$
6. return x^2

Lemma 8.21

Let B be the estimator returned by the algorithm Tug-of-War. Then $\mathsf{E}(B) = F_2$ *and* $\mathsf{Var}(B) \leq 2F_2^2$.

Averaging the Tug-of-War Estimator

Algorithm Avg-ToW(k)

1. draw $h_1, \ldots, h_k$ independently from $\mathcal{H}$
2. for $i = 1, \ldots, k$ do
3. $\quad x_i \leftarrow 0$
4. while not end of stream do
5. $\quad a \leftarrow$ next element from stream
6. $\quad$ for $i = 1, \ldots, k$ do
7. $\quad\quad x_i \leftarrow x_i + h_i(a)$
8. return $\frac{1}{k}\sum_{i=1}^{k} x_i^2$

Theorem 8.23

Let $\varepsilon, \delta > 0$ and $k = \lceil \frac{2}{\varepsilon^2 \delta} \rceil$. Let B be the estimator returned by the algorithm Avg-ToW(k). *Then $E(B) = F_2$ and*

$$\Pr\left(|B - F_2| < \varepsilon F_2\right) > 1 - \delta.$$

8.5 Sketching

We want to query a vector that changes over time according to a sequence of updates that arrive in a stream. Both the vector and the stream are too large to be stored in memory.

8.5.1 Turnstile Data Stream Model

- Universe — $\mathbb{U}$ with $|\mathbb{U}| = N$
- Data vector — $\boldsymbol{d}(i) = \left(d_u(i)\right)_{u \in \mathbb{U}} \in \mathbb{Z}^N$ with $0 \leq i \leq n$
- Final data vector — $\boldsymbol{d} = \boldsymbol{d}(n)$
- Stream of updates — $(a_1, c_1), \ldots, (a_n, c_n)$ where $a_i \in \mathbb{U}$ and $c_i \in \mathbb{Z}$

The data vector $\boldsymbol{d}(i)$ is defined for all $u \in \mathbb{U}, i \in [n]$ by

$$d_u(0) = 0$$
$$d_u(i+1) = \begin{cases} d_u(i) + c_i & u = a_i \\ d_u(i) & u \neq a_i \end{cases}$$

Restrictions

Strict Turnstile Model	$d_u(i) \geq 0$	for all $u \in \mathbb{U}, i \in [n]$
Cash Register Model	$c_i > 0$	for all $i \in [n]$

ℓ_1-norm

$$\|d\|_1 = \sum_{u \in \mathbb{U}} d_u$$

8.5.2 Simple Sketch Algorithm

Let $k \geq 1$ and let $\mathcal{H}$ be a universal family of hash functions from $\mathbb{U}$ to $[k]$.

Algorithm SIMPLE SKETCH(k)

1. draw h from $\mathcal{H}$
2. for $i = 1, \ldots, k$ do
3. $\quad S[i] := 0$
4. while not end of stream do
5. $\quad (a, c) \leftarrow$ next update
6. $\quad S[h(a)] \leftarrow S[h(a)] + c$
7. return S

To estimate d_u, we use the value $d_u^* = S[h(u)]$.

Lemma 8.24

Let $\varepsilon > 0$ such that $k \geq \frac{2}{\varepsilon}$. Then in the strict turnstile model, the following holds for the estimator d_u^*:

(i) $d_u \leq d_u^*$.

(ii) $\mathsf{E}(d_u^* - d_u) \leq \frac{\varepsilon}{2}\|\mathbf{d}\|_1$ *and* $\Pr\left(d_u^* - d_u \geq \varepsilon\|\mathbf{d}\|_1\right) \leq \frac{1}{2}$.

8.5.3 Count Min Sketch Algorithm

Let $k, l \geq 1$ and let $\mathcal{H}$ be a universal family of hash functions from $\mathbb{U}$ to $\{0, \ldots, k-1\}$.

Algorithm COUNT MIN SKETCH(k, ℓ)

1. draw $h_1, \ldots, h_\ell$ independently from $\mathcal{H}$
2. for $i = 1, \ldots, k$ do
3. $\quad$ for $j = 1, \ldots, \ell$ do
4. $\quad\quad S[i, j] \leftarrow 0$
5. while not end of stream do
6. $\quad (a, c) \leftarrow$ next update
7. $\quad$ for $j = 1, \ldots, \ell$ do
8. $\quad\quad S[h_j(a), j] \leftarrow S[h_j(a), j] + c$
9. return S

We call the returned array S a CM sketch with parameters k, l.

To estimate d_u, we use the value $d_u^* = \min_{j \in [l]} S\left[h_j(u), j\right]$.

Theorem 8.25

Let $\varepsilon, \delta > 0$ such that $k \geq \frac{2}{\varepsilon}$ and $\ell \geq \log \frac{1}{\delta}$. Then in the strict turnstile model, the following holds for the estimator d_u^*:

(i) $d_u \leq d_u^*$.

(ii) $\Pr\left(d_u^* - d_u \geq \varepsilon\|\mathbf{d}\|_1\right) \leq \delta$.

8.5.4 Heavy Hitters

An element $u \in \mathbb{U}$ is a heavy hitter with threshold $\tau > 0$ for a data vector $\mathbf{d}$ if

$$d_u \geq \tau\|\mathbf{d}\|_1.$$

Observation 8.26

There are at most $\frac{1}{\tau}$ heavy hitters with threshold τ.

Let $\tau > 0$ and $k, l \geq 1$ and let $\mathcal{H}$ be a universal family of hash functions from $\mathbb{U}$ to $\{0, \ldots, k-1\}$.

Algorithm CM Heavy Hitters(k, ℓ, τ)

- compute a CM sketch S with parameters k, ℓ
- maintain $\|\mathbf{d}\|_1$ during the computation
- during the computation, maintain a set H of elements $u \in \mathbb{U}$ whose estimated value $d_u^* := \min_j S[h_j(u), j]$ is at least $\tau\|\mathbf{d}\|_1$
- after each update (or whenever H gets too large), remove those elements u whose value d_u^* has dropped below $\tau\|\mathbf{d}\|_1$ from H
- return all $u \in H$ with $d_u^* \geq \tau\|\mathbf{d}\|_1$

Theorem 8.28

We assume the cash register model.
Let $\varepsilon, \delta, \tau > 0$ and $k \geq \frac{2}{\varepsilon}$ and $\ell \geq \log \frac{n}{\delta}$.
Then the algorithm CM Heavy Hitters(k, ℓ, τ) returns

(i) all elements u such that $d_u \geq \tau\|\mathbf{d}\|_1$,

(ii) with probability at least $1 - \delta$ no elements u such that $d_u \leq (\tau - \varepsilon)\|\mathbf{d}\|_1$.