1 Understanding Machine Learning Algorithms

1.1 Basic ML Concepts and Terminology

- *Domain* $\mathbb{R}$ (possible values), *instance space* $\mathbb{R}^l$ (feature vector space), dataset in $\mathbb{R}^{n \times l}$

Unsupervised Learning	detect patterns	clustering
Reinforcement Learning	find actions maximizing reward	
Supervised Learning	learn function from examples	classification/regression
Semi-Supervised Learning	sparse/faulty data	data completion

- *Batch Learning* (all examples at once) vs. Online Learning (improve hypothesis over time)
- *Passive Learning* (no influence on choice of examples) vs. Active Learning (actively choose specific data points)
- *Hypothesis space* $\mathcal{H}$: Set of functions the algorithm can return
  - Problem is *realizable*, if target function $f \in \mathcal{H}$
  - Hypothesis h is *consistent* with training sequence $S = ((\mathbf{x}_i, y_i))_{1 \leq i \leq m}$, if $h(\mathbf{x}_i) = y_i$ for all i
- *Occam's Razor*: Simpler hypotheses tend to generalize better

1.2 The Nearest Neighbor Algorithm

- **Metric space**: $(\mathbb{X}, d)$, with distance measure $d : \mathbb{X} \to \mathbb{R}$ s.t.
  - Nonnegativity: $d(x, y) \geq 0 \wedge [d(x, y) = 0 \Leftrightarrow x = y]$
  - Symmetry: $d(x, y) = d(y, x)$
  - Triangle inequality: $d(x, z) \leq d(x, y) + d(y, z)$
- Metric examples: Manhattan (l_1), Euclidean (l_2), Chebychev (l_∞), Hamming
- **k-NN Algorithm**
  - For $x \in \mathbb{X}$ find k nearest neighbors, majority vote on labels.
  - Use appropriate data structure or hashing to reduce $\mathcal{O}(n)$ search time.

1.3 Learning Decision Trees

- **Decision Tree** (DT): Finite-valued function on feature vectors, nodes labeled by input features, edges labeled by value or range of values for feature, leafs labeled with output feature
- DT Training: **ID3 Algorithm** (choose feature that discriminates best, create node with feature; iterate)
- Boolean function representation with DT: Features are variables X_i, feature values are $\{0, 1\}$
  - $f : \{0, 1\}^n \to \{0, 1\}$ represented by height $\leq n$ with $\leq 2^{n+1} - 1$ nodes

Decision Tree (NP-complete, $\geq_p$ Vertex Cover)

Given examples $(\mathbf{x}_i, y_i) \in \{0, 1\}^n \times \{0, 1\}$, $k \in \mathbb{N}_{>0}$.
Decide, if DT exists with at most k nodes consistent with examples.

1.4 Linear Classifiers

- Chauchy-Schwarz Inequality: $|\langle \mathbf{x}, \mathbf{y} \rangle| \leq ||\mathbf{x}|| \cdot ||\mathbf{y}||$
- **Hyperplane**: Affine subspace $P = \{\mathbf{x} \mid \langle \mathbf{a}, \mathbf{x} \rangle - b = 0\}$ ($\mathbf{a}$ orthogonal to hyperplane; b y-intercept)
- **Halfspace**: One side of hyperplane $H = \{\mathbf{x} \mid \langle \mathbf{a}, \mathbf{x} \rangle - b \geq 0\}$
- H or P *homogeneous*, if $b = 0$ (or $\mathbf{0} \in P$)
- *Linear Boolean Classification* $f : \mathbb{R}^l \to \{-1, +1\}$
  - $\mathcal{H} = \{h(\mathbf{x}) = \text{sgn}(\langle \mathbf{w}, \mathbf{x} \rangle - b)\}$
  - $\hat{\mathbf{x}} \mapsto \text{sgn}(\langle \hat{\mathbf{w}}, \hat{\mathbf{x}} \rangle)$ with $\hat{\mathbf{x}} = (\mathbf{x}, 1)^T$, $\hat{\mathbf{w}} = (\mathbf{w}, -b)^T$ is consistent homogeneous linear separator

Empirical Risk Minimization (ERM)

Given training data $S = ((\mathbf{x}_i, y_i))_{1 \leq i \leq m} \in (\mathbb{R}^l \times \{-1, 1\})^m$.
Compute $\mathbf{w} \in \mathbb{R}^l$ s.t. homogeneous linear separator $\mathbf{x} \mapsto \text{sgn}(\langle \mathbf{w}, \mathbf{x}_i \rangle)$ *minimizes* number of wrong predictions.

- *Linear Programming* (LP): Compute $\mathbf{x}$ minimizing $\langle \mathbf{c}, \mathbf{x} \rangle$ subject to $A\mathbf{x} \geq \mathbf{b}$ (solvable in polynomial time)
  - $(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle > 0 \Leftrightarrow y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1 \Leftrightarrow) \langle \mathbf{w}, y_i \mathbf{x}_i \rangle \geq 1$ solvable with LP ($\min_{\mathbf{w}} \langle \mathbf{0}, \mathbf{w} \rangle$ subject to $(y_i x_{ij})_{i,j} \mathbf{w} \geq 1$)
- **Perceptron Algorithm**: Init $\mathbf{w} = 0$, increment by $y_i \mathbf{x}_i$ for all wrongly classified $(\mathbf{x}_i, y_i) \in S$ as long as they exist
  - **Margin** of separator consistent with S: $\min_{(\mathbf{x},y) \in S} \frac{|\langle \mathbf{w}, \mathbf{x} \rangle|}{||\mathbf{w}||}$
  - With margin γ and $\lambda := \max_{(\mathbf{x},y) \in S} ||\mathbf{x}||$, the number of updates on $\mathbf{w}$ in the perceptron algorithm is bounded by $\left(\frac{\lambda}{\gamma}\right)^2$.

    SVM

    Given training data $S \in (\mathbb{R}^l \times \{-1, 1\})^m$.
    Compute $\mathbf{w}^* \in \mathbb{R}^l$, minimizing $||\mathbf{w}||^2$ subject to $y_i \langle \mathbf{w}, \mathbf{x}_i \rangle \geq 1, i \in [m]$.
  - $||\mathbf{w}||^2$ maximizes w.r.t. S. Since quadratic, problem is no longer solvable with LP, use, e.g., gradient descent.
  - Only in realizable case. *Soft margin* SVM required in general case.
- **Logistic Regression**
  - Return confidence: likely $-1 \mapsto 0$, likely $+1 \mapsto 1$ using sigmoid $\text{sig}(z) = 1/(1 + e^{-z})$

$$\text{sig}(y_i \langle \mathbf{w}, \mathbf{x}_i \rangle) = \begin{cases} \text{sig}(\langle \mathbf{w}, \mathbf{x}_i \rangle) & \text{if } y_i = +1 \\ 1 - \text{sig}(\langle \mathbf{w}, \mathbf{x}_i \rangle) & \text{if } y_i = -1 \end{cases}$$

  - Maximization of confidence product over all examples is minimization of the logarithmic sum (when taking $-\ln(\dots)$ of product); convex minimization problem solvable with gradient descent

    Logistic Regression

    Given training data $S \in (\mathbb{R}^l \times \{-1, 1\})^m$.
    Compute $\mathbf{w}^* \in \mathbb{R}^l$, minimizing $\sum_i \ln(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x}_i \rangle))$

- *Non-Linear Problems*: Map data to higher-dimensional space, where nonlinear functions may be transformed to linear ones
  - Example: $\tau(x_1, x_2) = (x_1^2, x_1x_2, x_2^2, x_1, x_2)$

1.5 k-Means Clustering

- *Algorithm*: Random init of centroids. Iteratively assign points to the cluster with the closest centroid (tie-breaking: cluster with smallest index), then update centroids as mean of assigned points. Repeat until no change in cluster.
- Always converges (local minimum), solution may depend on choice of initial centroids

Centroid Clustering (NP-complete)

Given points $\mathbf{x}_i \in \mathbb{R}^l, i \in [m], k \in \mathbb{N}$.
Find $\mathbf{z}^1, \ldots, \mathbf{z}^k$ and partition $C^1, \ldots, C^k$ of $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ minimizing $\sum_j \sum_{x \in C^j} ||\mathbf{x} - \mathbf{z}^j||^2$

2 Information and Compression

2.1 Background from Probability Theory

- **Probability spaces**
  - Finite sample space Ω, event space $\mathcal{E} = 2^\Omega$, prob. distr. $\mathcal{P} : \mathcal{E} \to [0, 1]$
  - Sufficient to specify $p : \Omega \to [0, 1]$ with $\sum_{\omega \in \Omega} p(\omega) = 1$ and $\mathcal{P}(A) := \sum_{\omega \in A} p(\omega)$ for $A \subseteq \Omega$ (discrete case)
- **Random variables**

Random variable	$X : \Omega \to \mathbb{R}$
Probability mass function of X	$f_X : \mathbb{R} \to [0, 1], x \mapsto \Pr(X = x)$
Cumulative distribution of X	$F_X : \mathbb{R} \to [0, 1], F_X(x) = \Pr(X \leq x) = \sum_{y \leq x} f_x(y)$
Probability distribution of X	$\mathcal{P}_X(A) := \sum_{x \in A} \Pr(X = x)$
Random vector	$\mathbf{X} = (X_1, \ldots, X_k)$
Joint distribution of $X_1, \ldots, X_k$	$\mathcal{P}_\mathbf{X}(\{\mathbf{x}\}) := \Pr(\mathbf{X} = \mathbf{x})$
Marginal distribution of X	$\Pr(X = x) = \sum_y \Pr((X, Y) = (x, y))$

- **Independence of events**: $\mathcal{P}(E_1, \ldots, E_k) = \mathcal{P}(E_1) \cdots \mathcal{P}(E_k)$
- **Independence of random variables** ($\mathbf{X}$): If for all $A_1, \ldots, A_k \subseteq \mathbb{R}$, $\Pr(X_1 \in A_1, \ldots, X_k \in A_k) = \Pr(X_1 \in A_1) \cdots \Pr(X_k \in A_k)$
- **Expectation**: $\mathrm{E}(X) = \sum_{x \in \mathbb{R}} x \cdot \Pr(X = x)$
- **Variance**: $\mathrm{Var}(X) = \mathrm{E}((X - E(X))^2) = E(X^2) - E(X)^2$
  - E is linear: $E(\alpha X + \beta Y) = \alpha E(X) + \beta E(Y)$
  - If independent: $\mathrm{E}(XY) = \mathrm{E}(X)\mathrm{E}(Y)$, $\mathrm{Var}(X + Y) = \mathrm{Var}(X) + \mathrm{Var}(Y)$
  - If pairwise independent: $\mathrm{Var}(\sum_{i=0}^n X_i) = \sum_{i=0}^n \mathrm{Var}(X_i)$
- Limits *loosely* bounding deviations of X from $E(X)$:

Markov's Inequality	$(X \overset{!}{\geq} 0, \forall a > 0)$	$\Pr(X \geq a) \leq \frac{\mathrm{E}(X)}{a}$
Chebychev's Inequality	$(\forall b > 0)$	$\Pr(\lvert X - \mathrm{E}(X)\rvert \geq b) \leq \frac{\mathrm{Var}(X)}{b^2}$

2.2 Concentration Inequalities

- **Concentration Inequality**: For $X = \sum_{i=1}^{n} X_i$ of pairwise independent X_i and $\mu = \mathrm{E}(X)$, we want to show concentration inequality $\Pr(|X - \mu| \geq \text{big}) \leq \text{small}$
  - *Weak Law of Large Numbers*: $\lim_{n\to\infty} \Pr\left(\frac{X}{n} - \mu \geq \varepsilon\right) = 0 \; \forall \varepsilon > 0$
    $X/n \approx \mu$ for $n \to \infty$, can be shown using Chebychev's inequality
- Limits *tightly* bounding deviation of X from μ:

Chernoff Bounds	$(X_i \in \{0,1\}$ ind., $0 \leq c \leq 1)$	$\Pr(X \geq (1+c)\mu) \leq \exp\left(-\frac{\mu c^2}{3}\right)$
		$\Pr(X \geq (1-c)\mu) \leq \exp\left(-\frac{\mu c^2}{2}\right)$
		$\Pr(\lvert X - \mu\rvert \geq c\mu) \leq 2\exp\left(-\frac{\mu c^2}{3}\right)$
Hoeffding Bounds	$(X_i \in \{0,1\}$ i.i.d., $0 \leq d \leq 1)$	$\Pr(X \geq \mu + dn) \leq \exp(-2nd^2)$
		$\Pr(X \geq \mu - dn) \leq \exp(-2nd^2)$
		$\Pr(\lvert X - \mu\rvert \geq dn) \leq 2\exp(-2nd^2)$

2.3 Entropy

- *Entropy*: Expected value of "information content" (expected number of bits) of the elementary event $\{\omega\}$, or measure of disarray
  - **Entropy** of prob. distr. $\mathcal{P}$: $H(\mathcal{P}) := \sum_{\omega\in\Omega} \mathcal{P}(\{\omega\}) \cdot \log_2 \frac{1}{\mathcal{P}(\{\omega\})}$
  - **Entropy** of X with finite range: $H(X) := \sum_{x\in\mathrm{rg}(X)} \Pr(X = x) \cdot \log_2 \frac{1}{\Pr(X=x)}$
- *Jensen's Inequality*: If $f : D \to \mathbb{R}$ convex function with $\mathrm{rg}(X) \subseteq D$, then $f(E(X)) \leq E(f(X))$
- *Log Sum Inequality*: $\sum_i p_i \log(p_i/q_i) \geq (\sum_i p_i) \log(\sum_i p_i / \sum_i q_i)$
- *Decision Trees*: Training data S over attributes $A \in \mathcal{A}$, $y \in \mathbb{Y}$ target value
  - Define $\mathcal{P}(\{y\})$ and $\mathcal{P}_{A=x}(\{y\})$ on $\mathbb{Y}$.
  - **Information Gain**: $G(S, A) := H(\mathcal{P}) - \sum_{x\in\mathbb{D}_A} \frac{|S_{A=x}|}{|S|} \cdot H(\mathcal{P}_{A=x})$
    ($\mathbb{D}_A$: domain of attribute A)
  - Pick attribute maximizing information gain and apply recursively.

2.4 Compression

- Let $\mathcal{P}$ prob. distr. over source alphabet Σ. Assume x_i in $\mathbf{x} \in \Sigma^n$ (denoted by $\mathcal{P}^n$) to be *i.i.d.*
- **Compression scheme**: $\Gamma = (\mathrm{com}_\Gamma, \mathrm{dec}_\Gamma)$ with $\mathrm{com}_\Gamma : \Sigma^* \to \{0,1\}^*$, $\mathrm{dec}_\Gamma : \{0,1\}^* \to \Sigma^*$
  - *Lossless*, if $\mathrm{dec}(\mathrm{com}(\mathbf{x})) = \mathbf{x}$
- **Compression rate** (of Γ): $\rho_\Gamma(n) := \max_{\mathbf{x}\in\Sigma^n} |\mathrm{com}(\mathbf{x})|/|\mathbf{x}|$
- **Loss rate**: $\lambda_{\Gamma,\mathcal{P}}(n) := \Pr_{\mathbf{x}\sim\mathcal{P}^n}(\mathbf{x} \neq \mathrm{dec}(\mathrm{com}(\mathbf{x})))$
  (prob. of lossy compression)
- There is *no lossless* compression with $\rho_\Gamma(n) < \log|\Sigma|$.
  (compromise between losslessness and compression rate needed)
- **Shannon's Source Coding Theorem**: For both lossy and lossless compression, optimal $\rho = H(\mathcal{P})$
  1. $\forall \varepsilon > 0$ ex. Γ_ε s.t $\lambda_{\Gamma_\varepsilon,\mathcal{P}}(n) \leq \varepsilon$ and $\lim_{n\to\infty} \rho_{\Gamma_\varepsilon}(n) = H(\mathcal{P})$.

- *Idea*: Define subsets of alphabet, such they have a high coverage of all words (length n) appearing, and for which compression is lossless. Then, overall compression is almost lossless, and the compression rate converges to the optimal compression rate $H(\mathcal{P})$.
- For every n choose $S_\varepsilon(n) \subseteq \Sigma^n$ s.t. $\mathcal{P}^n(S_\varepsilon(n)) \geq 1-\varepsilon$
- Define $\text{com}_\varepsilon(\Sigma^n) \subseteq \{0,1\}^{s_\varepsilon(n)}$ with $s_\varepsilon(n) := \lceil \log |S_\varepsilon(n)| \rceil$
- Define dec s.t. $\text{dec}(\text{com}(\mathbf{x})) = \mathbf{x}$ for all $\mathbf{x} \in S_\varepsilon(n)$.

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

3 Statistical Learning Theory

We aim to give statistical estimates on the generalization capabilities of an ML model.

3.1 The PAC Learning Framework

- Boolen Classification:
  - $\mathbb{X}$ instance space. Learn target function $f^* : \mathbb{X} \to \{0,1\}$
  - Data *i.i.d.* for *unkown* data generating distr. $\mathcal{D}$ on $\mathbb{X}$

Boolean Classification

Given training data $T = ((\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)) \in (\mathbb{X} \times \{0,1\})^m$ with $y_i = f^*(\mathbf{x}_i)$. Algorithm produces hyp. $h_T : \mathbb{X} \to \{0,1\}$ from hyp. space $\mathcal{H}$.

- **Generalization error/risk** of hyp. h: $\quad \text{err}_\mathcal{D}(h) := \Pr_{\mathbf{x}\sim\mathcal{D}}(h(\mathbf{x}) \neq f^*(\mathbf{x}))$
- **Empirical risk/training error** of hyp. h w.r.t. T: $\quad \text{err}_T(h) := \frac{1}{m}|\{i \in [m] \,|\, h(\mathbf{x}_i) \neq y_i\}|$
  - h *consistent* with T iff $err_T(h) = 0$

Probably Approximately Correct (PAC) Learning

Learning algorithm generating hyp. h is a *PAC learning algorithm*, if for all $\varepsilon, \delta > 0$ ex. $m = m(\varepsilon, \delta)$ s.t. for every $\mathcal{D}$ on $\mathbb{X}$ ($T \sim \mathcal{D}^m$ *i.i.d.*)

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

With sufficiently big training data, one can be arbitrarily certain, that the generalization error is arbitrarily small.

- *ERM*: An ERM algorithm returns h_T such that $\text{err}_T(h_T) = \min_{h\in\mathcal{H}} \text{err}_T(h)$
  - If problem is realizable, then ERM always returns a consistent hypothesis
  - *Risk of overfitting*, especially for high capacity $\mathcal{H}$
- **Regularization**: Counteract overfitting by considering "complexity cost" $\text{cost}(h)$, weighted by monotone (linear) function ρ
  - $h_T := \arg\min_{h\in\mathcal{H}}(\text{err}_T(h) + \rho(\text{cost}(h)))$

3.2 Sample Size Bounds

- An ERM algorithm tries to minimize generalization error by minimizing training error. Thus, ERM is good if *empirical risk is close to generalization error*.

- Assume *agnostic learning*: Target function isn't necessarily in $\mathcal{H}$ (*realizable*), therefore, find best possible hypothesis in $\mathcal{H}$.

Different Bounds

- **Simple Sample Size Bound**: $\mathcal{H}$ finite: Limit generalization error of hyp. consistent with T.
- **Uniform Convergence**: $\mathcal{H}$ finite: Limit difference of empirical risk and generalization error.
- **Agnostic PAC Learning SSB**: $\mathcal{H}$ finite: Limit ERM generalization error's deviation from optimum $h^* = \arg\min_{h\in\mathcal{H}} \mathrm{err}_{\mathcal{D}}(h)$.
- **Occam's Razor**: SSB depending on encoding length of hyp.
  - $\mathcal{H}$ not necessarily finite
  - Scheme Δ for describing h by strings in Σ^*
  - $|h|_\Delta$ length of the shortest description of h, used as measure of the "simplicity" of h

How to read: If bound for samples holds, then $\Pr_{T\sim\mathcal{D}^m}(\dots) > 1-\delta$ holds too.

	Req. sample size bound	$\Pr_{T\sim\mathcal{D}^m}(\dots) > 1-\delta$
Simple SSB	$m \geq \frac{1}{\varepsilon}\cdot\ln\left(\frac{\lvert\mathcal{H}\rvert}{\delta}\right)$	$\forall h\in\mathcal{H} : h \text{ cons. w/ } T \Rightarrow \mathrm{err}_{\mathcal{D}}(h)\leq\varepsilon$
Uniform Convergence	$m \geq \frac{1}{2\varepsilon^2}\cdot\log_2\left(\frac{2\lvert\mathcal{H}\rvert}{\delta}\right)$	$\forall h\in\mathcal{H} : \lvert\mathrm{err}_T(h)-\mathrm{err}_{\mathcal{D}}(h)\rvert\leq\varepsilon$
Agnostic PACL SSB	$m \geq \frac{2}{\varepsilon^2}\cdot\log_2\left(\frac{2\lvert\mathcal{H}\rvert}{\delta}\right)$	$\lvert\mathrm{err}_{\mathcal{D}}(h_T)-\mathrm{err}_{\mathcal{D}}(h^*)\rvert\leq\varepsilon$
Occam's Razor	$m \geq \frac{1}{\varepsilon}\left(n\cdot\ln\lvert\Sigma\rvert+\ln\left(\frac{2}{\delta}\right)\right)$	$\forall h : (\lvert h\rvert_\Delta \leq n \wedge \mathrm{err}_T(h)=0)\Rightarrow\mathrm{err}_{\mathcal{D}}(h)\leq\varepsilon$

- *Example*: 10d linear separator, $\mathbf{w}, b$ as 64-bit floats: $|\mathcal{H}| = 2^{11\cdot 64}$; with $\mathrm{err}_{\mathcal{D}}$ bound $\varepsilon := 0.1$ and confidence $\delta := 0.05$ (with prob. $> 95\%$ generalization error is $\geq 10\%$):

$$m \geq \frac{1}{\varepsilon=0.1}\cdot\ln\left(\frac{|\mathcal{H}|=2^{11\cdot 64}}{\delta=0.05}\right) = 10\cdot\ln(20\cdot 2^{704}) \approx 4910$$

- **VC dimension**: Quantify "complexity" of $\mathcal{H}$
  - $Y\subseteq\mathbb{X}$ is *shattered* by $\mathcal{H}$, if for every function $g : Y\to\{0,1\}$ there is $g = h|_Y$ for some $h\in\mathcal{H}$
  - $\mathrm{VC}(\mathcal{H})$: size of largest set shattered by $\mathcal{H}$ (might be ∞)
  - * Recipe for proof:
    1) Show $VC(\mathcal{H})\geq d$, by demonstrating, that every function $g : Y\to\{0,1\}$ with some *chosen* $Y\subseteq\mathbb{X}$ and $|Y| = d$ is representable.
    2) Show $VC(\mathcal{H}) < d{+}1$: Assume otherwise and argument, that for *any* $Y\subseteq\mathbb{X}$ with $|Y| = d+1$, there is no suitable $h\in\mathcal{H}$.
- **Uniform Convergence for VC dimension**: If $d = \mathrm{VC}(\mathcal{H})$ finite, uniform convergence condition is fulfilled for some $c > 0$ by

$$m \geq \frac{c}{\varepsilon^2}\cdot\left(d+\log_2\left(\frac{1}{\delta}\right)\right)$$

4 Multiplicative Weight Updates

4.1 The MWU Algorithm

- *Setup*:
  - $I = [n]$: set of **experts**
  - J : set of **events**
  - $L \in \mathbb{R}^{I \times J}$: loss matrix ($L_{i,j} \in [0,1]$ is loss when following expert i while event j happened)
- *Randomized Strategy*: Draw expert $i^{(t)}$ from prob. distr. $\mathcal{D}^{(t)}$. If event $j^{(t)}$ happens, loss will be $L_{i,j}$. $L^{(t)}$ is expected loss in round t.

$$\mathcal{D}^{(t)}(\{i\}) := p_i^{(t)} := \frac{w_i^{(t)}}{\sum_{i' \in I} w_{i'}^{(t)}}, \qquad L^{(t)} = \sum_{i \in I} p_i^{(t)} L_{i,j^{(t)}}$$

- *Algorithm*: Init $w_i^{(1)} := 1$, and update $w_i^{(t+1)} := (1-\alpha)^{L_{i,j^{(t)}}} w_i^{(t)}$.
- Randomized strategy in weighted majority setting beats naive one by almost a factor of 2 in expectation.
- Example: *Weighted Majority* (naive)
  - *Setup*:
    - $p^{(t)}$: event in round t (in $\{0,1\}$)
    - $a_i^{(t)}$: advice of expert $i \in I$ (in $\{0,1\}$)
    - $d^{(t)}$: our decision (in $\{0,1\}$)
    - $l^{(t)}$: our cumulated loss after t rounds with $l^{(t)} := \sum_{s=1}^{t} |d^{(s)} - p^{(s)}|$
    - $J = \{0,1\}^n \times \{0,1\}$, $L[i,(\mathbf{a},p)] := |a_i - p|$ ($L \in \{0,1\}^{n \times 2^{n+1}}$)
  - *Algorithm*: Base our decision on weighted majority of experts. For every i, decrease weight w_i of expert i by factor $(1-\alpha)$, if expert was wrong.
  - Guarantees our losses to be just a bit more than twice the losses of the best expert.

4.2 Boosting Weak Learning Algorithms

- **Strong Learner** (= PAC): $\forall \varepsilon, \delta > 0$ ex. $m = m(\varepsilon, \delta)$ s.t. $\forall \mathcal{D}$: $\Pr_{T \sim \mathcal{D}^m}(\mathrm{err}_{\mathcal{D}}(h_T) \leq \varepsilon) > 1 - \delta$
- **γ-Weak Learner** ($0 \leq \gamma < 0.5$): $\forall \delta > 0$ ex. $m = m(\delta)$ s.t. $\forall \mathcal{D}$: $\Pr_{T \sim \mathcal{D}^m}(\mathrm{err}_{\mathcal{D}}(h_T) \leq \gamma) > 1 - \delta$
- *Goal*: Reduce error of a weak learner, turning it into a strong learner.

Boosting Problem

Given γ-Weak Learner $\mathcal{L}$, training data $T := ((x_1, y_1), \ldots, (x_n, y_n)) \in (\mathbb{X} \times \{0,1\}^n)$, and error parameter $\varepsilon > 0$.
Compute hypothesis h with $\mathrm{err}_T(h) < \varepsilon$.

- Choosing $\varepsilon < \frac{1}{n}$ forces h to be consistent with T.
- *Algorithm*:
  - Apply weak learner $\mathcal{L}$ repeatedly to examples drawn from different distributions $\mathcal{D}^{(t)}$ on $X := \{x_1, \ldots, x_n\}$. Adapt distributions, reducing weight of correctly classified examples using MWU.

- Run $\mathcal{L}$ on $\mathcal{D}$, until it returns a *good* hypothesis (generalization error less that γ). Prob. for that is $1 - \delta_0 = 1 - \frac{1}{10}$.
- *Setup*:
  - Number of examples $\mathcal{L}$ needs for error bound γ: $m_0 = m(\delta_0) = m(\frac{1}{10})$
  - Training data as experts: $I := [n]$
  - Hypotheses as events: $J = \{h \mid h \text{ returned by } \mathcal{L}\}$
  - Loss matrix: $L_{i,j} = 1$, if $j(x_i) = y_i$, 0 otherwise ($\alpha := 0.5 - \gamma$)
- Iterate for $t := \frac{2}{\alpha^2} \ln \frac{1}{\varepsilon}$ rounds. Final hypothesis returns 0 or 1 by majority vote over all hypotheses:

$$h(x) := \begin{cases} 1 & \text{if } |\{s \leq t \mid j^{(s)}(x) = 1\}| \geq t/2 \\ 0 & \text{otherwise} \end{cases}$$

  Then, $\text{err}_T(h) < \varepsilon$.

4.3 Bandit Learning

- Choose slot machine $i \in [n]$, minimizing *regret* (difference between cumulative payoffs of our strategy and the best machine). We cannot observe the reward, that the other actions would have given.
- *Adversarial* Setting: Payoff of each machine in each round is manipulated to maximize our regret
- *Setup*:
  - γ : tradeoff between **exploration** ($\gamma = 1$) and **exploitation** ($\gamma = 0$)
  - $0 \leq q_a^{(s)} \leq 1$: *reward* for choosing action a in round s
  - $Q := (q_a^{(s)})_{a \in [n], s \geq 1}$: *payoff matrix*
  - $r(\mathbf{a}) := q_{max}^{(t)} - q(\mathbf{a})$: *regret* of sequence of actions $\mathbf{a}$ with cumulative reward $q(\mathbf{a}) := \sum_{s=1}^{t} q_{a^{(s)}}^{(s)}$, and maximal single-action reward after t rounds $q_{max}^{(t)} := \max_a \sum_{s=1}^{t} q_a^{(s)}$
- *Idea*: Pick in each round t an action $a^{(t)}$ only depending on previous $a^{(s)}$ and $q_{a^{(s)}}^{(s)}$.
- **EXP3**
  1. Update distribution:

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

  2. Draw a randomly from $\mathcal{D}$, observe reward $q \leftarrow q_a$.
  3. Update weight $w_a \leftarrow w_a \cdot \exp(\frac{\gamma q}{n p_a})$.

5 High-Dimensional Data

5.1 The Strange Geometry of High-Dimensional Spaces

Let $X \subseteq \mathbb{R}^l$ measurable set, $B^l := \{\mathbf{x} \in \mathbb{R}^l \mid ||\mathbf{x}|| \leq 1\}$ unit ball.

- **Volume** equations:
  - $\text{vol}(cX) = c^l \text{vol}(X)$

- $\text{vol}((1-\varepsilon)X)/\text{vol}(X) \leq e^{-\varepsilon l} \overset{l\to\infty}{\longrightarrow} 0$
  (volume is near the surface)
- $\lim_{l\to\infty} \text{vol}(B^l) = 0$
- At least $(1-\frac{2}{c}e^{-c^2/2})$ of the volume of B^l has distance $\leq c/\sqrt{l-1}$ from the equator $x_1 = 0$
  (volume is concentrated near the equator)
- Choose $\mathbf{x}$ from cube $Q^l = [0,1]^l$: $E(||\mathbf{x}||^2) = l/3$ and $\Pr(||\mathbf{x}||^2 \leq 1) = o(2^{-l})$ (prob. that vector lies in B^l)

5.2 Dimension Reduction by Random Projections

We aim to map $\mathbb{R}^l$ to $\mathbb{R}^k$ with $k \approx \log n$.

- **Spherical Gaussian Distribution**: $p(\mathbf{x}) = \frac{1}{(2\pi)^{l/2}\cdot\sigma^l} \cdot \exp\left(-\frac{||\mathbf{x}-\boldsymbol{\mu}||^2}{2\sigma^2}\right)$
  - Same as choosing coordinates x_i independently from 1d $\mathcal{N}(\mu_i, \sigma)$
- **Gaussian Annulus Theorem**: For $b \leq \sqrt{l}$ and $\mathbf{x}$ distributed spherically with $\boldsymbol{\mu} = \mathbf{0}, \sigma^2 = 1$:

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

  → "Probability mass of high-dim. spherical Gaussians is concentrated in a thin annulus of radius $\sigma\sqrt{l}$ around the mean"

- **Random Projection**: For $k \leq l$, draw vectors $\mathbf{u}_i \in \mathbb{R}^l$ independently from spherical Gaussian distr. $\mathcal{N}(\mathbf{0}, 1)$. Apply basis change $\mathbf{x} \to U\mathbf{x}$ (*"random projection"*) with

$$U := \frac{1}{\sqrt{k}} \begin{pmatrix} \mathbf{u}_1^T \\ \vdots \\ \mathbf{u}_k^T \end{pmatrix} \in \mathbb{R}^{k\times l}$$

  - **Random Projection Theorem**: For all $\varepsilon > 0, \mathbf{x} \in \mathbb{R}^l$:

$$\Pr\left(\left|||U\mathbf{x}|| - ||\mathbf{x}||\right| > \varepsilon||\mathbf{x}||\right) \leq 3e^{-c\varepsilon^2 k}$$

    → "With low prob. the length of the projection deviates from the original vector length by more than an insignificant amount."

  - **Johnson-Lindenstrauss Lemma**: Bound for quality preservation w.r.t. pairwise distance of points in set $X \subseteq \mathbb{R}^l$:

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

    with k depending on $|X| = n$, and ε.

    → "With high. prob. distances between points and distances between their projections are similar."

    → "Randomly chosen linear mapping will work with high prob."

5.3 Background from LA: Eigenvalues and Eigenvectors

- λ *eigenvalue* of A, $\mathbf{u}$ *eigenvector* associated with λ: $A\mathbf{u} = \lambda\mathbf{u}$
  - $\det(A - xI) = \prod_{i=1}^n (\lambda_i - x)$ $(*)$ resulting in *spectrum* $\{\lambda_i, \ldots, \lambda_n\}$, where the λ_i are the zeroes of the polynomial
  - *Algebraic Multiplicity*: amount of times λ_i appears in product $(*)$
  - **Spectral Radius** $\rho(A)$: max. absolute value of eigenvalues
- Matrix properties:

- $A \in \mathbb{C}^{n\times n}$ *diagonalizable*, if there are nonsingular (invertible) matrix U and diagonal matrix Λ s.t. $U^{-1}AU = \Lambda$
- *Orthonormal basis*: $\mathbf{u}_1, \ldots, \mathbf{u}_n \in \mathbb{R}^n$ with $||\mathbf{u}_i|| = 1$ and $\langle \mathbf{u}_i, \mathbf{u}_j \rangle = 0$
- $A \in \mathbb{R}^{m\times n}$ *orthonormal*: $A^T A = I$
- $A \in \mathbb{R}^{nxn}$ *orthogonal*: orthonormal + square OR columns of A form orthonormal basis of $\mathbb{R}^n$
- $A \in \mathbb{R}^{nxn}$ *irreducible* if directed graph $G_A = ([n], \{(i,j) \mid A_{i,j} \neq 0\})$ is strongly connected

- **Spectral Decomposition Theorem**: Let $A \in \mathbb{R}^{nxn}$ symmetric, then ex. U (matrix of eigenvectors) s.t.

$$A = U\Lambda U^T \Leftrightarrow U^T AU = \Lambda = \text{diag}(\lambda_1, \ldots, \lambda_m)$$

Let $A \in \mathbb{R}^{n\times n}$ irreducible and non-negative.

- **Perron-Frobenius Theorem**: $\rho(A) = \lambda_1$ is eigenvalue with algMult 1, and there are unique all-positive right/left unit eigenvectors $\mathbf{u}, \mathbf{v}$ (*Perron vectors* of A)

$$A\mathbf{u} = \rho\mathbf{u} \qquad \mathbf{v}^T A = \rho\mathbf{v}^T (\Leftrightarrow A^T\mathbf{v} = \rho\mathbf{v})$$

- **Limit Theorem for Non-Negative Matrices**: Let $\mathbf{u}, \mathbf{v}$, right/left Perron vectors of A:

$$\lim_{k\to\infty} \frac{1}{k} \sum_{i=1}^{k} \frac{A^i}{\rho(A)^i} = \frac{1}{\langle \mathbf{u}, \mathbf{v} \rangle} \cdot \mathbf{u}\mathbf{v}^T$$

5.4 Power Iteration

- Let A diagonalizable with spectrum $|\lambda_1| \geq \cdots \geq |\lambda_n|$, $\lambda_1 = \rho(A) \geq 0$ and $\lambda_1 > |\lambda_2|$.
  → λ_1 has to be positive and unique, otherwise, PI doesn't converge
- If $\langle \mathbf{x}, \mathbf{u}_1 \rangle \neq 0$, then $A^k\mathbf{x}$ converges to $c \cdot \mathbf{u}_1$.
- *Algorithm*: Pick $\mathbf{x}$ randomly, repeatedly set $x \leftarrow A\mathbf{x}/||A\mathbf{x}||$ until convergence.

5.5 Principal Component Analysis

We aim to achieve dimensionality reduction via *feature selection* or *feature extraction* by minimizing error or maximizing variance.

- Centering data A ($\mathbf{a}'_i = \mathbf{a}_i - \boldsymbol{\mu}$), then variance is $\sum_i ||\mathbf{a}'_i||^2$
- **PCA Transformation** of A is orthogonal $U = (\mathbf{u}_1 \cdots \mathbf{u}_l) \in \mathbb{R}^{l\times l}$ s.t.

$$\mathbf{u}_j = \underset{\substack{\mathbf{x}\in\mathbb{R}^l, ||\mathbf{x}||=1 \\ \mathbf{x}\perp\mathbf{u}_1,\ldots,\mathbf{u}_{j-1}}}{\arg\max} \sum_{i=1}^{n} \langle \mathbf{a}_i, \mathbf{x} \rangle^2 = ||A\mathbf{x}||^2$$

  → Maximizes variance $\sum_{i=1}^{n} \langle \mathbf{a}_i, \mathbf{x} \rangle^2$
  - Recipe: Iteratively choose orthonormal basis vectors $\mathbf{u}_j$ maximizing projection lengths $\langle \mathbf{a}_i, \mathbf{u}_j \rangle$
  - Lines $\mathbb{P}_i := \text{span}(\mathbf{u_i})$ are *principal components* of A
  - $\mathbb{U}_k = \text{span}(\mathbf{u}_1, \ldots, \mathbf{u}_k)$ is best-fit k-dim. subspace w.r.t. A (= max. variance)
- *PCA via Spectral Decomposition of Covariance Matrix*
  - **Covariance Matrix**: $C := A^T A$ with $A \in \mathbb{R}^{n\times l}$ centered. C is symm. pos. semi-definite
    → Eigenvalues of C are nonneg. real numbers
  - If $\lambda_1 \geq \cdots \geq \lambda_l$ eigenvalues of C, then the matrix of the corresponding eigenvectors $U = (\mathbf{u}_1 \cdots \mathbf{u}_l)$ is a PCA transformation of A.

5.6 Spectral Clustering

- *Goal*: Maximize *dissimilarity* between clusters
- With similarity matrix $S \in \mathbb{R}^{n\times n}$ defined by $S_{ij} := s(i,j)$, where $s : X \times X \to \mathbb{R}_{\geq 0}$ *similarity measure*:

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

→ mincut favors very small/large clusters; balcut computationally hard to minimize

- *Algorithm*:
  - *Input*: $k \in \mathbb{N}$, symm. similarity matrix $S \in \mathbb{R}^{m\times m}_{\geq 0}$
  - Compute *Laplacian* $L = D - S$ with $D = \text{diag}\left((\sum_{j=1}^{m} S_{ij})_{i=1}^{m}\right)$.
  - Find k smallest eigenvalues of L with eigenvectors $\mathbf{u}_1,\dots,\mathbf{u}_k$.
  - Perform k-Means on *rows* of $U = (\mathbf{u}_1 \cdots \mathbf{u}_k)$, yielding clusters $C^1,\dots,C^k$.
  - Return $D_1,\dots,D_k$ with $D_i := \{x_j \mid (u_{j1},\dots,u_{jk}) \in C^i\}$.
- *Properties*:
  - For $\mathbf{v} \in \mathbb{R}^m$: $\mathbf{v}^T L\mathbf{v} = \frac{1}{2}\sum_{i,j} S_{ij}(v_i - v_j)^2$
  - L is symm. pos. semi-definite ($\Rightarrow U^T LU = \Lambda$)
  - $U \in \mathbb{R}^{n\times k}$ is *partition matrix*, if it has k distinct rows. For an orthonormal partition matrix U with
$$U_{ip} = \begin{cases} 1/\sqrt{|C^p|} & i \in C^p \\ 0 & \text{else} \end{cases}$$
    then $\text{balcut}(C^1,\dots,C^k) = \text{trace}(U^T LU)$. (trace: sum of diagonal entries)
  - Finding such U is hard; Instead, find eigenvector matrix to k smallest eigenvalues of L.

6 Random Walks and Markov Chains

Monte Carlo (MC) methods aim to estimate values through sampling.

6.1 Estimation Through Sampling

- *Basic MC*: Estimate μ through $\hat{\mu} = \frac{1}{m}\sum_{i=1}^{m} f(\omega_i)$ with $m = m(\varepsilon,\delta)$ for independent samples $\omega_i \in \Omega$
  - For $b := \max\{|f(\omega)| \mid \omega \in \Omega\}$, $m \geq \frac{b\ln\left(\frac{2}{\delta}\right)}{\varepsilon^2}$: $\Pr(|\hat{\mu} - \mu| \leq \varepsilon) \geq 1 - \delta$

  → Estimation is good with high probability
- **Rejection Sampling**: We want to sample from $(\Omega, \mathcal{P})$ with $\mathcal{P}(\omega) = \mathcal{D}(\omega)/Z$, known up to Z.
  - Assume known function of upper bounds $\Delta : \Omega \to \mathbb{R}$ with $\mathcal{D}(\omega) \leq \Delta(\omega)$. Assume we sample from Ω with *proposal distribution* $\Pi(\omega) = \Delta(\omega)/\sum_{\omega'\in\Omega}\Delta(\omega')$. Π is uniform distr., if Δ const.
  - *Recipe*: Sample $\omega \in \Omega$ from Π. With prob. $\mathcal{D}(\omega)/\Delta(\omega)$, accept and return ω, otherwise reject.
  - Accepts in any iteration with prob. $a = Z/\sum_{\omega\in\Omega}\Delta(\omega)$

  - RS eventually accepts with prob. 1, and accepts at $\leq k$ iterations with prob. $1-(1-a)^k (\geq 1-e^{-ak})$
  - If accepts, ω is returned with prob. $\mathcal{P}(\omega)$
  - Example *Graph Matchings* (set of edges where no edges share (end-)vertices): $\Omega := 2^E, \mathcal{D} := \mathbb{1}_{\mathcal{M}}$ ($\mathcal{M}$ set of all matchings on graph), $\Delta = \mathbb{1}_{2^E}$

- **Karp-Luby Algorithm**: Ex. randomized *polynomial time* alg. that given a satisfiable Boolean formula ϕ in DNF ($\bigvee\bigwedge$), returns satisfying assignment sampled uniformly at random from the set of all satisfying assignments

  → RS: Let $\phi := \bigvee_i \gamma_i$. Choose $i \in [m]$ uniformly at random. Choose $\alpha \in \{0,1\}^n$ that satisfies γ_i uniformly at random. Reject α iff α satisfies any γ_j with $j < i$.

6.2 Random Walks and Markov Chains

- **Random walk**: Sequence of vertices on directed graph, where at each step, next edge is chosen randomly starting from initial vertex

- **Markov Chain** (MC): Next state in sequence in randomly chosen based on current state
  - n-state MC represented by $\mathcal{Q}$ represented by (stochastic) transition matrix $Q \in [0,1]^{n\times n}$ with row sums $= 1$, or by directed weighted graph G_Q with $w(i,j) = Q_{ij}$
  - MC is *connected* if G_Q is strongly connected / iff Q is irreducible
  - MC is *aperiodic* if the greatest common divisor of the cycle lengths on G_Q is 1.
  - MC is *ergodic* if it is connected and aperiodic.

- **Probability Vector** (PV): $\mathbf{p} \in \mathbb{R}^{1\times n}$, non-negative, sum $= 1$; $\mathbf{p}_t = \mathbf{p}_0 Q^t$

- Depending on initial PV $\mathbf{p}_0$: *Average distribution* $\mathbf{a}_t := \frac{1}{t}\sum_{i=0}^{t} \mathbf{p}_i$

Stationary Distribution (SD): Let $\mathcal{Q}$ connected.

- *Fundamental Theorem of MC*: Ex. unique PV $\boldsymbol{\pi}$ s.t. $\boldsymbol{\pi}Q = \boldsymbol{\pi}$. Also, $\boldsymbol{\pi} = \lim_{t\to\infty} \mathbf{a}_t$.
  - $\boldsymbol{\pi}$ is left Perron vector of Q with $\rho = 1$; computation of SD is then essentially power iteration

- If $p_i Q_{ij} = p_j Q_{ji}$ for $\forall i,j \in [n]$, then $\mathbf{p} = \boldsymbol{\pi}$

- If $\mathcal{Q}$ ergodic, then $\lim_{t\to\infty} \mathbf{p}_t = \boldsymbol{\pi}$ for every initial $\mathbf{p}_0$.

- For $0 < a < 1$, $Q_\alpha := \alpha Q + (1-\alpha)I$ is *ergodic*, and has the same stationary distr. as $\mathcal{Q}$
  - aperiodic, since Q_α introduces loops at every vertex of length 1

6.3 Markov Chain Monte Carlo

- *Problem Description*: Estimate $\mu = E(f)$ of random var. f defined on prob. space $(\Omega, \mathcal{P})$ by sampling ω_i through $\hat{\mu} = \frac{1}{m}\sum_{i=0}^{m} f(\omega_i)$. Difficult from large $(\Omega, \mathcal{P})$.

- **MCMC**: Set up ergodic $\mathcal{Q}$ with state space Ω and stationary distr. $\mathcal{P}$ (known up to Z). Simulate MC, until close to stationary distr. Take current state as sample from $(\Omega, \mathcal{P})$.
  Now, we want to approximate $\mathcal{P}$.
  - We define $\mathcal{Q}_{\mathcal{P}_0,t}(\omega_t)$ to be the sum of probabilities over all possible paths leading to ω_t, starting from ω_0:

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

$\mathcal{Q}_{\mathcal{P}_0,t}$: distr. of $\mathcal{Q}$ after t steps with initial distr. $\mathcal{P}_0$; $\mathcal{Q}(\omega_{i-1},\omega_i)$: transition prob. from ω_{i-1} to ω_i

- *Total variation distance* for prob. distr. $\mathcal{P},\mathcal{P}'$: $\text{dist}_{\text{TV}}(\mathcal{P},\mathcal{P}') = \frac{1}{2}\sum_{\omega\in\Omega}|\mathcal{P}(\omega)-\mathcal{P}'(\omega)|$
- $\mathcal{Q}_{\omega_0,t}$ eventually approximates $\mathcal{P}$ well: $\lim_{t\to\infty}\text{dist}_{\text{TV}}(\mathcal{Q}_{\omega_0,t},\mathcal{P}) = 0$

  $\mathcal{Q}_{\omega_0,t}$: $\mathcal{Q}_{\mathcal{P}_0,t}$ with 1-point distr. $\mathcal{P}_0(\omega_0) = 1$
- *Mixing Time*: $T_{\mathcal{Q}}(\varepsilon) = \min_{t\in\mathbb{N}}\{t\in\mathbb{N}\,|\,\forall\omega_0\in\Omega : \text{dist}_{\text{TV}}(\mathcal{Q}_{\omega_0,t},\mathcal{P})\le\varepsilon\}$

- Approximation of mean is upper bounded by accuracy of estimated $\mathcal{P}$:

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

- *MCMC Theorem*: Assuming some suitable ε,δ,m,t and $|f(\omega)|<b$, then $\Pr(|\hat{\mu}-\mu|\le\varepsilon)\ge 1-\delta$.
- **Metropolis-Hastings Sampling**: Create undirected connected graph $G=(\Omega,E)$ with vertices having max. degree of Δ

$$q_{\omega,\omega'} := \begin{cases} \frac{1}{\Delta+1} & \omega\omega'\in E(G)\wedge\mathcal{D}(\omega')\ge\mathcal{D}(\omega) \\ \frac{1}{\Delta+1}\cdot\frac{\mathcal{D}(\omega')}{\mathcal{D}(\omega)} & \omega\omega'\in E(G)\wedge\mathcal{D}(\omega')<\mathcal{D}(\omega) \\ 1-\sum_{\omega''\in N_G(\omega)} q_{\omega,\omega''} & \omega=\omega' \\ 0 & \text{otherwise} \end{cases}$$

  results from *transition algorithm* on current state ω:
  - With prob. $1-|N(\omega)|/(\Delta+1)$, stay in ω. Otherwise, choose $\omega'\in N_G(\omega)$ randomly. If $\mathcal{D}(\omega')\ge\mathcal{D}(\omega)$ move to ω', or if $\mathcal{D}(\omega')<\mathcal{D}(\omega)$ with prob $\mathcal{D}(\omega')/\mathcal{D}(\omega)$ move to ω', else stay in ω.

  $\rightarrow$ Essentially RS on neighbors of ω

  $\rightarrow$ $\mathcal{Q}$ is ergodic with stationary distr. $\mathcal{P}$
- *Sketch for graph matchings*: Graph with $V(G)=\mathcal{M}\subseteq 2^{E(G)}$, connected through $\subset$ relation

6.4 Coupling of Markov Chains

- **Coupling**: Run MCs $\mathcal{C} := (\mathcal{Q}_1,\mathcal{Q}_2)$ simultaneously on $\Omega\times\Omega$. Coupling of MCs is a general technique for bounding the mixing time of an MC.
  - *Coupling conditions* state, that $\mathcal{Q}_1,\mathcal{Q}_2$ are copies of $\mathcal{Q}$, but are not independent

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

  - *Coupling Lemma*: If $\mathcal{Q}_1,\mathcal{Q}_2$ are coupled s.t. with high prob. they are in the same state, they must be close to the stationary distr.

7 Highly Distributed Systems

7.1 Distributed Systems and Map-Reduce Environment

Basic setup:

- Computing clusters consisting of cluster nodes, that are arranged in racks of 16-24 nodes.
- *Challenges*: Distribute computation, write distributed programs, handle hardware failure.

- *Principles*: Data stored close to computation, data stored multiple times (for reliability)
  - Files split into chunks of 64 MB, replicated 3 times, stored in different racks.
- *Master node*: Stores metadata about chunks

Map-Reduce: User provides MAP and REDUCE functions (for data analysis tasks on DFS)

- MAP: Takes as input single key-value pair, and emits set of key-value pairs.
- SHUFFLE: Group key-value pairs by key.
- REDUCE: Aggregate groups of same key, emitting set of key-value pairs.
- Environment handles data partitioning and task assignment to nodes, executing shuffle, handling communication and failures.
- Master node handles task scheduling and failure detection.
- Might pre-aggregate values in Map to save communication cost

7.2 Map-Reduce Algorithms

Matrix-Vector Multiplication Av (A as (i, j, a_{ij}), v as (i, v_i))

MAP on (i, j, a_{ij}) emit $(i, a_{ij}v_j)$
REDUCE on $(i, vals)$ emit $(i, \sum_{v \in vals} v)$

- If **v** is big: Split **v** into segments, split A into *vertical* stripes accordingly.

Relational Data: Store row (tuple) t of schema $R(A_1, \ldots, A_k)$ of a relation $\mathcal{R}$ with attributes A_i as (R, t)

Projection $\mathcal{Q} = \pi_{A,C}(\mathcal{R})$ (tuple t in $R(A, B, C)$ as (R, t))

MAP on $(R, (a, b, c))$ emit $((a, c), 1)$
REDUCE on $((a, c), vals)$ emit $(Q, (a, c))$ (removes duplicates)

Intersection $\mathcal{Q} = \mathcal{R} \cap \mathcal{S}$ (tuple t in R/S as $(R/S, t)$)

MAP on (X, t) emit (t, X) $(X = R \vee X = S)$
REDUCE on $(t, vals)$ emit (Q, t) if $R, S \in vals$

Natural Join $\mathcal{Q} = \mathcal{R} \bowtie \mathcal{S}$ (schemas $R(A, B)$ and $S(B, C)$)

MAP on $(R, (a, b))$ emit $(b, (R, a))$
on $(S, (b, c))$ emit $(b, (S, c))$
REDUCE on $(b, vals)$ emit $(Q, (a, b, c))$ for all $(R, a), (S, c) \in vals$

Grouping & Aggregation (scheme $R(A, B, C)$, group by A, avg. by C)

MAP on $(R, (a, b, c))$ emit (a, c)
REDUCE on $(a, vals)$, compute average c^* of $c \in vals$ and emit $(Q, (a, c^*))$

Matrix Multiplication $C = A \cdot B$ (A as $(A, (i, j, A_{ij})$, B as $(B, (j, k, B_{jk}))$)

Map 1	on $(A, (i, j, a))$ emit $(j, (A, i, a))$ on $(B, (j, k, b))$ emit $(j, (B, k, b))$
Reduce 1	on $(j, vals)$ emit $((i, k), ab)$ for all $(A, i, a), (B, k, b) \in vals$ with $ab \neq 0$
Map 2	on $((i, k), x)$ emit identity
Reduce 2	on $((i, k), vals)$ emit $(C, (i, k, \sum vals))$

Matrix Multiplication $C = A \cdot B$ (one round)

Map	on $(A, (i, j, a))$ emit $((i, k), (A, j, a))$ $\forall k \in [n]$ on $(B, (j, k, b))$ emit $((i, k), (B, j, b))$ $\forall i \in [l]$
Reduce	on $((i, k), vals)$ compute $x = \sum_j ab$ for $(A, j, a), (B, j, b) \in vals$ and emit $(C, (i, k, x))$

7.3 Analysis of Map-Reduce Algorithms

Cost measures:

- **Wall-clock time**: total computation time; too complicated to analyze (system-dependent)
- **Number of rounds**: important, but not sufficient
- **Communication cost**: sum of all input sizes; comm. time dominates comp. time in nodes
- **Replication rate**: number of all key-value pairs divided by input size; only relevant for one-round MR-processes
- **Maximum load**: max length of values list in Reduce input ("Reducer Size")

7.4 Analysis of Matrix Multiplication

- Assume (small) prob. p, q for an entry of $A \in \mathbb{R}^{l \times m}, B \in \mathbb{R}^{m \times n}$ to be non-zero.

	Communication cost	Maximum load
Two-round	$\underbrace{2plm + 2qmn}_{\text{Map, Reduce 1}} + \underbrace{2pqlmn}_{\text{Map, Reduce 2}}$	1. $(pl + qn)$, 2. pqm
One-round	$\underbrace{plm + qmn}_{\text{Map}} + \underbrace{(p + q)lmn}_{\text{Reduce}}$	$(p + q)m$

→ For sparse matrices, the communication cost of the two-round algorithm is likely to be much lower. The maximal load is comparable.

- **Generalized single-round algorithm**: Split matrices into s stripes (vertical in A, horizontal in B), and use pairs of indices of the stripes as keys for the reducer; Reduces replication rate and communication cost.
- Assume $h : [n] \to [n]$ assigns each row/column index its stripe index.

Matrix Multiplication $C = A \cdot B$ (*generalized* one round)

Map	on $(A, (i, j, a))$ emit $((h(i), u), (A, i, j, a))$ for $u \in [s]$ on $(B, (j, k, b))$ emit $((t, h(k)), (B, j, k, b))$ for $t \in [s]$
Reduce	on $((t, u), vals)$ compute $c_{ik} = \sum_{(A,i,j,a),(B,j,k,b) \in vals} ab$ $\forall i \in h^{-1}(t), k \in h^{-1}(u)$, and emit $(C, (i, k, c_{ik}))$

	Communication cost	Maximum load
-	$\mathcal{O}(sn^2)$	$2n^2/s$
given p, q	$\mathcal{O}(s(p+q)n^2)$	$(p+q)n^2/s$

7.5 Multiway Joins in Map-Reduce

- **Hypercube Algorithm**: Correctly computes multi-natural join $\mathcal{Q} = \mathcal{R}_1 \bowtie \ldots \bowtie \mathcal{R}_l$

Hypercube Algorithm $\mathcal{Q} = \mathcal{R}_1 \bowtie \ldots \bowtie \mathcal{R}_l$

MAP on $(R_i, (a_i, \ldots, a_{k_i}))$ emit $((p_1, \ldots, p_k), (R_i, (a_1, \ldots, a_{k_i})))$
- s_i *share* of attribute A_i
- $\prod_i^k s_i = s$ number of reducers
- $h_i : V_i \to [s_i]$ hash function on domain of attribute A_i
- $p_j = h_j(a'_j)$ for all $j \in [k], j' \in [k_i]$ s.t. $A_{ij'} = A_j$
- for A_j not appearing in $(a_i, \ldots, a_{k_i})$: choose all possible vals for p_j

REDUCE on $(\bar{p}, vals)$ compute $\mathcal{Q}(\bar{p}) = \mathcal{R}_1(\bar{p}) \bowtie \ldots \bowtie \mathcal{R}_l(\bar{p})$, emit all (Q, t) for $t \in \mathcal{Q}(\bar{p})$
- $\mathcal{R}_i(\bar{p}) := \{t \mid (R_i, t) \in vals\}$

* Example:
- On $(R_1, (67, 48))$ with $R_1(A_1, A_2)$ MAP emits $((3, 3, 1), (R_1, (67, 48)))$ and $((3, 3, 2), (R_1, (67, 48)))$ (having $h_1(67) = 3, h_2(48) = 3, \text{rg}(h_3) = \{1, 2\}$)
- On $(R_1, x, y), (R_2, y, z), (R_3, z, x)$ in $vals$, REDUCER emits $(Q, (x, y, z))$

- Assume $Idx(i)$ set of indices of attributes of $\mathcal{R}_i$, $m_i := |\mathcal{R}_i|$, which are large compared to number of reducers, and hash functions chosen randomly.
  - *Replication rate*: $\frac{\sum_{i \in [l]} \left(m_i \cdot \prod_{j \in [k] \setminus Idx(i)} s_j \right)}{\sum_{i \in [l]} m_i}$
  - *Expected load*: $\sum_{i \in [l]} \frac{m_i}{\prod_{j \in Idx(i)} s_j}$
  - *Max. load*: $\mathcal{O}\left(\sum_{i \in [l]} \frac{m_i}{\min_{j \in Idx(i)} s_j} \right)$

  $\to$ depend on choice of shares, and therefore require tradeoff

- If relation is *skew-free*, max. load is close to expected load with high prob.: $\mathcal{O}\left(\sum_{i \in [l]} \frac{m_i}{\prod_{j \in Idx(i)} s_j} \log^k(s) \right)$
- In general, we want to choose shares s.t. we minimize max. load, which is a difficult optimization problem.

8 Streaming Algorithms

8.1 Basics

- Given **universe** $\mathbb{U}$ with $|\mathbb{U}| = N$, **stream a** $= (a_1, \ldots, a_n) \in \mathbb{U}^n$, n not known in advance.
- We aim to find algorithms using space $\text{polylog}(n + N) = \bigcup_{k \in \mathbb{N}} \log(n + N)^k$.
  $(\mathcal{O}(\text{polylog}(n)) = \mathcal{O}((\log n)^k))$
- SIMPLESAMPLE: Save each element a_i as sample with prob. $1/i$.
  - Resulting item index distributed uniformly.
  - Returns $a \in \mathbb{U}$ with prob. $|\{i \mid a_i = a\}|/n$

- ReservoirSample: Save first k elements. For $i > k$ with prob. k/i replace random element with a_i.
  - Resulting index set $\{i_1, \dots, i_k\} \in \binom{[n]}{k}$ distributed uniformly with prob. $1/\binom{n}{k}$.
  - → Space: $\mathcal{O}(\log n + k \log N)$

8.2 Hashing

- *Idea*: Fix small family of hash functions $\mathcal{H}$ from $\mathbb{U}$ to $\mathbb{T}$ and consider uniform distribution on this family
  - *Truly random* hash functions (uniformly sampled from family $\mathcal{H}$ of all possible functions) are not practical, since for very large universe, the space requirement is prohibitive.

Family $\mathcal{H}$ is

- **universal**, if for all distinct $x, x' \in \mathbb{U}$:

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

  - *Application*: Assign *k-bit signatures* to $S \subseteq \mathbb{U}$ using a universal hash function family. Then, for suitable $k = k(|S|, \delta)$, probability of no collisions (two elements with same signature) is high $(1 - \delta)$.

- **k-universal**, if $\forall x_1 \neq \dots \neq x_k \in \mathbb{U}$:

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

  - $\mathcal{H}$ 2-universal $\Leftrightarrow$ $\mathcal{H}$ universal

- **strongly k-universal**, if $\forall x_1 \neq \dots \neq x_k \in \mathbb{U}$ and $\forall y_1 \neq \dots \neq y_k \in \mathbb{T}$:

$$\Pr_{h \in \mathcal{H}}(\bigwedge_{i \in [k]} h(x_i) = y_i) = \frac{1}{|\mathbb{T}|^k}$$

  iff k-*independent* and *uniform*:

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

  - $\mathcal{H}$ strongly k-universal $\Rightarrow$ $\mathcal{H}$ k-universal

Constructing strongly k-universal hash families:

- Let $q \geq N$ prime power, $g_1 : \mathbb{U} \to \mathbb{F}_q$ injective, $g_2 : \mathbb{F}_q \to \{0, \dots, q-1\}$ bijective.
  - *Prime power*: number expressed as a prime number raised to a pos. integer exponent (e.g. $2^3, 5^2$)
  - *Field* $\mathbb{F}_q$: Körper

- For $\mathbf{a} \in \mathbb{F}_q^k$, let $p_{\mathbf{a}} : \mathbb{F}_q \to \mathbb{F}_q$ of degree $k-1$ the corresponding polynomial. Then we get hash func. $h : \mathbb{U} \to \{0, \dots, q-1\}$

$$\begin{array}{lll} \mathcal{H}_q^k & := \{x \mapsto g_2(p_{\mathbf{a}}(g_1(x))) \mid \mathbf{a} \in \mathbb{F}_q^k\} & \text{is strongly } k\text{-universal} \\ \mathcal{H}_{q,M}^k & := \{x \mapsto g_2(p_{\mathbf{a}}(g_1(x))) \mod M \mid \mathbf{a} \in \mathbb{F}_q^k\} & \text{is strongly } k\text{-universal, if } M \text{ divides } q \end{array}$$

  - $\mathcal{H}_{q,M}^k$ is *close* to strongly k-universal, as long $M \ll q$ (assumes independence and *almost* uniformity)

8.3 Counting Distinct Elements

- Problem cannot be solved (exactly) in space $< \min\{N, n\}$, approximation often sufficient anyway
- Use $\frac{N}{\min S}$ as estimator for $|S| = m$ of subset $S \subseteq \mathbb{U}$
- Count(ε): Let $M \geq 12N^2$, $\mathcal{H}$ strongly 2-universal from $\mathbb{U}$ to $[M]$, $t := \lceil \frac{24}{\varepsilon^2} \rceil$. Determine t-th smallest element of $[M]$ to compute estimator $\frac{tM}{x_t}$.
  - Init list L of t elements $x_i := M + 1$. Insert hash value $h(a)$ of upcoming stream element a in *sorted* L, if it is smaller than the largest element in the list (discard last element in L)
  - When stream ended, return $\frac{tM}{x_t}$ if $x_t \leq M$, else return index of maximal not initial value in L.

  → Space: $\mathcal{O}(t \log M) = \mathcal{O}(\frac{1}{\varepsilon^2} \log N)$

  → Depending on ε, prob., that estimator is close enough to true count is *only* $\frac{3}{4}$.
- MCount(ε, k): Run Count(ε) on independently drawn $2k - 1$ hash functions from $\mathcal{H}$. Return median of all resulting estimators.

  → Space: $\mathcal{O}(k \cdot \frac{1}{\varepsilon^2} \log N)$

  → Depending on ε, prob., that estimator is close enough to true count is *arbitrarily large* with $k = k(\delta)$. (considerable improvement to Count)

8.4 Frequency Moments

- **Frequency** of $u \in \mathbb{U}$ in stream $\mathbf{a}$: $f_u(\mathbf{a}) := |\{i \in [n] \mid a_i = u\}|$
- *Frequency vector* $\mathbf{f} = (f_u)_{u \in \mathbb{U}}$
- **p-th Frequency Moment** of $\mathbf{a}$: $F_p(\mathbf{a}) := \sum_{\substack{u \in \mathbb{U} \\ f_u(\mathbf{a}) > 0}} (f_u(\mathbf{a}))^p$
  - F_0 = distinct elements; $F_1 = n$; $F_2 = ||\mathbf{f}||_2^2$
  - $\sqrt[p]{F_p} = ||\mathbf{f}||_p \overset{p \to \infty}{\longrightarrow} \max_u\{f_u\}$

Estimate frequency moments, since direct computation is difficult.

- Pick random $i \in [n]$, let $r := |\{j \mid j \geq i, a_j = a_i\}|$.
  - $A_k := n(r^k - (r-1)^k)$ (for $k \geq 2$)

  → $E(A_k) = F_k$
- AMS-Estimator: While stream is running, sample a_i with prob. $1/i$ ("reservoir sampling"). Count further occurrences (r) of a_i (reset, if sample overwritten). Return $A_k = n(r^k - (r-1)^k)$.

  → A_k has high variance
- Tug-Of-War: Let $\mathcal{H} : \mathbb{U} \to \{-1, +1\}$ strongly 4-universal. Draw h. Compute $B = \left(\sum_i h(a_i)\right)^2$.

  → $E(B) = F_2, \mathrm{Var}(B) \leq 2F_2^2$
- AVG-ToW: Draw $h_1, \ldots, h_k \in \mathcal{H}$. Compute average of ToW results: $B = \frac{1}{k} \sum_{i=1}^{k} \left(\sum_j h_i(a_j)\right)^2$

  → $E(B) = F_2$ and for $\varepsilon, \delta > 0, k = k(\varepsilon, \delta) : \Pr(|B - F_2| < \varepsilon F_2) > 1 - \delta$

8.5 Sketching

Turnstile Model: Universe $\mathbb{U}$ of size N. Stream of **updates** $(a_1, c_1), \ldots, (a_n, c_n) \in \mathbb{U} \times \mathbb{Z}$.

- Implicitly: **data vector** $\mathbf{d}(i) = (d_u(i))_{u \in \mathbb{U}} \in \mathbb{Z}^{\mathbb{U}}$, init $\mathbf{d}(0) = \mathbf{0}$, *incremented* by update entry c_i at "time step" i for $u = a_i$.
- *Restrictions*:
  - Strict turnstile model: all $d_u(i) > 0$
  - Cash register model: all $c_i > 0$
  - "Normal" stream model: all $c_i = 1, \mathbf{d} = \mathbf{f}$

Instead of storing $\mathbf{d} := \mathbf{d}(n)$, let h map the entries of $\mathbf{d}$ to a much smaller vector of length k called **sketch**. Assume for all algorithms universal family of hash functions $\mathcal{H} : \mathbb{U} \to [k]$.

- Simple Sketch(k): Draw $h \in \mathcal{H}$. Init $\mathbf{0} =: S \in \mathbb{Z}^k$ and update $S[h(a_i)] \mathrel{+}= c_i$.
  - Estimator for data value d_u: $d_u^* := S[h(u)]$
  - $\to$ Assuming strict turnstile model: d_u^* overestimates and $\Pr(d_u^* - d_u \geq \varepsilon ||\mathbf{d}||_1) \leq \frac{1}{2}$ for $k = k(\varepsilon)$.
- Count Min Sketch(k, l): Draw $h_1, \ldots, h_l \in \mathcal{H}$. Update $S[h_j(a), j] \mathrel{+}= c$ for all j.
  - Estimator for data value d_u: $d_u^* := \min_{j \in [l]} S[h_j(u), j]$
  - Assuming strict turnstile model: d_u^* still overapproximates, but now large deviation from ground truth is less likely: $\Pr(d_u^* - d_u \geq \varepsilon ||\mathbf{d}||_1) \leq \delta$ for $l = l(\delta)$.
- **Heavy hitter** $u \in \mathbb{U}$ with $\tau > 0$: $d_u \geq \tau ||\mathbf{d}||_1$ for data vector $\mathbf{d}$ (data item "appears very often")
- CM Heavy Hitters(k, l, τ): Compute CM sketch S with params k, l. Maintain set H of heavy hitters $u \in \mathbb{U}$ from estimator d_u^*. Remove $u \in \mathbb{U}$ from H, when not heavy hitter anymore ($d_u^* < \tau ||\mathbf{d}||_1$). Return all heavy hitters in H.
  - $\to$ Assuming cash register model: With high prob., algorithm returns no u, which is barely a heavy hitter