0 Bound

Markov's Inequality

$$P(|X| \geq a) \leq \frac{E|X|^k}{a^k}$$

Chebyshev's Inequality

$$P(|X-E(X)| \geq a) \leq \frac{D(X)}{a^2}$$

Union Bound

$$P(\bigcup_{i=1}^{n} A_i) \leq \sum_{i=1}^{n} P(A_i)$$

1 High Dimensional Geometry

Properties for unit ball in $\mathbb{R}^d$

- Volume and surface area

$$A(d) = \frac{2(\sqrt{\pi})^d}{\Gamma(\frac{d}{2})}, V(d) = \frac{2(\sqrt{\pi})^d}{d\Gamma(\frac{d}{2})}$$

> 使用 $\pi^{\frac{d}{2}} = \int \cdots \int_{\mathbb{R}^d} \exp(-\sum_{i=1}^{\mathrm{d}} x_i^2)\mathrm{d}x_1 \cdots \mathrm{d}x_d$ 以及 $A(d,r)\mathrm{d}r = \mathrm{d}x_1 \cdots \mathrm{d}x_d$

- Concentration properties
  - $\forall c \geq 1, d \geq 3, \forall X \in \mathbb{R}^d \text{ s.t. } |X| \leq 1$, with probability at least $1 - \frac{2}{c}e^{\frac{c^2}{2}}$, we have $|X_1| \leq \frac{c}{\sqrt{d-1}}$
  - For n points $x_1, x_2, \cdots, x_n$ chosen randomly on the surface of a unit ball, with probability $1 - \mathcal{O}(1/n)$, the following hold:
    - For all i, $|x_i| \geq 1 - \frac{2\log n}{d}$
    - For all $i \neq j$, $|x_i \cdot x_j| \leq \frac{\sqrt{6\log n}}{\sqrt{d-1}}$

> 硬放, 莫得办法

- How to sample uniformly in the unit ball?
  - Sample $x_1, x_2, \cdots, x_d$ from $N(0,1)$, then normalize $x = \frac{1}{|x|}x$

Johnson-Lindenstrauss Lemma

- Randomly select k Gaussian random vectors $u_1, u_2, \cdots, u_k \in \mathbb{R}^d$, and define the mapping $f : \mathbb{R}^d \to \mathbb{R}^k$ as $f(v) = (u_1 \cdot v, u_2 \cdot v, \cdots, u_k \cdot v)$.
- Let v be any vector in $\mathbb{R}^d$, and let f be defined as described above. Then there exists a constant $c > 0$ such that for all $\varepsilon \in (0,1)$, the following holds:
  $P\left(\left|f(v) - \sqrt{k}|v|\right| \geq \varepsilon\sqrt{k}|v|\right) \leq 3e^{-ck\varepsilon^2}$
  The randomness arises from the random vectors $\boldsymbol{u}_i$ used to construct the mapping f.

2 Singular Value Decomposition

Definition and geometric interpretation

Best fit subspace and “greedy” construction

Low rank approximations: F-norm, 2-norm

Left singular vectors and its properties

Relations with the eigen decomposition of $A^\top A$

Power method

Centering data

3 Machine Learning

Perceptron Algorithm

- Algorithm procedure
  - Set the weight vector $w = 0$. Repeat the following steps until all samples have the correct sign:
    - Select a sample x where $x^T w$ has the wrong sign.
    - Update the weight vector:
      - If x is a positive example, $w \leftarrow w + x$.
      - If x is a negative example, $w \leftarrow w - x$.
  - final hypothesis: $h(x) = \mathrm{sign}(w^T x)$
- Why do we need to bias the perceptron?
  - Make sure the hyperplane does not need to pass through the origin
- **linearly separable**: there exists a hyperplane that separates the positive and negative examples.
  - **Hahn-Banach Theorem**: Two disjoint convex sets can be separated by a hyperplane.
  - If the data is linearly separable, the number of updates made by the Perceptron algorithm is at most: $N \leq R^2 \|w^*\|^2$
    - $R = \max_{x \in S} \|x\|$, the radius of the smallest ball enclosing the data.
    - $|x^\top w^*| \geq 1$ for all $x \in S$.
    - proof : every times $w^\top w^*$ increases more than 1, $w^\top w$ increase less than R^2
- **Kernel perceptron algorithm**
  - $f(x) = \mathrm{sign} \sum_{i \in [m]} \alpha_i K(x_i, x)$

Kernel Function

- **Kernel Function** : a function $K(x, x')$ is a kernel function if there exists a mapping $\varphi : \mathbb{R}^n \to \mathbb{R}^D \text{ s.t.}$
  $K(x, x') = \varphi(x)^\top \varphi(x')$
  where perhaps $n \gg d$.
- **Kernel Matrix** K: Defined for a dataset as $K_{ij} = K(x_i, x_j)$.
  - The kernel matrix is a **positive semi-definite matrix**.
- If $k_1(x, y)$ and $k_2(x, y)$ are kernel functions, the following operations also produce kernel functions:
  - $k_1 + k_2$
  - $k_1 \cdot k_2$
  - $f(x)f(y)k_1(x, y)$, where $f(x)$ is any real-valued function.
- **Common Kernel Functions**:
  - **Linear Kernel**: $K(x, y) = x^T y$
  - **Polynomial Kernel**: $K(x, y) = (x^T y + c)^d$
  - **Gaussian Kernel (RBF)**: $K(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right)$

VC Dimension

- The **VC dimension** (Vapnik-Chervonenkis dimension) of a hypothesis class $\mathcal{H}$ is the largest number d such that there **exists** a set of d points that can be **shattered** by $\mathcal{H}$.
- A set of points $\{x_1, x_2, \ldots, x_d\}$ is said to be **shattered** by a hypothesis class $\mathcal{H}$ if, for every possible labeling of the points (i.e., every combination of 0 and 1), there exists a hypothesis $h \in \mathcal{H}$ that correctly classifies the points according to that labeling.
- The **shatter function** (also known as the **growth function**) $\pi_{\mathcal{H}}(n)$ of a hypothesis class $\mathcal{H}$ is the maximum number of distinct labelings that can be realized by hypotheses in $\mathcal{H}$ on any set of n points.
  $\pi_{\mathcal{H}}(n) = \max_{S \subseteq \mathcal{X}, |S|=n} |\{(h(x_1), h(x_2), \ldots, h(x_n)) : h \in \mathcal{H}\}|$
  where $S = \{x_1, x_2, \ldots, x_n\}$ is a set of n points, and $h(x_i) \in \{0, 1\}$ is the label assigned to x_i by hypothesis h.
  - **Sauer's lemma** : $\pi_{\mathcal{H}}(n) \leq \sum_{i=0}^{d} \binom{n}{i} \leq \left(\frac{en}{d}\right)^d$, where d is the VC dimension of $\mathcal{H}$.
    - induction on n, $\binom{n}{\leq d} = \binom{n-1}{\leq d} + \binom{n-1}{\leq d-1}$
  - $\pi_{H_1 \cap H_2}(n) \leq \pi_{H_1}(n) \pi_{H_2}(n)$

Uniform Convergence and Generalization Bound

- true error : $err_D(h) = \mathrm{Prob}_{x \sim D}[h(x) \neq c^*(x)]$
- training error : $err_S(h) = \mathrm{Prob}_{x \in S}[h(x) \neq c^*(x)]$
- Let H be a hypothesis class, and let $\epsilon > 0$ and $\delta > 0$. If a training set S of size: $n \geq \frac{1}{\epsilon}\left(\ln |H| + \ln\left(\frac{1}{\delta}\right)\right)$, is drawn from a distribution D, then:
  - With probability at least $1 - \delta$, every $h \in H$ with true error $\mathrm{err}_D(h) > \epsilon$ has training error $\mathrm{err}_S(h) > 0$.
  - Equivalently, with probability at least $1 - \delta$, every $h \in H$ with training error $\mathrm{err}_S(h) = 0$ has true error $\mathrm{err}_D(h) < \epsilon$.
  - proof : assume true error of h_i is more than ϵ, let A_i denotes the event that h_i has training error 0, then $P(A_i) \leq (1 - \epsilon)^n$, thus $P(\cup A_i) \leq |\mathcal{H}|(1 - \epsilon)^n < \delta$

Online Learning

> Before are all batch learning, which means we have all the data at the beginning. Our goal is to generate a concept $h \in \mathcal{H}$ through this training set so that it rarely makes mistakes on new data.

- **Problem formulation of Online Learning** : The data comes one by one, and we need to make a decision l_t after each data point x_t.
  - Then we know the true label y_t, if $l_t \neq y_t$, we make a change to the model.
- **Halving algorithm** : Assume there is a perfect agent.
  - Each time we select the label given by the majority, and then remove all the wrong ones, so that we can reduce the number of agents by half every time a mistake is made.
  - That means the number of mistakes is less than $\log_2 |\mathcal{H}|$.
- **(randomized) Weighted Majority Algorithm** :
  - We assign a weight to each agent, each time we select the label given by the majority of the weighted sum, and then the weight of the wrong one is multiplied by $1-\epsilon$.
  - Let F_t denote the probability of error in round t, thus $F_t = \frac{\sum_{wrong} w_i}{\sum w_i}$. Let W_t denote the total weight of all agents, then $W_{t+1} = W_t(1 - F_t\epsilon)$.
  - Assume the best agent makes at most N_{opt} mistakes, then

$$W_T = n\prod_{t=1}^{T}(1-F_t\epsilon) \geq (1-\epsilon)^{N_{opt}}$$

  - Take the logarithm of both sides, and we bound the $E(mistakes) = \sum F_t$
- **Potential function method** :
  - Define a potential function, then we have $\Phi_{t+1} \leq \Phi_t - \epsilon$.
  - Thus, we have $\Phi_T \leq \Phi_0 - \epsilon T$, and we can bound the number of mistakes by $\frac{\Phi_0}{\epsilon}$.
  - The TA said the function will be given in the exam.

Boosting Algorithm

- Can a collection of "weak learners" generate a "strong learner"?
  - weak learners : a little better than random guess, $P(wrong) \leq \frac{1}{2} - \gamma$
  - strong learners : reach an arbitrarily small error rate with great probability given enough samples (for example, a polynomial in $\frac{1}{\epsilon}$)
- After each round, we will increase the weight of the wrong answer (multiply by $\alpha > 1$). After a sufficient number of rounds, we will vote all the obtained agents $\{h_i\}_t$ as the final agent.
  - In round $t+1$, the weight of samples with incorrect answers is less than or equal to $\frac{1}{2} - \gamma$:

$$W_{t+1} \leq W_t\left(\alpha\left(\frac{1}{2}-\gamma\right) + \frac{1}{2} + \gamma\right)$$

  - Let m be the number of samples that still make the t experts make wrong judgments after voting. Then we can know that these m samples make at least $\frac{t}{2}$ experts make wrong judgments.

$$m\alpha^{\frac{t}{2}} \leq W_t \leq n\left(\alpha\left(\frac{1}{2}-\gamma\right) + \frac{1}{2} + \gamma\right)^t$$

  - select $\alpha = \frac{\frac{1}{2}+\gamma}{\frac{1}{2}-\gamma}$, then when $t \geq \frac{\log n}{2\gamma^2}$, $m \leq 1$.

4 Streaming

Streaming Model

- n items $\{a_i\}_n$ arrive one at a time (We can never use information about $a_{t+1}, \ldots, a_n$ at time t).
- n is too large, while $a_i \in [1, m]$ is small.
- Goal: design algorithms with $poly(\log n, \log m)$ bit space.

Sampling from a Stream

- Suppose we have the solution at time t, now a_{t+1} comes, decide how should the solution change.
- For example, the probability of sampling a_{t+1} is $\frac{a_{t+1}}{\sum_{i\in[t+1]} a_i}$. The probability for sampling a_i changes from $\frac{a_i}{\sum_{i\in[t]} a_i}$ to $\frac{a_i}{\sum_{i\in[t+1]} a_i}$, becoming $\frac{\sum_{i\in[t]} a_i}{a_{t+1}+\sum_{i\in[t]} a_i}$. So we need to maintain $\sum_{i\in[t]} a_i$ which could help us change the samlpe from X_t to X_{t+1}.

Algorithm Frequent

- count the frequency (within an error of $\frac{n}{k+1}$) of each element of $\{1, \ldots, m\}$
- Algorithm : k counters and a k size list:
  - when encounter an item :
    - increment a counter
    - add the element to the list and set counter to 1
    - decreases each counter by 1

- Whenever an counter decreases 1, the gap between the sum of all counters and the element number we already encounter increases with $k+1$:

$$f_i - \frac{n}{k+1} \le \hat{f}_i \le f_i$$

5 Hash Functions

n-Universal

- A set of hash functions $H = \{h | h : \{1,2,\dots,m\} \to \{0,1,\dots,M-1\}\}$ is **n-universal** if for any $\{x_i\}_n \subset \{1,2,\dots,m\}$ and any $\{y_i\}_n \subset \{0,1,\dots,M-1\}$, we have, $P_{h\sim H}(\forall i \in [n], h(x_i) = y_i) = \frac{1}{M^n}$
  - Ramdomness come from h

Count Distinct Elements

- Algorithm : $S = \{a_i\}$, Keep track of the minimum of $h(a_i)$, use $\frac{M}{\min} - 1$ as estimation of $|S|$
- With the probability of over $\frac{2}{3}$, we have:

$$\frac{M}{6|S|} \le \min \le \frac{6M}{|S|}$$

  - left side : Union Bound
  - right side : $I_i = \mathbf{1}_{h(a_i) < \frac{6M}{|S|}}$, $Y = \sum I_i$, $P(Y=0) \le \frac{Var(Y)}{E^2(Y)} \le \frac{1}{6}$

6 Random Graph

G(n,p)

- Threshold Property : Many properties of graphs (such as connectivity) suddenly appear after p is greater than a certain threshold, while the probability of appearing before this threshold is almost zero.
  - For example, for the existence of isolated vertex, we discuss two cases, $p \ll \frac{\log n}{n}$ and $p \gg \frac{\log n}{n}$.
  - $X = \sum I$ means the total number of a structure
    - For the former : $P(X > a) \le \frac{E(X)}{a} \le 1$
    - For the latter : $P(X=0) \le P(|X - E(X)| \ge E(X)) \le \frac{D(X)}{E(X)^2}$

> **Second Moment Methods**
> Suppose $E(X) > 0$, if $Var(X) = o(E(X)^2)$, then X is almost surely greater that 0.