Probability.

random variable: outcome of the experiment.

The set of possible outcomes is called the sample space. If the sample space is finite, we can assign a probability of occurrence to each outcome. ~~In some~~

E.g. Consider an experiment such as flipping a coin whose outcome is determined by chance.

$$\mathbb{P}(X = H) = \frac{1}{2}.$$
$$\mathbb{P}(X = T) = \frac{1}{2}.$$

The function assigning the probabilities is called a probability distribution function.

Define a probability density function $p(x)$ such that

$$\mathbb{P}(a < x < b) = \int_a^b p(x)\,dx$$

- Expected value: of a random variable X:

$$\mathbb{E}[X] = \int_{-\infty}^{\infty} x\,p(x)\,dx \qquad \text{(continuous case)}$$

- Variance: $\text{Var}[X] = \mathbb{E}[X - \mathbb{E}[X]]^2$.

measures how close to the expected value the random variable is likely to be.

{

The standard deviation $\sigma = \sqrt{\text{Var}[X]}$.

* Gaussian or normal distribution:

if X is a Gaussian random variable

we denote $X \sim N(\mu, \sigma^2)$ $\quad p(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}}$

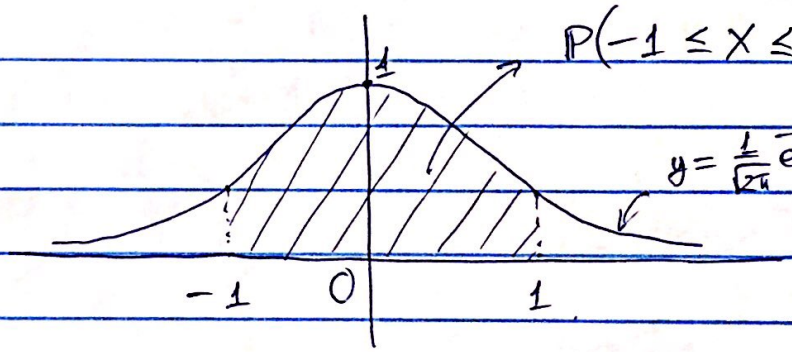
where μ is the mean and σ^2 is the variance.

E.g. Standard Gaussian distribution.

$\mu = 0$ and $\sigma^2 = 1$.

we denote $X \sim N(0, 1)$.

$$p(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}$$



* General Gaussians:

(There is a more formal definition for this, but I only restrict to the following definition)

$\vec{X} = (X_1, X_2, \ldots, X_d) \in \mathbb{R}^d$

is a (standard) Gaussian random vector if X_i's are (independend) standard Gaussian random variables.

$$p(\vec{X}) = \frac{1}{(2\pi)^{d/2}} e^{-\frac{x_1^2 + x_2^2 + \ldots + x_d^2}{2}}$$

Thm: (Gaussian Annulus Theorem) For a d-dimensional standard Gaussian random vector X, and for any $\beta \leq \sqrt{d}$, all but at most $3e^{-c\beta^2}$ of the probability mass lies within the annulus $\sqrt{d} - \beta \leq \|X\|_2 \leq \sqrt{d} + \beta$, where c is a fixed positive constant.

(pf: See p. 24 Foundation of Data Science)

In other words,

$$\mathbb{P}(\sqrt{d} - \beta \leq \|X\|_2 \leq \sqrt{d} + \beta) \geq 1 - 3e^{-c\beta^2}.$$

* Random Projection and Johnson-Lindenstrauss Lemma:

. Nearest neighbor search: Given a database of n points in $\mathbb{R}^d$ where n and d are usually large.

Given a 'query' points q in $\mathbb{R}^d$, we are asked to find the nearest or approximately nearest points to q. the query points. It may be take long time to compute/find or even to process these data points because there are many points in high dimensions.

⇒ dimensionality reduction can be helpful.

Projects the database points to a k-dimensional space with $k \ll d$.

· Let's define the projection function:

$$f: \mathbb{R}^d \to \mathbb{R}^k.$$

standard

Pick k Gaussian vectors $\vec{u}_1, \vec{u}_2, \ldots, \vec{u}_k \in \mathbb{R}^d$.

For any vector $\vec{v} \in \mathbb{R}^d$,

$$f(\vec{v}) := (u_1 \cdot v, u_2 \cdot v, \ldots, u_k \cdot v)$$

The projection $f(v)$ is the vector of dot ~~proj~~ products of v with u_i.

We will show that with high probability,

$$\|f(v)\|_2 \approx \sqrt{k}\, \|v\|_2.$$

Thm: (Random Projection Theorem) Let v be a fixed vector in $\mathbb{R}^d$, and let f be defined as above. $\exists\ c > 0$ and for $\varepsilon \in (0, 1)$,

$$\mathbb{P}\left(\left| \|f(v)\|_2 - \sqrt{k}\, \|v\|_2 \right| \geq \varepsilon \sqrt{k}\, \|v\|_2 \right) \leq 3e^{-ck\varepsilon^2}.$$

where the probability is taken over the random draws of the vector u_i used to construct f.

Facts: The sum of (independent) standard Gaussian r.v.s is also a Gaussian r.v. where the mean and variance are the sums of the individual means and variances.

Pf: of thm: Without loss of generality, suppose $\|v\|_2 = 1$

Then ce $u_i \cdot v = \sum_{j=1}^{d} u_{ij} v_j$

the random variable $u_i \cdot v$ has Gaussian density with zero mean and unit variance:

$$\text{Var}[u_i \cdot v] = \text{Var}\left[\sum_{j=1}^{d} u_{ij} v_j\right]$$

$$= \sum_{j=1}^{d} v_j^2 \,\text{Var}[u_{ij}] = \sum_{j=1}^{d} v_j^2 = 1.$$

Since $u_1 \cdot v, u_2 \cdot v, \ldots, u_k \cdot v$, are independent Gaussian random variables, $f(v)$ is a standard Gaussian in $\mathbb{R}^k$. By the Gaussian Annulus theorem, for $\varepsilon \in (0,1)$

$$\mathbb{P}(\sqrt{k} - \varepsilon\sqrt{k} \leq \|f(v)\|_2 \leq \sqrt{k} + \varepsilon\sqrt{k}) \geq 1 - 3e^{-ck\varepsilon^2}$$

□

The random projection theorem establishes that the probability of the length of the projection of a single vector differing significantly from its expected value is exponentially small in k.

Def: The mapping $f: \mathbb{R}^d \to \mathbb{R}^k$ is called K-bi-Lipschi for a subset $X \subseteq \mathbb{R}^d$ if there exists a constant $c > 0$ such that

$$cK^{-1}\|p-q\|_2 \leq \|f(p) - f(q)\|_2 \leq c\|p-q\|_2.$$

for all $p, q \in X$.

The least K for which f is K-bi-Lipschitz is called the distortion of f and is denoted dist(f). We will refer to f as a K-embedding of X.

Thm: (Johnson–Lindenstrauss Lemma)

For any $0 < \epsilon < 1$ and any integer n, let $k \geq \frac{3}{c\epsilon^2} \ln n$ for some positive constant c. For any set X of n points in $\mathbb{R}^d$, draw k random vectors $u_1, \ldots, u_k$ independently from a standard Gaussian distribution and define $f: \mathbb{R}^d \to \mathbb{R}^k$ as $f(v) = (u_1 \cdot v, \ldots, u_k \cdot v)$. Then for all pairs of points v_i and v_j in X, with probability at least $1 - \frac{3}{2n}$,

$$(1-\epsilon)\sqrt{k}\,\|v_i - v_j\|_2 \leq \|f(v_i) - f(v_j)\|_2 \leq (1+\epsilon)\sqrt{k}\,\|v_i - v_j\|_2$$

pf: Let $X = \{\vec{v}_1, \ldots, \vec{v}_n\}$.

Let $\vec{v}_{ij} = \vec{v}_i - \vec{v}_j$.

By the Random Projection theorem,

$$\mathbb{P}\left(\left| \|f(v_{ij})\|_2 - \sqrt{k}\,\|v_{ij}\|_2 \right| \geq \epsilon\sqrt{k}\,\|v_{ij}\|_2 \right) \leq 3e^{-ck\epsilon^2}.$$

$$\mathbb{P}\left(\|f(v_{ij})\|_2 \notin \left[(1-\epsilon)\sqrt{k}\,\|v_{ij}\|_2, (1+\epsilon)\sqrt{k}\,\|v_{ij}\|_2\right] \right) \leq 3e^{-ck\epsilon^2}.$$

Now we need to used the union bound.

Recall that if $A_1, \ldots, A_m$ are events of the outcomes:

$$\mathbb{P}(A_1 \cup A_2 \cup \ldots \cup A_m) \leq \sum_{i=1}^{m} \mathbb{P}(A_i)$$

Let $A_{ij} = \left\{ \|f(v_{ij})\|_2 \notin \left[(1-\epsilon)\sqrt{k}\,\|v_{ij}\|_2, (1+\epsilon)\sqrt{k}\,\|v_{ij}\|_2\right] \right.$

$$\mathbb{P}\left(\bigcup_{i,j} A_{ij} \right) \leq \sum_{i,j} \mathbb{P}(A_{ij}) = \binom{n}{2} 3e^{-ck\epsilon^2}$$

$$\leq \frac{n^2}{2} 3e^{-ck\epsilon^2} = \frac{3}{2} e^{2\ln n - ck\epsilon^2}$$

$$\leq \frac{3}{2} \cdot \frac{1}{n}.$$

~~Remark:~~

Remark: The number of dimensions in the projection is only dependent logarithmically on n. Since k is often much less than d, this is called a dimension reduction technique. In applications, the dominant term is typically the $\frac{1}{\varepsilon^2}$ term.

Finding Similar Items

For example, we want to find whether two documents are similar to detect plagiarism, mirror websites, etc.
Finding similar items are also helpful for recommender systems. (e.g., find users with similar buying patterns).

But how to measure similarity?

- Def: (Jaccard similarity) Given two sets S_1 and S_2, Jaccard similarity of S_1 and S_2 is defined as

$$\frac{|S_1 \cap S_2|}{|S_1 \cup S_2|}.$$

E.g. Let $S_1 = \{1, 2, 3, 4, 7\}$ and $S_2 = \{1, 4, 9, 7, 6\}$.
⇒ then $|S_1 \cup S_2| = 7$.
and $|S_1 \cap S_2| = 3$.

⇒ the Jaccard similarity of S_1 and S_2 is $\frac{3}{7}$.

- Shingling of Documents:
Def: (k-shingles) any substring of length k.

E.g. Let D be a document abcdabd.
Then 2-shingles = {ab, bc, cd, da, ~~cb~~, bd}.

⇒ To compare two documents to know how similar they are :

. Compute k shingles for a suitable value of k.

. Compare the set of k shingles based on Jaccard similarity. We will often map the set of shingles to a set of integers via hashing. For example, if $k = 9$, and consider the 26 letters and white space. The possible number of k-shingles is 27^9.

⇒ consider a hash function that map shingles to a range of $1, 2, \ldots, 27^9$.

* Challenges:

. documents are large ⇒ the number of shingles from a document could be very large.

⇒ storing them for all documents in the main memory for similarity comparsison is infeasible.

* Minhash:

We compute a single min-hash of a set of shingles S as follows. Generate a random permutation σ_1 of all possible shingles (e.g. for $k = 9$, it is a permutation of 1 to 27^9) and report the element in S that appears first in σ_1.

The fingerprint $\hat{S}$ of S consists of t minhashes computed from t randomly generated permutations $\sigma_1, \sigma_2, \ldots, \sigma_t$. To compute an estimate of the Jaccard similarity of S_1 and S_2, we compute the number r of minhashes that match. Then the estimated Jaccard similarity is r/t.

- Matrix representation: convert a series of sets to a single matrix. (similar to term-document matrix)

E.g. Consider $S_1 = \{1, 2, 5\}$.

$S_2 = \{3\}$

$S_3 = \{2, 3, 4, 5\}$.

$S_4 = \{1, 4, 6\}$.

Element	S_1	S_2	S_3	S_4
1	1	0	0	1
2	1	0	1	0
3	0	1	1	0
4	0	0	1	1
5	1	0	1	0
6	0	0	0	1

$= M.$

$$M_{ij} = \begin{cases} 1 & \text{if } i \in S_j \\ 0 & \text{otherwise} \end{cases}$$

Now, let's compute Minhash:

Step 1: Randomly permute items (by permuting the rows of the matrix)

Element	S_1	S_2	S_3	S_4
2	1	0	1	0
5	1	0	0	0
6	0	0	1	1
1	1	0	0	1
4	0	0	1	1
3	0	1	1	0

Step 2: Record the first 1 in each column.

$$m(S_1) = 2$$
$$m(S_2) = 3$$
$$m(S_3) = 2$$
$$m(S_4) = 6$$

Step 3: Estimate the Jaccard similarity $JS(S_1, S_2)$ as

$$\widehat{JS}(S_1, S_2) = \begin{cases} 1 & \text{if } m(S_i) = m(S_j) \\ 0 & \text{otherwise.} \end{cases}$$

Lemma: $P(m(S_i) = m(S_j)) = JS(S_i = S_j) = \dfrac{|S_i \cap S_j|}{|S_i \cup S_j|}$.

Pf: There are 3 types of rows:

1) There are x rows with 1 in both columns.
2) There are z rows with 0 in both columns.
3) There are y rows with 1 in one column and 0 in the other.

The total number of rows is $x + y + z$.

and $JS(S_i, S_j) = \dfrac{x}{x+y}$.

Let row l be $\min\{m(S_i), m(S_j)\}$. It is either type (1) or (3), and it is type (1) with probability $\dfrac{x}{x+y}$. This is the case that $m(S_i) = m(S_j)$.

We need to repeat the process several times. That is, we consider t random permutations. ~~and~~ $\{m_1, \dots, m_t\}$ and also t random variables $\{X_1, \dots, X_t\}$ where

$$X_\ell = \begin{cases} 1 & \text{if } m_\ell(S_i) = m_\ell(S_j). \\ 0 & \text{otherwise.} \end{cases}$$

Let $Y_\ell = \frac{1}{t}\left(X_\ell - JS(S_i, S_j)\right)$.

Let $M = \sum_{\ell=1}^{t} Y_\ell$ and $A = \sum_{\ell=1}^{t} X_\ell$, and note that $-1 \le Y_\ell \le 1$. and $\mathbb{E}[M] = 0$.

Thm: (Chernoff–Hoeffding inequality). Consider a set of n i.i.d. random variables $\{X_1, \dots, X_n\}$ such that $-\Delta \le X_i \le \Delta$, $\mathbb{E}[X_i] = 0$ and let $M = \sum_{i=1}^{n} X_i$. Then for any $\alpha \in (0, 1/2)$.

$$\mathbb{P}(|M| > \alpha) \le 2\exp\left(\frac{-\alpha^2}{2n\Delta^2}\right).$$

Applying Chernoff ineq. with $\Delta = 1$, $n = t = \frac{2}{\varepsilon^2}\ln\left(\frac{2}{\delta}\right)$ to say

$$\mathbb{P}(|JS(S_i, S_j) - A| < \varepsilon) > 1 - \delta.$$

That is, the Jaccard similarity is within ε error with probability at least $1-\delta$ if we repeat

$$t = \frac{2}{\varepsilon^2}\ln\left(\frac{2}{\delta}\right) \quad \text{times.}$$

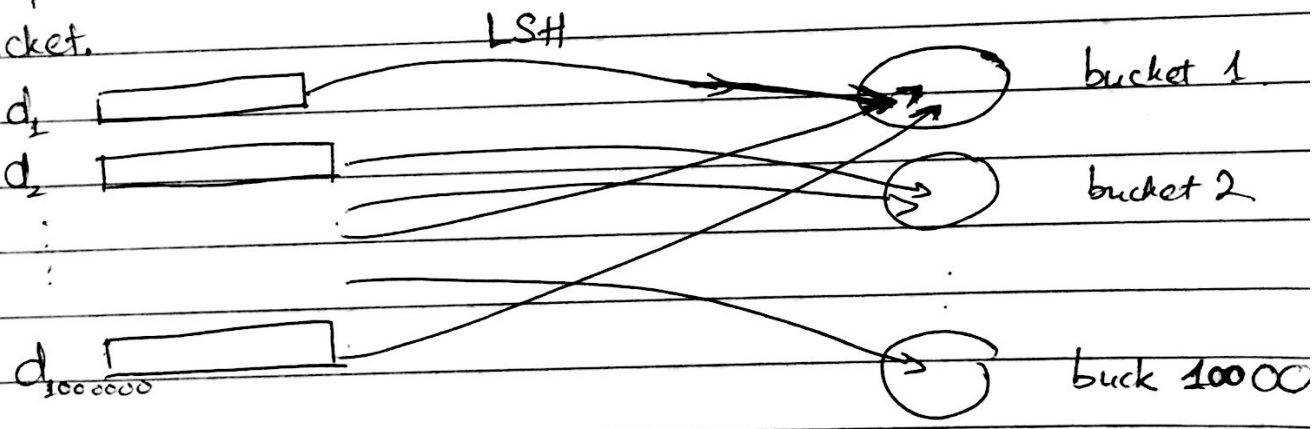
In 2007, Google reported using minhash for Google News personalization.

* Locality Sensative Hashing: (LSH)

Minhash is time ~~com~~ consuming.

=> A data structure that helps us here to reduce the running time significantly is locality sensitive hashing.

Roughly speaking, LSH is a hashing mechanism such that items with higher similarity have higher probability of colliding into the same bucket than other. We will use multiple such hash functions, and only compare the documents that are hashed to the same bucket.



But we would need to worry about

1) False positive: when two "non-similar" items are hashed to the same bucket => increases the search time

2) False negative: when two similar items are not hashed to the same bucket under any of the chosen hash functions from the family.

- LSH has found wide-spread applications in near-duplicate detection, nearest neighbor search, Anti-spam detection, etc. (See the textbook)

* Approximate Near Neighbor search:

- Def: (Near neighbor problem) Given a set of points V, a distance metric d and a query point q, find any point x close to query point q such that $d(x, q) \leq R$

The problem is easy to solve in low dimension (e.g. via brute force, or voronoi diagram construction). However, the complexity increases exponentially in dimension.

$\Rightarrow$ we relax the problem.

- Def: (Approximate Near neighbor pt Problem) Given a set of points V, a distance metric d, and a query point q, the (c, R)-approximate near neighbor pt problem if there exists a point x such that $d(x, q) \leq R$, then one must find a point x' such that $d(x', q) \leq cR$ with probability $> 1-\delta$ for a given $\delta > 0$.

Def: (Locality Sensitive Hashing) A family of hash functions $\mathcal{H}$ is said to be (c, R, P_1, P_2)-sensitive LSH for a metric d if it satisfies the following conditions:

1) $\mathbb{P}[h(x) = h(y)] \geq p_1$ for all x and y such that $d(x,y) \leq R$.

2) $\mathbb{P}[h(x) = h(y)] \leq p_2$ for all x and y such that $d(x,y) \geq cR$.

3) $p_1 > p_2$.

E.g. 1) Let $V \subseteq \{0,1\}^n$ and ~~$d(x,y) = \text{Hamm}$~~

$$d(x,y) = \sum_{i=1}^{n} \mathbb{1}\{x_i \neq y_i\}$$

Let R and cR be both less than n. Define $\mathcal{H} = \{h_1, \ldots, h_n\}$ such that $h_i(x) = x_i$.

If $d(x,y) \leq R \Rightarrow$ x and y have at most R different bits.

$\Rightarrow$ x and y have at least $n-R$ same bits.

$\Rightarrow \mathbb{P}(x_i = y_i) \geq \frac{n-R}{n} = 1 - \frac{R}{n}$.

$\Rightarrow p_1 \geq 1 - \frac{R}{n}$.

Similarly, $p_2 \leq 1 - \frac{cR}{n}$

~~E.g. 2) (Jaccard distance).~~

E.g. 2) (Cosine distance) The ~~cosid~~ cosine distance of two nonzero vectors $\vec{a}$ and $\vec{b}$ is defined as

~~$d(a,b) =$~~ $\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\|_2 \|\vec{b}\|_2}$

cosine distance $\theta = \arccos(\theta)$

* Algorithm:

We use functions of the form $g(p) = (h_1(p), h_2(p), \ldots, h_k(p))$

- In preprocessing, we build up our data structure as follows:
  - Select $g_1, \ldots, g_L$ independently at random. (k and L are functions of c and R)
  - hash every point $p \in P$ to L buckets $g_1(p), \ldots, g_L(p)$
- When querying, we proceed as follows:
  - retrieve points from buckets $g_1(q), \ldots, g_L(q)$ until either we have retrieved the points from all L buckets, or we have retrieved more than $3L$ points in total.
  - answer query based on retrieved points.

Note that hashing takes nk, and with L buckets, the total time is $O(nkL)$.

Lemma: (Indyk - Motwani '98)

The above algorithm solves c-approximate nearest neighbor with

- $L = Cn^{\rho}$ hash functions, where $\rho = \frac{\log p_1}{\log p_2}$, C is a function of p_1 and p_2
- a constant success probability for fixed query q.