

Table of Contents for CS246 – For Final Exam

1. Distributed file systems:	2
2. Association Rules + Frequent Itemsets	3
3. Finding similar items: Locality Sensitive Hashing (LSH)	4
5. Clustering	6
6. Dimensionality Reduction	7
7. Recommender Systems 1.....	9
8. Recommender Systems 2: Latent Factor	9
9. Pagerank	10
10. Link Analysis	10
11. Community detection in Graphs.....	11
12. Graph Representation Learning.....	11
13. Decision Trees	12
14. Support Vector Machine	13
15. Mining Data Streams 1.....	14
16. Mining Data Streams 2:.....	14
17. Advertising	16
18. Bandits	17

$$\sum_{k=0}^n z^k = \frac{1 - z^{n+1}}{1 - z}$$

Geometric series

1. Distributed file systems: [01-intro](#)

- a) MapReduce: p.37
 - a. Failures p.44
 - b. Problems suitable for MapReduce: p.58, 59
 - c. Join by Map-Reduce: p.60, 61
 - d. Problems not suitable for MapReduce: p.62
 - e. Cost of running MapReduce: p.64+66
- b) RDD: p.50
 - a. Operations: p.51
- c) PySpark DataFrame: p.53

2. Association Rules + Frequent Itemsets: [02-assocrules.pdf](#)

- a) Examples of items & baskets: p.5+7
- b) **Support** of itemset, association rule = number of times it appears in baskets
- c) Items appearing in at least “s (**support threshold**)” baskets: **frequent** itemsets (p.9)
- d) **Association** rule: $\{i_1, i_2, \dots, i_k\} \rightarrow j$ (p.11)
- e) **Confidence** of association rule: $\text{conf}(I \rightarrow j) = \text{support}(I \cup j) / \text{support}(I)$ (p.11)
- f) **Interest** of association rule $I \rightarrow j$: $\text{Interest}(I \rightarrow j) = |\text{conf}(I \rightarrow j) - \text{Pr}[j]|$ (interesting rules: have $\text{Interest} > 0.5$) (p.12)
- g) **Maximal frequent itemsets**: immediate supersets are not frequent (p.17, example: p.18)
- h) **Closed itemsets**: immediate supersets have smaller supports (p.17, example: p.18)
- i) We measure the cost by the **number of passes** an algorithm makes over the data (p.21)
- j) **Counting pairs** in memory: p.26 + p.28
- k) **Counting pairs with A-Priori** alg: p.32
 - a. Extension to frequent k-itemsets: p.35 → example: p.36
- l) **PCY** algorithm:
 - a. 1st pass: p.40 (top of the page);
 - b. Transition: p.42;
 - c. 2nd pass: p.43

3. Finding similar items: Locality Sensitive Hashing (LSH): [03-lsh.pdf](#)

- Naïve approach to finding all pairs of data (x_i, x_j) with distance $\leq s$: $O(N^2)$ (p.8)
- Min-Hash / LSH** steps: p.13+52 [**downside**: can produce false negatives: pairs of similar items not detected]
 - Shingling**: convert document to set of k-shingles then hash the shingles to 4-byte integers (p.16)

- Jaccard** similarity (p.18) for $C_i = S(D_i)$:

$\text{sim}(D1, D2) = |C1 \cap C2| / |C1 \cup C2|$ [numerator: dot product of two columns in matrix below]

	Documents			
Shingles	1	1	1	0
	1	1	0	1
	0	1	0	1
	0	0	0	1
	1	0	0	1
	1	1	1	0
	1	0	1	0

(p.19)

- Jaccard distance: $d(C1, C2) = 1 - |C1 \cap C2| / |C1 \cup C2|$

- Min-Hashing (suitable for Jaccard similarity)**: $h(C)$ s.t. $\text{sim}(C_1, C_2)$ is high $\rightarrow P(h(C_1) = h(C_2))$ is high + vice versa (p.22) $\rightarrow \mathbf{P(h(C1)=h(C2))=sim(C1,C2)}$

- Minhash function for permutation π : $h_\pi(C) = \min_{\pi} \pi(C)$ [first 1 in permuted column C] \rightarrow example: p.25
- $\mathbf{Pr[h_\pi(C_1) = h_\pi(C_2)] = sim(C_1, C_2)}$ [proof: p.27]
 - One-pass implementation: p.31
 - Universal hashing:

$$h_{a,b}(x) = ((a \cdot x + b) \bmod p) \bmod N$$
 [a,b: rnd int; p: prime > N]

iii. **Locality Sensitive Hashing**: (p.38)

- Divide signature matrix M into b bands of r rows [$M = b \times r$]
- Hash all bands to a hash table with k buckets (large k)
- Candidate pairs: hash to same bucket for ≥ 1 bands
- Tune b&r to catch most similar pairs but few non-similar pairs
- [assuming same bucket == identical bands]
- Example calculation of false negatives $((1-S')^b)$: p.41 + 47
- Example calculation of false positives $1 - ((1-S')^b)$: p.42 + 47
- S curve** (probability of sharing a bucket vs similarity of two sets) $(1 - (1-t')^b)$: p.48 + 49 (example for b = 20, r = 5)

4. LSH Theory: [04-lsh theory.pdf](#)

- Probability that C_1 and C_2 are candidate pairs: $1 - (1-S')^b$ (P.10)
- Distance measures: Jaccard, Cosine, L1 (Manhattan), L2 (Euclidean)** (p.16)
 - Cosine distance (p.43): $d(A,B) = \theta = \arccos \left(\frac{A \cdot B}{||A|| \cdot ||B||} \right)$ [range: 0 to π]
 - Cosine similarity (p.43): $1 - d(A,B) \rightarrow$ or $\rightarrow \cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||}$

3. **Locality-Sensitive (LS) Families:** family H of hash functs (d_1, d_2, p_1, p_2) -sensitive $[d(x,y) \leq d_1 \rightarrow \Pr(h(x)=h(y)) \geq p_1] [d(x,y) \geq d_2 \rightarrow \Pr(h(x)=h(y)) \leq p_2]$ (p.18)
 - a. Example for min-hashing with Jaccard distance (d):
 $\Pr[h(x) = h(y)] = 1 - d(x,y)$ for $h \in H$ (p.20)
 - b. **Min-Hashing for Jaccard similarity:** $(d_1, d_2, (1-d_1), (1-d_2))$ -sensitive family for any $d_1 < d_2$ (p.21)
 - c. **LSH for Cosine Distance:** $(d_1, d_2, (1-d_1/\pi), (1-d_2/\pi))$ -sensitive
 - i. Random hyperplanes: p.45
 - d. LSH for Euclidean distance: p.50, 55,
 - e. **AND-Construction:** e.g. rows in a band (p.22)
 - i. $H(x) = h(y)$ if and only if all $h_i(x) = h_i(y)$
 $\rightarrow (d_1, d_2, p_1^r, p_2^r)$ (p.24)
 - f. **OR-Construction:** e.g. many bands (p.22)
 - i. $H(x) = h(y)$ if and only if $h_i(x) = h_i(y)$ for at least 1 i
 $\rightarrow (d_1, d_2, 1-(1-p_1)^b, 1-(1-p_2)^b)$ (p.26)
 [Assumption: h_i 's are independent]
 - g. **Impacts of AND vs OR:** AND shrinks all probs.; OR grows all probs. (p.27) \rightarrow Summary: p.39 [ideal: get **p1 to 1, p2 to 0**]

**** False positive and false negative:** P.32 \rightarrow (increasing b : reduces false negative+increases false positives, increasing r : reduces false positives+increases false negatives [p.27])

- h. Composing constructions: AND \rightarrow OR or vice versa (p.29)
 - i. **r-way AND followed by b-way OR** (p.29):
 $\Pr[\text{min 1 shared bucket=candidate pair}] = 1-(1-s^r)^b$
 Threshold t : $1-(1-t^r)^b = t$
 we're improving the sensitivity ala low prob less than t , high prob greater than t (p.37)
 - ii. **b-way OR followed by r-way AND** (p.34):
 $\Pr[\text{min 1 shared bucket=candidate pair}] = (1-(1-s)^b)^r$
 - iii. (3; 4; 5) way OR-AND-OR $\rightarrow 1 - (1 - (1 - s)^3)^4)^5$ [p.31 exam 2019 solution]
- Example calculating number of hash functions required:**
 P.36

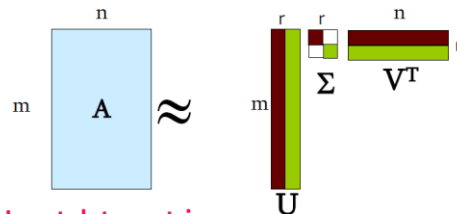
5. Clustering: [05-clustering.pdf](#)

- a. Representing documents (p.9):
 - i. Vectors → cosine distance for similarity
 - ii. Sets → Jaccard distance
 - iii. Points → Euclidean distance
- b. Curse of dimensionality: must traverse $(0.001)^{1/d}$ of space to capture 0.1% of data (p.12)
- c. Clustering (p.14):
 - i. **Agglomerative (bottom-up; hierarchical)**: each point a cluster → combine nearest every time
 - 1. Good for non-convex shapes [p.17]
 - 2. Dendrogram: p.19
 - ii. **Divisive (top-down)**: start one cluster, split gradually
 - iii. **Centroid**: avg of all data points in cluster VS **clustroid**: one point from cluster closest to all others (p.21)
 - iv. Merging clusters:
 - 1. Based on proximity of centroids: Good for convex clusters (p.26)
 - 2. Based on proximity of nearest points b/w two clusters: good for concentric clusters (p.27)
 - v. **K-Means Clustering**: uses Euclidean distance/space [assumes clusters normally distributed in each dimension + axes are fixed, no tilted ellipses allowed!]
 - 1. K-means++: p.30
 - 2. Choosing right k: p.35
 - 3. **BFR Algorithm**: [assumes clusters normally distributed in each dimension + axes are fixed, no tilted ellipses allowed!]
 - a. Overview: p.41
 - i. Step 1: p.42
 - ii. Step 2: p.43 & 45
 - iii. Step 3-4: p.47
 - iv. Step 5: p.48
 - b. Three classes of points: p.43,44
 - i. DS: p.45 & 46
 - c. Proximity criteria for adding a point to a cluster: **Mahalanobis Distance** (p.52, 53)
 - d. Merging criteria for two clusters: variance of combined below a threshold (p.55)
 - vi. Cure Algorithm: Euclidean distance + clusters of any shape:
 - 1. Step 1: p.59; Step 2: p.63

6. Dimensionality Reduction [06-dim_red.pdf](#)

- First dimension: direction with greatest variance; second: second largest (p.5)
- Rank of matrix A: Number of linearly independent rows of A (p.6)
- SVD**: A (m docs x n terms, input data matrix) \sim U (m x r concepts [=latent dimensions, latent factors], left singular vectors, **user-to-concept factor matrix**) . Σ (rxr, singular values, diagonal, **strength of each concept**, sorted in decreasing order) . V^T (rxn, right singular vectors, **movie-to-concept factor matrix**) [p.10] [example: p.22]

$$A \approx U \Sigma V^T = \sum_i \sigma_i \mathbf{u}_i \circ \mathbf{v}_i^T$$



- Real matrix A can be uniquely decomposed as $A = U \Sigma V^T$
 - U, V: column orthonormal $\rightarrow U^T U = I$
- First right singular vector = first row of V^T (p. 22)
- Variance (spread on the v_1 axis (element 1,1 from Σ . (p.23)
- $U\Sigma$ gives coords of points along projection axis [first column: projection of users on a concept] (p.24)
- Dimensionality reduction: set (r-k) smallest singular value (in Σ) to zero (i.e. finding the *rank k* approximation of A) [pick r s.t. the retained singular values have at least 90% of the total energy, i.e. sum of their squares (p.33)]

- Reconstruction error**: Frobenius Norm (p.30+31)

$$\|A - B\|_F = \sqrt{\sum_{ij} (A_{ij} - B_{ij})^2}$$

- Computing SVD: finding principal eigenvector and eigenvalue (p.35-39)
 - Steps: P.40
 - Complexity: p.41
- QUERY using SVD: map query into concept space: $q * V$ (p.45,46,47[find user d's similarity with query])
- Drawbacks of SVD: p.49

Table of Contents for CS246 – For Final Exam

- d. **CUR**: p.56 + p.58 (pick 4k points for a rank-k approximation)
 - i. SVD vs CUR: p. 60-61

7. Recommender Systems 1: [07-recsys1.pdf](#)

- a. X = set of customers; S = set of items
- b. $u: X \times S \rightarrow R$ (p.9)
 - i. R = set of ratings (ordered set)
- c. **Content-based Recommendation:** Recommend items to customer x like previous items rated highly by x
 - i. Creating item profiles for text mining: p.17-18
 - ii. Create user profiles: p.19
 - iii. Pros: p.20, cons: p.21
- d. **Collaborative Filtering:**
 - i. **Cosine Similarity+Pearson correlation coefficient:** p.24
 - ii. **User-user collaborative filtering (prediction):** p.26
 - iii. **Item-item collaborative filtering (prediction):** p.27+example: p.28-32 \rightarrow OFTEN WORKS BETTER THAN USER-USER
 - 1. Common practice formula: p.33
 - 2. Pros/Cons: p.35 [no feature selection needed unlike user-user]
 - iv. Complexity: **finding k nearest customers:** p.42
- e. Evaluating predictions: p.40 [Root-mean-square error **RMSE**]

8. Recommender Systems 2: Latent Factor Models [08-recsys2.pdf](#)

- a. Root mean square error (RMSE): p.5
- b. Estimation considering local and global effects (deviations): p.10 (definition) and p.15 (method)
- c. Using SVD for recommendation (needs fully defined R):
 - i. Rating of user x for item i : $A = R, Q = U, P^T = \Sigma V^T \rightarrow \hat{r}_{xi} = q_i \cdot p_x$ (p.25)
 - ii. SVD minimizes sum of squared error (=reconstruction error) \rightarrow
 $RMSE = 1/c * \sqrt{SSE}$ (SVD also minimizes SSE: p.26)
- d. **Latent Factor Model:**
 - i. Regularization of objective function (defined in p.29)+gradient descent: p.36
 - 1. Stochastic GD: p.39
 - ii. With biases: p.44 and p.45

9. Pagerank [09-pagerank.pdf](#)

- a. Definition: page j , importance $r_j \rightarrow n$ out-links $\rightarrow r_j/n$ each
 $r_j = \sum_{i \rightarrow j} \left(\frac{r_i}{d_i} \right)$ with $d_i = \text{out-degree of node } i$ (p.21)
 $r = \text{stationary distribution of random walk}$ (p.31) \rightarrow it is unique and it exists (p.32)
- b. **Stochastic adjacency matrix M :** $M_{ji} = \frac{1}{d_i}$ (p.23) & $r = M \cdot r$ (p.23) $\rightarrow r = \text{an eigenvector of } M$ with eigenvalue = 1 (p.26)

$$r_j^{(t+1)} = \sum_{i \rightarrow j} \frac{r_i^{(t)}}{d_i} \quad \text{or equivalently} \quad r = Mr \quad (\text{p.34})$$
 - i. Power iteration for finding r : p.27 \rightarrow example: p.29
- c. Problems with pagerank:
 spider traps (absorb pagerank, pagerank not what we want) p.37 \rightarrow solution: teleport with prob $1-\beta$ (p.39)
 dead-ends (leak page rank, adjacency matrix not column-stochastic anymore [columns sum to zero]): p.37 \rightarrow solution: teleport with prob 1 (p.41)
- d.
 - i. **Google's solution:** p. 44 [+Google matrix $\rightarrow r^{\text{new}} = A \cdot r^{\text{old}}$] $\rightarrow \beta \cong 0.8, 0.9$
 1. Full Algorithm: p.50 [more complete: in p.4 of [10-spam.pdf](#)]
 - a. Cost of power method: p.53
 2. Block-stripe update alg: p.56-57

10. Link Analysis [10-spam.pdf](#)

- a. **Full pagerank algorithm:** p.4
- b. **Topic-specific (aka Personalized) pagerank:** p.9
- c. **SimRank** (fixed teleport set): p.17
- d. Pixie random walk algorithm (get top 1k pins with highest visit count): p.31
 - i. Pros: p.34
- e. Comparison b/w all pageRank methods: p.35
- f. **TrustRank:**
 - i. Term spam: add important words to your page (p.40) \rightarrow Google's solution: p.41 [what links to a page say about the page]
 - ii. Type of pages from spammer's pov: p.48
 1. Pagerank of target page resulting from accessible and own pages: p.49
 - a. Fighting spam farms: p.54, 55
 - b. TrustRank == Pagerank with trusted pages as teleport set (p.55)
 - i. Justification: p.56
 - ii. Picking seed set: p.58
 - iii. Spam mass estimation: p.61

11. Community detection in Graphs [11-graphs1.pdf](#)

- a. Finding densely linked clusters; using **Personalized PageRank (PPR)**: p.10
- b. Cut of a cluster: p.13
- c. Graph partitioning criteria:
 - i. **Conductance**: p.15
 - ii. **Algorithm**: p.17+18 (calculating $\phi(A_{i+1})$ using $\phi(A_i)$) +
 - 1. Approximate PageRank (PageRank-Nibble):
 - a. Undirected graph, lazy random walk: p.20
 - b. Page rank vector: p.21
 - c. ALGORITHM: p.25 → example: p.26
 - i. Runtime + approximation guarantee: p.27
- d. Motif-based clustering:
 - i. **Conductance** for motifs: p.35
 - ii. Steps: p.36
- e. Network modularity:
 - i. Measure of how well network is partitioned into communities [sets of tightly connected nodes] (p.40)
 - ii. Null model: configuration model → expected number of edges b/w nodes i and j of deg k_i and k_j in G' (rewired network for G , with same degree distribution but random connections): $k_i k_j / 2m$ (p.41)
 - iii. Sum of degrees of all nodes = $2 \cdot m$ (number of edges)
 - iv. **Modularity of partitioning S of graph G** : p.42 → p.43 for weighted graph
 - v. Maximize Modularity → identify communities
- f. Louvain Algorithm for maximizing modularity: algorithm: p.51 [greedy strategy; greatly scalable]
 - i. 1st phase: p.47
 - 1. Modularity gain: p.48
 - ii. 2nd phase: p.50

12. Graph Representation Learning [12-graphs2.pdf](#)

- a. Encoder:
 - i. Shallow encoding: p.17
 - ii. Similarity: random-walk embedding: p.22, 23
 - 1. Pros of random walk: 24
 - 2. Optimization objective: p.29
 - a. Approximation: p.32-33
 - 3. ALGORITHM: p.34
- b. Generalized random walk (**node2vec**): biased random walk: p.39, 42,
 - i. Algorithm: p.43 → example: p.45
- c. Applications of embeddings [clustering/community detection, node classification, link prediction]: p.48

13. Decision Trees [13-dt.pdf](#)

- a. Regression:
 - i. Split criteria: **purity** (p.19)
 - ii. Prediction: p.30
 - iii. PLANET algorithm: settings: p.33
 - 1. Overview of steps: p.35 + p.53
 - 2. Master node: p.41
 - 3. MapReduce initialization: p.45-46
 - 4. MapReduce FindBestSplit: p.48,49, Map: p.50, reducer: p.51
- b. Classification:
 - i. Split criteria: **information gain** (p.20)
 - 1. Entropy: p.21
The entropy of X : $H(X) = -\sum_{j=1}^m p(X_j) \log p(X_j)$
 - 2. Conditional entropy: p.25
 - 3. $IG(Y|X) = H(Y) - H(Y|X)$ [p.26]
 - ii. Stopping criterion: p.29
 - iii. Prediction: p.30
- c. Ensembles, bagging, and RandomForest: p.58-61

14. Support Vector Machine (SVM) [14-svm.pdf](#)

- a. **Margin γ** : Distance of closest example from the decision line/hyperplane (basics: p.13; normalized: p.21)
- b. Distance from a point to the margin ($\gamma: w \cdot x + b = 0$): $|w \cdot A + b|$ (p.16)
- c. **Prediction** = $\text{sign}(w \cdot x + b)$; **confidence**: $(w \cdot x + b)\gamma$ [p.17]
- d. **OBJECTIVE**: Maximizing the margin: basic: p.18 \rightarrow simplified for linearly separable data: p.23 + **p.30**
- e. Num of support vectors: $d+1$ (for d dimensional data) [p.20]
- f. **Regularization+slack variable (ξ)**: p.25,27
- g. Finding w : Gradient descent: p.30 (cost function+gradient) + p.31(Gradient Descent)
 - i. Stochastic Gradient Descent: p.32
- h. Multiclass SVM: p.40-42

15. Mining Data Streams 1: [15-streams1.pdf](#):

- a. Sampling data from a stream
 - i. Random sample with fixed proportion
 - 1. Naïve approach: sample randomly (p.13) --> caveats: p.14
 - a. Solution: sample users: p.16
 - ii. Random sample with fixed size
 - 1. Reservoir sampling: p.19
 - a. Proof: p.20-21
- b. Queries over sliding windows
 - i. Number of items of type x in the last k elements of the stream
 - 1. Assuming uniformity: p.27
 - 2. **DGIM**: $O(\log^2 N)$ storage+at most 50% error [proof: p.42] (p.28)
 - a. Definition of bucket: p.34
 - b. Rules of representing a stream by buckets: p.35+36(figure)
 - c. **Updating buckets**: p.37+38 --> example: p.39
 - d. Estimating number of 1s in N most recent bits: p.40
 - e. Extensions
 - i. r or r-1 buckets: p.43
 - ii. Stream of positive integers: p.45

16. Mining Data Streams 2: [16-streams2.pdf](#)

- a. **Filtering a data stream**
 - i. Select elements with property x from the stream
 - 1. First cut solution: p.6
 - a. Creates false positives but no false negatives: If the item is in S we surely output it, if not we may still output it
 - b. Probability of m darts, n targets: a target getting hit by at least one dart: $1 - (1-1/n)^m$ [1/n: probability of hitting a target with one dart] $\sim 1 - e^{-m/n}$ = probability of false positives [p.10]
 - ii. **Bloom Filter**:
 - 1. Algorithm [false positive: $(1 - e^{-km/n})^k$: p.12-13 --> optimal k: $n/m \ln(2)$ [p.14]
 - 2. Guarantees no false negatives, use limited memory; 1 big B == k small Bs --> p.15
- b. Counting **distinct** elements (**Flajolet-Martin**)
 - i. Number of distinct elements in the last k elements of the stream
 - 1. Hash N elements to at least $\log_2 N$ bits; $2^{\max_a r(a)}$ with $r(a)$ = position of first 1 counting from the right. [p.20] --> proof [2^r will be around m]: p.23-24
 - a. Debugging $E[2^r]$ --> ∞ : p.25
- c. Estimating **moments**
 - i. K^{th} moment: p.28-29
 - ii. 2nd moment calculation: **AMS** method: p.31-32 --> proof: p.33-34
 - iii. **For all k^{th} moments**: p.35
 - iv. Dealing w Endless streams: p.37 [reservoir sampling]

Table of Contents for CS246 – For Final Exam

- v. Estimate avg./std dev of last k elements
- d. Counting Itemsets: [Finding frequent elements]
 - i. Using DGIM: p.39
 - ii. Exponentially Decaying window: p.41-42; sum over all weights = $1/c$ [p.43] --> example: p.44

17. Advertising [17-advertising.pdf](#)

- a. Perfect matching: all vertices of the graph matched
- b. Maximum matching; matching with largest possible number of matches
- c. Competitive ratio: $\min_{\text{all possible inputs } I} (|M_{\text{greedy}}| / |M_{\text{opt}}|)$ (p.14)
- d. $|M_{\text{greedy}}| / |M_{\text{opt}}| \geq \frac{1}{2}$ [proof: p.15-17]
- e. Cost per thousand impressions (CPM); click-through rates (CTR) [p.20]
- f. Goal of ads: max search engine's revenues: expected revenue per click (Bid * CTR)
- g. Simplified environment: p.33
- h. BALANCE Algorithm: p.35 (pick advertiser with largest unspent budget; ties broken alphabetically)
 - i. Optimal exhausts all budgets of advertisers (p.37)
 - ii. Balance revenue: minimum for $x = y = B/2 \rightarrow$ Min balance revenue: $3B/2 \rightarrow$ competitive ratio = $3/4$
 - 1. Proof: pl.39-40
 - 2. Worst case competitive ratio: 0.63
 - a. Proof: P.42-43, 45-46
 - b. General case: p.48

18. Bandits (Learning through experimentation) [18-bandits.pdf](#)

a. K-armed (ad) bandit (query):

- i. win (reward = 1) with fixed (unknown) prob. μ_a [ad's CTR: we want to estimate this]
- ii. loss (reward = 0) with fixed (unknown) prob. $1 - \mu_a$
- iii. all draws independent given μ_1 to μ_k
- iv. Performance metric: regret: p.13
- v. Allocation strategy: finding out μ_a : p.14
- vi. **Epsilon-Greedy algorithm**: p.18

1. **Confidence Intervals**: choose action with highest upper bound on confidence interval (p.22)
2. **Hoeffding's** inequality for upper bound on average deviation from expected value of μ_a (p.25-26)

▪ **Then:** $P(|\mu - \hat{\mu}_m| \geq b) \leq 2 \exp(-2b^2m) = \delta$

▪ δ ... is the confidence level

▪ To find out the confidence interval b (for a given confidence level δ) we solve:

▪ $2e^{-2b^2m} \leq \delta$ then $-2b^2m \leq \ln(\delta/2)$

▪ So: $b \geq \sqrt{\frac{\ln(\frac{2}{\delta})}{2m}}$

- vii. **Upper Confidence Sampling (UCB1 algorithm)**: (p.27-28) --> performance: $O(R_T/T) \leq k \ln(T)/T$ [p.29]
- viii. A/B Testing: p.35
 1. Thompson Sampling p.38-41 --> p.41: in general