Table of Contents for CS246 – For Final Exam

Created by Pouya Rezazadeh Kalehbasti (pouyar@stanford.edu) – Winter 2020

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

Geometric series

3. Finding similar items: Locality Sensitive Hashing (LSH): 03-lsh.pdf
   a. Naïve approach to finding all pairs of data (x_i, x_j) with distance <= s: $O(N^2)$ (p.8)
   b. **Min-Hash / LSH** steps: p.13+**52** [**downside**: can produce false negatives: pairs of similar items not detected]
      i. **Shingling**: convert document to set of k-shingles then hash the shingles to 4-byte integers (p.16)
         1. **Jaccard** similarity (p.18) for $C_i = S(D_i)$:
            sim(D1, D2) = |C1 ∩ C2|/|C1 ∪ 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)

         2. Jaccard distance: d(C1, C2) = 1 - |C1∩C2|/|C1∪C2|
      ii. **Min-Hashing (suitable for Jaccard similarity):** h(C) s.t. sim(C_1, C_2) is high → P(h(C_1) = h(C_2)) is high + vice versa (p.22) → **P(h(C1)=h(C2))=sim(C1,C2)**
         1. Minhash function for permutation π: $h_\pi(C) = \min_\pi \pi(C)$ [first 1 in permuted column C] → example: p.25
         2. **Pr[*hπ*(C_1) = *hπ*(C_2)] = *sim*(C_1, C_2)** [proof: p.27]
            a. One-pass implementation: p.31
            b. Universal hashing:
               $h_{a,b}(x)$ = ((a.x+b) mod p) mod N [a,b: rnd int; p: prime > N]
      iii. Locality Sensitive Hashing: (p.38)
         1. Divide signature matrix M into b bands of r rows [M = b × r]
         2. Hash all bands to a hash table with k buckets (large k)
         3. Candidate pairs: hash to same bucket for >= 1 bands
         4. Tune b&r to catch most similar pairs but few non-similar pairs
         5. [assuming same bucket == identical bands]
         6. Example calculation of false negatives ($(1-S^r)^b$): p.41 + 47
         7. Example calculation of false positives $1-((1-S^r)^b)$: p.42 + 47
         8. **S curve** (probability of sharing a bucket vs similarity of two sets) $(1-(1-t^r)^b)$: p.48 + 49 (example for b = 20, r = 5)
4. LSH Theory: 04-lsh_theory.pdf
   1. Probability that C_1 and C_2 are candidate pairs: $1 – (1-S^r)^b$ (P.10)
   2. **Distance measures: Jaccard, Cosine, L1 (Manhattan), L2 (Euclidean)** (p.16)
      a. Cosine distance (p.43): d(A,B) = $\theta = \arccos\left(\frac{A.B}{||A||.||B||}\right)$ [range: 0 to π]
      b. Cosine similarity (p.43): 1 – d(A,B) → or → $\cos(\theta) = \frac{A.B}{||A||.||B||}$

3. **Locality-Sensitive (LS) Families**: family H of hash functs (d1, d2, p1, p2)-sensitive [$d(x,y) <= d_1$ → $Pr(h(x)=h(y)) >= p_1$] [$d(x,y) >= d_2$ → $Pr(h(x)=h(y)) <= p_2$] (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:** *(d1, d2, (1-d1/π), (1-d2/π))-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 hi(x) = hi(y)
         → (d1, d2, p1^r, p2^r) (p.24)
   f. **OR-Construction**: e.g. many bands (p.22)
      i. H(x) = h(y) if and only if hi(x) = hi(y) for at least 1 i
         → (d1, d2, 1-(1-p1)^b, 1-(1-p2)^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) → Summary: p.39 **[ideal: get p1 to 1, p2 to 0]**

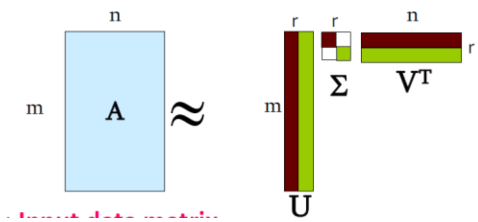
** **False positive and false negative**: P.32 --> (increasing b: reduces false negative+increases false positives, increasing r: reduces false positives+increases false negatives [p.27])

   h. Composing constructions: AND → OR or vice versa (p.29)
      i. **r-way AND followed by b-way OR** (p.29):
         Pr[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[min 1 shared bucket=candidate pair] = $(1-(1-s)^b)^r$
         iii. (3; 4; 5) way OR-AND-OR --> 1 - (1 - (1 - (1 - s)3)4)5 [p.31 exam 2019 solution]

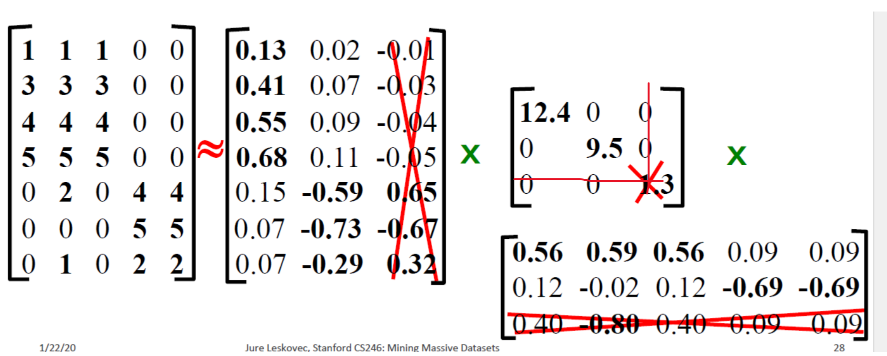
         Example calculating number of hash functions required: P.36

6. Dimensionality Reduction 06-dim_red.pdf
    a. First dimension: direction with greatest variance; second: second largest (p.5)
    b. Rank of matrix A: Number of linearly independent rows of A (p.6)
    c. **SVD**: A (m docs x n terms, input data matrix) ~ 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]

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



i. Real matrix A can be uniquely decomposed as A = U Σ V^T
    1. U, V: column orthonormal → $U^T U = I$
ii. First right singular vector = first row of V^T (p. 22)
iii. Variance (spread on the v_1 axis (element 1,1 from Σ. (p.23)
iv. $U\Sigma$ gives coords of points along projection axis [first column: projection of users on a concept] (p.24)
v. 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)]



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

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

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

7. Recommender Systems 1: 07-recsys1.pdf
    a. X = set of customers; S = set of items
    b. u: X x S → 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 → OFTEN WORKS BETTER THAN USER-USER
            1. Common practice formula: p.33
            2. Pors/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, $P^T = \Sigma V^T$ → $\hat{r}_{xi} = q_i . p_x$ (p.25)
        ii. SVD minimizes sum of squared error (=reconstruction error) → $\boldsymbol{RMSE} = \boldsymbol{1/c} * \sqrt{SSE}$ (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

Created by Pouya Rezazadeh Kalehbasti (pouyar@stanford.edu) – Winter 2020

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 $\mu_1 to\ \mu_k$
        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:** $\mathbf{P}(|\boldsymbol{\mu} - \widehat{\boldsymbol{\mu}_m}| \geq \boldsymbol{b}) \leq \mathbf{2}\ \boldsymbol{exp}(-\mathbf{2}\boldsymbol{b}^2\boldsymbol{m}) = \boldsymbol{\delta}$
                    - **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:** $b \geq \sqrt{\frac{\ln\left(\frac{2}{\delta}\right)}{2\,m}}$
        vii. **Upper Confidence Sampling (UCB1 algorithm)**: (p.27-28) --> performance: $O(R_T/T) <= k \ln(T)/T$ [p.29]
        viii. A/B Testing: p.35
            1. Thompson Sampling p.38-41 --> p.41: in general