Approximating All-Pairs Similarity Search by Rademacher Average

Shiyu Ji
shiyu@cs.ucsb.edu

Abstract

All-pairs similarity search (APSS) has abundant applications in many fields such as collaborative filtering, recommendations, search query suggestions, spam detection, etc. This paper proposes an approximation algorithm for cosine similarity based APSS. We give the upper bounds of the approximation errors for the worst case by using Rademacher average. To the best of our knowledge, we are the first to apply Rademacher average to bound the errors in APSS approximation problems. Our approximation algorithms can be used to efficiently evaluate false positive and false negative of the candidate pairs given by some popular Locality Sensitive Hashing based methods, which can locate the potentially similar pairs and significantly avoid unnecessary comparisons. We also evaluate our algorithms by experiments on real-world data sets to verify that our upper bounds are tight enough for practical use.

1 Introduction

All-pairs similarity search (APSS) has received extensive research interest recently [7, 48, 1, 42]. Collaborative filtering [40], similarity-based recommendations [35], search query suggestions [12], spam and plagiarism detection [14, 31] all find APSS useful in practice. However, it is time consuming to do an APSS since the time complexity grows quadratically with the problem scale [7, 1, 42]. To improve performance, many approximation approaches have been proposed [20, 18, 24, 15, 28]. Hence the computation [7, 17, 48, 1, 42] and approximation [28, 20, 18, 24, 15] of the all-pairs similarities have been popular in research for more than ten years.

One of the most popular methods to approximate APSS is by using Locality Sensitive Hashing (LSH) [28]. The basic idea is to use LSH to map potentially similar objects into the same bucket. Hence there is no need to compare dissimilar pairs, which are unlikely to be mapped into the same bucket. Very often the number of objects in one bucket is much less than the number of total objects. Thus this method can be efficient in practice. However the false positive and false negative of this LSH-based method can be non-negligible. In particular, if the cosine similarity between two vectors is $\cos\theta$, then with probability $1-(1-(1-\theta/\pi)^r)^b$ (where r is the number of rows and b is the number of bands [28]), these two vectors are mapped into the same bucket. Clearly assuming the product of r and b is the fixed signature length, if b is too large, even though the similarity is less than the threshold, the two vectors can still be mapped into one bucket with high probability, i.e., the LSH-based scheme has low precision. Conversely if b is too small, some similar vectors can be lost since they can be mapped into different buckets with high probability, i.e., the recall is low. Thus to tune r and b, one has to evaluate the quality of the candidate pairs in the buckets, e.g., 1) choose a subset of candidate pairs and compute (or approximate) their similarities, and check how many are above the threshold, and 2) choose a subset of non-candidate pairs and compute (or approximate) their similarities and check how many are below the threshold. For a large scale dataset, the number of pairs to be evaluated can be very large. Thus we can use approximation with high precision instead of exact computation for efficiency. This paper investigates how to approximate the similarities precisely and efficiently so that the time needed for evaluation can be greatly reduced.

This paper considers a sampling-based approximation method for cosine similarity based search [41, 48, 1, 42]. Our solution idea is to treat the cosine similarity between two vectors as a true average over random

variables, and then we can use sampled average to approximate the true average. We explain the details as follows. Cosine similarity between two vectors, each of which contains m features, is defined as

$$Sim(u, v) = \frac{1}{||u|| \cdot ||v||} \sum_{i=1}^{m} u_i \cdot v_i,$$

where $|| \cdot ||$ denotes 2-norm and u_i is the i-th component (feature) value. If we normalize each vector to make their norms all equal to $\sqrt{m}$, then cosine similarity is just the true average of the m feature products between two vectors:

$$Sim(u, v) = \frac{1}{m} \sum_{i=1}^{m} u_i \cdot v_i.$$

Suppose m is large, e.g., over 1K. Then it is time consuming to traverse all the m feature pairs and do the multiplications. A straightforward method to approximate the true average is by using the sampled average

$$\widetilde{Sim}(u, v) = \frac{1}{|S|} \sum_{s_i \in S} u_{s_i} \cdot v_{s_i},$$

where S is the set of sampled features. That is, we can choose k out of m features at random, and compute the average of the k sampled feature products as an approximation of the cosine similarity. Thanks to the recent advance in statistical learning theory (especially the results of Rademacher average), there are tight error bounds for this sampling method.

The approximation error of each our algorithm can be upper bounded by using Rademacher average [6, 33, 5]. We can treat the similarity approximation as a learning problem over large scale data set. In the analysis, we want to upper bound the approximation error for the worst case, i.e., to bound the maximum error $\overline{E}$ among the pairs of our observation:

$$\overline{E} = \max_{(u,v) \in P} |\widetilde{Sim}(u, v) - Sim(u, v)|.$$

We can use LSH-based scheme to choose the set of pairs P to be observed. Rademacher average does an excellent job on how to bound such maximum error [38, 39]. The key result we use in this paper is that with probability at least $1 - \delta$,

$$\overline{E} \leq \min \left\{ c\sqrt{\frac{\log p + \log(1/\delta)}{2k}}, R + c \left(1 + \sqrt{\frac{8}{k} \log \frac{2}{\delta}} + \sqrt{\frac{8}{k} \log \frac{2}{\delta} + \frac{8R}{c}} \right) \sqrt{\frac{\log \frac{8}{\delta}}{2k}} \right\},$$

where $p = |P|$ (the observation size), R denotes the Rademacher average of the data set, n is the number of vectors, k is the number of samples, and c is the number of features for cosine similarity. Often by using LSH our observation can be confined within a relatively small set of pairs, i.e., $p << n^2$. It is also possible to upper bound the Rademacher average by the state-of-art results [3, 38, 39]. Thus we can find an upper bound of the worst errors for our approximation algorithms.

Our Contributions. We are the first to give the approximation algorithms for APSS that can achieve upper bounds on maximum errors for the worst case, and to show the upper bounds by using Rademacher average. Our approximation algorithms can be used to help reduce the time used for evaluation in the popular LSH-based dissimilarity detection. We also evaluate our algorithms by using real-world data sets.

The rest of this paper is organized as follows. Section 2 discusses the related works. Section 3 introduces the problem settings and reviews some technical background. Section 4 approximates the cosine similarity based APSS and also gives the analysis on the upper bound of maximum errors. Section 5 introduces the approximation combining LSH and sampling and gives the analysis on the complexity and performance. Section 6 evaluates our algorithms and presents the experimental results. Section 7 concludes this paper and discusses some possible future works.

2 Related Works

All-pairs similarity search (APSS) is recently a popular research topic in the community of information retrieval [7, 48, 1, 42]. Given a big set of objects, the goal of APSS is to efficiently compute (or approximate) the similarities between each pair of objects. Many similarity measures have been proposed [41], e.g., Cosine Similarity [43], SimRank [25], Jaccard Index [21], Metric Distance [41], Pearson Correlation [8], etc. For similarity search, Cosine Similarity and SimRank are very popular ([43, 48, 1, 42] for Cosine, and [29, 19, 26, 49] for SimRank). A major challenge of APSS is the large volume of computation: given n objects, without any assumption on the similarity distribution (e.g., the similarity matrix is sparse), the time complexity is at least $O(n^2)$. To compute more efficiently, the state-of-art research works often use parallelism [13, 22] and dissimilarity detection [16, 4, 28, 1, 42]. One popular method to eliminate dissimilar pairs is to hash vectors into buckets by Locality Sensitive Hashing (LSH) [28], and the candidate pairs are only between the vectors in the same bucket. However, since such methods can introduce non-negligible false positive and false negative [2, 44], further verifications on the candidate pairs (or missing potential pairs) are needed. Partition-based Similarity Search (PSS) uses partitions to cluster the candidate pairs [42]. Very often some partitions are still large, and hence further approximations on the similarities may be needed to avoid redundant computation. If only approximations are needed, how to efficiently sample with negligible error for similarity search is another research focus [20, 18, 24, 15]. This paper focuses on the approximation problem.

We may treat the similarity approximation as a learning problem over large scale data, and we need to upper bound the learning error for the worst case. In statistical learning theory, such upper bounds are usually called risk bounds [45]. There are two classical methods to compute the risk bounds: by VapnikChervonenkis (VC) dimension [46, 47, 45] and by Rademacher average [33, 6, 5]. Many approximation algorithms that use these two techniques have been proposed [36, 37, 38, 39]. Riondato and Kornaropoulos [36, 37] proposed algorithms that use VC dimension to upper bound the sample size that is sufficient to approximate the betweenness centralities [11] of all nodes in a graph with guaranteed error bounds. One limitation of the VC-based algorithms is that the upper bounds of some characteristic quantities (e.g., the maximum length of any shortest path) are needed [36, 37, 39]. But such bounds are not always available. One year later, Riondato and Upfal used Rademacher average to approximate the frequent itemsets [38] and betweenness centralities [39]. They also found that by using Rademacher average, we can avoid the aforementioned limitation of VC-based solutions. It has been proved that Rademacher average can be applied to various approximation problems, which are out of the scope of classical learning framework. This paper basically follows this idea. To the best of our knowledge, there is no research work connecting similarity approximation with Rademacher average. We attempt to fill this void.

3 Problem Formulation and Preliminaries

3.1 Cosine Similarity

We consider the cosine similarity based all-pairs similarity search. Suppose there are n vectors (each vector can represent a user profile or a web page). Each vector contains m non-negative features. Define the cosine similarity between two vectors u and v as

$$Sim(u, v) = \frac{1}{||u|| \cdot ||v||} \sum_{i=1}^{m} u_i \cdot v_i,$$

where u_i, v_i denotes the i-th feature value of u, v. For simplicity, we assume all the vectors are adjusted with the same norm: $||v|| = \sqrt{m}$ for every v in the n vectors. Then the equation above can be simplified as

$$Sim(u, v) = \frac{1}{m} \sum_{i=1}^{m} u_i \cdot v_i.$$

That is, the similarity is defined as the average on the corresponding feature products between the vectors.

To do the all-pairs similarity search (APSS), we need to compute the similarity between each pair of vectors. Since there are $n(n-1)$ pairs, and for each pair we need m times of multiplication, the total complexity of a naiive algorithm is $O(n^2m)$. Fortunately, there are many methods to detect dissimilar pairs (two vectors without sharing any feature) [1, 42, 30], which can save a lot of computation. One typical detection method is based on Locality Sensitive Hashing (LSH) [28]. We can construct a LSH such that for two vectors u and v, if their cosine similarity is larger than $\cos\theta$, then with probability at least $1-\theta/\pi$ we can hash u and v into one bucket [28]. The vectors in one bucket are likely to be highly similar. Thus the number of vector pairs to be compared is greatly reduced. For the state-of-art works, to compute all the pairs, the complexity can be lowered to $O(nkm)$, where k is much less than n. However, to the best of our knowledge, there were few discussions on the size of features m. If we can only consider a part of the m features without significantly deteriorating the accuracy, then the total computation time for APSS can be lowered significantly (note that nk is still large for very big dataset). Moreover, the popular LSH based method can introduce false positive. Note that if we assume the cosine similarity ranges from 0 to 1, then the angle θ between any two vectors is between 0 and $\pi/2$. Hence even for two totally dissimilar vectors $u \cdot v = 0$, they still have probability at least $1-(1-0.5^r)^b$ (where r and b are the number of rows and bands that from the LSH signature [28]) to be hashed into one bucket and thus will be falsely believed to be similar. Hence we need further verifications on the candidate pairs. Since there could still be a lot of vectors in one bucket, in practice approximations on the cosine similarities are interesting to research.

3.2 Rademacher Average

This section will review some key results related to Rademacher Average. Suppose we want to compute the cosine similarities between a fixed vector u and other vectors v_1, v_2, $\cdots$, v_n. We take the m features as the sample space S:

$$S = \{s_1, \cdots, s_k\} \subseteq D$$

where k is the number of samples we take, and D is the feature space: $D = \{1, 2, \cdots, m\}$. Let F be a collection of functions from the features D to the interval $[0, m]$. For each function $f \in F$, define the *true average* $A_D(f)$ and *sampled average* $A_S(f)$ as follows:

$$A_D(f) = \frac{1}{m}\sum_{i=1}^{m} f(i), \quad A_S(f) = \frac{1}{k}\sum_{i=1}^{k} f(s_i).$$

Define the *uniform deviation* [34] of F given S as

$$U_S(F) = \sup_{f\in F}[A_S(f) - A_D(f)].$$

Note that if F is a finite set (in this paper we will see this is true), supreme can be replaced by maximum:

$$U_S(F) = \max_{f\in F}[A_S(f) - A_D(f)].$$

For the case when F is finite, we have the upper bound of the approximation error in the worst case as follows.

Theorem 1. If F is finite, then for any $\delta > 0$, with probability at least $1-\delta$,

$$\sup_{f\in F} |A_S(f) - A_D(f)| \leq m\sqrt{\frac{\log(2|F|) + \log(1/\delta)}{2k}}.$$

We leave the proof of Theorem 1 to the Appendix.

However if F is not necessarily infinite, the above bound cannot apply. We can use Rademacher average to deal with the general case. Define the *Rademacher average* [33, 6, 34] of F given S as

$$R_S(F) = \mathbb{E}_\sigma \left[\sup_{f \in F} \frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i) \right],$$

where each σ_i is a random variable uniformly distributed over $\{-1, 1\}$, and the mean $\mathbb{E}_\sigma$ takes randomness over all the σ_i's conditionally on S. Again, one can replace supreme by maximum if F is finite.

The main motivation of our algorithm is that the difference between true average and sampled average is upper bounded by the Rademacher average as follows. We leave the proofs to the Appendix.

Theorem 2. Let F be a collection of functions f mapping D to $[0, m]$. With probability at least $1 - \delta$, we have

$$\sup_{f \in F} |A_S(f) - A_D(f)| \leq R_S(F) + \left(m + m\sqrt{\frac{8}{k} \log \frac{2}{\delta}} + m\sqrt{\frac{8}{k} \log \frac{2}{\delta} + \frac{8R_S(F)}{m}} \right) \sqrt{\frac{\log \frac{8}{\delta}}{2k}}.$$

The remaining item we need to upper bound is $R_S(F)$. Our result is similar to Massart's lemma [3].

Theorem 3.

$$R_S(F) \leq \frac{\ell}{k} \sqrt{8 \log |F|},$$

where $\ell^2 = \sup_{f \in F} \sum_{i=1}^{k} f(s_i)^2$.

By the above two theorems, we can bound our approximation error even for the worst case.

4 Approximating Cosine Similarities

This section proposes our approximation algorithm and its analysis. The approximation algorithm takes as input a collection of n vectors V, a collection of p candidate pairs P, and two parameters (ϵ, δ) whose values are between 0 and 1. The candidate pairs P can be given by a dissimilarity detection algorithm, e.g., LSH based method [28], Feature Scoring [16], Diamond Sampling [4]. The algorithm outputs a set $C = \{\tilde{S}(u, v) : u, v \in V, (u, v) \in P\}$, where $\tilde{S}(u, v)$ is the (ϵ, δ)- approximation of cosine similarity between u and v, i.e., with probability at least $1 - \delta$, the worst approximation error in C is at most ϵ. Each vector in V contains some features. In practice we often consider the features in different weights, i.e., each vector may have several key features, while other features are minor. Since the number of key features are usually small, we can store the key features of a vector into an associated file. When estimating the cosine similarity between two vectors, we first take the overlapped key features of the two vectors and calculate the products on the overlapped features. The sum of these products give the major contribution of the cosine similarity. Note that the idea of many existing cosine similarity sampling algorithms [16, 4] is also based on this observation. Then we only need to estimate the remaining features outside of the overlapped key ones. Suppose we have m of them. We take these m features as the sample space $D = \{1, \cdots, m\}$. For each feature $s \in D$, let $f_{u,v}(s) = u_s \cdot v_s$, where u_s is the s-th feature value of the vector u. Let $F = \{f_{u,v} : u, v \in V, (u, v) \in P\}$. Thus $|F| = p$. It is clear that the true average of each $f_{u,v}$ equals to the cosine similarity between u and v. Given the sampled features of size k, the upper bound of $R_S(F)$ is

$$R_S(F) \leq \frac{\ell}{k} \sqrt{8 \log p},$$

where $\ell = \sqrt{\max_{f \in F} \sum_{i=1}^{k} f(s_i)^2}$. Since F is finite, we can replace supreme by maximum.

Algorithm 1 Cosine Similarity Approximation

Input: vectors V ($|V| = n$); key features $\mathbf{f}_i$ for each vector $i \in V$; candidate pairs P ($|P| = p$); all vectors' features U; weights of the features $W = \{w_1, \cdots, w_{|U|}\}$; $\epsilon \in (0, 1)$; $\delta \in (0, 1)$.
Output: $Sim(u, v)$ for each $(u, v) \in P$.

```
m ← |U|; k ← 0; S ← ∅;
S(u, v) ← Σ_{i∈f_u∩f_v} u[i] · v[i] for each (u, v) ∈ P;
while k < m do
    k' ← next-sample-size(k);
    for i from k to k' − 1 do
        Uniformly sample a feature s from U \ S;
        for each (u, v) ∈ P do
            if s ∉ f_u ∩ f_v then
                S(u, v) ← S(u, v) + u[s] · v[s];
            end if
        end for
        S ← S ∪ {s};
    end for
    Δ ← min{ m√((log p + log(1/δ))/2k), (ℓ√(8 log p))/k + (m + m√((8/k) log(2/δ)) + m√((8/k) log(2/δ) + (16ℓ√(2 log p))/(mk))) √((log(8/δ))/2k) };
    if Δ ≤ ε then
        break the while loop;
    end if
end while
S ← {S(u, v)/k : u, v ∈ V, (u, v) ∈ P}.
return S.
```

4.1 The Algorithm

The approximation algorithm works in an iterative mode. For each iteration, we sample some new features s_i's among D and aggregate the feature products given by $f_{u,v}(s_i)$ for each $f_{u,v}$ in F. Then we compute the error upper bound:

$$\Delta = \min\left\{ m\sqrt{\frac{\log p + \log(1/\delta)}{2k}}, \frac{\ell\sqrt{8\log p}}{k} + \left(m + m\sqrt{\frac{8}{k}\log\frac{2}{\delta}} + m\sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{16\ell\sqrt{2\log p}}{mk}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}} \right\},$$

where $\ell = \sqrt{\max_{f\in F}\sum_{i=1}^{k} f(s_i)^2}$, and k is the number of aggregated samples. If $\Delta \le \epsilon$, then we stop and return the averages $\frac{1}{k}\sum_{s_i\in S} f_{u,v}(s_i)$ for each pair u, v. Otherwise, we continue to the next round, where we will sample more features. If $\Delta \le \epsilon$ can never be satisfied, at the end we will sample all the m features and return the averages as the exact solution. Algorithm 1 gives the pseudocode of our solution. We use progressive approximation, i.e., for each round, the next sample size k' is larger than the previous size k. The algorithm stops once the upper bound ϵ can be achieved. Since we assume the number of features m is large, e.g., in word2vec m is around 500 to 1000, and the number of key features of each vector is small due to the sparsity, e.g., usually less than 10, in this algorithm we take m as the size of sample space.

It is clear to verify the correctness of our algorithm by Theorem 1, Theorem 2 and Theorem 3.

4.2 Improving the Upper Bound

Note that in the upper bound Δ of Algorithm 1, m is a dominating variable: it is proportional to Δ. Usually m can be quite large, e.g., from 100 to 10K. Thus it is worth investigating how to mitigate the negative effect of large m. The idea comes from Theorem 2, which assumes the range of each function f is $[0, m]$. However

in reality, most feature products $u[s] \cdot v[s]$ are close to zero due to sparsity, and its maximum value shall be much less than m. In practice, usually we can assume there is a good estimation (much less than m) on the maximum feature product based on historic experience. To further lower the upper bound Δ, a variant of Algorithm 1 may use the bound as follows:

$$\hat{m} = \max_{i \in [1,\cdots,m],(u,v) \in P} u[i] \cdot v[i],$$

where $[1, \cdots, m]$ denotes all the m features, and V is the set of all vectors. Intuitively, $\hat{m}$ is the maximum value of all the feature products, and we assume we have a good estimation of $\hat{m}$, which can be chosen by sampling from historical data. Then we may choose the upper bound as follows:

$$\Delta = \min \left\{ \hat{m}\sqrt{\frac{\log p + \log(1/\delta)}{2k}}, \frac{\ell\sqrt{8 \log p}}{k} + \hat{m}\left(1 + \sqrt{\frac{8}{k}\log\frac{2}{\delta}} + \sqrt{\frac{8}{k}\log\frac{2}{\delta} + \frac{16\ell\sqrt{2\log p}}{\hat{m}k}}\right)\sqrt{\frac{\log\frac{8}{\delta}}{2k}} \right\}.$$

We use experiments (in Section 6) to show that the improvement of the variant above is significant.

5 Similarity Detection Given Threshold and Top-K Approximation

In this section we discuss the possible methods to 1) detect potentially similar pairs given a threshold τ, and 2) approximate the top k most similar vector pairs. We compare four methods given as follows:

- **LSH + Sampling**. We use LSH to map potentially similar pairs into the same bucket, and then for the pairs in one bucket, we use our sampling-based (ϵ, δ)-approximation algorithm to validate their similarity, and find the top K most similar pairs if needed.
- **LSH + Exact Validation**. We use LSH to group potentially similar pairs, and then use exact computation to obtain the similarity between each pair in one bucket, and find the top K if needed.
- **Pure Sampling**. We directly use our sampling-based (ϵ, δ)-approximation algorithm on all the dataset, and find the top K if needed.
- **Pure LSH**. We only use LSH to map vector pairs into buckets, without further validation, simply assuming the pairs in one bucket have similarities larger than the threshold. Note that we cannot use this method to approximate the top K, since there can be more than K candidate pairs in the buckets.

For each method, we consider complexity and accuracy.

- *Complexity.* We consider time complexity (i.e., how much running time is needed) and space complexity (i.e., how much memory is needed).
- *Accuracy.* We consider precision (i.e., how many of the candidate pairs given by the algorithm are correct) and recall (i.e., how many of the correct pairs are identified by the algorithm).
- *Error Bound.* In the previous sections we have discussed the error upper bound for the worst case, i.e., LSH + Sampling gives $O(\sqrt{\log p})$ bound, while pure sampling on the entire dataset gives $O(\sqrt{\log n^2})$ bound. Often the size p of candidate pairs given by LSH is much less than the number of all pairs n^2. Hence LSH + Sampling outperforms with tighter bound and thus fewer samples.

Methods	LSH + Sampling	LSH + Exact Validation	Pure Sampling	Pure LSH
Time	$O(nbr + kp)$	$O(nbr + mp)$	$O(kn^2)$	$O(nbr)$
Space	$O(nbr + p)$	$O(nbr + p)$	$O(n^2)$	$O(nbr)$
Precision	$\geq (1 - P_\epsilon^\tau \cdot P_\theta^{r,b})(1 - \delta)$	1	$\geq (1 - P_\epsilon^\tau)(1 - \delta)$	$\geq 1 - (1 - 0.5^r)^b$
Recall	$\geq P_\theta^{r,b}(1 - P_\epsilon^\tau)(1 - \delta)$	$\geq P_\theta^{r,b}$	$\geq (1 - P_\epsilon^\tau)(1 - \delta)$	$\geq P_\theta^{r,b}$

Table 1: Complexity and accuracy analysis of our methods to detect similar pairs given threshold $\tau = \cos\theta$.

5.1 Similarity Detection Given Threshold

Table 1 gives the complexity and accuracy of the three above methods. We first explain the used notations as follows:

- $P_\theta^{r,b}$ is the *lower bound* of the probability that a pair of vectors u, v s.t. $u \cdot v \geq \cos\theta$, are mapped into the same bucket by LSH. In particular,

$$P_\theta^{r,b} = 1 - \left(1 - \left(1 - \frac{\theta}{\pi}\right)^r\right)^b,$$

  where r is the number of rows, b is the number of bands, and $r \cdot b$ is the signature length of LSH. Note that $\tau = \cos\theta$. $P_\theta^{r,b}$ is also the *upper bound* of the probability that a pair of vectors with similarity less than τ are mapped into the same bucket (which gives false positive).

- P_ϵ^τ is the probability that the cosine similarity of any vector pair is within the interval $\tau \pm \epsilon$.

All the other notations are the same as in the previous sections: n is the number of vectors; m is the number of features; k is the number of sampled features; p is the size of the candidate pairs P; ϵ and δ are the input parameters of our sampling algorithms.

We explain some bounds in Table 1.

- For LSH + Sampling, the only pairs that can impact the precision are the pairs that have cosine similarities within the interval $(\tau - \epsilon, \tau)$. These pairs close to the τ-boundary have probability at most $P_\theta^{r,b}$ to be mapped into the same bucket, and then might be falsely accepted by our (ϵ, δ)-approximation. With probability at least $1 - \delta$, we can assume our approximation algorithm will not accept any pair with similarity less than $\tau - \epsilon$. Hence the lower bound of precision follows.

- For LSH + Sampling, with probability at least $1 - \delta$, we assume our approximation makes no mistake on the pairs with similarity larger than $\tau + \epsilon$. The ratio of these safe pairs in all the correct pairs is at least $1 - P_\epsilon^\tau$. Hence the lower bound of recall follows.

Other bounds are clear to verify. With small enough errors ϵ and δ, LSH + Sampling is competitive with high accuracy and less time complexity.

5.2 Top-k Similarity Approximation

Table 2 gives the complexity and accuracy of our methods to approximate top K. The notations and results are similar to Table 1, except

- $P_{(K)}^{r,b}$ is the lower bound of the probability that any top-K most similar pair of vectors are mapped into the same bucket.

- $P_\epsilon^{(K)}$ is the probability that the similarity difference between any pair and the K-th most similar pair is within ϵ.

The analysis is similar to the case given a threshold. Note that since the algorithms always select K candidate pairs as the result, precision equals to recall. The $p \log p$ term is due to the sorting algorithm used to find the top K.

Methods	LSH + Sampling	LSH + Exact Validation	Pure Sampling
Time	$O(nbr + kp + p \log p)$	$O(nbr + mp + p \log p)$	$O(kn^2)$
Space	$O(nbr + p)$	$O(nbr + p)$	$O(n^2)$
Precision	$\geq P_{(K)}^{r,b}(1 - P_{\epsilon}^{(K)})(1 - \delta)$	$\geq P_{(K)}^{r,b}$	$\geq (1 - P_{\epsilon}^{(K)})(1 - \delta)$
Recall	$\geq P_{(K)}^{r,b}(1 - P_{\epsilon}^{(K)})(1 - \delta)$	$\geq P_{(K)}^{r,b}$	$\geq (1 - P_{\epsilon}^{(K)})(1 - \delta)$

Table 2: Complexity and accuracy analysis of our methods to approximate top K.

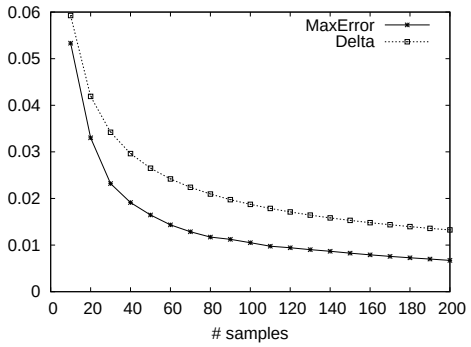

Figure 1: MaxErrors and upper bounds for cosine similarity for `Cit-HepPh`.

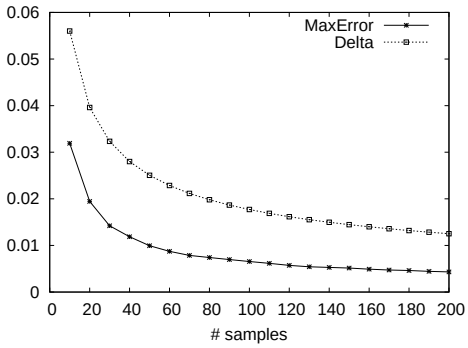

Figure 2: MaxError and upper bounds for cosine similarity for `Email-Enron`.

6 Evaluation

In this section we evaluate our algorithms. We use both synthesized and real-world data sets.

6.1 Setup

We implemented our algorithms by Python 3.4.3 and ran the programs on a Fedora 20 (Linux) cluster. Every measurement was averaged over 100 test cases, i.e., we repeated our algorithms for 100 times and took the averaged results.

We use real-world data sets to evaluate our algorithms. In particular, we use `Cit-HepPh` and `Email-Enron` from Stanford Large Network Dataset Collection (SNAP) [27]. `Cit-HepPh` has 34551 vectors, each of which has 34551 features. `Email-Enron` has 36697 vectors, each of which has 36697 features.

6.2 Approximation Errors and Sample Sizes

Figure 1 and Figure 2 give the maximum approximation errors and upper bound given by Algorithm 1 for the real-world data sets. Since there are more features (more than 30K), the upper bounds cannot be as tight as before. However, for `Cit-HepPh`, only 200 samples can guarantee an upper bound 0.015, which is quite small compared to the number of features.

6.3 Running Time

Figure 3 gives the running time of our algorithm on the real-world data sets given the sample sizes. Here we did not use any parallelism or detection techniques to implement our algorithms. Note that in APSS we need to compute for each pair of objects. The number of such pairs grows quadratically. As a result when the sample size is large, the computation can take a few hours. A good news is that since the computation of each pair is independent from each other, the approximation algorithms can be easily distributed over multiple cores.

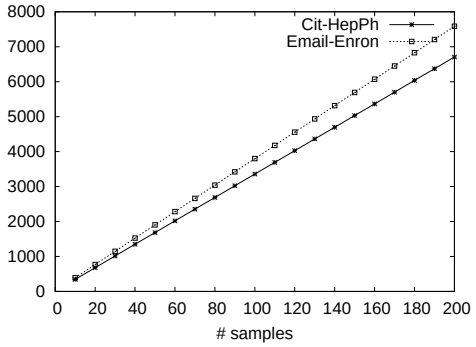

Figure 3: Running time (ms) of Algorithm 1 given sample sizes.

7 Conclusion and Future Research

In this paper we have given two approximation algorithms on cosine similarity for all-pairs similarity search. A rigorous analysis using Rademacher average has been presented. As an important application, we have discussed how to use our approximations to efficiently tune the parameters and improve the performance of the popular LSH-based dissimilarity detection. We also have evaluated their performance by experiments on both synthesized and real-world data sets. The results show that our algorithms can precisely approximate the similarities with acceptable time complexity and error probability.

We note that the upper bounds in this paper still have some space to improve. In the experiments our upper bounds are roughly 5 to 10 times the maximum absolute errors. In the proofs it is unclear whether the bounds we have used are the best possible. Hence the algorithms could be improved. Also it can be an interesting topic how to more efficiently use parallelism in APSS samplings.

References

[1] Maha Alabduljalil, Xun Tang, and Tao Yang. Cache-conscious performance optimization for similarity search. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 713–722. ACM, 2013.

[2] Maha Ahmed Alabduljalil, Xun Tang, and Tao Yang. Optimizing parallel algorithms for all pairs similarity search. In *Proceedings of the sixth ACM international conference on Web search and data mining*, pages 203–212. ACM, 2013.

[3] Davide Anguita, Alessandro Ghio, Luca Oneto, and Sandro Ridella. A deep connection between the vapnik–chervonenkis entropy and the rademacher complexity. *IEEE transactions on neural networks and learning systems*, 25(12):2202–2211, 2014.

[4] Grey Ballard, Tamara G Kolda, Ali Pinar, and C Seshadhri. Diamond sampling for approximate maximum all-pairs dot-product (mad) search. In *Data Mining (ICDM), 2015 IEEE International Conference on*, pages 11–20. IEEE, 2015.

[5] Peter L Bartlett, Olivier Bousquet, and Shahar Mendelson. Local rademacher complexities. *Annals of Statistics*, pages 1497–1537, 2005.

[6] Peter L Bartlett and Shahar Mendelson. Rademacher and gaussian complexities: Risk bounds and structural results. *Journal of Machine Learning Research*, 3(Nov):463–482, 2002.

[7] Roberto J Bayardo, Yiming Ma, and Ramakrishnan Srikant. Scaling up all pairs similarity search. In *Proceedings of the 16th international conference on World Wide Web*, pages 131–140. ACM, 2007.

[8] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. Pearson correlation coefficient. In *Noise reduction in speech processing*, pages 1–4. Springer, 2009.

[9] S Boucheron, G Lugosi, and Pascal Massart. A sharp concentration inequality with applications. *Random Structures and Algorithms*, 1999.

[10] Olivier Bousquet, Stéphane Boucheron, and Gábor Lugosi. Introduction to statistical learning theory. In *Advanced lectures on machine learning*, pages 169–207. Springer, 2004.

[11] Ulrik Brandes. A faster algorithm for betweenness centrality*. *Journal of mathematical sociology*, 25(2):163–177, 2001.

[12] Huanhuan Cao, Daxin Jiang, Jian Pei, Qi He, Zhen Liao, Enhong Chen, and Hang Li. Context-aware query suggestion by mining click-through and session data. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 875–883. ACM, 2008.

[13] Liangliang Cao, Brian Cho, Hyun Duk Kim, Zhen Li, Min-Hsuan Tsai, and Indranil Gupta. Delta-simrank computing on mapreduce. In *Proceedings of the 1st International Workshop on Big Data, Streams and Heterogeneous Source Mining: Algorithms, Systems, Programming Models and Applications*, pages 28–35. ACM, 2012.

[14] Carlos Castillo, Debora Donato, Aristides Gionis, Vanessa Murdock, and Fabrizio Silvestri. Know your neighbors: Web spam detection using the web topology. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 423–430. ACM, 2007.

[15] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

[16] Edith Cohen and David D Lewis. Approximating matrix multiplication for pattern recognition tasks. *Journal of Algorithms*, 30(2):211–252, 1999.

[17] Xin Dong, Alon Halevy, Jayant Madhavan, Ema Nemes, and Jun Zhang. Similarity search for web services. In *Proceedings of the Thirtieth international conference on Very large data bases-Volume 30*, pages 372–383. VLDB Endowment, 2004.

[18] Ronald Fagin, Ravi Kumar, and Dandapani Sivakumar. Efficient similarity search and classification via rank aggregation. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, pages 301–312. ACM, 2003.

[19] Yasuhiro Fujiwara, Makoto Nakatsuji, Hiroaki Shiokawa, and Makoto Onizuka. Efficient search algorithm for simrank. In *Data Engineering (ICDE), 2013 IEEE 29th International Conference on*, pages 589–600. IEEE, 2013.

[20] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. *VLDB*, 99(6):518–529, 1999.

[21] Lieve Hamers, Yves Hemeryck, Guido Herweyers, Marc Janssen, Hans Keters, Ronald Rousseau, and André Vanhoutte. Similarity measures in scientometric research: the jaccard index versus salton's cosine formula. *Information Processing & Management*, 25(3):315–318, 1989.

[22] Guoming He, Haijun Feng, Cuiping Li, and Hong Chen. Parallel simrank computation on large graphs with iterative aggregation. In *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 543–552. ACM, 2010.

[23] Wassily Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American statistical association*, 58(301):13–30, 1963.

[24] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613. ACM, 1998.

[25] Glen Jeh and Jennifer Widom. Simrank: a measure of structural-context similarity. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 538–543. ACM, 2002.

[26] Mitsuru Kusumoto, Takanori Maehara, and Ken-ichi Kawarabayashi. Scalable similarity search for simrank. In *Proceedings of the 2014 ACM SIGMOD international conference on Management of data*, pages 325–336. ACM, 2014.

[27] Jure Leskovec and Andrej Krevl. Snap datasets: Stanford large network dataset collection. *Stanford University*, 2015.

[28] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. *Mining of massive datasets*. Cambridge University Press, 2014.

[29] Cuiping Li, Jiawei Han, Guoming He, Xin Jin, Yizhou Sun, Yintao Yu, and Tianyi Wu. Fast computation of simrank for static and dynamic information networks. In *Proceedings of the 13th International Conference on Extending Database Technology*, pages 465–476. ACM, 2010.

[30] Jimmy Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 155–162. ACM, 2009.

[31] Chao Liu, Chen Chen, Jiawei Han, and Philip S Yu. Gplag: detection of software plagiarism by program dependence graph analysis. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 872–881. ACM, 2006.

[32] Colin McDiarmid. On the method of bounded differences. *Surveys in combinatorics*, 141(1):148–188, 1989.

[33] Mehryar Mohri and Afshin Rostamizadeh. Rademacher complexity bounds for non-iid processes. In *Advances in Neural Information Processing Systems*, pages 1097–1104, 2009.

[34] Luca Oneto, Alessandro Ghio, Davide Anguita, and Sandro Ridella. An improved analysis of the rademacher data-dependent bound using its self bounding property. *Neural Networks*, 44:107–111, 2013.

[35] Paul Resnick and Hal R Varian. Recommender systems. *Communications of the ACM*, 40(3):56–58, 1997.

[36] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. In *Proceedings of the 7th ACM international conference on Web search and data mining*, pages 413–422. ACM, 2014.

[37] Matteo Riondato and Evgenios M Kornaropoulos. Fast approximation of betweenness centrality through sampling. *Data Mining and Knowledge Discovery*, 30(2):438–475, 2016.

[38] Matteo Riondato and Eli Upfal. Mining frequent itemsets through progressive sampling with rademacher averages. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1005–1014. ACM, 2015.

[39] Matteo Riondato and Eli Upfal. Abra: Approximating betweenness centrality in static and dynamic graphs with rademacher averages. *arXiv preprint arXiv:1602.05866*, 2016.

[40] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web*, pages 285–295. ACM, 2001.

[41] Alexander Strehl, Joydeep Ghosh, and Raymond Mooney. Impact of similarity measures on web-page clustering. In *Workshop on Artificial Intelligence for Web Search (AAAI 2000)*, pages 58–64, 2000.

[42] Xun Tang, Maha Alabduljalil, Xin Jin, and Tao Yang. Load balancing for partition-based similarity search. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*, pages 193–202. ACM, 2014.

[43] Sandeep Tata and Jignesh M Patel. Estimating the selectivity of tf-idf based cosine similarity predicates. *ACM Sigmod Record*, 36(2):7–12, 2007.

[44] Ferhan Ture, Tamer Elsayed, and Jimmy Lin. No free lunch: brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, pages 943–952. ACM, 2011.

[45] Vladimir Vapnik. *The nature of statistical learning theory*. Springer Science & Business Media, 2013.

[46] Vladimir Vapnik, Esther Levin, and Yann Le Cun. Measuring the vc-dimension of a learning machine. *Neural Computation*, 6(5):851–876, 1994.

[47] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[48] Zhihua Xia, Xinhui Wang, Xingming Sun, and Qian Wang. A secure and dynamic multi-keyword ranked search scheme over encrypted cloud data. *IEEE Transactions on Parallel and Distributed Systems*, 27(2):340–352, 2016.

[49] Weiren Yu and Julie Ann McCann. High quality graph-based similarity search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 83–92. ACM, 2015.

8 Appendix

8.1 Proof of Theorem 1

First, for any $\epsilon > 0$,

$$\begin{aligned}\Pr[U_S(F) > \epsilon] &= \Pr[\exists f \in F, A_S(f) - A_D(f) > \epsilon] \\ &\leq \sum_{f \in F} \Pr[A_S(f) - A_D(f) > \epsilon] \\ &= \sum_{f \in F} \Pr[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - A_D(f) > \epsilon].\end{aligned}$$

By Hoeffding's inequality [10] we have

$$\Pr[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - A_D(f) > \epsilon] \leq \exp\left(-\frac{2k\epsilon^2}{m^2}\right).$$

Hence

$$\Pr[U_S(F) > \epsilon] \leq |F| \exp\left(-\frac{2k\epsilon^2}{m^2}\right).$$

Similarly one can show that

$$\Pr[\sup_{f \in F}[A_D(f) - A_S(f)] > \epsilon] \leq |F| \exp\left(-\frac{2k\epsilon^2}{m^2}\right).$$

Thus putting the two cases together,

$$\Pr[\sup_{f \in F} |A_S(f) - A_D(f)| > \epsilon] \leq 2|F| \exp\left(-\frac{2k\epsilon^2}{m^2}\right).$$

Equivalently for any $\delta > 0$, with probability at least $1 - \delta$,

$$\sup_{f \in F} |A_S(f) - A_D(f)| \leq m\sqrt{\frac{\log(2|F|) + \log(1/\delta)}{2k}}.$$

We have obtained the desired upper bound in Theorem 1.

8.2 Proof of Theorem 2

In this section, we show our main result (Theorem 2). We start from the definition of self bounding function [34].

Definition 1. *Let s_1, s_2, $\cdots$, s_k be independent random variables taking values from a set D. A function $f : D^k \to [0, +\infty]$ is a self bounding function if there exists a constant c and a function $g : D^{k-1} \to \mathbb{R}$ such that for any s_1, $\cdots$, s_{j-1}, s_{j+1}, $\cdots$, $s_k \in D$, the following conditions hold:*

$$0 \leq f(s_1, \cdots, s_k) - g(s_1, \cdots, s_{j-1}, s_{j+1}, \cdots, s_k) \leq c,$$

$$\sum_{j=1}^{k} [f(s_1, \cdots, s_k) - g(s_1, \cdots, s_{j-1}, s_{j+1}, \cdots, s_k)] \leq f(s_1, \cdots, s_k).$$

The following concentration inequality can be achieved for self bounding functions [9].

Lemma 1. *[9] If a function $Z = f(s_1, \cdots, s_k)$ is a self bounding function with constant c, then for $t \leq \mathbb{E}Z$,*

$$\Pr[\mathbb{E}Z - Z \geq t] \leq \exp\left(-\frac{t^2}{2c\mathbb{E}Z}\right).$$

For $t > \mathbb{E}Z$, the left probability is zero trivially. Here we take randomness over s_1, s_2, $\cdots$, s_k.

By using the above lemma, we can show a similar inequality for Rademacher average.

Lemma 2.

$$\Pr[\mathbb{E}R_S(F) \geq R_S(F) + t] \leq \exp\left(-\frac{kt^2}{4m\mathbb{E}R_S(F)}\right),$$

where $\mathbb{E}$ takes randomness over the samplings s_1, s_2, $\cdots$, s_k.

Proof. It suffices to show that $R_S(F)$ is a self bounding function with constant $c = 2m/k$. Define

$$Z = R_S(F) = \mathbb{E}_\sigma \sup_{f \in F}\left[\frac{2}{k}\sum_{i=1}^{k} \sigma_i f(s_i)\right],$$

$$G_j = \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{i \neq j} \sigma_i f(s_i) \right].$$

It is clear that Z is non-negative:

$$Z \geq \sup_{f \in F} \left[\mathbb{E}_\sigma \frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i) \right] = 0.$$

Also it is clear that $Z \geq G_j$ for each j: suppose $\tilde{f}$ achieves the supreme of G_j. Then

$$\begin{aligned} G_j &= \mathbb{E}_\sigma \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i \tilde{f}(s_i) - \frac{2}{k} \sigma_j \tilde{f}(s_j) \right] \\ &= \mathbb{E}_\sigma \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i \tilde{f}(s_i) \right] - \mathbb{E}_\sigma \left[\frac{2}{k} \sigma_j \tilde{f}(s_j) \right] \\ &= \mathbb{E}_\sigma \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i \tilde{f}(s_i) \right] \leq Z. \end{aligned}$$

Next we show $Z - G_j \leq 2m/k = c$:

$$\begin{aligned} G_j &= \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i) - \frac{2}{k} \sigma_j f(s_j) \right] \\ &\geq \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i) \right] - \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sigma_j f(s_j) \right] \\ &\geq \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{i=1}^{k} \sigma_i f(s_i) \right] - \frac{2m}{k}. \end{aligned}$$

Finally we need to verify $\sum_{j=1}^{k} Z - G_j \leq Z$:

$$\begin{aligned} \sum_{j=1}^{k} G_j &= \mathbb{E}_\sigma \sum_{j=1}^{k} \sup_{f \in F} \left[\frac{2}{k} \sum_{i \neq j} \sigma_i f(s_i) \right] \\ &\geq \mathbb{E}_\sigma \sup_{f \in F} \left[\frac{2}{k} \sum_{j=1}^{k} \sum_{i \neq j} \sigma_i f(s_i) \right] \\ &= \frac{2(k-1)}{k} \mathbb{E}_\sigma \sup_{f \in F} \left[\sum_{j=1}^{k} \sigma_i f(s_i) \right] = (k-1)Z. \end{aligned}$$

□

We still need the following lemma on the relation between uniform deviation and Rademacher average.

Lemma 3.

$$\mathbb{E} \sup_{f \in F} [A_S(f) - A_D(f)] \leq \mathbb{E} R_S(F),$$

$$\mathbb{E} \sup_{f \in F} [A_D(f) - A_F(f)] \leq \mathbb{E} R_S(F).$$

Here we take randomness over the k samplings.

Proof. The proof idea is based on ghost samplings, i.e., independently draw another k samples: $s'_1, \cdots, s'_k$, and then we have

$$A_D(f) = \frac{1}{m}\sum_{i=1}^{m} f(i) = \mathbb{E}\frac{1}{k}\sum_{j=1}^{k} f(s'_j),$$

where $\mathbb{E}$ takes randomness over the k ghost samples. Thus

$$\begin{aligned}\mathbb{E}\sup_{f\in F}[A_S(f) - A_D(f)] &= \mathbb{E}\sup_{f\in F}\left[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - \mathbb{E}\frac{1}{k}\sum_{j=1}^{k} f(s'_j)\right] \\ &\leq \mathbb{E}\sup_{f\in F}\left[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - \frac{1}{k}\sum_{j=1}^{k} f(s'_j)\right].\end{aligned}$$

Since all the samples s, s' are independently identically distributed, flipping the sign of $f(s_i) - f(s'_i)$ will not change the expected supreme, i.e.,

$$\mathbb{E}\sup_{f\in F}\left[\frac{1}{k}\sum_{i=1}^{k} f(s_i) - \frac{1}{k}\sum_{j=1}^{k} f(s'_j)\right] = \mathbb{E}\sup_{f\in F}\frac{1}{k}\sum_{i=1}^{k}\left[\sigma_i(f(s_i) - f(s'_i))\right],$$

where σ_i is uniformly distributed over $\{-1, 1\}$. Since

$$\mathbb{E}\sup_{f\in F}\frac{1}{k}\sum_{i=1}^{k}\left[\sigma_i(f(s_i) - f(s'_i))\right] \leq 2\mathbb{E}\sup_{f\in F}\frac{1}{k}\sum_{i=1}^{k}\sigma_i f(s_i) = \mathbb{E}R_S(F),$$

we have shown the first inequality. The second inequality is analogous. □

We also need McDiarmid's inequality [32].

Lemma 4. *[32] Let $s_1, \cdots, s_k$ be independent random variables taking values from a set D. Suppose a function $h : D^k \to \mathbb{R}$ satisfies*

$$\sup_{x_1,\cdots,x_k,x'_i\in D}|h(x_1,\cdots,x_k) - h(x_1,\cdots,x_{i-1},x'_i,x_{i+1},\cdots,x_k)| \leq c_i$$

for some constants c_i and every $1 \leq i \leq k$. Then for any $t > 0$, we have

$$\Pr[h(s_1,\cdots,s_k) - \mathbb{E}h(s_1,\cdots,s_k) \geq t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^{k} c_i^2}\right).$$

By the above three lemmas, we can bound the difference between true average and sampled average as follows.

Lemma 5.

$$\begin{aligned}&\Pr\left[\sup_{f\in F}|A_D(f) - A_S(f)| \geq R_S(F) + t\right] \\ \leq& 4\exp\left(-\frac{2kt^2}{(m + \sqrt{8m\mathbb{E}R_S(F)})^2}\right).\end{aligned}$$

Proof. First by Lemma 3,

$$\begin{aligned}
&\Pr\left[\sup_{f\in F}[A_D(f)-A_S(f)]\geq R_S(F)+t\right]\\
\leq&\Pr\left[\sup_{f\in F}[A_D(f)-A_S(f)]\geq \mathbb{E}\sup_{f\in F}[A_D(f)-A_S(f)]+at\right]\\
&+\Pr\left[\mathbb{E}R_S(F)\geq R_S(F)+(1-a)t\right]
\end{aligned}$$

for any $a\in[0,1]$. Let

$$h(s_1,\cdots,s_k)=A_D(f)-A_S(f)=A_D(f)-\frac{1}{k}\sum_{i=1}^{k}f(s_i).$$

It is clear that

$$\begin{aligned}
&\sup_{x_1,\cdots,x_k,x_i'\in D}|h(x_1,\cdots,x_k)-h(x_1,\cdots,x_{i-1},x_i',x_{i+1},\cdots,x_k)|\\
=&\sup_{x_1,\cdots,x_k,x_i'\in D}\left|\frac{1}{k}\sum_{j=1,j\neq i}^{k}f(x_j)+\frac{1}{k}f(x_i')-\frac{1}{k}\sum_{i=1}^{k}f(x_i)\right|\\
=&\sup_{x_1,\cdots,x_k,x_i'\in D}\left|\frac{1}{k}f(x_i')-\frac{1}{k}f(x_i)\right|\leq\frac{m}{k}.
\end{aligned}$$

By McDiarmid's inequality,

$$\begin{aligned}
&\Pr\left[\sup_{f\in F}[A_D(f)-A_S(f)]\geq \mathbb{E}\sup_{f\in F}[A_D(f)-A_S(f)]+at\right]\\
\leq&\exp\left(-\frac{2a^2t^2}{\sum_{i=1}^{k}m^2/k^2}\right)=\exp\left(-\frac{2ka^2t^2}{m^2}\right).
\end{aligned}$$

By Lemma 2,

$$\Pr\left[\mathbb{E}R_S(F)\geq R_S(F)+(1-a)t\right]\leq\exp\left(-\frac{k(1-a)^2t^2}{4m\mathbb{E}R_S(F)}\right).$$

Let $a=1/(1+\sqrt{8\mathbb{E}R_S(F)/m})$. Then putting everything together, we have

$$\begin{aligned}
&\Pr\left[\sup_{f\in F}[A_D(f)-A_S(f)]\geq R_S(F)+t\right]\\
\leq&2\exp\left(-\frac{2kt^2}{(m+\sqrt{8m\mathbb{E}R_S(F)})^2}\right).
\end{aligned}$$

Similarly one can show

$$\begin{aligned}
&\Pr\left[\sup_{f\in F}[A_S(f)-A_D(f)]\geq R_S(F)+t\right]\\
\leq&2\exp\left(-\frac{2kt^2}{(m+\sqrt{8m\mathbb{E}R_S(F)})^2}\right).
\end{aligned}$$

Thus we have the inequality as desired. □

By the above lemma we have the following important corollary.

Corollary 1. *With probability at least $1-\delta$, we have*

$$\sup_{f\in F}|A_S(f)-A_D(f)| \le R_S(F)+(m+\sqrt{8m\mathbb{E}R_S(F)})\sqrt{\frac{\log\frac{4}{\delta}}{2k}}.$$

We still need to upper bound $\mathbb{E}R_S(F)$. By Lemma 2, with probability at least $1-\delta$,

$$\mathbb{E}R_S(F) \le R_S(F)+\sqrt{4m\mathbb{E}R_S(F)\frac{\log\frac{1}{\delta}}{k}}.$$

Or equivalently,

$$\sqrt{\mathbb{E}R_S(F)} \le \sqrt{\frac{m}{k}\log\frac{1}{\delta}}+\sqrt{\frac{m}{k}\log\frac{1}{\delta}+R_S(F)}.$$

Hence with probability at least $1-2\delta$, we have

$$\sup_{f\in F}|A_S(f)-A_D(f)| \le R_S(F)+\left(m+m\sqrt{\frac{8}{k}\log\frac{1}{\delta}}+m\sqrt{\frac{8}{k}\log\frac{1}{\delta}+\frac{8R_S(F)}{m}}\right)\sqrt{\frac{\log\frac{4}{\delta}}{2k}}.$$

We have shown Theorem 2.

8.3 Proof of Theorem 3

For any $s>0$, by Jensen's inequality,

$$\begin{aligned}\exp(skR_S(F)) &= \exp\left(2s\mathbb{E}_\sigma\sup_{f\in F}\sum_{i=1}^k\sigma_i f(s_i)\right)\\ &\le \mathbb{E}_\sigma\exp\left(2s\sup_{f\in F}\sum_{i=1}^k\sigma_i f(s_i)\right)\\ &\le \mathbb{E}_\sigma\sum_{f\in F}\exp\left(2s\sum_{i=1}^k\sigma_i f(s_i)\right).\end{aligned}$$

By Hoeffding's Lemma [23],

$$\begin{aligned}&\mathbb{E}_\sigma\sum_{f\in F}\exp\left(2s\sum_{i=1}^k\sigma_i f(s_i)\right)\\ \le&\sum_{f\in F}\prod_{i=1}^k\exp\left(2s^2f(s_i)^2\right)\\ =&\sum_{f\in F}\exp\left(2s^2\sum_{i=1}^k f(s_i)^2\right).\end{aligned}$$

Let $\ell^2=\sup_{f\in F}\sum_{i=1}^k f(s_i)^2$, and then

$$\sum_{f\in F}\exp\left(2s^2\sum_{i=1}^k f(s_i)^2\right) \le |F|\exp\left(2s^2\ell^2\right).$$

Thus

$$R_S(F) \le \frac{1}{sk}(\log|F|+2s^2\ell^2),$$

for any $s > 0$. It turns out that to minimize the right hand side of the above equation, we have

$$s = \sqrt{\frac{\log |F|}{2\ell^2}}.$$

Then

$$R_S(F) \leq \frac{\ell}{k}\sqrt{8 \log |F|}.$$