COMPSCI 514: Problem Set 2 Answer Key

Do not disseminate this answer key in any fashion, online or otherwise. Doing so is a violation of UMass Amherst's academic honestly policy https://www.umass.edu/honesty/.

1. Some Useful Inequalities (6 points)

There are a number of very useful inequalities that come up over and over again in randomized and approximation algorithm analysis, and more generally in statistical/probabilistic analysis that are worth knowing well.

1. **(1 point)** Show that for any $\epsilon \in [0,1)$, $\frac{1}{1-\epsilon/2} \leq (1+\epsilon)$.

$$\frac{1}{1-\epsilon/2} = \frac{1-\epsilon/2}{1-\epsilon/2} + \frac{\epsilon/2}{1-\epsilon/2} = 1 + \frac{\epsilon}{2(1-\epsilon/2)} \leq 1+\epsilon,$$

where the last inequality follows since $\epsilon \in [0,1]$ and thus $1-\epsilon/2 \geq 1/2$.

2. **(1 point)** Show that for any $\epsilon \geq 0$, $\frac{1}{1+\epsilon} \geq (1-\epsilon)$.

$$\frac{1}{1+\epsilon} = \frac{1+\epsilon}{1+\epsilon} - \frac{\epsilon}{1+\epsilon} \geq 1-\epsilon,$$

where the last inequality follows since $\epsilon \geq 0$ so $\frac{\epsilon}{1+\epsilon} \leq \epsilon$.

3. **(1 point)** Show that for any $\epsilon \in [0,1)$, $(1+\epsilon)^2 \leq 1+3\epsilon$.

$$(1+\epsilon)^2 = 1+2\epsilon+\epsilon^2 \leq 1+3\epsilon,$$

where the last inequality follows since $0 \leq \epsilon \leq 1$ so $\epsilon^2 \leq \epsilon$.

4. **(1 point)** Show that for any $\epsilon \in [0,1)$ and any integer $t \geq 1$, $(1-\epsilon)^t \geq 1-t\epsilon$.

We can prove this via induction. The bound holds immediately in the base case of $t=1$. Consider $t>1$ and assume via induction that the bound holds for $t-1$. We have:

$$(1-\epsilon)^t = (1-\epsilon)\cdot(1-\epsilon)^{t-1} \geq (1-\epsilon)(1-(t-1)\epsilon) = 1-t\epsilon+(t-1)\epsilon^2 \geq 1-t\epsilon.$$

5. **(2 point)** For *any* x, $1+x \leq e^x$. Use this to show that for any $x,c,b>0$ with $\frac{c}{x} \leq 1$, $\left(1-\frac{c}{x}\right)^{x\cdot b} \leq e^{-c\cdot b}$. Use this show that, if I flip a random coin which is heads with probability $1/k$ and tails with probability $1-1/k$, that the probability I see at least $c\cdot k\ln k$ tails in a row is $\leq \frac{1}{k^c}$ for any $c \geq 0$.

Since $x, b > 0$ we can bound

$$\left(1 - \frac{c}{x}\right)^{x \cdot b} \leq \left(e^{-c/x}\right)^{x \cdot b} = e^{-c \cdot b}.$$

The probability I see *at least* $c \cdot k \ln k$ tails in a row is:

$$\left(1 - \frac{1}{k}\right)^{c \cdot k \ln k}.$$

Plugging into our previous bound with $c = 1$ and $b = c \ln k$, this probability is bounded by

$$\left(1 - \frac{1}{k}\right)^{c \cdot k \ln k} \leq e^{-c \ln k} = \frac{1}{k^c}.$$

2. Set Similarities and Locality Sensitive Hash Functions (11 points)

You would like to use shingling and locality sensitive hashing to identify possible plagiarism in student essays. One possibility is to compare an essay A with a publication B in your database using shingling and Jaccard similarity. Another possibility is to use shingling, but then to measure similarity with cosine similarity, where the shingle sets are viewed as binary vectors.

1. **(1 point)** Given an essay A with 1000 words in it an a book B with 10000 words in it, what is the maximum number of 7 word shingles that may appear in A? What is the maximum that may appear in B?

   A 7 word shingle is a set of 7 consecutive words. In A there are $1000 - 6 = 994$ possible sets: the ones starting at words $1, 2, \ldots, 994$. Similarly, in B there are $10000 - 6 = 9994$ such sets. In each document there may be fewer unique shingles – if some shingles appear multiple times. Thus these quantities are the maximum number of shingles appearing.

2. **(1 point)** Consider an essay A with 1000 unique shingles in it and a publication B with 3000 unique shingles, let S_A and S_B be the shingle sets for A and B respectively. So $|S_A| = 1000$ and $|S_B| = 3000$. Assuming the essay A was fully copied from a portion of B, what is the Jaccard similarity between these sets? What is $\Pr(MinHash(S_A) = MinHash(S_B))$.

   Since A is fully copied from B, all 1000 shingles in S_A also appear in S_B. Thus we have:

   $$J(S_A, S_B) = \frac{|S_A \cap S_B|}{|S_A \cup S_B|} = \frac{1000}{3000} = \frac{1}{3}.$$

   Additionally, $\Pr(MinHash(S_A) = MinHash(S_B)) = J(S_A, S_B) = 1/3$.

3. (1 point) Let m be the number of all possible shingles. Represent S_A and S_B by length m binary vectors x_A and x_B, with a 1 in every position corresponding to a shingle they contain. What is the cosine similarity between x_A and x_B (again assuming A was fully copied from a portion B)? What is $\Pr(SimHash(x_A) = SimHash(x_B))$?

   Hint: Use that for any two vectors z, w, $\Pr(SimHash(z) = SimHash(w)) = 1 - \frac{\theta}{\pi}$ where θ is the angle between z and w in radians.

   Note that since x_A and x_B are binary, $\langle x_A, x_B \rangle = |S_A \cap S_B| = 1000$. Additionally, $\|x_A\| = \sqrt{|S_A|} = \sqrt{1000}$ and $\|x_B\| = \sqrt{|S_B|} = \sqrt{3000}$. Thus we have:

   $$\cos(\theta(x_A, x_B)) = \frac{\langle x_A, x_B \rangle}{\|x_A\| \cdot \|x_B\|} = \frac{1000}{\sqrt{1000} \cdot \sqrt{3000}} = \frac{1}{\sqrt{3}} \approx .577.$$

From this, we can compute $\theta(x_A, x_B) = \cos^{-1}(1/\sqrt{3}) = .9553$ radians. Thus we have:

$$\Pr(SimHash(x_A) = SimHash(x_B)) = 1 - \frac{.9553}{\pi} = .6959.$$

4. (2 point) The number of all possible shingles m will generally be huge. Describe why $SimHash(x_A)$ and $SimHash(x_B)$ can be computed in $O(|S_A|)$ and $O(|S_B|)$ time respectively instead of $O(m)$ time. How does this compare to the big-O runtime of MinHash?

   Computing $SimHash(x_A)$ and $SimHash(x_B)$ requires taking the inner product of x_A and x_B with a random vector t. Since x_A and x_B are binary, with supports $|S_A|$ and $|S_B|$ respectively, these inner products only involve at most $|S_A| + |S_B|$ entries of t. Thus, we do not need to generate all of t, but just generate these $|S_A| + |S_B|$ entries, and use them to compute the inner products in $O(|S_A| + |S_B|)$ time.

   We can see that computing $\langle t, x_A \rangle = \sum_{w \in S_A} t(j(w))$ where $j(w)$ is the index of x_A, t corresponding to shingle w is equivalent to hashing each shingle to a random value (i.e., the value of t at the entry corresponding to that shingle $j(w)$) and adding these hash values up for all elements in S_A. Thus SimHash is very similar to MinHash, where instead of taking a min, we take a sum and then the sign.

   The big-O runtime is the same as that of MinHash.

5. (2 points) Consider another essay C also with 1000 unique shingles that is not copied and only shares 100 shingles with B. Compute $\Pr(MinHash(S_C) = MinHash(S_B))$ and $\Pr(SimHash(x_C) = SimHash(x_B))$.

   In this case we have:

$$J(S_C, S_B) = \frac{|S_C \cap S_B|}{|S_C \cup S_B|} = \frac{100}{3000 + 1000 - 100} = \frac{1}{39} \approx .02564.$$

   Thus, $\Pr(MinHash(S_C) = MinHash(S_B)) = J(S_C, S_B) = \frac{1}{39}$.

   We also have

$$\cos(\theta(x_C, x_B)) = \frac{\langle x_C, x_B \rangle}{\|x_C\| \cdot \|x_B\|} = \frac{100}{\sqrt{1000} \cdot \sqrt{3000}} = \frac{1}{10\sqrt{3}} \approx .0577.$$

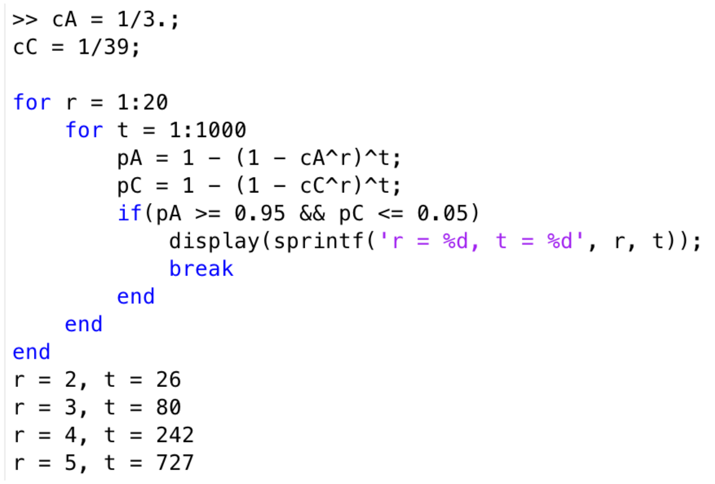
   From this, we can compute $\theta(x_C, x_B) = \cos^{-1}\left(\frac{1}{10\sqrt{3}}\right) = 1.5130$ radians. Thus we have:

$$\Pr(SimHash(x_C) = SimHash(x_B)) = 1 - \frac{1.5130}{\pi} = .5184.$$

6. (3 points) For both MinHash and SimHash, find a signature length r and repetition parameter t such that the fully copied essay A is identified with LSH-based similarity search with probability $\geq .95$ and the non-copied essay C is identified with probability $\leq .05$. Focus on minimizing the space complexity (i.e., the number of hash tables t used). By 'identified', we mean that the essay falls in the same bucket as B in at least one of the t hash tables.

   Hint: It may be helpful (although is not required) to write a very simple program to help solve this one.

Using signature length r and t repetitions, the probability that an essay falls in the same bucket as B in at least 1 table when the applied hash function collides with probability s is $1-(1-s^r)^t$. We can run the following simple program to check for r and t that satisfy the constraint for MinHash (When the collision probabilities for A and C are $1/3$ and $1/39$ respectively.

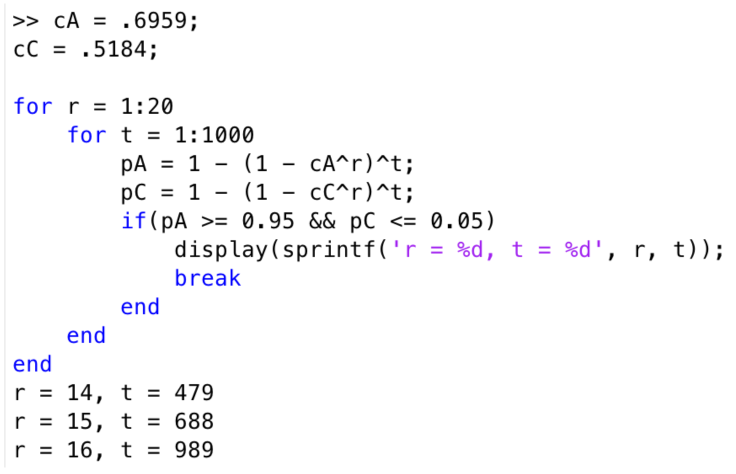


```
>> cA = 1/3.;
cC = 1/39;

for r = 1:20
    for t = 1:1000
        pA = 1 - (1 - cA^r)^t;
        pC = 1 - (1 - cC^r)^t;
        if(pA >= 0.95 && pC <= 0.05)
            display(sprintf('r = %d, t = %d', r, t));
            break
        end
    end
end
r = 2, t = 26
r = 3, t = 80
r = 4, t = 242
r = 5, t = 727
```

Thus we can pick **r** $=$ **2 and t** $=$ **26** to minimize the space complexity.

We can run a similar program for the SimHash collision probabilities of .6959 for A and .5184 for C. In this case the output is:



```
>> cA = .6959;
cC = .5184;

for r = 1:20
    for t = 1:1000
        pA = 1 - (1 - cA^r)^t;
        pC = 1 - (1 - cC^r)^t;
        if(pA >= 0.95 && pC <= 0.05)
            display(sprintf('r = %d, t = %d', r, t));
            break
        end
    end
end
r = 14, t = 479
r = 15, t = 688
r = 16, t = 989
```

Thus we can pick **r** $=$ **14 and t** $=$ **479** to minimize the space complexity.

7. (1 point) Given the above, which similarity metric and hash function would you pick for the plagiarism detection task?

   We should pick MinHash – the number of tables t and thus the space usage required is lower by almost a factor 20.

3. A Better Count-Min Sketch (5 points)

Consider the count-min sketch algorithm: when processing a stream of inputs $x_1, \ldots, x_n$, we store t length-m count arrays $A_1, \ldots, A_t$ and chose t random hash functions $\mathbf{h}_1, \ldots, \mathbf{h}_t$ mapping the universe of possible items to $[m]$. When input x_i comes in we increment each count $A_1[\mathbf{h}_1(x_i)], \ldots, A_t[\mathbf{h}_t(x_i)]$. Once the stream has been processed, we estimate the frequency of any element x in the stream, denoted $f(x)$, by $\tilde{f}(x) = \min_{j \in [t]} A_j[\mathbf{h}_j(x)]$.

1. (4 points) Consider a variation on count-min sketch: instead of incrementing each counter $A_1[\mathbf{h}_1(x_i)], \ldots, A_t[\mathbf{h}_t(x_i)]$ when x_i comes in, we compute $M = \min_{j \in [t]} A_j[\mathbf{h}_j(x_i)]$. Then we only increment $A_j[\mathbf{h}_j(x_i)]$ if $A_j[\mathbf{h}_j(x_i)] = M$. Show that the estimate output by this variation can *only be better* than the estimate of the count-min sketch algorithm presented in class.

   This modification to count-min sketch is known as *count-min sketch with conservative updates.* For any x, letting $f(x)$ be the true frequency of x in the stream, we first note that for all $j \in [t]$, $A_j[\mathbf{h}_j(x)] \geq f(x)$ once all items have been inserted. We prove this via induction. Let $f_s(x)$ be the number of times that x has appeared in the stream up until insertion s. Our goal is to show that after all items have been inserted, for all $j \in [t]$, $A_j[\mathbf{h}_j(x)] \geq f_n(x) = f(x)$.

   For $s = 0$, $f_s(x) = 0$ and $A_j[\mathbf{h}_j(x)] = 0$ and thus the claim $A_j[\mathbf{h}_j(x)] \geq f_0(x)$ holds trivially. Now consider $s \geq 1$ and assume the claim holds for $s - 1$. If $x_s \neq x$ then $f_s(x) = f_{s-1}(x)$ and since the counts in the arrays can only be incremented we still have $A_j[\mathbf{h}_j(x)] \geq f_s(x)$.

   If $x_s = x$ then $f_s(x) = f_{s-1}(x) + 1$. Additionally, by the induction assumption, $M = \min_{j \in [t]} A_j[\mathbf{h}_j(x_s)] \geq f_{s-1}(x_s) = f_{s-1}(x)$. Thus, after x_s is inserted, for all $j \in [t]$ we have $A_j[\mathbf{h}_j(x_s)] \geq M + 1 \geq f_{s-1}(x) + 1 = f_s(x)$. This gives the claim, and completes the proof.

   Finally, since count-min sketch with conservative updates only makes less increments than the original count-min sketch algorithm, we have that $\tilde{f}(x) = \min_{j \in [t]} A_j[\mathbf{h}_j(x)]$ is only lower than the estimate given by count-min sketch. Additionally, since by our argument above $\tilde{f}(x) \geq f(x)$, $\tilde{f}(x)$ can only be a better estimate.

2. (1 point) Why might you still use the original count-min sketch algorithm? Note that the updates for the original version are slightly faster, but we're looking for a more significant reason.

   The original count-min sketch algorithm is easily run in a distributed environment. Each server simply stores a count-min sketch datastructure for the elements that it processes and in the end, the arrays comprising these data structures are simply added together. This exactly simulates the centralized count-min sketch algorithm. It is less clear how to distribute count-min sketch with conservative updates – since the algorithm would require checking $\min_{j \in [t]} A_j[\mathbf{h}_j(x_i)]$ at each element insertion, which would require communicating with all the servers.

4. Randomized Dimensionality Reduction for Clustering (10 points)

One of the most popular clustering objectives is *k-means* clustering. Given a set of n data points $X = \{x_1, \ldots, x_n\}$ all in $\mathbb{R}^d$ the goal is to partition $[n]$ into k sets (clusters) $\mathcal{C} = \{C_1, \ldots C_k\}$ minimizing:

$$cost(\mathcal{C}, X) = \sum_{j=1}^{k} \sum_{i \in C_j} \|x_i - \mu_j\|_2^2 \qquad (1)$$

where $\mu_j = \frac{1}{|C_j|}\sum_{i\in C_j} x_i$ is the *centroid* of cluster C_j (i.e., the mean of the points in that cluster.)

1. (3 points) Show that $cost(\mathcal{C})$ can be equivalently written as:

$$cost(\mathcal{C}, X) = \sum_{j=1}^{k} \frac{1}{2|C_j|} \sum_{i_1\in C_j}\sum_{i_2\in C_j} \|x_{i_1} - x_{i_2}\|_2^2. \tag{2}$$

Intuitively, what does this reformulation of the cost function mean?

Hint: Show that both (1) and (2) can be rewritten as $\sum_{j=1}^{k}\left[\left(\sum_{i\in C_j}\|x_i\|_2^2\right) - |C_j|\cdot\|\mu_j\|_2^2\right]$. This will require some vector algebra. It will be helpful recall use that for any vector z, $\|z\|_2^2 = \sum_{i=1}^{d} z(i)^2 = \langle z, z\rangle$, as well as to use the linearity of inner product.

Focusing on a single cluster C_j we have:

$$\begin{aligned}
\sum_{i\in C_j}\|x_i-\mu_j\|_2^2 &= \sum_{i\in C_j}\langle x_i-\mu_j, x_i-\mu_j\rangle\\
&= \sum_{i\in C_j}\langle x_i, x_i\rangle - 2\langle x_i,\mu_j\rangle + \langle\mu_j,\mu_j\rangle\\
&= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) + |C_j|\cdot\|\mu_j\|_2^2 - 2\sum_{i\in C_j}\langle x_i,\mu_j\rangle\\
&= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) + |C_j|\cdot\|\mu_j\|_2^2 - 2\left\langle\sum_{i\in C_j} x_i,\mu_j\right\rangle.
\end{aligned}$$

Using the definition of the centroid $\mu_j = \frac{1}{|C_j|}\sum_{i\in C_j} x_i$ this gives:

$$\begin{aligned}
\sum_{i\in C_j}\|x_i-\mu_j\|_2^2 &= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) + |C_j|\cdot\|\mu_j\|_2^2 - 2|C_j|\langle\mu_j,\mu_j\rangle\\
&= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) - |C_j|\cdot\|\mu_j\|_2^2 \qquad (3)
\end{aligned}$$

Similarly, focusing on a single cluster C_j

$$\begin{aligned}
\frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|x_{i_1}-x_{i_2}\|_2^2 &= \frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|x_{i_1}\|_2^2 + \|x_{i_2}\|_2^2 - 2\langle x_{i_1}, x_{i_2}\rangle\\
&= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) - \frac{1}{|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\langle x_{i_1}, x_{i_2}\rangle\\
&= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) - \sum_{i_1\in C_j}\left\langle x_{i_1}, \frac{1}{|C_j|}\sum_{i_2\in C_j} x_{i_2}\right\rangle.
\end{aligned}$$

Again using the definition of μ_j we have:

$$\frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|x_{i_1}-x_{i_2}\|_2^2 = \left(\sum_{i\in C_j}\|x_i\|_2^2\right) - \sum_{i_1\in C_j}\langle x_{i_1},\mu_j\rangle$$
$$= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) - \left\langle \sum_{i_1\in C_j} x_{i_1},\mu_j\right\rangle$$
$$= \left(\sum_{i\in C_j}\|x_i\|_2^2\right) - |C_j|\cdot\|\mu_j\|_2^2. \tag{4}$$

Together (3) and (4) give that $\sum_{i\in C_j}\|x_i-\mu_j\|_2^2 = \frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|x_{i_1}-x_{i_2}\|_2^2$ and thus:

$$cost(\mathcal{C},X) = \sum_{j=1}^{k}\sum_{i\in C_j}\|x_i-\mu_j\|_2^2 = \sum_{j=1}^{k}\frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|x_{i_1}-x_{i_2}\|_2^2.$$

Intuitively, this reformulation writes the cost as the sum of *intracluster variances*: a good set of clusters minimizes the distances between points in the same clusters.

2. (4 points) Suppose that $\mathbf{\Pi}\in\mathbb{R}^{m\times d}$ is a random projection matrix with each entry chosen independently as $\frac{1}{\sqrt{m}}\mathcal{N}(0,1)$. For each x_i in the dataset, let $\tilde{x}_i = \mathbf{\Pi}x_i$ and let $\tilde{X} = \{\tilde{x}_1,\ldots,\tilde{x}_n\}$ denote our set of sketched data points in $\mathbb{R}^m$.

   Given parameters $\epsilon,\delta\in(0,1)$ how large must we set m so that, with probability $\geq 1-\delta$, for all clusterings $\mathcal{C}$,

$$(1-\epsilon)cost(\mathcal{C},X) \leq cost(\mathcal{C},\tilde{X}) \leq (1+\epsilon)cost(C,X).$$

   That is, how much can we reduce the dimension (from d to m) and still approximately preserve all cluster costs with high probability. Give the answer in big-O notation.

   Let $\epsilon' = \epsilon/3$. By the Johnson-Lindenstrauss lemma, if we set $m = O\left(\frac{\log n/\delta}{\epsilon'^2}\right) = O\left(\frac{\log n/\delta}{\epsilon^2}\right)$, then with probability $\geq 1-\delta$, for all i,j we will have $(1-\epsilon/3)\|x_i-x_j\|_2 \leq \|\tilde{x}_i-\tilde{x}_j\|_2 \leq (1+\epsilon/3)\|x_i-x_j\|_2$. By Problem 1, (3) and (4) this gives:

$$(1-\epsilon)\|x_i-x_j\|_2^2 \leq \|\tilde{x}_i-\tilde{x}_j\|_2^2 \leq (1+\epsilon)\|x_i-x_j\|_2^2.$$

   By part 1, we thus have:

$$cost(\mathcal{C},\tilde{X}) = \sum_{j=1}^{k}\frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|\tilde{x}_{i_1}-\tilde{x}_{i_2}\|_2^2$$
$$\leq \sum_{j=1}^{k}\frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}(1+\epsilon)\|x_{i_1}-x_{i_2}\|_2^2$$
$$= (1+\epsilon)\sum_{j=1}^{k}\frac{1}{2|C_j|}\sum_{i_1\in C_j}\sum_{i_2\in C_j}\|x_{i_1}-x_{i_2}\|_2^2$$
$$= cost(\mathcal{C},X).$$

   An identical argument gives that $cost(\mathcal{C},\tilde{X}) \geq (1-\epsilon)cost(\mathcal{C},X)$, which completes the claim.

3. (3 points) Use the above to show how large we must set m such that, if we find an *optimal clustering* for the dimension reduced dataset $\tilde{x}_1, \ldots, \tilde{x}_n$, then with probability $\geq 1 - \delta$ this clustering will have at most $(1+\epsilon)$ of the optimal cost on the original data set $x_1, \ldots, x_n$. Again, use big-O notation in your answer.

   Let $\epsilon' = \epsilon/3$ and set $m = O\left(\frac{\log n/\delta}{\epsilon'^2}\right) = O\left(\frac{\log n/\delta}{\epsilon^2}\right)$. Let $\mathcal{C}^*$ be an optimal clustering for X and let $\tilde{\mathcal{C}}^*$ be an optimal clustering for $\tilde{X}$. By part 2 applied with error $\epsilon' = \epsilon/3$ we have:

$$cost(\tilde{\mathcal{C}}^*, X) \leq \frac{1}{1-\epsilon/3} \cdot cost(\tilde{\mathcal{C}}^*, \tilde{X}) \leq \frac{1}{1-\epsilon/3} \cdot cost(\mathcal{C}^*, \tilde{X})$$

   where the second inequality follows from the optimality of $\tilde{\mathcal{C}}^*$ for $\tilde{X}$. Also applying part 2:

$$cost(\mathcal{C}^*, \tilde{X}) \leq (1+\epsilon/3) \cdot cost(\mathcal{C}^*, X).$$

   Combining these bounds give:

$$cost(\tilde{\mathcal{C}}^*, X) \leq \frac{1+\epsilon/3}{1-\epsilon/3} \cdot cost(\mathcal{C}^*, X).$$

   The claim follows after noting that for $\epsilon \in [0,1]$, $\frac{1+\epsilon/3}{1-\epsilon/3} = 1 + \frac{2\epsilon/3}{1-\epsilon/3} \leq 1 + \frac{2\epsilon/3}{2/3} \leq 1+\epsilon$.