CS-541 A Midterm

Name: Stephen Szemis

1. I find the idea of random projection very interesting (since it is a new topic for me). I suppose the idea that you can generate a random matrix for some size k (in lecture it's call A), then multiple it by some original data (called X), and get a lower dimensional representation without losing relative distance seems very counter intuitive. After all, compared to PCA, it's very straight foreward. Of course the proof uses Chebyshelv's Inequality to evantually reach Johnson Lindenstrauss Lemma, so RP does have a serious theoretical backing.

2. Let X be random variable.

- $E[X] = 1$, this does tell us something about $P(X)$. It tells us that $P(X)$ sampled many times, averages to 1. However it isn't very practical to base real world decisions soly on expected value.

- $Var[X] = 10$, this (along with $E(X)$) gives much stronger information about the behavior of $P(X)$. While before we could only make weak claims about $P(X)$ underlying behavior, now we can give tight bounds around how quickly we can reach accumulated values of X, say for gambling.

2. continued

This all comes from Markov's and Chebyshev's Inequalities.

If $E(X)$ gives us a "center" for our underlying distribution, $Var[X]$ gives us how "spread out" our distribution is from that center.

3.
- We want probability ≥ 0.99.
- No more then 100 dollars (i.e. $n \leq 100$)
- Let $Pr(X=1) = p$ where $X=1$ is a correct label. $Pr(X=-1) = 1-p$ where $X=-1$ is WRONG!

$$E[X] = Pr(X=1)\cdot 1 + Pr(X=0)\cdot 0 = p$$
$$Var[X] = p(1-p)$$

$$S = Pr\left(\sum_{i=1}^{n} x_i > 0\right)$$

- Use Chebyshev's Inequality for high probability argument

$$Pr(|X - E[X]| \geq t) \leq \frac{Var[X]}{t^2}$$

$$E(S) = \sum_{i=1}^{n}(p + p - 1) = n(2p-1)$$

$$Var[S] = \sum_{i=1}^{n}\cdot\left(1 + (2p-1)^2\right) = n(4p - 4p^2) = 4pn(1-p)$$

So...

$$Pr(|S - n(2p-1)| \geq t) \leq \frac{4pn(1-p)}{t^2}$$

↳ continue to next page

3. $n(2p-1) - t \geq 1$, at least majority

$\frac{4np(1-p)}{t^2} \leq 0.01$, at least 0.99

$n \leq \frac{0.01t^2}{4p(1-p)} \leq 99$, from $n < 100$ (cost)

$t = \sqrt{0.01 \cdot 99 \cdot 4p(1-p)}$

$99(2p-1) - \sqrt{0.01 \cdot 99 \cdot 4p(1-p)} \geq 1$

$$f(p) = 99(2p-1) - \sqrt{0.01 \cdot 99 \cdot 4p(1-p)} - 1 \geq 0$$

4.

- Major advantages:
  - Not dependant on input data.
  - Keeps relative distance
  - Not very expensive

- You would not gain a computational boost if your new k dimensions is not suitably smaller then d.

  For some $O(nd)$ algorithm you have computational gain of

  $$nd - nk = n(d-k)$$

  But RF cost some operations, say C then for it to be worth it

  $$n \geq \frac{C}{d-k}$$

  If n is not large enough, then dimension reduction is not worth it.

- Using another distrobution would probably work, but we still require a standard deviation of around $1/\sqrt{k}$ and mean of 0. Such a distrobution would need to be designed in the discrete case based on k. A bit more work. Though sampling discretely is probably less computationally costly than normal distrobution for large matrixes.

5. The main idea of collaborative filtering is to populate a sparse matrix of user data by "grouping" users by their similiar intrests. For instance I may have a row in an amazon product database

$$[\ldots, 0, 1, 0, 1, \ldots]$$
$$\quad\ n \quad n+1 \quad n+2 \quad n+3$$

And the professor may have this as his row

$$[\ldots, 1, 1, 0, 1 \ldots]$$
$$\quad\ n \quad n+1 \quad n+2 \quad n+3$$

We have similiar $n+1$ and $n+3$ interest, so we may have similiar n index interest. From that assumption we fill may n'th item index with some value. And our matrix is slightly less sparse.

Draw backs for this formulation is that we must assume there exists some low rank representation of our data, which is not always true in all applications. It also isn't clear how to build such a system in a scalable way. For instance amazon must deal with millian by millian matrix of user items. Very sparse. There are also more complications in a binary case, since we need to recover an actual continuous value in order to base our distance measures.

Some improvents might include clustering data so we can work on smaller subsets of our data. Lowering dimonionality of data. Using a machine learning model to find more arbetray patterns in our data.