Sample Questions for Preparing for the Final Exam

April 30th

1. **Parallel learning methods - perceptrons**

   Recall that iterative parameter mixing (IPM) algorithm for perceptrons works as follows: First, divide the data into S shards, and initialize a weight vector $\mathbf{w}^0$ to zero. Then, in each iteration t, run, in parallel, a perceptron for one pass over the a single shard, starting with weight vector $\mathbf{w}^{t-1}$; and uniformly average the final weight vectors from each shard to create the next weight vector $\mathbf{w}^t$.

   Suppose a dataset, where are all examples $\mathbf{x}$ are of bounded radius $\|\mathbf{x}\|^2 \leq R^2$, is separable with margin γ. Will a perceptron with IPM converge? why or why not?

2. In parameter learning for LDA, collapsed Gibbs Sampling is used to assign values to latent variables.

   (a) For a corpus with D documents with exactly M words in each document, a vocabulary sie of V, and K topics, how many latent variables will there be?

   (b) What is the minimum number of bits needed to store the values of the latent variables?

(c) How many parameters are stored?

3. Consider the cost of drawing one sample in a naive implementation of collapsed Gibbs sampling for LDA. In which parameters is this cost linear? (check all that apply)?

   ◯ the size of the corpus
   ◯ the vocabulary size
   ◯ the number of topics
   ◯ the number of iterations of sampling
   ◯ the maximum length of a document

4. Describe the AllReduce operation briefly, and explain why it cannot be implemented using ordinary map and reduce steps without loss of efficiency.

5. You have implemented the following WordCount MapReduce algorithm:

   Mapper: for each word in the input, emit (word, 1)

   Reducer: for each word as key, sum up all 1's as count.

   To optimize, you plan to implement a Combiner which aggregates intermediate map output. Can you reuse the code inside the reduce() function to implement the Combiner? Pick one answer:

   ◯ Yes, because the sum operation is associative and commutative and the input types to the reduce method match.
   ◯ No, because Combiner and Reducer have incompatible input and output types.
   ◯ Yes, because Java is polymorphic and will automatically convert from type required by the Reducer to the type required by the Combiner.

◯ No, because the Combiner is incompatible with a mapper which doesn't use the same data type for the key and value.

6. *Reservoir sampling* is a process for streaming through a list of items n and randomly sampling k of them, while using only $O(k)$ memory. (The basic idea is to keep a "reservoir" of k items, which are initialized to the first k items in the list, and when each new item is encountered at position i, swap it into the list with an appropriately-chosen probability that depends on i.) We'd like to scale up reservoir sampling by, in stage 1, creating a subsample of each of m shards of a large list, and in stage 2, combining the subsamples.

   Besides the subsamples themselves, what additional information would be needed in stage 2?

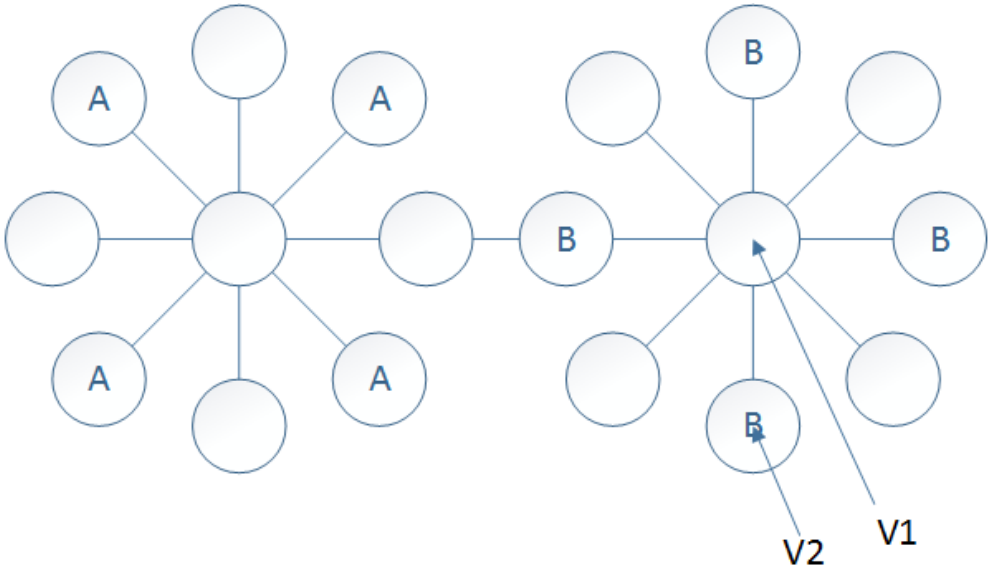
7. **Semi-supervised learning** We want to perform semi-supervised learning on a graph $G = (V, E)$ using the "MultiiRankWalk" method. V is composed of unlabeled vertices V^U and labeled vertices V^L with corresponding labels Y^L. You decide to use the following MultiiRankWalk code, with reset α, to label the unlabeled nodes V^U.

   A. For each class c
      (a) Let S_c be the seeds for class c, i.e., $S_c = |\{v \in V^L : y_v = c\}|$, and let $\mathbf{u}_c$ be a vector that gives uniform weight to the vertices in S_c.
      (b) Set $\mathbf{pr}_c \leftarrow PersonalizedPageRank(G, \mathbf{u}_c, \alpha)$, i.e., the vector of personalized PageRank scores for reset vector $\mathbf{u}_c$, computed using the usual power iteration method.
   B. For each unlabeled vertex $v \in V^U$, assign it the class that has the highest score, i.e.,

$$\arg\max_c \mathbf{pr}_c[v]$$

   Discuss briefly the advantages or disadvantages of the two following proposals for speeding up the algorithm. (A) Running the power iteration in the personalized PageRank algorithm for fewer iterations, so that it converges only approximately. (B) Subsampling the edges in the graph. (C) Using a different value of α.

8. If we run MultiRank Walk algorithm on the following graph, what is the result? The nodes labeled "A" and "B" are seeds, and you should fill in the predictions on the blank circles ("A" or "B").



9. Relative to the picture above, sort $\mathbf{pr}_{B,V1}, \mathbf{pr}_{B,V2}, \mathbf{pr}_{A,V1}, \mathbf{pr}_{A,V2}$ in descending order.

10. **SGD and matrix factorization - True/False**

    True or false: DSGD is called a *non-negative matrix factorization method*, because it cannot be be used on a matrix with negative entries.

11. **LDA - True/False**

    Mark each of the statements as true or false.

    ◯ AliasLDA uses a multinomial sampling algorithm that has amortized cost $O(n)$ where n is the dimension of the multinomial being sampled (e.g., the number of topics being sampled.)

    ◯ AliasLDA is faster than SparseLDA because it employs parallel sampling.

12. **Randomization and Sketches - True/False** True or False:

    A. Checking Bloom filters for the existence of an element never results in a false positive.

B. Using CountMinSketch for approximate counting of elements never provides underestimates for the count of an element.

13. You are given a CountMinSketch table with 10 hash functions, each of which has an output from 1 to 1000. Thus, the two-dimensional CmountMinSketch table is 10 by 1000. Each hash function maps uniformly over its range i.e. an input is equally likely to be hashed to any output between 1 to 1000. You insert two elements A and B into the CountMinSketch data structure. If you now query for the count of A, what is the probability that the count returned by querying the CountMinSketch data structure is wrong?

14. **Hashing.** True or False: Feature hashing (the "hashing trick") is most useful to increase the feature dimension for logistic regression so that a dataset becomes linearly separable.

15. **SGD for MF.** Matrix completion aims to decompose an $N \times M$ matrix $D \in \mathbb{R}^{N \times M}$ into two factor matrices $L \in \mathbb{R}^{K \times N}$ and $R \in \mathbb{R}^{K \times M}$. L2-regularized matrix factorization minimizes the following objective:

$$\min_{L,R} \sum_{(i,j) \in D_{obs}} ||D_{ij} - L_i^T R_j||^2 + \lambda \left(\sum_{i=1}^{N} ||L_i||_2^2 + \sum_{j=1}^{M} ||R_j||_2^2 \right) \tag{1}$$

where $L_i \in \mathbb{R}^{K \times 1}$ and $R_j \in \mathbb{R}^{K \times 1}$ are columns of L, R factor matrices respectively; $L_i^T R_j := \sum_k L_{ik} R_{jk}$ is the vector inner product; D_{ij} are user i's rating on product j; and λ is the regularization parameter.

A student made a mistake in deriving the gradient updates:

$$\begin{aligned} L_i &\leftarrow L_i + \gamma(e_{ij} R_j - \lambda L_i) \\ R_j &\leftarrow R_j + \gamma(e_{ij} L_i - \lambda R_j) \end{aligned} \tag{2}$$

where $e_{ij} = D_{ij} - L_i^T R_j$. γ is the step-size which can scale and absorbs necessary constants. Write down the objective function optimized by Eq. 2. You can use $n_i := \sum_{j=1}^{M} \mathbb{I}(D_{ij} \neq 0)$ to denote the number of non-zero elements in row i of the data matrix D, and m_j to denote the number of non-zeros in column j of D.

16. **Parallel learning methods - concepts**

You're studying the problem of learning to classify each base-pair position of a strand of human DNA as positive or negative, where "positive" means

"inherited from Neanderthals". This is a structured-output classification problem: for each example strand $\mathbf{x} = x_1 \ldots, x_n$, you must produce a sequence of predictions $\mathbf{y} = y_1 \ldots, y_n$, where y_i is the class corresponding to base pair x_i. Nearby classes are closely correlated: i.e., y_i and y_{i+1} are likely to be the same. There are millions of positions in each strand, and tens of thousands of examples.

You decide to design a parallel learning method. Which approach seems best? Mark it, and explain in a sentence or two why it seems reasonable.

- ◯ Split the data "horizontally": assign each instance to a random shard and process the shards separately (e.g, using IPM).
- ◯ Split the data "vertically" by dividing each example into contiguous groups of positions (e.g., $i = 1, \ldots, 10000$), and process the set of examples for each contiguous group of positions separately.
- ◯ Split the data "vertically" but place each position into a random shard.

Your explanation: