Exercise 4

Time tip: Roughly 45sec to 1min per 1pt
Tips on MapReduce Problems

General approach/advice on how to cast a data analytics computation onto the MapReduce API:

Step 1) Identify the exact data access pattern of the computation over the dataset. Draw it out to see it visually if you like.

Step 2) Identify how to decompose the bulk of the whole computation into a bunch of independent chunk computations on sub-elements (rows/columns/tiles). Typically, scalability along rows is the most preferable because most modern large-scale datasets have large numbers of rows.

Step 3) Identify how to aggregate those decomposed parts to get the final result as if it was computed in a single-threaded in-RAM manner. This aggregation step may not always be needed though.

Step 4) Align the sharding with Step 2. Put the independent chunk computations in the Mapper. Identify what the Mapper’s intermediate output (emit) data structure should be. Put the aggregation in Step 3 and any post processing in the Reducer.
Q1) [6pts] Write pseudocode for (or just describe precisely in prose) a MapReduce job to compute the Frobenius norm (aka L2 norm) of a given large matrix. It should be scalable along the number of rows.

Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Input Split: Shard row-wise so that it is scalable along # rows
Need full MR job due to global aggregation for norm; as per 4 steps:
1) Plain ol’ total sum of all cells x^2; addition order does not matter.
2) Can chunk whole computation into independent ones over rows/shards.
3) Just add up results of independently computed partial sum
4) - Map(): Given the shard with multiple rows, compute x^2 of each cell and add them up into partial sum; emit it with single global dummy key
   - Reduce(): Iterator with all partial sums; add them all up, take square root of global sum, emit that as final result: L2 norm
Q3) [8pts] (Quiz 1) Write pseudocode for a MapReduce job to compute the Gramian of a given large matrix. It should be scalable along the number of rows. Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Using the 4-step method, here is the access pattern of $G = X^T \times X$:

$$G[i, j] \leftarrow \sum_{k=1}^{n} X[k, i] \times X[k, j]$$

1) So, the access pattern is a sequential scan over $X$. 
Exercise

Q3) [8pts] (Quiz 1) Write pseudocode for a MapReduce job to compute the Gramian of a given large matrix. It should be scalable along the number of rows. Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Using the 4-step method, now we get to step 2:

\[ G[i, j] \leftarrow \sum_{k=1}^{n} X[k, i] \times X[k, j] \]

2) This summation expression for \( G[i,j] \) shows we can decompose it across the \( n \) rows of \( X \). So, each chunk is a row. This also ensures scalability along \# rows. But note that given \( k \)th row \( X[k,:) \) of \( X \), we can batch all \((i,j)\) pairs for \( G \) onto the same pass over \( X \). So, the chunk-level computation has the size of the entire matrix \( G[:]. \)
Exercise

Q3) [8pts] (Quiz 1) Write pseudocode for a MapReduce job to compute the Gramian of a given large matrix. It should be scalable along the number of rows. Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Using the 4-step method, now we get to step 3:

\[
G[i, j] \leftarrow \sum_{k=1}^{n} X[k, i] \times X[k, j]
\]

3) The aggregation part over the chunk-level computation is simple in this case: just the big summation over the chunk outputs.
Exercise

Q3) [8pts] (Quiz 1) Write pseudocode for a MapReduce job to compute the Gramian of a given large matrix. It should be scalable along the number of rows.
Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Using the 4-step method, now we get to step 4:

\[ G[i, j] \leftarrow \sum_{k=1}^{n} X[k, i] \times X[k, j] \]

4) So, we have the MR answer: Shard X row-wise. Say X is of size n x d.
Inside the Mapper, given a single row x, compute the outer vector product x \times x^T on that single row to obtain a d x d matrix. Emit that matrix as value (no key in emit). Reducer just gets the Iterator with all such partial matrices and adds them all up.
Q2) [8pts] Write pseudocode for (or just describe precisely in prose) a single MapReduce job to compute the Person correlation matrix of a given data matrix. It should be scalable along the number of rows of the dataset. Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Hint: The Pearson correlation coefficient between two variables \( x \) and \( y \) can be rewritten as follows (given \( n \) rows):

\[
r_{x,y} = \frac{n \sum_{i=1}^{n} (x_i y_i) - \sum_{i=1}^{n} x_i \sum_{i=1}^{n} y_i}{\sqrt{n(\sum_{i=1}^{n} x_i^2)} - (\sum_{i=1}^{n} x_i)^2} \frac{\sqrt{n(\sum_{i=1}^{n} y_i^2)} - (\sum_{i=1}^{n} y_i)^2}{\sqrt{n(\sum_{i=1}^{n} y_i^2)} - (\sum_{i=1}^{n} y_i)^2}
\]

Basically like Q3. Left as homework. Feel free to post on Piazza/Canvas and discuss with the class.
Q4) Suppose you are given a large dataset with 50 numeric and 9 categorical features (domain size of 50 each). The HDFS file size is 3 TB.

A. **[10pts]** Write pseudocode for (or just describe precisely in prose) MapReduce job(s) to compute this dataset’s correlation matrix. Hint: You would need could do two separate MapReduce jobs.

B. **[4pts]** What is the rough total disk I/O cost of the above in TB? Include both reads and writes of intermediate data and output.

C. **[4pts]** Briefly explain how you would scale this computation on an on-premise cluster.

D. **[4pts]** Briefly explain how you would scale this computation on AWS.

Suppose we one-hot encode all categorical features into 50-dimensional 0-1 vectors. Then total number of numerics is \(50 + 9 \times 50 = 500\). So, correlation matrix is of size \(500 \times 500\), which is 250,000 cells. Even with float64, it is only 2MB. So, we will use this as our aggregation state for Mappers to send to Reducer.
Exercise

**Q4)** Suppose you are given a large dataset with 50 numeric and 9 categorical features (domain size of 50 each). The HDFS file size is 3 TB.

A. **[10pts]** Write pseudocode for (or just describe precisely in prose) MapReduce job(s) to compute this dataset’s correlation matrix. Hint: You would need two separate MapReduce jobs.

A. Input Split: Shard tuple-wise as usual.
   - One approach to compute correlation matrix uses 2 MR jobs: the first to compute the per-feature means and stdevs; and the second to use those to finish the correlation computations.
   - First Map() reads tuple, converts each categorical feature to its respective one-hot encoded vector as in Q4.D earlier to stitch together full 500-dimensional numeric vector, and computes suff. stats for mean and stdev as 3-tuple (1, x, x^2); emits this 3-tuple as value with a single global dummy key.
   - First Reduce() aggregates all sufficient stats to emit 2 vectors, one with the mean of each feature and one with the stdev of each feature.
Exercise

Q4) Suppose you are given a large dataset with 50 numeric and 9 categorical features (domain size of 50 each). The HDFS file size is 3 TB.
A. [10pts] Write pseudocode for (or just describe precisely in prose) MapReduce job(s) to compute this dataset’s correlation matrix. Hint: You would need two separate MapReduce jobs.

A. - Second Map() reads tuple, gets the 500-dimensional numeric vector again as before and then emits as sufficient stats a 500x500 matrix representing pairwise products for the aggregation needed for the numerator of the Corr matrix formula below.
- Second Reduce() just adds up these individual matrices and divides all cells by total example count and the respective pairs of stdevs obtained from the first MR job.

\[ \text{Corr}(A, B) = \frac{E[(A - \mu_A)(B - \mu_B)]}{\sigma_A \sigma_B} \]
Exercise

Q4) Suppose you are given a large dataset with 50 numeric and 9 categorical features (domain size of 50 each). The HDFS file size is 3 TB.

B. [4pts] What is the rough total disk I/O cost of the above in TB? Include both reads and writes of intermediate data and output.

C. [4pts] Briefly explain how you would scale this computation on an on-premise cluster.

D. [4pts] Briefly explain how you would scale this computation on AWS.

B. 2 filescans amount to 6 TB. Intermediate data after first job has 500x2 numbers; output is 500x500 matrix. Assuming all numbers are float64, these amount to mere ~2 MB extra. So, overall I/O cost is still roughly 6 TB.

C. Shard data on Spark cluster; write Spark-MapReduce job. Note that Dask may not be a good fit, since 3 TB file may not fit even on single-node disk, let alone single-node DRAM!

D. Option 1: Q4.C’s approach on EMR. Option 2: single-machine Python and remote reads from S3. Latter has less parallelism and might be very slow.
Q5) [3 x 3pts] Suppose you are given a large dataset file for ML training that is of size 120 GB. What is the lowest possible I/O cost (in GB) of each of the following feature engineering operations? Ignore final output write costs and any potential gains due to caching.

A. Quadratic (order 2) feature interactions
B. Binning a numeric feature into 10 given intervals
C. Whitening a numeric feature
D. One-hot encoding of a categorical feature (assume feature’s domain has only 5000 unique values)

A. 120 GB (just one filescan read)
B. Same as A
C. 240 GB (one filescan read to compute mean and stdev; another filescan read to whiten feature values)
D. Same as A
Exercise

Q6) [4 x 8pts] Write pseudocode (or just describe precisely) using MapReduce/Spark operations to perform the following data science operations at scale:
A. Quadratic (order 2) feature interactions
B. Binning a numeric feature with given bins
C. Whitening a numeric feature
D. One-hot encoding of a categorical feature (assume feature’s domain has only 5000 unique values and is given)

For Input Split, assume data is sharded tuple-wise, as is common.
A. Map-only job suffices. Map() takes feature vector from tuple, performs feature interactions and emits the interacted vector with the same tuple ID.
B. Also a Map-only job. Map() takes feature value from tuple, performs binning based on given bins and emits the same tuple with same tuple ID, except this feature value is now different.
Exercise

Q6) [4 x 8pts] Write pseudocode (or just describe precisely) using MapReduce/Spark operations to perform the following data science operations at scale:
A. Quadratic (order 2) feature interactions
B. Binning a numeric feature with given bins
C. Whitening a numeric feature
D. One-hot encoding of a categorical feature (assume feature’s domain has only 5000 unique values and is given)

For Input Split, assume data is sharded tuple-wise, as is common.
C. 1 MR job + 1 Map-only job:
- First Map() takes feature values from tuple; computes suff. stats for mean and stdev as 3-tuple (1, x, x^2); emits this 3-tuple as value with a single global dummy key.
- Reduce() iterates over all suff. stats 3-tuples to compute global mean and stdev of this feature; emits that 2-tuple as output.
- Second Map() job takes feature value from tuple; whitens its based on (mean, stdev) 2-tuple from prior job; emits same tuple with same tuple ID, except this feature value is now different.
Q6) [4 x 8pts] Write pseudocode (or just describe precisely) using MapReduce/Spark operations to perform the following data science operations at scale:
A. Quadratic (order 2) feature interactions
B. Binning a numeric feature with given bins
C. Whitening a numeric feature
D. One-hot encoding of a categorical feature (assume feature’s domain has only 5000 unique values and is given)

For Input Split, assume data is sharded tuple-wise, as is common.
D. Also a Map-only job. Map() takes feature value from tuple; performs one-hot encoding based on dictionary to map category value to new feature index; obtains the 0-1 representation for that feature (potentially sparse vector); emits same tuple with same tuple ID, except this feature value is now replaced with the 0-1 vector.
Q7) [3pts] Which of the following hyperparameter tuning approaches is the most popular in practice as per surveys?

A. Grid search
B. Random search
C. Hyperband
D. All of A, B, C
E. None of the above
Q8) [4pts] Suppose you are performing model selection for a RandomForest model. For hyper-parameter tuning, you try 3 values of number of trees and 4 values of maximum tree height. To aid your interpretability, you also explore 5 different manually created subsets of features apart from the full feature set.

What is the total number of models built in this model selection workload?

\[ 72 = \text{#trees} \times 3 \times \text{max height} \times 4 \times \text{feature sets} (5+1) \]
(Advanced/Extra Credit; Optional)

Q9) [10pts] You are given a large training dataset of (Y,X1,X2) examples on HDFS for binary classification (i.e., Y = 0 or 1) with two categorical features X1 and X2. The domains of the features are known beforehand as DX1 and DX2 and have only tens of unique values.

Write pseudocode for a single MapReduce job to train a Naive Bayes model. It should be scalable along the number of rows.

Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Hint: Naive Bayes training only needs to estimate the distribution P(Y) and all class-conditional probability distributions P(Xi|Y) using frequency counts.
(Advanced/Extra Credit; Optional) Q9) [10pts] You are given a large training dataset of \((Y, X_1, X_2)\) examples on HDFS for binary classification (i.e., \(Y = 0\) or \(1\)) with two categorical features \(X_1\) and \(X_2\). The domains of the features are known beforehand as \(D_{X_1}\) and \(D_{X_2}\) and have only tens of unique values.

Write pseudocode for a MapReduce job to train a Naive Bayes model. It should be scalable along the number of rows. Make sure to explain your assumption on how the dataset is stored/sharded to begin with.

Hint: Naive Bayes training only needs to estimate the distribution \(P(Y)\) and all class-conditional probability distributions \(P(X_i|Y)\) using frequency counts.

Using the 4-step method, identify the access pattern over \(D\):

Step 1) Computing frequency counts for a probability distribution is akin to a SQL COUNT. So, that just requires a sequential scan over the dataset. Since \(D_{X_1}\) and \(D_{X_2}\) are small and \(Y\) is binary, the prob. distr. stats are small and can all be batched onto one pass over the dataset.
Step 2) The counts we need:

# tuples
# tuples with $Y = 0$
# tuples with $Y = 1$

For each $x_1$ in $D_{x_1}$:

# tuples with $(Y = 0 \& X_1 = x_1)$
# tuples with $(Y = 1 \& X_1 = x_1)$

Likewise for $X_2$ as with $X_1$

All of these are just counts of tuples with predicates applied on the record/tuple’s data. So, they are easily decomposed over the n tuples of $D$. 

(Advanced/Extra Credit; Optional) Q9)
Exercise

(Advanced/Extra Credit; Optional) Q9)

Step 3) The counts we need:
- # tuples
- # tuples with \( Y = 0 \)
- # tuples with \( Y = 1 \)

For each \( x_1 \) in \( D_{x_1} \):
- # tuples with \( (Y = 0 \& X_1 = x_1) \)
- # tuples with \( (Y = 1 \& X_1 = x_1) \)

Likewise for \( X_2 \) as with \( X_1 \)

The aggregation for each count is just one big sum over the records/tuples of \( D \). All counts can be calculated collectively in one pass.
Exercise

(Advanced/Extra Credit; Optional) Q9)

Step 4) The counts we need:
   - # tuples
   - # tuples with Y = 0
   - # tuples with Y = 1
   - For each x1 in Dx1:
     - # tuples with (Y = 0 & X1 = x1)
     - # tuples with (Y = 1 & X1 = x1)
   - Likewise for X2 as with X1

Input Split: Shard the table record-wise.

Map(): Calculates all counts per shard by iterating over the records in it; emit a value (no/dummy key) a vector of partial counts of length 1+2+2*|DX1|+2*|DX2|.

Reduce(): Get Iterator of all partial counts from Mappers; add them to get global counts; divide respective counts to get resp. prob. distr. entries, e.g., \( P[Y=1] = (# \text{tuples with } Y=1) / # \text{tuples}, \text{ etc.} \)