ECE4721J Final

Wang Yangyang

UM-SJTU JI

2022 Summer

Outline

1. **Dimensionality Reduction**
   - Small Data
   - Big Data
2. **Optimization**
   - Small Data
   - Big Data
   - Application
3. **Lab and Homework**
   - Supplements

Outline

PCA

Idea

- Provides the best perspective that emphasises similarities and differences.
- This new perspective combines the original characteristics in order to best summarize the data.

Process

- Standardize the range of variables. $Z_{i,j} = \frac{X_{i,j} - \overline{X}}{\sigma_{X_i}}$
- Determine the relationship between variables by constructing *Covariance Matrix*.
- Identify the principle components by computing *Eigenvalues* and *Eigenvectors* of covariance matrix. Reorder eigenvalues in non-increasing order.

SVD

Basic

If M is an $m \times n$ real matrix we can write $M = U\Sigma V^\top$, where both U and V are rotation matrices, and Σ is diagonal.

$$\circ M : m \times n \quad \circ U : m \times m \quad \circ \Sigma : m \times n \quad \circ V : n \times n$$

- Σ has exactly r strictly positive elements which are the square roots of the r ***eigenvalues*** of $X^\top X$, with corresponding multiplicities. (Elements on the diagonal of Σ: Singular Values)
- The columns of V are eigenvectors of $X^\top X$. (Row of V^T: right singular vectors)
- The columns of U are eigenvectors or $XX^\top$ (Column of U: left singular vectors)

SVD

Properties

- SVD is not unique and exists for any matrix, whether invertible or not
- Singular values can be re-ordered as long as their corresponding singular vectors are re-arranged accordingly
- If $U\Sigma V^\top$ is an SVD for $X^\top$, then $V\Sigma^\top U^\top$ is an SVD for X
- Slower than eigen decomposition
- More ***Numerically Stable*** than eigen decomposition.
  Increase factor:
  - $\left\|P^{-1}\right\| \|P\|$, for eigen decomposition
  - $\left\|U^{-1}\right\| \left\|\left(V^\top\right)^{-1}\right\| = \left\|U^\top\right\| \|V\| = 1$, for SVD

QR Decomposition

Basic

For any $m \times n$ matrix X with linearly independent columns, we can write $X = QR$, where

- Q is an $m \times n$ orthogonal matrix,
- R is an $n \times n$ upper triangular matrix.

Householder Computation

Determine an orthogonal matrix Q_1 such that $Q_1 c_1 = (\gamma_1, 0, \cdots, 0)^\top$. Repeat the above process $t = \min(m-1, n)$ times and obtain $R = Q_t Q_{t-1} \cdots Q_2 Q_1 X$. By orthogonality $Q^{-1} = Q^\top$, yielding

$$X = Q_1^\top \cdots Q_t^\top R = QR$$

Connection

Run SVD using QR

1. Write $X = QR$
2. Perform SVD on $R = U_1 \Sigma V^T$
3. Obtain the SVD for $X = QU_1 \Sigma V^T = U \Sigma V^T$

Run PCA using SVD

Slower but stable than the covariance matrix.
Because $X^\top X = \left(U \Sigma V^\top\right)^\top \left(U \Sigma V^\top\right) = V \Sigma^\top U^\top U \Sigma V^\top = V \Sigma^2 V^\top$,
Principal components are given by $XV = U \Sigma V^T V = U \Sigma$ and the singular values are the square of the eigenvalues.

Outline

Tall and Skinny - Cosine Similarity

Challenge and Preparation

To efficiently compute $X^\top X$, while ensuring the singular values of C are not significantly altered in the process.

- Store matrices row-by-row on disk
- Ensure all elements are in $[-1, 1]$ (divided by the largest element)

MapReduce Process

- Mapper: for all pairs (x_{ij}, x_{ik}) on row i, return the (key, value) pair $((c_j, c_k), x_{ij}x_{ik})$, with probability $\min\left(1, \frac{\gamma}{\|c_j\|\|c_k\|}\right)$
- Reducer: consider all the (key, value) pairs $((c_i, c_j), \langle v_1, \cdots, v_R \rangle)$, R is the number of non-zero products and proceed as follows.
  - If $\frac{\gamma}{\|c_j\|\|c_k\|} > 1$, then return $\frac{1}{\|c_j\|\|c_k\|} \sum_{i=1}^{R} v_i$
  - Otherwise, return $\frac{1}{\gamma} \sum_{i=1}^{R} v_i$

Tall and Skinny - Cosine Similarity

Remarks

- What the reducer returns is a matrix X' which contains the cosine similarities between the columns of X.
  It suffices to define a diagonal matrix D whose elements are exactly $\|c_i\|$, and then compute $X^\top X \approx DX'D$ to recover X.
- $||c_i||$: pre-computed. Requires all-to-all communication.
- The parameter γ can be adjusted depending on what is expected.

Distributed QR

- Obtain the QR decomposition of block $X_{k,k}$
- Adjust all the blocks $X_{k,j}$ on the right of $X_{k,k}$
- Adjust all the blocks $X_{i,k}$ below $X_{k,k}$ and the right blocks $X_{i,j}$
- Repeat for all blocks on the diagonal

Random Projection

Setup and Goals

- Map m points $(u_i)_{1 \leqslant i \leqslant n}$ in $\mathbb{R}^n$ into m points in $\mathbb{R}^d$, with $d \ll n$
  - Ensure that for all i, j, $\|v_i\|_2 \approx \|u_i\|_2$ and $\|v_i - v_j\|_2 \approx \|u_i - u_j\|_2$.
- Project a fixed vector onto a random subspace:
  - Consider m points in $\mathbb{R}^n$ and project them in a k-dim subspace
  - It prevents missing out on "important" coordinates
- The $r_{i,j}$ are independent identically distributed with zero mean, often symmetric with unit variance.

Remarks

- Random projection is likely not projection in the algebraic sense.
- The norm of a projected vector is $\sqrt{\frac{d}{n}}$ in expectation, with an error exponentially small in d and time complexity $\mathcal{O}(mnd)$.
- The projection can be made very sparse with little loss of accuracy and a major speedup.

Dim-Reduction Summary

Preserving important structural properties of the data:

- Tall and skinny can be handled using SVD
- For other cases random projections are a good alternative
- PCA will ensure a good result, random projections might not
- Randomized PCA:
  - ***Randomly project*** X to obtain X'
  - Perform a ***QR decomposition*** on $X' = QR$
  - Compute $B = Q^\top X$
  - Run ***SVD*** on $B = U_1 \Sigma V^\top$
  - Get the SVD of $X \approx QQ^\top X$ by computing $Q\left(U_1 \Sigma V^\top\right) = U \Sigma V^\top$
  - Complete ***PCA*** the usual way

Outline

Gradient Descent

Setup

- Iteratively create a sequence of X_i
- At each step, determine the gradient in each direction
- Select a direction and keep going down
- Repeat until the gradient is 0 in all directions

Remarks

- Function f should be *convex* to ensure a global minimum.
- Using a constant step could lead to either going very slowly or overstepping the minimum
- Computation can be speeded up by Taylor expansion. This involves inverting the Hessian matrix of f which has a cost of $\mathcal{O}\left(n^3\right)$.

Stochastic Gradient Descent

Gradient descent	Stochastic gradient descent
Uses the whole dataset	Randomly selects a sample
Deterministic method	Stochastic method
Slow but fast to converge	Fast but slow to converge
Yields an optimal solution	Yields a good enough solution
Slow to escape local minima	Faster to escape local minima

Outline

Work-depth Model

PRAM

- Dependencies between instructions are represented using a DAG. Depth of DAG is *depth*
- Each instruction is a node, number of node is *work*.
- An edge (u, v) represents the dependency of u upon v
- The root corresponds to the result of the computation

Brent's Theorem

The amount of time when using p CPUs is referred to as T_p

$$\frac{T_1}{p} \leqslant T_p \leqslant \frac{T_1}{p} + T_\infty$$

- If work is evenly shared among all p CPUs then we get $\frac{T_1}{p}$
- T_∞ helps define how far we are from the ideal case

Stochastic vs Batch

F is strongly convex and Lipschitz continuous
m corresponds to the data parallelism
n corresponds to the model parallelism

Gradient Descent

- Per iteration:
Work: $\mathcal{O}(mn)$ Depth: $\mathcal{O}(\log mn)$
- Total:
Work: $\mathcal{O}\left(mn \log \frac{1}{\varepsilon}\right)$ Depth: $\mathcal{O}\left(\log \frac{1}{\varepsilon} \log mn\right)$

Stochastic Gradient Descent

- Per iteration:
Work: $\mathcal{O}(n)$ Depth: $\mathcal{O}(\log n)$
- Total:
Work: $\mathcal{O}\left(\frac{n}{\varepsilon}\right)$ Depth: $\mathcal{O}\left(\frac{\log n}{\varepsilon}\right)$

Parallelism and Lock

Parallelism

- Save w in a shared piece of the memory
- All CPUs have access to w and the whole dataset

However, a model may have been updated by another CPU while being read and written in the memory.

Lock

Only one CPU can access w at a time. It solves the race condition but loses the benefits from parallelism.
In stochastic gradient descent:

- $w = \left(w^{(1)}, \cdots, w^{(n)}\right)$ is a vector of dimension n
- A different $w^{(i)}$ will be randomly chosen for each CPU

Hogwild!

Strategy

- Proceed in parallel over all available CPUs
- Until the expected error condition is met:
- Select a random index i from $\{1, \cdots, m\}$
- Compute $F_i(w)$ and $\nabla F_i(w)$ for w concurrently in the shared memory
- Update $w^{(i)}$ with $w^{(i)} - \alpha \left[\nabla F_i(w)\right]^{(i)}$

Remarks

- No locks are used
- The speedup is nearly linear in term of CPUs
- In general the locking overhead should be avoided when processing big data

Summary for GD and SGD

Looking back at gradient descent in PRAM:

- Stochastic had lower depth but was not easy to parallelize
- Batch could be easily parallelized
- Hogwild! renders stochastic almost embarrassingly parallel

Stochastic and batch on a distributed system:

- Communication at each iteration:
  - Batch: $\mathcal{O}\left(\log \frac{1}{\varepsilon}\right)$
  - Stochastic: $\mathcal{O}\left(\frac{1}{\varepsilon}\right)$
- Stochastic is often preferred on a single computer with many GPUs
- Batch is more common on distributed systems

Outline

Gradient Descent Spark

Gradient Descent

- Organise by row and store in RDD. Generate closure and use cache action.
- For each point p, apply a map to transform p into $\nabla F_p(w)$
- Apply a reduce to sum up all the $\nabla F_p(w)$

Stochastic Gradient Descent (Hogwild!)

- Broadcast needs to be completed to start the mappers
- All mappers and reducers must be done before broadcasting again
- Synchronisation barriers are essential to fault-tolerance

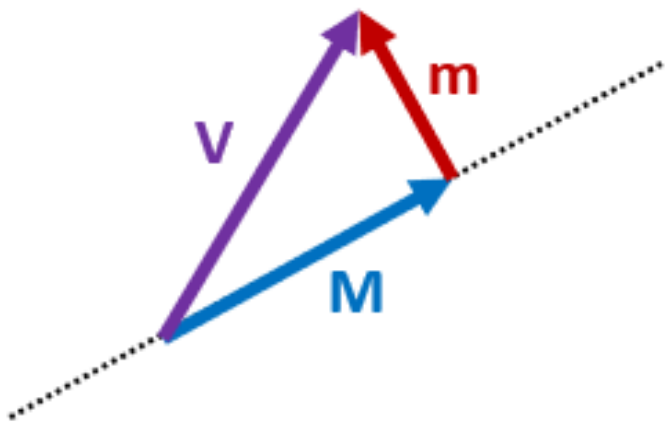
Mini-batches

- At each iteration select "many" samples, instead of one
- Apply batch gradient descent to them

Gradient Descent and PCA

Relating gradient descent to PCA:

- Find the axes which maximize the variance
- Find the direction for which the expectation of $XX^\top$ is maximized
- Find the direction which minimizes the residuals
- For mean centered data:
  - Pythagorean theorem: $V^2 = M^2 + m^2$
  - PCA maximizes M
  - Gradient descent minimizes m



Outline

Lab

Lab5

Docker is a set of platform as a service products that use OS-level virtualization to deliver software in packages called "containers".
A *Docker image* is a file used to execute code in a Docker container.

Lab6

Cholesky decomposition decompose a *symmetric positive* matrix into the product o a lower triangular matrix and its conjugate transpose.

Lab7

How can you look into how principal components explain the dataset?
How would you interpret the result?
python pca_model.explainedVariance
not very large - not much correlation between columns

Homework

Homework4

Basic SQL syntax

Homework5

Comparison between eigen decomposition and SVD

Homework6

Work with PRAM to determine work and depth
Look at how to speed up GD using Hessian
Implement (S)GD using Spark

Reminder

Similar with an Interview.

Tips

If not sure about question or answer, you can:

- Ask for clarification
- Share your relevant thoughts or experience.

You can add something later for previous questions if it was not well answered. But don't speak too many irrelevant things.

Project presentation on Friday. HW6 due on Sunday.
Good Luck!