大数据分析

Scalable Machine Learning

刘盛华

What is machine learning

- **Study of algorithms that**
  - **improve their performance**
  - **at some task**
  - **with experience**

Data → **Learning algorithm** → Understanding

(experience) **(task)** **(performance)**

Barnabás Póczos, CMU

Warnings about the Class

"There is nothing more practical than a good theory"

Lewin (1952)

Linear Algebra

Preliminaries

Vectors

- $\mathbf{x} = (x_1, x_2, \ldots, x_d) \in R^d$ (each x_i is a component)
  - A point in d-dimensional space
- Norm or magnitude $\|\mathbf{x}\| = (x^T x)^{1/2} = (x_1^2 + x_2^2 + \ldots + x_d^2)^{½}$
  - Length of the vector (Pythagorean theorem)
- Zero vector (norm zero), unit vector (norm one)
- Inner product $<\mathbf{x}, \mathbf{y}> = x_1 y_1 + \ldots x_d y_d$
  - Result is a scalar
  - $\|\mathbf{x}\| = (<\mathbf{x}, \mathbf{x}>)^{1/2}$
  - $<\mathbf{x}, \mathbf{y}> = 0$ implies $\mathbf{x} \perp \mathbf{y}$

Vector spaces

- Space where vectors live
- Formally, a collection of vectors which is closed under linear combination
  - If $\{\mathbf{x}, \mathbf{y}\}$ are in the space, so is $a\mathbf{x}+b\mathbf{y}$ for any scalars a, $b \in R$
  - Should always contain zero vector
- Examples: $\{0\}$, R^d, the line $x = 3y$ in R^2

Span and basis

- A set of vectors is said to span a vector space if one can write any vector in the vector space as a linear combination of the set
- $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ span the space $\{\sum a_i \mathbf{x}_i \mid a_i \in R\}$
- This set is called the basis set
- Examples
  - The vectors $\{(0,1), (1,0)\}$ span R^2
  - $\{(1, 1)\}$ spans $x=y$ which is a subspace of R^2
  - The vector $\{(0,1), (0,1), (1,1)\}$ also span R^2

Linear independence and orthonormality

- Linear independence – a notion to remove redundancy in the basis
  - $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ are linearly independent iff the only solution to $\sum a_i \mathbf{x}_i = 0$ is $a_1 = a_2 = \ldots = a_n = 0$. 齐次方程只有0解； 任意元素不能是其他向量的线性组合表示
  - Cannot express any vector $\mathbf{x}_i$ as a linear combination of the others
- Dimensionality of a vector space is the maximum number of linearly independent basis vectors 维度，最大线性无关组向量数量
- Orthonormal basis 标准正交基
  - $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n\}$ is orthonormal basis if $< \mathbf{x}_i , \mathbf{x}_j >= 1$ if $i=j$ and 0 otherwise
  - Coordinate axes for the vector space
- Example: The basis $\{(0, 1), (1,1)\}$ for R^2 is linear independent but not orthonormal.

Matrices

- Operator which transforms vectors from one vector space to another
  - $\mathbf{y} = A\mathbf{x}$
- The operator is linear, that is 线性算子

  $A(a\mathbf{x} + b\mathbf{y}) = a(A\mathbf{x}) + b(A\mathbf{y})$
- The result of applying the operator is a linear combination of the column vectors
  - Thus, $A\mathbf{x} = \mathbf{b}$ has an exact solution iff $\mathbf{b}$ is in the column space of A 列空间（值域空间）
- Eigen vectors of A are the special vectors are the special vectors $\mathbf{x}$ which satisfy

  $A\mathbf{x} = \lambda\mathbf{x}$ for some λ
  - λ is called the eigen value and $\mathbf{x}$ is the eigen vector
- How do we visualize the transformation geometrically?

Visualizing the matrix operator – special cases

- Identity matrix
  - Square matrix with diagonal elements 1 and non-diagonal elements 0
  - The transformed vector $A\mathbf{x}$ is same $\mathbf{x}$
- Diagonal matrix
  - Square matrix with non-diagonal elements 0
  - i^{th} component in $A\mathbf{x}$ is a scaled version of x_i (scaling = A_{ii})
- Orthonormal (or rotation) matrix 标准正交矩阵（旋转矩阵）
  - Matrix whose columns $\{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_n\}$ are such that $< \mathbf{a}_i, \mathbf{a}_j >= 1$ if i=j and 0 otherwise. That is, $A^TA = I$
  - Rotates the vector
  - Preserves norms $\|A\mathbf{x}\| = \|\mathbf{x}\|$ (why?)

General case – Singular Value Decomposition

- We have a rectangular matrix $A \in R^{m \times n}$
- It can be decomposed as

  $A = UDV^T$
- U and V are orthonormal, i.e., $U^TU = V^TV = I$ and D is a diagonal matrix containing singular values
  - Number of non-zero diagonal elements in D = rank of A
- Provides a nice way to understand the operator A
  - Rotation in n-dimensional space, scaling, rotation in m-dimensional space
- Can be computed in $\boxed{O(\min\{mn^2, m^2n\})}$ time (or better using fast matrix multiplication)

Example problem

- If singular values of $A \in R^{n\times n}$ all lie in [a, b], prove that

  $a\|\mathbf{x}\| \le \|A\mathbf{x}\| \le b\|\mathbf{x}\|$

Solution:

- Let $A = UDV^T$
- $\|A\mathbf{x}\| = \|UDV^T\mathbf{x}\|$
- Let $\mathbf{y} = V^T\mathbf{x}$. (note: $\|\mathbf{y}\| = \|\mathbf{x}\|$)
  - We can do this because we prove this for every $\mathbf{x}$
- $\|A\mathbf{x}\| = \|UD\mathbf{y}\| = \|D\mathbf{y}\|$
- As singular values lie in [a, b], $a\|\mathbf{y}\| \le \|D\mathbf{y}\| \le b\|\mathbf{y}\|$

Linear Regression

Sketching

Massive data sets 海量数据集

- Examples
  - Internet traffic logs
  - Financial data
  - etc.
- Algorithms
  - Want nearly linear time or less
  - Usually at the cost of a randomized approximation

Why linear time – big-data:

- $O(N^2)$ algorithms are ~intractable 难的 - N=1B
- N^2 seconds = 31B years (>2x age of universe)

Why linear time – big-data:

- $O(N^2)$ algorithms are ~intractable - N=1B

31M

- N^2 seconds = 31B years
- 1,000 machines

Why linear time – big-data:

- O(N^2) algorithms are ~intractable - N=1B

31K

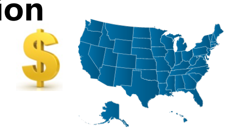
- N^2 seconds = 31B years
- 1M machines

Why linear time – big-data:

- O(N^2) algorithms are ~intractable - N=1B

3

- N^2 seconds = 31B years
- 10B machines ~ $10Trillion



Why linear time – big-data:

- O(N^2) algorithms are ~intractable - N=1B

And parallelism might not help

- N^2 seconds = 31B years
- 10B machines ~ $10Trillion

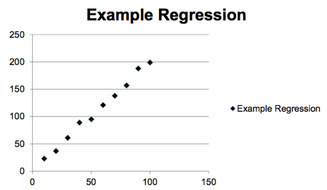
Regression analysis

- Regression analysis
  - Statistical method to study dependencies between variables in the presence of noise.

统计方法研究有噪声变量之间的相关性。

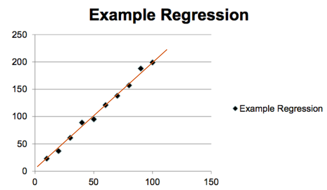
Regression analysis

- Linear Regression
  - Statistical method to study linear dependencies between variables in the presence of noise.
- Example
  - Ohm's law V = R · I



Regression analysis

- Linear Regression
  - Statistical method to study linear dependencies between variables in the presence of noise.
- Example
  - Ohm's law V = R · I
  - Find linear function that best fits the data



Regression analysis

- Linear Regression
  - Statistical method to study linear dependencies between variables in the presence of noise.
- Standard Setting
  - One measured variable b
  - A set of predictor variables $a_1, \ldots, a_d$
  - Assumption:

$$b = x_0 + a_1 x_1 + \ldots + a_d x_d + \varepsilon$$

  - ε is assumed to be noise and the x_i are model parameters we want to learn
  - Can assume $x_0 = 0$
  - Now consider n observations of b

Regression analysis

- Matrix form

Input: n×d-matrix A and a vector $b=(b_1,\ldots, b_n)$
n is the number of observations; d is the number of predictor variables

Output: x^* so that Ax^* and b are close

- Consider the over-constrained case, when $n \gg d$
- Can assume that A has full column rank

Regression analysis

- **Least Squares Method**
  - Find x^* that minimizes $|Ax-b|_2^2 = \Sigma (b_i - \langle A_{i*}, x\rangle)^2$
  - A_{i*} is i-th row of A
  - Certain desirable statistical properties

Regression analysis

- **Geometry of regression**
  - We want to find an x that minimizes $|Ax-b|_2$
  - The product Ax can be written as

$$A_{*1}x_1 + A_{*2}x_2 + \ldots + A_{*d}x_d$$

where A_{*i} is the i-th column of A

  - This is a linear d-dimensional subspace
  - The problem is equivalent to computing the point of the column space of A nearest to b in l_2-norm

Time Complexity

- **Solving least squares regression via the normal equations**
  - **Need to compute $x = A^- b$**
    - **Moore-Penrose Pseudoinverse $A^- = V\Sigma^{-1}U^T$**
  - **Naively this takes nd^2 time**
  - **Can do $nd^{1.376}$ using fast matrix multiplication**
  - **But we want much better running time!**

Sketching to solve least squares regression

- How to find an approximate solution x to $\min_x |Ax-b|_2$?
- Goal: output x' for which $|Ax'-b|_2 \leq (1+\varepsilon) \min_x |Ax-b|_2$ with high probability
- Draw S from a k x n random family of matrices, for a value k << n
- Compute S*A and S*b
- Output the solution x' to $\min_{x'} |(SA)x-(Sb)|_2$
  - $x' = (SA)^- Sb$

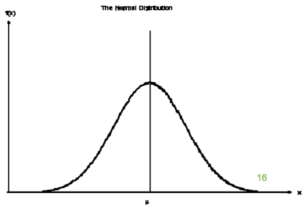
How to choose the right sketching matrix S?

- Recall: output the solution x' to $\min_{x'} |(SA)x-(Sb)|_2$
- Lots of matrices work
- S is $d/\varepsilon^2 \times n$ matrix of i.i.d. Normal random variables

- **S is a subspace embedding**

  For all x, $|SAx|_2 = (1\pm\varepsilon)|Ax|_2$

  * poof skipped



ref: David P. Woodruff, Sketching as a Tool for Numerical Linear Algebra, Foundations and Trends in Theoretical Computer Science, vol 10, issue 1-2, pp. 1-157 (ref to 10-40)

Subspace Embeddings for Regression

- Want x so that $|Ax-b|_2 \leq (1+\varepsilon) \min_y |Ay-b|_2$
- Consider subspace L spanned by columns of A together with b
- Then for all y in L, $|Sy|_2 = (1\pm \varepsilon) |y|_2$
- Hence, $|S(Ax-b)|_2 = (1\pm \varepsilon) |Ax-b|_2$ for all x
- Solve $\text{argmin}_y |(SA)y - (Sb)|_2$
- Given SA, Sb, can solve in poly(d/ε) time

Only problem is computing SA takes $O(nd^2)$ *time*

Faster Subspace Embeddings S

- **CountSketch matrix**
- **Define k x n matrix S, for** $k = O(d^2/\varepsilon^2)$
- **S is really sparse: single randomly chosen non-zero entry per column**

按列随机选一个位置，随机化+-1

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & -1 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Can compute $S \cdot A$ in nnz(A) time!

<< nd < nd²

nnz(A) is number of non-zero entries of A

High Probability and Complexity

- **Theorem 2.5.** ([27]) For $\mathbf{S}$ a sparse embedding matrix with $r = O(d^2/\varepsilon^2 \text{poly}(\log(d/\varepsilon)))$ rows, for any fixed $n \times d$ matrix $\mathbf{A}$, with probability .99, $\mathbf{S}$ is a $(1\pm\varepsilon)$ ℓ_2-subspace embedding for $\mathbf{A}$. Further, $\mathbf{S} \cdot \mathbf{A}$ can be computed in $O(\text{nnz}(\mathbf{A}))$ time.

- **Theorem 2.14.** The ℓ_2-Regression Problem can be solved with probability .99 in $O(\text{nnz}(A)) + \text{poly}(d/\varepsilon)$ time.