Proof sketch for JL Transform

Going back to JL Transform…

As before, let $Y = X_1 - X_2$

$$\Pr\left[\left|\|SY\|^2 - E\left[\|SY\|^2\right]\right| \geq \epsilon\|Y\|^2\right] \leq ?$$

Here $\|SY\|^2 = \sum_{i=1}^{k} U_i^2$ where $U_i \sim N\left(0, \frac{\|Y\|^2}{k}\right)$

From Lemma 1, $-\frac{k\epsilon^2}{c}$

$$\text{L.H.S.} \leq e$$

Q: What do we want to this value to be? (Think about one way of getting to the w.h.p (i.e. w.p 1/n) result)

Proof sketch for JL Transform

Going back to JL Transform…

$$\Pr\left[\left|\|SY\|^2 - E\left[\|SY\|^2\right)\right| \geq \epsilon\|Y\|^2\right] \leq e^{-\frac{k\epsilon^2}{c}}$$

choose

$$k = \frac{2c\log n}{\epsilon^2}$$

$$\leq \frac{1}{n^2}$$

Next by a union bound, all nChoose2 pairs of squared lengths are maintained with probability at least 1 − nChoose2 *1/n^2 ≥ ½

Shows that a randomized construction works with constant prob!

Johnson-Lindenstrauss Transform

So all pairwise distances preserved with prob. >= 1/2

We can check if a chosen transform is good. And keep repeating until we get a good one.

OR

Can increase K by a constant factor and make the probability of a single pair-wise distance not maintained $\leq \frac{1}{n^3}$.

Then, prob. of pairwise distances not maintained for any of the pairs $\leq \binom{n}{2} \cdot \frac{1}{n^3} \leq \frac{1}{n}$

Discussion

- Runtime:
  - Draw KD = $O\left(\frac{D\ logn}{\varepsilon^2}\right)$ Normal random variables to form A
  - Perform AX for each of the vectors (takes O(KDn) time)

- Can make the computations significantly faster:
  - Currently A is a dense matrix
  - Can transform X with a Fourier Transform (using Fast Fourier Transform) and this allows making A sparse
  - "Fast JL Transform"

- Instead of the entries of the K × D matrix A being Gaussians, we could have chosen them to be unbiased {−1, +1} R.V.s

Discussion

Use cases:

- Nearest neighbor search (PCA won't work since it does not maintain pairwise distances)
- Fast (but approximate) matrix multiplication
  - Preprocess the matrices with JL transform

$M_1^T M_2 \quad M_i \in \mathbb{R}^{D \times D} \longrightarrow$ Strassen take $D^{2.8}$

Use $\left(\frac{1}{\sqrt{k}} A M_1\right)^T \left(\frac{1}{\sqrt{k}} A M_2\right)$

gives good approximation

Takes kD^2 time for dim-red. $\quad AM_i$: $k \times D \quad D \times D$

& then multiply two matrices $(D \times k) \times (k \times D)$

$= O(kD^2)$ If $D \gg k$ then significantly better.

Discussion

Use cases (continued):

- Dimension reduction very useful in other settings as well. For example, in “Compressive Sensing”
  - Learn sparse vectors by taking only a few linear combination measurements
  - Shows up in a variety of applications, e.g., MRI, camera

Discussion

- The final dimension is independent of the original dimension D (can be very large) and is *dependent only on the number of points n and the accuracy parameter ε*

- Plus: Data independent ☺ So universal!
- Minus: Data independent ☹ So does not exploit special structure in the data (if present). E.g., data points could be lying in a low-dimensional plane

This is where PCA comes in…

PCA is a data dependent transform.

15-750: Graduate Algorithms

Dimensionality Reduction:

Johnson-Lindenstrauss Transform

Principal Component Analysis

Principal Component Analysis

In JL Transform, we did not assume any structure in the data points. Oblivious to the dataset. Cannot exploit any structure.

Toy examples:

- Points lying on one of axes
- Points lying on a line

- Need not necessarily be exact: What if the dataset is well-approximated by a low-dimensional affine subspace?
  - (will be formalized soon)

Applications

- Analysis of genome data and gene expression levels in the field of bioinformatics
  - Gene microarray data: Microarrays measure activity levels of a large number of genes, say D = 10, 000 genes. After testing m individuals, one obtains m vectors in R^D.
  - In practice it is found that this gene expression data is low-dimensional (some biological phenomenon that activates multiple genes at a time).

- Denoising of stock market signals

Principal Component Analysis

Dataset $X = \{x_1, x_2, \ldots, x_n\} \subseteq R^D$

For some small $K < D$, find vectors $v_1, v_2, \ldots, v_K \in R^D$ such that every x_i is close to the span of $v_1, v_2, \ldots, v_K$.

Principal Component Analysis

The goal of PCA is to find k (orthonormal) vectors such that the points in the datasets have a good approximation in the subspace generated these vectors

(We look for orthonormal vectors since we want the basis vectors for the low-dimensional space)

Good approximation: in the L2 sense

the L2 distance (a.k.a. mean squared error) between the given points and their closest approximation in the low-dimensional subspace obtained is minimized

Principal Component Analysis

Minimizing the L2 distance (a.k.a. mean squared error) between the given points and their closest approximation

= <on board>

Principal Component Analysis

Thus, Minimizing l2 error of approximation = maximizing the projected distances

That is, PCA maximizes the variance of the projected points

Let us first go through the 1 dimensional case for intuition

Principal Component Analysis: Preprocessing

PCA is very sensitive to scaling

Data needs to be preprocessed before performing PCA

- Data needs to be mean zero (since looking for linear subspace, i.e., through the origin)
  - Achieved by subtracting by sample mean
- Each coordinate needs to be scaled appropriately so that they are comparable
  - Empirically, dividing each coordinate (column) by sample standard deviation has been found to perform well

Solving PCA: 1-dimensional case

Given a unit vector u and a point x, the length of the projection of x onto u is given by $x^T u$

To maximize the projected distances:

$$\sum_{i=1}^{n} (x_i^T u)^2 = \sum_{i=1}^{n} u^T x_i x_i^T u$$

$$= u^T \left(\sum_{i=1}^{n} x_i x_i^T \right) u = u^T M u$$

(M is symmetric)

$$\underset{u \in \mathbb{R}^D, \|u\|_2 = 1}{\operatorname{argmax}} \; u^T M u = \text{principal eigen vector of } M$$

(Linear algebra)

Solving PCA: K-dimensional case

Let $M = \sum_{i=1}^{n} x_i x_i^T$ (DxD)

EVD of $M = V \Lambda V^T$ where $\Lambda = \begin{bmatrix} \lambda_1 & & \\ & \lambda_2 & \\ & & \ddots \end{bmatrix}$

Let $\lambda_1 \geq \lambda_2 \geq \lambda_3 \cdots$

Let $v_1, v_2, v_3 \cdots v_k$ be the first k columns of V

Then $v_1, v_2, v_3 \cdots v_k$ are the k orthonormal vectors that maximize the projected distances (i.e variance)

Solving PCA: K-dimensional case

Related note:

$M = X^T X$

All eigen values are non-negative

Such matrices are called “positive semidefinite”

PCA Algorithm

Preprocess the data

Compute the "covariance matrix" M = $\sum_{i=1}^{n} x_i x_i^T$

Find eigen value decomposition of M = $V \Lambda V^T$

where $\Lambda = \text{diag}(\lambda_1, \lambda_2 \dots)$ and $\lambda_1 \geq \lambda_2 \geq \lambda_3 \dots$

$V = [v_1 \; v_2 \; \dots \; v_D]$

Set the linear transformation matrix as $S = \begin{bmatrix} v_1^T \\ v_2^T \\ \vdots \\ v_k^T \end{bmatrix}$

Relation between PCA and SVD

Recall $M = X^T X$ where X is the matrix of given data points

Any matrix X, can be written as $X = U\Sigma V^T$

called "Singular Value Decomposition" where,

U is (n x r) , V is (r, D), both have columns orthonormal

Columns of U = right singular vectors of X

Columns of V = left singular vectors of X

Σ is (r x r) diagonal matrix

Diagonal elements of Σ are called the singular values of X

r = rank for X

To be continued in the next lecture…