Lecture 19: Kernel PCA, Random projections, tSNE, Isomaps

Nisha Chandramoorthy

November 2, 2023

Last time: PCA

- when E and D are linear $\rightarrow$ PCA.

Last time: PCA

- when E and D are linear $\rightarrow$ PCA.
- $E(x) = Wx$, $D(z) = W^{\top}z$.

Last time: PCA

- when E and D are linear $\to$ PCA.
- $E(x) = Wx$, $D(z) = W^\top z$.
- Let $C = \sum_{i=1}^{m} x_i x_i^\top = X^\top X$ be the data correlation matrix, neglecting the $1/m$ factor.

Last time: PCA

- when E and D are linear $\to$ PCA.
- $E(x) = Wx$, $D(z) = W^\top z$.
- Let $C = \sum_{i=1}^{m} x_i x_i^\top = X^\top X$ be the data correlation matrix, neglecting the $1/m$ factor.
- C is symmetric and positive semi-definite, $C = V \Lambda V^\top$.
- Theorem PCA: among linear hypothesis classes, $E^* = V^\top$, $D^* = V$, where V is the matrix of eigenvectors of $C = X^\top X$.

Linear algebra review: Rayleigh Quotient

- For a square matrix $A \in \mathbb{R}^{d \times d}$, the Rayleigh quotient is a scalar function,

$$r(x) = \frac{x^\top A x}{x^\top x}.$$

Linear algebra review: Rayleigh Quotient

- For a square matrix $A \in \mathbb{R}^{d \times d}$, the Rayleigh quotient is a scalar function,

$$r(x) = \frac{x^\top A x}{x^\top x}.$$

- Eigenvalues of A are the stationary points of $r(x)$.

Linear algebra review: Rayleigh Quotient

- For a square matrix $A \in \mathbb{R}^{d \times d}$, the Rayleigh quotient is a scalar function,

$$r(x) = \frac{x^\top A x}{x^\top x}.$$

- Eigenvalues of A are the stationary points of $r(x)$.
- $\nabla r(x) = \frac{2}{x^\top x}(Ax - r(x)x)$.

PCA by SVD

- When $m > d$, do eigenvalue decomposition of $X^\top X$ or SVD of X.

PCA by SVD

- When $m > d$, do eigenvalue decomposition of $X^\top X$ or SVD of X.
- When $m < d$, do eigenvalue decomposition of $XX^\top$. If $v_1, v_2, \cdots, v_n$ are the n largest eigenvectors, principal vectors are $\frac{1}{\|X^\top v_i\|} X^\top v_i$.

PCA by SVD

- When $m > d$, do eigenvalue decomposition of $X^\top X$ or SVD of X.
- When $m < d$, do eigenvalue decomposition of $XX^\top$. If $v_1, v_2, \cdots, v_n$ are the n largest eigenvectors, principal vectors are $\frac{1}{\|X^\top v_i\|} X^\top v_i$.
- Computational complexity: $O(\min(m^2 d, md^2))$.

PCA properties

- Exact when $X^\top X$ is rank n.

PCA properties

- Exact when $X^\top X$ is rank n.
- Maximizes variance. Let x be a random vector chosen uniformly from centered data $x_1, \cdots, x_m$. Then, for any $w \in \mathbb{R}^d$,
$$\mathrm{var}(x \cdot w) = \frac{1}{m}\sum_{i=1}^{m}(w \cdot x_i)^2.$$

PCA properties

- Exact when $X^\top X$ is rank n.
- Maximizes variance. Let x be a random vector chosen uniformly from centered data $x_1, \cdots, x_m$. Then, for any $w \in \mathbb{R}^d$,
$$\mathrm{var}(x \cdot w) = \frac{1}{m} \sum_{i=1}^{m} (w \cdot x_i)^2.$$
- First principal component maximizes $\mathrm{var}(x \cdot w)$ over all w with $\|w\| = 1$. (See Ex 23.4 in book.)

PCA properties

- Exact when $X^\top X$ is rank n.
- Maximizes variance. Let x be a random vector chosen uniformly from centered data $x_1, \cdots, x_m$. Then, for any $w \in \mathbb{R}^d$,

$$\mathrm{var}(x \cdot w) = \frac{1}{m} \sum_{i=1}^{m} (w \cdot x_i)^2.$$

- First principal component maximizes $\mathrm{var}(x \cdot w)$ over all w with $\|w\| = 1$. (See Ex 23.4 in book.)
- Informally, PCA rotates the data so that the variance is maximized along the first axis, then the second, and so on.

PCA properties

- Exact when $X^\top X$ is rank n.
- Maximizes variance. Let x be a random vector chosen uniformly from centered data $x_1, \cdots, x_m$. Then, for any $w \in \mathbb{R}^d$,
$$\mathrm{var}(x \cdot w) = \frac{1}{m} \sum_{i=1}^{m} (w \cdot x_i)^2.$$
- First principal component maximizes $\mathrm{var}(x \cdot w)$ over all w with $\|w\| = 1$. (See Ex 23.4 in book.)
- Informally, PCA rotates the data so that the variance is maximized along the first axis, then the second, and so on.
- Separates dissimilar points

Kernel PCA

- Let V be the matrix of the top n eigenvectors of $K = XX^\top \in \mathbb{R}^{m \times m}$.

Kernel PCA

- Let V be the matrix of the top n eigenvectors of $K = XX^\top \in \mathbb{R}^{m\times m}$.
- Then, principal vectors are $d_i^* = \frac{1}{\|X^\top v_i\|} X^\top v_i$, $i = 1, 2, \cdots, n$.

Kernel PCA

- Let V be the matrix of the top n eigenvectors of $K = XX^\top \in \mathbb{R}^{m\times m}$.
- Then, principal vectors are $d_i^* = \frac{1}{\|X^\top v_i\|} X^\top v_i$, $i = 1, 2, \cdots, n$.
- For some PD kernel, if $K_{ij} = k(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle = (XX^\top)_{ij}$, can compute K only using kernel evaluations.

Graphical reprensentation of X

- Choose weighting, such as, $w_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$. As $\sigma \to 0$, $w_{ij} \to \mathbb{1}_{i=j}$. The $m \times m$ matrix W is the adjacency matrix of a graph.

Graphical reprensentation of X

- Choose weighting, such as, $w_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$. As $\sigma \to 0$, $w_{ij} \to \mathbb{1}_{i=j}$. The $m \times m$ matrix W is the adjacency matrix of a graph.
- Let D be the diagonal matrix with $D_{ii} = \sum_{j=1}^{m} w_{ij}$.

Graphical reprensentation of X

- Choose weighting, such as, $w_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$. As $\sigma \to 0$, $w_{ij} \to \mathbb{1}_{i=j}$. The $m \times m$ matrix W is the adjacency matrix of a graph.
- Let D be the diagonal matrix with $D_{ii} = \sum_{j=1}^{m} w_{ij}$.
- Graph laplacian: $L = D - W$.

Graphical reprensentation of X

- Choose weighting, such as, $w_{ij} = \exp(-\|x_i - x_j\|^2/2\sigma^2)$. As $\sigma \to 0$, $w_{ij} \to \mathbb{1}_{i=j}$. The $m \times m$ matrix W is the adjacency matrix of a graph.
- Let D be the diagonal matrix with $D_{ii} = \sum_{j=1}^{m} w_{ij}$.
- Graph laplacian: $L = D - W$.
- Detects local structure / clusters in data.

Laplacian eigenmaps

- Want to solve: $\min_{y_1,\cdots,y_m} \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \|y_i - y_j\|^2$.

Laplacian eigenmaps

- Want to solve: $\min_{y_1,\cdots,y_m} \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \|y_i - y_j\|^2$.
- optimal embeddings: $y_i = E(x_i) = U[i, -n :]$ where U is the matrix of eigenvectors of L.

Laplacian eigenmaps

- Want to solve: $\min_{y_1,\cdots,y_m} \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \|y_i - y_j\|^2$.
- optimal embeddings: $y_i = E(x_i) = U[i, -n:]$ where U is the matrix of eigenvectors of L.
- For any vector v, $v^\top L v = (1/2) \sum_{i,j=1}^{m} w_{ij} (v_i - v_j)^2$.

Laplacian eigenmaps

- Want to solve: $\min_{y_1,\cdots,y_m} \sum_{i=1}^{m} \sum_{j=1}^{m} w_{ij} \|y_i - y_j\|^2$.
- optimal embeddings: $y_i = E(x_i) = U[i, -n:]$ where U is the matrix of eigenvectors of L.
- For any vector v, $v^\top L v = (1/2) \sum_{i,j=1}^{m} w_{ij}(v_i - v_j)^2$.
- L is positive semi-definite.

Bottom n eigenvectors

- Rayleigh quotient optimality

Bottom n eigenvectors

- Rayleigh quotient optimality
- Another interpretation: top n eigenvectors of $L^\dagger$. $L^\dagger_{ij}$ represents expected time for random walk $i \to j \to i$.

Bottom n eigenvectors

- Rayleigh quotient optimality
- Another interpretation: top n eigenvectors of $L^\dagger$. $L^\dagger_{ij}$ represents expected time for random walk $i \to j \to i$.
- Kernel PCA with $K = L^\dagger$ is equivalent to Laplacian eigenmaps.

Stochastic neighbor embedding [Hinton and Roweis 2002]

- Stochastic neighbor embedding(SNE): conditional probability that x_i would pick x_j as its neighbor, given by

$$p(x_j|x_i) = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}.$$

Stochastic neighbor embedding [Hinton and Roweis 2002]

- Stochastic neighbor embedding(SNE): conditional probability that x_i would pick x_j as its neighbor, given by

$$p(x_j|x_i) = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}.$$

- For the embeddings $y_i = E(x_i)$,

$$q(y_j|y_i) = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i}\exp(-\|y_i - y_k\|^2)}.$$

Stochastic neighbor embedding [Hinton and Roweis 2002]

- Stochastic neighbor embedding(SNE): conditional probability that x_i would pick x_j as its neighbor, given by

$$p(x_j|x_i) = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}.$$

- For the embeddings $y_i = E(x_i)$,

$$q(y_j|y_i) = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i}\exp(-\|y_i - y_k\|^2)}.$$

- SNE minimizes $\sum_{i=1}^{m} D_{\mathrm{KL}}(p_i\|q_i)$, where p_i and q_i are the conditional probabilities of x_i and y_i respectively.

Stochastic neighbor embedding [Hinton and Roweis 2002]

- Stochastic neighbor embedding(SNE): conditional probability that x_i would pick x_j as its neighbor, given by

$$p(x_j|x_i) = \frac{\exp(-\|x_i - x_j\|^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-\|x_i - x_k\|^2/2\sigma_i^2)}.$$

- For the embeddings $y_i = E(x_i)$,

$$q(y_j|y_i) = \frac{\exp(-\|y_i - y_j\|^2)}{\sum_{k\neq i}\exp(-\|y_i - y_k\|^2)}.$$

- SNE minimizes $\sum_{i=1}^{m} D_{\mathrm{KL}}(p_i\|q_i)$, where p_i and q_i are the conditional probabilities of x_i and y_i respectively.
- Penalizes large distances between x_i and x_j but also preserves local structure.

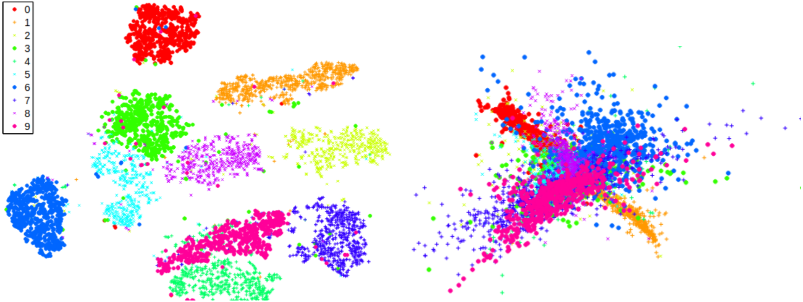
tSNE [Van der Maaten and Hinton 2008]

- tSNE cost function is $D_{\mathrm{KL}}(p \| q) = \sum_{i=1}^{m} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$, where $p_i j$ and q_{ij} are the joint probabilities of (x_i, x_j) and (y_i, y_j) respectively.
- Changes joint distribution to a heavy-tailed distribution, $q(y_j, y_i) = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i}(1+\|y_i - y_k\|^2)^{-1}}$.

tSNE [Van der Maaten and Hinton 2008]

- tSNE cost function is $D_{\mathrm{KL}}(p \| q) = \sum_{i=1}^{m} \sum_{j \neq i} p_{ij} \log \frac{p_{ij}}{q_{ij}}$, where $p_i j$ and q_{ij} are the joint probabilities of (x_i, x_j) and (y_i, y_j) respectively.
- Changes joint distribution to a heavy-tailed distribution, $q(y_j, y_i) = \frac{(1+\|y_i - y_j\|^2)^{-1}}{\sum_{k \neq i}(1+\|y_i - y_k\|^2)^{-1}}$.
- approaches inverse square law on embedded space.

tSNE visualization



From Van der Maaten and Hinton 2008. tSNE (left) and LLE (right) on MNIST dataset.

Convolutional Neural Networks (source: cs231n.stanford.edu)

- Suitable for image recognition. Won the 2012 ImageNet competition and subsequent ones.
- Three types of layers: convolutional, FC, pooling
- Convolutional layer: accepts a volume of size $W_1 \times H_1 \times D_1$ and outputs a volume of size $W_2 \times H_2 \times D_2$ where $W_2 = (W_1 - F + 2P)/S + 1$ and $H_2 = (H_1 - F + 2P)/S + 1$ and $D_2 = K$.
- K is number of filters, F is filter size, S is stride, P is padding.
- Pooling layer: downsamples along width and height, and optionally along depth.
- FC layer: computes class scores, resulting in volume of size $1 \times 1 \times K$.