Machine Learning

4771

Instructors:
Adrian Weller and Ilia Vovsha

Lecture 15: PCA & K-Means Clustering

- Principal Component Analysis (PCA) (Duda 3.8, Bishop 12.1)
- K-Means Clustering (Bishop 9.1)

Dimensionality Reduction

- Problem: data might have excessive dimensionality
- Not just a computational issue! May worsen even very effective algorithms (e.g. similarity measure between examples can be adversely affected)
- Solution: reduce data dimensionality by removing (redundant) features or combining them
- Idea: project high-dimensional data onto a lower dimensional space
- How to project data? What should the projection be?
    a. Best representation of the data in some sense (Principal Component Analysis)
    b. Best separation of the data (Multiple Discriminant Analysis)

Principal Component Analysis (PCA)

- Given a set of vectors, each with dimensionality = d, we wish to project the data onto a subspace of dimensionality M < D
- Goal: maximize the variance of the projected data
- Two cases:
  1. M is given a priori
  2. We choose M based on some criteria

$$\{x_1,\ldots,x_N\}, \quad x_i \in \Re^D$$

$$\Downarrow$$

$$\{p_1,\ldots,p_N\}, \quad p_i \in \Re^M$$

PCA (M = 1)

- Suppose M = 1: w.l.o.g choose a unit vector v_1 which defines the direction of the projected space
- Each data point {x} is then projected onto a scalar value (since M=1): $p_i = v_1^T x_i$
- The mean and variance of the projected data is given by:

$$\bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_N$$

$$mean\{p_1,\ldots,p_N\} = v_1^T \bar{x}$$

$$\mathrm{var}\{p_1,\ldots,p_N\} = \frac{1}{N}\sum_{i=1}^{N}\left(v_1^T x_i - v_1^T \bar{x}\right)^2$$

$$= v_1^T\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T\right]v_1$$

PCA (M = 1)

- Let S denote the covariance matrix:

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T, \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

$$mean\{p_1,\ldots,p_N\} = v_1^T \bar{x}$$

$$\mathrm{var}\{p_1,\ldots,p_N\} = \frac{1}{N}\sum_{i=1}^{N}\left(v_1^T x_i - v_1^T \bar{x}\right)^2$$

$$= v_1^T\left[\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T\right]v_1$$

$$= v_1^T S v_1$$

PCA (M = 1)

- Goal: maximize projected variance with respect to v_1 :
- Solution: constrained maximization (normalization condition on vector v)

$$S = \frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})(x_i - \bar{x})^T, \quad \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i, \quad v_1^T v_1 = 1$$

$$\mathrm{var}\{p_1, \ldots, p_N\} = v_1^T S v_1$$

$$Goal: \quad \max\left\{v_1^T S v_1 + \lambda_1\left(1 - v_1^T v_1\right)\right\}$$

- Setting derivative w.r.t to v_1 equal to zero, we obtain:

$$S v_1 - \lambda_1 v_1 = 0 \Rightarrow S v_1 = \lambda_1 v_1$$

1st Principal Component

- We have observed (1) that the vector v_1 must be an eigenvector of the covariance matrix S
- The variance is given by the corresponding eigenvalue (2)
- The variance is maximum when we choose the eigenvector corresponding to the largest eigenvalue
- This eigenvector is called the first principal component

$$(1) \quad Sv_1 = \lambda_1 v_1$$

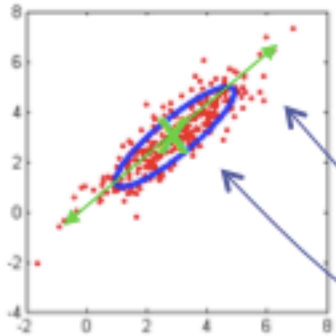
$$(2) \quad v_1^T S v_1 = \lambda_1$$

PCA (M < D)

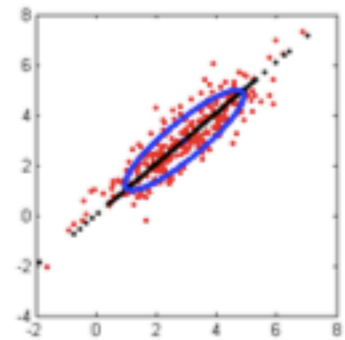
- We can define additional principal components in an incremental fashion:
    1. Compute the covariance matrix S (requires evaluating the data mean)
    2. Find M eigenvectors which correspond to the M largest eigenvalues
    3. Project the data onto the M principal components (eigenvectors)
- We proved the idea for M = 1. For M > 1, shown by induction.
- How do we choose M?

Principal Components Analysis

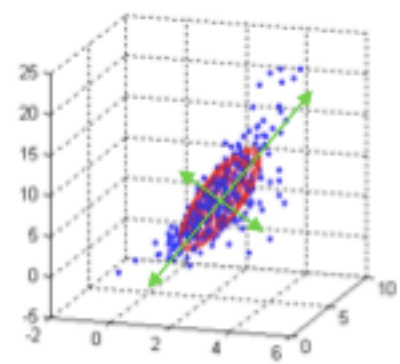
- Idea: instead of writing data in all its dimensions, only write it as mean + steps along one direction



$$\begin{bmatrix} x_i \\ y_i \end{bmatrix} \approx \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix} + c_i \begin{bmatrix} v_x \\ v_y \end{bmatrix}$$



- More generally, keep a subset of dimensions C from D (i.e. 2 of 3)



$$\vec{x}_i \approx \vec{\mu} + \sum_{j=1}^{C} c_{ij} \vec{v}_j$$

- Compression method: $\vec{x}_i \gg \vec{c}_i$
- Optimal directions: along eigenvectors of covariance
- Which directions to keep: highest eigenvalues (variances)

Principal Components Analysis

- If we have eigenvectors, mean and coefficients:

$$\vec{x}_i \approx \vec{\mu} + \sum_{j=1}^{C} c_{ij}\vec{v}_j$$

- Get eigenvectors (use eig() in Matlab): $\Sigma = V \Lambda V^T$

$$\begin{bmatrix} \Sigma(1,1) & \Sigma(1,2) & \Sigma(1,3) \\ \Sigma(1,2) & \Sigma(2,2) & \Sigma(2,3) \\ \Sigma(1,3) & \Sigma(2,3) & \Sigma(3,3) \end{bmatrix} = \begin{bmatrix} [\vec{v}_1] & [\vec{v}_2] & [\vec{v}_3] \end{bmatrix} \begin{bmatrix} \lambda_1 & 0 & 0 \\ 0 & \lambda_2 & 0 \\ 0 & 0 & \lambda_3 \end{bmatrix} \begin{bmatrix} [\vec{v}_1] & [\vec{v}_2] & [\vec{v}_3] \end{bmatrix}^T$$

- Eigenvectors are orthonormal: $\vec{v}_i^T \vec{v}_j = \delta_{ij}$
- In coordinates of v, Gaussian is diagonal, cov = Λ
- All eigenvalues are non-negative $\lambda_i \geq 0$
- Higher eigenvalues are higher variance, use the top C ones

$$\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq \lambda_4 \geq \ldots$$

- To compute the coefficients: $c_{ij} = \left(\vec{x}_i - \vec{\mu}\right)^T \vec{v}_j$

Eigenfaces

$$\left\{x_1, \ldots, x_N\right\}$$

$\vec{\mu}$ $\vec{v}_1$ $\vec{v}_C$

ENCODE

$$\boxed{c_{ij} = \left(\vec{x}_i - \vec{\mu}\right)^T \vec{v}_j}$$

DECODE

$$\left\{\left(\hat{x}_1 = \mu + \sum_{j=1}^{C} c_{1j}\vec{v}_j\right), \ldots, \left(\hat{x}_N = \mu + \sum_{j=1}^{C} c_{Nj}\vec{v}_j\right)\right\}$$

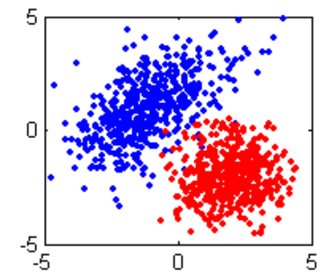
Clustering

- Problem: identify groups, or clusters, of data points in a multidimensional space

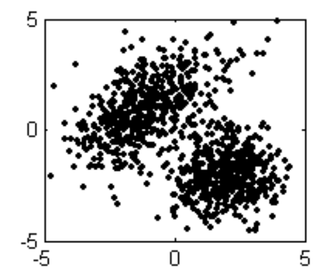
- Data is not labeled (*unsupervised* setting)

- Data is cheaper to obtain (no *annotation* needed)

- Goal: partition the data set into some number K of clusters



- Idea: a cluster is a group of data points whose inter-point distance are small compared with the distances to points outside of the cluster



- If K (# clusters) is not given a-priori, how do we choose K?

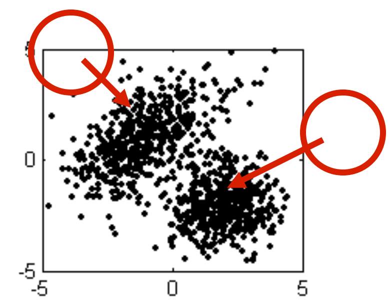
- What should be the approach/criteria to partition the data?

K-Means (idea)

- Given a set of vectors, each with dimensionality = d, we wish to partition the data into K clusters (where we assume K is given)
- Idea: introduce a prototype vector μ_k which represents the center of each cluster, and find
    a. An assignment of data points to clusters
    b. The set of vectors $\{\mu_k\}$
- Objective: minimize sum of squared distances of each data point to its closest center

$$\{x_1,\ldots,x_N\}, \quad x_i \in \Re^D$$

$$\{\mu_1,\ldots,\mu_K\}, \quad \mu_i \in \Re^D$$



K-Means (formal definition)

- For each data point, introduce binary indicator variables which denote whether the point belongs to a cluster:

$$x_i \to r_{i,k} \in \{0,1\} \quad k = 1,\ldots,K$$

- Define an objective function (sum of squared distances of each data point to its assigned cluster):

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{i,k} \left\| x_i - \mu_k \right\|^2$$

- Goal: find $\{r_{i,k}\}$ and $\{\mu_k\}$ which minimize ***J***
- Solution: iterative procedure

Iterative Procedure

1. Initialize $\{\mu_k\}$ to some (random) values
2. E-Step: Minimize J with respect to $\{r_{i,k}\}$, keeping the $\{\mu_k\}$ fixed
3. M-Step: Minimize J with respect to $\{\mu_k\}$, keeping the $\{r_{i,k}\}$ fixed
4. Repeat steps (2),(3) until convergence

- Steps (2-3) correspond to the Expectation and Maximization steps in the EM algorithm

K-Means (E Step)

- Since ***J*** is linear in $\{r_{i,k}\}$, and the terms are independent, we simply assign each data point to the closest cluster center:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{i,k} \left\| x_i - \mu_k \right\|^2$$

$$r_{i,k} = \begin{cases} 1 & if \ \ k = \arg\min_j \left\| x_i - \mu_j \right\|^2 \\ 0 & \text{otherwise} \end{cases}$$

K-Means (M Step)

- Since J is quadratic in $\{\mu_k\}$, it can be minimized by setting the derivative to zero and solving for $\{\mu_k\}$:

$$J = \sum_{i=1}^{N} \sum_{k=1}^{K} r_{i,k} \left\| x_i - \mu_k \right\|^2$$

$$\frac{\partial J}{\partial \mu_k} = 2 \sum_{i=1}^{N} r_{i,k} \left(x_i - \mu_k \right) = 0$$

$$\mu_k = \frac{\sum_{i=1}^{N} r_{i,k} x_i}{\sum_{i=1}^{N} r_{i,k}}$$

- Observe: μ_k is set to the mean of all points assigned to cluster k

K-Means Algorithm

1. Initialize $\{\mu_k\}$ to some (random) values
2. E-Step: assign each data point to a cluster

$$r_{i,k} = \begin{cases} 1 & \text{if } k = \arg\min_j \left\| x_i - \mu_j \right\|^2 \\ 0 & \text{otherwise} \end{cases}$$

3. M-Step: update means for all clusters

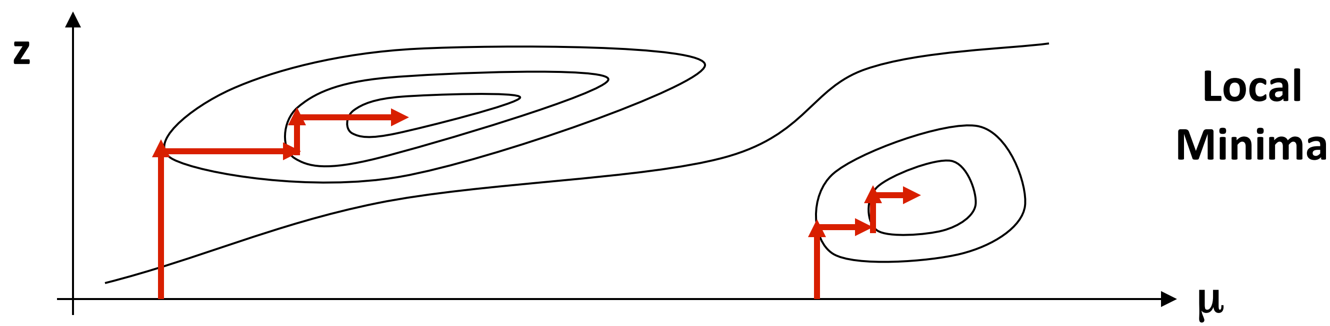
$$\mu_k = \frac{\sum_{i=1}^{N} r_{i,k} x_i}{\sum_{i=1}^{N} r_{i,k}}$$

4. Repeat steps (2-3) until convergence

K-Means Convergence

- How do we know that the algorithm converges?
- Each iteration reduces the value of the objective function ***J***
- May converge to local rather than global minimum
- When do we stop iterating?
    a. No further changes in assignment of points to clusters
    b. Limit on # of iterations exceeded

- Optimization procedure known as *Coordinate Descent* (fix one variable, optimize the other). Other terms in the literature: *Axis Parallel Optimization, Alternating Optimization*



Lossy Data Compression

- Clustering can be used to perform data compression
- If we cannot reconstruct the original data exactly from the compressed representation, we have *lossy data compression*
- K-Means to compress data (sometimes known as *vector quantization*):

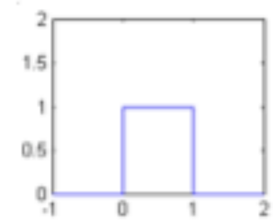
  1. Specify K << N and run the K-means algorithm on your data
  2. For each data point, store only the identity k of the cluster to which it was assigned
  3. Store the K cluster centers $\{\mu_k\}$

Side Note: Sampling from a Gaussian

- Sampling! Generating discrete data easy:

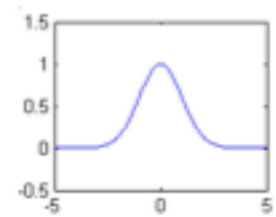
0.73	0.1	0.17

- Assume we can do uniform sampling:
  i.e. rand between (0,1)
  if 0.00 <= rand < 0.73 get A
  if 0.73 <= rand < 0.83 get B
  if 0.83 <= rand < 1.00 get C

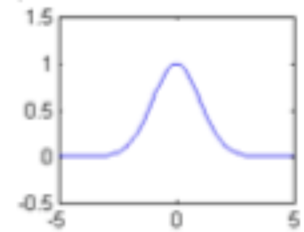


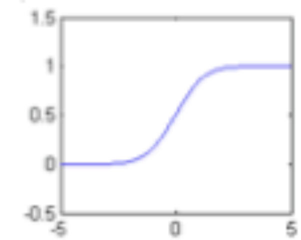
0.73	0.83	1.00

- What are we doing?
  Sum up the Probability Density Function (PDF)
  to get Cumulative Density Function (CDF)
- For 1d Gaussian, Integrate Probability Density
  Function get Cumulative Density Function
  Integral is like summing many discrete bars



Sampling from a Gaussian





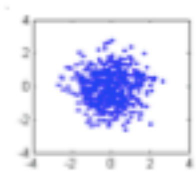
- Integrate 1d Gaussian to get CDF:

$$p(x) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}x^2\right)$$

$$F(x) = \int_{-\infty}^{x} p(t)\,dt = \frac{1}{2} erf\left(\frac{1}{\sqrt{2}}x\right) + \frac{1}{2}$$

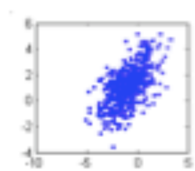
- If sample from uniform, get: $u \sim uniform(0,1)$
- Compute mapping: $x = F^{-1}(u) = \sqrt{2} erfinv(2u-1)$
- This is a Gaussian sample: $x \sim N(x \mid 0,1)$
- For D-dimensional Gaussian N(z|0,I) concatenate samples:

$$\vec{x} = \left[\vec{x}(1) \ldots \vec{x}(D)\right]^T \sim p(\vec{x} \mid 0, I) = \prod_{d=1}^{D} \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\vec{x}(d)^2\right)$$



- For N(z|μ,Σ), add mean & multiply by root cov

$$\vec{z} = \Sigma^{1/2}\vec{x} + \vec{\mu} \sim p(\vec{z} \mid \vec{\mu}, \Sigma)$$



- Example code: gendata.m