Machine Learning (CSE 446): Unsupervised Learning: K-means and Principal Component Analysis

Sham M Kakade

University of Washington
cse446-staff@cs.washington.edu

Announcements

- Please do Q1 (list your collaborators)
- Gradescope: Please correctly tag your pages.
- HW2: posted this friday!
- Office Hours change for Weds: time change for Tommy Merth
  see website

Unsupervised Learning objectives

- The Our dataset consists only of inputs: $\{x_1, \ldots x_N\}$.
  Suppose we do not have labels.
- Two natural objectives:
  - cluster into K groups.
  - project your data into less dimensions

Clustering: What would we like to do?

- **Objective function:** find k-means, $\mu_1, \ldots \mu_k$, which minimizes the following squared distance cost function:

$$\sum_{i=1}^{N} \left(\min_{k' \in \{1,\ldots,k-1\}} \|\mathbf{x}_i - \boldsymbol{\mu}_{k'}\|^2 \right)$$

- We can also write this objective function in terms of the assignments z_i's. How?

This is the general approach of loss function minimization: find parameters which make our objection function "small" (and which also "generalizes")

k-means Convergence Proof Sketch

- The cluster assignments, the z_i's take only finitely many values. So the cluster centers, the $\boldsymbol{\mu}_k$'s, also must only take a finite number of values. Each time we update any of them, we will never increase this function:

$$L(z_1, \ldots, z_N, \boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K) = \sum_{i=1}^{N} \left\| \mathbf{x}_i - \boldsymbol{\mu}_{z_i} \right\|^2 \geq 0$$

  L is the **objective function** of K-Means clustering.
- Convergence must occur in a **finite number** of steps, due to:
  L decreases at every step; L can only take on finitely many values.
  See CIML, Chapter 15 for more details.
- Does the solution depend on the random initialization of the means $\boldsymbol{\mu}_*$? Yes.
- Does K-means converge to the minimal cost solution? No! The objective is an NP-Hard problem, so we can't expect **any** algorithm to minimize the cost without essentially checking (near to) all assignments.

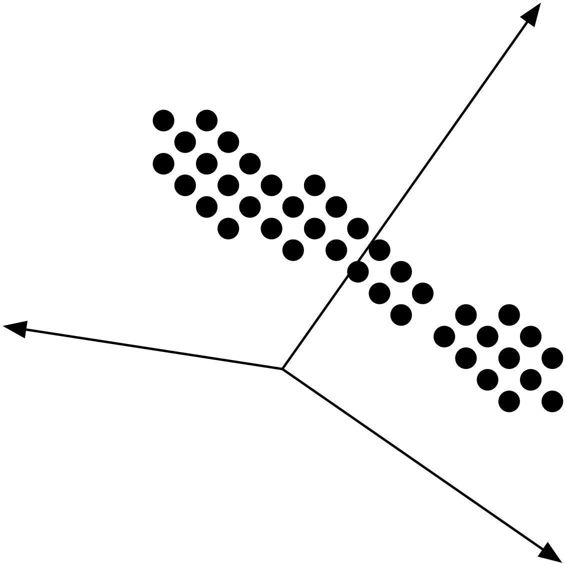
Linear Dimensionality Reduction

As before, you only have a training dataset consisting of $\langle \mathbf{x}_i \rangle_{i=1}^{N}$.

Is there a way to represent each $\mathbf{x}_i \in \mathbb{R}^d$ as a lower-dimensional vector?

(Why would we want to do this?)

Dimension of Greatest Variance



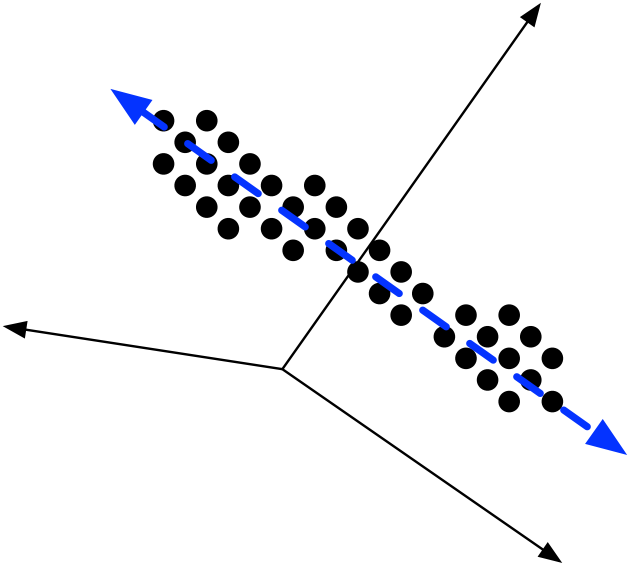
Assume that the data are *centered*, i.e., that mean $\left(\langle \mathbf{x}_i \rangle_{i=1}^{N}\right) = \mathbf{0}$.

Transformation:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \mu$$

where μ is the mean.

Dimension of Greatest Variance



Assume that the data are *centered*, i.e., that mean $\left(\langle \mathbf{x}_i \rangle_{i=1}^{N}\right) = \mathbf{0}$.

Transformation:

$$\mathbf{x}_i \leftarrow \mathbf{x}_i - \mu$$

where μ is the mean.

Projection into One Dimension

Let $\mathbf{u}$ be the dimension of greatest variance, and (without loss of generality) let $\|\mathbf{u}\|_2^2 = 1$.

$p_i = \mathbf{x}_i \cdot \mathbf{u}$ is the projection of the nth example onto $\mathbf{u}$.

Since the mean of the data is $\mathbf{0}$, the mean of $\langle p_1, \ldots, p_N \rangle$ is also 0.

This implies that the variance of $\langle p_1, \ldots, p_N \rangle$ is $\frac{1}{N} \sum_{i=1}^{N} p_i^2$.

The $\mathbf{u}$ that gives the greatest variance, then, is:

$$\underset{\mathbf{u}}{\operatorname{argmax}} \frac{1}{N} \sum_{i=1}^{N} (\mathbf{x}_i \cdot \mathbf{u})^2$$
$$\text{s.t. } \|\mathbf{u}\|_2^2 = 1$$

(This is PCA in one dimension!)

The optimization problem, in terms of matrices

$N \times d$ "data matrix" $\mathbf{X} = \begin{bmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_N^\top \end{bmatrix}$

- With X,

$$\underset{\mathbf{u}}{\operatorname{argmax}} \ \|\mathbf{X}\mathbf{u}\|_2^2$$
$$\text{s.t. } \|\mathbf{u}\|_2^2 = 1$$

- The covariance matrix (assuming mean is subtracted):

$$\Sigma = \frac{1}{N} X^\top X = \frac{1}{N} \sum_{i=1}^{N} x_i x_i^\top$$

and, equivalently,

$$\underset{\mathbf{u}}{\operatorname{argmax}} \ \mathbf{u}^\top \Sigma \mathbf{u}$$
$$\text{s.t. } \|\mathbf{u}\|_2^2 = 1$$

Deriving the Solution

(You are not responsible for the derivation).

$$\underset{\mathbf{u}}{\operatorname{argmax}}\ \mathbf{u}^\top \Sigma \mathbf{u}, \ \text{s.t. } \|\mathbf{u}\|_2^2 = 1$$

- The Lagrangian encoding of the problem moves the constraint into the objective:

$$\max_{\mathbf{u}} \min_{\lambda} \mathbf{u}^\top \Sigma \mathbf{u} - \lambda(\|\mathbf{u}\|_2^2 - 1) \quad \Rightarrow \quad \min_{\lambda} \max_{\mathbf{u}} \mathbf{u}^\top \Sigma \mathbf{u} - \lambda(\|\mathbf{u}\|_2^2 - 1)$$

- Gradient (first derivatives with respect to $\mathbf{u}$): $2\Sigma\mathbf{u} - 2\lambda\mathbf{u}$
- Setting equal to $\mathbf{0}$ leads to: $\lambda\mathbf{u} = \Sigma\mathbf{u}$
- You may recognize this as the definition of an eigenvector ($\mathbf{u}$) and eigenvalue (λ) for the matrix Σ.
- We take the first (largest) eigenvalue.

Projecting into Multiple Dimensions

So far, we've projected each $\mathbf{x}_i$ into one dimension.

To get a second projection $\mathbf{v}$, we solve the same problem again, but this time with another constraint:

$$\underset{\mathbf{v}}{\operatorname{argmax}}\ \mathbf{v}^\top \Sigma \mathbf{v}, \ \text{s.t. } \|\mathbf{v}\|_2^2 = 1 \text{ and } \boxed{\mathbf{u} \cdot \mathbf{v} = 0}$$

(That is, we want a dimension that's orthogonal to the $\mathbf{u}$ that we found earlier.)

Following the same steps we had for $\mathbf{u}$, we can show that the solution will be the *second* eigenvector.

Principal Components Analysis

Data: unlabeled data with mean $\mathbf{0}$, $\mathbf{X} = [\mathbf{x}_1 | \mathbf{x}_2 | \cdots | \mathbf{x}_N]^\top$, and dimensionality $K < d$

Result: K-dimensional projection of $\mathbf{X}$

let $\langle \lambda_1, \ldots, \lambda_K \rangle$ be the top K eigenvalues of $\Sigma = \frac{1}{N}\mathbf{X}^\top\mathbf{X}$
and $\langle \mathbf{u}_1, \ldots, \mathbf{u}_K \rangle$ be the corresponding eigenvectors;
let $\mathbf{U} = [\mathbf{u}_1 | \mathbf{u}_2 | \cdots | \mathbf{u}_K]$;
return $\mathbf{XU}$;

Algorithm 1: PCA

On your own time, you can read up about many algorithms for finding eigenstuff of a matrix.