Unsupervised Learning

Sidharth Baskaran

June 2021

Clustering

- Given data without labels
- Training set of the form $\{x^{(1)}, \dots, x^{(m)}\}$
- Clustering algorithm $\rightarrow$ finds clusters in data
- Examples
  - Market segmentation, social network analysis, organize computing clusters, astronomical data analysis

K-means clustering algorithm

- Randomly initialize 2 points $\rightarrow$ cluster centroids for 2 clusters
- Assigns data points to clusters based on proximity to a centroid
- Then move centroids to average of location of their cluster points
- Reassign centroids again and chante cluster assignments
- After certain number of iterations, k-means converges

Input

- $K \rightarrow$ number of clusters
- Training set $\{x^{(1)}, \dots, x^{(m)}\}$
- Use convention $x^{(i)} \in \mathbb{R}^n$ and drop $x_0 = 1$

```
Randomly initialize $K$ cluster centroids $\mu_1, \dots, \mu_K \in \mathbb{R}^n$
Repeat {
for $i = 1 \rightarrow m$
$c^{(i)}$ := idx of cluster centroid closest to $x^{(i)}$
for $k = 1 \rightarrow K$
$\mu_k$ := mean of points assigned to cluster $k$
}
```

- Can also apply to seemingly single-cluster set of data $\rightarrow$ k-means still finds clusters
  - Similar to market segmentation

Optimization Objective

- Notation
  - $c^{(i)}$ = index of cluster to which $x^{(i)}$ is assigned
  - μ_k = cluster centroid k where $\mu_k \in \mathbb{R}^n$
  - $\mu_{c^{(i)}}$ = cluster centroid of cluster to which example $x^{(i)}$ has been assigned

$$J(c^{(1)}, \dots, c^{(m)}, \mu_1, \dots, \mu_K) = \frac{1}{m}\sum_{i=1}^{m} ||x^{(i)} - \mu_{c^{(i)}}||^2$$

- Distortion cost function
  - Want to find $\mu_{c^{(i)}}$ and $c^{(i)}$ to minimize J
  - Must converge, cannot increase over number of iterations

Random Initialization

- Should have $K < k$
- Randomly pick K training examples
- Set $\mu_1, \dots, \mu_K$ equal to these K examples
- K-means converging to local optima $\rightarrow$ leads to bad clustering
  - Multiple random initializations help prevent local convergence
- Pick clustering that gives lowest cost J
- if K is small, then one random initialization is likely to give good clustering

Choosing number of clusters K

- Elbow method
  - Plot J vs K where K is independent var
  - Vertex point of curve gives choice of K to use
  - Is ambiguous
- Evaluate K-means based on metric for performing in a later purpose
  - Choose K based on how many divisions in the metric desired

Dimensionality Reduction

- Data compression through dimensional reduction $\mathbb{R}^n \rightarrow \mathbb{R}^c$ for $n > c$
  - Reducing data from 2D to 1D
  - $x^{(i)} \in \mathbb{R}^n \rightarrow z^{(i)} \in \mathbb{R}$ for $i \in \text{range}(1, \dots, m)$
  - Allows for faster running algorithms
- New features do not have defined meaning, need to be assigned

Principal Component Analysis

- Problem formulation
  - Finds a lower dimensional space to project data which minimizes distances to surface (projection error)
  - Find k vectors $u^{(1)}, \dots, u^{(k)} \in \mathbb{R}^n$ on which to project $\mathbb{R}^n \rightarrow \mathbb{R}^k$ where this is the subspace $\text{span}(u^{(1)}, \dots, u^{(m)}) \in \mathbb{R}^k$
- Is not linear regression
  - Distances are vertical in regression, not orthogonal vector magnitudes

Algorithm

- Data preprocessing
  - Perform feature scaling/mean normalization
  - Compute μ_j, mean of of data set
  - Replace $x_j^{(i)}$ with $x_j - \mu_j$
  - Scale features to have comparable values (e.g. divide by s_j)

- Reduced dimension vectors are $z^{(i)} \in \mathbb{R}^k$
- $u^{(i)} \in \mathbb{R}^n$ define the reduced dimension space
- Algorithm
  - Reduce data from dimension n to k
  - Compute covariance matrix
    - $\Sigma = \frac{1}{m} \sum_{i=1}^{n} (x^{(i)})(x^{(i)})^T in \mathbb{R}^{n \times n}$
  - Compute eigenvectors of matrix σ
    - `[U,S,V] = svd(Sigma)`
    - Singular value decomposition (SVD) or `eig(Sigma)` computes the eigenvectors
    - Output matrices are U, S, V

$$U = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(n)} \\ | & | & & | \end{bmatrix} \in \mathbb{R}^{n \times n}$$

  - Use first k columns to get usable u vectors

$$U_{\text{red}} \in \mathbb{R}^{n \times k}$$

$$z = \begin{bmatrix} | & | & & | \\ u^{(1)} & u^{(2)} & \dots & u^{(k)} \\ | & | & & | \end{bmatrix}^T x = \begin{bmatrix} -\left(u^{(1)}\right)^T - \\ \vdots \\ -\left(u^{(k)}\right)^T - \end{bmatrix} x \in \mathbb{R}^k$$

Summary

Perform mean normalization and feature scaling

```
Sigma = 1\m * x'*x;
[U,S,V] = svd(Sigma);
Ureduce = U(:,1:k);
Z = Ureduce'*x;
```

Reconstruction from compressed representation

- Reconstructing original representation
  - $U_{\text{red}} z = U_{\text{red}} U_{\text{red}}^T x \implies x_{\text{app}} = U_{\text{red}} z \approx x$

Choosing number of principal components

- Choosing k involves minimizing distortion (avg. sqd. projection) error $\frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - x_{\text{app}}^{(i)}||^2$
- Total variation in the data is $\frac{1}{m} \sum_{i=1}^{m} ||x^{(i)}||^2$
- Typically choose k under following constraint
  - 99% of variance is retained

$$\boxed{\frac{\frac{1}{m} \sum_{i=1}^{m} ||x^{(i)} - x_{\text{app}}^{(i)}||^2}{\frac{1}{m} \sum_{i=1}^{m} ||x^{(i)}||^2} \leq 0.01}$$

- Matrix S from `[U,S,V] = svd(Sigma)` is diagonal
  - Can be shown that quantity above is equivalent to $1 - \frac{\sum_{i=1}^{k} S_{ii}}{\sum_{i=1}^{n} S_{ii}}$ where $k \leq n$
  - More computationally efficient as requires one `svd` computation only

Advice for Applying PCA

- PCA can be used to speed up learning algorithm
- Example $\rightarrow$ supervised learning
  - Take labeled data set $(x^{(1)}, y^{(1)}), \ldots, (x^{(m)}, y^{(m)})$
  - $x^{(1)}, \ldots, x^{(m)} \in \mathbb{R}^{10000} \rightarrow z^{(1)}, \ldots, z^{(m)} \in \mathbb{R}^{1000}$
  - New training set is $(z^{(1)}, y^{(1)}), \ldots, (z^{(m)}, y^{(m)})$
    - Can then feed this to algorithm
  - Run PCA **only** on training set and use **this** mapping on CV and test sets
- PCA bad use $\rightarrow$ to prevent overfitting
  - Reduce number of features to $k <$
  - Less features, less likely to overfit
  - use regularization instead