Machine Learning for Signal Processing
Quantization and Clustering

Bhiksha Raj
Learning Representations: Problem so far

- **Problem:** Given a collection of data $X$, find a set of “bases” $B$, such that each vector $x_i$ can be expressed as a weighted combination of the bases

$$x_i = Bw_i$$

$$x_i = w_{11}B_1 + \ldots + w_{1K}B_K$$
Why is this important?

- With the right set of bases, the weights represent the data most effectively
  - We can now use the weights to represent the data
  - E.g. with notes as bases, the weights would be the score

- If the bases are agreed upon, we can also communicate the information about the data most efficiently
  - Just communicate the weights
  - E.g. enough to store eigen face weights to reconstruct face
  - E.g. just reading the score is sufficient for anyone to recreate music
What is the most accurate way to represent data

\[ f = \sum_i w_i d_i \]

\( w_k = 1, \quad w_j = 0 \text{ for } j \neq k \)

Selecting the kth face in the collection

- If, instead of bases, we had a dictionary of all possible data
  - A matrix that included every possible data vector as a column
  - And the weights vector simply selected the correct data instance
  - I.e. \( \mathbf{w} \) was one-sparse vector
    \[ |\mathbf{w}|_0 = 1 \]
    (actually a one-hot vector because the one non-zero entry of \( \mathbf{w} = 1 \), i.e. \( \sum_i w_i = 1 \))
What is the most accurate way to represent data

If, instead of bases, we had a *dictionary* of all possible data

- A matrix that included every possible data vector as a column
- And the weights vector simply selected the correct data instance

**Problem:** Infeasible to construct such a dictionary!

- Will require infinite entries
  - And our $\mathbf{w}$ vector too will require infinite bits to represent
- Alternately, will require storing the entire training data
  - And will not be useful to represent data outside the training set

Selecting the $k$th face in the collection

$$f = \sum_{i} w_i d_i$$

$w_k = 1$, $w_j = 0$ for $j \neq k$
Approximate representation with a dictionary

\[ f \approx \sum_i w_i d_i \]

Selecting the kth face in the collection

- **Problem:** Infeasible to construct a perfect dictionary
  - Will require too many (potentially infinite) entries

- **Solution:** Can we instead construct a smaller *finite* dictionary such that all data can be approximated well by one of the entries in the dictionary?
  - E.g. “The guy looks a lot like the 7th face in the dictionary”
  - E.g. The vector \( \mathbf{x} \) looks a lot like the \( d_i \), the i-th entry in the dictionary.

- **Questions:**
  - What do we mean by “looks a lot like”
  - How do we construct the dictionary?
Approximate representation with a dictionary

\[ f \approx \sum_{i} w_i d_i \]

\( w_k = 1, \ w_j = 0 \) for \( j \neq k \)

Selecting the kth face in the collection

- **Problem:** Infeasible to construct a perfect dictionary
  - Will require too many (potentially infinite) entries

- **Solution:** Can we instead construct a smaller *finite* dictionary such that all data can be approximated well by one of the entries in the dictionary?
  - E.g. “The guy looks a lot like the 7th face in the dictionary”
  - E.g. The vector \( \mathbf{x} \) looks a lot like the \( d_i \), the i-th entry in the dictionary.

- **Questions:**
  - What do we mean by “looks a lot like”
  - How do we construct the dictionary?
Quantifying the error

- Different error metrics will result in different solutions
- Let's generically represent the error as $\text{div}()$

$$\hat{f} = Dw, \quad |w|_0 = 1, \sum_i w_i = 1$$

$$\text{Error}(f) = \text{div}(f, \hat{f})$$

- A common choice is the L2 error

$$\text{Error}(f) = |f - \hat{f}|^2$$
Approximate representation with a dictionary

\[ f \approx \sum_{i} w_i d_i \]

Selecting the kth face in the collection

- **Problem:** Infeasible to construct a perfect dictionary
  - Will require too many (potentially infinite) entries

- **Solution:** Can we instead construct a smaller *finite* dictionary such that all data can be approximated well by one of the entries in the dictionary?
  - E.g. “The guy looks a lot like the 7th face in the dictionary”
  - E.g. The vector \( \mathbf{x} \) looks a lot like the \( d_i \), the i-th entry in the dictionary.

- **Questions:**
  - What do we mean by “looks a lot like”
  - How do we construct the dictionary?
Learning the Dictionary

- $\mathbf{V} = [V_1, V_2, V_3, \ldots]$ are the data for which the dictionary is being learned.
- $\mathbf{D} = [d_1, d_2, \ldots, d_K]$ is the matrix of dictionary vectors.
- $\mathbf{W} = [w_1, w_2, w_3, \ldots]$ is a set of one-hot vectors.
- Learning: Learn $\mathbf{D}$ and $\mathbf{W}$ to minimize total error on $\mathbf{V}$.

\[
\hat{\mathbf{D}}, \hat{\mathbf{W}} = \underset{\mathbf{D}, \mathbf{W}}{\text{argmin}} \ \text{div}(\mathbf{V}, \mathbf{D}\mathbf{W}) = \underset{\mathbf{D}, \mathbf{W}}{\text{argmin}} \sum_i \text{div}(V_i, Dw_i),
\]

s.t. $w_i = \text{one hot}$

- If we’re only interested in learning the dictionary

\[
\hat{\mathbf{D}} = \underset{\mathbf{D}}{\text{argmin}} \ \min_{\mathbf{W}} \sum_i \text{div}(V_i, Dw_i), \quad \text{s.t. } w_i = \text{one hot}
\]
Learning the Dictionary

• \( \hat{D} = \arg \min_D \min_W \sum_i \text{div}(V_i, Dw_i) \)

\[
= \arg \min_D \sum_i \min_{w_i} \text{div}(V_i, Dw_i)
\]

• Generally does not have a closed form solution, but can solved with the following iteration that provably reduces error in each step

\[
w_i = \arg \min_W \text{div}(V_i, Dw)
\]

\[
\hat{D} = \arg \min_D \sum_i \text{div}(V_i, Dw_i)
\]
Learning the Dictionary

- \( \hat{D} = \text{argmin} \min \sum_i \text{div}(V_i, D_{w_i}) \)

For \( \text{div}(.) = \|V_i - D_{w_i}\|^2 \) this gives us the well-known K-means algorithm

\[
= \text{argmin}_D \sum_i \min_{w_i} \text{div}(V_i, D_{w_i})
\]

- Generally does not have a closed form solution, but can solved with the following iteration that provably reduces error in each step

\[
w_i = \text{argmin}_w \text{div}(V_i, Dw)
\]

\[
\hat{D} = \text{argmin}_D \sum_i \text{div}(V_i, Dw_i)
\]
Learning the Dictionary

• \( \hat{D} = \arg\min_D \sum_i \text{div}(V_i, Dw_i) \)

For \( \text{div}(\cdot) = \|V_i - Dw_i\|^2 \) this gives us the well-known K-means algorithm

\[
\hat{D} = \arg\min_D \sum_i \text{div}(V_i, Dw_i)
\]

• Grouping \( V_i \) by the dictionary entries they are assigned to \( (w_i) \) results in clustering error in each step

\[
w_i = \arg\min_w \text{div}(V_i, Dw)
\]

\[
\hat{D} = \arg\min_D \sum_i \text{div}(V_i, Dw_i)
\]
So lets look at clustering

• From a more naïve, procedural perspective..
Clustering
Statistical Modelling and Latent Structure

• Much of statistical modelling attempts to identify *latent* structure in the data
  – Structure that is not immediately apparent from the observed data
  – But which, if known, helps us explain it better, and make predictions from or about it

• Clustering methods attempt to extract such structure from *proximity*
  – *First-level* structure (as opposed to deep structure)

• We will see still other forms of latent structure discovery later in the course
How
Clustering

• What is clustering
  – Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)
Clustering

• What is clustering
  – Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)
Clustering

- **What is clustering**
  - Clustering is the determination of naturally occurring grouping of data/instances (*with low within-group variability and high between-group variability)*

- **How is it done**
  - Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind
Clustering

• What is clustering
  – Clustering is the determination of naturally occurring grouping of data/instances (with low within-group variability and high between-group variability)

• How is it done
  – Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind
  – The objective function used affects the nature of the discovered clusters
    • E.g. Euclidean distance vs.
Clustering

- **What is clustering**
  - Clustering is the determination of naturally occurring grouping of data/instances *(with low within-group variability and high between-group variability)*

- **How is it done**
  - Find groupings of data such that the groups optimize a “within-group-variability” objective function of some kind
  - The objective function used affects the nature of the discovered clusters
    - E.g. Euclidean distance vs.
    - Distance from center
Why Clustering

• Automatic grouping into “Classes”
  – Different clusters may show different behavior

• **Representation**: Quantization
  – All data within a cluster are represented by a single point

• Preprocessing step for other algorithms
  – Indexing, categorization, etc.
Finding natural structure in data

- Find natural groupings in data for further analysis
- Discover *latent* structure in data
Some Applications of Clustering

- Image segmentation
• **Quantize every vector** to one of $K$ (vector) values

• What are the optimal $K$ vectors? How do we find them? How do we perform the quantization?

• **LBG algorithm**
Quantization: Formally

\[ V = \sum_i w_i d_i \]

\[ V = Dw \quad |w| = 1 \]
\[ |w|_0 = 1 \]

- \( d_i \) are the “representative” vectors of each cluster
- Restriction: only one of the \( w_i \) is 1, the rest are 0
  - \( \sum_i w_i = 0 \)
  - \( w \) is unit length and one-sparse
Representation: BOW

• How to retrieve all music videos by this guy?
• Build a classifier
  – But how do you represent the video?
Representation: BOW

\[ V_k = Dw_k \]
\[ f = \sum_k w_k \]

Training: Each point is a video frame

• Bag of words representations of video/audio/data
Obtaining “Meaningful” Clusters

• Two key aspects:
  – 1. The feature representation used to characterize your data
  – 2. The “clustering criteria” employed
Clustering Criterion

• The “Clustering criterion” actually has two aspects

• Cluster compactness criterion
  – Measure that shows how “good” clusters are
    • The objective function

• Distance of a point from a cluster
  – To determine the cluster a data vector belongs to
“Compactness” criteria for clustering

• Distance based measures
  – Total distance between each element in the cluster and every other element in the cluster
“Compactness” criteria for clustering

• Distance based measures

  – Total distance between each element in the cluster and every other element in the cluster
“Compactness” criteria for clustering

• Distance based measures
  – Total distance between each element in the cluster and every other element in the cluster
  – Distance between the two farthest points in the cluster
“Compactness” criteria for clustering

• Distance based measures
  – Total distance between each element in the cluster and every other element in the cluster
  – Distance between the two farthest points in the cluster
  – Total distance of every element in the cluster from the centroid of the cluster
“Compactness” criteria for clustering

• Distance based measures
  – Total distance between each element in the cluster and every other element in the cluster
  – Distance between the two farthest points in the cluster
  – Total distance of every element in the cluster from the centroid of the cluster
“Compactness” criteria for clustering

• Distance based measures
  – Total distance between each element in the cluster and every other element in the cluster
  – Distance between the two farthest points in the cluster
  – Total distance of every element in the cluster from the centroid of the cluster
“Compactness” criteria for clustering

- Distance based measures
  - Total distance between each element in the cluster and every other element in the cluster
  - Distance between the two farthest points in the cluster
  - Total distance of every element in the cluster from the centroid of the cluster
  - Distance measures are often weighted Minkowski metrics

\[
\text{dist} = \left( w_1 |a_1 - b_1|^n + w_2 |a_2 - b_2|^n + \ldots + w_M |a_M - b_M|^n \right)^{\frac{1}{n}}
\]
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
Clustering: Distance from cluster

• How far is a data point from a cluster?
  – Euclidean or Minkowski distance from the centroid of the cluster
  – Distance from the closest point in the cluster
Clustering: Distance from cluster

• How far is a data point from a cluster?
  – Euclidean or Minkowski distance from the centroid of the cluster
  – Distance from the closest point in the cluster
  – Distance from the farthest point in the cluster
Clustering: Distance from cluster

- How far is a data point from a cluster?
  - Euclidean or Minkowski distance from the centroid of the cluster
  - Distance from the closest point in the cluster
  - Distance from the farthest point in the cluster
  - Probability of data measured on cluster distribution
Clustering: Distance from cluster

• How far is a data point from a cluster?
  – Euclidean or Minkowski distance from the centroid of the cluster
  – Distance from the closest point in the cluster
  – Distance from the farthest point in the cluster
  – Probability of data measured on cluster distribution
  – Fit of data to cluster-based regression
Optimal clustering: Exhaustive enumeration

• All possible combinations of data must be evaluated
  – If there are M data points, and we desire N clusters, the number of ways of separating M instances into N clusters is

\[
\frac{1}{M!} \sum_{i=0}^{N} (-1)^i \binom{N}{i} (N-i)^M
\]

  – Exhaustive enumeration based clustering requires that the objective function (the “Goodness measure”) be evaluated for every one of these, and the best one chosen

• This is the only correct way of optimal clustering
  – Unfortunately, it is also computationally unrealistic
Not-quite non sequitur: Quantization

- Linear quantization (uniform quantization):
  - Each digital value represents an equally wide range of analog values
  - Regardless of distribution of data
  - Digital-to-analog conversion represented by a “uniform” table

<table>
<thead>
<tr>
<th>Signal Value</th>
<th>Bits</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \geq 3.75v$</td>
<td>11</td>
<td>$3 \times \text{const}$</td>
</tr>
<tr>
<td>$3.75v &gt; S \geq 2.5v$</td>
<td>10</td>
<td>$2 \times \text{const}$</td>
</tr>
<tr>
<td>$2.5v &gt; S \geq 1.25v$</td>
<td>01</td>
<td>$1 \times \text{const}$</td>
</tr>
<tr>
<td>$1.25v &gt; S \geq 0v$</td>
<td>00</td>
<td>0</td>
</tr>
</tbody>
</table>
Not-Quite Non Sequitur: Quantization

- Non-Linear quantization:
  - Each digital value represents a different range of analog values
    - Finer resolution in high-density areas
    - Mu-law / A-law assumes a Gaussian-like distribution of data
  - Digital-to-analog conversion represented by a “non-uniform” table

<table>
<thead>
<tr>
<th>Signal Value</th>
<th>Bits</th>
<th>Mapped to</th>
</tr>
</thead>
<tbody>
<tr>
<td>$S \geq 4v$</td>
<td>11</td>
<td>4.5</td>
</tr>
<tr>
<td>$4v &gt; S \geq 2.5v$</td>
<td>10</td>
<td>3.25</td>
</tr>
<tr>
<td>$2.5v &gt; S \geq 1v$</td>
<td>01</td>
<td>1.25</td>
</tr>
<tr>
<td>$1.0v &gt; S \geq 0v$</td>
<td>00</td>
<td>0.5</td>
</tr>
</tbody>
</table>

Analog value (arrows are quantization levels)
Non-uniform quantization

• If data distribution is not Gaussian-ished?
  – Mu-law / A-law are not optimal
  – How to compute the optimal ranges for quantization?
    • Or the optimal table
The Lloyd Quantizer

- Lloyd quantizer: An iterative algorithm for computing optimal quantization tables for non-uniformly distributed data
- Learned from “training” data
Lloyd Quantizer

- Randomly initialize quantization points
  - Right column entries of quantization table
Lloyd Quantizer

• Randomly initialize quantization points
  – Right column entries of quantization table

• Assign all training points to the nearest quantization point
  – Draw boundaries
Lloyd Quantizer

- Randomly initialize quantization points
  - Right column entries of quantization table

- Assign all training points to the nearest quantization point
  - Draw boundaries

- Reestimate quantization points
Lloyd Quantizer

- Randomly initialize quantization points
  - Right column entries of quantization table
- Assign all training points to the nearest quantization point
  - Draw boundaries
- Reestimate quantization points
- Iterate until convergence
Generalized Lloyd Algorithm: K–means clustering

• K means is an iterative algorithm for clustering \textit{vector} data

• General procedure:
  – Initially group data into the required number of clusters somehow (initialization)
  – Assign each data point to the closest cluster
  – Once all data points are assigned to clusters, redefine clusters
  – Iterate
K–means

- Problem: Given a set of data vectors, find natural clusters

- Clustering criterion is **scatter**: distance from the centroid
  - Every cluster has a centroid
  - The centroid represents the cluster

- **Definition:** The **centroid** is the weighted mean of the cluster
  - Weight = 1 for basic scheme

\[ m_{\text{cluster}} = \frac{1}{\sum W_i} \sum_{i \in \text{cluster}} W_i x_i \]
K–means

1. Initialize a set of centroids randomly

\[ d_{\text{cluster}} = \sum_{i=1}^{N} \left| x - \text{cluster}_i \right| \]
K–means

1. Initialize a set of centroids randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   - \( d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \)
**K–means**

1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
K-means

1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   • $d_{cluster} = \text{distance}(x, m_{cluster})$

3. Put data point in the cluster of the closest centroid
   • Cluster for which $d_{cluster}$ is minimum
1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   \[ d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \]

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
**K-means**

1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   
   \[ d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \]

3. Put data point in the cluster of the closest centroid
   
   Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of centroids randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of centroids randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   
   \[ d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \]

3. Put data point in the cluster of the closest centroid
   
   - Cluster for which \( d_{\text{cluster}} \) is minimum

4. When all data points are clustered, recompute centroids

\[ m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i x_i \]
K–means

1. Initialize a set of centroids randomly
2. For each data point \( x \), find the distance from the centroid for each cluster
   - \( d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \)
3. Put data point in the cluster of the closest centroid
   - Cluster for which \( d_{\text{cluster}} \) is minimum
4. When all data points are clustered, recompute centroids
   \[
   m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i x_i
   \]
5. If not converged, go back to 2
**K-Means comments**

- The distance metric determines the clusters
  - In the original formulation, the distance is $L_2$ distance
    - Euclidean norm, $w_i = 1$

  \[
  \text{distance}_{\text{cluster}}(x, m_{\text{cluster}}) = \| x - m_{\text{cluster}} \|_2
  \]

  - If we replace every $x$ by $m_{\text{cluster}}(x)$, we get Vector Quantization

- K-means is an instance of **generalized** EM

- Not guaranteed to converge for all distance metrics
Initialization

- Random initialization
- Top-down clustering
  - Initially partition the data into two (or a small number of) clusters using K means
  - Partition each of the resulting clusters into two (or a small number of) clusters, also using K means
  - Terminate when the desired number of clusters is obtained
K-Means for Top–Down clustering

1. Start with one cluster
K-Means for Top–Down clustering

1. Start with one cluster
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

3. Initialize K means with new set of centroids
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

3. Initialize K means with new set of centroids

4. Iterate Kmeans until convergence
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

3. Initialize K means with new set of centroids

4. Iterate Kmeans until convergence
K-Means for Top–Down clustering

1. Start with one cluster

2. Split each cluster into two:
   - Perturb centroid of cluster slightly (by < 5%) to generate two centroids

3. Initialize K means with new set of centroids

4. Iterate Kmeans until convergence

5. If the desired number of clusters is not obtained, return to 2
Non-Euclidean clusters

- Basic K-means results in good clusters in Euclidean spaces
  - Alternately stated, will only find clusters that are “good” in terms of Euclidean distances
- Will not find other types of clusters
For other forms of clusters we must modify the distance measure
- E.g. distance from a circle

May be viewed as a distance in a higher dimensional space
- I.e. *Kernel* distances
- *Kernel* K-means

Other related clustering mechanisms:
- Spectral clustering
  - Non-linear weighting of adjacency
- Normalized cuts
The Kernel Trick

- Transform the data into a synthetic higher-dimensional space where the desired patterns become natural clusters based on *Euclidean* distance
  - E.g. the quadratic transform above
- Problem: What is the function/space?
- Problem: Distances in higher dimensional-space are more expensive to compute
  - Yet only carry the same information in the lower-dimensional space

\[ f([x,y]) \rightarrow [x,y,z] \]
\[ x = x \]
\[ y = y \]
\[ z = \alpha(x^2 + y^2) \]
Distance in higher-dimensional space

• Transform data \( x \) through a possibly unknown function \( \Phi(x) \) into a higher (potentially infinite) dimensional space
  
  \[ z = \Phi(x) \]

• The distance between two points is computed in the higher-dimensional space
  
  \[ d(x_1, x_2) = \| z_1 - z_2 \|^2 = \| \Phi(x_1) - \Phi(x_2) \|^2 \]

• \( d(x_1, x_2) \) can be computed without computing \( z \)
  
  – Since it is a direct function of \( x_1 \) and \( x_2 \)
Distance in higher-dimensional space

• Distance in lower-dimensional space: A combination of dot products
  \[ ||z_1 - z_2||^2 = (z_1 - z_2)^T(z_1 - z_2) = z_1.z_1 + z_2.z_2 - 2 z_1.z_2 \]

• Distance in higher-dimensional space
  \[ d(x_1, x_2) = ||\Phi(x_1) - \Phi(x_2)||^2 \]
  \[ = \Phi(x_1).\Phi(x_1) + \Phi(x_2).\Phi(x_2) - 2 \Phi(x_1).\Phi(x_2) \]

• \( d(x_1, x_2) \) can be computed without knowing \( \Phi(x) \) if:
  \[ \Phi(x_1).\Phi(x_2) \] can be computed for any \( x_1 \) and \( x_2 \) without knowing \( \Phi(.) \)
The Kernel function

- A kernel function $K(x_1, x_2)$ is a function such that:
  - $K(x_1, x_2) = \Phi(x_1) \cdot \Phi(x_2)$

- Once such a kernel function is found, the distance in higher-dimensional space can be found in terms of the kernels
  - $d(x_1, x_2) = \|\Phi(x_1) - \Phi(x_2)\|^2$
    - $= \Phi(x_1) \cdot \Phi(x_1) + \Phi(x_2) \cdot \Phi(x_2) - 2 \Phi(x_1) \cdot \Phi(x_2)$
    - $= K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)$

- But what is $K(x_1, x_2)$?
A property of the dot product

• For any vector $\mathbf{v}$, $\mathbf{v}^T \mathbf{v} = ||\mathbf{v}||^2 \geq 0$
  – This is just the length of $\mathbf{v}$ and is therefore non-negative

• For any vector $\mathbf{u} = \sum_i a_i \mathbf{v}_i$, $||\mathbf{u}||^2 \geq 0$
  => $(\sum_i a_i \mathbf{v}_i)^T (\sum_i a_i \mathbf{v}_i) \geq 0$
  => $\sum_i \sum_j a_i a_j \mathbf{v}_i \cdot \mathbf{v}_j \geq 0$

• This holds for ANY real $\{a_1, a_2, \ldots\}$
The Mercer Condition

• If \( z = \Phi(x) \) is a high-dimensional vector derived from \( x \) then for all real \{a_1, a_2, \ldots\} and any set \{z_1, z_2, \ldots\} = \{\Phi(x_1), \Phi(x_2), \ldots\}
  
  \[
  \sum_i \sum_j a_i a_j z_i \cdot z_j \geq 0
  \]
  
  \[
  \sum_i \sum_j a_i a_j \Phi(x_i) \cdot \Phi(x_j) \geq 0
  \]

• If \( K(x_1, x_2) = \Phi(x_1) \cdot \Phi(x_2) \)
  
  \[
  \Rightarrow \sum_i \sum_j a_i a_j K(x_i, x_j) \geq 0
  \]

• Any function \( K() \) that satisfies the above condition is a valid kernel function
The Mercer Condition

• \( K(x_1, x_2) = \Phi(x_1). \Phi(x_2) \)
  \[ \Rightarrow \sum_i \sum_j a_i a_j K(x_i, x_j) \geq 0 \]

• **A corollary**: If any kernel \( K(\cdot) \) satisfies the Mercer condition

  \[ d(x_1, x_2) = K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2) \]

  satisfies the following requirements for a “distance”

  - \( d(x, x) = 0 \)
  - \( d(x, y) \geq 0 \)
  - \( d(x, w) + d(w, y) \geq d(x, y) \)
Typical Kernel Functions

- Linear: \( K(x,y) = x^T y + c \)
- Polynomial: \( K(x,y) = (ax^T y + c)^n \)
- Gaussian: \( K(x,y) = \exp(-||x-y||^2/\sigma^2) \)
- Exponential: \( K(x,y) = \exp(-||x-y||/\lambda) \)
- Several others
  - Choosing the right Kernel with the right parameters for your problem is an artform
Kernel K-means

\[ K(x,y) = (x^T y + c)^2 \]

- Perform the K-mean in the Kernel space
  - The space of \( z = \Phi(x) \)

- The algorithm.
The mean of a cluster

- The average value of the points in the cluster *computed in the high-dimensional space*

\[ m_{\text{cluster}} = \frac{1}{N_{\text{cluster}}} \sum_{i \in \text{cluster}} \Phi(x_i) \]

- Alternately the weighted average

\[ m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i \Phi(x_i) = C \sum_{i \in \text{cluster}} w_i \Phi(x_i) \]
The mean of a cluster

- The average value of the points in the cluster *computed in the high-dimensional space*

\[
m_{\text{cluster}} = \frac{1}{N_{\text{cluster}}} \sum_{i \in \text{cluster}} \Phi(x_i)
\]

RECALL: We may never actually be able to compute this mean because \(\Phi(x)\) is not known.

- Alternately the weighted average

\[
m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i \Phi(x_i) = C \sum_{i \in \text{cluster}} w_i \Phi(x_i)
\]
K–means

- Initialize the clusters with a random set of K points
  - \( N_{\text{cluster}} \) is no. of points in cluster

- For each data point \( x \), find the closest cluster

\[
\text{cluster}(x) = \min_{\text{cluster}} d(x, \text{cluster}) = \min_{\text{cluster}} \| \Phi(x) - m_{\text{cluster}} \|^2
\]

\[
d(x, \text{cluster}) = \| \Phi(x) - m_{\text{cluster}} \|^2 = \left( \Phi(x) - \frac{1}{N_{\text{cluster}}} \sum_{i \in \text{cluster}} \Phi(x_i) \right)^T \left( \Phi(x) - \frac{1}{N_{\text{cluster}}} \sum_{i \in \text{cluster}} \Phi(x_i) \right)
\]

\[
= \left( \Phi(x)^T \Phi(x) - \frac{2}{N_{\text{cluster}}} \sum_{i \in \text{cluster}} \Phi(x)^T \Phi(x_i) + \frac{1}{N_{\text{cluster}}^2} \sum_{i \in \text{cluster}} \sum_{j \in \text{cluster}} \Phi(x_i)^T \Phi(x_j) \right)
\]

\[
= K(x, x) - \frac{2}{N_{\text{cluster}}} \sum_{i \in \text{cluster}} K(x, x_i) + \frac{1}{N_{\text{cluster}}^2} \sum_{i \in \text{cluster}} \sum_{j \in \text{cluster}} K(x_i, x_j)
\]

Computed entirely using only the kernel function!
K–means

1. Initialize a set of clusters randomly
K–means

1. Initialize a set of clusters randomly

The centroids are virtual: we don’t actually compute them explicitly!

\[ m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i x_i \]
K–means

1. Initialize a set of clusters randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   \[ d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \]

\[
d_{\text{cluster}} = K(x, x) - 2C \sum_{i \in \text{cluster}} w_i K(x, x_i) + C^2 \sum_{i \in \text{cluster}} \sum_{j \in \text{cluster}} w_i w_j K(x_i, x_j)
\]
K–means

1. Initialize a set of clusters randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   - \( d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \)

3. Put data point in the cluster of the closest centroid
   - Cluster for which \( d_{\text{cluster}} \) is minimum
K-means

1. Initialize a set of clusters randomly
2. For each data point \( x \), find the distance from the centroid for each cluster
   - \( d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \)
3. Put data point in the cluster of the closest centroid
   - Cluster for which \( d_{\text{cluster}} \) is minimum
K-means

1. Initialize a set of clusters randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   • $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   • Cluster for which $d_{\text{cluster}}$ is minimum
**K-means**

1. Initialize a set of clusters randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of clusters randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   • $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   • Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of clusters randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
**K-means**

1. Initialize a set of clusters randomly

2. For each data point $x$, find the distance from the centroid for each cluster
   - $d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}})$

3. Put data point in the cluster of the closest centroid
   - Cluster for which $d_{\text{cluster}}$ is minimum
K–means

1. Initialize a set of clusters randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   
   \[ d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \]

3. Put data point in the cluster of the closest centroid
   
   • Cluster for which \( d_{\text{cluster}} \) is minimum

4. When all data points are clustered, recompute centroids

   \[ m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i x_i \]

   • We do not explicitly compute the means
   • May be impossible – we do not know the high-dimensional space
   • We only know how to compute inner products in it
Kernel K–means

1. Initialize a set of clusters randomly

2. For each data point \( x \), find the distance from the centroid for each cluster
   \[ d_{\text{cluster}} = \text{distance}(x, m_{\text{cluster}}) \]

3. Put data point in the cluster of the closest centroid
   - Cluster for which \( d_{\text{cluster}} \) is minimum

4. When all data points are clustered, recompute centroids
   \[ m_{\text{cluster}} = \frac{1}{\sum_{i \in \text{cluster}} w_i} \sum_{i \in \text{cluster}} w_i x_i \]

5. If not converged, go back to 2

- We do not explicitly compute the means
- May be impossible – we do not know the high-dimensional space
- We only know how to compute inner products in it
How many clusters?

• Assumptions:
  – Dimensionality of kernel space > no. of clusters
  – Clusters represent separate *directions* in Kernel spaces

• Kernel correlation matrix $\mathbf{K}$
  – $K_{ij} = K(x_i, x_j)$

• Find Eigen values $\Lambda$ and Eigen vectors $\mathbf{e}$ of kernel matrix
  – No. of clusters = no. of dominant $\lambda_i (1^T \mathbf{e}_i)$ terms
Spectral Methods

- “Spectral” methods attempt to find “principal” subspaces of the high-dimensional kernel space.
- Clustering is performed in the principal subspaces:
  - Normalized cuts
  - Spectral clustering
- Involves finding Eigenvectors and Eigen values of Kernel matrix
- Fortunately, provably analogous to Kernel K-means
Other clustering methods

- Regression based clustering
- Find a regression representing each cluster
- Associate each point to the cluster with the best regression
  - Related to kernel methods
Clustering..

• Many many other variants
  – Many applications..
  – Important: Appropriate choice of feature
    • Appropriate choice of feature may eliminate need for kernel trick..

• Key Features:
  – Identifies latent structure in the distribution of the data
  – Provides an L2-sense optimal quantized representation of the data
    • We will build on this in the next class