Definition: Support Vectors

- A regular linear discriminant function is:
  \[ g(x) = \mathbf{w}^T \mathbf{x} + b \]

- A support-vector based linear discriminant function is:
  \[ g(x) = \mathbf{w}^T \mathbf{x} + b = \sum_{i \in SV} \alpha_i \mathbf{x}_i^T \mathbf{x} + b \]
  Here we represent \( \mathbf{w} \) by a set of support vectors \( \mathbf{x}_i \).

Non-Linear Cluster Separation

- Clusters that are linearly separable work out great:

- But what to do when the separation is non-linear?

- How about… mapping data to a higher-dimensional space:

Non-linear Separation: Feature Space

- General idea: the original input space can be mapped to some higher-dimensional feature space where the training set is separable:

Adapted from Jinwei Gu

Adapted from www.iro.umontreal.ca/~pift6080/documents/papers/svm_tutorial.ppt
The Kernel Trick

- With this mapping, our discriminant function is now:

\[ g(x) = w^T \phi(x) + b = \sum_{i \in SV} \alpha_i \phi(x_i)^T \phi(x) + b \]

- No need to know this mapping explicitly, because we only use the dot product of feature vectors.

- A kernel function is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space:

\[ K(x_i, x_j) \equiv \phi(x_i)^T \phi(x_j) \]

A kernel function is defined as a function that corresponds to a dot product of two feature vectors in some expanded feature space.

### Examples of commonly-used kernel functions:

- Linear kernel:  \( K(x_i, x_j) = x_i^T x_j \)
- Polynomial kernel:  \( K(x_i, x_j) = (1 + x_i^T x_j)^p \)
- Gaussian (Radial-Basis Function (RBF)) kernel:

\[
K(x_i, x_j) = \exp\left( -\frac{\|x_i - x_j\|^2}{2\sigma^2} \right)
\]
- Sigmoid:

\[
K(x_i, x_j) = \tanh(\beta_0 x_i^T x_j + \beta_1)
\]

In general, functions that satisfy Mercer's condition can be kernel functions.

Adapted from Jinwei Gu

---

### The Kernel Trick

- An example:

2-dimensional vectors \( x=[x_1, x_2] \):

\[
K(x_i, x_j) = \begin{pmatrix} 1 + x_i^T x_j \end{pmatrix}^2.
\]

Need to show that \( K(x_i, x_j) = \phi(x_i)^T \phi(x_j) \):

\[
K(x_i, x_j) = \begin{pmatrix} 1 + x_i^T x_j \end{pmatrix}^2 = 1 + 2x_i^1 x_j^1 + 2x_i^1 x_j^2 + 2x_i^2 x_j^1 + 2x_i^2 x_j^2 = \phi(x_i)^T \phi(x_j),
\]

where \( \phi(x) = \begin{pmatrix} 1 & x_1^T x_2^T \end{pmatrix}^T \).

---

### Kernel k-Means

Standard k-means algorithm:

\[
D(\{\pi_i\}_{i=1}^k) = \sum_{c=1}^k \sum_{a_i \in \pi_c} \|a_i - m_c\|^2, \text{ where } m_c = \frac{\sum_{a_i \in \pi_c} a_i}{|\pi_c|}.
\]

Kernel k-means:

\[
D(\{\pi_i\}_{i=1}^k) = \sum_{c=1}^k \sum_{a_i \in \pi_c} \|\phi(a_i) - m_c\|^2, \text{ where } m_c = \frac{\sum_{a_i \in \pi_c} \phi(a_i)}{|\pi_c|}.
\]

Distance function (expanded):

\[
\phi(a_i) \cdot \phi(a_j) - \frac{2 \sum_{a_i \in \pi_c} \phi(a_i) \cdot \phi(a_j)}{|\pi_c|} + \frac{\sum_{a_i, a_j \in \pi_c} \phi(a_i) \cdot \phi(a_j)}{|\pi_c|^2}.
\]

Note that these are all dot products, so we can use a kernel:

\[
K_{ij} = \phi(a_i) \cdot \phi(a_j).
\]

Adapted from Jinwei Gu
Computational Complexity

Computational complexity is higher than with standard k-means:

\[ \phi(a_i) \cdot \phi(a_i) = \sum_{a_j \in \mathcal{C}_i} \phi(a_i) \cdot \phi(a_j) + \sum_{a_j, a_k \in \mathcal{C}_i} \frac{\phi(a_i) \cdot \phi(a_k)}{|\mathcal{C}_i|}. \]

- First term is constant for a given point \( a_i \).
- Second term is \( O(n) \) for each data point, so we get \( O(n^2) \) for all points.
- Third term is fixed per cluster.
- If \( K \) is sparse then the cost can be lower.
- Also incur the cost for evaluation of the kernel.
- So the total cost is \( O(n^2(t+m)) \)
  where \( t \) = number of iterations and \( m \) = number of dimensions.

Results

Standard k-means

Kernel k-means

Note:
- The use of kernels is very general.
- They can be used in many other clustering strategies.

Kernel PCA

An extension of PCA to non-linear distributions:
- Instead of directly doing a PCA, the \( n \) data points \( x_i \) are mapped into a higher-dimensional feature space:

\[ x_i \rightarrow \varphi(x_i) \]

- Then we solve the standard Eigenvalue problem:

\[ \lambda w = Cw, \quad (2.23) \]

with the covariance matrix \( C = \frac{1}{n} \sum_{i=1}^{n} \varphi(x_i) \varphi(x_i)^T \). From the definition of \( C \) follows that \( Cw \) is a linear combination of the vectors \( \varphi(x_i) \). Thus, \( w \) must lie in the span of \( \varphi(x_1), \ldots, \varphi(x_n) \). Hence, we can write

\[ w = \sum_{i=1}^{n} \alpha_i \varphi(x_i), \quad (2.24) \]

Combining (2.23) and (2.24) gives

\[ \lambda \sum_{i=1}^{n} \alpha_i \varphi(x_i) = \frac{1}{n} \sum_{i,j=1}^{n} \alpha_i (\varphi(x_j)^T \varphi(x_i)) (\varphi(x_j)^T \varphi(x_i)) \quad \forall \lambda. \]

- It defines all \( n \) projections in the span.

Again we can use the kernel trick:

\[ n\lambda K \alpha = K^2 \alpha \]

- Which we can simplify to:

\[ n\lambda = K \alpha \]

The vector \( \alpha \) for each principal component can be then obtained by extracting the eigenvectors of \( K \):
- \( K_{ij} = k(x_i, x_j) \)
Results

Graph Representations

We may represent a dataset as a graph
Clusters then form sub-graphs that can be separated by cuts

A number of graph cutting algorithms have been devised for the segmentation of images
• Mincut, RatioCut, Ncut

We will be using a spectral method to reduce complexity

Some Graph Notations

- Given: data points $X_1, ..., X_n$, pairwise similarities $w_{ij} = s(X_i, X_j)$
- $W = (w_{ij})$ adjacency matrix of the graph
- $d_i = \sum_j w_{ij}$ degree of a vertex
- $D = \text{diag}(d_1, ..., d_n)$ degree matrix
- $|A|$ = number of vertices in $A$
- $\text{vol}(A) = \sum_{i \in A} d_i$

In the following: vector $f = (f_1, ..., f_n)$ interpreted as function on the graph with $f(X_i) = f_i$.

Graph Laplacian

There are many options for the components of $W$
• use all neighbors $\rightarrow$ fully connected graph $\rightarrow$ dense $W$
• use k nearest neighbors only $\rightarrow$ sparse $W$
• use a kernel to weight distances $\rightarrow$ also sparse $W$
  this is also called the Affinity Matrix

Build the Graph Laplacian
• many metrics have been defined

$$L = D - W$$
$$L = D^{-1/2} W D^{-1/2}$$

Find the Eigenvectors and values of $L$
• some also use the adjacency matrix directly
Eigenvectors of the Graph Laplacian

The first Eigenvalues are zero
- the number of zero-values indicates the number of clusters, \( k \)

The next \( k \) Eigenvalues are non-zero
- a small gap between these values corresponds to the graph having small cuts
- a large gap means that the graph has no cuts

Set a matrix \( E \) with the Eigenvectors forming the columns
- an eigenvector can be interpreted as associating values (the coordinate entries of the eigenvector) with the vertices of the graph
- these vertices correspond to rows in the matrix \( E \)
  \[ \rightarrow \] thus k-means clustering of these rows will cluster similar vertices

Cluster \( E \) into \( k \) clusters
- assign a data point \( i \) to cluster \( j \) only if row \( i \) of \( E \) was assigned to cluster \( j \)
- this finalizes the spectral clustering
- in practice need to do some normalizations (omitted here)

Illustrative case:
- two isolated clusters

[Diagram: Kernel-linked graph of 2 isolated clusters, Its adjacency matrix after clustering (stylized)]

Toy Example

In the following slide:
- 200 points sampled from 4 Gaussians
- Gaussian kernel with \( \sigma=1 \)
- top two rows 10-connected graph, bottom two rows fully connected
- in each two rows see un-normalized and normalized Laplacian

Last row:
- threshold the second eigenvector at 0, then the part below 0 corresponds to clusters 1, 2, and the part above 0 to clusters 3, 4
- thresholding the third eigenvector separates clusters 1, 4 from clusters 2, 3
- thresholding the fourth eigenvector separates clusters 1, 3 from clusters 2, 4.
- the first 4 eigenvectors carry all the information about the 4 clusters.

Spectral clustering using k-means on the first 4 eigenvectors easily detects the correct 4 clusters.
Examples

Dataset exhibits complex cluster shapes
⇒ K-means performs very poorly in this space due bias toward dense spherical clusters.

In the embedded space given by two leading eigenvectors, clusters are trivial to separate.
Spectral Clustering - Derek Greene

How to Select k?

**Eigengap:** the difference between two consecutive eigenvalues.

Most stable clustering is generally given by the value $k$ that maximizes the expression

$$\Delta_k = |\lambda_k - \lambda_{k-1}|$$

Largest eigenvalues of Cisi/Medline data

$$\max \Delta_k = |\lambda_2 - \lambda_1|$$

⇒ Choose $k = 2$