Unsupervised Learning

Motivation:
Given a set of training examples with no teacher or critic, why do we learn?
- Feature extraction
- Data compression
- Signal detection and recovery
- Self organization
- Clustering

Information can be found about data from inputs.
Principal Component Analysis

Introduction

Consider a zero mean random vector $x \in \mathbb{R}^n$ with autocorrelation matrix $R = E(xx^T)$.

$R$ has eigenvectors $q(1), \ldots, q(n)$ and associated eigenvalues $\lambda(1) \geq \ldots \geq \lambda(n)$.

Let $Q = [q(1) \mid \ldots \mid q(n)]$ and $\Lambda$ be a diagonal matrix containing eigenvalues along diagonal.

Then $R = Q \Lambda Q^T$ can be decomposed into eigenvector and eigenvalue decomposition.
First Principal Component

Find \( \max w^T R w \) subject to \( ||w||=1 \).

Maximum obtained when \( w = q(1) \) as this corresponds to \( w^T R w = \lambda(1) \).

\( q(1) \) is first principal component of \( x \) and also yields direction of maximum variance.

\( y(1) = q(1)^T x \) is projection of \( x \) onto first principal component.
Other Principal Components

ith principal component denoted by \( q(i) \) and projection denoted by \( y(i) = q(i)^T x \) with \( E(y(i)) = 0 \) and \( E(y(i)^2) = \lambda(i) \).

Note that \( y = Q^T x \) and we can obtain data vector \( x \) from \( y \) by noting that \( x = Qy \).

We can approximate \( x \) by taking first \( m \) principal components (PC) to get \( z: z = q(1)x(1) + \ldots + q(m)x(m) \).

Error given by \( e = x - z \). \( e \) is orthogonal to \( q(i) \) when \( 1 \leq i \leq m \).

All PCA give eigenvalue / eigenvector decomposition of \( R \) and is also known as the Discrete Karhunen Loeve Transform.
Learning Principal Components

- Given m inputs (x(1), x(2), … x(m)) how can we find the Principal Components?
- Batch learning: Find sample correlation matrix 1/m X^TX and then find eigenvalue and eigenvector decomposition. Decomposition can be found using SVD methods.
- On-line learning: Oja’s rule learns first PCA. Generalized Hebbian Algorithm, APEX.
First PC gives more information than second PC.
Learning algorithms for PCA

Hebbian learning rule: when presynaptic and postsynaptic signal are positive, then weight associated with synapse increase in strength.

\[ \Delta w = \mu x y \]
Oja’s rule

Use normalized Hebbian rule applied to linear neuron.

\[ x \overset{w}{\rightarrow} \Sigma \overset{s=y}{\rightarrow} \]

Need normalized Hebbian rule otherwise weight vector will grow unbounded.
Oja’s rule continued

\[ w_i(k+1) = w_i(k) + \mu x_i(k) y(k) \] (apply Hebbian rule)
\[ w(k+1) = w(k+1) / ||w(k+1)|| \] (renormalize weight)

Above rule is difficult to implement so modify to get

\[ w_i(k+1) = w_i(k) + \frac{\mu x_i(k) y(k)}{||w(k+1)||} \]
\[ \approx w_i(k) + \frac{\mu x_i(k) y(k)}{1 + 2 \mu y(k)^2} \]
\[ \approx w_i(k) + \mu y(k) (x_i(k) - y(k) w_i(k)) \]

Similar to Hebbian rule with modified input.

Can show that \( w(k) \to q(1) \) with probability one given that \( x(k) \) is zero mean second order and drawn from a fixed distribution.
Convergence of Oja’s Learning rule

- Stochastic approximation algorithm that converges to \( w \) with probability 1 under set of assumptions including decreasing step size
- Mean convergence

\[
w(k+1) = w(k) + \mu(x(k)x(k)^Tw(k) - w^T(k)x(k)x(k)^T(w(k)w(k))
\]

- Take expectations of both sides and let \( k \) grow large to get that \( Rw = (w^TRw)w \) where \( w = \lim_{k \to \infty} w(k) \)
- Can then show that \( w = q(i) \) where \( q(i) \) is an eigenvector
- Perturb weight and show that \( q(i) \) is first eigenvector
Learning other Principal Components

- Generalized Hebbian Algorithm
  \[ y_j(k) = \sum w_{ji}(k) x_i(k) \text{ (jth output)} \]
  \[ \Delta w_{ji}(k) = \mu y_j(k) (x_i^j(k) - y_j(k) w_{ji}(k)) \text{ (update)} \]
  \[ x_i^j(k) = x_i(k) - \sum_{l=1,j-1} w_{li}(k) y_l(k) \text{ (modified input)} \]

- Adaptive Principal Extraction (APEX)
  Forward linear network and a linear feedback network

- SVD: \[ A = U\Sigma V^T \] (\( \Sigma \) contains singular values)
Applications of PCA

- Matched Filter problem: \( x(k) = s(k) + \sigma v(k) \).
- Image coding (data compression)