Unsupervised learning with neural nets: clustering

Single layer network

\[ x \rightarrow y \rightarrow w \rightarrow x \]

\[ M_i^2 \leq w_{ij} x_i \]

\[ y_i = \begin{cases} 1 & \text{if } M_i > M_j \land i \neq j \\ 0 & \text{otherwise} \end{cases} \]

Only 1 winner in the output layer.

Yet, call \( w_i \) the wt. vector into output neuron \( i \).

\[ w_i^T x^u \geq w_j^T x^u \quad \forall i \neq j, \text{ then} \]

\( i \) is the winner for input pattern \( u \).

\[ |w_i - x^u| \leq |w_j - x^u| \rightarrow \text{winner is neuron } i. \]

Define learning rule: Hebbian, Oja's rule if \( y_i \)'s are binary.

\[ \Delta w_{ij} = \begin{cases} \eta y_i^u (x_j^u - w_{ij}) & \text{if } i \text{ is winner} \\ 0 & \text{wt. decay otherwise} \end{cases} \]

Every time a neuron wins, rotate its wt. close to the input pattern.
Batch:
\[ \Delta w_{ij} = \sum_{\mu} \eta \gamma_{i} (x_{j}^{\mu} - w_{ij}) \]

\[ w_{ij} \rightarrow w_{ij} + \Delta w_{ij} \]

This algorithm is equivalent to the \( K \)-means (Lloyd's) algorithm for clustering:
- Start with any partitioning of data in \( K \)-sets.
- Pick the centroids of each set as the \( K \) cluster centers.
- \( K \) centers create a new partitioning of the data by associating each data point with its closest centroid.
- Recompute centroid.

Properties:
- Highly dependent on initial conditions.
- Not unique clustering.
- Run multiple times with different initial conditions.

"Good" solution maximizes the total inter-cluster variances:
\[ C = \frac{1}{2} \sum_{\mu} \left( x_{i}^{\mu} - w_{i}^{*} \right)^{2} \leq \frac{1}{2} \sum_{\mu, k} \left( x_{i}^{\mu} - w_{i}^{*} \right)^{2} \]

\[ \frac{\partial C}{\partial w_{ij}} \]

\( i \rightarrow \) winner in the output for input \( x^{\mu} \)
\[
\frac{\partial J}{\partial \omega_p} = - \sum_{i=1}^{N} (y_i - \hat{y}_i) \frac{\partial \hat{y}_i}{\partial \omega_p} = - \sum_{i=1}^{N} \frac{(x_i - \mu_x)(x_i - \mu_x)^T}{\sigma_i^2} \omega_p
\]

Linear subspace learning by Principal Component Analysis

\[X : N \times T\]

\[S.t. \quad \frac{1}{T} \sum_{i=1}^{T} x_i = 0\]

\[\text{mean centered for each variable.}\]

\[C = XX^T\]

\[\text{N \times N covariance matrix, square, symmetric}\]

\[\Rightarrow \text{eigenvalues } \lambda_i, \text{ and a diagonal matrix } \Lambda\]

\[C = V^T \Lambda V\]

\[\Lambda = \begin{pmatrix}
\lambda_1 & 0 \\
0 & \lambda_2 \\
& \ddots \\
& & \lambda_n
\end{pmatrix}\]

\[\text{where } \lambda_i \text{'s are the eigenvalues of } C, \text{ and columns of } V \text{ are the eigenvectors of } C.\]

\[\text{Total data variance } = \sum_{i=1}^{N} \lambda_i = tr(C)\]

\[K \text{- principal components of } X : \text{are the } K \text{- eigenvectors corresponding to the top } K \text{- eigenvalues}\]

\[\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_n\]

\[V_1, V_2, \ldots, V_K\]

\[K \text{- principal components.}\]
Dimensionality Reduction:

For any data vector \( \mathbf{x} = \sum_{d=1}^{N} c_d \mathbf{v}_d \)

\[
\mathbf{y} = \mathbf{X}_W \mathbf{x} = \sum_{d=1}^{k} c_d (\mathbf{v}_d^T \mathbf{x}) = \mathbf{V}_d \mathbf{V}_d^T \mathbf{x}
\]

where \( \mathbf{V}_d \) is the submatrix of \( \mathbf{V} \) containing only \( k \) eigenvectors.

- PCA result are unique (unlike k-means clustering) for a given data sample.
- PCA is the optimal linear dimensionality reduction technique in the least squares sense.
- PCA minimizes squared error:

\[
\sum_{t} \left\| \mathbf{x}^t - \mathbf{x}^{(k)}_t \right\|^2 \leq \sum_{t} \left\| \mathbf{x}^t - \mathbf{v}_t \mathbf{v}_t^T \right\|^2
\]
Neural PCA

Sec: Foldiak 1989
Pehlevan, Chklovskii 2015

$Y = WX + (-A)Y_t$  \[ Y_{t+1} = WX - Ay_t \]

$\Delta W_t = Y_{t} X^T - Y_{t} W_{t}$

$\Delta A_t = Y_{t} Y_{t}^T - Y_{t}^2 A_{t}$

$\epsilon_{t+1} = \leq \frac{t}{\epsilon_{t+1}}$

After learning: The $k$ output neurons together represent the $k$-dimensional principal subspace spanned by the input data (though the single neurons may be not represent the $k$-orthogonal eigenvectors of this subspace).

$Y = (I + A)^{-1}WX = F_X$

(neural filter)

Look at work by Pehlevan on similarity alignment (if interested)