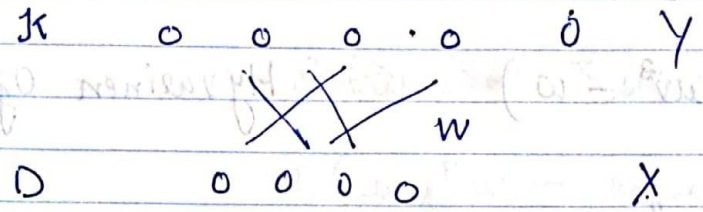
Nov 12, 2019

Unsupervised learning with neural nets: clustering

Single layer network




$u_i = \sum w_{ij} x_j$

$\boxed{\text{Only 1 winner in the output layer}}$

$$y_i = \begin{cases} 1 & \text{if } u_i > u_j \quad \forall j \neq i \\ 0 & \text{otherwise} \end{cases}$$

Lets call $w_i$ the wt. vector into output neuron $i$.

$w_i^T x^\mu \geq w_j^T x^\mu \quad \forall i \neq j$, then $i$ is the winner for input pattern $\mu$.

$|w_i - x^\mu| \leq |w_j - x^\mu| \rightarrow$ winner is neuron $i$.

Define learning rule: Oja's rule if $Y_i$'s are binary.

$$\Delta w_{ij} = \begin{cases} \eta \underbrace{y_i^\mu}_{\text{Hebbian}} \underbrace{(x_j^\mu - w_{ij})}_{\text{wt. decay}} & \text{if } i \text{ is winner} \\ 0 & \text{otherwise.} \end{cases}$$

Every time a neuron wins, rotate its wt. close to the input pattern.

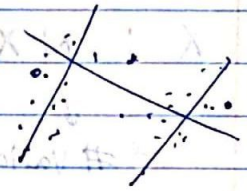
Batch:

$$\Delta w_{ij} = \sum_{\mu} \eta \, y_i \left( x_j^{\mu} - w_{ij} \right)$$

$$w_{ij} \longrightarrow w_{ij} + \Delta w_{ij}$$

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 dependant on initial conditions
- not unique clustering
- Run multiple times with diff. initial conditions.

"Good" solution maximizes the total inter-cluster variances

$$C = \frac{1}{2} \sum_{\mu} \left| x^{\mu} - w_i \right|^2 = \frac{1}{2} \sum_{\mu, k} \left( x_k^{\mu} - w_{i,k} \right)^2$$

center of cluster to which $x^{\mu}$ belongs

$$\frac{\partial C}{\partial w_{ij}}$$

$i \rightarrow$ winner in the output for input $x^{\mu}$

$$\frac{\partial C}{\partial w_{\alpha\beta}} = -\sum_{\mu,k} (x_k^\mu - w_{ik}) \underbrace{\frac{\partial w_{ik}}{\partial w_{\alpha\beta}}}_{\delta_{i\alpha}\,\delta_{i\beta}} = -\sum_{\mu} (x_\beta^\mu - w_{i\beta})\, \underset{y_i}{\delta_{i\alpha}}$$

Lineal subspace learning by Principal Component Analysis

$X : N \times T$ s.t $\frac{1}{T}\sum_{t=1}^{T} x_i^t = 0$

$N$ → # Variable

$T$ → # of samples of each variable

mean centered for each variable.

$C = X X^T$ $N \times N$ covariance matrix

square, symmetric ⇒ orthonormal matrix $V$, and a diagonal matrix $\Lambda$

$C = V^T \Lambda V$ $\Lambda = \begin{pmatrix} \lambda_1 & & 0 \\ & \lambda_2 & \\ 0 & & \ddots \; \lambda_n \end{pmatrix}$

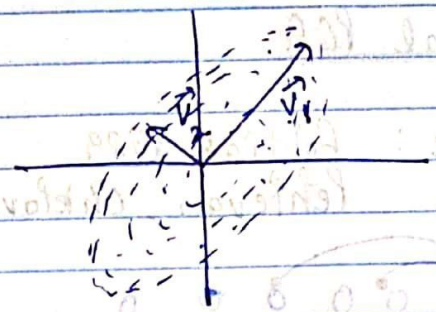
where $\lambda_i$'s are the eigenvalues of $C$, and columns of $V$ are the eigenvectors of $C$.

Total data Variance $= \sum_i \lambda_i = tr(C)$

$K$ - Principal components of $X$ : are the $K$- eigenvectors corresponding to the top $K$- eigenvalues

$\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n$

$V_1, V_2, \ldots V_n$ $K$- principal components.




Dimensionality Reduction:

For any data vector $x = \sum_{\alpha=1}^{N} c_\alpha V_\alpha$

$$= \sum V_\alpha \underbrace{(V_\alpha^T x)}_{c_\alpha} = VV^T x$$

change of basis

$$\hat{x}^{(k)} = \sum_{\alpha=1}^{k} c_\alpha V_\alpha = V^{(k)} V^{(k)T} x$$

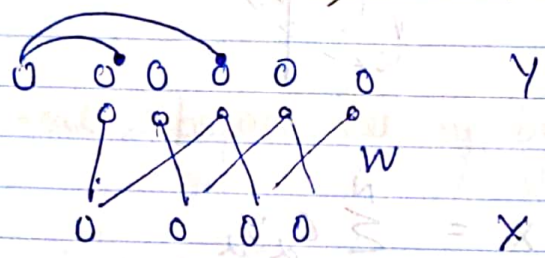
where $V^{(k)}$ is the submatrix of $V$ containing only '$k$' eigenvectors.

- PCA → results 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:

$$\sum_t \| x^t - \hat{x}^{(k)t} \|^2 = \sum_t \| x^t - V^{(k)} V^{(k)T} x \|^2$$

# Neural PCA

See : Foldiak 1989
Pehlevan, Chklovskii 2015




• : Anti Hebbian learning

o : Hebbian

$$y^{t} = Wx^{t} + (-A)y^{t} \Rightarrow y^{t} = Wx^{t} - Ay^{t}$$

$$\Delta W^{t} = \underbrace{y^{t}x^{t^T} - y^{t^2}W^{t}}_{d^{t+1}}$$

$$\Delta A^{t} = \underbrace{y^{t}y^{t^T} - y^{t^2}A^{t}}_{d^{t+1}}$$

$$d^{t+1} = \sum_{t'=1}^{t} y^{T}y^{t}$$

<u>After learning:</u> The 'k' output neurons together represent the k-dimensional principal subspace spanned by the input data (though the single neurons may not represent the k-orthogonal eigenvectors of this space).

$$y = \underbrace{(I+A)^{-1}W}_{F}x = Fx$$

$\hookrightarrow$ neural filter.

$\Rightarrow$ Look at work by 'Pehlevan' on similarity alignment (if interested)