PCA as an Unsupervised Hebbian Learning Methodology

Prashant Joshi (joshi@igi.tugraz.at)
Unsupervised Hebbian Learning

Introduction to PCA (very short)
- What is PCA?
- Algorithm for computing PCA

Single linear unit from Hebbian learning perspective

Oja’s rule
- Idea
- With spiking neurons
- Theory and proof

Sanger’s rule

Recent model of Oja like rule for synaptic scaling (Abbott et al, 2002)
Unsupervised Hebbian Learning

- No teacher available
- Network must discover for itself patterns, features, regularities, correlations or categories in the input data and code for them in the output
- Can only do something useful when there is redundancy in the input data
- Question: What sort of things such a network might tell us?
Unsupervised Hebbian Learning

Such networks can tell

- **Familiarity** – How similar is a new input to typical patterns seen in the past
- **Principal component analysis** – Extending familiarity to multiple-component basis to measure similarity
- **Clustering** – A set of binary valued outputs, with only one on at a time, telling which of the several categories an input pattern belongs to
- **Prototyping** – Network forms categories, but returns a typical prototype as output
- **Encoding** – Output could be encoded version of the input, in general using fewer bits.
- **Feature mapping** – If output units have a fixed geometrical arrangement with only one on at a time, they could map different inputs to different points in arrangement
Principal Component Analysis

- A.k.a Karhunen-Loe`ve transform in communication theory
- Uses the leading eigenvector directions of the input patterns correlation matrix
- Aim is to find a set of $M$ orthogonal vectors in a $N$ dimensional data space, that account for as much as possible of the data’s variance
  - Typically, $M \ll N$ making the reduced data much easier to handle
Algorithm for PCA

1. Take the N-dimensional data set and subtract the mean from each dimension.
2. Calculate the covariance matrix
3. Calculate the eigenvectors and eigenvalues of the covariance matrix
4. Order the eigenvectors in order of decreasing eigenvalue
5. Choose the first M eigenvectors to form the feature vector
6. Project the zero mean data onto these M eigenvectors
The case of one linear unit

- N-dimensional input patterns, $\xi$ drawn from a distribution $P(\xi)$
- After seeing enough samples, the network should tell how well a particular input conforms to the distribution

$$V = \sum_j w_j \xi_j = w^T \xi = \xi^T w$$

- Obviously the output $V$ should become a scalar measure of familiarity (first attempt: try hebbian learning)
  - $\Delta w_i = \eta V \xi_i$, $\eta$ = learning rate
- Above rule implies that frequent input patterns will have the most influence in the long run
- But there is a problem: the weights keep on growing without bound
The case of one linear unit

- Suppose for a moment that there was a stable equilibrium point for \( \mathbf{w} \).
- After learning, the weight vector should remain in the neighborhood of the equilibrium point.

\[
V = \sum_j w_j \xi_j = \mathbf{w}^T \mathbf{\xi} = \mathbf{\xi}^T \mathbf{w}
\]

\[
\Delta \mathbf{w}_i = \eta V \xi_i
\]

\[
0 = \langle \Delta \mathbf{w}_i \rangle = \langle V \xi_i \rangle = \langle \sum_j w_j \xi_j \xi_i \rangle = \sum_j C_{ij} w_j = C \mathbf{w}
\]
The case of one linear unit

\[ 0 = \langle \Delta w_i \rangle = \langle V \xi_i \rangle = \langle \sum_j w_j \xi_j \xi_i \rangle = \sum_j C_{ij}w_j = Cw \]

- The matrix \( C \):
  - is symmetric, i.e. \( C_{ij} = C_{ji} \) → All eigenvalues are real, and eigenvectors are orthogonal (spectral theorem)
  - \( C \) is positive semi-definite → all its eigenvalues are positive or zero
  - Equation above says that at the hypothetical equilibrium point, \( w \) is an eigenvector of \( C \) with eigenvalue 0
- This can’t be stable, as \( C \) necessarily has some positive eigenvalues, so any fluctuation along an eigenvector with positive eigenvalues will grow exponentially.
- The direction with the largest eigenvalue \( \lambda_{\text{max}} \) of \( C \) would eventually become dominant
- In any case \( w \) does not settle down
- There are only unstable fixed points for plain Hebbian learning
Oja’s rule

- Gives a method to stop unconstrained growth of weights as seen in Hebbian learning
- It is a modification of the basic Hebbian rule
- Using this rule the weight vector approaches a constant length $|w| = 1$
- No need of normalization by hand
- After learning, $w$ approaches an eigenvector of $C$ with the largest eigenvalue $\lambda_{\text{max}}$
Oja’s rule

- Corresponds to adding a weight decay proportional to $V^2$ to the plain Hebbian rule

$$\Delta w_i = \eta V (\xi_i - Vw_i)$$

- Question: Does Oja’s rule really make the output $V$ represent familiarity of a particular input $\xi$?
  - **Yes and No:** It does the best it can within the constraints of the architecture
  - As we chose linear units, the output $V$ is just the component of input $\xi$, along the $w$ direction
  - For zero-mean data it is zero on average, whatever the direction of $w$, but it is largest in magnitude for the direction found
  - For non-zero mean data, the average of $V$ itself is maximized for the direction found
  - So in both cases, the direction of $w$ found by the rule gives larger $|V|$’s on average then for any other direction

- Points drawn from another distribution – “unfamiliar” points, would give smaller values of $|V|$, unless they have larger magnitude on average

- Thus the network develops a familiarity index for the distribution as a whole, not necessarily for any particular $\xi$

- **FACT:** Actually Oja’s rule chooses the direction of $w$, to maximize $\langle V^2 \rangle$

- Corresponds to variance maximization at the output, and also to finding a principal component
Oja’s rule with spiking neurons

\[ \Delta w_j = \eta \nu^{post} v_j - \eta w_j (\nu^{post})^2 \]

- Where
  - \( \eta \) = learning rate
  - \( \nu^{post} \) = firing rate of post-synaptic neuron
  - \( v_j \) = firing rate of pre-synaptic neuron
  - \( w_j \) = weight of synapse from pre-synaptic neuron j to post-synaptic neuron i
Theory of Oja’s rule

- We have a lot of claims to prove:

- **Claims:** We have said that Oja’s rule converges to a weight vector \( \mathbf{w} \) with the following properties
  - **Unit length:** \( |\mathbf{w}| = 1 \), or \( \sum_i w_i^2 = 1 \)
  - **Eigenvector direction:** \( \mathbf{w} \) lies in a maximal eigenvector direction of \( \mathbf{C} \)
  - **Variance maximization:** \( \mathbf{w} \) lies in a direction that maximizes \( \langle \mathbf{V}^2 \rangle \)
Oja’s rule gives a weight vector that maximizes the mean square output \(<V^2>\)

This is just the first principal component

It would be desirable to have a M output network that extracts the first M principal components

Oja and Sanger have both designed one-layer feed-forward networks that do just this

In both cases, the networks are linear, with the i^{th} output \(V_i\) given by:

\[
V_i = \sum_j w_{ij} \xi_j = w_i^T \xi = \xi^T w_i
\]
Sanger’s rule

- Oja’s M-unit rule is: \[ \Delta w_{ij} = \eta V_i (\xi_j - \sum_{k=1}^{N} V_{kW_{kj}}) \]

- And Sanger’s rule is: \[ \Delta w_{ij} = \eta V_i (\xi_j - \sum_{k=1}^{i} V_{kW_{kj}}) \]

- We shall focus on Sanger’s rule only

- Note that:
  - Both rules differ only in the upper limit of summation
  - Both reduce to Oja’s one unit rule for M = 1, the single output case
Sanger’s rule

\[ \Delta w_{ij} = \eta V_i (\xi_j - \sum_{k=1}^{i} V_k w_{kj}) \]

- Extracts the first M principal components individually in order
- Performs exactly the Karhunen-Loève transform
- Different outputs are statistically uncorrelated and their variance decreases steadily with increasing i
- Note that both these learning rules are non-local
- Sanger suggested a modification to his rule which does allows a local implementation
Simply separate out the $k = i$ term, i.e.

$$\Delta w_{ij} = \eta V_i \left[ (\xi_j - \sum_{k=1}^{i-1} V_k w_{kj}) - V_i w_{ij} \right]$$
Hebbian plasticity is a positive-feedback process, and as a result is unstable by itself. An effective way of controlling this instability is to augment Hebbian modification with additional processes that are sensitive to the postsynaptic firing rate or to the total level of synaptic efficacy. A frequent approach in neural network models is synaptic scaling – to globally adjust all the synapses onto each postsynaptic neuron based on its level of activity. Can take two forms:
- subtractive – all synapses are modified by same amount
- multiplicative – synapses are modified proportional to their strength

Oja’s rule is an example of multiplicative synaptic scaling.
Synaptic scaling

- Both subtractive and multiplicative adjustments lead to competition due to effective normalization.
- Evidence exists for the presence of synaptic scaling in cultured networks of neocortical, hippocampal and spinal-cord neurons.
- Some biophysical mechanisms for synaptic scaling are understood.
- Direct application of glutamate and fluorescent labeling of receptors show that synaptic scaling is due to a postsynaptic change in the number of functional glutamate receptors.
Synaptic scaling
Synaptic scaling

- Synaptic scaling in combination with LTP and LTD seems to generate something similar to Oja’s rule.
- Oja’s rule combines Hebbian plasticity with a term that multiplicatively decreases the efficacy of all synapses at a rate proportional to the square of the postsynaptic firing rate.
Synaptic scaling

According to the current model, synaptic scaling can be modeled by adding a non-hebbian term to the normal hebbian modification, so that synapse modification rate is proportional to

\[ r_{\text{pre}} r_{\text{post}} - \phi(r_{\text{post}})w \]

Note that in Oja’s rule:

\[ \phi(r_{\text{post}}) = (r_{\text{post}})^2 \]

Experimental data support a \( \phi(.) \) that is either positive or negative depending on the postsynaptic firing rate
Conclusion

- Hebbian learning is one of the basic unsupervised learning techniques
- Hebbian plasticity by itself is unstable
- Adding a non-hebbian decay term makes the rule more stable
- In Oja’s rule this term is proportional to the square of the postsynaptic firing rate
- Oja’s rule gives us the first principal component of the data
- Sanger’s rule extends the Oja’s rule to give the first M principal components in an ordered fashion
- Both these rules have a local implementation
- Current experimental evidence suggests a synaptic scaling mechanism similar to Oja’s rule with the exception that the decay term is a function that can be positive or negative depending on the post-synaptic firing rate
References

1. Hertz, Krogh and Palmer. *Introduction to the theory of neural computation*
2. Chris Bishop. *Neural networks for pattern recognition*
3. Gerstner and Kistler. *Spiking neuron models*
4. Hagan, Demuth and Beale. *Neural network design*
5. Abbott and Nelson. *Synaptic Plasticity: taming the beast*