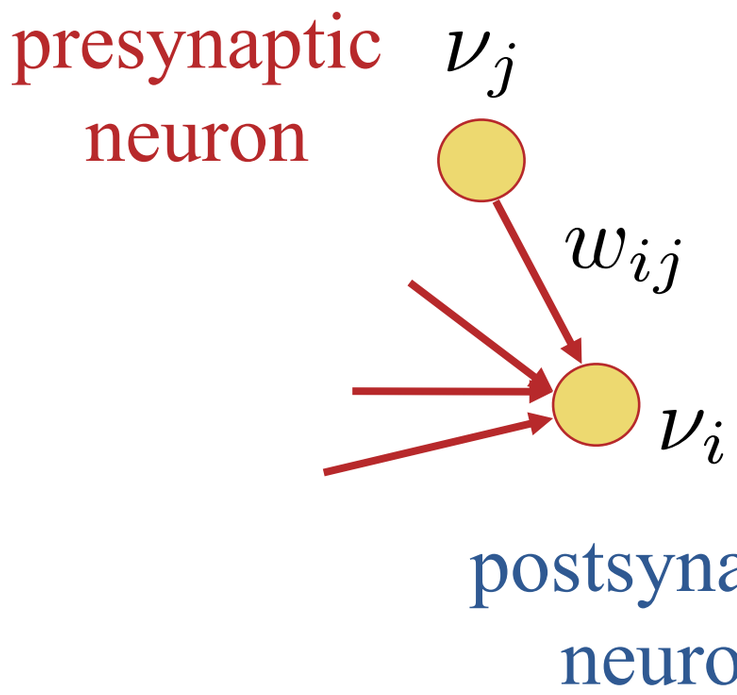
Adaptive Intelligence

Prof. Eleni Vasilaki

The “minimal” Hebbian Rule



In equations

$$\Delta w_{ij} = w_{ij}^{new} - w_{ij}^{old} = \alpha \nu_i \nu_j$$

w_{ij} : weight from neuron j to neuron i

Δw_{ij} : weight change

α : learning rate, >0

Hebbian rules

$$\Delta w_{ij} = \alpha \nu_i \nu_j$$

$$\Delta w_{ij} = \alpha \nu_i \nu_j - c$$

$$\Delta w_{ij} = \alpha (\nu_i - \theta) \nu_j$$

$$\Delta w_{ij} = \alpha (\nu_i - \theta)(\nu_j - \theta)$$

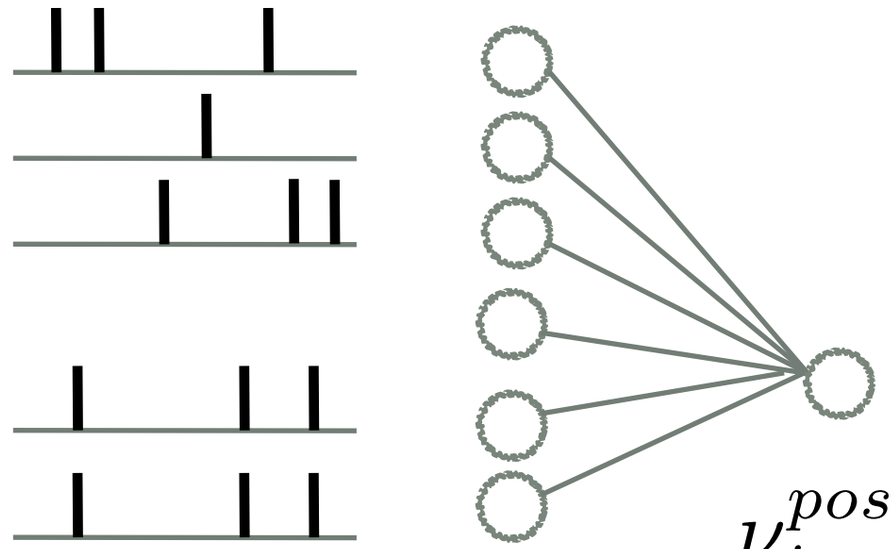
Common “ingredient”

$$\Delta w_{ij} = \alpha \nu_i \nu_j$$

Functional Consequences of Hebbian Learning

Uncorrelated input

Jointly variant (correlated) input



$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

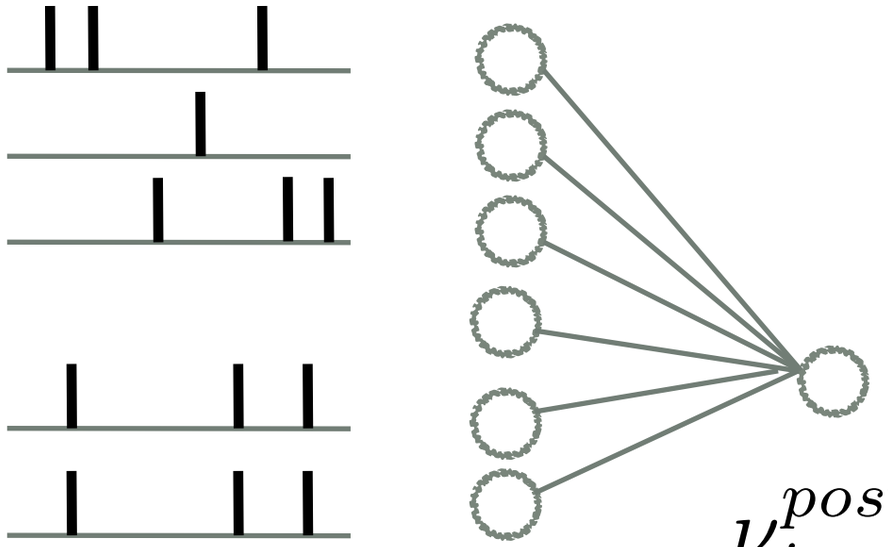
$$\Delta w_{ij} = \alpha \nu_i^{post} \nu_j^{pre}$$

Claim: Detects correlations in the inputs

Functional Consequences of Hebbian Learning

Uncorrelated input

Jointly variant (correlated) input



$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

$$\Delta w_{ij} = \alpha \nu_i^{post} \nu_j^{pre}$$

$$C_{kj} = \langle \nu_k \nu_j \rangle = \frac{1}{P} \sum_{\mu=1}^{P} \nu_k^{\mu} \nu_j^{\mu}$$

Functional Consequences of Hebbian Learning

Learning rule

$$\Delta w_{ij} = \alpha \nu_i^{post} \nu_j^{pre}$$

$$\Delta w_{ij} = \alpha \left(\sum_k w_{ik} \nu_k^{pre} \right) \nu_j^{pre}$$

$$\frac{1}{P} \sum_{\mu=1}^{P} \Delta w_{ij}^{\mu} = \frac{1}{P} \sum_{\mu=1}^{P} \alpha \left(\sum_k w_{ik} \nu_k^{\mu} \right) \nu_j^{\mu}$$

$$\langle \Delta w_{ij} \rangle = \alpha \sum_k w_{ik} \langle \nu_k \nu_j \rangle$$

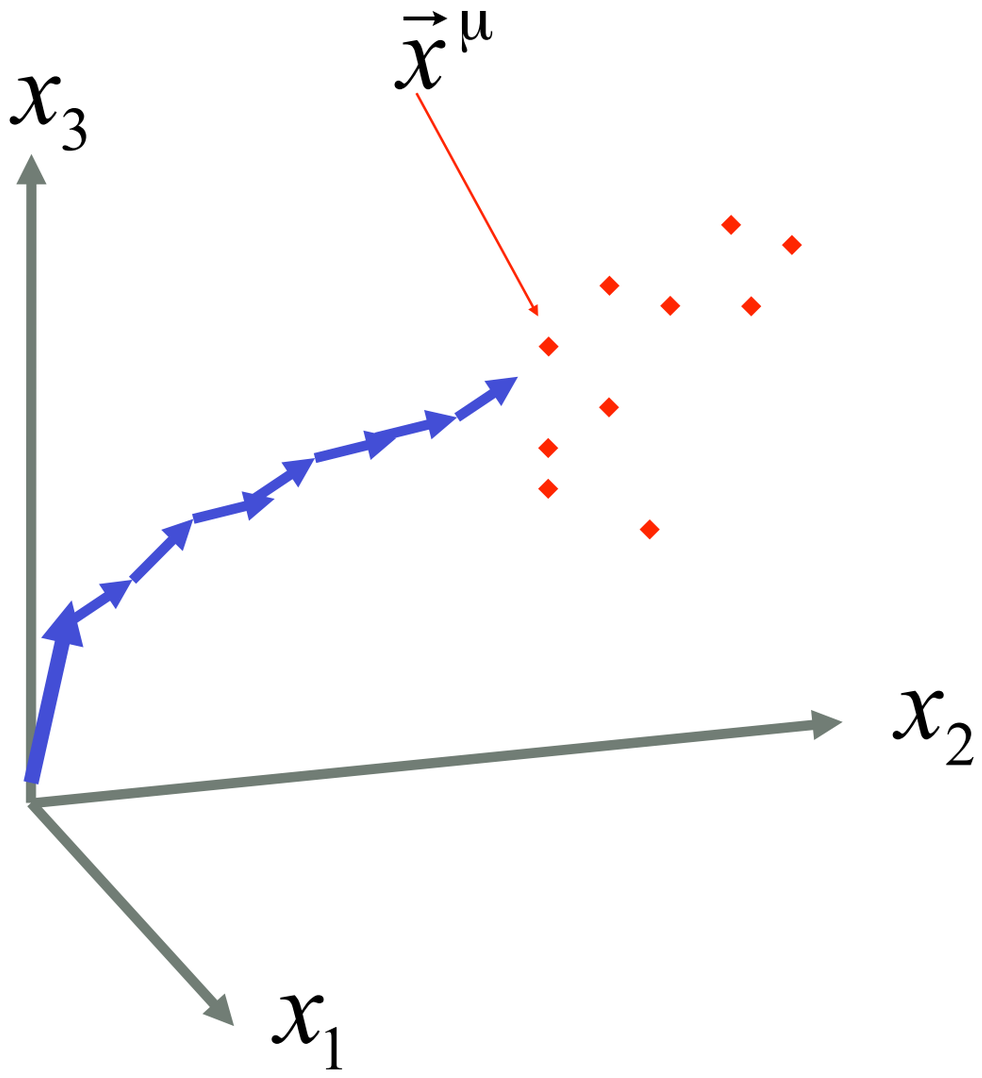
Neuron model

$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

$$C_{kj} = \langle \nu_k \nu_j \rangle = \frac{1}{P} \sum_{\mu=1}^{P} \nu_k^{\mu} \nu_j^{\mu}$$

Detects correlations in the inputs

The minimal Hebbian rule



$$\Delta w_{ij} = \alpha \nu_i^{post} \nu_j^{pre}$$

Unstable!

The BCM rule

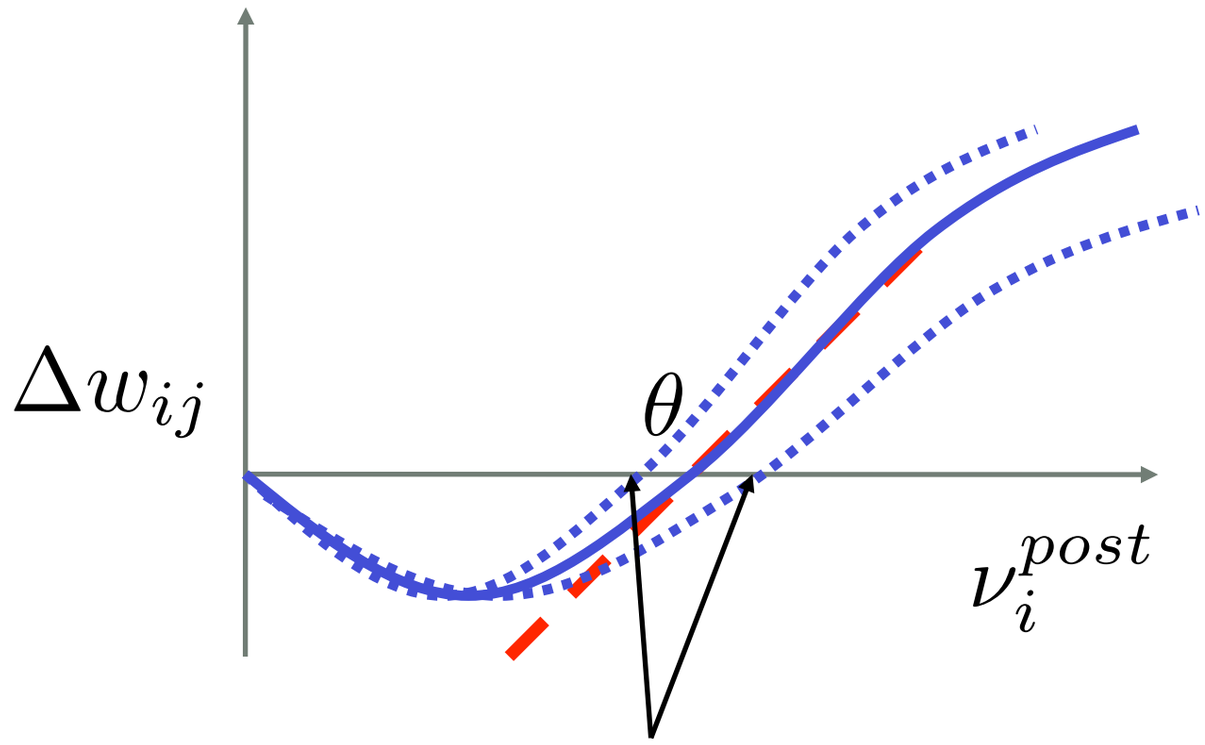
$$\Delta w_{ij} = \alpha \Phi \left(\nu_i^{post} - \theta \right) \nu_j^{pre}$$

Bienenstock, Cooper, Munro 1982

$$\Delta w_{ij} = \alpha \left(\nu_i^{post} - \theta \right) \nu_j^{pre}$$

presynaptically gated

$$\theta = f(< \nu_i^{post} >)$$

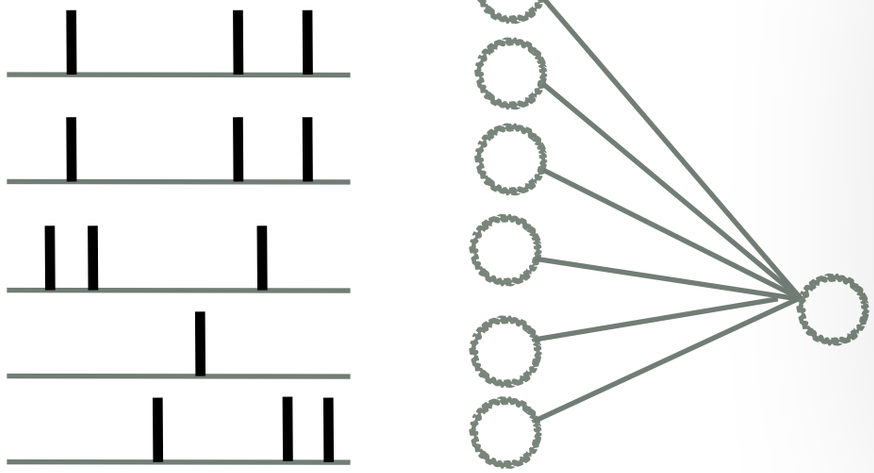


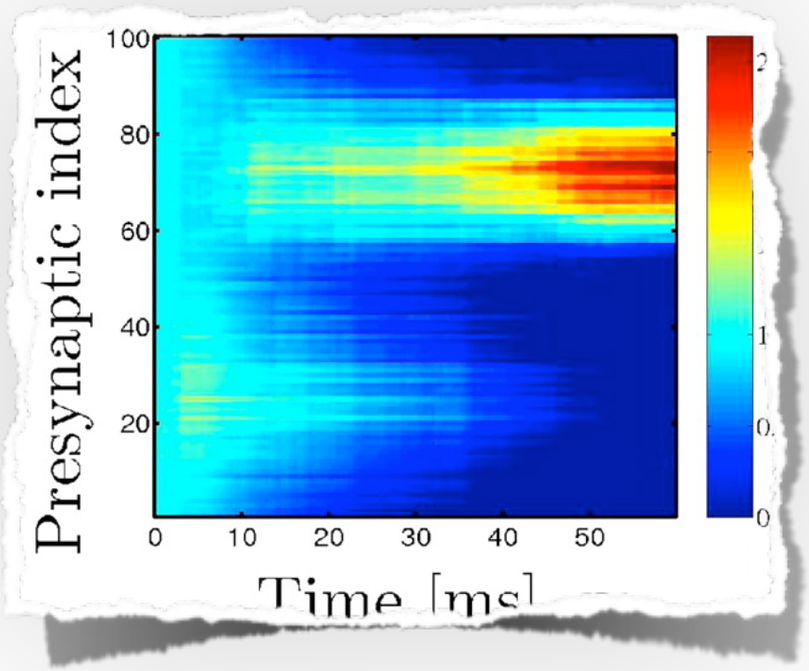
homeostasis

BCM

Jointly variant (correlated) input

Uncorrelated input





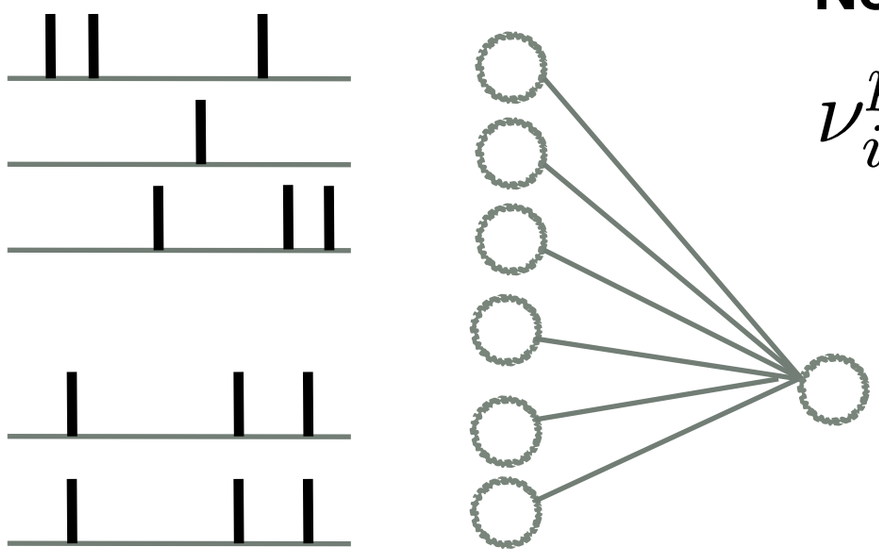
Weight evolution

$$\Delta w_{ij} = \alpha \Phi \left(\nu_i^{post} - \theta \right) \nu_j^{pre}$$

Is this a hebbian rule?

Detects where the ‘interesting’ input occurs (correlations): some synapses strengthened at the expense of others

BCM Rule



Neuron model

$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

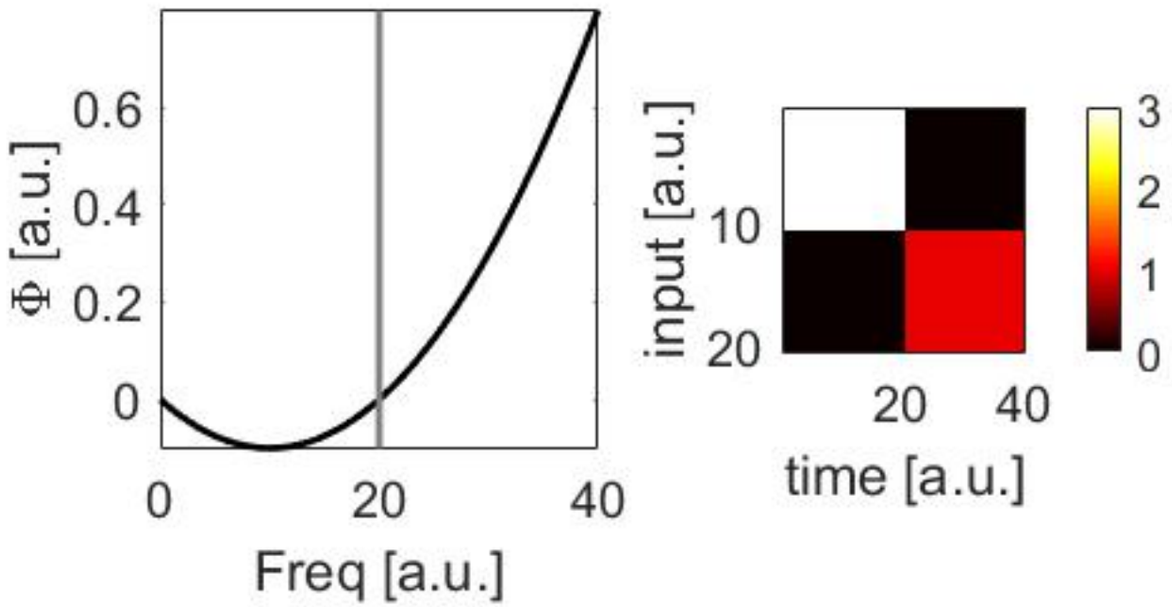
$$\Delta w_{ij} = \alpha \Phi \left(\nu_i^{post} - \theta \right) \nu_j^{pre}$$

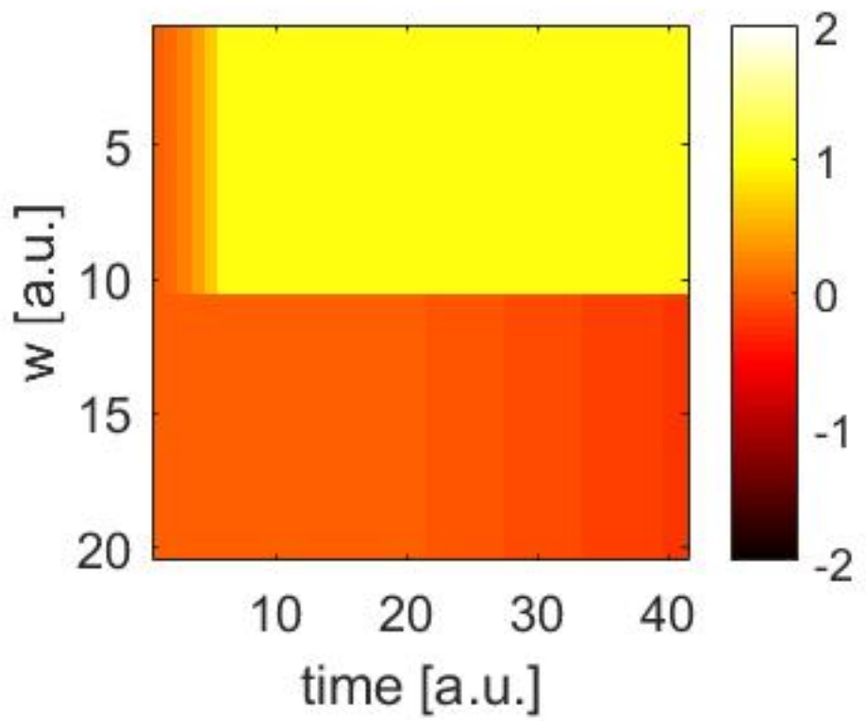
Assume two (presynaptic) groups of 10 neurons each. All weights = 1 and threshold *θ= 20Hz .*

1. Group 1 fires at 3 Hz, then group 2 at 1 Hz. What happens to the weights?
2. Group 1 fires at 3 Hz, then group 2 at 2.5 Hz. What happens?
3. As in 2, but make theta a function of the averaged rate. What happens?

BCM Rule

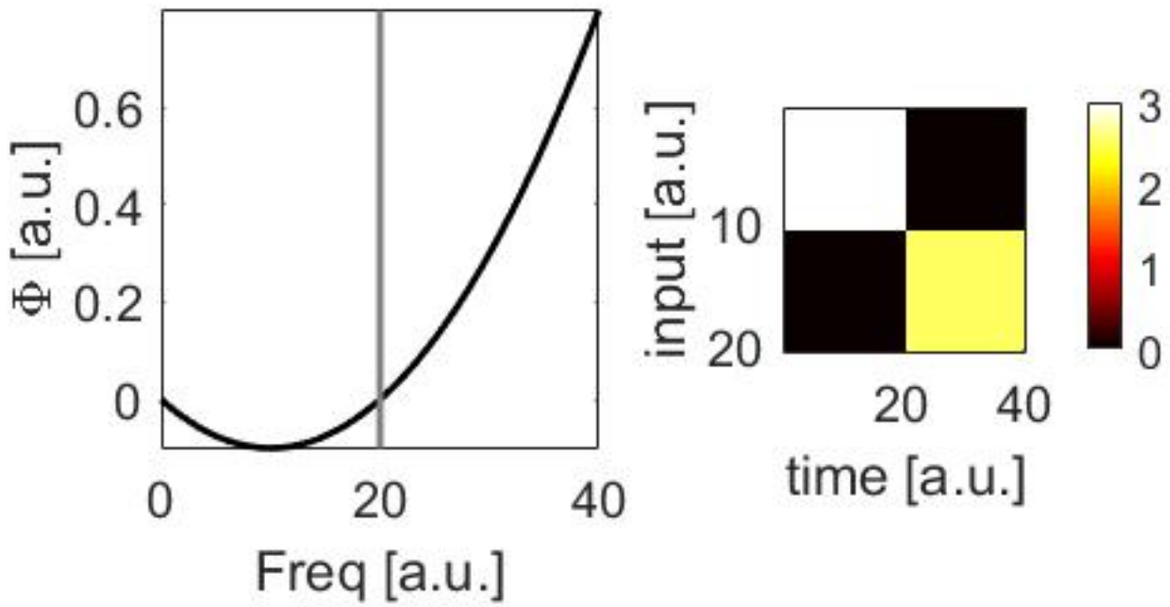
1.

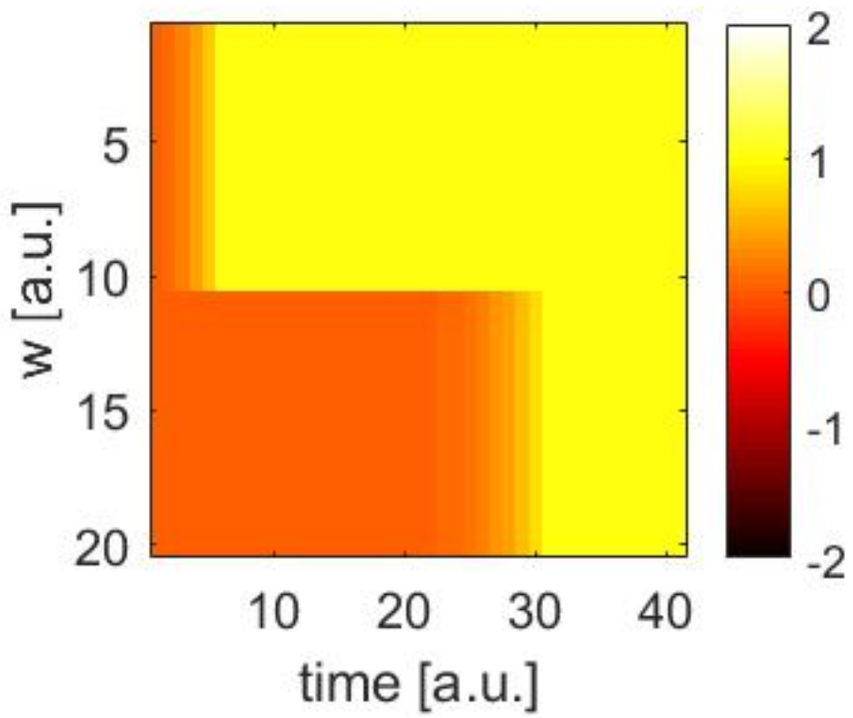

Φ [a.u.]
0.6
0.4
0.2
0
0
20
40
Freq [a.u.]
input [a.u.]
10
20
20
40
time [a.u.]
3
2
1
0


w [a.u.]
5
10
15
20
10
20
30
40
time [a.u.]
2
1
0
-1
-2

BCM Rule

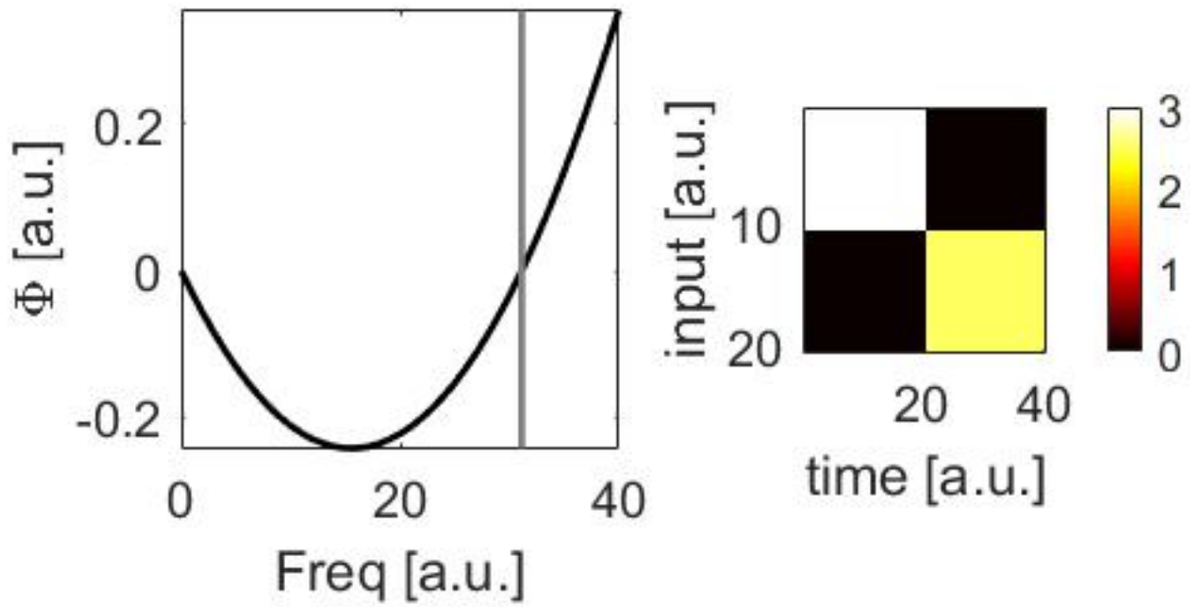
2.

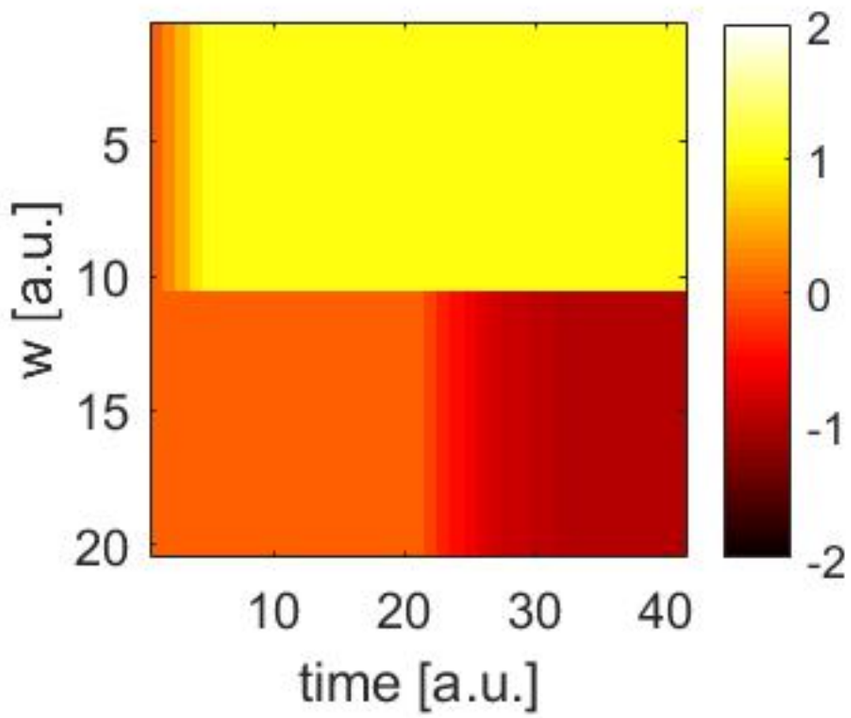

Φ [a.u.]
0.6
0.4
0.2
0
0
20
40
Freq [a.u.]
input [a.u.]
10
20
20
40
time [a.u.]
3
2
1
0


w [a.u.]
5
10
15
20
10
20
30
40
time [a.u.]
2
1
0
-1
-2

BCM Rule

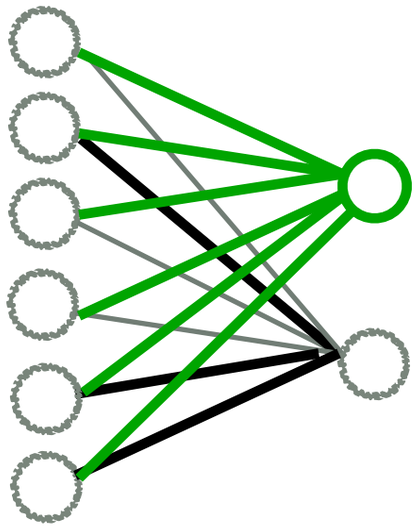
3.


Φ [a.u.]
0.2
0
-0.2
0
20
40
Freq [a.u.]
input [a.u.]
10
20
20
40
time [a.u.]
3
2
1
0


w [a.u.]
5
10
15
20
10
20
30
40
time [a.u.]
2
1
0
-1
-2

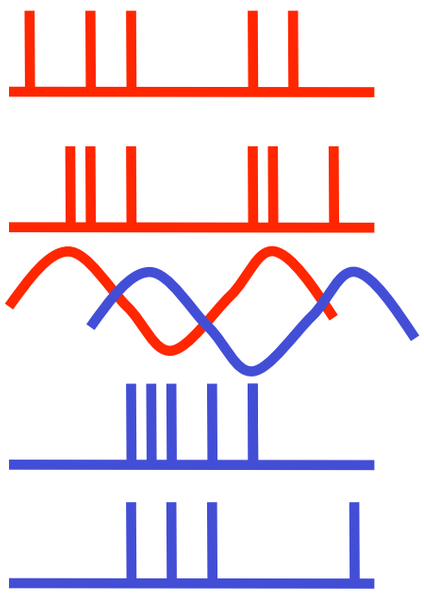
BCM leads to specialised neurons

Initial: random connections



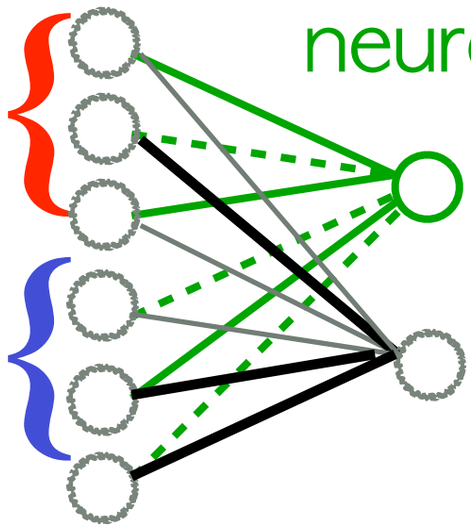
unselective neurons

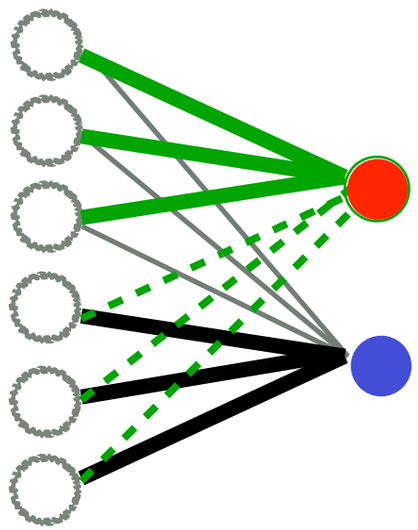
Receptive field development



Correlated input

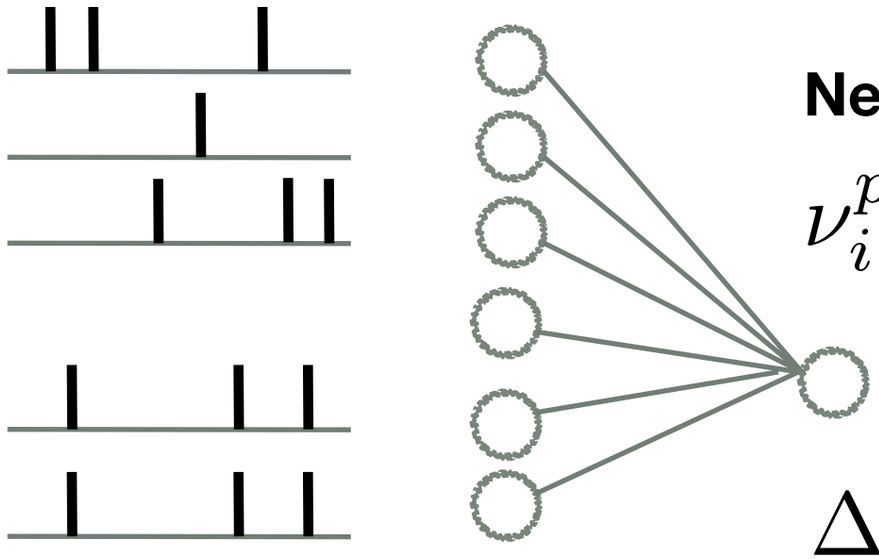
output neurons





output neurons specialize

Oja's rule



Neuron model

$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

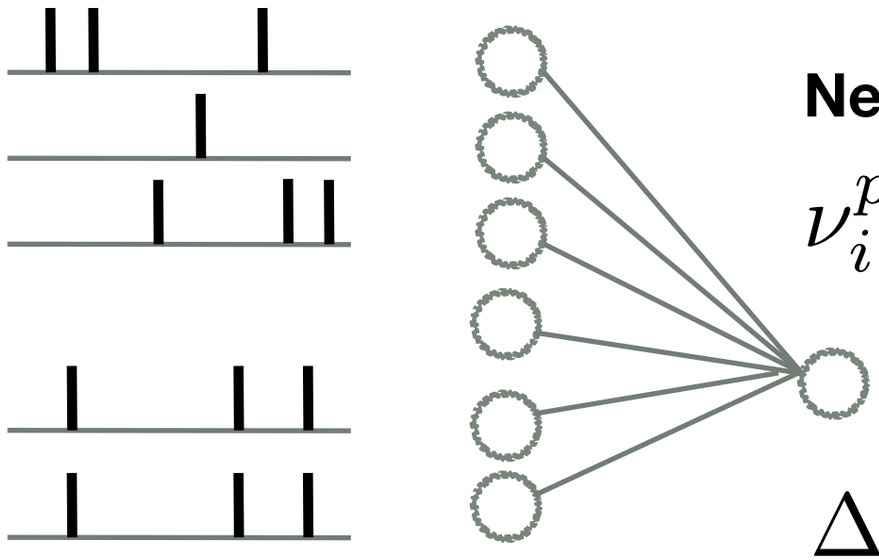
$$\Delta w_{ij} = \alpha \; \nu_i^{post} \nu_j^{pre} - \alpha \; w_{ij} \left(\nu_i^{post} \right)^2$$

Is this a hebbian rule?

What do we know from the earlier results on the "minimal" Hebb rule ?

What advantages this rule have over the "minimal" Hebb rule ?

Oja's rule



Neuron model

$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

$$\Delta w_{ij} = \alpha \; \nu_i^{post} \nu_j^{pre} - \alpha \; w_{ij} \left(\nu_i^{post} \right)^2$$

Show that:

$$\langle \Delta w_{ij} \rangle = \alpha \left(\sum_k w_{ik} \left\langle \nu_k^{pre} \nu_j^{pre} \right\rangle - w_{ij} \sum_{k,n} w_{ik} \left\langle \nu_k^{pre} \nu_n^{pre} \right\rangle w_{in} \right)$$

Oja's rule

Learning rule

$$\Delta w_{ij} = \alpha\ \nu_i^{post} \nu_j^{pre} - \alpha\ w_{ij} \left(\nu_i^{post}\right)^2$$

$$\Delta w_{ij} = \Delta w_{ij}^{+} + \Delta w_{ij}^{-}$$

$$\Delta w_{ij}^{-} = -\alpha\ w_{ij} \left(\nu_i^{post}\right)^2$$

Show $\left\langle \Delta w_{ij}^{-} \right\rangle = -\alpha\ w_{ij} \sum_{k,n} w_{ik} \left\langle \nu_k^{pre} \nu_n^{pre} \right\rangle w_{in}$

Neuron model

$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

$$C_{kj} = \langle \nu_k \nu_j \rangle = \frac{1}{P} \sum_{\mu=1}^{P} \nu_k^{\mu} \nu_j^{\mu}$$

Oja's rule

Show

$$\left\langle \Delta w_{ij}^{-} \right\rangle = -\alpha \; w_{ij} \sum_{k,n} w_{ik} \left\langle \nu_k^{pre} \nu_n^{pre} \right\rangle w_{in}$$

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\nu_i^{post} \right)^2$$

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\sum_k w_{ik} \nu_k^{pre} \right)^2$$

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\sum_k w_{ik} \nu_k^{pre} \right) \left(\sum_k w_{ik} \nu_k^{pre} \right)$$

Neuron model

$$\nu_i^{post} = \sum_k w_{ik} \nu_k^{pre}$$

$$C_{kj} = \left\langle \nu_k \nu_j \right\rangle = \frac{1}{P} \sum_{\mu=1}^{P} \nu_k^{\mu} \nu_j^{\mu}$$

Oja's rule

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\sum_k w_{ik} \nu_k^{pre} \right) \left(\sum_k w_{ik} \nu_k^{pre} \right)$$

Changing the index variable and dropping "pre" - all neurons are now presynaptic:

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\sum_k w_{ik} \nu_k \right) \left(\sum_n w_{in} \nu_n \right)$$

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\sum_{k,n} w_{ik} \nu_k \nu_n w_{in} \right)$$

Oja's rule

$$\Delta w_{ij}^{-} = -\alpha \; w_{ij} \left(\sum_{k,n} w_{ik} \nu_k \nu_n w_{in} \right)$$

$$\frac{1}{P} \sum_{\mu} \Delta w_{ij}^{-\mu} = -\frac{1}{P} \sum_{\mu} \alpha \; w_{ij} \left(\sum_{k,n} w_{ik} \nu_k^{\mu} \nu_n^{\mu} w_{in} \right)$$

By not indexing weights with lowercase mu, implicitly we assume they remain constant.

$$\left\langle \Delta w_{ij}^{-} \right\rangle = -\alpha \; w_{ij} \left(\sum_{k,n} w_{ik} (\frac{1}{P} \sum_{\mu} \nu_k^{\mu} \nu_n^{\mu}) w_{in} \right)$$

$$\left\langle \Delta w_{ij}^{-} \right\rangle = -\alpha \; w_{ij} \sum_{k,n} w_{ik} \left\langle \nu_k \nu_n \right\rangle w_{in}$$

Oja’s rule

- By identifying that Oja’s rule is a hebbian rule we could conclude that previous analysis results on the minimal hebb rule hold here as well.

- This made our work easier in proving that weight changes are driven by correlations in the inputs.

Theory of Oja's rule

- Oja's rule coverages to a weight vector with the following properties:

    1. length equal to 1.

    2. is an eigenvector of the correlation matrix.

❖ Convergence means weight changes equal 0.

Oja's rule

1. $\left|\mathbf{w}\right| = 1$ **for 1 postsynaptic neuron**

Boldface symbols denote vectors

$$\langle \Delta w_{ij} \rangle = \alpha \left(\sum_k w_{ik} \langle \nu_k \nu_j \rangle - w_{ij} \sum_{k,n} w_{ik} \langle \nu_k \nu_n \rangle w_{in} \right)$$

$$0 = \alpha \left(\sum_k w_{ik} \langle \nu_k \nu_j \rangle - w_{ij} \sum_{k,n} w_{ik} \langle \nu_k \nu_n \rangle w_{in} \right)$$

$$\sum_k w_{ik} \langle \nu_k \nu_j \rangle = w_{ij} \sum_{k,n} w_{ik} \langle \nu_k \nu_n \rangle w_{in}$$

Oja's rule

1. $\left|\mathbf{w}\right| = 1$ **for 1 postsynaptic neuron, no need for i index**

$$\sum_k w_k \left\langle \nu_k \nu_j \right\rangle = w_j \sum_{k,n} w_k \left\langle \nu_k \nu_n \right\rangle w_n$$

This equation only captures the weight change from presynaptic neuron j to a single postsynaptic neuron.

Converting it to vectorial form will capture weight changes for all presynaptic neurons.

$$\mathbf{C}\mathbf{w} = \mathbf{w}\left(\mathbf{w}^{\mathbf{T}}\mathbf{C}\mathbf{w}\right)$$

$$\mathbf{C}\mathbf{w} = \mathbf{w}\ \lambda$$

Oja's rule

$$\mathbf{C}\mathbf{w} = \mathbf{w}\left(\mathbf{w}^{\mathbf{T}}\mathbf{C}\mathbf{w}\right)$$

$$\mathbf{C}\mathbf{w} = \mathbf{w}\ \lambda$$

$$\mathbf{w}^{\mathbf{T}}\mathbf{C}\mathbf{w} = \mathbf{w}^{\mathbf{T}}\mathbf{w}\ \lambda$$

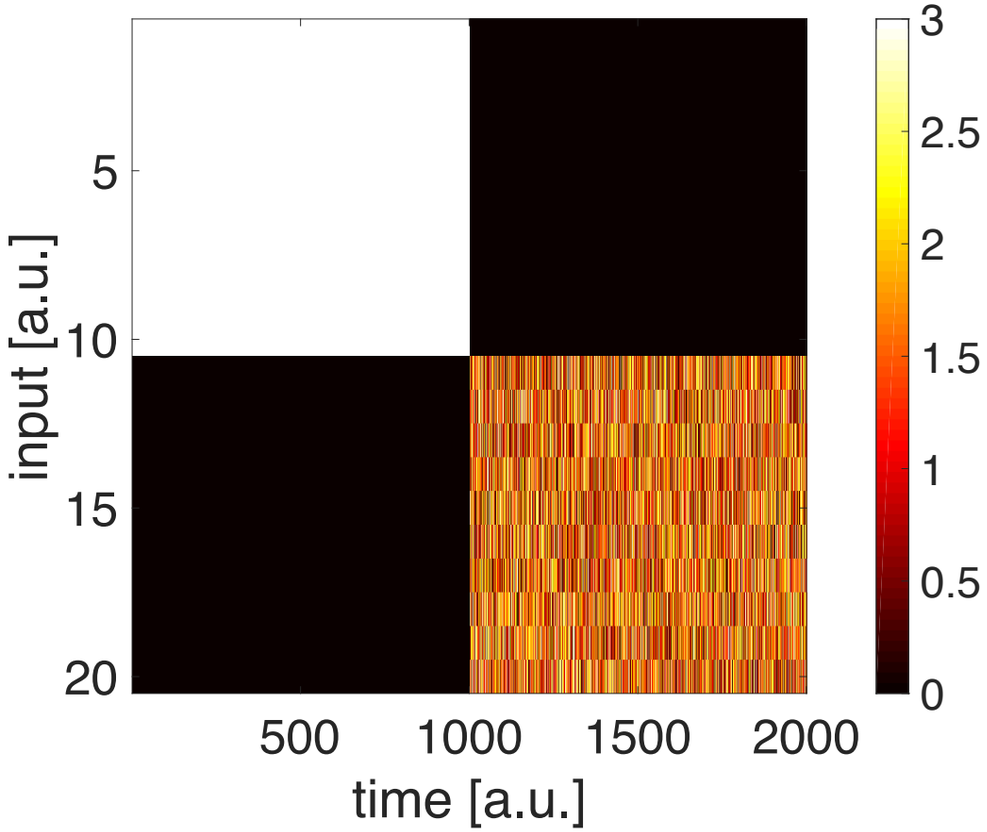
$$\lambda = |\mathbf{w}|^{\mathbf{2}}\ \lambda$$

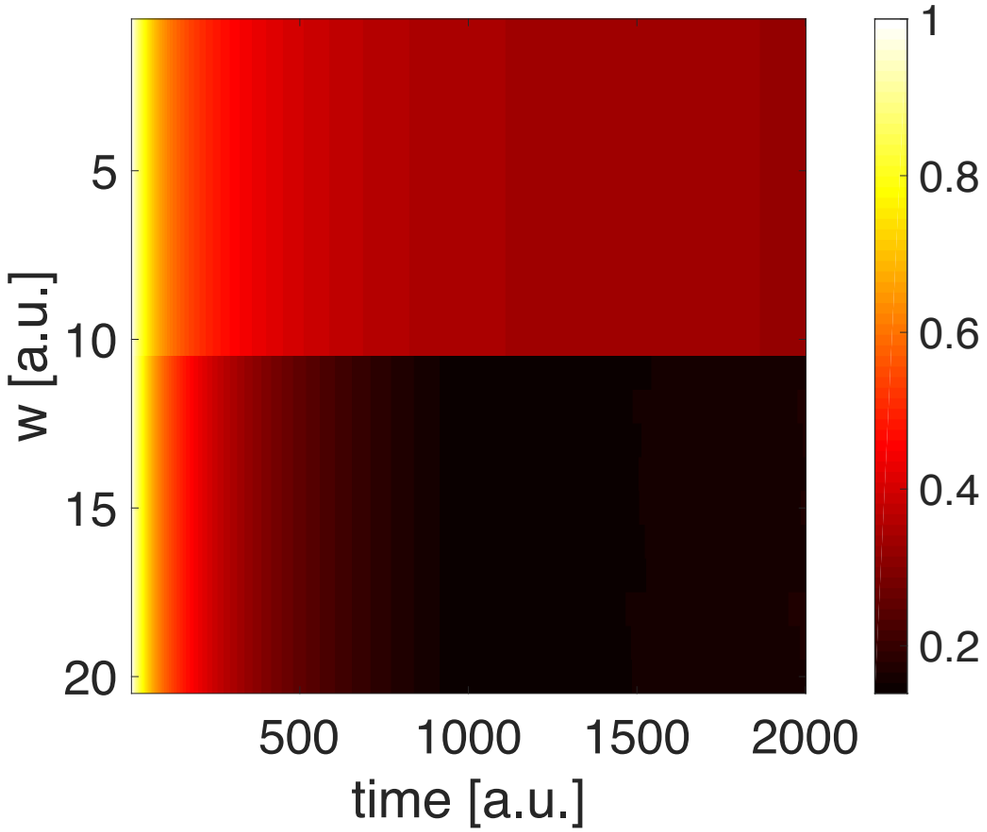
$$|\mathbf{w}|^{\mathbf{2}} = 1$$ **The weight vector is bounded.**

Predictions from the mathematical analysis?

Oja's rule

1. $|\mathbf{w}| = 1$

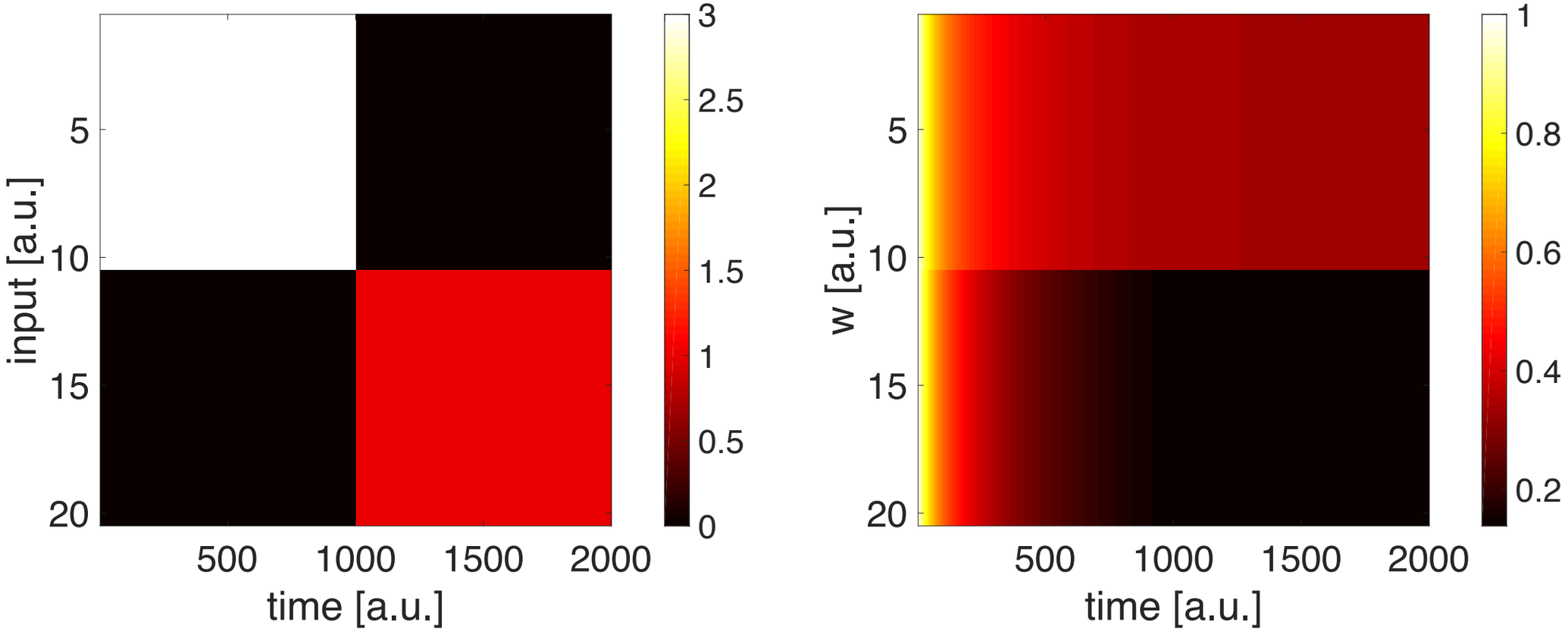




Group 1 fires at 3 Hz, then group 2 randomly.

Oja's rule

1. $|\mathbf{w}| = 1$



Group 1 fires at 3 Hz, then group 2 at 1 Hz.

Oja's rule

2. $\mathbf{w}$ is an eigenvector of the correlation matrix.

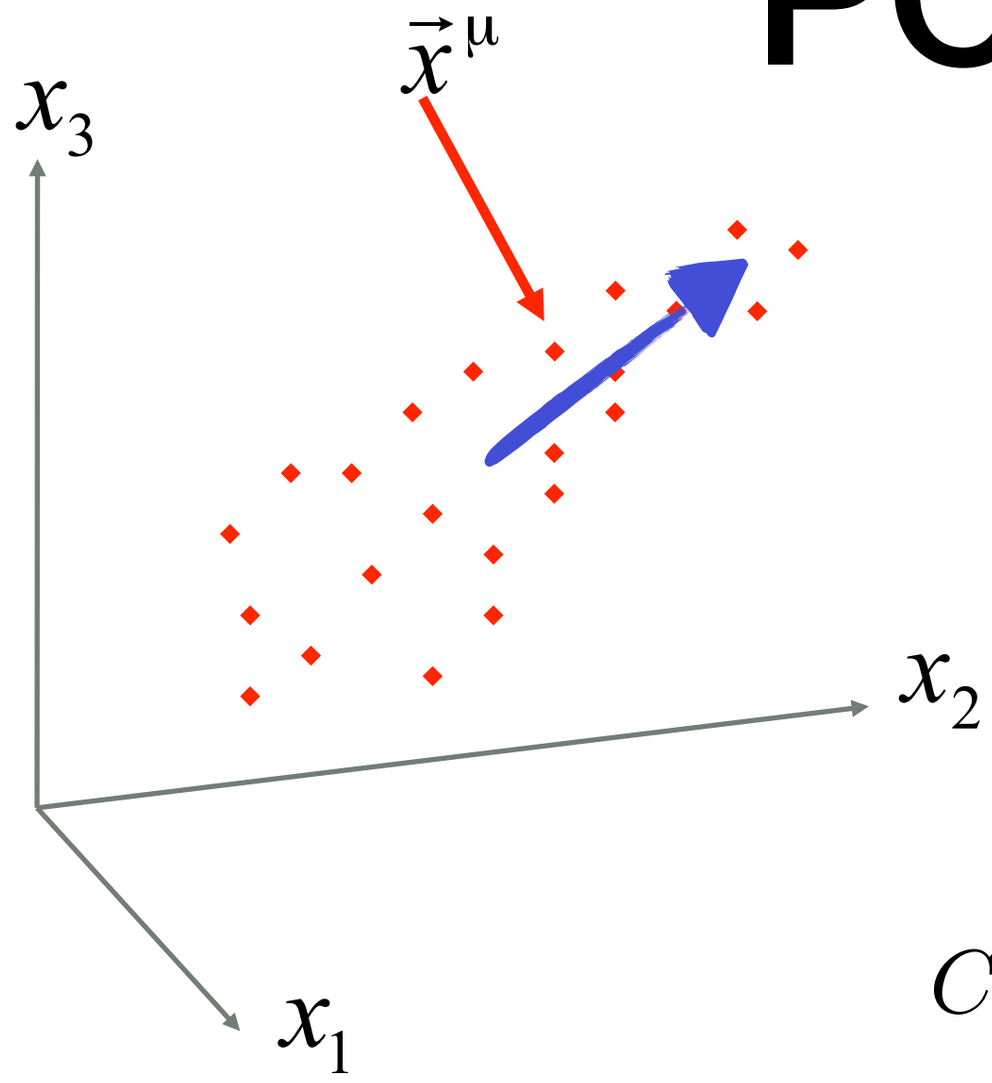
We already show as part of the previous proof:

$$\mathbf{C}\mathbf{w} = \lambda\ \mathbf{w}$$

In fact, the weight vector lies in the direction of the principle component (maximal eigenvector).

Principle Component Analysis

PCA



Correlation:

$$C_{kj} = \langle x_k x_j \rangle = \frac{1}{P} \sum_{\mu=1}^{P} x_k^{\mu} x_j^{\mu}$$

Covariance:

$$C_{kj}^{0} = \langle (x_k - \langle x_k \rangle)(x_j - \langle x_j \rangle) \rangle$$

$$\mathbf{C^0 e_g{}^n} = \lambda^{\mathbf{n}} \mathbf{e_g{}^n} \qquad \lambda^1 > \lambda^2 > \ldots > \lambda^N$$

$\mathbf{e_g{}^1}$ **Principal Component**

PCA

1. Calculate signal means, subtract means from signals.

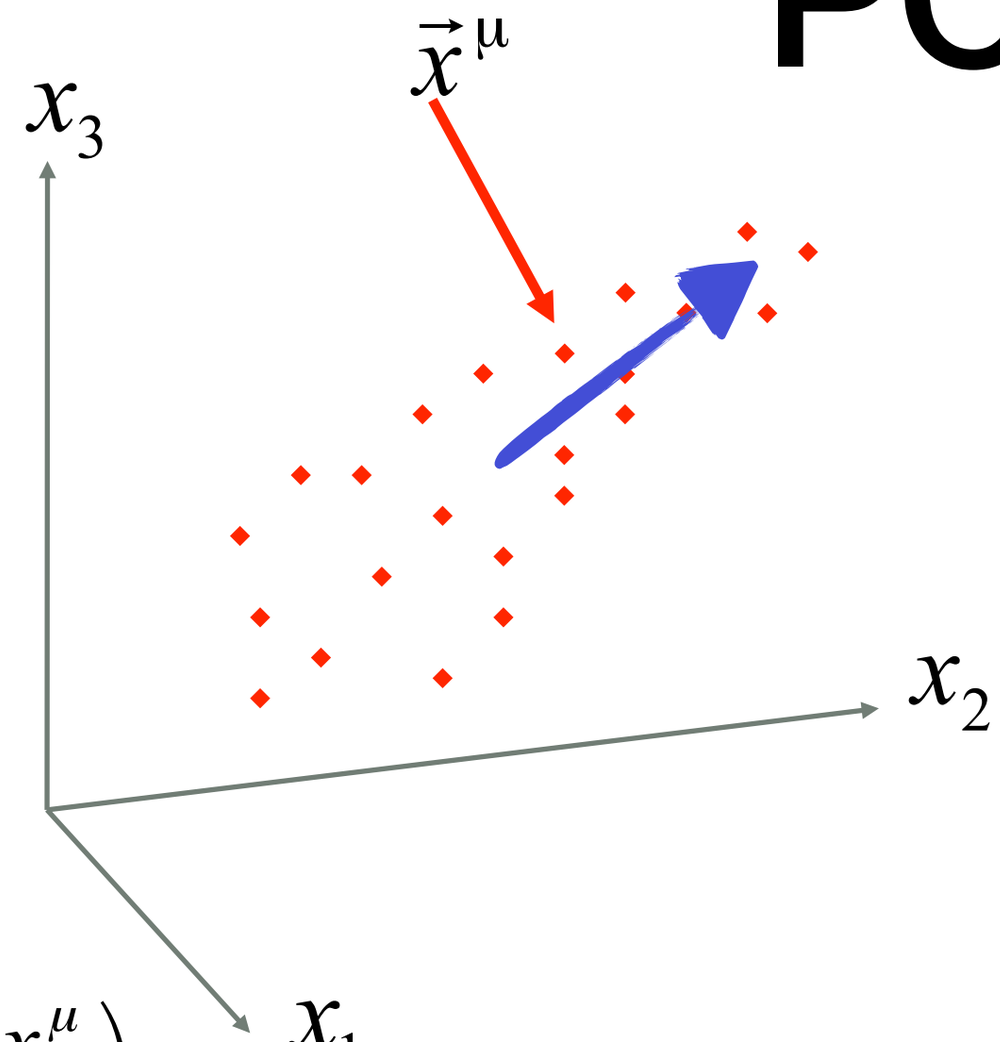
2. Write down the covariance formula. Calculate the covariance matrix.

3. Find the eigenvalues and eigenvectors. Make sure that the length of the eigenvectors is 1!

4. Decide how many eigenvectors you keep. From the eigenvectors form the “Feature Vector”.

5. Derive the new data:
   FinalData=RowFeatureVectorxRowDataAdjust.
   RowFeatureVector has the eigenvectors in the rows with the eigenvector with the highest eigenvalue On TOP.
   RowDataAdjust contains the zero mean data in each column, with each row holding a separate dimension.

PCA



PCA Tutorial:
Lindsay I Smith

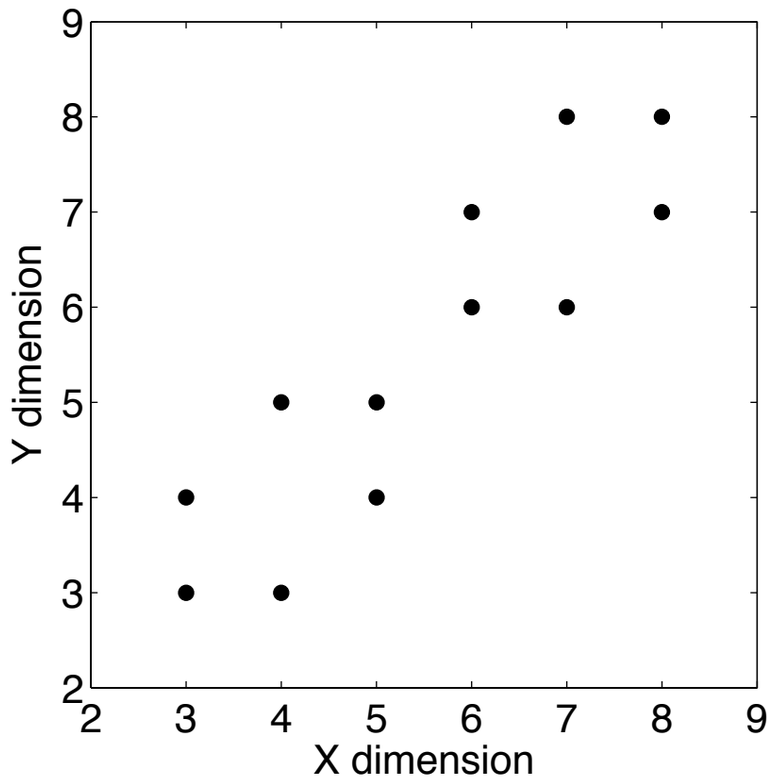
PCA for dimension reduction

$$\vec{x}^{\,\mu} = \begin{pmatrix} x_1^\mu \\ x_2^\mu \\ \dots \\ x_N^\mu \end{pmatrix} \xrightarrow[\text{Rotate via PCA}]{\text{Subtract mean}} \tilde{\vec{x}}^{\,\mu} = \begin{pmatrix} \tilde{x}_1^\mu \\ \tilde{x}_2^\mu \\ \dots \\ \tilde{x}_N^\mu \end{pmatrix} \xrightarrow[\text{First components}]{\text{Keep only}} \begin{pmatrix} \tilde{x}_1^\mu \\ \tilde{x}_2^\mu \end{pmatrix}$$

PCA

Apply the PCA technique on the dataset below in order to reduce its dimensionality.

X	3	3	4	4	5	5	6	6	7	7	8	8
Y	3	4	3	5	4	5	6	7	6	8	7	8



PCA

1. Calculate signal means, subtract means from signals.

X	3	3	4	4	5	5	6	6	7	7	8	8
Y	3	4	3	5	4	5	6	7	6	8	7	8

$$\overline{X} = \sum_{i=1}^{p} \frac{X_i}{p} = \frac{6 * 11}{12} = \frac{11}{2}$$

$$\overline{Y} = \frac{11}{2}$$

PCA

2. Write down the covariance formula. Calculate the covariance matrix.

$$var\left(X\right) = \sum_{i=1}^{p} \frac{\left(X_i - \overline{X}\right)\left(X_i - \overline{X}\right)}{p} = \sum_{i=1}^{p} \frac{\tilde{X}^2}{p} = 35/12$$

$$cov\left(X, Y\right) = \sum_{i=1}^{p} \frac{\left(X_i - \overline{X}\right)\left(Y_i - \overline{Y}\right)}{p} = \sum_{i=1}^{p} \frac{\tilde{X}\tilde{Y}}{p} = 31/12$$

PCA

2. Write down the covariance formula. Calculate the covariance matrix.

$$C = \begin{pmatrix} var(X) & cov(X,Y) \\ cov(Y,X) & var(Y) \end{pmatrix} = \begin{pmatrix} 35/12 & 31/12 \\ 31/12 & 35/12 \end{pmatrix}$$

PCA

3. Find the eigenvalues and eigenvectors. Make sure that the length of the eigenvectors is 1!

$$det(C - \lambda I) = det \begin{pmatrix} 35/12 - \lambda & 31/12 \\ 31/12 & 35/12 - \lambda \end{pmatrix} = 0$$

$$\lambda_1 = 11/2 \qquad \lambda_2 = 1/3$$

PCA

3. Find the eigenvalues and eigenvectors. Make sure that the length of the eigenvectors is 1!

$$CV_1 = \lambda_1 V_1 \qquad CV_2 = \lambda_2 V_2$$

$$V_1 = \begin{pmatrix} \sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix} \qquad V_2 = \begin{pmatrix} -\sqrt{2}/2 \\ \sqrt{2}/2 \end{pmatrix}$$

PCA

4. Decide how many eigenvectors you keep. From the eigenvectors form the “Feature Vector”.

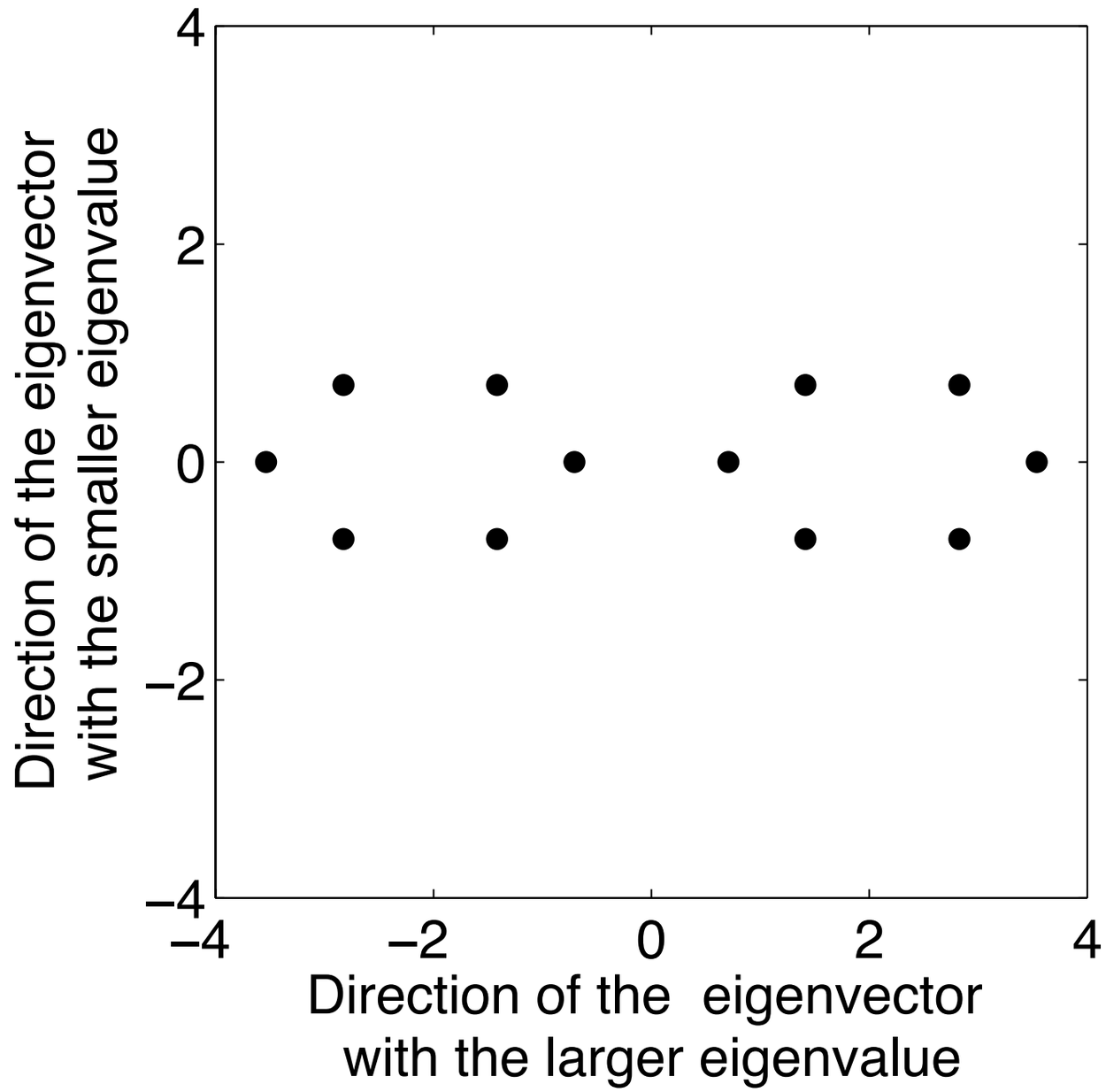
$$F = \begin{pmatrix} \sqrt{2}/2 & -\sqrt{2}/2 \\ \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix}$$

5. Derive the new data:
FinalData=RowFeatureVectorxRowDataAdjust.

RowFeatureVector has the eigenvectors in the rows with the eigenvector with the highest eigenvalue On TOP.
RowDataAdjust contains the zero mean data in each column, with each row holding a separate dimension.

$$D_n = F^T D_c$$

PCA


4
2
0
−2
−4
−4
−2
0
2
4
Direction of the eigenvector with the smaller eigenvalue
Direction of the eigenvector with the larger eigenvalue

PCA

4. Decide how many eigenvectors you keep. From the eigenvectors form the “Feature Vector”.

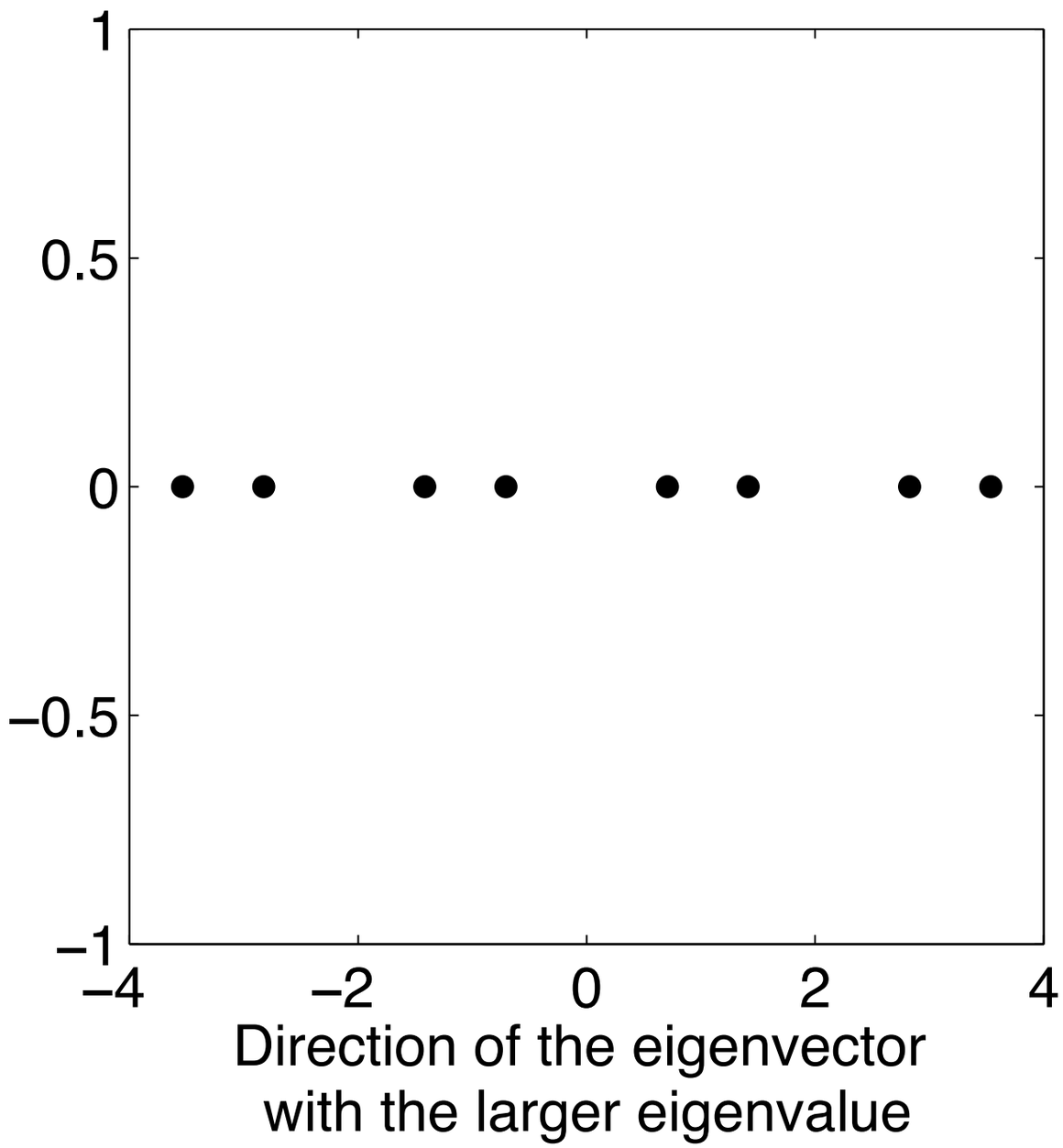
$$F^T = \begin{pmatrix} \sqrt{2}/2 & \sqrt{2}/2 \end{pmatrix}$$

5. Derive the new data:
FinalData=RowFeatureVectorxRowDataAdjust.

RowFeatureVector has the eigenvectors in the rows with the eigenvector with the highest eigenvalue On TOP.
RowDataAdjust contains the zero mean data in each column, with each row holding a separate dimension.

$$D_n = F^T D_c$$

PCA


1
0.5
0
−0.5
−1
−4
−2
0
2
4
Direction of the eigenvector
with the larger eigenvalue

Summary

- Oja’s rule is a hebbian rule with a homeostasis term.
- Homesostasis operates on a different principle to BCM rule.
    - It convergent to a weight vector with length 1.
- Oja’s rule relates to Principle Component Analysis.

Thank you!

Acknowledgements

- This module is an adaptation of the MSc module “Unsupervised and Reinforcement Learning in Neural Networks” by Prof. Wulfram Gerstner at EPFL, Switzerland.

Bibliography

- Neuronal Dynamics, by Gerstner et al.
- Introduction to the Theory of Neural Computation" by Hertz, Krogh & Palmer
- Reinforcement Learning: An Introduction" by Sutton & Barto
- Scholarpedia