Brain-Inspired Computing

Konstantinos Michmizos

Computational Brain Lab

Lecture 14

Thursday, April 4th, 2019

Linear Hebbian Learning

- We have shown that a linear neuron that updates its weights according to a simple Hebbian rule will grow its weight vector along the direction of the eigenvector of the input covariance matric C with maximum eigenvalue (or the **first principal component**).

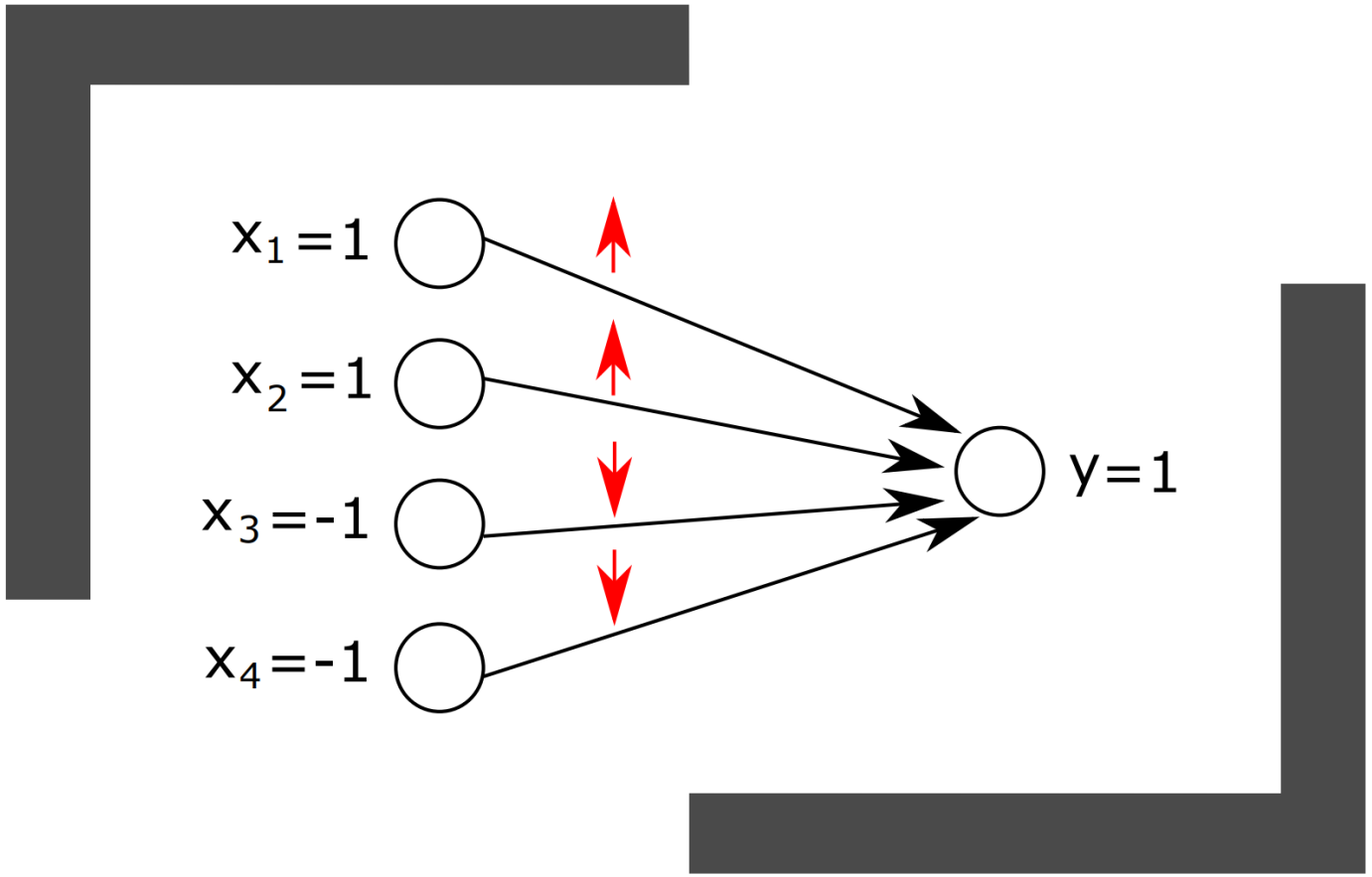
- In other words, w is attempting to move in a direction that captures the most amount of variance in the input distribution, which is the property of the first principal component.

Unsupervised Hebbian Learning

$$w_{ji}(t+1) = w_{ji}(t) + \epsilon\, y_j(t)\, x_i(t)$$

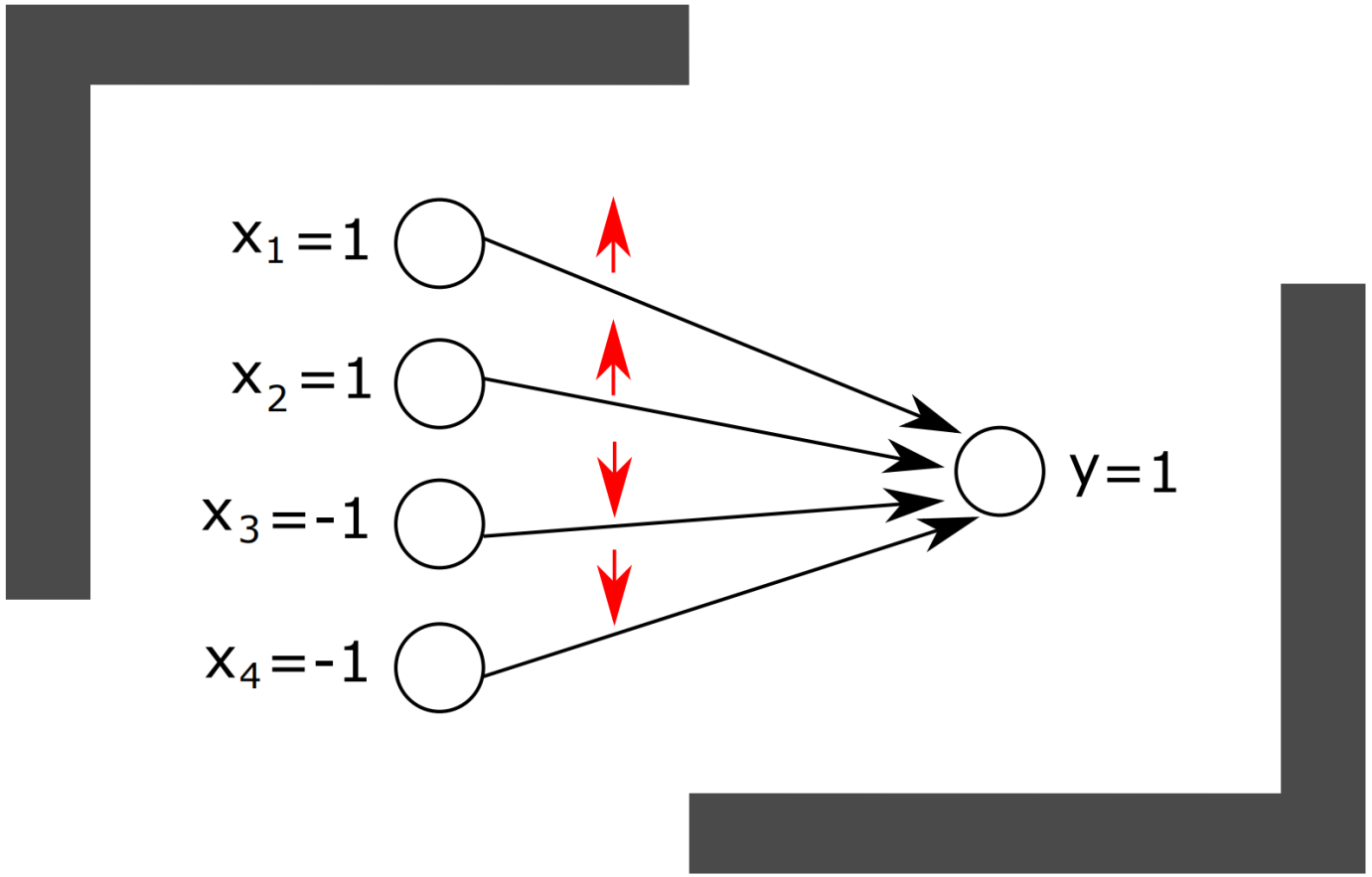
- **What does Unsupervised Hebbian Learning do?**
  - Detects correlations among inputs and outputs
    - *"Neurons that fire together, wire together"*

Unsupervised Hebbian Learning



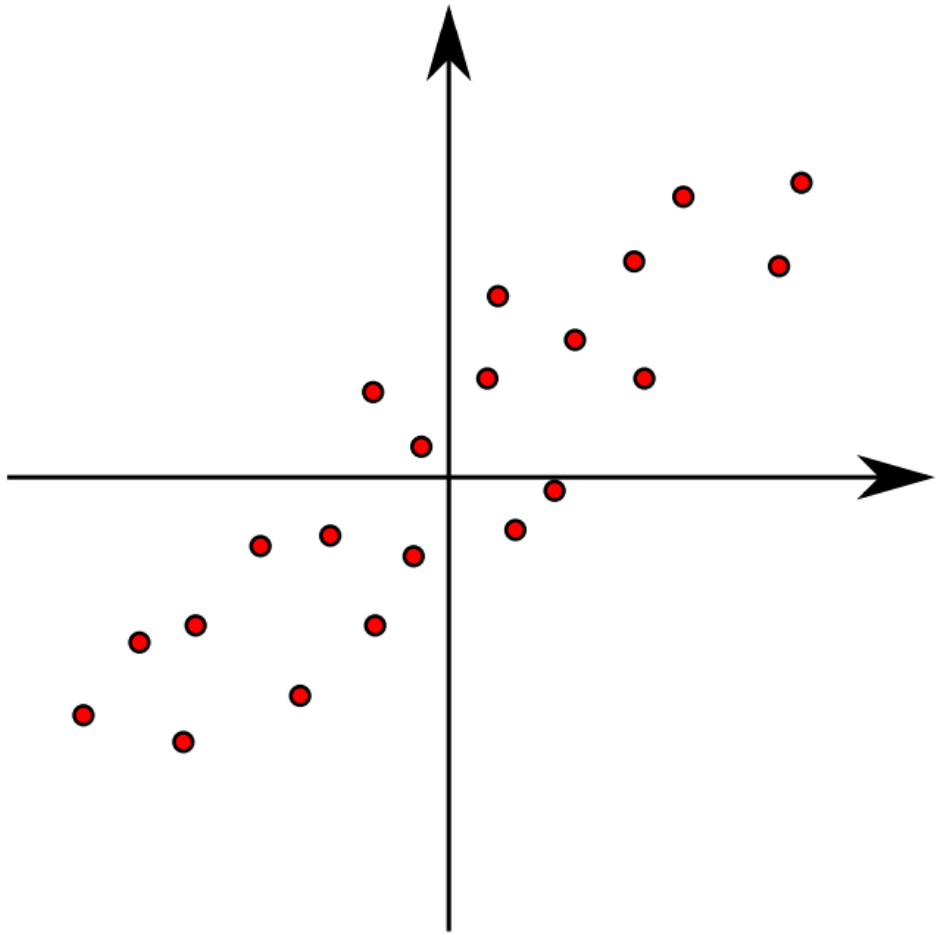
- **What does Unsupervised Hebbian Learning do?**
  - Detects correlations among inputs
    - Input units with positively correlated activations should have positively correlated weights
    - Input units with negatively correlated activations should have negatively correlated weights

Unsupervised Hebbian Learning

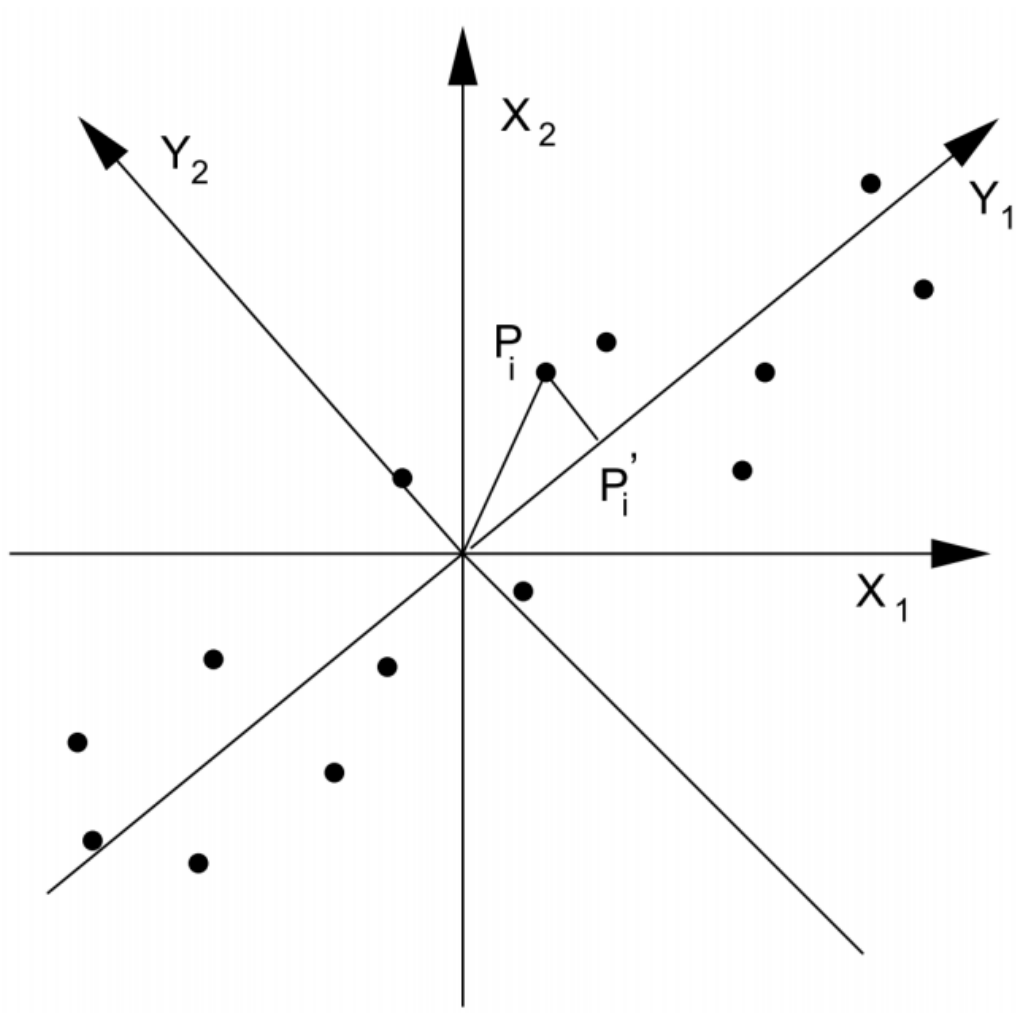


- **What does Unsupervised Hebbian Learning do?**
- Principal Component Analysis
  - Re-represent input data items in a lower dimensional space
  - Do so by finding directions in which data items have maximal variance
  - Ignore directions in which data items have minimal variance (thereby introducing some error in the new representation)

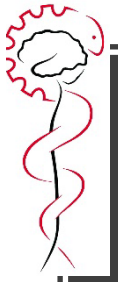
Principal Component Analysis



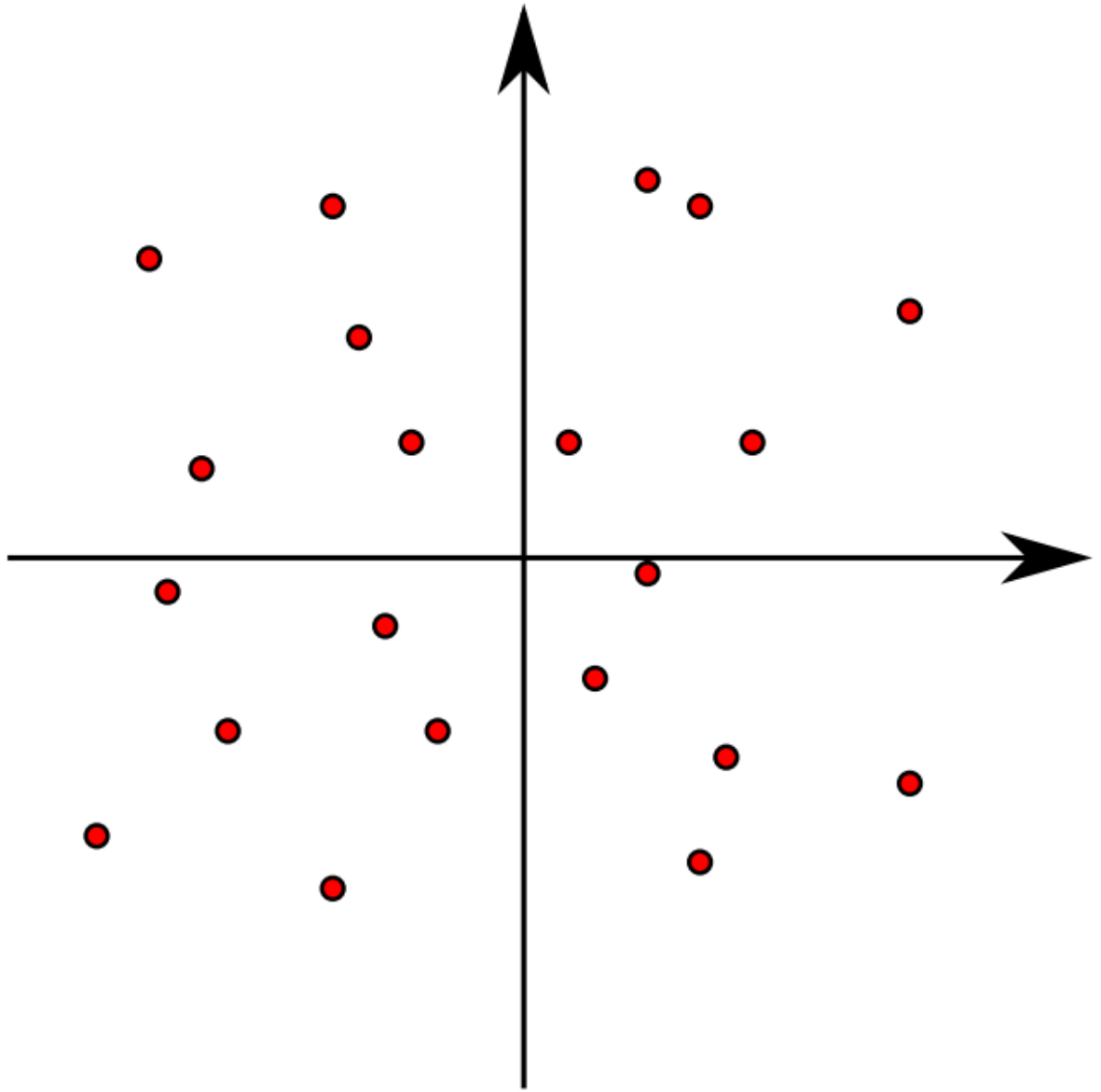
Principal Component Analysis



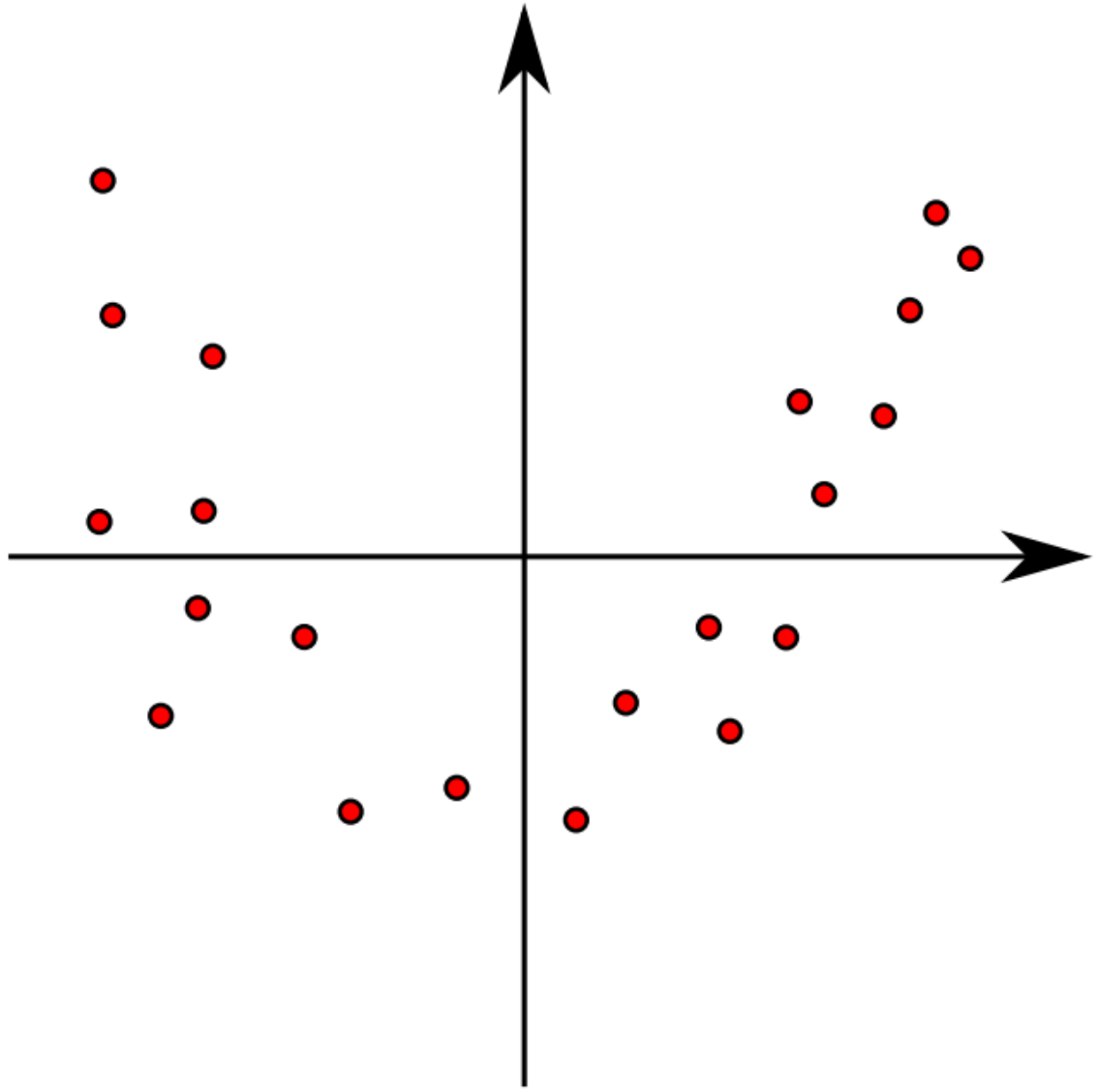
Axes of new representation are a rotation of axes from original representation
(i.e., they are orthogonal to each other)



Would you do dimensionality reduction in this case?



Would you do *linear* dimensionality reduction in this case?



Why choose directions that maximize variance?

- Maximum variance = maximum information
  - E.g., I could assign to all of you {A,B} or {A,B,C,D,…} – Which of the two sets conveys more information?

Linear Hebbian Learning and PCA

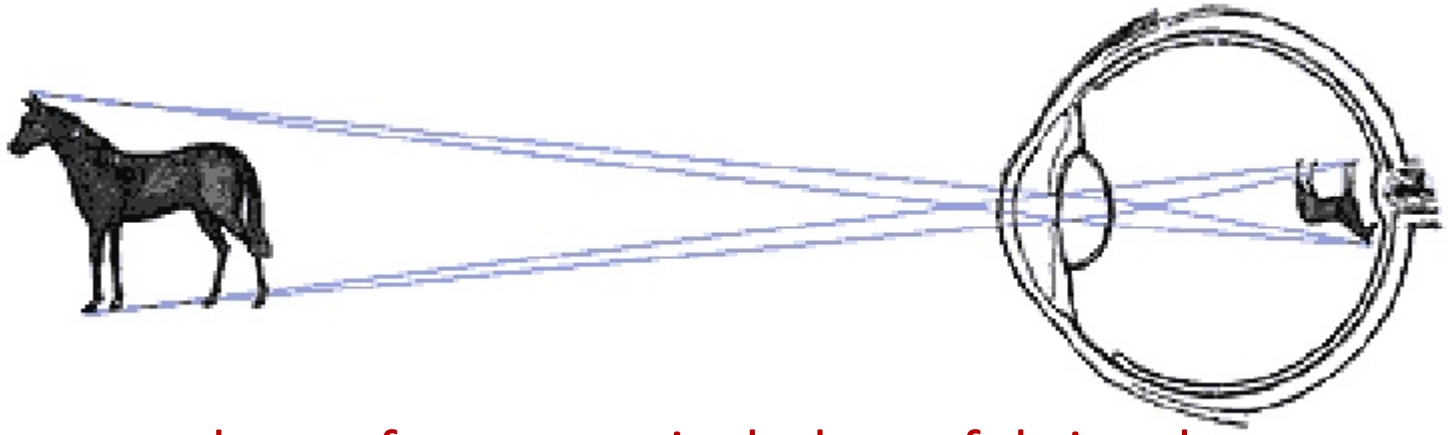
- 1960's Horace Barlow†

Redundancy Reduction: a useful goal of sensory coding is to transform the input in such manner that reduces the redundancy due to complex statistical dependencies among elements of the input stream.

† H. B. Barlow. Possible principles underlying the transformation of sensory messages. *Sensory Communication*, pp. 217–234, 1961

Understanding redundancy reduction

- Consider the process of Image Formation
  - Light reflects off of independent entities (objects)
  - And focused onto an array of photoreceptors in the retina



- The activities of the photoreceptors do not form a particularly useful signal
  - because the structure present in the world is not made explicit, but it is embedded in the form of complex statistical dependencies (redundancies) among photoreceptor activities
- The visual system's goal:
  - Extract these statistical dependencies so that images are explained in terms of a collection of independent events → Such a representation may recover an explicit representation of the underlying independent entities

The simplest form of redundancy

The response properties of neurons at early stages of the visual system can be accounted for in terms of a strategy for reducing the redundancy in natural images.

- examples?

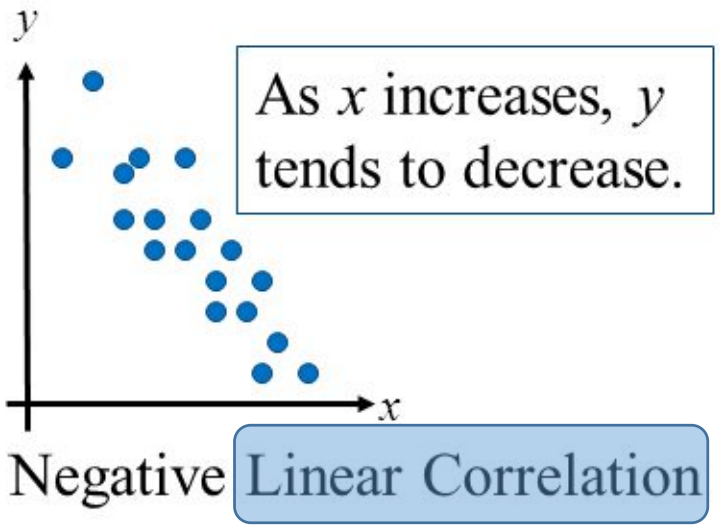
The simplest form of redundancy that may occur in an input stream is linear pairwise correlations

We will show that a network of linear neurons that modifies its weights according to a Hebbian learning rule will reduce the form of redundancy by performing principal component analysis (PCA)

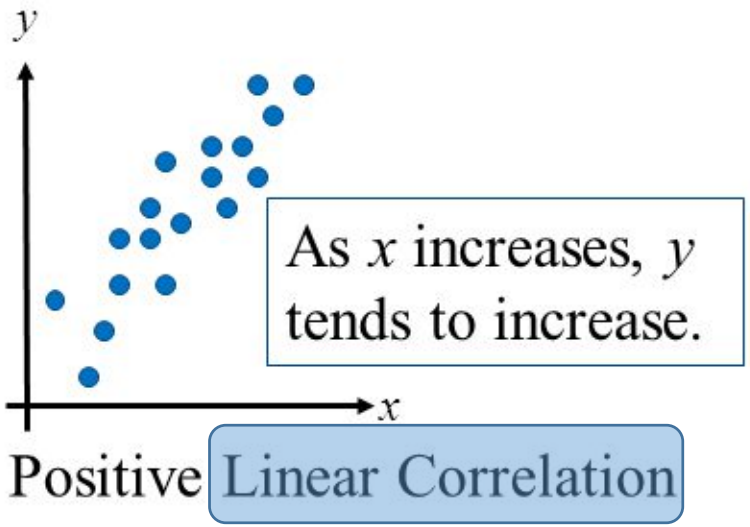
Interpretation of Linear Correlations

- The correlation between two random variables, X and Y, provides a measure of how similar their deviations from the respective means are
- Linear correlation has the property of being bounded between -1 and 1
- Correlation allows to easily understand the strength of the linear dependence between two random variables:
  - the closer correlation is to 1, the stronger the positive linear dependence between X and Y is
  - the closer it is to -1, the stronger the negative linear dependence between X and Y is

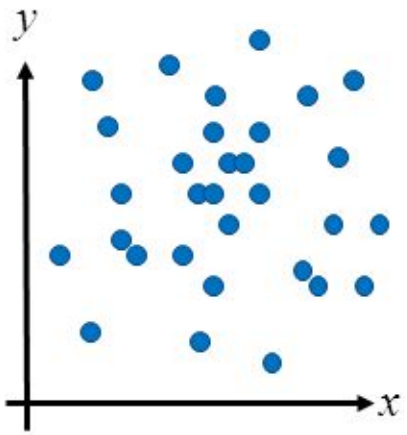
Insight on Linear Correlations



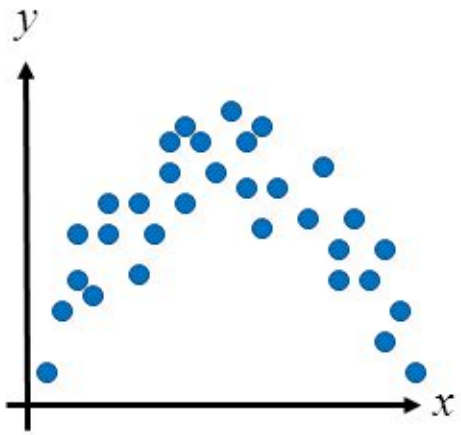
Negative Linear Correlation



Positive Linear Correlation



No Correlation



Nonlinear Correlation

Principal Component Analysis

- Let us consider an input stream x that has linear pairwise correlations among its elements

$$c_{ij} = \langle x_i \, x_j \rangle \neq 0$$

where the brackets < . > mean “average over many input examples”

We assume that all variables have zero mean $\langle x_i \rangle = 0$

So $c_{ij} \neq 0$ implies that there are statistical dependencies among the inputs

Principal Component Analysis

- The goal of PCA is to transform the input x to a new representation, y, in which the variables are pairwise decorrelated

$$y_i = \mathbf{e}_i \cdot \mathbf{x}$$

where $\langle y_i\, y_j \rangle = 0 \;\; \forall_{i \neq j}$

Thus the redundancy due to linear pairwise correlations is eliminated

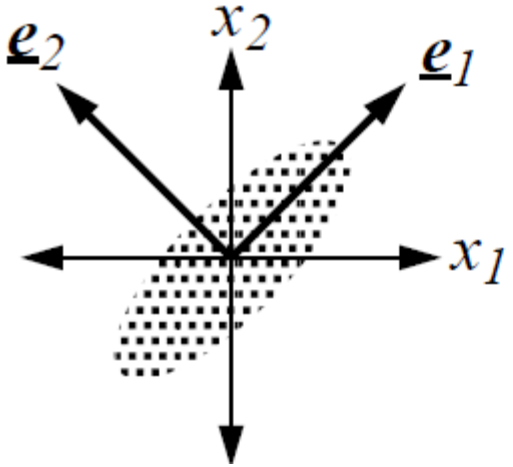
By definition, in PCA the vectors e_i are ordered according to the variance on the y_i such that $\text{var}(y_1) > \text{var}(y_2) > \ldots > \text{var}(y_n)$

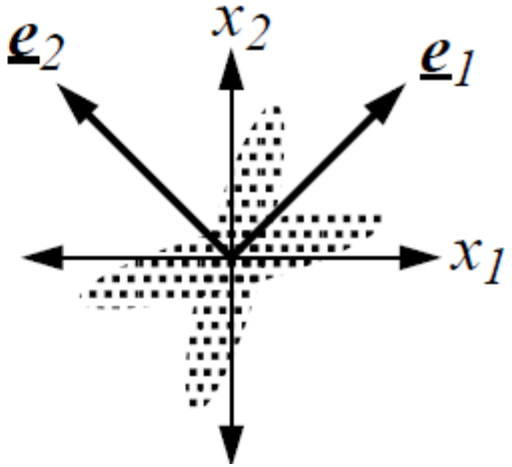
Limitations of PCA

- It takes into account only 2nd order statistics among input variables (correlations among **pairs** of inputs)
  - In real-life, there will be higher-order statistical dependencies among the variables (where PCA is blind to these forms of structure – e.g., $\langle x_i x_j x_k \rangle$)
- The vectors e_i are forced to be orthogonal
  - In real-life, there is no a-priori reason for thinking this is appropriate

- But we can still use PCA to find the principal axes that account for most of the variance in the data

PCA – Geometric Interpretation





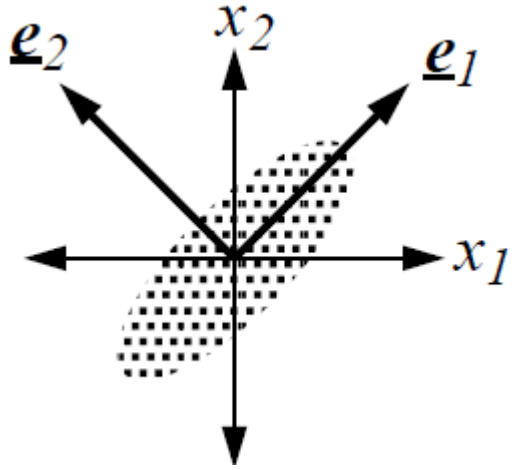
- An underlying assumption of PCA is that the data have Gaussian structure (left)
- If the data are not Gaussian distributed (right) then PCA is not an appropriate strategy for revealing the structure of the input ensemble

The vectors e_i constitute the "principal axes" of the input distribution

- The first principal component e_1 captures the most amount of variance in the input distribution
- The second principal component e_2 captures what is left after subtracting out the component along e_1.

PCA - Recap

- Despite its limitations, PCA can be used to find the
  - Effective dimensionality of an input space
    - i.e., the principal axes that account for most of the variance
  - Independent components *when the data have Gaussian structure*.



Linear Hebbian Learning

Hebb's rule:

"The synaptic weight between two neurons should be increased proportionally to the correlation between the pre-synaptic and post-synaptic activities"

For a linear neuron $$y = \sum_i w_i x_i$$

each weight should be increased proportional to the correlation between y and x_i

$$\dot{w}_i = \langle y\, x_i \rangle$$

Linear Hebbian Learning

$$\dot{w}_i = \langle y\, x_i \rangle$$

$$y = \sum_i w_i x_i$$

Since *y* depends on all the inputs, we can intuitively see that the evolution of w_i depends on the correlation between x_i and all the other inputs

$$\begin{aligned} \dot{w}_i &= \langle \sum_j w_j x_j\, x_i \rangle \\ &= \sum_j w_j \langle x_j\, x_i \rangle \end{aligned}$$

- Why we can move w_i outside of the ensemble average?
  - The w's are changing over a much slower time-scale than the x's

Linear Hebbian Learning

$$\begin{aligned} \dot{w}_i &= \langle \sum_j w_j x_j \, x_i \rangle \\ &= \sum_j w_j \langle x_j \, x_i \rangle \end{aligned}$$

can be written as $\dot{\mathbf{w}} = \mathbf{C}\mathbf{w}$

where $c_{ij} = \langle x_i \, x_j \rangle$

The growth of w depends solely on the input covariance matrix C

In other words, the evolution of w is governed by the linear pairwise statistics of the input ensemble

Linear Hebbian Learning – 1D system

1 weight, w

- To get a better idea of how w evolves, we examine a simple 1-D system of the form

$$\dot{w} = c\,w$$

which is just a linear, 1st order differential equation, with a solution:

$$w(t) = w(0)\,e^{c\,t}$$

where w(0) is the initial weight state at time zero.

Linear Hebbian Learning – 1D system

1 weight, w

$$w(t) = w(0)\, e^{c\, t}$$

- If $c > 0$, then w will grow exponentially
- If $c < 0$, then w will decay exponentially
- Constant c determines how (fast) w grows or decays

Linear Hebbian Learning – 2D system

2 weights, w_1 w_2

$$\dot{w} = c\,w$$

Let's examine a slightly more complex system consisting of two weights, w_1 and w_2

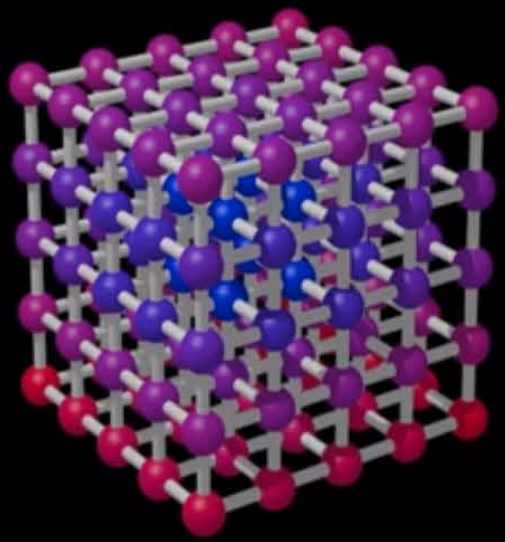
$$\begin{aligned} \dot{w}_1 &= c_{11}w_1 + c_{12}w_2 \\ \dot{w}_2 &= c_{21}w_1 + c_{22}w_2 \end{aligned}$$

This is just a 2-D version of the above equation, written out explicitly
in terms of the vector and matrix components.

It is difficult to "see" a simple solution here, **because the two d.e. are "coupled**:
The evolution of w_1 depends on w_2 and likewise, the evolution of w_2 depends on w_1

Let's try to transform the w_i to a new coordinate system in which the variables will be de-coupled

$$\mathbf{A}\vec{v} = \lambda\vec{v}$$



X — $n \times m$ matrix

n: samples

m: measurements

Principal Components Analysis

eigen decomposition of $(X^T X)$ → W, λ

$(X^T X)$: $m \times m$ matrix

$$T = X W$$

T: "scores" ($n \times m$)

X: ($n \times m$)

W: "loadings" ($m \times m$)

each col of W is a "Principal Component"

→ ORDERED columns by value of λ

$$T_r = X W_r$$

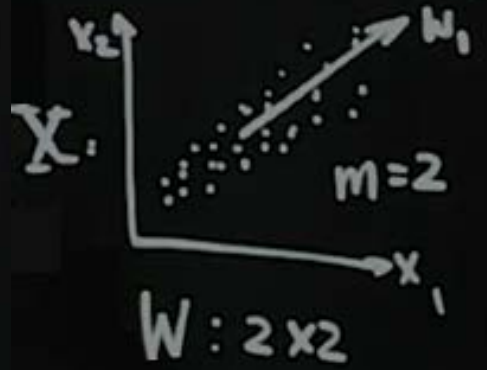
T_r: $n \times r$

W_r: $m \times r$

W_r = first r columns of W

$$\begin{bmatrix} | & | & & | \\ w_1 & w_2 & \cdots & w_n \\ | & | & & | \end{bmatrix}$$

r



W: 2×2

$X W_1$

W_1

Aside: Eigenvectors and Eigenvalues

Matrix Diagonilization

- If C is symmetric, then we can re-write it in the form

$$\mathbf{C} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$$

Which is in our case

$$c_{ij} = \langle x_i x_j \rangle = \langle x_j x_i \rangle = c_{ji}$$

$$\mathbf{E} = \begin{bmatrix} | & | & & | \\ \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \\ | & | & & | \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$$

$$\mathbf{e}_i \perp \mathbf{e}_i \ \forall_{i \neq j}$$

$$|\mathbf{e}_i| = 1 \ \forall_i$$

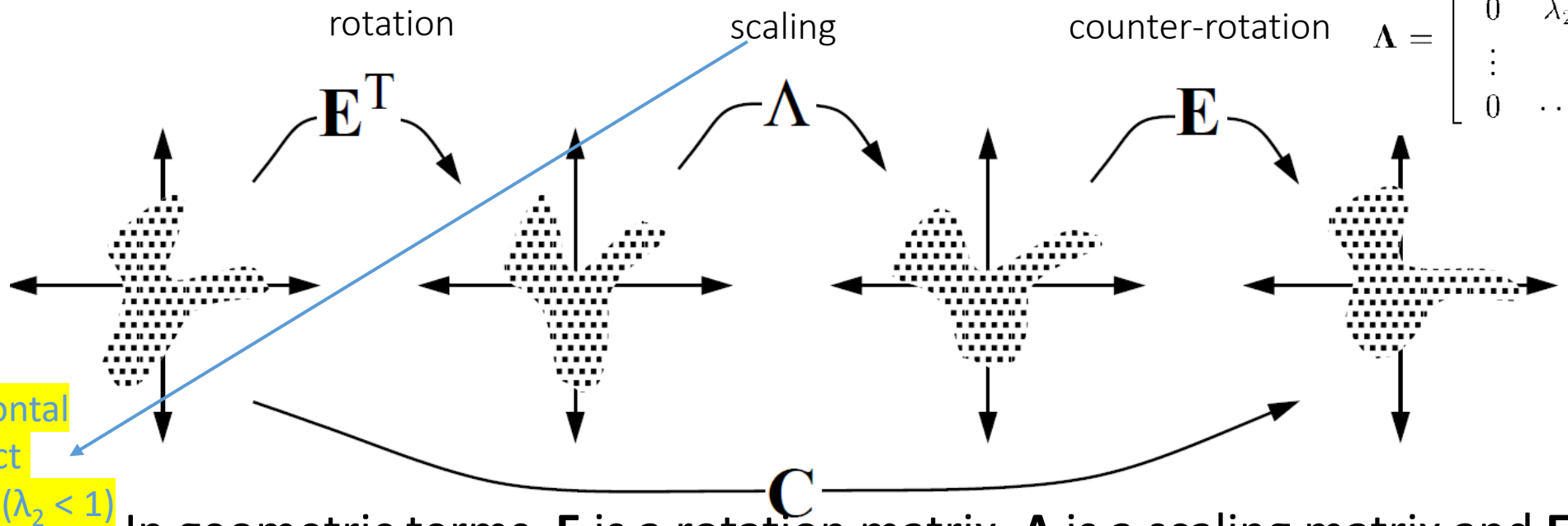
$$(\mathbf{E}^T)_{ij} = (\mathbf{E})_{ji}$$

Aside: Eigenvectors and Eigenvalues

$$\mathbf{E} = \begin{bmatrix} | & | & & | \\ \mathbf{e}_1 & \mathbf{e}_2 & \cdots & \mathbf{e}_n \\ | & | & & | \end{bmatrix}$$

$$\mathbf{\Lambda} = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & 0 & \lambda_n \end{bmatrix}$$

$$\mathbf{C} = \mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$$



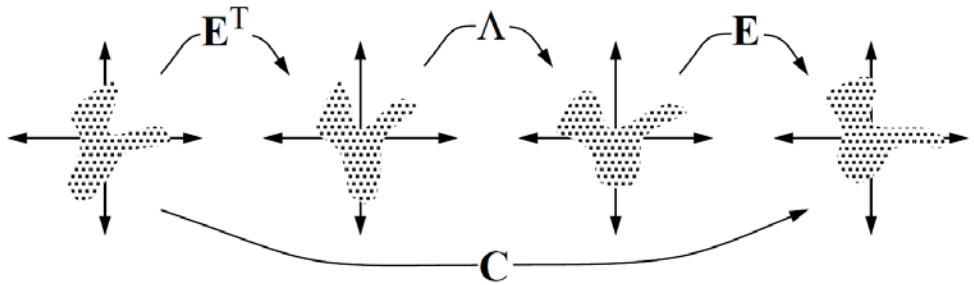
Dilate along the horizontal axis (λ_1>1) and contract along the vertical axis (λ_2 < 1)

In geometric terms, **E** is a rotation matrix, **Λ** is a scaling matrix and $\mathbf{E}^T$ is another rotation matrix that rotates in the opposite direction of **E**.

Multiplying a point or vector by the matrix **C** can be thought of as first rotating to another coordinate frame (multiplying by $\mathbf{E}^T$), then scaling each axis according to λ_i within this new coordinate frame (multiplying by **Λ**), and then rotating back to the original coordinate frame (multiplying by **E**).

Linear Hebbian Learning – 2D system

$$\dot{w} = c\,w$$



$$\begin{aligned} \dot{w}_1 &= c_{11}w_1 + c_{12}w_2 \\ \dot{w}_2 &= c_{21}w_1 + c_{22}w_2 \end{aligned}$$

Let us transform **w** to a new coordinate system by multiplying by $\mathbf{E}^T$

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \mathbf{E}^T\mathbf{w}$$

where: v_1 is the projection of w along the first eigenvector, $\boldsymbol{e}_1$

v_2 is the projection of w along the second eigenvector $\boldsymbol{e}_2$

Linear Hebbian Learning – 2D system

$$\mathbf{v} = \begin{bmatrix} v_1 \\ v_2 \end{bmatrix} = \mathbf{E}^T \mathbf{w}$$

$$\dot{\mathbf{v}} = \mathbf{\Lambda} \mathbf{v}$$

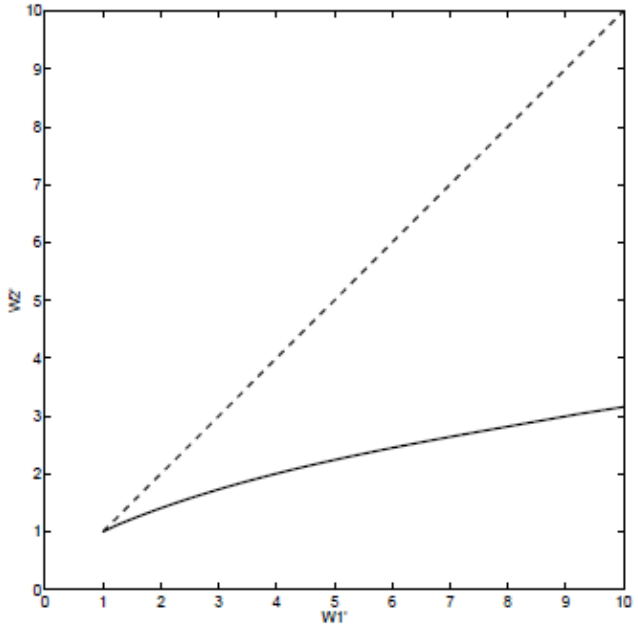
$$\begin{aligned} \dot{v}_1 &= \lambda_1 v_1 \\ \dot{v}_2 &= \lambda_2 v_2 \end{aligned}$$

$$\begin{aligned} v_1(t) &= v_1(0)\, e^{\lambda_1 t} \\ v_2(t) &= v_2(0)\, e^{\lambda_2 t} \end{aligned}$$

That is why we need the eigenvalues – to be able to solve for each parameter, separately

Hebbian Learning – 2D system

$$v_1(t) = v_1(0)\, e^{\lambda_1 t}$$
$$v_2(t) = v_2(0)\, e^{\lambda_2 t}$$

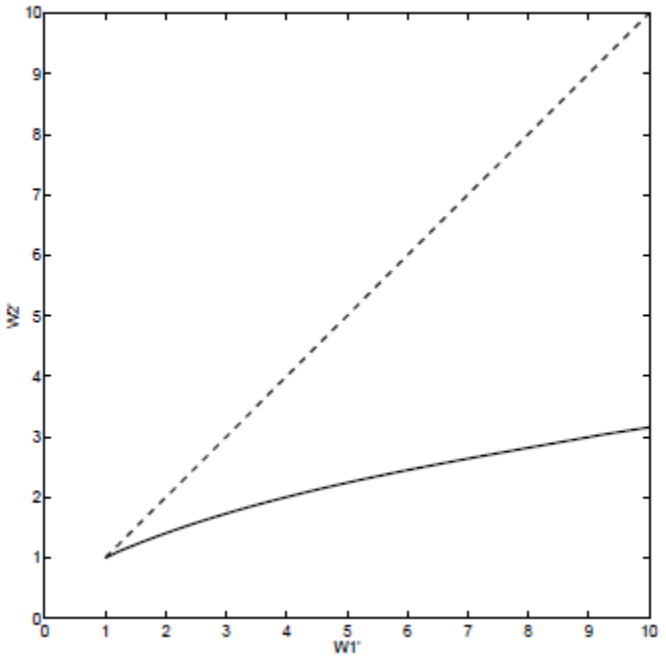


Because λ1 and λ2 represent exponential growth rates, even a slight imbalance between the two will result in one rapidly outpacing the other.

If $\lambda_1 > \lambda_2$, then **v** will grow in the direction [1,0], which in terms of our original reference frame is just in the direction of $\mathbf{e}_1$

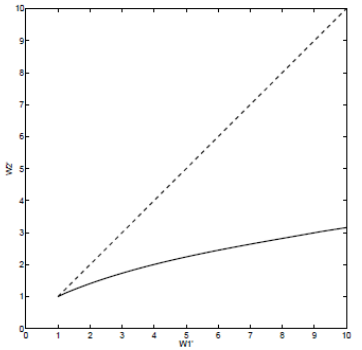
If $\lambda_2 > \lambda_1$, then **v** will grow in the direction [0,1], which in terms of our original reference frame is in the direction of $\mathbf{e}_2$.

Hebbian Learning – 2D system



- The evolution of **v**
- In this case $\lambda_1=2$ and $\lambda_2=1$, so the growth of v_1 rapidly outpaces v_2.
- Since the v_1 axis is simply the same as the vector e_1 in the original coordinate system, then w will grow in the direction of e_1
- The dotted line shows the expected evolution of v if v_1 and v_2 were to grow at equal rates

Hebbian Learning – Intuition 2.0

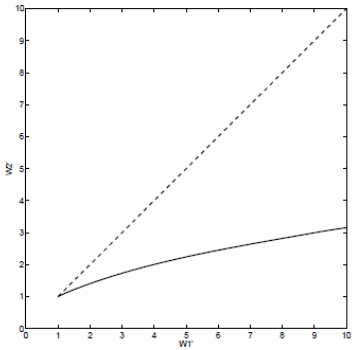


- For correlation or covariance matrices, all the eigenvalues, are real and non-negative, and we order them from largest to smallest.

- The exponential factors all grow over time, because the eigenvalues λ are positive for all μ values (the number of eigenvectors).

- For large t, **the term with the largest value of λμ** (assuming it is unique) **becomes much larger** than any of the other terms and **dominates the sum for w**

Hebbian Learning – Intuition 2.0



- This largest eigenvalue has the label μ = 1, and its corresponding eigenvector e1 is called the principal eigenvector

- After training, the response to an arbitrary input vector u is well-approximated by

$$v \propto \mathbf{e}_1 \cdot \mathbf{u}$$

- Because the dot product corresponds to a projection of one vector onto another, Hebbian plasticity can be interpreted as producing an output proportional to the projection of the input vector onto the principal eigenvector of the correlation matrix of the inputs used during training.

Hebbian Learning – Intuition 2.0

- The proportionality sign hides the large exponential factor which is a result of the positive feedback inherent in Hebbian plasticity.

- We have seen how we can impose saturation limits (haven't we?)

Disclaimer:

- If weight growth is limited by some form of multiplicative normalization, as in the Oja rule, the configuration of synaptic weights produced by Hebbian modification will typically be proportional to the principal eigenvector of the input correlation matrix.
- When subtractive normalization is used and the principal eigenvector is proportional to n, the eigenvector with the next largest eigenvalue provides an estimate of the configuration of final weights, again up to a proportionality factor.
- If, however, saturation constraints are used, as they must be in a subtractive scheme, this can invalidate the results of a simplified analysis based solely on these eigenvectors

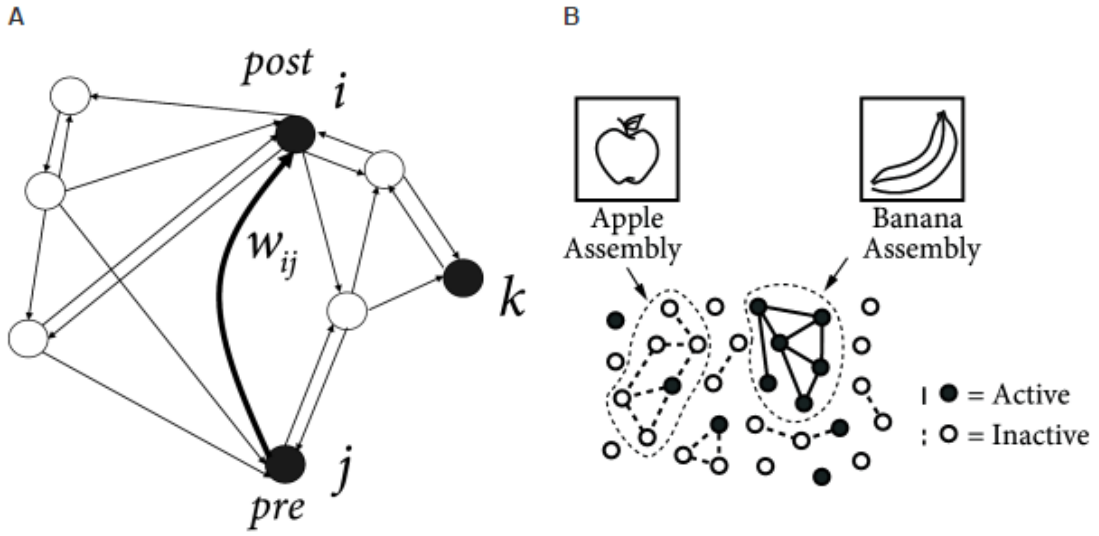
Hebb rule: Correlation based Learning

When an axon of cell A is near enough to excite cell B and repeatedly or persistently takes part in firing it, some growth process or metabolic change takes place in one or both cells such that A ’s efficiency, as one of the cells firing B, is increased.

Modifications of the synaptic transmission efficacy are driven by correlations in the firing activity of pre- and postsynaptic neurons

Neurons that ‘fire together, wire together’

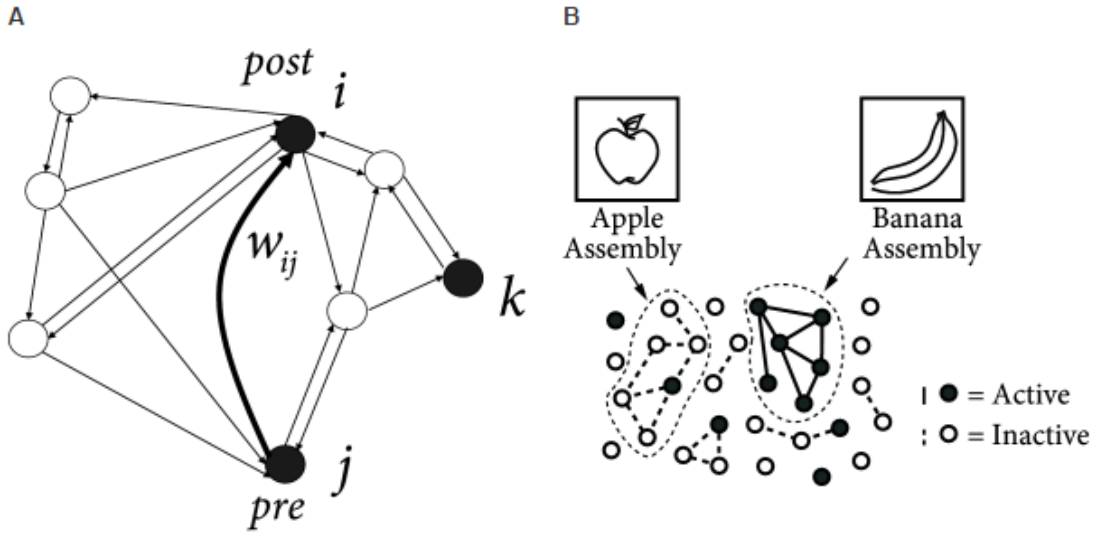
Correlation-based learning is now generally called *Hebbian learning*



Hebbian learning. **A**. The change of a synaptic weight w_{ij} depends on the state of the presynaptic neuron j and the postsynaptic neuron i and the present efficacy w_{ij}, but not on the state of other neurons k. B. Hebbian learning strengthens the connectivity within assemblies of neurons that fire together, e.g. during the perception of banana. Schematic figure.

Hebb rule: Correlation based Learning

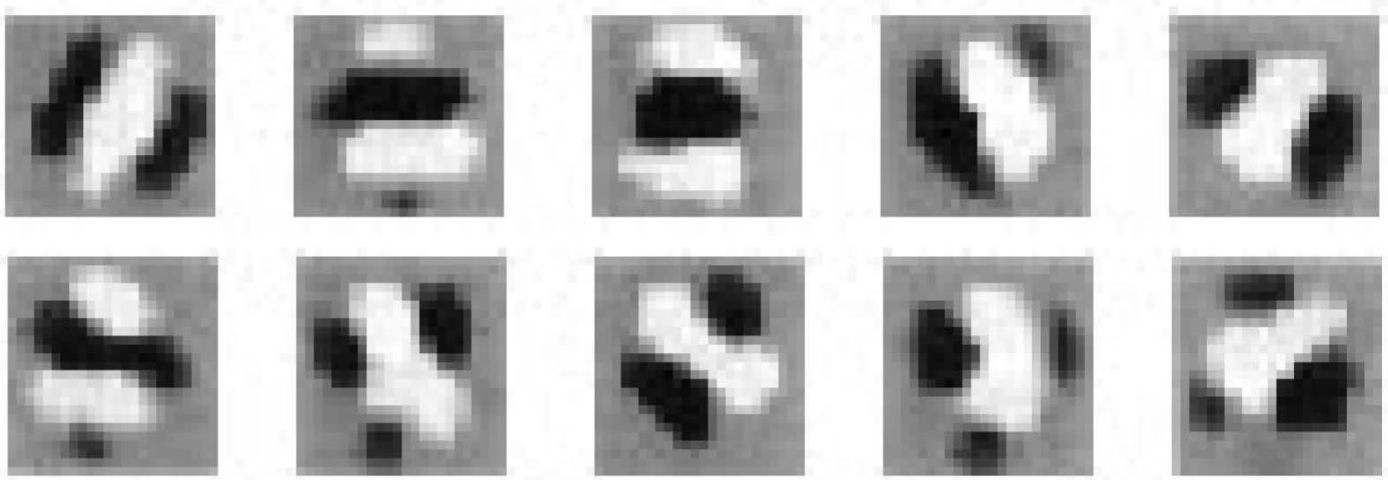
- Hebbian learning is unsupervised
  - There is no notion of ‘good’ or ‘bad’ changes of a synapse
- Synaptic changes happen whenever there is joint activity of pre- and postsynaptic neurons
  - Firing patterns may reflect sensory stimulation or ongoing brain activity
  - There is no feedback signal from a ‘supervisor’ or from the environment



Hebbian learning. **A**. The change of a synaptic weight w_{ij} depends on the state of the presynaptic neuron j and the postsynaptic neuron i and the present efficacy w_{ij}, but not on the state of other neurons k. **B**. Hebbian learning strengthens the connectivity within assemblies of neurons that fire together, e.g. during the perception of banana. Schematic figure.

Hebbian Development of Orientation Selectivity

- The necessary pattern of LGN inputs can arise from Hebbian plasticity on the basis of correlations between the responses of different LGN cells and competition between ON and OFF units.
- Such a model can be constructed by considering a simple cell receiving input from ON-center and OFF-center cells of the LGN and applying Hebbian plasticity, subject to appropriate constraints, to the feedforward weights of the model.



Different cortical receptive fields arising from a correlation-based developmental model. White and black regions correspond to areas in the visual field where ON-center cells (white regions) or OFF-center (black regions) LGN cells excite the cortical neuron being modeled. (Adapted from Miller, 1994.)

Hebbian Development of Orientation Selectivity

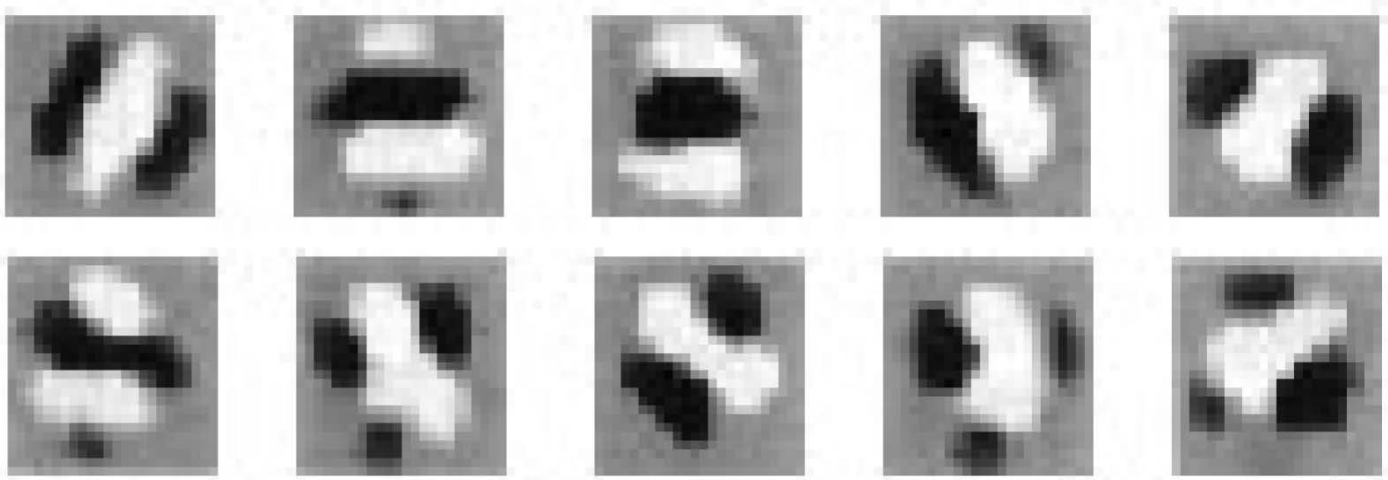
Neurons in primary visual cortex only receive afferents from LGN cells centered over a finite region of the visual space.

This anatomical constraint can be included in developmental models by introducing what is arbor function called an arbor function.

The arbor function, which is often taken to be Gaussian, characterizes the density of innervation from different visual locations to the cell being modeled.

As a simplification, this density is not altered during the Hebbian developmental process, but that the strengths of synapses within the arbor are modified by the Hebbian rule.

The outcome is oriented receptive fields of a limited spatial extent



Different cortical receptive fields arising from a correlation-based developmental model. White and black regions correspond to areas in the visual field where ON-center cells (white regions) or OFF-center (black regions) LGN cells excite the cortical neuron being modeled. (Adapted from Miller, 1994.)

Linear Hebbian Learning

- We have shown that a linear neuron that updates its weights according to a simple Hebbian rule will grow its weight vector along the direction of the eigenvector of the input covariance matric C with maximum eigenvalue (or the **first principal component**).

- In other words, w is attempting to move in a direction that captures the most amount of variance in the input distribution, which is the property of the first principal component.