CSE/NEUBEH 528

Lecture 12: Unsupervised Learning

(Chapters 8 & 10)

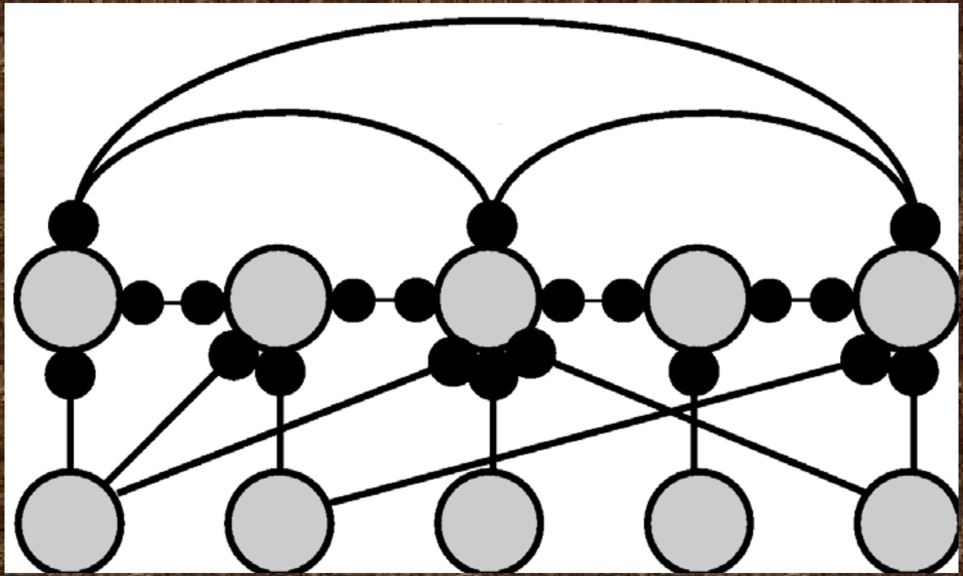
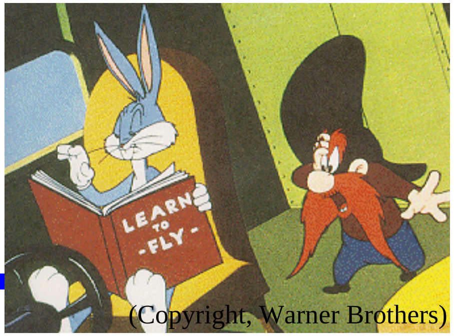


Image from http://clasdean.la.asu.edu/news/images/ubep2001/neuron3.jpg

Lecture figures are from Dayan & Abbott's book
http://people.brandeis.edu/~abbott/book/index.html

Gameplan for Today



(Copyright, Warner Brothers)

- Unsupervised (Representational) Learning
  - Temporally asymmetric learning
  - Hebb rule and Principal Component Analysis (PCA)
  - Causal Models
  - Generative versus Recognition Models
  - Density Estimation and EM
  - Sparse Coding
  - Independent Component Analysis (ICA)
  - Predictive Coding

Flashback: Hebb Rule

- "If neuron A frequently contributes to the firing of neuron B, then the synapse from A to B should be strengthened"
- Consider a linear neuron:

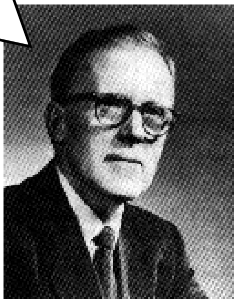
$$v = \mathbf{w}^T \mathbf{u} = \mathbf{u}^T \mathbf{w}$$

- Basic Hebb Rule:

$$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{u} v$$

Waittaminute…what did Hebb really say?

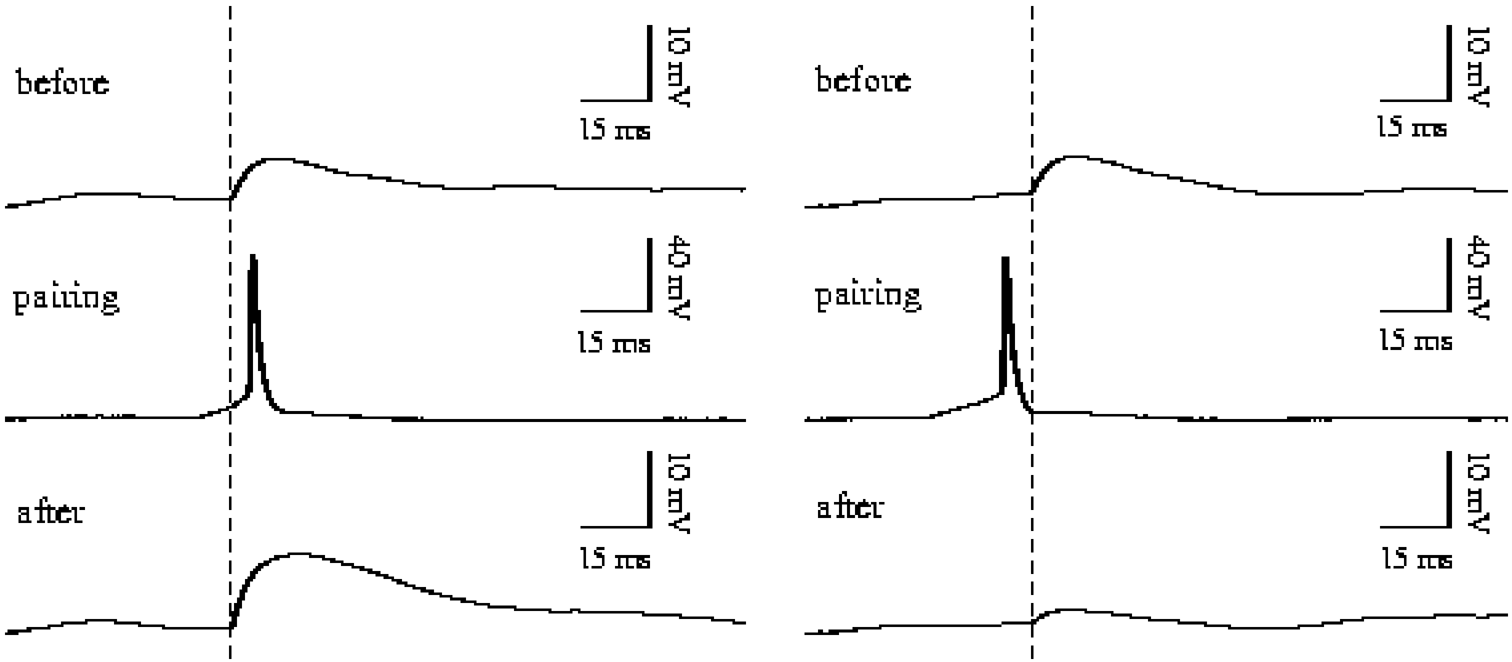
If neuron A frequently **contributes** to the firing of neuron B, then the synapse from A to B should be strengthened



Causality (order of input/output) is important, not just correlation

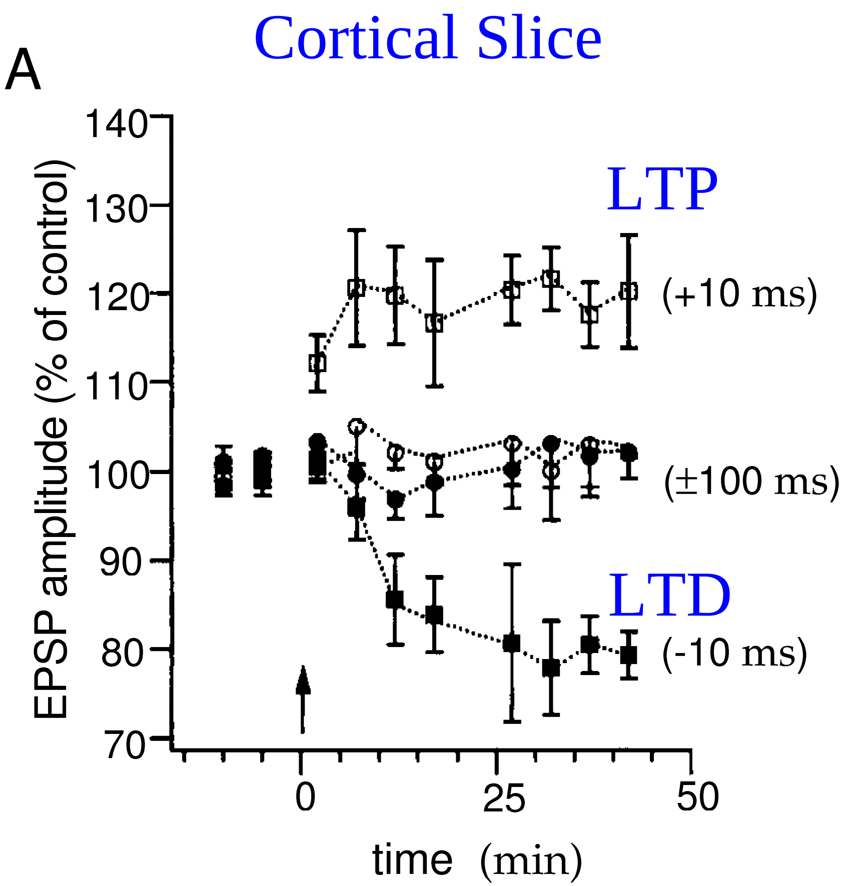
Evidence for Causal Learning Rules: Spike-Timing Dependent Synaptic Plasticity (STDP)

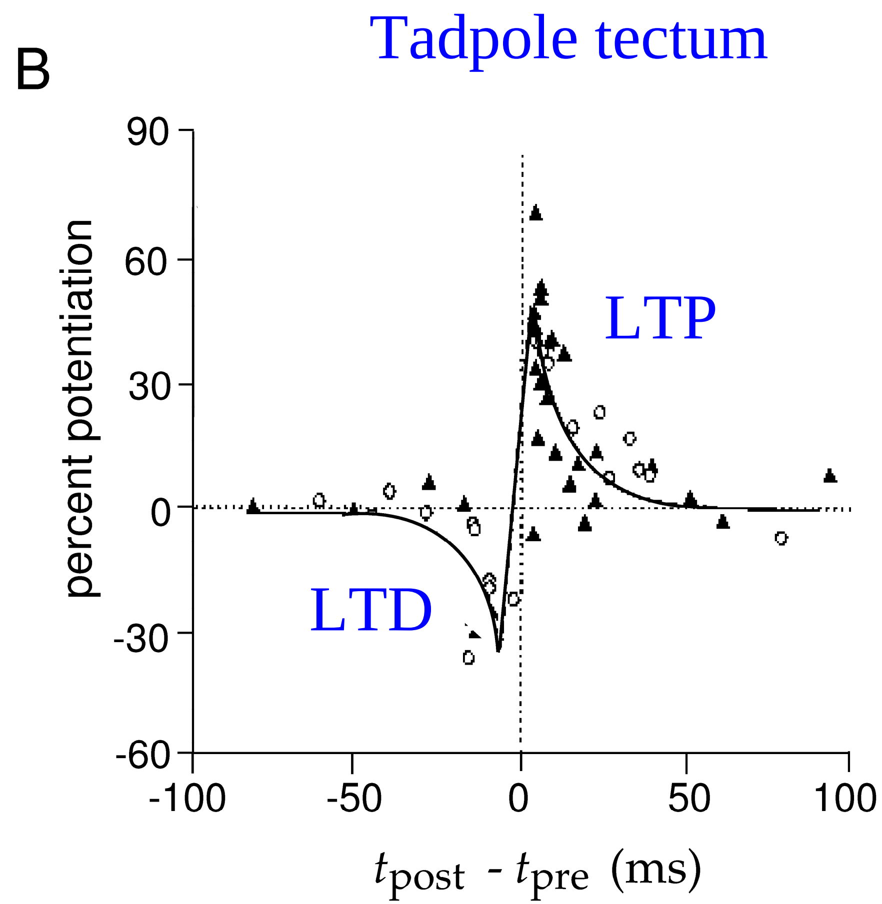
Input before Output Spike — Input after Output Spike



(Note: This is just a simulation I did a while back, not real data!)

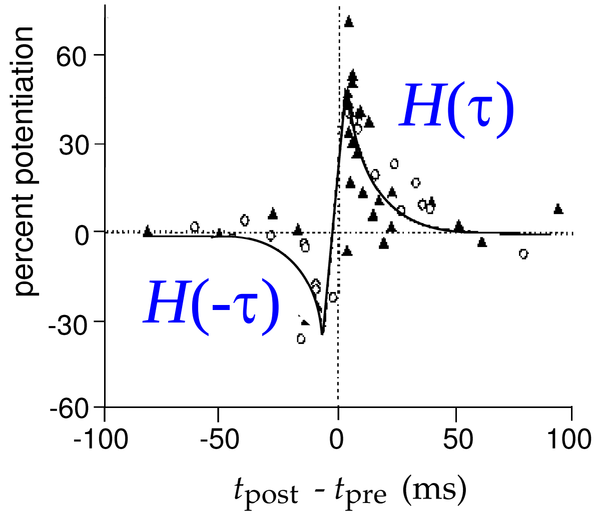
STDP in the Vertebrate Brain





(This is real data!)

Temporally Asymmetric Hebb Rule (STDP)

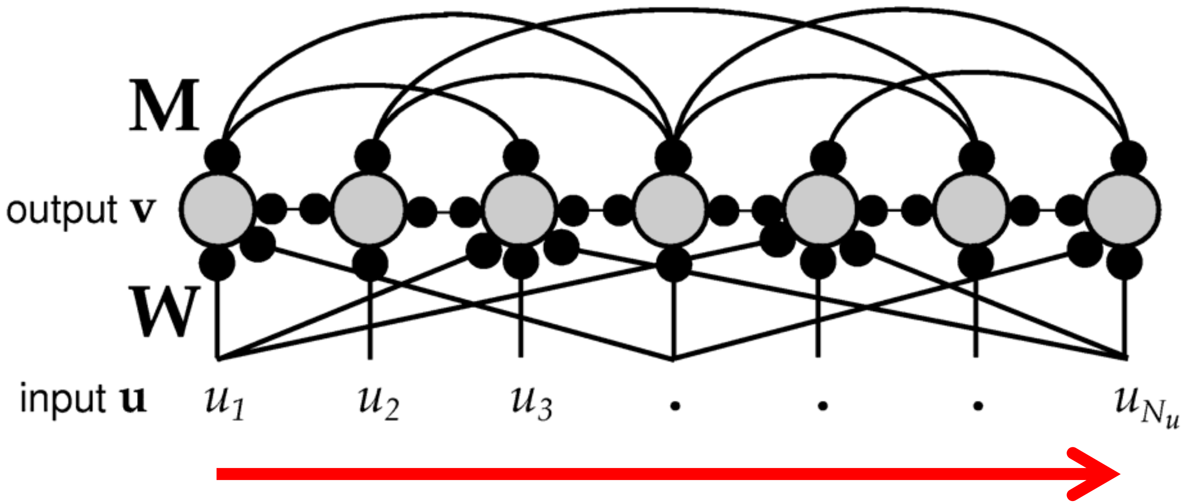


LTP LTD

$$\tau_w \frac{d\mathbf{w}}{dt} = \int_0^\infty \left[H(\tau)\mathbf{u}(t-\tau)v(t) + H(-\tau)\mathbf{u}(t)v(t-\tau) \right] d\tau$$

Past inputs Past outputs

What does STDP do in a Recurrent Network?

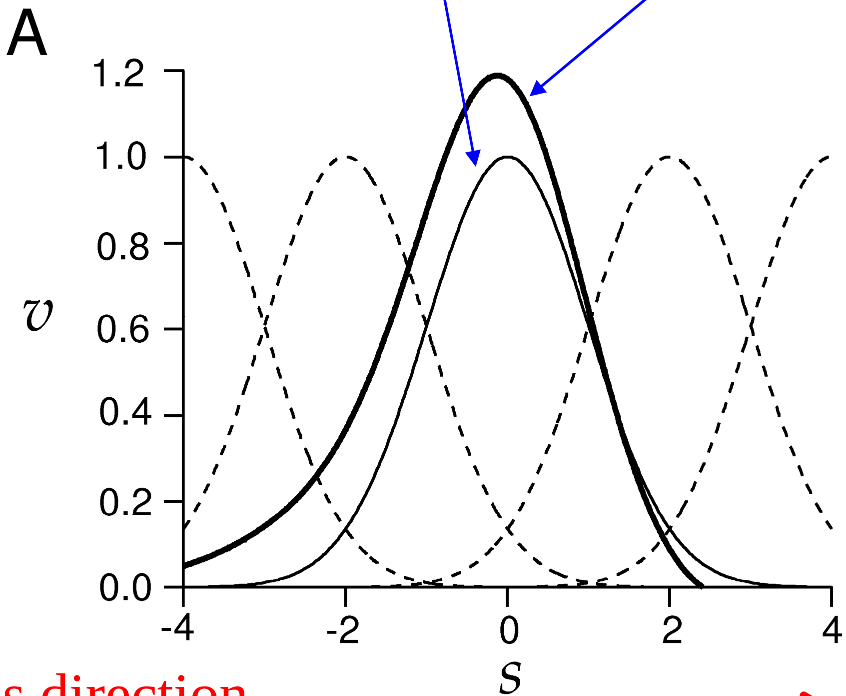


Adapt M using STDP, keep W fixed

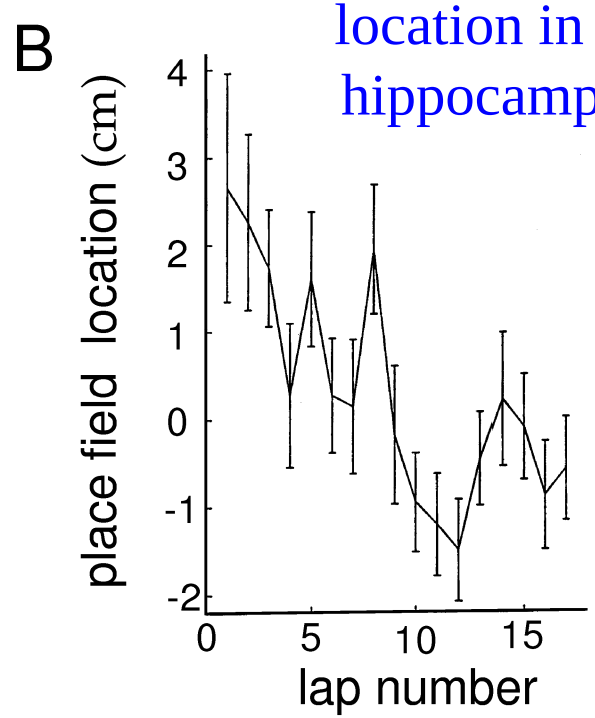
Direction of input sequence ($u_1 \rightarrow u_2 \rightarrow \ldots$)

STDP allows prediction in the navigating rat

Tuning curve before and after



Shift in place field location in rat hippocampus



Rat's direction of motion

Tuning curve shifts, generating anticipatory response

Back to traditional Hebb rule:

$$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{u}v$$

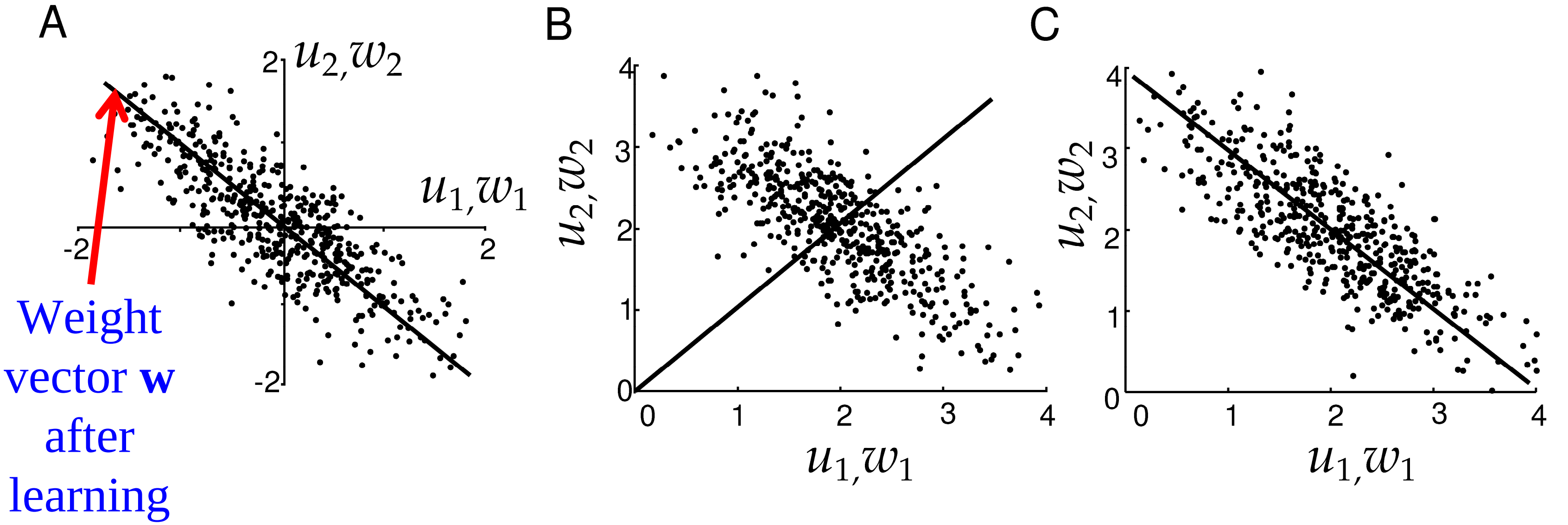
What does this do?

(Flashback from last time:

Eigenvector analysis shows that…)

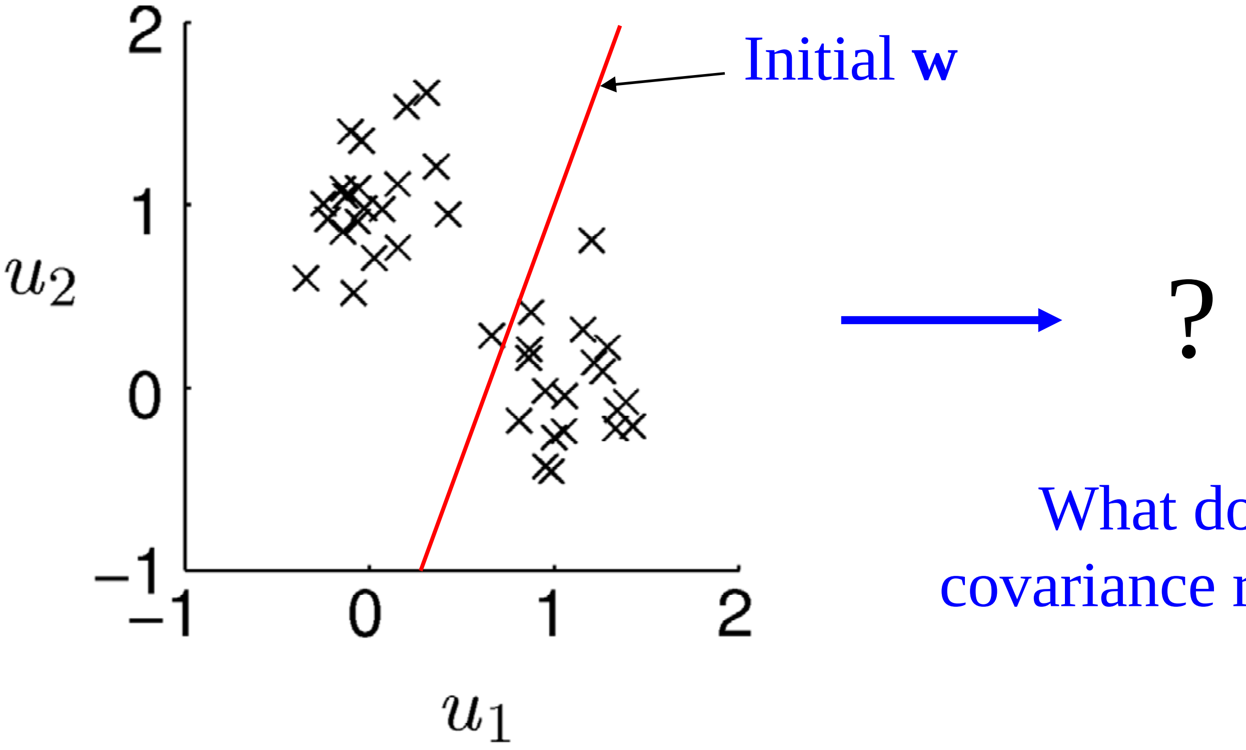
Hebb Rule implements PCA!

Pure Hebb	Pure Hebb	Covariance Rule
Input mean = (0,0)	Input mean = (2,2)	Input mean = (2,2)



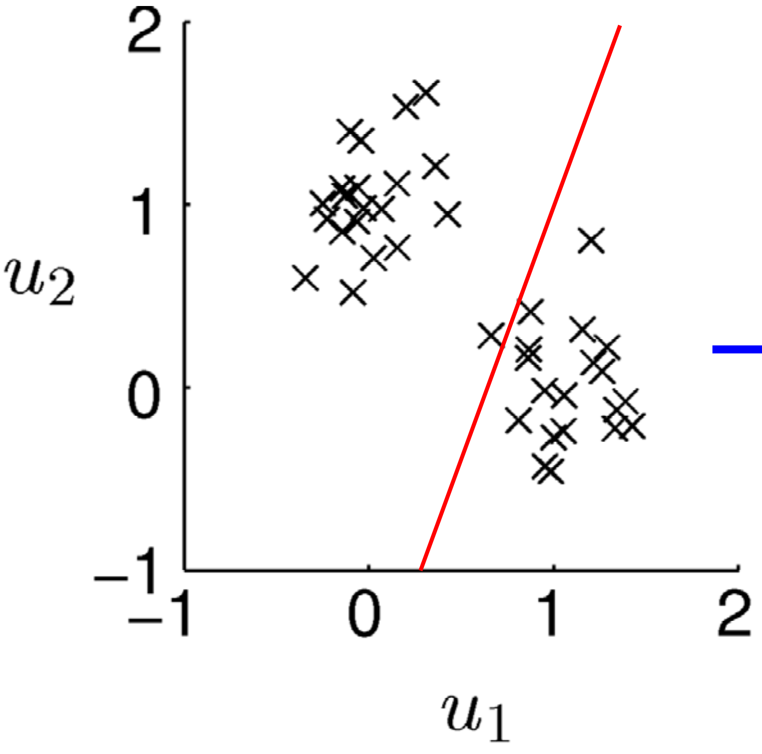
Hebb rule *rotates* weight vector to align with principal eigenvector of input correlation/covariance matrix (i.e. direction of maximum variance)

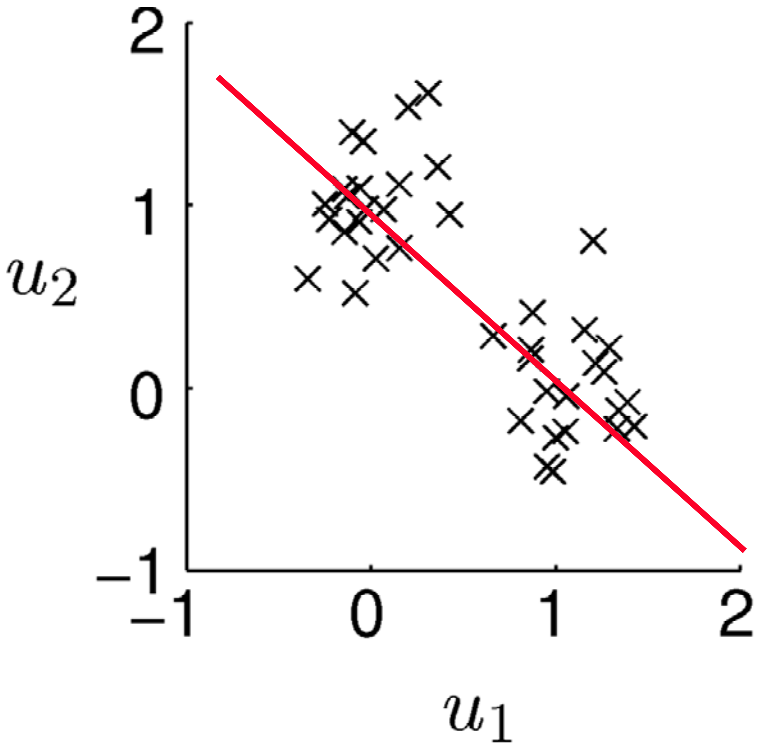
What about this data?



What does the covariance rule learn?

PCA does not correctly describe the data

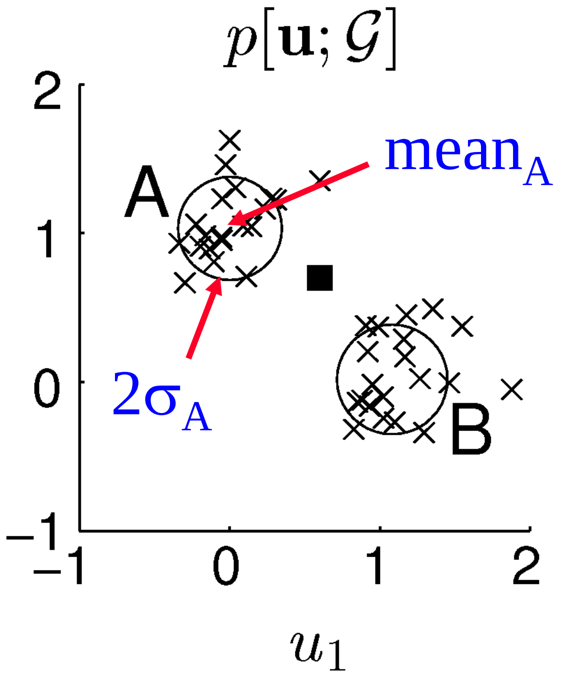




Input data is made up of two clusters (Gaussians) → two "causes"

Causal Models

- Main goal of unsupervised learning: Learn the "Causes" underlying the input data
- Example: Learn the means and variances of the two Gaussians A and B that generated this data
- Want: Two neurons A and B that learn the means and variances based solely on input data (samples from distribution)



Generative versus Recognition Models

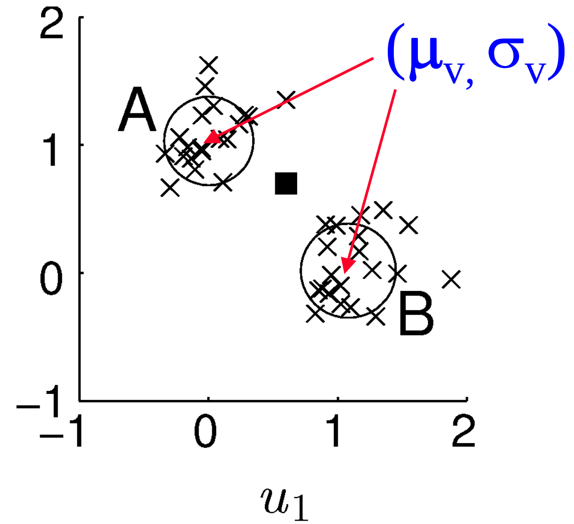
Causes $\mathbf{v}$ $p[\mathbf{v} \mid \mathbf{u}; G]$ (posterior)

Generative model

Recognition model

Data $\mathbf{u}$ $p[\mathbf{u} \mid \mathbf{v}; G]$ (likelihood)

(similar to decoding vs. encoding)

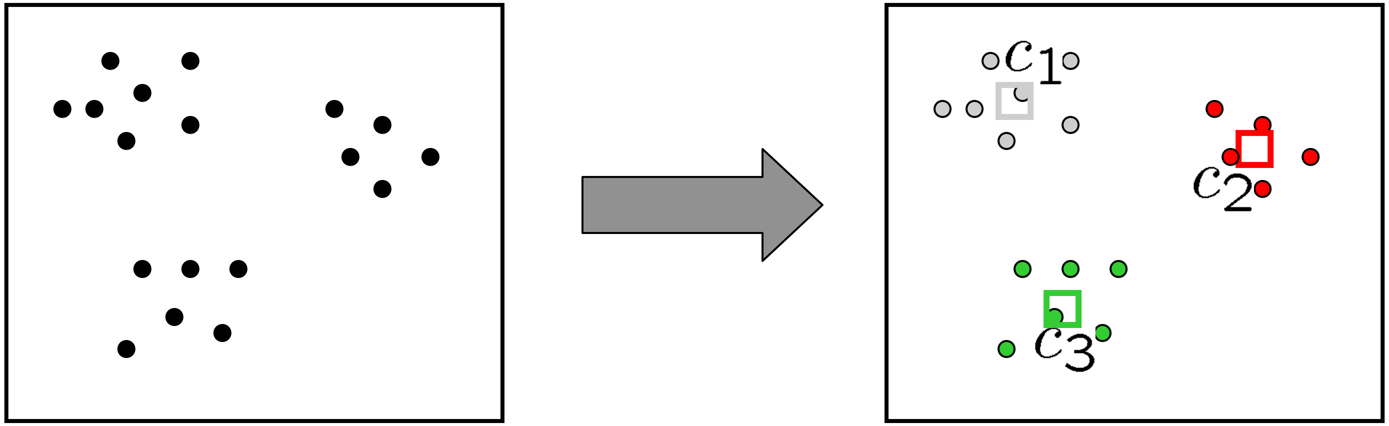


$$p[\mathbf{u}; G] = \sum_{\mathbf{v}} p[\mathbf{u} \mid \mathbf{v}; G] p[\mathbf{v}; G]$$

Prior $= \gamma_v$

Parameters $G = (\mu_v, \sigma_v, \gamma_v)$

How do we learn the parameters (e.g., mean)?



Idea: Each neuron represents one cluster

Minimize sum squared distance of each point to closest cluster center

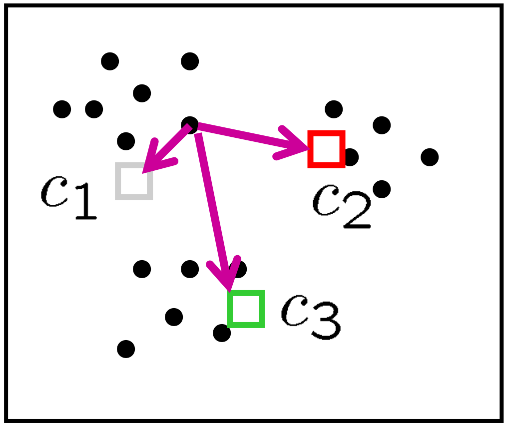
$$\sum_{\text{clusters } i} \quad \sum_{\text{points p in cluster } i} \|p - c_i\|^2$$

Break it down into 2 subproblems

Suppose you are given the cluster centers c_i

Q: how do you assign points to a cluster?

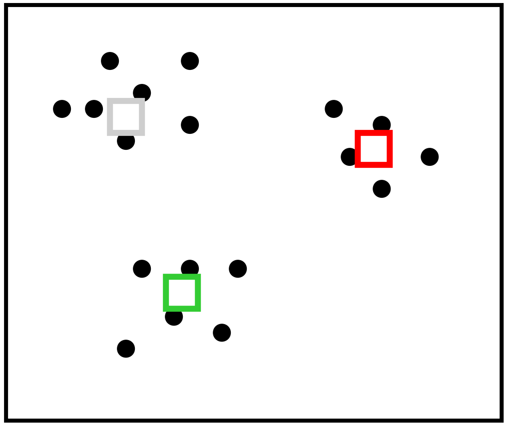
A: for each point p, choose closest c_i



Suppose you are given the points in each cluster

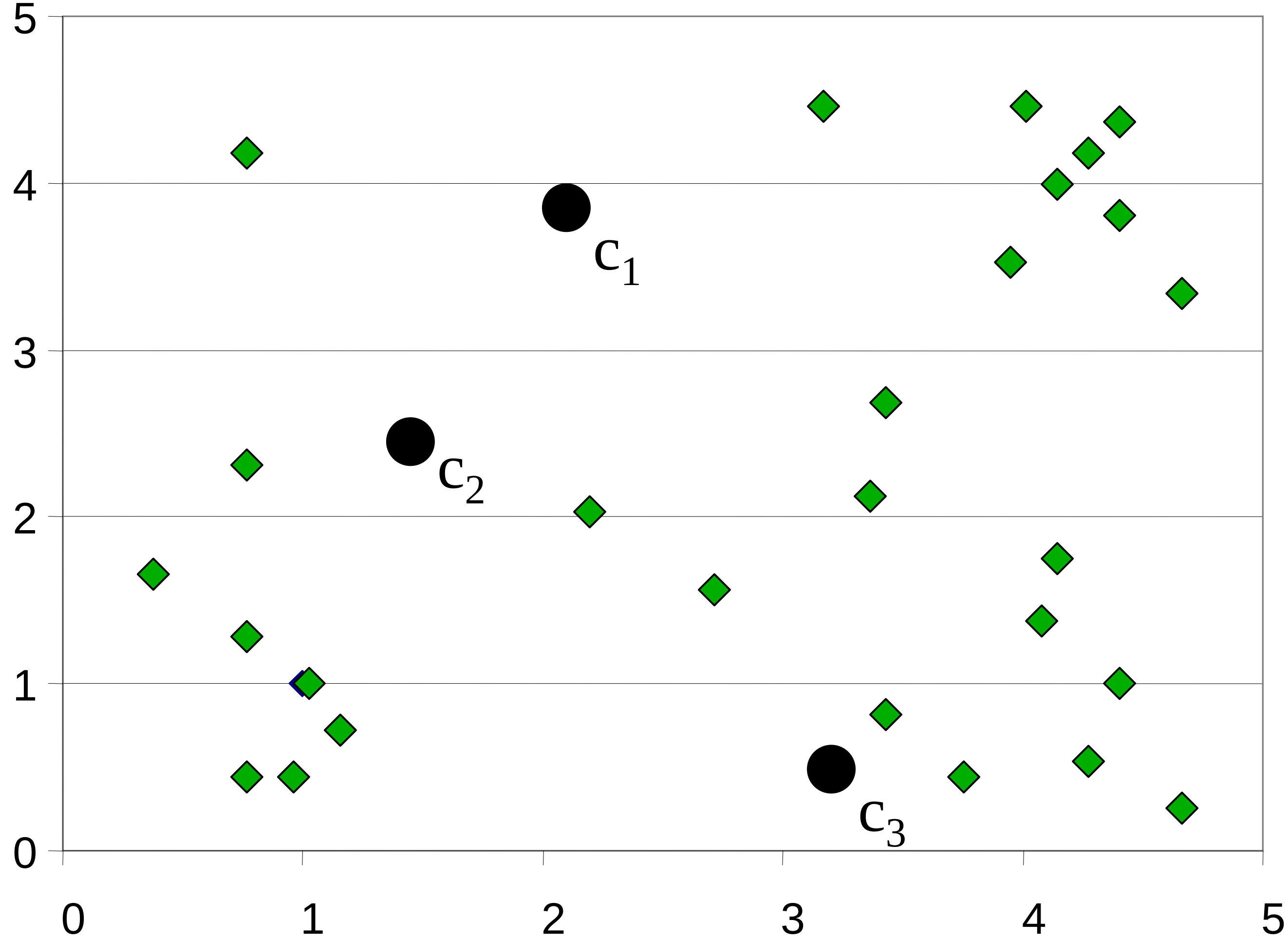
Q: how to re-compute each cluster's center?

A: choose c_i to be the mean of all the points in that cluster



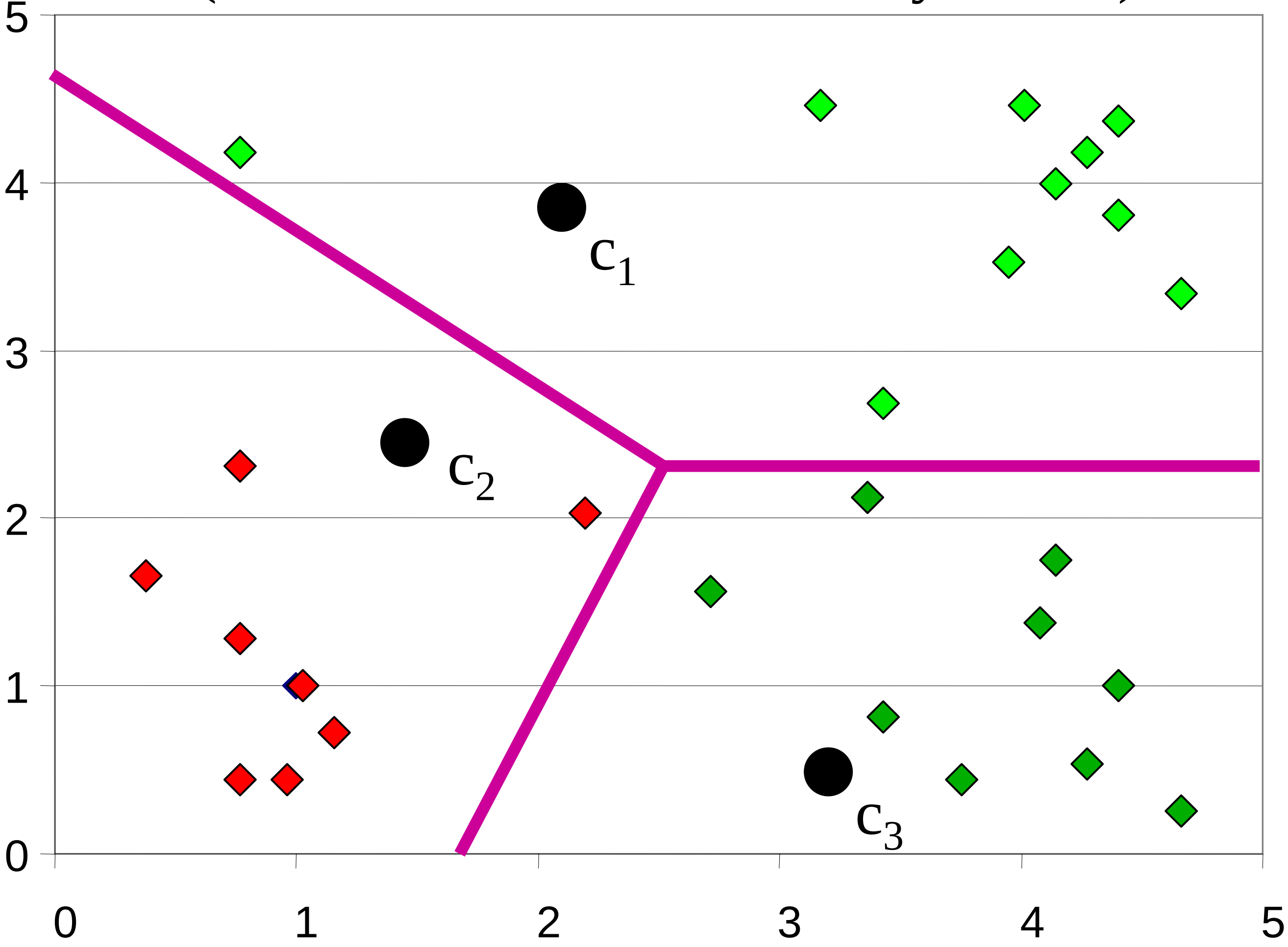
"K-means" clustering: Example

Randomly initialize the cluster centers (synaptic weights)



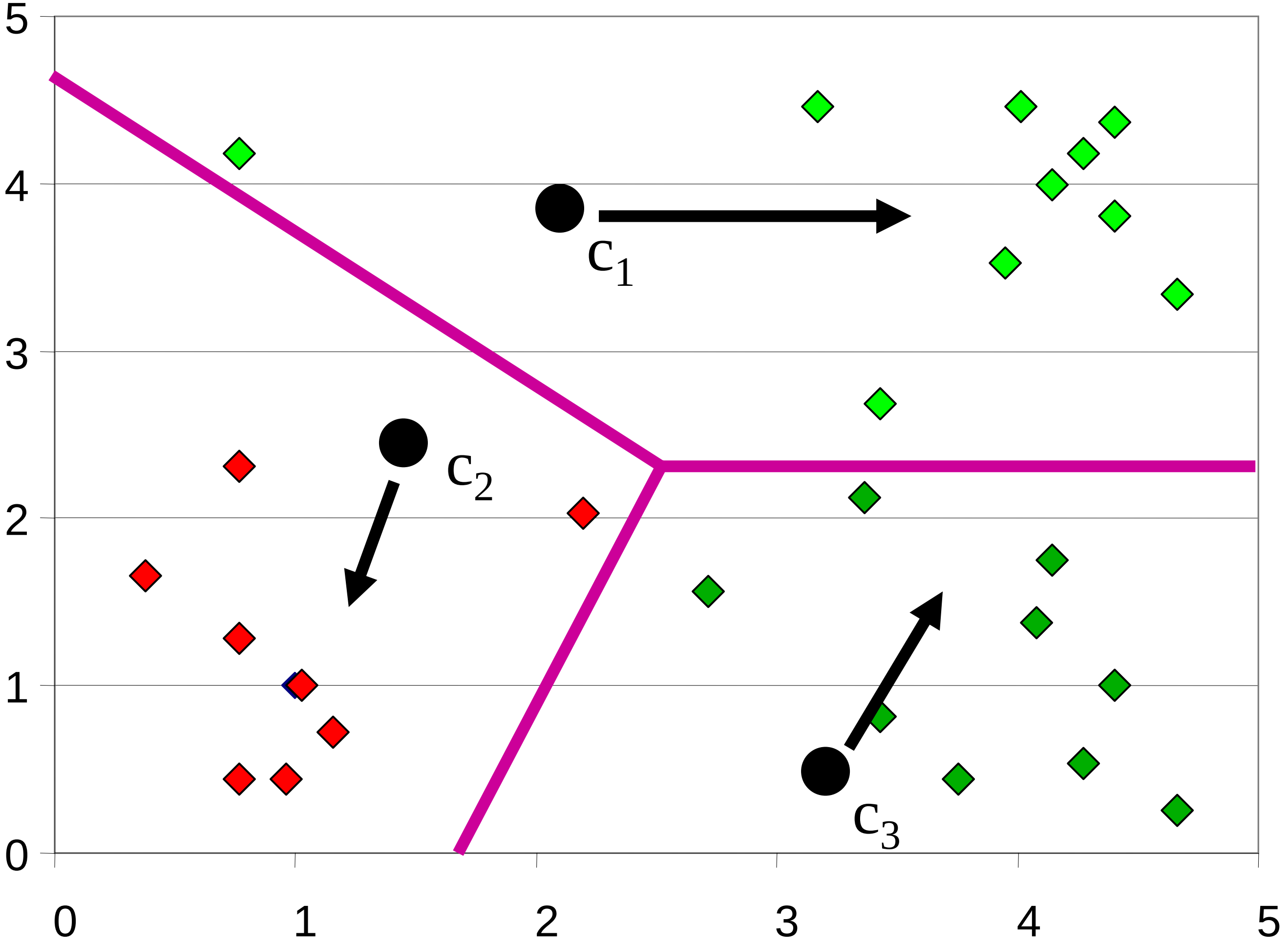
"K-means" clustering: Example

Determine cluster membership for each input
("winner-takes-all" inhibitory circuit)

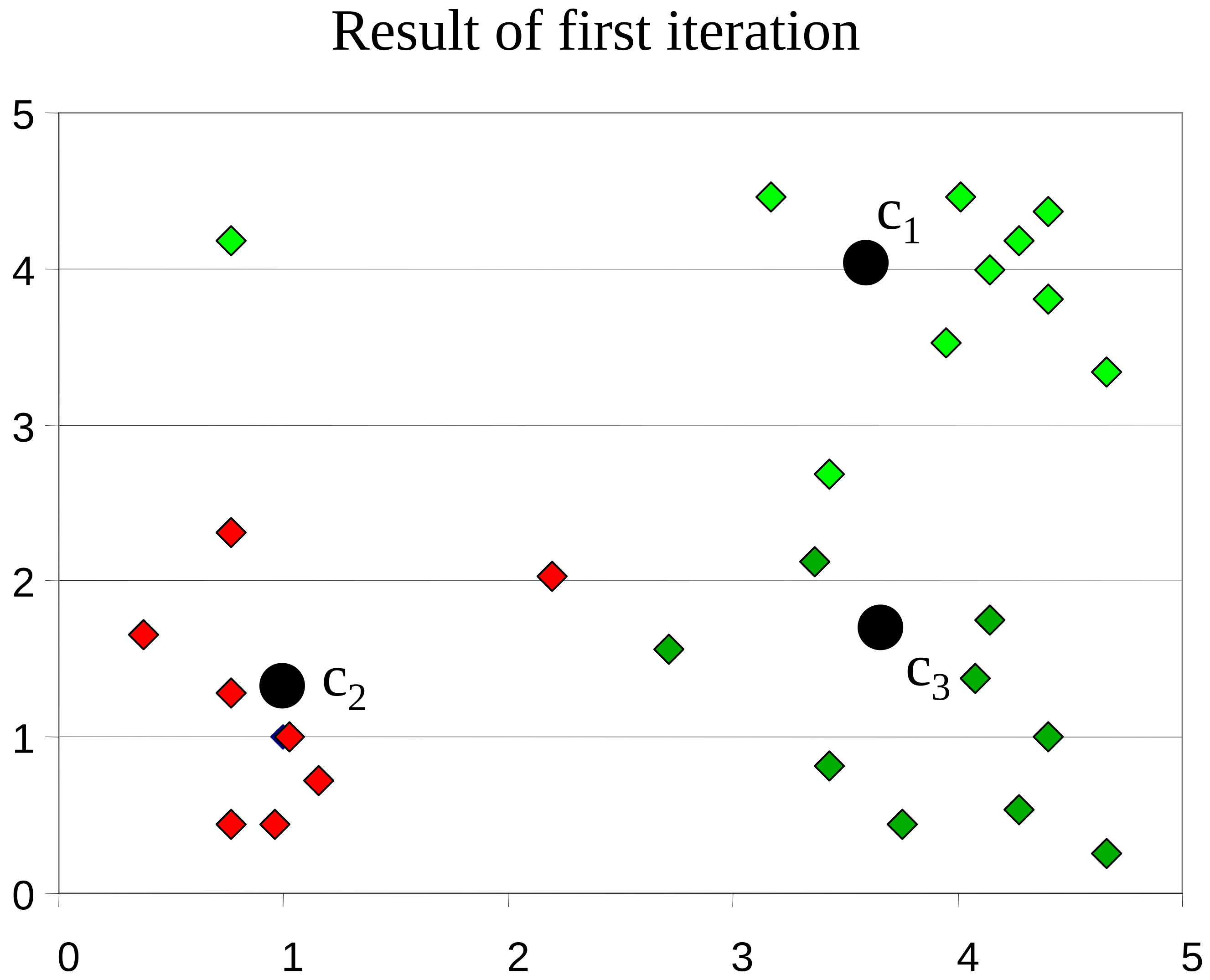


"K-means" clustering: Example

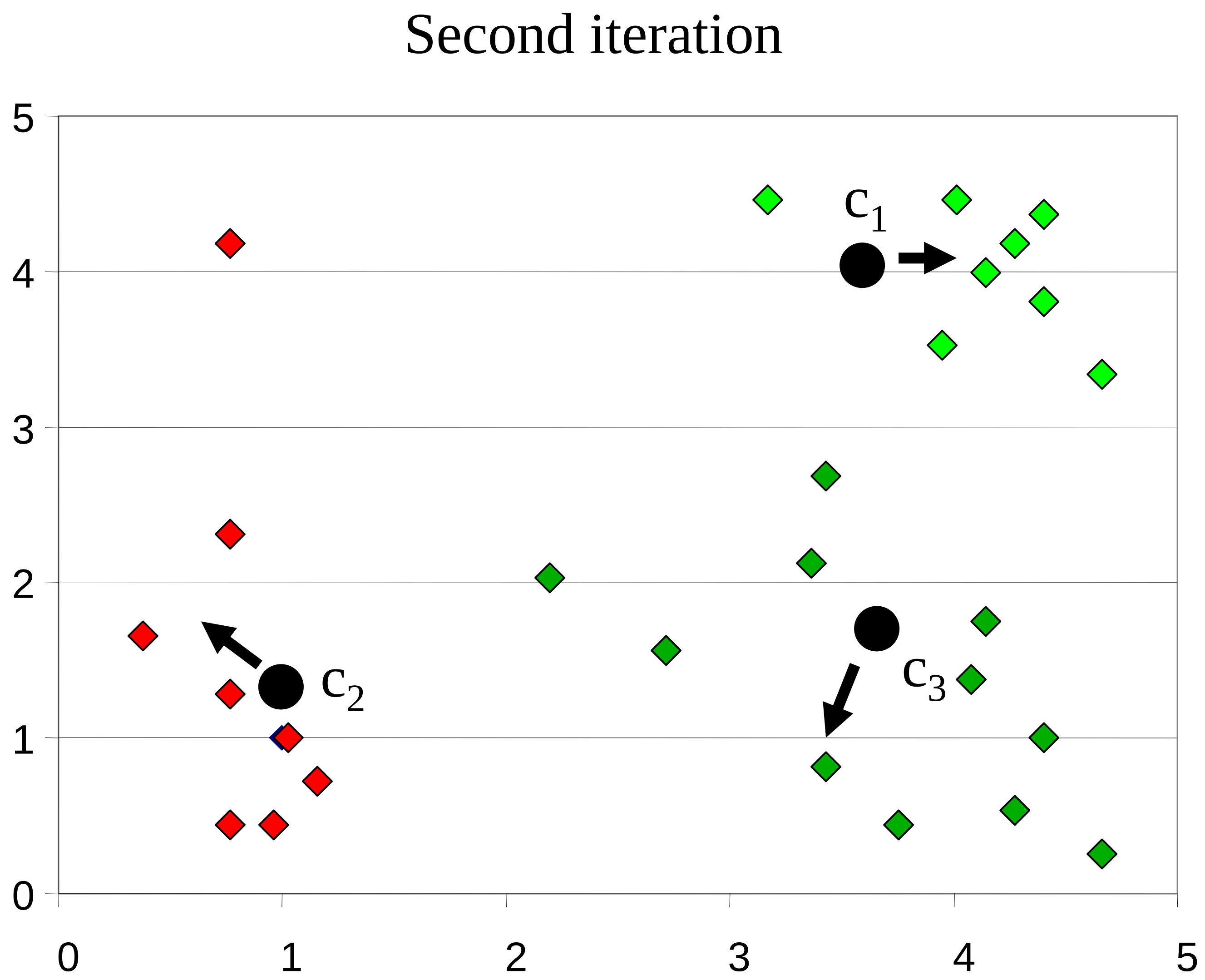
Re-estimate cluster centers (adapt synaptic weights)



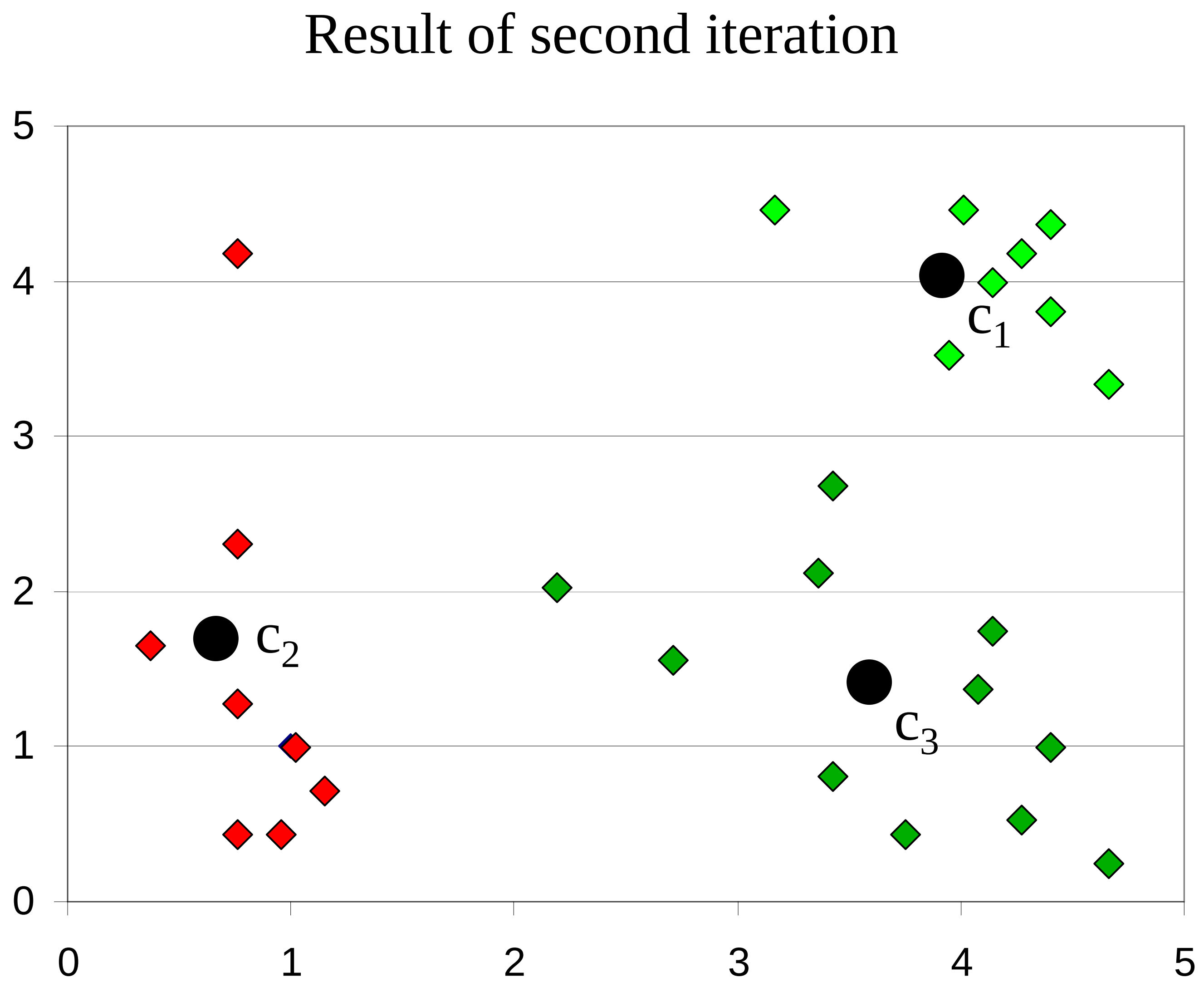
"K-means" clustering: Example



“K-means” clustering: Example



“K-means” clustering: Example



K-means clustering

- Properties
  - Will always converge to *some* solution
  - Can be a "local minimum"
    - does not always find the global minimum of objective function:

$$\sum_{\text{clusters } i} \sum_{\text{points p in cluster } i} ||p - c_i||^2$$

K-means and probability density estimation

- Can formalize K-means as *probability density estimation*
- Model data as a **mixture of K Gaussians:**

$$p[\mathbf{u};G] = \sum_{\mathbf{v}} p[\mathbf{u} \mid \mathbf{v};G]\,p[\mathbf{v};G]$$

- Estimate not only means but also covariances

K-means and the EM algorithm

Expectation Maximization (EM) Algorithm overview:

- Initialize K clusters: $C_1, \ldots, C_K$

  (μ_j, Σ_j) and $P(C_j)$ for each cluster j

1. Estimate which cluster each data point belongs to

   $p(C_j \mid x_i)$ ➡ Expectation step

2. Re-estimate cluster parameters

   $(\mu_j, \Sigma_j),\ p(C_j)$ ➡ Maximization step

3. Repeat 1 and 2 until convergence

EM algorithm for Mixture of Gaussians

- E step: Compute probability of membership in cluster based on output of previous M step ($p(x_i|C_j)$ = Gaussian(μ_j, Σ_j))

$$p(C_j \mid x_i) = \frac{p(x_i \mid C_j) \cdot p(C_j)}{p(x_i)} = \frac{p(x_i \mid C_j) \cdot p(C_j)}{\sum_j p(x_i \mid C_j) \cdot p(C_j)}$$

(Bayes rule)

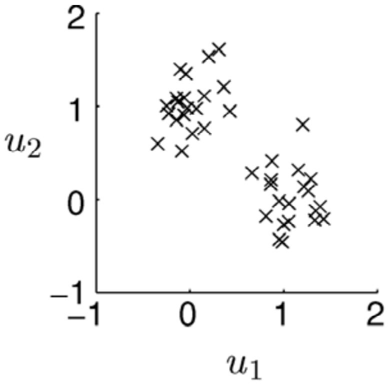
- M step: Re-estimate parameters based on output of E step

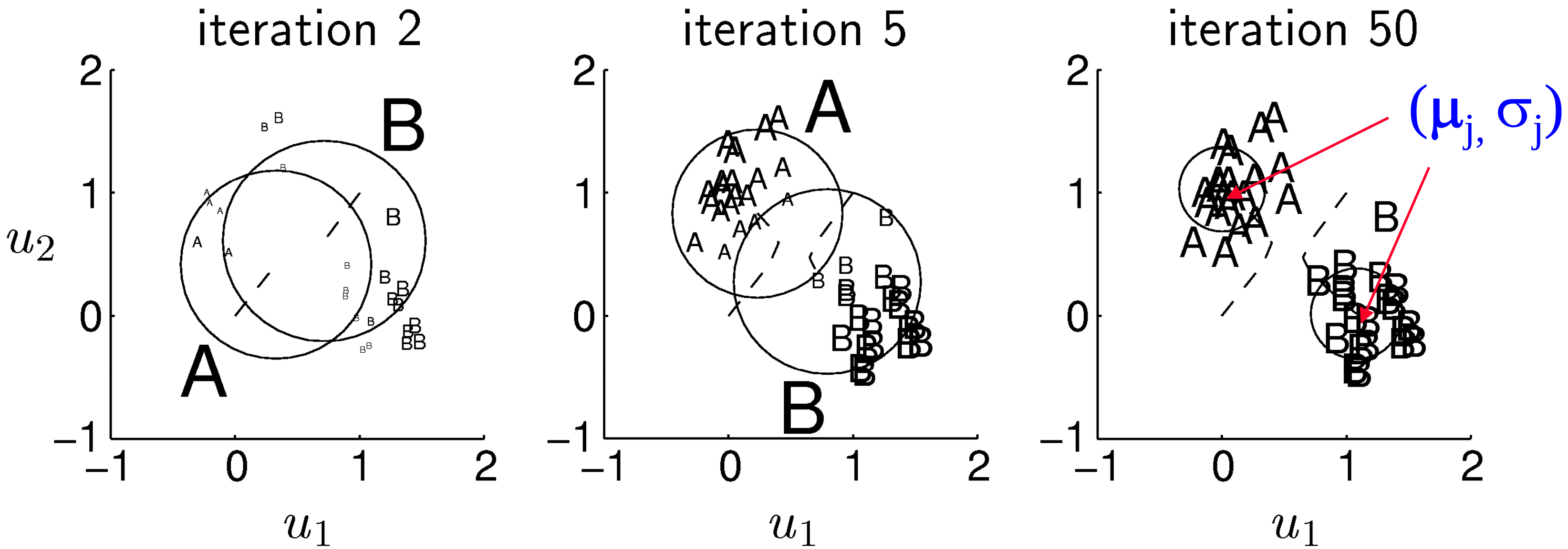
$$\mu_j = \frac{\sum_i p(C_j \mid x_i) \cdot x_i}{\sum_i p(C_j \mid x_i)} \qquad \Sigma_j = \frac{\sum_i p(C_j \mid x_i) \cdot (x_i - \mu_j) \cdot (x_i - \mu_j)^T}{\sum_i p(C_j \mid x_i)} \qquad p(C_j) = \frac{\sum_i p(C_j \mid x_i)}{N}$$

(Learn parameters)

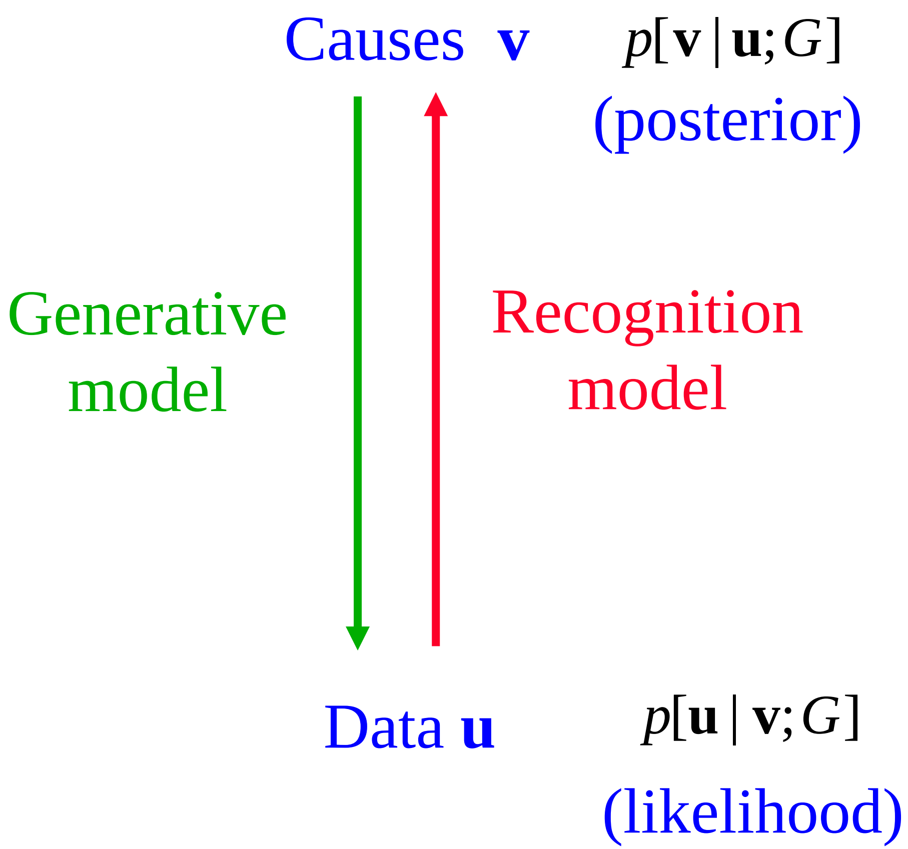
Results from the EM algorithm

Input data:





Recall: Generative versus Recognition Models



Instead of clusters, what if data was generated by linear superposition of causes?
(e.g., an image composed of several features)

Linear Generative Model

- Suppose input **u** is represented by linear superposition of causes $v_1, v_2, \ldots, v_k$ and "features" $\mathbf{g}_i$:

$$\mathbf{u} = \sum_i \mathbf{g}_i v_i = G\mathbf{v}$$

- Problem: For a set of inputs **u**, estimate causes v_i for each **u** and learn feature vectors $\mathbf{g}_i$ (also called basis vectors/filters)
- Idea: Find **v** and *G* that minimize reconstruction errors:

$$E = \frac{1}{2} |\mathbf{u} - \sum_i \mathbf{g}_i v_i|^2 = \frac{1}{2} (\mathbf{u} - G\mathbf{v})^T (\mathbf{u} - G\mathbf{v})$$

Probabilistic Interpretation

- E is the same as the negative log likelihood of data:
  Likelihood = Gaussian with mean $G\mathbf{v}$ and covariance I

$$p[\mathbf{u} \mid \mathbf{v}; G] = N(\mathbf{u}; G\mathbf{v}, I)$$

$$E = -\ln p[\mathbf{u} \mid \mathbf{v}; G] = \frac{1}{2}(\mathbf{u} - G\mathbf{v})^T(\mathbf{u} - G\mathbf{v}) + C$$

- Find $\mathbf{v}$ and G that maximize:

$$F(\mathbf{v}, G) = \langle \ln p[\mathbf{v}, \mathbf{u}; G] \rangle$$ Joint probability of **v** and **u**

$$= \langle \ln p[\mathbf{u} \mid \mathbf{v}; G] + \ln p[\mathbf{v}; G] \rangle$$

Prior for causes (what should this be?)

What do we know about the causes **v**?

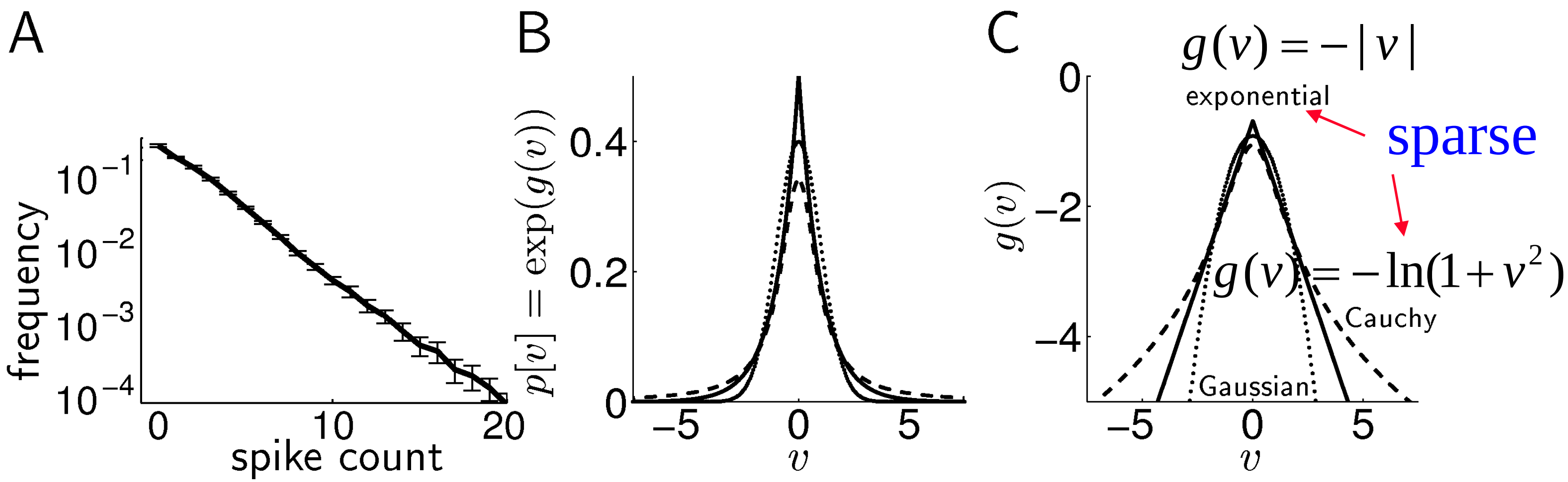
- We would like the causes to be *independent*
  - If cause A and cause B always occur together, then perhaps they should be treated as a single cause AB?
- Examples:
  - Image: Composed of several independent edges
  - Sound: Composed of independent spectral components
  - Objects: Composed of several independent parts
- Idea 1: We would like: $p[\mathbf{v};G] = \prod_a p[v_a;G]$
- Idea 2: If causes are independent, only a few of them will be active for any input → v_a will be 0 most of the time but high for certain inputs → sparse distribution for $p[v_a;G]$

Prior Distributions for Causes

Spikes in area IT in monkey viewing TV

Possible prior distributions

Log prior



$$p[\mathbf{v};G] \propto \prod_a \exp(g(v_a))$$

Finding the optimal **v** and *G*

- Want to maximize:

$$F(\mathbf{v},G) = \left\langle \ln p[\mathbf{u} \mid \mathbf{v};G] + \ln p[\mathbf{v};G] \right\rangle$$

$$= \left\langle -\frac{1}{2}(\mathbf{u}-G\mathbf{v})^T(\mathbf{u}-G\mathbf{v}) + \sum_a g(v_a) \right\rangle + K$$

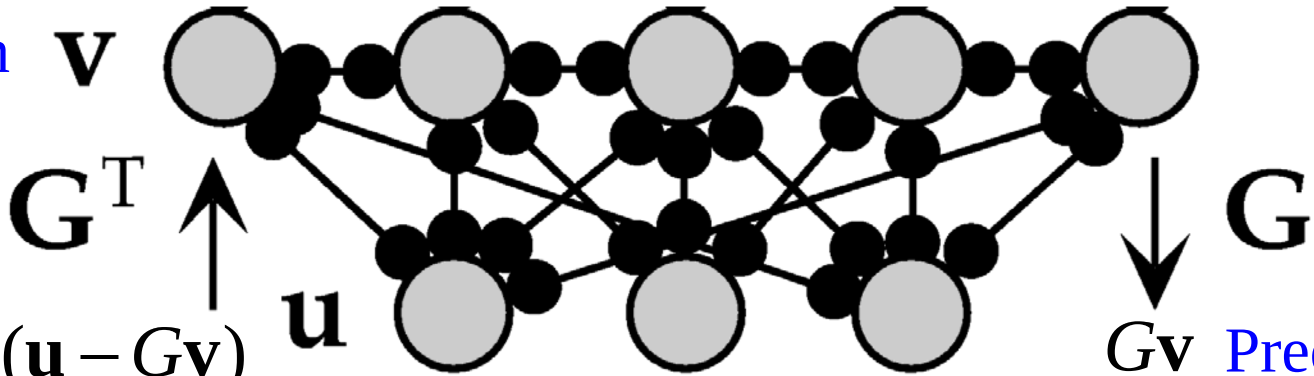
- Approximate EM algorithm:
  - E step: Maximize *F* with respect to **v** keeping G fixed
    - Set d**v**/dt $\propto$ d*F*/d**v** ("gradient ascent/hill-climbing")
  - M step: Maximize *F* with respect to G, given the **v** above
    - Set d*G*/dt $\propto$ d*F*/d*G* ("gradient ascent/hill-climbing")

Network for Estimating **v**

$$\tau \frac{d\mathbf{v}}{dt} = \frac{dF}{d\mathbf{v}} = G^T(\mathbf{u} - G\mathbf{v}) + g'(\mathbf{v})$$

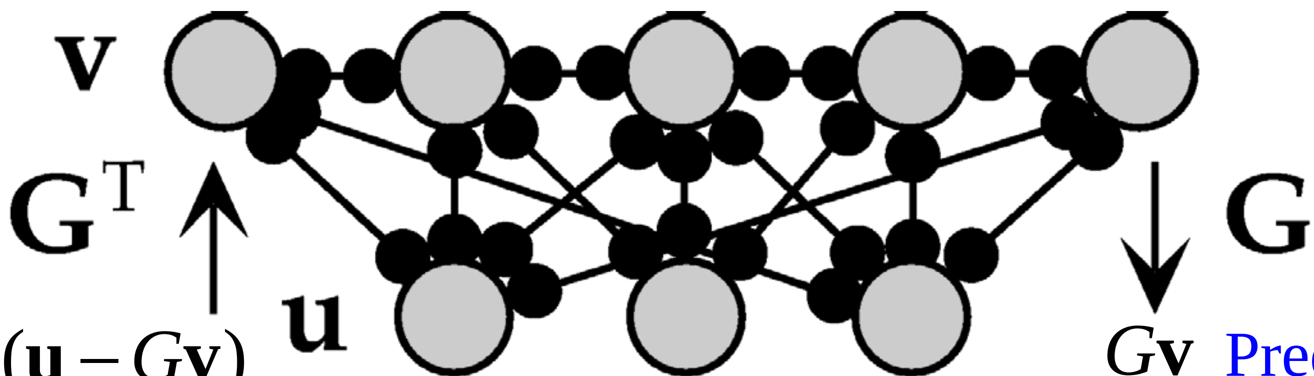
Firing rate dynamics

Error — Sparseness constraint



- Similar to Kalman filtering
- Suggests a role for feedback pathways in the cortex (Rao & Ballard, 1999)

Learning the Synaptic Weights G



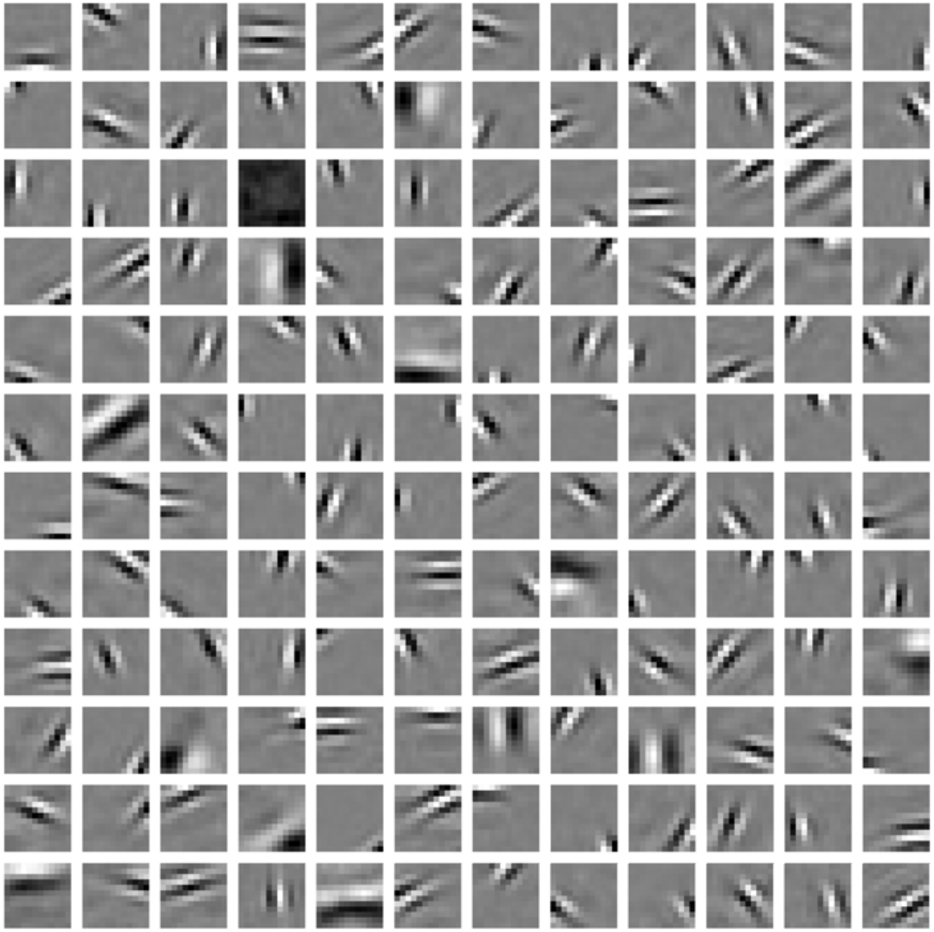
Error $(\mathbf{u} - G\mathbf{v})$ $G\mathbf{v}$ Prediction

Learning rule

$$\tau_G \frac{dG}{dt} = \frac{dF}{dG} = (\mathbf{u} - G\mathbf{v})\mathbf{v}^T$$

Hebbian!
(similar to Oja's rule)

Results of Learning G for Natural Images



Each square is a column $\mathbf{g}_i$ of G (obtained by collapsing rows of the square into a vector)

Almost all the $\mathbf{g}_i$ represent local edge features

Any image $\mathbf{u}$ can be expressed as:

$$\mathbf{u} = \sum_i \mathbf{g}_i v_i = G\mathbf{v}$$

Other Related Ideas

- Independent Component Analysis (ICA): Another algorithm for finding independent causes based on linear model
  - Assumes same number of inputs as outputs
  - Assumes G is invertible ($W = G^{-1}$)
  - Finds optimal W using sparse prior $p[v] \propto 1/\cosh(v)$
  - Reference: Bell & Sejnowski (1995), texbook p. 384
- Predictive Coding: An algorithm for eliminating redundancy by subtracting away predictable parts from a signal **u**
  - Sparse coding network does predictive coding: $(\mathbf{u} - G\mathbf{v})$
  - Can be extended to hierarchies
  - Related to Kalman filtering (Rao, 1999)
  - References: Rao & Ballard (1997, 1999)

Next Class: Supervised Learning

- Things to do:
  - Finish reading Chapters 8 and 10
  - Do Homework #4 (last homework!)
  - Work on mini-project