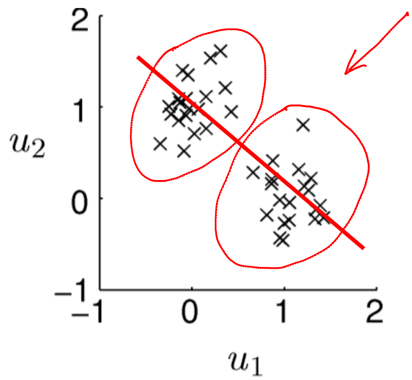
Introduction to Unsupervised Learning

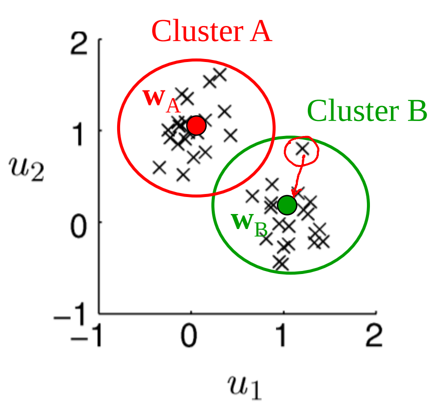


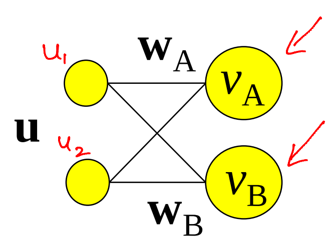
Input data seems to be made up of two clusters of points

Can neurons learn such clusters?

Image Source: Dayan & Abbott textbook

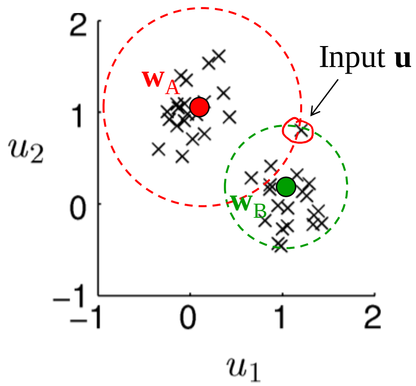
Using Neurons to represent Clusters

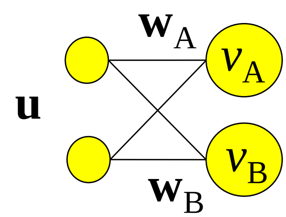




$$v_i = \mathbf{w}_i \cdot \mathbf{u} = \mathbf{w}_i^T \mathbf{u} = \mathbf{u}^T \mathbf{w}_i$$

Most active neuron is the one whose weight vector is closest to an input





$\|u\| = 1$

$\|w_i\| = 1$

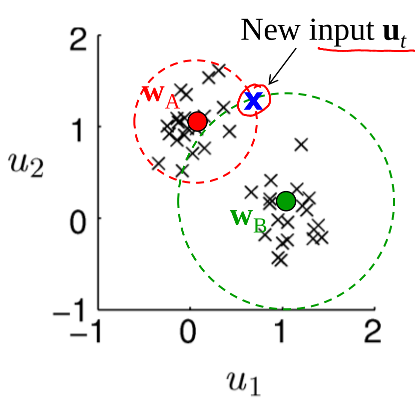
$$\|\mathbf{u}-\mathbf{w}_i\|^2 = (\mathbf{u}-\mathbf{w}_i)^T(\mathbf{u}-\mathbf{w}_i)$$

$$= \mathbf{u}^T\mathbf{u} + \mathbf{w}_i^T\mathbf{w}_i - \mathbf{u}^T\mathbf{w}_i - \mathbf{w}_i^T\mathbf{u}$$

$$= \|\mathbf{u}\|^2 + \|\mathbf{w}_i\|^2 - 2v_i$$

$$\arg\min_i \|\mathbf{u}-\mathbf{w}_i\|^2 = \arg\max_i v_i$$

Updating the Weights given a New Input



- Given new input, pick a cluster (weight vector closest to input)
- Set weight vector to **running average** of all inputs $\mathbf{u}_i$ *in that cluster*

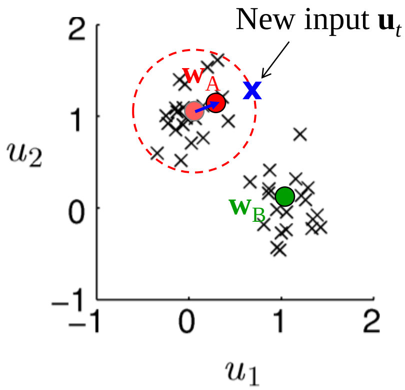
$$\mathbf{w}_A(t) = \left(\sum_{i=1}^{t}\mathbf{u}_i\right)/t = \left(\sum_{i=1}^{t-1}\mathbf{u}_i + \mathbf{u}_t\right)/t$$

$$= \frac{(t-1)}{t}\mathbf{w}_A(t-1) + \frac{\mathbf{u}_t}{t}$$

$$= \mathbf{w}_A(t-1) + (1/t)(\mathbf{u}_t - \mathbf{w}_A(t-1))$$

$$\Delta\mathbf{w}_A = \varepsilon\cdot(\mathbf{u}_t - \mathbf{w}_A)$$

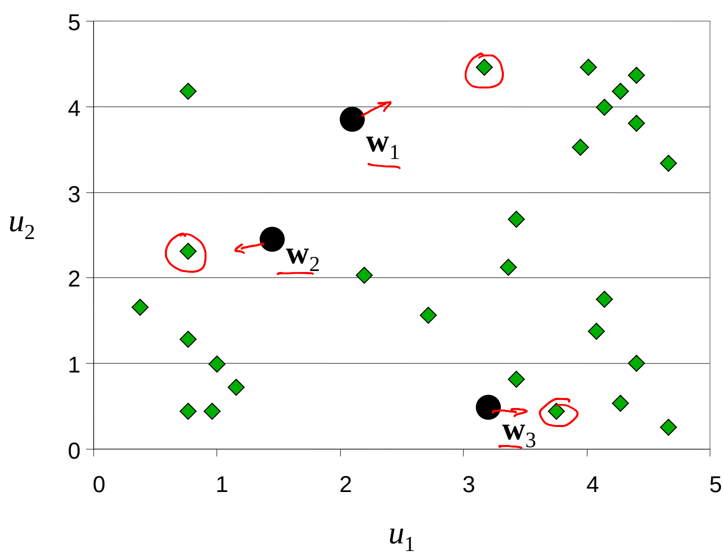
Competitive Learning



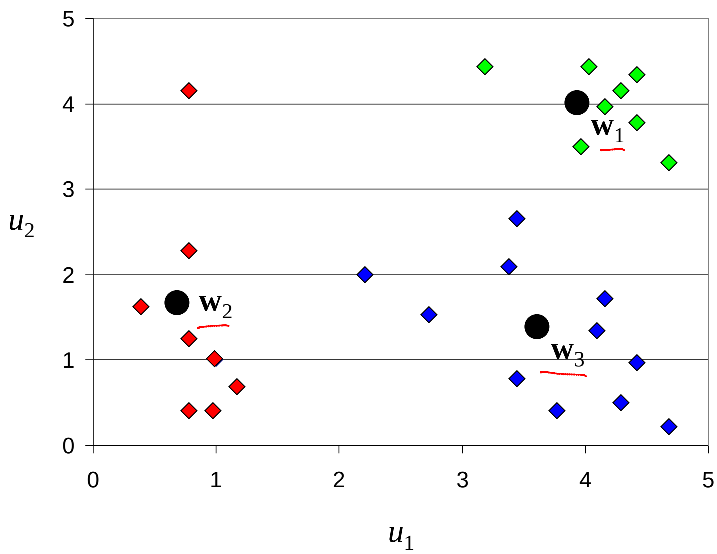
- Given a new input, pick the most active neuron ("winner-takes-all")
  - $\Rightarrow$ One whose weights are closest to new input
- Update weight vector for that neuron

$$\Delta \mathbf{w} = \varepsilon \cdot (\mathbf{u}_t - \mathbf{w})$$

Competitive Learning Example: Initial Weights

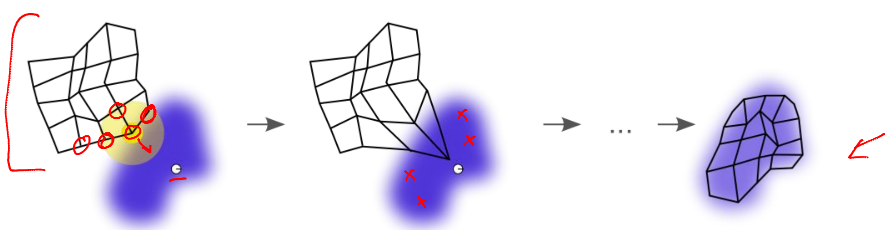


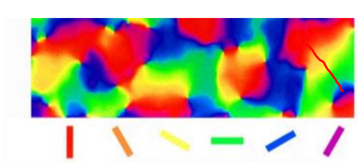
Competitive Learning Example: After Updates



Competitive Learning and Self Organizing Maps (a.k.a. Kohonen Maps)

- Given an input, pick the winning neuron
- Update weights for that neuron AND other neurons *in the neighborhood* of the winner

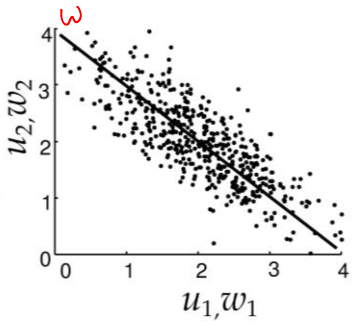




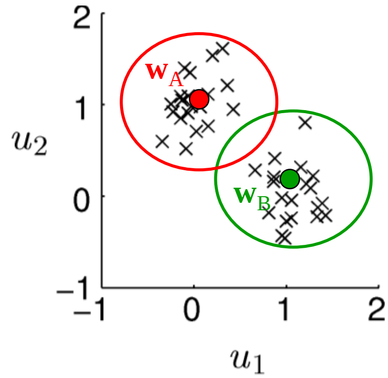
Orientation preference map in the primary visual cortex (V1)

Image Source: Wikimedia Commons

PCA/Hebb/Covariance rule



Competitive Learning



Is there a principled way of learning models of input data?

Image Source: Dayan & Abbott textbook

Enter…Unsupervised Learning

Causes v

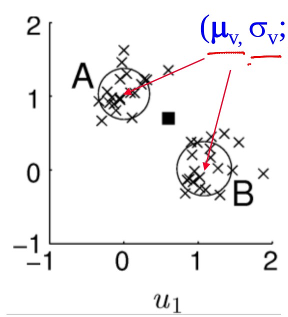
Prior $p[v;G]$

Generative model

Data points $\mathbf{u}$

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

$(\mu_v, \sigma_v; v = A, B)$



Mixture of Gaussians Model:

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

Parameters $G = (\mu_v, \sigma_v, \gamma_v)$ Gaussian Prior = γ_v

The Goal of Unsupervised Learning

Prior $p[\mathbf{v};G]$

Causes $\mathbf{v}$

Posterior $p[\mathbf{v}\,|\,\mathbf{u};G]$

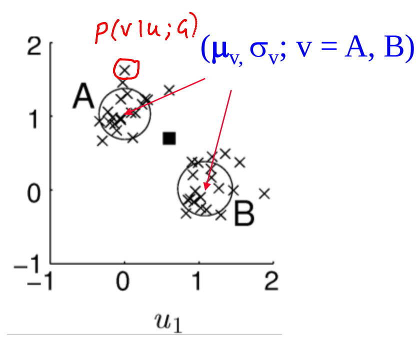
Generative model

Likelihood $p[\mathbf{u}\,|\,v;G]$

Data $\mathbf{u}$

- **Goal:** Learn a good "Generative Model" for the data you are seeing
  - Mimic the data generation process
- **General Approach:** Given data **u**, need to solve two sub-problems:
  - RECOGNITION Estimate causes **v** (compute posterior)
  - Learn parameters *G*

Clustering as Unsupervised Learning



How do we learn the model parameters $G = (\mu_v, \sigma_v, \gamma_v)$?

Mixture of Gaussians Model:

$$p[\mathbf{u};G] = \sum_v p[\mathbf{u}\,|\,v;G]\,p[v;G]$$

Gaussian Prior = γ_v

EM algorithm for Unsupervised Learning

- Stands for Expectation-Maximization algorithm

E STEP
M STEP

- *Iterate* the following two steps until convergence:
  - E step: Compute posterior distribution of v (= A, B) for each $\mathbf{u}$:

$$p[v\,|\,\mathbf{u};G] = \frac{p[\mathbf{u}\,|\,v;G]\,p[v;G]}{p[\mathbf{u};G]} = \frac{N(\mathbf{u};\boldsymbol{\mu}_v,\sigma_v I)\cdot\gamma_v}{\sum_v N(\mathbf{u};\mu_v,\sigma_v I)\cdot\gamma_v}$$

(Bayes rule)
"Soft" competition
(not winner-takes-all)

  - M step: Change parameters G using results from E step

$$\boldsymbol{\mu}_v = \frac{\sum_{\mathbf{u}} p[v\,|\,\mathbf{u};G]\cdot\mathbf{u}}{\sum_{\mathbf{u}} p[v\,|\,\mathbf{u};G]}, \quad \sigma_v^{\;2} = \frac{\sum_{\mathbf{u}} p[v\,|\,\mathbf{u};G]\,|\,\mathbf{u}-\boldsymbol{\mu}_v\,|^2}{\sum_{\mathbf{u}} p[v\,|\,\mathbf{u};G]}$$

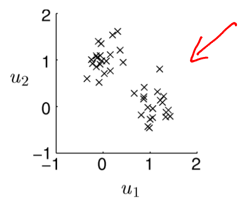
(Update parameters)

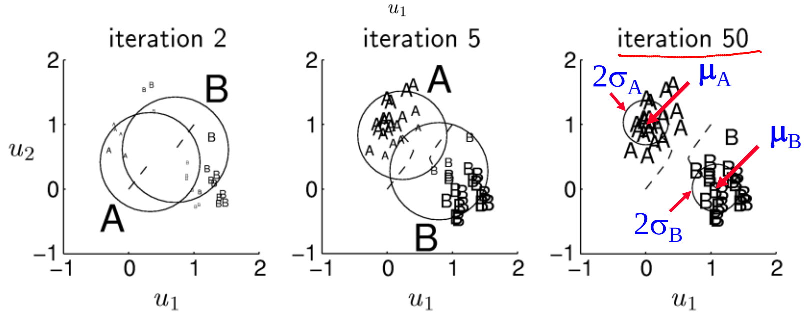
$$\gamma_v = \sum_{\mathbf{u}} p[v\,|\,\mathbf{u};G]\,/\,N_u$$

13

Results from the EM algorithm

Input data:

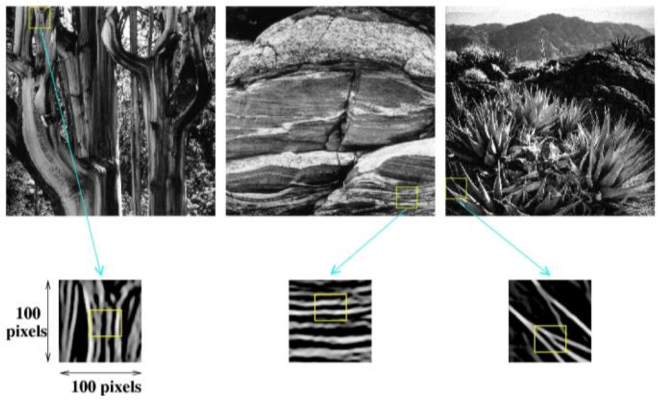




14

Image Source: Dayan & Abbott textbook

Suppose the Data are Natural Images…



Causes $\mathbf{v}$

$p[\mathbf{v}; G]$

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

Data $\mathbf{u}$

What kind of generative model would you use?
How do you learn the "causes" of such images?

Image Source: Rao & Ballard, 1999