CSE/NEUBEH 528

Lecture 13: Unsupervised Learning

(Chapters 8 & 10)

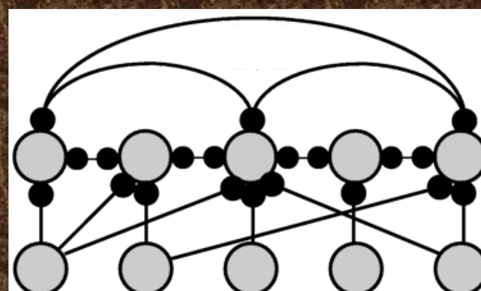


Image from http://clasdean.la.asu.edu/news/images/ubep2001/neuron3.jpg

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

Gameplan for Today

- Unsupervised (Representational) Learning
  - Flashback: Hebb rule and Principal Component Analysis (PCA)
  - Causal Models
  - Generative versus Recognition Models
  - Density Estimation
  - Sparse Coding & Independent Component Analysis (ICA)

Flashback: Hebb Rule

- Consider a linear neuron:
  $v = \mathbf{w}^T \mathbf{u} = \mathbf{u}^T \mathbf{w}$
- Basic Hebb Rule: $\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{u}v$
- What is the average effect of this rule over many inputs?
  $$\tau_w \frac{d\mathbf{w}}{dt} = \langle \mathbf{u}v \rangle = Q\mathbf{w}$$
- Q is the input correlation matrix: $Q = \langle \mathbf{u}\mathbf{u}^T \rangle$

Variants of the Hebb Rule

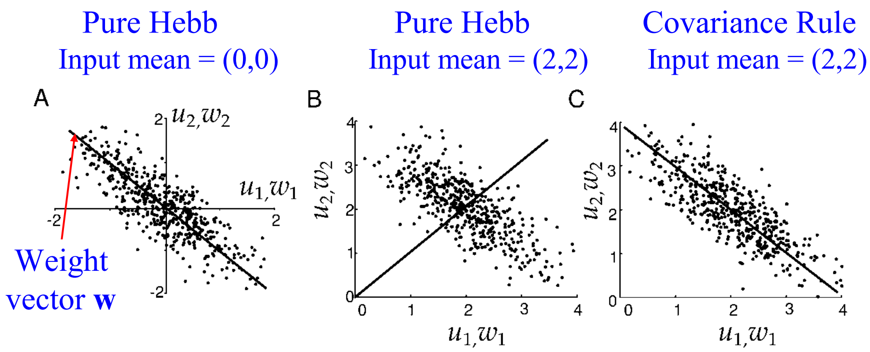
- Pure Hebb only increases synaptic weights (LTP)
  - What about LTD?
- Covariance rules:
  $$\tau_w \frac{d\mathbf{w}}{dt} = (\mathbf{u} - \boldsymbol{\theta}_{\mathrm{u}})v$$
  $$\tau_w \frac{d\mathbf{w}}{dt} = \mathbf{u}(v - \theta_v)$$
- Oja's Rule: $\tau_w \frac{d\mathbf{w}}{dt} = (\mathbf{u} - \alpha \mathbf{w} v)v$ (stable, $\|w\|^2 \to 1/\alpha$)

What does the Hebb rule do?

Recall from last time:

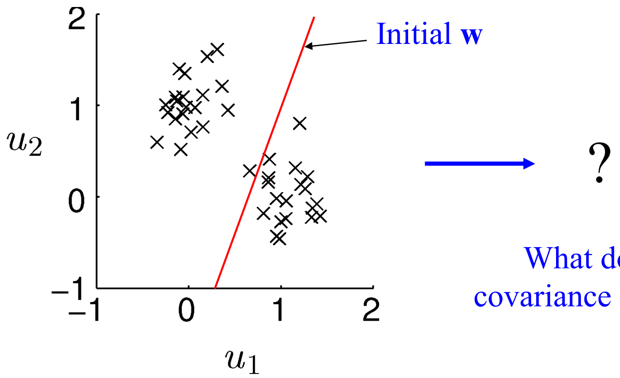
Eigenvector analysis of Hebb rule…

Hebb Rule implements PCA!



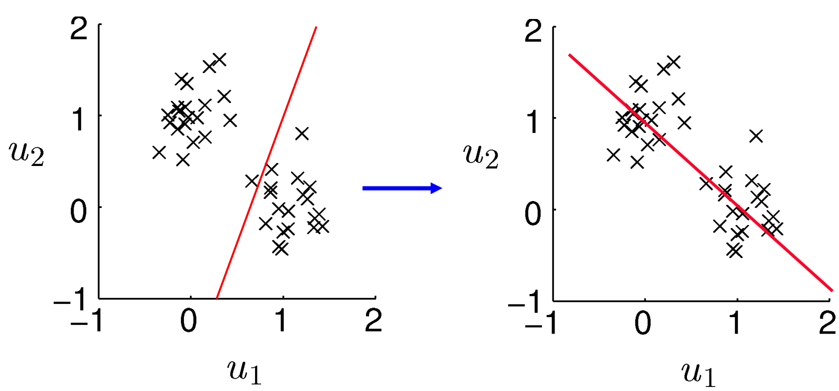
Hebb rule *rotates* weight vector to align with principal eigenvector of input correlation/covariance matrix (i.e. direction of maximum variance)

What about this data?



What does the covariance rule learn?

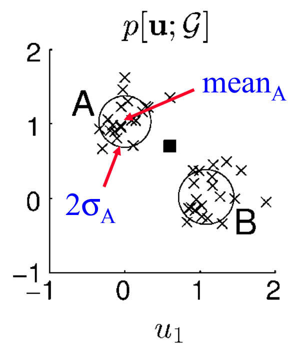
PCA does not correctly describe the data



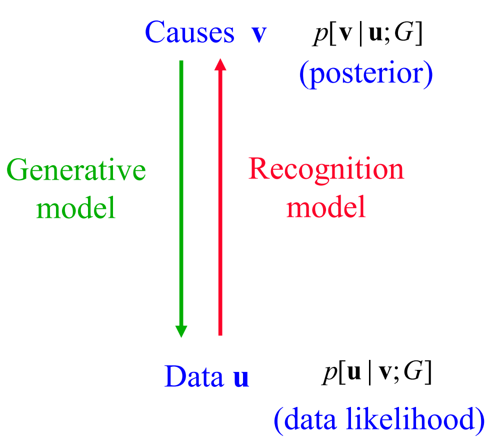
Input data is made up of two clusters (Gaussians) → two "causes"

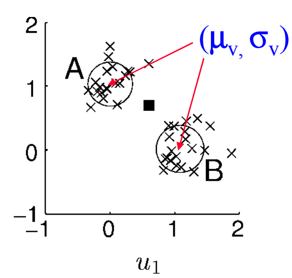
Causal Models

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



Generative versus Recognition Models





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

Prior $= \gamma_v$

EM algorithm for Learning Data Clusters

- Stands for Expectation-Maximization algorithm
- Repeat the following two steps:
  - E step: Compute recognition distribution (v = A or B) for each $\mathbf{u}$:

$$p[v \mid \mathbf{u}; G] = \frac{p[\mathbf{u} \mid v; G]\, p[v; G]}{p[\mathbf{u}; G]} \qquad \text{(Bayes rule)}$$

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

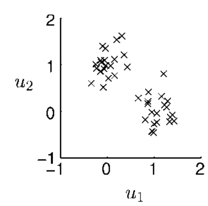
$$\gamma_v = \langle p[v \mid \mathbf{u}; G] \rangle, \quad \boldsymbol{\mu}_v = \frac{\langle p[v \mid \mathbf{u}; G]\mathbf{u} \rangle}{\gamma_v},$$

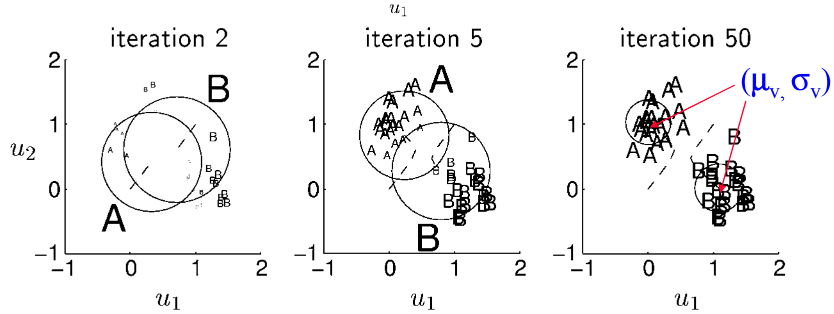
$$\sigma_v = \frac{\langle p[v \mid \mathbf{u}; G] \, | \mathbf{u} - \boldsymbol{\mu}_v |^2 \rangle}{2\gamma_v}$$

(Learn parameters)

Results from the EM algorithm

Input data:





Another Example: 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 and EM

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

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

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

- EM algorithm finds **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**?

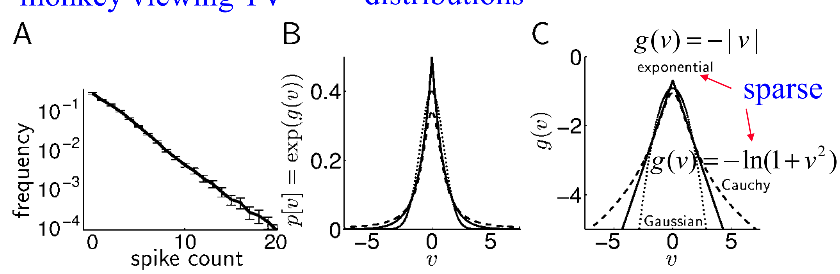
- We would like the causes to be *independent*
  - If cause A and cause B always occur together, then perhaps they should be treated as a single cause AB?
- Examples:
  - Image: Composed of several independent edges
  - Sound: Composed of independent spectral components
  - Objects: Composed of several 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:

$$F(\mathbf{v},G) = \langle \ln p[\mathbf{u} \mid \mathbf{v};G] + \ln p[\mathbf{v};G] \rangle$$
$$= \left\langle -\frac{1}{2}(\mathbf{u}-G\mathbf{v})^T(\mathbf{u}-G\mathbf{v}) + \sum_a g(v_a) \right\rangle + K$$

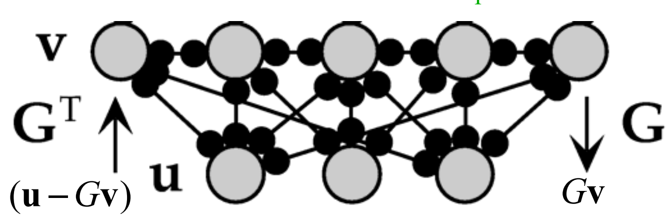
- EM:
  - E step: Maximize F with respect to **v** keeping G fixed
    - Set d**v**/dt ∝ dF/d**v** (“gradient ascent/hill-climbing”)
  - M step: Maximize F with respect to G, given the **v** above
    - Set dG/dt ∝ dF/dG (“gradient ascent/hill-climbing”)

Network for Estimating v and Learning G

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

Firing rate dynamics

Error — Sparseness constraint

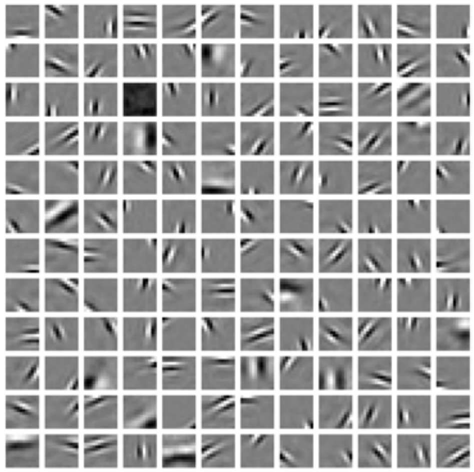


Learning rule

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

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:

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

Ideas related to Sparse Coding

- 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
  - Reference: Rao & Ballard (1997, 1999)

Next Class: Supervised Learning

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