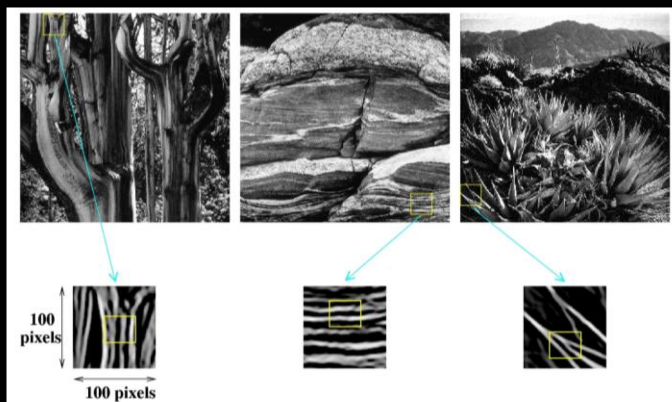
Sparse Coding and Predictive Coding



Can we learn a good representation of natural images?
What does the brain do?

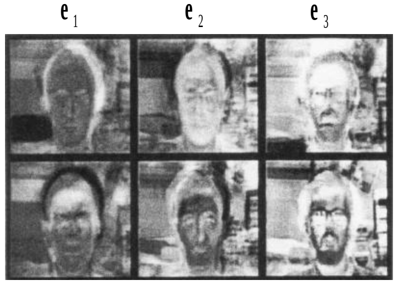
Eigenvectors strike again?

Input data **u**:
Face images
(*N* pixels total)

………

e_1 e_2 e_3

Eigenvectors of the Input Covariance Matrix



………

"Eigenfaces"

Image Source: Turk & Pentland, 1991

A Linear Model of Images using Eigenvectors

Input image **u** $\qquad \mathbf{e}_1 \qquad \mathbf{e}_2 \qquad \mathbf{e}_3$

 $=$  $\cdot v_1 +$  $\cdot v_2 +$  $\cdot v_3 + \cdots$

$$\mathbf{u} = \sum_{i=1}^{N} \mathbf{e}_i v_i$$

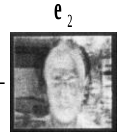
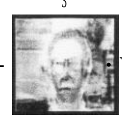
Suppose you use only the first M principal eigenvectors:

$$\mathbf{u} = \sum_{i=1}^{M} \mathbf{e}_i v_i + noise \qquad (\overset{=10}{M} < \overset{=10^6}{N})$$

1000 × 1000

Not so fast, Eigenvectors!

Input image **u** $\qquad \mathbf{e}_1 \qquad \mathbf{e}_2 \qquad \mathbf{e}_3$

 $=$  $\cdot v_1 +$  $\cdot v_2 +$  $\cdot v_3 + \cdots$

$$\mathbf{u} = \sum_{i=1}^{M} \mathbf{e}_i v_i + noise \qquad (M < N)$$

Eigenvector representation is good for compression but not so good if you want to extract the *local components* (or *parts*) of an image (e.g., parts of a face, local edges in a scene, etc.)

A Linear Model for Natural Images

Input image $\mathbf{u}$

$$= \underset{\mathbf{g}_1}{\boxed{?}} \cdot v_1 + \underset{\mathbf{g}_2}{\boxed{?}} \cdot v_2 + \underset{\mathbf{g}_3}{\boxed{?}} \cdot v_3 + \cdots$$

$$\mathbf{u} = \sum_{i=1}^{M} \mathbf{g}_i v_i + noise$$

$$= G\mathbf{v} + noise$$

(Note: M can be larger than N)

G = matrix whose columns are $\mathbf{g}_i$
$\mathbf{v}$ = vector whose elements are v_i

Defining the Generative Model: Likelihood

Causes $\mathbf{v}$

Prior
$p[\mathbf{v}]$

Generative model

Input Data
$\mathbf{u}$

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

Linear model:

$$\mathbf{u} = G\mathbf{v} + noise$$

Assume *noise* is Gaussian white noise:

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

$$\propto \exp(-\frac{1}{2}\|\mathbf{u} - G\mathbf{v}\|^2)$$

Log likelihood:

$$\log p[\mathbf{u} \mid \mathbf{v}; G] = -\frac{1}{2}\|\mathbf{u} - G\mathbf{v}\|^2 + C$$

Defining the Generative Model: Prior

Causes $\mathbf{v}$

Prior
$p[\mathbf{v}]$

Generative model

Input Data $\mathbf{u}$

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

Assuming causes v_i are independent:

$$p[\mathbf{v}] = \prod_i p[v_i]$$

$$\log p[\mathbf{v}] = \sum_i \log p[v_i; G]$$

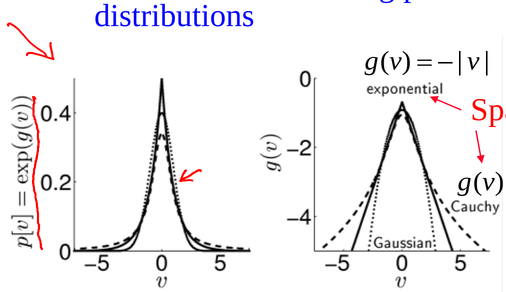
For any input, we want only a few causes v_i to be active

- $v_i = 0$ most of the time but high for some inputs
- Suggests sparse distribution for $p[v_i]$: peak at 0 but with heavy tail (also called *super-Gaussian* distribution)

Examples of Sparse Prior Distributions

Possible prior distributions

Log prior



$$p[\mathbf{v}] = c \cdot \prod_i \exp(g(v_i))$$

$$\log p[\mathbf{v}] = \sum_i g(v_i) + c$$

Image Source: Dayan & Abbott textbook

Bayesian approach to finding v and learning G

- Find **v** and G that maximize posterior probability of causes:

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

- Equivalently, maximize log posterior:

$$F(\mathbf{v},G) = \log p[\mathbf{u}\,|\,\mathbf{v};G] + \log p[\mathbf{v};G] + \log k$$
$$= -\frac{1}{2}\|\mathbf{u}-G\mathbf{v}\|^2 + \sum_i g(v_i) + K$$

Alternate between two steps
(similar to EM algorithm):

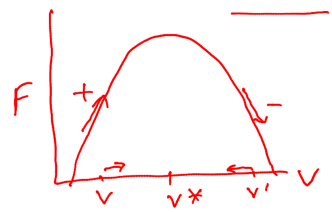
Maximize F with respect to **v**,
keeping G fixed

How?

Maximize F with respect to G,
given the **v** from above

How?

Gradient ascent $\frac{d\mathbf{v}}{dt} \propto \frac{dF}{d\mathbf{v}}$



Finding v for a given input

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

Derivative of g

Reconstruction
(prediction) of **u**

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

Firing rate dynamics
(Recurrent network)

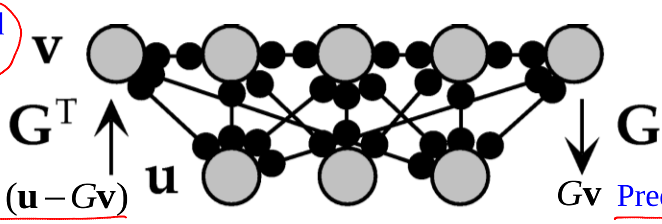
Error

Sparseness constraint

Recurrent Network Implementation of Sparse Coding

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

Corrected Estimate

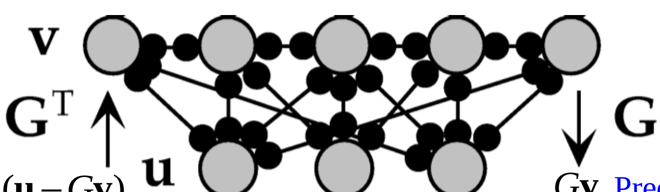


Error $(\mathbf{u} - G\mathbf{v})$

$G\mathbf{v}$ Prediction

Image Source: Dayan & Abbott textbook

Learning the Synaptic Weights G



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

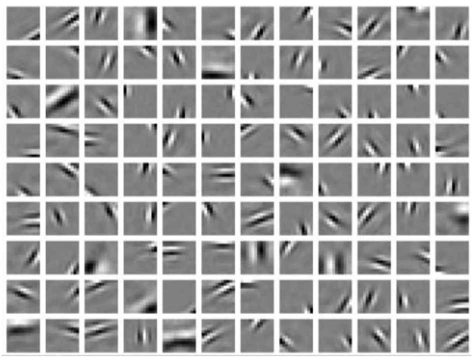
$G\mathbf{v}$ Prediction

Gradient ascent $\quad \frac{dG}{dt} \propto \frac{dF}{dG} = (\mathbf{u} - G\mathbf{v})\mathbf{v}^T$

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

Hebbian!
(similar to Oja's rule)

Result: Learning G for Natural Images

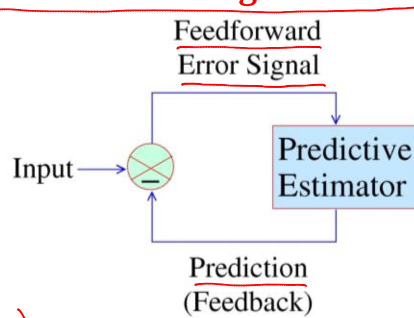


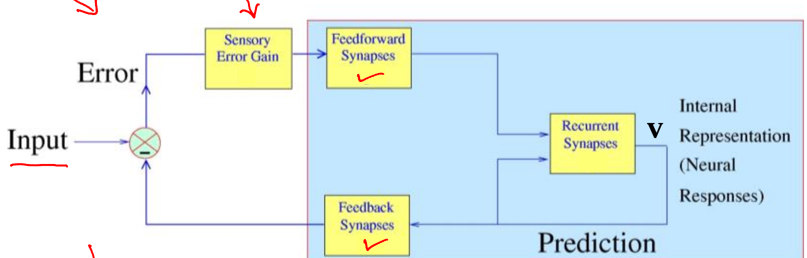
(Olshausen & Field, 1996)

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

The $\mathbf{g}_i$ look like local edge or bar features similar to receptive fields in primary visual cortex (V1)

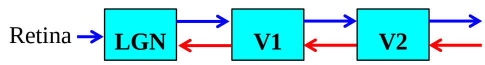
Sparse Coding Network is a special case of Predictive Coding Networks

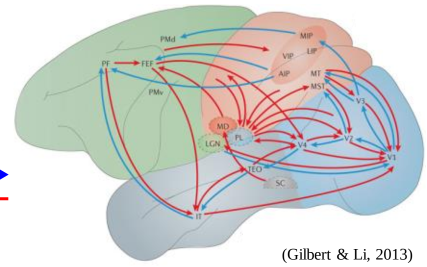




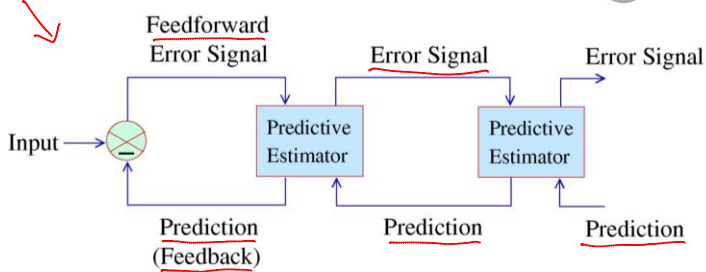
See Supplementary Materials: (Rao, *Vision Research*, 1999; Rao & Ballard, *Nature Neurosci*., 1999)

Predictive Coding Model of the Visual Cortex



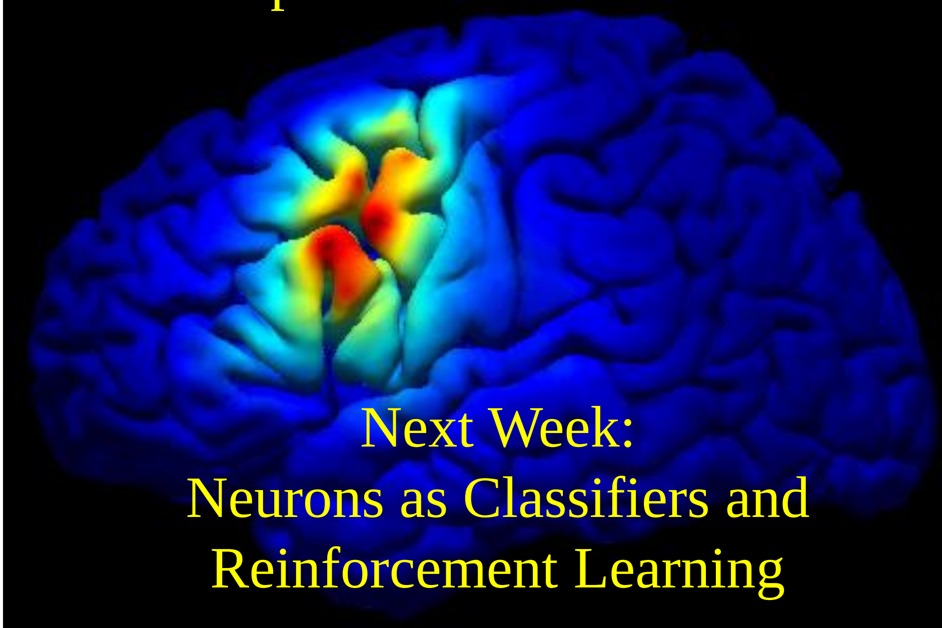


(Gilbert & Li, 2013)



See Supplementary Materials: (Rao & Ballard, *Nature Neurosci*., 1999)

Computational Neuroscience



Next Week:
Neurons as Classifiers and Reinforcement Learning