中国传媒大学

深度学习与类脑计算

（三）

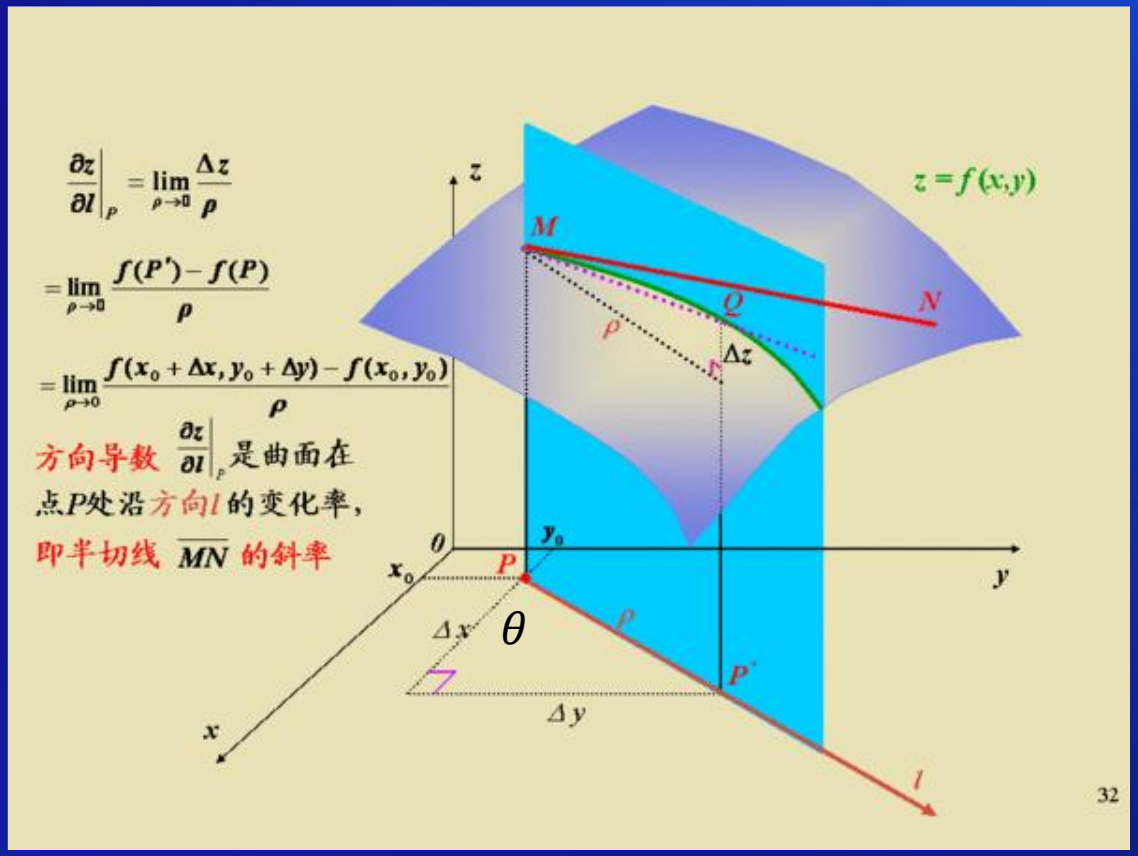
曹立宏

脑科学与智能媒体研究院

李世石 VS 阿狗

- AlphaGo采用了类脑计算+深度学习技术
- AlphaGo之父的PhD是脑科学，不是计算机!
- 类脑智能已经来临，将步入快速发展期!

方向导数-偏导-梯度



- 方向导数

$$y = ax \quad a = \tan\theta = \frac{\sin\theta}{\cos\theta}$$

$$\frac{\partial f}{\partial l} = \cos\theta\frac{\partial f}{\partial x} + \sin\theta\frac{\partial f}{\partial y}$$

$$\frac{\partial f}{\partial l} = \begin{pmatrix}\cos\theta\\ \sin\theta\end{pmatrix}^T \begin{pmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{pmatrix}$$

梯度是变化最大的方向 $\begin{pmatrix}\frac{\partial f}{\partial x}\\ \frac{\partial f}{\partial y}\end{pmatrix} \triangleq \nabla f(x,y)$

多变量展开

f: $R^n \rightarrow R^1$

- 偏导
- Jacobian矩阵 1xn
- Hessian矩阵 nxn

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} \qquad \mathrm{H}(f) = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1 \partial x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_n} \end{bmatrix}$$

$$f(\vec{x}) \approx f(\vec{x}_0) + [\nabla f(\vec{x}_0)]^T (\vec{x} - \vec{x}_0) + \frac{1}{2} (\vec{x} - \vec{x}_0)^T \mathrm{H}(f) (\vec{x} - \vec{x}_0)$$

多变量复合求导

$$f\colon R^n \to R^1 \qquad R^n \xrightarrow{g} R^m \xrightarrow{h} R^1$$

$$f(\vec{x}) = h\big(g(\vec{x})\big) = h\begin{pmatrix} g_1(\vec{x}) \\ \vdots \\ g_m(\vec{x}) \end{pmatrix}$$

$$\frac{\partial f}{\partial x_i} = \frac{\partial h(g(\vec{x}))}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial h\,(\vec{g})}{\partial g_k} \frac{\partial g_k(\vec{x})}{\partial x_i} = \sum_{k=1}^{m} \frac{\partial h}{\partial g_k} \frac{\partial g_k}{\partial x_i}$$

$$\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = [\nabla g]^T \nabla h \qquad \nabla g = \begin{bmatrix} \frac{\partial g_1}{\partial x_1} & \frac{\partial g_1}{\partial x_n} \\ \vdots \quad \cdots & \vdots \\ \frac{\partial g_m}{\partial x_1} & \frac{\partial g_m}{\partial x_n} \end{bmatrix}$$

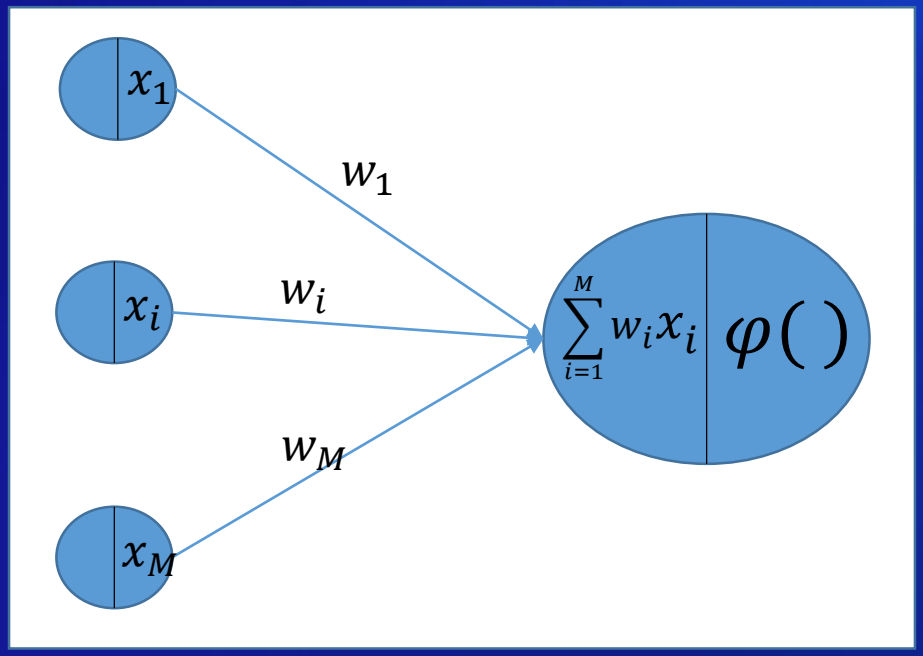
问题和参数法思路

$$R^n \rightarrow R^m \qquad \{\vec{x}_i, \vec{y}_i, \mid_{i=1, \cdots, N}\}$$

$$\vec{y} = f(\vec{x}) \approx g(\vec{x}; \vec{\theta})$$

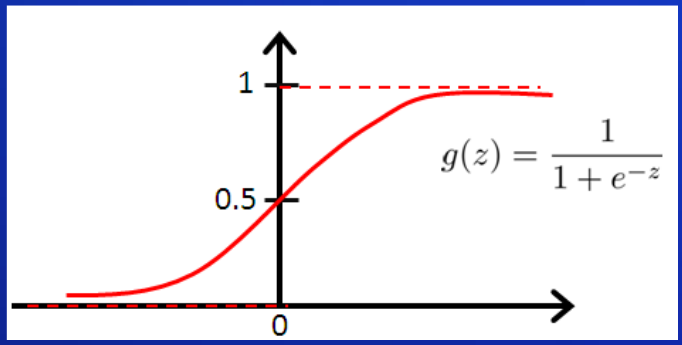
- Use parameters to reduce functional space
- Examples:
  - polynomial approximation
  - Spline approximation

ANN Neuron



- Activation function
  - *Logistic Function*

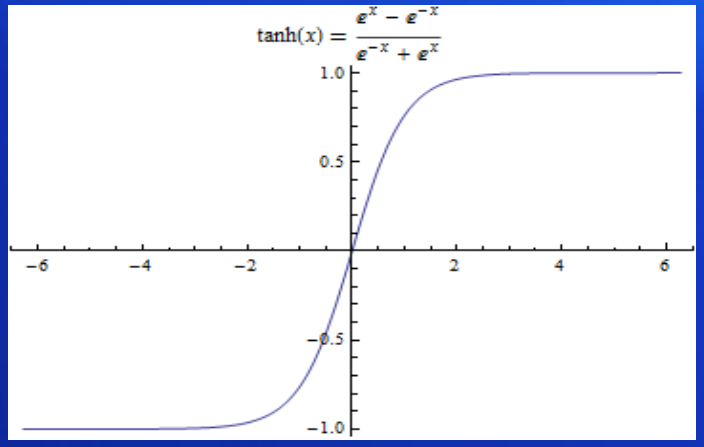
$$\varphi(z) = \frac{1}{1 + e^{-z}}$$



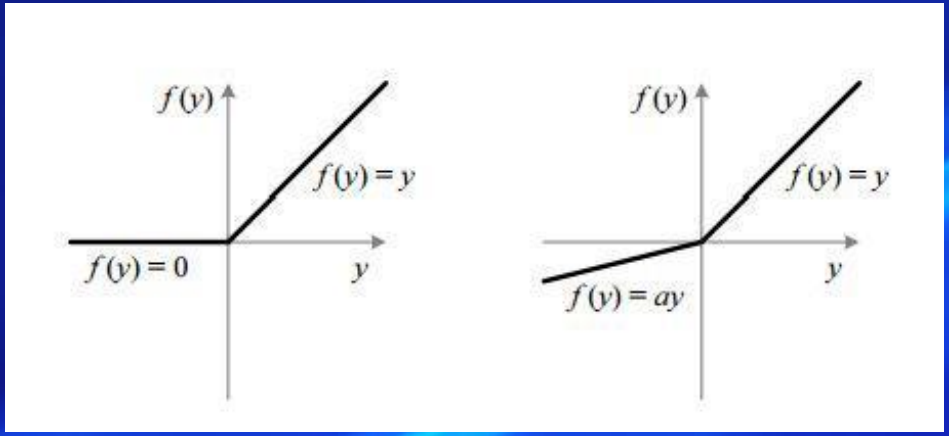
$$y = f(\vec{x}) = g(\vec{x}; \vec{w}) = \varphi(\vec{w}^T \vec{x})$$

Activation Functions

- *Logistic Function*
- *Hyperbolic tangent*
  - $\varphi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$
  - $\varphi'(z) = 1 - \varphi^{2}(z)$
  - *Not sensitive*



- *Rectified Linear Unit* or ReLU (Jarrett *et al.*, 2009; Nair and Hinton, 2010; Glorot *et al.*, 2011a)



Universal Approximation Theorem

Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotone-increasing continuous function. Let I_{m_0} denote the m_0-dimensional unit hypercube $[0, 1]^{m_0}$. The space of continuous functions on I_{m_0} is denoted by $C(I_{m_0})$. Then, given any function $f \ni C(I_{m_0})$ and $\varepsilon > 0$, there exist an integer m_1 and sets of real constants α_i, b_i, and w_{ij}, where $i = 1, \ldots, m_1$ and $j = 1, \ldots, m_0$ such that we may define

$$F(x_1, \ldots, x_{m_0}) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i \right) \tag{4.88}$$

as an approximate realization of the function $f(\cdot)$; that is,

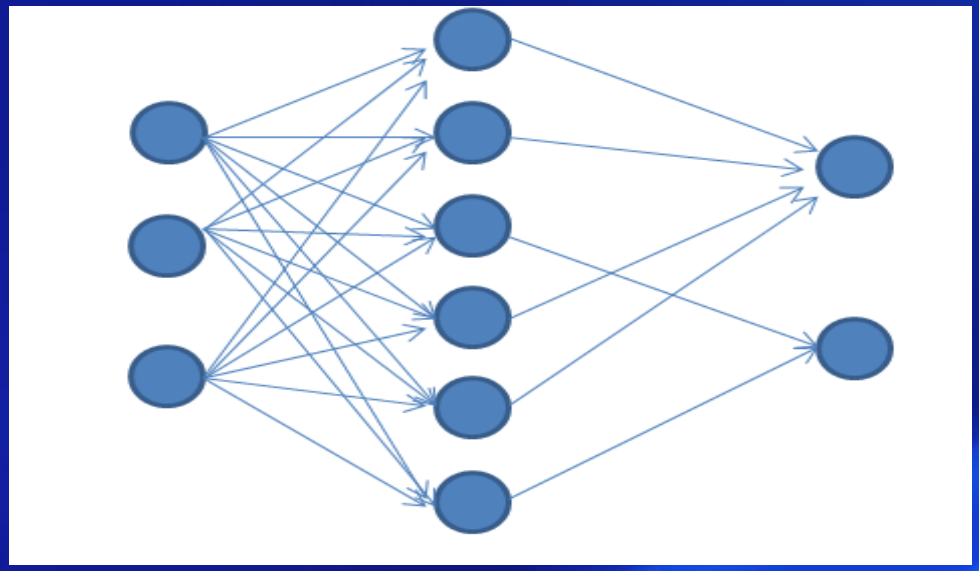
$$|F(x_1, \ldots, x_{m_0}) - f(x_1, \ldots, x_{m_0})| < \varepsilon$$

for all $x_1, x_2, \ldots, x_{m_0}$ that lie in the input space.

- P167. Haykin

Universal Approximation Theorem

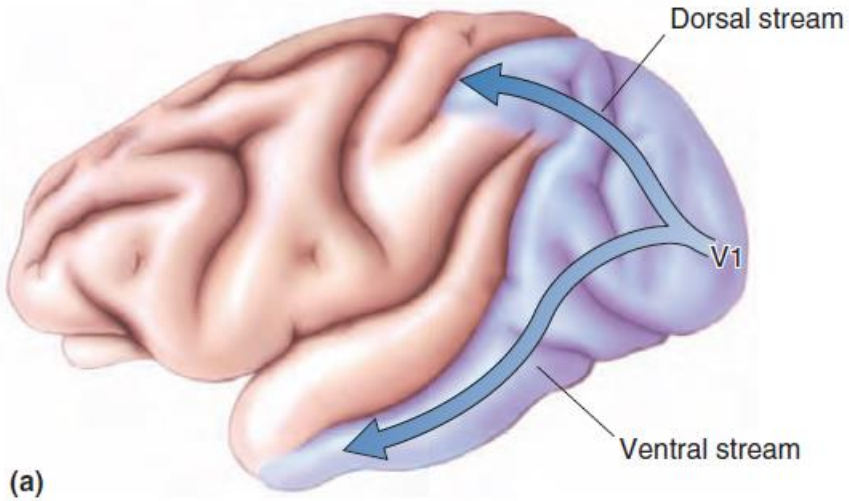
- A feedforward network with a linear output layer and at least one hidden layer with any “squashing” activation function (such as the logistic sigmoid activation function) can approximate any Borel measurable function from one finite-dimensional space to another with any desired non-zero amount of error, provided that the network is given enough hidden units. (Hornik *et al.*, 1989; Cybenko, 1989)
- The derivatives of the feedforward network can also approximate the derivatives of the function arbitrarily well. . (Hornik *et al.*, 1990)

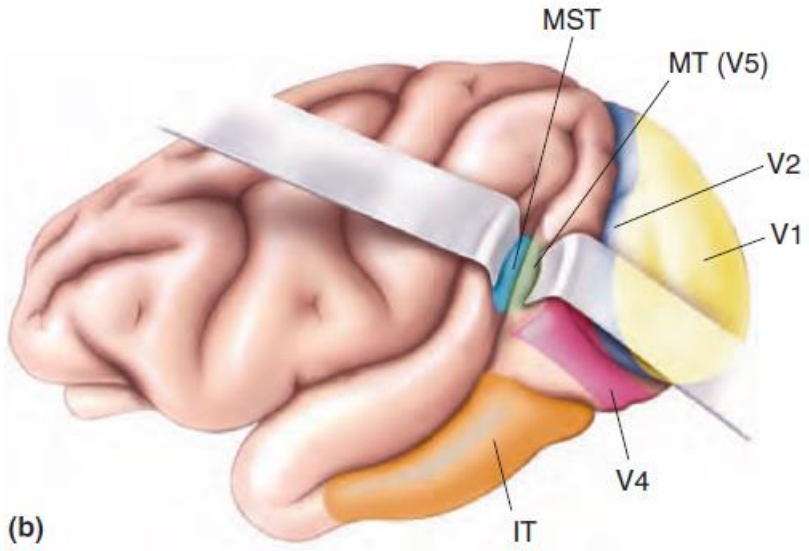


Overfitting

Underfitting

向大脑要答案





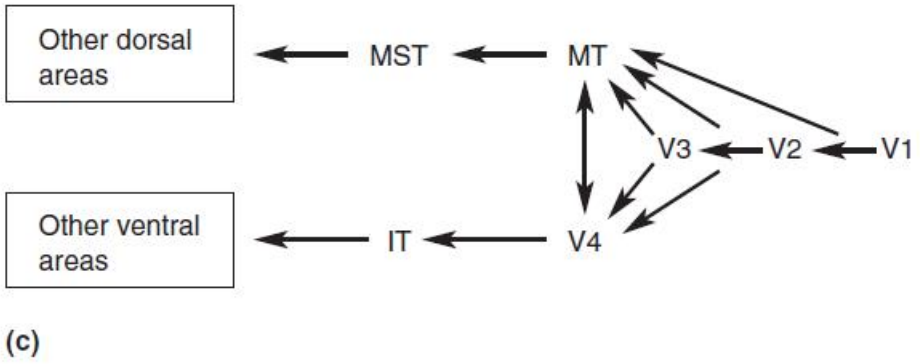
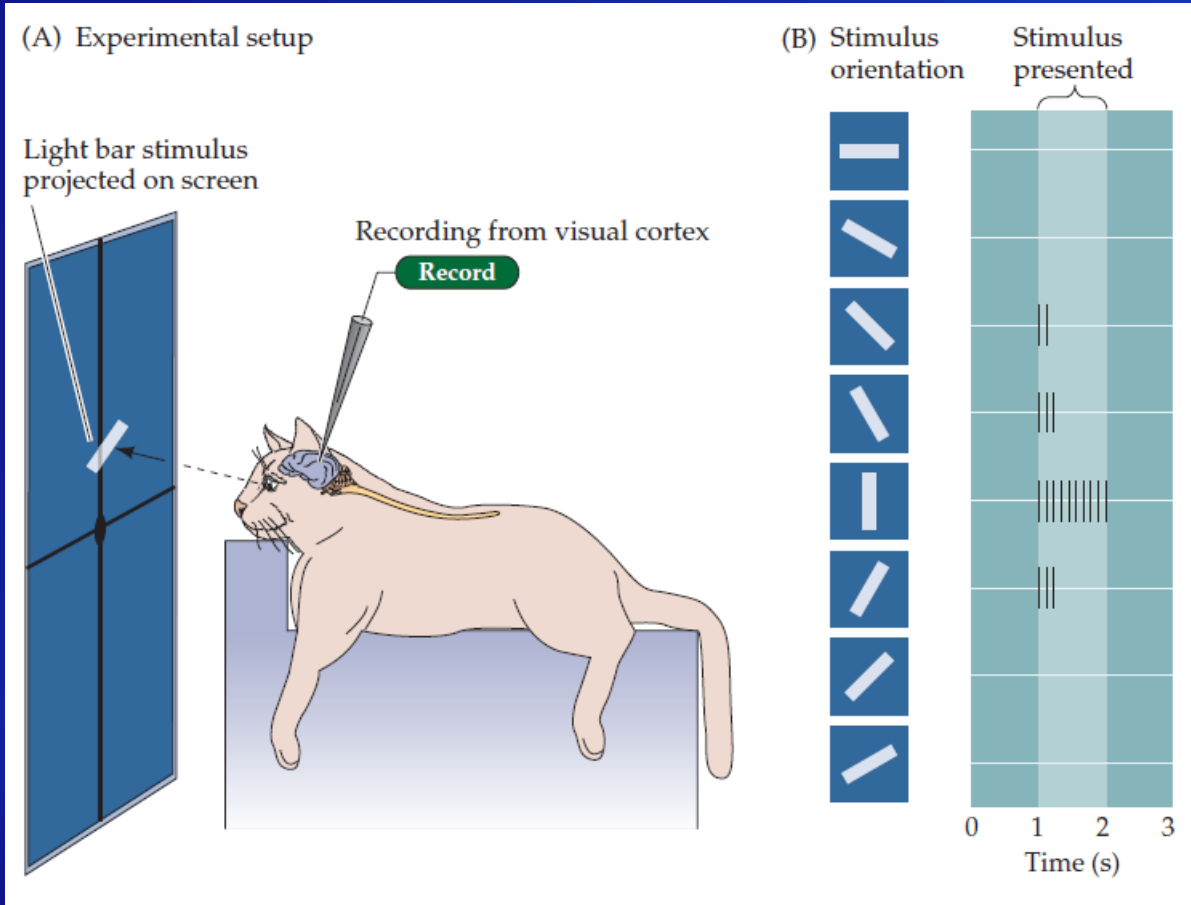


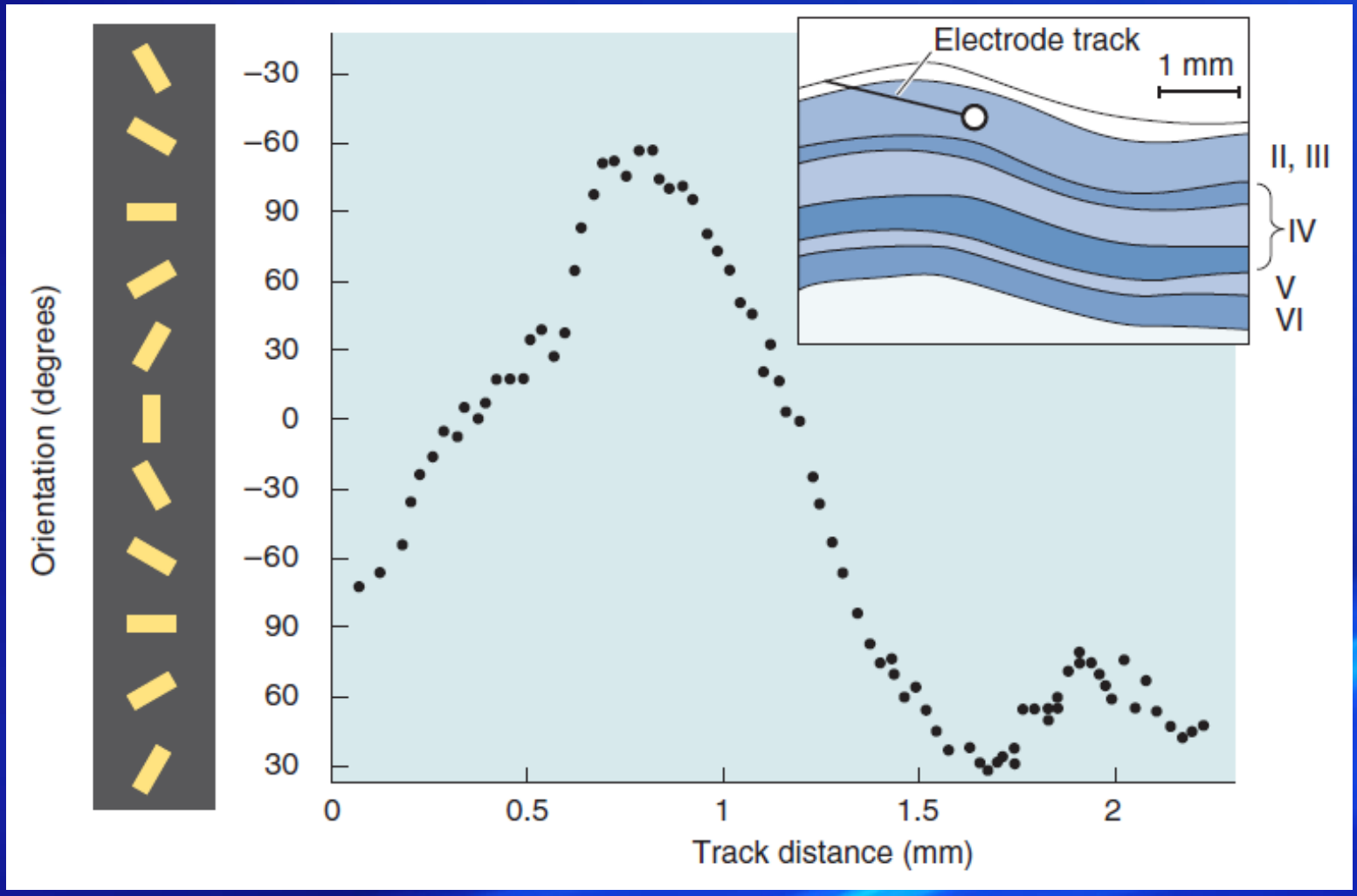
FIGURE 10.27
Beyond striate cortex in the macaque monkey brain. (a) Dorsal and ventral visual processing streams. **(b)** Extrastriate visual areas. **(c)** The flow of information in the dorsal and ventral streams.

Orientation Selectivity in V1

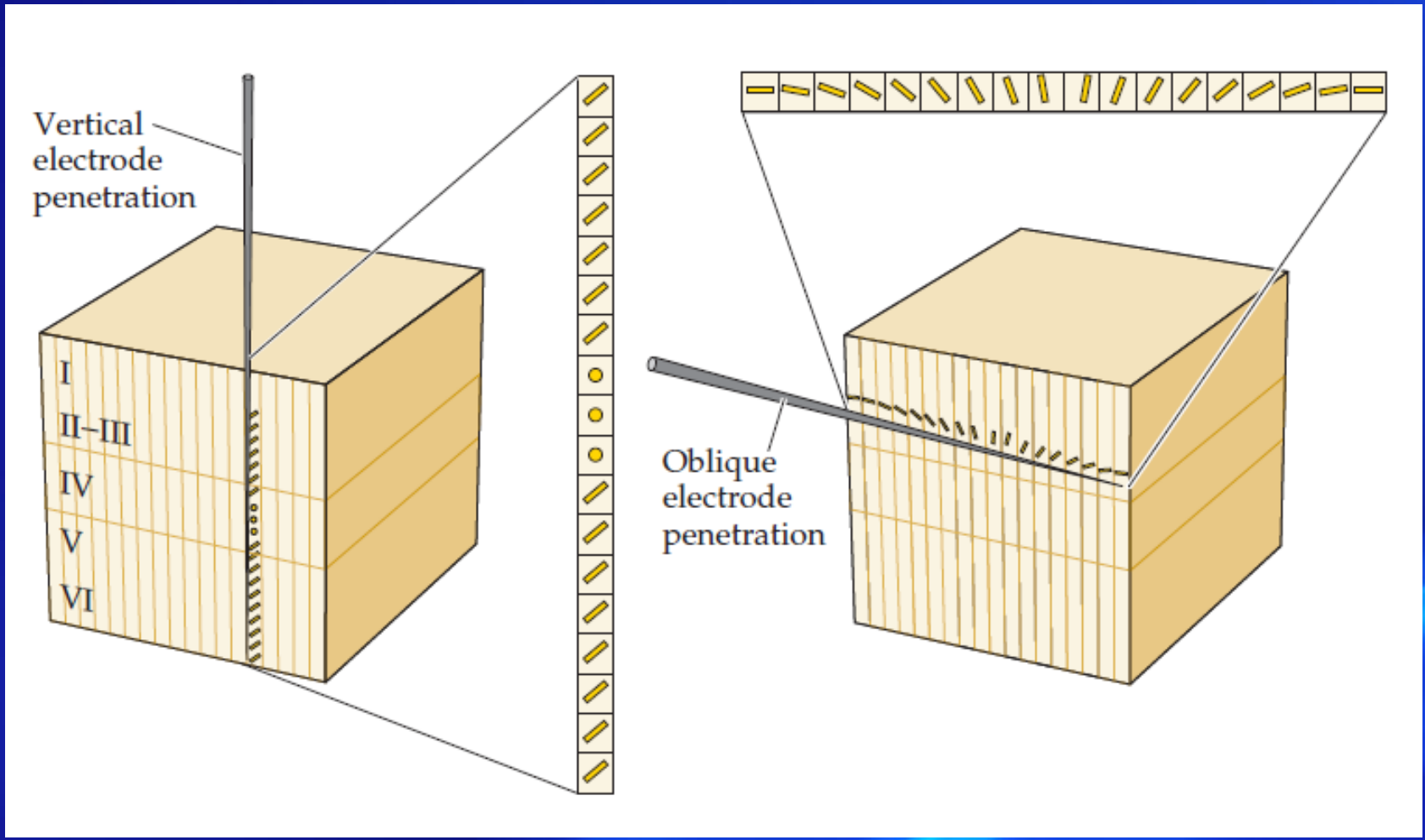
- Outside layer 4C
  - Small spots can elicit responses from many cortical neurons
  - Other stimuli, especially bars, can cause much greater response
  - But the orientation of the bar is critical!



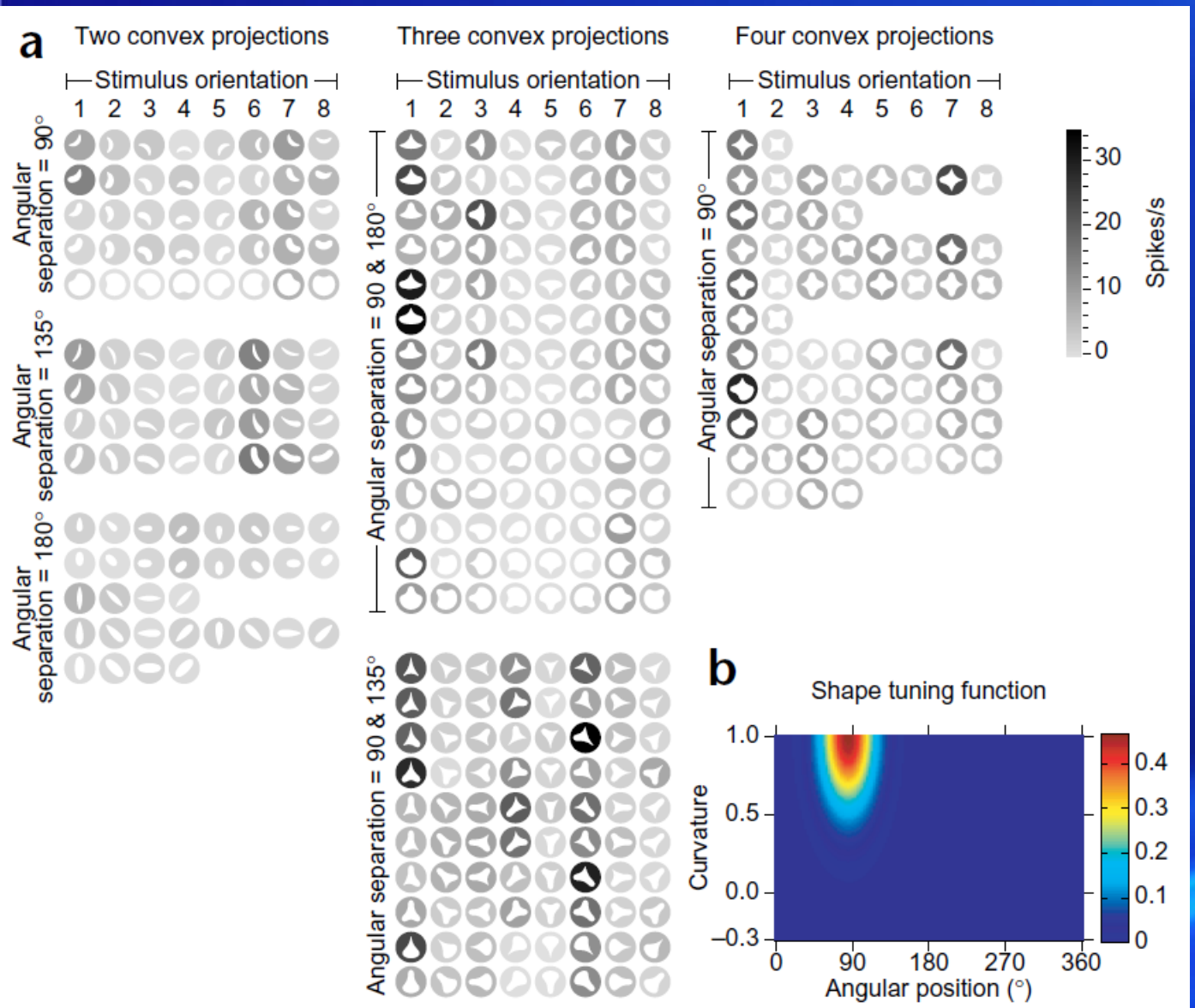
Orientation Preferences across V1



Columnar organization of orientation selectivity



V4-cell response to Curvature and Orientation

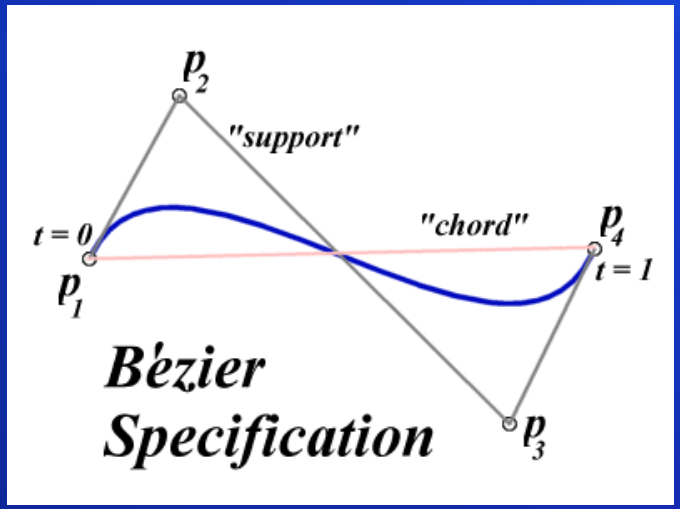


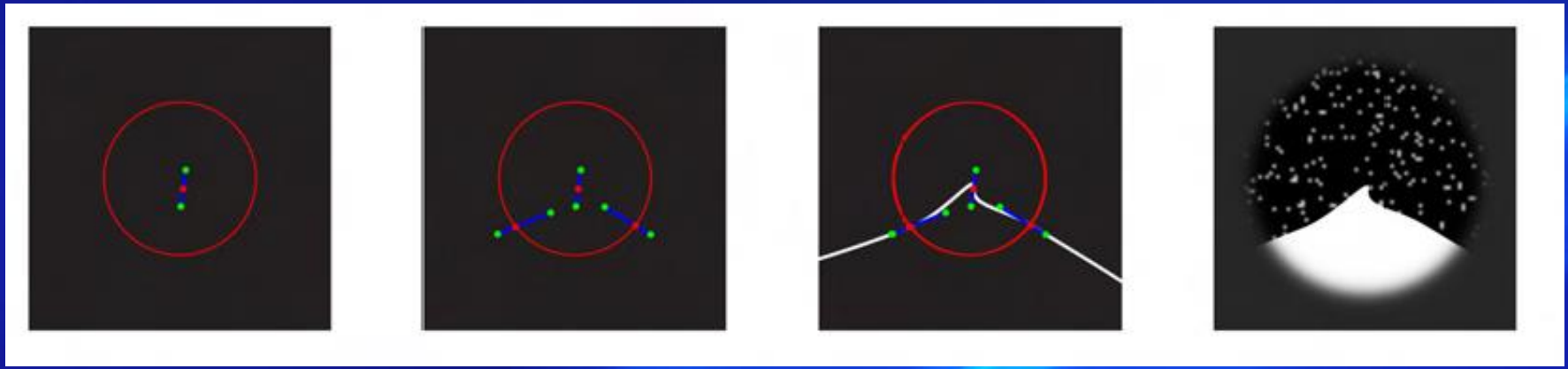
- Curvature
- Orientation

ANITHA PASUPATHY AND CHARLES E. CONNOR, 2001

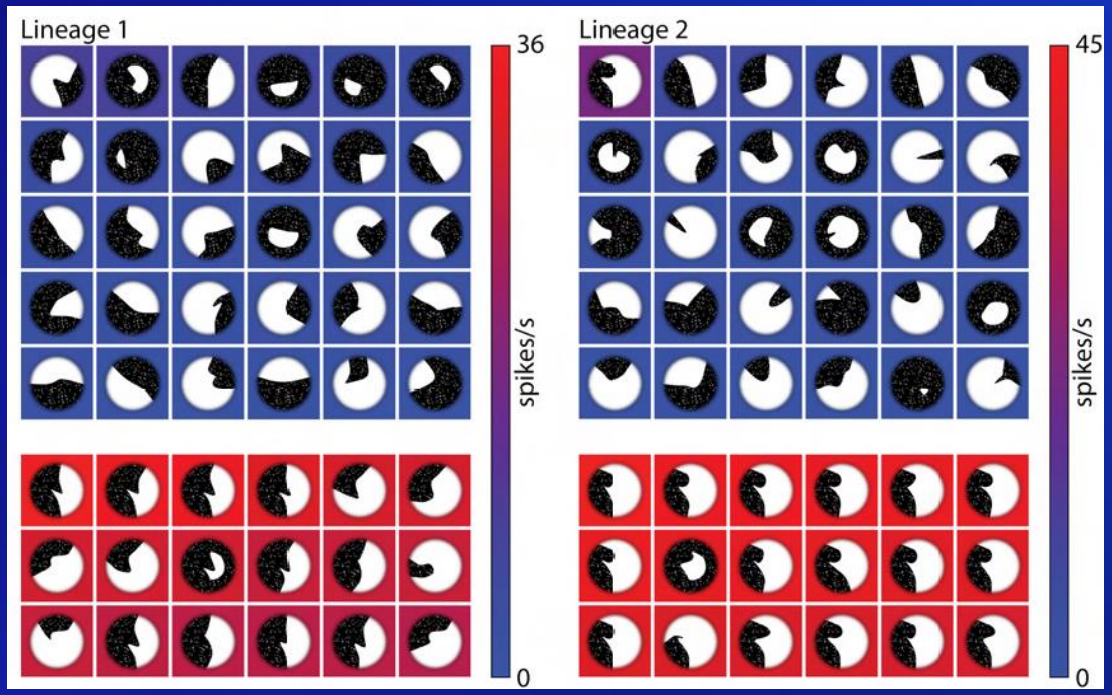
V4 Stimulus Construction

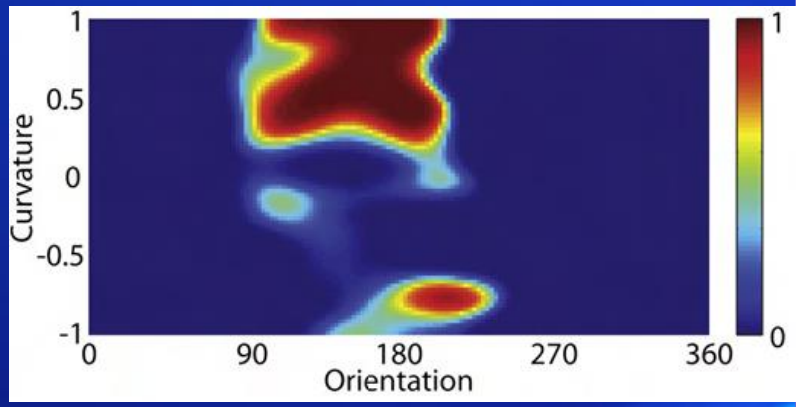
- Based on concatenated Bezier splines.
- Any Bezier spline segment is defined by two terminal points at either end and two intermediate points that determine the path between the endpoints.
- Each of our contours was composed of either two or three (equal probability) Bezier spline segments joined end to end.
- The example illustrated here is based on two spline segments.





One V4 cell response map

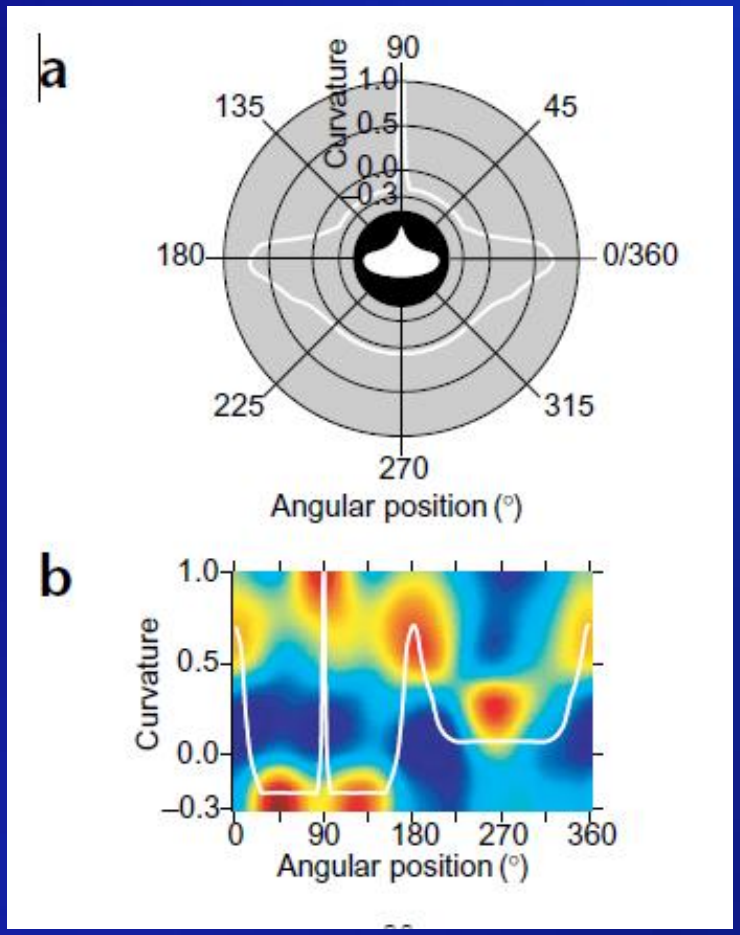




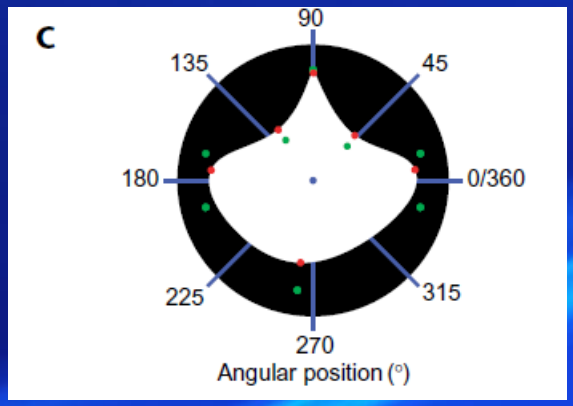
- Curvature
- Orientation

ANITHA PASUPATHY AND CHARLES E. CONNOR, 2001

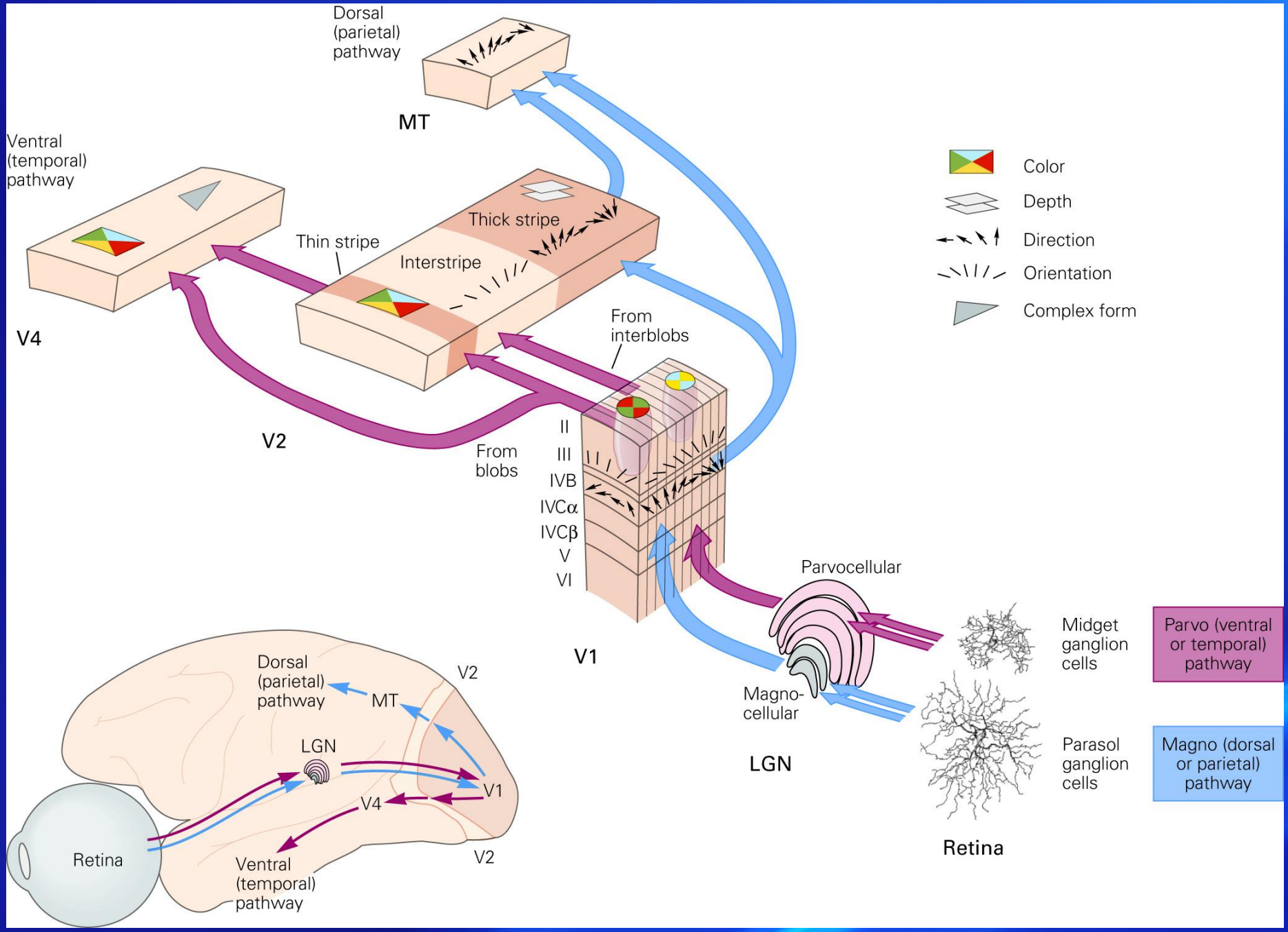
Population Coding for shape



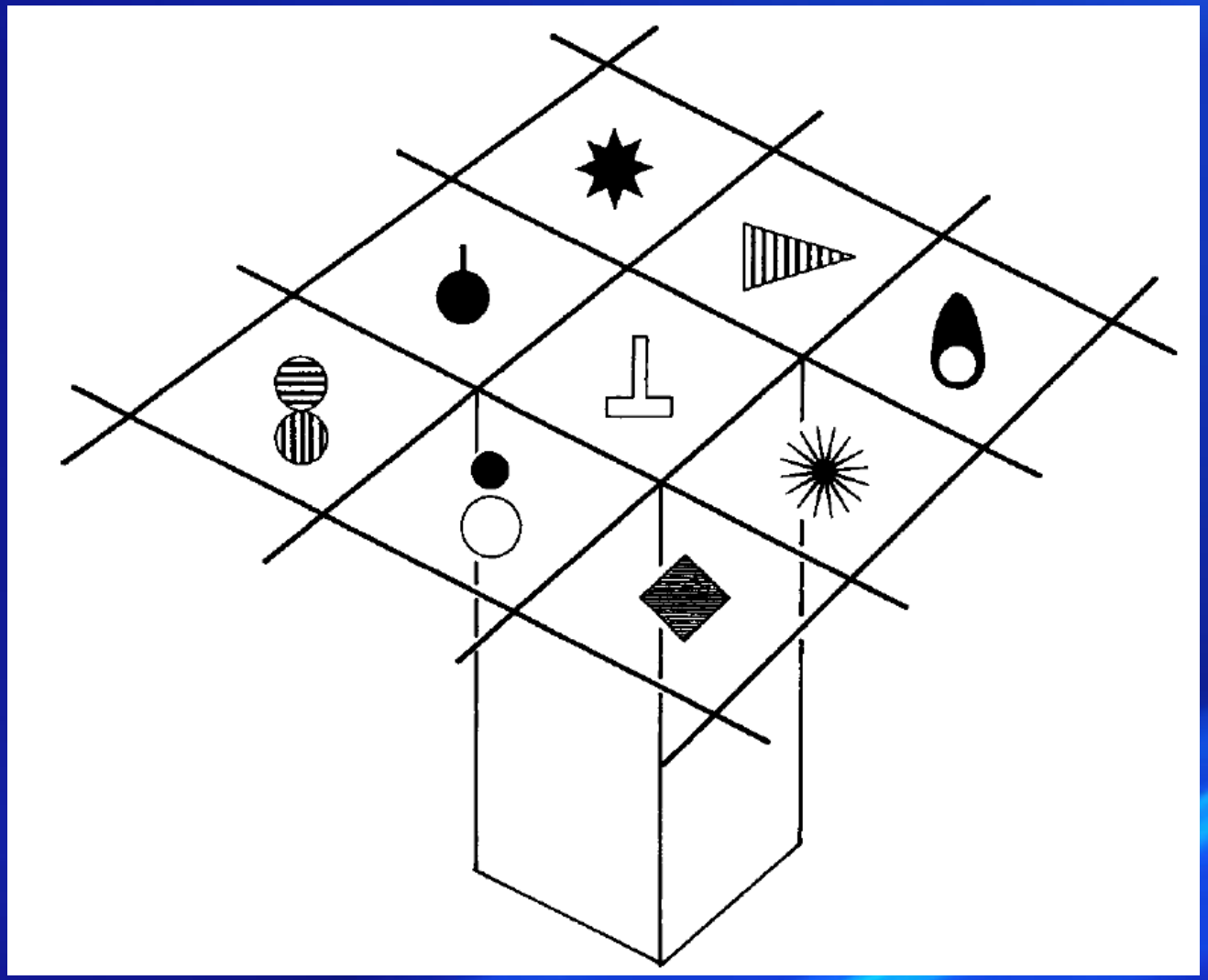
- Angular Position
- Curvature



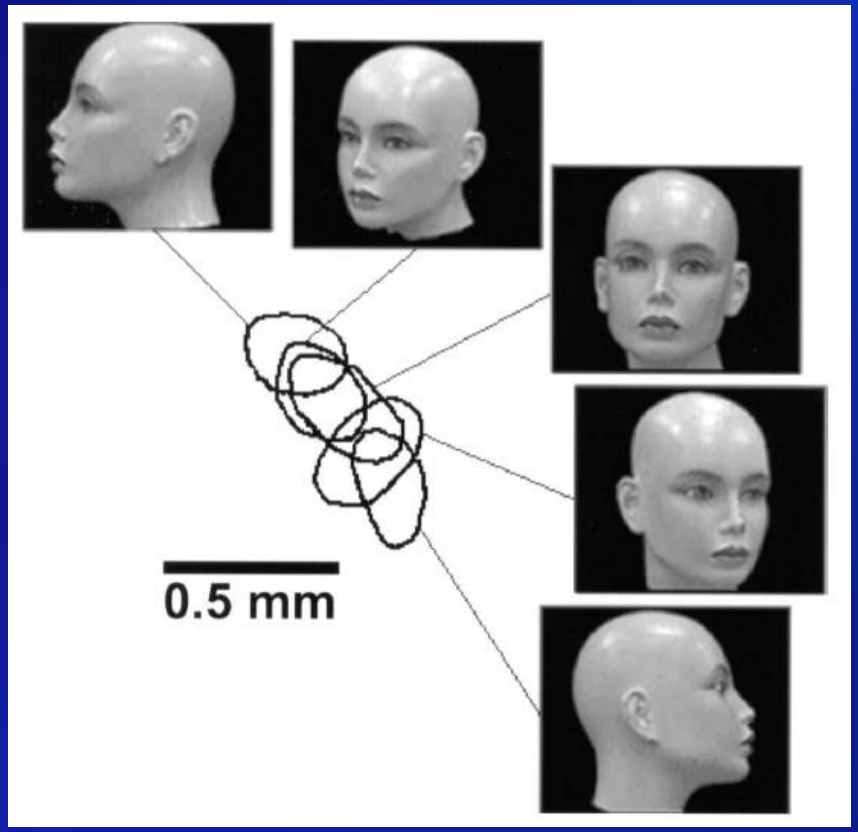
Reconstruction from B

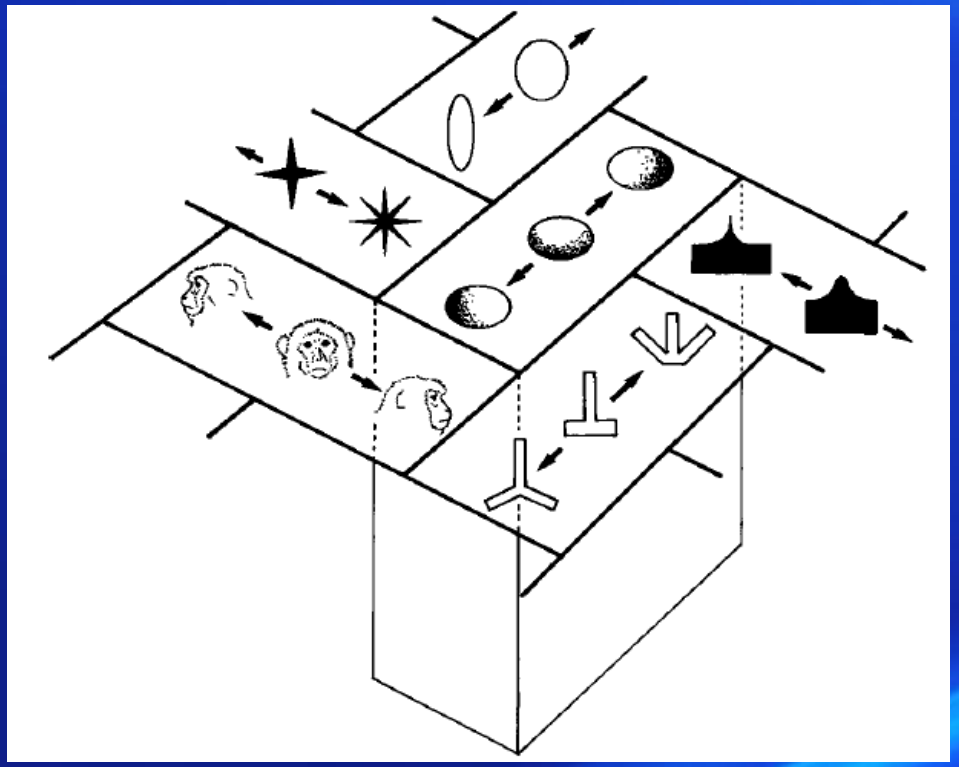

Dorsal (parietal) pathway
MT
Ventral (temporal) pathway
Thick stripe
Thin stripe
Interstripe
V4
V2
From interblobs
From blobs
II
III
IVB
IVCα
IVCβ
V
VI
V1
Color
Depth
Direction
Orientation
Complex form
Parvocellular
Magno-cellular
LGN
Midget ganglion cells
Parasol ganglion cells
Parvo (ventral or temporal) pathway
Magno (dorsal or parietal) pathway
Retina
Dorsal (parietal) pathway
MT
V2
LGN
V1
V4
Retina
Ventral (temporal) pathway
V2

Columnar organization in TE

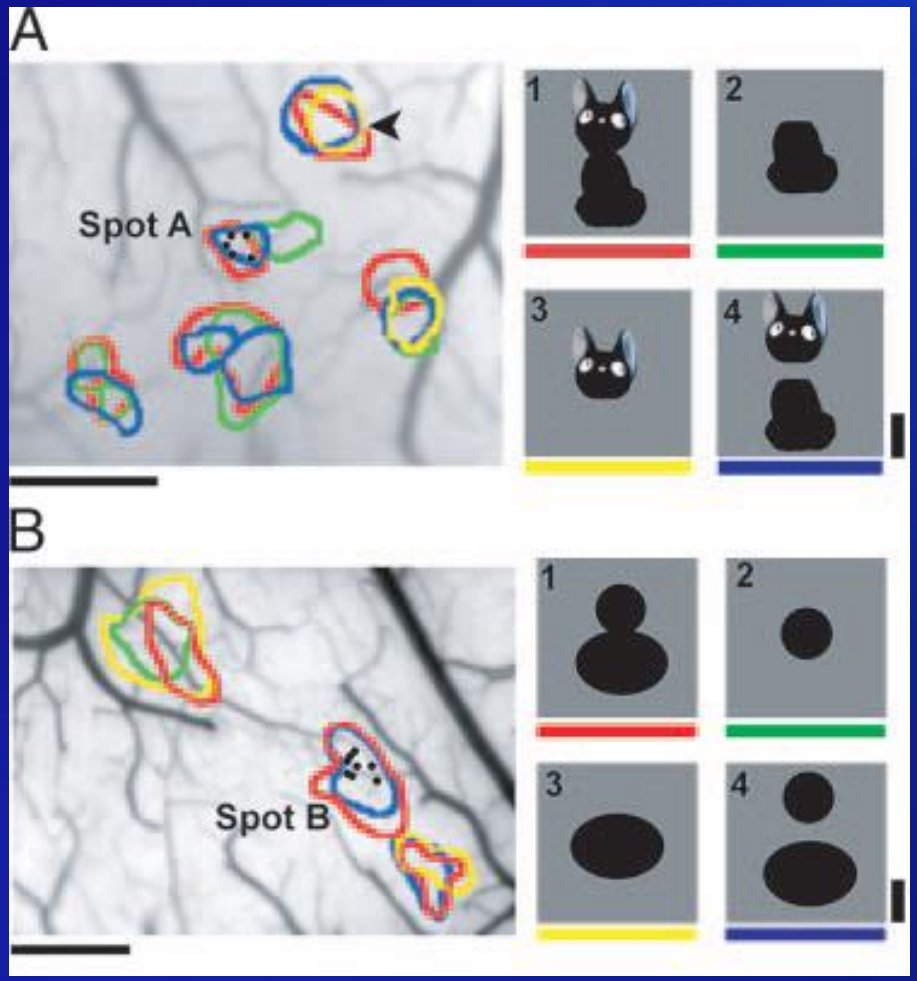


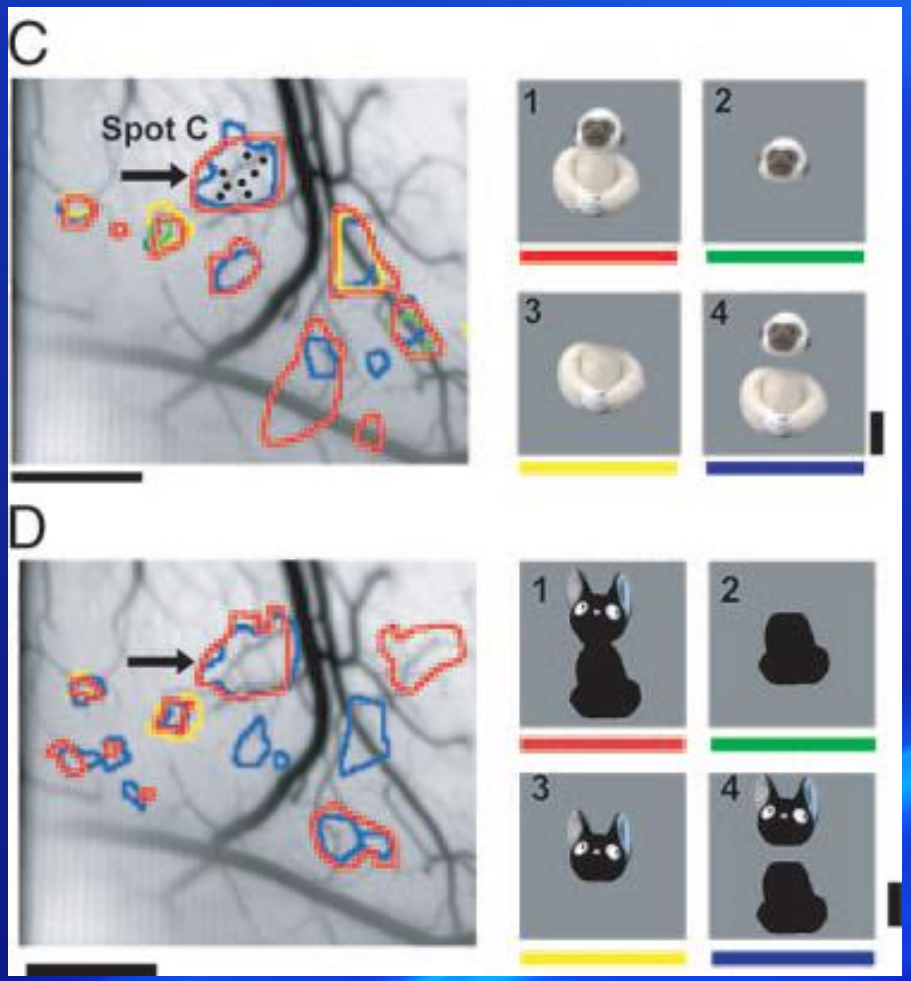
Overlapping





Parts

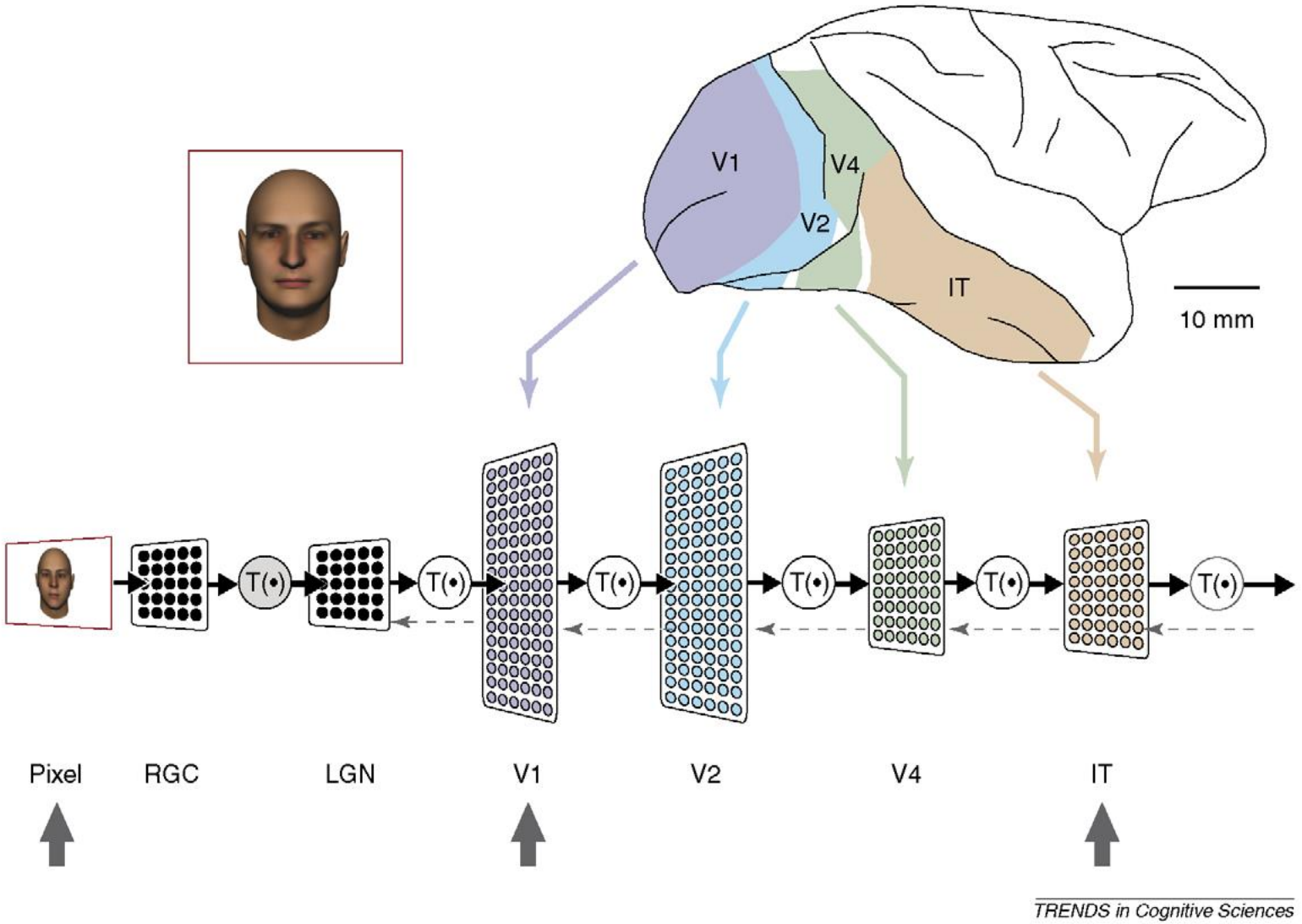




- A， B， C different brains
- C and D same area

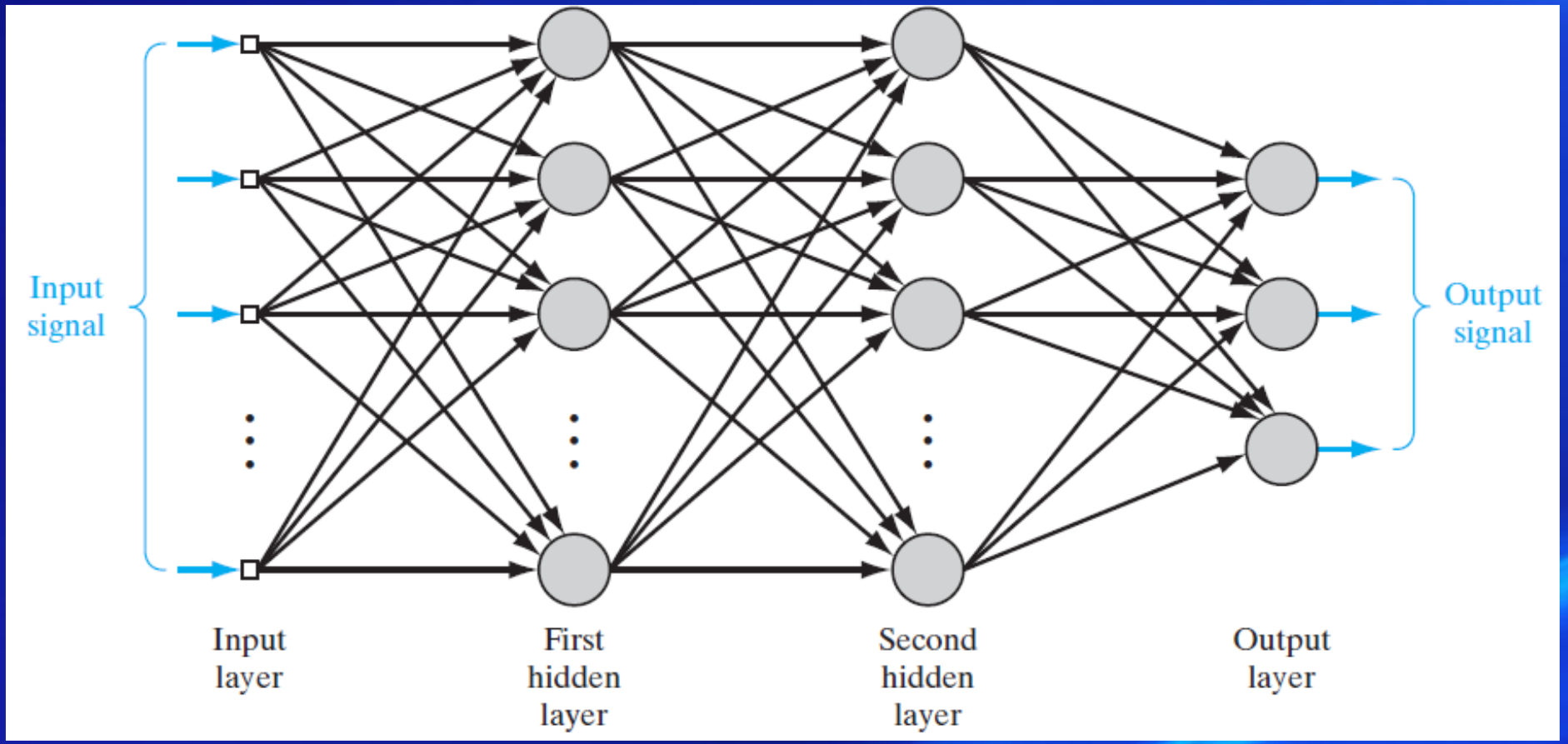
J Neurophysiol 96: 3147–3156, 2006

Inferior Temporal (IT) cortex

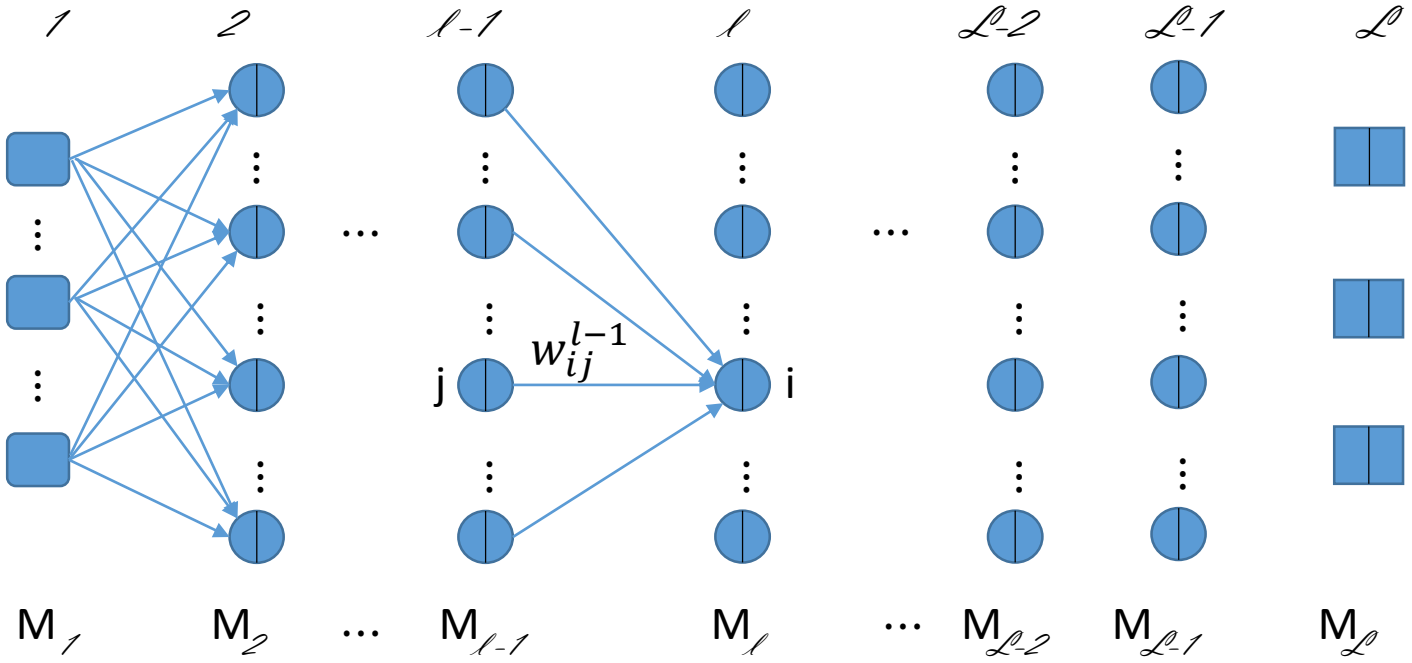


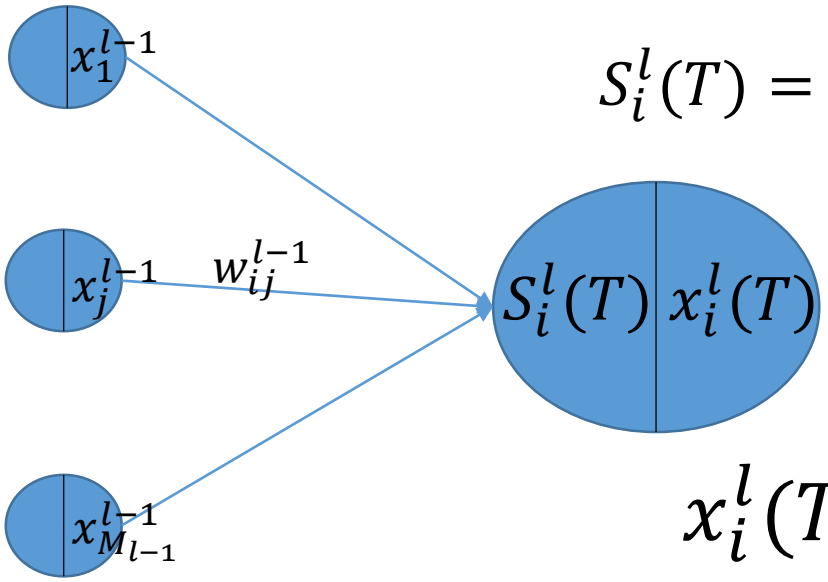
Each Bio-Layer here can mean many layers in ANN with complex local net.

Architectural graph of a MLP with two hidden layers



结构和参数一样，也是对映射函数的一种限制。结构本身也有参数-结构参数





$$S_i^l(T) = \sum_{j=1}^{M_{l-1}} w_{ij}^{l-1} x_j^{l-1}$$

$$x_i^l(T) = \varphi\left(S_i^l(T)\right)$$

目标函数

$$R^n \rightarrow R^m \qquad \{\vec{x}_i, \vec{y}_i, \mid_{i=1, \cdots, N}\}$$

$$\vec{y} = f(\vec{x}) \approx g(\vec{x}; \vec{\theta}) \triangleq \hat{\vec{y}}$$

error energy averaged over the training sample, or empirical risk

$$\mathrm{E}(\vec{\theta}) = \frac{1}{2}\sum_{i=1}^{N} \left\| \hat{\vec{y}}_i - \vec{y}_i \right\|^2 = \frac{1}{2}\sum_{i=1}^{N} \left\| g(\vec{x}_i; \vec{\theta}) - \vec{y}_i \right\|^2$$

$$\hat{\vec{\theta}} = Arg\, \underset{\vec{\theta}}{Min}\, \mathrm{E}(\vec{\theta}) \quad \Leftrightarrow \quad \nabla \mathrm{E}(\hat{\vec{\theta}}) = \vec{0}$$

- $\vec{\theta}$的维数会很高（链接数比神经元），计算量大。
- 样本变化，如增加，计算量会很大！

平均值的计算方法

- 数据　　$\{x_1, x_2, \cdots, x_n, \cdots\}$
- 矫正思想
- Mini-batch

梯度下降 – 找梯度

$$R^n \rightarrow R^m \qquad \{\vec{x}_i, \vec{y}_i, \mid_{i=1, \ \cdots, \ N}\}$$

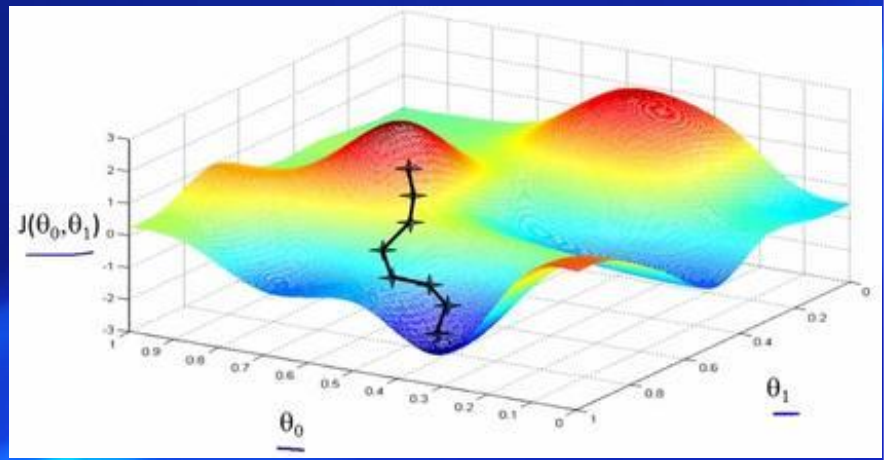
$$\vec{y} = f(\vec{x}) \approx g(\vec{x}; \vec{\theta}) \triangleq \hat{\vec{y}}$$

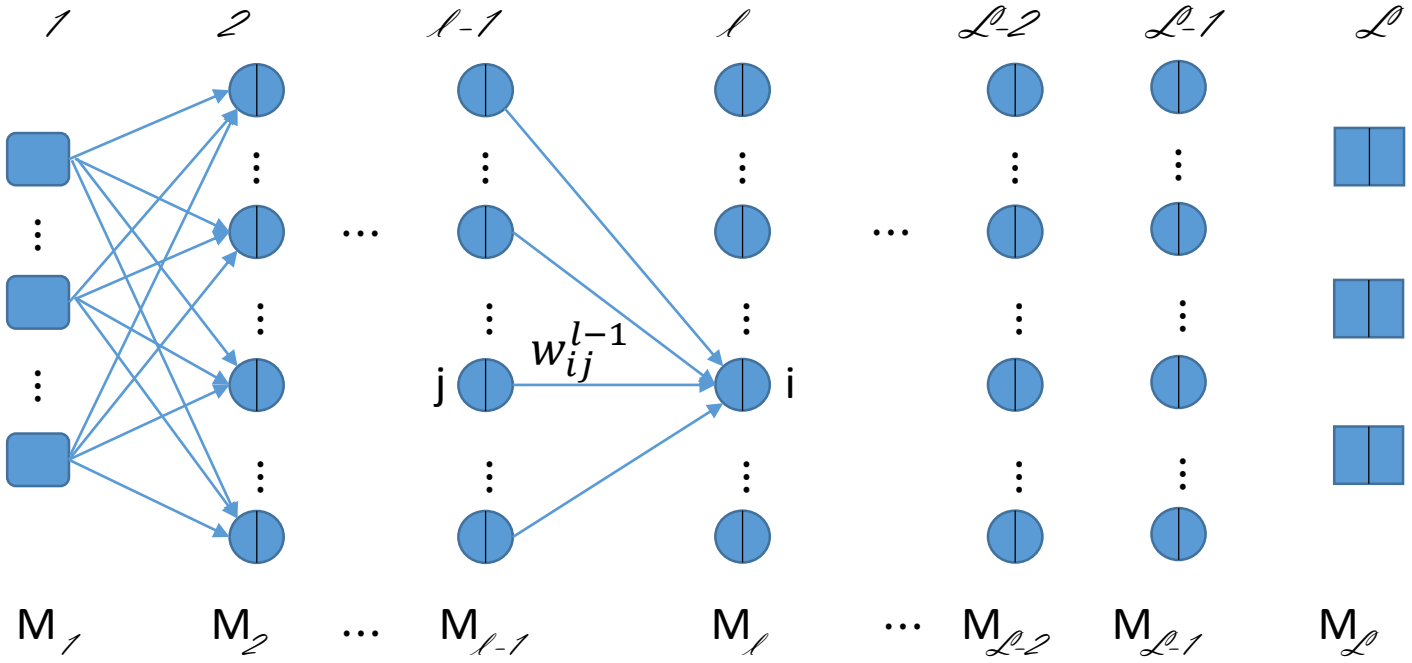
*total **instantaneous** error energy*

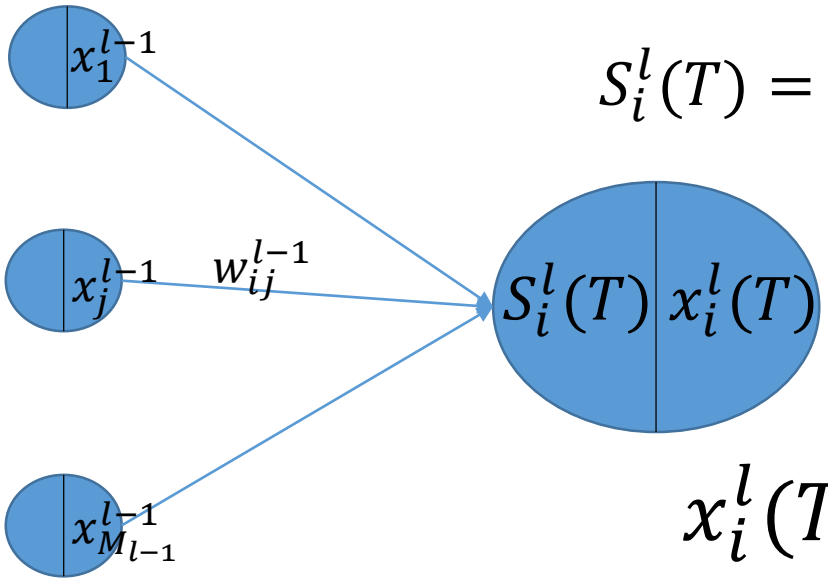
$$\mathrm{E}_i(\vec{\theta}) = \frac{1}{2}\|\hat{\vec{y}}_i - \vec{y}_i\|^2 = \frac{1}{2}\|g(\vec{x}_i; \vec{\theta}) - \vec{y}_i\|^2$$

$$\vec{\theta}(i+1) = \vec{\theta}(i) + \eta \nabla \mathrm{E}_i(\vec{\theta}(i))$$

- η – 学习速度
- $\nabla \mathrm{E}_i(\vec{\theta}(i))$ –如何计算？







$$S_i^l(T) = \sum_{j=1}^{M_{l-1}} w_{ij}^{l-1} x_j^{l-1}$$

$$x_i^l(T) = \varphi\left(S_i^l(T)\right)$$

BP

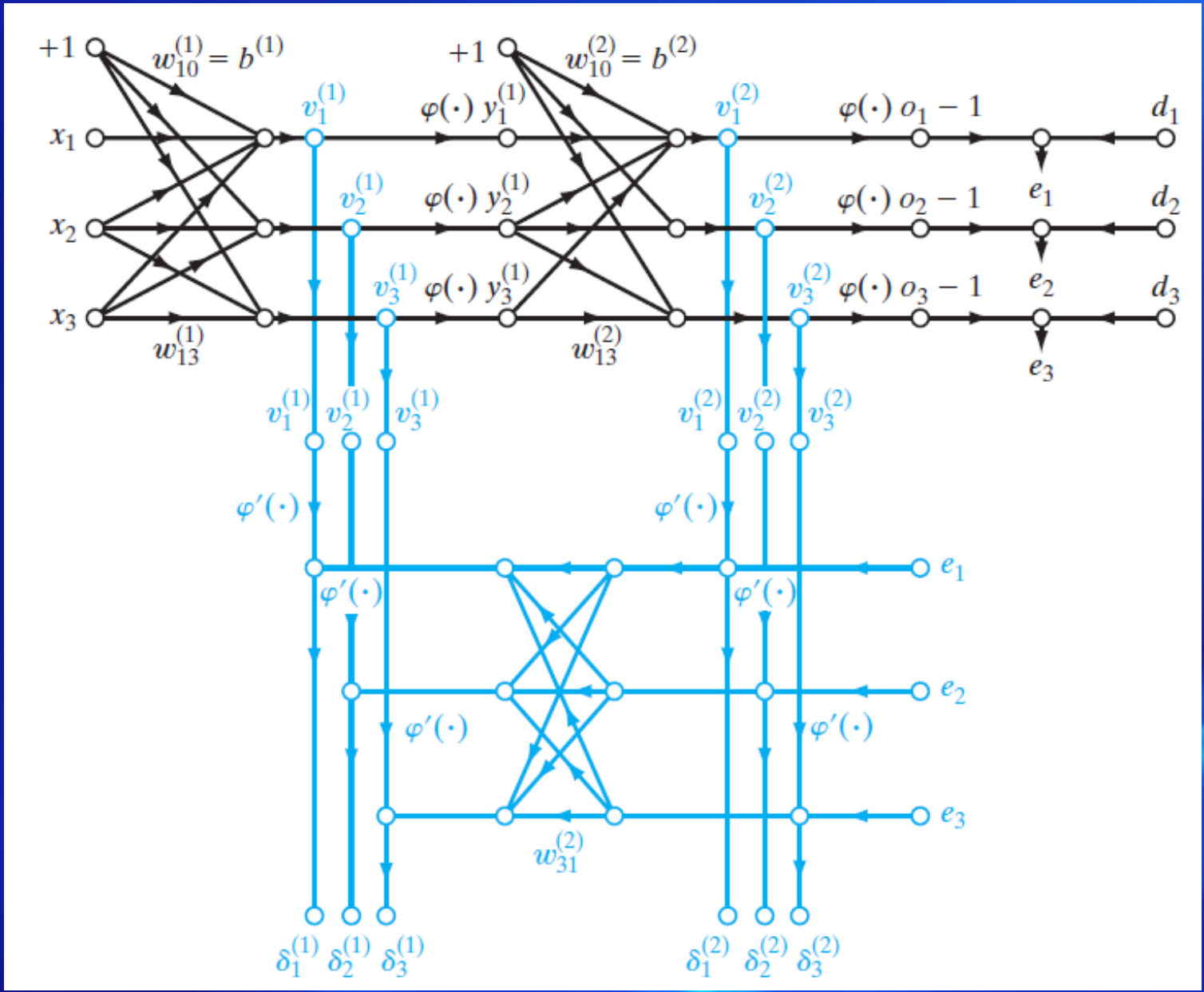
$$\Delta w_{ij}^{(L-1)}(T) = \eta \delta_i^{(L)}(T) x_j^{(L-1)}(T)$$

$$\delta_i^{(L)}(T) = \varphi'(S_i^{(L)}(T))(x_i^{(L)}(T) - y_i(T))$$

$$\Delta w_{ij}^{(L-2)}(T) = \eta \delta_i^{(L-1)}(T) x_j^{(L-2)}(T)$$

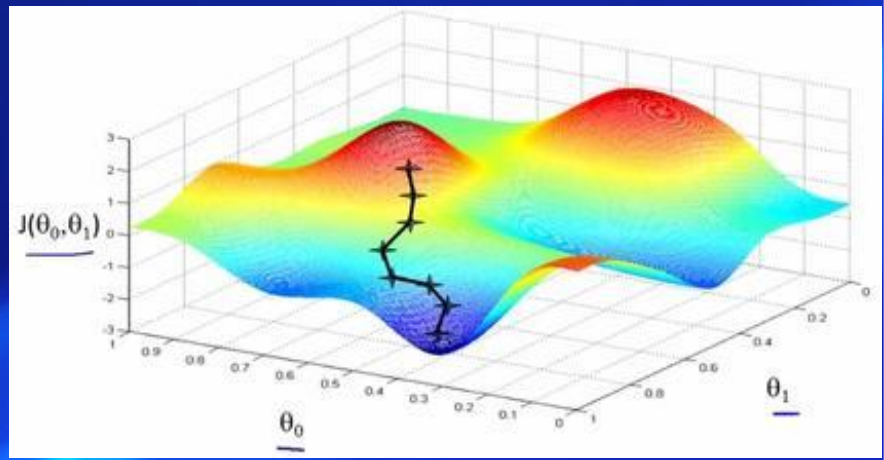
$$\delta_i^{(L-1)}(T) = \varphi'\left(S_i^{(L-1)}(T)\right)\left[\sum_{k=1}^{M^{(L)}} w_{ki}^{(L-1)}(T)\, \delta_k^{(L)}(T)\right]$$

Signal-flow graphical summary of back-propagation learning

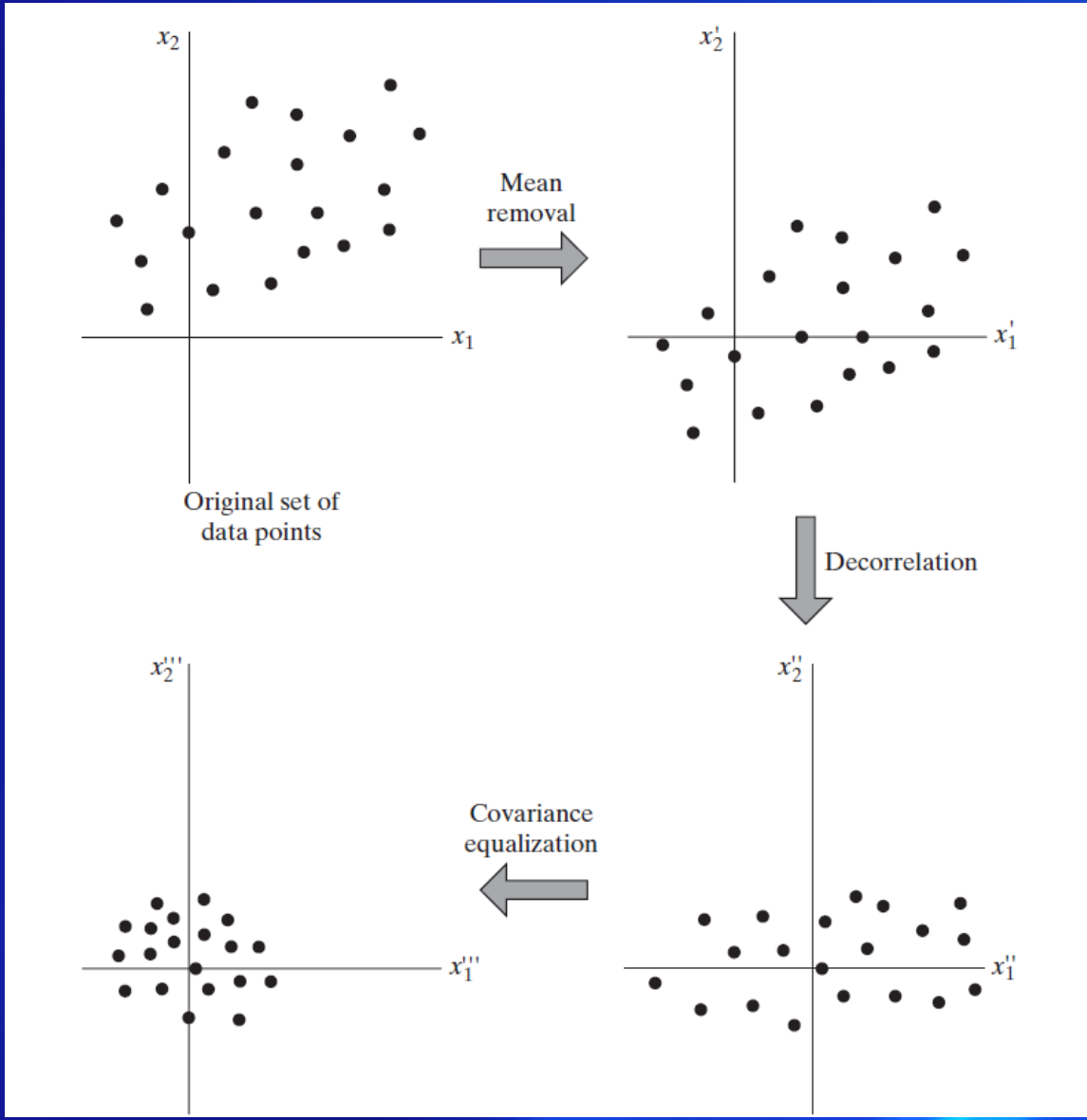


Stopping Criteria

- *When the Euclidean norm of the gradient vector reaches a sufficiently small gradient threshold*
- *When the absolute rate of change in the average squared error per epoch is sufficiently small.*
- When the generalization performance is adequate or when it is apparent that the generalization performance has peaked



Data Preprocess



- Mean zero
- Decorrelation
- Variance Equalization

ANN Algorithm with BP

- Init
  - Data Preprocess
  - Small random number from uniform distribution whose mean is zero and whose variance is chosen to make the standard deviation of the induced local fields of the neurons lie at the transition
- *Loop until meet stop condition*
  - *Presentations of Training Examples in epoch*
  - *Forward Computation*
  - *Backward Computation*

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha[\Delta w_{ji}^{(l)}(n-1)] + \eta\delta_j^{(l)}(n)y_i^{(l-1)}(n)$$

Where η is the learning-rate parameter and α is the momentum constant

Graph

- Graph
  - The *order* of $\mathbb{G}$ is $|\mathbb{V}|$, number of vertices.
  - The *size* of $\mathbb{G}$ is $|\mathbb{E}|$, number of edges.

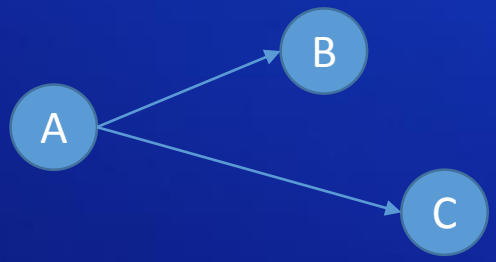
$$\mathbb{G} = \{\mathbb{V}, \mathbb{E}\}$$

顶点和边

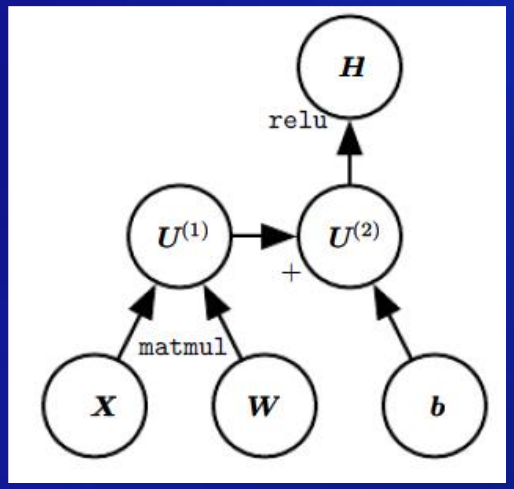
$$\mathbb{V} = \{\vec{x}_1, \vec{x}_2, \cdots \vec{x}_n\}$$

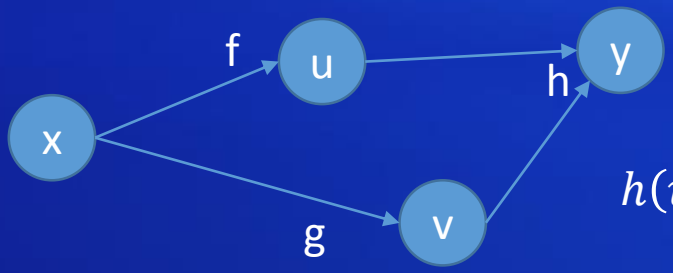
$$\mathbb{E} = \{(1,2), (1,4), (3,5), \cdots (4, n)\}$$

- 有向图
  - 父子关系



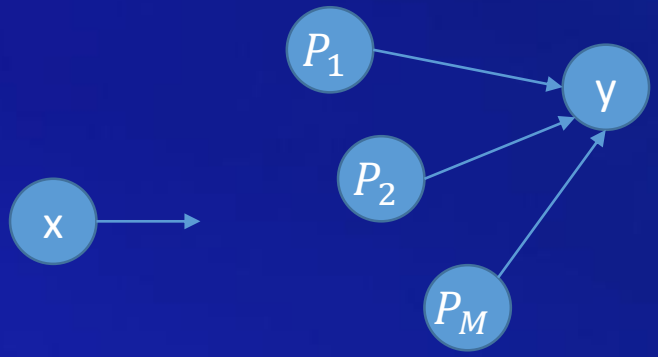
Computational Graphs





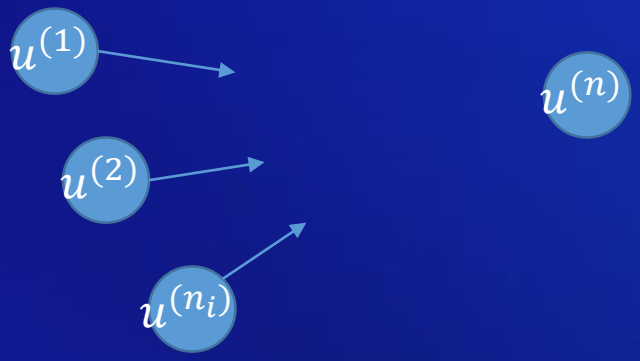
$$h(u,v) = h(f(x), g(x))$$

$$\frac{\partial y}{\partial x} = \frac{\partial h}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial h}{\partial v}\frac{\partial v}{\partial x}$$



$$\frac{\partial y}{\partial x} = \sum_{i=1}^{M} \frac{\partial y}{\partial P_i}\frac{\partial P_i}{\partial x}$$

Forward Algorithm



$$\mathbb{V} = \{u^{(1)}, u^{(2)}, \cdots u^{(n_i)}, \cdots, u^{(n)}\}$$

for $i = 1, \ldots, n_i$ **do**
  $u^{(i)} \leftarrow x_i$
end for
for $i = n_i + 1, \ldots, n$ **do**
  $\mathbb{A}^{(i)} \leftarrow \{u^{(j)} \mid j \in Pa(u^{(i)})\}$
  $u^{(i)} \leftarrow f^{(i)}(\mathbb{A}^{(i)})$
end for
return $u^{(n)}$

BP in general

Run forward propagation (Algorithm 6.1 for this example) to obtain the activations of the network

Initialize `grad_table`, a data structure that will store the derivatives that have been computed. The entry `grad_table`$[u^{(i)}]$ will store the computed value of $\frac{\partial u^{(n)}}{\partial u^{(i)}}$.

`grad_table`$[\partial u^{(n)}] \leftarrow 1$

for $j = n - 1$ down to 1 **do**

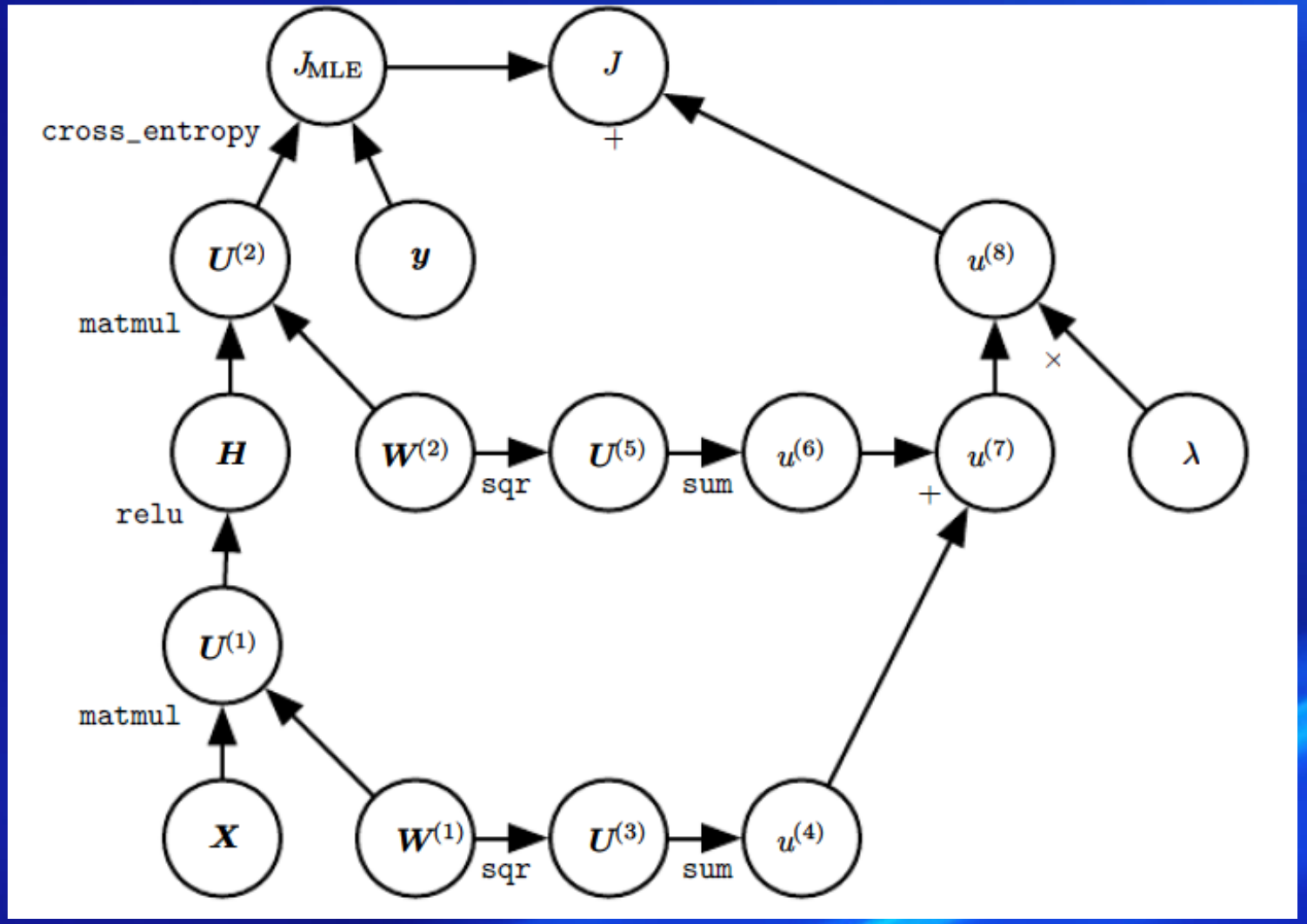
The next line computes $\frac{\partial u^{(n)}}{\partial u^{(j)}} = \sum_{i:j \in Pa(u^{(i)})} \frac{\partial u^{(n)}}{\partial u^{(i)}} \frac{\partial u^{(i)}}{\partial u^{(j)}}$ using stored values:

`grad_table`$[u^{(j)}] \leftarrow \sum_{i:j \in Pa(u^{(i)})}$ `grad_table`$[u^{(i)}] \frac{\partial u^{(i)}}{\partial u^{(j)}}$

end for

return {`grad_table`$[u^{(i)}] \mid i = 1, \ldots, n_i$}

- The computational graph used to compute the cost used to train a single-layer MLP using the cross-entropy loss and weight decay



Open Questions

- How to choose
  - The ANN architecture?
  - The learning rate?
  - The momentum constant?
  - …

Curse of Dimensionality

- Data dimension and Sample size
- *A function defined in high-dimensional space is likely to be much more complex than a function defined in a lower-dimensional space, and those complications are harder to discern.*

Homework

- Program BP algorithm in Python
- Pattern classification (ref. p150-153 on Haykin)
- Change the activation function of hidden layers to ReLU
- The MNIST Database （陈雯婕）