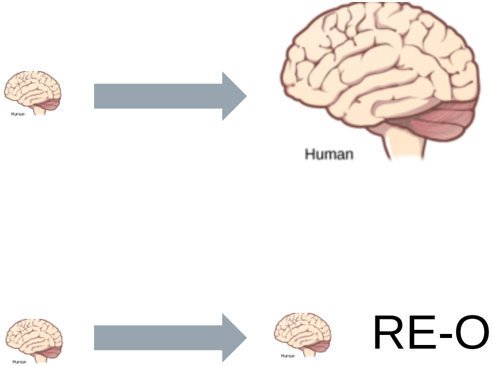
The brain is dynamic

Pexels.com (CC0)



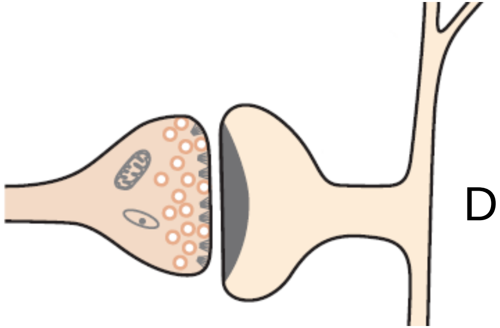
GROWTH

RE-ORGANIZATION

Neural plasticity

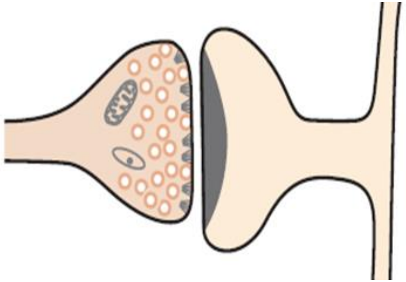
Changing the synapse strength

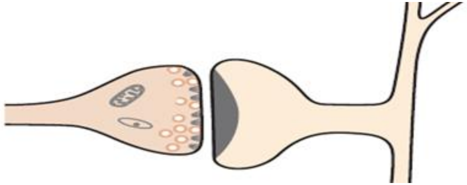
Input from other neuron



Dendrite

pre post



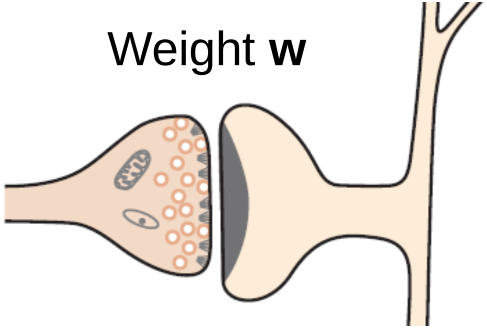


Intensify connection:
Long-term potentiation (LTP)

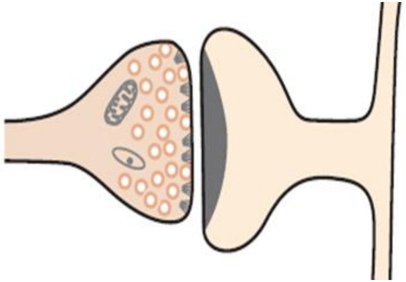
Loosen connection:
Long-term depression (LTD)

Changing the synapse strength

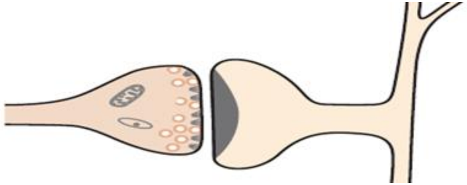
Weight **w**



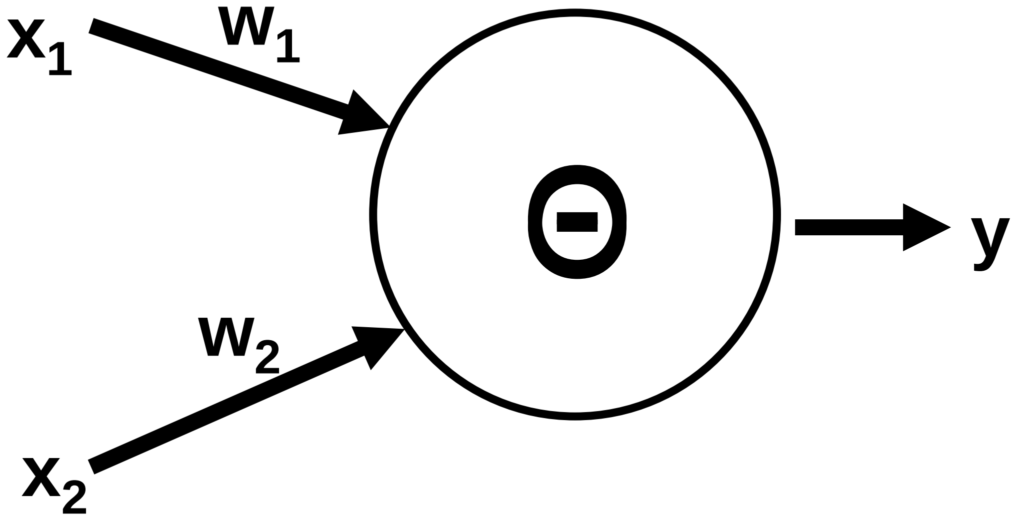
Weight **w**↑



Weight **w**↓

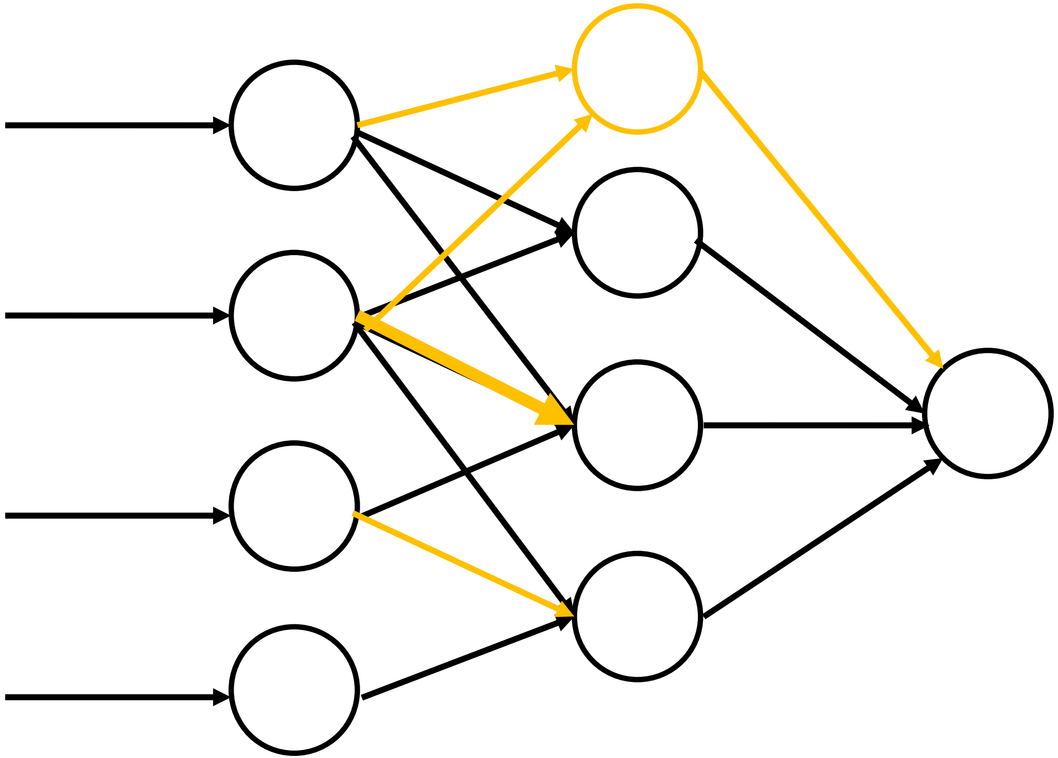


Perceptrons



Frank
Rosenblatt

Multilayer perceptrons



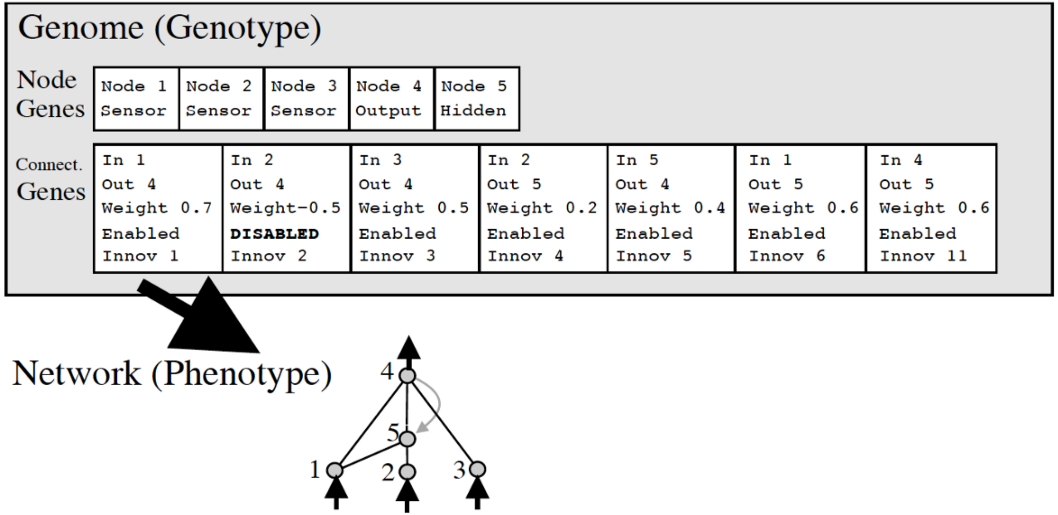
NEAT

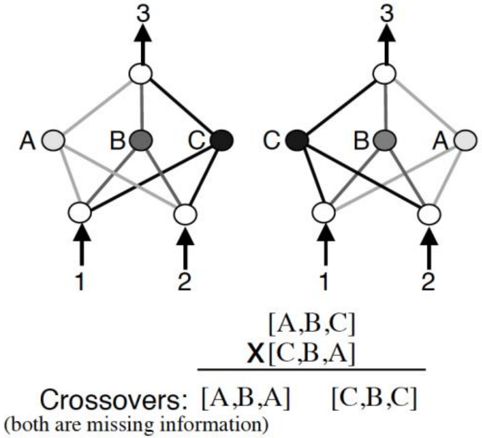
Evolving neural networks through augmenting topologies

KO Stanley, R Miikkulainen - Evolutionary computation, 2002 - MIT Press

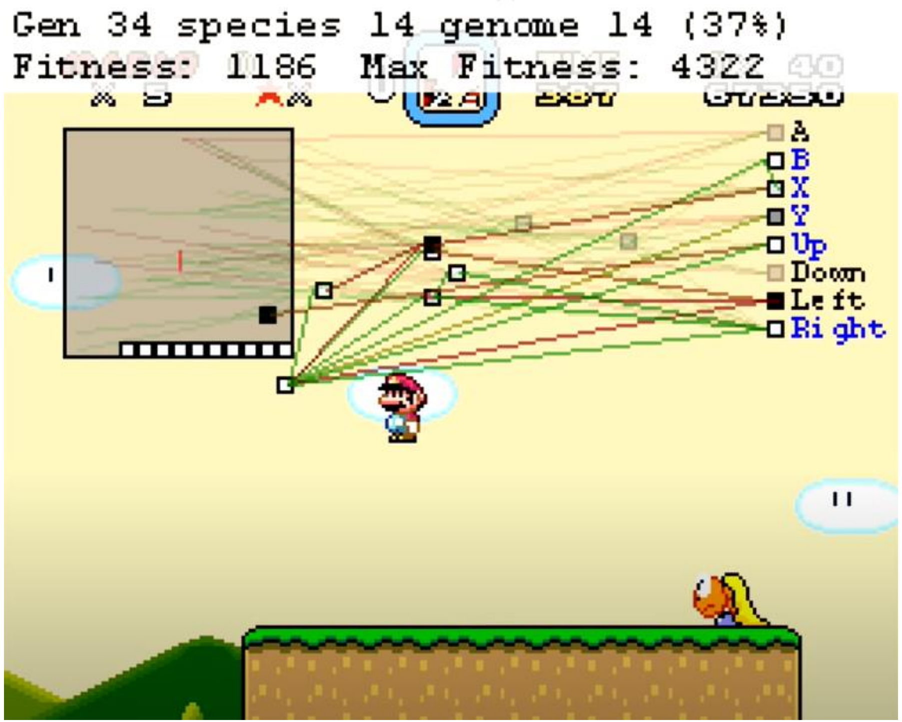
An important question in neuroevolution is how to gain an advantage from evolving neural network topologies along with weights. We present a method, NeuroEvolution of Augmenting Topologies (NEAT), which outperforms the best fixed-topology method on a challenging benchmark reinforcement learning task. We claim that the increased efficiency is due to (1) employing a principled method of crossover of different topologies,(2) protecting structural innovation using speciation, and (3) incrementally growing from minimal structure ...

☆ 99 Zitiert von: 3254 Ähnliche Artikel Alle 23 Versionen





Applied NEAT



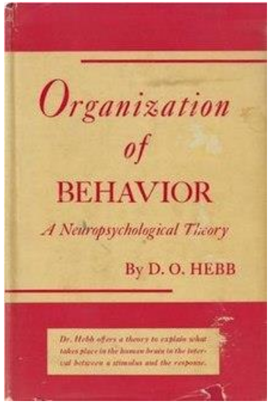
„**MarI/O** is a program made of neural networks and genetic algorithms that kicks butt at Super Mario World.”

© Seth Bling, YouTube https://www.youtube.com/watch?v=qv6UVOQ0F44

How to change weights?

Hebb‘s rule:

„Neurons that fire together, wire together.“

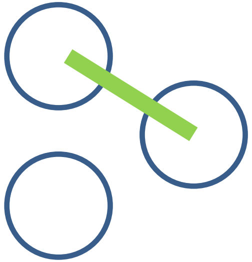


$$w_{ij} = x_i x_j$$

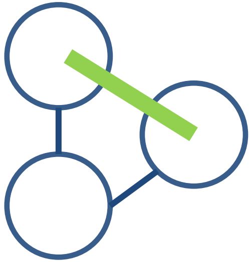
x_i	x_j	w_{ij}
0	0	0
1	0	0
0	1	0
1	1	1

https://en.wikipedia.org/wiki/Organization_of_Behavior; https://en.wikipedia.org/wiki/Donald_O._Hebb

Hebb‘s rule



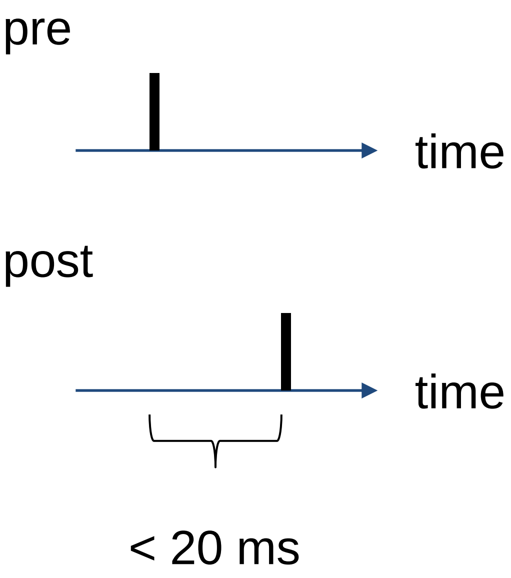
Structural
plasticity



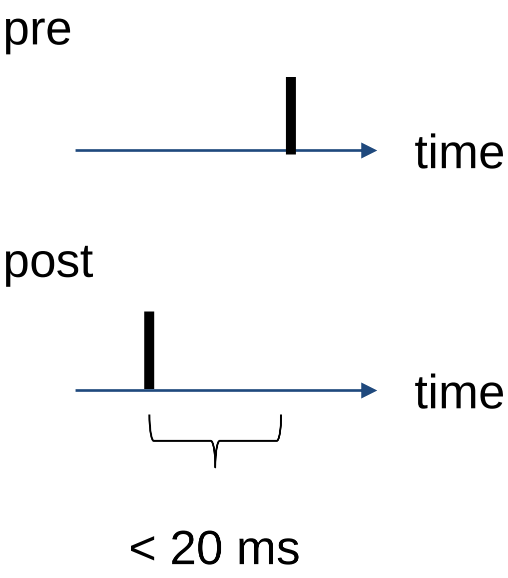
Synaptic
plasticity

Spike timing dependent plasticity (STDP)

Hebb‘s rule – cntd.



Synaptic **w**↑ → **LTP**



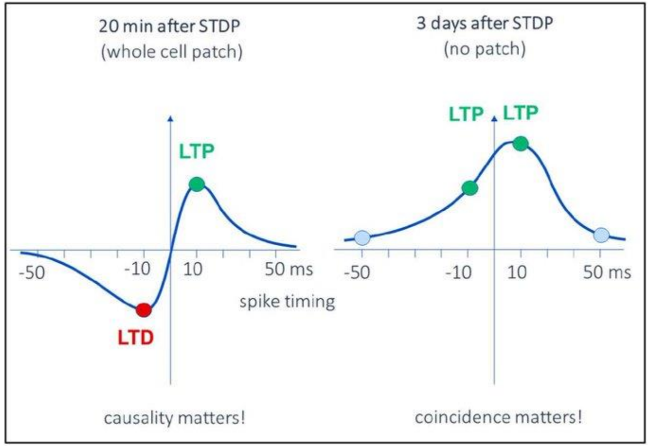
Synaptic **w**↓ → **LTD**

Spike timing dependent plasticity

Recent Research

Spike-timing-dependent plasticity rewards synchrony rather than causality

Margarita Anisimova[1], Bas van Bommel[1], Marina Mikhaylova[1,2], J. Simon Wiegert, Thomas G. Oertner[1*], Christine E. Gee[1,3*]



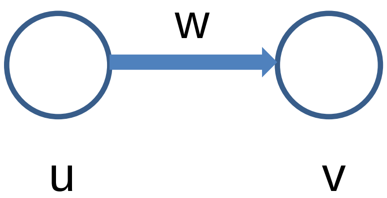
https://doi.org/10.1101/863365

“Our results confirm that neurons wire together if they fire together, but suggest that synaptic depression after anti-causal activation (tLTD) is a transient phenomenon.”

biorxiv, 2021

Synaptic plasticity rules

$$v = w \cdot u$$



$$v = \boldsymbol{w} \cdot \boldsymbol{u}^{\boldsymbol{T}}$$

The basic Hebb's rule:

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

$$\tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{w} \cdot \boldsymbol{u} \cdot \boldsymbol{u}^{\boldsymbol{T}}$$

Correlation matrix

$$\tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{Q} \cdot \boldsymbol{w}^{\boldsymbol{T}}$$

Correlation-based plasticity rule

$$\boldsymbol{w} \rightarrow \boldsymbol{w} + \epsilon \boldsymbol{Q} \cdot \boldsymbol{w}^{\boldsymbol{T}} \qquad \epsilon = \frac{1}{\tau_w}$$

Synaptic plasticity rules

Correlation-based plasticity rule

$$\tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{Q} \cdot \boldsymbol{w}^{\boldsymbol{T}}$$

→ Basic Hebb only allows LTP

Postsynaptic LTD/LTP switch

$$\tau_w \frac{d\boldsymbol{w}}{dt} = (v - \theta_v) \cdot \boldsymbol{u}$$

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot (\boldsymbol{u} - \boldsymbol{\theta_u})$$

Presynaptic LTD/LTP switch

Covariance matrix

$$v = \boldsymbol{w} \cdot \boldsymbol{u}^{\boldsymbol{T}} \Rightarrow \tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{w} \cdot \boldsymbol{u} \cdot (\boldsymbol{u} - \boldsymbol{\theta_u})^{\boldsymbol{T}} \Rightarrow \tau_w \frac{d\boldsymbol{w}}{dt} = \boldsymbol{C} \cdot \boldsymbol{w}^{\boldsymbol{T}}$$

Covariance-based plasticity rule

BCM rule

Hebbian learning suffers from instability

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u} \cdot (v - \theta_v)$$ If constant → unstable

➔ Threshold of postsynaptic activity that determins if synapse is strengthened or weakened.

Adapt threshold θ_v: $$\tau_\theta \frac{d\theta_v}{dt} = v^2 - \theta_v$$

Synaptic Normalization

$$\tau_\theta \frac{d\theta_v}{dt} = v^2 - \theta_v$$

→ Stabilize weights through postsynaptic activity

Can we use penalty terms directly on the weight vector?

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u} - \frac{v(\boldsymbol{n} \cdot \boldsymbol{u}^T)\boldsymbol{n}}{N_u}$$

Normalize by subtracting the same quantity

Oja‘s rule

$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot \boldsymbol{u} - \alpha \cdot v^2 \cdot \boldsymbol{w} \qquad \alpha > 0$$

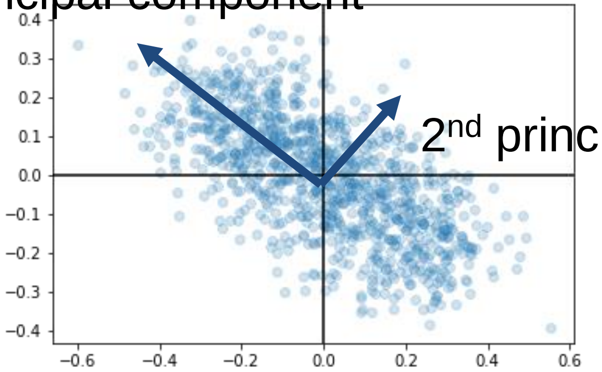
$$\tau_w \frac{d\boldsymbol{w}}{dt} = v \cdot (\boldsymbol{u} - \alpha \cdot v \cdot \boldsymbol{w})$$

Oja‘s rule

Unsupervised learning

PRINCIPAL COMPONENT ANALYSIS (PCA)

1st principal component



2nd principal component

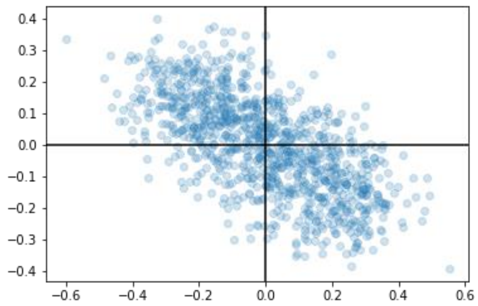
Oja‘s rule

Simulation

Steps:

1. Data generation
2. Variable initialization
3. Iterate through data
    1. Compute pre-synaptic input
    2. Compute post-synaptic activation
    3. Compute Δw and update
4. Plot result

```python
# That you see the same what I see
np.random.seed(42)
N = 1000

# Generate random data
x = np.linspace(-.3, .3, N)
np.random.shuffle(x)
y = -.7 * x

x += np.random.randn(x.size) / 10
y += np.random.randn(y.size) / 10

```



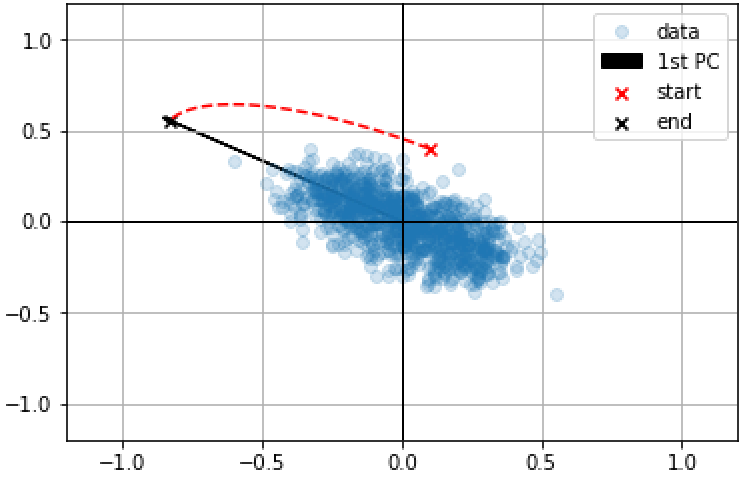

```python
# Initialize some random weights
w = np.array([0.1, 0.4])
# # Training iterations
N = 1000
eta = 0.1
ws = []

# Training
for i in range(N):
    rPre  = np.asarray([x[ix], y[ix]])
    rPost = w @ rPre
    w = w + eta * rPost * (rPre - rPost * w)
```

Oja's rule

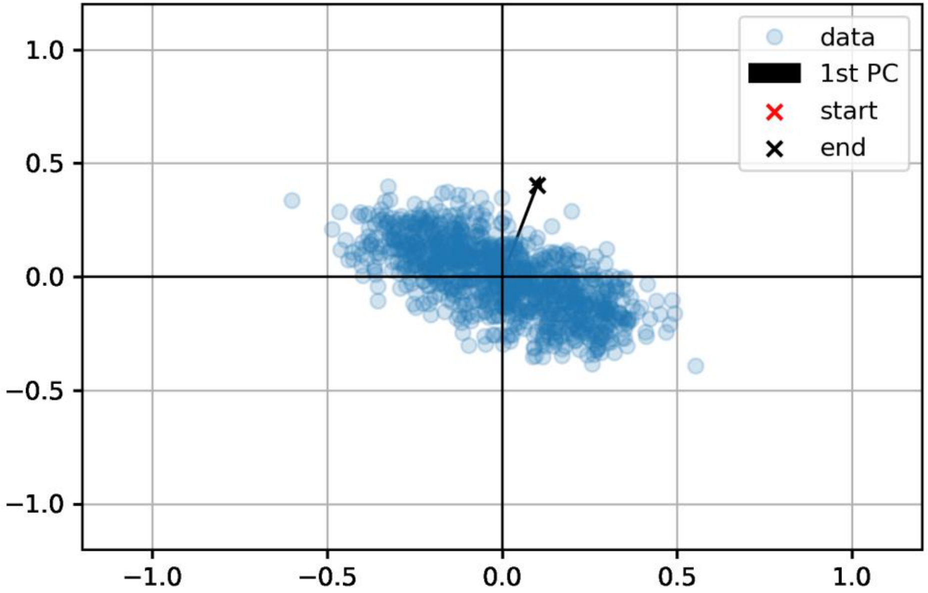
Simulation

Steps:

1. Data generation
2. Variable initialization
3. Iterate through data
   1. Compute pre-synaptic input
   2. Compute post-synaptic activation
   3. Compute Δw and update
4. Plot result

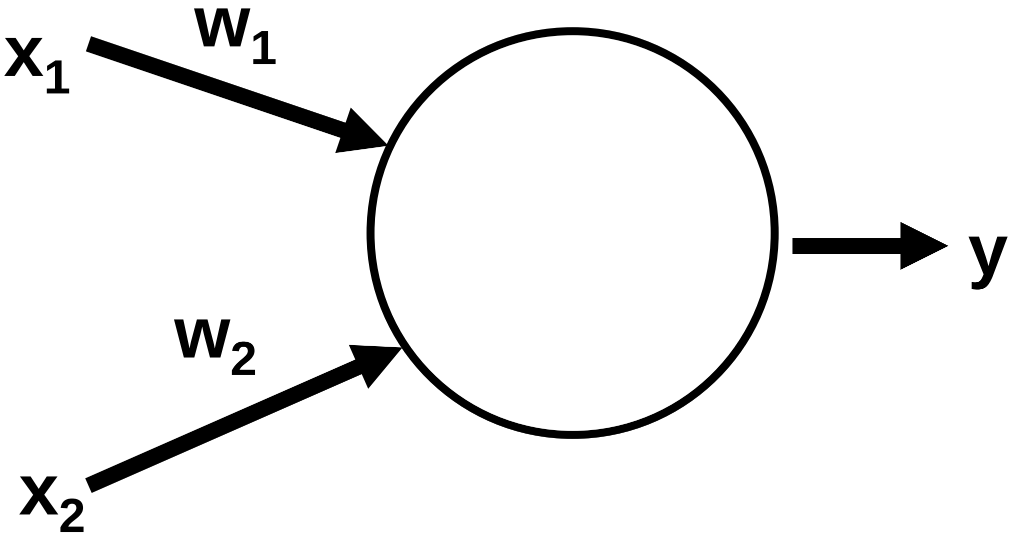


Simulation



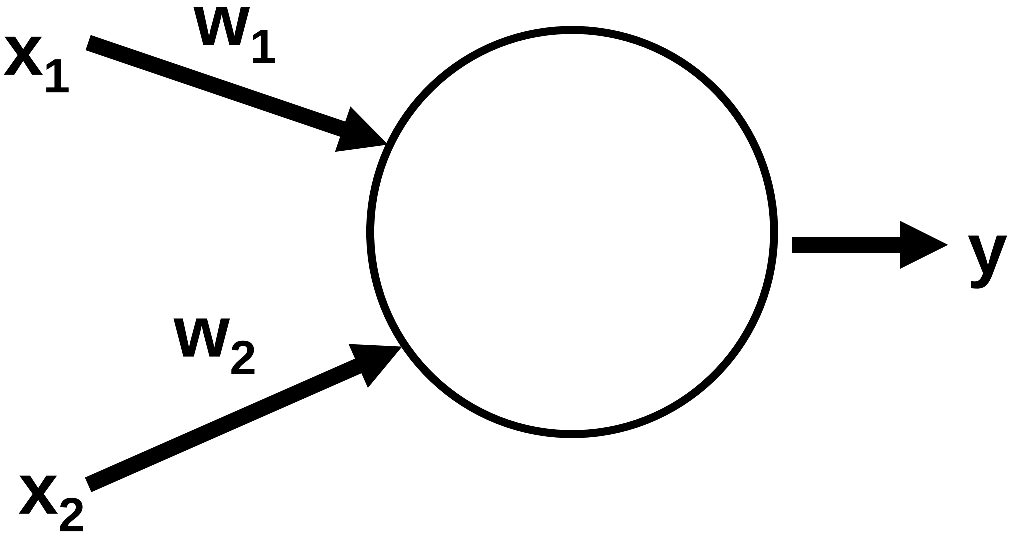
Supervised Learning

Perceptron



Unsupervised learning → no target emposed
Supervised learning → target emposed

Supervised Learning



$$\hat{y} = \sum_i w_i \cdot x_i$$

Target y: 2

Prediction $\hat{y}$: 1

Error = „Target – Prediction“ =

$$= \frac{1}{2}(\mathrm{y} - \hat{y})^2$$

$$\frac{dE}{dw_i} E = \frac{dE}{dw_i} \frac{1}{2}(y - \hat{y})^2 = -(y - \hat{y}) \frac{d\hat{y}}{dw_i} \sum_i w_i \cdot x_i = -(y - \hat{y}) \cdot x_i$$

Delta rule

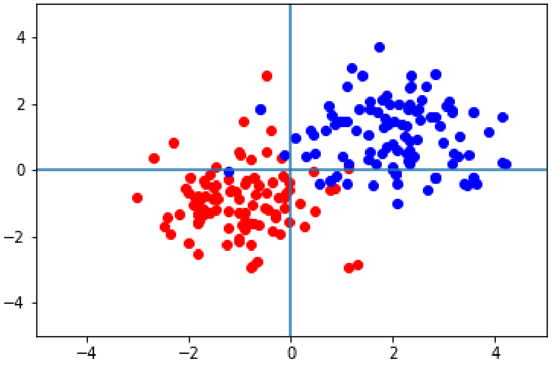
$$\Delta w_i = \alpha \cdot (y - \hat{y}) \cdot x_i$$

Simulation Delta Rule

Steps:

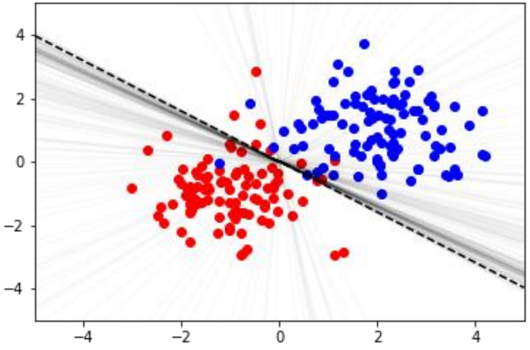
1. Data generation
2. Variable initialization
3. Iterating
    1. Compute prediction
    2. Update weights
4. Plotting

```
np.random.seed(42)

N = 100

xs = []

for _ in range(N):
    x0 = np.random.randn()-1
    x1 = np.random.randn()-1
    xs.append((x0, x1))

    x0 = np.random.randn()+2
    x1 = np.random.randn()+1
    xs.append((x0, x1))

xs = np.asarray(xs)
```

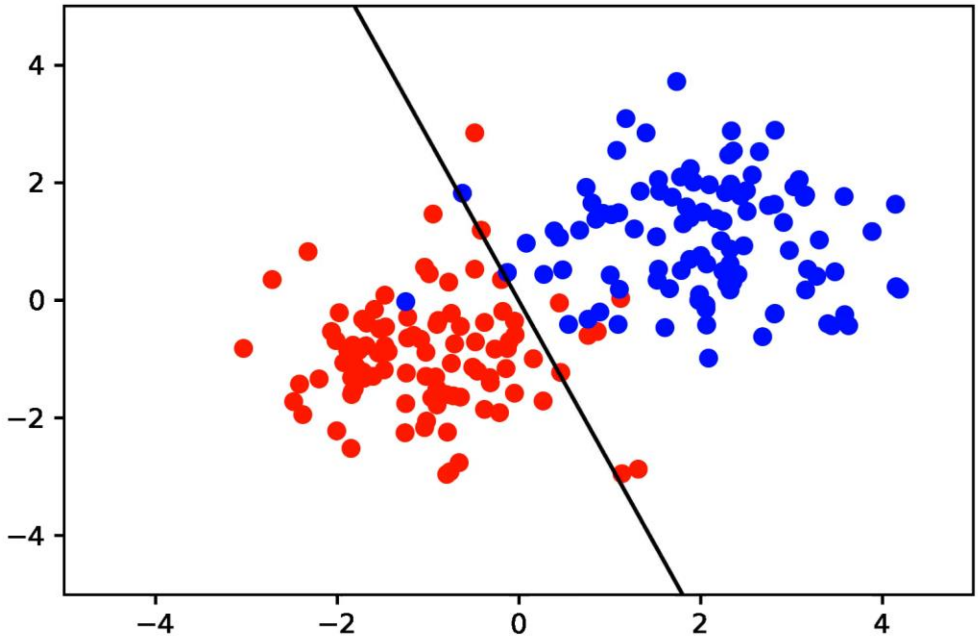



```
w = np.random.randn(2)
eta = 0.01
```

```
for i in range(2*N):
    # Determine class/target
    t = -1 if i % 2 == 0 else 1

    # Compute prediction
    pred = w @ xs[i]

    # Update weights
    w = w + eta * (t-pred) * xs[i]

```

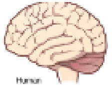


Simulation over time



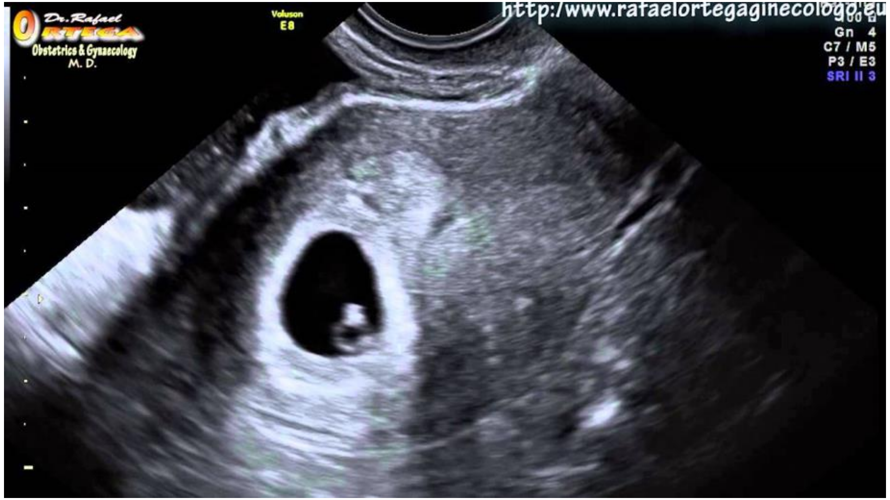
The brain

Pexels.com (CC0)



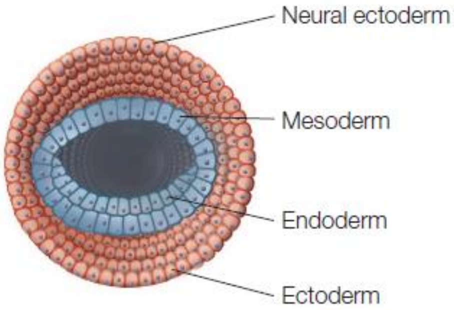
How does it emerge?

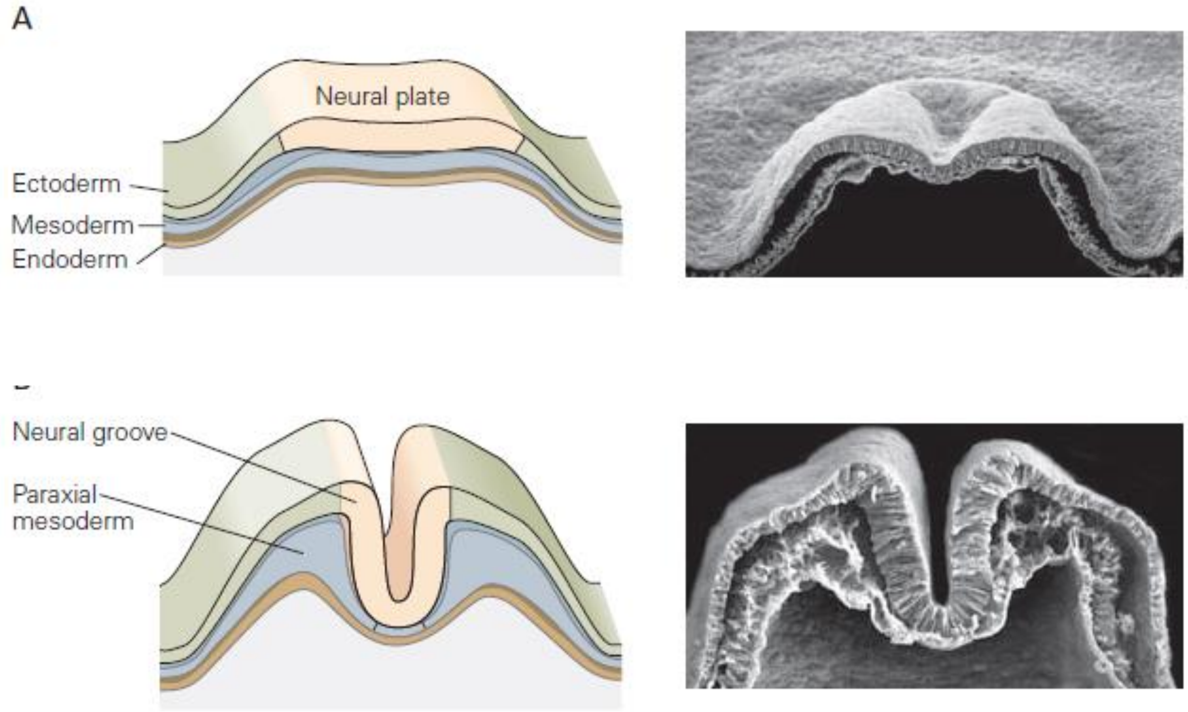
Back in the days…



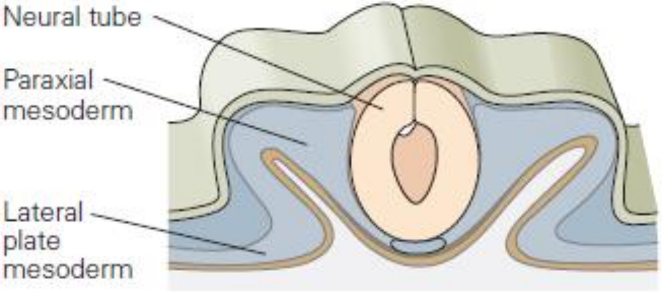
© Pauline Breijer

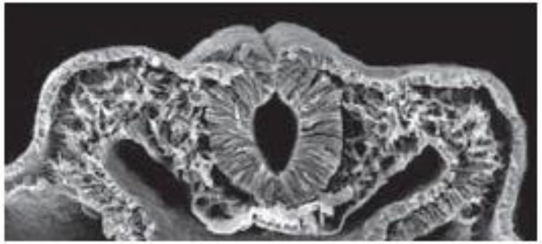
What is happening there?

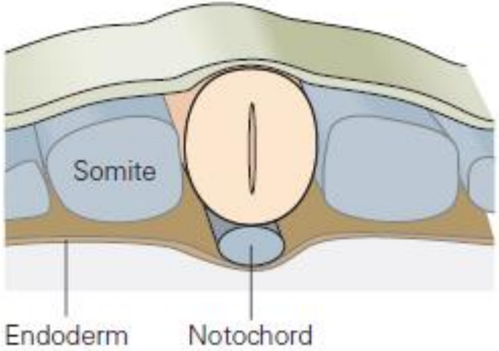


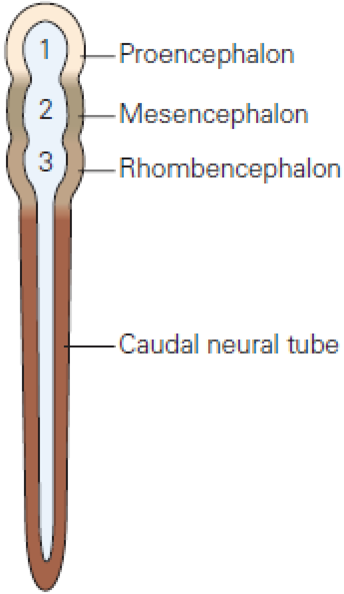


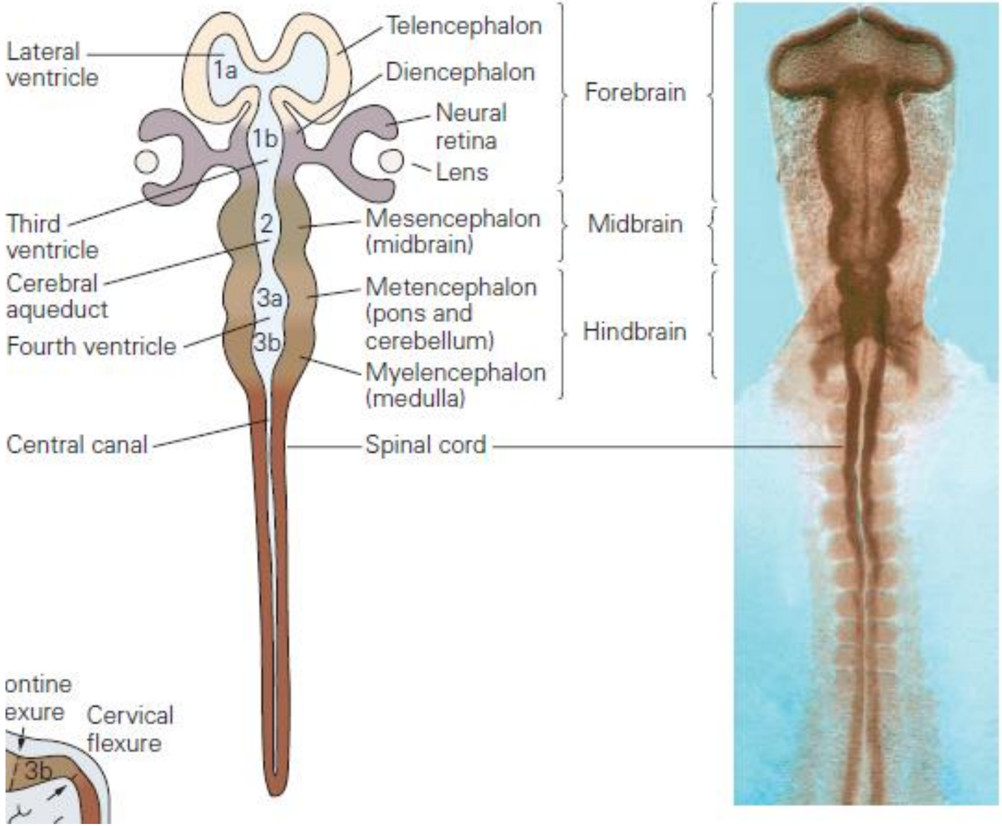
What is happening there?



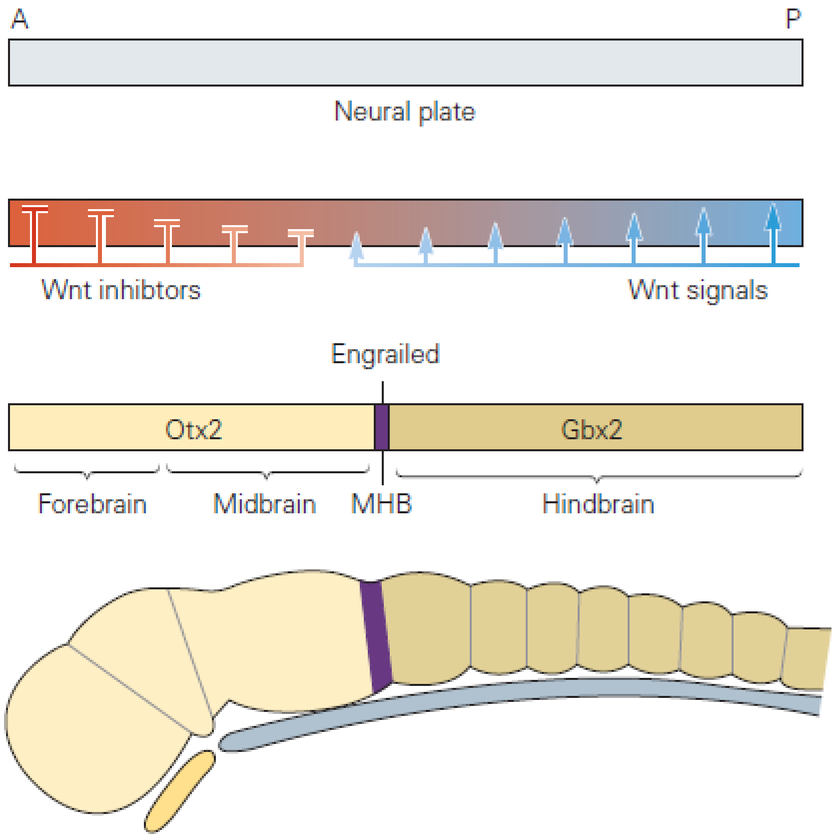


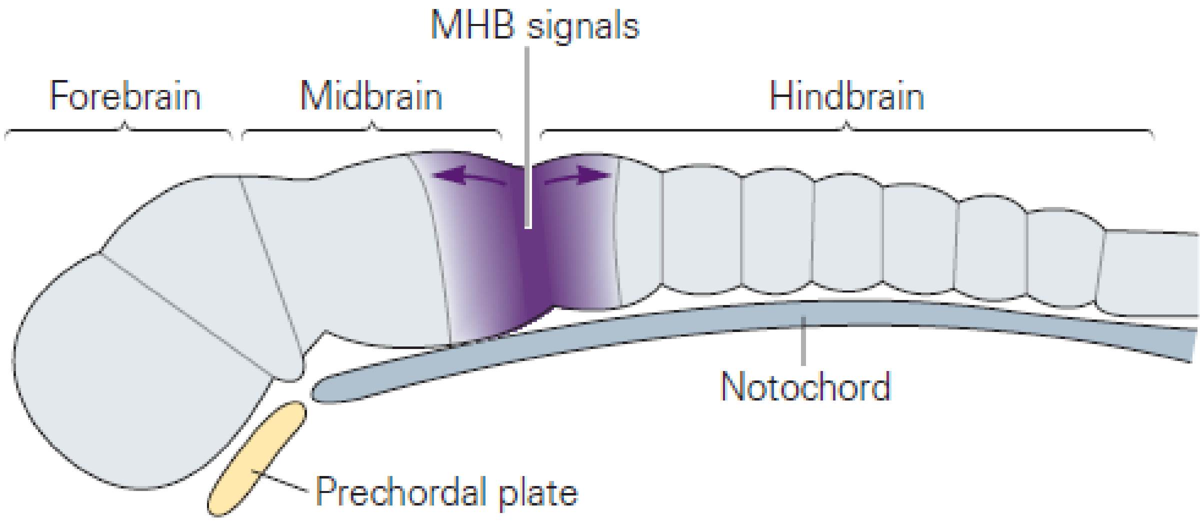



1
2
3
Proencephalon
Mesencephalon
Rhombencephalon
Caudal neural tube

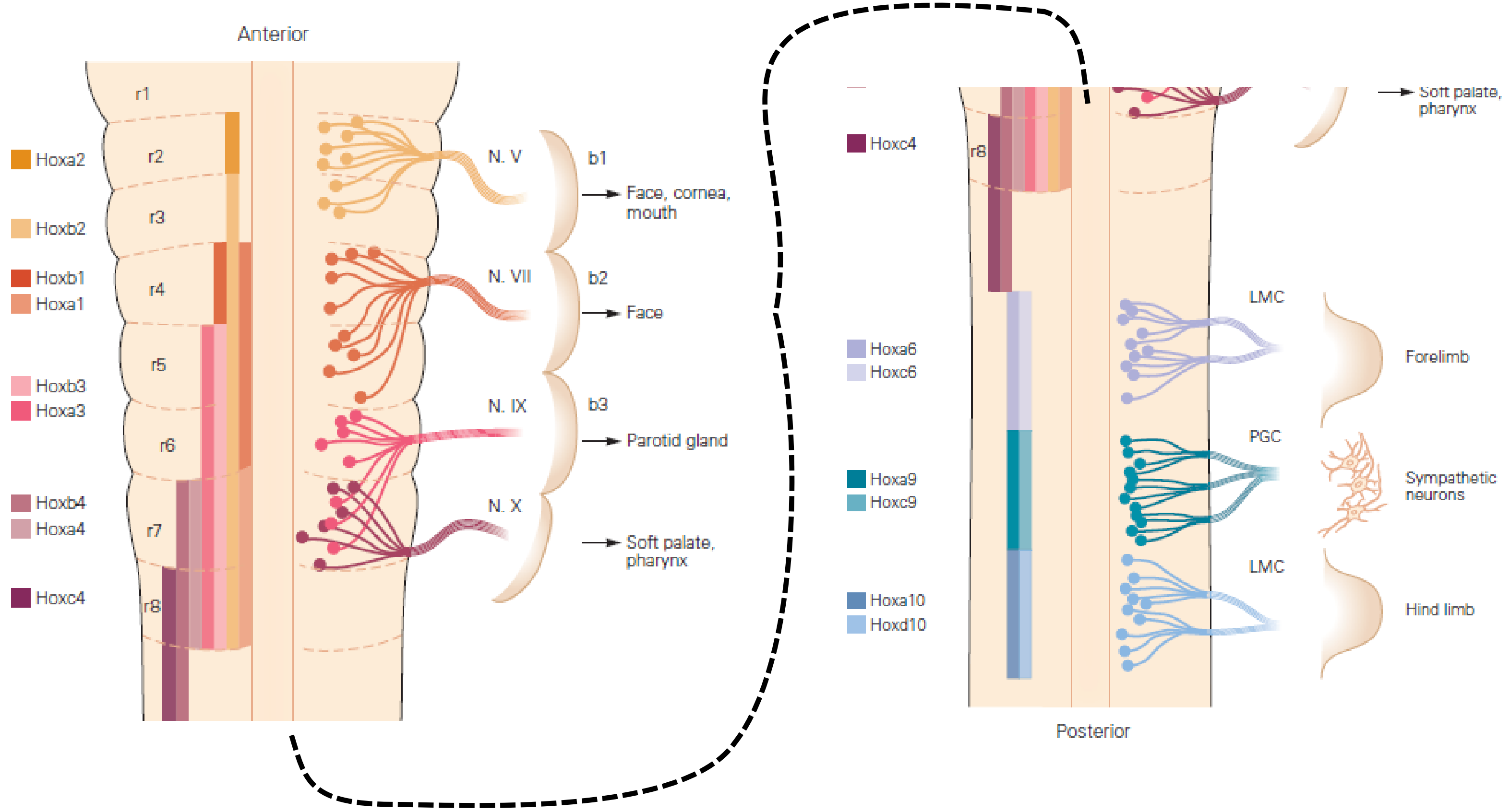

Lateral ventricle
1a
1b
2
3a
3b
Telencephalon
Diencephalon
Neural retina
Lens
Forebrain
Third ventricle
Mesencephalon (midbrain)
Midbrain
Cerebral aqueduct
Metencephalon (pons and cerebellum)
Hindbrain
Fourth ventricle
Myelencephalon (medulla)
Central canal
Spinal cord
ontine exure
Cervical flexure

Signaling pathways define neural development

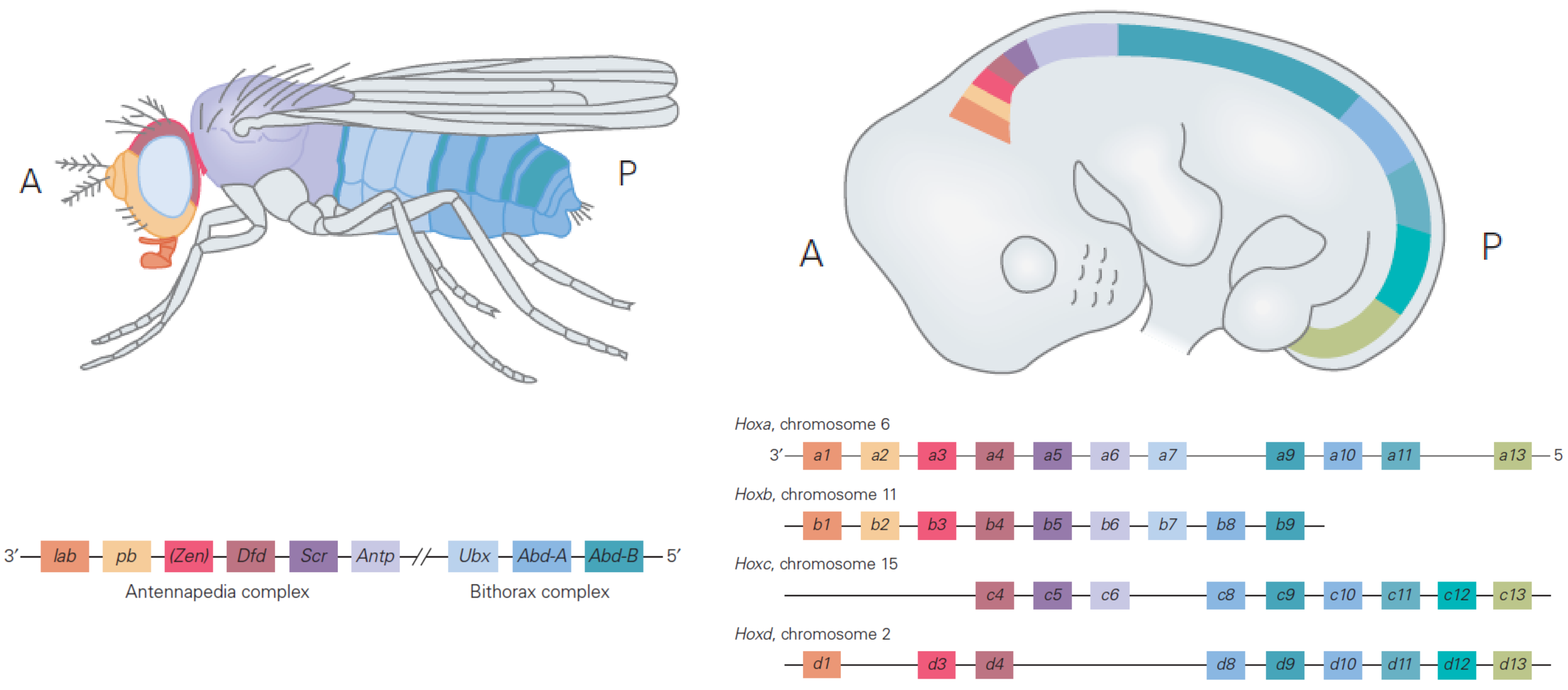



MHB signals
Forebrain
Midbrain
Hindbrain
Notochord
Prechordal plate

Hox genes determines motor neurons



Positional development is highly conserved

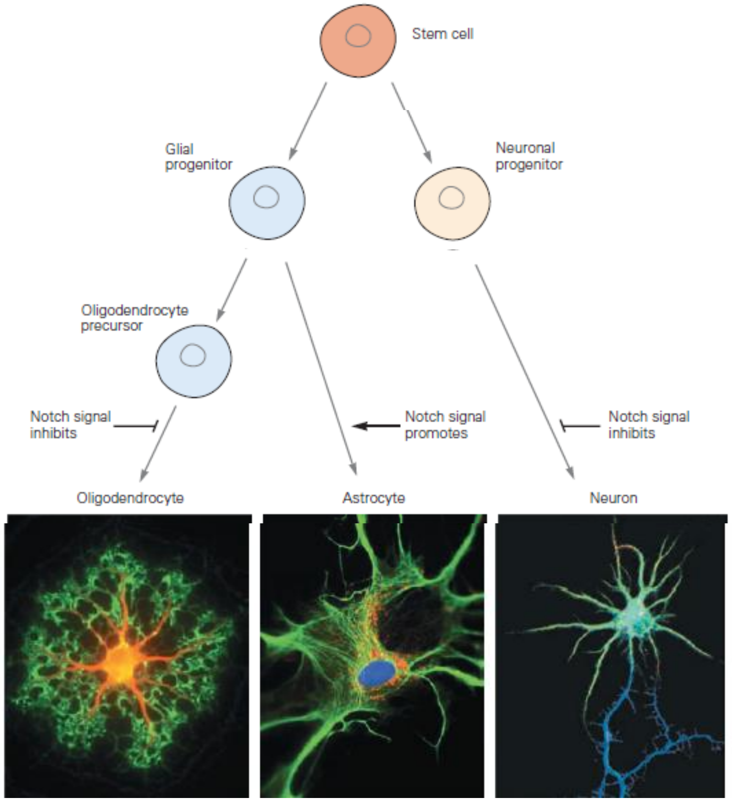


Mammals have very similar development

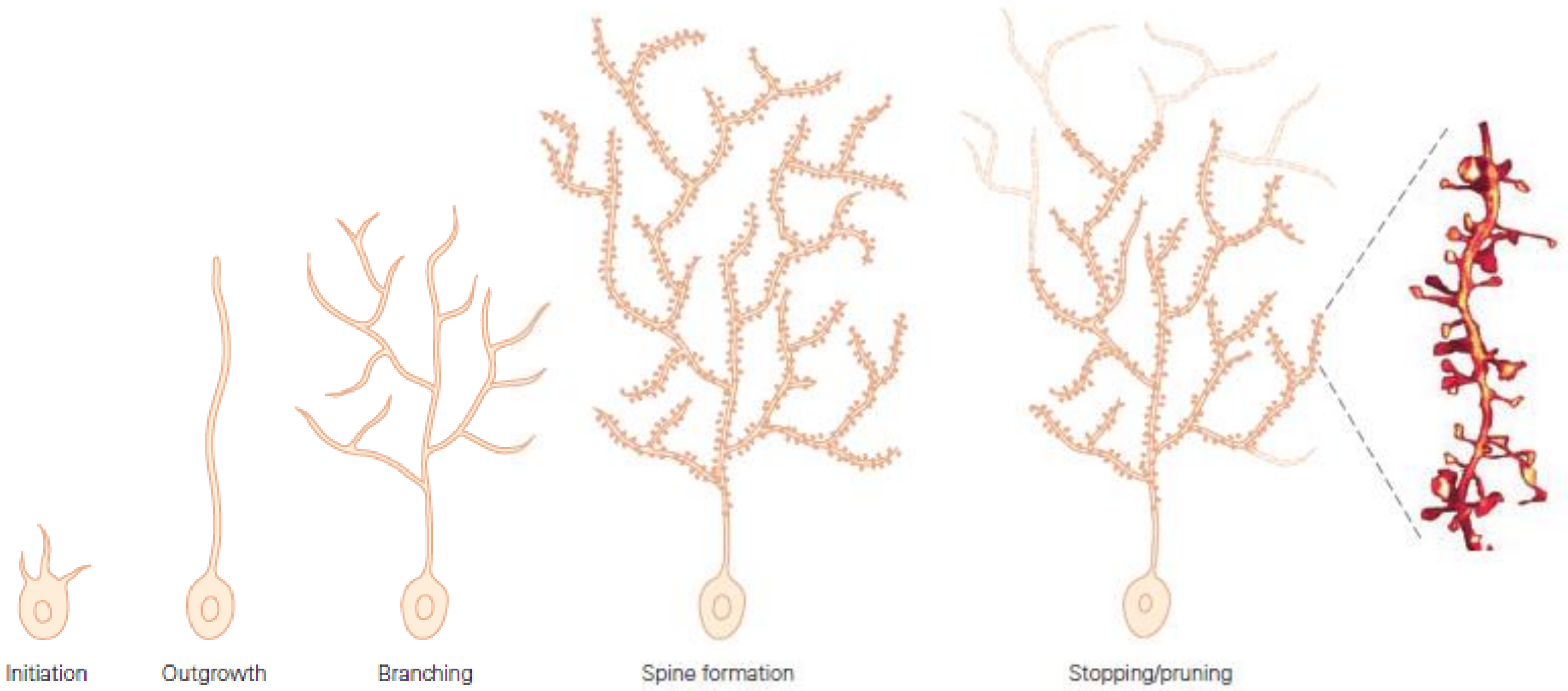
Human

Pig

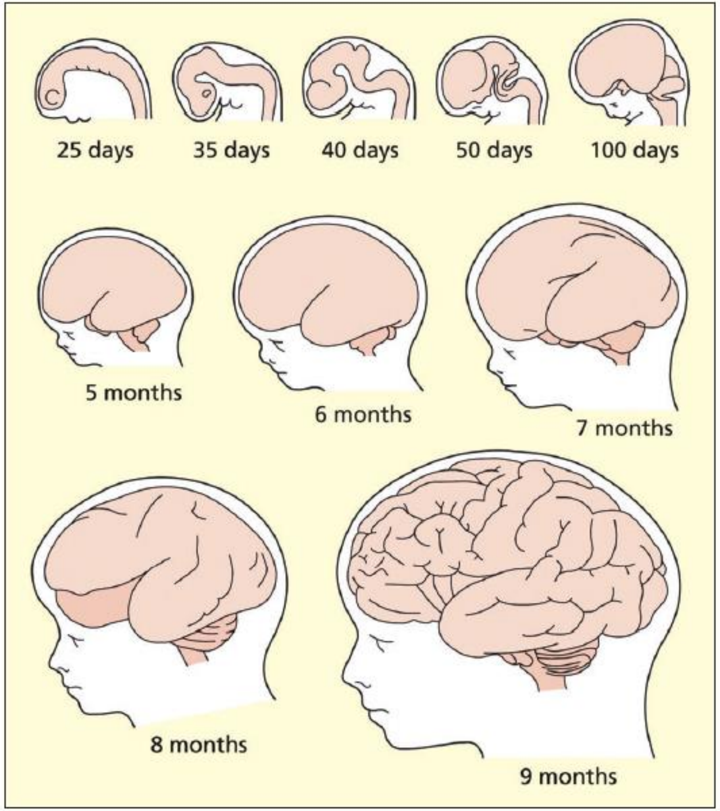
Brain cells have a common ancestor



Neuron maturation

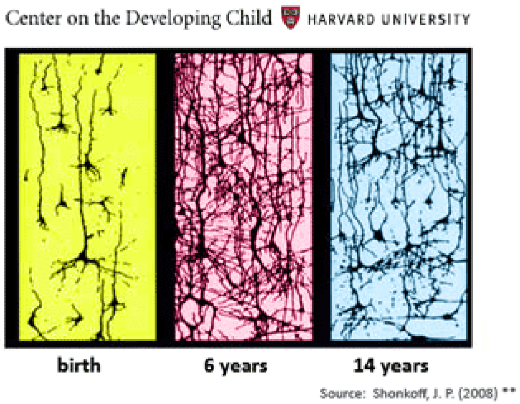


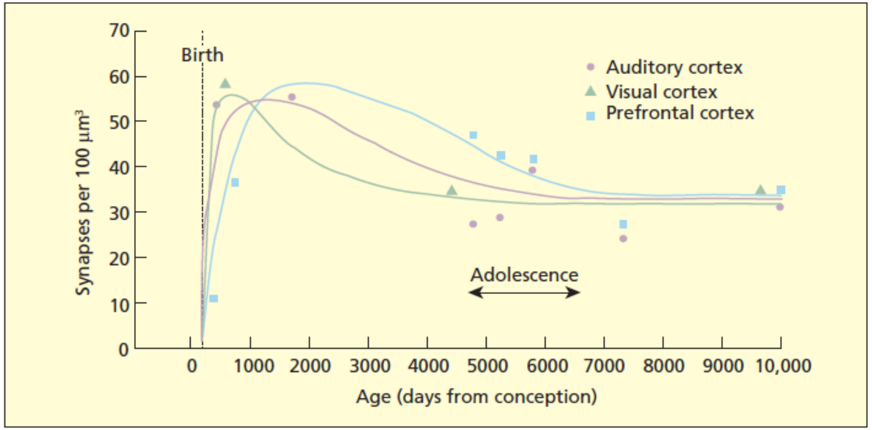
Pruning over lifetime



From Cowan, 1979. © 1979 by Scientific American, Inc. All rights reserved.

Experience Shapes Brain Architecture by Over-Production Followed by Pruning





From Huttenlocher and Dabholkar, 1997. Reprinted with permission of John Wiley & Sons Inc.

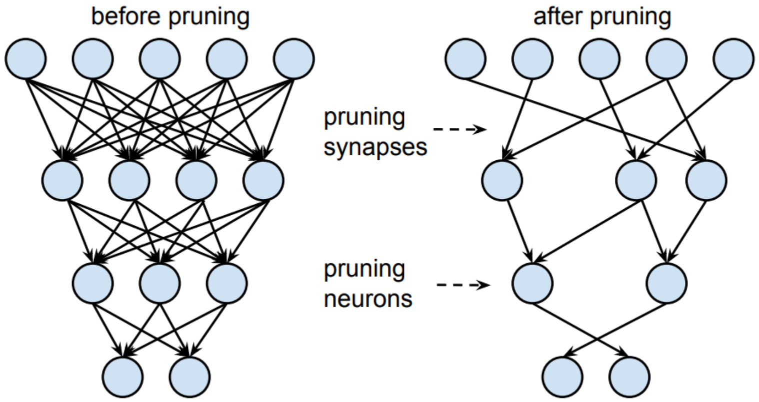
Pruning in deep neural networks

Optimal Brain Damage

Yann Le Cun, John S. Denker and Sara A. Solla
AT&T Bell Laboratories, Holmdel, N. J. 07733

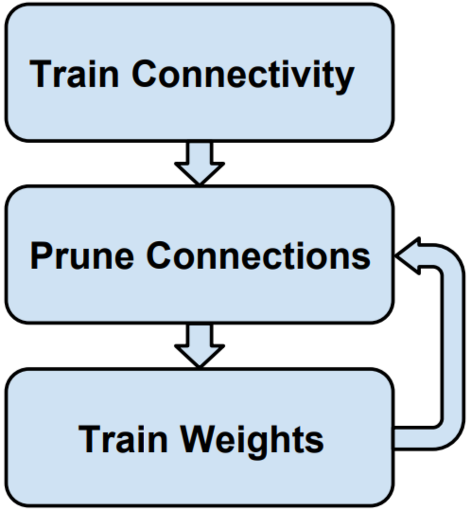
ABSTRACT

We have used information-theoretic ideas to derive a class of practical and nearly optimal schemes for adapting the size of a neural network. By removing unimportant weights from a network, several improvements can be expected: better generalization, fewer training examples required, and improved speed of learning and/or classification. The basic idea is to use second-derivative information to make a tradeoff between network complexity and training set error. Experiments confirm the usefulness of the methods on a real-world application.

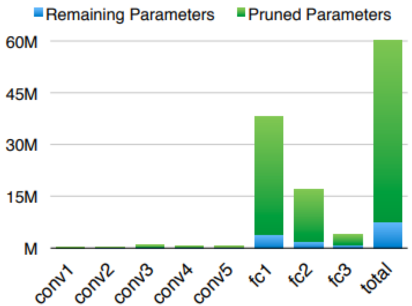


Pruning synapses: making network sparse
Pruning neurons: Making network dense

Le Cun et al., 1990



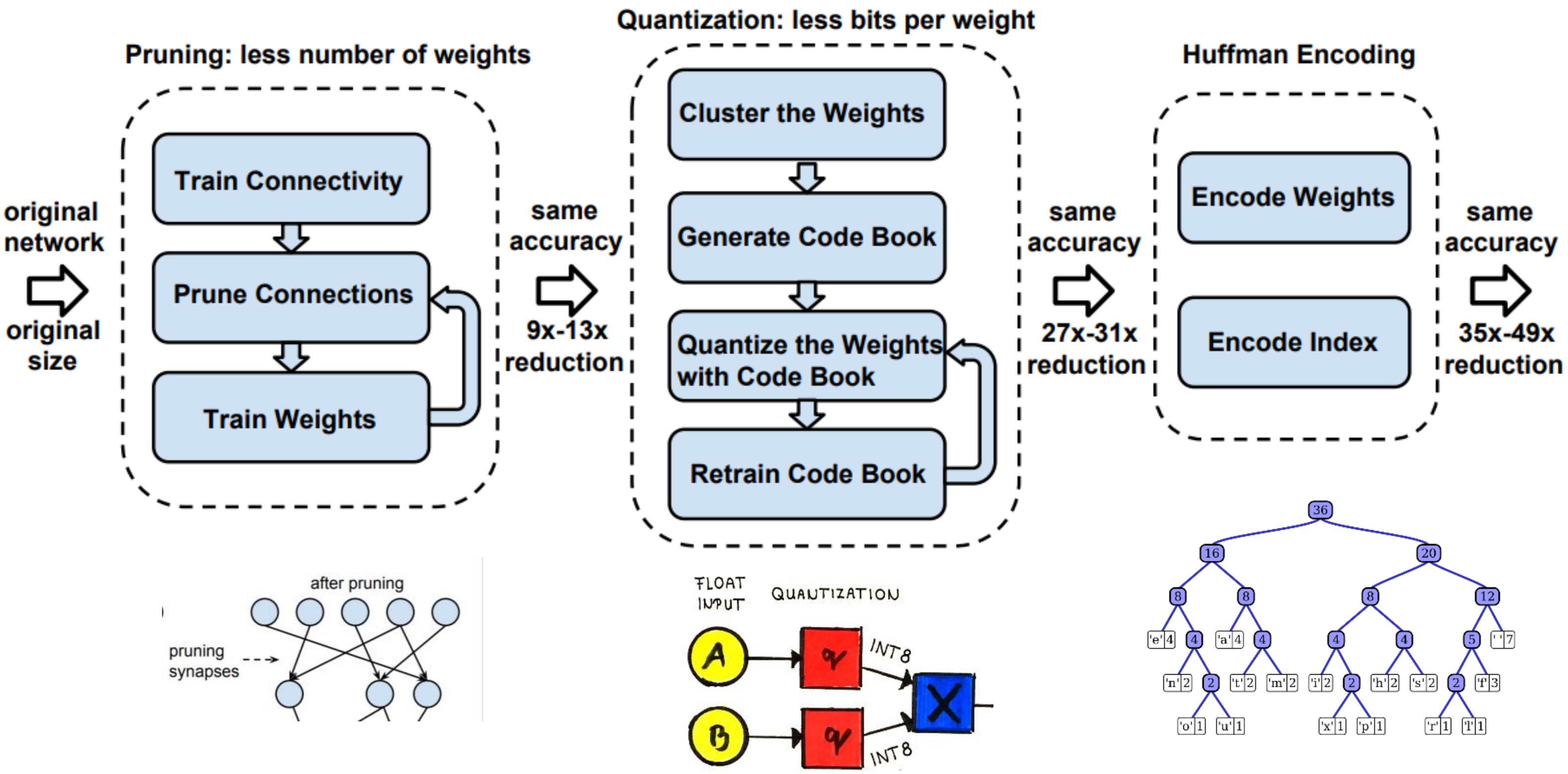
Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1	35K	211M	88%	84%	84%
conv2	307K	448M	52%	38%	33%
conv3	885K	299M	37%	35%	18%
conv4	663K	224M	40%	37%	14%
conv5	442K	150M	34%	37%	14%
fc1	38M	75M	36%	9%	3%
fc2	17M	34M	40%	9%	3%
fc3	4M	8M	100%	25%	10%
Total	61M	1.5B	54%	**11%**	**30%**



Layer	Weights	FLOP	Act%	Weights%	FLOP%
conv1_1	2K	0.2B	53%	58%	58%
conv1_2	37K	3.7B	89%	22%	12%
conv2_1	74K	1.8B	80%	34%	30%
conv2_2	148K	3.7B	81%	36%	29%
conv3_1	295K	1.8B	68%	53%	43%
conv3_2	590K	3.7B	70%	24%	16%
conv3_3	590K	3.7B	64%	42%	29%
conv4_1	1M	1.8B	51%	32%	21%
conv4_2	2M	3.7B	45%	27%	14%
conv4_3	2M	3.7B	34%	34%	15%
conv5_1	2M	925M	32%	35%	12%
conv5_2	2M	925M	29%	29%	9%
conv5_3	2M	925M	19%	36%	11%
fc6	103M	206M	38%	4%	1%
fc7	17M	34M	42%	4%	2%
fc8	4M	8M	100%	23%	9%
total	138M	30.9B	64%	**7.5%**	**21%**

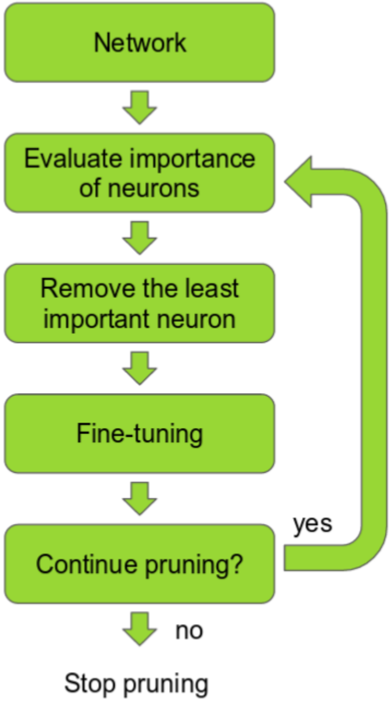
Han et al., NeurIPS 2015

Pruning in deep neural networks



Han et al., Deep Compression ICLR 2016

Bruteforce NVIDIA study



2.1 ORACLE PRUNING

Minimizing the difference in accuracy between the full and pruned models depends on the criterion for identifying the "least important" parameters, called *saliency*, at each step. The best criterion would be an exact empirical evaluation of each parameter, which we denote the *oracle* criterion, accomplished by ablating each non-zero parameter $w \in \mathcal{W}'$ in turn and recording the cost's difference.

To compute the oracle, we evaluate the change in loss caused by removing each individual feature map from the fine-tuned VGG-16 network. (See Appendix A.3 for additional analysis.) We rank

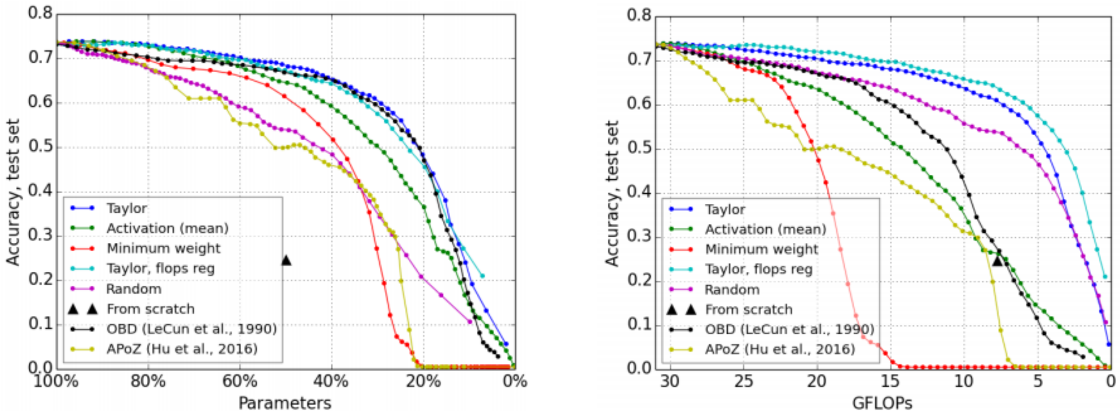
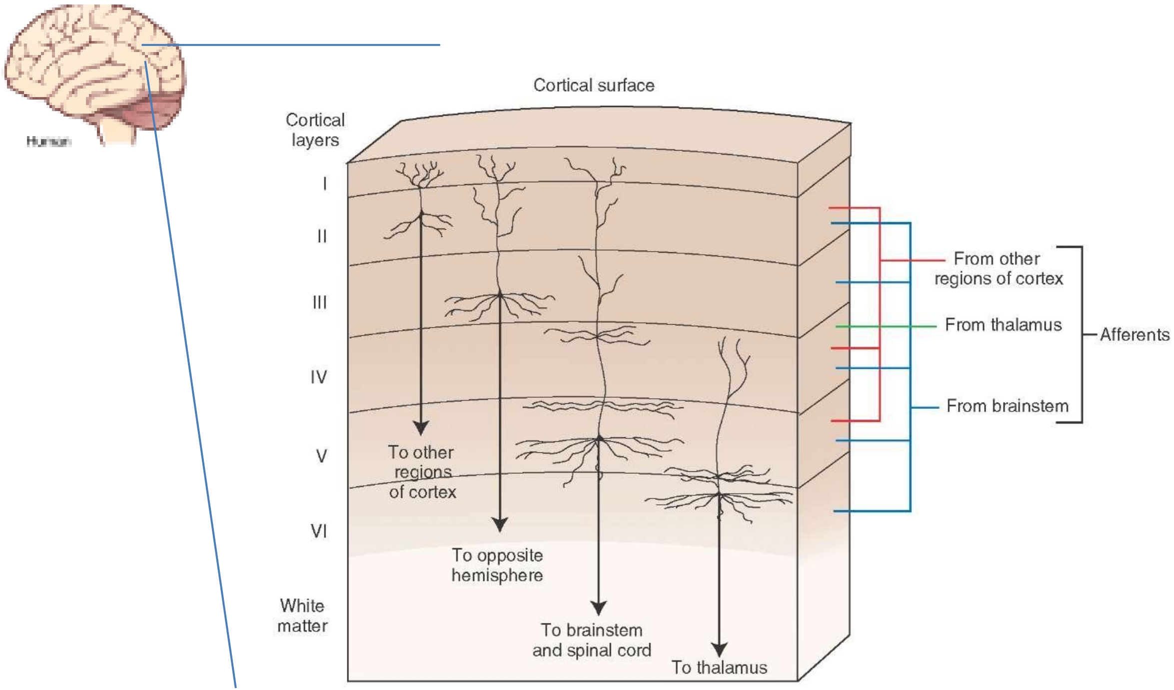


Figure 4: Pruning of feature maps in VGG-16 fine-tuned on the Birds-200 dataset.

Molchanov et al., ICLR 2017

Cortical networks



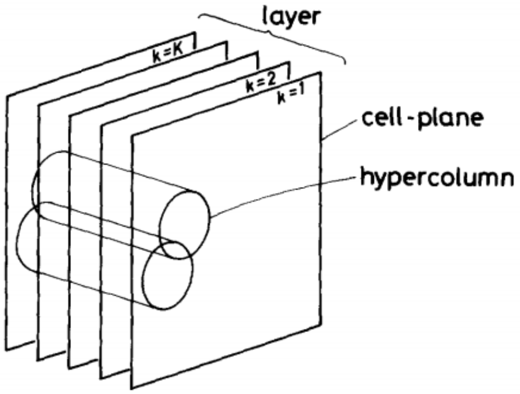
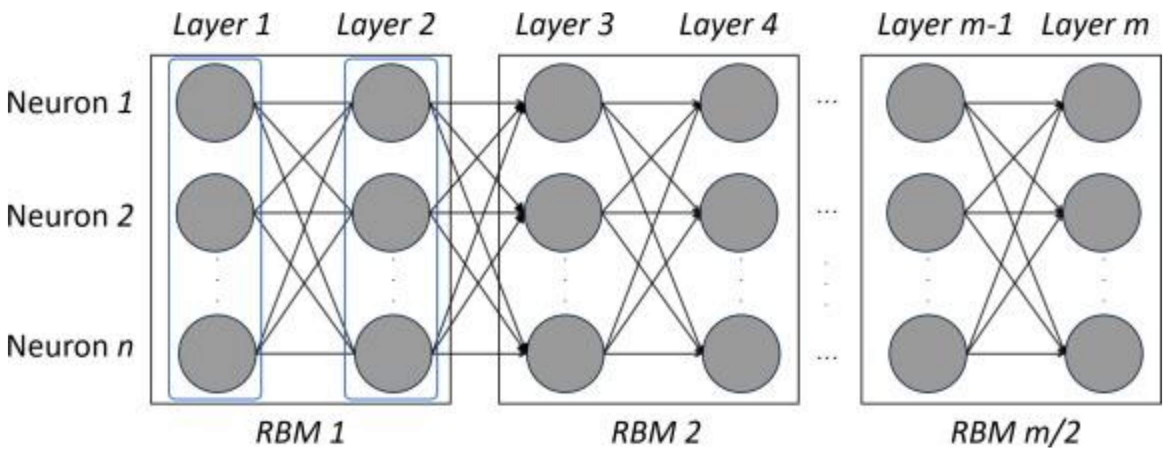


FIGURE 6. Relation between cell-planes and hypercolumns within a layer.

Deep Belief Networks



Geoffrey Hinton, © WIRED

Restricted
Boltzmann
Machines

First: unsupervised → pre-training
Second: supervised → classification

Cortical Algorithms

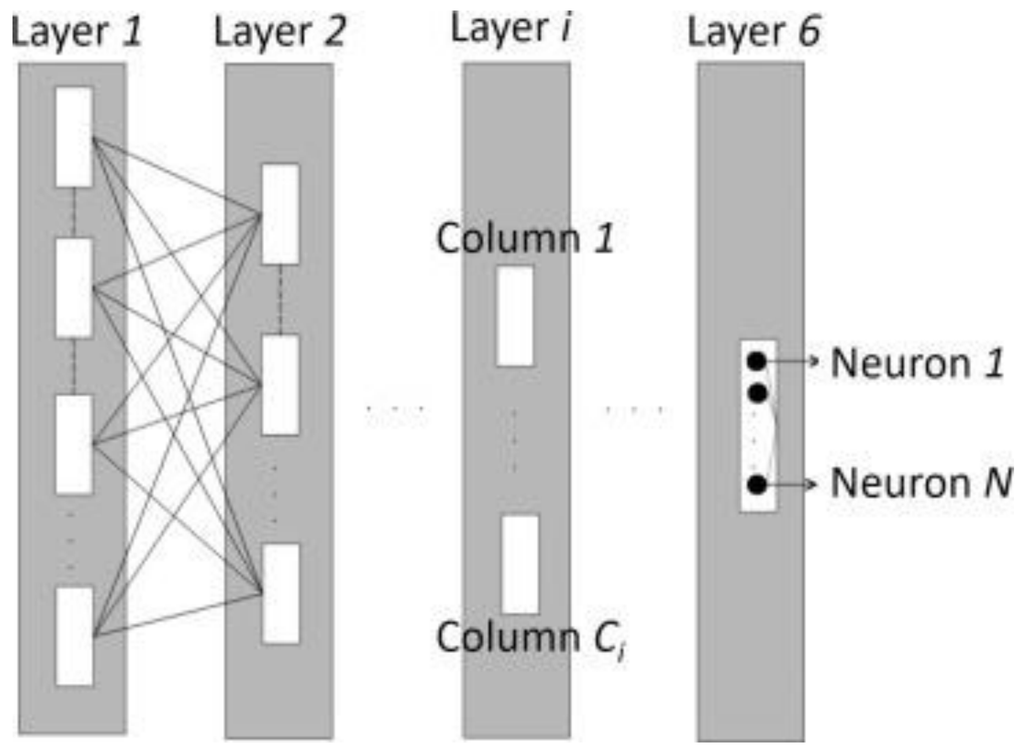


Table 9
Classification results.

Dataset	Net. size	DBN Acc. (%)	DBN Connec. (%)	CA Acc. (%)	CA Connec. (%)
MNIST	N1	38.8	88.1	96.2	89.8
	N2	84.1	91.8	97.1	85.2
	N3	84.2	98.8	98.5	79.1
	N4	89.0	47.2	98.5	58.7
	N5	88.3	54.9	99.2	43.5
	N6	**89.4**	53.9	99.8	37.6
	N7	86.5	40.8	**99.8**	28.5

DEPTH

Neurodevelopment happens during our lifetime