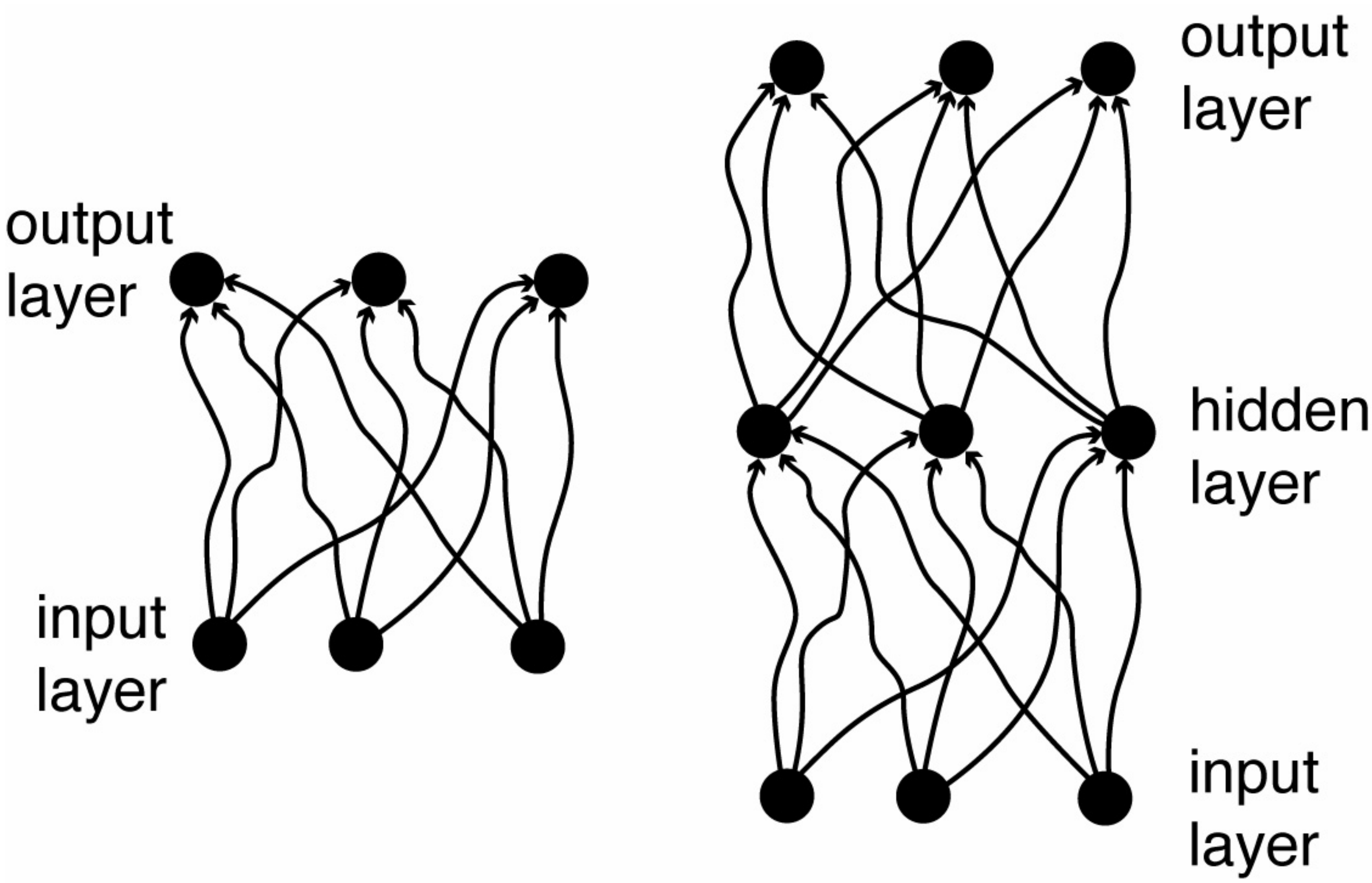
Backpropagation learning

Sebastian Seung

Simple vs. multilayer perceptron



Hidden layer problem

- Radical change for the supervised learning problem.
- No desired values for the hidden layer.
- The network must find its own hidden layer activations.

Generalized delta rule

- Delta rule only works for the output layer.
- Backpropagation, or the generalized delta rule, is a way of creating desired values for hidden layers

Multilayer perceptron

- L layers of weights and biases
- L+1 layers of neurons

$$\mathbf{x}^0 \xrightarrow{W^1,\mathbf{b}^1} \mathbf{x}^1 \xrightarrow{W^2,\mathbf{b}^2} \cdots \xrightarrow{W^L,\mathbf{b}^L} \mathbf{x}^L$$

$$x_i^{\ l} = f\left(\sum_{j=1}^{n_{l-1}} W_{ij}^{\ l} x_j^{\ l-1} + b_i^{\ l} \right)$$

Reward function

- Depends on activity of the output layer only.

$$R\left(\mathbf{x}^{L}\right)$$

- Maximize reward with respect to weights and biases.

Example: squared error

- Square of desired minus actual output, with minus sign.

$$R\left(\mathbf{x}^L\right) = -\frac{1}{2}\sum_{i=1}^{n_L}\left(d_i - x_i^{\,L}\right)^2$$

Phases of backpropagation learning

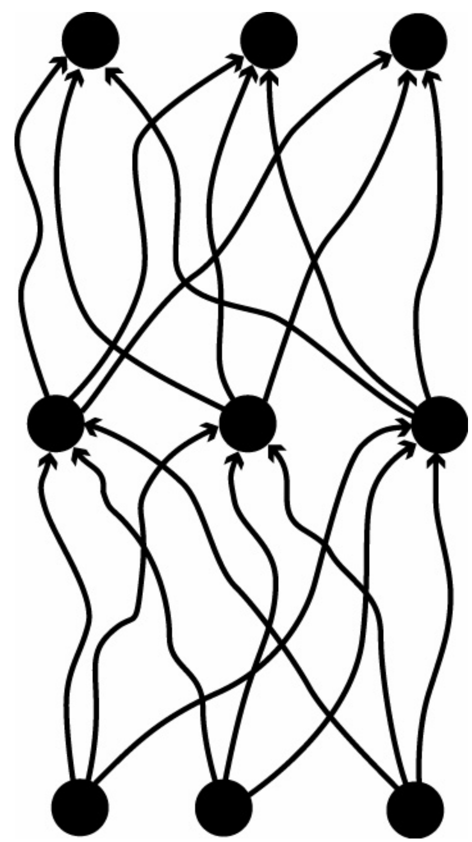
- Forward pass
- Reward/error computation
- Backward pass
- Update of weights and biases

Forward pass

For $l = 1$ to L,

synaptic input $\quad u_i^{\,l} = \sum_{j=1}^{n_{l-1}} W_{ij}^{\,l} x_j^{\,l-1} + b_i^{\,l}$

activity $\quad x_i^{\,l} = f\left(u_i^{\,l}\right)$



Sensitivity computation

- The sensitivity is also called “delta.”

$$\hat{x}_i^L = \frac{\partial R}{\partial x_i^L}$$

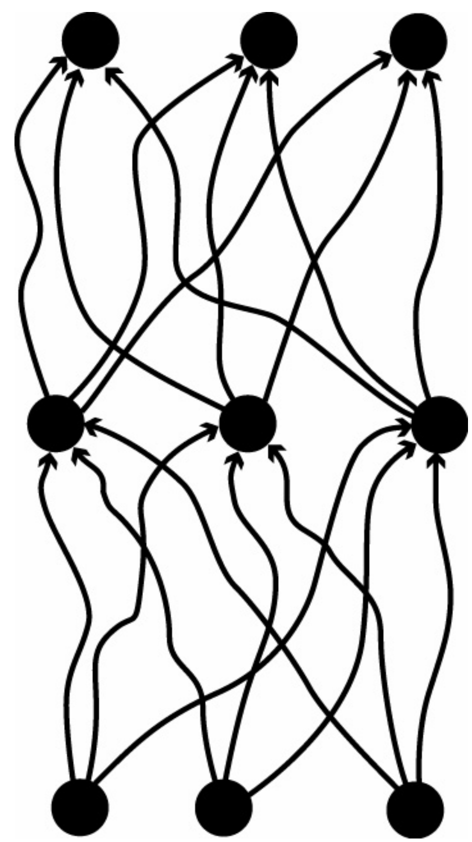
$$= d_i - x_i^L$$

Backward pass

for $l = L$ to 1

$$\hat{u}_j^{\,l} = f'\left(u_j^{\,l}\right)\hat{x}_j^{\,l}$$

$$\hat{x}_j^{\,l-1} = \sum_{i=1}^{n_l} \hat{u}_i^{\,l} W_{ij}^{\,l}$$



- (actually $\mathbf{x}^0$ is not needed)

Learning update

- In any order

$$\Delta W_{ij}^{\,l} \propto \hat{u}_i^{\,l} x_j^{\,l-1}$$

$$\Delta b_i^{\,l} \propto \hat{u}_i^{\,l}$$

Backprop is a gradient update

- Consider R as function of weights and biases.

$$\Delta W_{ij}^{\,l} \propto \frac{\partial R}{\partial W_{ij}^{\,l}} = \hat{u}_i^{\,l} x_j^{\,l-1}$$

$$\Delta b_i^{\,l} \propto \frac{\partial R}{\partial b_i^{\,l}} = \hat{u}_i^{\,l}$$

Sensitivities

- How sensitive is R to perturbations in parameters?
- Sensitivity matrix $\dfrac{\partial R}{\partial W_{ij}^{\ l}}$
- Sensitivity vector $\dfrac{\partial R}{\partial b_i^{\ l}}$

Sensitivity lemma

- Sensitivity matrix = outer product
  - sensitivity vector
  - activity vector

$$\frac{\partial R}{\partial W_{ij}^{l}} = \frac{\partial R}{\partial b_{i}^{l}} x_{j}^{l-1}$$

- The sensitivity vector is sufficient.
- Generalization of “delta.”

Sensitivity is the gradient with respect to synaptic input

- total synaptic input $$u_i^{\,l} = \sum_{j=1}^{n_{l-1}} W_{ij}^{\,l} x_j^{\,l-1} + b_i^{\,l}$$
- applying the chain rule to

$$b_i^l \to u_i^l \to R$$

- yields

$$\frac{\partial R}{\partial b_i^{\,l}} = \frac{\partial R}{\partial u_i^{\,l}} \frac{\partial u_i^{\,l}}{\partial b_i^{\,l}} = \frac{\partial R}{\partial u_i^{\,l}}$$

Recursive calculation of the sensitivity

- Given $$\frac{\partial R}{\partial u_i^{\,l}}$$
- The chain rule can be used to calculate

$$\frac{\partial R}{\partial u_i^{\,l-1}}$$

Forward and backward passes

$$\mathbf{x}^0 \xrightarrow{W^1,\mathbf{b}^1} \mathbf{u}^1 \xrightarrow{f} \mathbf{x}^1 \cdots \mathbf{x}^{l-1} \xrightarrow{W^l,\mathbf{b}^l} \mathbf{u}^l \xrightarrow{f} \mathbf{x}^l \cdots \mathbf{x}^{L-1} \xrightarrow{W^L,\mathbf{b}^L} \mathbf{u}^L \xrightarrow{f} \mathbf{x}^L$$

$$\hat{x}_i^l = \frac{\partial R}{\partial x_i^l} \qquad \hat{u}_i^l = \frac{\partial R}{\partial u_i^l}$$

$$\hat{\mathbf{u}}^1 \xleftarrow{f'} \hat{\mathbf{x}}^1 \cdots \hat{\mathbf{x}}^{l-1} \xleftarrow{(W^l)^T} \hat{\mathbf{u}}^l \xleftarrow{f'} \hat{\mathbf{x}}^l \cdots \hat{\mathbf{x}}^{L-1} \xleftarrow{(W^L)^T} \hat{\mathbf{u}}^L \xleftarrow{f'} \hat{\mathbf{x}}^L$$

Chain rule

$$u_i^l \rightarrow x_i^l \rightarrow R$$

$$\frac{\partial R}{\partial u_i^l} = \frac{\partial R}{\partial x_i^l} \frac{\partial x_i^l}{\partial u_i^l}$$

$$= \frac{\partial R}{\partial x_i^l} f'\left(u_i^l\right)$$

$$\hat{u}_i^l = \hat{x}_i^l f'\left(u_i^l\right)$$

Jacobian matrix

$$u_i^l = \sum_{j=1}^{n_{l-1}} W_{ij}^l x_j^{l-1} + b_i^l$$

$$\frac{\partial u_i^l}{\partial x_j^{l-1}} = W_{ij}^l$$

Chain rule

$$\mathbf{x}^{l-1} \rightarrow \mathbf{u}^{l} \rightarrow R$$

$$\frac{\partial R}{\partial x_j^{\,l-1}} = \sum_i \frac{\partial R}{\partial u_i^{\,l}} \frac{\partial u_i^{\,l}}{\partial x_j^{\,l-1}} = \sum_i \frac{\partial R}{\partial u_i^{\,l}} W_{ij}^{\,l}$$

$$\hat{x}_j^{\,l-1} = \sum_i \hat{u}_i^l W_{ij}^{\,l}$$

$$\hat{\mathbf{x}}^{l-1} = \left(W^{\,l}\right)^T \hat{\mathbf{u}}^l$$

Computational complexity

- Naïve estimate
  - network output: order N
  - each component of the gradient: order N
  - N components: order N^2
- With backprop: order N

Biological plausibility

- Local: pre- and postsynaptic variables

$$x_j^{l-1} \xrightarrow{W_{ij}^{l}} u_i^{l}, \quad \hat{x}_j^{l-1} \xleftarrow{W_{ij}^{l}} \hat{u}_i^{l}$$

- Extra set of variables: sensitivities
  - Not clear what biophysical quantity could represent sensitivity
- Forward and backward passes use same weights

Backprop for brain modeling

- Backprop may not be a plausible account of learning in the brain.
- But perhaps the networks created by it are similar to biological neural networks.
- Zipser and Andersen:
  - train network
  - compare hidden neurons with those found in the brain.

More general definition on a directed acyclic graph

$$\begin{aligned} u_i^\alpha &= \sum_{\beta j} W_{ij}^{\alpha\beta} x_j^\beta + b_i^\alpha \\ x_i^\alpha &= f(u_i^\alpha) \end{aligned}$$

$$\begin{aligned} \hat{x}_j^\beta &= \sum_{i\alpha} \hat{u}_i^\alpha W_{ij}^{\alpha\beta} \\ \hat{u}_j^\beta &= f'(u_j^\beta)\hat{x}_j^\beta \end{aligned}$$

Convolutional network

$$W_{ij}^{\alpha\beta} = w_{i-j}^{\alpha\beta} \qquad b_i^{\alpha} = b^{\alpha}$$

$$\begin{aligned} \mathbf{u}^{\alpha} &= \sum_{\beta} \mathbf{w}^{\alpha\beta} * \mathbf{x}^{\beta} + b^{\alpha}\mathbf{1} \\ \mathbf{x}^{\alpha} &= \mathbf{f}(\mathbf{u}^{\alpha}) \end{aligned}$$

Backward pass (convolutional networks)

- Use the spatial inversion of the filter
  - i.e., flip the filter about each axis
- Convolution is multiplication by a matrix of the form $W_{ij} = w_{i-j}$
- The matrix transpose is $W_{ji} = w_{j-i}$

Gradient “sharing”

$$x(t) = t \qquad y(t) = t$$

$$\begin{aligned}
\frac{d}{dt} f(x(t), y(t)) &= \frac{\partial f}{\partial x}\frac{dx}{dt} + \frac{\partial f}{\partial y}\frac{dy}{dt} \\
&= \frac{\partial f}{\partial x} + \frac{\partial f}{\partial y}
\end{aligned}$$

Gradient learning with parameter sharing

- Find the derivatives with respect to each of the individual parameters, treating them as independent. Then sum all of these to find the derivative with respect to the shared parameter.

For Whoever Shares, to Him More Gradient Will Be Given

corruption of Mark 4:25

Weight update (convolutional networks)

$$\Delta W_{ij}^{\alpha\beta} \propto \hat{u}_i^\alpha x_j^\beta$$

$$\begin{aligned}\Delta w_k^{\alpha\beta} &\propto \sum_{\substack{i,j \\ i-j=k}} \hat{u}_i^\alpha x_j^\beta \\ &= \sum_j \hat{u}_{j+k}^\alpha x_j^\beta\end{aligned}$$

LeNet

- Trained with backprop
- Convolution
- Subsampling