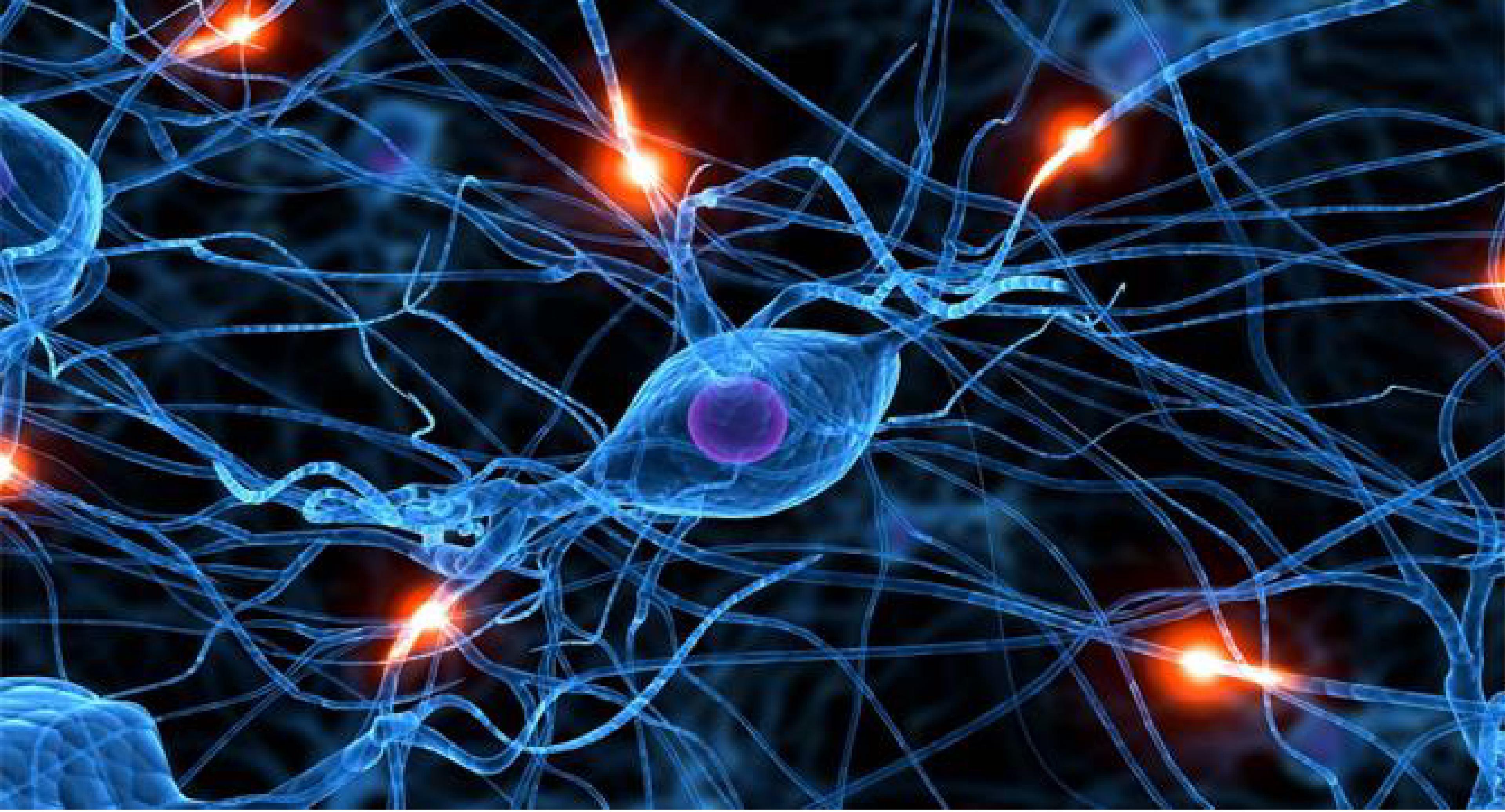


Neural Networks: Learning

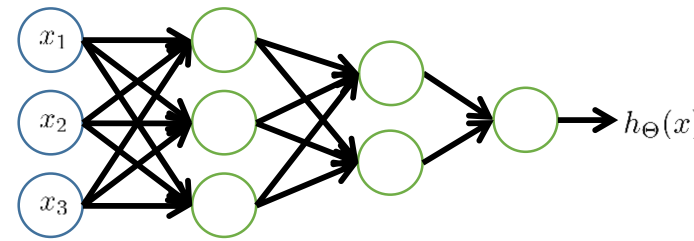
Backpropagation

Last time

Neural network chains together many layers of "neurons" such as logistic units

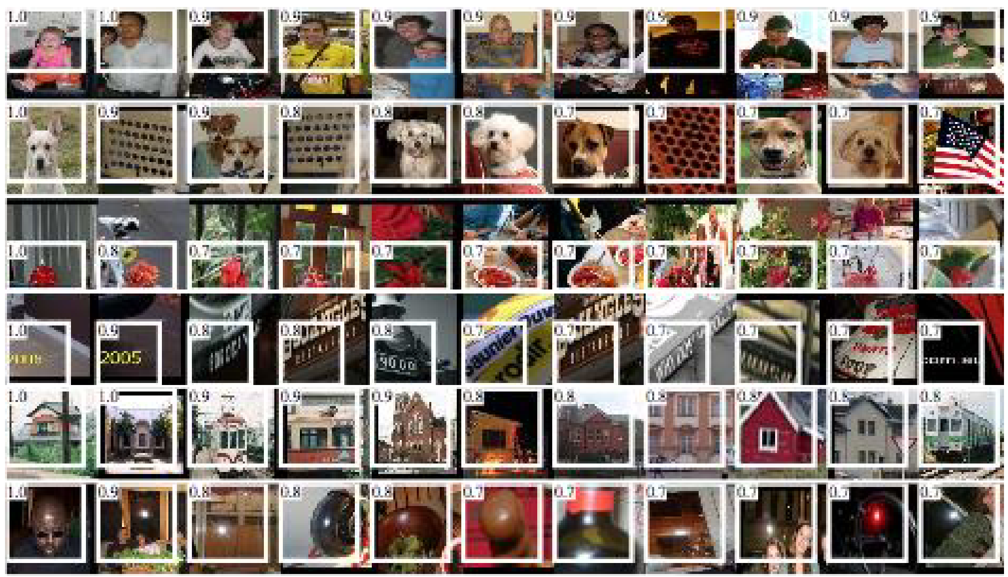
Hidden neurons learn more and more abstract non-linear features

Fully connected neural net



Layer 1 Layer 2 Layer 3 Layer 4

The Unreasonable Effectiveness of Deep Features



Maximal activations of pool$_5$ units [R-CNN]

How can we learn parameters of a multilayer network?

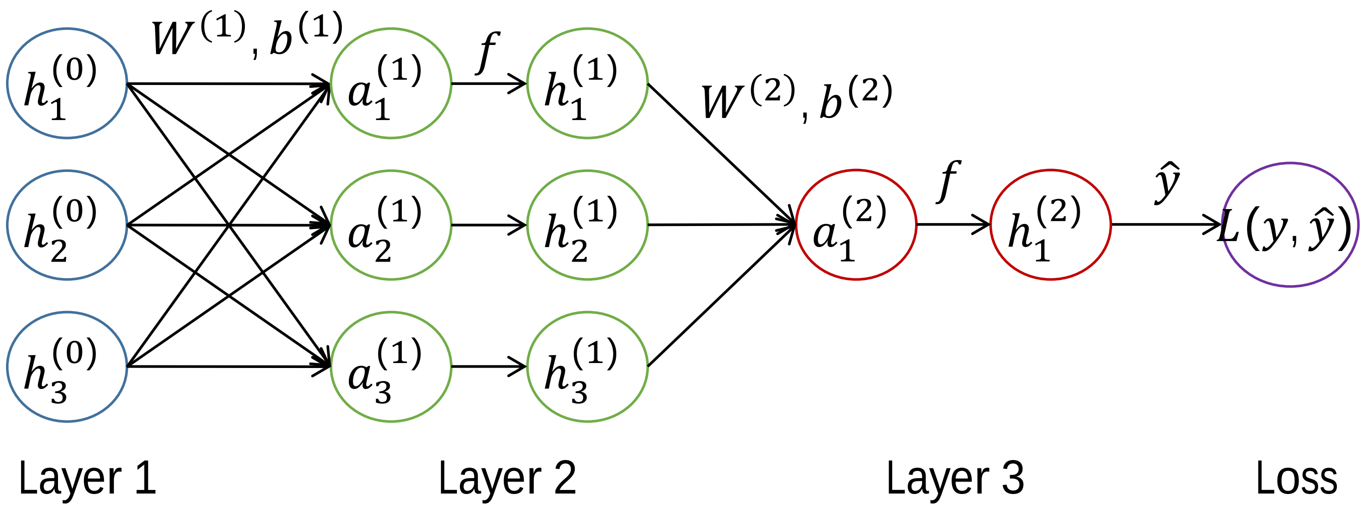
- Use gradient descent
- Compute gradient of loss using chain rule
- Efficient application of chain rule == backpropagation

Backpropagation

- [Slides from Stanford course]

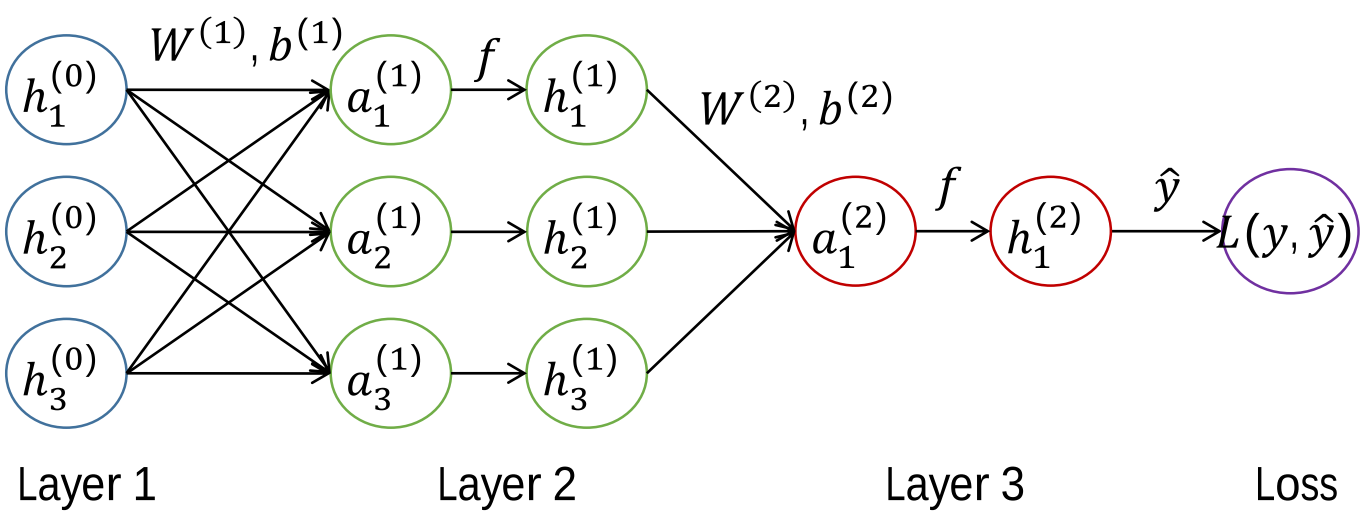
Backpropagation for a neural net

- Example for a simple 1-hidden layer neural net
- What is the computational graph?



Require: Network depth, l
Require: $\boldsymbol{W}^{(i)}, i \in \{1, \dots, l\}$, the weight matrices of the model
Require: $\boldsymbol{b}^{(i)}, i \in \{1, \dots, l\}$, the bias parameters of the model
Require: $\boldsymbol{x}$, the input to process
Require: $\boldsymbol{y}$, the target output

$\boldsymbol{h}^{(0)} = \boldsymbol{x}$
for $k = 1, \dots, l$ **do**
  $\boldsymbol{a}^{(k)} = \boldsymbol{b}^{(k)} + \boldsymbol{W}^{(k)} \boldsymbol{h}^{(k-1)}$
  $\boldsymbol{h}^{(k)} = f(\boldsymbol{a}^{(k)})$
end for
$\hat{\boldsymbol{y}} = \boldsymbol{h}^{(l)}$
$J = L(\hat{\boldsymbol{y}}, \boldsymbol{y}) + \lambda \Omega(\theta)$

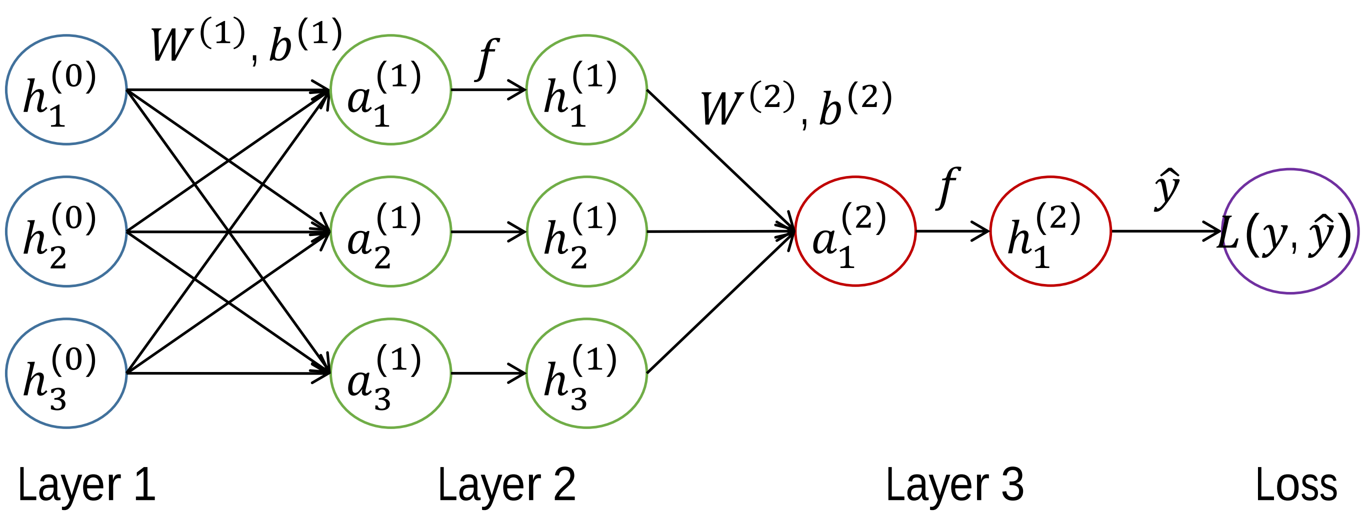


After the forward computation, compute the gradient on the output layer:
$\boldsymbol{g} \leftarrow \nabla_{\hat{\boldsymbol{y}}} J = \nabla_{\hat{\boldsymbol{y}}} L(\hat{\boldsymbol{y}}, \boldsymbol{y})$
for $k = l, l-1, \ldots, 1$ **do**

Convert the gradient on the layer's output into a gradient into the pre-nonlinearity activation (element-wise multiplication if f is element-wise):
$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{a}^{(k)}} J = \boldsymbol{g} \odot f'(\boldsymbol{a}^{(k)})$
Compute gradients on weights and biases (including the regularization term, where needed):
$\nabla_{\boldsymbol{b}^{(k)}} J = \boldsymbol{g} + \lambda \nabla_{\boldsymbol{b}^{(k)}} \Omega(\theta)$
$\nabla_{\boldsymbol{W}^{(k)}} J = \boldsymbol{g}\, \boldsymbol{h}^{(k-1)\top} + \lambda \nabla_{\boldsymbol{W}^{(k)}} \Omega(\theta)$
Propagate the gradients w.r.t. the next lower-level hidden layer's activations:
$\boldsymbol{g} \leftarrow \nabla_{\boldsymbol{h}^{(k-1)}} J = \boldsymbol{W}^{(k)\top} \boldsymbol{g}$

end for

ps2



Artificial Neural Network:

general notation

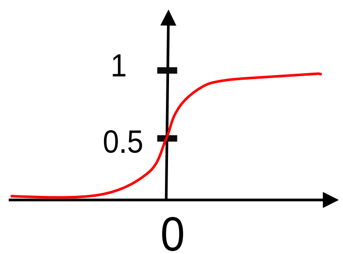
input

$$x = \begin{bmatrix} x_1 \\ \cdots \\ x_5 \end{bmatrix}$$

hidden layer activations

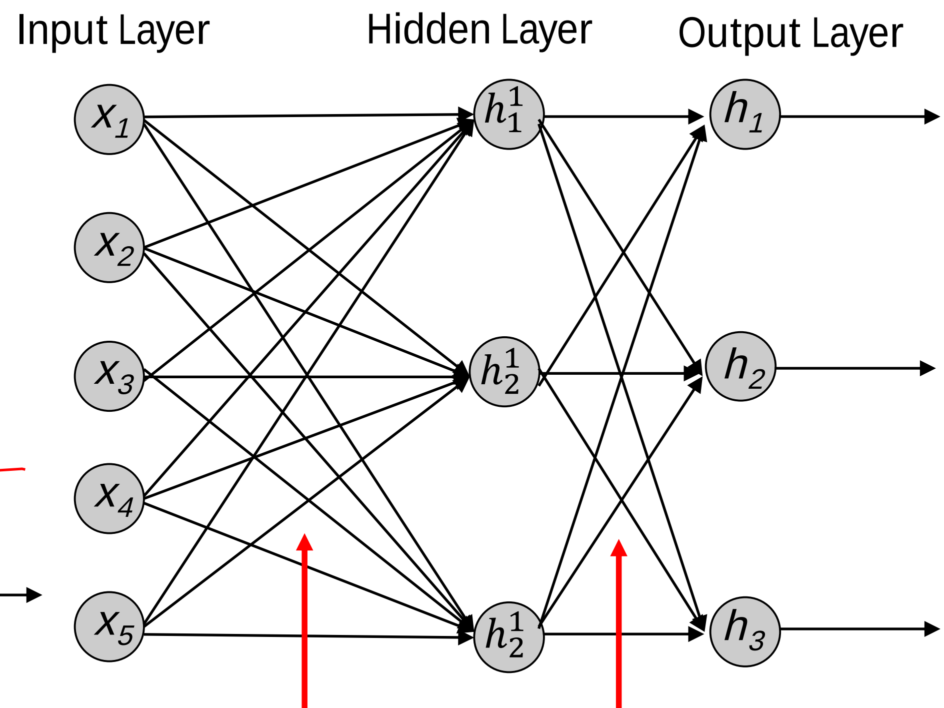
$$h^i = g(\Theta^{(i)} x)$$

$$g(z) = \frac{1}{1 + \exp(-z)}$$



output

$$h_\Theta(\mathrm{x}) = g(\Theta^{(2)} a)$$



weights

$$\Theta^{(1)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{15} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{35} \end{pmatrix} \qquad \Theta^{(2)} = \begin{pmatrix} \theta_{11} & \cdots & \theta_{13} \\ \vdots & \ddots & \vdots \\ \theta_{31} & \cdots & \theta_{33} \end{pmatrix}$$