Roadmap

Feedforward neural networks

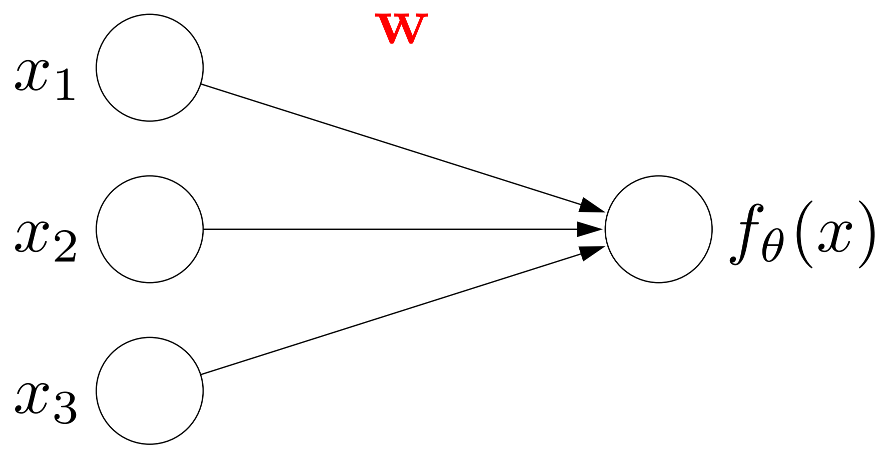
Convolutional networks

Sequence models

Unsupervised learning

Final remarks

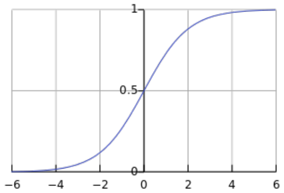
Review: linear predictors



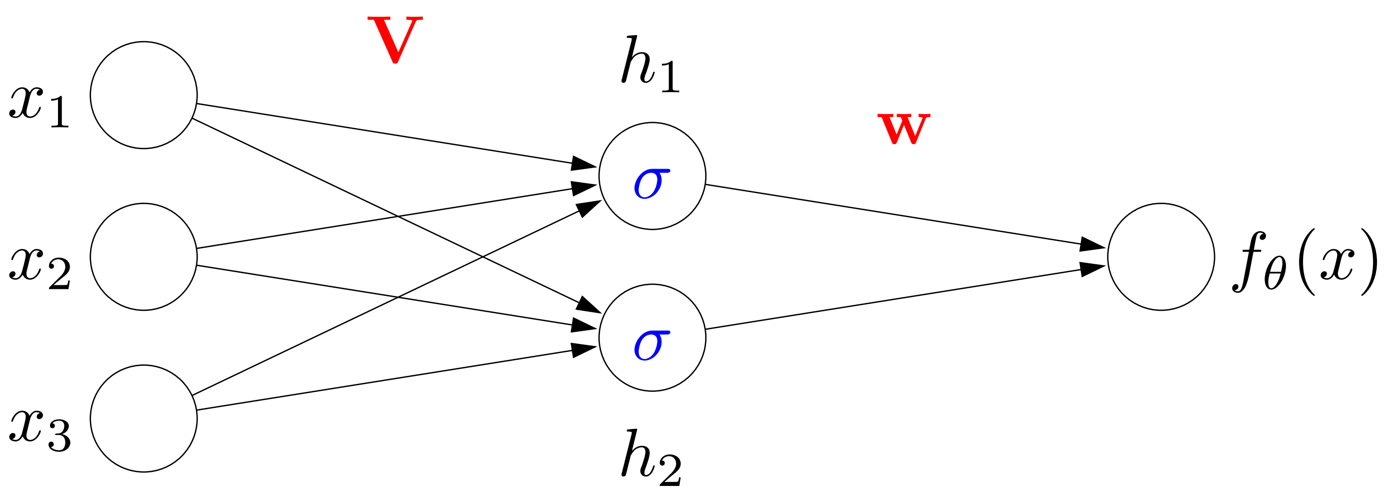
Output:

$$f_\theta(x) = \mathbf{w} \cdot x$$

Parameters: $\theta = \mathbf{w}$



Review: neural networks



Intermediate hidden units:

$$h_j(x) = \sigma(\mathbf{v}_j \cdot x) \quad \sigma(z) = (1 + e^{-z})^{-1}$$

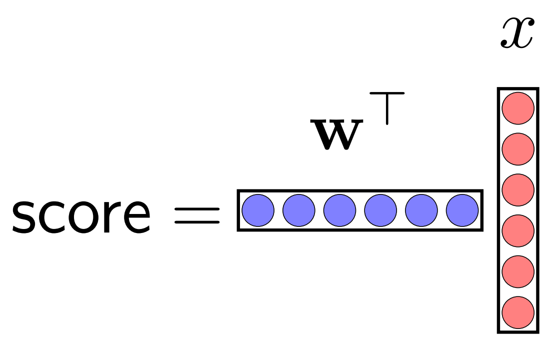
Output:

$$f_\theta(x) = \mathbf{w} \cdot \mathbf{h}(x)$$

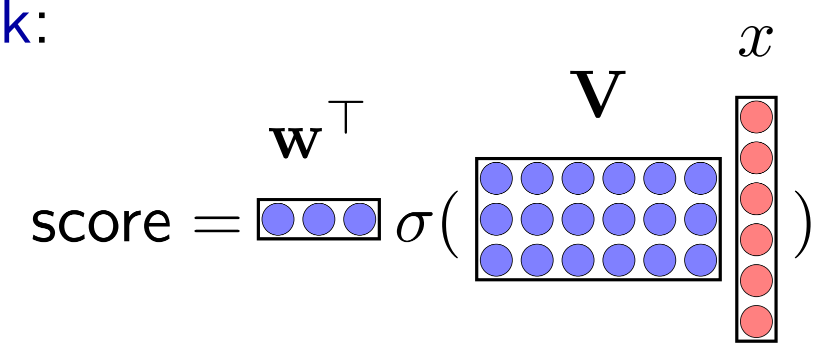
Parameters: $\theta = (\mathbf{V}, \mathbf{w})$

Deep neural networks

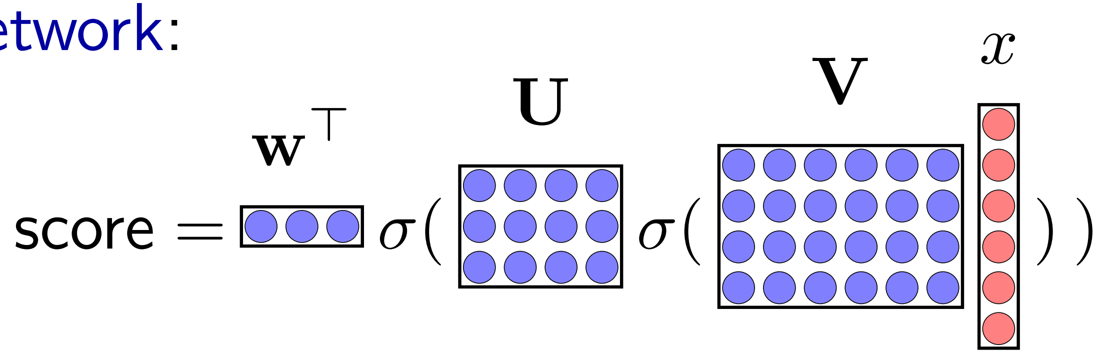
1-layer neural network:



2-layer neural network:

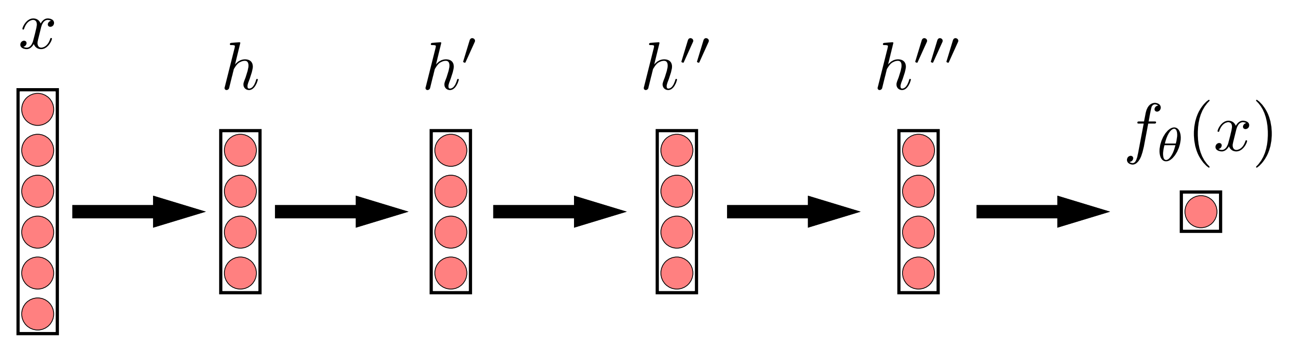


3-layer neural network:



...

Depth

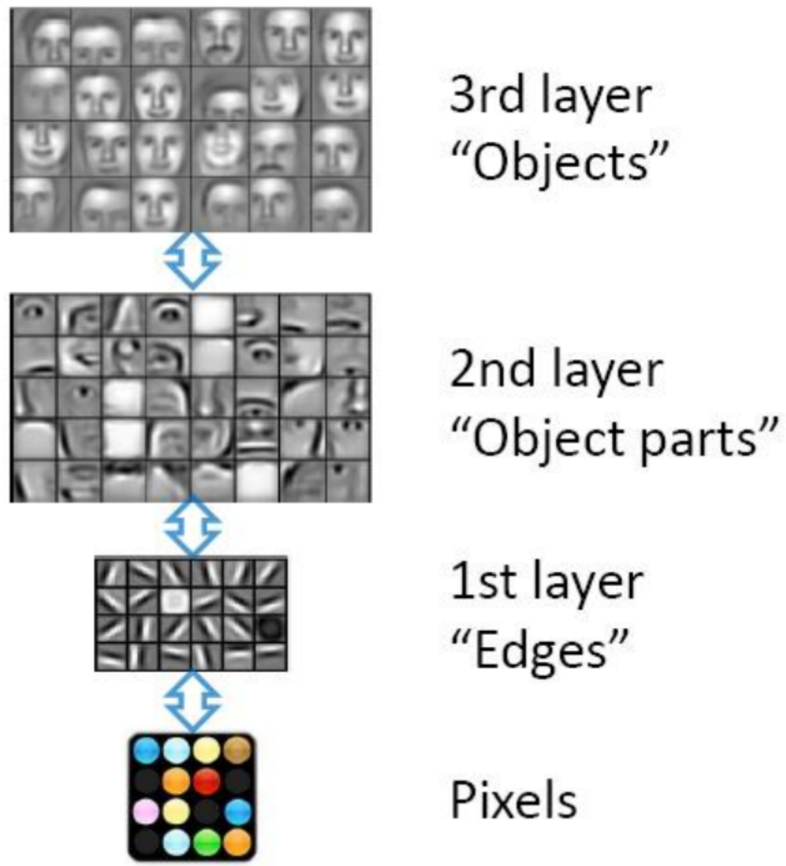


Intuitions:

- Hierarchical feature representations
- Can simulate a bounded computation logic circuit (original motivation from McCulloch/Pitts, 1943)
- Learn this computation (and potentially more because networks are real-valued)
- Formal theory/understanding is still incomplete
- Some hypotheses emerging: double descent, lottery ticket hypothesis

[figure from Honglak Lee]

What's learned?



Review: optimization

Regression:

$$\text{Loss}(x, y, \theta) = (f_\theta(x) - y)^2$$

Key idea: minimize training loss

$$\text{TrainLoss}(\theta) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{(x,y) \in \mathcal{D}_{\text{train}}} \text{Loss}(x, y, \theta)$$

$$\min_{\theta \in \mathbb{R}^d} \text{TrainLoss}(\theta)$$

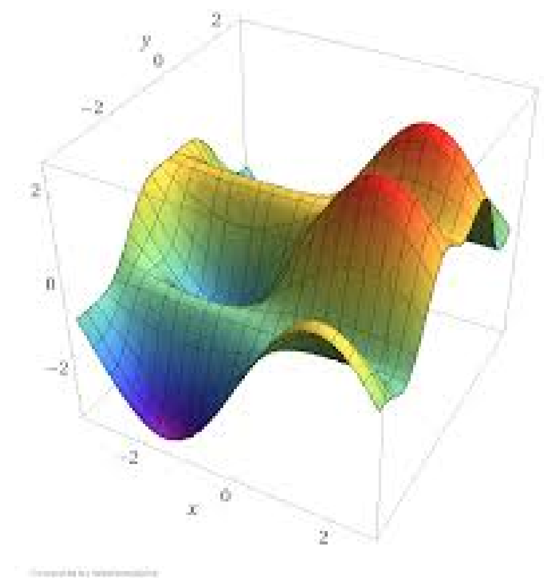
Algorithm: stochastic gradient descent

For $t = 1, \ldots, T$:

For $(x, y) \in \mathcal{D}_{\text{train}}$:

$\theta \leftarrow \theta - \eta_t \nabla_\theta \text{Loss}(x, y, \theta)$

Training



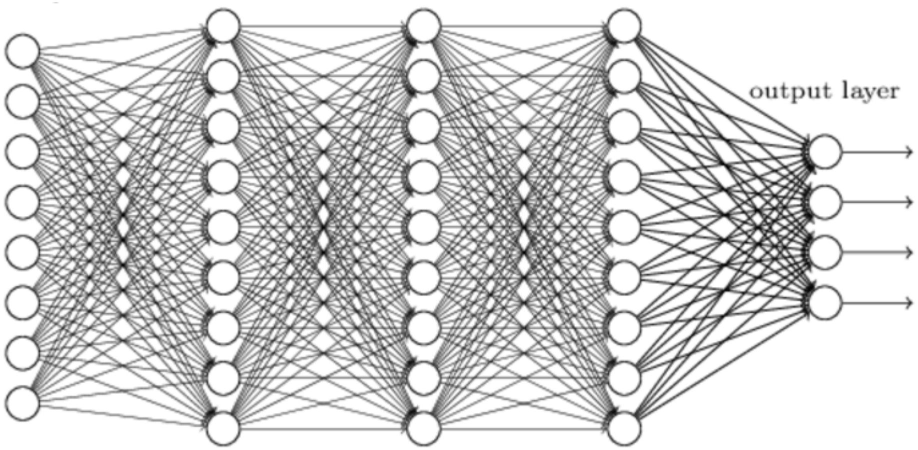
- Non-convex optimization
- No theoretical guarantees that it works
- Before 2000s, empirically very difficult to get working

What's different today

Computation (time/memory)

Information (data)

How to make it work



- More hidden units (over-parameterization)
- Adaptive step sizes (AdaGrad, Adam)
- Dropout to guard against overfitting
- Careful initialization (pre-training)
- Batch normalization

Model and optimization are tightly coupled

Summary

- Deep networks learn hierarchical representations of data

- Train via SGD, use backpropagation to compute gradients

- Non-convex optimization, but works empirically given enough compute and data