Deep Learning

Andrea Danyluk
April 21, 2017

Parts of this, as well as parts of the earlier neural net slides, were adapted from Tom Mitchell.
Other parts draw from *Deep Learning* by Goodfellow, Bengio, and Courville

Announcements

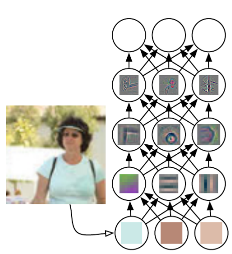
- Reading assignment (and paper response) for Monday: "Do Convolutional Nets Need to Be Deep and Convolutional?"

Today's Lecture

- Deep Learning

What is Deep Learning?

- Represents the world as a nested hierarchy of concepts
  - Each concept defined in relation to simpler concepts
  - More abstract representations computed in terms of less abstract ones
- An artificial neural network with many layers
- Success generally not due to simply to the fact that they have many layers
  - Autoencoding
  - Convolution
  - Recurrence



From *Deep Learning*, Goodfellow, Bengio, Courville, page 6

Long History

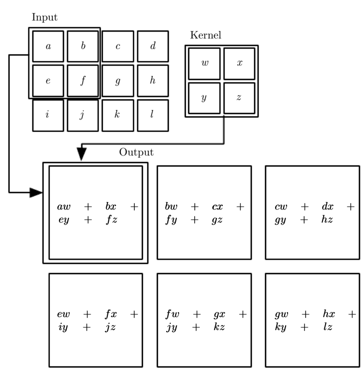
- Foundational work done in the 1900s
  - 1940s-1960s: Cybernetics [McCulloch and Pitts 1943, Hebb 1949, Rosenblatt 1958]
  - 1980s-mid 1990s: Connectionism [Rumelhart 1986]
  - 1990s: modern convolutional networks [LeCun et al. 1998], LSTM [Hochreiter & Schmidhuber 1997, MNIST and other large datasets]
- Recent success due to
  - Availability of data
  - Availability of computing power
  - Continued work on algorithms and theory in this area

Architectural Choices

As with artificial neural networks generally:
- Number of layers
- Number and type of hidden and output units
- Connectivity

Convolutional Networks

- For data that has a known grid-like topology
  - Time series data (a 1-D grid taking samples at regular intervals)
  - Image data (2-D grid of pixels)
- Use convolution in place of general matrix multiplication* in at least one of their layers
- Most convolutional networks also make use of pooling



Convolution in Convolutional Net Terminology

- First argument is the input – for example, a segment of a 2-D image
- Second argument is the kernel – for example, a 2-D matrix
  - The entries in the matrix are parameters adapted by the learning algorithm
- Output is sometimes referred to as the feature map

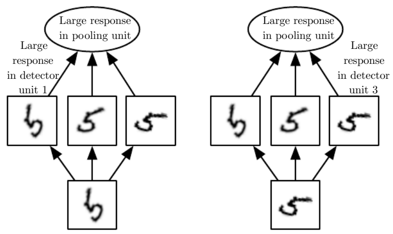
Benefits of Convolution

Convolution leverages three ideas that can help a learning system:
- Sparse interactions
  - Accomplished by making the kernel smaller than the input
- Parameter sharing
  - Rather than learning a separate set of parameters for every location, we learn only one set
- Equivariant representations
  - Parameter sharing causes the network layer to be equivariant to translation

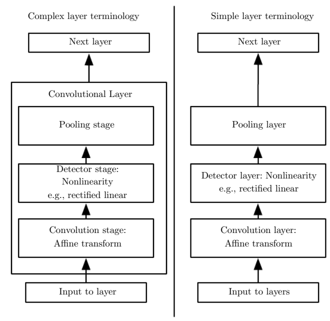
Pooling

A typical convolutional net layer consists of three stages
- Perform several convolutions in parallel to produce a set of linear activations
- Run each linear activation through a nonlinear activation function
- Apply a pooling function
  - Replaces the output at a location with a summary statistic of the nearby outputs
  - Helps to make the representation approximately invariant to small translations of the input

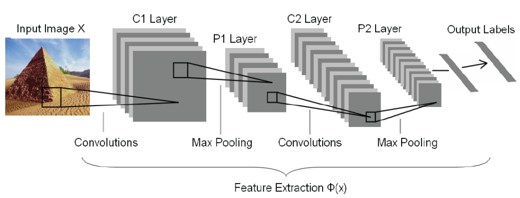
How Pooling Helps with Translation



A Full Convolutional Layer



A Convolutional Network

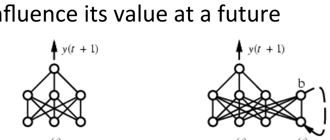


Questions

- What is the smallest size kernel that could reasonably serve as an edge detector?
- How are the kernel parameters learned?

Recurrent Networks

- Specialized for processing sequences of values $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^d$
- Use hidden layer in the network to capture state history.
- Cycles in the graph allow the present value of a variable to influence its value at a future time step.

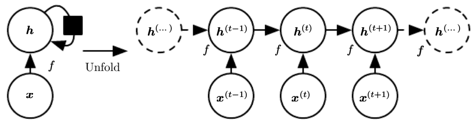


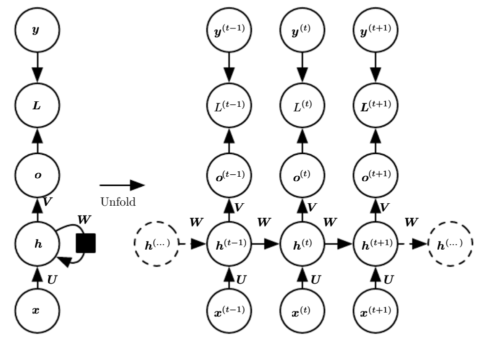
(a) Feedforward network (b) Recurrent network

Recurrent Networks

- Specialized for processing sequences of values $\boldsymbol{x}^1, \boldsymbol{x}^2, \boldsymbol{x}^3, \ldots, \boldsymbol{x}^d$
- We refer to recurrent networks as operating on sequences of such vectors.
- Actually, usually operate on minibatches of such sequences, with a different sequence length d for each sequence in the minibatch.
- Add a special “end of sequence” symbol, to the end of each sequence.

Unfolding





Shared Parameters

- Relies on the assumption that the same parameters can be used for different time steps.
- Assumes the conditional probability distribution over the variables at time t+1 given the variables at time t is stationary.

The Challenge of Long-Term Dependencies

- Gradients propagated over many stages tend to either vanish (most of the time) or explode (rarely, but with much damage).
- Exponentially smaller weights given to long-term interactions.
- Solutions include:
  - Gradient clipping.
  - Use ReLU (rectified linear unit) rather than sigmoid or tanh.
  - Long Short Term Memory (LSTM): weight self-loops, conditioned on the context

An LSTM Cell