Neural Networks - Part 1

Wenhu Chen

Lecture 6

Outline

Learning Goals

- Describe the simple mathematical model of a neuron.
- Describe desirable properties of an activation function.
- Distinguish feedforward and recurrent neural networks.
- Learn a perceptron that represents a simple logical function.
- Determine the logical function represented by a perceptron.
- Explain why a perceptron cannot represent the XOR function.
- Understanding recurrent neural networks.

Learning complex relationships

- Image interpretation, speech recognition, and translation.
- The relationship between inputs and outputs can be extremely complex.
- How can we build a model to learn such complex relationships?

$\rightarrow$ We need a model that can learn complex relationships, that can be learned efficiently, and does not overfit the data.

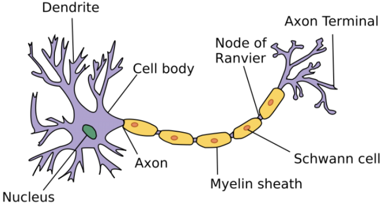
Humans can learn complex relationships well.

Can we build a model that mimics the human brain?

Human brains

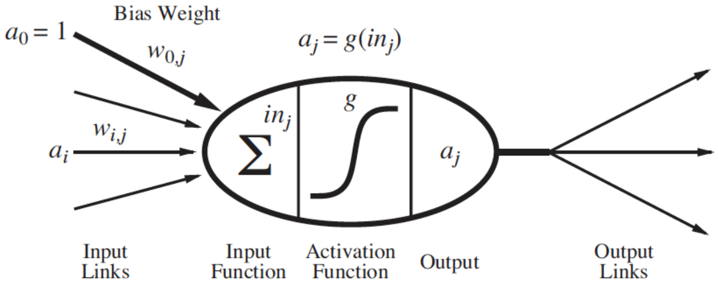
- A brain is a set of densely connected neurons.
- Components of a neuron: dendrites, soma, axon, synapse
  - *Dendrites* receive input signals from other neurons.
  - *Soma* controls activity of the neuron.
  - *Axon* sends output signals to other neurons.
  - *Synapses* are the links between neurons.
- Depending on the input signals, the neuron performs computations and decides to fire or not.

→ Conventional models of neurons have few complex components. Neural networks have many simple components.

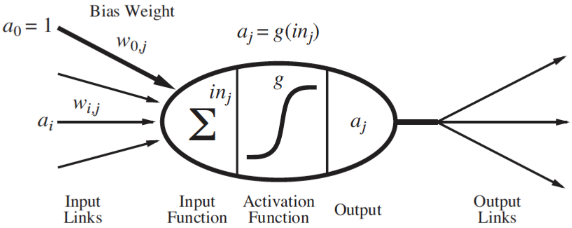


A simple mathematical model of a neuron

- McCulloch and Pitts 1943.
- A linear classfier — it "fires" when a linear combination of its inputs exceeds some threshold.



A simple mathematical model of a neuron



- Neuron j computes a weighted sum of its input signals. $in_j = \sum_{i=0}^{n} w_{ij} a_i$.
- Neuron j applies an activation function g to the weighted sum to derive the output. $a_j = g(in_j) = g\left(\sum_{i=0}^{n} w_{ij} a_i\right)$.

$\rightarrow$ Neuron i sends input signal a_i to neuron j. The link between i and j has weight w_{ij}, which is the strength of the connection. The neuron has a dummy input $a_0 = 1$ with an associated weight w_{0j}.

Desirable Properties of The Activation Function

What are some desirable properties of the activation function?

- It should be nonlinear.

  $\rightarrow$ Complex relationships are often nonlinear. Combining linear functions will not give us a nonlinear function. We can interleave linear and nonlinear functions to represent complex relationships.

Desirable Properties of The Activation Function

What are some desirable properties of the activation function?

- It should be nonlinear.

  $\rightarrow$ Complex relationships are often nonlinear. Combining linear functions will not give us a nonlinear function. We can interleave linear and nonlinear functions to represent complex relationships.

- It should mimic the behaviour of real neurons.

  $\rightarrow$ If the weighted sum of the input signals is large enough, then the neuron fires (sends an output signal of 1). Otherwise, the neuron does not fire (sends an output signal of 0).

Desirable Properties of The Activation Function

What are some desirable properties of the activation function?

- It should be nonlinear.

  $\rightarrow$ Complex relationships are often nonlinear. Combining linear functions will not give us a nonlinear function. We can interleave linear and nonlinear functions to represent complex relationships.

- It should mimic the behaviour of real neurons.

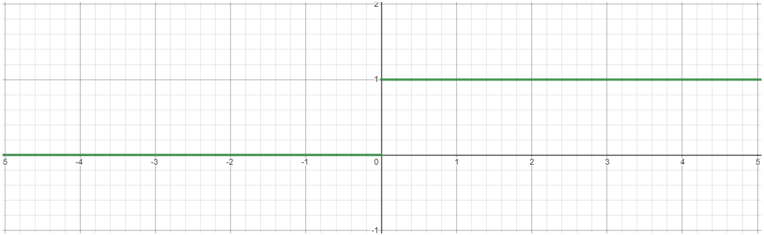
  $\rightarrow$ If the weighted sum of the input signals is large enough, then the neuron fires (sends an output signal of 1). Otherwise, the neuron does not fire (sends an output signal of 0).

- It should be differentiable almost everywhere.

  $\rightarrow$ We learn a neural network using optimization algorithms such as gradient descent. Many such optimization algorithms require a function to be differentiable.

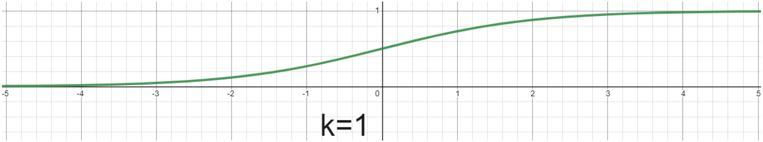
Common activation functions

- **Step function:** $g(x) = 1$ if $x > 0$. $g(x) = 0$ if $x \leq 0$.
  - Simple to use, but not differentiable.
  - Not used in practice, but useful to explain concepts.



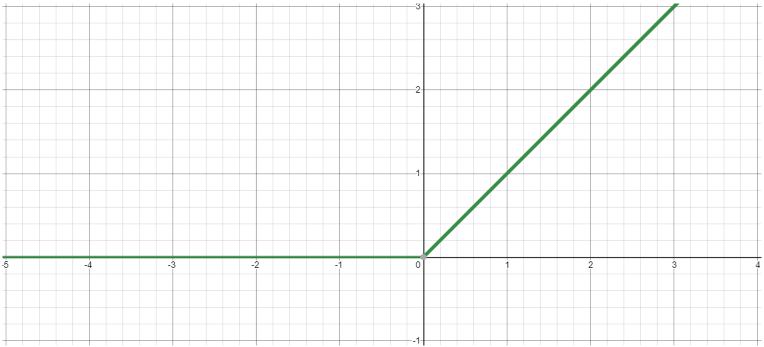
Common activation functions (continued)

- **Sigmoid function:** $g(x) = \dfrac{1}{1 + e^{-kx}}$.
  - For very large or very small x, $g(x)$ is very close to 1 or 0.
  - Can approximate the step function by tuning k. As k increases, the sigmoid function becomes steeper and is closer to the step function. Usually in practice $k = 1$.
  - Differentiable.
  - **Vanishing gradient problem:** when x is very large or very small, $g(x)$ responds little to changes in x. The network does not learn further or learns very slowly.
  - Computationally expensive.



Common activation functions (continued)

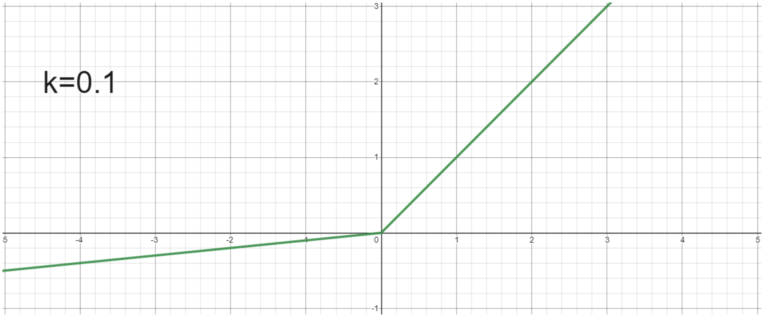
- **Rectified linear unit (ReLU):** $g(x) = max(0, x)$.
  - Computationally efficient; network converges quickly.
  - Differentiable.
  - **The dying ReLU problem:** when inputs approach 0 or are negative, the gradient becomes 0 and the network cannot learn.



Common activation functions (continued)

- **Leaky ReLU:** $g(x) = max(0, x) + k \cdot min(0, x)$

  $\rightarrow$ Small positive slope k in the negative area. Enables learning for negative input values.



Connecting the neurons together into a network

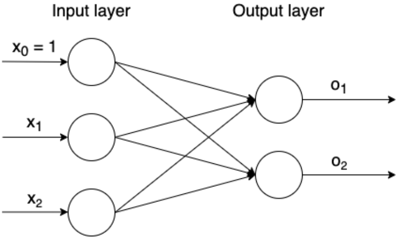
- **Feedforward network**
  - Forms a directed acyclic graph (no loops).
  - Have connections only in one direction.
  - Represents a function of its inputs.

- **Recurrent network**
  - Feeds its outputs back into its inputs.
  - Can support short-term memory. For the given inputs, the behaviour of the network depends on its initial state, which may depend on previous inputs.
  - The model is more interesting, but more difficult to understand and to learn.

Perceptrons

- Single-layer feedforward neural network
- The inputs are connected directly to the outputs.
- Can represent some logical functions, e.g. AND, OR, and NOT.

  $\rightarrow$ A big deal at the time. People believed that AI is solved if computers could perform formal logical reasoning.

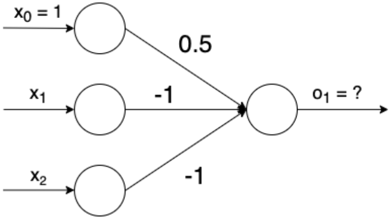


$\rightarrow$ Learn the perceptron for each output separately.

Q: What does the perceptron compute?

Q #1: Consider the following perceptron, where the activation function is the step function. ($g(x) = 1$ if $x > 0$. $g(x) = 0$ if $x \leq 0$.). Which of the following logical functions does the perceptron compute?

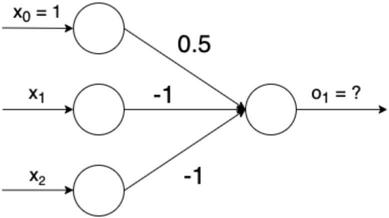
(A) $x_1 \wedge x_2$
(B) $\neg(x_1 \wedge x_2)$
(C) $x_1 \vee x_2$
(D) $\neg(x_1 \vee x_2)$



Q: What does the perceptron compute?

Q #1: Consider the following perceptron, where the activation function is the step function. ($g(x) = 1$ if $x > 0$. $g(x) = 0$ if $x \leq 0$.). Which of the following logical functions does the perceptron compute?

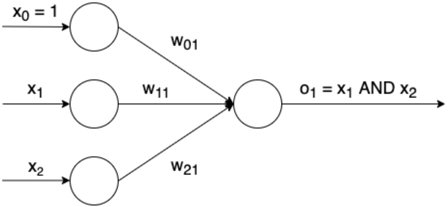
(A) $x_1 \wedge x_2$
(B) $\neg(x_1 \wedge x_2)$
(C) $x_1 \vee x_2$
(D) $\neg(x_1 \vee x_2)$



→ (D) is correct. This perceptron computes a NOR gate.

Q: Learning a perceptron for the AND function

Q #2: Consider the perceptron below where the activation function is the step function ($g(x) = 1$ if $x > 0$. $g(x) = 0$ if $x \leq 0$.). What should the weights w_{01}, w_{11} and w_{21} be such that the perceptron represents an AND function?



x_1	x_2	o_1
0	0	0
0	1	0
1	0	0
1	1	1

(A) $w_{01} = -1$. $w_{11} = 0.5$. $w_{21} = 0.5$

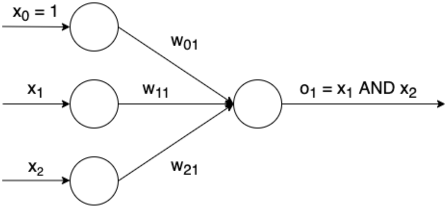
(B) $w_{01} = 0.5$. $w_{11} = -1$. $w_{21} = 1$

(C) $w_{01} = 1.5$. $w_{11} = -1$. $w_{21} = -1$

(D) $w_{01} = -1.5$. $w_{11} = 1$. $w_{21} = 1$

Q: Learning a perceptron for the AND function

Q #2: Consider the perceptron below where the activation function is the step function ($g(x) = 1$ if $x > 0$. $g(x) = 0$ if $x \leq 0$.). What should the weights w_{01}, w_{11} and w_{21} be such that the perceptron represents an AND function?



x_1	x_2	o_1
0	0	0
0	1	0
1	0	0
1	1	1

(A) $w_{01} = -1.\ w_{11} = 0.5.\ w_{21} = 0.5$

(B) $w_{01} = 0.5.\ w_{11} = -1.\ w_{21} = 1$

(C) $w_{01} = 1.5.\ w_{11} = -1.\ w_{21} = -1$

(D) $w_{01} = -1.5.\ w_{11} = 1.\ w_{21} = 1$

$\rightarrow$ (D) is correct. $(-1.5)x_0 + 1x_1 + 1x_2 \geq 0.$

Q: What does this perceptron compute?

Q #3: What does h_1 compute?

$$h_1 = g(x_1 + x_2 - 0.5)$$

where $g(x) = 1$ if $x > 0$ and $g(x) = 0$ if $x \leq 0$.

(A) $(x_1 \vee x_2)$

(B) $(x_1 \wedge x_2)$

(C) $(\neg(x_1 \vee x_2))$

(D) $(\neg(x_1 \wedge x_2))$

Q: What does this perceptron compute?

Q #3: What does h_1 compute?

$$h_1 = g(x_1 + x_2 - 0.5)$$

where $g(x) = 1$ if $x > 0$ and $g(x) = 0$ if $x \leq 0$.

(A) $(x_1 \vee x_2)$

(B) $(x_1 \wedge x_2)$

(C) $(\neg(x_1 \vee x_2))$

(D) $(\neg(x_1 \wedge x_2))$

$\rightarrow$ (A) is the correct answer. This perceptron computes an OR gate.

Q: What does this perceptron compute?

Q #4: What does h_2 compute?

$$h_2 = g(-x_1 - x_2 + 1.5)$$

where $g(x) = 1$ if $x > 0$ and $g(x) = 0$ if $x \leq 0$.

(A) $(x_1 \vee x_2)$

(B) $(x_1 \wedge x_2)$

(C) $(\neg(x_1 \vee x_2))$

(D) $(\neg(x_1 \wedge x_2))$

Q: What does this perceptron compute?

Q #4: What does h_2 compute?

$$h_2 = g(-x_1 - x_2 + 1.5)$$

where $g(x) = 1$ if $x > 0$ and $g(x) = 0$ if $x \leq 0$.

(A) $(x_1 \lor x_2)$

(B) $(x_1 \land x_2)$

(C) $(\neg(x_1 \lor x_2))$

(D) $(\neg(x_1 \land x_2))$

$\rightarrow$ (D) is the correct answer. This perceptron computes a NAND gate.

Learning Goals

Introduction to Artificial Neural Networks

Introduction to Perceptrons

Limitations of Perceptrons

Convolutional Neural Network

Recurrent Neural Network

Revisiting Learning Goals

Limitations of perceptrons

- Perceptrons: An introduction to computational geometry. Minsky and Papert. MIT Press. Cambridge MA 1969.

  → Marvin Minsky (founder of MIT AI lab)

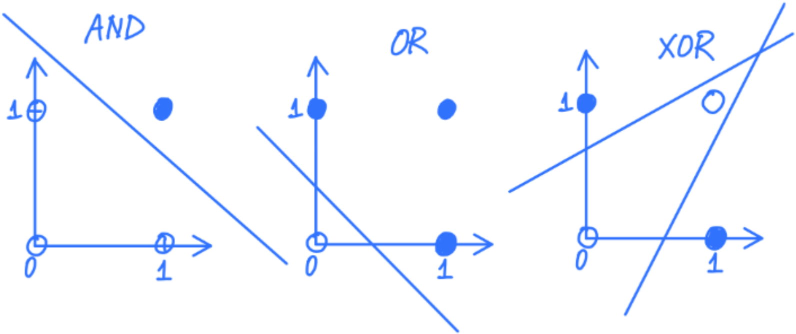
  Seymour Papert (director of the lab).

- Results:
  - XOR cannot be represented using perceptrons.
    We need a deeper network.
  - No one knew how to train deeper networks.
- Led to the first AI winter.

  → This approach was a dead end. A freeze to funding and publications.

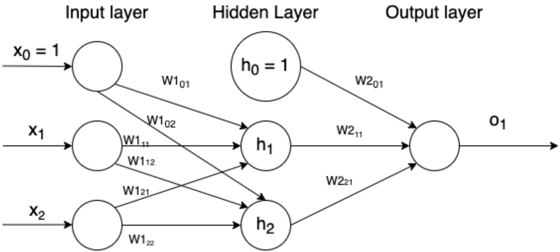
Why can't a perceptron represent XOR?

→ Intuition: a perceptron is a linear classifier. XOR is not linearly separable.



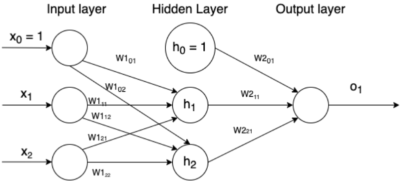
XOR as a 2-Layer Neural Network

Can you come up with the weights such that the following network represents the XOR function?



→ Hint: Start by converting XOR to logical formula composed of logical operations that can be represented by a perceptron:
$o_1 = (x_1 \lor x_2) \land (\neg(x_1 \land x_2)) = h_1 \land h_2$

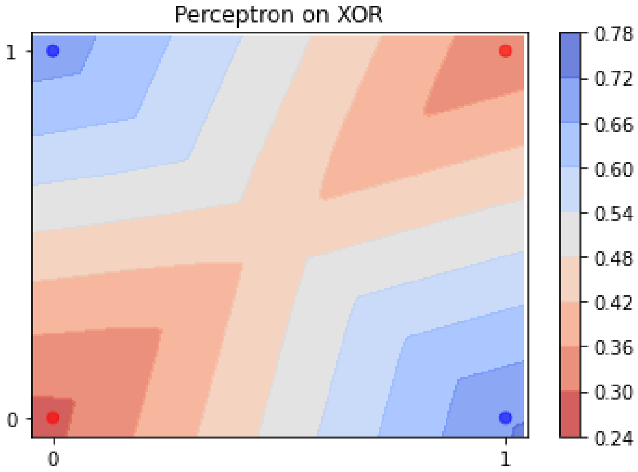
XOR as a 2-Layer Neural Network



$$o_1 = (x_1 \vee x_2) \wedge (\neg(x_1 \wedge x_2)) = h_1 \wedge h_2$$

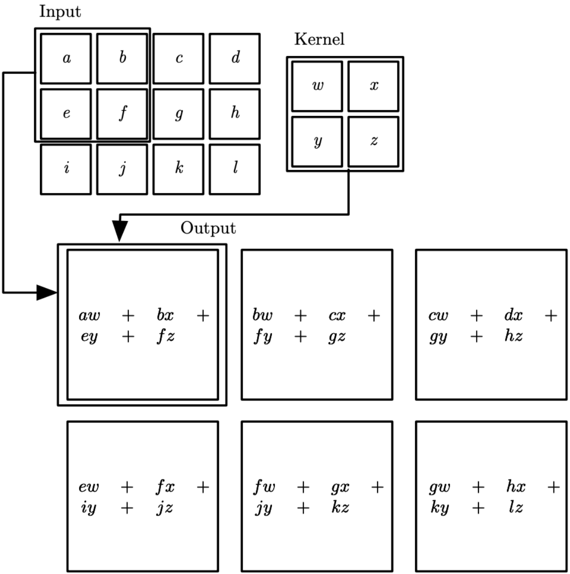
XOR function

The decision boundary of XOR network:



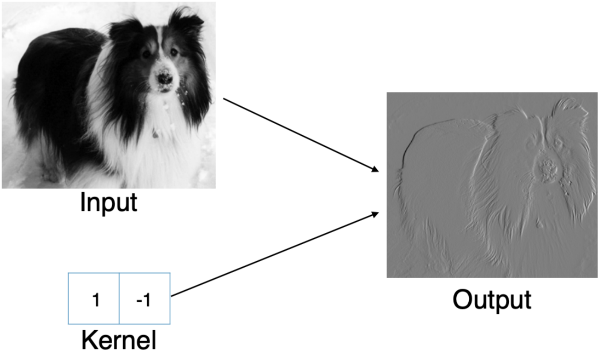
Convolution Kernel

Convolutional Kernal:



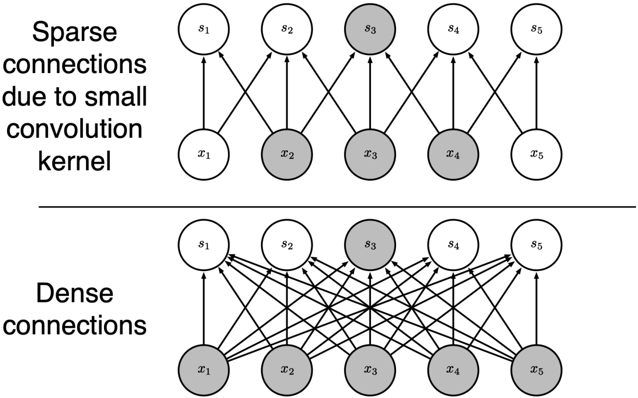
Convolution Kernel

Convolutional 1x1 Kernal:

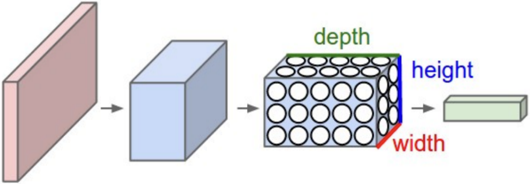


Sparse Connection

Sparse Connection to share weights:

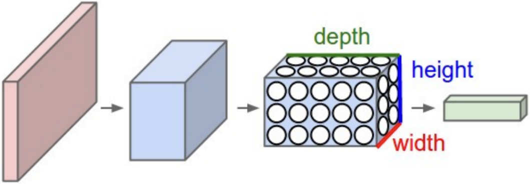


Convolutional Neural Networks



- A ConvNet is made of layers
- Every layer transforms an input 3D volume to an output 3D volume with some function
- Neurons in a layer will only be connected to a small region of the layer

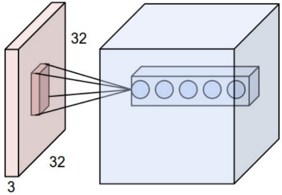
Convolutional Layer



$$o_{h,w,j} = \sum_i \sum_m \sum_n w^j_{m,n,i} I_{h+m,w+n,i} \quad (1)$$

where m and n denotes the offset of the kernel. If the kernel is 3 by 3, then $m \in \{-1, 0, 1\}$ and $n \in \{-1, 0, 1\}$. i denotes the input channel, j denotes the output channel.

Convolutional Layer



Question: we are processing an image of 32x32x3. We use 3x3 kernel with output channel of 8. We slide the window with a stride of 1. What is the parameter size? What is the output dimension?

Convolutional Layer

Question: we are processing an image of 32x32x3. We use 5x5 kernel with output channel of 8. We slide the window with a stride of 1. What is the parameter size? What is the output dimension?

Convolutional Layer

Question: we are processing an image of 32x32x3. We use 5x5 kernel with output channel of 8. We slide the window with a stride of 1. What is the parameter size? What is the output dimension?

Answer: the parameter size is $3 \text{x} 5 \text{x} 5 \text{x} 8 = 600$.

Convolutional Layer

Question: we are processing an image of 32x32x3. We use 5x5 kernel with output channel of 8. We slide the window with a stride of 1. What is the parameter size? What is the output dimension?

Answer: the parameter size is $3 \times 5 \times 5 \times 8 = 600$.

Answer: With the sliding window, the output has a height/width of $32 - 4 = 28$. The 3-d tensor has a depth of 8. So the entire dimensionality is 28x28x8 with a total of 6272 neurons.

Convolutional Layer

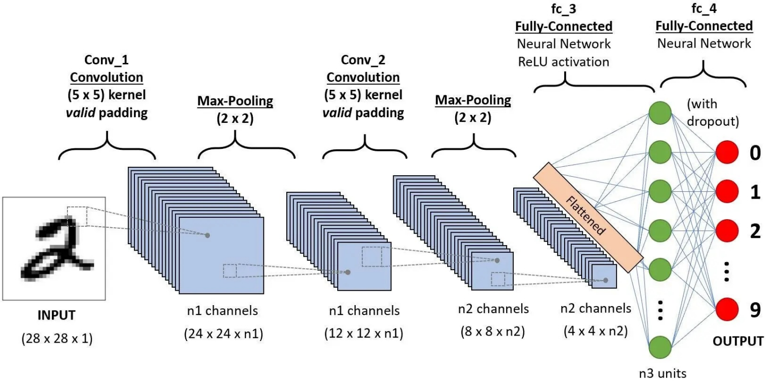
What if I stack another convolution layer of 3x3 with depth of 16 but with stride of 2? What's the output dimensionality?

Convolutional Layer

What if I stack another convolution layer of 3x3 with depth of 16 but with stride of 2? What's the output dimensionality?

Answer: With the sliding window, the output has a height/width of (28 - 2) / 2 = 13. The 3-d tensor has a depth of 4. So the entire dimentionality is 13x13x16 with a total of 2704 neurons.

Convolutional Layer



Stacking multiple layers of conv network will decrease the overall dimensionality. Finally, we can flatten the 3-d tensor to a 1-d vector, which is used to do classification.

Learning Goals

Introduction to Artificial Neural Networks

Introduction to Perceptrons

Limitations of Perceptrons

Convolutional Neural Network

Recurrent Neural Network

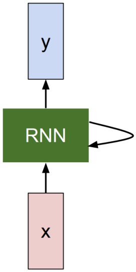
Revisiting Learning Goals

Recurrent Neural Network

We can process a sequence of vectors x by applying a recurrence formula at every time step:

We need to maintain a state variable h_t to keep track of the history of all the seen information.

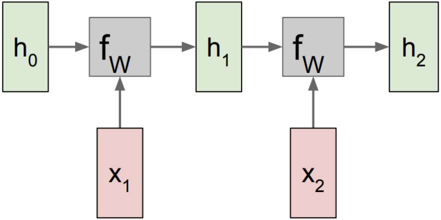
$$h_t = f_W(h_{t-1}, x_t) \quad y_t = f_Y(h_t)$$



Vanilla Recurrent Neural Network

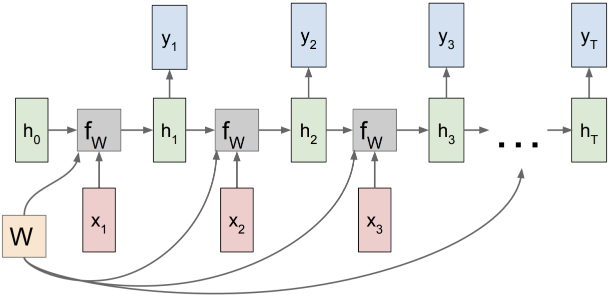
$$h_t = f_W(h_{t-1}, x_t)$$

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t); \quad y_t = W_{hy}h_t$$



Vanilla Recurrent Neural Network

Proceed from $x_1 \cdots x_n$, the weights of the transition is being shared among all the transitions.

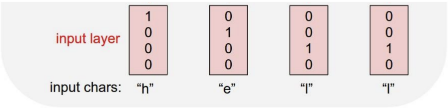


Modeling Language

**Example:
Character-level
Language Model**

Vocabulary:
[h,e,l,o]

Example training
sequence:
“hello”



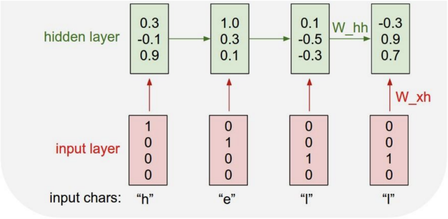
Modeling Language

Example: Character-level Language Model

$$\boxed{h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)}$$

Vocabulary: [h,e,l,o]

Example training sequence: **“hello”**

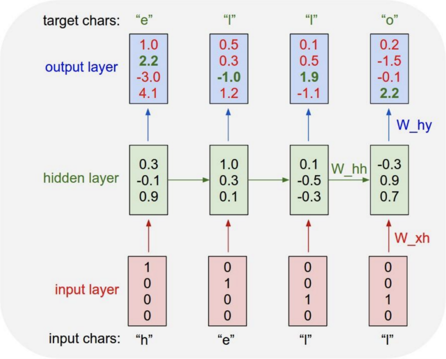


Modeling Language

Example: Character-level Language Model

Vocabulary: [h,e,l,o]

Example training sequence: **"hello"**

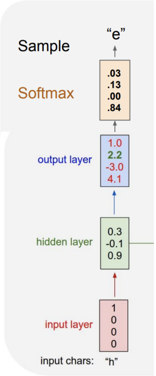


Modeling Language

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

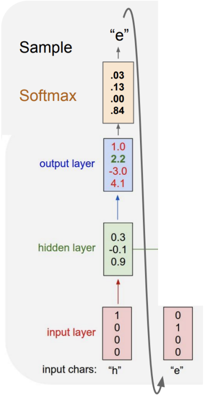


Modeling Language

Example: Character-level Language Model Sampling

Vocabulary:
[h,e,l,o]

At test-time sample characters one at a time, feed back to model

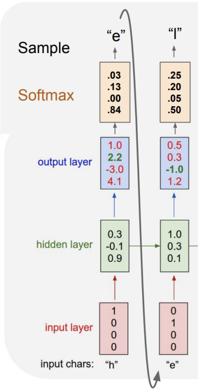


Modeling Language

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model

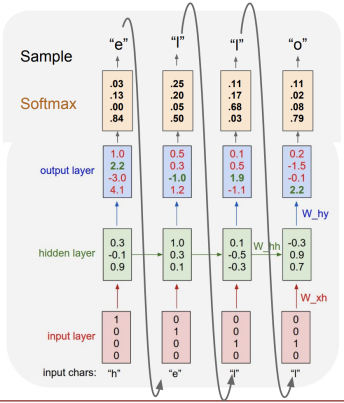


Modeling Language

Example: Character-level Language Model Sampling

Vocabulary: [h,e,l,o]

At test-time sample characters one at a time, feed back to model



Revisiting Learning Goals

- Describe the simple mathematical model of a neuron.
- Describe desirable properties of an activation function.
- Distinguish feedforward and recurrent neural networks.
- Learn a perceptron that represents a simple logical function.
- Determine the logical function represented by a perceptron.
- Explain why a perceptron cannot represent the XOR function.
- Understanding recurrent neural networks.