Machine learning: Part 5

Deep neural networks (DNN)

- Why deep?
- Avoiding vanishing gradients
- Avoiding overfitting

*Slides based on those of Pascal Poupart

Representation matters

- The choice of representation has an enormous effect on the performance of machine learning algorithms.
- This dependence on representations is a general phenomenon that appears throughout computer science and even daily life.
- In computer science, operations such as searching a collection of data can proceed exponentially faster if the collection is structured and indexed intelligently.

Representation learning

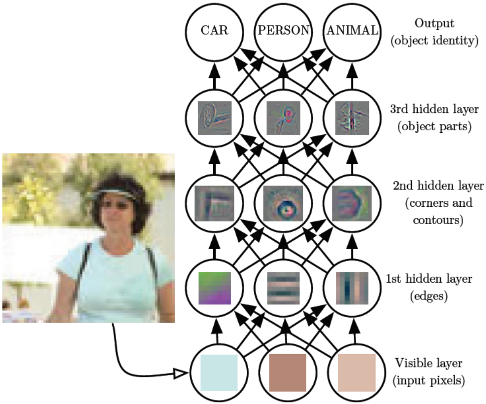
- Many AI tasks can be solved by designing the right set of features to extract for that task, then providing these features to a simple machine learning algorithm.
- For many tasks, however, it is difficult to know what features should be extracted.
- Manually designing features for a complex task requires a great deal of human time and efforts.
- One solution to this problem is to use machine learning to discover not only the mapping from representation to output but also the representation itself.

Hierarchical representations

- When designing features, our goal is usually to separate the factors of variation that explain the observed data.
- Such factors are often not quantities that are directly observed.
- Instead, they may exist as either unobserved objects or unobserved forces in the physical world that affect observable quantities.
- They can be thought of as concepts or abstractions that help us make sense of the rich variability in the data.
- When analyzing a speech recording, the factors of variation include the speaker's age, their sex, their accent and the words they are speaking.

Deep learning

Deep learning enables computers to build complex concepts out of simpler ones.

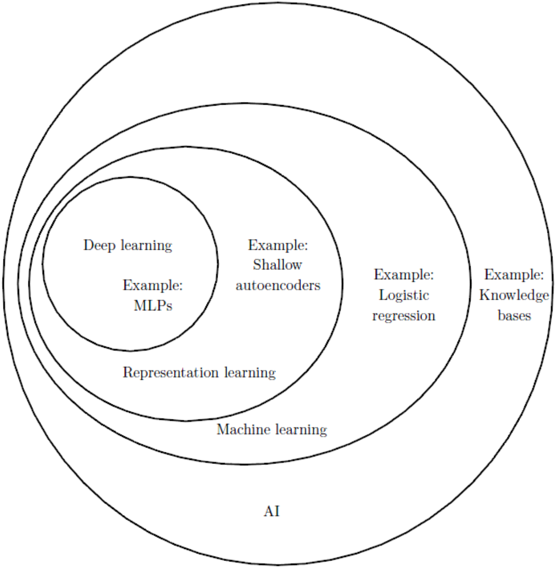


Images here are visualizations of features represented

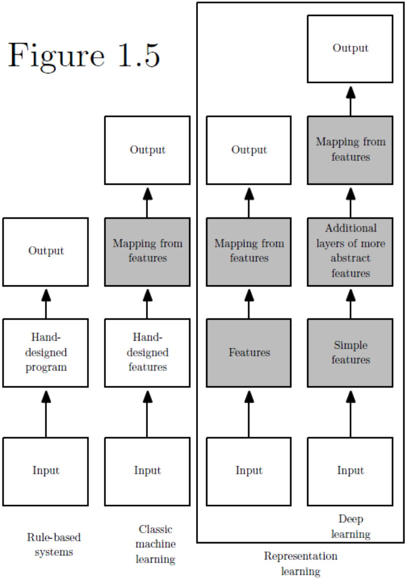
Another perspective

- Depth enables the computer to learn a multistep computer program.
- Each layer of the representation can be thought of as the state of the computer's memory after executing another set of instructions in parallel.
- Networks with greater depth can execute more instructions in sequence.
- Sequential instructions offer great power because later instructions can refer back to the results of earlier instructions.

Machine learning and AI



Machine learning and AI



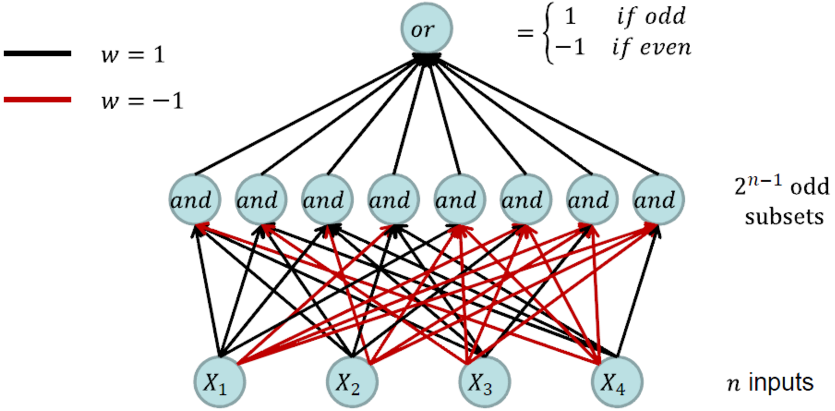
Why deep?

- Universal Approximator Theorem: One hidden layer is enough to represent (not learn) an approximation of any function to an arbitrary degree of accuracy
一致近似理论: 具有至少一个隐层的深层神经网络可以无限逼近任意连续函数

- However as we increase the number of layers, the number of units needed may decrease exponentially (with the number of layers)

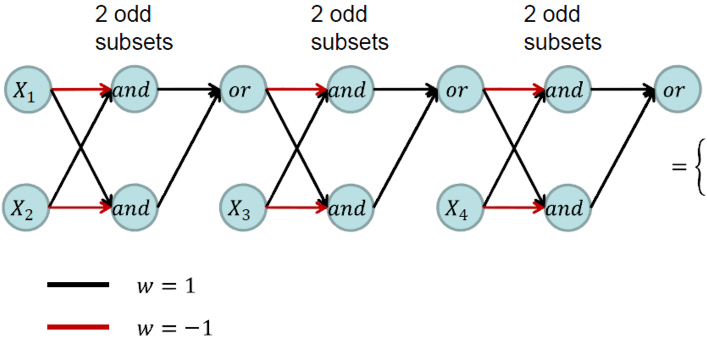
Example – Parity Function

- Single layer of hidden nodes



Example – Parity Function

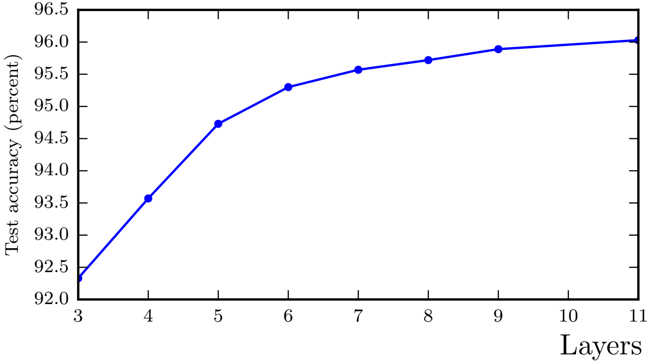
- $2n - 2$ layers of hidden nodes



$$= \begin{cases} 1 & if\ odd \\ -1 & if\ even \end{cases}$$

Better Generalization with Greater Depth

Shallow net may overfit more



The brain has a deep architecture

- For example, the visual cortex is well-studied and shows a sequence of areas each of which contains a representation of the input, and signals flow from one to the next.
- Each level of this feature hierarchy represents the input at a different level of abstraction, with more abstract features further up in the hierarchy, defined in terms of the lower-level ones.

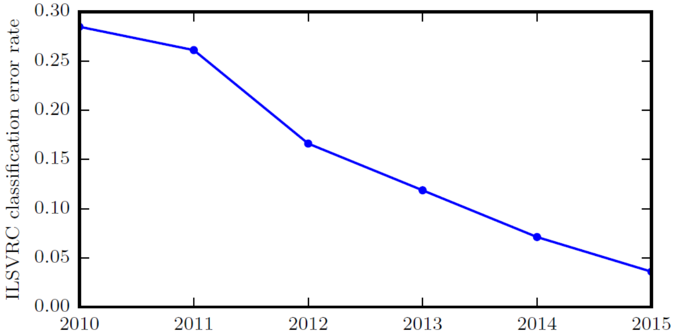
Cognitive processes seem deep

- Humans organize their ideas and concepts hierarchically.
- Humans first learn simpler concepts and then compose them to represent more abstract ones.
- Engineers break-up solutions into multiple levels of abstraction and processing

Breakthrough in Learning Deep Architectures

- Before 2006, attempts at training deep architectures failed
- Three papers changed that in 2006, led by Hinton'ss revolutionary work on Deep Belief Networks
- Key principles are found in all three papers:
  - Unsupervised learning of representations is used to train each layer.
  - The representation learned at each level is the input for the next layer.
  - Use supervised training to fine-tune all the layers

Decreasing error rate over time



ILSVRC: ImageNet Large Scale Visual Recognition Challenge

Many applications of deep learning

- computer vision, speech and audio processing,
- natural language processing, robotics,
- bioinformatics and chemistry, video games,
- search engines, online advertising and finance

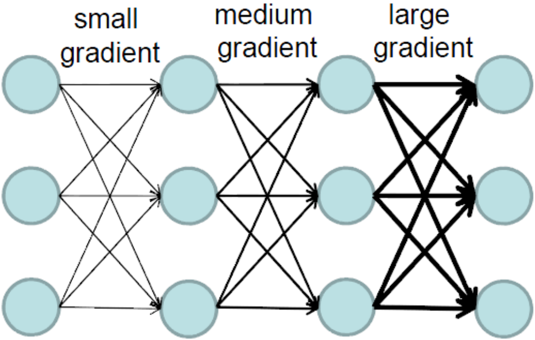
Deep Neural Network

- Definition: neural network with many hidden layers
- Advantage: high expressivity
- Challenges:
  - How should we train a deep neural network?
  - How can we avoid overfitting?

Vanishing gradients (梯度消失)

Gradient descent: $w_{ij} \leftarrow w_{ij} - \alpha \partial Loss / \partial w_{ij}$

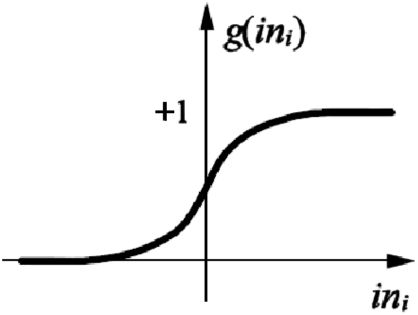
Deep neural networks often suffer from vanishing gradients



Avoiding vanishing gradients: Rectified linear units and maxout units

Sigmoid units

- $g(x) = 1/(1 + e^{-x})$
- $g' = g(1 - g) \in [0, 0.25]$
- Derivative is always less than 1



Simple example

- $Y = \sigma\left(W_4\, \sigma\left(W_3\, \sigma(W_2\, \sigma(W_1\, X))\right)\right)$

$$X \xrightarrow{W_1} H_1 \xrightarrow{W_2} H_2 \xrightarrow{W_3} H_3 \xrightarrow{W_4} Y$$

- Common weight initialization in (-1,1)
- Sigmoid function and its derivative always less than 1
- This leads to vanishing gradients:

$$\frac{\partial Y}{\partial W_4} = \sigma'(in_4)\sigma(in_3)$$

$$\frac{\partial Y}{\partial W_3} = \sigma'(in_4)W_4\sigma'(in_3)\sigma(in_2) \leq \frac{\partial Y}{\partial W_4}$$

$$\frac{\partial Y}{\partial W_2} = \sigma'(in_4)W_4\sigma'(in_3)W_3\sigma'(in_2)\sigma(in_1) \leq \frac{\partial Y}{\partial W_3}$$

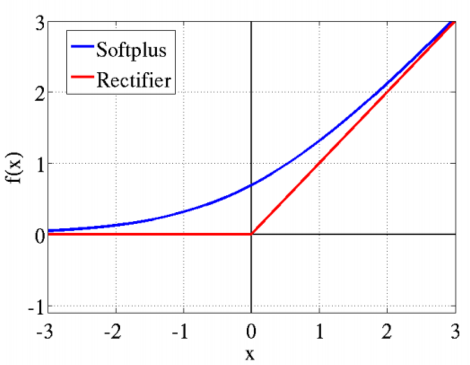
$$\frac{\partial Y}{\partial W_1} = \sigma'(in_4)W_4\sigma'(in_3)W_3\sigma'(in_2)W_2\sigma'(in_1)X \leq \frac{\partial Y}{\partial W_2}$$

12

Since $W_{i+1}\sigma'(in_i) < \frac{1}{4}$, $\frac{\partial Y}{\partial W_i}$ decreases exponentially with i

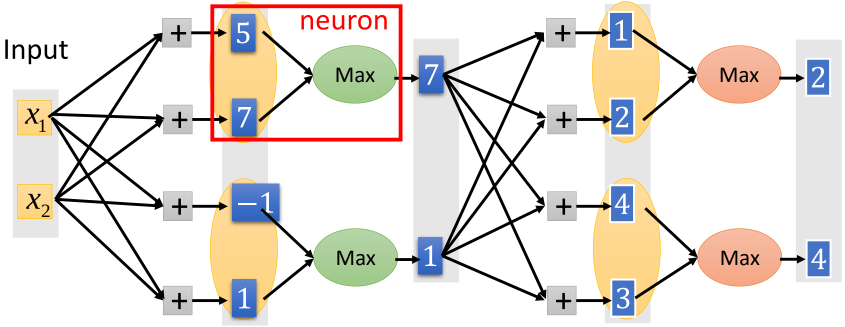
Rectified linear units (ReLU) (整流线性单元)

- $g(x) = max(0, x)$
- Gradient is 0 or 1
- Soft version: Softplus: $g(x) = \log(1 + e^{-x})$



Maxout units

ReLU is a special cases of Maxout



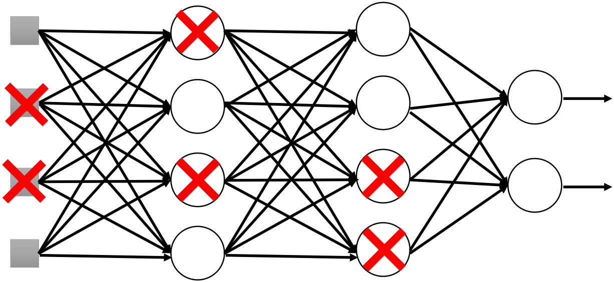
You can have more than 2 elements in a group.

Avoiding overfitting: Dropout

- High expressivity increases the risk of overfitting
    - # of parameters is often larger than the amount of data
- Idea: randomly "drop" some units from the network when training
- Training: at each iteration of gradient descent
    - Each hidden unit is dropped with prob. 0.5
    - Each input unit is dropped with prob. 0.2
- Prediction (testing):
    - Multiply the output of each unit by one minus its drop probability

Dropout

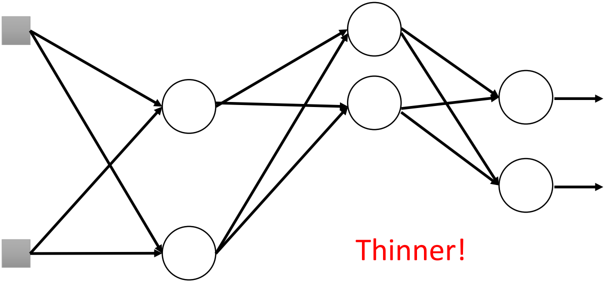
Training:



- **Each time before updating the parameters**
  - Each neuron has p% to dropout

Dropout

Training:

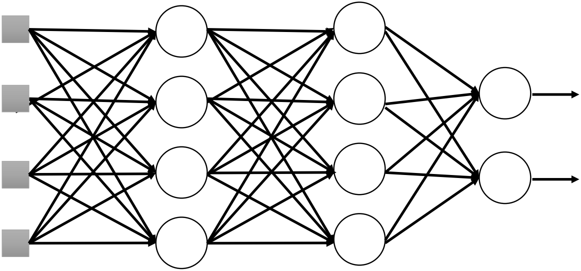


- **Each time before updating the parameters**
  - Each neuron has p% to dropout

    ➡ **The structure of the network is changed.**
  - Using the new network for training

Dropout

Testing:



➢ **No dropout**

- If the dropout rate at training is p%, all the weights times (1-p)%
- Assume that the dropout rate is 50%. If a weight $w = 1$ by training, set $w = 0.5$ for testing.

Intuition

- Ensemble learning takes a number of learning algorithms and combines their output to make a prediction.
- Dropout can be viewed as an approximate form of ensemble learning
- In each training iteration, a different subnetwork is trained
- At test time, these subnetworks are merged by averaging their weights