Neural Network and Deep Learning

Early history of deep learning

- Deep learning history dates back to 1940s: known as cybernetics in the 1940s-60s, connectionism in the 1980s-90s, and under the current name starting in 2006.
- Deep learning has gone by name Artificial Neural Networks (ANNs), one of machine learning methods originally aiming to understand brain function (although nothing to do with actual biological functions; Hinton and Shallice, 1991) The first wave was simple linear models to relate input to output (McCulloch and Pitts, 1943) from a neuroscientific perspective.

Early history: continue

- Deep learning gradually deviates from neuroscience due to lack of knowledge about brain function, but retains common belief that much of the mammalian brain might solve different tasks using a single algorithm (True or False?), including language processing, vision, motion planning and speech recognition.
- The second wave of deep learning (neural network) started in 1980s and lasted till the mid-1990s due to emerging movement called connectionism (parallel distributed processing). However, kernel machines (SVM) and graphical models became dominating later.

Recent development in deep learning

- The explosive revival of deep learning started from Hinton et al. (2006) with its outperforming other machine learning methods.
- Why deep learning becomes more exciting than ever:
  - huge amount of training data, especially data with repetitive structures (image, speech), have lessened the concern on statistical generalizability, the point most criticized by statistical learning communities;
  - more powerful computers and better software infrastructures enable computation with a highly complex model containing parallelizable components.

Deep learning applications

- Deep learning has been successful in recent applications:
  - In the ImageNet Large Scale Visual Recognition Challenge of 2012, it improved the top-5 error rate from 26.1% to 15.3% and further down to 3.6% in 2015.
  - It also had a dramatic impact on speech recognition, resulting a sudden drop in error rates with some cut in half.
  - It showed superhuman performance in traffic sign classification and image segmentation.
  - It is incorporated with reinforcement learning to reach human-level performance (e.g., DeepMind).
- Deep learning has been successfully used in other applications, eg., predicting how molecules interact, searching for subatomic particles and automatically parsing microscope images to construct a 3-D brain map.

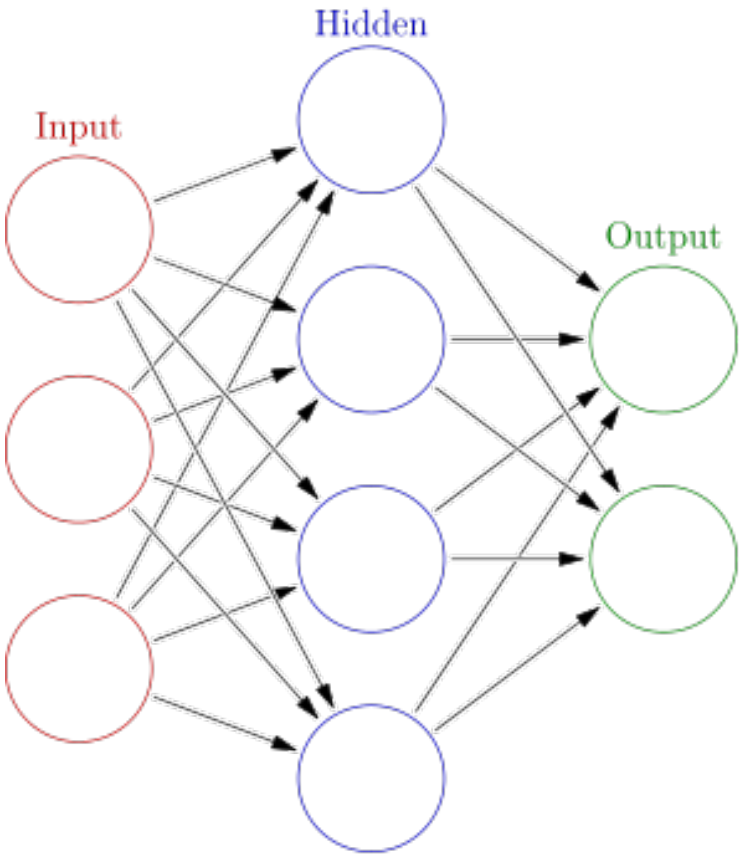
Artificial Neural Networks

- It has other names: deep feedforward networks, feedforward neural networks, or multilayer perceptrons (MLPs).
- They are feedforward since information flows through function from input X to output Y and there are no feedback.
- A typical network is

$$X \to f^{(1)}(X) \to f^{(2)}(f^{(1)}(X)) \to \cdots \to Y.$$

- $f^{(1)}$: first layer; $f^{(2)}$: second layer
  the overall length of the chain is the depth of the model
  all the layers between X and Y are hidden layers.
- The architecture of ANN is closely related to direct acylic graph or structural equation model.

ANN example



Single-layer neural network

- Let $Z_1, ..., Z_m$ be the variables in the hidden layer (hidden units) and $Z_k = h_k(X)$.
- Furthermore, we use $g(Z_1, ..., Z_m)$ to predict Y.
- Questions: how to specify the link functions (also called activation functions) $h_1, ..., h_m$ and g?
- The key motivation is: we want these functions to be simple and conveniently computed, since by increasing the number of Z's, simple functions are sufficient approximate any possibly nonlinear relationship between X and Y.
- **Universal approximation theorem** A feedforward network with a linear output layer and at least one hidden layer with any "squashing" activation functions can approximate any Borel measurable function, provided that the network is given enough hidden units.

Activation functions

- Commonly used function for h's
  - linear function:
$$h(x) = \beta_0 + \beta^T x$$
  - sigmoid function:
$$h(x) = \max\{0, \min\{1, \beta_0 + \beta^T x\}\}$$
  - logit sigmoid function:
$$h(x) = \exp\{\beta_0 + \beta^T x\}/[1 + \exp\{\beta_0 + \beta^T x\}]$$
  - rectified linear function:
$$h(x) = \max(0, \beta_0 + \beta^T x)$$
  - hyperbolic tangent function:
$$h(x) = \tanh(\beta_0 + \beta^T x)$$
- The choice of g can also be one of these link functions depending on the type of output Y.

Network architecture

- One main feature of the architecture is characterized by depth (the number of hidden layers) and width (the number of hidden units at each layer).
- Empirically, greater depth results in better performance as compared to a shallow network with large width.
- Another feature of the architecture is how to connect two neighboring layers
  – input layer can be connected to a subset of the units in the output layer;
  – connection coefficients can be shared across some units in the input layer (convolutional networks);
  – the advantage over a full connected network is the reduced number of parameters and less computation.
- Remark: the architecture of network is task-specific!

Computation algorithm for ANN

- Despite of the complexity in the network architecture, estimation of the parameters can be effectively obtained due to simplicity of the activation function and the so-called forward- and backward-propagation algorithm.
- Suppose $Z_k = \sigma_k(X^T\alpha_k),\ \ k = 1, ..., m$, and

$$E[Y|X] = g(\beta_1 Z_1 + ... + \beta_m Z_m + \beta_0) \equiv f(X).$$

- Based on n observations, we wish to minimize

$$\sum_{i=1}^{n} \{Y_i - g(\beta_1 \sigma_1(X_i^T\alpha_1) + ... + \beta_m \sigma_m(X_i^T\alpha_m))\}^2$$

if Y is continuous, or

$$-\sum_{i=1}^{n} Y_i \log g(\beta_1 \sigma_1(X_i^T\alpha_1) + ... + \beta_m \sigma_m(X_i^T\alpha_m))$$

if Y is binary.

Gradient-descent algorithm

- The key is to compute the gradient with respect to all parameters.
- At $(r+1)$st iteration, from the chain-rule,

$$\beta_k^{(r+1)} = \beta_k^{(r)} - \gamma_r \sum_{i=1}^{n} \delta_i Z_{ki},$$

$$\alpha_{kl}^{(r+1)} = \alpha_{kl}^{(r)} - \gamma_r \sum_{i=1}^{n} s_{ik} X_{il},$$

where γ_r is the step size in the decent algorithm (called learning rate) and $s_{ik} = \sigma_k'(X_i^T \alpha_k)\beta_k \delta_i$, and

$$\delta_i = -2(Y_i - f(X_i))f'(\beta_1 Z_{i1} + ... + \beta_m Z_{im} + \beta_0)$$

for the continuous Y and

$$\delta_i = -Y_i / f(X_i) f'(\beta_1 Z_{i1} + ... + \beta_m Z_{im} + \beta_0)$$

for the binary Y.

Remarks on computation

- The update for the parameters can be carried in two-pass algorithm. In the forward pass, we use the current parameters to estimate $f(\cdot)$; in the backward pass, we compute δ_i then s_{ik}.
- Each hidden unit passes and receives information only to and from units that share a connection, this algorithm can be implemented efficiently on a parallel architecture computer.
- Using the chain rule, we can also develop recursive computation for the hessian matrix.

Improve estimation in ANN

- Parameter regularization: L_2 or L_1 penalization
- Data augmentation: create fake data and augment it to training sample (boostrap sample, add noises?) One particularly effective technique for structured data is to introduce perturbation in data augmentation: for example, for object recognition, we translate images in a few pixels in each direction to improve generalization. Rotation or scaling are also effective.
- Multitask learning: it assumes that some factors are shared across two or more tasks.
- Early stopping, dropout, bagging or other ensemble methods to avoid overfitting

Alternative optimization algorithms

- stochastic gradient descent
- adaptive learning rates
- conjugate gradient method
- Boryden-Fletcher-Goldfarb-Shanno (BFGS) algorithm
- coordinate descent

Convolution Networks

- It is also known as convolutional neural networks (CNNs).
- It involves convolution and pooling operations in neural network.
- It has been quite successful for processing data with grid-like topology, such as time-series data and image data.

Convolution

- Recall convolution operation

$$s(t) = \int x(a)w(t-a)da,$$

  where w is call the convolution kernel.
- In a discrete version, it is $\sum_a x(a)w(t-a)$ for a 2-d image,

$$s(i,j) = \sum_{m,n} x(m,n)w(i-m, j-n).$$

- Discrete convolution can be viewed as multiplication by a kernel matrix with constrained entries (Toeplitz matrix, doubly block circulant matrix)

Features in CNN

- Sparse connectivity: convolution can be treated as another layer network from input data x to the transformed data s. Since kernel matrix is much smaller than input matrix, the connection is sparse.
- Parameter sharing: the parameters for the same input unit are the same for different units in the layer of s.
- Equivariance: if the input changes, the output should changes in the same way. For example, every image pixel is shifted by one unit to the right, the output of ANN should shift by one pixel too (use invariance property to improve prediction).

Pooling operation

- It adds a pooling layer so replaces the input layer data with some summary statistics of the nearby outputs (for example, the maximum value in the neighborhood, or some weighted average).
- Pooling increases robustness to small translation of the input.
- Pooling results in data reduction so improves computation efficiency.

Convolution diagram

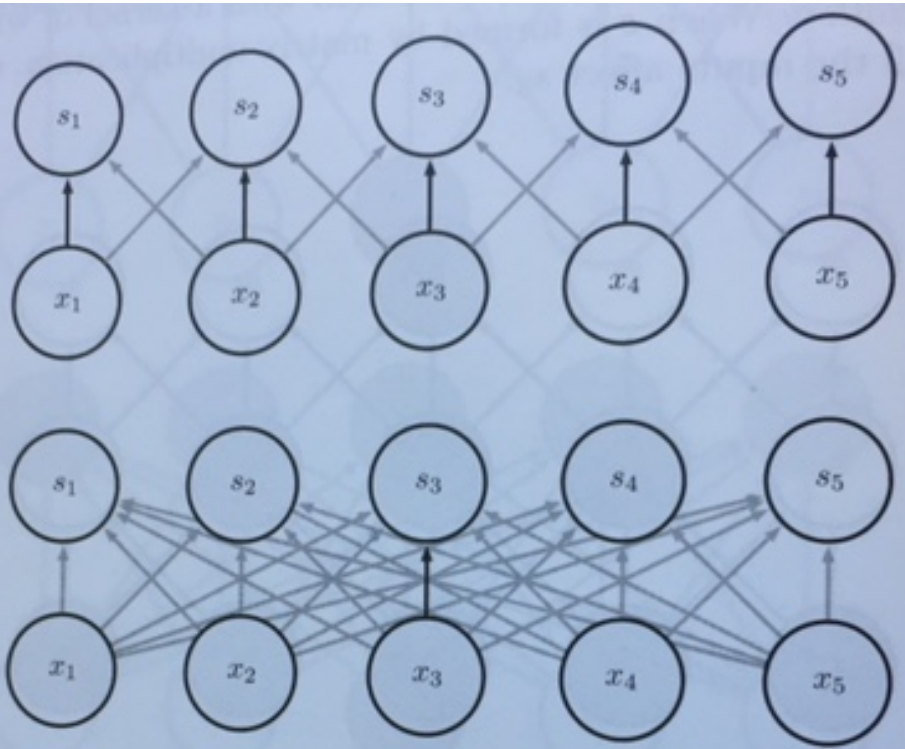


Figure 9.5: Parameter sharing. Black arrows indicate the connections that use a particular parameter in two different models. *(Top)*The black arrows indicate uses of the central element of a 3-element kernel in a convolutional model. Because of parameter sharing, this single parameter is used at all input locations. *(Bottom)*The single black arrow indicates the use of the central element of the weight matrix in a fully connected model. This model has no parameter sharing, so the parameter is used only once.

328

Pooling diagram

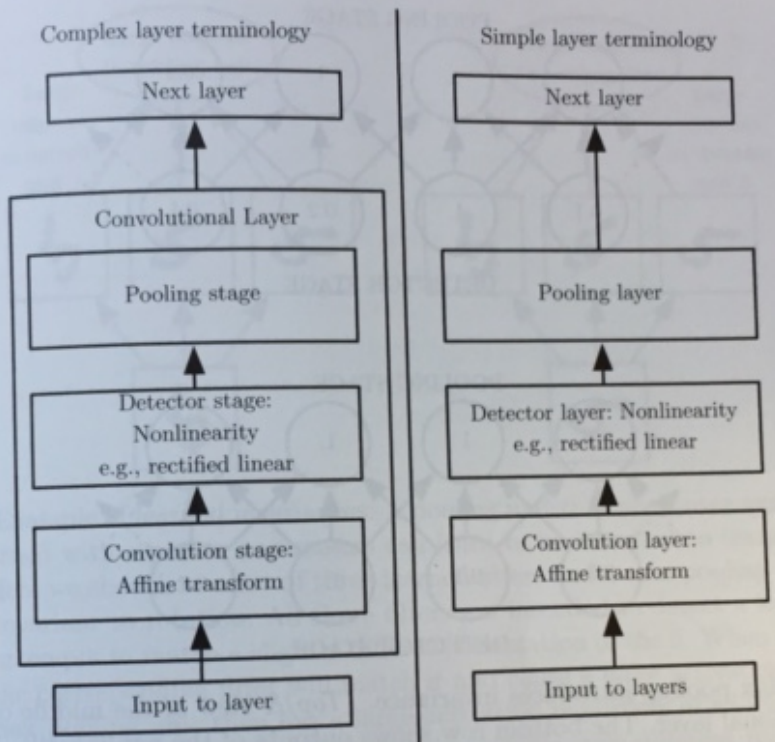


Figure 9.7: The components of a typical convolutional neural network layer. There are two commonly used sets of terminology for describing these layers. *(Left)*In this terminology, the convolutional net is viewed as a small number of relatively complex layers, with each layer having many "stages." In this terminology, there is a one-to-one mapping between kernel tensors and network layers. In this book we generally use this terminology. *(Right)*In this terminology, the convolutional net is viewed as a larger number of simple layers; every step of processing is regarded as a layer in its own right. This means that not every "layer" has parameters.

Additional deep learning

- Recurrent neural network: input data are in a sequence $X^{(1)}, Y^{(1)}, X^{(2)}, \cdots$ and learning uses sharing parameters across different parts of a model.
- The application of recurrent neural networks includes computer vision, speech recognition, natural language processing, contextual bandits in reinforcement learning
- Autoencoders: it is a neural network that is trained to copy its input to its output $(X \to X)$ so is useful for dimension reduction or feature learning. Some variants include sparse autoencoders, denoising autoencoders.
- Representation learning
- Graphical models

Strength and limitation with deep learning

- Strength
  - Hierarchical structure can model very complex function.
  - Simple activation function between layers enable gradient calculation efficient.
  - Such structure is exceptionally useful when processing a large number of training data with abundant computing resources, so makes deep learning really stand out in big data analytics.
- Limitation
  - Evidence-based design of network architecture and many tuning parameters make algorithm difficult to be generalized.
  - Need a large number of training data, which is difficult to ensure data quality and evade bias.
  - Scientific knowledge of general principle is still unclear.