A Brief Introduction to Neural Network Models

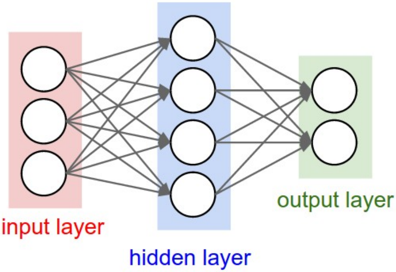
Lei Wu
leiwu@princeton.edu

July 1, 2019

Outline

Outline

Two-layer networks



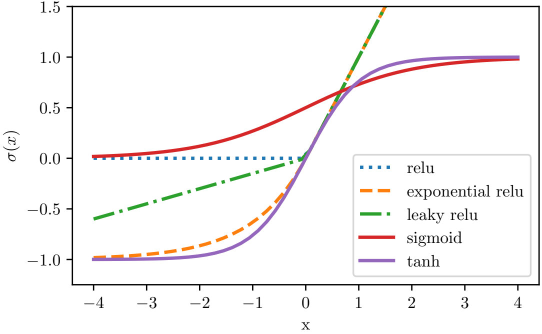
- A two-layer network defines function mapping from $\mathbb{R}^d$ to $\mathbb{R}^k$

$$
\begin{aligned}
f(x) &= \sum_{i=1}^{m} a_k \sigma(b_k \cdot x + c_k) \\
&= A^T \sigma(Bx)
\end{aligned}
$$

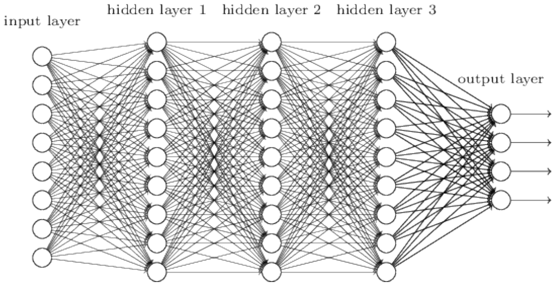
Nonlinear Activation Functions

Saturating	Sigmoid Tanh	$\frac{1}{1+e^{-x}}$ $\frac{e^x-e^{-x}}{e^x+e^{-x}}$
Non-saturating	ReLU Leaky ReLU Parametric ReLU	$\max(0,x)$ $\max(ax,x)$, with $a=0.01$ $\max(ax,x)$, with a learnable

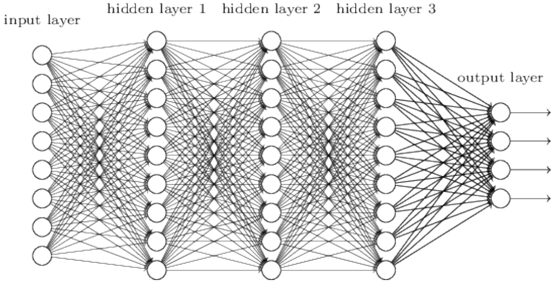
Table: Commonly used activation functions



Multi-layer Fully Connected Networks



Multi-layer Fully Connected Networks



Let

$$x^0 = x$$
$$x^{\ell+1} = \sigma(A^\ell x^\ell + b^\ell),$$

A L-layer network is defined as $f(x) = x^L$. It can also be written as

$$f(x) = A^L \sigma(A^{L-1} \sigma(A^{L-2} \cdots \sigma(A^1 x)))$$

Convolutional Networks

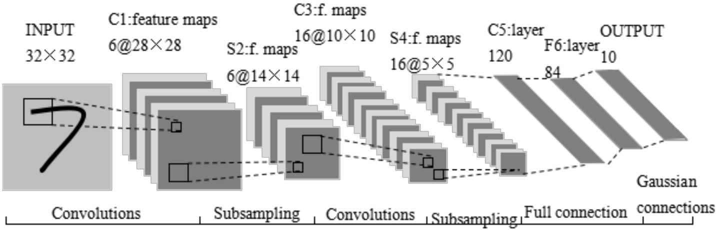


图 2.3 LeNet-5 结构

- Convolutional networks are similar to fully connected networks,

$$f(x) = A^L\sigma(A^{L-1}\sigma(A^{L-2}\cdots\sigma(A^1x))).$$

The only difference is that $A^\ell x = x * w$ is convolutional transformation.

Basic Definition of Recurrent Networks

- **Input:** $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, with $x_t \in \mathbb{R}^{d_x}$.

Basic Definition of Recurrent Networks

- **Input:** $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, with $x_t \in \mathbb{R}^{d_x}$.
- **Code:** $\boldsymbol{h} = (h_1, h_2, \ldots, h_T)$, with $h_t \in \mathbb{R}^{d_h}$ encodes the information of $x_1, x_2, \ldots, x_t$ through

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

Basic Definition of Recurrent Networks

- **Input:** $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, with $x_t \in \mathbb{R}^{d_x}$.
- **Code:** $\boldsymbol{h} = (h_1, h_2, \ldots, h_T)$, with $h_t \in \mathbb{R}^{d_h}$ encodes the information of $x_1, x_2, \ldots, x_t$ through

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

- **Output:** $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$ with

$$y_t = g(y_{t-1}, h_t)$$

Basic Definition of Recurrent Networks

- **Input:** $\boldsymbol{x} = (x_1, x_2, \ldots, x_T)$, with $x_t \in \mathbb{R}^{d_x}$.
- **Code:** $\boldsymbol{h} = (h_1, h_2, \ldots, h_T)$, with $h_t \in \mathbb{R}^{d_h}$ encodes the information of $x_1, x_2, \ldots, x_t$ through

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

- **Output:** $\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$ with

$$y_t = g(y_{t-1}, h_t)$$

- **Parameterization:** Use fully or convolutional networks to parameterize f and g.

Vanilla Recurrent Network

- **Update Formulation:**

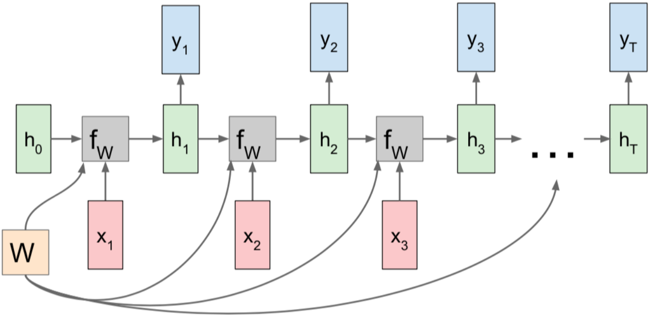
$$
\begin{aligned}
h_t &= \texttt{tanh}(W_{hh} h_{t-1} + W_{hx} x_t) \\
y_t &= W_{yh} h_t
\end{aligned}
$$

Vanilla Recurrent Network

- **Update Formulation:**

$$h_t = \mathtt{tanh}(W_{hh}h_{t-1} + W_{hx}x_t)$$
$$y_t = W_{yh}h_t$$

- **Visualization:**



Long Short Term Memory (LSTM)

- **Update Formulation:**

$$h_t = o_t \odot c_t$$

$$c_t = (1 - f_t) \odot c_{t-1} + i_t \odot \texttt{tanh}\left(W_c x_t + U_c h_t + b_c\right)$$

$$\begin{pmatrix} f_t \\ i_t \\ o_t \end{pmatrix} = \texttt{sigmoid} \begin{pmatrix} W_f x_t + U_f h_{t-1} + b_f \\ W_i x_t + U_i h_{t-1} + b_i \\ W_o x_t + U_o h_{t-1} + b_o \end{pmatrix}$$

where $o_t, f_t, i_t \in [0, 1]$ represent the output gate, forget gate and input gate, respectively.

Long Short Term Memory (LSTM)

- **Update Formulation:**

$$
\begin{aligned}
h_t &= o_t \odot c_t \\
c_t &= (1 - f_t) \odot c_{t-1} + i_t \odot \mathtt{tanh}\left(W_c x_t + U_c h_t + b_c\right)
\end{aligned}
$$

$$
\begin{pmatrix} f_t \\ i_t \\ o_t \end{pmatrix} = \mathtt{sigmoid}\begin{pmatrix} W_f x_t + U_f h_{t-1} + b_f \\ W_i x_t + U_i h_{t-1} + b_i \\ W_o x_t + U_o h_{t-1} + b_o \end{pmatrix}
$$

where $o_t, f_t, i_t \in [0, 1]$ represent the output gate, forget gate and input gate, respectively.

- **Key Factors:**

Long Short Term Memory (LSTM)

- **Update Formulation:**

$$
\begin{aligned}
h_t &= o_t \odot c_t \\
c_t &= (1 - f_t) \odot c_{t-1} + i_t \odot \texttt{tanh}\left(W_c x_t + U_c h_t + b_c\right) \\
\begin{pmatrix} f_t \\ i_t \\ o_t \end{pmatrix} &= \texttt{sigmoid}\begin{pmatrix} W_f x_t + U_f h_{t-1} + b_f \\ W_i x_t + U_i h_{t-1} + b_i \\ W_o x_t + U_o h_{t-1} + b_o \end{pmatrix}
\end{aligned}
$$

  where $o_t, f_t, i_t \in [0, 1]$ represent the output gate, forget gate and input gate, respectively.

- **Key Factors:**
  - The extra state c_t is used to store long time memory.

Long Short Term Memory (LSTM)

- **Update Formulation:**

$$h_t = o_t \odot c_t$$
$$c_t = (1 - f_t) \odot c_{t-1} + i_t \odot \texttt{tanh}\left(W_c x_t + U_c h_t + b_c\right)$$
$$\begin{pmatrix} f_t \\ i_t \\ o_t \end{pmatrix} = \texttt{sigmoid}\begin{pmatrix} W_f x_t + U_f h_{t-1} + b_f \\ W_i x_t + U_i h_{t-1} + b_i \\ W_o x_t + U_o h_{t-1} + b_o \end{pmatrix}$$

  where $o_t, f_t, i_t \in [0, 1]$ represent the output gate, forget gate and input gate, respectively.

- **Key Factors:**
  - The extra state c_t is used to store long time memory.
  - Gate mechanism.

Outline

Empirical Risk Minimization:

- **Cost function:**

$$J(\theta) := \frac{1}{n}\sum_{i=1}^{n} \ell(f(x_i;\theta), y_i)$$

- **Optimizer:**

$$g_t = \frac{1}{|S_t|}\sum_{i\in S_t} \nabla_\theta \ell(f(x_i;\theta^t), y_i)$$

$$\theta_{t+1} = \theta_t + G(g_t;\eta),$$

where G can correspond to SGD, Adam, RMSProp, etc..

Back-propogation

- Let $z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell, x^\ell = \sigma(z^\ell)$, and $x^0 = x, f(x) = z^L$.

Back-propogation

- Let $z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell, x^\ell = \sigma(z^\ell)$, and $x^0 = x, f(x) = z^L$.
- Let $E = l(f(x), y)$. Then

$$\frac{\partial E}{\partial b^\ell} = \frac{\partial E}{\partial z^\ell}$$
$$\frac{\partial E}{\partial A^\ell} = \frac{\partial E}{\partial z^\ell} \cdot \left(x^{\ell-1}\right)^T$$

Back-propogation

- Let $z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell, x^\ell = \sigma(z^\ell)$, and $x^0 = x, f(x) = z^L$.
- Let $E = l(f(x), y)$. Then

$$\frac{\partial E}{\partial b^\ell} = \frac{\partial E}{\partial z^\ell}$$
$$\frac{\partial E}{\partial A^\ell} = \frac{\partial E}{\partial z^\ell} \cdot \left(x^{\ell-1}\right)^T$$

- By chain rule,

$$\begin{aligned}\frac{\partial E}{\partial z^{l-1}} &= \frac{\partial E}{\partial z^\ell} \frac{\partial z^\ell}{\partial x^{\ell-1}} \frac{\partial x^{\ell-1}}{\partial z^{\ell-1}} \\ &= \sigma'(z^{\ell-1}) \odot (A^\ell)^T \frac{\partial E}{\partial z^\ell}\end{aligned}$$

with $\frac{\partial E}{\partial z^L} = l'(f, y)$.

Backpropogation

- Let $\delta^\ell = \frac{\partial E}{\partial z^l}$ denote the gradient signal. We have

Backpropogation

- Let $\delta^\ell = \frac{\partial E}{\partial z^l}$ denote the gradient signal. We have

Forward Propagation

$$z^0 = x$$
$$z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell$$

Backpropogation

- Let $\delta^\ell = \frac{\partial E}{\partial z^l}$ denote the gradient signal. We have

Forward Propagation

$$z^0 = x$$
$$z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell$$

Back Propagation

$$\delta^L = l'(f, y)$$
$$\delta^{\ell-1} = \sigma'(z^{\ell-1}) \odot (A^\ell)^T \delta^\ell$$

Backpropogation

- Let $\delta^\ell = \frac{\partial E}{\partial z^l}$ denote the gradient signal. We have

Forward Propagation

$$z^0 = x$$
$$z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell$$

Back Propagation

$$\delta^L = l'(f, y)$$
$$\delta^{\ell-1} = \sigma'(z^{\ell-1}) \odot (A^\ell)^T \delta^\ell$$

Backpropogation

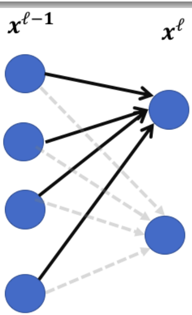
- Let $\delta^\ell = \frac{\partial E}{\partial z^l}$ denote the gradient signal. We have

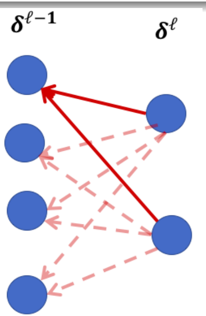
Forward Propagation

$$z^0 = x$$
$$z^\ell = A^\ell \sigma(z^{\ell-1}) + b^\ell$$

Back Propagation

$$\delta^L = l'(f, y)$$
$$\delta^{\ell-1} = \sigma'(z^{\ell-1}) \odot (A^\ell)^T \delta^\ell$$





Gradient Vanishing and Exploding

- **Gradient Vanishing:**

  $$\delta^\ell = [\sigma'(z^\ell)\odot(A^{\ell+1})^T][\sigma'(z^{l+1})\odot(A^{\ell+2})^T]\cdots[\sigma'(z^{L-1})\odot(A^L)^T\delta^L]$$

  The value is approximately the multiplication of $L-l$ term. If $\sigma'(z^\ell) < 1$ or $\|A^\ell\|_2 < 1$, then δ^ℓ will be exponentially small.

Gradient Vanishing and Exploding

- **Gradient Vanishing:**

  $$\delta^\ell = [\sigma'(z^\ell)\odot(A^{\ell+1})^T][\sigma'(z^{l+1})\odot(A^{\ell+2})^T]\cdots[\sigma'(z^{L-1})\odot(A^L)^T\delta^L]$$

  The value is approximately the multiplication of $L-l$ term. If $\sigma'(z^\ell)<1$ or $\|A^\ell\|_2<1$, then δ^ℓ will be exponentially small.

- **Depth:** $\delta^\ell \approx (\sigma'(z)\|A\|_2)^{L-\ell}$.

Gradient Vanishing and Exploding

- **Gradient Vanishing:**

  $$\delta^\ell = [\sigma'(z^\ell)\odot(A^{\ell+1})^T][\sigma'(z^{l+1})\odot(A^{\ell+2})^T]\cdots[\sigma'(z^{L-1})\odot(A^L)^T\delta^L]$$

  The value is approximately the multiplication of $L-l$ term. If $\sigma'(z^\ell) < 1$ or $\|A^\ell\|_2 < 1$, then δ^ℓ will be exponentially small.
- **Depth:** $\delta^\ell \approx (\sigma'(z)\|A\|_2)^{L-\ell}$.

Gradient Vanishing and Exploding

- **Gradient Vanishing:**

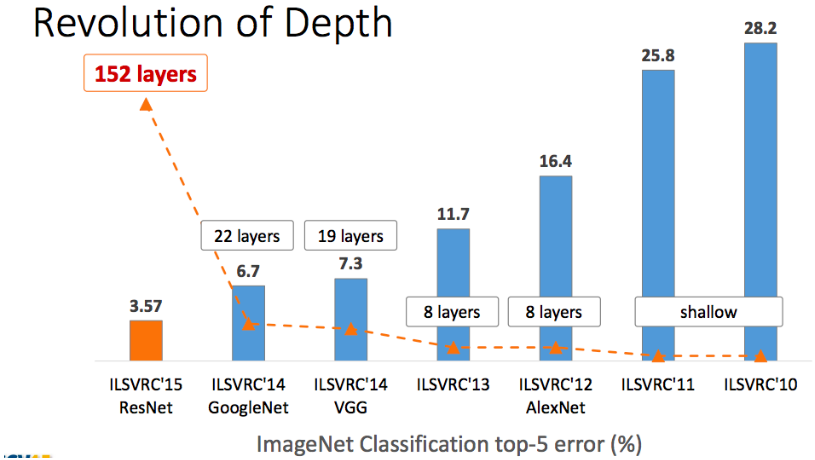
  $$\delta^\ell = [\sigma'(z^\ell)\odot(A^{\ell+1})^T][\sigma'(z^{l+1})\odot(A^{\ell+2})^T]\cdots[\sigma'(z^{L-1})\odot(A^L)^T\delta^L]$$

  The value is approximately the multiplication of $L-l$ term. If $\sigma'(z^\ell)<1$ or $\|A^\ell\|_2<1$, then δ^ℓ will be exponentially small.
- **Depth:** $\delta^\ell \approx (\sigma'(z)\|A\|_2)^{L-\ell}$.

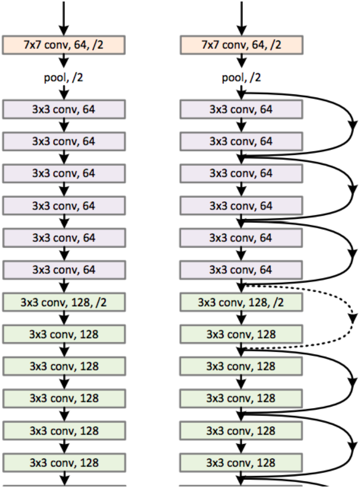
Observation

The vanishing gradient is the key difficulty to training deep neural networks.

Outline


Revolution of Depth
152 layers
22 layers
19 layers
8 layers
8 layers
shallow
3.57
6.7
7.3
11.7
16.4
25.8
28.2
ILSVRC'15
ResNet
ILSVRC'14
GoogleNet
ILSVRC'14
VGG
ILSVRC'13
ILSVRC'12
AlexNet
ILSVRC'11
ILSVRC'10
ImageNet Classification top-5 error (%)

ResNet: Basic Structure



Vanilla net

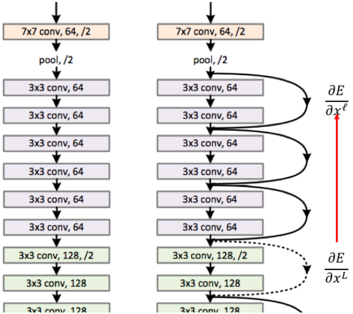
$$x^{n+1} = F(x^n; \theta)$$

Residual net

$$x^{n+1} = F(x^n; \theta) + x^n$$

ResNet: The Importance of Identity Connection

- $x^{\ell+1} = x^\ell + F(x^\ell)$
- $x^L = x^\ell + \sum_{i=\ell}^{L-1} F(x^i)$
- $\frac{\partial E}{\partial x^\ell} = \frac{\partial E}{\partial x^L}\left(1 + \frac{\partial}{\partial x^\ell}\sum_{i=\ell}^{L-1} F(x^i)\right)$



Observation

The skip connection can alleviate the vanishing of gradient.