Deep Learning
Deep learning attracts lots of attention.

• I believe you have seen lots of exciting results before.

Deep learning trends at Google. Source: SIGMOD/Jeff Dean
Ups and downs of Deep Learning

- **1958**: Perceptron (linear model)
- **1969**: Perceptron has limitation
- **1980s**: Multi-layer perceptron
  - Do not have significant difference from DNN today
- **1986**: Backpropagation
  - Usually more than 3 hidden layers is not helpful
- **1989**: 1 hidden layer is “good enough”, why deep?
- **2006**: RBM initialization (breakthrough)
- **2009**: GPU
- **2011**: Start to be popular in speech recognition
- **2012**: win ILSVRC image competition
Three Steps for Deep Learning

Step 1: Neural Network

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ......
Neural Network

Different connection leads to different network structures

Network parameter $\theta$: all the weights and biases in the “neurons”
Fully Connect Feedforward Network

Sigmoid Function

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$
Fully Connect Feedforward Network
This is a function.
Input vector, output vector

\[ f \left( \begin{bmatrix} 1 \\ -1 \end{bmatrix} \right) = \begin{bmatrix} 0.62 \\ 0.83 \end{bmatrix} \quad f \left( \begin{bmatrix} 0 \end{bmatrix} \right) = \begin{bmatrix} 0.51 \\ 0.85 \end{bmatrix} \]

Given network structure, define a function set
Fully Connect Feedforward Network

Input Layer

Layer 1

Layer 2

Layer L

Output Layer

$\mathbf{x}_1$

$\mathbf{x}_2$

$\mathbf{x}_N$

Input

Hidden Layers

Output

Layer 1

Layer 2

Layer L

$\mathbf{y}_1$

$\mathbf{y}_2$

$\mathbf{y}_M$

neuron
Deep = Many hidden layers


AlexNet (2012) 8 layers 16.4%
VGG (2014) 19 layers 7.3%
GoogleNet (2014) 22 layers 6.7%
Deep = Many hidden layers

AlexNet (2012) 152 layers 3.57%
VGG (2014) 101 layers 16.4%
GoogleNet (2014) 101 layers 7.3%
Residual Net (2015) 152 layers 6.7%
Taipei 101
Matrix Operation

\[
\begin{bmatrix}
1 & -2 \\
-1 & 1
\end{bmatrix}
\begin{bmatrix}
1 \\
-1
\end{bmatrix} +
\begin{bmatrix}
1 \\
0
\end{bmatrix} =
\begin{bmatrix}
0.98 \\
0.12
\end{bmatrix}
\]
Neural Network

\[ \sigma \left( W^1 x + b^1 \right) \]

\[ \sigma \left( W^2 a^1 + b^2 \right) \]

\[ \sigma \left( W^L a^{L-1} + b^L \right) \]
Neural Network

\[ y = f(x) \]

\[ = \sigma(W_L) \cdots \sigma(W^2) \sigma(W^1)x + b^1 + b^2 + \cdots + b^L \]

Using parallel computing techniques to speed up matrix operation.
Output Layer

Feature extractor replacing feature engineering

Input Layer −→ Hidden Layers −→ Output Layer

= Multi-class Classifier

\( x \) \( \cdots \) \( x_k \) \( \cdots \) \( x_{K} \) −→ \( y_1 \) \( \cdots \) \( y_M \)
Example Application

Input

\[16 \times 16 = 256\]
Ink → 1
No ink → 0

Output

\[\begin{align*}
x_1 & = 0.1 \quad \text{is 1} \\
x_2 & = 0.7 \quad \text{is 2} \\
\vdots & \quad \vdots \\
x_{256} & = 0.2 \quad \text{is 0}
\end{align*}\]

The image is “2”

Each dimension represents the confidence of a digit.
Example Application

- Handwriting Digit Recognition

Input: 256-dim vector

What is needed is a function ......

Neural Network

Output: 10-dim vector

$y_1$ is 1
$y_2$ is 2
$y_{10}$ is 0
Example Application

A function set containing the candidates for Handwriting Digit Recognition

You need to decide the network structure to let a good function in your function set.
FAQ

• Q: How many layers? How many neurons for each layer?

  Trial and Error + Intuition

• Q: Can the structure be automatically determined?
  • E.g. Evolutionary Artificial Neural Networks

• Q: Can we design the network structure?

Convolutional Neural Network (CNN)
Three Steps for Deep Learning

Step 1: Neural Network

Step 2: Goodness of function

Step 3: Pick the best function

Deep Learning is so simple ......
Loss for an Example

Given a set of parameters

$$C(y, \hat{y}) = - \sum_{i=1}^{10} \hat{y}_i \ln y_i$$
Total Loss

For all training data ...

Total Loss:

\[ L = \sum_{n=1}^{N} C^n \]

Find a function in function set that minimizes total loss \( L \)

Find the network parameters \( \theta^* \) that minimize total loss \( L \)
Three Steps for Deep Learning

Step 1: Define a set of function

Step 2: Goodness of function

Step 3: Pick the best function

Deep Learning is so simple ......
Gradient Descent

\[ \nabla L = \begin{bmatrix} \frac{\partial L}{\partial w_1} \\ \frac{\partial L}{\partial w_2} \\ \vdots \\ \frac{\partial L}{\partial b_1} \end{bmatrix} \]

\[ \text{gradient} \]
Gradient Descent

\[ \nabla \theta \]

\[ \begin{align*}  
  \theta & \quad \text{Compute } \frac{\partial L}{\partial w_1} \\
  w_1 & \quad -\mu \frac{\partial L}{\partial w_1} \quad \rightarrow \quad 0.15 \\
  w_2 & \quad -\mu \frac{\partial L}{\partial w_2} \quad \rightarrow \quad 0.05 \\
  b_1 & \quad -\mu \frac{\partial L}{\partial b_1} \quad \rightarrow \quad 0.2 \\
  & \quad \vdots
\end{align*} \]
Gradient Descent

This is the “learning” of machines in deep learning ......

Even alpha go using this approach.

People image ...... Actually ..... 

I hope you are not too disappointed :p
Backpropagation

- Backpropagation: an efficient way to compute $\frac{\partial L}{\partial w}$ in neural network

Ref:
Step 1: define a set of function
Step 2: goodness of function
Step 3: pick the best function

What are the benefits of deep architecture?
## Deeper is Better?

<table>
<thead>
<tr>
<th>Layer X Size</th>
<th>Word Error Rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 X 2k</td>
<td>24.2</td>
</tr>
<tr>
<td>2 X 2k</td>
<td>20.4</td>
</tr>
<tr>
<td>3 X 2k</td>
<td>18.4</td>
</tr>
<tr>
<td>4 X 2k</td>
<td>17.8</td>
</tr>
<tr>
<td>5 X 2k</td>
<td>17.2</td>
</tr>
<tr>
<td>7 X 2k</td>
<td>17.1</td>
</tr>
</tbody>
</table>

Universality Theorem

Any continuous function $f$

$$f : \mathbb{R}^N \rightarrow \mathbb{R}^M$$

Can be realized by a network with one hidden layer

(given enough hidden neurons)


Why “Deep” neural network not “Fat” neural network?

(next lecture)
“深度學習”

- **My Course: Machine learning and having it deep and structured**
  - [http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html](http://speech.ee.ntu.edu.tw/~tlkagk/courses_MLSD15_2.html)
  - [6 hour version: http://www.slideshare.net/tw_dsconf/ss62245351](http://www.slideshare.net/tw_dsconf/ss62245351)

- **“Neural Networks and Deep Learning”**
  - written by Michael Nielsen

- **“Deep Learning”**
  - written by Yoshua Bengio, Ian J. Goodfellow and Aaron Courville
  - [http://www.deeplearningbook.org](http://www.deeplearningbook.org)
Acknowledgment

• 感謝 Victor Chen 發現投影片上的打字錯誤