Deep Learning Tutorial

李宏毅

Hung-yi Lee
Deep learning attracts lots of attention.

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

This talk focuses on the basic techniques.
Outline

- Lecture I: Introduction of Deep Learning
- Lecture II: Variants of Neural Network
- Lecture III: Beyond Supervised Learning
Lecture I: Introduction of Deep Learning
Outline

Introduction of Deep Learning

“Hello World” for Deep Learning

Tips for Deep Learning
Machine Learning
≈ Looking for a Function

• Speech Recognition
  \[ f(\text{(what the user said)}) = \text{“How are you”} \]

• Image Recognition
  \[ f(\text{(an image)}) = \text{“Cat”} \]

• Playing Go
  \[ f(\text{(next move)}) = \text{“5-5”} \]

• Dialogue System
  \[ f(\text{“Hi”}) = \text{“Hello”} \]
  (what the user said)  (system response)
Framework

A set of function

Model

$$f_1(\text{“cat”}) = \text{“cat”}$$

$$f_2(\text{“dog”}) = \text{“dog”}$$

$$f_2(\text{“money”}) = \text{“money”}$$

$$f_2(\text{“snake”}) = \text{“snake”}$$
Framework

A set of function $f_1, f_2 \ldots$

Goodness of function $f$

Training Data

Model

Supervised Learning

Image Recognition:

$f(\text{cat}) = \text{“cat”}$

Better!

$f_1(\text{cat}) = \text{“cat”} \quad f_2(\text{money}) = \text{“money”}$

$f_1(\text{dog}) = \text{“dog”} \quad f_2(\text{snake}) = \text{“snake”}$

function input: “monkey” “cat” “dog”

function output: “monkey” “cat” “dog”
Image Recognition:

$$f(\text{cat}) = \text{“cat”}$$

**Framework**

- **Step 1**: A set of functions $$f_1, f_2, \ldots$$
- **Step 2**: Goodness of function $$f$$
- **Step 3**: Pick the “Best” Function $$f^*$$

**Model**

- Training Data
  - “monkey”
  - “cat”
  - “dog”

**Training**

**Testing**

Using $$f^*$$
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function
Neural Network

Neuron

\[ z = a_1 w_1 + \cdots + a_k w_k + \cdots + a_K w_K + b \]
Neural Network

**Neuron**

The diagram illustrates a single neuron with weighted inputs and a bias. The inputs are multiplied by their respective weights and summed, then passed through a Sigmoid function to produce an output.

**Sigmoid Function**

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]

The neuron's output is 0.98 after applying the Sigmoid function to the weighted sum of inputs and bias.
Neural Network

Different connections lead to different network structures.

The neurons have different values of weights and biases.

Weights and biases are network parameters $\theta$. 
Sigmoid Function

\[
\sigma(z) = \frac{1}{1 + e^{-z}}
\]
Fully Connect Feedforward Network

-1 -> 1
-1 -> -2
1 -> 1
1 -> -1

1 -> 0.98 -> 2
1 -> -1 -> 0.12 -> -2
2 -> 0.86 -> 3
3 -> -1 -> 0.11 -> -1
4 -> -2 -> 0.62

0 -> 0.83 -> 2
Given parameters $\theta$, define a function

$\mathbf{f}([1 -1]) = [0.62 \ 0.83] \quad \mathbf{f}([0]) = [0.51 \ 0.85]$
Fully Connect Feedforward Network

- **Input Layer**: $x_1, x_2, \ldots, x_N$
- **Hidden Layers**: Layer 1, Layer 2, …, Layer L
- **Output Layer**: $y_1, y_2, \ldots, y_M$

**Deep means many hidden layers**
Why Deep? 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?

Reference for the reason:
Why Deep? Analogy

Logic circuits

• Logic circuits consists of gates

• A two layers of logic gates can represent any Boolean function.

• Using multiple layers of logic gates to build some functions are much simpler

Neural network

• Neural network consists of neurons

• A hidden layer network can represent any continuous function.

• Using multiple layers of neurons to represent some functions are much simpler

less gates needed

less parameters

less data?

More reason:
https://www.youtube.com/watch?v=XsC9byQkUH8&list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49&index=13
Deep = Many hidden layers


AlexNet (2012)

VGG (2014)

GoogleNet (2014)

8 layers

19 layers

22 layers

16.4%

7.3%

6.7%
Deep = Many hidden layers

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

• Softmax layer as the output layer

\[
\begin{align*}
\text{Ordinary Layer} & \\
\sigma & \\
\end{align*}
\]

\[
\begin{align*}
z_1 & \quad \sigma \quad y_1 = \sigma(z_1) \\
z_2 & \quad \sigma \quad y_2 = \sigma(z_2) \\
z_3 & \quad \sigma \quad y_3 = \sigma(z_3)
\end{align*}
\]

In general, the output of network can be any value. May not be easy to interpret.
Output Layer

- Softmax layer as the output layer

**Softmax Layer**

$$y_i = \frac{e^{z_i}}{\sum_{j=1}^{3} e^{z_j}}$$

**Probability:**
- $1 > y_i > 0$
- $\sum_i y_i = 1$
Example Application

Input

16 x 16 = 256
Ink → 1
No ink → 0

Output

The image is “2”
Each dimension represents the confidence of a digit.
Example Application

• Handwriting Digit Recognition

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

Input: 256-dim vector

output: 10-dim vector
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 we design the network structure?
  Convolutional Neural Network (CNN) in the next lecture

• Q: Can the structure be automatically determined?
  • Yes, but not widely studied yet.
Highway Network

- Residual Network

Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385

- Highway Network

Gate controller

Training Very Deep Networks
Highway Network automatically determines the layers needed!
Three Steps for Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function
Training Data

• Preparing training data: images and their labels

```
5 “5” 0 “0” 4 “4” 1 “1”
9 “9” 2 “2” 1 “1” 3 “3”
```

The learning target is defined on the training data.
The learning target is ……

Input: 

- $y_1$ has the maximum value
- $y_2$ has the maximum value

16 x 16 = 256
Ink → 1
No ink → 0

$y_1$ is 1
$y_2$ is 2
$y_{10}$ is 0
Loss

A good function should make the loss of all examples as small as possible.

Loss can be **square error** or **cross entropy** between the network output and target.
Total Loss

For all training data ...

\[ L = \sum_{r=1}^{R} l_r \]

As small as possible

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
How to pick the best function

Find **network parameters** $\theta^*$ that minimize total loss $L$

Enumerate all possible values

Network parameters $\theta = \{w_1, w_2, w_3, \ldots, b_1, b_2, b_3, \ldots\}$

Millions of parameters

E.g. speech recognition: 8 layers and 1000 neurons each layer
Gradient Descent

Network parameters $\theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

Find *network parameters* $\theta^*$ that minimize total loss $L$

- Pick an initial value for $w$
  - Random, RBM pre-train
  - Usually good enough
Gradient Descent

Network parameters $\theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

Find network parameters $\theta^*$ that minimize total loss $L$

- Pick an initial value for $w$
- Compute $\partial L / \partial w$

Decrease $w$ if $\partial L / \partial w$ is negative
Increase $w$ if $\partial L / \partial w$ is positive

http://chico386.pixnet.net/album/photo/171572850
Gradient Descent

Network parameters $\theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots\}$

Find **network parameters $\theta^*$** that minimize total loss $L$

- Pick an initial value for $w$
- Compute $\partial L / \partial w$
  
  $w \leftarrow w - \eta \partial L / \partial w$

$\eta$ is called "learning rate"
Gradient Descent

Network parameters \( \theta = \{w_1, w_2, \ldots, b_1, b_2, \ldots \} \)

Find \emph{network parameters} \( \theta^* \) that minimize total loss \( L \)

- Pick an initial value for \( w \)
- Compute \( \frac{\partial L}{\partial w} \)
  \[ w \leftarrow w - \eta \frac{\partial L}{\partial w} \]

Repeat Until \( \frac{\partial L}{\partial w} \) is approximately small (when update is little)
Gradient Descent

Randomly pick a starting point

Color: Value of Total Loss L
Gradient Descent

Hopfully, we would reach a minima .....
Local Minima

The value of a network parameter $w$ is very slow at the plateau, stuck at saddle point, and stuck at local minima.

$\frac{\partial L}{\partial w} \approx 0$
$\frac{\partial L}{\partial w} = 0$
$\frac{\partial L}{\partial w} = 0$
Local Minima

- Gradient descent never guarantee global minima
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: https://www.youtube.com/watch?v=ibJpTrp5mcE
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 ......

Now If you want to find a function
If you have lots of function input/output (?) as training data
You can use deep learning
For example, you can do ……

• Image Recognition
For example, you can do ......

**Spam filtering**

“Talk” in e-mail

“free” in e-mail

(Network)

1/0 (Yes/No)

1 (Yes)

0 (No)

(http://spam-filter-review.toptenreviews.com/)
For example, you can do .......

“stock” in document

“president” in document

http://top-breaking-news.com/
Outline

Introduction of Deep Learning

“Hello World” for Deep Learning

Tips for Deep Learning
Keras

If you want to learn Theano:

Very flexible
Need some effort to learn

TensorFlow or Theano

Easy to learn and use
(still have some flexibility)
You can modify it if you can write TensorFlow or Theano
Keras

• François Chollet is the author of Keras.
  • He currently works for Google as a deep learning engineer and researcher.

• Keras means *horn* in Greek

• Documentation: [http://keras.io/](http://keras.io/)

• Example: 
  [https://github.com/fchollet/keras/tree/master/examples](https://github.com/fchollet/keras/tree/master/examples)
使用 Keras 心得
Example Application

- Handwriting Digit Recognition

MNIST Data: http://yann.lecun.com/exdb/mnist/
“Hello world” for deep learning

Keras provides data sets loading function: http://keras.io/datasets/
Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

```
model = Sequential()
model.add(Dense(input_dim=28*28, output_dim=500))
model.add(Activation('sigmoid'))
model.add(Dense(output_dim=500))
model.add(Activation('sigmoid'))
model.add(Dense(output_dim=10))
model.add(Activation('softmax'))
```
Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=[{'accuracy']})
Keras

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Step 3.1: Configuration

```python
model.compile(loss='mse', optimizer=SGD(lr=0.1), metrics=[['accuracy']])
```

\[ w \leftarrow w - \eta \frac{\partial L}{\partial w} \]

0.1

Step 3.2: Find the optimal network parameters

```python
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```
**Keras**

**Step 1:** Define a set of function

**Step 2:** Goodness of function

**Step 3:** Pick the best function

**Step 3.2: Find the optimal network parameters**

```python
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

Number of training examples: 28 x 28 = 784

Number of training examples: 10
Keras

Save and load models
http://keras.io/getting-started/faq/#how-can-i-save-a-keras-model

How to use the neural network (testing):

```python
score = model.evaluate(x_test, y_test)
print('Total loss on Testing Set:', score[0])
print('Accuracy of Testing Set:', score[1])

case 2: result = model.predict(x_test)
```
Keras

• Using GPU to speed training
  • Way 1
    • THEANO_FLAGS=device=gpu0 python YourCode.py
  • Way 2 (in your code)
    • import os
    • os.environ["THEANO_FLAGS"] = "device=gpu0"
Demo
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 ......
Recipe of Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Good Results on Training Data?

Good Results on Testing Data?

Overfitting!
Do not always blame Overfitting

Deep Residual Learning for Image Recognition
http://arxiv.org/abs/1512.03385
Recipe of Deep Learning

Different approaches for different problems.

e.g. dropout for good results on testing data

Good Results on Testing Data?

Good Results on Training Data?

Neural Network
Recipe of Deep Learning

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

Good Results on Training Data?

Good Results on Testing Data?

YES

YES
Choosing Proper Loss

Which one is better?

Square Error: \[
\sum_{i=1}^{10} (y_i - \hat{y}_i)^2 = 0
\]

Cross Entropy: \[
-\sum_{i=1}^{10} \hat{y}_i \ln y_i = 0
\]
Demo

Square Error

```python
model.compile(loss='mse',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Cross Entropy

```python
model.compile(loss='categorical_crossentropy',
              optimizer=SGD(lr=0.1),
              metrics=['accuracy'])
```

Several alternatives: https://keras.io/objectives/
Demo
Choosing Proper Loss

When using softmax output layer, choose cross entropy

http://jmlr.org/proceedings/papers/v9/glorot10a/glorot10a.pdf
Recipe of Deep Learning

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

Good Results on Training Data?

Good Results on Testing Data?

model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
Mini-batch

- Randomly initialize network parameters
- Pick the 1\textsuperscript{st} batch
  \[ L' = l^1 + l^{31} + \ldots \]
  Update parameters once
- Pick the 2\textsuperscript{nd} batch
  \[ L'' = l^2 + l^{16} + \ldots \]
  Update parameters once
- Until all mini-batches have been picked

We do not really minimize total loss!
Mini-batch

```python
model.fit(x_train, y_train, batch_size=100, nb_epoch=20)
```

- Pick the 1\textsuperscript{st} batch
  
  \[ L' = l^1 + l^{31} + \cdots \]

  Update parameters once

- Pick the 2\textsuperscript{nd} batch
  
  \[ L'' = l^2 + l^{16} + \cdots \]

  Update parameters once

- Until all mini-batches have been picked

100 examples in a mini-batch

Repeat 20 times

one epoch
Mini-batch

**Original Gradient Descent**

**With Mini-batch**

The colors represent the total loss.
Mini-batch is Faster

**Original Gradient Descent**
Update after seeing all examples

**With Mini-batch**
If there are 20 batches, update 20 times in one epoch.

Not always true with parallel computing.

- Original Gradient Descent: This method updates the parameters after seeing all examples in an epoch. However, if the dataset is large, this can be computationally expensive.

- With Mini-batch: This method updates the parameters after seeing a small fraction of the dataset (a mini-batch) in each iteration. This can be faster, especially with large datasets, but not always true with parallel computing.

Can have the same speed (not super large data set)

Mini-batch has better performance!
Demo
Shuffle the training examples for each epoch

Epoch 1

Mini-batch

\( x^1 \rightarrow \text{NN} \rightarrow y^1 \rightarrow \hat{y}^1 \)

\( x^{31} \rightarrow \text{NN} \rightarrow y^{31} \rightarrow \hat{y}^{31} \)

\( \cdots \)

Mini-batch

\( x^2 \rightarrow \text{NN} \rightarrow y^2 \rightarrow \hat{y}^2 \)

\( x^{16} \rightarrow \text{NN} \rightarrow y^{16} \rightarrow \hat{y}^{16} \)

Mini-batch

\( x^{17} \rightarrow \text{NN} \rightarrow y^{17} \rightarrow \hat{y}^{17} \)

\( \cdots \)

Mini-batch

\( x^{26} \rightarrow \text{NN} \rightarrow y^{26} \rightarrow \hat{y}^{26} \)

Don’t worry. This is the default of Keras.

Epoch 2

Mini-batch

\( x^1 \rightarrow \text{NN} \rightarrow y^1 \rightarrow \hat{y}^1 \)

\( x^{17} \rightarrow \text{NN} \rightarrow y^{17} \rightarrow \hat{y}^{17} \)

\( \cdots \)

Mini-batch

\( x^2 \rightarrow \text{NN} \rightarrow y^2 \rightarrow \hat{y}^2 \)

\( x^{26} \rightarrow \text{NN} \rightarrow y^{26} \rightarrow \hat{y}^{26} \)

\( \cdots \)
Recipe of Deep Learning

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

Good Results on Training Data?

Good Results on Testing Data?
Hard to get the power of Deep ... 

Results on Training Data

Deeper usually does not imply better.
Demo
Vanishing Gradient Problem

- Smaller gradients
  - Learn very slow
  - Almost random

- Larger gradients
  - Learn very fast
  - Already converge

*Based on random!?
Vanishing Gradient Problem

Smaller gradients

Intuitive way to compute the derivatives ...

$$\frac{\partial l}{\partial w} = ? \frac{\Delta l}{\Delta w}$$
Hard to get the power of Deep ...

In 2006, people used RBM pre-training.
In 2015, people use ReLU.
ReLU

- Rectified Linear Unit (ReLU)

**Reason:**

1. Fast to compute
2. Biological reason
3. Infinite sigmoid with different biases
4. Vanishing gradient problem

\[ a = \begin{cases} 
0 & \text{if } z < 0 \\
z & \text{if } z \geq 0 
\end{cases} \]

[Andrew L. Maas, ICML’13]
[Xavier Glorot, AISTATS’11]
[Kaiming He, arXiv’15]
ReLU

\[ a = 0 \]

\[ a = z \]

\[ x_1, x_2 \rightarrow 0 \]

\[ y_1, y_2 \rightarrow 0 \]
ReLU

A Thinner linear network

Do not have smaller gradients
Demo
ReLU - variant

Leaky ReLU

$$a = 0.01z$$

$$a = z$$

Parametric ReLU

$$a = az$$

$$a = z$$

$\alpha$ also learned by gradient descent
Maxout

• Learnable activation function [Ian J. Goodfellow, ICML’13]

ReLU is a special case of Maxout

You can have more than 2 elements in a group.
Maxout

- Learnable activation function [Ian J. Goodfellow, ICML’13]
  - Activation function in maxout network can be any piecewise linear convex function
  - How many pieces depending on how many elements in a group

ReLU is a special cases of Maxout

2 elements in a group

3 elements in a group
Recipe of Deep Learning

Choosing proper loss

Mini-batch

New activation function

Adaptive Learning Rate

Momentum

Good Results on Training Data?

YES

Good Results on Testing Data?

YES
Learning Rates

**Set the learning rate $\eta$ carefully**

If learning rate is too large

Total loss may not decrease after each update
Learning Rates

- If learning rate is too large, set the learning rate $\eta$ carefully.
- If learning rate is too small, training would be too slow.
- Total loss may not decrease after each update.
Learning Rates

• Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  • At the beginning, we are far from the destination, so we use larger learning rate
  • After several epochs, we are close to the destination, so we reduce the learning rate
  • E.g. 1/t decay: $\eta^t = \eta / \sqrt{t} + 1$

• Learning rate cannot be one-size-fits-all
  • Giving different parameters different learning rates
Adagrad

Original: \[ w \leftarrow w - \eta \frac{\partial L}{\partial w} \]

Adagrad: \[ w \leftarrow w - \eta_w \frac{\partial L}{\partial w} \]

Parameter dependent learning rate

\[ \eta_w = \frac{\eta}{\sqrt{\sum_{i=0}^{t} (g_i)^2}} \]

\( g^i \) is \( \frac{\partial L}{\partial w} \) obtained at the i-th update

Summation of the square of the previous derivatives
Adagrad

Observation:
1. Learning rate is smaller and smaller for all parameters
2. Smaller derivatives, larger learning rate, and vice versa
2. Smaller derivatives, larger learning rate, and vice versa
Not the whole story ......

- **Adagrad** [John Duchi, JMLR’11]
- **RMSprop**
  - [https://www.youtube.com/watch?v=O3sxAc4hxZU](https://www.youtube.com/watch?v=O3sxAc4hxZU)
- **Adadelta** [Matthew D. Zeiler, arXiv’12]
- “No more pesky learning rates” [Tom Schaul, arXiv’12]
- **AdaSecant** [Caglar Gulcehre, arXiv’14]
- **Adam** [Diederik P. Kingma, ICLR’15]
- **Nadam**
**Recipe of Deep Learning**

- Choosing proper loss
- Mini-batch
- New activation function
- Adaptive Learning Rate
- Momentum

Good Results on Training Data?

Good Results on Testing Data?

YES
Hard to find optimal network parameters

Total Loss

Very slow at the plateau

\[ \frac{\partial L}{\partial w} \approx 0 \]

Stuck at saddle point

\[ \frac{\partial L}{\partial w} = 0 \]

Stuck at local minima

The value of a network parameter w
In physical world ......

- Momentum

How about put this phenomenon in gradient descent?
Movement = Negative of $\frac{\partial L}{\partial w}$ + Momentum

Still not guarantee reaching global minima, but give some hope ......

Movement = Negative of $\frac{\partial L}{\partial w}$ + Momentum

- Red: Negative of $\frac{\partial L}{\partial w}$
- Green: Momentum
- Blue: Real Movement

$\frac{\partial L}{\partial w} = 0$
Algorithm 1: Adam, our proposed algorithm for stochastic optimization. See section 2 for details, and for a slightly more efficient (but less clear) order of computation. $g_t^2$ indicates the elementwise square $g_t \odot g_t$. Good default settings for the tested machine learning problems are $\alpha = 0.001$, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. All operations on vectors are element-wise. With $\beta_1^t$ and $\beta_2^t$ we denote $\beta_1$ and $\beta_2$ to the power $t$.

Require: $\alpha$: Stepsize
Require: $\beta_1, \beta_2 \in [0, 1)$: Exponential decay rates for the moment estimates
Require: $f(\theta)$: Stochastic objective function with parameters $\theta$
Require: $\theta_0$: Initial parameter vector

$m_0 \leftarrow 0$ (Initialize 1st moment vector)
$v_0 \leftarrow 0$ (Initialize 2nd moment vector)
t $\leftarrow 0$ (Initialize timestep)

while $\theta_t$ not converged do
    $t \leftarrow t + 1$
    $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$ (Get gradients w.r.t. stochastic objective at timestep $t$)
    $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$ (Update biased first moment estimate)
    $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$ (Update biased second raw moment estimate)
    $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$ (Compute bias-corrected first moment estimate)
    $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$ (Compute bias-corrected second raw moment estimate)
    $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$ (Update parameters)
end while

return $\theta_t$ (Resulting parameters)
Demo
Recipe of Deep Learning

- Early Stopping
- Regularization
- Dropout
- Network Structure

Good Results on Testing Data?

YES

Good Results on Training Data?

YES

YES
Panacea for Overfitting

• Have more training data
• *Create* more training data (?)

Handwriting recognition:

Original Training Data: \( \begin{array}{c}
\text{Original} \\
\text{Training Data:} \\
\end{array} \) \\
Created Training Data: \( \begin{array}{c}
\text{Created} \\
\text{Training Data:} \\
\end{array} \)

Shift 15°
Recipe of Deep Learning

- Early Stopping
- Regularization
- Dropout
- Network Structure

Good Results on Training Data?

Good Results on Testing Data?

YES

YES

Recipe of Deep Learning

- Early Stopping
- Regularization
- Dropout
- Network Structure
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

For each mini-batch, we resample the dropout neurons.
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.
Dropout - Intuitive Reason

**Training**
Dropout (腳上綁重物)

**Testing**
No dropout
(拿下重物後就變很強)
Dropout - Intuitive Reason

- Why the weights should multiply (1-p)% (dropout rate) when testing?

**Training of Dropout**
Assume dropout rate is 50%

**Testing of Dropout**
No dropout

Weights from training

\[ z' \approx 2z \]

Weights multiply 1-p%

\[ z' \approx z \]
Dropout is a kind of ensemble.

**Ensemble**

Train a bunch of networks with different structures
Dropout is a kind of ensemble.

**Ensemble**

Testing data $x$

Network 1

Network 2

Network 3

Network 4

$y_1$

$y_2$

$y_3$

$y_4$

average
Dropout is a kind of ensemble.

- Using one mini-batch to train one network
- Some parameters in the network are shared

Training of Dropout:

- M neurons
- $2^M$ possible networks
Dropout is a kind of ensemble.

**Testing of Dropout**

All the weights multiply $1-p\%$

$$y \approx \text{average of } y_1, y_2, y_3$$
More about dropout


• Dropout works better with Maxout [Ian J. Goodfellow, ICML’13]

• Dropconnect [Li Wan, ICML’13]
  • Dropout delete neurons
  • Dropconnect deletes the connection between neurons

• Annealed dropout [S.J. Rennie, SLT’14]
  • Dropout rate decreases by epochs

• Standout [J. Ba, NISP’13]
  • Each neural has different dropout rate
Demo

```python
model = Sequential()
model.add( Dense( input_dim=28*28, output_dim=500 ) )
model.add( Activation('sigmoid') )
model.add( dropout(0.8) )
model.add( Dense(output_dim=10) )
model.add( Activation('softmax') )
```
Demo
Recipe of Deep Learning

- Early Stopping
- Regularization
- Dropout
- Network Structure

CNN is a very good example! (next lecture)
Concluding Remarks
Recipe of Deep Learning

Step 1: define a set of function

Step 2: goodness of function

Step 3: pick the best function

Good Results on Testing Data?

Good Results on Training Data?

YES

NO
Lecture II:
Variants of Neural Networks
Variants of Neural Networks

Convolutional Neural Network (CNN)
Widely used in image processing

Recurrent Neural Network (RNN)
Why CNN for Image?

Can the network be simplified by considering the properties of images?

The most basic classifiers

Use 1st layer as module to build classifiers

Use 2nd layer as module ......

Represented as pixels

[Zeiler, M. D., ECCV 2014]
Why CNN for Image

• Some patterns are much smaller than the whole image

A neuron does not have to see the whole image to discover the pattern.

Connecting to small region with less parameters

“beak” detector
Why CNN for Image

• The same patterns appear in different regions.

Do almost the same thing
They can use the same set of parameters.
Why CNN for Image

- Subsampling the pixels will not change the object

We can subsample the pixels to make image smaller

Less parameters for the network to process the image
Three Steps for Deep Learning

1. Step 1: Convolutional Neural Network
2. Step 2: Goodness of function
3. Step 3: Pick the best function

Deep Learning is so simple ......
The whole CNN

[Diagram of CNN architecture]

- Fully Connected Feedforward network
- Convolution
- Max Pooling
- Convolution
- Max Pooling
- Flatten

Can repeat many times

cat dog .......
The whole CNN

Property 1
➢ Some patterns are much smaller than the whole image

Property 2
➢ The same patterns appear in different regions.

Property 3
➢ Subsampling the pixels will not change the object

Can repeat many times
The whole CNN

- Convolution
- Max Pooling
- Convolution
- Max Pooling
- Flatten

Can repeat many times
CNN – Convolution

6 x 6 image

Each filter detects a small pattern (3 x 3).

Those are the network parameters to be learned.

Property 1
CNN – Convolution

Filter 1

\[
\begin{pmatrix}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1 \\
\end{pmatrix}
\]

stride=1

6 x 6 image
CNN – Convolution

If stride=2

6 x 6 image

Filter 1

We set stride=1 below
CNN – Convolution

**Filter 1**

\[
\begin{array}{ccc}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1 \\
\end{array}
\]

**6 x 6 image**

\[
\begin{array}{cccccccc}
1 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 1 & 0 & 0 & 1 & 0 & \ \\
0 & 0 & 1 & 1 & 0 & 0 & \ \\
1 & 0 & 0 & 0 & 1 & 0 & \ \\
0 & 1 & 0 & 0 & 1 & 0 & \ \\
0 & 0 & 1 & 0 & 1 & 0 & \ \\
0 & 0 & 1 & 0 & 1 & 0 & \ \\
\end{array}
\]

**stride=1**

**Property 2**
CNN – Convolution

Do the same process for every filter

6 x 6 image

4 x 4 image

Feature Map
CNN – Zero Padding

You will get another 6 x 6 images in this way

Zero padding
CNN – Colorful image

Colorful image

Filter 1

Filter 2

Filter 1

Filter 2

Colorful image
**Convolution v.s. Fully Connected**

**Image**

```
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
```

**Convolution**

```
-1 -1 -1
-1 1 -1
-1 1 -1
```

**Fully-connected**

```
1 0 0 0 0 1
0 1 0 0 1 0
0 0 1 1 0 0
1 0 0 0 1 0
0 1 0 0 1 0
0 0 1 0 1 0
```

```
x_1
x_2
...
x_36
```
Filter 1

6 x 6 image

Less parameters!

Only connect to 9 input, not fully connected

Less parameters!
Filter 1

6 x 6 image

Less parameters!
Even less parameters!

Shared weights
The whole CNN

cat dog ......

Fully Connected Feedforward network

Flatten

Convolution

Max Pooling

Convolution

Max Pooling

Can repeat many times
CNN – Max Pooling

Filter 1:

\[
\begin{array}{ccc}
1 & -1 & -1 \\
-1 & 1 & -1 \\
-1 & -1 & 1 \\
\end{array}
\]

Filter 2:

\[
\begin{array}{ccc}
-1 & 1 & -1 \\
-1 & 1 & -1 \\
-1 & 1 & -1 \\
\end{array}
\]

Input:

\[
\begin{array}{cccc}
3 & -1 & -3 & -1 \\
-3 & 1 & 0 & -3 \\
-3 & -3 & 0 & 1 \\
3 & -2 & -2 & -1 \\
\end{array}
\]

Output:

\[
\begin{array}{cccc}
-1 & -1 & -1 & -1 \\
-1 & -1 & -2 & 1 \\
-1 & -1 & -2 & 1 \\
-1 & 0 & -4 & 3 \\
\end{array}
\]
CNN – Max Pooling

6 x 6 image

New image but smaller

Each filter is a channel
The whole CNN

A new image

Smaller than the original image

The number of the channel is the number of filters

Can repeat many times
The whole CNN

cat dog ......

Fully Connected Feedforward network

Convolution → Max Pooling → A new image

Convolution → Max Pooling → A new image

Flatten
Flatten

Fully Connected Feedforward network
Convolutional Neural Network

Step 1: Convolutional Neural Network

Step 2: Goodness of function

Step 3: Pick the best function

Step 1:
Define a set of functions.

Step 2:
Evaluate the goodness of each function.

Step 3:
Select the best function.

Convolution, Max Pooling, fully connected

Learning: Nothing special, just gradient descent …..
**CNN in Keras**

Only modified the **network structure** and **input format** (vector -> 3-D tensor)

```python
model2.add(Convolution2D(25, 3, 3, input_shape=(1, 28, 28)))
```

Input_shape = (1, 28, 28)

1: black/weight, 3: RGB  
28 x 28 pixels

There are **25** 3x3 filters.

```python
model2.add(MaxPooling2D((2, 2)))
```

Max Pooling

Convolution

Max Pooling

Convolution
Only modified the network structure and input format (vector -> 3-D tensor)

**CNN in Keras**

Model parameters:
- **Input:** 1 x 28 x 28
- **First convolutional layer:** 25 x 26 x 26, 9 parameters
- **Max pooling:** 25 x 13 x 13
- **Second convolutional layer:** 50 x 11 x 11, 225 parameters
- **Max pooling:** 50 x 5 x 5
CNN in Keras

Only modified the **network structure** and input format (vector -> 3-D tensor)

input

1 x 28 x 28

Convolution

25 x 26 x 26

Max Pooling

25 x 13 x 13

Convolution

50 x 11 x 11

Max Pooling

50 x 5 x 5

Fully Connected Feedforward network

output

1250

Flatten

model2.add(Dense(output_dim=100))
model2.add(Activation('relu'))
model2.add(Dense(output_dim=10))
model2.add(Activation('softmax'))
Live Demo
What does CNN learn?

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter:

\[ a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a_{ij}^k \]

\[ x^* = \arg \max_x a^k \text{ (gradient ascent)} \]
**What does CNN learn?**

The output of the k-th filter is a 11 x 11 matrix.

Degree of the activation of the k-th filter:

\[ a^k = \sum_{i=1}^{11} \sum_{j=1}^{11} a^k_{ij} \]

Degree of the activation of the k-th filter:

\[ x^* = \arg\max_x a^k \] (gradient ascent)

For each filter
What does CNN learn?

\[ x^* = \arg \max_x y_i \]  
Can we see digits?

Deep Neural Networks are Easily Fooled

https://www.youtube.com/watch?v=M2IebCN9Ht4
What does CNN learn?

\[ x^* = \arg \max_x y^i \]

\[ x^* = \arg \max_x \left( y^i + \sum_{i,j} |x_{ij}| \right) \]
Deep Dream

• Given a photo, machine adds what it sees ……

http://deepdreamgenerator.com/

CNN exaggerates what it sees
Deep Dream

• Given a photo, machine adds what it sees ......

http://deepdreamgenerator.com/
Deep Style

• Given a photo, make its style like famous paintings

https://dreamscopeapp.com/
Deep Style

• Given a photo, make its style like famous paintings

https://dreamscopeapp.com/
Deep Style

A Neural Algorithm of Artistic Style
https://arxiv.org/abs/1508.06576
More Application: Playing Go

19 x 19 matrix (image)

Black: 1
white: -1
none: 0

19 x 19 vector

Fully-connected feedforward network can be used

But CNN performs much better.

Next move (19 x 19 positions)
More Application: Playing Go

Training: record of previous plays

黑: 5之五 → 白: 天元 → 黑: 五之5 ...

Target: “天元” = 1
else = 0

Target: “五之5” = 1
else = 0
Why CNN for playing Go?

• Some patterns are much smaller than the whole image

  Alpha Go uses 5 x 5 for first layer

• The same patterns appear in different regions.
Why CNN for playing Go?

- Subsampling the pixels will not change the object.

Max Pooling

How to explain this???

Neural network architecture. The input to the policy network is a $19 \times 19 \times 48$ image stack consisting of 48 feature planes. The first hidden layer zero pads the input into a $23 \times 23$ image, then convolves $k$ filters of kernel size $5 \times 5$ with stride 1 with the input image and applies a rectifier nonlinearity. Each of the subsequent hidden layers 2 to 12 zero pads the respective previous hidden layer into a $21 \times 21$ image, then convolves $k$ filters of kernel size $3 \times 3$ with stride 1, again followed by a rectifier nonlinearity. The final layer convolves 1 filter of kernel size $1 \times 1$ with stride 1, with a different bias for each position, and applies a softmax function. The Extended Data Table 3 additionally show the results of training with $k = 128, 256$ and 384 filters.

Alpha Go does not use Max Pooling ......
Variants of Neural Networks

Convolutional Neural Network (CNN)

Recurrent Neural Network (RNN)
Example Application

• Slot Filling

I would like to arrive Taipei on November 2\textsuperscript{nd}.

ticket booking system

Slot

\{\text{Destination: Taipei, time of arrival: November 2}\textsuperscript{nd}\}
Example Application

Solving slot filling by Feedforward network?

Input: a word

(Each word is represented as a vector)
1-of-N encoding

How to represent each word as a vector?

1-of-N Encoding  

<table>
<thead>
<tr>
<th>Lexicon</th>
<th>Vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>{apple, bag, cat, dog, elephant}</td>
<td></td>
</tr>
</tbody>
</table>

The vector is lexicon size. apple = [ 1 0 0 0 0 0 ]
Each dimension corresponds to a word in the lexicon. bag = [ 0 1 0 0 0 0 ]
The dimension for the word is 1, and others are 0. cat = [ 0 0 1 0 0 0 ]
dog = [ 0 0 0 1 0 ]
elephant = [ 0 0 0 0 1 ]
Beyond 1-of-N encoding

**Dimension for “Other”**

<table>
<thead>
<tr>
<th>Word</th>
<th>Representation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>apple</td>
<td>a-a-a</td>
<td>0</td>
</tr>
<tr>
<td>bag</td>
<td>a-a-b</td>
<td>0</td>
</tr>
<tr>
<td>cat</td>
<td>a-p-p</td>
<td>0</td>
</tr>
<tr>
<td>dog</td>
<td>p-l-e</td>
<td>0</td>
</tr>
<tr>
<td>elephant</td>
<td>p-p-l</td>
<td>0</td>
</tr>
<tr>
<td>“other”</td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

**Word hashing**

<table>
<thead>
<tr>
<th>Word</th>
<th>Representation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>“other”</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

w = “Gandalf”

w = “Sauron”

26 X 26 X 26

w = “apple”
Example Application

Solving slot filling by Feedforward network?

Input: a word
      (Each word is represented as a vector)

Output:
      Probability distribution that the input word belonging to the slots

Taipei
Example Application

arrive  
Taipei  on  November  2^{nd}

other  dest  other  time  time

Problem?

leave  
Taipei  on  November  2^{nd}

place of departure

Neural network needs memory!

time of departure

dest

\begin{align*}
y_1 &= x_1 \\
y_2 &= x_2
\end{align*}
Three Steps for Deep Learning

Step 1: Recurrent Neural Network

Step 2: goodness of function

Step 3: pick the best function

Deep Learning is so simple ......
Recurrent Neural Network (RNN)

The output of hidden layer are stored in the memory.

Memory can be considered as another input.
RNN

The same network is used again and again.

Probability of “arrive” in each slot

Probability of “Taipei” in each slot

Probability of “on” in each slot

Probability of “arrive” in each slot

Probability of “Taipei” in each slot

Probability of “on” in each slot

The same network is used again and again.
RNN

Prob of “leave” in each slot

Prob of “Taipei” in each slot

Prob of “arrive” in each slot

Prob of “Taipei” in each slot

The values stored in the memory is different.
Of course it can be deep ...
Bidirectional RNN
Long Short-term Memory (LSTM)

Memory Cell

Long Short-term Memory (LSTM)

Input Gate

Signal control the input gate
(Other part of the network)

Output Gate

Signal control the output gate
(Other part of the network)

Forget Gate

Signal control the forget gate
(Other part of the network)

Special Neuron: 4 inputs, 1 output

Other part of the network

Other part of the network

LSTM
Activation function $f$ is usually a sigmoid function

Between 0 and 1

Mimic open and close gate

$$c' = g(z)f(z_i) + cf(z_f)$$
LSTM

\[ c^{t-1} \]

vector

\[ x^t \)

4 vectors
LSTM

\[
\begin{align*}
\text{ct}^{-1} & \quad \text{zt} \quad \text{yt} \\
\text{xf} & \quad + & \quad \text{xi} & \quad \text{zf} & \quad \text{zo} \\
\text{zf} & \quad \times & \quad \text{yi} & \quad \text{zo} \\
\end{align*}
\]
LSTM

Extension: “peephole”
Multiple-layer LSTM

Don’t worry if you cannot understand this. Keras can handle it.

Keras supports “LSTM”, “GRU”, “SimpleRNN” layers

This is quite standard now.

https://img.komicolle.org/2015-09-20/src/14426967627131.gif
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 ......
Learning Target

Training Sentences:

arrive Taipei on November 2nd

other dest other time time
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 ......
Learning

RNN Learning is very difficult in practice.

Backpropagation through time (BPTT)

\[ w \leftarrow w - \eta \frac{\partial L}{\partial w} \]
Unfortunately ……

- RNN-based network is not always easy to learn

Real experiments on Language modeling

Total Loss

Epoch

sometimes

Lucky
The error surface is rough.

The error surface is either very flat or very steep.

[Clipping]

[Total Loss]

[Razvan Pascanu, ICML’13]
Why?

$w = 1 \quad \Rightarrow \quad y^{1000} = 1$
$w = 1.01 \quad \Rightarrow \quad y^{1000} \approx 20000$
$w = 0.99 \quad \Rightarrow \quad y^{1000} \approx 0$
$w = 0.01 \quad \Rightarrow \quad y^{1000} \approx 0$

Large $\frac{\partial L}{\partial w}$
Small Learning rate?

Small $\frac{\partial L}{\partial w}$
Large Learning rate?

Toy Example
Helpful Techniques

- Long Short-term Memory (LSTM)
  - Can deal with gradient vanishing (not gradient explode)
  - Memory and input are added
  - The influence never disappears unless forget gate is closed

  No Gradient vanishing
  (If forget gate is opened.)

Gated Recurrent Unit (GRU): simpler than LSTM

[Cho, EMNLP’14]
Helpful Techniques

Clockwise RNN

[Jan Koutnik, JMLR’14]

Structurally Constrained Recurrent Network (SCRN)

[ Tomas Mikolov, ICLR’15]

Vanilla RNN Initialized with Identity matrix + ReLU activation function [Quoc V. Le, arXiv’15]

➢ Outperform or be comparable with LSTM in 4 different tasks
More Applications …

Probability of “arrive” in each slot

Probability of “Taipei” in each slot

Probability of “on” in each slot

Input and output are both sequences with the same length

RNN can do more than that!

arrive   Taipei   on   November 2\textsuperscript{nd}
Many to one

- Input is a vector sequence, but output is only one vector

**Sentiment Analysis**

- Positive (正雷)
- Negative (負雷)
- Positive (正雷)

我覺得很糟了
這部電影太糟了
這部電影很棒
看了這部電影覺得很高興

超好雷
好雷
普雷
負雷
超負雷
Many to one

• Input is a vector sequence, but output is only one vector

Key Term Extraction

[Shen & Lee, Interspeech 16]
Many to Many (Output is shorter)

- Both input and output are both sequences, **but the output is shorter.**
  - E.g. **Speech Recognition**

**Problem?**

Why can’t it be “好棒棒”?

<table>
<thead>
<tr>
<th>Input:</th>
<th>Output: “好棒” (character sequence)</th>
</tr>
</thead>
</table>

Trimming
Many to Many (Output is shorter)

• Both input and output are both sequences, **but the output is shorter.**

• Connectionist Temporal Classification (CTC) [Alex Graves, ICML’06][Alex Graves, ICML’14][Haşim Sak, Interspeech’15][Jie Li, Interspeech’15][Andrew Senior, ASRU’15]

Add an extra symbol “φ” representing “null”
Many to Many (No Limitation)

- Both input and output are both sequences with different lengths. → **Sequence to sequence learning**
  - E.g. **Machine Translation** (machine learning→機器學習)

[Diagram showing the relationship between machine and learning, containing all information about input sequence]
Many to Many (No Limitation)

• Both input and output are both sequences *with different lengths*. → *Sequence to sequence learning*
  • E.g. **Machine Translation** (machine learning→機器學習)

Don’t know when to stop
Many to Many (No Limitation)

Ref: http://zh.pttpedia.wikia.com/wiki/%E6%8E%A5%E9%BE%8D%E6%8E%A8%E6%96%87 (鄉民百科)
Many to Many (No Limitation)

• Both input and output are both sequences with different lengths. → Sequence to sequence learning
  • E.g. Machine Translation (machine learning \(\rightarrow\) 機器學習)

Add a symbol “===” (斷)

[Ilya Sutskever, NIPS’14][Dzmitry Bahdanau, arXiv’15]
Image Caption Generation

• Input an image, but output a sequence of words

[Kelvin Xu, arXiv’15][Li Yao, ICCV’15]
Image Caption Generation

• Can machine describe what it see from image?
• Demo: 台大電機系 大四 蘇子睿、林奕辰、徐翊祥、陳奕安

MTK 產學大聯盟
http://news.ltn.com.tw/photo/politics/breakingnews/975542_1
Video Caption Generation

Video

A girl is running.

A group of people is knocked by a tree.

A group of people is walking in the forest.
Video Caption Generation

• Can machine describe what it see from video?
• Demo: 台大語音處理實驗室 曾柏翔、吳柏瑜、盧宏宗
Chat-bot

電視影集 (~40,000 sentences)、美國總統大選辯論
Demo

• Develop Team
  • Interface design: Prof. Lin-Lin Chen & Arron Lu
  • Web programming: Shi-Yun Huang
  • Data collection: Chao-Chuang Shih
  • System implementation: Kevin Wu, Derek Chuang, & Zhi-Wei Lee
  • System design: Richard Tsai & Hung-Yi Lee
Attention-based Model

What is deep learning?

What you learned in these lectures

Breakfast today

summer vacation 10 years ago

http://henrylo1605.blogspot.tw/2015/05/blog-post_56.html
Attention-based Model

Ref:
Attention-based Model v2

Input \rightarrow \text{DNN/RNN} \rightarrow \text{output}

\begin{itemize}
\item \text{Reading Head Controller}
\item \text{Writing Head Controller}
\item \text{Writing Head}
\item \text{Reading Head}
\item \text{Machine’s Memory}
\end{itemize}

Neural Turing Machine
Reading Comprehension

Query → DNN/RNN → answer

Reading Head Controller → Semantic Analysis

Each sentence becomes a vector.
Reading Comprehension


The position of reading head:

<table>
<thead>
<tr>
<th>Story (16: basic induction)</th>
<th>Support</th>
<th>Hop 1</th>
<th>Hop 2</th>
<th>Hop 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Brian is a frog.</td>
<td>yes</td>
<td>0.00</td>
<td>0.98</td>
<td>0.00</td>
</tr>
<tr>
<td>Lily is gray.</td>
<td></td>
<td>0.07</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Brian is yellow.</td>
<td>yes</td>
<td>0.07</td>
<td>0.00</td>
<td>1.00</td>
</tr>
<tr>
<td>Julius is green.</td>
<td></td>
<td>0.06</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>Greg is a frog.</td>
<td>yes</td>
<td>0.76</td>
<td>0.02</td>
<td>0.00</td>
</tr>
</tbody>
</table>

What color is Greg? Answer: yellow Prediction: yellow

Keras has example:
https://github.com/fchollet/keras/blob/master/examples/babi_memnn.py
Visual Question Answering

What is the mustache made of?

source: http://visualqa.org/
Visual Question Answering

Query → DNN/RNN → Reading Head Controller → CNN → answer

A vector for each region
Speech Question Answering

• TOEFL Listening Comprehension Test by Machine
• Example:

Audio Story:  (The original story is 5 min long.)
Question: “What is a possible origin of Venus’ clouds?”
Choices:
(A) gases released as a result of volcanic activity
(B) chemical reactions caused by high surface temperatures
(C) bursts of radio energy from the plane's surface
(D) strong winds that blow dust into the atmosphere
... It be quite possible that this be due to volcanic eruption because volcanic eruption often emit gas. If that be the case volcanism could very well be the root cause of Venus 's thick cloud cover. And also we have observe burst of radio energy from the planet 's surface. These burst be similar to what we see when volcano erupt on earth ......

Question: “what is a possible origin of Venus‘ clouds?"
Simple Baselines

Experimental setup: 717 for training, 124 for validation, 122 for testing

(2) select the **shortest** choice as answer

(4) the choice with semantic most similar to others
Memory Network

Memory Network: 39.2%
(proposed by FB AI group)

Naive Approaches
Proposed Approach

Memory Network: 39.2%
(proposed by FB AI group)

Proposed Approach: 48.8%

Naive Approaches

[Tseng & Lee, Interspeech 16]
[Fang & Hsu & Lee, SLT 16]
Concluding Remarks

- Convolutional Neural Network (CNN)
- Recurrent Neural Network (RNN)
Lecture III: Beyond Supervised Learning
Outline

Unsupervised Learning

• 化繁為簡
  • Auto-encoder
  • Word Vector and Audio Word Vector

• 無中生有

Reinforcement Learning
Unsupervised Learning

• 化繁為簡
• 無中生有

only having function input

function

only having function output

function

code
Outline

Unsupervised Learning

- 化繁為簡
  - Auto-encoder
  - Word Vector and Audio Word Vector
- 無中生有

Reinforcement Learning
Motivation

• In MNIST, a digit is 28 x 28 dims.
• Most 28 x 28 dim vectors are not digits
Outline

Unsupervised Learning

• 化繁為簡
  • Auto-encoder
  • Word Vector and Audio Word Vector
• 無中生有

Reinforcement Learning
Auto-encoder

28 x 28 = 784

Usually < 784

Compact representation of the input object

Learn together

Can reconstruct the original object

As close as possible
Deep Auto-encoder

- NN encoder + NN decoder = a deep network

Deep Auto-encoder

Original Image

PCA

Deep Auto-encoder

0 / 2 3 4

0 / 2 3 4

784 → 30 → 784

q → q

784 → 1000 → 500 → 250 → 30 → 250 → 500 → 1000 → 784

q → q
Auto-encoder

- De-noising auto-encoder

As close as possible

Add noise


Auto-encoder – Pre-training DNN

• Greedy Layer-wise Pre-training *again*

Target

Input 784

\[
\begin{align*}
\text{Target} & : 
1000 \\
& \uparrow \\
& 500 \\
& \uparrow \\
& 10 \\
& \uparrow \\
\end{align*}
\]

\[
\begin{align*}
\text{Input} & : 
784 \\
\end{align*}
\]
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

Input: 784

Output: 10

Target:
- Input: 784
- fix: 1000
- $W^1$
- $W^2$
- $W^2'$
- $a^1$
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training again

```
<table>
<thead>
<tr>
<th>Layer</th>
<th>Input</th>
<th>784</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Output</td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>Target</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td></td>
<td>500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td></td>
</tr>
<tr>
<td>Input</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
```
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training \textit{again}

\begin{align*}
\text{Input} & : 784 \\
W^1 & : \text{Random init} \\
W^2 & : \text{Find-tune by backpropagation} \\
W^3 & : \text{Find-tune by backpropagation} \\
W^4 & : \text{Find-tune by backpropagation}
\end{align*}
Outline

Unsupervised Learning

• 化繁為簡
  • Auto-encoder
  • Word Vector and Audio Word Vector

• 無中生有

Reinforcement Learning
Word Vector/Embedding

- Machine learn the meaning of words from reading a lot of documents without supervision
Word Embedding

• Machine learn the meaning of words from reading a lot of documents without supervision
• A word can be understood by its context

蔡英文、馬英九 are something very similar

You shall know a word by the company it keeps
How to exploit the context?

• **Count based**
  • If two words $w_i$ and $w_j$ frequently co-occur, $V(w_i)$ and $V(w_j)$ would be close to each other

\[ V(w_i) \cdot V(w_j) \quad N_{i,j} \]

  - Inner product
  - Number of times $w_i$ and $w_j$ in the same document

• **Prediction based**
Prediction-based

1-of-N encoding of the word $w_{i-1}$

Take out the input of the neurons in the first layer

Use it to represent a word $w$

Word vector, word embedding feature: $V(w)$

The probability for each word as the next word $w_i$
Prediction-based

Collect data:
潮水 退了 就 知道 … 不爽 不要 買 … 公道價 八萬 一 … ……

Minimizing cross entropy
Prediction-based

You shall know a word by the company it keeps

The probability for each word as the next word $w_i$

“宣誓就職” should have large probability

Training text:

...... 蔡英文 宣誓就職 ...... $w_{i-1}$ $w_i$

...... 馬英九 宣誓就職 ...... $w_{i-1}$ $w_i$
Prediction-based – Various Architectures

• Continuous bag of word (CBOW) model

• Skip-gram

predicting the word given its context

predicting the context given a word
Word Embedding

Source: http://www.slideshare.net/hustwj/cikm-keynotenov2014
Word Embedding

- **Characteristics**

  \[ V(Germany) \approx V(Berlin) - V(Rome) + V(Italy) \]

  \[ V(hotter) - V(hot) \approx V(bigger) - V(big) \]

  \[ V(Rome) - V(Italy) \approx V(Berlin) - V(Germany) \]

  \[ V(king) - V(queen) \approx V(uncle) - V(aunt) \]

- **Solving analogies**

  Rome : Italy = Berlin : ?

  Compute \[ V(Berlin) - V(Rome) + V(Italy) \]

  Find the word w with the closest V(w)
Demo

• Machine learn the meaning of words from reading a lot of documents without supervision
Demo

- Model used in demo is provided by 陳仰德
- Part of the project done by 陳仰德、林資偉
- TA: 劉元銘
- Training data is from PTT (collected by 葉青峰)
Document to Vector

- **Paragraph Vector**: Le, Quoc, and Tomas Mikolov. "Distributed Representations of Sentences and Documents." ICML, 2014


- **Exploiting other kind of labels:**
Audio Word to Vector

Machine does not have any prior knowledge

Machine listens to lots of audio book

Like an infant

[Chung, Interspeech 16]
Audio Word to Vector

- Dimension reduction for a sequence with variable length audio segments (word-level) → Fixed-length vector

Sequence-to-sequence Auto-encoder

The values in the memory represent the whole audio segment.

The vector we want.

How to train RNN Encoder?

acoustic features

audio segment
Sequence-to-sequence Auto-encoder

The RNN encoder and decoder are jointly trained.

RNN Encoder

Input acoustic features

RNN Decoder

acoustic features

audio segment
Sequence-to-sequence Auto-encoder

• Visualizing embedding vectors of the words
Compute similarity between spoken queries and audio files on acoustic level, and find the query term.
Audio Word to Vector – Application

Audio archive divided into variable-length audio segments

Off-line

On-line

Spoken Query

Audio Segment to Vector

Audio Segment to Vector

Similarity

Search Result
Experimental Results

• Query-by-Example Spoken Term Detection

**SA**: sequence auto-encoder

**DSA**: de-noising sequence auto-encoder

**Input**: clean speech + noise

**output**: clean speech
Next Step ......

• Can we include semantics?
Outline

Unsupervised Learning

• 化繁為簡
  • Auto-encoder
  • Word Vector and Audio Word Vector

Reinforcement Learning
Creation

Draw something!
Creation

• Generative Models:
  https://openai.com/blog/generative-models/

What I cannot create, I do not understand.

Richard Feynman

PixelRNN

• To create an image, generating a pixel each time

E.g. 3 x 3 images

Can be trained just with a large collection of images without any annotation

PixelRNN

Real World

PixelRNN – beyond Image


Auto-encoder

As close as possible

Randomly generate a vector as code

Variation Auto-encoder (VAE)

Ref: Auto-Encoding Variational Bayes,
https://arxiv.org/abs/1312.6114
Auto-encoder

From a normal distribution

Minimize reconstruction error

\[
\sum_{i=1}^{3} (\exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)
\]
Why VAE?

decode

code

encode

?
VAE

Cifar-10

https://github.com/openai/iaf

**VAE - Writing Poetry**

![Diagram of the VAE model for writing poetry]

---

Ref: [http://www.wired.co.uk/article/google-artificial-intelligence-poetry](http://www.wired.co.uk/article/google-artificial-intelligence-poetry)
Problems of VAE

• It does not really try to simulate real images
Generative Adversarial Network (GAN)

What are some recent and potentially upcoming breakthroughs in unsupervised learning?

Yann LeCun, Director of AI Research at Facebook and Professor at NYU
Written Jul 29 · Upvoted by Joaquin Quiñonero Candela, Director Applied Machine Learning at Facebook and Huang Xiao

Adversarial training is the coolest thing since sliced bread.

I’ve listed a bunch of relevant papers in a previous answer.

Expect more impressive results with this technique in the coming years.

What’s missing at the moment is a good understanding of it so we can make it work reliably. It’s very finicky. Sort of like ConvNet were in the 1990s, when I had the reputation of being the only person who could make them work (which wasn’t true).

擬態的演化

蝴蝶不是棕色
蝴蝶沒有葉脈

棕色
葉脈

http://peellden.pixnet.net/blog/post/40406899-2013-%E7%AC%AC%E5%9B%9B%E5%AD%A3%EF%BC%8C%E5%86%AC%E8%9D%B6%E5%AF%82%E5%AF%A5

……..
The evolution of generation

NN Generator v1

NN Generator v2

NN Generator v3

Discriminator v1

Discriminator v2

Discriminator v3

Real images: 5 0 4 1

Binary Classifier
Cifar-10

- Which one is machine-generated?

Ref: https://openai.com/blog/generative-models/
畫漫畫

• Ref: https://github.com/mattya/chainer-DCGAN
『畫漫畫』

- Ref: http://qiita.com/mattyacom/items/e5bfe5e04b9d2f0bbd47

長髪化ベクトル

一番左のキャラクターが元画像で、右に行くほど長髪化ベクトルを強く足している
Want to practice Generation Models?
Pokémon Creation

• Small images of 792 Pokémon's
  • Can machine learn to create new Pokémons?

Don't catch them! Create them!

• Source of image:
  http://bulbapedia.bulbagarden.net/wiki/List_of_Pokémon_by_base_stats_(Generation_VI)

Original image is 40 x 40
Making them into 20 x 20
Pokémon Creation

➢ Each pixel is represented by 3 numbers (corresponding to RGB)

➢ Each pixel is represented by a 1-of-N encoding feature

Clustering the similar color

167 colors in total
Real Pokémon
Never seen by machine!

Cover 50%
It is difficult to evaluate generation.

Cover 75%
Pokémon Creation

Drawing from scratch
Need some randomness
Pick two dim, and fix the rest eight
Pokémon Creation - Data

• Original image (40 x 40):
  [Link](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/image.rar)

• Pixels (20 x 20):
  [Link](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/pixel_color.txt)
  • Each line corresponds to an image, and each number corresponds to a pixel
  • [Link](http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2016/Pokemon_creation/colormap.txt)

You can use the data without permission
Outline

Unsupervised Learning

• 化繁為簡
  • Example: Word Vector and Audio Word Vector

• 無中生有

Reinforcement Learning
Scenario of Reinforcement Learning

Agent

Observation → Action

Don’t do that

Reward

Environment
Scenario of Reinforcement Learning

Agent learns to take actions to maximize expected reward.

http://www.sznews.com/news/content/2013-11/26/content_8800180.htm
Supervised v.s. Reinforcement

• Supervised
  Learning from teacher
  Hello
  • Reinforcement
  Learning from critics
  Agent

“Hello” Say “Hi”
“Bye bye” Say “Good bye”

Bad
Scenario of Reinforcement Learning

Agent learns to take actions to maximize expected reward.

If win, reward = 1
If loss, reward = -1
Otherwise, reward = 0
Supervised v.s. Reinforcement

• Supervised:

First move 

...... many moves ...... 

Win!

Next move: 

“5-5”

Next move: 

“3-3”

• Reinforcement Learning

Alpha Go is supervised learning + reinforcement learning.
Difficulties of Reinforcement Learning

• It may be better to sacrifice immediate reward to gain more long-term reward
  • E.g. Playing Go
• Agent’s actions affect the subsequent data it receives
  • E.g. Exploration
Deep Reinforcement Learning

- **Observation**
- **Function Input**
- **DNN**
- **Function Output**
- **Action**
- **Environment**
- **Reward**

Used to pick the best function
Application: Interactive Retrieval

• Interactive retrieval is helpful. [Wu & Lee, INTERSPEECH 16]

“Deep Learning” related to Machine Learning?
“Deep Learning” related to Education?
Deep Reinforcement Learning

• Different network depth

![Graph showing total reward over training epochs for linear, 2-layer, and 4-layer models. The graph indicates that while the 2-layer model shows some improvement, the task cannot be addressed by linear model.]

- Better retrieval performance
- Less user labor

Some depth is needed.

The task cannot be addressed by linear model.
More applications

• Alpha Go, Playing Video Games, Dialogue
• Flying Helicopter
  • https://www.youtube.com/watch?v=0JL04JJjocc
• Driving
  • https://www.youtube.com/watch?v=0xo1Ldx3L5Q
• Google Cuts Its Giant Electricity Bill With DeepMind-Powered AI
To learn deep reinforcement learning ......

• Lectures of David Silver
  • http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Teaching.html
  • 10 lectures (1:30 each)

• Deep Reinforcement Learning
  • http://videolectures.net/rldm2015_silver_reinforcement_learning/
Conclusion
如何成為武林高手

• 內外兼修
  • 內功充沛，恃強克弱
  • 招數精妙，以快打慢

• Deep Learning 也需要內外兼修
  • 內力：運算資源
  • 招數：各種技巧

• 內力充沛, 平常的招式也有可能發會巨大的威力

• 只有內力，沒有招數
  • WavNet 並不是只憑蠻力

希望大家都可以成為內外兼修的高手