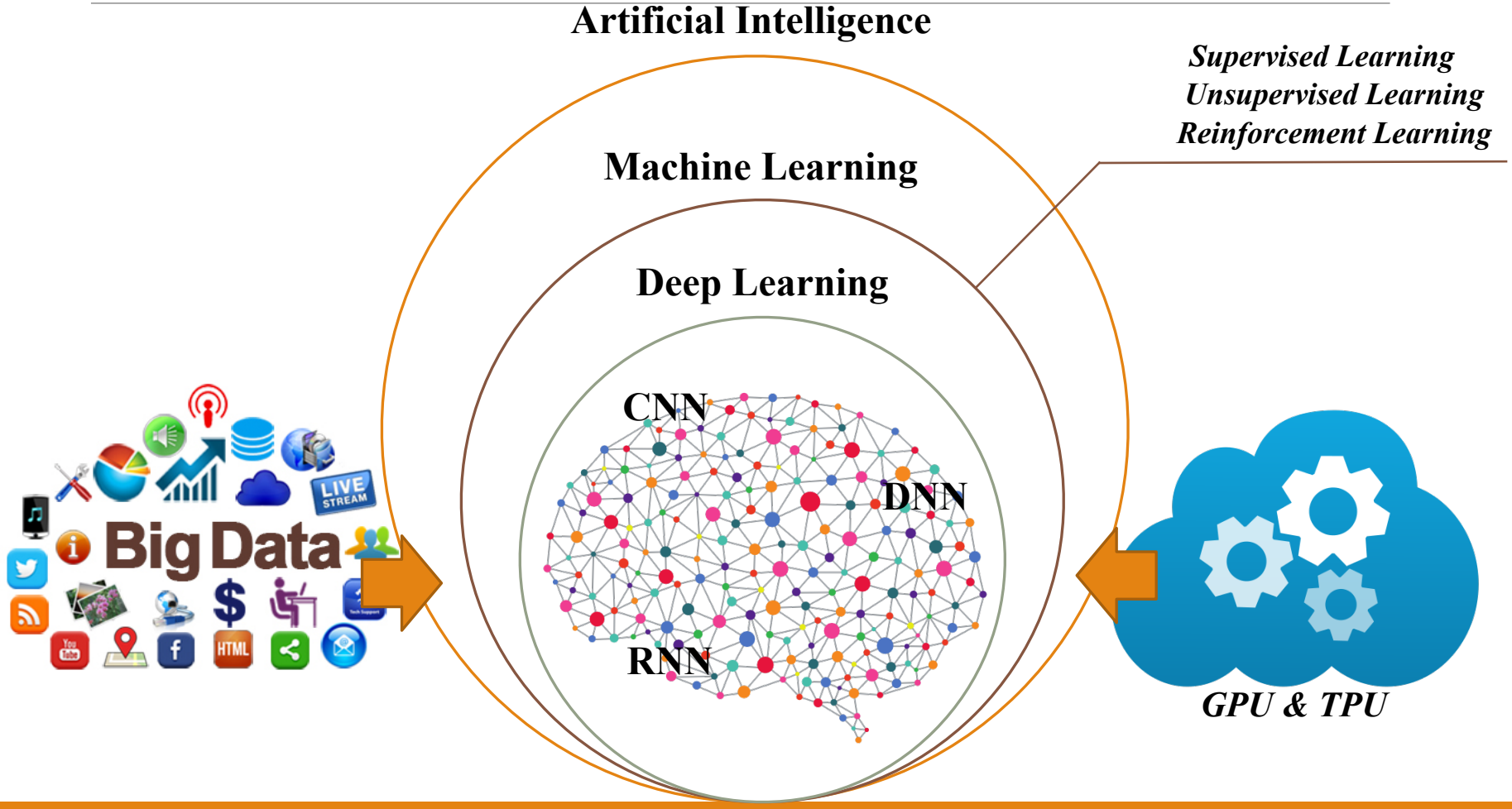
Neural Network for Classification

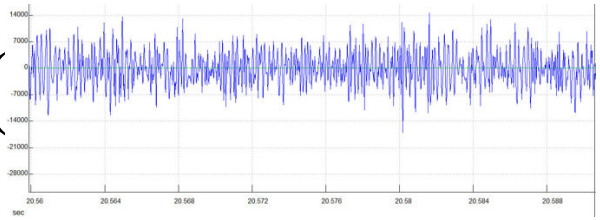
YUNG-CHUN CHANG

The relationship between AI, ML, and DL



Applications

Speech Recognition

$f^*($  $) = \text{Hello}$

Handwritten Recognition

$f^*($  $) = 5$

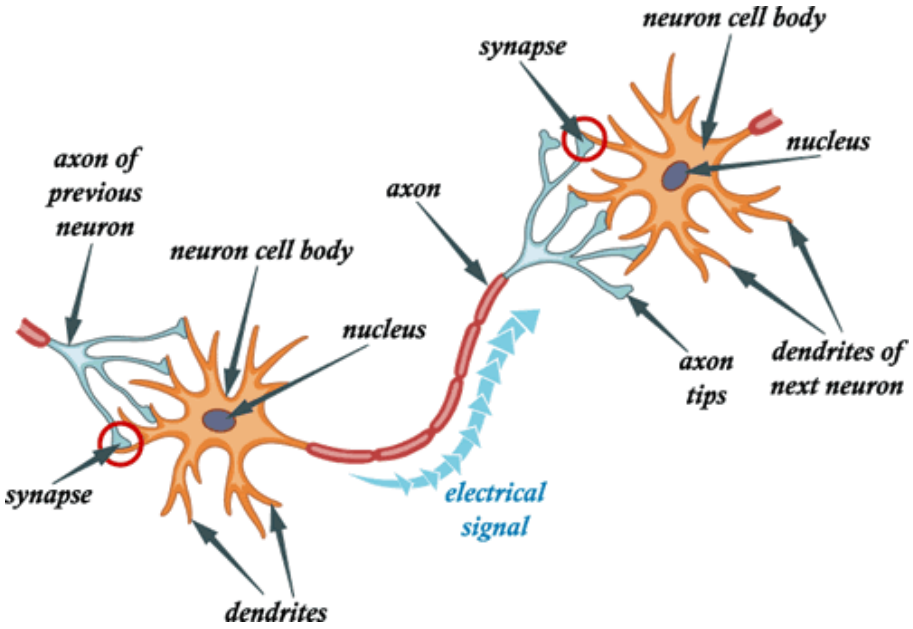
Playing Go

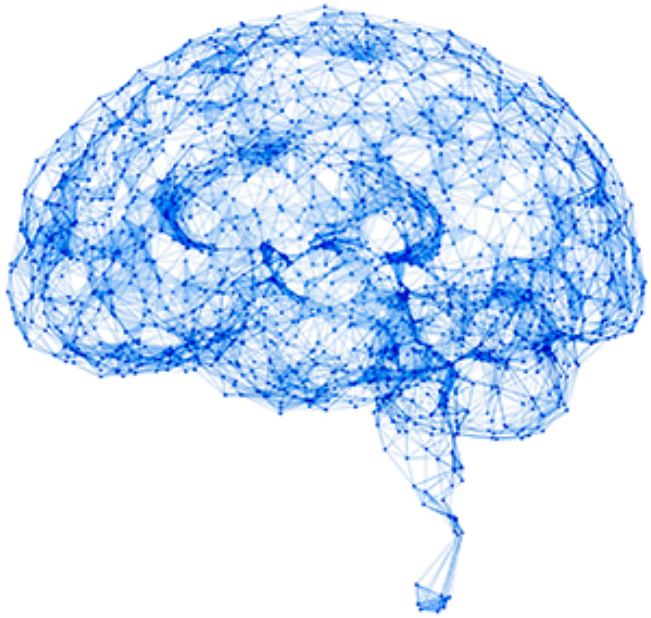
$f^*($  $) = 5 \text{ -> } 15$

Dialogue System

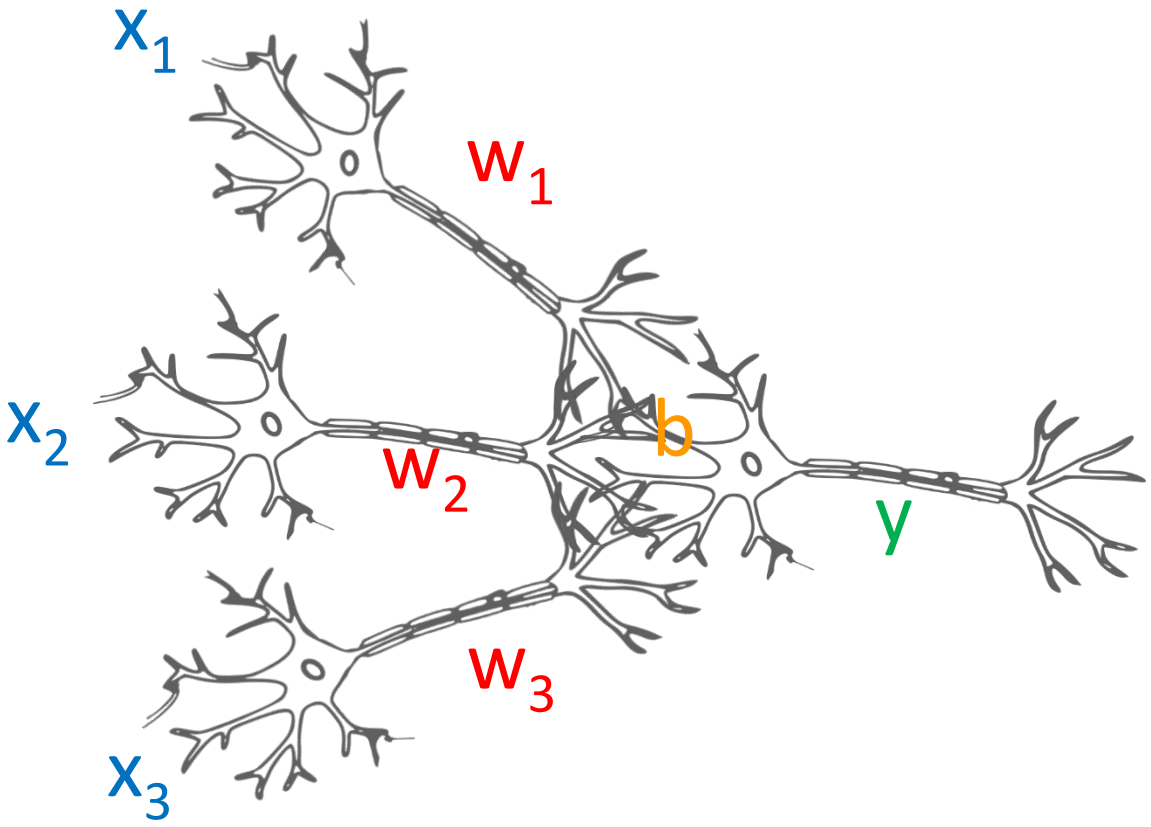
$f^*($ “Hello” $) =$ “Hi”

Basic Ideas

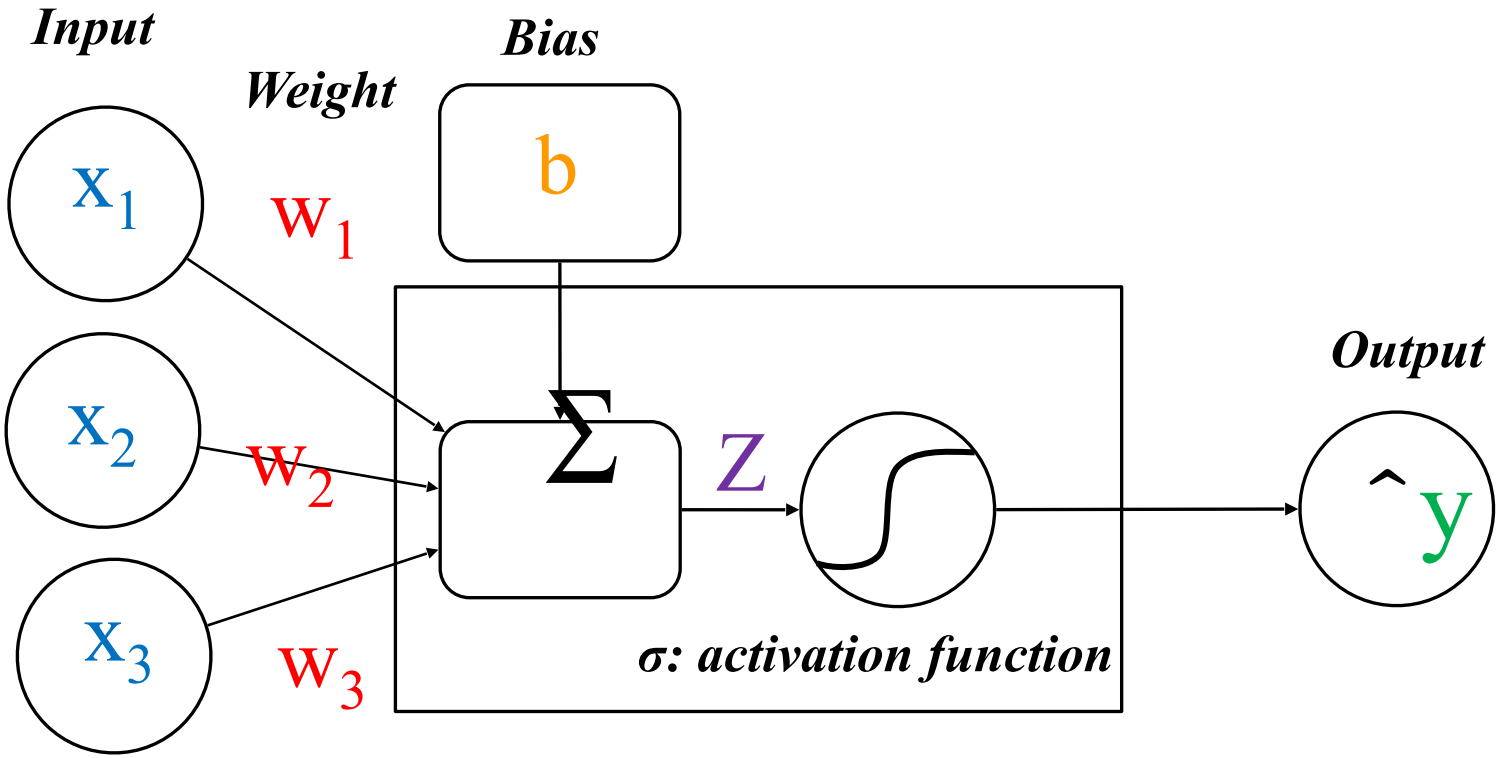

axon of previous neuron
neuron cell body
nucleus
synapse
dendrites
axon
electrical signal
synapse
neuron cell body
nucleus
axon tips
dendrites of next neuron



Basic Ideas (cont.)



Basic Ideas (cont.)



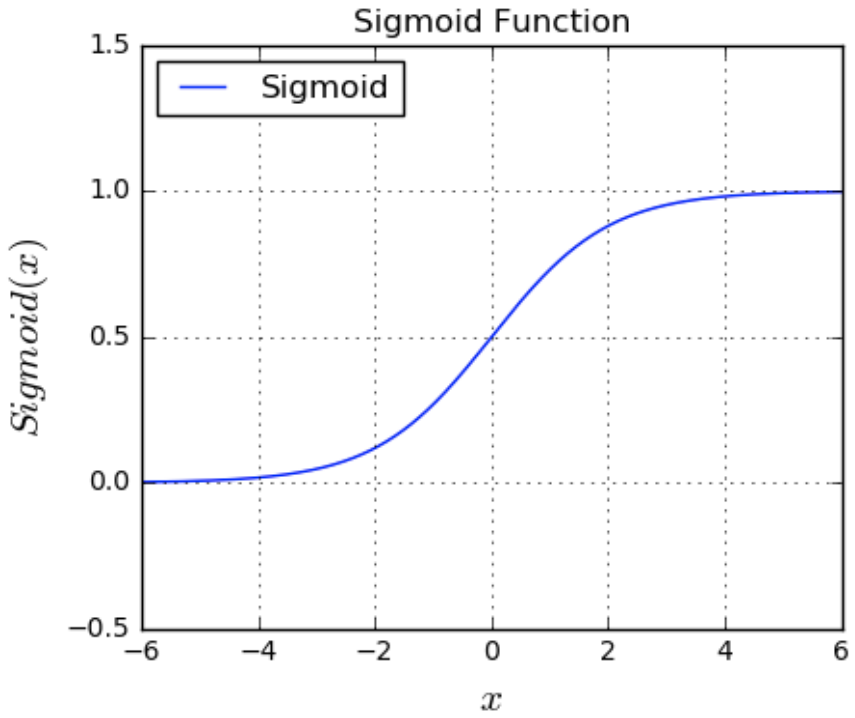
$$Z = (x1 * w1 + x2 * w2 + x3 * w3 + b)$$

$$\hat{y} = \sigma(Z)$$

Activation Function - Sigmoid

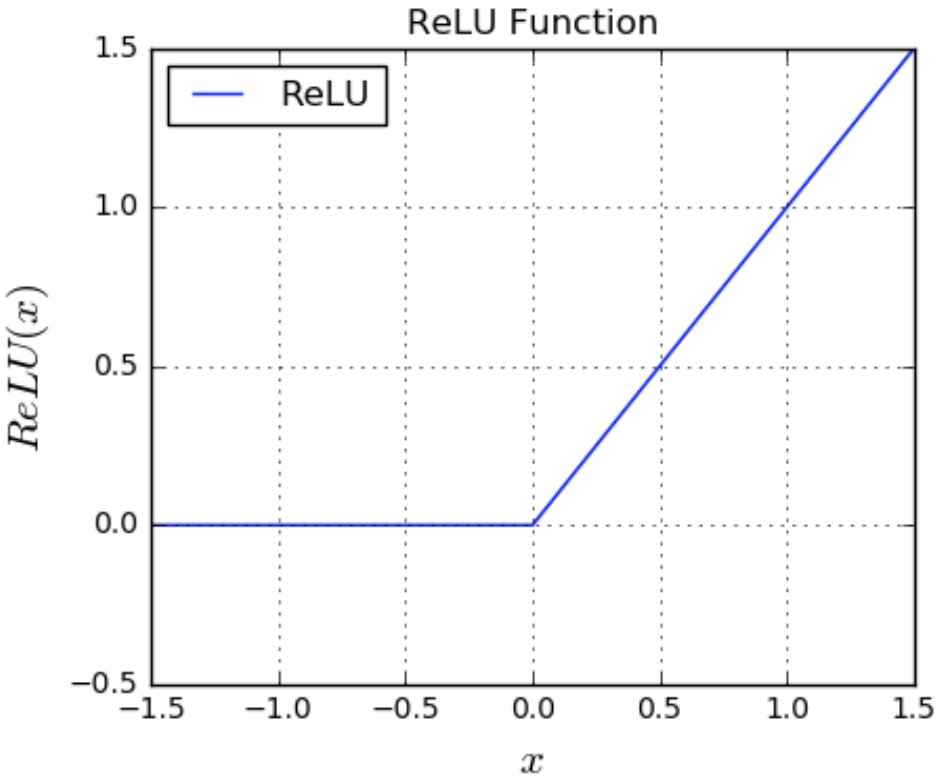
Ranging from 0 to 1

$$f(x) = \frac{1}{1 + e^{-x}}$$



Activation Function - ReLU

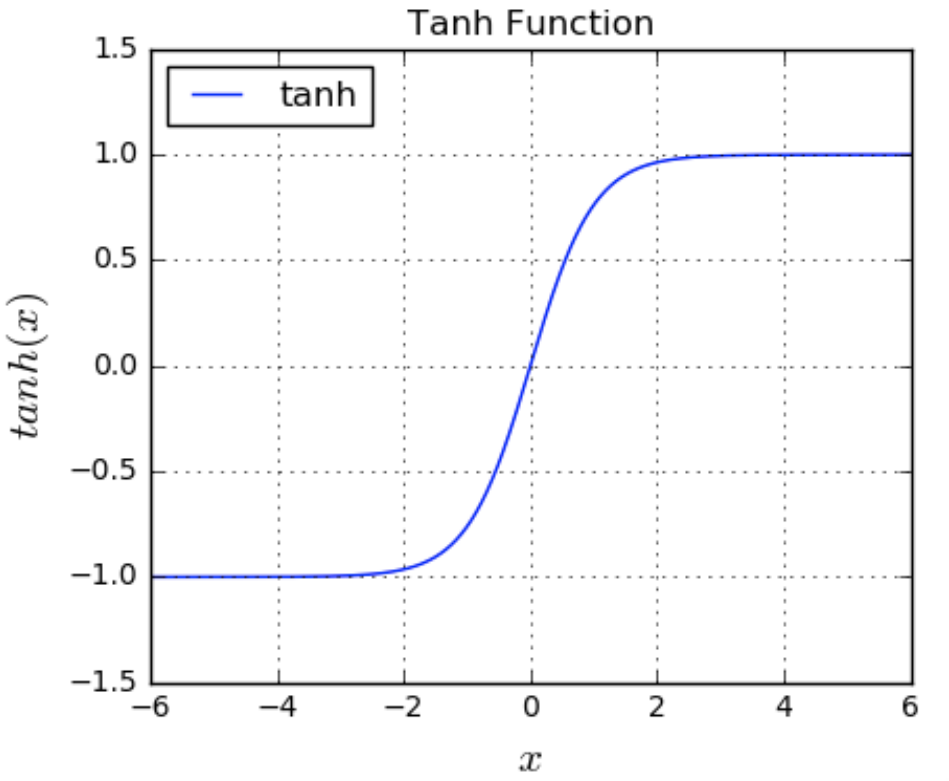
Ranging from 0 to x $f(x) = \max(0,x)$



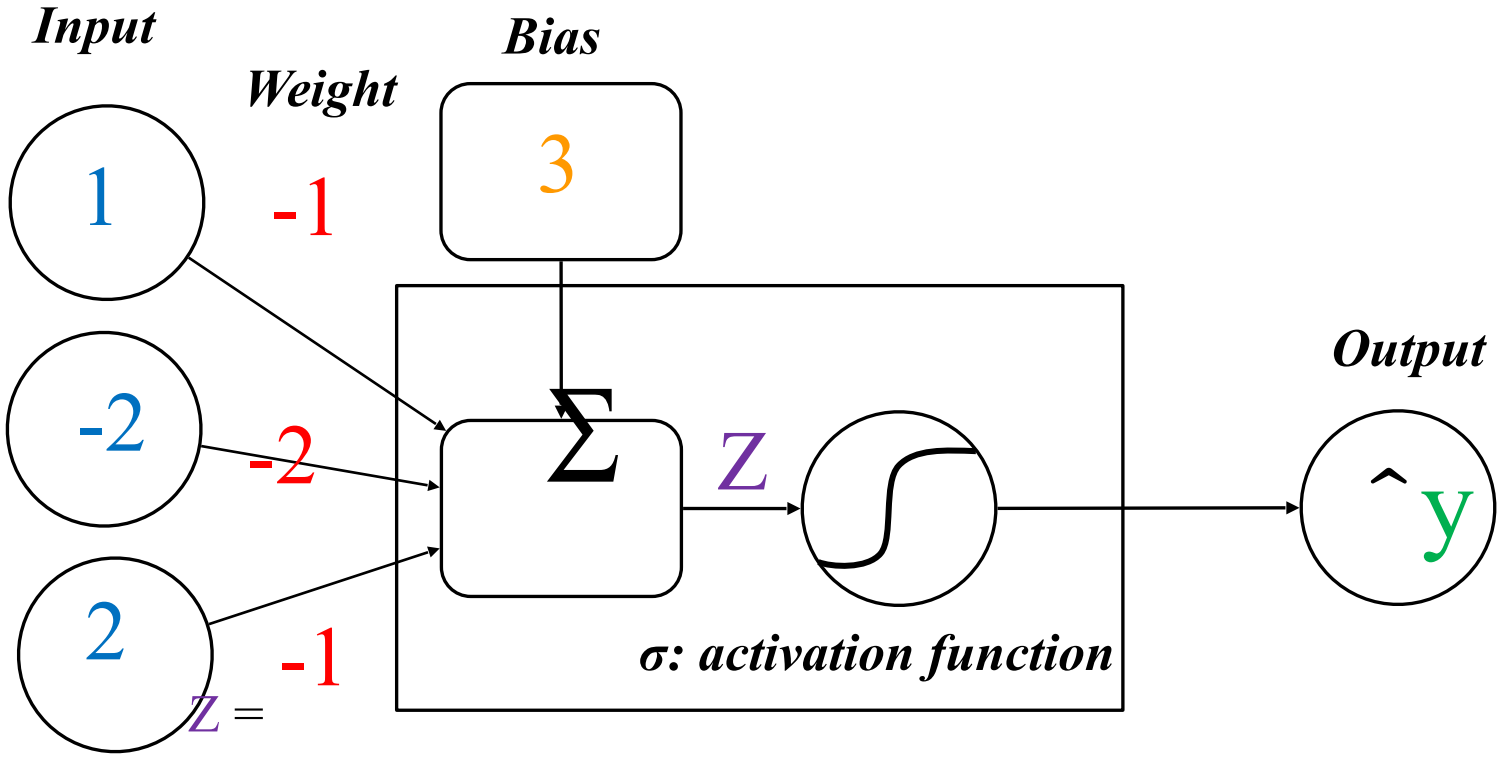
Activation Function - Tanh

Ranging from -1 to 1

$$f(x) = \frac{sinhx}{conshx} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$



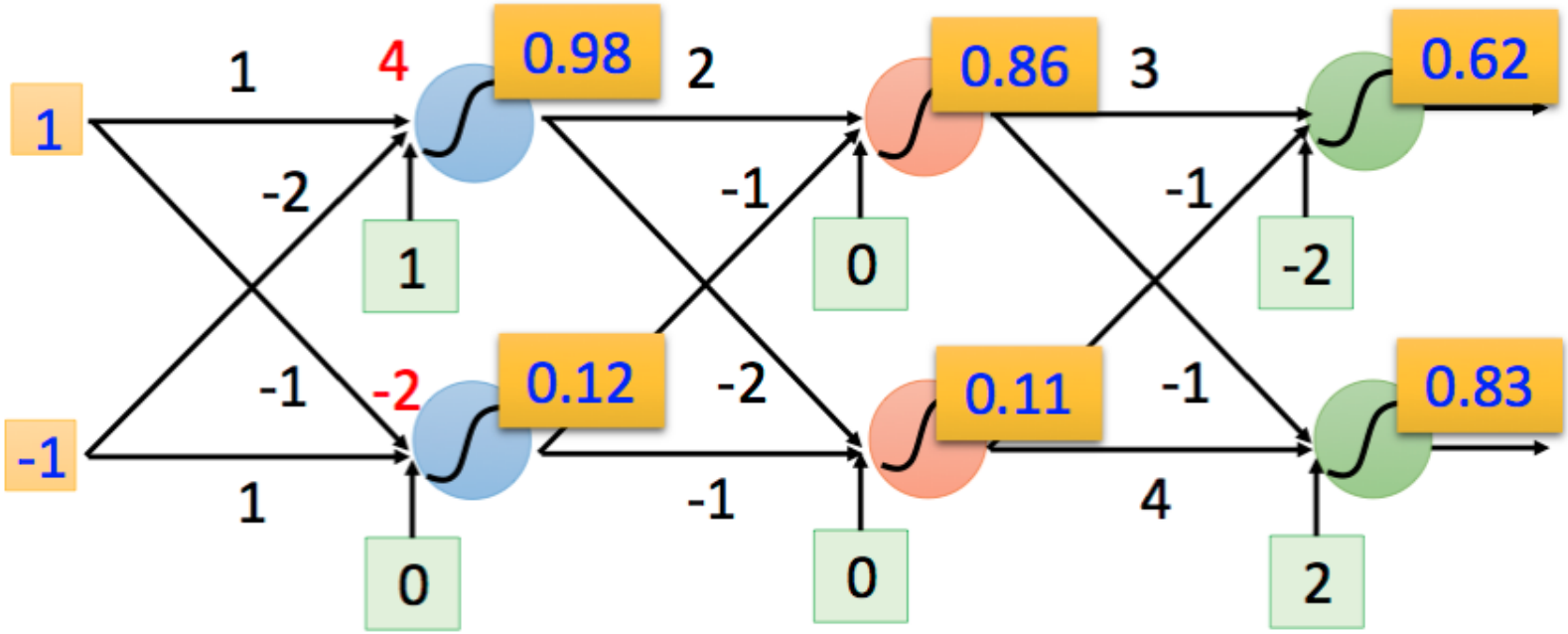
Forward Calculation



$$(1 * (-1) + (-2) * (-2) + 2 * (-1) + 3)$$

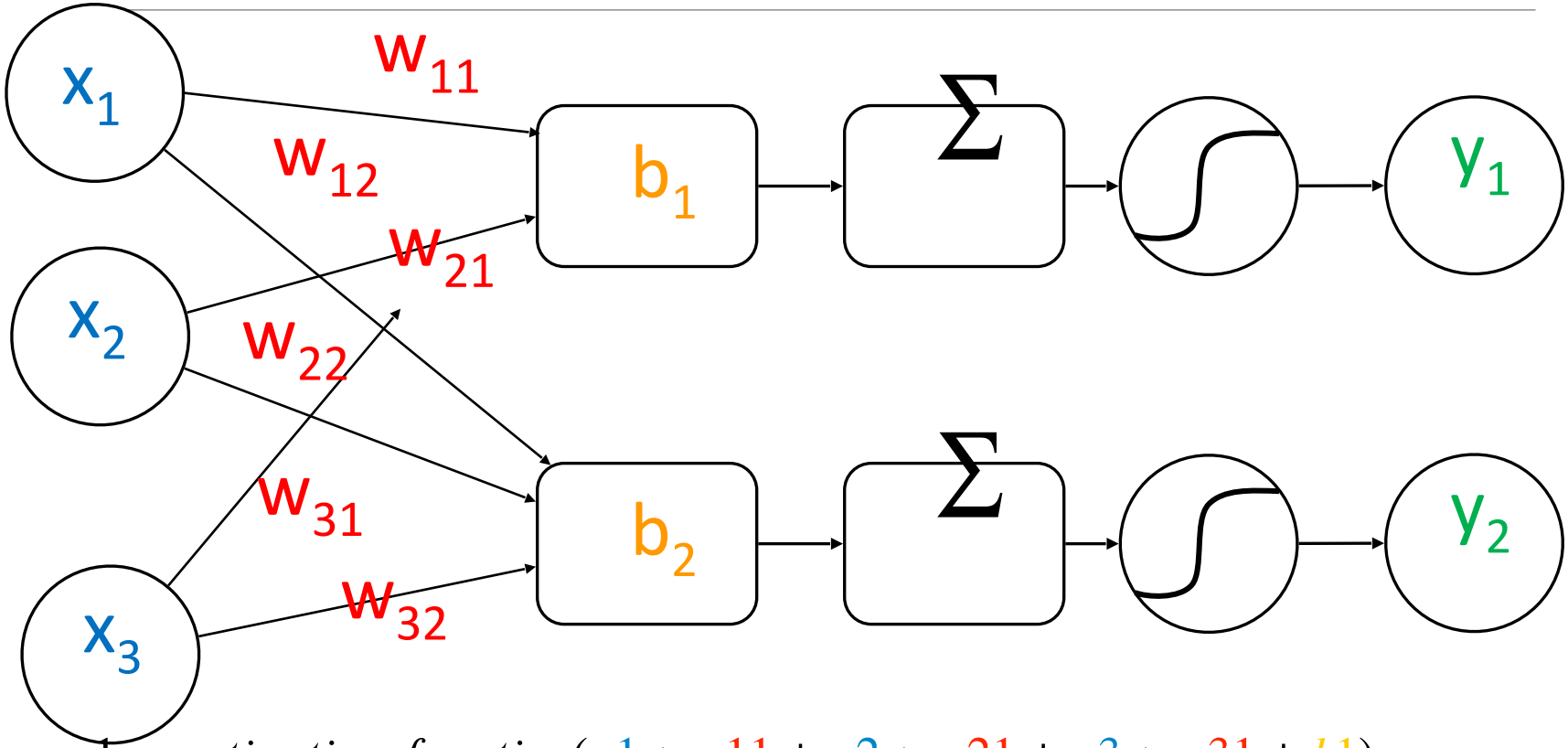
$$y = \sigma(4) = \frac{1}{1.0183} = 0.98$$

Forward Calculation (cont.)



Source: http://speech.ee.ntu.edu.tw/~tlkagk/research.html

Computation of Matrix



$$y1 = activation\ function(x1 * w11 + x2 * w21 + x3 * w31 + b1)$$

$$y2 = activation\ function(x1 * w12 + x2 * w22 + x3 * w32 + b2)$$

Computation of Matrix (cont.)

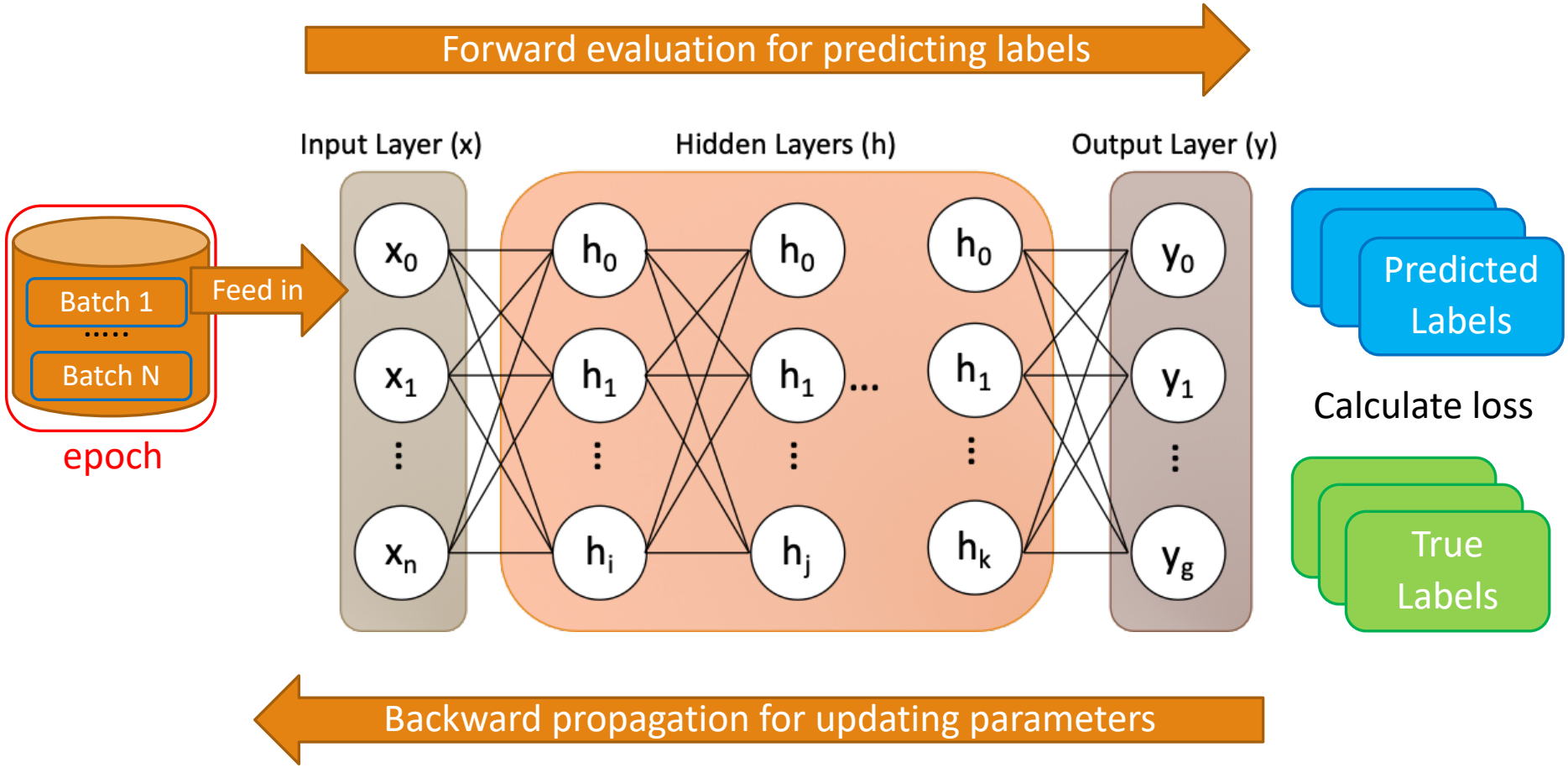
$$y1 = activation\ function(x1 * w11 + x2 * w21 + x3 * w31 + b1)$$
$$y2 = activation\ function(x1 * w12 + x2 * w22 + x3 * w32 + b2)$$

$$\begin{bmatrix} y_1 & y_2 \end{bmatrix} = activation\left(\begin{bmatrix} x_1 & x_2 & x_3 \end{bmatrix} \times \begin{bmatrix} w_{11} & w_{12} \\ w_{21} & w_{22} \\ w_{31} & w_{32} \end{bmatrix} \right) + \begin{bmatrix} b_1 & b_2 \end{bmatrix}$$

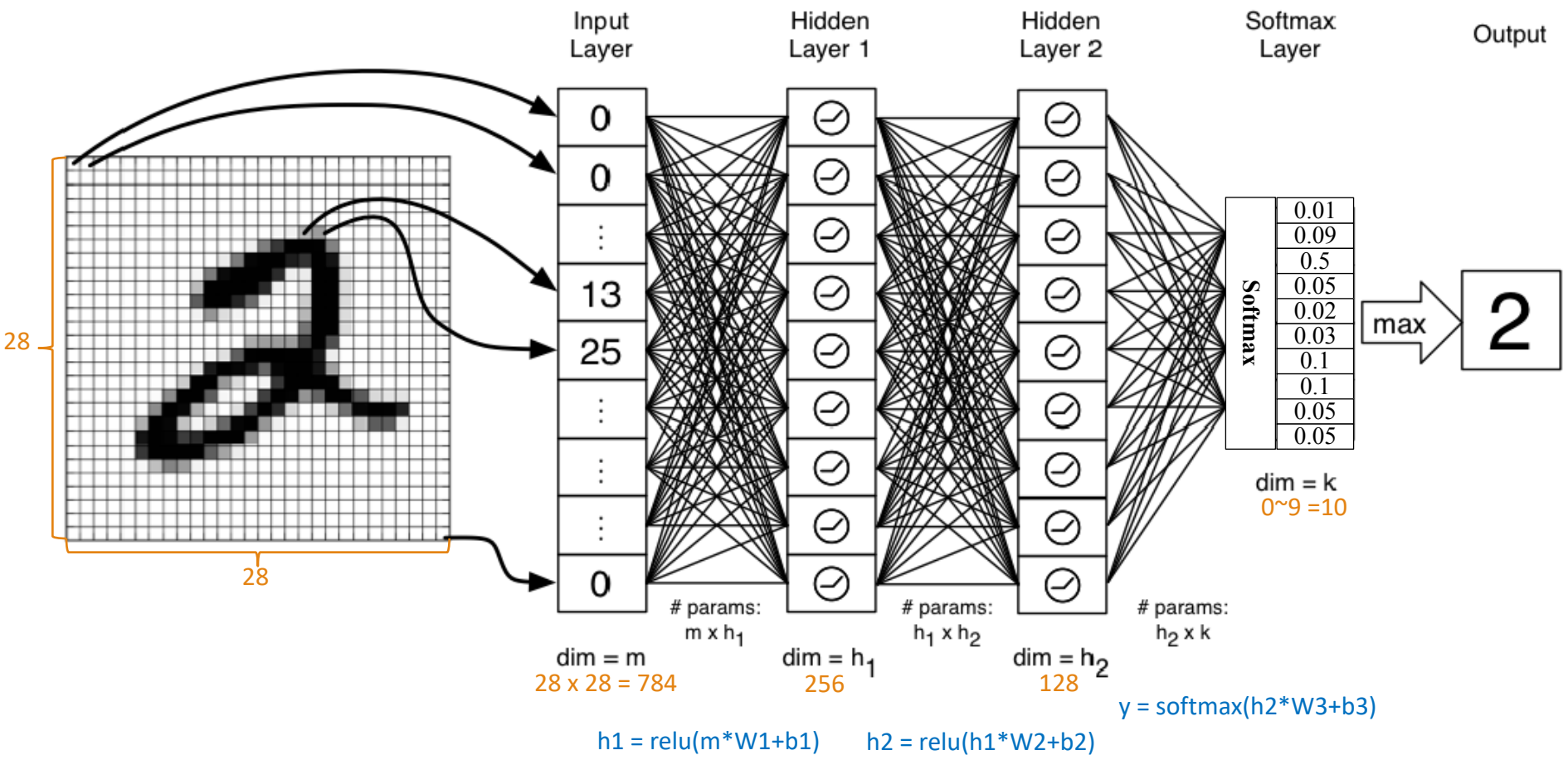
Y=activation(X*W+b)

output = activation function (input * weighting + bias)

Network Structure



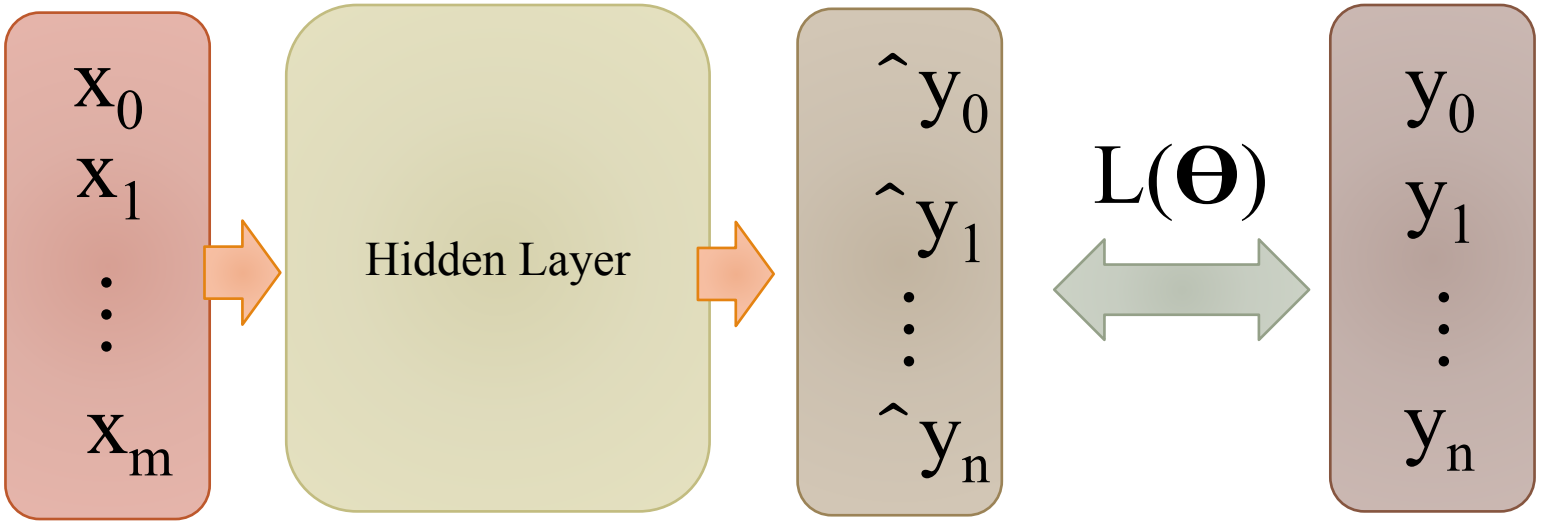
Example of Handwritten Recognition



How to evaluate model?

A loss function is to quantify the gap between network outputs and actual values.

Loss function is a function of **ϴ.**



Optimize the Total Loss

Find **the best function** that minimize total loss.

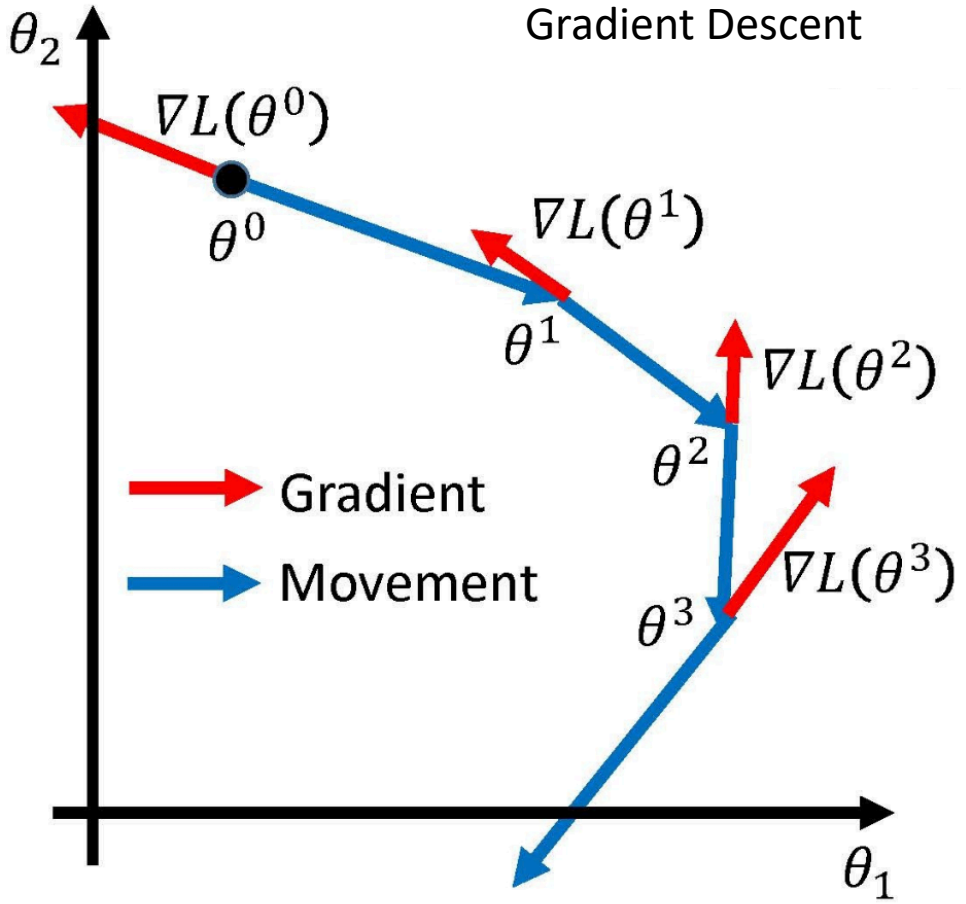
- Find the best network weights that can achieve minimize total loss ɵ*.

$$\theta^* = \underset{\theta}{argmin}\ L(\theta)$$

L: loss function; θ: parameters

- Suppose that θ has two variables $\{\theta_0, \theta_1\}$
- Have some function $L(\theta_0, \theta_1)$
  - Want $\min_{\theta_0,\theta_1} L(\theta_0, \theta_1)$
  - Start with some θ_0, θ_1
  - Keep changing θ_0, θ_1 to reduce $L(\theta_0, \theta_1)$
  - until we hopefully end up at a minimum

Optimize the Total Loss (cont.)



Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

⋮

Source: http://speech.ee.ntu.edu.tw/~tlkagk/research.html

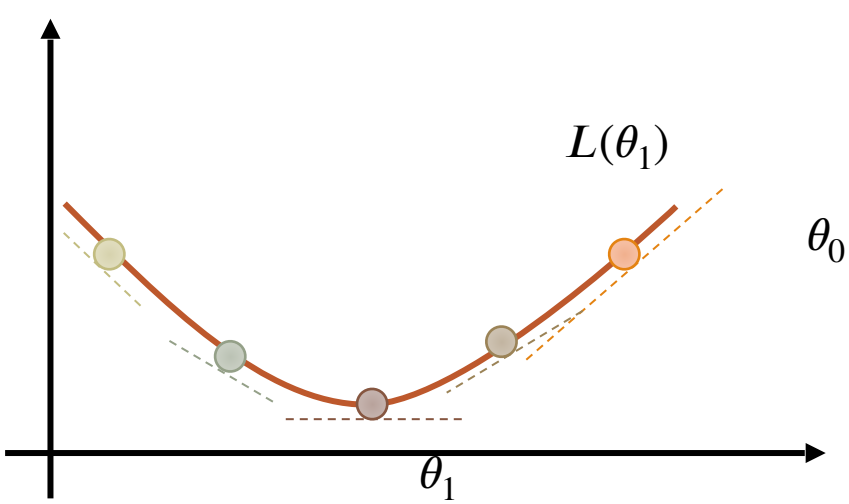
Optimize the Total Loss (cont.)

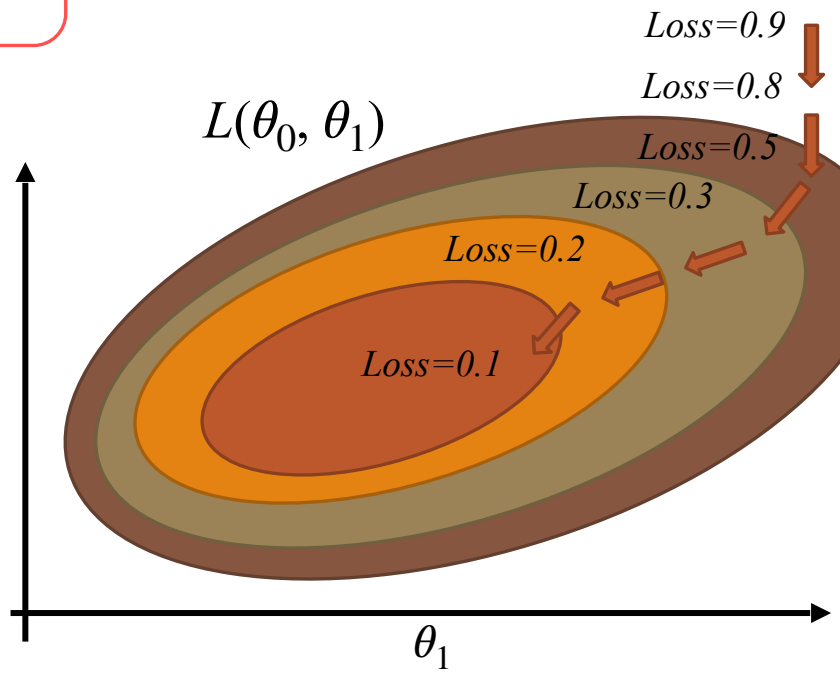
along to the opposite direction of the gradient

$$\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} L(\theta_0, \theta_1)$$

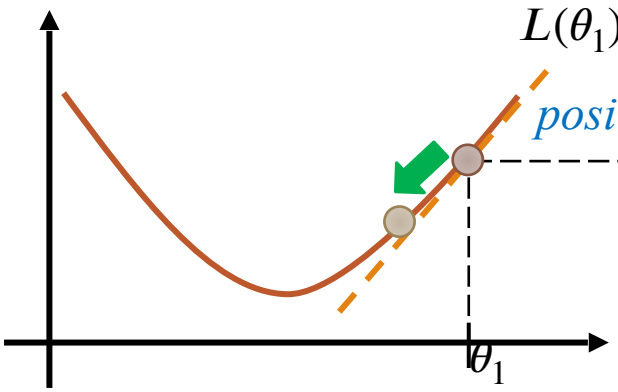
(for j=0 and j=1)

learning rate



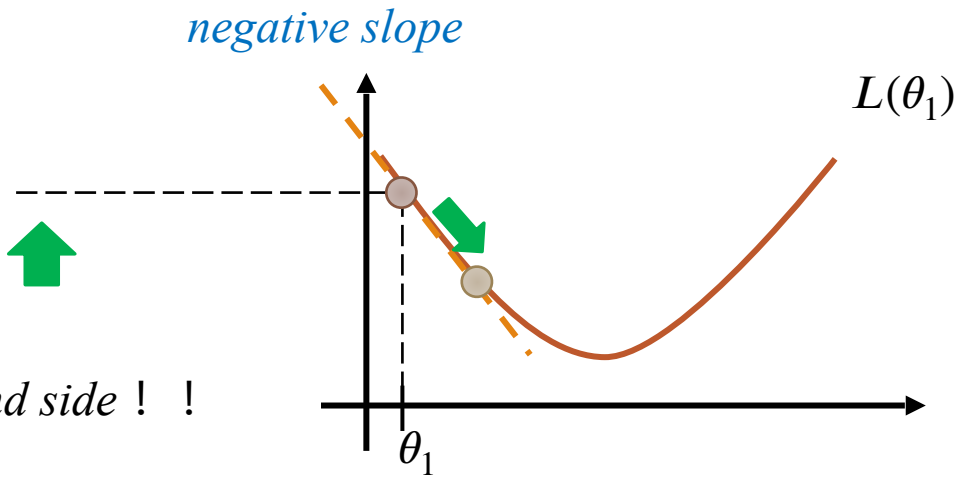


Optimize the Total Loss (cont.)



$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} L(\theta_1)}_{\text{slope of } \theta_1}$$

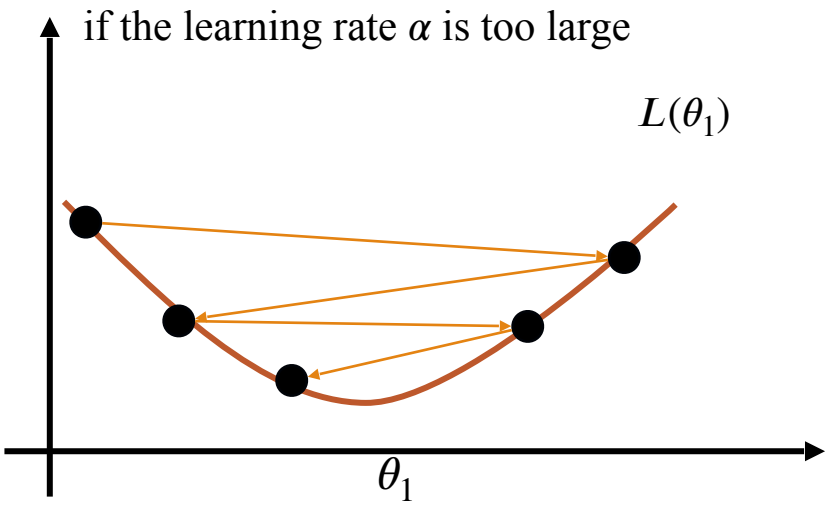
Move toward the left hand side ! !

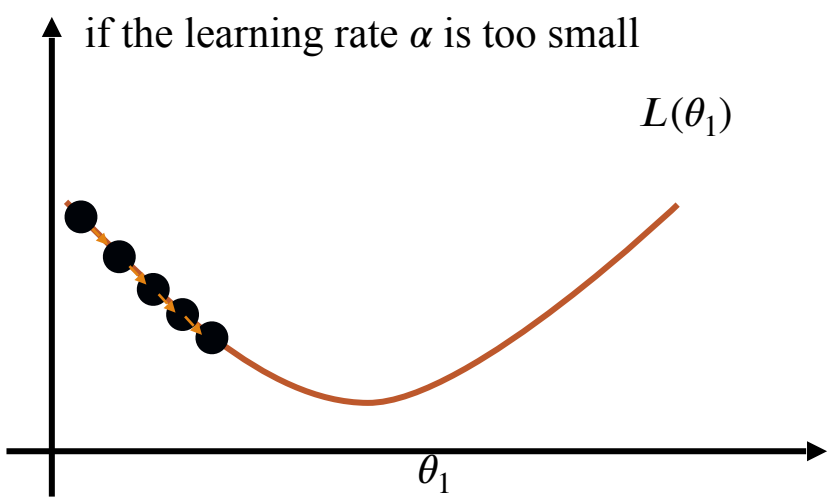


$$\theta_1 := \theta_1 - \alpha \underbrace{\frac{\partial}{\partial \theta_1} L(\theta_1)}_{\text{slope of } \theta_1}$$

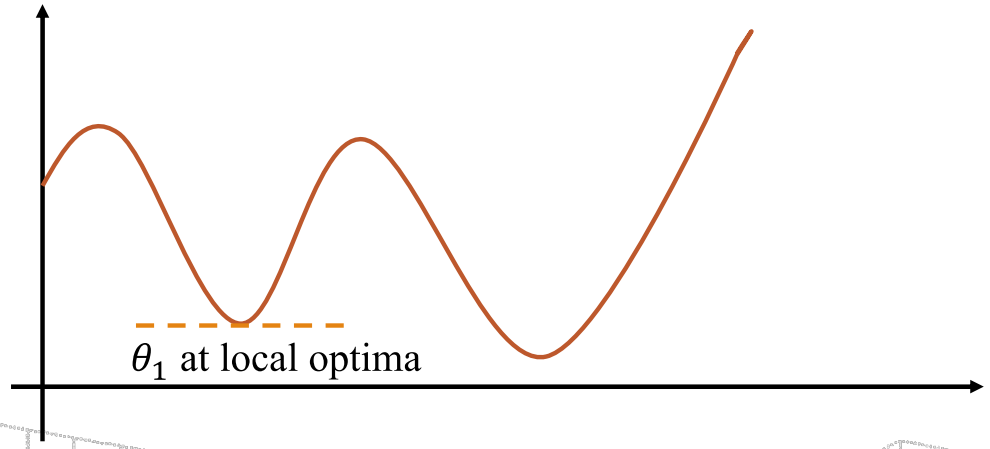
Move toward the right hand side ! !

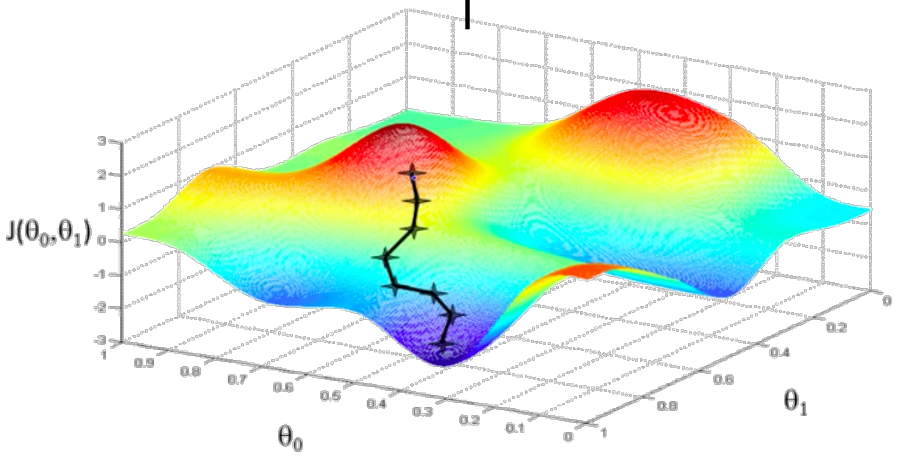
Optimize the Total Loss (cont.)

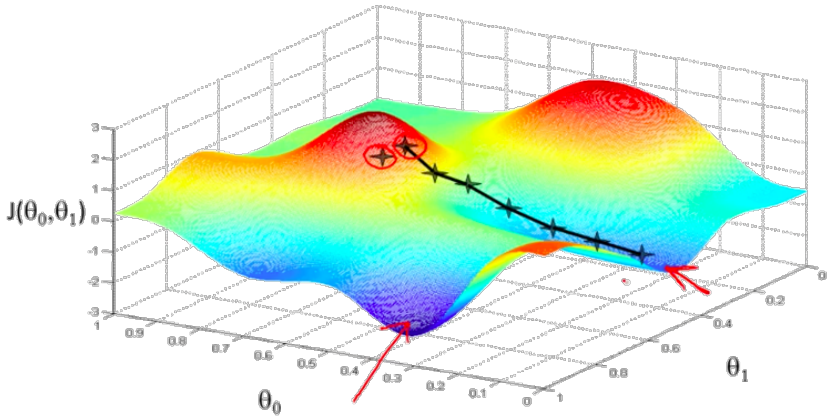




Optimize the Total Loss (cont.)







Stochastic Gradient Descent

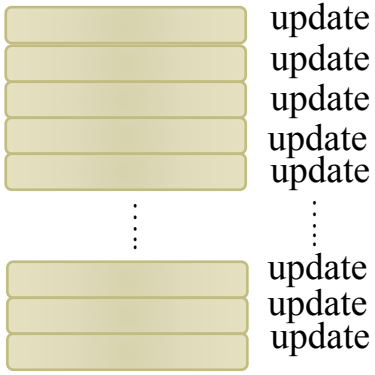
Since one epoch (whole training dataset) update one time, the convergence of Gradient Descent is slow.

Stochastic Gradient Descent (cont.)

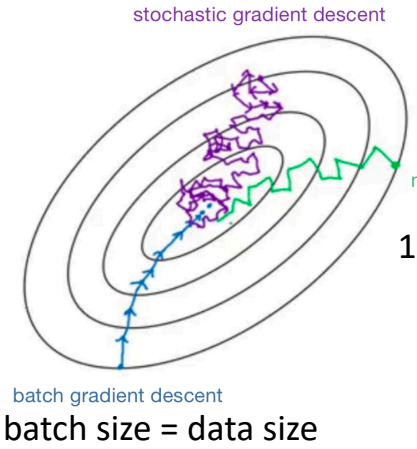
Faster to complete one epoch, and also faster to converge

One epoch update one time

One mini-batch update one time



batch size = 1



1 < batch size < data size

batch size = data size

The batch size is usually set as 28, 32, 128, 256

Loss Function

Learning from errors

Cross Entropy Loss Function

$$-\sum Y_i'.\log(Y_i)$$

Prediction

0	1	2	3	4	5	6	7	8	9
0.1	0.1	0.1	0.1	0.1	0.05	0.05	0.3	0.05	0.05

probability

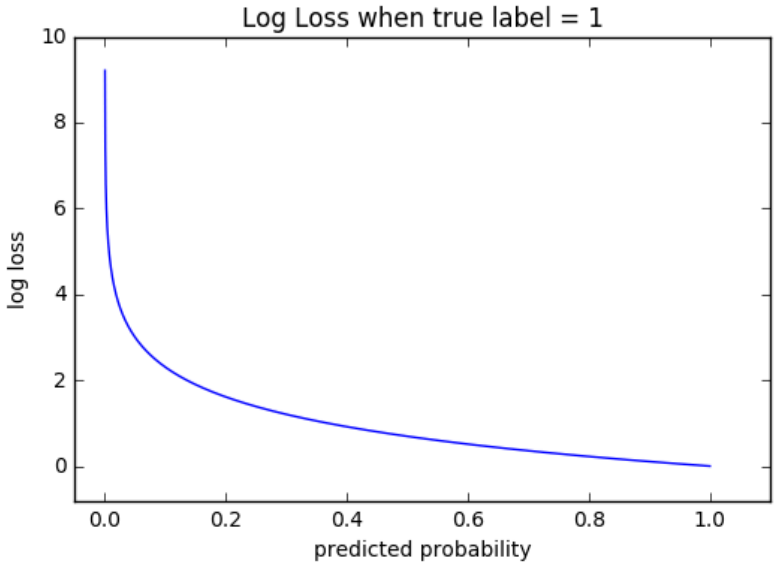
Label

0	1	2	3	4	5	6	7	8	9
0	0	0	0	0	0	0	1	0	0

One-hot Encoding

Cross-entropy loss

Cross-entropy loss measures the performance of a classification model whose output is a probability value between 0 and 1. Cross-entropy loss increases as the predicted probability diverges from the actual label. So predicting a probability of .012 when the actual observation label is 1 would be bad and result in a high loss value. A perfect model would have a log loss of 0.



http://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

Install Tensorflow and Keras

Install TensorFlow

pip install tensorflow

```
1 !pip install tensorflow
```

```
Requirement already satisfied: tensorflow in /Use
rs/yung-chunchang/anaconda3/lib/python3.6/site-pa
ckages (1.8.0)
Requirement already satisfied: protobuf>=3.4.0 in
/Users/yung-chunchang/anaconda3/lib/python3.6/sit
e-packages (from tensorflow) (3.14.0)
Collecting tensorboard<1.9.0,>=1.8.0
  Using cached tensorboard-1.8.0-py3-none-any.whl
(3.1 MB)
```

Install keras

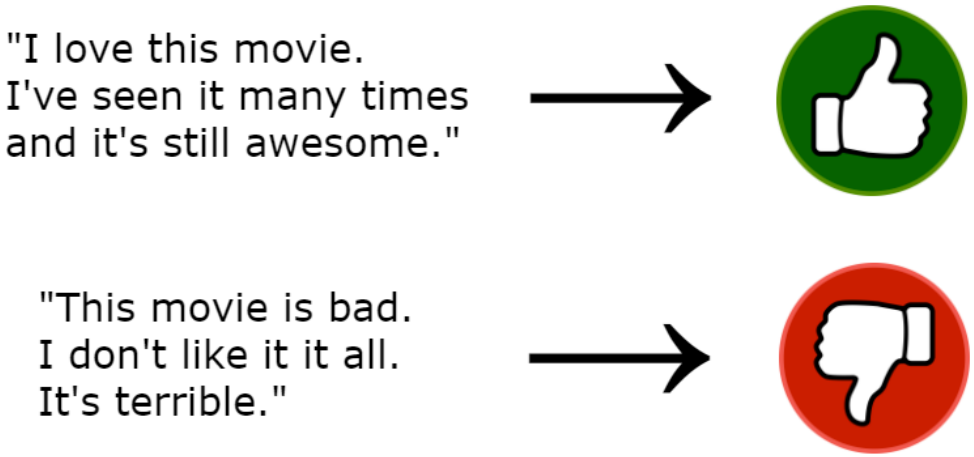
pip install keras (conda install keras)

```
1 !pip install keras
```

```
Collecting keras
  Downloading Keras-2.4.3-py2.py3-none-any.whl (3
6 kB)
Requirement already satisfied: h5py in /Users/yun
g-chunchang/anaconda3/lib/python3.6/site-packages
(from keras) (2.10.0)
Requirement already satisfied: numpy>=1.9.1 in /U
sers/yung-chunchang/anaconda3/lib/python3.6/site-
packages (from keras) (1.18.5)
Requirement already satisfied: scipy>=0.14 in /Us
ers/yung-chunchang/anaconda3/lib/python3.6/site-p
ackages (from keras) (1.3.0)
```

Sentiment Classification of IMDB Movie Review using Multilayer Perceptron

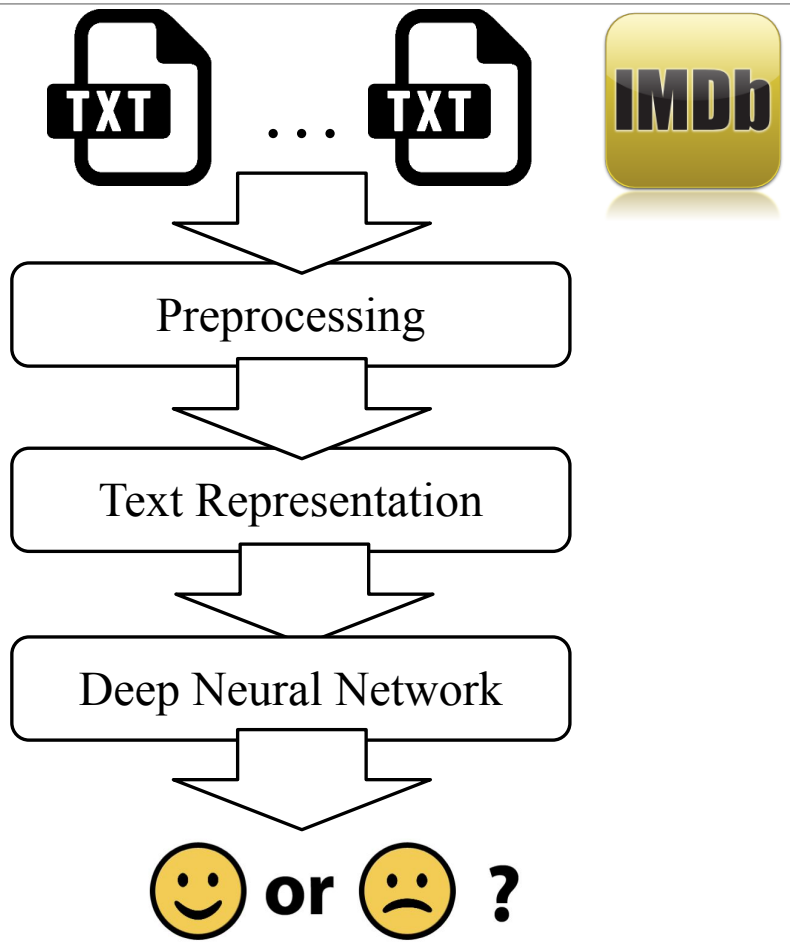
This dataset is a binary sentiment classification containing substantially more data than previous benchmark datasets. There are a set of 25,000 highly polar movie reviews for training, and 25,000 for testing.



Maas et al., Learning Word Vectors for Sentiment Analysis. ACL 2011.

http://ai.stanford.edu/~amaas/data/sentiment/aclImdb_v1.tar.gz

The workflow



Read Data

```python
import re, os
def rm_tags(text):
    re_tag = re.compile(r'<[^>]+>')
    return re_tag.sub('', text)

def read_files(filetype):
    path = "./dataset/aclImdb/"
    file_list = []

    positive_path = path + filetype + "/pos/"
    for f in os.listdir(positive_path):
        file_list += [positive_path + f]

    negative_path = path + filetype + "/neg/"
    for f in os.listdir(negative_path):
        file_list += [negative_path + f]

    print('read', filetype, 'files: ', len(file_list))

    all_labels = ([1] * 12500 + [0] * 12500)
    all_texts = []

    for fi in file_list:
        with open(fi, encoding = 'utf8') as file_input:
            all_texts += [rm_tags(" ".join(file_input.readlines()))]

    return all_labels, all_texts
```

Preprocessing
Remove html tag

Read Data (cont.)

```
1 y_train, x_train = read_files("train")
2 y_test, x_test = read_files("test")
3 print (x_train[0])
4 print (y_train[0])
```

```
read train files:  25000
read test files:  25000
For a movie that gets no respect there sure are a lot of memorable quotes
listed for this gem. Imagine a movie where Joe Piscopo is actually funny!
Maureen Stapleton is a scene stealer. The Moroni character is an absolute
scream. Watch for Alan "The Skipper" Hale jr. as a police Sgt.
1
```

Text Representation

Here we use sklearn's TfidfVectorizer to convert text to vector.

```
from sklearn.feature_extraction.text import TfidfVectorizer
tfidf_vector = TfidfVectorizer(token_pattern=r"(?u)\b\w+\b", max_features=100,
                               max_df = 0.9, min_df = 0.1, stop_words=["是", "的"])

vectors_training = tfidf_vector.fit_transform(x_train)
vectors_test = tfidf_vector.transform(x_test)
```

Naïve Bayes Classifier

```
1  from sklearn.naive_bayes import MultinomialNB
2  model = MultinomialNB(alpha=.01)
3  model.fit(vectors_training, y_train)
```

```
1  from sklearn import metrics
2  predicted = model.predict(vectors_test)
3  print("Macro-average: {0}".format(metrics.f1_score(y_test, predicted,
4  print("Micro-average: {0}".format(metrics.f1_score(y_test, predicted,
5  print(metrics.classification_report(y_test, predicted))
6  print(metrics.confusion_matrix(y_test, predicted))
```

```
Macro-average: 0.696749228047379
Micro-average: 0.69676
              precision    recall  f1-score   support

           0       0.69      0.70      0.70     12500
           1       0.70      0.69      0.69     12500

    accuracy                           0.70     25000
   macro avg       0.70      0.70      0.70     25000
weighted avg       0.70      0.70      0.70     25000

[[8784 3716]
 [3865 8635]]
```

Multilayer Perceptron

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras import utils

y_train_onehot = utils.to_categorical(y_train)

model = Sequential()
model.add(Dense(64, input_shape=(100,)))
model.add(Dropout(0.25))
model.add(Dense(32, activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 2, activation = 'softmax'))
model.summary()
model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
train_history = model.fit(vectors_training.toarray(), y_train_onehot, batch_size = 32, epochs = 10,
                          verbose = 2, validation_split = 0.2)
```

Multilayer Perceptron (cont.)

```
from sklearn import metrics
predicted = model.predict(vectors_test).argmax(axis=-1)
print("Macro-average: {0}".format(metrics.f1_score(y_test, predicted, average='macro')))
print("Micro-average: {0}".format(metrics.f1_score(y_test, predicted, average='micro')))
print(metrics.classification_report(y_test, predicted))
print(metrics.confusion_matrix(y_test, predicted))
```

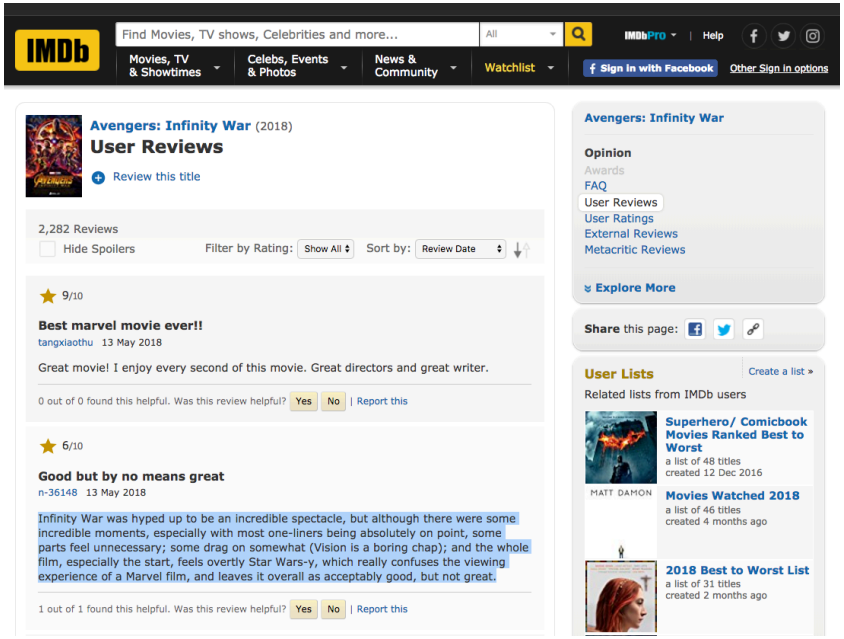
```
Macro-average: 0.7127440151455979
Micro-average: 0.7184
              precision    recall  f1-score   support

           0       0.80      0.58      0.67     12500
           1       0.67      0.86      0.75     12500

    accuracy                           0.72     25000
   macro avg       0.74      0.72      0.71     25000
weighted avg       0.74      0.72      0.71     25000

[[ 7226  5274]
 [ 1766 10734]]
```

Display document content

```
1  SentimentDict = {1: 'positive', 0: 'negative'}
2  def display_test_Sentiment(idx):
3      print(x_test_text[idx])
4      print('Label: ', SentimentDict[y_test[idx]],
5            'Prediction: ', SentimentDict[predicted[idx]])
6
7  display_test_Sentiment(2)
```

```
I really like this show. It has drama, romance, and comedy all rolled into one. I am 28 and I
am a married mother, so I can identify both with Lorelei's and Rory's experiences in the sho
w. I have been watching mostly the repeats on the Family Channel lately, so I am not up-to-da
te on what is going on now. I think females would like this show more than males, but I know
some men out there would enjoy it! I really like that is an hour long and not a half hour, as
th hour seems to fly by when I am watching it! Give it a chance if you have never seen the sh
ow! I think Lorelei and Luke are my favorite characters on the show though, mainly because of
the way they are with one another. How could you not see something was there (or take that lo
ng to see it I guess I should say)? Happy viewing!
Label:  positive Prediction:  positive
```

Predict a Movie Review




```
input_text = "Infinity War was hyped up to be an incredible spectacle, but although the
input_seq = tfidf_vector.transform([input_text])
predict_result = model.predict_classes(input_seq.toarray())
print(SentimentDict[predict_result[0]])
```

```
positive
```

MLP with Word2Vec

```
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.preprocessing.text import Tokenizer
token = Tokenizer(num_words = 2000)
token.fit_on_texts(train_text)

MAX_LEN = 150
x_train_seq = token.texts_to_sequences(train_text)
x_test_seq = token.texts_to_sequences(test_text)
x_train = sequence.pad_sequences(x_train_seq, maxlen = MAX_LEN, padding='pre', truncating='post')
x_test = sequence.pad_sequences(x_test_seq, maxlen = MAX_LEN, padding='pre', truncating='post')
```

MLP with Word2Vec (cont.)

```
print('Before pad_sequences length=', len(x_train_seq[0]))
print(x_train_seq[0])

print('After pad_sequences length=', len(x_train[0]))
print(x_train[0])
```

```
Before pad_sequences length= 41
[14, 3, 16, 11, 210, 53, 1157, 46, 248, 22, 3, 172, 4, 902, 14, 10, 1524, 833, 3, 16, 117,
912, 6, 161, 158, 6, 3, 132, 1, 105, 6, 31, 1551, 102, 14, 1604, 1, 1787, 13, 3, 564]
After pad_sequences length= 150
[   0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0    0    0    0
    0    0    0    0    0    0    0    0    0    0    0   14    3   16
   11  210   53 1157   46  248   22    3  172    4  902   14   10 1524
  833    3   16  117  912    6  161  158    6    3  132    1  105    6
   31 1551  102   14 1604    1 1787   13    3  564]
```

MLP with Word2Vec (cont.)

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation, Flatten
from tensorflow.keras.layers import Embedding
import numpy as np

model = Sequential()
model.add(Embedding(input_dim=2000, output_dim=100, input_length=MAX_LEN))
model.add(Dropout(0.25))
model.add(Flatten())
model.add(Dense(units = 64, activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 32, activation = 'relu'))
model.add(Dropout(0.25))
model.add(Dense(units = 1, activation = 'sigmoid'))
model.summary()
model.compile(loss = 'binary_crossentropy', optimizer = 'adam', metrics = ['accuracy'])
train_history = model.fit(np.array(x_train), y_train, batch_size = 100,
                          epochs = 10, verbose = 2, validation_split = 0.2)
```

MLP with Word2Vec (cont.)

```
1 from sklearn import metrics
2 predicted = model.predict(x_test).argmax(axis=-1)
3 print("Macro-average: {0}".format(metrics.f1_score(y_test, predicted, average='macro')))
4 print("Micro-average: {0}".format(metrics.f1_score(y_test, predicted, average='micro')))
5 print(metrics.classification_report(y_test, predicted))
6 print(metrics.confusion_matrix(y_test, predicted))
```

```
Macro-average: 0.8088526355626413
Micro-average: 0.80936
              precision    recall  f1-score   support

           0       0.84      0.76      0.80     12500
           1       0.78      0.86      0.82     12500

    accuracy                           0.81     25000
   macro avg       0.81      0.81      0.81     25000
weighted avg       0.81      0.81      0.81     25000

[[ 9473  3027]
 [ 1739 10761]]
```

Chinese Topic Detection in News Articles

The Dataset

```
1  sports  NBA／Wade出書談爸爸經　澄清和教練爭執 記者邵瑞峰／綜合報導熱火「閃電俠」Dwyane Wade近日推出了新書「A Father First
2  sports  美網／拒看過往輸球片段　小威：我從不折磨自己 記者洪偵源／綜合報導看自己以前打輸的影片，對Serena Williams來說非常痛苦
3  sports  MLB／雖然輸球　金鸞教頭稱讚陳偉殷投的非常好 記者方正東／綜合報導金鸞隊陳偉殷台北時間2日對洋基先發，投6.2局失4分、3分責
4  sports  MLB／期末考不合格　王建民復健賽只投2局丟6分 記者李玟／綜合報導國民隊台灣投手王建民在今（2日）於AA小聯盟Harrisburg S
5  sports  MLB　／遊騎兵好日本　建山義紀昇格上大聯盟 記者吳政紘／綜合報導美國職棒德州遊騎兵隊把日籍投手建山義紀叫上大聯盟，由於大聯
6  sports  MLB／不怪牛棚沒守住　陳偉殷懊惱自己第7局沒投好 記者李玟／綜合報導金鸞隊台灣左投陳偉殷在今（2日）凌晨出戰洋基，殷仔在7局
7  sports  NBA／馬刺波帥：Leonard會成為馬刺看板球星 記者邵瑞峰／綜合報導馬刺主帥Gregg Popovich毫無疑問是NBA近代最偉大的教練之
8  sports  美網／輕取西班牙好手　費爸前進十六強 記者洪偵源／綜合報導在自己曾經5度封王的戰場上，瑞士球王Roger Federer十六之路沒有
9  sports  MLB　／小李飛刀本季首度連勝　勇士遭刺傷吞敗 記者吳政紘／綜合報導美國職棒費城人隊今天(2日)派出「小李飛刀」Cliff Lee出戰
10 sports  旅外球星／陳品捷連3場獲保送　林旺衛挨觸身球 記者李玟／綜合報導台北時間2日於美國小聯盟出賽的台灣打者只有雙城隊林旺衛和小
```

Chinese Segmentation

CKIP

http://ckipsvr.iis.sinica.edu.tw/

Stanford Word Segmenter

http://nlp.stanford.edu/software/segmenter.shtml

MONPA

http://monpa.iis.sinica.edu.tw:9000/chunk

Jieba

https://github.com/fxsjy/jieba

Read Dataset

```
1  import jieba
2  from sklearn import preprocessing
3
4  def get_dataset(filepath):
5      labels = []
6      texts = []
7      jieba.set_dictionary('./dataset/dict.txt.big')
8      with open(filepath, 'r', encoding='UTF-8') as f :
9          for line in f.read().splitlines() :
10             items = line.split('\t')
11             if(len(items) == 2):
12                 labels.append(items[0])
13                 seg_content = ' '.join(jieba.cut(items[1], cut_all = False))
14                 texts.append(seg_content)
15
16     return get_encoded_label(labels), texts
17
18 def get_encoded_label(labels):
19     le = preprocessing.LabelEncoder()
20     le.fit(["sports", "politics", "health"]);
21     return le.transform(labels)
22
23
24 labels, texts = get_dataset("./dataset/CTD.txt")
```

Performance Evaluation – Naïve Bayes

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import KFold
from sklearn.naive_bayes import MultinomialNB
from sklearn import metrics

tfidf_BOW = TfidfVectorizer(token_pattern=r"(?u)\b\w+\b",
                            max_df = 0.9, min_df = 0.1, stop_words=["是", "的"])

predicted = []
expected = []
for train_idx, test_idx in KFold(n_splits = 10, shuffle = True).split(texts):
    x_train = np.array(texts)[train_idx]
    y_train = np.array(labels)[train_idx]

    x_test = np.array(texts)[test_idx]
    y_test = np.array(labels)[test_idx]

    vectors_training = tfidf_BOW.fit_transform(x_train)
    vectors_test = tfidf_BOW.transform(x_test)

    model = MultinomialNB(alpha=.01)
    model.fit(vectors_training, y_train)

    expected.extend(y_test)
    predicted.extend(model.predict(vectors_test))
```

Performance Evaluation – Naïve Bayes (cont.)

```
1 print("Macro-average: {0}".format(metrics.f1_score(expected, predicted, average='macro')))
2 print("Micro-average: {0}".format(metrics.f1_score(expected, predicted, average='micro')))
3 print(metrics.classification_report(expected, predicted))
4 print(metrics.confusion_matrix(expected, predicted))
```

```
Macro-average: 0.9445530271257718
Micro-average: 0.9446112779446113
              precision    recall  f1-score   support

           0       0.94      0.92      0.93       999
           1       0.93      0.95      0.94       997
           2       0.97      0.97      0.97      1001

    accuracy                           0.94      2997
   macro avg       0.94      0.94      0.94      2997
weighted avg       0.94      0.94      0.94      2997

[[919  56  24]
 [ 42 944  11]
 [ 18  15 968]]
```

Performance Evaluation – MLP

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import KFold
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense, Dropout, Activation
from tensorflow.keras import utils

tfidf_BOW = TfidfVectorizer(token_pattern=r"(?u)\b\w+\b", max_features=100,
                            max_df = 0.9, min_df = 0.1, stop_words=["是", "的"])


predicted = []
expected = []
for train_idx, test_idx in KFold(n_splits = 10, shuffle = True).split(texts):
    x_train = np.array(texts)[train_idx]
    y_train = np.array(labels)[train_idx]

    x_test = np.array(texts)[test_idx]
    y_test = np.array(labels)[test_idx]

    y_train_onehot = utils.to_categorical(y_train)
    y_test_onehot = utils.to_categorical(y_test)

    vectors_training = tfidf_BOW.fit_transform(x_train)
    vectors_test = tfidf_BOW.transform(x_test)

    model = Sequential()
    model.add(Dense(64, input_shape=(100,)))
    model.add(Dropout(0.25))
    model.add(Dense(32, activation = 'relu'))
    model.add(Dropout(0.25))
    model.add(Dense(units = 3, activation = 'softmax'))
    model.compile(loss = 'categorical_crossentropy', optimizer = 'adam', metrics = ['accuracy'])

    train_history = model.fit(vectors_training.toarray(), y_train_onehot, batch_size = 32,
                              epochs = 10, verbose = 1, validation_split = 0.2)

    expected.extend(y_test)
    predicted.extend(model.predict(vectors_test.toarray()).argmax(axis=-1))
```

Performance Evaluation – MLP (cont.)

```
1 from sklearn import metrics
2 print("Macro-average: {0}".format(metrics.f1_score(expected, predicted, average='macro')))
3 print("Micro-average: {0}".format(metrics.f1_score(expected, predicted, average='micro')))
4 print(metrics.classification_report(expected, predicted))
5 print(metrics.confusion_matrix(expected, predicted))
```

```
Macro-average: 0.9149111987192868
Micro-average: 0.915915915915916
              precision    recall  f1-score   support

           0       0.97      0.81      0.88       999
           1       0.86      0.96      0.91       997
           2       0.93      0.97      0.95      1001

    accuracy                           0.92      2997
   macro avg       0.92      0.92      0.91      2997
weighted avg       0.92      0.92      0.91      2997

[[812 133  54]
 [ 19 961  17]
 [  7  22 972]]
```