bias เพิ่มความยืดหยุ่น/ปรับ w ดีกว่าในบางกรณี

$net_{h_i} = (w_i \cdot i_i) + (w_i \cdot i_i) + b_i$

$out_{h_i} = \frac{1}{1+e^{-net_{h_i}}}$

$err = 1/2$ (ค่าจริง − ค่าทำนาย)^2

$E_{total} = \Sigma err$

learning rate

$w_i^+ = w_i - \eta \cdot \frac{\partial E_{total}}{\partial w_i}$

sigmoid

$$\frac{\partial E_{total}}{\partial w_i} = \frac{\partial E_{total}}{\partial out_{o_i}} \cdot \frac{\partial out_{o_i}}{\partial net_{o_i}} \cdot \frac{\partial net_{o_i}}{\partial w_i}$$

w_i = w ของ edge
o_i = output node
h_i = hidden node

layer ถัดไป
เพราะเป็น err
$\frac{\partial E_{O_i}}{\partial out_{h_i}}$

$\frac{\partial E_{oi}}{\partial out_{oi}}$ แต่ค่าคูณ w แล้ว

$\frac{\partial E_{total}}{\partial out_{o_i}} = -$(ค่าจริง − out_{o_i})

$\frac{\partial out_{o_i}}{\partial net_{o_i}} = out_{o_i}(1 - out_{o_i})$

ของ node

$\frac{\partial net_{o_i}}{\partial w_i} = out_{h_i}$ → w ของ edge นั้นๆ (layer ตัวเอง)

Out put

$\frac{\partial E_{total}}{\partial out_{O_i}} = \Sigma \left(\frac{\partial E_{total}}{\partial out_{O_i}} \cdot \frac{\partial out_{O_i}}{\partial net_{O_i}} \right) \cdot w_i$

$\frac{\partial out_{O_i}}{\partial net_{O_i}}$ = เหมือนเดิม

จำนวน neuron node

$\frac{\partial net_{O_i}}{\partial w_i}$ = เหมือนเดิม (อาจใช้ i_i แทน $out\ h_i$) (layer ก่อนหน้า)

hidden

param
MLP = $w * h * k + k$
CNN = filter size$^2 + 1$

kernel

CNN Convolution สกัด feature ด้วย filter และ + bias ตอนจบ
hyperparameter 1. ขนาด filter | 3. stride จำนวน shift (เลื่อน = ขยับ)
2. จำนวน filter | 4. padding จำนวนขอบที่เติม (เติม 0)

pooling
max ← นิยม
avg
ไม่มี param
hyper 1. pooling size
2. stride
3. type

ขนาด output convolution

$n_{out} = \left[\frac{n_{in} + 2p - k}{s} \right] + 1$ | p = padding size, k = filter size, s = stride size

$n_{out} = 10$ จะได้ $10 \times 10 \times$ channel

Avg precision $n-1$ และ $k+1$ เกิน

$AP = \sum_{k=0}^{k=n-1} [r(k) - r(k+1)] \cdot p(k)$

Precision recall plot
plot ปกติ precision เป็น y recall เป็น x

precision (มากไปน้อย) | recall (น้อยไปมาก)

อย่าลืม smooth กราฟ

class เดียว → AP = Σ(Pขวา − Pซ้าย) × สูงขวา

Binary class	sigmoid	Binary cross entropy	$-t_1 \log(f(s_1)) - (1-t_1)\log(1-f(s-1))$
Multi class	softmax	Categorical cross~	$CE = -\sum_i^c t_i \log(f(s)_i)$

Mean avg Precision

$mAP = \frac{1}{k} \sum_i^k AP_i$ | k = จำนวน class

Yolo detect location ด้วย bounding box แล้ว label ด้วย CNN

① Traditional (sliding window) pyramid search issue random CNN

② Region base CNN (RCNN 2014): Random bounding box มา cnn 2 ขั้นตอน

③ Yolo (You only look once 2016): speed, high acc, ez 2 use

ใช้ CNN สร้าง vector (img = input / vector = output)

vector: $\begin{bmatrix} 0,1 \\ float \\ float \\ float \\ float \\ 0 \\ 1 \end{bmatrix}$ ← มีวัตถุ? 1:0
x
y
h
w
} label one hot

0 = random ทิ้งแล้ว

1 รูปมี k·k cell
Anchor box: กำหนดจำนวนเงา
object ต่อ 1 cell
→ hyparameter
IOU (∩ over ∪)

$IOU(A,B) = \frac{|A \cap B|}{|A \cup B|}$

เก็บ box ที่ confident

NMS (Non Maximum Suppression) ลบ กล่องที่ IOU ถึง threshold

performance

train test (vali)

accuracy = เดาถูก/จำนวน sample

overfit = high acc / low vali acc

underfit = low both

ลด overfit
1. เลือก model err น้อย
2. L1/L2 regularization
3. Drop out
4. Batch Normalization
5. early stopping
6. noisy dataset

ทดสอบว่า model นี้ err เท่าไร
① hold out method
split train/test
ignore valid btw
② Random sub-samling
ทำ hold out หลายรอบ
③ K Cross-validation
แบ่งเป็น k กลุ่ม มี 1 กลุ่มที่เป็น test และวน for จนครบ

		label 1	label 0
predict	1	TP	FP
	0	FN	TN

$precision = \frac{tp}{tp+fp}$

$recall = \frac{tp}{tp+fn}$

$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall}$

True Position Rate $\frac{TP}{TP+FN}$

False Position Rate $\frac{FP}{FP+TN}$

Unsupervised Learning

no label req/for clustering

Measure

① Distance Metric: euclidean distance $= \sqrt{(x_2-x_1)^2 + (y_2-y_1)^2}$ no sqrt is ok

② Cosine Similarity: dot product $= \frac{\sum_{i=0}^{n} x_i y_i}{\sqrt{\sum_{i=0}^{n} x_i^2} \cdot \sqrt{\sum_{i=0}^{n} y_i^2}}$

plot Roc

1. Rank sample โดยอิงจาก probility กับ label (base on)
2. เริ่ม 0,0
3. เจอ +,1 เพิ่ม y
4. เจอ -,0 เพิ่ม x

k mean (k = จำนวน cluster)

1. cal dist between each centroid to each point
2. choose nearest centroid for each point
3. cal new centroid by avg x y

กิตติศักดิ์ [illegible]
6604062610225

cluster base rec sys

① collaborative filtering algor: user เหมือน
cold start | ใช้ user item metric

② content filtering algor: item คล้าย
item – itemfeature matric

cluster base cal step

① split cluster

② avg feature that u want

RNN

Many to one

```
def GenXY(dat, timstep):
    X, Y = [], []
    for i in range(len(dat) - timstep):
        X.append(dat[i:i+timstep])
        Y.append(dat[i+timstep])
    x = np.array(X)  Y = np.array(Y)
    return X, Y
```

ใช้

```
timstep = 10
trainX, trainY = GenXY(train, timstep)
```

Many to Many

```
trainX, trainY = GenXY(train, 5, 3)
```

```
def GenXY(dat, timstep, outlen):
    X, Y = [], []
    for i in range(len(dat) - timstep - outlen + 1):
        X.append(dat[i:i+timstep])
        Y.append(dat[i+timstep : i+timstep+outlen])
    X = np.array(X)  Y = np.array(Y)
    return X, Y
```

```
df = df.drop(columns = ['col1', 'col2'])
scaler = MinMaxScaler(feature_range=(0,1))
df = scaler.fit_tranform(df.value.reshape(-1,1))
n = int(len(df) * 0.8) | train = df[:n] | test = df[n:]
```

ใช้ gen | from tensorflow.keras.layer import Input, simpleRNN, GRU, LSTM, Dense | .keras.model import Sequential |

```
model = Sequential([
    Input(shape=(timstep, 1)),
    GRU(32, activation='tanh'),
    Dense(1, activation='sigmoid')
])
```

simpleRNN ↳ LSTM,

```
model.summary()
model.compile(loss='mean_square_error', optimizer='adam')
model.fit(xTrain, yTrain, epochs=100
```

CNN

```
from tenserflow.keras.preprocessing.image import ImageDataGenerator
trainPath = "./~"
train = ImageDataGenerator(
    rescale = 1./255, validation_split = 0.1)
trainGen = train.flow_from_directory(
    trainPath, target_size=(100,100), batch_size=32,
    class_mode='categorical', subset='training')
```

validation

```
from tensorflow.keras.utils import to_categorical
y = to_categorical(train_label)
from tensorflow.keras.model import Sequential
from tensorflow.keras.layers import Input, Conv2D, Maxpooling, Flatten, Dense
model = Sequential([
    Input(shape=(28,28,1)),
    Conv2D(30, (3,3), activation='relu',
    strides=(1,1), padding='same'),
    Flatten(), Dense(64, activation='relu'),
    Dense(10, activation='softmax')])
model.compile(optimizer= SGD(learning_rate=0.005),
loss='categorical_crossentropy', metrics=['accuracy']
```

```
hist = model.fit(train, Y, validation_split=0.1, epochs=50,)
acc = CNN.evaluate(test_img, Y)
```