[모두를 위한 딥러닝 시즌1. Sung Kim]

DL① (Sung Kim)

Lecture 1 ML Basic.

- limitation of explicit programming.
- Supervised learning : learning with labled example
- unsupervised learning : un-labeled data

MLlab1 Tensorflow.

→ open source software library for numerical computation using dataflow graphs.

- Graph ; Node & edges.
  - Node ↓ mathematical operations
  - edges ↓ tensors communicated between them
- TensorFlow mechanics
  - ① Build graph using TF operations
    - ↳ can use placeholders !! (& feeddict)
  - ② feed data and run graph (operation)
    - sess.run(op, feed_dict = {x : x_data})
  - ③ update variables in the graph (& return values)
- Tensor rank / shape / Types

rank	shape	Types
0	[ ]	Scalar
1	[D0]	1D vector
2	[D0, D1]	matrix
3	[D0, D1, D2]	3-tensor.
⋮		

Lecture 2,3,4 Linear regression.

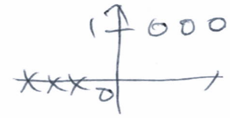
- $H(x) = Wx + b$ & Cost function $\frac{1}{m}\sum_{i}^{m}(y-H)^2$

$$\boxed{\begin{array}{l}\text{minimize cost}(W, b)\\ W, b\end{array}}$$

- gradient descent ← manual update / GradientDescentOptimizer
  - for convex cost fth.

Lecture 5 logistic classification?

- Regression $R \rightarrow R$
- Classification $R \rightarrow B^n$ $(B = \{0, 1\})$
  - ↳ Facebook feed, Credit card Fraudulent, Spam, Radiology, Finance,
- (0/1) encoding



- Logistic Hypothesis ⇒ $g(z) = \frac{1}{1+e^{-z}}$; sigmoid.
  $= \frac{1}{1+e^{W^T X}}$
- cost function? ⇒ cross-entropy
  - for sigmoid function?

$$C(H(x), y) = y\log(H(x)) - (1-y)\log(1-H(x))$$

- Grad. Desc. ⇒ $\Delta W = -\alpha \frac{\partial}{\partial W} cost(W)$

(Lab5) logistic Classifier !

```
hypothesis = tf.sigmoid(tf.matmul(X, W) + b)
cost       = tf.reduce_mean(Y * tf.log(hypothesis)
                 + (1-Y) * tf.log(1-hyp))
train = tf.train.gradient optimizer(learning rate = 0.01)  minimize(cost)
predicted = tf.cast(hyp > 0.5)
sess.run(tf.global_init~)
for step in range(~):
    cost_val, _ = sess.run([cost, train], feed_dict ~)
```

Lecture 6 : Softmax classification (multinomial)

logistic regression multinomial classification

$$\Rightarrow [w_{A1}\ w_{A2}\ w_{A3}]\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = z_A$$

$$\Rightarrow \begin{bmatrix} w_{A1} & w_{A2} & w_{A3} \\ & \cdots & \\ & \cdots & w_{C3}\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\end{bmatrix} = \begin{bmatrix}z_A\\z_B\\z_C\end{bmatrix} \xrightarrow{\boxed{S}} \begin{bmatrix}y_A\\y_B\\y_C\end{bmatrix}$$

- Softmax → $y \begin{bmatrix} 2.0 \\ 1.0 \\ 0.1 \end{bmatrix} \rightarrow \left[S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}} \right] \rightarrow \begin{bmatrix} 0.7 \\ 0.2 \\ 0.1 \end{bmatrix}$ probability

Scores

'one-hot' encoding $\begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}$

- Cost function = Cross entropy.

$S(y) \rightarrow \boxed{D(S, L) = -\sum_i L_i \log(S_i)}$

$S(y) = \bar{Y}$, Y

data label or target

predicted probability logit

$Y = L = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \Rightarrow B$

L(label) log(prediction) cost → cost

case1 $\bar{Y} = \begin{array}{c} A \\ B \end{array}\begin{bmatrix} 0 \\ 1 \end{bmatrix} \rightarrow B$; $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot \left(-\log \begin{bmatrix} 0 \\ 1 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ 0 \end{bmatrix} \rightarrow 0$

case2 $\bar{Y} = \begin{array}{c} A \\ B \end{array}\begin{bmatrix} 1 \\ 0 \end{bmatrix} \rightarrow A$; $\begin{bmatrix} 0 \\ 1 \end{bmatrix} \odot \left(-\log \begin{bmatrix} 1 \\ 0 \end{bmatrix}\right) = \begin{bmatrix} 0 \\ \infty \end{bmatrix} \rightarrow \infty$

- logistic cost ; $C(H(x), y) = y \log(H(x)) - (1-y)\log(1-H)$
- cross entropy ; $D(S, L) = -\sum_i L_i \log(S_i)$ (S: predict, L: label)

same

↳ total loss

$$\mathcal{L} = \frac{1}{N} \sum_i D_i(S_i, L_i) = -\frac{1}{N} \sum_i \sum_j L_{ij} \log(S_{ij})$$

(i: training data index)

Lab6 Softmax classifier.

- hypothesis = tf.matmul(X, W) + b
  ↓ (linear) logit
  tf.nn.softmax(tf.matmul(X,W)+b) — logit.

- loss function
  cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hyp), axis=1))
  (y_i 총합 / axis=0 ⇒ data index)
  opt = tf.train.GradientDescentOptimizer(learning_rate=0.1)
  .minimize(cost)

- Test & one-hot encoding
  hyp = tf.nn.softmax(tf.matmul(X,W)+b)
  a = sess.run(hyp, feed_dict X ...)
  print(a, sess.run(tf.arg_max(a,1)))

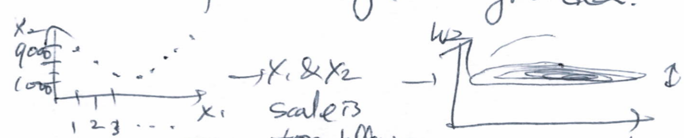
- Fancy Softmax Classifier
  logits = tf.matmul(X,W)+b , hyp = softmax
  cost = tf.reduce_mean(-tf.reduce_sum(Y * tf.log(hyp), axis=1)
  cost_i = tf.nn.softmax_cross_entropy_with_logits(logits=logits, labels=Y_one_hot)
  cost = tf.reduce_mean(cost_i)

- Discrete label into onehot vector!
  ① Y = tf.placeholder(tf.int32, [None, 1]) ← shape=(?,1)
  ② Y_one_hot = tf.one_hot(Y, nb_classes)
  ③ Y_one_hot = ~~Y_one_hot~~, → shape=(?,1,7) → =7 (label index size)
  tf.reshape(Y_one_hot, [-1, nb_classes])
  ① [1, 3, 0, 6]
  ② [[0100000] [0001000] [1000000] [0000001]] → shape (?, 7)
  ③ [0100000, 0001000, 1000000, 0000001]
  (? = # of training examples)

Lecture 7. Learning Rate (LR) // Data Preprocessing // Overfitting

- Gradient Descent
  ↳ overshooting problem : large LR.
  small learning rate ; takes too long, stops at local minimum
  ∴ try several LR.

- Data(X) Preprocessing for grad.desc.
  x_1 & x_2 scale is too different → too narrow



★ Standardization ; $x_j' = \frac{x_j - \mu_j}{\sigma_j}$

X_std[:,0] = (X[:,0] - X[:,0].mean()) / X[:,0].std()

ML③

- Overfitting Problem

solution:
- More training data
- Reduce the # of parameters
- Regularization
  - Let's not have too big numbers in the weight

$$\mathcal{L} = \frac{1}{N}\sum_i D(S(WX_i+b), L_i) + \lambda \sum W^2$$

λ: regularization strength

l2reg = 0.001 X tf.reduce_sum(tf.square(W))

- Performance Evaluation ?

data → training → model → $\bar{Y}$ vs Y → adjust & learning → finish

test → predict / classify

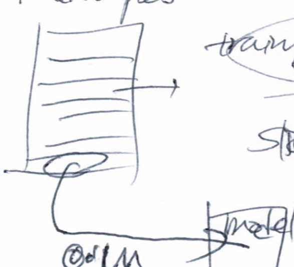
original set

Training		Testing
Training	Validation	testing

↓ 공부 | ↓ 모의고사 | ↓ 시험

- Online learning ?

100M examples



training at once → too expensive

⇓

Slice them into small batches

ex) MNIST → Accuracy ?!

Lab 7 Learning rate & Evaluation

- 6 x_data = [[⋯],[⋯],[⋯]⋯[⋯]]

  y_data = [[~],[~],[~]⋯[~]] (one hot)

  & x_test = ~

  y_test = ~

- learning rate → in tf.train.GradientDescent(learnrate) = XX.min(cost)

- non-normalized input ?

  xy = MinMaxScaler(xy)

- MNIST

[28x28x1] Image ⇒ X = tf.placeholder(tf.float32, [None, 784])

Y = tf.placeholder(tf.float32, [None, nb_classes]) (= 10)

```
from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('MNIST_data/', one_hot=True)
  :
batch_xs, batch_ys = mnist.train.next_batch(100)
  :
print("Accuracy: ", accuracy.eval(session=sess,
      feed_dict={X: mnist.test.images,
                 Y: mnist.test.labels}))
```

testing

```
is_correct = tf.equal(tf.arg_max(hyp, 1),
                      tf.arg_max(Y, 1))
accuracy = tf.reduce_mean(tf.cast(is_correct, tf.float32))
```

T/F → 1/0.0

training epochs = 15 → Full dataset execution training

batch size = 100 → Training unit for a single weight update.

```
with tf.Session() as sess:
    sess.run(tf ~ initialize)
    for epoch in range(training_epochs):
        avg_cost = 0
        total_batch = int(mnist.train.num_examples / batch_size)
        for i in range(total_batch):
            batch_xs, batch_ys = mnist.train.next_batch(batch_size)
            c, _ = sess.run([cost, opt], feed_dict={X: batch_xs, Y: batch_ys})
            avgcost += c / total_batch
```

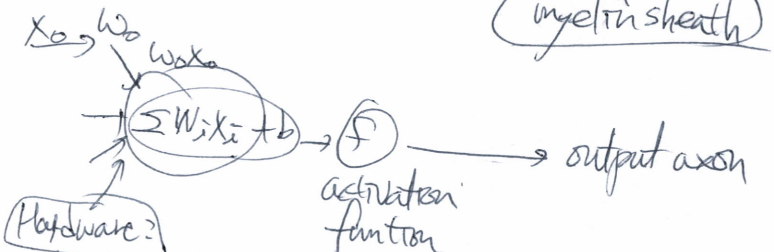
sess.run(node) = node.eval()

mnist visualization ⇒ plt.imshow(
mnist.test.images[r:r+1].
reshape(28,28), cmap='Greys',
interpolation='nearest')
plt.show()

Lecture 8 Deep Neural Network

neuron → dendrite — cell nucleus — axon — axon terminal

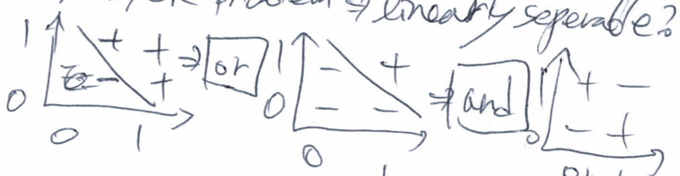
myelin sheath



→ Frank Rosenblatt ~1957: Perceptron
Widrow & Hoff ~1960: Adaline/Madaline

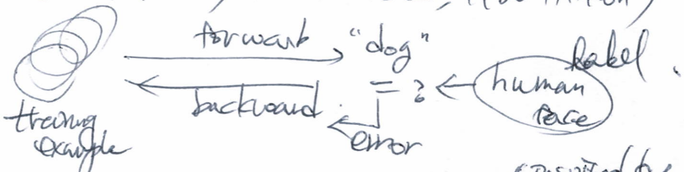
<False promises> of Rosenblatt
→ July 08, 1958...

Sample AND/OR problem ⇒ linearly separable?



- Perceptrons (1969)
  → Marvin Minsky.
  XOR → x
  not linearly separable.
  → Need use MLP (multilayer perceptron)
  → Not possible to train it...
  "No one on earth had found a viable way to train" 1969

- Backpropagation (1974, 1982 by Paul Werbos, 1986 Hinton)



- Convolutional Neural Network ⇒ local connection (1980, LeCun) (inspired by Hubel & Wiesel 1959)

- NavLab 1984-1994: 주행로..
- A Big Problem ★
  → Back Propagation → (Not work for many layers) Vanishing Gradient problem!
  Other existing algorithm!
  ⇒ SVM, Random Forest.

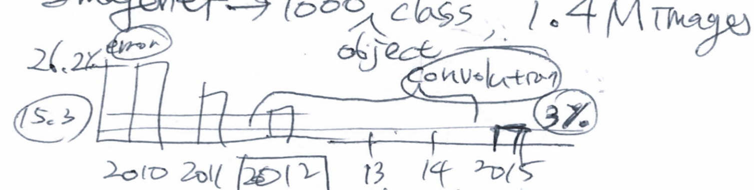
1995 by LeCun → New approach! agree

★ CIFAR → HINTON!

2006 Hinton "A fast learning algorithm for deep belief nets"
2007 Y. Bengio "Greedy layer-wise training of deep network"

- If the weight are initialized in a clever way rather than randomly, deep neural network can be trained well

- Imagenet → 1000 class, 1.4M images
  object



- Deep API learning (Dr. Sung Kim's Lab)..
  alphago, AI game

- G. Hinton's summary of finding up to today
  ① Our labeled datasets were thousands of times too small.
  ② Our computers were millions of times too slow
  ③ We initialized the weights in a stupid way
  ④ We used the wrong type of non-linearity.
  (from "a brief history of neural nets and deep learning part 4")

Why DNN? / youtube (translation), Facebook (user traing), Google (web search), netflix (recommendation), amazon
⋮

Lab 8. Tensor manipulation

1D t = np.array([0, 1, 2, ... 6])
- t.ndim
- t.shape
- t[0], t[1] ...
- t[:2], t[2:] ← slicing

2D t = np.array([[1,2,3], [4,5,6], [:], [ ]]) → t.ndim = 2 = rank, t.shape = (4,3)

Tensor
↳ • t = tf.costant([1,2,3,4])
tf.shape(t).eval() → array([4], dtype=int32)
• t = tf.costant([[1,2],[3,4]])
number of [ ] = rank
tf.shape(t).eval() ⇒ [2,2] (x, y)

axis? ⇒ rank = 4? ⇒ 4 axes!!
• inner-most axis = 3 = -1 → axis 3 [[[[0000], [0000]], [[0000], [0000]]], [...]]
• outer-most axis = 0 → axis 0

• matmul vs multiply
⇒ A: shape (2,2), B: shap (2,1) ⇒ tf.matmul(A,B).eval()
(2,2)×(2,1) ⇒ shape (2,1)
A * B ⇒ element-wise multiplication
But... if shape is not same → Broadcasting → Be careful!

• Reduce_mean.
⇒ tf.reduce_mean([1,2], axis=0).eval() ← int? → output int
• stack
• tf.ones_like(x). & tf.zeros_like(x)

[[1, 2], [3, 4]] → reduce_mean(x, axis=0) → $\frac{1}{2}$([1,2] + [3,4])
reduce_mean(x, axis=-1) → ($\frac{1}{2}$[1+2], $\frac{1}{2}$[3+4])
axis=0, axis=1

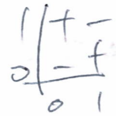
• Argmax [[0,1,2], [2,1,0]]
- tf.argmax(x, axis=0).eval() ↳ [1,0,0]
- tf.argmax(x, axis=-1).eval() ↳ [2,0]
argmax = "position of maximum along the axis"

• Reshape → [[[0,1,2], [3,4,5]], [[6,7,8], [9,10,11]]]
tf.reshape(t, shape[-1,3]).eval()
↳ [[0 1 2] [3 4 5] [6 7 8] [9 10 11]]
tf.reshape(t, shape[-1,1,3]).eval()
"remain the last two axes"

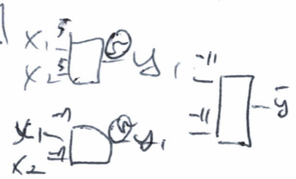
• tf.squeeze, expand
↳ = flatten in → tf.expand_dims ([0,1,2], 1).eval()
• zip (python) numpy
• one hot
tf.one_hot([[0][1][2][0]], depth=3).eval()
↳ [[[1 0 0]] [[0 1 0]] [[0 0 1]] [[1 0 0]]], dtype=float32)
& tf.reshape(t, shape=[-1,3]).eval()
(∵ redundant rank is generated after tf.onehot)
• cast
⇒ tf.cast(X, int or float)

Lecture 9. Deep learning 정리!

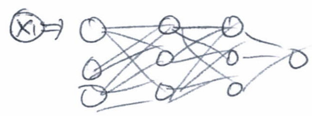
* XOR ?

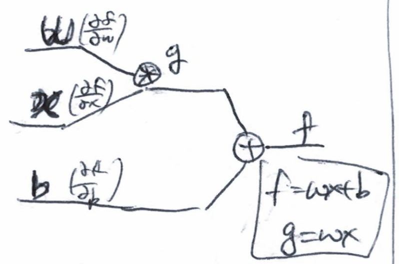


neural net



* Backpropagation (Chain rule)





single layer

$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w} = 1 * 5 = 5$

$\frac{\partial f}{\partial x} = \frac{\partial f}{\partial g}\frac{\partial g}{\partial w}$ ......

→ 즉.. f에 w 어떠한 변화 (forward)

$\frac{\partial f}{\partial g} \rightarrow \frac{\partial f}{\partial w}$ (Backward) 알아내기

∴ graph 로 알면 BP는 바로 풀어짐 !!

~~tensor board~~ 유용함★

LAB 9

```
w1 = tf.va~
b1 = tf.va~
layer1 = tf.sigmoid(tf.matmul(X, W1) + b1)
w2 = tf.v~
b2 = tf.va~
layer2 = tf.sigmoid(tf.matmul(layer1, W2) + b2)
```
hypothesis

★★ TensorBoard : TF logging / debugging tool

① From TF graph, decide which tensors you want to log.

- w2_hist = tf.summary.histogram("weights2", W2)
- cost_summ = tf.summary.scalar("cost", cost)

② Merge all summaries

- summary = tf.summary.merge_all()

③ Create writer and add graph

- writer = tf.summary.FileWriter("./logs") file
- writer.add_graph(sess.graph)

④ Run summary merge and add_summary (ML6)

- s, _ = sess.run([summary, optimizer], feed_dict=feed_dict)
- writer.add_summary(s, global_step=global_step)

⑤ Launch TensorBoard

- tensorboard --logdir=./logs

• What tensor ?

- Scalar tensor ⇒ tf.summary.scalar()
- Histogram for multiple-valued tensor

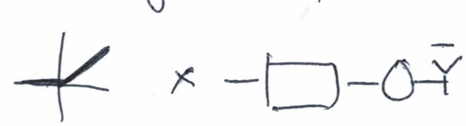
• Add scope for better graph hierarchy

```
with tf.name_scope("layer1") as scope:
    {
with tf.name_scope("layer2") as scope:
```

Lecture 10. ReLU

Sigmoid ; Vanishing Gradient problem!

ReLU



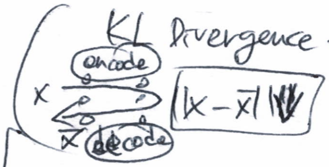
L1 = tf.nn.relu(tf.matmul(X, W1) + b1)

Leaky ReLU & Maxout & ELU

Initialization (smart way)

- Not all 0's
- Challenging issue
- Hinton et. al. (2006) DBN with RBM (restricted boltzman machine)

KL Divergence = compare actual input to recreated input



- No need to use RBM !
- simple method OK
  - Xavier initialization (2010) (Glorot)
  - He's initialization (2015)

Xavier: W = np.random.randn(fan_in, fan_out) / np.sqrt(fan_in)

He : W = np.randn (fan_in, fan_out) / np.sqrt(fan_in/2)

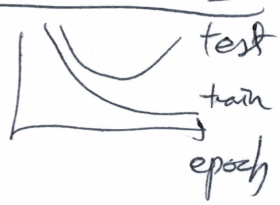
- prettytensor implementation.

Batch normalization

- Layer sequential uniform variance.

Dropout & Model Ensemble

overfitting.



→ Regularization.

↳ ① L2 Regularization

↳ ② Dropout (2014)

- Tensorflow implementation.

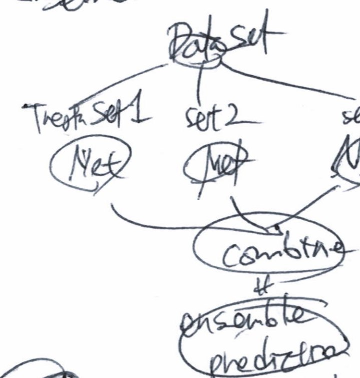
dropout_rate = tf.placeholder("float")

L_1 = tf.nn.relu(tf.add(tf.matmul(X, W1), B1))

L_1 = tf.nn.dropout(_L1, dropout_rate)

주의 evaluation 할 때 dropout rate = 1.0

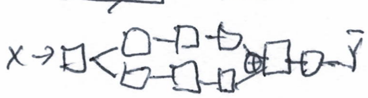
- Ensemble ?



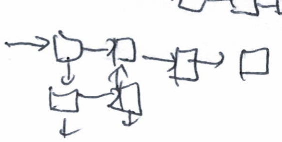
- Build DNN !

- Fast forward (He's ResNet)

- Split & Merge



- Recurrent



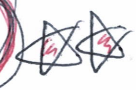
- Optimizers

↳ GradientDescent, AdadeltaOpt

Stoch. AdagradOpt. MomentumOpt, AdamOpt

FtrlOpt, ...

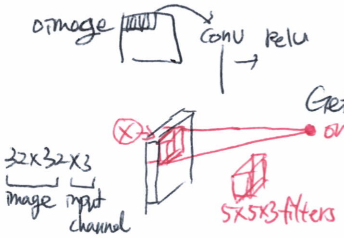
CNN



↳ Next page.

☆ CNN

- Hubel & Wiesel 1959 Receptive fields of single neurons in the cat's striate cortex.



→ How many numbers we get?

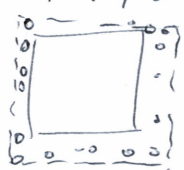
ex) 7×7 input (spatially), N; assume 3×3 filter (F) → (1 stride) → 5×5 output $(= 7 - [3-1])$

→ (2 stride) → 3×3 output $\left(\frac{7-3}{2}+1\right)$

Output size $= \boxed{\frac{N-F}{\text{stride}}} + 1$

In practice: Common to zero pad the board.

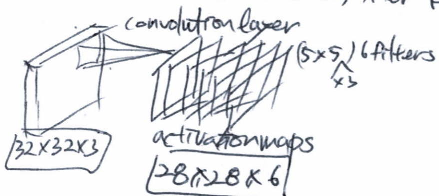
eg: input 7×7, 3×3 filter, stride 1 ⇒ pad with 1 pixel boarder



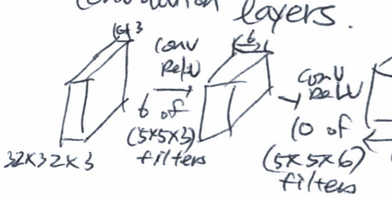
⇒ what is the output?

⇒ 7×7 output.

common — stride 1, filter F×F, zero padding $\left(\frac{F-1}{2}\right)$

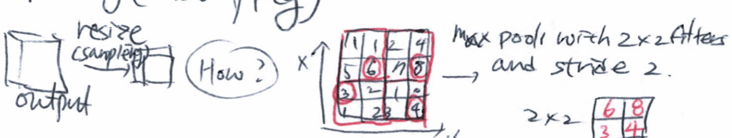


- Convolution layers.



Q. How many weight variables?
$(5\times5\times3)\times6 + (5\times5\times6)\times10$
$(F_1 \; F_2 \; 10) \times N_3$

- Pooling (= sampling)



- Fully connected layer (FC layer)

- ConvNetJS demo: training on CIFAR-10

(cs.stanford.edu/people/karpathy/convnetjs/demo/cifar10.html)

MLE

Case Study

LeNet-5 (LeCun, 1998)

[Input 32×32×1] → (6 filters w/ (5×5×1)) → [C. feature map 1 28×28×6] → (subsampling (maxpooling) = (2×2×6) & stride 2.) → [S. feature map 1 14×14×6]

→ (16 filters w/ (5×5×6)) → [C. feature map 2 10×10×16] → (subsampling (maxpool) = ~~2×2×6~~ stride 2) → [S. feature map 2 5×5×16] → 400

→ (fully connected 1 400×120) → [C5 layer 120] → (120×84) → [F6 layer 84] → (84×10) → Output

gaussian connection (or softmax)

AlexNet (ImageNet competition 1st place, Krizhevsky et al. 2012)

8 layer

1st conv: Input 227×227×3 → ① conv 96 ⓐ 11×11×3 w/ stride 4 (~35K) → Out ① 55×55×96 →

$\frac{227-11}{4}+1 = 55$ (216, 54)

2nd pool: (3×3 w/ stride 2) max pooling → zero parameters → 27×27×96 ....

$\frac{55-3}{2}+1 = 27$ (52, 26)

<summary>

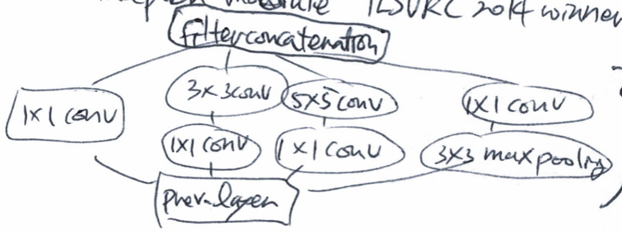
[227 × 227 × 3] Input
[55 × 55 × 96] conv1 : 96 @ 11×11 (×3) filter w/ s=4 & pad=0.
[27 × 27 × 96] Maxpool : 3×3 filter w/ s=2
[27 × 27 × 96] norm1 : Normalization layer
[27 × 27 × 256] conv2 : 256 @ 5×5 (×96) filter s=1 pad=2
[13 × 13 × 256] maxpool : 3×3 filter w/ s=2.
[13 × 13 × 256] Norm2
[13 × 13 × 384] conv3 : 384 @ 3×3 (×256) s=1, pad 1.
[13 × 13 × 384] conv4 : 384 @ 3×3 (×384) s=1, p=1
[13 × 13 × 256] conv5 : 256 @ 3×3 (×384) s=1, p=1
[6 × 6 × 256] Maxpool3 : 3×3 filter, s=2.
[4096] FC6
[4096] FC7
[1000] FC8 (class scores)

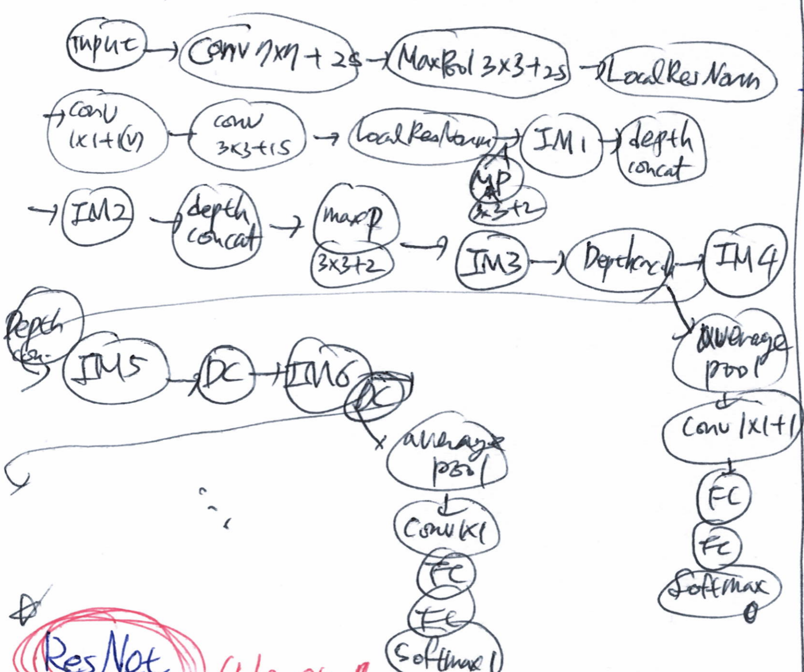
AlexNet continued....

⇒ 1st use of ReLU
- Norm layers (not common anymore)
- heavy data augmentation
- dropout = 0.5
- batchsize = 128
- SGD Momentum = 0.9
- learning rate = 1e-2. reduced by 10 manually when val accuracy plateaus.
- L2 weight decay = 5e-4
- 7 CNN ensemble : 18.2% → 15.4%

GoogLeNet Szegedy et. al. 2014

↳ Inception module ILSVRC 2014 winner 6.7% top5 error





ResNet (He et al., 2015)

→ ILSVRC2015 winner (3.6% top5 error)

1st place in 5 main tracks.

→ Imagenet classification (152 layers)
- detection
- localization
- coco detection
- coco segmentation

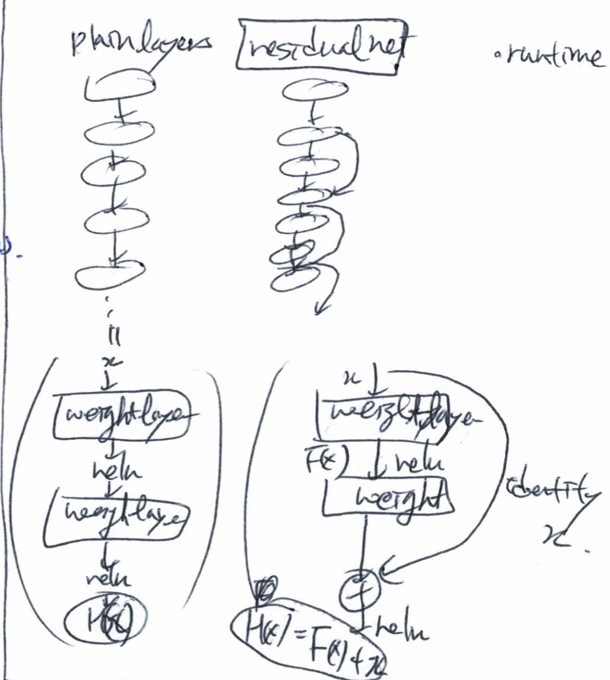
2017 = 2.3%

- Revolution of depth

AlexNet 8 layers → VGG 19 layer → ResNet 152 layer
2012 — 2014 — 2015

(2~3 week training GPU) 8

- runtime = faster than VGG Net



- CNN for sentence classification (Yoon Kim) 2014
- Alphago

<Tensorflow CNN>

input → convolution subsampling → FC → output

<= feature extraction> <classification>

image vector → filter w/stride → subsampling (pooling)

- simple convolution layer.

3x3x1 image
3x2x1 filter(W) → 2x2x1 output.

<Tensor shape>
- Image (#data index (??), x, y, color) "-1" is good.
- filter → (x_f, y_f, color, #filters)
- stride → (s_x, s_y)
- padding → "VALID"

→ tf.nn.conv2d(image, weight, strides=[1,1,1,1], padding='VALID')

→ padding = SAME → output map size = input map size

; pooling

↳ tf.nn.max_pool(image, ksize=[1,2,2,1], stride=[1,1,1,1], padding='SAME')

⓪ MNIST Conv layer

```
sess = tf.InteractiveSession()
img = img.reshape(-1, 28, 28, 1)                  # filters
W1 = tf.Variable(tf.random_normal([3,3,1,5],
                                  stddev=0.01))
conv2d = tf.nn.conv2d(img, W1, strides=[1,2,2,1],
                      padding='SAME')
sess.run(tf.global_initial~())
conv2d_img = conv2d.eval()
conv2d_img = np.swapaxes(conv2d_img, 0, 3)
for i, one_img in enumerate(conv2d_img):
    plt.subplot(1,5,i+1), plt.imshow(one_img.
                  reshape(14,14),
                  cmap='grey')
```

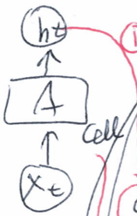
① MNIST Maxpooling.

```
pool = tf.nn.max_pool(conv2d, ksize=[1,2,2,1],
                      strides=[1,2,2,1], padding
                      ='same')
print(pool)

sess.run(tf.global~)
pool_img = pool.eval()
pool_img = np.swapaxes(pool_img, 0, 3)
for ~.
```

RNN ~ next page.

<RNN in Tensorflow>



① 사용생성

Cell = tf. contrib. rnn. BasicRNNCell
(num_units = hidden_size)

⋮

outputs, _states = ② tf. nn. dynamic_rnn (
cell, x_data, dtype = tf. float 32)

cell = tf. contrib. rnn. BasicLSTMCell (~)

(ex) One node : 4 (input_dim) in 2 (hidden_size)

h = [1 0 0 0]
e = [0 1 0 0]
l = [0 0 1 0]
o = [0 0 0 1]

(X) → [[[1, 0, 0, 0]]]
shape = (1, 1, 4)

(h)
hidden size = 2.
shape (output) = (1, 1, 2)
[[[x, x]]]

(code)

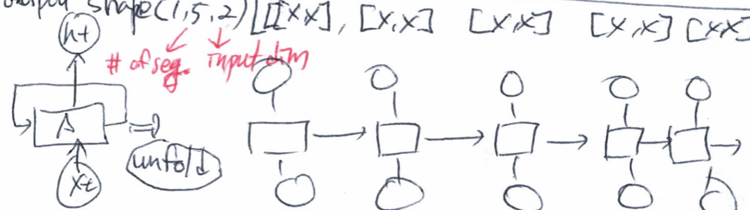
```
hidden_size = 2
cell = tf.contrib.rnn.BasicLSTMCell(
        num_units = hidden_size)
x_data = np.array([[[1, 0, 0, 0]]], dtype = np.float32)
outputs, _states = tf.nn.dynamic_rnn(cell,
        x_data, dtype = tf.float32)
sess.run (tf.global_variables_initializer())
pp.print (outputs.eval())
```

↓

array([[[-0.424~ , 0.64 ~]]])

output shape (1, 5, 2) [[[x, x], [x, x] [x, x] [x, x] [x, x]]]

# of seq. input dim



input shape = (1, 5, 4) : [[[1 0 0 0] [0 1 0 0] [0 0 1 0] [0 0 1 0] [0 0 0 1]]]

# of sequence   input dim

Coding (2)

\# One cell RNN input_dim (4) → output_dim (2). seq = 5

hidden size = 2.
cell ~   h = [1, 0, 0, 0]
x_data = np.array([[h, e, l, l, o]], dtype = float 32)
outputs, _states = tf. nn. dynamic_rnn (cell, x_data, dtype ...)
sess. run (tf. glob ~)
pp. print (outputs. eval())

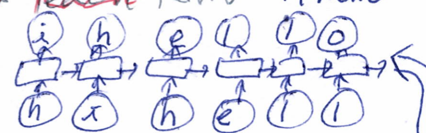
- Batching Input.

batch size = 3

shape = (3, 5, 2)

batch_size   sequence (or hiddensize)   (input dim)

Teach RNN 'hihello'



text : 'hihello'
unique char = h i e l o →

[1 0 0 0 0] h
⋮
[0 0 0 0 1] o

∴ input shape = [[1, 6, 5]]
output shape = [[1, 6, 5]]
single sentence.

- Creating rnn cell

rnn_cell = rnn_cell. BasicRNNCell (rnn_size)
BasicLSTMCell
GRUCell

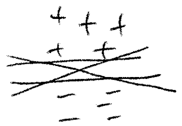
outputs, _states = tf. nn. dynamic_rnn (rnn_cell, X,
initial_state = initial_state,
dtype = tf. float 32)

- Loss !

sequence_loss =

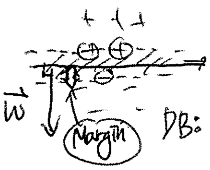
<SUPPORT Vector Machine> aailab kaist channel

- Decision Boundary w/o Prob..



여러가지 DB 가능함 → DB가 data point에 너무 가까우면 data에 noise 낄때 에러발생!!

- Decision Boundary with Margin ⇐ [∴ + & − region에서 가운데 영역을 찾아야해.]



3개의 점의 모음(support vector)을 뽑아서 margin 큰 DB 찾자.

Margin DB: $w \cdot x + b = 0$ positive case (+1) $(w \cdot x + b > 0)$, negative case (−1) $(w \cdot x + b < 0)$

- Margin distance

$f(x) = w \cdot x + b = 0 \rightarrow$ on decision boundary; $= a > 0 \rightarrow$ A positive point x

x_p (DB), r, x

- $x = x_p + r\frac{w}{\|w\|}$, $f(x_p) = 0 \Rightarrow f(x) = w x + b$
$= w\left(x_p + r\frac{w}{\|w\|}\right) + b$
$= wx_p + b + r\frac{ww}{\|w\|}$ $r\|w\|$ = ⓐ

∴ distance $r = \frac{f(x)}{\|w\|}$

- Maximizing the margin.
  - good DB = maximum margin = max(r)!
  - Optimization problem $\begin{cases} \max_{w,b} 2r = \frac{2a}{\|w\|} \\ s.t.\ (wx_j + b)y_j \geq a,\ \forall j \end{cases}$
  - a = arbitrary number and can be normalized

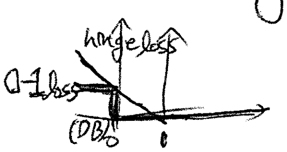
$\boxed{\min_{w,b} \|w\| \quad s.t.\ (wx_j + b)y_j \geq 1,\ \forall j}$

⇒ quadratic optimization problem.
- Quadratic programming 이용해서 optimization.

$\begin{pmatrix} +++ \\ --- \\ + \end{pmatrix}$ → 이런경우엔 "모든 positive 데이터에 대해" 란 조건 만족 X ... ㅠㅠ

- SVM with hard margin
  - Hard margin : No errors case are allowed.
  - solution
    - Soft margin : 에러 인정.
    - Kernel trick : (+ + + + / − − (+) − − −) non linear boundary 사용!!

- "Error" cases in SVM
  - Error handling! ⇒ (Option 1) $\min_{w,b} \|w\| + C \times \#error$ (C: trade-off parameter)
    $s.t.\ (wx_j + b)y_j \geq 1, \forall j$



(Option 2) Slack variable $\xi_j > 1$ when misclassified. error.
$\min_{w,b} \|w\| + C\sum \xi_j$
$s.t.\ (wx_j + b)y_j \geq 1 - \xi_j, \forall j, (\xi_j \geq 0, \forall j)$

- Soft Margin SVM.
$\begin{cases} \min_{w,b} \|w\| + C\sum_j \xi_j \quad s.t.\ (wx_j + b)y_j \geq 1 - \xi_j, \forall j \\ \xi_j \geq 0, \forall j \end{cases}$

- Comparison to Logistic Regression
  - Loss function $\xi_j = loss(f(x_j), y_j)$
  - SVM loss function : Hinge loss

  - $\xi_j = (1 - (wx_j + b)y_j)_+$
  - Logistic Regression loss function : log loss.
    - $\hat{\theta} = \arg\max_\theta \sum_{1 \leq i \leq N} \log(P(Y_i|X_i; \theta))$ ....
  - which loss ftn is preferable?
    - Around the decision boundary?
    - overall place?

$\xi_j = -\log(P(Y_j|X_j, w, b))$

- Strength of the Loss Function $= \log(1 + e^{(wx_j + b)y_j})$

- non-linear decision boundary.

Feature Mapping to Expand Dim.

- $\min_{w,b,\xi_j} \|w\| + C\sum_j \xi_j \quad s.t.\ (w\varphi(x_j) + b)y_j \geq 1 - \xi_j, \forall j$
$\xi_j \geq 0, \forall j$
- $\varphi(\langle x_1, x_2 \rangle) =$
$\langle x_1, x_2, x_1^2, x_2^2, x_1x_2, x_1^3 \ldots \rangle$

⋮