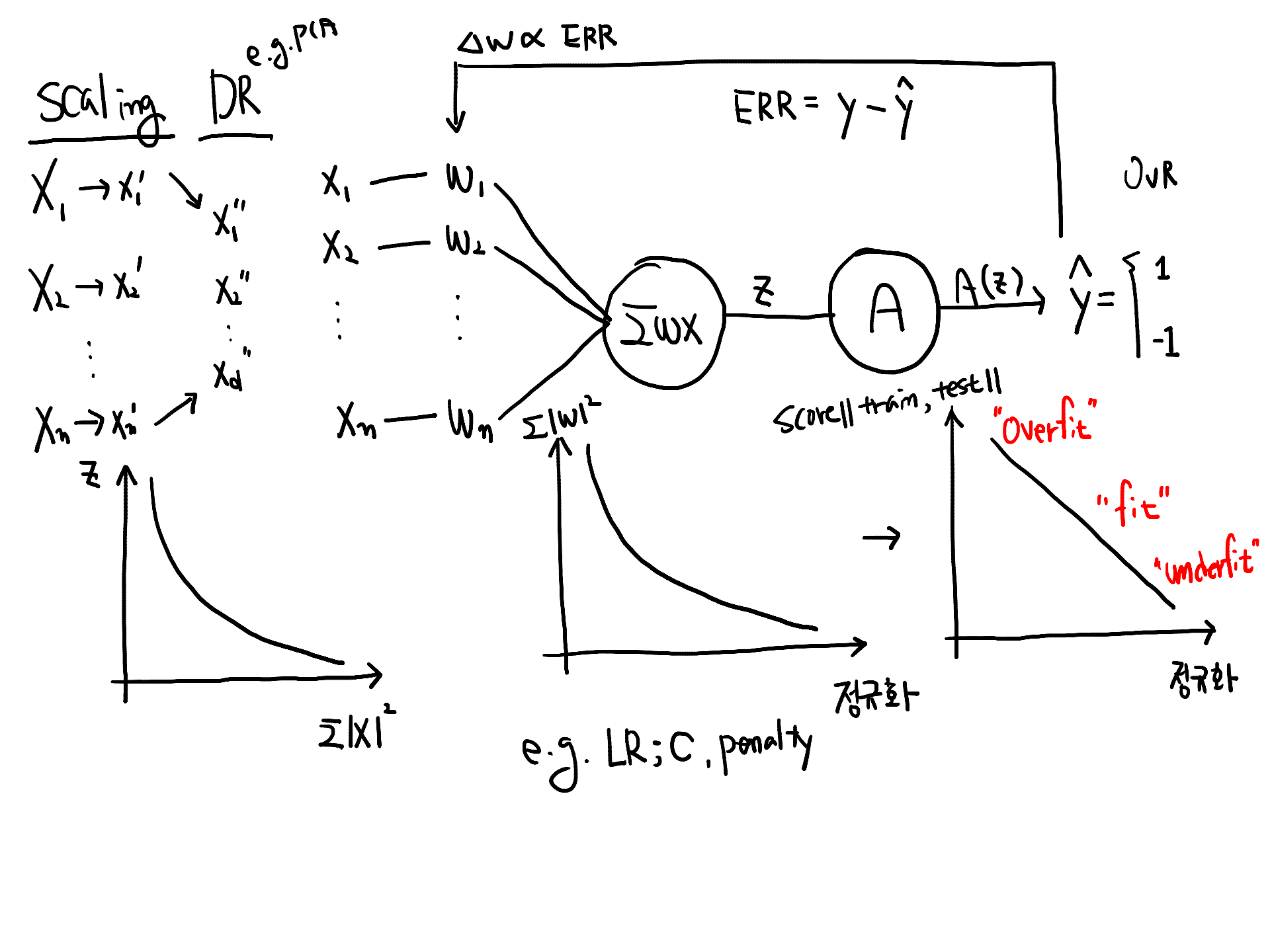
파이썬 딥러닝

이성주

seongjoo@codebasic.io


Scaling
DR
e.g. PCA
$X_1 \to X_1' \to X_1''$
$X_2 \to X_2' \to X_2''$
$X_n \to X_n' \to X_d''$
Z
$\Sigma|X|^2$
$\Delta w \propto ERR$
$ERR = y - \hat{y}$
X_1 — W_1
X_2 — W_2
X_n — W_n
ΣWX
Z
A
$A(Z)$
$\hat{y} = \begin{cases} 1 \\ -1 \end{cases}$
OvR
$\Sigma|W|^2$
정규화
e.g. LR; C, penalty
score|| train, test ||
"Overfit"
"fit"
"underfit"
정규화

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [7]:
```
scaler = StandardScaler()
```

In [14]:
```
Xstd = scaler.fit_transform(X)
```

In [15]:
```
from sklearn.decomposition import PCA
```

In [16]:
```
pca2 = PCA(n_components=2)
```

In [17]:
```
Xpca = pca2.fit_transform(Xstd)
```

In [18]:
```
X_tr, X_te, Xpca_tr, Xpca_te, y_tr, y_te = train_test_split(
    Xstd, Xpca, y, test_size=0.3)
```

In [ ]:

jupyter Day 4

Logout

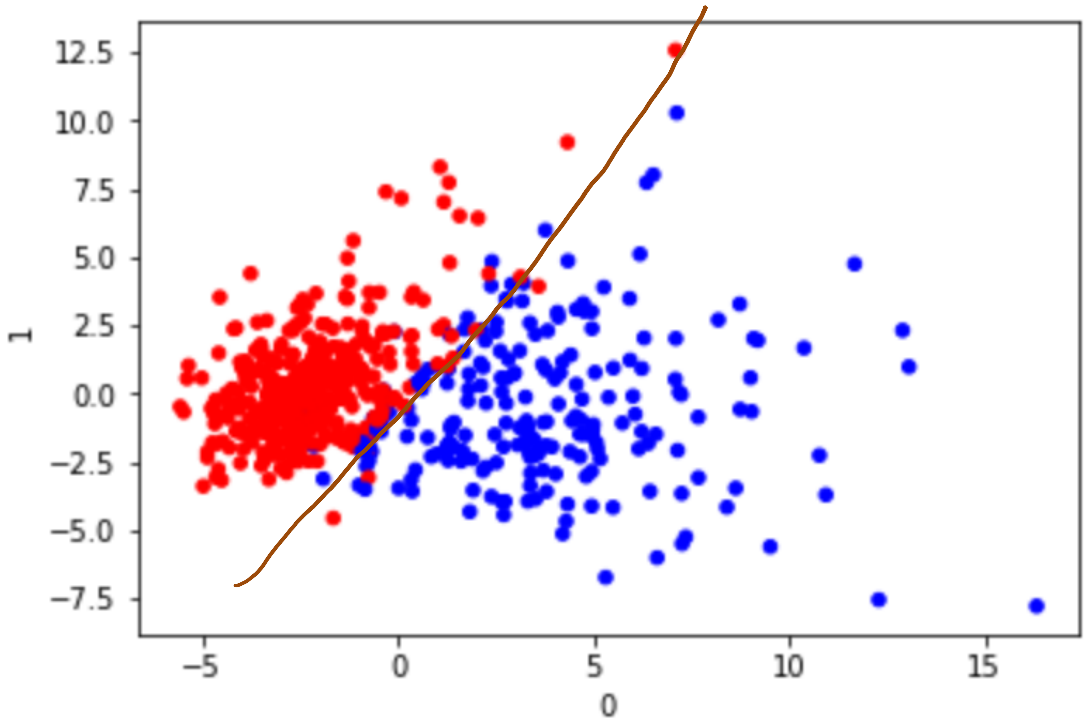
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [29]:
```
c = get_colormap(y, colors='rb')
DataFrame(Xpca).plot(kind='scatter', x=0, y=1, c=c)
```

Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0xbeae2e8>



jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [4]:
```
cancer = pd.read_csv('data/wdbc.data', header=None)
```

In [6]:
```
cancer[:10]
```

y = {M, B}

Out[6]:

	0	1	2	3	4	5	6	7	8	
0	842302	M	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.30010	0.1
1	842517	M	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.08690	0.0
2	84300903	M	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.19740	0.1
3	84348301	M	11.42	20.38	77.58	386.1	0.14250	0.28390	0.24140	0.1
4	84358402	M	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.19800	0.1
5	843786	M	12.45	15.70	82.57	477.1	0.12780	0.17000	0.15780	0.0
6	844359	M	18.25	19.98	119.60	1040.0	0.09463	0.10900	0.11270	0.0
7	84458202	M	13.71	20.83	90.20	577.9	0.11890	0.16450	0.09366	0.0
8	844981	M	13.00	21.82	87.50	519.8	0.12730	0.19320	0.18590	0.0
9	84501001	M	12.46	24.04	83.97	475.9	0.11860	0.23960	0.22730	0.0

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [14]:
```
Xstd = scaler.fit_transform(X)
```

In [15]:
```
from sklearn.decomposition import PCA
```

In [16]:
```
pca2 = PCA(n_components=2)
```

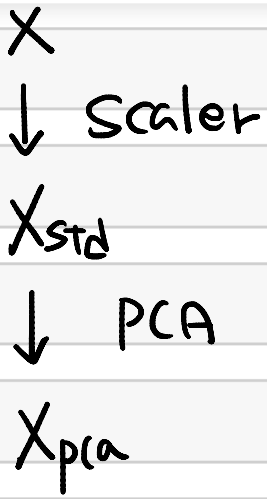
In [33]:
```
Xpca2 = pca2.fit_transform(Xstd)
```

In [30]:
```
pca3 = PCA(n_components=3)
```

In [31]:
```
Xpca3 = pca3.fit_transform(Xstd)
```

In [34]:
```
X_tr, X_te, Xpca2_tr, Xpca2_te, Xpca3_tr, Xpca3_te, y_tr, y_te = tra
    Xstd, Xpca2, Xpca3, y, test_size=0.3)
```

In [35]:
```
model = LogisticRegression(C=1.0)
```



jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [49]:
```
from sklearn.pipeline import Pipeline
```

In [50]:
```
pipe1 = Pipeline([
    ('scaler', StandardScaler()),
    ('model', LogisticRegression(C=1.))
])
```

In [51]:
```
pipe_pca = Pipeline([
    ('scaler', StandardScaler()),
    ('pca', PCA(n_components=2)),
    ('model', LogisticRegression(C=1.))
])
```

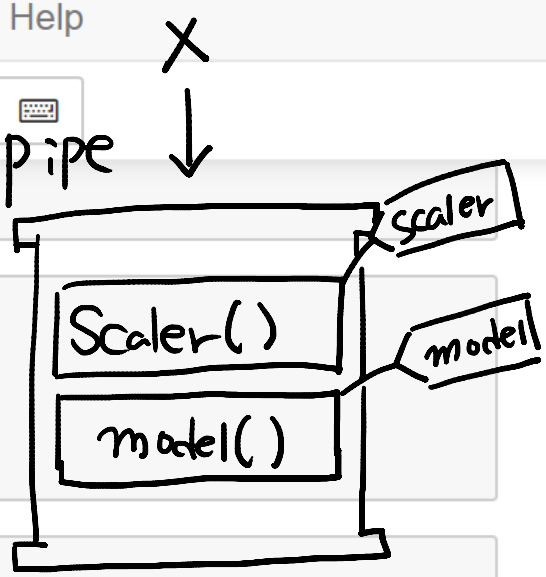
In [ ]:

fit-transform()

(라벨, 변환기)

⋮

(라벨, 모델)



jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [53]:
```
pipe1.fit(X_train, y_train)
```

...

In [54]:
```
pipe1.score(X_test, y_test)
```

Out[54]: 0.9883040935672514 6

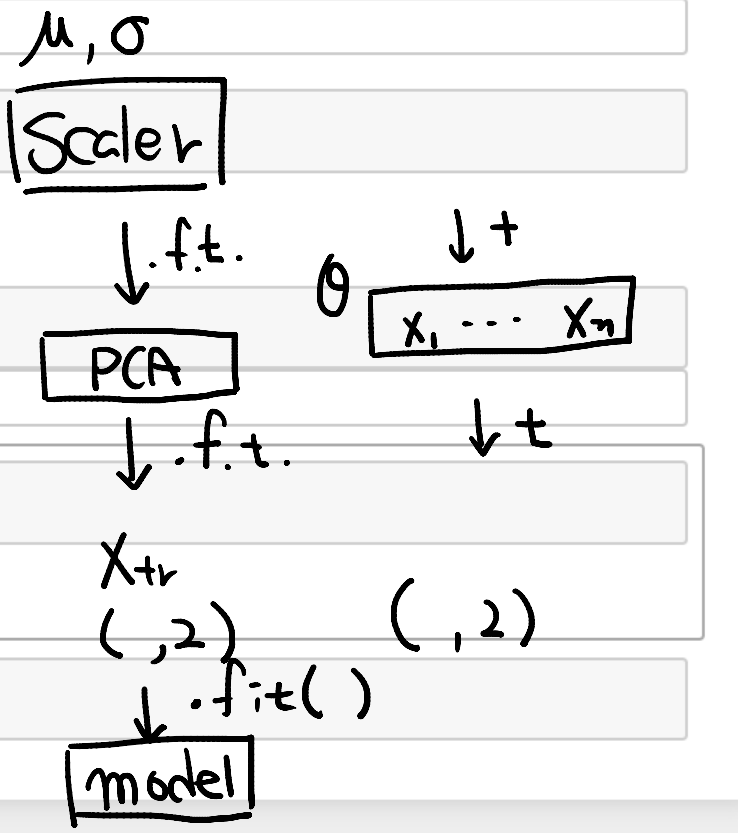
In [55]:
```
pipe_pca.fit(X_train, y_train)
```

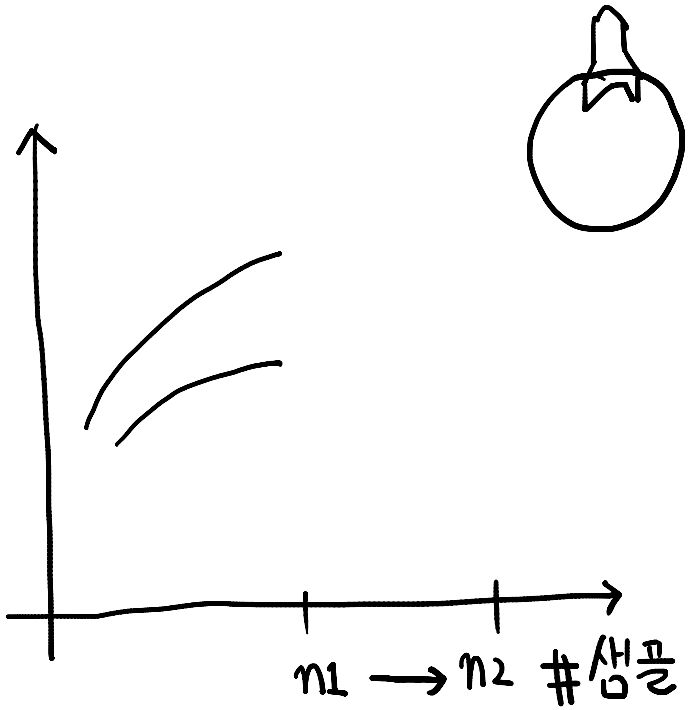
...

In [56]:
```
pipe_pca.score(X_test, y_test)
```

Out[56]: 0.9590643274853801 4

In [ ]:




n1 → n2 #샘플

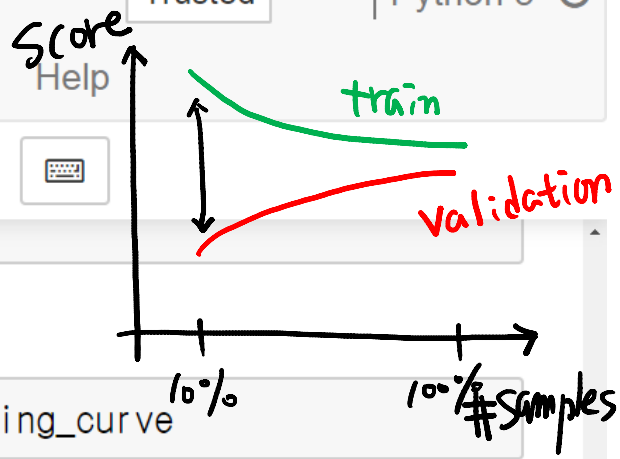
jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

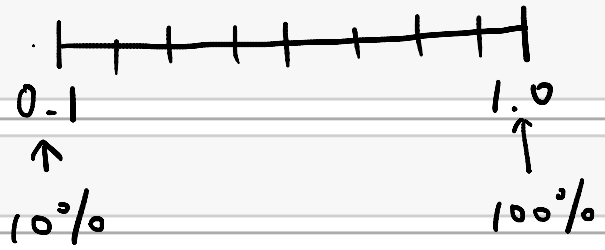


In [56]: pipe_pca.score(X_test, y_test)

Out[56]: 0.95906432748538006

In [57]:
```
from sklearn.model_selection import learning_curve
```

In [58]:
```
train_sizes, train_scores, val_scores = learning_curve(
    estimator=pipe1,
    X=X_train, y=y_train,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=5
)
```



In [ ]:

jupyter Day 4

Logout

Trusted | Python 3

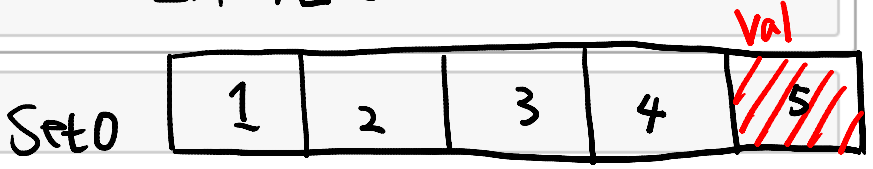
File Edit View Insert Cell Kernel Widgets Help

Code

In [58]:
```
train_sizes, train_scores, val_scores = learning_curve(
    estimator=pipe1,
    X=X_train, y=y_train,
    train_sizes=np.linspace(0.1, 1.0, 10),
    cv=5
)
```

교차 확인 (Cross-Validation; CV)

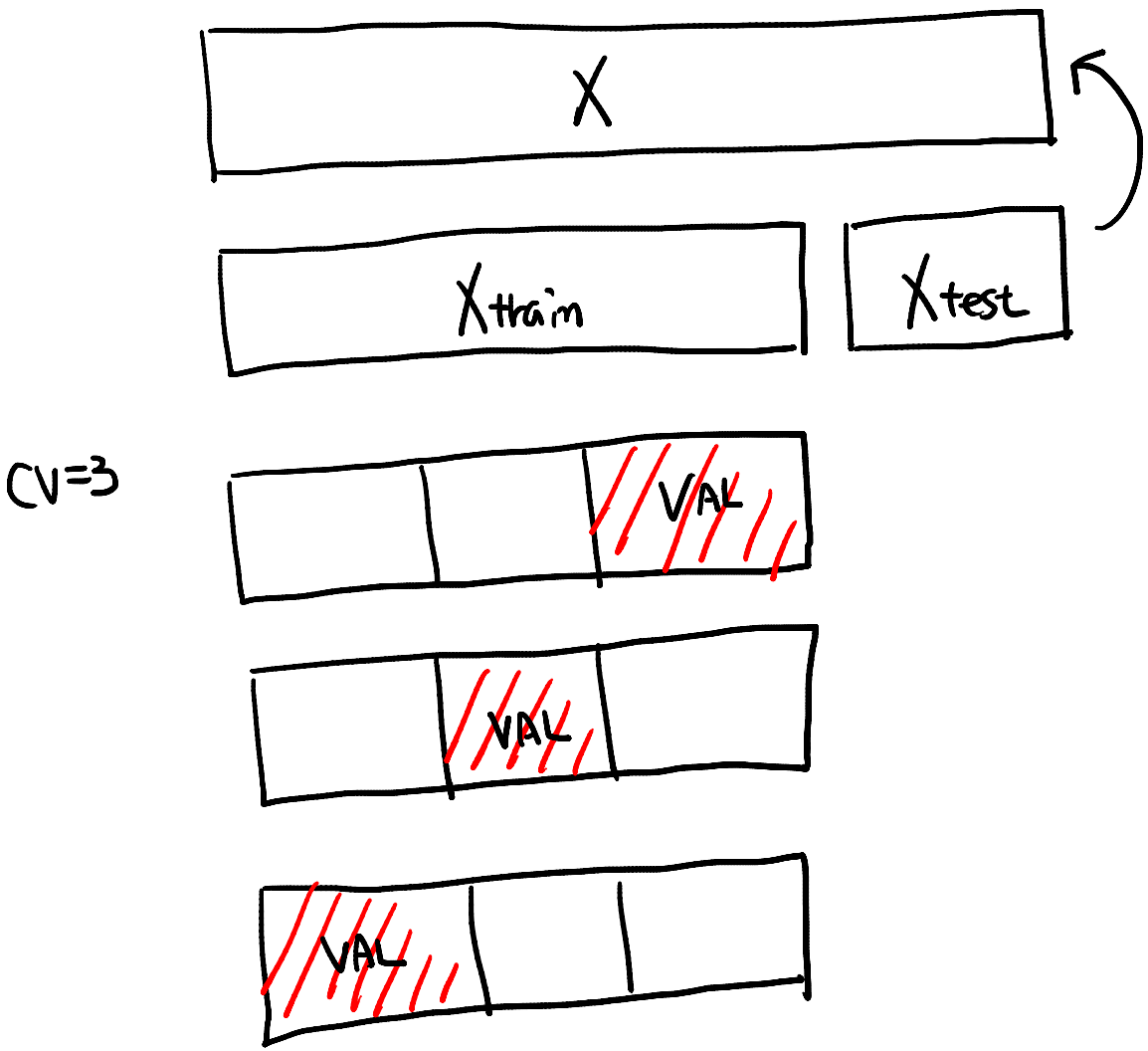
.fit(train)



In [59]:
```
DataFrame(train_scores)
```

Out[59]:

#samples	Set 0	Set 1	Set 2	Set 3	Set 4
10% 0	1.000000	0.967742	0.967742	0.967742	0.967742 → μ
20% 1	0.984127	0.984127	0.984127	0.984127	0.984127 → μ
2	0.989474	0.989474	0.989474	0.989474	0.989474
3	0.992063	0.992063	0.984127	0.984127	0.984127
4	0.993671	0.993671	0.987342	0.987342	0.987342
5	0.994737	0.994737	0.989474	0.989474	0.989474
6	0.990950	0.981900	0.981900	0.986425	0.986425


X
Xtrain
Xtest
CV=3
VAL
VAL
VAL


출구
투표함

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

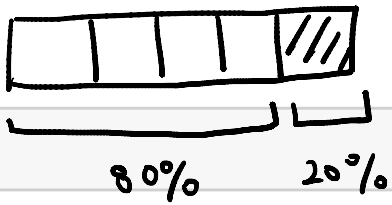
In [61]:
```
X_train.shape
```

Out[61]: (398, 30)

#Sample

In [62]:
```
398 * 0.1
```

Out[62]: 39.800000000000004



In [64]:
```
39 * 0.8
```

Out[64]: 31.200000000000003

In [60]:
```
DataFrame(train_scores, index=train_sizes)
```

Out[60]:

	0	1	2	3	4
31	1.000000	0.967742	0.967742	0.967742	0.967742
63	0.984127	0.984127	0.984127	0.984127	0.984127
95	0.989474	0.989474	0.989474	0.989474	0.989474
126	0.992063	0.992063	0.984127	0.984127	0.984127

10% →

jupyter Day 4

Logout

Trusted | Python 3

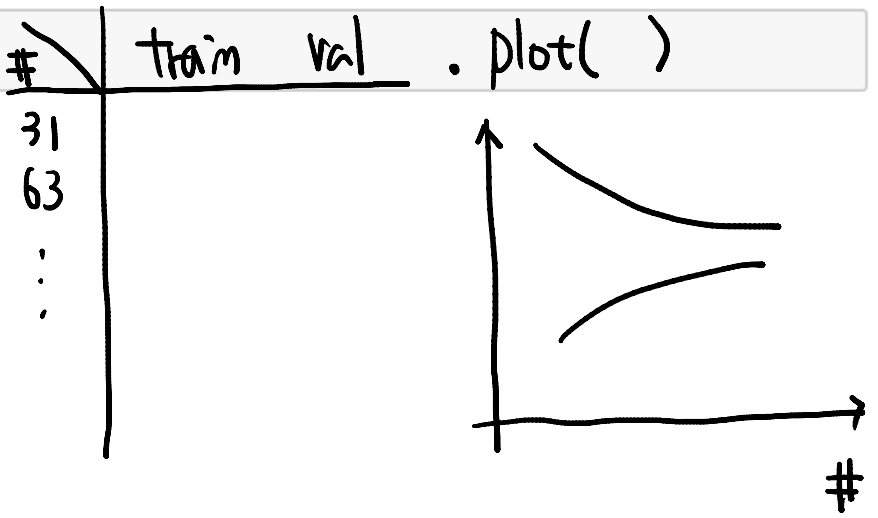
File Edit View Insert Cell Kernel Widgets Help

Code

In [73]:
```
train_results.name = 'train'
```

In [74]:
```
train_results
```

Out[74]:
```
31     0.974194
63     0.984127
95     0.989474
126    0.987302
158    0.989873
190    0.991579
221    0.985520
253    0.987352
285    0.986667
317    0.986751
Name: train, dtype: float64
```



In [70]:
```
val_results = DataFrame(val_scores, index=train_sizes).mean(1)
```

In [72]:
```
val_results
```

Out[72]: 31 0.952275

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [76]:
```
learn_results = pd.concat([train_results, val_results], axis=1)
```

Out[76]:

axis → 1

0 ↓

Concatenate "이어붙이다"

[Series, Series, ···]

	train	validation
31	0.974194	0.952275
63	0.984127	0.964870
95	0.989474	0.969934
126	0.987302	0.967402
158	0.989873	0.967402
190	0.991579	0.967402
221	0.985520	0.969871
253	0.987352	0.967402
285	0.986667	0.969871
317	0.986751	0.972403

In [ ]:

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

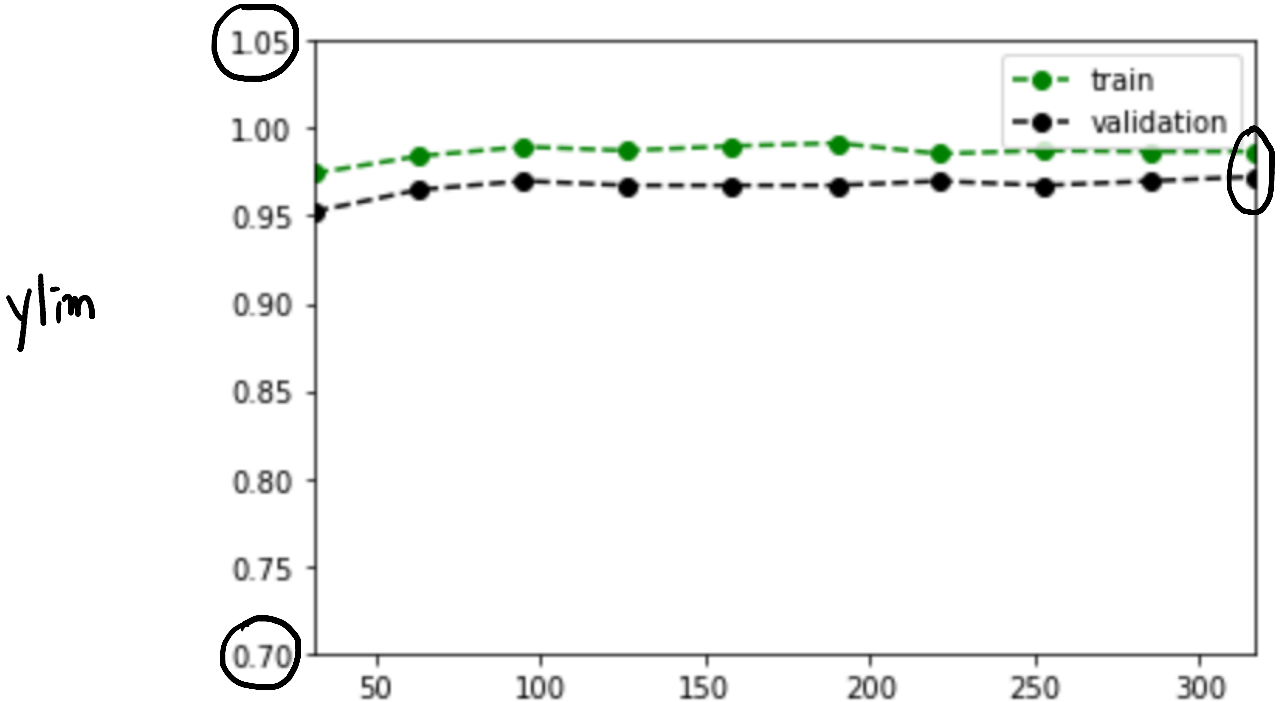
Code

In [77]:
```
learn_results = pd.concat([train_results, val_results], axis=1)
```

black

In [79]:
```
learn_results.plot(style=['go--', 'ko--'], ylim=(0.7, 1.05))
```

Out[79]: <matplotlib.axes._subplots.AxesSubplot at 0xc510a58>



ylim

overfit

jupyter Day 4

Logout

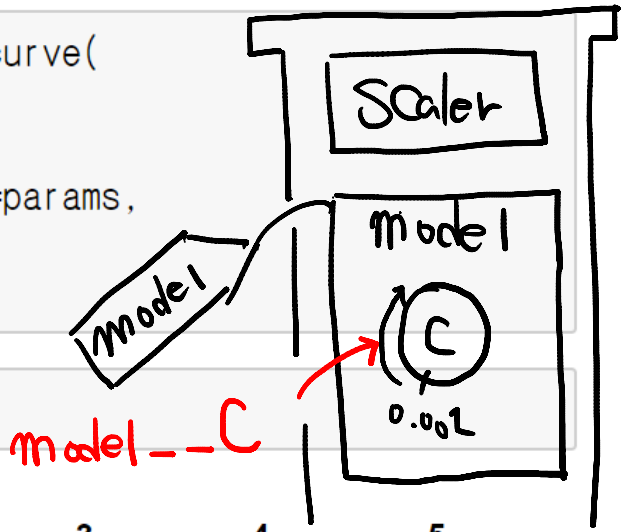
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [81]:
```
params = [0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]
```

In [82]:
```
train_scores, val_scores = validation_curve(
    estimator=pipe1,
    X=X_train, y=y_train,
    param_name='model__C', param_range=params,
    cv=10
)
```



In [83]:
```
DataFrame(train_scores, index=params)
```

Out[83]:

	Set 0	Set 1	Set 2	3	4	5	
0.001	0.938375	0.946927	0.941341	0.938547	0.946927	0.944134	0.946
0.010	0.971989	0.966480	0.966480	0.966480	0.972067	0.966480	0.969
0.100	0.988796	0.980447	0.983240	0.983240	0.986034	0.983240	0.980
1.000	0.991597	0.983240	0.983240	0.988827	0.986034	0.980447	0.986
10.000	0.991597	0.986034	0.988827	0.986034	0.986034	0.986034	0.988

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

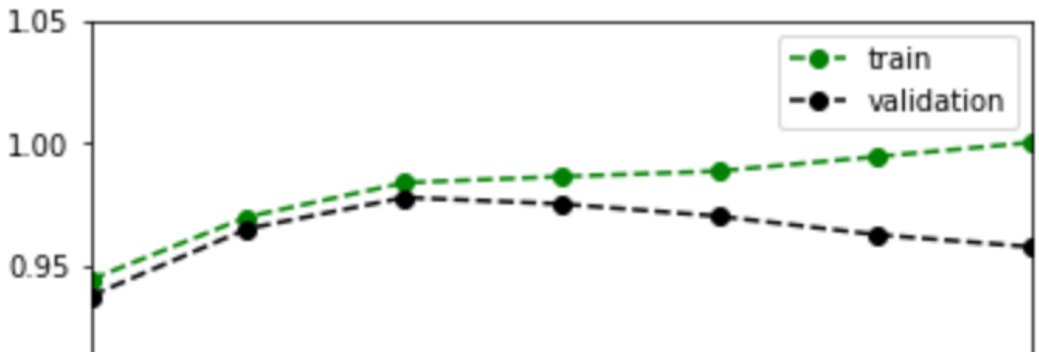
In [85]:
```
train_results = DataFrame(train_scores, index=params).mean(1)
train_results.name = 'train'
```

In [87]:
```
val_results = DataFrame(val_scores, index=params).mean(1)
val_results.name = 'validation'
```

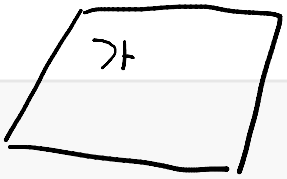
In [89]:
```
param_results = pd.concat([train_results, val_results], axis=1)
```

In [90]:
```
param_results.plot(
    logx=True, ylim=(0.8, 1.05), style=['go--', 'ko--'])
```

Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0xc481518>



jupyter Day 4



Logout

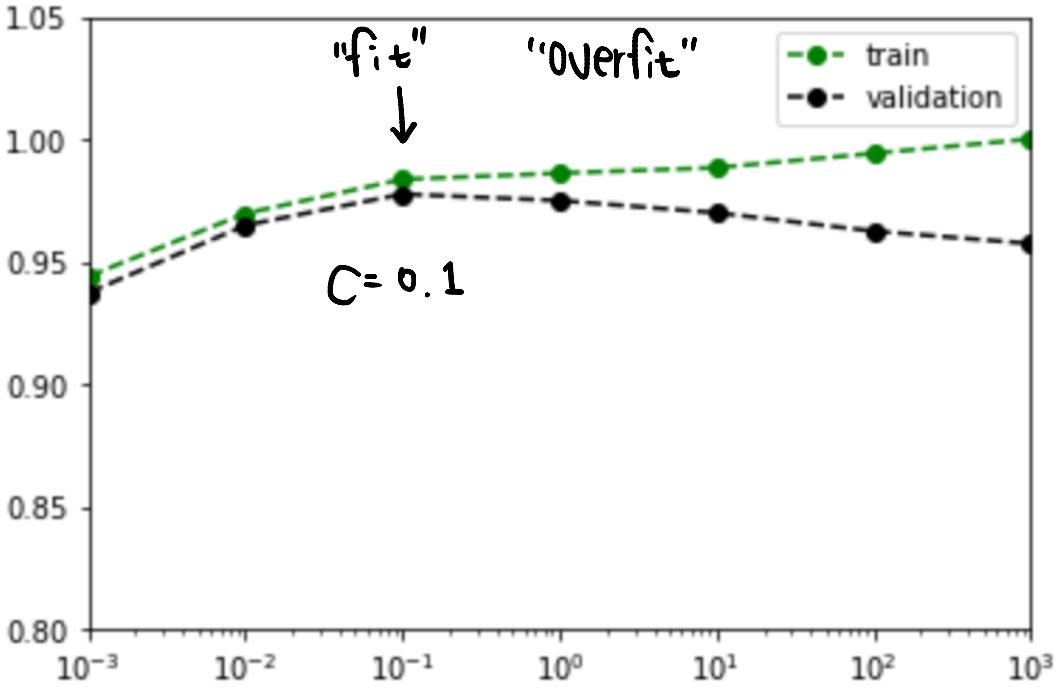
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

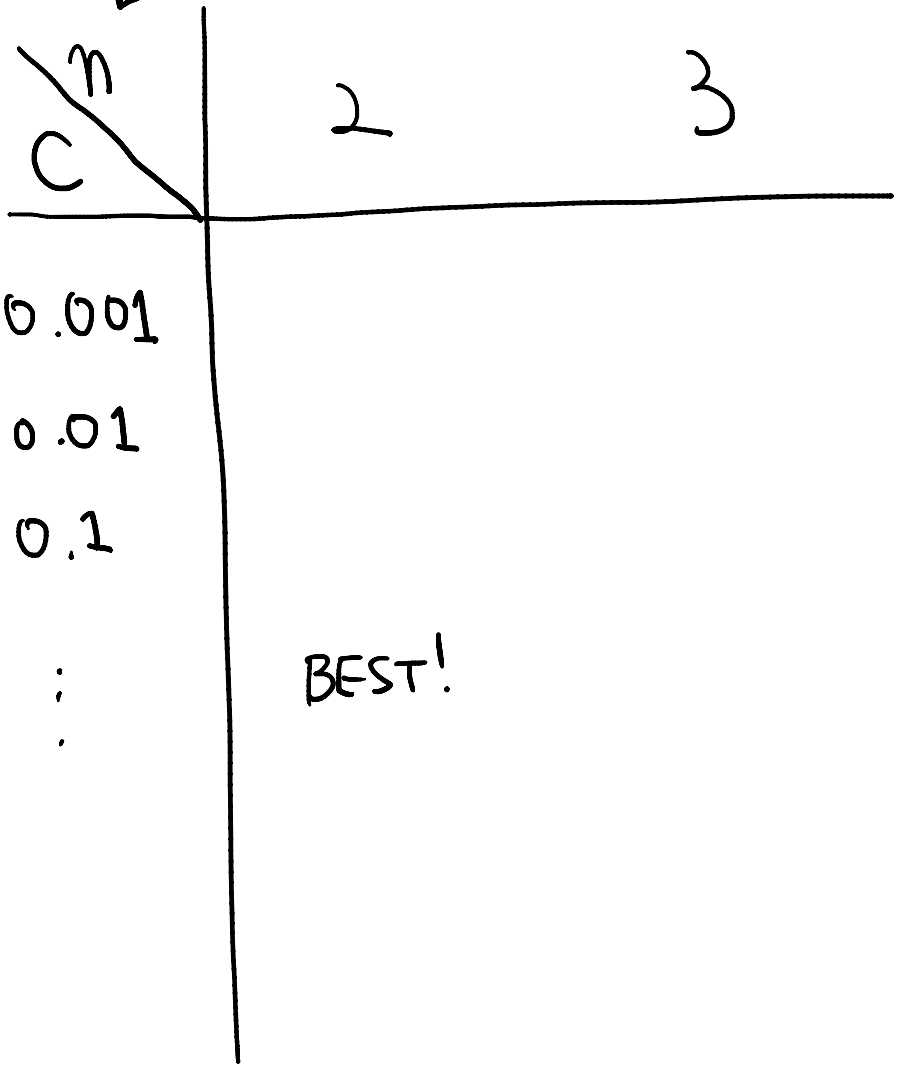
Code

In [90]:
```
param_results.plot(
    logx=True, ylim=(0.8, 1.05), style=['go--', 'ko--'])
```

Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0xc481518>



Grid Search, PCA



jupyter Day 4

Logout

Trusted | Python 3

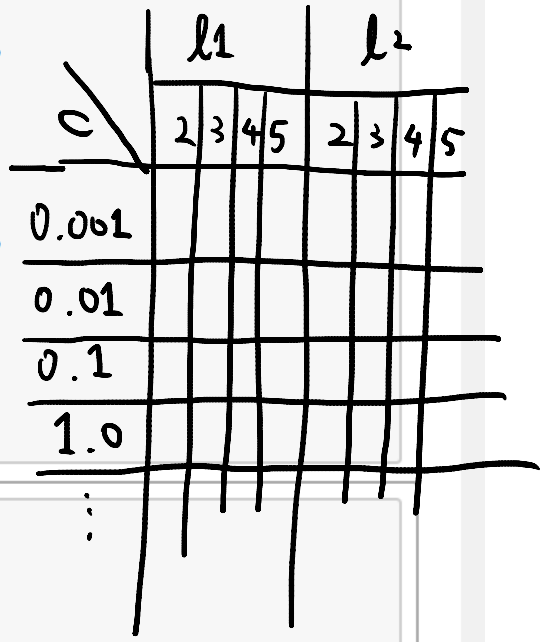
File Edit View Insert Cell Kernel Widgets Help

Code

In [92]:
```
param_grid = [
    {
        'pca__n_components': [None, 2, 3, 4, 5],
        'model__C': params
    },
    {
        'pca__n_components': [None, 2, 3, 4, 5],
        'model__C': params,
        'model__penalty': ['l1', 'l2']
    }
]
```

set1

set2



In [95]:
```
gs = GridSearchCV(
    estimator=pipe_pca,
    param_grid=param_grid,
    scoring='accuracy',
    cv=10
)
```

In [ ]:

jupyter Day 4

Logout

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

Code

In [103]:
```
gs.fit(X_train, y_train)
```

n_components

[None, ...]

...

In [104]:
```
gs.best_score_
```

Out[104]:
```
0.9773869346733668
```

In [105]:
```
gs.best_params_
```

"l1"

"l2"

$ERROR += \sum |w|^2$

Out[105]:
```
{'model__C': 0.1, 'model__penalty': 'l2', 'pca__n_components': None}
```

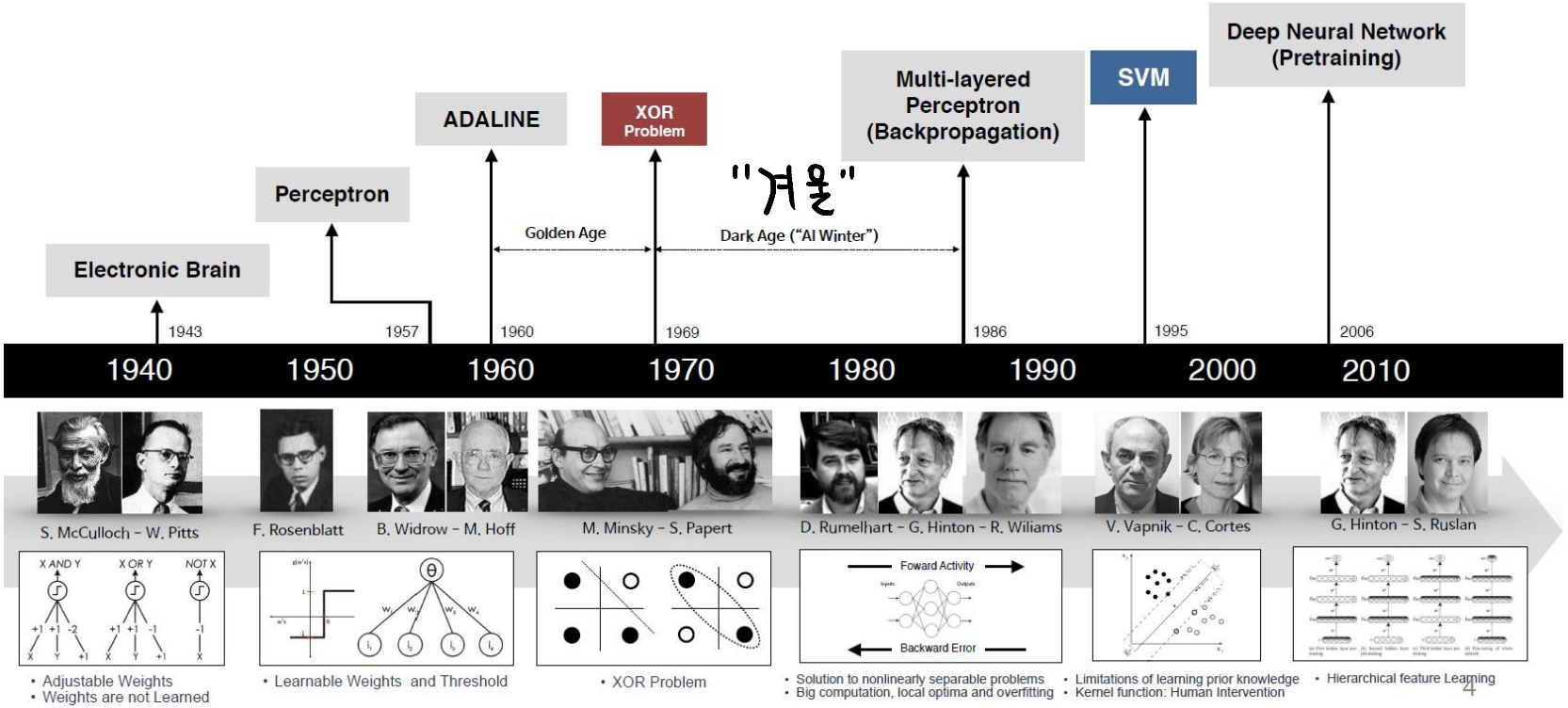
In [106]:
```
best_model = gs.best_estimator_
```

In [107]:
```
best_model.score(X_test, y_test)
```

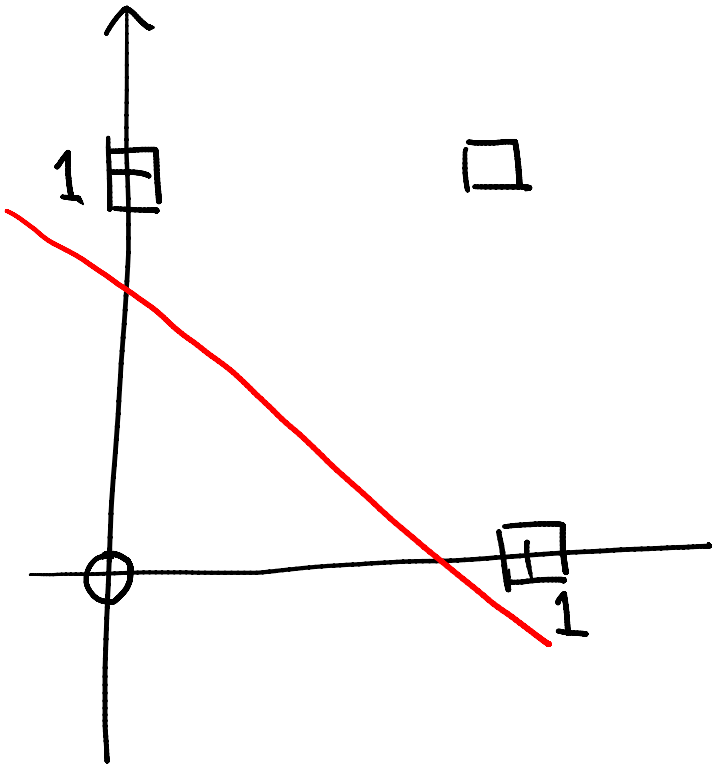
Out[107]:
```
0.9883040935672515
```

In [ ]:

인공 신경망의 발전사

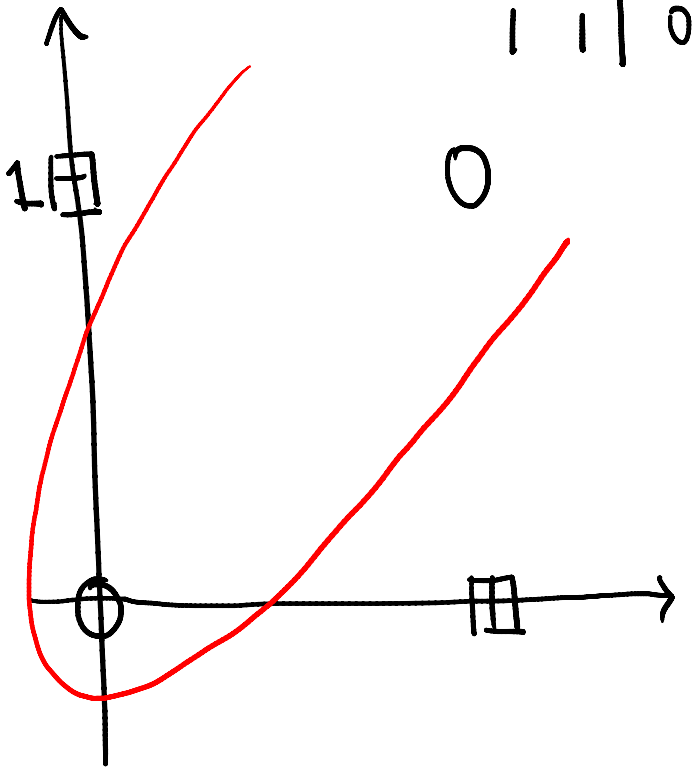

Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
XOR Problem
ADALINE
Perceptron
Electronic Brain
"겨울"
Golden Age
Dark Age ("AI Winter")
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

OR



XOR

0	0	0
1	0	1
0	1	1
1	1	0



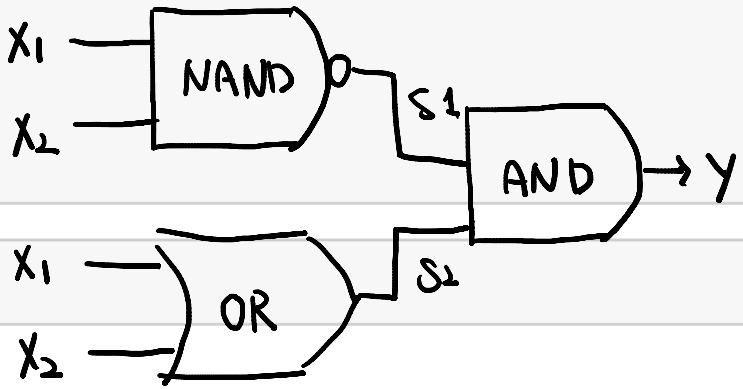
```
        print(x1, x2, '|', y)
```

In [117]:
```
AND = make_logic_gate(w=np.array([0.5, 0.5]), b=0.7)
```

In [118]:
```
OR = make_logic_gate(w=np.array([0.5, 0.5]), b=0.2)
```

In [119]:
```
NAND = make_logic_gate(w=np.array([-0.5, -0.5]), b=-0.7)
```

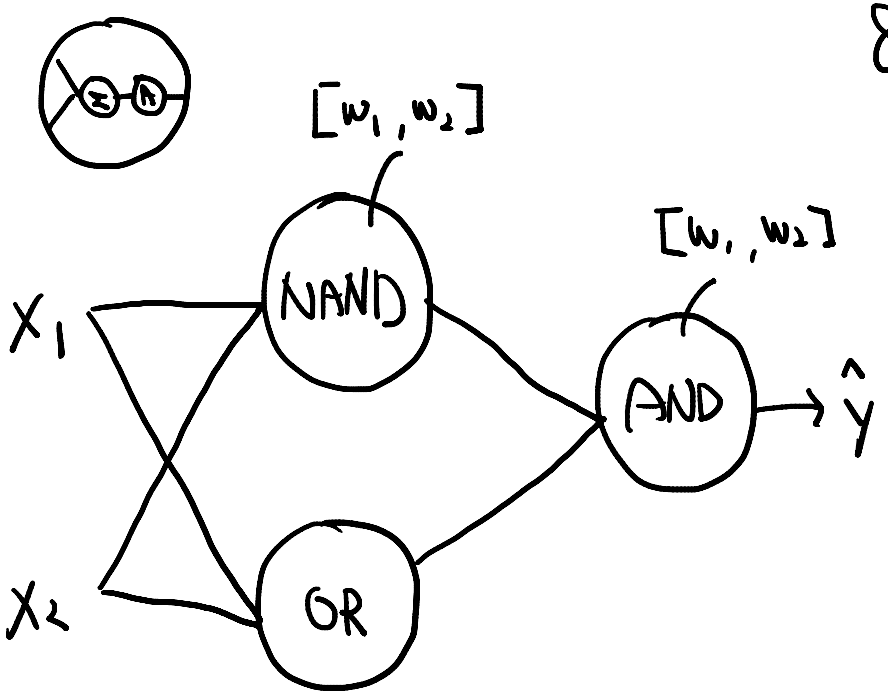
In [120]:
```
def XOR(x1, x2):
    s1 = NAND(x1, x2)
    s2 = OR(x1, x2)
    return AND(s1, s2)
```



In [121]:
```
test_logic(XOR)
```

```
0 0 | 0
0 1 | 1
1 0 | 1
1 1 | 0
```

In [ ]:



$$\begin{bmatrix} w^*_{11} & w^*_{12} \\ w^*_{21} & w^*_{22} \end{bmatrix}$$

80s 인공신경망 ⇝ 21C Deep learning

	Δw			
X_1	0	0		
X_2	0	0	... 0	
⋮	0	0	0	0
	0	0	0	0
	0	0	0	
	0	0		
	100k	100k	10k	

X_{1k} "2 레벨"

Perceptron

x_1 w_1

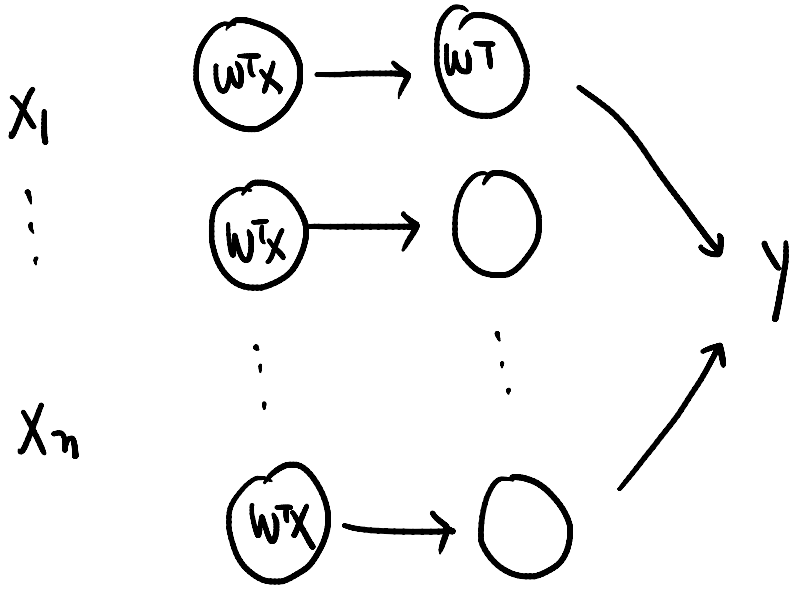
x_2 w_2

$\vdots$ $\vdots$

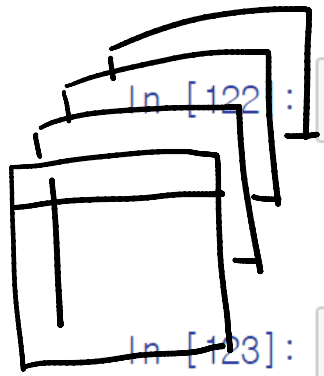
x_n w_n

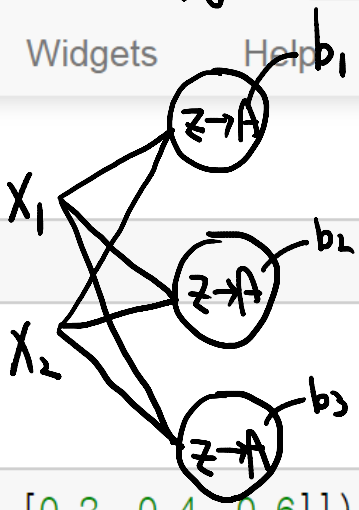
$$w^T x = \sum w x$$

ANN



1 1 | 0





In [122]:
```
x = np.array([1.0, 0.5])
```

1층

In [123]:
```
W1 = np.array([[0.1, 0.3, 0.5], [0.2, 0.4, 0.6]])
b1 = np.array([0.1, 0.2, 0.3])
```

In [125]:
```
z1 = np.dot(x, W1) + b1
```

$$X_1 \rightarrow \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \end{bmatrix} \leftarrow X_2$$

In [126]:
```
z1
```

Out[126]:
```
array([ 0.3,  0.7,  1.1])
```

In [ ]:
```

```

$X_1 \rightarrow W_1$

$X_2 \rightarrow W_2$

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

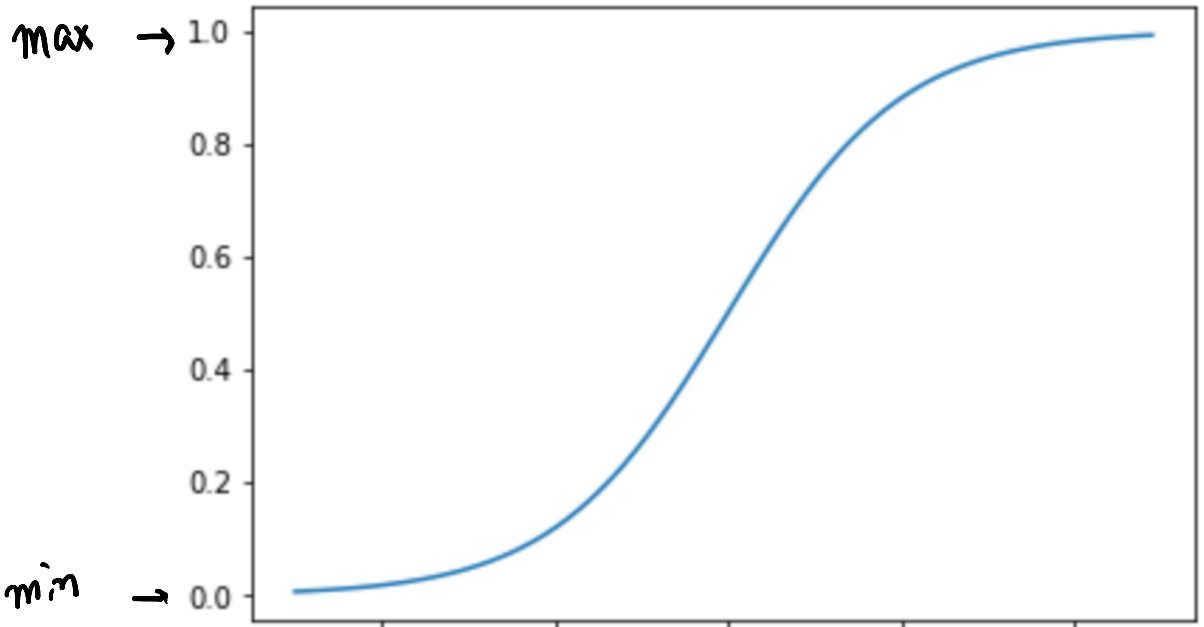
In [127]:
```
def sigmoid(x):
    return 1 / (1 + np.exp(-x))
```

In [128]:
```
x = np.arange(-5., 5., 0.1)
```

In [129]:
```
y = sigmoid(x)
```

In [130]:
```
plt.plot(x, y)
```

Out[130]: [<matplotlib.lines.Line2D at 0xcf91b00>]



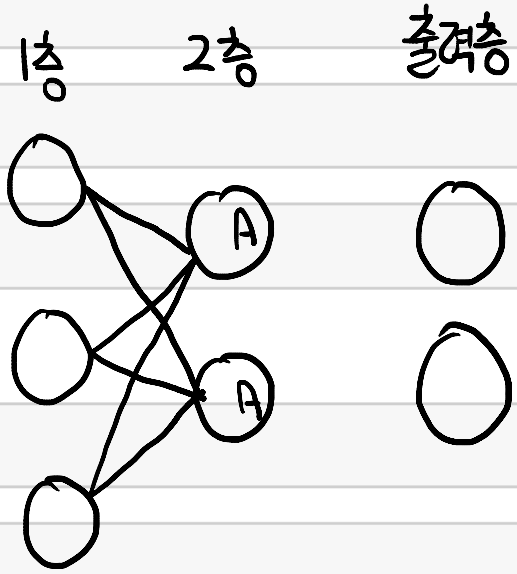
Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

In [132]:
```
a1
```

Out[132]:
```
array([ 0.57444252,  0.66818777,  0.75026011])
```

In [133]:
```
W2 = np.array([[0.1, 0.4], [0.2, 0.5], [0.3, 0.6]])
b2 = np.array([0.1, 0.2])
```



In [134]:
```
z2 = np.dot(a1, W2) + b2
```

In [135]:
```
z2
```

Out[135]:
```
array([ 0.51615984,  1.21402696])
```

In [136]:
```
a2 = sigmoid(z2)
```

In [137]:
```
a2
```

Out[137]:
```
array([ 0.62624937,  0.7710107 ])
```

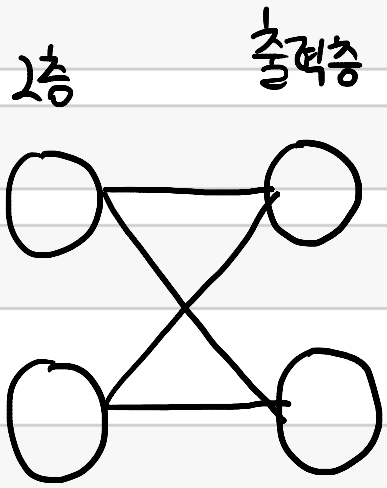
In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

출력

In [138]:
```
W3 = np.array([[0.1, 0.3], [0.2, 0.4]])
b3 = np.array([0.1, .02])
```



In [139]:
```
z3 = np.dot(a2, W3) + b3
```

In [140]:
```
z3
```

Out[140]:
```
array([ 0.31682708,  0.51627909])
```

In [141]:
```
z3.sum()
```

Out[141]:
```
0.83310616627299661
```

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

In [140]:
```
z3
```

Out[140]:
```
array([ 0.31682708,  0.51627909])
```

In [143]:
```
sigmoid(z3)
```

Out[143]:
```
array([ 0.57855079,  0.62627728])
```

In [144]:
```
def softmax(a):
    c = np.max(a)
    ea = np.exp(a-c)
    return ea / np.sum(ea)
```

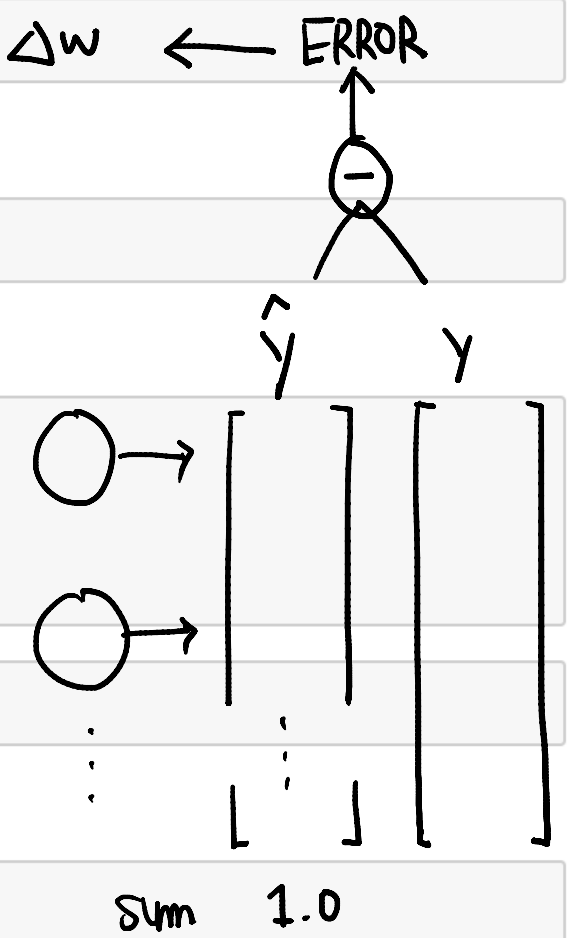
In [145]:
```
softmax(z3)
```

Out[145]:
```
array([ 0.45030164,  0.54969836])
```

In [146]:
```
softmax(z3).sum()
```

Out[146]:
```
1.0
```

In [ ]:



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

정답 → 0 1 2 3 4 5 6 7 8 9

In [147]:
```
y = np.array([0, 0, 1, 0, 0, 0, 0, 0, 0, 0])
```
→ sum() == 1

In [148]:
```
len(y)
```

Out[148]: 10

0 ~ 1

In [149]:
```
y_pred1 = np.array([
    0.1, 0.05, 0.6, 0.0, 0.05, 0.1, 0.0, 0.1, 0.0, 0.0])
```
0 1 2 "max"

In [150]:
```
len(y_pred1)
```

Out[150]: 10

0.6 0.7 0.8

"max"

In [151]:
```
y_pred2 = np.array([
    0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0])
```
7 8 9

In [152]:
```
len(y_pred2)
```

Out[152]: 10

In [ ]:

$y - \hat{y}$
0 -2
1 -1
2 0
1
2
3
⋮

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

In [151]:
```
y_pred2 = np.array([
    0.1, 0.05, 0.1, 0.0, 0.05, 0.1, 0.0, 0.6, 0.0, 0.0])
```

In [152]:
```
len(y_pred2)
```

Out[152]: 10

In [153]:
```
def 평균제곱오차(y, y_pred):
    return np.sum((y-y_pred)**2).mean()
```

$\frac{1}{n}\sum(y-\hat{y})^2$

In [154]:
```
평균제곱오차(y, y_pred1)
```

Out[154]: 0.19500000000000006

0.6 0.7 0.8

In [155]:
```
평균제곱오차(y, y_pred2)
```

Out[155]: 1.1950000000000001

$+\Delta w$

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

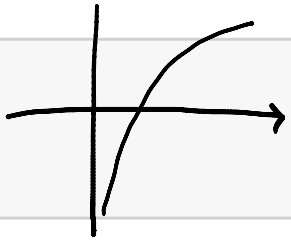
Out[154]: 0.19500000000000006

In [155]:
```
평균제곱오차(y, y_pred2)
```

Out[155]: 1.1950000000000001

다중분류 기본

In [156]:
```
def cross_entropy_error(y, y_pred):
    delta = 1e-7
    return -np.sum(y*np.log(y_pred+delta))
```



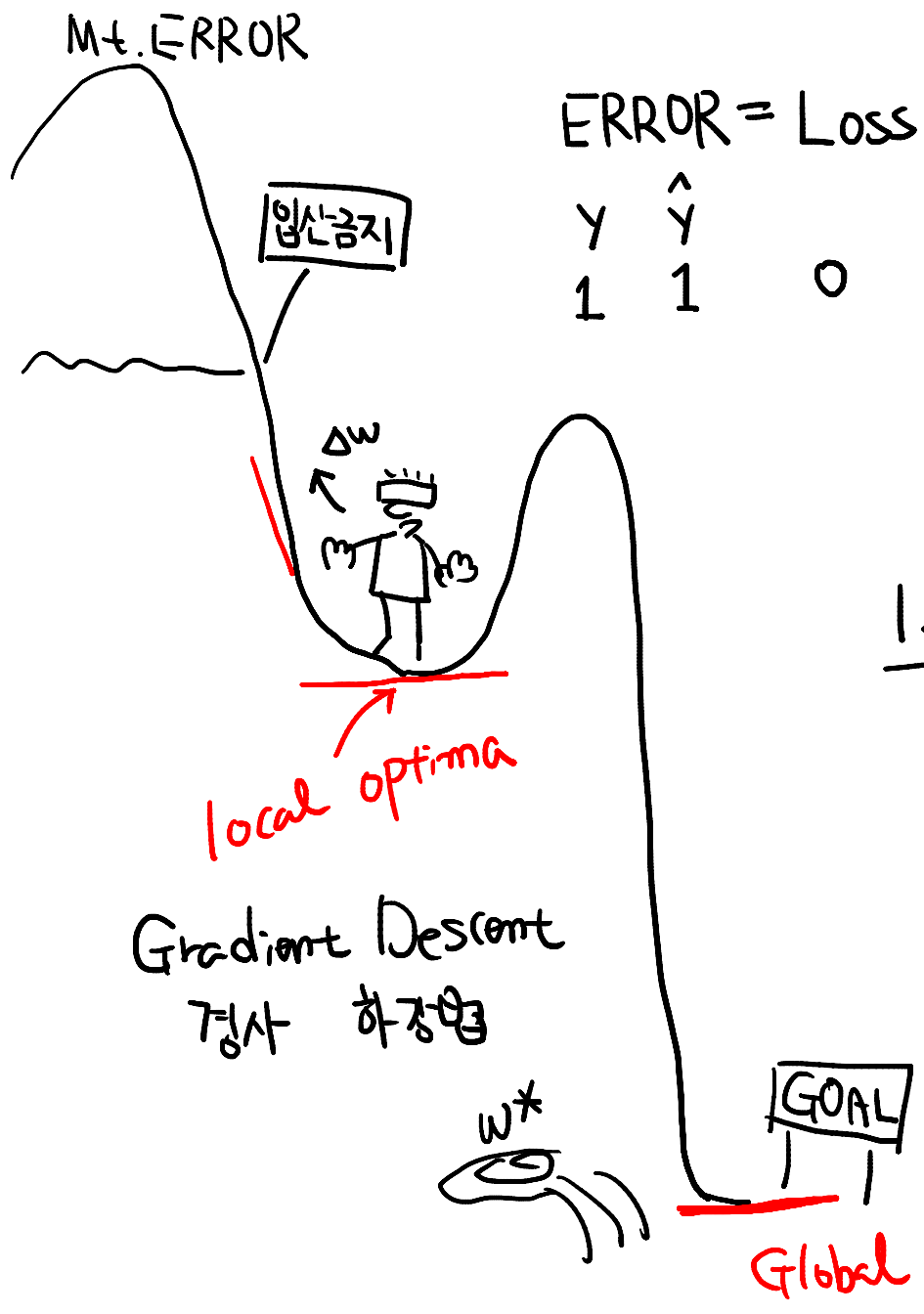
In [157]:
```
cross_entropy_error(y, y_pred1)
```

Out[157]: 0.51082545709933802

In [158]:
```
cross_entropy_error(y, y_pred2)
```

Out[158]: 2.3025840929945458

In [ ]:



ERROR = Loss

y	$\hat{y}$	
1	1	0

$$Loss(w, A) = Diff(y, \hat{y})$$

$$= Diff(y, A(z))$$

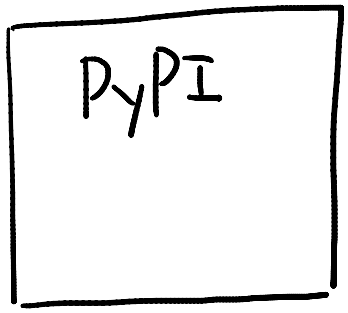
$$= Diff(y, A(w^T x))$$

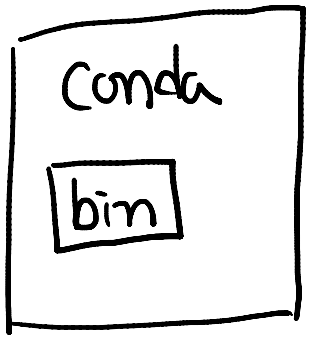
1. 최적화

$$w^* = \underset{w}{\operatorname{argmin}}\, Loss(w, A)$$

$$\frac{\partial L}{\partial w} \rightarrow 0$$

Python Package





$ pip install pkg

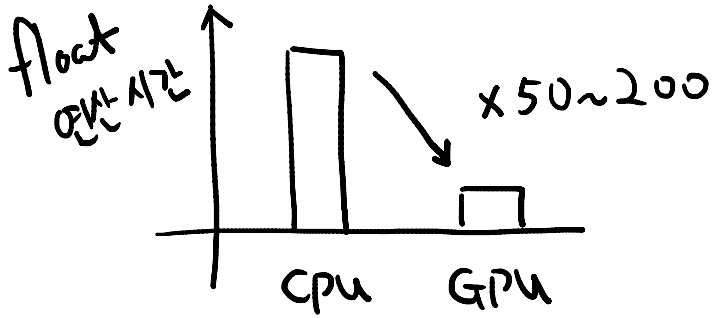
Download ...

Install ...

>>> import pkg

$ Conda install pkg

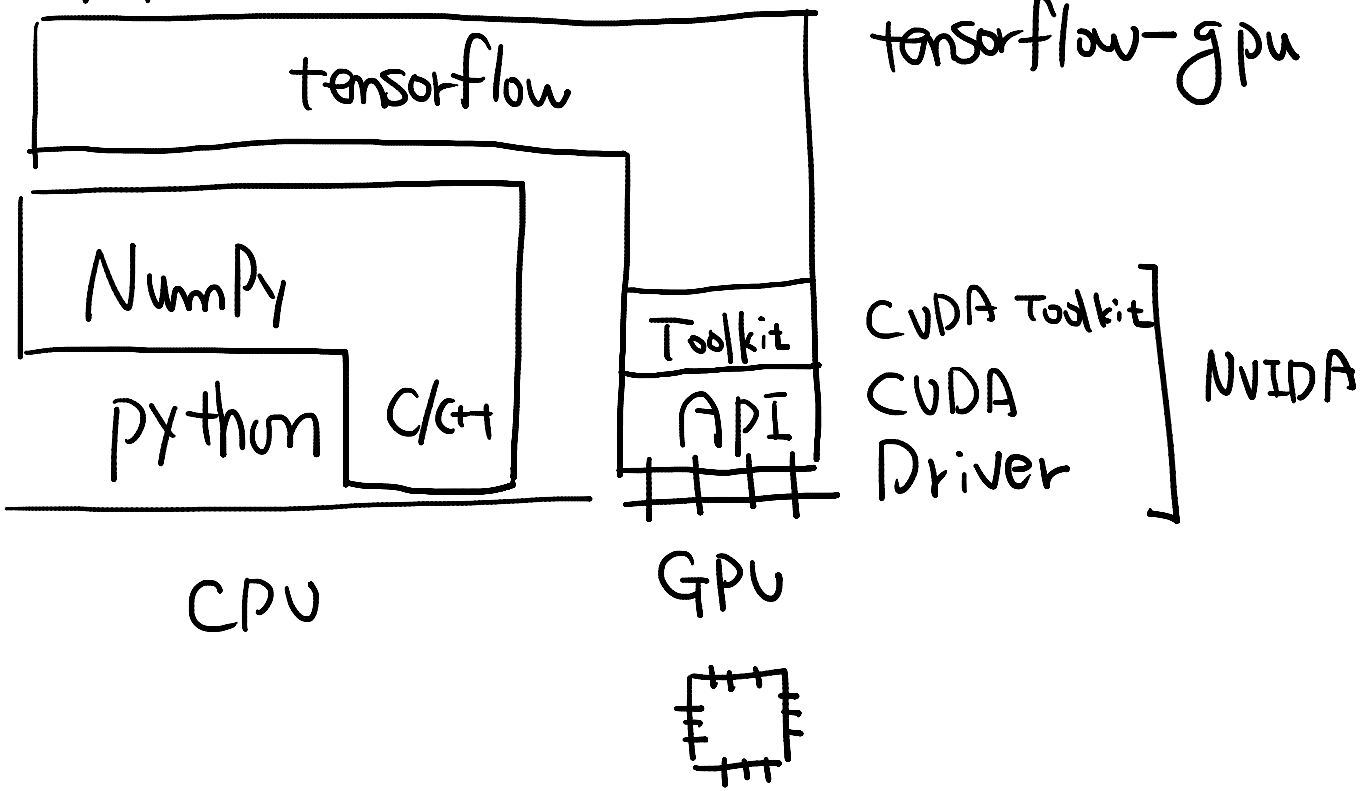
⋮



$ pip install

tensorflow

tensorflow-gpu



Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

In [171]:
```
x = mnist.train.images[0]
```

In [172]:
```
x.shape
```

Out[172]: (784,)

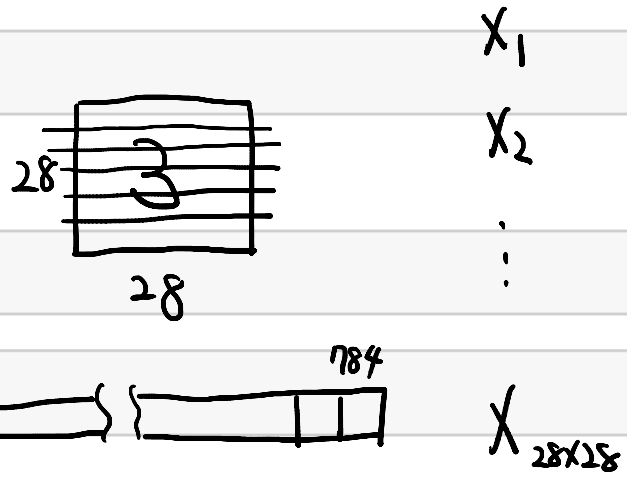
In [173]:
```
type(x)
```

Out[173]: numpy.ndarray

In [174]:
```
x = x.reshape(28, 28)
```

In [175]:
```
x.shape
```

Out[175]: (28, 28)



In [176]:
```
from scipy.misc import imsave
```

In [177]:
```
imsave('mnist sample.png', x)
```

In [ ]:

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

```
Out[173]: numpy.ndarray

In [174]: x = x.reshape(28, 28)

In [175]: x.shape
Out[175]: (28, 28)

In [176]: from scipy.misc import imsave

In [177]: imsave('mnist sample.png', x)

In [179]: mnist.train.labels[0]
Out[179]: array([ 0.,  0.,  0.,  0.,  0.,  0.,  0.,  1.,  0.,  0.])

In [ ]:
```

One Hot Encoding

					HOT	Binary
S	0	RED	0	0	1	00
M	1	Green	0	1	0	01
L	2	Blue	1	0	0	10

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

In [180]:
```
x = tf.placeholder(tf.float32, [None, 784])
```

In [181]:
```
W = tf.Variable(tf.zeros([784, 10]))
b = tf.Variable(tf.zeros([10]))
```

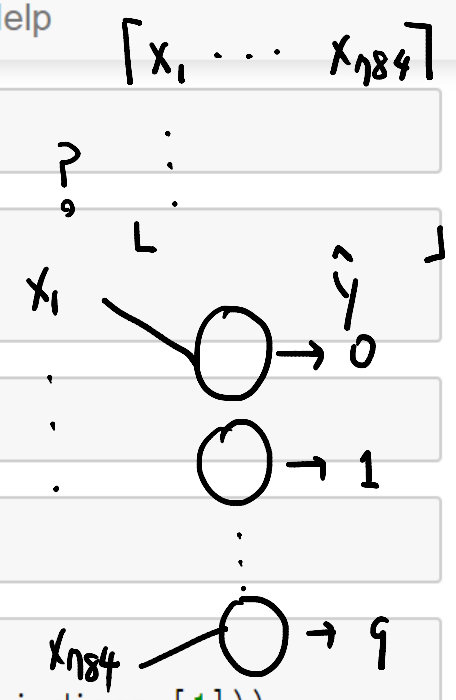
In [182]:
```
y = tf.nn.softmax(tf.matmul(x, W) + b)
```

In [183]:
```
y_ = tf.placeholder(tf.float32, [None, 10])
```

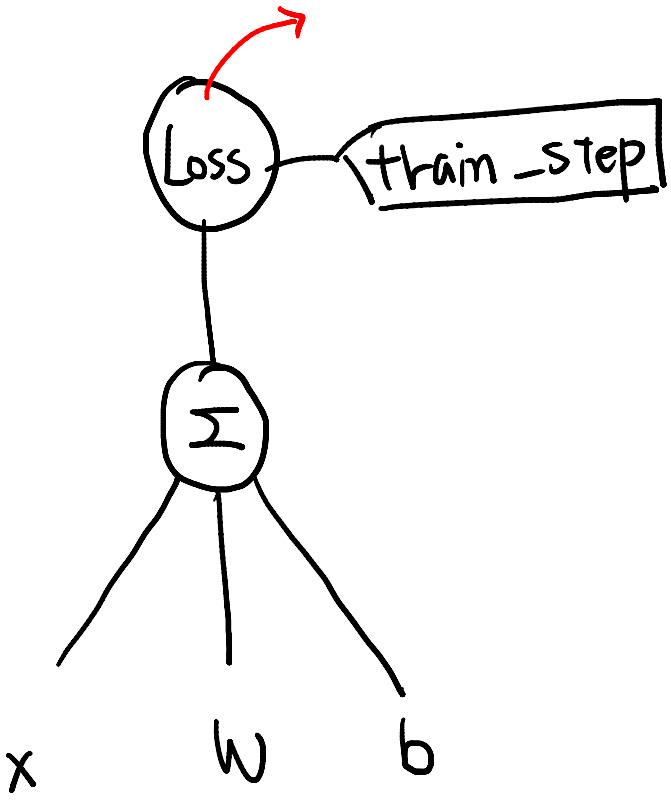
In [184]:
```
cross_entropy = tf.reduce_mean(
    -tf.reduce_sum(y_ * tf.log(y), reduction_indices=[1]))
```

In [185]:
```
train_step = tf.train.GradientDescentOptimizer(0.5).minimize(
    cross_entropy)
```

In [ ]:



정답
np.dot
OHE
손실함수
= 목표


Loss
train_step
Σ
x
W
b

Trusted | Python 3

File Edit View Insert Cell Kernel Widgets Help

In [186]:
```
sess = tf.InteractiveSession()
```

In [187]:
```
tf.global_variables_initializer().run()
```

In [189]:
```
for _ in range(1000):
    batch_xs, batch_ys = mnist.train.next_batch(100)
    sess.run(train_step, feed_dict={x: batch_xs, y_: batch_ys})
```

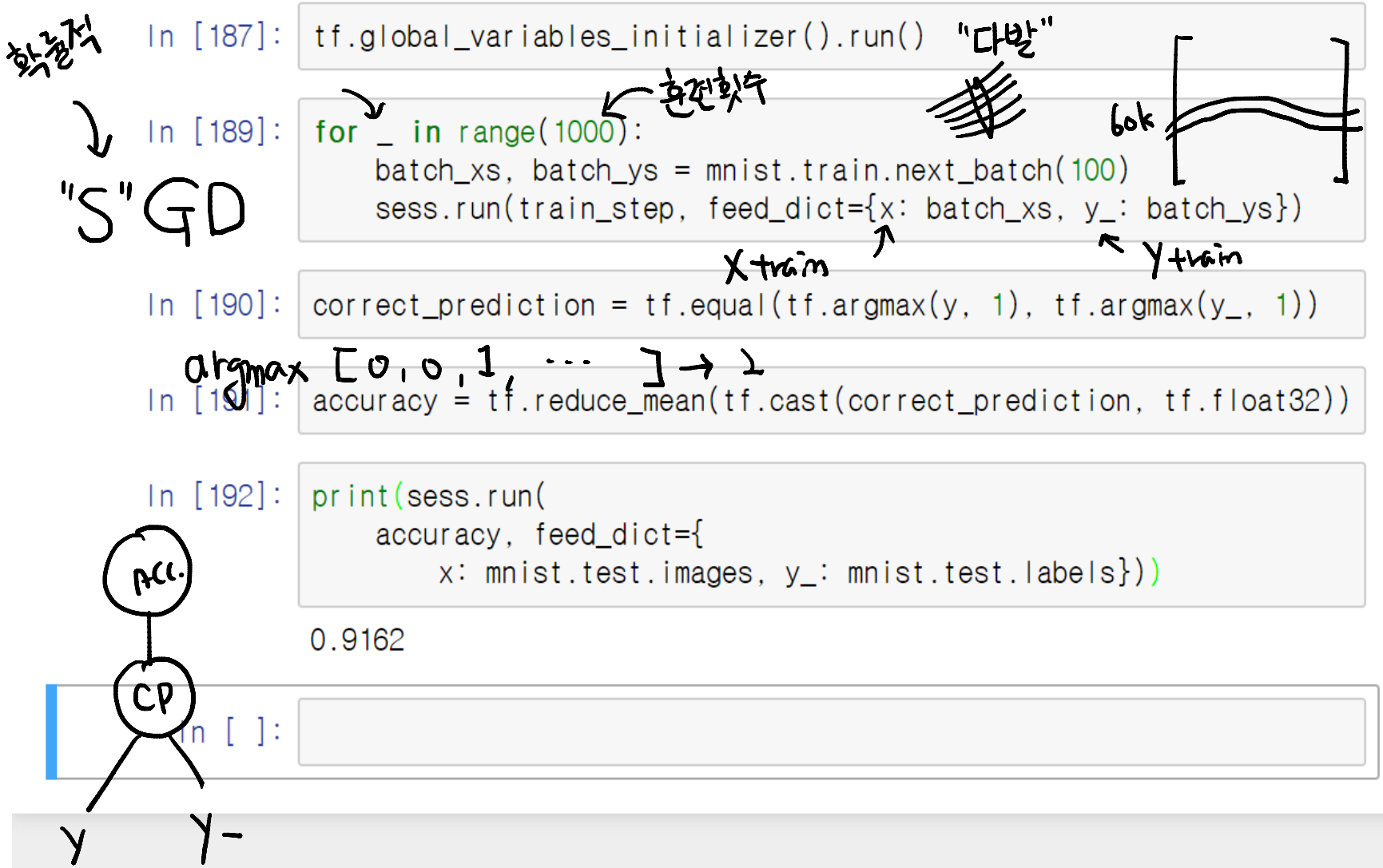
In [190]:
```
correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_, 1))
```

In [191]:
```
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
```

In [192]:
```
print(sess.run(
    accuracy, feed_dict={
        x: mnist.test.images, y_: mnist.test.labels}))
```

0.9162

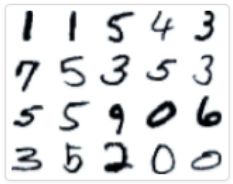
In [ ]:



ILSVRC2012 task 1

MNIST

who is the best in MNIST ?



MNIST 50 results collected

Units: error %

> Classify handwriten digits. Some additional results are available on the original dataset page.

Result	Method	Venue	Details
0.21%	Regularization of Neural Networks using DropConnect	ICML 2013	
0.23%	Multi-column Deep Neural Networks for Image Classification	CVPR 2012	
0.23%	APAC: Augmented PAttern Classification with Neural Networks	arXiv 2015	
0.24%	Batch-normalized Maxout Network in Network	arXiv 2015	Details
0.29%	Generalizing Pooling Functions in Convolutional Neural Networks: Mixed, Gated, and Tree	AISTATS 2016	Details
0.31%	Recurrent Convolutional Neural Network for Object Recognition	CVPR 2015	
0.31%	On the Importance of Normalisation Layers in Deep Learning with Piecewise Linear Activation Units	arXiv 2015	
0.32%	Fractional Max-Pooling	arXiv 2015	Details
0.33%	Competitive Multi-scale Convolution	arXiv 2015	
0.35%	Deep Big Simple Neural Nets Excel on Handwritten Digit Recognition	Neural Computation 2010	Details
0.35%	C-SVDDNet: An Effective Single-Layer Network for Unsupervised Feature Learning	arXiv 2014	
0.37%	Enhanced Image Classification With a Fast-Learning Shallow Convolutional Neural Network	arXiv 2015	Details

99.79%