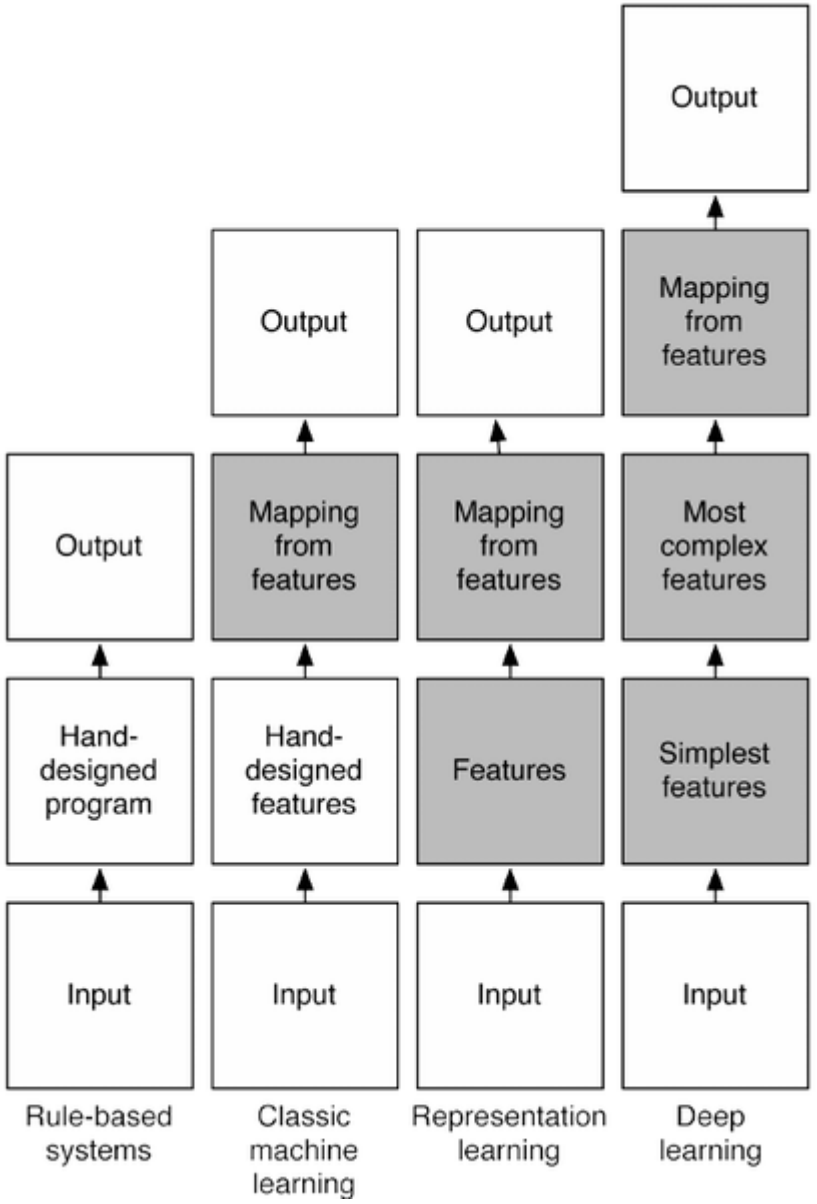
What do MLs learn from sklearn.digits and what do I learn from MLs

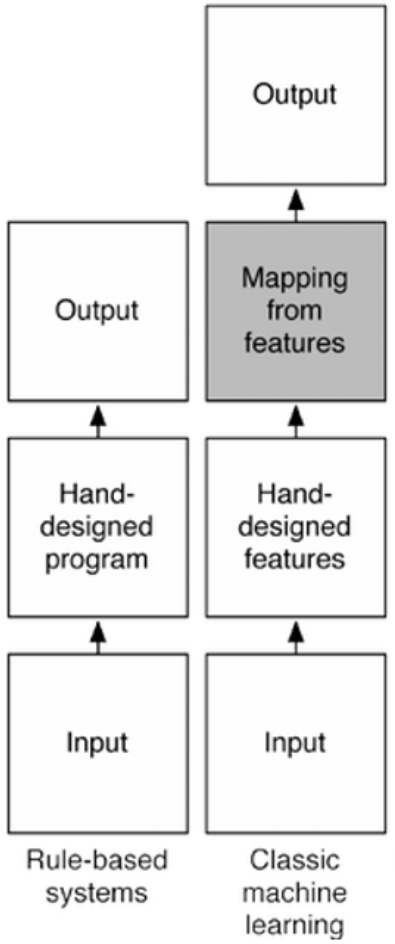
Abner.Zhang

DeepAI

3 types of MLs

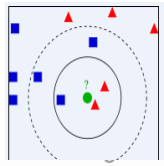


What's Classic ML learnt from Sklearn.digit

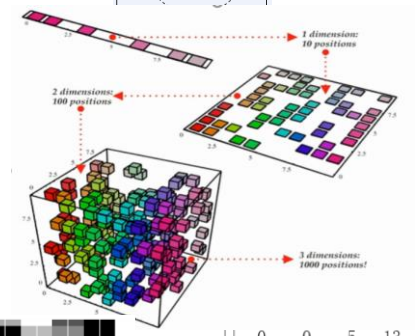


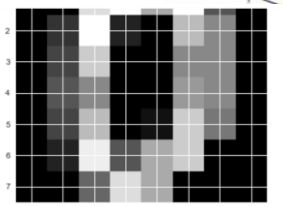
[1. 0. 0. 0. 0. 0. 0. 0. 0. 0.]

KNN

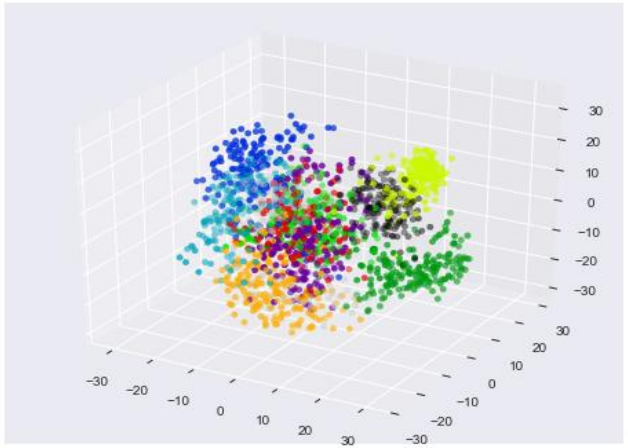


PCA

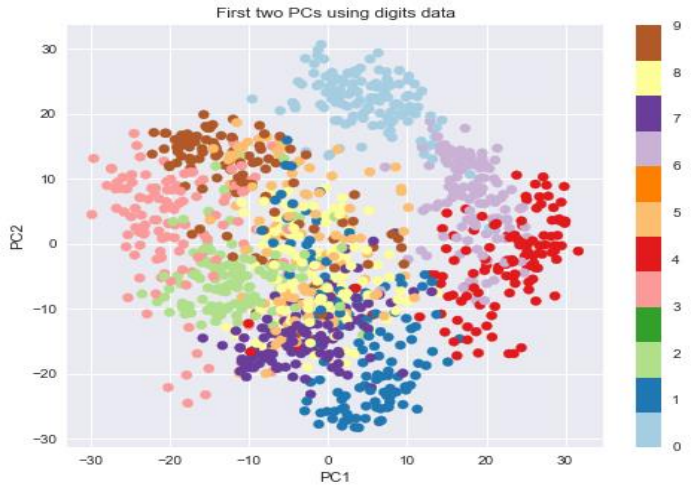




```
[[ 0.  0.  5. 13.  9.  1.  0.  0.]
 [ 0.  0. 13. 15. 10. 15.  5.  0.]
 [ 0.  3. 15.  2.  0. 11.  8.  0.]
 [ 0.  4. 12.  0.  0.  8.  8.  0.]
 [ 0.  5.  8.  0.  0.  9.  8.  0.]
 [ 0.  4. 11.  0.  1. 12.  7.  0.]
 [ 0.  2. 14.  5. 10. 12.  0.  0.]
 [ 0.  0.  6. 13. 10.  0.  0.  0.]]
```

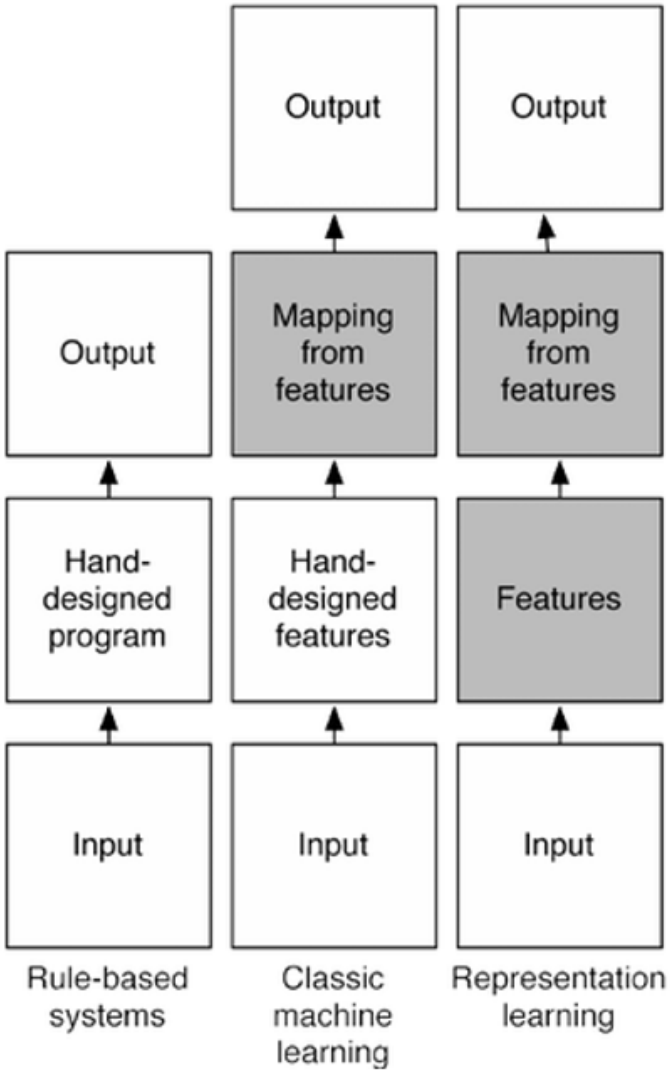


PCA 3D



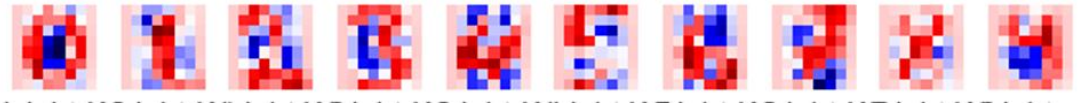
PCA 2D

What's Representation ML learnt from Sklearn.digit

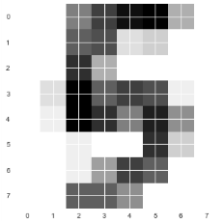


[0.09 0.08 0.06 0.09 0.11 0.18 0.1 0.1 0.08 0.11]

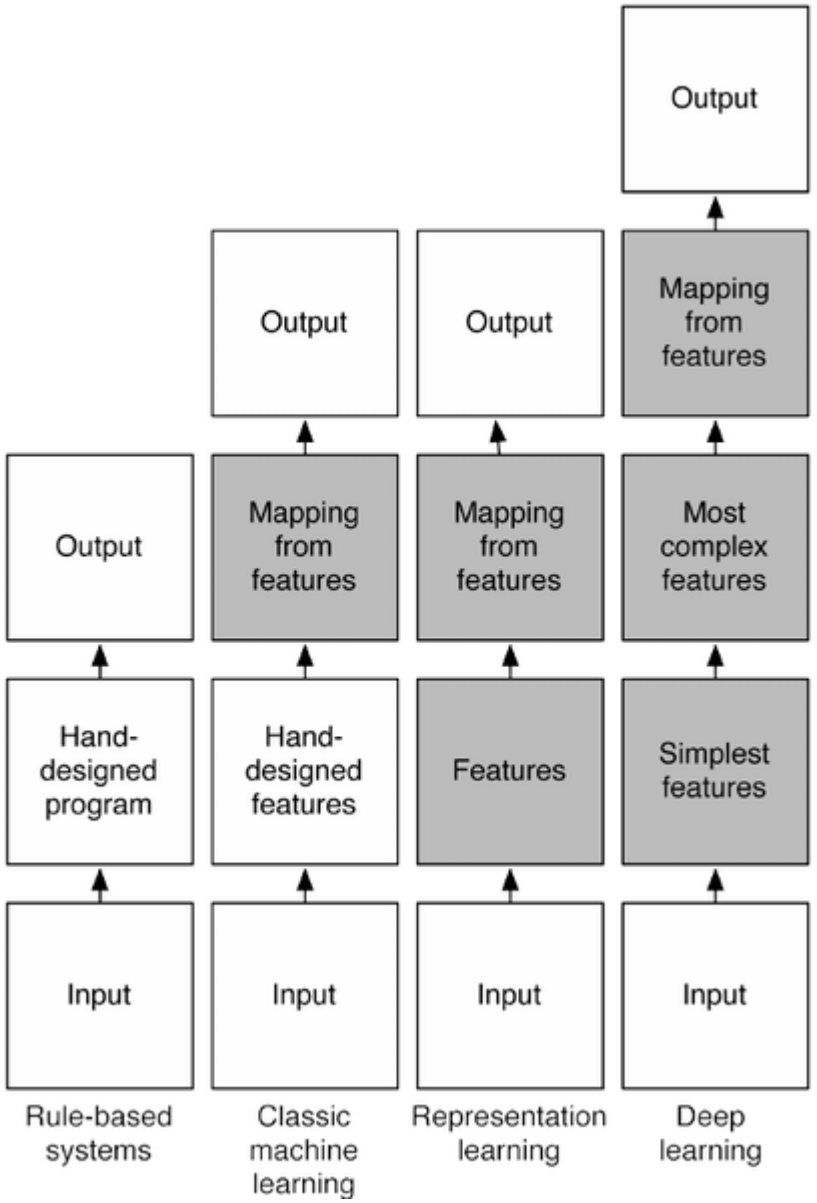
y_pred = tf.nn.softmax(logits)



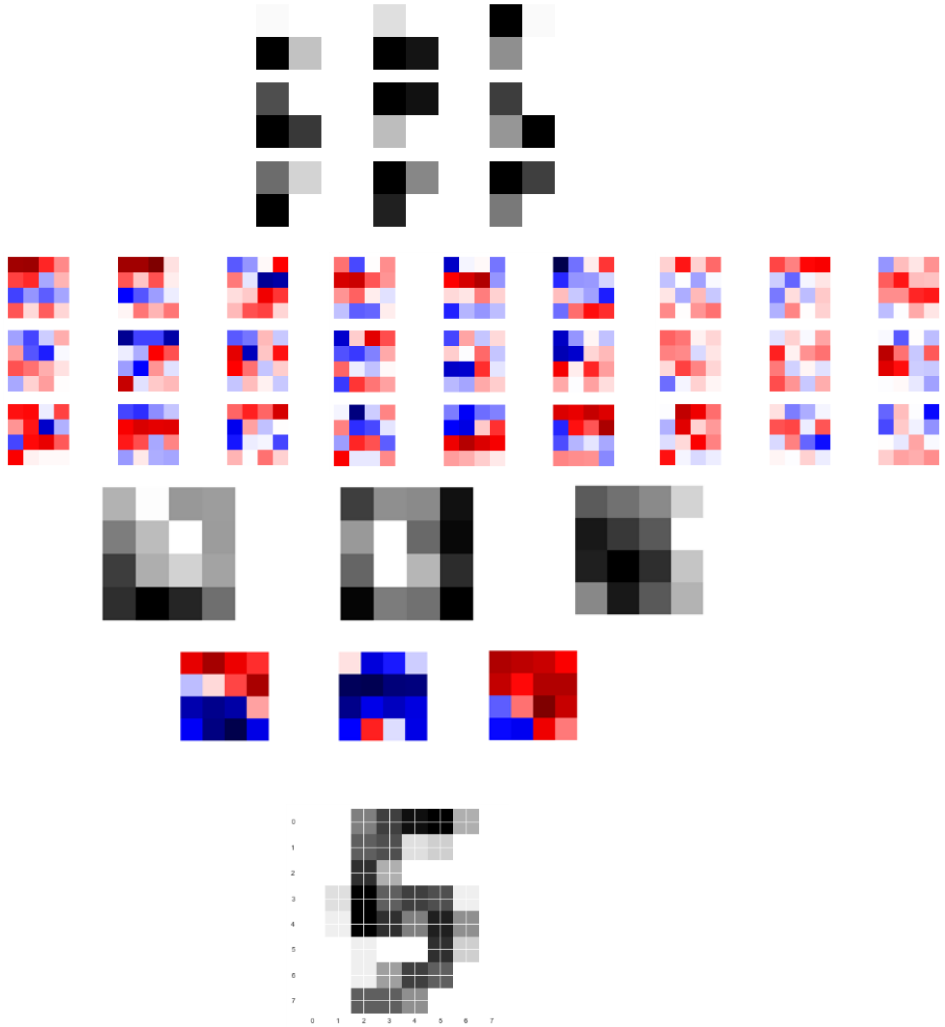
logits = tf.matmul(x, weights) + biases



What's Deep ML learnt from Sklearn.digit



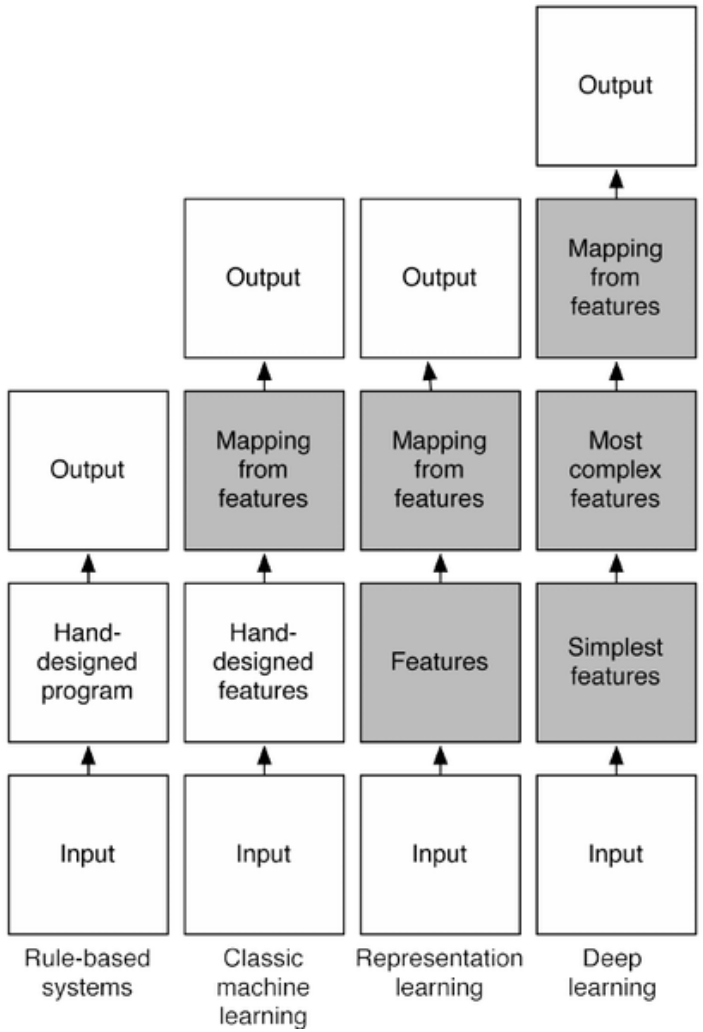
[0.01 0.13 0. 0. 0.21 0.56 0. 0. 0.09 0.]



# Fully-connected layer.
fc_size = 64 .
num_channels = 1

# Convolutional Layer 2.
filter_size2 = 4
num_filters2 = 9
use_pooling=True

# Convolutional Layer 1.
filter_size1 = 4
num_filters1 = 3
use_pooling=True



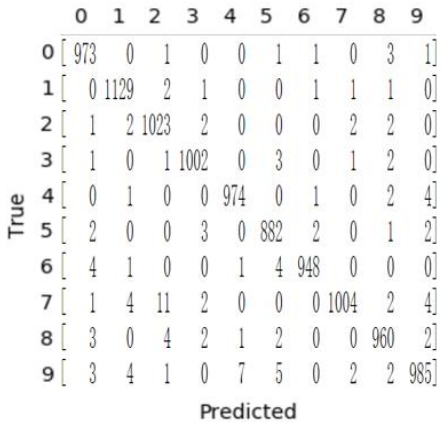
MLs:	PCA+KNN	Fullconn	CNN
Accuracy on test-set:	98.%	86.2%	94.2%
Time usage:	0:01:55	0:01:21	0:13:21

batch_size = 64
num_iterations=100000

What I learn from 3MLs

Translate from Predict score and accuracy to possibility

```
[14  0  0  0  1  0  0  0  0  0]
[ 0 23  0  0  0  0  0  0  0  0]
[ 0  0 26  0  0  0  0  0  0  0]
[ 0  0  0 20  0  0  0  0  0  0]
[ 0  0  0  0 20  0  0  0  0  0]
[ 0  0  0  0  0 14  0  0  0  0]
[ 0  0  0  0  0  0 19  0  0  0]
[ 0  0  0  0  0  0  0 18  0  0]
[ 0  0  0  0  1  0  1  0 22  1]
[ 0  0  0  0  0  1  0  1  0 15]
```



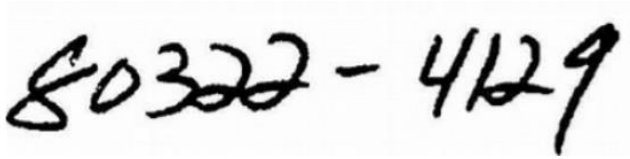
P(Y=0|Y_pred=0 and CNN&MNIST)=973/(973+15)

P(Y!=0|Y_pred=0 and CNN&MNIST)=15/(973+15)

P(Y_pred!=0|Y=0 and CNN&MNIST)=7/(973+7)

	False Negative	False Positive
0	0.0071	0.0152
1	0.0053	0.0105
2	0.0087	0.0192
3	0.0079	0.0099
4	0.0081	0.0092
5	0.0112	0.0167
6	0.0104	0.0052
7	0.0233	0.0059
8	0.0144	0.0154
9	0.0238	0.0130

P(predict correctly)=0.9360



P(predict correctly)=0.8766

What I learn from 3MLs

- Design robust program rather than heavily rely on ML

1. Multi-DataSet: sklearn.digit and MNIST
2. Multi-MLs: Classic, CNN
3. Multi-super parameters for CNN: filter size, channel numbers
4. Detecting adversarial noise
5. Preparing for errors

P(y'|D,x',a)
D=sklearn.digits.train
x' y' = sklearn.digits.test
a=PCA+KNN
a=fullconnected NN
a=CNN

CNN=2layers4*4 and 3+9 channels + fullconnected
Parmeters=random init + SGD 100000 iterations
from sklearn.digits.train

- How to update your MLs when you get a error in real case?

1. Bayesian Neural Network: learn from the wrong predict picture
2. ImageDataGenerator : learn from the variations of the wrong predict picture

- No free lunch and no Silver Bullet