The machine learning 101 and Basic of Tensorflow

Yi-Fan Liou

Agenda

- Course
  - The basic of machine learning
  - Multilayer perceptron
  - Convolutional neural network
- Estimated time
  - 4 hours

About the speaker

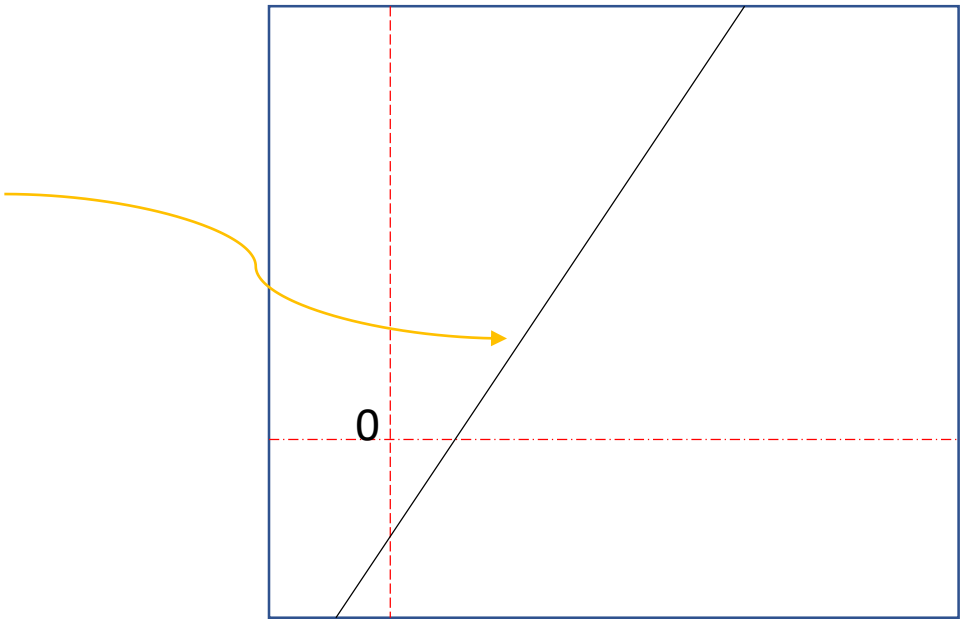
- Yi-Fan Liou
  - You can call me Mark as well
  - I am not a teacher… I don't think I am qualified …
  - Ex-cofounders of many start-up team
  - Confounder of the Taipei Bioinformatics Omnibus (Facebook 社團，歡迎有興趣的人加入)
- Specialist
  - Bioinformatics and Proteomics
    - I know you are not interested in …
  - But … if we only focus on machine learning and GPU using experience
    - Machine learning since 2009
    - CUDA programing since 2011
- Finally
  - This is a casual course, please feel free to ask any questions. (我目前單身，先回答各位)
  - I am not always right. Tell me anything if I am wrong.
  - 很重要所以打中文: 如果我詢問各位問題希望各位舉手時，我只是想了解目前各位的狀況，不會強迫各位發言(如果有人迫切想發言當然也不會阻止)

The Basic

Linear regression

- Can you please google this?
  - Just kidding

$$y = Ax + b$$



Logistic regression – from statistic

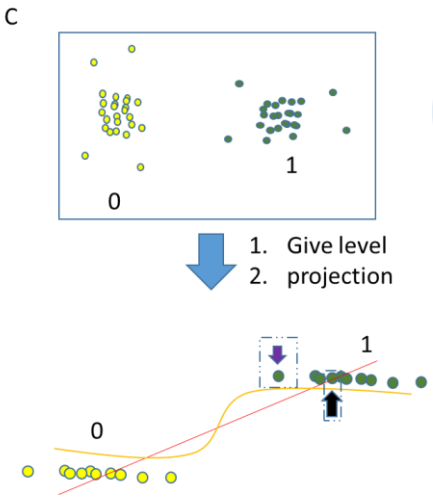
Set: All the dependent variables are linear

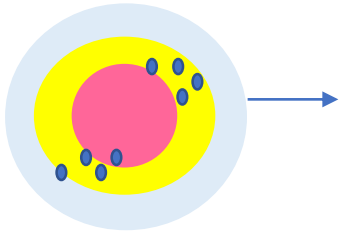
continuous : least square errors

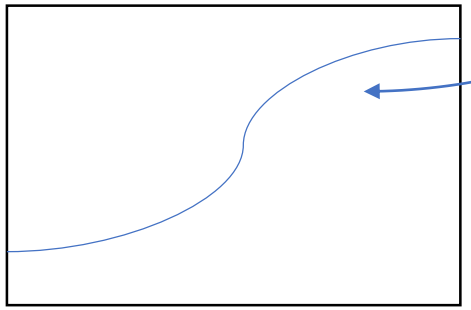
categorical : continuous **with some errors**

Y = a1x1 + a2x2 + …. + anxn + b

Classification case







NOT Normal distribution

Y = a1x1 + a2x2 + …. + anxn + b

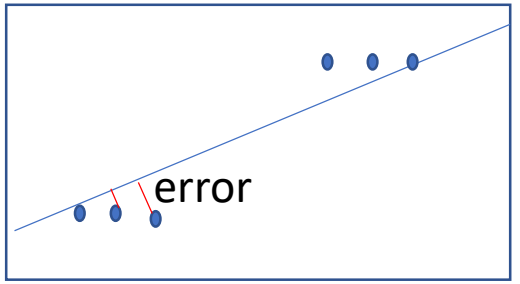
If you want to fit this line ….

This concept also make a straight line
It will be like this ….

It is obviously : the errors between the line and data

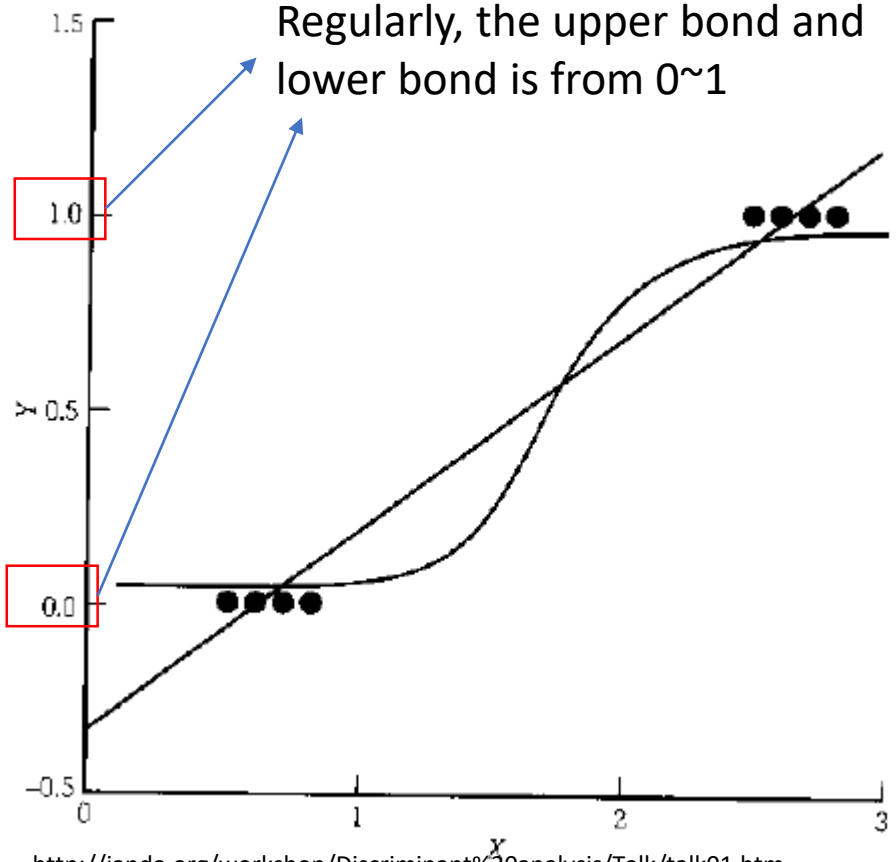
won't be normal distribution

That's why the classification problem never use MSE as loss
(you can use MSE, but it will make a tragedy)



1 or 0

Logistic regression – from statistic



Regularly, the upper bond and lower bond is from 0~1

http://janda.org/workshop/Discriminant%20analysis/Talk/talk01.htm

Why not use “ratio”?

$$log(P/(1-P)) = \sum_{k=i}^{n} a_i x_i + b$$

case

Control
(total samples except the cases)

Odds ratio

According to “maximum likelihood”, using the optimization method can get the odds ratio

1. Newton
2. Gradient decent

Maximum likelihood V.S. cross entropy

- In machine learning, the loss usually use **cross entropy**
- In statistics, the gain usually use **maximum likelihood**
- But Don’t worry, they are the same …

Why logistic regression?

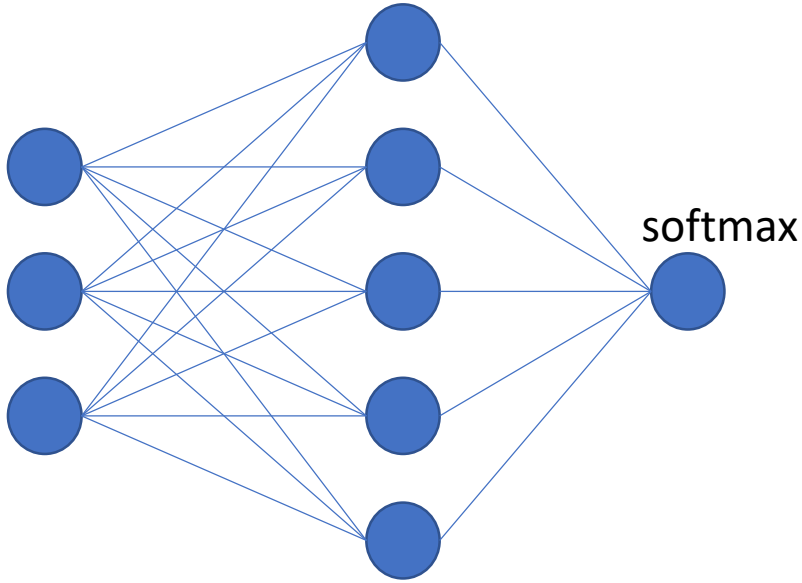
Some concepts in machine learning is also from the statistics.

They are quite similar!!

You can find the logit models in:

1. Traditional statistics – ex. Survival analysis
2. Machine learning

The architecture of Logit model

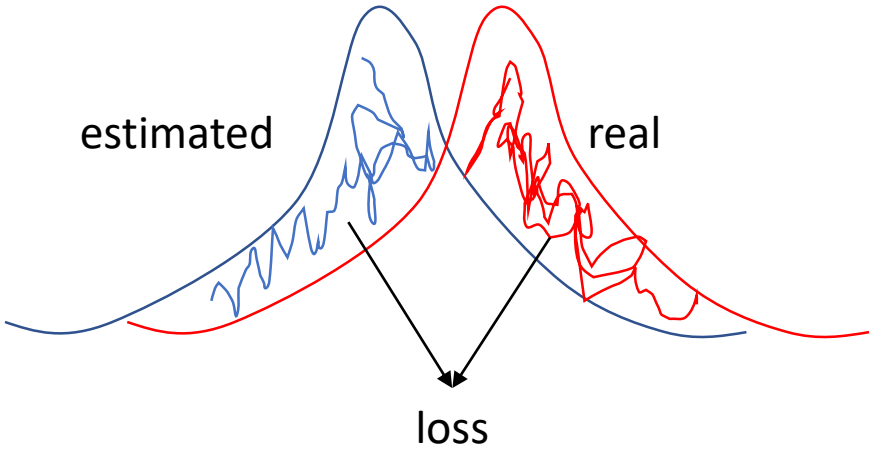


$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{l=0}^{m} w_l x_l = \mathbf{w}^T \mathbf{x}.$$

Basic Terms of machine learning

Loss

- The basic concept of machine learning is to estimate the probability of the dataset.
- We use the probability to predict the unknown ones.
- This term, loss, give the information of how far between the estimated probability and real probability



In category classification task, we usually use "**cross entropy**" to estimate the difference between the distributions.

Estimating the training

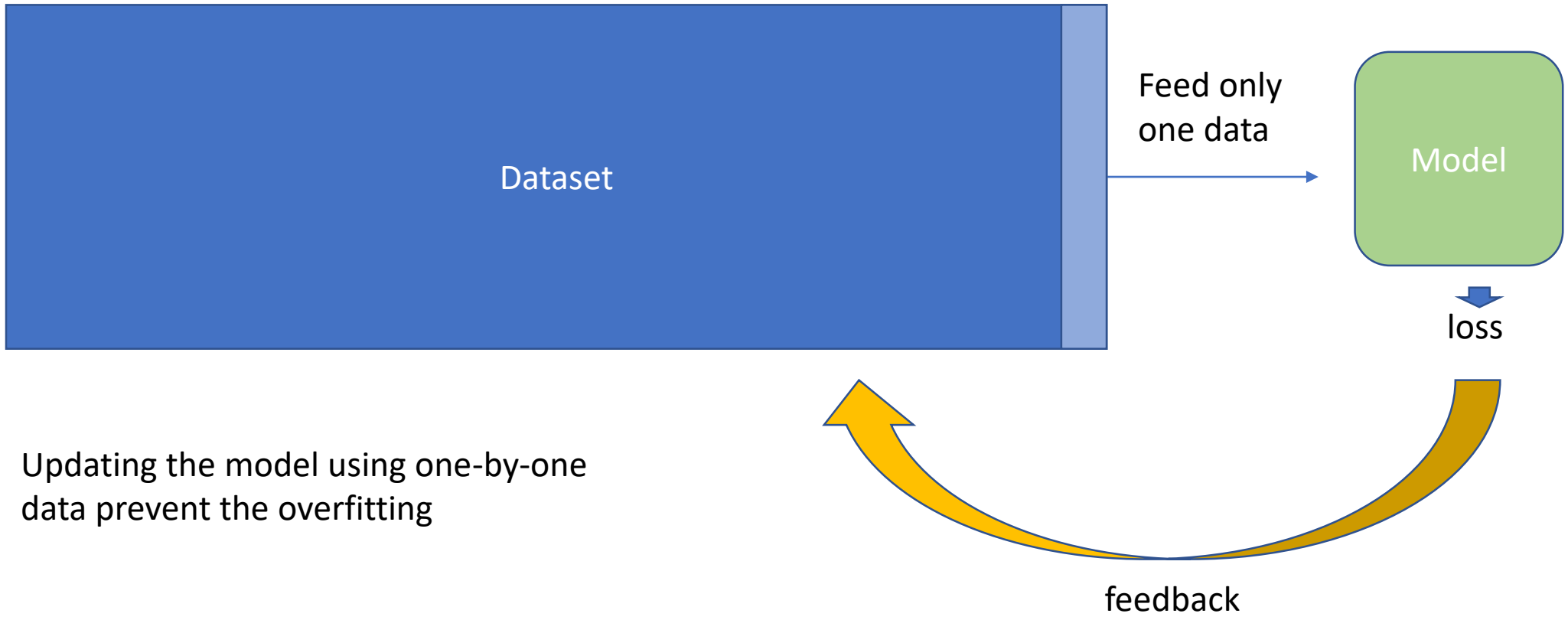
- The training strategies are simply discussed with two domain
  - For “weights” – calculate the loss for directly updating model
    - Batch
    - Epoch
  - For “hyerparamters” – calculating the loss for estimating the performance
    - Validation
    - Corss validation
      - N-fold
      - Leave one out
    - Holeout validation

Updating the weights – whole dataset

This is most intuiting method for updating the weights

Updating the weights – original stochastic method

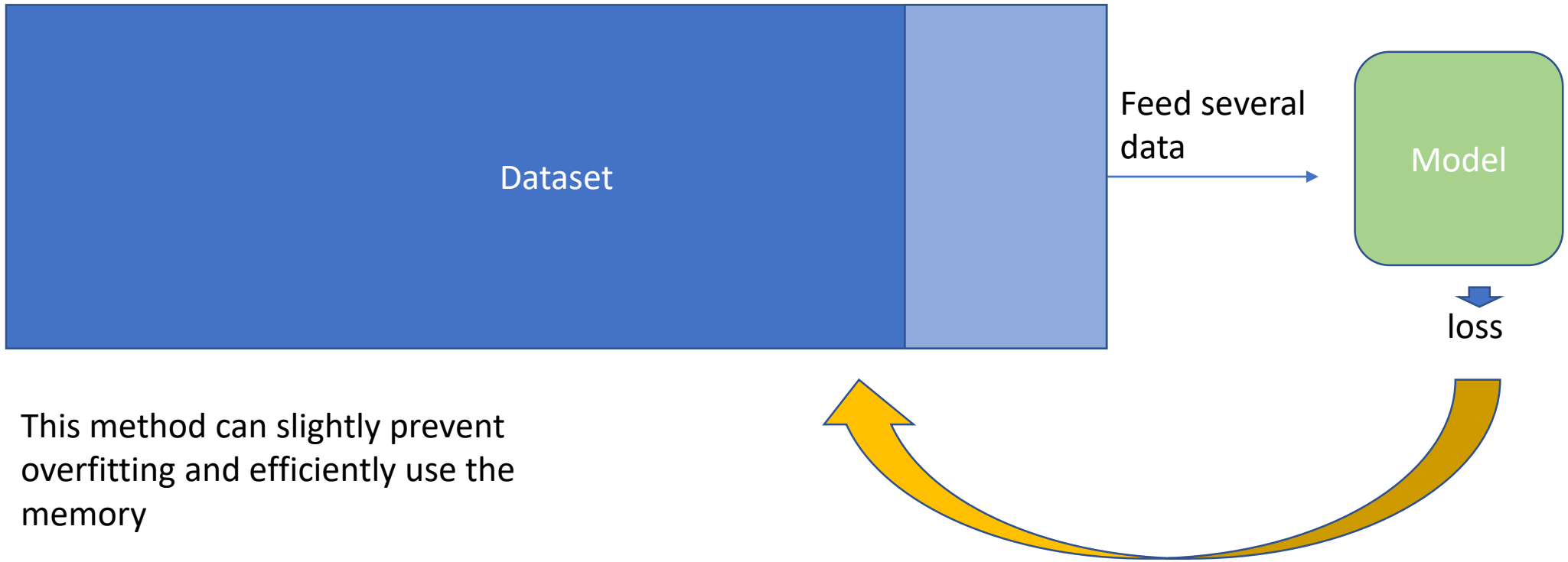
Updating the model using whole dataset cause the “overfitting” issue



Updating the model using one-by-one data prevent the overfitting

Updating the weights – stochastic mini-batch method

Updating the model using stochastic method would be influence by the noise.

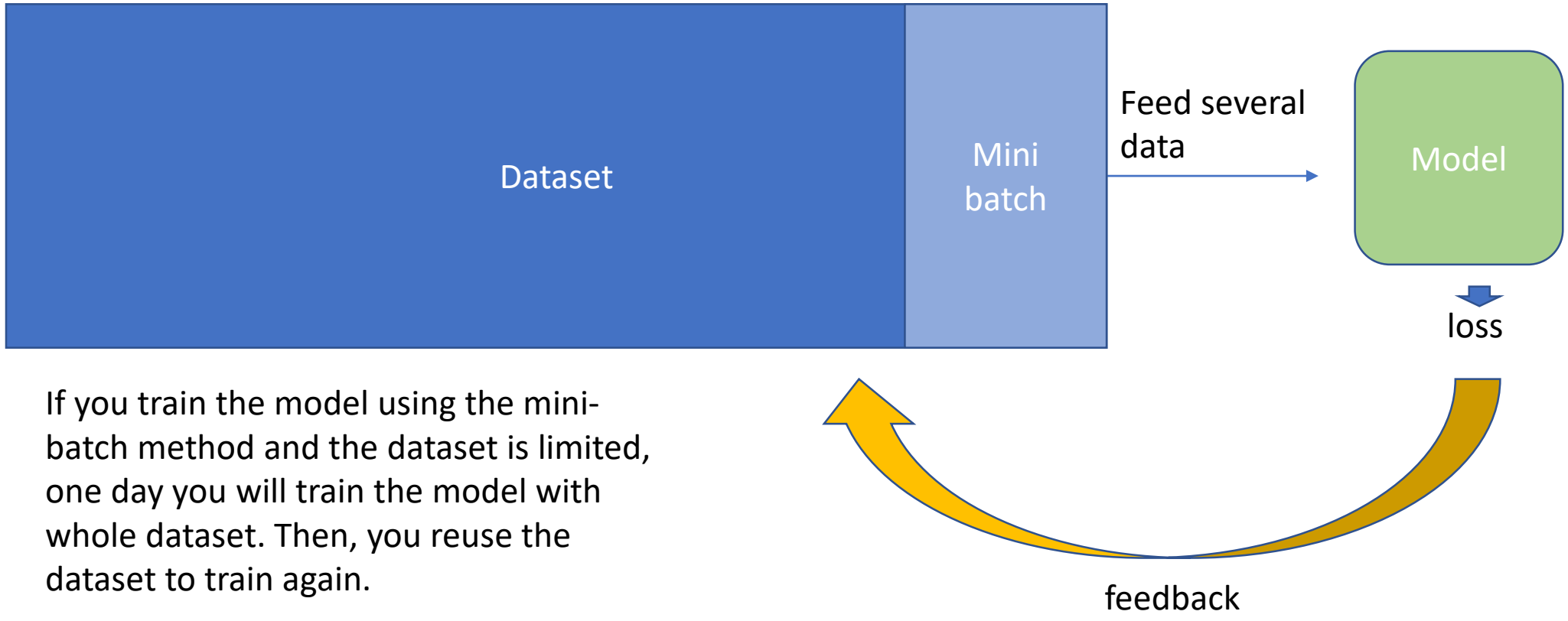


This method can slightly prevent overfitting and efficiently use the memory

This method becomes a “classical” in training model

Batch? Epoch?

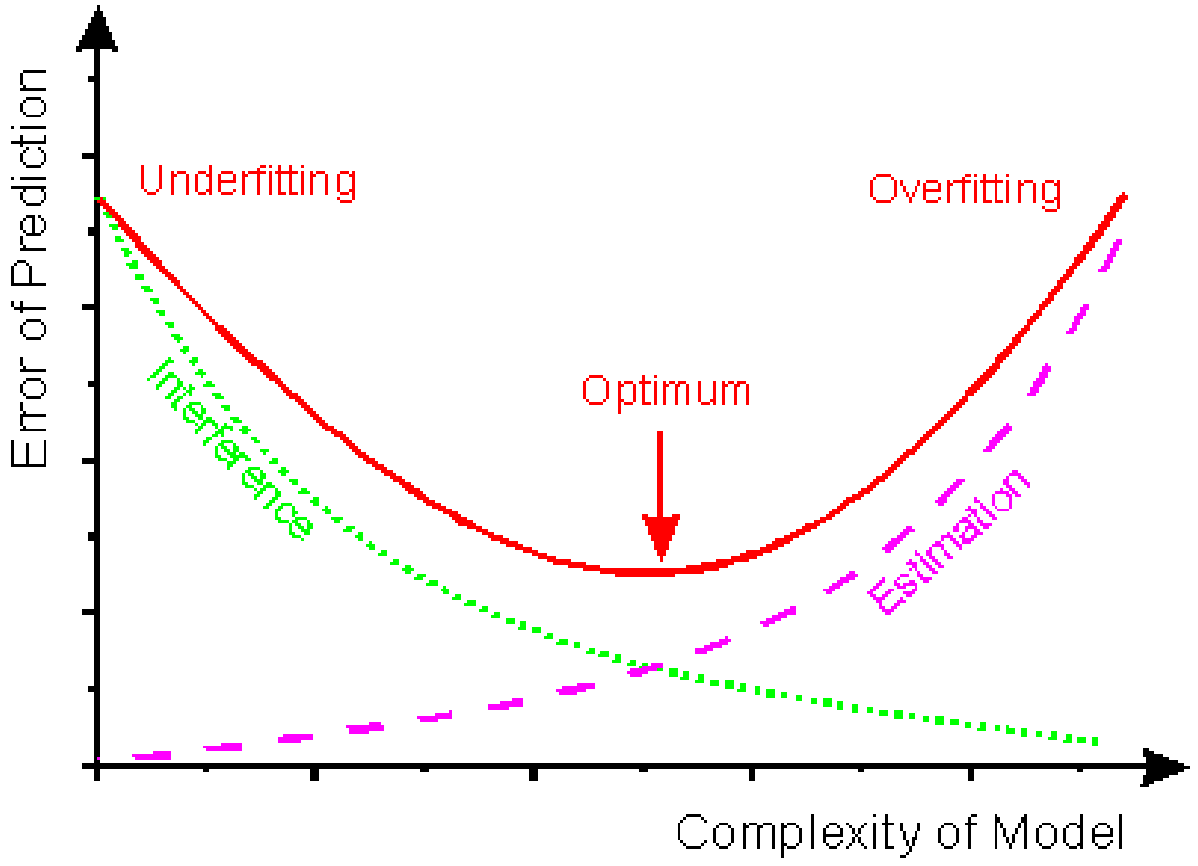
The dataset will feed into the model.
The model will be train again and again



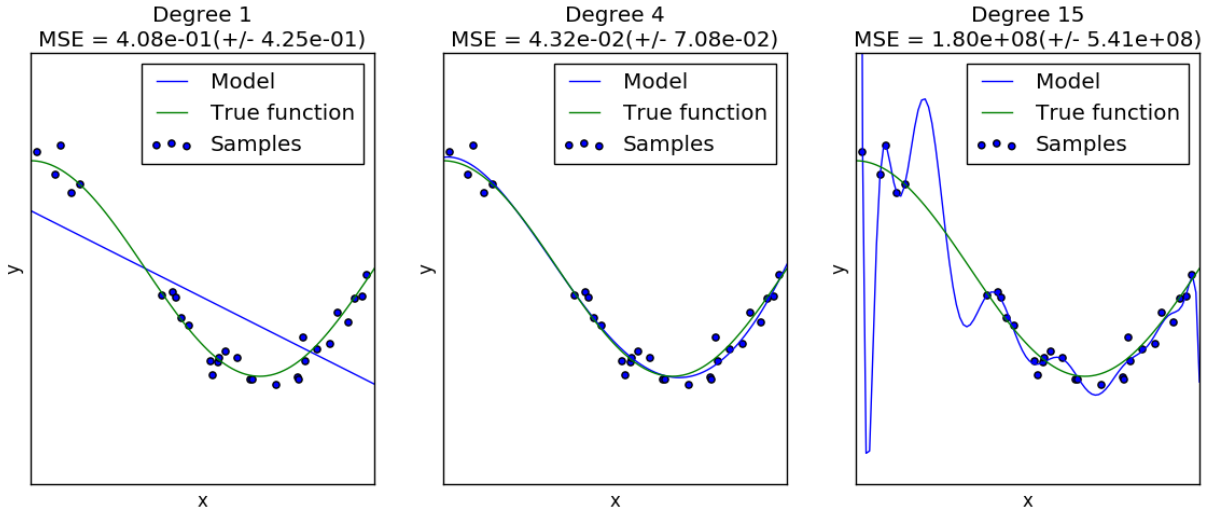
If you train the model using the mini-batch method and the dataset is limited, one day you will train the model with whole dataset. Then, you reuse the dataset to train again.

One cycle mean one epoch

The classic overfitting issue



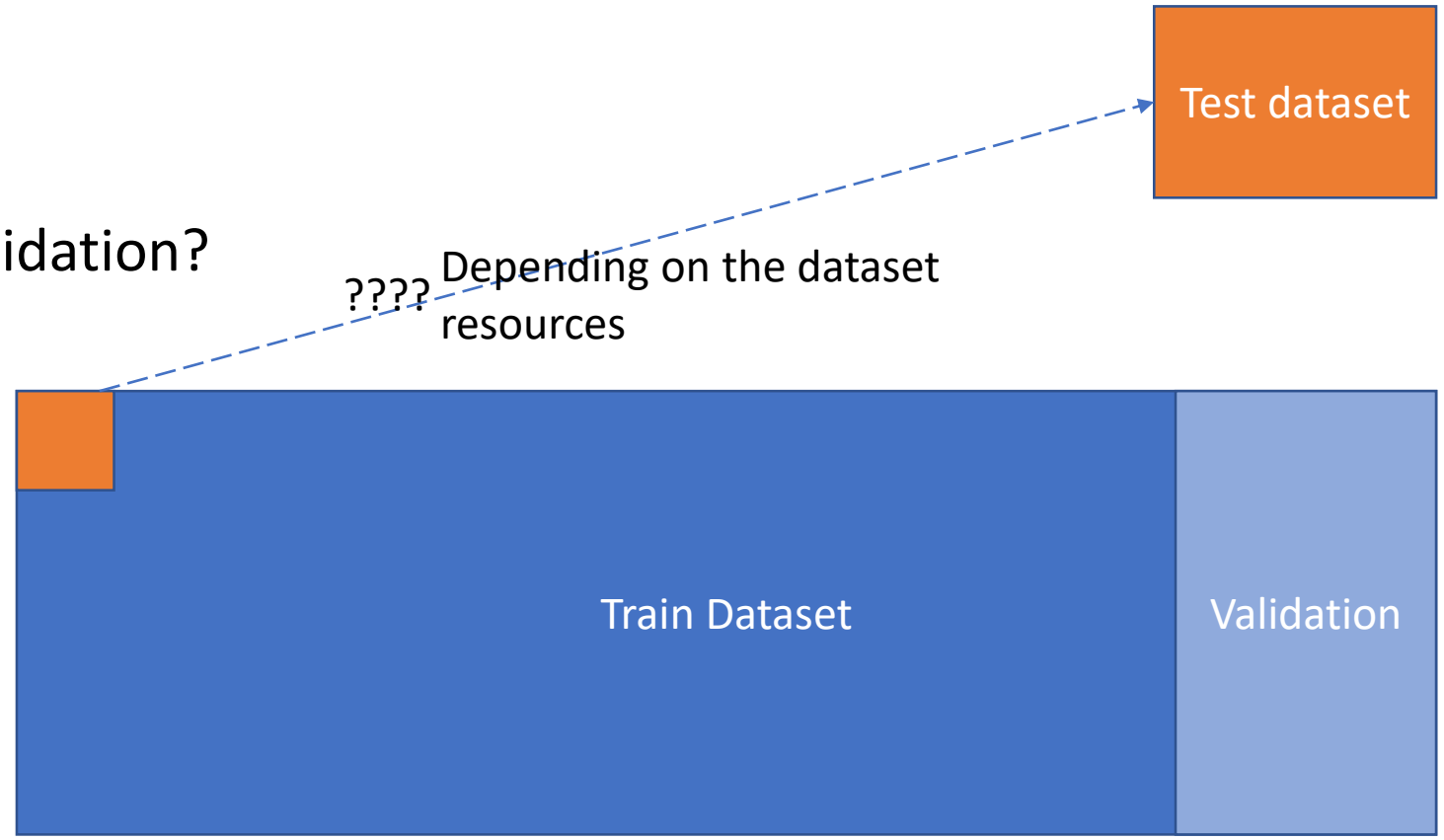
http://www.frank-dieterle.de/phd/images/image024.gif



http://lijiancheng0614.github.io/scikit-learn/auto_examples/model_selection/plot_underfitting_overfitting.html

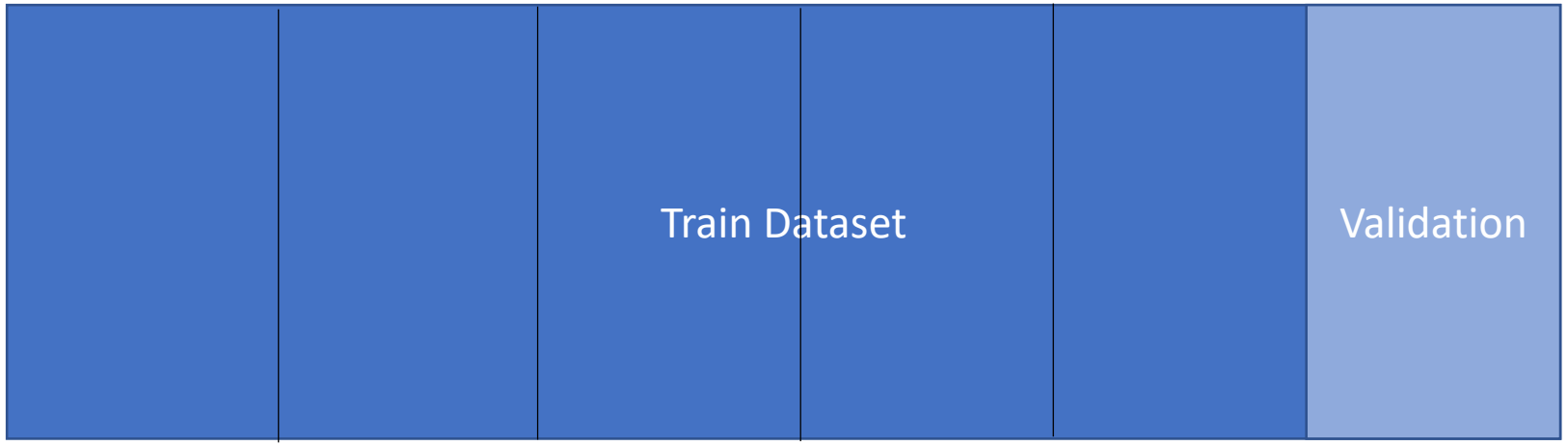
Training, Test, Validation

- Training
- Test
- Validation
  - Why validation?



More about validations

- holdout validation
  - Most use in modern model, especially with “early-stopping”
- Cross-validation
- Leave-one-out cross validation



Cross validation in deep learning ?

Yoshua Bengio, My lab has been one of the three that started the deep learning approach, back in 2006, along with Hinton's...

Answered Jan 20, 2016 · Upvoted by Naran Bayanbat, MSCS with focus in machine learning and Boxun Zhang, Data Scientist at Spotify; PhD in Computer Science · Author has **174** answers and **3.9m** answer views

We mostly have large datasets when it is not worth the trouble to do something like k-fold cross-validation. We just use a train/valid/test split. Cross-validation becomes useful when the dataset is tiny (like hundreds of examples), but then you can't typically learn a complex model.

41k views · View 242 Upvoters

https://www.quora.com/Is-cross-validation-heavily-used-in-deep-learning-or-is-it-too-expensive-to-be-used

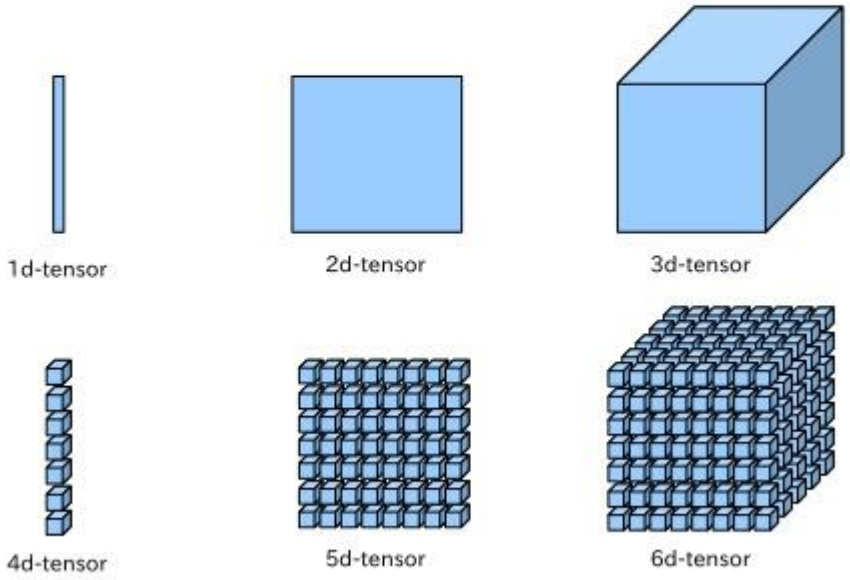
The basic of the tensors

Using Numpy and Tensorflow

What is “tensor” ?

What's Tensor

Tensor is a general name of multi-way array data. For example, 1d-tensor is a vector, 2d-tensor is a matrix and 3d-tensor is a cube. We can image 4d-tensor as a vector of cubes. In similar way, 5d-tensor is a matrix of cubes, and 6d-tensor is a cube of cubes.



February 2, 2012 7/26

“A tensor is a generalization of vectors and matrices to potentially higher dimension”

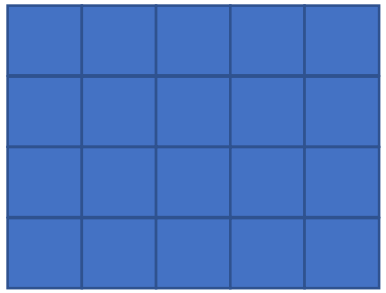
--https://www.tensorflow.org/programmers_guide/tensors

https://stackoverflow.com/questions/37849322/how-to-understand-the-term-tensor-in-tensorflow

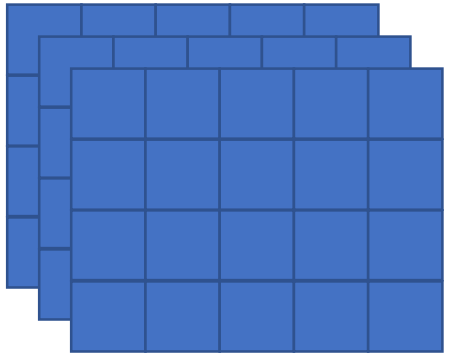
The shape of tensor

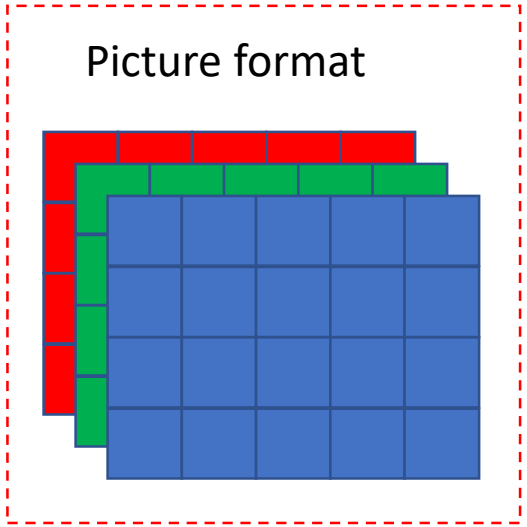
[5]

[4,5]



[3,4,5]





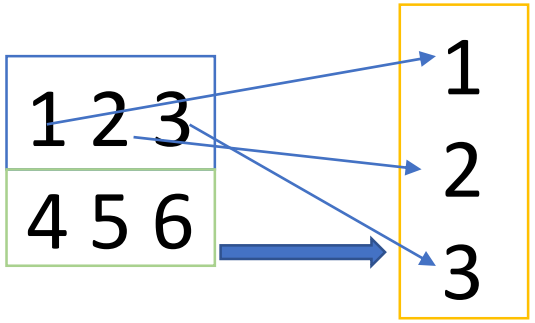
The format:(NHWC)
(batch size, height, width, channel)

- `data_format`: A string, one of `channels_last` (default) or `channels_first`. The ordering of the dimensions in the inputs. `channels_last` corresponds to inputs with shape `(batch, height, width, channels)` while `channels_first` corresponds to inputs with shape `(batch, channels, height, width)`. It defaults to the `image_data_format` value found in your Keras config file at `~/.keras/keras.json`. If you never set it, then it will be "channels_last".

https://www.tensorflow.org/api_docs/python/tf/keras/layers/Conv2D

tf.matmul

This is same to weighted sum



1 * 1 + 2 * 2 + 3 * 3 = 14

4 * 1 + 5 * 2 + 6 * 3 = 32

14
32

```
a = tf.Variable([[1,2,3],[4,5,6]])
b = tf.Variable([[1,2,3]])
c = tf.matmul(a,b)
```

1 2 3
4 5 6

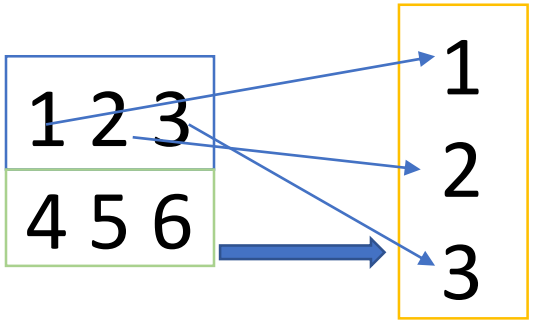
1 2
3 4
5 6

22 28
49 64

What's the answer?

```
a = tf.Variable([[1,2,3],[4,5,6]])
b = tf.Variable([[1,2],[3,4],[5,6]])
c = tf.matmul(a,b)
```

Weighted sum

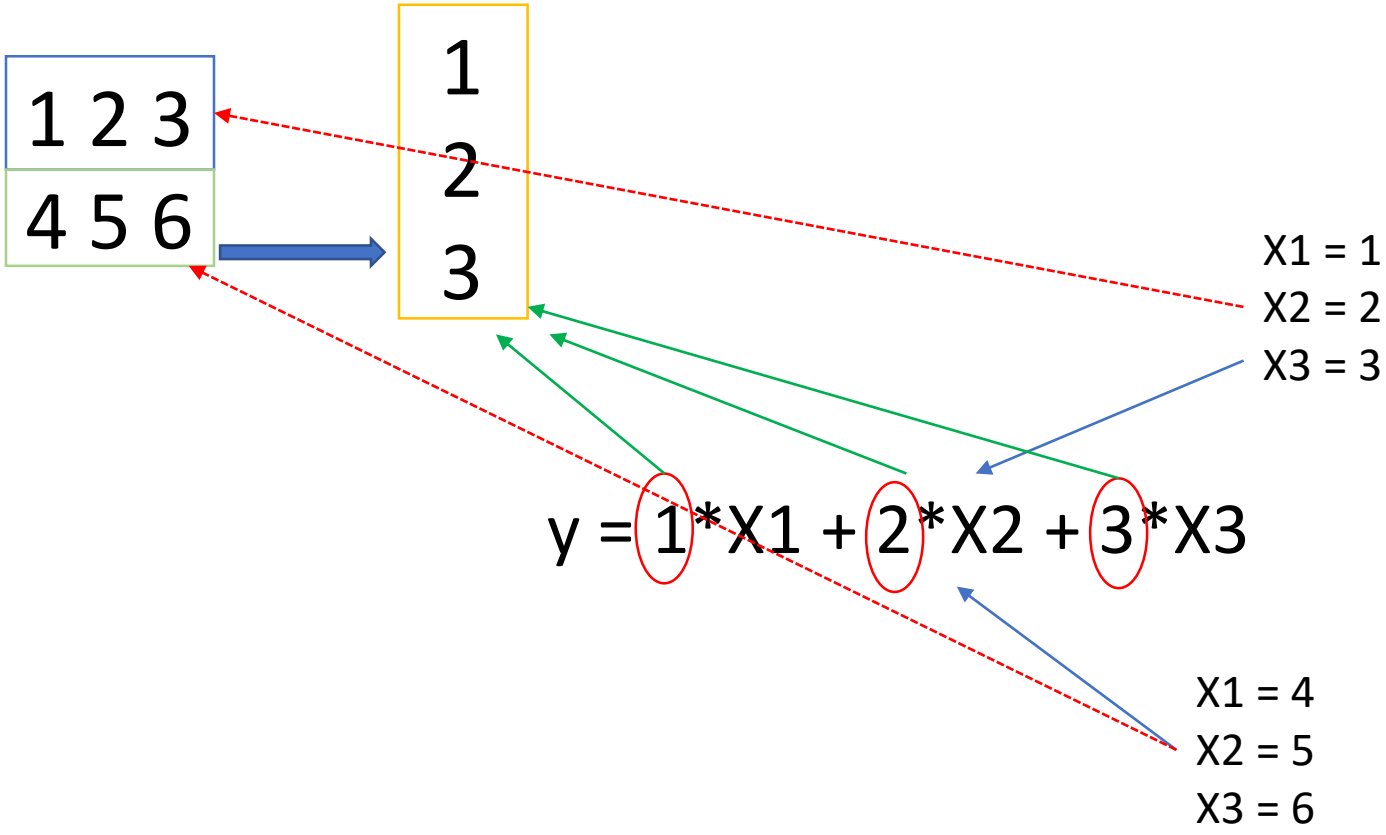


X1 = 1
X2 = 2
X3 = 3

$$y = 1*X1 + 2*X2 + 3*X3$$

X1 = 4
X2 = 5
X3 = 6

Weighted sum

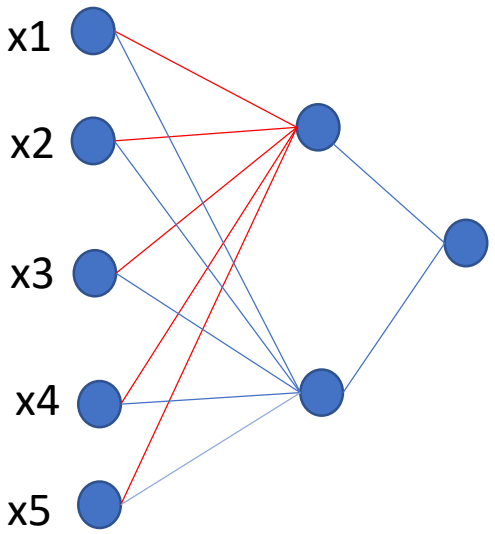


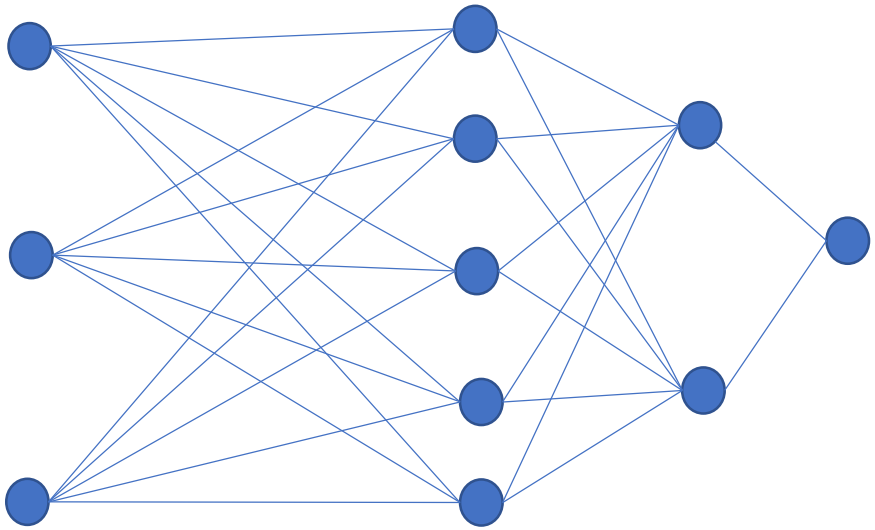
How weighted-sum used in neural network

Assume we have a simple linear function:

$$F(x) = 2.5\ X_1 + 1.4\ X_2 + 0.2\ X_3 + 0.2\ X_4 + 4.5\ X_5$$

$$f(x) = 25\ X_1 + 14\ X_2 + 2\ X_3 + 2\ X_4 + 45\ X_5$$



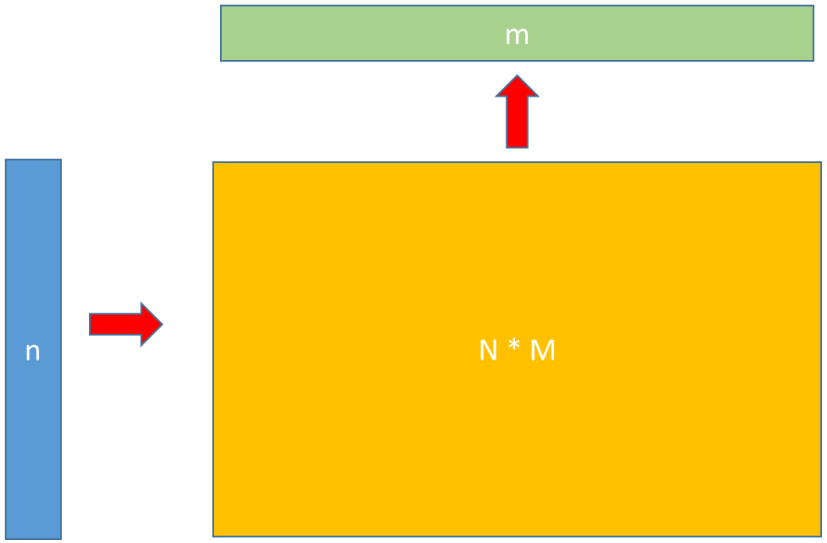


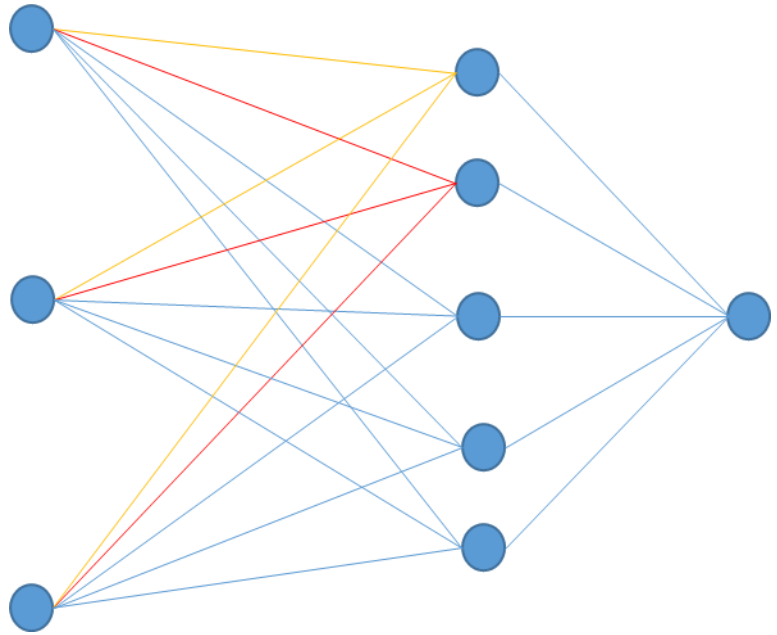
0mutmul_sample.py

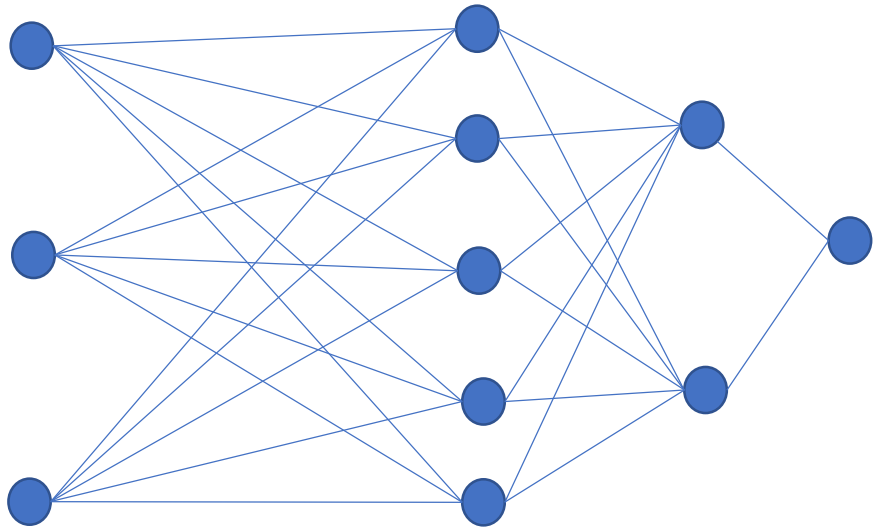
Another words

Use MNIST as example.

Make a MLP network and optimize it using gradient decent







But… How to train it??

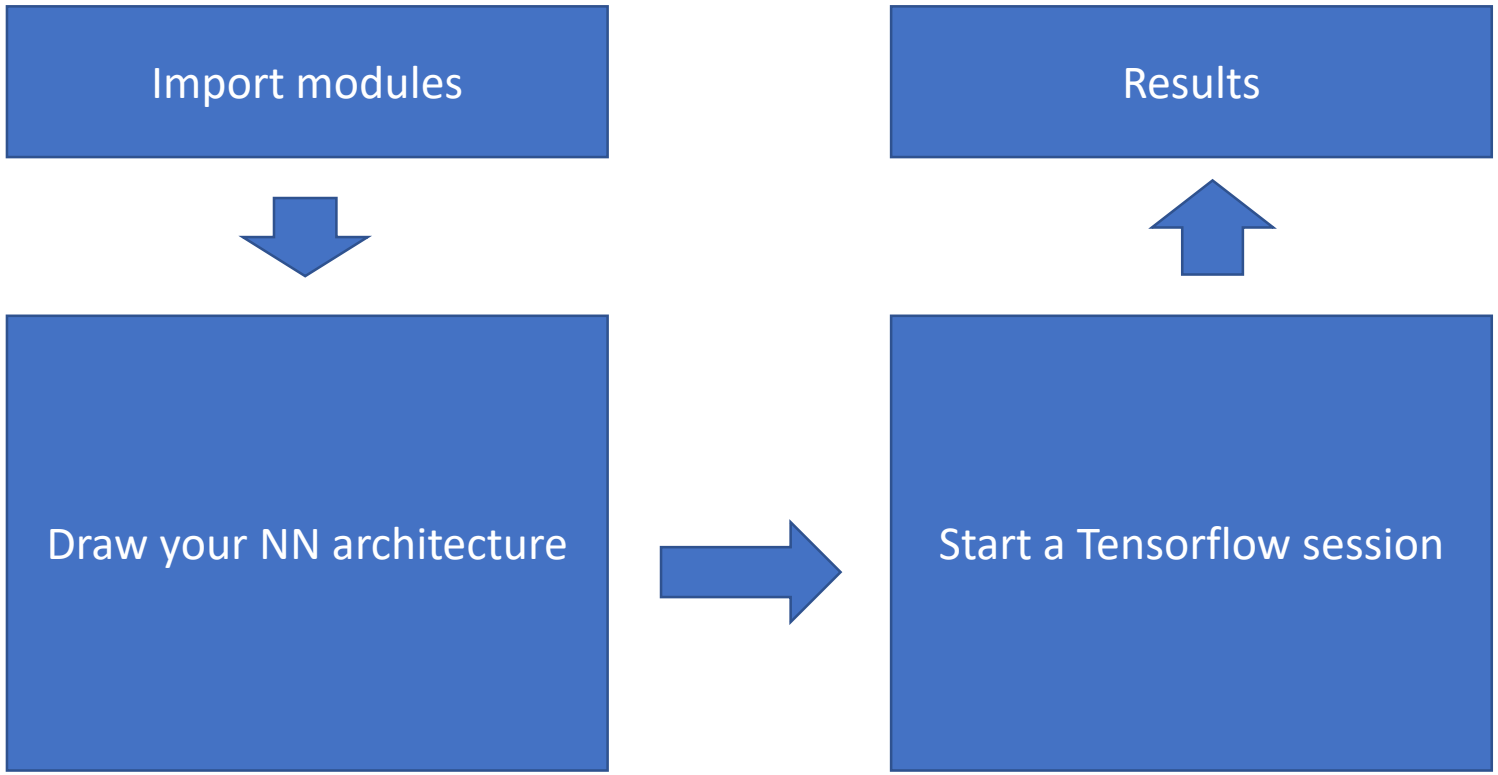
I will try to explain it just use the essential math concept.

http://bimay.pixnet.net/blog/post/11410360-%E8%AA%AA%E8%A9%B1%E7%9A%84%E8%97%9D%E8%A1%93

The basic concept of Tensorflow

- Static computation graph
  - You need to define the whole graph before you run a complete process
  - The framework will deploy your job to the corresponding computing nodes
- By the way
  - Dynamic computational graph
    - pyTorch
    - Eager execution

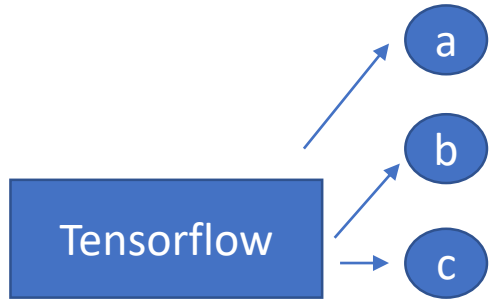
Process diagram



Session is a “canvas”

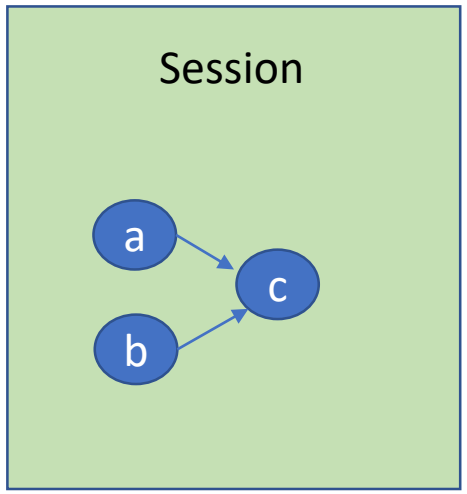
Import tensorflow as tf

a = tf.Variable([1,2,3])
b = tf.Variable([4,5,6])
c = a + b



They are just occupy in the python

sess = tf.session()
sess.run(c)
sess.close()



Tip:

1. Once the Session announced, the architecture fixed.
2. Once the operators announced, it will live until you close Python (advance)
3. Since the architecture is fixed, we usually use ‘ delivery pores’ for feeding the data.

The basic operation in tensor calculating

We will compare the basic operator between python and Tensorflow

- Since python did not have an exact ‘array’, we use the “list” to instead

add

tf.add(x,y)

multiply

tf.matmul(x,y)

tf multiply(x,y)??

How about “weighted sum” ?

tf.reduce_sum(tf.matmul(x,y))

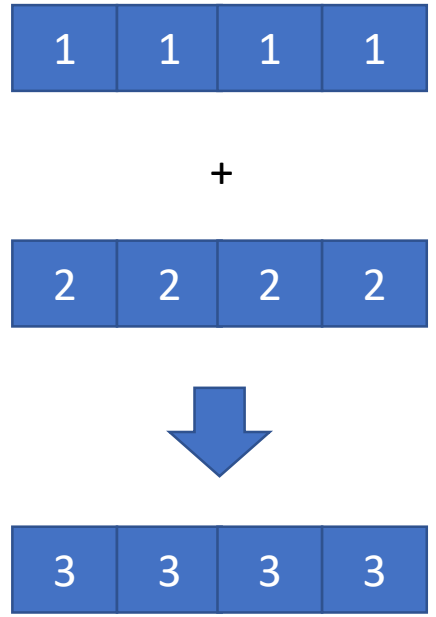
Example 1.

Task : we use Tensorflow to complete some simple operation

1. Add 2 tensors
2. Multiply 2 tensors
3. Division 2 tensors

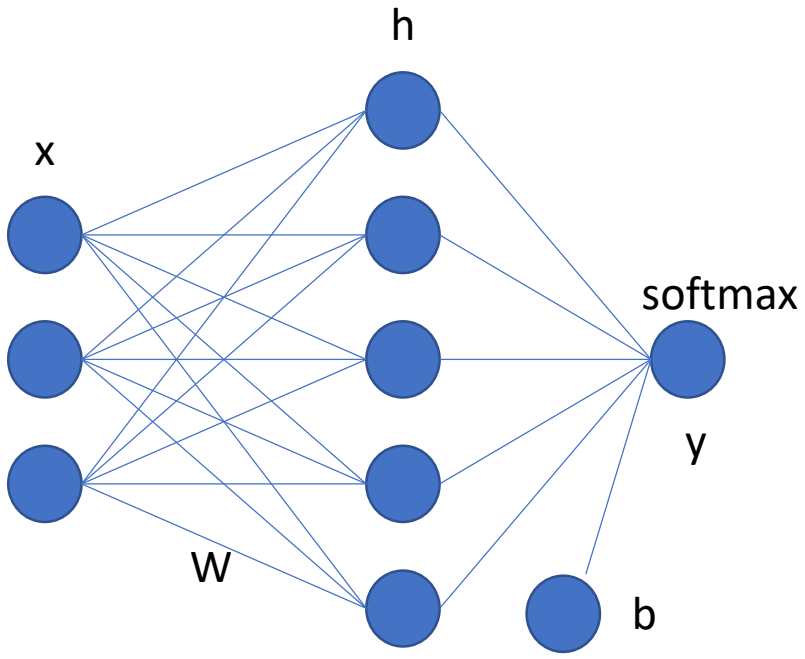
tf.add

```
>>> a = tf.Variable([1,1,1])
>>> b = tf.Variable([2,2,2])
>>> c = tf.add(a,b)
>>> sess = tf.Session()
2018-06-07 16:21:08.059528: I C:\tf_jenkins\workspace\re
.cc:137] Your CPU supports instructions that this Tensor
>>> c
<tf.Tensor 'Add:0' shape=(3,) dtype=int32>
>>> sess.run(tf.global_variables_initializer())
>>> sess.run(c)
array([3, 3, 3])
>>>
```



Try to make a model with Tensorflow

The objective



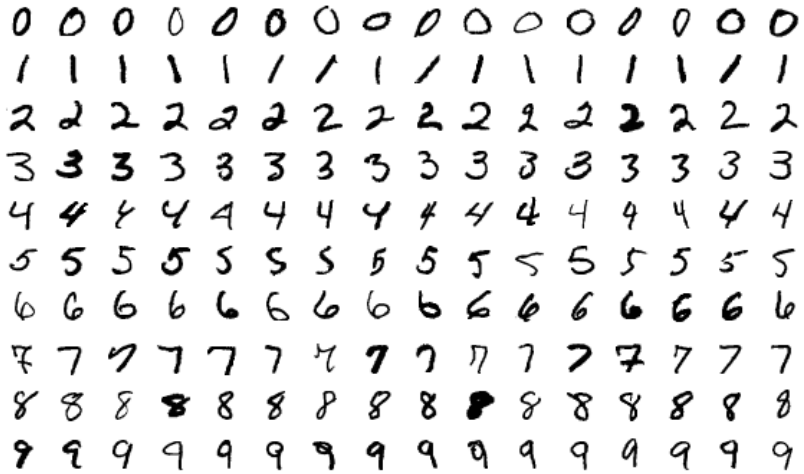
$$z = w_0 x_0 + w_1 x_1 + \ldots + w_m x_m = \sum_{i=0}^{m} w_i x_i = \mathbf{w}^T \mathbf{x}.$$

Find them!!

Image that …

If we get some error in the function, can we modify the coefficients of the function to get more correct results?

The most method is stochastic gradient decent …



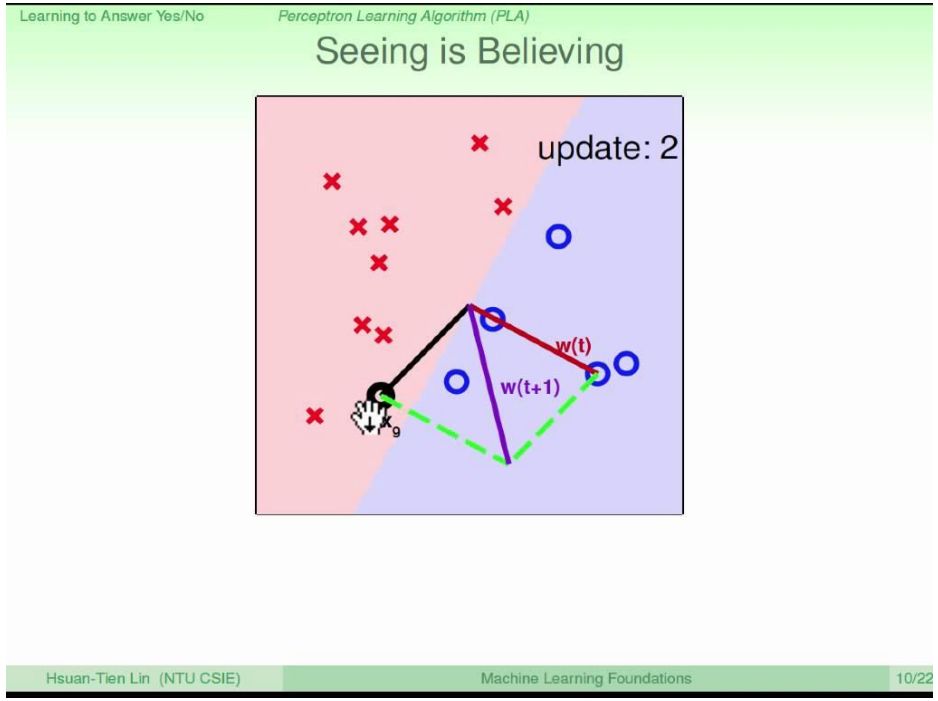
Only 1 channel with the width and height of 28 and 28, respectively.

1softmax_regression_MNIST.py

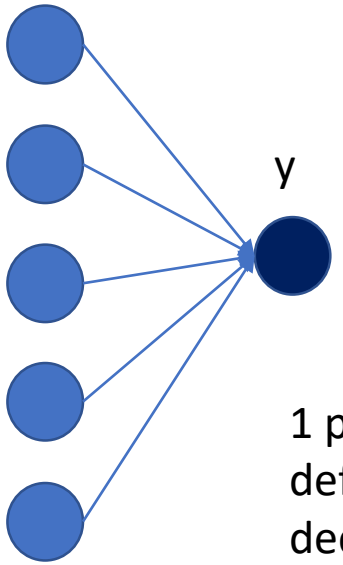
Multilayer perceptrons

Small story – Perceptron learning

Target: We want to use a simple line as decision boundary



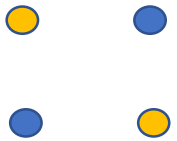
H.T. Lin, “Machine Learning Foundations”



1 if y >0
-1 if y <0

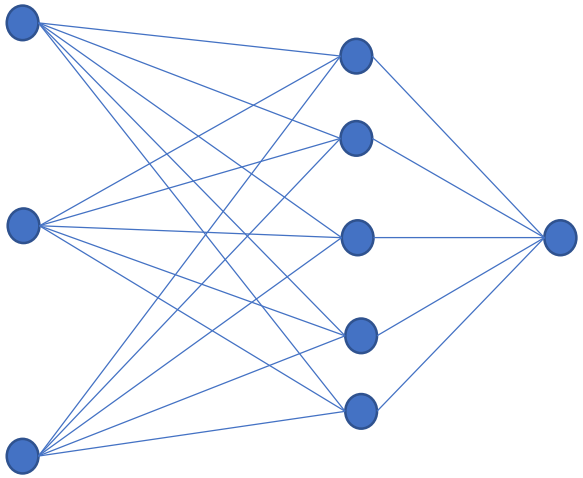
1 perceptron can define a simple decision boundary.

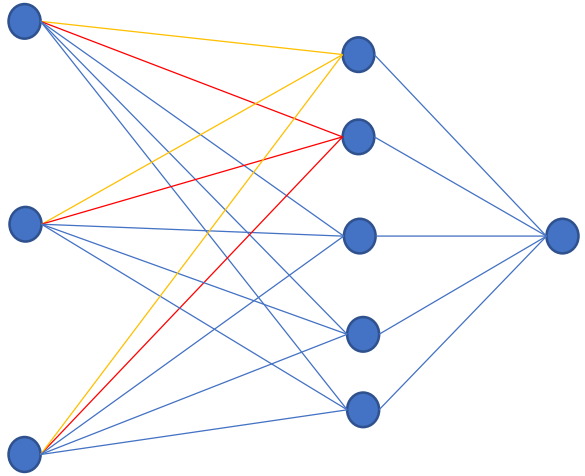
XOR problem

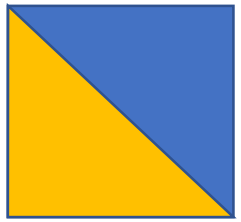


"Perceptrons: An Introduction to Computational Geometry", Minsky and Papert

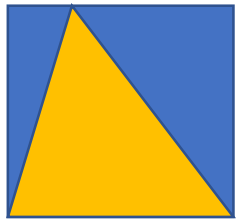
Multi-layer perceptron





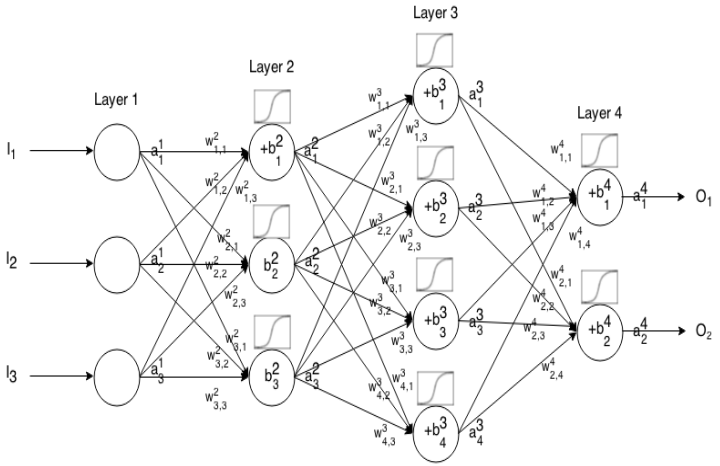


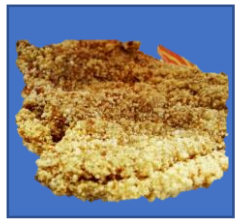
N = 1



N = 2

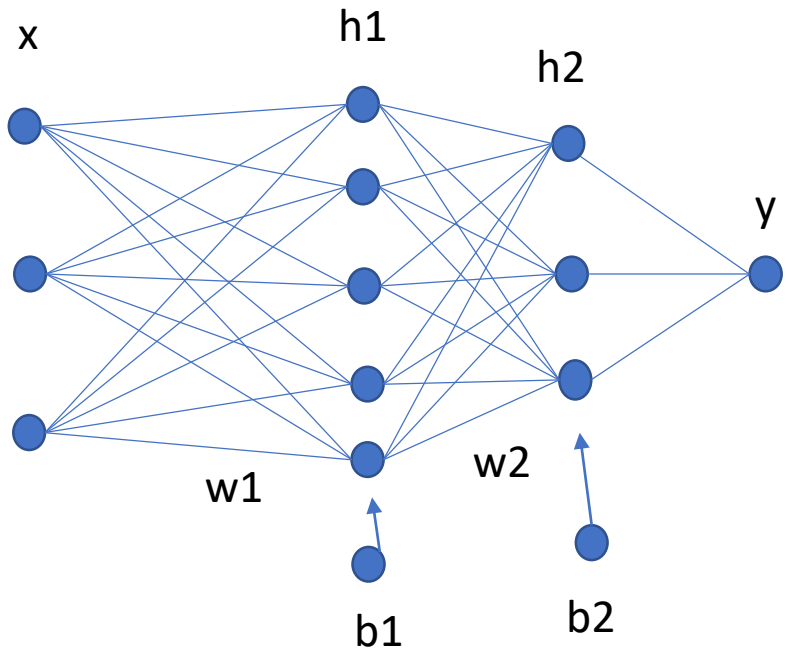
Weighted sum

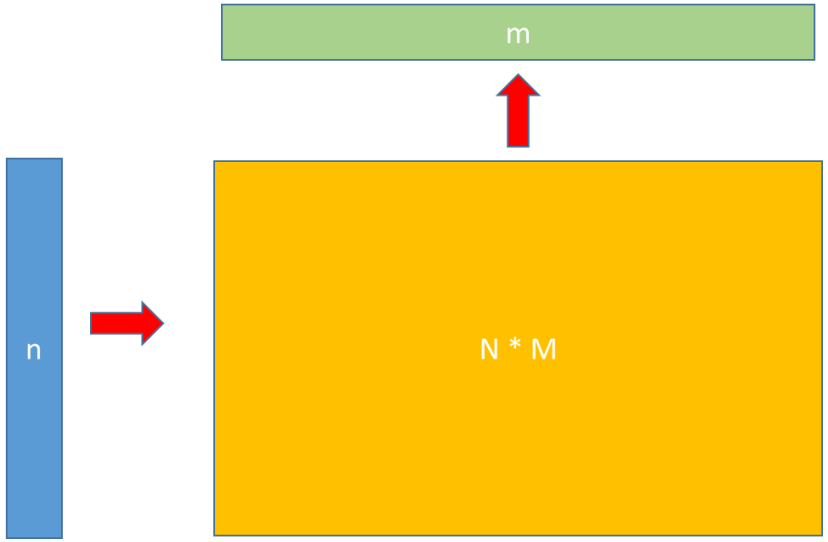




N = m

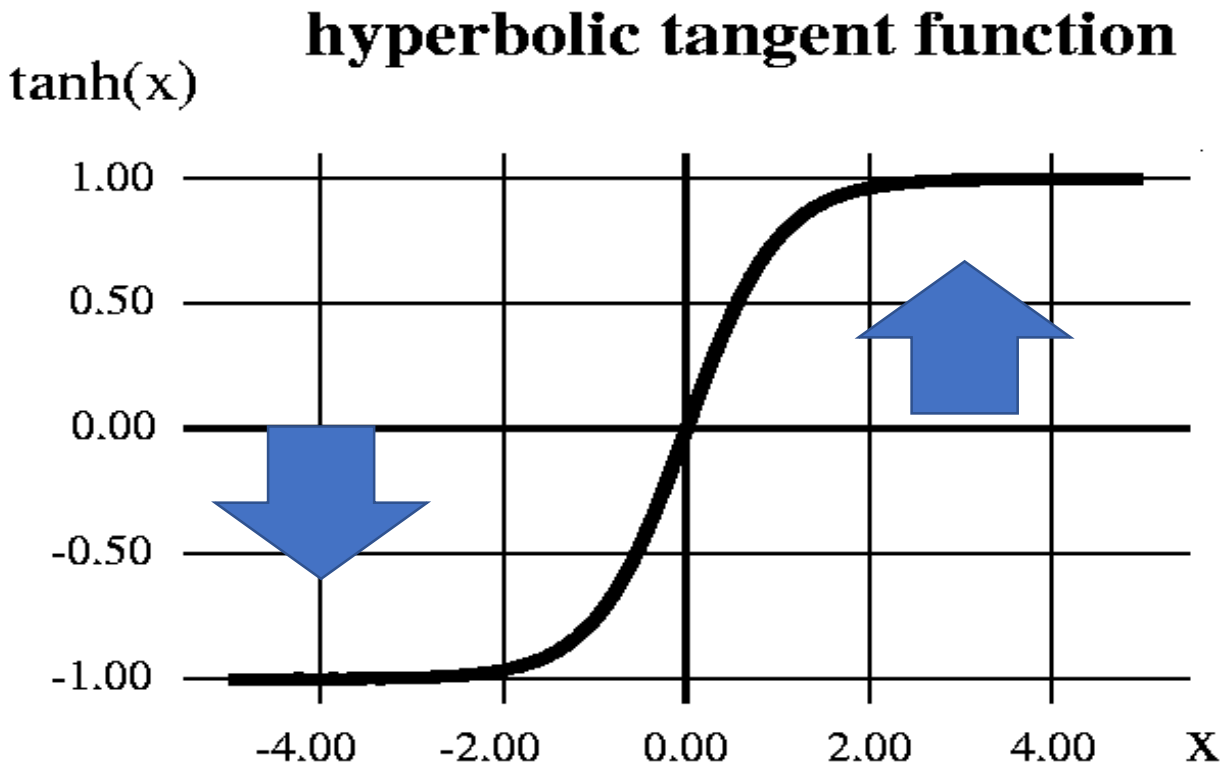
MLP practice





2MLP_MINST.py

The activation function



https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f

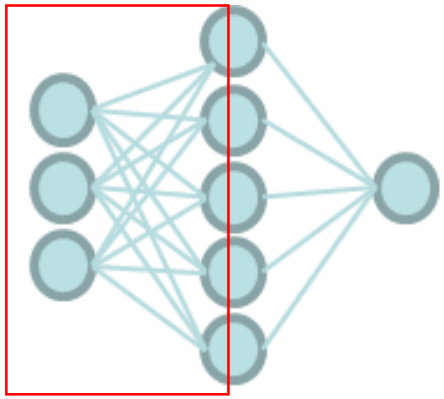
2sparseMLP_MINST.py

Demo time~

- Many interesting things happened. Many people cannot answer them using current math theories.

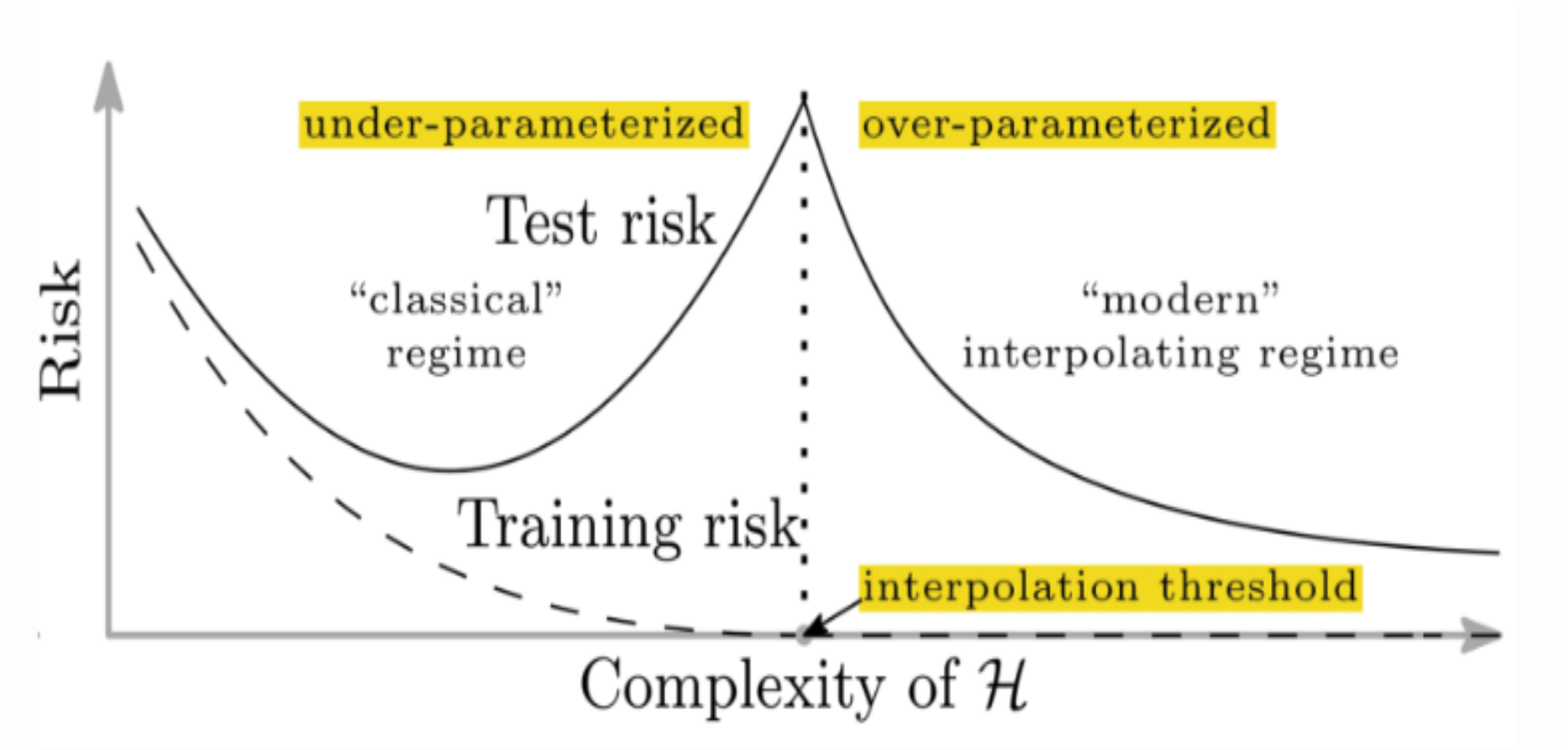
https://github.com/Bcondux/WorkShopCentSci20180609/blob/master/SampleCode/multilayer_perceptron_overfit.py

matmul :



Basic architecture

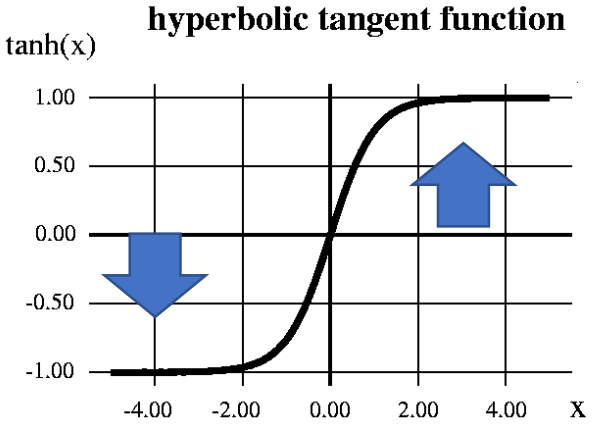
Overparameterization – The update



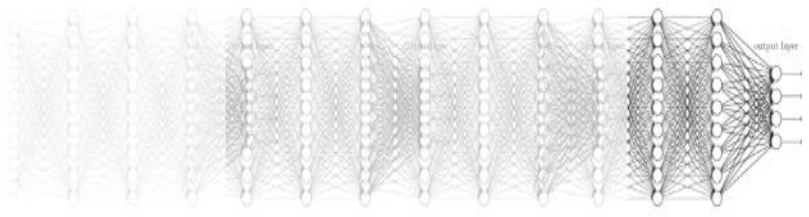
https://arxiv.org/abs/1812.11118

How hard will deeper models be?

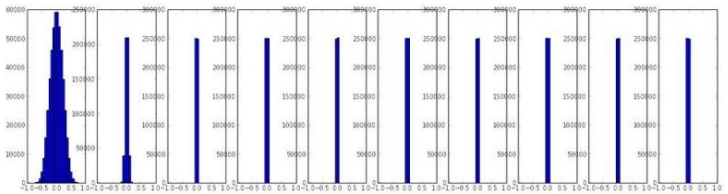
- Gradient vanishing



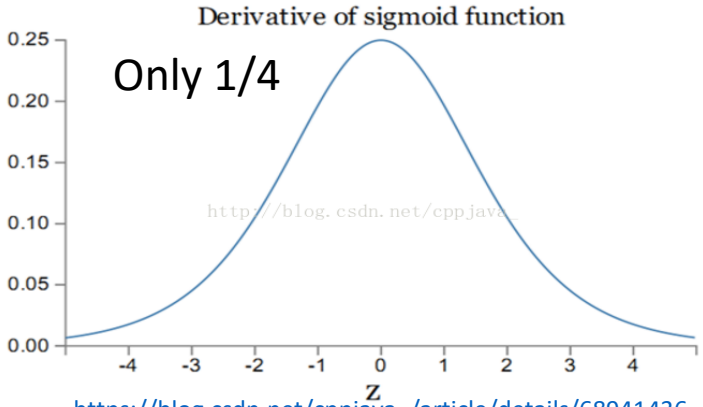
https://towardsdatascience.com/activation-functions-and-its-types-which-is-better-a9a5310cc8f



http://cswithjames.com/keras-6-vanishing-gradient-problem-relu/



http://blog.csdn.net/zjucor/article/details/7815

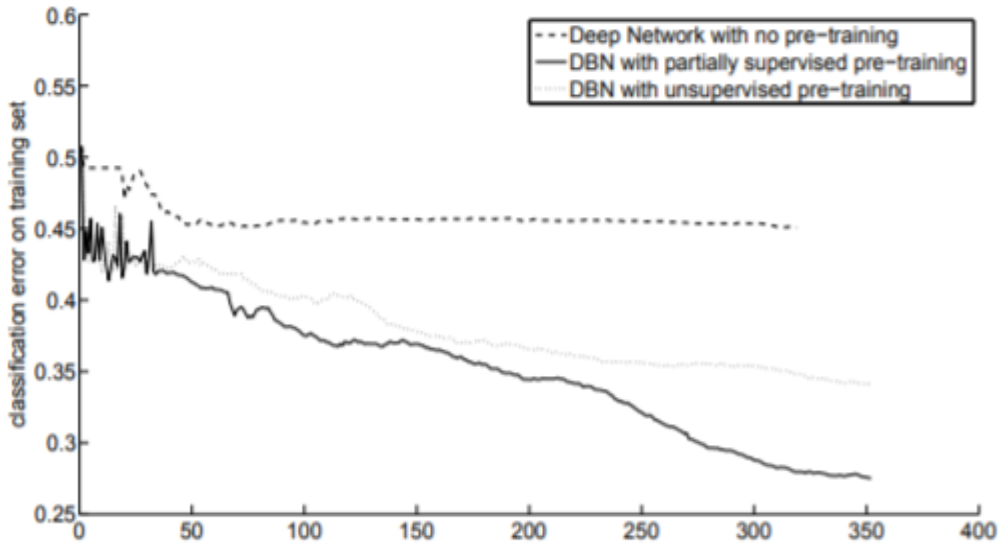


https://blog.csdn.net/cppjava_/article/details/68941436

$$
\begin{aligned}
g'_{\text{logistic}}(z) &= \frac{\partial}{\partial z}\left(\frac{1}{1+e^{-z}}\right) \\
&= \frac{e^{-z}}{(1+e^{-z})^2}(\text{chain rule}) \\
&= \frac{1+e^{-z}-1}{(1+e^{-z})^2} \\
&= \frac{1+e^{-z}}{(1+e^{-z})^2} - \left(\frac{1}{1+e^{-z}}\right)^2 \\
&= \frac{1}{(1+e^{-z})} - \left(\frac{1}{1+e^{-z}}\right)^2 \\
&= g_{\text{logistic}}(z) - g_{\text{logistic}}(z)^2 \\
&= g_{\text{logistic}}(z)(1 - g_{\text{logistic}}(z))
\end{aligned}
$$

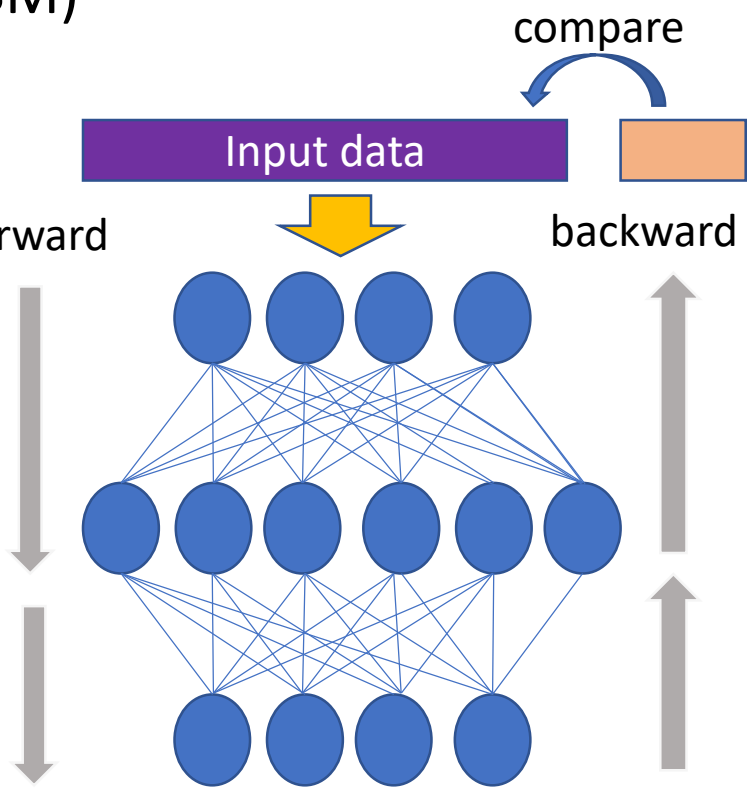
The power of deeper model

- Pretrain ---- the old fashion style
  - Restricted Boltzmann machine (RBM)



Greedy Layer-Wise Training of Deep Networks

Yoshua Bengio, Pascal Lamblin, Dan Popovici, Hugo Larochelle
Université de Montréal
Montréal, Québec

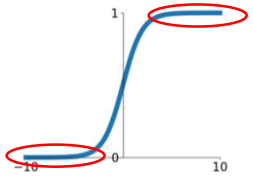


Activation functions

Activation Functions

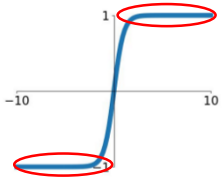
Sigmoid

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



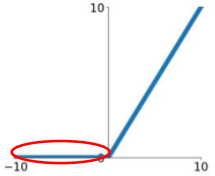
tanh

$\tanh(x)$



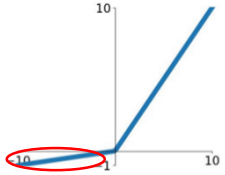
ReLU

$\max(0, x)$



Leaky ReLU

$\max(0.1x, x)$

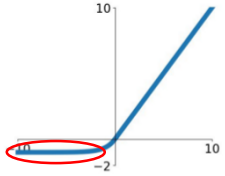


Maxout

$\max(w_1^T x + b_1, w_2^T x + b_2)$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$

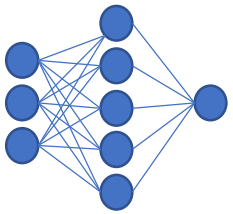


- The activation function help the NN
  - Now, the problem is mainly the computing resources

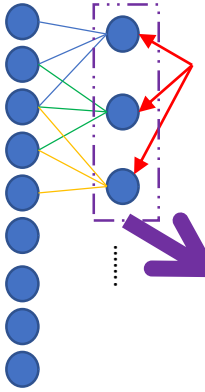
https://medium.com/machine-learning-world/how-to-debug-neural-networks-manual-dc2a200f10f2

Convolutional neural network

Considering the transform of the NN



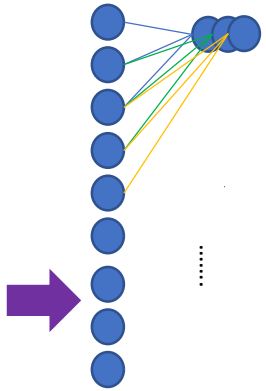
Basic architecture

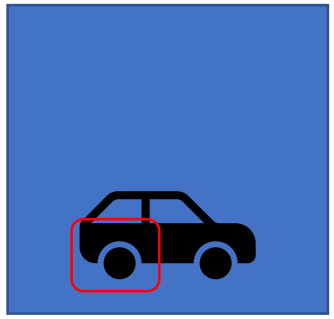


Weighted sum

How about…
If we already know the "triplet pattern" appears anywhere of the input?

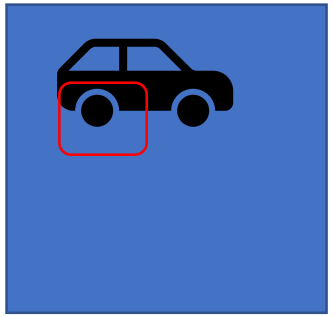
=> Yes, we can use the same weights for weighted sum



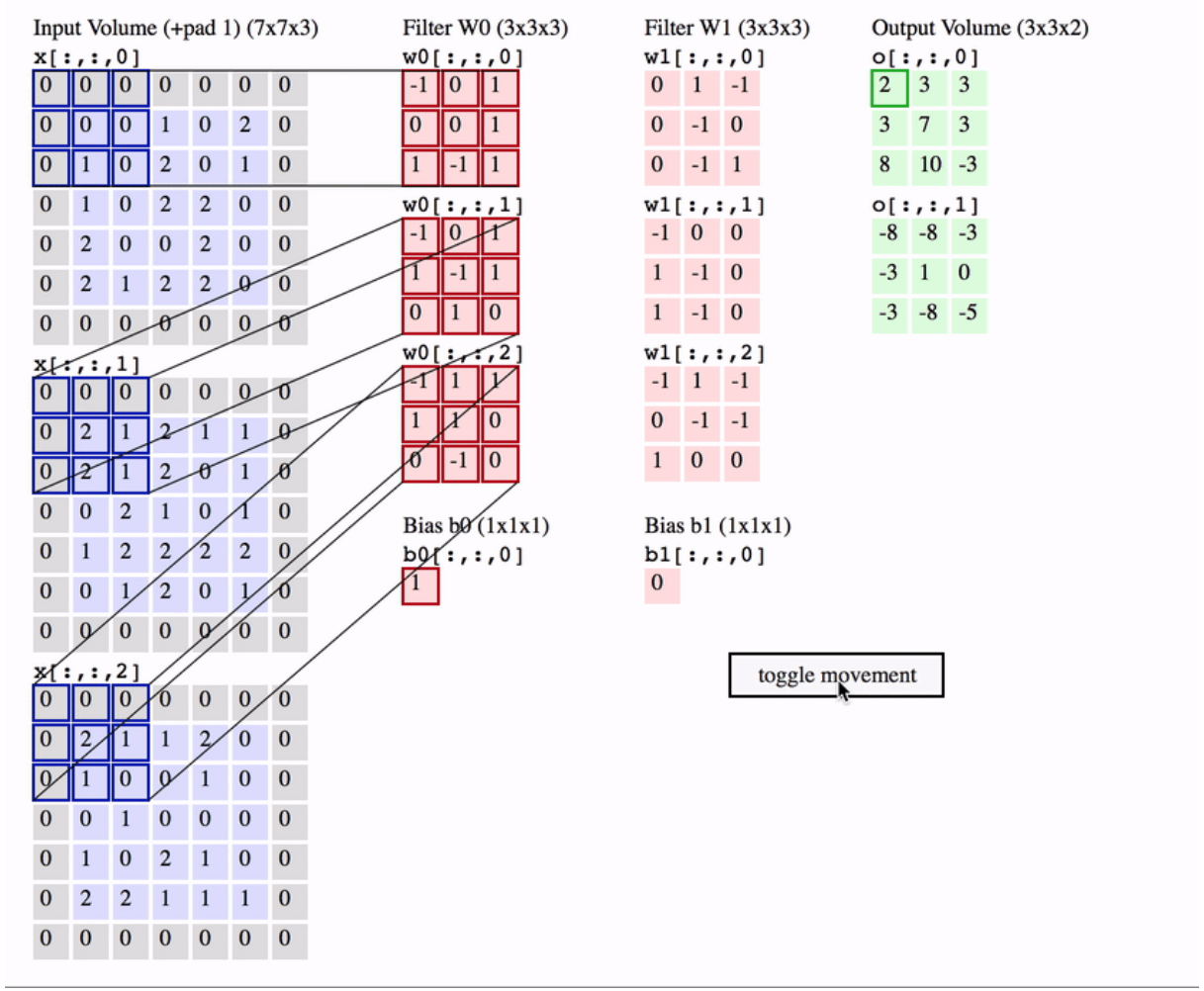


- Assume we already know the input data have "triplet pattern"
  - receptive field

- We use the same weight for weighted-sum operation
  - Weight sharing
  - Save more variables
  - Regulating the weights



Convolutional neural network

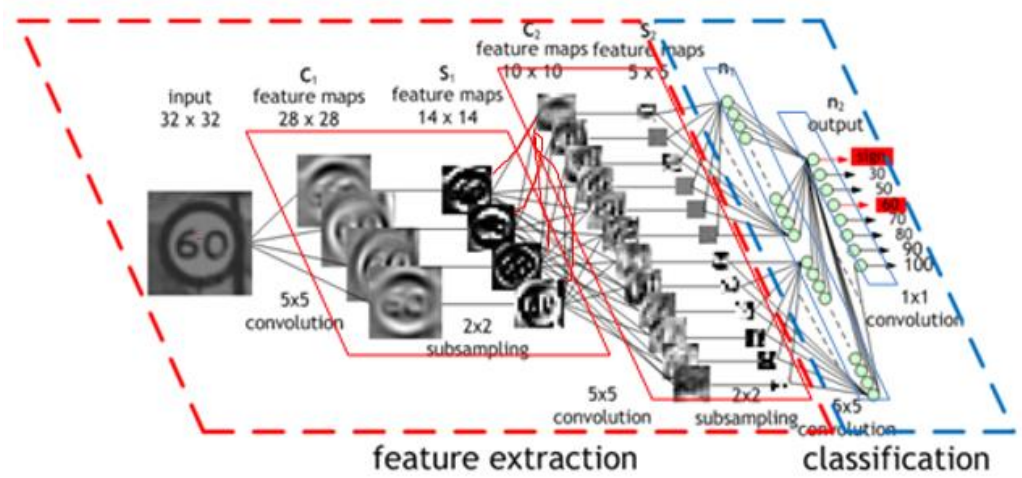


Normal terms:

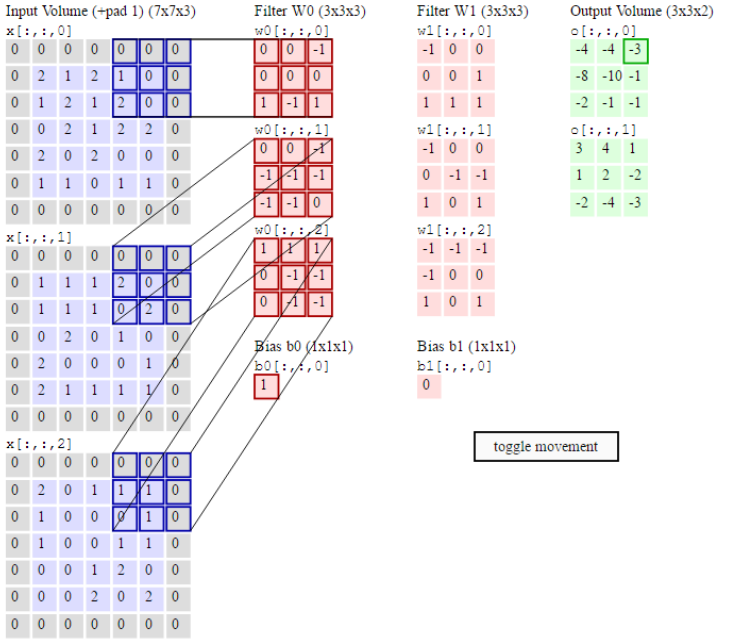
- Kernel size
- Stride
- padding

https://medium.com/@phidaouss/convolutional-neural-networks-cnn-or-convnets-d7c688b0a207

How about convolutional neural network



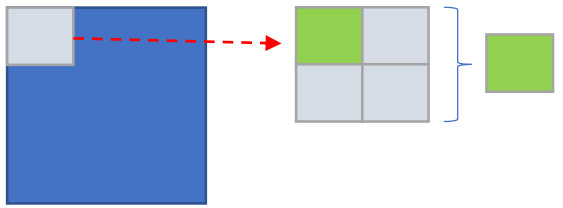
http://www.cnblogs.com/walccott/p/4957076.html

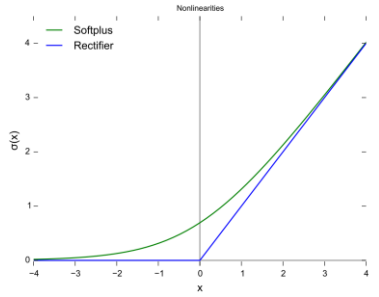


https://www.quora.com/In-a-convolutional-neural-network-how-is-convolution-defined-with-an-RGB-image

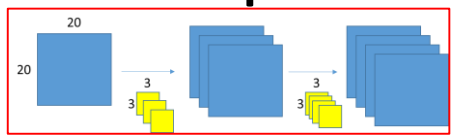
- CNN is also a kind of NNs, but the hidden nodes have their own receptive fields and sharing their weights
- Each “feature map” is a node.
- The ability for classification is still from the fully connected NN.
  - This also means, appending deeper layer here would cause gradient vanishment

Subsampling: pooling method





The parameters – CNN as example in detail



original

20

20

3

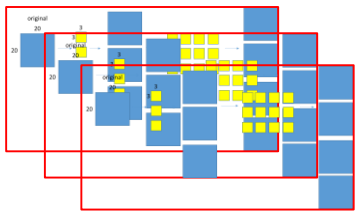
3

batch

100

??

accelerating



The precious counting miss these condition !

(This is also HMM)

Condition :

1. No pooling layer
2. Do not consider the activation function

Correction:

1. 20 * 20 = 400
2. 3 * 3 * 3 = 27
3. 400 *3 = 1200
4. 3 * 3 * 4 * 3 = 108
5. 400 * 4 = 1600

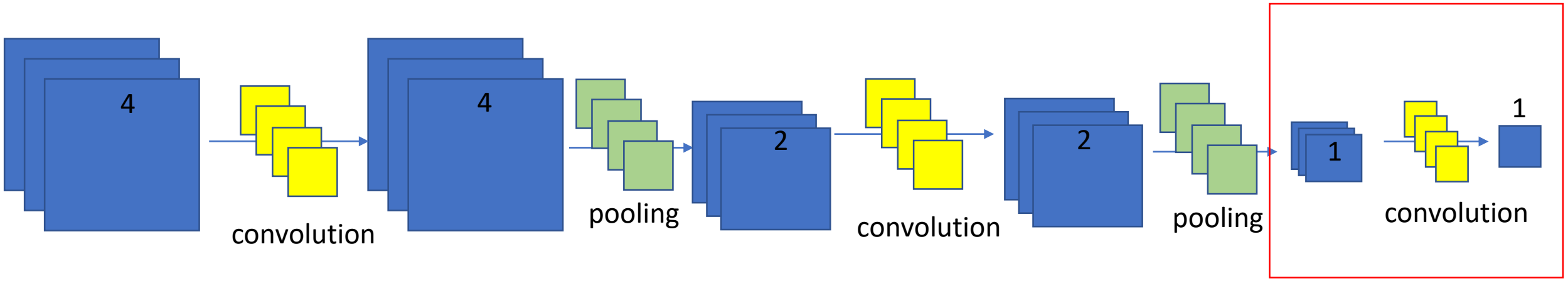
Total:

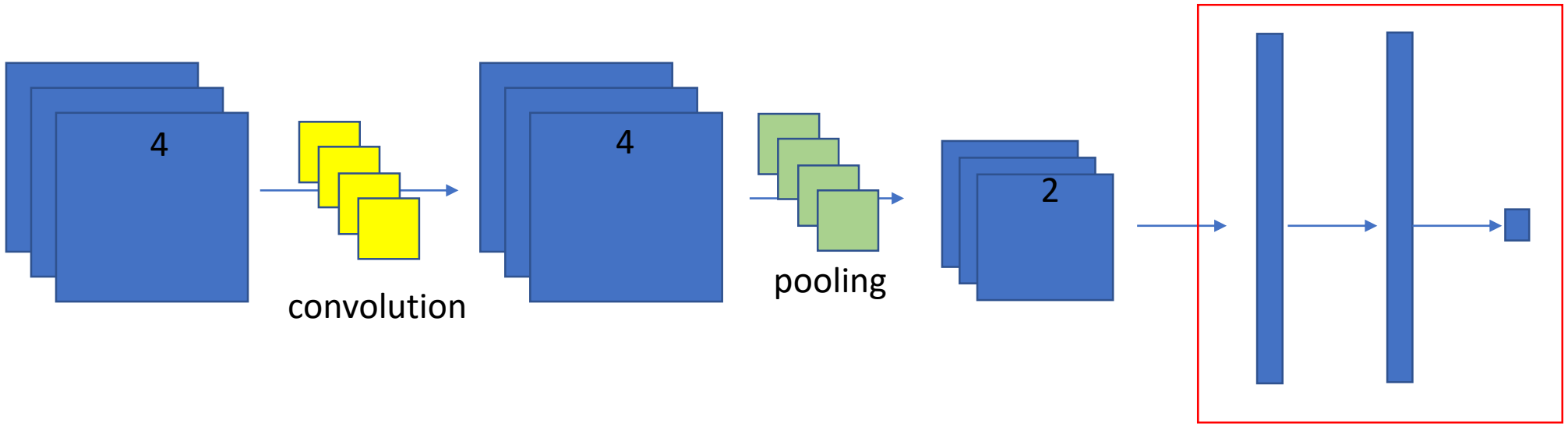
400 + 27 + 1200 + 108 + 1600 = 3335

3335 * 4 (byte) = **13340**

(previous : 556 bytes)

BTW: Fully convolutional neural network





Convolutional neural network practice

- Many convenient tools are already invented. But to better understanding how the tensors flow, we try to implement it as detailed as possible.
  - 3CNN.py

- The next step is the Keras module in Tensorflow
  - 3CNN_keras.py

Thank you~