Deep Learning Interview Questions

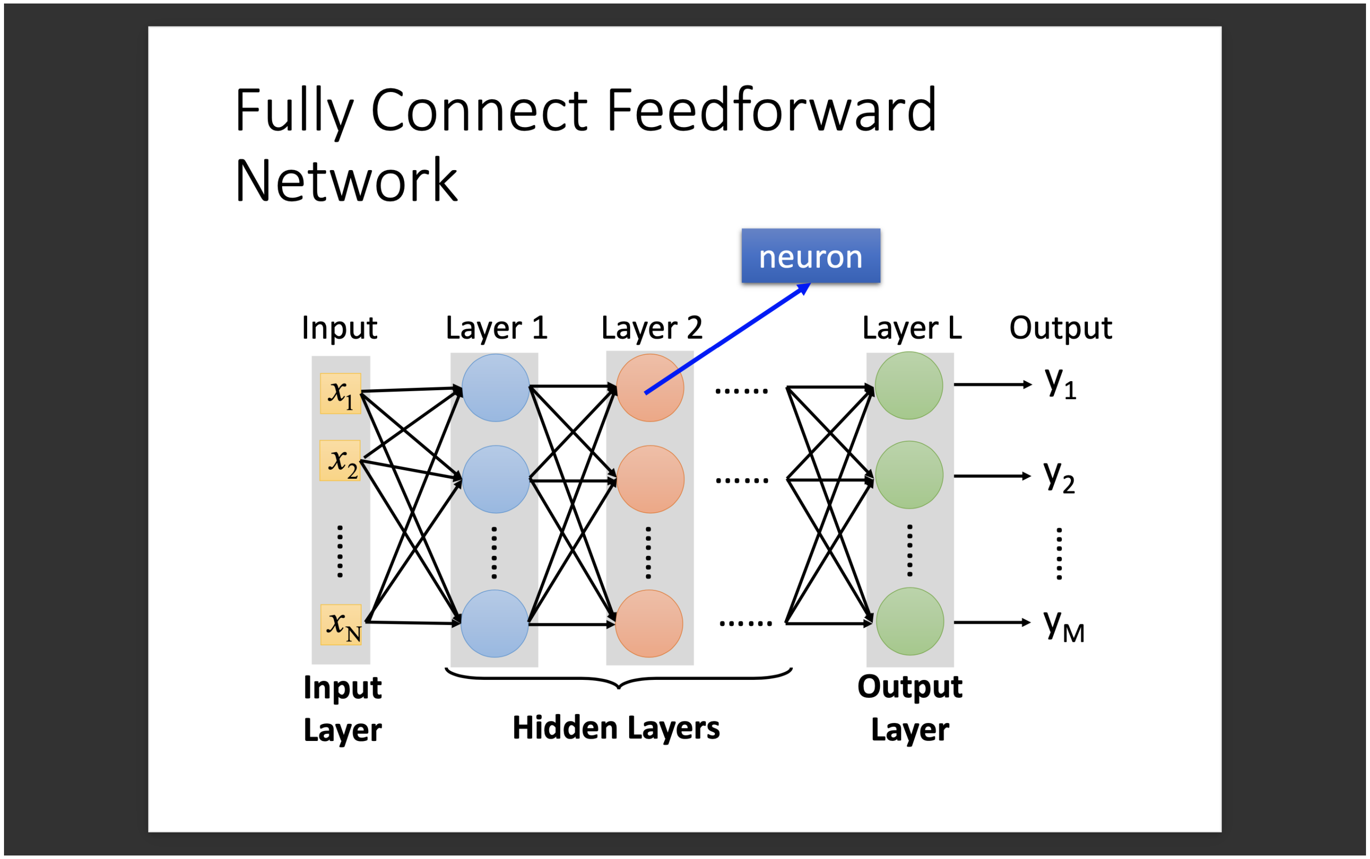
Tseng1026 Follow
Mar 1 · 8 min read

In this article, I will focus more on algorithms and theory, including neural network itself, activation function, gradient descent, learning rate, and loss function. And the next piece would discuss programming skills.

What are the steps of deep learning?
1. create a function (neural network)
2. evaluate the goodness of the function
3. pick the best function as the final question answer machine

What is a neural network?

Neural network simulates the ways human learn but is much simpler. It can be imagined as a function. When you feed it input, you are supposed to get an output. It commonly consists of an input layer, hidden layer(s), and an output layer.



Fully Connect Feedforward Network

Reference: Hung-Yi Lee's Lecture Slides

Why is it necessary to introduce non-linearities in the neural network?
If there are all linear functions, they actually compose a new linear function, which gives a linear model. A linear model has a much smaller number of parameters and is therefore limited in its complexity.

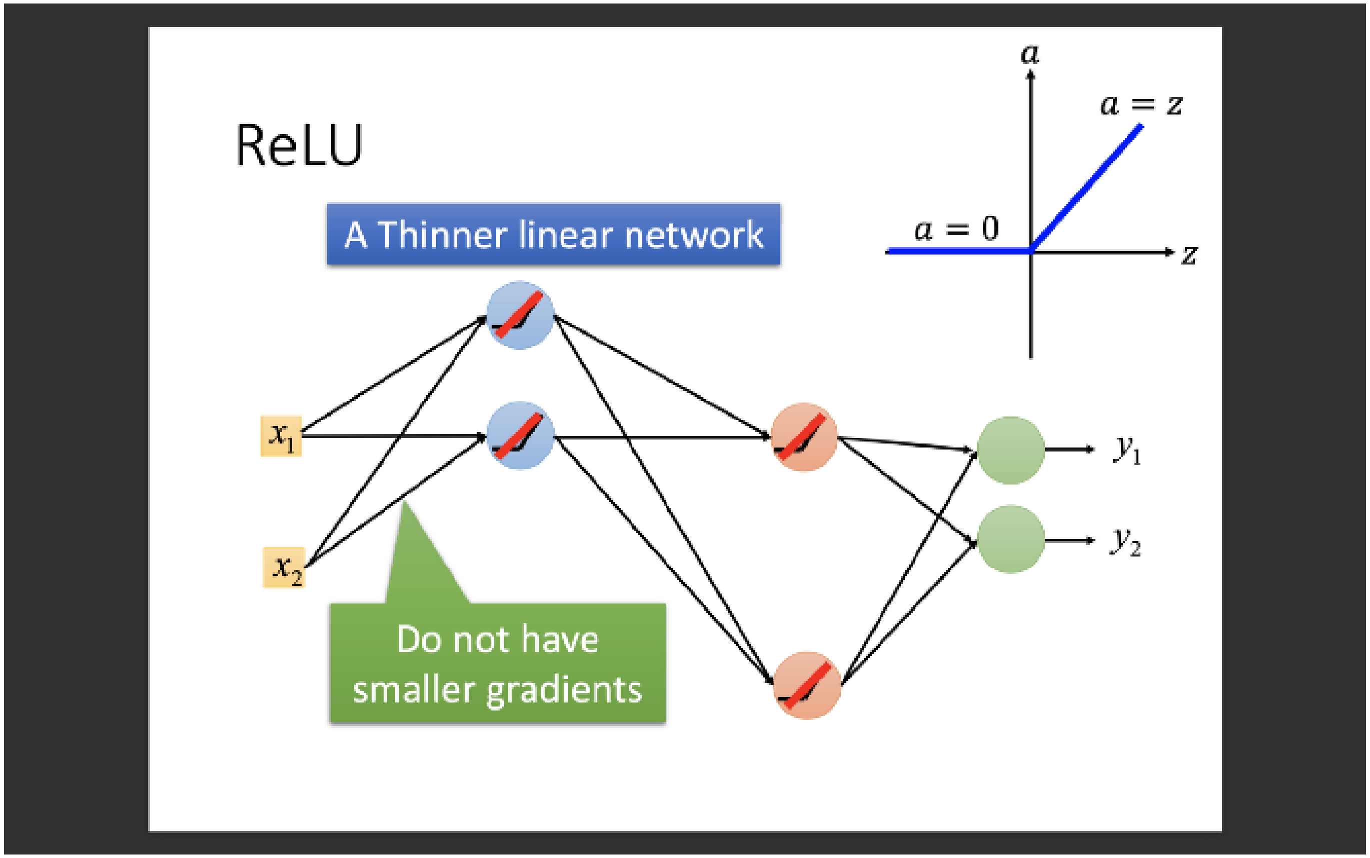
What is the difference between single-layer perceptron and multi-layer perceptron?
The main difference between them is the existence of hidden layers. Multi-layer perceptron can classify nonlinear data and withstand great numbers of parameters. (Except for the input layer, each node in the other layers uses a nonlinear activation function.)

Which one is better, shallow networks or deep networks?
Both shallow and deep networks are good enough and capable of approximating any function. But for the same level of accuracy, deeper networks can be much more efficient in terms of computation and number of parameters. Deeper networks can create deep representations. At every layer, the network learns a new, more abstract feartures of the input.

What is a activation function?
At the most basic level, an activation function decides whether a neuron should be activated or not. It accepts the weighted sum of the inputs and

bias as input to any activation function. Sigmoid, ReLU, Maxout, Tanh, and Softmax are examples of activation functions.

For Sigmoid, the function transforms the values to the range [0, 1], but it is hard to compute. Besides, we might have a large input difference but lead to a small output gap, which causes gradient vanishing problems.
For ReLU, it simulates the biological neurons and is much faster to compute. The most important is ReLU solves the vanishing gradient problem.



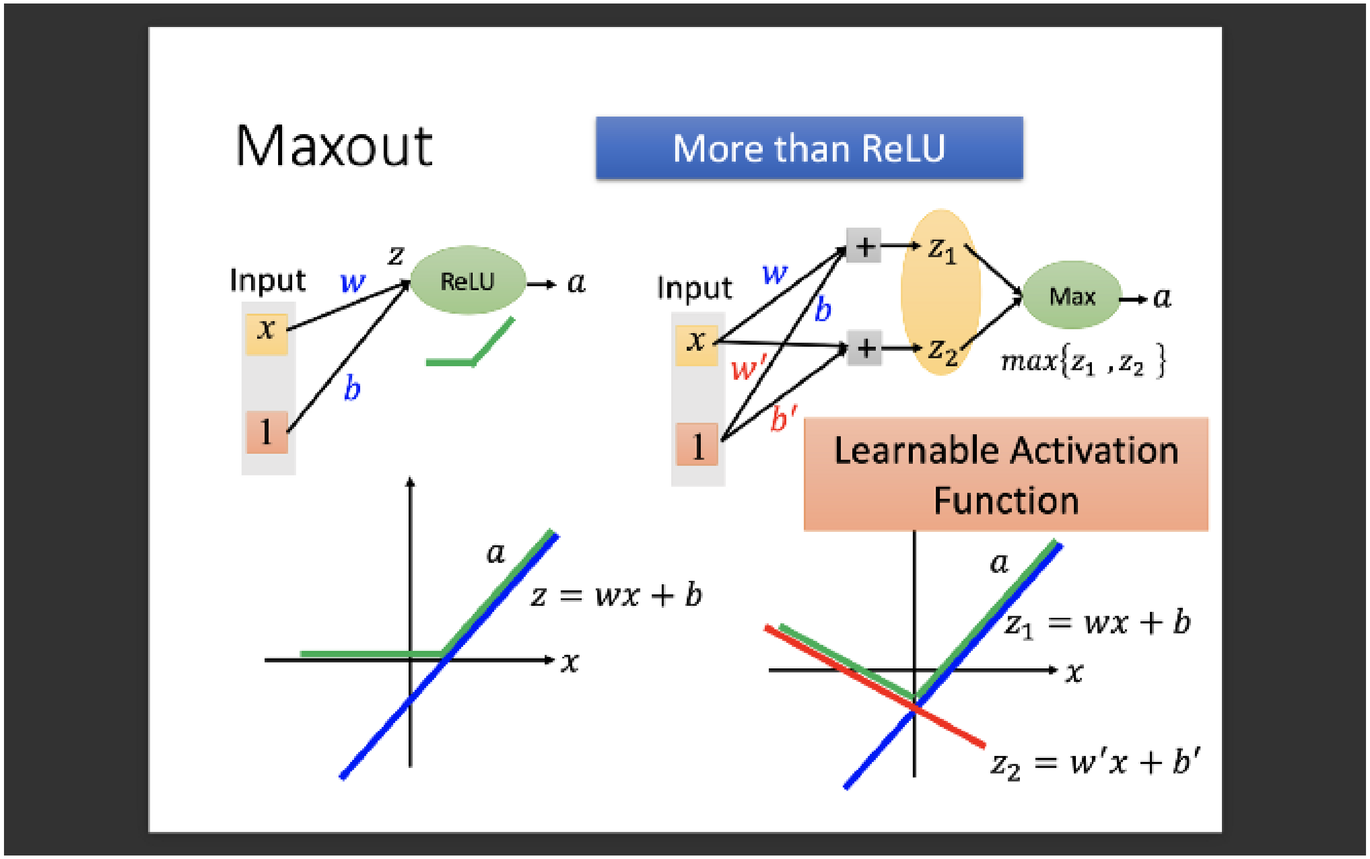
ReLU

A Thinner linear network

Do not have smaller gradients

Reference: Hung-Yi Lee's Lecture Slides

For Maxout, instead of replacing the negative values to 0, we retain the maximum values among neurons in the same layer. In other words, we can say that ReLU is a special case of Maxout, which contains an always 0 neuron.



Maxout

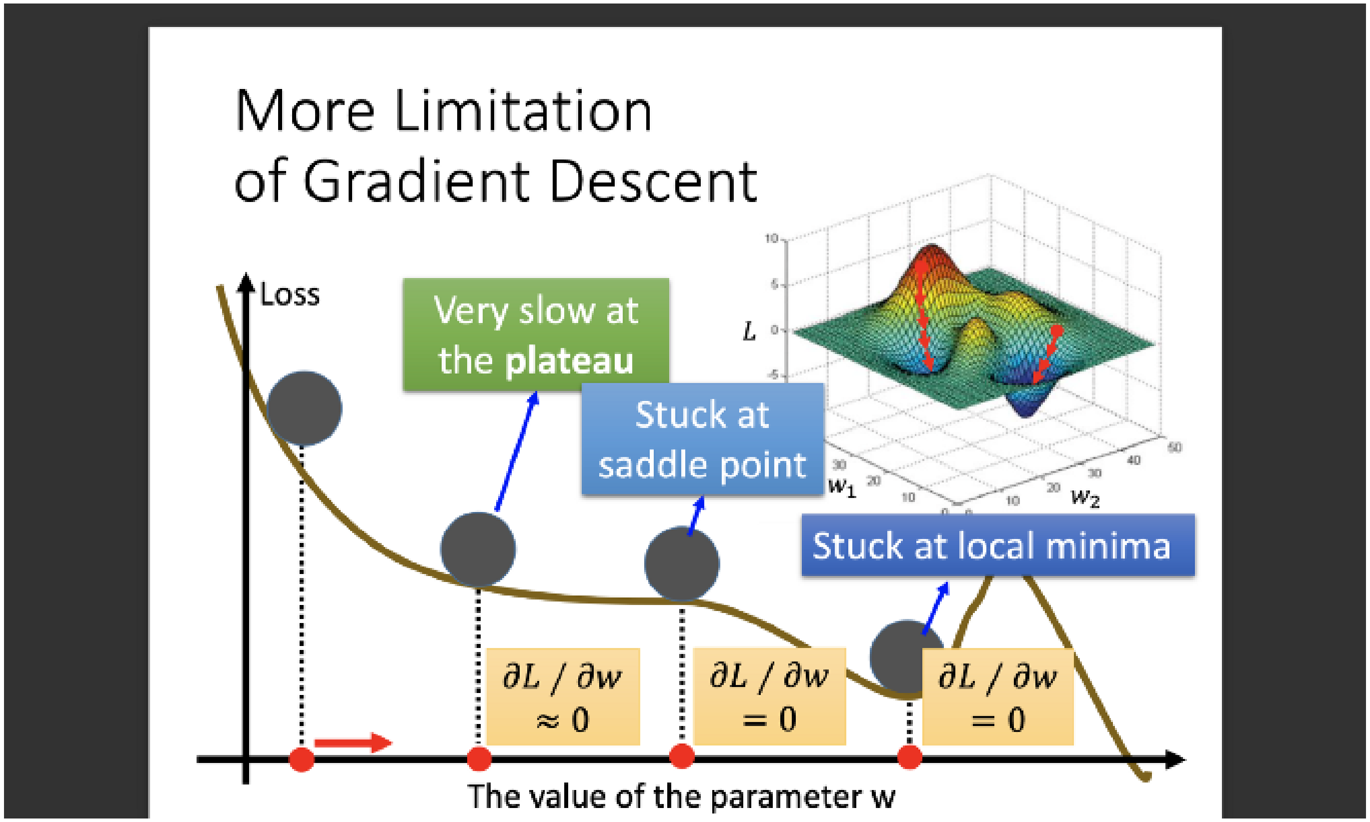
More than ReLU

Learnable Activation Function

Reference: Hung-Yi Lee's Lecture Slides

What is gradient descent, and what is the difference between batch gradient descent and stochastic gradient descent?
We would like to minimize the errors; therefore we move toward the opposite direction of the gradient of losses.

We consider the whole batch for batch gradient descent, which costs much time, but it is more stable.
For stochastic gradient descent, on the contrary, we consider only one example, making the process much faster.



More Limitation of Gradient Descent

Very slow at the **plateau**

Stuck at saddle point

Stuck at local minima

$\partial L / \partial w \approx 0$

$\partial L / \partial w = 0$

$\partial L / \partial w = 0$

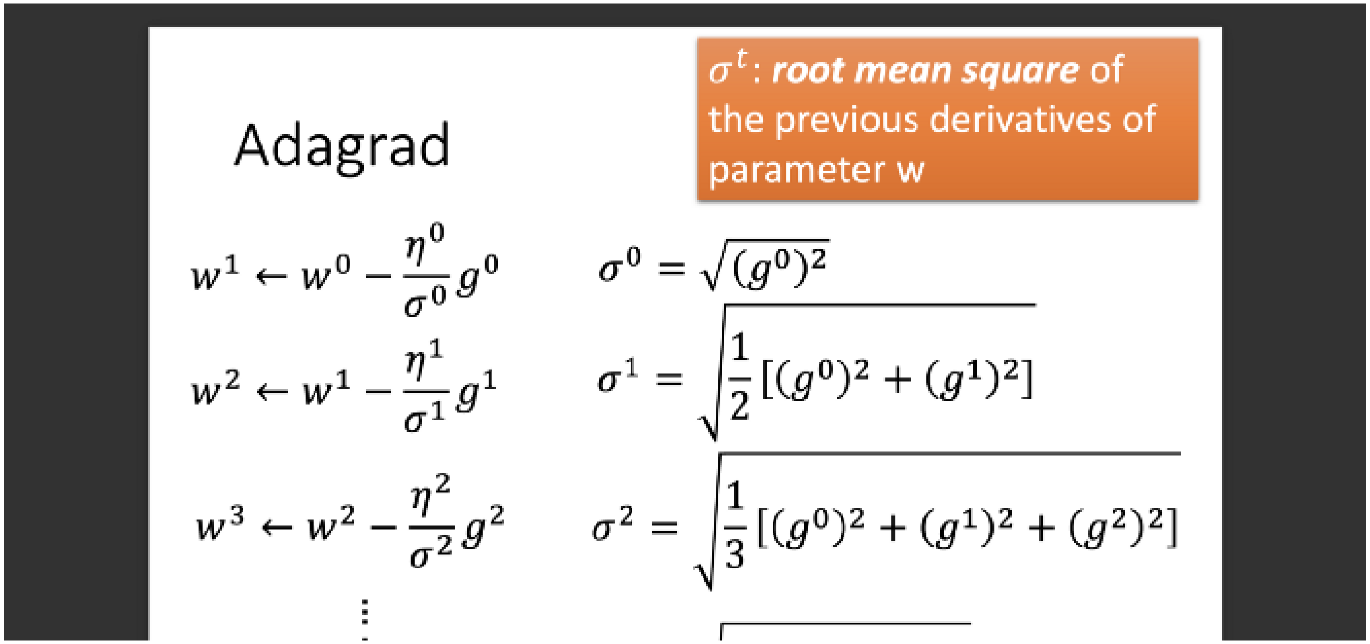
The value of the parameter w

Reference: Hung-Yi Lee's Lecture Slides

What is the adaptive learning rate?

With intuition, we know that we would need a larger learning rate at the begin, and reduced as the steps go through. It determines how much we are moving to the direction computing by the gradient. Adagrad, RMSprop, and Adam are examples of adaptive learning rates.

For Adagrad, we divide the learning rate of each parameter by the root mean square of its previous derivatives, leading to smaller step when having moved a long distance.



Adagrad

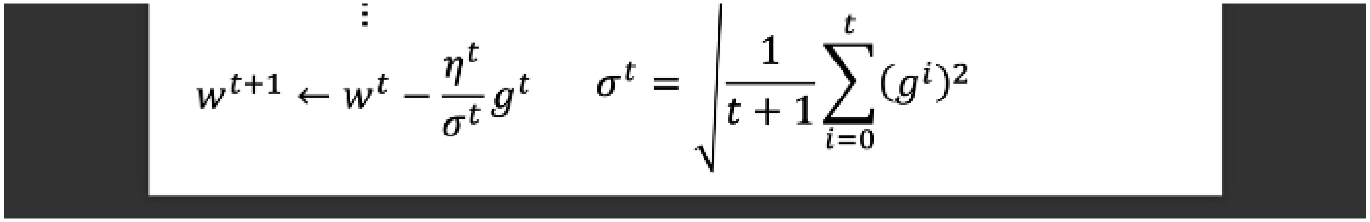
σ^t: ***root mean square*** of the previous derivatives of parameter w

$$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0 \qquad \sigma^0 = \sqrt{(g^0)^2}$$

$$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\frac{1}{2}[(g^0)^2 + (g^1)^2]}$$

$$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\frac{1}{3}[(g^0)^2 + (g^1)^2 + (g^2)^2]}$$

$$\vdots$$

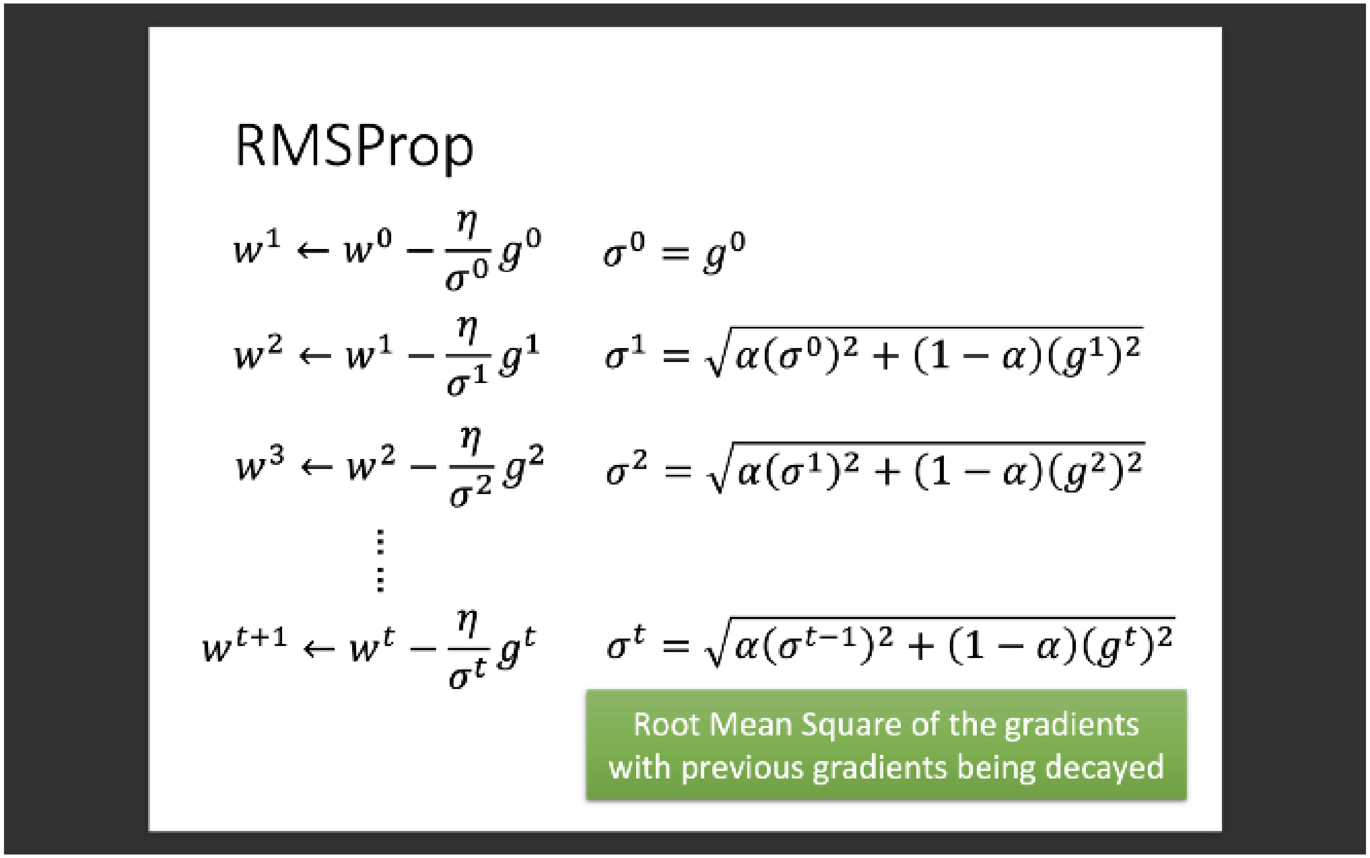


$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

Reference: Hung-Yi Lee’s Lecture Slides

For RMSprop, we divide the root mean square of the decayed previous derivatives instead of the average.



RMSProp

$$w^1 \leftarrow w^0 - \frac{\eta}{\sigma^0} g^0 \qquad \sigma^0 = g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\alpha(\sigma^0)^2 + (1-\alpha)(g^1)^2}$$

$$w^3 \leftarrow w^2 - \frac{\eta}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\alpha(\sigma^1)^2 + (1-\alpha)(g^2)^2}$$

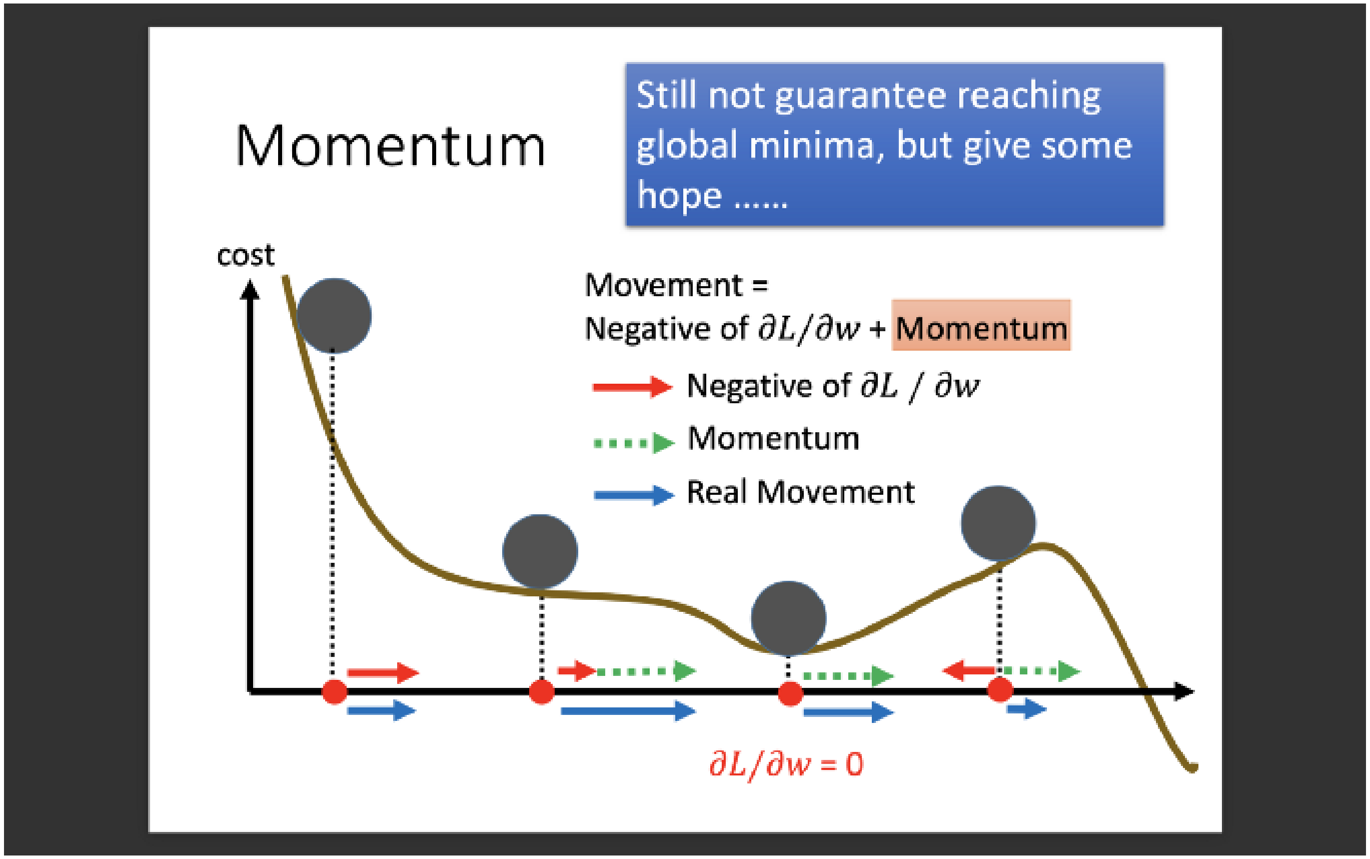
$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\alpha(\sigma^{t-1})^2 + (1-\alpha)(g^t)^2}$$

Root Mean Square of the gradients with previous gradients being decayed

Reference: Hung-Yi Lee's Lecture Slides

As for Adam, we combine the concepts of momentum and RMSprop. In this case, the movement is not just based on gradient, but also the previous movement, and can perhaps conquer the situation of local minima or saddle point.



Momentum

Still not guarantee reaching global minima, but give some hope ……

Movement =
Negative of $\partial L / \partial w$ + Momentum

🟥 Negative of ∂L / ∂w

🟩 Momentum

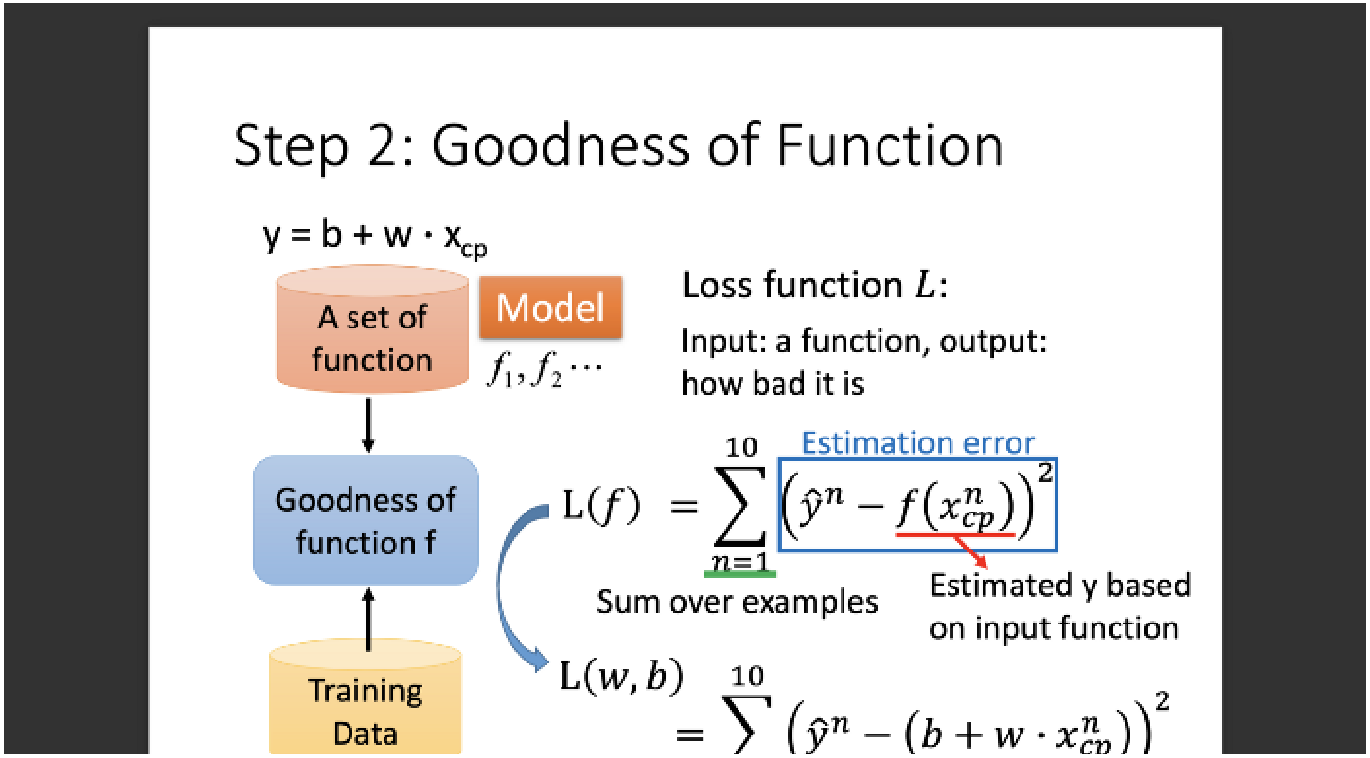
🟦 Real Movement

Reference: Hung-Yi Lee's Lecture Slides

What is loss function?

The loss function is used as a measure of accuracy to see if a neural network has learned accurately from the training data or not. In Deep Learning, a good performing network will have a low loss function at all times when training.

For regression tasks, the most commonly used method is mean square error. It computes the distance between the actual value and the predicted value so that we would gradually move toward the correct value by gradient descent algorithm.



Step 2: Goodness of Function

$y = b + w \cdot x_{cp}$

A set of function

Model

$f_1, f_2 \cdots$

Goodness of function f

Training Data

Loss function L:

Input: a function, output: how bad it is

Estimation error

$$\mathrm{L}(f) = \sum_{n=1}^{10} \boxed{\left(\hat{y}^n - f(x_{cp}^n)\right)^2}$$

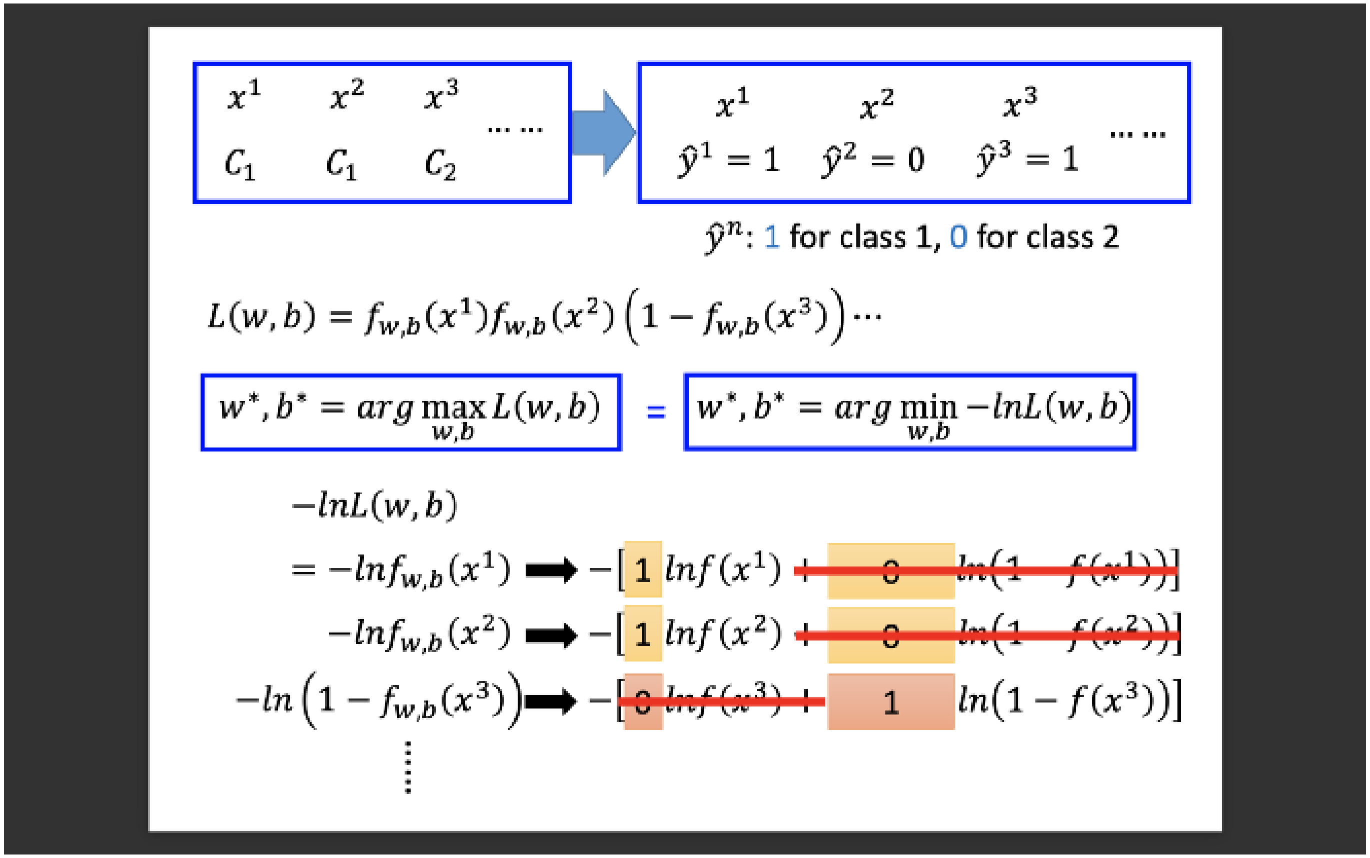
Sum over examples

Estimated y based on input function

$$\mathrm{L}(w, b) = \sum_{n=1}^{10} \left(\hat{y}^n - (b + w \cdot x_{cp}^n)\right)^2$$

Reference: Hung-Yi Lee’s Lecture Slides

For classification problems, we usually use cross entropy loss. Instead of measure the difference between specific values, we consider the distributions among different classes.



$$\boxed{\begin{matrix} x^1 & x^2 & x^3 & \\ C_1 & C_1 & C_2 & \cdots\cdots \end{matrix}} \Rightarrow \boxed{\begin{matrix} x^1 & x^2 & x^3 & \\ \hat{y}^1 = 1 & \hat{y}^2 = 0 & \hat{y}^3 = 1 & \cdots\cdots \end{matrix}}$$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$L(w,b) = f_{w,b}(x^1) f_{w,b}(x^2) \left(1 - f_{w,b}(x^3)\right) \cdots$$

$$\boxed{w^*, b^* = arg \max_{w,b} L(w,b)} = \boxed{w^*, b^* = arg \min_{w,b} -lnL(w,b)}$$

$$-lnL(w,b)$$

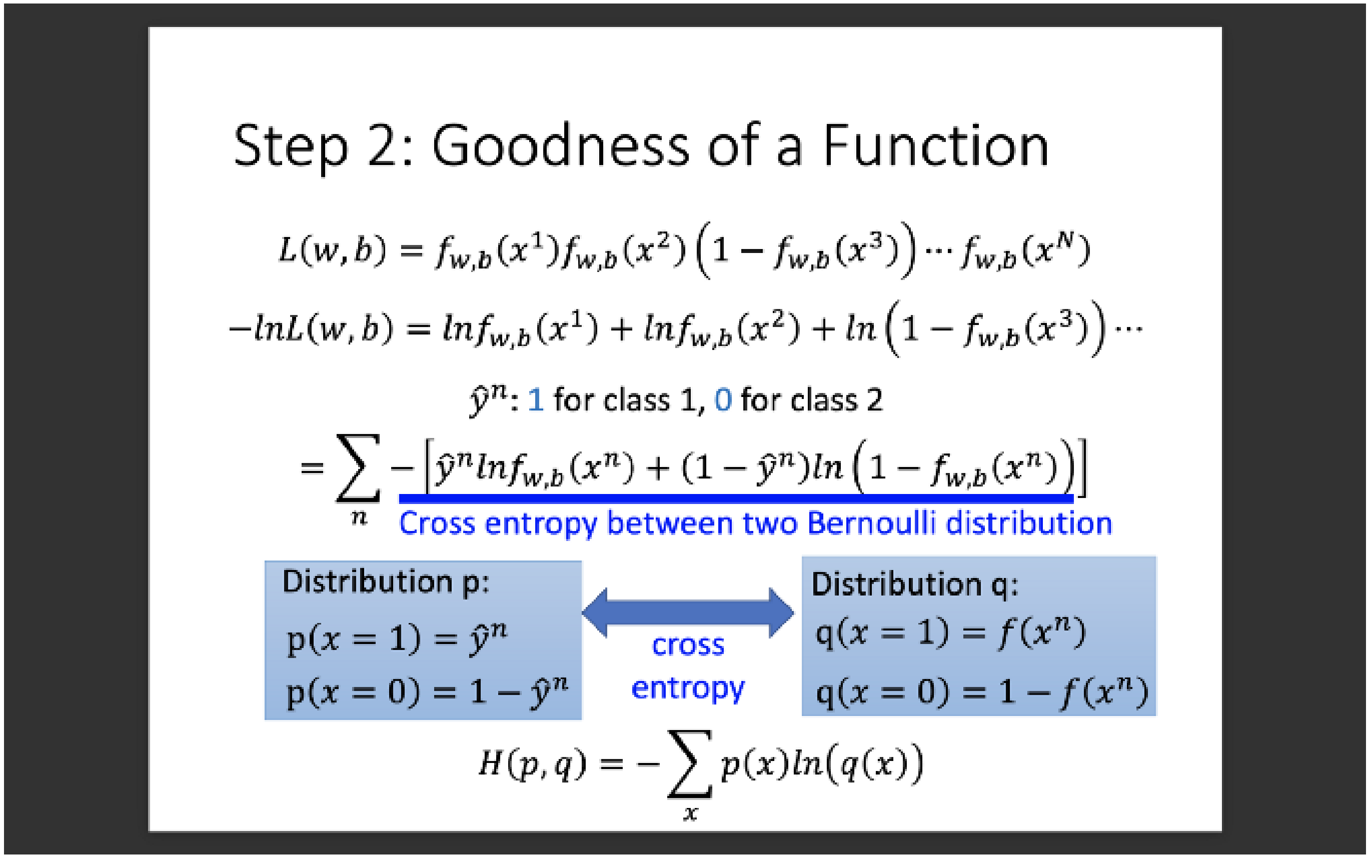
$$= -lnf_{w,b}(x^1) \Rightarrow -[1\, lnf(x^1) \cancel{+ 0\, ln(1 - f(x^1))}]$$

$$-lnf_{w,b}(x^2) \Rightarrow -[1\, lnf(x^2) \cancel{+ 0\, ln(1 - f(x^2))}]$$

$$-ln\left(1 - f_{w,b}(x^3)\right) \Rightarrow -[\cancel{0\, lnf(x^3) +}\ 1\, ln(1 - f(x^3))]$$

$$\vdots$$

Reference: Hung-Yi Lee’s Lecture Slides



Step 2: Goodness of a Function

$$L(w,b) = f_{w,b}(x^1)f_{w,b}(x^2)\left(1 - f_{w,b}(x^3)\right)\cdots f_{w,b}(x^N)$$

$$-lnL(w,b) = lnf_{w,b}(x^1) + lnf_{w,b}(x^2) + ln\left(1 - f_{w,b}(x^3)\right)\cdots$$

$\hat{y}^n$: 1 for class 1, 0 for class 2

$$= \sum_n -\underline{\left[\hat{y}^n lnf_{w,b}(x^n) + (1-\hat{y}^n)ln\left(1 - f_{w,b}(x^n)\right)\right]}$$

Cross entropy between two Bernoulli distribution

Distribution p:

$p(x = 1) = \hat{y}^n$

$p(x = 0) = 1 - \hat{y}^n$

cross entropy

Distribution q:

$q(x = 1) = f(x^n)$

$q(x = 0) = 1 - f(x^n)$

$$H(p,q) = -\sum_x p(x)ln\left(q(x)\right)$$

Reference: Hung-Yi Lee's Lecture Slides

How are weights initialized in a network?

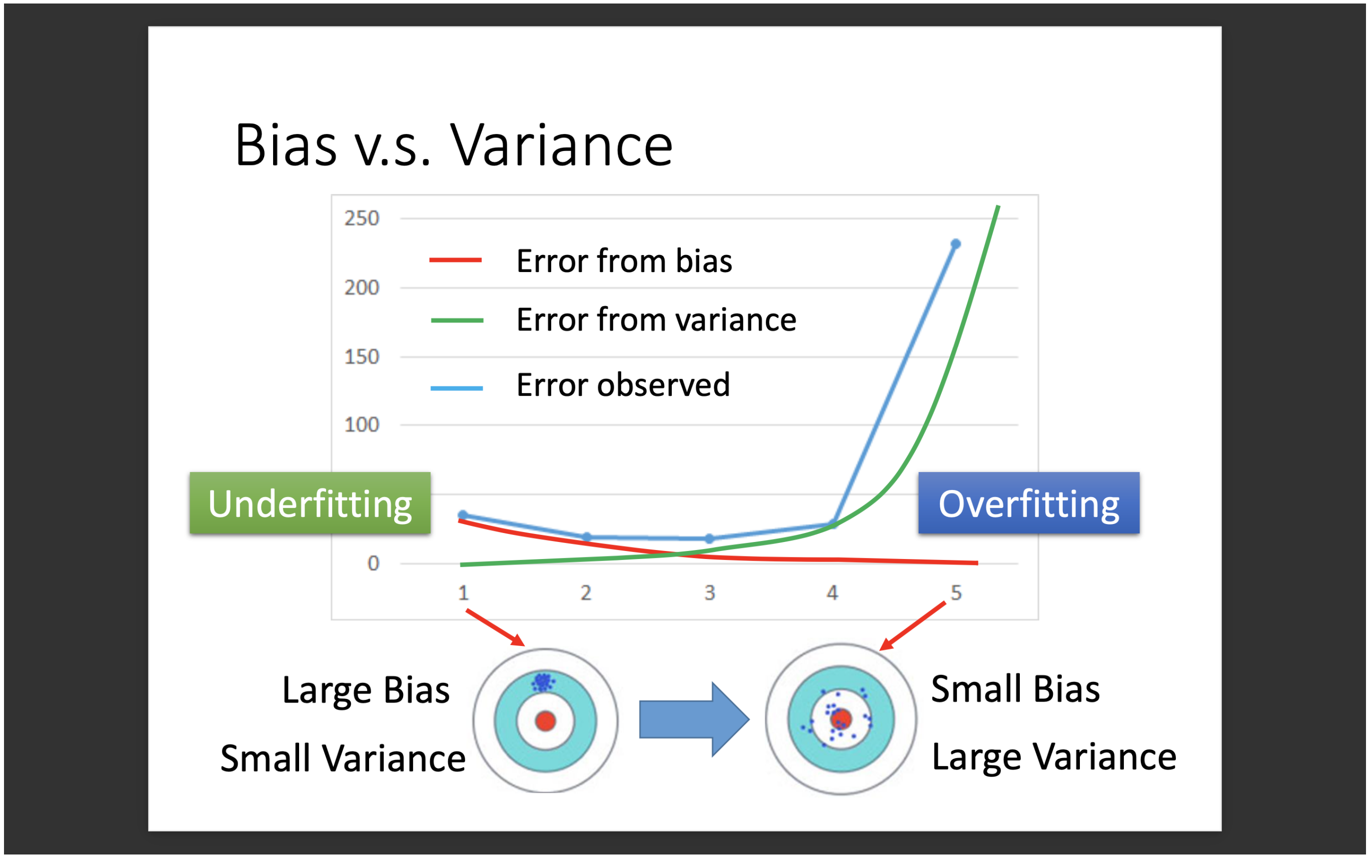
Initializing all weights to 0: This makes your model similar to a linear model. All the neurons and every layer perform the same operation, giving the same output and making the deep network useless.

Initializing all weights randomly: Here, the weights are assigned randomly by initializing them very close to 0. It gives better accuracy to the model since every neuron performs different computations. This is the most commonly used method.

What is the difference between bias and variance?

Bias is an error caused by simple assumptions, leading to the model underfitting the data (poor performance on both training and validation data). We should change the model architecture to a more complex one, increase the number of iterations, or increase the number of used feature dimensions.

Variance is an error due to too much complexity in the algorithm, making the model overfitting (good performance on training data, but poor performance on validation data). In this case, we should do some regularization, increase data size, or reduce the number of used feature dimensions.



Bias v.s. Variance

🟥 Error from bias
🟩 Error from variance
🟦 Error observed

Underfitting

Overfitting

Large Bias
Small Variance

Small Bias
Large Variance

Reference: Hung-Yi Lee’s Lecture Slides

What is dropout?

Dropout is a regularization technique for reducing overfitting in neural networks by dropping out the neurons with probability p>0. It forces the model to avoid relying too much on particular sets of features.

What is batch normalization?

To facilitate learning, we typically normalize the initial values of our parameters by initializing them with zero mean and unit variance. As training progresses and we update parameters to different extents, we lose this normalization, which slows down training and amplifies changes as the network becomes deeper.

Batch normalization re-establishes these normalizations for every mini-batch, and changes are back-propagated through the operation as well. By making normalization part of the model architecture, we are able to use higher learning rates and pay less attention to the initialization parameters. Batch normalization additionally acts as a regularizer, reducing and even eliminating the need for dropout.

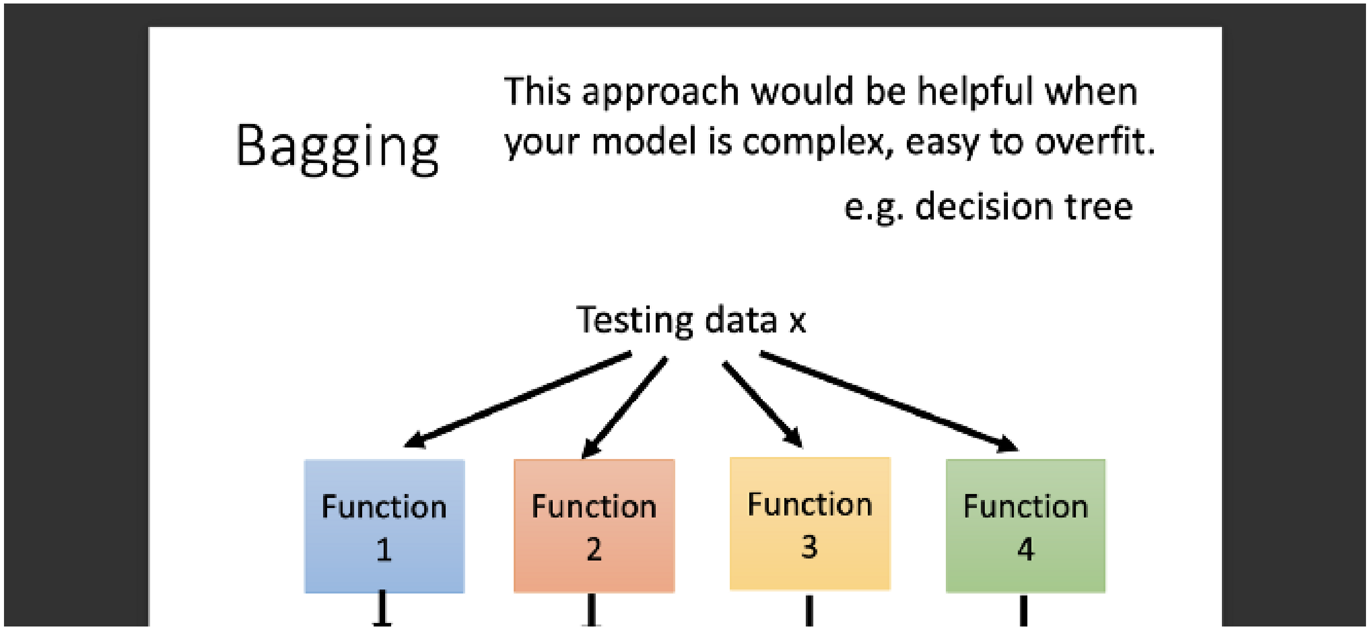
What is early stopping?
It is a regularization technique that stops the training process as soon as the validation loss reaches a plateau or starts to increase.

What is parameter norm penalty?
When computing cost function, our goal is to find the minimal loss. However, it may sometimes cause overfitting, which might not generalize well on other datasets. Therefore, we add a penalty term in the loss function avoid the network using extreme weights; L1 and L2 are two commonly seen examples.

L1 norm is the summation of all the dimensions and could select the more useful features. Moreover, it is more robust than L2 norm, which might consider the outlier data.
L2 norm is the summation of the square of dimension. Unlike L1 norm, L2 does not have the attribute sparsity. L2 norm can compute efficiently to find the minimum value, so it is used in algorithms more.

What is difference between bagging and boosting?
They are both approaches for increasing the accuracy.

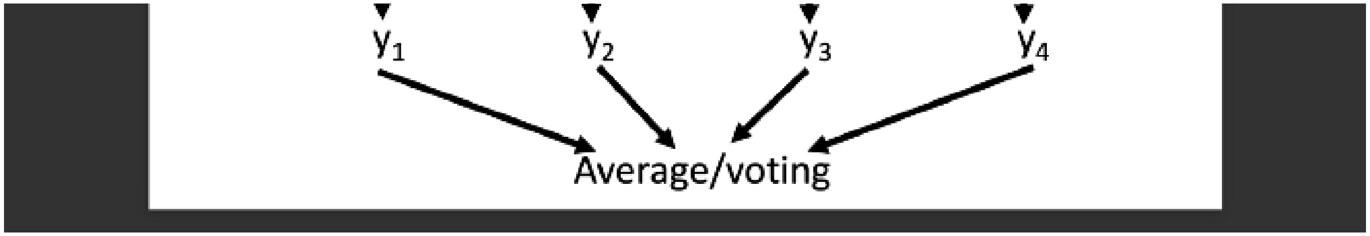
The basic concept for bagging is voting. It combines the parallelly predicted results of several strong models to avoid overfitting.



Bagging

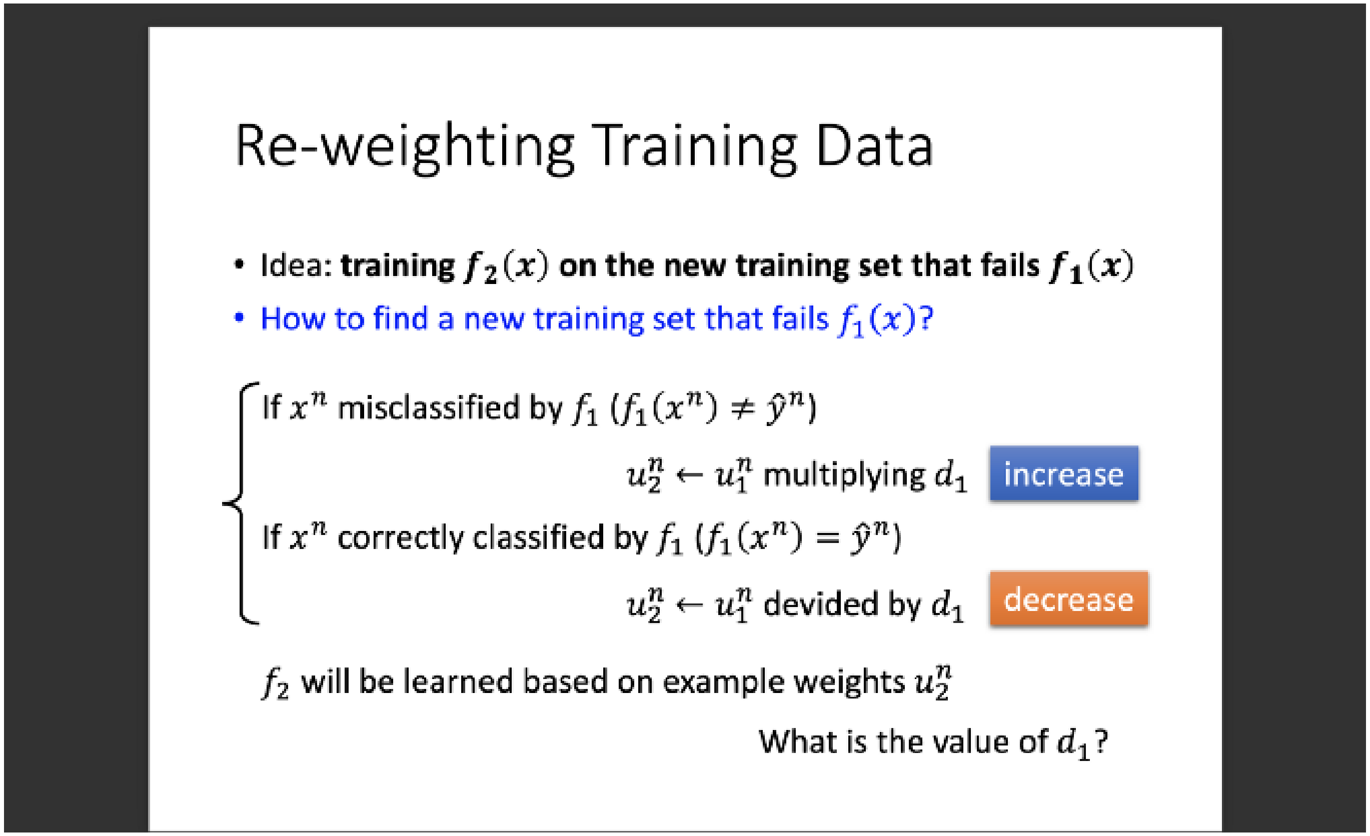
This approach would be helpful when your model is complex, easy to overfit.

e.g. decision tree



Reference: Hung-Yi Lee's Lecture Slides

As for boosting, it iteratively trains the model to focus more on the incorrect predicted data (confusion area). It helps weak models to reduce the situation of underfitting.



Re-weighting Training Data

- Idea: **training $\boldsymbol{f_2(x)}$ on the new training set that fails $\boldsymbol{f_1(x)}$**
- How to find a new training set that fails $f_1(x)$?

If x^n misclassified by f_1 ($f_1(x^n) \neq \hat{y}^n$)

$u_2^n \leftarrow u_1^n$ multiplying d_1 increase

If x^n correctly classified by f_1 ($f_1(x^n) = \hat{y}^n$)

$u_2^n \leftarrow u_1^n$ devided by d_1 decrease

f_2 will be learned based on example weights u_2^n

What is the value of d_1?

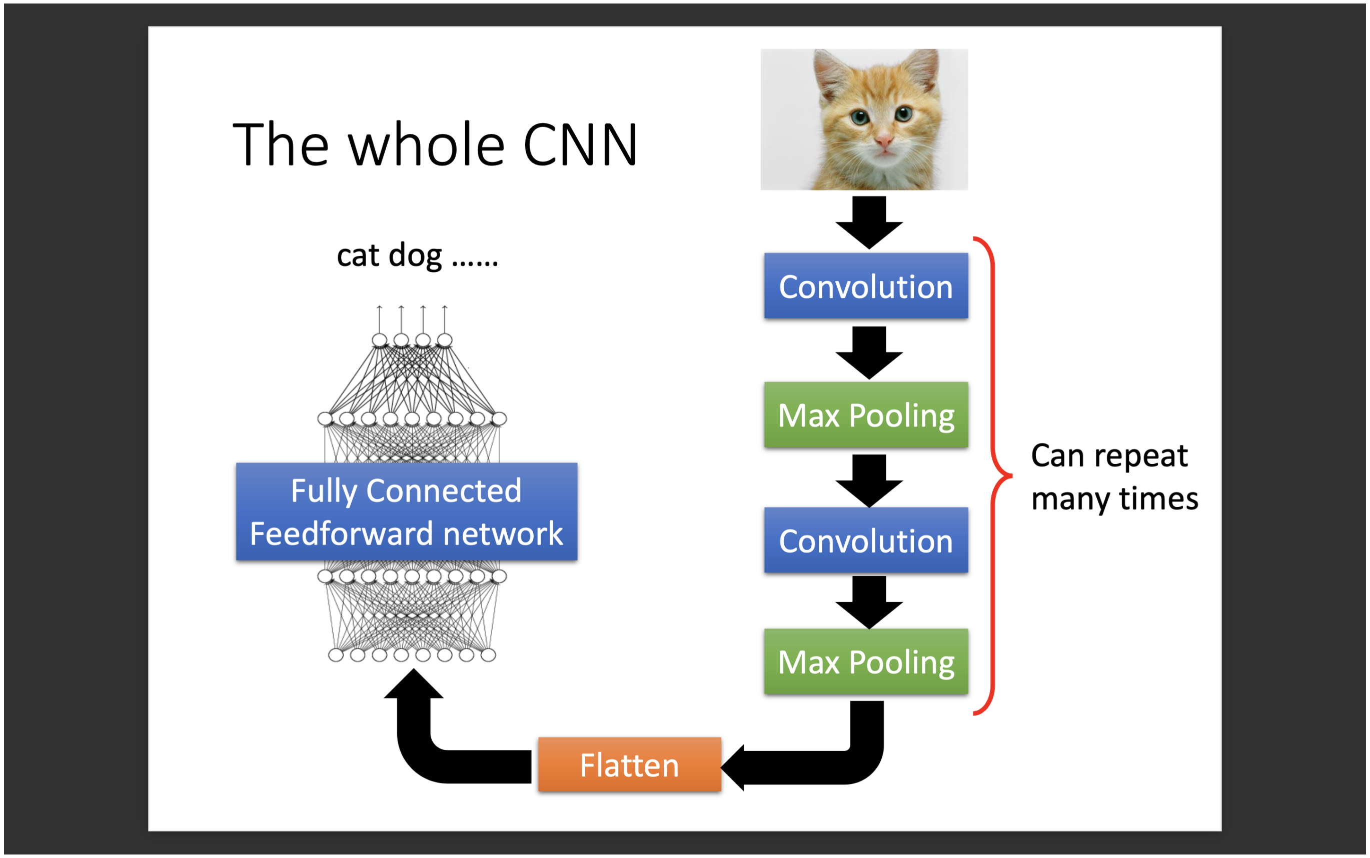
Reference: Hung-Yi Lee’s Lecture Slides

What is convolution neural network?

In the task related to images, some patterns are much smaller than the whole image and might appear in different areas. In addition, subsampling the pixels will not change the object. Unlike neural networks, where the input is a vector, here the input is a multi-channeled image. CNNs use a variation of multilayer perceptrons designed to require minimal preprocessing.

Convolution layer: These are layers consisting of entities called filters used as parameters to train the network.
ReLU: It is used as the activation function and is always used with the convolution layer.
Maxpooling: Pooling is the concept of shrinking the complex data entities that form after convolution and is primarily used to maintain an image’s size after shrinkage.
Fully connected layer: This is used to ensure that all of the layers in the

neural network are fully connected and activation can be computed using the bias easily.



The whole CNN

cat dog ......

Fully Connected Feedforward network

Convolution

Max Pooling

Convolution

Max Pooling

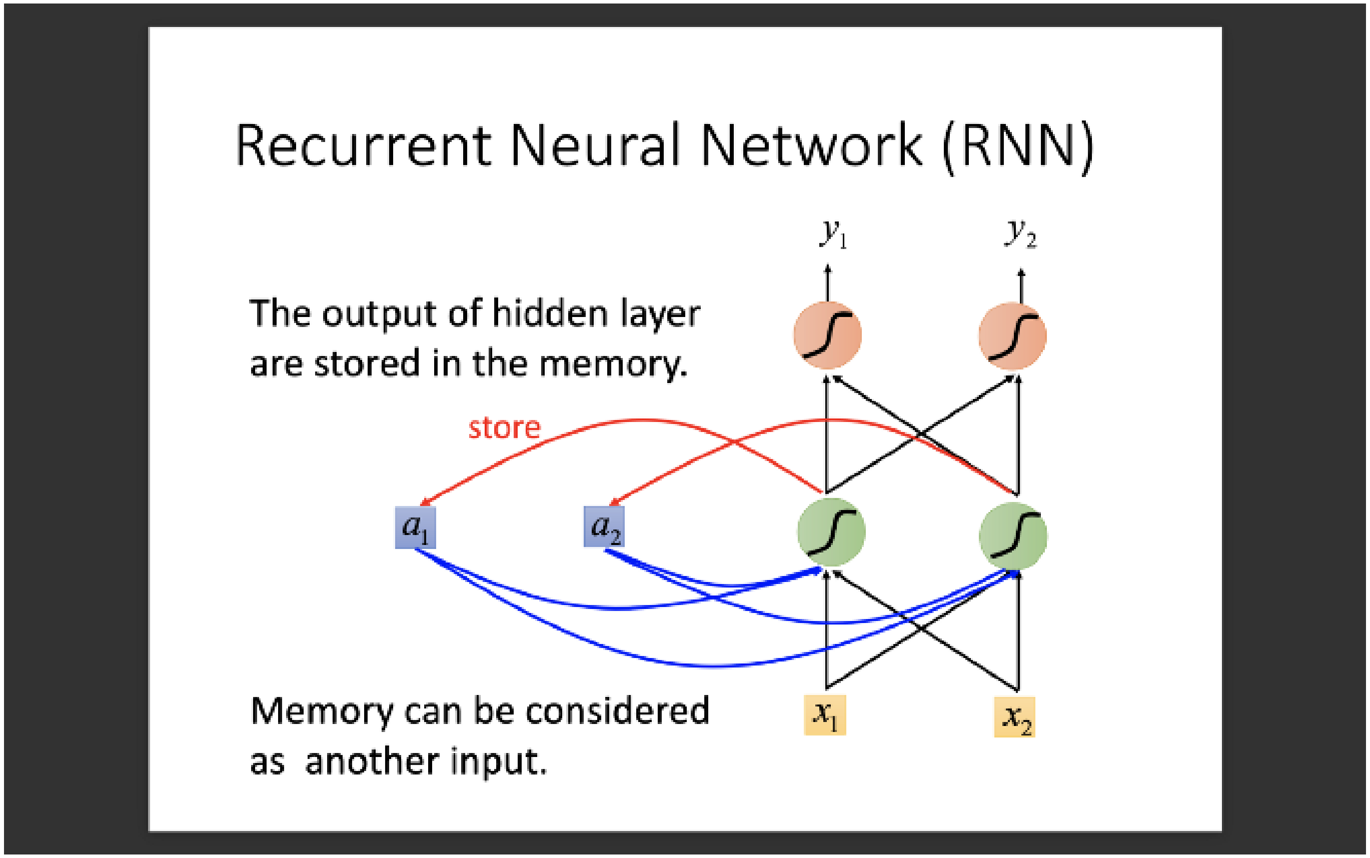
Can repeat many times

Flatten

Reference: Hung-Yi Lee's Lecture Slides

What is recurrent neural network?

We sometimes need the neural network to memorize the past values. The same network is used again and again. And change the input sequence order will change the output. RNN use backpropagation (BPTT) algorithm for training, but it is applied for every timestamp, but may causes gradient vanishing or exploding problems.



Recurrent Neural Network (RNN)

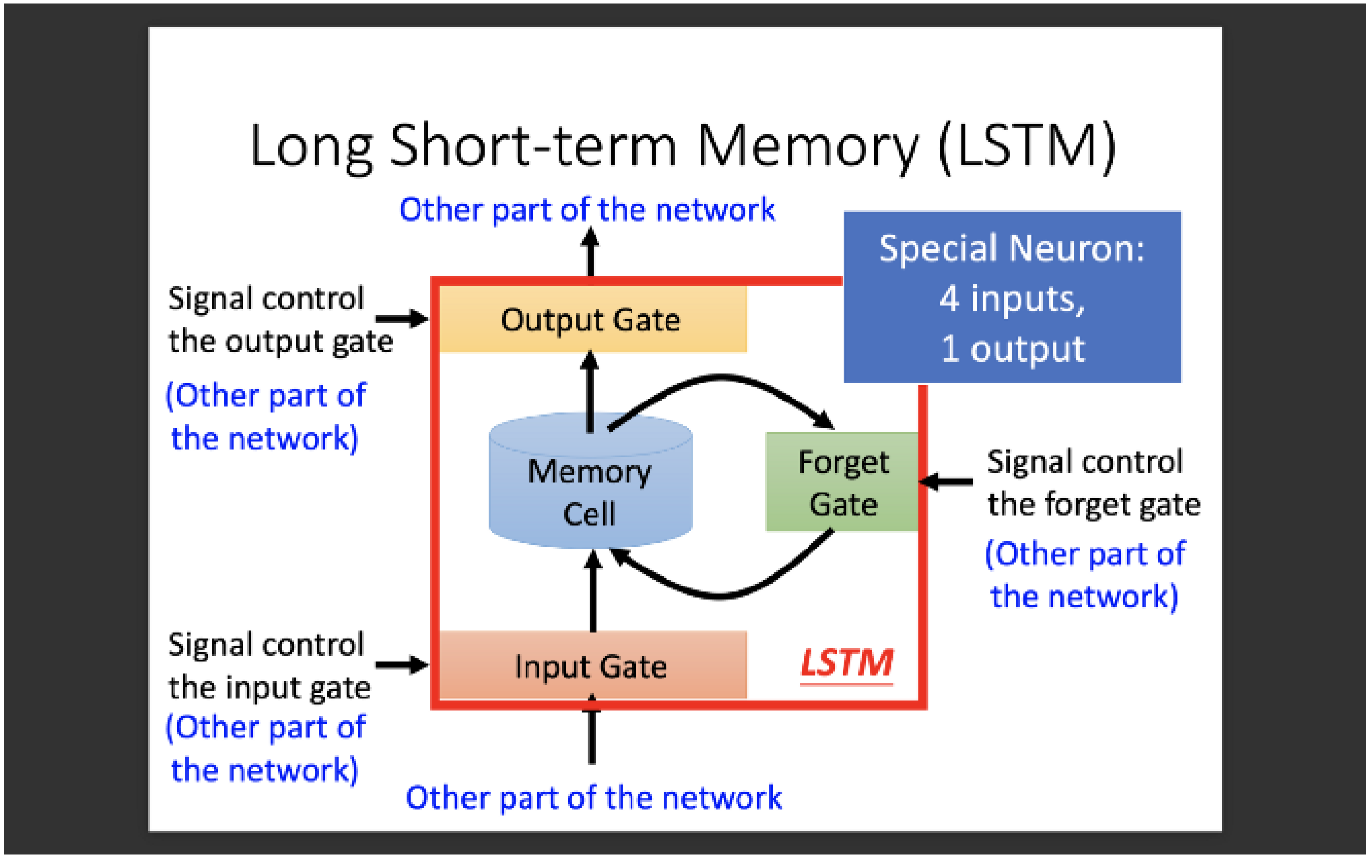
The output of hidden layer are stored in the memory.

Memory can be considered as another input.

Reference: Hung-Yi Lee's Lecture Slides

What is long short-term memory?

It is a type of RNN that is used to sequence a string of data. It consists of feedback chains that give it the ability to perform like a general-purpose computational entity. LSTM can deal with gradient vanishing (not gradient explode).



Long Short-term Memory (LSTM)

Other part of the network

Special Neuron:
4 inputs,
1 output

Signal control the output gate

(Other part of the network)

Output Gate

Memory Cell

Forget Gate

Signal control the forget gate

(Other part of the network)

Signal control the input gate

(Other part of the network)

Input Gate

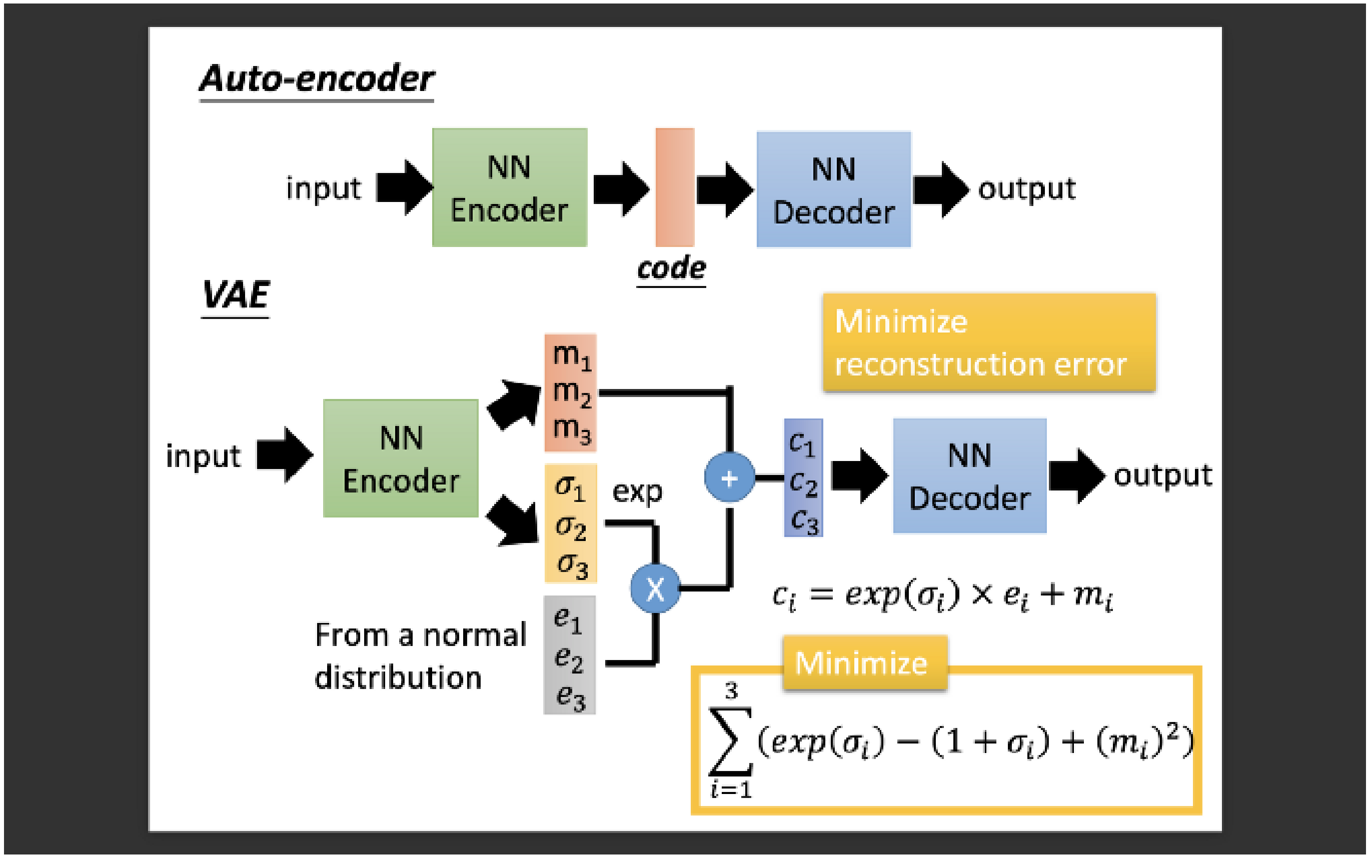
LSTM

Other part of the network

Reference: Hung-Yi Lee's Lecture Slides

What is autoencoder?

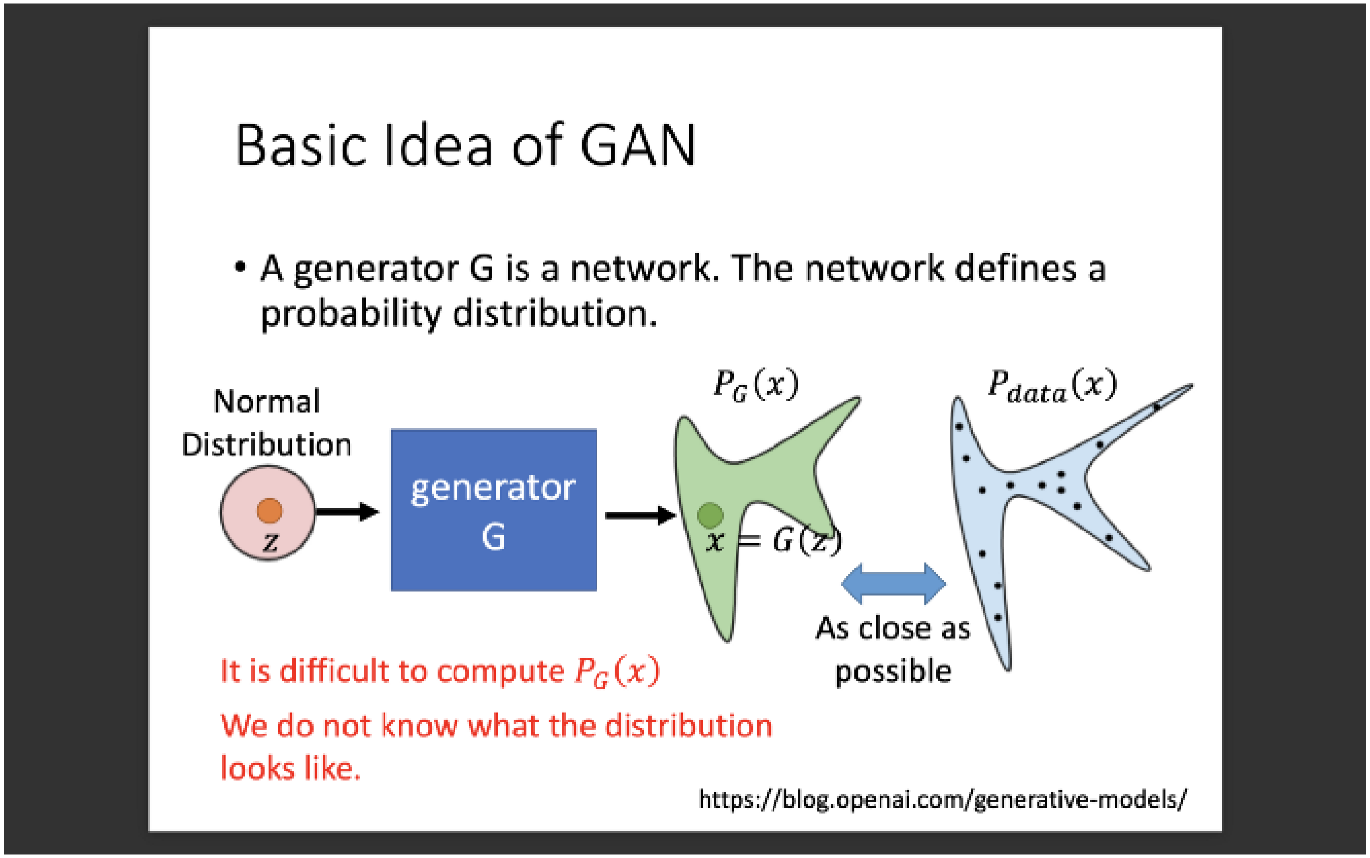
An autoencoder neural network is an unsupervised machine learning algorithm that applies backpropagation, setting the target values to be equal to the inputs. Autoencoders are used to reduce the size of our inputs into a smaller representation. If anyone needs the original data, they can reconstruct it from the compressed data.


Auto-encoder
input
NN Encoder
code
NN Decoder
output
VAE
input
NN Encoder
m_1 m_2 m_3
σ_1 σ_2 σ_3
exp
e_1 e_2 e_3
From a normal distribution
X
+
c_1 c_2 c_3
NN Decoder
output
Minimize reconstruction error
$c_i = exp(\sigma_i) \times e_i + m_i$
Minimize
$\sum_{i=1}^{3} (exp(\sigma_i) - (1 + \sigma_i) + (m_i)^2)$

Reference: Hung-Yi Lee's Lecture Slides

What is generative adversarial network?

Generative adversarial network is used to achieve generative modeling in Deep Learning. It is an unsupervised task that involves the discovery of patterns in the input data to generate the output. The generator is used to generate new examples, while the discriminator is used to classify the examples generated by the generator.



Basic Idea of GAN

- A generator G is a network. The network defines a probability distribution.

Normal Distribution

z

generator G

$P_G(x)$

$x = G(z)$

$P_{data}(x)$

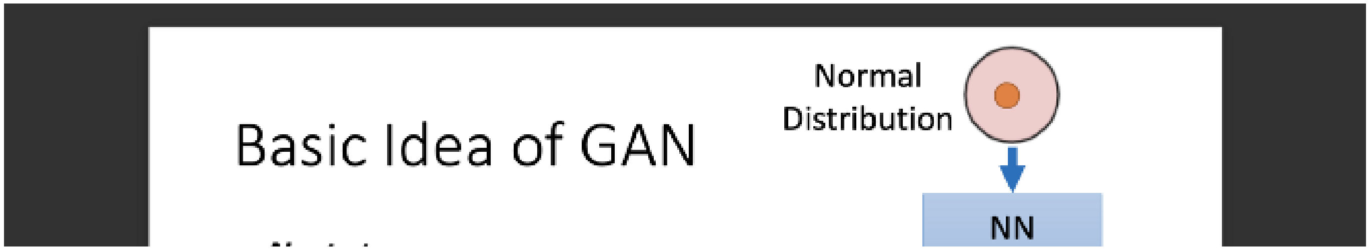
As close as possible

It is difficult to compute $P_G(x)$

We do not know what the distribution looks like.

https://blog.openai.com/generative-models/

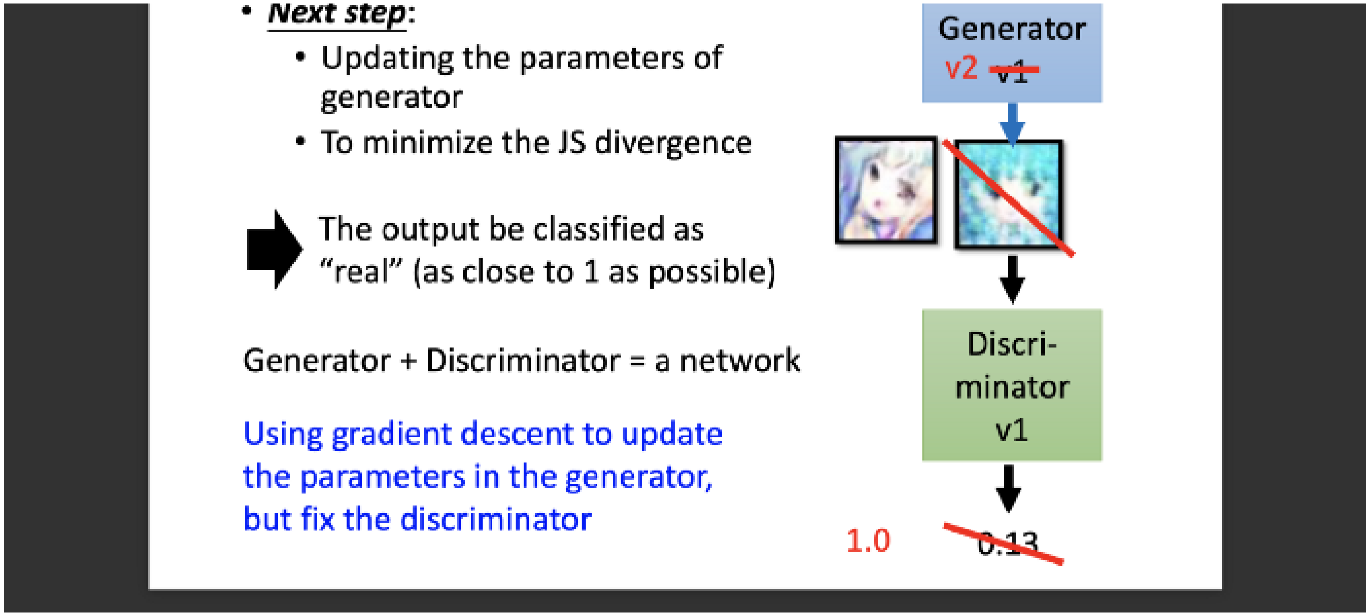
Reference: Hung-Yi Lee's Lecture Slides



Basic Idea of GAN

Normal Distribution

NN



- ***Next step***:
  - Updating the parameters of generator
  - To minimize the JS divergence

➡ The output be classified as "real" (as close to 1 as possible)

Generator + Discriminator = a network

Using gradient descent to update the parameters in the generator, but fix the discriminator

Generator v2 ~~v1~~

Discri-minator v1

1.0 ~~0.13~~

Reference: Hung-Yi Lee's Lecture Slides

Deep Learning　Interview Questions　Algorithms　Theory