

Introduction to Neural Networks

PHYS591000 2022.03.23

Outline

- Here we come: The most famous AI algorithms
- How neurons work: Activation function
- The goal of the neural network: Minimize Loss function
- Make the machine learn faster: Optimizers
- A word on Regularization

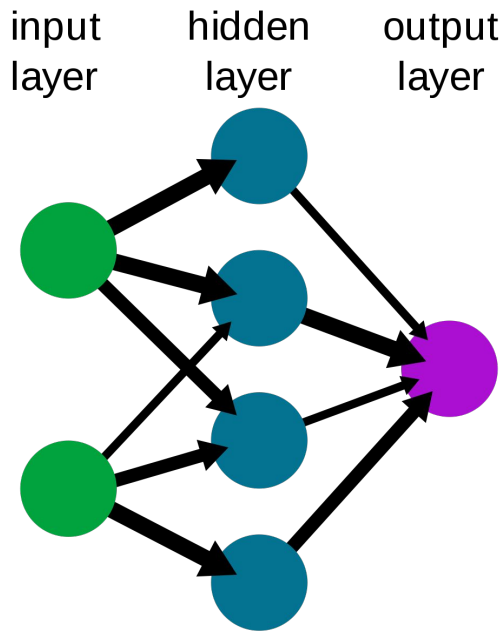
Warming up

- As usual, take 3 mins to introduce yourself to your teammate for this week!
 - “Were you alright during the earthquakes this morning?”
 - “What do you know about neural networks? Have you played with them before?”

Neural Network

- Artificial Neural Network (ANN), or just neural network (NN), is a model which tries to simulate the functions of biological neural networks.
- Each circle on the plot is a 'neuron'.
Can have multiple neurons for a layer.
Can have many hidden layers.

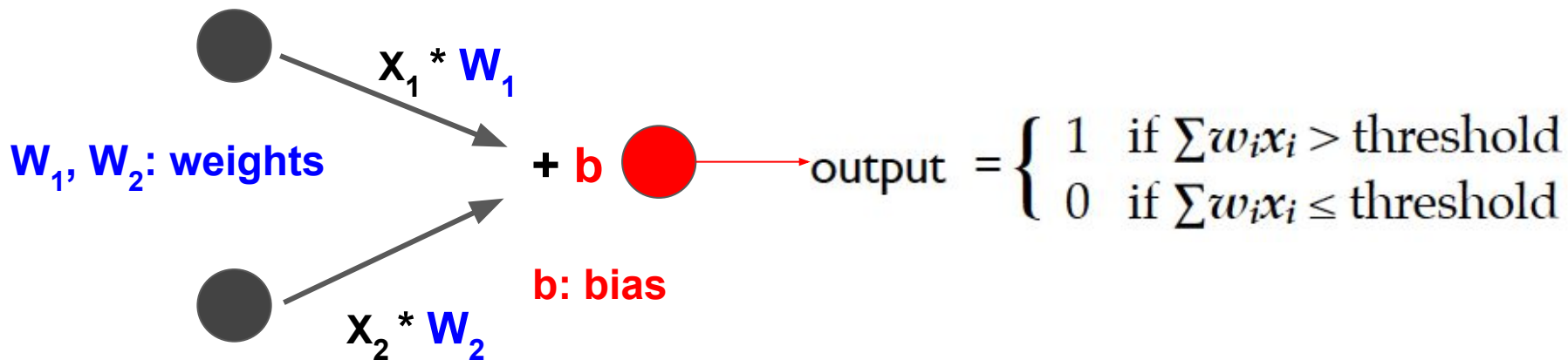
A simple neural network



Source: Wikipedia

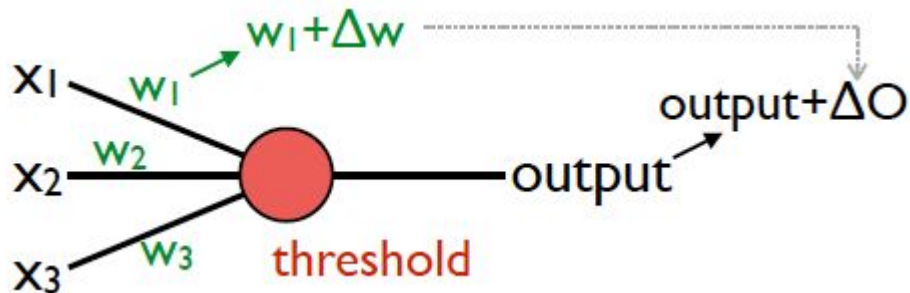
How does a neuron work?

- A neuron works like a 'switch': its output is determined by summing over inputs with different **weights** (importance of each input), plus a **bias** (a 'threshold' value for this neuron)



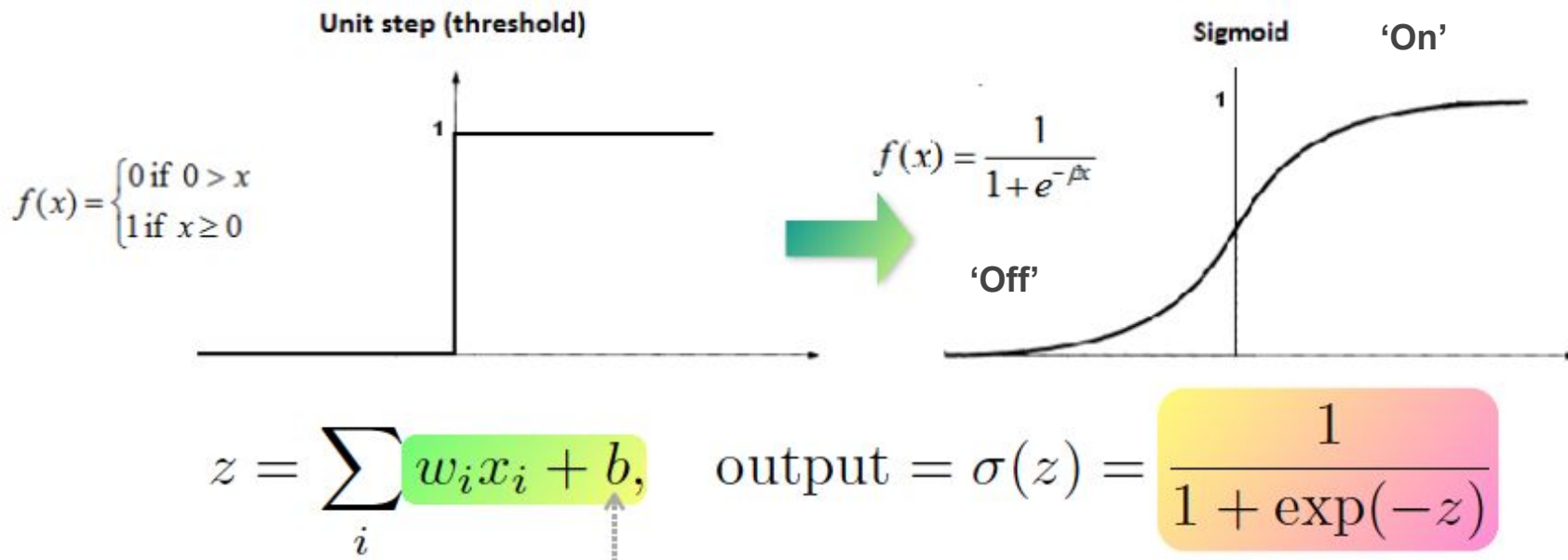
How does a neuron work?

- When training the model we want to change the output 'little by little', so a simple 0/1 binary output is not ideal.
 - We'd like a smooth output which acts like a switch (0/1)



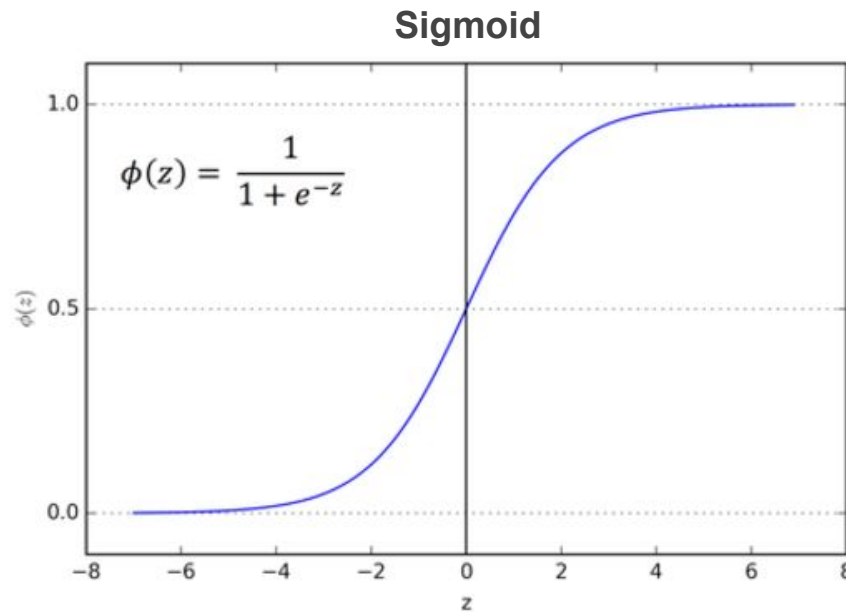
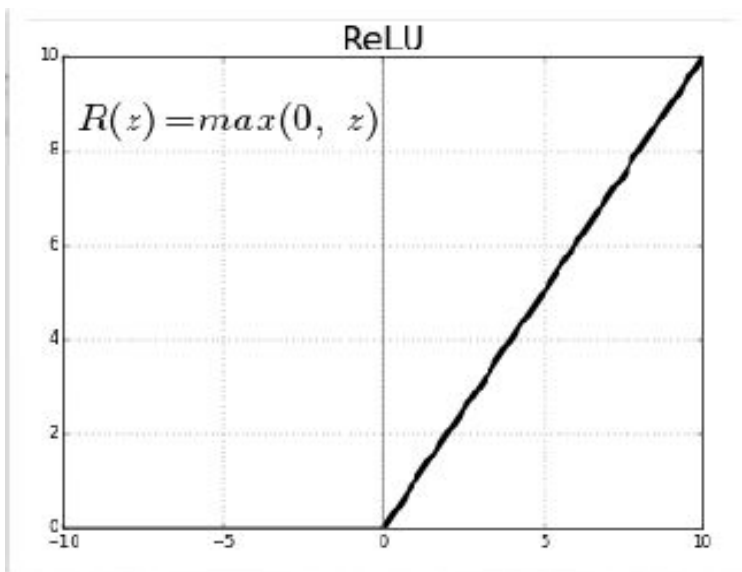
Activation Function

- Idea: Introduce an activation function that smooths the output



Activation Function

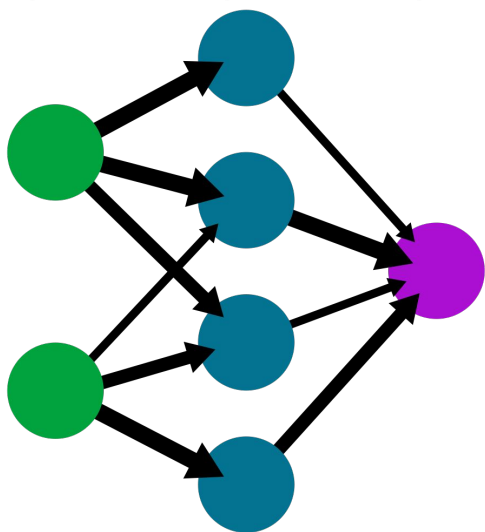
- Two common activation functions are the rectified linear unit (**ReLU**) function and **sigmoid** function



Network Architecture

A simple neural network

input layer hidden layer output layer



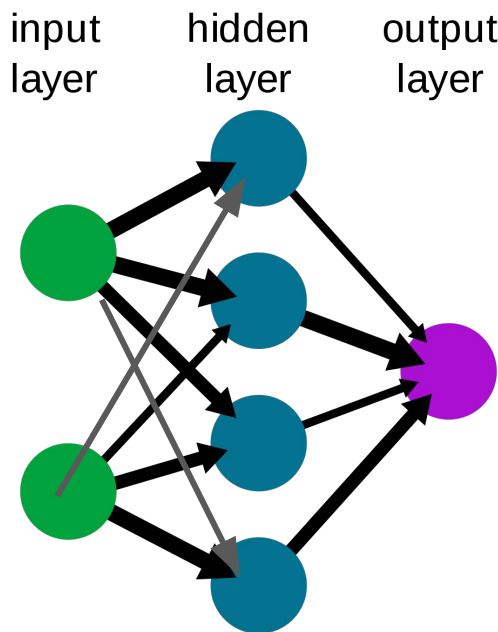
multilayer perceptrons (MLP)

- There can be multiple **input neurons**, for handling the actual input features.
- There can be multiple **hidden layers** of neurons, without connecting to the input nor output directly.
- There can be multiple **output neurons** as well.

Courtesy of Prof. Kai-Feng Chen (NTU)

Network Architecture

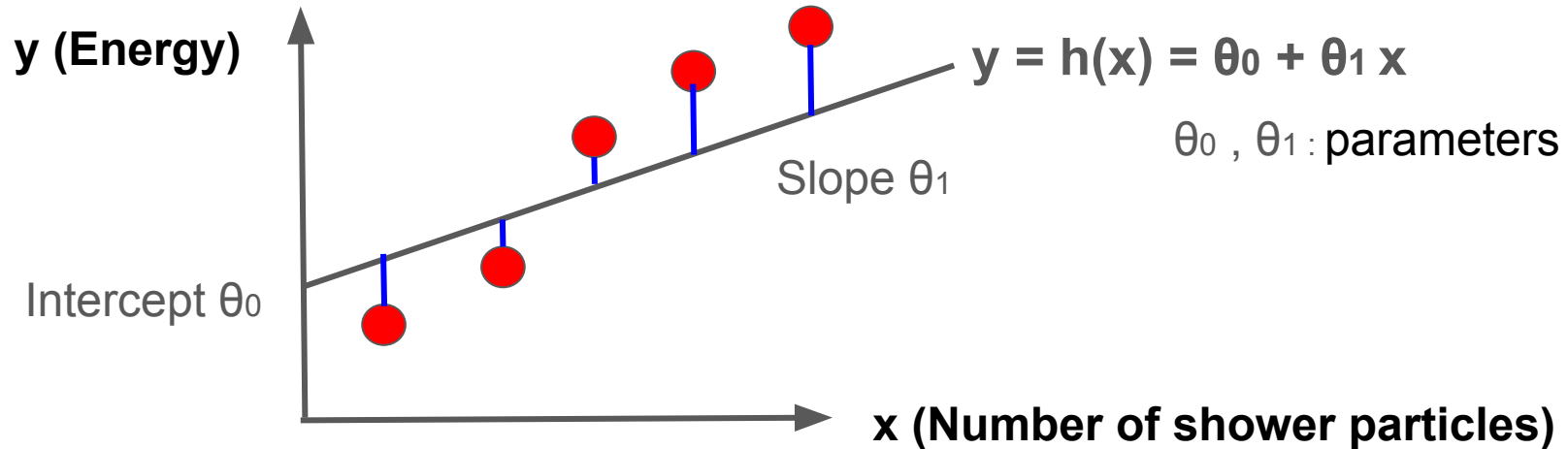
A simple neural network



- If all neurons are connected to every neuron in the next layer, it is called a **fully-connected** network.
- If the information always flows from left to right (input \rightarrow hidden layer 1 \rightarrow ... \rightarrow output layer), this is called a **feedforward** network.

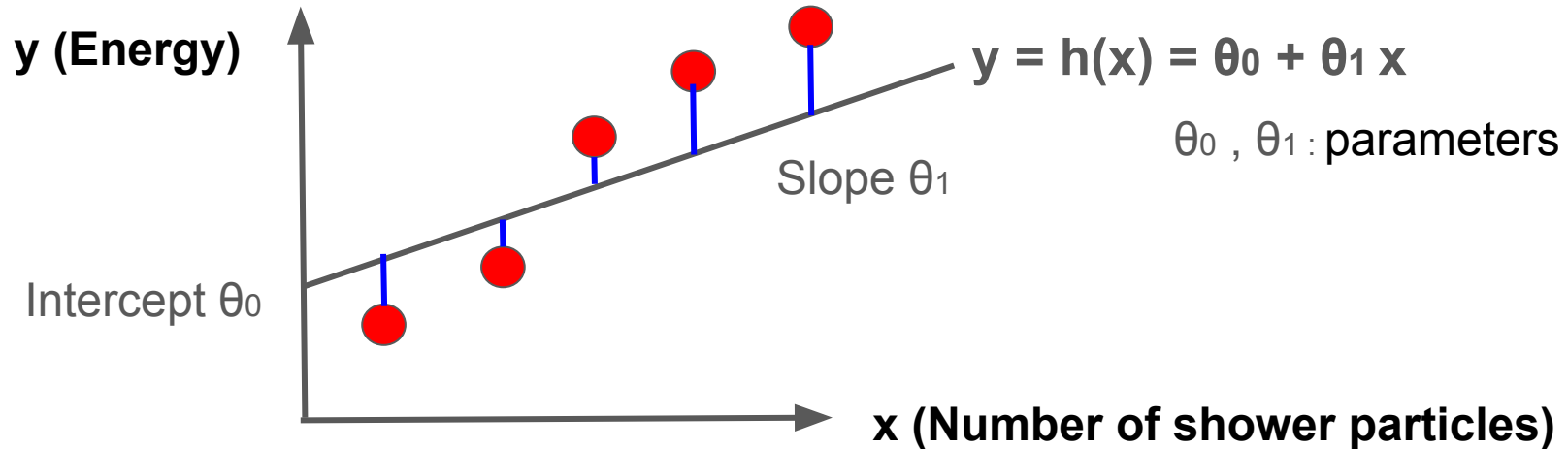
Review: Goal of training a model

- Recall in Regression the way to find the line (parameters) is to minimize the function $\sum_i (h(x_i) - y_i)^2 \rightarrow$ method of least squares



Review: Goal of training a model

- $\sum_i (h(x_i) - y_i)^2$ is **loss function**. The goal of training is to find the optimal parameters which minimize the loss function.

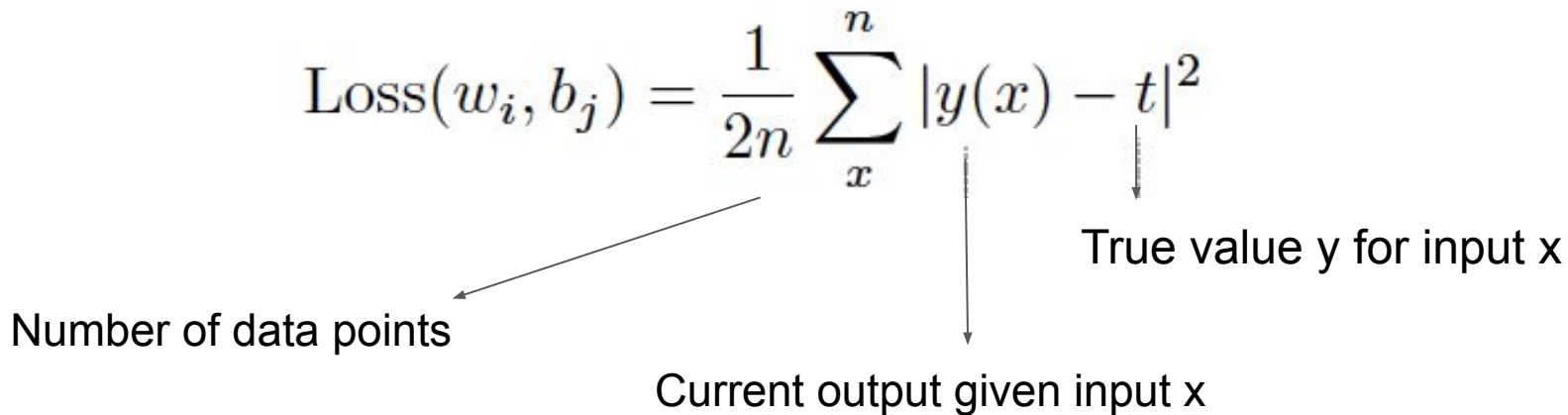


How to Train Neural Network

- Same for NN: The goal of training is to find optimal weights (w_i) and biases (b_j) which minimize the loss function. A typical choice of loss function is the mean square error (MSE):

$$\text{Loss}(w_i, b_j) = \frac{1}{2n} \sum_x^n |y(x) - t|^2$$

Number of data points

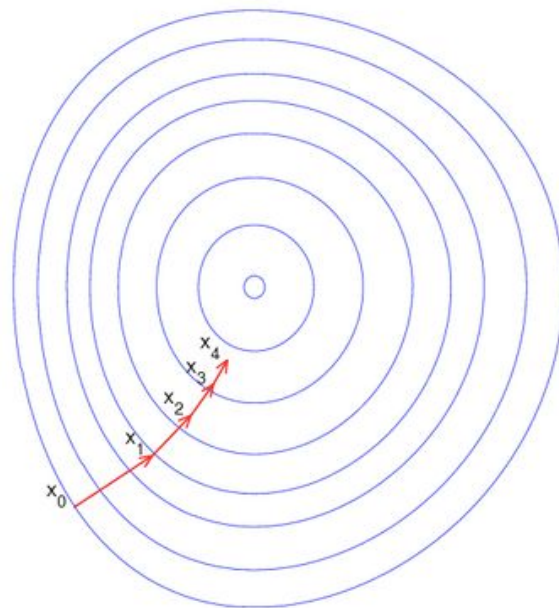


Current output given input x

True value y for input x

How to Train Neural Network

- If we need to optimize a lot of w_i and b_j we need a smart algorithm to do so, e.g., the method of *gradient descent*



How to minimize the loss function

- Method of gradient descent: The next step is proportional to the negative of the local gradient

Gradient of Loss (average over the input data) $\nabla L = \frac{1}{n} \sum_x^n \nabla L_x$

$$w_i \rightarrow w'_i = w_i - \eta \frac{\partial L}{\partial w_i}$$

$$b_j \rightarrow b'_j = b_j - \eta \frac{\partial L}{\partial b_j}$$

η : learning rate

a tunable hyperparameter

Optimizer/Solver for minimizing loss function

- When the input data size is large it will take a lot of time calculating the gradients, and make the NN too slow. There are several **optimizer**/solver to speed up the learning process.
- A common solution is to use a random subset of input data instead: Gradient of Loss approximated by average over a subset of input data

$$\nabla L \approx \frac{1}{m} \sum_{x_k}^m \nabla L_{x_k}$$

The method is called **stochastic gradient descent (SGD)**.

Optimizer choice

- SGD is an easy and useful algorithm for minimizing the loss function, but there can be problems like slow training.
- Many other choices of optimizers. E.g.
 - Adam: Increase the 'step size' (learning rate) when gradient becomes smaller
 - Adagrad: Regularized learning rate – Larger/smaller gradient would give smaller/larger learning rate.
 - Adadelta : extended Adagrad with reduced dependence to the global learning rate.

Training Process hyperparameters

- This subset is called **mini-batch**. We can specify how many samples to process at one time ('batch size').
- **Epoch**: One cycle through the **full** training dataset (all subsets in batches). Training a NN usually requires $O(10-1000)$ epochs.

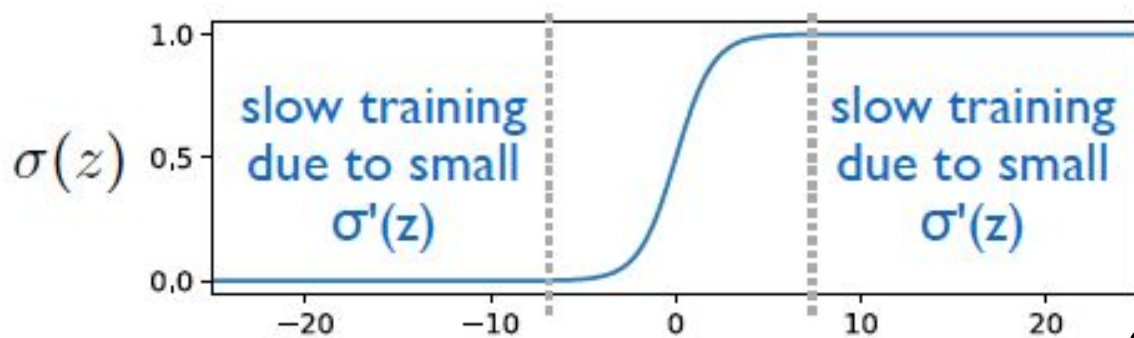
Neural Network Parameters

- When you train a NN you have to decide
 - Number of hidden layers and number of neurons for each hidden layer
 - Which activation function to use
 - Which loss function to use
 - How to minimize the loss function (optimizer)
 - ...

Choice of Loss Function

- Training with MSE can be slow since the gradient depends on the slope of the activation function $\sigma(z)$:

$$L(w, b) = \frac{1}{2} |\sigma(z) - t|^2 \quad \longrightarrow \quad \frac{\partial L}{\partial w} = [\sigma(z) - t] \sigma'(z) \frac{\partial z}{\partial w}$$



Choice of Loss Function

- Another choice of loss functions is the **cross entropy function**:
The form of binary (two-class) cross entropy looks like

$$Loss(w_i, b_j) = \frac{-1}{n} \sum_x^n [t \ln y + (1 - t) \ln(1 - y)]$$

$$\longrightarrow \frac{\partial L}{\partial w} = [\sigma(z) - t] \frac{\partial z}{\partial w}$$

Not depending on the first derivative $\sigma'(\mathbf{z})$ anymore!

Courtesy of Prof. Kai-Feng Chen (NTU)

Choice of Loss Function and Output Layer

- For multi-classification we can use categorical cross entropy:

$$L = - \sum_j t_j \ln(y_j) \quad j: \text{classes}$$

- Which often combines with an *output layer* using **Softmax**:

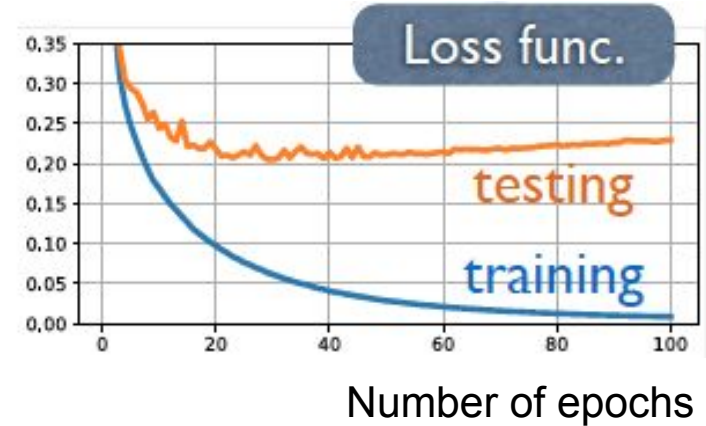
$$y_j = \frac{\exp(z_j)}{\sum_k \exp(z_k)} \quad k: \text{classes}$$

Choice of Loss Function and Output Layer

- Categorical cross entropy loss + softmax output again makes the gradient independent of the first derivative of the activation function, and speeds up the learning.
- Furthermore, the softmax output is not just 0 or 1 (yes or no for one class) – it's continuous between 0 to 1, and thus can be interpreted as *probability* of being in this class.

Regularization

- As usual we need to be careful about overtraining (overfitting)
- L1/L2 regularization: Add extra term $\lambda \sum_i |\mathbf{W}_i|$ (L1) or $\lambda \sum_i \mathbf{W}_i^2$ (L2) to the loss function



In-class exercise



- We'll use Keras with Tensorflow backend on the MNIST data!
 - Tensorflow: an open source software library for high performance numerical computation originally developed by Google.
 - Keras: a kind of “wrapper” or package which can help to build most of the conventional NN models, and the real calculation can be carried out by TensorFlow, as one of the supported backends.

In-class exercise

- We'll build a 784-30-10 NN:

of biases:

0 (no biases for input)+

30 (hidden neurons)+

10 (output neurons).

of weights:

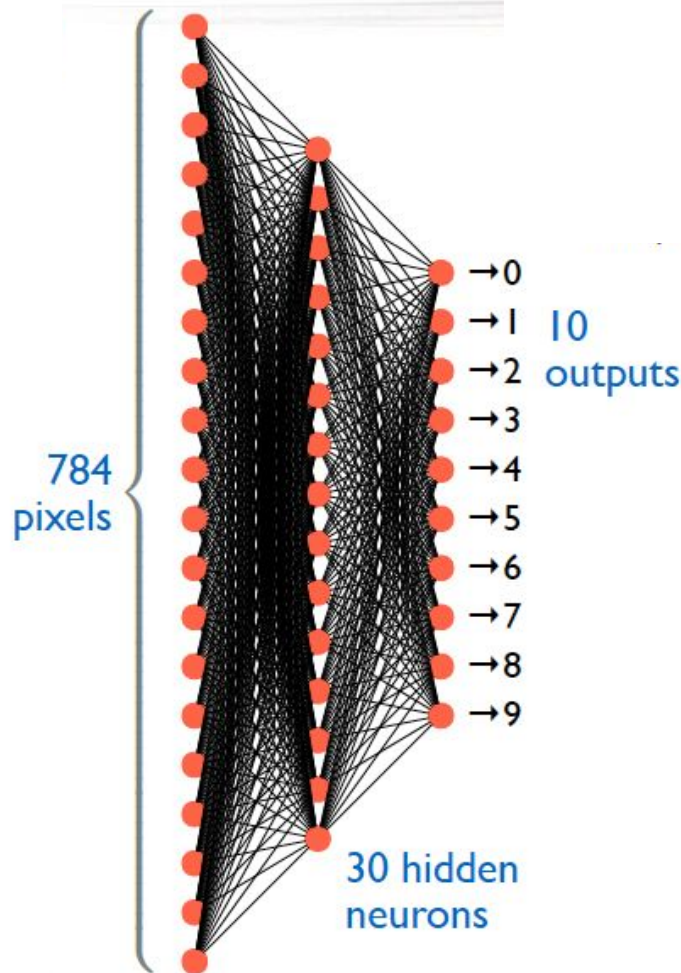
784×30 (input to hidden)+

30×10 (hidden to output).

23860 parameters in total



Lots of parameters to be optimized in the training!



Lab for this week

- For the Lab this week, we'll apply what we've learned about NN to a binary classification task: classify jets from
 - 1) W/Z (a kind of heavy gauge boson) bosons
 - 2) light quark or gluons ('QCD')
- You'll compare results with different hyperparameters and see what you can learn about *hyperparameter tuning*.

Backup

Neural Network Information Flow

- If the NN processes the information only in one way (input \rightarrow output without any loops) it is called *feedforward*.
- Focus on feedforward NN now. Later we'll talk about RNN (recurrent NN) which is non-feedforward.

