

AI Engineer Training: IV

In the Era of Deep Learning

IT21 Learning
Alvin Jin

Weekly AI News

- Nvidia opened AI research centre in Toronto.
- Google published AI principles, and leaving the Maven project.
- India released national AI strategy

www.it21learning.com

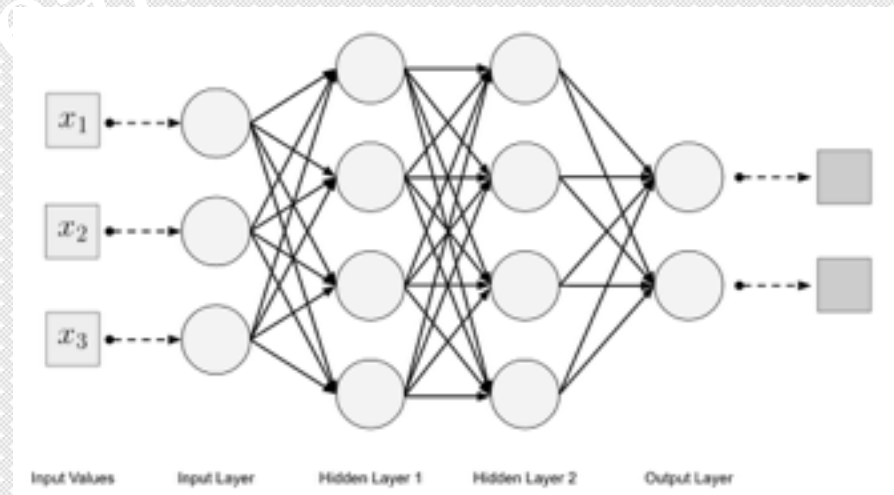
Agenda

- Artificial Neural Network
- Case Studies:
 - Handwritten Digits Recognition

www.it21learning.com

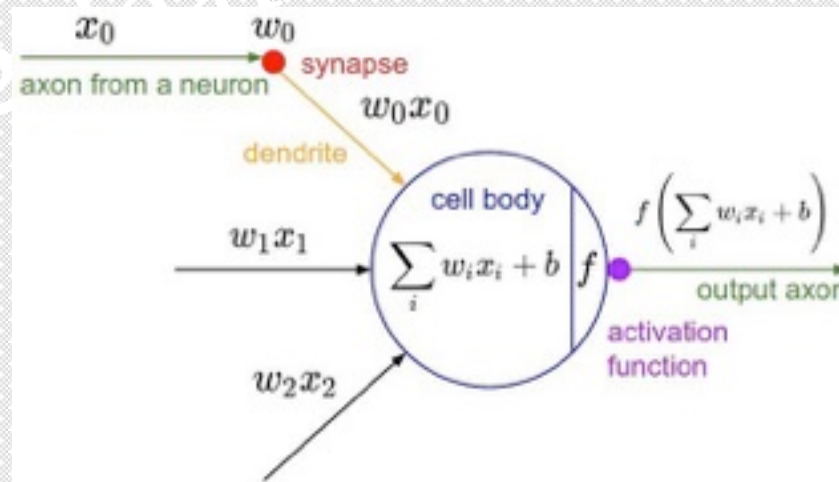
Artificial Neural Network

- Many simple units are working in parallel with no centralized control unit.
- ANN consists of input/output layers and hidden layers.
- The units in each layer are fully connected to all units in the adjacent layer.



Neurons(Units)

- Neuron is a mathematical function and fundamental processing element of an ANN.
- Each neuron in an ANN does a weighted sum of all of its inputs, adds a constant bias and then feeds the result through non-linear activation function.



Layers

- A layer is a data-processing module takes tensors as input, and output tensors.
- Hidden layers are layers not observable outside the network.
- Most layers have weights, while some don't have.
- Each layer have different number of units.

Connection Weights & Biases

- The coefficients that amplify the input signal and dampen the noise to units in the network.
- By making some weights smaller and others larger, significance certain features and minimize others, learn which ones are useful to the outcomes.
- Biases are scalar values added to the input to ensure that at least a few units per layer are activated regardless of signal strength.
- Essentially, ANN is to optimize the weights and biases to minimize error

Activation Functions

- When a unit passes on a nonzero value to another unit, it is said to be activated.
- Activation functions transform the combination of inputs, weights, and biases from one layer to the next layer.
- Introducing nonlinearity into the network's modelling capabilities.

Popular Activation Functions

- Sigmoid
- Tanh
- ReLU
- Leaky ReLU
- SoftPlus
- Softmax

www.it21learning.com

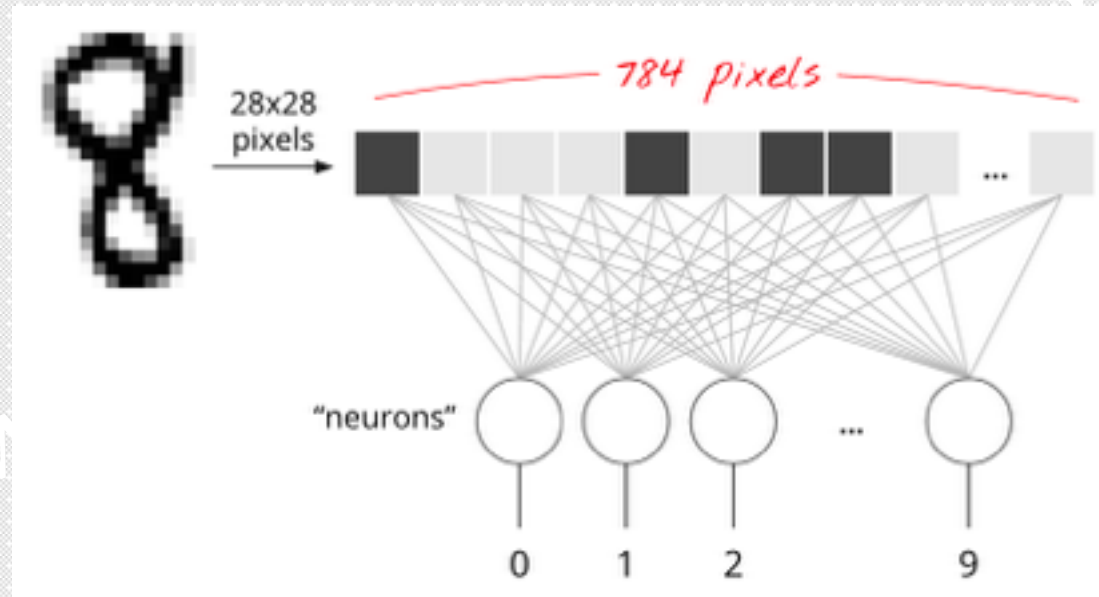
Handwritten Digit Classification

- MNIST dataset assembled in 1980s by LeCun, etc.
- 70,000 sample images with 28 x 28 pixel square
- Multi-classes classification problem

www.it21learning.com

Input Layer

- Use the $28 \times 28 = 784$ pixels as input vector in the input layer

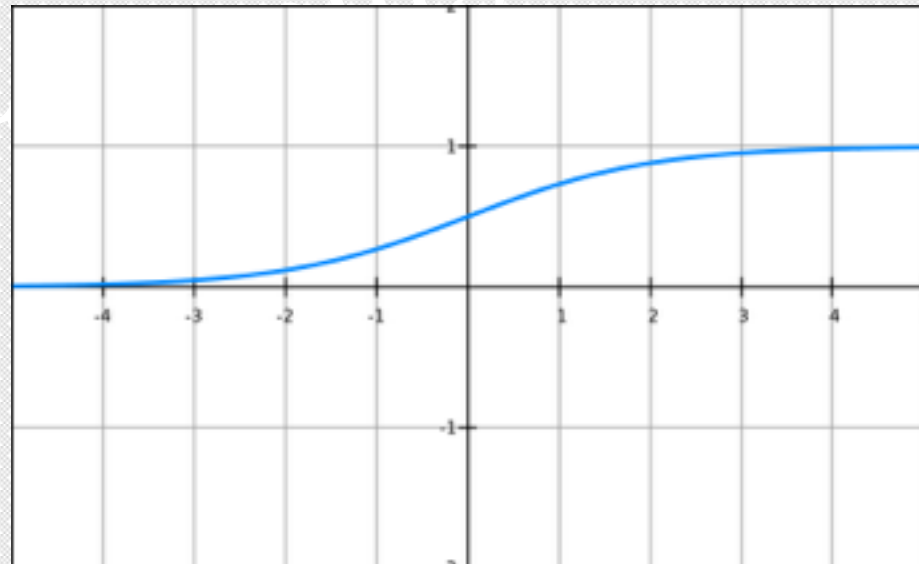


Sigmoid Functions

- A logistic functions to predict the probability of the output by converting arbitrary values into $[0, 1]$.
- Every time the gradient signal flows through a sigmoid gate, its magnitude always diminishes by at least 0.25

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

$$f'(x) = f(x)(1 - f(x))$$

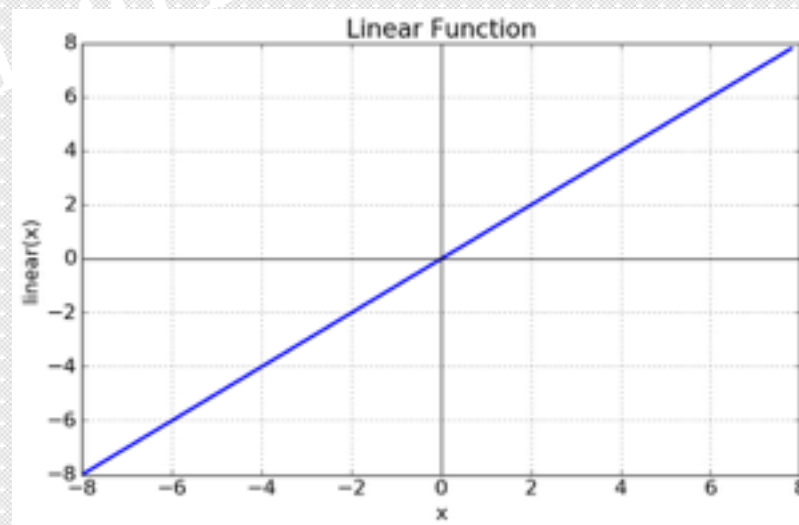


Linear Function

- A straight line function where activation is proportional to input signal.
- The derivative is a constant, means the gradient has no relationship with inputs.
- If all layers are linear, the activation function of last layer is a linear of the input of the first layer.

$$f(x) = x$$

$$f'(x) = 1$$



Softmax Functions

- Softmax returns the probability distribution over mutually exclusive output classes.
- The result vector clearly shows the max element, which close to 1, and retains the order.

$$S(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

Output Layer

- Softmax is used on the output layer to turn the outputs into probability-like values

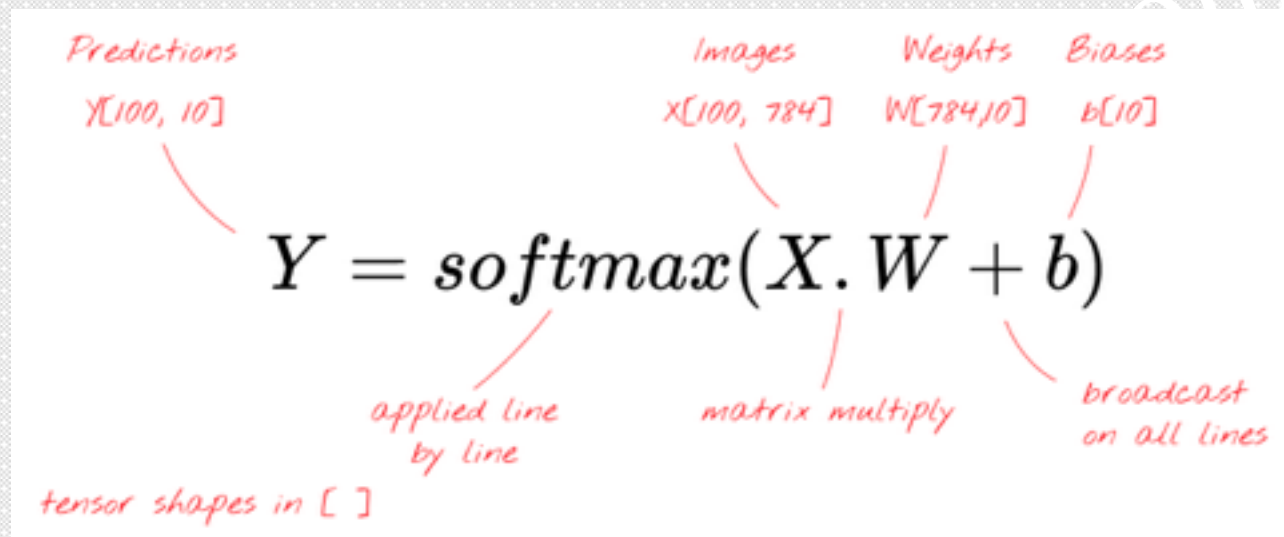


Diagram illustrating the output layer calculation:

$$Y = \text{softmax}(X.W + b)$$

Annotations (handwritten in red):

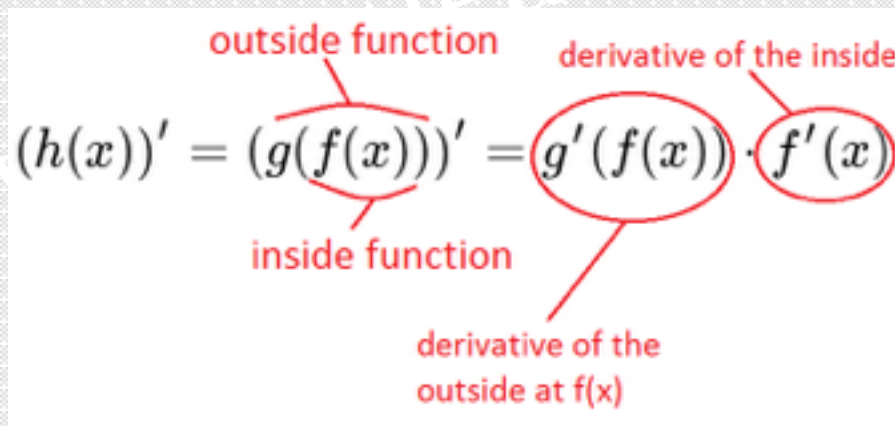
- Predictions**: $Y[100, 10]$
- Images**: $X[100, 784]$
- Weights**: $W[784, 10]$
- Biases**: $b[10]$
- applied line by line**: points to the softmax function
- matrix multiply**: points to the $X.W$ term
- broadcast on all lines**: points to the $+ b$ term
- tensor shapes in []**: points to the shapes in the tensors

ANN Model Training

- Compute the partial derivatives of the loss function relatively to all the weights and biases(millions of parameters) to obtain gradient.
- Update the weights and biases by a fraction of the gradient(learning rate), repeatedly after each batch, to minimize the loss function

Chain Rules

- Each layer in NN takes the output of previous layer's function, and using it as input into its function.
- Chain rule is to calculate the gradient of "function of a function"
- A NN consists of many tensor operations chained together, e.g. $h(x) = g(f(x))$


$$(h(x))' = (g(f(x)))' = g'(f(x)) \cdot f'(x)$$

outside function

derivative of the inside

inside function

derivative of the outside at $f(x)$

Back-Propagation

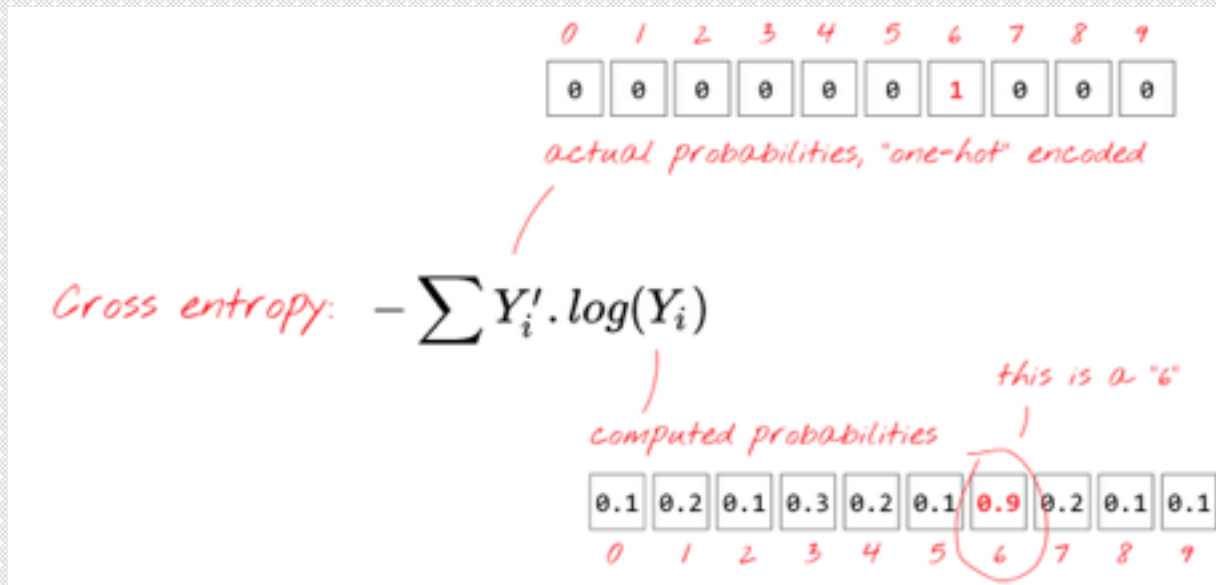
- Applying the chain rule to compute gradient values of a NN is called Backprop.
- Starts with the final loss value and works backward from the top layers(output layer) to the bottom layers(input layer)
- Based on Chain Rules, calculate the derivative at any layer by simply multiplying the gradients of that layer and all of its following layers together

Back-Propagation Algorithm

- Uses gradient descent on the weights of the connections in a NN to minimize the error on the output.
- Walk back across the hidden layers, updating the connection between each one until reaching the input layer.
- Proposed at 1980s, but the computation is slow. Distributed systems, GPU make it attractive again.

Categorical Cross Entropy

- Measures the performance of multi-classification model whose output is a probability in [0,1]



Q & A

Categorical Cross Entropy			
Label	e^{score} (unnormalized prob)	score	
0	0.1	0.043478261	
1	0.2	0.086956522	
2	0.1	0.043478261	
3	0.3	0.130434783	
4	0.2	0.086956522	
5	0.1	0.043478261	
6	0.9	0.391304348	
7	0.2	0.086956522	
8	0.1	0.043478261	
9	0.1	0.043478261	
sum of unnormalized pro		sum	
	2.3	1	

VS

Softmax Function

$$S(y_i) = \frac{\exp(y_i)}{\sum_j \exp(y_j)}$$