Deep Learning School

Lesson 6
Multilayer Neural Networks

Plan

1. Recap: Neuron
2. Multilayer Perceptron
   - Input Layer
   - Hidden Layers
   - Output Layer
3. Multiclass Classification
   - One-hot Encoding
   - Softmax
   - Cross-Entropy Loss
4. Backpropagation

Artificial Neuron

- Input: vector **x** with the size (1, *num_features*)

- Neuron parameters:
  - weight vector **w** with the size (1, *num_features*)
  - the real number **b** (bias)

- Activation function: **f** (Sigmoid, ReLU, Swish, MaxOut...)

- Output: **y** -- class label (0 or 1 in the binary case)

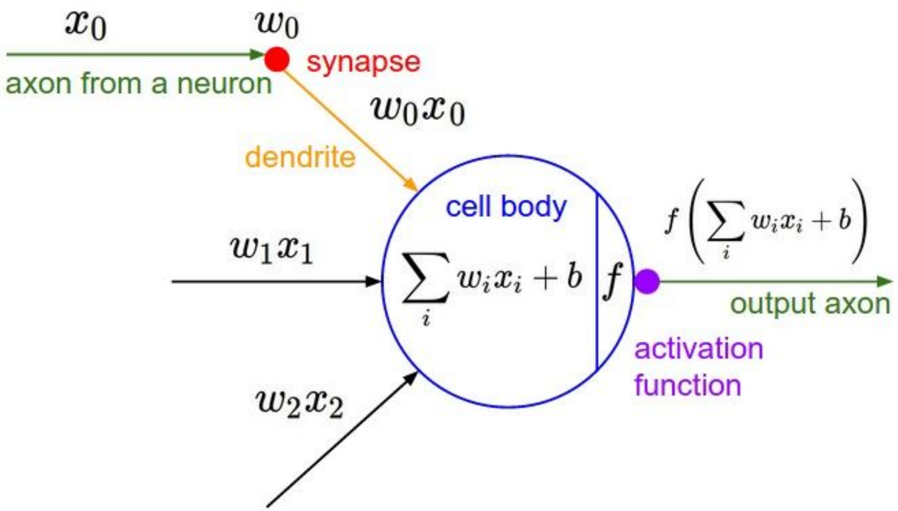


*image: https://compsci682.github.io/assets/nn1/neuron_model.jpeg

Artificial Neuron: training

- Loss function (MSE, LogLoss...)
- Supervised learning (we have the true labels)
- Optimize with gradient descent (or its variants)

$$w^{j+1} = w^j - \alpha \frac{\partial Loss}{\partial w}(w^j)$$

Multilayer Fully-Connected Network

- Input layer: **X** (feature matrix)
- Hidden layers: weight matrices
- Output layer: answers matrix **y**
    - class label (classification task)
    - real number (regression task)

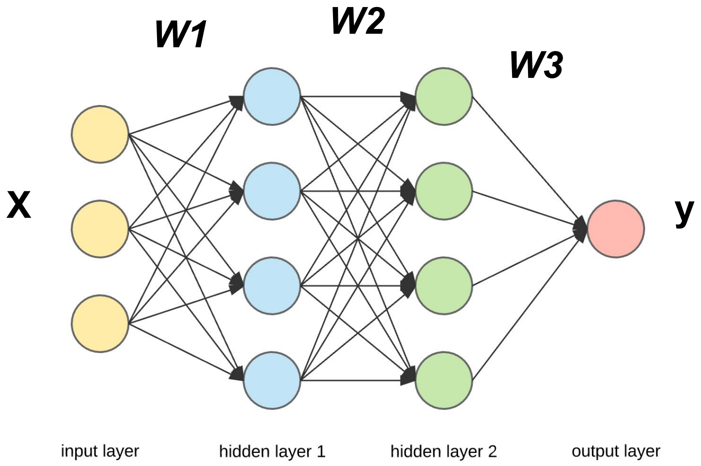


*image: https://www.armortechs.com/upload/image/neural-network.png

Input Layer

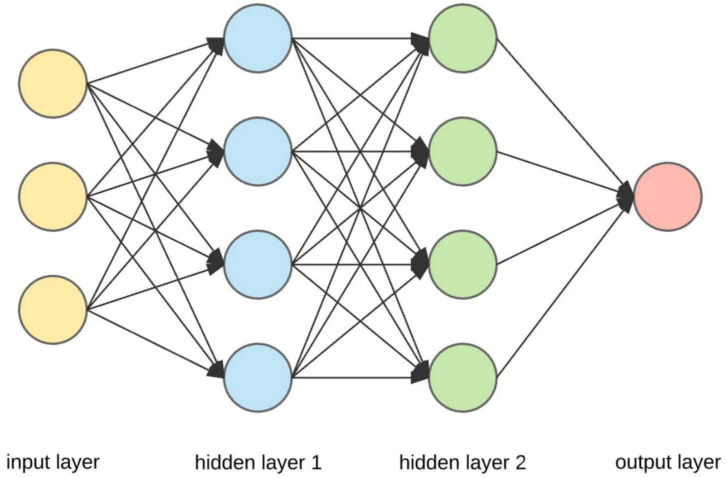


*image: https://www.armortechs.com/upload/image/neural-network.png

Hidden Layers

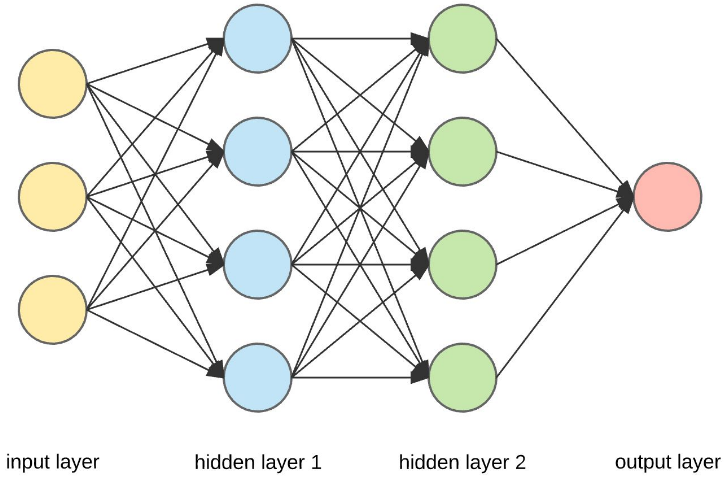


*image: https://www.armortechs.com/upload/image/neural-network.png

Hidden Layers

Hidden Layers

Deep Learning School

Lesson 6
Multiclass Classification

Multiclass classification

Dunker = 0

Estonian Hound = 1

East Siberian Laika = 2

https://en.wikipedia.org/wiki/List_of_dog_breeds

Multiclass classification

https://en.wikipedia.org/wiki/List_of_dog_breeds

One-hot Encoding

Activations of the Output layer

Softmax

$$softmax(z_i) = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

Output Layer: multiclass classification

- K classes = K neurons in the Output layer
- Output layer itself doesn't have any activation function
- Softmax function to predict the "class probability distribution"

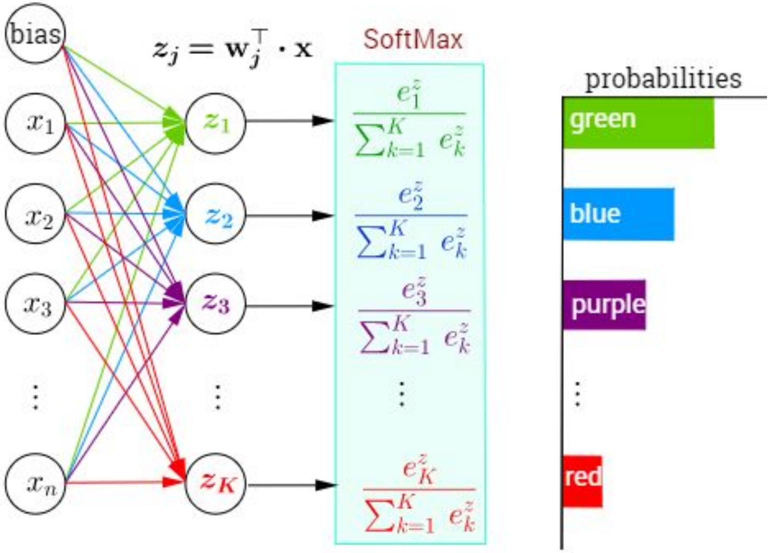


*image: https://deepnotes.io/public/images/softmax.png

Forward pass

https://en.wikipedia.org/wiki/List_of_dog_breeds

MLP: training

- Two class probability distributions:
  predicted distribution and true distribution

- Difference measure is needed

- -> Cross-Entropy Loss

$$\text{cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} t_{i,j}\log(p_{i,j})$$

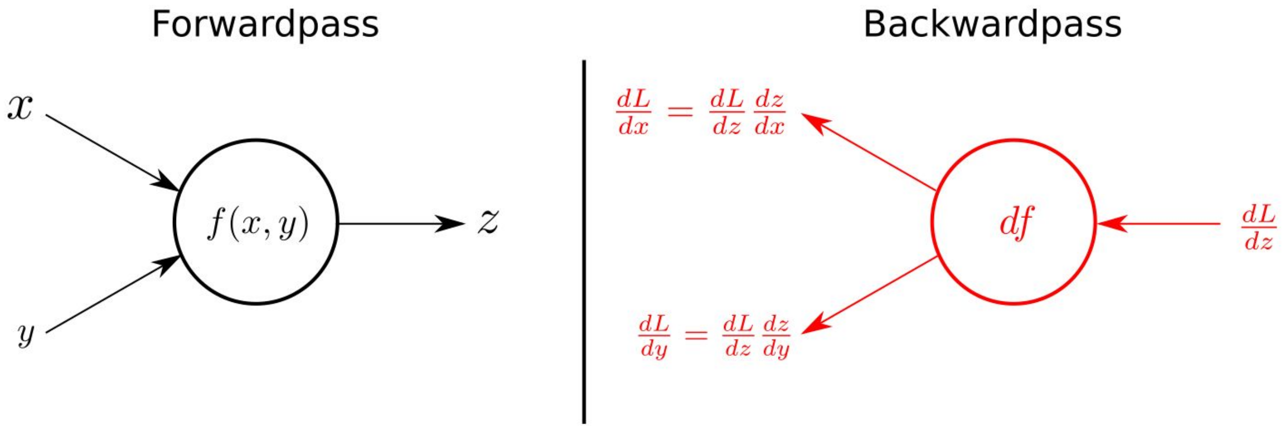
Cross-Entropy Loss

$$\text{cross-entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{k} t_{i,j}\log(p_{i,j})$$

Deep Learning School

Lesson 6
Backpropagation

Backpropagation



L - loss function

https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199

Backpropagation

https://en.wikipedia.org/wiki/Backpropagation

Backpropagation

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j}\frac{\partial o_j}{\partial \mathrm{net}_j}\frac{\partial \mathrm{net}_j}{\partial w_{ij}}$$

https://en.wikipedia.org/wiki/Backpropagation

Backpropagation

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \mathrm{net}_j} \frac{\partial \mathrm{net}_j}{\partial w_{ij}}$$

$$\frac{\partial \mathrm{net}_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^{n} w_{kj} o_k \right) = \frac{\partial}{\partial w_{ij}} w_{ij} o_i = o_i .$$

$$\frac{\partial o_j}{\partial \mathrm{net}_j} = \frac{\partial}{\partial \mathrm{net}_j} \varphi(\mathrm{net}_j) = \varphi(\mathrm{net}_j)(1 - \varphi(\mathrm{net}_j))$$

https://en.wikipedia.org/wiki/Backpropagation

Backpropagation

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \mathrm{net}_j} \frac{\partial \mathrm{net}_j}{\partial w_{ij}}$$

$$\frac{\partial \mathrm{net}_j}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \left(\sum_{k=1}^{n} w_{kj} o_k \right) = \frac{\partial}{\partial w_{ij}} w_{ij} o_i = o_i .$$

$$\frac{\partial o_j}{\partial \mathrm{net}_j} = \frac{\partial}{\partial \mathrm{net}_j} \varphi(\mathrm{net}_j) = \varphi(\mathrm{net}_j)(1 - \varphi(\mathrm{net}_j))$$

$$\frac{\partial E(o_j)}{\partial o_j} = \frac{\partial E(\mathrm{net}_u, \mathrm{net}_v, \dots, \mathrm{net}_w)}{\partial o_j}$$

$$\frac{\partial E}{\partial o_j} = \sum_{\ell \in L} \left(\frac{\partial E}{\partial \mathrm{net}_\ell} \frac{\partial \mathrm{net}_\ell}{\partial o_j} \right) = \sum_{\ell \in L} \left(\frac{\partial E}{\partial o_\ell} \frac{\partial o_\ell}{\partial \mathrm{net}_\ell} w_{j\ell} \right)$$

https://en.wikipedia.org/wiki/Backpropagation

Backpropagation

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \mathrm{net}_j} \frac{\partial \mathrm{net}_j}{\partial w_{ij}}$$

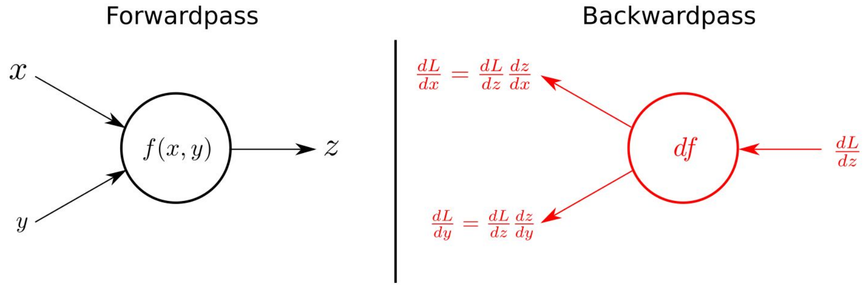
$$\frac{\partial E}{\partial w_{ij}} = \delta_j o_i$$

$$\delta_j = \frac{\partial E}{\partial o_j} \frac{\partial o_j}{\partial \mathrm{net}_j} = \begin{cases} (o_j - t_j) o_j (1 - o_j) & \text{if } j \text{ is an output neuron,} \\ (\sum_{\ell \in L} w_{j\ell} \delta_\ell) o_j (1 - o_j) & \text{if } j \text{ is an inner neuron.} \end{cases}$$

https://en.wikipedia.org/wiki/Backpropagation

Backpropagation

- Trains the network as a whole system
- Uses Chain rule
- Calculates the gradients effectively for each neuron



Good visualization of the backprop:
https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/

https://becominghuman.ai/back-propagation-in-convolutional-neural-networks-intuition-and-code-714ef1c38199