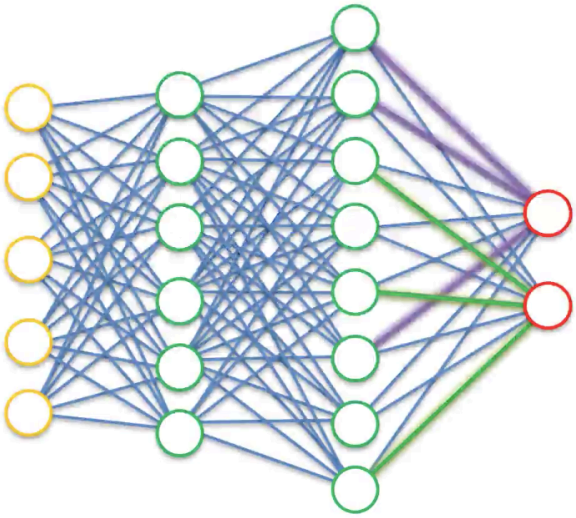
Softmax & Cross-Entropy

Softmax & Cross-Entropy

Flattening

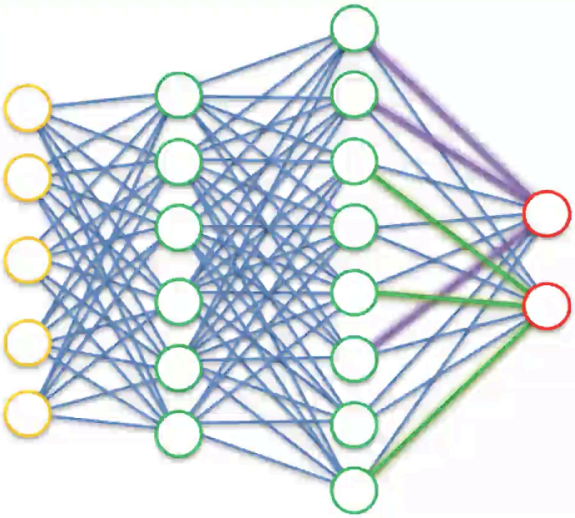


Dog 0.95

Cat 0.05

Softmax & Cross-Entropy

Flattening

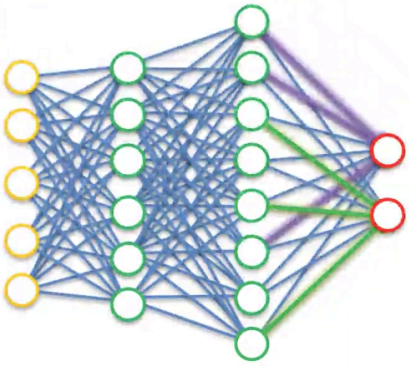


Dog 0.95

Cat 0.05

Softmax & Cross-Entropy

Flattening



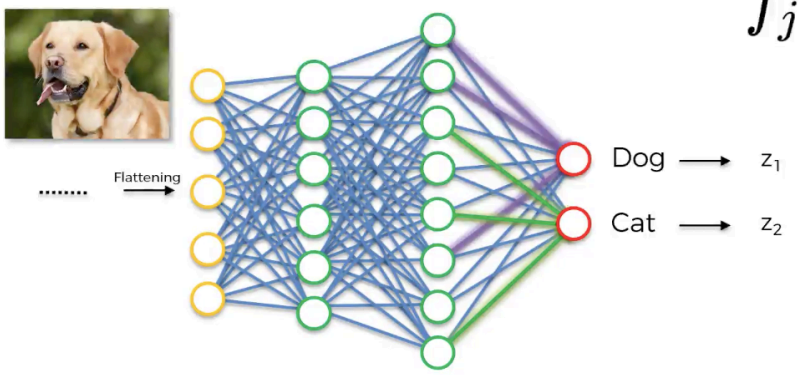
Dog

Cat

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

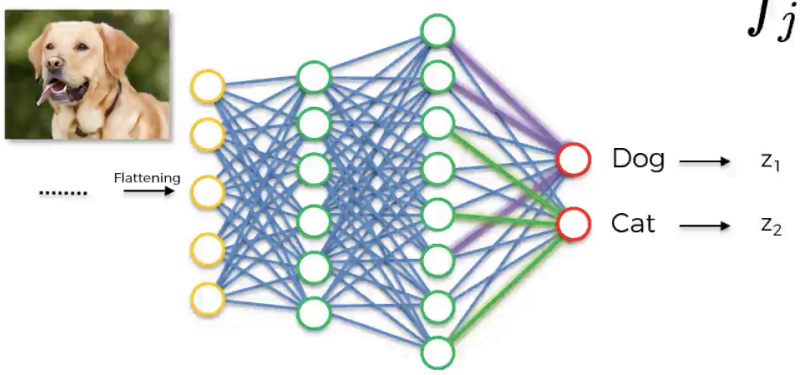
Softmax & Cross-Entropy

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



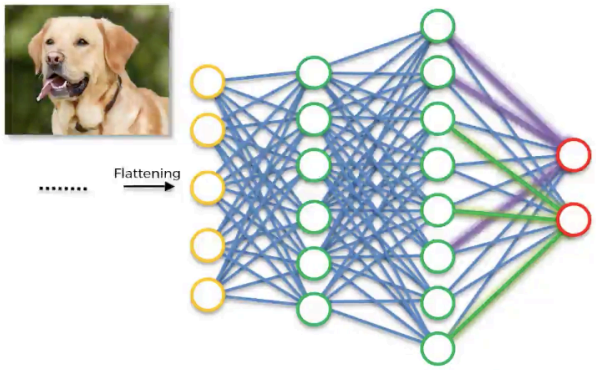
Softmax & Cross-Entropy

$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$



Deep Learning A-Z
© SuperDataScience

Softmax & Cross-Entropy



$$f_j(z) = \frac{e^{z_j}}{\sum_k e^{z_k}}$$

In here we can see that the z_1 and z_2 are two random number but after applying softmax function the sum of our two numbers become 1.

Softmax & Cross-Entropy

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

Cross entropy function

Softmax & Cross-Entropy

$$L_i = -\log\left(\frac{e^{f_{y_i}}}{\sum_j e^{f_j}}\right)$$

$$H(p,q) = -\sum_x p(x)\,\log q(x)$$

After applying the softmax, one good option for our cost function is cross-entropy. But remember that in here it doesn't call cost, it called loss function. Cost and loss have some different terminology but in here for our propose we suppose it's the same.
So, our purpose is to minimize the loss.

Softmax & Cross-Entropy

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

1
0

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

$$H(p,q) = -\sum_x p(x)\ \log q(x)$$

1
0

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

$$H(p,q) = -\sum_x p(x)\ \log q(x)$$

1
0

Softmax & Cross-Entropy

Dog 0.9

Cat 0.1

$$H(p,q) = -\sum_{x} p(x) \log q(x)$$

1
0

Softmax & Cross-Entropy

NN1

Softmax & Cross-Entropy

NN1 NN2

Softmax & Cross-Entropy

NN1 NN2

Dog	1
Cat	0

Softmax & Cross-Entropy

NN1 NN2

Dog	1
Cat	0

Dog	0
Cat	1

Softmax & Cross-Entropy

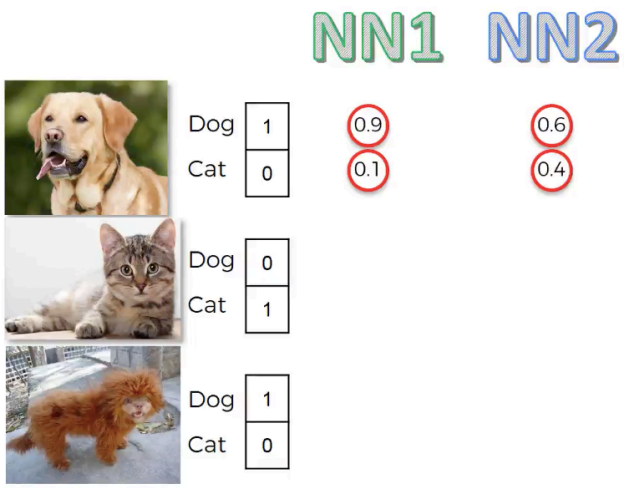
NN1 NN2

Dog	1
Cat	0

Dog	0
Cat	1

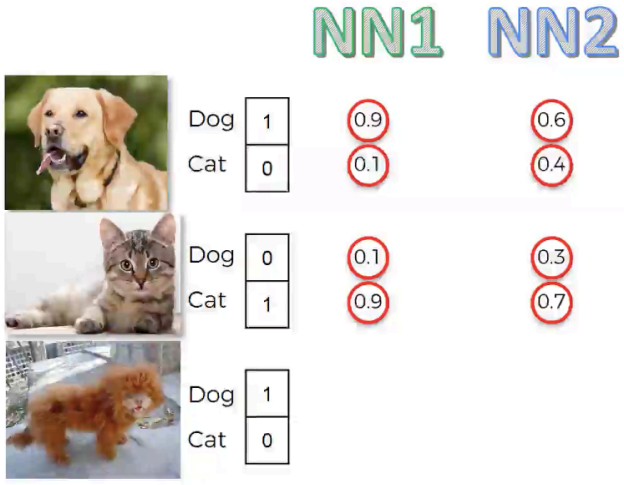
Dog	1
Cat	0

Softmax & Cross-Entropy



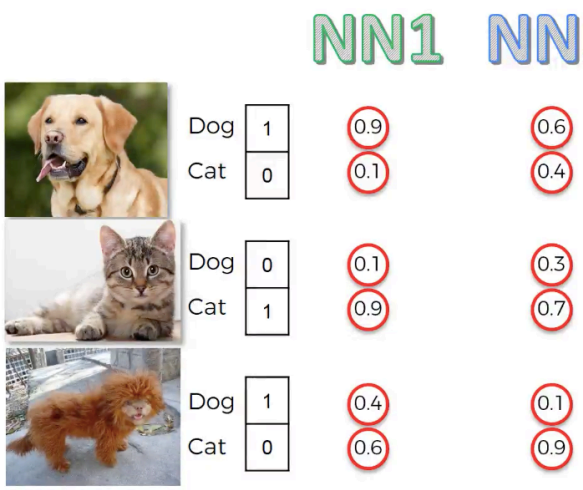
First neuron network I correct. Second neuron network is correct but worse.

Softmax & Cross-Entropy



Second neuron network I correct. Second neuron network is correct but worse.

Softmax & Cross-Entropy



First neuron network I incorrect. Second neuron network is incorrect and worse.
In here if you notice, NN1 outperformed NN2.

Softmax & Cross-Entropy

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

Softmax & Cross-Entropy

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Softmax & Cross-Entropy

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

1/3 = 0.33

1/3 = 0.33

Deep Learning A-Z © SuperDataScience

In classification error, we care whether we get it right or not and we don't care about the probabilities. This is 1 out of 3 wrong, 0.33 is the error rate. this is not true for our case.

Softmax & Cross-Entropy

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

1/3 = 0.33 | 1/3 = 0.33

Mean Squared Error

0.25 | 0.71

We saw the two other error formula before.

Softmax & Cross-Entropy

NN1

Row	Dog^	Cat^	Dog	Cat
#1	0.9	0.1	1	0
#2	0.1	0.9	0	1
#3	0.4	0.6	1	0

NN2

Row	Dog^	Cat^	Dog	Cat
#1	0.6	0.4	1	0
#2	0.3	0.7	0	1
#3	0.1	0.9	1	0

Classification Error

1/3 = 0.33 | 1/3 = 0.33

Mean Squared Error

0.25 | 0.71

Cross-Entropy

0.38 | 1.06

Deep Learning A-Z © SuperDataScience

Advantage of cross-entropy over mean squared error:

1. Cross-entropy asses even a small error at the very start; at start of back propagation, the output value is going to be much smaller value than what we want. So, at the very start our gradient decent would be very low and won't be enough for NN to start adjusting and moving to the right direction.

Remember cross-entropy works well in classification and so CNN, for example if in our ANN we had regression then we would go with mean squared error.

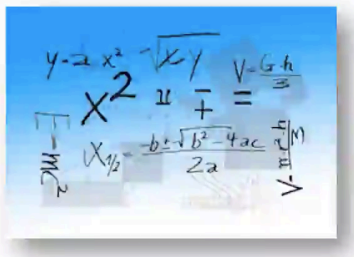
For learning more search YouTube: The softmax output function by Jeffrey Hinton

Softmax & Cross-Entropy

Additional Reading:

A Friendly Introduction to Cross-Entropy Loss

By Rob DiPietro (2016)



Link:

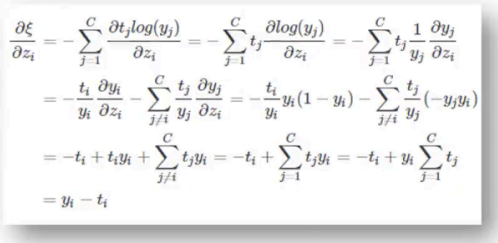
https://rdipietro.github.io/friendly-intro-to-cross-entropy-loss/

Softmax & Cross-Entropy

Additional Reading:

How to implement a neural network Intermezzo 2

By Peter Roelants (2016)



$$\frac{\partial \xi}{\partial z_i} = -\sum_{j=1}^{C} \frac{\partial t_j log(y_j)}{\partial z_i} = -\sum_{j=1}^{C} t_j \frac{\partial log(y_j)}{\partial z_i} = -\sum_{j=1}^{C} t_j \frac{1}{y_j} \frac{\partial y_j}{\partial z_i}$$
$$= -\frac{t_i}{y_i} \frac{\partial y_i}{\partial z_i} - \sum_{j \neq i}^{C} \frac{t_j}{y_j} \frac{\partial y_j}{\partial z_i} = -\frac{t_i}{y_i} y_i (1 - y_i) - \sum_{j \neq i}^{C} \frac{t_j}{y_j} (-y_j y_i)$$
$$= -t_i + t_i y_i + \sum_{j \neq i}^{C} t_j y_i = -t_i + \sum_{j=1}^{C} t_j y_i = -t_i + y_i \sum_{j=1}^{C} t_j$$
$$= y_i - t_i$$

Link:

http://peterroelants.github.io/posts/neural_network_implementation_intermezzo02/