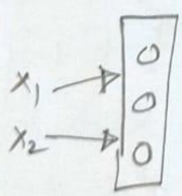
multiclass classification & multi label classification & PyTorch

1. One Vs all (eg we have 3 classes)

   - Need to prepare 3 fake datasets
   - Need to train 3 classifier
   - At Test time
     - Need to test the 3 classifier & pick the one that have highest prob. score.

2. Soft max

   - No need to do so.
   - Consider unit at output = # of classes



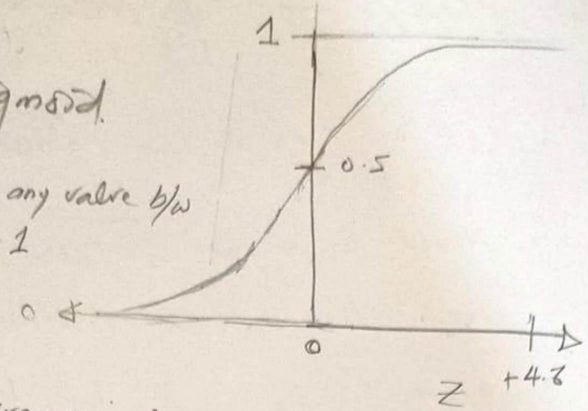
if # class = 3.

   - Use soft max as an activation function to produce probabilities of each class using the following function.

$$A = \frac{e^{z}}{\sum_{i=1}^{n} e^{z}}$$

② Multiclassification and Multi label classif.

① sigmoid



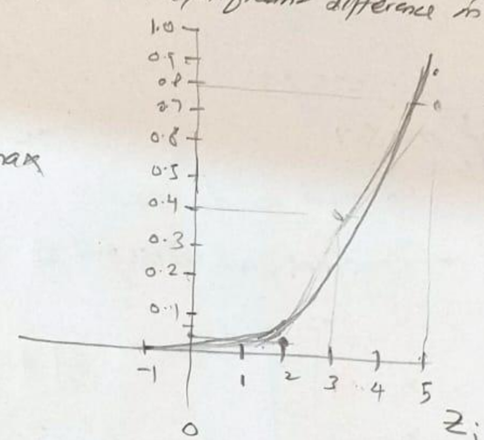
$$a = \frac{1}{1+e^{-z}}$$
or
$$a = \frac{e^z}{1+e^z}$$

$Z(+4.6) \simeq 1$
$Z(-4.6) \simeq 0$

- map any value b/w 0 & 1

- increase in the input value (Z) the sigmoid score till 1
- The high value will have the high probability but not the higher. (not a significant difference in values)

② softmax



$$a = \frac{e^z}{\sum_{i=1}^{n} e^{z_i}}$$

$$Z = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix}, \quad a = \begin{bmatrix} e^5/(e^5+e^2+e^{-1}+e^3) \\ e^2/(e^5+e^2+e^{-1}+e^3) \\ e^{-1}/(e^5+e^2+e^{-1}+e^3) \\ e^3/(e^5+e^2+e^{-1}+e^3) \end{bmatrix} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

- high input will have high probability.
- output prob. dependent on the Z_i value of each output unit.

Recognizing cats, dogs, and baby chicks , other

1 2 3 0

3 1 2 0 3 2 0 1

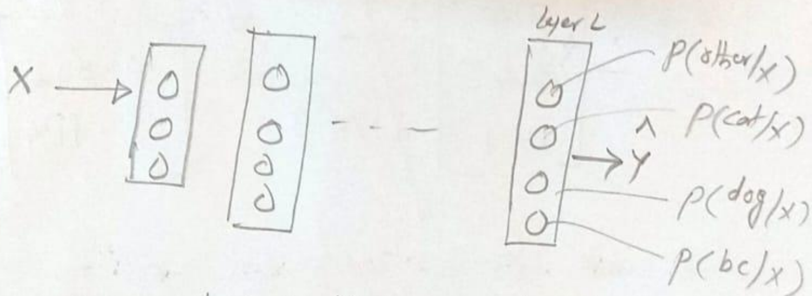
③

- Generalization of logistic regression is called softmax
- lets we want to recognize cats, dogs, baby chicks, we have the images of cats, dogs, baby chicks, and other animals. we label them:

baby chick	cat	dog	other animal	baby chick	dog	other animal
3	1	2	0	3	2	0

#of classes = 4 (0, 1, 2, 3)

0: none of them / other



- We want to build NN that tells us what is prob of each class. & sum of prob is equal to 1.
- For this purpose we use softmax.

$$\underset{(4,1)}{z^{[L]}} = W^{[L]} a^{[L-1]} + b^{[L]}$$

Activation Function: softmax

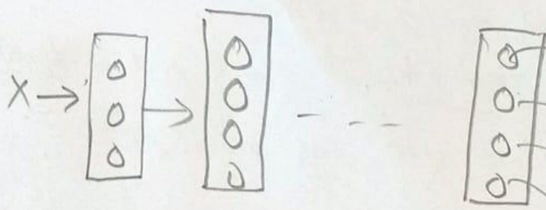
$$a^{[L]} = \frac{e^{z^{[L]}}}{e^{z_1^{[L]}} + e^{z_2^{[L]}} + e^{z_3^{[L]}} + e^{z_4^{[L]}}} = \frac{e^{z^{[L]}}}{\sum_{i=1}^{n^{[L]}} e^{z_i^{[L]}}}$$

- Example

$$Z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix}$$

$$a^{[L]} = \begin{bmatrix} e^{5} \\ e^{2} \\ e^{-1} \\ e^{3} \end{bmatrix} \Big/ \left(e^{5} + e^{2} + e^{-1} + e^{3}\right)$$

$$\underset{(4,1)}{a^{[L]}} = \begin{bmatrix} 148.4 \\ 7.4 \\ 0.4 \\ 20.1 \end{bmatrix} \Big/ 176.3 = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$



$e^{5}/176.3 = 0.842$

$e^{2}/176.3 = 0.042$

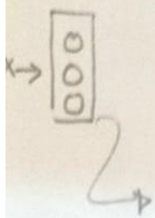
$e^{-1}/176.3 = 0.002$

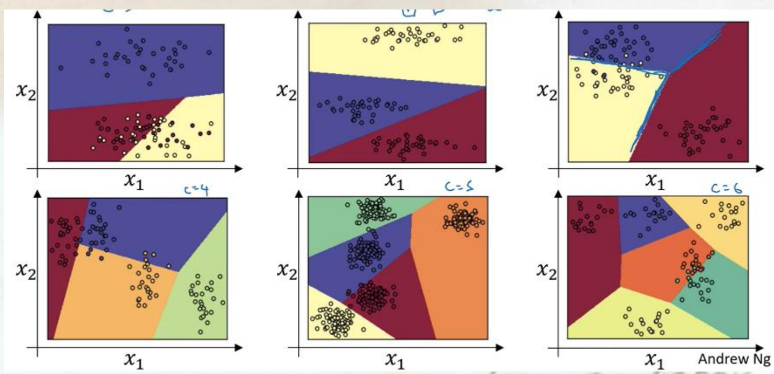
$e^{3}/176.3 = 0.114$

$+$ 1

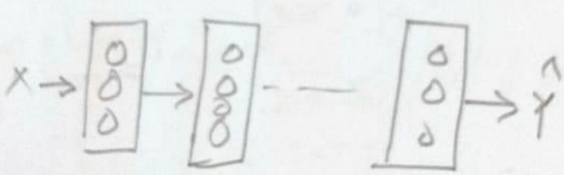
- sigmoid & ReLU activation take a single value ($z_1^{[l]}$) and output a single value ($a_1^{[l]}$), whereas softmax takes a vector as input ($\underset{(4,1)}{z^{[l]}}$) and output a vector ($\underset{(4,1)}{a^{[l]}}$)

- softmax learns decision boundaries b/w classes
  → if we consider a shallow NN (logistic regression) for # classes = C = 3, we get linear decision boundaries.





→ with NN we can learn non linear decision boundaries



Understanding of softmax

As $C = 4$

$$Z^{[L]} = \begin{bmatrix} 5 \\ 2 \\ -1 \\ 3 \end{bmatrix} \xrightarrow{\text{map}} a^{[L]} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

(hardmax)

$$a^{[L]} = \begin{bmatrix} 0.842 \\ 0.042 \\ 0.002 \\ 0.114 \end{bmatrix}$$

(softmax)
a gentle mapping

- Softmax regression generalizes logistic regression to C classes rather than 2 classes
- If $C = 2$, softmax reduces to logistic regression

$$a^{[L]} = \begin{bmatrix} 0.842 \\ \cancel{0.158} \end{bmatrix}$$

no need to consider it for logistic regression

$$\Sigma = 1$$

(7)

Training soft max classifier

⇒ Need to map no of classes with labels using one hot encoding

$$\begin{bmatrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{bmatrix}$$

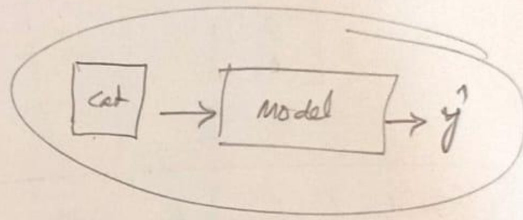
if $C=4$ & our image is of cat

(→ It make ease to use cross entropy function to compute cost)

one hot encoding

Then.

$$\underset{(4,1)}{y} = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$$ cat

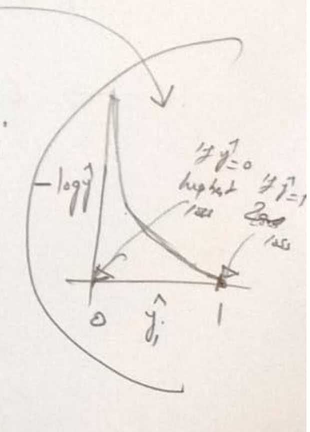


Let $\hat{y} = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}$

NN is not doing well on this example as Given input image is a cat but NN says 0.2 (20%) chance the given input image is a cat.

$\boxed{\text{Loss Function}}$ ⇒ cross entropy

$$\mathcal{L}(\hat{y}, y) = -\sum_{i=1}^{4} y_i \log \hat{y}_i$$



As $y = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$ & $\hat{y}_i = \begin{bmatrix} 0.3 \\ 0.2 \\ 0.1 \\ 0.4 \end{bmatrix}$

→ $y_1 = y_2 = y_3 = 0$

∴ $\mathcal{L}(\hat{y}, y) = -\log(0.2) = 0.698$

- For m training example (a mini-batch):

$$J(w^{[l]}, b^{[l]} \dots) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

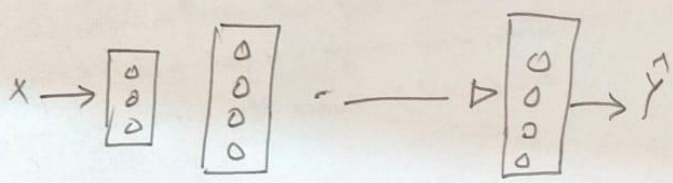
- Initially

$$Y^{\{t\}} = [y^{(1)}, y^{(2)} \dots y^{(m)}], \quad \hat{Y} = [\hat{y}^{(1)}, \hat{y}^{(2)} \dots \hat{y}^{(m)}]$$

- But now for multi-class classification

$$Y_{(4,1)} = \begin{bmatrix} 0 & 0 & \\ 1 & 0 & \dots \\ 0 & 1 & \\ 0 & 0 & \end{bmatrix} \qquad \hat{Y} = \begin{bmatrix} 0.3 & \\ 0.2 & \dots \\ 0.1 & \\ 0.4 & \end{bmatrix}$$

- $\boxed{\text{Gradient descent with softmax}}$



→ Forward pass: Compute output prob. & cost

→ Backward pass: compute gradients as we did using logistic regression

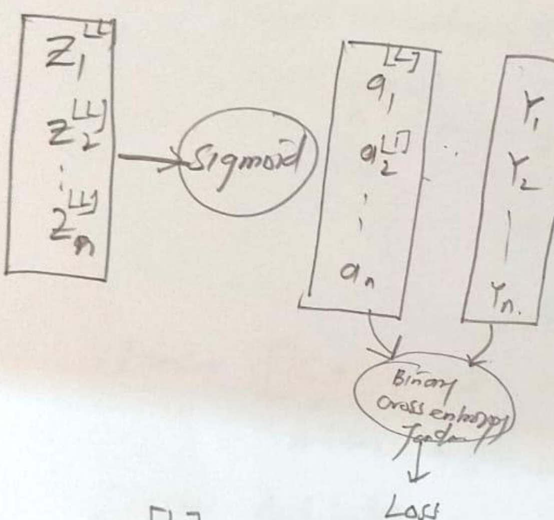
$$L = -y \log a^{[L]}$$

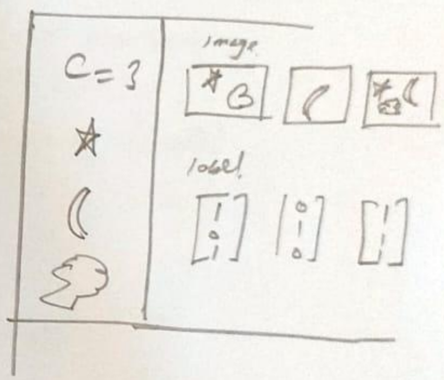
$$da^{[L]} = \frac{\partial L}{\partial a^{[L]}} = -y / a^{[L]}$$

$$dz^{[L]} = \frac{\partial L}{\partial a^{[L]}} \times \frac{\partial a^{[L]}}{\partial z^{[L]}} = da^{[L]} \cdot \frac{\partial a^{[L]}}{\partial z}$$

- We can compute gradients ourself...
- But now onwards we will use programming framework (PyTorch), it will compute Forward pass & backward pass calculations for us

* Multi_label classification (mask-RCNN)



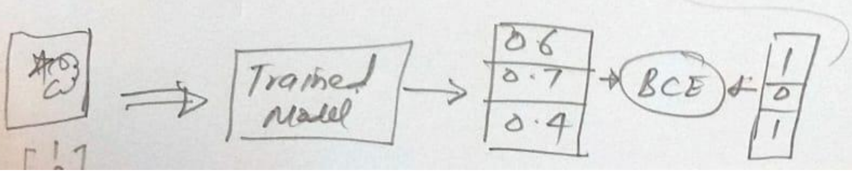


- $$a_i^{[L]} = \frac{1}{1+e^{-z_i^{[L]}}}$$

Not $\sum a_i^{[L]} \neq 1$

- Binary Cross entropy Loss $= - \sum_{i=1}^{n} \left[Y_i \log a_i^{[L]} + (1-Y_i) \log (1-a_i^{[L]}) \right]$

$\rightarrow =$



10

$$loss = -\sum_{i=1}^{3} (y_i a_i^{[L]} + (1-y_i) \log(1-a_i^{[L]})]$$

$$loss = -[1 \log(0.6) + \log(1-0.7) + \log(0.4)]$$

$$loss = 2.63$$

(11)

Python API to Interact with Pytorch Framework

Torch

nn

optim

nn.Functional (ReLU)

util.data

torchvision

transforms (transform numpy to tensor)

datasets (help to get torchvision dataset)

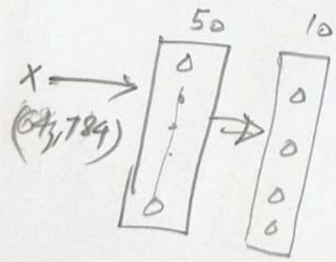
- Import DataLoader (Extract data from source & creat mini batch
- Import Dataset

(help to get data from our dataset) need to imp __len__() & __getitem__()

- Tensor [64 1 28 28]

64 = batchsize, 1 = no of chanels, 28 = width, 28 = height

- NW Architecture



Pytorch Neural Network example

https://www.youtube.com/watch?v=Jy4wM2X21u0

https://aladdinpersson.medium.com/pytorch-neural-network-tutorial-7e871d6be7c4

PyTorch Tutorials

https://github.com/aladdinpersson/Machine-Learning-Collection

https://www.youtube.com/playlist?list=PLhhyoLH6IjfxeoooqP9rhU3HJIAVAJ3Vz

Conversion from TensorFlow to Pytorch:

https://neptune.ai/blog/moving-from-tensorflow-to-pytorch

Tensor flow intro by Andrew Ng:

https://www.youtube.com/watch?v=S9ElPZupUsE&list=PLpFsSf5Dm-pd5d3rjNtIXUHT-v7bdaEIe&index=77

Deeplizard:

https://www.youtube.com/watch?v=v5cngxo4mIg&list=PLZbbT5o_s2xrfNyHZsM6ufI0iZENK9xgG

Assignment #6 (Building your neural network in Pytorch for SIGN dataset). Update the following Tensorflow program into a Pytorch.

https://github.com/Kulbear/deep-learning-coursera/blob/master/Improving%20Deep%20Neural%20Networks%20Hyperparameter%20tuning%2C%20Regularization%20and%20Optimization/Tensorflow%20Tutorial.ipynb

a. Exploring the Tensorflow Library => do it with pytorch (PRACTICE)
b. Building your first neural network in Tensorflow = > do it with pytorch(SUBMISSION)