# CS230 Hands-on session 3: “Backpropagation”

This section will teach you the backpropagation equations in a neural network.

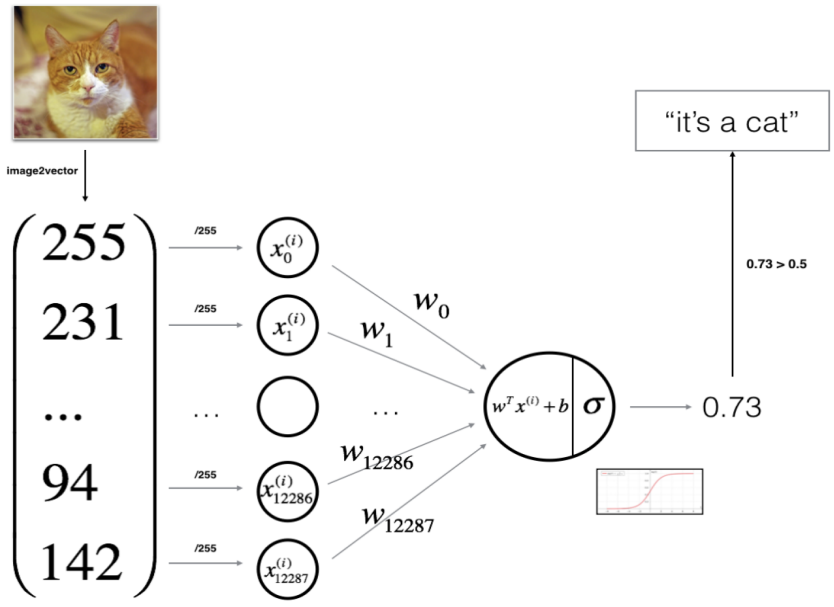
We will use mentimeter.
Please connect on www.menti.com and enter the access code provided by the TA.

There are different approaches to obtain the optimal set of weights/parameters for a neural network. The naive approach is randomly generating new weights at every iteration. A better approach uses local information of the loss function (e.g its gradient) to better choose the weights at every iteration.

Warm-up question: Does backpropagation compute a *numerical* or *analytical* value of the gradients in a neural network? (Answer on Menti)

## Part I – Logistic regression backpropagation with a single training example

In this part, you are using Stochastic Gradient Optimization to train your Logistic Regression. Consequently, the gradients leading to the parameter updates are computed on a single training example.




a) Write down the forward propagation equations leading to $L$.
b) Analyze the dimensions of all the variables in your forward propagation equations.
c) Write down the backpropagation equations to compute $\frac{\partial L}{\partial w}$ .

## Part II - Logistic regression backpropagation with a batch of *m* training examples

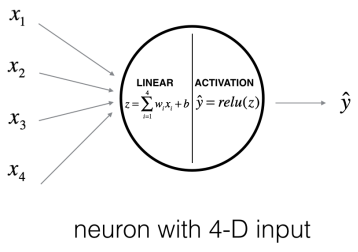
In this part, you are using Batch Gradient Optimization to train your Logistic Regression. Consequently, the gradients leading to the parameter updates are computed on the entire batch of $m$ training examples.

a) Write down the forward propagation equations leading to $J$.
b) Analyze the dimensions of all the variables in your forward propagation equations.
c) Write down the backpropagation equations to compute $\frac{\partial J}{\partial w}$ .

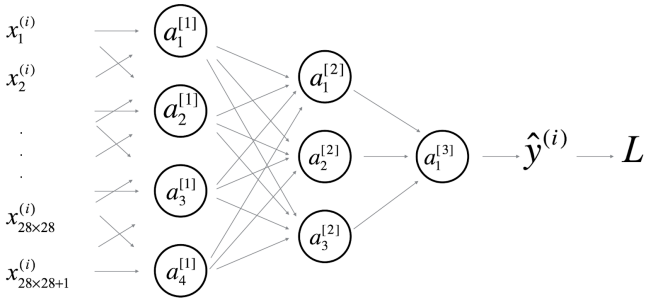
## Part III - Revisiting Backpropagation

In this part, you will see examples of backpropagation in (1) a shallow neural network, (2) a custom network computing a univariate scalar function, (3) a custom network computing a multivariate scalar function and finally (4) a custom network with matrix inputs. These questions are toy problems to teach you how to compute gradients in a broader context. As far as we know, the example functions in (2), (3) and (4) don't have applications.

1. You are given the shallow neural network on the right and your goal is to compute $\frac{\partial L}{\partial a_1^{[1]}}$ . The right figure recalls the structure of a neuron.



neuron with 4-D input




a) What other derivatives do you need to compute before finding $\frac{\partial L}{\partial a_1^{[1]}}$ ?
b) What values do you need to cache during the forward propagation in order to compute $\frac{\partial L}{\partial a_1^{[2]}}$ ?

2. **Backpropagation example on a univariate scalar function (e.g. $f: R \rightarrow R$):**

Let's suppose that you have built a model that uses the following loss function:

$$L = (\hat{y} - y)^2 \quad \text{where} \quad \hat{y} = tanh\left[\sigma\left(wx^2 + b\right)\right]$$

Assume that all the above variables are scalars. Using backpropagation, calculate $\frac{\partial L}{\partial w}$.

3. **Backpropagation example on a multivariate scalar function (e.g. $f : R^n \rightarrow R$):**

Let's suppose that you have built a model that uses the following loss function:

$$L = -y \log(\hat{y}) \quad \text{where} \quad \hat{y} = ReLU\left(w^T x + b\right)$$

a) Assume $x \in R^n$. What's the shape of $w$?
b) Using backpropagation, obtain $\frac{\partial L}{\partial w}$.

4. **Backpropagation applied to scalar-matrix functions ($f : R^{n \times m} \rightarrow R$):**

The final case that is worth exploring is:

$$L = \| \hat{y} - y \|_2^2 \quad \text{where} \quad \hat{y} = \sigma(x) \cdot W$$

a) Assume that $\hat{y} \in R^{1 \times m}$ and $x \in R^{1 \times n}$. What is the shape of $W$?
b) Using backpropagation, calculate $\frac{\partial L}{\partial x}$.