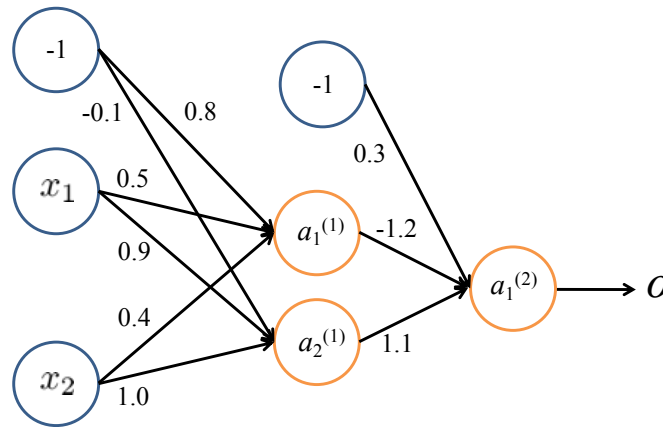


## Homework2, Due Date: Mon, Mar 18

Instructor: Dr. Mohammad Pourhomayoun

### Problem1: Backpropagation

In the following Neural Network, we have initialized the weights randomly. Use a training sample  $(X,y) = ((1,1), 0)$  to update the weights (perform one round of backpropagation using one training sample). Use learning rate parameter  $\alpha = 0.1$ . Note that we have bias terms with value of -1 in this network (no need for coding for this question).



### Problem2: Adjusting the hyperparameters of MNIST Digit Recognition using ANN model in Keras+TensorFlow and Grid-Search in SciKitLearn:

In this problem, we want to apply an interesting method to use the grid search capability of sklearn library to adjust the hyperparameters of Keras+Tensorflow ANN models!

To do this, we use **KerasClassifier** class as a wrapper for Keras models and then use it in sklearn.

**Note: Please be aware that it may take a long time (hours) to finish running this code. Please start working on it early!**

- a- Download the Kears+Tensorflow tutorial from CSNS. Import all required modules including the following:

```
from keras.wrappers.scikit_learn import KerasClassifier  
from sklearn.model_selection import GridSearchCV
```

- b- Import the minst dataset, and split it into testing and training as we saw in the tutorial. Then, reshape each sample into a row vector, and scale it by dividing by 255.
- c- Perform OneHotEncoding for the label **y**. So, your label will be a vector of 10 elements for each data sample.

- d- Now, define a function called ***model\_creator***. This function will define, create, and compile your neural network model according to your structure, and then return the built model as the output. For the ANN neurons/layers, use the same structure as we had in the tutorial:

```
def model_creator():  
    # define:  
    model = Sequential()  
    # design the structure:  
    model.add(...)  
    # compile:  
    model.compile(loss='categorical_crossentropy', optimizer='adam',  
                  metrics=['accuracy'])  
    # return:  
    return model
```

- e- Fix the random state for reproducibility:  
***seed = 2, np.random.seed(seed)***
- f- Use ***KerasClassifier*** class to wrap your model as an object:  
***model = KerasClassifier(build\_fn = model\_creator, verbose=2)***
- g- Now, run sklearn GridSearch to find the best batch\_size and epochs. Search in this range: batch\_size = [30 , 50 , 100 ] , epochs = [10 , 15 , 20].  
In your GridSearch, the estimator is the above ***model***, the scoring should be 'neg\_log\_loss', and you have to use 10-fold CV.
- h- Based on your results, what is the best batch\_size and epochs?  
Now, test your model with the best batch\_size and epochs on the testing set.  
***grid.best\_estimator\_.model*** gives you the best model found and trained in the grid-search. What is the prediction accuracy on the testing set?

**WARNING: It may take a long time to finish the process (depending on your system it can be up to hours). So, don't leave it for the last minutes (late submissions will not be accepted!)**