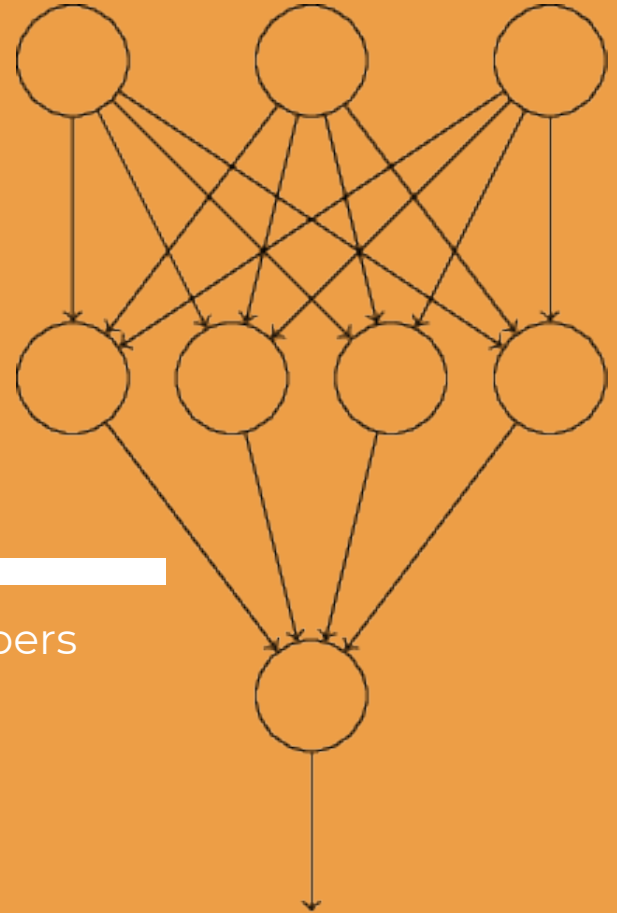


NEURAL NETWORKS

a **machine learning** classifier for handwritten numbers
by: sravya balasa



WHAT IS A **NEURAL NETWORK**?

- Take in a large amount of **training** examples
 - Greater # of training examples = Greater accuracy
- Develop a system that can learn from those examples
- Then evaluates **testing** examples

Neurons are in **layers** that **translate** information

■ MINIMUM VIABLE PRODUCT

1. To understand and implement a neural network (NN)
2. Using NN to classify the MNIST dataset of handwritten numbers (0-9)
3. Display the accuracy of the neural networks' classification
4. Implementation of successful **user input** for demo purposes
 - a. Modifications to testing dataset

 = **504192**

■ IMPLEMENTATION

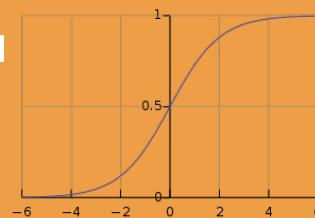
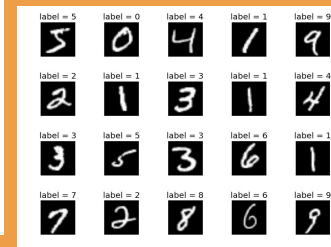
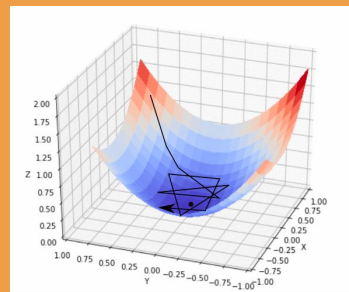
LIBRARIES

- random
- numpy
- pickle
- gzip
- sciPy → misc
- PIL → image
- json
- sys

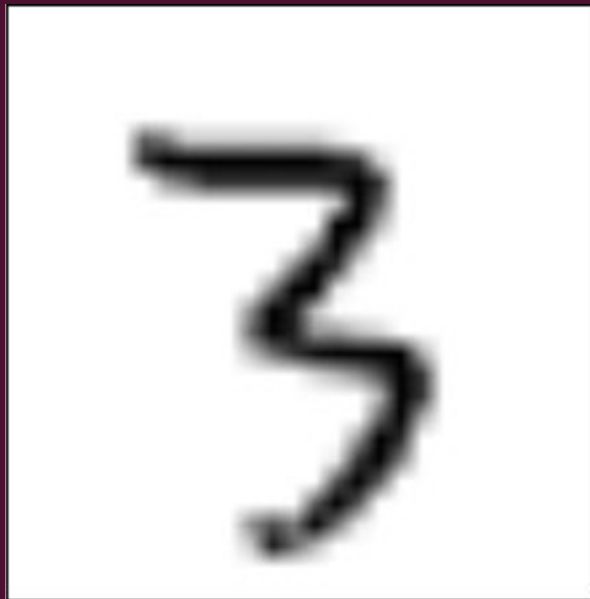
CHALLENGES

- Understanding the python functions and math used in the algorithm
- Finding the individual output from each image that contributes to the accuracy rate
- Discovering the exact pattern of the data structures of the input
- Implementing user input facility
- Implementing new libraries

RESULTS & METHODS



■ WHAT'S THE USER'S INPUT?



1. transforms an image to 28*28 image
2. transforms it to a numpy (matrix) array
3. adds it to the **testing data**
4. evaluates!

WHAT'S THE USER'S INPUT?

USER INPUT:
asks for which image
should be processed

```
>>>  
RESTART: /home/linux/ieng6/spis18/spis18ab/github/Spis-Final-Project/SPIS/expand_mnist.py  
Input number between 0-9 you would like to test: 3  
Expanding the MNIST testing set  
Expanding image number 1000  
Expanding image number 2000  
Expanding image number 3000  
Expanding image number 4000  
Expanding image number 5000  
Expanding image number 6000  
Expanding image number 7000  
Expanding image number 8000  
Expanding image number 9000  
Expanding image number 10000  
Saving expanded data. This may take a few minutes.  
>>>
```

Ln: 209 Col:

WHAT'S THE OUTPUT?

EPOCH:
one full training
of network

```
RESTART : /home/linux/ieng6/spis18/spis18ab/github/Spis-Final-Project/SPIS/runner.py
Epoch 0 training complete
Real Ouput: 3
Desired Output: 3
Real Ouput: 3
Desired Output: 3
Accuracy on evaluation data: 39828 / 50001
Epoch 1 training complete
Real Ouput: 3
Desired Output: 3
Real Ouput: 3
Desired Output: 3
Accuracy on evaluation data: 41591 / 50001
Epoch 2 training complete
Real Ouput: 3
Desired Output: 3
Real Ouput: 3
Desired Output: 3
Accuracy on evaluation data: 42518 / 50001
Epoch 3 training complete
Real Ouput: 3
Desired Output: 3
Real Ouput: 3
Desired Output: 3
Accuracy on evaluation data: 42828 / 50001
Epoch 4 training complete
Real Ouput: 3
Desired Output: 3
Real Ouput: 3
Desired Output: 3
Accuracy on evaluation data: 43610 / 50001
>>>
```

REAL:
result of user's input after
run through network
DESIRED:
user's predicted result

ACCURACY:
-number of testing data
inputs classified correctly
-accuracy increases with each
epoch

ACTIVATION ENERGIES

- Activation energies only adjusted by **weights** and **biases**
 - Allows for slow adjustment
- Last layer = desired output layer
- **Ex:** 3rd neuron has activation 1.0, inputted number is a 2

highest
ACTIVATION →



BACKPROPAGATION

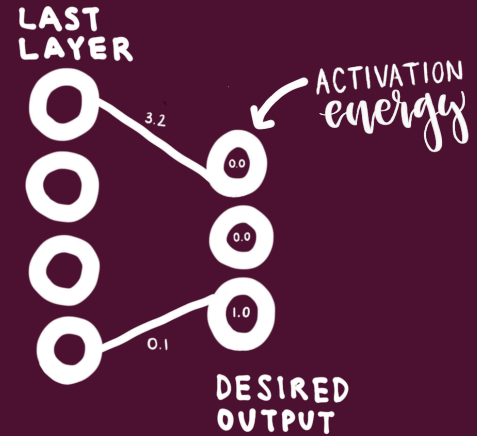
THE ALGORITHM ITSELF.

- How do changes in weights, biases, activation affect final activation?
- **Each layer depends on previous...**
 - Adding desired effects to get to the output layer
- Ex: $a_{\text{desired}} = a_{\text{desired-1}} * w$

POSSIBLE CHANGES: WEIGHTS, BIASES, INITIAL ACTIVATION

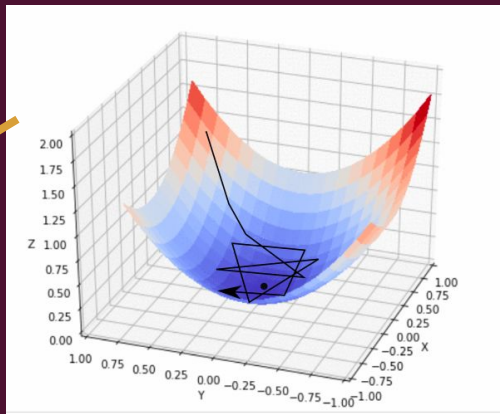
COST FUNCTION

- Measures **changes in activation** from one layer to the next
 - COST of this DIFFERENCE = GRADIENT
- Ex: Between last layer and desired output layer
- Goal:
 - Minimize the cost through **GRADIENT DESCENT**



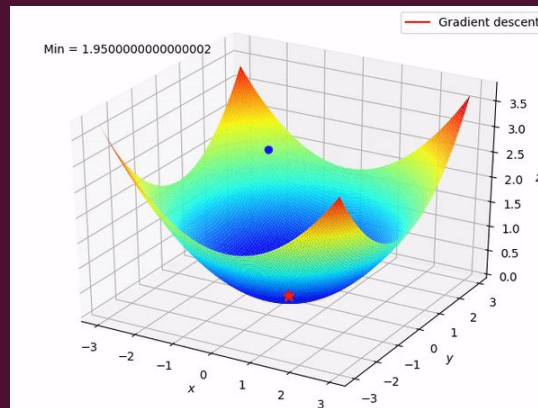
STOCHASTIC GRADIENT DESCENT

SGD



GRADIENT
=
Effect on cost

- Division into mini-batches
- **Effect on cost** computed each batch
- Total cost adjusted for each batch
- **Less accurate + Fast** → Local min

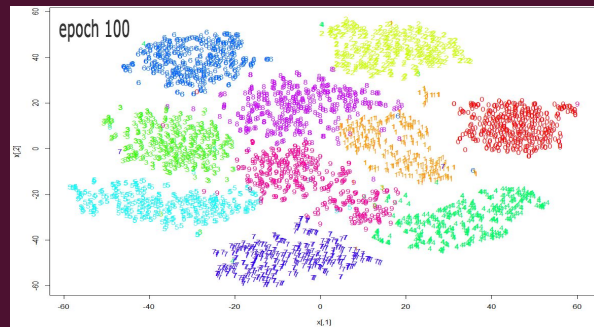


NORMAL

- **More accurate + slow** → Local min
- Careful because takes in ALL training data in one batch

■ FURTHER DEVELOPMENT

1. Segmentation of numbers and analyzing letters
2. Save and load a trained network
3. Creating a **tSNE** display



neural
network
