Machine Learning and Object Recognition

Simple Neural Network Introduction
Road map:

1) Simple artificial neural network example
2) Learning/training
3) Apply to image recognition
4) Drop mic
Let’s consider a simple program.

Its job is to **predict test scores** given:

- **hours slept** the night before
- **hours spent studying**

It will use previous data to make this prediction (say we recorded sleep and study times and grades for the last three tests)
Artificial Neural Network (ANN)

- Based on how neurons in our brains work
Back to the example:

In this example, we have two inputs: hours of sleep and hours of study.

This data gets pushed through a bunch of layers (if we have more than one layer, we call it “deep”).

Then we get our output, in this case an estimate of our grade.
What is a layer?

A layer is made up of neurons (circles) and synapses (lines).

A synapse multiplies its input by a weight and outputs the result.

A neuron adds the outputs from all its synapses and applies an activation function.

\[ f(x) = \frac{1}{1 + e^{-\beta x}} \]
Why the activation function?

Given our summed inputs, what do we output?

The simplest activation fn would be an on/off switch with a threshold.

Benefits of nonlinear functions:

- Model neurons better
- Allow us to solve nontrivial problems.

NOTE: There are many different functions we could choose from.
Hyperparameters are constants that we define about our network:

- Input layer size (ex: 2 -> sleep hours and study hours)
- Output layer size (ex: 1 -> grade)
- Hidden layer size (ex: 3)
- Learning rate: how big are your steps during learning? (more on this later)

Our network isn’t smart enough to decide it needs another layer.

**Learning** essentially means **optimizing the weights of our synapses**.
Forward Propagation

This is the process of pushing our inputs through the network.

Made really easy with matrix math.

Next step is to apply our activation function to each element in this big matrix.
Cool.
Training and learning

Still using the example, we now have some predictions based on random weights we assigned to the synapses initially.

Q:

How bad are our predictions?

A:

sum of squared differences of errors
Cost function

Function of inputs and weights.

We can’t change the inputs so we really just fiddle with the weights of our synapses to find the best configuration.

We want to minimize this function because then we get our predictions closest to the outputs overall.

This is hard because of dimensionality.

9 weights with say 100 ‘ticks’ each gives $100^9$ possible configurations to check
Minimizing cost function: backpropagation

There are a few strategies for this:

- Gradient Descent
  - Figure out which way is ‘downhill’ for each weight and test in that direction using partial derivatives.
- Backpropagation
  - Figure out which weight is most responsible for error and change that more when performing gradient descent
  - We start at the end and move backwards
- AKA complicated math to find which way is downhill. No guarantees though.
So how do we train our network?

- Propagate forward
- Propagate backward
- Make small changes to weights based on gradient
- Repeat a bunch of times

In the end, we hope to have found really good weights to minimize our cost function.
HELLO LADIES

CAN I INTRODUCE YOU TO A PROBLEM OF OVERFITTING
Application to Computer Vision
Image Recognition and Neural Nets

Convolutional Neural Networks
Order of operations - happens organically

1) Gray your image: cuz color is unreliable
2) Convolve with an edge detecting filter eg gabor filters or laplacian filters
3) Extract features using convolution with more complicated filters

This is NOT programmed into the net, it just happens from training.
**FACIAL RECOGNITION**

Deep-learning neural networks use layers of increasingly complex rules to categorize complicated shapes such as faces.

Layer 1: The computer identifies pixels of light and dark.

Layer 2: The computer learns to identify edges and simple shapes.

Layer 3: The computer learns to identify more complex shapes and objects.

Layer 4: The computer learns which shapes and objects can be used to define a human face.
Figure 3. The proposed pipeline of attribute inference.
Sources:

https://www.youtube.com/watch?v=n6hpQwq7hmy

http://white.stanford.edu/teach/index.php/An_Introduction_to_Convolutional_Neural_Networks

http://www.holehouse.org/mlclass/09_Neural_Networks_Learning.html


https://takinginitiative.wordpress.com/2008/04/03/basic-neural-network-tutorial-theory/

https://www.youtube.com/watch?v=LodC7Zm3X8Q

https://www.youtube.com/watch?v=S4ZUwgosj88

http://stevenmiller888.github.io/mind-how-to-build-a-neural-network/

http://iamtrask.github.io/2015/07/12/basic-python-network/

http://www.cs.toronto.edu/~hinton/absps/montrealTR.pdf

https://docs.google.com/viewer?a=v&pid=sites&srcid=ZGVPvYMV5dHrvbWFpbnxsc3Ydh4Vb3JpYVxjdNywMTRBZ3p6Njy9MnZkZTM1M0hlhZWNmZA