Neural Networks for Machine Learning

introduction
**Biological neural activity**

**Neurons** have body, axon and many dendrites

- In one of two states: firing and rest
- They fire if total incoming stimulus > threshold

**Synapse:** thin gap between axon of one neuron and dendrite of another

- Signal exchange
Artificial neural network

- Set of **nodes** with inputs and outputs
  Node performs computation via its **activation function**
- **Weighted connections** between nodes
- Connectivity gives network architecture
- NN computations depend on connections & weights

\[
\sum_{i=1}^{m} (w_i x_i) + bias
\]

\[
f(x) = \begin{cases} 
1 & \text{if } \sum w_i x_i + b \geq 0 \\
0 & \text{if } \sum w_i x_i + b < 0 
\end{cases}
\]
Common Activation Functions

Choice of activation function depends on problem and available computational power
Single Layer **Perceptron**

- Full 1958 NYT article above [here](#)
- Rosenblatt: it can *learn* to compute functions by learning weights on inputs from examples
- Not all functions 😞, cf. **Perceptrons**
A man adjusting the random wiring network between the light sensors and association unit of scientist Frank Rosenblatt's Perceptron, or MARK 1 computer, at the Cornell Aeronautical Laboratory, Buffalo, New York, circa 1960. The machine is designed to use a type of artificial neural network, known as a perceptron. (Photo by Frederic Lewis/Archive Photos/Getty Images)
MLP: Multilayer Perceptron

- $\geq 1$ “hidden layers” between inputs & output
- Can compute non-linear functions
- Training: adjust weights slightly to reduce error between output $y$ and target value $t$; repeat
- Introduced in 1980s, still used today
Backpropagation

Forward direction

Calculate network and error

Backpropagation: from output to input, recursively compute
\[ \frac{\partial E}{\partial w_{ij}} = \nabla_w E \]
and adjust weights.
Neural Network Architectures

Current focus on large networks with different “architectures” suited for different kinds of tasks

• Feedforward Neural Network
• CNN: Convolutional Neural Network
• RNN: Recurrent Neural Network
• LSTM: Long Short Term Memory
• GAN: Generative Adversarial Network
Feedforward Neural Network

• Connections allowed from a node in layer \( i \) only to nodes in layer \( i+1 \)
  i.e., no cycles or loops

• Simple, widely used architecture.

HTTP://PLAYGROUND.TENSORFLOW.ORG/
Tinker With a **Neural Network** Right Here in Your Browser.  
Don’t Worry, You Can’t Break It. We Promise.
CNN: Convolutional Neural Network

- Good for image processing: classification, object recognition, automobile lane tracking, etc.
- Classic demo: learn to recognize hand-written digits from MNIST data with 70K examples

![Diagram of a CNN model with layers and operations like convolution, pooling, and fully connected layers.](image)
RNN: Recurrent Neural Networks

- Good for learning over sequences of data, e.g., a sentence or words
- LSTM (Long Short Term Memory) a popular architecture

![Diagram of RNN](image)

Output so far: Machine

gif from Adam Geitgey
Deep Learning Frameworks

• Popular open source deep learning frameworks use Python at top-level; C++ in backend
  – **TensorFlow** (via Google)
  – **PyTorch** (via Facebook)
  – **MxNet** (Apache)
  – **Caffe** (Berkeley)

• **Keras**: popular API works with the first two and provides good support at architecture level
Good at Transfer Learning

• Neural networks effective for transfer learning
  Using parts of a model trained on a task as an initial model to train on a different task

• Particularly effective for image recognition
Good at Transfer Learning

• For images, the initial stages of a model learn high-level visual features (lines, edges) from pixels

• Final stages predict task-specific labels

source: http://ruder.io/transfer-learning/
Fine Tuning a NN Model

• Special kind of transfer learning
  – Start with a pre-trained model
  – Replace last output layer with a new one
  – Fix all but last layer by marking as trainable: false

• Retraining on new task and data very fast
  – Only the weights for the last layer are adjusted

• Example
  – Start: NN to classify animal pix with 100s of categories
  – Finetune on new task to classify pix of 15 common pets
Conclusions

• Quick introduction to neural networks and deep learning

• Learn more by
  – Take UMBC’s CMSC 478 machine learning class
  – Try scikit-learn’s neural network models
  – Explore Google’s Machine Learning Crash Course
  – Try Miner/Kasch tutorial on applied deep learning
  – Work through examples

• and then try your own project idea