How Do Neural Networks Work?

Dr. Tiberiu Stan – tiberiu.stan@northwestern.edu

Northwestern University

MatSci 395 – ML Lecture 3 – March 5, 2021
Motivation

• Neural networks are a powerful tool for classification and/or segmentation of large materials science imaging datasets

At the end of this lesson you should be able to:

• Understand some fundamental machine learning jargon (neuron, activation, weight, sigmoid, ReLU, etc.)
• Calculate neuron activations
• Explain how a neural network can classify images
Artificial Intelligence Overview

Artificial Intelligence (AI) - The science of making things smart
• Object and speech recognition
• Natural Language Processing (NLP)

Machine Learning (ML) - Achieving AI though systems that can “learn from experience” to find patterns in data
• Systems recognize patterns by example, rather than programming it with specific rules
• ML algorithms learn complex functions (or patterns) from the data and make predictions on it

Deep Learning - Layered-code structures that mimic the brain
• Convolutional Neural Networks (CNNs) – MatCNN
• Generative Adversarial Networks (GANs)
Neural Networks

Neural Networks in the Brain

Artificial Neural Networks
Classification

- Image “classification” is the process of labeling an image according to its visual content
- We will use a simple neural network to classify 4-pixel images into one of four “classes”
Neural Network

-1 0 +1

Normalization

Solid

Horizontal 100%

Vertical

Diagonal
First layer is always the “input layer” and has a number of “neurons” equal to the number of pixels in the image.
Input Layer

- First layer is always the “input layer” and has a number of “neurons” equal to the number of pixels in the image
- “Activation” ($a^n_i$) is the value held by the neuron, where $n$ is the layer number and $i$ is the neuron number
- “Receptive field” indicates what input gives the neuron the highest possible value
Neurons in the second layer are connected to those in the first layer using “weights”

\[ a^2_1 = w_{11}a^1_1 + w_{21}a^1_2 + w_{31}a^1_3 + w_{41}a^1_4 \]

- Weight values close to 0 have less impact on neuron activation
Fully connected layers” have neurons that are connected to every neuron in the previous layer:

\[ a^2_1 = w_{11}a^1_1 + w_{21}a^1_2 + w_{31}a^1_3 + w_{41}a^1_4 \]
\[ a^2_2 = w_{12}a^1_1 + w_{22}a^1_2 + w_{32}a^1_3 + w_{42}a^1_4 \]

\[ a^{n+1} = w^na^n \] where \( w \) and \( a \) are matrices.

Want activations to be between -1 and +1...
**Sigmoid Layer**

- "Sigmoid layer" uses a sigmoidal (squish) function to bound activations to -1 to +1.

- Sigmoid function is differentiable (important for network training) and non-linear (allows network to learn more complex patterns).

- Sigmoid layer: $a_{n+1} = \sigma(w^n a^n)$

\[
\sigma(x) = \frac{2}{1 + e^{-10x}} - 1
\]
• Weights are set through network training or by network design
• Missing links indicate zero weight
Exercise

- **Exercise**: what is activation of neuron $a^2_2$?
- $a^{n+1} = \sigma[w^n a^n]$
- $a^2 = \sigma[w^1 a^1]$

\[
\begin{bmatrix}
    a^2_1 \\
    a^2_2 \\
    a^2_3 \\
    a^2_4 \\
\end{bmatrix} = \sigma
\begin{bmatrix}
    1 & 1 & 0 & 0 \\
    0 & 0 & 1 & 1 \\
    -1 & 1 & 0 & 0 \\
    0 & 0 & -1 & 1 \\
\end{bmatrix}
\begin{bmatrix}
    a^1_1 \\
    a^1_2 \\
    a^1_3 \\
    a^1_4 \\
\end{bmatrix}
\]

- $a^2_2 = \sigma[w_{12} a^1_1 + w_{22} a^1_2 + w_{32} a^1_3 + w_{42} a^1_4]$
- $a^2_2 = \sigma[(0)(1) + (0)(1) + (1)(-1) + (1)(-1)]$
- $a^2_2 = \sigma[0 + 0 - 1 - 1]$
- $a^2_2 = \sigma[-2]$
- $a^2_2 \approx -1$
Exercise

• Other activations:
  • $a^2_1 \approx +1$
  • $a^2_2 \approx -1$
  • $a^2_3 = 0$
  • $a^2_4 = 0$

• Can get the same answers by inspecting receptive fields
Add a Second Sigmoid Layer

Input \( \mapsto \) Sigmoid \( \mapsto \) Sigmoid

-1 0 +1

-1 0 +1

-1 0 +1
Add a Second Sigmoid Layer

Input $\mapsto$ Sigmoid $\mapsto$ Sigmoid
ReLU

- “Rectified Linear Unit” or “ReLU” layers transform all negative inputs into 0

- Rectified linear function looks and acts like a linear function (good for network training), but it in fact not (allows network to learn complex patterns)
Softmax

• “Softmax” layers are usually the final layer and output a probability (or confidence level) of each class

\[ S(\alpha^n) = \frac{e^{\alpha_i}}{\sum_{j=1}^{k} e^{\alpha_j}} \]

• Softmax layers are useful for calculating probabilities in neural networks that do not have activations bound by -1 to 1
Normalization

- In this lecture example, we use a simpler Normalization function for the 5th layer

\[ N(a^n) = \frac{a_i}{\sum_{j=1}^{k} a_j} \]

Example:
- \( a^5_2 = N[w_{12}a^4_1 + w_{22}a^4_2 + w_{32}a^4_3 + w_{42}a^4_4 + \]
  \[ + w_{52}a^4_5 + w_{62}a^4_6 + w_{72}a^4_7 + w_{82}a^4_8] \]
- \( a^5_2 = N[1x0 + 1x0 + 1x0 + 1x1 + \]
  \[ + 1x0 + 1x0 + 1x0 + 1x0] \]
- \( a^5_2 = N[1] \)
- \( a^5_2 = \frac{1}{0 + 1 + 0 + 0} = 1 \quad \text{100%} \)
Neural Network

Input \( \implies \) Sigmoid \( \implies \) Sigmoid \( \implies \) ReLU \( \implies \) Normalization

Normalization

- Solid
- Horizontal
- Vertical
- Diagonal

Input values: \(-1 \quad 0 \quad +1\)
Neural Network

Input \xrightarrow{\text{Sigmoid}} \xrightarrow{\text{Sigmoid}} \xrightarrow{\text{ReLU}} \text{Normalization}

-1 0 +1

Normalization:
- Solid: 33%
- Horizontal: 33%
- Vertical: 33%
- Diagonal: 33%
Conclusions and Future Work

You should now be able to:

• Understand some fundamental machine learning jargon (neuron, activation, weight, sigmoid, ReLU, etc.)
• Calculate neuron activations
• Explain how a neural network can classify images

Future topics:

• Network training, hyperparameters, convolutional layers, segmentation

Credit to End-to-End Machine Learning Course