Introduction to Neural Networks

Klint Kanopka

kkanopka@stanford.edu

Wednesday, May 15, 2019

Outline

1. Introduction
   - Logistic Regression
   - Maximum Likelihood Estimation
   - Motivations for Neural Networks
2. Structure and Design
3. Estimation and Application
4. Other Neural Architectures
   - Recurrent Neural Networks
   - Convolutional Neural Network
   - Deep Learning

Logistic Regression

Logistic Regression

- Computer scientists call the logistic function the “sigmoid”

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

Logistic Regression

- Computer scientists call the logistic function the "sigmoid"

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Maps inputs to real numbers on the interval $(0, 1)$

Logistic Regression

- Computer scientists call the logistic function the “sigmoid”

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Maps inputs to real numbers on the interval $(0, 1)$
- Good for turning numbers into probabilities

Logistic Regression

- Computer scientists call the logistic function the "sigmoid"

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Maps inputs to real numbers on the interval $(0, 1)$
- Good for turning numbers into probabilities
- Has a really simple derivative:

$$\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x))$$

Logistic Regression

- Computer scientists call the logistic function the "sigmoid"

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

- Maps inputs to real numbers on the interval $(0, 1)$
- Good for turning numbers into probabilities
- Has a really simple derivative:

$$\frac{d\sigma}{dx} = \sigma(x)(1 - \sigma(x))$$

- This is super useful for neural nets, as we'll discuss later.

Maximum Likelihood Estimation

Maximum Likelihood Estimation

- Models use parameters to relate data within observations

Maximum Likelihood Estimation

- Models use parameters to relate data within observations
- MLE specifies a likelihood function that describes how likely (probable) it is to observe the data we have, given a set of model parameters

Maximum Likelihood Estimation

- Models use parameters to relate data within observations
- MLE specifies a likelihood function that describes how likely (probable) it is to observe the data we have, given a set of model parameters
- MLE searches for model parameters that maximize the likelihood function

Maximum Likelihood Estimation

- Models use parameters to relate data within observations
- MLE specifies a likelihood function that describes how likely (probable) it is to observe the data we have, given a set of model parameters
- MLE searches for model parameters that maximize the likelihood function
- Often work in terms of log likelihood

Machine Learning

Machine Learning

- Subfield of Artificial Intelligence

Machine Learning

- Subfield of Artificial Intelligence
- Computers “learn” models from data

Machine Learning

- Subfield of Artificial Intelligence
- Computers "learn" models from data
  - Instead of specifying a strict functional form (linear, logistic, etc.) and doing a regression, the computer learns the shape of the function and its parameters from the data

Machine Learning

- Subfield of Artificial Intelligence
- Computers “learn” models from data
    - Instead of specifying a strict functional form (linear, logistic, etc.) and doing a regression, the computer learns the shape of the function and its parameters from the data
    - The data the computer learns from is called the “training set”

Machine Learning

- Subfield of Artificial Intelligence
- Computers "learn" models from data
  - Instead of specifying a strict functional form (linear, logistic, etc.) and doing a regression, the computer learns the shape of the function and its parameters from the data
  - The data the computer learns from is called the "training set"
- Works great for prediction and classification tasks

Machine Learning

- Subfield of Artificial Intelligence
- Computers "learn" models from data
  - Instead of specifying a strict functional form (linear, logistic, etc.) and doing a regression, the computer learns the shape of the function and its parameters from the data
  - The data the computer learns from is called the "training set"
- Works great for prediction and classification tasks
- Less useful for inference (and causal inference), but this is changing

Why Neural Nets?

Why Neural Nets?

- We have tons of data
- Compute time is getting increasingly cheap

Why Neural Nets?

- We have tons of data
- Compute time is getting increasingly cheap
- The quality of an estimate is highly dependent on model specification

Why Neural Nets?

- We have tons of data
- Compute time is getting increasingly cheap
- The quality of an estimate is highly dependent on model specification
- More flexible estimation techniques allow computers to learn functional forms

Why Neural Nets?

- We have tons of data
- Compute time is getting increasingly cheap
- The quality of an estimate is highly dependent on model specification
- More flexible estimation techniques allow computers to learn functional forms
- Neural networks are "universal approximators" and can learn arbitrarily complex functional forms

Guiding Idea

Guiding Idea

- Neural networks are built out of *nodes*.

Guiding Idea

- Neural networks are built out of *nodes.*
- These nodes are organized into *hidden layers.*

Guiding Idea

- Neural networks are built out of *nodes.*
- These nodes are organized into *hidden layers.*
- For now, each node consists of an individual sigmoid function (logistic regression)

Guiding Idea

- Neural networks are built out of *nodes.*
- These nodes are organized into *hidden layers.*
- For now, each node consists of an individual sigmoid function (logistic regression)
- *Examples on the board...*

Training/Test Data

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data
- This is often terrible advice

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data
- This is often terrible advice
- The purpose of this is to hold out some data to test how well your model generalizes

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data
- This is often terrible advice
- The purpose of this is to hold out some data to test how well your model generalizes
- If you only have a little bit of data, this will work. If you have more, use a *dev set*.

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data
- This is often terrible advice
- The purpose of this is to hold out some data to test how well your model generalizes
- If you only have a little bit of data, this will work. If you have more, use a *dev set.*
- The dev set helps you tune *hyperparameters* like number of nodes, number of hidden layers, and learning rate

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data
- This is often terrible advice
- The purpose of this is to hold out some data to test how well your model generalizes
- If you only have a little bit of data, this will work. If you have more, use a *dev set.*
- The dev set helps you tune *hyperparameters* like number of nodes, number of hidden layers, and learning rate
- Good splits (train/dev/test) are 80/10/10, 90/5/5, or even 98/1/1 if you have a ton of data

Training/Test Data

- Conventional wisdom says to split your data 80/20 into training/test data
- This is often terrible advice
- The purpose of this is to hold out some data to test how well your model generalizes
- If you only have a little bit of data, this will work. If you have more, use a *dev set.*
- The dev set helps you tune *hyperparameters* like number of nodes, number of hidden layers, and learning rate
- Good splits (train/dev/test) are 80/10/10, 90/5/5, or even 98/1/1 if you have a ton of data
- More training data translates into a more robust model

Loss Functions

Loss Functions

- A function that defines how wrong your current predictions are

Loss Functions

- A function that defines how wrong your current predictions are
- For binary classification (1 or 0):

$$L = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

Loss Functions

- A function that defines how wrong your current predictions are
- For binary classification (1 or 0):

$$L = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

- For continuous variables:

$$L = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Loss Functions

- A function that defines how wrong your current predictions are
- For binary classification (1 or 0):

$$L = -\sum_{i=1}^{N} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

- For continuous variables:

$$L = \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

- General goal: Minimize how wrong you are

Estimation

- Estimation is done with an algorithm called *backpropagation*
- Similar to MLE, except now we search over parameters to *minimize* the loss function
- Updates to the individual parameters in the model are typically done via *gradient descent*
- Similar to Newton-Raphson, but doesn't require inverting a Hessian and requires more iterations to converge - in big data applications, often faster (smaller but quicker steps)

Application

Clone or download the files from the github repo:

https://github.com/klintkanopka/nn_workshop

Feed-Forward Neural Networks

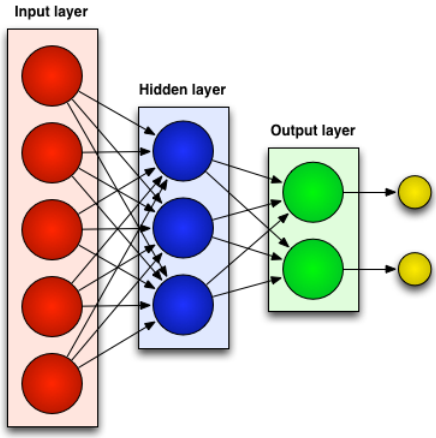


Figure 1: Feed-Forward Neural Network. Image Credit: Joseph Wilk

Recurrent Neural Networks

Recurrent Neural Networks

- Operate on series or sequence of data

Recurrent Neural Networks

- Operate on series or sequence of data
- The same network is applied at each time step

Recurrent Neural Networks

- Operate on series or sequence of data
- The same network is applied at each time step
- Each time step also takes the output of the previous time step as input

Recurrent Neural Networks

- Operate on series or sequence of data
- The same network is applied at each time step
- Each time step also takes the output of the previous time step as input
- Specialized versions of the RNN work to help "remember" older data:

Recurrent Neural Networks

- Operate on series or sequence of data
- The same network is applied at each time step
- Each time step also takes the output of the previous time step as input
- Specialized versions of the RNN work to help "remember" older data:
    - Gated Recurrent Unit (GRU)

Recurrent Neural Networks

- Operate on series or sequence of data
- The same network is applied at each time step
- Each time step also takes the output of the previous time step as input
- Specialized versions of the RNN work to help “remember” older data:
    - Gated Recurrent Unit (GRU)
    - Long Short Term Memory (LSTM)

Recurrent Neural Networks

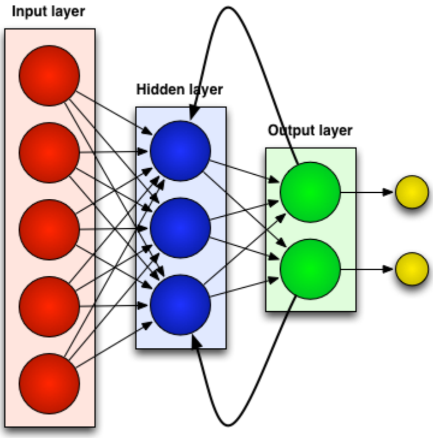


Figure 2: Recurrent Neural Network. Image Credit: Joseph Wilk

Recurrent Neural Networks

Useful for:

Recurrent Neural Networks

Useful for:

- Sequences of data

Recurrent Neural Networks

Useful for:

- Sequences of data
- Item responses

Recurrent Neural Networks

Useful for:

- Sequences of data
- Item responses
- Text

Recurrent Neural Networks

Useful for:

- Sequences of data
- Item responses
- Text
- Time series data

Useful for:

- Sequences of data
- Item responses
- Text
- Time series data
- Panel data

Convolutional Neural Networks

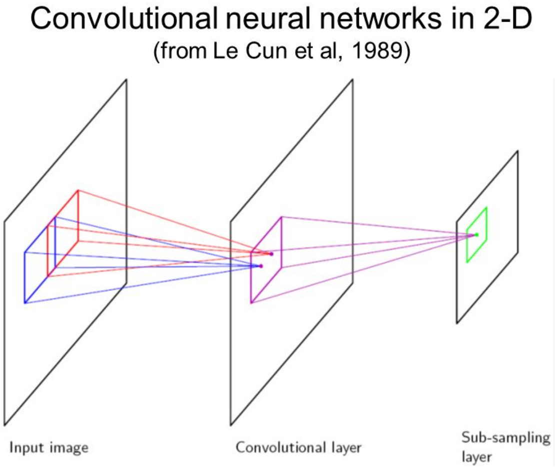
Convolutional Neural Networks

- Uses a window (or kernel) to perform arithmetic reductions on subsets of the input

Convolutional Neural Networks

- Uses a window (or kernel) to perform arithmetic reductions on subsets of the input
- These windows are called filters and can learn to detect features in the input

Convolutional Neural Networks



Convolutional Neural Networks

Useful for:

Convolutional Neural Networks

Useful for:

- Arrays of data

Convolutional Neural Networks

Useful for:

- Arrays of data
- Images

Convolutional Neural Networks

Useful for:

- Arrays of data
- Images
- Sequences of data

Convolutional Neural Networks

Useful for:

- Arrays of data
- Images
- Sequences of data
- Text

Convolutional Neural Networks

Useful for:

- Arrays of data
- Images
- Sequences of data
- Text
- Signal processing

Deep Learning

Deep Learning

- Tech bro buzzword for large neural networks

Deep Learning

- Tech bro buzzword for large neural networks
- "Deep" refers lots of hidden layers

Deep Learning

- Tech bro buzzword for large neural networks
- "Deep" refers lots of hidden layers
- Leverages huge amounts of data

Deep Learning

- Tech bro buzzword for large neural networks
- "Deep" refers lots of hidden layers
- Leverages huge amounts of data
- Trained/run on cloud servers with GPUs/TPUs

Deep Learning

- Tech bro buzzword for large neural networks
- "Deep" refers lots of hidden layers
- Leverages huge amounts of data
- Trained/run on cloud servers with GPUs/TPUs
- Good word to stick into things you want to get published or funded

I'm Done

Thank you for your attention.

Klint Kanopka
kkanopka@stanford.edu

To download today's materials:
https://github.com/klintkanopka/nn_workshop