Neural Networks

PSC 8185: Machine Learning for Social Science

Iris Malone

March 28, 2022

Recap

Where We've Been:

- Slow learning methods tend to outperform other methods
- Hyperplanes are a type of non-parametric classifier
- Non-parametric classifiers tradeoff low bias for more flexibility
- Kernels measure the similarity of two observations

Recap

Where We've Been:

- Slow learning methods tend to outperform other methods
- Hyperplanes are a type of non-parametric classifier
- Non-parametric classifiers tradeoff low bias for more flexibility
- Kernels measure the similarity of two observations

New Terminology:

- Maximal Margin Classifier
- Support Vectors
- Support Vector Machine
- Kernels

Agenda

1. Deep Learning
2. Neural Network Architecture
3. Recurrent Neural Nets
4. Convolutional Neural Nets

Deep Learning

Deep Learning

Motivation: We have surveyed a number of historical parametric and non-parametric approaches to pattern recognition

Deep Learning

Motivation: We have surveyed a number of historical parametric and non-parametric approaches to pattern recognition

- 1800s-1950s: Linear Models + Principal Component Analysis
- 1960s-2010s: Random Forests, Boosting, Support Vector Machines, Bayesian
- Today: Deep Learning

Limits to Conventional Approaches

- Computational Expenses
    - Struggle to handle growth in big data
    - Slow
    - Dependent on assigned inputs
    - Limits to pattern recognition

Limits to Conventional Approaches

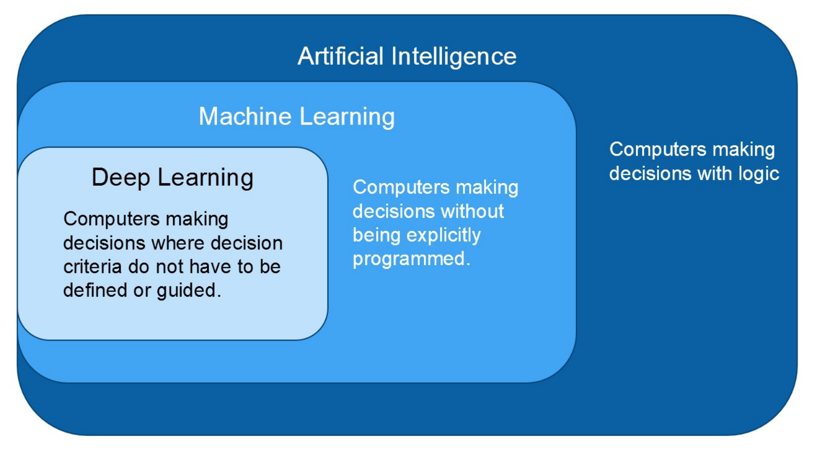
- Computational Expenses
  - Struggle to handle growth in big data
  - Slow
  - Dependent on assigned inputs
  - Limits to pattern recognition
- Static Approaches
  - Little to no learning
  - Little iterative processes
  - Little reaction to prior information/events

Limits to Conventional Approaches

- Computational Expenses
  - Struggle to handle growth in big data
  - Slow
  - Dependent on assigned inputs
  - Limits to pattern recognition
- Static Approaches
  - Little to no learning
  - Little iterative processes
  - Little reaction to prior information/events

Solution: Deep Learning

Comparison of Deep Learning and ML



Deep Learning Results Improve on Conventional ML

- More Data
- More Model Complexity
- More Computational Power

Characteristics of Deep Learning

- Input of highly complex and multi-dimensional data (e.g. images, audio, time series data)

Characteristics of Deep Learning

- Input of highly complex and multi-dimensional data (e.g. images, audio, time series data)
- **Automated Feature Learning:** Automatic variable selection and optimization

Characteristics of Deep Learning

- Input of highly complex and multi-dimensional data (e.g. images, audio, time series data)
- **Automated Feature Learning:** Automatic variable selection and optimization
- **Artificial Neural Networks**

Intuition of Deep Learning

Main Idea: Deep learning mimics human decision-making in pattern recognition and classification

Intuition of Deep Learning

Main Idea: Deep learning mimics human decision-making in pattern recognition and classification

Intuition of Deep Learning

Main Idea: Deep learning mimics human decision-making in pattern recognition and classification

- Take a classification task and breaks it down into different steps (layers).

Intuition of Deep Learning

Main Idea: Deep learning mimics human decision-making in pattern recognition and classification

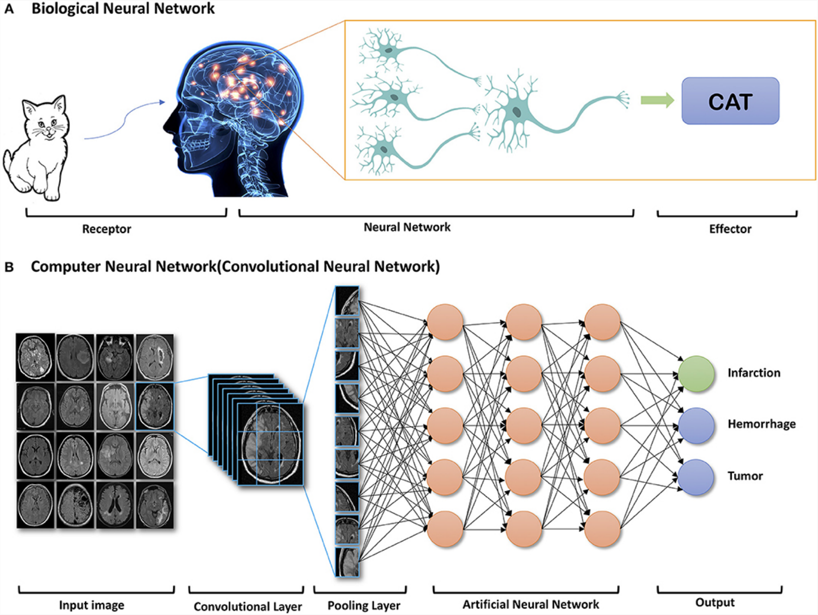
- Take a classification task and breaks it down into different steps (layers).
- Feed-forward (push forward) information from each previous input layer to assemble next layer

Intuition of Deep Learning

Main Idea: Deep learning mimics human decision-making in pattern recognition and classification

- Take a classification task and breaks it down into different steps (layers).
- Feed-forward (push forward) information from each previous input layer to assemble next layer
- Output is function of many inter-related input layers

Intuition of Deep Learning



Artificial Neural Networks May Sound Scary, But Aren't

Kareem Carr
@kareem_carr

I hate to tell you this but most deep learning models are just three or more logistic regressions in a trench coat.

8:20 PM · Jan 18, 2021 · TweetDeck

396 Retweets 41 Quote Tweets 4.3K Likes

Neural Network Architecture

Neural Network Vocabulary

- Network Layers
  - **Input Layer:** Predictors/information fed into model
  - **Hidden Layer:** Unobservable function in middle of network which relates different predictors together
  - **Output Layer:** Predictions from model

Neural Network Vocabulary

- Network Layers
  - **Input Layer:** Predictors/information fed into model
  - **Hidden Layer:** Unobservable function in middle of network which relates different predictors together
  - **Output Layer:** Predictions from model
- **Neurons:** The underlying modeling structures - they take inputs, applies mathematical transformation, and produces ('fires') an output

$$\hat{y} = \sum(\text{weights} \times \text{inputs}) + \text{bias}$$

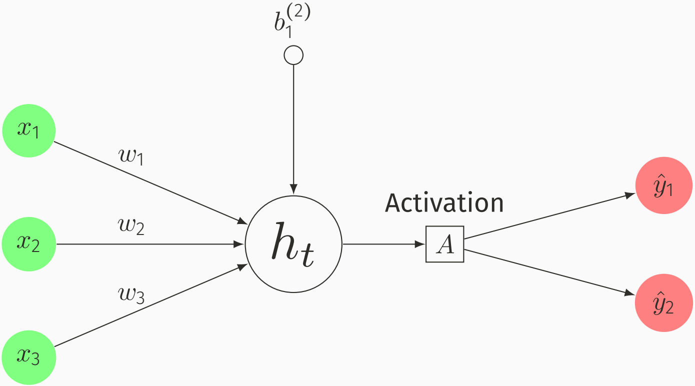
Neural Network Vocabulary

- Network Layers
  - **Input Layer:** Predictors/information fed into model
  - **Hidden Layer:** Unobservable function in middle of network which relates different predictors together
  - **Output Layer:** Predictions from model
- **Neurons:** The underlying modeling structures - they take inputs, applies mathematical transformation, and produces ('fires') an output

$$\hat{y} = \sum(\text{weights} \times \text{inputs}) + \text{bias}$$

- **Activation Function:** Determines whether the expected output is in acceptable bounds and neuron should be 'fired' ('activated')

Basic Neural Network Architecture



An Analogy: 2SLS

Recall: In conventional regression methods, we sometimes want to model a process, but can't directly observe the effect of X on Y due to endogeneity concerns. The solution is **two-stage least squares modeling**

An Analogy: 2SLS

Recall: In conventional regression methods, we sometimes want to model a process, but can't directly observe the effect of X on Y due to endogeneity concerns. The solution is **two-stage least squares modeling**

Analogy: A NN is like a two-stage regression model where the hidden layer h_t is not directly observable

An Analogy: 2SLS

Two Equations:
First-Stage (estimate the effect of instrument on treatment):

$$h = \pi_o + \pi_1(X) + \nu$$

Second-Stage (estimate the effect of treatment on outcome):

$$Y = \beta_0 + \beta_1 h + u$$

Neural Network Procedure

- Each input is multiplied by a weight, e.g. $(x_1 * w_1), (x_2 * w_2), \ldots (x_p * w_p)$

Neural Network Procedure

- Each input is multiplied by a weight, e.g. $(x_1 * w_1), (x_2 * w_2), \ldots (x_p * w_p)$
- Hidden layer is function of weighted inputs (plus bias term b)

$$h_t = (x_1 * w_1) + (x_2 * w_2) + \cdots + (x_p * w_p) + b$$

Neural Network Procedure

- Each input is multiplied by a weight, e.g. $(x_1 * w_1), (x_2 * w_2), \ldots (x_p * w_p)$
- Hidden layer is function of weighted inputs (plus bias term b)

$$h_t = (x_1 * w_1) + (x_2 * w_2) + \cdots + (x_p * w_p) + b$$

- Activation function transforms hidden layer to be within some scale, i.e. unbounded input → bounded input

$$A(h_t) = f(x_1 * w_1 + x_2 * w_2 + \cdots + x_p * w_p + b)$$

Neural Network Procedure

- Each input is multiplied by a weight, e.g. $(x_1 * w_1), (x_2 * w_2), \dots (x_p * w_p)$
- Hidden layer is function of weighted inputs (plus bias term b)

$$h_t = (x_1 * w_1) + (x_2 * w_2) + \cdots + (x_p * w_p) + b$$

- Activation function transforms hidden layer to be within some scale, i.e. unbounded input → bounded input

$$A(h_t) = f(x_1 * w_1 + x_2 * w_2 + \cdots + x_p * w_p + b)$$

- Output $\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n$ is transformed hidden layer

Neural Network Loss Function

Key Hyperparameters:

- weights w_i
- bias b

Loss functions $L(w_i, b)$ aims to minimize error with respect to weights.

Backpropagation

Minimize loss function through **backpropagation**, which is a system of calculating partial derivatives to find optimal w

$$\frac{\partial L}{\partial w_i} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w}$$

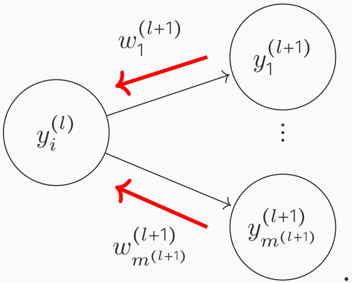
$$\frac{\partial L}{\partial y} = \frac{\partial (1-\hat{y})^2}{\partial y}$$

$$\frac{\partial y}{\partial w} = \frac{\partial \hat{y}}{\partial h}\frac{\partial h}{\partial w}$$

$$\frac{\partial \hat{y}}{\partial h} = w_i \times f(x_1 * w_1 + x_2 * w_2 + \cdots + x_p * w_p + b)$$

$$\frac{\partial h}{\partial w} = x_i \times f'(x_1 * w_1 + x_2 * w_2 + \cdots + x_p * w_p + b)$$

Backpropagation



Stochastic Gradient Descent

We apply a specific algorithm known as **stochastic gradient descent** which minimizes loss function by changing weights and bias:

$$w_1 - \lambda \frac{\partial L}{\partial w_1} \to w_1^*$$

Stochastic Gradient Descent

We apply a specific algorithm known as **stochastic gradient descent** which minimizes loss function by changing weights and bias:

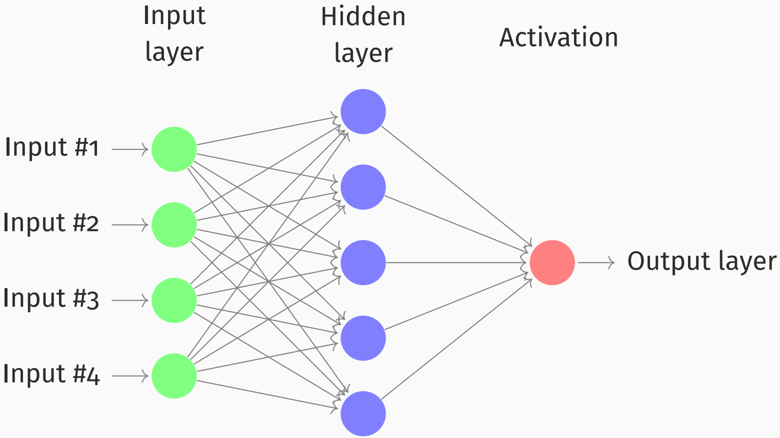
$$w_1 - \lambda \frac{\partial L}{\partial w_1} \to w_1^*$$

This is similar to how boosting worked:

- λ is a learning rate
- If $\frac{\partial L}{\partial w_1} > 0$, then weight decreases and loss decreases
- If $\frac{\partial L}{\partial w_1} < 0$, then weight increases and loss decreases

Complicated Neural Network Architecture

A neural network can have any number of layers with any number of neurons in those layers.



Choosing Number of Neurons

Neurons affect the bias-variance tradeoff of the model:

- More neurons → increased flexibility, but higher bias
- Fewer neurons → underfitting by blocking flow of information

Choosing Number of Neurons

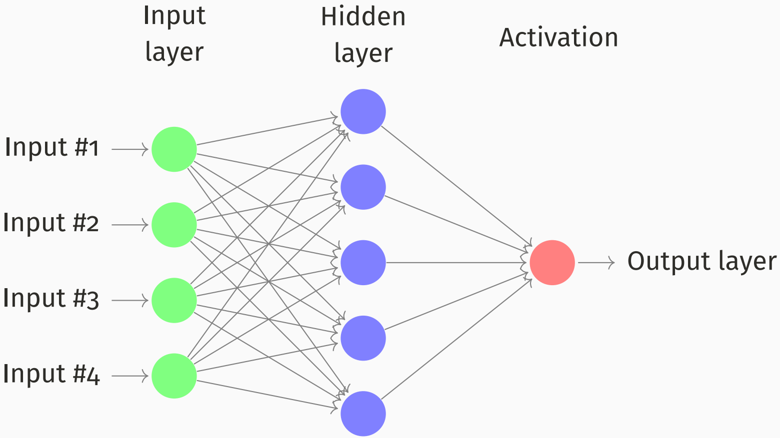
Neurons affect the bias-variance tradeoff of the model:

- More neurons → increased flexibility, but higher bias
- Fewer neurons → underfitting by blocking flow of information

Rule of Thumb: Number Neurons = Number of Predictors +1

Complicated Neural Network Architecture

A neural network can have any number of layers with any number of neurons in those layers.



Activation Functions

Main Idea: Activation functions transform the hidden layers by constraining them to be in between particular values. It takes an unbounded input ($-\infty$ to ∞) $\rightarrow$ bounded input $[a, b]$

Activation Functions

Main Idea: Activation functions transform the hidden layers by constraining them to be in between particular values. It takes an unbounded input ($-\infty$ to ∞) $\rightarrow$ bounded input $[a, b]$

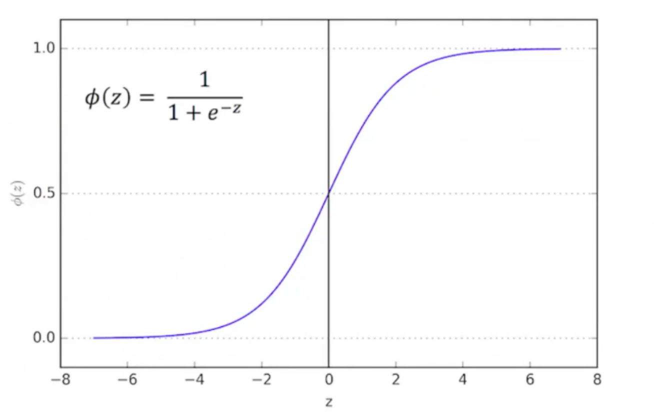
Common Activation Functions:

- Sigmoid Function: $[0, 1]$
- Rectified Linear Unit (ReLU): $[0, \infty]$
- Hyperbolic Tangent (TanH): $[-1, 1]$

Recall: Logit Function

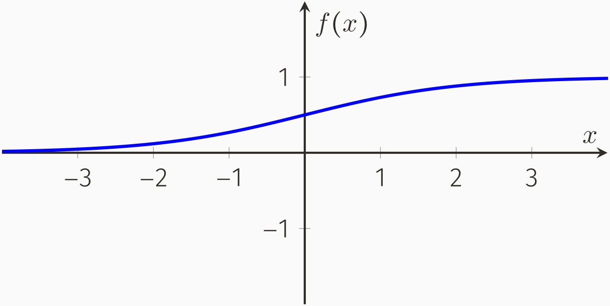
Logit Function (Sigmoid Function):

$$P(y = 1 \mid x) = f(X\beta) = \frac{e^{X\beta}}{1 + e^{X\beta}} = \frac{1}{1 + e^{-X\beta}}$$



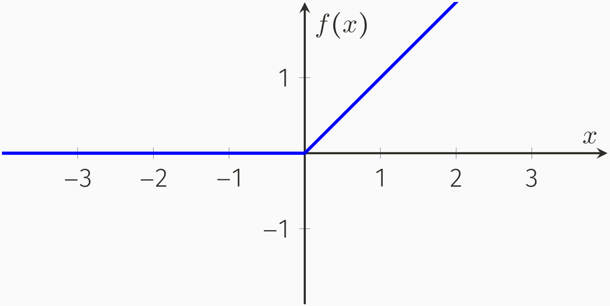
Sigmoid Activation Function

Sigmoid Activation Function: $A = \frac{1}{1+e^{-z}}$



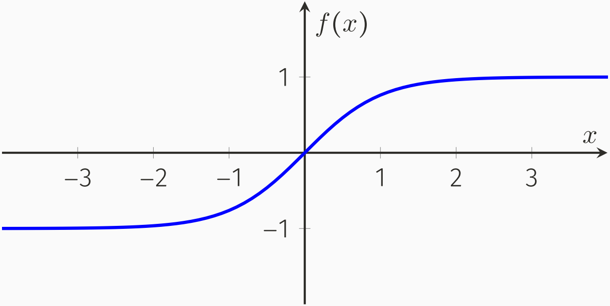
ReLU Activation Function

ReLU Activation FunctionTanh: $A = \max(0, z)$



Hyperbolic Tangent Function

Tanh Activation Function: $A = \frac{e^z e^{-z}}{e^z + e^{-z}}$



How does the Model Learn?

- **Batch Size:**

- **Epochs:**

- **Dropout:**

How does the Model Learn?

- **Batch Size:** Number of sequences the model studies before updating the weights.
  - Small batch size → more frequent updating (without observing all data)
  - Small batch size → fast, but worse accuracy
  - Large batch size → less frequent updating, but less sensitive to noise
- **Epochs:**
- **Dropout:**

How does the Model Learn?

- **Batch Size:** Number of sequences the model studies before updating the weights.

- **Epochs:** Measures the number of times the model iterates through all of the training data.
  - Large batch size → fewer iterations to go through full pass of data
  - Increasing passes is like bootstrap → increased likelihood population convergence
- **Dropout:**

How does the Model Learn?

- **Batch Size:** Number of sequences the model studies before updating the weights.
- **Epochs:** Measures the number of times the model iterates through all of the training data.
- **Dropout:** Prevents overfitting. Randomly removes a certain number of observations from each input when creating the model.

Types of Neural Networks

1. **Feed-forward Neural Networks:** Perform basic pattern recognition (information passes in a single direction)

Types of Neural Networks

1. **Feed-forward Neural Networks:** Perform basic pattern recognition (information passes in a single direction)
2. **Recurrent Neural Networks (RNN):** Performs Natural Language Processing (NLP), Speech Recognition, and Times Series Forecasting

Types of Neural Networks

1. **Feed-forward Neural Networks:** Perform basic pattern recognition (information passes in a single direction)
2. **Recurrent Neural Networks (RNN):** Performs Natural Language Processing (NLP), Speech Recognition, and Times Series Forecasting
3. **Convolutional Neural Networks (CNN):** Performs object recognition and image analysis

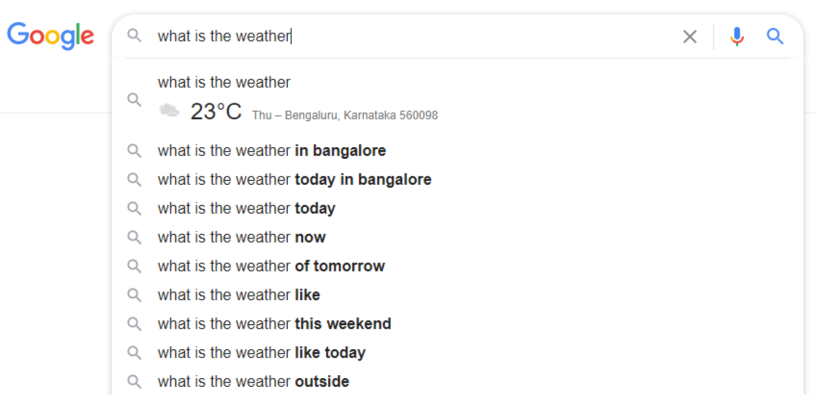
Recurrent Neural Nets

Motivation for RNN

Motivation: There are many prediction problems which are time- or sequence-dependent

- Stock Market Forecasts
- Supply/Demand Needs
- Conflict Risk
- Text Prediction

Google RNN Prediction



RNN and NLP

I forced a bot to watch over 1,000 hours of Olive Garden commercials and then asked it to write an Olive Garden commercial of its own. Here is the first page.

I forced a bot to watch over 1,000 hours of Olive Garden commercials and then asked it to write an Olive Garden commercial of its own. Here is the first page.

'E GARDEN COMMERCIAL

. OLIVE GARDEN RESTAURANT

:oup of FRIENDS laughs at a dinner table. A WAITR
leliver what could be considered food.

WAITRESS
Pasta nachos for you.

ee the pasta nachos. They're warm and defeated.

FRIEND 1
The menu is here.

WAITRESS
Lasagna wings with extra Italy.

ee the lasagna wings. There's more Italy than ne

FRIEND 2
I shall eat Italian citizens.

FRIEND 3
Leave without me. I'm home.

WAITRESS
Gluten Classico. From the kitchen.

Gluten Classico. We believe the waitress tl
he kitchen. We have no reason not to believ

4 says nothing.

FRIEND 1
What is wrong, Friend 4?

4 says nothing.

FRIEND 2
Friend 4, what is wrong, Friend 4?

4 smiles wide. Her mouth is full of secret

An Actual RNN NLP Example

In The Alteri Silence by Forest_of_Holly for roscreens41

Snape receives life after plants to do by work over whether they get into. Just Hell.

A Second Chance by DarkCorgi

Snape had a second thing, and that is better than anything for for the rest of his life.

Mirror by orphan_account

Severus Snape tries to get a lot of dragons and that was to be more than he didn't expect to continue. He has always been a bit of an old and a baby to stay the way he'd been the brother at Hogwarts and he keeps the chance of meeting…

Deception by FlyingEyes

Snape is a British Robes of interesting things and worrys like a little fun and sees the pretty battle for a while.

Conventional Approaches to Time Series Problems

- Conventional Approaches
  - ARIMA
  - CART

Conventional Approaches to Time Series Problems

- Conventional Approaches
  - ARIMA
  - CART
- Limits
  - ARIMA: Parametric/Inflexibility
  - CART: Miss Temporal Dependencies/Sequencing
  - Both: Poor Long-Term Forecasting

Conventional Approaches to Time Series Problems

- Conventional Approaches
  - ARIMA
  - CART
- Limits
  - ARIMA: Parametric/Inflexibility
  - CART: Miss Temporal Dependencies/Sequencing
  - Both: Poor Long-Term Forecasting

Solution: Recurrent Neural Nets

RNN

Main Idea:

- RNNs loop through previous outputs and learn from previous cases to improve model performance
- Memory previous states + new input states → outputs

RNN

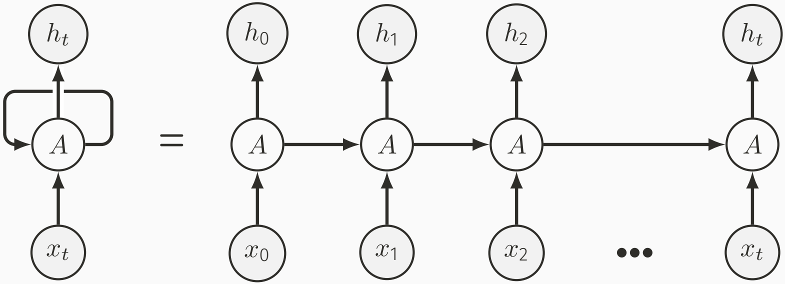
Main Idea:

- RNNs loop through previous outputs and learn from previous cases to improve model performance
- Memory previous states + new input states → outputs

Hidden layer output h_t is now function of weighted input state x_t and previous hidden states h_{t-1} transformed by A.

$$h_t = A(h_{t-1}, x_t) = tanh(W_{yh}h_{t-1} + W_{xh}x_t) \tag{1}$$

How Recurrent Neural Networks Work



Advantages to RNN

- Assumes current events dependent on deep past
- Long Memory $\rightarrow$ takes input of any length
- Learns from temporal dependencies
- Improved long-term accuracy forecasting

Limits to RNN

- Black Box

Limits to RNN

- Black Box
- Assumes spatial independence between countries → CNN

Limits to RNN

- Black Box
- Assumes spatial independence between countries → CNN
- Vanishing Gradient Problem
  - If $\sum_{i=1}^{n} \frac{\partial L(w_i)}{\partial w_i} \rightarrow 0$, then slow to no learning
  - Solution: Long-Short Term Memory (gates govern flow of information)

Limits to RNN

- Black Box
- Assumes spatial independence between countries → CNN
- Vanishing Gradient Problem
  - If $\sum_{i=1}^{n} \frac{\partial L(w_i)}{\partial w_i} \to 0$, then slow to no learning
  - Solution: Long-Short Term Memory (gates govern flow of information)
- Risk of Over-Fitting
  - Solution: Dropout Set

Limits to RNN

- Black Box
- Assumes spatial independence between countries → CNN
- Vanishing Gradient Problem
  - If $\sum_{i=1}^{n} \frac{\partial L(w_i)}{\partial w_i} \to 0$, then slow to no learning
  - Solution: Long-Short Term Memory (gates govern flow of information)
- Risk of Over-Fitting
  - Solution: Dropout Set
- Training Challenges
  - Hyperparameter Tuning
    - Epochs
    - Neurons
    - Batch Size
  - Solution: Loops, updating, numerous iterations

Convolutional Neural Nets

Motivation for CNN

Motivation: There are many prediction problems which are spatial-dependent

- Image Classification
- Disease Contagion
- Natural Disaster Risk Zones
- Conflict Spillover

CNN

Main Idea: Take a 2-dimensional input and apply a 2-dimensional weight (kernel) to perform feature selection and create a neural network.

CNN

Main Idea: Take a 2-dimensional input and apply a 2-dimensional weight (kernel) to perform feature selection and create a neural network.

New Attributes:

CNN

Main Idea: Take a 2-dimensional input and apply a 2-dimensional weight (kernel) to perform feature selection and create a neural network.

New Attributes:

- **Filter kernel**: the 2d array of weights (w_i) applied to the input data

CNN

Main Idea: Take a 2-dimensional input and apply a 2-dimensional weight (kernel) to perform feature selection and create a neural network.

New Attributes:

- **Filter kernel**: the 2d array of weights (w_i) applied to the input data
- **Convolutional layer:** performs linear operation to multiply and weight 2d inputs (like hidden layer)

CNN

Main Idea: Take a 2-dimensional input and apply a 2-dimensional weight (kernel) to perform feature selection and create a neural network.

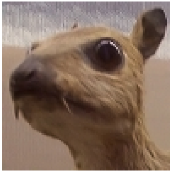
New Attributes:

- **Filter kernel**: the 2d array of weights (w_i) applied to the input data
- **Convolutional layer:** performs linear operation to multiply and weight 2d inputs (like hidden layer)
- **Feature Map** makes the links - it encodes the presence and degree of presence of the feature it detects.

CNN Procedure

- Take 2d input and apply **filter kernel** to the input layer
- Product of input & filter kernel → **Feature Map**

Input image



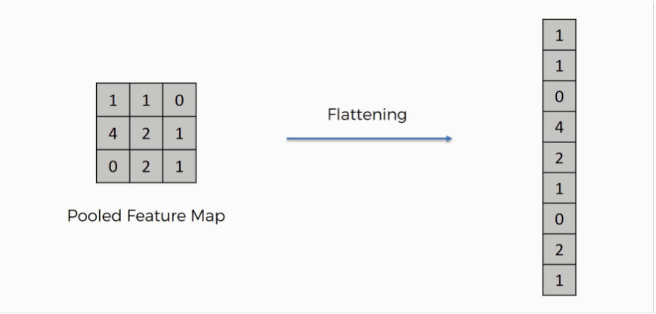
Convolution Kernel

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

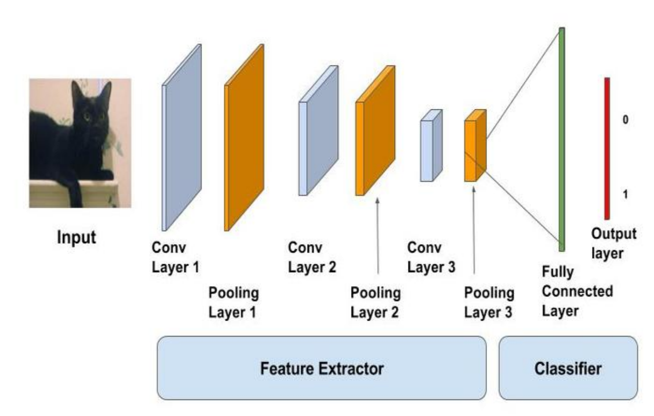
Feature map

CNN Procedure

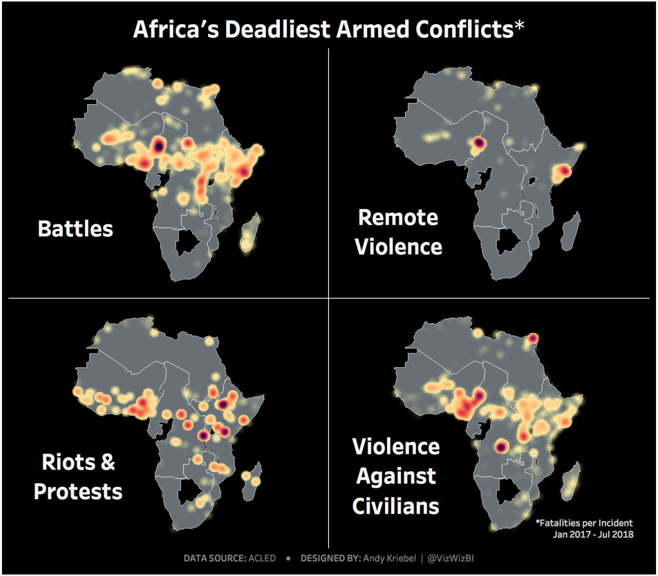
- Take **feature map** and apply flattening transformation
- Use flattened transformation as input layer for artificial layer
- Build conventional NN and get 1d output
- Apply transformation to restore to 2d output
- Iteratively repeat



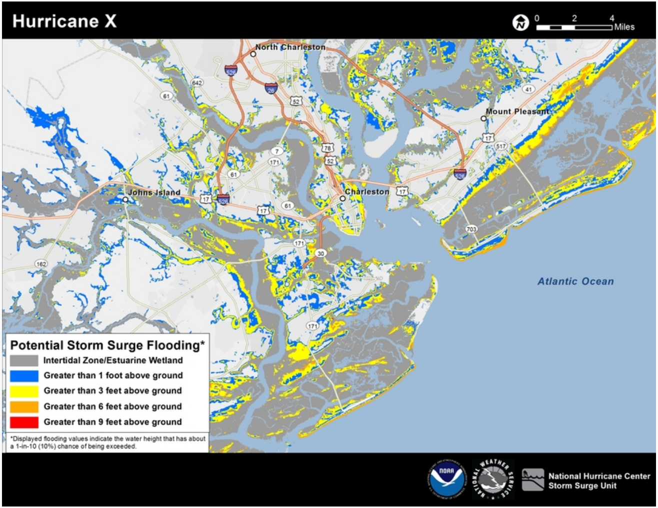
2D CNN Architecture



Examples of CNN



Examples of CNN



Limits to CNN

- Black Box
- Assumes temporal independence
- Vanishing gradient problem
- Very hard to train
- Computational Expensive (image quality)

Conclusion

- NN are like 2SLS; they feed forward information from inputs to output via hidden layer
- 3 types of artificial neural nets: feedforward, recurrent, and convolutional
- NN are hard to tune, but perform very well when they work