Task 3 - Team 17

Book Name and link:
Speech and Language Processing, Jurafsky, 3rd Ed, 2022, Pearson
https://web.stanford.edu/~jurafsky/slp3/

Chapter Name and link:
7 - Neural Networks and Neural Language Models https://web.stanford.edu/~jurafsky/slp3/7.pdf

Submitted by:
22366020 - Muhammed Yaseen Morshed Adib
20201228 - Munim Bin Muquith
22366023 - Sovon Chakraborty

Neural Networks and Neural Language Models

Key points of this chapter:
- Feed Forward Networks
- Huge number of interprocessing units
- How Informations are stored in neurons?
- Weight of each link
- Backward propagation for error recovery
- Has a learning ability
- Complex computational power

Units

- bias
  Z = b +Σ wixi
- Weighted Sum using vectors
  z = wx + b
- Activation functions
  - Types
    - Sigmoid Function
      y = σ(z) = 1/1+e^{-z}
      it maps the output in [0,1]
    - tanh Function
      y = (e^{z} - e^{-z})/(e^{z} + e^{-z})
      it maps output in [1,-1]
    - Relu function
      y=Relu(z)=max(z,0)
      - Provides non-linearity
      - Removes vanishing gradient problem

XOR Problems

- Single perceptron can not solve those functions which are not linearly separable
- For example.XOR is an example
- To solve this problem, Neural network can play a vital role
- Multilayer perceptron can solve Xor problem

Feed Forward Neural Networks

- Is a multilayer network
- No cycle
- Output is passed from lower layer to higher layer
- Has a higher number of hidden layers
- Hidden layer consists of weighted sum for non linearity
- Fully connected layer allows flexibility

Working Procedure

$z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$
$a^{[1]} = g^{[1]}(z^{[1]})$
$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$
$a^{[2]} = g^{[2]}(z^{[2]})$
$\hat{y} = a^{[2]}$

here subscriptors in bracket is the layer number.

$W^{[1]}$ = weight matrix of first hidden layer
$b^{[1]}$ = bias
$a^{[1]}$ = output layer
$g^{[1]}$ = activation function

Feedforward networks for NLP: Classification

- NLP Classification
  - classifier Approach's
    - x= [x1, x2...xN]
      h=σ(wx+b)
      Z=Uh
      y=softmax(Z)
      - Continuosly learns features
      - Pooling is used to the embedding of words
      - Multiple embedding can be converted
      - Into single embedding by just summing

      $$x_{mean} = 1/x \sum_{i=1}^{n} e(w_i)$$
    - Element wise embedding
      - h=σ(wx+b)
      - Z=Uh
      - y=softmax(Z)
      - Pre-training is required for the approach

Feedforward Neural Language Modeling

- Neural Language Modeling
  - Classifier Approaches
    - Neural Language Models(NLM) can handle more histories than n-gram
      - Complex, slower, less-interpretable
      - Takes previous words(w1, w2...wn)at time 't' and outputs probability distribution of nextwords.
      - Represents words by their prior word-embeddings
      - Word embedding allows NLM's to generalize unseen data properly
    - Forward interfacing is running a forward pass in the network to produce a probability distribution
      - One-hot vector is used here where one element is equal to 1 and other are '0'
      - Has a moving window that can watch words from past
      - Select embedding from E
      - Multiply I,y,w
      - Multiply by U
      - Apply softmax

Training Neural Nets

- Supervised machine learning approach
- Loss function for modeling distance between the system output and the goal output
- Cross entropy loss is most common
  $L_{CE}(\hat{y}, y) = -\log(y|x) = -[y\log\hat{y} + (1-y)\log(1-\hat{y})]$
  used for binary classification
- Gradient descent for minimizing loss function
- Gradient descent is calculated by Error back propagation
- Backward differentiation is a notion of computation graph
  - Computing mathematical expression
  - In forward pass of computation graph the value of inputs are computed
  - In backward pass, it is very important
  - Used for weight update
  - Backward differentiats uses chain rule
  - Gradients are computed in these process
  - For backward pass loss is always computed
  - Updating learning rate in backward propagration
    - Dropout is applied to prevent overfitting
    - Tuning hyper parameters are important
    - Trainable parameters should be reduce

Training the neural language model

- Let parameters be,θ=E, W, U, σ
- Freeze E
- Done using gradient descent
- Predicts upcoming words

$L_{CE}(\hat{y}, y) = -\log \hat{y}_i$
$L_{CE} = -\text{Logp}(w_t|w_{t-1}...w_{t-n+1})$