CS 6120/CS 4120: Natural Language Processing
Instructor: Prof. Lu Wang
Northeastern University
Webpage: www.ccs.neu.edu/home/luwang
Outline

• Maximum Entropy
• Feedforward Neural Networks
• Recurrent Neural Networks
Maximum Entropy (MaxEnt)

• Or logistic regression
Features

- In these slides and most MaxEnt work: features (or feature functions) $f$ are elementary pieces of evidence that link aspects of what we observe $d$ with a category $c$ that we want to predict.
- A feature is a function with a bounded real value: $f: C \times D \rightarrow \mathbb{R}$.
Example Task: Named Entity Type

LOCATION in Arcadia
LOCATION in Québec
DRUG taking Zantac
PERSON saw Sue
Example features

- $f_1(c, d) \equiv [c = \text{LOCATION} \land w_{-1} = \text{“in”} \land \text{isCapitalized}(w)]$
- $f_2(c, d) \equiv [c = \text{LOCATION} \land \text{hasAccentedLatinChar}(w)]$
- $f_3(c, d) \equiv [c = \text{DRUG} \land \text{ends}(w, \text{“c”})]$

Models will assign to each feature a **weight**:  
- A positive weight votes that this configuration is likely correct  
- A negative weight votes that this configuration is likely incorrect
Example features

- \( f_1(c, d) \equiv [c = \text{LOCATION} \land w_{-1} = "in" \land \text{isCapitalized}(w)] \rightarrow \text{weight} \ 1.8 \)
- \( f_2(c, d) \equiv [c = \text{LOCATION} \land \text{hasAccentedLatinChar}(w)] \rightarrow \text{weight} \ -0.6 \)
- \( f_3(c, d) \equiv [c = \text{DRUG} \land \text{ends}(w, "c") ] \rightarrow \text{weight} \ 0.3 \)

- Weights will be learned by training on a labeled dataset
More about feature functions:
an indicator function – a yes/no boolean matching function – of properties of the input and a particular class

\[ f_i(c, d) \equiv [\Phi(d) \land c = c_j] \]  
[Value is 0 or 1]
Feature-Based Models

• The decision about a data point is based only on the features active at that point.

**Text Classification**

Data

**BUSINESS:** Stocks hit a yearly low …

Label: **BUSINESS**
Features
{…, stocks, hit, a, yearly, low, …}

**Word Sense Disambiguation**

Data

… to restructure bank:MONEY debt.

Label: **MONEY**
Features
{…, $w_{-1}$=restructure, $w_{+1}$=debt, $L=12$, …}

**POS Tagging**

Data

DT JJ NN …
The previous fall …

Label: **NN**
Features
{$w$=fall, $t_{-1}$=JJ $w_{-1}$=previous}
Feature-Based Linear Classifiers

• Linear classifiers at classification time:
  • Linear function from feature sets \{f_i\} to classes \{c\}.
  • Assign a weight \lambda_i to each feature \(f_i\).
  • We consider each class for sample \(d\).
  • For a pair \((c,d)\), features vote with their weights:
    • \(\text{vote}(c) = \sum_i \lambda_i f_i(c,d)\)

PERSON
  in Québec

LOCATION
  in Québec

DRUG
  in Québec

• Choose the class \(c\) which maximizes \(\sum_i \lambda_i f_i(c,d)\)
• Maximum Entropy:
  • Make a probabilistic model from the linear combination $\sum \lambda_i f_i(c,d)$

$$P(c \mid d, \lambda) = \frac{\exp \sum_i \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

- Makes votes positive
- Normalizes votes
Feature-Based Linear Classifiers

- $f_1(c, d) \equiv [c = \text{LOCATION} \land w_{-1} = \text{“in”} \land \text{isCapitalized}(w)] \rightarrow \text{weight 1.8}$
- $f_2(c, d) \equiv [c = \text{LOCATION} \land \text{hasAccentedLatinChar}(w)] \rightarrow \text{weight } -0.6$
- $f_3(c, d) \equiv [c = \text{DRUG} \land \text{ends}(w, \text{“c”})] \rightarrow \text{weight 0.3}$
Maximum Entropy:

- Make a probabilistic model from the linear combination $\sum \lambda_i f_i(c,d)$

$$P(c \mid d, \lambda) = \frac{\exp \sum \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

- Makes votes positive
- Normalizes votes
Maximum Entropy:

- Make a probabilistic model from the linear combination $\sum \lambda_i f_i(c,d)$

$$P(c | d, \lambda) = \frac{\exp \sum \lambda_i f_i(c,d)}{\sum_{c'} \exp \sum_i \lambda_i f_i(c',d)}$$

- $P(LOCATION|\text{in Québec}) = e^{1.8} e^{-0.6} / (e^{1.8} e^{-0.6} + e^{0.3} + e^0) = 0.586$
- $P(\text{DRUG}|\text{in Québec}) = e^{0.3} / (e^{1.8} e^{-0.6} + e^{0.3} + e^0) = 0.238$
- $P(\text{PERSON}|\text{in Québec}) = e^0 / (e^{1.8} e^{-0.6} + e^{0.3} + e^0) = 0.176$

- The weights are the parameters of the probability model, combined via a “soft max” function
Feature-Based Linear Classifiers

• Given this model form, we will choose parameters \( \{ \lambda_i \} \) that maximize the conditional likelihood of the data according to this model.

• Parameter learning is omitted and not required for this course, but is often discussed in a machine learning class.
  • E.g. gradient descent for parameter learning
Outline

• Maximum Entropy
• Feedforward Neural Networks
• Recurrent Neural Networks
Neural Network Learning

• Learning approach based on modeling adaptation in biological neural systems.

• Perceptron: Initial algorithm for learning simple neural networks (single layer) developed in the 1950’s.

• Backpropagation: More complex algorithm for learning multi-layer neural networks developed in the 1980’s. (not required for this class)
**Topics:** connection weights, bias, activation function

- Neuron pre-activation (or input activation):

  \[ a(x) = b + \sum_i w_i x_i = b + \mathbf{w}^\top \mathbf{x} \]

- Neuron (output) activation

  \[ h(x) = g(a(x)) = g(b + \sum_i w_i x_i) \]

- \( \mathbf{w} \) are the connection weights
- \( b \) is the neuron bias
- \( g(\cdot) \) is called the activation function
ARTIFICIAL NEURON

Topics: connection weights, bias, activation function

Range determined by $g(\cdot)$

Bias $b$ only changes the position of the riff

(from Pascal Vincent's slides)
ACTIVATION FUNCTION

Topics: linear activation function

- Performs no input squashing
- Not very interesting...

\[ g(a) = a \]
ACTIVATION FUNCTION

**Topics:** sigmoid activation function

- Squashes the neuron’s pre-activation between 0 and 1
- Always positive
- Bounded
- Strictly increasing

\[ g(a) = \text{sigm}(a) = \frac{1}{1 + \exp(-a)} \]
ACTIVATION FUNCTION

**Topics:** hyperbolic tangent ("tanh") activation function

- Squashes the neuron's pre-activation between -1 and 1
- Can be positive or negative
- Bounded
- Strictly increasing

\[ g(a) = \tanh(a) = \frac{\exp(a) - \exp(-a)}{\exp(a) + \exp(-a)} = \frac{\exp(2a) - 1}{\exp(2a) + 1} \]
**ACTIVATION FUNCTION**

**Topics:** rectified linear activation function

- Bounded below by 0 (always non-negative)
- Not upper bounded
- Strictly increasing
- Tends to give neurons with sparse activities

\[ g(a) = \text{reclin}(a) = \max(0, a) \]
class Neuron(object):
    
    # ...

def forward(inputs):
    
    """ assume inputs and weights are 1-D numpy arrays and bias is a number """
    cell_body_sum = np.sum(inputs * self.weights) + self.bias
    firing_rate = 1.0 / (1.0 + math.exp(-cell_body_sum))  # sigmoid activation function
    return firing_rate
Linear Separator

- Since one-layer neuron (aka perceptron) uses linear threshold function, it is searching for a linear separator that discriminates the classes.
**ARTIFICIAL NEURON**

**Topics:** capacity of single neuron

- Can solve linearly separable problems
**ARTIFICIAL NEURON**

**Topics:** capacity of single neuron

- Can’t solve non linearly separable problems...

- ... unless the input is transformed in a better representation
Topics: single hidden layer neural network

- Hidden layer pre-activation:
  \[ a(x) = b^{(1)} + W^{(1)}x \]
  \[ (a(x)_i = b^{(1)}_i + \sum_j W^{(1)}_{i,j} x_j) \]

- Hidden layer activation:
  \[ h(x) = g(a(x)) \]

- Output layer activation:
  \[ f(x) = o \left( b^{(2)} + w^{(2)\top} h^{(1)}x \right) \]

output activation function
Topics: softmax activation function

- For multi-class classification:
  - we need multiple outputs (1 output per class)
  - we would like to estimate the conditional probability $p(y = c|\mathbf{x})$

- We use the softmax activation function at the output:

$$\mathbf{o}(\mathbf{a}) = \text{softmax}(\mathbf{a}) = \left[ \frac{\exp(a_1)}{\sum_c \exp(a_c)} \cdots \frac{\exp(a_C)}{\sum_c \exp(a_c)} \right]^T$$

  - strictly positive
  - sums to one

- Predicted class is the one with highest estimated probability
Topics: multilayer neural network

- Could have $L$ hidden layers:
  - layer pre-activation for $k>0$: $h^{(k)}(x) = a^{(k)}(x) = b^{(k)} + W^{(k)}h^{(k-1)}(x)$
  - hidden layer activation ($k$ from 1 to $L$): $h^{(k)}(x) = g(a^{(k)}(x))$
  - output layer activation ($k=L+1$): $h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x)$
# forward-pass of a 3-layer neural network:

\[
f = \text{lambda } x: \frac{1.0}{1.0 + \text{np.exp}(-x)}\]  # activation function (use sigmoid)

\[
x = \text{np.random.randn}(3, 1)\]  # random input vector of three numbers (3x1)

\[
h1 = f(\text{np.dot}(W1, x) + b1)\]  # calculate first hidden layer activations (4x1)

\[
h2 = f(\text{np.dot}(W2, h1) + b2)\]  # calculate second hidden layer activations (4x1)

\[
\text{out} = \text{np.dot}(W3, h2) + b3\]  # output neuron (1x1)
CAPACITY OF NEURAL NETWORK

Topics: single hidden layer neural network
CAPACITY OF NEURAL NETWORK

Topics: single hidden layer neural network

(from Pascal Vincent's slides)
CAPACITY OF NEURAL NETWORK

Topics: single hidden layer neural network
CAPACITY OF NEURAL NETWORK

**Topics:** universal approximation

- Universal approximation theorem (Hornik, 1991):
  
  “a single hidden layer neural network with a linear output unit can approximate any continuous function arbitrarily well, given enough hidden units”

- The result applies for sigmoid, tanh and many other hidden layer activation functions

- This is a good result, but it doesn’t mean there is a learning algorithm that can find the necessary parameter values!
How to train a neural network? (Not covered in this course, only for reference)

**Topics:** multilayer neural network

- Could have \( L \) hidden layers:
  - layer input activation for \( k \geq 0 \) \( (h^{(0)}(x) = x) \)
  - \( a^{(k)}(x) = b^{(k)} + W^{(k)} h^{(k-1)}(x) \)

- hidden layer activation \((k\) from 1 to \(L)):
  - \( h^{(k)}(x) = g(a^{(k)}(x)) \)

- output layer activation \((k=L+1)):
  - \( h^{(L+1)}(x) = o(a^{(L+1)}(x)) = f(x) \)
Empirical Risk Minimization

Topics: empirical risk minimization, regularization

• Empirical risk minimization
  ‣ framework to design learning algorithms

\[
\arg \min_{\theta} \frac{1}{T} \sum_{t} l(f(x^{(t)}; \theta), y^{(t)}) + \lambda \Omega(\theta)
\]

• \( l(f(x^{(t)}; \theta), y^{(t)}) \) is a loss function
• \( \Omega(\theta) \) is a regularizer (penalizes certain values of \( \theta \))

• Learning is cast as optimization
  ‣ ideally, we'd optimize classification error, but it's not smooth
  ‣ loss function is a surrogate for what we truly should optimize (e.g. upper bound)
**LOSS FUNCTION**

**Topics:** loss function for classification

- Neural network estimates $f(x)_c = p(y = c|x)$
  - we could maximize the probabilities of $y^{(t)}$ given $x^{(t)}$ in the training set

- To frame as minimization, we minimize the negative log-likelihood

\[
l(f(x), y) = - \sum_c 1(y=c) \log f(x)_c = - \log f(x)_y
\]

- we take the log to simplify for numerical stability and math simplicity
- sometimes referred to as cross-entropy
REGULARIZATION

**Topics:** L2 regularization

\[ \Omega(\theta) = \sum_k \sum_i \sum_j \left( W_{i,j}^{(k)} \right)^2 = \sum_k \| W^{(k)} \|_F^2 \]
Empirical Risk Minimization

**Topics:** empirical risk minimization, regularization

- **Empirical risk minimization**
  - framework to design learning algorithms

\[
\arg \min_{\theta} \frac{1}{T} \sum_{t} l(f(x^{(t)}; \theta), y^{(t)}) + \lambda \Omega(\theta)
\]

- \( l(f(x^{(t)}; \theta), y^{(t)}) \) is a loss function
- \( \Omega(\theta) \) is a regularizer (penalizes certain values of \( \theta \))

- **Learning is cast as optimization**
  - ideally, we'd optimize classification error, but it's not smooth
  - loss function is a surrogate for what we truly should optimize (e.g. upper bound)
[http://cs231n.github.io/neural-networks-1/]
INITIALIZATION

**Topics:** initialization

- For biases
  - initialize all to 0

- For weights
  - Can’t initialize weights to 0 with tanh activation
    - we can show that all gradients would then be 0 (saddle point)
  - Can’t initialize all weights to the same value
    - we can show that all hidden units in a layer will always behave the same
    - need to break symmetry
  - Recipe: sample $W_{i,j}^{(k)}$ from $U[-b,b]$ where $b = \frac{\sqrt{6}}{\sqrt{H_k + H_{k-1}}}$
    - the idea is to sample around 0 but break symmetry
    - other values of $b$ could work well (not an exact science) (see Glorot & Bengio, 2010)
Model Learning

• Backpropagation (BP) algorithm (not required for this course)
• Further reading on BP:
  • [https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd](https://towardsdatascience.com/understanding-backpropagation-algorithm-7bb3aa2f95fd)
Outline

• Maximum Entropy
• Feedforward Neural Networks
• Recurrent Neural Networks
Long Distance Dependencies

• It is very difficult to train NNs to retain information over many time steps.
• This makes it very difficult to handle long-distance dependencies, such as subject-verb agreement.
• E.g. Jane walked into the room. John walked in too. It was late in the day. Jane said hi to _?_

```
X_t
\downarrow
A
```

```
h_t
\downarrow
A
```

```
h_0 \quad h_1 \quad h_2 \quad ... \quad h_t
\downarrow \quad \downarrow \quad \downarrow \quad \downarrow
X_0 \quad X_1 \quad X_2 \quad ... \quad X_t
```
Recurrent Neural Networks

Feed-forward NN
\[
\begin{align*}
  h &= g(Vx + c) \\
  \hat{y} &= Wh + b
\end{align*}
\]

Recurrent NN
\[
\begin{align*}
  h_t &= g(Vx_t + Uh_{t-1} + c) \\
  \hat{y}_t &= Wh_t + b
\end{align*}
\]
Long-Short Term Memory Networks (LSTMs)
Sequence to Sequence

- Encoder/Decoder framework maps one sequence to a "deep vector" then another LSTM maps this vector to an output sequence.

This is my cat

C'est mon chat
Summary of LSTM Application Architectures

- Image Captioning
- Video Activity Recognition
- Text Classification
- Video Captioning
- Machine Translation
- POS Tagging
Successful Applications of LSTMs

• Speech recognition: Language and acoustic modeling
• Sequence labeling
  • POS Tagging
  • NER
  • Phrase Chunking
• Neural syntactic and semantic parsing
• Image captioning
• Sequence to Sequence
  • Machine Translation *(Sustkever, Vinyals, & Le, 2014)*
  • Summarization
  • Video Captioning (input sequence of CNN frame outputs)