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

- Maps entities to categories
- Uses features to predict categories

Example features

- $f_{LOCATION}(d) = \left[ c \text{ = LOCATION } \land \text{ w is Capitalized(w)} \right]$
- $f_{LOCATION}(d) = \left[ c \text{ = LOCATION } \land \text{ hasAccentedLatinChar(w)} \right]$
- $f_{DRUG}(d) = \left[ c \text{ = DRUG } \land \text{ ends(w, "c"�)} \right]$

- 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

- $f_1(c, d) \equiv [c = \text{LOCATION}, w_1 = \text{"in"}, \text{isCapitalized}(w)] \rightarrow \text{weight 1.8}$
- $f_2(c, d) \equiv [c = \text{LOCATION}, \text{hasAccentedLatinChar}(w)] \rightarrow \text{weight -0.6}$
- $f_3(c, d) \equiv [c = \text{DRUG}, \text{ends}(w, \text{"c")}] \rightarrow \text{weight 0.3}$

- 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

$$f(c, d) \equiv [\Phi(d) \land c = c_j] \quad \text{[Value is 0 or 1]}$$

Feature-Based Models

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

Feature-Based Linear Classifiers

- Linear classifiers at classification time:
  - Linear function from feature sets $\{d\}$ to classes $\{c\}$.
  - Assign a weight $\lambda_i$ to each feature $f_i$.
  - We consider each class for sample $d$.
  - For a pair $\langle c, d \rangle$, features vote with their weights:
    - $\text{vote}(c) = \sum \lambda_i f_i(c, d)$
  - Choose the class $c$ which maximizes $\sum \lambda_i f_i(c, d)$

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

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

  - Makes votes positive
  - Normalizes votes

Feature-Based Linear Classifiers

- $f_1(c, d) \equiv [c = \text{LOCATION}, w_1 = \text{"in"}, \text{isCapitalized}(w)] \rightarrow \text{weight 1.8}$
- $f_2(c, d) \equiv [c = \text{LOCATION}, \text{hasAccentedLatinChar}(w)] \rightarrow \text{weight -0.6}$
- $f_3(c, d) \equiv [c = \text{DRUG}, \text{ends}(w, \text{"c")}] \rightarrow \text{weight 0.3}$

Feature-Based Models

- Data: Stocks hit a yearly low …
  - Label: BUSINESS
    - Features: {..., stocks, hit, a, yearly, low, ...}
  - Text Classification

- Data: Money to restructure bank: MONEY debt.
  - Label: MONEY
    - Features: {..., w-1 = restructure, w+1 = debt, L=12, ...}
  - Word Sense Disambiguation

- Data: DT NN NN … The previous fall …
  - Label: NN
    - Features: {..., w-1 = fall, w-1 = previous}
  - POS Tagging

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

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

  - Makes votes positive
  - Normalizes votes
Maximum Entropy:

- Recurrent Neural Networks
- Feedforward Neural Networks
- Maximum Entropy

Feature-Based Linear Classifiers

- Given this model form, we will choose parameters ($\lambda$) 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
- We construct not only classifications, but probability distributions over classifications.
- There are other (good!) ways of discriminating classes — SVMs, boosting, even perceptrons — but these methods are not as trivial to interpret as distributions over classes.

Other MaxEnt Classifier Examples

- You can use a MaxEnt classifier whenever you want to assign data points to one of a number of classes:
  - Sentence boundary detection (Johnson et al. 1999, etc.)
  - Is a period end of sentence or abbreviation?
  - Sentiment analysis ( Pang and Lee 2002)
  - Word unigrams, bigrams, POS counts, …
  - Prepositional phrase attachment (Ratnaparkhi 1997; Johnson et al. 1999, etc.)
  - Attends to verb or noun? Features of head noun, preposition, etc.
  - Parsing decisions (Kiperwasser 2018; Johnson et al. 1999, etc.)

Outline

- Maximum Entropy
- Feedforward Neural Networks
- Recurrent Neural Networks

Neural Network Learning

- Learning approach based on modelling 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)
**ARTIFICIAL NEURON**

- **Topics:** connection weights, bias, activation function
  - Neuron pre-activation (or input activation):
    \[ a(x) = b + \sum w_v x_v = b + w^T x \]
  - Neuron (output) activation:
    \[ A(x) = g(a(x)) = g(b + \sum w_v x_v) \]
  - \( w \) are the connection weights
  - \( b \) is the neuron bias
  - \( g(.) \) is called the activation function

**ACTIVATION FUNCTION**

- **Topics:** linear activation function
  - Replaces no input squashing
  - Not very interesting...
  - Graph: \( g(a) = a \)

- **Topics:** sigmoid activation function
  - Squashes the neuron’s pre-activation between 0 and 1
    - Always positive
    - Bounded
    - Strictly increasing
  - Graph: \( g(a) = \text{sigmoid}(a) = \frac{1}{1 + e^{-a}} \)

- **Topics:** hyperbolic tangent (“tanh”) activation function
  - Squashes the neuron’s pre-activation between -1 and 1
  - Can be positive or negative
  - Bounded
  - Strictly increasing
  - Graph: \( g(a) = \tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} = \frac{e^{2a} - 1}{e^{2a} + 1} \)

- **Topics:** rectified linear activation function
  - Bounded below by 0 (always non-negative)
  - Not upper bounded
  - Strictly increasing
  - Tends to give neurons with sparse activities
  - Graph: \( g(a) = \text{rectlin}(a) = \max(0,a) \)
Since one-layer neuron (aka perceptron) uses linear threshold function, it is searching for a linear separator that discriminates the classes.
Topics: Multilayer neural network
- Can have L hidden layers
  1. Layer pre-activation for i-th layer
     \( a^{(i)}(x) = h^{(i)}(x) = W^{(i)}h^{(i-1)}(x) \)
  2. Hidden layer activation (2 from 1 to L)
     \( h^{(i)}(x) = g^{(i)}(a^{(i)}(x)) \)
  3. Output layer activation (0-1, L)
     \( h^{(L+1)}(x) = w^{(L+1)}(h^{(L)}(x)) = f(x) \)

\# Example of a 3-layer neural network:
\[ f = \text{tanh}(x) \times 0.1 + \text{tanh}(x) \]  
\[ x = \text{np.random.randn}(1) \]  
\[ h_1 = \text{np.dot}(X, W_1) + b_1 \]  
\[ h_2 = \text{np.dot}(h_1, W_2) + b_2 \]  
\[ \text{output} = \text{np.dot}(h_2, W_3) + b_3 \]  

Topics: Capacity of neural network
- Single hidden layer neural network
  - The result applies for sigmoid, tanh, and many other hidden layer activation functions
  - Universal approximation theorem (Halkin, 1987)
  - A single hidden layer neural network with a linear output layer can approximate any continuous function adequately well given enough hidden units.
  - 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?

- Could have \( L \) hidden layers:
  - layer input activation for \( l \) \( a_l(x) = \sigma^{(l-1)}(x) \)
  - hidden layer activation \( l \) to \( l+1 \) \( b_l(x) = g^{(l)}(a_l(x)) \)
  - output layer activation \( l = 0 \) \( \hat{y}(x) = f(x) \)

**Empirical Risk Minimization**

Topics: empirical risk minimization, regularization

- Empirical risk minimization
  - framework to design learning algorithms
  - \( \hat{\theta} = \arg \min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (f(x_i; \theta), y_i) + \lambda \Omega(\theta) \)
  - \( f(x; \theta), y_i \) is a loss function
  - \( \Omega(\theta) \) is a regularizer (penalizes certain values of \( \theta \))
  - Learning is cast as optimization
    - ideally, we would minimize 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 \( f(x) = \phi^L(\theta) \)
  - we could minimize the probability of \( \tilde{y} \neq y \) in the training set
- To frame as minimization, we minimize the negative log likelihood
  \( L = -\sum \log f(x_i) \)

**Regularization**

Topics: L2 regularization

\[ \Omega(\theta) = \sum_{l=1}^{L} \sum_{j=1}^{L} (W_{l,j}^2) + \sum |W_{l,j}|. \]
Empirical Risk Minimization

- **Maximum Entropy**
- **Feedforward Neural Networks**
- **Recurrent Neural Networks**

Model Learning

- Backpropagation (BP) algorithm (not required for this course)
- Further reading on BP:
  - [Understanding Backpropagation Algorithm](https://towardsdatascience.com/understanding-backpropagation-algorithm-75036e4f09fd)
  - [Matt Mazur's Backpropagation Example](https://mattmazur.com/2015/03/17/a-step-by-step-backpropagation-example/)

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 _?

\[
\lambda = 0.001 \quad \lambda = 0.01 \quad \lambda = 0.1
\]
Recurrent Neural Networks

Feed-forward NN
\[ h = g(Vx + c) \]
\[ y = Wh + b \]

Recurrent NN
\[ h_0 = g(Vx_0 + Uh_{-1} + c) \]
\[ y_1 = Wh_1 + b \]

Long-Short Term Memory Networks (LSTMs)

Use gates to control the information to be added from the input, forgotten from the previous memories, and outputted.
\[ h_t = c_t \cdot c_{t-1} + f_t \cdot c_t \]

Another Visualization

Capable of modeling long-distant dependencies between states.

Figure: Christopher Olah

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
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 (Sutskever, Vinyals, & Le, 2014)
  - Summarization
- Video Captioning (input sequence of CNN frame outputs)