Natural Language Processing

Sargur N. Srihari
srihari@cedar.buffalo.edu

This is part of lecture slides on Deep Learning:
http://www.cedar.buffalo.edu/~srihari/CSE676

Topics

1. N-gram Models
2. Neural Language Models
3. High-dimensional Outputs
4. Combining Neural Language Models with n-grams
5. Neural Machine Translation
6. Other Applications

What is NLP?

- Processing by computer of human languages
  - E.g., English or French
  - Computers process unambiguous languages
    - e.g., formal languages
  - Naturally occurring languages are ambiguous
    - defy formal description
- NLP Application: Machine Translation
  - Learner reads sentence in a language and emits a sequence in another
- NLP is based on language models
  - that define distributions over word sequences

Neural Networks for NLP

- Very generic neural network techniques can be successfully applied to NLP
- To scale to large applications, need domain-specific strategies Need techniques for processing sequential data
- NL regarded as word sequences
  - Since no. of possible sequences is large, language models must operate on an extremely high-dimensional and sparse space
    - Several strategies developed to make models of such a space efficient

N-Grams

- An n-gram is a sequence of tokens, e.g., words
- n-gram models define the conditional probability of the n^{th} token given the previous n-1 tokens
- Products of conditional distributions define probability distributions of longer sequences

$$P(x_1,...,x_\tau) = P(x_1,...,x_{n-1}) \prod_{t=n}^{\tau} P(x_t \mid \underbrace{x_{t-n+1},...,x_{\tau-1}}_{\text{Sequence of length } n})$$

  - Comes from chain rule of probability
  - Distribution of $P(x_1,..,x_{n-1})$ may be defined by a different model with a smaller value of n
- Models based on n-grams have been core building block of NLP

Training N-Gram Models

- Count how many times each possible n-gram occurs in the training set
- For small values of n, we have

  $n=1$: unigram

  $n=2$: bigram

  $n=3$: trigram

- Usually train both an n-gram model and an n-1 gram model making it easy to compute

$$P(x_t \mid x_{t-n+1},..,x_{\tau-1}) = \frac{P_n(x_{t-n+1},..,x_t)}{P_{n-1}(x_{t-n+1},..,x_{t-1})}$$

Limitation of ML for n-gram models

- P_n estimated from training samples is very likely to be zero in many cases even though the tuple $x_{t-n+1},..,x_t$ may appear in test set
  - When P_{n-1} is zero the ratio is undefined
  - When P_{n-1} is non-zero but P_n is zero the log-likelihood is $-\infty$
- n-gram methods employ smoothing
  - Shift probability mass of observed tuples to unobserved similar ones

Example of Trigram Model

- To compute probability of the sentence "THE DOG RAN AWAY"

  P(THE DOG RAN AWAY) =

  P_3(THE DOG RAN)P_3(DOG RAN AWAY)/P_2(DOG RAN)
- x

Disadvantage of n-gram models

- Vulnerable to curse of dimensionality
- There are $|V|^n$ possible n-grams and $|V|$ is large
- Even with a massive training set most n-grams will not occur
- Any two words are at same distance from each other

Class-based language models

- Introduce notion of word categories
  - Share statistics of words in same categories
- Idea is to use a clustering algorithm to partition words into clusters based on their co-occurrence frequencies with other words
- Model can then use word-class IDs rather than individual word-IDs to represent context