A Neural Probabilistic Language Model

Paper Presentation (Y Bengio, et. al. 2003)

Zeming Lin

Department of Computer Science at University of Virginia

March 19 2015

Table of Contents

Review of Language Models

- Predict $P(w_1^T) = P(w_1, w_2, w_3, \ldots, w_T)$
  - As a conditional probability: $P(w_1^T) = \prod_{i=1}^{T} P(w_t | w_1^{t-1})$

Review of Language Models

- Predict $P(w_1^T) = P(w_1, w_2, w_3, \ldots, w_T)$
  - As a conditional probability: $P(w_1^T) = \prod_{i=1}^{T} P(w_t | w_1^{t-1})$
  - Too many conditional probabilities! Simplify using n-gram model:
  - $P(w_t | w_1^T) \approx P(w_t | w_{T-n+1}^T)$

Problems

- We want to optimize on large n, but vocabulary size $V > 100000$.

Problems

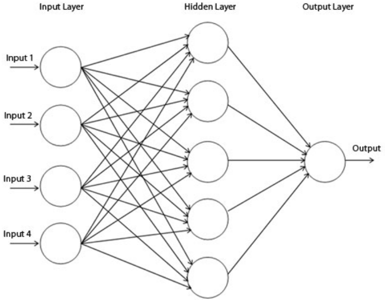
- We want to optimize on large n, but vocabulary size $V > 100000$.
- State of the art (as of 2003) was to use 2 or 3-gram models.
  - 10 gram models has $100000^{10} - 1 = 10^{50} - 1$ parameters

Problems

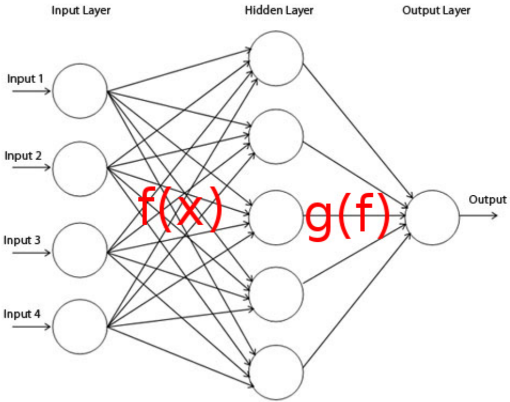
- We want to optimize on large n, but vocabulary size $V > 100000$.
- State of the art (as of 2003) was to use 2 or 3-gram models.
  - 10 gram models has $100000^{10} - 1 = 10^{50} - 1$ parameters
- Does not take in account of similarity between words
  - *A cat is walking in the bedroom*
  - *The dog was running in a room*

Table of Contents

Neural Networks

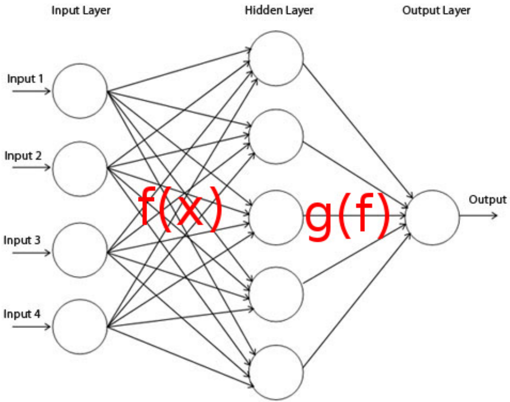


Neural Networks



$f : \mathbb{R}^4 \to \mathbb{R}^5$; Example: $F_1(x) = f(\mathbf{x}) = tanh(\mathbf{A}\mathbf{x})$, $\mathbf{A} \in \mathbb{R}^{5\times 4}$
$g : \mathbb{R}^5 \to \mathbb{R}$; Example: $F_2(x) = g(\mathbf{x}) = tanh(\mathbf{B}\mathbf{x})$, $\mathbf{B} \in \mathbb{R}^{1\times 5}$

Neural Networks



$f : \mathbb{R}^4 \to \mathbb{R}^5$; Example: $F_1(x) = f(\mathbf{x}) = tanh(\mathbf{A}\mathbf{x})$, $\mathbf{A} \in \mathbb{R}^{5\times 4}$
$g : \mathbb{R}^5 \to \mathbb{R}$; Example: $F_2(x) = g(\mathbf{x}) = tanh(\mathbf{B}\mathbf{x})$, $\mathbf{B} \in \mathbb{R}^{1\times 5}$
A and **B** are parameters! Our network is just $F(\mathbf{x}) = F_2(F_1(\mathbf{x}))$

Table of Contents

Neural Network

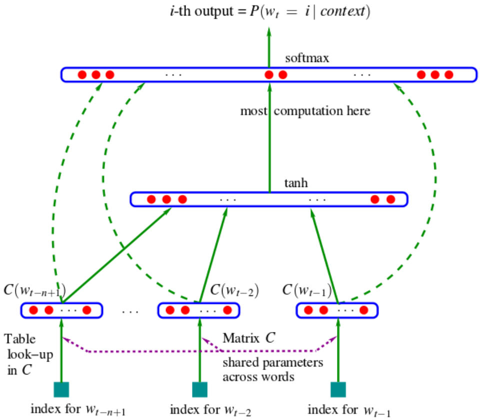


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

Model

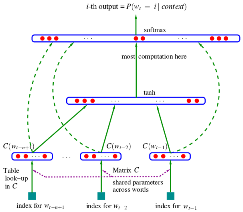


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

$$\hat{P}(w_t = i | w_1^{t-1}) = f(i, g(w_{t-n+1}^{t-1}))$$

Model

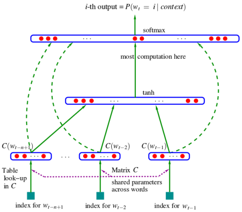


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

$g(w_1^t) = (C(w_1), C(w_2), \ldots C(w_T))$

$C(w_i)$ gives the w_i-th row of $|V| \times m$ matrix

"Lookup table"

Model

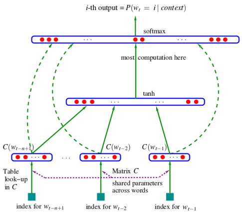


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

$$y(x) = b + Wx + U \tanh(d + Hx)$$
$$s(x) = \frac{e^x}{\sum_i (e^x)_i}$$
$$f(i, x) = s(y(x))_i$$

H is a $h \times (n-1)m$ matrix
d is a h length vector
U is a $|V| \times h$ matrix
W is a $|V| \times (n-1)m$ matrix
b is a $|V|$ length vector

Model

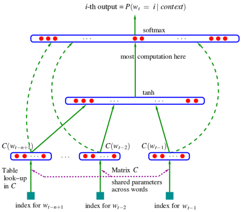


Figure 1: Neural architecture: $f(i, w_{t-1}, \cdots, w_{t-n+1}) = g(i, C(w_{t-1}), \cdots, C(w_{t-n+1}))$ where g is the neural network and $C(i)$ is the i-th word feature vector.

$\hat{P}(w_t = i | w_1^{t-1}) = s(y(C(w_{t-n+1}^{t-1})))_i$

Parameters: $\theta = (b, d, W, U, H, C)$

$O(|V|(nm + h))$ parameters

Linear in n-gram size and vocab size!

Table of Contents

Training

$$\theta \leftarrow \theta + \epsilon \frac{\partial \hat{P}}{\partial \theta}$$

Training

$$\theta \leftarrow \theta + \epsilon \frac{\partial \hat{P}}{\partial \theta}$$

- $\frac{\partial \hat{P}}{\partial \theta}$ can be found analytically using the chain rule
- ϵ is the learning rate

Data Parallelism

- Assume shared memory processor, communication costs are low
- Split data up and have each CPU update the parameters

Data Parallelism

- Assume shared memory processor, communication costs are low
- Split data up and have each CPU update the parameters
  - trick: asynchronously update parameters
  - May lose parameters, but occasional noise did not impact performance

Parameter Parallelism

- Assume computer cluster, communication overhead is large
- Limiting factor during computation of $f(i,x) = s(y(x))_i = s(b + Wx + U\tanh(d + Hx))_i$

Parameter Parallelism

- Assume computer cluster, communication overhead is large
- Limiting factor during computation of
  $f(i,x) = s(y(x))_i = s(b + Wx + U \tanh(d + Hx))_i$
  - Every processor computes $g(x)$.
  - i-th processor computes i-th block of $y(x)$
  - Update central server with sum, receive sum across $y(x)$.

Parameter Parallelism

- Assume computer cluster, communication overhead is large
- Limiting factor during computation of $f(i,x) = s(y(x))_i = s(b + Wx + U\tanh(d + Hx))_i$
  - Every processor computes $g(x)$.
  - i-th processor computes i-th block of $y(x)$
  - Update central server with sum, receive sum across $y(x)$.
- 99.7% calculations not repeated
- $\frac{1}{15}$ total time spent during network communications

Table of Contents

Brown Corpus

- 1,181,041 different words from english texts and books
- 47578 different words, including case, and puctuation.
- $|V| = 16383$ after introducing "rare words" symbol
  - no words with frequency less than 4

Brown Corpus

	n	c	h	m	direct	mix	train.	valid.	test.
MLP1	5		50	60	yes	no	182	284	268
MLP2	5		50	60	yes	yes		275	257
MLP3	5		0	60	yes	no	201	327	310
MLP4	5		0	60	yes	yes		286	272
MLP5	5		50	30	yes	no	209	296	279
MLP6	5		50	30	yes	yes		273	259
MLP7	3		50	30	yes	no	210	309	293
MLP8	3		50	30	yes	yes		284	270
MLP9	5		100	30	no	no	175	280	276
MLP10	5		100	30	no	yes		265	**252**
Del. Int.	3						31	352	336
Kneser-Ney back-off	3							334	323
Kneser-Ney back-off	4							332	321
Kneser-Ney back-off	5							332	321
class-based back-off	3	150						348	334
class-based back-off	3	200						354	340
class-based back-off	3	500						326	**312**
class-based back-off	3	1000						335	319
class-based back-off	3	2000						343	326
class-based back-off	4	500						327	312
class-based back-off	5	500						327	312

Table 1: Comparative results on the Brown corpus. The deleted interpolation trigram has a test perplexity that is 33% above that of the neural network with the lowest validation perplexity. The difference is 24% in the case of the best n-gram (a class-based model with 500 word classes). n : order of the model. c : number of word classes in class-based n-grams. h : number of hidden units. m : number of word features for MLPs, number of classes for class-based n-grams. *direct*: whether there are direct connections from word features to outputs. *mix*: whether the output probabilities of the neural network are mixed with the output of the trigram (with a weight of 0.5 on each). The last three columns give perplexity on the training, validation and test sets.

AP News

- Associated press news from 1995-1996.
- ~14 million words
- 148,721 different words
- $|V| = 17964$
  - No words with frequency < 4.
  - Preprocessed to lower case
  - Preprocessed numbers, rare words, and proper nouns to special symbols

AP News

	n	h	m	direct	mix	train.	valid.	test.
MLP10	6	60	100	yes	yes		104	**109**
Del. Int.	3						126	132
Back-off KN	3						121	127
Back-off KN	4						113	119
Back-off KN	5						112	**117**

Table 2: Comparative results on the AP News corpus. See the previous table for the column labels.