CS371N: Natural Language Processing

Lecture 7: Word Embeddings

Greg Durrett

TEXAS
The University of Texas at Austin

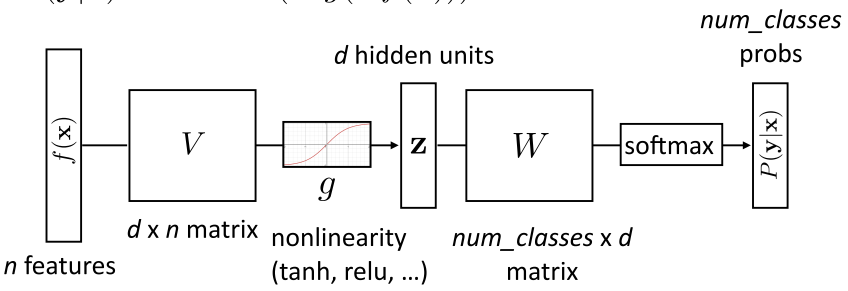
Announcements

- Fairness response due today
- A2 due in 9 days

Recap

Recap: Neural Networks for Classification

$$P(\mathbf{y}|\mathbf{x}) = \text{softmax}(W g(V f(\mathbf{x})))$$



$f(\mathbf{x})$

n features

V

d x *n* matrix

g

nonlinearity
(tanh, relu, …)

d hidden units

$\mathbf{z}$

W

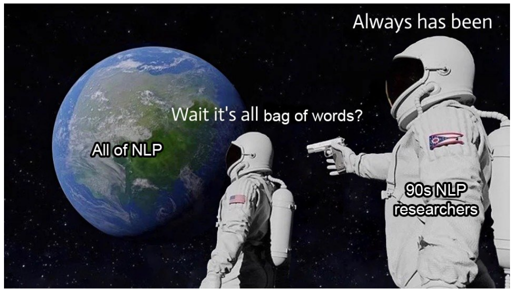
num_classes x *d*
matrix

softmax

num_classes
probs

$P(\mathbf{y}|\mathbf{x})$

DANs



Credit: Stephen Roller

Word Embeddings

- Currently we think of words as "one-hot" vectors

  the = v_{the} = [1, 0, 0, 0, 0, 0, …]

  good = v_{good} = [0, 0, 0, 1, 0, 0, …]

  great = v_{great} = [0, 0, 0, 0, 0, 1, …]

- *good* and *great* seem as dissimilar as *good* and *the*

  the movie was great = $v_{the} + v_{movie} + v_{was} + v_{great}$

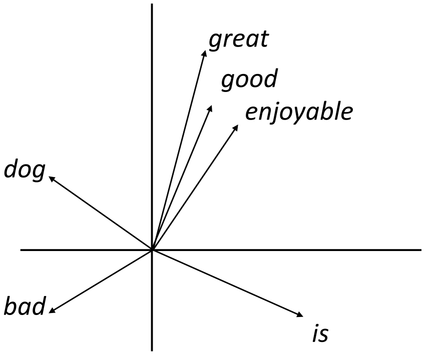
- Neural networks are built to learn sophisticated nonlinear functions of continuous inputs; our inputs are discrete and high-dimensional

Word Embeddings

- Want a vector space where similar words have similar embeddings

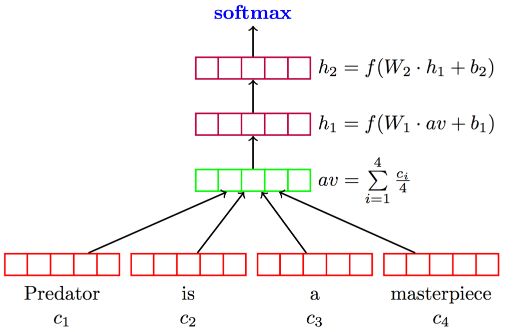
  great ≈ *good*

- This lecture: come up with a way to produce these embeddings
- For each word, want "medium" dimensional vector (50-300 dims) representing it



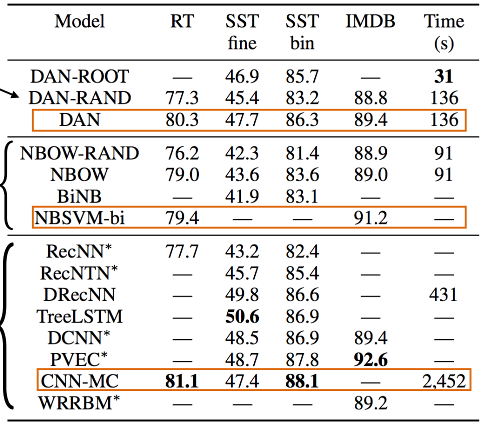
Deep Averaging Networks

- Deep Averaging Networks: feedforward neural network on average of word embeddings from input


softmax
$h_2 = f(W_2 \cdot h_1 + b_2)$
$h_1 = f(W_1 \cdot av + b_1)$
$av = \sum_{i=1}^{4} \frac{c_i}{4}$
Predator c_1
is c_2
a c_3
masterpiece c_4

Iyyer et al. (2015)

Sentiment Analysis



Model	RT	SST fine	SST bin	IMDB	Time (s)
DAN-ROOT	—	46.9	85.7	—	**31**
DAN-RAND	77.3	45.4	83.2	88.8	136
DAN	80.3	47.7	86.3	89.4	136
NBOW-RAND	76.2	42.3	81.4	88.9	91
NBOW	79.0	43.6	83.6	89.0	91
BiNB	—	41.9	83.1	—	—
NBSVM-bi	79.4	—	—	91.2	—
RecNN*	77.7	43.2	82.4	—	—
RecNTN*	—	45.7	85.4	—	—
DRecNN	—	49.8	86.6	—	431
TreeLSTM	—	**50.6**	86.9	—	—
DCNN*	—	48.5	86.9	89.4	—
PVEC*	—	48.7	87.8	**92.6**	—
CNN-MC	**81.1**	47.4	**88.1**	—	2,452
WRRBM*	—	—	—	89.2	—

No pretrained embeddings

Bag-of-words

Tree-structured neural networks

Iyyer et al. (2015)

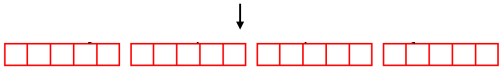
Wang and Manning (2012)

Kim (2014)

Word Embeddings in PyTorch

- torch.nn.Embedding: maps vector of indices to matrix of word vectors

Predator	is	a	masterpiece
1820	24	1	2047



- *n* indices => *n* x *d* matrix of *d*-dimensional word embeddings
- *b* x *n* indices => *b* x *n* x *d* tensor of *d*-dimensional word embeddings

Word Embeddings

Word Embeddings

- J.R. Firth, 1957: “You shall know a word by the company it keeps.”

I watched the movie

I watched the film

The movie inspired me

The film inspired me

I watched the baby

The baby inspired me

There was film on the liquid