word2vec (with a vengeance)

Eduardo Ponce

University of Tennessee, Knoxville
COSC 690 - Evidence Engineering

April 29, 2017

Vector representation of words

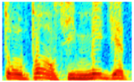
- Image and audio processing work with high-dimensional data

$$\text{encoded as vectors} \begin{cases} \textit{pixel intensities} \\ \textit{power spectra coefficients} \end{cases}$$

- For object and speech recognition, all info is in the raw data
  Traditionally, NLP systems treat words as atomic units

$$\text{arbitrary encodings} \begin{cases} cat \rightarrow \text{“}Id537\text{”} \\ dog \rightarrow \text{“}Id143\text{”} \end{cases}$$

  - No useful info on relationships that may exist between words
  - Representing words as unique, discrete IDs $\rightarrow$ sparsity



Audio Spectrogram

DENSE

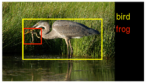
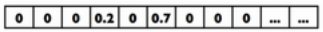


Image pixels

DENSE



Word, context, or document vectors

SPARSE

THE UNIVERSITY of TENNESSEE KNOXVILLE

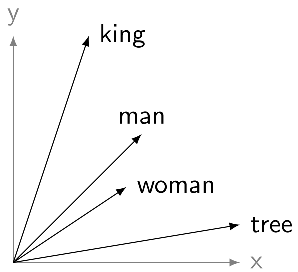
Vector representation of words (a.k.a. word embeddings)

Vector space models represent words in a continuous vector space, where semantically similar words are mapped to nearby points.

Distributional hypothesis:

- words with similar distributions have similar meanings
- words in similar contexts share semantic meaning

1. count-based methods (LSA)
   - compute statistics of neighbors co-occurrences from large text corpus
   - for each word, map statistics to a small and dense vector
2. predictive methods (NPL)
   - predict word from neighbors using learned small and dense vectors
   - embedding vectors are parameters of the model



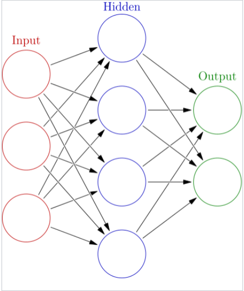
Learning vector representation of words

Language models:

- Feedforward neural network
- Recurrent neural network
- Continuous bag-of-words
- Continuous skip-gram

word2vec

Important architecture in neural network language models
Statistical approach coupled with machine learning



- Continuous bag-of-words (CBOW)
- Skip-gram

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

CBOW based architecture

Log linear classifier with input averaged over past and future word vectors. Predicts a missing word from a given context of word sequence.

Example:

- latent Dirichlet **allocation**

Useful for these cases:

- missing word in sentence or long phrase
- meaningful bigrams $\rightarrow$ state – capital
- effective sentiment orientation

Skip-gram based architecture

Log linear classifier.
Predicts missing context of word sequence from a given word

Example:

- **latent** Dirichlet **allocation**

Vector representation of words (a.k.a. word embeddings)

Consider 3 sentences, vocabulary of 5 distinct words, and window size of 2 words.

S1: w1 w2 w3
S2: w2 w4
S3: w1 w5 w4

w1 w2	w2 w3
w2 w4	
w1 w5	w5 w4

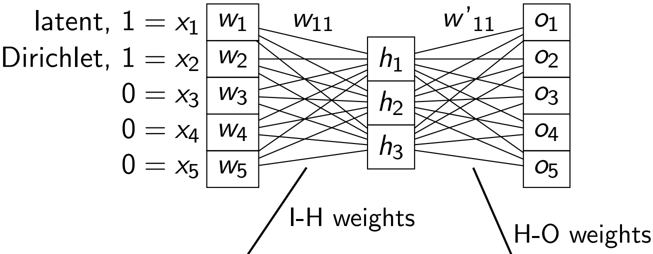
	w1	**w2**	**w3**	**w4**	**w5**
w1	0	1	0	0	1
w2	0	0	1	1	0
w3	0	0	0	0	0
w4	0	0	0	0	0
w5	0	0	0	1	0

- window size = k-words
- number of rows = vocabulary size
- number of columns = vector dimensionality

THE UNIVERSITY of TENNESSEE UT KNOXVILLE

CBOW Example

- latent Dirichlet **allocation**



w_{11}	w_{12}	w_{13}
w_{21}	w_{22}	w_{23}
w_{31}	w_{32}	w_{33}
w_{41}	w_{42}	w_{43}
w_{51}	w_{52}	w_{53}

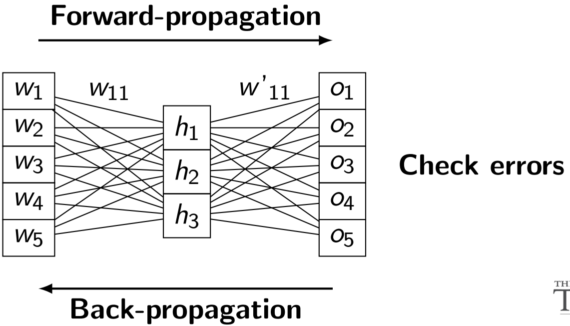
w'_{11}	w'_{12}	w'_{13}	w'_{14}	w'_{15}
w'_{21}	w'_{22}	w'_{23}	w'_{24}	w'_{25}
w'_{31}	w'_{32}	w'_{33}	w'_{34}	w'_{35}

Training CBOW model

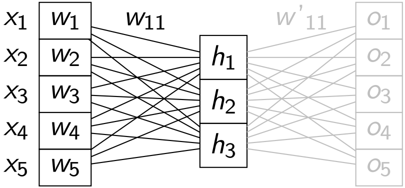
1. Forward-propagation (optimization objective function)
2. Check errors (stochastic gradient descent)
3. Back-propagation

Perform steps 1 – 3 until neuron weights are optimized.

- initially selected from uniform random distribution $[-1, 1]$.



Forward-propagation (input-to-hidden layer)

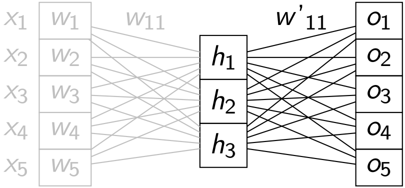


$$\mathbf{h} = \mathbf{W}^{\mathbf{T}}\mathbf{x}$$

$$h_1 = \left(w_{11}x_1 + w_{21}x_2 + \cdots + w_{51}x_5\right)$$
$$h_2 = \left(w_{12}x_1 + w_{22}x_2 + \cdots + w_{52}x_5\right)$$
$$h_3 = \left(w_{13}x_1 + w_{23}x_2 + \cdots + w_{53}x_5\right)$$

$$\begin{array}{|c|c|c|c|c|} \hline w_{11} & w_{21} & w_{31} & w_{41} & w_{51} \\ \hline w_{12} & w_{22} & w_{32} & w_{42} & w_{52} \\ \hline w_{13} & w_{23} & w_{33} & w_{43} & w_{53} \\ \hline \end{array} \quad \begin{array}{|c|} \hline x_1 \\ \hline x_2 \\ \hline x_3 \\ \hline x_4 \\ \hline x_5 \\ \hline \end{array}$$

Forward-propagation (hidden-to-output layer)



$$\mathbf{Net(O)} = \mathbf{u} = \mathbf{W'^{T}h}$$

$$Net(o_1) = u_1 = (w'_{11}h_1 + w'_{21}h_2 + w'_{31}h_3)$$

$$Net(o_2) = u_2 = (w'_{12}h_1 + w'_{22}h_2 + w'_{32}h_3)$$

$$\vdots$$

$$Net(o_5) = u_5 = (w'_{15}h_1 + w'_{25}h_2 + w'_{35}h_3)$$

$$\begin{bmatrix} w'_{11} & w'_{21} & w'_{31} \\ w'_{12} & w'_{22} & w'_{32} \\ w'_{13} & w'_{23} & w'_{33} \\ w'_{14} & w'_{24} & w'_{34} \\ w'_{15} & w'_{25} & w'_{35} \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \end{bmatrix}$$

Forward-propagation (softmax classifier)

$$Out(o_1) = y_1 = \frac{e^{u_1}}{e^{u_1} + e^{u_2} + \cdots + e^{u_5}}$$

$$Out(o_2) = y_2 = \frac{e^{u_2}}{e^{u_1} + e^{u_2} + \cdots + e^{u_5}}$$

$$\vdots$$

$$Out(o_5) = y_5 = \frac{e^{u_5}}{e^{u_1} + e^{u_2} + \cdots + e^{u_5}}$$

Softmax output for word w_j w.r.t. context w_I $\rightarrow$ conditional probability

$$\mathbf{P(w_j|w_I) = y_j = \frac{e^{u_j}}{\sum_{j'=1}^{V} e^{u_{j'}}}}$$

Check errors

Assume:

- w_o = output word
- w_I = context words
- V = size of input context

$$max\ P(w_o|w_I) = max(y_{j^*}) = max(log(y_{j^*}))$$

where j^* = index of output word

Example:

$$E(o_4) = log(P(w_{o_4}|w_I)) = log_e\left(\frac{e^{u_j}}{\sum_{j'=1}^{V} e^{u_{j'}}}\right)$$

$$= u_4 - log_e\left(\sum_{j'=1}^{V} e^{u_{j'}}\right)$$

THE UNIVERSITY of TENNESSEE KNOXVILLE

Check errors

To minimize errors, $E = -log(P(w_o|w_I))$

$$E = log\left(\sum_{j'=1}^{V} e^{u_{j'}}\right) - u_{j^*}$$

Derivate of E w.r.t. u_4,

$$\frac{dE(o_4)}{d(u_4)} = Out(o_4) - \frac{du_4}{du_4} = y_4 - 1$$

$$\frac{dE}{dj} = y_j - t_j = e_j \begin{cases} t_j = 1, \ if \ t_j = t_{j^*} \\ t_j = 0, \ otherwise \end{cases}$$

Backward-propagation (output-to-hidden layer)

Take gradient of E, w.r.t. w'_{11}

$$\frac{dE(o_1)}{dw'_{11}} = \frac{dE(o_1)}{du_1} \times \frac{du_1}{dw'_{11}}$$

$$\frac{dE(o_1)}{du_1} = y_1 - 0 = e_1$$

$$\frac{du_1}{dw'_{11}} = \frac{d(w'_{11}h_1 + w'_{21}h_2 + w'_{31}h_3)}{dw'_{11}} = h_1$$

$$\mathbf{\frac{dE(o_1)}{dw'_{11}} = e_1 \times h_1}$$

Update weight of w'_{11}:

$$(\mathbf{w'_{11}})^{\mathbf{new}} = \mathbf{w'_{11}} - \eta(\mathbf{e_1} \times \mathbf{h_1})$$

$\eta = [0, 1] \rightarrow$ learning rate

Need to update all neurons w'_{ij}

Backward-propagation (hidden-to-input layer)

Take gradient of E, w.r.t. w_{11}

$$\frac{dE}{dw_{11}} = \frac{dE}{dh_1} \times \frac{dh_1}{dw_{11}}$$

$$\frac{dE}{dh_1} = \left(\frac{dE}{du_1} \times \frac{du_1}{dh_1}\right) + \cdots + \left(\frac{dE}{du_5} \times \frac{du_5}{dh_1}\right) = (e_1 w'_{11}) + \cdots + (e_5 w'_{15})$$

$$\frac{dh_1}{dw_{11}} = \frac{d(w_{11}x_1 + w_{21}x_2 + \cdots + w_{31}x_3)}{dw_{11}}$$

$$\mathbf{\frac{dE}{dw_{11}} = x_1 \frac{dE}{dh_1}}$$

Update weight of w_{11}:

$$(\mathbf{w_{11}})^{\mathbf{new}} = \mathbf{w_{11}} - \eta \mathbf{\frac{dE}{dw_{11}}}$$

$\eta = [0, 1] \rightarrow$ learning rate
Need to update all neurons w_{ij}

THE UNIVERSITY of TENNESSEE KNOXVILLE

Training optimizations

Problem: neural network is huge

- two weight matrices (hidden and output layers)
- $10K$ words $\times$ 300 size embedding vectors $= \frac{3\ G\ weights}{matrix}$
- need large amount of training data to tune weights
- training and gradient descent are slow on such NN

Training optimizations

Solutions:

- remove words that occur less than some minimum
- treat common pairs or phrases as single "words" - Boston Globe
- subsampling frequent words to decrease number of training examples

$$P(w_i) = \left(\sqrt{\frac{z(w_i)}{0.001}} + 1\right) \times \frac{0.001}{z(w_i)}$$

w_i is a word, $z(w_i)$ is w_i frequency in corpus, $P(w_i)$ is the probability of keeping w_i

Training optimizations

Solutions:

- modify the optimization objective function
  - negative sampling - Each training sample updates some weights. Negative samples are chosen from a "unigram distribution". The probability for selecting a word is related to its frequency.

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=0}^{n} f(w_j)^{3/4}}$$

  - hierarchical softmax layers - reduce output layer to $log_2|V|$

THE UNIVERSITY of TENNESSEE KNOXVILLE

THE END