Word Embedding using Deep Learning

Natural Language Processing

Topics

- Introduction
- Traditional Approaches
- Gradient Descent Optimization
- Word2Vec
  - Continuous Bag-of-words Model
  - Skip-Gram Model
  - Negative Sampling
- Applications
- Conclusion

Introduction

- Language is defined by its words.
- To do Language Processing, we need to tell a computer what a word means.
- Can we encode a word into a number or a vector such that it makes sense semantically and syntactically?

Traditional Approaches

- One-hot Encoding
- Co-occurrence Matrix

One-hot Encoding

- Represent every word as an $\mathbb{R}^{1 \times |V|}$ vector with all 0s and one 1 at the index of that word in the sorted English language.

  e.g.

$$w^{cat} = [\, 0\ 0\ 0\ \ \ldots\ \ 0\ 1\ 0\ \ \ldots\ \ 0\ 0\ 0\,]$$

- Drawbacks
  - No notion of word similarity.

$$similarity(w^{hotel}, w^{motel}) = 0$$

  - High dimensional and sparse.

Co-occurrence Matrix

We loop over a massive dataset and accumulate word co-occurrence counts in some form of a matrix.

e.g.

I enjoy flying.

I like NLP.

I like deep learning

	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy			0					
deep				0				
learning					0			
NLP						0		
flying							0	
.								0

Co-occurrence Matrix Drawbacks

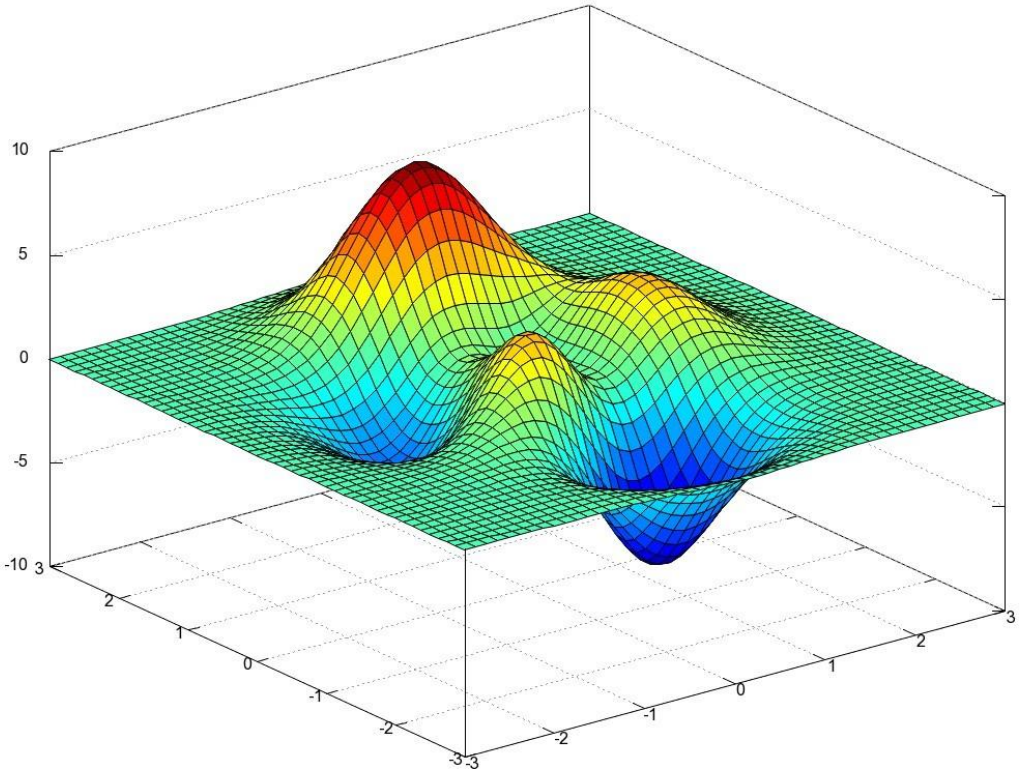
- High Dimensional.
  - Use Singular Value Decomposition to reduce the dimension and size.
- Dimensions change with more adding more words to vocabulary.
- Extremely sparse as many words does not co-occur.
- Performing SVD is very computationally heavy.

Can we overcome these drawbacks?

YES!!!

Gradient Descent Optimization

- An optimization algorithm to find local minimum.



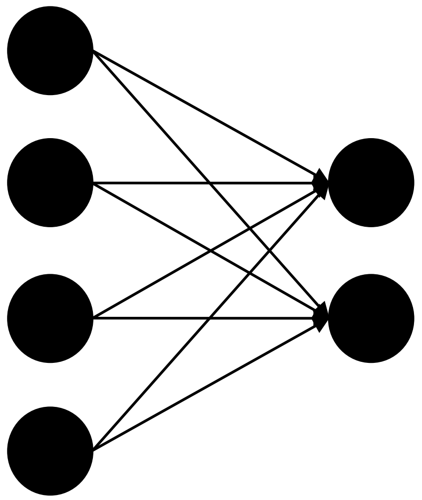
Gradient Descent Optimization contd.

- Objective function is our loss function which we want to minimize.
- Linear Classifier

$$h = W * x$$

$$\hat{y} = softmax(h)$$

$$L(y, \hat{y}) = -\sum_i y_i * \log \hat{y}_i$$

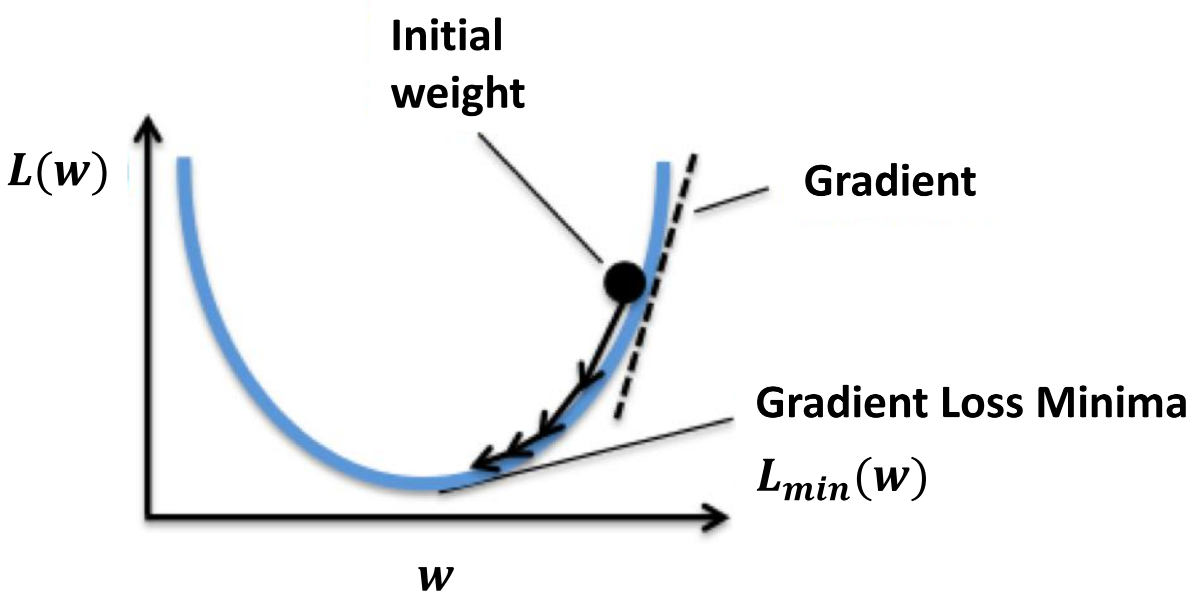


- Update equation

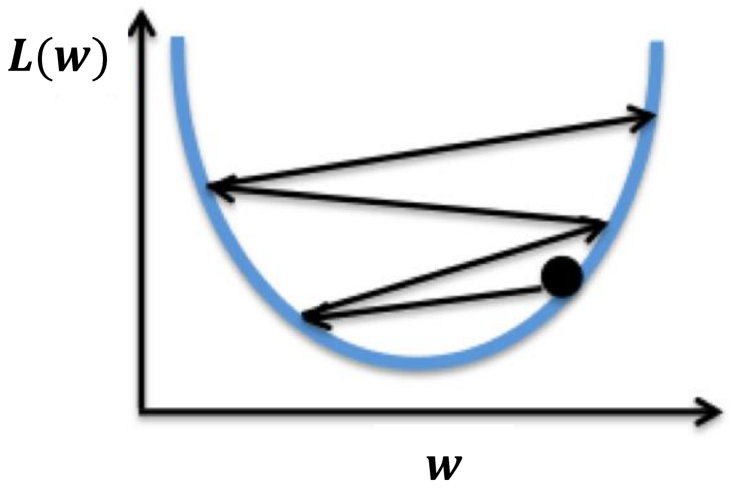
$$W = W - \alpha * \frac{\partial L}{\partial W}$$

Gradient Descent Optimization contd.

$$W = W - \alpha * \frac{\partial L}{\partial W}$$



Small learning rate



Large learning rate

Gradient Descent Optimization contd.

- Forward Propagation

$$h = W * x$$

$$\hat{y} \ = softmax(h)$$

$$L(y, \hat{y}) = -\sum_i y_i * \log \hat{y}_i$$

$$= -\log \hat{y}_c$$

$$= -\log \frac{e^{h_c}}{\sum_i e^{h_i}}$$

$$= -\ h_c \ + \ \log \sum_i e^{h_i}$$

- Back propagation

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial h} * \frac{\partial h}{\partial W}$$

$$\frac{\partial L}{\partial h} = \hat{y} - \ y$$

$$\frac{\partial h}{\partial W} = x$$

$$\frac{\partial L}{\partial W} = (\hat{y} - \ y) * x$$

For more information - https://cs231n.github.io - Module 1

Word2Vec

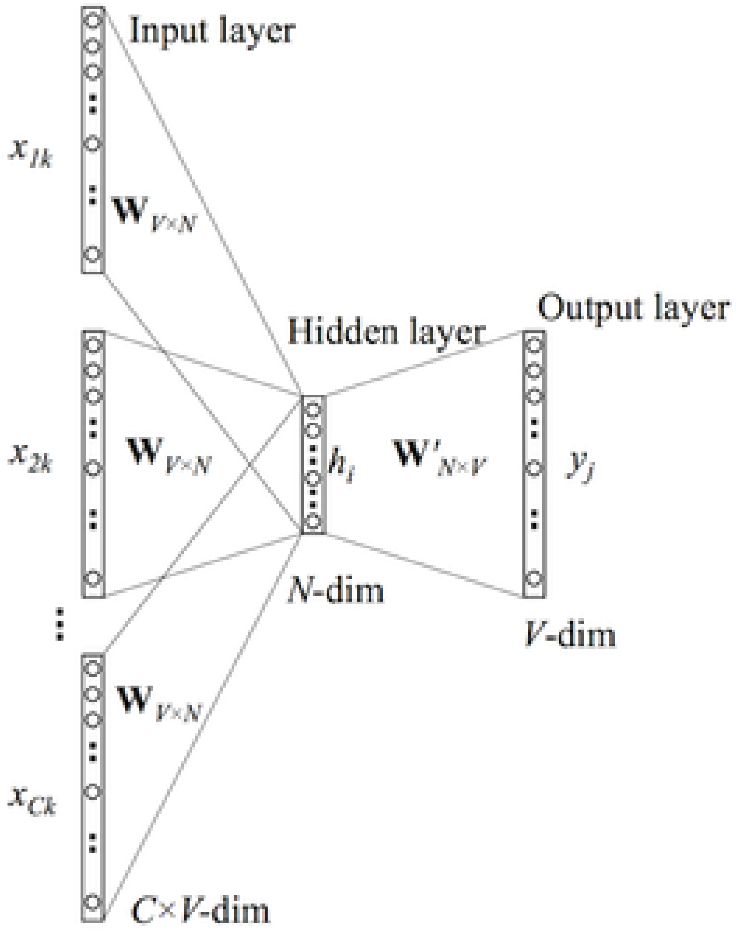
- We use neural networks to define a model to predict between context words and center word.
- CBOW predicts the center word given the context window words.
- Skip Gram predicts the context words given the center words.

e.g. “The cat jumped over the puddle”

$$\begin{bmatrix} The \\ cat \\ over \\ the \\ puddle \end{bmatrix} \longrightarrow [\,jumped\,] \qquad\qquad [\,jumped\,] \longrightarrow \begin{bmatrix} The \\ cat \\ over \\ the \\ puddle \end{bmatrix}$$

Continuous Bag-of-words (CBOW)

1. Initialize the weights V(input) and U(output) based on the word vector size we want, say H.
2. Generate one-hot vector of size N (vocabulary size) for the input context.
$$[\, x^{c-m} \;\; x^{c-m+1} \;\dots\; x^{c+m-1} \;\; x^{c+m} \,]$$
3. Get the embedded vectors for the context.
$$\begin{aligned} v^{c-m} &= Vx^{c-m} \\ v^{c-m+1} &= Vx^{c-m+1} \\ &\dots \\ v^{c+m-1} &= Vx^{c+m-1} \\ v^{c+m} &= Vx^{c+m} \end{aligned}$$
4. Average the embedded vectors to get $\hat{v}$,
$$\hat{v} = (v^{c-m} + \cdots + v^{c+m})/2m$$
5. Get the score vector z,
$$z = U\hat{v}$$
6. Get the probabilities using softmax, $\hat{y} = softmax(z)$
7. Calculate the cross-entropy loss, $L(y, \hat{y}) = -\sum_i y_i * \log \hat{y}_i$



Skip-Gram Model

1. Initialize the weights V(input) and U(output) based on the word vector size we want, say H.
2. Generate one-hot vector of size N (vocabulary size) for the center word, $[\, x^c]$
3. Get the embedded vector for the context.

   $$\hat{v} = Vx^c$$

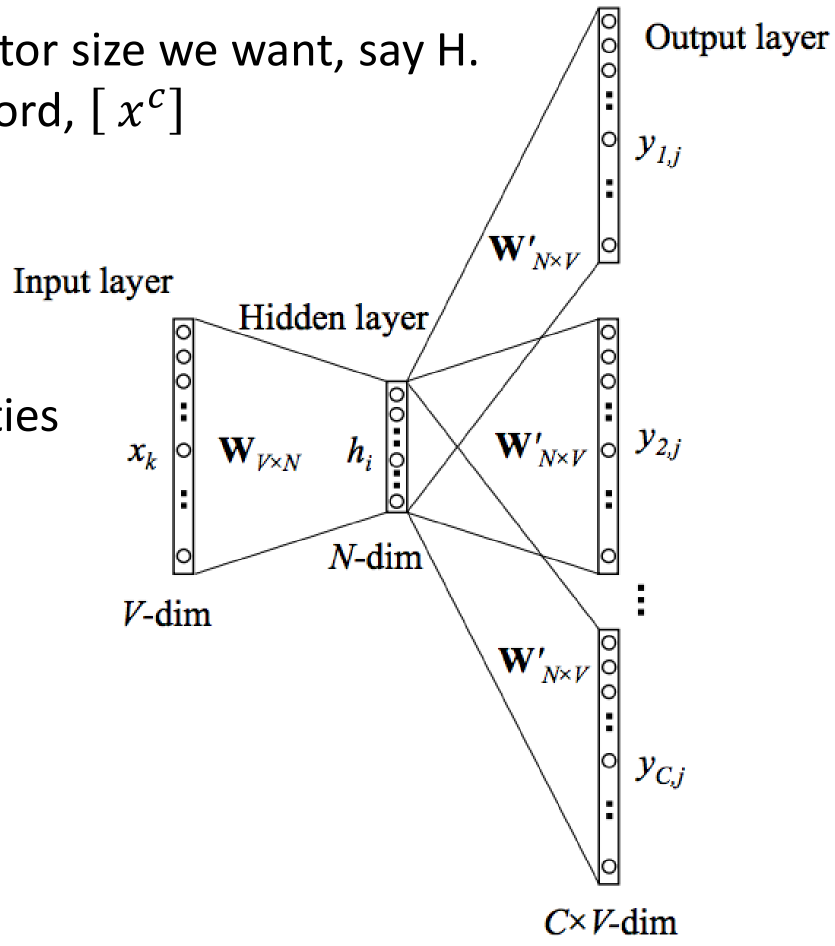
4. Get the score vector $z = U\hat{v}$
5. Get the probabilities using softmax, $\hat{y} = softmax(z)$
   - Note that $\hat{y}^{c-m}, \hat{y}^{c-m+1}, \ldots, \hat{y}^{c+m-1}, \hat{y}^{c+m}$ are the probabilities of observing context words.
6. Loss would be,

$$L = -\log P(w^{c-m}, \ldots, w^{c+m} | w^c)$$

$$= -\log \prod_{j=0, j\neq m}^{2m} P\left(w^{c-m+j} \middle| w^c\right)$$

$$= -\log \prod_{j=0, j\neq m}^{2m} \frac{\exp(u^{c-m+j}\hat{v})}{\sum_k \exp(u^k \hat{v})}$$

$$= -\sum_{j=0, j\neq m}^{2m} u^{c-m+j}\hat{v} \; + \; 2m \log \sum_k \exp(u^k \hat{v})$$



Calculate gradients for CBOW and Skip-Gram

- For Skip-Gram,

$$\frac{\partial L}{\partial u^j} = -\hat{v} + 2m \frac{\hat{v}\exp\left(u^j \hat{v}\right)}{\sum_k \exp\left(u^k \hat{v}\right)}\ ,\ j\ \in \{c-m, , \dots, c+m\}\ - c$$

$$\frac{\partial L}{\partial u^j} = 2m \frac{\hat{v}\exp(u^j \hat{v})}{\sum_k \exp(u^k \hat{v})}\ ,\ j\ \notin \{c-m, , \dots, c+m\}\ - c$$

- For homework, try calculating other gradients for CBOW and Skip-Gram.

Negative Sampling

- Loss function for Skip-Gram,

$$L = -\sum_{j=0, j\neq m}^{2m} u^{c-m+j}\hat{v} \; + \; 2m \log \boxed{\sum_{k}^{N} \exp(u^{k}\hat{v})}$$ - Summation over N computationally heavy.

- Instead of looping over all the vocabulary, we can generate some negative examples and update our loss function.
- Modified Loss function,

$$L = -\sum_{(w,c) \,\in\, D} \log \frac{1}{1+\exp(-u^{w}v^{c})} - \sum_{(w,c) \,\in\, \widetilde{D}} \log \frac{1}{1+\exp(u^{w}v^{c})}$$

For more information - http://web.stanford.edu/class/cs224n/lecture_notes/cs224n-2017-notes1.pdf

Advanced Word Vectors

- GloVe
- Fasttext

Applications

- Dependency parsing
- Machine Translation
- Named Entity Recognition
- Text Summarization
- Basically most of the NLP tasks!

Thank you