Word Window Classification

Pawan Goyal

CSE, IIT Kharagpur

February 8th, 2017

How to use Word Vectors?

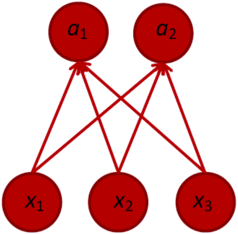
- Basic building block for the Deep learning models used for NLP
- Let's start with a simple single word classifier, e.g., for sentiments, named entities etc (without context, will add context later)

Softmax for simple single word classification

We can use a softmax classification on word vector $x \in R^d$ to obtain probability for class y.

Softmax for simple single word classification

We can use a softmax classification on word vector $x \in R^d$ to obtain probability for class y.



$W \in R^{C \times d}$, let W_y denotes the y^{th} row of W, corresponding to class y, then

$$p(y|x) = \frac{exp(W_y \cdot x)}{\sum_{c=1}^{C} exp(W_c \cdot x)}$$

Cross-entropy error

- We assume a ground truth (target) probability distribution, that is 1 at the right class and 0 everywhere (one-hot)
- let q be the computed probability distribution

Cross-entropy error

- We assume a ground truth (target) probability distribution, that is 1 at the right class and 0 everywhere (one-hot)
- let q be the computed probability distribution
- Cross-entropy error

$$H(p,q) = -\sum_{c=1}^{C} p(c) log\ q(c)$$

Cross-entropy error

- We assume a ground truth (target) probability distribution, that is 1 at the right class and 0 everywhere (one-hot)
- let q be the computed probability distribution
- Cross-entropy error

$$H(p,q) = -\sum_{c=1}^{C} p(c) log\ q(c)$$

- Because of one-hot p, the only term left is the negative probability of the true class.

Single word classification

Two options for training

- Fix your word vectors and train only softmax weights W, or
- Also train word vectors

Single word classification

Two options for training

- Fix your word vectors and train only softmax weights W, or
- Also train word vectors

What are the advantages and disadvantages?

Pro: Pretrained word vectors (on instrinsic task) can be trained further using the extrinsic task to perform better
Con: Can be risky because the words move in the vector space

Classification with word Vectors: Learning parameters

Common in deep learning: learn both W and word vectors

Classification with word Vectors: Learning parameters

Common in deep learning: learn both W and word vectors

$$\nabla_\theta J(\theta) = \begin{bmatrix} \nabla_{W_{\cdot 1}} \\ \vdots \\ \nabla_{W_{\cdot d}} \\ \nabla_{x_{aardvark}} \\ \vdots \\ \nabla_{x_{zebra}} \end{bmatrix} \in \mathbb{R}^{Cd + \boxed{Vd}}$$

Very large!

Overfitting Danger!

What is the risk?

- We need to make sure that the training set is large enough to cover most words from the vocabulary

What is the risk?

- We need to make sure that the training set is large enough to cover most words from the vocabulary
- Because otherwise, only some words are shifted in the vector space, others remain the same, and thus the performance could actually reduce.

Word Vector retraining

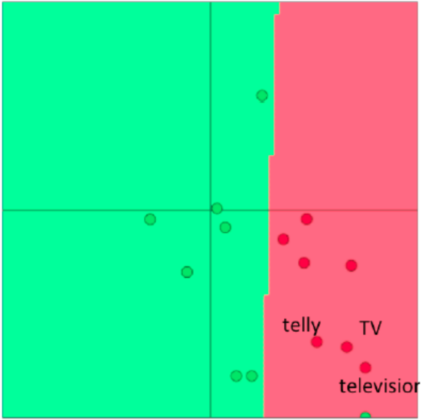


Figure 6: Here, we see that the words "Telly", "TV", and "Television" are classified correctly before retraining. "Telly" and "TV" are present in the extrinsic task training set while "Television" is only present in the test set.

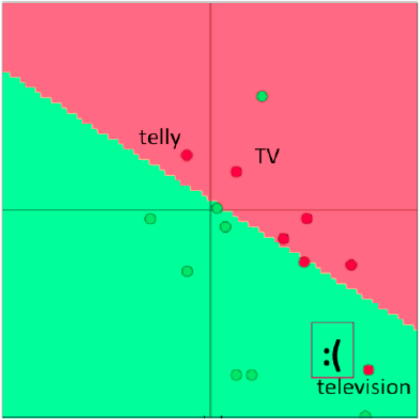


Figure 7: Here, we see that the words "Telly" and "TV" are classified correctly after traininng, but "Television" is not since it was not present in the training set.

Losing Generalization by re-training word vectors

- If you only have a small training data set, do not train the word vectors
- If you have a very large dataset, it may work better to train word vectors to the task

Window Classification

Window Classification

- **Idea:** Classify a word in its context window of neighboring words

Window Classification

- **Idea:** Classify a word in its context window of neighboring words
- **Why?** Natural languages tend to use the same word for very different meanings and we typically need to know the context of the word usage to discriminate.

Window Classification

- **Idea:** Classify a word in its context window of neighboring words
- **Why?** Natural languages tend to use the same word for very different meanings and we typically need to know the context of the word usage to discriminate.
- We thus use a sequence – central word vector preceded and succeeded by context word vectors. *Context wondow size depends on the problem being solved.*

Window Classification

- **Idea:** Classify a word in its context window of neighboring words
- **Why?** Natural languages tend to use the same word for very different meanings and we typically need to know the context of the word usage to discriminate.
- We thus use a sequence – central word vector preceded and succeeded by context word vectors. *Context wondow size depends on the problem being solved.*
- For example, named entity recognition into 4 classes: Person, location, organization, none

Window Classification

- **Idea:** Classify a word in its context window of neighboring words
- **Why?** Natural languages tend to use the same word for very different meanings and we typically need to know the context of the word usage to discriminate.
- We thus use a sequence – central word vector preceded and succeeded by context word vectors. *Context wondow size depends on the problem being solved.*
- For example, named entity recognition into 4 classes: Person, location, organization, none
- Other examples: POS tagging: DT, NN, NNP, JJ, VB ...

Window Classification

- **Idea:** Classify a word in its context window of neighboring words
- **Why?** Natural languages tend to use the same word for very different meanings and we typically need to know the context of the word usage to discriminate.
- We thus use a sequence – central word vector preceded and succeeded by context word vectors. *Context wondow size depends on the problem being solved.*
- For example, named entity recognition into 4 classes: Person, location, organization, none
- Other examples: POS tagging: DT, NN, NNP, JJ, VB ...
- We use window to not loose the position information

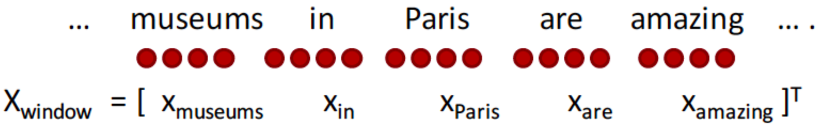
Window Classification

Example: Classify 'Paris' in the context of this sentence with window length 2:

$$X_{window} = [\ x_{museums} \quad x_{in} \quad x_{Paris} \quad x_{are} \quad x_{amazing}\ ]^T$$

Window Classification

Example: Classify 'Paris' in the context of this sentence with window length 2:



Resulting vector $x_{window} \in R^{5d}$ is a column vector.

How would you modify the Softmax?

Substitute $x^{(i)}$ with $x^{(i)}_{window}$ as

$$x^{(i)}_{window} = \begin{bmatrix} x^{(i-2)} \\ x^{(i-1)} \\ x^{(i)} \\ x^{(i+1)} \\ x^{(i+2)} \end{bmatrix}$$

Evaluating the gradient of the loss with respect to the words:

$$\delta_{window} = \begin{bmatrix} \nabla_{x^{(i-2)}} \\ \nabla_{x^{(i-1)}} \\ \nabla_{x^{(i)}} \\ \nabla_{x^{(i+1)}} \\ \nabla_{x^{(i+2)}} \end{bmatrix}$$

Which is distributed to update the corresponding word vectors.

From linear to non-linear classifiers

- Have already discussed the motivation for this
- We will see how to use neural networks as the non-linear models for this task.

NER Problem (simplified)

- For the text, “Museums in Paris are amazing”, we want to classify whether or not the center word “Paris” is a named-entity.

NER Problem (simplified)

- For the text, “Museums in Paris are amazing”, we want to classify whether or not the center word “Paris” is a named-entity.
- The input to the network is the embeddings for this window. Suppose we have 4-dim embeddings and 5-word window.

NER Problem (simplified)

- For the text, “Museums in Paris are amazing”, we want to classify whether or not the center word “Paris” is a named-entity.
- The input to the network is the embeddings for this window. Suppose we have 4-dim embeddings and 5-word window. $x \in R^{20}$

NER Problem (simplified)

- For the text, “Museums in Paris are amazing”, we want to classify whether or not the center word “Paris” is a named-entity.
- The input to the network is the embeddings for this window. Suppose we have 4-dim embeddings and 5-word window. $x \in R^{20}$
- We want a single score output from the network $s \in R$

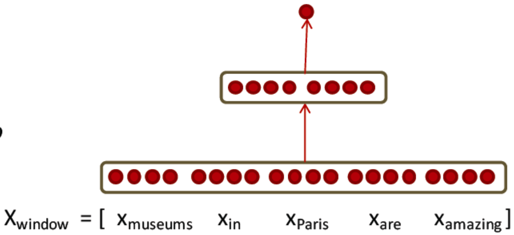
NER Problem (simplified)

- For the text, “Museums in Paris are amazing”, we want to classify whether or not the center word “Paris” is a named-entity.
- The input to the network is the embeddings for this window. Suppose we have 4-dim embeddings and 5-word window. $x \in R^{20}$
- We want a single score output from the network $s \in R$
- Suppose we use 8 sigmoid units in the hidden layer.

Feed-forward computation

$$s = U^T f(Wx + b) \qquad x \in \mathbb{R}^{20\times 1}, W \in \mathbb{R}^{8\times 20}, U \in \mathbb{R}^{8\times 1}$$

$$\begin{aligned} s &= U^T a \\ a &= f(z) \\ z &= Wx + b \end{aligned}$$



Maximum Margin Objective Function

Idea

Ensure that the score computed for “true” labeled data points is higher than the score computed for “false” labeled data points.

Maximum Margin Objective Function

Idea

Ensure that the score computed for “true” labeled data points is higher than the score computed for “false” labeled data points.

- s = score(museums in Paris are amazing)
- s_c = score(Not all museums in Paris)

Maximum Margin Objective Function

Objective

Maximize $(s - s_c)$ or to minimize $(s_c - s)$. One possible objective function: minimize $J = max(s_c - s, 0)$

Maximum Margin Objective Function

Objective

Maximize $(s - s_c)$ or to minimize $(s_c - s)$. One possible objective function: minimize $J = max(s_c - s, 0)$

What is the problem with this?

- Does not attempt to create a margin of safety. We would want the “true” labeled data point to score higher than the “false” labeled data point by some positive margin Δ.

Maximum Margin Objective Function

Objective

Maximize $(s - s_c)$ or to minimize $(s_c - s)$. One possible objective function: minimize $J = max(s_c - s, 0)$

What is the problem with this?

- Does not attempt to create a margin of safety. We would want the "true" labeled data point to score higher than the "false" labeled data point by some positive margin Δ.
- We would want error to be calculated if $(s - s_c < \Delta)$ and not just when $(s - s_c < 0)$. The modified objective:
  minimize $J = max(\Delta + s_c - s, 0)$

Maximum Margin Objective Function

Objective

Maximize $(s - s_c)$ or to minimize $(s_c - s)$. One possible objective function: minimize $J = max(s_c - s, 0)$

What is the problem with this?

- Does not attempt to create a margin of safety. We would want the "true" labeled data point to score higher than the "false" labeled data point by some positive margin Δ.
- We would want error to be calculated if $(s - s_c < \Delta)$ and not just when $(s - s_c < 0)$. The modified objective:
  minimize $J = max(\Delta + s_c - s, 0)$ $\Delta = 1$

Maximum Margin Objective Function

- Objective for a single window: $J = max(1 + s_c - s, 0)$
- Each window with a location at its center should have a score +1 higher than any window without a named entity at its center.
- $s = U^T f(Wx + b)$, $s_c = U^T f(Wx_c + b)$
- Assuming cost J is > 0, compute the derivatives of s and s_c with respect to the involved variables: U, W, b, x

Training with backpropagation

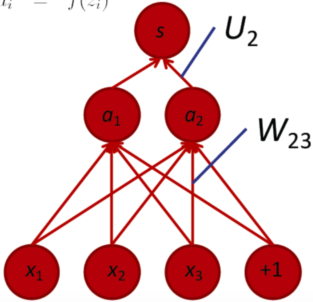
Derivative of weight W_{ij}:

Derivative of weight W_{ij}:

$$\frac{\partial}{\partial W_{ij}} U^T a \rightarrow \frac{\partial}{\partial W_{ij}} U_i a_i$$

$$\begin{aligned}
U_i \frac{\partial}{\partial W_{ij}} a_i &= U_i \frac{\partial a_i}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} \\
&= U_i \frac{\partial f(z_i)}{\partial z_i} \frac{\partial z_i}{\partial W_{ij}} \\
&= U_i f'(z_i) \frac{\partial z_i}{\partial W_{ij}} \\
&= U_i f'(z_i) \frac{\partial W_{i.}x + b_i}{\partial W_{ij}}
\end{aligned}$$

$$\begin{aligned}
z_i &= W_{i.}x + b_i = \sum_{j=1}^{3} W_{ij} x_j + b_i \\
a_i &= f(z_i)
\end{aligned}$$



Derivative continued ...

$$
\begin{aligned}
U_i \frac{\partial}{\partial W_{ij}} a_i &= U_i f'(z_i) \frac{\partial W_{i.}x + b_i}{\partial W_{ij}} \\
&= U_i f'(z_i) \frac{\partial}{\partial W_{ij}} \sum_k W_{ik} x_k \\
&= \underbrace{U_i f'(z_i)} x_j \\
&= \underbrace{\delta_i}_{\text{Local error signal}} \quad \underbrace{x_j}_{\text{Local input signal}}
\end{aligned}
$$

where $f'(z) = f(z)(1 - f(z))$ for logistic f

From single weight W_{ij} to full W:

$$\frac{\partial s}{\partial W_{ij}} = \delta_i x_j$$

- We want all combinations of $i = 1, 2, \ldots$ and $j = 1, 2, 3, \ldots$

From single weight W_{ij} to full W:

$$\frac{\partial s}{\partial W_{ij}} = \delta_i x_j$$

- We want all combinations of $i = 1, 2, \ldots$ and $j = 1, 2, 3, \ldots$
- Solution: Outer product

$$\frac{\partial J}{\partial W} = \delta x^T$$