Deep Learning

Language Modelling

Syed Irtaza Muzaffar

Outline

1. Modelling input text as numeric vectors
2. Text generation
3. Language translation

Modelling text as numeric vectors

- *Corpus*: Consider a dataset of news articles.
- *Vocabulary*: Set V^{in} of (all or most frequent) unique words in the corpus.
- Assume size of vocabulary is K^{in} words.
- Each word can be represented using 1-of-K coding.
- For example, k-th word in V can be represented as

k=5
w1=[1 0 0 0 0]
w2=[0 1 0 0 0]
w3=[0 0 1 0 0]
w4=[0 0 0 1 0]
w5=[0 0 0 0 1]

$$\mathbf{y}_k = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

where 1 appears at the k-th index.

Inefficiency of 1-hot vectors

- 1-of-K coding is
  1. *tremendously inefficient* since K^2 numbers represent K words only, and
  2. *highly unrealistic* since 1-hot vectors are orthogonal while words have similarities.

Workaround: Embedding Matrix

- Project word vectors onto lower dimensional space via *projection/embedding* matrix E.

$$\mathbf{e} = E\mathbf{y}$$

- Matrix E is of size $D \times K^{\text{in}}$ where $D << K^{\text{in}}$.
- Optimal matrix E can be learned as part of the network parameters.

Output

- Let output language have a vocabulary V^{out} of K^{out} words.
- Then output layer is softmax on K^{out} neurons.

Output

- Let output language have a vocabulary V_{out} of K_{out} words.
- Then output layer is softmax on K_{out} neurons.

$$\mathrm{softmax}\left(W'\vec{h}^{(t)} + \vec{b}'\right) \longrightarrow \begin{bmatrix} P(V_1) \\ P(V_2) \\ \vdots \\ P(V_{K_{out}}) \end{bmatrix}$$

$$\uparrow$$

$$\vec{h}^{(t)} = \tanh\left(W^0 \vec{e}_k + \vec{b}^0 + W'' \vec{h}^{(t-1)}\right)$$

$$\uparrow$$

$$\vec{e}_k = E\vec{y}_k$$

$$\uparrow$$

$$\vec{y}_k$$

$$\uparrow$$

$$V_k$$

Loss

- For a sentence of T_n words, we can use cross-entropy between output sequence and target sequence.

$$\begin{aligned}\mathcal{L}_n\left(\left(\mathbf{y}^{(1)},\mathbf{y}^{(2)},\ldots,\mathbf{y}^{(T_n)}\right),\left(\mathbf{t}^{(1)},\mathbf{t}^{(2)},\ldots,\mathbf{t}^{(T_n)}\right)\right) &= -\sum_{t=1}^{T_n}\sum_{j=1}^{K^{\text{out}}} t_j^{(t)} \ln y_j^{(t)} \\ &= -\sum_{t=1}^{T_n} \ln y_{\text{target}}^{(t)}\end{aligned}$$

- Training can be performed using BPTT on a corpus (typically) containing millions of words.
- Each sentence constitutes one training example.

Loss

n-th training sample $\{\vec{x}^{(1)}, \vec{x}^{(2)}, \dots\}, \{\vec{y}^{(1)}, \vec{y}^{(2)}, \dots, \vec{y}^{(T_n)}\}, \{\vec{t}^{(1)}, \vec{t}^{(2)}, \dots, \vec{t}^{(T_n)}\}$

- For a sentence of T_n words, we can use cross-entropy between output sequence and target sequence.

$$\mathcal{L}_n\left(\left(\mathbf{y}^{(1)}, \mathbf{y}^{(2)}, \dots, \mathbf{y}^{(T_n)}\right), \left(\mathbf{t}^{(1)}, \mathbf{t}^{(2)}, \dots, \mathbf{t}^{(T_n)}\right)\right) = -\sum_{t=1}^{T_n}\sum_{j=1}^{J} t_j^{(t)} \ln y_j^{(t)}$$

$$= -\sum_{t=1}^{T_n} \ln y_{\text{target}}^{(t)}$$

$$\underbrace{\begin{bmatrix} P(V_1) \\ P(V_2) \\ \vdots \\ P(V_J) \end{bmatrix}}_{J\times 1} \quad \underbrace{\begin{bmatrix} - \\ - \\ \vdots \\ - \end{bmatrix}}_{J\times 1} \cdots \underbrace{\begin{bmatrix} - \\ \vdots \\ - \end{bmatrix}}_{J\times 1}$$

1-hot $J\times 1$ 1-hot $J\times 1$ 1-hot $J\times 1$

- Training can be performed using BPTT on a corpus (typically) containing millions of words.
- Each sentence constitutes one training example.

$\vec{t}^{(1)} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 1 \end{bmatrix}$

$\ln y_4^{(1)}$

$\ln P(V_4)^{(1)}$

Text Generation

- Problem: generate a sequence of words $w^{(1)}, w^{(2)}, \ldots$.
- We will add two new words to each vocabulary.
  - sos: start of sentence
  - eos: end of sentence
- Solution:
  1. At time $t = 1$, feed $w^{(0)}$ the sos word. That is, starting vector is $\mathbf{x}^{(0)} = \mathbf{0}$.
  2. Compute probability distribution $\mathbf{y}^{(1)}$.
  3. Sample a word $w^{(1)}$ from this distribution.
     3.1 argmax, or
     3.2 random sampling based on probabilities in $\mathbf{y}^{(1)}$, or
     3.3 any other sampling method.
  4. At every time step $t = 1, \ldots$, feed $w(t-1)$ as input, generate probability distribution $\mathbf{y}^{(t)}$ and sample next word $w(t)$ from it.
  5. Continue until eos is sampled.

Text Generation

characters

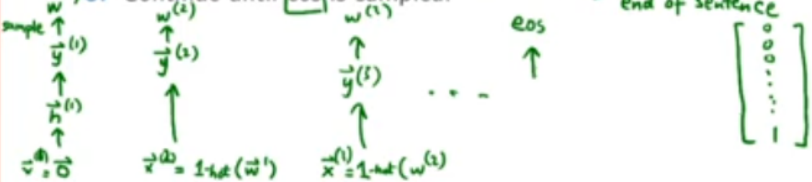
- Problem: generate a sequence of words $w^{(1)}, w^{(2)}, \ldots$.
- Solution:

start of sentence

1. At time $t = 1$, feed $w^{(0)}$ the sos word. That is, starting vector is $\mathbf{x}^{(0)} = \mathbf{0}$.
2. Compute probability distribution $\mathbf{y}^{(1)}$.
3. Sample a word $w^{(1)}$ from this distribution.
   - 3.1 argmax, or
   - 3.2 random sampling based on probabilities in $\mathbf{y}^{(1)}$, or
   - 3.3 any other sampling method.
4. At every time step $t = 1, \ldots$, feed $w(t-1)$ as input, generate probability distribution $\mathbf{y}^{(t)}$ and sample next word $w(t)$ from it.
5. Continue until eos is sampled.

Uniform

end of sentence

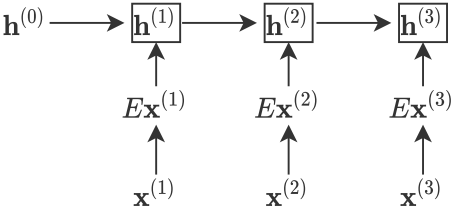


Language Translation

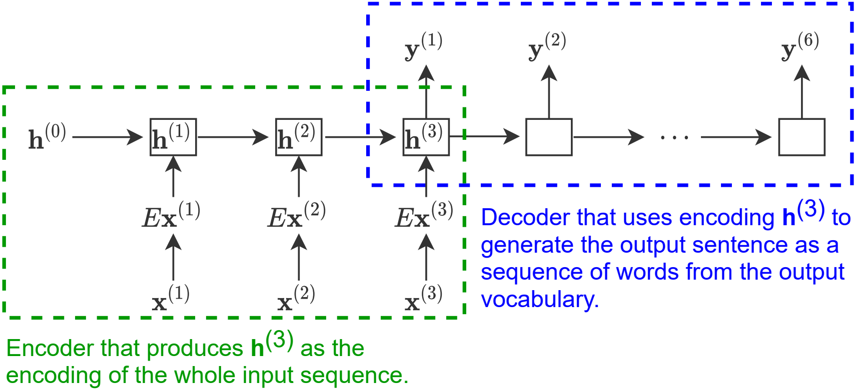
Zaid slapped Khalid ⟶ زید نے خالد کو تھپڑ مارا

$\mathbf{x}^{(1)}$ $\mathbf{x}^{(2)}$ $\mathbf{x}^{(3)}$ $\mathbf{y}^{(6)}$ $\mathbf{y}^{(5)}$ $\mathbf{y}^{(4)}$ $\mathbf{y}^{(3)}$ $\mathbf{y}^{(2)}$ $\mathbf{y}^{(1)}$

Language Translation

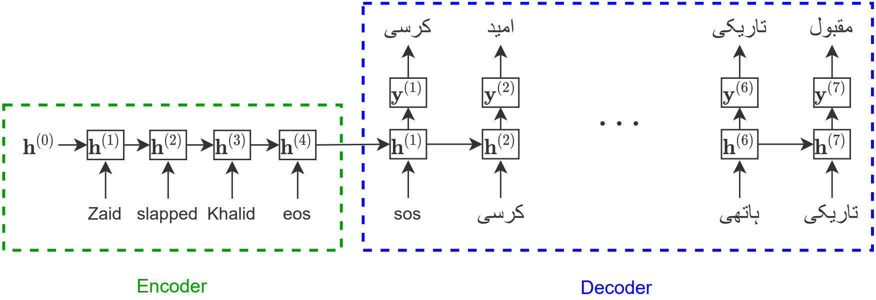


Language Translation



Language Translation

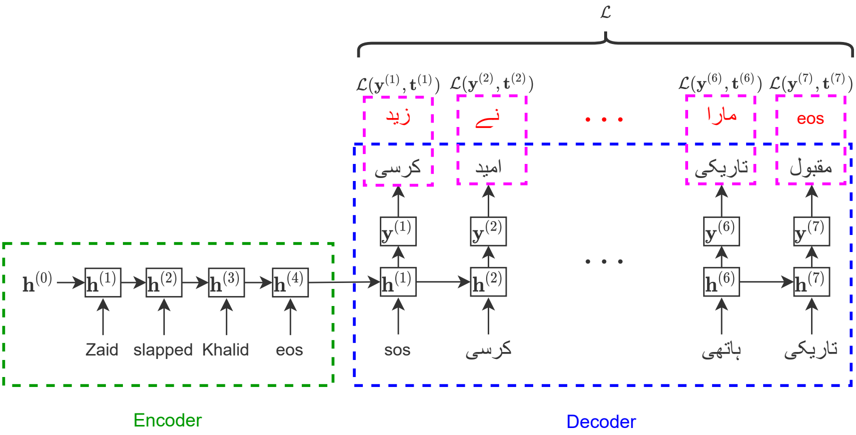
A better decoder



Make probability distribution $\mathbf{y}^{(t+1)}$ depend on *word drawn* from $\mathbf{y}^{(t)}$ as well.

$$y_j^{(t)} = P(o^{(t)} = V_j | \underbrace{o^{(t-1)}, o^{(t-2)}, \ldots, o^{(1)}}_{\text{all words output so far}}, \underbrace{w^{(1)}, w^{(2)}, \ldots, w^{(T_{\text{in}})}}_{\text{all input words}})$$

Language Translation
Training



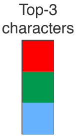
Language Translation

Testing: Finding the most likely output

- As mentioned earlier, sampling of words can be accomplished via
  1. argmax on each $\mathbf{y}^{(t)}$, or
  2. random sampling from each $\mathbf{y}^{(t)}$
- Both sampling methods produce locally optimal words.
- A better but costlier alternative is to find a globally optimal output sequence.

Beam Search

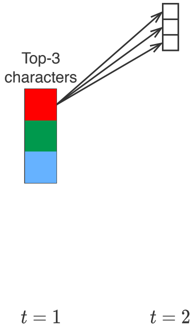
At time $t = 1$, pick the M most probable options instead of all K^{out} options.



$t = 1$

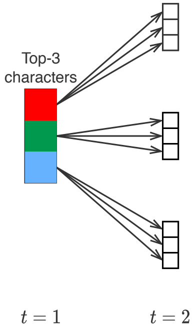
Beam Search

Conditioned on each option at $t = 1$, pick the M most probable options at $t = 2$.



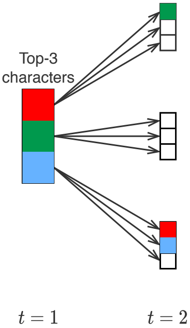
Beam Search

Conditioned on each option at $t = 1$, pick the M most probable options at $t = 2$.



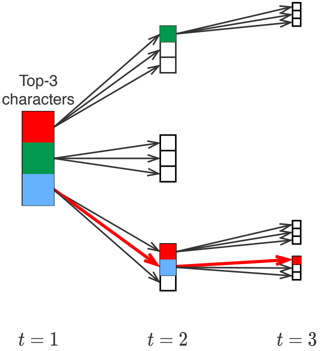
Beam Search

Conditioned on each option at $t = 1$, pick the M most probable options at $t = 2$.



Beam Search

Conditioned on each *path* at $t = 2$, pick the M most probable options at $t = 3$.



Beam Search

- A sequence is terminated when eos is drawn.
- When no unterminated sequence remains, select the most likely sequence across all terminating sequences.

Summary

- Words in a language can be modeled as 1-hot vectors.
- Learnable embedding matrices can reduce dimensions.
- Text generation models are *stochastic parrots*.
- Language translation can be achieved through the encoder-decoder framework.
- Beam-search makes decoding approximate but tractable.