Chapter 5 – Language Modelling

Fabien Petit

Department of Economics, University of Barcelona
Centre for Education Policy and Equalising Opportunities, UCL

Introduction

Information encoded into text has many applications in social sciences and economics (Gentzkow et al. 2019)

- Political economy (e.g., speech), macro-economics (e.g., uncertainty), media economics (e.g., slant)
- Applications include topic classification, sentiment analysis, and question answering (LeCun et al. 2015)
- Enormous progress in computational linguistics and machine learning over the past decade

Introduction

Sentences can have similar representations, but different meanings; we need to account for the **syntactic structure**

- "The lecture was bad, not good" and "The lecture was good, not bad" have the same bagged representations
- The feed-forward network studied previously can be applied to a document's stacked word vectors
- The model can capture non-linearities and interactions among the input word vectors

Introduction

Recurrent networks (Rumelhart et al. 1986) are network architectures optimized for sequential data

- State-of-the-art performance in natural language processing, sentiment analysis, and topic modelling
- They are called "recurrent" because the state of the hidden units depends on their previous values
- Recurrence imposes constraints on the model, i.e., sequential connectivity, shared parameters

Text as Data

Structure

Data structured as a corpus (sample) of documents (observations) containing sequences of words (variables)

- **Semantic dimension**: encode the meaning of words and their relative distance (e.g., synonyms, homonyms)
- **Syntactic dimension**: capture interactions between words that create meaning (e.g., Aix-Marseille School of Economics, not)
- Traditional approaches often ignore one or both dimensions (e.g., count variables, additive models)

Structure

Natural language is a **mechanism** to encode and decode ideas into words and sentences; there exists a mapping

- **Dimensionality** due to the number of words in a language and their effective numerical representation
- **Interactions and non-linearities** among words in a document (e.g., sequence, grammatical structure)
- Applications with text data often involve a prediction problem, so flexible methods are available

Pre-Processing

Pre-processing operations include tokenization, breaking the document into individual tokens (e.g., words)

- Encoding, spelling, converting to lowercase, removing numbers, punctuation, special characters
- Stemming and lemmatization, reducing words to their root using a vocabulary and morphological analysis
- Filtering out common (e.g., *a*, *an*) and rare words (i.e., dimensions) using distinctiveness metrics

Word Representations

Word Representations

Words have **no obvious numerical representation**, while models require ordered numerical values

- A naive approach is to use binary variables of dimension v (i.e., vocabulary), the number of unique words
- Each variable takes the value 1 at the word's index position in the dictionary and 0 otherwise
- These representations are potentially high-dimensional, sparse, and do not capture similarity among words

Word Representations – Binary

Consider the following document: "University students learn about language modelling"

Tokens	w_i					
about	1	0	0	0	0	0
language	0	1	0	0	0	0
learn	0	0	1	0	0	0
modelling	0	0	0	1	0	0
students	0	0	0	0	1	0
university	0	0	0	0	0	1

Binary

Document x_i can be represented as **a sequence** x_{it}, with $t = \{1\ldots, T\}$ with v-dimensional word vectors

$$\begin{array}{cccccc} \text{university} & \text{students} & \text{learn} & \text{about} & \text{language} & \text{modelling} \\ \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix} & \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} \end{array}$$

This representation does not encode the distance between words e.g., students should be closer to learn than about

Similarity

Cosine similarity is used to measure distance between vectors w_i and w_j of dimension k (here $k = v$)

$$S_c(w_i, w_j) = \frac{w_i \cdot w_j}{\|w_i\| \, \|w_j\|} = \frac{\sum_{d=1}^{k} w_{id} * w_{jd}}{\sqrt{\sum_{d=1}^{k} w_{id}^2}\sqrt{\sum_{d=1}^{k} w_{jd}^2}}$$

where $\cdot$ is the dot product and $\|.\|$ denotes the vector norm

- The similarity ranges from -1 (i.e., exact opposite) to 1 (i.e., exactly the same), 0 indicating orthogonality
- Binary representations are orthogonal and do not capture semantic similarity between words

Distributed

The distributional structure of language (Harris 1954; Firth 1957) can be used to compute better word representations

- Fundamental idea, the distribution of words and their co-occurrence in a large corpus is **not random**
- Words appearing in similar contexts tend to have similar meanings, i.e., "know a word by the company it keeps"
- Context can be the entire document or, more commonly, a small moving window around the word

Co-occurrence

Consider a target word w_i (i.e. orange) and its neighboring context words c_i (i.e. window)

$\boxed{\text{university students learn}}$			about	language	modelling
university	$\boxed{\text{students learn about}}$			language	modelling
university	students	$\boxed{\text{learn about language}}$			modelling
university	students	learn	$\boxed{\text{about language modelling}}$		

Similar to a 1-dimensional convolution, capturing interactions between words and their context. The window size s controls the context size (e.g,. typically between 3 to 9). This procedure is applied to the entire corpus.

Co-occurrence

When applied to a large corpus, this approach gives each word w_i scores along each dimension of c

students learn language skills at university
university offers language modelling courses
students explore language nuances
learning about modelling, students develop skills
university teaches language modelling concepts
students at university learn language intricacies
modelling techniques fascinate students
university students study language
$\vdots$

Co-occurrence

A **co-occurrence matrix** with dimensions $v \times v$ counts the context words appearing in the window of the target word

		c_i					
		about	language	learn	modelling	students	university
w_i	about	0	4	3	2	3	3
	language	4	0	6	4	5	5
	learn	3	6	0	4	4	4
	modelling	2	4	4	0	3	3
	students	3	5	4	3	0	6
	university	3	5	4	3	6	0

Co-occurrence

This representation is distributed since each context word is used to represent multiple target words

- Captures semantics, different words appearing in a similar context have a strong cosine similarity
- Frequent words are present in many contexts, they are not distinctive, and bias the representations
- **Computationally inefficient**, this representation remains high-dimensional (i.e, v) and sparse (i.e., many 0)

Weights

Pointwise mutual information (PMI) is a measure of association between two words that improves on simple counts

$$
\begin{aligned}
\mathrm{PMI}\left(w_i, w_j\right) &= \log_2\left(\frac{p\left(w_i, w_j\right)}{p\left(w_i\right) p\left(w_j\right)}\right) \\
&= \log_2\left(\frac{n * \mathrm{count}\left(w_i, w_j\right)}{\mathrm{count}\left(w_i\right) * \mathrm{count}\left(w_j\right)}\right)
\end{aligned}
$$

where n is the total number of words in the corpus

- The ratio compares the probability of co-occurrence to the product of their individual probabilities
- Interpreted as the expected co-occurrence with respect to co-occurrence by chance (i.e., independence)

Weights

The PMI penalizes frequent words in favor of co-occurrences between words that appear less frequently

- When w_i and w_j are independent, their joint probability equals the product of their individual probabilities
- $PMI(w_i, w_j)$ varies between $[-\infty; +\infty]$ and becomes negative when the probability ratio is smaller than 1
- Negative PMIs are not well defined and are usually clipped at 0 (i.e., co-occur less than expected by chance)

Network Embeddings

Network Embeddings

Learned embedding (Bengio et al. 2003) represented a breakthrough for natural language modelling problems

- Embedding networks compute meaningful (more on this), dense and distributed word representations
- Word2vec (Mikolov et al. 2013a,b) are two models relating words to their context (i.e., CBOW and Skip-Gram)
- Alternatives include e.g. GloVE (Pennington et al. 2014), ELMo (Peters et al. 2018), BERT (Devlin et al. 2019)

Intuition

Consider the following words: king, queen, man, woman, prince, princess, boy, girl

- Which three dimensions can be used to represent all these words, i.e., distributed?
- How can we discover automatically these latent dimensions from the data?

Intuition

These words could be represented along the power, gender, and age dimensions (using binary scores for simplicity)

			c_i	
		power	female	age
	king	1	0	1
	queen	1	1	1
	man	0	0	1
w_i	woman	0	1	1
	prince	1	0	0
	princess	1	1	0
	boy	0	0	0
	girl	0	1	0

Intuition

Distances across dimensions are consistent

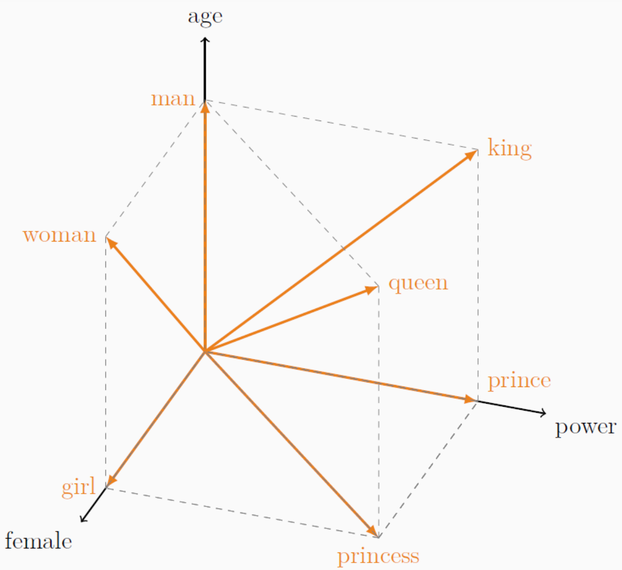
$$\begin{array}{ccccccc} \text{king} & & \text{prince} & & \text{man} & & \text{boy} \\ \begin{bmatrix}1\\0\\1\end{bmatrix} & - & \begin{bmatrix}1\\0\\0\end{bmatrix} & = & \begin{bmatrix}0\\0\\1\end{bmatrix} & - & \begin{bmatrix}0\\0\\0\end{bmatrix} \end{array}$$

These distances reflect associations in language

$$\begin{array}{ccccccc} \text{king} & & \text{man} & & \text{woman} & & \text{queen} \\ \begin{bmatrix}1\\0\\1\end{bmatrix} & - & \begin{bmatrix}0\\0\\1\end{bmatrix} & + & \begin{bmatrix}0\\1\\1\end{bmatrix} & = & \begin{bmatrix}1\\1\\1\end{bmatrix} \end{array}$$

In practice, continuous rather than binary scores are used

Intuition



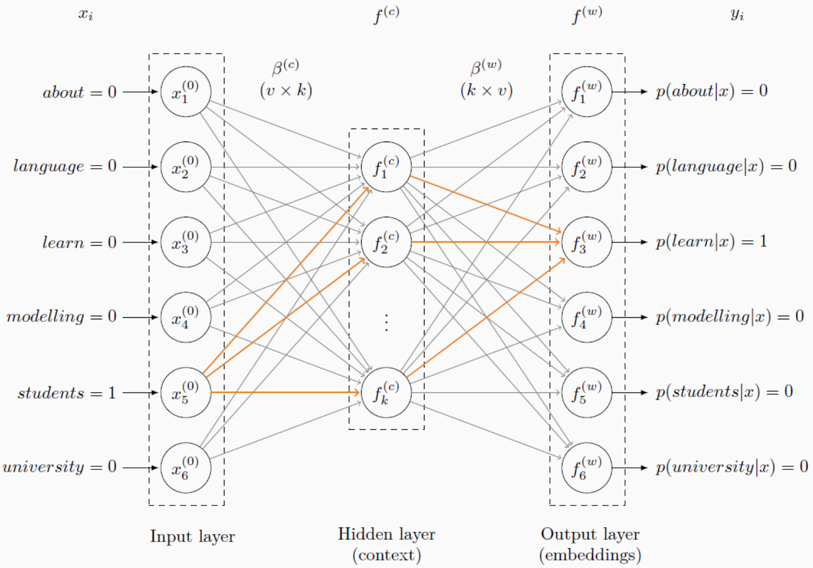
Word2Vec

For simplicity, we predict the conditional probability of a word from the preceding context word (i.e., bigrams)

x_i or context word(s) c_i		y_i or target w_i	
university	$\begin{bmatrix}0 & 0 & 0 & 0 & 0 & 1\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$	students
students	$\begin{bmatrix}0 & 0 & 0 & 0 & 1 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}$	learn
learn	$\begin{bmatrix}0 & 0 & 1 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$	about
about	$\begin{bmatrix}1 & 0 & 0 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}$	language
language	$\begin{bmatrix}0 & 1 & 0 & 0 & 0 & 0\end{bmatrix}$	$\begin{bmatrix}0 & 0 & 0 & 1 & 0 & 0\end{bmatrix}$	modelling

Consider a particular observation e.g. $x_i =$ students and $y_i =$ learn

Word2Vec



x_i
$f^{(c)}$
$f^{(w)}$
y_i
$\beta^{(c)}$
$(v \times k)$
$\beta^{(w)}$
$(k \times v)$
$about = 0$
$language = 0$
$learn = 0$
$modelling = 0$
$students = 1$
$university = 0$
$x_1^{(0)}$
$x_2^{(0)}$
$x_3^{(0)}$
$x_4^{(0)}$
$x_5^{(0)}$
$x_6^{(0)}$
$f_1^{(c)}$
$f_2^{(c)}$
$\vdots$
$f_k^{(c)}$
$f_1^{(w)}$
$f_2^{(w)}$
$f_3^{(w)}$
$f_4^{(w)}$
$f_5^{(w)}$
$f_6^{(w)}$
$p(about|x) = 0$
$p(language|x) = 0$
$p(learn|x) = 1$
$p(modelling|x) = 0$
$p(students|x) = 0$
$p(university|x) = 0$
Input layer
Hidden layer
(context)
Output layer
(embeddings)

Word2Vec

The hidden layer has k units (i.e., embedding dimension) and the identity activation function (i.e., $x_i^{(c)} = x_i^{(0)} \cdot \beta^{(c)}$)

$$x_i^{(c)} = \begin{bmatrix} 0 & \cdots & 1 & 0 \end{bmatrix}_{1 \times v} \cdot \begin{bmatrix} \beta_{11}^{(c)} & \beta_{21}^{(c)} & \cdots & \beta_{k1}^{(c)} \\ \vdots & \vdots & & \vdots \\ \beta_{15}^{(c)} & \beta_{25}^{(c)} & \cdots & \beta_{k5}^{(c)} \\ \beta_{16}^{(c)} & \beta_{26}^{(c)} & \cdots & \beta_{k6}^{(c)} \end{bmatrix}_{v \times k}$$

$$x_i^{(c)} = \begin{bmatrix} \beta_{15}^{(c)} & \beta_{25}^{(c)} & \cdots & \beta_{k5}^{(c)} \end{bmatrix}_{1 \times k}$$

The hidden layer effectively selects for each unit the parameters corresponding to the input word

Word2Vec

For an observation (e.g. $y_i = learn$), the model performs a dot product between the context and the word representation

$$p\left(y_i = 1 \mid x_i\right) = \sigma\left(\beta^{(c)}_{[x_i]} \cdot \beta^{(w)}_{[y_i]}\right)$$

where $\beta^{(c)}_{[x_i]}$ and $\beta^{(w)}_{[y_i]}$ are the row $(1 \times k)$ and column $(k \times 1)$ of $\beta^{(c)}$ and $\beta^{(w)}$ corresponding x_i and y_i, respectively

- $\beta^{(c)}_{[w_i]}$ represents the word when it appears as a context, and $\beta^{(w)}_{[w_i]}$ the other when it appears as a target
- The model computes two representations for every word, as a context and as a target word

Word2Vec

Computing the gradients for $\beta^{(w)}_{[y_i]}$, we could show that the update rule can be written as

$$\beta^{(w)}_{[y_i]} \Leftarrow \beta^{(w)}_{[y_i]} + \eta \beta^{(c)}_{[x_i]} (1 - \hat{y}_i)$$

where $\hat{y}_i$ is the model prediction for y_i

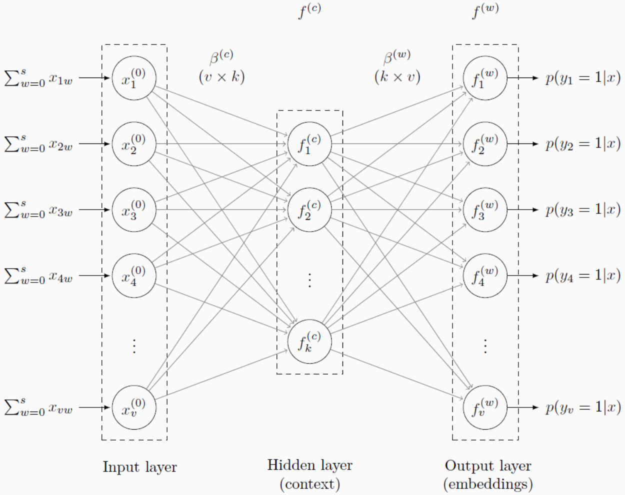
- When $\hat{y}_i < 1$, $\beta^{(w)}_{[y_i]}$ is updated with a share of $\beta^{(c)}_{[x_i]}$ so the two vectors become closer (e.g. cosine similarity)
- During training, two target words appearing in similar context have their $\beta^{(w)}_{[y_i]}$ converge toward similar $\beta^{(c)}_{[x_i]}$

Word2Vec

The columns parameters of $\beta^{(w)}$ contains the k-dimensional embeddings for every word in the dictionary

- The embedding for "student" should come close to e.g., "learn", "university" in the context matrix
- Words that appear in the same context, e.g., "professor", should converge toward similar representations
- Importantly, this does not mean that "student" is similar to "learn" or "university" in the embedding matrix

Continuous Bag of Words



The model is usually estimated with a context size $s > 1$. The input is the sum of the binary context vectors (i.e. the order is ignored). This hidden layer effectively sums up the $\beta^{(c)}_{[x_i]}$ of the context words. For each (x_i, y_i) pair, $\beta^{(c)}_{[x_i]}$ and all the $\beta^{(w)}$ parameters are updated.

Evaluating Embeddings

Evaluating Embeddings – Synonyms

Synonym detection can be used to evaluate word embeddings, e.g,. for w_{learn} we could have

$$w_{\text{train}},\ w_{\text{atmosphere}},\ w_{\text{defend}},\ w_{\text{internet}},\ w_{\text{route}},\ldots$$

- For every tested embedding, we choose a synonym (e.g. synonym dictionary) and a sample of random words
- Using the cosine similarity, we count the number of times the closest embedding is the synonym (i.e., accuracy)

Evaluating Embeddings – Analogies

Network embeddings effectively capture semantic and syntactic analogies present in natural language

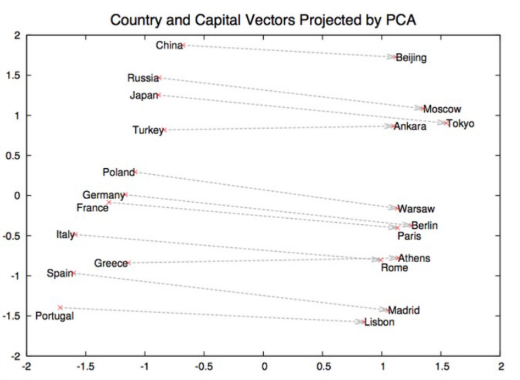
$$\begin{aligned} \text{Semantic:} \quad & w_{\text{queen}} && \approx w_{\text{king}} - w_{\text{man}} + w_{\text{woman}} \\ \text{Syntactic:} \quad & w_{\text{speaks}} - w_{\text{speak}} && \approx w_{\text{listens}} - w_{\text{listen}} \end{aligned}$$

- This is possible because distances along each embedding dimension are relatively well-defined and consistent
- Possible mathematical operations include positive (addition) and negative (subtraction) associations

Evaluating Embeddings – Analogies

Figure: Additional Examples (Mikolov et al. 2013a)

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs



Evaluating Embeddings – Visualization

T-distributed Stochastic Neighbor Embedding (Maaten and Hinton 2008) represents high-dimensional data points[1]

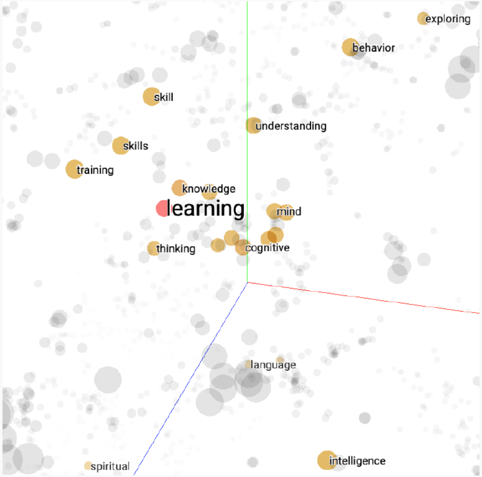
- T-SNE performs non-linear dimensionality reduction, while preserving the neighborhood between points
- Dimensionality reduction involves distortions, e.g., don't interpret cluster sizes and the distances between clusters
- See Wattenberg et al. (2016) for practical guidelines and interpretation (i.e., tuning parameters)

[1] Alternatives such as Uniform Manifold Approximation and Projection (McInnes et al. 2018) preserves both the local and the global structure.

Evaluating Embeddings – Visualization

Use the TensorFlow embedding projector to visualize the embedding space

- Use the T-SNE 3D visualization on Word2Vec 10k embeddings
- Search for words and their connections using cosine similarity



Recurrent Networks

Recurrent Networks

We want to predict the probability that movie reviews x_i express a positive sentiment $y_i = 1$

- We approximate the function mapping words sequences x_{it}, $t = \{1, \ldots, T\}$ to a probability $P(y_i = 1 \mid x_i)$
- The input can be either v-dimensional binary vectors or lower-dimensional embedding vectors

Recurrent Networks

The Large Movie Review Dataset (Maas et al. 2011) contains 782 labeled reviews for training and 782 for testing

x_i	y_i
"...I really loved it. The cast is excellent and the plot is sometimes absolutely hilarious. Go and see. It's great..."	1
"...bad acting, horrible cinematography, lame plot and some decent special effects do not make a good movie..."	0
$\vdots$	$\vdots$

Documents with similar meaning can have very different representations, and conversely

Recurrent Networks

We explicitly model the sequential structure of the data, i.e., the interactions of words within a sentence

- Documents have varying lengths and an obvious sequential ordering (e.g., potentially two-way)
- The inputs at different **time-steps** are not independent and interact at different levels (e.g., grammar)
- Model the transition between two inputs, the parameters are shared across **time-steps** (e.g., language rules)

Units in Recurrent Neural Networks (RNN)

Using t as the **time-step** and s for the **hidden state**, the equation for a **hidden unit** is:

$$s_u^{(t)} = f\left(s^{(t-1)}, \beta_u^s\right) = \sigma^s\left(s^{(t-1)} \cdot \beta_u^s\right),$$

where $s^{(t-1)}$ is a row vector and β_u^s are the weights for the previous state. Here, the current state depends only on the previous state.

To make this unit **recurrent**, we allow the state at time-step t to be:

$$s_u^{(t)} = \sigma^s\left(s^{(t-1)} \cdot \beta_u^s + x^{(t)} \cdot \beta_u^x\right),$$

where $x^{(t)}$ is the new input, $s^{(0)} = 0$ and σ^s, β_u^s and β_u^x are shared across all time steps.

The constant (bias) is omitted for simplicity

Units

Recurrence implies that the current state contains information about all the previous states

$$s^{(t)} = f\left(s^{(t-1)}, x^{(t)}, \beta^s\right) \tag{1}$$

$$s^{(t)} = f\left(f\left(s^{(t-2)}, x^{(t-1)}, \beta^s\right), x^{(t)}, \beta^s\right) \tag{2}$$

$$s^{(t)} = f\left(f\left(f\left(s^{(t-3)}, x^{(t-2)}, \beta^s\right), x^{(t-1)}, \beta^s\right), x^{(t)}, \beta^s\right) \tag{3}$$

where $s^{(t)}$ is the state of all the units of a single hidden layer

- This mechanism allows the model to capture non-linear interactions between inputs at different time-steps
- The model can process input sequences of arbitrary lengths, the parameters are shared across iterations

Units

Using the state of the network at time t, the model can (potentially) produce a prediction at every time step

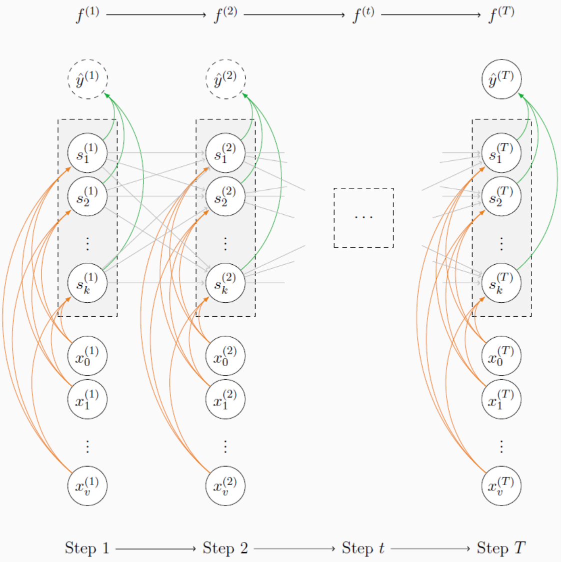
$$s^{(t)} = \sigma^s \left(s^{(t-1)} \cdot \beta^s + x^{(t)} \cdot \beta^x \right), \tag{4}$$

$$y^{(t)} = \sigma^y \left(s^{(t)} \cdot \beta^y \right), \tag{5}$$

where σ^y is the output activation function and β^y is also shared across time-steps. Constants are excluded

- Some applications require a single prediction at the end of the sequence (e.g., sentiment analysis)
- Others require a prediction at every time-step (e.g., time series forecasting, next character prediction)

Structure



Recurrent network with a single hidden layer, unfolded across time-steps. The states of hidden units are updated at every time step. For each unit, the parameters β_u^s (i.e. gray), β_u^x (i.e. orange) and β_u^y (i.e. green) are shared across time-steps. This mechanism reduces the number of parameters, while allowing the model to generalize across entire sequences (i.e., same interaction patterns).

Abstract Recurrent Network

$$f^{(t)}$$

$$\hat{y}^{(t)}$$

$$\uparrow$$

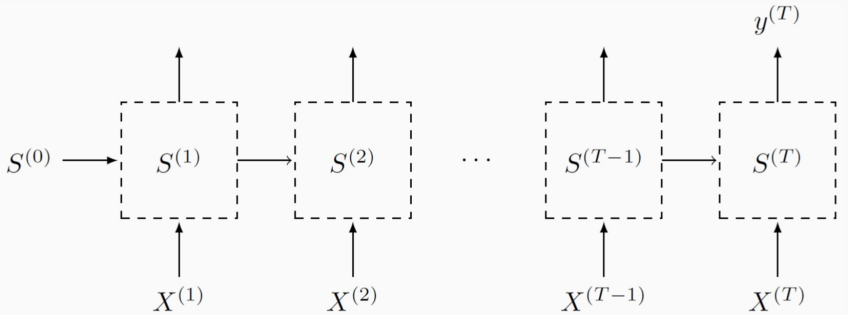
$$\boxed{\begin{aligned} S^{(t)} &= \sigma^h\left(S^{(t-1)} \cdot \beta^s + X^{(t)} \cdot \beta^x\right) \\ \hat{y}^{(t)} &= \sigma^y\left(S^{(t)} \cdot \beta^y\right) \end{aligned}}$$

$$\uparrow$$

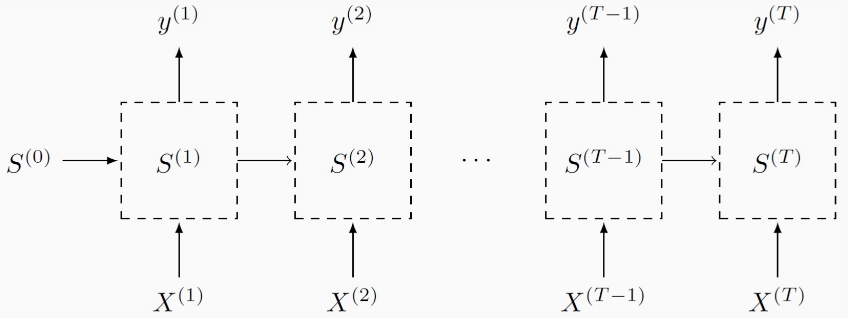
$$X^{(t)}$$

The looping arrow denotes recursion. Generalizing to multiple observations, $X^{(t)}$ has dimensions $n \times v$ and $S^{(t)}$ has dimensions $n \times k$. Therefore, β^x has dimensions $v \times k, \beta^s$ has dimensions $k \times k$. Constants are excluded.

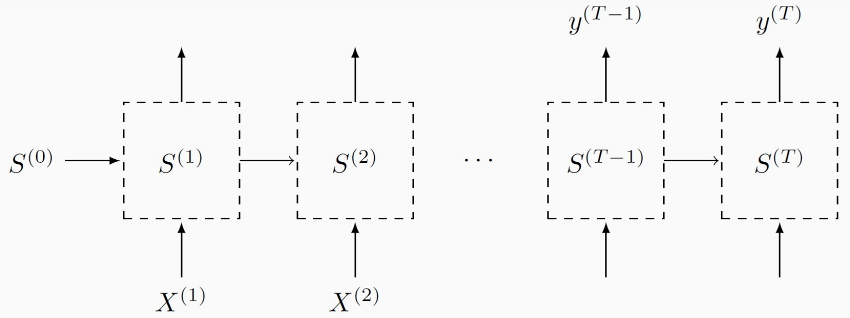
Recurrent Networks – Applications



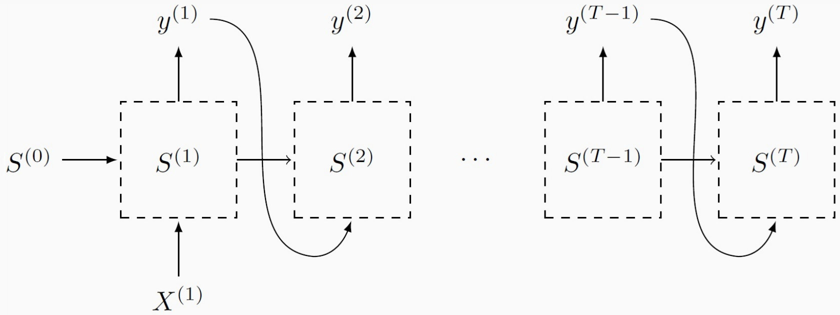
Recurrent Networks – Applications



Recurrent Networks – Applications



Recurrent Networks – Applications



Optimization

Optimization

With outputs at multiple time-steps, the loss for an observation is summed across time-steps

$$\mathcal{L}_i = \sum_{t=1}^{T} \mathcal{L}\left(y_{it}, \hat{y}_{it}\right)$$

- We need to compute the average gradients of the loss function with respect to β^s, β^x, and β^y
- For $\partial\mathcal{L}(\beta)/\partial\beta^y$ we can compute the (summed) gradient for each time step with the usual formula
- For $\partial\mathcal{L}(\beta)/\partial\beta^s$ and $\partial\mathcal{L}(\beta)/\partial\beta^x$ backpropagation must be modified because of recurrence

Optimization – Backpropagation Through Time

Backpropagation through time (BPTT) is a modified optimization algorithm for recurrent networks

$$\frac{\partial \mathcal{L}^{(T)}}{\partial \beta^s} = \frac{\partial \mathcal{L}^{(T)}}{\partial s^{(T)}} \frac{\partial s^{(T)}}{\partial \beta^s}$$

Notice that $s^{(T-1)}$ cannot be considered as a constant when computing the partial derivatives, specifically

$$s^{(T)} = \sigma^s \left(s^{(T-1)} \cdot \beta^s \right) \tag{6}$$

$$s^{(T-1)} = \sigma^s \left(s^{(T-2)} \cdot \beta^s \right) \tag{7}$$

Optimization – Backpropagation Through Time

However, we can express this partial derivative as the sum of an **explicit** and an **implicit** component

$$\frac{\partial s^{(T)}}{\partial \beta^{s}} = \underbrace{\frac{\partial^{+} s^{(T)}}{\partial \beta^{s}}}_{\text{explicit}} + \underbrace{\frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial s^{(T-1)}}{\partial \beta^{s}}}_{\text{implicit}}$$

where ∂^{+}denotes that inputs are assumed constant

- The explicit component captures the contribution of the parameters to the state at time t
- The implicit component sums across all the previous paths and can be developed for all time-steps

Optimization – Backpropagation Through Time

$$\frac{\partial s^{(T)}}{\partial \beta^s} = \frac{\partial^+ s^{(T)}}{\partial \beta^s} + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \left(\frac{\partial^+ s^{(T-1)}}{\partial \beta^s} + \frac{\partial s^{(T-1)}}{\partial s^{(T-2)}} \frac{\partial s^{(T-2)}}{\partial \beta^s} \right)$$

$$\begin{aligned}
\frac{\partial s^{(T)}}{\partial \beta^s} &= \frac{\partial^+ s^{(T)}}{\partial \beta^s} + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial^+ s^{(T-1)}}{\partial \beta^s} \\
&\quad + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial s^{(T-1)}}{\partial s^{(T-2)}} \left(\frac{\partial^+ s^{(T-2)}}{\partial \beta^s} + \frac{\partial s^{(T-2)}}{\partial s^{(T-3)}} \frac{\partial s^{(T-3)}}{\partial \beta^s} \right)
\end{aligned}$$

$$\frac{\partial s^{(T)}}{\partial \beta^s} = \dots$$

Optimization – Backpropagation Through Time

$$
\begin{aligned}
\frac{\partial s^{(T)}}{\partial \beta^s} &= \frac{\partial^+ s^{(T)}}{\partial \beta^s} + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial^+ s^{(T-1)}}{\partial \beta^s} + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial s^{(T-1)}}{\partial s^{(T-2)}} \frac{\partial^+ s^{(T-2)}}{\partial \beta^s} \\
&\quad + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial s^{(T-1)}}{\partial s^{(T-2)}} \frac{\partial s^{(T-2)}}{\partial s^{(T-3)}} \frac{\partial s^{(T-3)}}{\partial \beta^s} + \dots \\
\frac{\partial s^{(T)}}{\partial \beta^s} &= \frac{\partial s^{(T)}}{\partial s^{(T)}} \frac{\partial^+ s^{(T)}}{\partial \beta^s} + \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial^+ s^{(T-1)}}{\partial \beta^s} + \frac{\partial s^{(T)}}{\partial s^{(T-2)}} \frac{\partial^+ s^{(T-2)}}{\partial \beta^s} \\
&\quad + \frac{\partial s^{(T)}}{\partial s^{(T-3)}} \frac{\partial s^{(T-3)}}{\partial \beta^s} + \dots \\
\frac{\partial s^{(T)}}{\partial \beta^s} &= \sum_{t=1}^{T} \frac{\partial s^{(T)}}{\partial s^{(t)}} \frac{\partial^+ s^{(t)}}{\partial \beta^s}
\end{aligned}
\tag{1}
$$

Issues

Consider the first term of the equation (1) using $t = 1$. Developing this term gives

$$\frac{\partial s^{(T)}}{\partial s^{(1)}} = \frac{\partial s^{(T)}}{\partial s^{(T-1)}} \frac{\partial s^{(T-1)}}{\partial s^{(T-2)}} \cdots \frac{\partial s^{(2)}}{\partial s^{(1)}} = \prod_{t=2}^{T} \frac{\partial s^{(t)}}{\partial s^{(t-1)}} \tag{2}$$

Remember the equation for a given unit, and the partial derivative of the sigmoid activation function

$$s^{(t)} = \sigma^s\left(z^{(t)}\right) = \sigma^s\left(s^{(t-1)} \cdot \beta^s + \cdots\right)$$

$$\frac{\partial s^{(t)}}{\partial s^{(t-1)}} = \frac{\partial s^{(t)}}{\partial z^{(t)}} \frac{\partial z^{(t)}}{\partial s^{(t-1)}} \tag{3}$$

Issues

Since equation (2) involves the product of many such terms, the parameters may not be updated effectively

- A value for equation (3) less than unity causes the gradient to gradually vanish (or explode when > 1)
- This depends on the range of the activation function's derivative and the parameter values
- Common solutions to these problems are truncated backpropagation and gradient clipping

Long Term Dependencies

Intuition

The state of hidden units contains information about all the previous states, combined with the current input

- As the states are updated recursively, the signal from earlier time-steps gradually disappears (e.g., whiteboard)
- This happens during forward propagation, but also during backpropagation (i.e., vanishing gradient)
- With finite memory and large amounts of information, we need mechanisms to process information **selectively**

Intuition

A recurrent layer can be modified to selectively read, write, and forget the processed information (i.e., previous state, input)

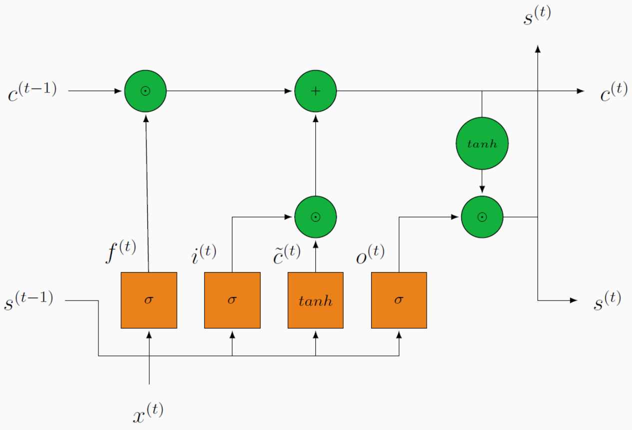
- **Selective read** allows the units to use a fraction of the previous state and input (e.g., pronoun, gender)
- **Selective write** allows units to pass on only the important information in the current state (e.g., not, but)
- **Selective forget** allows units to discard no longer relevant information stored in the state (e.g., a, ".")

Long Short-Term Memory (LSTM)

Long short-term memory (LSTM) networks were originally proposed by Hochreiter & Schmidhuber (1997)

- The cell state retains information across time steps, while selectively updating or forgetting the information
- Multiple parametrized information gates control the flow of information and update the cell state
- LSTMs typically have a forget, input, and output gates, whose parameters are optimized using backpropagation

Long Short-Term Memory (LSTM)



Source: Olah (n.d.). Each quantity is a vector. Orange blocks represent dense network layers, while green nodes represent element-wise operations. Merging links denote concatenation and splitting links indicate copy. Predictions can be made using the current state. This LSTM cell is repeated for each time step.

Long Short-Term Memory (LSTM)

Forget gate	$f^{(t)} = \sigma\left(\left[s^{(t-1)}, x^{(t)}\right] \cdot \beta_f\right)$ $c^{(t)} = f^{(t)} \odot c^{(t-1)} \ldots$
Input gate	$i^{(t)} = \sigma\left(\left[s^{(t-1)}, x^{(t)}\right] \cdot \beta_i\right)$ $\tilde{c}^{(t)} = \tanh\left(\left[s^{(t-1)}, x^{(t)}\right] \cdot \beta_s\right)$ $c^{(t)} = \cdots + i^{(t)} \odot \tilde{c}^{(t)}$
Output gate	$o^{(t)} = \sigma\left(\left[s^{(t-1)}, x^{(t)}\right] \cdot \beta_o\right)$ $s^{(t)} = o^{(t)} \odot \tanh\left(c^{(t)}\right)$

For simplicity, we use [] for concatenation and constants are ignored in $f^{(t)}, i^{(t)}, \tilde{s}^{(t)}$ and $o^{(t)}$. For instance, the equation $f^{(t)} = \left(\left[s^{(t-1)}, x^{(t)}\right] \cdot \beta_f\right)$ should be read as $f^{(t)} = \sigma\left(\left[s^{(t-1)} \cdot \beta_h + b_h\right] + \left[x^{(t)} \cdot \beta_x + b_x\right]\right)$.

Long Short-Term Memory (LSTM)

There are numerous formulations of LSTMs using different gates and arrangements of gates (Greff et al. 2017)

- We studied a popular implementation, but other formulations may perform equally well
- For instance, the Gated Recurrent Unit (GRU) implementation merges the input gate and the forget gate
- There is no explicit supervision for the gates, they enable the model to selectively read, write, and/or forget

Long Short-Term Memory (LSTM)

Regarding optimization, LSTM avoids the vanishing gradient problem as the gates control the flow of information

- During forward propagation, the gates prevent irrelevant information from being written to the state
- The gates control the flow during backpropagation by separating the gradients from the different states
- E.g., the gradient of the previous states that did not contribute to the current one vanishes, but others won't

Summary

Summary

Information encoded as text has many applications in social sciences and economics

- Good approaches should explicitly account for the semantic and the syntactic structure of text data
- Representations for words can be estimated by relating them to their **context** (e.g., SVD, CBOW, Skip-Gram)
- The resulting **embeddings** have interesting properties and are high-quality inputs for the sequent models

Summary

Word2Vec and GloVE are bag-of-words approaches and the **sequence of words** is not taken into account

- These approaches cannot handle words outside the vocabulary (i.e., pre-trained) and are context-independent
- Other embeddings use LSTM (e.g. ELMo, Peters et al. 2018) or Transformer models (e.g., BERT, Devlin et al. 2019)
- **Context-sensitive embeddings** producing different representations for homonyms (e.g., bank, cell)

Summary

Recurrent networks can be used to model data in the form of **sequences with complex interactions across time-steps**

- The original formulation suffers from long-term dependencies and vanishing gradients
- LSTM networks include selective read, write, and forget mechanisms (i.e., GRU, bi-directional LSTMs)
- Current state-of-the-art are **transformer networks** (Vaswani et al. 2017) using attention mechanisms

References

References

- Harris, Zellig S. (1954). "Distributional Structure". In: WORD 10.2, pp. 146-162 (cit. on p. 15).
- Firth, J.R. (1957). "A synopsis of linguistic theory, 1930-1955". In: Studies in Linguistic Analysis. Blackwell, Oxford, pp. 1-32 (cit. on p. 15).
- Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning representations by back-propagating errors". In: Nature 323.6088, pp. 533-536 (cit. on p. 5).
- Bengio, Yoshua et al. (2003). "A neural probabilistic language model". In: Journal of Machine Learning Research 3, pp. 1137-1155 (cit. on p. 23).
- Maaten, Laurens van der and Geoffrey Hinton (2008). "Visualizing data using t-SNE". In: Journal of Machine Learning Research 9, pp. 2579-2606 (cit. on p. 39).
- Maas, Andrew L. et al. (2011). "Learning Word Vectors for Sentiment Analysis". In: Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pp. 142-150 (cit. on p. 43).
- Mikolov, Tomas et al. (2013a). "Distributed representations of words and phrases and their compositionality". In: Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2. Curran Associates Inc., pp. 3111-3119 (cit. on pp. 23, 38).
- Mikolov, Tomas et al. (2013b). "Efficient estimation of word representations in vector space". In: arXiv preprint arXiv:1301.3781 (cit. on p. 23).

References

- Pennington, Jeffrey, Richard Socher, and Christopher Manning (2014). "GloVe: Global Vectors for Word Representation". In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1532-1543 (cit. on p. 23).
- LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton (2015). "Deep learning". In: Nature 521, pp. 436-444 (cit. on p. 3).
- Wattenberg, Martin, Fernanda Viégas, and Ian Johnson (2016). "How to use t-SNE effectively". In: Distill (cit. on p. 39).
- Greff, Klaus et al. (2017). "LSTM: A search space odyssey". In: IEEE Transactions on Neural Networks and Learning Systems 28.10, pp. 2222-2232 (cit. on p. 68).
- Vaswani, Ashish et al. (2017). "Attention is all you need". In: Advances in Neural Information Processing Systems. Vol. 30 (cit. on p. 73).
- McInnes, Leland et al. (2018). "UMAP: Uniform Manifold Approximation and Projection". In: Journal of Open Source Software 3.29, p. 861 (cit. on p. 39).
- Peters, Matthew E. et al. (2018). "Deep contextualized word representations". In: CoRR abs/1802.05365 (cit. on pp. 23, 72).

References

- Devlin, Jacob et al. (2019). "BERT: Pre-training of deep bidirectional transformers for language understanding". In: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics, pp. 4171-4186 (cit. on pp. 23, 72).
- Gentzkow, Matthew, Bryan Kelly, and Matt Taddy (2019). "Text as data". In: Journal of Economic Literature 57.3, pp. 535-574 (cit. on p. 3).
- Olah, Christopher (n.d.). Understanding LSTM networks. (Visited on 2015) (cit. on p. 66).