Aprendizado de Máquina em Linguagem Natural

Beatriz Albiero
https://github.com/beatrizalbiero
Felipe Salvatore
https://felipessalvatore.github.io/

November 13, 2017

USP:University of São Paulo

Aprendizado de máquina

Introdução Aprendizado de Máquina

- ML é realizar uma tarefa baseado em dados.

Fazer desenho das coisas abaixo!!!!!!!!!!!!!!

- **Regressão** $x_1, \ldots, x_N \to y_1, \ldots, y_N$.
- **classificação** $x_1, \ldots, x_N \to L_1, \ldots, l_N$.

Regressão

colocar um plot falso mas verossivel !!!!!!!!!!!!!!!!!!!

Classificação

arrumar flechas !!!!!!!!!!!!!!!!!!!

28×28

$\boldsymbol{x}$

$$\begin{bmatrix} 0.34 \\ \vdots \\ 0.06 \end{bmatrix}$$

784×1

$y = 1$

$\boldsymbol{y}$

$$\begin{bmatrix} 1 \\ \vdots \\ 0 \end{bmatrix}$$

10×1

Aprendizado por Redes Neurais

O que sabemos sobre o cérebro:

Aprendizado por Redes Neurais

O que sabemos sobre o cérebro:

- neurônios em rede

Aprendizado por Redes Neurais

O que sabemos sobre o cérebro:

- neurônios em rede
- neurônios emitem sinais elétricos (disparam)

Aprendizado por Redes Neurais

O que sabemos sobre o cérebro:

- neurônios em rede
- neurônios emitem sinais elétricos (disparam)
- dendritos e axônios

Aprendizado por Redes Neurais

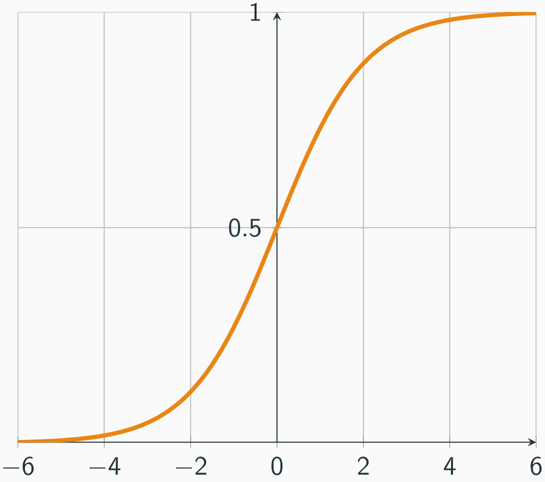
Traduzindo para o modelo de redes neurais artificiais:

- Energia recebida: Input
- Energia enviada: Output
- Carga mínima: Threshold
- Uma função que recebe um input e emite um output mas leva em consideração um threshold mínimo:

Traduzindo para o modelo de redes neurais artificiais:

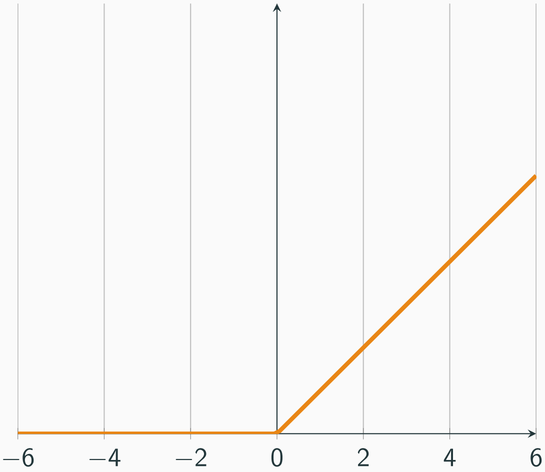
- Energia recebida: Input
- Energia enviada: Output
- Carga mínima: Threshold
- Uma função que recebe um input e emite um output mas leva em consideração um threshold mínimo: Função de ativação

Função de ativação 1: sigmoid

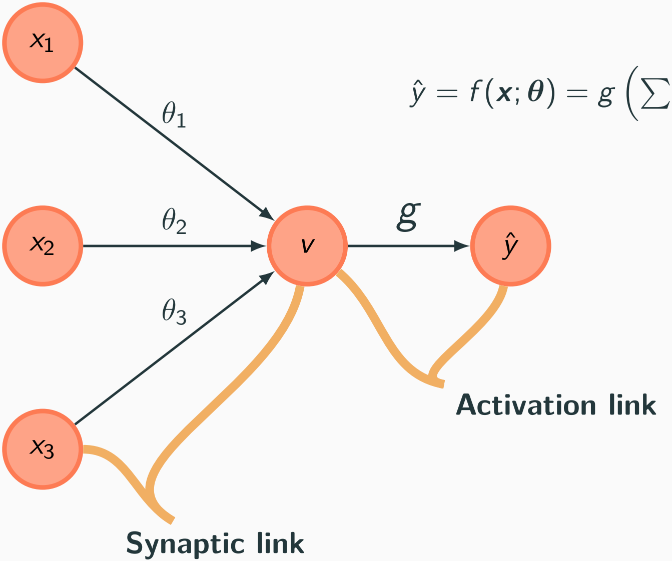


$$\sigma(x) = \frac{1}{1+e^{-x}}$$

função de ativação 2: relu

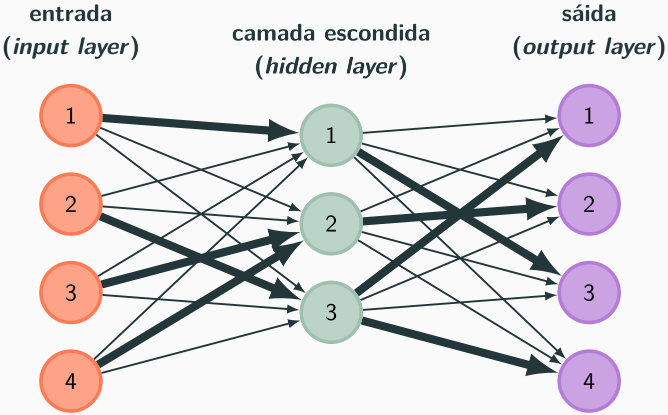


$$g(x) = max\{0, x\}$$

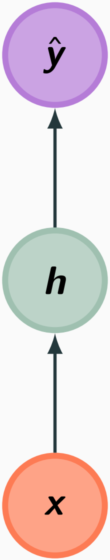


$$\hat{y} = f(\boldsymbol{x};\boldsymbol{\theta}) = g\left(\sum_{i=1}^{3} \theta_i x_i\right)$$

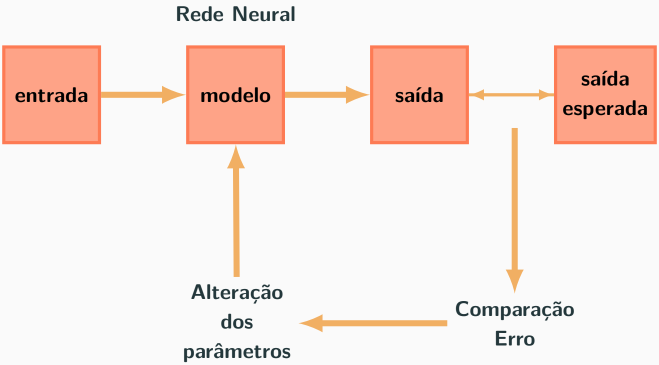
Rede Neural


entrada
(*input layer*)
camada escondida
(*hidden layer*)
sáida
(*output layer*)
1
2
3
4
1
2
3
1
2
3
4

Versão resumida de uma rede neural

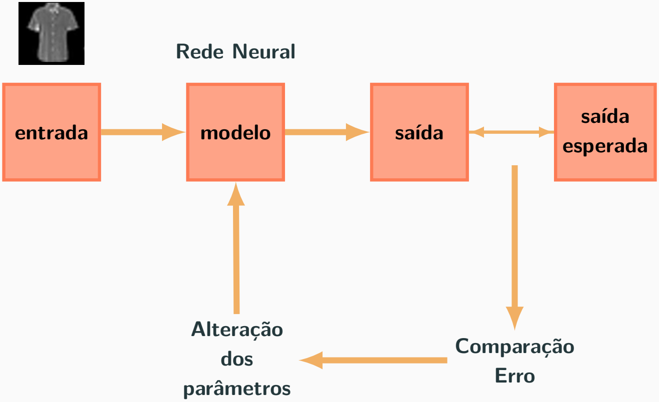


Resumo

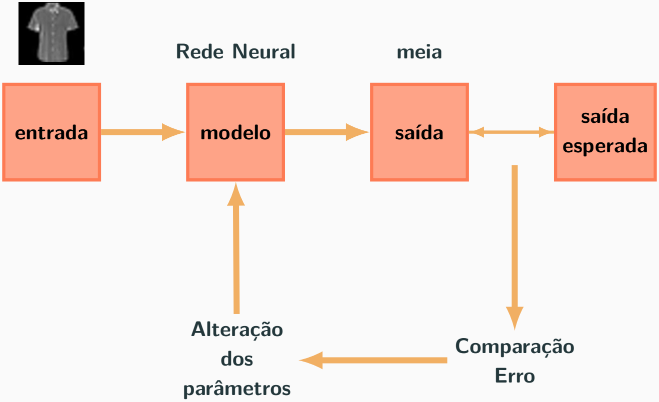


Rede Neural
entrada
modelo
saída
saída esperada
Alteração dos parâmetros
Comparação Erro

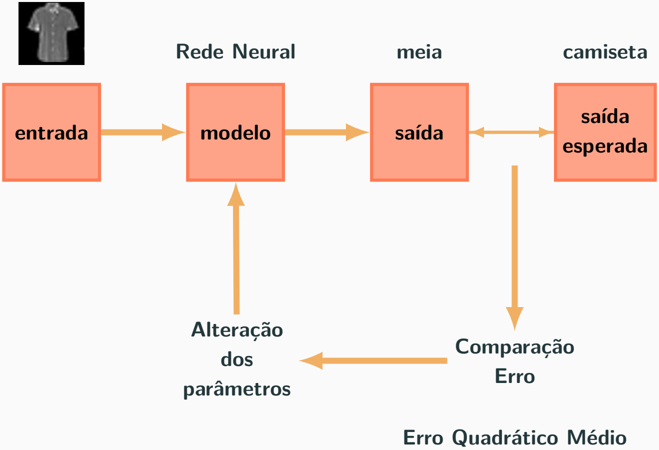
Resumo


Rede Neural
entrada
modelo
saída
saída esperada
Alteração dos parâmetros
Comparação Erro

Resumo

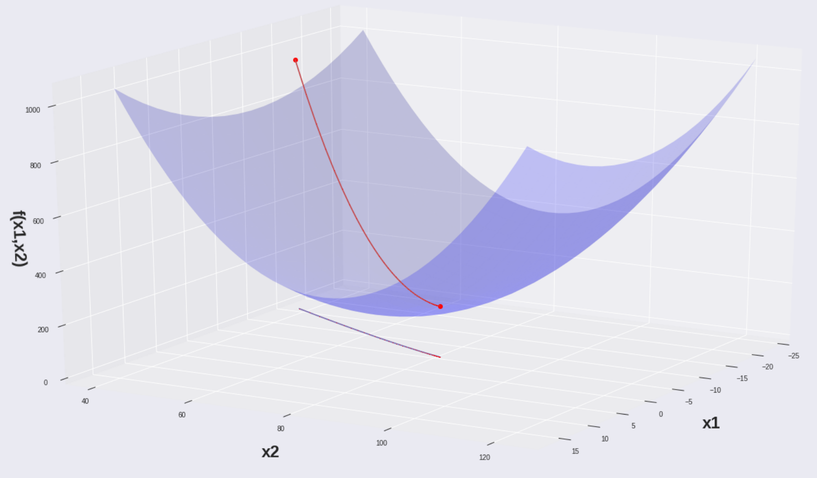

Rede Neural
meia
entrada
modelo
saída
saída esperada
Alteração dos parâmetros
Comparação Erro

Resumo



Rede Neural
meia
camiseta
entrada
modelo
saída
saída esperada
Alteração dos parâmetros
Comparação Erro
Erro Quadrático Médio

Descida do gradiente



Aprendizado de máquina em linguística

On Learning the Past Tenses of English Verbs

Problema:

- Aprendizado dos verbos "irregulares" no passado do inglês

On Learning the Past Tenses of English Verbs

Problema:

- Aprendizado dos verbos "irregulares" no passado do inglês
- Famílias de verbos irregulares:

On Learning the Past Tenses of English Verbs

Problema:

- Aprendizado dos verbos "irregulares" no passado do inglês
- Famílias de verbos irregulares:
    - “blow–blew, grow–grew, know–knew, throw–threw”
    - “bind–bound, find–found, grind–ground, wind–wound”
    - “drink–drank, shrink–shrank, sink–sank, stink–stank”

On Learning the Past Tenses of English Verbs

Problema:

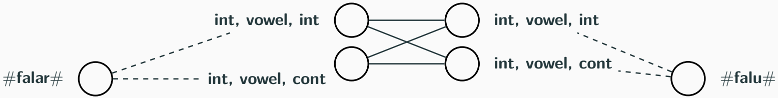
- Aprendizado dos verbos "irregulares" no passado do inglês
- Famílias de verbos irregulares:
  - “blow–blew, grow–grew, know–knew, throw–threw”
  - “bind–bound, find–found, grind–ground, wind–wound”
  - “drink–drank, shrink–shrank, sink–sank, stink–stank”
- Chomsky vs Rumelhart e McClelland

On Learning the Past Tenses of English Verbs

Problema:

- Aprendizado dos verbos "irregulares" no passado do inglês
- Famílias de verbos irregulares:
    - “blow–blew, grow–grew, know–knew, throw–threw”
    - “bind–bound, find–found, grind–ground, wind–wound”
    - “drink–drank, shrink–shrank, sink–sank, stink–stank”
- Chomsky vs Rumelhart e McClelland
- Aprendizado por Regras (Racionalismo) vs Aprendizado por Analogias (Conexionismo)

Rede Neural


#falar#
int, vowel, int
int, vowel, cont
int, vowel, int
int, vowel, cont
#falu#

On Learning the Past Tenses of English Verbs

Exemplo de entrada (x) e saída (y):

$$(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}) = (\text{begin, began}).$$
$$(\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}) = (\text{love, loved})$$
$$(\boldsymbol{x}^{(3)}, \boldsymbol{y}^{(3)}) = (\text{drink, drank})$$
$$(\boldsymbol{x}^{(4)}, \boldsymbol{y}^{(4)}) = (\text{hate, hated})$$
$$(\boldsymbol{x}^{(5)}, \boldsymbol{y}^{(5)}) = (\text{grow, grew})$$
$$(\boldsymbol{x}^{(6)}, \boldsymbol{y}^{(6)}) = (\text{bind, bound})$$
$$(\boldsymbol{x}^{(7)}, \boldsymbol{y}^{(7)}) = (\text{hit, hit})$$
$$\ldots$$

X será uma combinação de traços (features) fonológicos.

Table 1: Bia, lembre de colocar uma legenda

		Place					
		Front		Middle		Back	
		V/L	U/S	V/L	U/S	V/L	U/S
Int.	Stop	b	p	d	t	g	k
	Nasal	m	-	n	-	N	-
Cont	Fric	v/D	f/T	z	s	Z/j	S/C
	Liq/SV	w/l	-	r	-	y	h
Vowel	High	E	i	O	-	U	u
	Low	A	e	I	a/α	W	o

On Learning the Past Tenses of English Verbs

$$\boldsymbol{x} = \begin{bmatrix} 1 \\ 0 \\ \vdots \\ 1 \\ \vdots \\ 0 \end{bmatrix}$$

On Learning the Past Tenses of English Verbs

explicar a saida do modelo e funcao de custo

On Learning the Past Tenses of English Verbs

Resultados (PROS):

On Learning the Past Tenses of English Verbs

Resultados (PROS):

- Identificou padrões corretamente entre todos os 420 verbos do treinamento;

On Learning the Past Tenses of English Verbs

Resultados (PROS):

- Identificou padrões corretamente entre todos os 420 verbos do treinamento;
- Taxa de acerto de 92% para verbos regulares ausentes no treinamento;

On Learning the Past Tenses of English Verbs

Resultados (PROS):

- Identificou padrões corretamente entre todos os 420 verbos do treinamento;
- Taxa de acerto de 92% para verbos regulares ausentes no treinamento;
- Taxa de acerto de 84% para verbos irregulares ausentes no treinamento;

On Learning the Past Tenses of English Verbs

Resultados (PROS):

- Identificou padrões corretamente entre todos os 420 verbos do treinamento;
- Taxa de acerto de 92% para verbos regulares ausentes no treinamento;
- Taxa de acerto de 84% para verbos irregulares ausentes no treinamento;
- U-shaped Development

Resultados (CONS):

Resultados (CONS):

- A rede falha ao tentar fazer predições com palavras que compartilham muitas features em comum;
  Exemplo: "algalgal" - Oykangand

On Learning the Past Tenses of English Verbs

Resultados (CONS):

- A rede falha ao tentar fazer predições com palavras que compartilham muitas features em comum;
  Exemplo: "algalgal" - Oykangand
- Muitos problemas como uma teoria da mente

As Irregularidades no Português Brasileiro

Desafios:

As Irregularidades no Português Brasileiro

Desafios:

- "Wug Test";
  Exemplos: "poguir", "redir", "atover"

As Irregularidades no Português Brasileiro

Desafios:

- "Wug Test";
  Exemplos: "poguir", "redir", "atover"
- Adaptar a rede para a língua portuguesa

As Irregularidades no Português Brasileiro

Desafios:

- "Wug Test";
  Exemplos: "poguir", "redir", "atover"
- Adaptar a rede para a língua portuguesa
- Melhorar o desempenho da rede

Modelos de linguagem e redes recorrentes

Definition

We call language model a probability distribution over sequences of tokens in a natural language.

$$P(x_1, x_2, x_3, x_4) = p$$

Used for:

- speech recognition
- machine translation
- text auto-completion
- spell correction
- question answering
- summarization

revisao probabilidade

independencia, prop condicional

How do we build these probabilities?

Using the chain rule of probability:

$$P(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_1x_2)P(x_4|x_1x_2x_3)$$

To make things simple we use a **Markovian assumption**, i.e., for a specific n we assume that:

$$P(x_1, \ldots, x_T) = \prod_{t=1}^{T} P(x_t|x_1, \ldots, x_{t-1}) = \prod_{t=1}^{T} P(x_t|x_{t-(n+1)}, \ldots, x_{t-1})$$

Models based on n-gram statistics

The choice of n yields different models.

Unigram language model ($n = 1$):

$$P_{uni}(x_1, x_2, x_3, x_4) = P(x_1)P(x_2)P(x_3)P(x_4)$$

where $P(x_i) = count(x_i)$.

Bigram language model ($n = 2$):

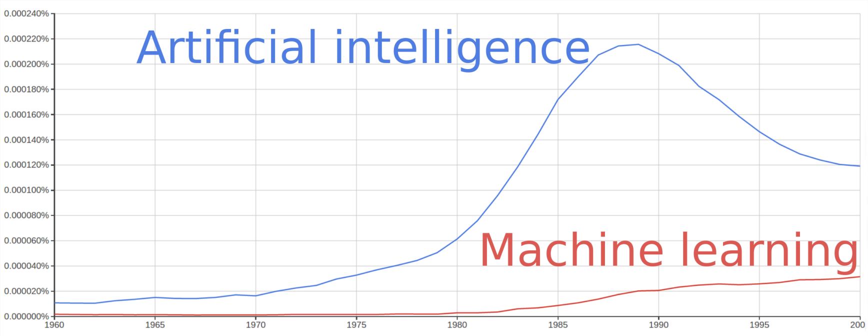
$$P_{bi}(x_1, x_2, x_3, x_4) = P(x_1)P(x_2|x_1)P(x_3|x_2)P(x_4|x_3)$$

where

$$P(x_i|x_j) = \frac{count(x_i, x_j)}{count(x_j)}$$

n-gram statistics

`https://books.google.com/ngrams`



Models based on *n*-gram statistics

- Higher *n*-grams yields better performance.
- Higher *n*-grams requires a lot of memory!

> *"Using one machine* ***with 140 GB RAM for 2.8 days****, we built an unpruned model on 126 billion tokens."*
>
> *Scalable Modified Kneser-Ney Language Model Estimation* by Heafield et al.

Language model as sequential data prediction

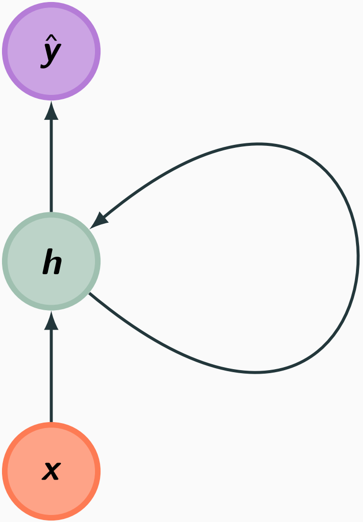
Instead of using one approach that is specific for the language domain, we can use a general model for sequential data prediction: a **RNN**.

Our learning task is to estimate the probability distribution

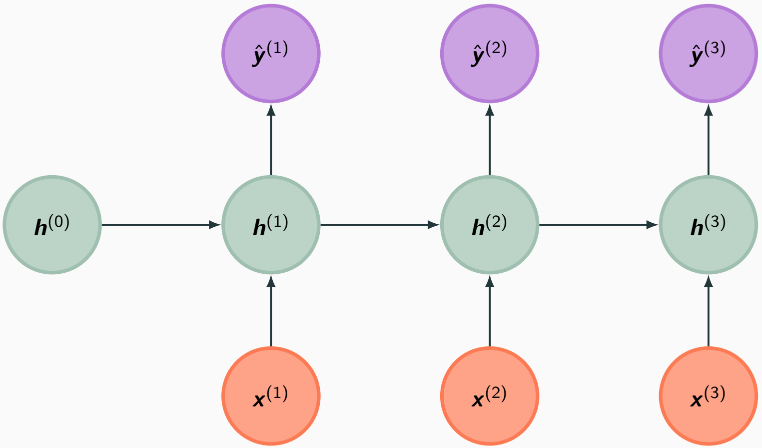
$$P(x_n = \text{word}_{j^*} | x_1, \ldots, x_{n-1})$$

for any $(n-1)$-sequence of words $x_1, \ldots, x_{n-1}$.

RNNs



RNNs



$\hat{\boldsymbol{y}}^{(1)}$
$\hat{\boldsymbol{y}}^{(2)}$
$\hat{\boldsymbol{y}}^{(3)}$
$\boldsymbol{h}^{(0)}$
$\boldsymbol{h}^{(1)}$
$\boldsymbol{h}^{(2)}$
$\boldsymbol{h}^{(3)}$
$\boldsymbol{x}^{(1)}$
$\boldsymbol{x}^{(2)}$
$\boldsymbol{x}^{(3)}$

Classificação com uma rede neural

$$\underset{784 \times 1}{\overset{\boldsymbol{x}}{\begin{bmatrix} 0.34 \\ \vdots \\ 0.06 \end{bmatrix}}} \longrightarrow \textbf{DFN} \longrightarrow \underset{10 \times 1}{\overset{\hat{\boldsymbol{y}}}{\begin{bmatrix} p(y = 1 | \boldsymbol{x}; \boldsymbol{\theta}) \\ \vdots \\ p(y = 10 | \boldsymbol{x}; \boldsymbol{\theta}) \end{bmatrix}}}$$

Building the dataset

We start with a corpus C with T tokens and a vocabulary $\mathbb{V}$.

Example: **Make Some Noise** by the Beastie Boys.

Yes, here we go again, give you more, nothing lesser
Back on the mic is the anti-depressor
Ad-Rock, the pressure, yes, we need this
The best is yet to come, and yes, believe this
...

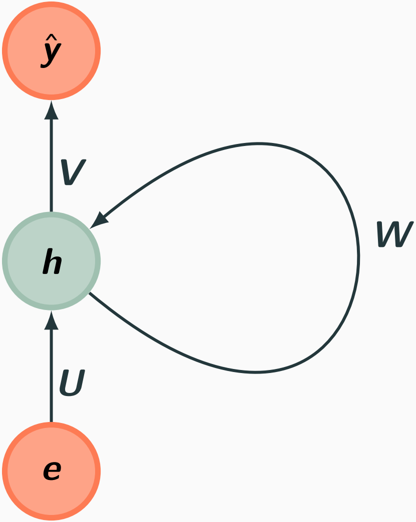
- $T = 378$
- $|\mathbb{V}| = 186$

Building the dataset

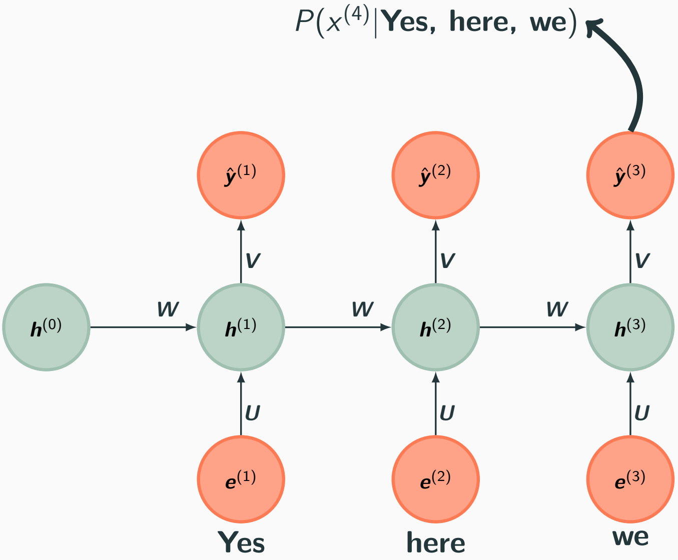
The dataset is a collection of pairs $(\boldsymbol{x}, \boldsymbol{y})$ where $\boldsymbol{x}$ is one word and $\boldsymbol{y}$ is the immediately next word. For example:

$$
\begin{aligned}
(\boldsymbol{x}^{(1)}, \boldsymbol{y}^{(1)}) &= (\text{Yes, here}). \\
(\boldsymbol{x}^{(2)}, \boldsymbol{y}^{(2)}) &= (\text{here, we}) \\
(\boldsymbol{x}^{(3)}, \boldsymbol{y}^{(3)}) &= (\text{we, go}) \\
(\boldsymbol{x}^{(4)}, \boldsymbol{y}^{(4)}) &= (\text{go, again}) \\
(\boldsymbol{x}^{(5)}, \boldsymbol{y}^{(5)}) &= (\text{again, give}) \\
(\boldsymbol{x}^{(6)}, \boldsymbol{y}^{(6)}) &= (\text{give, you}) \\
(\boldsymbol{x}^{(7)}, \boldsymbol{y}^{(7)}) &= (\text{you, more}) \\
&\ldots
\end{aligned}
$$

The language model: graph



The Language model: unfolding example



Loss and Perplexity

So another definition of perplexity is

$$2^L = PP(C)$$

Vanishing

If we initialize $\boldsymbol{W}$ such that $||\boldsymbol{W}|| < 1$, the gradient for further time steps will be very small (vanishing problem).

`https://www.youtube.com/watch?v=xAl8fu8myW0`

Exploding

If $||\boldsymbol{W}|| > 1$, the gradient for further time steps will be larger and larger (exploding problem).

https://www.youtube.com/watch?v=dqW-jw5qKK8

The vanishing problem

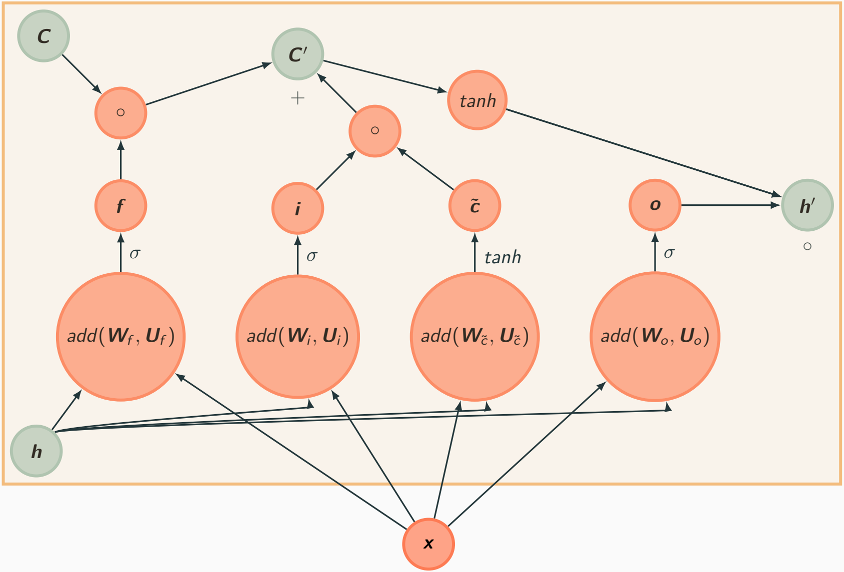
The gradients from the steps closed to τ (the last step) have more influence than the ones very far back.

This is bad for capturing long-term dependecies.

Possible solutions (hacks)

- Clip gradients to a maximum value.
- Choosing the right activation functions, e.g. ReLU.
- Initialize weights to the identity matrix.
- LSTM (Long Short-Term Memory), GRU (Gated Recurrent Unit), etc

LSTM: recurrence



MytwitterBot: TrumpBot

https://github.com/felipessalvatore/MyTwitterBot

Felipe Salvatore
@Felipessalvador

Hillary can make america great again. @greta @MarkBurnettTV #DinheiroNãoCompra #SecretBallot #خسوف_القمر

Traduzir do inglês

15:10 - 7 de ago de 2017

Felipe Salvatore
@Felipessalvador

Obama is all beautiful. I agree with people attacking me. Amazing. @CLewandowski_ #SecretBallot @garyplayer @greta

Traduzir do inglês

14:40 - 7 de ago de 2017

MytwitterBot: SakaBot

https://github.com/felipessalvatore/MyTwitterBot

Felipe Salvatore
@Felipessalvador

Eduardo Cunha deve ser denunciado pelos frigoríficos ainda. Podem apostar no máximo #AGoodDayIncludes #لعبه_مريم

10:19 - 7 de ago de 2017

Felipe Salvatore
@Felipessalvador

Neymar é na verdade algo que o cara vomitou na rua. Lá ele se torna mais rico #WannaOneDebut #العيسي_للطلاب_اشتكوني_للمظالم

09:19 - 7 de ago de 2017

Conclusion

What's next?

After some experiments with the hyper parameters my best result on the Penn Treebank (PTB) corpus was

Model	Val	Test
Mikolov et al (2011)[2]	163.2	149.9

Novas arquiteturas: `https://arxiv.org/abs/1708.02182`

Richard
@RichardSocher

Seguindo

When Zoph & Le at Google got 62 perplexity on PTB, I thought it'd be impossible to beat. Amazing progress in AI atm.
arxiv.org/abs/1708.02182

Traduzir do inglês

Model results over Penn Treebank (PTB)	Params	Val	Test
Grave et al. (2016) - LSTM	–	–	82.3
Grave et al. (2016) - LSTM + continuous cache pointer	–	–	72.1
Inan et al. (2016) - Variational LSTM (tied) + augmented loss	24M	75.7	73.2
Inan et al. (2016) - Variational LSTM (tied) + augmented loss	51M	71.1	68.5
Zilly et al. (2016) - Variational RHN (tied)	23M	67.9	65.4
Zoph & Le (2016) - NAS Cell (tied)	25M	–	64.0
Zoph & Le (2016) - NAS Cell (tied)	54M	–	62.4
Melis et al. (2017) - 4-layer skip connection LSTM (tied)	24M	60.9	58.3
AWD-LSTM - 3-layer LSTM (tied)	24M	**60.0**	**57.3**
AWD-LSTM - 3-layer LSTM (tied) + continuous cache pointer	24M	**53.9**	**52.8**

01:47 - 8 de ago de 2017

References I

I. Goodfellow, Y. Bengio, and A. Courville.
***Deep Learning*.**
MIT Press, 2017.

T. Mikolov, S. Kombrink, L. Burget, J. Cernocký, and S. Khudanpur.

Extensions of recurrent neural network language.
IEEE, pages 5528–5531, 2011.