Natural Language Processing

Language models

Based on slides from Michael Collins, Chris Manning and Richard Soccer

Plan

- Problem definition
- Trigram models
- Evaluation
- Estimation
  - Interpolation
  - Discounting

Motivations

- Define a probability distribution over sentences
- Why? $$p(01010 \mid \sim) \propto p(\sim \mid 01010) \cdot p(01010)$$
  - Machine translation
    - P(“high winds”) > P(“large winds”)
  - Spelling correction
    - P(“The office is fifteen minutes from here”) > P(“The office is fifteen minuets from here”)
  - Speech recognition (that’s where it started!)
    - P(“recognize speech”) > P(“wreck a nice beach”)
  - And more!

Motivations

- Techniques will be useful later

Problem definition

- Given a finite vocabulary

$$\mathcal{V} = \{\text{the}, \text{a}, \text{man}, \text{telescope}, \text{Beckham}, \text{two}, \dots\}$$

- We have an infinite language L, which is V^* concatenated with the special symbol STOP

the STOP
a STOP
the fan STOP
the fan saw Beckham STOP
the fan saw saw STOP
the fan saw Beckham play for Real Madrid STOP

Problem definition

- Input: a training set of example sentences
  - Currently: hundreds of billions of words
- Output: a probability distribution p over L

$$\sum_{x \in L} p(x) = 1, p(x) \geq 0 \text{ for all } x \in \mathcal{L}$$

p(“the STOP”) = 10-12
p(“the fan saw Beckham STOP”) = 2 x 10-8
p(“the fan saw saw STOP”) = 10-15

A naive method

- Assume we have N training sentences
- Let $x_1, x_2, \ldots, x_n$ be a sentence, and $c(x_1, x_2, \ldots, x_n)$ be the number of times it appeared in the training data.
- Define a language model:

$$p(x_1, \ldots, x_n) = \frac{c(x_1, \ldots, x_n)}{N}$$

- No generalization!

Markov processes

- Markov processes:
  - Given a sequence of *n* random variables:
  - We want a sequence probability model

$$X_1, X_2, \ldots, X_n,\ n = 100,\ X_i \in \mathcal{V}$$
$$p(X_1 = x_1, X_2 = x_2, \cdots X_n = x_n)$$

  - There are $|V|^n$ possible sequences

First-order Markov process

Chain rule

$$p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) =$$
$$p(X_1 = x_1) \prod_{i=2}^{n} p(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1})$$

Markov assumption

$$p(X_i = x_i \mid X_1 = x_1, \ldots, X_{i-1} = x_{i-1}) = p(X_i = x_i \mid X_{i-1} = x_{i-1})$$

Second-order Markov process

Relax independence assumption:

$$p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) =$$
$$p(X_1 = x_1) \times p(X_2 = x_2 \mid X_1 = x_1)$$
$$\times \prod_{i=3}^{n} p(X_i = x_i \mid X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

Simplify notation

$$x_0 = *, x_{-1} = *$$

Is this reasonable?

Detail: variable length

- Probability distribution over sequences of any length
- Define always X_n=STOP, and obtain a probability distribution over all sequences
- Intuition: at every step you have probability $\boldsymbol{\alpha}_h$ to stop (conditioned on history) and (1-$\boldsymbol{\alpha}_h$) to keep going

$$p(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) =$$

$$\prod_{i=1}^{n} p(X_i = x_i \mid X_{i-2} = x_{i-2}, X_{i-1} = x_{i-1})$$

Trigram language model

- A trigram language model contains
  - A vocabulary V
  - A non negative parameters q(w|u,v) for every trigram, such that

$$w \in \mathcal{V} \cup \{\text{STOP}\},\ u, v \in \mathcal{V} \cup \{*\}$$

- The probability of a sentence $x_1, \ldots, x_n$, where x_n=STOP is

$$p(x_1, \ldots, x_n) = \prod_{i=1}^{n} q(x_i \mid x_{i-1}, x_{i-2})$$

Example

$$
\begin{aligned}
p(\text{the dog barks STOP}) = & q(the \mid *, *) \times \\
& q(dog \mid *, the) \times \\
& q(barks \mid the, dog) \times \\
& q(STOP \mid dog, barks) \times
\end{aligned}
$$

Limitation

- Markovian assumption is false

He is from France, so it makes sense that his first language is…

- We would want to model longer dependencies

Sparseness

- Maximum likelihood for estimating q
  - Let c(w_1, …, w_n) be the number of times that n-gram appears in a corpus

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

- If vocabulary has 20,000 words —> Number of parameters is 8 x 10^{12}!
- Most sentences will have zero or undefined probabilities

Berkeley restaurant project sentences

- can you tell me about any good cantonese restaurants close by
- mid priced that food is what i’m looking for
- tell me about chez pansies
- can you give me a listing of the kinds of food that are available
- i’m looking for a good place to eat breakfast
- when is caffe venezia open during the day

Bigram counts

	i	want	to	eat	chinese	food	lunch	spend
i	5	827	0	9	0	0	0	2
want	2	0	608	1	6	6	5	1
to	2	0	4	686	2	0	6	211
eat	0	0	2	0	16	2	42	0
chinese	1	0	0	0	0	82	1	0
food	15	0	15	0	1	4	0	0
lunch	2	0	0	0	0	1	0	0
spend	1	0	1	0	0	0	0	0

Bigram probabilities

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

What did we learn

- p(English | want) < p(Chinese | want) - people like Chinese stuff more when it comes to this corpus

- p(to | want) = 0.66 - English behaves in a certain way

- p(eat | to) = 0.28 - English behaves in a certain way

Evaluation: perplexity

- Test data: $S = \{s_1, s_2, \ldots, s_M\}$
  - Parameters are not estimated from S
  - A good language model has high p(S) and low perplexity
    - Perplexity is the normalized inverse probability of S

$$p(\mathcal{S}) = \prod_{i=1}^{M} p(s_i)$$

$$\log_2 p(\mathcal{S}) = \sum_{i=1}^{M} \log_2 p(s_i)$$

$$\text{perplexity} = 2^{-l},\ l = \frac{1}{M} \sum_{i=1}^{M} \log_2 p(s_i)$$

- M is the number of words in the corpus

Evaluation: perplexity

- Say we have a vocabulary V and N = |V| + 1 and a trigram model with uniform distribution
  - q(w | u, v) = 1/N.

$$
\begin{aligned}
l &= \frac{1}{M}\sum_{i=1}^{M}\log_2 p(s_i) \\
&= \frac{1}{M}\log\left(\frac{1}{N}\right)^M = \log\frac{1}{N} \\
\text{perplexity} &= 2^{-l} = 2^{\log N} = N
\end{aligned}
$$

- Perplexity is the “effective” vocabulary size.

Typical values of perplexity

- When |V| = 50,000
  - trigram model perplexity: 74 (<< 50000)
  - bigram model: 137
  - unigram model: 955

Evaluation

- Extrinsic evaluations: MT, speech, spelling correction, …

History

- Shannon (1950) estimated the perplexity score that humans get for printed English (we are good!)
  - Test your perplexity

Estimating parameters

- Recall that the number of parameters for a trigram model with |V| = 20,000 is 8 x 10^{12}, leading to zeros and undefined probabilities

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})}$$

Bias-variance tradeoff

- Given a corpus of length M
  - Trigram model:

$$q(w_i \mid w_{i-2}, w_{i-1}) = \frac{c(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-1}, w_i)}$$

  - Bigram model:

$$q(w_i \mid w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

  - Unigram model:

$$q(w_i) = \frac{c(w_i)}{M}$$

Unigram

To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have
Every enter now severally so, let
Hill he late speaks; or! a more to leg less first you enter
Are where exeunt and sighs have rise excellency took of.. Sleep knave we. near; vile like

Bigram

What means, sir. I confess she? then all sorts, he is trim, captain.
Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow.
What we, hath got so she that I rest and sent to scold and nature bankrupt, nor the first gentleman?

Trigram

Sweet prince, Falstaff shall die. Harry of Monmouth's grave.
This shall forbid it should be branded, if renown made it empty.
Indeed the duke; and had a very good friend.
Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done.

Quadrigram

King Henry.What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in;
Will you not tell me who I am?
It cannot be but so.
Indeed the short and the long. Marry, 'tis a noble Lepidus.

Linear interpolation

$$q_{LI}(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 \times q(w_i \mid w_{i-2}, w_{i-1}) + \lambda_2 \times q(w_i \mid w_{i-1}) + \lambda_3 \times q(w_i)$$

$$\lambda_i \geq 0,\ \lambda_1 + \lambda_2 + \lambda_3 = 1$$

- Combine the three models to get all benefits

Linear interpolation

- Need to verify the parameters define a probability distribution

$$\sum_{w \in \mathcal{V}} q_{LI}(w \mid u, v)$$

$$= \sum_{w \in \mathcal{V}} \lambda_1 \times q(w \mid u, v) + \lambda_2 \times q(w \mid v) + \lambda_3 \times q(w)$$

$$= \lambda_1 \sum_{w \in \mathcal{V}} q(w \mid u, v) + \lambda_2 \sum_{w \in \mathcal{V}} q(w \mid v) + \lambda_3 \sum_{w \in \mathcal{V}} \times q(w)$$

$$= \lambda_1 + \lambda_2 + \lambda_3 = 1$$

Estimating coefficients

- Use validation/development set (intro to ML!)
- Partition training data to training (90%?) and dev (10%?) data and optimize the coefficient to minimize the perplexity (the measure we care about!) on the development data

Discounting methods

x	c(x)	$q(w_i \mid w_{i-1})$
the	48	
the, dog	15	15/48
the, woman	11	11/48
the, man	10	10/48
the, park	5	5/48
the, job	2	2/48
the, telescope	1	1/48
the, manual	1	1/48
the, afternoon	1	1/48
the, country	1	1/48
the, street	1	1/48

Low count bigrams have high estimates

Discounting methods

$$c^*(x) = c(x) - 0.5$$

x	c(x)	c*(x)	$q(w_i \mid w_{i-1})$
the	48		
the, dog	15	14.5	14.5/48
the, woman	11	10.5	10.5/48
the, man	10	9.5	9.5/48
the, park	5	4.5	4.5/48
the, job	2	1.5	1.5/48
the, telescope	1	0.5	0.5/48
the, manual	1	0.5	0.5/48
the, afternoon	1	0.5	0.5/48
the, country	1	0.5	0.5/48
the, street	1	0.5	0.5/48

Katz back-off

In a bigram model

$$\mathcal{A}(w_{i-1}) = \{w : c(w_{i-1}, w) > 0\}$$
$$\mathcal{B}(w_{i-1}) = \{w : c(w_{i-1}, w) = 0\}$$

$$q_{\mathrm{BO}}(w_i \mid w_{i-1}) = \begin{cases} \frac{c^*(w_{i-1}, w_i)}{c(w_{i-1})} & w_i \in \mathcal{A}(w_{i-1}) \\ \alpha(w_{i-1}) \frac{q(w_i)}{\sum_{w \in \mathcal{B}(w_{i-1})} q(w)} & w_i \in \mathcal{B}(w_{i-1}) \end{cases}$$

$$\alpha(w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-1})} \frac{c^*(w_{i-1}, w)}{c(w_{i-1})}$$

Katz back-off

In a trigram model

$$\mathcal{A}(w_{i-2}, w_{i-1}) = \{w : c(w_{i-2}, w_{i-1}, w) > 0\}$$
$$\mathcal{B}(w_{i-2}, w_{i-1}) = \{w : c(w_{i-2}, w_{i-1}, w) = 0\}$$

$$q_{\mathrm{BO}}(w_i \mid w_{i-2}, w_{i-1}) = \begin{cases} \frac{c^*(w_{i-2}, w_{i-1}, w_i)}{c(w_{i-2}, w_{i-1})} & w_i \in \mathcal{A}(w_{i-2}, w_{i-1}) \\ \alpha(w_{i-2}, w_{i-1}) \frac{q_{\mathrm{BO}}(w_i | w_{i-1})}{\sum_{w \in \mathcal{B}(w_{i-2}, w_{i-1})} q_{\mathrm{BO}}(w | w_{i-1})} & w_i \in \mathcal{B}(w_{i-2}, w_{i-1}) \end{cases}$$

$$\alpha(w_{i-2}, w_{i-1}) = 1 - \sum_{w \in \mathcal{A}(w_{i-2}, w_{i-1})} \frac{c^*(w_{i-2}, w_{i-1}, w)}{c(w_{i-2}, w_{i-1})}$$

Class 3

Debt 1: Gradient checks

$$\frac{\partial J(\theta)}{\partial \theta} = \lim_{\epsilon \to 0} \frac{J(\theta + \epsilon) - J(\theta - \epsilon)}{2\epsilon}$$

- Compute for every parameters for small epsilon.

Debt 2: matrix factorization

- We are in a sample space of pairs *#(c,o)*
- *#(c) = $\sum_o$ #(c,o)*
- *#(o) = $\sum_c$ #(c,o)*
- *T = $\sum_{c,o}$ #(c,o)*
- Let *m* be the context size, *|D|* be the corpus length, and *c(w)* be the number of times word *w* appears in the corpus

$$p(w) = \frac{c(w)}{|D|} = \frac{m \cdot c(w)}{m \cdot |D|} = \frac{\#(o)}{T}$$

Debt 3: linear interpolation

$$
\begin{aligned}
q_{LI}(w_i \mid w_{i-2}, w_{i-1}) = \lambda_1 &\times q(w_i \mid w_{i-2}, w_{i-1}) \\
+ \lambda_2 &\times q(w_i \mid w_{i-1}) \\
+ \lambda_3 &\times q(w_i)
\end{aligned}
$$

$$
\lambda_i \geq 0, \;\; \lambda_1 + \lambda_2 + \lambda_3 = 1
$$

Debt 3: linear interpolation

$$\Pi(w_{i-2}, w_{i-1}) = \begin{cases} 1 & c(w_{i-2}, w_{i-1}) = 0 \\ 2 & 1 \leq c(w_{i-2}, w_{i-1}) \leq 2 \\ 3 & 3 \leq c(w_{i-2}, w_{i-1}) \leq 10 \\ 4 & \text{otherwise} \end{cases}$$

$$\begin{aligned} q_{LI}(w_i \mid w_{i-2}, w_{i-1}) = {} & \lambda_1^{\Pi(w_{i-2}, w_{i-1})} \times q(w_i \mid w_{i-2}, w_{i-1}) \\ & + \lambda_2^{\Pi(w_{i-2}, w_{i-1})} \times q(w_i \mid w_{i-1}) \\ & + \lambda_3^{\Pi(w_{i-2}, w_{i-1})} \times q(w_i) \end{aligned}$$

$$\lambda_i^{\Pi(w_{i-2}, w_{i-1})} \geq 0$$

$$\lambda_1^{\Pi(w_{i-2}, w_{i-1})} + \lambda_2^{\Pi(w_{i-2}, w_{i-1})} + \lambda_3^{\Pi(w_{i-2}, w_{i-1})} = 1$$

Debt 4: unknown words

- What if we see a completely new words at test time?
  - p(s) = p(S) = 0 —> infinite perplexity
- Solution: create a special token *<unk>*
  - Fix vocabulary *V* and replace any word in the training set not in *V* with *<unk>*
  - Train
  - At test time, use p(<unk>) for words not in the training set

Administration

- Projects
  - word vectors
  - language models
- Late HW
  - you can have 5 late days during the semester. Afterwards it is 5 points deduction per day.

Summary and re-cap

- Sentence probability: decompose with the chain rule
- Use a Markov independence assumption
- Smooth estimates to do better on rare events
- Many attempts to improve LMs using syntax but not easy
- More complex smoothing methods exist for handling rare events (Kneser-Ney, Good-Turing…)

Problem

- Our estimator q(w | u, v) is based on a one-hot representation. There is no relation between words.
  - p(played | the, actress)
  - p(played | the actor)
- Can we use distributed representations?

Neural networks

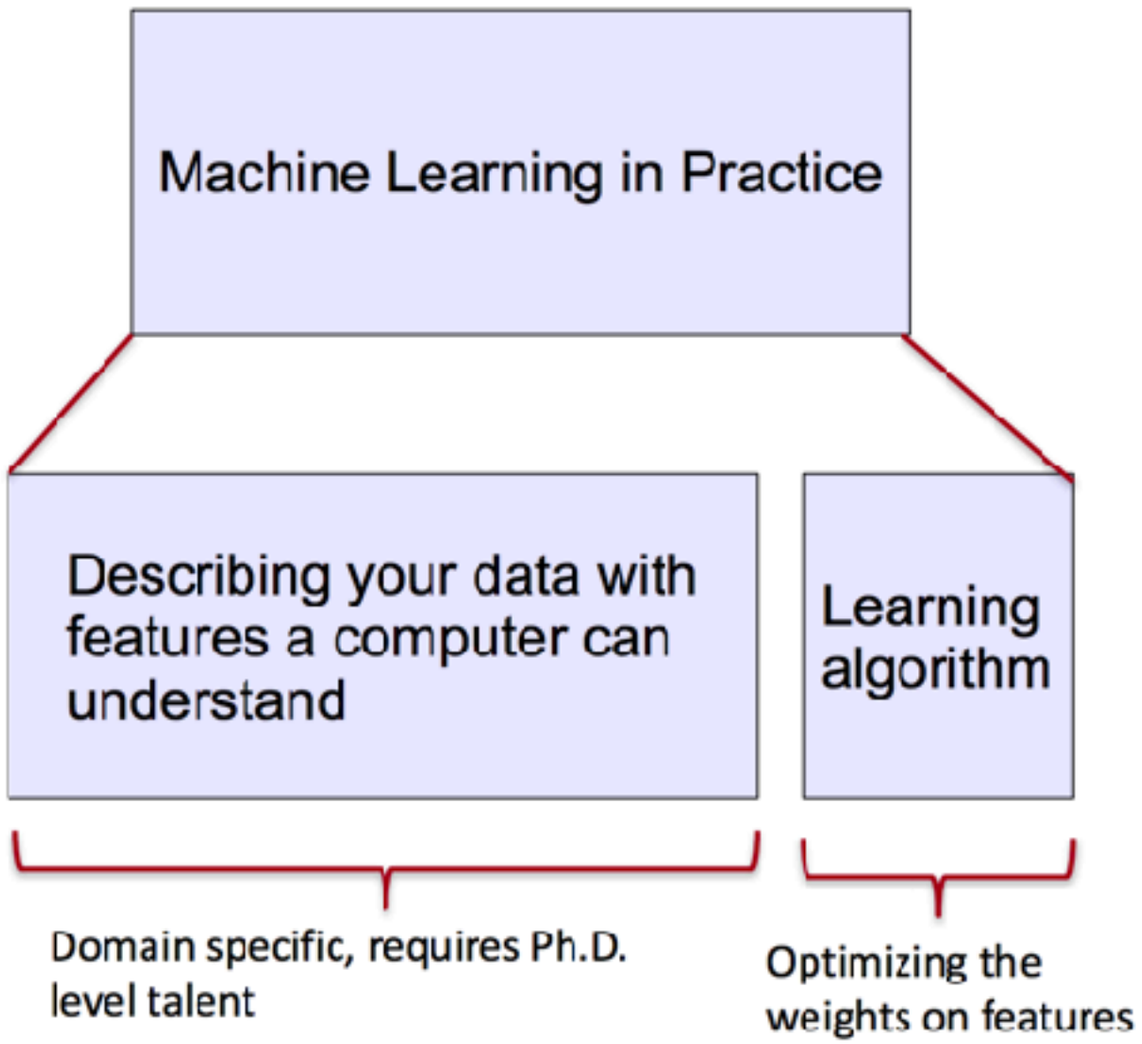
Plan for today

- Feed-forward neural networks for language modeling
- Recurrent neural networks for language modeling
- Vanishing/exploding gradients
- Computation graphs

Neural networks: history

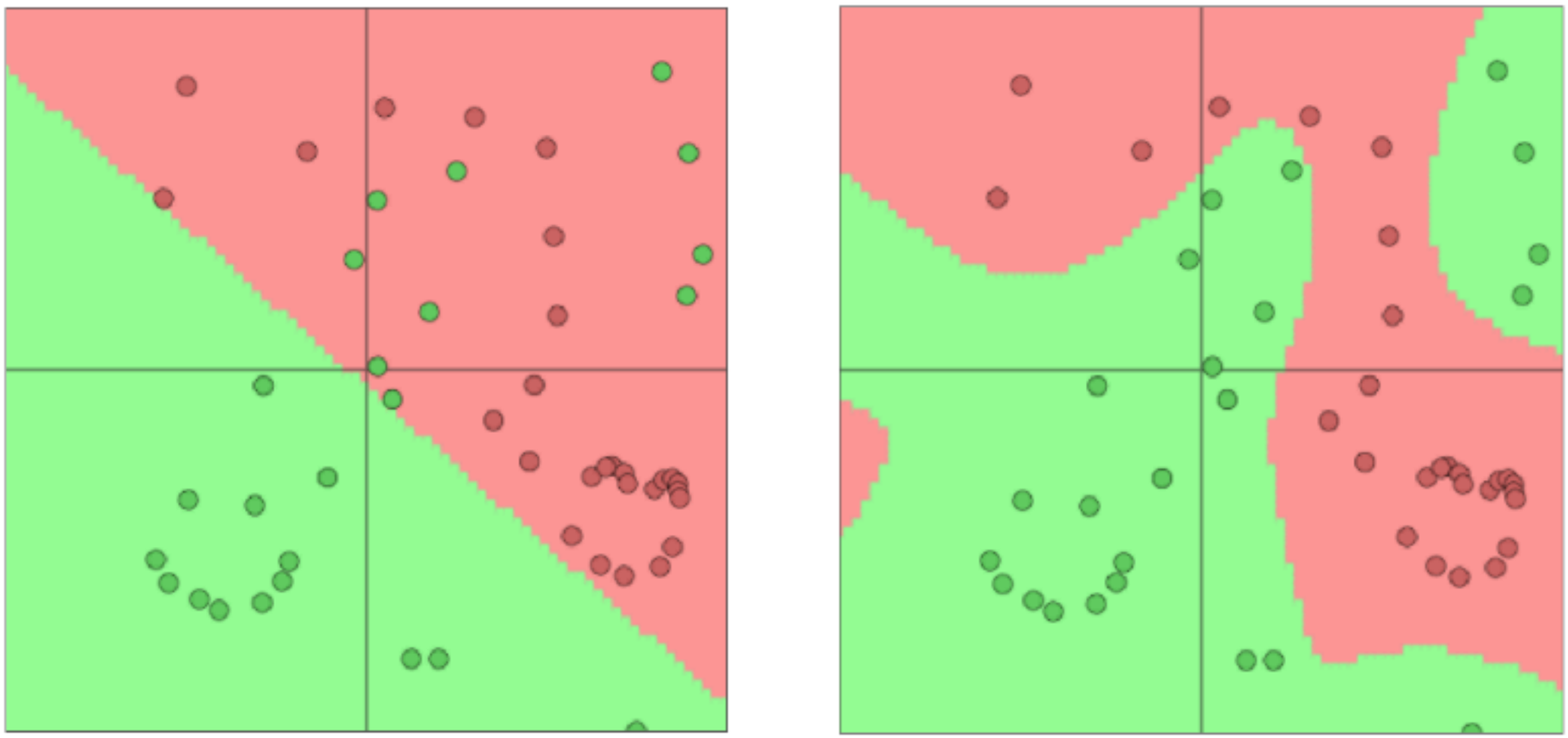
- Proposed in the mid-20th century, advanced by back propagation in the 1980s
- In the 1990s SVMs appeared and were more successful
- Since the beginning of this decade show great success in speech, vision, and language

Motivation 1



Can we learn representations from raw data?

Motivation 2



Can we learn non-linear decision boundaries

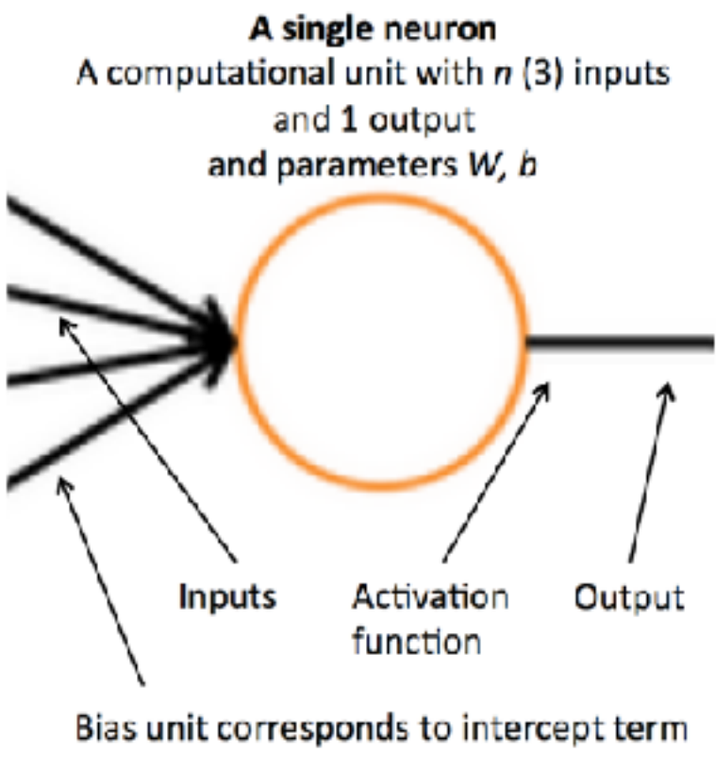
Actually very similar to motivation 1

A single neuron

- A neuron is a computational unit of the form

$$f_{w,b}(x) = f(w^\top x + b)$$

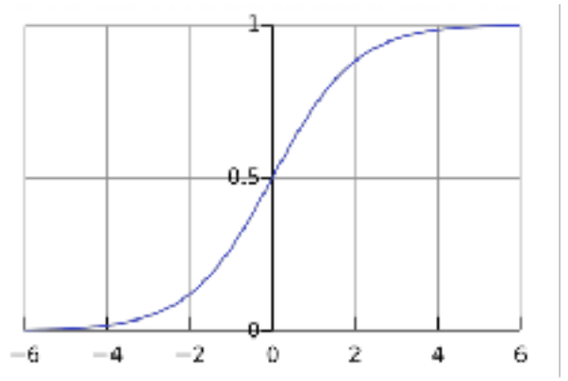
- x: input vector
- w: weights vector
- b: bias
- f: activation function



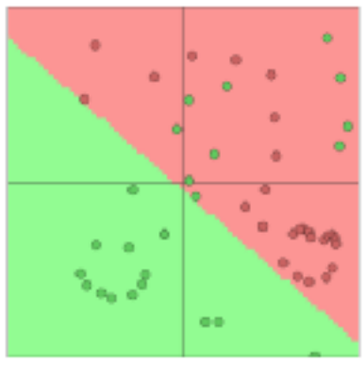
A single neuron

- If *f* is sigmoid, a neuron is logistic regression
- Let x, y be a binary classification training example

$$p(y = 1 \mid x) = \frac{1}{1 + e^{-w^\top x - b}} = \sigma(w^\top x + b)$$



Provides a linear decision boundary

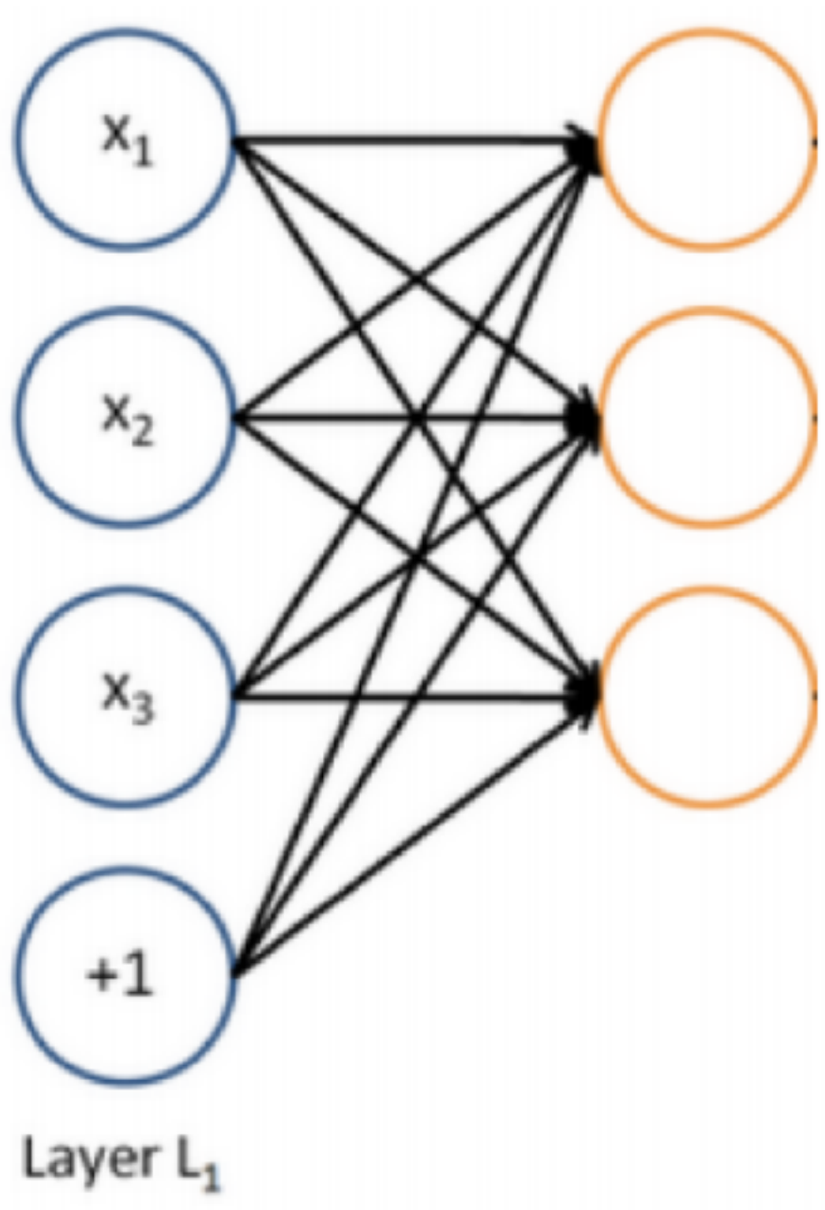


Single layer network

- Perform logistic regression in parallel multiple times (with different parameters)

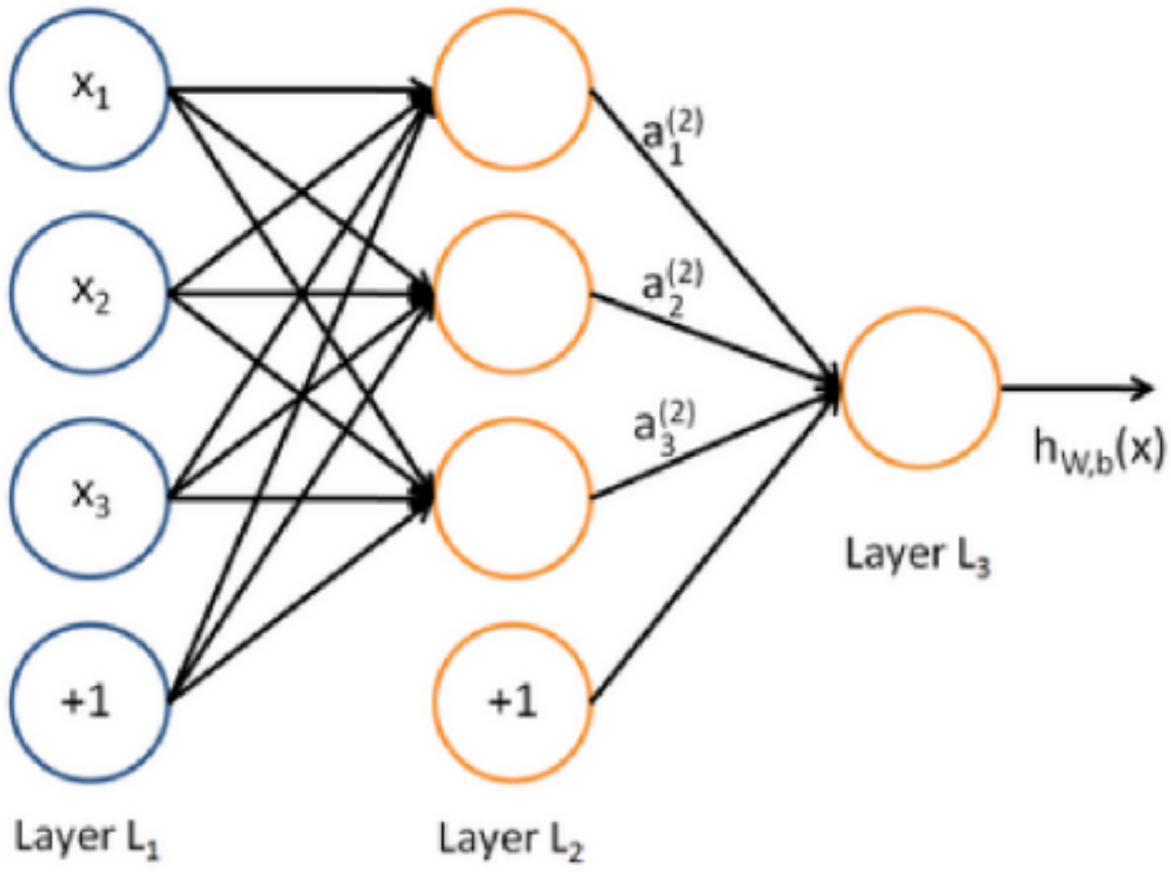
$$y = \sigma(\hat{w}^\top \hat{x} + b) = \sigma(w^\top x)$$

$$x, w \in \mathbb{R}^{d+1},\ w_{d+1} = 1,\ x_{d+1} = b$$



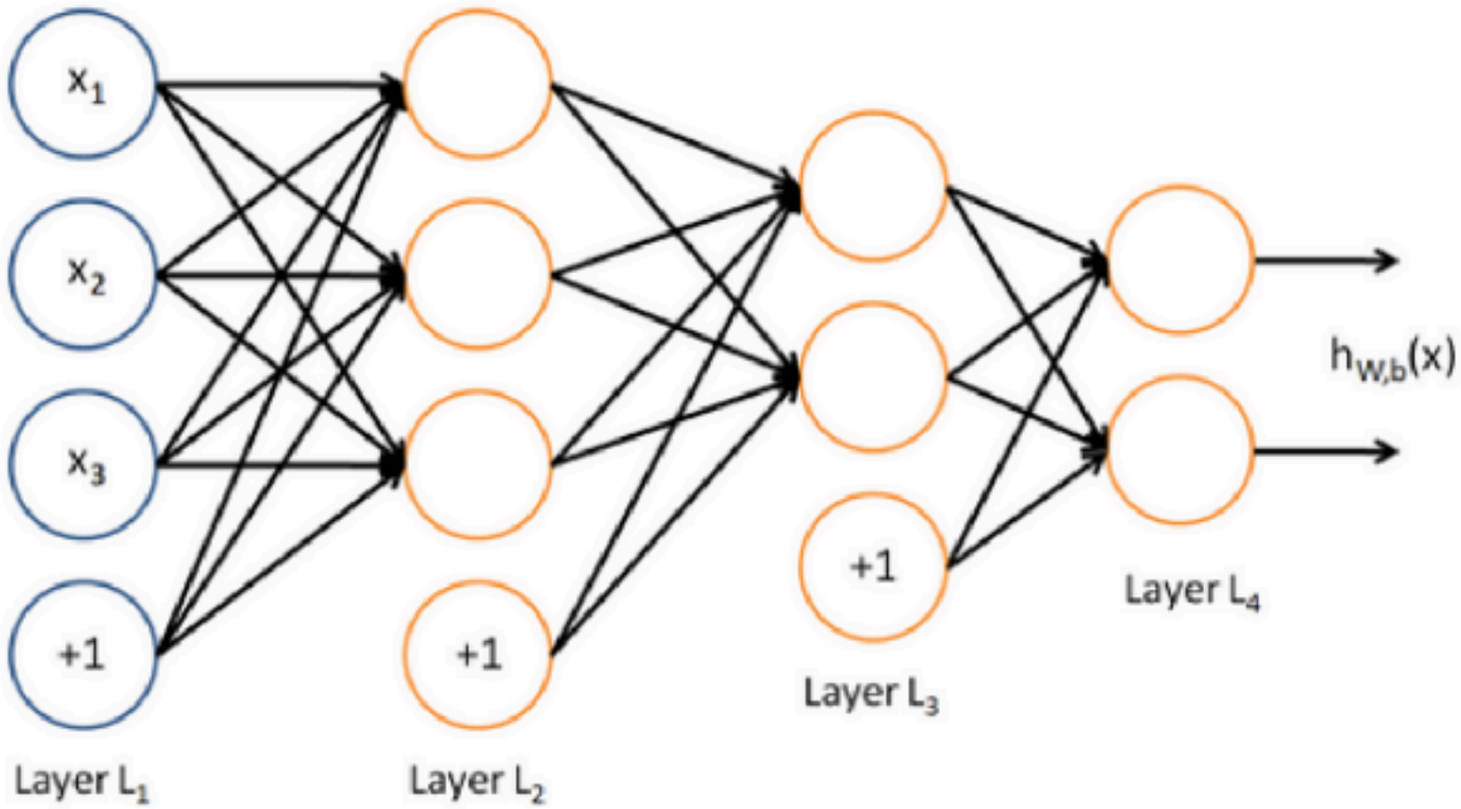
Single layer network

- L1: input layer
- L2: hidden layer
- L3: output layer
- Output layer provides the prediction
- Hidden layer is the **learned representation**



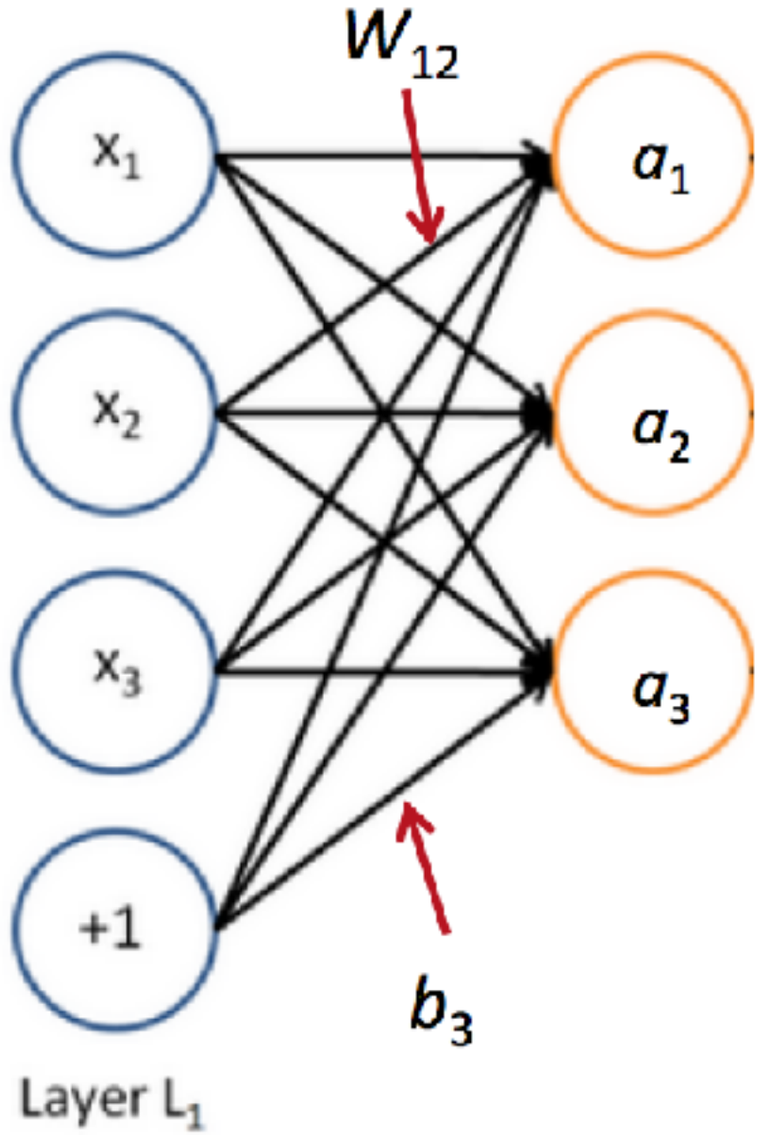
Multi-layer network

Repeat:



Matrix notation

$$a_1 = f(W_{11}x_1 + W_{12}x_2 + W_{13}x_3 + b_1)$$
$$a_2 = f(W_{21}x_1 + W_{22}x_2 + W_{23}x_3 + b_2)$$
$$a_3 = f(W_{31}x_1 + W_{32}x_2 + W_{33}x_3 + b_3)$$
$$z = Wx + b,\ a = f(z)$$
$$x \in \mathbb{R}^4,\ z \in \mathbb{R}^3,\ W \in \mathbb{R}^{3\times 4}$$



Language modeling with NNs

- Keep the Markov assumption

- Learn a probability q(u | v, w) with distributed representations

Bengio et al, 2003

Language modeling with NNs

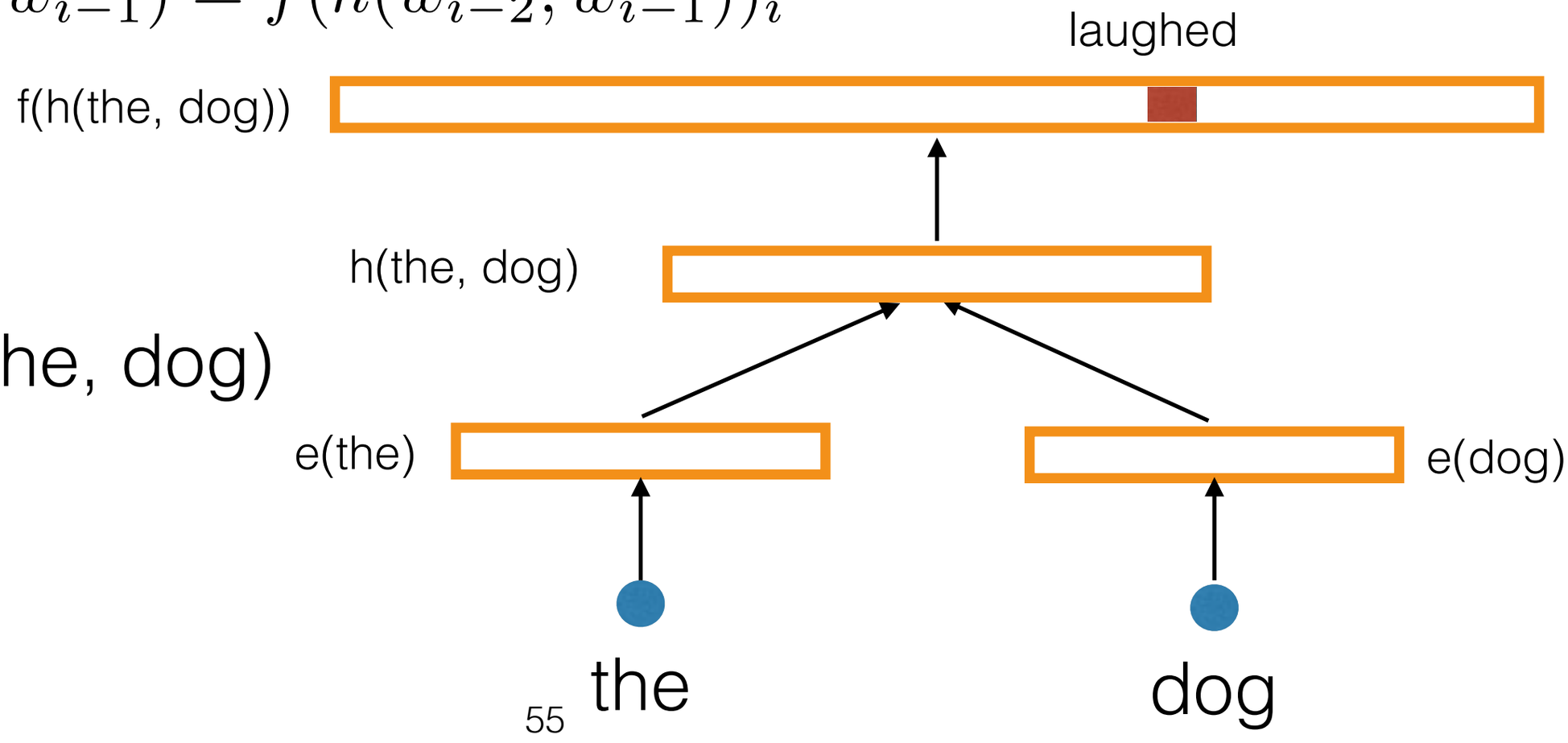
$$e(w) = W_e \cdot w, \ , W_e \in \mathbb{R}^{d \times |\mathcal{V}|}, w \in \mathbb{R}^{|\mathcal{V}| \times 1}$$

$$h(w_{i-2}, w_{i-1}) = \sigma(W_h[e(w_{i-2}); e(w_{i-2})]), \ W_h \in \mathbb{R}^{m \times 2d}$$

$$f(z) = \text{softmax}(W_o \cdot z), \ W_o \in \mathbb{R}^{|\mathcal{V}| \times m}, z \in \mathbb{R}^{m \times 1}$$

$$p(w_i \mid w_{i-2}, w_{i-1}) = f(h(w_{i-2}, w_{i-1}))_i$$

p(laughed | the, dog)



Loss function

- Minimize negative log-likelihood

    - When we learned word vectors, we were basically training a language model

Advantages

- If we see in the training set

The cat is walking in the bedroom

- We can hope to learn

A dog was running through a room

- We can use n-grams with n > 3 and pay only a linear price
  - Compared to what?

Parameter estimation

- We train with SGD
  - How to efficiently compute the gradients?
    - **backpropagation** (Rumelhart, Hinton and Williams, 1986)
    - Proof in intro to ML, will repeat algorithm only and give an example here

Backpropagation

- Notation:

$$W_t : \text{weight matrix at the input of layer } t$$

$$z_t : \text{ output vector at layer } t$$

$$x = z_0 : \text{ input vector}$$

$$y : \text{ gold scalar}$$

$$\hat{y} = z_L : \text{ predicted scalar}$$

$$l(y, \hat{y}) : \text{ loss function}$$

$$v_t = W_t \cdot z_{t-1} : \text{ pre-activations}$$

$$\delta_t = \frac{\partial l(y, z_t)}{\partial z_t} : \text{ gradient vector}$$

Backpropagation

- Run the network forward to obtain all values v_t, z_t
- Base:

$$\delta_L = l'(y, z_L)$$

- Recursion:

$$\delta_t = W_{t+1}^\top(\sigma'(v_{t+1}) \circ \delta_{t+1})$$

$$\sigma'(v_{t+1}), \delta_{t+1} \in \mathbb{R}^{d_{t+1} \times 1}, W_{t+1} \in \mathbb{R}^{d_{t+1} \times d_t}$$

- Gradients:

$$\frac{\partial l}{\partial W_t} = (\delta_t \circ \sigma'(v_t))z_{t-1}^\top$$

Bigram LM example

- Forward pass:

$$z_0 \in \mathbb{R}^{|\mathcal{V}| \times 1} : \text{one-hot vector input}$$

$$z_1 = W_1 \cdot z_0, W_1 \in \mathbb{R}^{d_1 \times |\mathcal{V}|}, z_1 \in \mathbb{R}^{d_1 \times 1}$$

$$z_2 = \sigma(W_2 \cdot z_1), W_2 \in \mathbb{R}^{d_2 \times d_1}, z_2 \in \mathbb{R}^{d_2 \times 1}$$

$$z_3 = \text{softmax}(W_3 \cdot z_2), W_3 \in \mathbb{R}^{|\mathcal{V}| \times d_2}, z_3 \in \mathbb{R}^{|\mathcal{V}| \times 1}$$

$$l(y, z_3) = \sum_i y^{(i)} \log z_3^{(i)}$$

Bigram LM example

- Backward pass:

$$\sigma'(v_3) \circ \delta_3 = (z_3 - y)$$

$$\delta_2 = W_3^\top (\sigma'(v_3) \circ \delta_3) = W_3^\top (z_3 - y)$$

$$\delta_1 = W_2^\top (\sigma'(v_2) \circ \delta_2) = W_2^\top (z_2 \circ (1 - z_2) \circ \delta_2)$$

$$\frac{\partial l}{\partial W_3} = (\delta_3 \circ \sigma'(v_3)) z_2^\top = (z_3 - y) z_2^\top$$

$$\frac{\partial l}{\partial W_2} = (\delta_2 \circ \sigma'(v_2)) z_1^\top = (\delta_2 \circ z_2 \circ (1 - z_2)) z_1^\top$$

$$\frac{\partial l}{\partial W_1} = (\delta_1 \circ \sigma'(v_1)) z_0^\top = \delta_1 z_0^\top$$

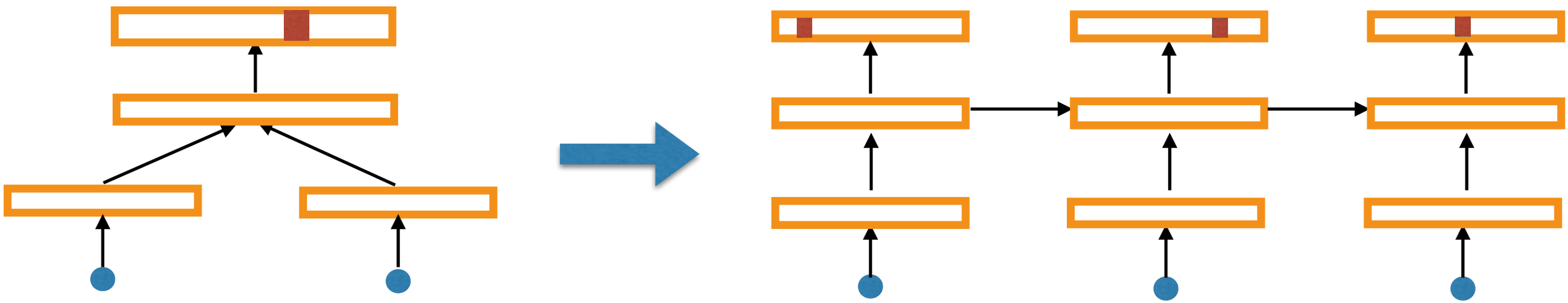
Summary

- Neural nets can improve language models:
  - better scalability for larger N
  - use of word similarity
  - complex decision boundaries
- Training through backpropagation

But we still have a Markov assumption

He is from France, so it makes sense that his first language is…

Recurrent neural networks



$$\text{Input: } w_1, \ldots, w_{t-1}, w_t, w_{t+1}, \ldots, w_T \quad , w_i \in \mathbb{R}^{\mathcal{V}}$$

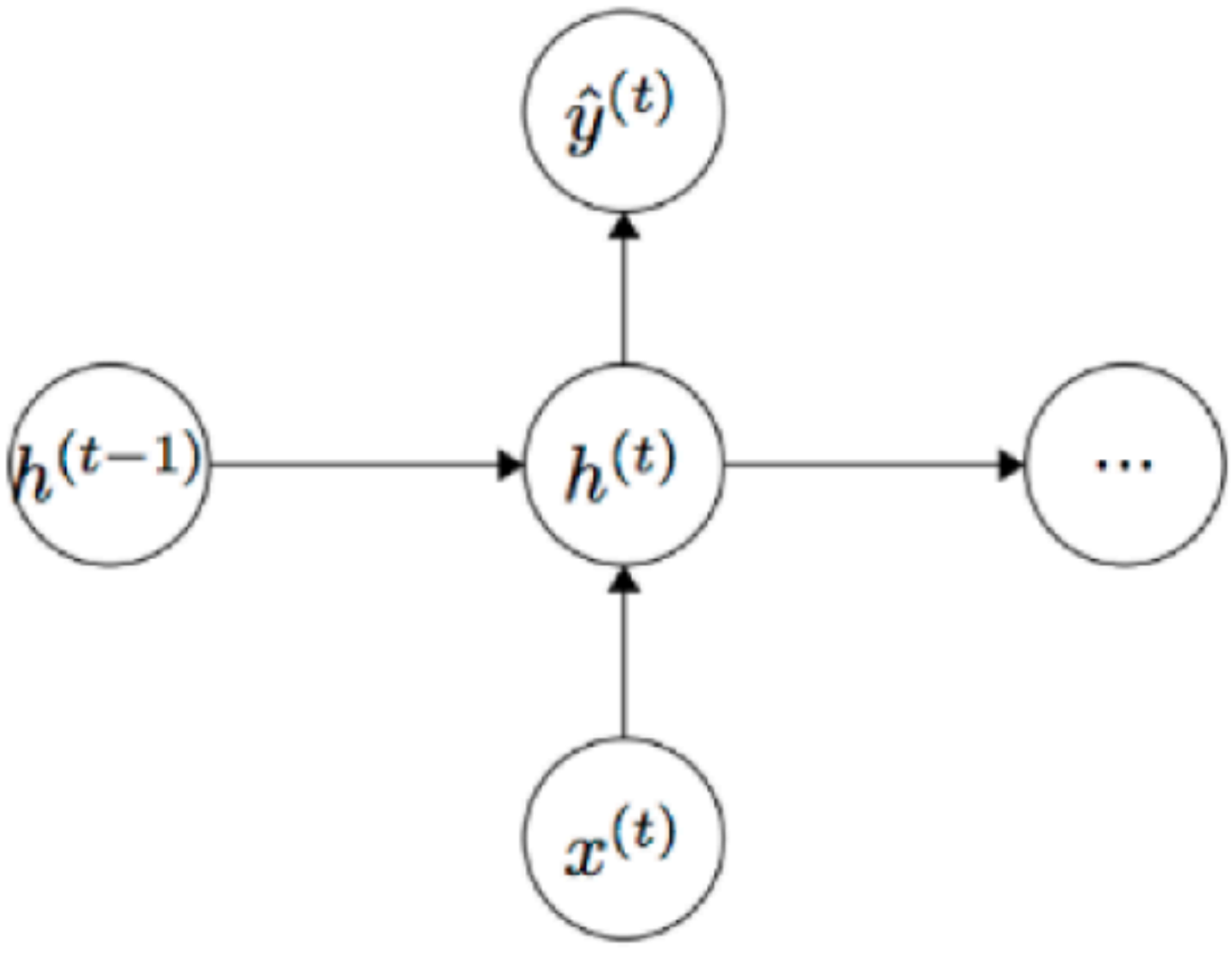
$$\text{Model: } x_t = W^{(e)} \cdot w_t \quad , W^{(e)} \in \mathbb{R}^{d \times \mathcal{V}}$$

$$h_t = \sigma(W^{(hh)} \cdot h_{t-1} + W^{(hx)} \cdot x_t), W^{(hh)} \in \mathbb{R}^{D_h \times D_h}, W^{(hx)} \in \mathbb{R}^{D_h \times d}$$

$$\hat{y}_t = \text{softmax}(W^{(s)} \cdot h_t) \quad , W^{(s)} \in \mathbb{R}^{\mathcal{V} \times D_h}$$

64

Recurrent neural networks



Recurrent neural networks

- Can exploit long range dependencies
- Each layer has the same weights (weight sharing/ tying)
- What is the loss function?

$$J(\theta) = \sum_{t=1}^{T} \mathrm{CE}(y_t, \hat{y}_t)$$

Recurrent neural networks

- Component:
    - model? RNN (saw before)
    - Loss? sum of cross-entropy over time
    - Optimization? SGD
    - Gradient computation? back-propagation through time

Training RNNs

- Capturing long-range dependencies with RNNs is difficult
- Vanishing/exploding gradients
- Small changes in hidden layer value in step *k* cause huge/minuscule changes to values in hidden layer *t* for *t* >> *k*

Explanation

Consider a simple linear RNN with no input:

$$h_t = W \cdot h_{t-1}$$

$$h_t = W^t \cdot h_0$$

$$h_t = (Q\Lambda Q^{-1})^t \cdot h_0$$

$$h_t = (Q\Lambda^t Q^{-1}) \cdot h_0$$

$$\text{where } W = Q\Lambda Q^{-1} \text{ is an eigendecomposition}$$

- Some eigenvalues will explode and some will shrink to zero
  - Stretch input in the direction of eigenvector with largest eigenvalue

Pascanu et al, 2013

Explanation

$$h_t = W^{(hh)}\sigma(h_{t-1}) + W^{(hx)}x_t$$

$$\frac{\partial J_t(\theta)}{\partial \theta} = \sum_{k=1}^{t} \frac{\partial J_t(\theta)}{\partial h_t} \frac{\partial h_t}{\partial h_k} \frac{\partial h_t}{\partial \theta}$$

$$\frac{\partial h_t}{\partial h_k} = \prod_{i=k+1}^{t} \frac{\partial h_i}{\partial h_{i-1}} = \prod_{i=k+1}^{t} W^{(hh)^\top} \mathrm{diag}(\sigma'(h_{i-1}))$$

$$||\frac{\partial h_i}{\partial h_{i-1}}|| \leq ||W^{(hh)^\top}|| \cdot ||\mathrm{diag}(\sigma'(h_{i-1}))|| \leq \eta$$

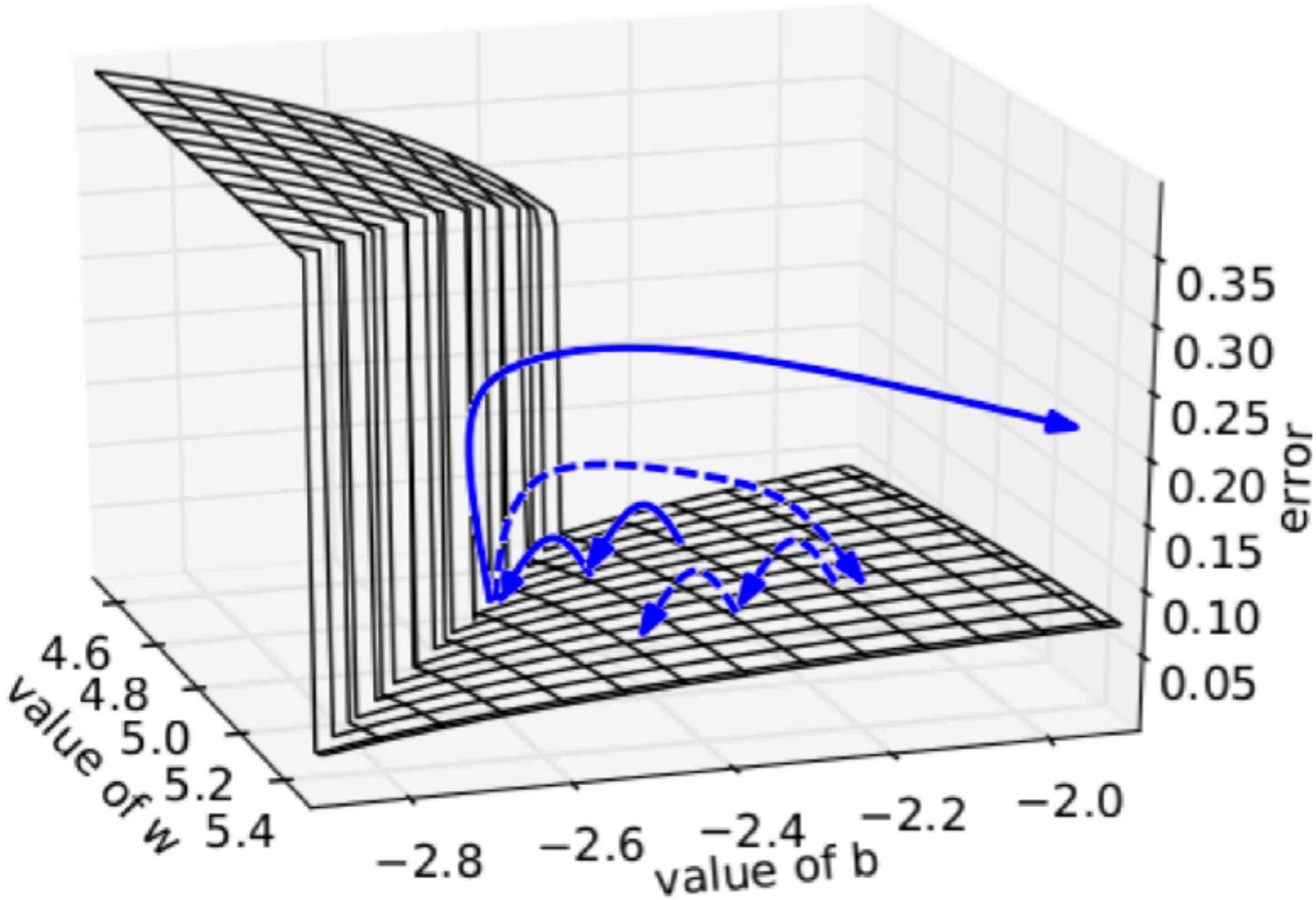
The contribution of step k to the total gradient at step t is small

Solutions

- Exploding gradient: gradient clipping
  - Re-normalize gradient to be less than C (this is no longer the true gradient)
    - Exploding gradients are easy to detect
- The problem is with the model!
  - Change it (LSTMs, GRUs)
  - Maybe later in this class

Pascanu et al, 2013

Illustration



Jozefowicz et al, 2016

Results

Model	Test Perplexity	Number of Params [billions]
Sigmoid-RNN-2048 (Ji et al., 2015a)	68.3	4.1
Interpolated KN 5-gram, 1.1B n-grams (Chelba et al., 2013)	67.6	1.76
Sparse Non-Negative Matrix LM (Shazeer et al., 2015)	52.9	33
RNN-1024 + MaxEnt 9-gram features (Chelba et al., 2013)	51.3	20
LSTM-512-512	54.1	0.82
LSTM-1024-512	48.2	0.82
LSTM-2048-512	43.7	0.83
LSTM-8192-2048 (No Dropout)	37.9	3.3
LSTM-8192-2048 (50% Dropout)	32.2	3.3
2-Layer LSTM-8192-1024 (BIG LSTM)	30.6	1.8
BIG LSTM+CNN Inputs	**30.0**	**1.04**
BIG LSTM+CNN Inputs + CNN Softmax	39.8	**0.29**
BIG LSTM+CNN Inputs + CNN Softmax + 128-dim correction	35.8	**0.39**
BIG LSTM+CNN Inputs + Char LSTM predictions	47.9	**0.23**

Summary

- Language modeling is a fundamental NLP task used in machine translation, spelling correction, speech recognition, etc.
- Traditional models use n-gram counts and smoothing
- Feed-forward take into account word similarity to generalize better
- Recurrent models can potentially learn to exploit long-range interactions
- Neural models dramatically reduced perplexity
- Recurrent networks are now used in many other NLP tasks (bidirectional RNNs, deep RNNs)