Algorithms for NLP

Language Modeling III

Taylor Berg-Kirkpatrick – CMU

Slides: Dan Klein – UC Berkeley

Efficient Hashing

- Closed address hashing
  - Resolve collisions with chains
  - Easier to understand but bigger
- Open address hashing
  - Resolve collisions with probe sequences
  - Smaller but easy to mess up
- Direct-address hashing
  - No collision resolution
  - Just eject previous entries
  - Not suitable for core LM storage

Integer Encodings

word ids

7 1 15

the cat laughed → 233

n-gram count

Bit Packing

Got 3 numbers under 2^{20} to store?

7	1	15
0…0011 1	0...00001	0...01111
20 bits	20 bits	20 bits

Fits in a primitive 64-bit long

Integer Encodings

n-gram encoding

15176595 = | 20 bits | 20 bits | 20 bits |

~~the cat laughed~~ → 233

n-gram count

Rank Values

$c(\text{the}) = 23135851162 < 2^{35}$

35 bits to represent integers between 0 and 2^{35}

$$\overbrace{15176595}^{\text{60 bits}} \longrightarrow \overbrace{233}^{\text{35 bits}}$$

n-gram encoding　　　　count

Rank Values

$\#$ unique counts = 770000 < 2^{20}

20 bits to represent ranks of all counts

$$\overbrace{15176595}^{\text{60 bits}} \longrightarrow \overbrace{3}^{\text{20 bits}}$$

n-gram encoding → rank

rank	freq
0	1
1	2
2	51
3	233

So Far

Word indexer

word	id
cat	0
the	1
was	2
ran	3

N-gram encoding scheme

unigram: $f(id) = id$

bigram: $f(id_1, id_2) = ?$

trigram: $f(id_1, id_2, id_3) = ?$

Rank lookup

rank	freq
0	1
1	2
2	51
3	233

Count DB

unigram

16078820	0381
15176595	0051
15176583	0076
—	—
16576628	0021
—	—
15176600	0018
16089320	0171
15176583	0039
14980420	0030
—	—
15020330	0482

bigram

16078820	0381
15176595	0051
15176583	0076
—	—
16576628	0021
—	—
15176600	0018
16089320	0171
15176583	0039
14980420	0030
—	—
15020330	0482

trigram

16078820	0381
15176595	0051
15176583	0076
—	—
16576628	0021
—	—
15176600	0018
16089320	0171
15176583	0039
14980420	0030
—	—
15020330	0482

Hashing vs Sorting

query: 15176595

Sorting

c	val
15176583	0076
15176595	0051
15176600	0018
16078820	0381
16089320	0171
16576628	0021
16980420	0030
17020330	0482
17176583	0039

Hashing

c	val
16078820	0381
15176595	0051
15176583	0076
—	—
16576628	0021
—	—
15176600	0018
16089320	0171
15176583	0039
14980420	0030
—	—
15020330	0482

Maximum Entropy Models

Improving on N-Grams?

- N-grams don't combine multiple sources of evidence well

P(construction | After the demolition was completed, the)

- Here:
  - "the" gives syntactic constraint
  - "demolition" gives semantic constraint
  - Unlikely the interaction between these two has been densely observed in this specific n-gram

- We'd like a model that can be more statistically efficient

Some Definitions

INPUTS	$\mathbf{x}_i$	*close the ____*
CANDIDATE SET	$\mathcal{Y}(\mathbf{x})$	*{door, table, …}*
CANDIDATES	$\mathbf{y}$	*table*
TRUE OUTPUTS	$\mathbf{y}_i^*$	*door*
FEATURE VECTORS	$\mathrm{f}(\mathrm{x}, \mathbf{y})$	[0 0 1 0 0 0 1 0 0 0 0 0]

x_{-1}="the" ∧ y="door"

x_{-1}="the" ∧ y="table"

"close" in x ∧ y="door"

y occurs in x

More Features, Less Interaction

x = closing the ____, y = doors

- N-Grams x_{-1}="*the*" $\wedge$ y="doors"
- Skips x_{-2}="*closing*" $\wedge$ y="doors"
- Lemmas x_{-2}="*close*" $\wedge$ y="door"
- Caching *y occurs in x*

Data: Feature Impact

Features	Train Perplexity	Test Perplexity
3 gram indicators	241	350
1-3 grams	126	172
1-3 grams + skips	101	164

Exponential Form

- Weights $\mathbf{w}$ Features $\mathbf{f}(\mathbf{x}, \mathbf{y})$
- Linear score $\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y})$
- Unnormalized probability

$$\mathsf{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) \propto \mathsf{exp}(\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}))$$

- Probability

$$\mathsf{P}(\mathbf{y}|\mathbf{x}, \mathbf{w}) = \frac{\mathsf{exp}(\mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}))}{\sum_{\mathbf{y}'} \mathsf{exp}(\mathbf{w}^\top \mathbf{f}(\mathbf{x},\mathbf{y}'))}$$

Likelihood Objective

- Model form:

$$P(y|x,w) = \frac{\exp(w^\top f(x,y))}{\sum_{y'} \exp(w^\top f(x,y'))}$$

- Log-likelihood of training data

$$L(w) = \log \prod_i P(y_i^*|x_i, w) = \sum_i \log \left(\frac{\exp(w^\top f(x_i, y_i^*))}{\sum_{y'} \exp(w^\top f(x_i, y'))} \right)$$

$$= \sum_i \left(w^\top f(x_i, y_i^*) - \log \sum_{y'} \exp(w^\top f(x_i, y')) \right)$$

Training

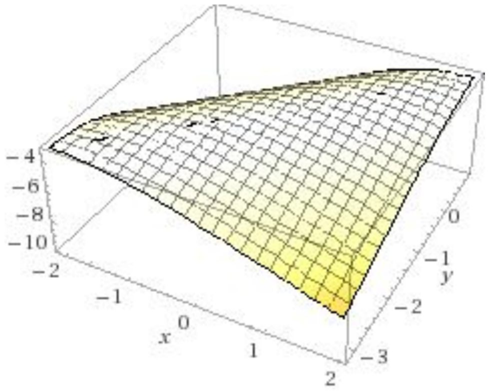
History of Training

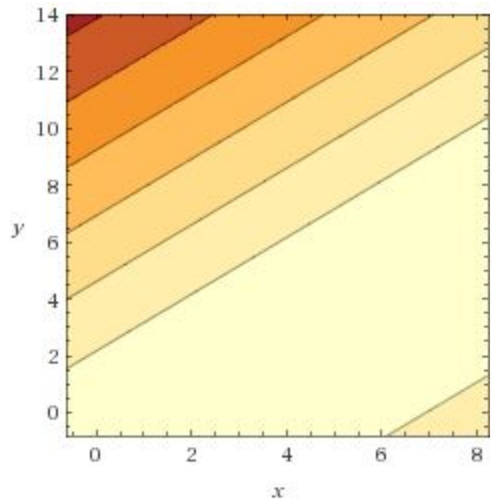
- 1990’s: Specialized methods (e.g. iterative scaling)
- 2000’s: General-purpose methods (e.g. conjugate gradient)
- 2010’s: Online methods (e.g. stochastic gradient)

What Does LL Look Like?

- Example
  - Data: xxxy
  - Two outcomes, x and y
  - One indicator for each
  - Likelihood

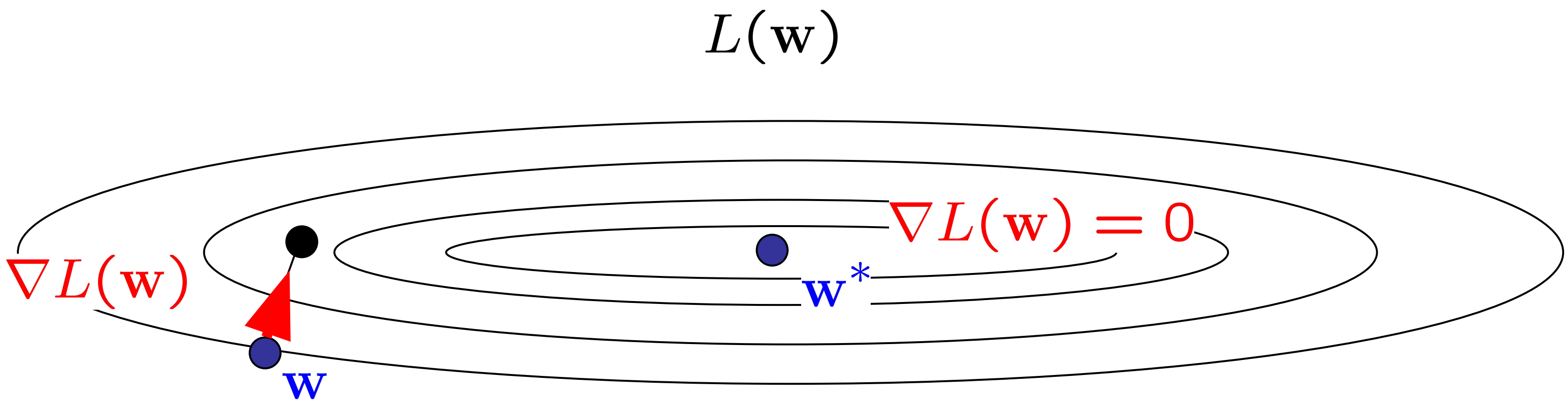
$$\log\left(\left(\frac{e^x}{e^x+e^y}\right)^3 \times \frac{e^y}{e^x+e^y}\right)$$





Convex Optimization

- The maxent objective is an unconstrained convex problem



- One optimal value*, gradients point the way

Gradients

$$L(\mathbf{w}) = \sum_i \left(\mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y})) \right)$$

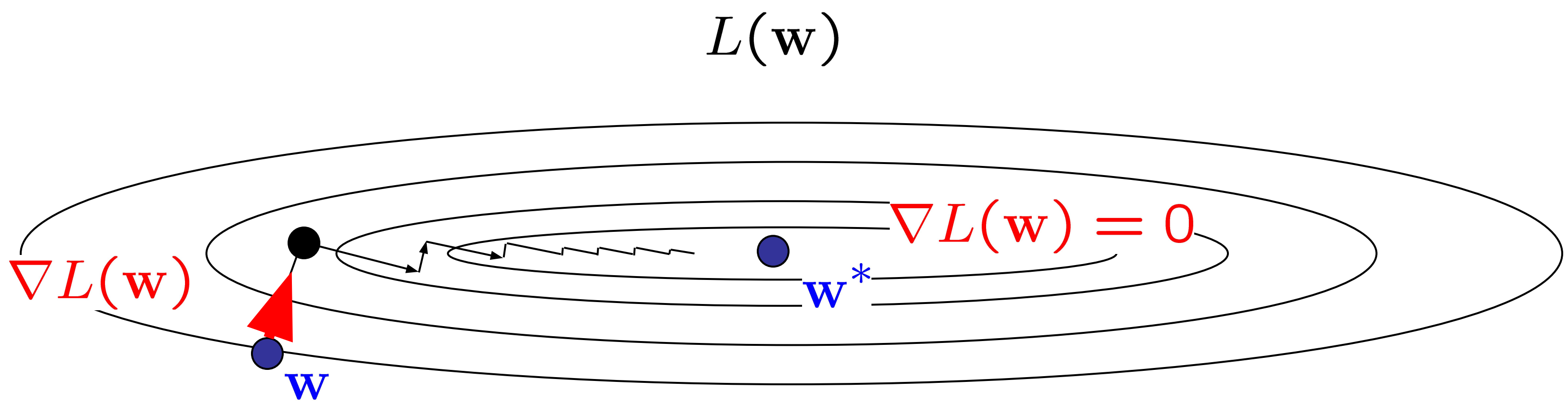
$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \left(\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i) \mathbf{f}(\mathbf{x}_i, \mathbf{y}) \right)$$

Count of features under target labels

Expected count of features under model predicted label distribution

Gradient Ascent

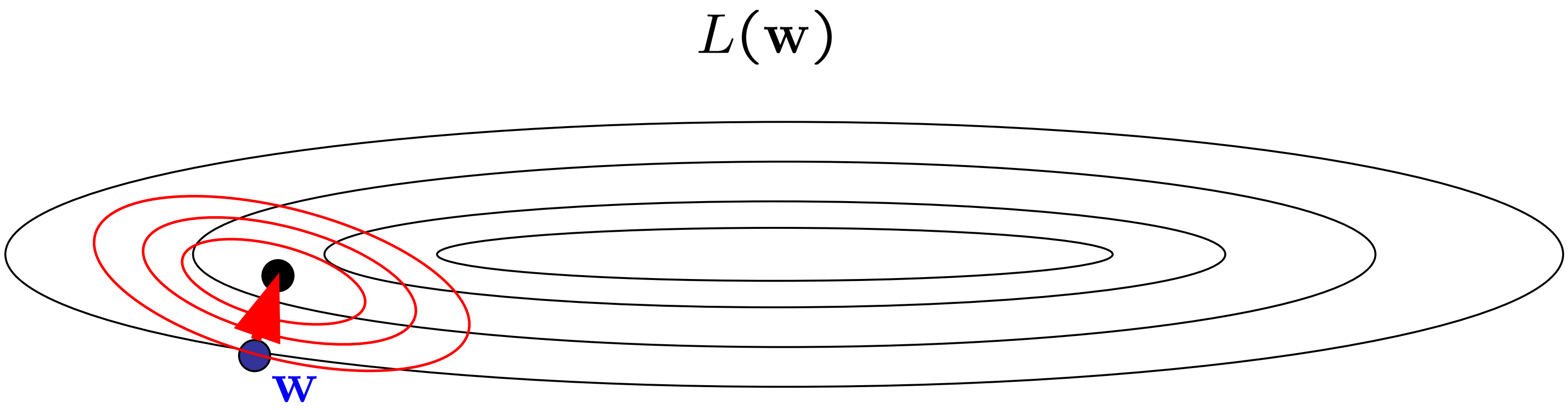
- The maxent objective is an unconstrained optimization problem



- Gradient Ascent
  - Basic idea: move uphill from current guess
  - Gradient ascent / descent follows the gradient incrementally
  - At local optimum, derivative vector is zero
  - Will converge if step sizes are small enough, but not efficient
  - All we need is to be able to evaluate the function and its derivative

(Quasi)-Newton Methods

- 2^{nd}-Order methods: repeatedly create a quadratic approximation and solve it



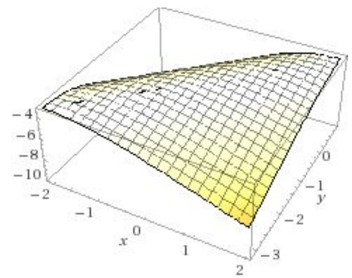
$$L(\mathbf{w}_0) + \nabla L(\mathbf{w})^\top (\mathbf{w} - \mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)^\top \nabla^2 L(\mathbf{w}) (\mathbf{w} - \mathbf{w}_0)$$

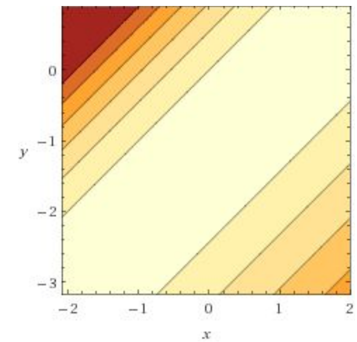
- E.g. LBFGS, which tracks derivative to approximate (inverse) Hessian

Regularization

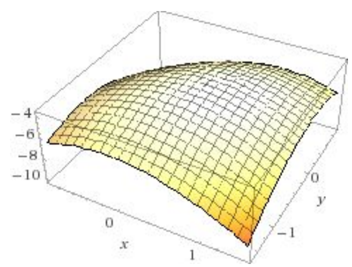
Regularization Methods

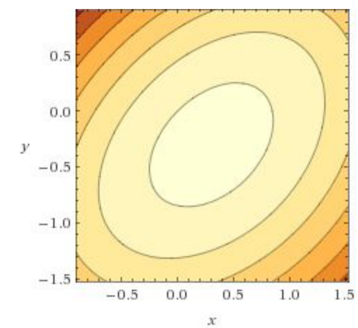
- Early stopping



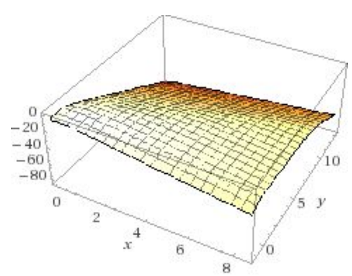


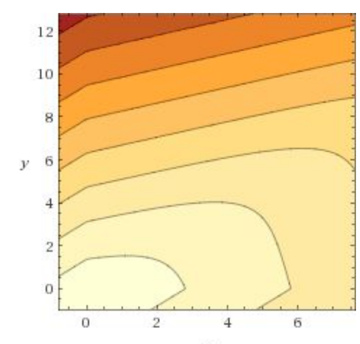
- L2: $L(w)-|w|_2^2$





- L1: $L(w)-|w|$





Regularization Effects

- Early stopping: don’t do this
- L2: weights stay small but non-zero
- L1: many weights driven to zero
  - Good for sparsity
  - Usually bad for accuracy for NLP

Scaling

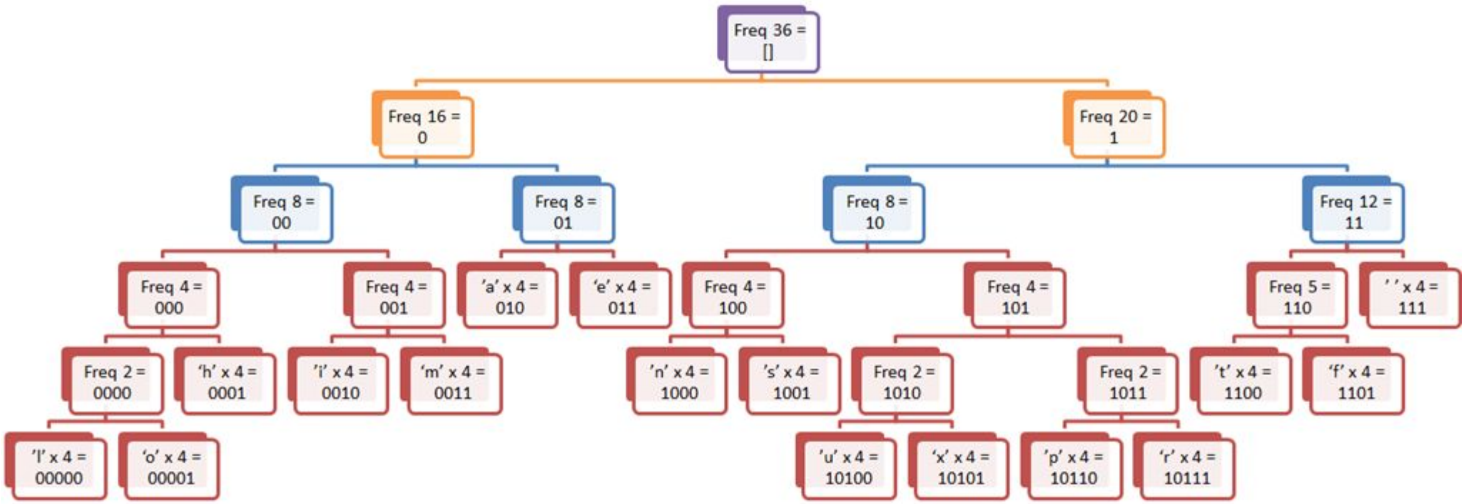
Why is Scaling Hard?

$$L(\mathbf{w}) = \sum_i \left(\mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \log \sum_{\mathbf{y}} \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{x}_i, \mathbf{y})) \right)$$

- Big normalization terms
- Lots of data points

Hierarchical Prediction

- Hierarchical prediction / softmax [Mikolov et al 2013]



- Noise-Contrastive Estimation [Mnih, 2013]

- Self-Normalization [Devlin, 2014]

Image: ayende.com

Stochastic Gradient

- View the gradient as an average over data points

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \frac{1}{N} \sum_i \left(\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i) \mathbf{f}(\mathbf{x}_i, \mathbf{y}) \right)$$

- Stochastic gradient: take a step each example (or mini-batch)

$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} \approx \frac{1}{1} \left(\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i) \mathbf{f}(\mathbf{x}_i, \mathbf{y}) \right)$$

- Substantial improvements exist, e.g. AdaGrad (Duchi, 11)

Other Methods

Neural Net LMs

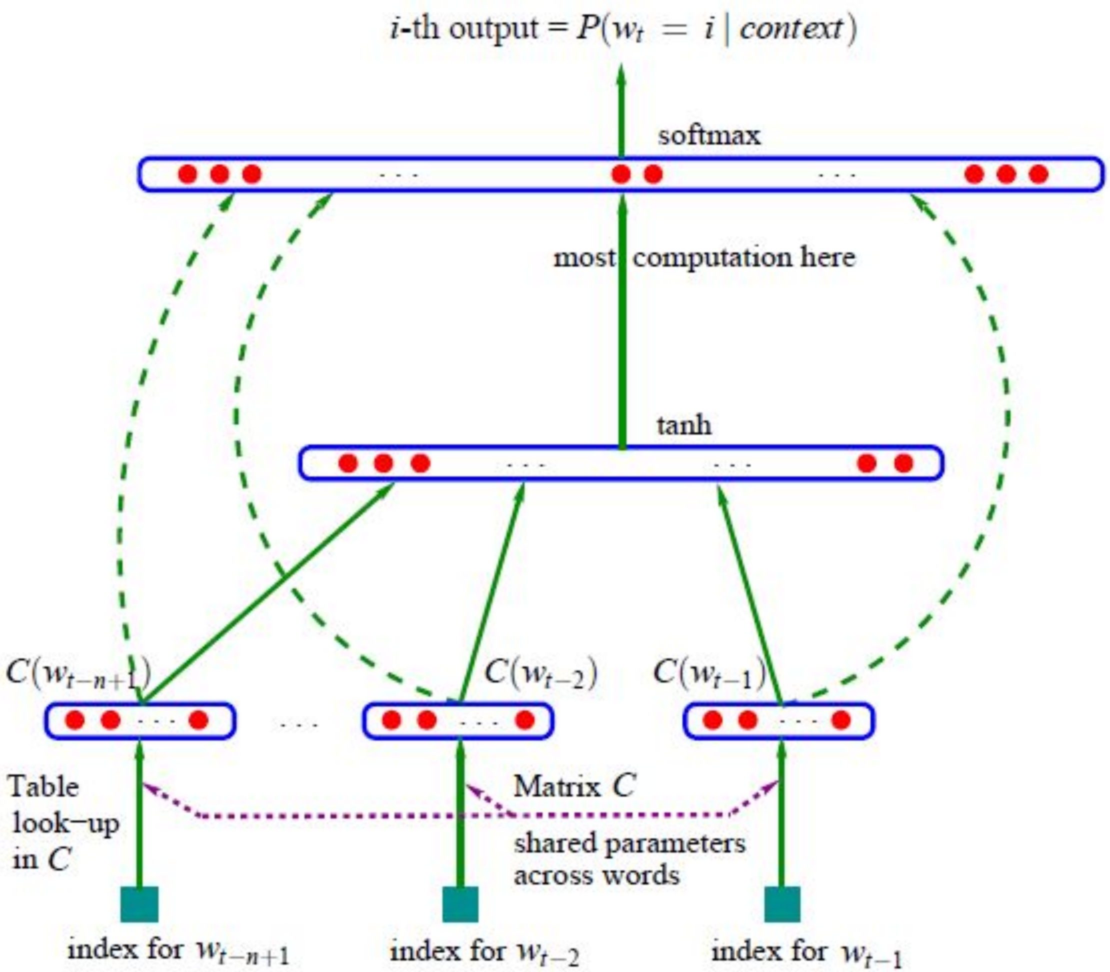


Image: (Bengio et al, 03)

Neural vs Maxent

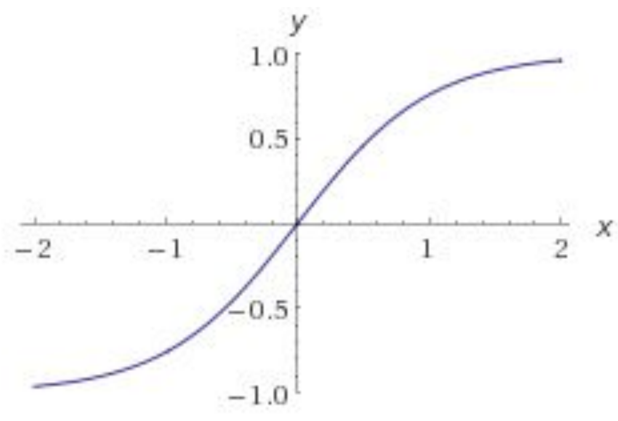
- Maxent LM

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \propto \exp(\mathbf{w}^\top \mathbf{f}(\mathbf{x}, \mathbf{y}))$$

- Neural Net LM

$$P(\mathbf{y}|\mathbf{x}, \mathbf{w}) \propto \exp\left(B\sigma\left(Af(x)\right)\right)$$

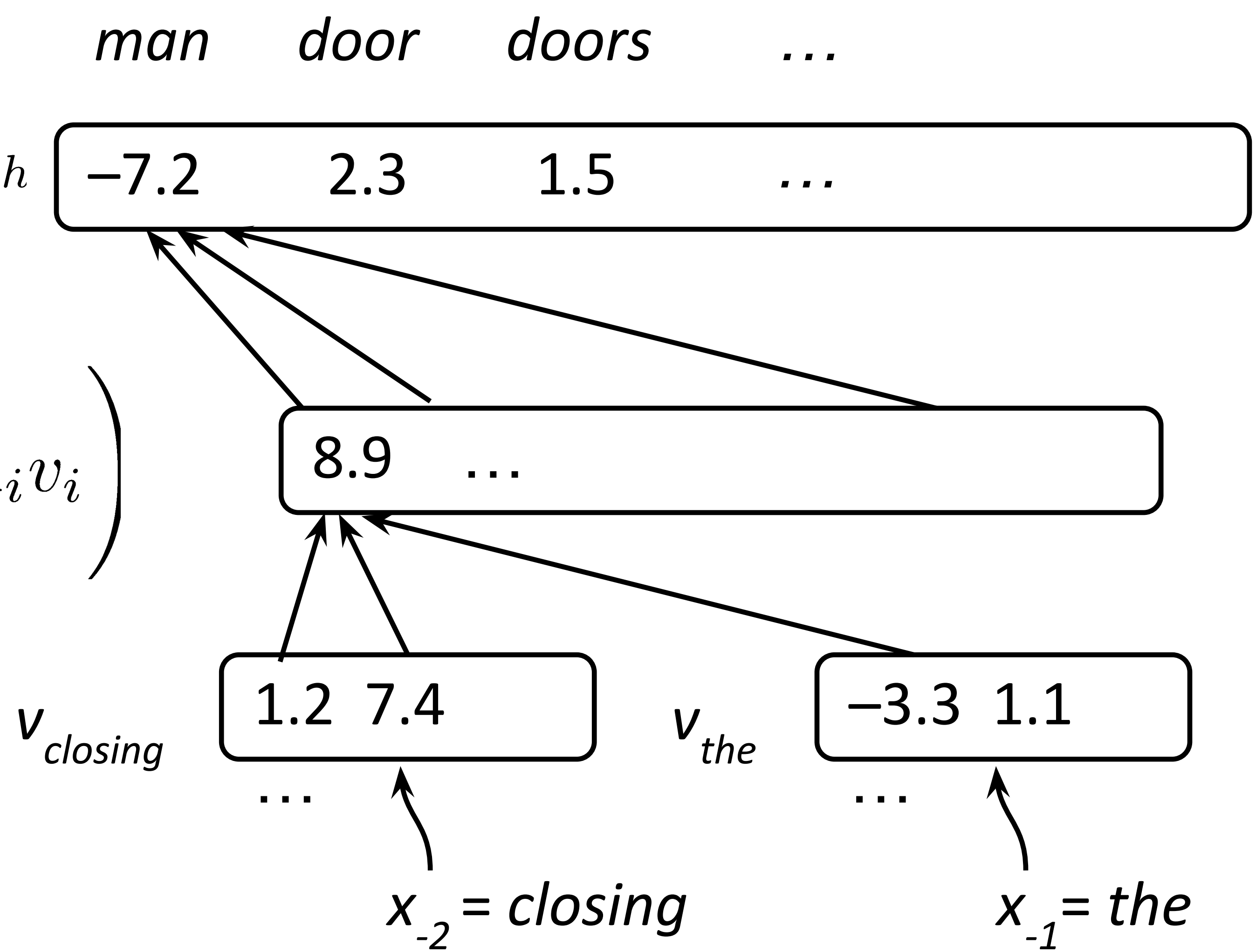
σ nonlinear, e.g. tanh



Neural Net LMs

$$P(y|w,x) \propto e^{Bh}$$

$$h = \sigma\left(\sum_i A_i v_i\right)$$



Maximum Entropy LMs

- Want a model over completions y given a context x:

$$P(y|x) = P(\text{ close the door | close the })$$

- Want to characterize the important aspects of y = (v,x) using a feature function f

- F might include
  - Indicator of v (unigram)
  - Indicator of v, previous word (bigram)
  - Indicator whether v occurs in x (cache)
  - Indicator of v and each non-adjacent previous word
  - …