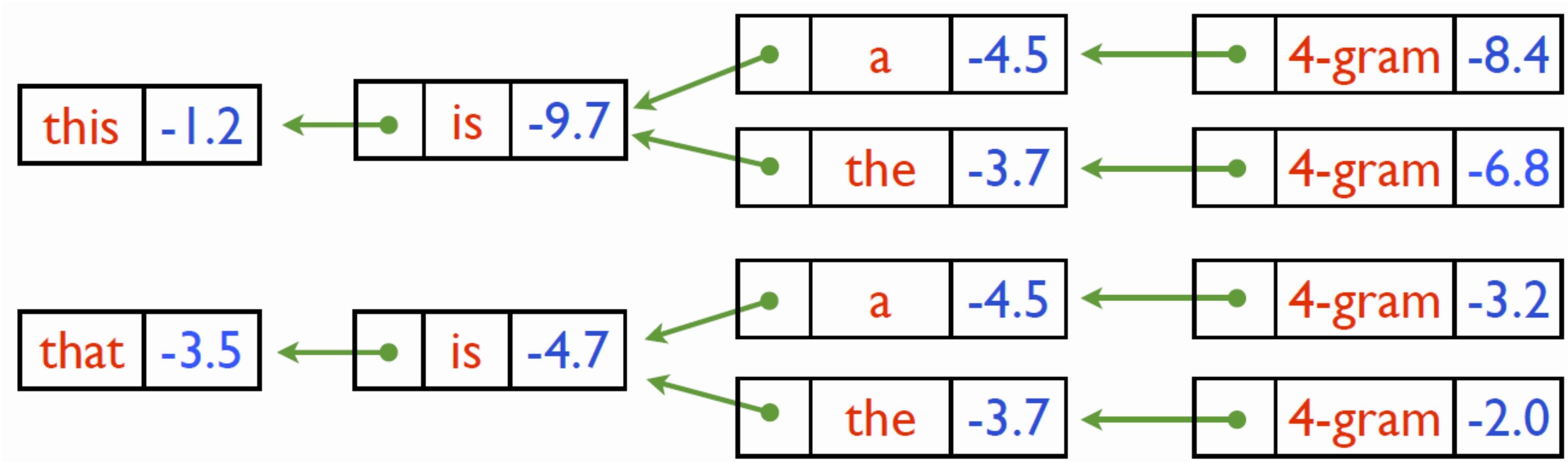
Algorithms for NLP

Language Modeling III

Taylor Berg-Kirkpatrick – CMU

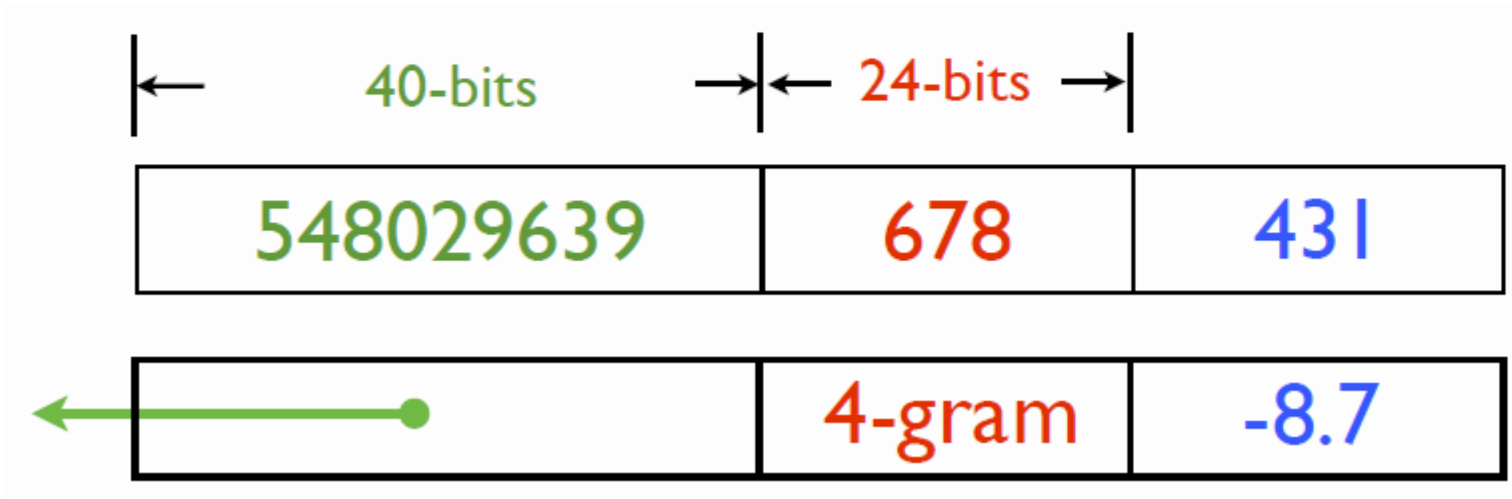
Slides: Dan Klein – UC Berkeley

Tries



[Hsu and Glass 2008]

Context Encodings



Google N-grams

- 10.5 bytes/n-gram
- 37 GB total

[Many details from Pauls and Klein, 2011]

Context Encodings

1-grams

	w	val	
675		0127	"this"
676		9008	
677		0137	
678		0090	"a"
679		1192	
680		0050	"the"
681		0040	
682		0201	"is"
683		3010	"was"

val: 20 bits

2-grams

	c	w	val	
15176582	00000480	682	0065	"is"
15176583	00000675	682	0808	"is"
15176584	00000802	682	0012	"is"
15176585	00001321	682	0400	"is"
15176586	00002482	682	0030	"is"
15176587	00002588	682	0260	"is"
15176588	00000390	683	0013	"was"
15176589	00000676	683	0025	"was"
15176590	00000984	683	0086	"was"

c + w: 64 bits; val: 20 bits

3-grams

	c	w	val	
42276773	15176583	678	0076	"a"
42276774	15176595	678	0051	"a"
42276775	15176600	678	0018	"a"
42276776	16078820	678	0381	"a"
42276777	16089320	678	0171	"a"
42276778	16576628	678	0021	"a"
42276779	14980420	680	0030	"the"
42276780	15020330	680	0482	"the"
42276781	15176583	680	0039	"the"

c + w: 64 bits; val: 20 bits

Compression

Idea: Differential Compression

c	*w*	*val*
15176585	678	3
15176587	678	2
15176593	678	1
15176613	678	8
15179801	678	1
15176585	680	298
15176589	680	1

Δc	Δw	*val*
15176583	678	3
+2	+0	2
+6	+0	1
+40	+0	8
+188	+0	1
15176585	+2	298
+4	+0	1

$\|\Delta w\|$	$\|\Delta c\|$	$\|val\|$
40	24	3
3	2	3
3	2	3
9	2	6
12	2	3
36	4	15
6	2	3

15176585	678	563097887	956	3	0	+2	+0	2	+6	+0	1	+40	+2	8	. . .

Variable Length Encodings

Encoding "9"

000 1001

Length in Unary | Number in Binary

Google N-grams

- 2.9 bytes/n-gram
- 10 GB total

[Elias, 75]

Speed-Ups

Context Encodings

1-grams

w	val	
675	0127	"this"
676	9008	
677	0137	
678	0090	"a"
679	1192	
680	0050	"the"
681	0040	
682	0201	"is"
683	3010	"was"

val: 20 bits

2-grams

	c	w	val	
15176582	00000480	682	0065	"is"
15176583	00000675	682	0808	"is"
15176584	00000802	682	0012	"is"
15176585	00001321	682	0400	"is"
15176586	00002482	682	0030	"is"
15176587	00002588	682	0260	"is"
15176588	00000390	683	0013	"was"
15176589	00000676	683	0025	"was"
15176590	00000984	683	0086	"was"

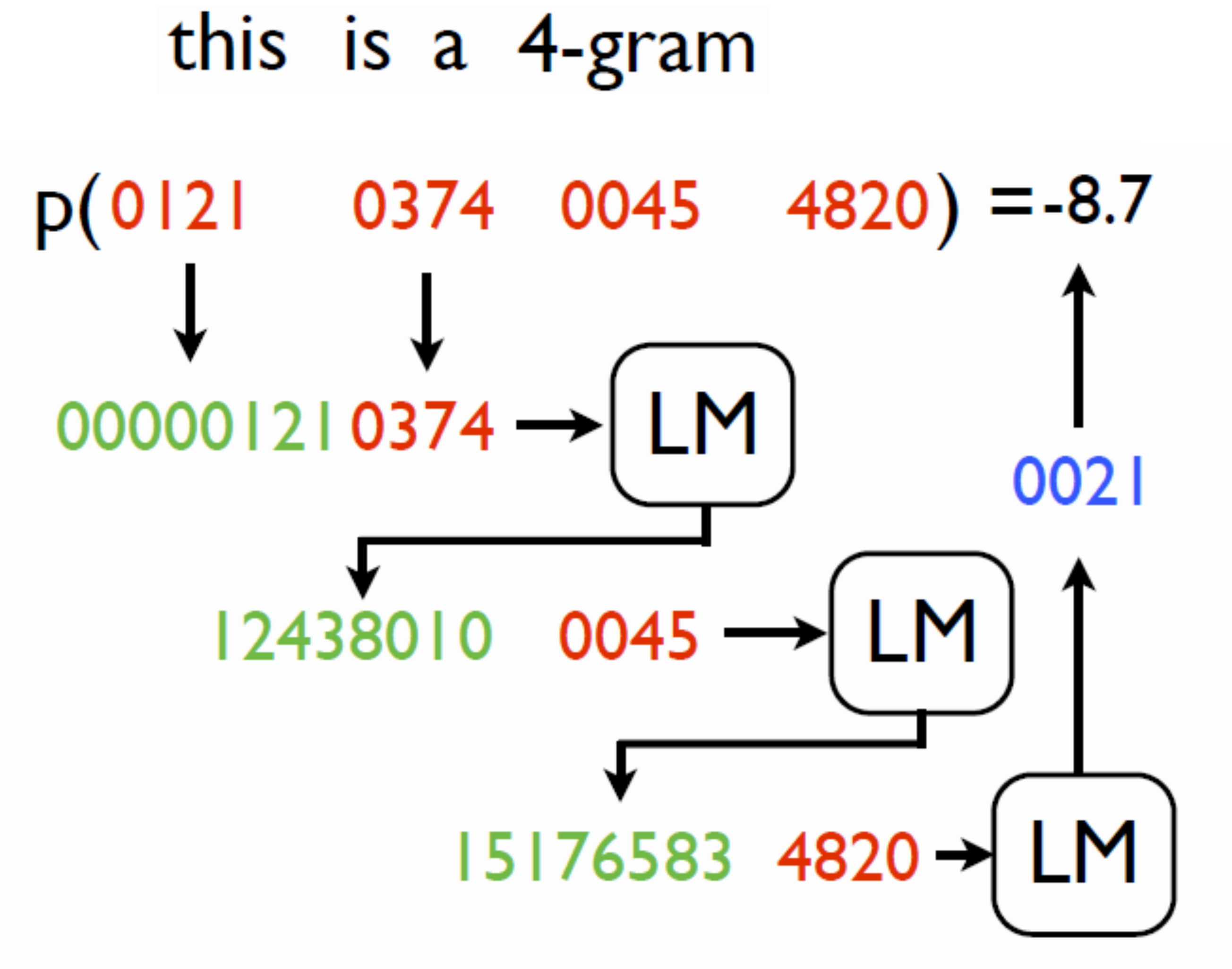
c, w: 64 bits; val: 20 bits

3-grams

	c	w	val	
42276773	15176583	678	0076	"a"
42276774	15176595	678	0051	"a"
42276775	15176600	678	0018	"a"
42276776	16078820	678	0381	"a"
42276777	16089320	678	0171	"a"
42276778	16576628	678	0021	"a"
42276779	14980420	680	0030	"the"
42276780	15020330	680	0482	"the"
42276781	15176583	680	0039	"the"

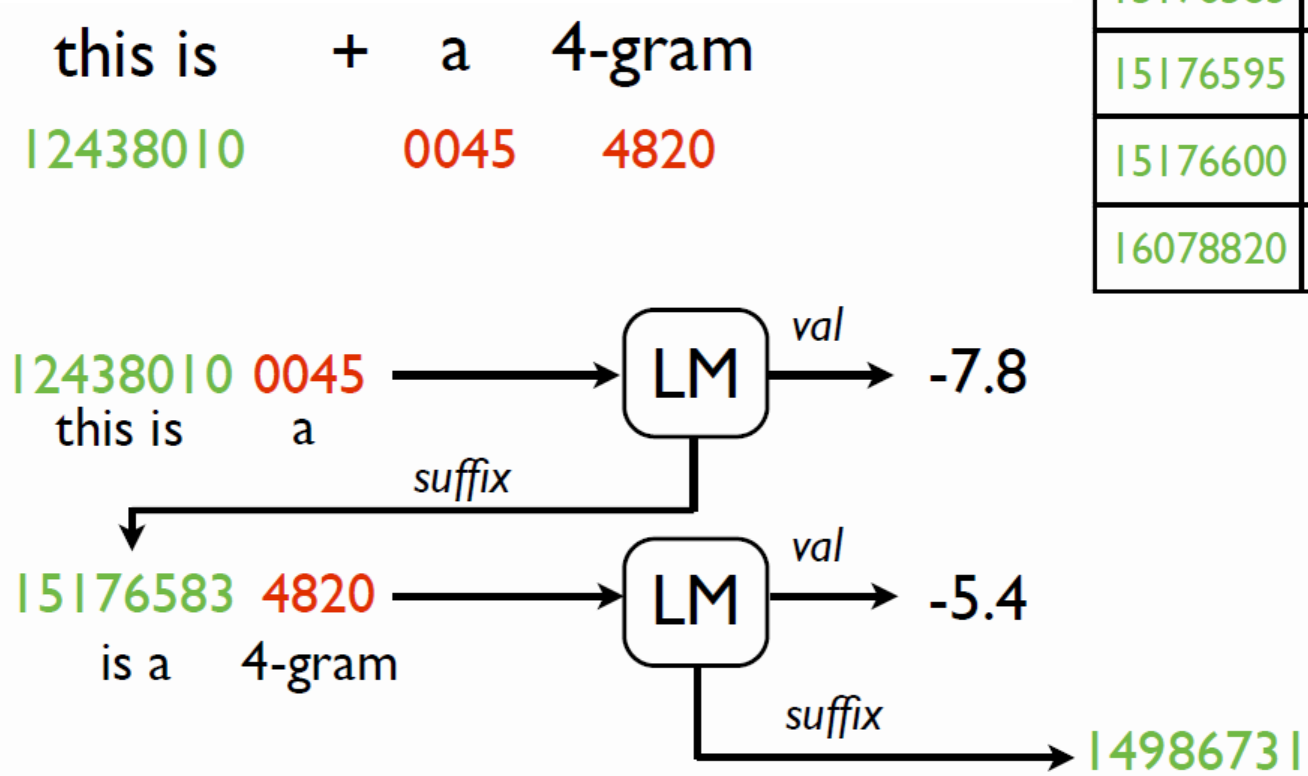
c, w: 64 bits; val: 20 bits

Naïve N-Gram Lookup



Rolling Queries

c	w	val	suffix
15176583	682	0065	00000480
15176595	682	0808	00000675
15176600	682	0012	00000802
16078820	682	0400	00001321



Idea: Fast Caching

	n-gram	probability
0	124 80 42 1243	-7.034
1	37 2435 243 21	-2.394
2	804 42 4298 43	-8.008

hash(124 80 42 1243) =0

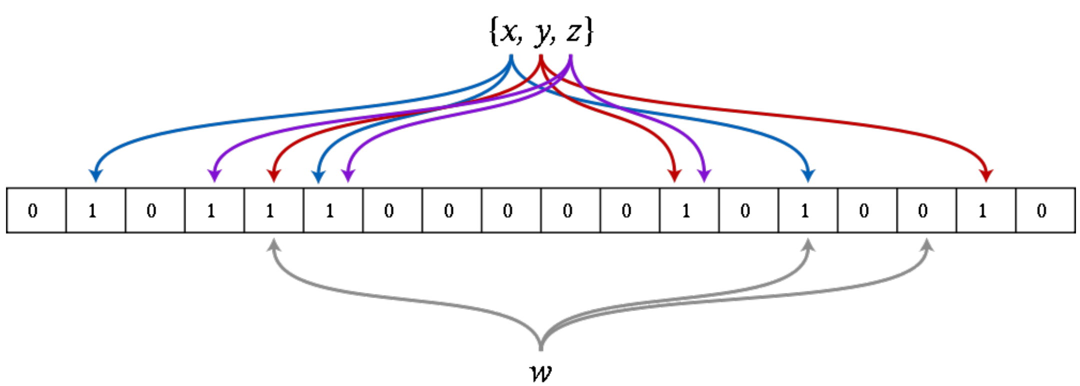
hash(1423 43 42 400) =1

LM can be more than 10x faster w/ direct-address caching

Approximate LMs

- Simplest option: hash-and-hope
  - Array of size K ~ N
  - (optional) store hash of keys
  - Store values in direct-address
  - Collisions: store the max
  - What kind of errors can there be?

- More complex options, like bloom filters (originally for membership, but see Talbot and Osborne 07), perfect hashing, etc



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	$\mathbf{f}(\mathbf{x}, \mathbf{y})$	[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"* ∧ y="doors"
- Skips x_{-2}=*"closing"* ∧ y="doors"
- Lemmas x_{-2}=*"close"* ∧ 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

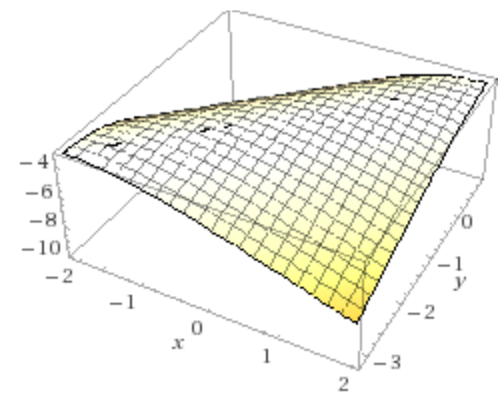
History of Training

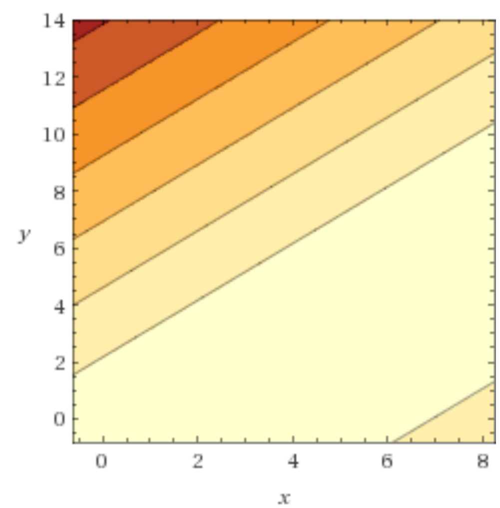
- 1990's: Specialized methods (e.g. iterative scaling)
- 2000's: General-purpose methods (e.g. conjugate gradient)
- 2010's: Online methods (e.g. stochastic gradient)

What Does LL Look Like?

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

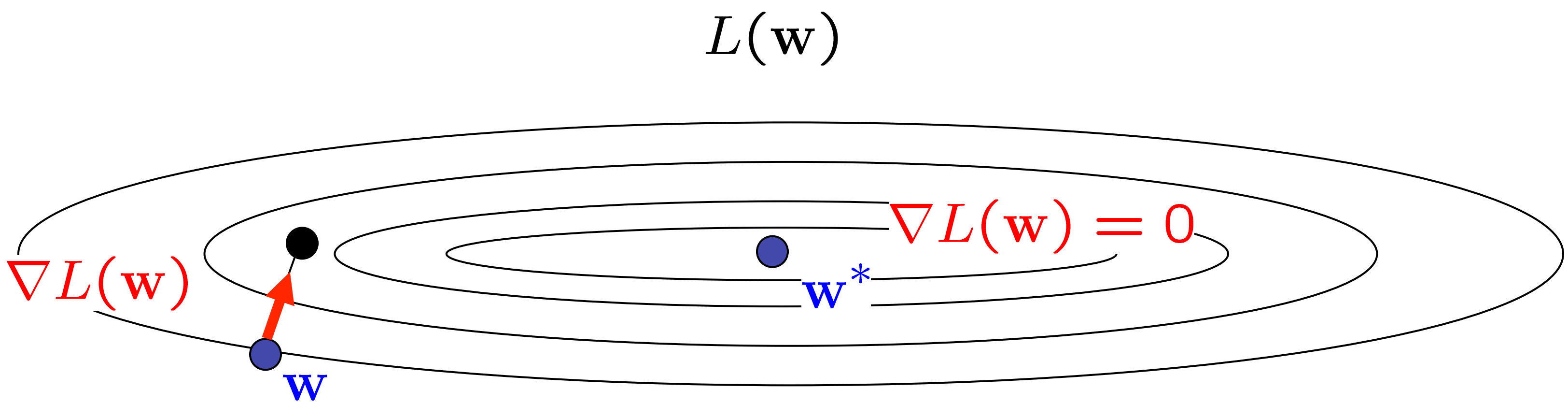
$$\log\left(\left(\frac{e^{x}}{e^{x}+e^{y}}\right)^{3}\times\frac{e^{y}}{e^{x}+e^{y}}\right)$$





Convex Optimization

- The maxent objective is an unconstrained convex problem



- One optimal value*, gradients point the way

Gradients

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

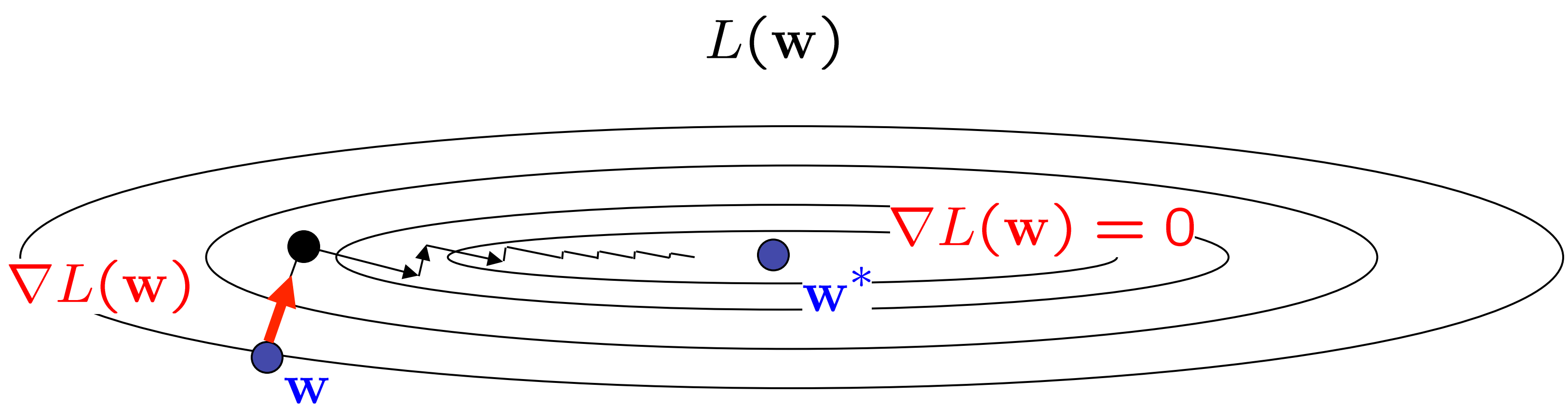
$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \left(\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i) \mathbf{f}(\mathbf{x}_i, \mathbf{y}) \right)$$

Count of features under target labels

Expected count of features under model predicted label distribution

Gradient Ascent

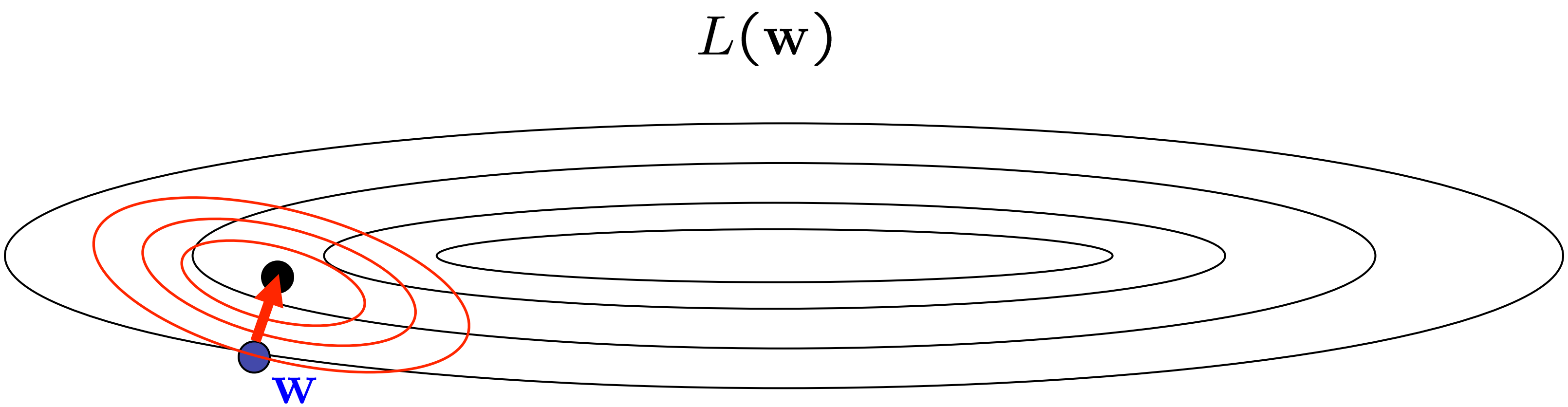
- The maxent objective is an unconstrained optimization problem



- 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



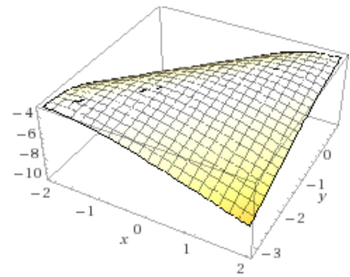
$$L(\mathbf{w}_0) + \nabla L(\mathbf{w})^\top (\mathbf{w} - \mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)^\top \nabla^2 L(\mathbf{w})(\mathbf{w} - \mathbf{w}_0)$$

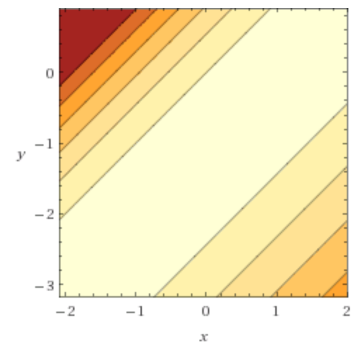
- E.g. LBFGS, which tracks derivative to approximate (inverse) Hessian

Regularization

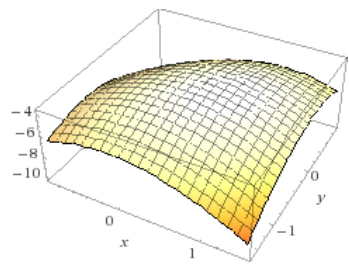
Regularization Methods

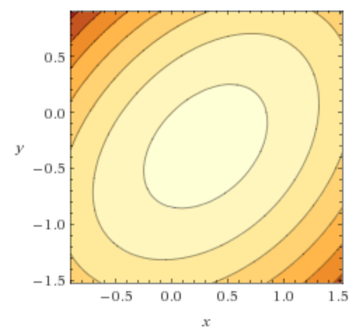
- Early stopping



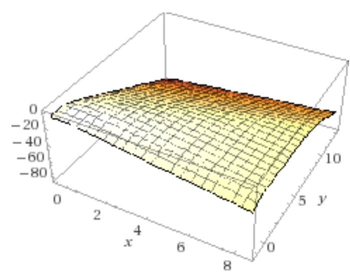


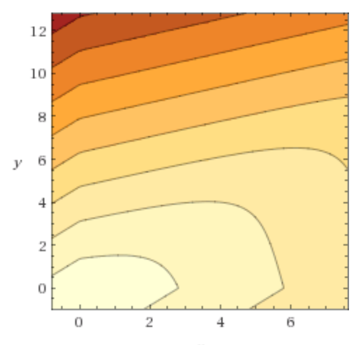
- L2: L(w)-|w|$_2^2$





- 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

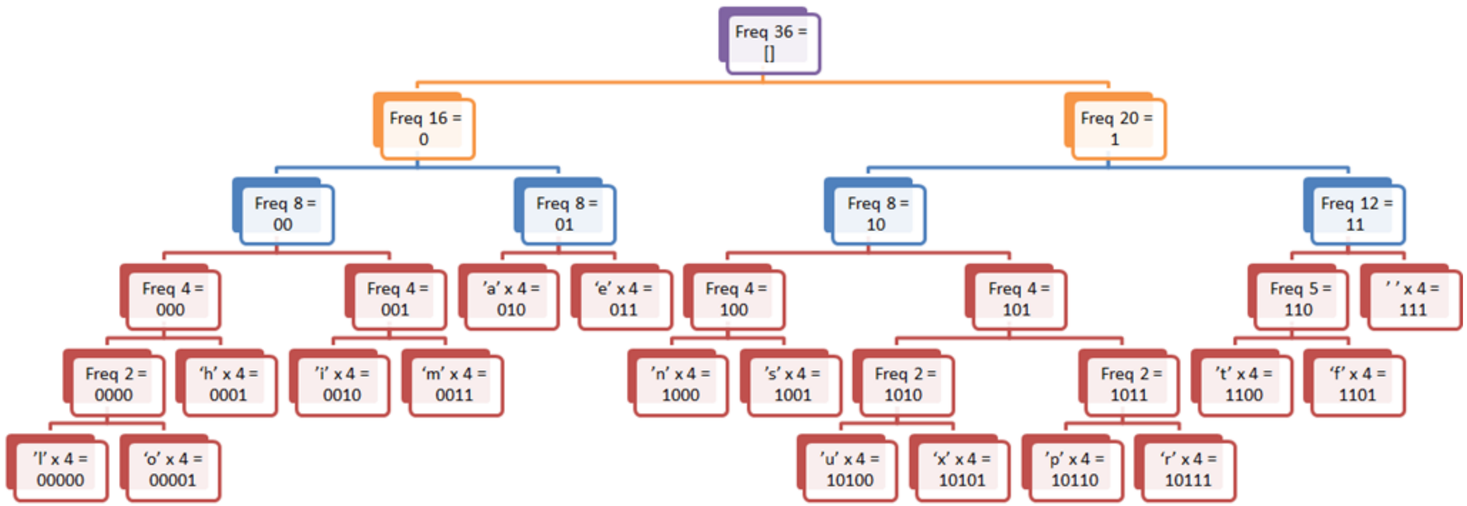
Why is Scaling Hard?

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

- 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

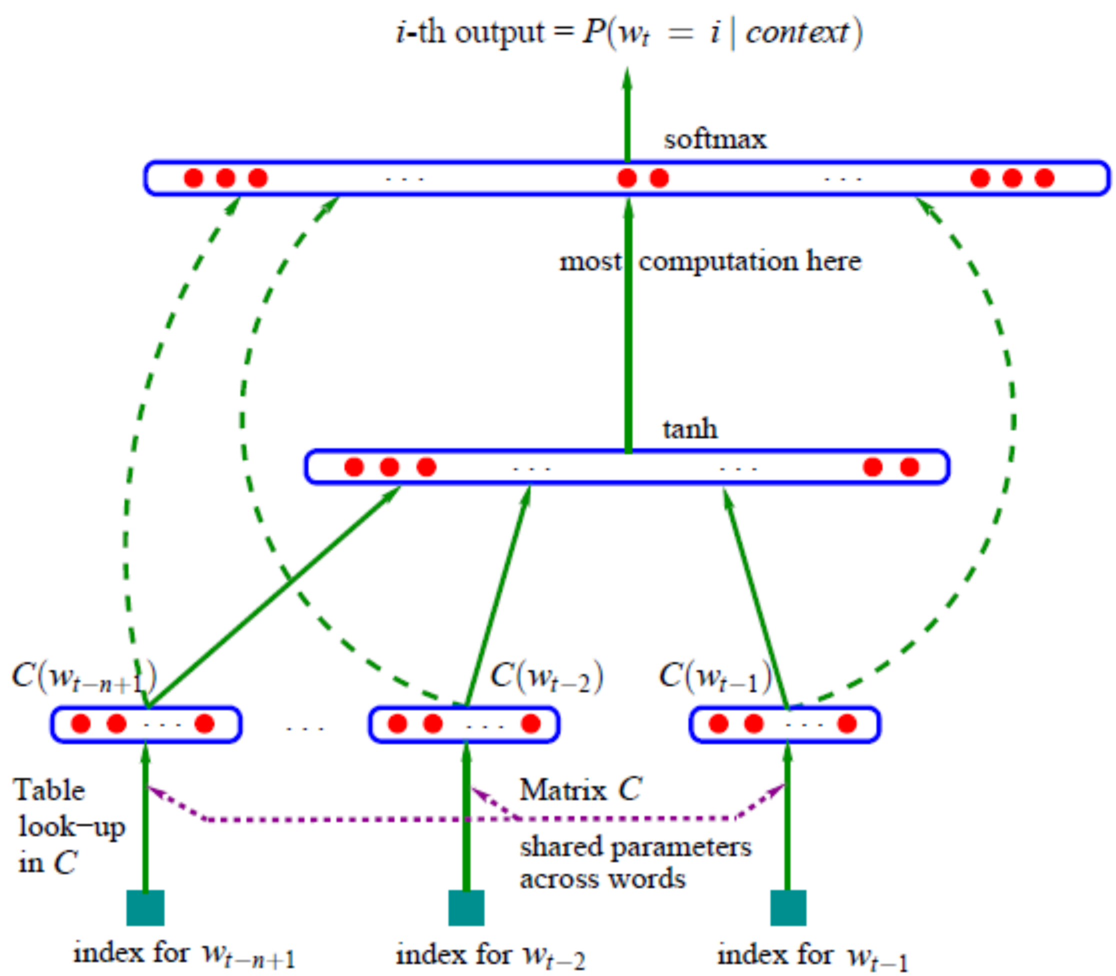


Image: (Bengio et al, 03)

Neural vs Maxent

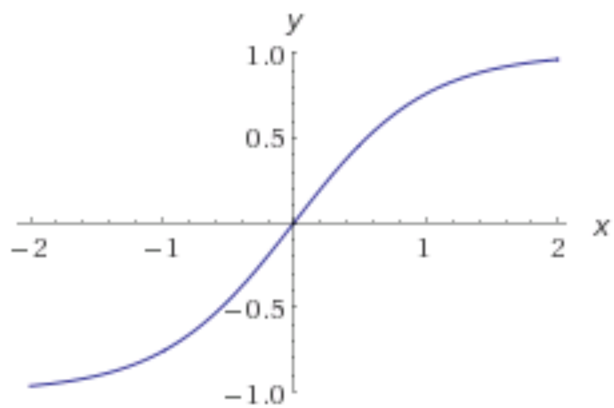
- Maxent LM

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

- 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

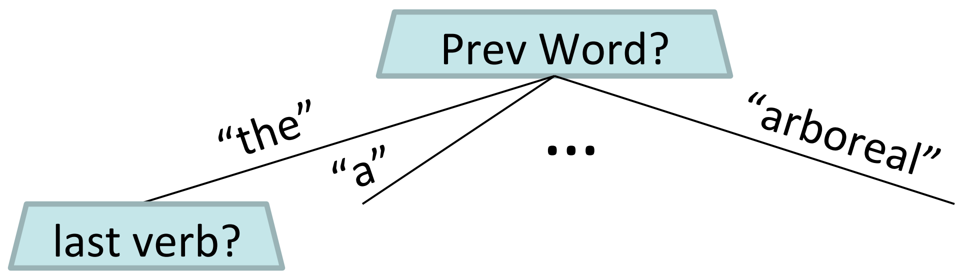


Mixed Interpolation

- But can't we just interpolate:
  - P(w|most recent words)
  - P(w|skip contexts)
  - P(w|caching)
  - …

- Yes, and people do (well, did)
  - But additive combination tends to flatten distributions, not zero out candidates

Decision Trees / Forests



- Decision trees?
  - Good for non-linear decision problems
  - Random forests can improve further [Xu and Jelinek, 2004]
  - Paths to leaves basically learn conjunctions
  - General contrast between DTs and linear models