Natural Language Processing

Language Modeling III

Dan Klein – UC Berkeley

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

Maximum Entropy Models

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

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

- Probability

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

Likelihood Objective

- Model form:

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

- Likelihood of training data

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

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

Training

History of Training

- 1990's: Specialized methods (e.g. iterative scaling)

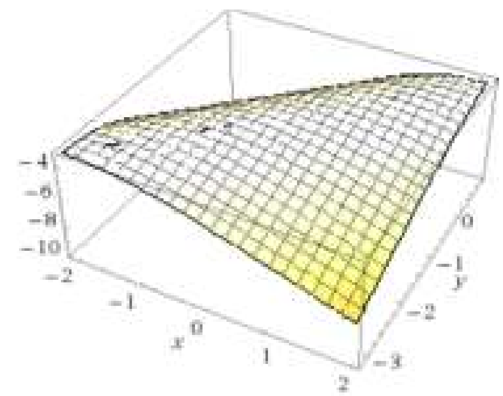
- 2000's: General-purpose methods (e.g. conjugate gradient)

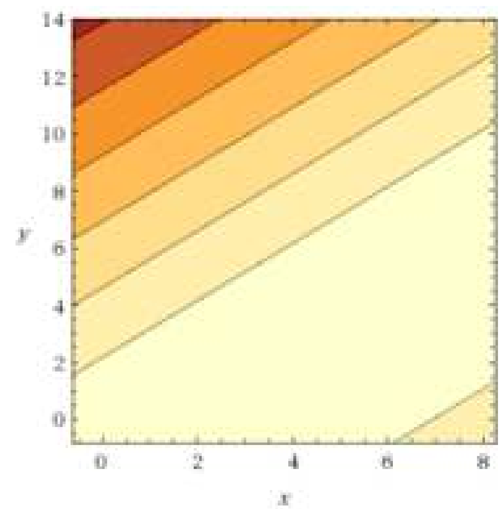
- 2010's: Online methods (e.g. stochastic gradient)

What Does LL Look Like?

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

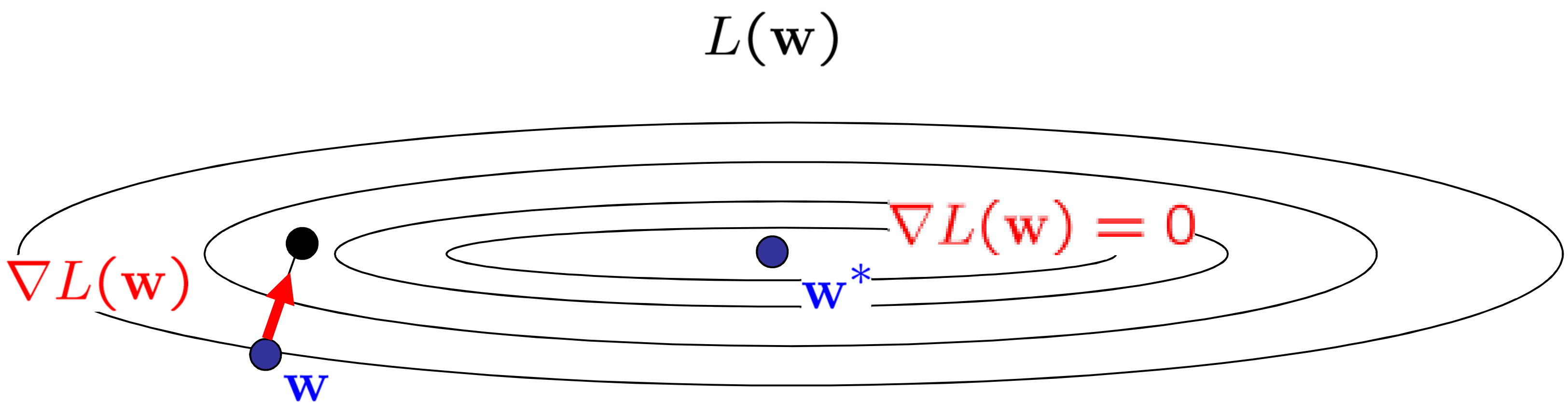
$$\log\left(\left(\frac{e^x}{e^x+e^y}\right)^3 \times \frac{e^y}{e^x+e^y}\right)$$





Convex Optimization

- The maxent objective is an unconstrained convex problem



- One optimal value*, gradients point the way

Gradients

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

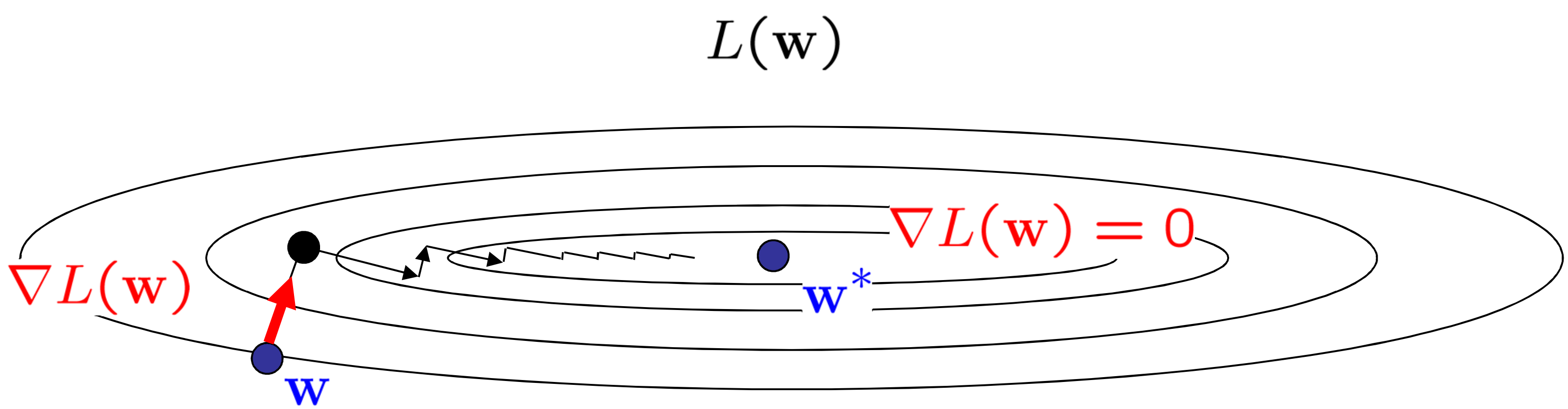
$$\frac{\partial L(\mathbf{w})}{\partial \mathbf{w}} = \sum_i \left(\mathbf{f}(\mathbf{x}_i, \mathbf{y}_i^*) - \sum_{\mathbf{y}} P(\mathbf{y}|\mathbf{x}_i) \mathbf{f}(\mathbf{x}_i, \mathbf{y}) \right)$$

Count of features under target labels

Expected count of features under model predicted label distribution

Gradient Ascent

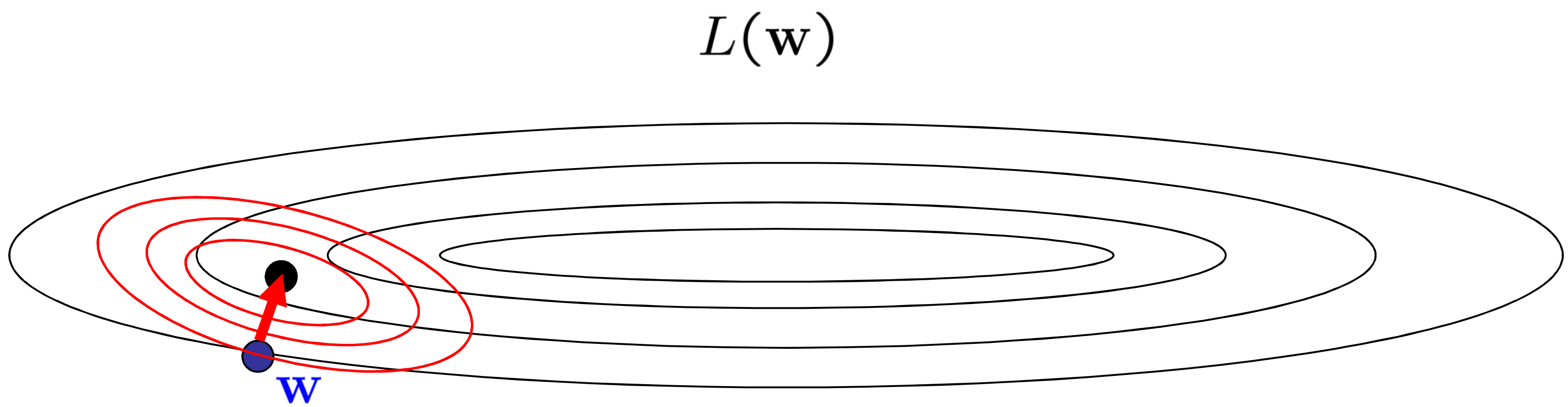
- The maxent objective is an unconstrained optimization problem



- 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

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



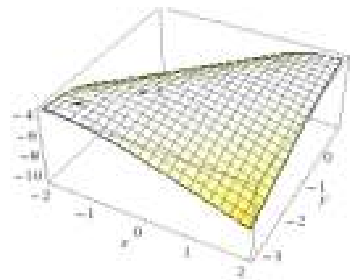
$$L(\mathbf{w}_0) + \nabla L(\mathbf{w})^\top (\mathbf{w} - \mathbf{w}_0) + (\mathbf{w} - \mathbf{w}_0)^\top \nabla^2 L(\mathbf{w}) (\mathbf{w} - \mathbf{w}_0)$$

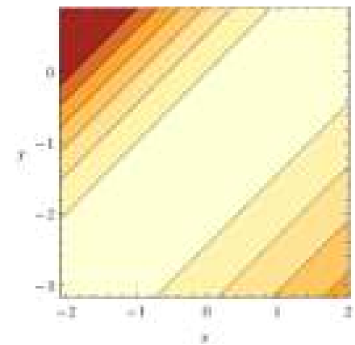
- E.g. LBFGS, which tracks derivative to approximate (inverse) Hessian

Regularization

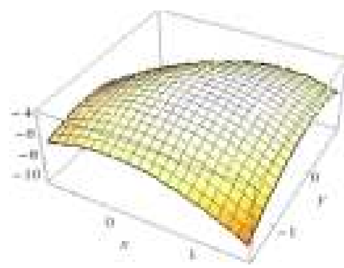
Regularization Methods

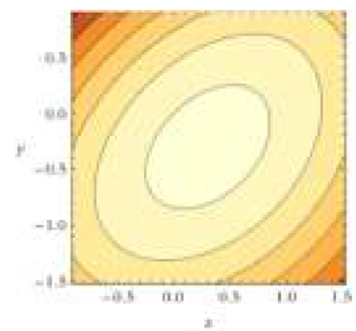
- Early stopping



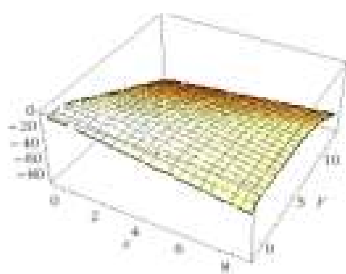


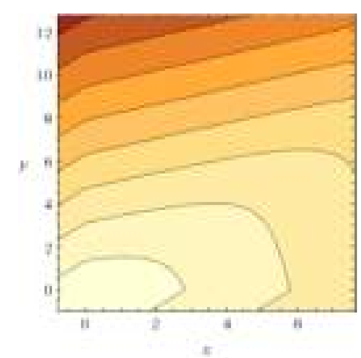
- L2: $LL(w)-|w|_2^2$





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





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

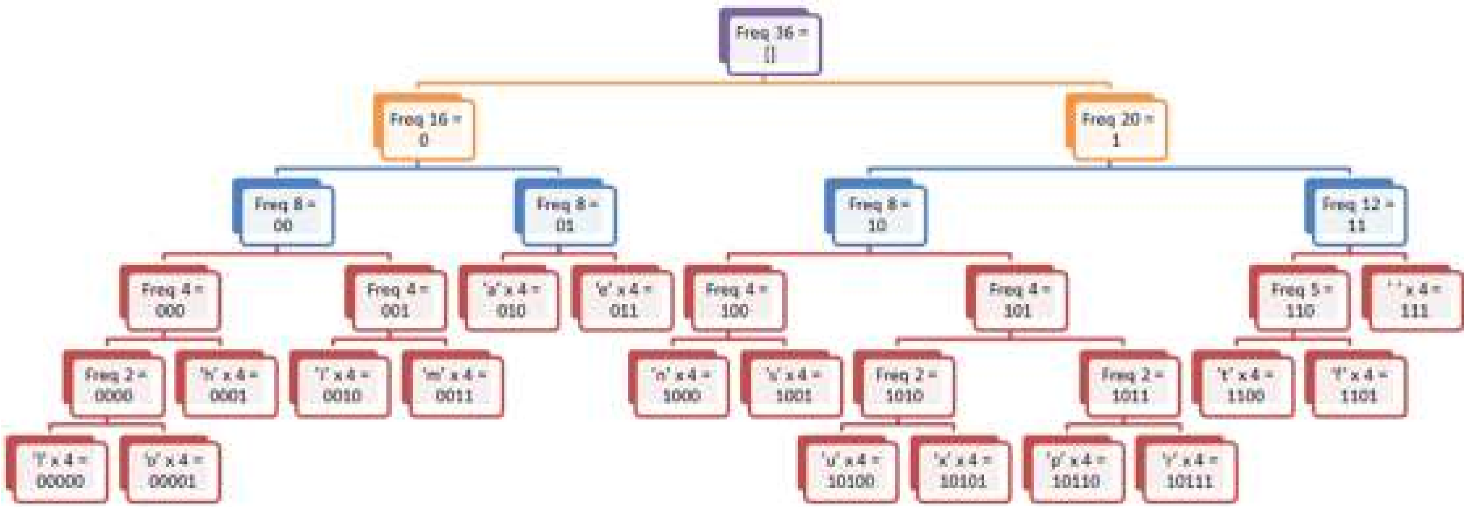
Why is Scaling Hard?

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

- 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

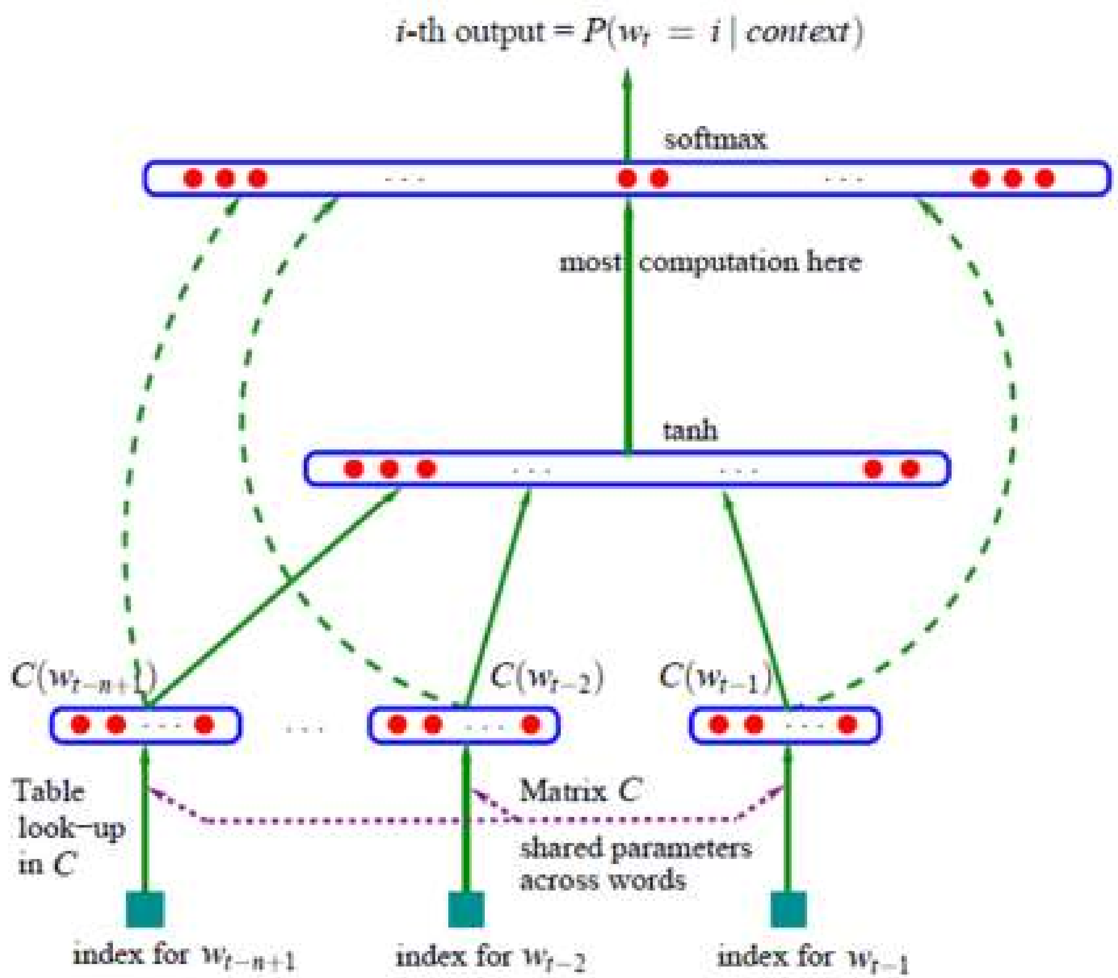


Image: (Bengio et al, 03)

Neural vs Maxent

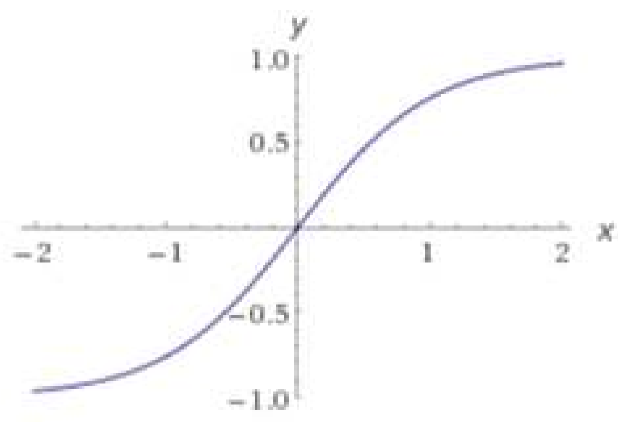
- Maxent LM

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

- Neural Net LM

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

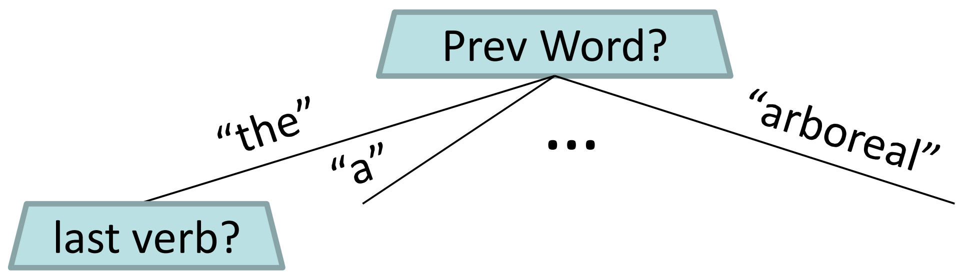
σ nonlinear, e.g. tanh



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