Scaling Hidden Markov Models

April 27, 2021

Latent Variables Models in NLP

- NLP benchmarks are dominated by fully observed models
  - Transformers
  - Previously, Recurrent Neural Networks
- We instead explore latent variable models

Latent Variable Models: Motivation

- LVMs posit a generative process involving unseen variables
- Maintain uncertainty over latent representations, rather than just output correlations
- Often improves interpretability and controllability
- Bottlenecked by the computational complexity of inference

Research Question

To what extent is the performance of tractable latent variable models limited by scale and choices in parameterization?

Language Modeling

To answer this question, we focus on the task of language modeling

- Given the words seen so far, predict the next word
- Language requires modeling long-range phenomena

How now, brown ______

Hidden Markov Models in NLP

- Simplest latent variable models for time series data
- Are thought to be very poor language models
- We show they are better than previously thought, once scaled

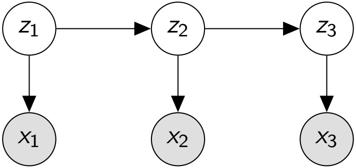
Background: Hidden Markov Models

Hidden Markov Models (HMMs)

- Classical models for unsupervised per-word tag induction in NLP
  - Part-of-speech induction
  - Word alignment for translation
- Generative model separates temporal dynamics from emissions
  - First picks a sequence of latent states
  - Emits words given only the corresponding state

Hidden Markov Models (HMMs)

For times t, model states $z_t \in [Z]$, and tokens $x_t \in [X]$,



This yields the joint distribution

$$p(x, z) = \prod_t p(x_t \mid z_t) p(z_t \mid z_{t-1})$$

with

start state	$p(z_1)$,
transitions	$p(z_t \mid z_{t-1})$,
and emissions	$p(x_t \mid z_t)$

Inference

Given observed $x = (x_1, \dots, x_T)$ We wish to maximize

$$p(x) = \sum_{z_1} \cdots \sum_{z_T} p(x, z) = \alpha_1^\top \Lambda_2 \Lambda_3 \cdots \Lambda_T \mathbf{1},$$

where we have the

$$\begin{aligned} &\text{start,} && [\alpha_1]_{z_1} = p(x_1 \mid z_1) p(z_1), \\ &\text{and transition operators,} && [\Lambda_t]_{z_{t-1}, z_t} = p(x_t \mid z_t) p(z_t \mid z_{t-1}) \end{aligned}$$

The result of each matvec has the alphas of the forward algorithm, i.e. $\alpha_3 = \alpha_1 \Lambda_2 \Lambda_3$ has entries corresponding to $p(z_3, x_{1:3})$

Transition Operators

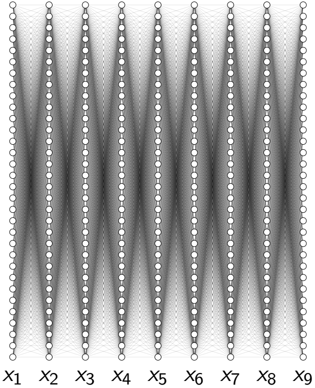
skip slide? mainly to make the connection between distributional parameters and inference really concrete.

- The transition operators α_1, Λ_t are functions of the distributional parameters: starting state π, state transitions A, and emissions O
- In particular

$$\begin{aligned}
&\text{the start} && \alpha_1 = \pi \circ [O]_{\cdot, x_1}, \\
&\text{and the transition operators} && \Lambda_t = A \, \mathrm{diag}([O]_{\cdot, x_t})
\end{aligned}$$

Inference

Given the start and transition operators, we can visualize inference



- Each node corresponds to a state
- Each edge to an entry in the transition operator matrix

Scaling HMMs

Lessons from Large Neural Language Models

Large models perform better but are ...

1. Slow to train
2. Prone to overfitting

We must overcome these issues when scaling HMMs

3 Techniques for Training Large HMMs

- Compact neural parameterization

  ⬆ Generalization

- State dropout

  ⬆ Speed ⬆ Generalization

- Block-sparse emission constraints

  ⬆ Speed

- Will cover a fourth in the second part of this talk

Technique 1: Neural Parameterization

- The transition A and emission O matrices have Z^2 and ZX entries
- Causes the number of parameters to explode as the state size increases
- We instead use a low-dimensional factorization of all conditional distributions that greatly reduces the number of parameters

Neural Parameterization

For both the transition and emission matrices, we use a nonlinear D-dimensional decomposition

$$\boxed{W} \propto \exp\left(\boxed{U} \times \boxed{V^\top}\right)$$

with embeddings $U \in \mathbb{R}^{Z \times D}$, $V \in \mathbb{R}^{Z \times D}$ or $\mathbb{R}^{X \times D}$

- Can further parameterize U or $V = \mathrm{MLP}(E_u)$

Technique 2: State Dropout

- Dropout is a common technique for regularizing neural networks
  - Reduces a network's reliance on a particular neuron
- Extend dropout to the states of an HMM
  - Encourage broad utilization of all states

State Dropout

- At each batch, sample dropout mask $\mathbf{b} \in \{0,1\}^Z$
- Compute distributional parameters by indexing into embeddings U, V

$$\left(\mathbf{b} \circ \boxed{U_A}\right) \times \left(\mathbf{b} \circ \boxed{V_A}\right)^\top$$

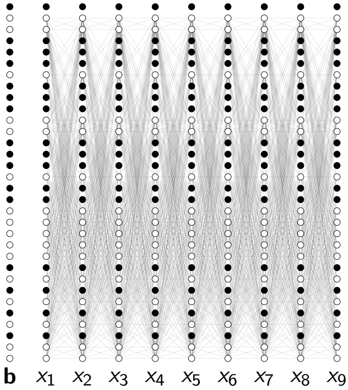
(a) Unnormalized transition logits

$$\left(\mathbf{b} \circ \boxed{U_O}\right) \times \boxed{V_O}^\top$$

(b) Unnormalized emission logits

State Dropout: Inference

The cost of inference is reduced by state dropout



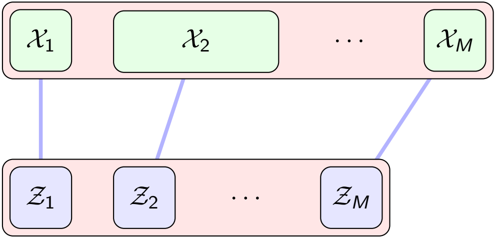
- Shaded nodes depict dropped states
- Ignore dropped states during inference

Technique 3: Block-Sparse Emission Constraints

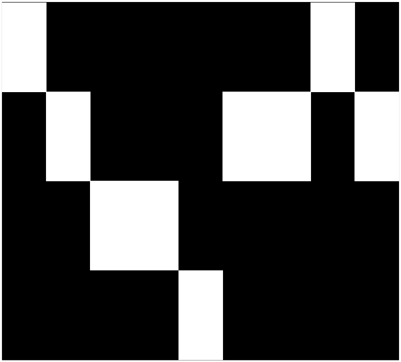
- Reduce cost of marginalization by enforcing structure
- Only allow each word to be emit by a subset of states
- Obtain an alignment by partitioning words and states jointly

Block-Sparse Emission Constraints: Inference

- Indices $m \in [M]$
- State partitions $\mathcal{Z}_m$
- Word partitions $\mathcal{X}_m$

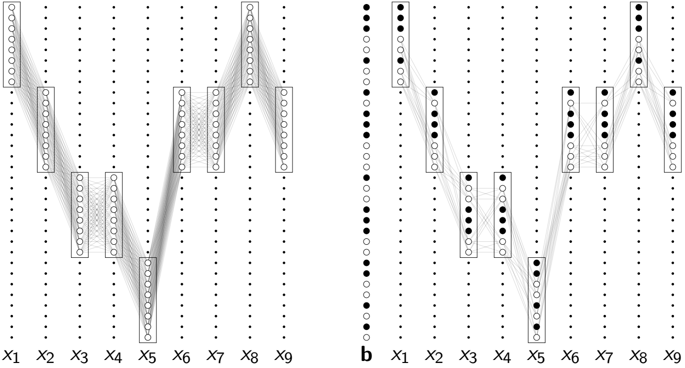


Block-Sparse Emission Constraints



Block-Sparse Emissions: Inference

Given each word x_t, only the states in the correct group can occur



(a) Block-sparse emission (b) With state dropout

Technique 4: Generalized Softmax

Focusing on the transition distribution,

- Softmax

$$p(z_t \mid z_{t-1}) = \frac{\exp(\mathbf{u}_{z_{t-1}}^\top \mathbf{v}_{z_t})}{\sum_z \exp(\mathbf{u}_{z_{t-1}}^\top \mathbf{v}_z)}$$

- Generalized Softmax

$$p(z_t \mid z_{t-1}) = \frac{K(\mathbf{u}, \mathbf{v})}{\sum_z K(\mathbf{u}, \mathbf{v}_z)} = \frac{\phi(\mathbf{u})^\top \phi(\mathbf{v})}{\sum_z \phi(\mathbf{u})^\top \phi(\mathbf{v}_z)},$$

  for positive kernel $K : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}_+$ and feature map $\phi : \mathbb{R}^d \to \mathbb{R}^f$

Generalized Softmax: Inference

- The key $O(Z^2)$ step in the forward algorithm:

$$p(z_t \mid x_{<t}) = \sum_{z_{t-1}} p(z_t \mid z_{t-1}) p(z_{t-1} \mid x_{<t})$$

- In matrix form,

$$\boldsymbol{\gamma}_t = \underbrace{\boldsymbol{\alpha}_{t-1}}_{\mathbb{R}^Z} \underbrace{\Lambda}_{\mathbb{R}^{Z \times Z}},$$

  where we have the probability of the

$$\begin{aligned}
&\text{current state,} && [\boldsymbol{\gamma}_t]_{z_t} = p(z_t \mid x_{<t}), \\
&\text{last state,} && [\boldsymbol{\alpha}_{t-1}]_{z_{t-1}} = p(z_{t-1} \mid x_{<t}), \\
&\text{transition probability,} && [\Lambda]_{z_{t-1}, z_t} = p(z_t \mid z_{t-1})
\end{aligned}$$

Generalized Softmax: Inference

- Use generalized softmax in transition distribution

$$[\Lambda]_{z_{t-1},z_t} = p(z_t \mid z_{t-1}) \propto \phi(\mathbf{u}_{z_{t-1}})^\top \phi(\mathbf{v}_{z_t})$$

- Allows us to apply associative property of matrix multiplication

$$\begin{aligned} \boldsymbol{\gamma}_t &= \boldsymbol{\alpha}_{t-1}\Lambda \\ &= \boldsymbol{\alpha}_{t-1}(\mathrm{diag}(d)\phi(U)\phi(V)^\top) \\ &= \underbrace{(\boldsymbol{\alpha}_{t-1} \circ d)}_{\mathbb{R}^Z} \underbrace{\phi(U)}_{\mathbb{R}^{Z\times f}} \underbrace{\phi(V)^\top}_{\mathbb{R}^{f\times Z}}, \end{aligned}$$

with stacked embeddings $\phi(U), \phi(V) = [\phi(\mathbf{v}_1), \dots, \phi(\mathbf{v}_Z)]$ and normalizing constants d

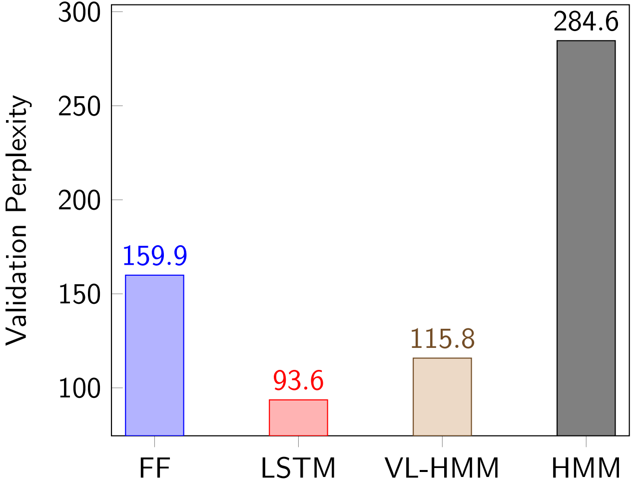
- Takes $O(Zf)$ time from left to right!

Experiments

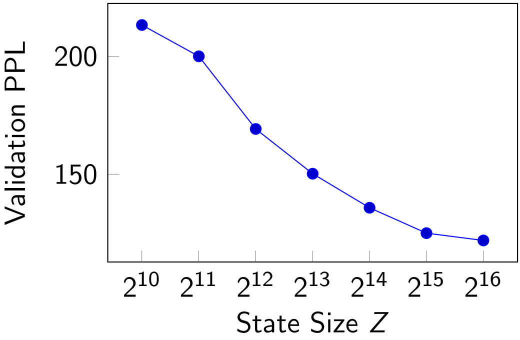
Experiments

- Language modeling on Penn Treebank
- Baselines
  - Feedforward 5-gram model
  - 2-layer LSTM
  - A 900 state HMM (Buys et al 2018)
- Model
  - 2^{15} (32k) state very large HMM (VL-HMM)
  - $M = 128$ groups (256 states per type), obtained via Brown Clustering
  - Dropout rate of 0.5 during training

Results on PTB Validation Data



State Size Ablation



Validation perplexity on PTB by state size ($\lambda = 0.5$ and $M = 128$)

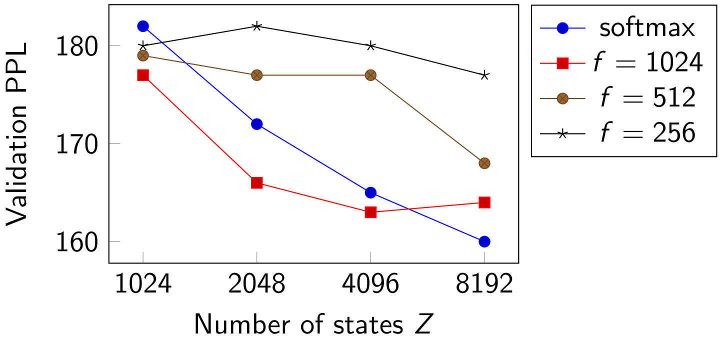
Other Ablations

Model	Param	Train	Val
VL-HMM (2^{14})	7.2M	115	134
- neural param	423M	119	169
- state dropout	7.2M	88	157

Experiments

- Language modeling on PTB
- Work directly with feature map $\phi(\mathbf{x}) = \exp\left(W\mathbf{x} - \frac{\|x\|^2}{2}\right)$, with learned $W \in \mathbb{R}^{d \times f}$
- No dropout or sparsity constraints

Results on PTB Validation



- Holding number of features fixed, perplexity mostly improves or remains the same with an increasing number of states
- Achieve similar performance as softmax with around 4:1 state to feature ratio (also holds for 8k and 16k states)

Conclusion

- Hopeful that HMMs can be competitive language models
- Introduced 4 techniques for tackling speed and overfitting
- Future work will extend to other discrete latent variable models

EOS

Citations