Scaling Hidden Markov Models

April 24, 2021

Latent Variables Models in NLP

Chicken and egg problem:

- NLP benchmarks are dominated by fully observed models
- Most tasks are fully supervised
- Evaluation of unsupervised tasks is also difficult
- Goal: Demonstrate efficacy of latent variable models by making them competitive on an existing task

Language Modeling

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

PICTURE

Research Question

- How far can we scale simple latent variables models?
- Under the assumption that tasks will only be developed if models are reasonably performant

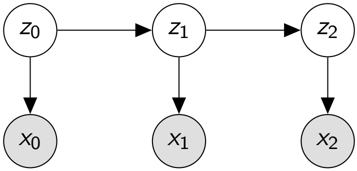
Hidden Markov Models in NLP

- 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: HMMs

HMMs

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



- Joint distribution $p(x, z) = \prod_t p(x_t \mid z_t) p(z_t \mid z_{t-1})$
- Start vector $\pi \in [0, 1]^Z$, with $[\pi]_{z_0} = p(z_0)$
- Transition matrix $A \in [0, 1]^{Z \times Z}$, with $[A]_{z_{t-1}, z_t} = p(z_t \mid z_{t-1})$
- Emission matrix $O \in [0, 1]^{Z \times X}$, with $[O]_{z_t, x_t} = p(x_t \mid z_t)$

Inference

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

$$p(x) = \sum_{z_1} \cdots \sum_{z_T} p(x, z),$$

computed via the forward algorithm

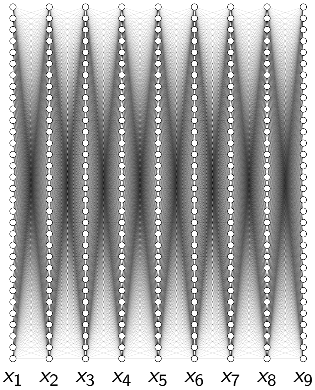
- Start $\alpha_1 = \pi[O]_{\cdot, x_1}$
- Transition operators $\Lambda_t = A \, \mathrm{diag}([O]_{\cdot, x_t}) \in [0, 1]^{Z \times Z}$
- Forward algorithm computes evidence

$$p(x) = \underbrace{\alpha_1^\top \Lambda_2 \Lambda_3}_{\alpha_3} \cdots \Lambda_T \mathbf{1}$$

- Each $[\alpha_t]_{z_t} = p(z_t, x_{\leq t})$

Inference

- Nodes correspond to states
- Edges to entries in Λ_t
- Sequentially compute posterior state probabilities



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

4 Techniques for Training Large HMMs

- Compact neural parameterization

  ⇧ Generalization

- State dropout

  ⇧ Speed ⇧ Generalization

- Block-sparse emission constraints

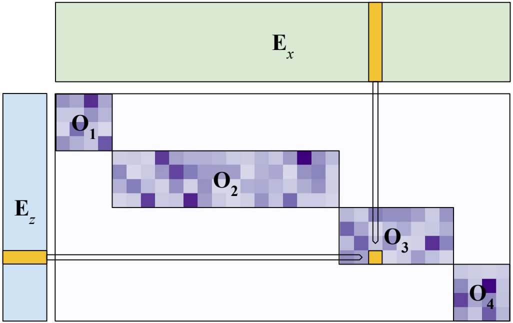
  ⇧ Speed

- Kernel-based generalized softmax

  ⇧ Speed

Technique 1: Neural Parameterization

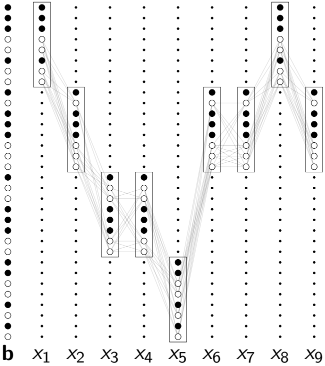
- A neural parameterization allows for parameter sharing
- Generate conditional distributions from state $\mathbf{E}_z$ and token representations $\mathbf{E}_x$



REDO. picture should show $p(x \mid z) = \exp(LR^\top)$, generating L and R from neural networks

Technique 2: State Dropout

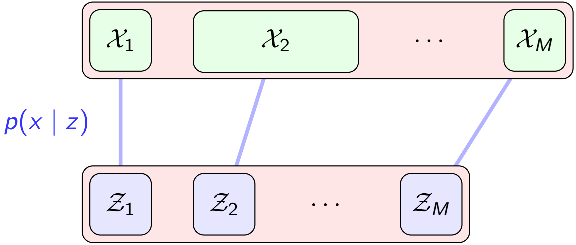
- State dropout encourages broad state usage
- At each batch, sample dropout mask $\mathbf{b} \in \{0,1\}^Z$



Technique 3: Block-Sparse Emission Constraints

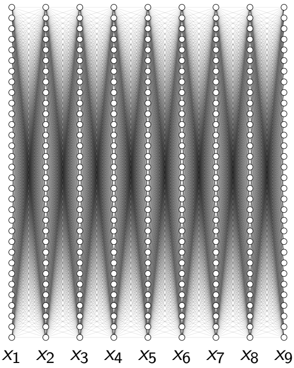
This slide sucks, redo

- Reduce cost of marginalization by enforcing structure
- Partition words and states jointly
- Words can only be emit by states in the aligned group

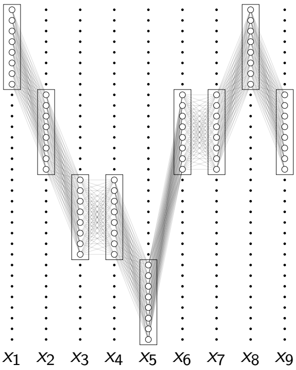


Block-Sparse Emissions: Effect on Inference

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



(a) No constraints



(b) Block-sparse emission

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,

$$\gamma_t = \underbrace{\alpha_{t-1}}_{\mathbb{R}^Z} \underbrace{\Lambda}_{\mathbb{R}^{Z \times Z}},$$

  where we have the probability of the

current state,	$[\gamma_t]_{z_t} = p(z_t \mid x_{<t})$,
last state,	$[\alpha_{t-1}]_{z_{t-1}} = p(z_{t-1} \mid x_{<t})$,
transition probability,	$[\Lambda]_{z_{t-1}, z_t} = p(z_t \mid z_{t-1})$

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

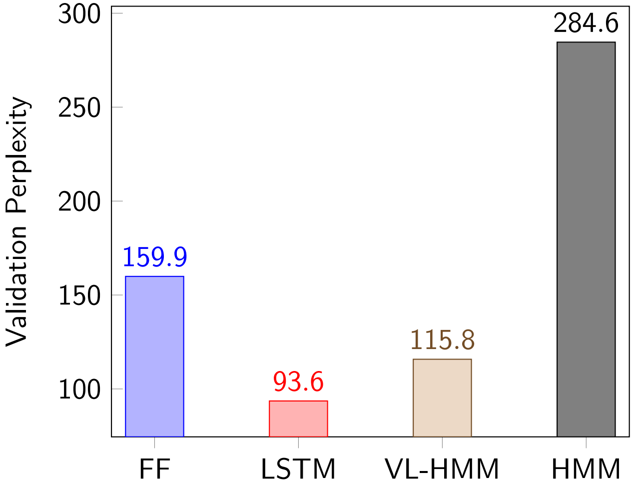
- Takes $O(Zf)$ time from left to right!

Experiments

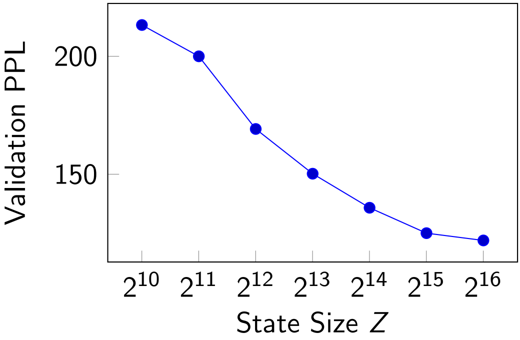
Experiments

- 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 $M = 128$)

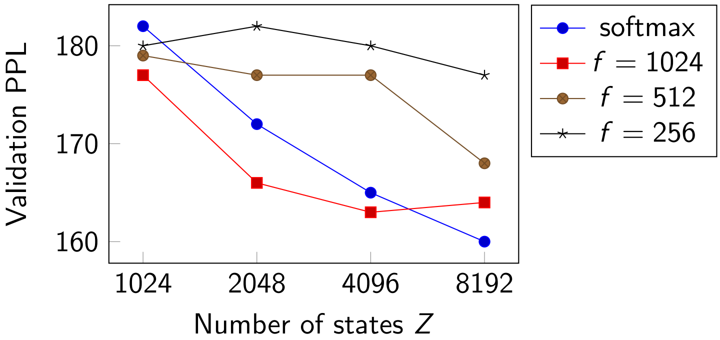
Other Ablations

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