Scaling Hidden Markov Models

March 12, 2021

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 not!

Scaling HMMs with Emission Sparsity

Lessons from Large Neural Language Models

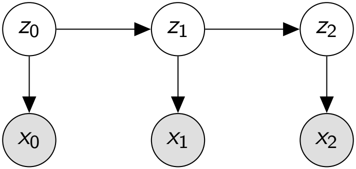
Large models perform better but are ...

1. Slow to train
2. Prone to overfitting

We must overcome these issues when scaling HMMs

HMMs

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



We wish to optimize

$$p(x) = \sum_z p(x, z),$$

computed via the forward algorithm

3 Techniques for Training Large HMMs

- Block-sparse emission constraints

  ⇧ Speed

- Compact neural parameterization

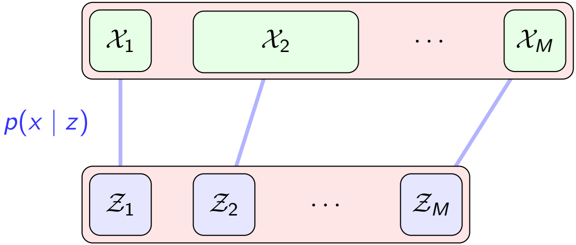
  ⇧ Generalization

- State dropout

  ⇧ Speed ⇧ Generalization

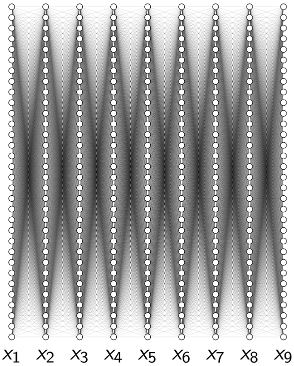
Technique 1: Block-Sparse Emission Constraints

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

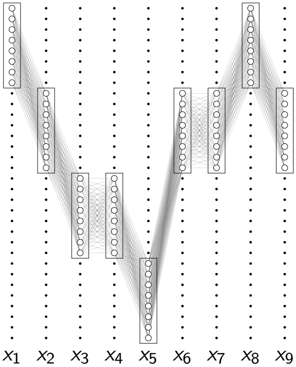


Block-Sparse Emissions: Effect on Inference

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



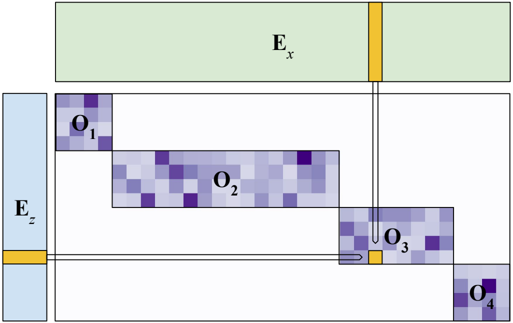
(a) No constraints



(b) Block-sparse emission

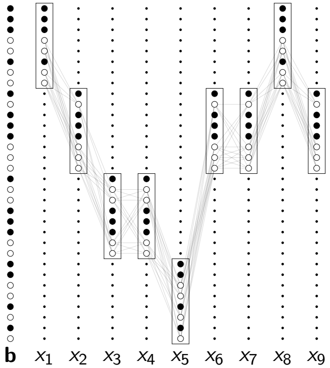
Technique 2: Neural Parameterization

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



Technique 3: State Dropout

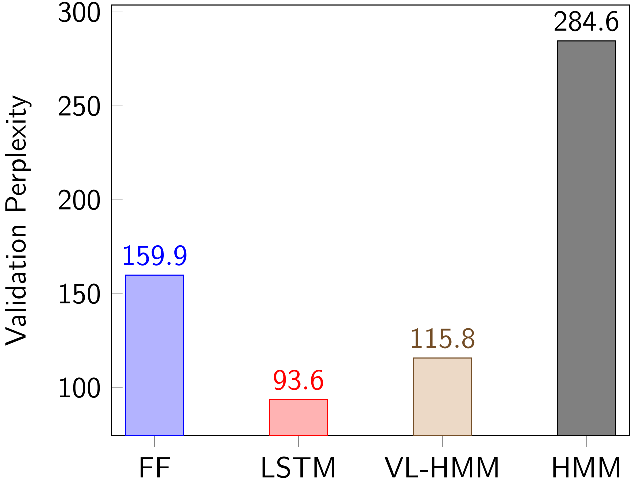
- State dropout encourages broad state usage
- At each batch, sample dropout mask $\mathbf{b} \in \{0, 1\}^Z$



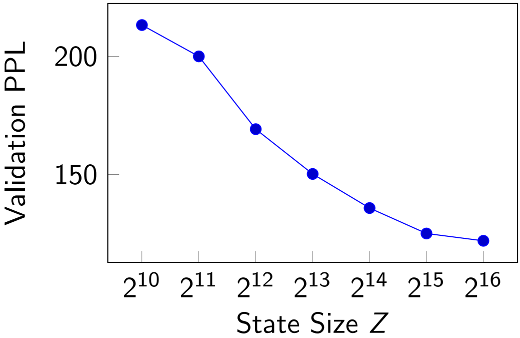
Experiments

- 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

Scaling HMMs with Kernel Methods

Embedded Structure Prediction

- The previous work relied on emission sparsity constraints
- Can we scale with weaker assumptions?
- Reduce the quadratic dependence of inference on number of states to linear with kernel trick

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 $\phi : \mathbb{R}^d \to \mathbb{R}^f$

Inference

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

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

- In matrix form,

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

where

$$\begin{aligned} [\gamma_t]_{z_t} &= p(z_t \mid x_{<t}) \\ [\boldsymbol{\alpha}_{t-1}]_{z_{t-1}} &= p(z_{t-1} \mid x_{<t}) \\ [\Lambda]_{z_{t-1}, z_t} &= p(z_t \mid z_{t-1}) \end{aligned}$$

Embedded Inference

- Use generalized softmax in transition distribution

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

- In matrix form,

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

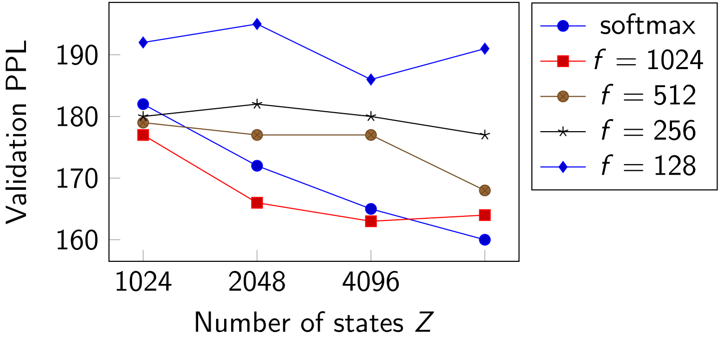
  with stacked embeddings $\phi(U), \phi(V)$ and normalizing constants d

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

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 improves or remains the same with an increasing number of states

Conclusion

- HMMs are competitive language models
- Introduced 4 techniques for tackling speed and overfitting
- A great time to revisit other discrete latent variable models

EOS

Citations