Scaling Hidden Markov Language Models

J Chiu and Y Deng and A Rush

May 12, 2021

Language Modeling

> *'BERT and GPT models change the game for NLP'*
>
> Demi Ajayi (IBM)

Able to achieve strong performance in modern NLP tasks with

- Large, opaque models
- Pretrained via language modeling on large data
- Fine-tuned for downstream task

Language modeling either has useful information for or is a necessary component of downstream tasks

Language Modeling

How now, brown ______

- Given the words seen so far, predict the next word
- Requires encoding long-range context
- Most success with uninterpretable models

Interpretable Models

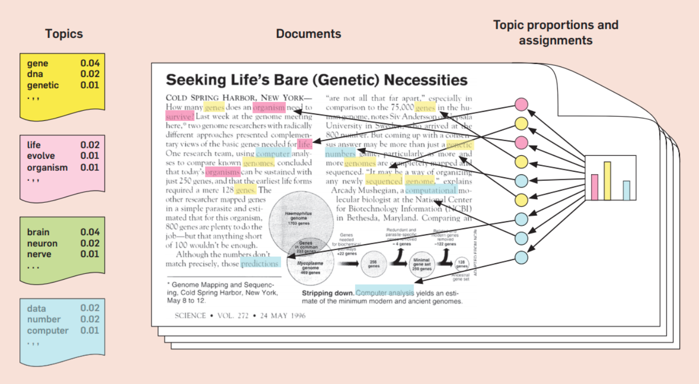
- Interpretable models give qualitative insight into data
- Generative models provide interpretability through stories
- Generative stories provide intermediate decisions

Interpretable Models: Examples

Examples of interpretable generative models:

- Topic Model
- Template Model
- Entity Language Model

Interpretable Models: Topic Model[1]

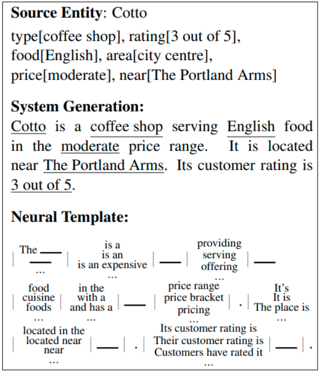


Answers the question

- What is this document broadly about?

[1]Blei, Ng, and Jordan, 'Latent dirichlet allocation'.

Interpretable Models: Template Model[2]



Answers the question

- Where did the information in this text come from?

[2]Wiseman, Shieber, and Rush, 'Learning Neural Templates for Text Generation'.

Interpretable Models: Entity Model[3]

[*John*]$_1$ wanted to go to [*the coffee shop*]$_2$ in [*downtown Copenhagen*]$_3$. [*He*]$_1$ was told that [*it*]$_2$ sold [*the best beans*]$_4$.

Answers the question

- Who is this text talking about?

[3] Ji et al., 'Dynamic Entity Representations in Neural Language Models'.

Performance of Interpretable Generative Models

Generative stories are interpretable when they

- Decompose data into a sequence simple decisions

Generative stories provide interpretability at a cost

- Must consider all alternatives for unobserved decisions

Can we match the performance of uninterpretable language models while maintaining intepretability?

Hidden Markov Language Models

- Interpretability as a design decision
- Focus on first-order HMMs, where context is encoded as a single integer
- Thought to be poor language models

Research Question

To what extent is the performance of HMMs limited by scale and choices in parameterization?

This work: Scale HMMs on language modeling using techniques drawn from recent advances in neural networks

Background: Hidden Markov Models

Hidden Markov Models (HMMs)

- Classical models for unsupervised per-word tag induction
  - Part-of-speech induction[4]
  - Word alignment for translation[5]
- Admits tractable exact inference
  - Strong conditional independence assumptions
  - Simple transition dynamics
  - Finite set of discrete latent states

[4] Merialdo, 'Tagging English Text with a Probabilistic Model'.

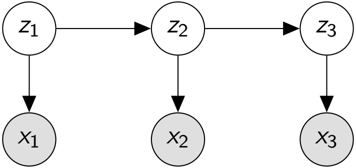
[5] Vogel, Ney, and Tillmann, 'HMM-Based Word Alignment in Statistical Translation'.

HMM State Sizes

Year	Data	Model	States
1989	Phoneme Segmentation	HMM	7
1994	POS	HMM	76
2005	Activity Monitoring	DMC HMM	1k
2006	2D Image Tracking	Convolutional HMM	100k
2009	LM	Split-POS HMM	450
2016	POS	Neural HMM	37
2016	Web Traffic Analysis	FFT HMM	81
2019	Char LM	Cloned HMM	30k

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	$p(x_t \mid z_t)$

represented as vectors and matrices

Inference

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

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

where we have the

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

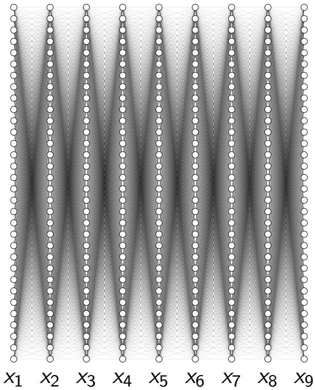
- Matrix representation of forward algorithm:

$$\alpha_t = \alpha_{t-1} \Lambda_t$$

- Requires $O(TZ^2)$ operations in total!

Inference

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



- 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

- Transition and emission matrices have Z^2 and ZX entries
- More states lead to explosion in parameter count
- Solution: Low dimensional factorization

Neural Parameterization: Softmax Parameterization

The transition matrix A is factorized as follows:

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

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

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

Technique 2: State Dropout

- Dropout is a common technique for regularizing neural networks[6]
  - Reduces a network's reliance on any particular neuron by via random masking
- Extend dropout to the states of an HMM
  - Encourage broad utilization of all states

[6] Srivastava et al., 'Dropout: A Simple Way to Prevent Neural Networks from Overfitting'.

State Dropout

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

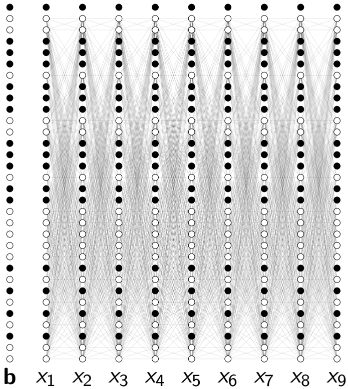
$$\left(\mathbf{b} \circ \boxed{U_{\text{trans}}}\right) \times \left(\mathbf{b} \circ \boxed{V_{\text{trans}}}\right)^\top$$

(a) Unnormalized transition logits

$$\left(\mathbf{b} \circ \boxed{U_{\text{emit}}}\right) \times \boxed{V_{\text{emit}}}^\top$$

(b) Unnormalized emission logits

State Dropout: Inference



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

Technique 3: Block-Sparse Emission Constraints

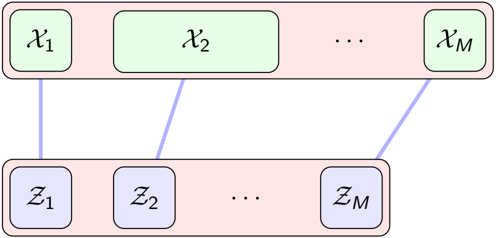
- Reduce cost of marginalization by enforcing structure
- Introduce emission constraints inspired by Cloned HMMs[7]
- Only allow each word to be emit by a subset of states
- Cost of inference is quadratic in the size of the largest subset due to sparsity

[7] Dedieu et al., *Learning higher-order sequential structure with cloned HMMs*.

Block-Sparse Emission Constraints: Alignment

Start with a joint partitioning of both states and words

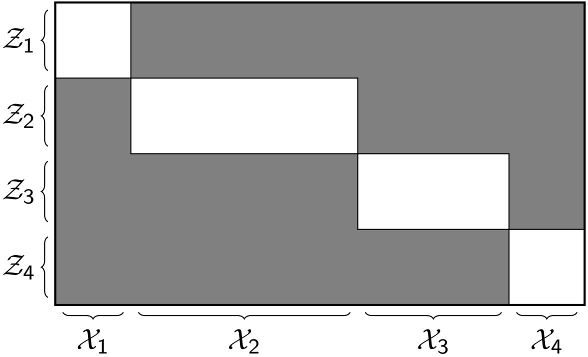
Indices $m \in [M]$ State partitions $\mathcal{Z}_m$ Word partitions $\mathcal{X}_m$



Block-Sparse Emission Constraints

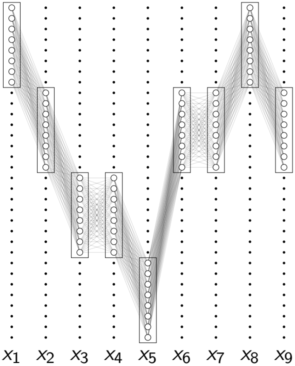
Given the unnormalized emission logits,

- Mask out unaligned state-word entries
- Normalize rows across words in aligned partition

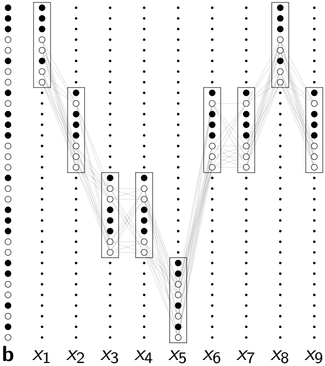


Block-Sparse Emissions: Inference

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



(a) Block-sparse emission



(b) With state dropout

Method Recap

- Compact neural parameterization

  ⇧ Generalization

- State dropout

  ⇧ Speed ⇧ Generalization

- Block-sparse emission constraints

  ⇧ Speed

- A fourth after experiments

Experiments

Experiments

- Language modeling on Penn Treebank

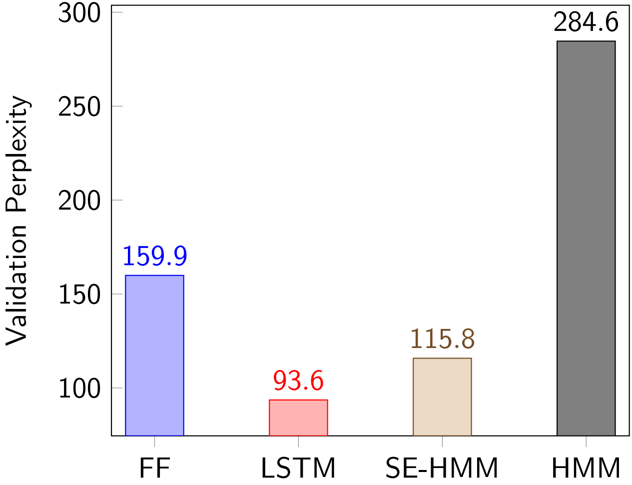
- Evaluate perplexity
  - Function of $p(x)$
  - Lower is better

- Baselines
  - Feedforward 5-gram model
  - 2-layer LSTM
  - A 900 state HMM[8]

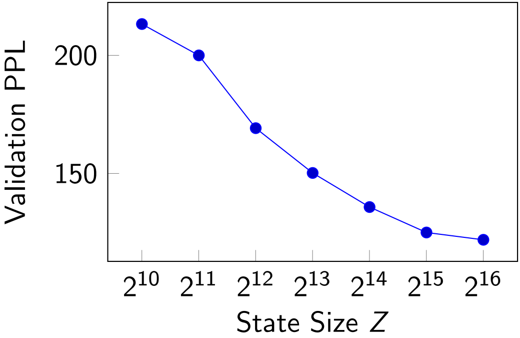
- Model
  - 2^{15} (32k) state sparse emission HMM (SE-HMM)
  - $M = 128$ groups (256 states per group), obtained via Brown Clustering
  - Dropout rate of 0.5 during training

[8] Buys, Bisk, and Choi, 'Bridging HMMs and RNNs through Architectural Transformations'.

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
SE-HMM (2^{14})	7.2M	115	134
- neural param	423M	119	169
- state dropout	7.2M	88	157

Discussion

- Greatly scaled the state size of HMMs
- Performance improved with increasing state size
- Still a large gap between RNNs and HMMs
- Does the emission sparsity constraint improve computation complexity at the price of accuracy?

Speeding up HMMs with Low-Rank Factorizations

A work in progress

Fast Inference with Low-Rank Factorizations

- The previous approach relied a pre-specified emission sparsity constraint
- Can we scale inference with a weaker constraint?
- Exploit structure in the transition matrix to speed up inference

Inference

Start by unpacking inference to reveal the most expensive step

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

with

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

Inference

Decompose transition operators into transition matrix A and emission matrix O

$$
\begin{aligned}
p(x) &= \alpha_1^\top \Lambda_2 \cdot \Lambda_T \mathbf{1} \\
&= \alpha_1^\top (A \operatorname{diag}([O]_{\cdot,x_2})) \cdots \Lambda_T \mathbf{1} \\
&= \alpha_1^\top A \operatorname{diag}([O]_{\cdot,x_2}) \cdots A \operatorname{diag}([O]_{\cdot,x_T}) \mathbf{1}
\end{aligned}
$$

where the most expensive steps are the matrix-vector products $\alpha_t^\top A$, which take $O(Z^2)$ computation

Fast Matrix-Vector Products

- Goal is to reduce the naive matvec complexity of $O(Z^2)$
- Various methods
  - Sparsity (nnz entries)
  - Fast Fourier Transform ($Z \log Z$)
  - Low-Rank factorization (ZR)
- We utilize low-rank factorizations
- Connected to work in efficient attention and kernel approximations[9]

[9] Choromanski et al., *Rethinking Attention with Performers*; Peng et al., *Random Feature Attention*; Blanc and Rendle, *Adaptive Sampled Softmax with Kernel Based Sampling*.

Low-Rank Factorization

Factor transitions $A \in [0,1]^{Z \times Z}$ into product of $U, V \in \mathbb{R}^{Z \times R}$

$$\boxed{\alpha^\top} \times \boxed{A} = \left(\boxed{\alpha^\top} \times \boxed{U} \right) \times \boxed{V^\top}$$

resulting in two matrix-vector products of cost $O(ZR)$ each

- Constraint: Entries of A must be nonnegative
- Solution: Use a nonnegative matrix factorization (NMF)

$$A = \phi(U)\phi(V)^\top,$$

with $\phi : \mathbb{R}^{Z \times R} \to \mathbb{R}_+^{Z \times R}$

Method Recap

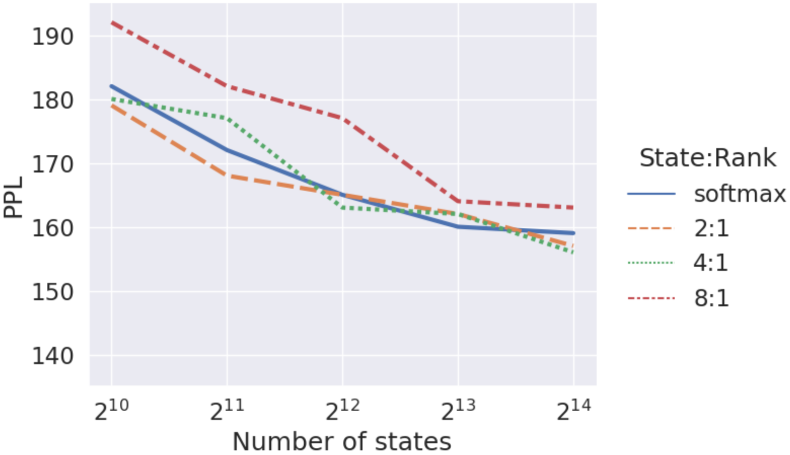
- Target key $O(Z^2)$ matvec step in inference
- Use NMF to reduce cost to $O(ZR)$
- How small can R be relative to Z without sacrificing accuracy?

Experiments

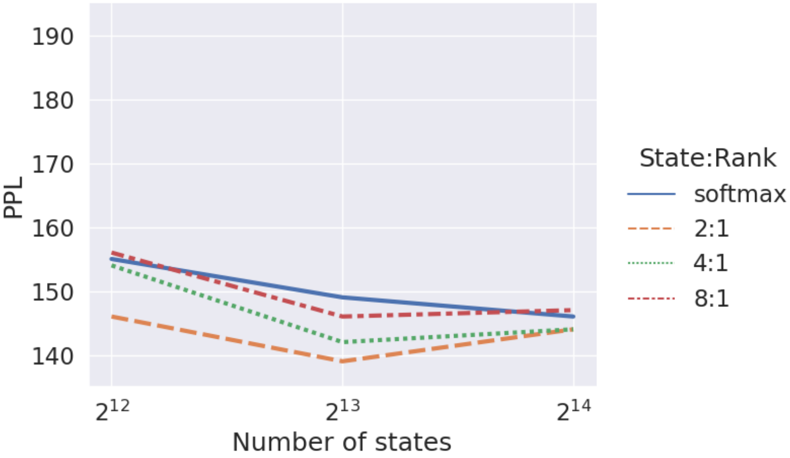
Experiments

- Language modeling on PTB
- Feature map $\phi(U) = \exp(UW)$, with learned $W \in \mathbb{R}^{R \times R}$
- Baseline: Softmax HMM

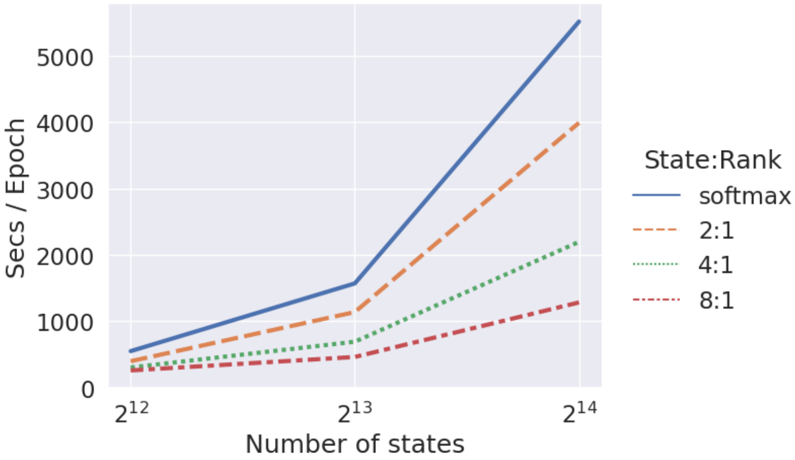
Scaling on PTB (Validation)



Further Scaling on PTB with Dropout (Validation)



Speed Comparison[10]



[10] 2^{14} (16k) state SE-HMM takes 506 s/epoch on the same data

Discussion

- Reduced computation complexity of inference by 4x with NMF vs softmax HMM
- Scaling factor not as large as SE-HMM
- Validation PPL worse than SE-HMM

Conclusion

- Extended techniques from neural networks to HMMs
- Sped up inference using structure in both the emission and transition matrices
- Demonstrated improvements in perplexity with larger state spaces

Future Work

- Explore the performance of more complex interpretable models
  - Hierarchical HMMs
  - Factorial HMMs
  - Probabilistic context-free grammars
  - Switching linear dynamical systems[11]
  - Latent vector grammars[12]
- Explore other structure for fast matrix-vector products and tensor generalizations
  - FFT-inspired algorithms[13]
- Other forms of regularization for HMMs
  - Diversity with DPPs[14]
- Learn sparsity constraints in SE-HMM
- Apply sparsity constraints to embedded HMMs[15] for use in approximate inference

[11] Foerster et al., 'Intelligible Language Modeling with Input Switched Affine Networks'.

[12] Zhao, Zhang, and Tu, 'Gaussian Mixture Latent Vector Grammars'.

[13] Dao et al., 'Kaleidoscope: An Efficient, Learnable Representation For All Structured Linear Maps'.

[14] Qiao et al., 'Diversified Hidden Markov Models for Sequential Labeling'.

EOS

Citations

Blanc, Guy and Steffen Rendle. *Adaptive Sampled Softmax with Kernel Based Sampling*. 2018. arXiv: 1712.00527 [cs.LG].

Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 'Latent dirichlet allocation'. In: *J. Mach. Learn. Res.* 3 (2003), pp. 993–1022. ISSN: 1532-4435. DOI: http://dx.doi.org/10.1162/jmlr.2003.3.4-5.993. URL: http://portal.acm.org/citation.cfm?id=944937.

Buys, Jan, Yonatan Bisk, and Yejin Choi. 'Bridging HMMs and RNNs through Architectural Transformations'. In: 2018.

Choromanski, Krzysztof et al. *Rethinking Attention with Performers*. 2021. arXiv: 2009.14794 [cs.LG].

Dao, Tri et al. 'Kaleidoscope: An Efficient, Learnable Representation For All Structured Linear Maps'. In: *International Conference on Learning Representations*. 2020. URL: https://openreview.net/forum?id=BkgrBgSYDS.

Dedieu, Antoine et al. *Learning higher-order sequential structure with cloned HMMs*. 2019. arXiv: 1905.00507 [stat.ML].

Foerster, Jakob N. et al. 'Intelligible Language Modeling with

Generalized Softmax

- 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 $\phi : \mathbb{R}^D \to \mathbb{R}^R$

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,

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

where we have the probability of the

current state,	$[\gamma_t]_{z_t} = p(z_t \mid x_{<t})$,
last state,	$[\boldsymbol{\alpha}_{t-1}]_{z_{t-1}} = p(z_{t-1} \mid x_{<t})$,
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 $\phi(U), \phi(V) = [\phi(\mathbf{v}_1), \dots, \phi(\mathbf{v}_Z)]$ and normalizing constants d

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