Sequence Models: RNNs

CPSC 440/550: Advanced Machine Learning

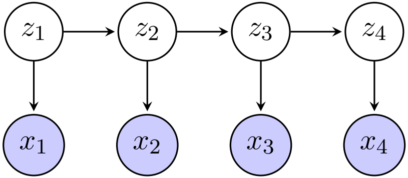
`cs.ubc.ca/~dsuth/440/24w2`

University of British Columbia, on unceded Musqueam land

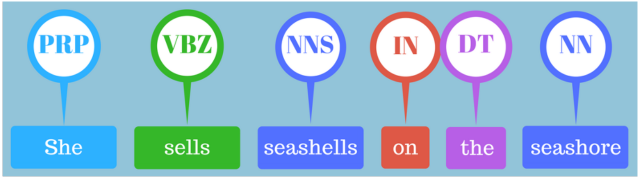
2024-25 Winter Term 2 (Jan–Apr 2025)

Hidden Markov Models for Text Processing

- Recall hidden Markov models, with a hidden Markov chain + observations



- One (formerly) common application: part-of-speech tagging



- Input is a sequence of words
- Output is a categorical label for each word
  - Usual analyses of English have up to about 40 categories
  - Some dependencies (adverbs need to be “attached” to a verb)

Review: representing words

review

- How do we represent words in a useful way?
- One common approach: one-hot
  - Choose your vocabulary of V possible words
  - Give the first word the feature vector $(1, 0, 0, \ldots, 0) \in \mathbb{R}^V$
  - Give the second word the feature vector $(0, 1, 0, \ldots, 0) \in \mathbb{R}^V$, . . .
  - $\mathbf{X}W$ for a $V \times k$ matrix W is $\begin{bmatrix} -w_1- \\ \vdots \\ -w_n- \end{bmatrix}$
    - Usually implemented as a lookup table (`torch.nn.Embedding`)
- Almost the same: choose uniformly from the unit sphere in $\mathbb{R}^k$
  - Dot products will be almost zero even with $k \ll V$ (Johnson–Lindenstrauss lemma)
- Latent-factor models like word2vec, GloVe, fasttext
  - Unsupervised learning of features in $\mathbb{R}^k$ that try to “mean something”

Individual-word classifier

- We could try to do part-of-speech tagging by looking at one word at a time
- Implement as nearest-neighbour, linear classifier, neural net...
  - $\mathrm{POS}(\texttt{she}) = \texttt{PRP}$
  - $\mathrm{POS}(\texttt{desert}) = \ldots$
    - "Don't desert me in the desert!"
- Problem: this isn't always enough information to tell!

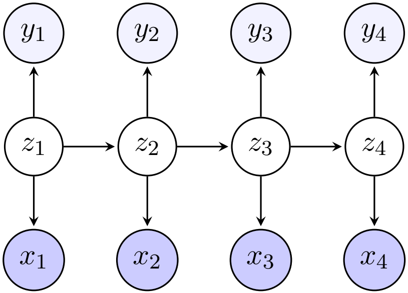
Hidden Markov-type model for part-of-speech tagging

- Hidden states are our part-of-speech tags
- $p(\texttt{determiner} \to \texttt{adjective}) = 0.27$, $p(\texttt{determiner} \to \texttt{noun}) = 0.51$
- Given a fully-labeled dataset, MLE/MAP training is easy
  - Training a fully-observed categorical DAG: just count all conditionals
- To classify, run a variant of Viterbi decoding to find most likely sequence of tags
  - Full inference can be helpful: "The old man the boat."

- Problem: the Markov structure / number of states might not be sufficient
- Markov structure doesn't understand many grammatical constraints
- Example: often we want to distinguish proper nouns vs. normal ones, but:
  - "Turkey will make a good sandwich."
  - "Turkey will host a political conference."

More complex graphical models

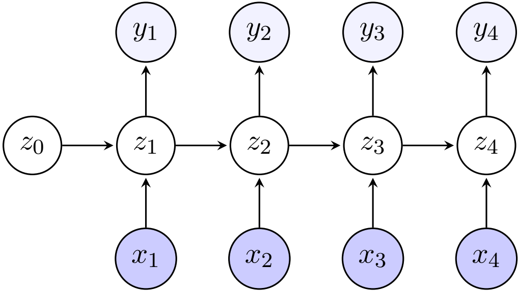
- We could draw a more complex graphical model:



  - x_t are the words, always observed
  - y_t are the POS tags, observed in training but not at inference time
  - h_t are always-hidden states that contain POS information "plus more"
- It'll be hard to do this well with discrete states, tabular transitions
- What about continuous state with deep net-parameterized transitions?
- Can do this generative model, but learning and inference are now both hard

RNNs

- Things are a little easier if we turn to a discriminative model

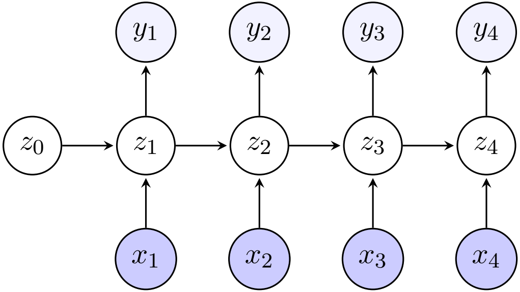


- Usually the hidden states $z_t = f_i(x_t, z_{t-1})$ are given by deterministic neural nets
- The outputs $y_t \sim \mathrm{Cat}(g_i(z_i))$ have probabilities parameterized by another net

- f and g are usually time-invariant (homogeneous): $f_t = f$, $g_t = g$
  - Allows us to handle sequences of different lengths
  - Many fewer parameters, or equivalently can tie many parameters
  - Will be harder (but not impossible) to have behaviour vary through time

RNN inference

- No need to "go backwards" on any arrows, so inference is easy:



- Typically z_0 is a fixed vector
- Compute $z_1 = f(x_1, z_0)$ with a network forward pass; sample $y_1 \sim \mathrm{Cat}(g(z_1))$
- Compute $z_2 = f(x_2, z_1)$ with a network forward pass; sample $y_2 \sim \mathrm{Cat}(g(z_2))$
- ...
- Constant cost per item in the sequence, but need to go sequentially

RNN learning

- Training data: n sequences $(x_1^{(i)}, \ldots, x_{T^{(i)}}^{(i)}), (y_1^{(i)}, \ldots, y_{T^{(i)}}^{(i)})$
- Can train by minimizing (possibly regularized) NLL:

$$\arg\min_{f,g} - \sum_{i=1}^{n} \sum_{t=1}^{T^{(i)}} \log p(y_t^{(i)} \mid x_{1:t}^{(i)}, f, g)$$

- Computing gradients is sometimes called backpropagation through time
    - Exactly the same as usual backprop/autodiff, as long as you handle parameter tying
- Usually trained with SGD; all the usual deep learning challenges, plus. . .

RNN learning: extra challenges

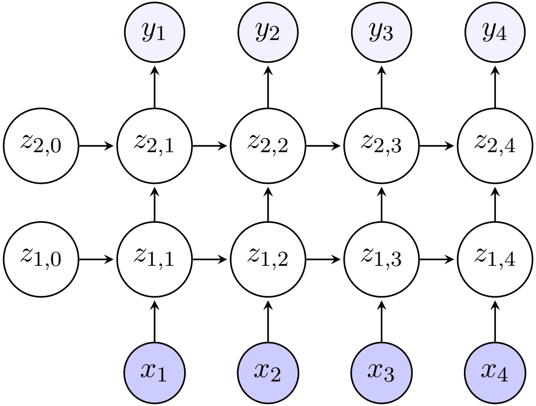
- Memory cost grows with sequence length
  - For long sequences, there are a lot of intermediate terms
  - Changing f affects all the z_t at once
- Parameter tying often leads to vanishing/exploding gradients
  - Illustration: say we use a (silly) linear RNN that ignores the inputs:

$$f(x_t, z_{t-1}) = U z_{t-1} \qquad \text{so} \quad z_T = U U \cdots U z_0 = U^T z_0$$

  - Usually, z_T either diverges exponentially or collapses to zero exponentially
    - If largest singular value of U is > 1, then $\|z_t\|$ explodes with t
    - If largest singular value of U is < 1, then $\|z_t\| \to 0$ with t
  - For more realistic RNNs, same problem happens (but a little more complicated)
- "Default SGD" tends to not work well
  - Adam can help
  - So can gradient clipping: if $\|g\| > u$, use $g \cdot u / \|g\|$ instead
  - Special parameterizations for U so that all singular values are 1: mixed results

Deep RNNs

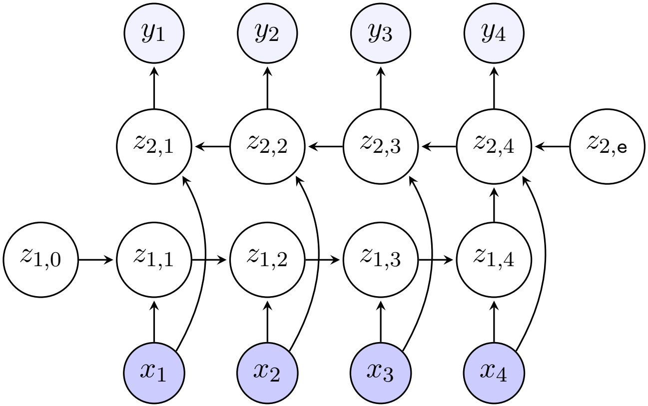
- Can have multiple hidden layers per time:



- Inference goes "up and right"; still a DAG
- Might be easier to model having multiple timescales of effects

Bi-Directional RNNs

- Sometimes inference "needs to go backward"; regular RNNs can't do that
  - "I've had a perfectly wonderful evening, but this wasn't it." ("paraprosdokian")
  - "The old man the boat." ("garden path sentence")



Summary

- Sequence modeling: can use hidden Markov models or similar variants
- But a discriminative version, RNNs, is easier to use complex states
  - Easy inference: forward-only
  - Easy to compute likelihoods
  - Optimization challenges: memory, vanishing/exploding gradients

- Next time: fancier sequence models to do more varied things