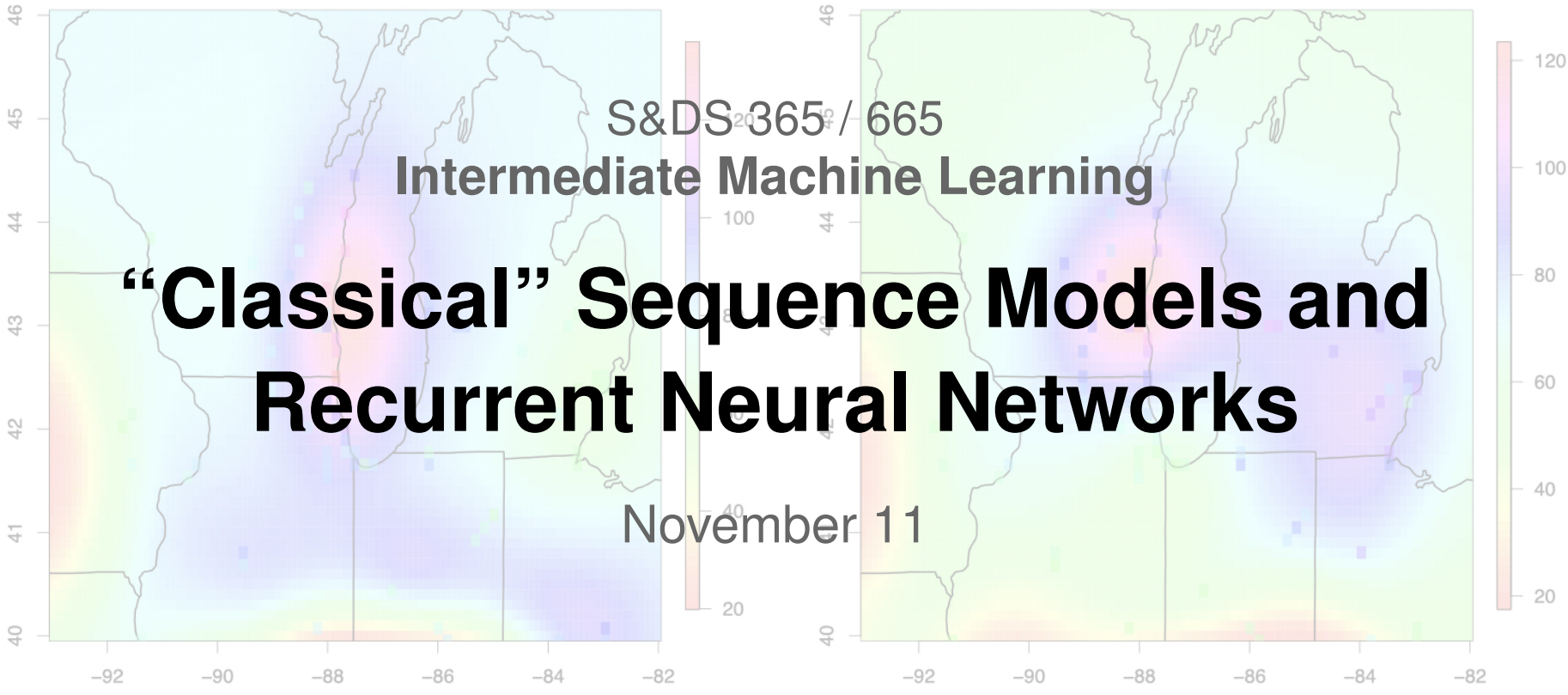


S&DS 365 / 665

Intermediate Machine Learning

“Classical” Sequence Models and Recurrent Neural Networks

November 11

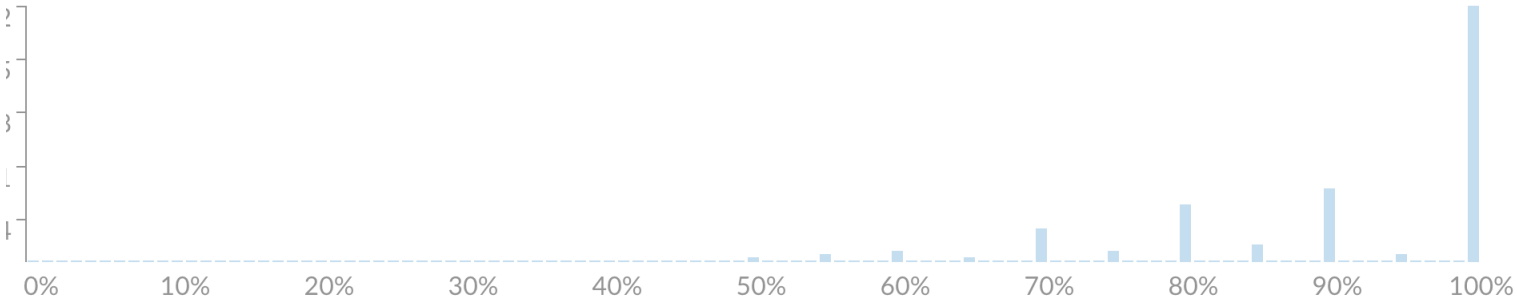
Yale

Reminders

- Final exam: Sunday, Dec 15, 2pm LC 102
- Assignment 4 is out, due next Monday
  - Graph neural nets
  - Policy gradient
  - Deep Q-Learning
- Assignment 5 (last!) posted Wednesday

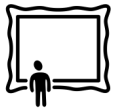
Quiz 4

Average Score	High Score	Low Score	Standard Deviation	Average Time
91%	100%	50%	1.21	24:06

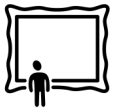


For Today

- Actor-critic: quick recap
- Sequence models
  - LMs, HMMs, KFs, RNNs
  - RNN examples

Recap:  Actor-critic approaches

- Estimate policy and value function together
- Actor: policy used to select actions
- Critic: value function used to criticize actor
- Error signal from the critic drives all learning
- An on-policy approach



Actor-critic approaches

- After each selected action, critic evaluates new state:
  - Have things gone better or worse than expected?
- The error signal is used to update actor and value function

Actor-critic approaches

Error signal is

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

$\delta_t > 0$: action was better than expected

$\delta_t < 0$: action was worse than expected

Actor-critic approaches

Error signal is

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

Value function is updated as

$$V(s_t) \leftarrow V(s_t) + \alpha\delta_t$$

Actor-critic approaches

Error signal is

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

Used to update parameters of policy.

If δ_t is positive (negative), action a_t should become more (less) probable in state s_t

For example, with

$$\pi_\theta(a \mid s) = \text{Softmax}\Big\{ f_\theta(s, a_1), \ldots, f_\theta(s, a_D) \Big\}$$

parameters θ adjusted so $f_\theta(s_t, a_t)$ increases (decreases)

Actor-critic approaches

Error signal is

$$\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$$

Used to update parameters of policy.

For example, with

$$\pi_\theta(a \mid s) = \text{Softmax}\Big\{ f_\theta(s, a_1), \ldots, f_\theta(s, a_D) \Big\}$$

parameters θ adjusted so $f_\theta(s_t, a_t)$ increases (decreases)

$$\theta \leftarrow \theta + \eta \delta_t \nabla_\theta \log \pi_\theta(a_t \mid s_t)$$

Next few classes

We'll talk about different approaches to sequence modeling (including language models), leading up to transformers.

Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

Language models

- A language model is a way of assigning a probability to any sequence of words (or string of text)

$$p(w_1, \ldots, w_n)$$

- By the basic rules of conditional probability we can factor this as

$$p(w_1, \ldots, w_n) = p(w_1)p(w_2 \mid w_1) \ldots p(w_n \mid w_1, \ldots, w_{n-1})$$

Language models

- A language model is a way of *generating* any sequence of words

$$
\begin{aligned}
&P(\text{“the whole forest had been anesthetized”}) = \\
&P(\text{“the”}) \times P(\text{“whole”} \mid \text{“the”}) \\
&\times P(\text{“forest”} \mid \text{“the whole”}) \\
&\times P(\text{“had”} \mid \text{“the whole forest”}) \\
&\times P(\text{“been”} \mid \text{“the whole forest had”}) \\
&\times P(\text{“anesthetized”} \mid \text{“the whole forest had been”})
\end{aligned}
$$

Remixing Noon

Text generated from Channel Skin by Jeff Noon

"The whole forest had been anesthetised, her temples wired, her senses stimulated, her eyes were not on Eva, not on Eva, not on Eva, not on anybody in that same realm, the land of dreams and nightmares."

Viability: 0.000000326%

https://revdancatt.com/2017/03/01/markov_noon

Text generation

- Words generated one-by-one
- A word is chosen by sampling from a probability distribution
- Condition on previously generated text, as if “real”
- Result is purely synthetic text

Modern language models

Suppose a computer program assigns a "score" to possible next words:

$$f(v; \underbrace{s(w_1, \ldots, w_n)}_{\text{state of word history}})$$

Modern language models

Suppose a computer program assigns a "score" to possible next words:

$$f(v; \underbrace{s(w_1, \ldots, w_n)}_{\text{state of word history}})$$

Can convert this to a language model by the "softmax" operation:

$$p(w \mid w_1, \ldots, w_n) = \frac{\exp(f(w; s(w_1, \ldots, w_n)))}{\sum_{v \in V} \exp(f(v; s(w_1, \ldots, w_n)))}$$

First things first

- In ChatGPT, the function $f(v; s(w_{1:n}))$ is learned on large amounts of text to predict the next word (unsupervised) using a type of deep neural network called a *transformer*
- We'll first discuss "classical" sequence models — HMMs, KFs, RNNs

This type of training is sometimes called "self-supervised" or "teacher-forced" learning

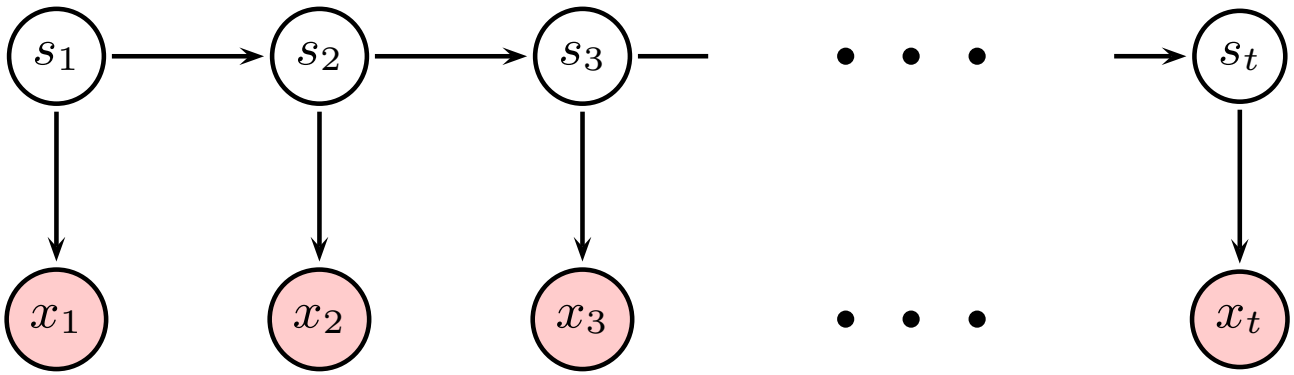
Hidden Markov Model

In an HMM, the next word is generated from a latent variable

- State is chosen stochastically (that is, probabilistically)
- Introduces dependence on earlier words that are further back than the previous time (non-Markov on observations)

Hidden Markov Model

The graphical model looks like this:



Here x_t is observable (word) at time t and s_t is unobserved state.

Hidden Markov Model

Probability of word sequence:

$$p(x_1, \ldots, x_n) = \sum_{s_1,\ldots,s_n} \prod_{t=1}^{n} p(s_t \mid s_{t-1}) p(x_t \mid s_t)$$

Hidden Markov Model

Probability of word sequence:

$$p(x_1, \ldots, x_n) = \sum_{s_1, \ldots, s_n} \prod_{t=1}^{n} p(s_t \mid s_{t-1}) p(x_t \mid s_t)$$

For topic models:

$$p(x_1, \ldots, x_n) = \int p(\theta \mid \alpha) \sum_{s_1, \ldots, s_n} \prod_{t=1}^{n} p(s_t \mid \theta) p(x_t \mid s_t) \, d\theta$$

Word order doesn't matter

Hidden Markov Model

Probability of word sequence:

$$p(x_1, \ldots, x_n) = \sum_{s_1, \ldots, s_n} \prod_{t=1}^{n} p(s_t \mid s_{t-1}) p(x_t \mid s_t)$$

Can be efficiently computed using the “forward-backward algorithm” which is a type of dynamic program

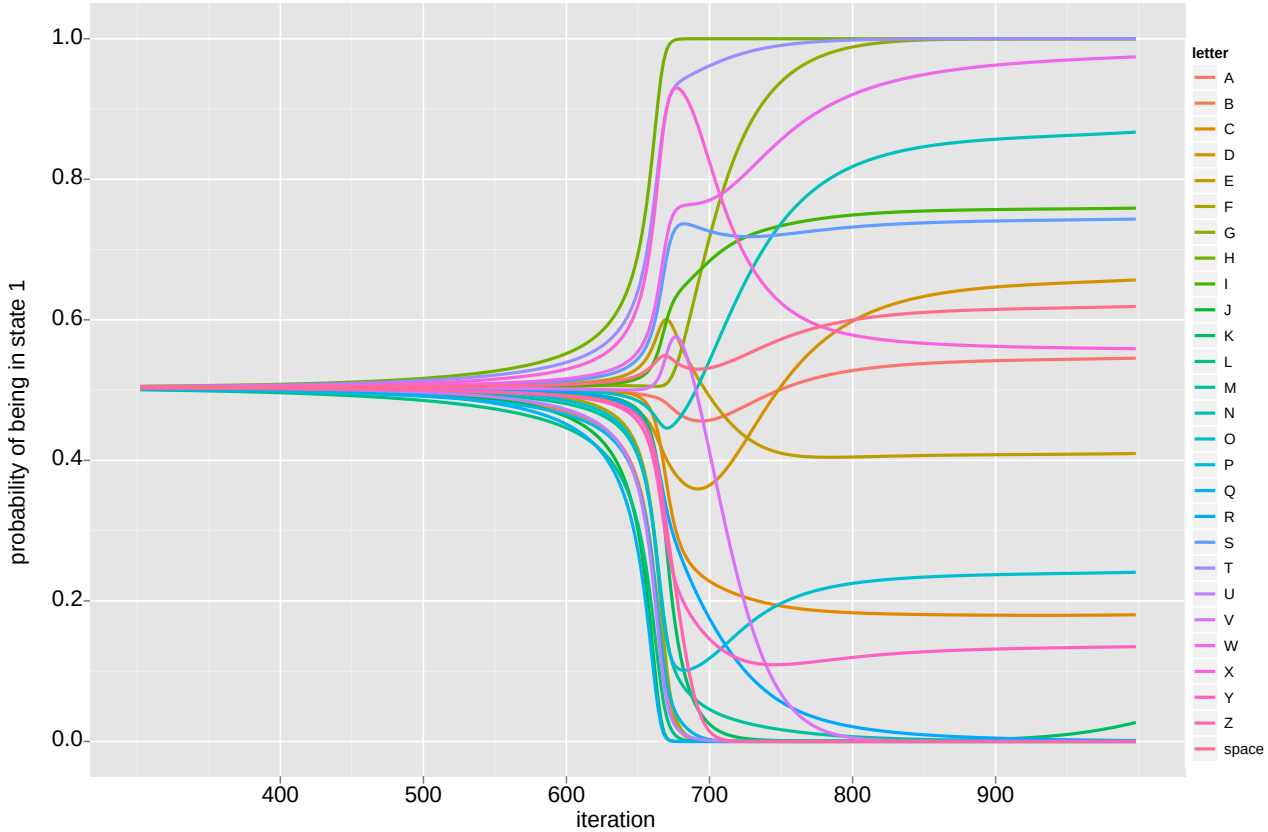
Hidden Markov Models: Estimation

The usual algorithms for estimating the parameters are iterative:

- Use the forward-backward algorithm to compute the probability that the outputs are generated from each state
- Treat the data as labeled, with these weights, and compute the relative frequencies
- Repeat
- Maximum likelihood (or MAP) estimation

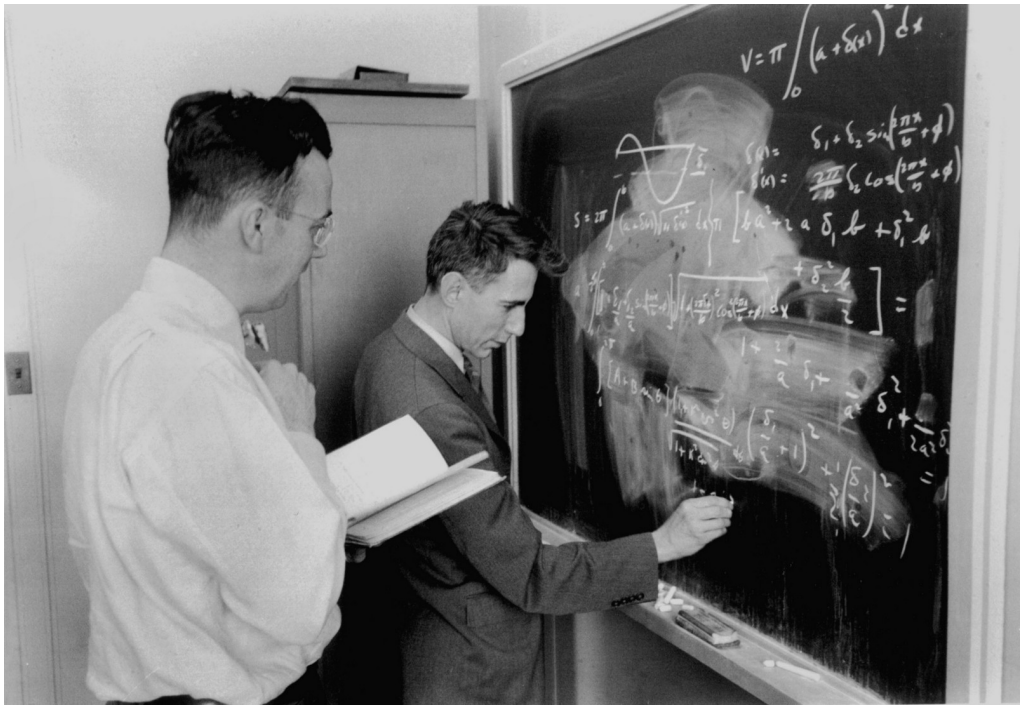
This is called the EM algorithm; “Expectation-Maximization”

Hidden Markov Models: Example



HMM has two states, emits one of the twenty-six letters A-Z. Trained on 20,000 characters of English text, initial parameters close to uniform. Curves show the posterior probability $\mathbb{P}(\text{state} = 1 \mid \text{letter}, \mathcal{D})$ of the probability each letter is generated from state one. “phase transition” at around 650 iterations.

Claude Shannon



Model underlying Kalman filters

A Gaussian HMM:

$$S_t \mid S_{t-1} \sim N(AS_{t-1}, \Gamma)$$

$$X_t \mid S_t \sim N(BS_t, \Sigma)$$

where A and B are matrices for the means; Γ and Σ for the covariances. Everything is linear.

- State is distributed (real vector)
- State evolves stochastically
- Schur complements and forward-backward used to compute conditional probabilities; similar to Gaussian processes

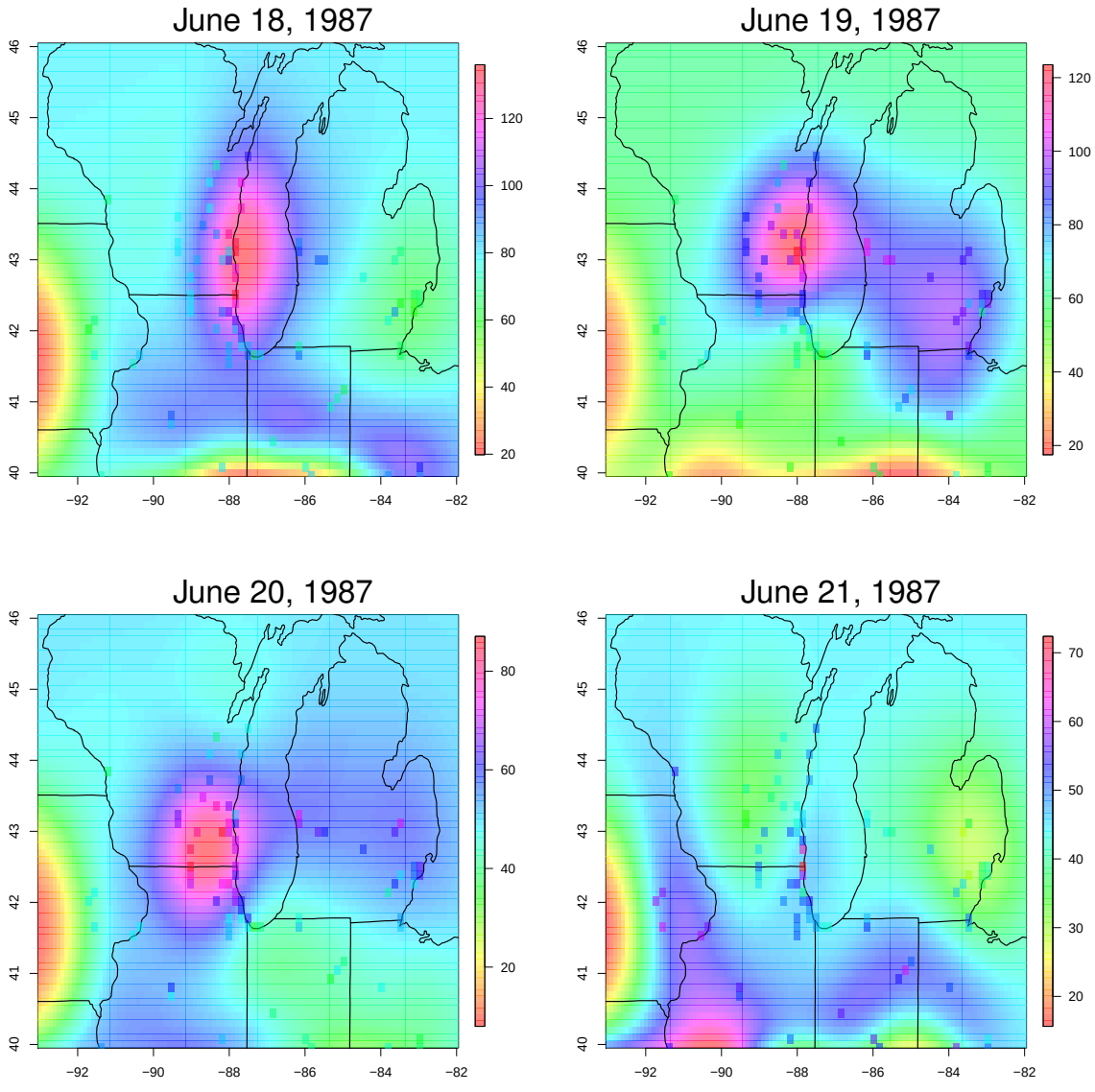
Example: Ozone

Data are daily maximum 8-hour ozone concentrations (in parts-per-billion) at 153 sites in the US midwest near Lake Michigan, for 89 days during the summer of 1987.

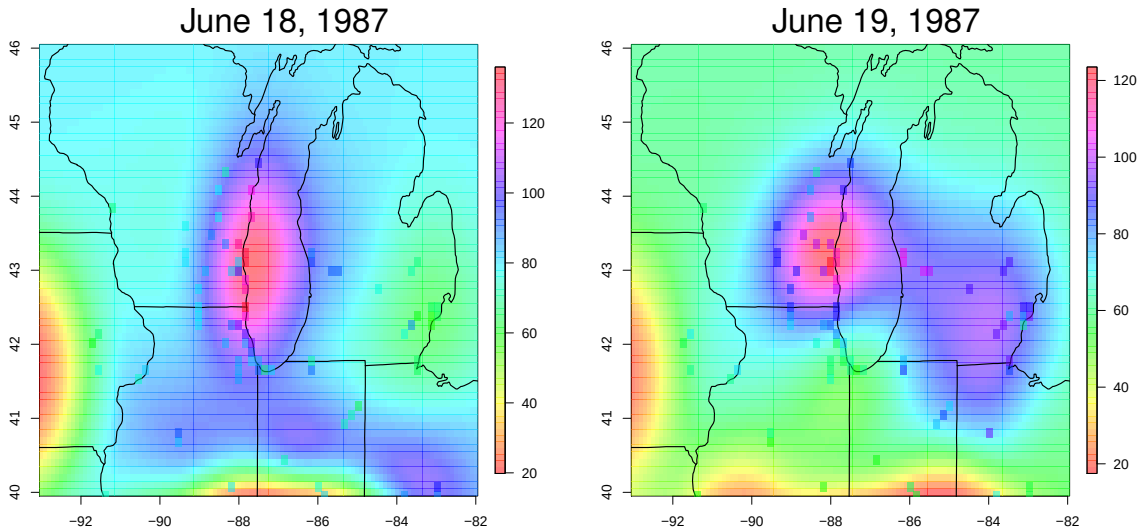
- `lon.lat`: longitudes and latitudes of the 153 stations;
- `y`: measurements at each stations on each day;
- `station.id` and `dates`.

Can use Kalman filtering/smoothing to infer ozone concentrations of the area

Kalman filtering: Example

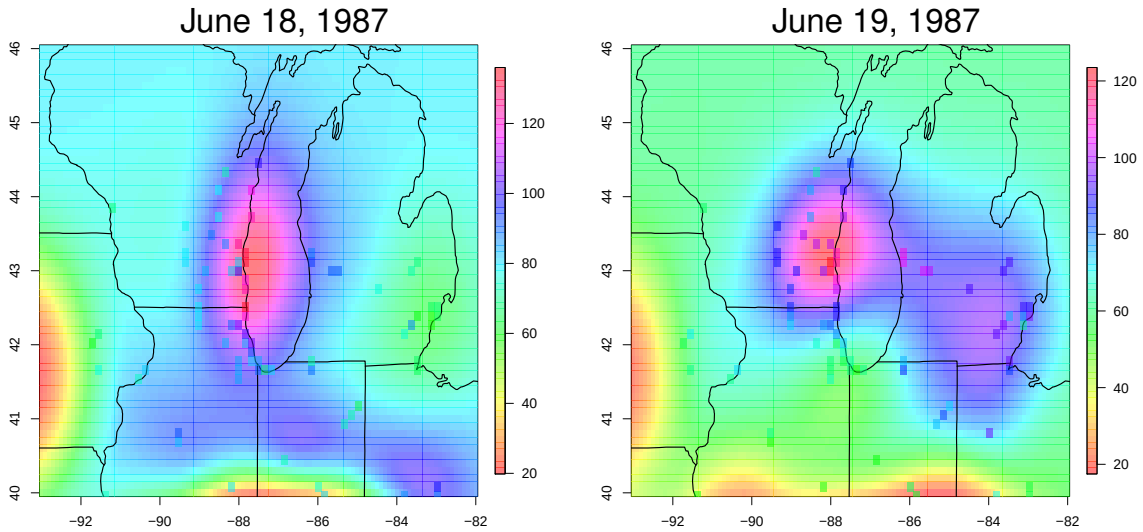


Kalman filtering: Example



- In this example, $A = I$ (just add noise)
- $B_j =$ "average over locations near j".

Kalman filtering: Example



- The Kalman filter is a generative model
- You can “synthesize” ozone levels on a given day.
- Could use kernel smoothing (a discriminative model) or a Gaussian process

Kalman filtering: Discrete data

We can also work with state space models for discrete data, like documents.

$$S_t \mid S_{t-1} \sim N(AS_{t-1}, \Gamma)$$

$$W_t \mid S_t \sim \text{Softmax}(BS_t)$$

where

$$\text{Softmax}(BS_t)_j = \frac{\exp\left(B_j^T S_t\right)}{\sum_k \exp\left(B_k^T S_t\right)}$$

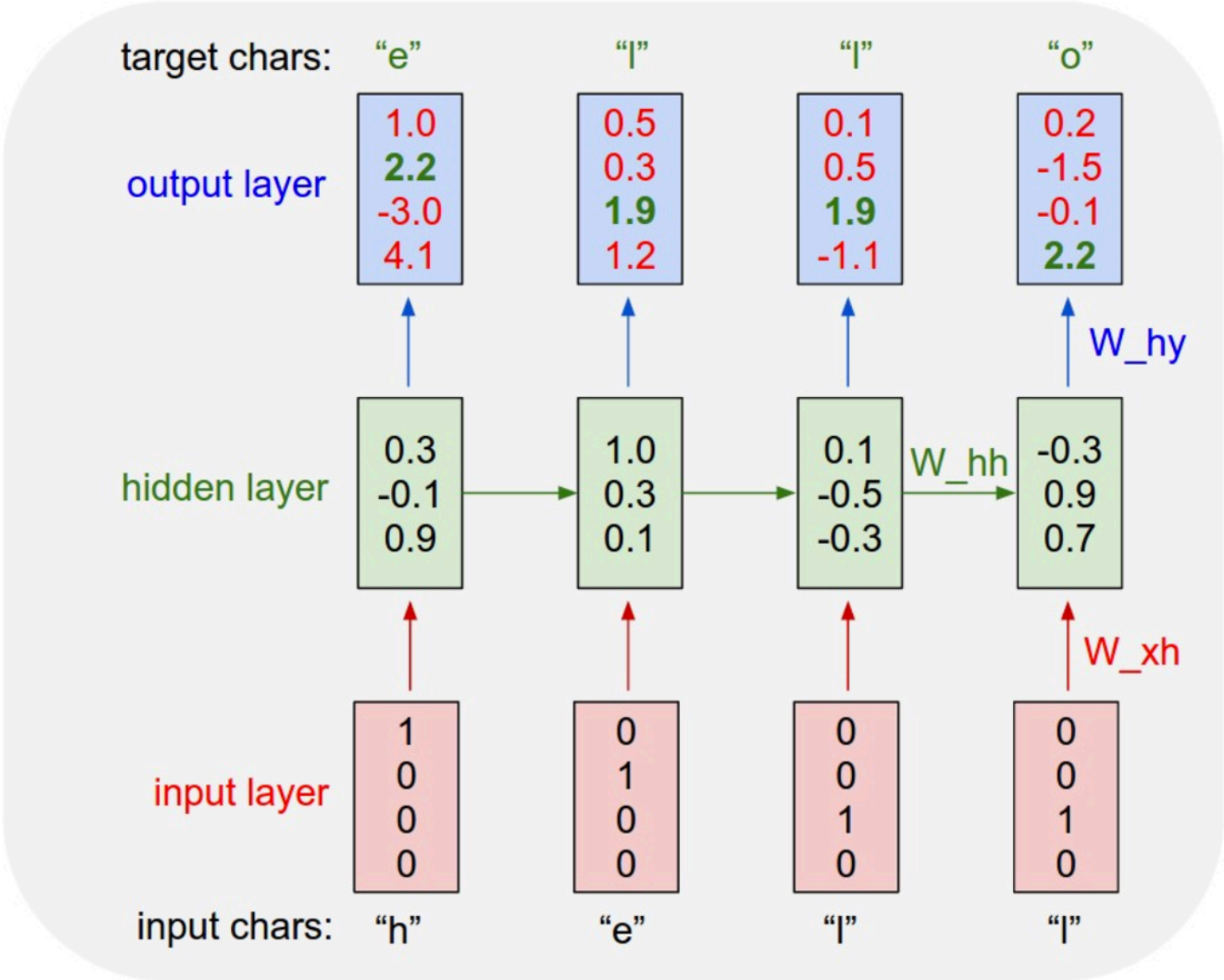
This is what's done with dynamic topic models, which use a Kalman filter as a variational approximation.
https://en.wikipedia.org/wiki/Dynamic_topic_model

Next: Recurrent neural networks

- Can be thought of as a type of language model
- Similar to hidden Markov models and Kalman filters, but the hidden layer is not stochastic
- The state is distributed (a vector), as in Kalman filters, not categorical as for HMMs

Classical models like Kalman filters also have distributed state, but use linear models

RNNs http://karpathy.github.io/2015/05/21/rnn-effectiveness/



RNNs

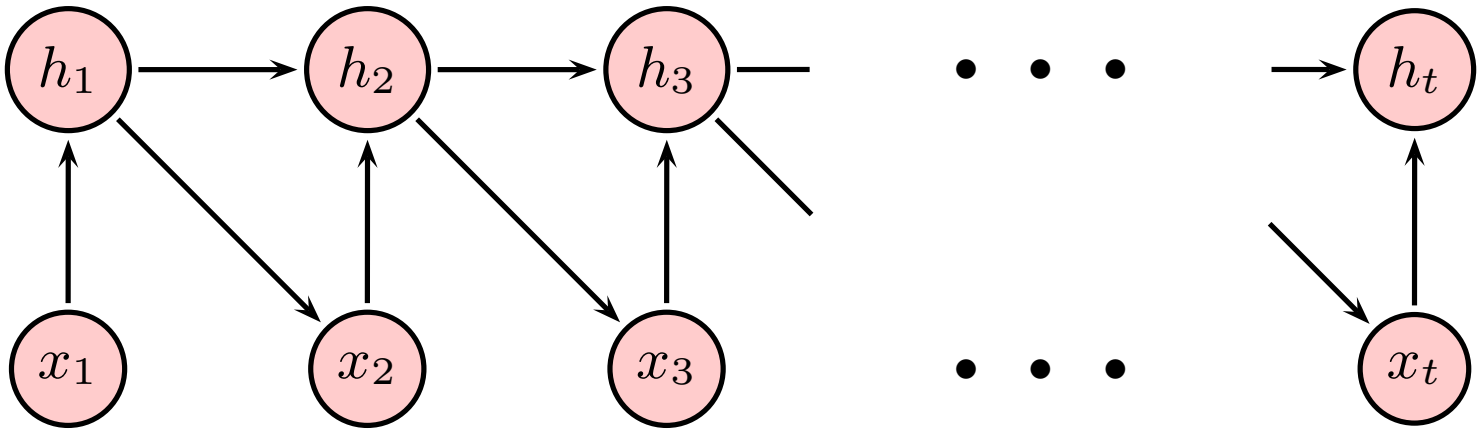
This means

$$
\begin{aligned}
h_t &= \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t\right) \\
y_t &= W_{hy} h_t \\
x_{t+1} \mid y_t &\sim \text{Multinomial}(\pi(y_t))
\end{aligned}
$$

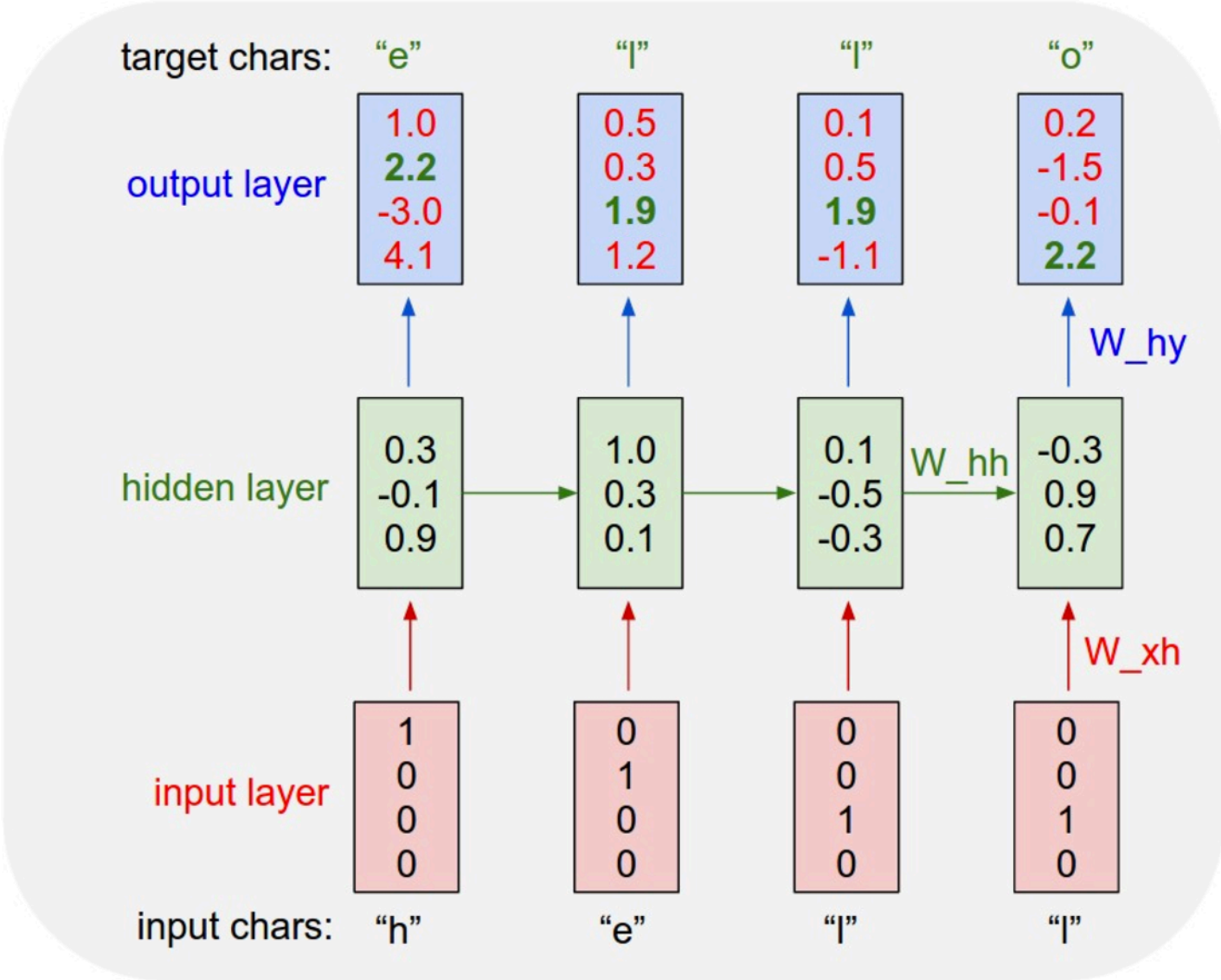
where $\pi(\cdot)$ is the soft-max function.

In this illustration, x_t is the "1-hot" representation of a character, $W_{xh} \in \mathbb{R}^{3\times 4}$, $W_{hh} \in \mathbb{R}^{3\times 3}$ and $W_{hy} \in \mathbb{R}^{4\times 3}$.

RNN graphical structure



RNN architecture



RNN examples

http://karpathy.github.io/2015/05/21/rnn-effectiveness/

LSTMs

A variant called “Long Short-Term Memory” RNNs has a special hidden later that “includes” or “forgets” information from the past.

Intuition: In language modeling, may be useful to remember/forget gender or number of subject so that personal pronouns (“he” vs. ”she” vs. ”they”) can be used appropriately.

Useful for things like matching parentheses, etc.

A simpler alternative to the LSTM circuit is called the *Gated Recurrent Unit* (GRU)

“Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation”, Cho et al. 2014
https://arxiv.org/pdf/1406.1078v3.pdf

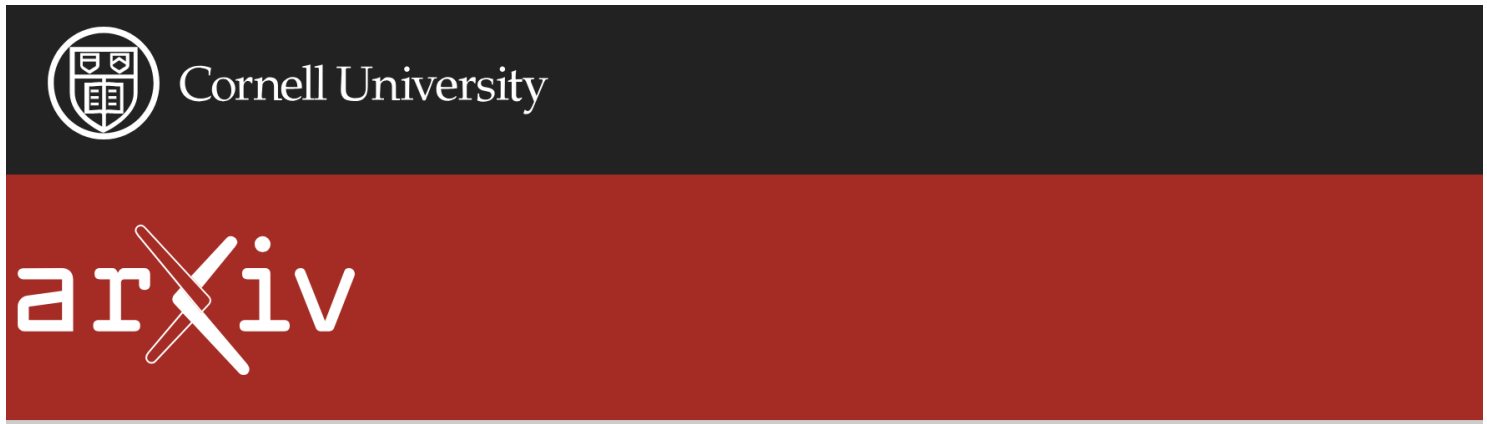
Synthetic Shakespeare

```
PANDARUS:
Alas, I think he shall be come approached and the day
When little srain would be attain'd into being never fed,
And who is but a chain and subjects of his death,
I should not sleep.

Second Senator:
They are away this miseries, produced upon my soul,
Breaking and strongly should be buried, when I perish
The earth and thoughts of many states.

DUKE VINCENTIO:
Well, your wit is in the care of side and that.

Second Lord:
They would be ruled after this chamber, and
my fair nues begun out of the fact, to be conveyed,
Whose noble souls I'll have the heart of the wars.

Clown:
Come, sir, I will make did behold your worship.

VIOLA:
I'll drink it.
```

Topics in scientific texts

- We combined topic modeling for text and recurrent neural networks for equations
- The topics of a scientific article were inferred from the words
- Conditioned on the topics, the mathematical equations were generated using an RNN

Topics in scientific texts



arXiv is a free distribution service and an open-access archive for nearly 2.4 million scholarly articles in the fields of physics, mathematics, computer science, quantitative biology, quantitative finance, statistics, electrical engineering and systems science, and economics. Materials on this site are not peer-reviewed by arXiv.

Subject search and browse:

Physics | Search | Form Interface | Catchup

Topics in scientific texts

Quantum physics	spin energy field electron magnetic state states hamiltonian
Particle physics	higgs neutrino coupling decay scale masses mixing quark
Astrophysics	mass gas star stellar galaxies disk halo radius luminosity
Relativity	black metric hole schwarzschild gravity holes einstein
Number theory	prime integer numbers conjecture integers degree modulo
Graph theory	graph vertex vertices edges node edge number set tree
Linear algebra	matrix matrices vector basis vectors diagonal rank linear
Optimization	problem optimization algorithm function solution gradient
Probability	random probability distribution process measure time
Machine learning	layer word image feature sentence model cnn lstm training

Topics in scientific texts

Latent topic vector θ included as additional information in RNN state transitions:

$$
\begin{aligned}
h_t &= \tanh\left(W_{hh} h_{t-1} + W_{xh} x_t + W_{x\theta} \theta_d\right) \\
y_t &= W_{hy} h_t \\
x_{t+1} \mid y_t &\sim \text{Multinomial}(\pi(y_t))
\end{aligned}
$$

where $\pi(\cdot)$ is the soft-max function.

---

https://aaai.org/ojs/index.php/AAAI/article/view/4728

Topic modeling for equations

Topic	Generated Equations
Quantum physics	• $E = \hbar \frac{\partial^2 S}{\partial t^2} (\frac{\partial \varphi}{\partial c}) - \frac{k}{\hbar^2} \frac{\partial B}{\partial t} (t + \partial_t \delta).$ • $\Psi_{\text{pr}} = \sum_{\mathbf{l}} (\psi_{\mathbf{r}+\uparrow} - \psi^{\dagger}_{\mathbf{r}\downarrow}) + \sum_{\mathbf{r}'} (\psi^{\dagger}_{\mathbf{r}\downarrow,\uparrow} - \psi_{\mathbf{r}\downarrow} \sigma^{\dagger}).$
Particle physics	• $\mathcal{H} = \frac{2}{4} (\partial_\mu \phi)^2 + 2m\phi_\nu(\phi) + \frac{1}{2} m^2(\phi)(1-\phi^2)^2.$ • $m_{\text{eff}}(M) = 1.4 \cdot 10^{-13}\,\text{GeV}.$
Relativity	• $\mathcal{M} = \frac{2}{2} g^{\mu\nu} (f_{\mu\nu,\mu} - g_{\mu\nu,\nu} + g_{\nu\nu,b} f_{\mu,\nu}) + \frac{1}{2} g^{\mu\nu}.$ • $T_{\mu\nu} = \int_0^\infty ds_{\mu\nu} ds^2 + a_\mu^2 dr^2 + r^2 d\Omega^2.$

LSTM with two hidden layers, 512 neurons in each layer, trained on 40,000 scientific articles, using LaTeX source. The generated LaTeX compiles about 80% of the time; otherwise needs minor tweaks.

Topic modeling for equations

Number theory	• $(2^{k})^{k} + (1^{n} + 1)(1 + p^{k}) = 1.$
Linear algebra	• $\operatorname{tr}(E_{\varepsilon} X^{*}) = U^{\top}(\operatorname{tr}(V_{\varepsilon} X)).$ • $\phi_{h}(\theta, y) = \left\{ X \in \operatorname{Span}\left(P_{c}(\mathbf{T}[x, \mathtt{x}]) \right) \right].$
Optimization	• $\min_{p} p(x) \text{ subject to } \|p^{x} - y\|_{2} \leq m_{p}.$ • $w^{+} = w_{t} + g_{t} \|u_{t} - \nabla \mathbf{u}^{*}\|_{2}^{2}.$
Probability	• $\mathbb{P}(r_{\tau} < t) = \mathbb{E}_{\tau_{\text{wist}}}(N_{\tau}).$ • $T^{*}(t) = \lim_{t \to \infty} \mathbb{E}[\|N(t) + \mathbb{E}[\varphi_{t}(x)]^{*}\|]$

LSTM with two hidden layers, 512 neurons in each layer, trained on 40,000 scientific articles, using LaTeX source. The generated LaTeX compiles about 80% of the time; otherwise needs minor tweaks.

Topic modeling for equations

Context words	Inferred Topics	Generated Equations
star gravity einstein mass galaxies	58% Astrophysics 36% Relativity 3% Quantum physics	• $\left(\frac{m_b}{M_\odot}\right)^{b_\nu} \ll g\sqrt{\frac{\tilde{\Phi}_0}{\eta_{\text{eff}}}}$. • $G(r) = \int_{r_0}^{r_0} dr\sqrt{\log(r)(\overline{r} + r_0(w))}$.
data training likelihood model gradient	62% Machine learning 21% Statistics 15% Optimization	• $L = -\frac{1}{N}\sum_{i=1}^{N} \mathcal{R}_{R_i}(\mathbf{r}_i) + V^r(\mathbf{r}_i)$. • $\operatorname{argmax}_{\mathcal{U}} \mathbb{E}_{W\sim\Psi} \log\exp\left[\Lambda(\widetilde{W}) - H\right]$.

LSTM with two hidden layers, 512 neurons in each layer, trained on 40,000 scientific articles, using LaTeX source. The generated LaTeX compiles about 80% of the time; otherwise needs minor tweaks.

Fake νs

Which is fake?

$$L = \int_0^{e^{T-1}} \psi \sin^2 \theta d\lambda \sin^2 \theta d\theta^2 - C^{2/\sigma}$$

$$L = \int_0^L dz \|x_z\| = \int_0^L dz \sqrt{x_z \cdot x_z} \equiv \int_0^L dz \sqrt{g}$$

Fake νs

$$L = \int_0^{e^{T-1}} \psi \sin^2\theta d\lambda \sin^2\theta d\theta^2 - C^{2/\sigma}$$

$$L = \int_0^L dz \|x_z\| = \int_0^L dz \sqrt{x_z \cdot x_z} \equiv \int_0^L dz \sqrt{g}$$

Fake νs

Which is fake?

$$(xy^{-1})_{N(y)} = s^{\epsilon}(sxy^{-1})_{N(y)} = s^{\epsilon}(sx)_{\emptyset}y_{\emptyset}^{-1}$$

$$(xy^2 - x^2)(y(x) - Ty(x)) \leq \int_0^T e^{-x't(w)}dt$$

Fake νs

$$(xy^{-1})_{N(y)} = s^{\epsilon}(sxy^{-1})_{N(y)} = s^{\epsilon}(sx)_{\emptyset}y_{\emptyset}^{-1}$$

$$(xy^2 - x^2)(y(x) - Ty(x)) \leq \int_0^T e^{-x't(w)}dt$$

Fake νs with topics

"optimization"

$$X_L = -\frac{1}{\sum_i y_i}\pi(y_i, x_i, x > 0) \text{ subject to } x_i^T X^H x_i \leq \epsilon^2$$

$$\text{minimize } -\sum_{i=1}^{K}\sum_{c=1}^{Z} a_{mij}\|g_i\|_2^2$$

Fake νs with topics

“physics”

$$\hat{b}_{\mathrm{prc}} = \frac{\hbar^2}{R} - \mu_{\mathrm{sit}} F_{d}$$

$$\frac{\partial \delta C}{\partial r} R e_{a} a_{\mu} - \int_{V_{bor,quet}} \left\langle K^{\alpha\beta} \right\rangle_{\alpha}^{\beta} = \mathcal{H} E r + \beta_{\alpha\mu\nu}^{\top} R_{\mathrm{conseds}}^{\mu\lambda} \partial_{\mu}$$

$$\rho_{deg} \propto 11 eV \left[L(003B)/20 \pm 10 i f s . 1.8 V (10^{-2} e r g s^{-1})/5; 1.13 n e V \right]$$

rnn-demo.ipynb: numpy version

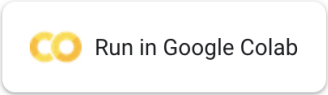
We'll train an "np-complete" RNN on the complete works of William Shakespeare, downloaded from Project Gutenberg (specifically, this link).

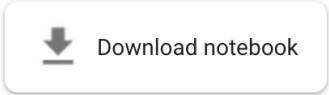
This uses a simple, direct implementation of backpropagation for RNNs, paralleling what we did for simple feedforward networks.

In [ ]:
```
def lossFun(inputs, targets, hprev):
  """
  inputs,targets are both list of integers.
  hprev is Hx1 array of initial hidden state
  returns the loss, gradients on model parameters, and last hidden state
  """
  xs, hs, ys, ps = {}, {}, {}, {}
  hs[-1] = np.copy(hprev)
  loss = 0

  # forward pass
  for t in range(len(inputs)):
    xs[t] = np.zeros((vocab_size,1)) # encode in 1-of-k representation
    xs[t][inputs[t]] = 1
    hs[t] = np.tanh(np.dot(Wxh, xs[t]) + np.dot(Whh, hs[t-1]) + bh) # hidden state
    ys[t] = np.dot(Why, hs[t]) + by # unnormalized log probabilities for next chars
    ps[t] = np.exp(ys[t]) / np.sum(np.exp(ys[t])) # probabilities for next chars
    loss += -np.log(ps[t][targets[t],0]) # softmax (cross-entropy loss)

  # backward pass: compute gradients going backwards
  dWxh, dWhh, dWhy = np.zeros_like(Wxh), np.zeros_like(Whh), np.zeros_like(Why)
  dbh, dby = np.zeros_like(bh), np.zeros_like(by)
  dhnext = np.zeros_like(hs[0])
  for t in reversed(range(len(inputs))):
    dy = np.copy(ps[t])
    dy[targets[t]] -= 1 # backprop into y. see http://cs231n.github.io/neural-networks-case-stu
    dWhy += np.dot(dy, hs[t].T)
    dby += dy
    dh = np.dot(Why.T, dy) + dhnext # backprop into h
    dhraw = (1 - hs[t] * hs[t]) * dh # backprop through tanh nonlinearity
    dbh += dhraw
    dWxh += np.dot(dhraw, xs[t].T)
    dWhh += np.dot(dhraw, hs[t-1].T)
    dhnext = np.dot(Whh.T, dhraw)

  for dparam in [dWxh, dWhh, dWhy, dbh, dby]:
    np.clip(dparam, -5, 5, out=dparam) # clip to mitigate exploding gradients

  return loss, dWxh, dWhh, dWhy, dbh, dby, hs[len(inputs)-1]
```

Tensorflow implementation

Text generation with an RNN





This tutorial demonstrates how to generate text using a character-based RNN. You will work with a dataset of Shakespeare's writing from Andrej Karpathy's The Unreasonable Effectiveness of Recurrent Neural Networks. Given a sequence of characters from this data ("Shakespear"), train a model to predict the next character in the sequence ("e"). Longer sequences of text can be generated by calling the model repeatedly.

Note: Enable GPU acceleration to execute this notebook faster. In Colab: *Runtime > Change runtime type > Hardware accelerator > GPU*. If running locally make sure TensorFlow version >= 1.11.

This tutorial includes runnable code implemented using tf.keras and eager execution. The following is sample output when the model in this tutorial trained for 30 epochs, and started with the string "Q":

```
QUEENE:
I had thought thou hadst a Roman; for the oracle,
Thus by All bids the man against the word,
Which are so weak of care, by old care done;
Your children were in your holy love,
And the precipitation through the bleeding throne.

BISHOP OF ELY:
Marry, and will, my lord, to weep in such a one were prettiest;
Yet now I was adopted heir
Of the world's lamentable day,
To watch the next way with his father with his face?

ESCALUS:
The cause why then we are all resolved more sons.
```

Summary

- HMMs and Kalman Filters are classical models for sequences
- HMMs have discrete states, KFs have distributed states and Gaussian transitions/emissions
- Recurrent neural networks are also used for sequential data, like language
- As usual, linear mappings followed by activation functions.
- RNNs are unsupervised, generative models, able to generate realistic looking sequences when sufficiently well trained.
- LSTMs and GRUs model longer range dependencies (next time)