S&DS 365 / 665

Intermediate Machine Learning

Sequence Models and Recurrent Neural Networks

April 18

Yale

Home stretch!

- Final exam: Saturday May 7 at 2pm in SPL 59
- Assignment 4 (last!) is out
  - Graph neural nets
  - Policy gradient
  - Deep Q-Learning
  - Recurrent neural networks
- Quiz 4 (last!) this Wednesday
  - RL concepts

For Today

Sequence models

- Recurrent neural networks
- RNN examples
- A look at the code

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

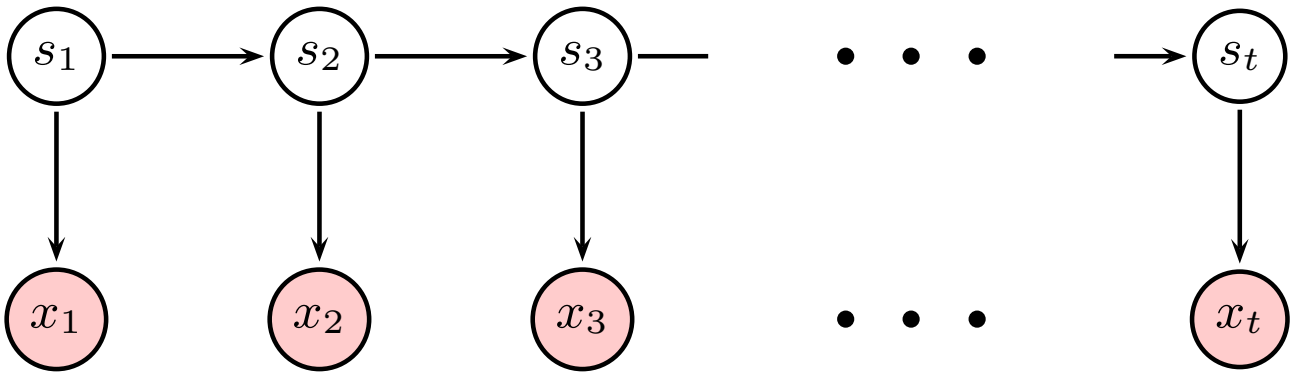
Hidden Markov Model

In an HMM, current word generated from a latent variable.

- State is chosen stochastically (that is, probabilistically, or randomly)
- As a consequence, the dependence on earlier words extends much further back in time.

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

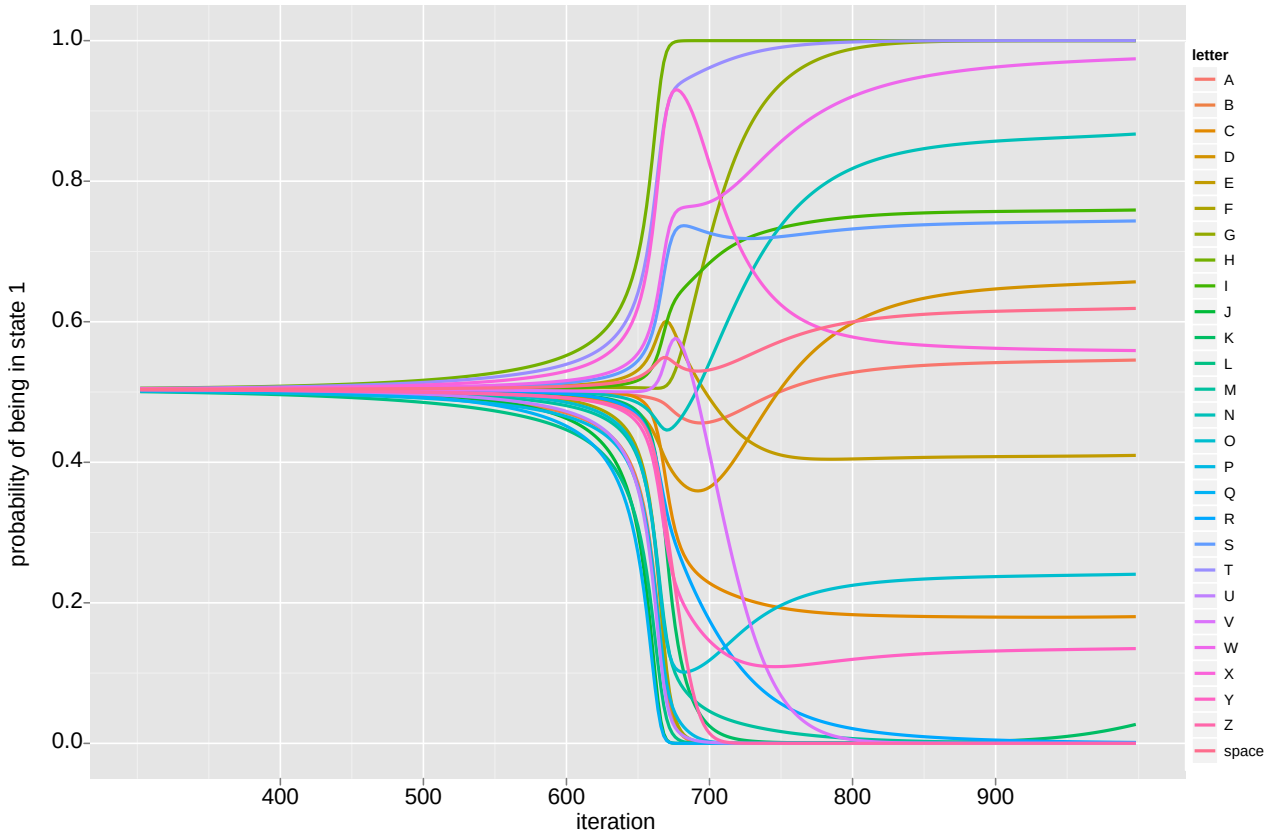
Hidden Markov Models: Estimation

The usual algorithns 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

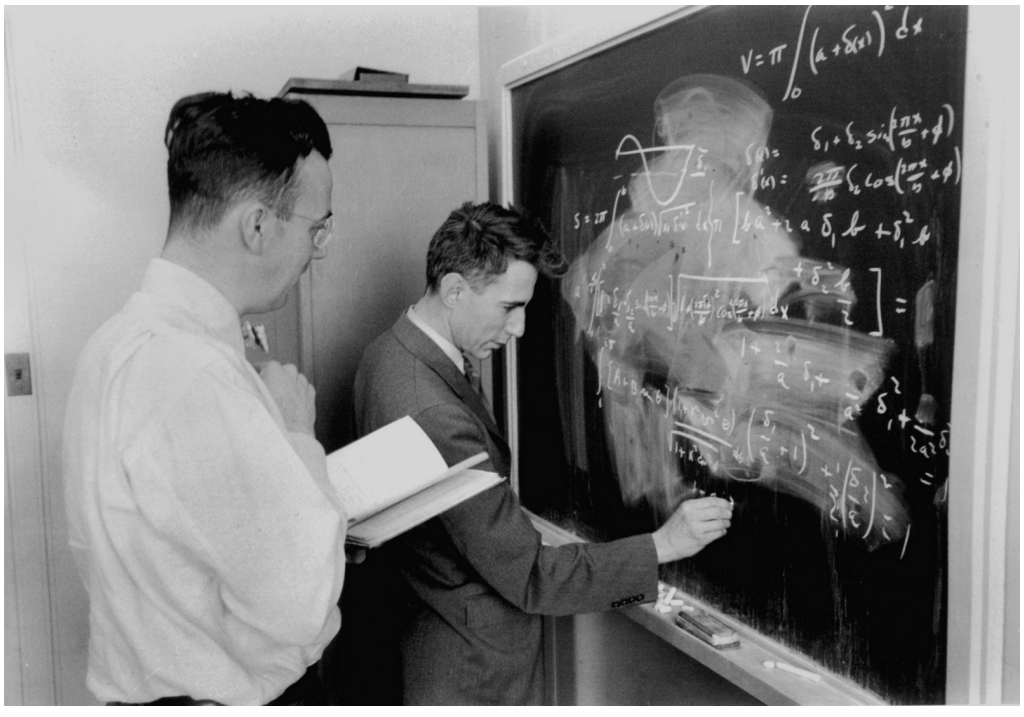
This is called the EM algorithm

Hidden Markov Models: Example

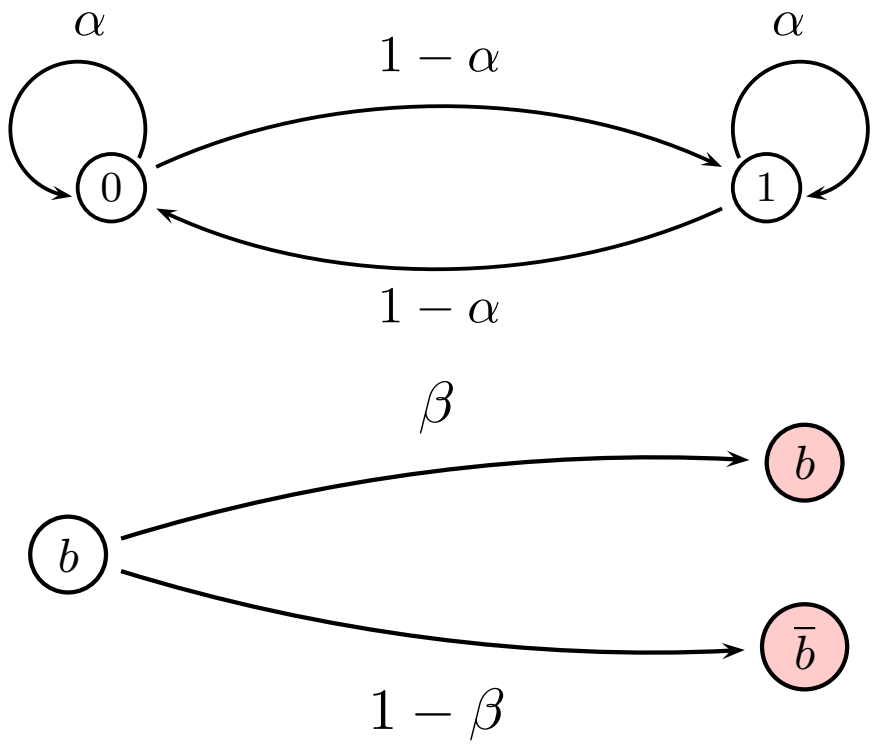


HMM has two states, emits one of the twenty-six letters A-Z. Trained on 20,000 characters of English text, initial parameters close to uniform. Curves show the posterior probability $\mathbb{P}(\text{state} = 1 \mid \text{letter}, \mathcal{D})$ of the probability each letter is generated from state one.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

Claude Shannon



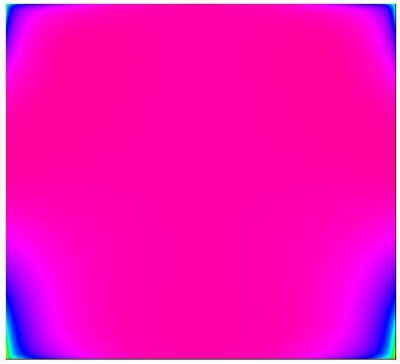
Hidden Markov Models: Instability



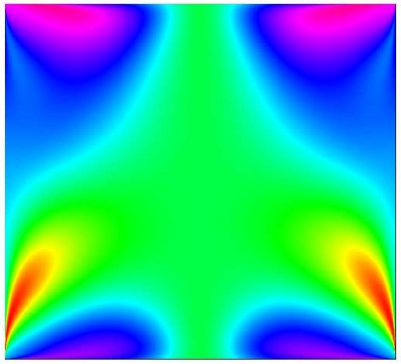
Two parameter HMM, emitting bits 0/1.

Hidden Markov Models: Instability

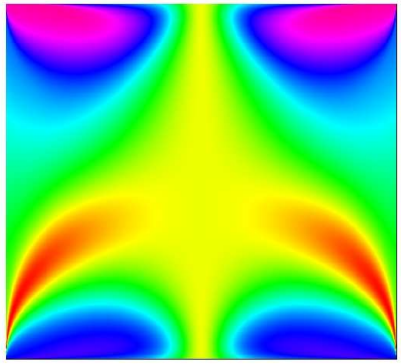
iteration 1



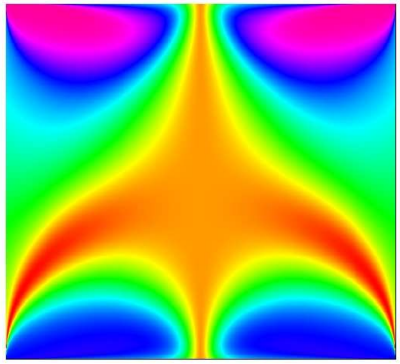
iteration 2



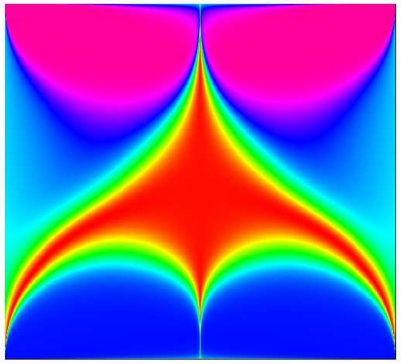
iteration 3



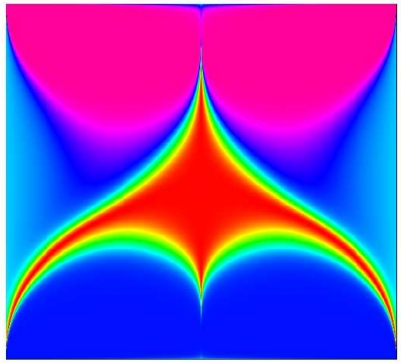
iteration 4



iteration 10



iteration 15



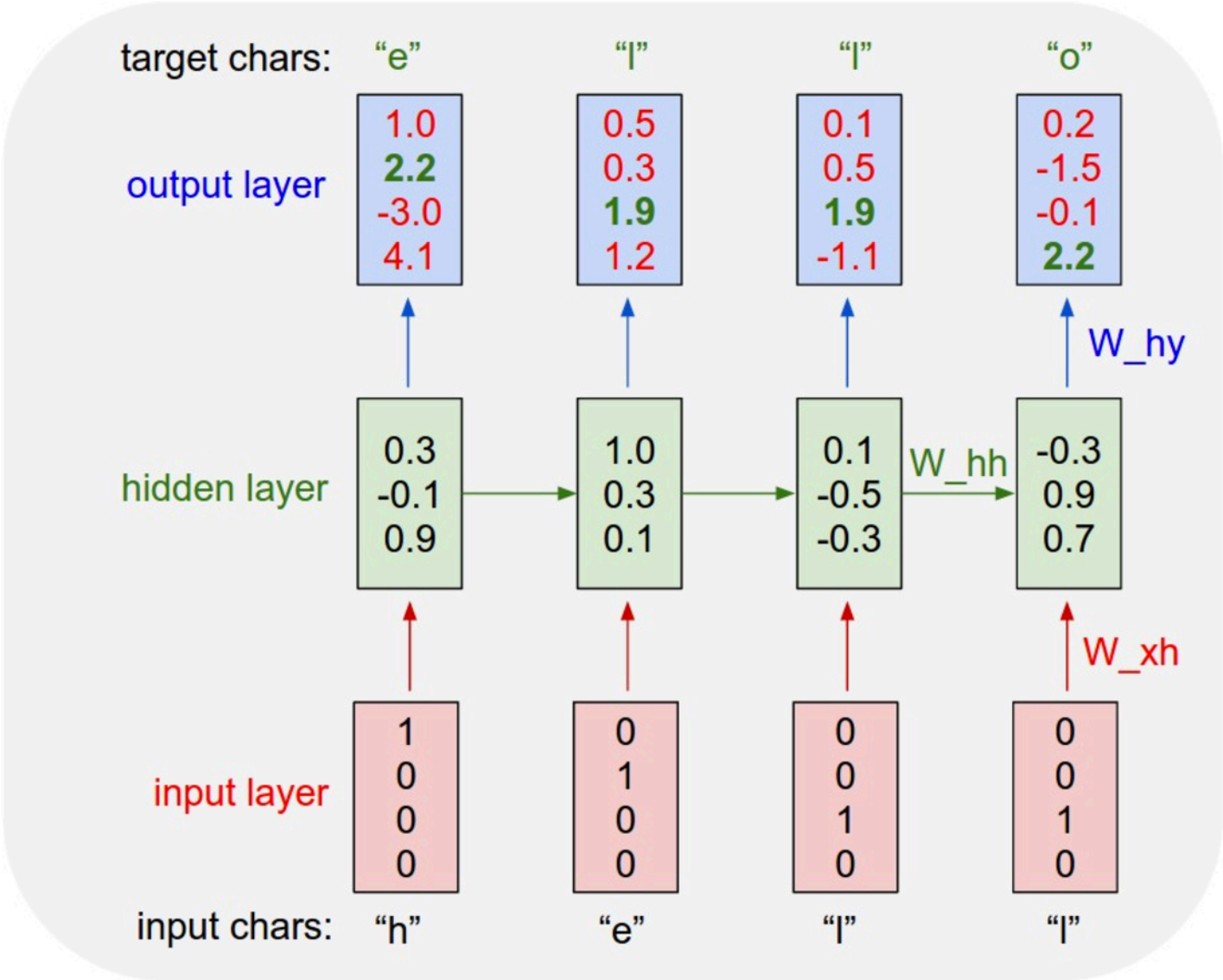
Hidden Markov Models: Instability

- Color at point (α_0, β_0) indicates probability of training data assigned by the model, when initialized at (α_0, β_0)
- Takeaway point: training can be very sensitive to initialization

Recurrent neural networks

- Can be thought of as a type of language model
- Similar to hidden Markov models, but the hidden layer is not stochastic
- The state is distributed (like embeddings), not categorical as for HMMs
- We'll describe this using characters rather than words — could do either

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



RNNs

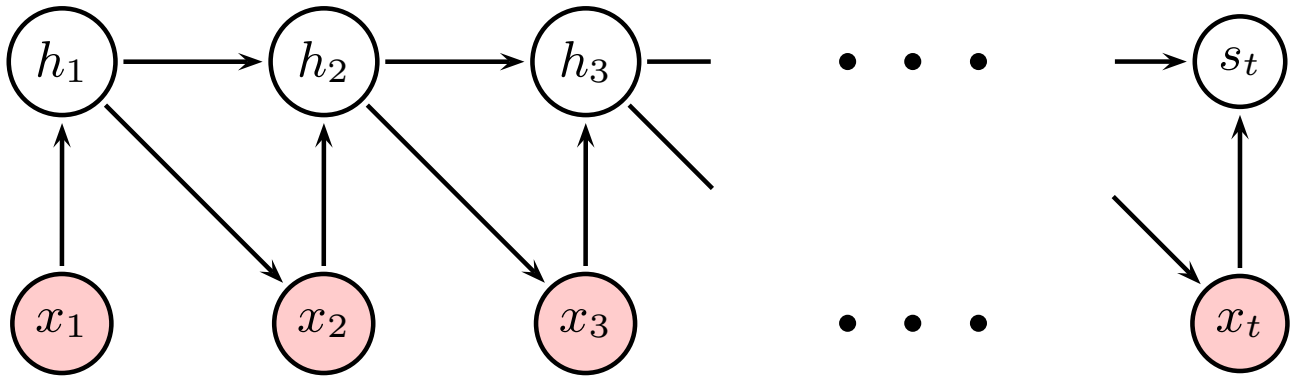
This means

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

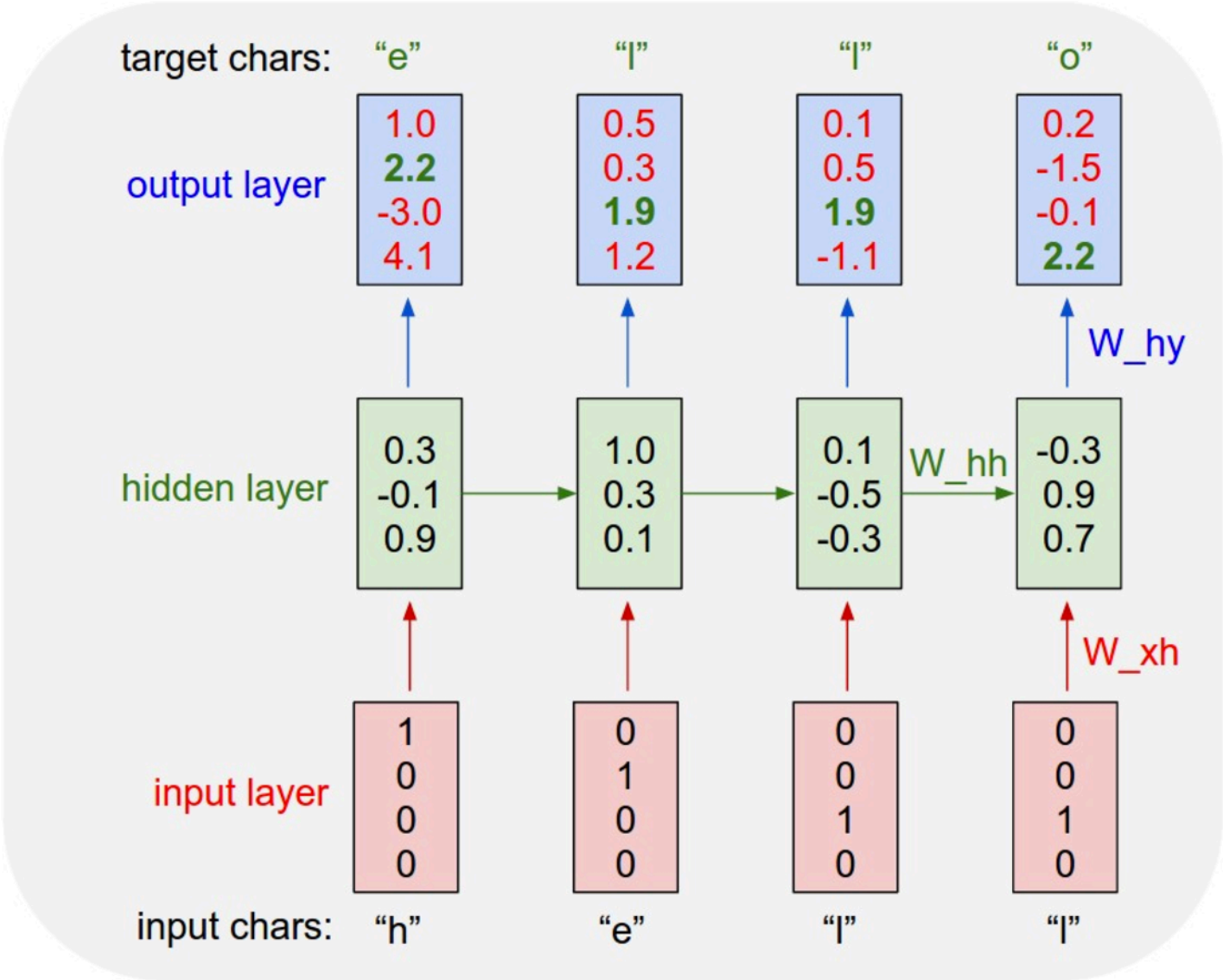
where $\pi(\cdot)$ is the soft-max function.

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



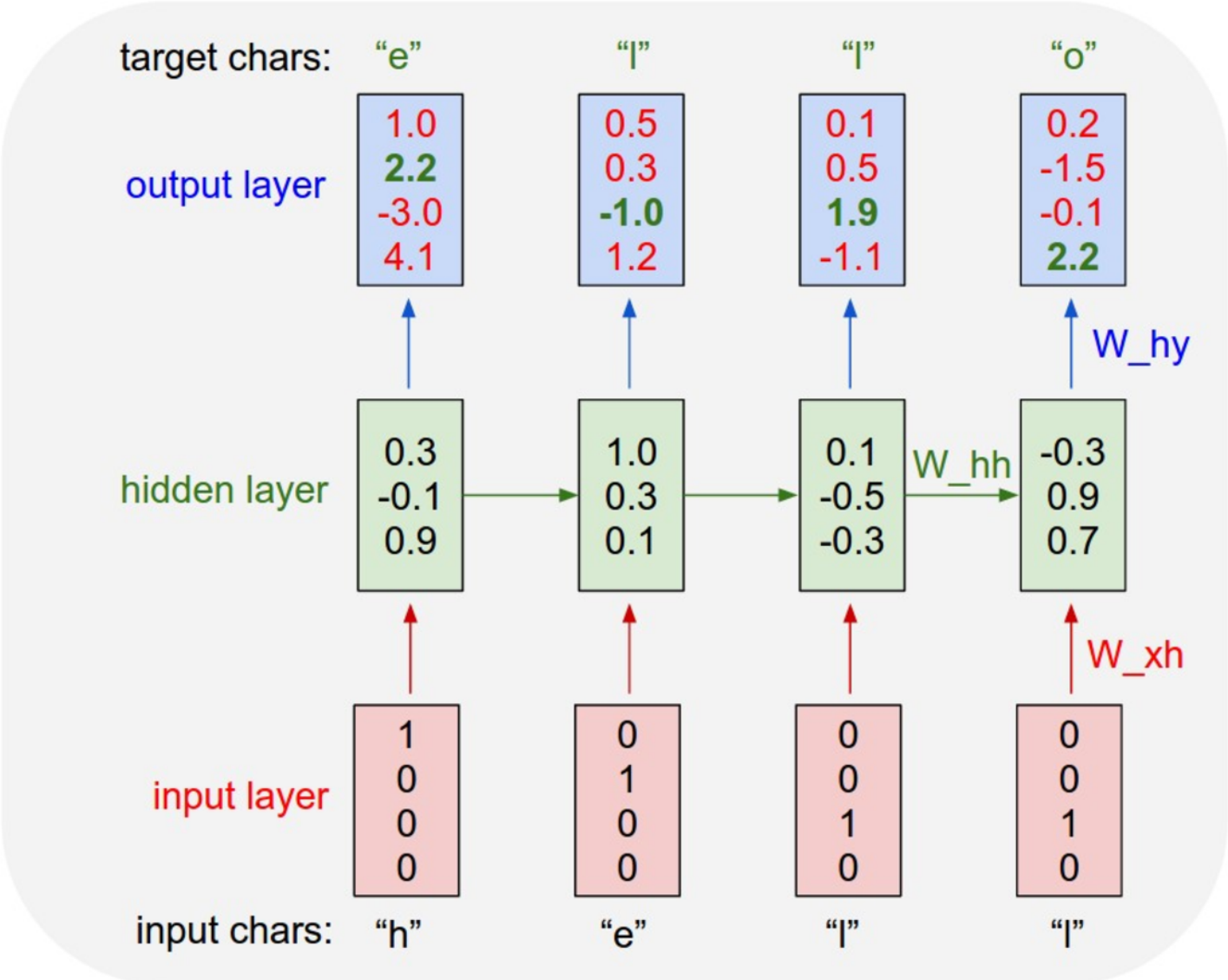
RNN Python Code

```
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

One hidden layer:

```
rnn = RNN()
y = rnn.step(x) # x is an input vector, y is the RNN's output vector
```

One hidden layer



RNN Python Code

```
class RNN:
  # ...
  def step(self, x):
    # update the hidden state
    self.h = np.tanh(np.dot(self.W_hh, self.h) + np.dot(self.W_xh, x))
    # compute the output vector
    y = np.dot(self.W_hy, self.h)
    return y
```

Two hidden layers:

```
y1 = rnn1.step(x)
y = rnn2.step(y1)
```

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

Hallucinated Wikipedia

```
Naturalism and decision for the majority of Arab countries' capitalide was grounded
by the Irish language by [[John Clair]], [[An Imperial Japanese Revolt]], associated
with Guangzham's sovereignty. His generals were the powerful ruler of the Portugal
in the [[Protestant Immineners]], which could be said to be directly in Cantonese
Communication, which followed a ceremony and set inspired prison, training. The
emperor travelled back to [[Antioch, Perth, October 25|21]] to note, the Kingdom
of Costa Rica, unsuccessful fashioned the [[Thrales]], [[Cynth's Dajoard]], known
in western [[Scotland]], near Italy to the conquest of India with the conflict.
Copyright was the succession of independence in the slop of Syrian influence that
was a famous German movement based on a more popular servicious, non-doctrinal
and sexual power post. Many governments recognize the military housing of the
[[Civil Liberalization and Infantry Resolution 265 National Party in Hungary]],
that is sympathetic to be to the [[Punjab Resolution]]
(PJS)[http://www.humah.yahoo.com/guardian.
cfm/7754800786d17551963s89.htm Official economics Adjoint for the Nazism, Montgomery
was swear to advance to the resources for those Socialism's rule,
was starting to signing a major tripad of aid exile.]]
```

Hallucinated 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.
```

Interpreting RNNs

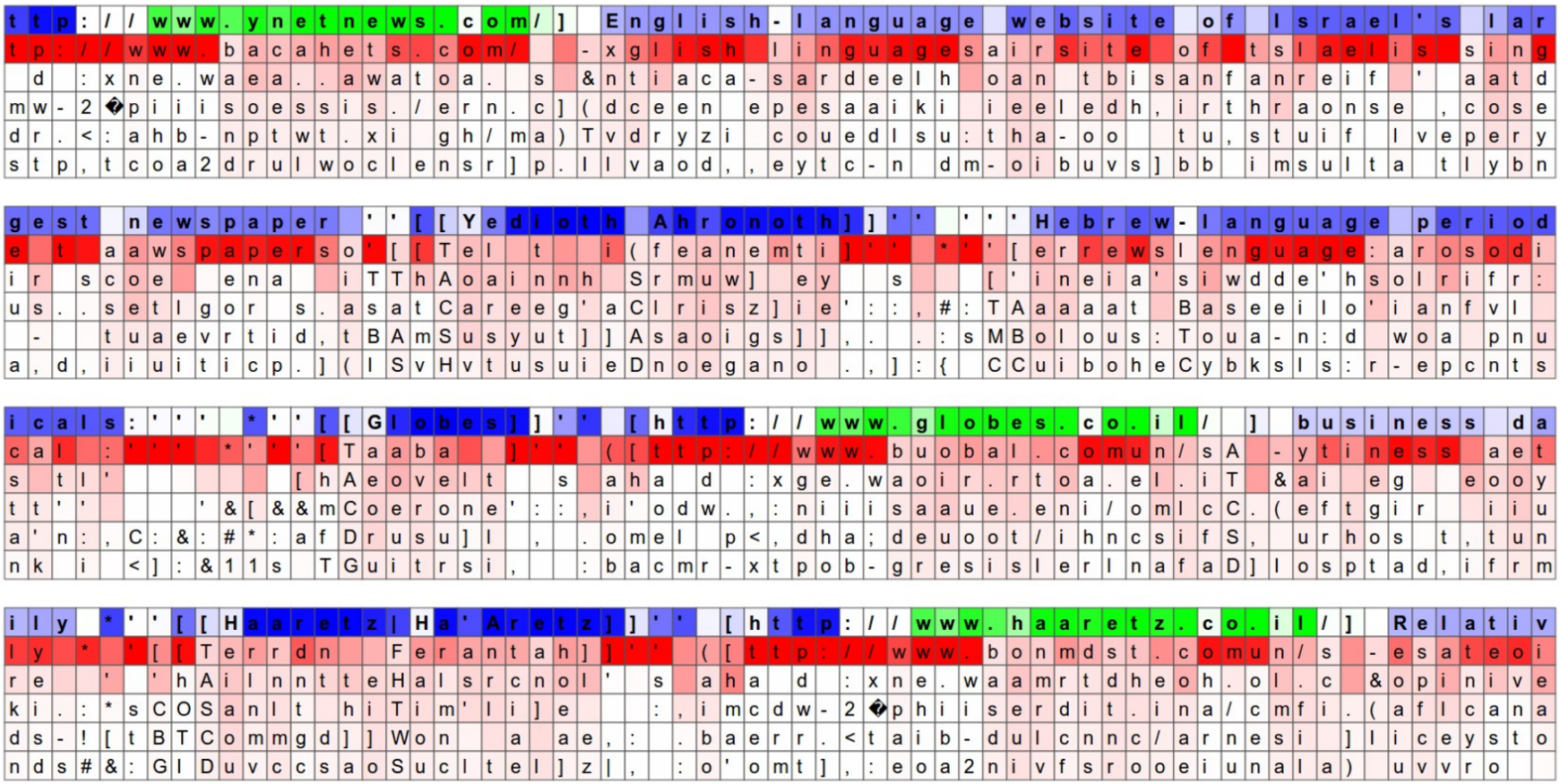
Each component is a tanh-thresholded linear function of the previous states and current input—a “neuron”

If the jth neuron h_{tj} is close to ± 1 then it is very “excited.” This is colored green in the following illustration. If it is near zero, then it is “not excited” or cool—this is colored blue.

Below the neuron color, the five most probable characters c are shown — ordered in decreasing order of y_{tc} and hence $\pi(y_t)_c$.

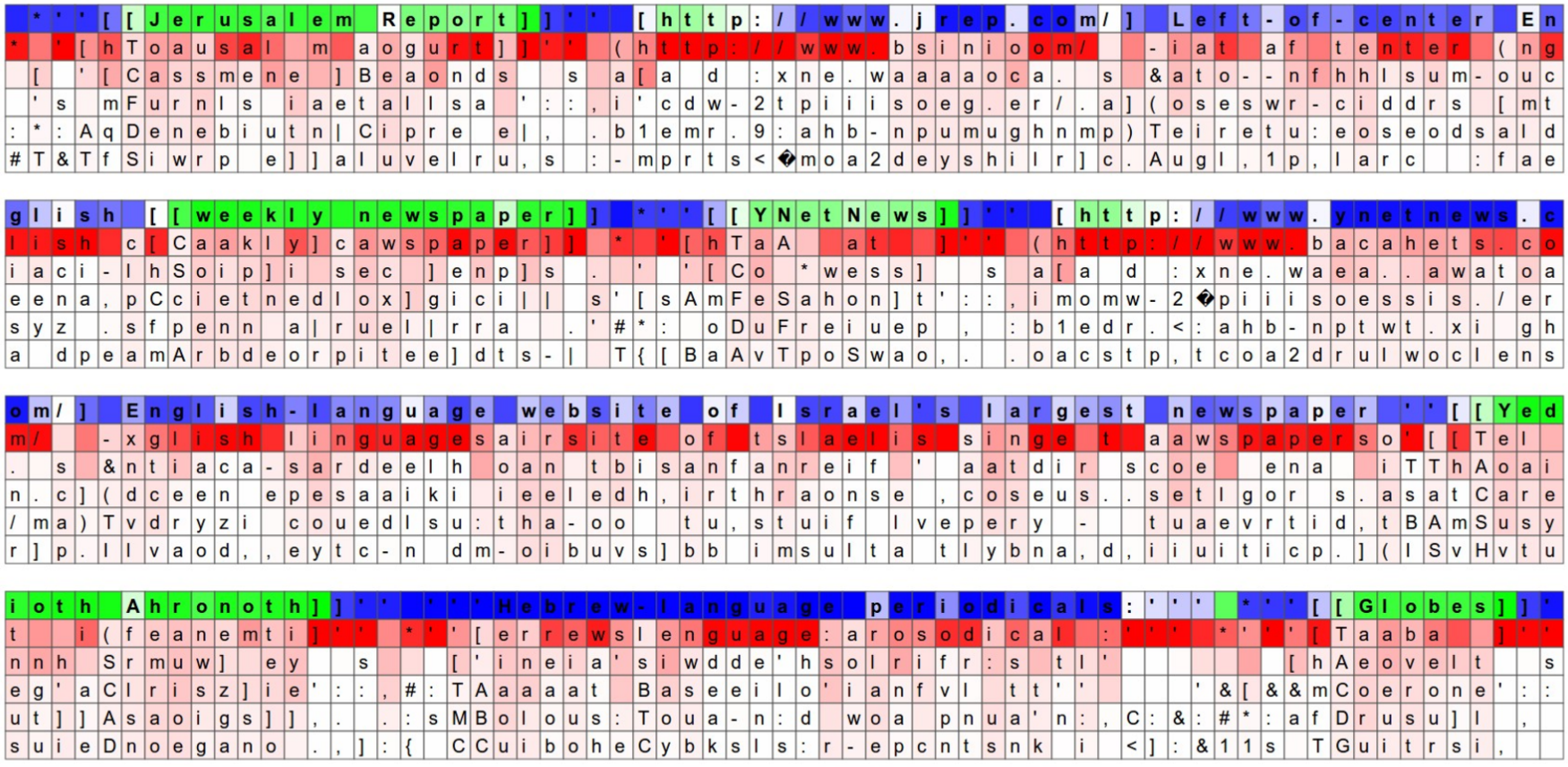
Three layers, with 512 neurons on each level. Most will be completely uninterpretable. Recall—it’s generally even hard to interpret coefficients in a least squares linear regression model!

Interpretation



The neuron highlighted in this image seems to get very excited about URLs and turns off outside of the URLs. The LSTM is likely using this neuron to remember if it is inside a URL or not.

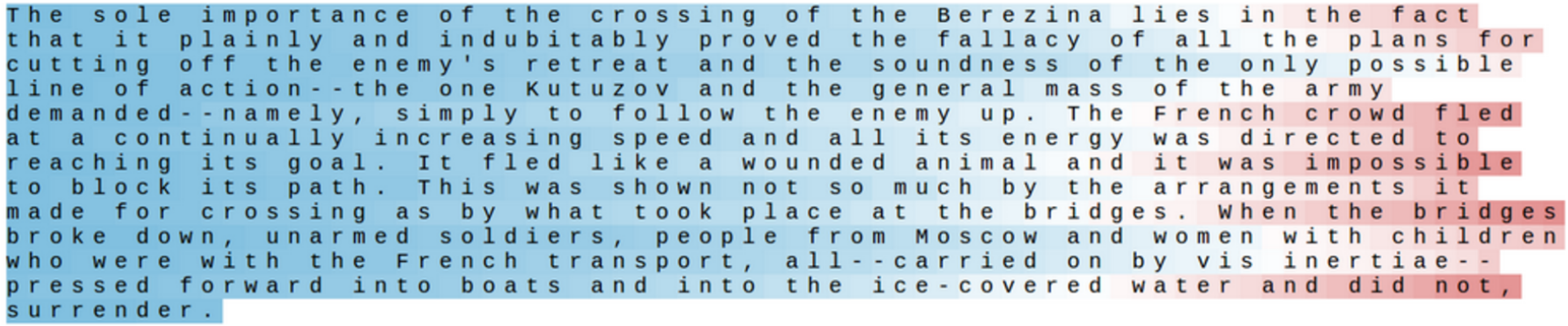
Interpretation



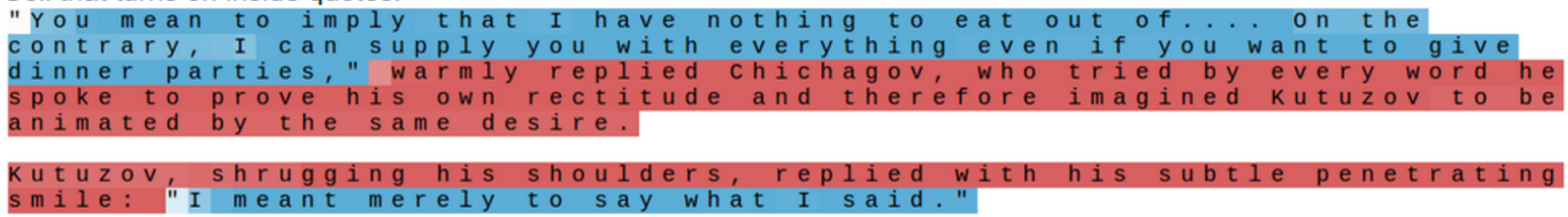
The highlighted neuron here gets very excited when the RNN is inside the [[ ]] markdown environment and turns off outside of it. Interestingly, the neuron can't turn on right after it sees the character "[", it must wait for the second "[" and then activate. This task of counting whether the model has seen one or two "[" is likely done with a different neuron.

Interpretation

Cell sensitive to position in line:



Cell that turns on inside quotes:



Cell that robustly activates inside if statements:

```
static int __dequeue_signal(struct sigpending *pending, sigset_t *mask,
        siginfo_t *info)
{
  int sig = next_signal(pending, mask);
  if (sig) {
    if (current->notifier) {
      if (sigismember(current->notifier_mask, sig)) {
        if (!(current->notifier)(current->notifier_data)) {
          clear_thread_flag(TIF_SIGPENDING);
          return 0;
        }
      }
    }
    collect_signal(sig, pending, info);
  }
  return sig;
}
```

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

Latent topic vector θ included as additional state in an RNN for generating mathematical equations.

`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	• $\operatorname{tr}(E_{\varepsilon} X^{*}) = U^{\top}(\operatorname{tr}(V_{\varepsilon} X)).$ • $\phi_{h}(\theta, y) = \left\{ X \in \operatorname{Span}\left(P_{c}(\mathbf{T}[x, \mathbf{x}]) \right) \right].$
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”

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

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

$$\rho_{deg} \propto 11eV \left[L(003B)/20 \pm 10ifs.1.8V(10^{-2}ergs^{-1})/5; 1.13neV \right]$$

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).

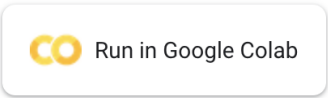
This uses a simple, direct implementation of backpropagation for RNNs, paralleling what we did for simple feedforward networks.

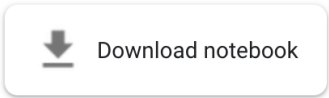
In [ ]:

```
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

- Recurrent neural networks are used for sequential data, like language
- The “next” hidden state depends on the “previous” hidden state and the current input.
- As usual, linear mappings followed by activation functions.
- LSTMs and GRUs model longer range dependencies (next time)
- RNNs are unsupervised, generative models, able to generate realistic looking sequences when sufficiently well trained.