LSTM Playground

MIND 2019, NIT Kurukshetra

Twisha Naik and Falak Shah

Infocusp Innovations Private Limited

2nd March 2019

Contents

Machine Learning

Machine learning is the science of getting computers to act without being explicitly programmed.

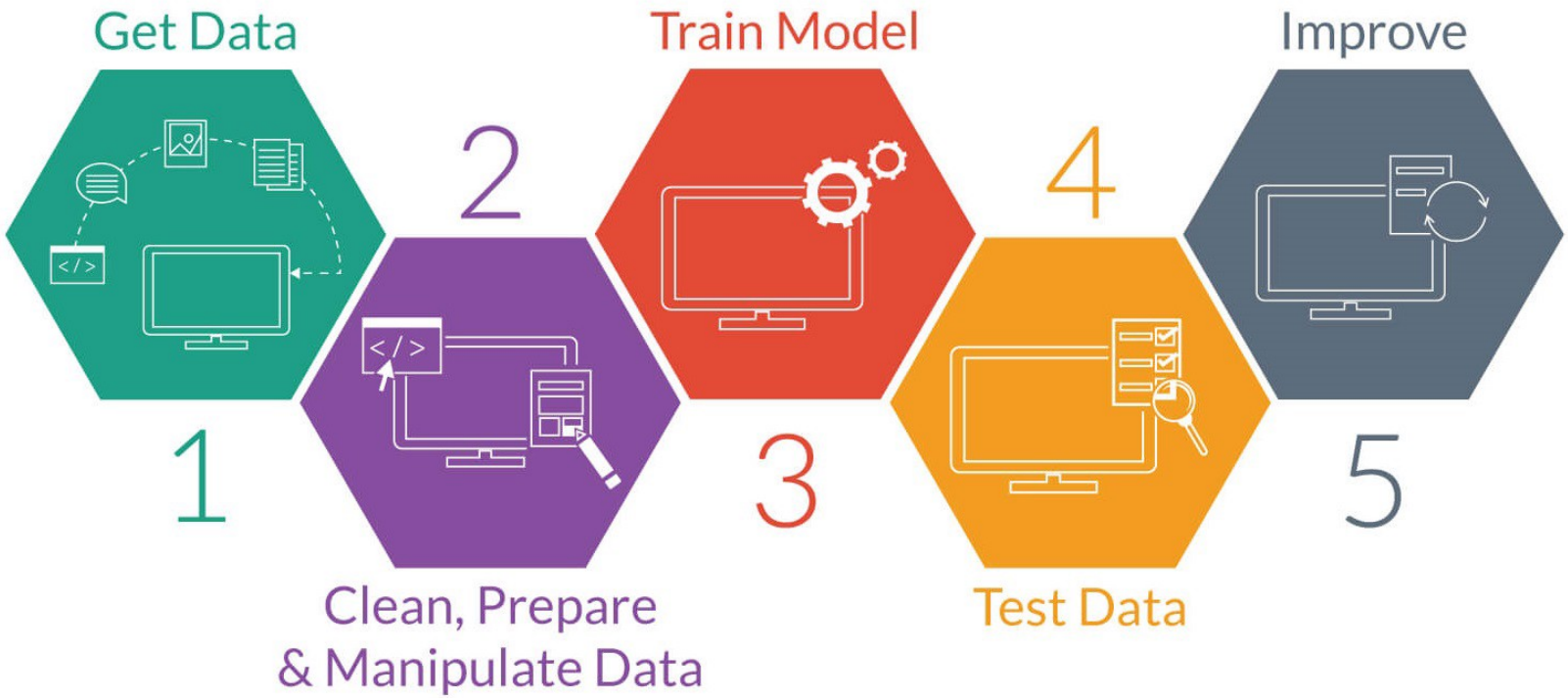


Figure: Machine Learning pipeline [1]

[1] https://upxacademy.com/artificial-intelligence-foundation/

Supervised Learning

Trying to learn a function which maps input (features) to correct output (continuous or discrete).

Area (feet^2)	Price (1000\$s)
1210	234
1450	266
750	172
1200	232
1380	257
840	184

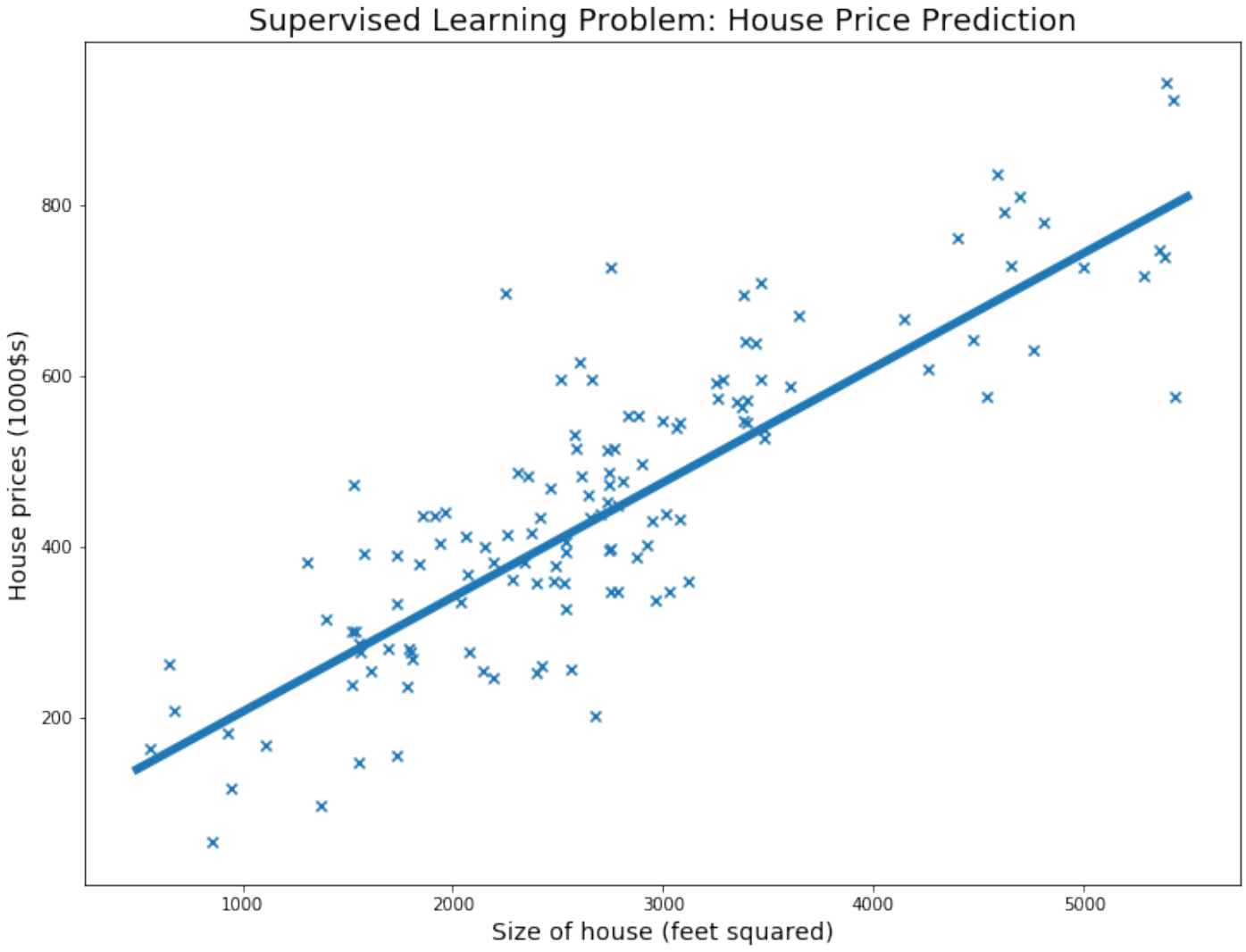


Figure: Line fit for house price data- function is a line

Unsupervised Learning

Actual labels are not provided in the data. Patterns and clusters are determined from the given data. Example - Clustering news stories/ documents into categories.

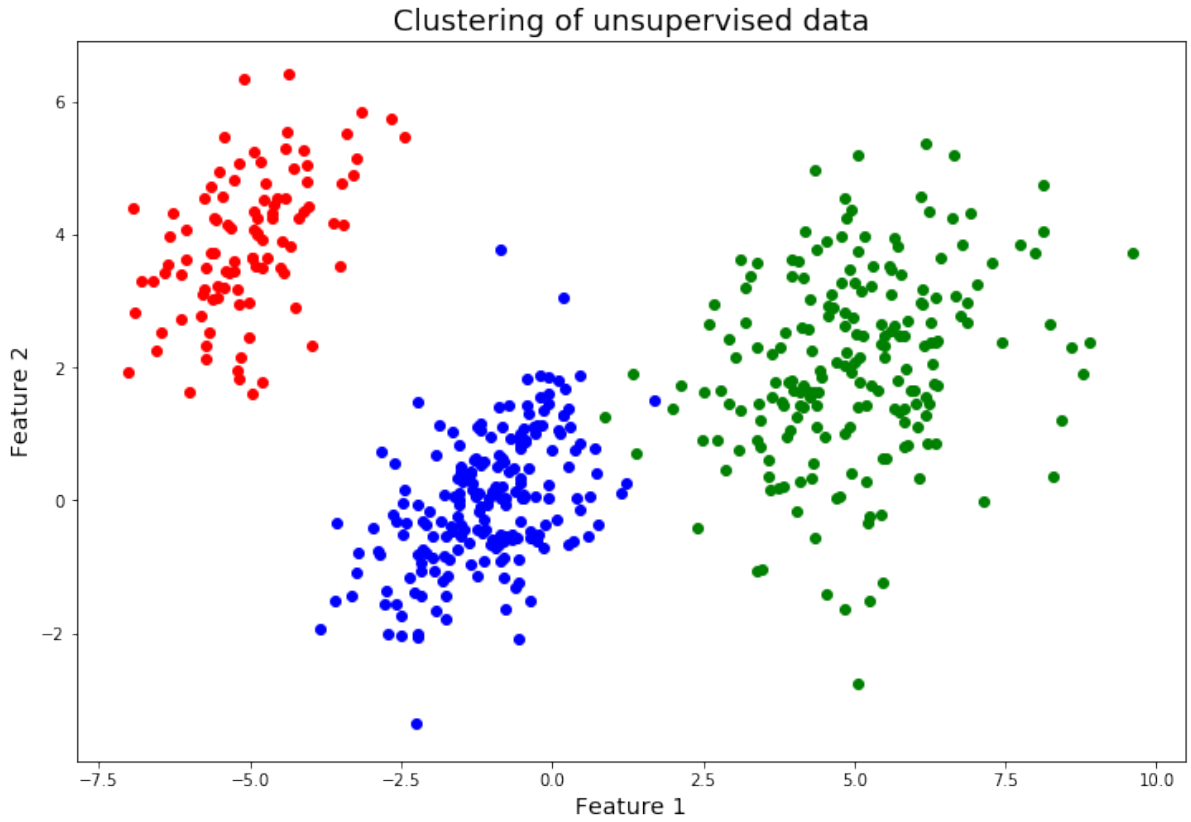


Figure: Unsupervised Learning task

Semi - supervised Learning

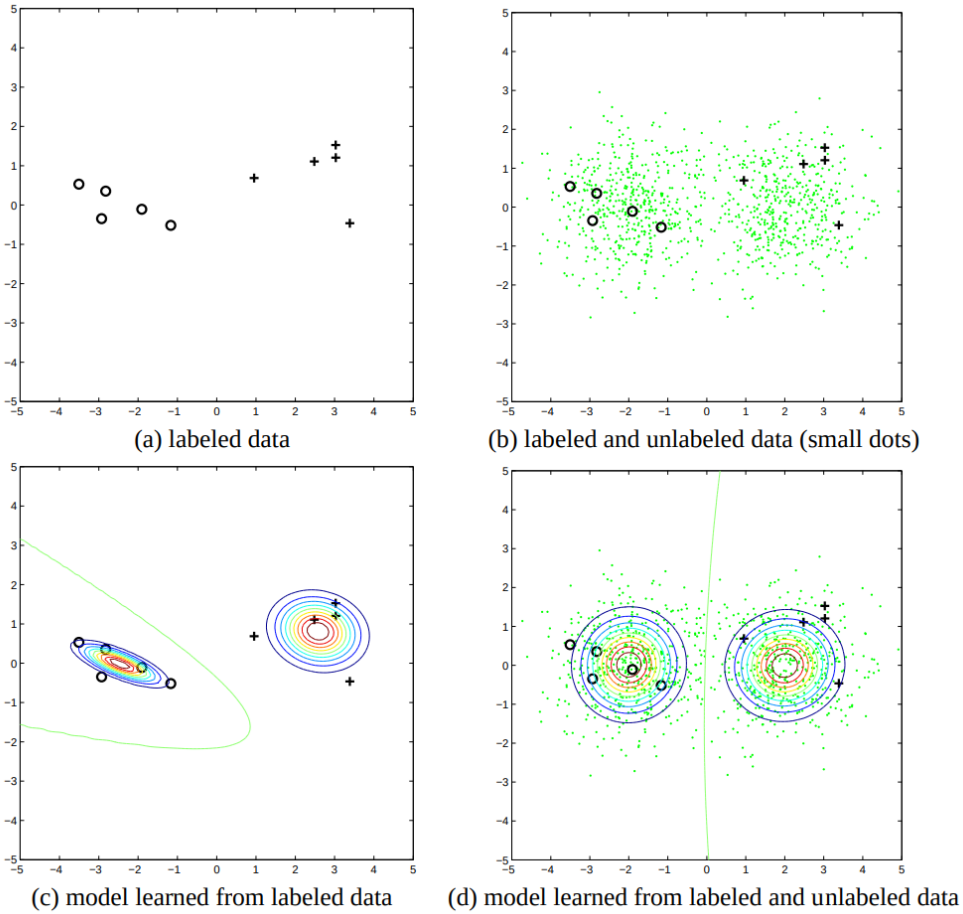


Figure: Semi supervised Learning

Semi - supervised Learning can be interpreted as:

- Supervised Learning + Additional Unlabelled Data
- Unsupervised Learning + Additional Labelled Data

Here, we are trying to predict Gaussian distributions that generated the given data.

[2] http://pages.cs.wisc.edu/~jerryzhu/pub/ssl_survey.pdf

Reinforcement Learning

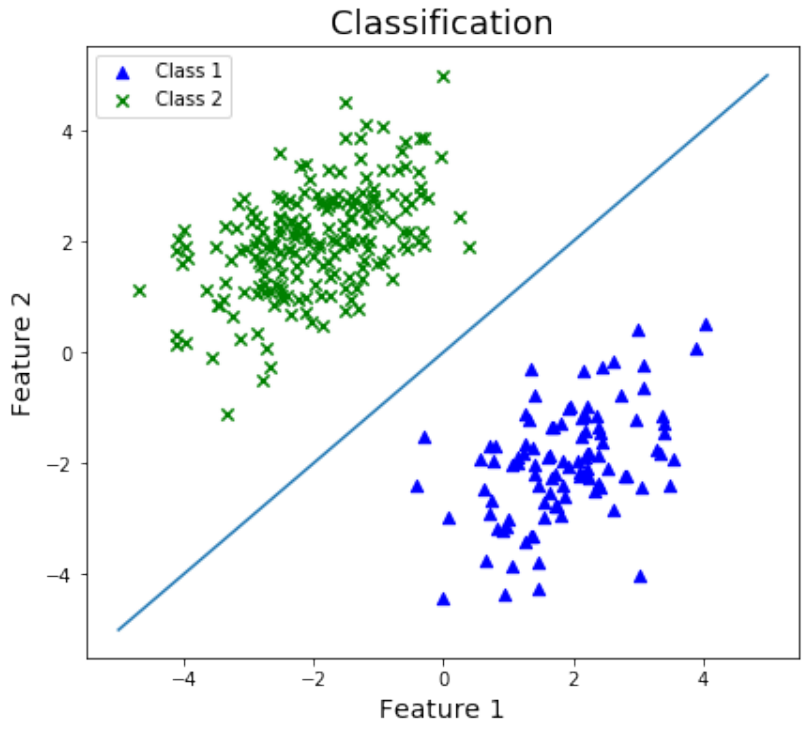
The "cause and effect" idea can be translated into the following steps for an RL agent:

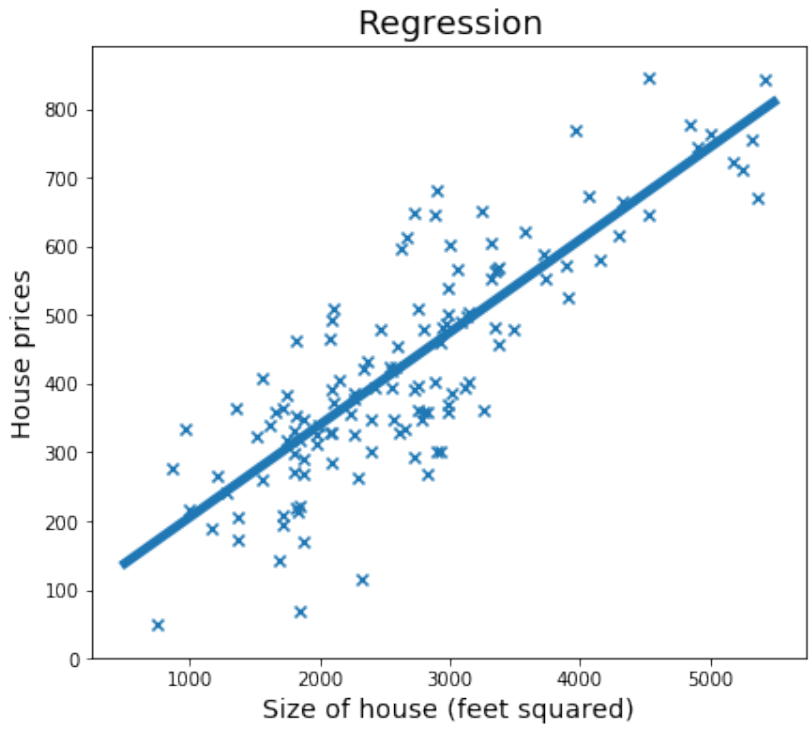
1. The agent observes an input state
2. An action is determined by a decision making function (policy)
3. The action is performed
4. The agent receives a scalar reward or reinforcement from the environment
5. Information about the reward given for that state / action pair is recorded

Our focus: Supervised Learning

Given: Pair of input and output values.
Goal: Define a loss function and minimize that to predict accurate output values **for unseen data**.
Real life problem $\rightarrow$ Optimization problem $\rightarrow$ Minimizing loss function

Regression and Classification problems





Examples:

- Classify images into different classes.
- Housing price prediction based on features of house.

Linear Regression

- One of the simplest ML algorithms
- Trying to find a linear mapping from input to output

$$y = ax + b$$

$$J = \frac{1}{2}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

$$(a_{opt}, b_{opt}) = \underset{a,b}{\operatorname{argmin}}(J)$$

Gradient Descent Intuition

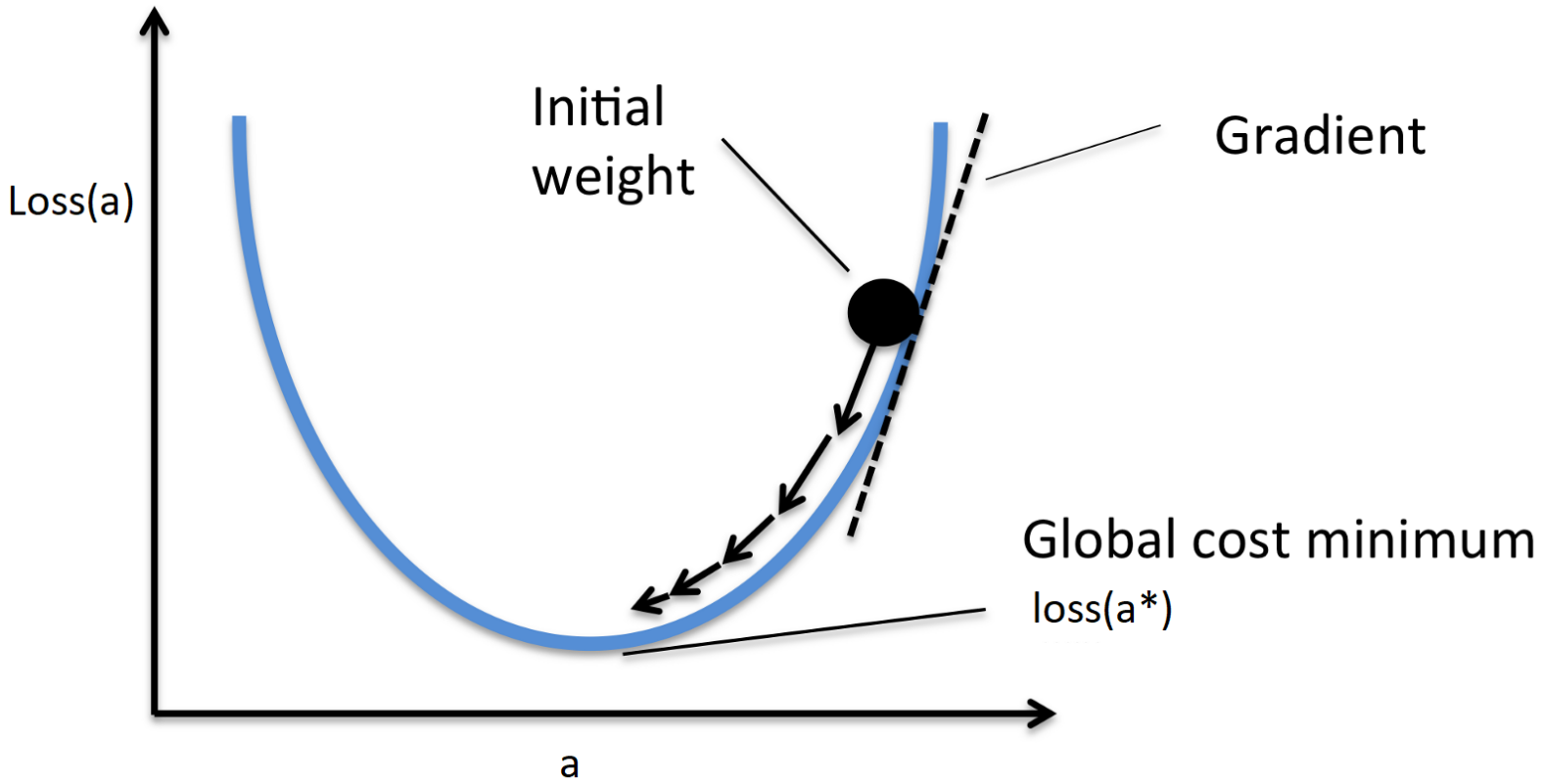
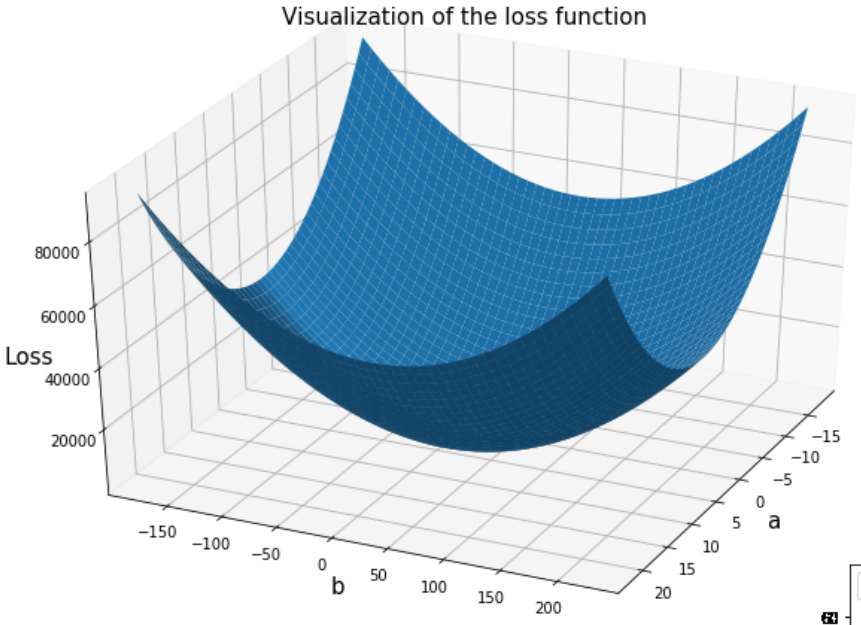
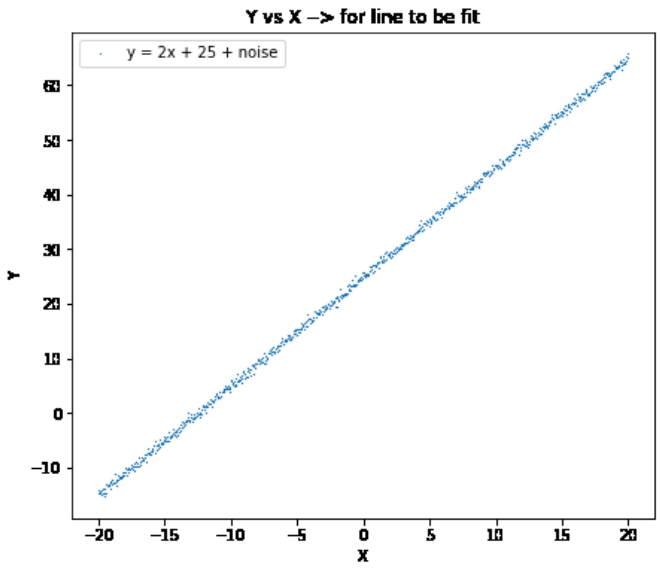


Figure: Gradient descent intuition [3]

[3] https://medium.com/@lachlanmiller_52885/machine-learning-week-1-cost-function-gradient-descent-and-univariate-linear-regression-8f5fe69815fd

Data and loss visualization





Gradient descent for line fitting

$$J = \frac{1}{2}\sum_{i=1}^{n}(y_i - (ax_i + b))^2$$

$$J = \frac{1}{2}\sum_{i=1}^{n}(y_i^2 + (ax_i)^2 + b^2 + 2ax_ib - 2ax_iy_i - 2y_ib)$$

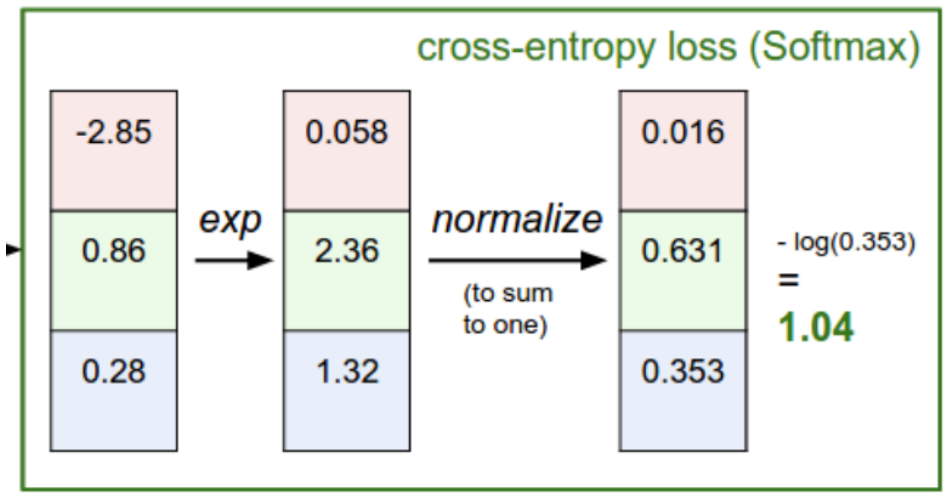
$$\frac{\partial J}{\partial a} = \frac{1}{2}\sum_{i=1}^{n}(2ax_i^2 + 2x_ib - 2x_iy_i)$$

$$\frac{\partial J}{\partial a} = \sum_{i=1}^{n} x_i(ax_i + b - y_i)$$

$$\frac{\partial J}{\partial b} = \sum_{i=1}^{n}(ax_i + b - y_i)$$

Classification problem

Softmax Loss



4

$$t = \begin{bmatrix} t1 \\ t2 \\ t3 \end{bmatrix}$$

$$loss = -\sum_{i=1}^{3} y_i log(P_i)$$

where $P_i = \frac{e^{ti}}{\sum_{j=1}^{3} e^{tj}}$

[4] CS231n lecture slides

Artificial Neural Networks

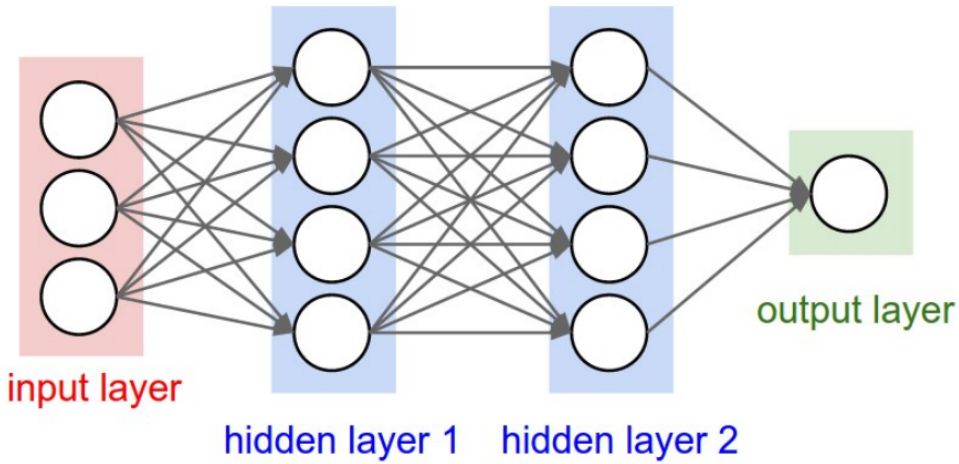


Figure: ANN

ANNs are used for modelling Non-linear hypothesis

$$H = activation(WX + b)$$

$$Y_{pred} = activation(WH + b)$$

$$Loss = f(y, y_{pred})$$

$$Eg \quad Softmaxloss = -log(p_{y_{true}})$$

[5] CS231n lecture slides

Activation Functions

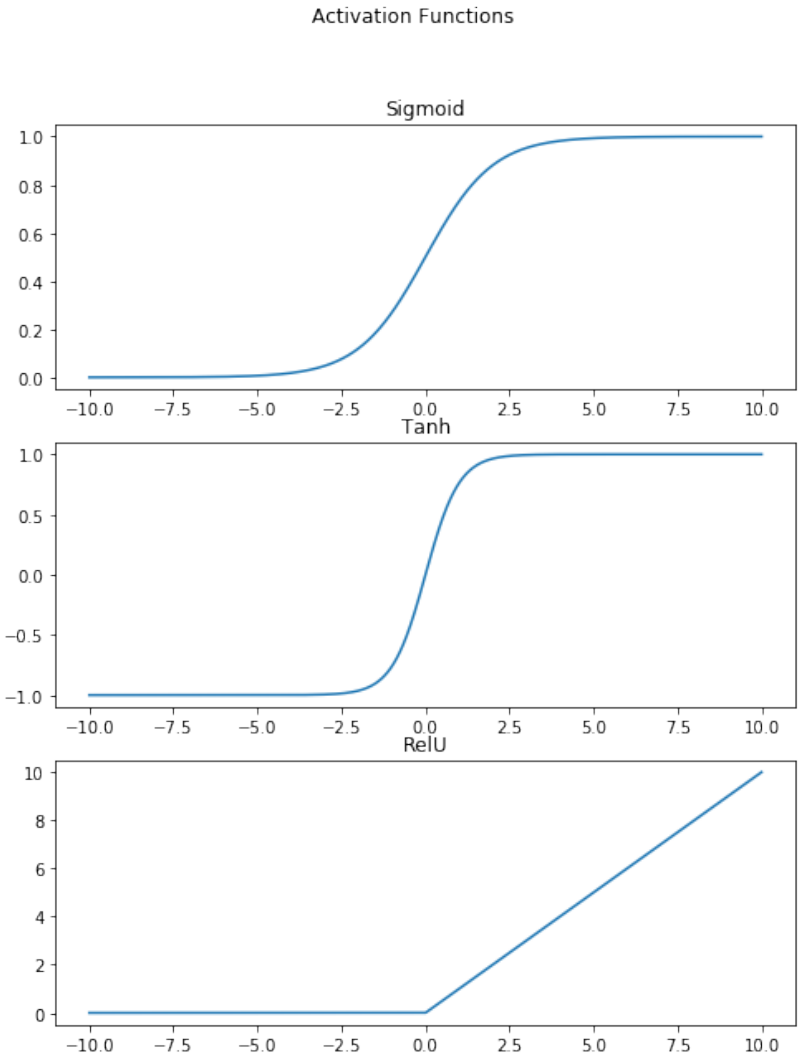
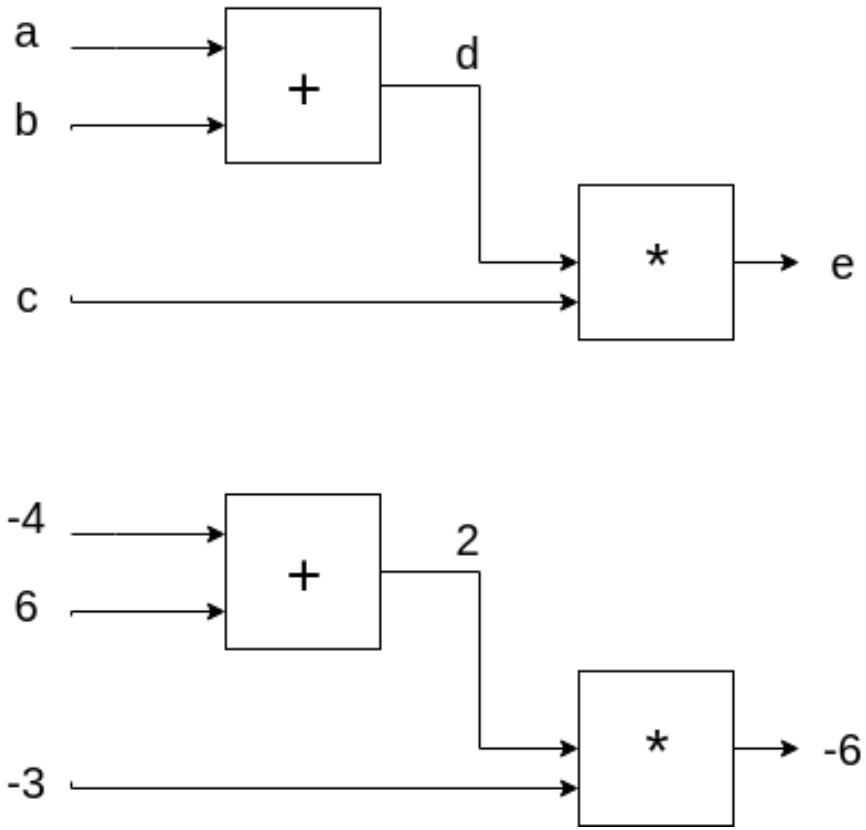


Figure: Activation functions

1. Sigmoid/ Logistic : $1/(1+e^{-x})$
2. Tanh - Hyperbolic tangent $(tanh(x))$
3. ReLU - Rectified Linear Units $(max(0,x))$

Backprop Intuition



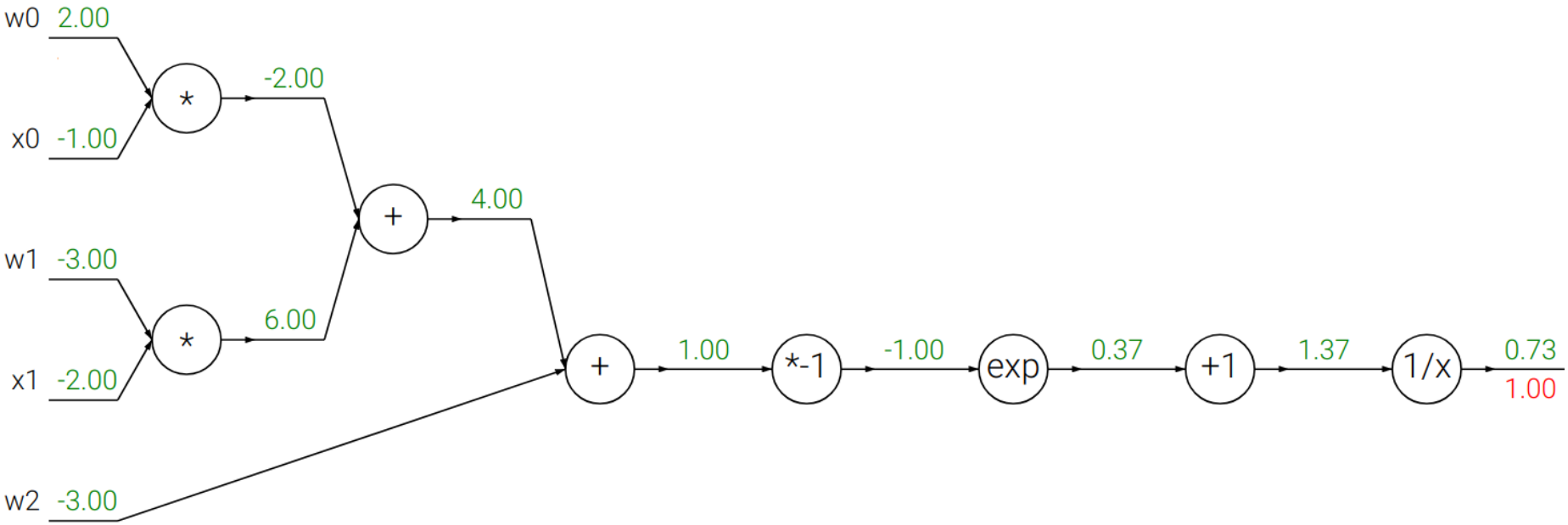
Once we have the loss at output layer, we want to back propagate that till the input layer such at we penalise the appropriate cells according to their contribution in the loss.

$d(a, b) = a + b$

$e(a, b, c) = (a + b) * c$

Code at Link

Backprop Example



Code in notebook

Backprop Example

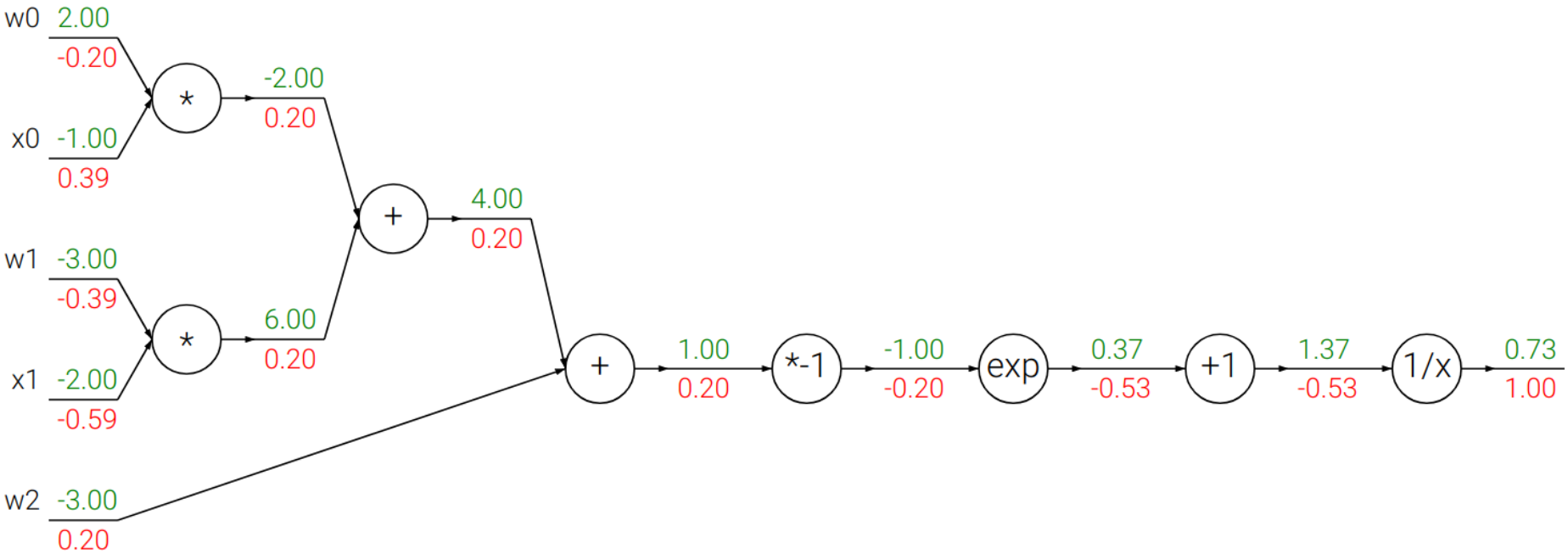


Figure: Backprop [6]

Code in notebook

[6] http://cs231n.github.io/

Drawbacks of ANN

- Fixed size input and output
- No concept of Memory
- Does not take into consideration dependency between sequential data

Recurrent Neural Networks (RNNs)

- RNNs take into consideration the current input as well as the inputs it has seen previously.
- This helps it understand the context of the input and make better predictions for sequential data.

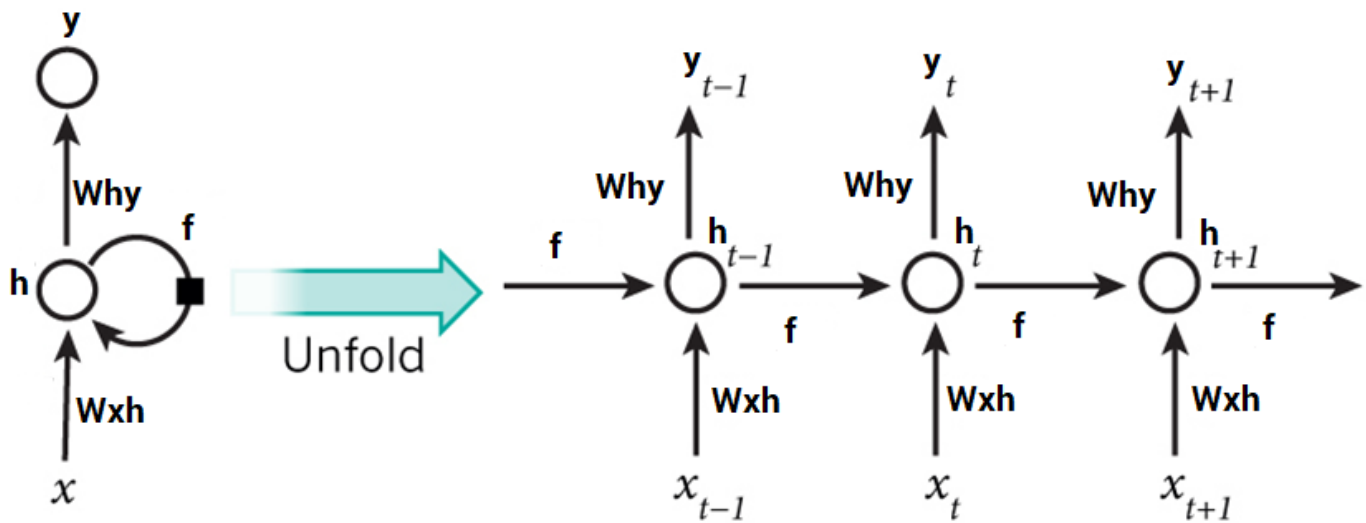


Figure: RNN Structure [7]

[7] https://www.analyticsvidhya.com/blog/2017/12/introduction-to-recurrent-neural-networks/

Introduction to RNNs

- Operates over sequences of vectors for input, output or both.
- Examples
  1. Image classification (Fixed size i/p and Fixed size o/p - Without RNN)
  2. Image captioning (Sequence o/p)
  3. Sentiment analysis (Sequence i/p)
  4. Machine translation (Sequence i/p and Sequence o/p)
  5. Video frame classification (Synced sequence i/p and o/p)

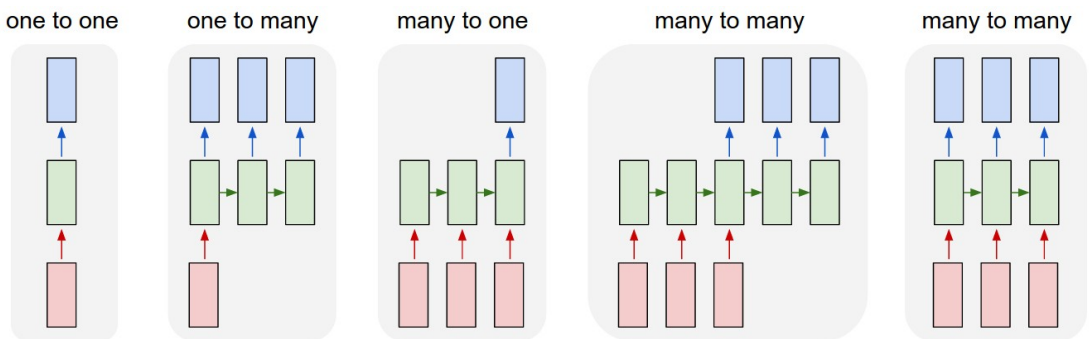


Figure: Examples of RNN [8]

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

(Vanilla) RNN equations

$$h_t = f_W(h_{t-1}, x_t)$$

new state

old state

input vector at some time step

some function with parameters W

Figure: RNN State Equation

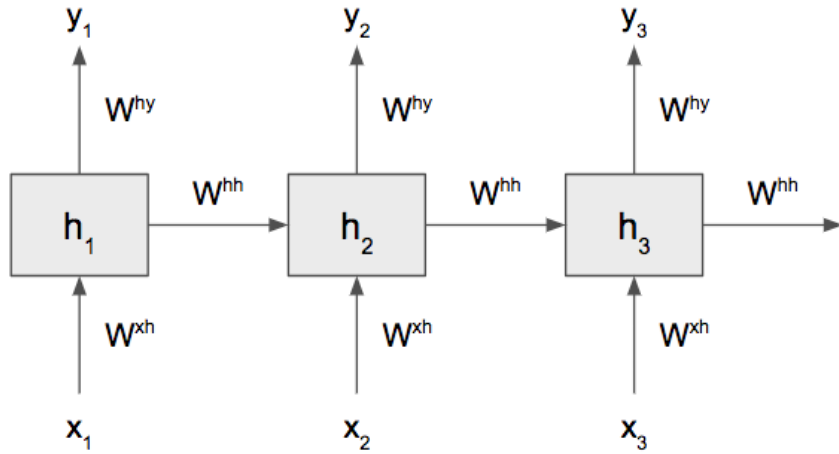
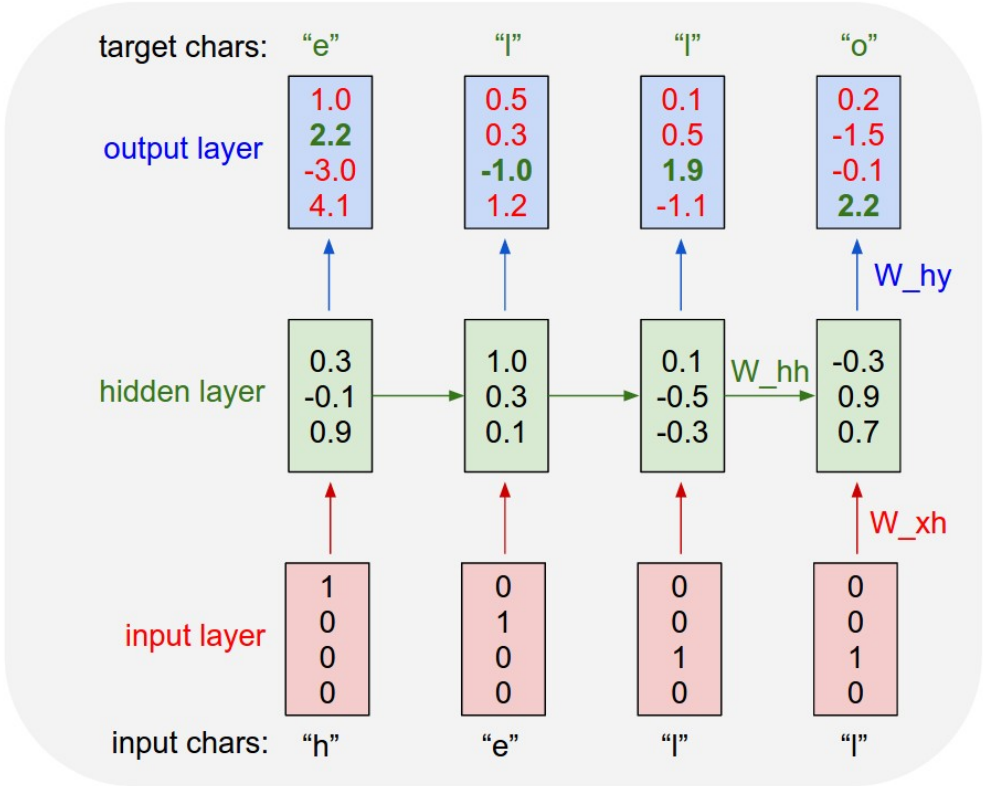


Figure: Unrolled RNN

$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t) \quad (1)$$

$$y_t = W_{hy}h_t \quad (2)$$

Char prediction using RNN



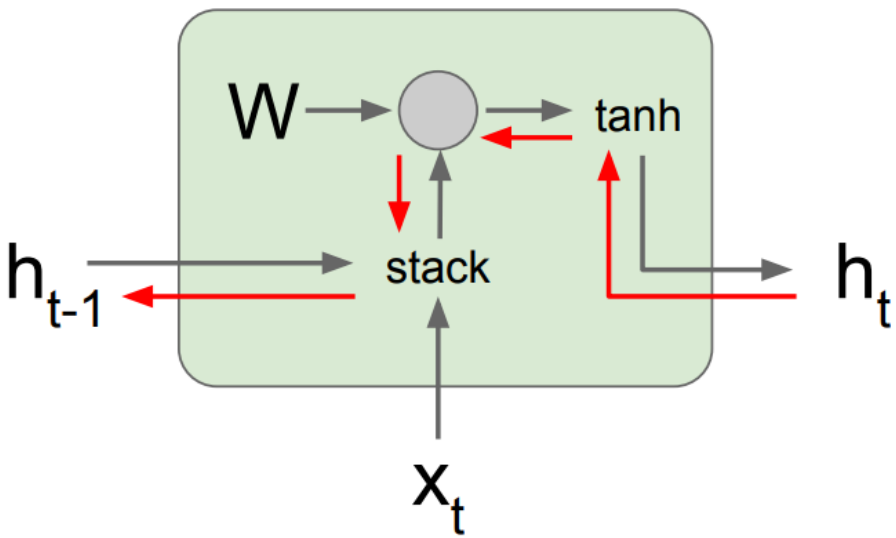
This will work better than ANN as it stores the information of previous character sequence in the hidden states

Figure: Char sequence generation using RNN

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

Backpropagation in RNN

When we backpropagate through an RNN cell, at each iteration we have to go through tanh and a multiplication with the weight matrix W.



$$h_t = tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$
$$= tanh\left((W_{hh}W_{xh})\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$
$$= tanh\left(W\begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}\right)$$

Figure: RNN
Backpropagation flow

Character Prediction Demo

Min-char RNN - a simple char prediction code using RNN
Code at Link

Drawbacks of RNN

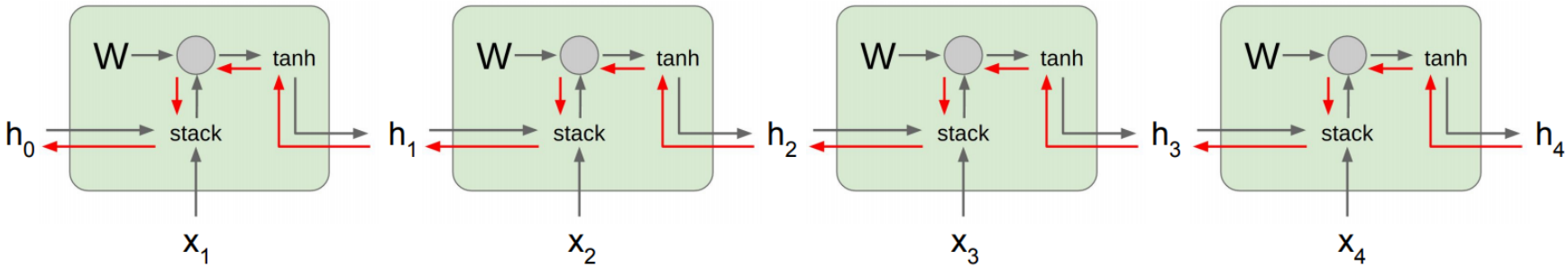


Figure: RNN Backpropagation chain

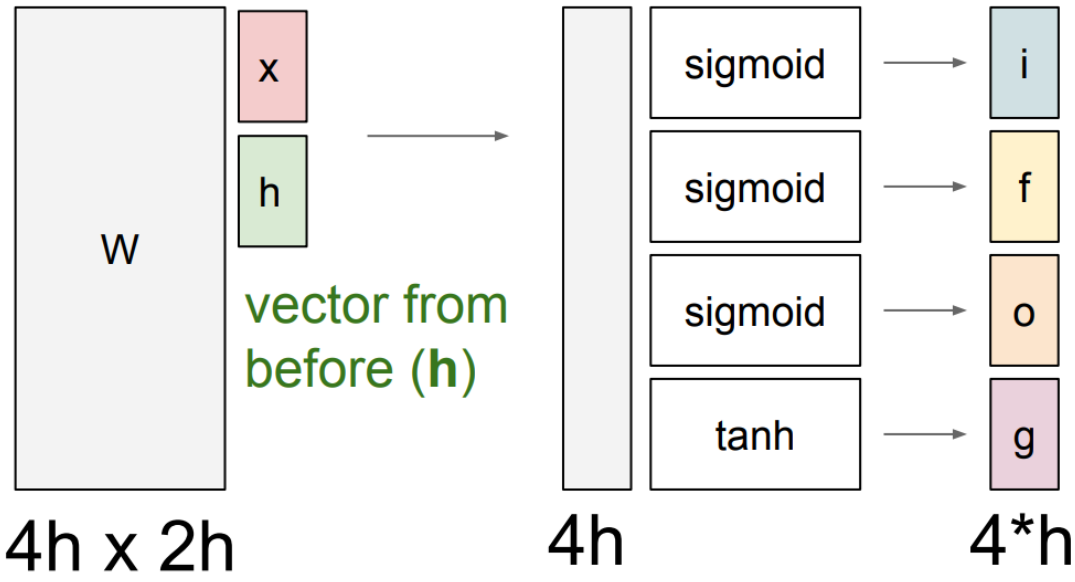
Two problems arise while backpropagating through an RNN network owing to repeated multiplication of gradients with W and application of tanh.

1. Exploding Gradient
2. Vanishing Gradient

Solutions to the RNN Problems

- The problem of exploding gradient can be solved by Gradient Clipping.
- To solve the problem of Vanishing gradient, we need to change the structure of RNN.

Long Short Term Memory Networks



$$\begin{pmatrix} i \\ f \\ o \\ g \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} W \begin{pmatrix} h_{t-1} \\ x_t \end{pmatrix}$$

$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

- i: Input gate, Whether to write to cell
- f: Forget gate, Whether to erase cell state
- o: Output gate, How much to reveal to the cell state
- g: Update gate, How much to write to cell state

LSTM Structure

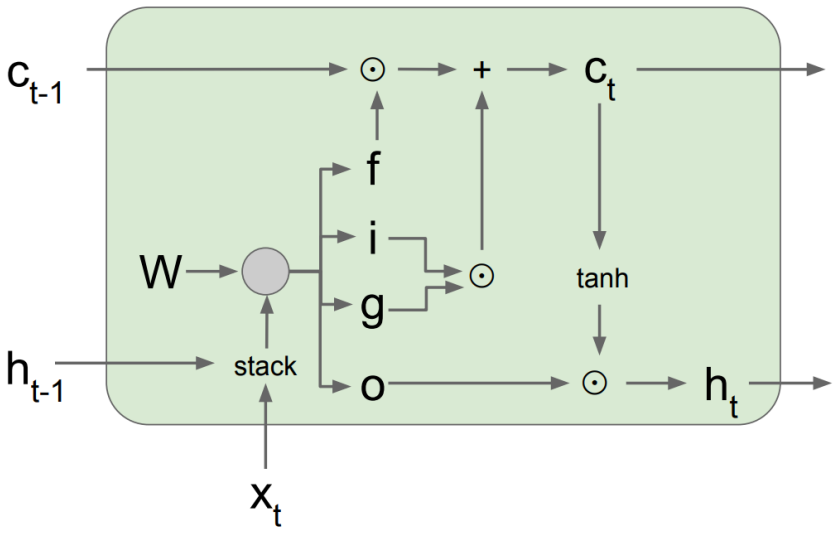


Figure: LSTM Structure

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o)$$

$$g_t = \tanh(W_{xg}x_t + W_{hg}h_{t-1} + b_g)$$

$$c_t = f_t \cdot c_{t-1} + i_t \cdot g_t$$

$$h_t = o_t \cdot \tanh(c_t)$$

Backpropagation in LSTM

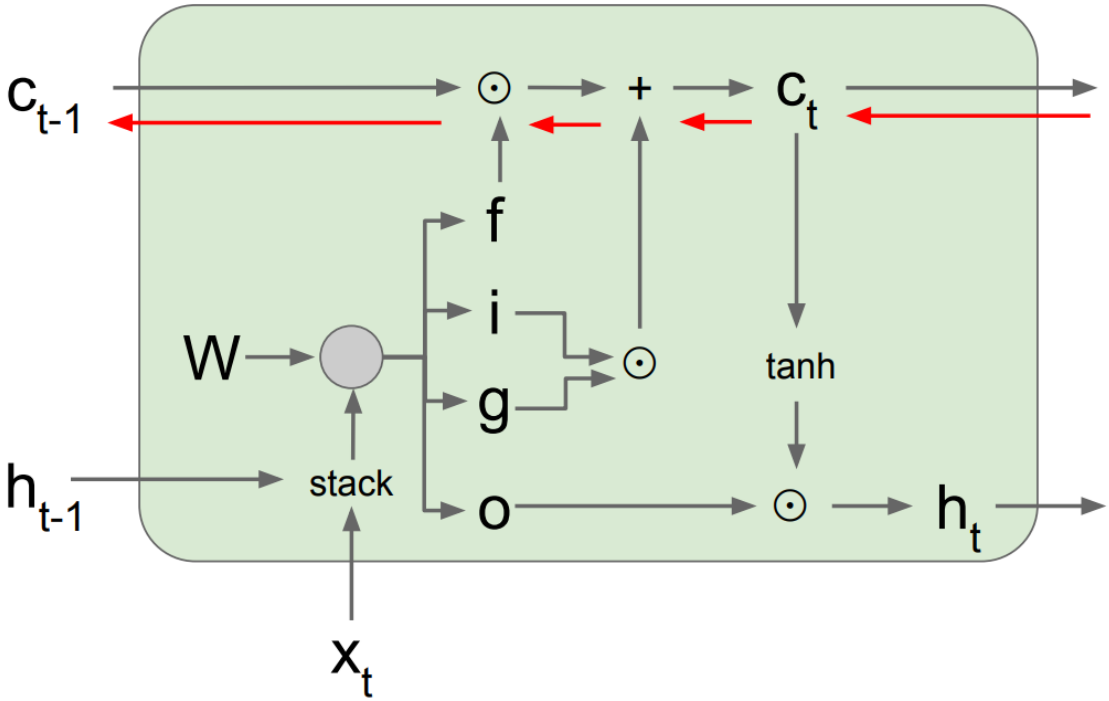


Figure: LSTM Structure

Backpropagation from c_t to c_{t-1} only elementwise multiplication by f, no matrix multiplication by W

Gradient Flow in LSTM

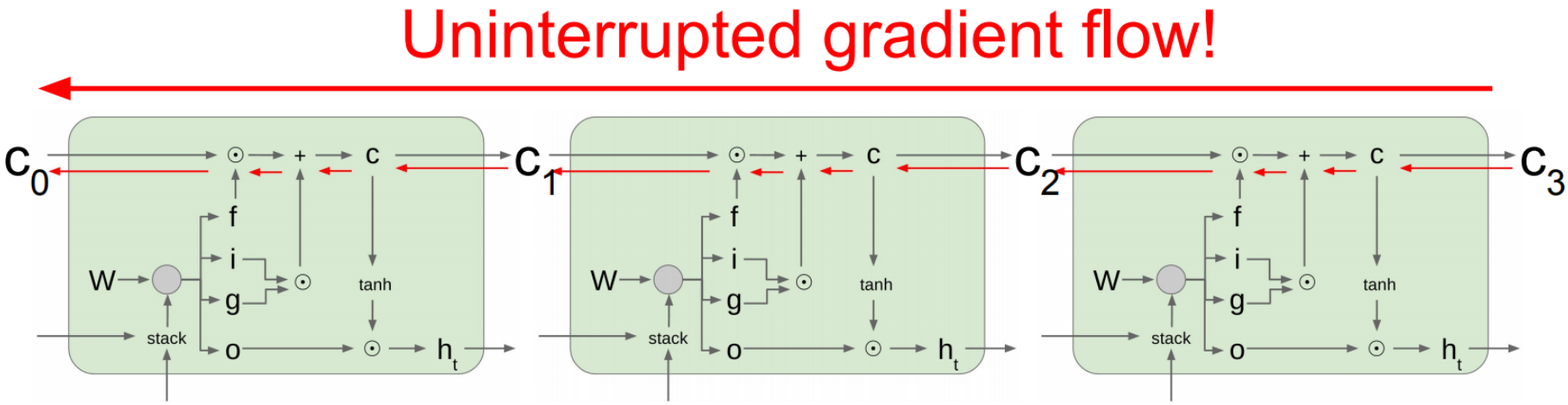


Figure: LSTM Structure

Super-highway for gradient flow which solves the issue of vanishing gradients.

Stacked RNN structure

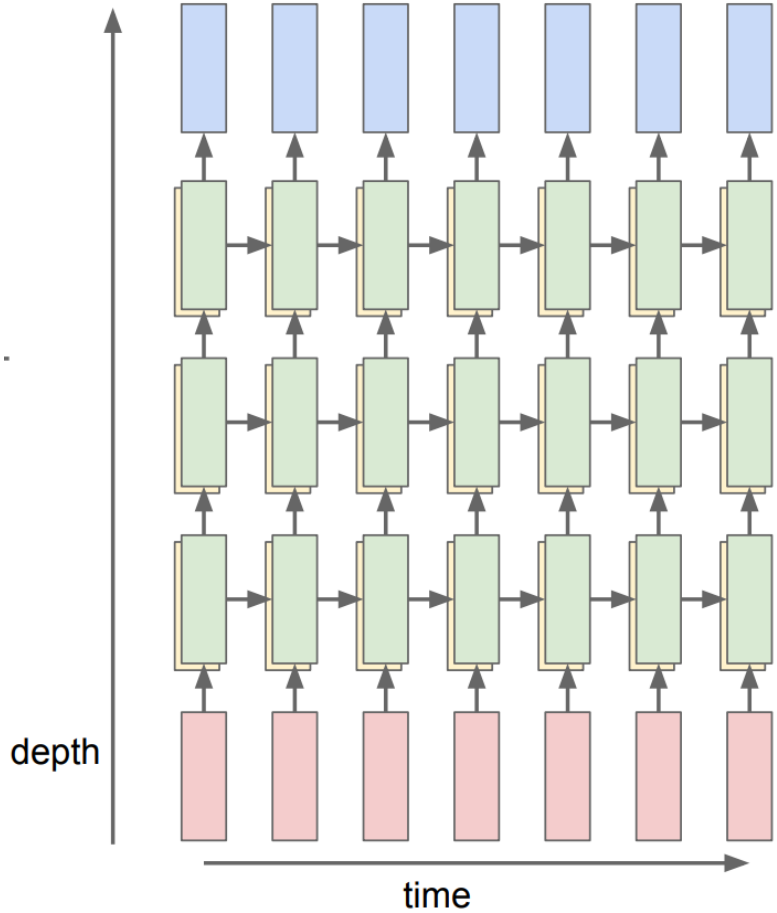


Figure: Stacked RNN

- When we have a stacked structure, for the higher layers, we will have outputs from the previous layer RNN as an input.
- The new equation will be:

$h_t^l = tanh(W^l \begin{bmatrix} h_t^{l-1} \\ h_{t-1}^l \end{bmatrix})$ where $h \in \mathbb{R}^n$

Modeling time-series data in Keras

- Keras API - stateful LSTMs and LSTMs for time series
- Demo Link

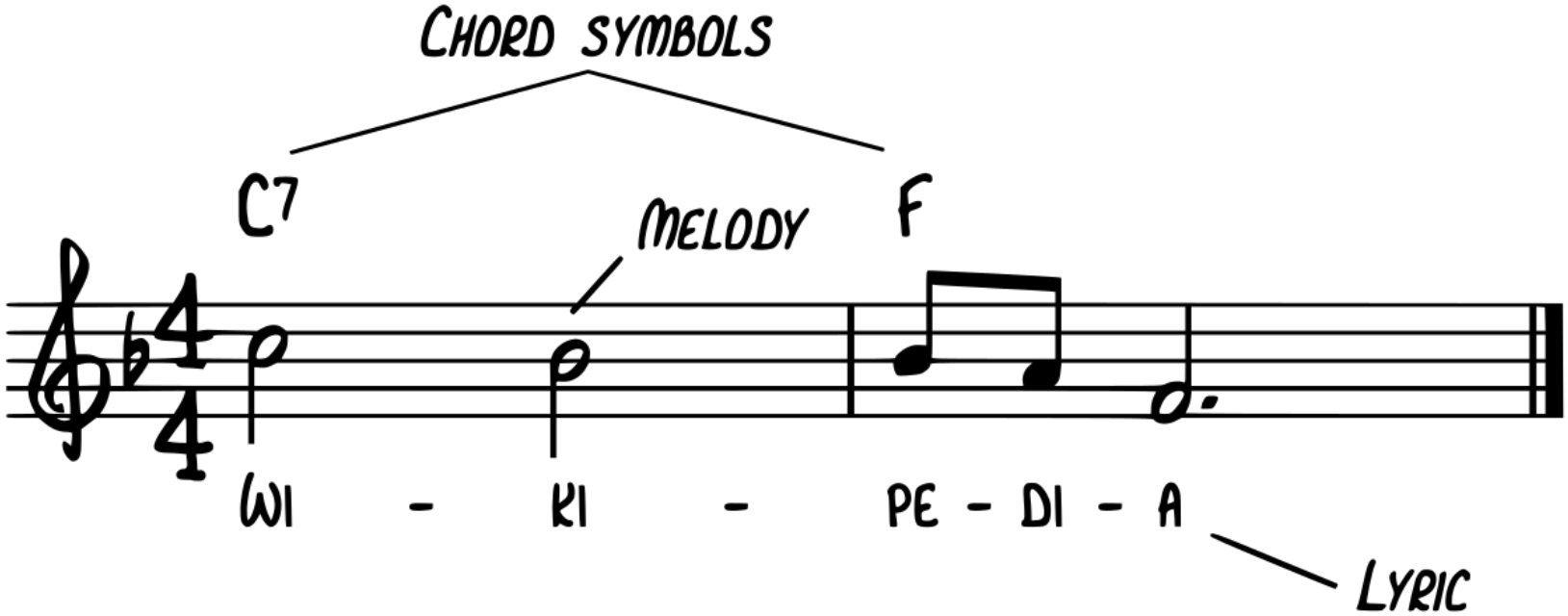
LSTM in action

- Keras character generation code - LSTM replicating Shakespeare, Harry Potter and Famous Five
- Demo Link
- Cell Learning Visualization - what are the cells learning
- Demo Link

Music series demo

- Magenta is an open source research project exploring the role of machine learning as a tool in creative process.
- One of its aspect is using deep learning models for music generation. (Another one is for sketch generation - one of the later demos)

Music Represenation



Music terminology

- Notes - Sa, Re, Ga, Ma, Pa, Dha, Ni (C to B) in different octaves.

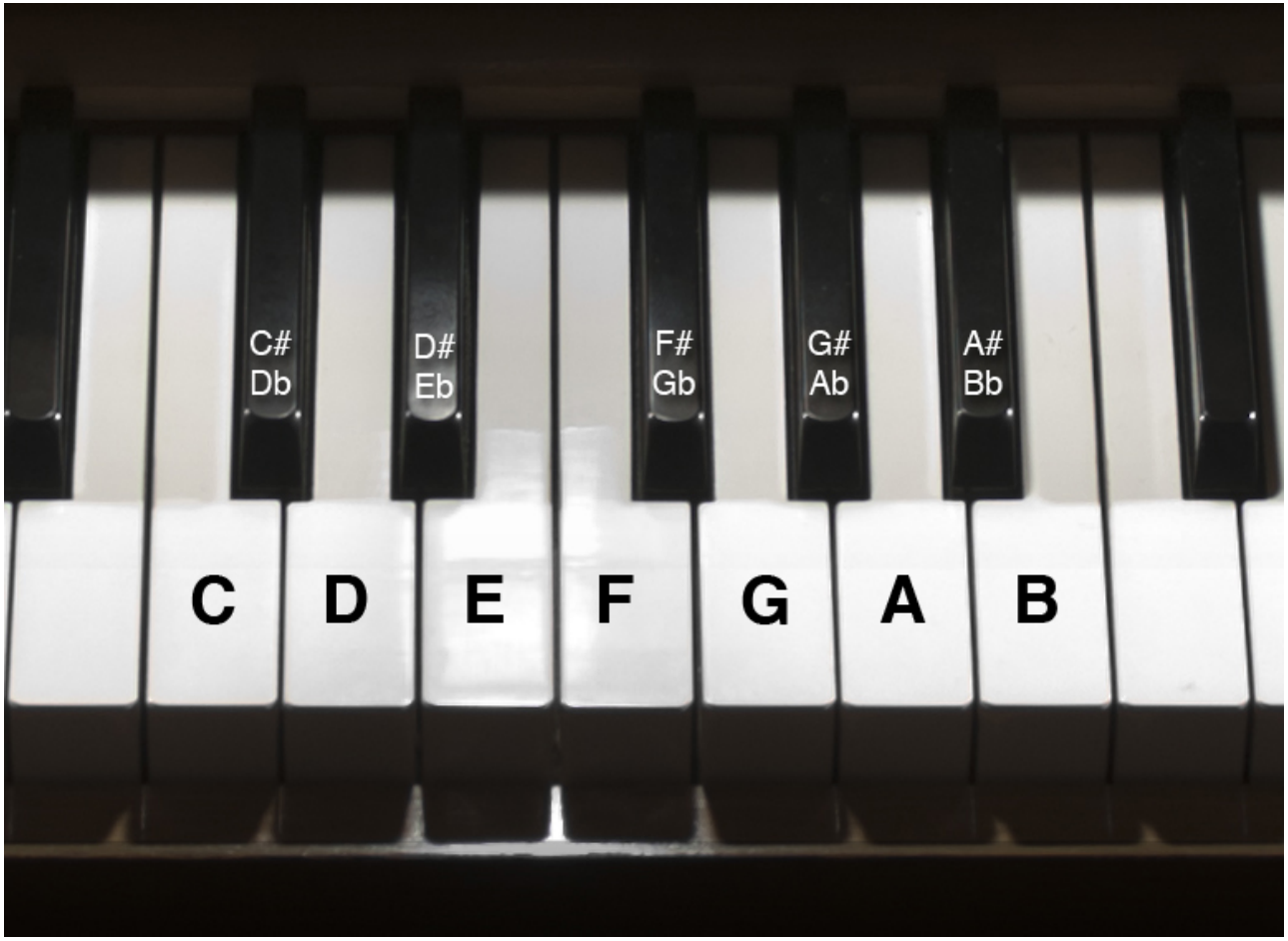


Figure: Piano notes

Music terminology

- Melodies - A sequence of notes
- Chords - Multiple notes played together to support melody
- Time Signature - A number denoted in form numerator over denominator (4/4) - numerator showing number of beats in a bar and denominator showing what kind of notes constitute one beat (one quarter note constitutes a beat if denominator is 4)
- QPM (Quarters per minute)

Music Data

- Digital representation of music: Music XML/ MIDI/ Note Sequence:

C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	1	2	3	4	5	6	7	8	9	10	11

- Convert notes to pitch:
  (12 + pitchclass + alter) + (octave * 12)
  Example: C4 = (12+0+0) + (4*12) = 12 + 48 = 60
- Special Events
  - Note Off event - Turn off a note currently playing.
  - No event - A note that is currently playing keeps on playing.

Music series demo

- Stacked LSTMs: Two layers of LSTMs with 128 hidden units each
- Initially, a primer sequence is given which sets the cell states of LSTM and which will help in predicting the next notes.
- Input - Previous note in one-hot encoded form and hidden state
- Output - Current note
- Demo Link

Melody Lookback RNN

- Provides some additional information to the input vector:
- In addition to the previous event, we also input the events from 1 and 2 bars ago. Easily recognize patterns that occur across 1 and 2 bars,
- We also input whether the last event was repeating the event from 1 or 2 bars before it. This signals if the last event was creating something new, or just repeating an already established melody..

Attention RNN

- Attention is one of the ways to use information from multiple previous states.
- x are the RNN intputs from the previous steps $(x_{t-n}, \ldots, x_{t-1})$ and s_{t-1} is the previous RNN cell state. These values are used to calculate e, an n length vector with one attention value for each of the previous n steps, which is then normalized by softmax to get a.
- The RNN inputs from the previous steps are then multiplied by these attention values and then summed together to get c_t.

$$e = f(s_{t-1}, s_{t-2}, .., s_{t-n}, x_t) \tag{3}$$

$$a = softmax(e) \tag{4}$$

$$c_t = \sum_{j=t-n}^{t-1} (a_j x_j) \tag{5}$$

Attention RNN Example

- For example, let's assume we are on the 4th step of our sequence and n=3, (look at just last 3 steps).
- For this example, the RNN output vectors will be length 4 vectors. If the RNN inputs from the first 3 steps are:

Step 1: [1.0,0.0,0.0,1.0]
Step 2: [0.0,1.0,0.0,1.0]
Step 3: [0.0,0.0,0.5,0.0]

- Suppose, calculated attention mask: $a_t = [0.7,0.1,0.2]$
- Then the previous step would get 20% attention, 2 steps ago would get 10% attention, and 3 steps ago would get 70% attention.

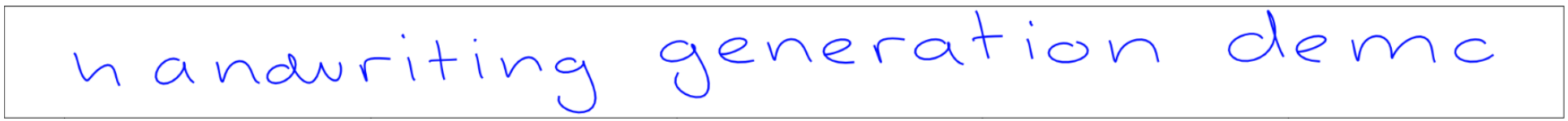
Attention RNN Example

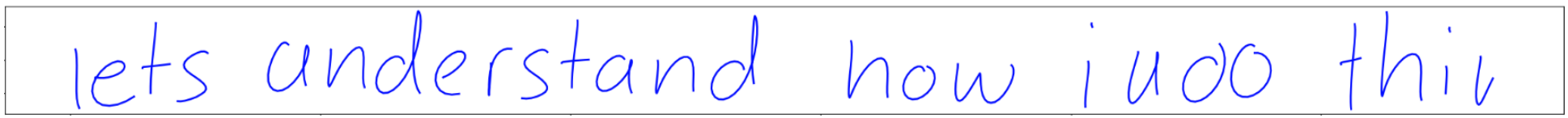
- So their masked values would be:

Step 1 (70%): [0.7,0.0,0.0,0.7],
Step 2 (10%): [0.0,0.1,0.0,0.1],
Step 3 (20%): [0.0,0.0,0.1,0.0]

- And then they'd be summed together to get c_t :
  $c_t = [0.7, 0.1, 0.1, 0.8]$
- The c_t vector is essentially all previous outputs combined together, but each output contributing a different amount relative to how much attention that step received.

Handwriting generation using LSTMs

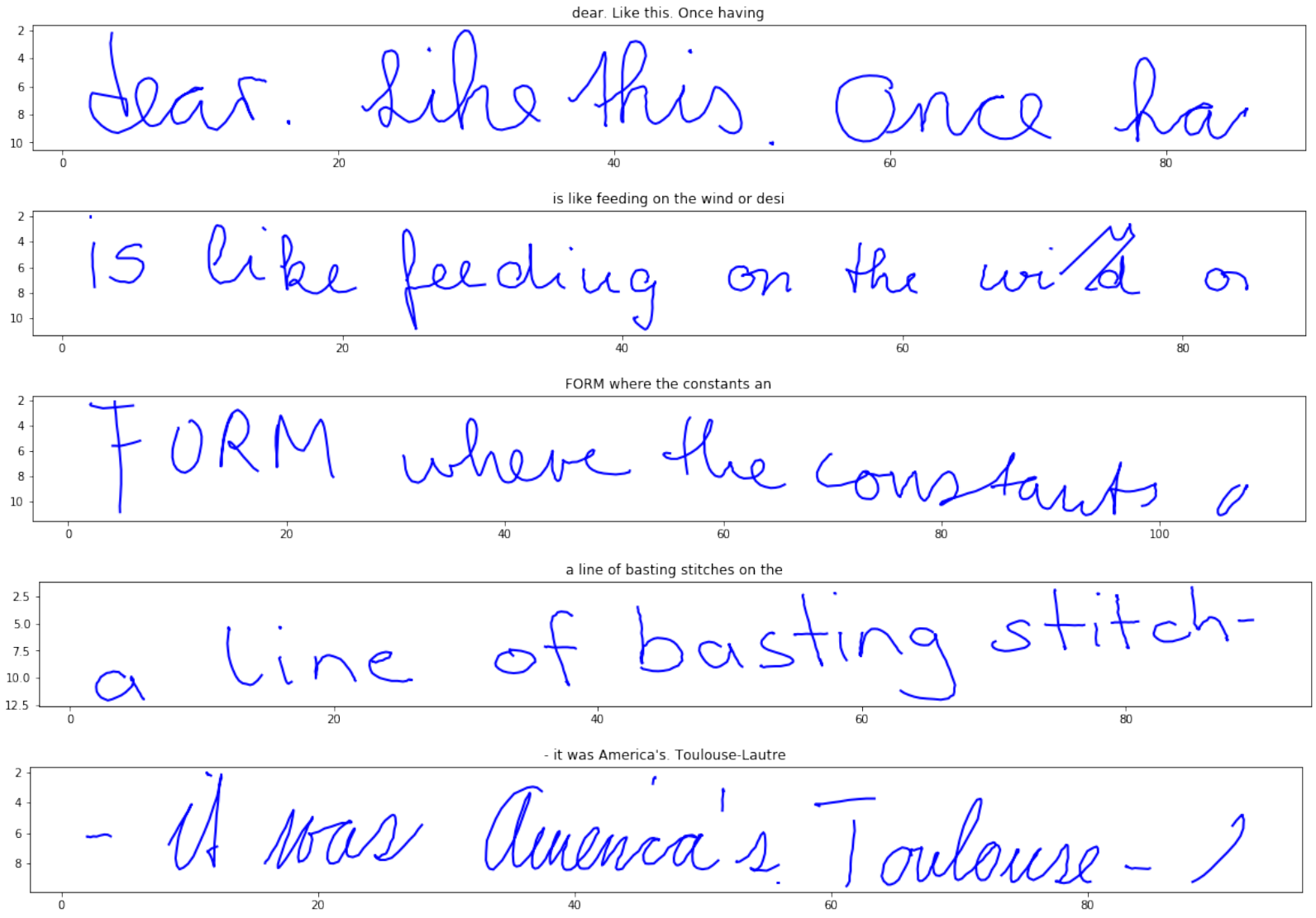




About the dataset

- Online handwriting data: the writing is recorded as sequence of pen tip locations as opposed to offline handwriting, where only page images are available.
- IAM online handwriting database
- The data has sample sentences written by 221 writers with the text and its corresponding stroke information.
- The stroke information is sequence of pen-tip locations (x and y pen coordinates + points when pen was lifted off the whiteboard) describing the entire trace of how the pen moved over time to write a specific sentence.

Data Samples



About the model

- Data: (x, y, pen_up), where x and y are offsets from previous inputs and *pen_up* signifies end of stroke.
- The outputs of a trained LSTM network are used to set parameters of a Gaussian mixture model which is then used to predict the three values.
- Outputs are drawn from mixture of gaussians whose distribution is being predicted by the network

Hand writing generation model

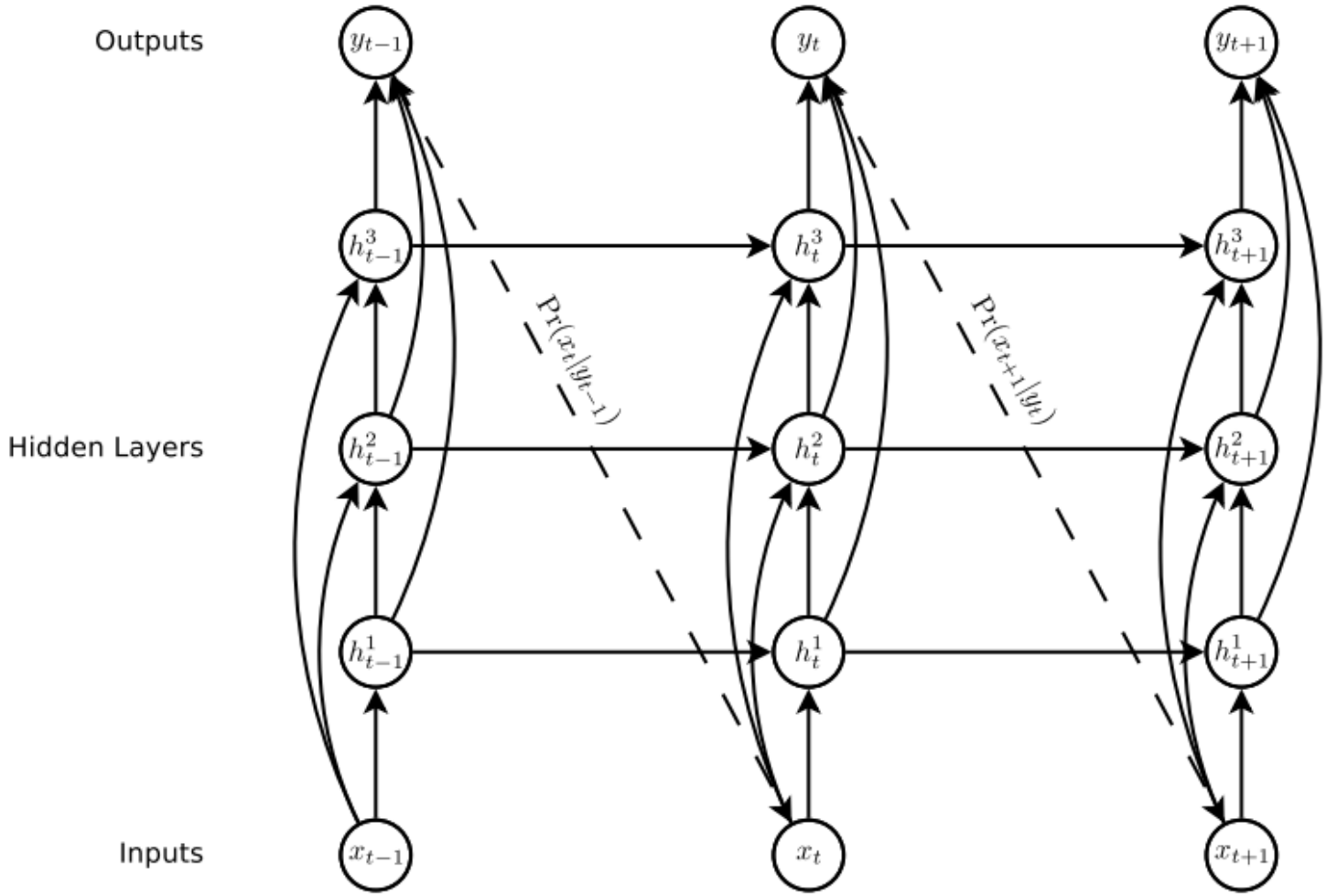
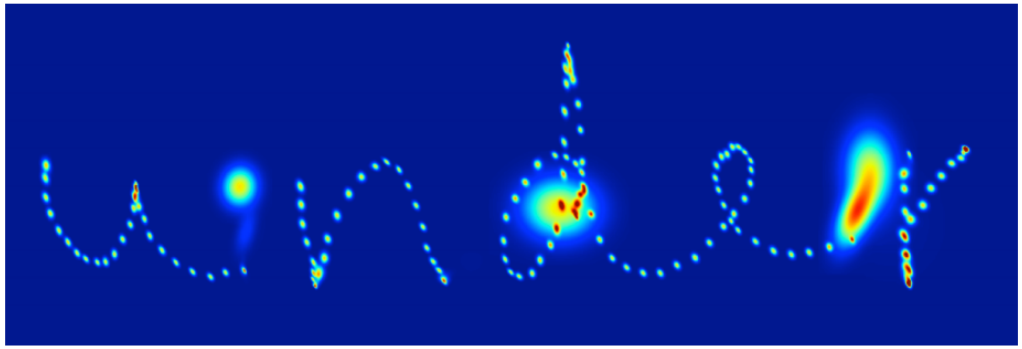


Figure: Model for handwriting generation

Generation Task



- (x, y, pen_up), where x and y are offsets from previous inputs and pen_up signifies end of stroke.
- The outputs of a trained LSTM network is used to set parameters of a Gaussian mixture model which is then used to predict the three values.
- It's a combination of LSTM and Mixture density networks
- Outputs are drawn from mixture of gaussians whose distribution is being predicted by the network

About the model

- A mixture of bivariate Gaussians is used to predict δ_x and δ_y.
- Bernoulli distribution is used for penup. Hence, each output vector y_t consists of the end of stroke probability along with the parameter for Gaussian mixture components.
- Model input and output:
  - Input $\in \mathbb{R} \times \mathbb{R} \times 0, 1$
  - Output: $y_t = (e_t, \{\pi_t^j, \mu_t^j, \sigma_t^j, \rho_t^j\}_{j=1}^M)$

Output of the model

Output of the model calculated using weight matrices and hidden states:

$$\hat{y}_t = (\hat{e}_t, \{\hat{\pi}_t^j, \hat{\mu}_t^j, \hat{\sigma}_t^j, \hat{\rho}_t^j\}_{j=1}^M) = b_y + \sum_{n=1}^{N} W_{h^n y} h_t^n$$

Calculating parameters of GMM

The parameters of Gaussian mixture models are calculated using the corresponding elements of model output:

$$e_t = \frac{1}{1 + exp(\hat{e}_t)} \implies e_t \in (0, 1)$$

$$\pi_t^j = \frac{exp(\hat{\pi}_t^j)}{\sum_{j'=1}^{M} exp(\hat{\pi}_t^{j'})} \implies \hat{\pi}_t^j \in (0, 1), \sum_j exp(\hat{\pi}_t^j) = 1$$

$$\mu_t^j = \hat{\mu}_t^j \implies \mu_t^j \in \mathbb{R}$$

$$\sigma_t^j = exp(\hat{\sigma}_t^j) \implies \sigma_t^j > 0$$

$$\rho_t^j = \tanh(\hat{\rho}_t^j) \implies \rho_t^j \in (-1, 1)$$

Prediction

The probability density $Pr(x_{t+1}|y_t)$ of the next input x_{t+1} given the output vector y_t is defined as follows:

$$Pr(x_{t+1}|y) = \sum_{j=1}^{M} \pi_t^j \mathcal{N}(x_{t+1}|\mu_t^j, \sigma_t^j, \rho_t^j) \begin{cases} e_t & \text{if } (x_{t+1})_3 = 1 \\ 1 - e_t & \text{otherwise} \end{cases} \tag{6}$$

Synthesis task

- Given text - character sequence Predict (x, y, pen_up), where x and y are offsets from previous inputs
- Now, the predictions have to be conditioned on the text that is provided as an input. For conditioning the predictions on character from text, we must align the predictions and the input text.
- Demo Link

Conditional sythesis model

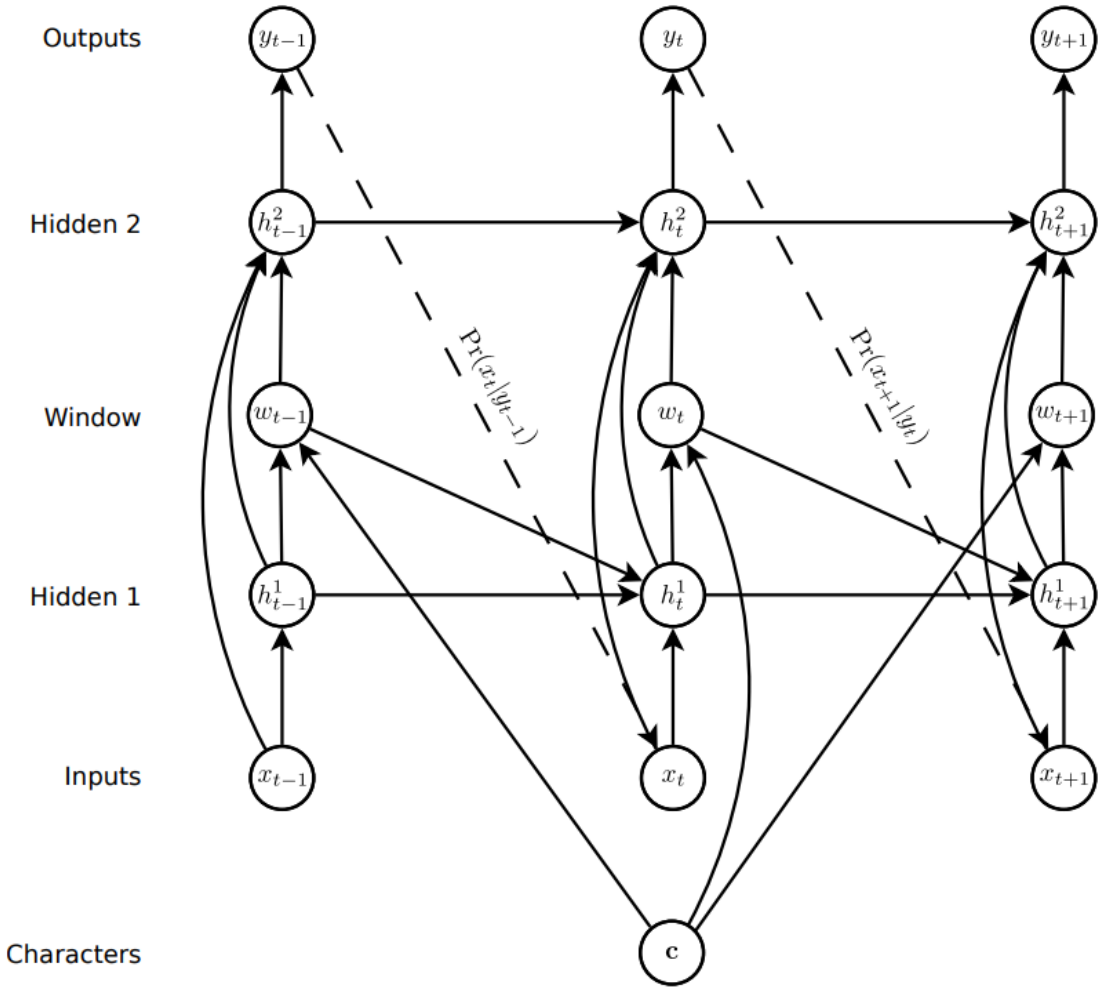


Figure: Model for conditional handwriting generation

Hand writing attention

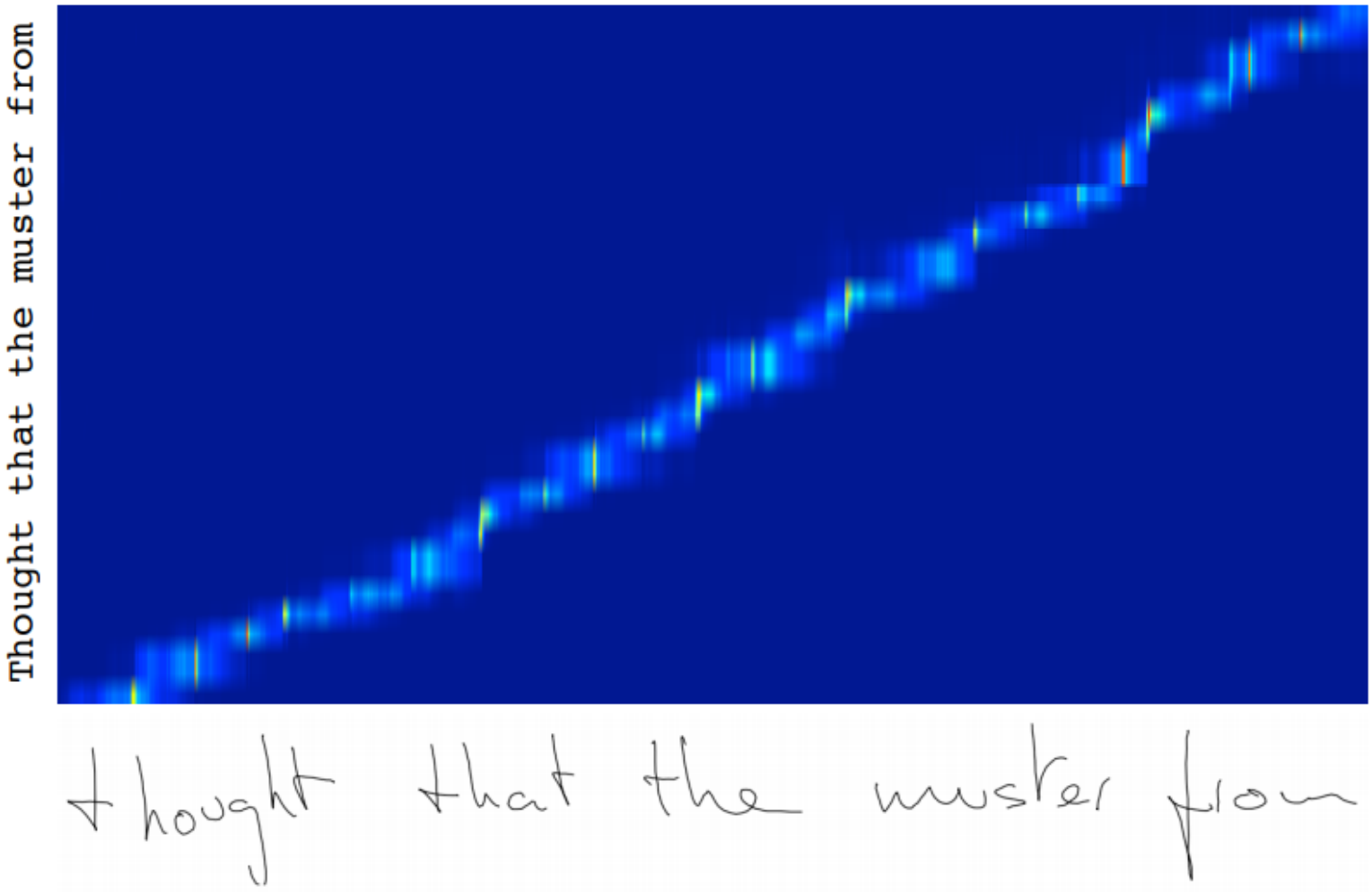


Figure: Attention on characters handwriting generation

Image caption generation

infocusp Innovations

Generated Captions

Captions for image uploaded_file.jpg:
1. a man flying through the air while riding a snowboard .
2. a man flying through the air while riding skis .
3. a man flying through the air on top of a snow board .

Overview of the task

- Complex task - learn not only objects in the image, but also the connection between them and the semantics of natural language.
- Requires visual understanding and also natural language model understanding.
- Previous solutions stitch together sub problems - image to list of words and then formation of sentences.
- CNN acts as the image encoder.

Image caption generation model: show and tell

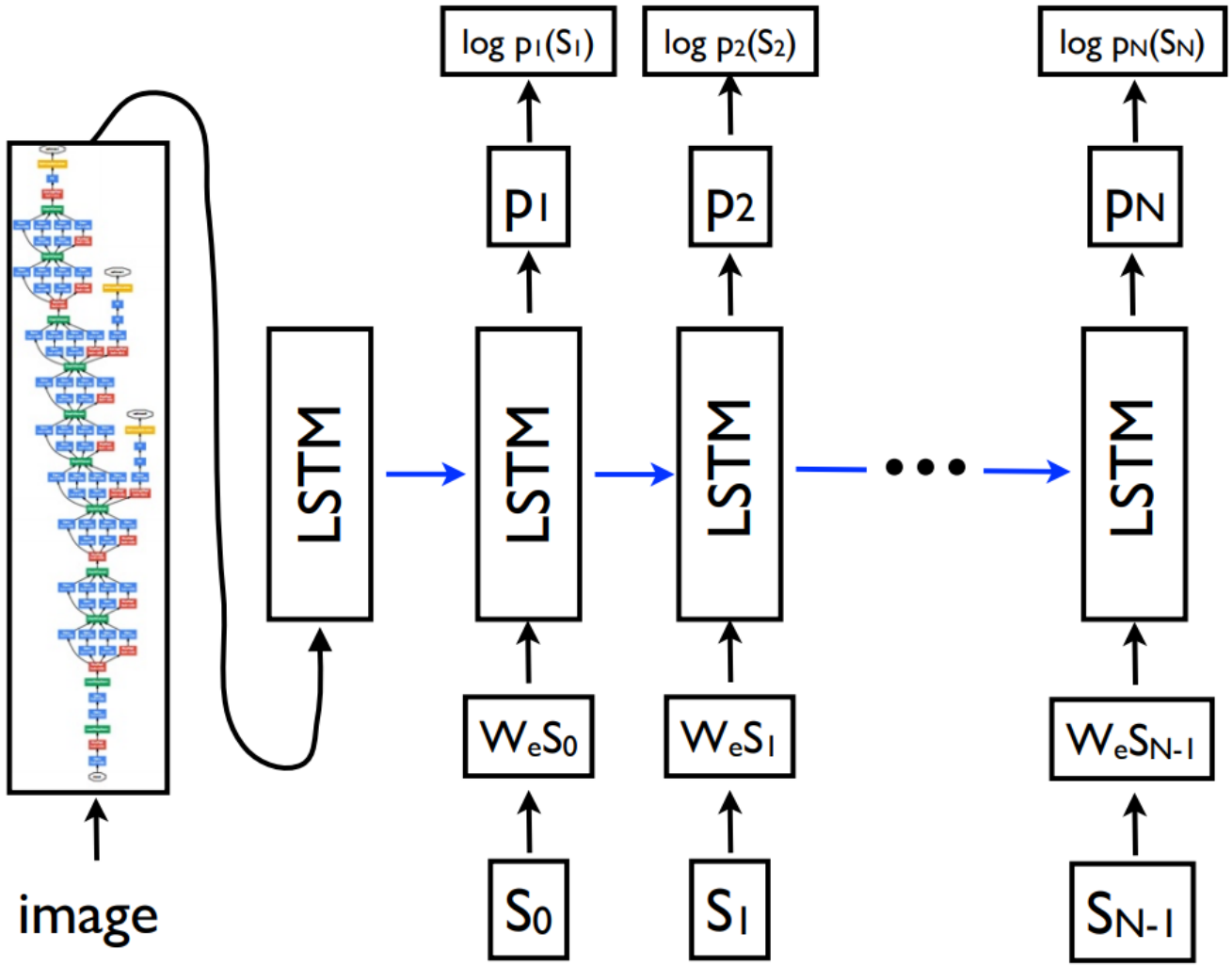


Figure: Image captioning model [10]

[10] https://arxiv.org/abs/1411.4555

Model details

- An “encoder” reads the source sentence and transforms it into a rich fixed-length vector representation, which in turn in used as the initial hidden state of a “decoder” RNN that generates the target sentence.
- Combine deep CNNs for image classification with RNNs for sequence modeling, to create a single network that generates descriptions of images.

Generating captions

- CNN encodes data into a lower dimension and this goes in as first input RNN then generates the word sequence output
- It is natural to model output probabilities with a Recurrent Neural Network (RNN), where the variable number of words we condition upon up to t $-$ 1 is expressed by a fixed length hidden state or memory ht.
- This memory is updated after seeing a new input x_t by using a non-linear function f : $h_t = f(h_{t-1}, x_t)$ – f being an LSTM
- Demo Link

Inference

There are multiple approaches that can be used to generate a sentence given an image, with NIC.

1. Sampling where we just sample the first word according to p_1, then provide the corresponding embedding as input and sample p_2, continuing like this until we sample the special end-of-sentence token or some maximum length.
2. Beam Search: iteratively consider the set of the k best sentences up to time t as candidates to generate sentences of size t + 1, and keep only the resulting best k of them.

Sketch RNN

How to gather a dataset by showcasing a game - draw in less than 20 seconds

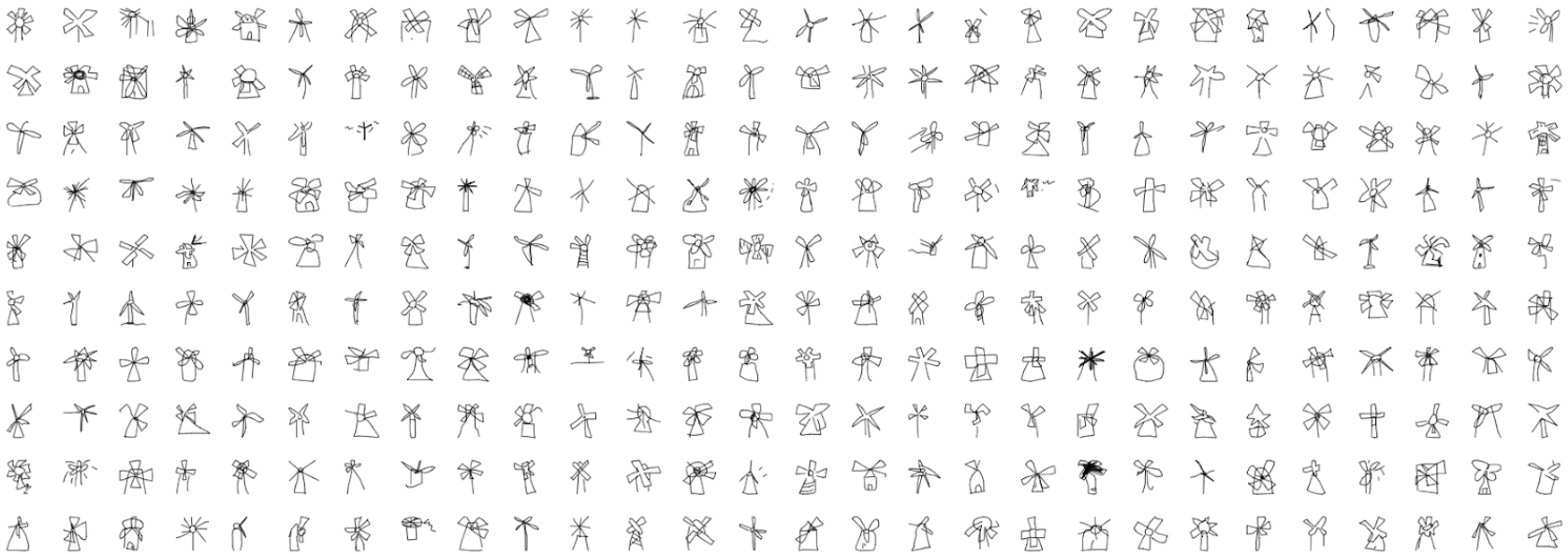


Figure: Sketch RNN dataset samples [11]

[11] https://ai.googleblog.com/2017/04/teaching-machines-to-draw.html

The dataset

Record images which are representation of abstract concepts in people's minds

Figure: Sketch RNN dataset samples [12]

[12] https://quickdraw.withgoogle.com/

The dataset

- Each sketch encoded as a series of pen strokes, which direction to move and when to lift up and when to end
- Dataset with 70k images training data of each class along with 2.5k validation and 2.5k test set images. Format which records pen strokes: (δx, δy, p1 , p2 , p3)
  δx: shift in x
  δy: shift in y
  p1: Pen down
  p2: Pen up
  p3: End of sketch (p1,p2, p3 are one-hot encoded)

The model

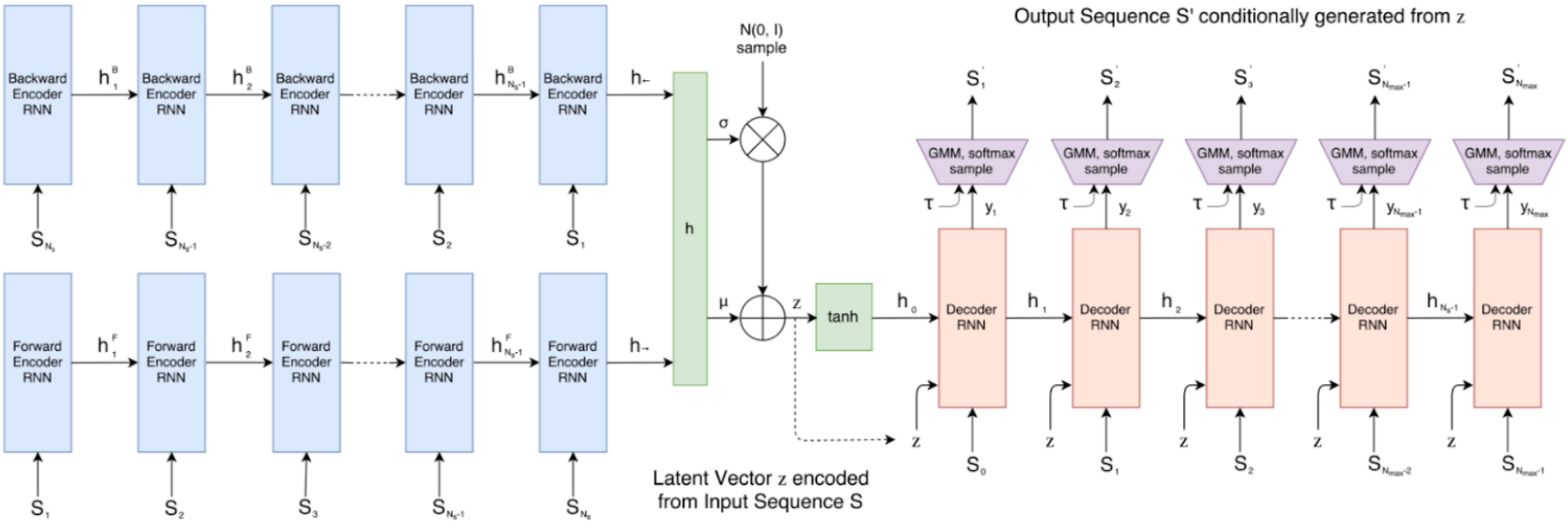


Figure: Sketch RNN model [13]

[13] https://arxiv.org/abs/1411.4555

Sketch RNN description

The vector h is used to generate mean and standard deviation for the distribution from which z is sampled.

$\mu = W_\mu h + b_\mu$

$\hat{\sigma} = W_\sigma h + b_\sigma$

$\sigma = exp(\hat{\sigma}^2)$

$z = \mu + \sigma N(0, I)$

Sketch RNN description

- The stroke sequence is fed in forward and reverse order into forward and reverse encoder RNN respectively which encodes that sequence into vectors $h \rightarrow$ and $h \leftarrow$.
- $h = [h \rightarrow; h \leftarrow]$
- Encoder encodes the complete training sequence into a latent vector z, which is not fixed but rather conditioned on h.
- Decoder is an RNN where we pass in the concatenation of (latent vector z along with the state (or output) vector) at each time step as input and the outputs are the parameters for the next data point.
- Demo Link

About the model

- Initial state S_0 defined as (δx, δy, p1 , p2 , p3) = (0,0,1,0,0) δx, δy are modelled as mixture of M gaussians with parameters : $(\mu_x, \mu_y, \sigma_x, \sigma_y, \rho_{xy})$.
- An additional vector of length M, also a categorical distribution, are the mixture weights of the Gaussian mixture model. Hence the size of the output vector y is 5M + M + 3, which includes the 3 logits needed to generate (q1, q2, q3) which are estimates of (p1, p2, p3).
- Exp on stdev values, tanh on correlation values and softmax on probabilities ensure that outputs are in correct range. One major issue when solving for (q1, q2, q3) is that q1 is much more common compared to q2, q3.

Multiple possible applications

- Use for prediction: Pass in the input strokes into the decoder RNN and allow it to set the state.
- Next, use this state for predictions coming out to complete the sketch
- Interpolation from one to other also possible by interpolating the latent vector.

Summary

- RNNs were introduced to bring more flexibility in the ANN structure and learn more complex problems.
- Looking at the backpropagation in RNN, we saw problems of exploding and vanishing gradients.
- LSTM Network which is a variant of RNN, was introduced for learning longer sequences.
- From the examples and demos, it can be concluded that RNNs and LSTM Networks are powerful architectures that can learn complex sequential tasks.

References

- http://karpathy.github.io/2015/05/21/rnn-effectiveness/
- http://colah.github.io/posts/2015-08-Understanding-LSTMs/
- http://cs231n.github.io/
- Sketch RNN: https://arxiv.org/pdf/1704.03477.pdf
- Generating Sequences With Recurrent Neural Networks: https://arxiv.org/abs/1308.0850
- Show and Tell: A Neural Image Caption Generator: https://arxiv.org/abs/1411.4555
- Min char RNN: https://gist.github.com/karpathy/d4dee566867f8291f086


Thank
You