Machine Learning 10-601

Tom M. Mitchell
Machine Learning Department
Carnegie Mellon University

October 4, 2017

Today:
- Cross Entropy
- Recurrent networks
- Sequential models

Reading:
- Goodfellow: Sequence Modeling: Recurrent Nets

What is cross entropy?

- Our negative log likelihood loss is also called cross entropy. Why?

$$J(\theta) = \sum_{\langle \mathbf{x}, y\rangle \in D} -\log P(Y = y | X = \mathbf{x})$$

- We can view the set of training examples D as representing an "empirical" probability distribution $P_D(X,Y)$ where

$$P_D(X = x, Y = y) = \frac{1}{|D|} \sum_{\langle x, y\rangle \in D} \delta(X = x, Y = y)$$

- and then write negative log conditional likelihood

$$J(\theta) = \sum_{\text{every } \langle x,y\rangle} P_D(X = x, Y = x)(-\log P(Y = y | X = x))$$

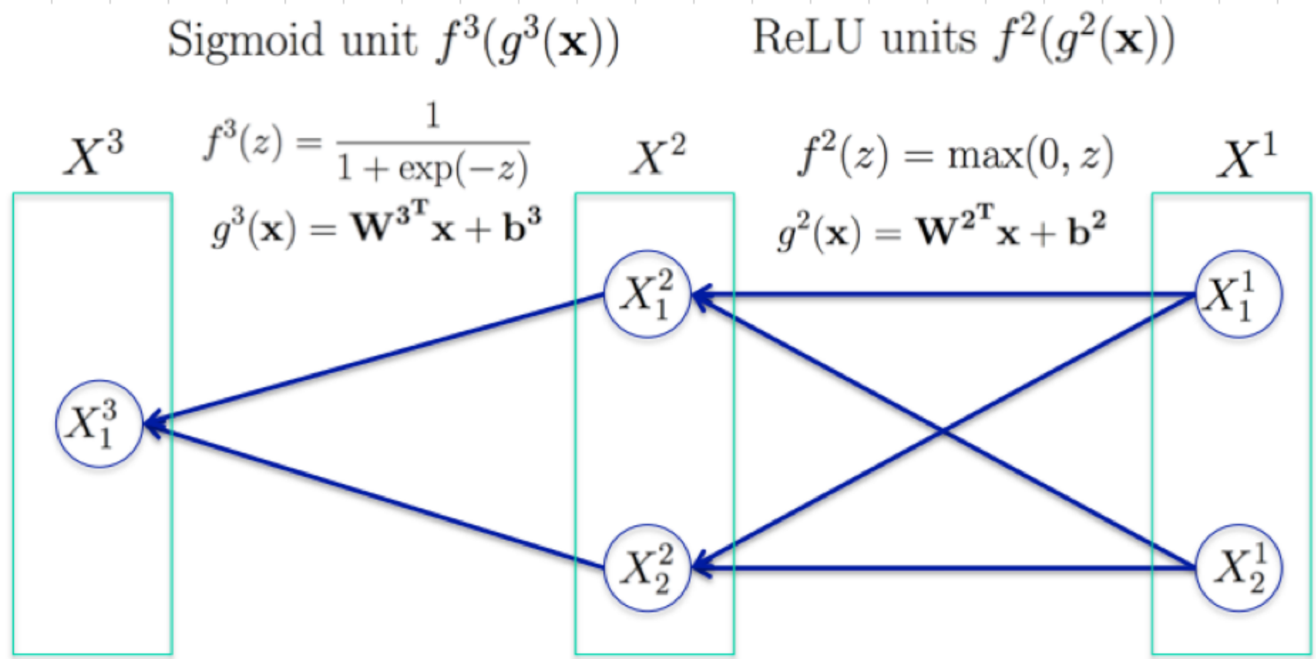
$$= E_{P_D(X,Y)}[-\log P(Y = y | X = x)]$$

- which is often called "<u>cross entropy</u>" because conditional entropy H(Y|X)

$$H(Y|X) = \sum_{\text{every } \langle x,y\rangle} P(X = x, Y = x)(-\log P(Y = y | X = x))$$

$$= E_{P(X,Y)}[-\log P(Y = y | X = x)]$$

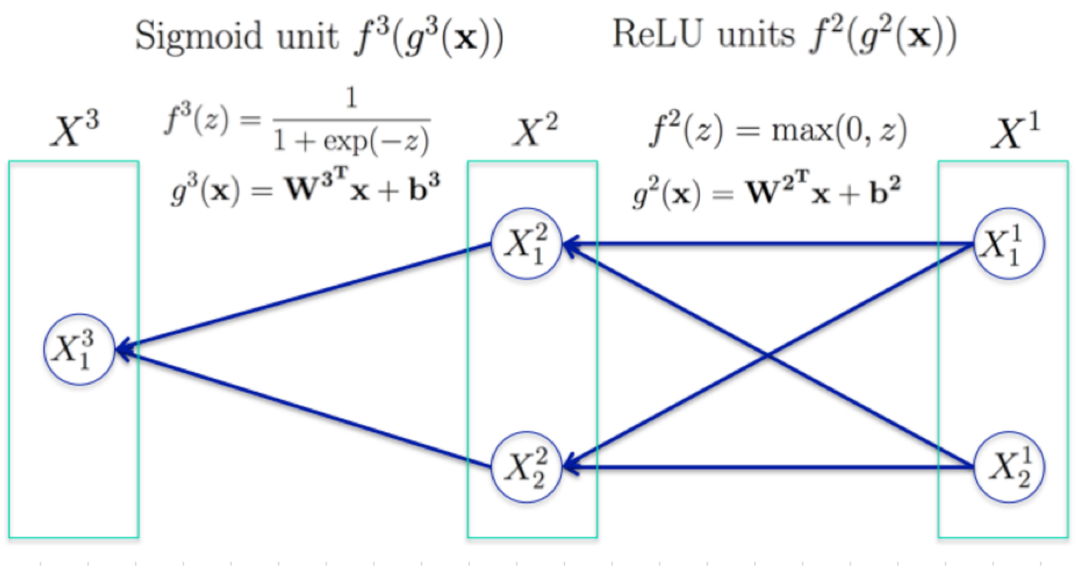
Back propagation



Ytrue	1		
dJ/dX3	-1.89		
dJ/dg3	-0.47		
dJ/dW3T	-0.09	0.00	-0.47
dJ/dX2	-0.05		
	0.04		
dJ/dg2	-0.05		
	0.00		
dJ/dW2T	-0.05	0.00	-0.05
	0.00	0.00	0.00

X^3	$g^3(X^2)$	W^{3T}		b^3	X^2	$g^2(X^1)$	W^{2T}		b^2	X^1
0.53	0.12	0.10	-0.09	0.10	0.20	0.20	0.10	-0.10	0.10	1
					0.00	-0.15	-0.20	0.10	0.05	0
					1					*1*

$$\boxed{\text{update each parameter according to } \theta_i \leftarrow \quad \theta_i - \eta \frac{\partial J(\theta)}{\partial \theta_i}}$$



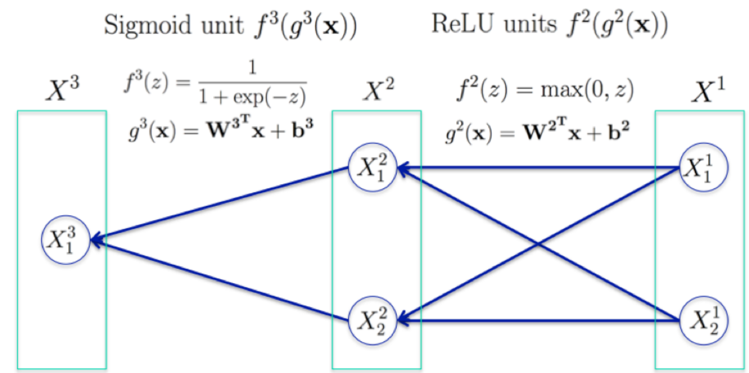
Given boolean Y, X_1, X_2 learn P(Y|X_1,X_2), where

$$P(Y = 0 | X_1 = X_2) = 0.9$$
$$P(Y = 1 | X_1 \neq X_2) = 0.9$$

Training:

- stochastic gradient descent
- minibatch size 4
- 20,000 iterations
- no momentum, regulariation, ...

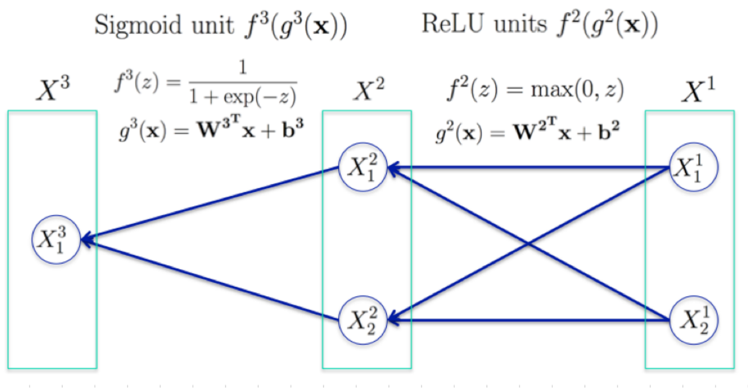
Learned P(Y=1|X1, X2)



Input: [0 1]	[1 1]	[1 0]	[0 0]
[0.5324]	[0.5343]	[0.5390]	[0.5348]
[0.5139]	[0.5145]	[0.5230]	[0.5149]
[0.5045]	[0.5048]	[0.5203]	[0.5046]
[0.5076]	[0.5073]	[0.5393]	[0.5075]
[0.4988]	[0.4976]	[0.5577]	[0.4987]
[0.4913]	[0.4883]	[0.5971]	[0.4895]
[0.4950]	[0.4877]	[0.6723]	[0.4898]
[0.4655]	[0.4541]	[0.7168]	[0.4553]
[0.4677]	[0.4527]	[0.7778]	[0.4422]
[0.4748]	[0.4093]	[0.8068]	[0.4093]
[0.5119]	[0.3754]	[0.8337]	[0.3752]
[0.5928]	[0.3440]	[0.8603]	[0.3450]
[0.6882]	[0.3108]	[0.8783]	[0.3083]
[0.7903]	[0.2789]	[0.8856]	[0.2789]
[0.8103]	[0.2315]	[0.8794]	[0.2315]
[0.8509]	[0.2062]	[0.8807]	[0.2062]
[0.8634]	[0.1860]	[0.8938]	[0.1860]
[0.8676]	[0.1626]	[0.8915]	[0.1626]
[0.8919]	[0.1511]	[0.8965]	[0.1511]
[0.8821]	[0.1392]	[0.8850]	[0.1392]
[0.8769]	[0.1294]	[0.9053]	[0.1292]

Training iterations

Learned representation for X^2

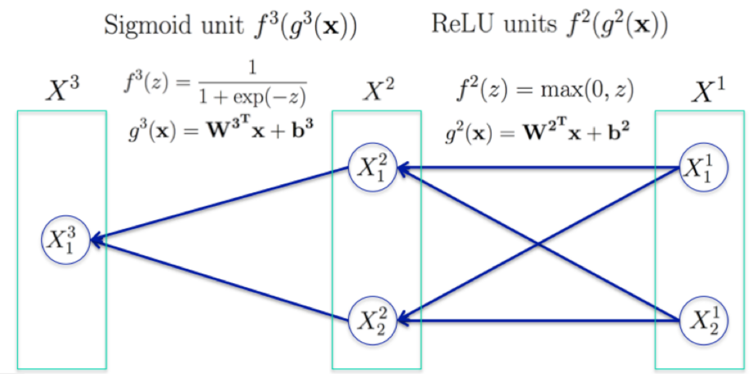


Input: [0 1]   [1 1]   [1 0]   [0 0]

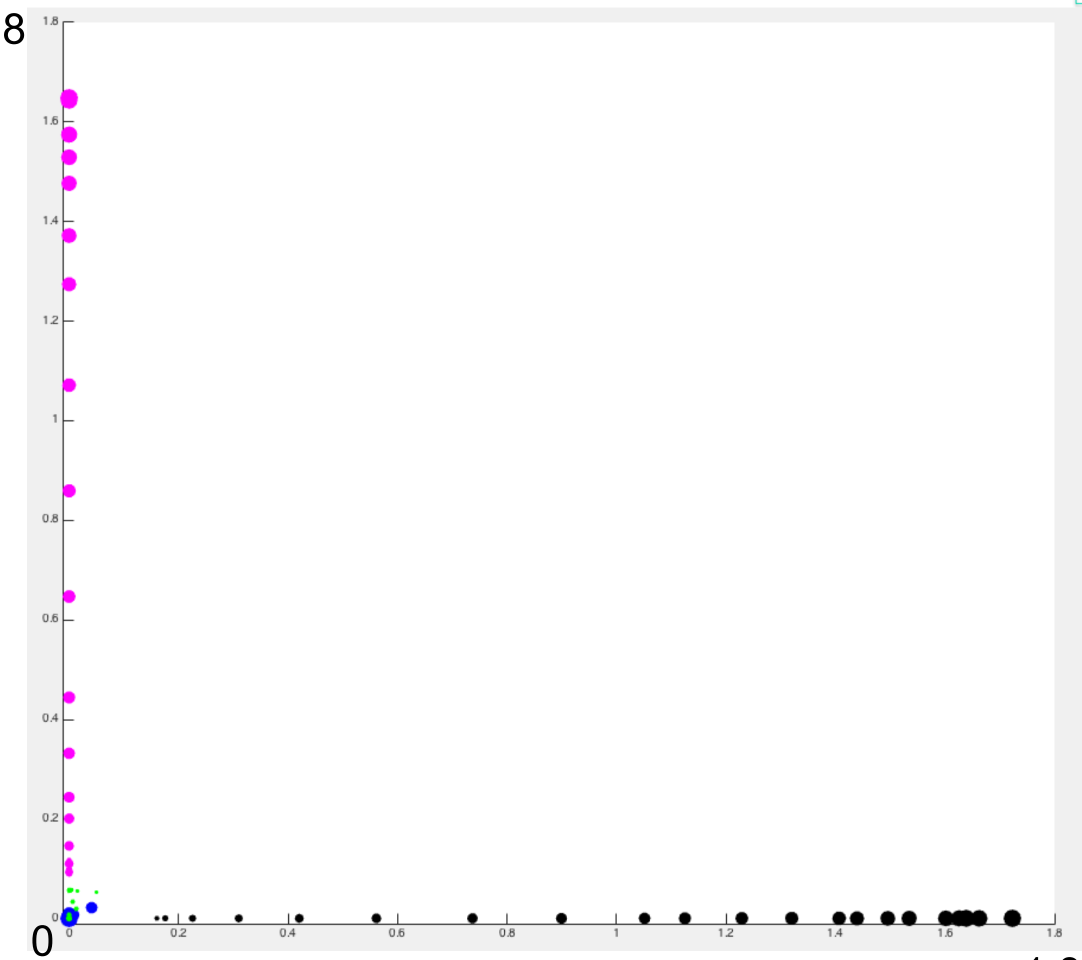
Training iterations

Learned representation for X^2

Input: [0 1] [1 1] [1 0] [0 0]



1.8

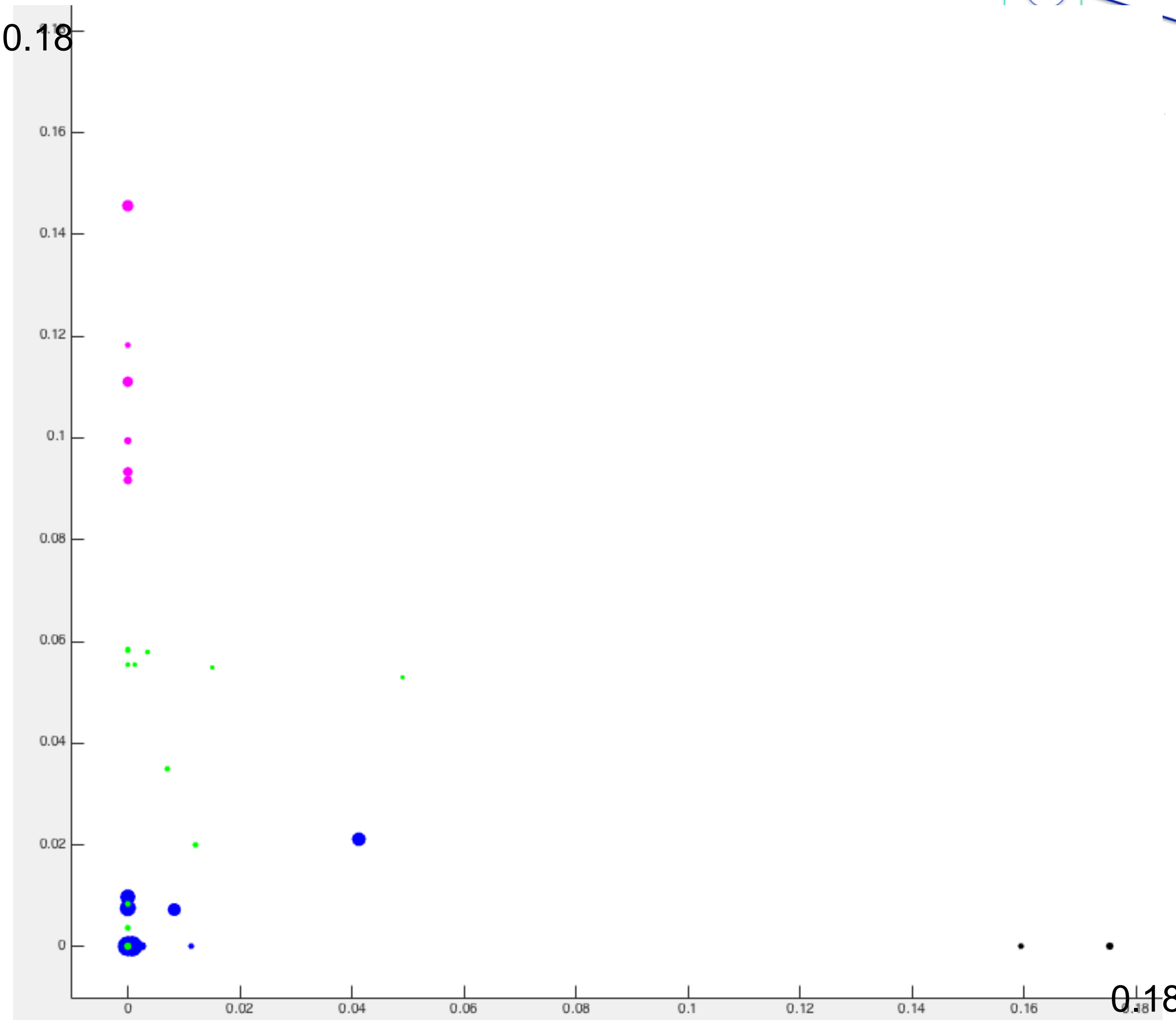


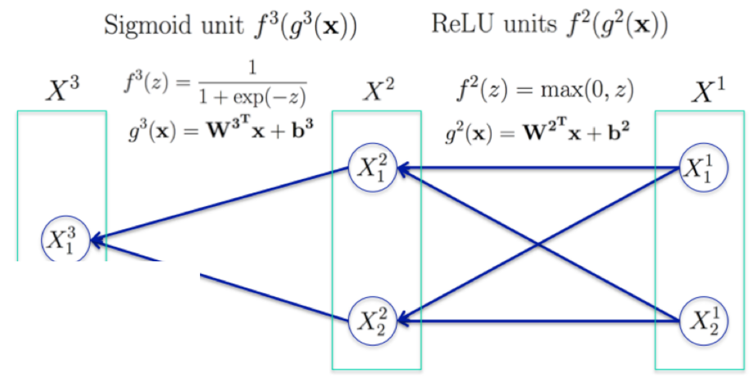
0

1.8

Learned representation for X^2

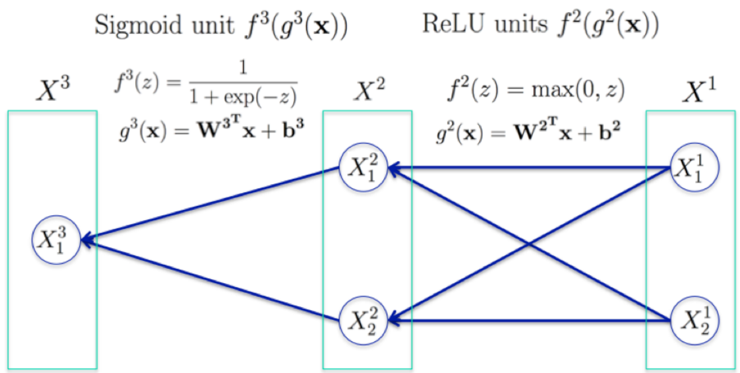
Input: [0 1] [1 1] [1 0] [0 0]

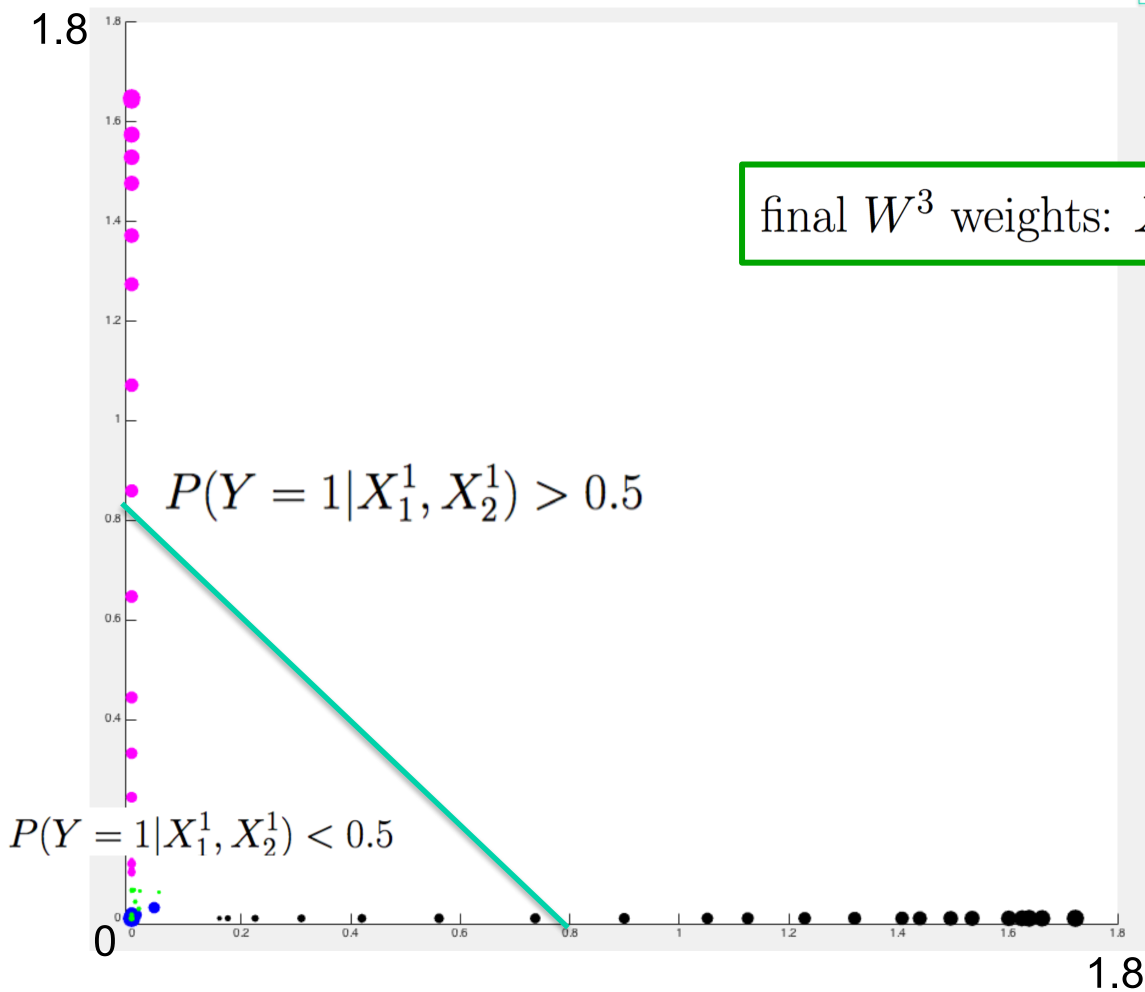




Final decision surface in terms of X^2

Input: [0 1] [1 1] [1 0] [0 0]





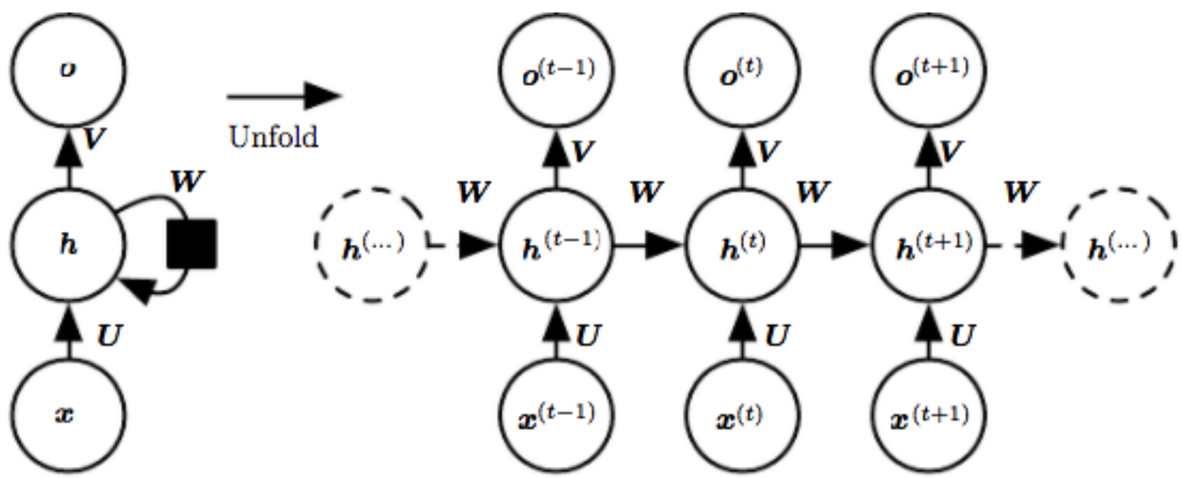
final W^3 weights: $X^3 = 2.417\ X_1^2 + 2.346\ X_2^2 - 1.908$

Sequential Neural Nets

Recurrent Networks

- Many tasks involve sequential data
  - predict stock price at time t+1 based on prices at t, t-1, t-2, …
  - translate sentences (word sequences) from Spanish to English
  - transcribe speech (sound sequences) to text (word sequences)

- Key idea: recurrent network uses (part of) its state at t as input for t+1

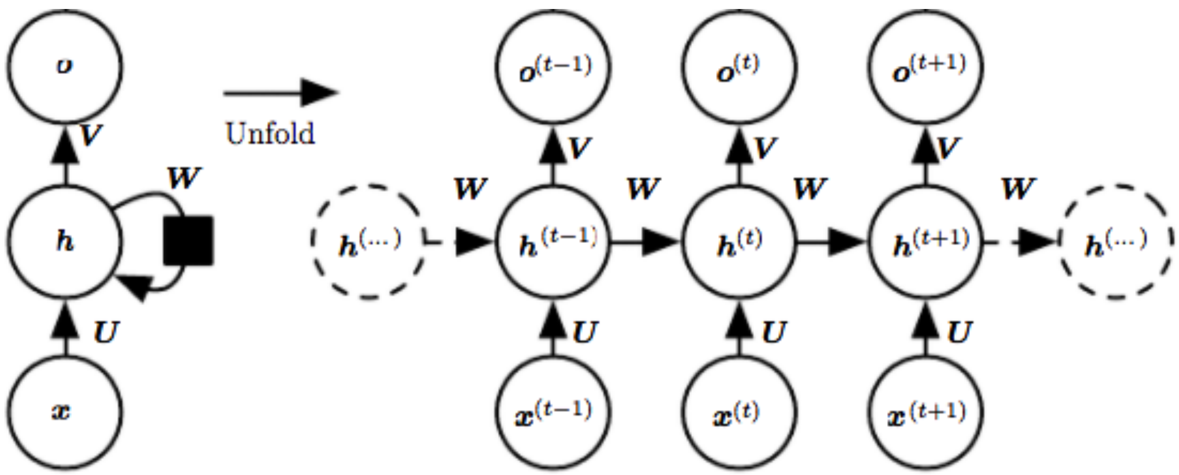


[Goodfellow et al., 2016]

Training Recurrent Networks

Key principle for training:

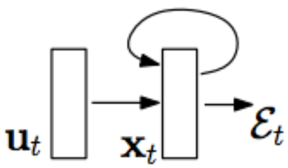
1. Treat as if unfolded in time, resulting in directed acyclic graph
2. Note shared parameters in unfolded net → sum the gradients



* problem: vanishing and/or exploding gradients

[Goodfellow et al., 2016]

Gradient Clipping: Managing exploding gradients



$$\mathbf{x}_t = \mathbf{W}_{rec}\sigma(\mathbf{x}_{t-1}) + \mathbf{W}_{in}\mathbf{u}_t + \mathbf{b}$$

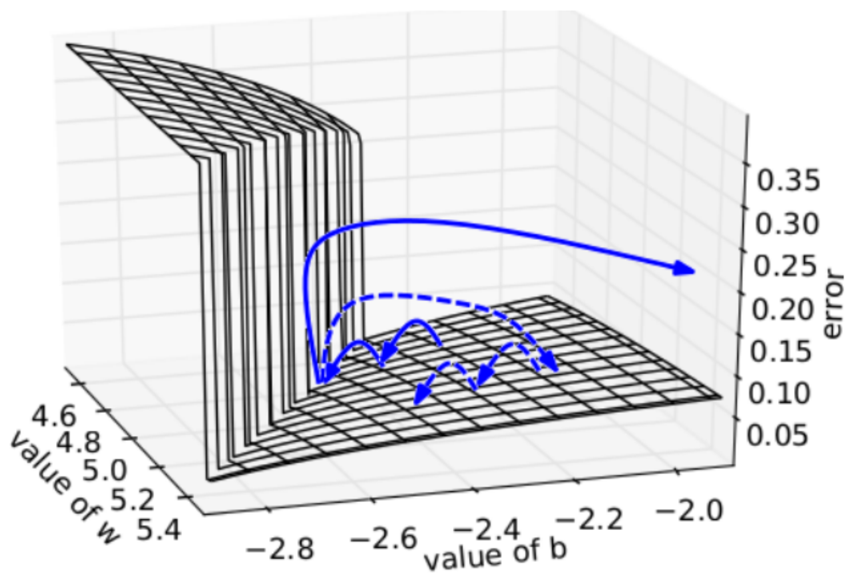


Figure 6. We plot the error surface of a single hidden unit recurrent network, highlighting the existence of high curvature walls. The solid lines depicts standard trajectories that gradient descent might follow. Using dashed arrow the diagram shows what would happen if the gradients is rescaled to a fixed size when its norm is above a threshold.

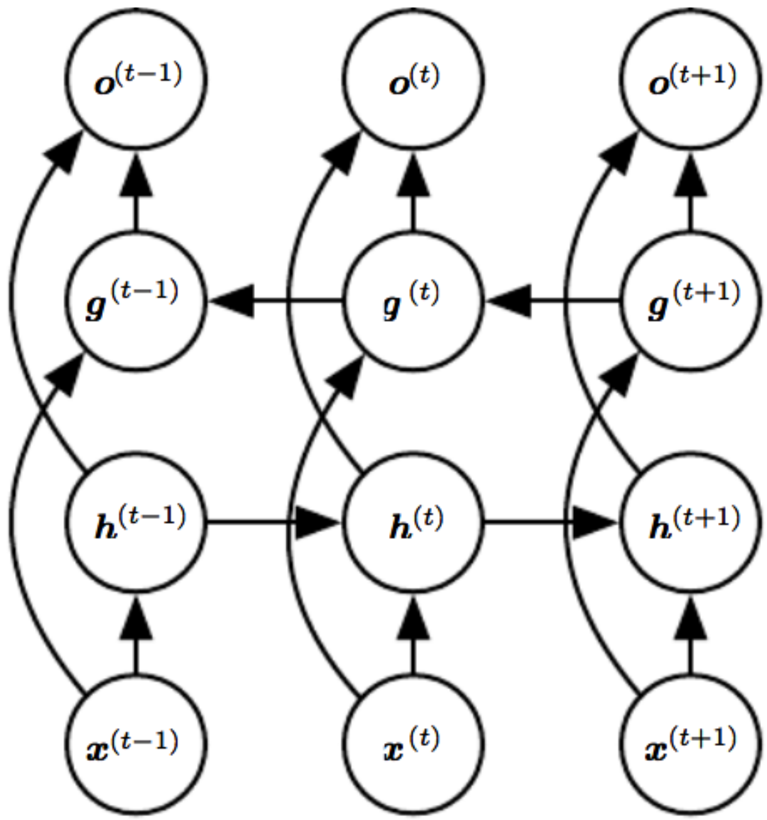
Algorithm 1 Pseudo-code for norm clipping dients whenever they explode

$\hat{\mathbf{g}} \leftarrow \frac{\partial \mathcal{E}}{\partial \theta}$
if $\|\hat{\mathbf{g}}\| \geq threshold$ **then**
  $\hat{\mathbf{g}} \leftarrow \frac{threshold}{\|\hat{\mathbf{g}}\|}\hat{\mathbf{g}}$
end if

[Pascanu et al., 2013]

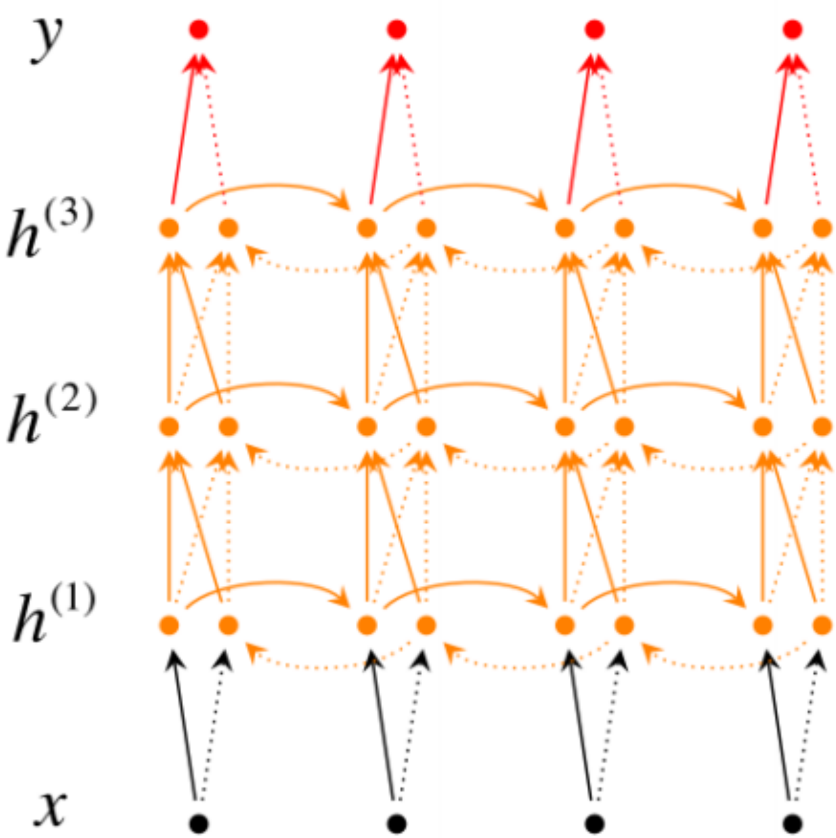
Bi-directional Recurrent Neural Networks

- Key idea: processing of word at position t can depend on following words too, not just preceding words



[Goodfellow et al., 2016]

Deep Bidirectional Recurrent Network



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Each bidirectional layer builds on the one below

[Irsoy & Cardie, 2014]

Deep Bidirectional Recurrent Network: Opinion Mining

(4)

	[In any case] , [it is high time] that a social debate be organized ...
DEEPRNN	[In any case] , it is HIGH TIME that a social debate be organized ...
SHALLOW	In ANY case , it is high TIME that a social debate be organized ...

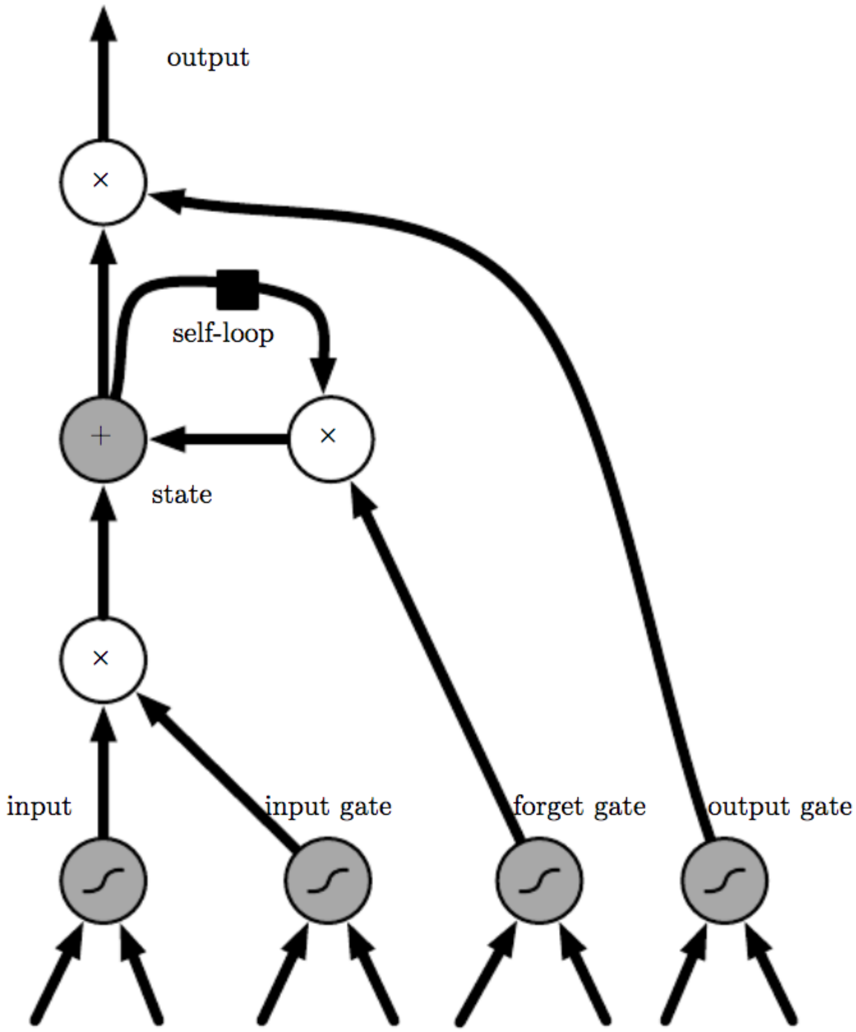
(5)

	Mr. Stoiber [has come a long way] from his refusal to [sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances] run against the wishes...
DEEPRNN	Mr. Stoiber [has come a long way from] his [refusal to sacrifice himself] for the CDU in an election that [once looked impossible to win] , through his statement that he would [under no circumstances run against] the wishes...
SHALLOW	Mr. Stoiber has come A LONG WAY FROM his refusal to sacrifice himself for the CDU in an election that [once looked impossible] to win , through his statement that he would under NO CIRCUMSTANCES run against the wishes...

Figure 3: DEEPRNN Output vs. SHALLOWRNN Output. In each set of examples, the gold-standard annotations are shown in the first line. Tokens assigned a label of Inside with no preceding Begin tag are shown in ALL CAPS.

[Irsoy & Cardie, 2014]

Long Short Term Memory (LSTM) unit



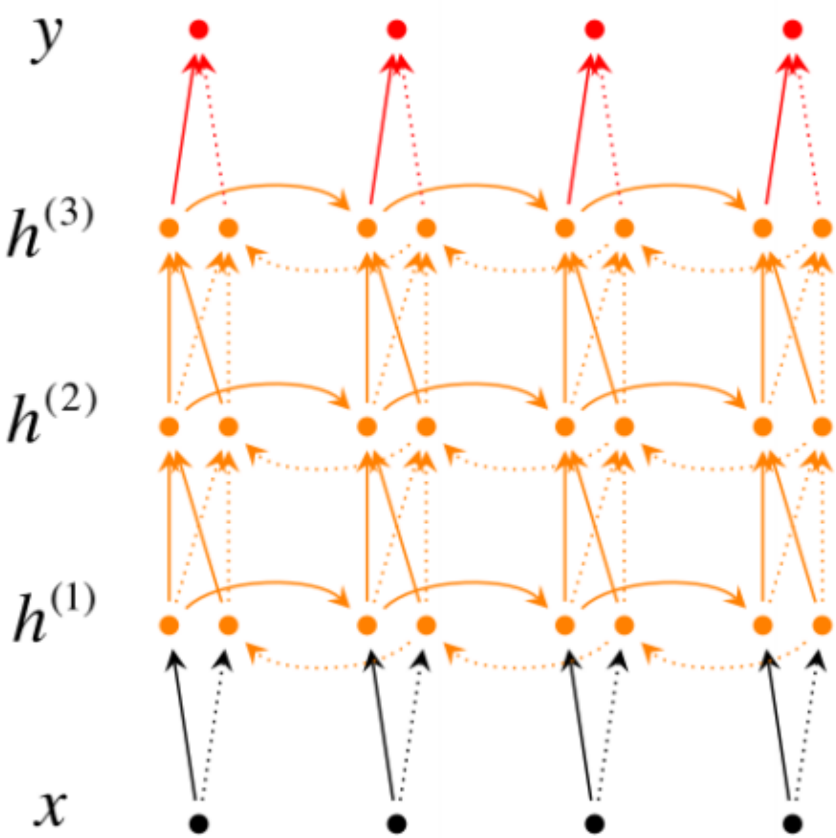
input gate: if 1, add input to memory state

forget gate: if 0, zero out memory state, else retain (partially)

output gate: if 1, read memory to output

[Goodfellow et al., 2016]

Deep Bidirectional Recurrent Network



$$\overrightarrow{h}_t^{(i)} = f(\overrightarrow{W}^{(i)} h_t^{(i-1)} + \overrightarrow{V}^{(i)} \overrightarrow{h}_{t-1}^{(i)} + \overrightarrow{b}^{(i)})$$

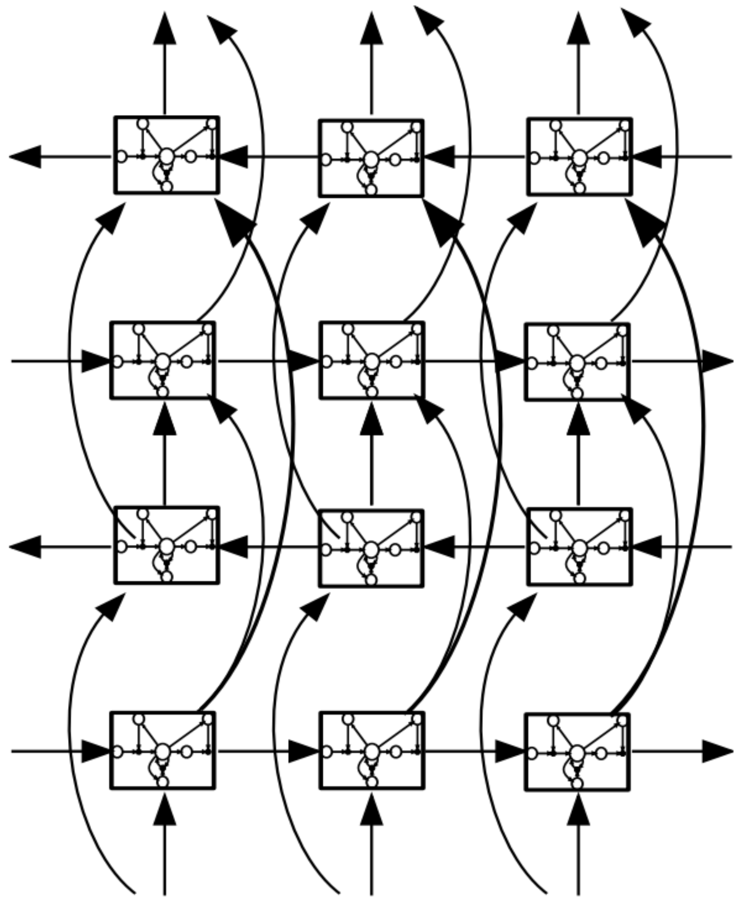
$$\overleftarrow{h}_t^{(i)} = f(\overleftarrow{W}^{(i)} h_t^{(i-1)} + \overleftarrow{V}^{(i)} \overleftarrow{h}_{t+1}^{(i)} + \overleftarrow{b}^{(i)})$$

$$y_t = g(U[\overrightarrow{h}_t^{(L)}; \overleftarrow{h}_t^{(L)}] + c)$$

Each bidirectional layer builds on the one below

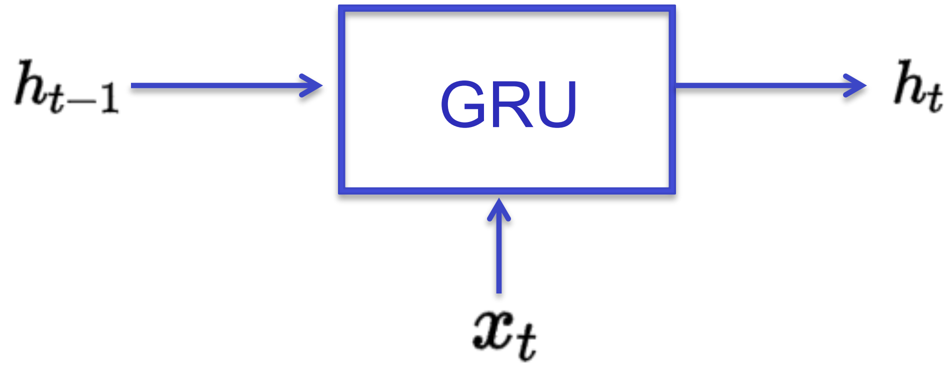
[Irsoy & Cardie, 2014]

Deep Bidirectional LSTM Network



["Hybrid Speech Recognition with Deep Bidirectional LSTM,"
Graves et al., 2013]

Gated Recurrent Units (GRUs)



$\circ$ denotes the Hadamard product. $h_0 = 0$.

$$z_t = \sigma_g(W_z x_t + U_z h_{t-1} + b_z)$$
$$r_t = \sigma_g(W_r x_t + U_r h_{t-1} + b_r)$$
$$h_t = z_t \circ h_{t-1} + (1 - z_t) \circ \sigma_h(W_h x_t + U_h(r_t \circ h_{t-1}) + b_h)$$

Variables

- x_t: input vector
- h_t: output vector
- z_t: update gate vector
- r_t: reset gate vector
- W, U and b: parameter matrices and vector

Activation functions

- σ_g: The original is a sigmoid function.
- σ_h: The original is a hyperbolic tangent.

fewer parameters than LSTM
found equally effective for
- speech recognition
- music analysis

see [Chung et al., 2014]

Programming Frameworks for Deep Nets

- TensorFlow (Google)
- TFLearn (runs on top of TensorFlow, but simpler to use)
- Theano (University of Montreal)
- Pytorch (Facebook)
- CNTK (Microsoft)
- Keras (can run on top of Theano, CNTK, TensorFlow)

Many support use of Graphics Processing Units (GPU's)

Major factor in dissemination of Deep Network technology

```python
# Specify that all features have real-value data
feature_columns = [tf.feature_column.numeric_column("x", shape=[4])]

# Build 3 layer DNN with 10, 20, 10 units respectively.
classifier = tf.estimator.DNNClassifier(feature_columns=feature_columns,
                                        hidden_units=[10, 20, 10],
                                        n_classes=3,
                                        model_dir="/tmp/iris_model")
# Define the training inputs
train_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": np.array(training_set.data)},
    y=np.array(training_set.target),
    num_epochs=None,
    shuffle=True)

# Train model.
classifier.train(input_fn=train_input_fn, steps=2000)

# Define the test inputs
test_input_fn = tf.estimator.inputs.numpy_input_fn(
    x={"x": np.array(test_set.data)},
    y=np.array(test_set.target),
    num_epochs=1,
    shuffle=False)

# Evaluate accuracy.
accuracy_score = classifier.evaluate(input_fn=test_input_fn)["accuracy"]

print("\nTest Accuracy: {0:f}\n".format(accuracy_score))
```

TensorFlow example

Modern Deep Networks: 2017 vs 1987

- vastly more online data
- GPU's, TPU's
- homogenous units
  - Relu, sigmoid, tanh, linear
- including memory units
  - LSTM, GRU, …
- wild new architectures
  - 100 layers deep, bidirectional LSTMs, Convolutional nets widespread ...
- new ideas for gradient descent
  - dropout, batch normalization, weight initialization, ...
- unification with probabilistic models
  - train to output probabilities
- frameworks like TensorFlow

What you should know:

- Representation learning in neural networks
  - hidden layers re-represent inputs to predict outputs
  - auto-encoders
  - word embeddings
- Sequential models
  - recurrent networks, and unfolding them
  - memory units: LSTM, etc.
  - deep sequential neural networks
- Pragmatics of training deep nets
  - vanishing gradients and exploding gradients
  - gradient clipping, batch normalization, …
  - frameworks such as TensorFlow, Pytorch, ...