Recurrent Neural Networks Tutorial **(DRAFT)**

Mark A. Austin

University of Maryland

austin@umd.edu
ENCE 688P, Spring Semester 2022

December 9, 2022

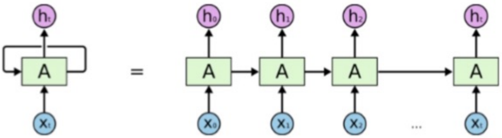
Overview

1. Standard Recurrent Neural Networks (RNN)
   - Learning Sequences
   - Single- and Multi-Layer RNN Architectures
2. Long Short-Term Memory (1997)
3. Gated Recurrent Units (2014)
4. Examples
   - Memorize pi to 20 decimal places (3.14159265358979323846)
   - Learning a Time Series (Weather Conditions in Seattle)
   - Generating New Text (Learn Shakespeare, Write Shakespeare)
   - LSTM Model to Identify Trends in Chart Sequences

Quick Review

Machine Learning Capabilities (1997-2014)

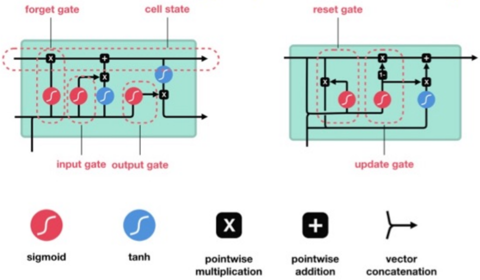
Recurrent Neural Networks (RNN): Learn sequences in data streams (text, speech)



Hidden state "h" serves two purposes:

- Make an output prediction.
- Represent features in the previous steps ....

Long Short-Term Memory (1997) Gated Recurrent Units (2014)



Key Features of LSTM:

- Standard RNN suffers from vanishing gradients for modeling of long-term dependencies.
- LSTM gives cells the ability to remember values for long periods of time.
- Gates regulate the flow of information in / out of the cell, and what should be remembered or discarded.

Applications:

- Time series prediction.
- Time-series anomaly detection.

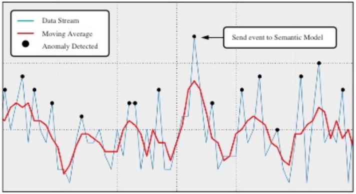
Machine Learning Capabilities (1997-2014)

Learning Streams of Text

- Download complete works of Shakespeare (5.4 million characters)
- Train machine to remember text.
- Write new Shakespeare!

```
Sample after 1000 Iterations
----------------------------

  OTHERS. Allow up, whe, Will a moutice,
  Reason thot boy Eckents alliem; comn jusgius,
  And encarnembover hath last)!
  TASTOR. When no; devires at though me beligo jody?
  MANCELOT. Trom juther and bur itnot spock,
  That as take have wendisho

Sample after 2000 Iterations
----------------------------

  VIRGTARD. Walk this boy and door as am 'stone!
  NTRURLEN. Being entainure af Haton.
  His that by much that, I mion's now, who make foll the kidl!
  CLOTE. Why which Hamm'd?
  QUICKLY. And stand quast of I; this Pi

Sample after 2500 Iterations
----------------------------

  FORD. Nay, You're, excount: and now did yet.
  PAROLLES. Take DUTBY
  This who is begin Gnoban a bows; but yet which that have be,
  Oll, thou stam, and me not ready withered gids
  And he in the pleasues or pardon us.
  Mer. I pray you, how can, and tu

Sample after 3000 Iterations
----------------------------

  Hume, 'tigoning, dear?
  My lam, an hour, chork'd more in my grain,
  I am thing forent innonion, madam! I thank I do speak you?
  ABESSITGBO. But his bosines, giving to know; foward to
  the distvesail.' The to you well know yes, my lovi.
  SECOND CMONBOR. He needs, for the reforeds are;
```

Time Series Anomaly Detection



Time Series Prediction

Standard Recurrent Neural Networks

Sequence Modeling: Design Criteria

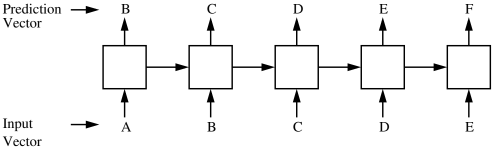
Learning Sequences

To model sequences we need to:

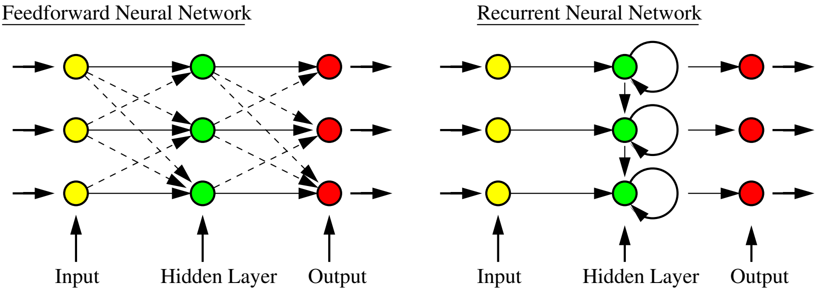
- Handle variable-length sequences
- Maintain information about order
- Share parameters across the sequence
- Track long-term dependencies

Simple Example:

The sequence A $\rightarrow$ B $\rightarrow$ C $\rightarrow$ D $\rightarrow$ E $\rightarrow$ F can be modeled:



Feedforward Neural Networks vs RNNs

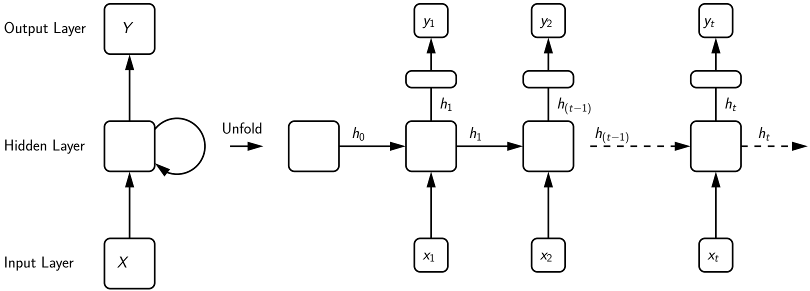


Key Points:

- Feedforward neural networks have no memory.
- Standard RNN's have short-term memory.

Standard RNN Model

RNN Model with One Hidden Layer ...



Hidden states:

- Represent features in the previous steps.
- Make an output prediction.

Standard RNN Model

Update in Activation

$$h_t = g_1(W_{aa}h_{(t-1)} + W_{ax}x_t + b_a). \tag{1}$$

Update in Output

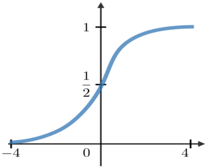
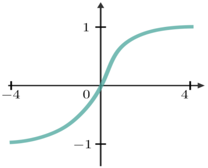
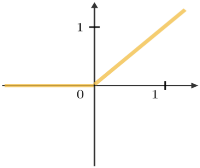
$$y_t = g_2(W_{ya}h_t + b_y) \tag{2}$$

Here:

- h_t is output from state cell at time t.
- W_{aa}, W_{ax}, W_{ya}, b_a, and b_y are coefficients that are shared temporally.
- $g_1()$ and $g_2()$ are activation functions.

Standard RNN Model

Commonly Used Activation Functions

Sigmoid	Tanh	RELU
$g(z) = \dfrac{1}{1+e^{-z}}$	$g(z) = \dfrac{e^{z}-e^{-z}}{e^{z}+e^{-z}}$	$g(z) = \max(0,z)$
		

Note. The $\sigma()$ and tanh() activations squish values to always be in the $[-1,1]$ range.

Standard RNN Model: Training Procedure

Loss Function: Loss L is the sum of loss functions evaluated at preceeding time steps, i.e.,

$$L_n(\hat{y}, y) = \sum_{t=1}^{n} L(\hat{y}_t, y_t). \tag{3}$$

Training Procedure: Backpropogation is done at each point in time.

At timestep t, the derivative of the loss L with respect to the weight matrices W can be written:

$$\frac{\partial L}{\partial W} = \sum_{i=1}^{T} \frac{\partial L_i}{\partial W} \tag{4}$$

Standard RNN Model: Training Procedure

Gradient Update Rule

- Gradients are used to update the weights in a neural network,

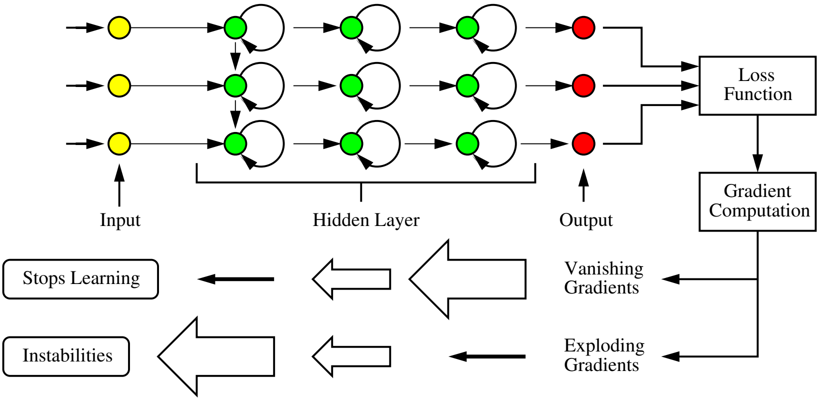
$$\text{New Weight} = \text{Weight} - \text{Learning Rate} * \text{Gradient}. \tag{5}$$

Vanishing and Exploding Gradients

- Layers that get very small gradient updates stop learning. And because this usually happens in the early layers, layers beyond that don't learn.
- Exploding gradients lead to oscillations – instability – in learning of the weights.

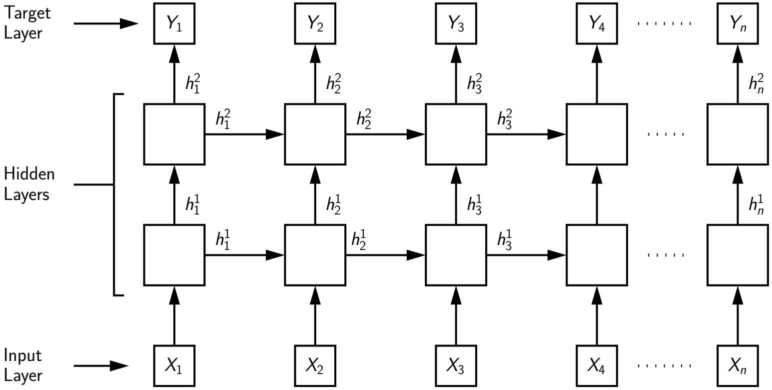
Standard RNN Model: Training Procedure

Challenge: Vanishing and Exploding Gradients ...

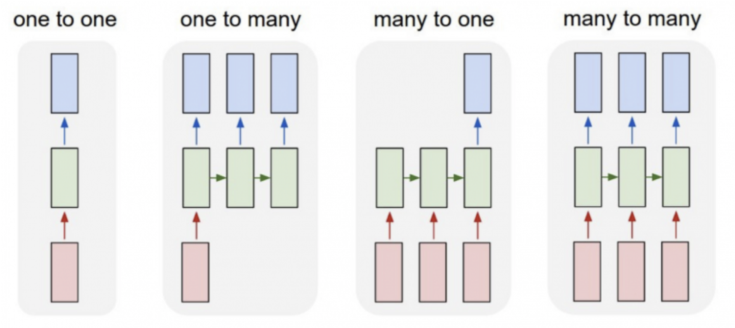


RNN Model Extesions

RNN Model with Multiple Layers



RNN Model Extesions and Applications



One-to-One: Standard neural network.

One-to-Many: Music and text generation.

Many-to-One: Sentiment classification

Many-to-Many: Name-entity recognition, machine translation.

Standard RNN Model: Summary

Advantages:

- Model can process input of any length.
- Model size does not increase with length of input.
- Model takes into account historical information.

Weaknesses:

- Computation time can be slow.
- Accessing information from a long time ago can be difficult.
- Cannot look ahead and consider future input.

Long Short-Term Memory

(Under the Hood)

Long Short-Term Memory (LSTM)

Motivation

- Standard RNNs forget what is seen in longer sequences – hence, the term short-term memory.

Key Features

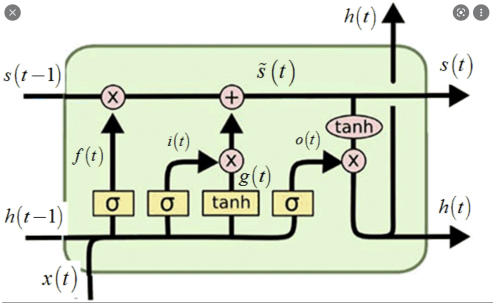
- LSTM gives cells the ability to remember values for long periods of time.

Approach

- Cell states act as a communications highway that can transfer information all the way down the sequence chain. This is long-term memory. Hidden states serve as short-term memory.
- Gates regulate the flow of information in/out of the cell, as well as what should be remembered or discarded.

Long Short-Term Memory (LSTM)

LSTM Architecture

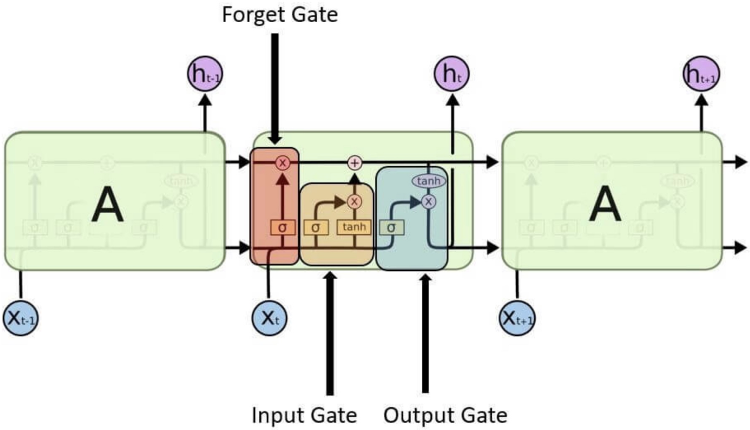


Here:

- $s_{(t)}$ is the cell state passed forward to the next block.
- $h_{(t)}$ is output passed forward to the next block.
- $x_{(t)}$ is block input.

LSTM Core Concepts

LSTM Core Concepts: Cell state information. Forget, input, and output gate operations.



LSTM Core Concepts

Forget Gate Operation: $f_{(t)} \in (0, 1)$.

- Forget gates pass information from previous state and current input is to a sigmoid function,

$$f_{(t)} = \sigma(W_f \cdot [h_{(t-1)}, x_{(t)}] + b_f). \tag{6}$$

Points to note:

- Forget gates decide what information should be discarded or kept?
- Output values close to 0 mean discard; output values close to 1 mean keep.
- Here, W_f = weight matrix between forget gate and input gate. b_f = connection bias at timestep t.

LSTM Core Concepts

Input Gate Operation: $i_{(t)} \in (0,1)$.

$$i_{(t)} = \sigma(W_i \cdot [h_{(t-1)}, x_{(t)}] + b_i). \tag{7}$$

Output Gate Operation: $o_{(t)}$.

- The output gate operation decides the value of the next hidden state.

$$o_{(t)} = \sigma(W_o \cdot [h_{(t-1)}, x_{(t)}] + b_o). \tag{8}$$

Cell Input Activation Vector: $g_{(t)} \in (-1,1)$.

$$g_{(t)} = \tanh(W_c \cdot [h_{(t-1)}, x_{(t)}] + b_c). \tag{9}$$

LSTM Core Concepts and Equations

Cell State Vector: $s_{(t)} \in \mathbb{R}$.

$$s_{(t)} = f_{(t)} \odot s_{(t-1)} + i_{(t)} \odot g_{(t)}. \tag{10}$$

Here, $g_{(t)}$ is the cell state information produced by tanh(); $s_{(t)}$ is the cell state information; $\odot$ is the Hadamord element-wise product operator.

LSTM Output: $h_{(t)} \in (-1, 1)$.

$$h_{(t)} = o_{(t)} \odot \tanh(s_{(t)}). \tag{11}$$

LSTM output is a sequence of hidden states , h_1, h_2, $\cdots$ h_T.

LSTM Mathematical Equations

Sequence of Equations:

$$
\begin{aligned}
f_{(t)} &= \sigma(W_f \cdot [h_{(t-1)}, x_{(t)}] + b_f),\\
i_{(t)} &= \sigma(W_i \cdot [h_{(t-1)}, x_{(t)}] + b_i),\\
o_{(t)} &= \sigma(W_o \cdot [h_{(t-1)}, x_{(t)}] + b_o),\\
g_{(t)} &= \tanh(W_c \cdot [h_{(t-1)}, x_{(t)}] + b_c),\\
s_{(t)} &= f_{(t)} \odot s_{(t-1)} + i_{(t)} \odot g_{(t)},\\
h_{(t)} &= o_{(t)} \odot \tanh(s_{(t)}).
\end{aligned}
$$

For More Information on Variants:

- Google: LSTM wikipedia.

Gated Recurrent Units

Gated Recurrent Units

Motivation

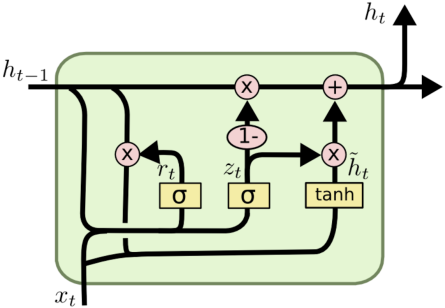
- Overcome shortcomings of standard RNN.
- Simplify implementation (c.f., LSTM).
- Improve ease of learning and performance.

Approach

- LSTM employs forget and output gates to control information change in the hidden state.
- GRU uses a single reset gate to achieve the same goal.

Gated Recurrent Units

GRU Architecture



Here:

- $x_{(t)}$ is the block input.
- $h_{(t)}$ is output passed forward to the next block.

Gated Recurrent Units

Reset Gate Vector: $r_{(t)}$

$$r_{(t)} = \sigma(W_r x_{(t)} + U_r h_{(t-1)} + b_r). \tag{12}$$

The reset gate $r_{(t)}$ determines how much of a hidden state to carry forward from a previous time-step. W, U and b are parameter matrices and bias vector.

Update Gate Vector: $z_{(t)}$

$$z_{(t)} = \sigma(W_z x_{(t)} + U_z h_{(t-1)} + b_z). \tag{13}$$

The update gate determines the relative strength of the contribution of the matrix-based update and a more direct contribution from the hidden vector $h_{(t-1)}$.

Gated Recurrent Units

The model can decide, for example, to copy all information from the past and eliminate the vanishing gradient problem.

Candidate Activation Vector: $\hat{h}_{(t)}$

$$\hat{h}_{(t)} = \tanh(Wx_{(t)} + U(r_{(t)} \odot h_{(t-1)})) \tag{14}$$

Here, $\odot$ is the Hadamord element-wise product operator.

Output Vector: $h_{(t)}$

$$h_{(t)} = z_{(t)} \odot h_{(t-1)} + \left(1 - z_{(t)}\right) \odot \hat{h}_{(t)}. \tag{15}$$

The output vector is passed to the next cell.

GRU Mathematical Equations

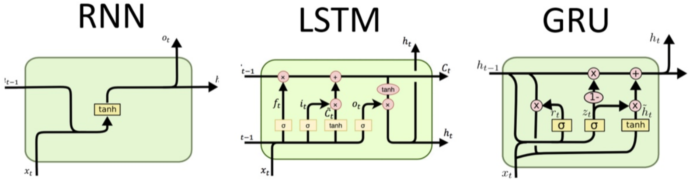
Summary: The output of GRU is a sequence of hidden states (output vectors), h_1, h_2, $\cdots$ h_T, calculated by the sequence of equations:

$$
\begin{aligned}
r_{(t)} &= \sigma(W_r x_{(t)} + U_r h_{(t-1)} + b_r) \\
z_{(t)} &= \sigma(W_z x_{(t)} + U_z h_{(t-1)} + b_z) \\
\hat{h}_{(t)} &= \tanh(W x_{(t)} + U(r_{(t)} \odot h_{(t-1)})) \\
h_{(t)} &= z_{(t)} \odot h_{(t-1)} + \left(1 - z_{(t)}\right) \odot \hat{h}_{(t)}.
\end{aligned}
$$

We assume that the input $x_{(t)}$ is a d-dimensional vector and the hidden states are p-dimensional. The transformation matrices in the first layer are sized accordingly.

Gated Recurrent Units

Comparison of Models: Simple vs LSTM vs GRU



LSTM vs GRU

- LSTM cells employ three gates – input, forget and output. GRU uses only reset and update gates.
- LSTM has cell states (long-term memory) and hidden states (short-term memory). GRU has only one hidden state.

Example 1

Memorize pi to 20 (and 100) decimal places

Example 1: Memorize pi to 20 decimal places

Problem Statement

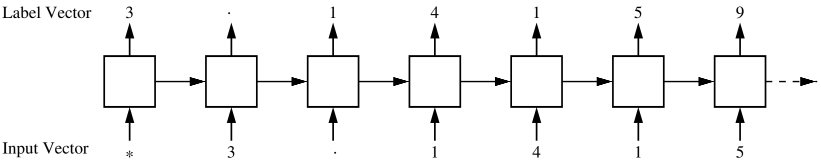
- Memorize first 20 digits of π (3.14159265358979323846).

Alphabet (11 characters: 9 digits + * + .)

```
[ *, 3, ., 1, 4, 5, 9, 2, 6, 8, 7 ]
```

Sequence and High-level Architecture (23 data points)

```
* 3 . 1 4 1 5 9 2 6 5 3 5 8 9 7 9 3 2 3 8 4 6
```



Example 1: Memorize pi to 20 decimal places

Annotated Encoding for Input Vector

```
[[ [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0],    <--- *
   [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, ... 0, 0],    <--- 3
   [ 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0],    <--- .
   [ 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, ... 0, 0],    <--- 1
   [ 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ... 1, 0],    <--- 4
   [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, ... 0, 0],    <--- 5
   [ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ... 0, 0],    <--- 9
   [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ... 0, 0],    <--- 2
   [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, ... 0, 1],    <--- 6
   [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0],    <--- 8
   [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0] ]] <--- 9
```

Size: (length $= 253$ (11×23))

Note: Input vector uses $*$ character to initialize the input.

Example 1: Memorize pi to 20 decimal places

Annotated Encoding for Label/Output Vector

```
[[ [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 1 ],    <--- *
   [ 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, ... 0, 0 ],    <--- 3
   [ 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0 ],    <--- .
   [ 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0 ],    <--- 1
   [ 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, ... 0, 0 ],    <--- 4
   [ 0 ,0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, ... 0, 0 ],    <--- 5
   [ 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, ... 0, 0 ],    <--- 9
   [ 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, ... 0, 0 ],    <--- 2
   [ 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, ... 1, 0 ],    <--- 6
   [ 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, ... 0, 0 ],    <--- 8
```

Size: (length $= 253$ (11×23))

Note: Label vector uses $*$ character to terminate sequence.

Example 1: Memorize pi to 20 decimal places

DL4J: Initialize Test Sentence and Alphabet

```
// Create test sequence ...

private static String testSentence = "*3.14159265358979323846";

// Convert character string to char arry ...

private static final char[] LEARNSTRING = testSentence.toCharArray();

// A list of all possible characters

private static final List<Character> LEARNSTRING_CHARS_LIST = new ArrayList<>();

// Create a dedicated list of possible chars in LEARNSTRING_CHARS_LIST

LinkedHashSet<Character> LEARNSTRING_CHARS = new LinkedHashSet<>();
for (char c : LEARNSTRING)
    LEARNSTRING_CHARS.add(c);

// Populate list of possibe characters ...

LEARNSTRING_CHARS_LIST.addAll(LEARNSTRING_CHARS);
```

Example 1: Memorize pi to 20 decimal places

DL4J: Initialize Input Data and Label Data Arrays

```
INDArray input  = Nd4j.zeros(1, LEARNSTRING_CHARS_LIST.size(), LEARNSTRING.length);
INDArray labels = Nd4j.zeros(1, LEARNSTRING_CHARS_LIST.size(), LEARNSTRING.length);
```

DL4J: Populate Input and Label Data Arrays

```
int samplePos = 0;
for (char currentChar : LEARNSTRING) {
   char nextChar = LEARNSTRING[(samplePos + 1) % (LEARNSTRING.length)];

   // input neuron for current-char is 1 at "samplePos"

   input.putScalar(
        new int[] { 0, LEARNSTRING_CHARS_LIST.indexOf(currentChar), samplePos }, 1);

   // output neuron for next-char is 1 at "samplePos"

   labels.putScalar(
        new int[] { 0, LEARNSTRING_CHARS_LIST.indexOf(nextChar), samplePos }, 1);

   samplePos++;
}
```

Example 1: Memorize pi to 20 decimal places

DL4J: RNN Architecture Assembly

```
// Set parameters in neural network configuration ...

NeuralNetConfiguration.Builder builder = new NeuralNetConfiguration.Builder();

builder.seed(123);
builder.biasInit(0);
builder.miniBatch(false);
builder.updater(new RmsProp(0.001));
builder.weightInit(WeightInit.XAVIER);

// Assemble network layers ...

ListBuilder listBuilder = builder.list();

listBuilder.layer(0, new LSTM.Builder().nIn( LEARNSTRING_CHARS.size() )
          .nOut(HIDDEN_LAYER_WIDTH)
          .activation(Activation.TANH).build() );

listBuilder.layer(1, new LSTM.Builder().nIn(HIDDEN_LAYER_WIDTH)
          .nOut(HIDDEN_LAYER_WIDTH)
          .activation(Activation.TANH).build() );
```

Example 1: Memorize pi to 20 decimal places

DL4J: RNN Architecture Assembly (continued)

```java

    // Create RNN output layer ...

    RnnOutputLayer.Builder outputLayerBuilder = new RnnOutputLayer.Builder(LossFunction.M
    outputLayerBuilder.activation(Activation.SOFTMAX);
    outputLayerBuilder.nIn(HIDDEN_LAYER_WIDTH);
    outputLayerBuilder.nOut(LEARNSTRING_CHARS.size());

    listBuilder.layer(HIDDEN_LAYER_CONT, outputLayerBuilder.build());

    // Create multilayer network configuration ...

    MultiLayerConfiguration conf = listBuilder.build();
    MultiLayerNetwork net = new MultiLayerNetwork(conf);

    net.init();
    net.setListeners(new ScoreIterationListener(1));
```

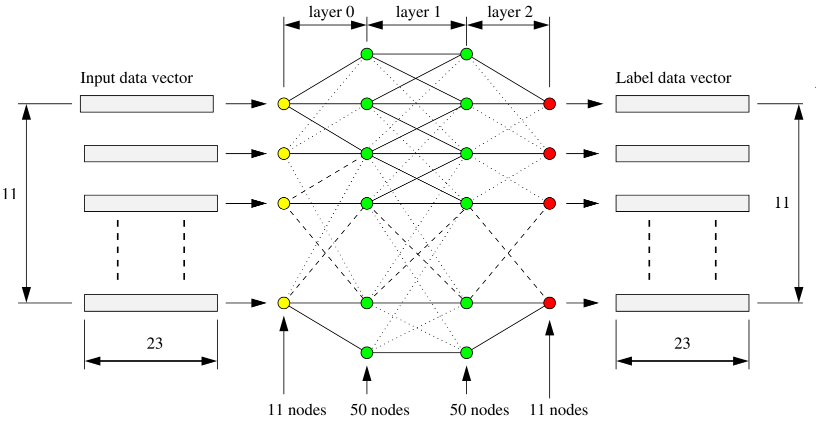
Example 1: Memorize pi to 20 decimal places

DL4J: RNN Architecture Summary

```
=================================================================
LayerName (Type)    nIn,nOut  NoParams   ParamsShape
=================================================================
   layer0 (LSTM)       11,50    12,400   W:{11,200}, RW:{50,200},
                                         b:{1,200}
   layer1 (LSTM)       50,50    20,200   W:{50,200}, RW:{50,200},
                                         b:{1,200}
   layer2 (RnnOutput)  50,11       561   W:{50,11}, b:{1,11}
-----------------------------------------------------------------
   Total Parameters:  33,161         Frozen Parameters:  0
   Trainable Parameters:  33,161
=================================================================
```

Example 1: Memorize pi to 20 decimal places

RNN Architecture + Input data vector + Label data vector



Example 1: Memorize pi to 20 decimal places

DL4J: RNN Training Procedure (1,000 epochs)

```
for (int epoch = 0; epoch < 1001; epoch++) {
    // train the data

    net.fit(trainingData);

    // clear current stance from the last example

    net.rnnClearPreviousState();

    // put the first character into the rrn as an initialisation

    INDArray testInit = Nd4j.zeros(1,LEARNSTRING_CHARS_LIST.size(), 1);
    testInit.putScalar(LEARNSTRING_CHARS_LIST.indexOf(LEARNSTRING[0]), 1);

    // run one step -> IMPORTANT: rnnTimeStep() must be called, not
    // output()
    // the output shows what the net thinks what should come next

    INDArray output = net.rnnTimeStep(testInit);
```

Example 1: Memorize pi to 20 decimal places

DL4J: RNN Training Procedure (continued)

```
        // Now the net should guess LEARNSTRING.length more characters

        for (char ignored : LEARNSTRING) {

            // First process the last output of the network to a concrete
            // neuron, the neuron with the highest output has the highest
            // chance to get chosen

            int sampledCharacterIdx = Nd4j.getExecutioner().exec(new IMax(output, 1)).ge

            // Use the last output as input

            INDArray nextInput = Nd4j.zeros(1, LEARNSTRING_CHARS_LIST.size(), 1);
            nextInput.putScalar(sampledCharacterIdx, 1);
            output = net.rnnTimeStep(nextInput);
        }
    }
```

Example 1: Memorize pi to 20 decimal places

DL4J: Iterations of Learning (1,000 epochs)

```
--- Epoch    0: 99944555993333333333333
--- Epoch   50: 111155555599333386****
--- Epoch  100: 1111525555899333846***
--- Epoch  150: 1111526555897933846***
--- Epoch  200: .11159265358979323846**
--- Epoch  250: ..1115265358979323846**
--- Epoch  300: ..14152265358979323846*
--- Epoch  350: ..14159263589799323846*
--- Epoch  400: ..1415265358979323846**
--- Epoch  450: 3.14159265358979323846*

... output removed ...

--- Epoch 1000: 3.14159265358979323846*
```

Example 1: Memorize pi to 20 decimal places

DL4J: Retrieve Outcomes from RNN Model (Loop over epochs)

```
for (int epoch = 0; epoch < 1001; epoch++) {

    if(epoch == 0 || epoch%50 == 0 ) System.out.printf("--- Epoch %4d: ", epoch);

    // train the data, then clear stance from last example ...

    net.fit(trainingData);
    net.rnnClearPreviousState();

    // put the first character into the rrn as an initialisation

    INDArray testInit = Nd4j.zeros(1,LEARNSTRING_CHARS_LIST.size(), 1);
    testInit.putScalar(LEARNSTRING_CHARS_LIST.indexOf(LEARNSTRING[0]), 1);

    // run one step -> IMPORTANT: rnnTimeStep() must be called, not
    // output()
    // the output shows what the net thinks what should come next

    INDArray output = net.rnnTimeStep(testInit);

    // Now the net should guess LEARNSTRING.length more characters
```

Example 1: Memorize pi to 20 decimal places

DL4J: Retrieve Outcomes from RNN Model (continued)

```
        for (char ignored : LEARNSTRING) {

            // First process the last output of the network to a concrete
            // neuron, the neuron with the highest output has the highest
            // chance to get chosen

            int sampledCharacterIdx = Nd4j.getExecutioner().exec(new IMax(output, 1)).g

            // Print the chosen output

            if(epoch == 0 || epoch%50 == 0 )
               System.out.print( LEARNSTRING_CHARS_LIST.get(sampledCharacterIdx) );

            // Use the last output as input

            INDArray nextInput = Nd4j.zeros(1, LEARNSTRING_CHARS_LIST.size(), 1);
            nextInput.putScalar(sampledCharacterIdx, 1);
            output = net.rnnTimeStep(nextInput);
         }

         if(epoch == 0 || epoch%50 == 0 )
            System.out.print("\n");
    }
```

Example 1: Memorize pi to 100 decimal places

Extend Model to 100 digits of pi

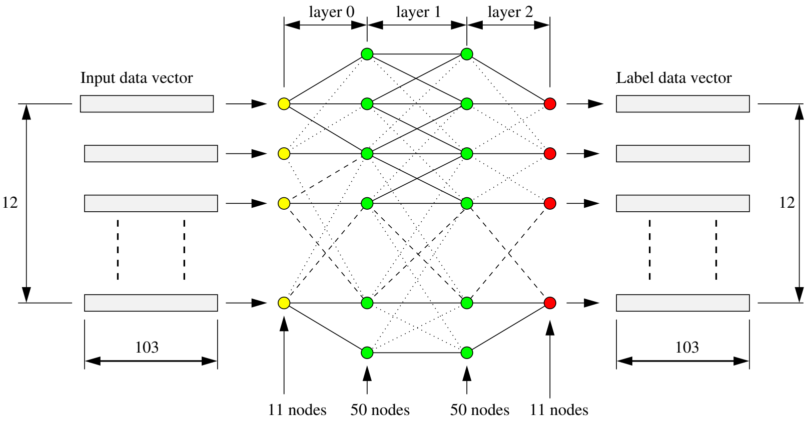
- The first 20 digits of π does not contain the digit zero.
- Hence, we only need an 11 character alphabet to memorize the sequence.
- The first 100 digits of π contains eight zeros:

```
*3.14159265358979323846
   26433832795028841971
   69399375105820974944
   59230781640628620899
   86280348253421170679
```

- The alphabet size is 12 and input data vector has dimension: 12×103.

Example 1: Memorize pi to 100 decimal places

RNN Architecture + Input data vector + Label data vector



Example 1: Memorize pi to 100 decimal places

Iterations of Learning (Compute 1,000 epochs)

```
-- Epoch    0: 9776333332229999997990000000000000000888888888888888
                888888888888888888888888888888888888888888888888888

-- Epoch   50: 5555555443333333333333333333333333333333333433333399
                999999999999999999888222266666888888888888888221111

... output removed ...

-- Epoch  900: .111595535897932384626433382275028841971693993375105
                 82097494459230781640028626089986280348253421170679*

-- Epoch  950: 3.14159265358979323846264338327950288419716939937510
                 58209749445923078164062862089986280348253421170679*

-- Epoch 1000: 3.14159265358979323846264338327950288419716939937510
                 58209749445923078164062862089986280348253421170679*
```

Example 2

Learning a Time Series

(Weather Conditions in Seattle)

Example 2: Weather in Seattle

Problem Statement

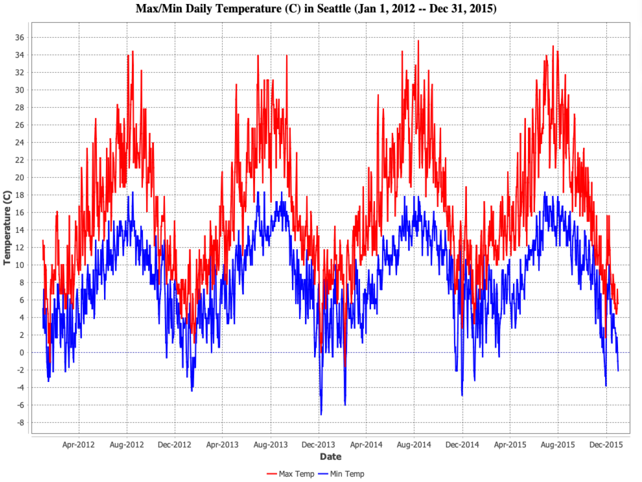
- Assemble LSTM RNN model for time-series analysis of weather conditions in Seattle.
- Forecast time-variation of weather conditions (i.e., min/max temperture, rainfall, general conditions).

Step-by-Step Procedure

- Import and clean the dataset.
- Preprocess (scale) and transform the dataset.
- Identify trends and seasonality.
- Uncover relationships (correlations) between variables.
- Train the machine learning model.

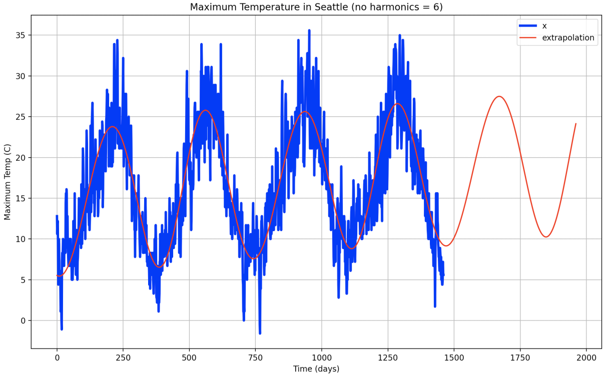
Example 2a: Max/min temperature in Seattle

Time-History: Max/min temperature in Seattle



Example 2a: Max/min temperature in Seattle

Preliminary Analysis: Extrapolated Fourier series with no harmonics = 6.



Source Code: See python-code.d/math/

Example 2a: Max/min temperature in Seattle

Fourier Analysis:

- Provides a way for general functions to be represented as the sum of simpler trigonometric (or exponential) functions.
- Google: fourier analysis examples.

Preliminary Analysis: Points to note:

- It seems that while extrapolated fourier series do a great job of capturing the general periodic nature of max/min temperature measurements, predictions of extreme values may be poor.
- We need to understand: by using a combination of weather measurements, can RNN outperform fourier series in the prediction of extreme values?

Example 2a: Weather in Seattle

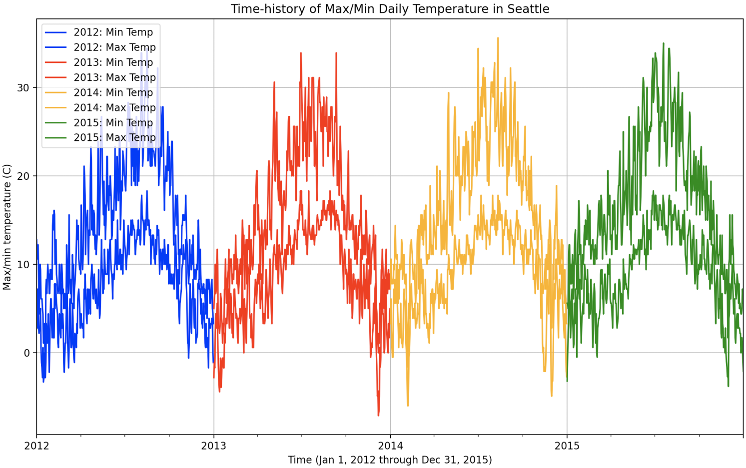
Weather Data in CSV format

Daily Weather Measurements: Jan 1, 2012 through Dec. 31, 2015

```
Date,Precipitation,TempMax,TempMin,Wind,Weather
2012-01-01,0.0,12.8,5.0,4.7,drizzle
2012-01-02,10.9,10.6,2.8,4.5,rain
2012-01-03,0.8,11.7,7.2,2.3,rain

.... data removed ...

2015-12-27,8.6,4.4,1.7,2.9,rain
2015-12-28,1.5,5.0,1.7,1.3,rain
2015-12-29,0.0,7.2,0.6,2.6,fog
2015-12-30,0.0,5.6,-1.0,3.4,sun
2015-12-31,0.0,5.6,-2.1,3.5,sun
```

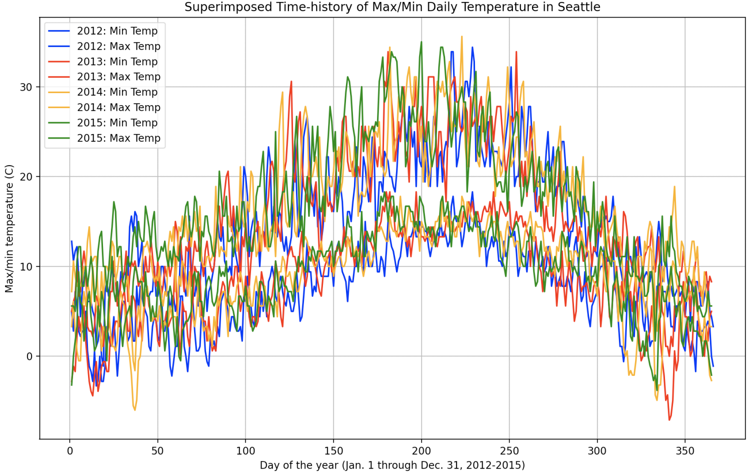
Example 2a: Weather in Seattle

Organize Data: Partition data into year-long batches:



Example 2a: Weather in Seattle

Organize Data: Map time to day-of-the-year.



Example 2a: Weather in Seattle

DL4J: Data Schema.

```
Schema():
idx    name         type          meta data
  0    "Date"       String        StringMetaData(name="Date",)
  1    "Prec ..n"   Double        DoubleMetaData(name="Precipitation",
                                    allowNaN=false,allowInfinite=false)
  2    "TempMax"    Double        DoubleMetaData(name="TempMax",
                                    allowNaN=false,allowInfinite=false)
  3    "TempMin"    Double        DoubleMetaData(name="TempMin",
                                    allowNaN=false,allowInfinite=false)
  4    "Wind"       Double        DoubleMetaData(name="Wind",
                                    allowNaN=false,allowInfinite=false)
  5    "Weather"    Categorical   CategoricalMetaData( name="Weather",
                                    stateNames=[ "fog", "rain",
                                     "drizzle",  "sun", "snow" ])
```

Example 2a: Max/min temperature in Seattle

DL4J: Data Transformation Process ...

```
TransformProcess(initialSchema=Schema():
idx    name      type      meta data
  0    "Date"    String    StringMetaData(name="Date",)
  ...
  actionList=[ DataAction(
               RemoveColumnsTransform([Precipitation, Wind, Weather]))
             ])
```

DL4J: Transformed Data Schema ....

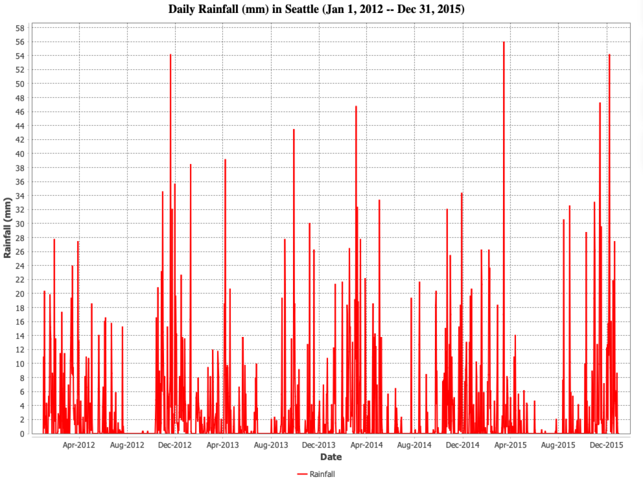
```
Schema():
idx   name        type      meta data
  0   "Date"      String    StringMetaData(name="Date",)
  1   "TempMax"   Double    DoubleMetaData(name="TempMax", ... )
  2   "TempMin"   Double    DoubleMetaData(name="TempMin", ... )
```

Example 2a: Max/min temperature in Seattle

DL4J: Training procedure ...

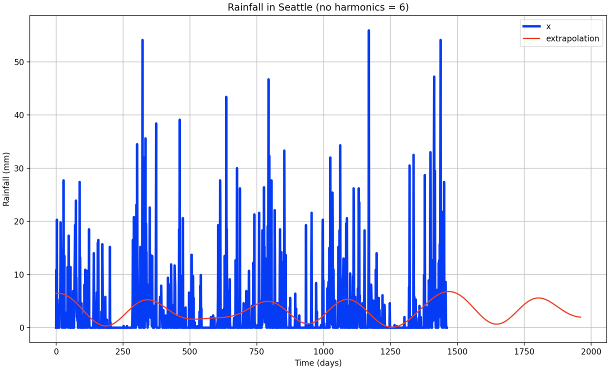
Example 2b: Daily Rainfall in Seattle

Time-History: Rainfall in Seattle



Example 2b: Daily Rainfall in Seattle

Preliminary Analysis: Extrapolated Fourier series with no harmonics = 6.



Source Code: See python-code.d/math/

Example 2b: Daily Rainfall in Seattle

DL4J: Data Transformation Process ...

```
TransformProcess(initialSchema=Schema():
idx   name             type         meta data
  0   "Date"           String       StringMetaData(name="Date",)
  1   "Precipitation"  Double       DoubleMetaData(name="Precipitation", ... )
  2   "TempMax"        Double       DoubleMetaData(name="TempMax", ...)
  ....
  5   "Weather"        Categorical  CategoricalMetaData(name="Weather", ... ),
  actionList=[
       DataAction(RemoveColumnsTransform([TempMax, TempMin, Wind, Weather]))
  ])
```

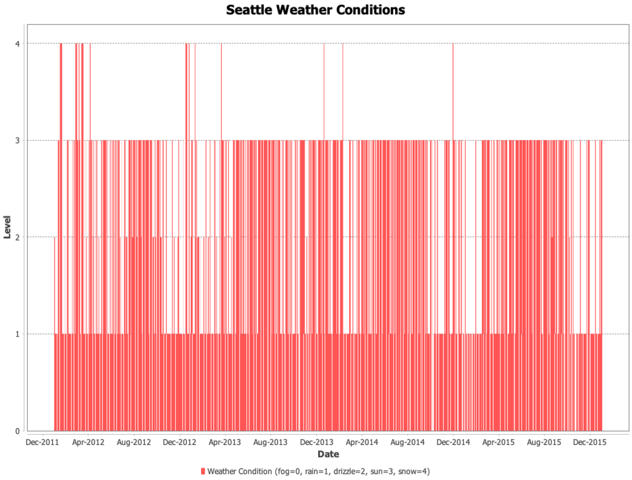
DL4J: Transformed Data Schema ....

```
Schema():
idx   name              type      meta data
  0   "Date"            String    StringMetaData(name="Date",)
  1   "Precipitation"   Double    DoubleMetaData(name="Precipitation", ... )
```

Example 2b: Daily Rainfall in Seattle

Example 2c: Weather conditions in Seattle

Weather Conditions: Rain, fog, drizzle, clear and sunny.



Example 2c: Weather conditions in Seattle

DL4J: Data Transformation Process ...

```
TransformProcess(initialSchema=Schema():
idx  name      type         meta data
  0  "Date"    String       StringMetaData(name="Date",)
  ....
  5  "Weather" Categorical  CategoricalMetaData(name="Weather",
                            stateNames=["rain","snow","drizzle","sun","fog"])
actionList = [
   DataAction( RemoveColumnsTransform([Precipitation, TempMax, TempMin, Wind]))
   DataAction( CategoricalToIntegerTransform(columnName=Weather, columnIdx=1,
              stateNames=[snow, rain, drizzle, fog, sun],
              statesMap={fog=0, rain=1, drizzle=2, sun=3, snow=4})) ])
```

DL4J: Transformed Data Schema ....

```
idx  name      type    meta data
  0  "Date"    String  StringMetaData( name="Date",)
  1  "Weather" Integer IntegerMetaData( name="Weather",
                                        minAllowed=0,maxAllowed=4)
```

Example 3

Generating New Text

(Learn Shakespeare, Write Shakespeare)

Example 3: Learn Shakespeare, Write Shakespeare

Problem Statement

- Train machine to remember complete works of Shakespeare, then use to model write new Shakespeare!

Sample of Shakespeare

```
onh with here yet, thou sleever'd thy tingress!
  Lold afoness Standous most cry is avain firsts.
  PORIO, SERVONT, BELANTIER Roiso folbathing better.
  WILLIAMA. Would weres this faith; mert too.
  I mean his lady's known TIMON and ASEDATN stones.
  ANTONIOUS Core, I had,,
  Com. Now thou art thou h
```

Example 3: Learn Shakespeare, Write Shakespeare

Solution Procedure

- Define problem parameters.
- Download complete works of Shakespeare (5.4 million characters).
- Setup single DatasetIterator.
- ...
- ...

Example 3: Learn Shakespeare, Write Shakespeare

DL4J: Set Problem Parameters

```
int lstmLayerSize =  200; // Number of units in each LSTM layer.
int miniBatchSize =   32; // Size of mini batch during training.
int exampleLength = 1000; // Length of training example sequence.

int tbpttLength   =   50; // Length for truncated backpropagation
                             through time.

int numEpochs = 1;        // Total number of training epochs.

int generateSamplesEveryNMinibatches = 10;  // How frequently to generate
                                               samples from the network?

int nSamplesToGenerate  =   4;    // No samples to generate after
                                     each training epoch.

int nCharactersToSample = 300;    // Length of each sample to generate.

String generationInitialization = null; // Optional character
                                           initialization.
```

Example 3: Learn Shakespeare, Write Shakespeare

DL4J: Downloading the Data

Example 3: Learn Shakespeare, Write Shakespeare

DL4J: Organize Data into Batches

Example 3: Learn Shakespeare, Write Shakespeare

DL4J: Assemble Neural Network Architecture

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()
    .seed(12345)
    .l2(0.0001)
    .weightInit(WeightInit.XAVIER)
    .updater(new Adam(0.005))
    .list()
    .layer( new LSTM.Builder().nIn(iter.inputColumns()).nOut(lstmLayerSize)
                              .activation(Activation.TANH).build()
          )
    .layer( new LSTM.Builder().nIn(lstmLayerSize).nOut(lstmLayerSize)
                              .activation(Activation.TANH).build()
          )

    // MCXENT + softmax for classification

    .layer( new RnnOutputLayer.Builder(LossFunction.MCXENT)
                .activation(Activation.SOFTMAX)
                .nIn(lstmLayerSize).nOut(nOut).build())
    .backpropType(BackpropType.TruncatedBPTT)
                .tBPTTForwardLength(tbpttLength)
                .tBPTTBackwardLength(tbpttLength).build();

MultiLayerNetwork net = new MultiLayerNetwork(conf);
net.init();
net.setListeners(new ScoreIterationListener(1));
```

Example 3: Learn Shakespeare, Write Shakespeare

DL4J: Neural Network Architecture

```
=====================================================================
LayerName (Type)    nIn,nOut  TotalParams  ParamsShape
=====================================================================
layer0 (LSTM)       77,200    222,400      W:{77,800}, RW:{200,800},
                                           b:{1,800}
layer1 (LSTM)       200,200   320,800      W:{200,800}, RW:{200,800},
                                           b:{1,800}
layer2 (RnnOutput)  200,77    15,477       W:{200,77}, b:{1,77}
---------------------------------------------------------------------
      Total Parameters:  558,677
  Trainable Parameters:  558,677
     Frozen Parameters:  0
=====================================================================
```

Example 3: Learn Shakespeare, Write Shakespeare

DL4J: Training the Model

Example 3: Learn Shakespeare, Write Shakespeare

New Shakespeare after completion of 10 minibatches

```
--- Sample: 0 ---

lshe wasle;
  FEGPMEON.SI. Ande ouncelor homith ans, theo nomi. we gome ming.
  CEIKMIAE. . ard satung oingese blomess sole:
  ' Thontlly hat aom, oud et  sigetere blerl'erdyanolith'sor.
  HSCGETERC. Pud ger fopee shilche'd coree.
  So hat wove gurgeninno, angemid treaot soucldor thiiy iotbense;

--- Sample: 1 ---

Fos Oow in wose adnasqyee anserintee mavith, ancn',
  dith, notherinoth, I ullelesd, Thou coRdy's,
  Cerenne resilg,
  Gleurt and, sor,
  Hond. Lhtoun, she ohid bowteruncicistutun. VOcrene,, Hiffall.
  Thivells, hil wither?
  JAROD. been,
```

Example 3: Learn Shakespeare, Write Shakespeare

New Shakespeare after completion of 170 minibatches

```
--- Sample: 0 ---

our reward is hope thy giad, be.
  Whithister's been, I, mind. Jien, Penmorace may far'et,
  Thou art not begit a' tears. 'That. Peacish, there-bitch!
  SIR TOBY. Where and thou that by amun. so you!
  That is you, kneel, for no furst,
  End be use essess the news.
  YORK. No, he is give think to

--- Sample: 1 ---

onh with here yet, thou sleever'd thy tingress!
  Lold afoness Standous most cry is avain firsts.
  PORIO, SERVONT, BELANTIER Roiso folbathing better.
  WILLIAMA. Would weres this faith; mert too.
  I mean his lady's known TIMON and ASEDATN stones.
  ANTONIOUS Core, I had,,
  Com. Now thou art thou h
```

Example 4

Use LSTM Recurrent Neural Network to

Identify trends in Control Chart Sequence Data

Example 4: Control Chart Sequence Data

Problem Statement

- This example learns how to classify univariate time series as belonging to one of six categories:

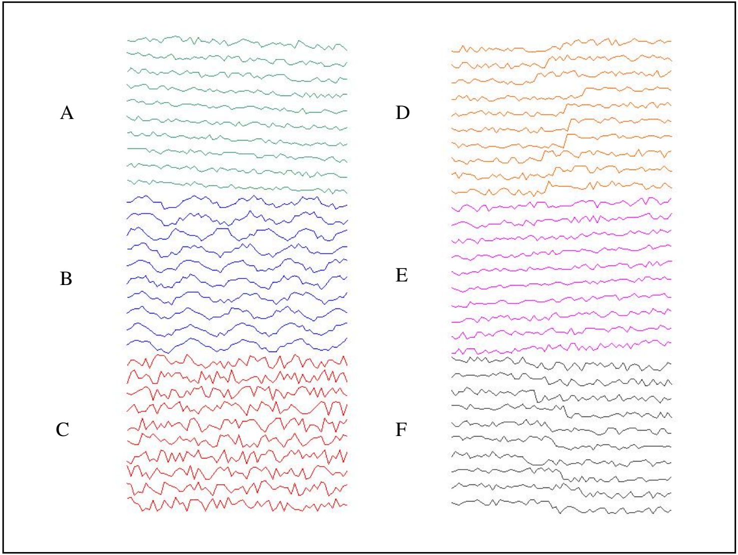
Data

- UCI Synthetic Control Chart Time Series Data Set contains 600 sequences of data.
- Partition data: 450 items for training; 150 items for testing.

Six Categories of Datastream

- **A and E:** Decreasing and Increasing Trend.
- **B:** Cyclic.
- **C:** Normal.
- **D and F:** Upward and Downward Shift.

Example 4: Representative Data Streams



Example 4: Control Chart Sequence Data

Training and Testing Corpus

- uci
  - train
    - features: 0.csv, 1.csv, 2.csv, 3.csv, ... ,449.
    - labels: 0.csv, 1.csv, 2.csv, 3.csv, ... ,449.
  - test
    - features: 0.csv, 1.csv, 2.csv, 3.csv, ... ,149.
    - labels: 0.csv, 1.csv, 2.csv, 3.csv, ... ,149.

CSV Data File Format and Label Mappings

Features: 0.csv

20.59 ← line 01
18.51
⋮
16.32
10.72 ← line 60

Labels: 0.csv

3

Output Mapping

0 ← A
1 ← B
2 ← C
3 ← D
4 ← E
5 ← F

Example 4: Control Chart Sequence Data

DL4J: Load Training Data, Features and Labels

```
// Initialize features and labels training directory ...

private static File baseDir = new File("resources/uci/");
private static File baseTrainDir     = new File(baseDir, "train");
private static File featuresDirTrain = new File(baseTrainDir, "features");
private static File labelsDirTrain   = new File(baseTrainDir, "labels");

// Read sequence of data files: 0.csv, 1.csv, ... 449.csv.

SequenceRecordReader trainFeatures = new CSVSequenceRecordReader();
trainFeatures.initialize(
   new NumberedFileInputSplit(
       featuresDirTrain.getAbsolutePath() + "/%d.csv", 0, 449));

SequenceRecordReader trainLabels = new CSVSequenceRecordReader();
trainLabels.initialize(
   new NumberedFileInputSplit(
       labelsDirTrain.getAbsolutePath() + "/%d.csv", 0, 449));
```

Example 4: Control Chart Sequence Data

DL4J: Create Minibatches and Dataset Iterator

```
int miniBatchSize   = 10;
int numLabelClasses =  6;

DataSetIterator trainData = new SequenceRecordReaderDataSetIterator(
    trainFeatures, trainLabels, miniBatchSize, numLabelClasses,
    false, SequenceRecordReaderDataSetIterator.AlignmentMode.ALIGN_END );
```

DL4J: Normalize the Training Data

```
DataNormalization normalizer = new NormalizerStandardize();
normalizer.fit(trainData);              //Collect training data statistics
trainData.reset();
```

Example 4: Control Chart Sequence Data

DL4J: Assemble Neural Network Architecture

```
MultiLayerConfiguration conf = new NeuralNetConfiguration.Builder()

 // Random number generator seed for improved repeatability. Optional.

 .seed(123)
 .weightInit(WeightInit.XAVIER)
 .updater(new Nadam())

 // Not always required, but helps with this data set

 .gradientNormalization(GradientNormalization.ClipElementWiseAbsoluteValue)
 .gradientNormalizationThreshold(0.5)
 .list()
 .layer( new LSTM.Builder().activation(Activation.TANH).nIn(1).nOut(10).build()
 .layer( new RnnOutputLayer.Builder( LossFunctions.LossFunction.MCXENT )
 .activation( Activation.SOFTMAX).nIn(10).nOut(numLabelClasses).build() )
 .build();

MultiLayerNetwork net = new MultiLayerNetwork(conf);
net.init();
```

Example 4: Control Chart Sequence Data

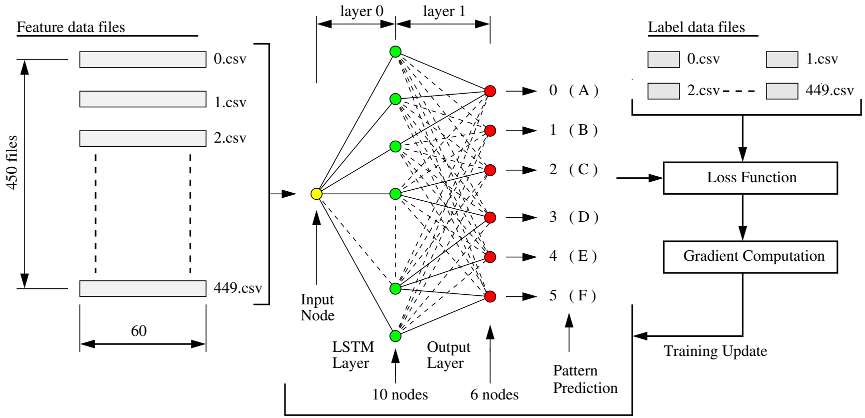
DL4J: Recurrent Neural Network Architecture

```
=================================================================
                                          Total     Params
LayerName (LayerType)       nIn,nOut   Params    Shape
=================================================================
layer0 (LSTM)               1,10        480       W:{1,40}, RW:{10,40},
                                                  b:{1,40}
layer1 (RnnOutputLayer)     10,6         66       W:{10,6}, b:{1,6}
-----------------------------------------------------------------
      Total Parameters:    546
  Trainable Parameters:    546
     Frozen Parameters:      0
=================================================================
```

Note: 1 input; 6 outputs.

Example 4: Control Chart Sequence Data

RNN Architecture + Sequences of Feature and Label vectors



Example 4: Control Chart Sequence Data

DL4J: Training the Model for 40 Epochs

```
int nEpochs = 40;
net.fit(trainData, nEpochs);
```

Evaluation Metrics and Confusion Matrix

```
Accuracy:  0.8867   Recall:   0.8890   <--- It works!
Precision: 0.8886   F1 Score: 0.8883

   0  1  2  3  4  5
-------------------
  26  0  0  0  0  0 |  0 = 0
   0 29  0  0  0  0 |  1 = 1
   0  0 15  0  7  0 |  2 = 2
   0  0  0 20  0  1 |  3 = 3
   0  0  9  0 21  0 |  4 = 4
   0  0  0  0  0 22 |  5 = 5
```

References

- Amidi A., and Amidi S., Cheetsheats on Machine Learning and Deep Learning, Various courses on ML at Stanford and MIT, 2018 – 2020.
- Apache DataVec. Library for machine learning ETL (Extract, Transform, Load) Operations. See: https://github.com/deeplearning4j/DataVec.
- Foster D., Generative Deep Learning: Teaching Machines to Paint, Write, Compose and Play, O'Reilly, 2019.
- Hochreiter S., and Schmidhuber J., Long Short-Term Memory, Neural Computation, Vol. 9, No. 8, 1997, pp. 1735-1780.
- Nielsen A., Practical Time Series Analysis: Prediction with Statistics and Machine Learning, OReilly, 2020.