Computational Intelligence (CI): Core Technologies and their need

<table>
<thead>
<tr>
<th>CI Core Technologies</th>
<th>Designed to mimic one or more aspects of biological intelligence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Neural Networks</td>
<td>• Main Components:</td>
</tr>
<tr>
<td>Fuzzy Systems</td>
<td>- Linguistic Information and Approximate Reasoning:</td>
</tr>
<tr>
<td>Evolutionary Algorithms</td>
<td>- Learning and Generalization:</td>
</tr>
<tr>
<td></td>
<td>- Neural Networks:</td>
</tr>
<tr>
<td></td>
<td>- Search &amp; Optimization:</td>
</tr>
</tbody>
</table>

The guiding principle of CI is: exploit the tolerance for imprecision, uncertainty and partial truth to achieve tractability, robustness and low solution cost.

A Difficult Problem

- $38450.6792 \times \log_{45} \cos 68 = ?$

Effortless Information Extraction

What we perceive...

\[ \bullet \quad \bullet \quad \bullet \quad \bullet \]
The Kanizsa Square (G. Kanizsa, 1976)

The Nature of Human Language

- It usually takes about three hours to drive from Delhi to Agra in normal traffic
- Most experts believe that the likelihood of a severe earthquake in the near future is very low
- Most of us believe that it is likely that global warming is the cause of major climatic changes all over the world

Human language is fuzzy

- The Fuzzy Moral: Precision Carries a High Cost

- Kinds of advice that can be given in a driving lesson when approaching a red light (Bezdek, 93):
  - “Begin braking 74 feet from the crosswalk…”
  - “You’d better apply the brakes pretty soon…”

The Most Complex Network

- 100,000,000,000 neurons
- 10,000 connections on average per neuron
- 1,000,000,000,000,000 connections in the human brain...
- The brain is a massively parallel feedback dynamical system

Pigeons are Art Experts (Watanabe et al. 1995)

- Demonstrate the generalization power of biological neural networks
- Experiment:
  - Pigeon in Skinner box
  - Present paintings of two different artists
    - Chapall / Van Gogh
  - Reward for pecking when presented a particular artist
    - Van Gogh

Van Gogh    Chagall

Observe the differences in style...
Results of the Pigeon Experiment

- Discrimination performance on training data (pictures they had been trained on): 95%
- Discrimination performance on unseen data (paintings of the artists they had not seen before): 85%
- Observations:
  - Pigeons do not simply memorise the pictures: they extract and recognise ‘style’ in the patterns
  - They generalise from the already seen to make predictions
- Generalization is what makes neural networks (biological and artificial) powerful!

The Inspiration for Artificial Intelligence

Pressing questions...

How does the human brain work?
How can we exploit the brain metaphor to build intelligent machines?

Deep Learning and Computational Intelligence

Good candidates for CI

- Fuzzy, imprecise or imperfect data
- No available mathematical algorithm
- Optimal solution unknown
- Rapid prototyping required
- Only domain experts available
- Robust system required

In News (2017)....

Diabetic retinopathy:
• A complication of diabetes that damages blood vessels in the retina and can lead to blindness.

Eyeagnosis:
a smartphone app plus 3D-printed lens that seeks to change the diagnostic procedure

In News (2016):
Deep Learning has Mastered Go
2017: AlphaGo Zero defeats AlphaGo 100-0

Mastering the game of Go without human knowledge algorithms became the first program to defeat a world champion in the game of Go. The tree search in AlphaZero evaluated positions and selected moves using deep neural networks. These neural networks were trained by supervised learning from human expert games, and by reinforcement learning from self-play. Here we introduce an algorithm based solely on reinforcement learning: without human data, guidance or domain knowledge beyond game rules, AlphaGo becomes an even teacher: a neural network is trained to predict AlphaGo's own move selections and also the winner of AlphaGo's games. This neural network improves the strength of the tree search, resulting in higher-quality move selection and strategy play in the next iteration. Starting safely new, our new program AlphaZero achieved superhuman performance, winning 100-0 against the previously published champion, defeating AlphaGo.

Advancements of deep learning have been transferred to Clinical Care

Off the shelf, pre-trained deep neural networks + 130,000 images = expert level diagnostic accuracy

Deep Learning in Genomics

Deep Learning can Imitate Style

Deep Learning in Computer Vision

Deep Learning can Play Video Games

Human-level control through deep reinforcement learning

Semantic Segmentation

Deep Learning can Imitate Style

A Neural Algorithm of Artistic Style

Leon A. Gatys, 1,2,3 Alexandre I. Efros 1,3,4 Matthew Bethge 2,3

Deep Learning in Genomics

Advancements of deep learning have been transferred to Clinical Care

Deep Learning can Play Video Games

Deep Learning in Computer Vision
Predicting Poverty from satellite imagery

http://sustain.stanford.edu/predicting-poverty/

The New Translation

- Neural nets reinvented image recognition - Google Photos
- Speech recognition to new levels via digital assistants - Google Now and Microsoft Cortana.
- Big leap in machine translation, the ability to automatically translate speech from one language to another - Google Neural Machine Translation, which operates entirely through neural networks. According to the company, this new engine has reduced error rates between 55 and 85 percent when translating between certain languages.
**Application Domains**

1. **CLASSIFIER DESIGN**
   - Normal
   - Abnormal
   - Airplane image partially occluded by clouds
   - Retrieved airplane image
   - Associative memory
   - Yann Lecun, Bell Labs introduced LeNet
   - Integrated into handwriting recognition systems
   - MNIST data set

2. **CLUSTERING**

3. **FUNCTION APPROXIMATION**
   - Over-fitting to noisy training data

4. **FORECASTING**

5. **CONTENT ADDRESSABLE MEMORY**

6. **CONTROL**

7. **OPTIMIZATION**

**What are Neural Networks?**

- Weight
- Sum
- Transform
- Signal
- Linear neuron
- Sigmoidal neuron
- Input layer
- Hidden layer
- Output layer

LeNet-5, convolutional neural networks, 1989

- Yann Lecun, Bell Labs introduced LeNet
- Integrated into handwriting recognition systems
- MNIST data set

http://yann.lecun.com/exdb/lenet/index.html

Unsupervised pre-training of “deep belief nets” allowed for large and deeper models


The Breakthrough (2012)

ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)

Large Scale Visual Recognition Challenge (ILSVRC)
• Annual ImageNet Challenge starting in 2010
• Many tracks including classification and localization
• Standardized training and test set.
• Competitors upload predictions for test set and are automatically scored

Imagenet Database
• Millions of labeled images
• Objects in images fall into 1 of a possible 1,000 categories
• Relatively high-resolution
• Bounding boxes giving exact location of object - useful for both classification and localization

ImageNet Dataset

ImageNet: A Large-Scale Hierarchical Image Database

The Breakthrough (2012)

• In 2011, a misclassification rate of 25% was near state of the art on ILSVRC
• In 2012, Geoff Hinton and two graduate students, Alex Krizhevsky and Ilya Sutskever, entered ILSVRC with one of the first deep neural networks trained on GPUs, now known as “Alexnet”

Result: An error rate of 16%, nearly half what the second place entry was able to achieve.
• The computer vision world immediately took notice

ILSVRC aftermath

Alexnet paper has ~ 10,000 citations since being published in 2012!

Networks keep getting deeper...

Deep = Many hidden layers


8 layers
16.4%
7.3%
6.7%
Deep = Many hidden layers

AlexNet (2012) 16.4%
VGG (2014) 7.3%
GoogleNet (2014) 6.7%
Residual Net (2015) 3.57%

Activations Measure Similarities

- The activation $x_j$ is an inner product!
- $S = (s_0, \ldots, s_n)^T$, $W_j = (w_{0j}, \ldots, w_{nj})^T$

Activation compute the similarity between the input vector and weight vector!

Artificial Neuron (Basic Computing Unit)

$x_j = \sum_{i=0}^{n} w_{ij} s_i$

Activations compute the similarity between the input vector and weight vector!

Binary Threshold Signal Function

$s_j = \begin{cases} 1 & x_j \geq 0 \\ 0 & x_j < 0 \end{cases}$

Linear Threshold Signal Function

$s_j = \begin{cases} 0 & x_j \leq -\Theta_j \\ \frac{x_j}{\alpha_j} & -\Theta_j < x_j < \Theta_j \\ 1 & x_j \geq \Theta_j \end{cases}$

Interpretation of Threshold

- From the point of view of the net activation $x_j$
  - $s = 1$ if $x_j = q_j + \Theta_j \geq 0$, or $q_j \geq -\Theta_j$
  - $s = 0$ if $q_j < -\Theta_j$
- Positive bias makes the neuron to fire easily
- Negative bias makes the neuron to fire difficulty

$\alpha_j = 1/\theta_m$ is the slope parameter
Sigmoidal Signal Function
- $\lambda_j$ is a gain scale factor
- monotonic, continuous, bounded

$$S_j(x_j) = \frac{1}{1 + e^{-\lambda_j x_j}}$$

Gaussian Signal Function
- $\sigma_j$ is the Gaussian spread factor and $c_j$ is the center
- Non-monotonic

$$S_j(x_j) = \exp \left( -\frac{(x_j - c_j)^2}{2\sigma_j^2} \right)$$

Stochastic Neurons
- The signal is assumed to be two state
  - $x_j \in \{0, 1\}$
- Neuron switches into these states depending upon a probability function $P(x_j)$

A Pattern Classifier
- search for structure in data
- placing an input pattern into one of possibly many decision classes

A Decision Boundary

Basic Pattern Classification Problem
- Given some samples of data measurements of a real world system along with correct classifications for patterns in that data set, make accurate decisions for future unseen examples
Iris subspecies – Iris Virginica, Iris Versicolor, Iris Setosa

Iris Data Classification
- It is easy to separate iris setosa data from iris versicolor and iris virginica.
- It is much more difficult to decide where to place a separating line between iris versicolor and iris virginica.
- Any placement of the straight line will cause some pattern to get misclassified.

Convex Sets
- Let $X, Y \in S \subset \mathbb{R}^n$, then $S$ is convex iff $\lambda X + (1 - \lambda)Y \in S$, $0 \leq \lambda \leq 1$, $\forall X, Y \in S$.
- Equivalently, a set $S$ is convex if it contains all points on all line segments with end points in $S$.

Convex Hull
- The convex hull, $C(X_i)$, of a pattern set $X_i$ is the smallest convex set in $\mathbb{R}^n$ which contains the set $X_i$.

Linearly Separable Classes and Separating Hyperplane
- Two pattern sets $X_i$ and $X_j$ are said to be linearly separable if their convex hulls are disjoint, that is if $C(X_i) \cap C(X_j) = \emptyset$. 

Feature Scatter Plots
The Boolean AND Function is Linearly Separable

- (0,0), (0,1), and (1,0) map to 0, as indicated by unfilled circles.
- (1,1) maps to 1, as indicated by a filled circle.

Separating line:

\[ x_2 = -(w_1/w_2)x_1 - w_0/w_2 \]

Small arrow on the straight line indicates the half plane that translates to +1 signal value.

Truth Table

<table>
<thead>
<tr>
<th>( x_1 )</th>
<th>( x_2 )</th>
<th>( f(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Discriminant Function

- \( x_2 = -\langle w_1/w_2 \rangle x_1 - \langle w_0/w_2 \rangle \)
- Slope: \( m = -w_1/w_2 = -1 \)
- Intercept: \( c = -w_0/w_2 = 1.5 \)
- Choose \( w_1 = w_2 = 1 \) and \( w_0 = -1.5 \)

Geometrical Design of AND Classifier

- \( x_2 = -(w_1/w_2)x_1 - (w_0/w_2) \)
- Slope: \( m = -w_1/w_2 = -1 \)
- Intercept: \( c = -w_0/w_2 = 1.5 \)
- Choose \( w_1 = w_2 = 1 \) and \( w_0 = -1.5 \)

Geometrical Design of NAND Classifier

- \( x_2 = -\langle w_1/w_2 \rangle x_1 - \langle w_0/w_2 \rangle \)
- Slope: \( m = -w_1/w_2 = -1 \)
- Intercept: \( c = -w_0/w_2 = 1.5 \)
- Choose \( w_1 = w_2 = -1 \) and \( w_0 = 1.5 \)

Geometrical Design of OR Classifier

- \( x_2 = -(w_1/w_2)x_1 - (w_0/w_2) \)
- Slope: \( m = -w_1/w_2 = -1 \)
- Intercept: \( c = -w_0/w_2 = 0.5 \)
- Choose \( w_1 = w_2 = 1 \) and \( w_0 = -0.5 \)
Non-linearly Separable Problems

XOR is Not Linearly Separable

- The geometry of the Boolean XOR function shows that two straight lines are required for proper class separation.

Truth Table

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Region of 1's

Solving XOR Problem

Network of TLNs to solve XOR

Solution 1: Reduce the Number of Points

- The logical functions implemented by each TLN (first layer nodes do not compute anything) is indicated in the figure.
- The XOR problem has been solved in two steps:
  - first there is a map $f_1: B^2 \rightarrow B$
  - Then, there is a second map $f_2: B^2 \rightarrow B$
- Note that $Y = f_1(x)$ where $x_1 = x_1\cdot x_2$ and $y_1 = x_1\cdot x_2$

Four Points In 2-d Mapped To Three Make XOR Linearly Separable

$y_1 = x_1 + x_2\cdot y_2 = x_1\cdot y_2$
Solving XOR by Increase of Dimension

The AND Function Added As The Third Dimension

Multilayered TLN Networks

Typical Multilayered Neural Network Architecture

Neural Network Components

Pattern of Connectivity of Neural Network

- Neurons
  - Activation (How are inputs aggregated?)
  - Signal function (How are activations transformed?)
- Pattern of connectivity:
  - How do you connect basic computing units “neurons” to create a network?
- Learning rule
  - How do you learn the “best” weights?
Various Neural Network Models...

Summary of Neural Networks Models

- Multilayer perceptron (MLP)
- Radial Basis Function Network (RBFN)
- Support Vector Machine (SVM)
- Hopfield Network
- Bidirectional Associative Memory (BAM)
- Adaptive Resonance Theory (ART) models
- Self Organizing Feature Maps (SOFMs)
- Boltzmann Machine (BM)
- Convolutional Neural Network (CNN)
- Restricted Boltzmann Machine (RBM)
- Stacked Auto Encoder (SAE)
- Deep Belief Network (DBN)
- Long Short Term Memory Network (LSTM)
- Extreme Learning Machine (ELM)
- Echo State Network (ESN)
- Spiking Neural Networks (SNNs)

Salient Properties

- Robustness
  - Ability to operate, albeit with some performance loss, in the event of damage to internal structure.
- Associative Recall
  - Ability to invoke related memories from one concept.
- Function Approximation and Generalization
  - Ability to approximate functions using learning algorithms by creating internal representations
    - Knowledge of mathematical model of how outputs depend on inputs not required
  - Can use such internal models to extrapolate to new inputs

The Supervised Learning Procedure

1. Select a pattern $X_k$ from the training set $T$ present it to the network.
2. Forward Pass: Compute activations and signals of input, hidden and output neurons in that sequence.
3. Error Computation: Compute the error over the output neurons by comparing the generated outputs with the desired outputs.
4. Compute Weight Changes: Use the error to compute the change in the hidden to output layer weights, and the change in input to hidden layer weights such that a global error measure gets reduced.
5. Update all weights of the network.
6. Repeat Steps 1 through 5 until the global error falls below a predefined threshold.

Square Error Function

- The instantaneous summed squared error $e_k$ is the sum of the squares of each individual output error $e_i^k$, scaled by one-half:

$$E_k = (e{1}, \ldots, e_i^k, \ldots, e_n) = (d^i_1 - S(x^i_1), \ldots, d^i_m - S(x^i_m))^2$$

$$\bar{E}_k = \frac{1}{2} \sum_{j=1}^m (d^i_j - S(x^i_j))^2 = \frac{1}{2} E_k^2 E_k$$

$$\bar{\varepsilon} = \frac{1}{n} \sum_{k=1}^n \bar{E}_k$$

Error Surface
Gradient Descent Procedure

Recall: Gradient Descent Update Equation

- It follows logically therefore, that the weight component should be updated in proportion with the negative of the gradient as follows:
  \[ w_i^{k+1} = w_i^k + \eta \left( \frac{\delta E}{\delta w_i^k} \right) \]
  \( i = 0, 1, \ldots, n \)

Square Error Performance Function

- The \( k \)th training pair (\( X_k, D_k \)) then defines the instantaneous error:
  \[ E_k = D_k - S(Y^k) \]
  \[ E_k = (e_1^k, \ldots, e_p^k) \]
  \[ = (d_1^k - S(y_1^k), \ldots, d_p^k - S(y_p^k)) \]

- The instantaneous summed squared error \( E_k \) is the sum of the squares of each individual output error \( e_j^k \), scaled by one-half:
  \[ E_k = \frac{1}{2} \sum_{j=1}^{p} (d_j^k - S(y_j^k))^2 \]

Backpropagation Learning

1. For hidden to output layer weights:
   \[ w_{ij}^{k+1} = w_{ij}^k + \Delta w_{ij}^k \]
   \[ = w_{ij}^k + \eta \left( \frac{\delta E_k}{\delta w_{ij}^k} \right) \]
   \[ = w_{ij}^k + \eta \frac{\delta E_k}{\delta w_{ij}^k} \]

2. For input to hidden layer weights:
   \[ w_{ik}^{k+1} = w_{ik}^k + \Delta w_{ik}^k \]
   \[ = w_{ik}^k + \eta \left( \frac{\delta E_k}{\delta w_{ik}^k} \right) \]
   \[ = w_{ik}^k + \eta \frac{\delta E_k}{\delta w_{ik}^k} \]

How do we learn the "best" values for weights?

Gradient Descent
- Give weights random initial values
- Evaluate the gradient of error surface at current weight value
- Take a step in direction opposite to the gradient
- Repeat

Backpropagation
- Backpropagation: an efficient way to compute \( \frac{\partial E}{\partial w} \) in neural network

Ref: https://www.youtube.com/watch?v=ibJpTrp5mcE
Some tips on learning...

- Approximation:
  - Is the Network performing well on training data?
    - Modify network (signal function, structure)
    - Better optimization strategy (learning rate, momentum...)

- Generalization:
  - Is the Network performing well on the test data (true unseen data)?
    - Prevent overfitting - dropout

"Feature Learning" is the hallmark of deep learning approaches

- Deep learning methods aim at learning feature hierarchies with features from higher levels of the hierarchy formed by the composition of lower level features.

Traditional approach has the following limitations:
- It is very tedious and costly to develop hand-crafted features
- The hand-crafted features are usually highly dependent on one application, and cannot be transferred easily to other applications

Some Popular Deep Learning Models

- Convolutional Neural Networks (CNNs)
- Deep Belief Networks (DBNs)
- Autoencoders and stacked autoencoders (SAE)
- Long Short Term Memory (LSTM)
- Generative Adversarial Networks (GANs)

Convolutional Neural Network

- Input can have very high dimension. Using a fully-connected neural network would need a large amount of parameters.
- Inspired by the neurophysiological experiments conducted by [Hubel & Wiesel 1962], CNNs are a special type of neural network whose hidden units are only connected to local receptive field. The number of parameters needed by CNNs is much smaller.

Example: 200x200 image
a) fully-connected: 40,000 hidden units => 1.6 Billion parameters
b) CNN: 5x5 kernel, 100 feature maps => 2,500 parameters
CNN architecture

1. Convolution stage
2. Nonlinearity: a nonlinear transform such as rectified linear or tanh
3. Pooling: output a summary statistics of local input, such as max pooling and average pooling

Deep CNN: winner of ImageNet 2012

A typical CNN

What’s Changed? Why Now?

• Several key advancements has enabled the modern deep learning revolution
  – Availability of massive datasets with high-quality labels
  – Standardized benchmarks of progress and open source tools
  – Community acknowledgment that open data -> everyone gets better

Software Frameworks

• Frameworks like tensorflow, theano, chainer, and mxnet that provide automatic differentiation allow for seamless GPU computing and make prototyping faster and less error-prone.
• They let you focus on your model structure without having to worry about low-level details like gradients and GPU management.
What’s changed? Why now?

• Backprop-friendly activation functions
• Improved architectures
• New regularization techniques
  – Techniques like dropout, batch normalization, and data-augmentation allow us to train larger and larger networks without (or with less) overfitting
• Robust optimizers
  – Modifications of the SGD procedure including momentum, RMSprop, and ADAM have helped with higher performance

Encouragement and Caution!!

• Barrier to entry for deep learning is actually low
  ... but a few things might stand in your way:
  – Need to make sure your problem is a good fit
    • Lots of labeled data and appropriate signal/noise ratio
  – Access to GPUs
  – Many design choices and hyper parameter selections
  – Training model require a high level of skill and experience

Benchmark Data Sets

http://deeplearning.net/datasets/

NIPS 2017


Kaggle Datasets

https://www.kaggle.com/datasets
Journals and Conferences

- Journals
  - IEEE Pattern Analysis and Machine Intelligence (PAMI),
  - IEEE Transaction on Neural Networks and Learning Systems
  - Neural Networks
- Conferences
  - NIPS
  - ICML
  - IEEE IJCNN
  - IEEE WCCI

Conclusions

- Neural Networks and Deep learning are probably here to stay
- Synergistic integration with CI will reap benefits
- Could potentially impact many fields
- Prerequisites: Data (lots) + GPUs (more = better)
- Deep learning models are like Legos,
  - Need to know what blocks you have and how they fit together
- Need to have a sense of sensible default parameter values to get started
- Training the model using the learning process is a skill

Some Examples

<table>
<thead>
<tr>
<th>S. No.</th>
<th>Sepal Length</th>
<th>Sepal Width</th>
<th>Petal Length</th>
<th>Petal Width</th>
<th>Species</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5.1</td>
<td>3.5</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>2</td>
<td>4.9</td>
<td>3.0</td>
<td>1.4</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>3</td>
<td>4.7</td>
<td>3.1</td>
<td>1.3</td>
<td>0.2</td>
<td>setosa</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>51</td>
<td>7.0</td>
<td>3.2</td>
<td>4.7</td>
<td>1.4</td>
<td>versicolor</td>
</tr>
<tr>
<td>52</td>
<td>6.4</td>
<td>3.2</td>
<td>4.5</td>
<td>1.5</td>
<td>versicolor</td>
</tr>
<tr>
<td>53</td>
<td>6.0</td>
<td>3.3</td>
<td>4.0</td>
<td>1.5</td>
<td>versicolor</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>101</td>
<td>6.3</td>
<td>3.3</td>
<td>6.0</td>
<td>2.5</td>
<td>virginica</td>
</tr>
<tr>
<td>102</td>
<td>5.8</td>
<td>2.7</td>
<td>5.1</td>
<td>1.9</td>
<td>virginica</td>
</tr>
<tr>
<td>103</td>
<td>7.1</td>
<td>3.9</td>
<td>5.9</td>
<td>2.1</td>
<td>virginica</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Iris Classification

Presentation of Data Patterns

Modify weights to reduce error

Repeat process (sweep) for all training pairs

- Present data
- Calculate error
- Back-propagate error
- Adjust weights
- Repeat process
After training

<table>
<thead>
<tr>
<th>Feature</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sepal length</td>
<td>5.1</td>
</tr>
<tr>
<td>Sepal width</td>
<td>3.5</td>
</tr>
<tr>
<td>Petal length</td>
<td>1.4</td>
</tr>
<tr>
<td>Petal width</td>
<td>0.2</td>
</tr>
</tbody>
</table>

Results

- Network learns to classify the patterns
- Network can classify within reasonable accuracy, patterns it has not seen before
- Salient properties that neural networks demonstrate
  - Robustness
  - Associative Recall
  - Generalization

Function Approximation

Error vs Epochs

Simulation Snapshots

Error Histogram and Error Mesh
Time Series Prediction

- Time series prediction is very similar to function approximation except time plays an important role
  - In “static” function approximation, all information needed to create the output is contained in the current input
  - In time series prediction (dynamic function approximation), information from the past is needed to determine the output

Prediction Network:
Function approximation with future values as output

Past and present values of input parameters
Future values to be predicted

Time Series Prediction Example

- Predict Mackey Glass chaotic signal
  - Chaos is a signal that has characteristics similar to randomness, but can be predicted accurately in the short term (e.g., weather)
  - Accurate predictions can be made only a few samples in advance

\[
\begin{align*}
\phi(t) &= 0.2\phi(t - \tau) - 0.1\phi(t) \\
\phi(t) &= 1 + x^2(t - \tau)
\end{align*}
\]

Truck Backer Upper Control

- Backing up a truck to loading dock

\[
\begin{align*}
x(t + 1) &= x(t) + \cos(\phi(t) + \Theta(t)) + \sin(\Theta(t)) \sin(\phi(t)) \\
y(t + 1) &= y(t) + \sin(\phi(t) + \Theta(t)) - \sin(\Theta(t)) \cos(\phi(t)) \\
\phi(t + 1) &= \phi(t) - \sin(\frac{1}{2})
\end{align*}
\]

Design of Truck Backer Upper Control

- Enough clearance assumed between the truck and the loading dock such that the coordinate y can be ignored.
- Numeric data comprises of 238 pairs accumulated from 18 sequences of desired \((x, \phi, \theta)\) values (Wang/Mendel, 1992).
- Range of \(x\): 0 to 20
- Range of \(\phi\): -90° to 270°
- Range of \(\theta\): -40° to +40°

Control value of \(\theta\) such that the final state \((x_f, \phi_f)\) = (10, 90°)

Truck Backer Upper Trajectories