Deep Neural Networks

Introduction to Actuarial Applications for emerging AI
CAS RPM 2017

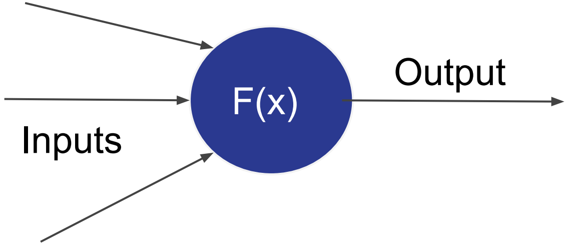
Prashant.De@gmail.com

Agenda

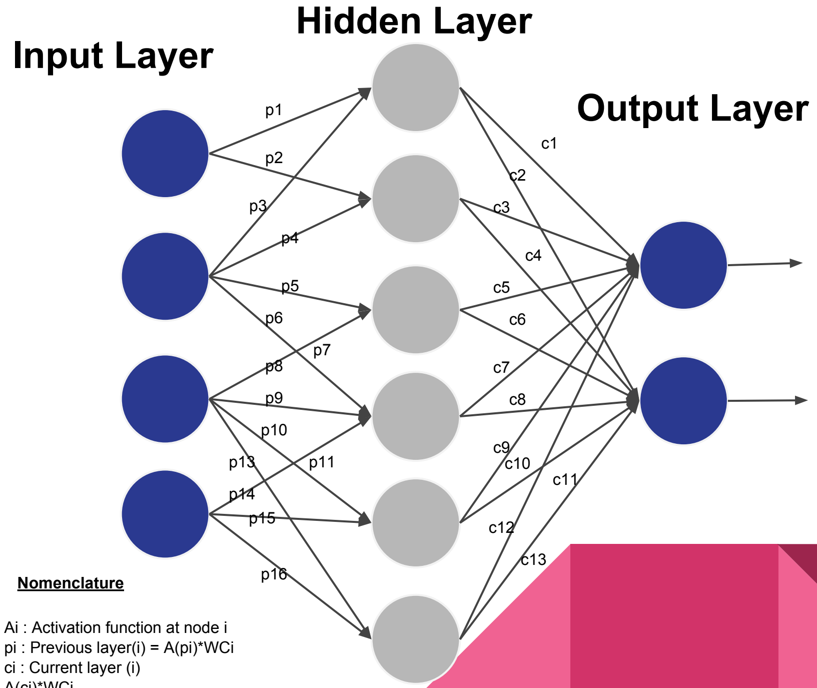
1) Neural Networks - Introduction
2) What does “Deep” mean in Neural Network terminology
   a) Stochastic Gradient Descent
   b) Backpropagation is the key
   c) What drives the network growth
3) Architecture is everything: How Deep Learning architectures solve problems
   a) Convolutional Neural Networks with Images + Demo
   b) RNN/LSTM with Text + Demo
4) Actuarial Applications
   a) Applications for deep neural nets in insurance
   b) Challenges and risks : Questions actuaries should ask during neural net discussions

1) Perceptrons the building blocks of neural nets

Perceptron: basis of a neural network



F(X) is a definable function



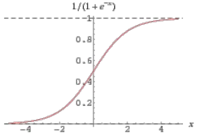
Nomenclature

Ai : Activation function at node i
pi : Previous layer(i) = A(pi)*WCi
ci : Current layer (i)
A(ci)*WCi

Terminology before we dive in

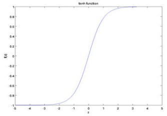
- **“Pre-processing”:** Data transforms readying data for input to a neural net. Key decisions include
- **“Feed-Forward”:** Neural Networks where signal from functions travel in direction from input to output
- **“Convolutional”:** Convolving across a larger matrix using a smaller filter
- **“Recurrent”:** Networks where outputs are re-used as inputs at the next time step
- **“Long Short Term Memory”:** Commonly used in a recurrent network, a static laye recording the previous output with functions to control input, memory and output to the next step
- **“Pooling”:** Downsample layer to reduce overfitting and parameters to learn
- **“Dropout”:** A method to randomly drop connections to reduce over-fitting
- **“Fully Connected”:** All neurons receive an input signal and deliver an output signal
- **“Backpropagation”:** Adjusting the network weights by first minimizing the loss function and working back towards the input, adjusting connected weights at each step
- **“Stochastic Gradient Descent”:** A step towards a smaller loss function using partial derivatives
- **“Hidden Layer”:** Series of functions that process input signal and output to another layer. Hidden since input and output layers are transparent
- **“Epoch”:** A single run through the neural network from input to output (iteration)

Sigmoid Function(σ): (1)/(1+exp(-x))



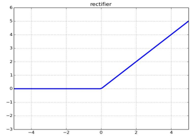
Purpose: Similar to logistic, binomial classification

Tanh Function: tanh(x)=2σ(2x)−1



Purpose: To transform a -ve 1 scale to +ve **1**

ReLU function: ~ max(0 ,ln(1+exp(x))) x:[0:1]



Purpose: Activate as +ve multiplier at gradient 1 to output

SoftMax Function: Prob (yi | input) = exp(xi*wi)/(Sum: exp(xi*wi: for 1 to D)

Purpose: Transform multinomial scalar outputs to posterior log normalized vector squashed between 0 and 1.Multinomial.

2) The “Deep” in Deep Neural Nets

GoogleNet for Image Classification

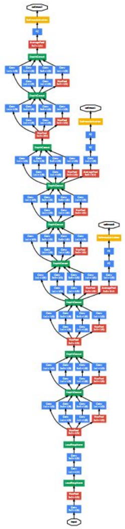
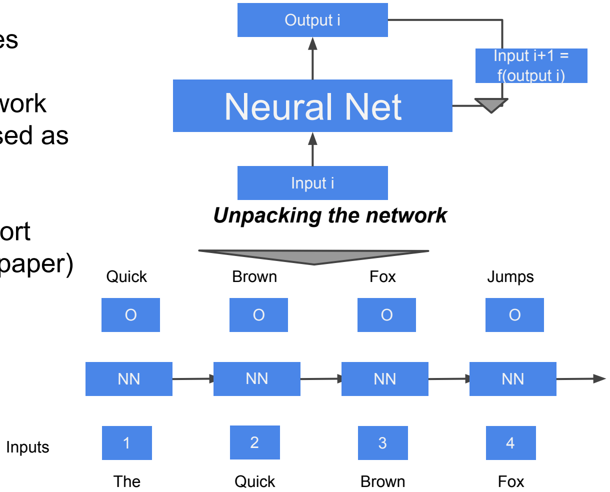


Figure 3: GoogLeNet network with all the bells and whistles.

- 22 Layers deep
- Multiple entry points and merges
- Image classification focus
- Example of a feed-forward network where the outputs are not re-used as inputs
- Purpose is to classify images
- A tangential purpose is to support internet memes (referenced in paper)

https://www.cs.unc.edu/~wliu/papers/GoogLeNet.pdf : Szegedy et al. “In this paper, we will focus on an efficient deep neural network architecture for computer vision, codenamed Inception, which derives its name from the Network in network paper by Lin et al [12] in conjunction with the famous “we need to go deeper” internet meme [1]”

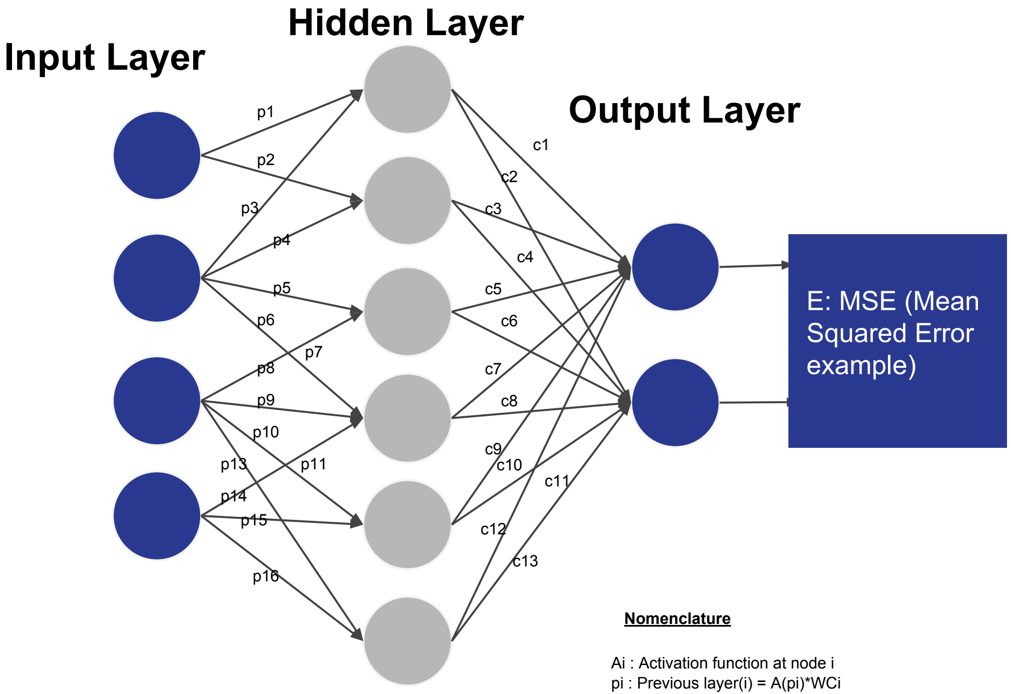
Recurrent Neural Networks for Text Processing



Notes to consider:

1. **The layers of abstraction are akin to “feature engineering”**. By abstracting the data into “hypotheses” to test that a human might not arrive at to develop as a hypothesis for the model
2. **Works well for specific problems** such as images, and text in some cases where there is complex data problem that needs a general solution
3. **This is important to Actuaries for three specific reasons:**
   a. Abstraction layers are not easy to explain, nor are results
   b. Network architecture are important but complex; mistakes can be made
   c. Overfitting is an issue(use dropouts)

2a) Stochastic Gradient Descent



E: MSE (Mean Squared Error example)

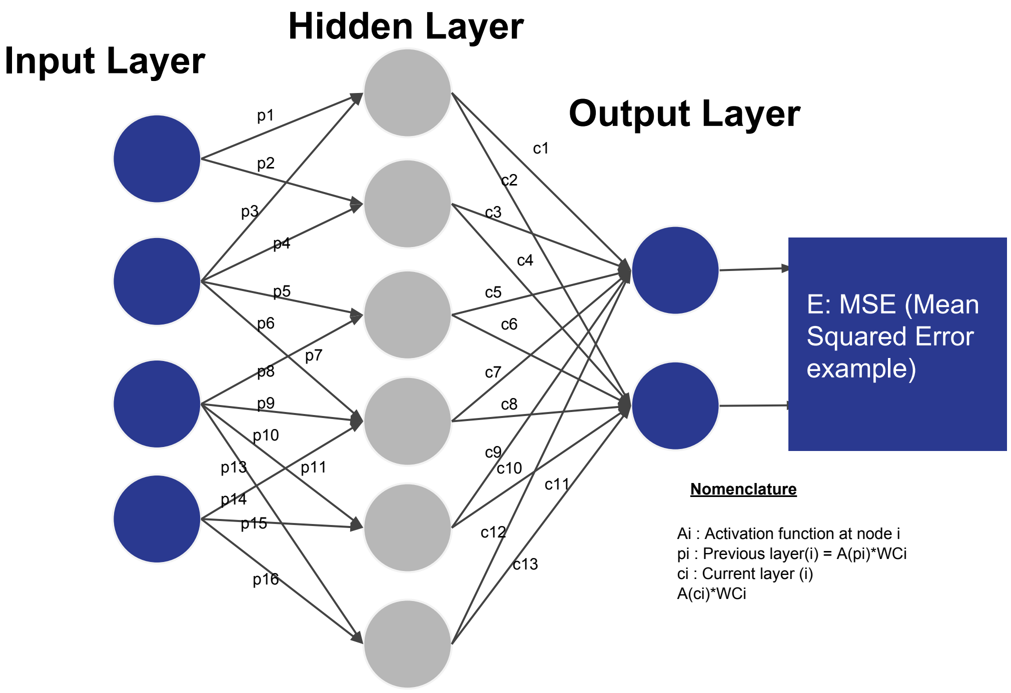
Stochastic Gradient Descent is a popular method to reduce the error function and fit the model closer to the data

- Efficient because it assesses one connection at a time (unlike batch gradient descent which tries the whole or a large sample of the network)
- Series of partial derivatives. Let's take one example:
  - Let's assume an error function that measures the deviance between observed(o) and expected(e)
  - Squared difference between o and e is loss function L - shuffle input
  - Derivative with respect to single weight wi is△L(wi)
  - A single step learning parameter lamda(lam) is introduced
  - The weight is re-adjusted to wi=w0 - lam*△L(wi)
  - Why negative gradient? We want to reduce the error!
- **This is done repetitively until a stopping criteria is reached**
- Some issues
  - Saddle points explored in Bengio
  - Overfitting

Nomenclature

Ai : Activation function at node i
pi : Previous layer(i) = A(pi)*WCi
ci : Current layer (i)
A(ci)*WCi

2b) Backpropagation is key to fitting models



Nomenclature

Ai : Activation function at node i
pi : Previous layer(i) = A(pi)*WCi
ci : Current layer (i)
A(ci)*WCi

Backpropagation Steps:

1) Initialize the weights at each pi and ci
2) Calculate the Error Value
3) Take a random connection ci
4) Peturb the weight, ci*wi, value by a small amount δ(ci)wi
5) Relate back to connected pi by derivative of activation function (δpi = A’(pi) Z wij δcj)
6) Re-calculate the Error Value
7) Repeat until a stopping criteria is activated

3a) Convolutional Neural Net with images in R

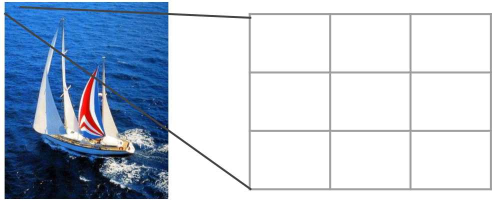
With images as an input we can make a simplifying assumption:

A **fully connected neural** net layer where every pixel in the image is connected to every other pixel for a 800 x 800 x 3 (RGB) = 1.92M individual weights. This is large

However we can use a **convolutional layer** to explore a local region with a filter and stack the results. Filters of NxNx3 can slide (convolve!) across the image to create a map of activations for different filters (RGB for example)

We then stack the activation maps of each filter to get the outcome

We can add more **depth (**For example, a filter that focuses on edges)



Our input is a 800*600 image of my my dream sailing yacht

EG: A 3x3 filter slides through each group of pixels and records the results of the function.

Input Volume: (W)
Depth: The number of filters (F)
Padding : A block of zeros around the image to control output volumes (P)
Stride: Movement step across convolution (S)

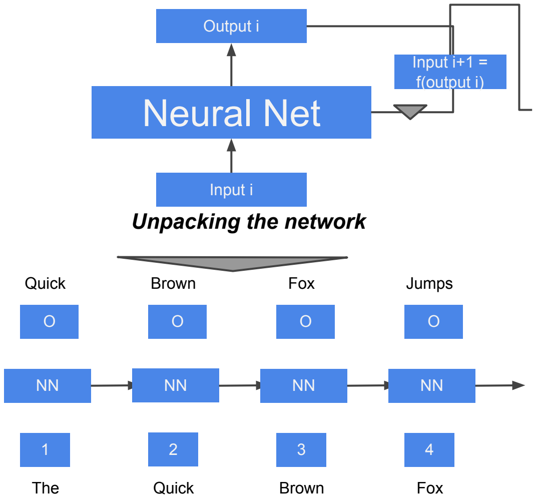
So with a 800x800 input and a F=3 size filter, a S=1 Stride and a P=1 block of zeroes:
We obtain an output volume of(W-F+2P)/S+1:
(800-3+2*1)/1+1= 800x800x3 as an output volume - **It retains the input volume**
Here we make a second assumption: we are looking at the same filter for each pixel. We can constrain the weights and bias for each filter to be the same. So the number of weights reduces to (3)x(3x3x3) = **81** weights

R Demo on MNIST

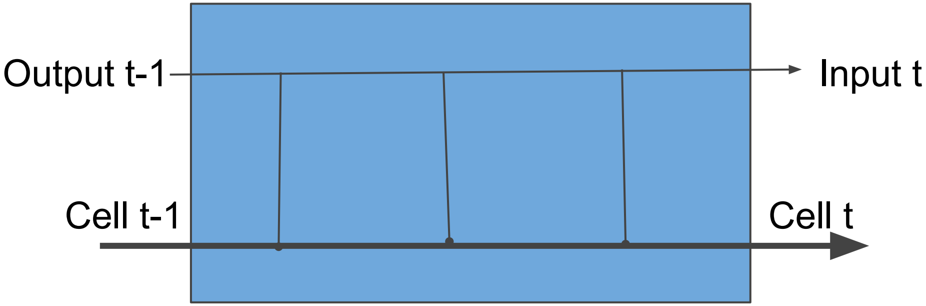
3b) RNN/LSTM with Text

Key issues: are vanishing or exploding gradients (Mikolov Bengio) and long term dependency(Bengio)

One Solution: Long Short Term Memory (Hochreiter) (LSTM) manages the information flow



LSTM Unit



Forget Gate: Controls how much of the previous cell we forget

Input Gate layer: control "How much of the new input to add"

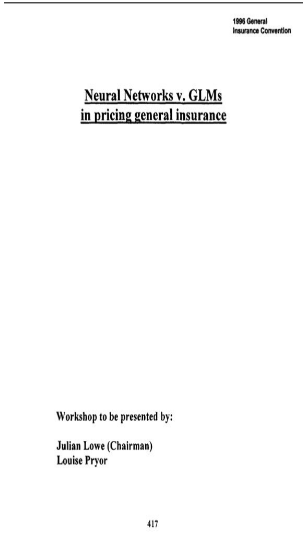
Filter Gate layer: control "How much of the new values to output"

RNN-LST
M R
Demo

4a) Deep Learning Actuarial Applications 1/2

Actuaries have been exploring neural networks for some time!

Some examples from Actuarial Lit or "Actuarial Neural Network" history:



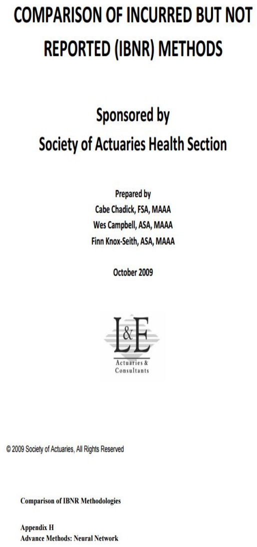
1996 General
Insurance Convention

Neural Networks v. GLMs
in pricing general insurance

Workshop to be presented by:

Julian Lowe (Chairman)
Louise Pryor

417



COMPARISON OF INCURRED BUT NOT
REPORTED (IBNR) METHODS

Sponsored by
Society of Actuaries Health Section

Prepared by
Cabe Chadick, FSA, MAAA
Wes Campbell, ASA, MAAA
Finn Knox-Seith, ASA, MAAA

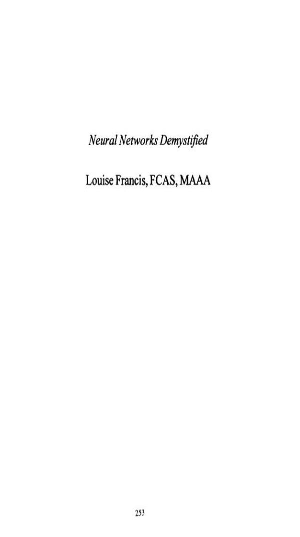
October 2009

L&E
Actuaries &
Consultants

© 2009 Society of Actuaries, All Rights Reserved

Comparison of IBNR Methodologies

Appendix H
Advance Methods: Neural Network



Neural Networks Demystified

Louise Francis, FCAS, MAAA

253

Major changes in the industry since these papers

- **Development** of open source software to build neural networks at scale
  - The Apache software foundation creating Spark, Storm and Cassandra and Hadoop
  - Python and R emerging as open source statistical and data munging programs with a vibrant community of developers
  - Data Science Community open sourcing code: TensorFlow from Google, Caffe, Keras, Torch, Theano
- **Hardware** and specifically GPUs that support parallel processing
- **Web services** such as Amazon AWS, Google, Microsoft Azure and Rackspace offering managed services
- **Data** especially **unstructured data** such as text holding value with an objective in mind

4a) Deep Learning Actuarial Applications 2/2

	Observed in Industry	Value add with DL in value chain	Benefits
Submission	Logistic models for underwriters for a final **(Yes/No) using thresholds**	Finer tuned and multinomial models detailing Yes/No/Potential to write with change as an example of multiple outcomes with actions attached	1) You own the model architecture 2) Can use increasingly dimensional and complex data being created 3) Finer tuning of model to objective (dangers of overfitting)
Pricing	Frequency-Severity or Pure Premium **parametric or non-parametric** (typically shallow ML) on historical data	Finer tuned multinomial pricing - company owns the architecture of the model	
Claims	Shallow Machine Learning **Triage** models using some n-gram text models to predict Severity	Deep NN architecture development that can combine text, sound, image and regular structured data - with valuable additional capabilities to absorb highly dimensional data	
Reserving	Chain Ladder,BF and Cape Cod at the AY-Dev period grain. Stochastic reserving using parametric methods	Claims grain with fine tuned cohort prediction across time, including RNN architectures that can estimate future states	
Fraud	Unsupervised clustering, network and graph analyses and shallow ML where target data available	Ability to combine data types and develop finely tuned suspicion models	

4b) Challenges and risks: Questions to ask

- What is the objective of the model and how does this solve the business problem?
- What pre-processing steps have you taken?
- Is there a resource constraint? (In terms of people, systems and time)
- Why does the architecture actually work? Have you tested other architectures?
- Are you collecting human decision data? How are you adjusting for bias?
- Is the model overfitting? How have you controlled for this?
- Does the model need to be explainable to a business?
- How much upkeep is needed?
- Has ensembling with another model improved results?