PGE 383 Lecture xx
Neural Networks

- Neural Networks
- Neural Network Example

Michael Pyrcz, The University of Texas at Austin

Introduction

Prerequisites

Data Preparation

Univariate Analysis

Multivariate Analysis

Spatial Characterization

Spatial Estimation

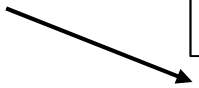
Spatial Simulation

Uncertainty Analysis

Model Checking

Decision Making

Machine Learning



PGE 383 Lecture xx
Neural Networks

- Neural Networks

Michael Pyrcz, The University of Texas at Austin

Machine Learning

Introduction

Prerequisites

Data Preparation

Univariate Analysis

Multivariate Analysis

Spatial Characterization

Spatial Estimation

Spatial Simulation

Uncertainty Analysis

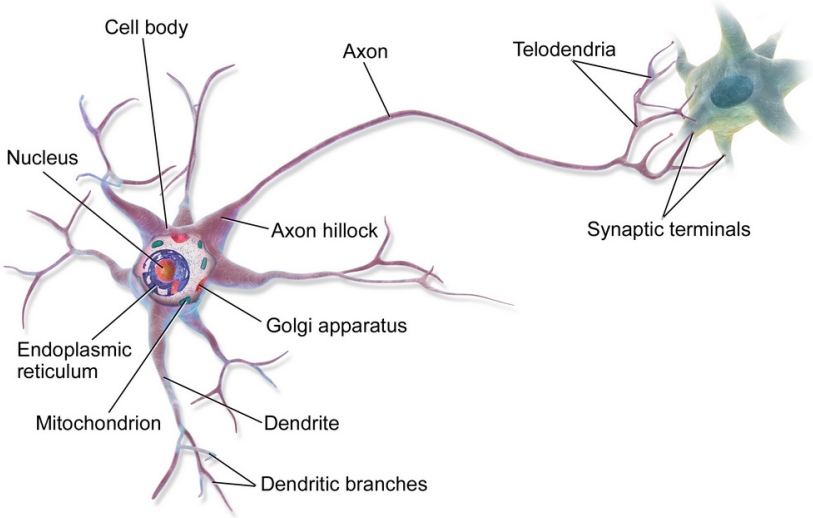
Model Checking

Decision Making

Natural Computing

Looking to nature for inspiration to develop novel problem solving methods.

- artificial neural networks are inspired by biological neural networks
- the nodes in our model are artificial neurons
- the connections are artificial synapses



- intelligence emmerges from many connected processors

Image from https://commons.wikimedia.org/wiki/File:Blausen_0657_MultipolarNeuron.png

Neural Nets

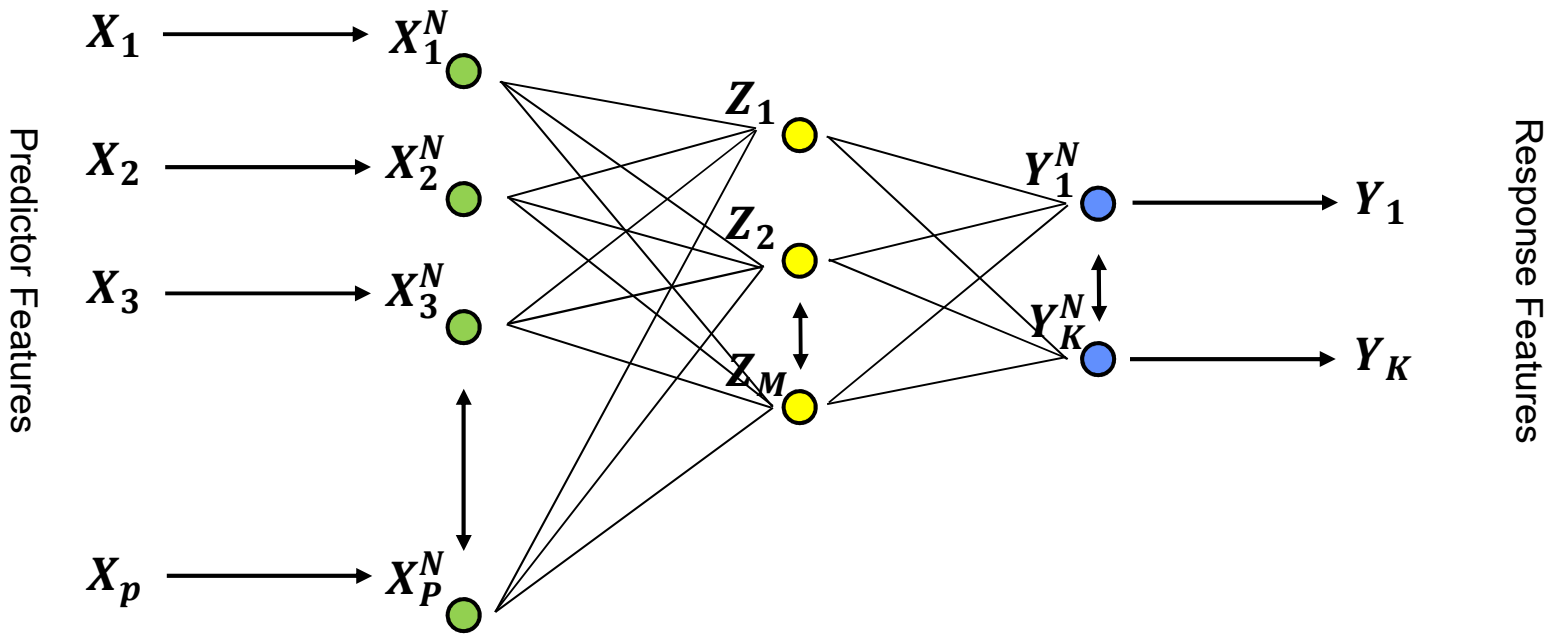
We want a prediction system

$$Y = f(X)$$

- Supervised learning – we will provide training data with predictor features, $X_1, \ldots, X_P$ and response features $Y_1, \ldots, Y_K$
- Nonlinear - that can capture / predict with complicated features

Neural Nets

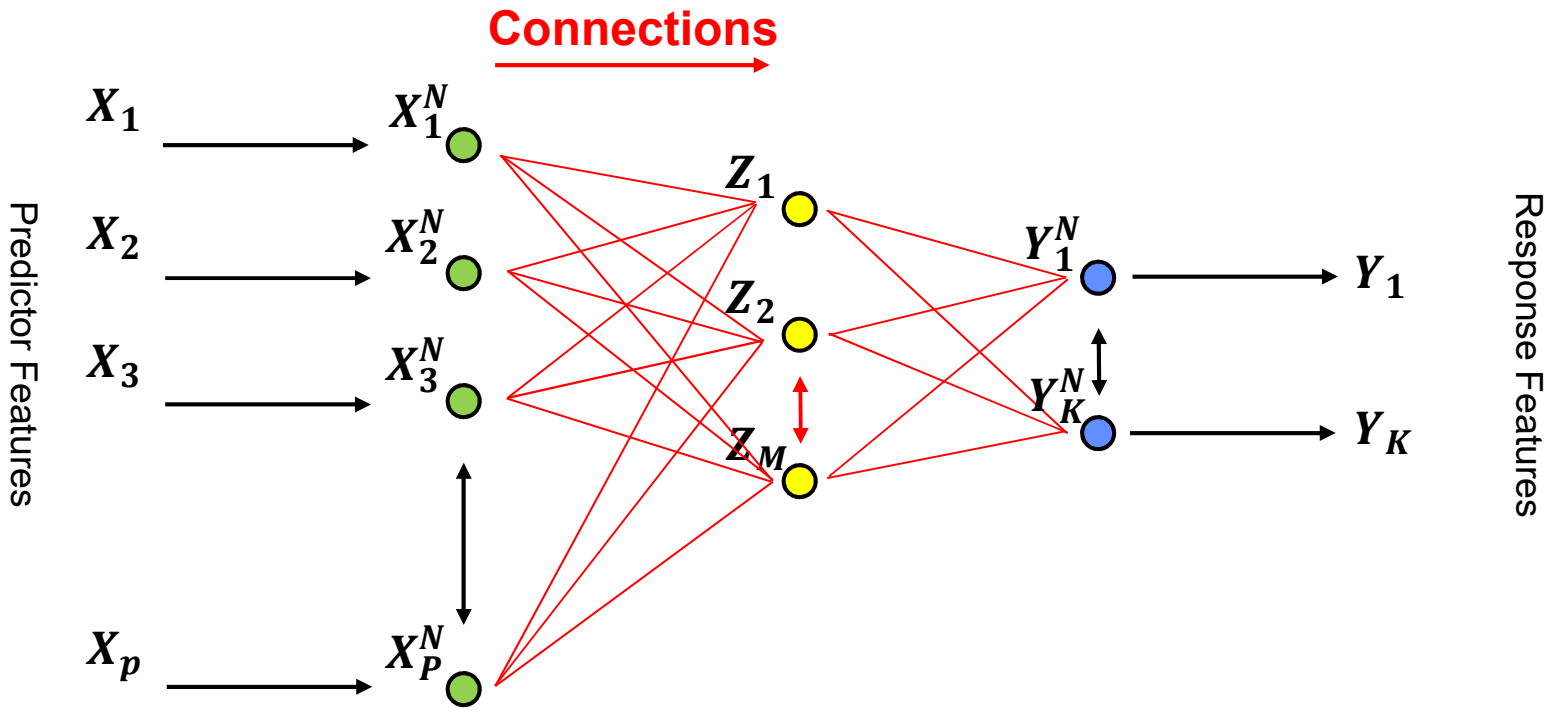
- Let's build a neural net, **single hidden layer, feed-forward neural network** to accomplish this.



- **Feed-forward** – all information flows from left to right.

Neural Nets

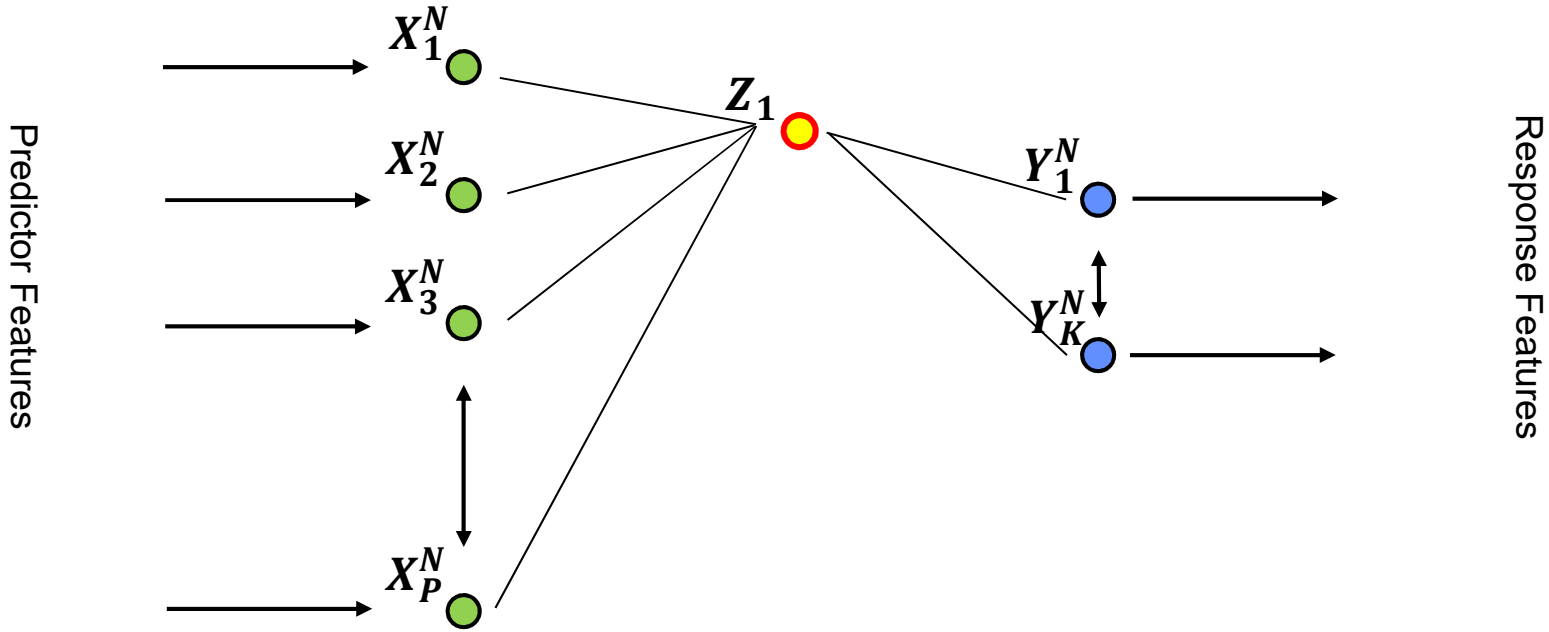
- **Connections**: output from a node is passed to nodes in the next layer as inputs.



Neural Nets

Nodes: take linearly weighted combinations of inputs and then nonlinearly transform the result.

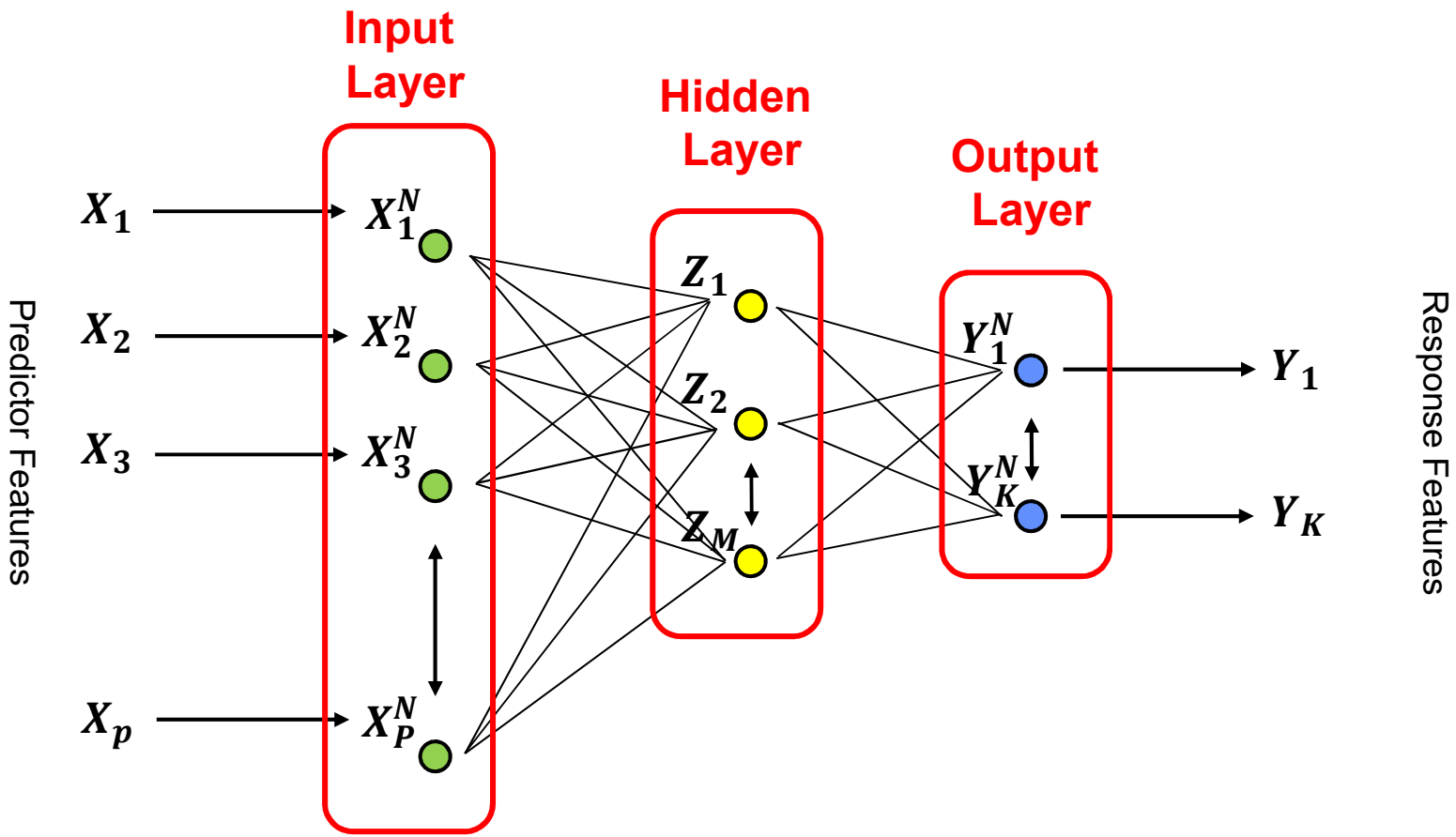
- A very simple processor!
- Through a large number of interconnected nodes we get emergent prediction of complicated patterns.



Neural Nets

Layers: for our feed-forward design we organize the nodes into layers.

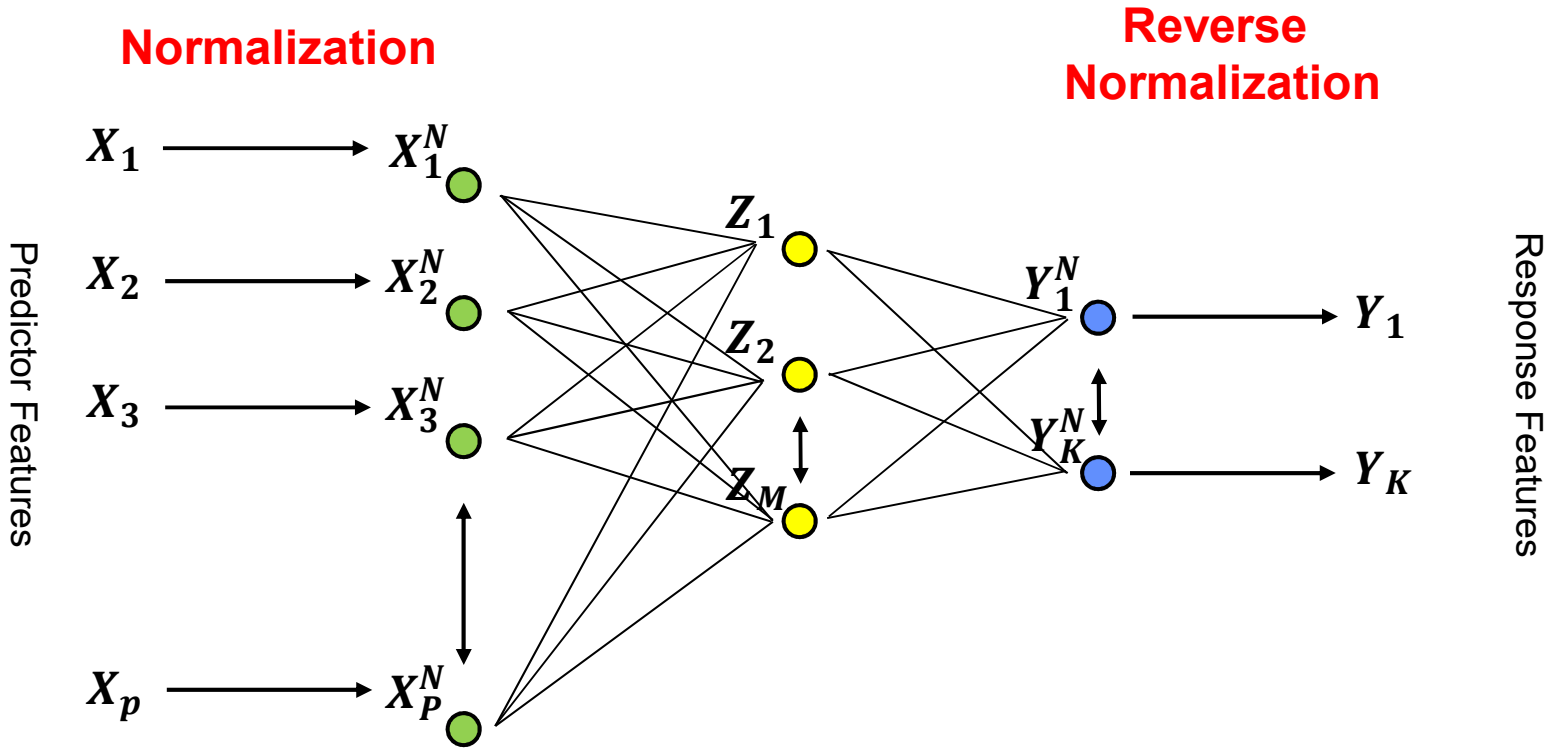
- **Deep Learning** is the use of more than one hidden layer



Neural Nets

Normalization: transforming the features to a specific range (commonly -1 ~ 1), centered on 0.0.

- remove the influence of scale differences in predictor features
- improve activation function sensitivity



- predictions are in normalized response features, then backtransform.

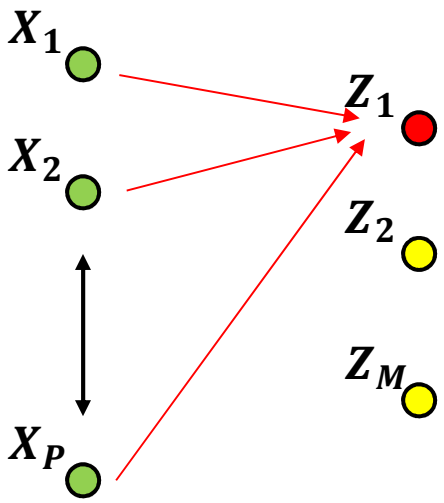
Neural Nets

Now let's walk through our neural network, let's visit a single node in our hidden layer, Z_1

Node result at Z_1, as a function of a linear combination of the inputs, X_p

$$Z_1 = f\left(\alpha_{0,1} + \alpha_1^T X\right)$$

Where α_m^T are weights and α_{0m} is a bias term that are fit with training data.



Weights

Bias

$$\alpha_1 + \sum_{p=1}^{P} \alpha_p X_p$$

a linear combination of the inputs $X_1,\ldots, X_p$ hidden layer node 1

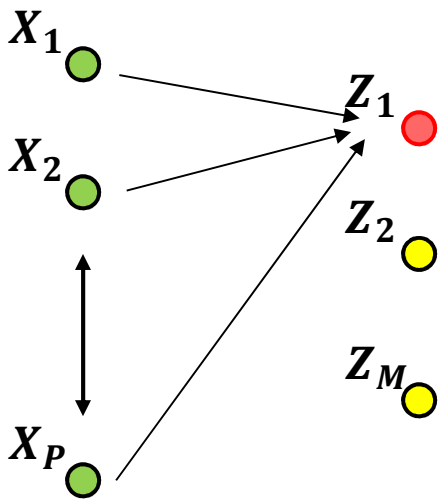
Neural Nets

Now let's track through our neural network, let's visit a single node in our hidden layer, Z_1

Z_1, is the result of the activation function, σ, applied to the linear combination of the inputs, X_p

$$Z_1 = \sigma(\alpha_{0m} + \alpha_m^T X)$$

Where σ is a user defined activation function.



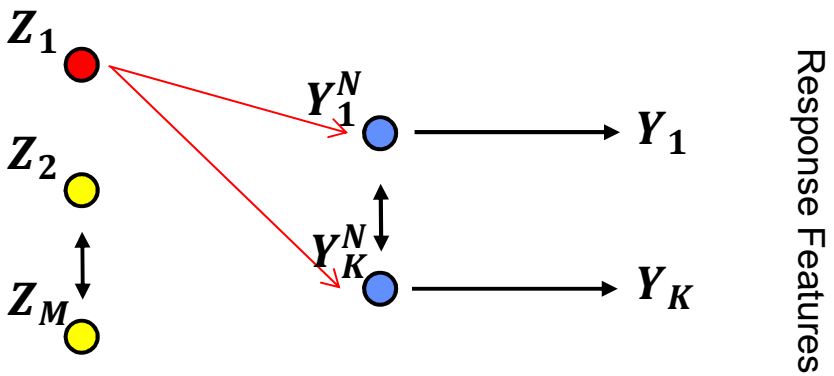
$$Z_1 = \sigma\left(\alpha_1 + \sum_{p=1}^{P} \alpha_p X_p\right)$$

nonlinear transform of a linear combination of the inputs $X_1,\ldots, X_p$ hidden layer node 1

Neural Nets

Now let's track through our neural network, let's continue to the output layer.

The result from the activation function, Z_1, is then passed to the next layer.

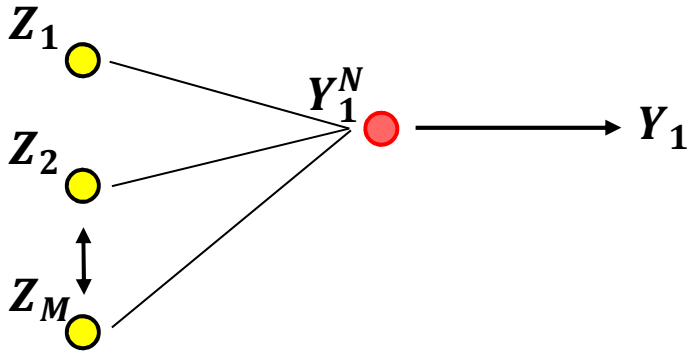


Neural Nets

Now let's track through our neural network, let's focus on a single node in the output layer, Y_1^N.

At the output layer node, Y_1^N, the outputs from the hidden layer nodes are linearly weighted and transformed by ouput function, g_k.

$$Y_1^N = g_k\left(\beta_1 + \sum_{m=1}^{M} \beta_m Z_m\right)$$



Response Features

For regression:

$$g_k(T) = T$$

For classification:

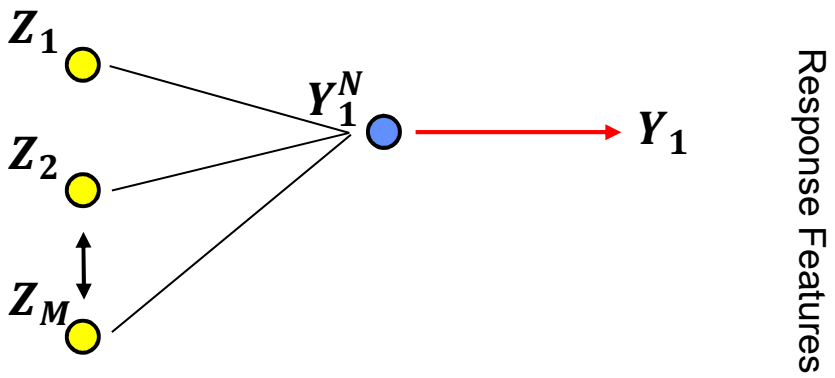
$$g_k(T) = \frac{e^{T_k}}{\sum_{\iota=1}^{K} e^{T_\iota}}$$

forces positive and sum to one

Neural Nets

Now let's track through our neural network, now we have a result!

The output at the output layer node, Y_1^N, is back transformed to the original units of the response feature, Y_1.



Neural Nets Parameters

The Parameters Trained in the Neural Network

For every connection there is a weight:

$\alpha_{p,m}$ - hidden layer

$\beta_{m,k}$ - output layer

with full connectivity the number of weights is $p \times m$ and $m \times k$!

At each node there is a bias term (the constant)

α_m and β_k

one at each hidden layer node and output layer node.

Neural Nets
Activitation Functions

The Activation Function is a nonlinear transform of the linear combination of the inputs to a node.

- introduce non-linear properties to the network
- without the activation function we would have linear regression
- increases the power to learn complicate patterns

Universal Function Approximator

- ability to learn any possible function.

Neural Nets Activitation Functions

Here's some examples of activation functions:

- Sigmoid or Logistic

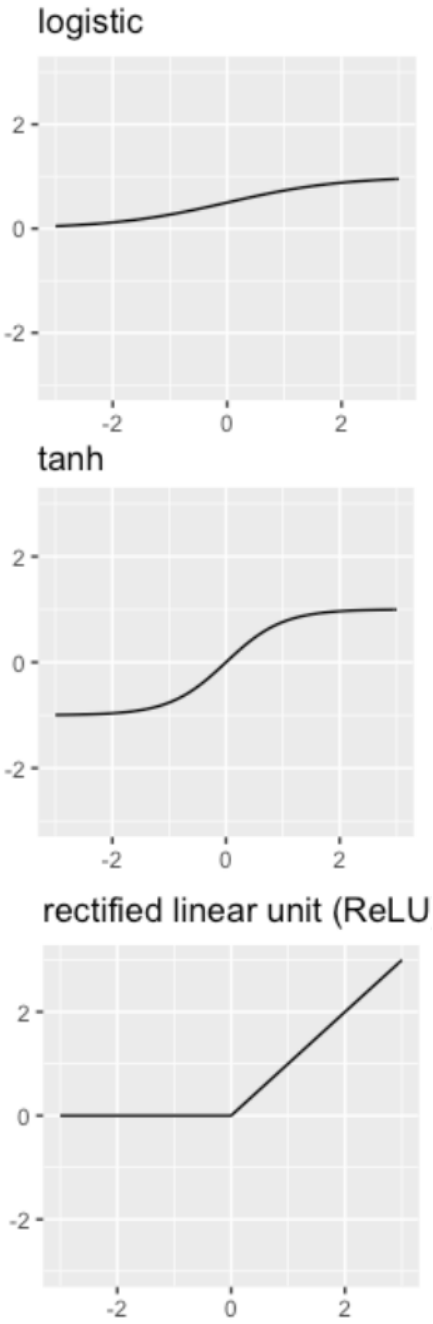
$$f(x) = \frac{1}{1 + \exp(-x)}$$

- Tanh – hyperbolic tangent

$$f(x) = \frac{1 - \exp(-2x)}{1 + \exp(-2x)}$$

- ReLu – rectified linear units

$$f(x) = \max(0, x)$$



Images from http://uc-r.github.io/ann_fundamentals

Neural Nets Training

How Do We Train the Neural Net?

- Recall the parameters are the connection weights and the node biases.

1. Set the neural net parameters to random values
2. Forward Pass – run our neural net with the training data

$$\overrightarrow{\hat{y}} = f(X, \overrightarrow{\alpha_{p,m}}, T)$$

T is the other assumptions like activation functions and number of nodes and layers, we have assume identity function for output layer,

Neural Nets Training

Define a Performance or Loss function, e.g. L_2 Loss

- We could consider the sum of the square error over the training data

$$P = L_2 = \sum_{\alpha=1,..,n} \left(\vec{\hat{y}} - \vec{y}\right)^2$$

- We could also consider the average square error over the training data

$$E\{L_2\} = \frac{1}{n} \sum_{\alpha=1,..,n} \left(\vec{\hat{y}} - \vec{y}\right)^2$$

Neural Nets Training

Calculate the Partial Derivatives for the Performance Given Each Weight.

$$\vec{w} = \frac{\partial P}{\partial \alpha_{p,m}}$$

for all weights, $\alpha_{p,m}$, over all $p = 1, \ldots, P \times \ m = 1, \ldots, M$ connections.

We now have a gradient, the change to the individual weights is scaled by a learning rate, r.

$$\Delta\vec{w} = r * \frac{\partial P}{\partial \alpha_{p,m}}$$

- This process of gradient calculation and weight updating is iterated.
- This is known as **back-propogation**.

Neural Nets Training

The Training is Iterative

Epochs: each forward pass – back-propogation, training cycle, is known as an Epoch.

Batch: the number of training data considered in each epoch.

Comments:

Larger batches result in better estimates of the error, but slower epochs (more computational time)

Smaller batches result in noiser estimates of the error, but faster epochs

- often results in faster learning and even possibly more robust models

Neural Nets Training

Finding the Best Weights. We Need to Avoid Local Minimums by Setting these Hyperparameters.

Learning Rate:

The r hyperparmeter that controls the rate of weight updating in response to the gradient of model fit relative to the weight.

$$\Delta\vec{w} = r * \frac{\partial P}{\partial \alpha_{p,m}}$$

Mommentum:

The integration of memory from the previous update.

$$\boldsymbol{v_{i+1} = \alpha v_i + \theta_{i+1}}$$

where v_i is the previous update vector, α, is the momentum parameter, θ_{i+1}, is the new update and v_{i+1} is the new update vector.

Neural Nets Design

Hyperparameters:

Learning Rate, r , rate of weight adjustment in back propagation

Mommentum, α, memory in weight adjustment in back propagation

Performance metric, P – measure of goodness of the predictions

Width of the ANN – the number of nodes in each layer

Depth of the ANN – the number of hidden layers

Activation Functions – hidden layers

Output Function – output layer

Neural Nets Design

Limitations of Neural Nets

No Free Lunch Theorem – cannot garauntee the model is always optimum for predicting new, unobserved cases.

Parameter Rich – requires a large number of training data.

Low Interpretability – generally difficult to interrogate the model, not compact.

High Complexity Model – resulting in high model variance and lower model bias.

Neural Nets Variants

There are a variety of designs based on the single hidden layer, feed-forward design that we reviewed.

Deep Learning – use of multiple hidden layers

Recurrent Neural Networks – information can flow backwards

Convolutional Neural Networks – accounts for 2D / 3D information

Auto Encoder – dimensionality reduction

PGE 383 Lecture xx
Neural Networks

- **Neural Network Example**

Michael Pyrcz, The University of Texas at Austin

Machine Learning

Introduction

Prerequisites

Data Preparation

Univariate Analysis

Multivariate Analysis

Spatial Characterization

Spatial Estimation

Spatial Simulation

Uncertainty Analysis

Model Checking

Decision Making

Neural Nets
Example in Python

We will use a well and seismic data to build a neural net-based predictive model.

Train with 80% of well data (144 wells)

Test with 20% of well data.

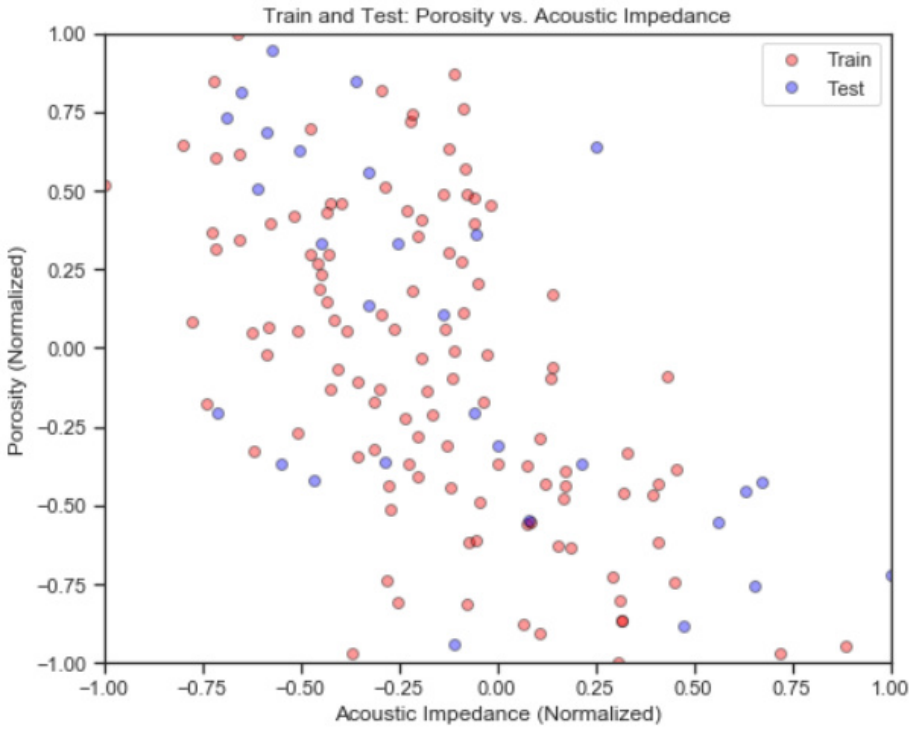
Application: Apply the model to predict porosity from a acoustic impedance map!

between well estimation of porosity in a single unit

Neural Nets
Example in Python

Normalize the variables to range from -1 to 1.

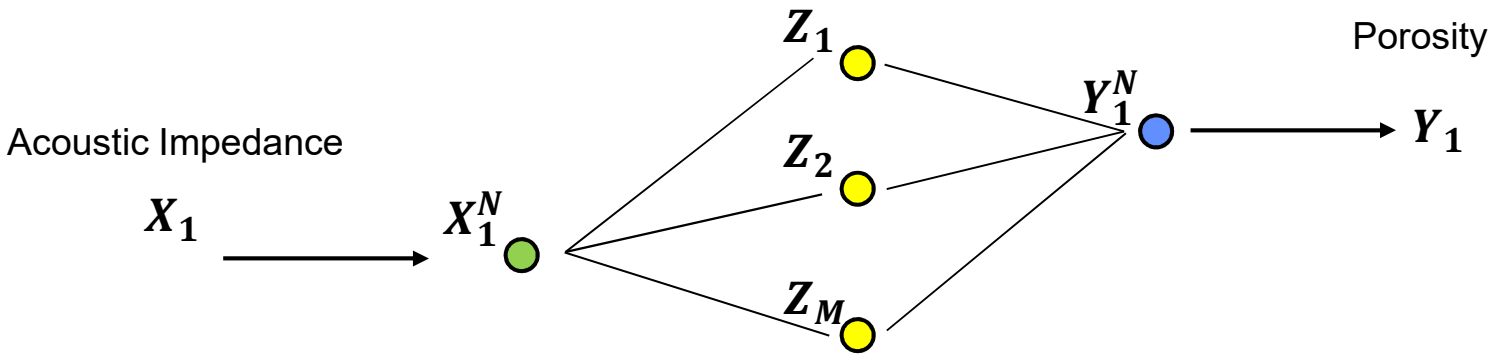
- the relationship between acoustic impedance and porosity.



Neural Nets
Example in Python

Design a Simple Neural Network

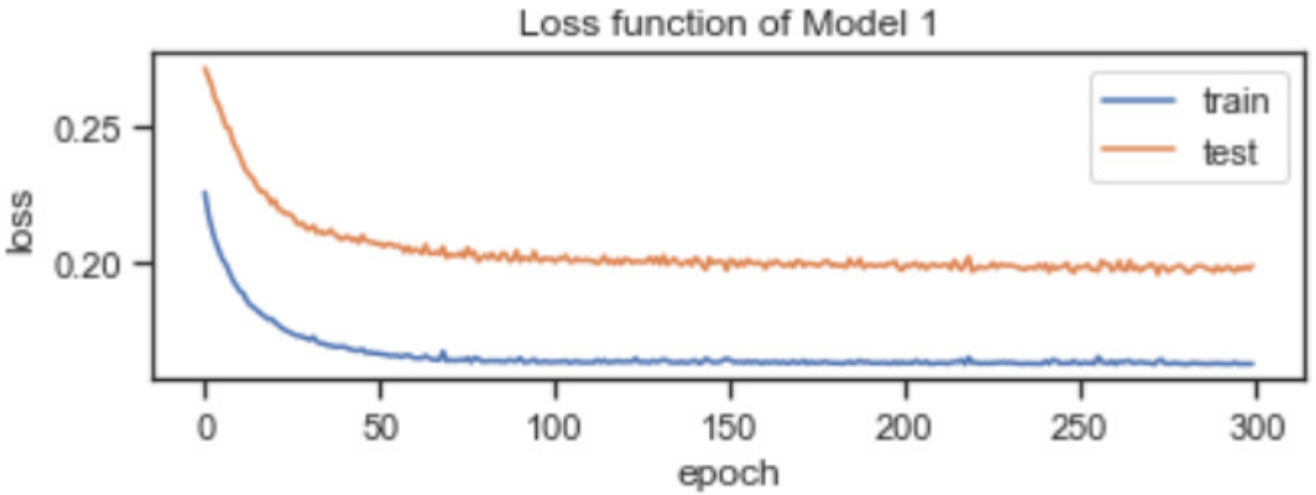
- 1 hidden layer, 3 nodes in hidden layer
- 1 node in input and output layer
- ReLu activation functions on hidden layer nodes



Neural Nets
Example in Python

Training a Simple Neural Network

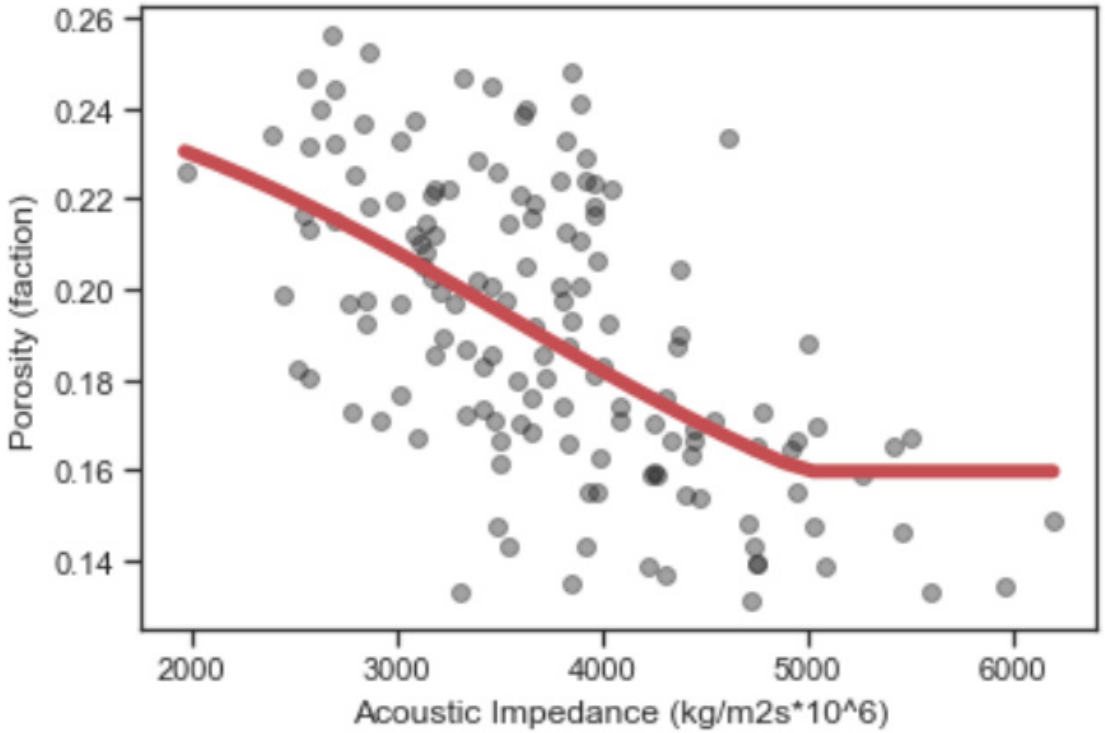
- 300 epochs with batches of 5
- train and test loss



Neural Nets
Example in Python

Model Preditions

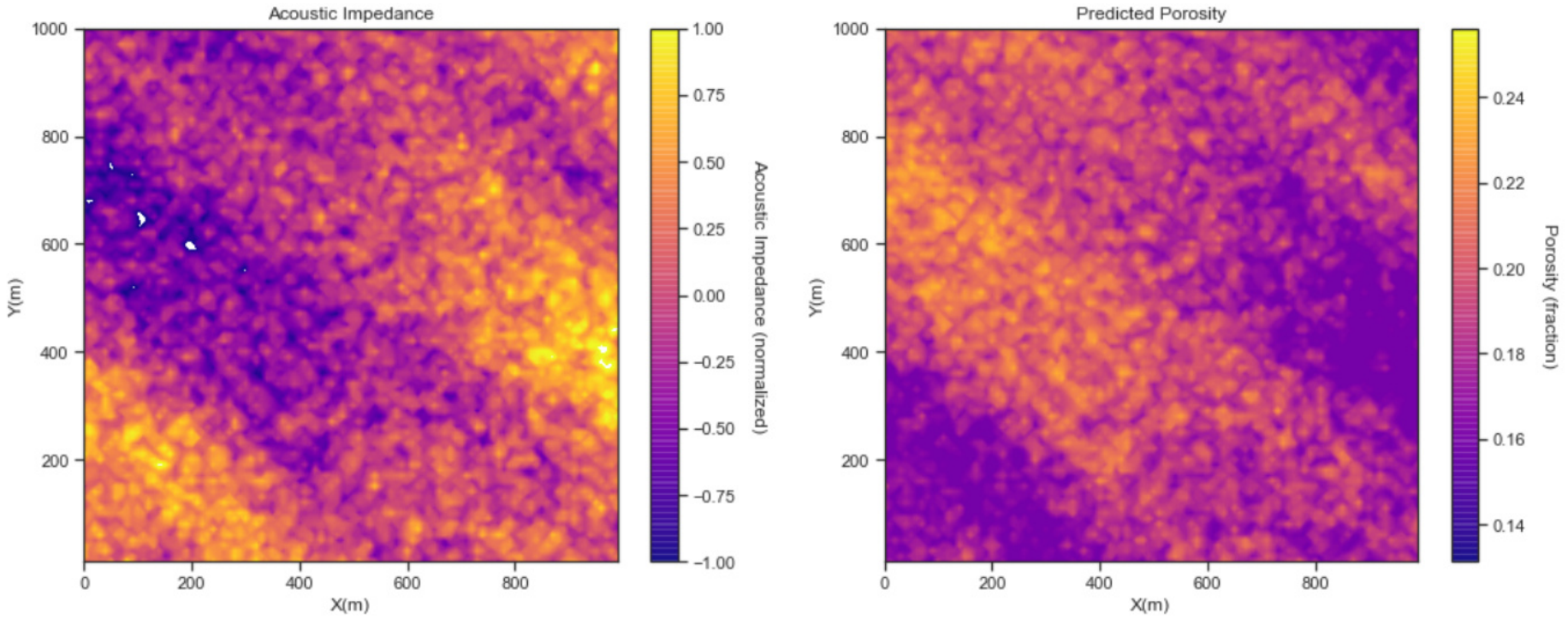
- the model predictions with all training and testing data
- demonstrates the ability to fit non-linear data



Neural Nets Example in Python

Model Preditions

- prediction of porosity from acoustic impedance between well locations.



Neural Net Demonstration

Demonstration workflow with artificial neural network for supervised learning from training data.

Subsurface Data Analytics

Artificial Neural Networks for Prediction in Python

Honggeun Jo, Graduate Student, The University of Texas at Austin

LinkedIn | *Twitter*

Michael Pyrcz, Associate Professor, University of Texas at Austin

Twitter | *GitHub* | *Website* | *GoogleScholar* | *Book* | *YouTube* | *LinkedIn* | *GeostatsPy*

PGE 383 Exercise: Support Vector Machine for Subsurface Modeling in Python

Here's a simple workflow, demonstration of neural networks for subsurface modeling workflows. This should help you get started with building subsurface models that use data analytics and machine learning. Here's some basic details about neural networks.

Neural Networks

Machine learning method for supervised learning for classification and regression analysis. Here are some key aspects of support vector machines.

Basic Design *"...a computing system made up of a number of simple, highly interconnected processing elements, which process information by their dynamic state response to external inputs."* Caudill (1989).

Nature-inspire Computing based on the neuronal structure in the brain, including many interconnected simple, processing units, known as nodes that are capable of complicated emergent pattern detection due to a large number of nodes and interconnectivity.

Training and Testing just like and other predictive model (e.g. linear regression, decision trees and support vector machines) we perform training to fit parameters and testing to tune hyper parameters.

File SubsurfaceDataAnalytics_NeuralNet.ipynb at https://git.io/fjlao.

PGE 383 Lecture xx
Neural Networks

- Neural Networks
- Neural Network Example

Michael Pyrcz, The University of Texas at Austin

Introduction

Prerequisites

Data Preparation

Univariate Analysis

Multivariate Analysis

Spatial Characterization

Spatial Estimation

Spatial Simulation

Uncertainty Analysis

Model Checking

Decision Making

Machine Learning