Deepnets

Modeling with Linear Algebra

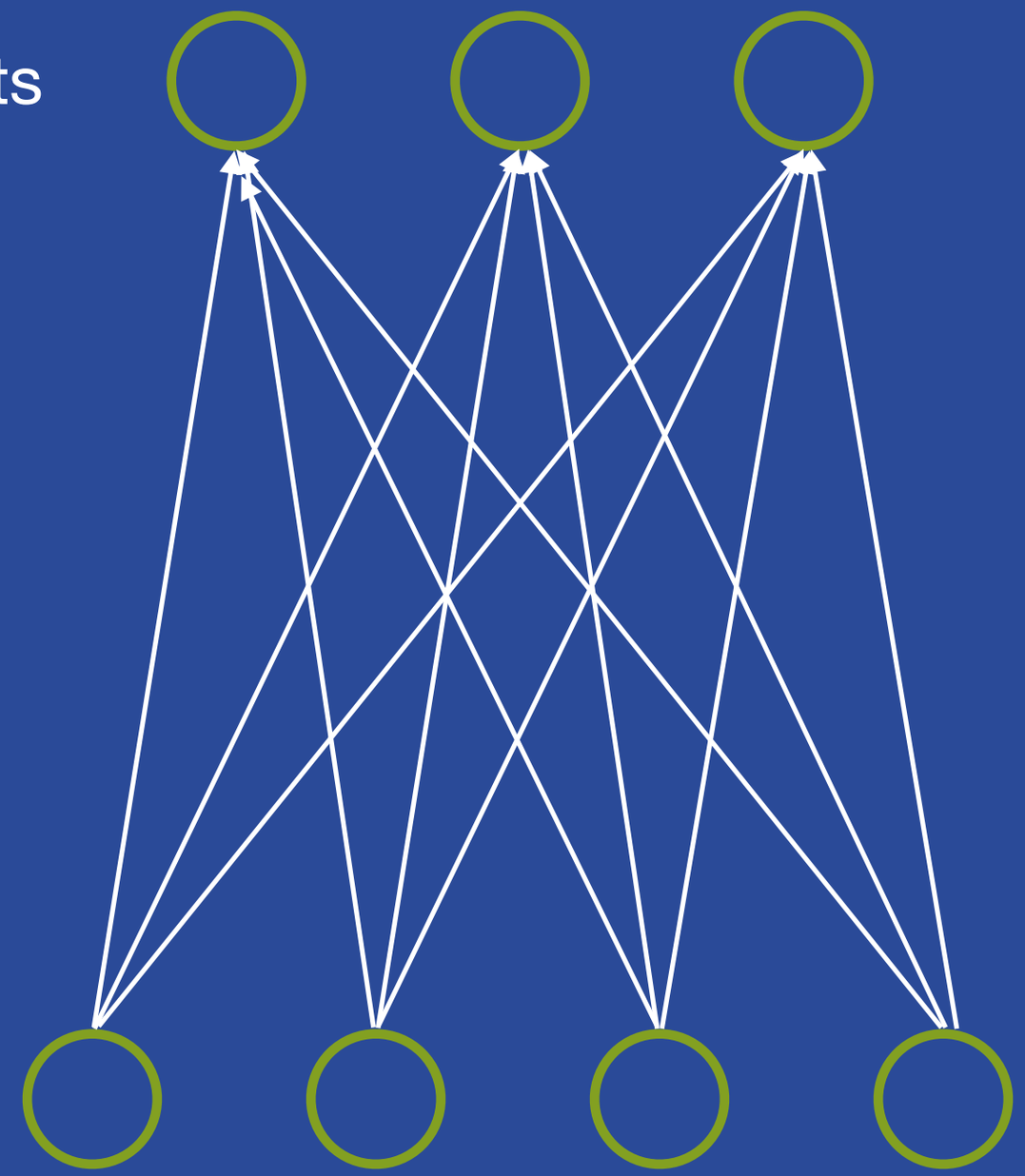
Charles Parker
VP ML Algorithms, BigML, Inc

What is / is not a Deepnet?

- What is a Deepnet?
  - Supervised learning algorithm
  - Classification and regression
  - Logistic regression leveled up

Logistic Level Up

Outputs

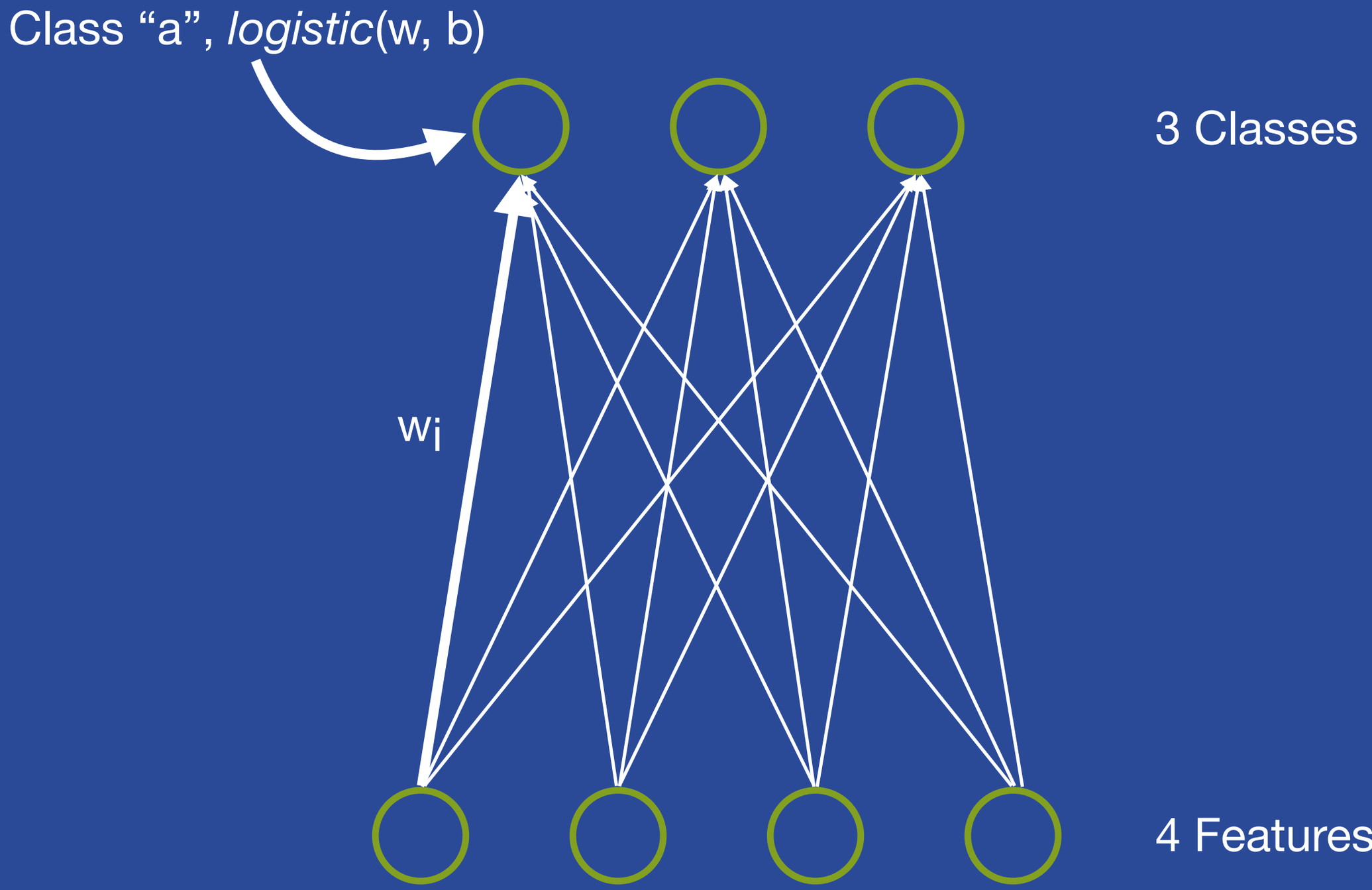


3 Classes

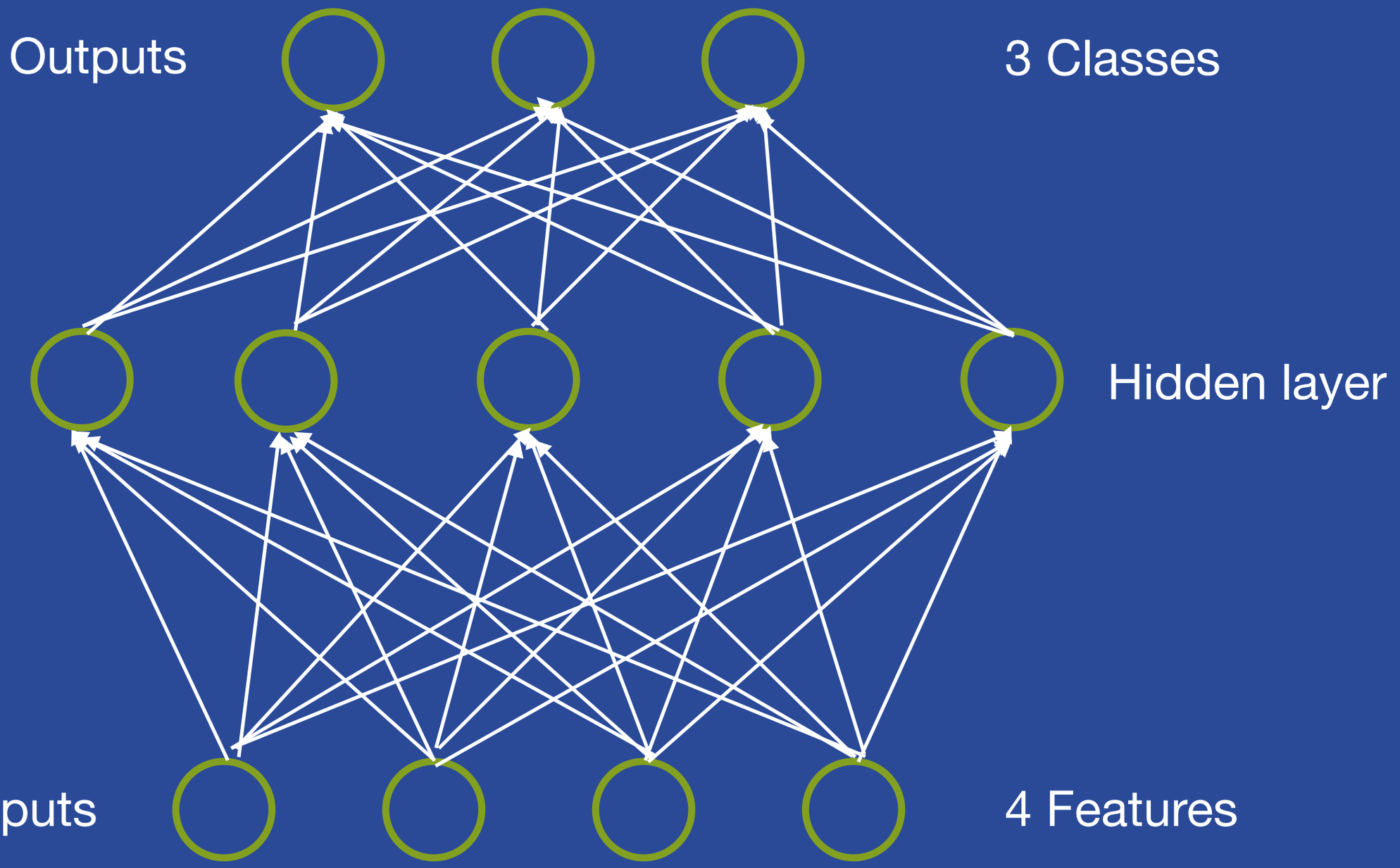
Inputs

4 Features

Logistic Level Up

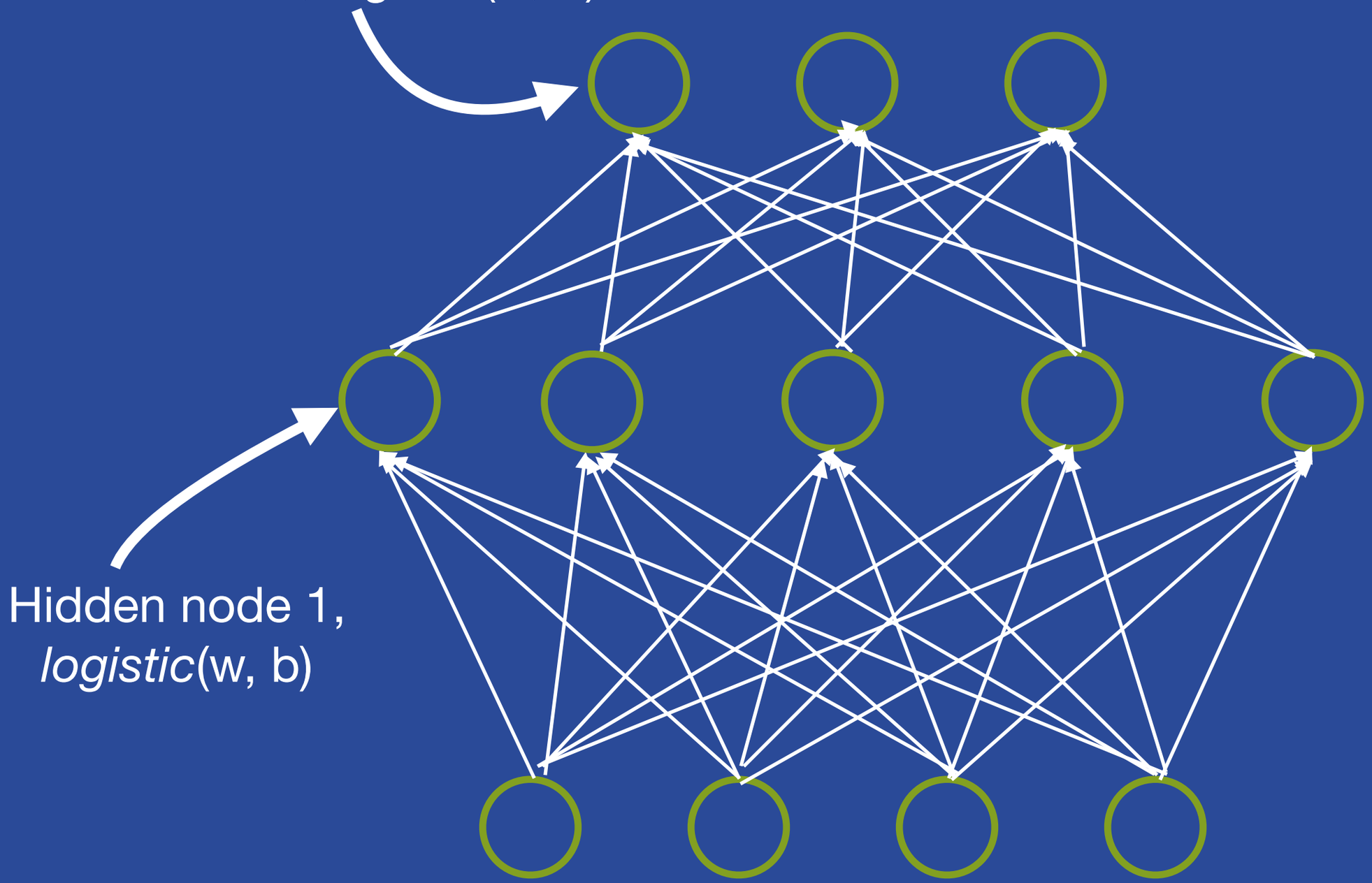


Deepnet



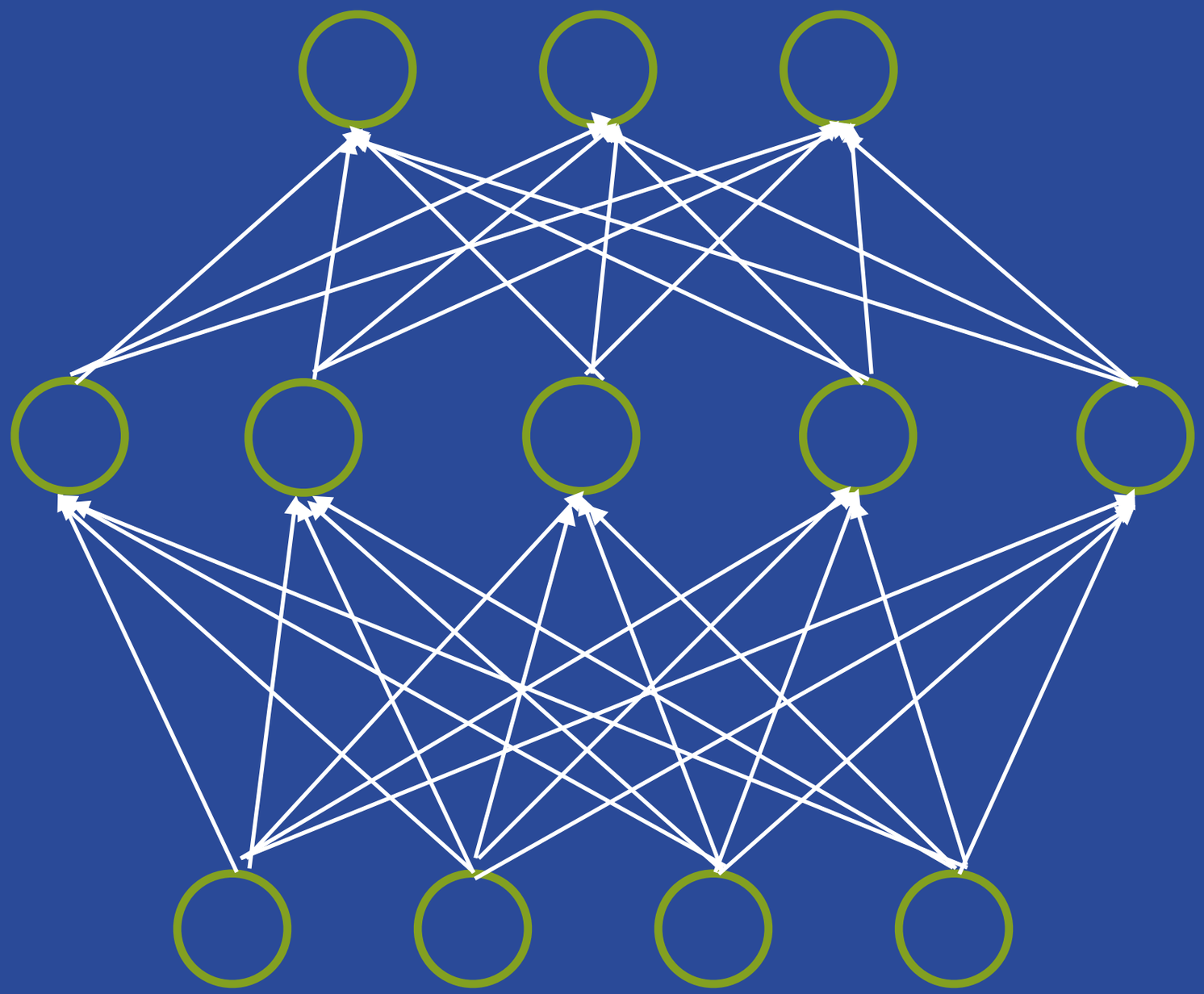
Deepnet

Class “a”, *logistic*(w, b)



Hidden node 1,
logistic(w, b)

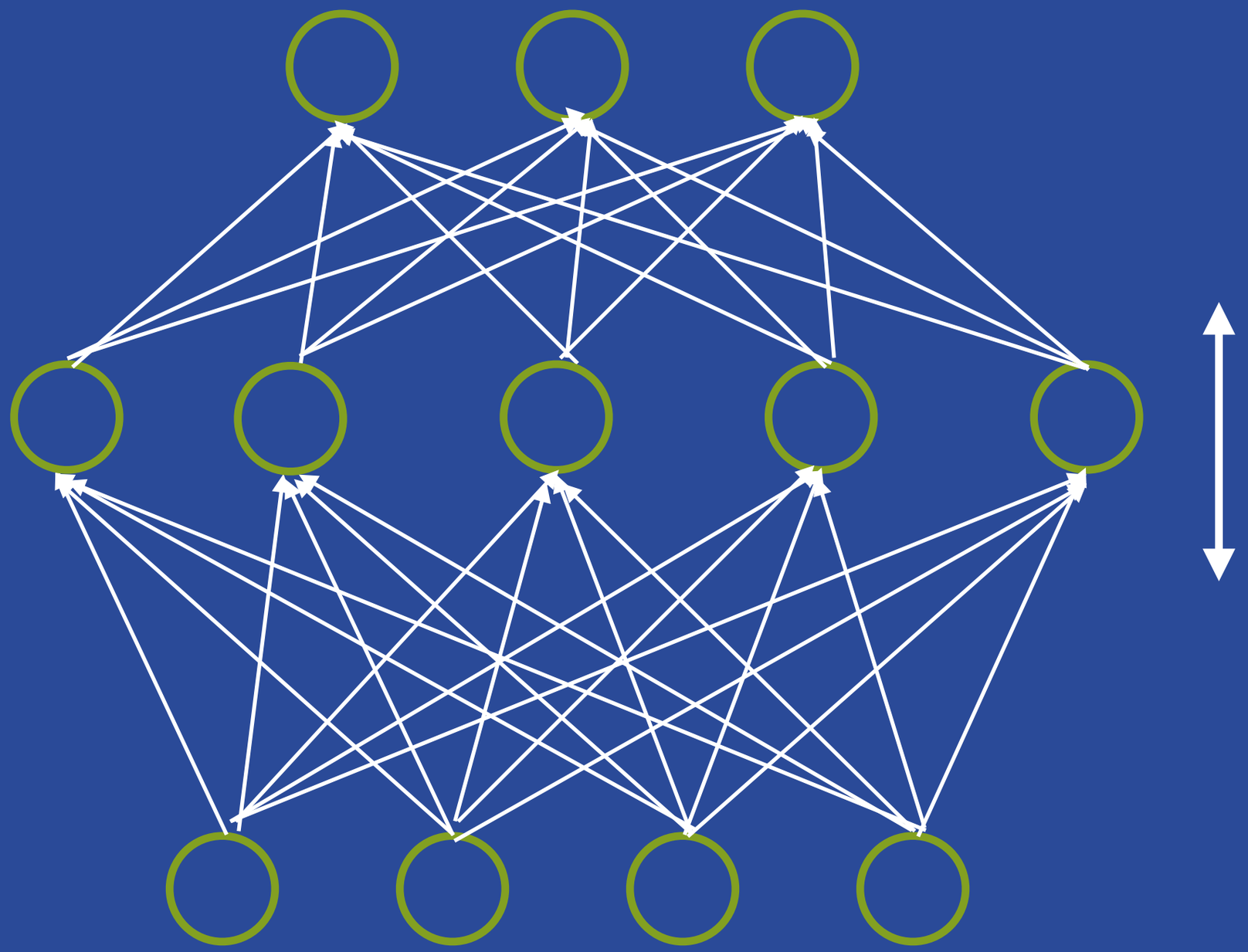
Deepnet



n

hidden nodes?

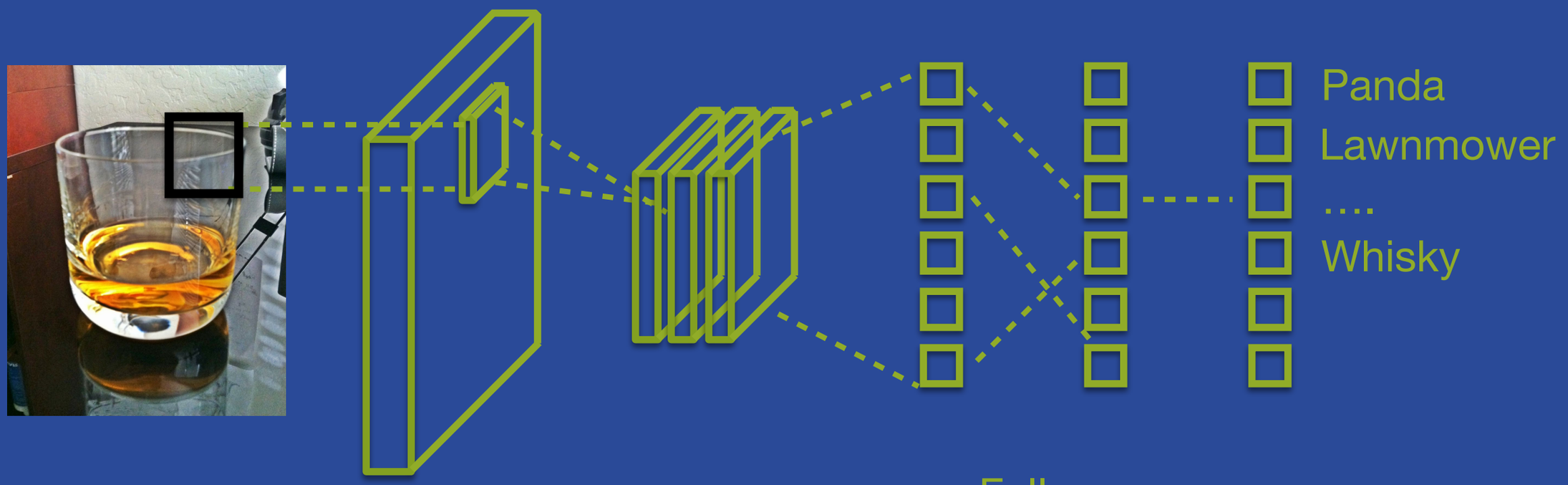
Deepnet



What is / is not a Deepnet?

- What is a Deepnet?
  - Supervised learning algorithm
  - Classification and regression
  - Logistic regression leveled up
- What is a Deepnet NOT?
  - A Convolutional Neural Network

Convolutional NN



- A CNN is a special Deepnet architecture which is particularly adept at classifying images
- BigML does not support these… yet…

What is / is not a Deepnet?

- What is a Deepnet?
  - Supervised learning algorithm
  - Classification and regression
  - Logistic regression leveled up
- What is a Deepnet NOT?
  - A Convolutional Neural Network
  - A Recurrent Neural Network or LSTM - these are special architectures that are adept at NLP and speech recognition
  - The last ML algorithm you will ever need…

Why Deepnets?

- **Trees**
  - Pro: Massive representational power that expands as the data gets larger; efficient search through this space
  - Con: Difficult to represent smooth functions and functions of many variables
  - Ensembles mitigate some of these difficulties
- **Logistic Regression**
  - Pro: Some smooth, multivariate, functions are not a problem; fast optimization of chosen
  - Con: Parametric - If decision boundary is nonlinear, tough luck

Why Deepnets?

- Like Trees / Ensembles, we have arbitrary representational power by modifying the structure
- Like logistic regression, smooth, multivariate objectives aren't a problem (provided we have the right structure)
- So what have we lost?
  - Interpretability: We've gotten quite far away from the interpretability of trees
  - Ease of use: The right structure for a given dataset is not easily found
  - And most structures are bad!

Parameter Paralysis

Parameter Name	Possible Values
Descent Algorithm	Adam, RMSProp, Adagrad, Momentum, FTRL
Number of hidden layers	0 - 32
Activation Function (per layer)	relu, tanh, sigmoid, softplus, etc.
Number of nodes (per layer)	1 - 8192
Learning Rate	0 - 1
Dropout Rate	0 - 1
Batch size	1 - 1024
Batch Normalization	True, False
Learn Residuals	True, False
Missing Numerics	True, False
Objective weights	Weight per class

. . . and that's ignoring the parameters that are specific to the descent algorithm.

Deepnets Demo #1

Let's use ML

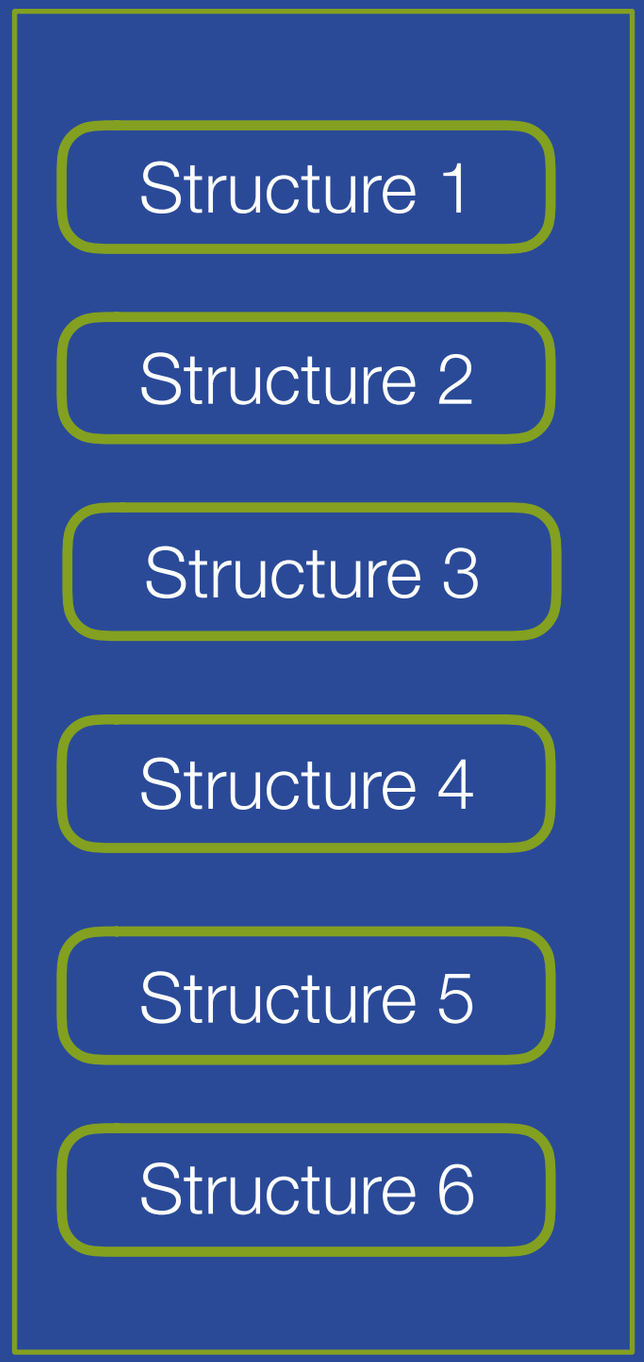
- Deepnet Problem:
  - The success of a Deepnet is dependent on getting the right network structure for the dataset
  - But, there are too many parameters…
  - And setting them takes significant expert knowledge
- Solution:
  - Metalearning (a good initial guess)
  - Network search (try a bunch)

Metalearning

We trained 296,748 deep neural networks
so you don't have to!

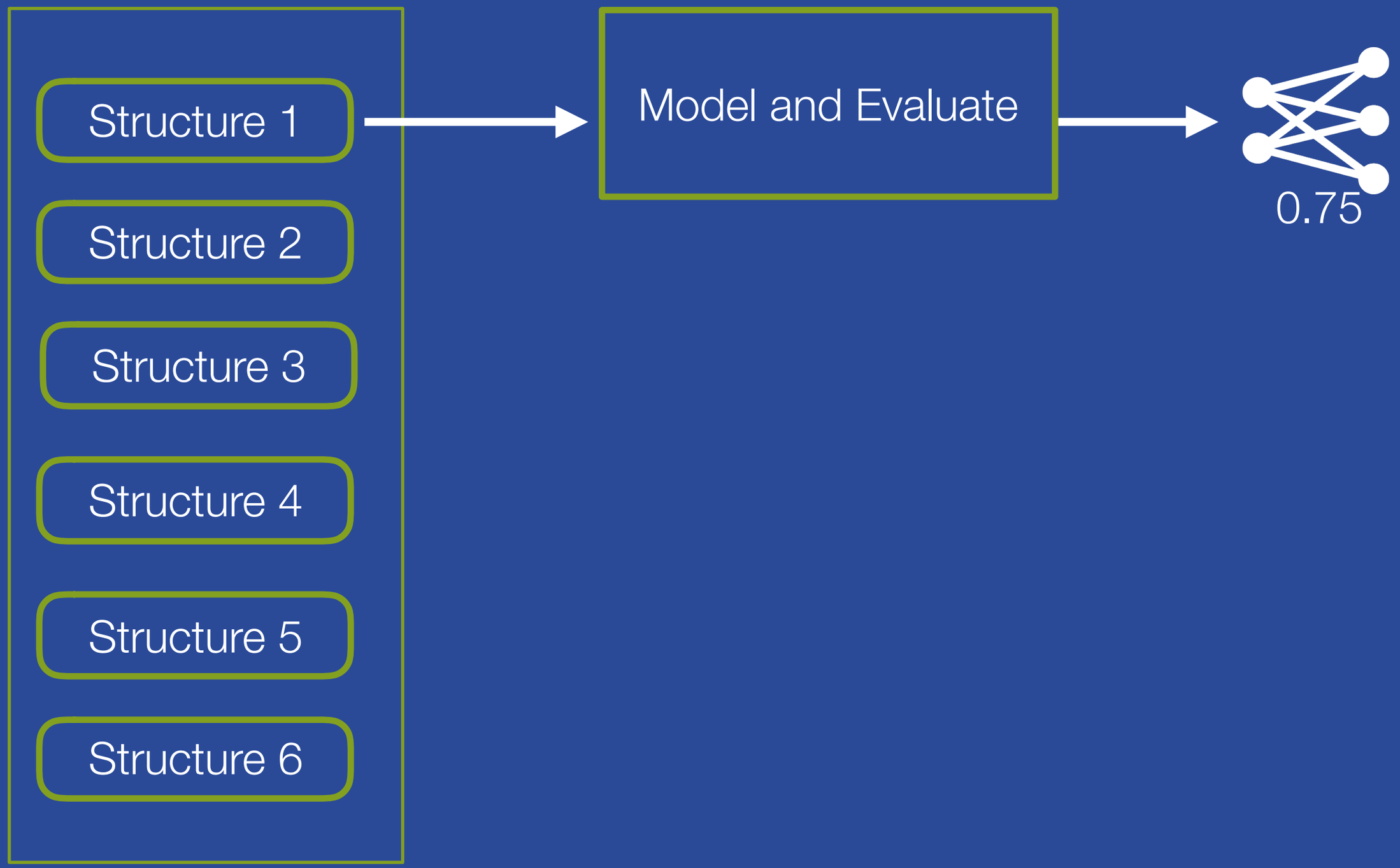
- We trained 296,748+ deep neural networks
- For each one, recorded the optimum network structure for the given dataset structure (number of fields, types of fields, etc)
- Trained a model which can predict the optimum network structure for any given dataset..
- This predicted structure can be used directly or as a seed for a more intensive network search

Network Search

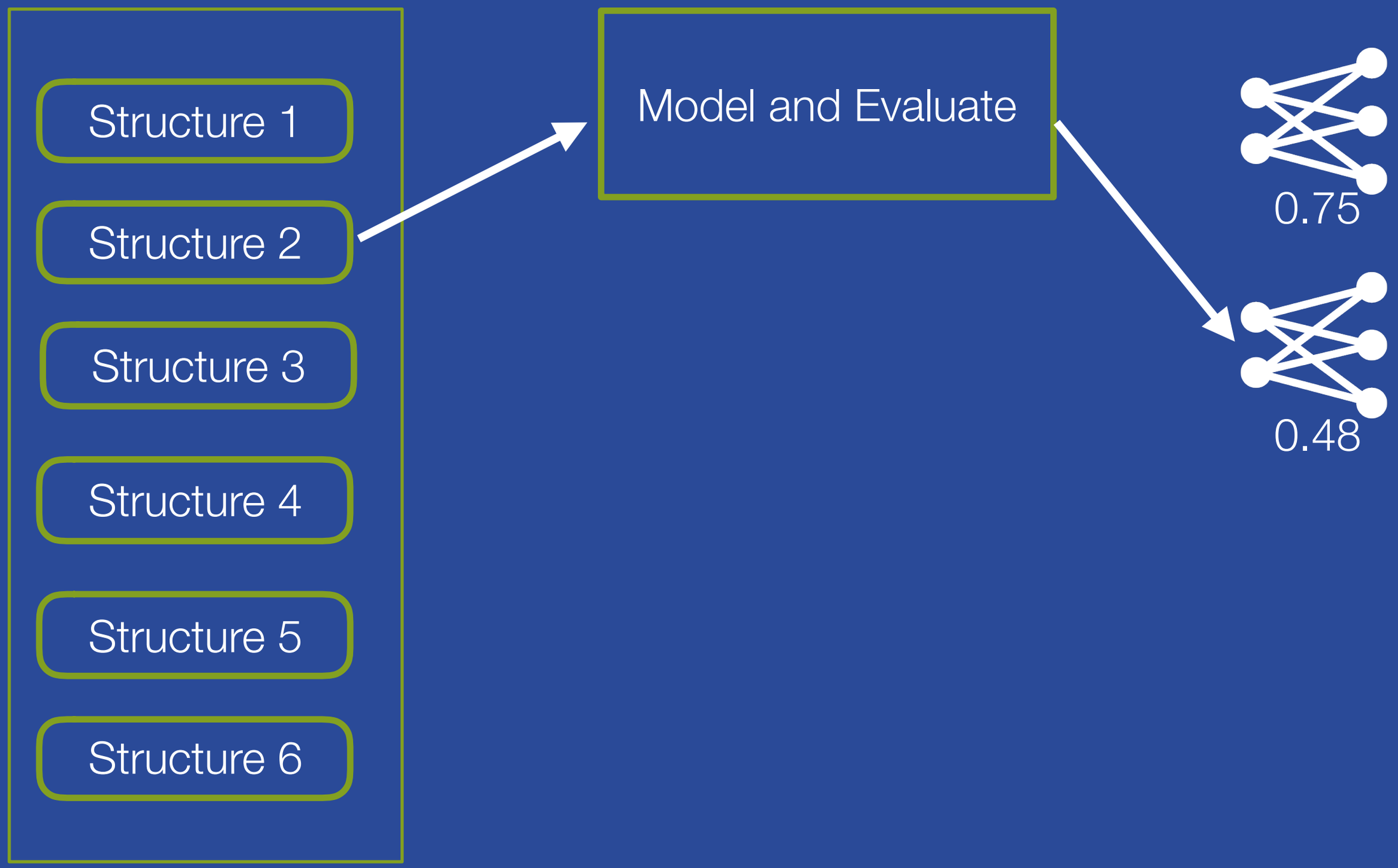


Model and Evaluate

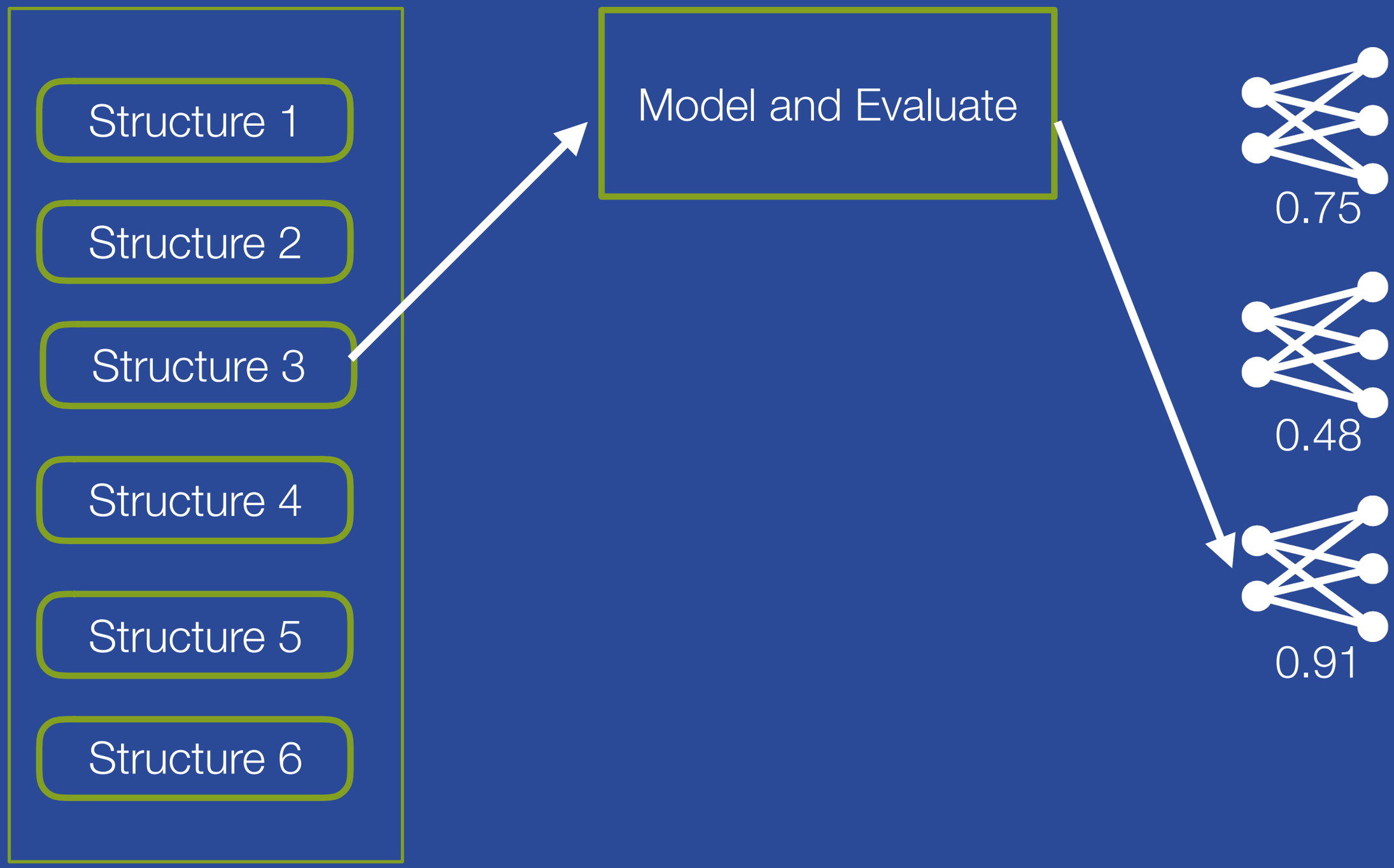
Network Search



Network Search



Network Search



Network Search

Structure 1

Structure 2

Structure 3

Structure 5

Structure 6

Structure 4

Model and Evaluate

0.75

0.48

0.91

Machine Learning!

Structure → performance

Deepnet Demo #2

Prediction Explanation

Can we gain back some interpretability?

- When making a prediction, vary the inputs slightly
- Generate, say, 1,000 more predictions “near” the original
- Measure how much the predicted output changes
- Compute how important each input is at changing the output

Deepnet Demo #3

When to use Deepnets

- When to use Deepnets
  - When the absolute best performance is worth the computational time
  - Ex: Predicting cancer versus predicting churn
- When not to use Deepnets
  - Small data (could still be thousands of instances)
  - As a first model - start with something simpler!
  - Feature engineering always beats model selection!
  - Problems that are easy or where retraining often
- Remember deep learning is just another sort of supervised learning algorithm

> "…deep learning has existed in the neural network community for over 20 years. Recent advances are driven by some relatively minor improvements in algorithms and models and by the availability of large data sets and much more powerful collections of computers." — Stuart Russell

Your Turn!

- Build a 1-click Deepnet with the Diabetes 80% Training set
- Evaluate the Deepnet with the Diabetes 20% Test set
- Compare the Evaluation with the rest that you have created
- Which performs best?

- Configure a second Deepnet:
  - Enable automatic network search
  - Limit the number of networks evaluated to 5
- Evaluate again and compare to the first Deepnet


bigml