



# Improving Business Decision Making with Bayesian Artificial Intelligence

Dr. Michael Green

2017-11-03

# Agenda

- Overview of AI and Machine learning
- Why do we need more?
- Our Bayesian Brains
- Probabilistic programming
- Tying it all together

# Overview of AI and Machine learning

“ *AI is the behaviour  
shown by an agent in  
an environment that  
seems to optimize the  
concept of future  
freedom*

# What is Artificial Intelligence?

## Artificial Narrow Intelligence

- Classifying disease
- Self driving cars
- Playing Go

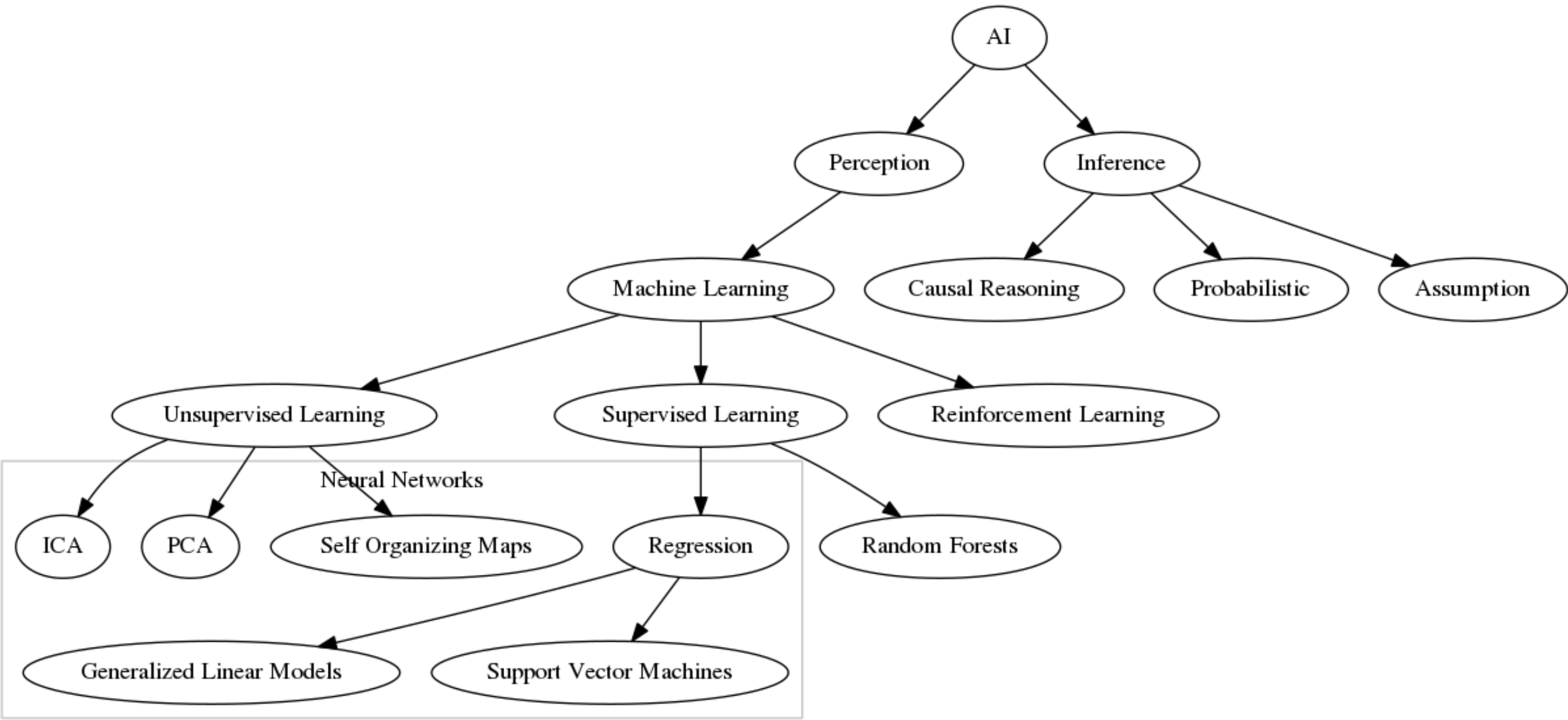
## Artificial General Intelligence

- Using the knowledge of driving a car and applying it to another domain specific task
- In general transcending domains

## Artificial Super Intelligence

- Scaling intelligence and moving beyond human capabilities in all fields
- Far away?

# The AI algorithmic landscape



Why do we need more?

# Machine learning can only take us so far

Why is that?

- **Data:** Data is not available in cardinality needed for many real world interesting applications
- **Structure:** Problem structure is hard to detect without domain knowledge
- **Identifiability:** For any given data set there are many possible models that fit really well to it with fundamentally different interpretations
- **Priors:** The ability to add prior knowledge about a problem is crucial as it is the only way to do science
- **Uncertainty:** Machine learning application based on maximum likelihood cannot express uncertainty about it's model



# The Bayesian brain

Domain space

$$p(x, y, \theta)$$

Machine learning

$$p(y|\theta, x)$$

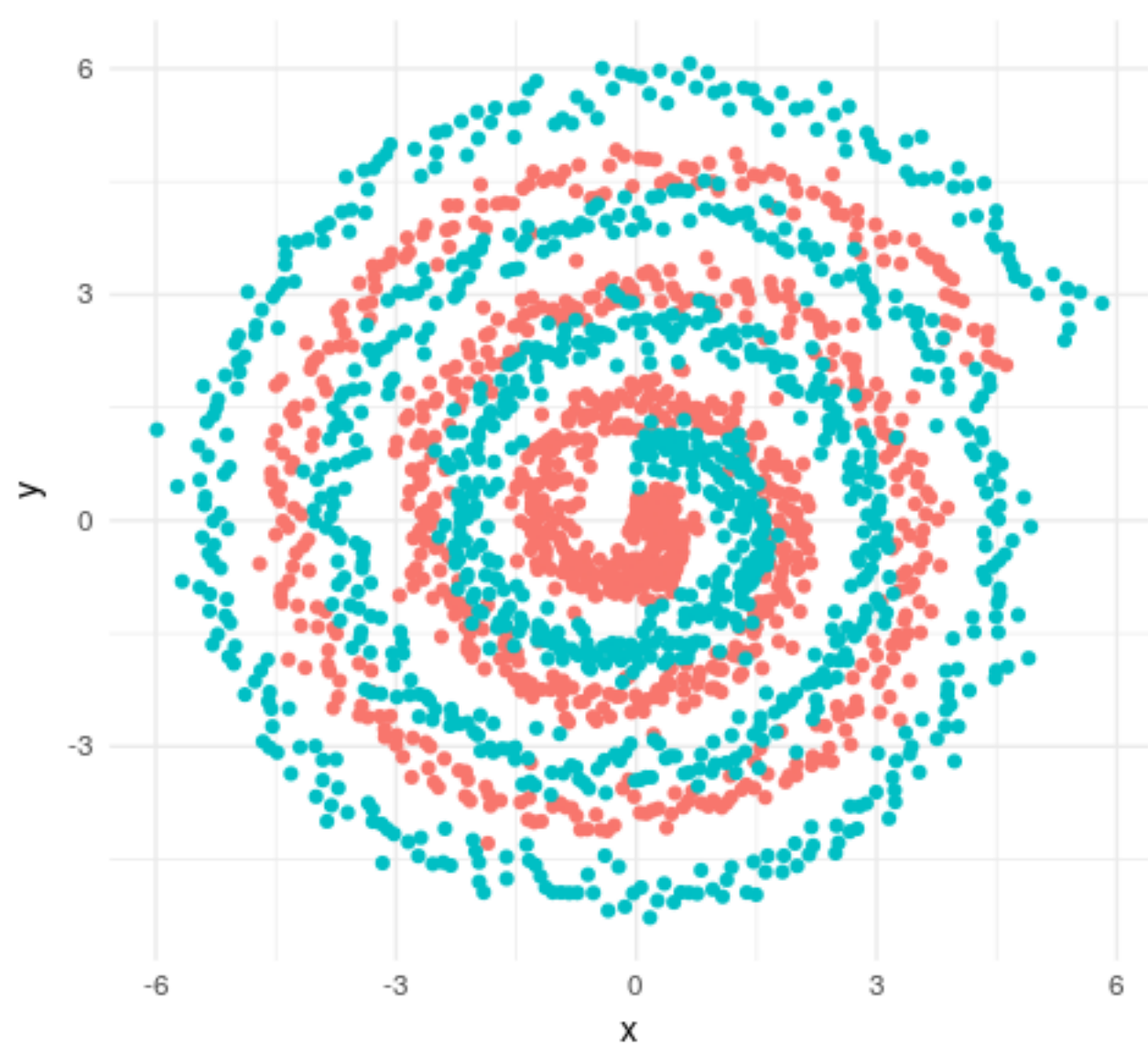
Inference

$$p(\theta|y, x) = \frac{p(y|\theta, x) p(\theta|x)}{\int p(y, \theta|x) d\theta}$$

“ *You cannot do science  
without assumption!*

A Neural Networks example

# Spiral data



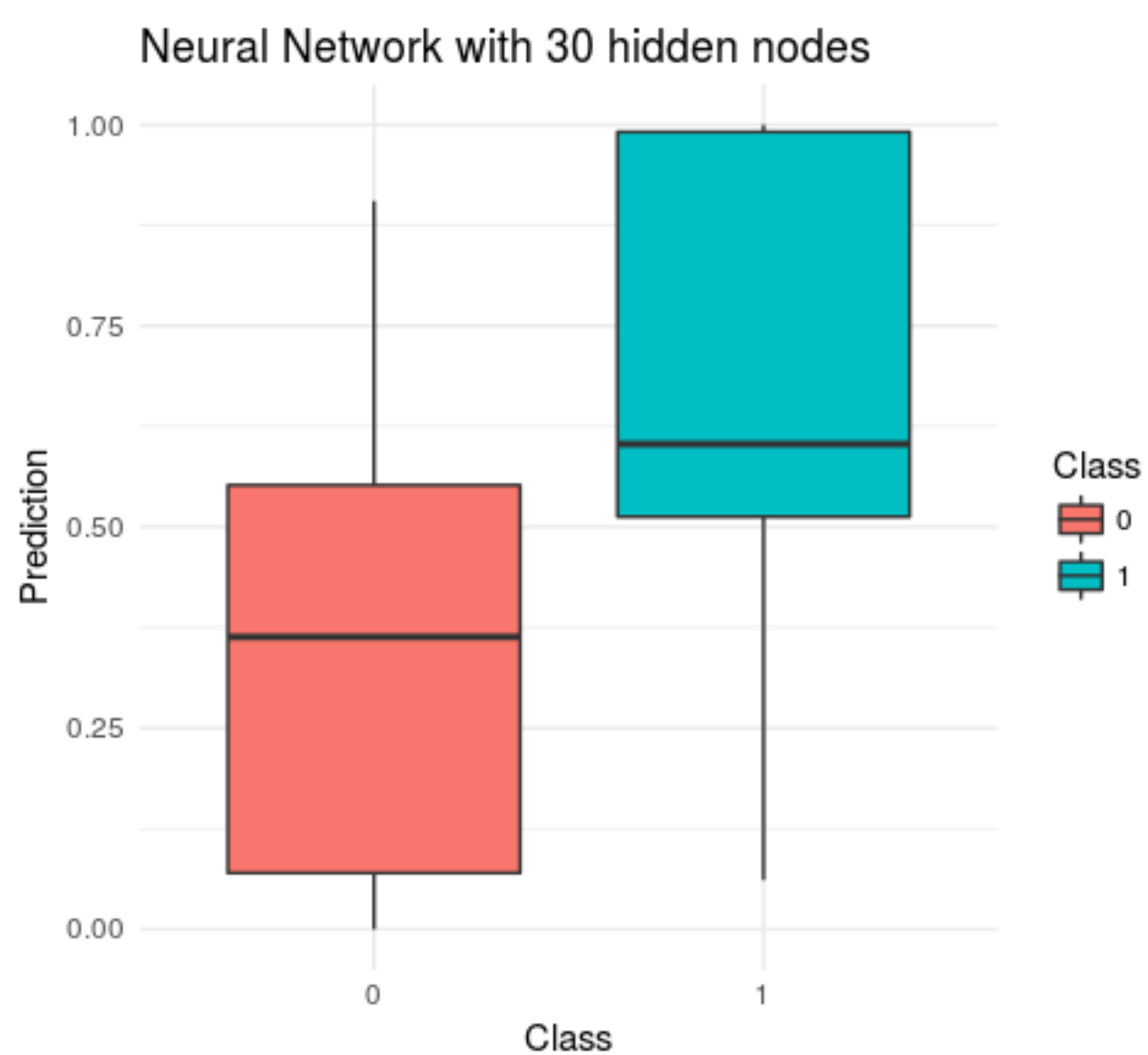
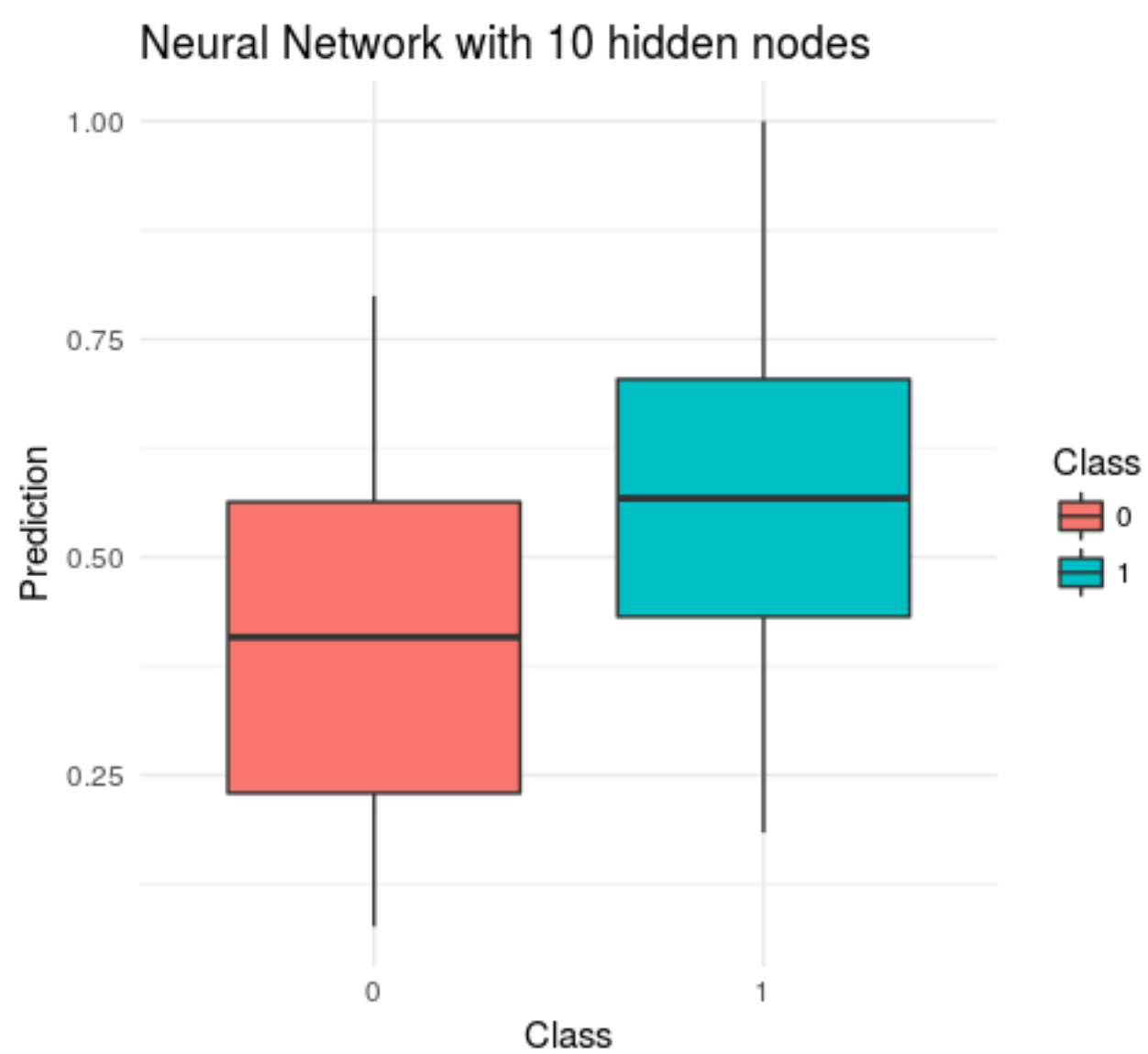
## Overview

This spiral data feature two classes and the task is to correctly classify future data points

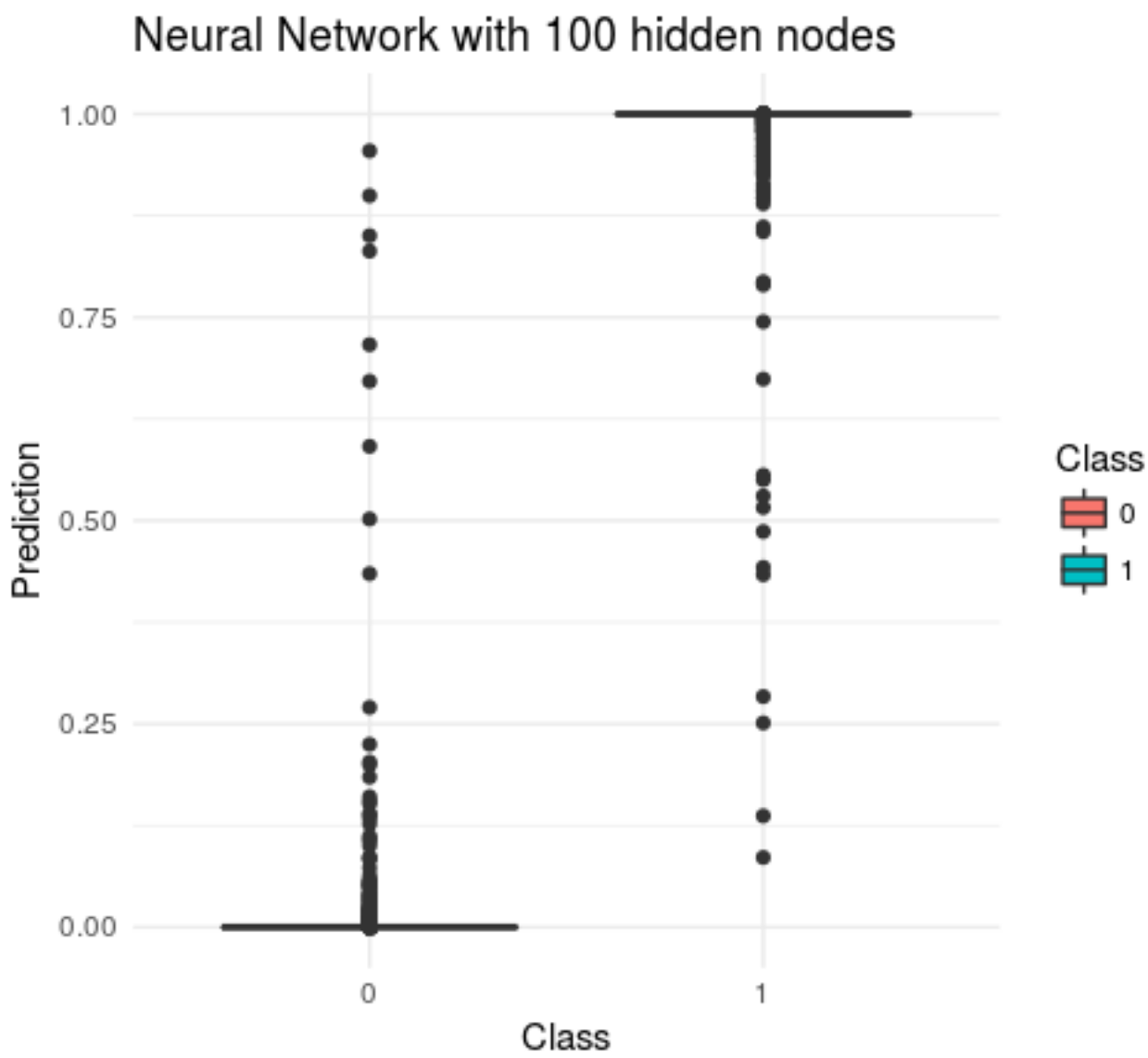
## Features of this data

- Highly nonlinear
- Noisy
- Apparent structure

# Running a Neural Network



# Running a Neural Network



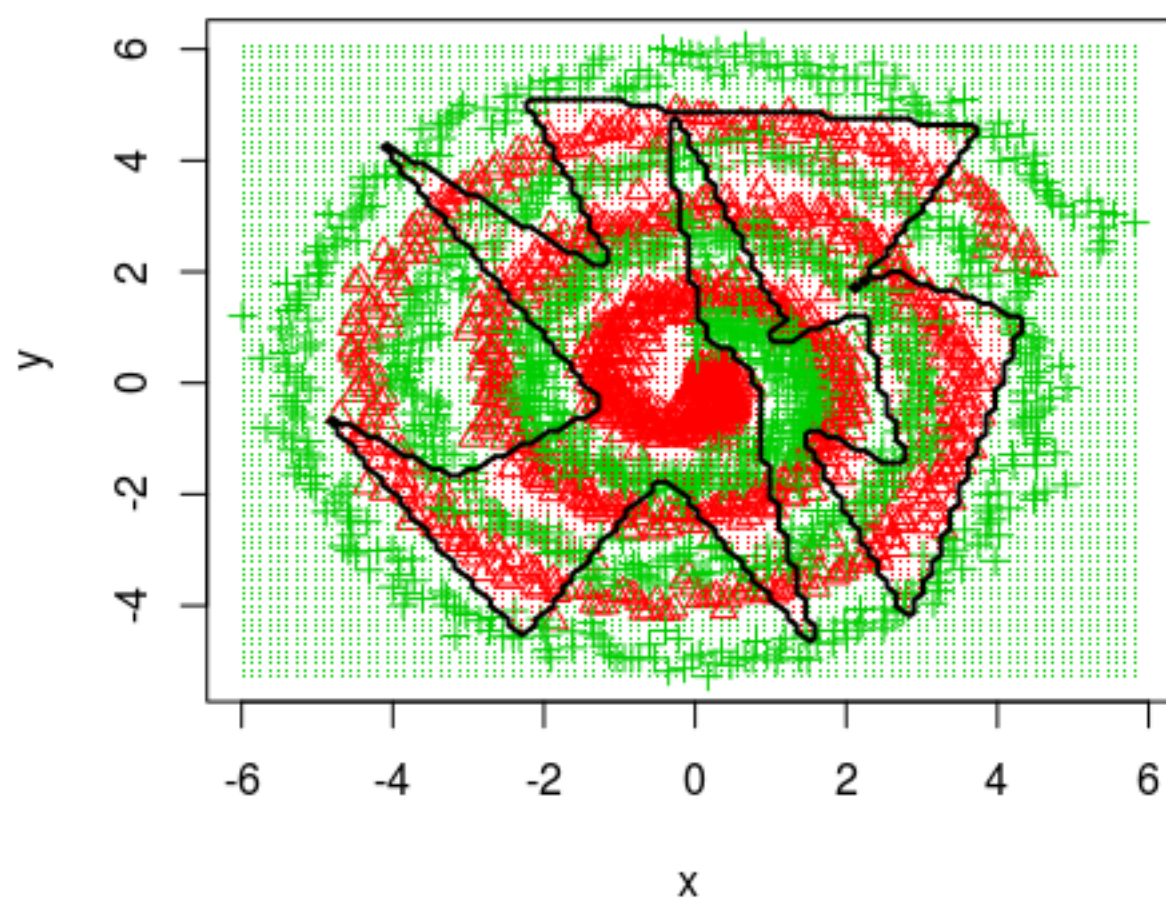
## Accuracy

Hidden nodes	Accuracy	AUC
10	65%	74%
30	71%	82%
100	99%	100%

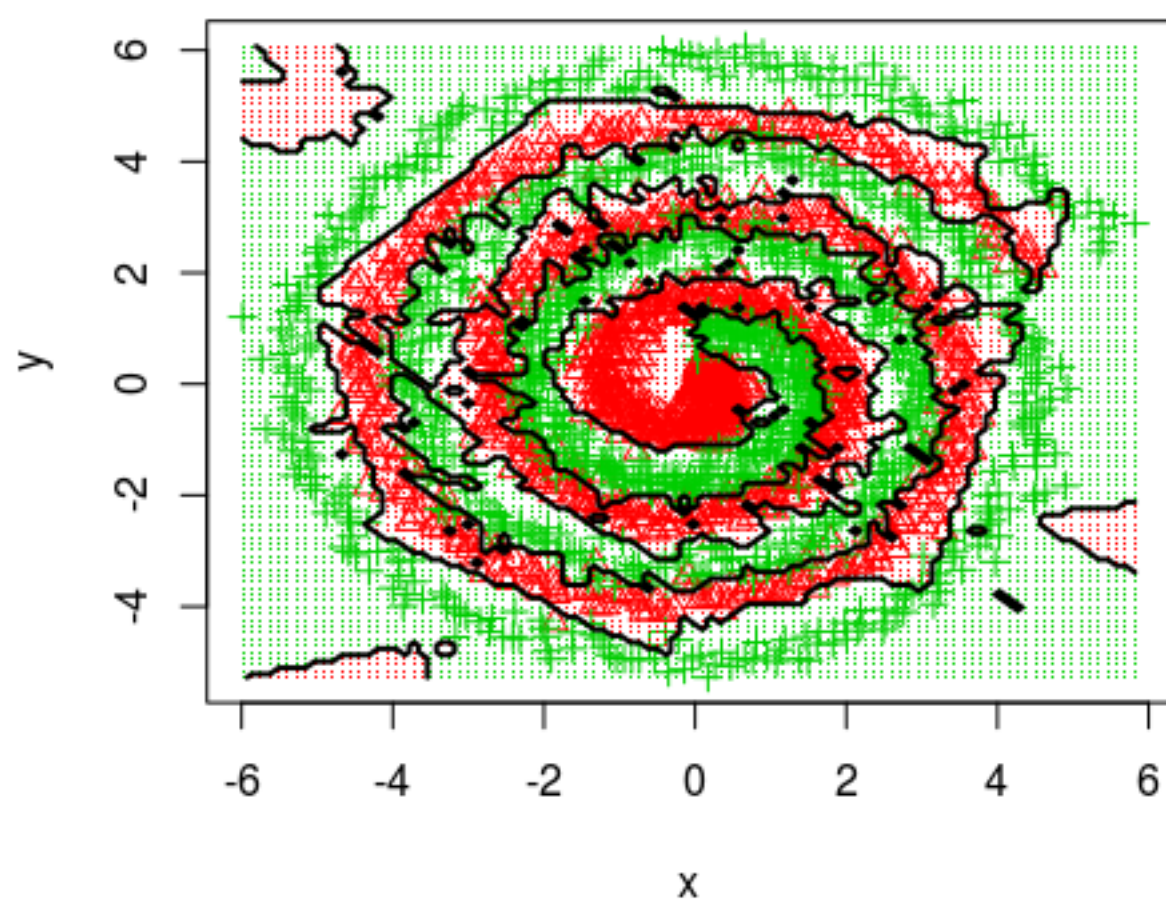
Only at 100 latent variables in the hidden layer do we reach the accuracy we want

# Decision boundaries

10 hidden nodes



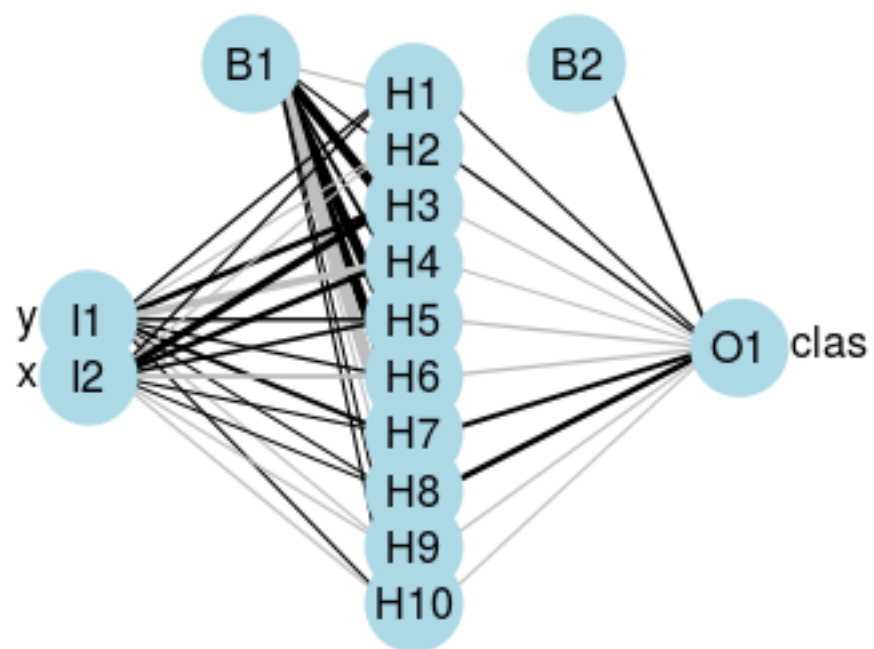
100 hidden nodes



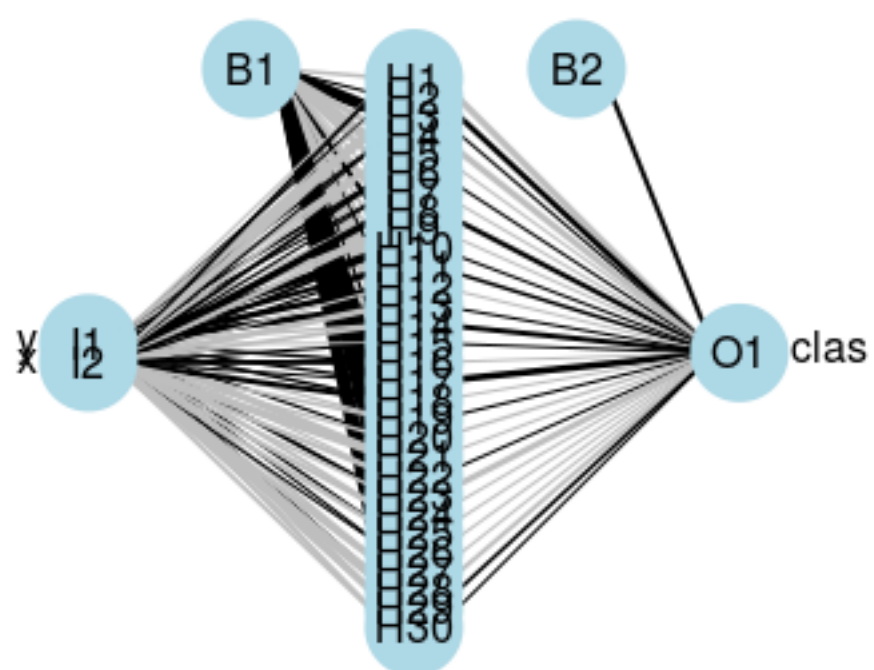


# Network architectures

10 Hidden nodes

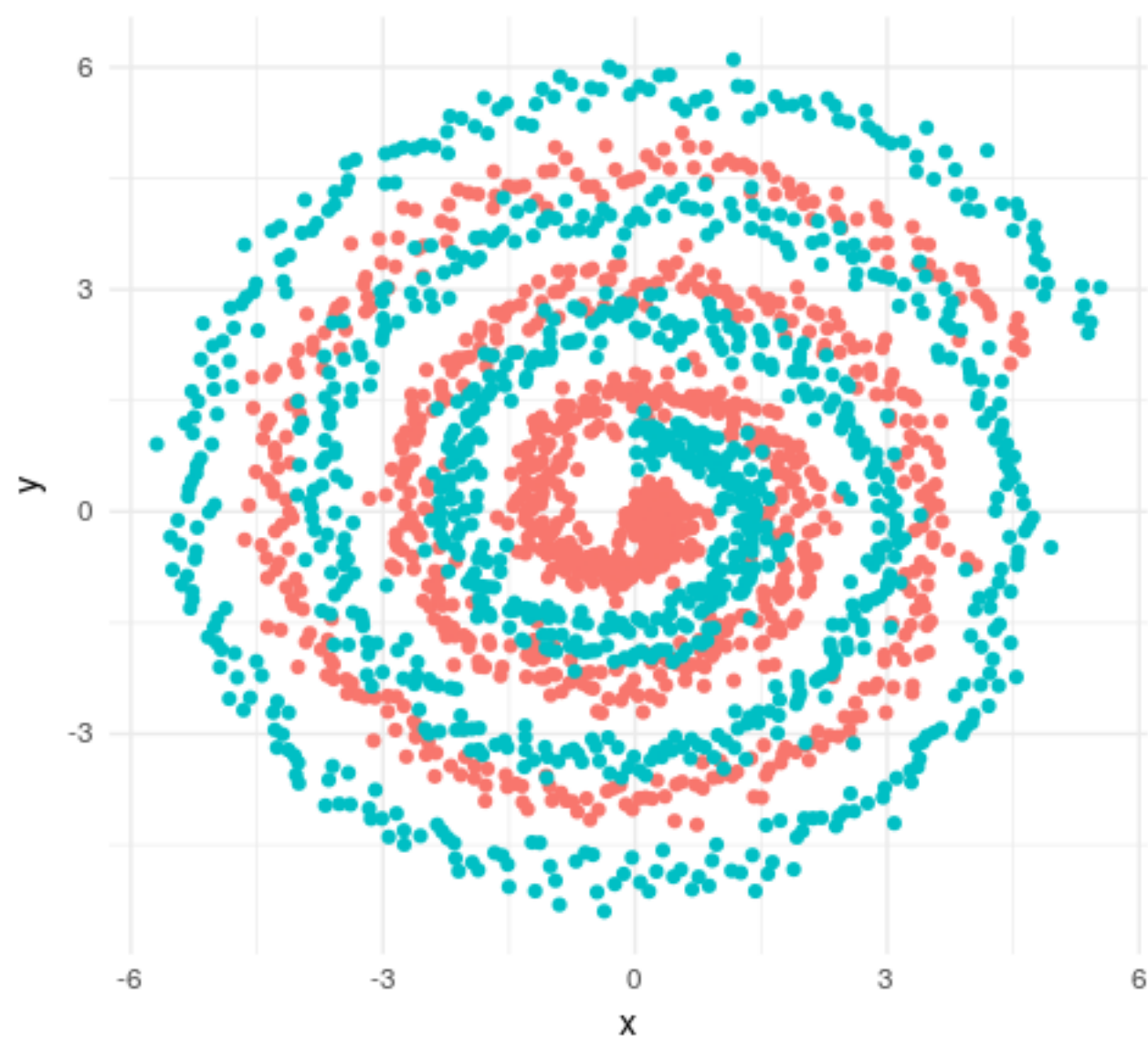


30 Hidden nodes

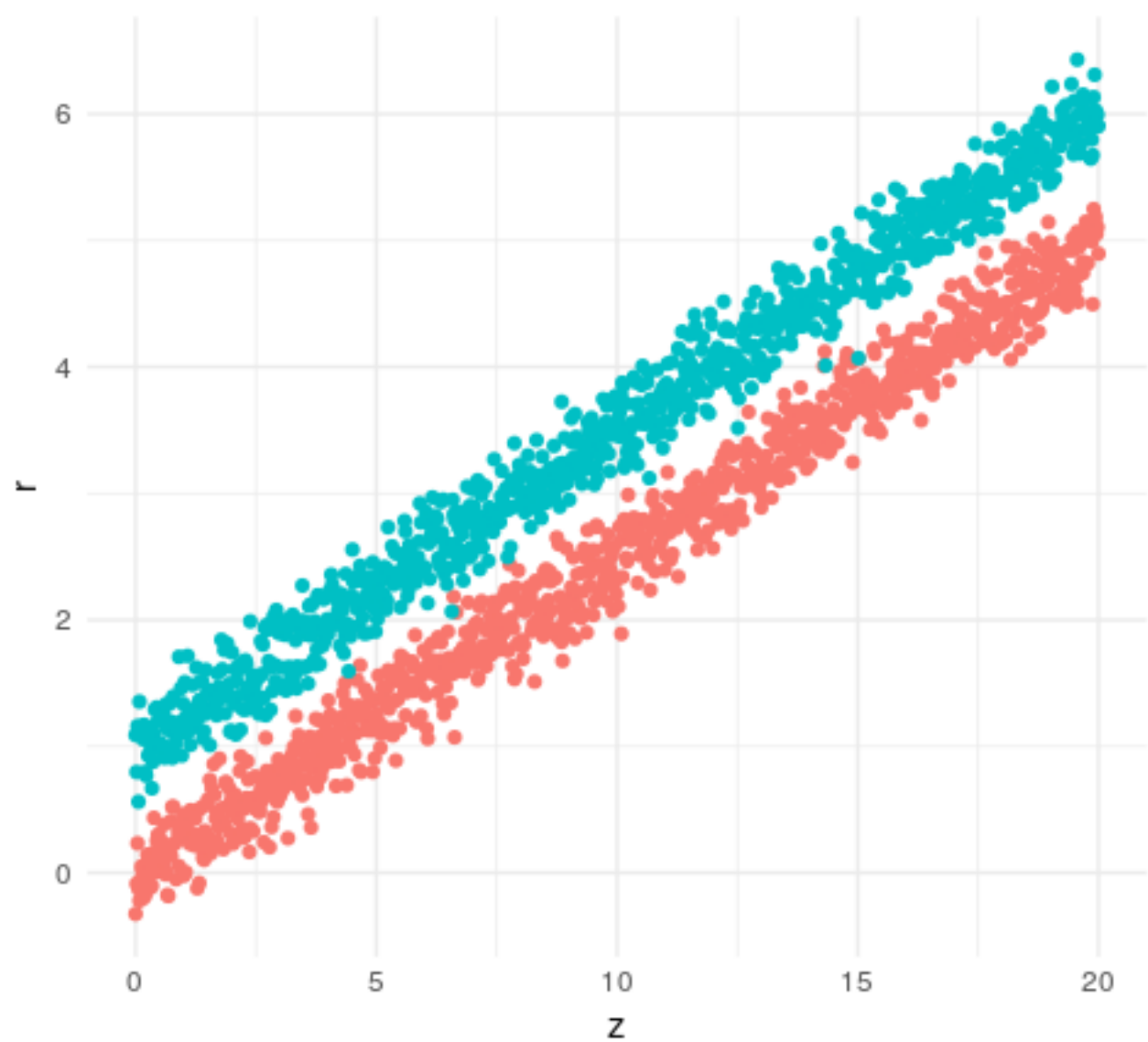




# Proper modeling of the problem



Cartesian coordinates



Polar coordinates

A probabilistic programming take

“ *Probabilistic programming is an attempt to unify general purpose programming with probabilistic modeling*

# Learning the data

$$x \sim \mathcal{N}(\mu_x, \sigma_x)$$

$$y \sim \mathcal{N}(\mu_y, \sigma_y)$$

$$\mu_x = (r + \delta) \cos\left(\frac{t}{2\pi}\right)$$

$$\mu_y = (r + \delta) \sin\left(\frac{t}{2\pi}\right)$$

$$\delta \sim \mathcal{N}(0.5, 0.1)$$

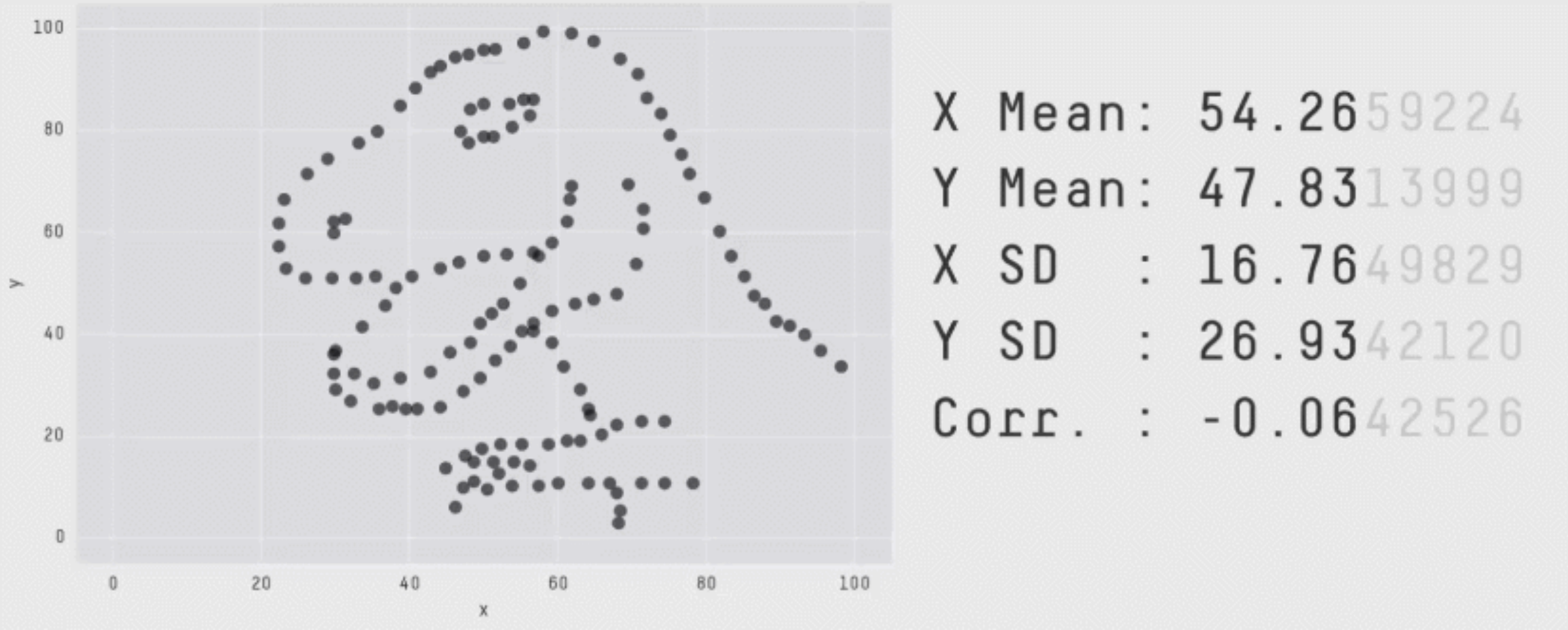
- Instead of throwing a lot of nonlinear generic functions at this beast we could do something different
- From just looking at the data we can see that the generating functions must look like
- Which fortunatly can be programmed using a probabilistic programming language

# What we gain from this

- We get to put our knowledge into the model solving for mathematical **structure**
- A generative model can be realized
- Direct measures of **uncertainty** comes out of the model
- No crazy statistical only results due to **identifiability** problems

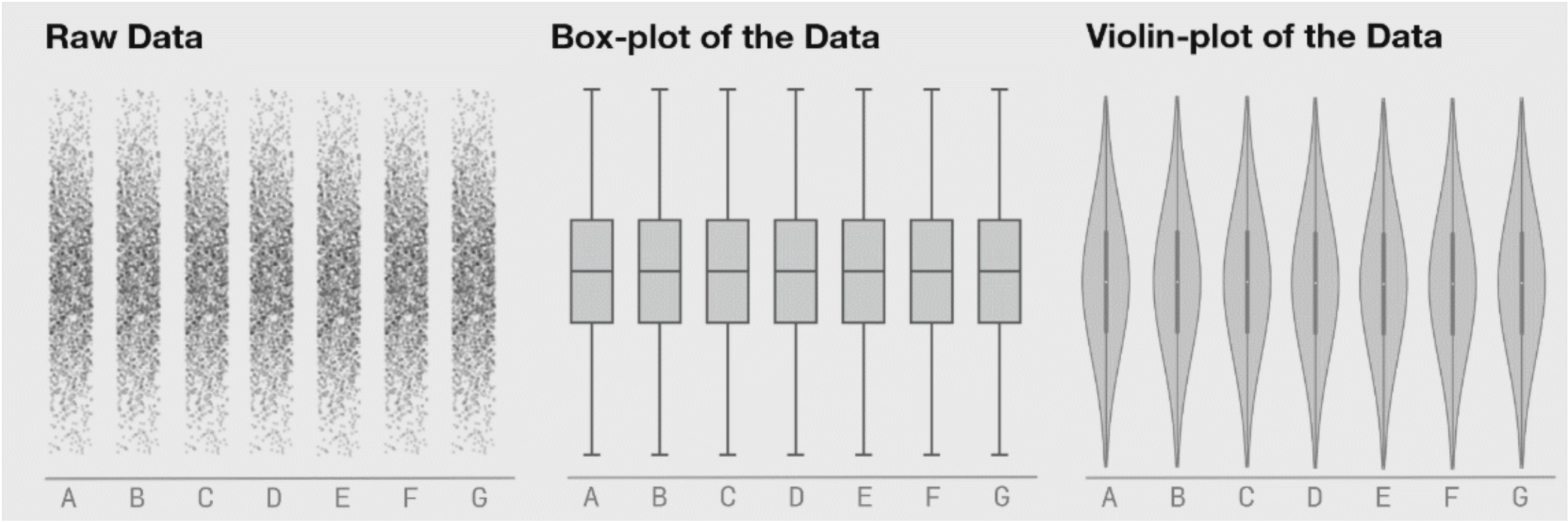
Summary Statistics is Dangerous

# Enter the Datasaurus



All datasets, and all frames of the animations, have the same summary statistics ( $\mu_x = 54.26$ ,  $\mu_y = 47.83$ ,  $\sigma_x = 16.76$ ,  $\sigma_y = 26.93$ ,  $\rho_{x,y} = -0.06$ ).

# Visualization matters!

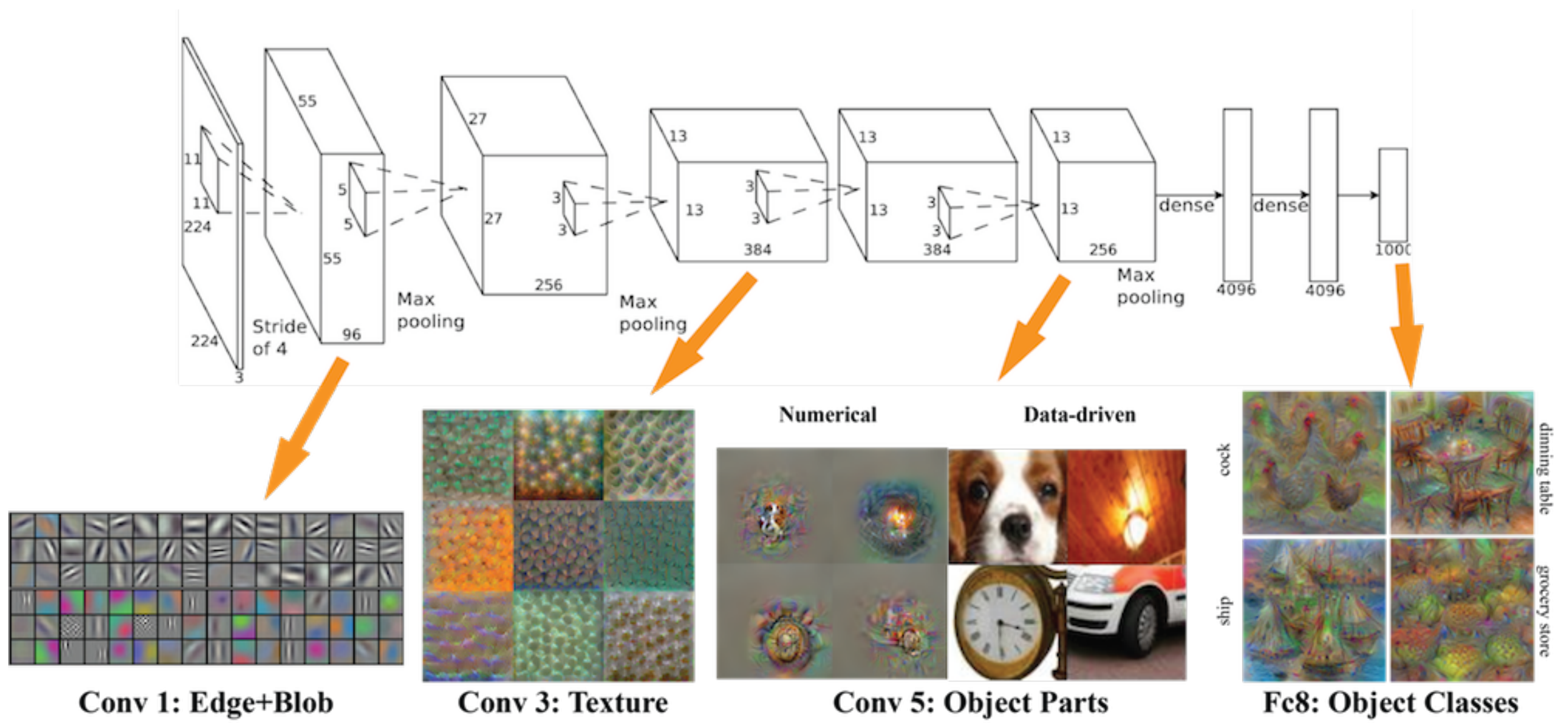


Seven distributions of data, shown as raw data points (or strip-plots), as box-plots, and as violin-plots.

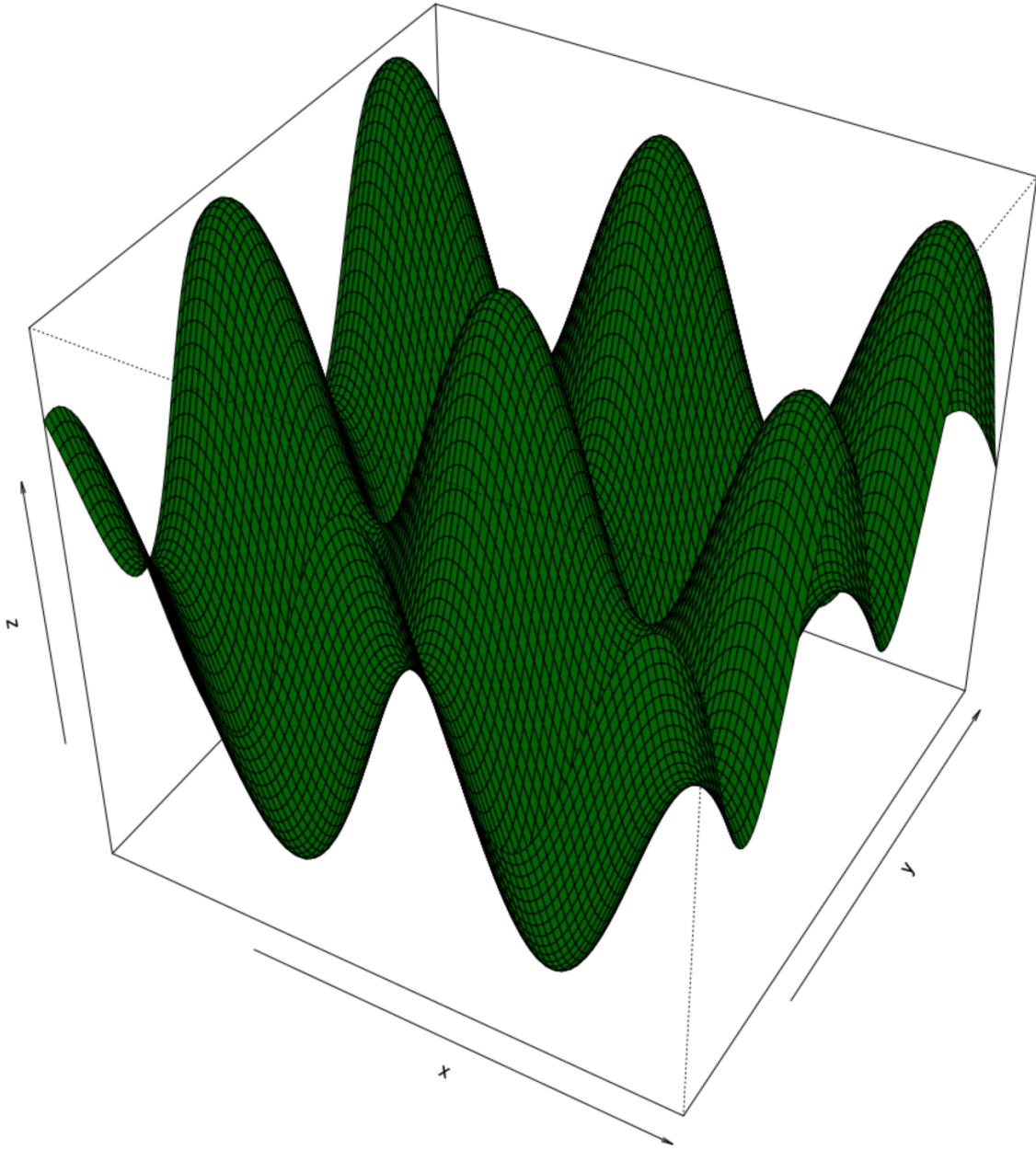


# Deep Learning

# Deep learning is just a stacked neural network

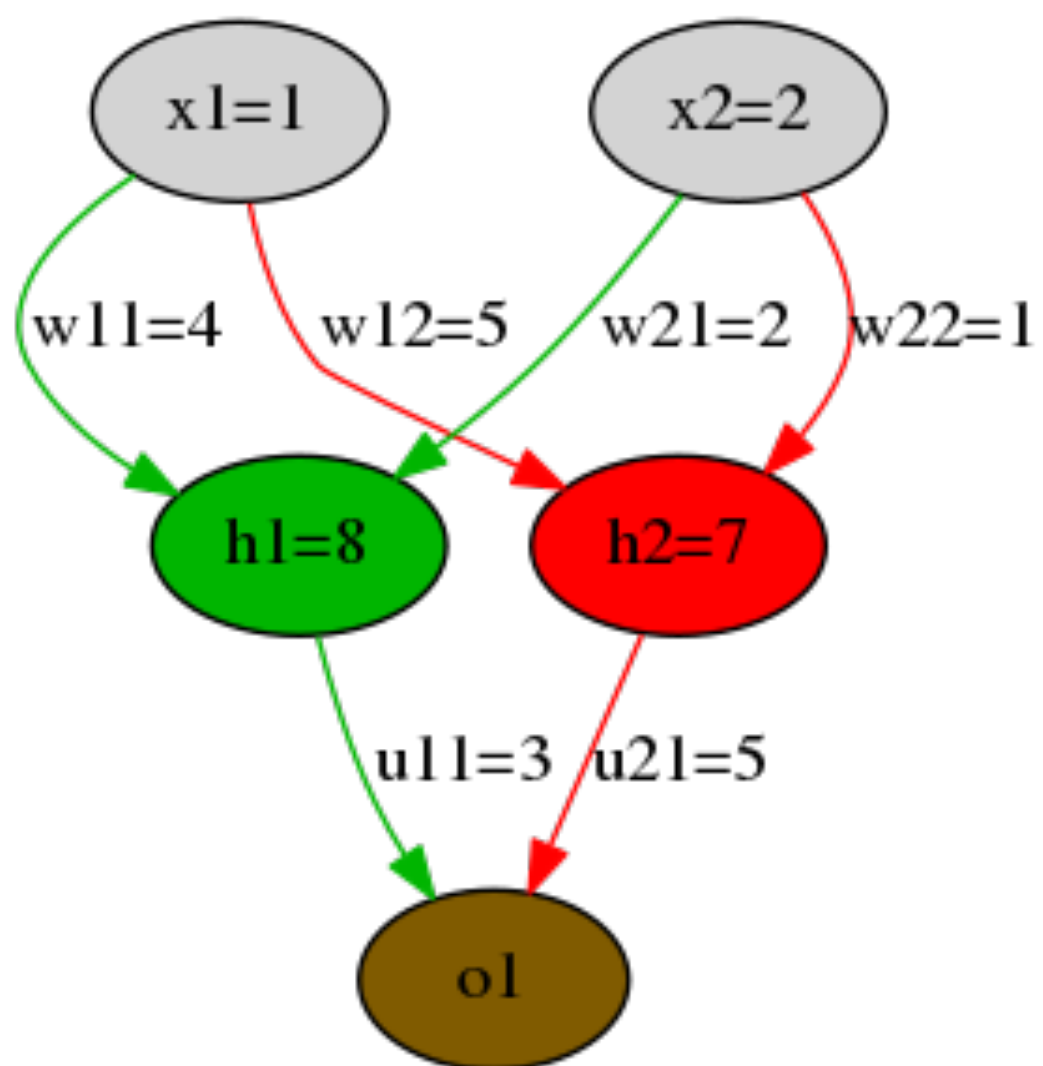
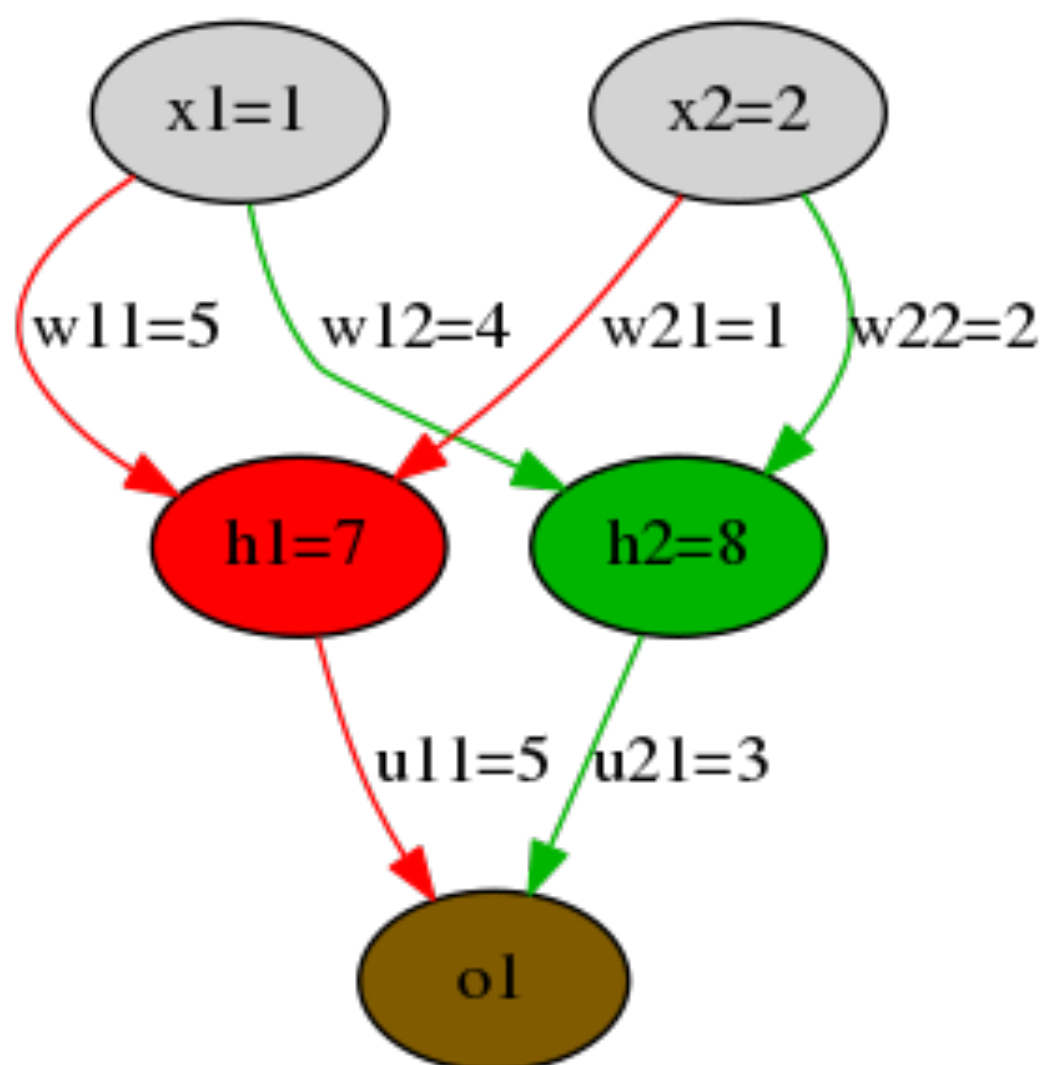


# Degeneracy in Neural Networks

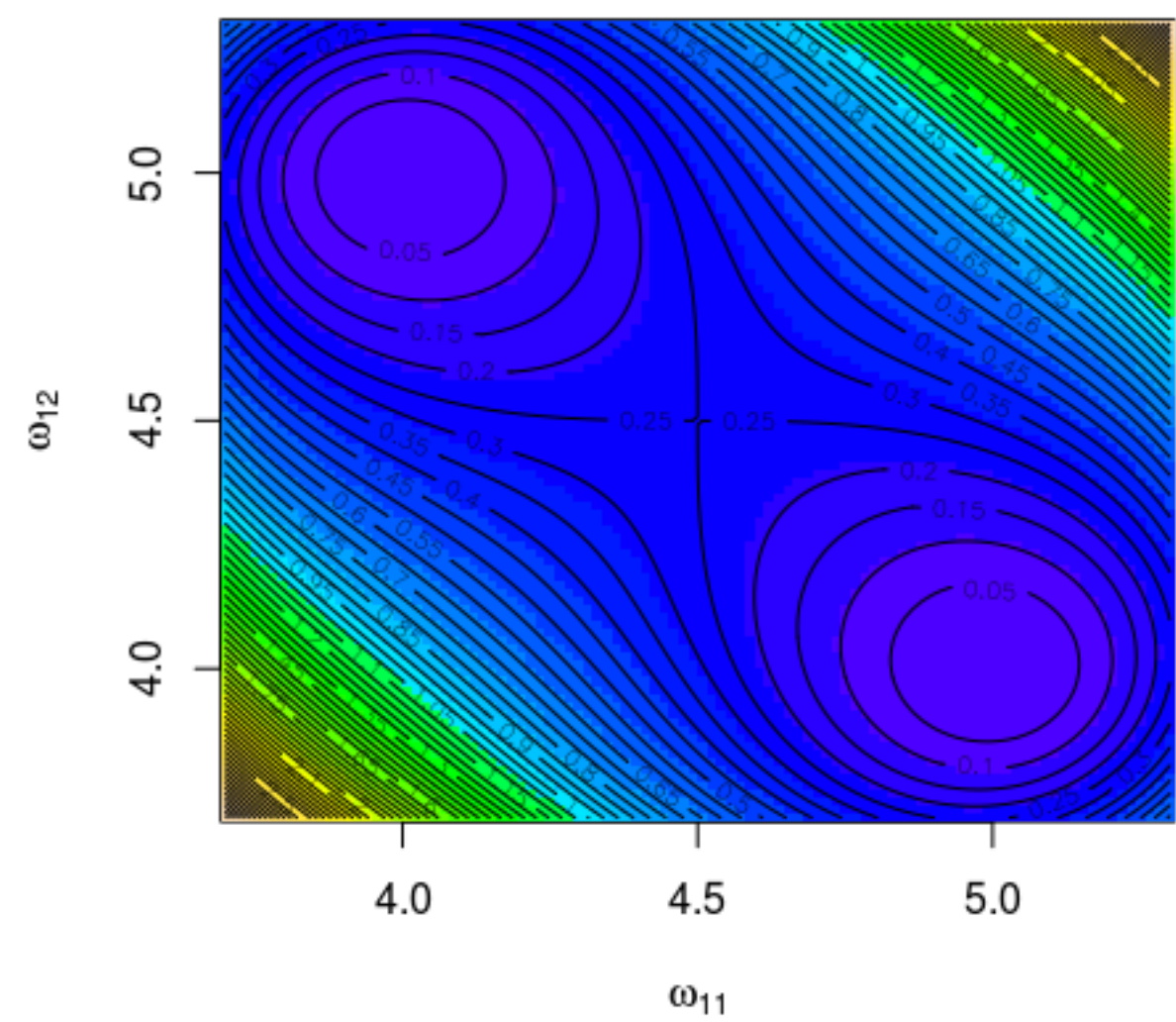
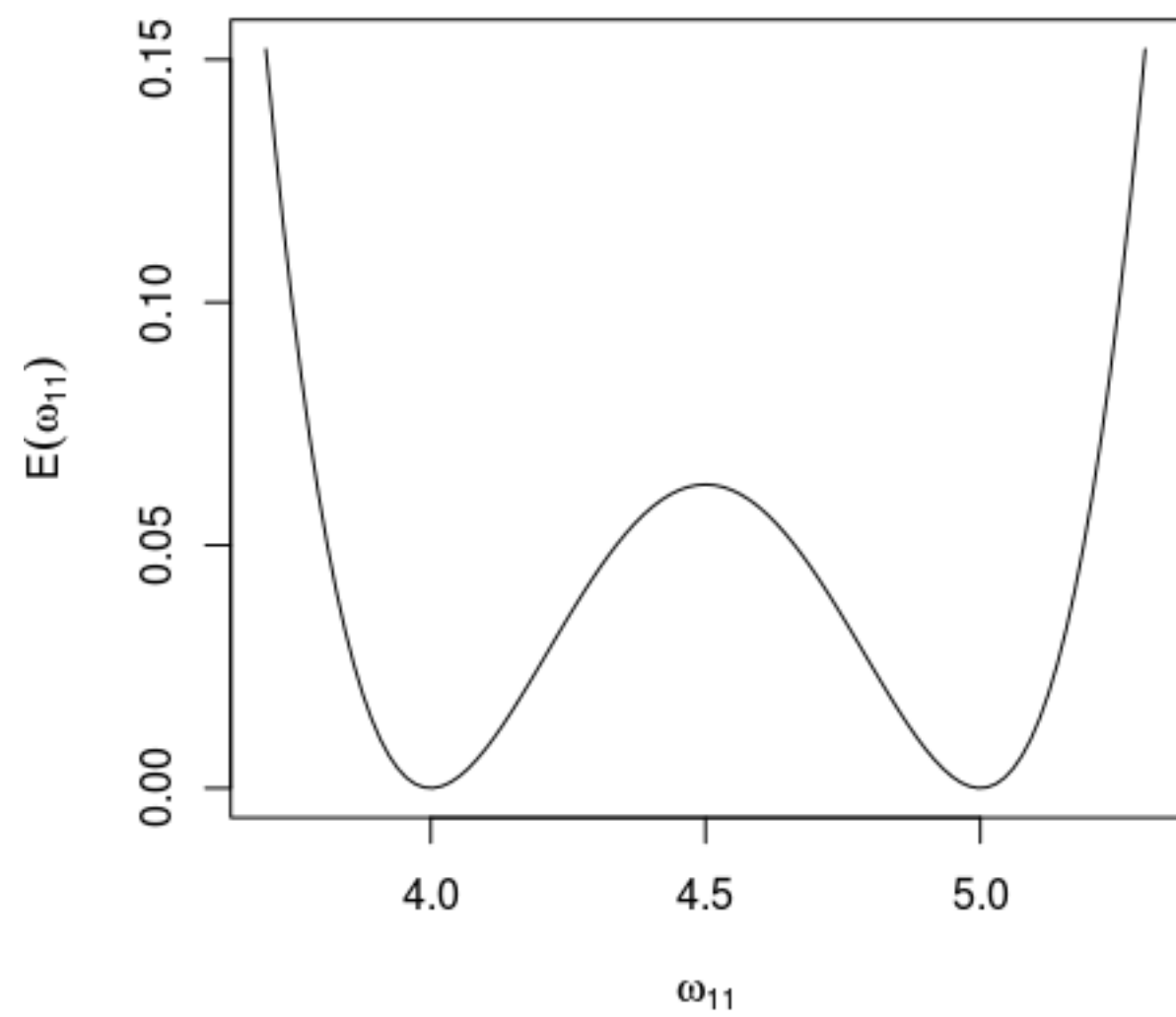


- A neural network is looking for the deepest valleys in this landscape
- As you can see there are many available
- Are they equivalent?
- Parameter space may have multiple "optimal" configurations not corresponding to physical reality
- As a consequence most Neural network models are overparameterized

# Degeneracy is in the structure

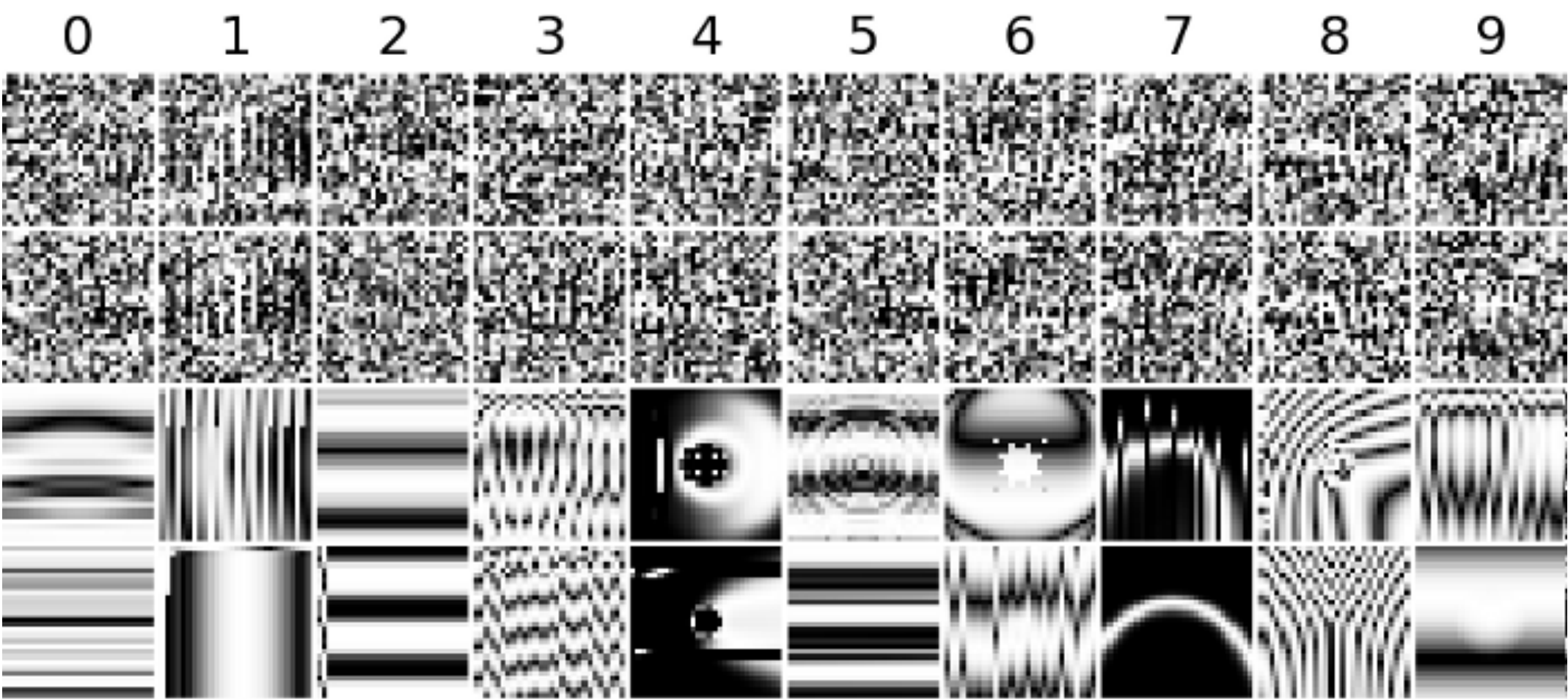


# Energy landscape in the $\omega_{11}$ , $\omega_{12}$ parameters





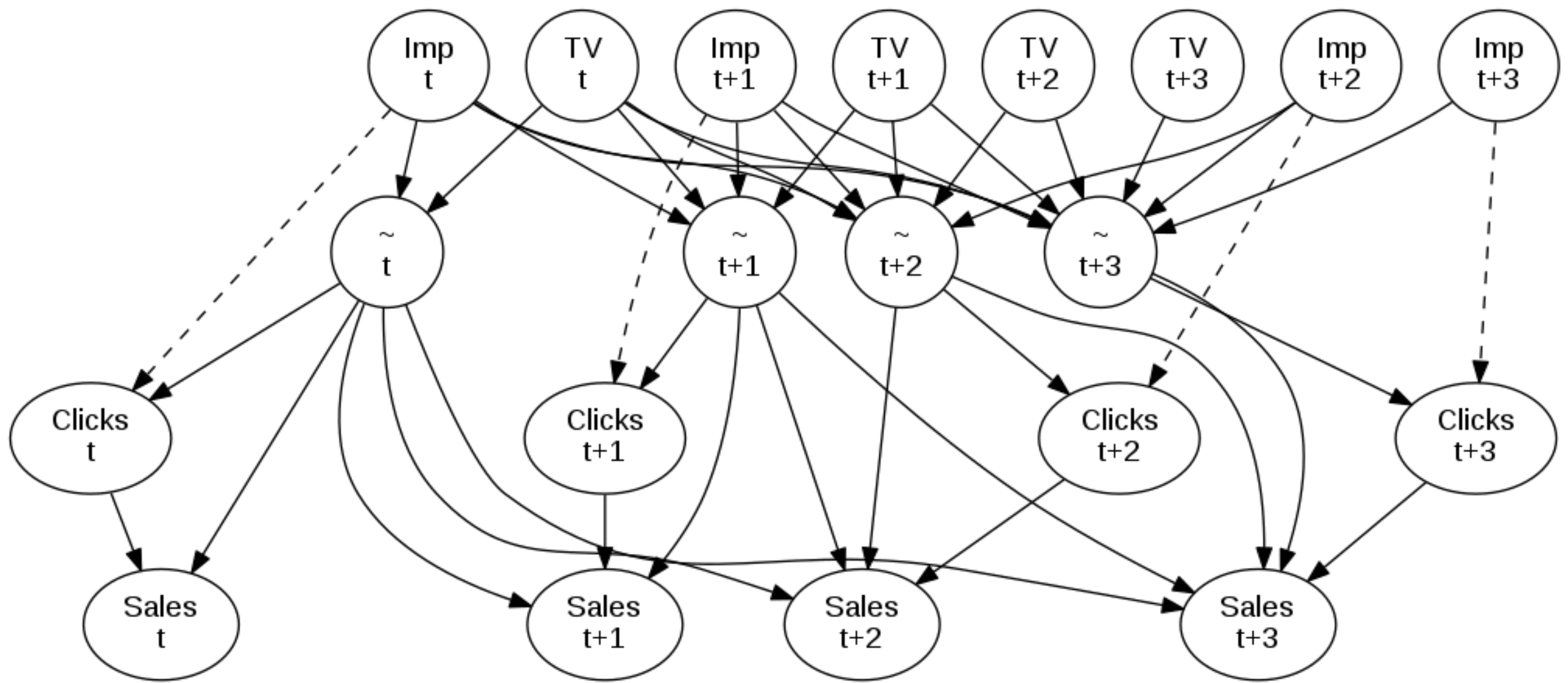
# So what's my point?



The point is that these spurious patterns will be realized in most if not all neural networks and their representation of the reality they're trying to predict will be inherently wrong. Read the paper by Nguyen A, Yosinski J, Clune J

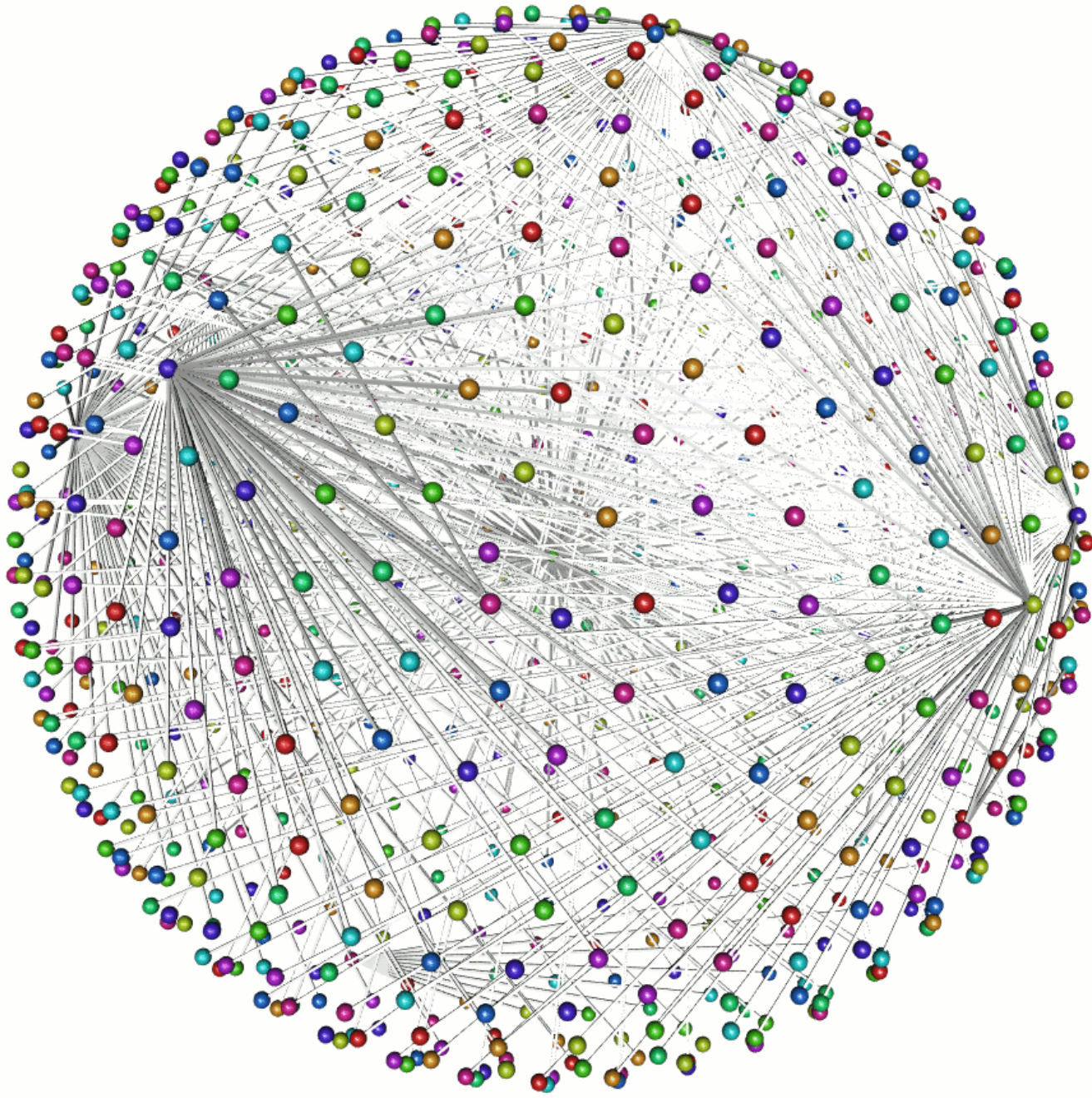
An example regarding time

# Events are not temporally independent





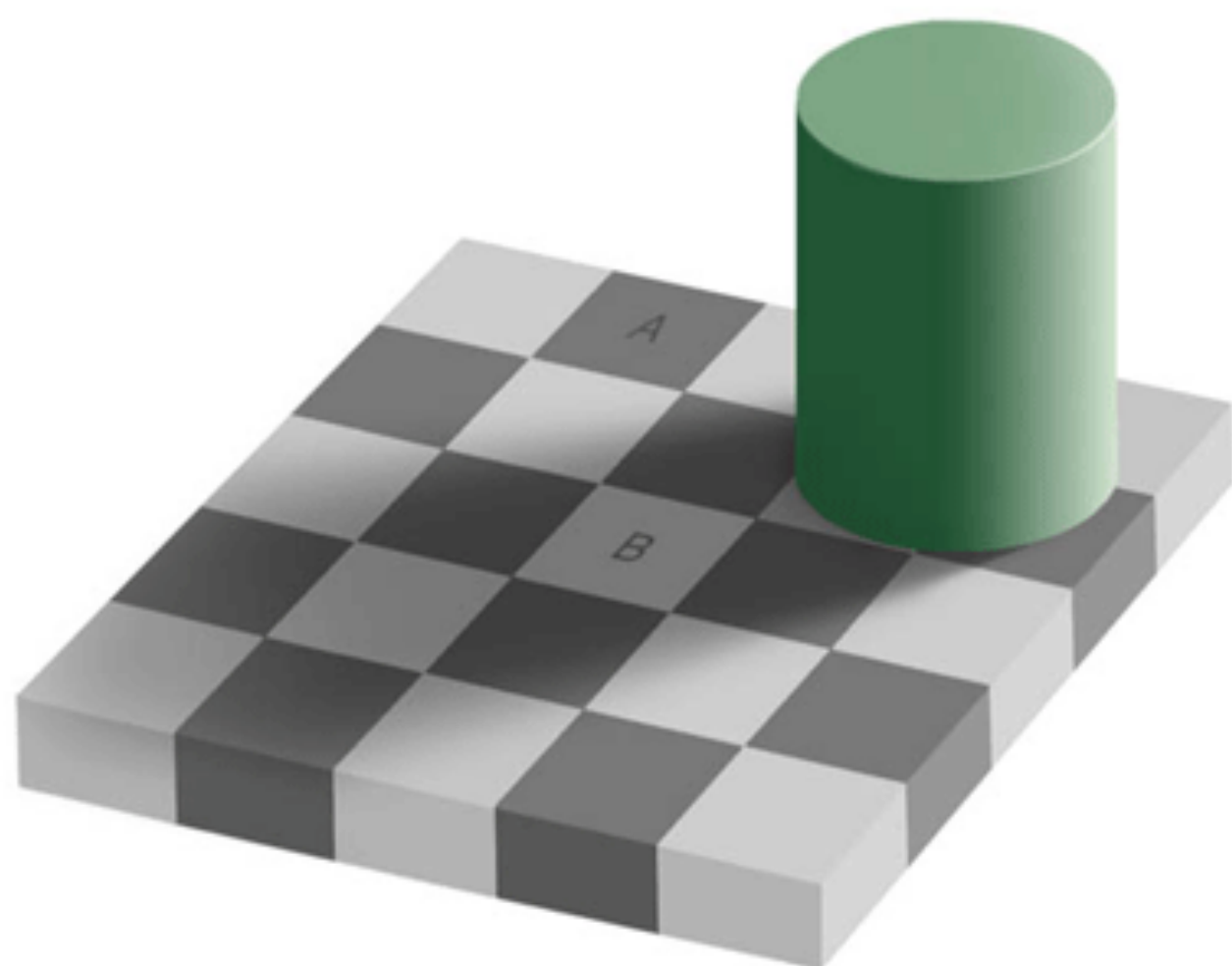
# A real world example from Blackwood



- Every node in the network represents a latent or observed variable and the edges between them represents interactions

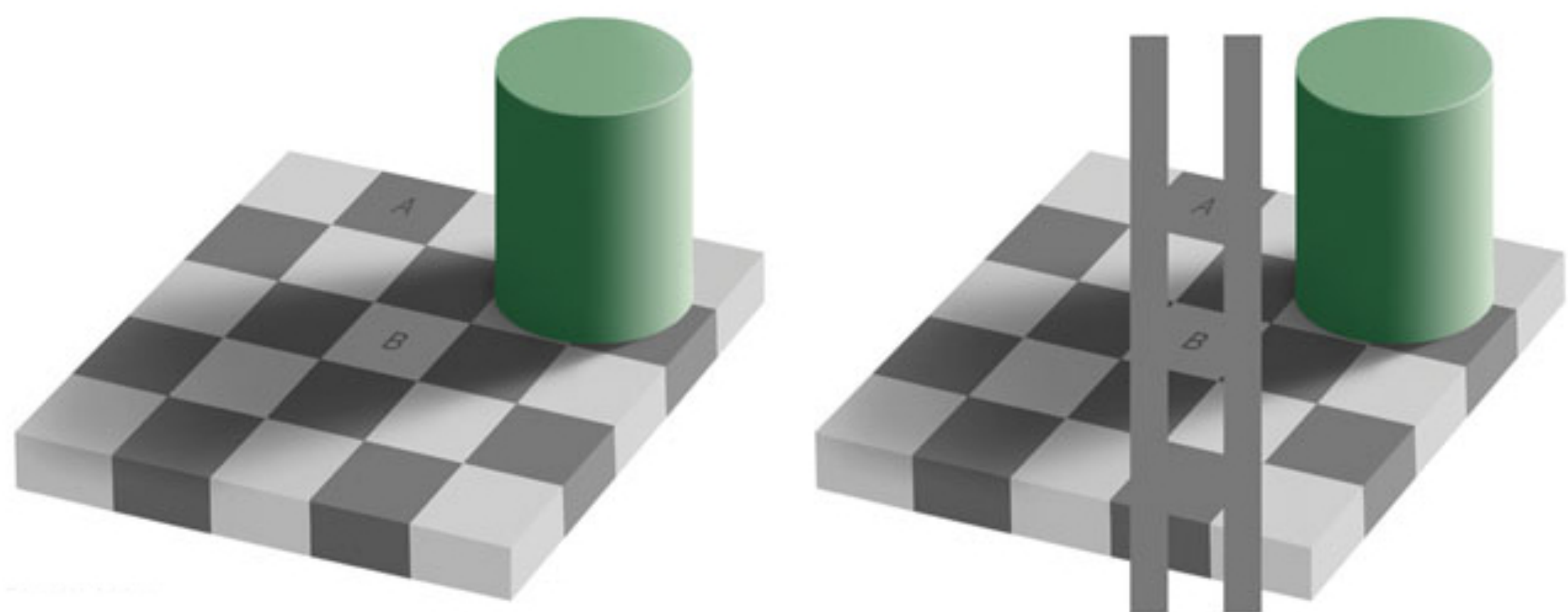
Our Bayesian brains

# About cognitive strength



- Our brain is so successful because it has a strong anticipation about what will come
- Look at the tiles to the left and judge the color of the A and B tile
- To a human this task is easy because we know what to expect and we quickly realize that A and B have different hues

The problem is only that you are wrong



Probabilistic programming

# What is it?

Probabilistic programming creates systems that help make decisions in the face of uncertainty. Probabilistic reasoning combines knowledge of a situation with the laws of probability. Until recently, probabilistic reasoning systems have been limited in scope, and have not successfully addressed real world situations.

- It allows us to specify the models as we see fit
- Curse of dimensionality is gone
- We get uncertainty measures for all parameters
- We can stay true to the scientific principle
- We do not need to be experts in MCMC to use it!

# Enter Stan a probabilistic programming language

Users specify log density functions in Stan's probabilistic programming language and get:

- full Bayesian statistical inference with MCMC sampling (NUTS, HMC)
- approximate Bayesian inference with variational inference (ADVI)
- penalized maximum likelihood estimation with optimization (L-BFGS)

Stan's math library provides differentiable probability functions & linear algebra (C++ autodiff). Additional R packages provide expression-based linear modeling, posterior visualization, and leave-one-out cross-validation.

# A note about uncertainty

## Task

- Suppose I gave you a task of investing 1 million USD in either Radio or TV advertising
- The average ROI for Radio and TV is 0.5
- How would you invest?

## Further information

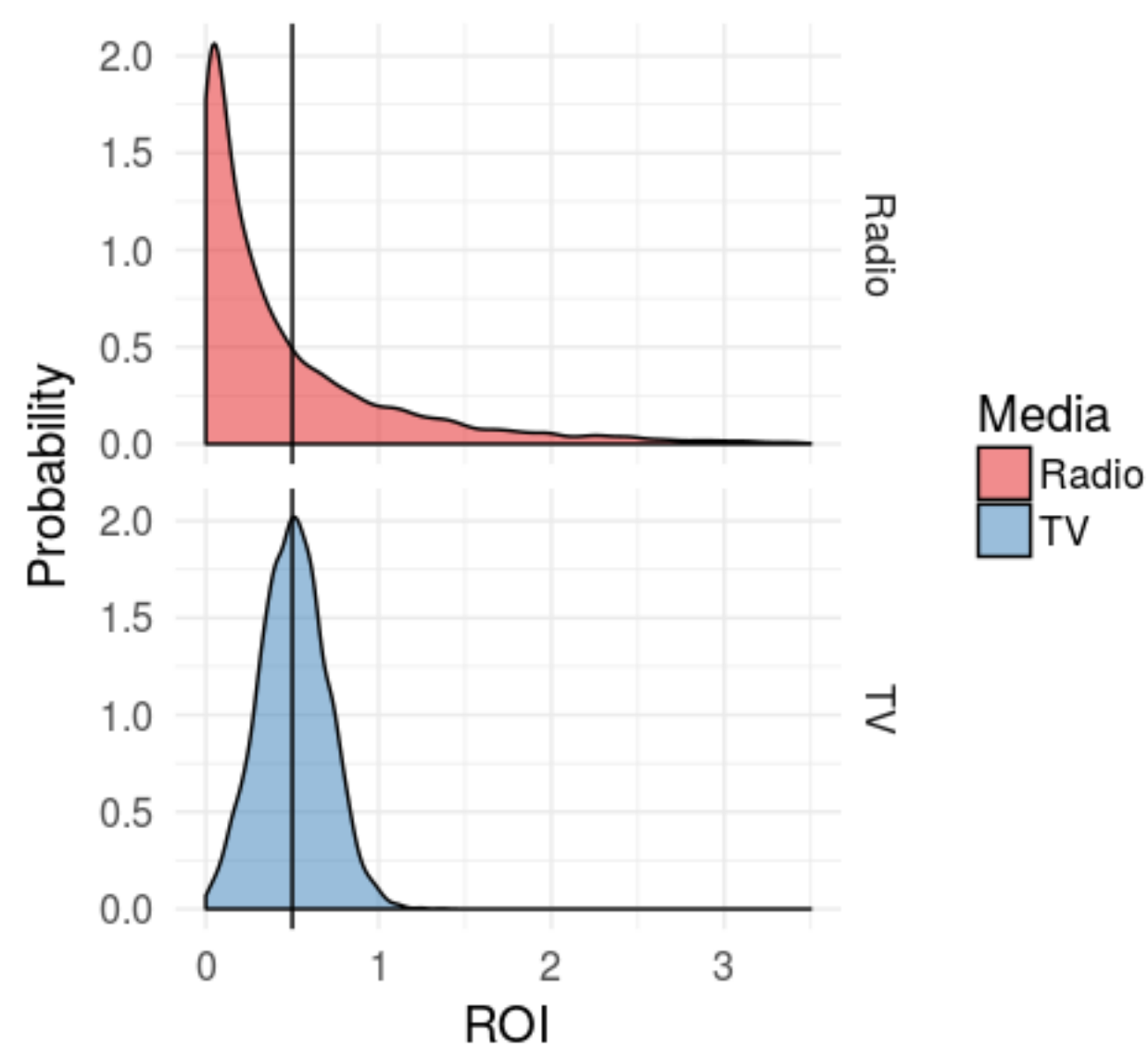
- Now I will tell you that the ROI's are actually distributions
- Radio and TV both have a minimum value of 0
- Radio and TV have a maximum of 9.3 and 1.4 respectively
- Where do you invest?

## Solution

- How to think about this?
- You need to ask the following question
- What is  $p(ROI > 0.3)$ ?
- The answer to that question is around 40 and 90 percent for Radio and TV respectively!



# A note about uncertainty - Continued



	Radio	TV
Mean	0.5	0.5
Min	0.0	-0.3
Max	9.3	1.4
Median	0.2	0.5
Mass	0.4	0.9
Sharpe	0.7	2.5

“*You cannot make  
optimal decisions  
without quantifying  
what you don't know*”

Tying it all together

# Deploying a Bayesian model using R

- There's a Docker image freely available with an up to date R version installed and the most common packages
- <https://hub.docker.com/r/drmike/r-bayesian/>

## Features

- R: Well you know
- RStan: Run the Bayesian model
- OpenCPU: Immediately turn your R packages into REST API's

# How to use it

First you need to get it

- `sudo docker pull dr mike/r-bayesian`
- `sudo docker run -it dr mike/r-bayesian bash`

You can also test the imbedded stupid application

- `docker run -d -p 80:80 -p 443:443 -p 8004:8004 dr mike/r-bayesian`
- `curl http://localhost:8004/ocpu/library/stupidweather/R/predictweather/json -H "Content-Type: application/json" -d '{"n":6}'`

# Conclusion

# Take home messages

- The time is ripe for marrying machine learning and inference machines
- Don't get stuck in patterns using existing model structures
- Stay true to the scientific principle
- Always state your mind!
- Be free, be creative and most of all have fun!

# Session Information

For those who care

```
## setting value
## version R version 3.4.2 (2017-09-28)
## system x86_64, linux-gnu
## ui X11
## language en_US:en
## collate en_US.UTF-8
## tz Europe/Copenhagen
## date 2017-11-03
##
## package * version date source
## assertthat 0.2.0 2017-04-11 CRAN (R 3.3.3)
## backports 1.1.1 2017-09-25 CRAN (R 3.4.2)
## base * 3.4.2 2017-10-28 local
## bindr 0.1 2016-11-13 cran (@0.1)
## bindrcpp * 0.2 2017-06-17 cran (@0.2)
## bitops 1.0-6 2013-08-17 CRAN (R 3.3.0)
## caTools 1.17.1 2014-09-10 CRAN (R 3.4.0)
## colorspace 1.3-2 2016-12-14 CRAN (R 3.4.0)
## compiler 3.4.2 2017-10-28 local
## datasets * 3.4.2 2017-10-28 local
## devtools 1.13.3 2017-08-02 CRAN (R 3.4.1)
## digest 0.6.12 2017-01-27 CRAN (R 3.4.0)
## dplyr * 0.7.4 2017-09-28 cran (@0.7.4)
## evaluate 0.10.1 2017-06-24 cran (@0.10.1)
## gdata 2.18.0 2017-06-06 cran (@2.18.0)
## ggplot2 * 2.2.1 2016-12-30 CRAN (R 3.3.2)
## glue 1.1.1 2017-06-21 cran (@1.1.1)
## gplots * 3.0.1 2016-03-30 CRAN (R 3.4.0)
## graphics * 3.4.2 2017-10-28 local
## grDevices * 3.4.2 2017-10-28 local
## grid 3.4.2 2017-10-28 local
## gtable 0.2.0 2016-02-26 CRAN (R 3.3.0)
## gtools 3.5.0 2015-05-29 CRAN (R 3.4.0)
## highr 0.6 2016-05-09 CRAN (R 3.3.0)
## htmltools 0.3.6 2017-04-28 CRAN (R 3.4.0)
## KernSmooth 2.23-15 2015-06-29 CRAN (R 3.4.0)
## knitr 1.17 2017-08-10 cran (@1.17)
## labeling 0.3 2014-08-23 CRAN (R 3.3.0)
## lazyeval 0.2.0 2016-06-12 CRAN (R 3.4.0)
## magrittr 1.5 2014-11-22 CRAN (R 3.3.0)
## MASS * 7.3-47 2017-04-21 CRAN (R 3.4.0)
## memoise 1.1.0 2017-04-21 CRAN (R 3.4.0)
## methods * 3.4.2 2017-10-28 local
## munsell 0.4.3 2016-02-13 CRAN (R 3.3.0)
## nnet * 7.3-12 2016-02-02 CRAN (R 3.4.0)
## pkgconfig 2.0.1 2017-03-21 cran (@2.0.1)
## plyr 1.8.4 2016-06-08 CRAN (R 3.4.0)
## purrr 0.2.2.2 2017-05-11 cran (@0.2.2.2)
## R6 2.2.2 2017-06-17 cran (@2.2.2)
## RColorBrewer 1.1-2 2014-12-07 CRAN (R 3.3.0)
## Rcpp 0.12.13 2017-09-28 cran (@0.12.13)
## reshape * 0.8.7 2017-08-06 CRAN (R 3.4.1)
## reshape2 1.4.2 2016-10-22 CRAN (R 3.4.0)
## rlang 0.1.2.9000 2017-10-23 Github (hadley/rlang@cbdc3f3)
## rmarkdown 1.6 2017-06-15 CRAN (R 3.4.2)
## ROCR * 1.0-7 2015-03-26 CRAN (R 3.4.0)
## rprojroot 1.2 2017-01-16 CRAN (R 3.3.2)
## scales * 0.5.0 2017-08-24 CRAN (R 3.4.1)
## stats * 3.4.2 2017-10-28 local
## stringi 1.1.5 2017-04-07 CRAN (R 3.4.0)
## stringr 1.2.0 2017-02-18 CRAN (R 3.3.2)
## tibble * 1.3.4 2017-08-22 cran (@1.3.4)
## tidyr * 0.7.2 2017-10-16 CRAN (R 3.4.2)
## tools 3.4.2 2017-10-28 local
## utils * 3.4.2 2017-10-28 local
## withr 2.0.0 2017-07-28 CRAN (R 3.4.1)
```