

Neural Nets: A Simple Example

Hugh Chipman
Acadia University

Review: form of neural net model

- Neural networks have an output with functional form

$$\Psi \left[b_0 + \sum_i b_i \Phi(w_{i0} + \sum_j w_{ij} x_j) \right]$$

with Ψ, Φ known functions.

- Here, x_j is the j^{th} feature or input.
- In this example I am considering 2-class classification, so we'll take

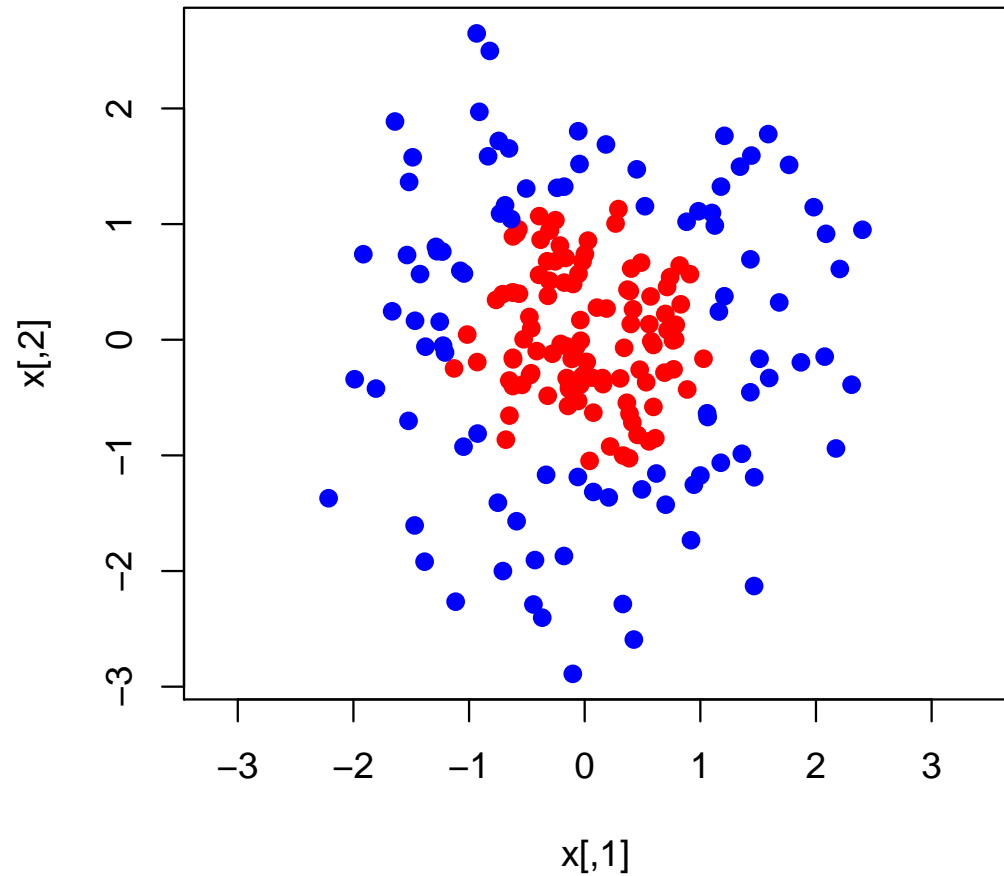
$$\Phi(x) = \Psi(x) = \frac{\exp(x)}{1 + \exp(x)},$$

the logistic or sigmoid activation function.

- This means that our output is the predicted probability of belonging to class 1.
- The purpose of my talk is to give you an idea of how a neural net represents functions, and related issues.

An example:

- Two class classification
- Features are simulated as $N(0,1)$
- 200 observations
- All points within a circle of radius roughly 1.2 belong to class 0 (red)

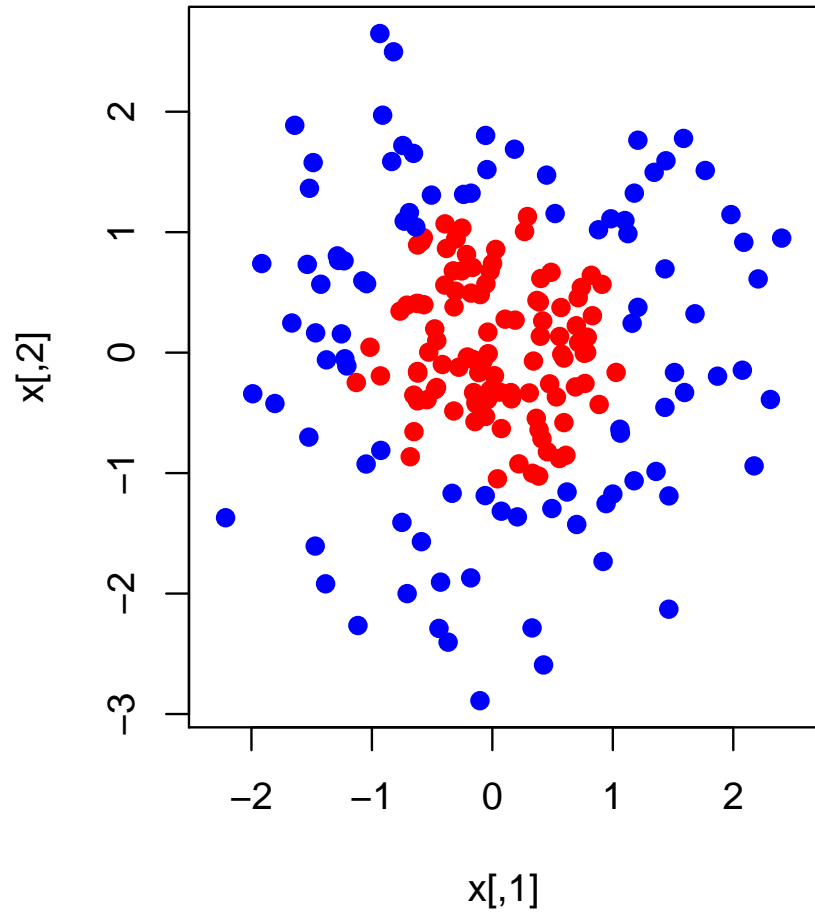


Neural net: attempt 1

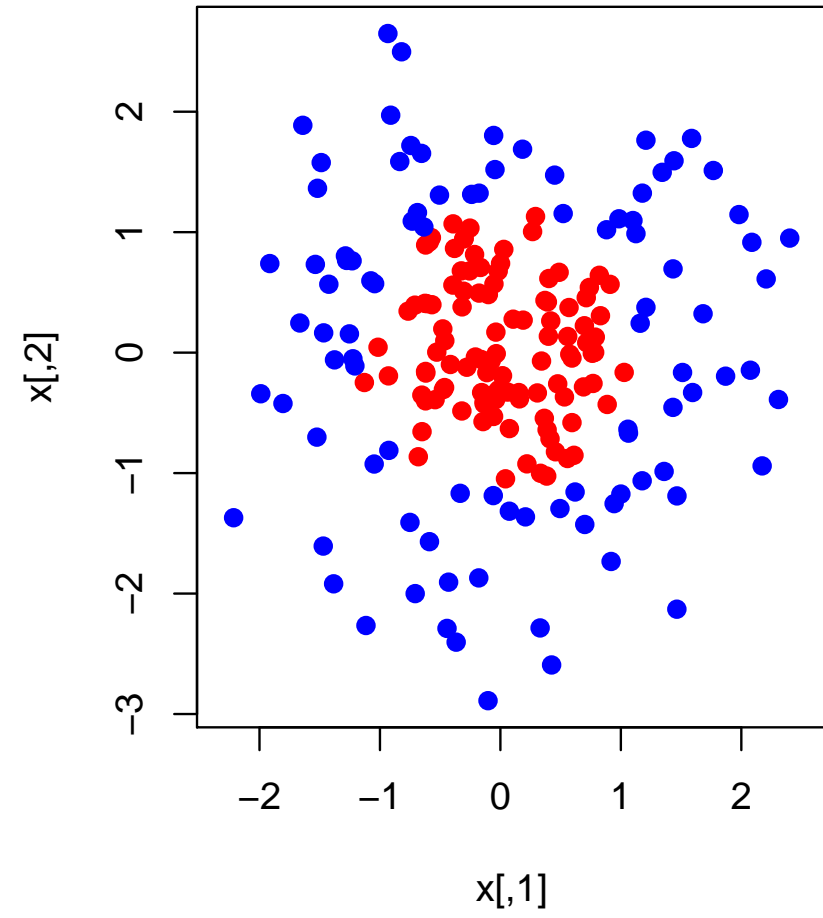
I fit a neural network to this data:

- Used `nnet` library in R.
- 3 hidden units
- logistic (a.k.a. sigmoid) activation functions
- **It works!** For the training set, predicted class = actual class for all 200 observations. See plots next page...

actual labels



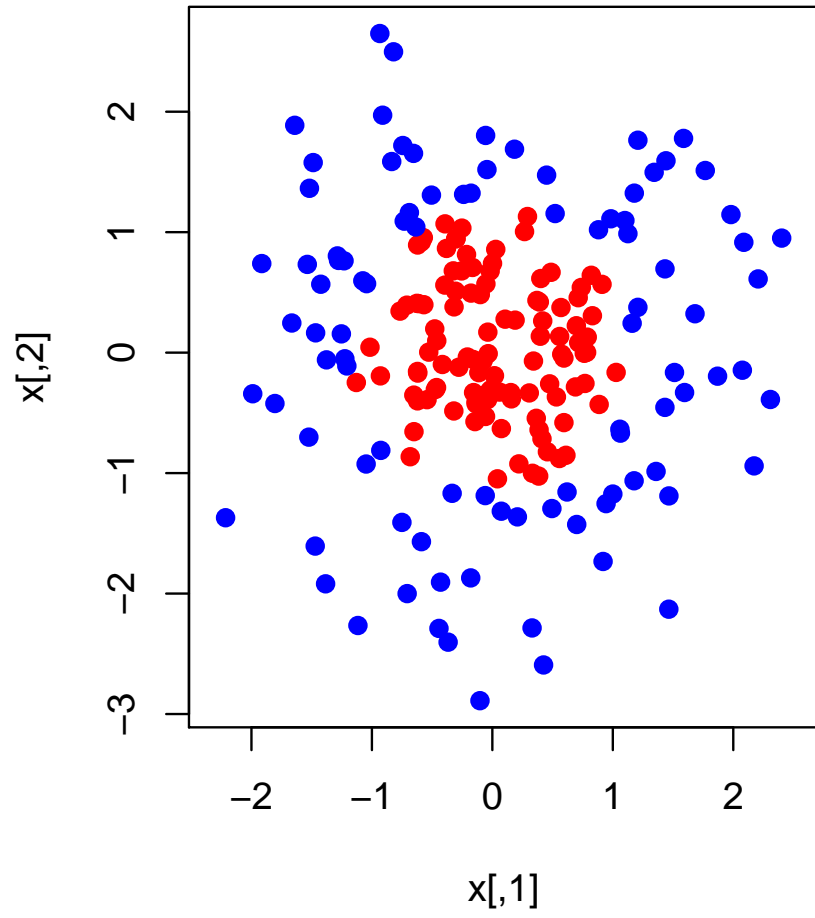
predicted labels



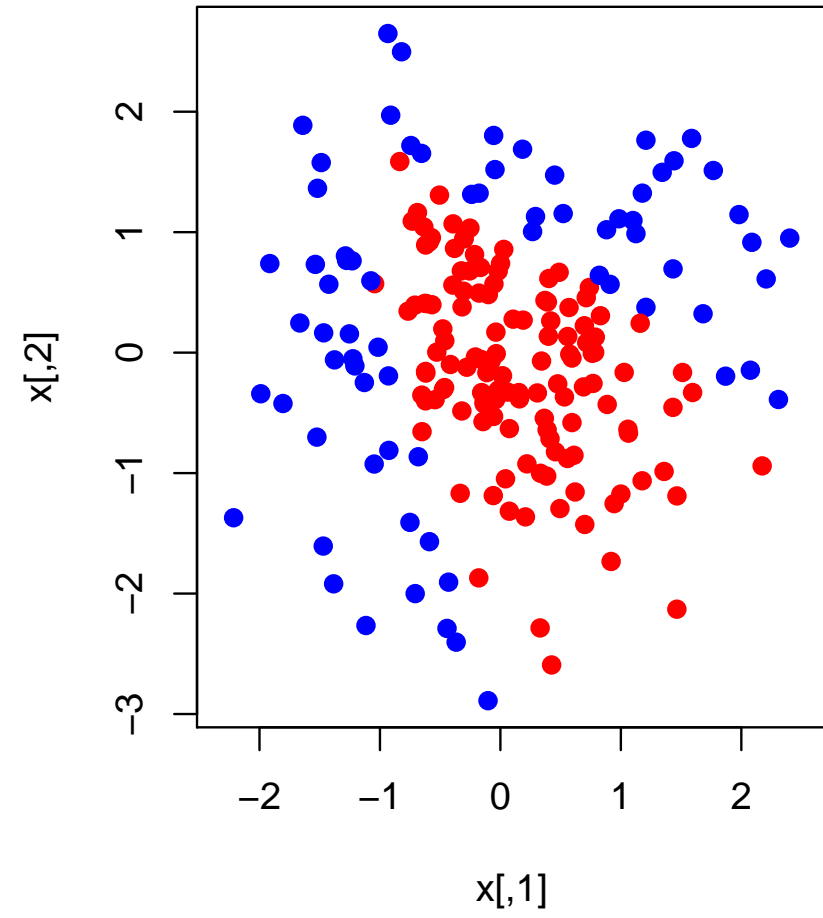
Trouble in paradise: local optima

- Fitting a neural net is a complicated optimization problem.
 - There are many local optima.
- The optimization algorithm that fits neural nets (BFGS, a gradient-descent method) uses random starting values for the parameters (weights), and can find different solutions for different starting points.
- Here's the result of another iteration. It found a (bad) local optimum (right hand plot on next page).

actual labels



predicted labels



Trouble in paradise: local optima

So what should we do? How do we know we have a good optimum? Is this the end of the world as we know it?

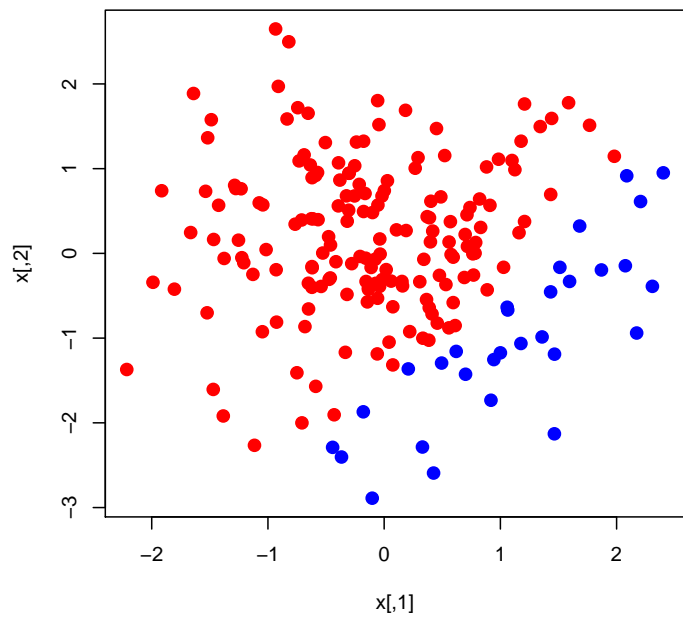
Common solutions:

- Run optimizer from multiple start points and pick best solution.
- Some tuning helps:
 - scale the inputs
 - choose sensible starting values
 - regularization can help stabilize as well (weight decay)

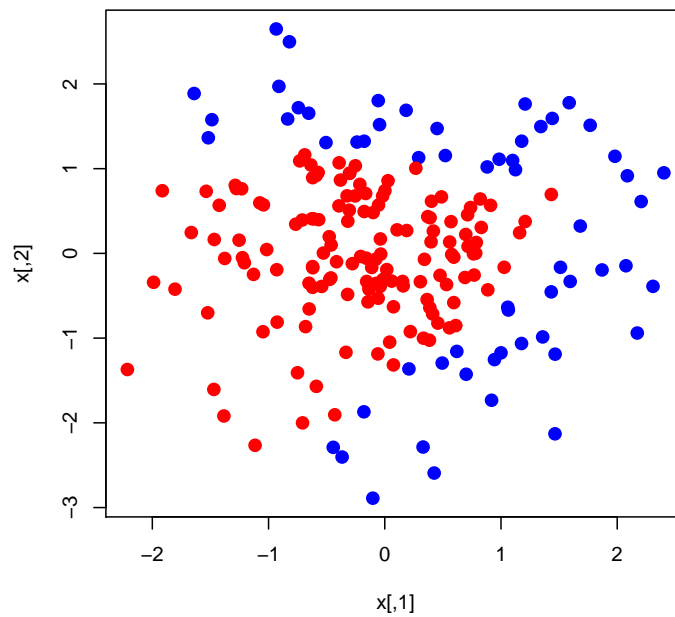
Another question: How many hidden units?

- Rather than think about it, let's just fit several neural nets with differing numbers of hidden units.
- I'll consider 1, 2, 3, and 4 hidden units.
- Results are on next page. Based on this, how many do we need?

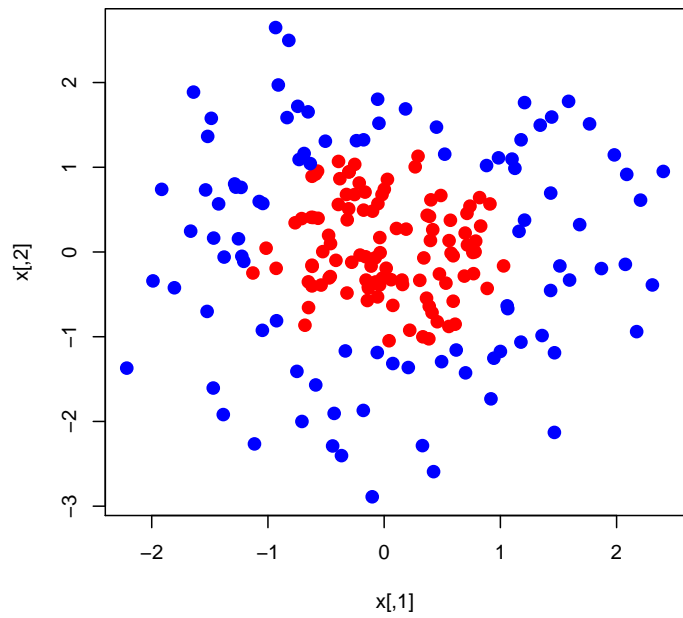
nnet with 1 hidden units



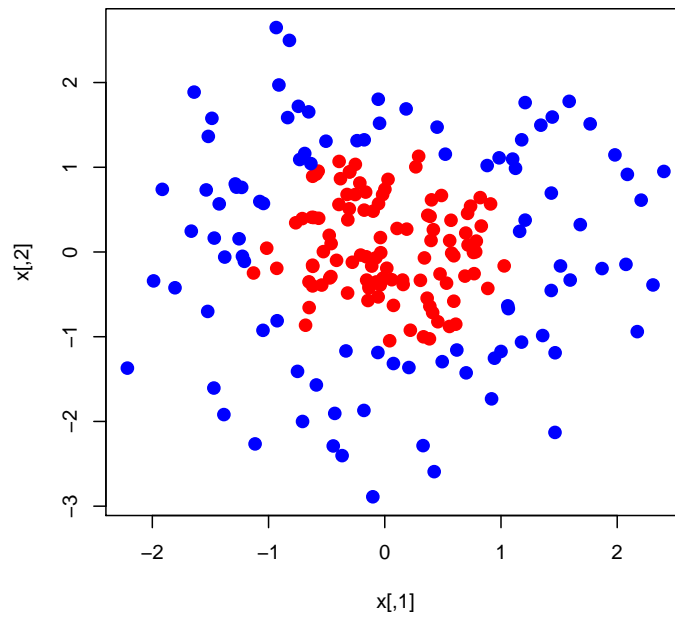
nnet with 2 hidden units



nnet with 3 hidden units

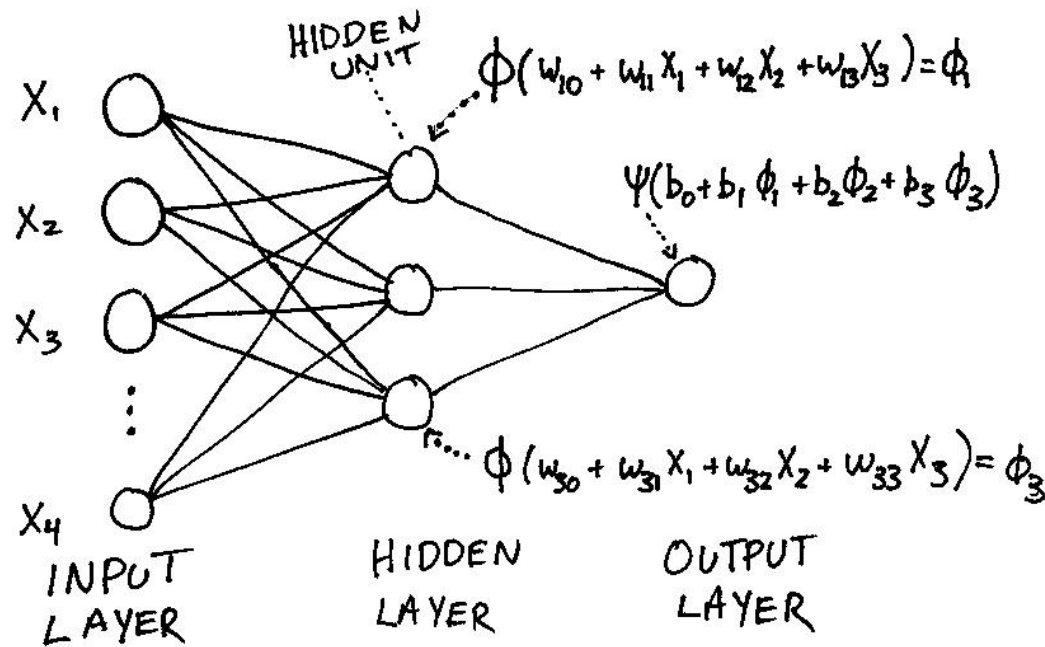


nnet with 4 hidden units



So 3 hidden units is enough. Why??

To explore this question we'll actually dig into the form of the neural net model, and look at the estimated model.



Here's the form of the output again:

$$\psi \left[b_0 + \sum_i b_i \phi(w_{i0} + \sum_j w_{ij}x_j) \right]$$

A neural net takes:

A nonlinear function of ...

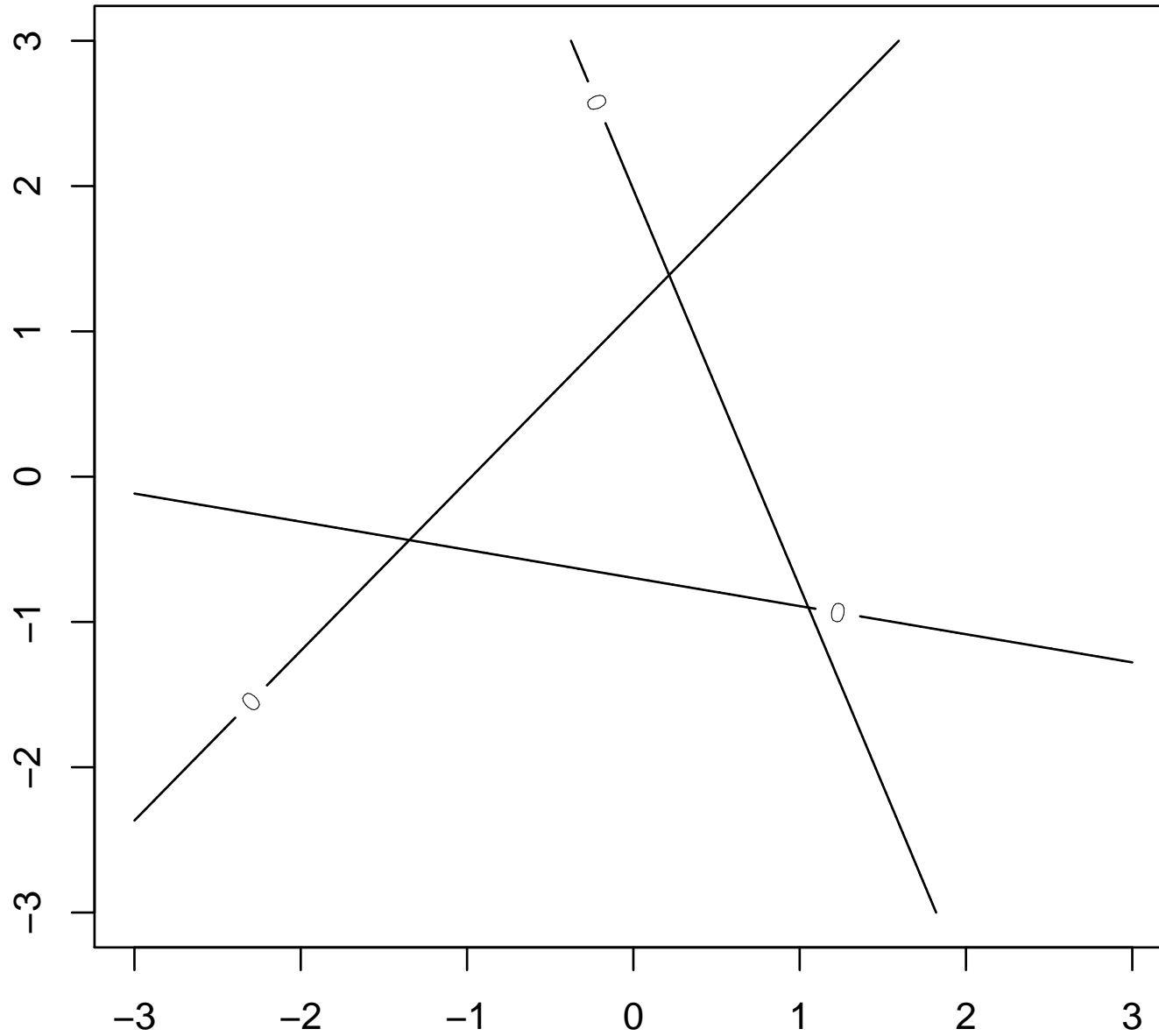
A linear combination of...

Nonlinear transformations of ...

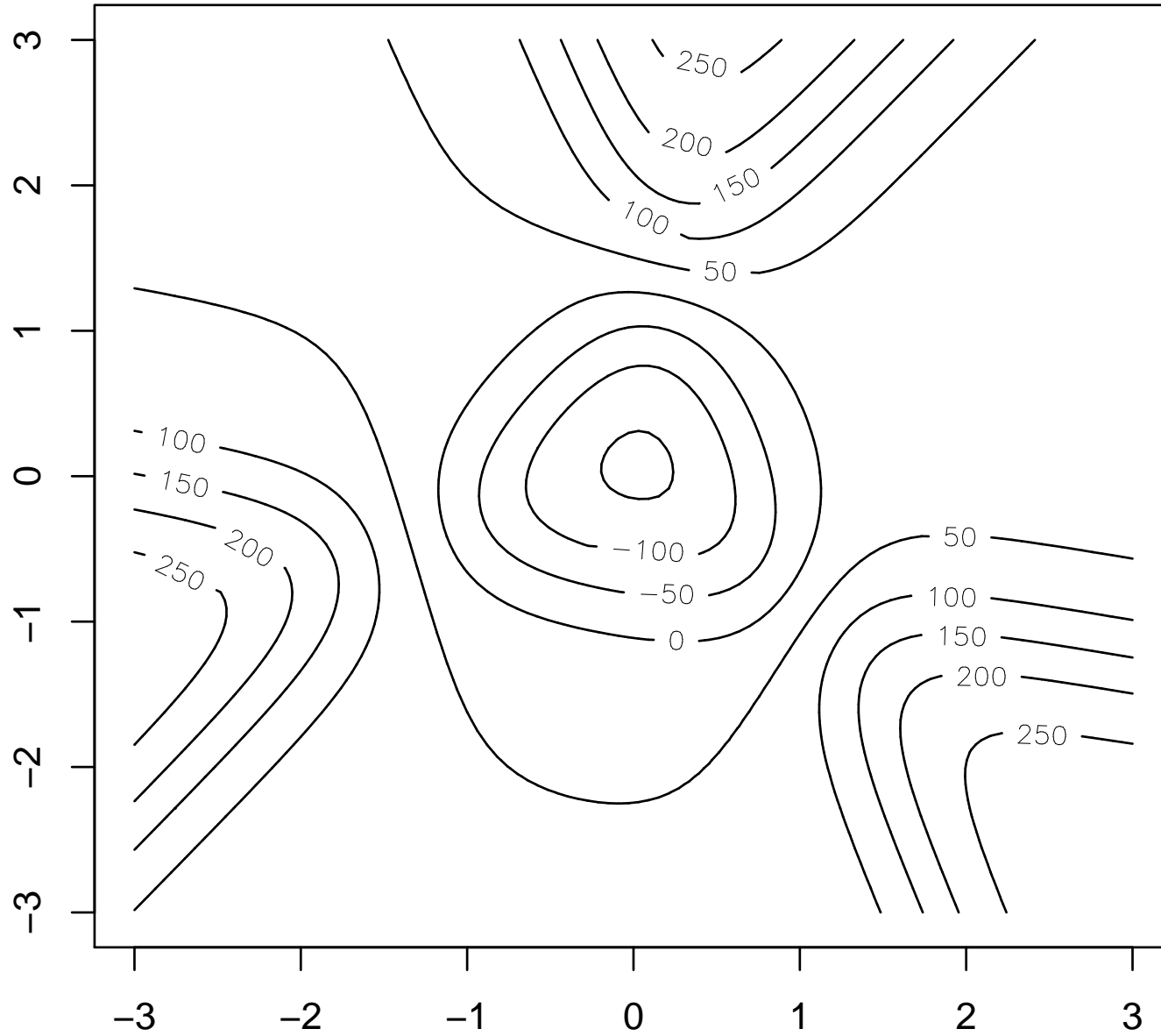
Several linear combinations of ...

the original inputs

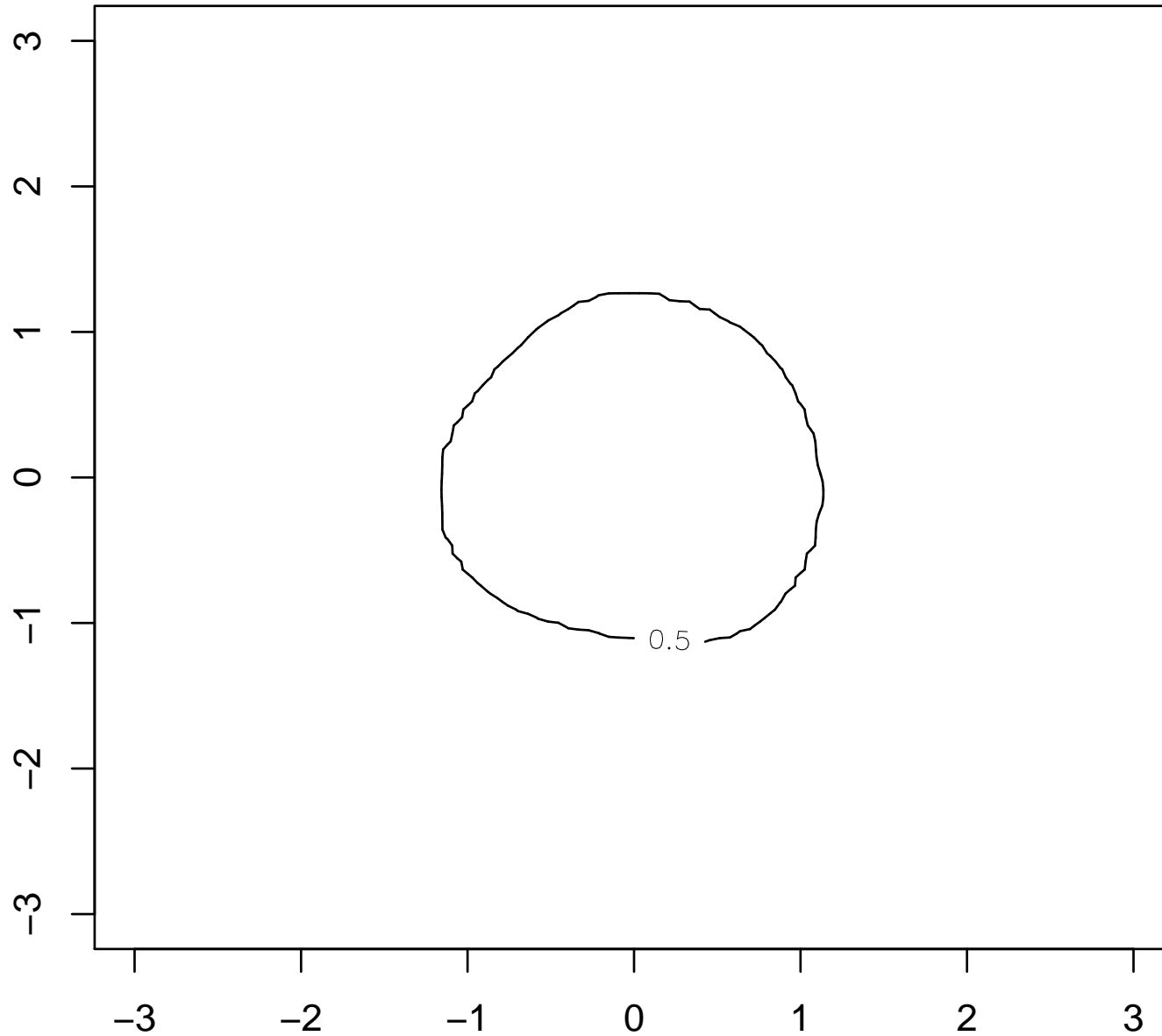
boundaries from linear functions in hidden units



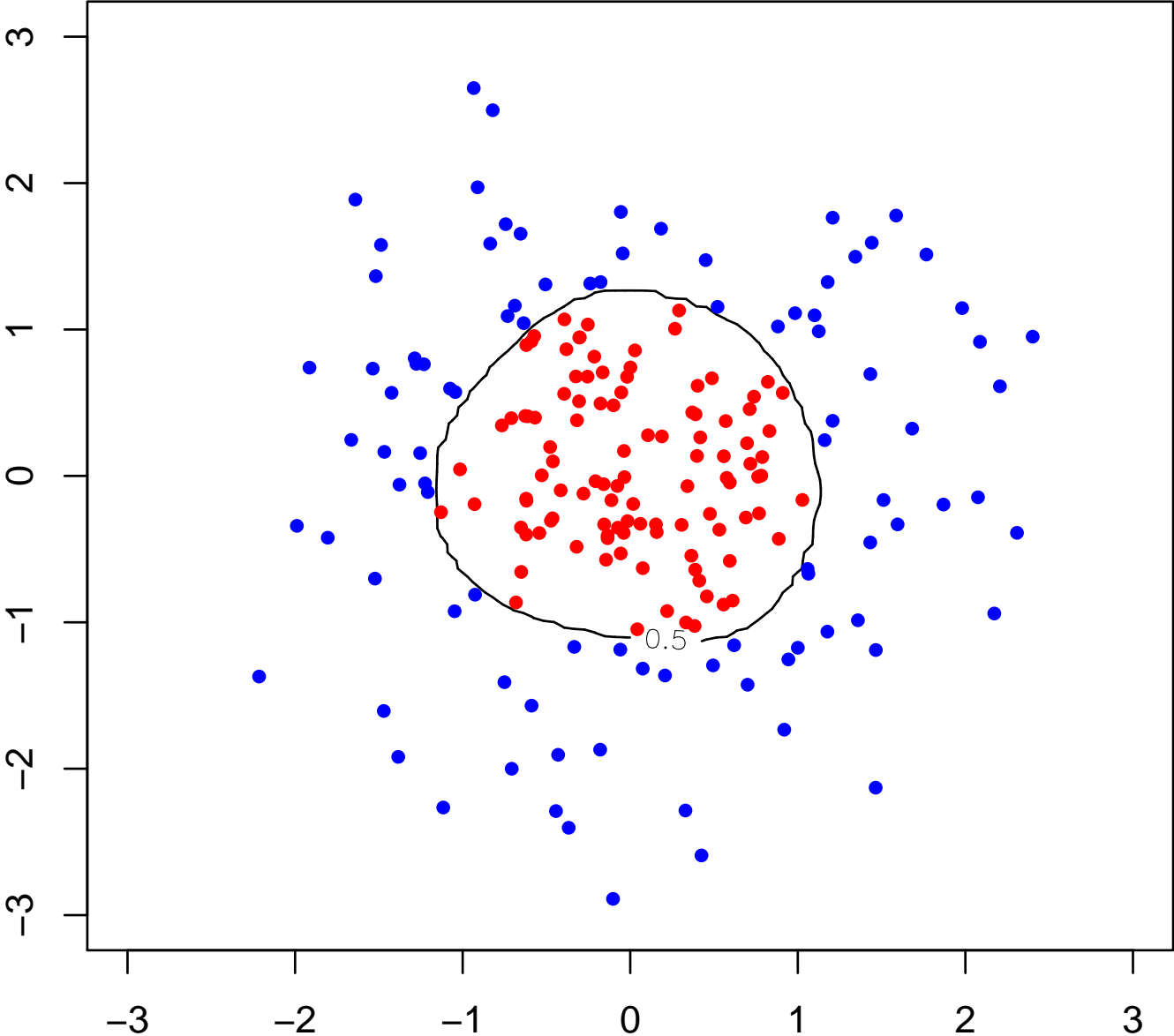
sum of sigmoids of linear functions



sigmoid of sum of sigmoids of linear functions



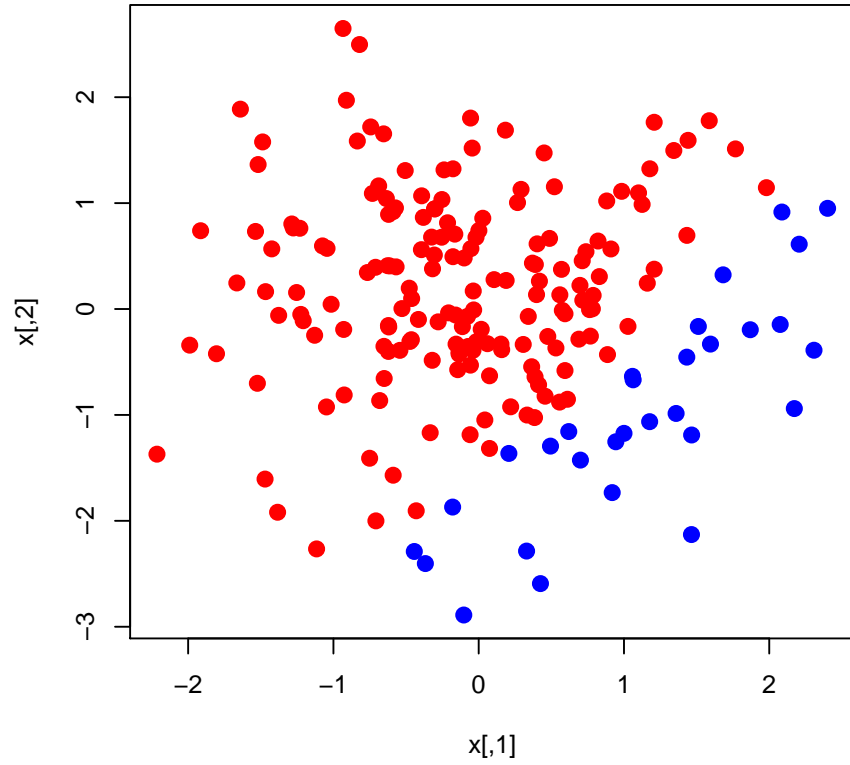
data values



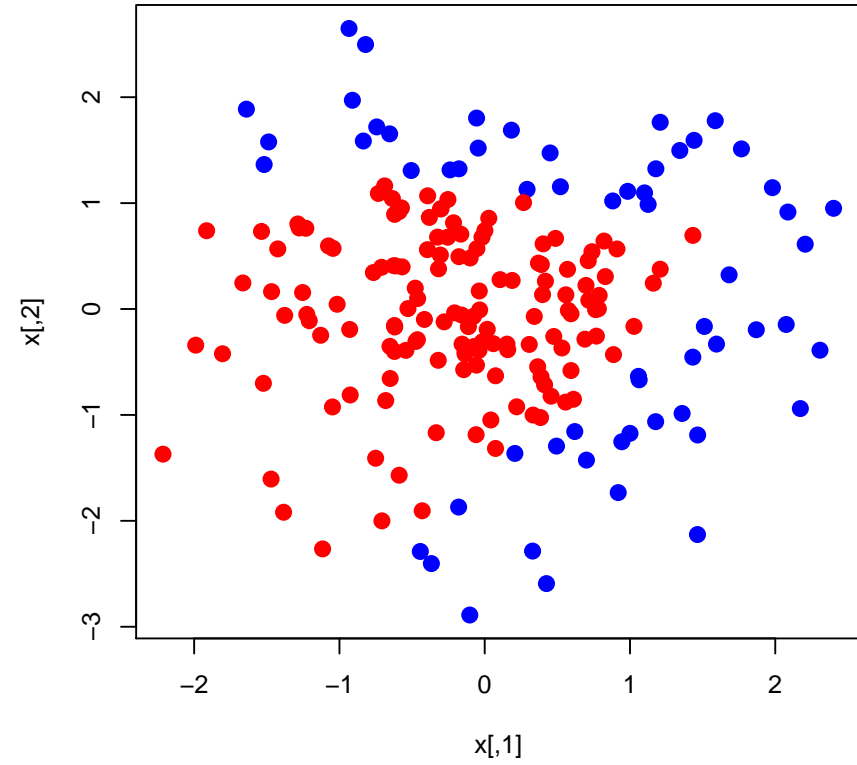
Can you see why 1 and 2 hidden units fail?

Look at the plots of fitted values again...

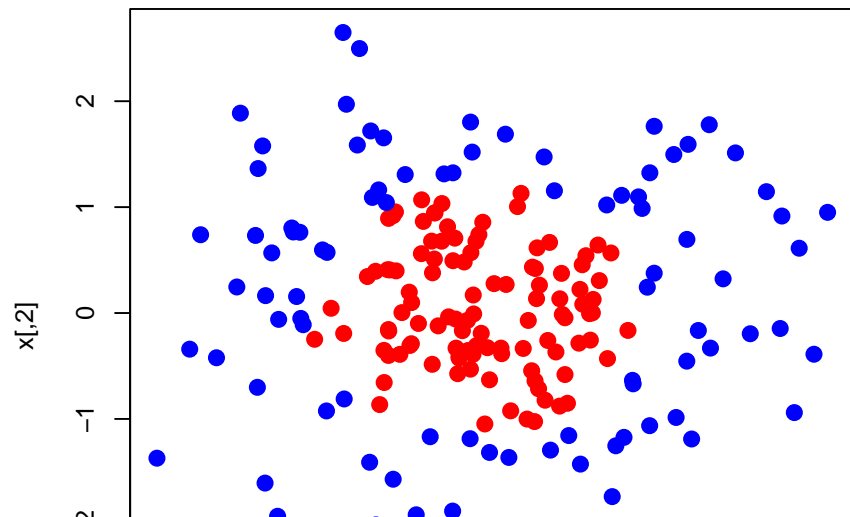
nnet with 1 hidden units



nnet with 2 hidden units



nnet with 3 hidden units



nnet with 4 hidden units

