INFSCI 2595

Fall 2019

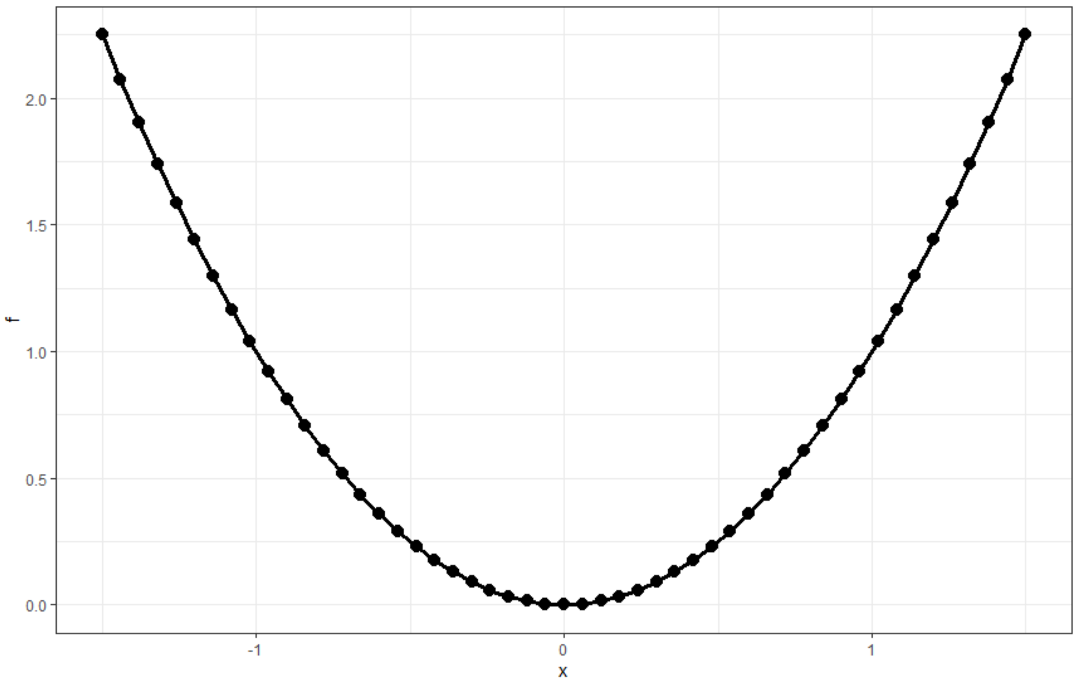
Information Sciences Building: Room 403

Week 10: Neural Networks

Demystify the neural network

- The (artificial) neural network and deep learning in general receive a lot of attention and hype.
- Neural networks are powerful but are not magical.
- They are statistical models, based on the principals and tools you have seen in this course.

We will build up a neural network through a simple demonstration problem



- Goal is to approximate the simple quadratic function:

$$f(x) = x^2$$

- Start with 51 noise-free points evenly spaced between -1.5 and 1.5.

Neural network architecture

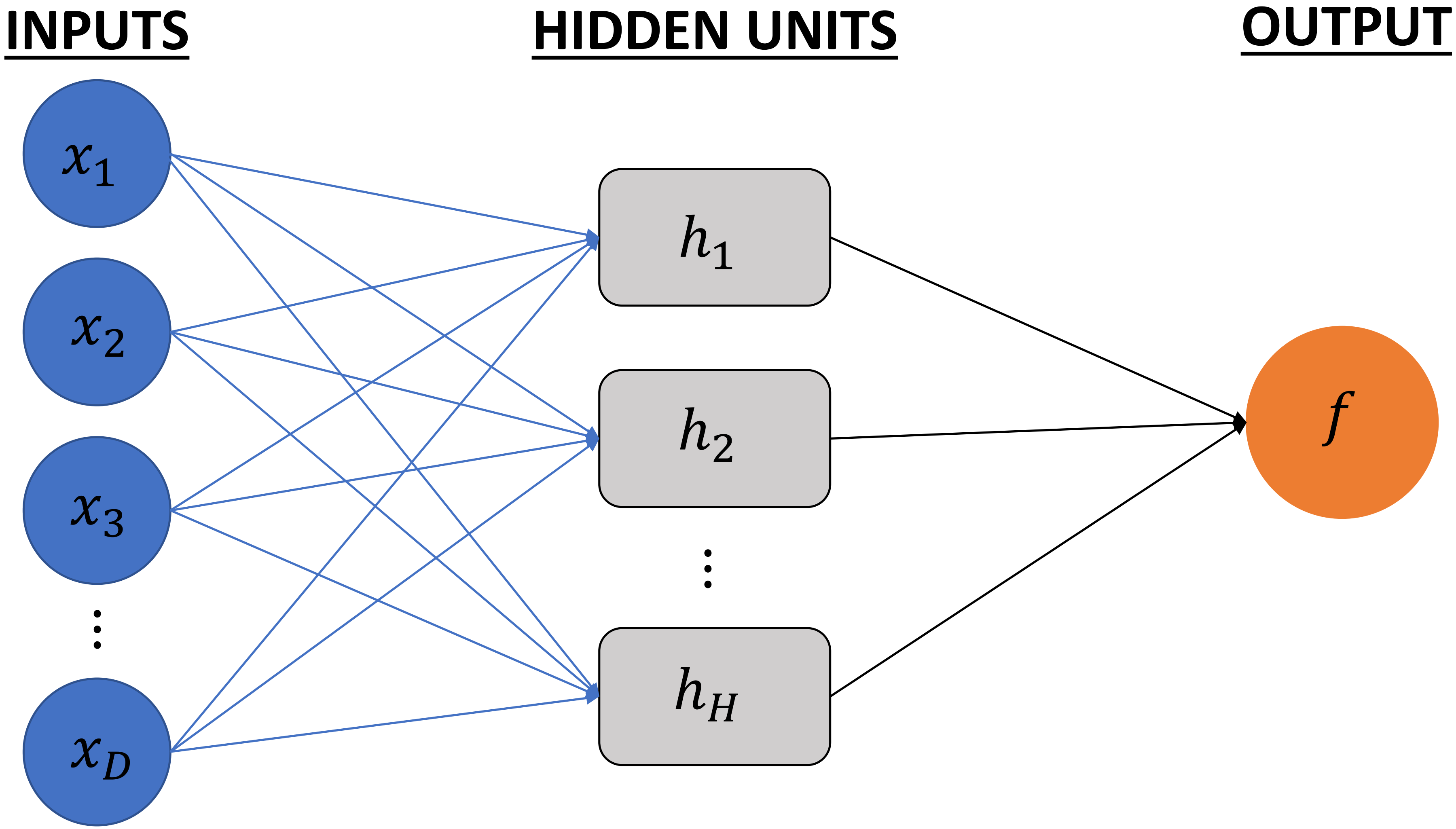
- A neural network is a series of functional transformations.

- The output is modeled through an intermediary set of unobserved variables called **hidden units**.

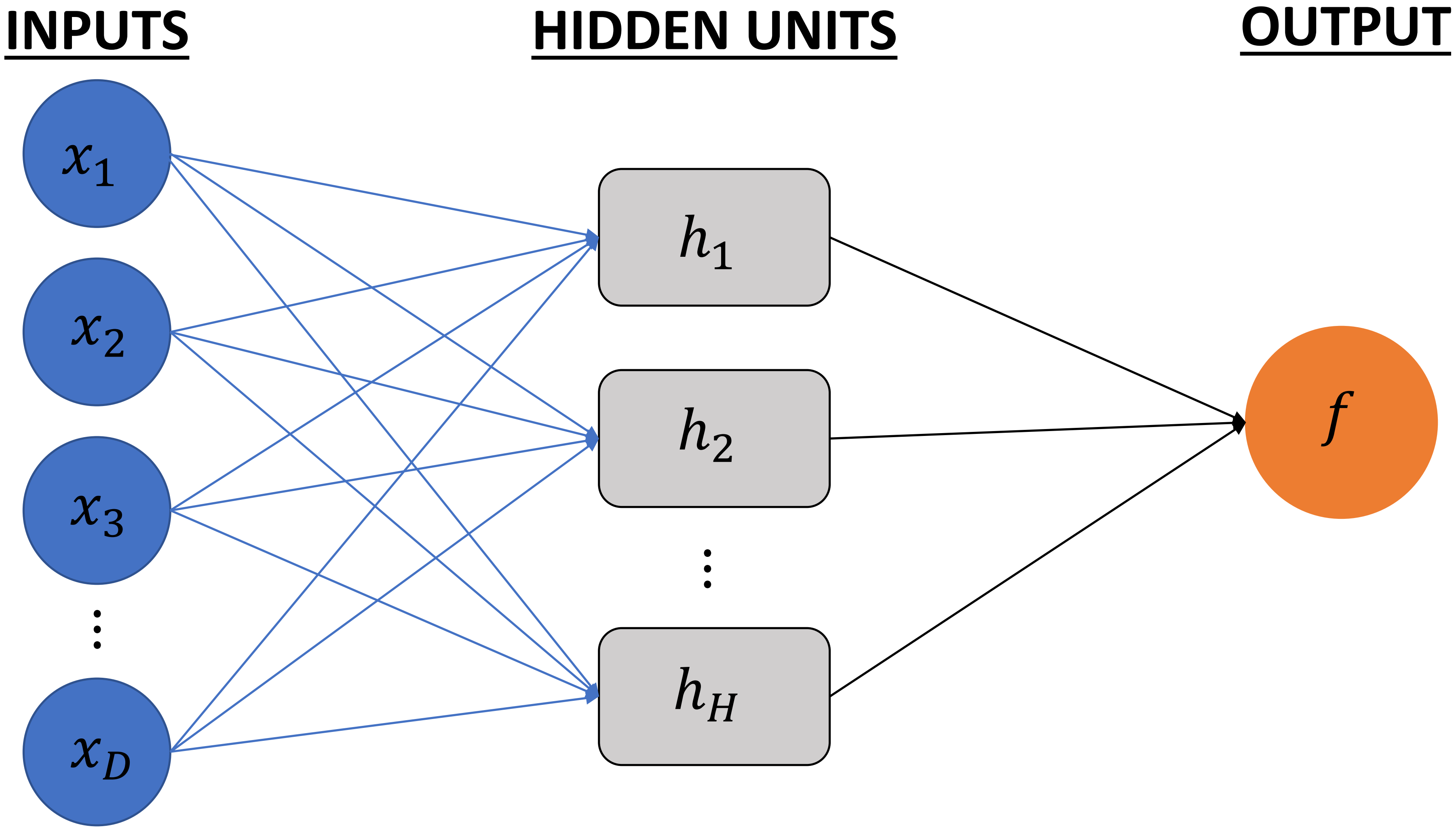
- The hidden units are derived as linear combinations of the inputs, transformed through a non-linear function.

- The response is modeled as a combination of the **hidden units**.

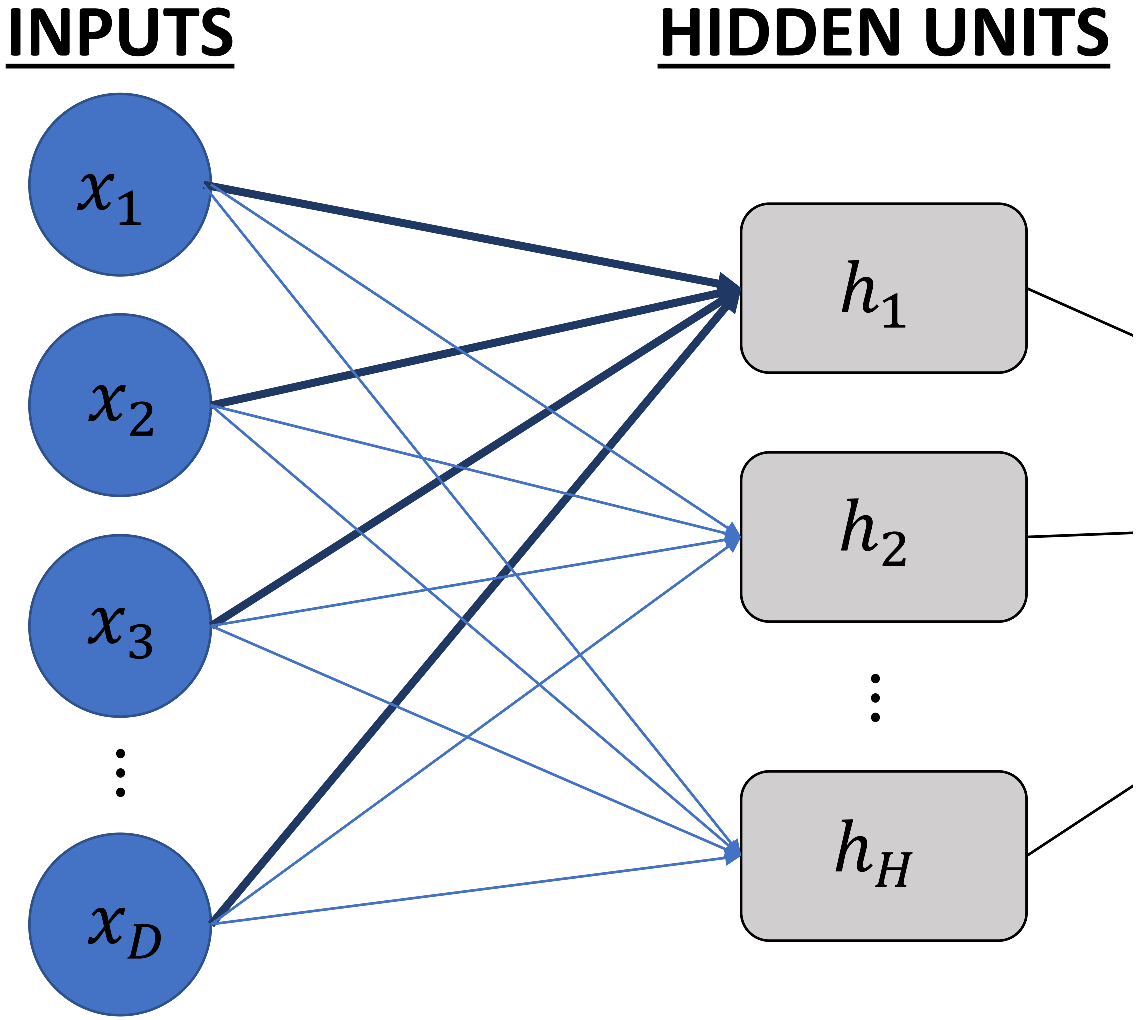
Schematic of a single layer neural network for a continuous response



Hidden units are non-linear functions of linear combinations of the inputs



Hidden unit 1



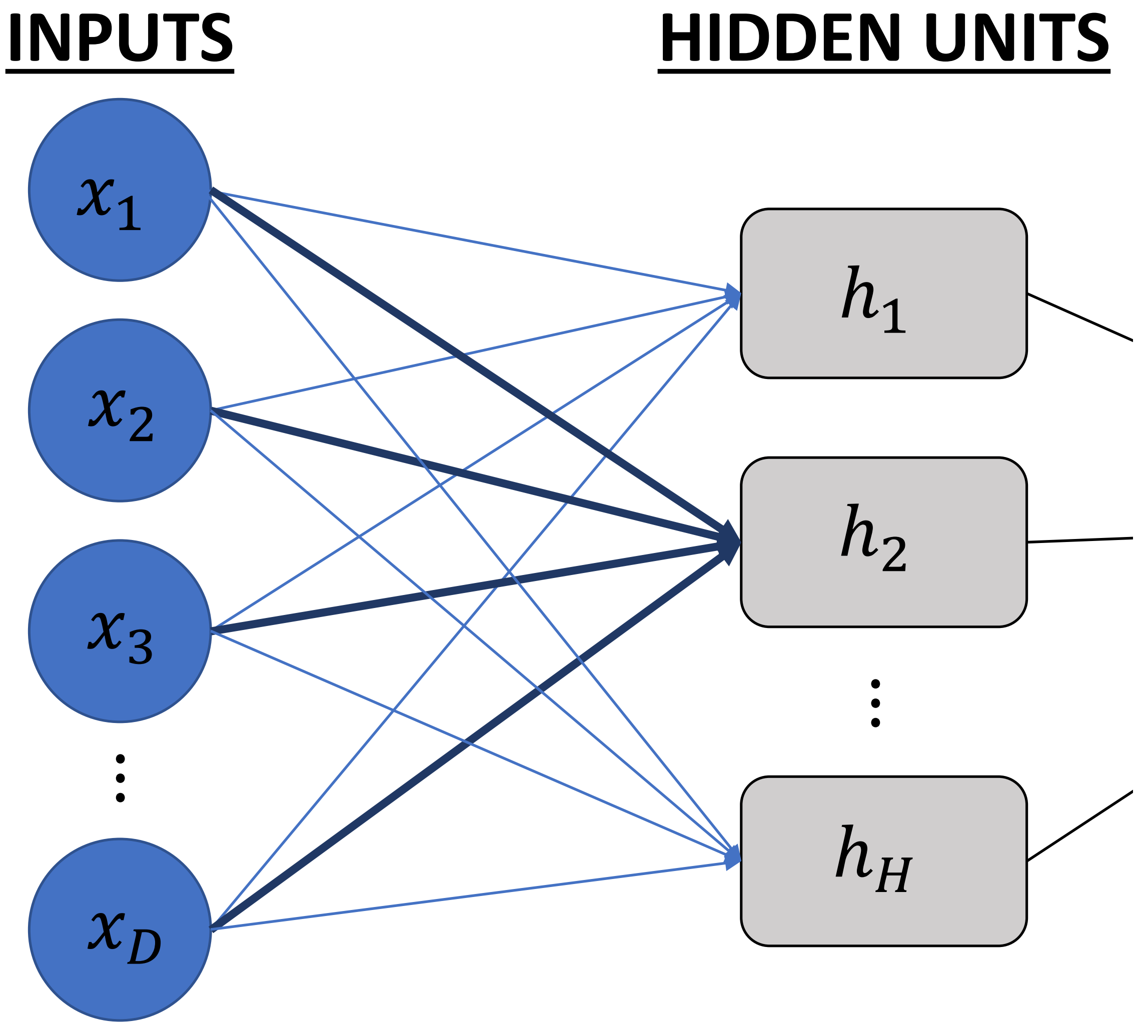
Linear combination of the inputs:

$$\beta_{0,1} + \sum_{d=1}^{D} x_d \beta_{d,1}$$

Pass through a non-linear function:

$$h_1(\mathbf{x}) = g\left(\beta_{0,1} + \sum_{d=1}^{D} x_d \beta_{d,1}\right)$$

Likewise for hidden unit 2



Linear combination of the inputs:

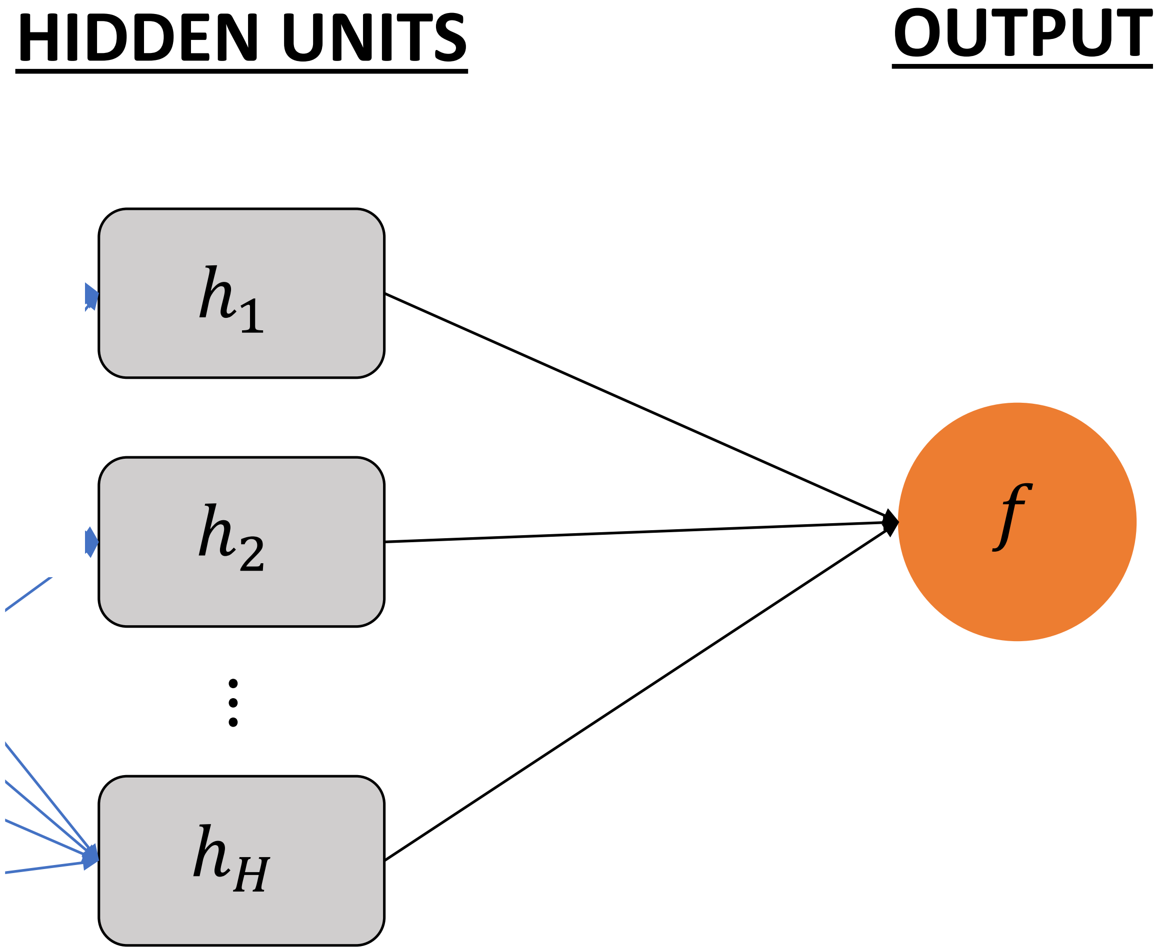
$$\beta_{0,2} + \sum_{d=1}^{D} x_d \beta_{d,2}$$

Pass through a non-linear function:

$$h_2(\mathbf{x}) = g\left(\beta_{0,2} + \sum_{d=1}^{D} x_d \beta_{d,2}\right)$$

For a continuous response, the output is a linear combination of the hidden units

$$f(\mathbf{x}) = \alpha_0 + \sum_{k=1}^{H} \alpha_k h_k$$



The linear combinations of the inputs are essentially linear models!

- If we have N training points of D inputs, we can assemble those inputs into the $N \times (D+1)$ design matrix, $\mathbf{X}$ (include the intercept column of 1s).

- The k-th hidden unit's parameters are stored in a $(D+1) \times 1$ column vector $\boldsymbol{\beta}_k$.

- The linear combination of the inputs for the k-th hidden unit:

$$\boldsymbol{\eta}_k = \mathbf{X}\boldsymbol{\beta}_k$$

There are a wide variety of non-linear transformation functions to use

- A common function is the logistic function!

$$g(u) = \frac{1}{1 + \exp(-u)} = \frac{\exp(u)}{\exp(u) + 1} = \text{logit}^{-1}(u)$$

- The k-th hidden unit is therefore equal to:

$$\mathbf{h}_k = \text{logit}^{-1}(\boldsymbol{\eta}_k)$$

The output layer

- Assemble the hidden unit variables into a matrix:

$$\mathbf{H} = [\mathbf{h}_1 \quad \mathbf{h}_2 \quad \cdots \quad \mathbf{h}_H]$$

- The output layer parameters, $\{\alpha_0, \alpha_1, \alpha_2, \ldots, \alpha_H\}$, are structured into a $H \times 1$ column vector $\boldsymbol{\alpha}$ and the scalar intercept, α_0.

- The response is a linear model relative to the hidden units!

$$\mathbf{f} = \alpha_0 + H\boldsymbol{\alpha}$$

The hidden linear models can be calculated all at once with matrix math!

- Bind the H hidden unit parameter vectors into a matrix:

$$\mathbf{B} = [\boldsymbol{\beta}_1 \quad \boldsymbol{\beta}_2 \quad \cdots \quad \boldsymbol{\beta}_H]$$

- The transformed hidden units can then be calculated as:

$$\mathbf{H} = \mathrm{logit}^{-1}(\mathbf{XB})$$

The neural network for a continuous response is fit by **minimizing** the sum of squared residuals

$$\sum_{n=1}^{N}\left((y_n - f_n)^2\right)$$

How many parameters are in the model?

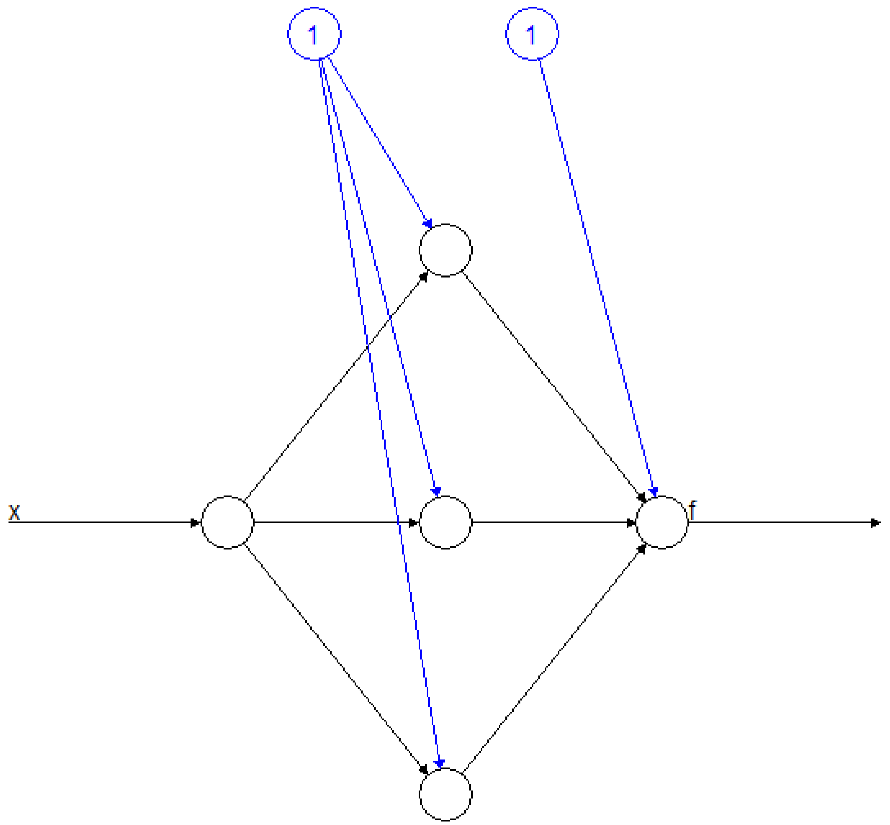
- $D + 1$ β-parameters in EACH hidden unit $\rightarrow H(D + 1)$ total parameters.
- $H + 1$ α-parameters in the output layer.
- TOTAL number of unknown parameters:

$$H(D + 1) + H + 1$$

For a **SINGLE LAYER** neural network!

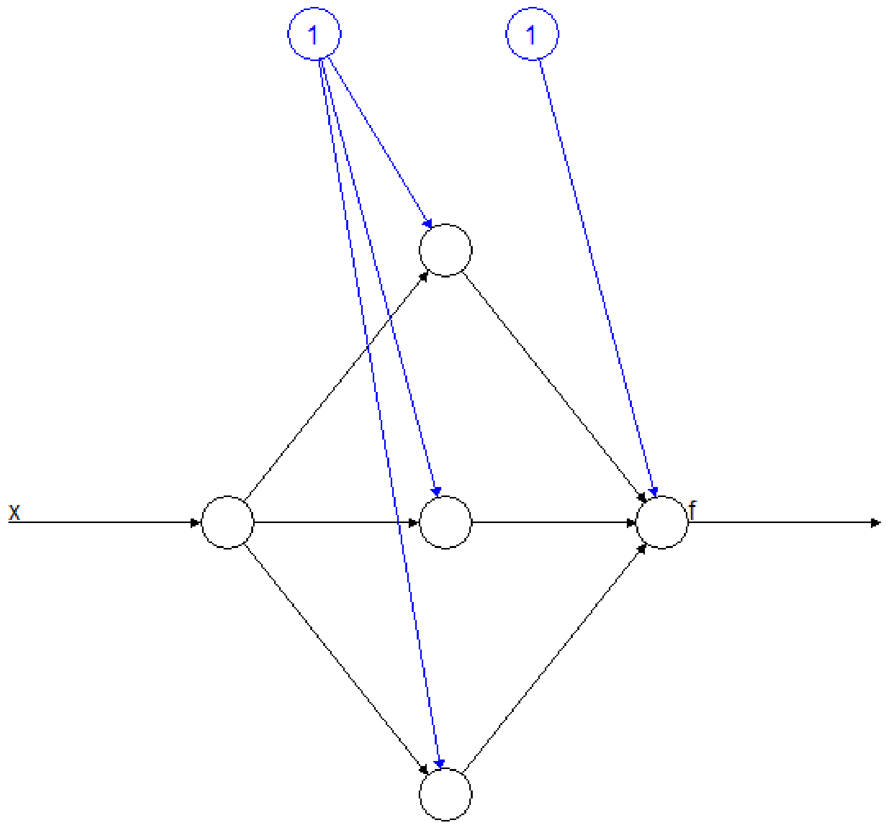
Now, let's apply our neural network to the simple quadratic example problem

- We will use a single layer with 3 hidden units.
- Single input x is on the left-hand side of the network.
- Single continuous response f is on the right-hand side.
- Intercepts (called biases) are either shown above the nodes or not shown at all.



Now, let's apply our neural network to the simple quadratic example problem

With $H = 3$ and $D = 1$ we have a total of **10** unknown parameters to learn!



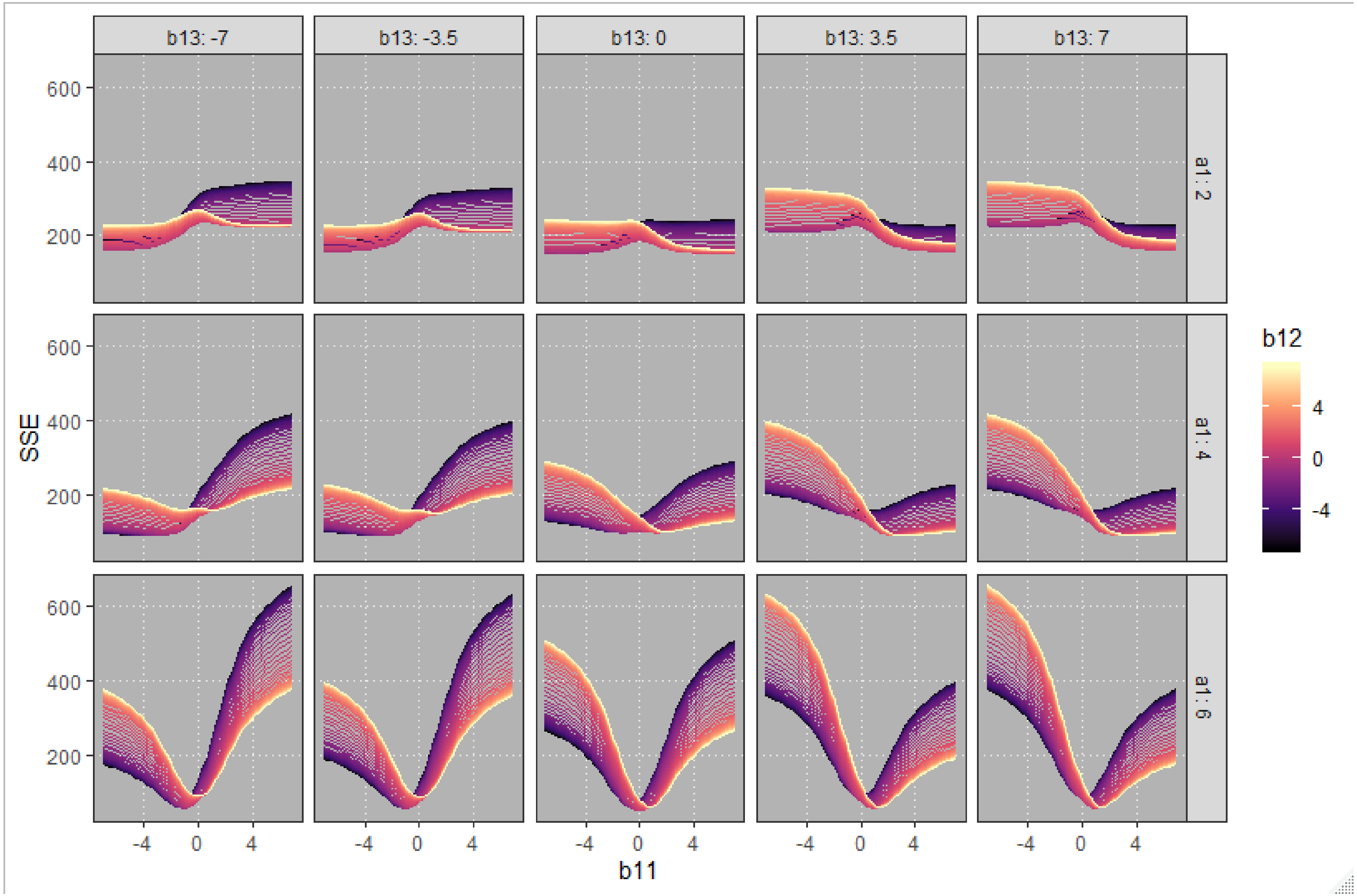
Let's get an idea about the behavior of the objective (the loss) function with respect to a few of the unknowns

- We have 10 unknowns:

- $\boldsymbol{\beta} = \{\beta_{0,1}, \beta_{1,1}, \beta_{0,2}, \beta_{1,2}, \beta_{0,3}, \beta_{1,3}\}$
- $\boldsymbol{\alpha} = \{\alpha_0, \alpha_1, \alpha_2, \alpha_3\}$

- Fix all parameters at -1, except $\beta_{1,1}$, $\beta_{1,2}$, $\beta_{1,3}$, and α_1

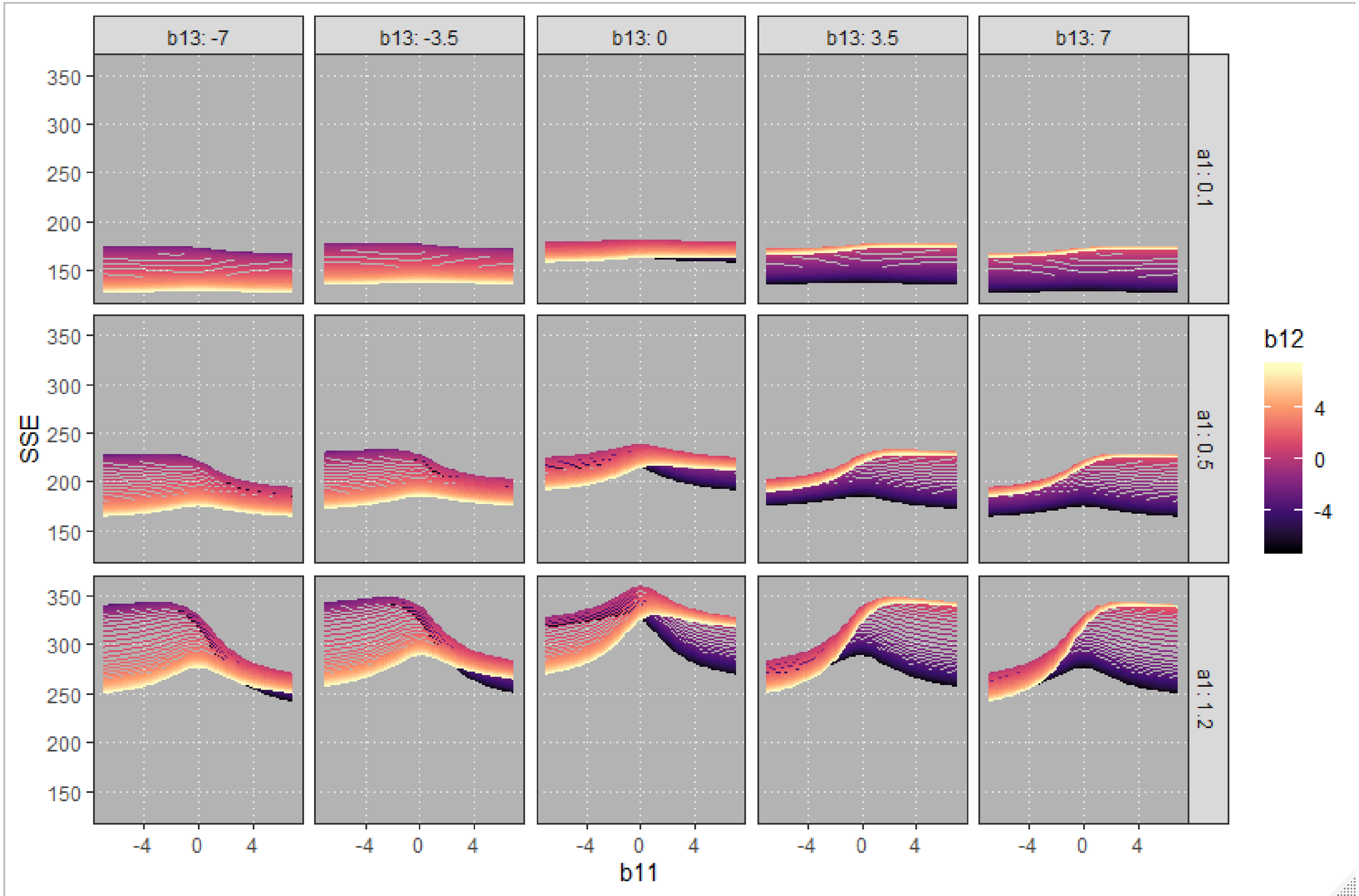
- Visualize the Sum of Squared Errors (SSE) wrt $\beta_{1,1}$



b13: -7
b13: -3.5
b13: 0
b13: 3.5
b13: 7
a1: 2
a1: 4
a1: 6
SSE
600
400
200
-4
0
4
b11
b12
4
0
-4

Try a different grid

- This time have the other parameters all fixed at +1



b13: -7
b13: -3.5
b13: 0
b13: 3.5
b13: 7
a1: 0.1
a1: 0.5
a1: 1.2
SSE
350
300
250
200
150
-4
0
4
b11
b12
4
0
-4

Difficult to visualize a 10 dimensional space!

- In-class live coding demo!