2.

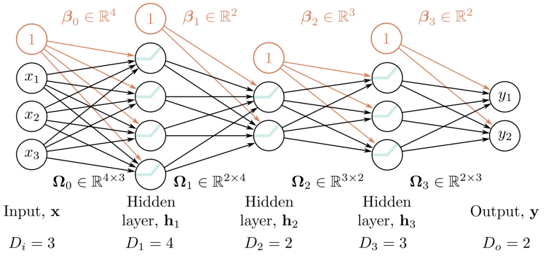
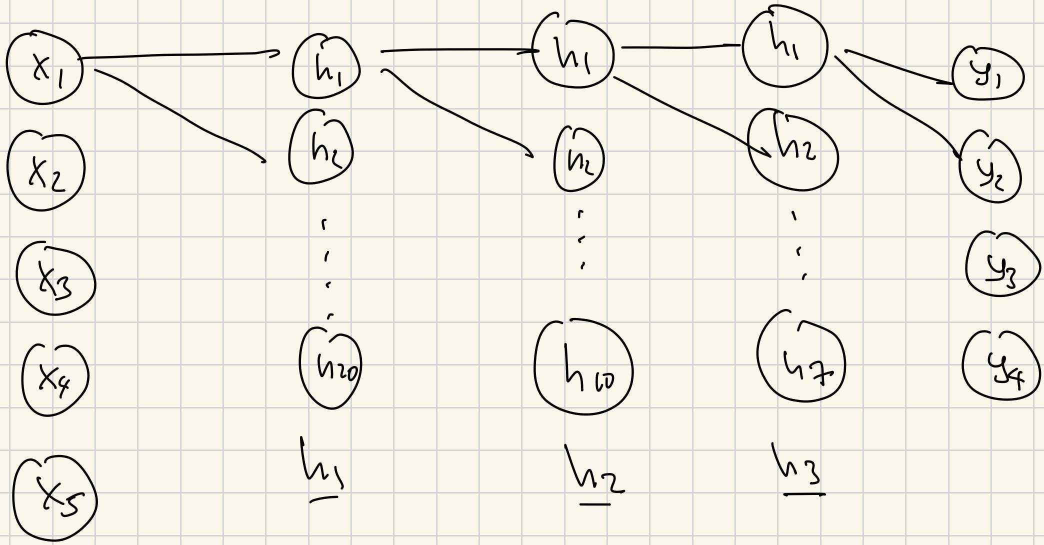


Figure 4.6 Matrix notation for network with $D_i = 3$-dimensional input $\mathbf{x}$, $D_o = 2$-dimensional output $\mathbf{y}$, and $K = 3$ hidden layers $\mathbf{h}_1, \mathbf{h}_2$, and $\mathbf{h}_3$ of dimensions $D_1 = 4$, $D_2 = 2$, and $D_3 = 3$ respectively. The weights are stored in matrices $\boldsymbol{\Omega}_k$ that multiply the activations from the preceding layer to create the pre-activations at the subsequent layer. For example, the weight matrix $\boldsymbol{\Omega}_1$ that computes the pre-activations at $\mathbf{h}_2$ from the activations at $\mathbf{h}_1$ has dimension 2×4. It is applied to the four hidden units in layer one and creates the inputs to the two hidden units at layer two. The biases are stored in vectors $\boldsymbol{\beta}_k$ and have the dimension of the layer into which they feed. For example, the bias vector $\boldsymbol{\beta}_2$ is length three because layer $\mathbf{h}_3$ contains three hidden units.

- Number of layers
- Hidden units in layer 1
- Hidden units in layer 2
- Hidden units in layer 3

4.



$$\underline{h}_1 = a[\beta_0 + \underline{\Omega}_0 \underline{x}]$$
$$\underline{h}_2 = a[\beta_1 + \underline{\Omega}_1 \underline{h}_1]$$
$$\underline{h}_3 = a[\beta_2 + \underline{\Omega}_2 \underline{h}_2]$$
$$\underline{y} = o[\beta_3 + \underline{\Omega}_3 \underline{h}_3]$$

Some output activation func.

$x \to 5\times1$
$\Omega_0 \to 20\times5$
$\beta_0 \to 20\times1$
$\Omega_1 \to 10\times20$
$\beta_1 \to 10\times1$
$\Omega_2 \to 7\times10$
$\beta_2 \to 7\times1$
$\Omega_3 \to 4\times7$
$\beta_3 \to 4\times1$

6. (Consider just weights, not biases).

There are currently,

$1 \times 10 + 9(10 \times 10) + 1 \times 10 = 10 + 900 + 10 = 920$ weights

Increasing a layer (depth) gives 100 more weights $= 1020$ weights

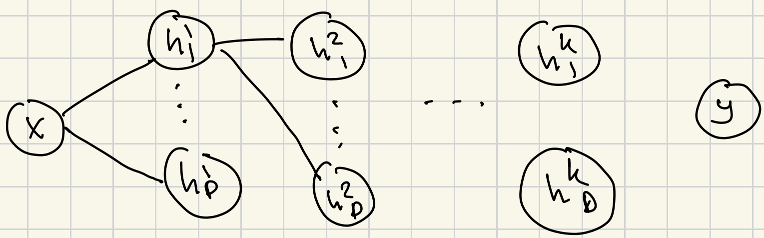
Increasing width (no nodes in each layer) gives

$1 \times 11 + 9(11 \times 11) + 1 \times 11 = 1111$ weights.

So, more weights added by increasing width.

8. In the 'odd' answers.

10. Looks like.



Between the input and layer 1 there are $D + D$ params.

Between final layer and output there are $D + 1$ params

Inbetween layers there are $(k-1)(D \times D + D)$ params.

this becomes

$$D + D + D + 1 + (k-1)(D \times D + D)$$

$= 3D + 1 + (k-1)D(D+1)$ parameters.