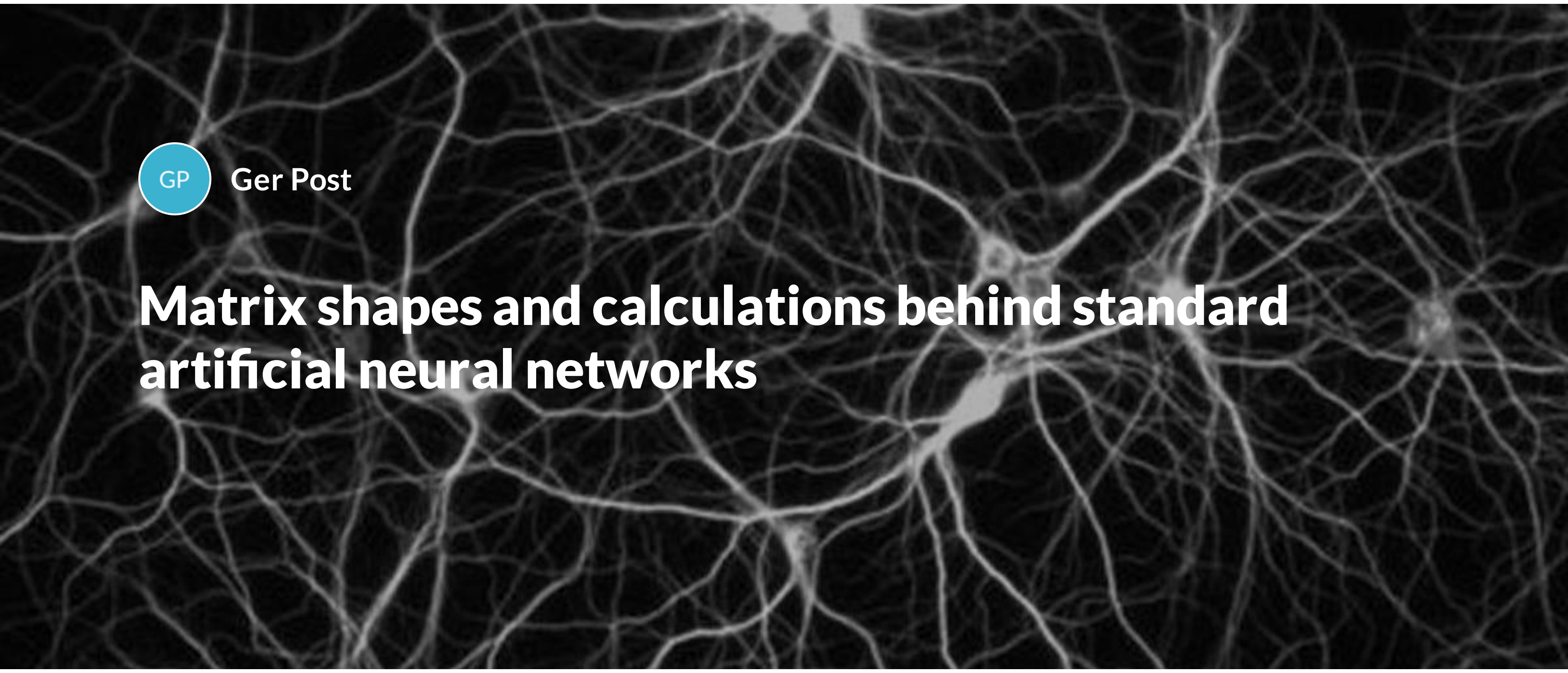


GP Ger Post

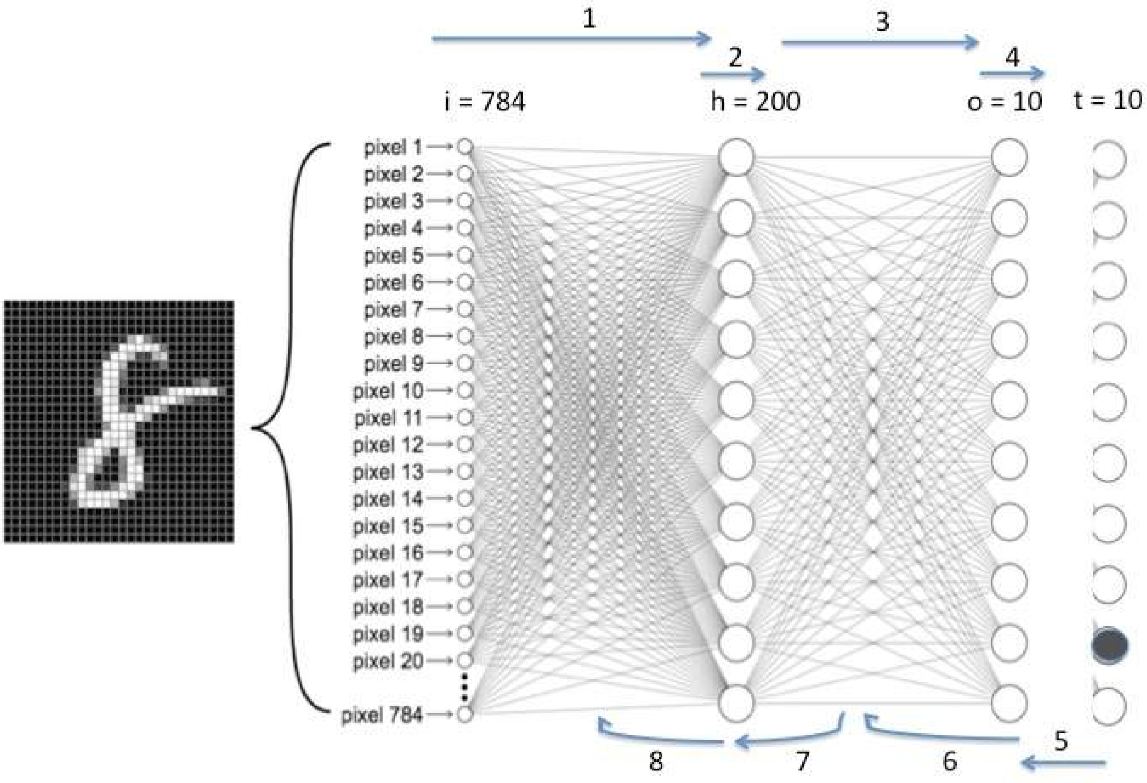
Matrix shapes and calculations behind standard artificial neural networks

THE UNIVERSITY OF MELBOURNE

The ANN and matrix calculations

The ANN and matrix calculations

GP Ger Post



i = input, h = hidden, o = output, t = target

For each step through the network (both the forward pass and back propagation) the code is given below. Write down for each step the dimensions of the matrix calculations. For step 1, for example:

wih weights * inputs = inputs to hidden layer

(200 x 784) * (784 x 1) = (200 x 1)

Note that the input matrix has been transposed from a (1 x 784) matrix in this bit of code (and that the same will happen later for targets):

- inputs = numpy.array(inputs_list, ndmin=2).^{T}
- targets = numpy.array(targets_list, ndmin=2).^{T}

If you get stuck you can check your answers at the bottom of this page.

What matrix product(s) are coded?

1. hidden_inputs = numpy.dot(self.wih,inputs)
2. hidden_outputs = self.activation_function(hidden_inputs)
3. final_inputs = numpy.dot(self.who, hidden_outputs)
4. final_outputs = self.activation_function(final_inputs)
5. output_errors = target - final_outputs
6. self.who += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)), numpy.transpose(hidden_outputs)
7. hidden_errors = numpy.dot(self.who^{T}, output_errors)
8. self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs * (1.0 - hidden_outputs)), numpy.transpose(inputs)

Check your answers below

1 hidden_inputs = numpy.dot(self.wih,inputs)

hidden inputs = weights wih * inputs

(200 x 784) * (784 x 1) = (200 x 1)

2 hidden_outputs = self.activation_function(hidden_inputs)

Shape of the matrix doesn't change

(200 x 1)

3 final_inputs = numpy.dot(self.who, hidden_outputs)

output (final) layer inputs = weights who * hidden outputs

(10 x 200) * (200 x 1) = (10 x 1)

4 final_outputs = self.activation_function(final_inputs)

(10 x 1)

5 output_errors = target - final_outputs

(10 x 1) - (10 x 1) = (10 x 1)

6 self.who += self.lr * numpy.dot((output_errors * final_outputs * (1.0 - final_outputs)), numpy.transpose(hidden_outputs)

new who = lr * weight adjustment

(10 x 200) += self.lr * (10 x 200)

weight adjustment who = output_errors * hidden_output^{T}

(10 x 1) * (1 x 200) = (10 x 200)

7 hidden_errors = numpy.dot(self.who^{T}, output_errors)

hidden errors = transpose of who * output errors

(200 x 10) * (10 x 1) = (200 x 1)

8

self.wih += self.lr * numpy.dot((hidden_errors * hidden_outputs * (1.0 - hidden_outputs)), numpy.transpose(inputs)

new wih = lr * weight adjustment

(200 x 784) += self.lr * (200 x 784)

weight adjustment wih = hidden_errors * input^T

(200 x 1) * (1 x 784) = (200 x 784)