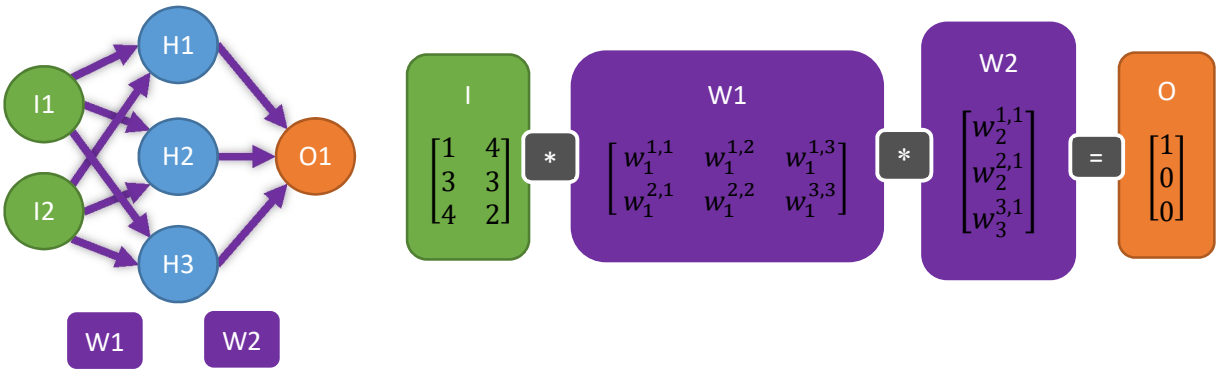
Hi, and welcome to the first challenge for the Advanced Ruby on Rails course. Normally we try to find training challenges that are already close to real world work. This time however we need to do some ground work to start with. We however will use this basic example to build upon for a real world challenge in the next round.

This challenge is quit complex. We will dig into one of the hot topics at the moment namely machine learning using neural networks. In this challenge you will build your very own implementation neural network. As this is not completely trivial we provide some source code and a data set to test your implementation. Our (a bit primitive) example implementation of the neural network achieved an average accuracy on this set of M = 0.97 (SD = 0.01) with 450 training iterations. Can you beat it? There are some really good tutorials out there on neural networks. We recommend building an easy one such as a feed forward network.

If you are not familiar with machine learning and neural networks just start the challenge and enjoy the ride. If you are unfamiliar with the concept please read on we will explain the idea of both concepts a little more in detail.

Imagine you want to predict if the next homework will be hard or easy. You want to predict that from two variables: teacher mood (1 grumpy to 5 happy) and class behavior (1 mean to 5 lovely). You have some samples from previous homework assignments from the same teacher. You can use a neural network to predict the next homework based on this data. One similar to the sketch below. We use two nodes in the input layer (green) one for each variable (teacher mood I1, and class behavior I2), a number of hidden nodes (H1-H3) in the second layer (blue), and a single node (O1) in the output layer (orange).



For a moment imagine arrows between these nodes just as in the figure. The information flows from the input nodes along these arrows to the output node. This is also called forward propagation. In reality these arrows are matrices of weights. The first matrix has as many rows as the previous layer nodes and as many columns as the next layer has nodes. This means that each node in the first layer is complete connected to the next layer. For W1 this means we have a matrix of the form 2 by 3. The second matrix w2 has the form 3 by 1. Each entry in this matrix is a weight. This weight defines how much of the value of the previous node is propagated to the next node. This process is also called forward propagation.

In our example the first column in the input matrix I is teacher mood the second row is class behavior. After applying our matrices we hope our final matrix O will give us the correct answers weather the homework was hard or easy. As the weight matrices are initialized randomly in most nets that is very

unlikely. To get the answers we expect we have to train our network. The process we use for the training is called backpropagation. For this process to work we let our network in its untrained state predict the outcome and calculate the difference between the prediction of our network and the actual observed results. This gives us an error term. With this error term we adapt the weights of each layer and then predict again until we achieve the results we expect.

This is a very crude explanation of how a neural network works but it gives you (hopefully) a slight understanding and a starting point to now investigate how to implement such a model.