COMP4702/COMP7703/DATA7703 - Machine Learning
Homework 6 - Neural Networks
Solutions

Marcus Gallagher

Core Questions

1. Write a short (approx. 2-4 sentences) explanation of why is it useful to have bias weights in a single or multi-layer perceptron (MLP)? Hint: use the example of a single neuron with a linear activation function and generalise to a larger network.

   From the point of view of the model (single neuron, linear activation), this boils down to the equation for a straight line. If it has no "intercept" term, the line is constrained to pass through the origin, so the bias adds this flexibility. The second point is that having the bias represented as weights (and "dummy inputs") allows the bias to be treated the same as the other adaptive parameters of the model and makes the maths look neater. The first point is a bit more important but a good response for either point is enough to be correct.

2. Figures 1-3 show some results of training an MLP on a 2D test classification problem. The data and all hyperparameter/training settings are identical apart from the random initialization of weights. Compare the results in a few sentences, specifically in terms of the discriminant function, the training and test error and the expected generalisation performance of the trained model.

   - Discriminant function: Fig1 has failed to capture the underlying distribution of the data. Fig2 and Fig3 do better - creating a polygon around the blue class.
   - Errors: Fig1 has poor training error and worse test error. Fig2 and 3 have much better errors - Fig3 is better on training error than Fig2 but worse on test error
   - Generalisation: the hexagon shape in Fig2 is closer to the presumably "true" circle containing the blue class and lower test error suggest better generalisation. Fig1 will generalise poorly.

3. Figure 4 shows another MLP training example. The setup is identical to Figures 1-3 except a `tanh()` activation function has been used instead of a rectified linear (`relu()`) function. Explain why the discriminant function looks different to those in Figures 1-3.

   The smooth sigmoidal-shaped tanh() has a bias towards inducing decision boundaries that are smooth curves, while the relu() is a piecewise linear function, which has a bias towards inducing piecewise linear decision boundaries.

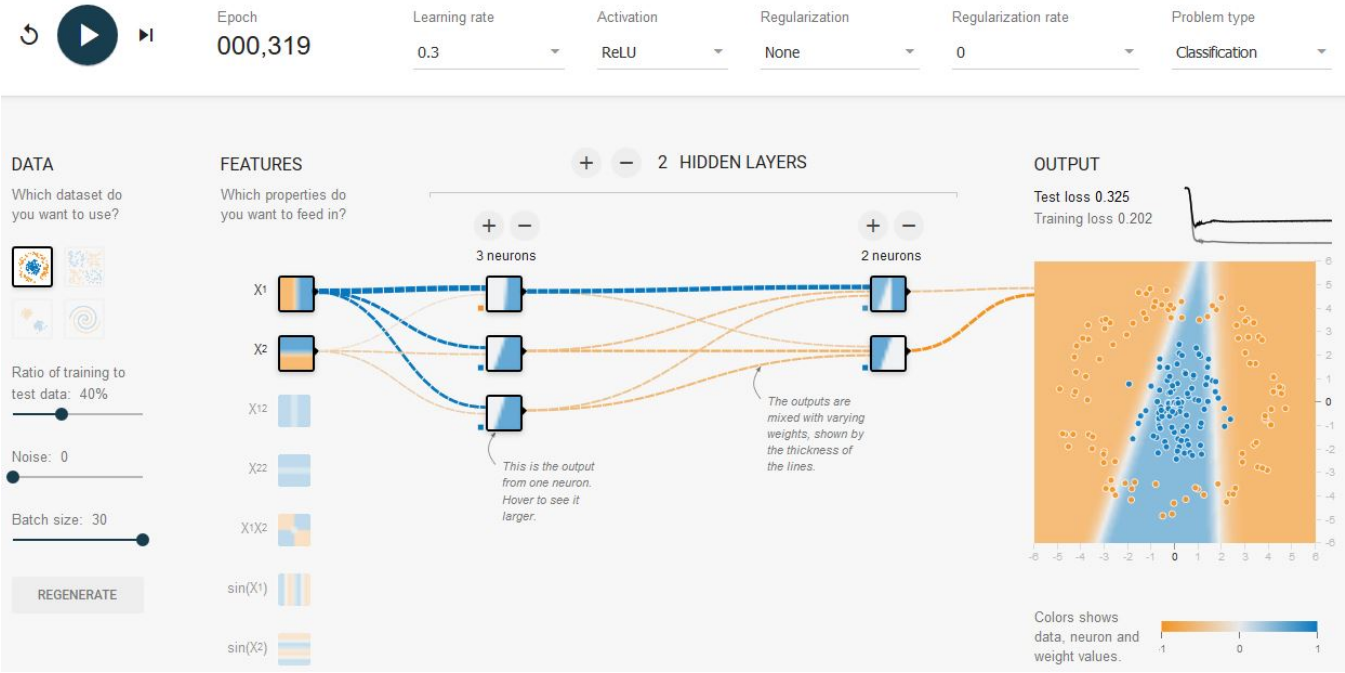

Figure 1: MLP training example.

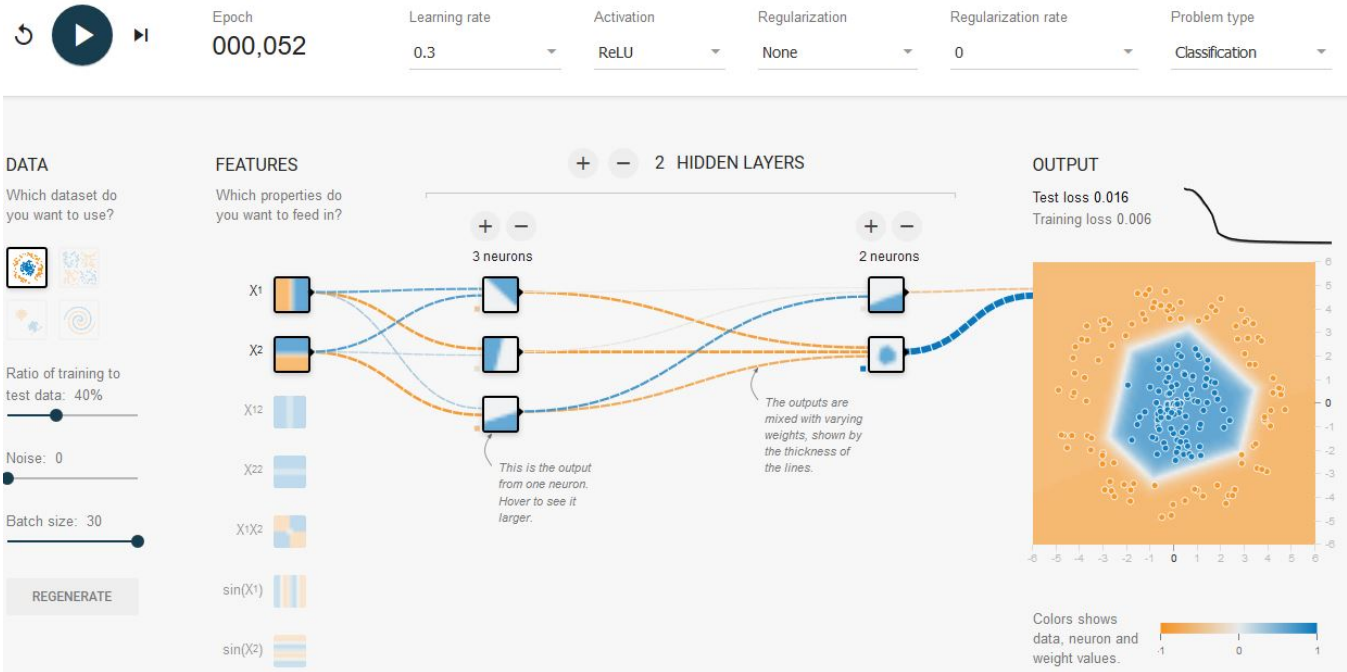

Figure 2: MLP training example.

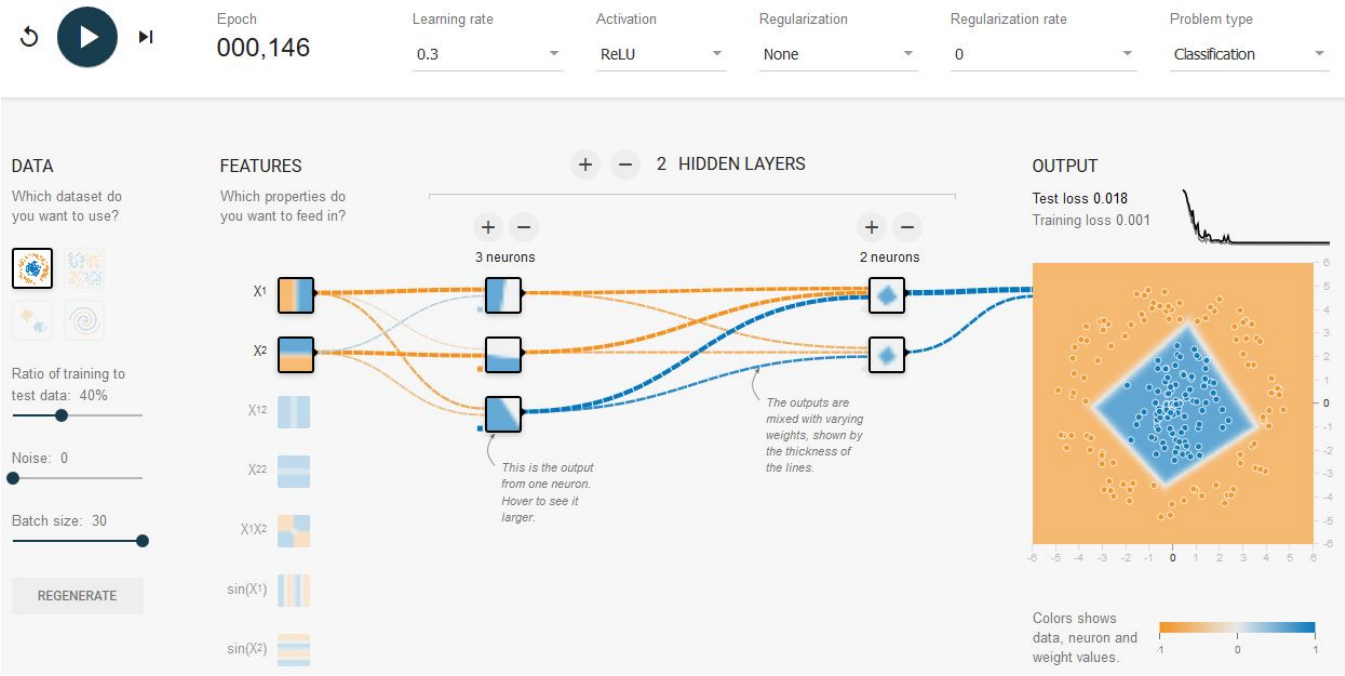

Figure 3: MLP training example.

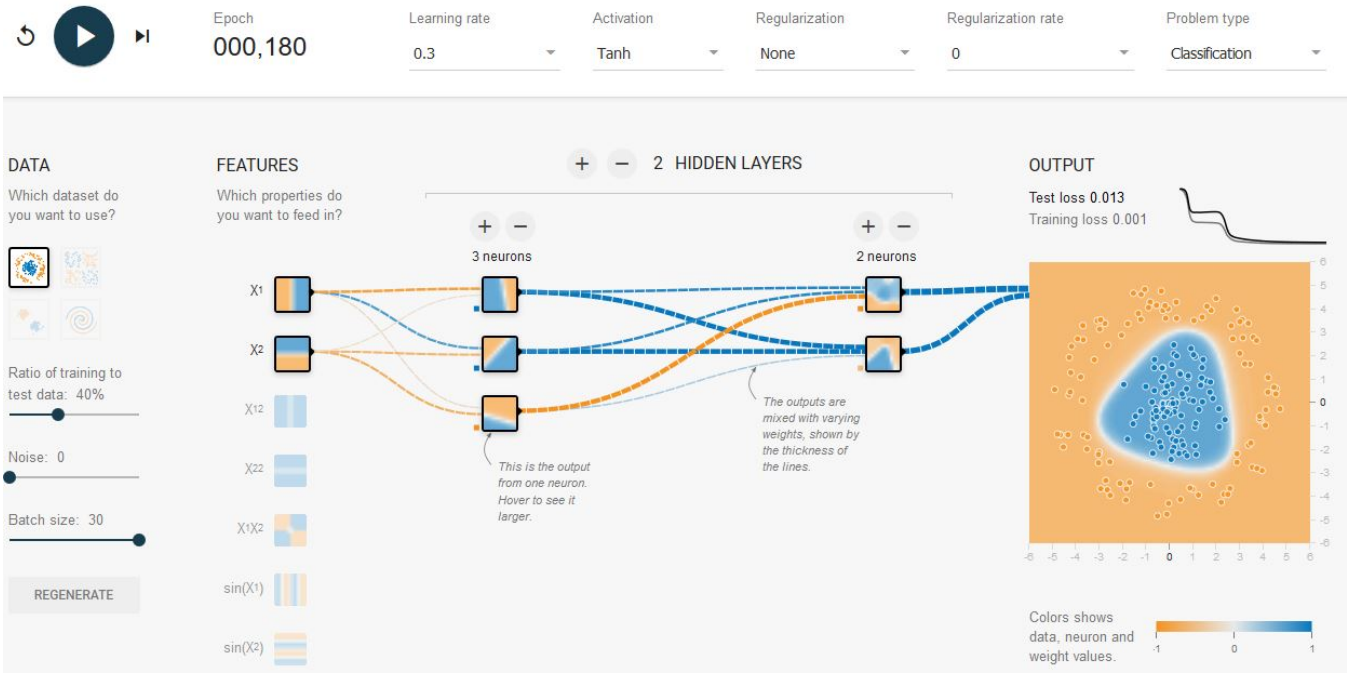

Figure 4: MLP training example.

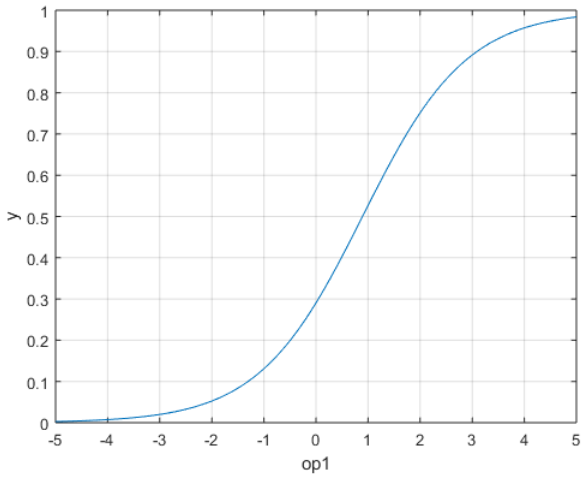

Figure 5: Softmax plot.

Extension Questions

4. The softmax activation function:

$$y_i = \frac{\exp o_i}{\sum_k \exp o_k}$$

is often used in MLPs with multiple output units. Consider an MLP with 3 output units. If outputs o_2 and o_3 (before activation), are fixed at 0.2, plot the value of softmax as a function of the other output value, o_1 at an appropriate scale.

```
%softmax script

op1 = -5:0.1:5;
op2 = 0.2;
op3 = 0.2;

y = zeros(1,size(op1,2));
for i=1:size(op1,2)
    y(i) = exp(op1(i))/sum(exp(op1(i))+exp(op2)+exp(op3));
end
figure
plot(op1,y);
grid;
xlabel('op1');
ylabel('y');
```