The Multi-layer Perceptron

1. What is the difference between a batch and a single stochastic update? What are the reasons to prefer one over the other?

   A batch update computes the gradient using a set of misclassified points, while an online update uses a single point. The batch update uses a better estimate of the gradient: indeed if the set of used points is the whole set of misclassfied points, the computed gradient is exact. On the other hand, stochastic gradient descent converges in expectation, and the single update may increase the error on the whole dataset (but it will decrease on average). A Batch update is therefore preferable, but more computationally expensive. On the other hand, the single stochastic update is faster to compute, and can be performed online as new data points arrive, if the training set grows over time.

2. Why do we need to substitute the step function with the sigmoid for the Multi-Layer Perceptron (MLP)?

   Because the step function is not differentiable in 0, and the gradient would be of no use anywhere else as well. The sigmoid function, on the other hand, is smooth.

3. What optimization algorithm do we use, both for the perceptron and the MLP?

   Gradient descent.

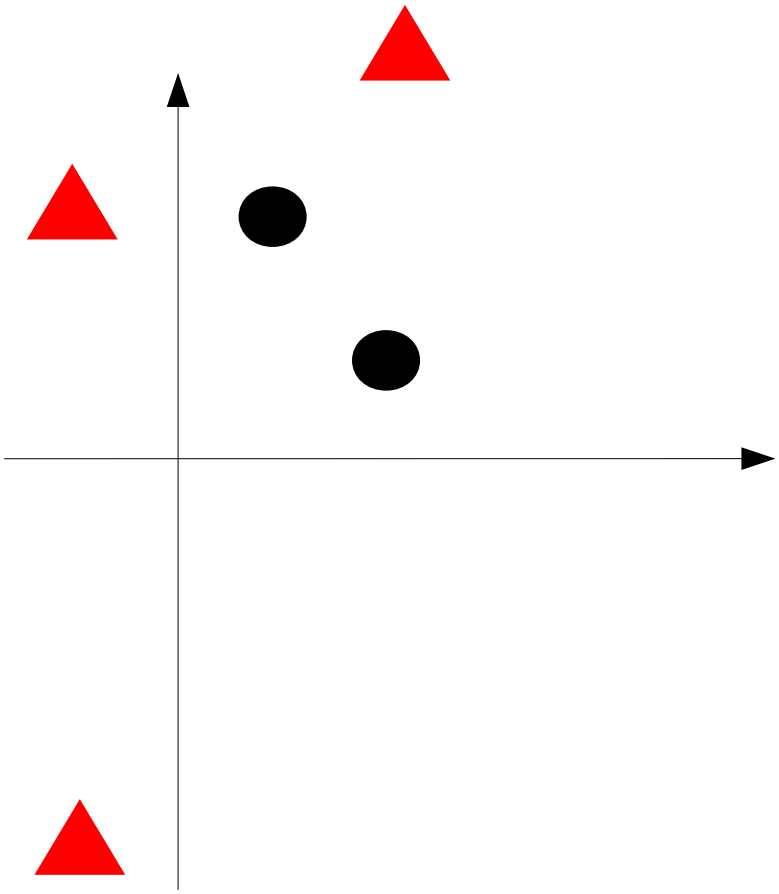
4. Given the dataset: <1,1,0>,<2,1,0>,<1,3,1>,<2,-1,1>, where the last element of each vector is the class, construct an MLP that separates the classes.

   This data set is not linearly separable, therefore we can't use a single neuron. There are infinitely many solutions, an easy one to compute is to use the two neurons with decision boundaries y=2 and y=0 , corresponding to the weight vectors: $\langle -2,0,1\rangle$ and $\langle 0,0,1\rangle$ respectively. The first vector ($\langle 0,1\rangle$) points upward , and is therefore OK. The second vector ($\langle 0,1\rangle$) also points upward, while in this case we want the neuron to return 1 below the line y=0 . Therefore, we it is best to invert the second vector of weights, which becomes $\langle 0,0,-1\rangle$. These two vectors constitute the first layer of our MLP.

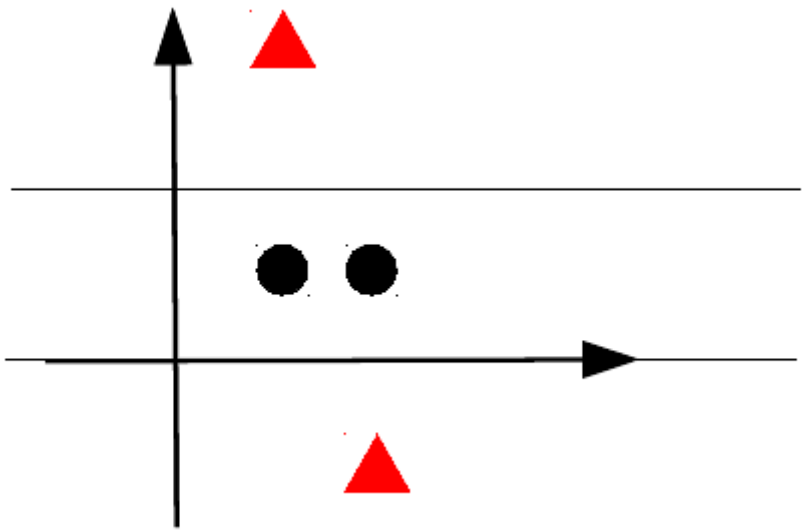
   The hidden layer has to compute the union of the two areas in which the first layer returns 1. This corresponds to a logical OR operator:

   Realised, for instance, by the discriminating boundary $y=-x+\frac{1}{2}$, which corresponds to the weight vector $\langle -\frac{1}{2},1,1\rangle$. The final, resulting, MLP is:

5. Given the dataset: <2,1,0>,<1,2,0>,<-1,2,1>,<-1,-3,1>,<2,3,1>, construct an MLP that separates the classes.



Again, we begin by choosing two possible discriminating lines, for instance x = 0, and y = 2.5, with vectors $\boldsymbol{w}_1 = \langle 0, 1, 0 \rangle$ and $\boldsymbol{w}_2 = \langle -2.5, 0, 1 \rangle$. The first vector returns 1 on the right-hand side of



the corresponding vertical line, so we need to invert it: $w_1 = \langle 0, -1, 0 \rangle$. These two vectors form the first layer. The second layer is also an OR, so is the same as the previous answer. The resulting MLP is: