Tutorial 2 Solutions:

1. Replace `a1 = sigmoid(z1)` in `bp.py` (line 63) with:

   ```
    a1 = np.maximum(0,z1) # if we want relu
   ```

   Replace `d1 = d_sigmoid(z1)*np.dot(d2, W2.T)` in `bp.py` (line 87) with:

   ```
   d1 = np.dot(d2, W2.T)
   d1[a1 <= 0] = 0
   ```

2. To the loss function in `bp.py` (line 69), add:

   ```
   0.5*reg*np.sum(W1*W1) + 0.5*reg*np.sum(W2*W2)
   ```

   To the derivatives of the gradient, add the following anywhere after you have computed the gradient with respect to the cross entropy part of the gradient:

   ```
   # if we add regularization, get its contribution to the gradient
   dW2 += reg * W2
   dW1 += reg * W1
   ```

5. Attach the following to the end of the code to produce data and write it to the file '`hidden_input.dat`':

   ```
   f = open('hidden_input.dat', 'w')
   for ii in range(Ntemp*samples_per_T):
     batch=(mnist.test.images[ii,:].reshape(1,lx*lx),
   mnist.test.labels[ii,:].reshape((1,numberlabels)))
     mag=np.sum((2*batch[0]-1 ))/(lx*lx)
     output=sess.run(hl,feed_dict={x:batch[0]})
     f.write(str(mag)+'  '+' '.join(map(str, output[0,:]))+"\n")

   f.close()
   ```

   Then, use this data to plot $W_1X + b_1$ vs. m(X).