APM120, Homework #10

Applied Linear Algebra and Big Data

Last updated: Tuesday 11th April, 2023, 14:48

Assigned April 11, due **Tuesday, April 18**, 1:00 pm, via gradescope, **in pdf $\leq$ 20Mb, $\leq$ 30 pages.**

Supervised learning: classification II, Neural networks

Show all steps in all calculations explicitly. Attach code used, well documented, and relevant plots and Matlab/python output, attaching code and figures *immediately following* the relevant question solution. A code printout is not a substitute for complete solutions, your solution should stand alone without the Matlab/python code or output. See needed python preliminaries at end of this HW[1]. It is fine to use Matlab/python unless a hand calculation (using only a simple calculator) is required *in orange*. For all questions, make sure you show all steps as if you are doing the problem by hand, and do not use library functions unless explicitly allowed in the question. Make sure you can do all calculations using no more than a hand-held calculator. See the end-note for guidelines and examples of hand-calculations.[2]

1. **Neural networks, back-propagation:** consider the following training data point,

   ```
   X=[-1; 2];   y=2;
   ```

   for a neural network with all-sigmoid nodes, and with the following initial weights,

   ```
   % Matlab:
   w2=[ -0.2,  -0.4;
        0.4,  -0.4;
        0.1,  0.1];
   b2=[ -0.5;
        0.2;
        0.1];
   w3=[ 0.1,  0.1,  0.1];
   b3=[ -0.1];
   ```

   ```
   # python:
   w2=[[-0.2, -0.4 ],
       [ 0.4, -0.4 ],
       [ 0.1,  0.1 ]]
   b2=[[-0.5 ],
       [ 0.2 ],
       [ 0.1 ]]
   w3=[ 0.1,  0.1,  0.1 ]
   b3=[-0.1 ]
   ```

   and assume a quadratic cost function $C = \frac{1}{2}(y - a^L)^2$. (**A**) Write the explicit equations for the network, and draw its detailed architecture. Use the feedforward network to calculate the output and the cost for the training data point and label; show the values of all intermediate neurons. (**B**) Write the explicit backpropagation algorithm for this network. (**C**) Use the algorithm to hand-calculate the gradient with respect to $w^2_{3,2}$ and b^2_3. (**D**) Calculate the gradient with respect to all weights w^l and biases b^l. (**E**) Calculate the gradient with respect to w^2_{12} using a direct perturbation approach with $\epsilon = 0.002$ and compare it to the appropriate gradient element calculated using backpropagation. (**F**) Use a learning rate of $\eta = 1$ to adjust the weights and perform a steepest-descent iteration for the data point provided. Calculate the cost for the revised weights. Is it lower? Discuss.

2. **Neural networks using Matlab/python routines:** consider the following training data for a 2d classification/ regression problem,

   ```
   X=[3,6,1,0,5,5,4,4,3,4,6,3,2,5,4,8,6,8, 9,8,3,6, 1,4,7,2,9,1,4,2,9,1,5;
      9,1,5,7,5,5,2,2,2,0,2,4,1,2,9,8,8,10,1,3,4,6,10,7,2,7,4,9,4,6,8,9,4];
   y=[2,3,1,1,3,3,3,3,1,3,3,1,1,3,2,2,2, 2,3,3,1,2, 1,2,3,1,2,1,1,1,2,1,3];
   ```

(a) Use the built-in Matlab Neural Network toolbox or python's Keras package plus Google's TensorFlow backend[3]) to construct a feed-forward *classification* neural network and calculate the labels on the finer grid specified by `x1=[0:0.1:10]; x2=[0:0.1:10];` placing the labels in a 2D array Z. Contour the results using Matlab: contourf(x1,x2,Z), python: plt.contourf(x1,x2,Z).

(b) Use the same training data to construct a feedforward *regression* neural network and calculate and contour the regression values calculated for the above `x1,x2` grid.

Note: In both cases, you need to choose appropriate architectures that are able to classify the given data. *Hint:* you may start with the following two examples on the course web page:
neural_networks3_simple_2d_classification_example.m/py
neural_networks4_simple_2d_regression_example.m/py
Try different architectures (number of hidden layers, number of neurons) until you find one that reasonably classifies the training data and the grid of points. This requires some experimentation, although the programming effort is minimal given the above two codes. Discuss your results...

3. **Overfitting in a neural network:** use the network based on the weights and biases given in the HW data file:

```
% Matlab
load saved-network-HW-10.mat
```

```
# python
from scipy import io
mat = scipy.io.loadmat('saved-network-HW-10.mat');
w2=mat['w2']; b2=mat['b2']; w3=mat['w3'];
b3=mat['b3']; w4=mat['w4']; b4=mat['b4'];
```

Assume "tansig" transfer functions for all layers but the last and a linear output transfer function. First, calculate and create a contour plot for the classification for (x, y) in the range 0:0.1:1. These values are equal to the training data. Then use the same network to classify in the range given by 0:0.02:1. Discuss the difference between the two results, what it indicates, why it occurred and how the problem may be overcome.

* [What's the point of ***optional extra credit challenge problems: apart from the fun of doing them, they may bring the total score of this HW assignment up to 110%, making up for problems you may have missed in this or other HW assignments...]

Python preliminaries & notes

1 python commands within the HW assume you have first used the followings: `import numpy as np; from numpy import linalg; import scipy as scipy; from scipy import linalg; import matplotlib.pyplot as plt; import matplotlib;`
Input a matrix A, column vector **b** and row vector **c** into python in the form
`A=np.array([[a11,a12,a13],[a21,a22,a23],[a31,a32,a33]]; b=np.array([[b1],[b2],[b3]]); c=np.array([[c1,c2,c3]]);` or convert Matlab arrays given in HW directly to python arrays using, e.g., `A=np.array(np.matrixlib.defmatrix.matrix('[1 2 3; 4 5 6]'));`

2 **Hand calculations**, in which you are asked to use only a simple hand-held calculator, are required only when we want to make sure you understand exactly what each step of an algorithm is doing. These also prepare you for the quizzes that involve similar hand calculations. **How much work to show?** Just don't use scratch paper, show all steps that you actually use for the hand calculations, but no more, carrying out calculations to **three significant digits**. Trust the graders to be reasonable, they were students in the course last year. **Examples: (i)** If you are asked, *not* in orange, to calculate the LU decomposition of a matrix, you may use Matlab/python as in `A(2,:)=A(2,:)-A(1,:)*(A(2,1)/A(1,1))` etc, but you may not use a library routine as in `[L,U,P]=lu(A)` except for checking your results. **(ii)** If required to calculate by hand the element $C_{2,3}$ of a matrix product $\mathsf{C} = \mathsf{AB}$, you need to explicitly multiply using a hand calculator the second row of A with the third column of B. You may not use Matlab/python to calculate `C=A*B` and then take the needed element from that product, nor to calculate `C23=A(2,:)*B(:,3)`.

3 See installation instructions