10/01/2022

Foundations of Artificial Intelligence

Problem set 3

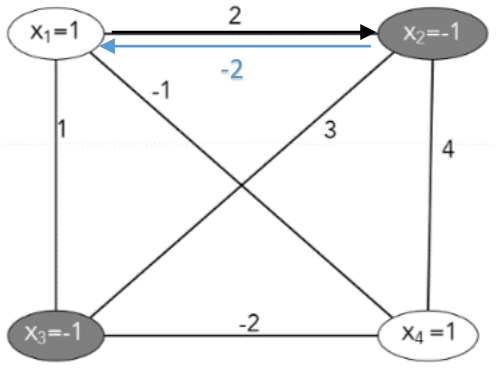
Presenter ID: 314975442

*** I used the libraries: pandas, numpy, random, pprint.

1. About the code:
   Due to poor time organization of mine, I created all the decision tree but the implementation of chi-square test. I understand the consequences of create a decision tree without pruning, means overfitting which will be caused a great error for the test results.
   I assume that the great error might cause also from one split I've made for continues ranges, I mean to divide the data to below or above the value that raised from the entropy.
   Except from the first function of the build tree, I created the next two functions as needed, consider that the tree is a valid input.
2. A) Defining symmetric matrix and how it resembles to the neural net:
   In linear algebra, a symmetric matrix is a square matrix that is equal to its transpose.
   The entries of a symmetric matrix are symmetric with respect to the main diagonal.
   i.e. a symmetric matrix resembles to the net which the weight between two neurons for both sides are the same: $w_{ij} = w_{ji}$
   So, by make a non-symmetric matrix means the weight from neuron i to neuron j would be different.
   For the example I'll use a Skew-symmetric matrix which $A^T = -A$
   And use the example from class($x_{1..n}$ are sigh functions) :

$$x_1 = 1$$
$$x_2 = -1$$
$$x_3 = -1$$
$$x_4 = 1$$

$$W = \begin{pmatrix} 0 & 2 & 1 & -1 \\ -2 & 0 & 3 & 4 \\ -1 & -3 & 0 & -2 \\ 1 & -4 & 2 & 0 \end{pmatrix}$$



I'll call row i in matrix W as W_i:

First iterations:

$$w_1 * \begin{pmatrix} \mathbf{1} \\ -1 \\ -1 \\ 1 \end{pmatrix} = -2 - 1 - 1 = -4$$

$$w_2 * \begin{pmatrix} -1 \\ \mathbf{-1} \\ -1 \\ 1 \end{pmatrix} = 2 - 3 + 4 = 3$$

$$w_3 * \begin{pmatrix} -1 \\ 1 \\ \mathbf{-1} \\ 1 \end{pmatrix} = 1 + 3 - 2 = 2$$

$$w_4 * \begin{pmatrix} -1 \\ 1 \\ 1 \\ \mathbf{1} \end{pmatrix} = -1 - 4 + 2 = -3$$

Second iterations:

$$w_1 * \begin{pmatrix} \mathbf{-1} \\ 1 \\ 1 \\ -1 \end{pmatrix} = 2 + 1 + 1 = 4$$

$$w_2 * \begin{pmatrix} 1 \\ \mathbf{1} \\ 1 \\ -1 \end{pmatrix} = -2 + 3 - 4 = -3$$

$$w_3 * \begin{pmatrix} 1 \\ -1 \\ 1 \\ -1 \end{pmatrix} = -1 + 3 + 2 = 4$$

$$w_4 * \begin{pmatrix} 1 \\ -1 \\ 1 \\ \mathbf{-1} \end{pmatrix} = 1 + 4 + 2 = 7$$

Third iterations:

$$w_1 * \begin{pmatrix} \mathbf{1} \\ -1 \\ 1 \\ 1 \end{pmatrix} = -2 + 1 - 1 = -2$$

$$w_2 * \begin{pmatrix} -1 \\ 1 \\ 1 \\ 1 \end{pmatrix} = 2 + 3 + 4 = 9$$

$$w_3 * \begin{pmatrix} -1 \\ 1 \\ \mathbf{1} \\ 1 \end{pmatrix} = 1 - 3 - 2 = -4$$

$$w_4 * \begin{pmatrix} -1 \\ 1 \\ -1 \\ \mathbf{1} \end{pmatrix} = -1 - 4 - 2 = -7$$

Fourth iterations:

$$w_1 * \begin{pmatrix} \mathbf{-1} \\ 1 \\ -1 \\ -1 \end{pmatrix} = 2 - 1 + 1 = 2$$

$$w_2 * \begin{pmatrix} 1 \\ \mathbf{1} \\ -1 \\ -1 \end{pmatrix} = -2 - 3 - 4 = -9$$

$$w_3 * \begin{pmatrix} 1 \\ -1 \\ \mathbf{-1} \\ -1 \end{pmatrix} = -1 + 3 + 2 = 4$$

$$w_4 * \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = 1 + 4 + 2 = 7$$

Right after the fourth iteration I will start over in the third iteration and so on, means that example doesn't converge.

B) By using linear algebra, matrix multiplication will serve as an updates simultaneously of the neurons. I'll use again the example from class and demonstrate that the same situation of two positions that never enable to converge:

First iteration:

$$\begin{pmatrix} 0 & 2 & 1 & -1 \\ 2 & 0 & 3 & 4 \\ 1 & 3 & 0 & -2 \\ -1 & 4 & -2 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \\ -1 \\ 1 \end{pmatrix} = \begin{pmatrix} -4 \\ 3 \\ -4 \\ -3 \end{pmatrix}$$

Second iteration:

$$\begin{pmatrix} 0 & 2 & 1 & -1 \\ 2 & 0 & 3 & 4 \\ 1 & 3 & 0 & -2 \\ -1 & 4 & -2 & 0 \end{pmatrix} \cdot \begin{pmatrix} -1 \\ 1 \\ -1 \\ -1 \end{pmatrix} = \begin{pmatrix} 2 \\ -9 \\ 4 \\ 7 \end{pmatrix}$$

Third iteration:

$$\begin{pmatrix} 0 & 2 & 1 & -1 \\ 2 & 0 & 3 & 4 \\ 1 & 3 & 0 & -2 \\ -1 & 4 & -2 & 0 \end{pmatrix} \cdot \begin{pmatrix} 1 \\ -1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -2 \\ 9 \\ -4 \\ -7 \end{pmatrix}$$

3. The idea behind boosting algorithm is, create with a group of "weak learners" – some learning algorithm, that strives to improve the learners by focusing on area where the system is not performing well.
   The process of create a new learner is by test the learner with the data he was built from and discover the areas it couldn't cover.
   When building the next learner, the data will come again from the training, but each instance is weighted according to the error of the last learner, and so on…
   As we know Decision trees use the information gain that calculated by Entropy or Gini to give weight to the next questions, that's how we could improve our algorithm by force the algorithm with higher weight to ask the questions that are more likely will figure the path of finding the right answer next time.
   Neural nets using gradient to give weight to the edges from the output layer recursively threw the hidden layers till the input layer by process called back propagation.
   I reckon that it is not possible to use both neural nets and decision trees together in a boosting algorithm because we are not entirely sure how does the hidden layers take their decisions and what are the path that the information go throw.
   That's why I assume decision trees will be chosen to use by boost algorithms, and Neural nets Would train each other in other ways as Generative Adversarial Network.