Theoretical Task

Arinas Bertlans
EKSfum-16

1. a.) After third time rethinking the task, this is ~~how~~ I understood it: system gathers data for each ONE specific product and makes a decision of how many of such items should be kept in stockpile. Though what is stopping system from saying: "I want 50 of everything (1-50 € range)". I also consider that there are a number of user controlled settings: price range control (as mentioned), stockpiling priority (keep more, less, average amount in stockpile, is it Christmas-ish type of product (expect more sales in December), flag to omit ~~from~~ system, ~~some~~ some thresholds, etc.

b.) Step one: get sales history, get service history, get current inventory, get ~~sea~~ search history, get view history, get user settings. Step two: make a decision based on that data.

c.) For each item system:

I. Gets sales history - how many where sold in previous time frame.

II. Gets search history - how many times it appeared in search ~~wd.~~ results in this time frame.

III. ~~Ed~~ Gets view history - how many times viewed in...etc.

IV. Gets service history - how many items were returned or serviced

V. Provides UI so that some settings could be changed.

VI. Determine the ~~a~~ amount to stockpile.

VII. Take some ~~at~~ action

2. Now focus on ~~part~~ part VI as that where neural networks are.

a). System has a bunch of numbers coming (effectively ~~f~~ features) so there are few ways to use them

I. ~~basi~~ Basic approach: make custom formula.

II. MLP with as many neurons on output as needed (50, 20, 6), by which I mean neuron that gives 1 indicates the amount to stockpile

III. MLP with non-linear activation function.

IV. NN plus fuzzy logic.

V. SOM

b.) I. Have all features so just make a custom formula that utilizes them to get a number. In the end though it would end up as NN formula $y = \Sigma(x\,w) + b$. Too many variables ~~to~~ be effective...

[illegible] Andrius [illegible] 16

II. From what I've experinced ~~in~~ MATLAB before getting single class out of 90 to be 1 is hard, but in ~~theor~~ theory could work providing a vector in the end where index ~~to~~ of 1 is desired stockpile ~~amount~~.

III Only diference from II is output layer neuron ~~cou~~ count and activation function. This outputs a normalized number ~~dir~~directly.

IV ~~to~~ Looking for some papers on the subject saw this variant, but as I imagine it working it assigns a class and fuzzy logic states certainty of it belonging to some class.

V. SOM apparently can be used anywhere, just give some numbers and it'll give you some answers, sometimes correct.

3. Ladies and gentlemen, engineers and time wasters; I'm here to offer you the ~~ul~~ ultimate solution to your goods stockpiling problem! Don't know how ~~many~~ many items to stockup for next time period? Well then let this multi-layer perceptron based artificial neural network do the the thinking for you so that you can use your neural network for more pressing matters. (Some configuration and ~~ov~~ oversight will be needed).

a.) Well I wasted a few hours wrestling with nntools in MATLAB to ~~g~~design feed-forward backdrop MLP, thus concluding that MLP with singe output and sigmoid activation function is good enough for the task.

b.) Honestly, first option, costum function, wasn't an option - too many variables. Fourth option is mostly guesswork ~~to~~ how it works in the first place ~~so~~ and seem to give a fuzzy answer when we need a precise-ish number. Second option was a contender for a while but speed is not an issue in this case so migh as well go precision. Also there's SOM.

So non-linear activation function MLP is precise, relatively ~~simple~~, not much maintenance and few training ~~cy~~ cycles.

4. ~~As~~ As I imagine system would start working correctly after few time periods, but would be sped up by copying data for inputs VI and VII from some other already trained similar product. Decision making part, not gathering or taking action parts, ~~are~~ is as follows per item:

a) Inputs:

I. ~~Latest~~ This time period sales - how many sold.

II. This time period search appearances - how many times appeared

Arūnas [illegible]
Eksfmu-16

In search results.
III This time period view count ~~- acto~~ actual description view count.
IV This time period service data - where any returned or serviced.
V Leftover stockpile - how many weren't sold.
VI. Previous time periods data arrays - stored data of above inputs plus how many items ideally should be stockpiled for those periods.
VII NN parameters - weights and biases.
VIII Applicable user settings - settings that ~~inp~~ influence results.
Outputs:
I. Number of item there ~~should be~~ should be stockpiled.
b.) I. if inputs VI and VII are available, train/adjust NN parameters with the old ~~da~~ data
II Simulate new result with updated NN using inputs I through V.
c.) Basic MLP formulas are applicable. I tested two-layer MLP. Hidden layer had ~~some~~ same amount ~~of~~ neurons as there were inputs. Output layer - one neuron. First layer output attained using:

$$y_m^{(1)} = \varphi\left(x_1 w_{m1}^{(1)} + \dots + x_n w_{mn}^{(1)} + b_m^{(1)}\right) = \varphi\left(v_m^{(1)}\right),$$

* where x is input value, n is input number, m is hidden layer perceptron number, w are weights, b ~~are~~ is ~~biases~~ bias, φ is the activation function. I tested with

$$y = \tanh(\cancel{w} v_m),$$

but since negative numbers are are not needed and to avoid shifting it all up or some other workaround/fix/improvement

$$y = \frac{1}{1 - e^{-v_m}},$$

should be used. Then second layer output is calculated using:

$$y_2^{(2)} = \varphi\left(\sum y_m^{(1)} w_{2m}^{(2)} + b_2^{(2)}\right),$$

where 2 is second/output layer ~~per~~ neuron number, that can be omitted here, but this is a general formula.

[illegible]
[illegible] 16

NN ~~par~~ parameter updates go in reverse, thus back propagation. New last layer weights are found using:

$$\Delta w_{2m}^{(2)} = \eta y_2^{(2)} (1 - y_2^{(2)}) e_2 y_m^{(1)} = \eta \delta_2^{(2)} y_m^{(1)},$$

where η is rate of change and e is error obtained by checking difference between expected and received outputs - how many items should've stockpiled and how many system said to stockpile. Bias is attained using same formula just omitting $y_m^{(1)}$. Hidden layer weights update using:

$$\Delta w_{mn}^{(1)} = \eta y_m^{(1)} (1 - y_m^{(1)}) (\delta_2^{(2)} w_{2m}^{(2)}) x_n$$

And to get bias omitt x_n.
~~Final decision making~~ ~~At~~ At this point ~~yout~~ you'd have normalized number, but after after multplying back up to normal ~~in~~ integers and subtracting surplus, you'd get how many item to add to stockpile

Notes:
1. First day was depressed thinking I don't understand it at all. Second day 8am sat down and started decyphering. Two hours later think I understood the task, but also hit "whatever" stage so it is not the most sciency type of work and full of "jokes", but after 8 hours of work I don't think it's that bad.
2. Lately I increasingly notice having memory issues - honestly don't remember 90-98% of the stuff from this semester; so in places might seem like I missuse terms, but idea is solid I just might ~~t~~ call it a different thing.
3. As for search in internet: most links were to ScienceDirect and such and I'm not paying for those articles.
4. Arm is barely functioning now after copying it all, so reading it all might be an issue, this .docx version is in: github.com/EpicFail42/open-source-repo/~~tree~~/master/~~[illegible]~~ /NeuralNetworks/15.E.docx