CS585 Final

Fall term, 12/12/18
Duration: 1 hour

Instructions/notes

- the exam is closed books/notes/devices/neighbors, and open mind :)
- there are 8 questions, and a 'non-data-related' bonus
- there are no 'trick' questions, or ones with long calculations or formulae
- **please do NOT cheat; you get a 0 if you are found to have cheated**
- **when time is up, stop your work**; you get a 0 if you continue

Q	Your score	Max possible score
1		5
2		5
3		4
4		5
5		4
6		5
7		4
8		3
Bonus		1
Total		**36**

Q1 (3+2=5 points).

a. What is the most straightforward way to transfer (ie. use in a different app, or server or device etc) the results of training a neural network on a large set of training data?

A: Use of a weights-only file, or a config file with architecture+weights - eg. the .m5 weights file is what we used in the ML homework, to transfer the training results to the classification part.

'Transfer learning' is not the right answer - the question mentions 'a different app...', not a different learning domain.

b. Name two practical applications where you might do such transferring (train, use elsewhere).

A: Self-driving car [where the weights are transfered to hardware], a smartphone app to identify birds/mushrooms/flowers/clouds....

Q2 (2+3=5 points). Machine Learning, ie. "ML", has enjoyed runaway success within the last decade, eg. in the form of Alexa, self-driving cars, etc. This is on account of the availability of big datasets, large computing power, adequate memory, and good algorithms/APIs. The modern version of ML is DL, ie. "Deep Learning".

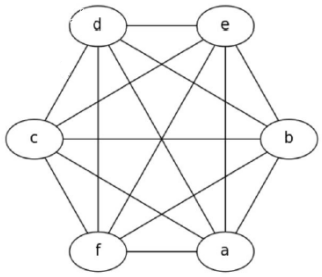
a. What makes DL "deep"?

A: The number of intermediate/'hidden' layers.

b. Even with DL, there is a serious, fundamental, show-stopper flaw in the entire approach to AI. What is it? In other words, what is ML's/DL's limitation, one that cannot be solved by faster processing, more memory, more training data, etc?

A: The limitation is that there is no genuine UNDERSTANDING of what the ML/DL is able to learn/classify! For example, the HW5 NN could tell apart cats and dogs, but it does not know that cats and dogs are the most common type of pets [doesn't even know what a pet is, etc], and, has no way of being 'told'. Also, ML/DL has no way to tell apart, correlation in data, from causation (where a part of the data ('output columns'), RESULT from factors that the input columns describe).

Q3 (4 points). Consider the following graph:



As you know, graph data can be represented via JSON, to make it be universally readable via a simple parser. Following are two representations; what is a third? You need to show your representation clearly, using valid and complete JSON like below.

```
{
  "graphData": {
    "vertices": ["a","b","c","d","e","f"],
    "edges":[
       [a,b], [b,c], [c,d], [d,e], [e,f], [a,f], [a,c], [a,d], [a,e], [b,d], [b,e], [b,f], [c,e],
       [c,f], [d,f]
     ]
  }
}
```

```
{
  "graphData": {
    "edges":[
       [a,b], [b,c], [c,d], [d,e], [e,f], [a,f], [a,c], [a,d], [a,e], [b,d], [b,e], [b,f], [c,e],
       [c,f], [d,f]
     ]
  }
}
```

A:

```
{
 "graphData": {
   "neighbors":[{"a":["f","c","d","e","b"]}, {..}, {..}, {..}, {..}, {..}]
 }
}
```

A variation of the above, also acceptable, would be the elimination of the 'neighbors' key, and simply make the value of 'graphData' be an array of objects like the one shown above.

Another more creative variation (which only works for a fully connected graph!) would be to list each loop, ie. make the value of 'graphData' be [["a","b","f"],["a","b","e"]...]. Note that there are 3-element loops, 4-element and 5-element ones, and a 6-element one.

Q4 (1+4=5 points). MapReduce is a great architecture, for executing mappers in parallel, then aggregating their outputs via a reducer step; cascading these provides enough flexibility to handle a variety of data-processing tasks.

There is another architecture [not YARN], a "MapReduce++", if you will, which extends the MapReduce paradigm.

a. What is it called?

A: Flink.

b. What are a couple of enhancements that it provides (just name them)?

A: additional transformations (beyond map(), reduce()) such as Join, Filter; additional datatypes (based on Java and Scala).

OK if the answer lists Join, Filter etc. as the 'couple' of enhancements.

Q5 (4 points). Geo-spatial data is inherently 2D, being composed of (lat,long) [or (long,lat)] pairs. What is the fundamental difference in how we set up the DB engine to query such spatial data, compared to standard querying (of non-spatial data)? Illustrate with a diagram.

A: the use of two-level processing - at the first level (filter step), MBRs are used to discard entities outside the query region; at the second stage (refine step), candidates from the first stage are queried exactly, to output the final results.

Q6 (5 points). As you know, there is a variety of algorithms used for data mining. If our data needs to be binary-classified (A or B, yes or no, low or high...), what are our choices, in other words, what algorithms will help us do this? Name/discuss briefly, 5 of them.

A [just names are here - see notes for descriptions]:

a. decision tree
b. clustering
c. regression
d. neural network
e. SVM
f. sigmoid (logistic regression)
...

Q7 (4 points). Augmented Reality (AR) is where we superpose computer graphical (CG) rendering over live (video) imagery, and modify the graphics to sync with changes in viewpoint (camera motion) - this makes it possible to 'pin' the CG renders on to arbitrary real-world surfaces.

How would you use AR, for data visualization and interaction? Be imaginative - this is an open-ended question.

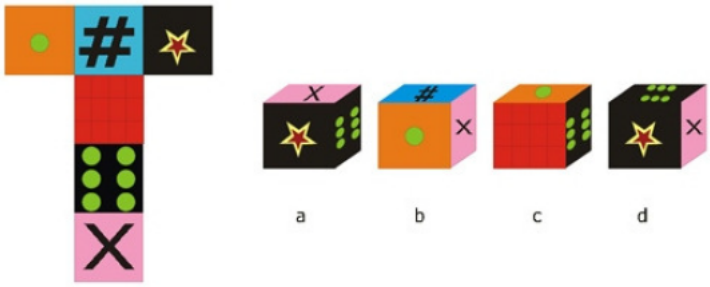
A: a flat surface on a wall, eg. a blank wall, or a blank whiteboard on it, or a poster... can be used to display 2D visualizations; or, a tabletop or coffeetable etc. can be used to display 3D viz, eg. a multi-linear (two inputs) regression plane, 3D stacked bar charts, SVM plane, 3D clusters...

Q8 (3 points). The use of JSON for representing semi-structured data provides us flexibility, compared to relational tables, when it comes to handling missing data (eg. a customer in a bank does not provide an email address while signing up for an account by walking into a bank, because "the Government will track me because of it"). What are some options for handling missing data in a JSON representation [eg. the value for an 'Email' key] of the customer's account? Name/list 3 valid ways [be imaginative] - they all don't need to be equally practical/efficient.

A:
a. just leave the missing key out!
b. "email":""
c. "email":"null"

The first is the best option.

Bonus (1 point). Look at the flattened cube below on the left. Which of the four shown cubes would produce the flattening?



A: ‘a’ [look at the photo below, that’s one way to solve - create a paper cube by folding the flattened ‘T’ pattern:)]