CSYE 7245–
Big-Data Systems and Intelligence Analytics
Sample Exam and Solutions One

Q1 (10 Points)

You are organizing a game hack-a-thon and want to make sure there is at least one instructor who is skilled at each of the n skills required to build a game (e.g. programming, art, animation, modeling, artificial intelligence, analytics, etc.) You have received job applications from m potential instructors. For each of n skills, there is some subset of potential instructors qualified to teach it. The question is: For a given number $k \leq m$, is is possible to hire at most k instructors that can teach all of the k skills. We'll call this the Cheapest Teacher Set.

Is the Cheapest Teacher Set in NP

Solution:

The problem is in NP. We can easily find a poly-time certifier. That is, IF given a solution for an NP problem you could verify it yes or no in polynomial time.
Poly-time certifier
Given a set of k instructors we can easily check if they cover the n skills just by putting the n skills in a list and then looping thru the k instructors and removing any skills they have from the list. If the list is empty then the k instructors have covered the list and we return yes. If we loop thru all k instructors and any skills remain in the list then we return no.

Q2 (5 Points)

1. How many people must there be before the probability that at least two people have a birthday on October 31 is greater than 1/2?

Solution:

Note: This is NOT the birthday problem or the probability that, in a set of n randomly chosen people, some pair of them will have the same birthday and a 50% probability with 23 people.

http://en.wikipedia.org/wiki/Birthday_problem

In that problem we are comparing every person to all of the others. We think of that problem in terms of number of possible pair-wise combinations.

In the question asked we are comparing n people with a fixed date, October 31, not making all possible pair-wise combinations.

- Probability that someone in the room has a birthday on October 31, denoted by P(B) is 1− probability that no one in the room has a birthday on October 31, $P(B) = 1 - (364/365)^n$.
- We wish $P(B) \geq 1/2$,

thus $1 - (364/365)^n \geq 1/2$. (*This is fine for full credit*)
Taking logs,
$\log(1/2) \geq \log(364/365)^n$
$-\log 2 \geq n \log(364/365)$
$\log 2 \leq n \log(364/365)$
$253 \leq n$

Q3 (5 Points) Consider a six-sided die that gets a 1 with probability p = 1/6. What is the probability that you can get a 1 after rolling the die 3 times?

Solution:

Because these are independent events (the roll of one die doesn't affect another) we have p = 1/6 chance on each of the die throws. You want probability of at least a 1 in 3 rolls that means total probability (1 - none of the dice has 1). Hence the probability comes out to be $1-(5/6)^3=1-0.58=0.42$

However if we want exactly one success (a roll of 1) in three tries this is just the binomial expansion. If we run n trials, where the probability of success for each single trial is p, what is the probability of exactly k successes?

$$\frac{1-p}{1} \quad \frac{p}{2} \quad \frac{p}{3} \quad \frac{1-p}{4} \quad \frac{p}{5} \quad \ldots\ldots \quad \frac{}{n}$$

k slots where prob. success is p, n-k slots where prob. failure is $1-p$
Thus, the probability of obtaining a specific configuration as denoted above is $p^k(1-p)^{n-k}$. From here, we must ask ourselves, how many configurations lead to exactly k successes. The answer to this question is simply, "the number of ways to choose k slots out of the n slots above. This is $\binom{n}{k}$. Thus, we must add $p^k(1-p)^{n-k}$ with itself exactly $\binom{n}{k}$ times.
This leads to the formula:

$$\binom{n}{k} p^k (1-p)^{n-k}$$

X = 1, with probability $\binom{3}{1}(\frac{1}{6})^1(\frac{5}{6})^2 = \frac{3}{8}$
Note: 3 choose 1 is 3. The 3/8 comes from $3^* 1/2^3$

3* = 0.3442 (3 choose 1)(1/6)(5/6)^2

Q4 (5 Points) Is $n^2 \log(n)$ big-O of n^2? That is, which function bounds the other from above as n gets large (if either does). Prove your answer. The intersection of the two functions is $x = e \approx 2.7183$

Solution:

Yes. A function f(n) is O(g) iff there exists a C and N such that $f(x) \leq C\,|g(x)|$ for all $x \geq N$. In the case of proving big-O, you need to find the C and N. We can see after an N=3 and a C=1 the x of $n^2 \log(n) \geq n^2$ as the x goes to infinity.

Q5 (10 Points)

Master Theorem
For each of the following recurrences, give an expression for the runtime T(n) if the recurrence can be solved with the Master Theorem. Otherwise, indicate that the Master Theorem does not apply.

i. $T(n) = 4T(n/2) - n^2$

ii. $T(n) = 2T(n/2) + c$

iii. $T(n) = nT(n/2) + n \log n$

iv. $T(n) = 27T(n/3) + n$

v. $T(n) = 2T(n/2) + O(n)$

Solution:

The Master Theorem applies to recurrences of the following form: $T(n) = aT(n/b) + f(n)$
where $a \geq 1$ and $b > 1$ are constants and $f(n)$ is an asymptotically positive function. There are 3 cases:
1. If $f(n) = O(n^{\log_b a - \varepsilon})$ for some constant $\varepsilon > 0$, then $T(n) = \Theta(n^{\log_b a})$. (CASE 1)
2. If $f(n) = \Theta(n \log_b a \log^k n)$ with $k \geq 0$, then $T(n) = \Theta(n \log_b a \log^{k+1} n)$. (CASE 2)
3. If $f(n) = \Omega(n \log_b a + \varepsilon)$ with $\varepsilon > 0$, and $f(n)$ satisfies the regularity condition, then $T(n) = \Theta(f(n))$.
Regularity condition: $af(n/b) \leq cf(n)$ for some constant $c < 1$ and all sufficiently large n. (CASE 3)

i. $T(n) = 4T(n/2) - n^2$

Does not apply. $f(n) = -n^2$

ii. $T(n) = 2T(n/2) + c$

$a=b=2$; $k=\log(2) \log(2)=1$ $f(n) = c$ $1>0$ so (CASE 1)
$(n) = \Theta(n^{\log_b a}) = \Theta(n^1) = \Theta(n)$

iii. $T(n) = nT(n/2) + n \log n$

Does not apply. a is not constant.

iv. $T(n) = 27T(n/3) + n$

$a=27$ $b=3$; $k=\log(27) \log(3)=3$ $f(n) = n$ $3>1$ so (CASE 1)
$(n) = \Theta(n^{\log_b a}) = \Theta(n^3)$

v. $T(n) = 2T(n/2) + O(n)$

a=b=2; k=log(2) log(2)=1 f(n) = n 1=1 so (CASE 2)
$f(n) = \Theta(n) = \Theta(n^k \log^p n) = \Theta(n^1 \log^0 n)$ so p=1

$\Theta(n^1 \log^{0+1} n)$ which is $\Theta(n \log n)$

Q6 (10 Points) Write the recurrence relations for merge-sort and quicksort. Make sure you include worst case and average case recurrence relations.

Solution:

The recurrence for merge-sort is T(n) = 2T(n/2)+n or (T(n) = 2 T(n/2) + O(n)) The worst case and average case recurrence relations are the same. The recurrence for this binary search is T(n) = T(n/2)+n or T(n) = 2T(n/2)+n Which was solved in solved in Q3-v

So this is Case 2 and O(n log n) or Θ(n log n)

The recurrence for quicksort (worst case) is T(n) = T(0)+T(n-1)+n = T(n-1)+n. So this is Case 2 and O(n log n) or Θ(n log n). I also gave credit for $O(n^2)$

The recurrence for quicksort (average case) is T(n) = 2T(n/2)+n. (Which is the recurrence for merge-sort and solved above)

See http://en.wikipedia.org/wiki/Quicksort#Average-case_analysis_using_recurrences

Q7 (5 Points)

Arrange the following functions in increasing order of growth:

- log(55n)
- $55^n + 11^n$
- $0.99^n + 1$
- n / log(n)
- $\log(n) + n^{0.5}$
- log(log(n))
- $\log(n^3)$
- $n!\ e^n$
- √n
- 5^n

Solution:

1. 0.99^n + 1 Θ(1) (Exponentially decreasing)
2. log(log n) Θ(log(log n))
3. log 55n Θ(log n)
4. log (n^3) Θ(log n)
5. √n Θ($n^{1/2}$)
6. log n + $n^{0.5}$ Θ($n^{0.5}$)

7. n / log(n) Θ(n/log n)
8. 5^n Θ(5^n)
9. $55^n + 11^n$ Θ(55^n) (Exponentially increasing)
10. n! e^n Θ(n(n+1/2))

Q8 (15 Points)

Consider a randomized version of SAT called Max-SAT which tries to satisfy as many clauses as possible with a random polynomial-time algorithm. More precisely, we define Max-SAT as follows: Given a set of k clauses C= {C_1, C_2 … C_k} and n literals X = {X_1, X_2 … X_n} find a truth assignment satisfying as many clauses as possible. Each clause must have at least on literal in it, and all of the literals within a single clause are distinct.

A. What is the expected number of satisfied clauses if each clause has just one literal and we randomly assign the truth value by flipping a fair coin?
B. If each clause has just one literal can we always find a solution that will satisfy all k clauses?
C. Consider a Max-SAT problem whose clauses can have any number of n literals. That is with k clauses, n literals and each clause can have from 1 up to n literals (1,2, … n literals in a clause). Give a randomized polynomial-time algorithm that satisfies at least 55% of the k clauses.

Solution:

A. What is the expected number of satisfied clauses if each clause has just one literal and we randomly assign the truth value by flipping a fair coin?

Consider a clause C with just one literal probability that it is satisfied is 1/2. If you want to say that the literals assigned to clauses are independent then the expectation of the sum of the clauses is the same the expected number of satisfied clauses which is k/2 clauses. If you made an argument that clauses are not independent then an answer of < k/2 clauses was also accepted.

B. If each clause has just one literal can we *always* find a solution that will satisfy all k clauses?

No. Counter example C_1 = x and C_2 is not x {x} ^ {~x}

C. Consider a Max-SAT problem whose clauses can have any number of n literals. That is with k clauses, n literals and each clause can have from 1 up to n literals (1,2, … n literals in a clause). Give a randomized polynomial-time algorithm that satisfies at least 55% of the k clauses.

Note: A wide range of reasonable arguments was accepted for full credit.

Notice that for 2 literals in a clause the coin-flipping algorithm that satisfies $(2^2 – 1)/2^2$ or 75% of the k clauses. The proportion increase $(2^n – 1)/2^n$. So to get 55% we really just need to improve the coin-flipping algorithm for single literal clauses or show that the expected value for the coin-flipping algorithm is ≥ 55%. For variables that occur in single variable clauses, let the probability of setting the variable so as to satisfy the clause be p ≥ 1/2. Now for a clause C, with l literals, l ≥ 2, the probability of satisfying it is at worst $(1-1/2^l) \geq (1 – p^2)$ since p ≥ 1/2 . Now to solve for p, we want to satisfy all clauses, so solve $p = 1 – p^2$ to get p ≥ 0.62. And hence the expected number of satisfied clauses

is 0.62k. So just coin-flipping gets p ≥ 55% when each clause can have from 1 up to n literals (1,2, … n literals in a clause)

To improve on this we can use a gradient descent type local search. Let the total number of clauses be k. For each non-satisfied clause flip the state of the literal and accept the change if the overall number of clauses has increased. Assume we have removed m non-satisfied clauses. Our expectation from above is to satisfy an additional 0.62 m of these. So we get a 0.62k satisfied clauses and 0.38k non-satisfied clauses with coin flipping and 0.62k satisfied clauses and (0.62k)(0.38k) additional clauses with the gradient descent type local search improvement. This analysis uses the assumption that literals assigned to clauses are independent (from the book and slides). This is a pretty big assumption and one would want to verify that empirically with code.

Q9 (5 Points)
What is the backpropagation algorithm?

Solution:

Backpropagation, short for "backward propagation of errors," is an algorithm for supervised learning of artificial neural networks using gradient descent. Given an artificial neural network and an error function, the method calculates the gradient of the error function.

Q10 (5 Points)
What is a Convolutional Neural Network (CNN)?

Solution:

A Convolutional Neural Network (CNN) is comprised of one or more convolutional layers (often with a subsampling step) and then followed by one or more fully connected layers as in a standard multilayer neural network. The architecture of a CNN is designed to take advantage of the 2D structure of an input image (or other 2D input such as a speech signal). This is achieved with local connections and tied weights followed by some form of pooling which results in translation invariant features. Another benefit of CNNs is that they are easier to train and have many fewer parameters than fully connected networks with the same number of hidden units. In this article we will discuss the architecture of a CNN and the back propagation algorithm to compute the gradient with respect to the parameters of the model in order to use gradient based optimization.

Q11 (5 Points)
Stacked Ensembles can almost always outperform a single model and are often at the top of H2O leaderboards. What is a Stacked Ensemble?

model_id	auc	logloss	mean_per_class_error	rmse	mse
StackedEnsemble_AllModels_AutoML_20190208_233938	0.949618	0.180591	0.148592	0.227044	0.0515489
StackedEnsemble_BestOfFamily_AutoML_20190208_233938	0.949501	0.18083	0.14058	0.227047	0.0515502
XGBoost_2_AutoML_20190208_233938	0.948528	0.173139	0.147784	0.225906	0.0510337
XGBoost_1_AutoML_20190208_233938	0.948208	0.1731	0.149	0.2262	0.0511664
GBM_4_AutoML_20190208_233938	0.948131	0.173882	0.148656	0.226523	0.0513126
XGBoost_3_AutoML_20190208_233938	0.946808	0.175423	0.164833	0.22719	0.0516155
GBM_3_AutoML_20190208_233938	0.946387	0.176609	0.142449	0.228175	0.0520639
GBM_2_AutoML_20190208_233938	0.944955	0.179339	0.155776	0.229637	0.0527332
GBM_1_AutoML_20190208_233938	0.94195	0.18388	0.164705	0.232089	0.0538654
XRT_1_AutoML_20190208_233938	0.935325	0.203641	0.169474	0.242563	0.0588367

Solution:

Stacking, also called Super Learning or Stacked Regression, is a class of algorithms that involves training a second-level "metalearner" to find the optimal combination of the base diverse models. Unlike bagging and boosting, the goal in stacking is to ensemble strong, diverse sets of learners together.

Q12 (5 Points)
How do we partition the data into training, validation and test sets in H2O?

Solution:

```
# Partition data into 70%, 15%, 15% chunks

train,valid, test = data.split_frame(ratios=[0.7, 0.15])
```

Q13 (5 Points)
Below is a Confusion Matrix (Act/Pred) for max f1 threshold by H2O. What is max f1?

Confusion Matrix (Act/Pred) for max f1 @ threshold = 0.193355175264:

	0	1	Error	Rate
0	13646.0	6345.0	0.3174	(6345.0/19991.0)
1	1939.0	2651.0	0.4224	(1939.0/4590.0)
Total	15585.0	8996.0	0.337	(8284.0/24581.0)

Solution:

In statistical analysis of binary classification, the F1 score (also F-score or F-measure) is a measure of a test's accuracy. It considers both the precision p and the recall r of the test to compute the score: p is the number of correct positive results divided by the number of all positive results returned by the classifier, and r is the number of correct positive results divided by the number of all relevant samples (all samples that should have been identified as positive). The F1 score is the harmonic average of the precision and recall, where an F1 score reaches its best value at 1 (perfect precision and recall) and worst at 0.

Q14 (5 Points)

Explain about transformations and actions in the context of RDDs. Give two examples.

Solution:

Transformations are functions executed on demand, to produce a new RDD. All transformations are followed by actions. Some examples of transformations include map, filter and reduceByKey.

Actions are the results of RDD computations or transformations. After an action is performed, the data from RDD moves back to the local machine. Some examples of actions include reduce, collect, first, and take.

Q15 (5 Points)

You have 2 dataframes df1 and df2. Both dataframes have common column called "IncidentNumber". Write a Query to create df3 by joining these 2 dataframes and print the schema of Df3.

Solution:

```
Df3 = df1.join(df2,df1.IncidentNumber == df2.IncidentNumber)
Df3.printSchema()
```