# Midterm - 10-605

Oct 18, 2016

**10-605** **Name:**_____
**Fall 2016**
**Midterm Exam** **Andrew ID:**_____
**Time Limit: 80 Minutes**

---

Grade Table (for teacher use only)

| Question | Points | Score |
|---|---|---|
| 1 | 10 | |
| 2 | 4 | |
| 3 | 10 | |
| 4 | 12 | |
| 5 | 12 | |
| 6 | 8 | |
| 7 | 8 | |
| 8 | 8 | |
| 9 | 6 | |
| 10 | 12 | |
| Total: | 90 | |

1. (10 points) SGD: true or false and multiple choice. Circle the correct answer.
   (a) The lazy update of regularized stochastic gradient descent algorithm for logistic regression on a sparse data set is an approximation to the normal regularized stochastic gradient descent algorithm. **True** **False**
   (b) SGD for logistic regression is independent of the order of the training data shown to it: **True** **False**
   (c) The hashing trick would be useful for a dataset with dense examples, e.g., where each example is a vector of daily stock prices for the last week. **True** **False**
   (d) The hash trick is sometimes called a "hash kernel" because it approximates the true feature values. **True** **False**

2. (4 points) The *block size M* of a filesystem is the minimum amount of physically contiguous disk space allocated when storing a file.
   (a) Suppose you have two hard disk drives – one does not contain any data and the other is nearly full with only 120MB space left. The filesystem block-size is 8KB and you store a 100MB file on both of them. Now you try to read the file sequentially. Which drive would be faster to read from? Why?

   (b) Circle the correct answer: the block-size for the Hadoop File System **larger/smaller** than a regular filesystem? (Select one)

3. (10 points) You are given a large and noisy Knowledge-Base (KB) of facts extracted from text. Each fact consists of a triple of the form *(head, relation, tail)* which expresses a relation between the head and tail entities. Your job is to remove the following redundancies from the KB:

   A. Duplicate triples – which express the same relation in different words. Ex: *(Obama, presidentOf, USA)* and *(Obama, leaderOf, USA)*.

   B. Inverse triples – which express the same relation in reverse order. Ex: *(Obama, presidentOf, USA)* and *(USA, hasPresident, Obama)*.

   As one step of this process, you need to implement a MapReduce program to collect all relations between a pair of entities along with their directions. Show this program below (pseudo-code is OK). For example, one line in the output of your program might look like – *[(Obama,USA), ((presidentOf,forward), (leaderOf,forward), (hasPresident,reverse))]*. Here *forward* and *reverse* indicate the direction of the relation.

4. (12 points) (a) (4 points) Circle the correct answer:
   Spark RDDs support two types of operations: **transformations / actions**, which lazily create a new RDDs from existing ones, and **transformations / actions**, which return a value to the driver program after running a computation on the RDDs.

   (b) (4 points) Both Hadoop and Spark are popular frameworks to process large-scale data. Briefly describe the difference between them and give a use case where Spark should outperform Hadoop

   (c) (4 points) The following is a Python code for training a logistic regression model using Spark. You are allowed to modify *one line* of the code to speed it up. Edit the program and write a short explanation of your change. (The output of the program should be the same.)

```
#Assume get_gradient is a function
#returning the gradient of w given a point p
points = spark.textFile("train").map(parsePoint)
w = numpy.zeros(10)
for i in range(100):
    gradient = points \
        .map(lambda p: get_gradient(p, w)) \
        .reduce(lambda a, b: a + b)
    w -= 0.1 * gradient
```

5. (12 points) (a) (6 points) Briefly explain: what does the following piece of codes do?

```
data = ReadLines('data.txt') \
| Flatten(by = lambda line : line.strip().split()) \
| Filter(by = removeStopWords ) \
| Group(by = lambda w : w, reducingTo = ReduceToCount())
```

```
output = ReadLines('labels.txt') \
| Map(by = lambda line: line.strip().split()) \
| JoinTo( Jin(data, by = lambda (w,c): w), by = lambda(w, label): w) \
| Map(by = lambda ((w,c),(w,label)) : (w,c,label))
```

How many abstract map-reduce tasks will be performed by the code?

(b) (6 points) In class, we gave an example of how the "ReduceTo" object is used in GuineaPig: in particular

```
Group(wc, by=lambda (w,c):w, retaining=lambda (w,c):c,
    reducingTo=ReduceToSum())
```

is equivalent to

```
Group(wc, by=lambda (w,c):w,
    reducingTo=ReduceTo(int, lambda (accum,(w,c)): accum+c))
```

Suppose we have a GPig view that contains the word counts for each document. Its format is **(docid, word, count)**. Now we want to obtain a list of all the word counts within a document.

Write an Group command using a custom "ReduceTo" object but NOT using the "retaining" keyword. You can write helper functions if you like.

Hint: the arguments to "ReduceTo"' are both functions, and when called with no arguments, the Python function "int" acts the same as the function "initAccum" below:

```
def initAccum():
    return 0
```

6. (8 points) The following figure plots three different methods for the same Named Entity Recognizer (NER) task.

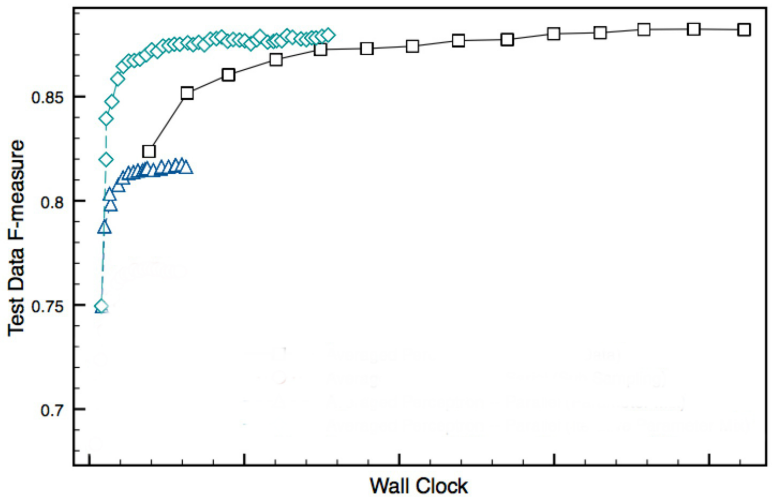


Figure 1: Perceptron Performance

(a) (2 points) The line marked with □ corrresponds to the method
   A. Serial Averaged Perceptron
   B. Iterative Parameter Mixing with Averaged Perceptron
   C. Parameter Mixing with Average Perceptron

(b) (2 points) The line marked with ◊ corresponds to the method
   A. Serial Averaged Perceptron
   B. Iterative Parameter Mixing with Averaged Perceptron
   C. Parameter Mixing with Average Perceptron

(c) (2 points) Give one reason why the line marked with ◊ converges to a higher F-measure score than the line marked with △

7. (8 points) Averaged Perceptron

As we know, the averaged perceptron calculates the average weight vector $\mathbf{v} = \frac{1}{Tm}\sum_{t=1..T,i=1..m}\mathbf{v}_{t,i}$, where $m$ is the number if examples and $T$ is the number of epochs. It is an approximation to the voted perceptron. The following pseudo-code is an implementation of a structured averaged perceptron with lazy updates. Fill in the 3 blanks below.

**Inputs:**

training data $(\mathbf{x}_1, \mathbf{y}_1), \ldots, (\mathbf{x}_m, \mathbf{y}_m)$

number of epochs $T$

**Initialize:**

$\mathbf{vk} = 0$

$\mathbf{va} = 0$

**Learning:**

for each epoch $t = 1, \ldots, T$

for each example $(\mathbf{x}_i, \mathbf{y}_i)$:

$\mathbf{y} = \arg\max_{\mathbf{y}'} F(\mathbf{x}_i, \mathbf{y}') \cdot \mathbf{vk}$

**correction** $= F(\mathbf{x}_i, \mathbf{y}_i) - F(\mathbf{x}_i, \mathbf{y})$

$\mathbf{vk} = \mathbf{vk} +$ _____

$\mathbf{va} = \mathbf{va} +$ (_____) $*$ _____

return $\frac{1}{Tm}\mathbf{va}$

8. (8 points) **Element Interchangeability and Strata**

Below are some decompositions of a matrix into blocks, and selection of a subset of blocks into "strata". Here "$X$" indicates the blocks of the matrix that are included in a stratum, while "." indicates the blocks that are excluded from the stratum. Gemulla et al in their pape4r on distributed matrix factorization define a stratum as *valid* if all blocks are interchangeable, and *efficient* if it is at large as it can be, without compromising interchangeability.

For each decomposition below, annotate it as "valid" if it is valid, and "efficient" if it is efficient. If it's not valid/efficient, give a very brief explanation why.

A.

$$X = \begin{bmatrix} X & . & . & . & . \\ . & X & . & . & . \\ . & . & X & . & . \\ . & . & . & . & . \\ . & . & . & . & X \end{bmatrix}$$

B.

$$Y = \begin{bmatrix} X & . & . & . & X \\ . & X & . & X & . \\ . & . & X & . & . \\ . & X & . & X & . \\ X & . & . & . & X \end{bmatrix}$$

C.

$$Z = \begin{bmatrix} X & . & . & . & . \\ . & . & . & . & X \\ . & . & X & . & . \\ . & X & . & . & . \\ . & . & . & X & . \end{bmatrix}$$

D.

$$W = \begin{bmatrix} . & . & . & . & X \\ . & . & . & X & . \\ . & . & X & . & . \\ . & X & . & . & . \\ X & . & . & . & . \end{bmatrix}$$

9. (6 points) **DSGD for Matrix Factorization**

   (a) Circle true or false:
   The Gemulla et al method discussed in class does *not* need global lock to avoid conflicts of updating parameters: **True** **False**

10. (12 points) **Hadoop**

   (a) Circle true or false:

   (2 points) Hadoop is suitable for iterative algorithms (e.g. PageRank, k-means, SGD), because distributed nodes can work in parallel.
   **True** **False**

   (2 points) Reducers do not start reducing until all Mappers are done. (Shuffling is not counted in)
   **True** **False**

   (b) (8 points) Rachel works in Yelp. She's given a task to generate the **probability of a restaurant being rated as 5 stars** for all the restaurants in the Yelp database.

   **Input:** key: *restaurant_id*, value: *rating*.
   **Output:** key: *restaurant_id*, value: $P(rating = 5)$

   Write a one-pass Map-Reduce program to do that, and optimize it with **Combiner** if you can. You don't have to write the actual code, but please specify the key-value pairs for the **Mapper**, **Combiner**, and **Reducer** respectively.

   - Reminder: The Combiner has a constraint that input/output key and value types must match the output types of your Mapper.
   - Hint: you may generate tuple as value.