Exam Review Session

William Cohen
General hints in studying

• Understand what you’ve done and why
  – There will be questions that test your understanding of the techniques implemented
    • why will/won’t this shortcut work?
    • what does the analysis say about the method?
• We’re mostly about learning meets computation here
  – there won’t be many “pure 601” questions
General hints in studying

• Techniques covered in class but not assignments:
  – When/where/how to use them
  – That usually includes understanding the \textit{analytic} results presented in class
  – Eg:
    • is the lazy regularization update an approximation or not? when does it help? when does it not help?
General hints in studying

• What about assignments you haven’t done?
  – You should **read through the assignments** and be familiar with the algorithms being implemented

• There won’t be questions about programming details that you could look up on line
  – but you should know how architectures like Hadoop work (eg, when and where they communicate)
  – you should be able to sketch out simple map-reduce algorithms
  – No spark, but you should be able to read workflow operators and discuss how they’d be implemented
    • how would you do a groupByKey in Hadoop?
General hints in studying

• There are not detailed questions on the guest speakers or the student projects (this year)
• If you use a previous year’s exam as guidance – the topics are a little different each year
General hints in exam taking

• You can bring in one 8 ½ by 11” sheet (front and back)
• Look over everything quickly and skip around
  – probably nobody will know everything on the test
• If you’re not sure what we’ve got in mind: state your assumptions clearly in your answer.
  – There’s room for this even on true/false
• If you look at a question and don’t know the answer:
  – we probably haven’t told you the answer
  – but we’ve told you enough to work it out
  – imagine arguing for some answer and see if you like it
Outline – major topics

• Hadoop
  – stream-and-sort is how I ease you into that, not really a topic on its own
• Parallelizing learners (perceptron, LDA, ...)
• Hash kernels and streaming SGD
• Distributed SGD for Matrix Factorization
• Randomized Algorithms
• Graph Algorithms: PR, PPR, SSL on graphs
  – no assignment != not on test
• Some of these are easier to ask questions about than others.
HADOOP
What data gets lost if the job tracker is rebooted? If I have a 1Tb file and shard it 1000 ways will it take longer than sharding it 10 ways? Where should a combiner run?
$ hadoop fs -ls rcv1/small/sharded
Found 10 items
-rw-r--r-- 3 ...  606405 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00000
-rw-r--r-- 3 ...  1347611 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00001
-rw-r--r-- 3 ...  939307 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00002
-rw-r--r-- 3 ...  1284062 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00003
-rw-r--r-- 3 ...  1009890 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00004
-rw-r--r-- 3 ...  1206196 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00005
-rw-r--r-- 3 ...  1384658 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00006
-rw-r--r-- 3 ...  1299698 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00007
-rw-r--r-- 3 ...   928752 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00008
-rw-r--r-- 3 ...   806030 2013-01-22 16:28 /user/wcohen/rcv1/small/sharded/part-00009

$ hadoop fs -tail rcv1/small/sharded/part-00005
weak as the arrival of arbitraged cargoes from the West has put the local market under pressure...
M14,M143,MCAT The Brent crude market on the Singapore International ...

Where is this data? How many disks is it on?
If I set up a directory on /afs that looks the same will it work the same? what about a local disk?
### Hadoop job_201301231150_0778 on hadoopjt

**User:** wcohen  
**Job Name:** streamjob6055532903853567038.jar  
**Job File:** hdfs://hdfsname.opencloud/l/a2/scratch/hadoop-data/global/mapred/system/job_201301231150_0778/job.xml  
**Job Setup:** Successful  
**Status:** Failed  
**Started at:** Wed Jan 30 11:46:47 EST 2013  
**Failed at:** Wed Jan 30 11:47:28 EST 2013  
**Failed in:** 41sec  
**Job Cleanup:** Successful  
**Black-listed TaskTrackers:** 2

**Job Scheduling information:** 5 running map tasks using 5 map slots, 0 running reduce tasks using 0 reduce slots.

<table>
<thead>
<tr>
<th>Kind</th>
<th>% Complete</th>
<th>Num Tasks</th>
<th>Pending</th>
<th>Running</th>
<th>Complete</th>
<th>Killed</th>
<th>Failed/Killed Task Attempts</th>
</tr>
</thead>
<tbody>
<tr>
<td>map</td>
<td>100.00%</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>35 / 5</td>
</tr>
<tr>
<td>reduce</td>
<td>0%</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>0 / 0</td>
</tr>
</tbody>
</table>

### Job Counters

<table>
<thead>
<tr>
<th>Counter</th>
<th>Map</th>
<th>Reduce</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Rack-local map tasks</td>
<td>0</td>
<td>0</td>
<td>38</td>
</tr>
<tr>
<td>Launched map tasks</td>
<td>0</td>
<td>0</td>
<td>40</td>
</tr>
<tr>
<td>Data-local map tasks</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>Failed map tasks</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

**Map Completion Graph** - close
## All Tasks

<table>
<thead>
<tr>
<th>Task</th>
<th>Complete</th>
<th>Status</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>Errors</th>
</tr>
</thead>
</table>
| task_201301231150_0778   | 0.00%    |            | 30-Jan-2013 11:47:01 | 30-Jan-2013 11:47:25 (24sec) | java.lang.RuntimeException: PipeMainVolatilityCheck 
  at org.apache.hadoop.streaming.main.PipeMainVolatilityCheck 
  at org.apache.hadoop.streaming.main.PipeMain 
  at org.apache.hadoop.streaming.PipeDriver.run(PipeDriver.java:386) 
  at java.lang.Thread.run(Thread.java:745) |
|                          |          |            |              |             | java.lang.RuntimeException: PipeMainVolatilityCheck 
  at org.apache.hadoop.streaming.main.PipeMainVolatilityCheck 
  at org.apache.hadoop.streaming.main.PipeMain 
  at org.apache.hadoop.streaming.PipeDriver.run(PipeDriver.java:386) 
  at java.lang.Thread.run(Thread.java:745) |
|                          |          |            |              |             | java.lang.RuntimeException: PipeMainVolatilityCheck 
  at org.apache.hadoop.streaming.main.PipeMainVolatilityCheck 
  at org.apache.hadoop.streaming.main.PipeMain 
  at org.apache.hadoop.streaming.PipeDriver.run(PipeDriver.java:386) 
  at java.lang.Thread.run(Thread.java:745) |
# Job job_201301231150_0778

## All Task Attempts

<table>
<thead>
<tr>
<th>Task Attempts</th>
<th>Machine</th>
<th>Status</th>
<th>Progress</th>
<th>Start Time</th>
<th>Finish Time</th>
<th>Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>attempt_201301231150_0778_m_000000_0</td>
<td>/default-rack/cloud3u12.opencloud</td>
<td>FAILED</td>
<td>0.00%</td>
<td>30-Jan-2013 11:47:01</td>
<td>30-Jan-2013 11:47:06 (4sec)</td>
<td>java.</td>
</tr>
<tr>
<td>attempt_201301231150_0778_m_000000_1</td>
<td>/default-rack/cloud2u28.opencloud</td>
<td>FAILED</td>
<td>0.00%</td>
<td>30-Jan-2013 11:47:07</td>
<td>30-Jan-2013 11:47:11 (4sec)</td>
<td>java.</td>
</tr>
</tbody>
</table>
Why do I see this same error over and over again?
PARALLEL LEARNERS
Parallelizing perceptrons – take 2

Split into example subsets

Compute local $v_k$'s

Combine by some sort of weighted averaging
Theorem 3. Assume a training set $\mathcal{T}$ is separable by margin $\gamma$. Let $k_{i,n}$ be the number of mistakes that occurred on shard $i$ during the $n$th epoch of training. For any $N$, when training the perceptron with iterative parameter mixing (Figure 3),

$$\sum_{n=1}^{N} \sum_{i=1}^{S} \mu_{i,n}k_{i,n} \leq \frac{R^2}{\gamma^2}$$

**Corollary:** if we weight the vectors uniformly, then the number of mistakes is still bounded.

**I.e.,** this is “enough communication” to guarantee convergence.

I probably won’t ask about the proof, but I could definitely ask about the theorem.
What does the word “structured” mean here? why is it important? would the results be better or worse with a regular perceptron?
STREAMING SGD
Learning as optimization for regularized logistic regression

• Algorithm: 

\[ w^j = w^j + \lambda (y - p) x^j - \lambda 2\mu w^j \]

1. Initialize a hashtable \( W \)

2. For \( t = 1, \ldots, T \)
   • For each example \( x_i, y_i \):
     - Compute the prediction for \( x_i \):
       \[
p_i = \frac{1}{1 + \exp(-\sum_{j : x_i^j > 0} x_i^j w^j)}
       \]
     - For each non-zero feature of \( x_i \) with index \( j \) and value \( x^j \):
       * If \( j \) is not in \( W \), set \( W[j] = 0 \).
       * Set \( W[j] = W[j] + \lambda (y - p) x^j - \lambda 2\mu w^j \)

3. Output the hash table \( W \).
Formalization of the “Hash Trick”:

First: Review of Kernels

What is it? how does it work? what aspects of performance does it help? What did we say about it formally?
RANDOMIZED ALGORITHMS
Randomized Algorithms

• Hash kernels
• Countmin sketch
• Bloom filters
• LSH

• What are they, and what are they used for? When would you use which one?
• Why do they work - ie, what analytic results have we looked at?
Bloom filters - review

• An implementation
  – Allocate M bits, bit[0]..., bit[1-M]
  – Pick K hash functions hash(1,2), hash(2,s), ....
    • E.g: hash(i,s) = hash(s + randomString[i])
  – To add string s:
    • For i=1 to k, set bit[hash(i,s)] = 1
  – To check contains(s):
    • For i=1 to k, test bit[hash(i,s)]
    • Return “true” if they’re all set; otherwise, return “false”
  – We’ll discuss how to set M and K soon, but for now:
    • Let M = 1.5*maxSize // less than two bits per item!
    • Let K = 2*log(1/p) // about right with this M
Bloom filters

• An example application
  – discarding rare features from a classifier
  – seldom hurts much, can speed up experiments
• Scan through data once and check each \( w \):
  – if \( \text{bf1}.\text{contains}(w) \):
    • if \( \text{bf2}.\text{contains}(w) \): \( \text{bf3}.\text{add}(w) \)
    • else \( \text{bf2}.\text{add}(w) \)
  – else \( \text{bf1}.\text{add}(w) \)
• Now:
  – \( \text{bf2}.\text{contains}(w) \Leftrightarrow w \) appears \( \geq 2x \)
  – \( \text{bf3}.\text{contains}(w) \Leftrightarrow w \) appears \( \geq 3x \)
• Then train, ignoring words not in \( \text{bf3} \)

which needs more storage, \( \text{bf1} \) or \( \text{bf3} \)? (same false positive rate)
**Bloom filters**

- Here's two ideas for using Bloom filters for learning from sparse binary examples:
  - compress every example with a BF
  - either
    - use each bit of the BF as a feature for a classifier
    - or: reconstruct the example at training time and train an ordinary classifier

– pros and cons?
Each string is mapped to one bucket per row

Estimate $A[j]$ by taking $\min_k \{ \text{CM}[k,h_k(j)] \}$

Errors are always over-estimates

Sizes: $d=\log 1/\delta$, $w=2/\varepsilon \implies$ error is usually less than $\varepsilon \|A\|_1$
CM Sketch Guarantees

- **[Cormode, Muthukrishnan’04]**  CM sketch guarantees approximation error on point queries less than $\varepsilon \|A\|_1$ in space $O(1/\varepsilon \log 1/\delta)$
  - Probability of more error is less than $1-\delta$

- This is sometimes enough:
  - Estimating a multinomial: if $A[s] = \Pr(s|\ldots)$ then $\|A\|_1 = 1$
  - Multiclass classification: if $A_x[s] = \Pr(x \text{ in class } s)$ then $\|A_x\|_1$ is probably small, since most $x$’s will be in only a few classes
An Application of a Count-Min Sketch

Problem: find the semantic orientation of a work (positive or negative) using a large corpus.

Idea:
- positive words co-occur more frequently than expected near positive words; likewise for negative words
- so pick a few pos/neg seeds and compute

\[
\text{pmi}(x; y) = \log \frac{p(x, y)}{p(x)p(y)}
\]

\[
\text{SO}(w) = \sum_{p \in \text{Pos}} \text{PMI}(p, w) - \sum_{n \in \text{Neg}} \text{PMI}(n, w)
\]
LSH: key ideas

- **Goal:**
  - map feature vector \( \mathbf{x} \) to bit vector \( \mathbf{b}_x \)
  - ensure that \( \mathbf{b}_x \) preserves “similarity”

- **Basic idea:** use random projections of \( \mathbf{x} \)
  - Repeat many times:
    - Pick a random hyperplane \( \mathbf{r} \) by picking random weights for each feature (say from a Gaussian)
    - Compute the inner product of \( \mathbf{r} \) with \( \mathbf{x} \)
    - Record if \( \mathbf{x} \) is “close to” \( \mathbf{r} \) (\( \mathbf{r} \cdot \mathbf{x} \geq 0 \))
      - the next bit in \( \mathbf{b}_x \)
    - Theory says that if \( \mathbf{x}' \) and \( \mathbf{x} \) have small cosine distance then \( \mathbf{b}_x \) and \( \mathbf{b}_x' \) will have small Hamming distance
LSH applications

• Compact storage of data
  – and we can still compute similarities
• LSH also gives very fast approximations:
  – approx nearest neighbor method
    • just look at other items with \( b_x' = b_x \)
    • also very fast nearest-neighbor methods for Hamming distance
  – very fast clustering
    • cluster = all things with same \( b_x \) vector
LSH

• What are some other ways of using LSH?
• What are some other ways of using CountMin sketches?
GRAPH ALGORITHMS
Graph Algorithms

• The “standard” way of computing PageRank, iteratively, using the power method
• Properties of large data graphs
  — ...
• The “subsampling problem”
• APR algorithm - how and why it works
  — when we’d use it
  — what the analysis says
• SSL on graphs
  — example algorithms, example use of CM Sketches
<table>
<thead>
<tr>
<th>network</th>
<th>type</th>
<th>(n)</th>
<th>(m)</th>
<th>(z)</th>
<th>(l)</th>
<th>(\alpha)</th>
<th>(C^{(1)})</th>
<th>(C^{(2)})</th>
</tr>
</thead>
<tbody>
<tr>
<td>film actors</td>
<td>undirected</td>
<td>449 913</td>
<td>25 516 482</td>
<td>113.43</td>
<td>3.48</td>
<td>2.3</td>
<td>0.20</td>
<td>0.78</td>
</tr>
<tr>
<td>company directors</td>
<td>undirected</td>
<td>7 673</td>
<td>55 392</td>
<td>14.44</td>
<td>4.60</td>
<td>–</td>
<td>0.59</td>
<td>0.88</td>
</tr>
<tr>
<td>math coauthorship</td>
<td>undirected</td>
<td>253 339</td>
<td>496 489</td>
<td>3.92</td>
<td>7.57</td>
<td>–</td>
<td>0.15</td>
<td>0.34</td>
</tr>
<tr>
<td>physics coauthorship</td>
<td>undirected</td>
<td>52 909</td>
<td>245 300</td>
<td>9.27</td>
<td>6.19</td>
<td>–</td>
<td>0.45</td>
<td>0.56</td>
</tr>
<tr>
<td>biology coauthorship</td>
<td>undirected</td>
<td>1 520 251</td>
<td>11 803 064</td>
<td>15.53</td>
<td>4.92</td>
<td>–</td>
<td>0.088</td>
<td>0.60</td>
</tr>
<tr>
<td>telephone call graph</td>
<td>undirected</td>
<td>47 000 000</td>
<td>80 000 000</td>
<td>3.16</td>
<td></td>
<td>2.1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>email messages</td>
<td>directed</td>
<td>59 912</td>
<td>86 300</td>
<td>1.44</td>
<td>4.95</td>
<td>1.5/2.0</td>
<td>0.16</td>
<td></td>
</tr>
<tr>
<td>email address books</td>
<td>directed</td>
<td>16 881</td>
<td>57 029</td>
<td>3.38</td>
<td>5.22</td>
<td>–</td>
<td>0.17</td>
<td>0.13</td>
</tr>
<tr>
<td>student relationships</td>
<td>undirected</td>
<td>573</td>
<td>477</td>
<td>1.66</td>
<td>16.01</td>
<td>–</td>
<td>0.005</td>
<td>0.001</td>
</tr>
<tr>
<td>sexual contacts</td>
<td>undirected</td>
<td>2 810</td>
<td>3 2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>WWW nd.edu</td>
<td>directed</td>
<td>269 504</td>
<td>1 497 135</td>
<td>5.55</td>
<td>11.27</td>
<td>2.1/2.4</td>
<td>0.11</td>
<td>0.29</td>
</tr>
<tr>
<td>WWW Altavista</td>
<td>directed</td>
<td>203 549 046</td>
<td>2 130 000 000</td>
<td>10.46</td>
<td>16.18</td>
<td>2.1/2.7</td>
<td></td>
<td></td>
</tr>
<tr>
<td>citation network</td>
<td>directed</td>
<td>783 339</td>
<td>6 716 198</td>
<td>8.57</td>
<td></td>
<td>3.0/-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Roget’s Thesaurus</td>
<td>directed</td>
<td>1 022</td>
<td>5 103</td>
<td>4.99</td>
<td>4.87</td>
<td>–</td>
<td>0.13</td>
<td>0.15</td>
</tr>
<tr>
<td>word co-occurrence</td>
<td>undirected</td>
<td>460 902</td>
<td>17 000 000</td>
<td>70.13</td>
<td></td>
<td>2.7</td>
<td></td>
<td>0.44</td>
</tr>
<tr>
<td>Internet</td>
<td>undirected</td>
<td>10 697</td>
<td>31 992</td>
<td>5.98</td>
<td>3.31</td>
<td>2.5</td>
<td>0.035</td>
<td>0.39</td>
</tr>
<tr>
<td>power grid</td>
<td>undirected</td>
<td>4 941</td>
<td>6 594</td>
<td>2.67</td>
<td>18.99</td>
<td>–</td>
<td>0.10</td>
<td>0.080</td>
</tr>
<tr>
<td>train routes</td>
<td>undirected</td>
<td>587</td>
<td>19 603</td>
<td>66.79</td>
<td>2.16</td>
<td>–</td>
<td></td>
<td>0.69</td>
</tr>
<tr>
<td>software packages</td>
<td>directed</td>
<td>1 439</td>
<td>1 723</td>
<td>1.20</td>
<td>2.42</td>
<td>1.6/1.4</td>
<td>0.070</td>
<td>0.082</td>
</tr>
<tr>
<td>software classes</td>
<td>directed</td>
<td>1 377</td>
<td>2 213</td>
<td>1.61</td>
<td>1.51</td>
<td>–</td>
<td>0.033</td>
<td>0.012</td>
</tr>
<tr>
<td>electronic circuits</td>
<td>undirected</td>
<td>24 097</td>
<td>53 248</td>
<td>4.34</td>
<td>11.05</td>
<td>3.0</td>
<td>0.010</td>
<td>0.030</td>
</tr>
<tr>
<td>peer-to-peer network</td>
<td>undirected</td>
<td>880</td>
<td>1 296</td>
<td>1.47</td>
<td>4.28</td>
<td>2.1</td>
<td>0.012</td>
<td>0.011</td>
</tr>
<tr>
<td>metabolic network</td>
<td>undirected</td>
<td>765</td>
<td>3 686</td>
<td>9.64</td>
<td>2.56</td>
<td>2.2</td>
<td>0.090</td>
<td>0.67</td>
</tr>
<tr>
<td>protein interactions</td>
<td>undirected</td>
<td>2 115</td>
<td>2 240</td>
<td>2.12</td>
<td>6.80</td>
<td>2.4</td>
<td>0.072</td>
<td>0.071</td>
</tr>
<tr>
<td>marine food web</td>
<td>directed</td>
<td>135</td>
<td>598</td>
<td>4.43</td>
<td>2.05</td>
<td>–</td>
<td>0.16</td>
<td>0.23</td>
</tr>
<tr>
<td>freshwater food web</td>
<td>directed</td>
<td>92</td>
<td>997</td>
<td>10.84</td>
<td>1.90</td>
<td>–</td>
<td>0.20</td>
<td>0.087</td>
</tr>
<tr>
<td>neural network</td>
<td>directed</td>
<td>307</td>
<td>2 359</td>
<td>7.68</td>
<td>3.97</td>
<td>–</td>
<td>0.18</td>
<td>0.28</td>
</tr>
</tbody>
</table>
Degree distribution

- Plot cumulative degree
  - X axis is degree
  - Y axis is #nodes that have degree at least $k$
- Typically use a log-log scale
  - Straight lines are a power law; normal curve dives to zero at some point
  - Left: trust network in epinions web site from Richardson & Domingos
Homophily

• Another def’n: excess edges between common neighbors of \( v \)

\[
CC(v) = \frac{\# \text{triangles connected to } v}{\# \text{pairs connected to } v}
\]

\[
CC(V, E) = \frac{1}{|V|} \sum_v CC(v)
\]

\[
CC'(V, E) = \frac{\# \text{triangles in graph}}{\# \text{length 3 paths in graph}}
\]
Approximate PageRank: Algorithm

ApproximatePageRank \((v, \alpha, \epsilon)\):

1. Let \(p = \vec{0}\), and \(r = \chi_v\).

2. While \(\max_{u \in V} \frac{r(u)}{d(u)} \geq \epsilon\):
   
   (a) Choose any vertex \(u\) where \(\frac{r(u)}{d(u)} \geq \epsilon\).
   
   (b) Apply \text{push}_u\) at vertex \(u\), updating \(p\) and \(r\).

3. Return \(p\), which satisfies \(p = \operatorname{apr}(\alpha, \chi_v, r)\) with \(\max_{u \in V} \frac{r(u)}{d(u)} < \epsilon\).

\text{push}_u(p, r):

1. Let \(p' = p\) and \(r' = r\), except for the following changes:
   
   (a) \(p'(u) = p(u) + \alpha r(u)\).
   
   (b) \(r'(u) = (1 - \alpha)r(u)/2\).
   
   (c) For each \(v\) such that \((u, v) \in E\): \(r'(v) = r(v) + (1 - \alpha)r(u)/(2d(u))\).

2. Return \((p', r')\).
Approximate PageRank: Key Idea

By definition PageRank is fixed point of:

$$\text{pr}(\alpha, s) = \alpha s + (1 - \alpha) \text{pr}(\alpha, s) W,$$

$$\text{pr}(\alpha, s) = \alpha s + (1 - \alpha) \text{pr}(\alpha, sW).$$

Claim:

Recursively compute PageRank of “neighbors of s” (=sW), then adjust

Key idea in apr:

- do this “recursive step” repeatedly
- focus on nodes where finding PageRank from neighbors will be useful
Lemma 1. Let $p'$ and $r'$ be the result of the operation $\text{push}_u$ on $p$ and $r$. Then

$$p' + \text{pr}(\alpha, r') = p + \text{pr}(\alpha, r).$$

Proof of Lemma 1. After the push operation, we have

$$p' = p + \alpha r(u) \chi_u.$$  
$$r' = r - r(u) \chi_u + (1 - \alpha) r(u) \chi_u W.$$  

Using equation (5),

$$p + \text{pr}(\alpha, r) = p + \text{pr}(\alpha, r - r(u) \chi_u) + \text{pr}(\alpha, r(u) \chi_u)$$  
$$= p + \text{pr}(\alpha, r - r(u) \chi_u) + [\alpha r(u) \chi_u + (1 - \alpha) \text{pr}(\alpha, r(u) \chi_u W)]$$  
$$= [p + \alpha r(u) \chi_u] + \text{pr}(\alpha, [r - r(u) \chi_u + (1 - \alpha) r(u) \chi_u W])$$  
$$= p' + \text{pr}(\alpha, r').$$

$$\text{pr}(\alpha, r - r(u) \chi_u) + (1 - \alpha) \text{pr}(\alpha, r(u) \chi_u W) = \text{pr}(\alpha, r - r(u) \chi_u + (1 - \alpha) r(u) \chi_u W)$$

$$\text{pr}(\alpha, s) = \alpha s + (1 - \alpha) \text{pr}(\alpha, sW).$$
Q/A