Criss-Crossing the Org Chart
Predicting Colleague Interactions with R

Eric Sun <esun@facebook.com>
Facebook Data Science
useR! 2010 Focus Session: Social Networks
NIST, 7/21/2010
Introduction: How Facebook uses R

- Experimentation for large machine learning models
  - Picking models
  - Feature selection
- Small/medium one-off analyses
  - user engagement studies
  - analyses to improve internal processes
How Facebook uses R (cont.)

- Analysis and visualization for social networks research

from Chang, et al. (ICWSM 2009)

from Backstrom, et al. (WWW 2010)

- Facebook Data Team: http://www.facebook.com/data
Predicting Colleague Interactions
Potential Applications

- Suggesting peer reviewers during performance review season
- Setting up optimally-constructed teams within a company
- Optimizing seating charts for maximum productivity
- Automatically filtering internal feeds of employee content (such as commit logs) to deliver personalized content to each employee
- Suggesting new colleague interactions (based on second-degree connections) that may be useful to one’s work
- Giving managers more insight into their employees’ interactions
Goals

- Attempt to predict the total # of colleague interactions in the next 4 weeks across all internal tools
- Provide an API for other engineers to use in their internal tools, and publish a daily dashboard to show each employee their current results

Pipeline Overview
Final Results

- Daily cron job: R predictions -> RMySQL -> web page dashboard

<table>
<thead>
<tr>
<th>User</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alexander Strehl</td>
<td>9.3318008583172</td>
</tr>
<tr>
<td>Aaron Binbin Liao</td>
<td>8.1162385345038</td>
</tr>
<tr>
<td>Srinivas Narayan</td>
<td>3.293821239025</td>
</tr>
<tr>
<td>Jack Zhao</td>
<td>3.0129828038274</td>
</tr>
<tr>
<td>Austin Haugen</td>
<td>2.4928669082428</td>
</tr>
</tbody>
</table>
Predictors

- Direct communication metrics
  - # code reviews requested
  - # mailing list threads shared
  - # shared threads on internal task management tool
  - # shared threads on internal message boards

- Implicit interaction
  - # meetings co-attended

- Org chart dummy variables (manager, report, peer)
Feature Generation

- For each set, use Cartesian product to generate pairs of interactions
- All features weighted by \( \frac{1}{(# \text{ participants} - 1)} \)
- Example:
  - Alice, Bob, and Charlie attend a meeting.
  - Generate A->B, A->C, B->A, B->C, C->A, C->B with weight 0.5 for the ‘meeting’ variable, then aggregate across IDs and type
The Data

> crisscross.data <- read.csv('crisscross_training_data.csv')

> nrow(crisscross.data) # low-weight observations filtered out

[1] 98802

> crisscross.data[sample(nrow(crisscross.data), 5),]

<table>
<thead>
<tr>
<th>id1</th>
<th>id2 meeting</th>
<th>mailinglists</th>
<th>codereviews</th>
<th>tasks</th>
<th>messageboards</th>
<th>imageshare</th>
<th>manager</th>
<th>peer report</th>
<th>eval_weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;hidden&gt;</td>
<td>&lt;hidden&gt;</td>
<td>0.0000</td>
<td>1</td>
<td>0.2834</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;hidden&gt;</td>
<td>&lt;hidden&gt;</td>
<td>1.7833</td>
<td>0</td>
<td>0.0000</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;hidden&gt;</td>
<td>&lt;hidden&gt;</td>
<td>0.0041</td>
<td>0</td>
<td>0.2000</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;hidden&gt;</td>
<td>&lt;hidden&gt;</td>
<td>0.0000</td>
<td>0</td>
<td>0.0000</td>
<td>0.125</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>&lt;hidden&gt;</td>
<td>&lt;hidden&gt;</td>
<td>0.0000</td>
<td>0</td>
<td>0.1250</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>id1</th>
<th>id2 meeting</th>
<th>mailinglists</th>
<th>codereviews</th>
<th>tasks</th>
<th>messageboards</th>
<th>imageshare</th>
<th>manager</th>
<th>peer report</th>
<th>eval_weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;hidden&gt;</td>
<td>&lt;hidden&gt;</td>
<td>0.0000</td>
<td>0</td>
<td>0.1250</td>
<td>0.000</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>


Long Tail of Interactions
Machine Learning

- All subsequent stats computed on a single Linux machine
- Dual-core 2 Mhz server, 16GB RAM
- Tried a variety of techniques: linear regression, random forests, boosted trees, etc.
- Used standard 2/3 – 1/3 split for training and test data

```r
> nrow(full_data)
[1] 98802  # ~50 per employee

> nrow(test_data)
[1] 32934

> nrow(train_data)
[1] 65868
```
Linear Regression

```r
> system.time(crossvalidate.lm <- lm(eval_weight ~ ., data = train_data))
  user  system elapsed
 0.866  0.106  0.973
> summary(crossvalidate.lm)

Call:
  lm(formula = eval_weight ~ ., data = train_data)

Residuals:
  Min     1Q   Median     3Q    Max
-38.31223 -0.28494 -0.15137  0.02445  57.11911

Coefficients:
                     Estimate Std. Error t value Pr(>|t|)
(Intercept)       -0.071232   0.009224  -7.723 1.16e-14 ***
meeting           0.572376   0.024526   23.338  < 2e-16 ***
mailinglists      0.257495   0.009799   26.278  < 2e-16 ***
codereviews       0.693076   0.005563  124.598  < 2e-16 ***
tasks             0.890425   0.004206  211.704  < 2e-16 ***
messageboards     0.767063   0.063873   12.009  < 2e-16 ***
imageshare       0.810893   0.152074    5.332 9.73e-08 ***
manager           0.488565   0.046915   10.414  < 2e-16 ***
peer              0.255186   0.017801   14.336  < 2e-16 ***
report            0.483349   0.047139   10.254  < 2e-16 ***
---
Signif. codes:  < ****  0.001 ***  0.01 **  0.05 *  0.1 .  1

Residual standard error: 1.878 on 65858 degrees of freedom
Multiple R-squared:  0.5451,   Adjusted R-squared:  0.5451
F-statistic: 8770 on 9 and 65858 DF,  p-value: < 2.2e-16
```
Random Forests

```r
> require(randomForest)
Loading required package: randomForest
randomForest 4.5-30
Type rfNews() to see new features/changes/bug fixes.
> reg.x <- train_data[, -which(names(train_data) == 'eval_weight')]
> system.time(crisscross.rf <- randomForest(x = reg.x, y = train_data$eval_weight))

   user  system elapsed
   631.132  180.815  812.141
> summary(crisscross.rf)

   Length Class Mode
call     3  -none-  call
type     1  -none- character
predicted   65868  -none- numeric
mse       500  -none- numeric
rsq       500  -none- numeric
oob.times  65868  -none- numeric
importance 9  -none- numeric
importanceSD 0  -none- NULL
localImportance 0  -none- NULL
proximity  0  -none- NULL
ntree     1  -none- numeric
mtry      1  -none- numeric
forest    11  -none- list
coefs     0  -none- NULL
y         65868  -none- numeric
test      0  -none- NULL
inbag     0  -none- NULL
```
Boosted Trees

```r
> require(gbm)
Loading required package: gbm
Loading required package: survival
Loading required package: splines
Loading required package: lattice
Loaded gbm 1.6-3
> system.time(crisscross.gbm <- gbm(eval_weight ~ ., data = train_data,
+ n.trees = 1000, cv.folds = 5, distribution = 'laplace',
+ interaction.depth = 2))
  user  system elapsed
270.108  0.594  270.810
> summary(crisscross.gbm)
  var         rel.inf
1  tasks 84.97515128
2  meeting 15.00431911
3 peer  0.02052960
4 mailinglists 0.00000000
5 coderviews 0.00000000
6 messageboards 0.00000000
7 imageshare 0.00000000
8 manager 0.00000000
9 report  0.00000000
```
Comparison of Techniques

- Use held-out test set to evaluate results

<table>
<thead>
<tr>
<th></th>
<th>Linear Regression</th>
<th>Random Forests</th>
<th>Boosted Trees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Running Time (seconds)</td>
<td>0.973</td>
<td>812.141</td>
<td>270.81</td>
</tr>
<tr>
<td>Sum of Squared Errors</td>
<td>111,188.59</td>
<td>60,316.21</td>
<td>226,923.10</td>
</tr>
<tr>
<td>Mean Squared Error</td>
<td>3.74</td>
<td>2.03</td>
<td>7.64</td>
</tr>
<tr>
<td>Median Squared Error</td>
<td>0.09</td>
<td>0.05</td>
<td>0.01</td>
</tr>
<tr>
<td>Quantiles of Squared Errors:</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0%</td>
<td>0.00</td>
<td>0.00</td>
<td>0.00</td>
</tr>
<tr>
<td>20%</td>
<td>0.01</td>
<td>0.02</td>
<td>0.00</td>
</tr>
<tr>
<td>40%</td>
<td>0.04</td>
<td>0.03</td>
<td>0.00</td>
</tr>
<tr>
<td>60%</td>
<td>0.11</td>
<td>0.08</td>
<td>0.02</td>
</tr>
<tr>
<td>80%</td>
<td>0.55</td>
<td>0.35</td>
<td>0.28</td>
</tr>
<tr>
<td>100%</td>
<td>5,829.69</td>
<td>2,644.87</td>
<td>13,713.43</td>
</tr>
</tbody>
</table>
Summary & Future Work

- Remarkably simple, automatic pipeline delivers useful insight into organizational behavior
- Pipeline integrates R seamlessly with data stored in databases
- Many useful applications for internal tools

- To do: integrate into applications mentioned previously
- To do: explore visualization techniques with R graphics!

- These slides are posted at http://www.stanford.edu/~esun/