Predicting Optimality of Network Paths

Team #6
Aditya | Alok | Ankit | Dhara | Dhyey | Karthik | Shubham

Problem Statement

- Given a chronological series of n graphs, predicting the probability of optimality of paths in future k graphs

(If I have a known optimal way to reach the goal, can I predict if it will stay optimal?)

Business Value - Problem

- Mapping a large, dynamic graph (viz. internet) is an expensive operation
- More so, when the optimal path also has to be computed
- Objective of our project is to avoid that by using past information to predict future optimal paths rather than recalculating them
- At a given point of time, assuming the nodes remain the same, we can say with certain confidence which optimal route we should take in the future

Data Preprocessing

- Reduced the size from 350K entries to 37k entries
- Fingerprinting
- Generation of fingerprints of a string is a three stage process;
    - From standard string generate n-grams
    - Use rolling hash function to generates the hash values for each n-gram
    - Use winnowing to generate the fingerprints from the hash values
- Assigned Numbers to unique fingerprints

Why This is Important

- Before

ASVHC1 - Arlington Arlington Hospital
ASVHC1 - &%Arlington
ASVHC1 - Arlington - Hospital
Arlington ASVHC1 - Arlington Hospital

- After

1acghilnorstv

Benchmarking

- Calculating Historical Mode to match benchmarking
- Benchmark - Takes the mode over the 10 training graphs.
  If the given path is optimal in the graph, it receives a value of 1. If not, it gets a 0.

Features

	Timestamp1	Timestamp2	Timestamp3	Timestamp4	Timestamp5	Timestamp6	Timestamp7	Timestamp8	Timestamp9	Timestamp10	Timestamp11	Timestamp12
Path1	0	0	0	0	0	0	0	0	0	0	0	0
Path2	1	1	1	1	1	1	1	1	1	1	1	1
Path3	0	0	0	0	0	1	0	1	0	0	0	0
Path4	1	1	1	0	1	1	1	0	0	1	1	1

Algorithms

- Markov Chain
- Random Forest
- Gradient Boosting Machine
- Logistic Regression

Used *Weighted Decay* during feature extraction

Methodology

	Timestamp1	Timestamp2	Timestamp3	Timestamp4	Timestamp5	Timestamp6	Timestamp7	Timestamp8	Timestamp9	Timestamp10	Timestamp11	Timestamp12	Prediction
Path1	0	0	0	0	0	0	0	0	0	1	0	1	0.3333333333
Path2	1	1	0	0	0	0	1	0	0	0	1	0	0.3333333333
Path3	0	0	0	0	0	0	0	1	0	0	0	0	0.1666666667
Path4	1	1	1	1	1	1	1	1	1	1	1	1	0.9230769231
Path5	0	0	0	1	0	0	0	1	1	0	0	1	0.4
Path6	0	0	0	1	0	0	0	0	0	0	0	0	0.1666666667

Markov Chain

AUC Values using threshold of 0.3:

- 13 : 0.803355943468
- 14 : 0.776529124256
- 15 : 0.78091871177

Mean Value - 0.786934593165

Gradient Boosting Machine

```
> confusionMatrix(pred_13, test$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 523   0
         1 281 196

               Accuracy : 0.719
                 95% CI : (0.69, 0.7467)
    No Information Rate : 0.804
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4218
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6505
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.4109
             Prevalence : 0.8040
         Detection Rate : 0.5230
   Detection Prevalence : 0.5230
      Balanced Accuracy : 0.8252

       'Positive' Class : 0
```

```
> confusionMatrix(pred_14, test2$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 588   0
         1 275 137

               Accuracy : 0.725
                 95% CI : (0.6962, 0.7525)
    No Information Rate : 0.863
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3694
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6813
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.3325
             Prevalence : 0.8630
         Detection Rate : 0.5880
   Detection Prevalence : 0.5880
      Balanced Accuracy : 0.8407

       'Positive' Class : 0
```

```
> confusionMatrix(pred_15, test3$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 577   0
         1 276 147

               Accuracy : 0.724
                 95% CI : (0.6952, 0.7515)
    No Information Rate : 0.853
    P-Value [Acc > NIR] : 1

                  Kappa : 0.3807
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.6764
            Specificity : 1.0000
         Pos Pred Value : 1.0000
         Neg Pred Value : 0.3475
             Prevalence : 0.8530
         Detection Rate : 0.5770
   Detection Prevalence : 0.5770
      Balanced Accuracy : 0.8382

       'Positive' Class : 0
```

Random Forest

```
> confusionMatrix(RFpred_13, test$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 609   2
         1 195 194

               Accuracy : 0.803
                 95% CI : (0.777, 0.8272)
    No Information Rate : 0.804
    P-Value [Acc > NIR] : 0.5507

                  Kappa : 0.5445
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7575
            Specificity : 0.9898
         Pos Pred Value : 0.9967
         Neg Pred Value : 0.4987
             Prevalence : 0.8040
         Detection Rate : 0.6090
   Detection Prevalence : 0.6110
      Balanced Accuracy : 0.8736

       'Positive' Class : 0
```

```
> confusionMatrix(RFpred_14, test2$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 714   2
         1 149 135

               Accuracy : 0.849
                 95% CI : (0.8253, 0.8706)
    No Information Rate : 0.863
    P-Value [Acc > NIR] : 0.9075

                  Kappa : 0.56
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8273
            Specificity : 0.9854
         Pos Pred Value : 0.9972
         Neg Pred Value : 0.4754
             Prevalence : 0.8630
         Detection Rate : 0.7140
   Detection Prevalence : 0.7160
      Balanced Accuracy : 0.9064

       'Positive' Class : 0
```

```
> confusionMatrix(RFpred_15, test3$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 677   4
         1 176 143

               Accuracy : 0.82
                 95% CI : (0.7948, 0.8433)
    No Information Rate : 0.853
    P-Value [Acc > NIR] : 0.9982

                  Kappa : 0.5164
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7937
            Specificity : 0.9728
         Pos Pred Value : 0.9941
         Neg Pred Value : 0.4483
             Prevalence : 0.8530
         Detection Rate : 0.6770
   Detection Prevalence : 0.6810
      Balanced Accuracy : 0.8832

       'Positive' Class : 0
```

Logistic Regression

```
> confusionMatrix(pred_13, test$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 585   2
         1 219 194

               Accuracy : 0.779
                 95% CI : (0.752, 0.8044)
    No Information Rate : 0.804
    P-Value [Acc > NIR] : 0.9777

                  Kappa : 0.5057
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7276
            Specificity : 0.9898
         Pos Pred Value : 0.9966
         Neg Pred Value : 0.4697
             Prevalence : 0.8040
         Detection Rate : 0.5850
   Detection Prevalence : 0.5870
      Balanced Accuracy : 0.8587

       'Positive' Class : 0
```

```
> confusionMatrix(pred_14, test2$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 696   3
         1 167 134

               Accuracy : 0.83
                 95% CI : (0.8053, 0.8528)
    No Information Rate : 0.863
    P-Value [Acc > NIR] : 0.9986

                  Kappa : 0.5218
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.8065
            Specificity : 0.9781
         Pos Pred Value : 0.9957
         Neg Pred Value : 0.4452
             Prevalence : 0.8630
         Detection Rate : 0.6960
   Detection Prevalence : 0.6990
      Balanced Accuracy : 0.8923

       'Positive' Class : 0
```

```
> confusionMatrix(pred_15, test3$optimal)
Confusion Matrix and Statistics

          Reference
Prediction   0   1
         0 660   4
         1 193 143

               Accuracy : 0.803
                 95% CI : (0.777, 0.8272)
    No Information Rate : 0.853
    P-Value [Acc > NIR] : 1

                  Kappa : 0.4873
 Mcnemar's Test P-Value : <2e-16

            Sensitivity : 0.7737
            Specificity : 0.9728
         Pos Pred Value : 0.9940
         Neg Pred Value : 0.4256
             Prevalence : 0.8530
         Detection Rate : 0.6600
   Detection Prevalence : 0.6640
      Balanced Accuracy : 0.8733

       'Positive' Class : 0
```

Business Value - Solution

- Reduce CPU cycles for new path calculations
- Although not really user centric, this is a problem large corporations might face
- Data centers, server farms
- Any social network - be it Facebook, LinkedIn or Pinterest - would be interested in knowing the optimal path based on previous instances instead of calculating it at every instance

Thank You

References

- https://www.kaggle.com/c/facebook-ii
- Prediction and Smoothing for partially observed Markov Chains, Mats Rudemo.
- https://pypi.python.org/pypi/fingerprint/0.1.1
- http://networkx.github.io/documentation.html