Estimating the Expected Time of Arrival

Ahmad Khajehnejad
Summer 98
Basic Modules

- Map
- Directed Graph
- How to make:
  - Use third party’s map
  - Extract from a third party
  - Government - Satellite images
  - Use GPS data
  - Manual Edits
Basic Modules

- Map Matcher
Basic Modules

- Routing
- Scale (memory - computation)
- Edge types
- Middle points
Basic Modules

● Expected Time of Arrival (ETA)

● Challenge: few number of active users
● Solution: Asking third parties
  ○ 3k-4k queries / 15 minutes
  ○ 300k edges in Tehran
  ○ T0 and T
  ○ Prediction and Online
First Order Statistics

- 100k varying edges
- 50k edges varying more than 20%
- Edge types
- Varying edges’ lengths
How to overcome the problem?

- Find a proper request rate for each edge
- Find a proper request rate for each <edge, time>
How to overcome the problem?

- Find the state of the traffic
  - Learning an autoencoder
How to overcome the problem?

- Solve a set of regression problems:
  - Ask a set of large routes
  - Ask a subset of edges

ETA of a set of routes or edges \[\rightarrow\] Regression Model \[\rightarrow\] ETA of all edges
The Regression Problem

- Finding the correlations
  - Computational problem
  - Linear relations

- Using neural networks (easy to solve the optimization problem)
  - structure
  - Input edges
  - Loss function
## Neural Network Structure

<p>| | | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>12</td>
<td>13</td>
<td>14</td>
<td>15</td>
<td>16</td>
<td>17</td>
<td>18</td>
<td>19</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td>21</td>
<td>22</td>
<td>23</td>
<td>24</td>
<td>25</td>
<td>26</td>
<td>27</td>
<td>28</td>
<td>29</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>31</td>
<td>32</td>
<td>33</td>
<td>34</td>
<td>35</td>
<td>36</td>
<td>37</td>
<td>38</td>
<td>39</td>
<td>49</td>
<td></td>
</tr>
<tr>
<td>41</td>
<td>42</td>
<td>43</td>
<td>44</td>
<td>45</td>
<td>46</td>
<td>47</td>
<td>48</td>
<td>49</td>
<td>50</td>
<td></td>
</tr>
<tr>
<td>51</td>
<td>52</td>
<td>53</td>
<td>54</td>
<td>55</td>
<td>56</td>
<td>57</td>
<td>58</td>
<td>59</td>
<td>60</td>
<td></td>
</tr>
<tr>
<td>61</td>
<td>62</td>
<td>63</td>
<td>64</td>
<td>65</td>
<td>66</td>
<td>67</td>
<td>68</td>
<td>69</td>
<td>70</td>
<td></td>
</tr>
<tr>
<td>71</td>
<td>72</td>
<td>73</td>
<td>74</td>
<td>75</td>
<td>76</td>
<td>77</td>
<td>78</td>
<td>79</td>
<td>80</td>
<td></td>
</tr>
<tr>
<td>81</td>
<td>82</td>
<td>83</td>
<td>84</td>
<td>85</td>
<td>86</td>
<td>87</td>
<td>88</td>
<td>89</td>
<td>90</td>
<td></td>
</tr>
<tr>
<td>91</td>
<td>92</td>
<td>93</td>
<td>94</td>
<td>95</td>
<td>96</td>
<td>97</td>
<td>98</td>
<td>99</td>
<td>100</td>
<td></td>
</tr>
</tbody>
</table>
Neural Network Structure

Input

1, 2
1, 11
2, 3
2, 12
99, 100

20 Neurons
20 Neurons
20 Neurons
20 Neurons
20 Neurons

Output

1
2
3
4
100

4K
100K
Input Edges

\[
\text{Edge-length} \geq 1000 \\
\text{or} \\
\text{Edge-length} \geq 120 \quad \& \quad \frac{\text{max-eta}}{\text{min-eta}} \geq 10
\]

\( \cup \)

1000 most frequent edges
Loss function

\[ \sum_i (T_{pred} - T_{true})^2 \times T_0^{(i)} \times freq^{(i)} \]

\[ \frac{\sum_i T_0^{(i)} \times freq^{(i)}}{\sum_i T_0^{(i)} \times freq^{(i)}} \]
Results

80% of <edge-time> pairs ---> error <= 20%

70% of times ---> worst error <= 30%

Q: What KPI to use?
Using GPS data

- GPS fields
  - Edge_id
  - Fraction
  - Fractoin_delta
  - Accuracy
  - Ip
  - Latitude
  - Longitude
  - Provider_is_gps
  - Speed
  - Time
  - Uid
  - act_VEHICLE
  - act_BIKE
  - act_WALKRUN
  - act_STILL
  - act_UNKNOWN
  - act_TILTING
  - act_WALKING
  - act_RUNNING
Dirty data Challenges

- Duplicate data
- Vehicle action is not valid
- Zero speed for moving vehicles
- Incompatibility in time zone
- Time in future
How much data we have

- 30k daily active users
Available data for each edge

- Sessions
- Tracks
- Update the estimations every 5 minutes
- Store the data in Reddis
Weighted Average

Average on duration:

\[
\frac{\sum_i w(i) D(i)}{\sum_i w(i)}
\]

Average on speed:

\[
\frac{\sum_i w(i) V(i)}{\sum_i w(i)}
\]
Weighted Average

Average over duration: \[ \bar{D} = \frac{\beta e^{-\lambda \Delta_0} T_0 + \alpha \sum_{i=1}^{n} e^{-\lambda \Delta^{(i)}} \frac{D(i)}{\text{frac}(i)}}{\beta e^{-\lambda \Delta_0} + \alpha \sum_{i=1}^{n} e^{-\lambda \Delta^{(i)}} \frac{\text{frac}(i)}} \]

Average over speed: \[ \bar{V} = \frac{\beta e^{-\lambda \Delta_0} V_{T_0} + \alpha \sum_{i=1}^{n} e^{-\lambda \Delta^{(i)}} \frac{V(i)}{\text{frac}(i)}}{\beta e^{-\lambda \Delta_0} + \alpha \sum_{i=1}^{n} e^{-\lambda \Delta^{(i)}} \frac{\text{frac}(i)}} \]

\[ \Delta^{(i)} = \frac{\text{Now} - \text{Arrival time of the } i^{th} \text{ track}}{\text{time Quantum}} \]
Avg Duration vs Avg Velocity

Edge length: 1000  V1: 100  V2: 10

Avg D = (10+100)/2 = 55

Avg V = (100+10)/2 = 55 => estimated D = 1000/55 = 18.18

Avg Duration: sensitive to the fast tracks

Avg Velocity: sensitive to the slow tracks
T0 and Profile

Profile: Adaptive with time, instead of T0
Low Memory Implementation

\[
\bar{D}_n = \frac{\beta e^{-\lambda \Delta_0} T_0 + \alpha \sum_i^n e^{-\lambda \Delta^{(i)}} \frac{\lambda}{\alpha} D^{(i)}}{\beta e^{-\lambda \Delta_0} + \alpha \sum_i^n e^{-\lambda \Delta^{(i)}} \frac{\lambda}{\alpha} D^{(i)}}
\]

\[
A_n = \sum_i^n e^{-\lambda \Delta^{(i)}} \frac{\lambda}{\alpha} D^{(i)}
\]

\[
B_n = \sum_i^n e^{-\lambda \Delta^{(i)}} \frac{\lambda}{\alpha} D^{(i)}
\]

\[
\bar{D}_n = \frac{\beta e^{-\lambda \Delta_0} T_0 + \alpha A_n}{\beta e^{-\lambda \Delta_0} + \alpha B_n}
\]

\[
A_{n+1} = e^{-\lambda \Delta^{(n+1)}} \frac{\lambda}{\alpha} D^{(n+1)} + e^{-\lambda \tilde{\Delta}} A_n
\]

\[
B_{n+1} = e^{-\lambda \Delta^{(n+1)}} \frac{\lambda}{\alpha} D^{(n+1)} + e^{-\lambda \tilde{\Delta}} B_n
\]

\[\tilde{\Delta} : \text{Passed time from last update}\]
Memory less model vs lazy computation

- Other types of coefficients
- The problem of congestion

- Using a min Heap (on occurrence times of the tracks)
- Bounding a maximum sum of the tracks’ fractions
Weighted Average

Regression point of view:

- Computationally inefficient
- Hard to implement

Segment-based estimation
Confidence

\[ C = \sum_{i}^{n} e^{-\Theta \Delta^{(i)}} frac{a}{c^{(i)}} \]

Select 1000 most confident edges
Predicting Unconfident Edges

How should the mask be selected? Set the masked inputs to 0 or T0? Pass the confidence into the input layer?
Confident data from third parties

\[ \bar{D}_n = \frac{\beta e^{-\lambda \Delta_0} T_0 + \alpha \sum_{i}^{n} e^{-\lambda \Delta(i)} \text{frac}(i) D(i)}{\beta e^{-\lambda \Delta_0} + \alpha \sum_{i}^{n} e^{-\lambda \Delta(i)} \text{frac}(i)} \]

\[ \Delta(i) = \frac{\text{Now} - \text{Arrival time of the } i^{th} \text{ track}}{\text{time Quantum}} \]

Step 1:
- Receiving limited data (every 15 minutes)
- From heavy traffics
- On a different non-matched map

\[ \lambda \uparrow \text{ time Quantum} = 15 \times 60 \]

Step 2:
- More clean data
Future: How to use ML more?

● Detecting bad patterns

● End-to-end ETA estimator for one edge

● End-to-end ETA estimator for all the edges

Dilemma: Using complex models or domain knowledge?
Future

- Probabilistic estimation
- Learning new profiles
- Traffic prediction
- Routing based on dynamic ETA’s
THANK YOU