Appying Supervised Learning

To Improve Url Ordering In Autocomplete

Softmax Learning Model

- Probability that url i gets selected from N urls in the autocomplete list:

$$P(y = i;\theta) = \frac{\exp(\theta^T \mathbf{x}^{(i)})}{\sum_{j=1}^{N} \exp(\theta^T \mathbf{x}^{(j)})}$$

- Use Newton's method to find parameters that maximize log likelihood:

$$l(\theta) = \log \prod_{i=1}^{m} P(y = y^{(i)};\theta)$$

Examples of Features Used

- Does the url contain a trailing slash?
- Frequency
- Recency
- Was the url typed by the user?
- Size of the url
- Number of ‘/’s in the url
- …
- (51 features in all)

First try – Mozilla beats Softmax

- Score on an autocomplete event = 1/i, where i is the position of the url selected by the user after the list is sorted by the Mozilla/Softmax algorithm
- Mozilla scored 3609.23 out of 4343
- Softmax scored 3460.85 out of 4343
- We needed to try something new!

New approach

- Simulate history and generate autocomplete list parameterized on the number of characters typed by the user.
- In other words, see if softmax does better url ordering when fewer characters are typed.
- It did!

But, …

- From a user's perspective, it's the number of keys typed to get to the next url that is the important metric.
- We decided to measure that more directly.
- We simulated typing a url character by character and calculated the minimum number of keys needed by Softmax and Mozilla to get to the user's url.
- **Result: Softmax saved 0.89 keys per autocomplete event over Mozilla.**

Online learning

- Till now, we'd kept our parameter vector fixed.
- We now tried to improve the vector on each autocomplete event by taking a stochastic gradient ascent step.
- We got a marginally better result: **Softmax saved 0.94 characters per autocomplete event over Mozilla** using a learning rate of 0.0001 and a heuristic that used a higher learning rate at events where the user selected url was high on the list.