Algorithmic Fairness

Andres Aradillas Fernandez

22 October 2022

The following is a detailed description of the algorithm presented in "Policy Learning for Fairness in Rankings" (Singh and Joachims 2019).

Definition of Class of Fair Ranking Functions

Link to paper: `https://www.cs.cornell.edu/people/tj/publications/singh_joachims_19a.pdf`.

The objective of a fair ranking policy maximizing utility would be the following:

$$\pi_\delta^\star = \text{argmax}_\pi \mathbb{E}_{q\sim\mathcal{Q}}[U(\pi|q)] \text{ s.t. } \mathbb{E}_{q\sim\mathcal{Q}}[\mathcal{D}(\pi|q)] \leq \delta$$

where $\mathcal{Q}$ is the distribution from which queries q are drawn, π refers to a specific stochastic ranking function, and then $\pi(r|q)$ is the distribution over rankings r given query q. $\mathcal{D}(\pi|q) \geq 0$ is then a disparity measure that must be below some bound δ. $U(\pi|q)$ refers to utility that is defined as:

$$U(\pi|q) = \mathbb{E}_{r\sim\pi(r|q)}[\Delta(r, rel^q)]$$

where $\Delta(r, rel^q)$ is the ranking metric based on the relevance factors rel^q of a given query q, as well as the rankings r.

This objective, therefore, defines the entire class of fair ranking policies. We can then observe the specific Fair-PG-Rank algorithm later presented in the paper, but first a sub-algorithm used in Fair-PG-Rank: Plackett-Luce Ranking Policies.

Plackett-Luce Ranking Model

Ranking policies π are made up of (1) a scoring model defining ranking distributions and (2) a sampling method. Starting with scoring model h_θ (θ refers to the given parameters of the model) and given input features $\mathbf{x}^q$ for a query q, the scoring model then assigns scores $h_\theta(\mathbf{x}^q) = (h_\theta(x_1^q), h_\theta(x_2^q), ..., h_\theta(x_{n_q}^q))$ (note: n_q is the number of results in a query q) in the form of a vector. Then, this score vector is used to find the probability $\pi_\theta(r|q)$ of a particular ranking $r =< r(1), ..., r(n_q) >$ is:

$$\pi_\theta(r|q) = \prod_{i=1}^{n_q} \frac{exp(h_\theta(x_{r(i)}^q))}{exp(h_\theta(x_{r(i)}^q)) + ... + exp(h_\theta(x_{r(n_q)}^q))}$$

Per the paper, this probability is computationally efficient, and so is the associated sampling method. The following R package allows for all this to be implemented: `https://cran.r-project.org/web/packages/PlackettLuce/PlackettLuce.pdf`.

Fair-PG-Rank Algorithm

The algorithm, taken directly from the paper, is as follows:

Input: $\mathcal{T} = \{(\mathbf{x}^q, rel^q)\}_{i=1}^N$ (training set), disparity measure $\mathcal{D}$, utility/fairness trade-off λ.

Parameters: model h_θ, learning rate η, entropy regulation γ

Initialize h_θ with parameters θ_0

repeat until convergence on validation set:

$q = (\mathbf{x}^q, rel^q) \sim \mathcal{T}$ (draw query from training set)

$h_\theta(\mathbf{x}^q)$ (get scores by applying scoring model)

Do Plackett-Luce sampling for r_i from 1 to S (S is desired number of samples)

Compute gradient average (see paper for details on this)

Update parameters to account for new gradient. **End of algorithm.**

All of the code for this that is used in the paper, which analyzes both simulated and real data, is housed in the following GitHub repository: `https://github.com/ashudeep/Fair-PGRank`.

References

The contents of these notes are derived entirely from "Policy Learning for Fairness in Rankings" (2019) by Ashudeep Singh and Thorsten Joachims.

Proposed Model

Suppose we want to rank a set of job postings with different salaries ($salary_i$) for a particular user that has a previously-calculated employability score $empscore$. The location difference $locdiff_i$ of each job posting can be given punishment weight of w'_i and $d_g(*)$ represents a fairness constraint on every ranking r.

$$min_r \sum_{i=1_r}^{n_r} \left| norm_{max}\left(w_{sum}\left(rank_i, \frac{salary_i}{meansalary_i}\right)\right) - empscore \right| + w'_i locdiff_i \text{ s.t. } d_g(r) \leq \delta$$