NYPC Code Battle 퍼포먼스 시스템

NEXON Korea

July 2025

1 퍼포먼스 모델

대부분의 레이팅 시스템은 실력의 변동을 가정하지만 제출된 하나의 코드에 대한 실력의 변동은 없다고 가정하는게 더 맞습니다. 이를 위해서 Code Battle의 퍼포먼스 시스템은 변형된 Bradley-Terry Model을 사용합니다. Bradley-Terry Model은 각자에게 고정된 퍼포먼스 π_i를 할당하고, i가 j를 이길 확률 $\Pr(i > j)$를 다음과 같이 가정합니다. [2]

$$\Pr(i > j) = \frac{\pi_i}{\pi_i + \pi_j}$$

1.1 샘플 AI와의 퍼포먼스

샘플 AI와의 퍼포먼스는 참가자의 대략적인 퍼포먼스 수준을 계산하기 위해 도입했습니다.

샘플 AI와 승리 횟수를 w, 패배 횟수를 l이라고 할 때, 승리 확률을 $\frac{2w+1}{2w+2l+2}$으로 추정합니다. 여기서 샘플 AI의 퍼포먼스를 1이라고 하면 유저의 퍼포먼스는 $\pi_i = \frac{2w+1}{2l+1}$이 됩니다.

1.2 중간평가의 퍼포먼스

우리는 한 가지의 가정 $\log(\pi_i) \sim Z$를 더 추가해서, 참가자들의 실력을 정규분포에 맞게 배치하려고 합니다. 여기서, 샘플 AI의 퍼포먼스는 1로 고정합니다.

w_{ij}를 i가 j를 이긴 횟수라고 할때, log-likelihood는 다음과 같습니다.

$$\begin{aligned} l(\vec{\pi}) &= \log\left(\prod_{i,j}\left[\Pr(i>j)\right]^{w_{ij}} \times \prod_i \exp\left(-\frac{(\log\pi_i)^2}{2}\right)\right) \\ &= \sum_{i,j} w_{ij}\log\left(\frac{\pi_i}{\pi_i+\pi_j}\right) + \sum_i\left(-\frac{(\log\pi_i)^2}{2}\right) \\ &= \sum_{i,j} w_{ij}\left(\log(\pi_i) - \log(\pi_i+\pi_j)\right) - \sum_i \frac{\log(\pi_i)^2}{2} \end{aligned}$$

이 $l(\vec{\pi})$의 maximum을 구하기 위해서 π_i에 대해서 미분을 한 후, [1]의 방법을 이용하여 식을 정리합니다.

$$\begin{aligned}\frac{\partial l(\vec{\pi})}{\partial \pi_i} &= \sum_j \left(\frac{w_{ij}}{\pi_i} - \frac{w_{ij}+w_{ji}}{\pi_i+\pi_j} \right) - \frac{\log \pi_i}{\pi_i} \\ &= \sum_j \left(\frac{\pi_j w_{ij}}{\pi_i(\pi_i+\pi_j)} - \frac{w_{ji}}{\pi_i+\pi_j} \right) - \frac{\log \pi_i}{\pi_i} \\ &= \frac{1}{\pi_i} \left(\sum_j \frac{\pi_j w_{ij}}{\pi_i+\pi_j} - \log \pi_i \right) - \sum_j \frac{w_{ji}}{\pi_i+\pi_j} = 0\end{aligned}$$

이 해를 직접 구하는 것은 어려운 일이므로, 적당한 방법으로 해를 근사해야합니다.

1.2.1 계산

이제, 양변에 π_i를 곱한 이후 이항을 해 $\log \pi_i$에 대해 정리하면

$$\log \pi_i = \sum_j \frac{\pi_j w_{ij} - \pi_i w_{ji}}{\pi_i + \pi_j}$$

가 됩니다.

$\beta_i = \log \pi_i$로 정의합니다. β_i에 대해 식을 정리하면

$$\beta_i = \sum_j \frac{\left(e^{\beta_j - \beta_i} \cdot w_{ij}\right) - w_{ji}}{1 + e^{\beta_j - \beta_i}}$$

가 됩니다. $\beta_{j \neq i}$가 고정되었을 때, β_i는 다음 증가함수 f 의 유일한 근입니다.

$$\begin{aligned}g(\lambda) &= \sum_j \frac{\left(e^{\beta_j - \lambda} \cdot w_{ij}\right) - w_{ji}}{1 + e^{\beta_j - \lambda}} \\ f(\lambda) &= \lambda - g(\lambda)\end{aligned}$$

이 해를 뉴턴-랩슨법으로 찾기 위해 $g'(\lambda)$를 계산합시다. $g(\lambda)$가 $e^{\beta_j - \lambda}$에 관한 함수의 합이기 때문에, $x_j = e^{\beta_j - \lambda}$라고 놓으면

$$\begin{aligned}\frac{dg}{d\lambda} &= \sum_j \left[\frac{\partial g}{\partial x_j} \frac{\partial x_j}{\partial \lambda} \right] \\ &= \sum_j \left[\frac{\partial}{\partial x_j} \left(\frac{x_j w_{ij} - w_{ji}}{1 + x_j} \right) \cdot -x_j \right] \\ &= -\sum_j \left[\frac{(w_{ij} + w_{ji}) x_j}{(1+x_j)^2} \right]\end{aligned}$$

이고

$$\frac{df}{d\lambda} = 1 + \sum_j \left[\frac{e^{\beta_j - \lambda} \cdot (w_{ij} + w_{ji})}{(1 + e^{\beta_j - \lambda})^2} \right]$$

입니다.

이제 새로운 β_i'를 $\beta_i - \frac{f(\beta_i)}{f'(\beta_i)}$로 대체하는 것을 원하는 정밀도가 될 때까지 반복합니다.

2 퍼포먼스 표시

계산이 끝난 이후 β_i의 값에 대해 $p_i = 1500 + 400\beta_i$를 계산하고, P_i를 50의 배수 (샘플 AI와의 퍼포먼스) 혹은 10의 배수 (중간평가의 퍼포먼스) 단위로 보여줍니다.

$$P_i = \begin{cases} 300 \times \exp((p_i - 300)/400) & \text{if } p_i < 300 \\ p_i & \text{if } 300 \leq p_i \leq 2700 \\ 2700 + 400 \log(1 + (p_i - 2700)/400) & \text{if } p_i > 2700 \end{cases}$$

3 구현

사용할 수 있는 Rust 및 node.js 라이브러리 형태로 `https://github.com/nypc/nypc-perf`에 구현되어있습니다.

References

[1] M. E. J. Newman. "Efficient Computation of Rankings from Pairwise Comparisons". In: *Journal of Machine Learning Research* 24.238 (2023), pp. 1–25. URL: `http://jmlr.org/papers/v24/22-1086.html`.

[2] Ernst Zermelo. "Die Berechnung der Turnier-Ergebnisse als ein Maximumproblem der Wahrscheinlichkeitsrechnung". In: *Mathematische Zeitschrift* 29.1 (1929), pp. 436–460. DOI: `10.1007/BF01180541`.