Ideal Timing for a Starting Pitcher Change

Depending on the latent class for each situation

2024/06/07

YSAL 4기 TEAM BASEBALL
김지원, 송화윤, 이주원, 이환욱, 전희연, 정현지

Contents.

01.
Introduction

Work Cited.

“A Prediction Model of Runs Allowed Based on Latent-Class Markov Chain for Starters of Professional Baseball Pitchers”

1. Consider pitcher’s ability & the condition of the game day & batting order

2. EM algorithm을 사용하여 상황 별 잠재 클래스 할당

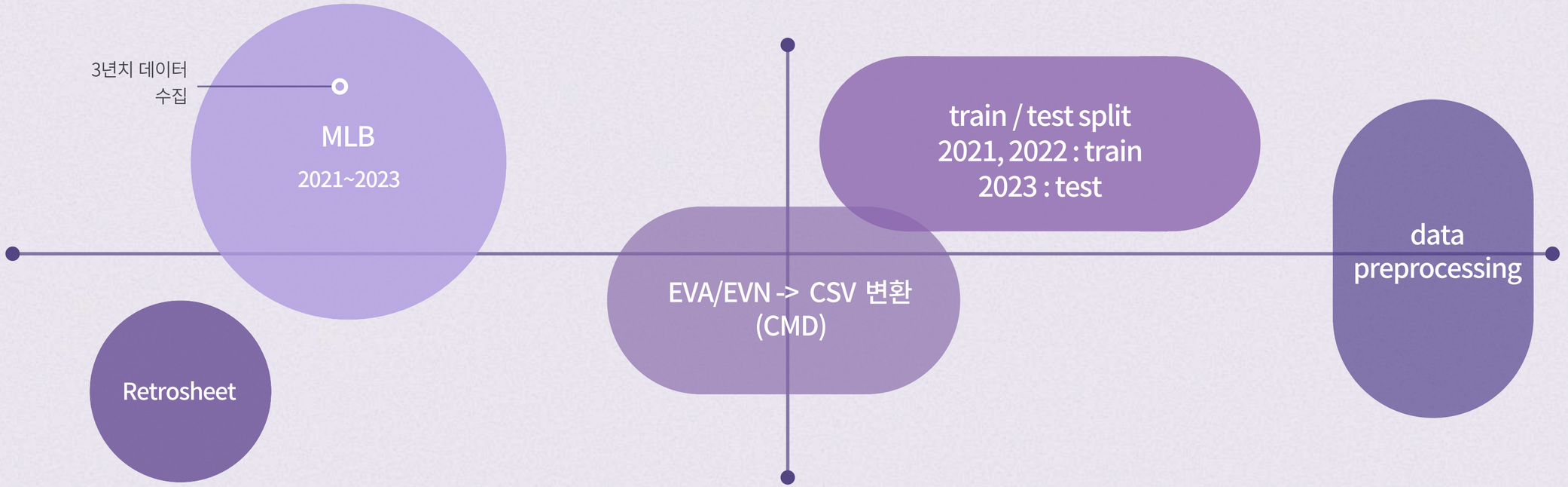
3. 잠재 클래스 별로 각각의 transition matrix 생성

4. 이닝별 기대 득점 계산

5. 선발 투수 교체 타이밍 제안

02.
Process

Data.


3년치 데이터
수집
MLB
2021~2023
Retrosheet
EVA/EVN -> CSV 변환
(CMD)
train / test split
2021, 2022 : train
2023 : test
data
preprocessing

전처리

Data : Player's ability.

투수의 능력
Pitcher's ability

직전 게임까지의 누적

WHIP
ERA
GO/FO%
K/9

투수의 컨디션
Pitcher's condition

바로 직전 이닝의

WHIP
ERA
GO/FO%
K/9

1이닝은 모든 변수를 0으로 설정

타자의 능력
batter's ability

현재 상황의

상대 타자의 타순

- WHIP (walks and hits per play) : 이닝당 출루허용률
- ERA (earned run average) : 평균자책점, 9이닝당 실점율
- GO/FO% (Ground out/ fly out ratio) : 땅볼 아웃 / 플라이 아웃 비율
- K/9 : 9이닝당 탈삼진 수

전처리

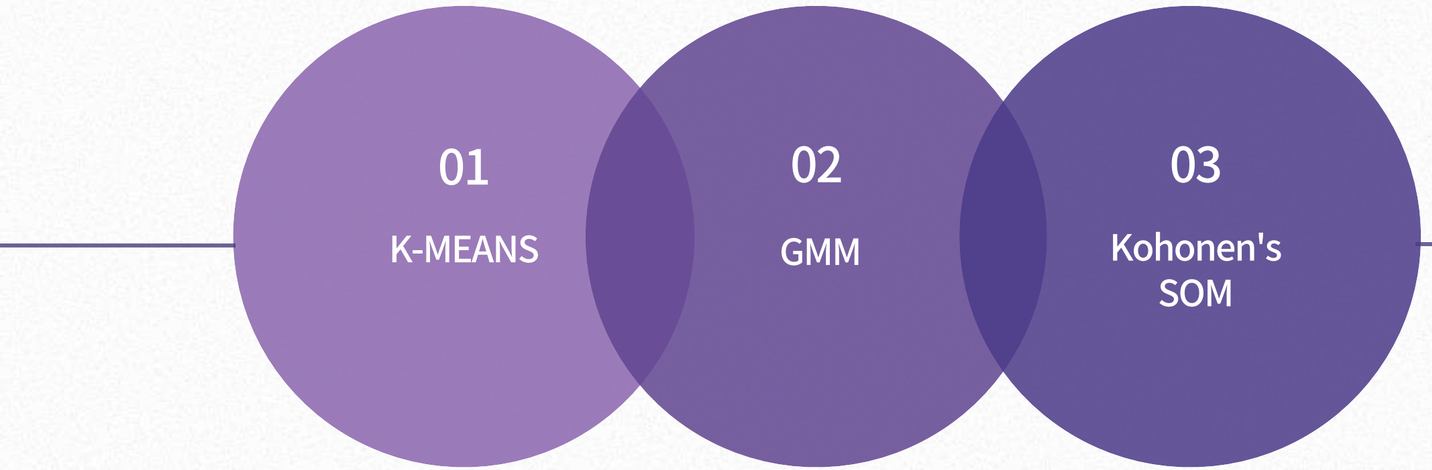
Data : State.

- 총 25개의 state존재
- transition matrix : 특정 state에서 다른 state로 갈 확률을 Matrix로 나타낸 것
- 특정 state를 시작점 지정한 뒤 simulation을 계속 반복하여 기대 득점을 구할 수 있음

CLUSTERING.

EM algorithm에 넣기 전, 직전 이닝의 지표들을[[WHIP, ERA, GO/FO%, K/9]]이용하여 투수의 컨디션을 4가지 index로
누적 [[WHIP, ERA, GO/FO%, K/9]]을 이용하여 투수의 능력을 4가지 index로 클러스터링

trial.

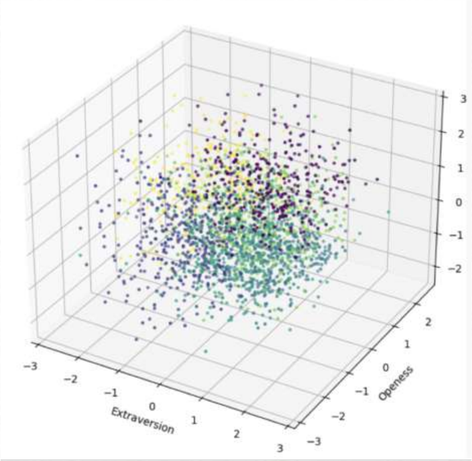


final.

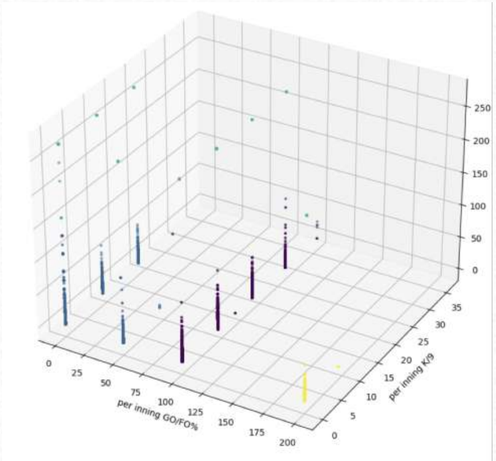
CLUSTERING.

K-Means / GMM

expected...

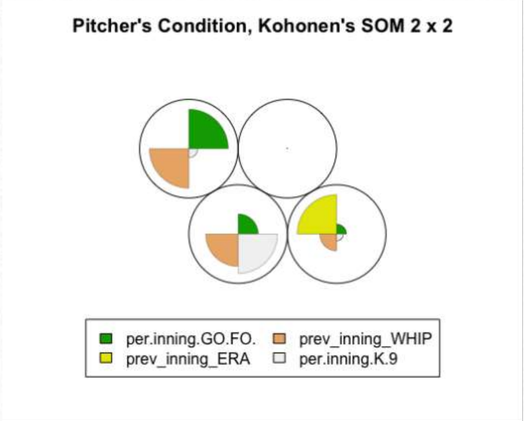


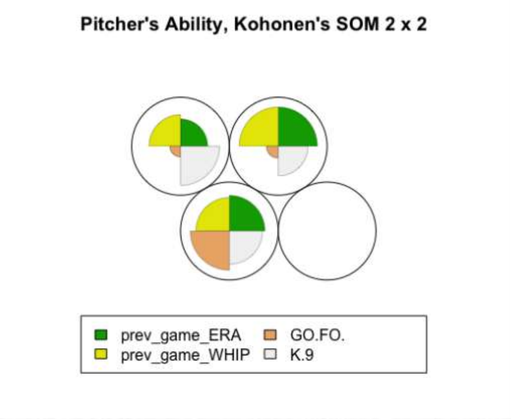
but....



1. 변수 네 개를 한 번에 설명하기 어려움.
2. pitcher's condition에 대한 클러스터링이 잘 수행되지 않음

KOHONEN'S SOM





1. 연산속도가 빠름
2. 변수의 개수가 많더라도 직관적으로 시각화 가능

EM algorithm.

: expectation–maximization algorithm

Definition

관측되지 않는 잠재변수에 의존하는 확률 모델에서 최대가능도(maximum likelihood)나 최대사후확률(maximum a posteriori, MAP)을 갖는 모수의 추정값을 찾는 반복적인 알고리즘

E - Step

Likelihood의 expectation을 계산

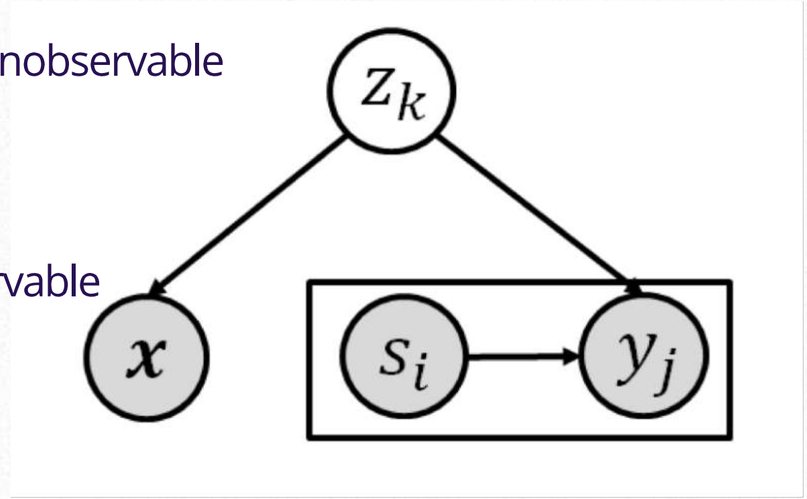
M - Step

Likelihood의 expectation을 최대화하는 새로운 모수 제안

EM algorithm.

: expectation–maximization algorithm

In this case..



- Z : 모든 상황을 담은 Latent Class
- S : 타자의 타석 전 State (현재 상황)
- Y : 타자의 타석 후 State (미래 상황)
- X : Row별로 투수의 능력, 당일 선수 컨디션, 타순에 대한 정보를 담은 모든 Row

E-step

Posterior : $$p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)=\frac{p(\boldsymbol{x}|z_k)p\left(y_j\middle|s_i,z_k\right)p(z_k)}{\sum_{k=1}^{K}p(\boldsymbol{x}|z_k)p\left(y_j\middle|s_i,z_k\right)p(z_k)}. \quad (14)$$

M-step

Prior : $$p(z_k)=\frac{\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{m_1=1}^{M_1}\cdots\sum_{m_L=1}^{M_L}p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)}{\sum_{k=1}^{K}\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{m_1=1}^{M_1}\cdots\sum_{m_L=1}^{M_L}p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)}. \quad (15)$$

Emission1 : $$p\left(x_{m_l}^{l}|z_k\right)=\frac{\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{m_1=1}^{M_1}\cdots\sum_{m_{l-1}}^{M_{l-1}}\sum_{m_{l+1}}^{M_{l+1}}\cdots\sum_{m_L=1}^{M_L}p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)}{\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{m_1=1}^{M_1}\cdots\sum_{m_L=1}^{M_L}p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)}. \quad (16)$$

Emission2 : $$p\left(y_j|s_i,z_k\right)=\frac{\sum_{m_1=1}^{M_1}\cdots\sum_{m_L=1}^{M_L}p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)}{\sum_{j=1}^{J}\sum_{i=1}^{I}\sum_{m_1=1}^{M_1}\cdots\sum_{m_L=1}^{M_L}p\left(z_k\middle|\boldsymbol{x},y_j,s_i\right)}. \quad (17)$$

The EM algorithm iterates the E-step and M-step until the log-likelihood converges.

EM algorithm.

: expectation–maximization algorithm

k = 6으로 설정

1. Multivariate Multinomial Mixture Model 가정

2. Assign random values

- prior : p(z_k)

summation : 1

p(z_0) p(z_1) p(z_2) p(z_3) p(z_4) p(z_5) 6 X 1

- emssion1 : p(x | z_k)

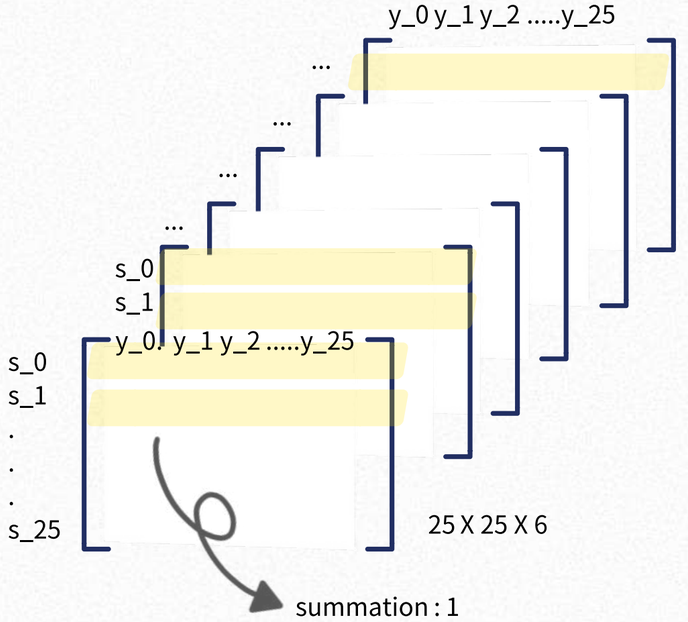
p (x = 0| z_0) p (x = 0 | z_1) .....
p (x = 1| z_0)
.
.
p (x = 143| z_0)

144 X 6

summation : 1

* 가능한 x의 값은 총 144개 (4 X 4 X 9)

- emission2 : p(y_j | s_i , z_k)



3. Compute posterior (앞 페이지의 수식에 따라)
4. Compute the log likelihood of the posterior (len(df) X 25 X 25 X 6)
5. Update prior / emissions
6. keep updating while new likelihood is less than old likelihood
7. train완료한 3개의 matrix를 .npz 파일로 저장
8. 새로운 data의 x,s,y를 넣어 (6,1)의 posterior probs를 얻음

EM algorithm.

: expectation–maximization algorithm

E - step

```
numerator1 = np.multiply(self.emissions2, np.reshape(self.prior, (1, self.C))) # (25,25,6)
numerator2 = self.emissions1[dataset["x"],:] #(len(train_),6)
numerator2_ = numerator2[:, np.newaxis, np.newaxis, :]

numerator = np.multiply(numerator2_, numerator1)

denominator = np.sum(numerator, axis=3)
denominator_ = np.expand_dims(denominator, axis=3)
posterior_estimate = np.divide(numerator, denominator_) # (len(train),25,25,6)

likelihood = np.sum(np.log(numerator))

likelihood_list.append(likelihood)
print("Mixture model training: epoch ", current_epoch, ",  likelihood = ", likelihood)

delta = likelihood - old_likelihood
old_likelihood = likelihood
```

M - step

```
numerator = self.smoothing + np.sum(np.sum(np.sum(posterior_estimate, axis=0),axis=0),axis=0)
denominator = self.smoothing * self.C + np.sum(posterior_estimate)
self.prior = np.divide(numerator, denominator)
numerator2 = self.smoothing + np.zeros((self.ks[0], self.C))
np.add.at(numerator2, dataset["x"], np.sum(np.sum(posterior_estimate,axis=1),axis=1))

numerator1 = np.sum(posterior_estimate, axis=0)

denominator2 = self.smoothing * self.ks[0] + np.sum(np.sum(np.sum(posterior_estimate,axis=0),axis=0),axis=0)
denominator1 = self.smoothing * self.ks[1] + np.sum(np.sum(np.sum(posterior_estimate,axis=0),axis=0),axis=0)

self.emissions1 = np.divide(numerator2, np.reshape(denominator2, (1, self.C)))
self.emissions2 = np.divide(numerator1, denominator1)
```

```
if likelihood > max_likelihood :

  np.savez(path + '/em4.npz', array1=self.prior, array2= self.emissions1, array3=self.emissions2)

  max_likelihood = likelihood

current_epoch += 1
```

Predict

```
predict(0,0,0)
```

```
array([0.19437797, 0.07252927, 0.14780596, 0.38548487, 0.16664072,
       0.0331612 ])
```

>> 해당 sequence에서의 상황을 형식을 맞춰 x,s,y로 넣어주면
그 상황이 어떤 class에 속하는 지 확률값을 반환해줌

EM clustering.

: 6개의 cluster (z0 ~ z5)로 타격 상황 및 투수 특징 분석

Class	Z1	Z2	Z5
Pitcher's ability	Good WHIP,ERA High FO ratio Bad K/9	Good WHIP,ERA High FO ratio Bad K/9	Bad WHIP,ERA High FO ratio Good K/9
Pitcher's condition	Bad WHIP Good K/9 Normal GO/FO ratio	Bad WHIP, K/9 High FO ratio Good ERA	Bad WHIP Good K/9 Normal GO/FO ratio
Batting order	4th	1st	2nd
Example	Darin Ruf vs Kim.K	Acuna Jr. vs Will Crowe	Juan Soto vs Tayler Widener

Expected Runs.

transition matrix

$$P = \begin{pmatrix} A & B & C_0 & d_0 \\ 0 & A & B & e_1 \\ 0 & 0 & A & f \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

특정 Base occupation과 out 상황에서 다른 State로의 변환 확률을 나타내는 (25,25) Matrix

$$A = \begin{pmatrix} p_{HR} & p_S + p_W & p_D & p_T & 0 & 0 & 0 & 0 \\ p_{HR} & 0 & 0 & p_T & p_S + p_W & 0 & p_D & 0 \\ p_{HR} & p_S & p_D & p_T & p_W & 0 & 0 & 0 \\ p_{HR} & p_S & p_D & p_T & 0 & p_W & 0 & 0 \\ p_{HR} & 0 & 0 & p_T & p_S & 0 & p_D & p_W \\ p_{HR} & 0 & 0 & p_T & p_S & 0 & p_D & p_W \\ p_{HR} & p_S & p_D & p_T & 0 & 0 & 0 & p_W \\ p_{HR} & 0 & 0 & p_T & p_S & 0 & p_D & p_W \end{pmatrix}$$

A : no out-count increase

$$B = p_0 I,$$

B: out-count after at bat
C: 0 to 2 outs (double play)
d: 0 to 3 outs (triple play)
e: double play from 1 out

$$f = (p_0, \ldots, p_0)^{\mathrm{T}}.$$

f : 2 to 3 outs

Real Matrix

	000 0	000 1	000 2	001 0	001 1
000 0	0.0351985151	0.6808592512	0	0.004142333288	0
000 1	0	0.03187399231	0.6910889247	0	0.003596676175
000 2	0	0	0.03216363006	0	0
001 0	0.03378378378	0.1936936937	0.004504504505	0.009009009009	0.4189189189
001 1	0	0.02884615385	0.2025641026	0	0.005128205128
001 2	0	0	0.03194390339	0	0
010 0	0.02584954981	0.001742666279	0.004647110078	0.005227998838	0.2065059541
010 1	0	0.02980265807	0.003221908981	0	0.005034232783

Expected Runs.

algorithm

$$\boldsymbol{U}_{n|i} = \sum_{r=0}^{4} \boldsymbol{U}_{n-1|i-r}\boldsymbol{P}^{(r)} .$$

$$(i = 1, 2, \ldots, R_{\max} + 1), i > r$$

z0~z5에 대한 matrix P

$$ER = \sum_{i=1}^{R_{\max}+1} u_i \times (i-1) .$$

Expected Runs.

compute U

- U_0

0000 1000 ...

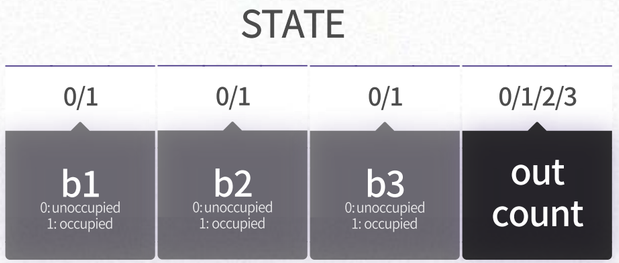
$$\begin{matrix} 0 \\ 1 \\ . \\ . \end{matrix} \begin{bmatrix} 1 & 0 & \dots \\ 0 & 0 & \dots \\ . & . & \\ . & . & \end{bmatrix}$$

- Un

STATE 0000 1000 ...

R (runs)

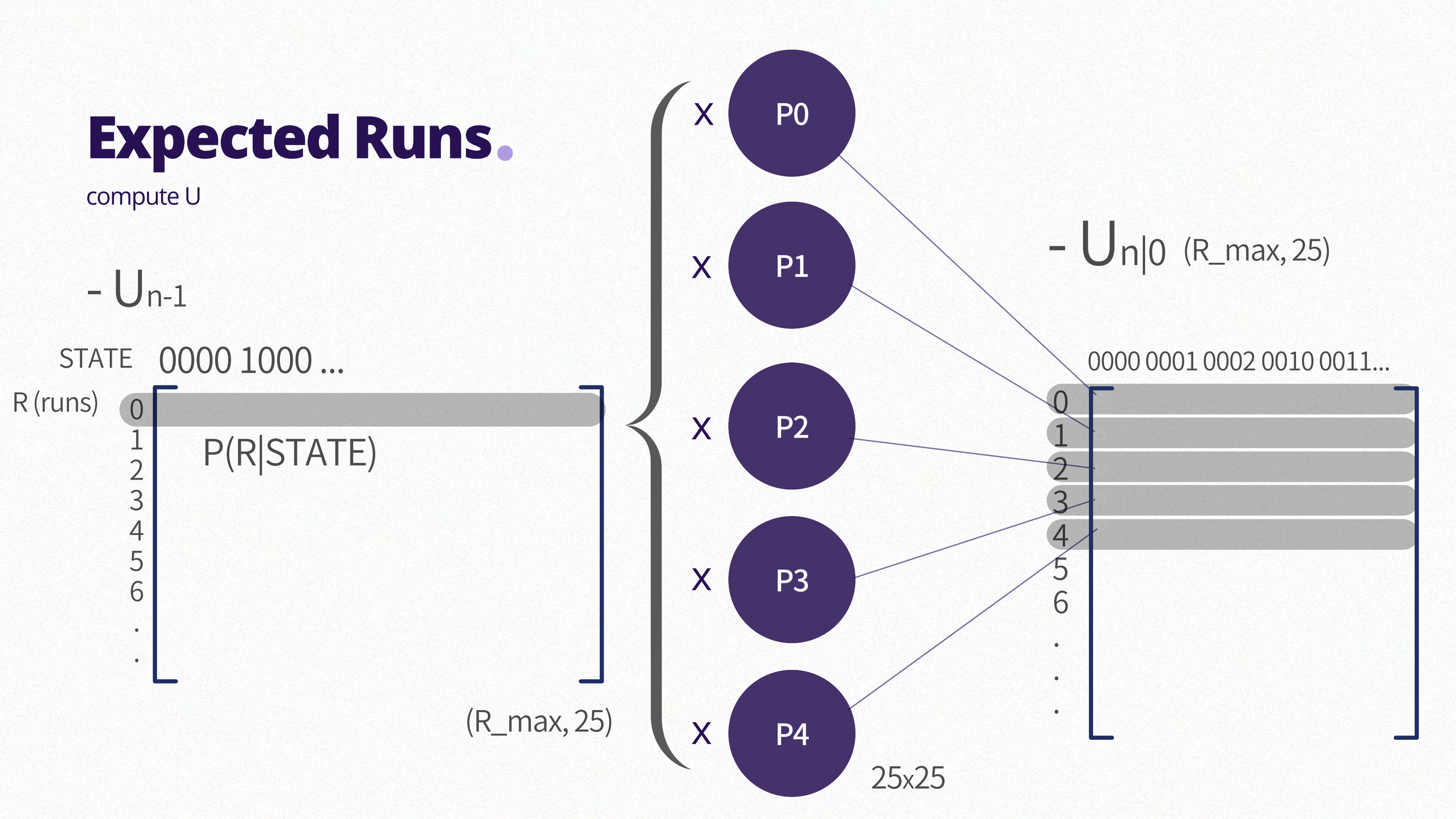
$$\begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \\ 6 \\ . \\ . \end{matrix} \begin{bmatrix} P(R|STATE) \\ \\ \\ \\ \\ \\ \\ \\ \end{bmatrix} (R\_max, 25)$$



Expected Runs.

compute U

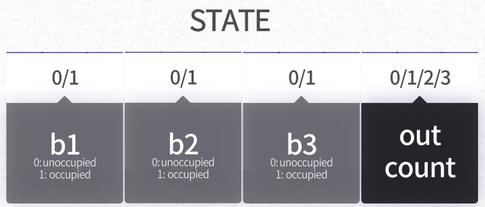
- U_{n-1}



Expected Runs.

calculation

- Un



STATE 0000 1000 ...

R (runs)

0
1
2
3
4
5
6
.
.

P(R|STATE)

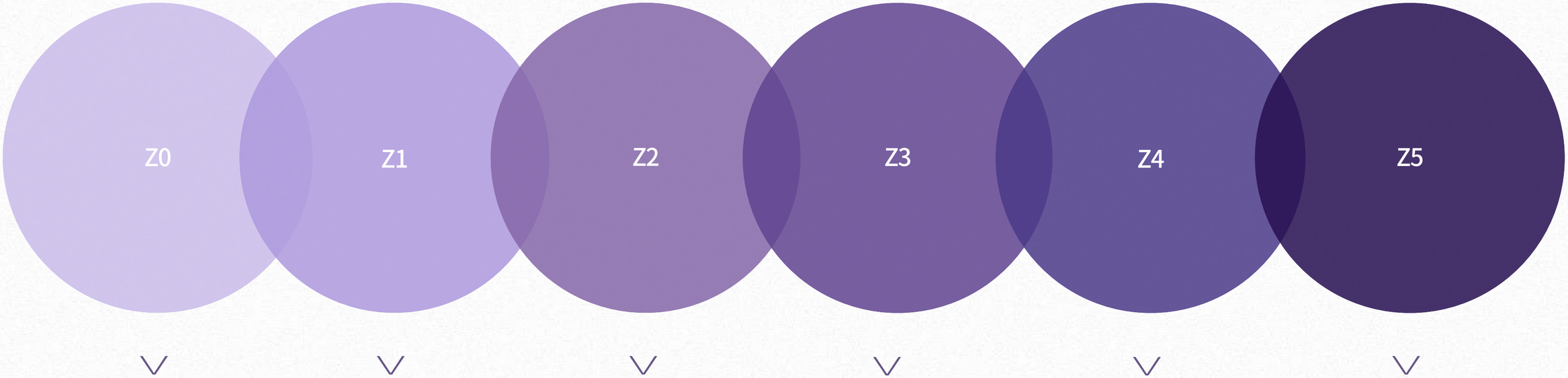
(R_max, 25)

>

- Expected Runs (Goal)

$$\sum_{r=0}^{R\_max} r \times U_{n[r]} = ER$$

Expected Runs.



Weighted ER

특정 상황에서 각각의 Latent Class에 속할 확률을 가중치로 가지는 Transition Matrix를 생성하여
상황마다의 각기 다른 Transition Matrix을 이용하여 Expected Run 계산

03.
Result

Result.

: Expected Runs

- 상황 별 ER 계산

p_ability	x	s	y	p_z	z	bat_count	ER
2	13	0	1	[0.207482	5	3	0.77
2	29	1	2	[0.152748	5	3	0.66
2	45	2	24	[0.125761	5	3	0.66
2	61	0	1	[0.198935	3	4	0.54
2	77	1	2	[0.186851	5	4	0.43
2	93	2	14	[0.101376	5	4	0.47
2	109	14	24	[0.136136	1	4	0.44
2	125	0	12	[0.162641	5	3	0.73
2	141	12	13	[0.160563	3	3	0.71
2	13	13	24	[0.243387	0	3	0.7
2	29	0	12	[0.160573	5	4	0.57
2	45	12	13	[0.109541	5	4	0.42
2	61	13	14	[0.258376	0	4	0.55
2	77	14	24	[0.213560	0	4	0.38
2	89	0	0	[0.186695	2	6	0
2	105	0	1	[0.101118	5	6	0
2	121	1	13	[0.129475	3	6	0.06
2	137	13	14	[0.212484	5	6	0.06
2	9	14	20	[0.254499	0	6	0.07
2	25	20	24	[0.283830	0	6	0.08

- 성능 평가 (RMSE)

```
np.sqrt(np.mean((data["Runs.Inning"] - data["ER"]) ** 2))
```

✓ 0.0s

```
1.3130698111616383
```

>> 실제 득점 값과 비교한 결과 1.3의 RMSE값을 가짐 (논문은 1.7)

Result.

의의

투수와 타자의 능력과 매치업을 모두 고려한 기대득점 계산 가능
Threshold를 설정 한 뒤 특정 이닝에서 기대 득점이 해당 임계치를 초과할 때 투수 교체의 가이드라인을 제공

Proposal

중간계투 교체 타이밍 계산으로 모델 확장
기대득점 대신 기대피안타율, 출루허용률을 계산하여 투수 교체 타이밍을 제안하는 모델 생성

04.
Discussion