SSY316 Advanced probabilistic machine learning:

True Skill

Weilong Chen, Jingkai Zhou

Take-home exam for course
SSY316 Advanced Probabilistic Machine Learning

CHALMERS
UNIVERSITY OF TECHNOLOGY

2023/01/14

Q.1 Modeling

According to the publications. The ranking is processed between two players with their skills' means and variances(uncertainties), followed by the outcome of the game with a determined mean and varying variance.

Since variables s_1, s_2, and t are Gaussians, $\mathcal{N}(variable; \mu, \sigma^2)$ can be used to describe their distribution.

variable	μ	σ^2
s_1	μ_1	σ_1^2
s_2	μ_2	σ_2^2
t	$\mu_1 - \mu_2$	σ_O^2

Table 1: Hyper-parameters

The table above represents 5 hyper-parameters and *Outcome*'s mean determined by μ_1 and μ_2, as for the discrete variable y.

$$p(y) = \delta(y = sign(t))$$

The Bayesian model is a joint distribution of all the random variables.

$$p(s_1, s_2, t, y) = p(s_1)p(s_2)p(t|s_1, s_2)p(y|t)$$

Q.2 Bayesian Network

Here is an illustration of the Bayesian network.

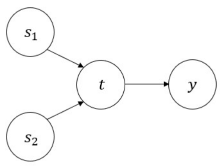


Figure 1: Bayesian network

The first set is obvious. There is no relationship between any random pair of players.

$$s_1 \perp s_2 | \emptyset$$

In addition, if given the outcome t, set $s = \{s_1, s_2\}$ is conditionally independent of the final result y.

$$s \perp y | t$$

Q.3 Computing with the model

$p(s_1, s_2|t, y)$

From the previous results, if t is given, s would be independent of y.

$$p(s_1, s_2|t, y) = p(s_1, s_2|t)$$

With the Bayesian formula:

$$p(s_1, s_2|t) = \frac{p(t|s_1, s_2)p(s_1, s_2)}{p(t)}$$

In it, $p(t|s_1, s_2)p(s_1, s_2) = p(s, t)$

$$p(s, t) = \mathcal{N}(\begin{bmatrix} s \\ t \end{bmatrix}; \begin{bmatrix} \mu_s \\ \mu_t \end{bmatrix}, \begin{bmatrix} \Sigma_s & 0 \\ 0 & \sigma_O^2 \end{bmatrix})$$

Where $\mu_t = \begin{bmatrix} 1 & -1 \end{bmatrix} s$, and $\Sigma_s = \begin{bmatrix} \sigma_1^2 & 0 \\ 0 & \sigma_2^2 \end{bmatrix}$. According to affine transformation. $p(s|t)$ is Gaussian distributed with mean $\mu_{s|t}$ and covariance $\Sigma_{s|t}$.

$$\mu_{s|t} = \Sigma_{s|t}(\Sigma_s^{-1}\mu_s + [1, -1]^T \sigma_O^{-2} t), \quad \Sigma_{s|t} = (\Sigma_s^{-1} + [1, -1]^T \sigma_O^{-2}[1, -1])^{-1}$$

$p(t|s1, s2, y)$

From the proportionality $p(t|s1, s2, y) \propto p(y|t)p(t|s1, s2)$, $p(y|t)$ can only be binary -1, and 1. While $s \in \mathcal{R}$. So the $p(t|s1, s2, y)$ is truncated Gaussian distributed.

- if y is given 1,

$$p(t|s1, s2, y) \propto \begin{cases} \mathcal{N}(t; s_1 - s_2, \sigma_O^2), & \text{if } t > 0. \\ 0, & \text{otherwise.} \end{cases}$$

- if y is given -1,

$$p(t|s1, s2, y) \propto \begin{cases} \mathcal{N}(t; s_1 - s_2, \sigma_O^2), & \text{if } t < 0. \\ 0, & \text{otherwise.} \end{cases}$$

$p(y = 1)$

From the truncated Gaussian distribution, $p(y = 1) = p(t > 0)$. According to the marginalization of affine transformation, marginalizing $p(t, s)$ over s results in $p(t)$ being Gaussian distributed.

$$\mu_t = [1, -1]\mu_s, \quad \sigma_t^2 = \sigma_O^2 + [1, -1]\Sigma_S[1, -1]^T$$

Q.4 Gibbs Sampler

Gibbs sampling

Firstly, the coding step is to sample s with given t and y. See appendix listing 1. Secondly, is to sample t with given s and y(listing 2). Then, the *GibbsSampling* function gives out samples of s(listing 3).

Here is the plot of sampling s with number of sampling = 200:

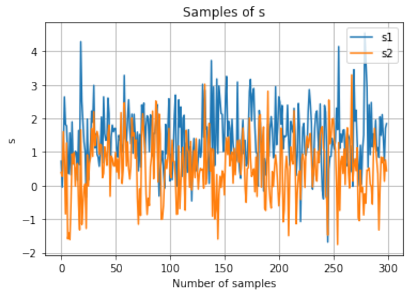


Figure 2: Samples of s

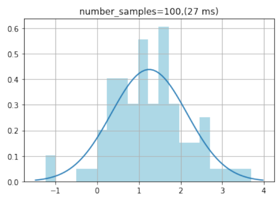
Since the Gibbs sampling needs some iterations to converge, we set burn-in to 20.

Convert to distributions

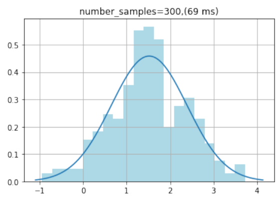
Function for converting samples into Gaussian distribution can be found.(listing 4

Comparisons

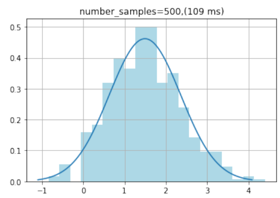
Here is the running time on six different numbers of sampling.



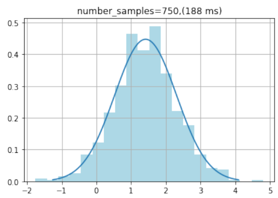
(a) N = 100



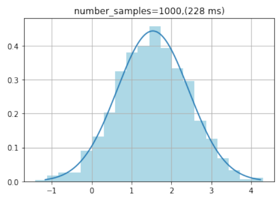
(b) N = 300



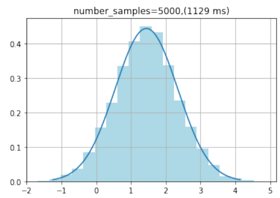
(c) N = 500



(d) N = 750



(e) N = 1000



(f) N = 5000

Figure 3: Running time on four different numbers of sampling.

We see the number of sampling is practical and time-saving is around $750 - 1000$. Which gives fairly precise results and acceptable running time.

Prior and posterior

The prior of all players was set as Gaussian distribution with $\mu = 1$ and $variance = 1$. Here is the plot of the comparison between prior and posterior with the given condition $y = 1$.

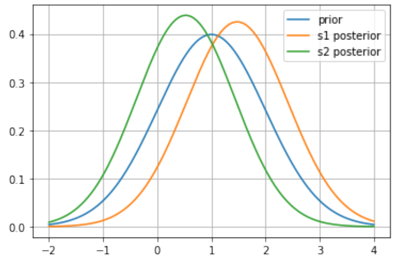


Figure 4: Prior and posterior

It is correct to see s_1 has a bigger 'win-mean' than s_2.

Q.5 Assumed Density Filtering

Rankings

The final skill ranking is shown here.

rank	team name
1	Juventus
2	Napoli
3	Milan
4	Inter
5	Roma
6	Atalanta
7	Torino
8	Lazio
9	Sampdoria
10	Bologna
11	Spal
12	Udinese
13	Sassuolo
14	Parma
15	Empoli
16	Cagliari
17	Genoa
18	Fiorentina
19	Frosinone
20	Chievo

Change the order of the matches

After shuffling the order of the matches, the ranking may change. Since the skill of each team is represented by mean and variance. After initializing all means and variances into 1 and 1 respectively, during the matches, the variances are all becoming smaller and smaller. As a result of this, early matches have greater uncertainty. Thus, a weaker team has a greater chance of beating a stronger one. Because of this, the order in which matches are played is important.

Q.6 Using the model for predictions

Since there are draws in the dataset, the threshold of setting the draw is needed. This threshold can be treated as a hyperparameter. The accuracy of the whole dataset is 0.468 this number may be varying. The accuracy is obviously higher than random guessing.

Q.7 Factor graph

According to the joint distribution, the factor graph is drawn below.

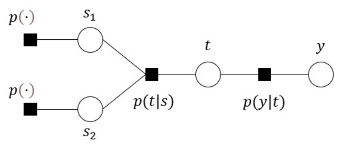


Figure 5: Factor graph

And according formulas are here.

$$
\begin{aligned}
\mu_{p(\cdot)\to s_1} &= \mathcal{N}(s_1; \mu_{s_1}, \sigma_{s_1}^2) \\
\mu_{p(\cdot)\to s_2} &= \mathcal{N}(s_2; \mu_{s_2}, \sigma_{s_2}^2) \\
\mu_{s_1\to p(t|s)} &= 1 \cdot \mathcal{N}(s_1; \mu_{s_1}, \sigma_{s_1}^2) \\
\mu_{s_2\to p(t|s)} &= 1 \cdot \mathcal{N}(s_2; \mu_{s_2}, \sigma_{s_2}^2) \\
\mu_{p(t|s)\to t} &= \mathcal{N}(s; \mu_{t|s}, \sigma_{sO}^2) \cdot \mu_{s_1\to p(t|s)} \cdot \mu_{s_2\to p(t|s)}
\end{aligned}
$$

Q.8 A message-passing algorithm

The factor $\mu_{p(t|s)\to t}$ is not Gaussian. Use Moment matching on the truncated Gaussian, an approximation can be obtained.

$$
\begin{aligned}
\mu_{p(t|s)\to t} &= \mu_{p(y|t)\to t} = \mathcal{N}(t, \mu_t, \sigma_t^2) \\
\mu_{y\to p(y|t)} &= 1 \\
\mu_{p(y|t)\to t} &= p(y|t)
\end{aligned}
$$

The marginal of s_1 and s_2 can be obtained as below.

$$
\begin{aligned}
p(s_1|t) &= \mathcal{N}(s_1; \mu_{s_1}, \sigma_{s_1}^2) \cdot \mathcal{N}(s_1; \mu_{y\to p(y|t)} + \mu_{p(y|t)\to t}, \sigma_{t|s}^2 + \sigma_1^2 + \sigma_2^2) \\
p(s_2|t) &= \mathcal{N}(s_2; \mu_{s_2}, \sigma_{s_2}^2) \cdot \mathcal{N}(s_2; \mu_{y\to p(y|t)} - \mu_{p(y|t)\to t}, \sigma_{t|s}^2 + \sigma_1^2 + \sigma_2^2) Z
\end{aligned}
$$

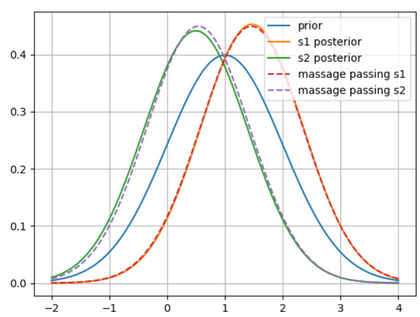


Figure 6: Massage passing and Gibbs Sampling

The posteriors using moment-matching and Gibbs sampling looks very similar.

Q.9 Your own data

We will test TrueSkill using English Premier League (EPL) data. The purpose is to predict the result of EPL's season 21/22 by the learned prior from season 20/21.

Firstly, implement Gibbs sampling to get priors from season 20/21. Here is the ranking list for the season 20/21.

rank	team name
1	Manchester City
2	Manchester United
3	Liverpool
4	Chelsea
5	Leicester City
6	West Ham United
7	Arsenal
8	Tottenham Hotspur
9	Aston Villa
10	Leeds United
11	Everton
12	Brighton & Hove Albion
13	Newcastle United
14	Wolverhampton Wanderers
15	Southampton
16	Burnley
17	Crystal Palace
18	Fulham
19	Sheffield United
20	West Bromwich Albion

Since EPL has a promotion and relegation regulation. The first three teams from English First League (EFL) promote to EPL and the last three EPL teams are relegated to EFL. Inheriting the skills of three tail teams from last season, here we approximate the skills of the three new teams from EFL.

After these operations, the prediction of season 21/22 is shown below. The matches prediction's accuracy is 0.539.

rank	team name
1	Liverpool
2	Manchester City
3	Chelsea
4	Arsenal
5	Manchester United
6	Tottenham Hotspur
7	Newcastle United
8	Crystal Palace
9	Brighton & Hove Albion
10	West Ham United
11	Aston Villa
12	Leicester City
13	Everton
14	Brentford
15	Southampton
16	Wolverhampton Wanderers
17	Burnley
18	Leeds United
19	Watford
20	Norwich City

Q.10 Open-ended project extension

We have tried to make the model more reasonable considering the score instead of just the win or lose. We set to give more weight to the winner if the score difference is larger. Samely, give less weigt to the loser. See code listing 5. Give bias to the score difference. The final result has a 5% accuracy improvement.

Appendix A Code

```
def sampling_s_ty(sigma_0, mu_s, sigma_s, t):
    ### sampling variable s=s1,s2 from condtion t and y(eliminated in this case)
    ### Input: sigma_0, mu_s, sigma_s, t
    ### Ontput: one vrs with Gaussian(mu_s_ty, sigma_s_ty)
    A = np.array([[1,-1]])
    sigma_s_ty = inv(inv(sigma_s) + A.T * (1/sigma_0) @ A)
    mu_s_ty = sigma_s_ty @ (inv(sigma_s) @ mu_s + A.T * (1/sigma_0) * t)
    return stats.multivariate_normal.rvs(mu_s_ty.reshape(-1), sigma_s_ty)
```

Listing 1: sampling

```
def sampling_t_sy(y, s_1, s_2, sigma_0):
    ### sampling variable t from condition =s1,s2 and y
    ### Input: y, s_1, s_2, sigma_0
    ### Output: one vrs with truncated Gaussian(a, b, loc, scale)
    loc = s_1 - s_2
    scale = sigma_0
    if y == 1:
        myclip_a = 0
        myclip_b = np.Inf
    elif y== -1:
        myclip_a = -np.Inf
        myclip_b = 0
    a, b = (myclip_a - loc) / scale, (myclip_b - loc) / scale
    return stats.truncnorm.rvs(a, b, loc, scale)
```

Listing 2: sampling

```
def GibbsSampling(initial_point, y, sigma_s, sigma_0, num_samples):
    ### Input: initial points for s_1, s_2, y, mu_s, sigma_s, sigma_0, num_samples
    ### Output: samples for s_1 and s_2
    samples_s_1 = []
    samples_s_2 = []
    s_1 = initial_point[0]
    s_2 = initial_point[1]
    mu_s = np.array([[1, 1]]).T
    for i in range(num_samples):
        t = sampling_t_sy(y, s_1, s_2, sigma_0)
        s_1,s_2 = sampling_s_ty(sigma_0, mu_s, sigma_s, t)
        samples_s_1.append(s_1)
        samples_s_2.append(s_2)
    return samples_s_1,samples_s_2
```

Listing 3: Gibbs Sampling

```
def cvrt_gaussian(s,mean,var):
    x = np.linspace(mean - var * 3, mean + var * 3, num_samples - burn)
    y = stats.norm.pdf(x, loc=mean, scale=var)
    plt.plot(x, y)
    plt.hist(s, bins=20, density=True, color='lightblue')
```

```
    plt.grid()
```

Listing 4: cvrt gaussian

```
    if score1 - score2 == 2 :
        mu_s = np.array([[prior1[0]*1.1, prior2[0]/1.1]]).T
        #if the score difference is two, then it's biased with coefficient 1.1
    if score1 - score2 >2 :
        mu_s = np.array([[prior1[0]*1.2, prior2[0]/1.2]]).T
        #if the score difference is larger than two, then it's biased with
    coefficient 1.2
    if score1 - score2 == -2 :
        mu_s = np.array([[prior1[0]/1.1, prior2[0]*1.1]]).T
    if score1 - score2 < -2 :
        mu_s = np.array([[prior1[0]/1.2, prior2[0]*1.2]]).T
    else:
        mu_s = np.array([[prior1[0], prior2[0]]]).T
```

Listing 5: score difference bias