Machine Learning with Graphs (MLG)

RecSys: Bayesian Personalized Ranking (BPR) Loss

Learning to rank items for users

Cheng-Te Li (李政德)

Institute of Data Science

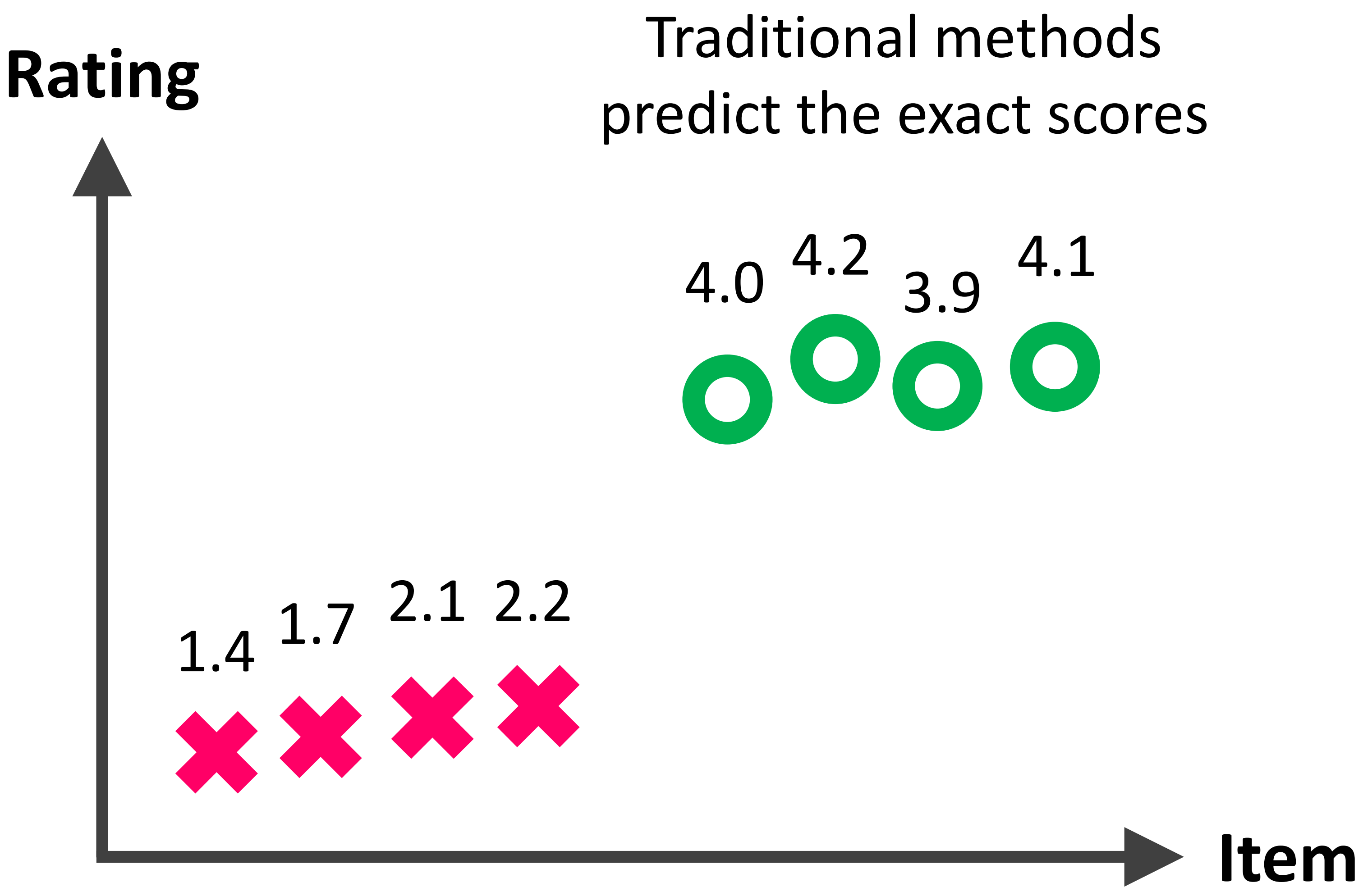
National Cheng Kung University

chengte@mail.ncku.edu.tw

Explicit vs. Implicit Feedback

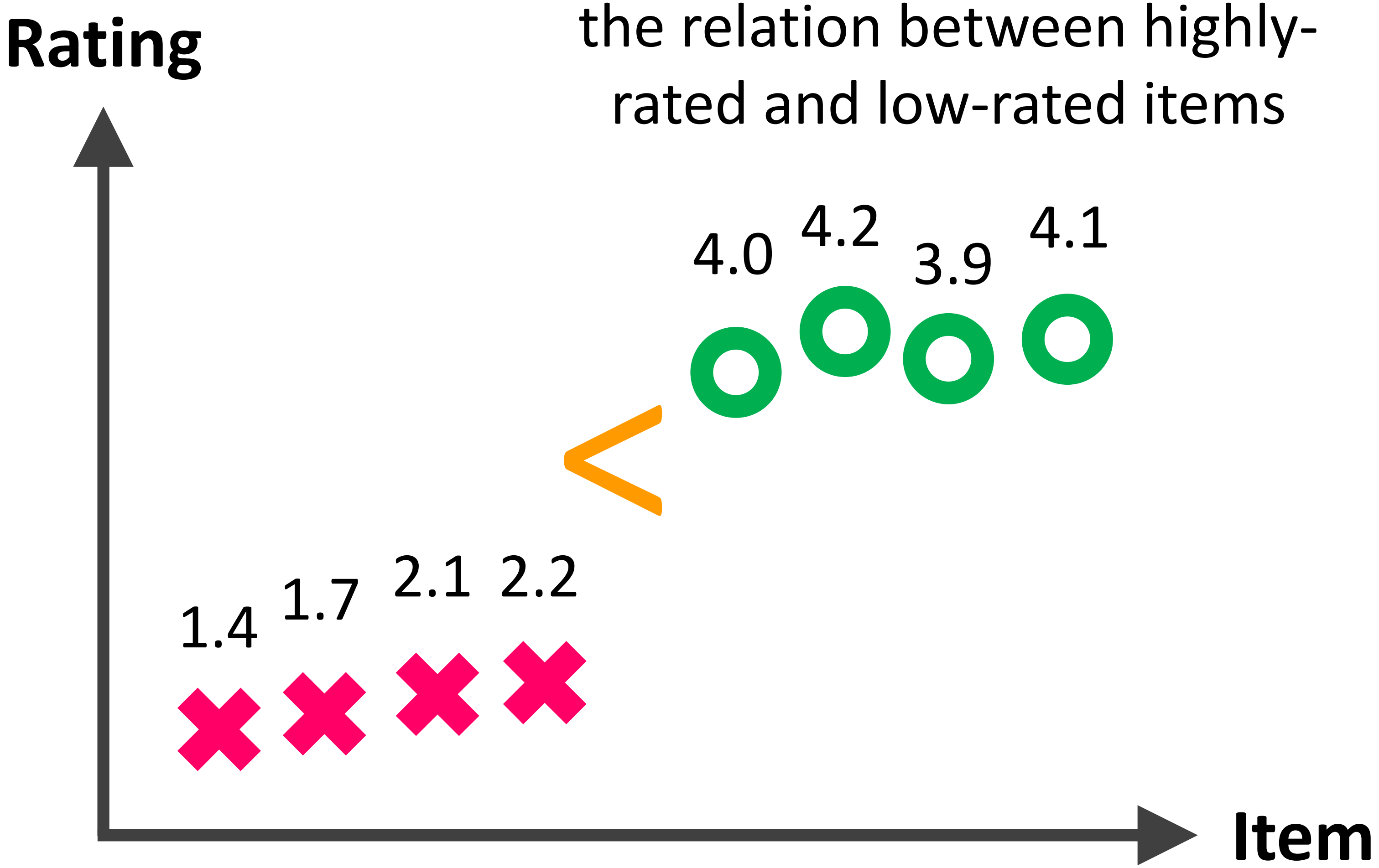
- Learning from explicit feedback (e.g., ratings)
  - CF, MF, and FM
  - → However, at most time, users provide no ratings
- Much easier to collect implicit feedback
  - Clicks on pages and URLs
  - Purchases
  - View times
  - → Already available in log files at the web servers
- Can we learn personalized ranking from implicit data for recommendation?

Ranking on Items



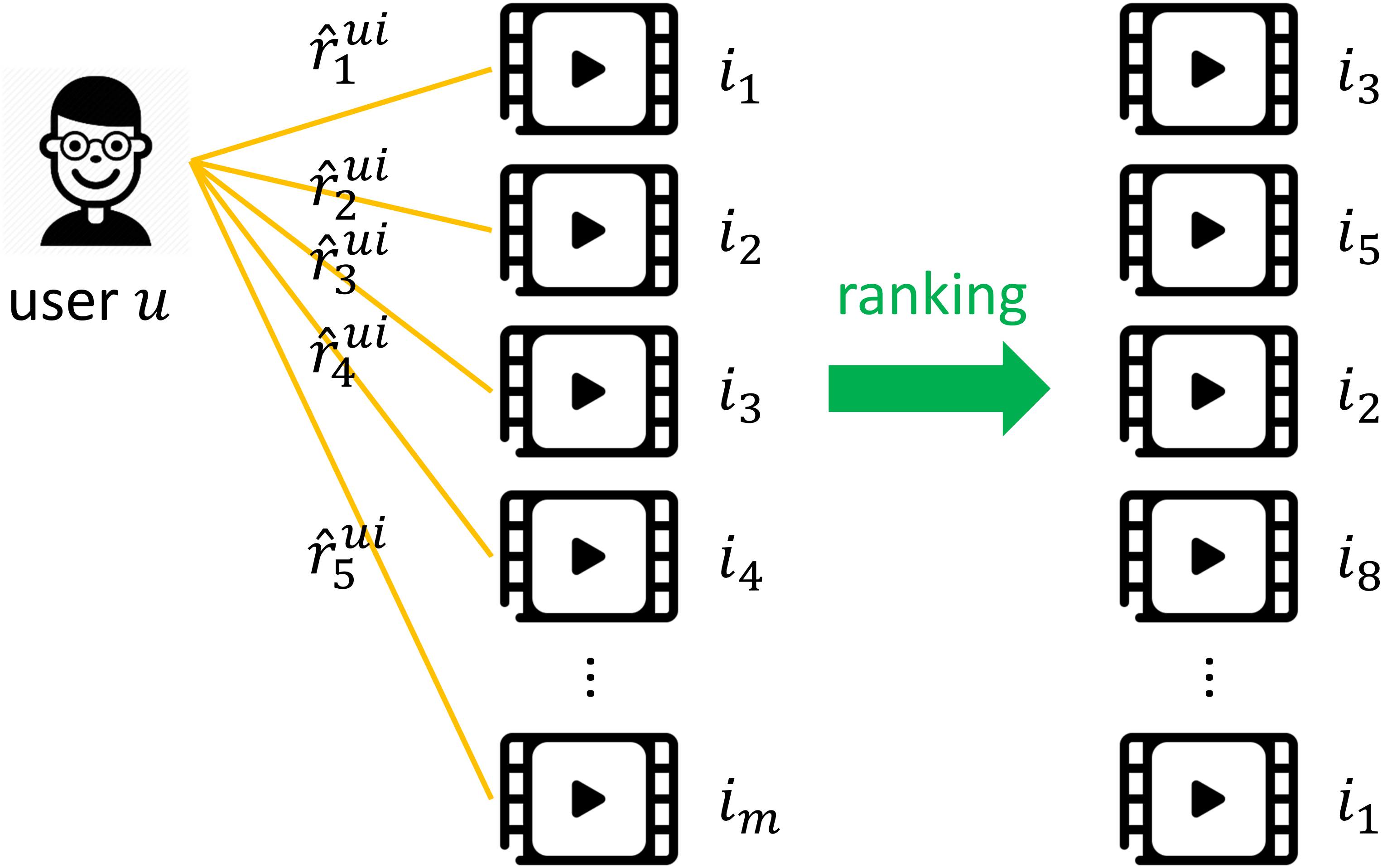
Ranking on Items

BPR only models
the relation between highly-rated and low-rated items



Personalized Ranking

- Goal: provide a user with a ranked list of items
  - Ranking is inferred from implicit behaviors



Filling with “0” ?

- Implicit feedback contains only positive classes, the remaining is a mixture of unknown and negative
  - Filling 0 for unknown/negative, then do CF/MF/FM?
  - This approach tend to predict 0, it cannot work!

user \ item	i_1	i_2	i_3	i_4
u_1	?	+	+	?
u_2	+	?	?	+
u_3	+	+	?	?
u_4	?	?	+	+
u_5	?	?	+	?

user \ item	i_1	i_2	i_3	i_4
u_1	0	1	1	0
u_2	1	0	0	1
u_3	1	1	0	0
u_4	0	0	1	1
u_5	0	0	1	0

Ranking Settings

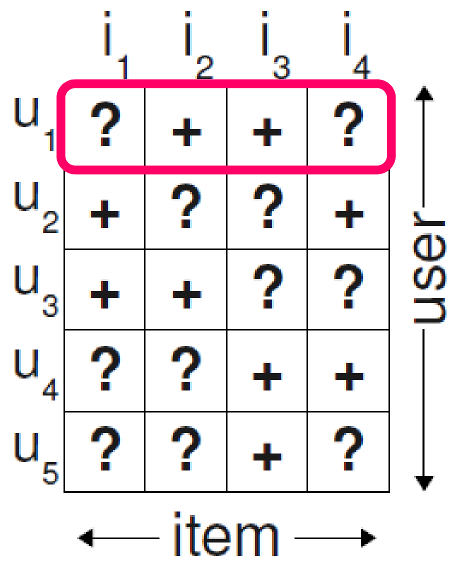
- Only know the ranking between two items but not exactly scores
- Reformat each user-item pairs
  - Training contains both positive and negative
  - Missing to be ranked
- Training data

$$D_S : U \times I \times I$$

$$D_S = \{(u, i, j) \mid i \in I_u^+ \wedge j \in I \backslash I_u^+\}$$

D_S: set of triples that user u likes item i more than item j

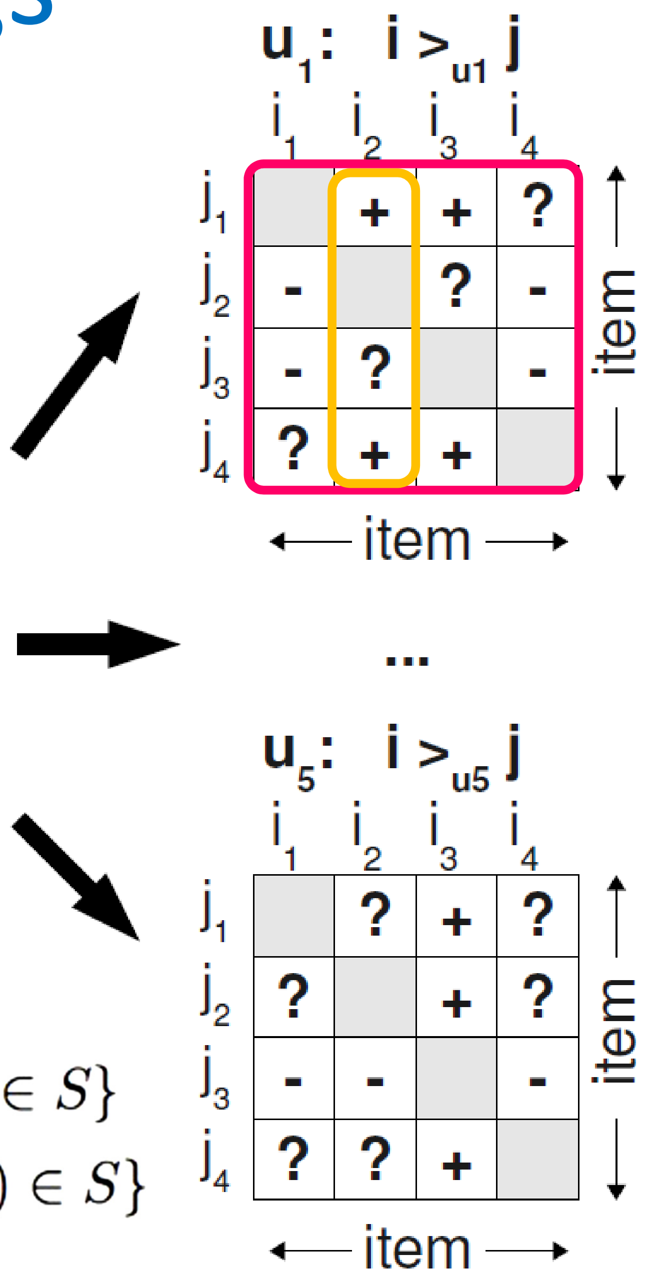
Implicit feedback (only positive)



$S : U \times I$

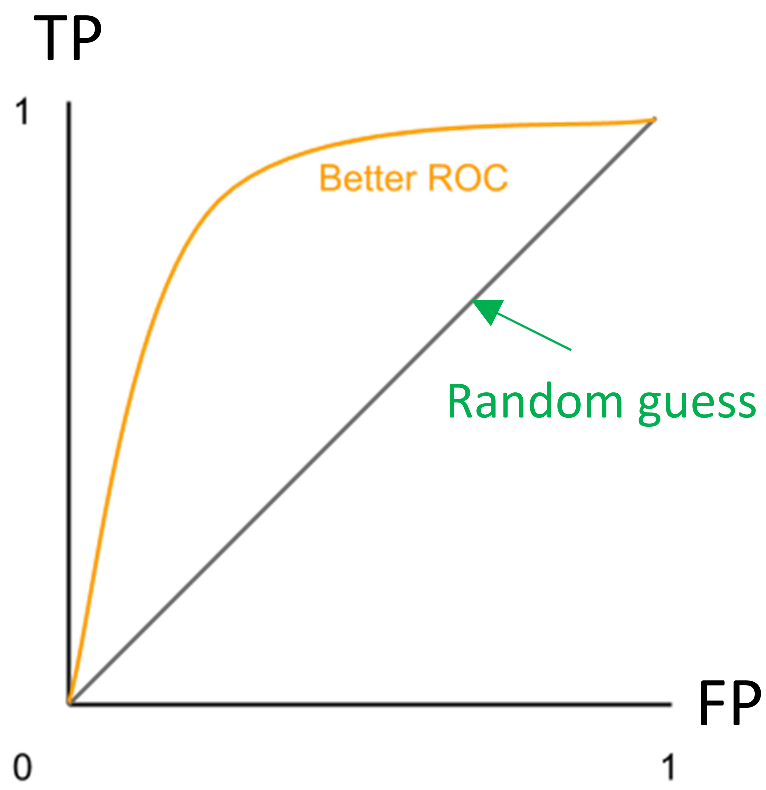
$$I_u^+ := \{i \in I : (u, i) \in S\}$$

$$U_i^+ := \{u \in U : (u, i) \in S\}$$



AUC behind BPR

- Goal: rank item i higher than item j (i.e., $i >_u j$)
  - $\rightarrow$ Given item i higher score than j
- AUC = Area under ROC curve
  - $0 \leq \text{AUC} \leq 1$
- Correctly ranking $i >_u j$ $\approx$ Maximize AUC
- When $\text{AUC} > 0.5$: the prediction is above "slope=1" $\rightarrow$ Tend to predict $i >_u j$ and better than random guess



Optimize AUC

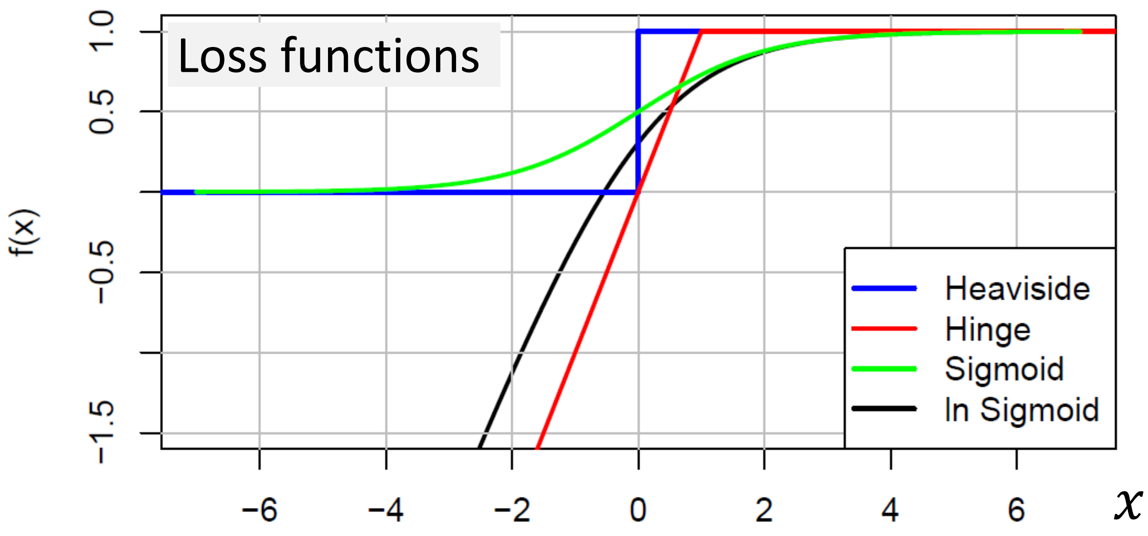
$$\mathrm{AUC}(u) := \frac{1}{|I_u^+|\,|I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0)$$

$\hat{x}_{uij}$: any real-valued function that gives the ranking order between i and j

$$\mathrm{AUC} := \frac{1}{|U|} \sum_{u \in U} AUC(u)$$

$$\delta(x > 0) = H(x) := \begin{cases} 1, & x > 0 \\ 0, & \text{else} \end{cases}$$

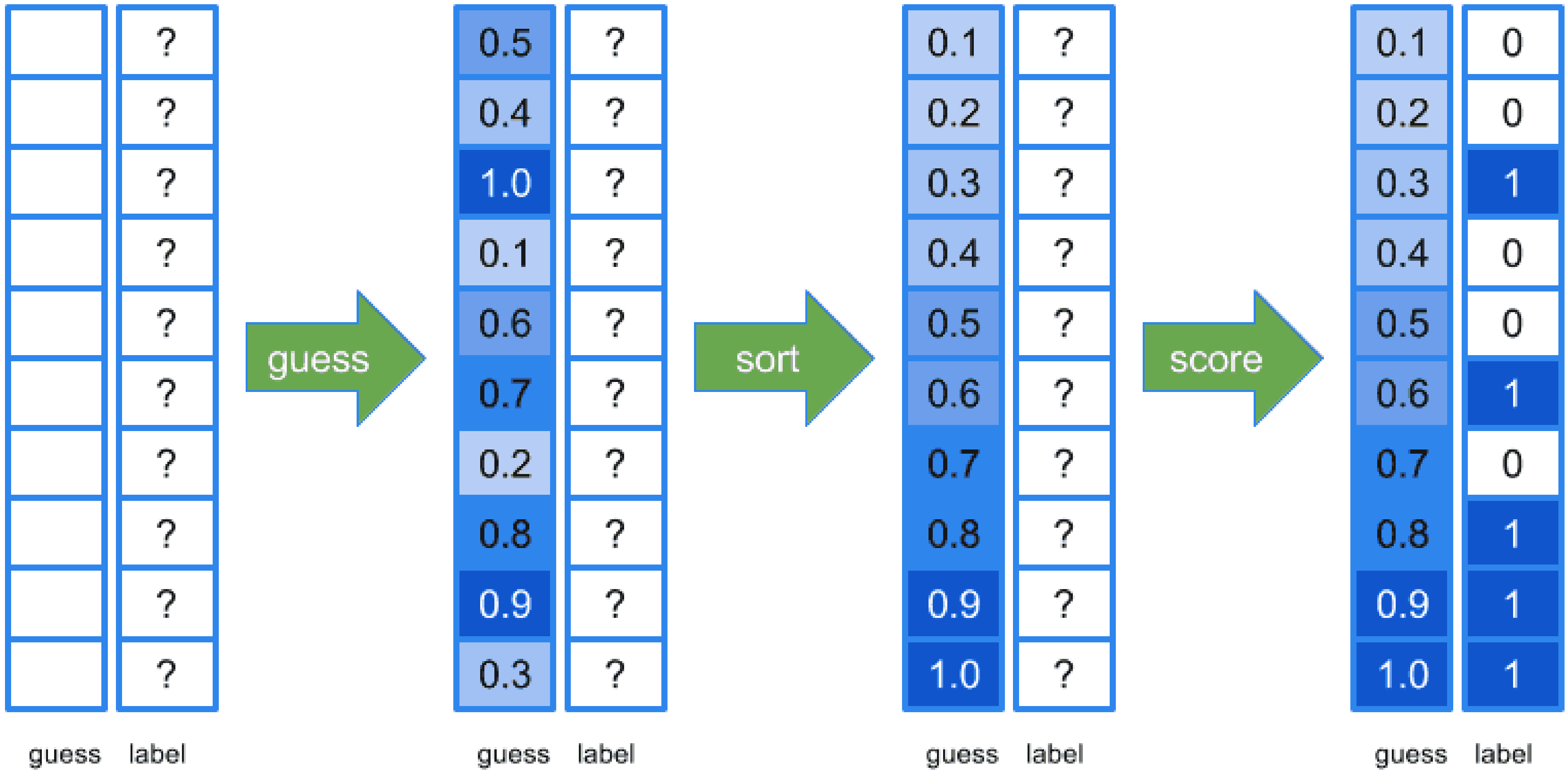
Heaviside function (單位階梯函數)



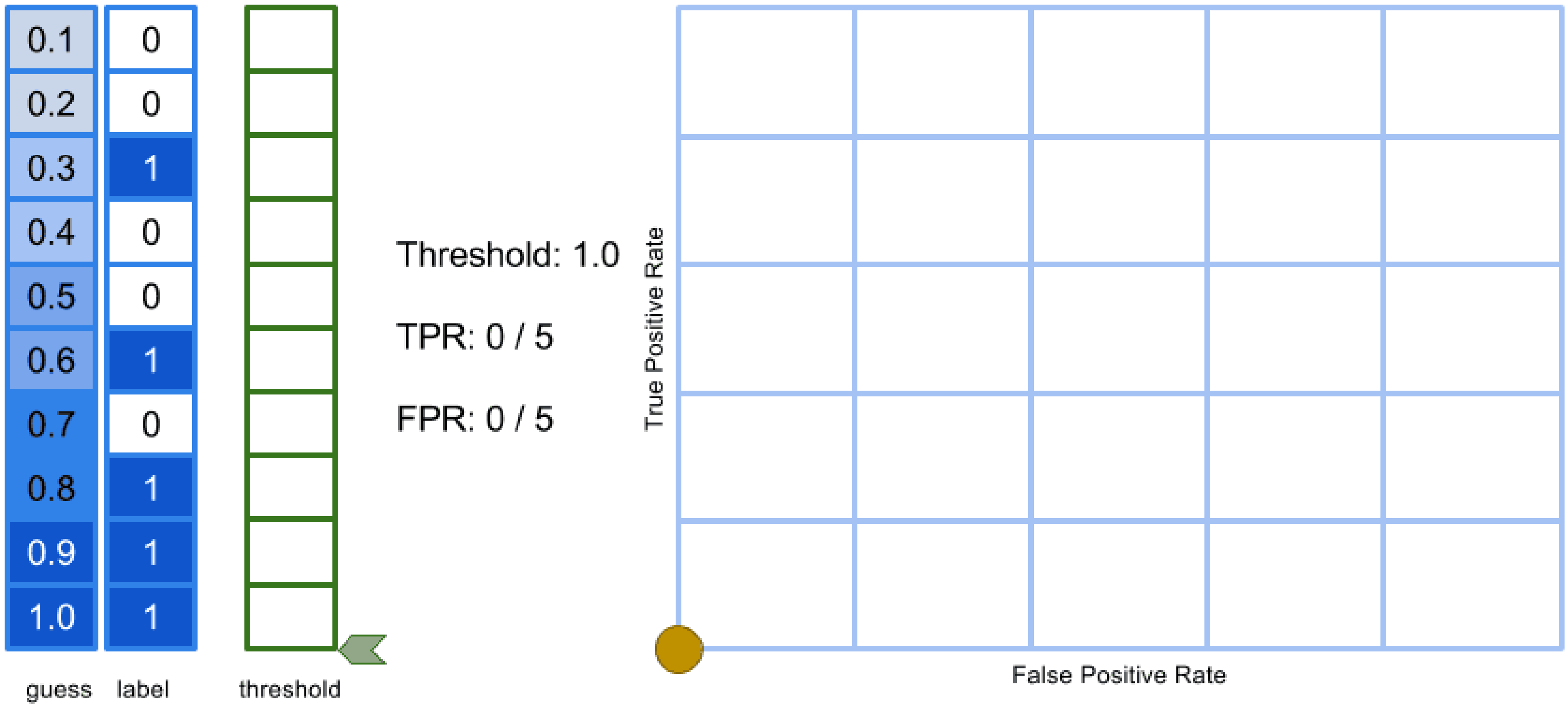
→ non-differentiable (不可微分)

→ ROC is NOT a differentiable smooth curve
i.e., ROC curve is stepwise shape
→ We cannot use gradient descent to optimize AUC score

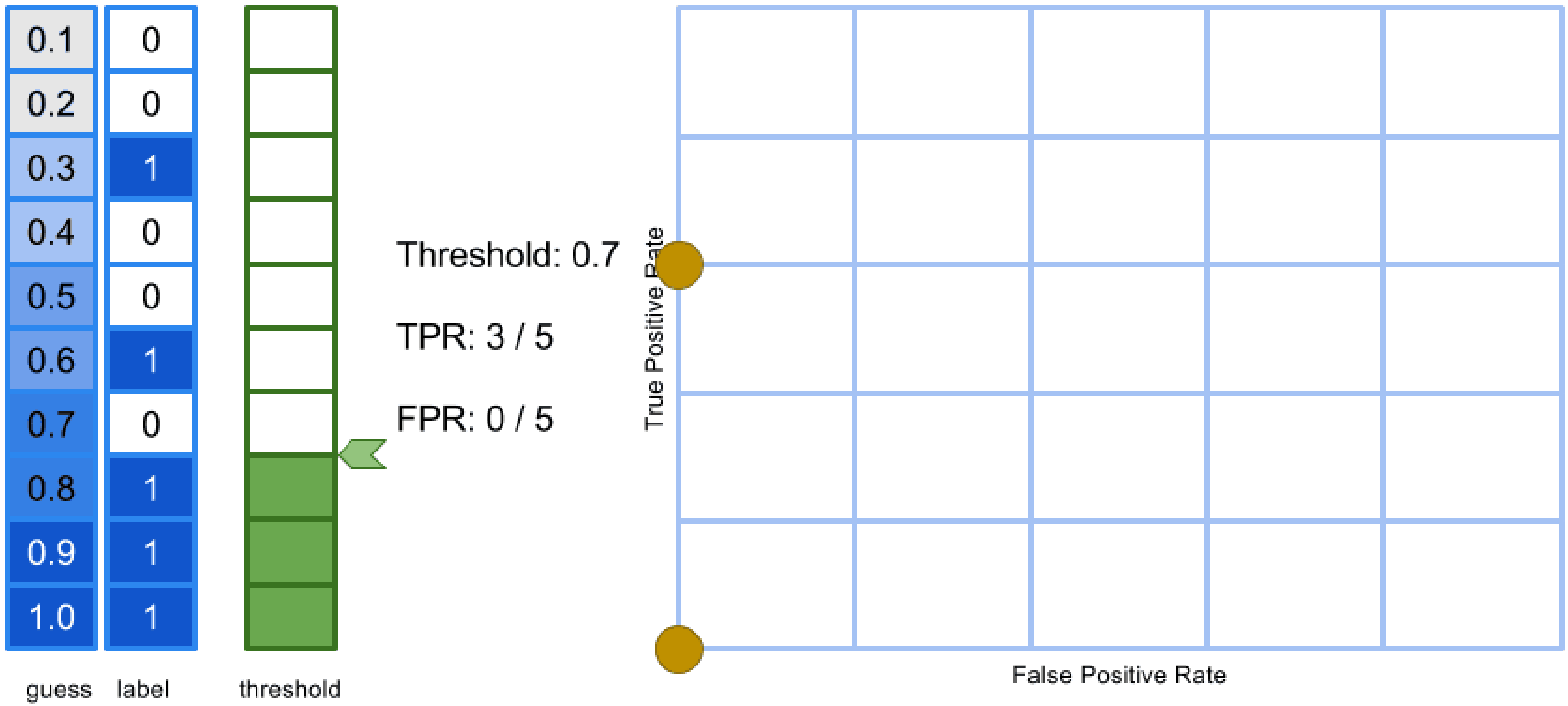
AUC Explained



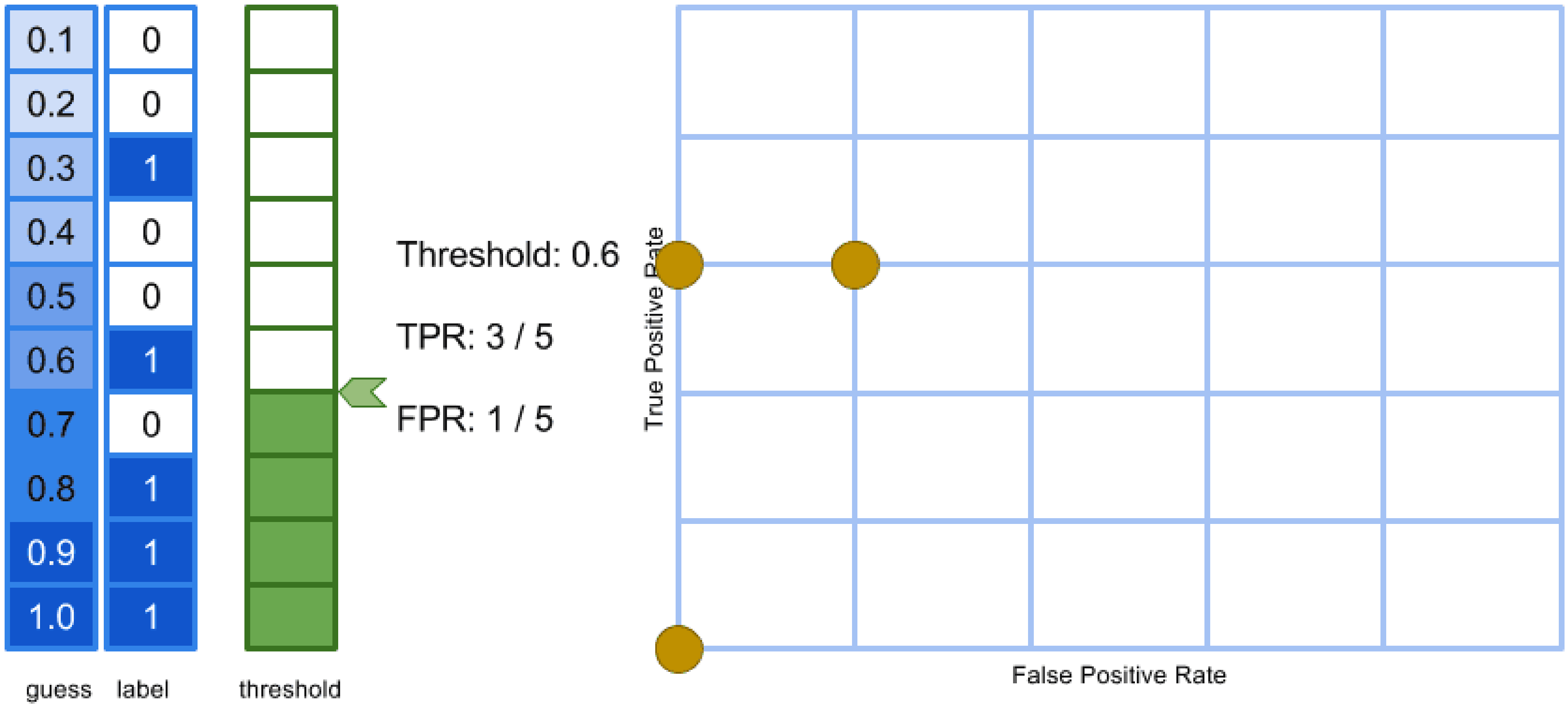
AUC Explained



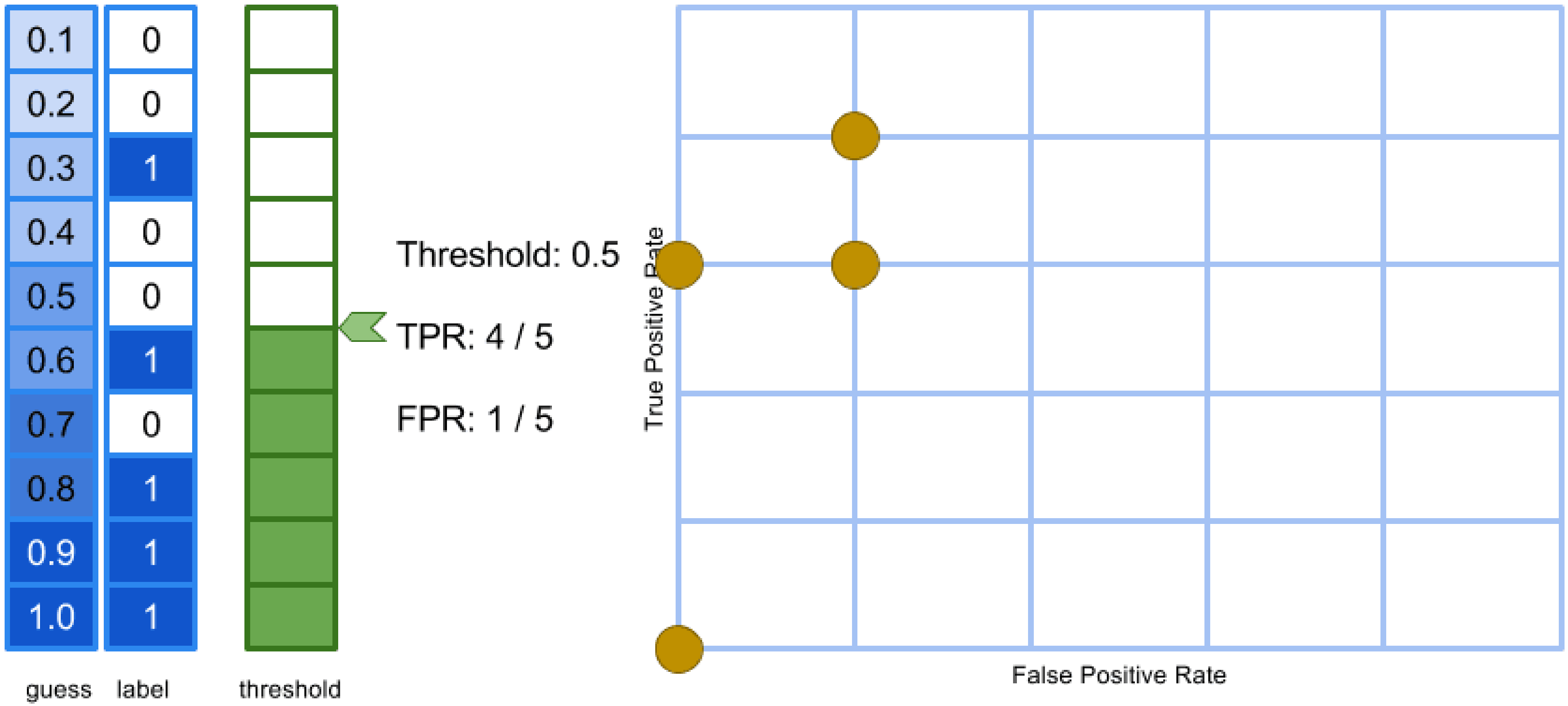
AUC Explained



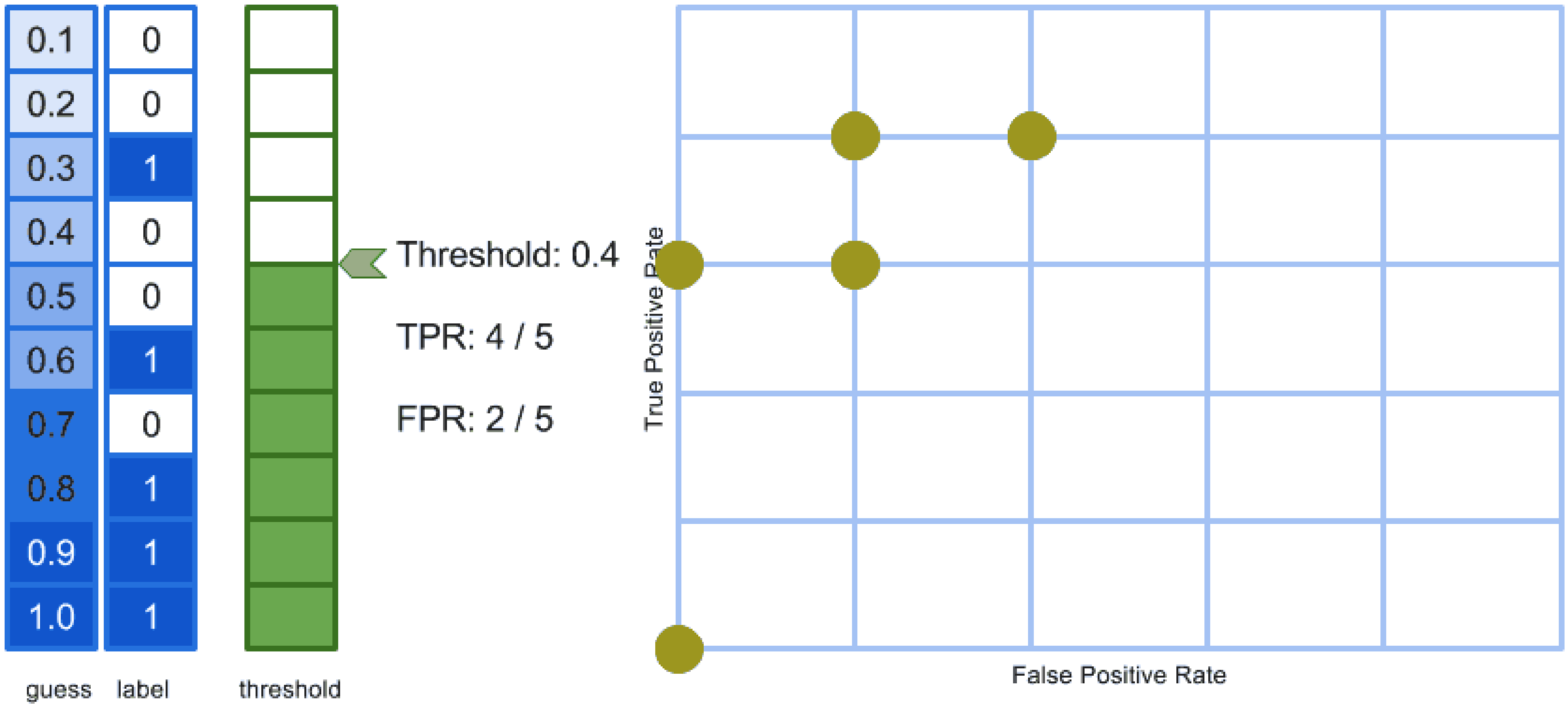
AUC Explained



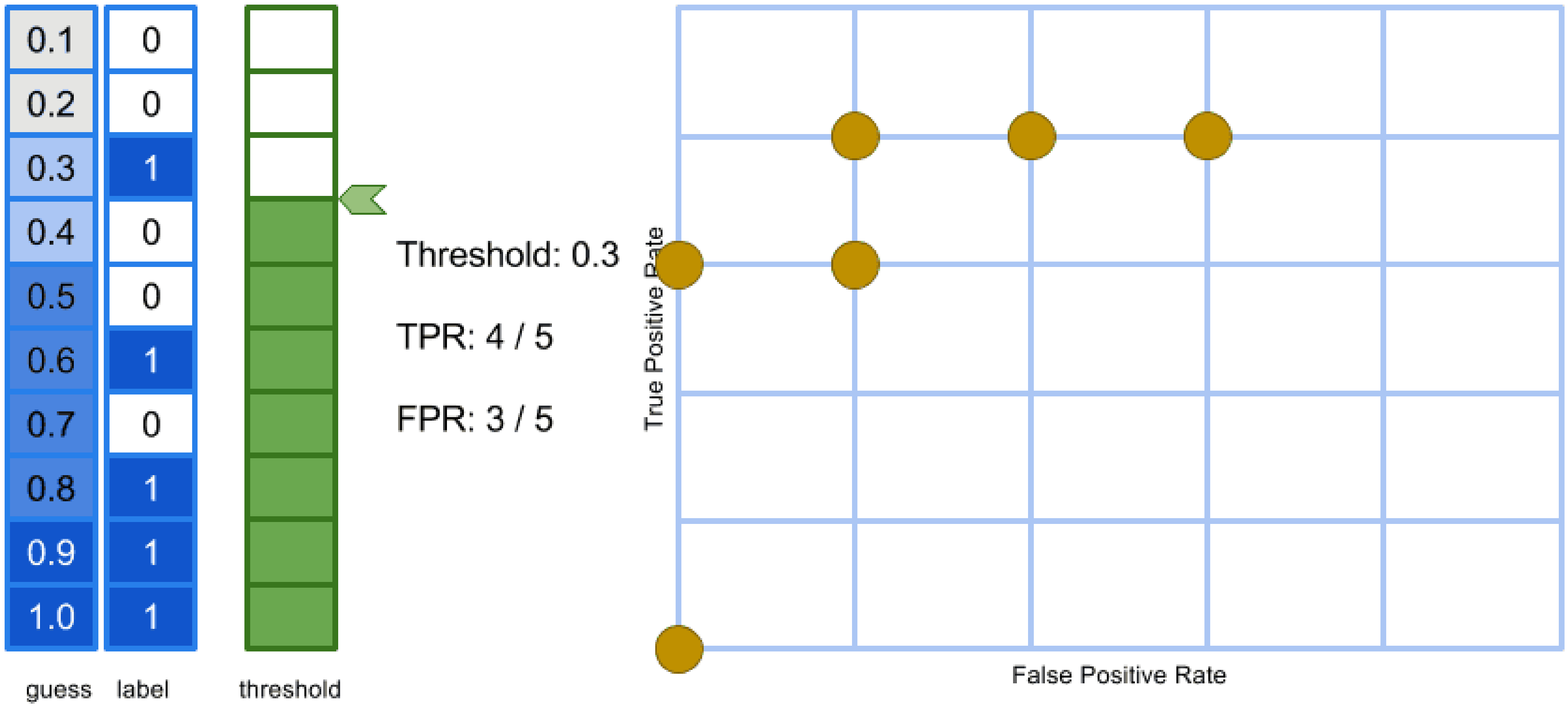
AUC Explained



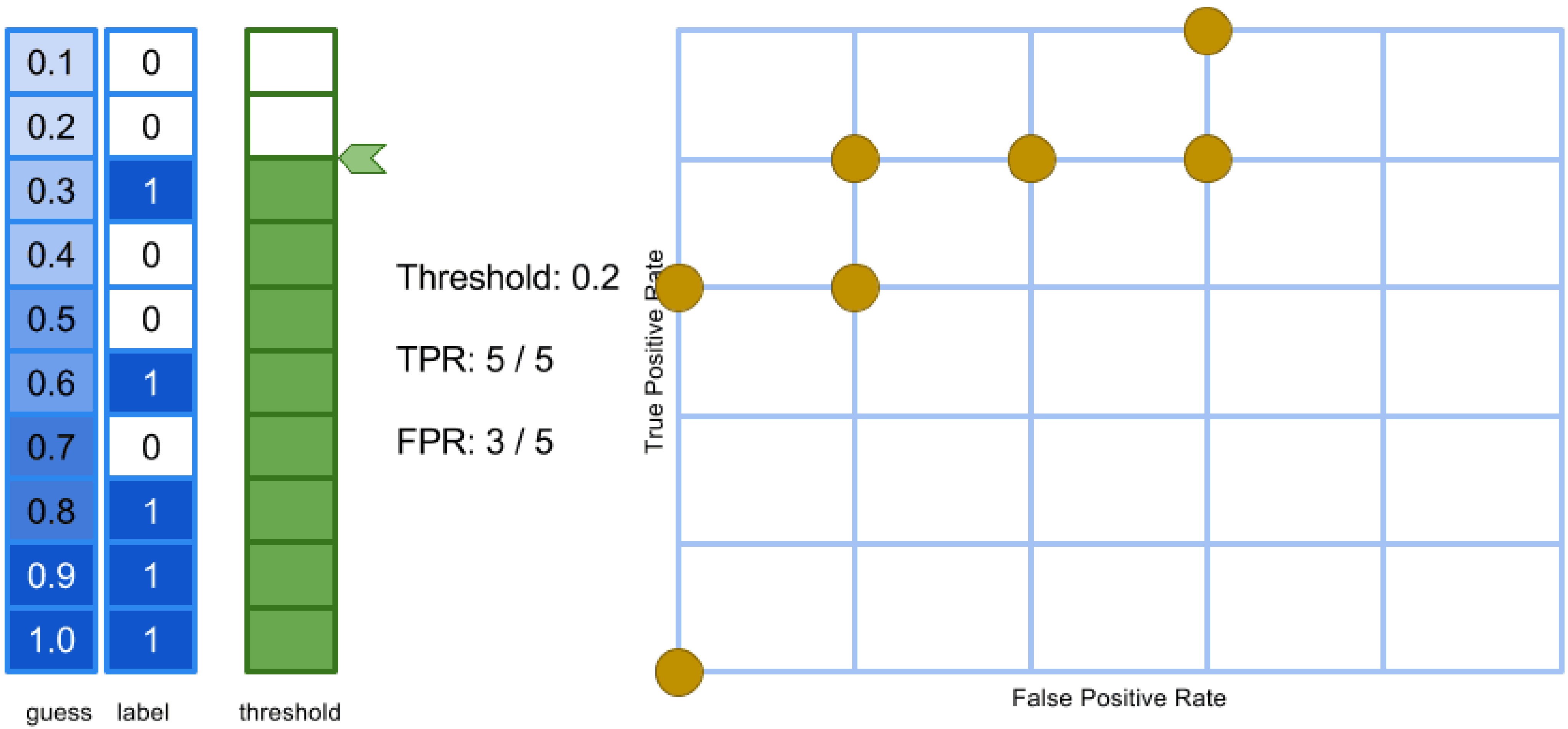
AUC Explained



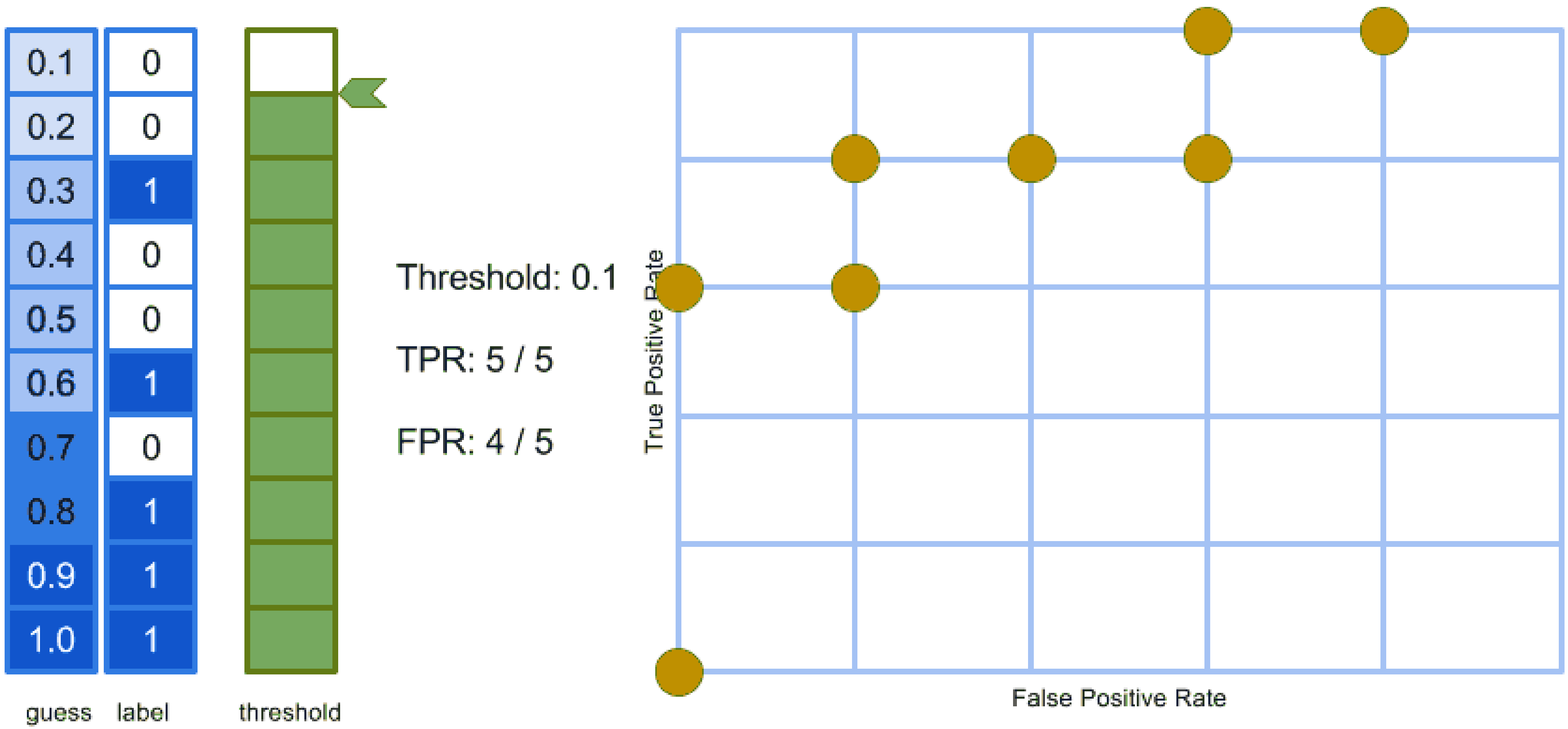
AUC Explained



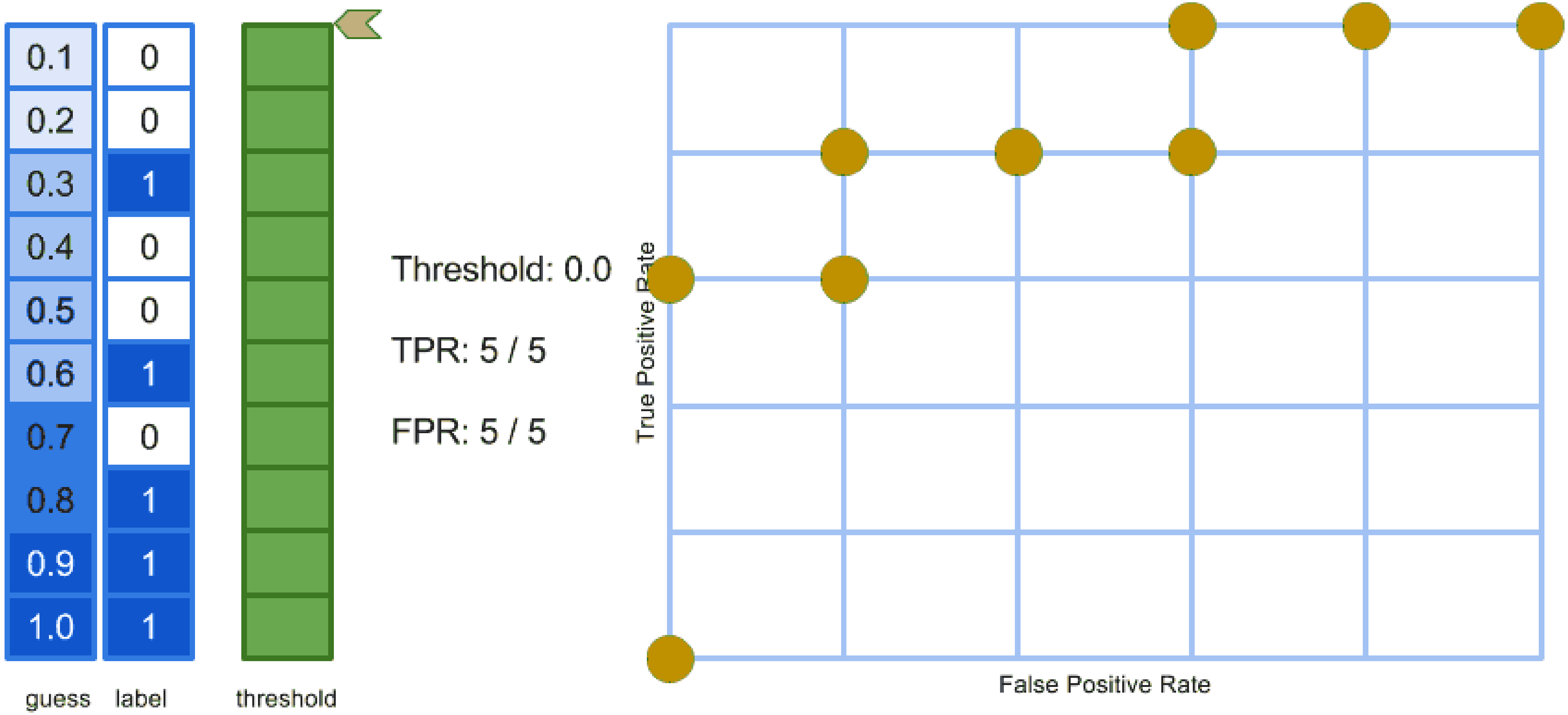
AUC Explained



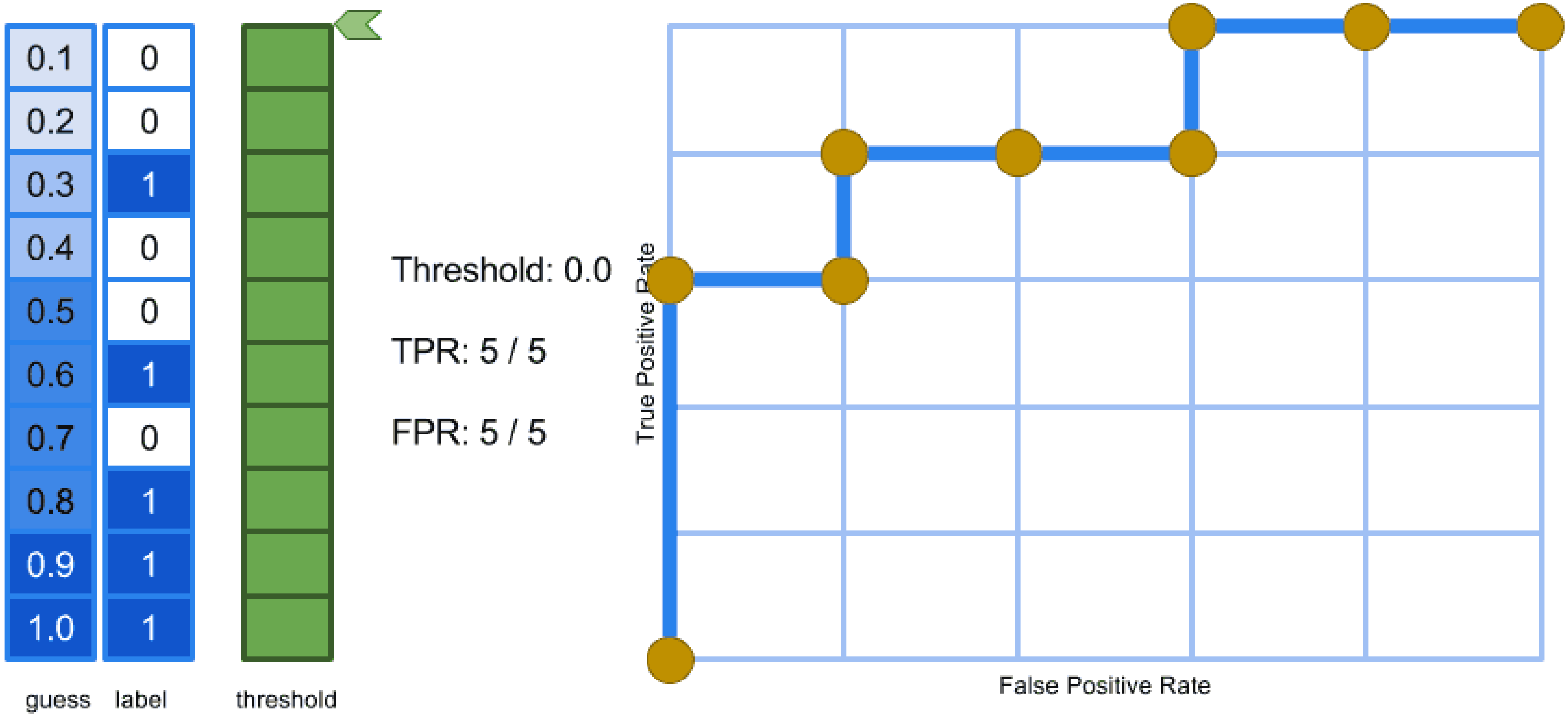
AUC Explained



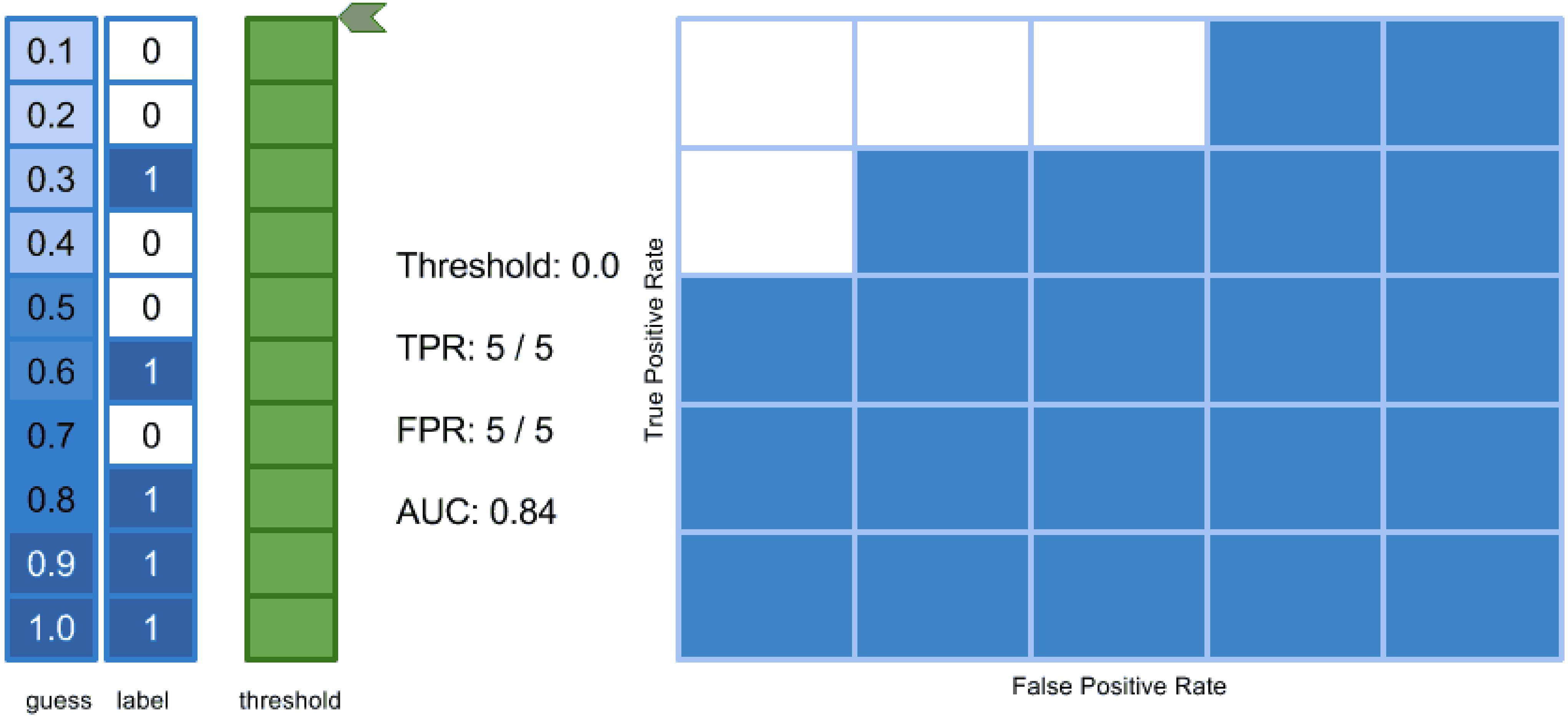
AUC Explained



AUC Explained



AUC Explained



Bayesian Personalized Ranking

- Goal: optimize the posterior probability

$$p(\Theta| >_u) \propto p(>_u |\Theta)p(\Theta)$$

Θ: parameters of any model (e.g., MF)
$>_u$: the desired ranking of items for user u

- Assume every user-item pair (u, i) is independent of each other, and users' preference are also independent

$$\prod_{u \in U} p(>_u |\Theta) = \prod_{(u,i,j) \in D_S} p(i >_u j|\Theta)$$

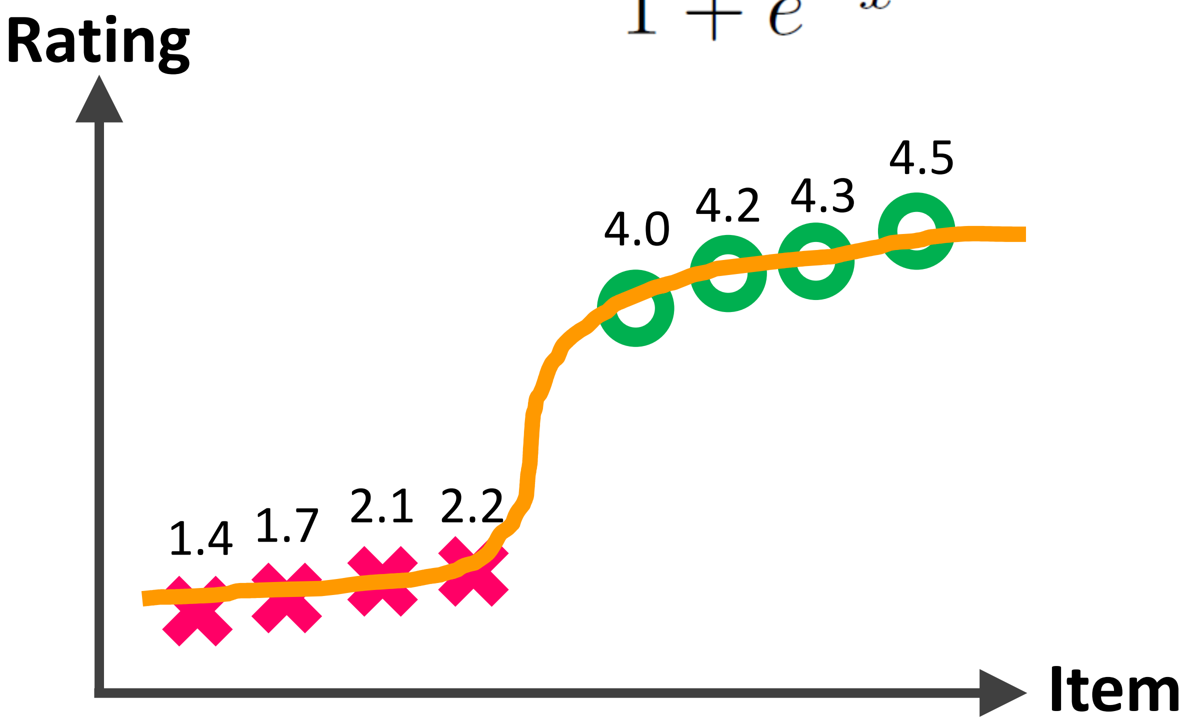
Bayesian Personalized Ranking

- Use sigmoid function to estimate probability $p(>_u |\Theta)$
  - For gradient descent, the objective needs to be differentiable

Rather than Heaviside function

$$p(i >_u j|\Theta) := \sigma(\hat{x}_{uij}(\Theta))$$

$$\sigma(x) := \frac{1}{1+e^{-x}}$$



For regularization and for taking log:

BPR Optimization

$\therefore$ assume: $\Theta \sim N(0, \lambda_\Theta I)$

$$p(\Theta) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\|\Theta\|^2}{2\sigma^2}\right)$$

$$\ln p(\Theta) = -\lambda \|\Theta\|^2$$

$$\begin{aligned}
\text{BPR-OPT} &:= \ln p(\Theta \mid >_u) \\
&= \ln p(>_u \mid \Theta)\, p(\Theta) \\
&= \ln \prod_{(u,i,j) \in D_S} \sigma(\hat{x}_{uij})\, p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) + \ln p(\Theta) \\
&= \sum_{(u,i,j) \in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta ||\Theta||^2
\end{aligned}$$

BPR Optimization

- Gradient descent $$\frac{\partial \text{BPR-O}\textsc{pt}}{\partial \Theta} = \sum_{(u,i,j)\in D_S} \frac{\partial}{\partial \Theta} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \frac{\partial}{\partial \Theta} ||\Theta||^2$$ $$\propto \sum_{(u,i,j)\in D_S} \frac{-e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} - \lambda_\Theta \Theta$$

procedure $\textsc{LearnBPR}(D_S, \Theta)$
    initialize Θ
    repeat
        draw (u, i, j) from D_S Bootstrap sampling
        $\Theta \leftarrow \Theta + \alpha \left(\frac{e^{-\hat{x}_{uij}}}{1+e^{-\hat{x}_{uij}}} \cdot \frac{\partial}{\partial \Theta} \hat{x}_{uij} + \lambda_\Theta \cdot \Theta \right)$
    until convergence
    return $\hat{\Theta}$
end procedure

Matrix Factorization with BPR (BPR-MF)

- BPR is an ranking-based optimization idea, NOT RecSys
  - Can be adopted for existing RecSys models and deep models that can produce real value $\hat{x}_{ui}$ for a user-item pair (u, i)
- MF: $\hat{X} = WH^{\mathrm{T}}$, i.e., $\hat{x}_{ui} = \sum_{f=1}^{k} w_{uf}\, h_{if}$
  - Model parameters $\Theta = \{W, H\}$ (latent features of users / items)

$$BPR = \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2$$

σ: sigmoid

$\hat{x}_{uij}$: any real-valued function that gives the ranking order between i and j

- Define the predicted ranking order $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$

$$\mathrm{AUC}(u) := \frac{1}{|I_u^+|\,|I \setminus I_u^+|} \sum_{i \in I_u^+} \sum_{j \in |I \setminus I_u^+|} \delta(\hat{x}_{uij} > 0)$$

NOT to **regress** a single value $\hat{x}_{ui}$, but to **classify** $\hat{x}_{ui} - \hat{x}_{uj}$

BPR-MF $BPR = \sum_{(u,i,j)\in D_S} \ln \sigma(\hat{x}_{uij}) - \lambda_\Theta \|\Theta\|^2$

- Training data: $D_S = \{(u,i,j) \mid i \in I_u^+ \wedge j \in I \backslash I_u^+\}$
- The predicted ranking order $\hat{x}_{uij} = \hat{x}_{ui} - \hat{x}_{uj}$

$$\hat{x}_{ui} = \sum_{f=1}^{k} w_{uf}\, h_{if}$$

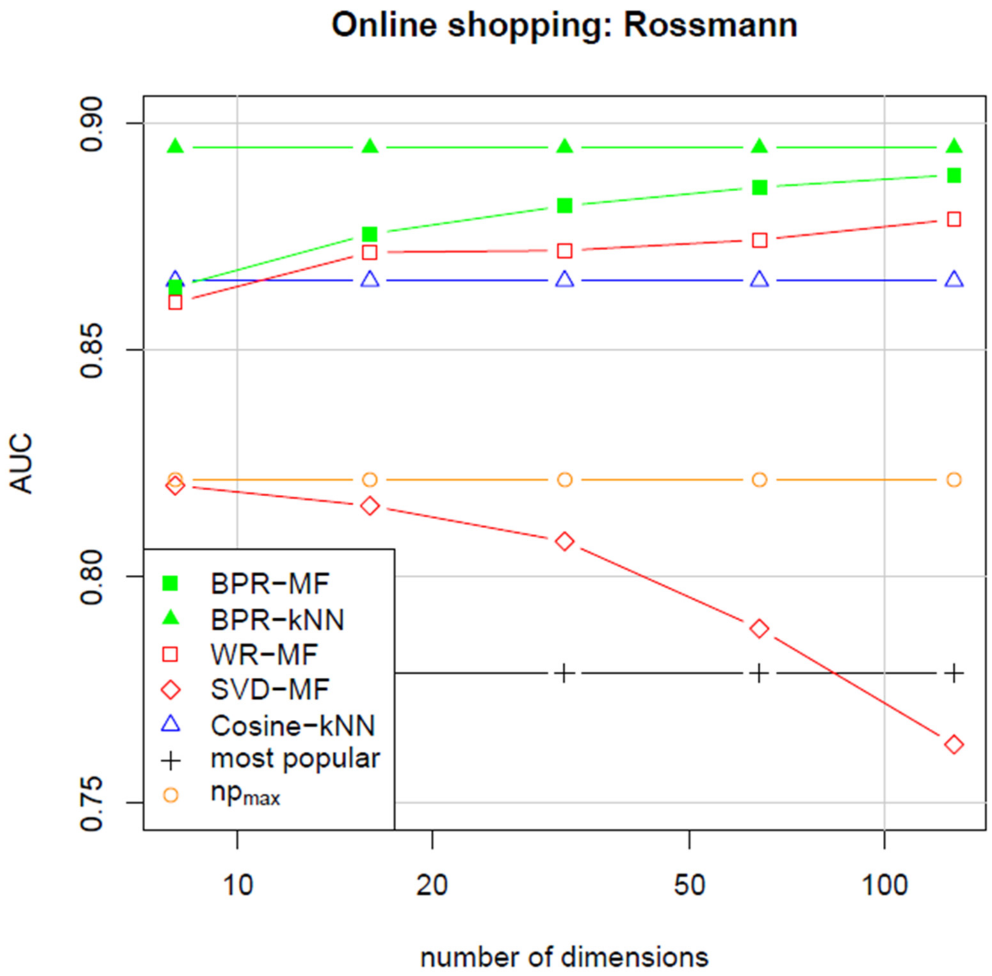
$$\sigma(x) = \frac{1}{1+\exp(-x)}$$

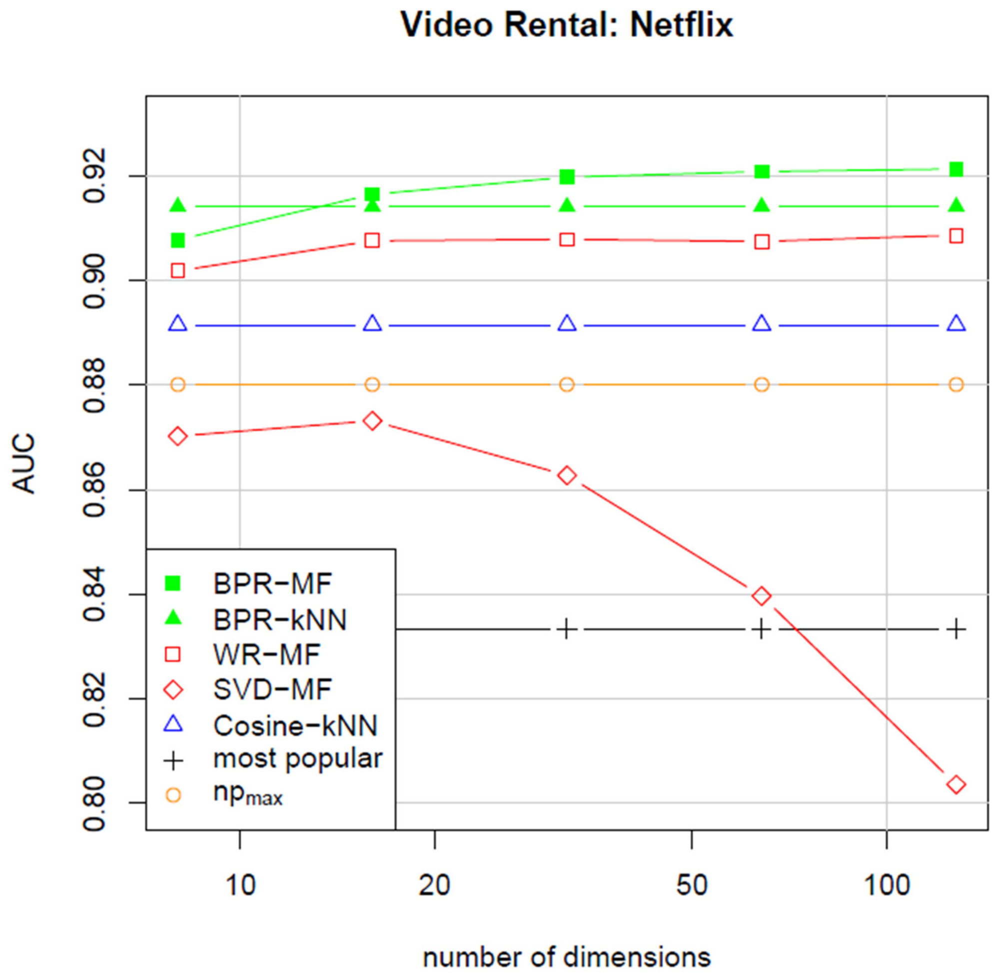
```
# compute x_uij
Wu = W[u].view(1, W[u].size()[0])
x_uij = torch.mv(Wu, H[i]) - torch.mv(Wu, H[j])

# compute e^-x_uij / (1 + e^-x_uij)
exp_x = np.exp(-x_uij)
partial_BPR = exp_x / (1 + exp_x)

# 對第 1 ~ k 個 feature 更新
for f in range(k):
    W[u][f] -= lr * (partial_BPR * (H[i][f] - H[j][f]) + rr * W[u][f])
    H[i][f] -= lr * (partial_BPR * W[u][f] + rr * H[i][f])
    H[j][f] -= lr * (partial_BPR * (-W[u][f]) + rr * H[i][f])
```

$$\frac{\partial}{\partial \theta}\hat{x}_{uij} = \begin{cases} (h_{if} - h_{jf}) & \text{if } \theta = w_{uf}, \\ w_{uf} & \text{if } \theta = h_{if}, \\ -w_{uf} & \text{if } \theta = h_{jf}, \\ 0 & \text{else} \end{cases}$$

https://github.com/leafinity/gradient_dscent_svd/blob/master/bpr.ipynb

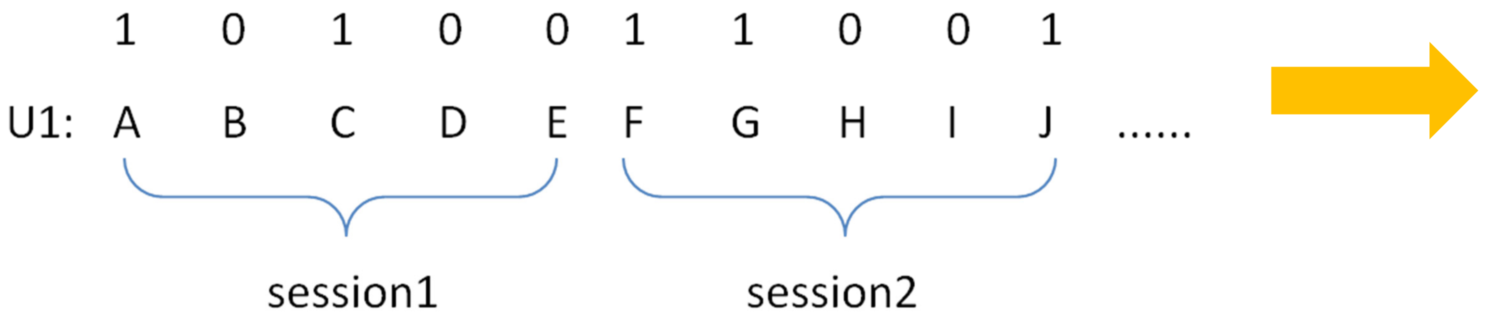
BPR Performance





Short Summary

- BPR is ranking-aware optimization
  - Separate unknown from missing in "0"s
  - Use gradient descent to optimize non-differential AUC
  - Bootstrap sampling for $D_S = \{(u, i, j)\}$
  - Can be combined with different models
- Better selection of D_S
  - E.g., session-based selection
- We will see BPR many times soon



U1,A,B
U1,A,D
U1,A,E
U1,C,B
U1,C,D
U1,C,E
U1,F,H
U1,F,I
U1,G,H
U1,G,I
U1,J,H
U1,J,I

References

- S. Rendle et al. "BPR: Bayesian Personalized Ranking from Implicit Feedback" UAI 2009 2480 cites
- https://medium.com/@radleaf/bpr-and-recommendation-system-3d9a3975c132
- https://blog.csdn.net/sigmeta/article/details/80517828
- https://www.cnblogs.com/wkang/p/10217172.html
- https://www.biaodianfu.com/bpr.html

BPR Packages/Codes

- Case Recommender
  - https://github.com/caserec/CaseRecommender
- **Spotlight** [complete, recommended!!]
  - https://github.com/maciejkula/spotlight
- **LightFM** [recommended!!]
  - https://github.com/lyst/lightfm/
- RecSys tutorial
  - https://github.com/MaurizioFD/RecSys_Course_AT_PoliMi
- **NeuRec** [complete, recommended!!]
  - https://github.com/wubinzzu/NeuRec