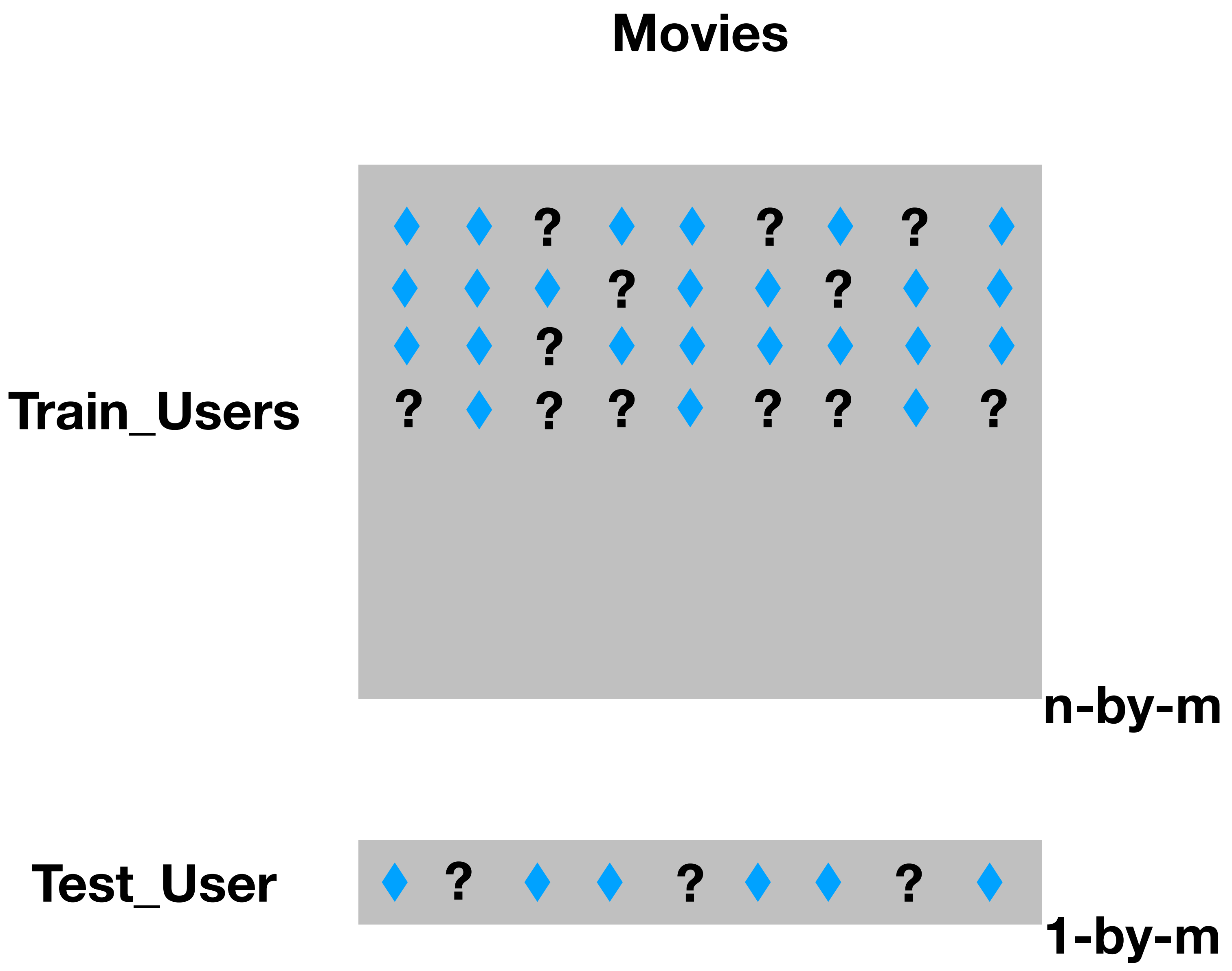


Movies

Train_Users

♦ ♦ ? ♦ ♦ ? ♦ ? ♦
♦ ♦ ♦ ? ♦ ♦ ? ♦ ♦
♦ ♦ ? ♦ ♦ ♦ ♦ ♦ ♦
? ♦ ? ? ♦ ? ? ♦ ?

n-by-m

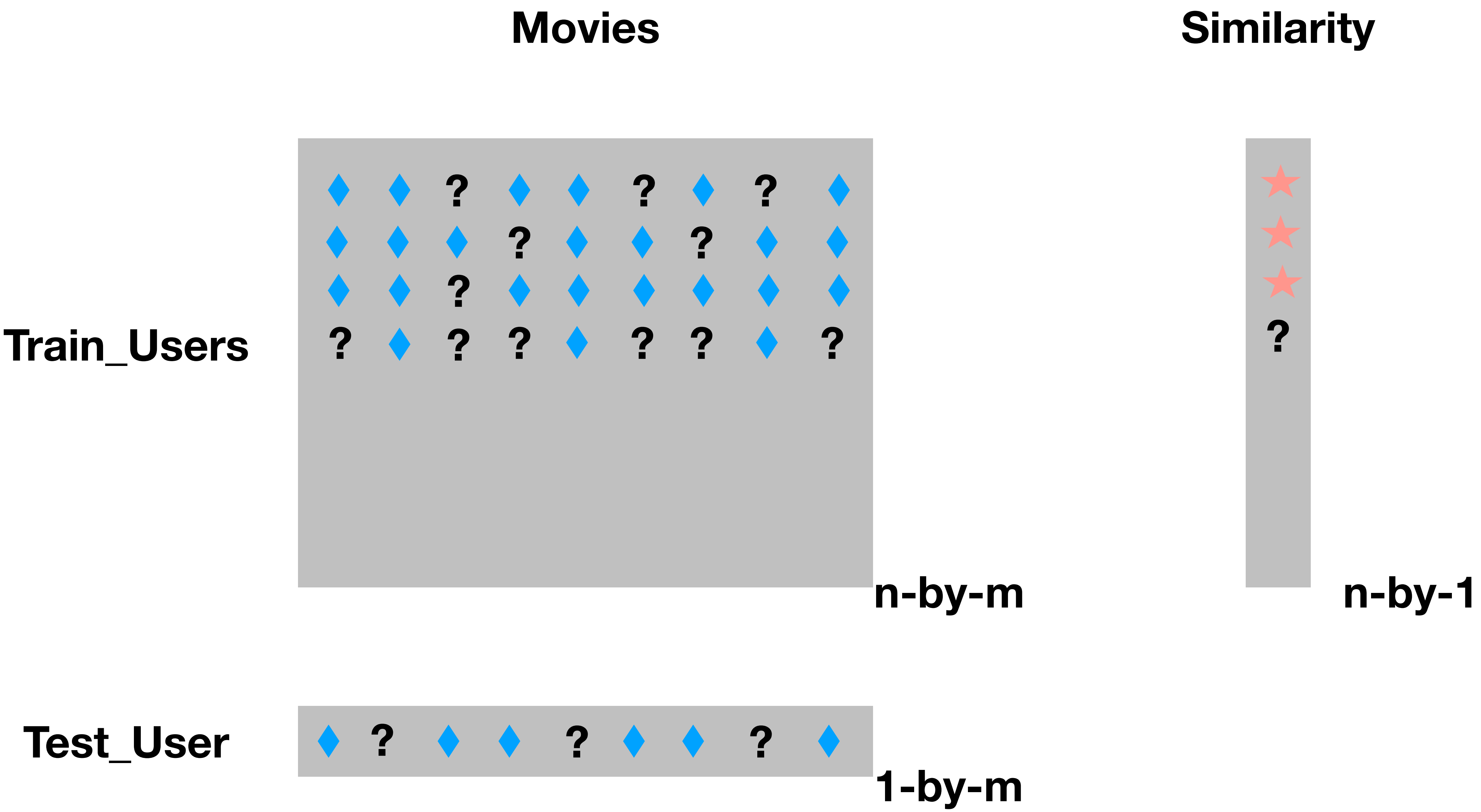
Test_User

♦ ? ♦ ♦ ? ♦ ♦ ? ♦

1-by-m

UBCF

— Normalization: Center each row for both train and test
— Compute similarity (cosine) between test and train
— Find K nearest users (i.e., users with highest similarities)
— Compute the weighted average of those K users (K = 20)



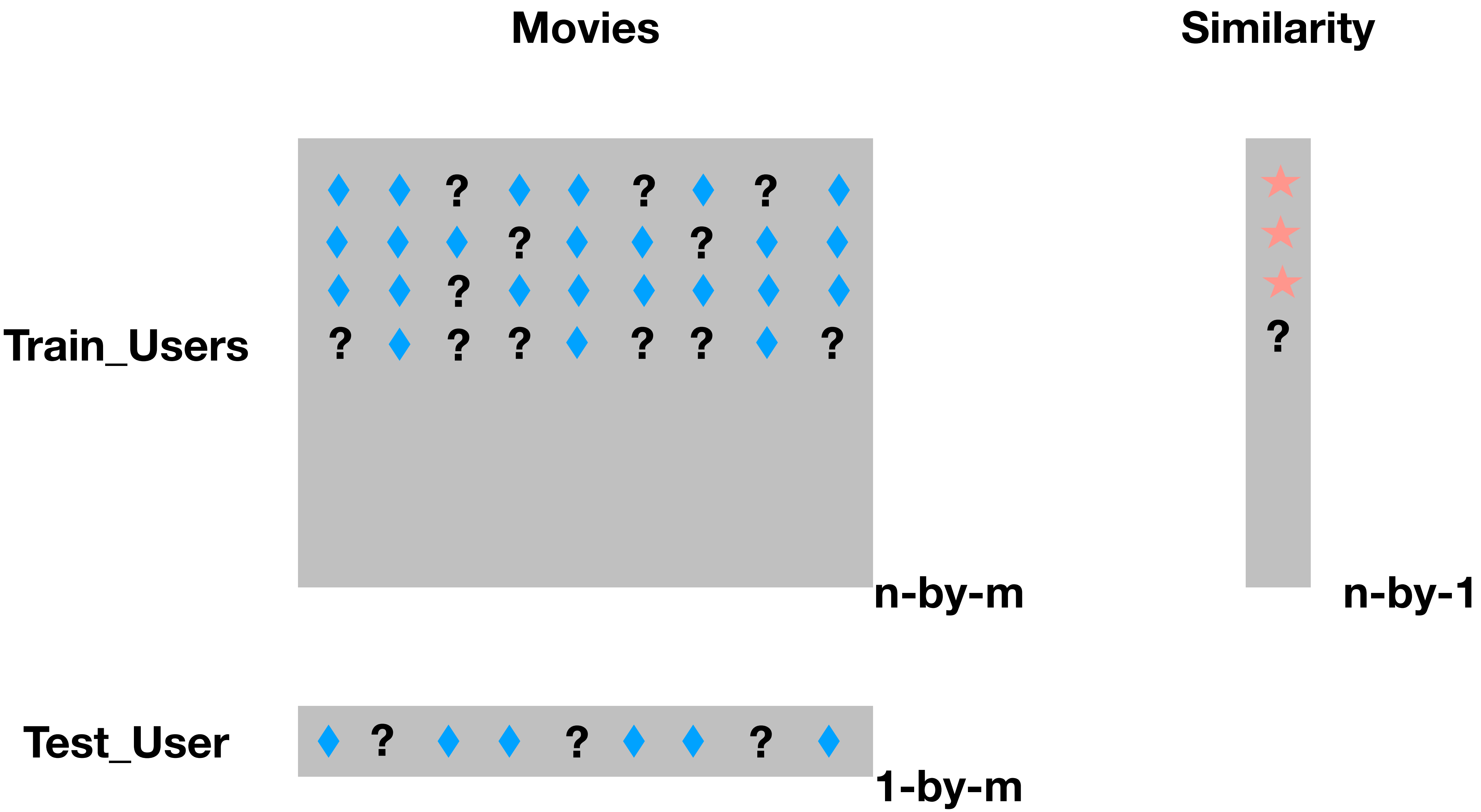
UBCF

— Normalization: Center each row for both train and test

— Compute similarity (cosine) between test and train

— Find K nearest users (i.e., users with highest similarities)

— Compute the weighted average of those K users (K = 20)



UBCF

Be careful with NAs

- — Normalization: Center each row for both train and test
- — Compute similarity (cosine) between test and train
- — Find K nearest users (i.e., users with highest similarities)
- — Compute the weighted average of those K users (K = 20)

- — Some additional steps before output your prediction

- Normalization: Center each row for both train and test

```
> train = Rmat[1:500, ]
> test = Rmat[501, ]

> data = as(train, "matrix")
> user.means = rowMeans(data, na.rm = TRUE)
> data = data - user.means

> newdata = as(Rmat[501, ], "matrix")
> newuser.mean = mean(newdata, na.rm = TRUE)
> newdata = newdata - newuser.mean
```

- Normalization: Center each row for both train and test

```
> train = Rmat[1:500, ]
> test = Rmat[501, ]

> data = as(train, "matrix")
> user.means = rowMeans(data, na.rm = TRUE)
> data = data - user.means

> newdata = as(Rmat[501, ], "matrix")
> newuser.mean = mean(newdata, na.rm = TRUE)
> newdata = newdata - newuser.mean
```

$$\frac{\sum_{i \in S} x_i y_i}{\sqrt{\sum_{i \in S} x_i^2}\sqrt{\sum_{i \in S} y_i^2}}$$

$$S = \{i : x_i \text{ and } y_i \neq NA\}$$

- Compute similarity (cosine)

```
> sim = rep(0, dim(data)[1])
> for(i in 1:length(sim))
+   {
+   tmp.y = as.vector(newdata)
+   ind.y = which(!is.na(tmp.y))
+   tmp.x = data[i, ]
+   ind.x = which(!is.na(tmp.x))
+   ind  = intersect(ind.x, ind.y)
+   if (length(ind) > 0){
+     tmp.x = tmp.x[ind]
+     tmp.y = tmp.y[ind]
+     sim[i] = sum(tmp.x * tmp.y) / sqrt(sum(tmp.x^2) * sum(tmp.y^2))
+   }
+   }
```

– Normalization: Center each row for both train and test

```
> train = Rmat[1:500, ]
> test = Rmat[501, ]

> data = as(train, "matrix")
> user.means = rowMeans(data, na.rm = TRUE)
> data = data - user.means

> newdata = as(Rmat[501, ], "matrix")
> newuser.mean = mean(newdata, na.rm = TRUE)
> newdata = newdata - newuser.mean
```

$$\frac{\sum_{i \in S} x_i y_i}{\sqrt{\sum_{i \in S} x_i^2} \sqrt{\sum_{i \in S} y_i^2}}$$

$$S = \{i : x_i \text{ and } y_i \neq NA\}$$

– Compute similarity (cosine)

```
> sim = rep(0, dim(data)[1])
> for(i in 1:length(sim))
+   {
+   tmp.y = as.vector(newdata)
+   ind.y = which(!is.na(tmp.y))
+   tmp.x = data[i, ]
+   ind.x = which(!is.na(tmp.x))
+   ind  = intersect(ind.x, ind.y)
+   if (length(ind) > 0){
+     tmp.x = tmp.x[ind]
+     tmp.y = tmp.y[ind]
+     sim[i] = sum(tmp.x * tmp.y) / sqrt(sum(tmp.x^2) * sum(tmp.y^2))
+   }
+   }
> sim = (1 + sim)/2
```

Don't forget the transformation

– Normalization: Center each row for both train and test

```
> train = Rmat[1:500, ]
> test = Rmat[501, ]

> data = as(train, "matrix")
> user.means = rowMeans(data, na.rm = TRUE)
> data = data - user.means

> newdata = as(Rmat[501, ], "matrix")
> newuser.mean = mean(newdata, na.rm = TRUE)
> newdata = newdata - newuser.mean
```

$$\frac{\sum_{i \in S} x_i y_i}{\sqrt{\sum_{i \in S} x_i^2}\sqrt{\sum_{i \in S} y_i^2}}$$

$$S = \{i : x_i \text{ and } y_i \neq NA\}$$

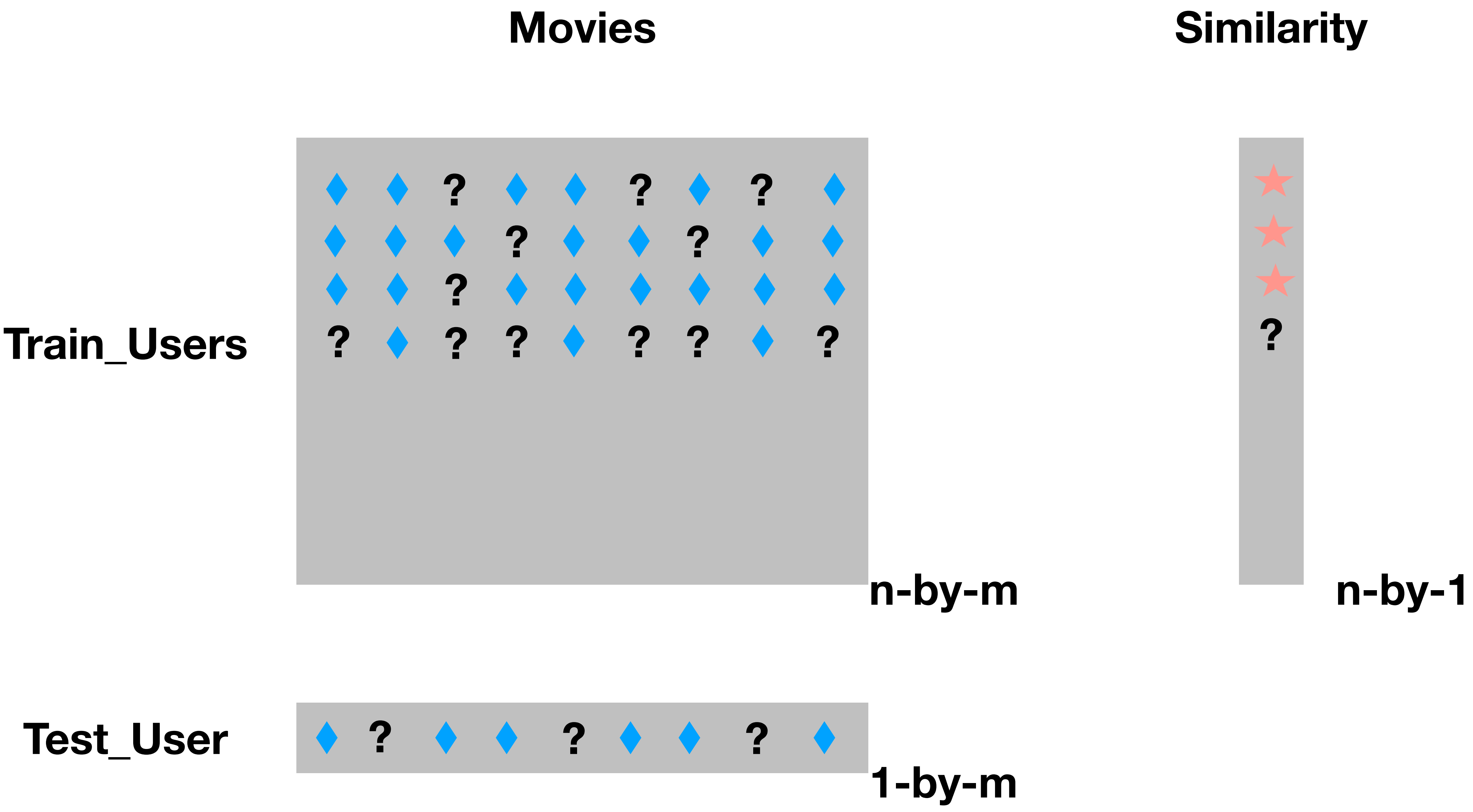
– Compute similarity (cosine)

```
> sim = rep(0, dim(data)[1])
> for(i in 1:length(sim))
+   {
+   tmp.y = as.vector(newdata)
+   ind.y = which(!is.na(tmp.y))
+   tmp.x = data[i, ]
+   ind.x = which(!is.na(tmp.x))
+   ind  = intersect(ind.x, ind.y)
+   if (length(ind) > 0){
+     tmp.x = tmp.x[ind]
+     tmp.y = tmp.y[ind]
+     sim[i] = sum(tmp.x * tmp.y) / sqrt(sum(tmp.x^2) * sum(tmp.y^2))
+   }
+   }
> sim = (1 + sim)/2
```

Don’t forget the transformation

Alternative command

```
> sim1 = proxy::simil(data, newdata, method = "cosine")
> sim1 = (1 + sim1)/2
> sum((sim - sim1)^2)
[1] 3.37577e-29
```



Movies

Similarity

Train_Users

n-by-m

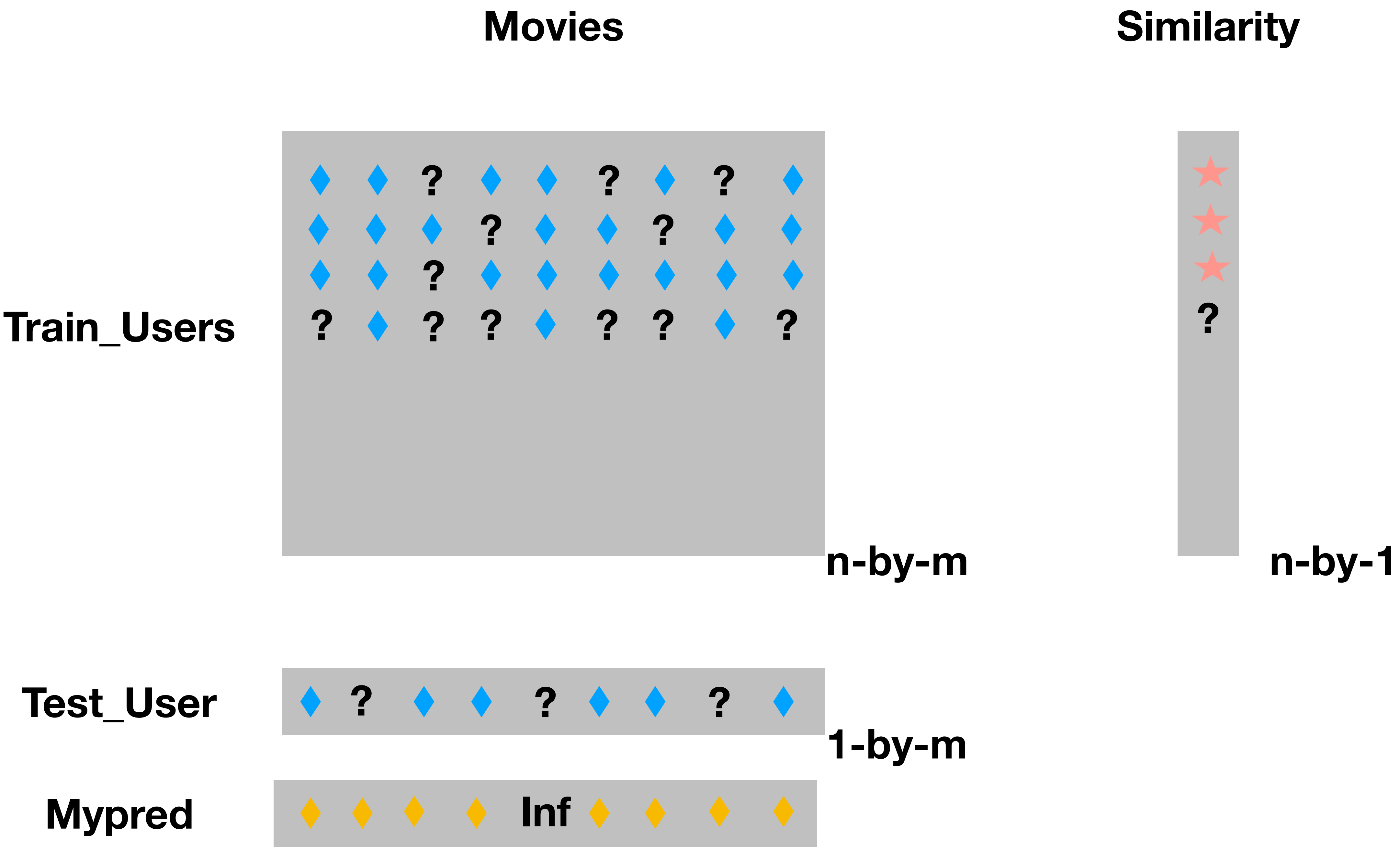
n-by-1

Test_User

1-by-m

– Compute the weighted average of those K users (K = 20)

$$\text{mypred}[j] = \frac{\sum_{i \in S} s_i r_{ij}}{\sum_{i \in S} s_i} \qquad S = \{i : s_i \text{ and } r_{ij} \text{ not NA}\}$$



Some additional steps before output your prediction

1. Add back the mean of the test_user
2. Set infinite values to NA
3. Set movies watched by the test_user to NA