Building Streaming Recommendation Engines on Spark

Rui Vieira
rui@redhat.com

Overview

- Collaborative Filtering
    - Batch Alternating Least Squares (ALS)
    - Streaming ALS
- Apache Spark
    - Distributed Streaming ALS
- OpenShift deployment

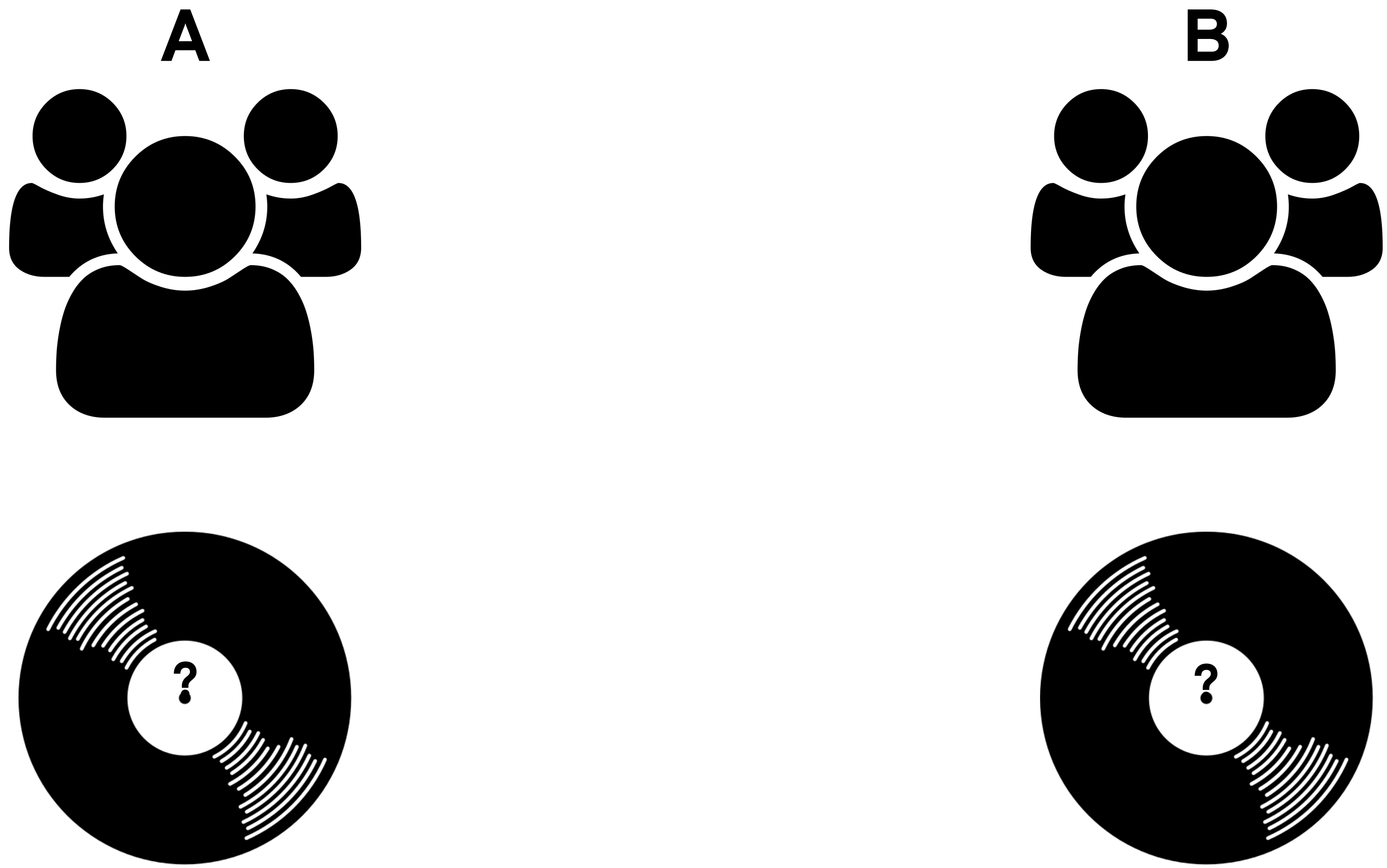
Collaborative Filtering

Collaborative Filtering

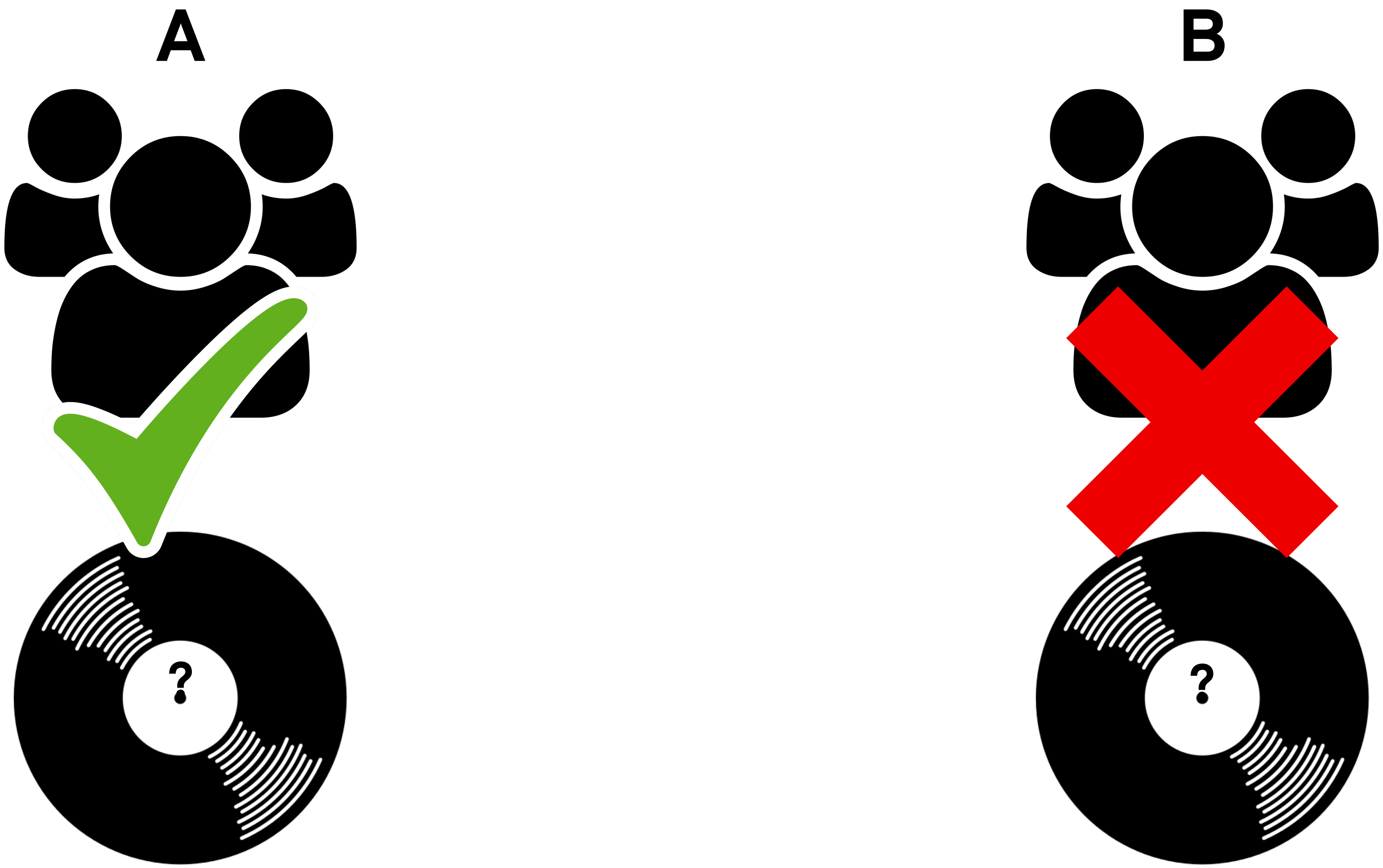
- Users, products and ratings
  - (user, product) $\longmapsto$ rating
- Collaborative
- “Filtering”

Collaborative Filtering

Collaborative Filtering



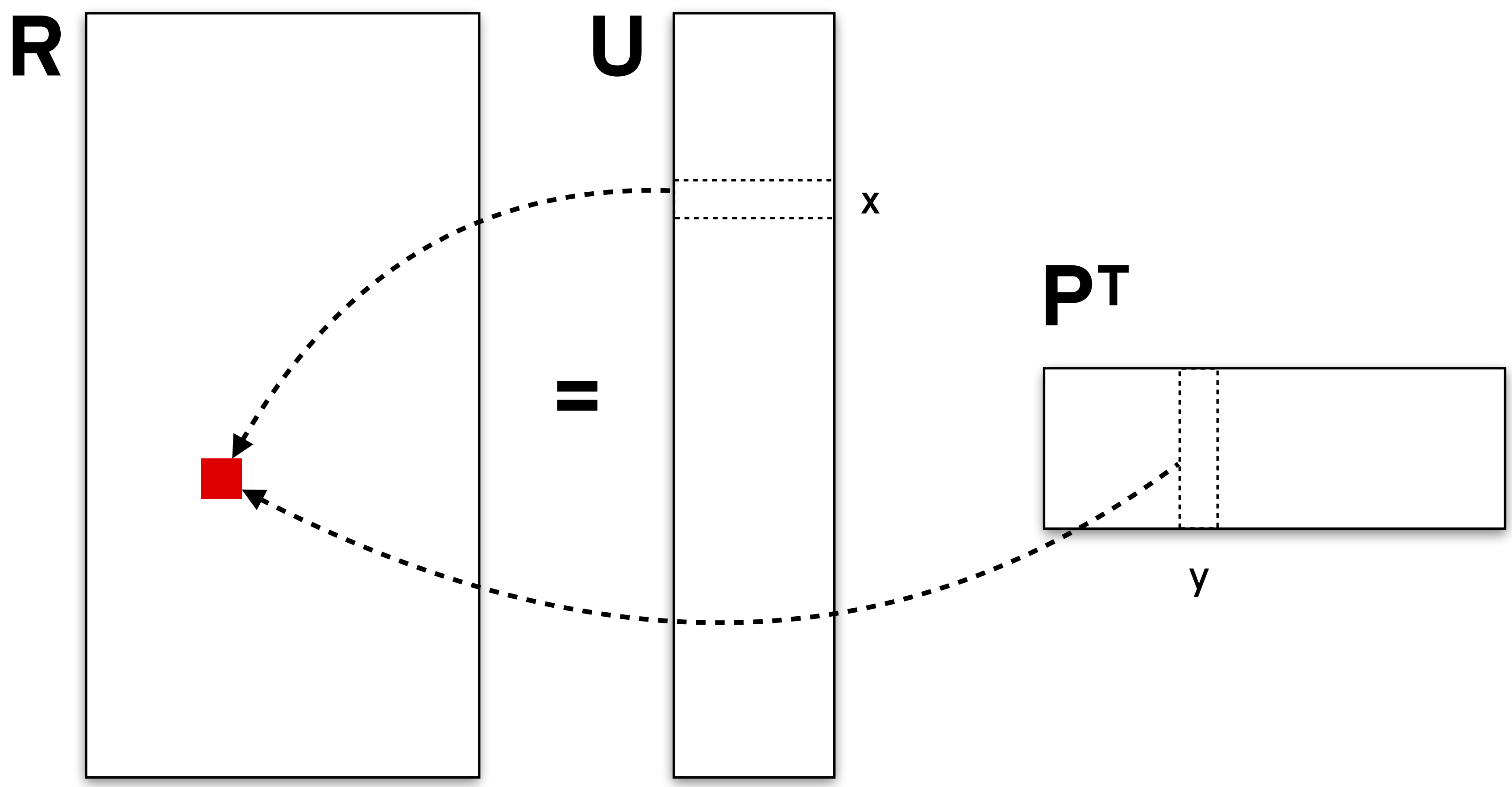
Collaborative Filtering



Alternating Least Squares

$$
R = \begin{array}{c}
\begin{array}{ccccc} \text{user 1} & \text{user 2} & \text{user 3} & \cdots & \text{user N} \end{array} \\
\left[\begin{array}{ccccc}
1 & 4.5 & ? & \cdots & 3 \\
? & 3 & 3 & \cdots & 4 \\
5 & 3 & ? & \cdots & ? \\
\vdots & \vdots & \vdots & \ddots & \vdots \\
2 & 4 & 1 & \cdots & ?
\end{array}\right]
\end{array}
\begin{array}{l}
\\
\text{product 1} \\
\text{product 2} \\
\text{product 3} \\
\vdots \\
\text{product M}
\end{array}
$$

Alternating Least Squares



$$\hat{r}_{x,y} = \mathsf{U}_x \mathsf{P}_y^T$$

Batch ALS

$$\textsf{loss} = \sum_{x,y} \Bigg(\underbrace{r_{x,y} - \hat{r}_{x,y}}_{\epsilon_{x,y}} \Bigg)^2 + \lambda_x \sum_{x} \|\mathsf{U}_x\|^2 + \lambda_y \sum_{y} \|\mathsf{P}_y\|^2$$

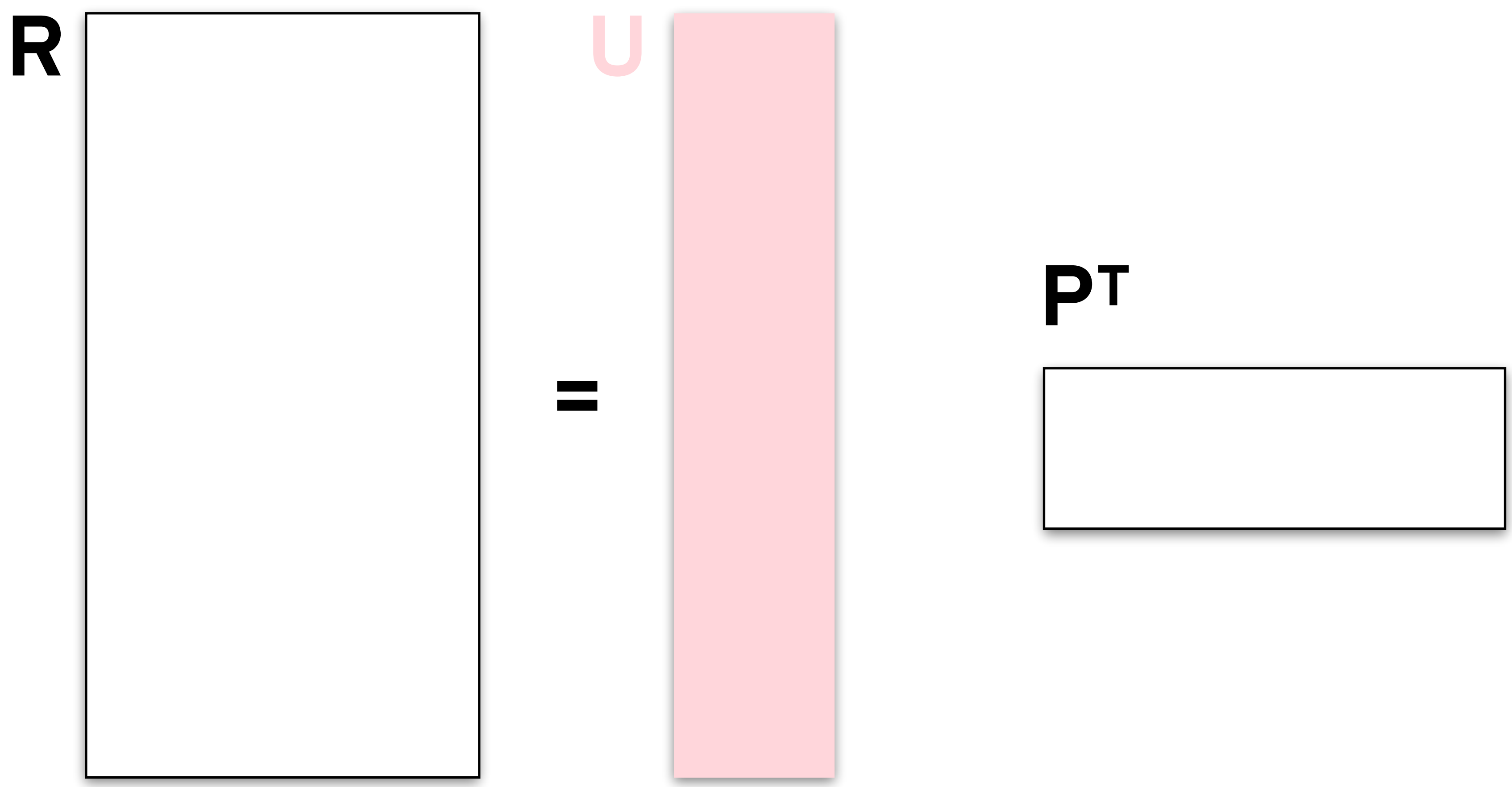
Batch ALS

$$\mathsf{loss} = \sum_{x,y} \left(\underbrace{r_{x,y} - \hat{r}_{x,y}}_{\epsilon_{x,y}} \right)^2 + \lambda_x \sum_x \|\mathsf{U}_x\|^2 + \lambda_y \sum_y \|\mathsf{P}_y\|^2$$

(minimize)

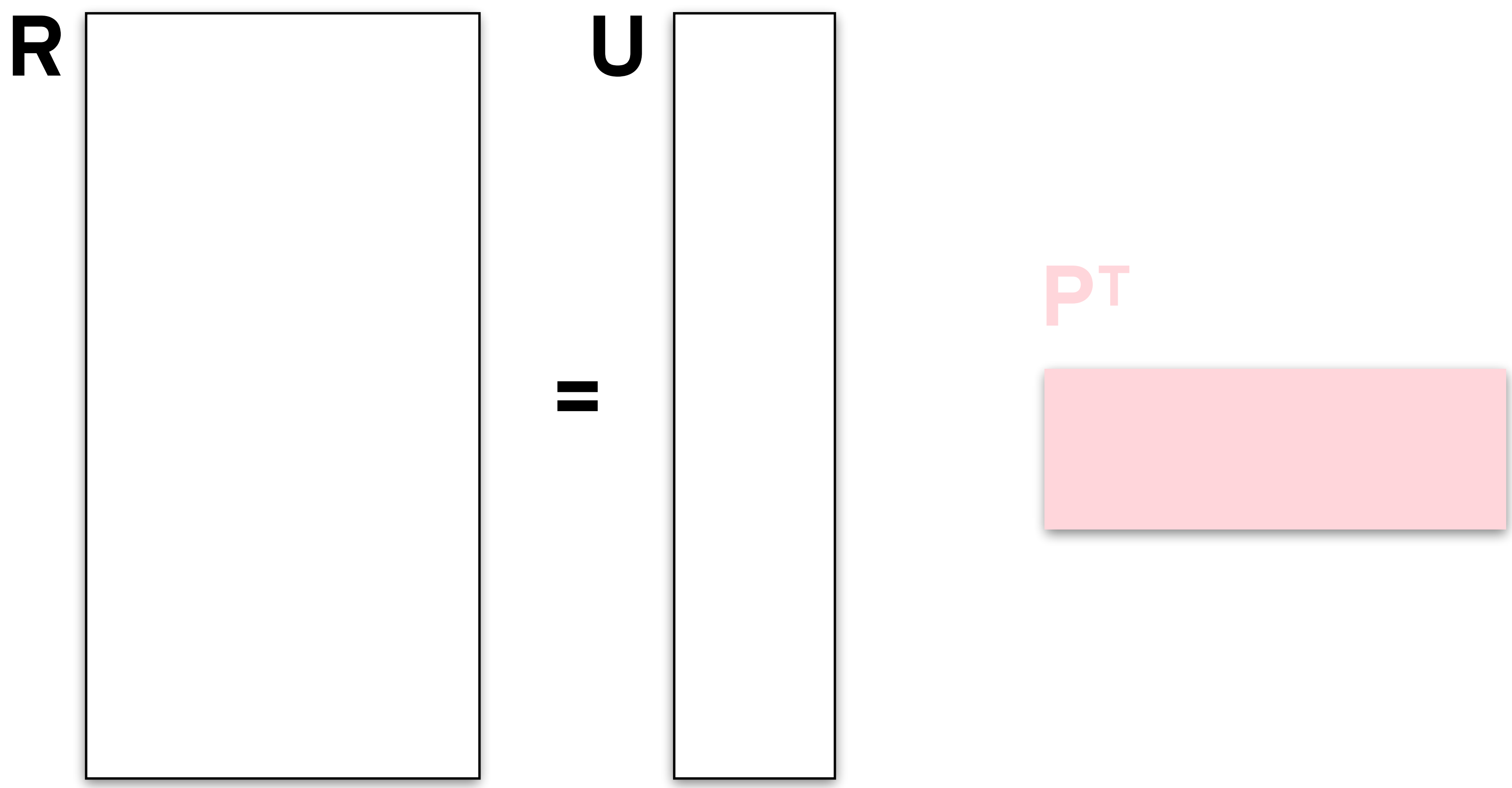
$$\frac{\partial \mathsf{loss}}{\partial \mathsf{U}_x} = 0, \qquad \frac{\partial \mathsf{loss}}{\partial \mathsf{P}_y} = 0$$

Alternating Least Squares



$$\mathsf{P}_y = r_y \mathsf{X} \left(X^T X + \lambda_y \mathsf{I} \right)^{-1}$$

Alternating Least Squares



$$\mathrm{U}_x = r_x \mathrm{Y} \left(Y^T Y + \lambda_x \mathrm{I} \right)^{-1}$$

Alternating Least Squares

$$R = \begin{array}{c} \\ \left[\begin{array}{ccccc} \text{user 1} & \text{user 2} & \text{user 3} & \cdots & \text{user N} \\ 1 & 4.5 & \color{red}{3.8} & \cdots & 3 \\ \color{red}{3.2} & 3 & 3 & \cdots & 4 \\ 5 & 3 & \color{red}{3.4} & \cdots & \color{red}{3.1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2 & 4 & 1 & \cdots & \color{red}{2.7} \end{array}\right] \end{array} \begin{array}{c} \text{product 1} \\ \text{product 2} \\ \text{product 3} \\ \vdots \\ \text{product M} \end{array}$$

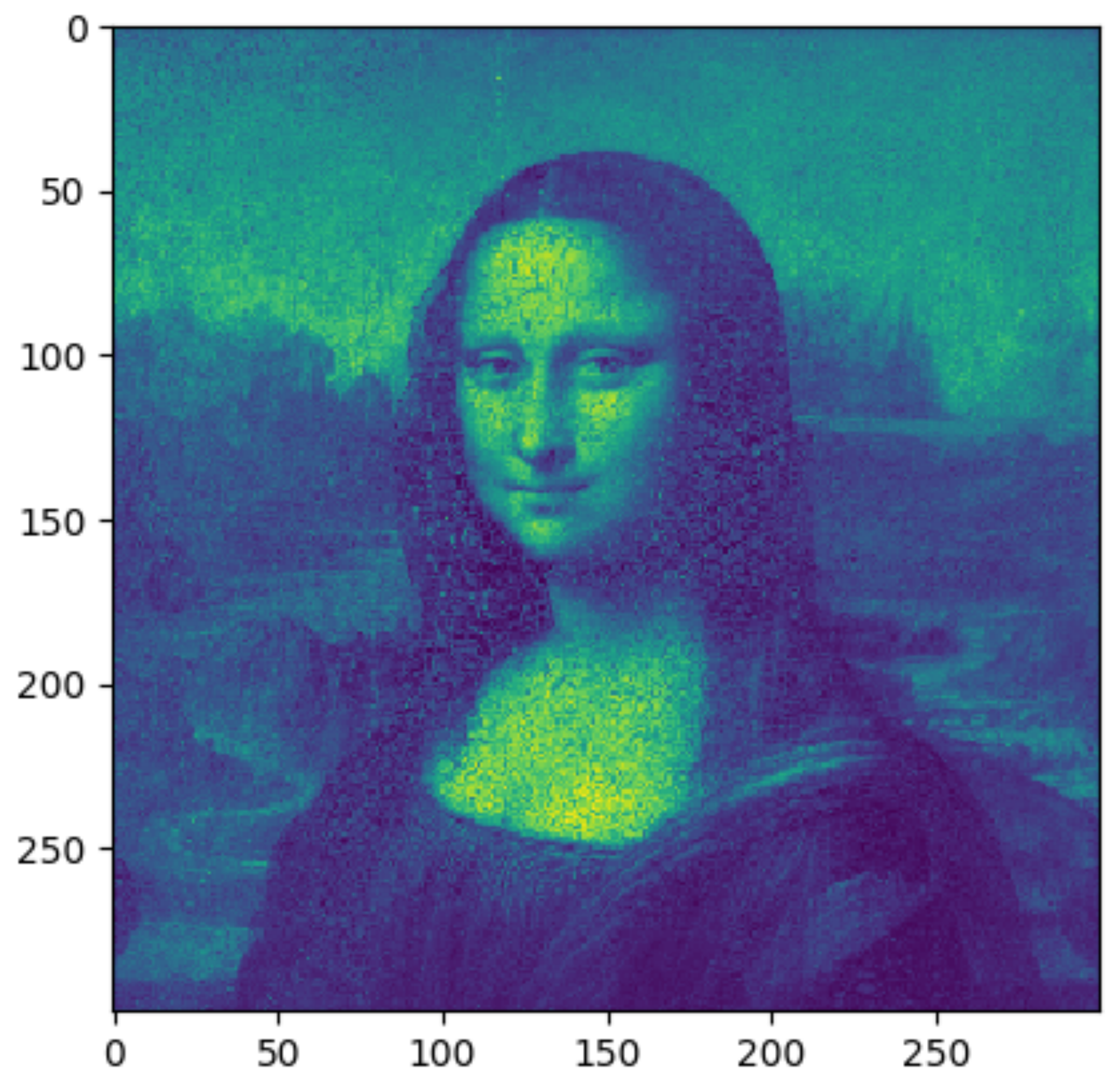
Batch ALS

	1	2	3	4	... 300
1	70	82	60	54	65
2	70	86	68	67	72
3	96	103	82	82	77
4	90	87	68	93	82
... 300	38	48	44	51	35

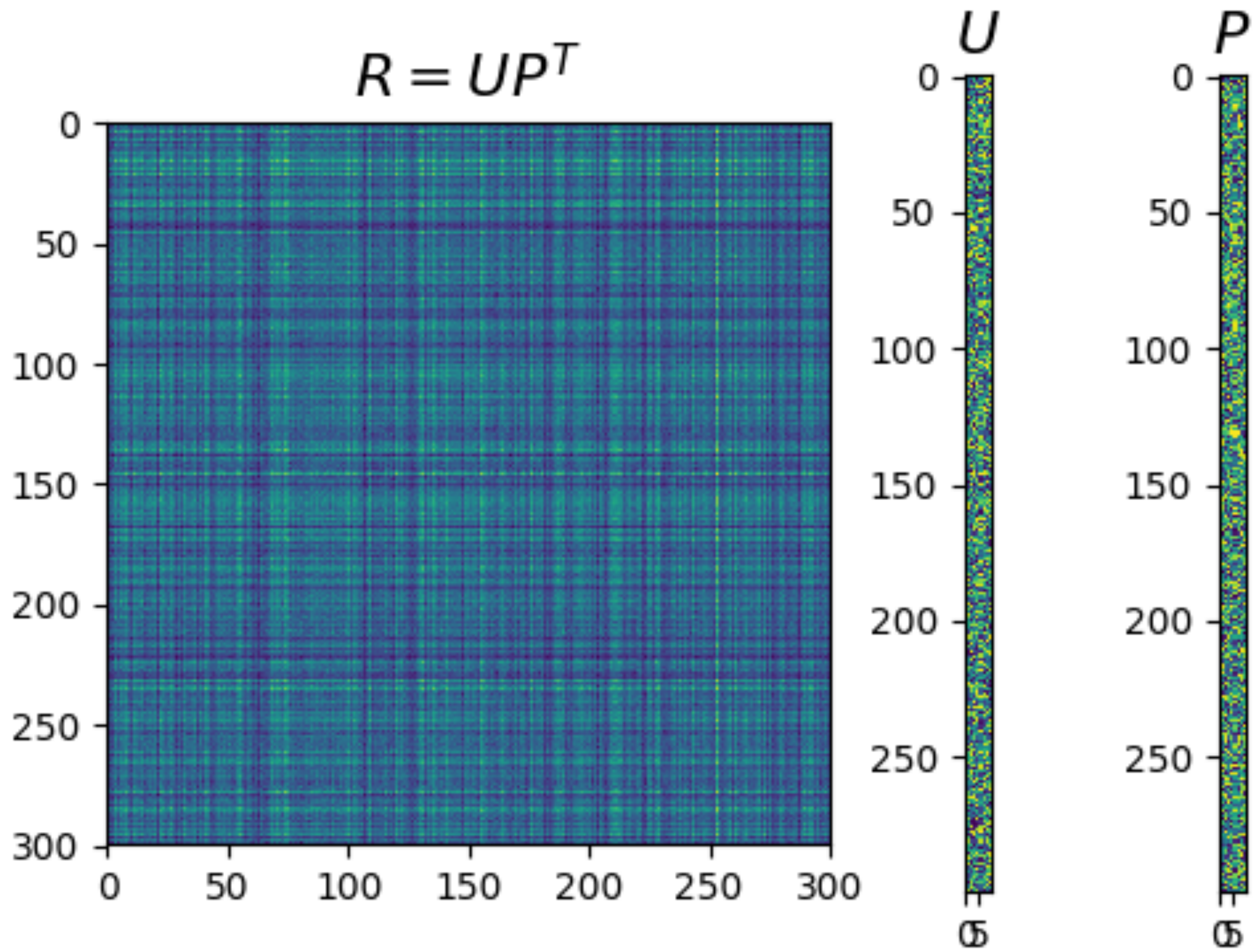
Batch ALS

	1	2	3	4	... 300
1	70	82	60	54	65
2	70	86	68	67	72
3	96	103	82	82	77
4	90	87	68	93	82
... 300	38	48	44	51	35

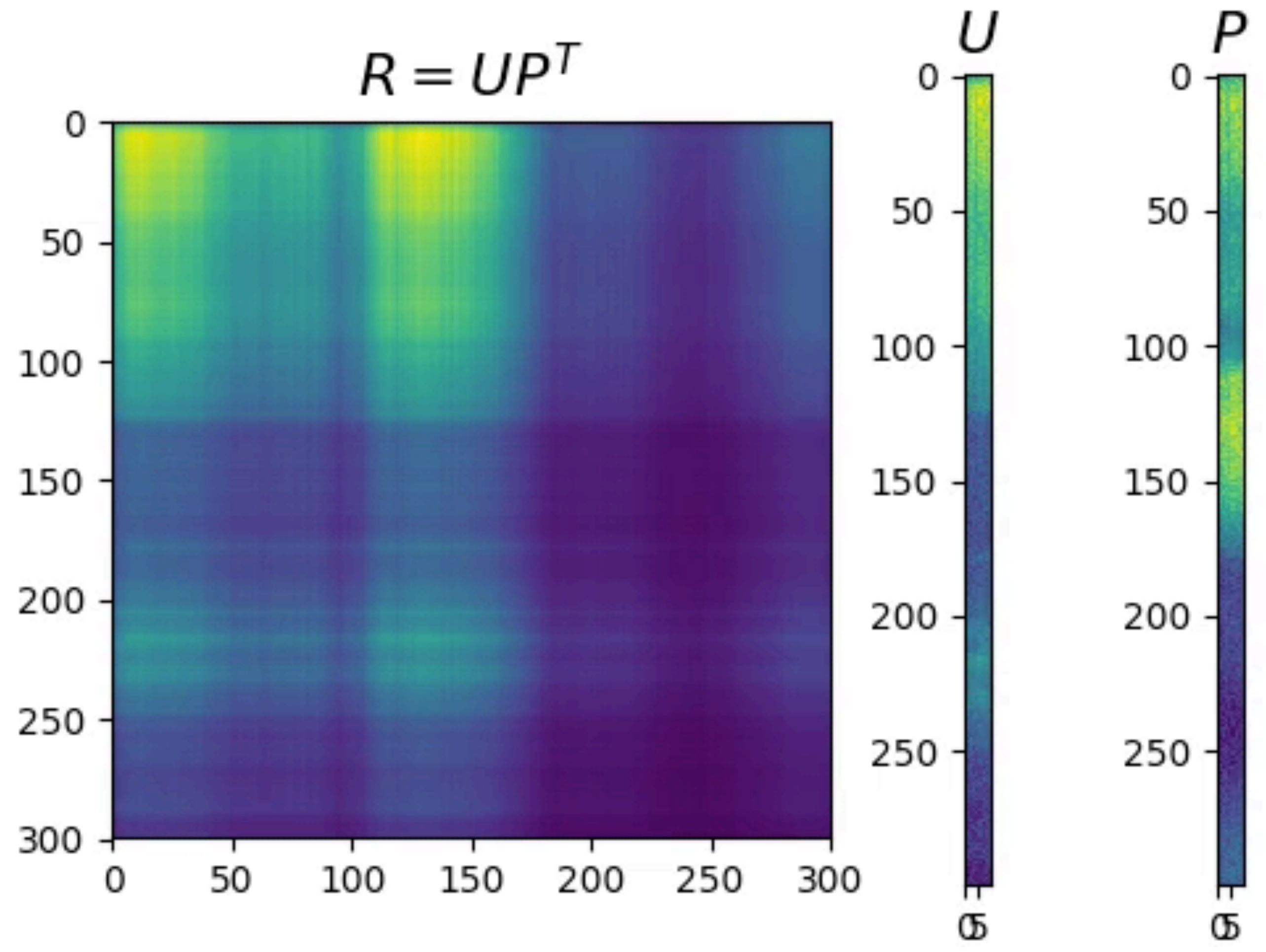
Batch ALS



Batch ALS

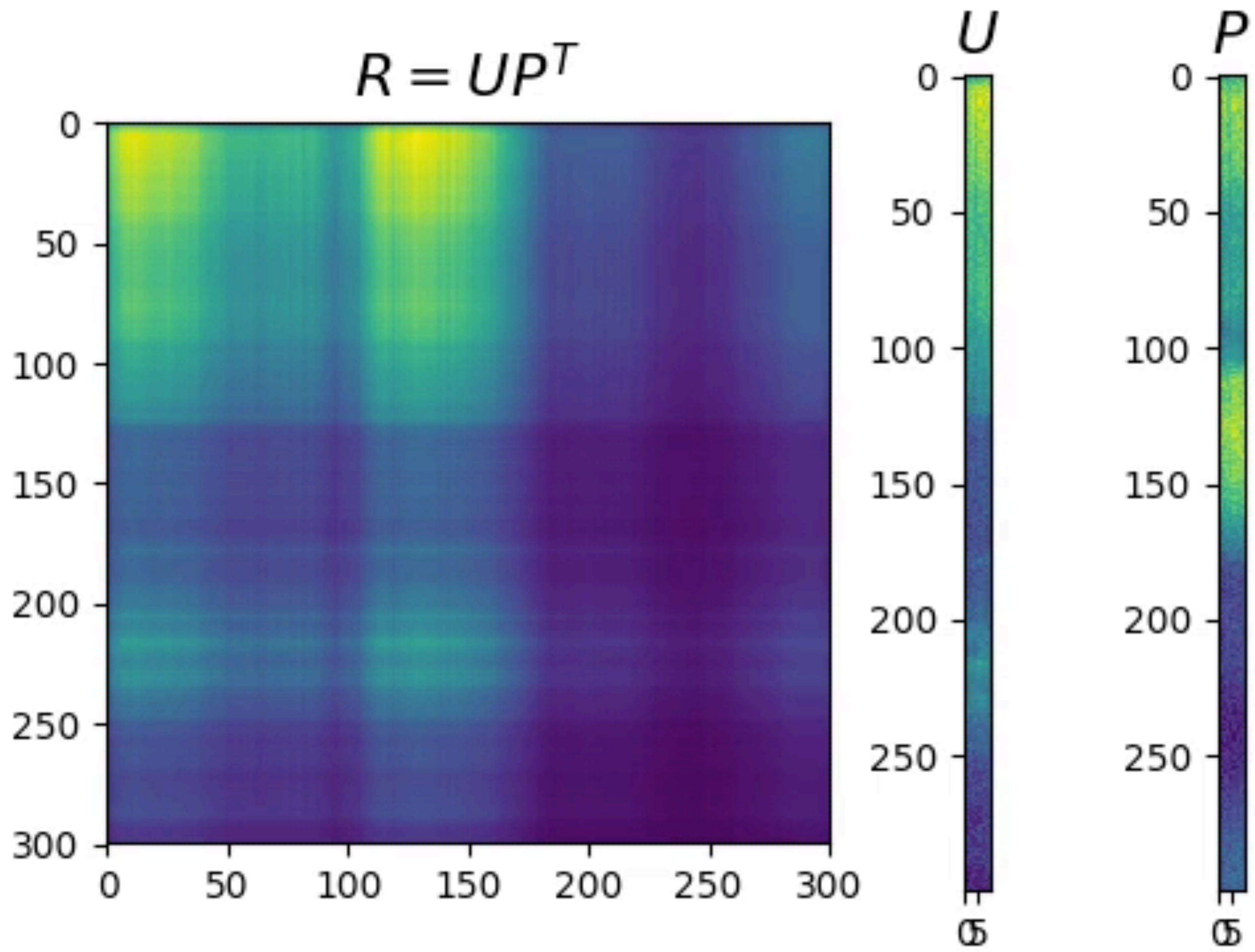


Batch ALS



$R = UP^T$
0
50
100
150
200
250
300
0
50
100
150
200
250
300
U
0
50
100
150
200
250
0
5
P
0
50
100
150
200
250
0
5

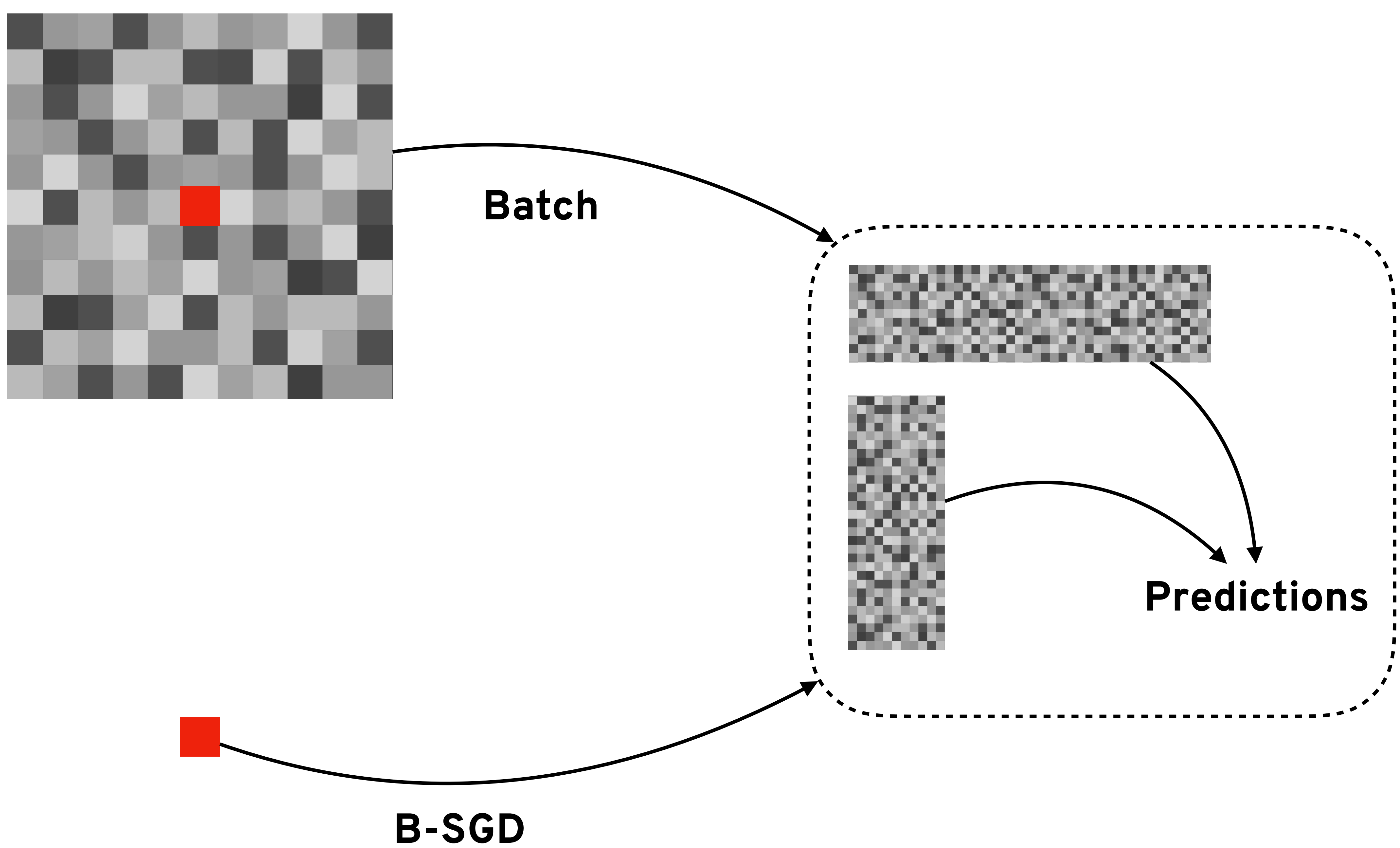
Batch ALS



Streaming ALS

- Can we update the model with a data stream?
- Stochastic Gradient Descent (SGD)
  - Bias SGD (B-SGD)

Streaming ALS



Streaming ALS

$$b_{x,y} = \mu + b_x + b_y$$

$$\hat{r}_{x,y} = \mu + b_x + b_y + \mathsf{U}_x \cdot \mathsf{P}_y^T$$

Streaming ALS

$$\hat{r}_{x,y} = \mu + b_x + b_y + \mathsf{U}_x \cdot \mathsf{P}_y^T$$

$$\textsf{loss} = \sum_{x,y} \left(\underbrace{r_{x,y} - \hat{r}_{x,y}}_{\epsilon_{x,y}} \right)^2 + \cdots$$

Streaming ALS

bias

$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x \right)$$

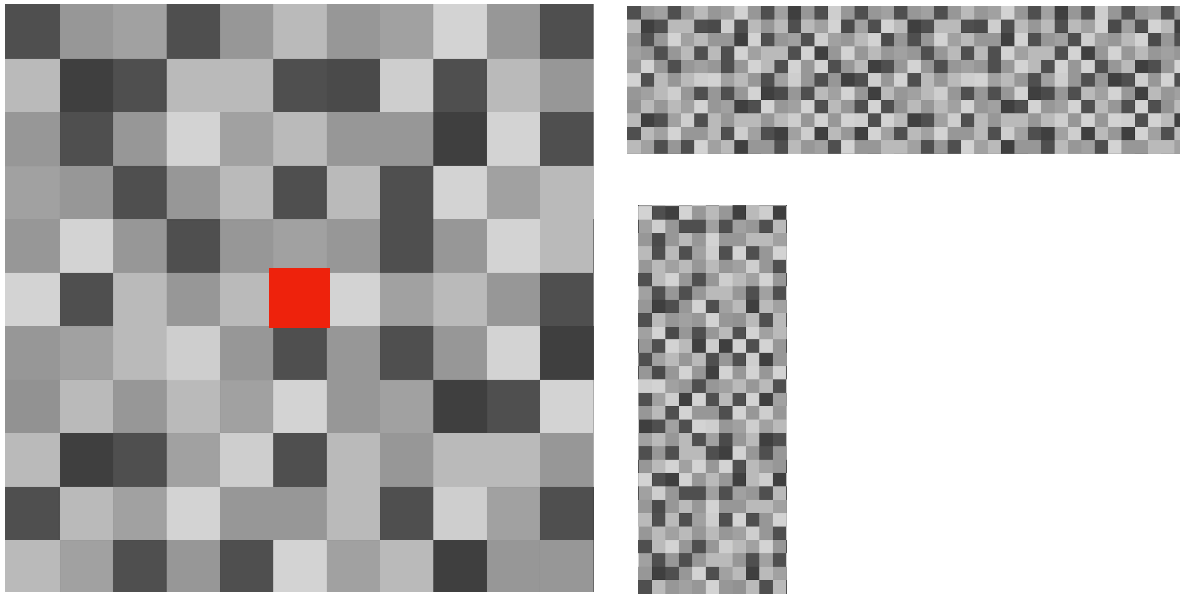
$$b_y \leftarrow b_y + \gamma \left(\epsilon_{x,y} - \lambda_y b_y \right)$$

factors

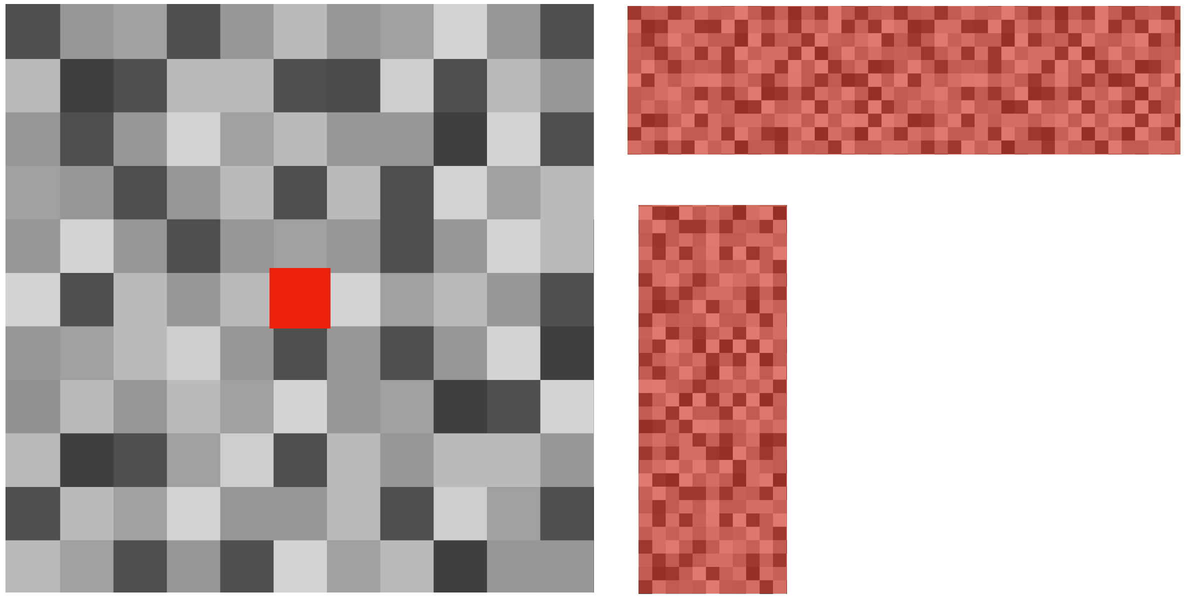
$$\mathsf{U}_x \leftarrow \mathsf{U}_x + \gamma \left(\epsilon_{x,y} \mathsf{P}_y - \lambda'_x \mathsf{U}_x \right)$$

$$\mathsf{P}_y \leftarrow \mathsf{P}_y + \gamma \left(\epsilon_{x,y} \mathsf{U}_x - \lambda'_y \mathsf{P}_y \right)$$

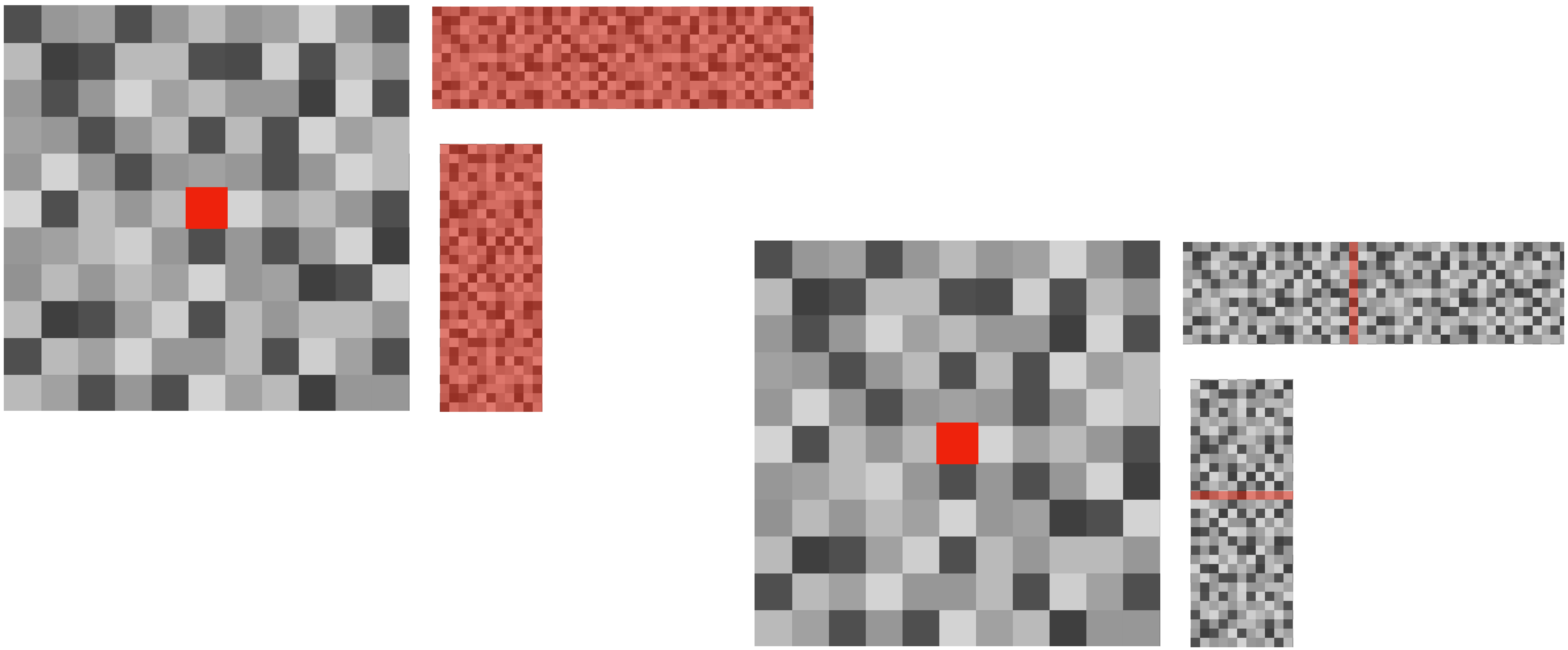
Streaming ALS



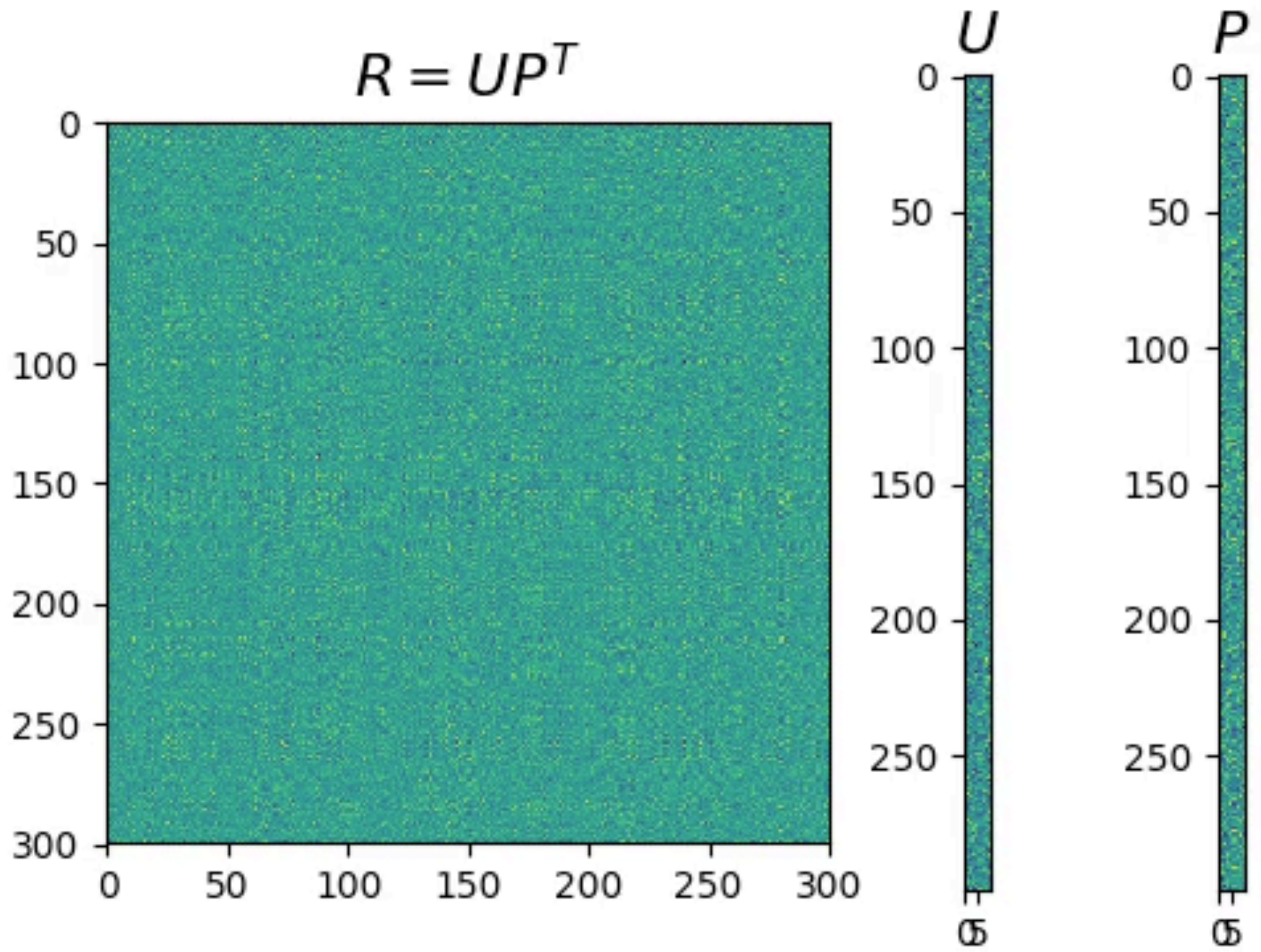
Streaming ALS



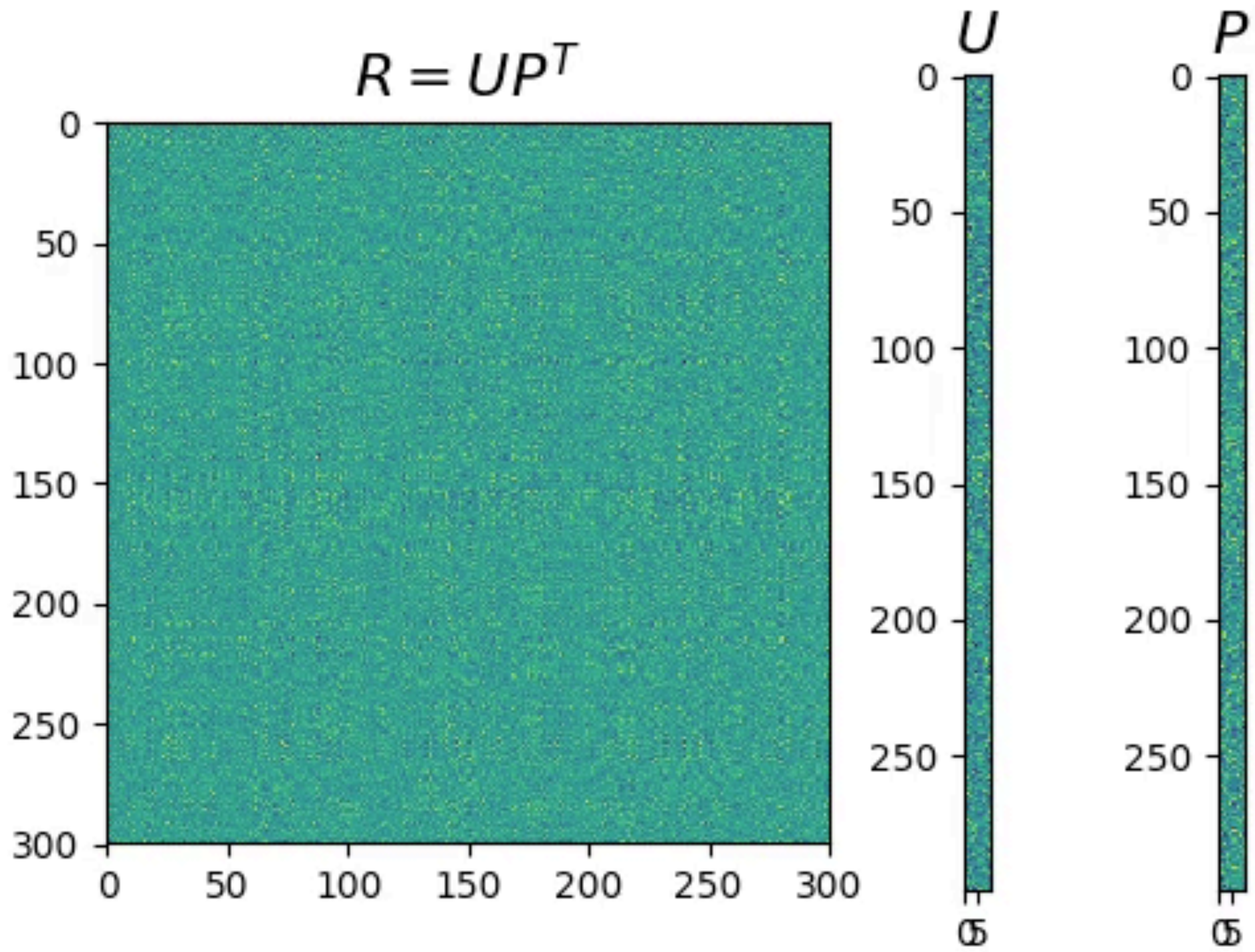
Streaming ALS



Streaming ALS

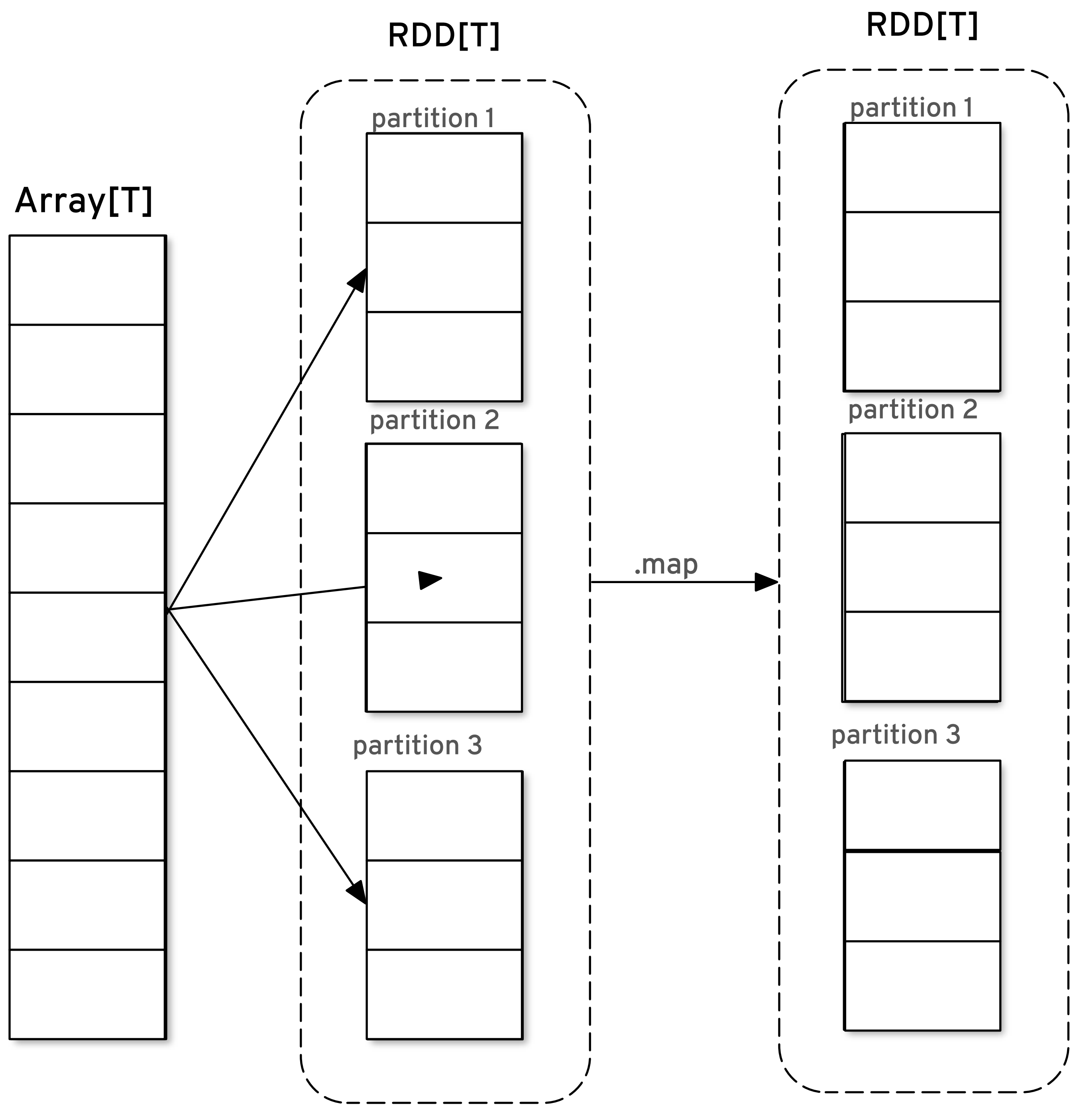


Streaming ALS



Apache Spark

Apache Spark



MLlib ALS

```
val model = ALS.train(ratings, rank, iterations, lambda)
```

MLlib ALS

```
val model = ALS.train(ratings, rank, iterations, lambda)
```

```
case class Rating(int user, int product, double rating)

val ratings: RDD[Rating]
```

MLlib ALS

```
val model = ALS.train(ratings, rank, iterations, lambda)
```

```
case class Rating(int user, int product, double rating)

val ratings: RDD[Rating]

val rank: int
```

MLlib ALS

```
val model = ALS.train(ratings, rank, iterations, lambda)
```

```
case class Rating(int user, int product, double rating)

val ratings: RDD[Rating]

val rank: int

val iterations: int
```

MLlib ALS

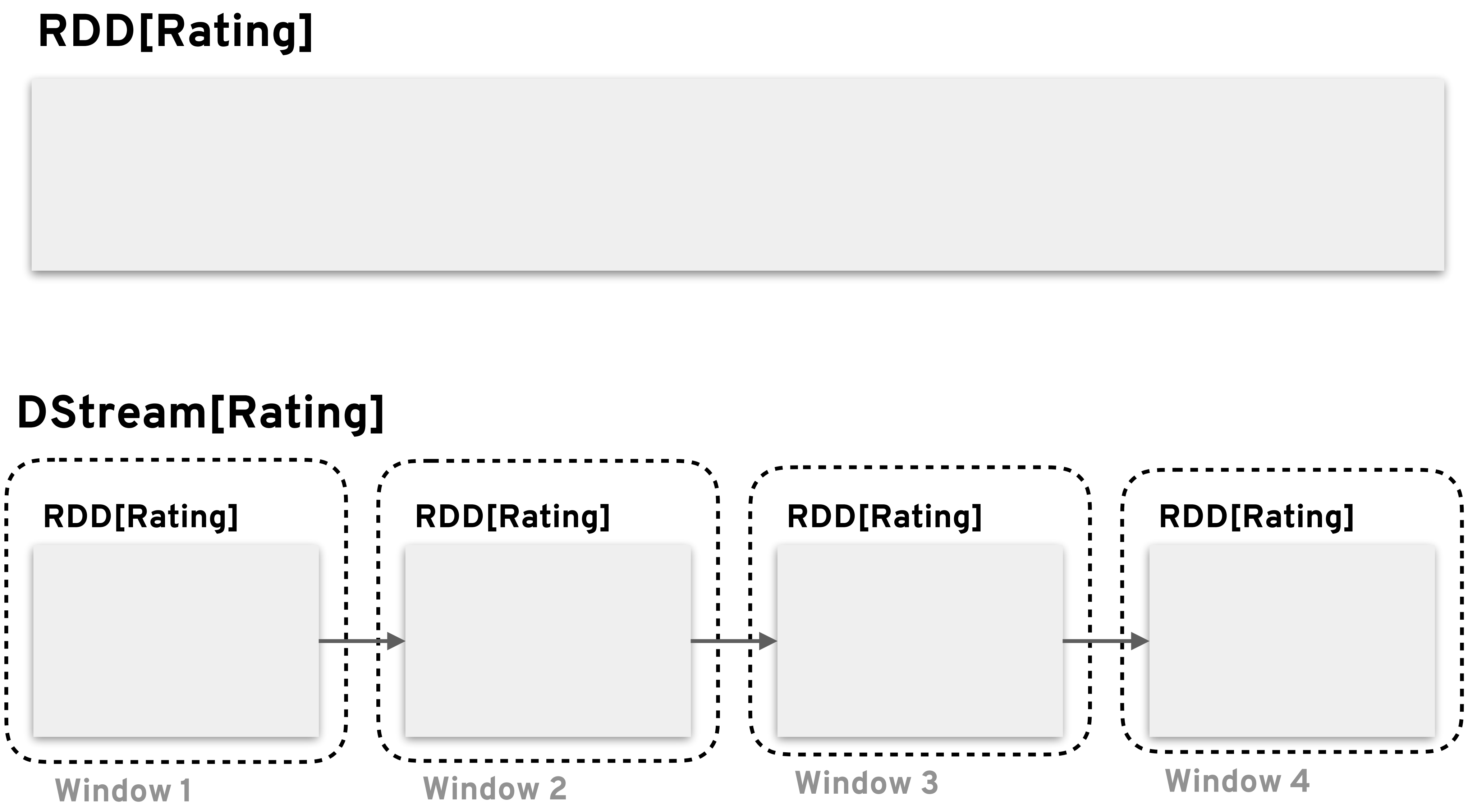
```
val model = ALS.train(ratings, rank, iterations, lambda)
```

```
case class Rating(int user, int product, double rating)

val ratings: RDD[Rating]

val rank: int

val iterations: int

val lambda: Double
```

MLlib ALS

```
> val model = ALS.train(ratings, rank, iterations, lambda)

model: MatrixFactorizationModel

class MatrixFactorizationModel {

   val userFeatures: RDD[(Int, Array[Double])]
   val productFeatures: RDD[(Int, Array[Double])]

}
```

Spark Streaming ALS

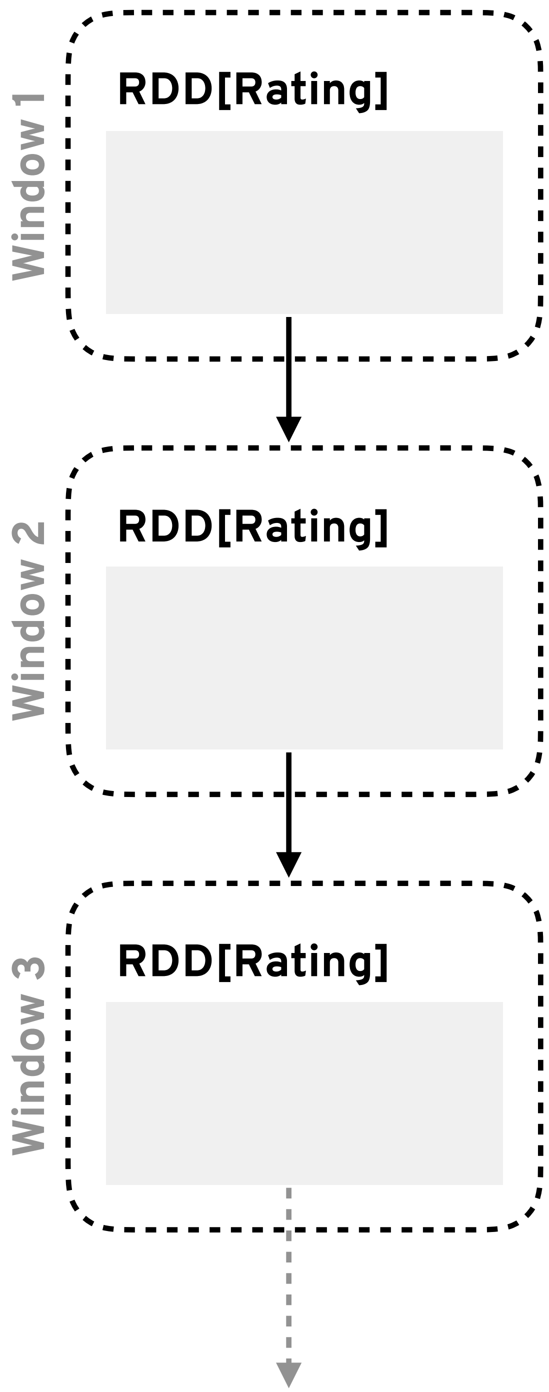


Spark Streaming ALS

DStream[Rating]

Spark Streaming ALS

DStream[Rating]




```
model = StreamingALS.train(rdd1, params)

model = model.train(rdd2)

model = model.train(rdd3)
```

Spark Streaming ALS

```
userBias += gamma * (error - lambda * userBias)

userFeature(i) += gamma * (error *  prodFeature(j) - lambda * userFeature(i))
```

Spark Streaming ALS

```
userBias += gamma * (error - lambda * userBias)

userFeature(i) += gamma * (error *  prodFeature(j) - lambda * userFeature(i))
```

```
case class Factor(var bias: Double, features: Array[Double])
  extends Serializable {

}
```

Batch → **Streaming**

RDD[(Int, Array[Double])] → **RDD[(Int, Factor)]**

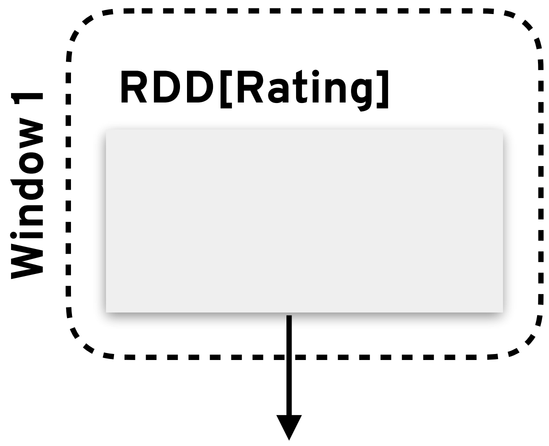
What do we need?

- user latent factors
- product latent factors
- calculate the global bias
- calculate user specific bias
- calculate product specific bias

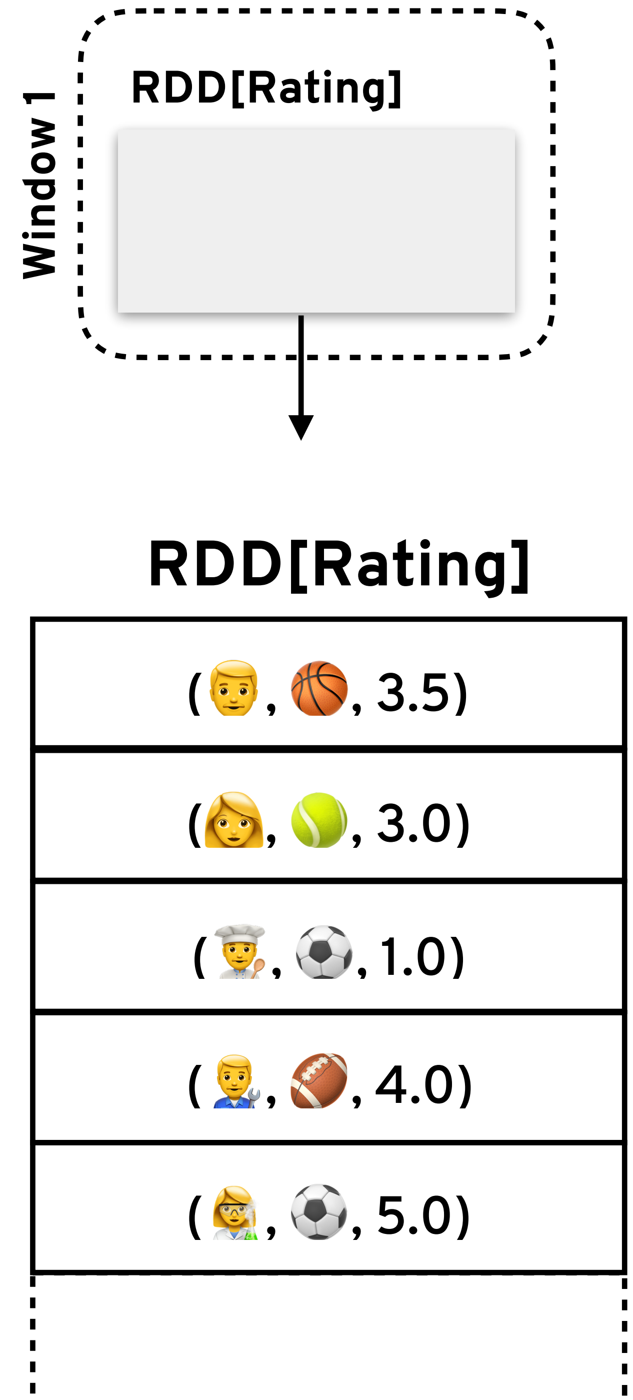
What do we need?

- user latent factors
- product latent factors
- calculate the global bias
- calculate user specific bias
- calculate product specific bias

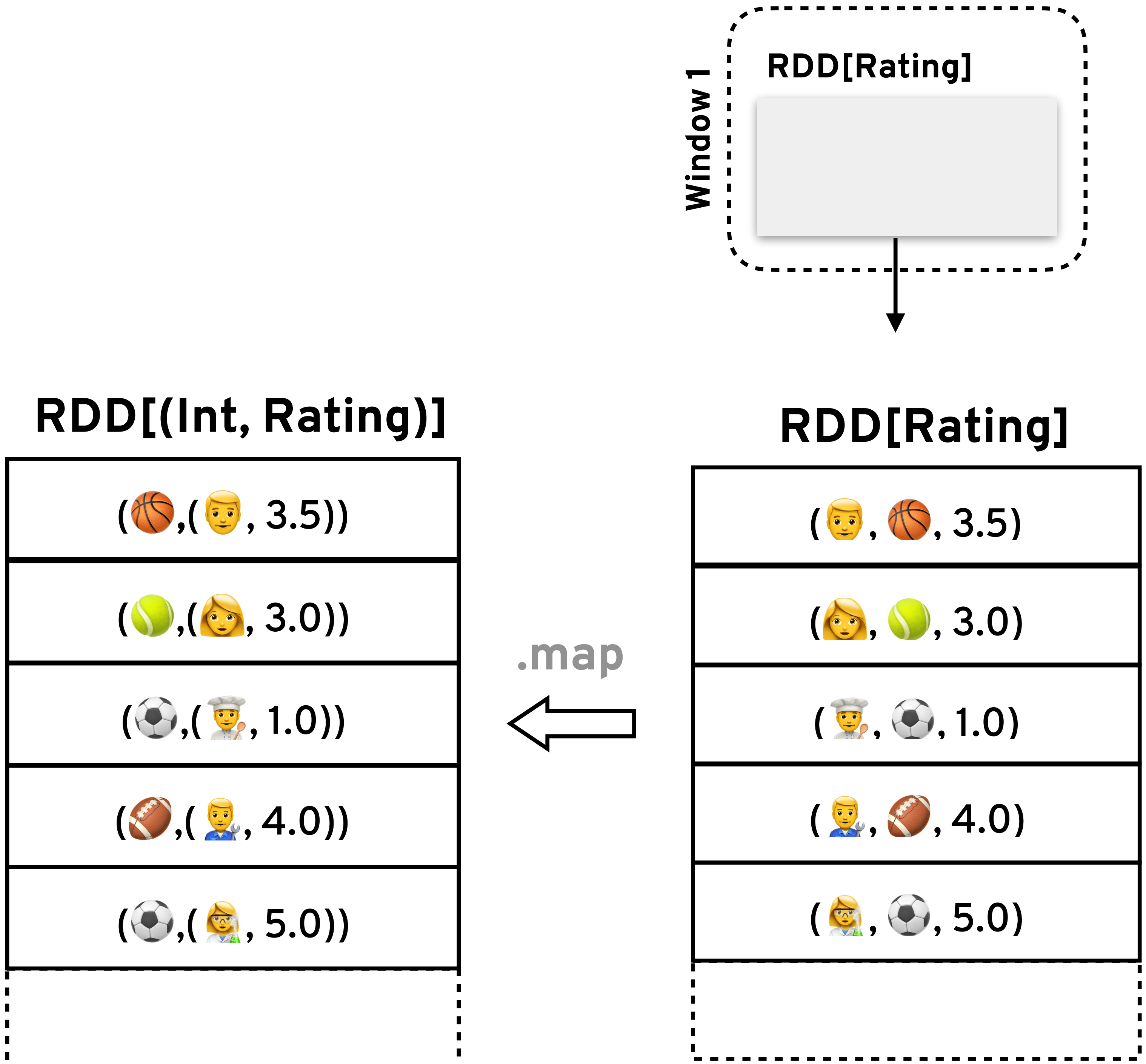
Spark Streaming ALS



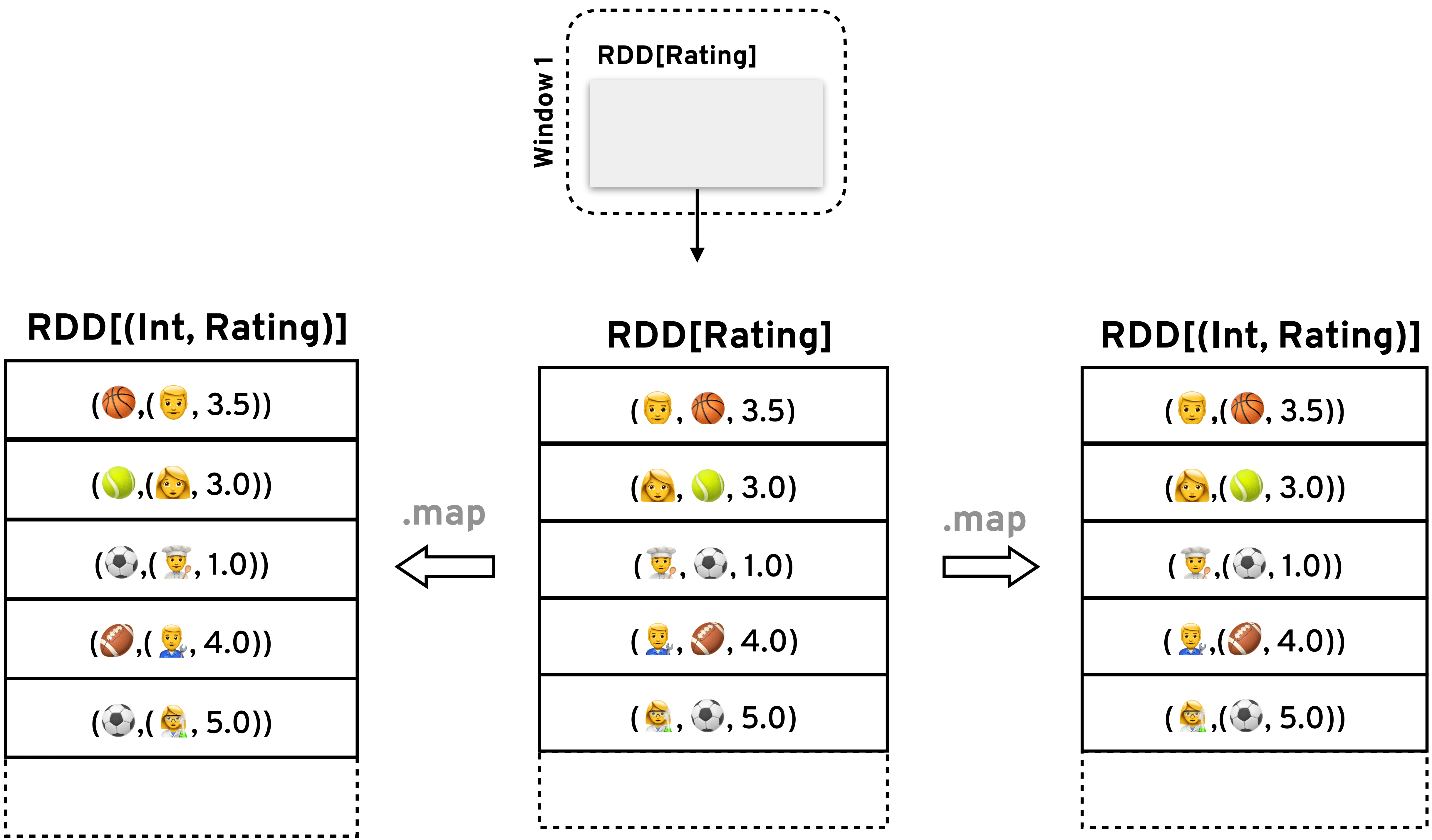
Spark Streaming ALS



Spark Streaming ALS



Spark Streaming ALS



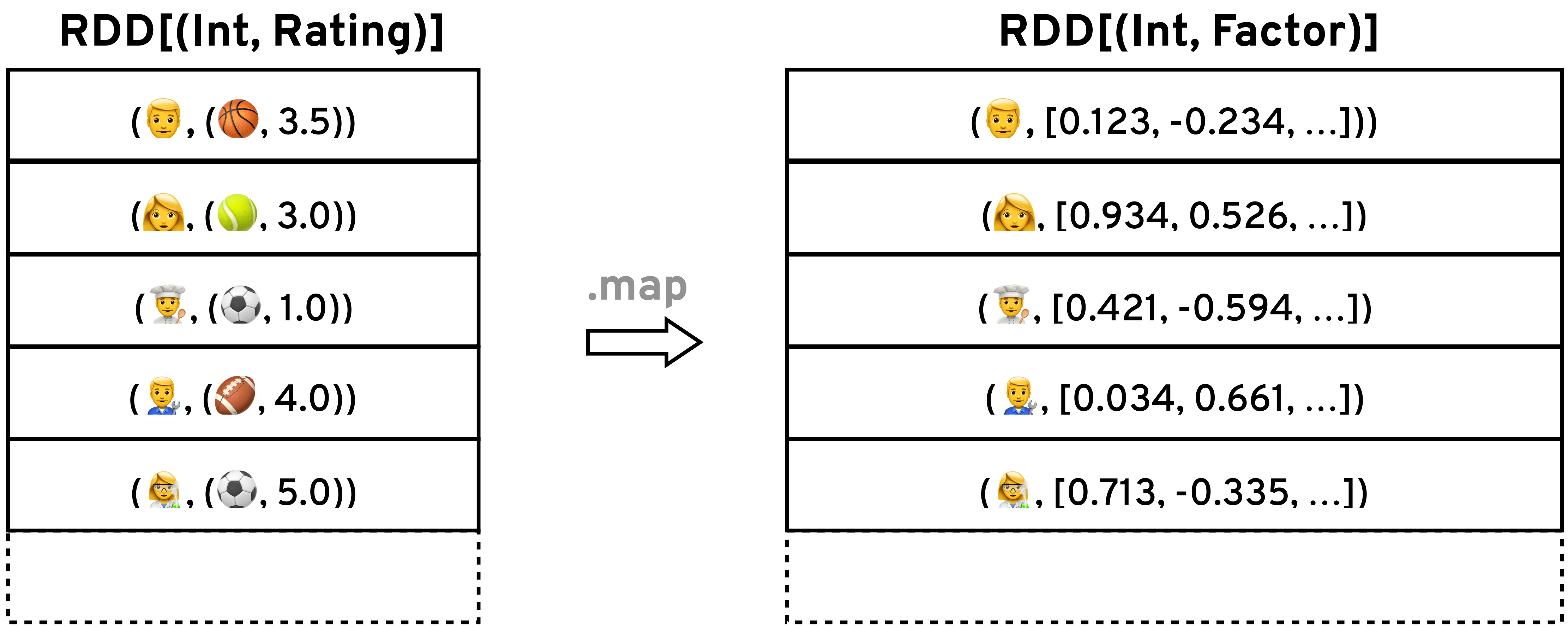
Spark Streaming ALS

Spark Streaming ALS

RDD[(Int, Rating)]

(👱‍♂️, (🏀, 3.5))
(👱‍♀️, (🎾, 3.0))
(👨‍🍳, (⚽, 1.0))
(👨‍🔧, (🏈, 4.0))
(👩‍🔬, (⚽, 5.0))

Spark Streaming ALS



Spark Streaming ALS

Spark Streaming ALS

RDD[(Int, Rating)]

(👨,(🏀, 3.5))
(👩, (🎾, 3.0))
(👨‍🍳,(⚽, 1.0))
(👨‍🔧,(🏈, 4.0))
(👩‍🔬,(⚽, 5.0))

Spark Streaming ALS

RDD[(Int, Rating)]
(👱‍♂️,(🏀, 3.5))
(👱‍♀️, (🎾, 3.0))
(👨‍🍳,(⚽, 1.0))
(👨‍🔧,(🏈, 4.0))
(👩‍🔬,(⚽, 5.0))

RDD[(Int, Factor)]
(👱‍♂️, [0.123, -0.234, ...])
(👱‍♀️, [0.934, 0.526, ...])
(👨‍🍳, [0.421, -0.594, ...])
(👨‍🔧, [0.034, 0.661, ...])
(👩‍🔬, [0.713, -0.335, ...])

Spark Streaming ALS

RDD[(Int, Rating)]
(👨,(🏀, 3.5))
(👩, (🎾, 3.0))
(👨‍🍳,(⚽, 1.0))
(👨‍🔧,(🏈, 4.0))
(👩‍🔬,(⚽, 5.0))

.join

RDD[(Int, Factor)]
(👨, [0.123, -0.234, ...])
(👩, [0.934, 0.526, ...])
(👨‍🍳, [0.421, -0.594, ...])
(👨‍🔧, [0.034, 0.661, ...])
(👩‍🔬, [0.713, -0.335, ...])

Spark Streaming ALS

RDD[(Int, (Int, Double, Factor))]

(🏀, (👨, 3.5, [0.123, -0.234, ...]))
(🎾, (👩, 3.0, [0.934, 0.526, ...]))
(⚽, (👨‍🍳, 1.0, [0.421, -0.594, ...]))
(🏈, (👨‍🔧, 4.0, [0.034, 0.661, ...]))
(⚽, (👩‍🔬, 5.0, [0.713, -0.335, ...]))

user latent factors

Spark Streaming ALS

Spark Streaming ALS

RDD[(Int, (Int, Double, Factor))]

(🏀, (👨, 3.5, [0.123, -0.234, …]))
(🎾, (👩, 3.0, [0.934, 0.526, …]))
(⚽, (👨‍🍳, 1.0, [0.421, -0.594, …]))
(🏈, (👨‍🔧, 4.0, [0.034, 0.661, …]))
(⚽, (👩‍🔬, 5.0, [0.713, -0.335, …]))

Spark Streaming ALS

RDD[(Int, (Int, Double, Factor))]
(🏀, (👨, 3.5, [0.123, -0.234, ...]))
(🎾, (👩, 3.0, [0.934, 0.526, ...]))
(⚽, (👨‍🍳, 1.0, [0.421, -0.594, ...]))
(🏈, (👨‍🔧, 4.0, [0.034, 0.661, ...]))
(⚽, (👩‍🌾, 5.0, [0.713, -0.335, ...]))

.join

RDD[(Int, Factor)]
(🏀, [0.764, 0.254, ...]))
(⚽, [0.136, 0.933, ...])
(🎾, [0.663, -0.134, ...])
(🎾, [0.811, 0.535, ...])
(🏈, [0.234, -0.579, ...])

Spark Streaming ALS

RDD[(Int, (Int, Double, Factor, Factor))]

(🏀, (👨, 3.5, [0.123, ...], [0.764, ...]))
(🎾, (👩, 3.0, [0.934, 0.526, ...], [0.933, ...]))
(⚽, (👨‍🍳, 1.0, [0.421, -0.594, ...], [0.663, ...]))
(🏈, (👨‍🔧, 4.0, [0.034, 0.661, ...], [0.811, ...]))
(⚽, (👩‍🔬, 5.0, [0.713, -0.335, ...], [0.234, ...]))

What do we need?

- user latent factors
- product latent factors
- calculate the global bias
- calculate user specific bias
- calculate product specific bias

Spark Streaming ALS

RDD[(Int, (Int, Double, Factor, Factor))]

(🏀, (👨, 3.5, [0.123, …], [0.764, …]))
(🎾, (👩, 3.0, [0.934, 0.526, …], [0.933, …]))
(⚽, (👨‍🍳, 1.0, [0.421, -0.594, …], [0.663, …]))
(🏈, (👨‍🔧, 4.0, [0.034, 0.661, …], [0.811, …]))
(⚽, (👩‍🔬, 5.0, [0.713, -0.335, …], [0.234, …]))

Spark Streaming ALS

RDD[(Int, (Int, Double, Factor, Factor))]

(🏀, (👱‍♂️, 3.5, [0.123, …], [0.764, …]))
(🎾, (👱‍♀️, 3.0, [0.934, 0.526, …], [0.933, …]))
(⚽, (👨‍🍳, 1.0, [0.421, -0.594, …], [0.663, …]))
(🏈, (👨‍🔧, 4.0, [0.034, 0.661, …], [0.811, …]))
(⚽, (👩‍🔬, 5.0, [0.713, -0.335, …], [0.234, …]))

What do we need?

- user latent factors
- product latent factors
- calculate the global bias
- calculate user specific bias
- calculate product specific bias

Spark Streaming ALS

$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x \right)$$

$$\epsilon_{x,y} = r_{x,y} - \hat{r}_{x,y}$$

$$\hat{r}_{x,y} = \mu + b_x + b_y + \mathsf{U}_x \cdot \mathsf{P}_y^T$$

Spark Streaming ALS

$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x \right)$$

$$\epsilon_{x,y} = r_{x,y} - \hat{r}_{x,y}$$

$$\hat{r}_{x,y} = \mu + b_x + b_y + \mathsf{U}_x \cdot \mathsf{P}_y^T$$

Spark Streaming ALS

$$\hat{r}_{x,y} = \mu + b_x + b_y + \mathsf{U}_x \cdot \mathsf{P}_y^T$$

Spark Streaming ALS

$$\hat{r}_{x,y} = \mu + b_x + b_y + \mathsf{U}_x \cdot \mathsf{P}_y^T$$

RDD[(Int, (Int, Double, Factor, Factor))]

(⚽, (👨‍🍳, 1.0, [0.421, -0.594, ...], [0.663, ...]))

predicted(⚽, 👨‍🍳) = μ + b_x + b_y + [0.421, -0.594, …] x $[0.663, …]^T$ = 2.3

Spark Streaming ALS

$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x \right)$$

$$\epsilon_{x,y} = r_{x,y} - \hat{r}_{x,y}$$

Spark Streaming ALS

$$\epsilon_{x,y} = r_{x,y} - \hat{r}_{x,y}$$

Spark Streaming ALS

$$\epsilon_{x,y} = r_{x,y} - \hat{r}_{x,y}$$

RDD[(Int, (Int, Double, Factor))]

(⚽, (👨‍🍳, 1.0, [0.421, -0.594, ...], [0.663, ...]))

```
rating(👨‍🍳,⚽) = 1.0

predicted(👨‍🍳,⚽) = 2.3

error(👨‍🍳,⚽) = rating(👨‍🍳,⚽) - predicted(👨‍🍳,⚽) = -1.3
```

Spark Streaming ALS

gradients

$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x\right)$$

$$b_y \leftarrow b_y + \gamma \left(\epsilon_{x,y} - \lambda_y b_y\right)$$

Spark Streaming ALS

gradients

$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x \right)$$

$$b_y \leftarrow b_y + \gamma \left(\epsilon_{x,y} - \lambda_y b_y \right)$$

RDD[(Int, Double, Factor)]
(👨, 0.932, [0.123, -0.140, ...])
(👩, 0.101, [0.334, 0.273, ...])
(👨‍🍳, 0.128, [0.957, -0.247, ...])
(👨‍🔧, 0.242, [0.038, 0.883, ...])
(👩‍🔬, 0.245, [0.283, -0.953, ...])

Spark Streaming ALS

gradients

$$b_x \leftarrow b_x + \gamma\left(\epsilon_{x,y} - \lambda_x b_x\right)$$

$$b_y \leftarrow b_y + \gamma\left(\epsilon_{x,y} - \lambda_y b_y\right)$$

RDD[(Int, Double, Factor)]
(👨, 0.932, [0.123, -0.140, ...])
(👱‍♀️, 0.101, [0.334, 0.273, ...])
(👨‍🍳, 0.128, [0.957, -0.247, ...])
(👨‍🔧, 0.242, [0.038, 0.883, ...])
(👩‍🔬, 0.245, [0.283, -0.953, ...])

RDD[(Int, Double, Factor)]
(🏀, 0.274, [0.445, -0.233, ...])
(🎾, 0.483, [0.843, 0.023, ...])
(⚽, 0.595, [0.284, -0.987, ...])
(🏈, 0.103, [0.340, 0.328, ...])
(⚽, 0.253, [0.472, -0.274, ...])

Spark Streaming ALS

Spark Streaming ALS

RDD[(Int, Double, Factor)]

(👱, 0.932, [0.123, -0.140, ...])
(👩, 0.101, [0.334, 0.273, ...])

Spark Streaming ALS

RDD[(Int, Double, Factor)]

(👱, 0.932, [0.123, -0.140, ...])

(👩, 0.101, [0.334, 0.273, ...])

RDD[(Int, Double)]

(👱, 0.932)

(👩, 0.101)

∇b_x

Spark Streaming ALS

RDD[(Int, Double, Factor)]

(👨, 0.932, [0.123, -0.140, ...])

(👩, 0.101, [0.334, 0.273, ...])

RDD[(Int, Double)]

(👨, 0.932)

(👩, 0.101)

RDD[(Int, Factor)]

(👨, [0.123, ...])

(👩, [0.334, ...])

∇b_x

∇U_x

Spark Streaming ALS

RDD[(Int, Double, Factor)]

(👱‍♂️, 0.932, [0.123, -0.140, ...])
(👱‍♀️, 0.101, [0.334, 0.273, ...])

RDD[(Int, Double, Factor)]

(🏀, 0.274, [0.445, -0.233, ...])
(🎾, 0.483, [0.843, 0.023, ...])

RDD[(Int, Double)]

(👱‍♂️, 0.932)
(👱‍♀️, 0.101)

∇b_x

RDD[(Int, Factor)]

(👱‍♂️, [0.123, ...])
(👱‍♀️, [0.334, ...])

∇U_x

Spark Streaming ALS

RDD[(Int, Double, Factor)]

(👱, 0.932, [0.123, -0.140, ...])
(👩, 0.101, [0.334, 0.273, ...])

RDD[(Int, Double, Factor)]

(🏀, 0.274, [0.445, -0.233, ...])
(🎾, 0.483, [0.843, 0.023, ...])

RDD[(Int, Double)]

(👱, 0.932)
(👩, 0.101)

∇b_x

RDD[(Int, Factor)]

(👱, [0.123, ...])
(👩, [0.334, ...])

∇U_x

RDD[(Int, Double)]

(🏀, 0.274)
(🎾, 0.483)

∇b_y

Spark Streaming ALS

RDD[(Int, Double, Factor)]

(👱, 0.932, [0.123, -0.140, ...])
(👩, 0.101, [0.334, 0.273, ...])

RDD[(Int, Double, Factor)]

(🏀, 0.274, [0.445, -0.233, ...])
(🎾, 0.483, [0.843, 0.023, ...])

RDD[(Int, Double)]

(👱, 0.932)
(👩, 0.101)

∇b_x

RDD[(Int, Factor)]

(👱, [0.123, ...])
(👩, [0.334, ...])

∇U_x

RDD[(Int, Double)]

(🏀, 0.274)
(🎾, 0.483)

∇b_y

RDD[(Int, Factor)]

(🏀, [0.445, ...])
(🎾, [0.843, ...])

∇P_y

Spark Streaming ALS

Spark Streaming ALS

RDD[(Int, Double)]

(👱, 0.932)
(👱, 0.101)

b(👱) += Σ∇b(👱)

Spark Streaming ALS

RDD[(Int, Double)]
(👨, 0.932)
(👨, 0.101)

RDD[(Int, Factor)]
(👨, [0.123, ...])
(👨, [0.334, ...])

b(👨) += Σ∇b(👨)

U(👨) += Σ∇U(👨)

Spark Streaming ALS

RDD[(Int, Double)]

(👨, 0.932)
(👨, 0.101)

RDD[(Int, Factor)]

(👨, [0.123, ...])
(👨, [0.334, ...])

RDD[(Int, Double)]

(⚽, 0.274)
(⚽, 0.483)

b(👨) += Σ∇b(👨)

U(👨) += Σ∇U(👨)

b(⚽) += Σ∇b(⚽)

Spark Streaming ALS

RDD[(Int, Double)]	RDD[(Int, Factor)]	RDD[(Int, Double)]	RDD[(Int, Factor)]
(👨, 0.932)	(👨, [0.123, ...])	(⚽, 0.274)	(⚽, [0.445, ...])
(👨, 0.101)	(👨, [0.334, ...])	(⚽, 0.483)	(⚽, [0.843, ...])

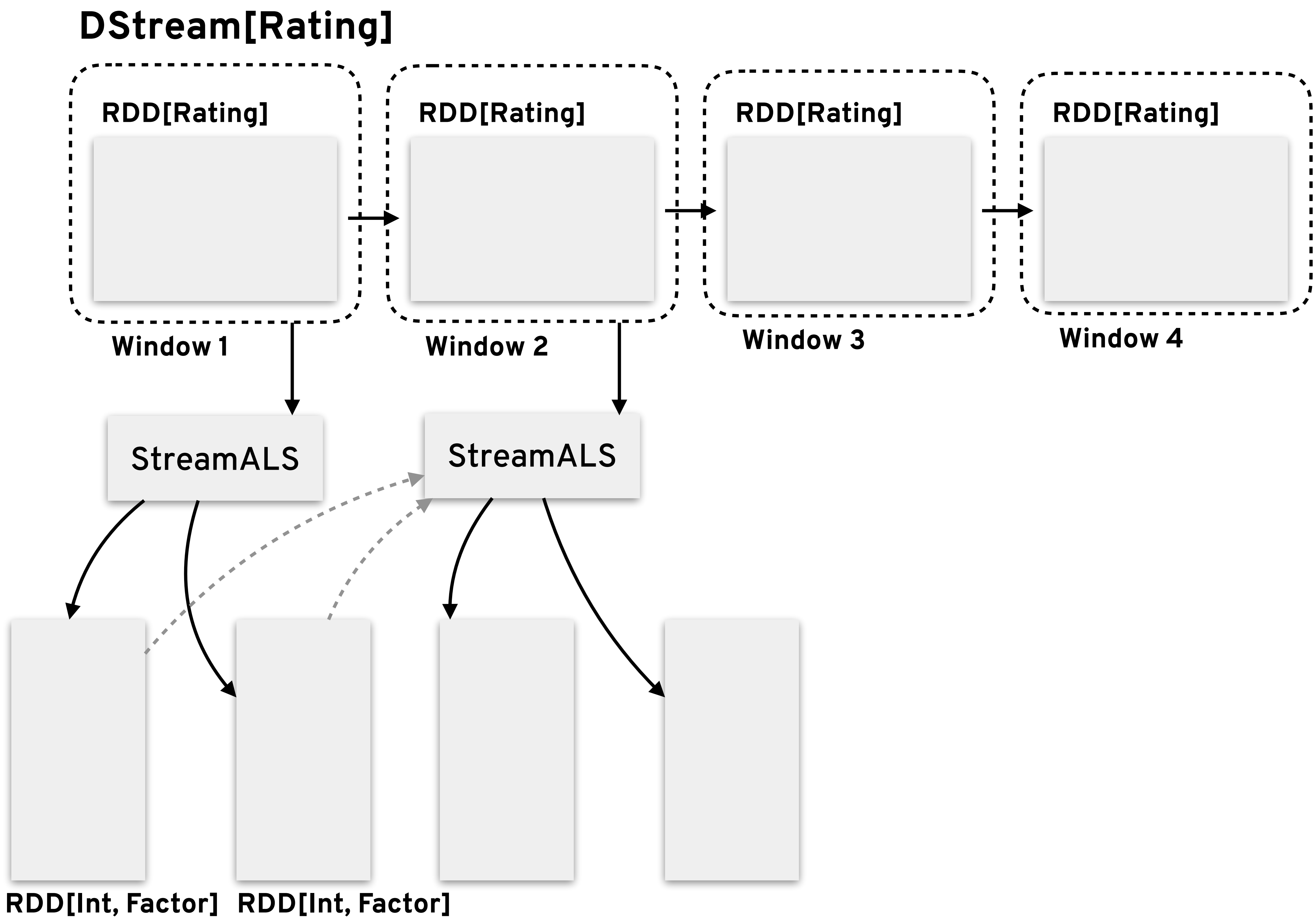
b(👨) += Σ∇b(👨)

U(👨) += Σ∇U(👨)

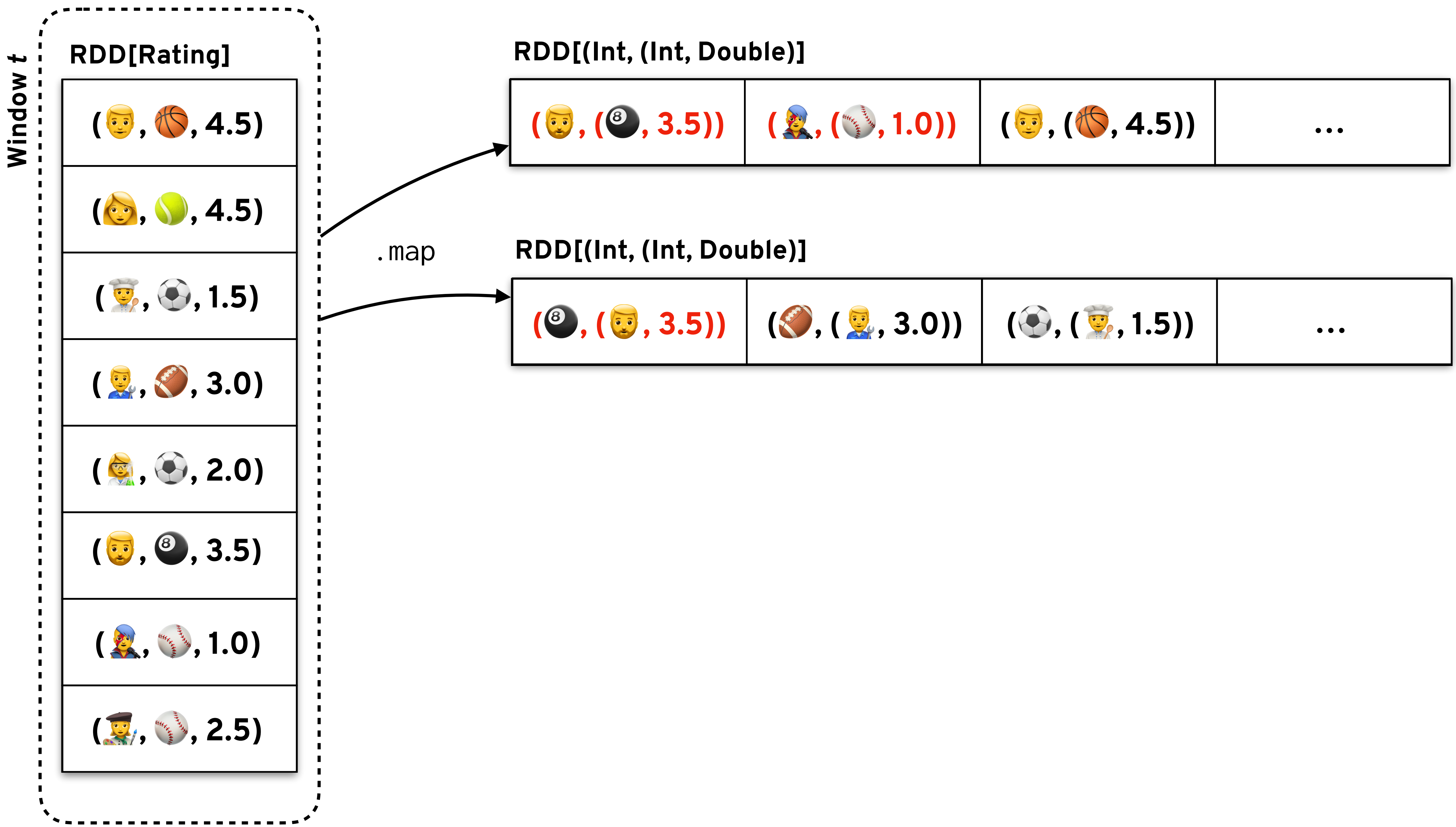
b(⚽) += Σ∇b(⚽)

U(⚽) += Σ∇U(⚽)

Spark Streaming ALS



Spark Streaming ALS



Spark Streaming ALS

RDD[(Int, (Int, Double)]

(👨, (🏀, 4.5))	(👩, (🎾, 4.5))	(🧔, (🎱, 1.5))	…

RDD[(Int, (Int, Factor)]

(👨, (0.12, [0.9,…]))
(👩, (-0.1, [1.37,…]))
(👨‍🍳, (1, [0.123,…]))
…

.fullOuterJoin

(👨, (🏀, 4.5), 👨, (0.12, [0.9,…]))
(👩, (🎾, 4.5), (👩, (-0.1, [1.37,…])))
(🧔, (🎱, 1.5), None)
…

RDD[(Int, (Int, Double), Int, Factor)]

```
userRatings.fullOuterJoin(userFactors).map {
  case (userId, (_, userFactors)) =>
    (userId, userFactors(featureGenerator.nextValue()))
}
```

Data

- MovieLens
- Widely used in recommendation engine research
- Variants
  - Small - 100,000 ratings / 9,000 movies / 700 users
  - Full - 26 million ratings / 45,000 movies / 270,000 users
- CSV data
  - Ratings
    - `(userId, movieId, rating, timestamp)`
    - `(100, 200, 3.5, 2010-12-10 12:00:00)`

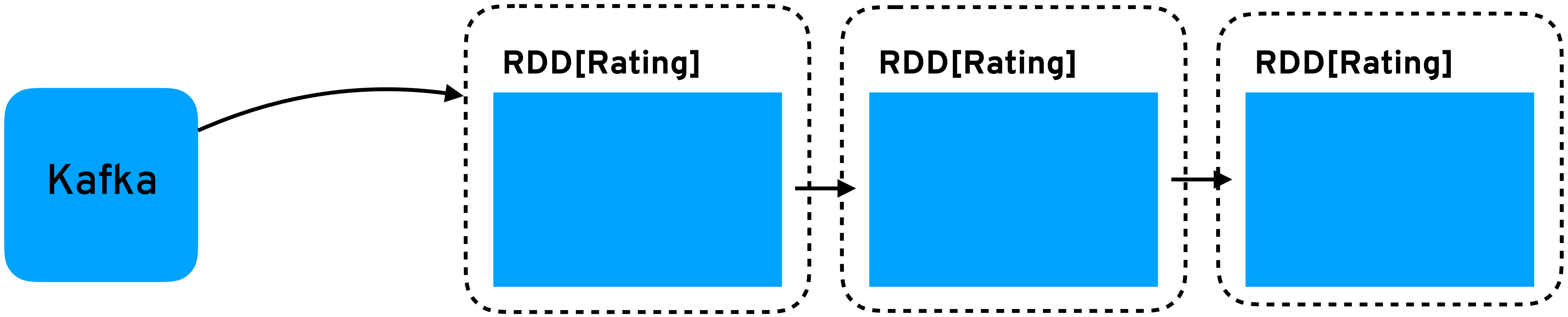
Training batch ALS

```
val split: Array[RDD[Rating]] = ratings.randomSplit(0.8, 0.2)

val model = ALS.train(split(0), rank, iter, lambda)
```

Training batch ALS

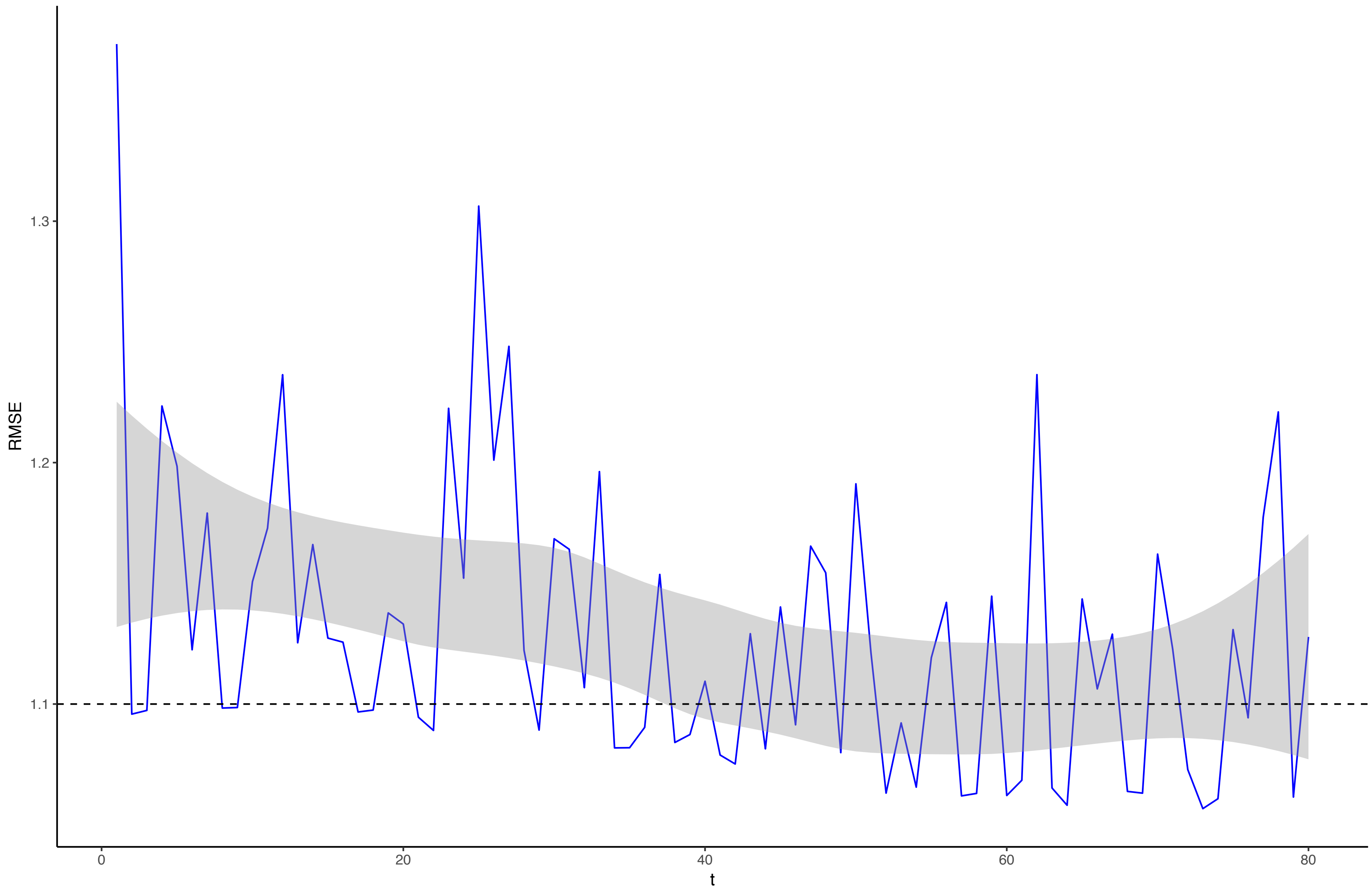
```
   val split: Array[RDD[Rating]] = ratings.randomSplit(0.8, 0.2)

   val model = ALS.train(split(0), rank, iter, lambda)


val predictions: RDD[Rating] = model.predict(split(1).map { x =>
   (x.user, x.product))
  }

val pairs = predictions.map(x => ((x.user, x.product), x.rating))
  .join(split(1).map(x => ((x.user, x.product), x.rating))
  .values

Val RMSE = math.sqrt(pairs.map(x => math.pow(x._1 - x._2, 2)).mean())
```

Training streaming ALS




```
val model = StreamingALS(rank, iterations, lambda, gamma)

trainingStreamSet.foreachRDD { rdd =>

  model.train(rdd)

  val RMSE = calculateRMSE(model, validation)

}
```

Comparison

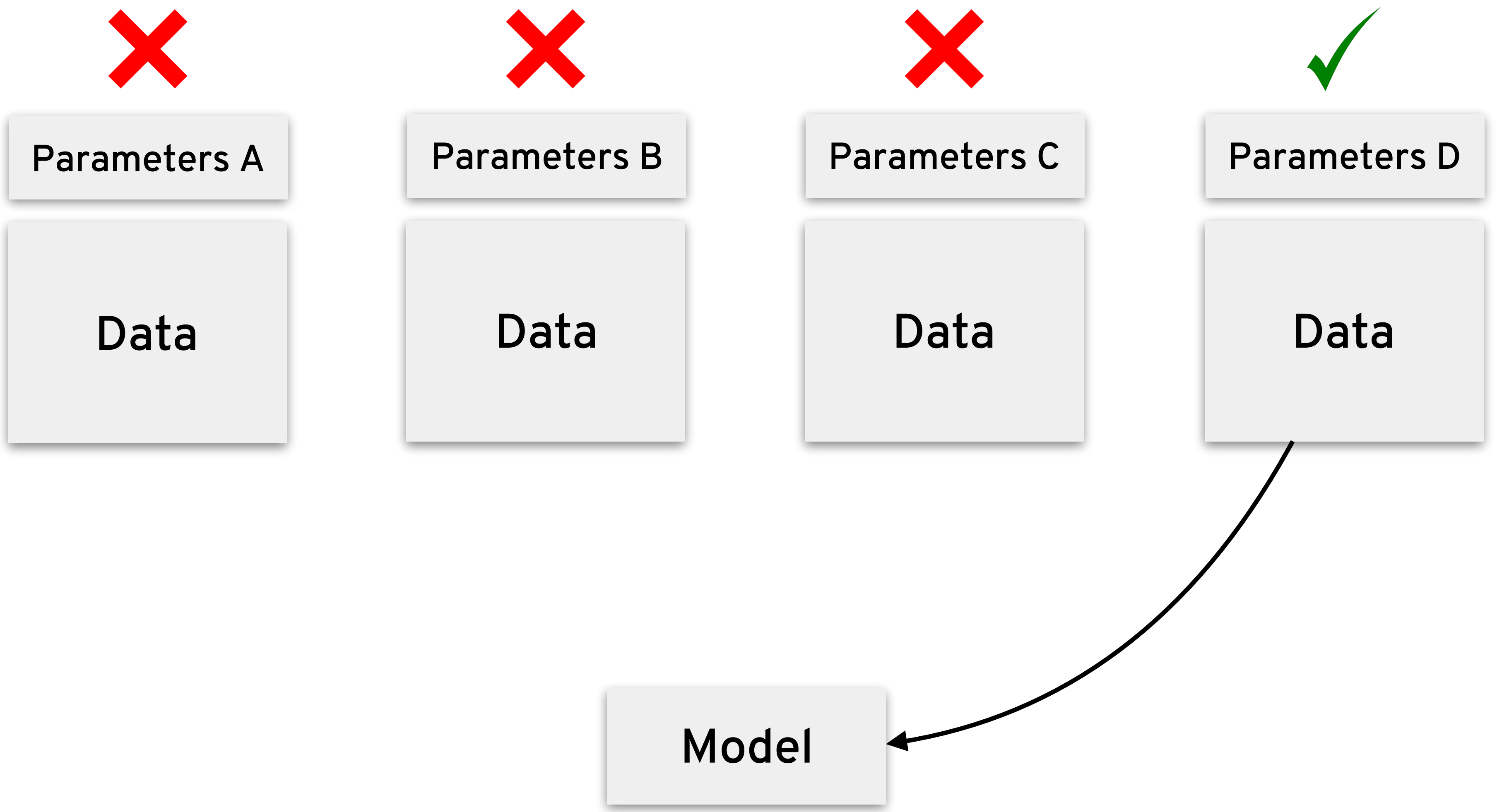


To consider

- “Cold start”
- Same as batch ALS
- Too few observations = meaningless
- Train offline

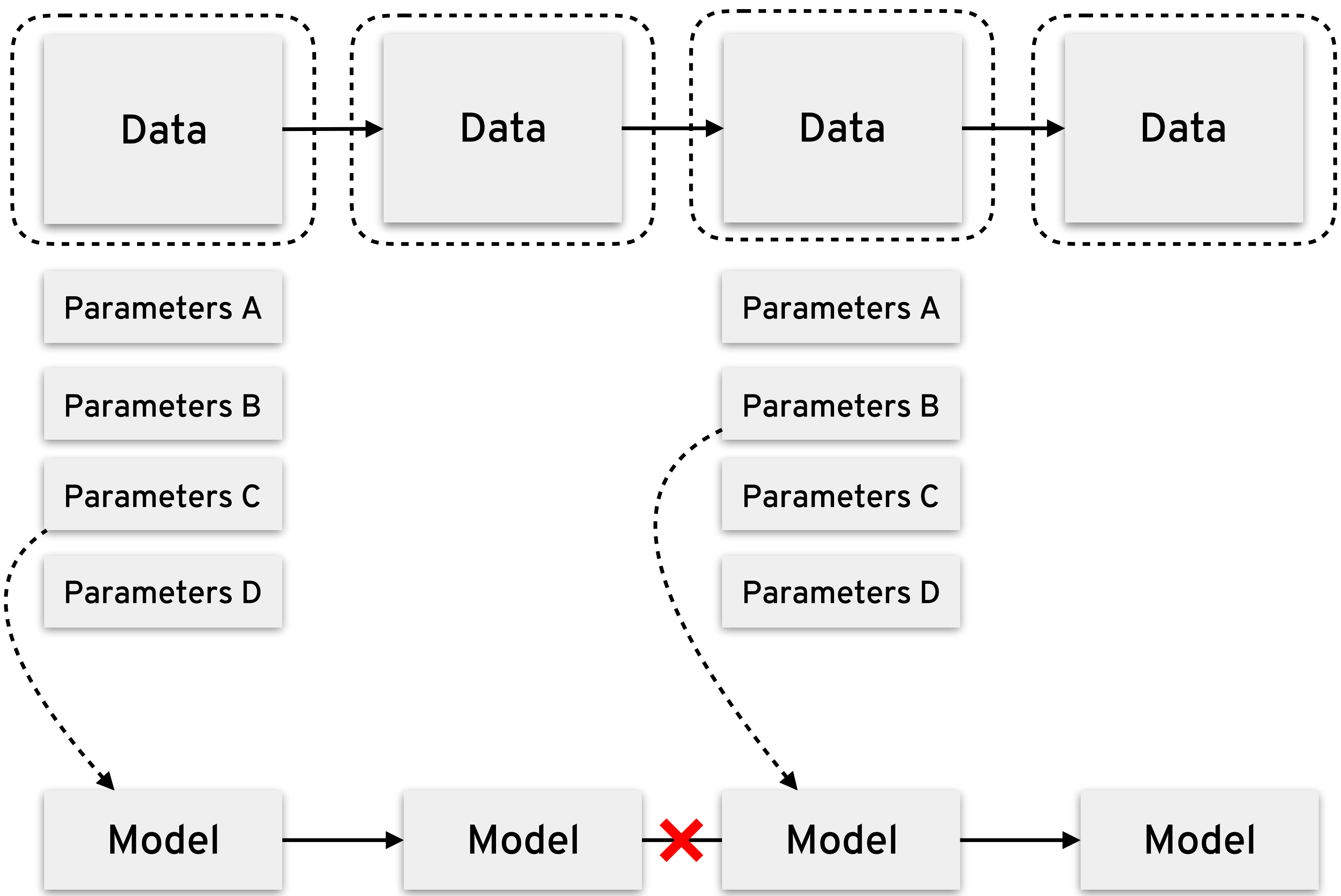
To consider

- **Hyper-parameter estimation**



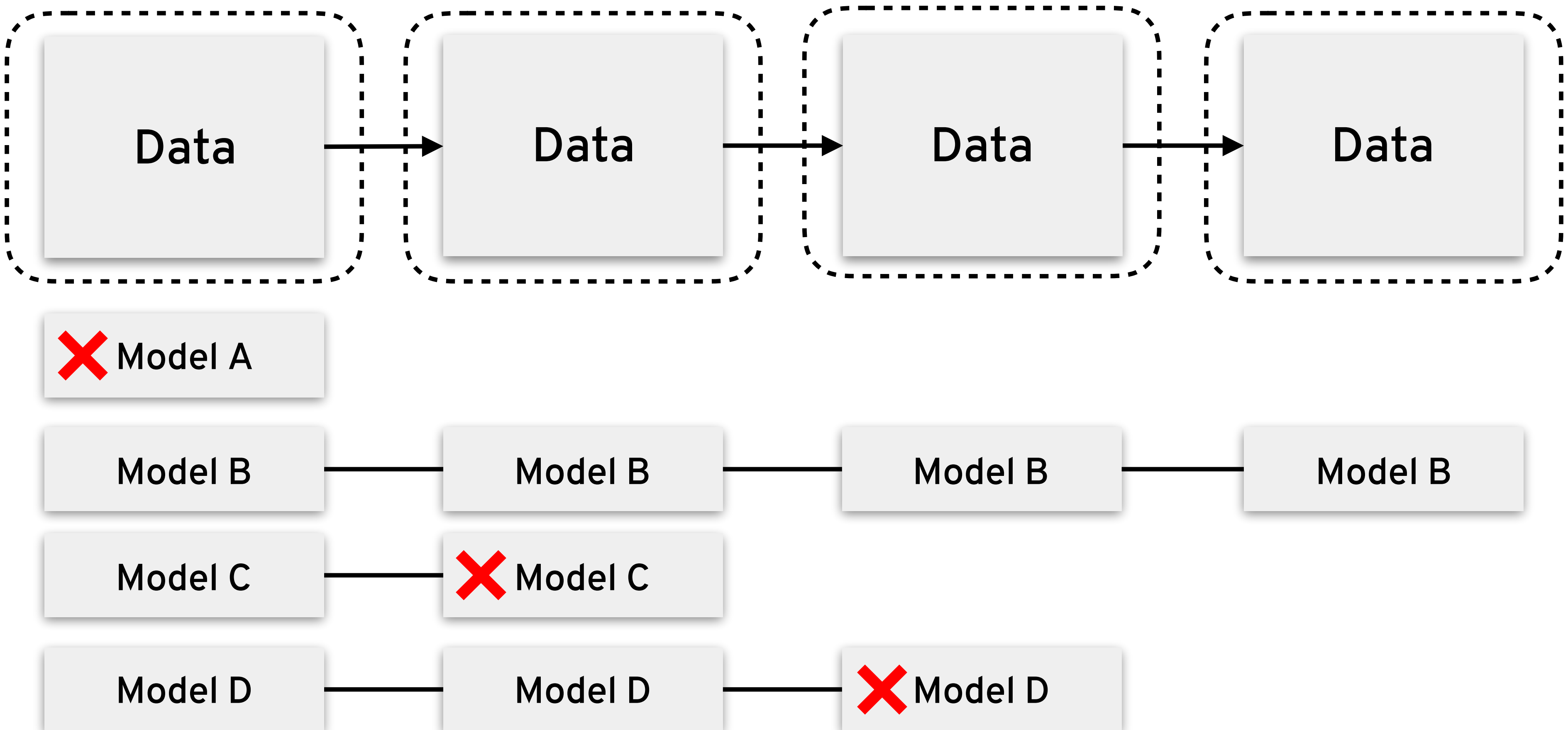
To consider

- **Hyper-parameter estimation**



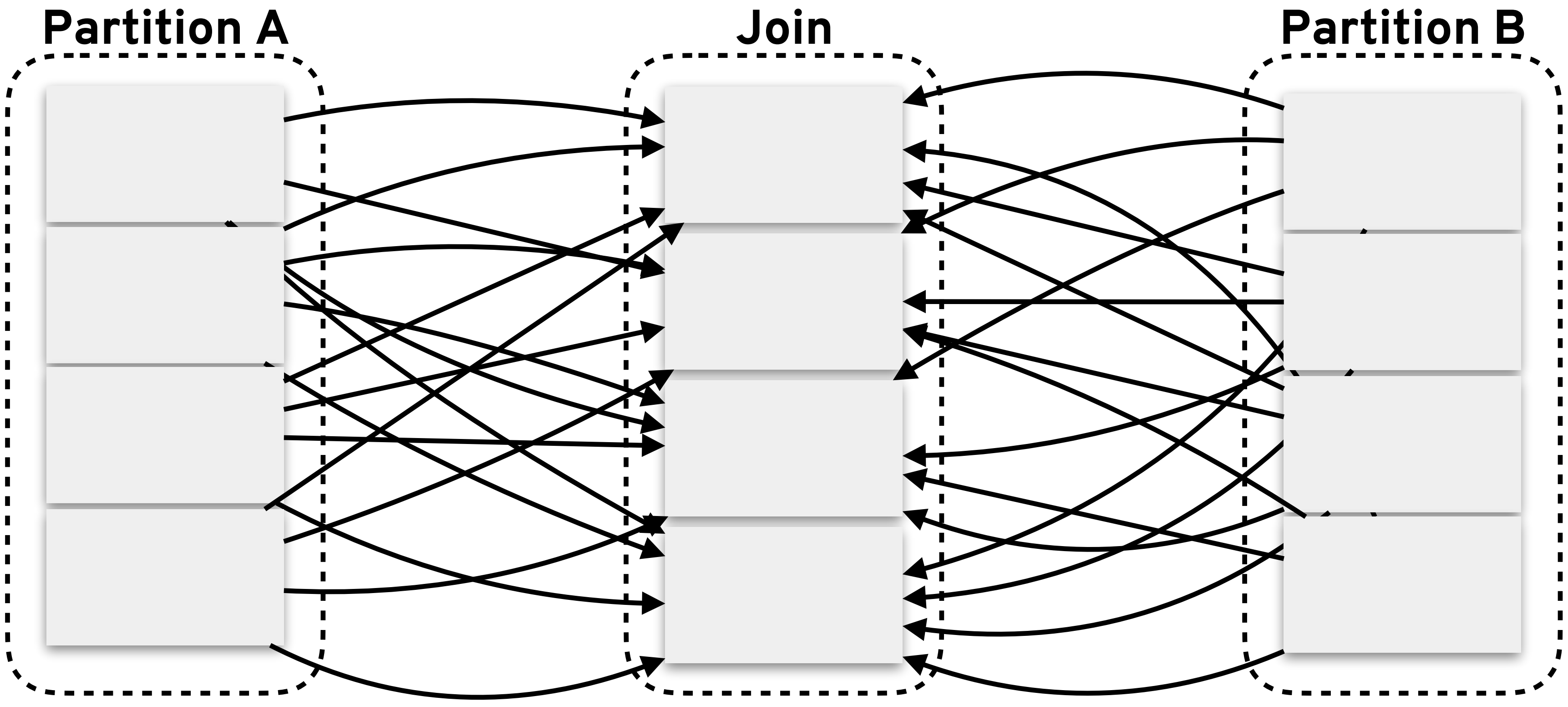
To consider

- **Hyper-parameter estimation**



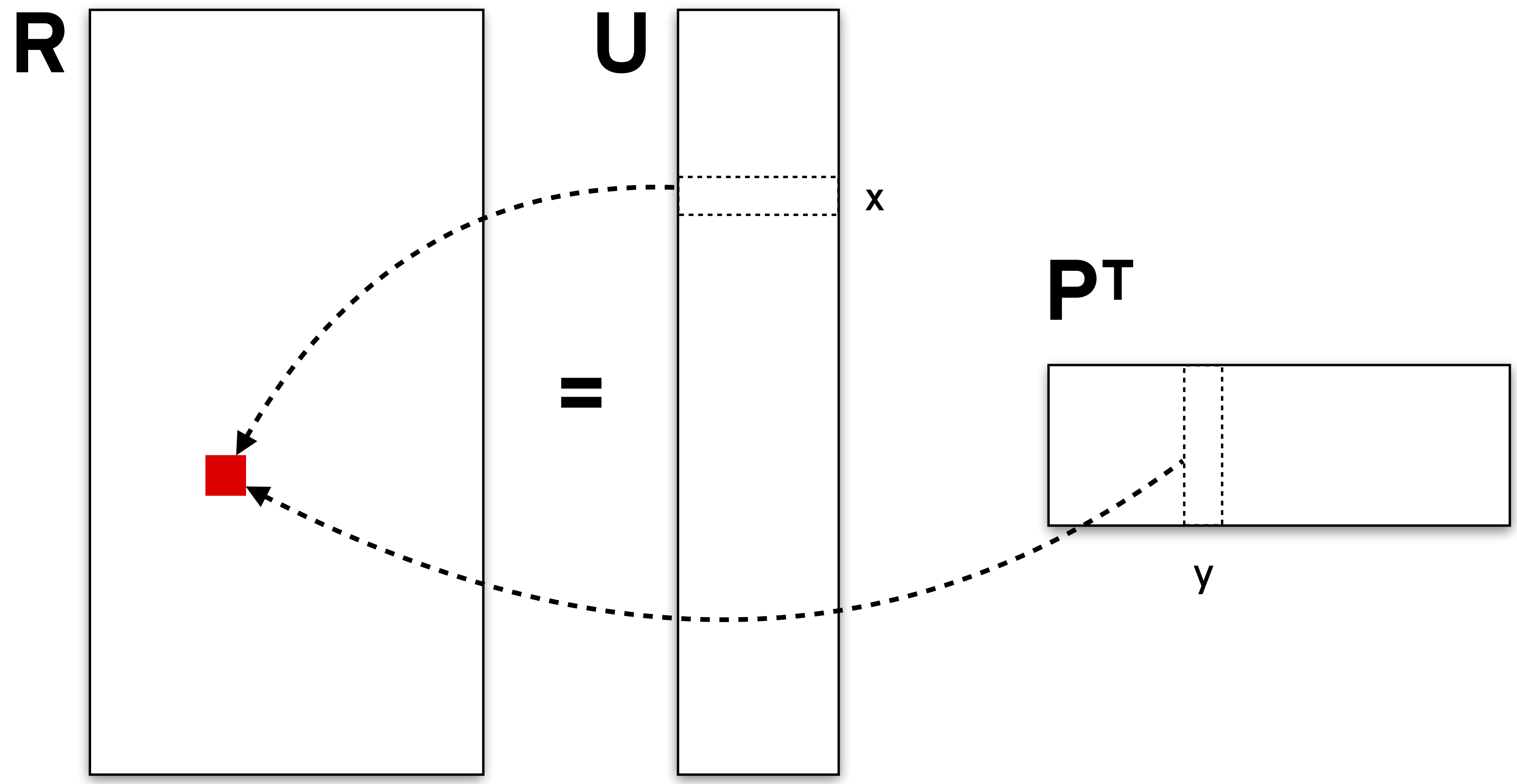
To consider

Partitioning



To consider

RDD random access?




```
val u = userFeatures.lookup(userId)
val p = productFeatures.lookup(productId)
val predicted = model.predict(userId, productId, u, p, globalBias)
```

Links

- Blog:
  - https://ruivieira.github.io/
- radanalytics.io

Thank you!