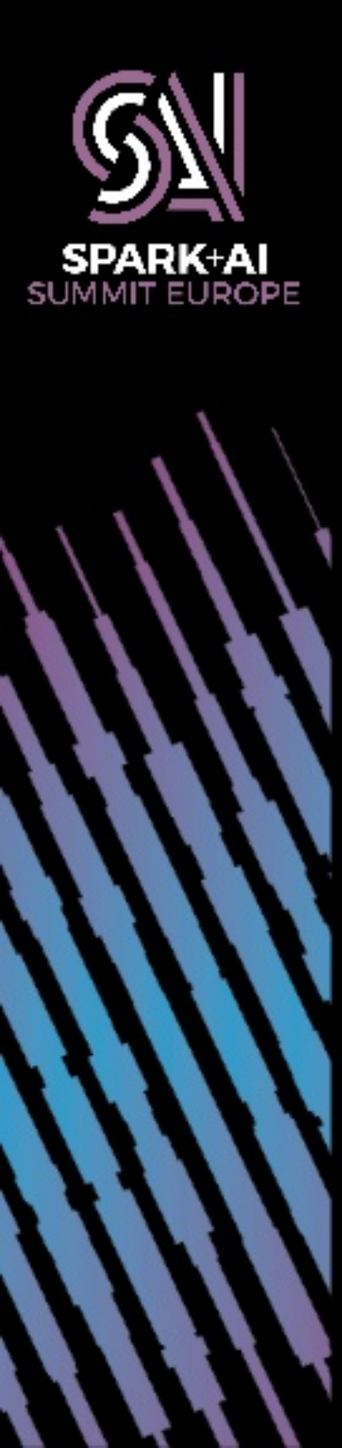


BUILDING STREAMING RECOMMENDATION ENGINES ON APACHE SPARK

Rui Vieira
Software Engineer

Overview

Collaborative Filtering

Batch Alternating Least Squares (ALS)

Streaming ALS

Apache Spark

Distributed Streaming ALS

Collaborative Filtering

Collaborative Filtering

Users, products and ratings

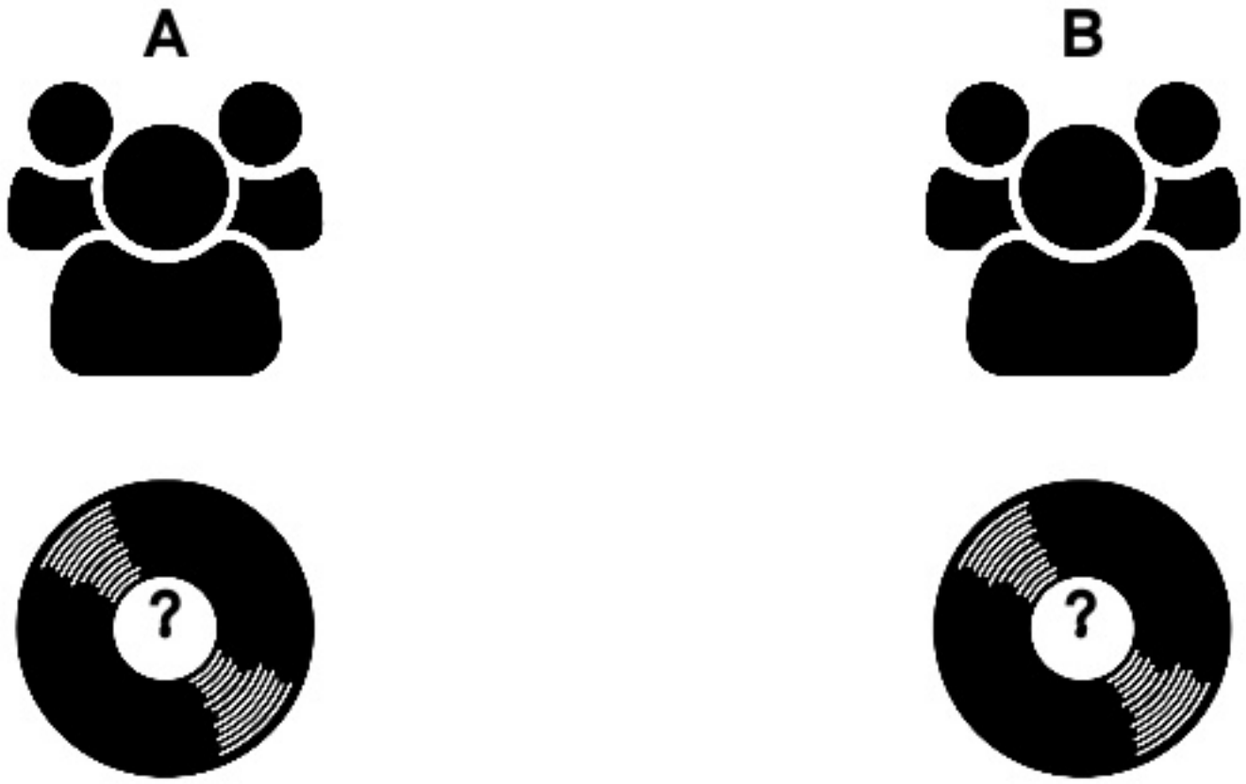
(user, product) $\longmapsto$ rating

Collaborative

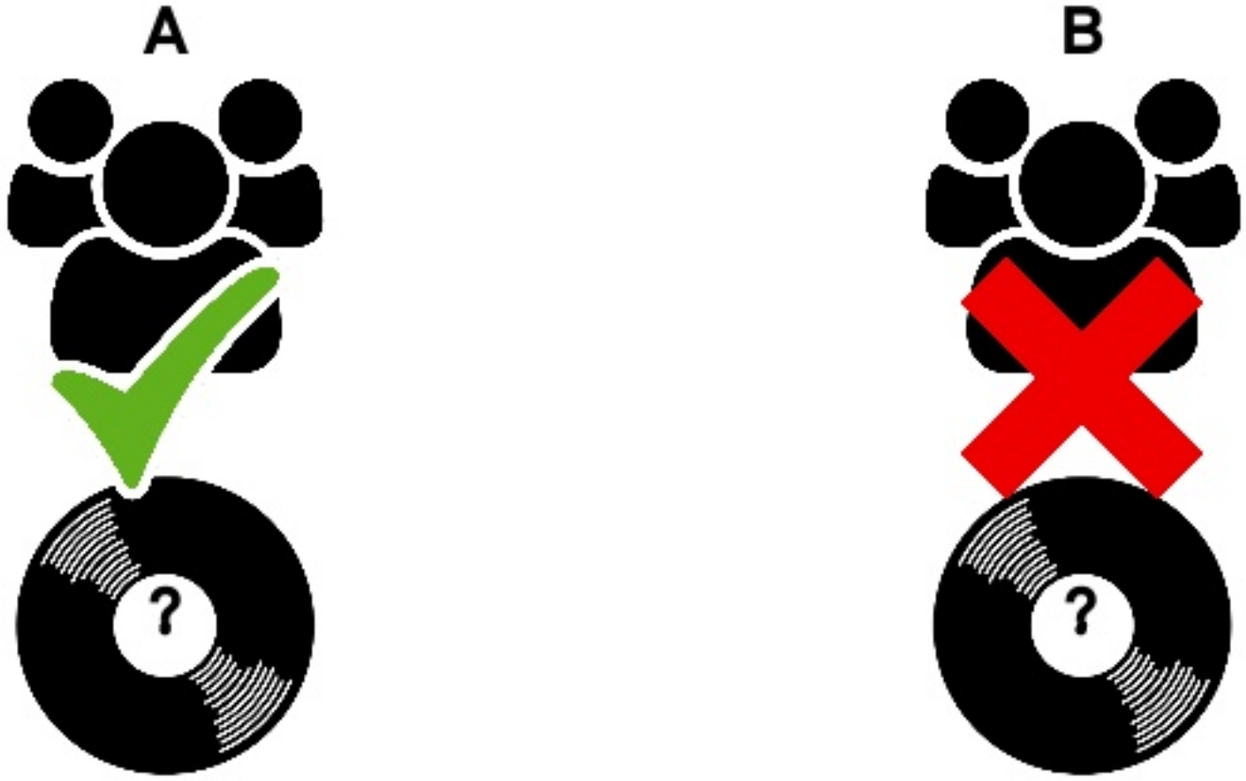
“Filtering”

Collaborative Filtering

Collaborative Filtering



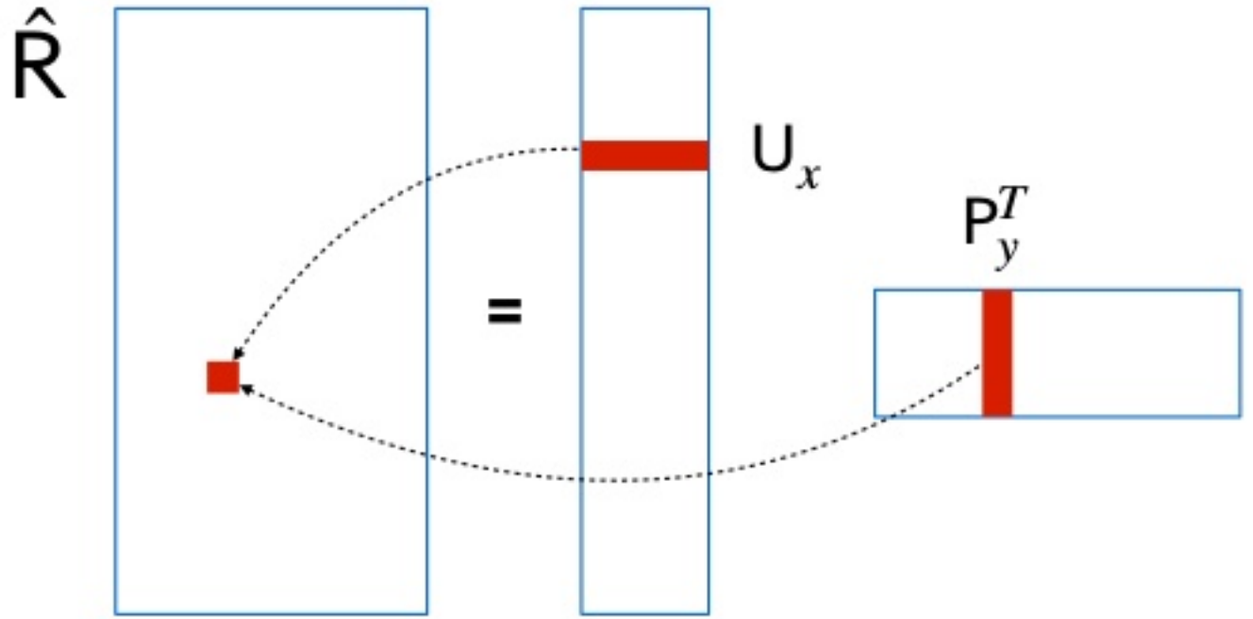
Collaborative Filtering



Alternating Least Squares

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

Alternating Least Squares



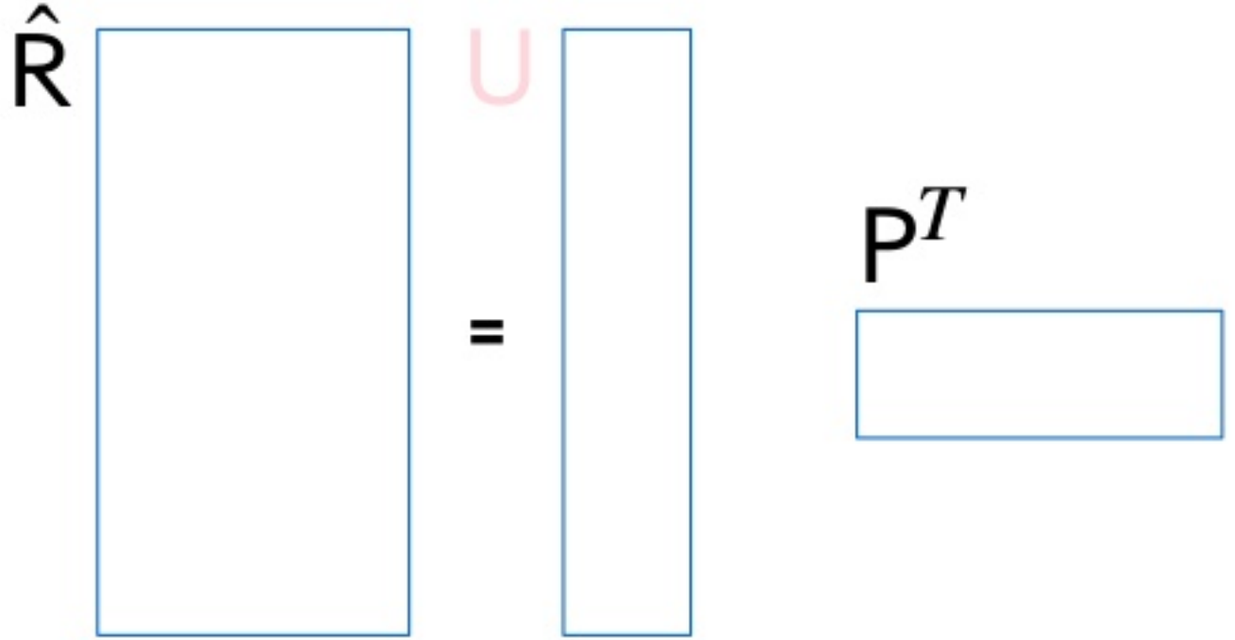
$$\hat{R}_{x,y} = U_x P_y^T$$

Batch ALS

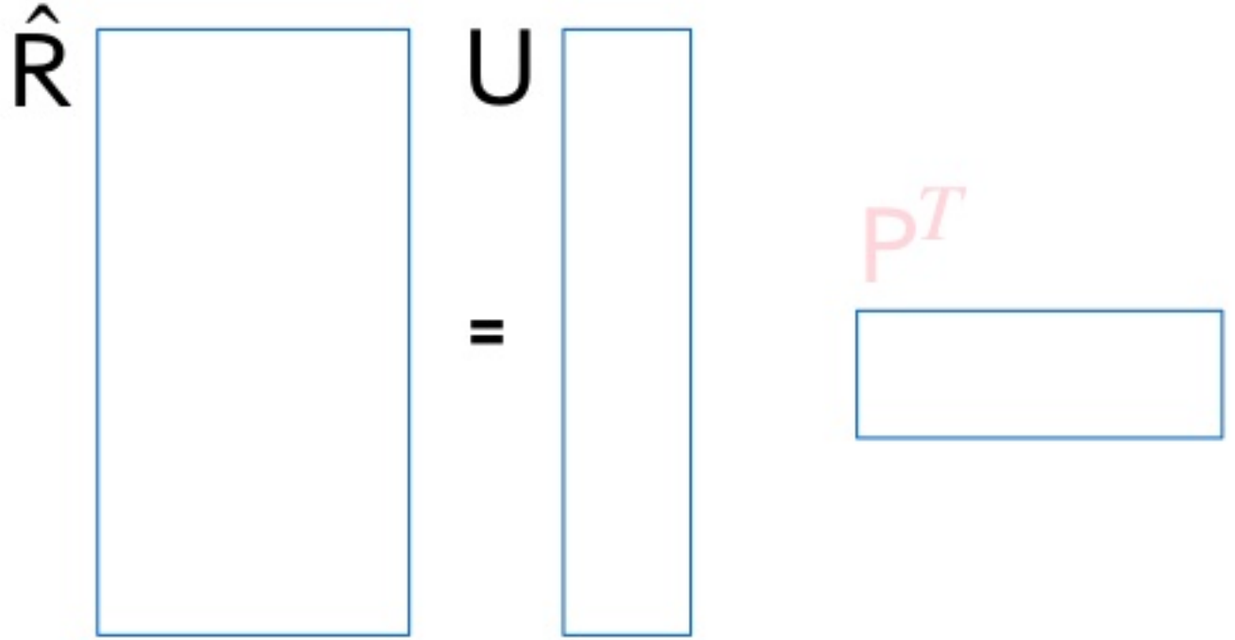
$$loss = \sum_{x,y} \left(\epsilon_{x,y} \right)^2 + \lambda_x + \lambda_y$$

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

Alternating Least Squares



Alternating Least Squares



Alternating Least Squares

$$R = \begin{array}{c} \begin{array}{ccccc} \text{user 1} & \text{user 2} & \text{user 3} & \cdots & \text{user N} \end{array} \\ \begin{bmatrix} 1 & 4.5 & 3.8 & \cdots & 3 \\ 3.2 & 3 & 3 & \cdots & 4 \\ 5 & 3 & 3.4 & \cdots & 3.1 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 2 & 4 & 1 & \cdots & 2.7 \end{bmatrix} \end{array} \begin{array}{l} \text{product 1} \\ \text{product 2} \\ \text{product 3} \\ \vdots \\ \text{product M} \end{array}$$

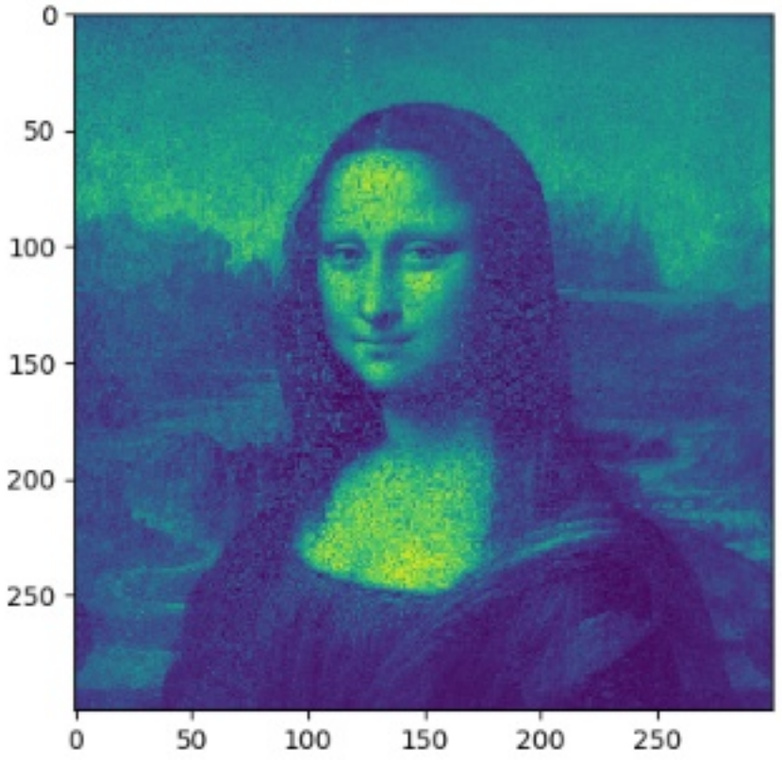
Batch ALS

	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

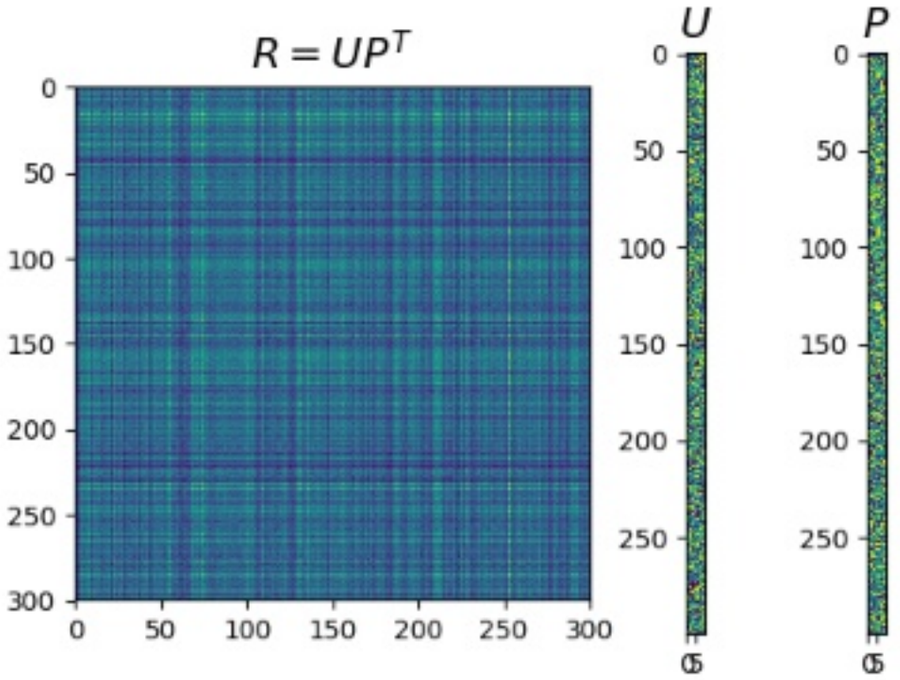
Batch ALS

	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

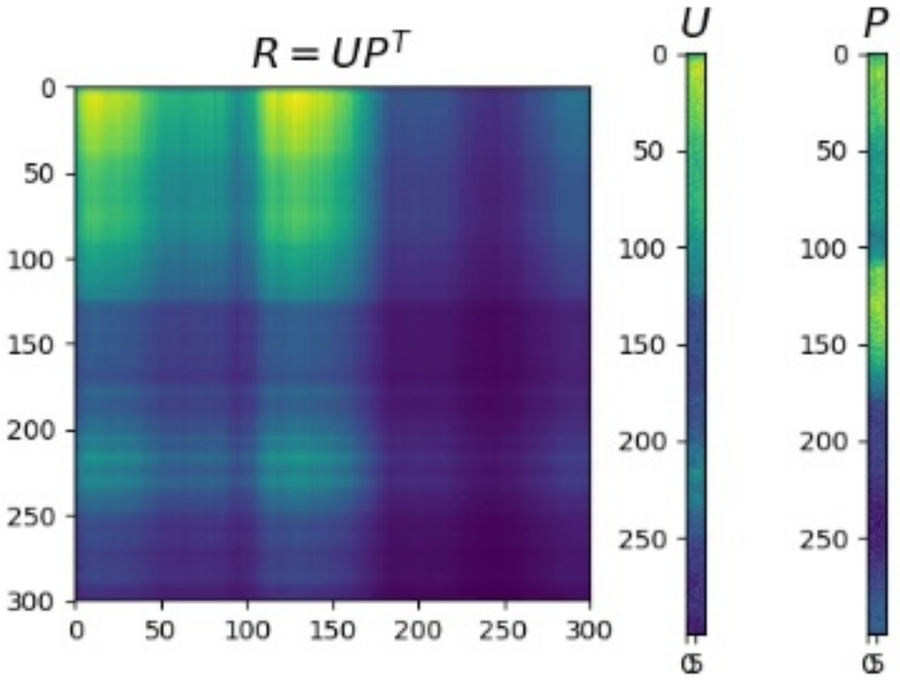
Batch ALS



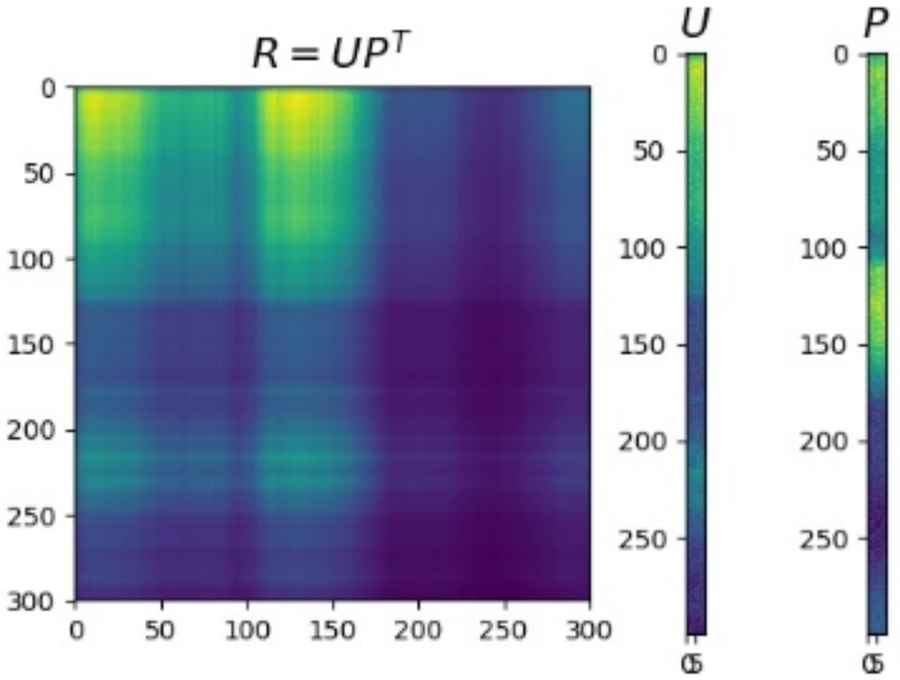
Batch ALS



Batch ALS



Batch ALS



Streaming ALS

Can we update the model with a data stream?

Stochastic Gradient Descent (SGD)

Bias SGD (B-SGD)

Streaming ALS

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

$$\hat{r}^{\star}_{x,y} = b_{x,y} + \hat{r}_{x,y}$$

Streaming ALS

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

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

Streaming ALS

bias

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

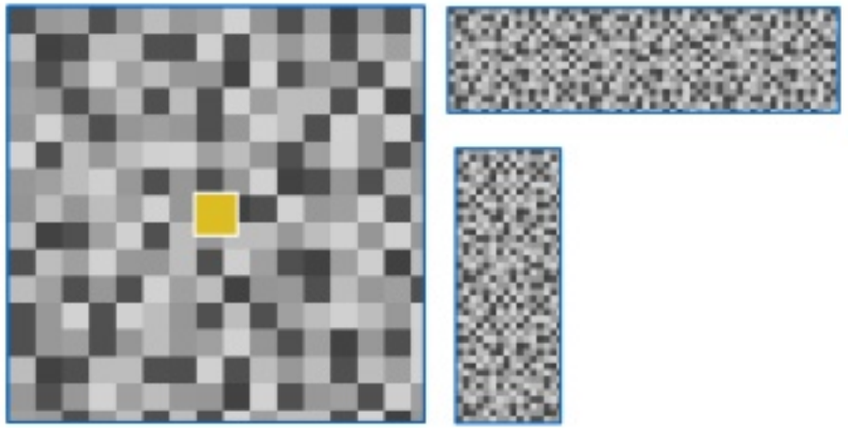
$$b_y \leftarrow b_y + \gamma \left(\epsilon_{x,y} - \lambda_y b_y\right)$$

factors

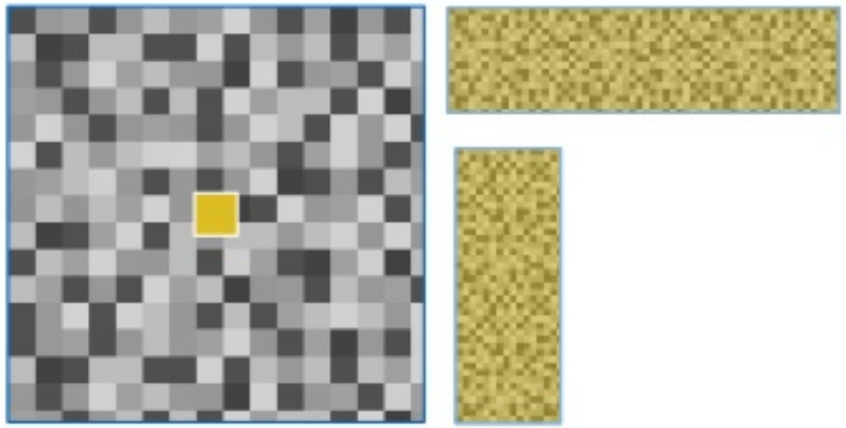
$$\mathsf{U}_x \leftarrow \mathsf{U}_x + \gamma \left(\epsilon_{x,y}\mathsf{P}_y - \lambda'_x \mathsf{U}_x\right)$$

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

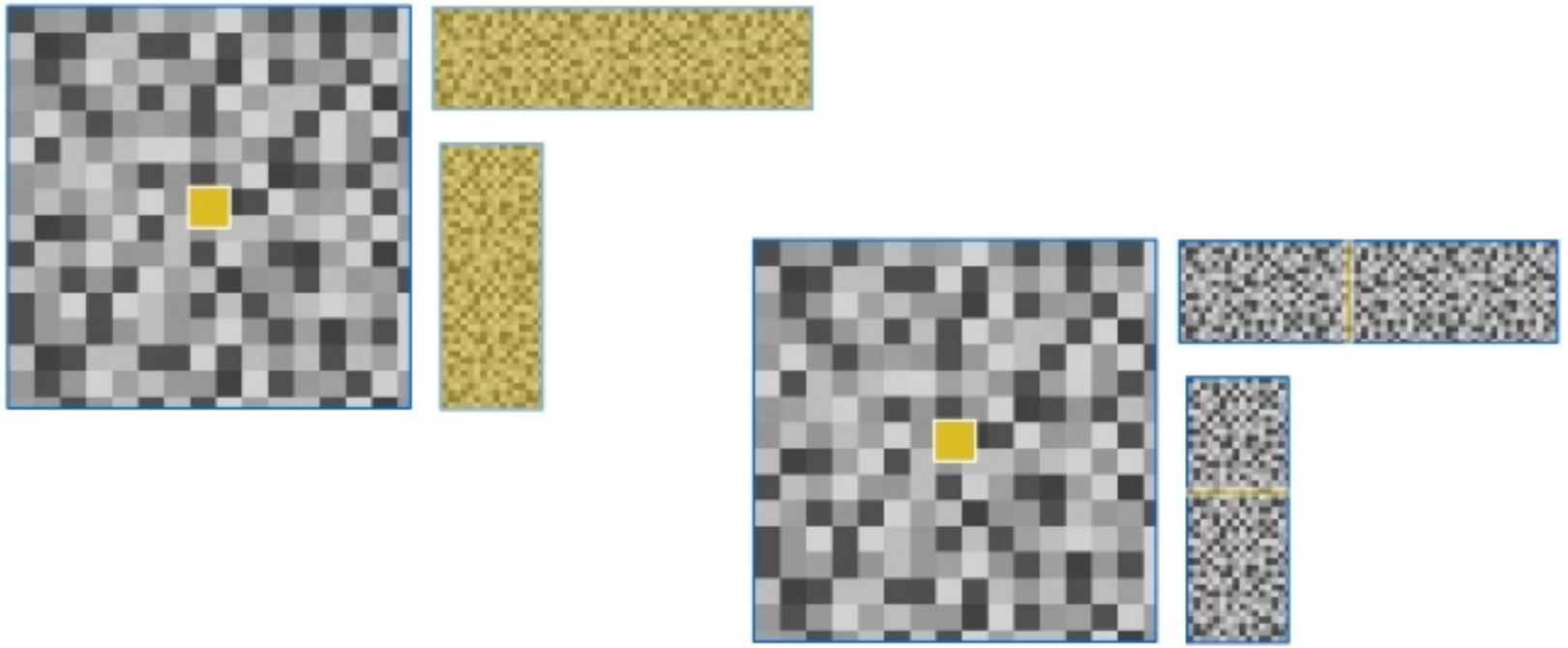
Streaming ALS



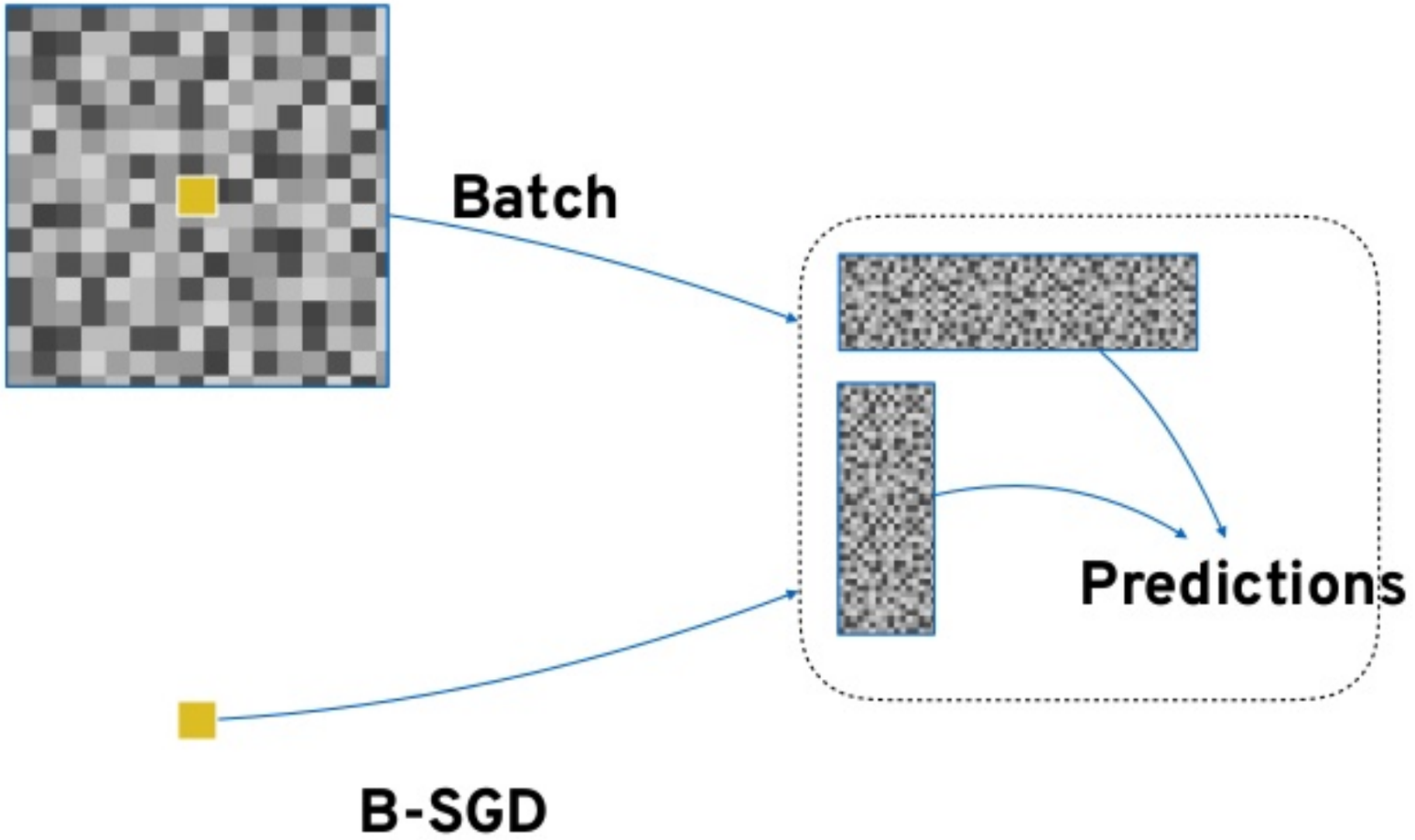
Streaming ALS



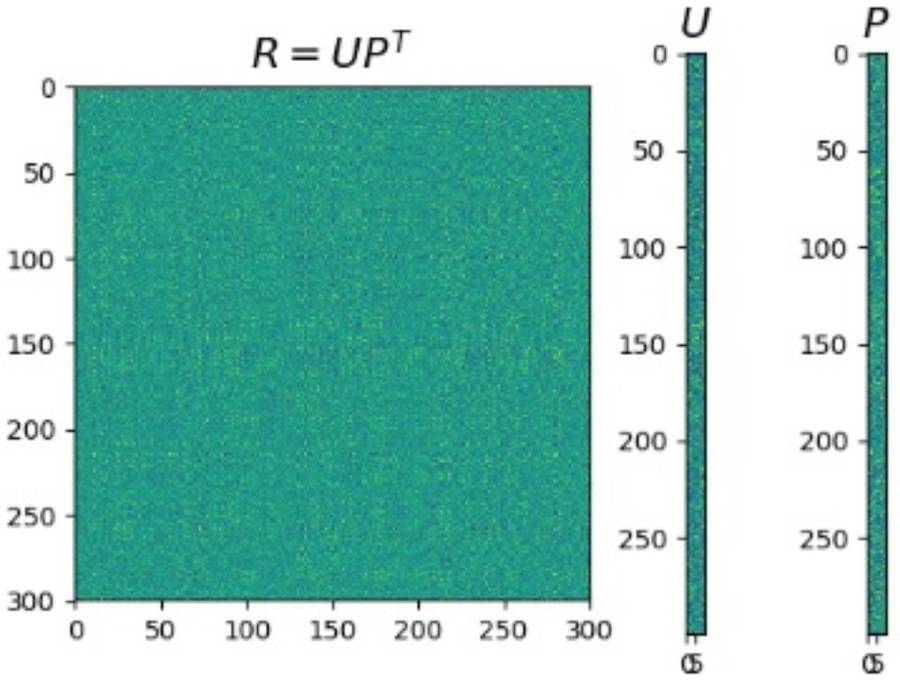
Streaming ALS



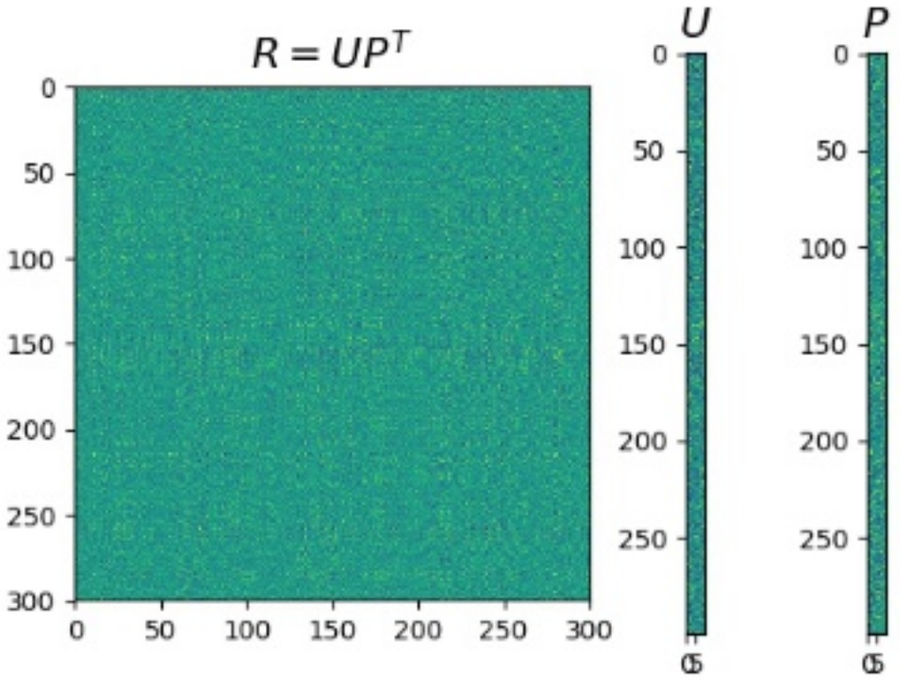
Streaming ALS



Streaming ALS

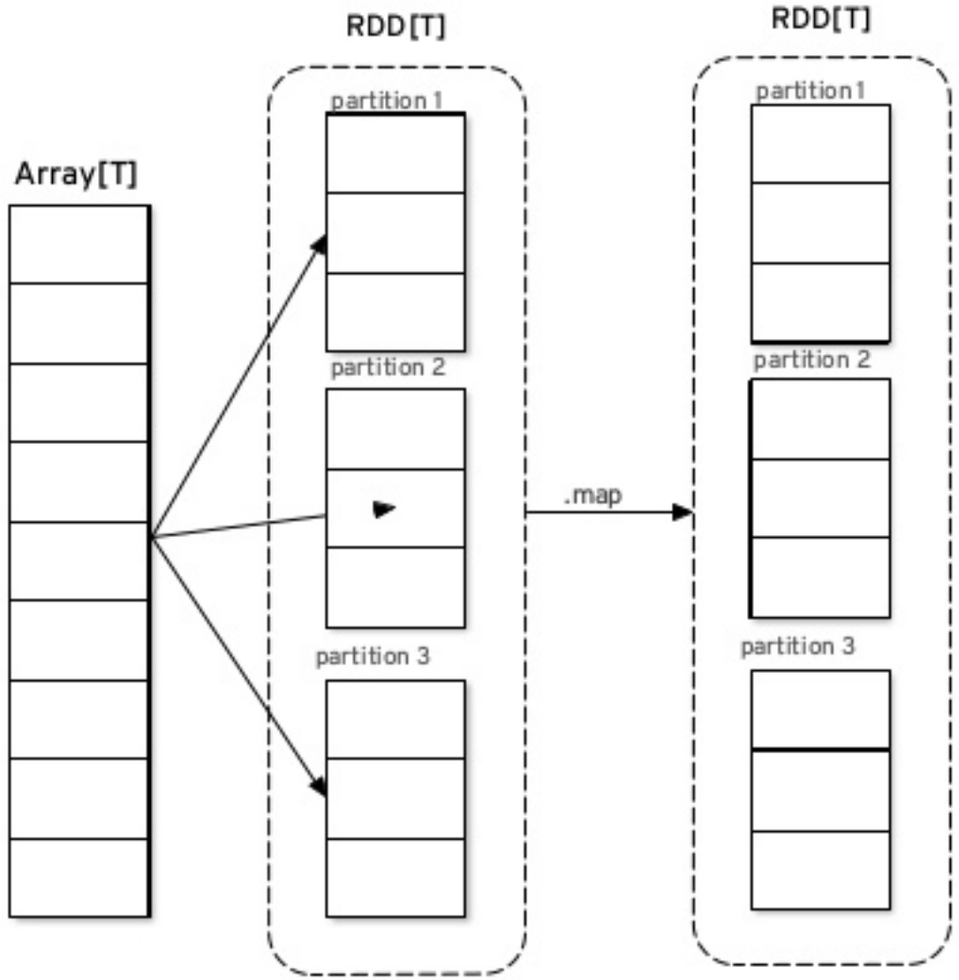


Streaming ALS



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
```

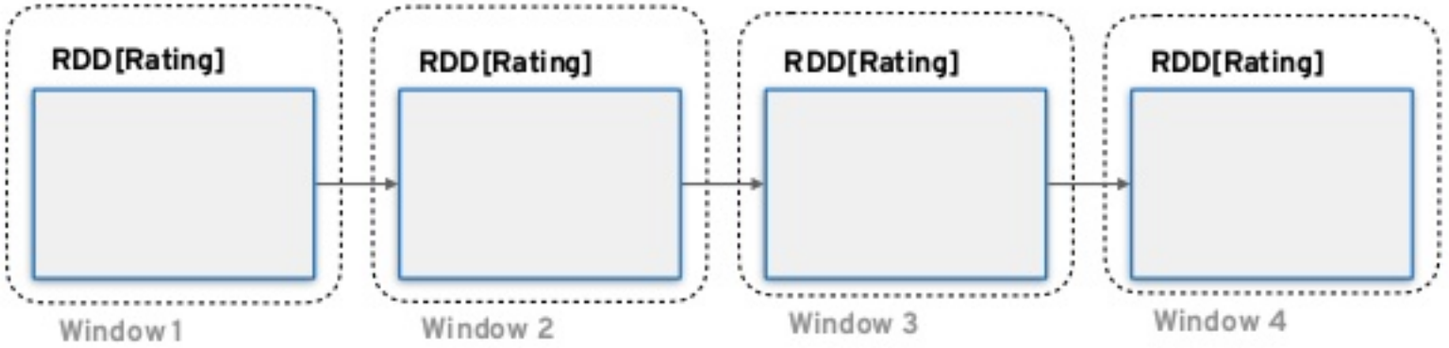

MLlib ALS

```
> 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

RDD[Rating]

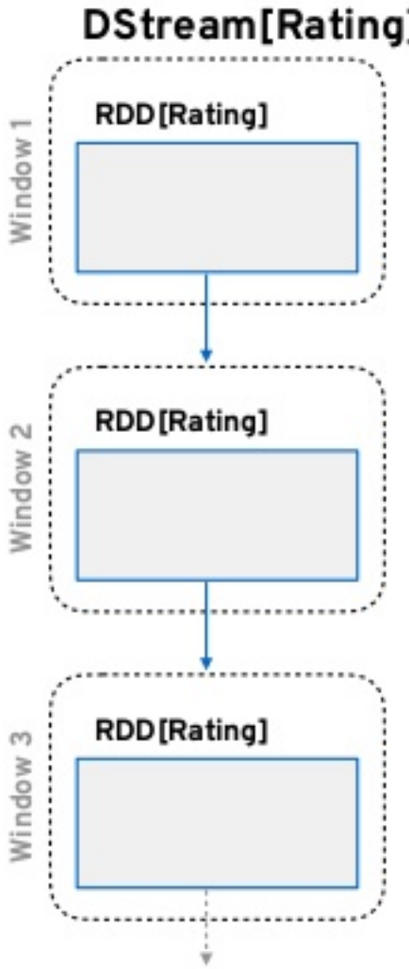
DStream[Rating]



Spark Streaming ALS

DStream[Rating]

Spark Streaming ALS




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

```
model = model.train(rdd2)
```

```
model = model.train(rdd3)
```

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

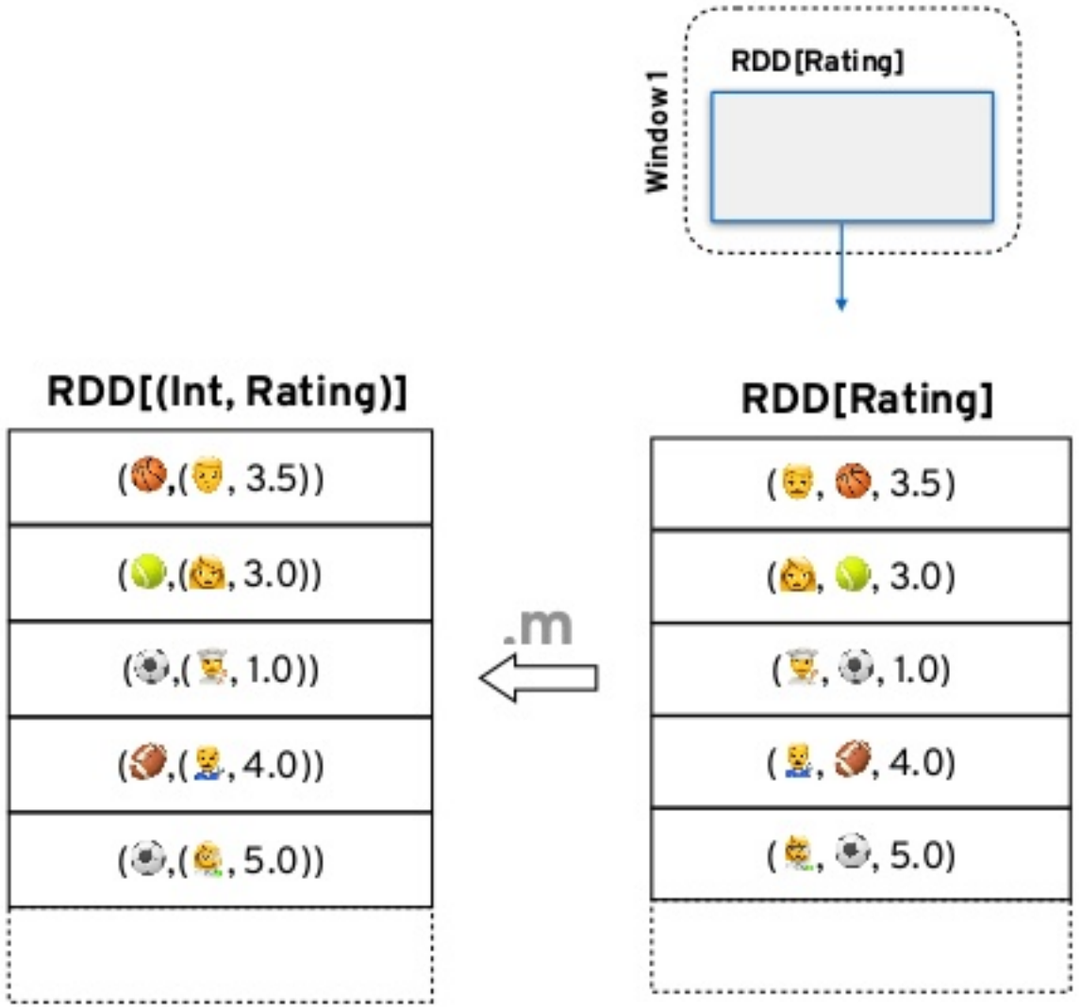
Spark Streaming ALS

Spark Streaming ALS

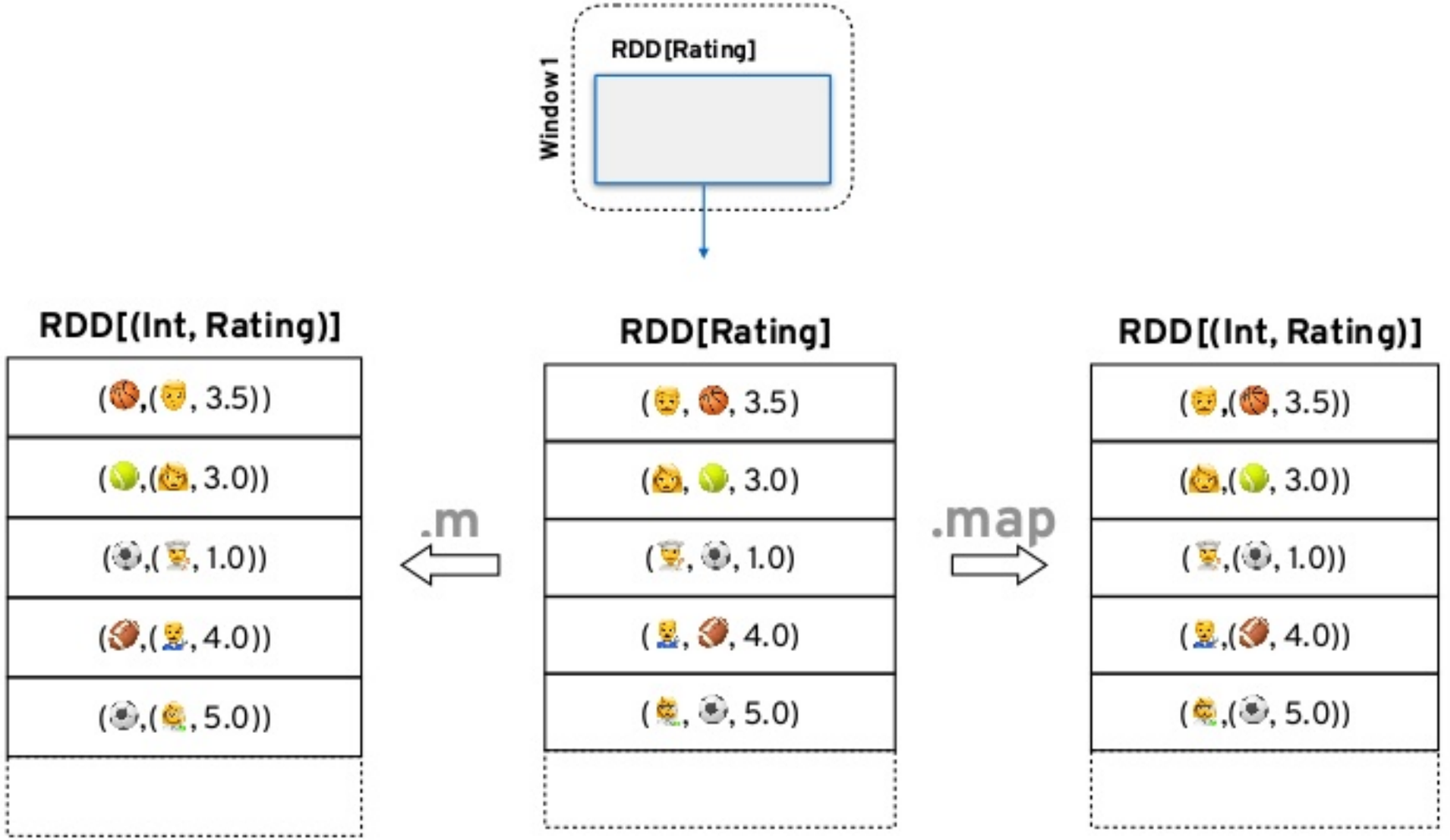
RDD[Rating]

(👱, 🏀, 3.5)
(👩, 🎾, 3.0)
(👨‍🍳, ⚽, 1.0)
(🤹, 🏈, 4.0)
(🤦, ⚽, 5.0)

Spark Streaming ALS



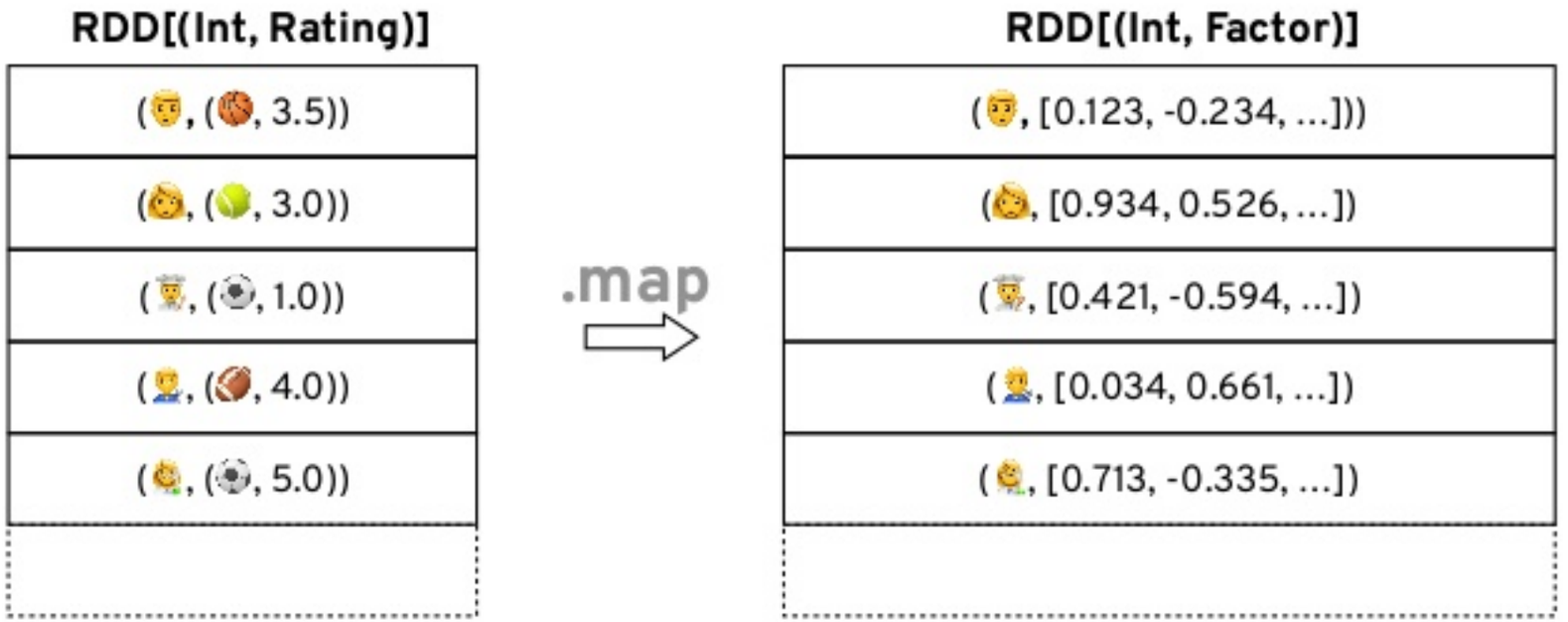
Spark Streaming ALS



Spark Streaming ALS

Spark Streaming ALS

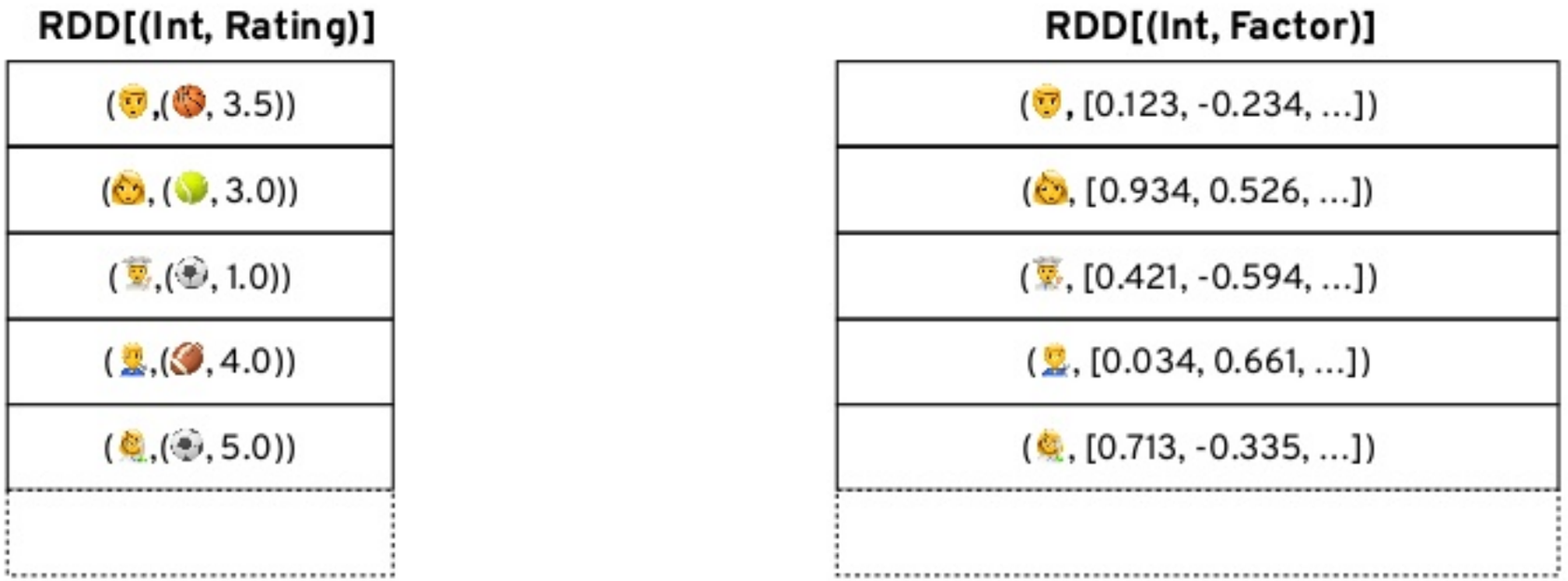
Spark Streaming ALS



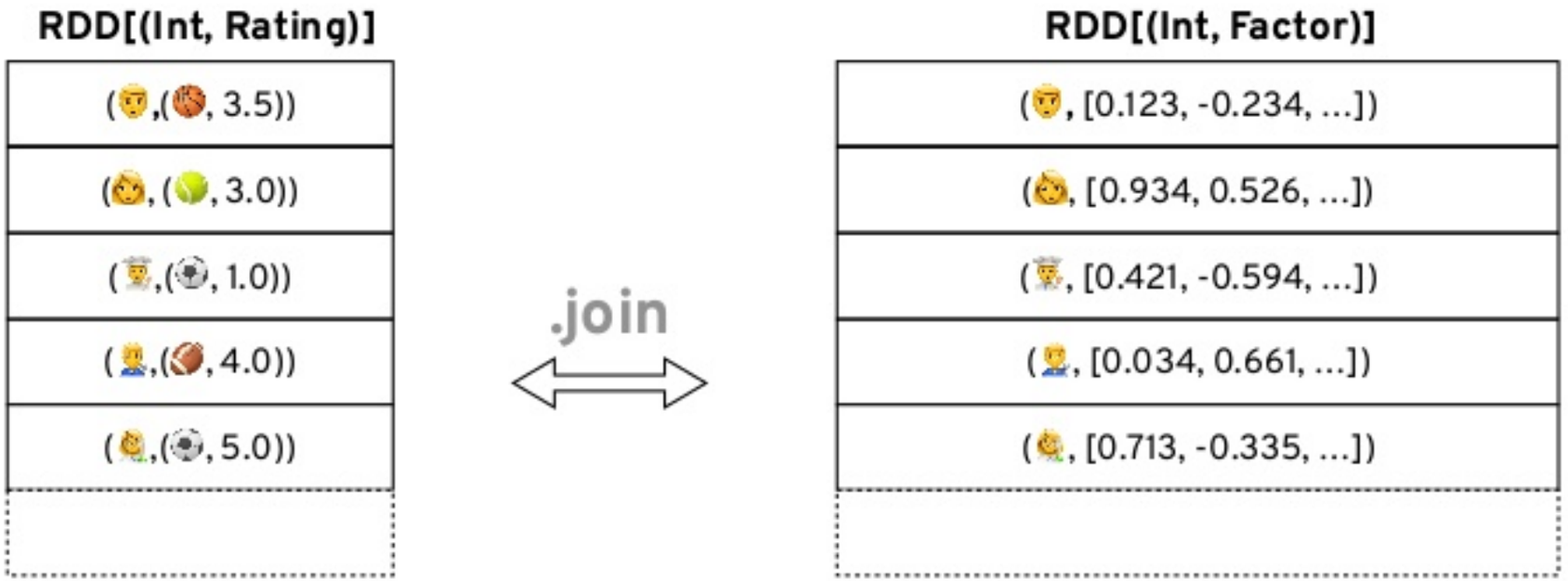
Spark Streaming ALS

Spark Streaming ALS

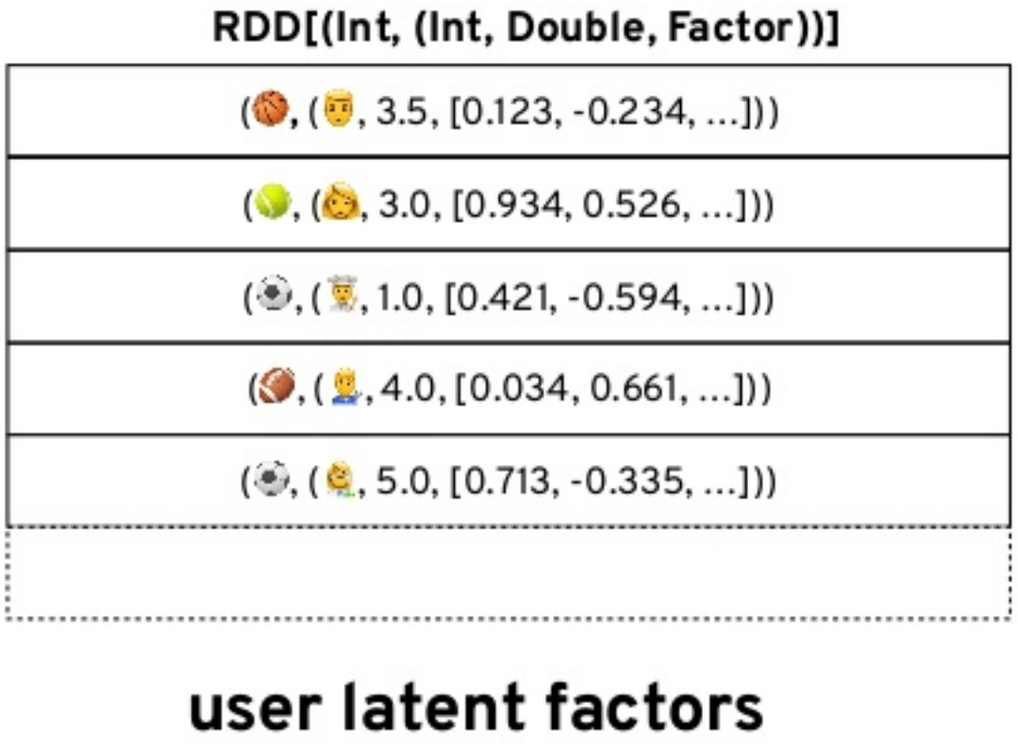
Spark Streaming ALS



Spark Streaming ALS



Spark Streaming ALS



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

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

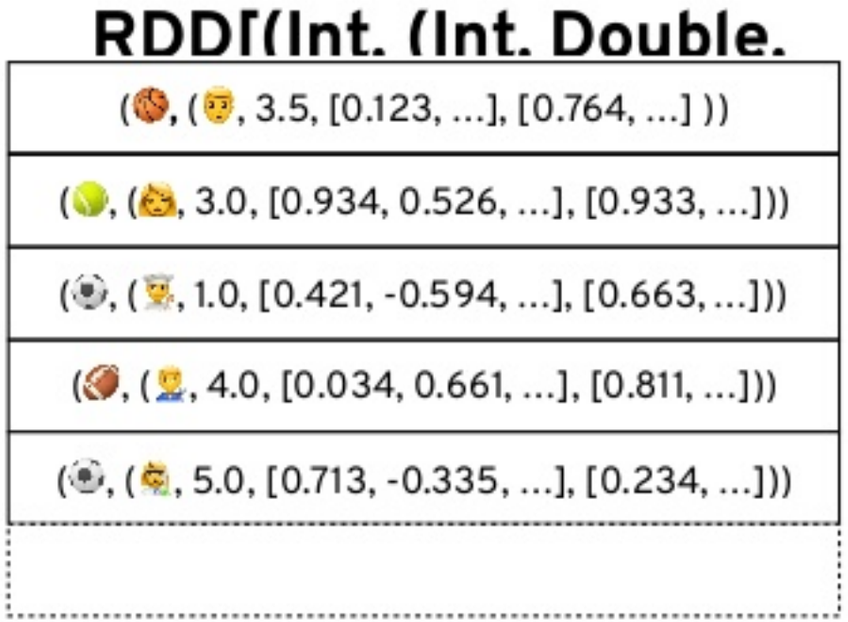
product latent factors

calculate the global bias

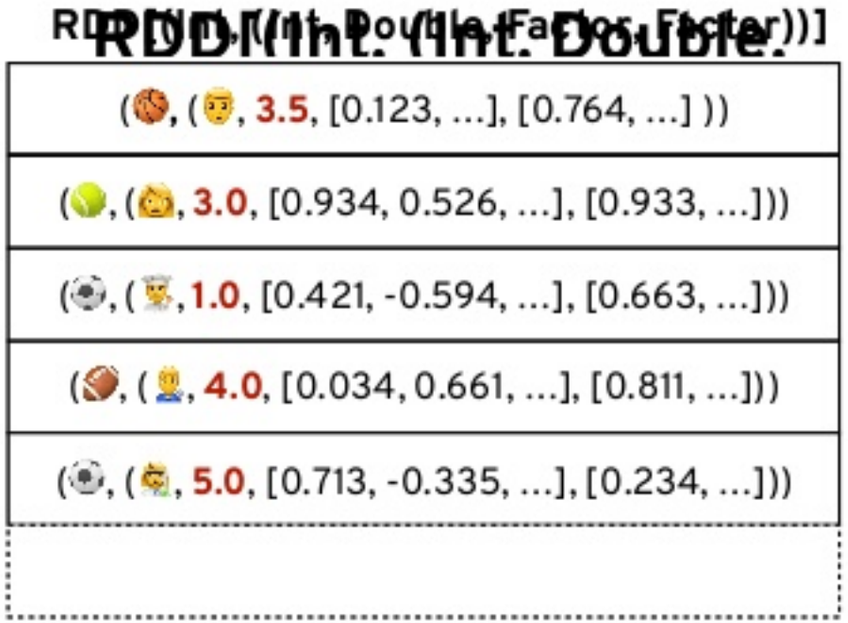
calculate user specific bias

calculate product specific bias

Spark Streaming ALS



Spark Streaming ALS



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(⚽, 👨) = $\mu + b_x + b_y$ + [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

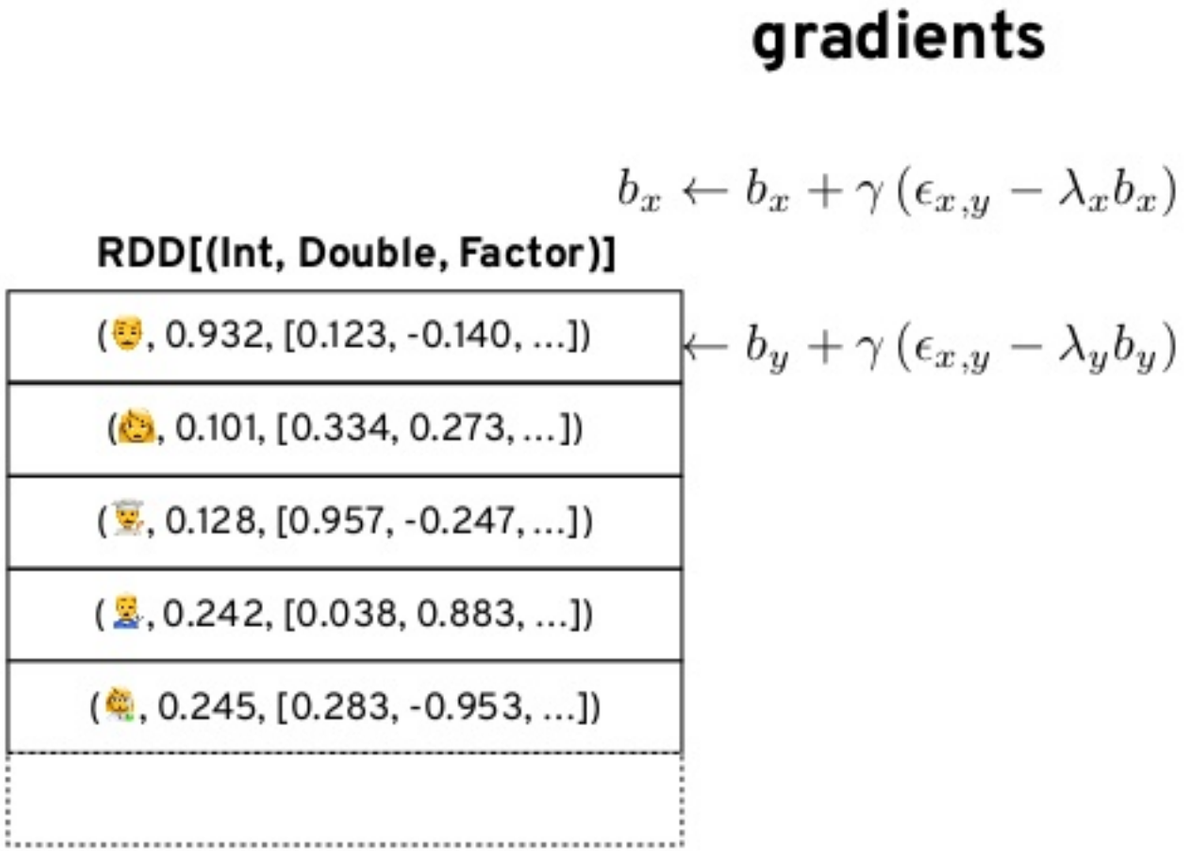
Spark Streaming ALS

gradients

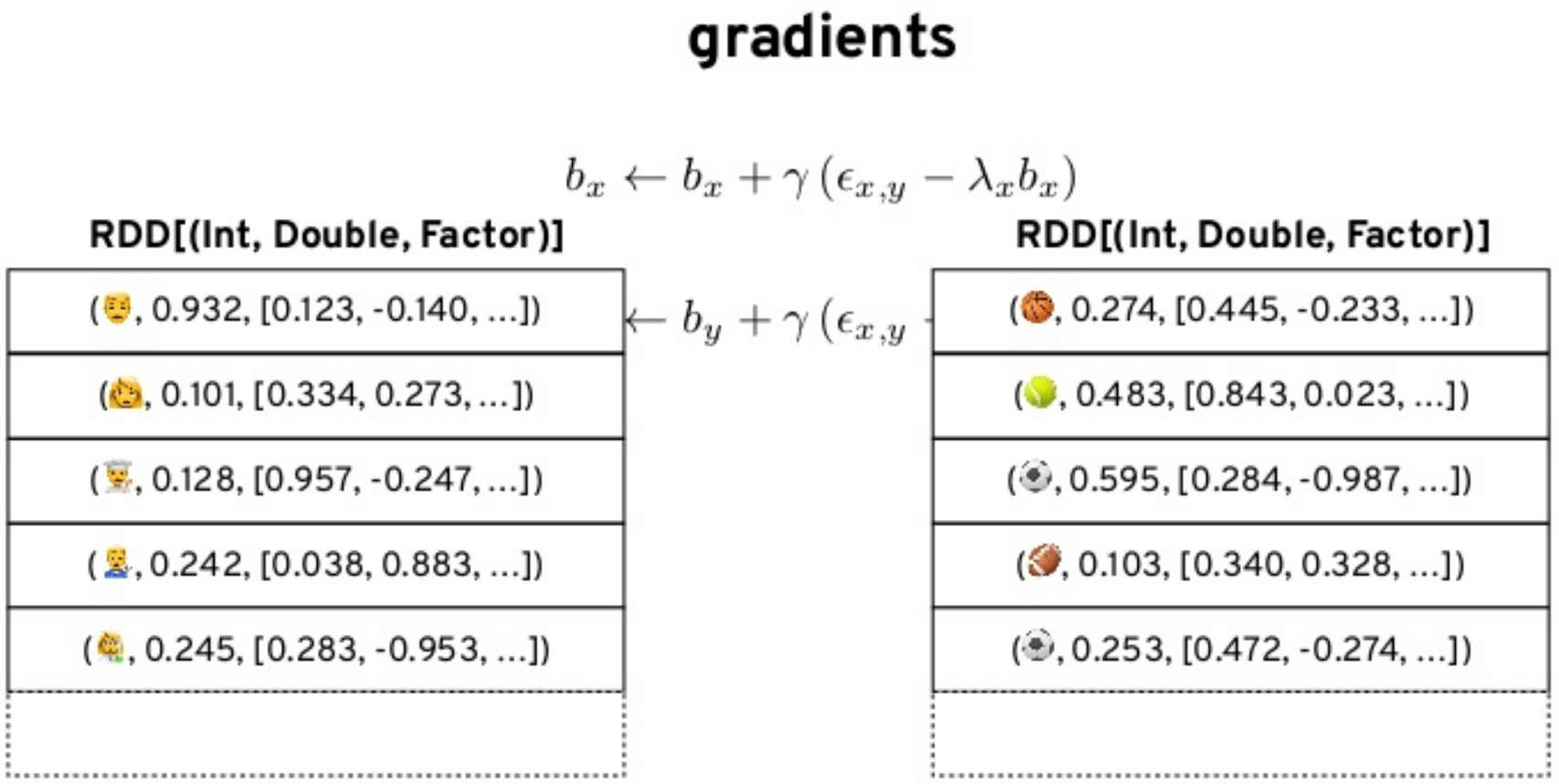
$$b_x \leftarrow b_x + \gamma \left(\epsilon_{x,y} - \lambda_x b_x\right)$$

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

Spark Streaming ALS

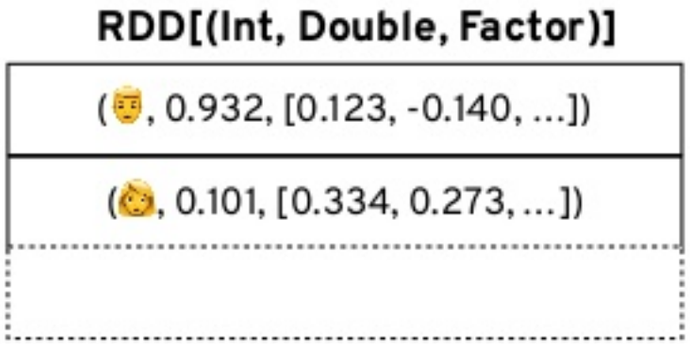


Spark Streaming ALS

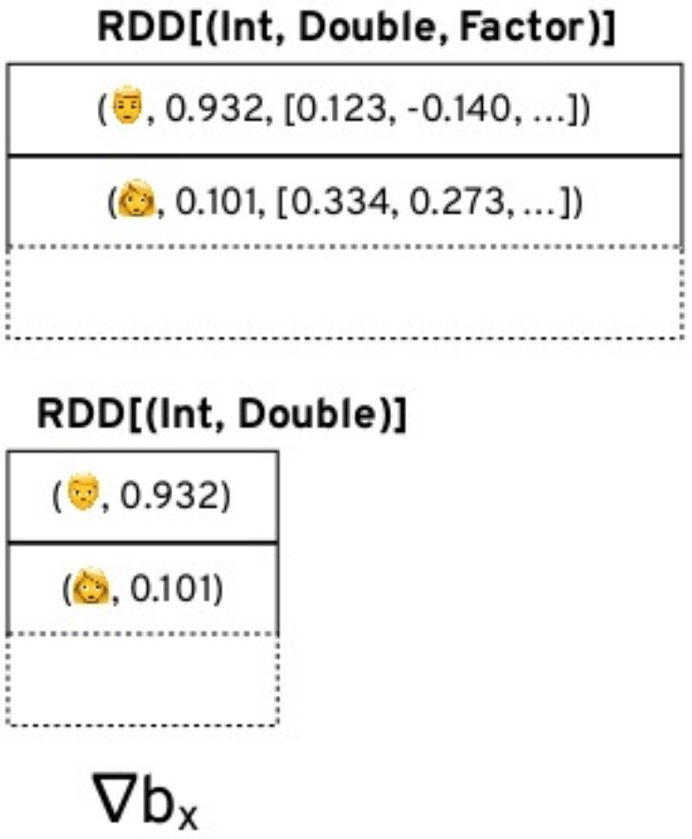


Spark Streaming ALS

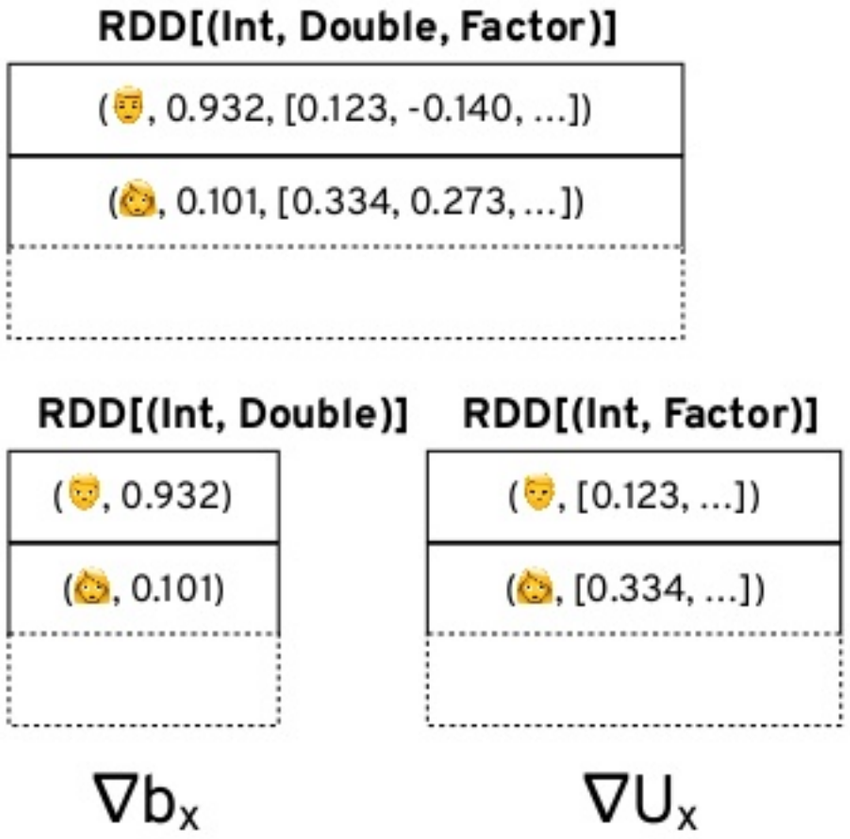
Spark Streaming ALS



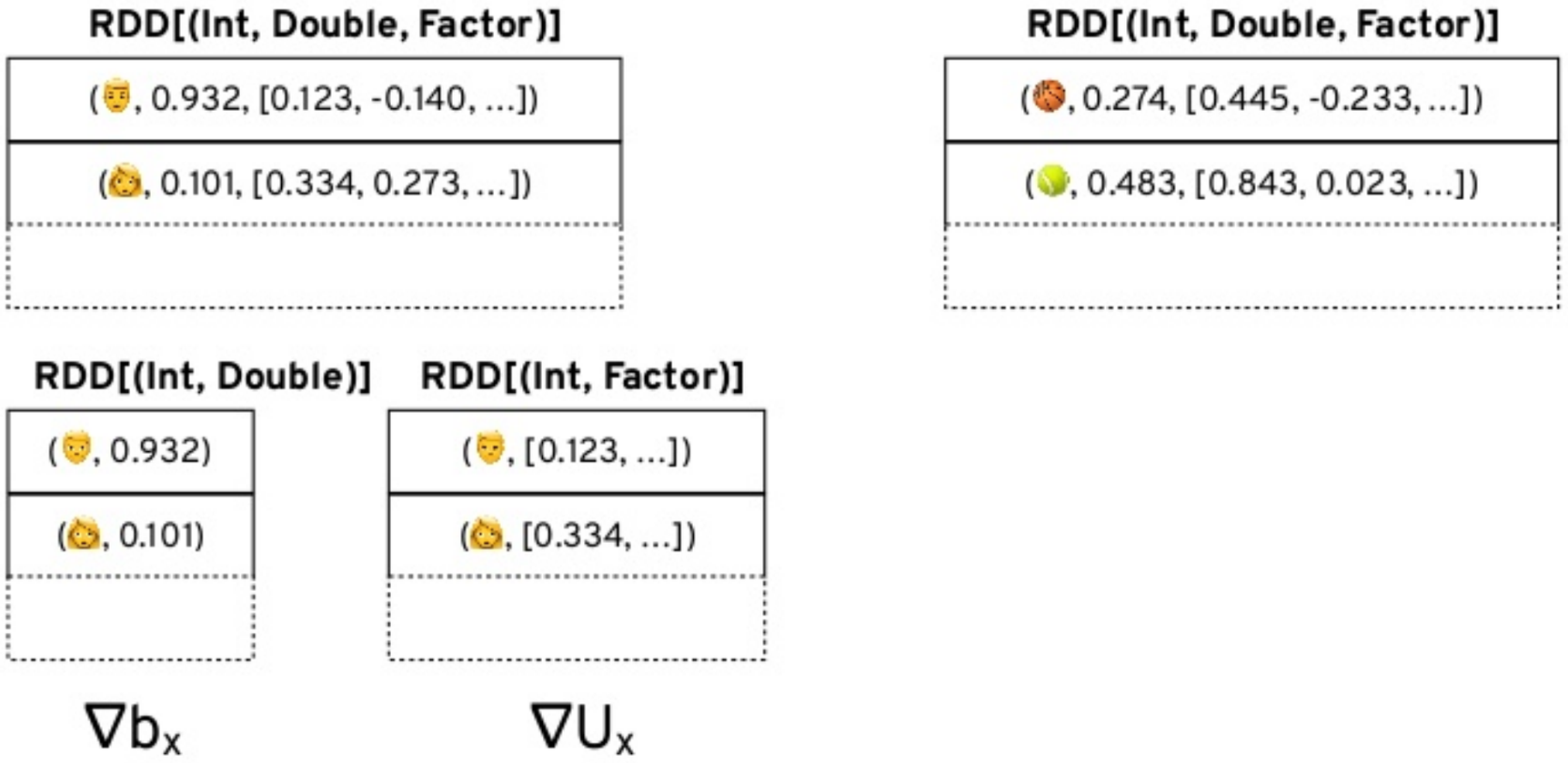
Spark Streaming ALS



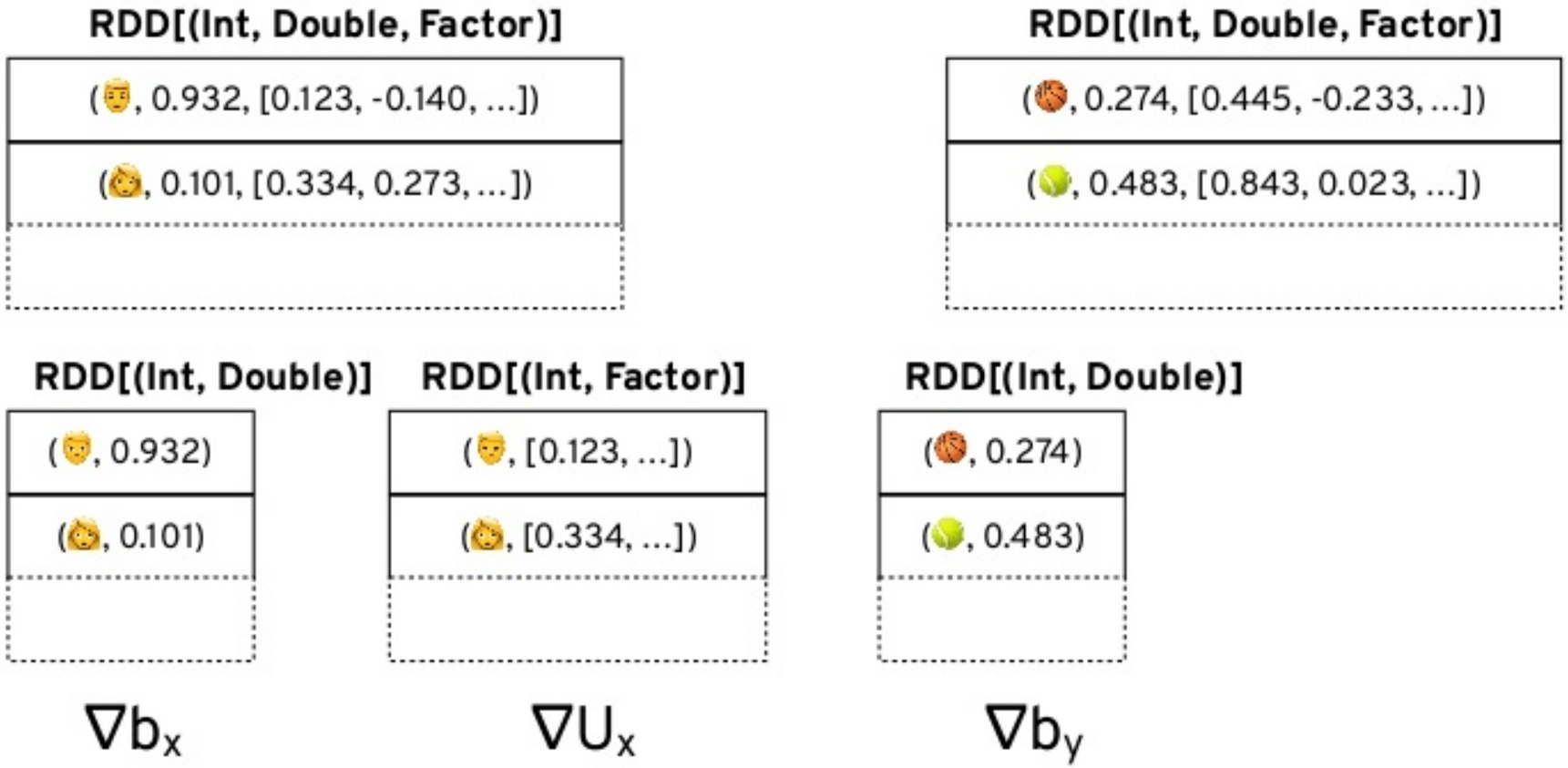
Spark Streaming ALS



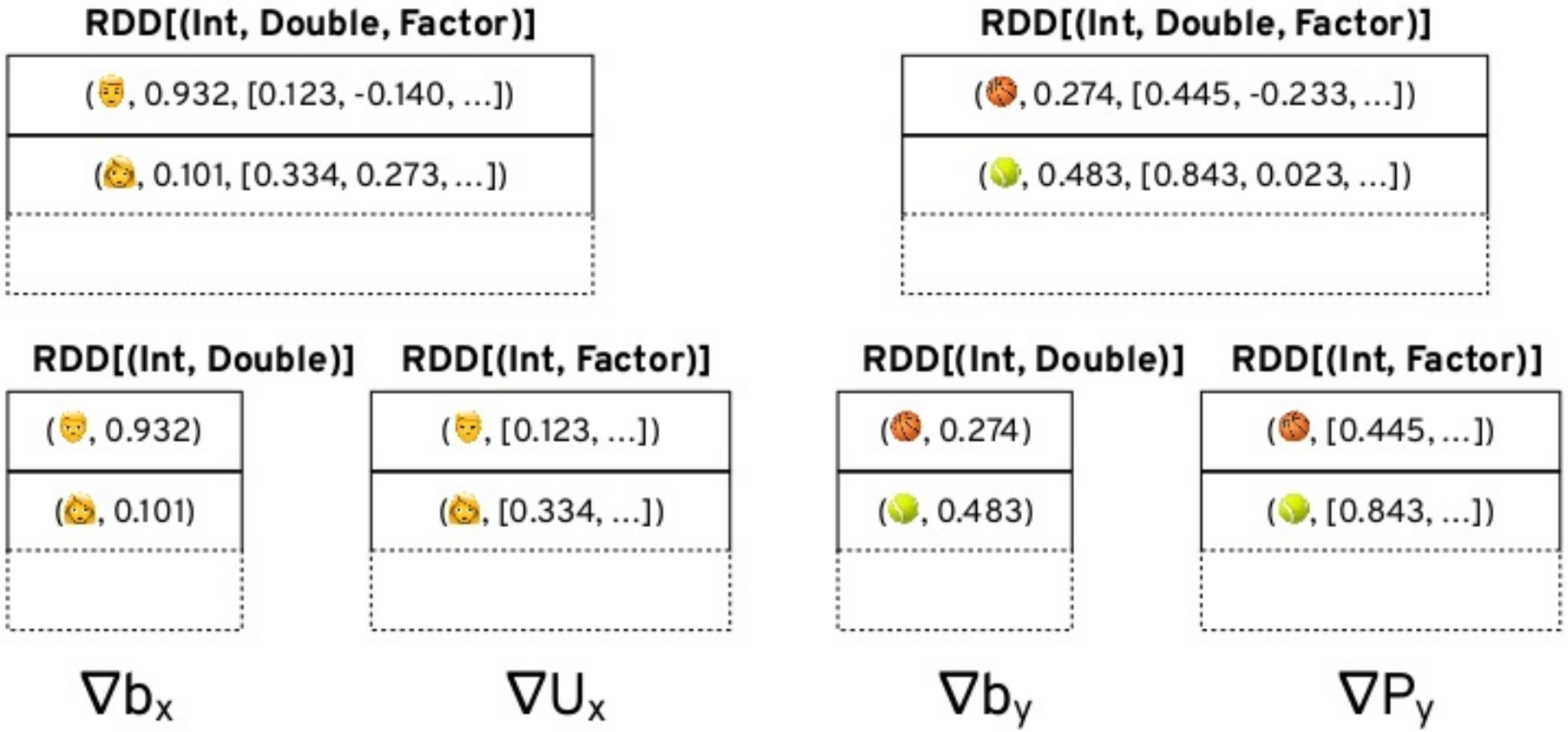
Spark Streaming ALS



Spark Streaming ALS

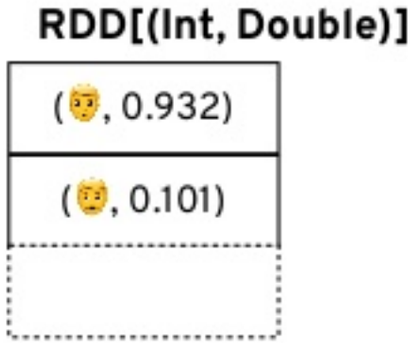


Spark Streaming ALS



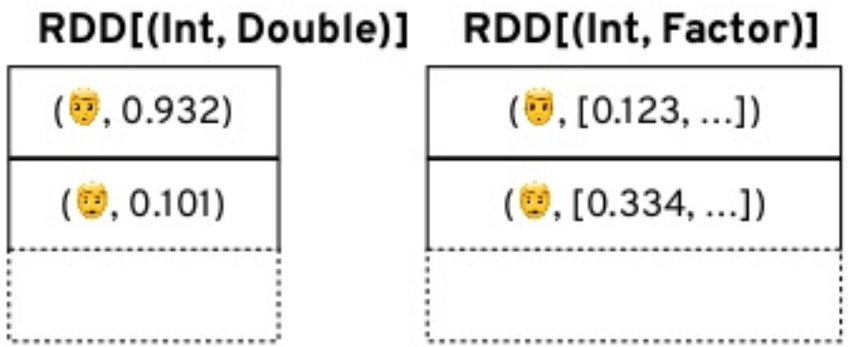
Spark Streaming ALS

Spark Streaming ALS

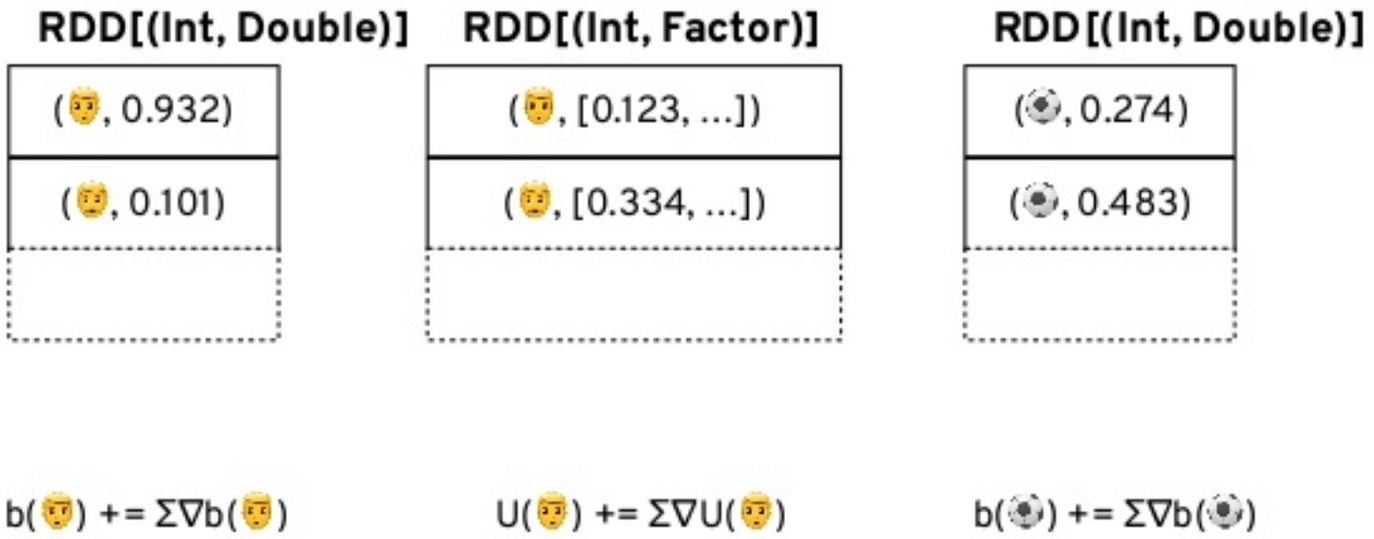


b(👨) += Σ∇b(👨)

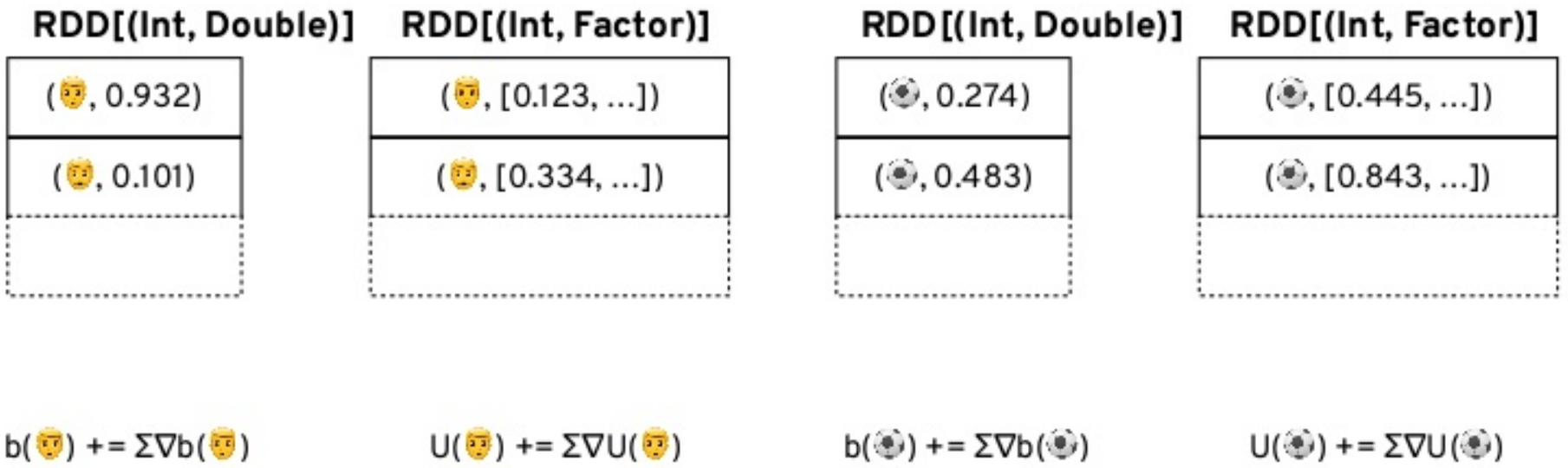
Spark Streaming ALS



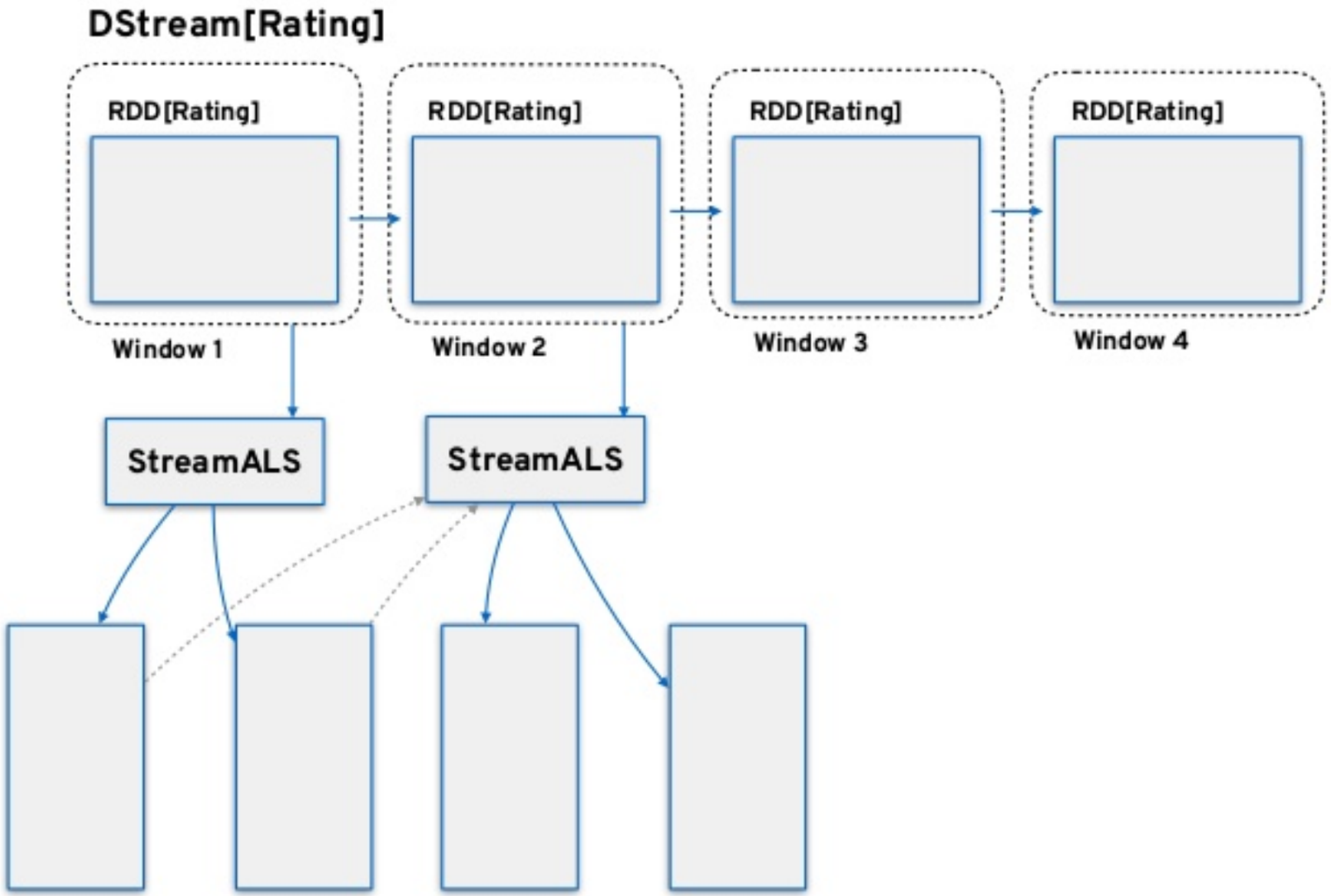
Spark Streaming ALS



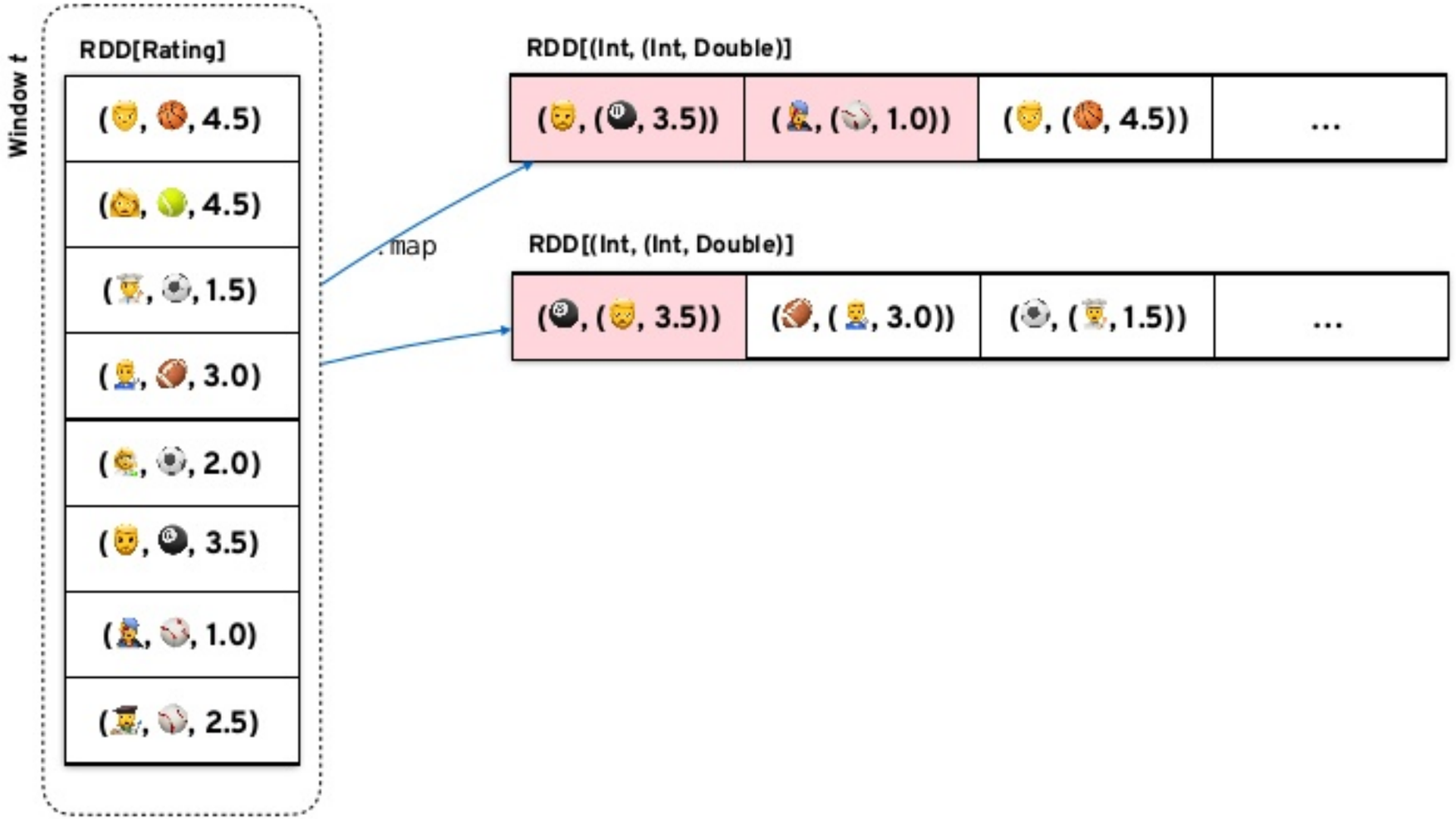
Spark Streaming ALS



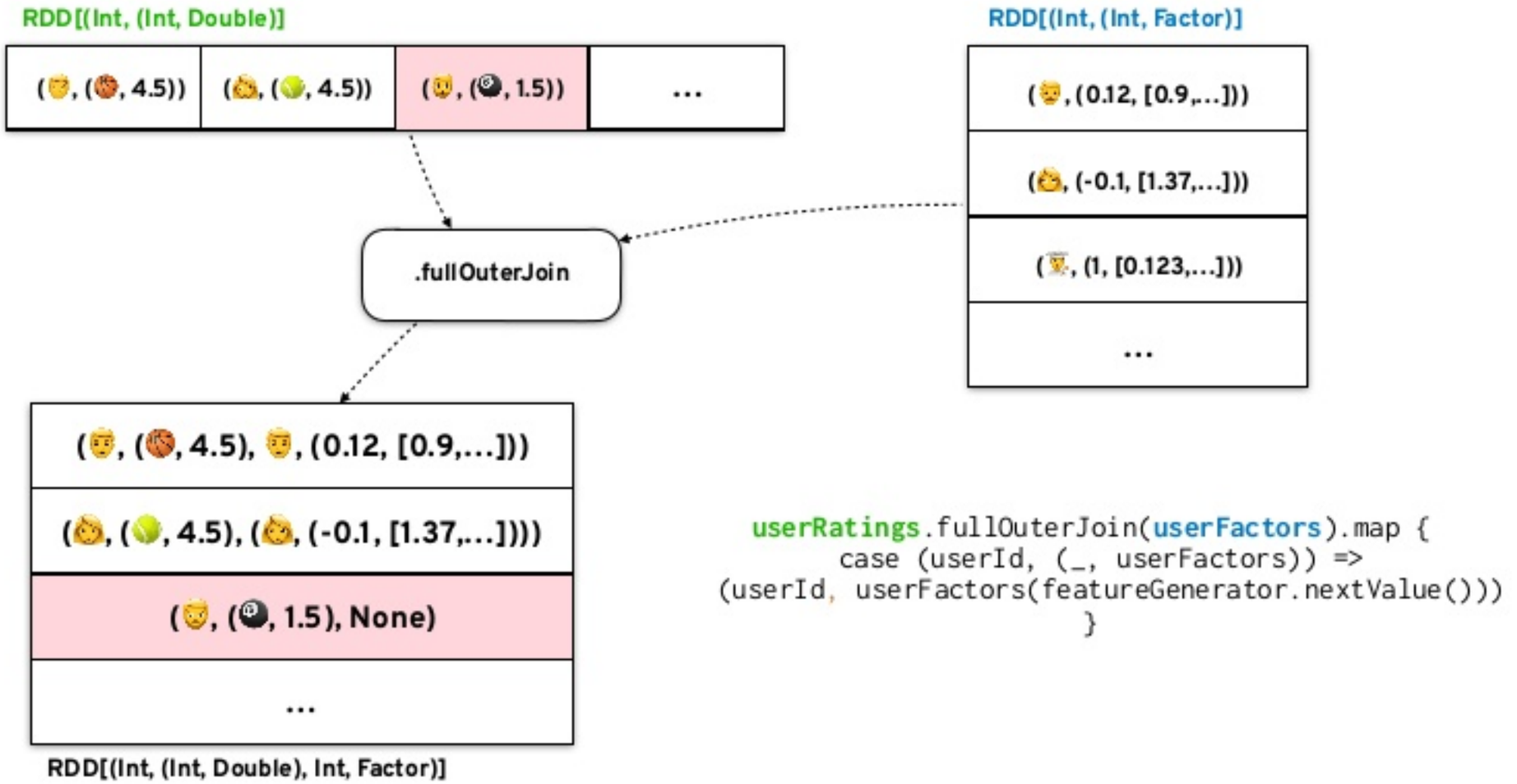
Spark Streaming ALS



Spark Streaming ALS



Spark Streaming ALS




```
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

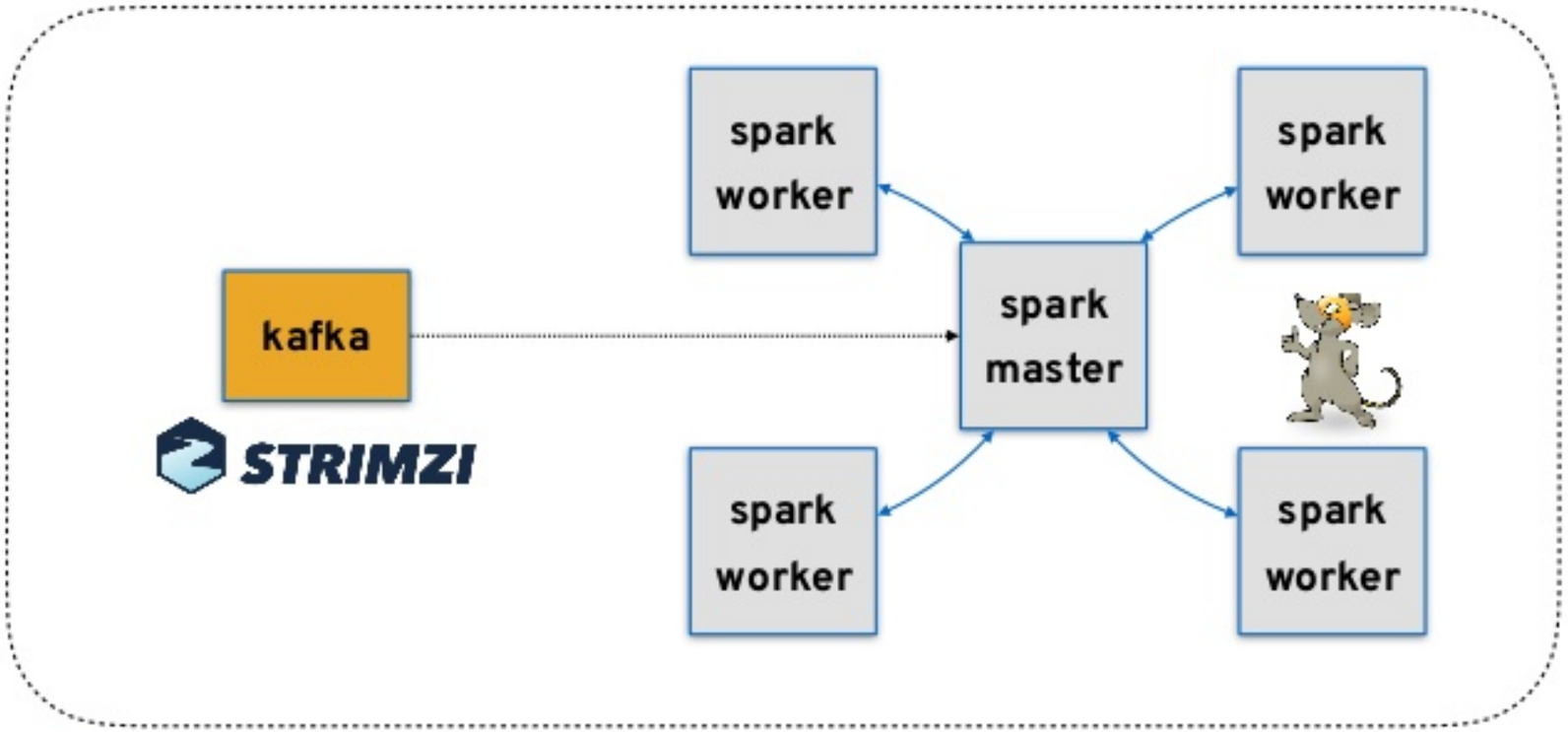
Training batch ALS

```
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

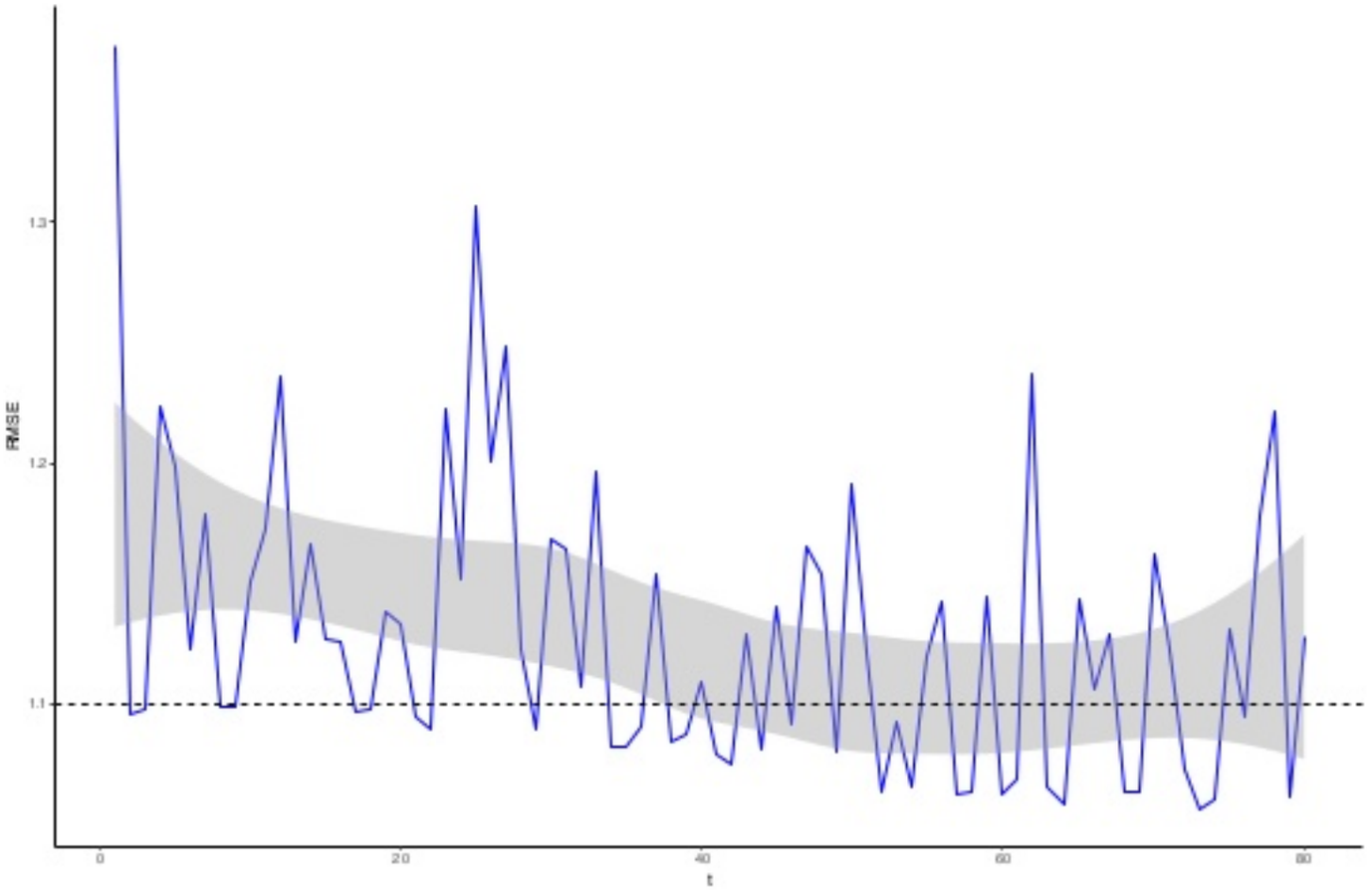


RED HAT OPENSHIFT

Training streaming ALS


```
val model = StreamingALS(rank, iterations, lambda, gamma)
trainingStreamSet.foreachRDD { rdd =>
 model.train(rdd)
 val RMSE = calculateRMSE(model, validation)
}
```

Comparison



TO CONSIDER...

To consider

"Cold start"

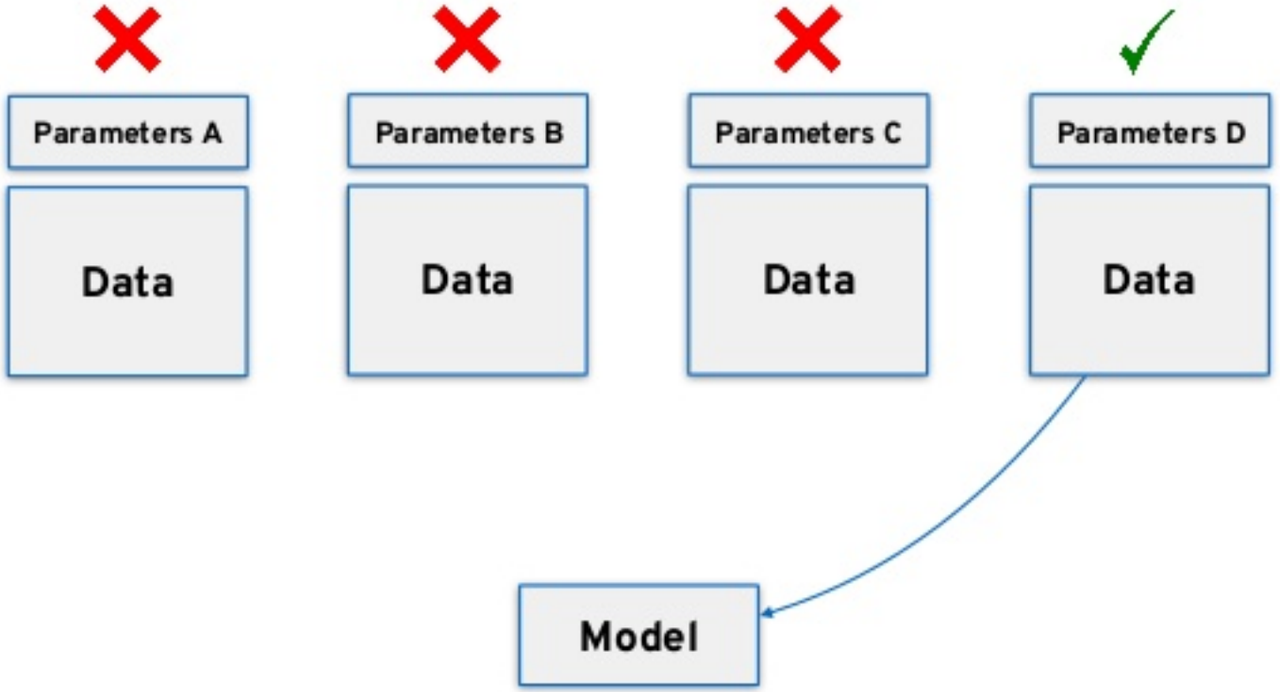
Same as batch ALS

Too few observations = meaningless

Train offline

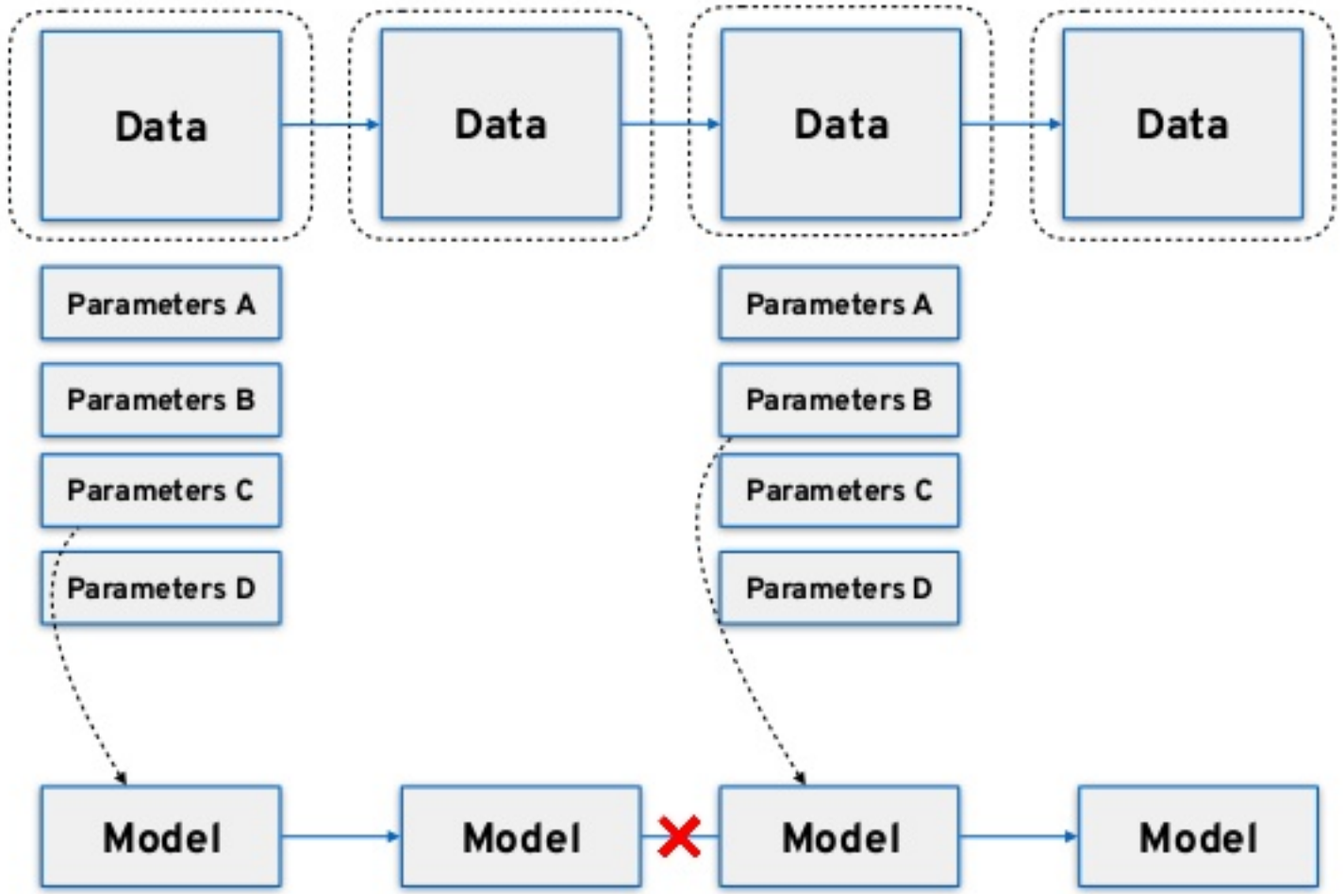
To consider

Hyper-parameter estimation



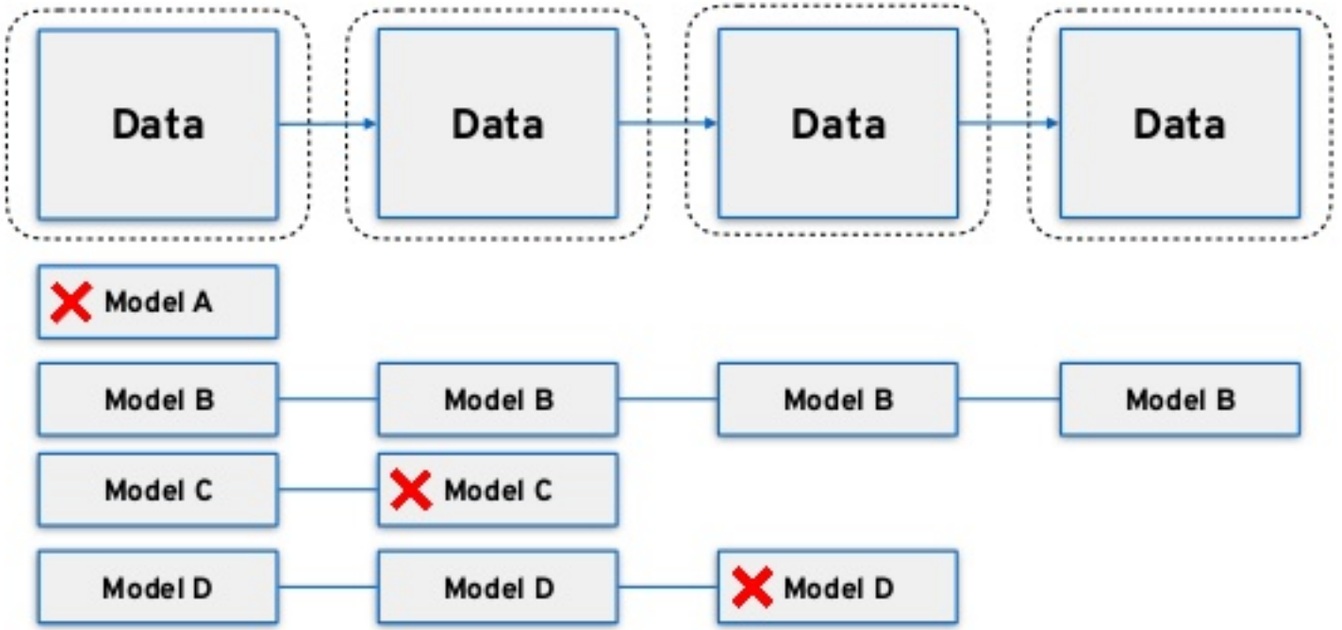
To consider

Hyper-parameter estimation



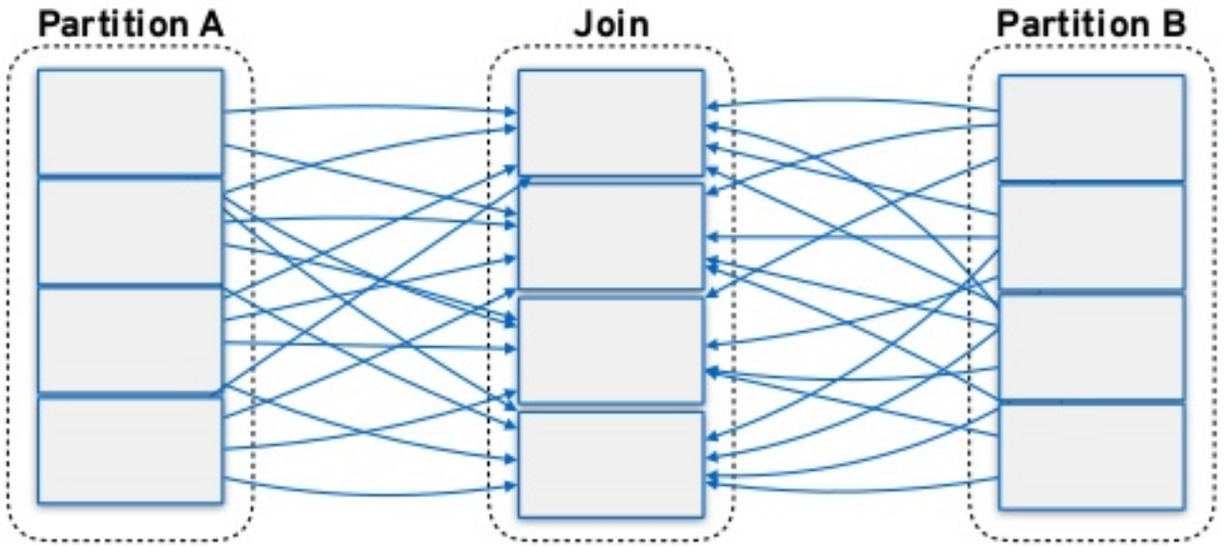
To consider

Hyper-parameter estimation



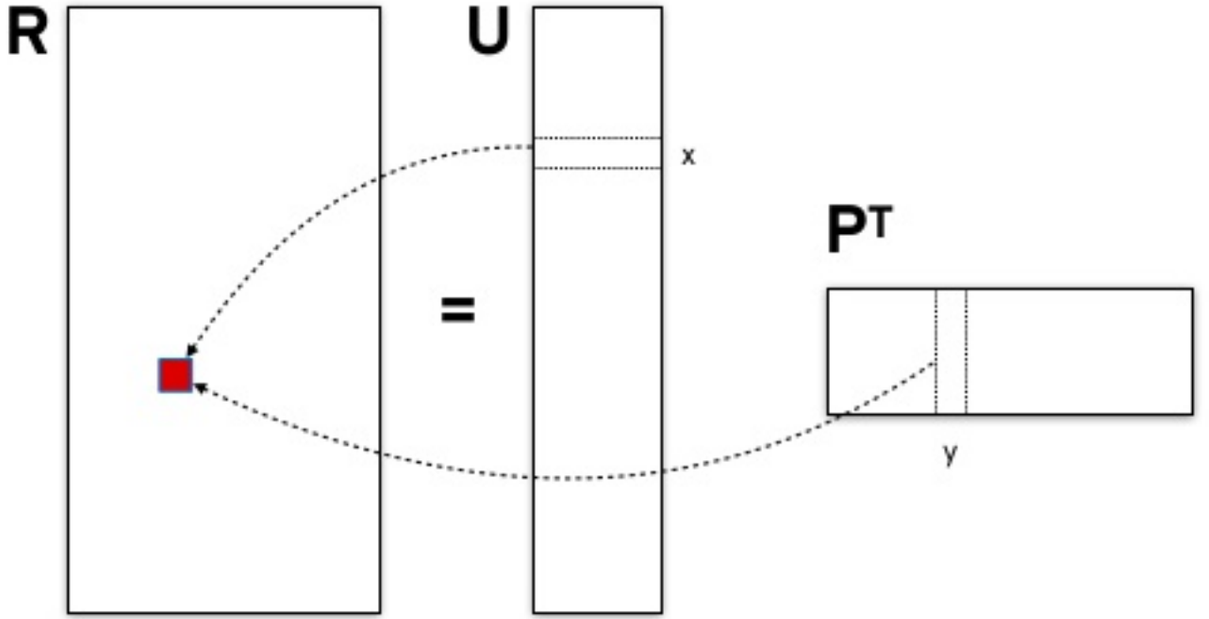
To consider

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 post:

https://ruivieira.github.io/a-streaming-als-implementation.html

radanalytics.io


SPARK+AI
SUMMIT EUROPE
THANK YOU