Recommending Products Based on the Latent Factors Model

Swetha Kolalapudi

CO-FOUNDER, LOONYCORN

www.loonycorn.com

Overview

Understand the latent factors model for collaborative filtering

Contrast the latent factors model and the nearest neighbors model

Use optimization techniques to solve for latent factors

- Stochastic gradient descent

Collaborative Filtering Techniques

Nearest Neighbors Model

Use the ratings of “most similar” users

Latent Factor Analysis

Solve for underlying factors that drive the ratings

Latent Factor Analysis

Analogous to PCA (Principal Components Analysis)

in matrix algebra

Latent Factor Analysis

Why are some products rated high and others low?

Why do some users like certain products?

Are there some underlying factors that influence all users' ratings?

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

Once underlying factors are known

Predict any rating for a product by a user

Latent Factor Analysis

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Latent Factor Analysis

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	2	4	3
U_2	3	2	5	1	5
U_3	4	2	3	5	4
U_4	1	2	4	3	5
U_5	1	4	2	5	3
U_6	3	4	3	1	5

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Users described by some underlying factors

Latent Factor Analysis

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	2	4	3
U_2	3	2	5	1	5
U_3	4	2	3	5	4
U_4	1	2	4	3	5
U_5	1	4	2	5	3
U_6	3	4	3	1	5

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Products described by the same underlying factors

Latent Factor Analysis

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

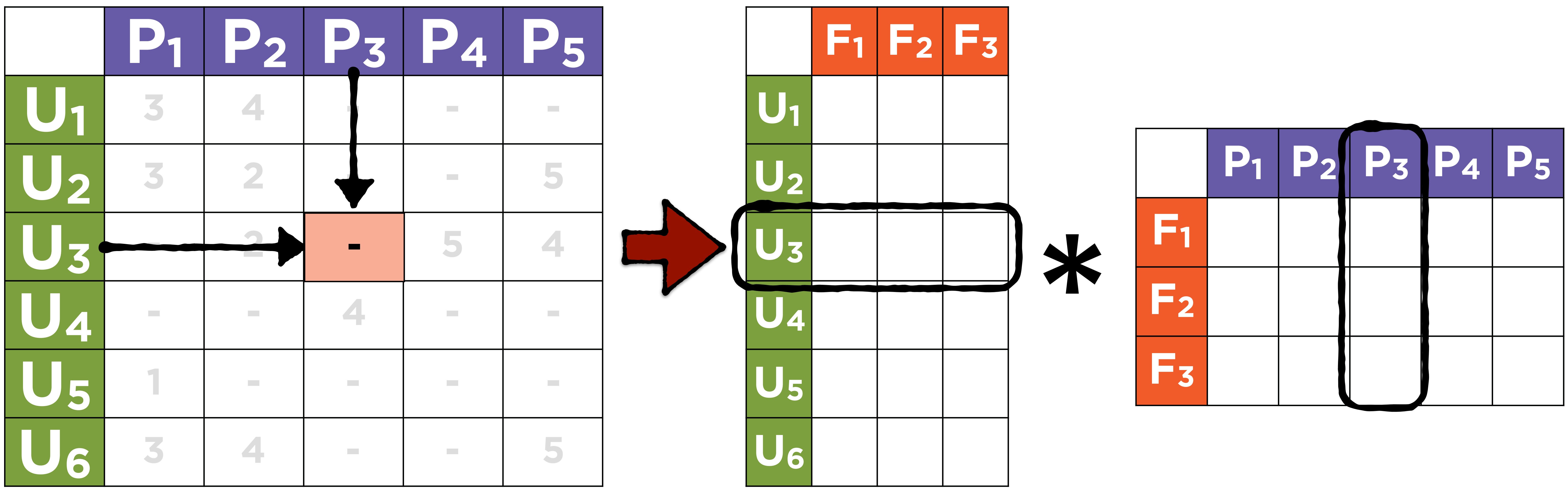
	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Solve using known ratings

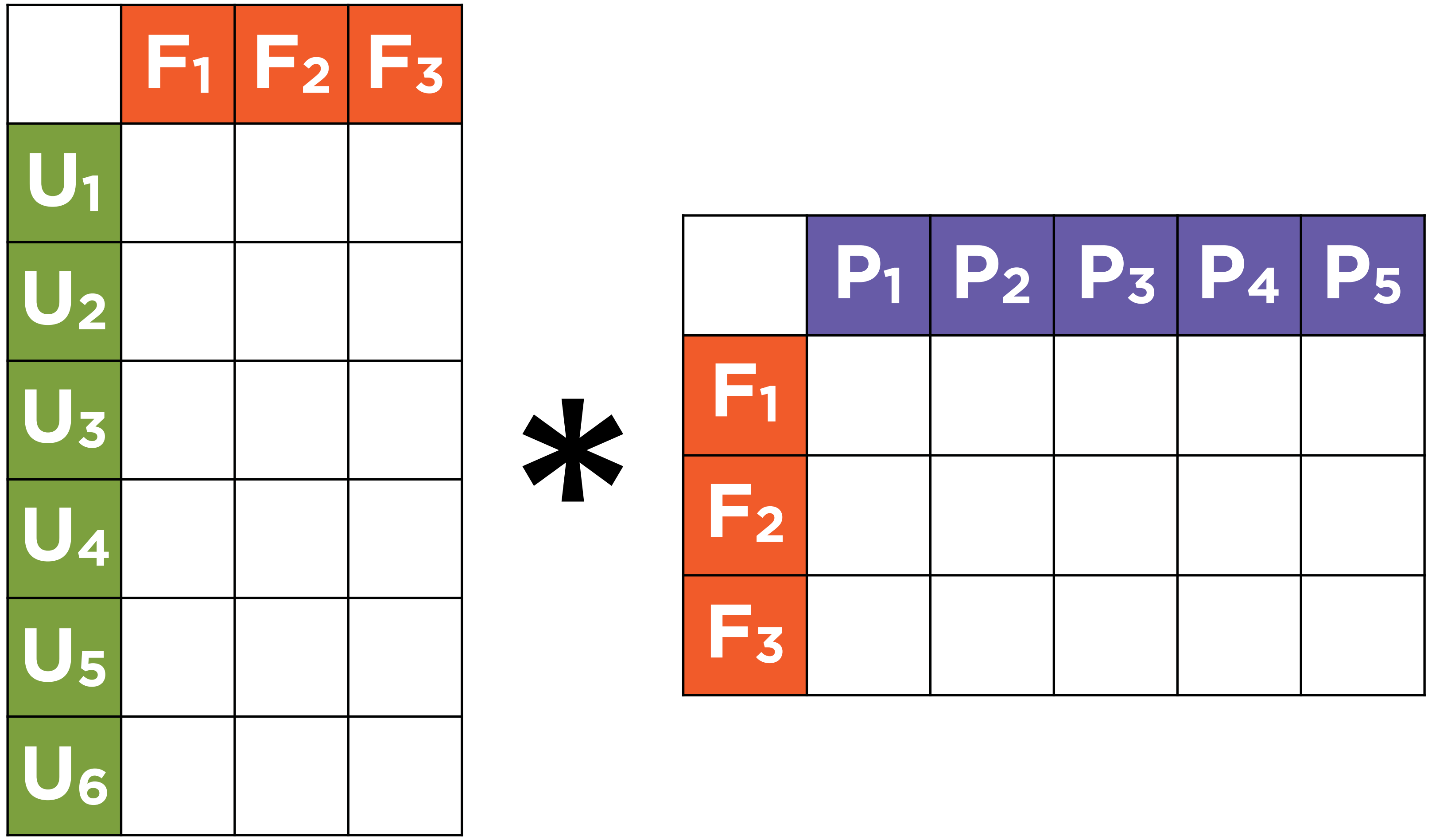
Latent Factor Analysis



Use the solution to find unknown ratings

Latent Factor Analysis

This representation is analogous to content based filtering



Content Based Filtering

Rate every product against the relevant attributes

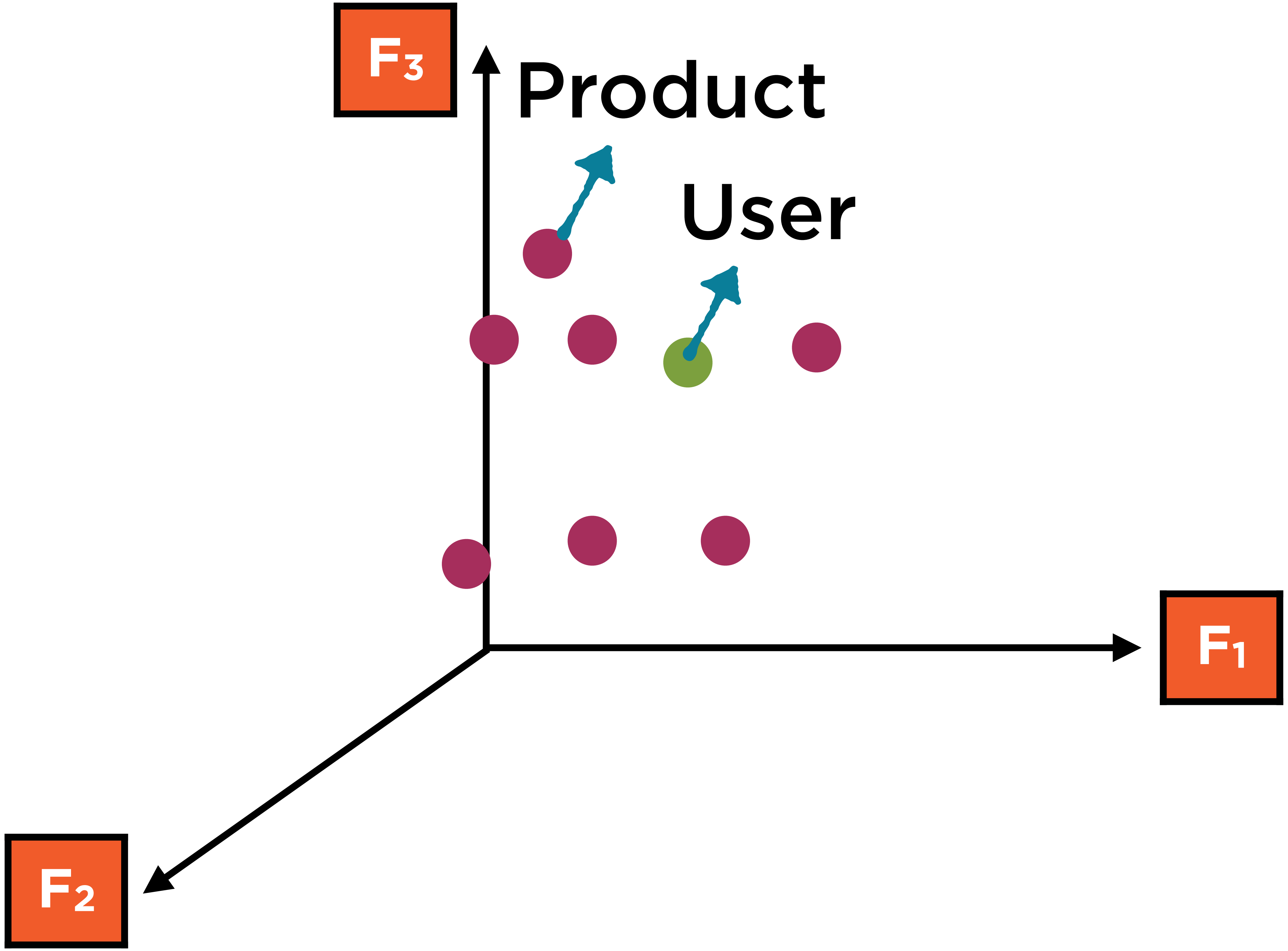
Product	F1	F2	F3	F4
A	0	3	2	5
B	5	2	3	4
C	4	5	2	1
D	3	4	5	2

Rate the user on the importance he/she gives to these factors

User	F1	F2	F3	F4
A	0	3	2	5

Ex : Average of ratings of products that the user already likes

Data Representation



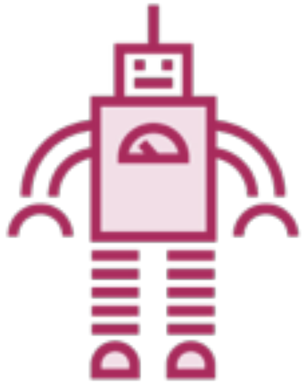
Content Based Filtering vs Latent Factor Analysis

Factors are identified by experts

Factors are product attributes

Factors are derived using machine learning techniques

Factors may be related to product attributes or may be abstract



Contrasting the Nearest Neighbors Model and Latent Factor Analysis

Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Model Updates

Memory Usage

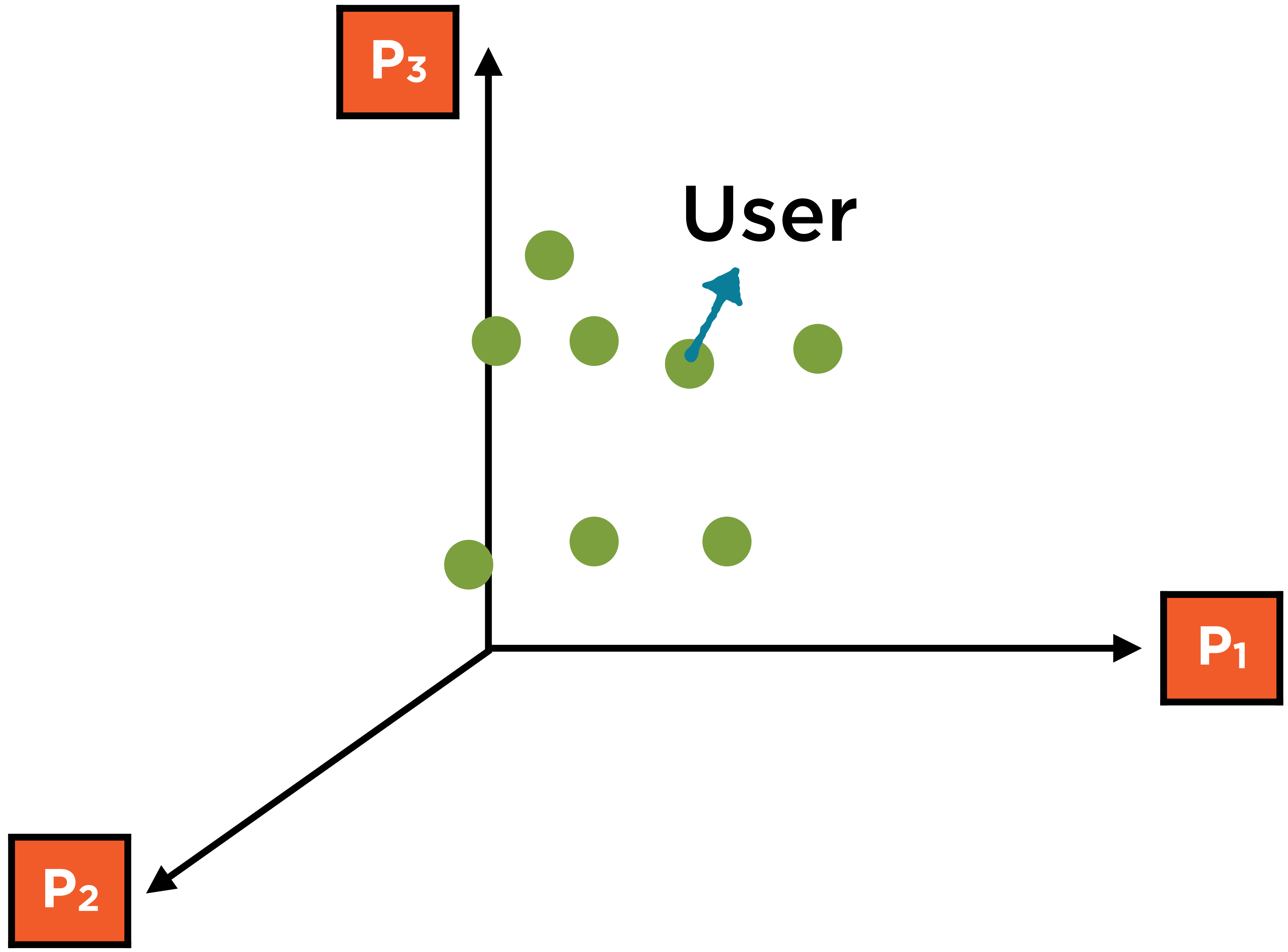
Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

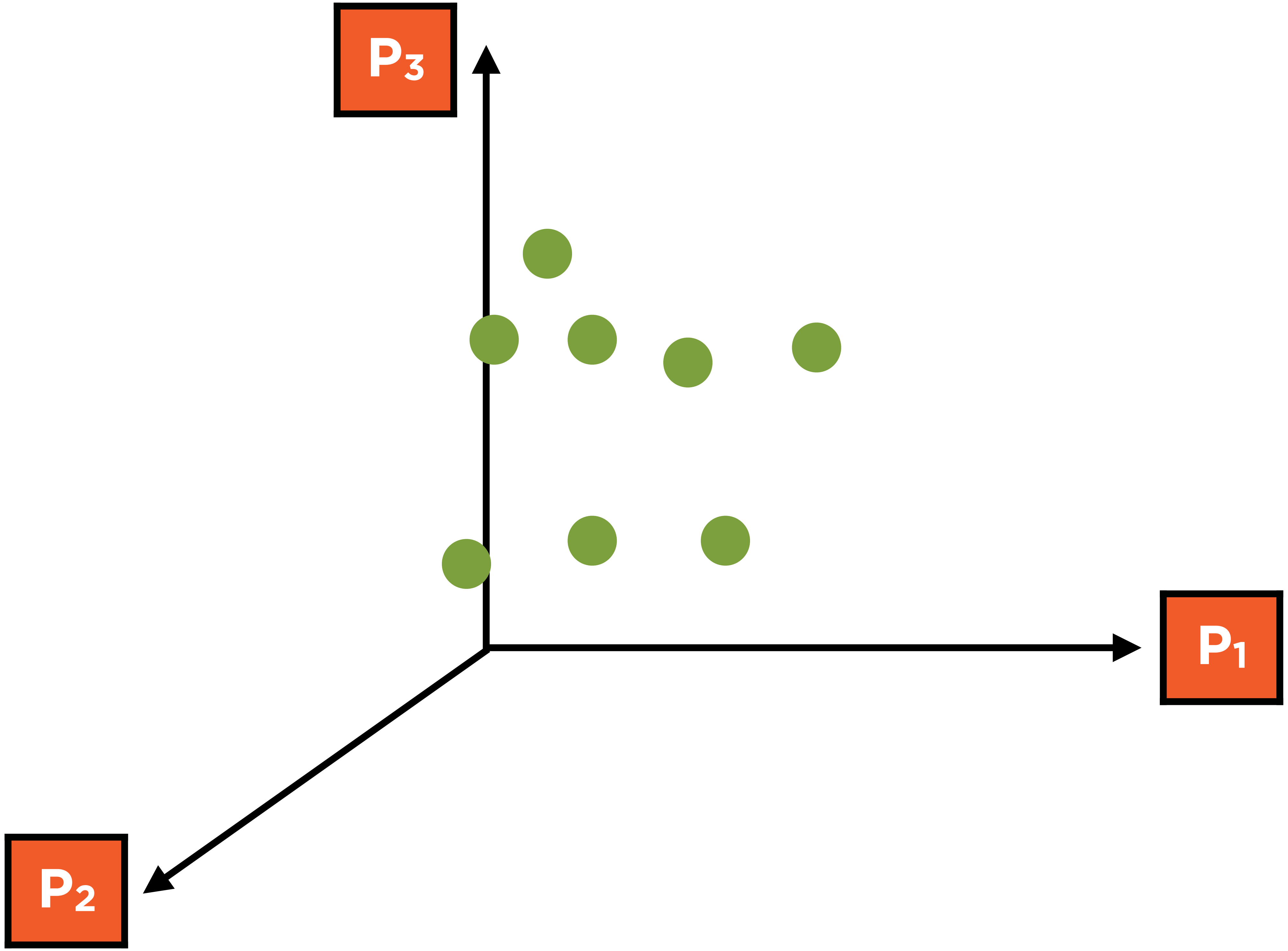
Model Updates

Memory Usage

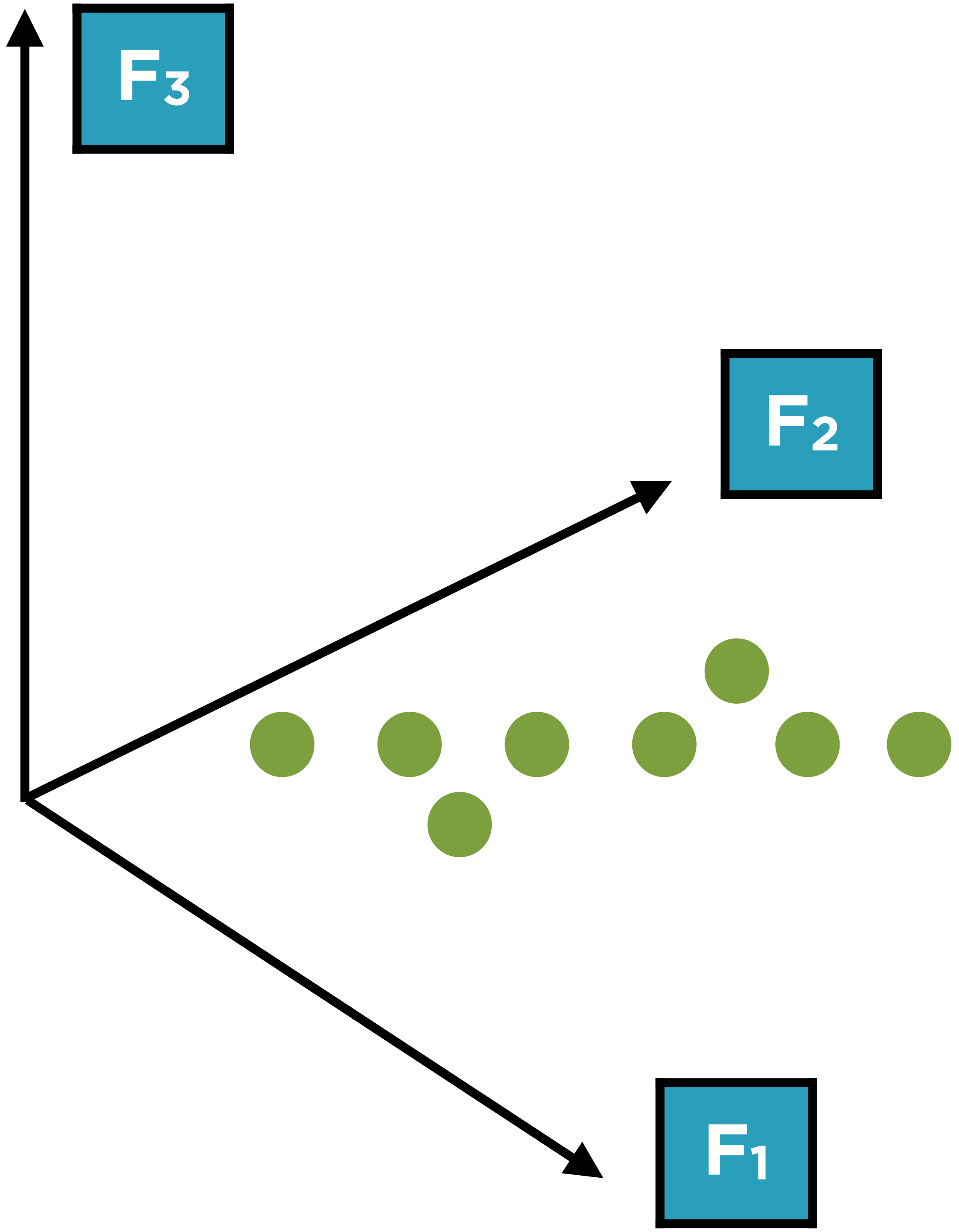
Nearest Neighbors Model



Latent Factor Analysis



Latent Factor Analysis



Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Observable attributes vs hidden driving factors

Model Updates

Memory Usage

Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Observable attributes vs hidden driving factors

Model Updates

Memory Usage

Nearest Neighbors Model

Active user →

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

→ Recommendations

Nearest Neighbors Model

Online updates

Active user → Recommendations

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	7	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

Latent Factor Analysis

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Latent Factor Analysis

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Recommendations

Active user

Latent Factor Analysis

Offline updates

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	7	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

	P_1	P_2	P_3	P_4	P_5
F_1					
F_2					
F_3					

Recommendations

Active user

Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Observable attributes vs hidden driving factors

Model Updates

Online vs offline updates

Memory Usage

Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Observable attributes vs hidden driving factors

Model Updates

Online vs offline updates

Memory Usage

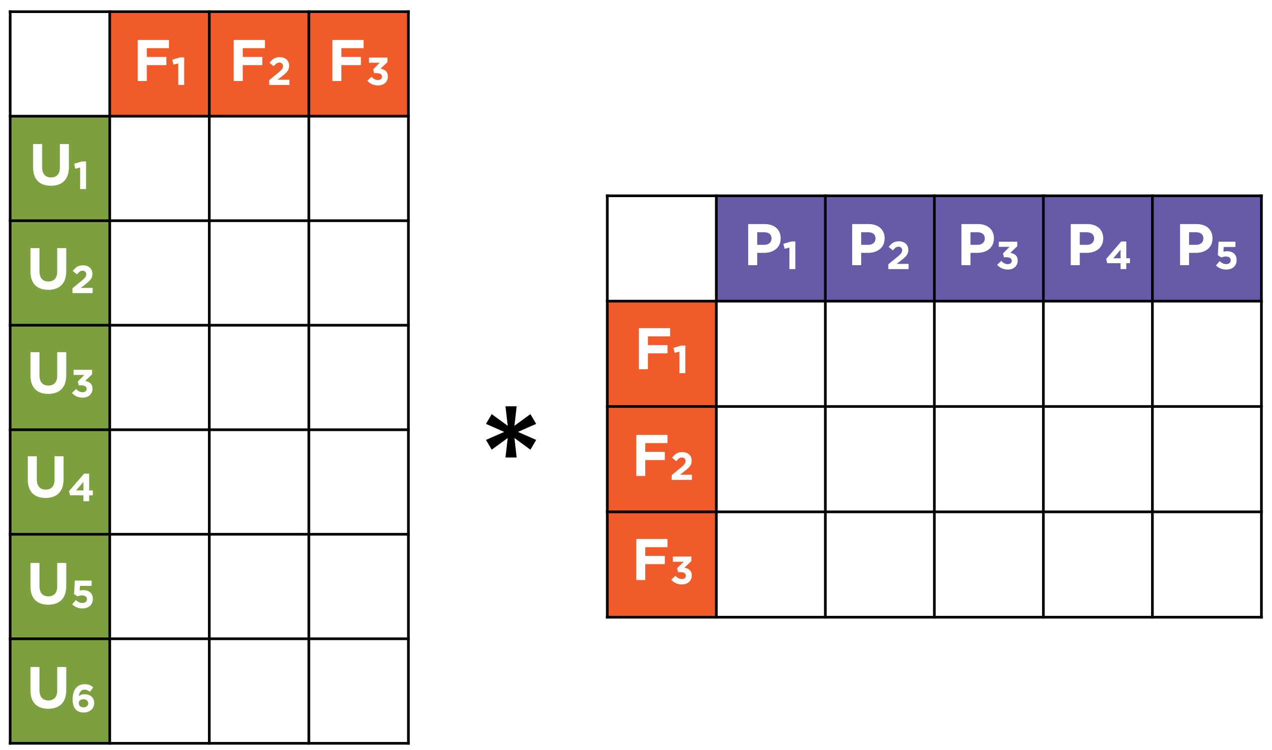
Nearest Neighbors Model

	P_1	P_2	P_3	P_4	P_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

Ratings kept in-memory at all times

User recommendations computed on-demand

Latent Factor Analysis



Rating matrix is decomposed offline

All recommendations can easily be pre-computed as a one off

Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Observable attributes vs hidden driving factors

Model Updates

Online vs offline updates

Memory Usage

In-memory vs pre-computed

Nearest Neighbors Model vs Latent Factor Analysis

Data Representation

Observable attributes vs hidden driving factors

Model Updates

Online vs offline updates

Memory Usage

In-memory vs pre-computed

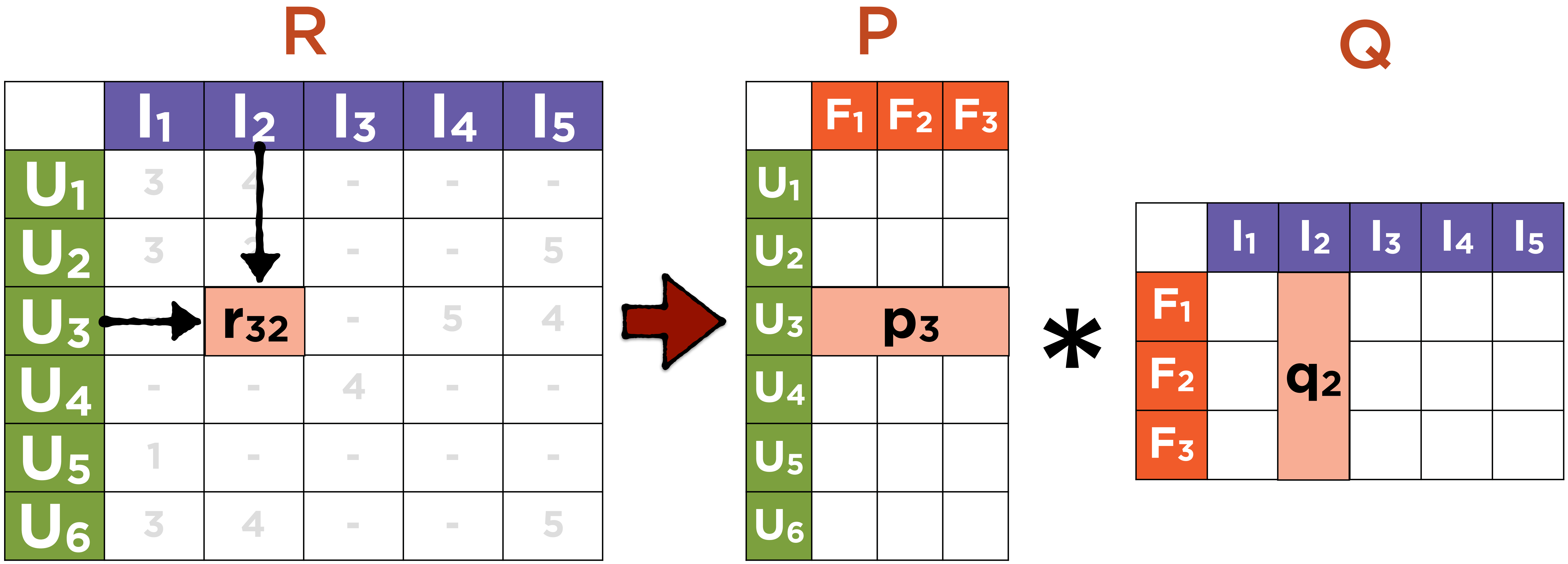
Decomposing the Rating Matrix

Using is Easy, Building is Hard

Driving a car

Building an engine

Solving for Factors



Solving for Factors

R

	I_1	I_2	I_3	I_4	I_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

P

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

$*$

Q

	I_1	I_2	I_3	I_4	I_5
F_1					
F_2					
F_3					

$$r_{ui} = p_u \cdot q_i$$

Solve the set of equations

$$r_{ui} = p_u \cdot q_i$$

Equations = # Known ratings

Solving for Factors

R

	I_1	I_2	I_3	I_4	I_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

P

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

Q

	I_1	I_2	I_3	I_4	I_5
F_1					
F_2					
F_3					

$$r_{ui} = p_u \cdot q_i$$

Solving for Factors

R

	I_1	I_2	I_3	I_4	I_5
U_1	3	4	-	-	-
U_2	3	2	-	-	5
U_3	-	2	-	5	4
U_4	-	-	4	-	-
U_5	1	-	-	-	-
U_6	3	4	-	-	5

P

	F_1	F_2	F_3
U_1			
U_2			
U_3			
U_4			
U_5			
U_6			

*

Q

	I_1	I_2	I_3	I_4	I_5
F_1					
F_2					
F_3					

$$r_{ui} - p_u \cdot q_i = \varepsilon_{ui}$$

Find p_u and q_i such that

$$\Sigma(r_{ui} - p_u \cdot q_i)^2$$

Total error is minimized

$$\min \Sigma(r_{ui} - p_u \cdot q_i)^2$$

Add a term to penalize the model for the number of factors

$$\min \sum (r_{ui} - p_u . q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2)$$

Add a term to penalize the model for the number of factors

$$\min \sum (r_{ui} - p_u . q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2)$$

Regularization term

$$\min \sum (r_{ui} - p_u . q_i)^2 + \lambda(||p_u||^2 + ||q_i||^2)$$

Regularization factor

$$\min E(p_u, q_i)$$

A standard optimization problem

Demo

Solve for latent factors using the Stochastic Gradient Descent method (SGD)

- Set up a function to compute error
- Set up a function to implement SGD

Solving for Latent Factors

Set up the data

Functions to access relevant information

Construct a rating matrix

The representation needed for collaborative filtering

Initialize factor matrices

Compute the error

Set up a function to measure RMSE

Update the factor matrices

Minimize the error using SGD

Solving for Latent Factors

Set up the data

Functions to access relevant information

Construct a rating matrix

The representation needed for collaborative filtering

Setting Up the Rating Matrix

User	ISBN	Rating

	I_1	I_2	I_3	I_4	I_5
U_1	3	4	-	-	4
U_2	3	5	3	4	5
U_3	4	2	-	5	4
U_4	3	-	4	5	2
U_5	1	-	4	2	1
U_6	3	4	-	2	5

pandas.pivot_table

Setting Up the Rating Matrix

User	ISBN	Rating

	I_1	I_2	I_3	I_4	I_5
U_1	3	4	-	-	4
U_2	3	5	3	4	5
U_3	4	2	-	5	4
U_4	3	-	4	5	2
U_5	1	-	4	2	1
U_6	3	4	-	2	5

scipy.coo_matrix

```
coo_matrix((values,(rowsource, columnsource)))
```

Creating a Rating Matrix

Takes a tuple with rating data, row names and column names

User	ISBN	Rating

```
coo_matrix((values,(rowsource, columnsource)))
```

Creating a Rating Matrix

Takes a tuple with rating data, row names and column names

User	ISBN	Rating

```
coo_matrix((values,(rowsource, columnsource)))
```

Creating a Rating Matrix

Takes a tuple with rating data, row names and column names

User	ISBN	Rating

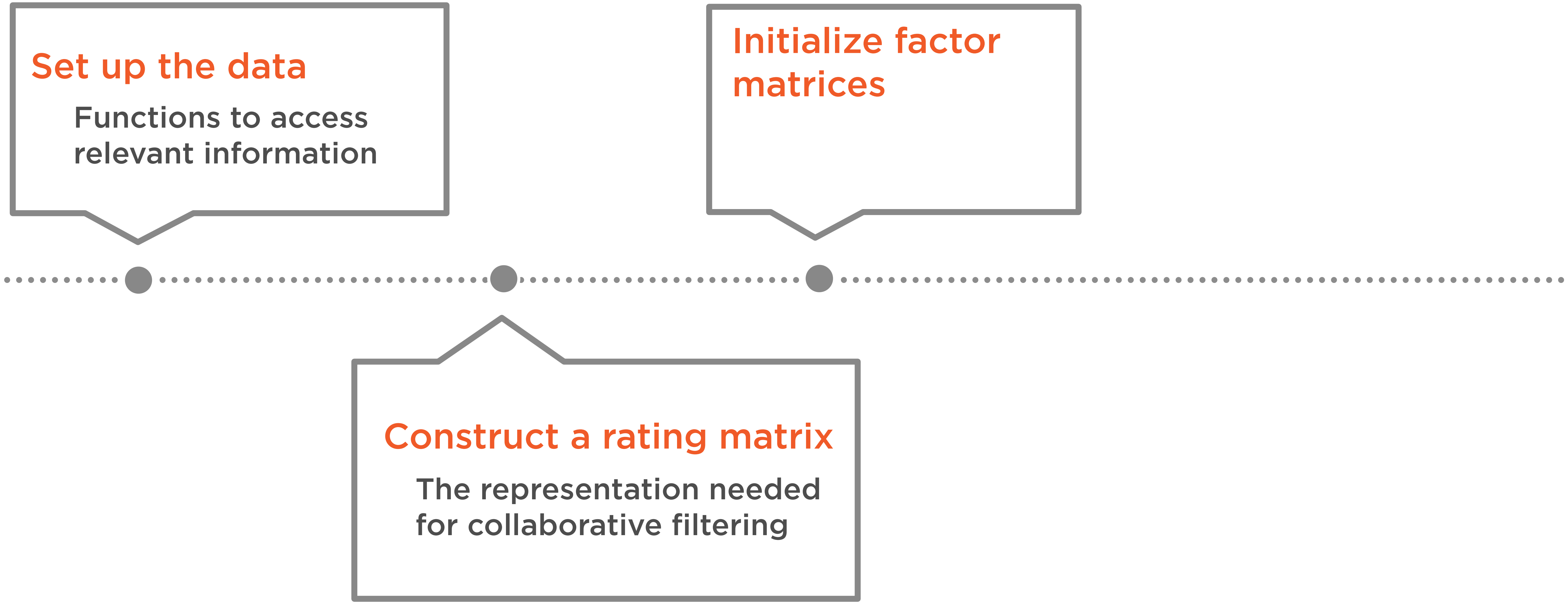
```
coo_matrix((values,(rowsource, columnsource)))
```

Creating a Rating Matrix

Takes a tuple with rating data, row names and column names

	..	3053	..	..
..	4	-	-	-
..	2	-	-	5
10633	2	0	5	4
..	-	4	-	-
..	-	-	-	-
..	4	-	-	5

Solving for Latent Factors



Solving for Latent Factors

Set up the data

Functions to access relevant information

Construct a rating matrix

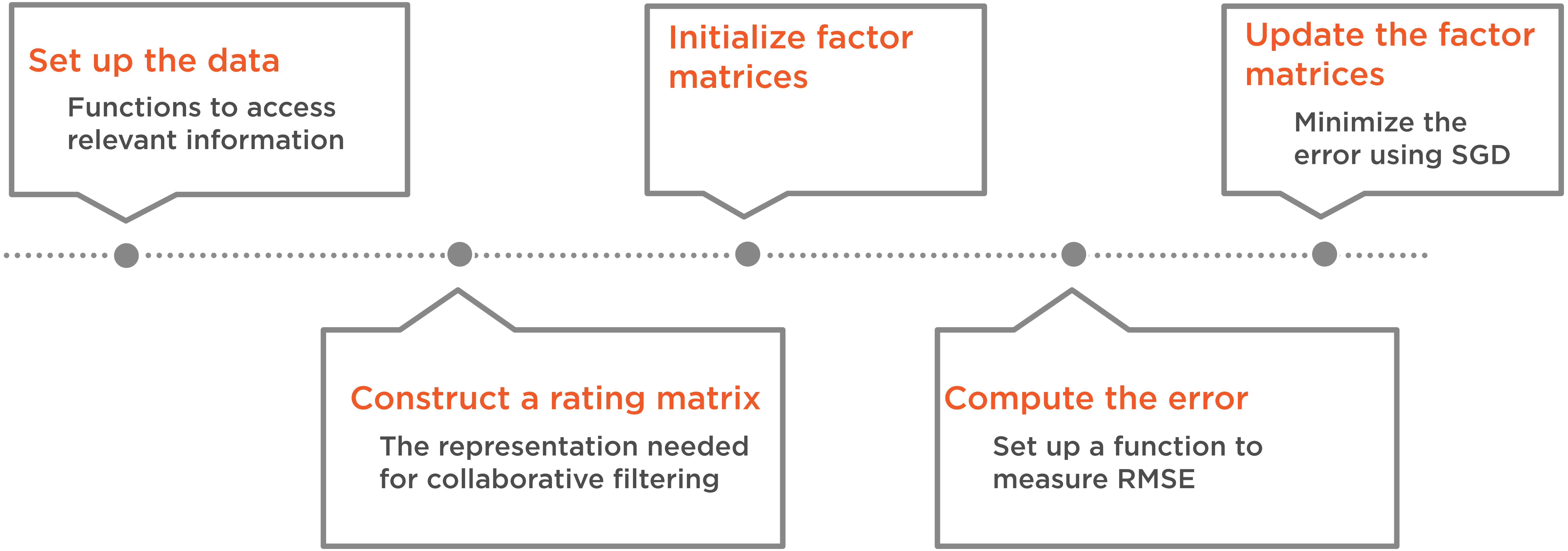
The representation needed for collaborative filtering

Initialize factor matrices

Compute the error

Set up a function to measure RMSE

Solving for Latent Factors



Stochastic Gradient Descent

$$\sum(r_{ui}-p_u.q_i)^2+\lambda(||p_u||^2+||q_i||^2)$$

Total error across all ratings

Stochastic Gradient Descent

$$\sum(r_{ui}-p_u.q_i)^2+\lambda(||p_u||^2+||q_i||^2)$$

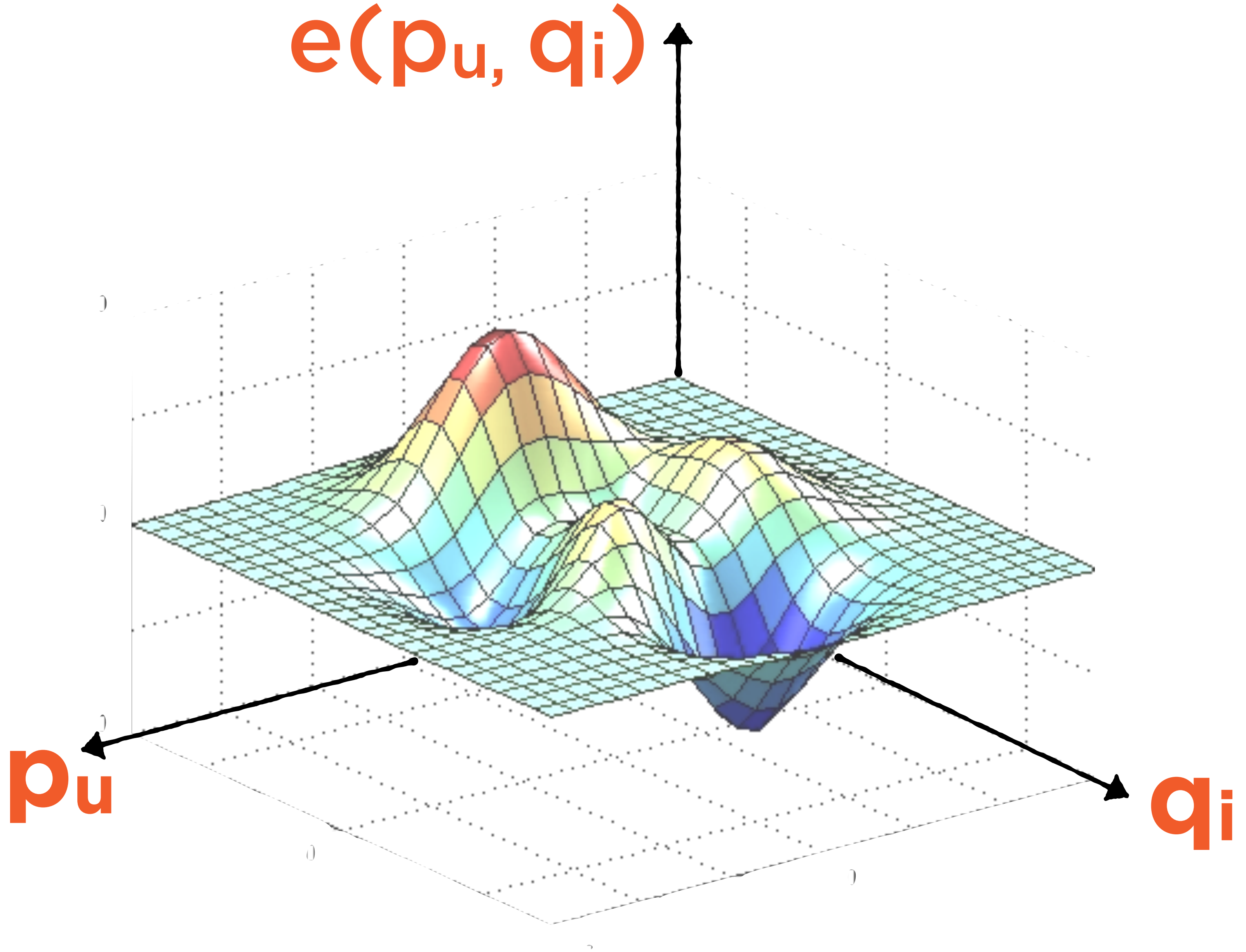
Look at the error for 1 rating

Stochastic Gradient Descent

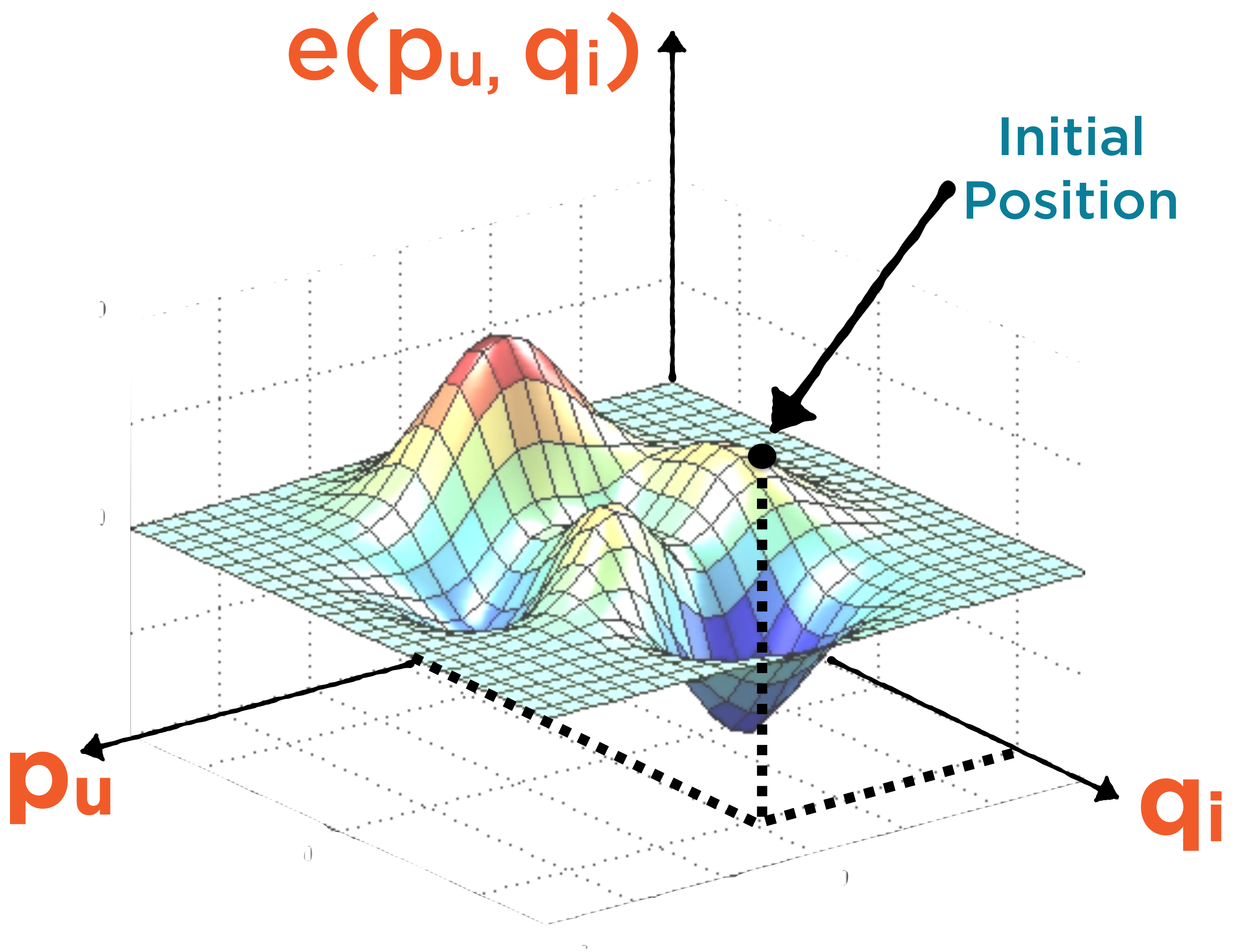
$$e(p_u, q_i)=(r_{ui}-p_u.q_i)^2+\lambda(||p_u||^2+||q_i||^2)$$

Look at the error for 1 rating

Stochastic Gradient Descent

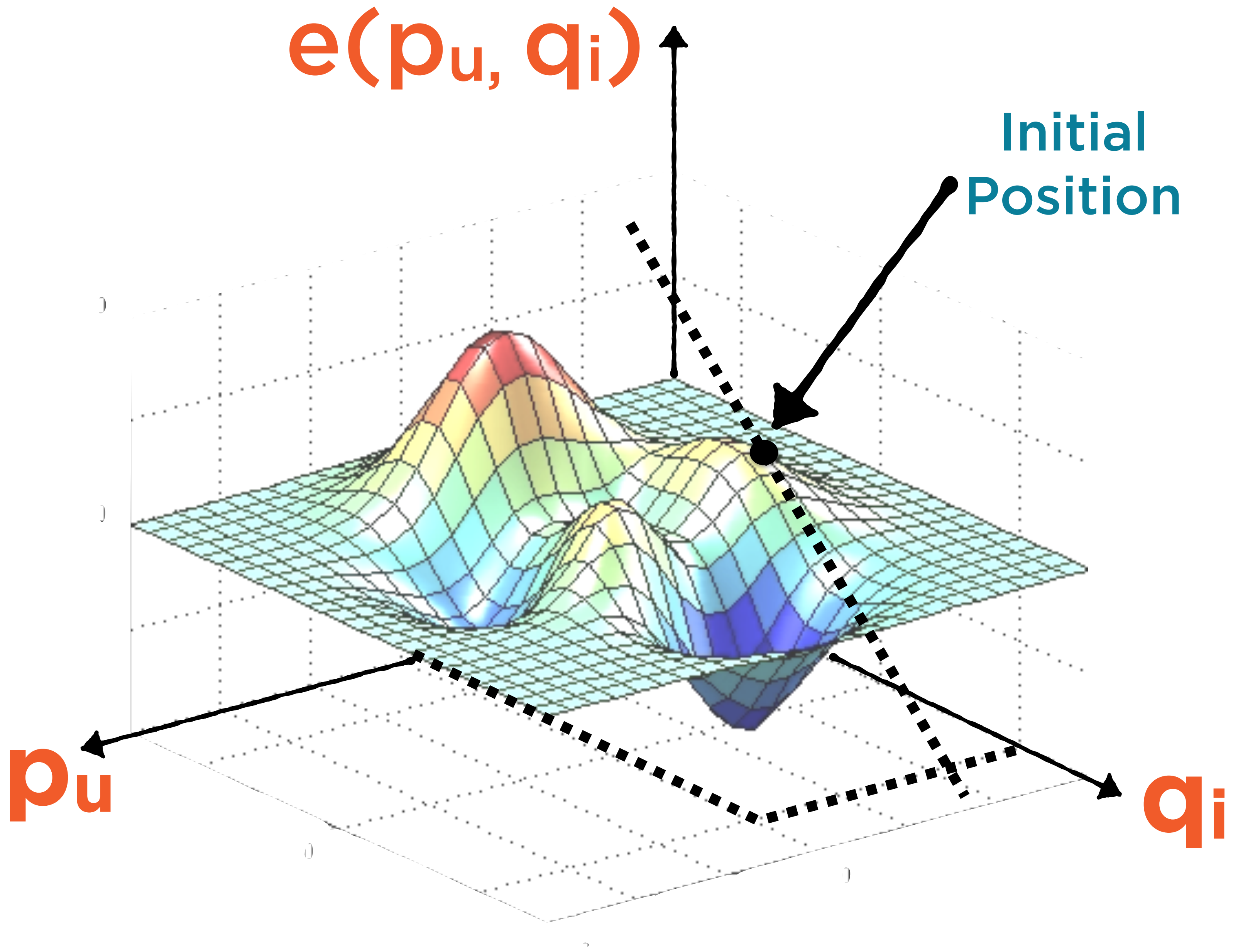


Stochastic Gradient Descent



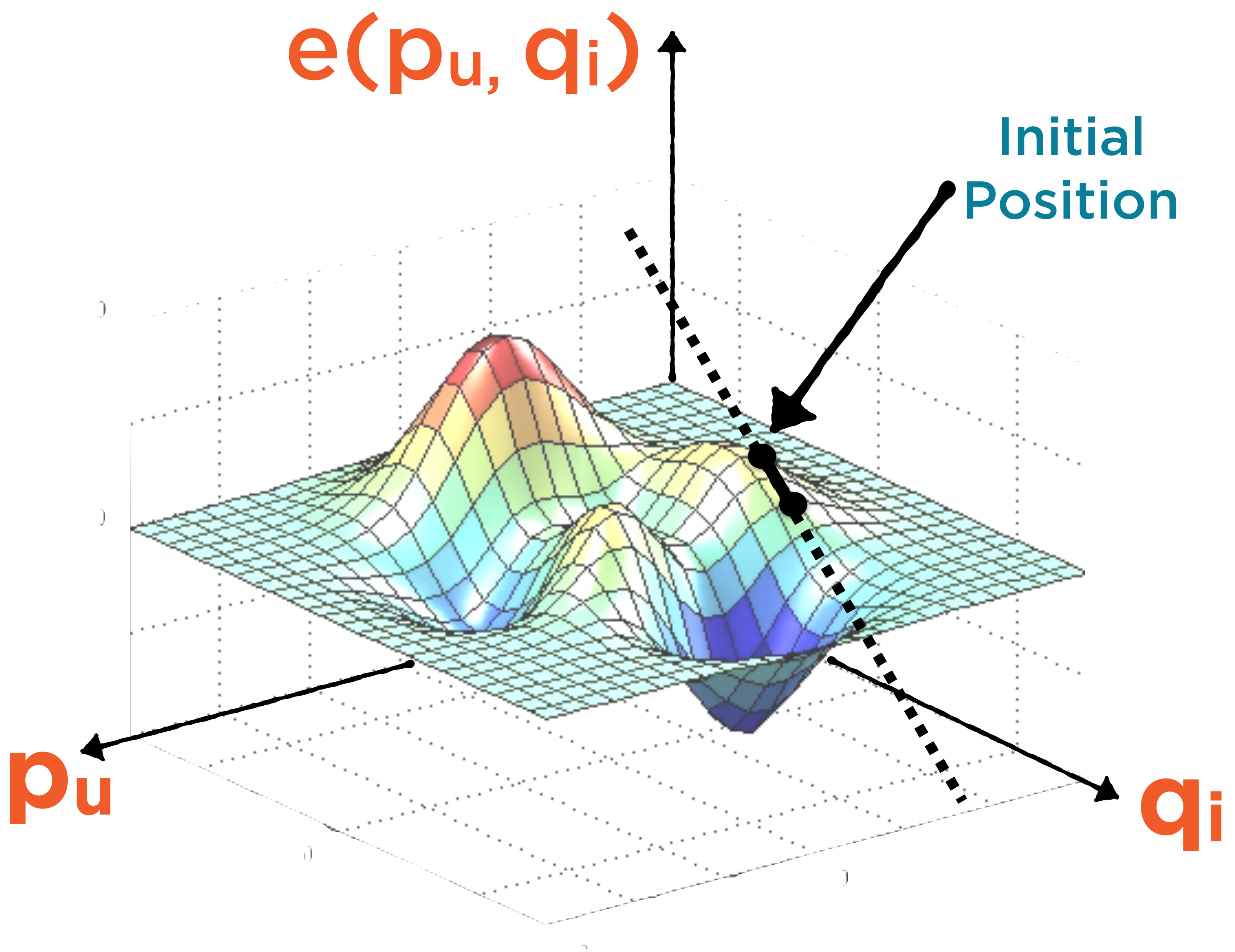
Compute the initial value of the error

Stochastic Gradient Descent



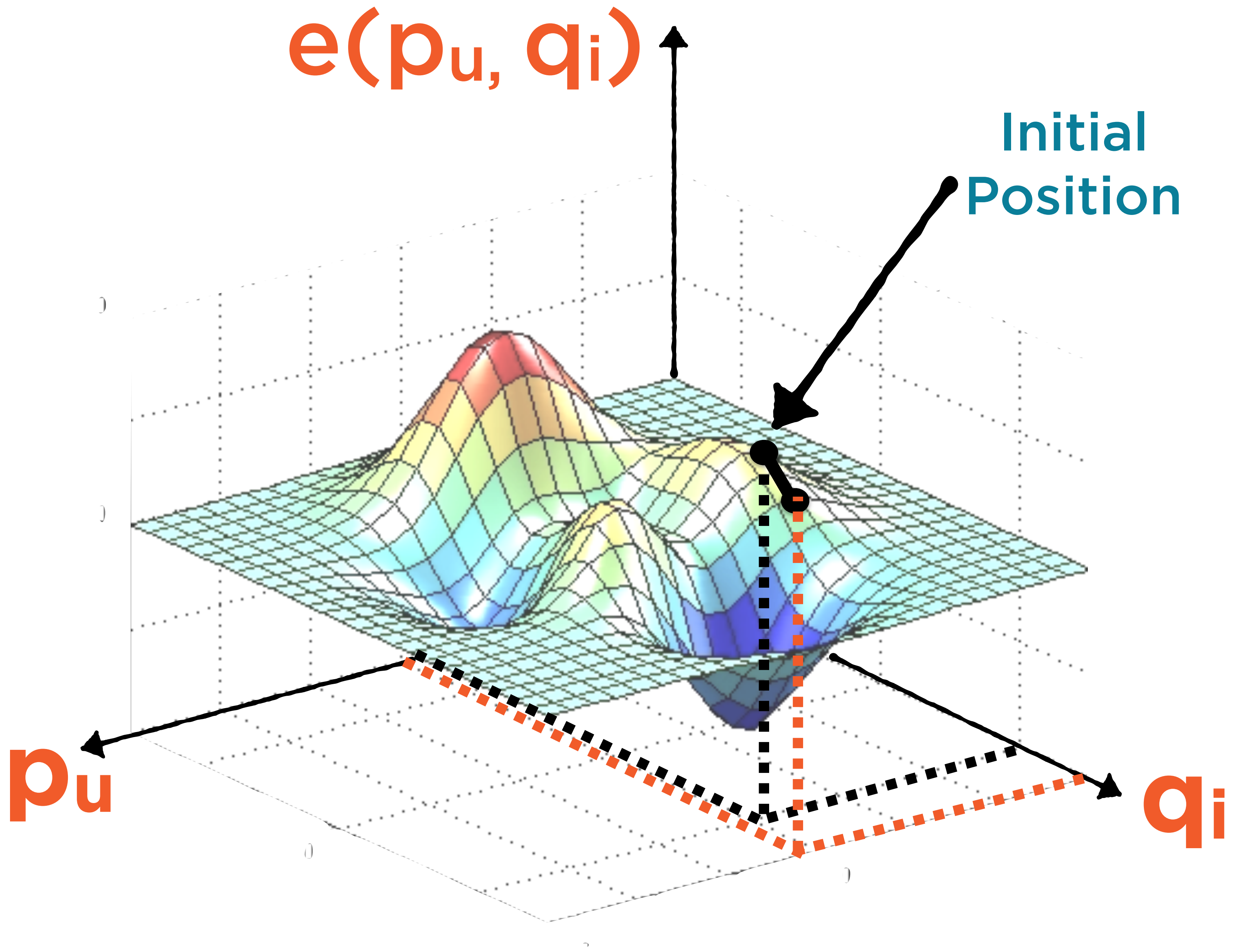
Compute the slope at that point

Stochastic Gradient Descent



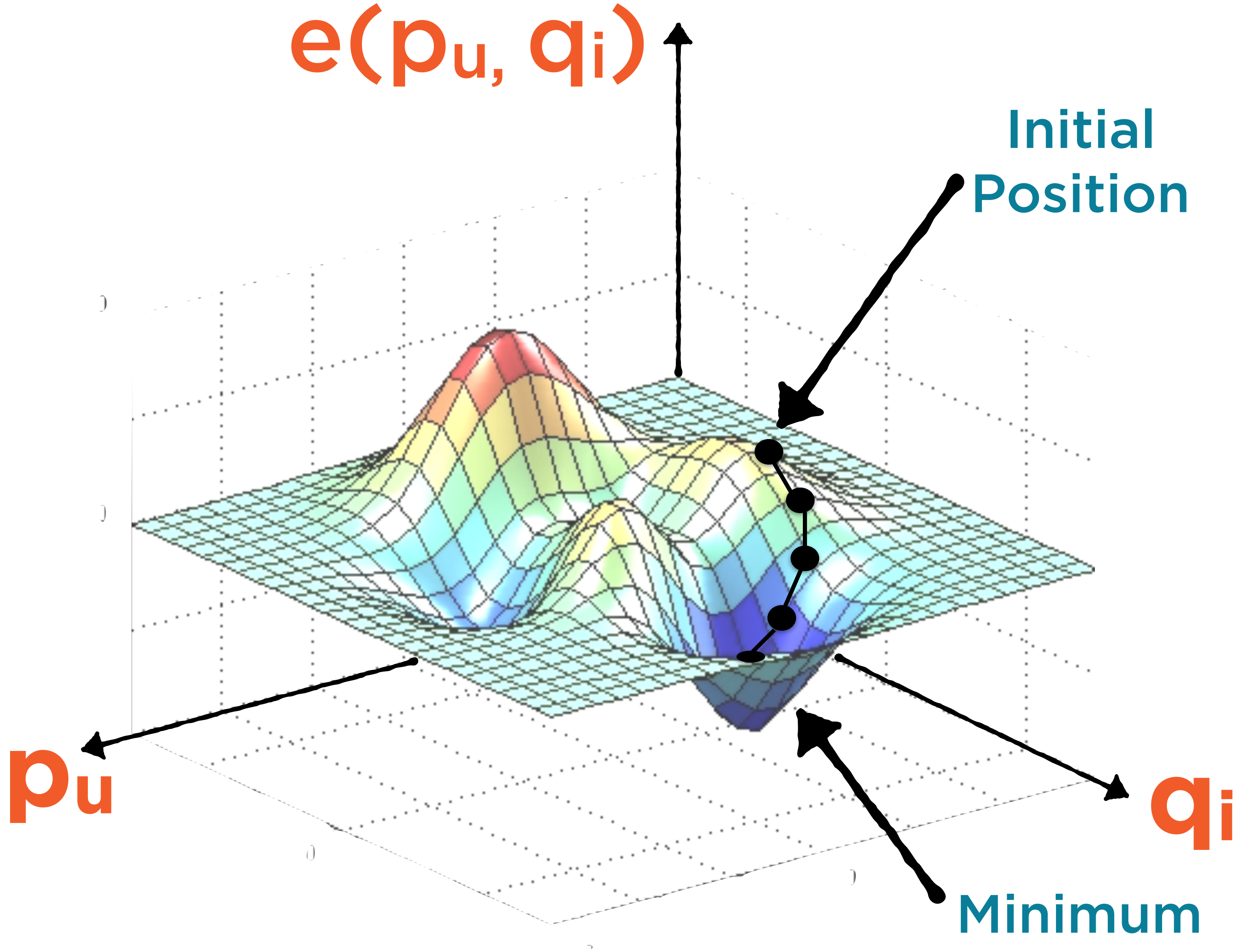
Take a small step in the downward direction of the slope

Stochastic Gradient Descent



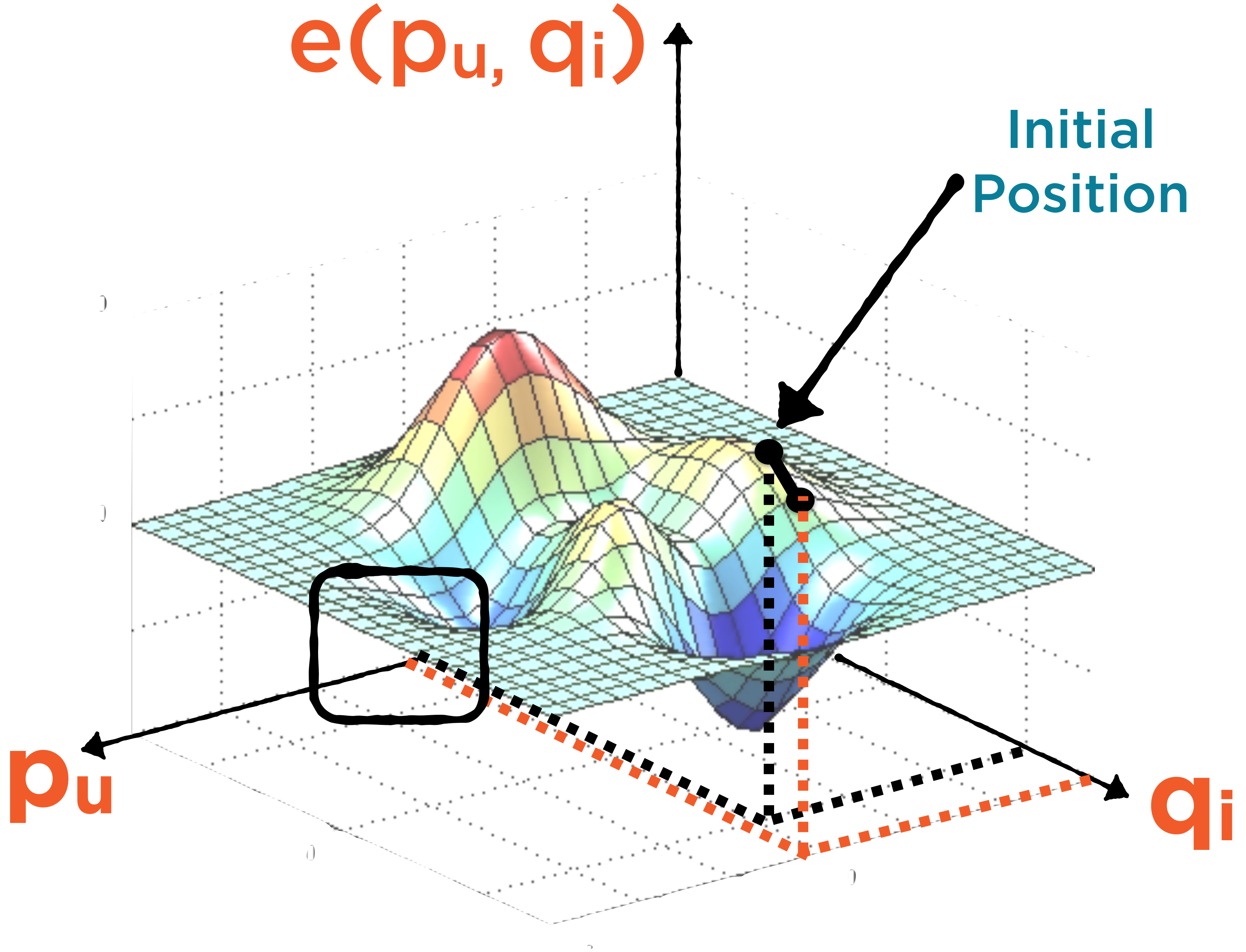
Take a small step i.e. update p_u and q_i by a small value

Stochastic Gradient Descent



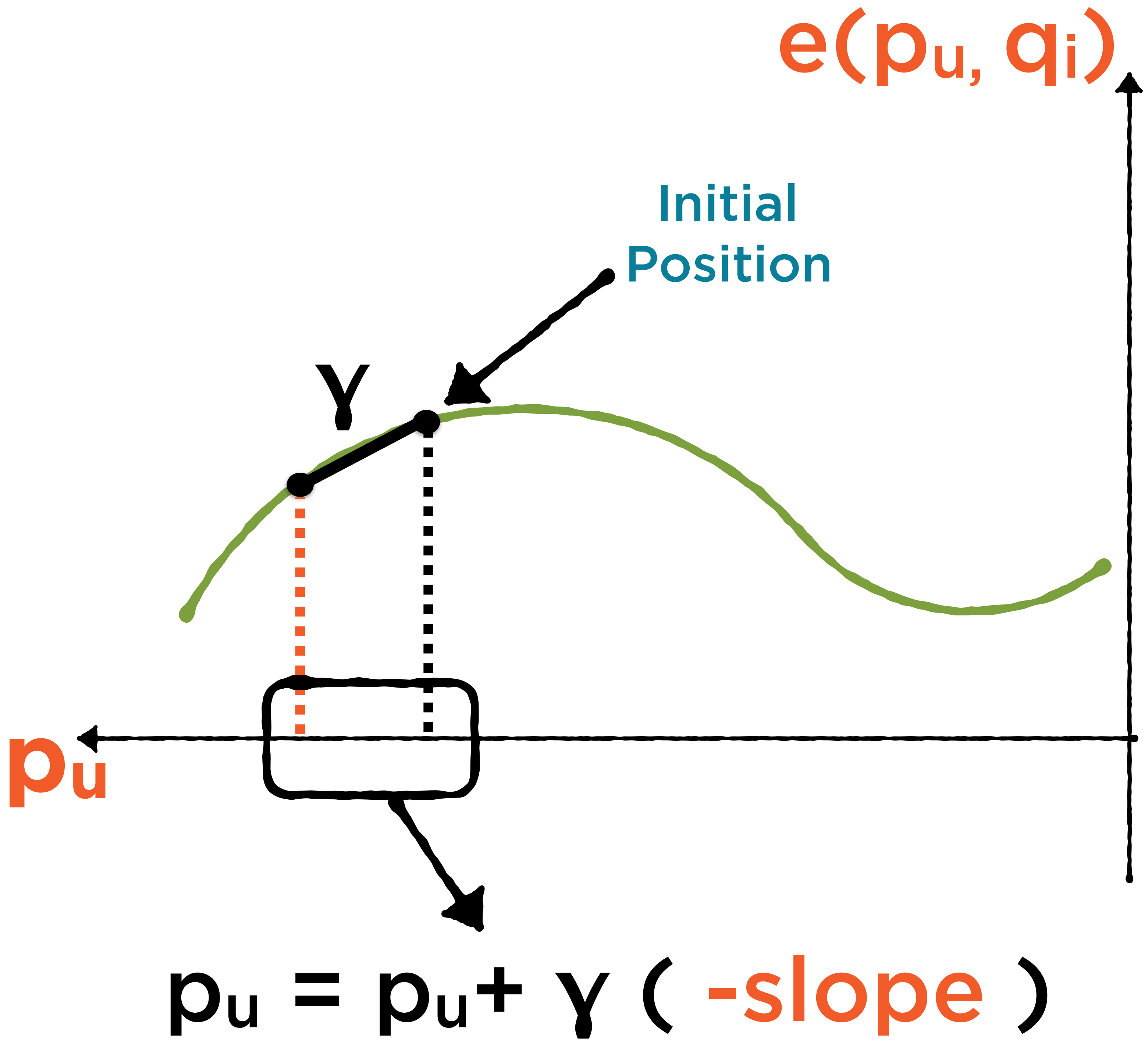
Repeat until you reach a minimum

Stochastic Gradient Descent

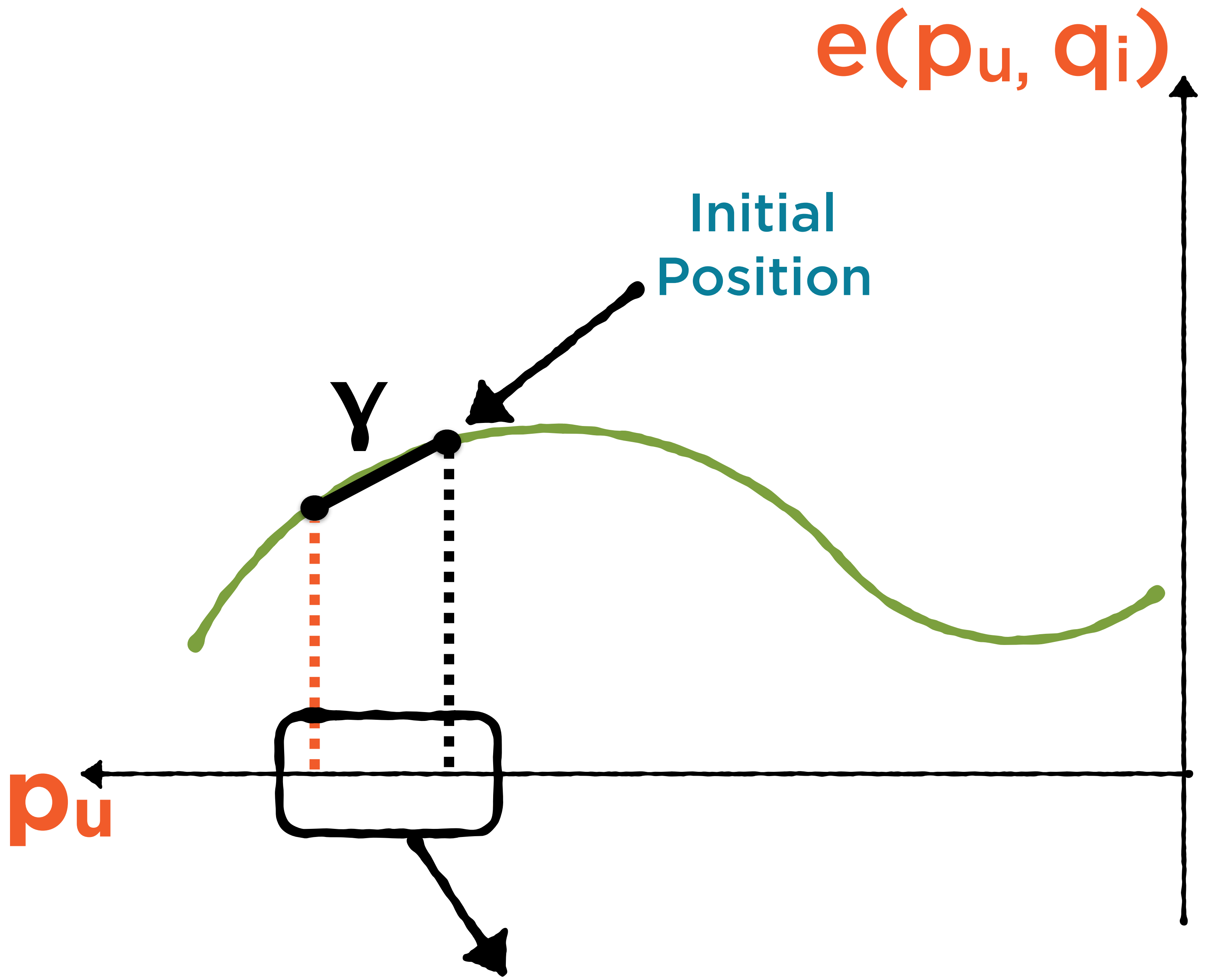


Take a small step i.e. update p_u and q_i by a small value

Stochastic Gradient Descent

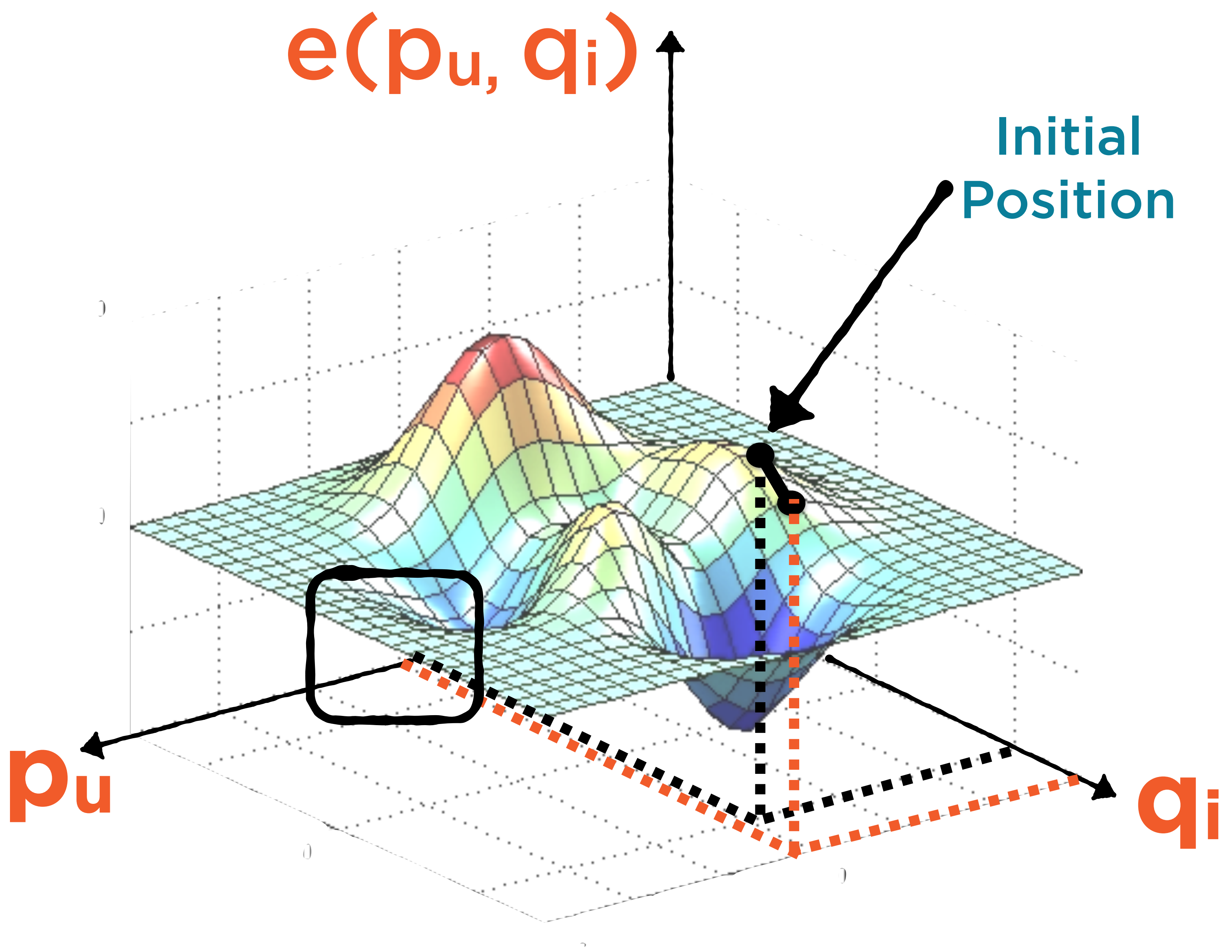


Stochastic Gradient Descent

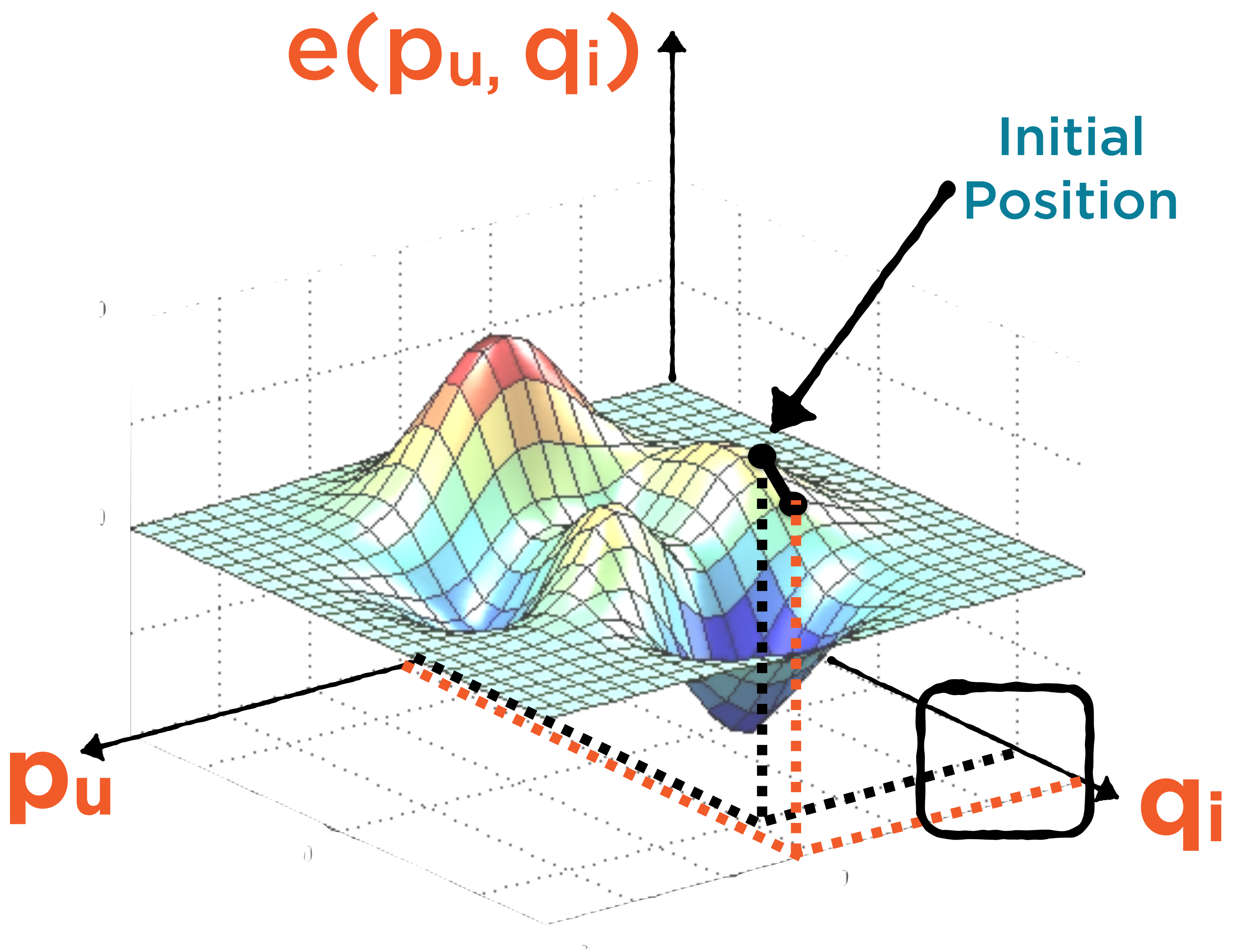


$$p_u = p_u + \gamma \left(- \frac{\partial E(p_u, q_i)}{\partial p_u} \right)$$

Stochastic Gradient Descent



Stochastic Gradient Descent



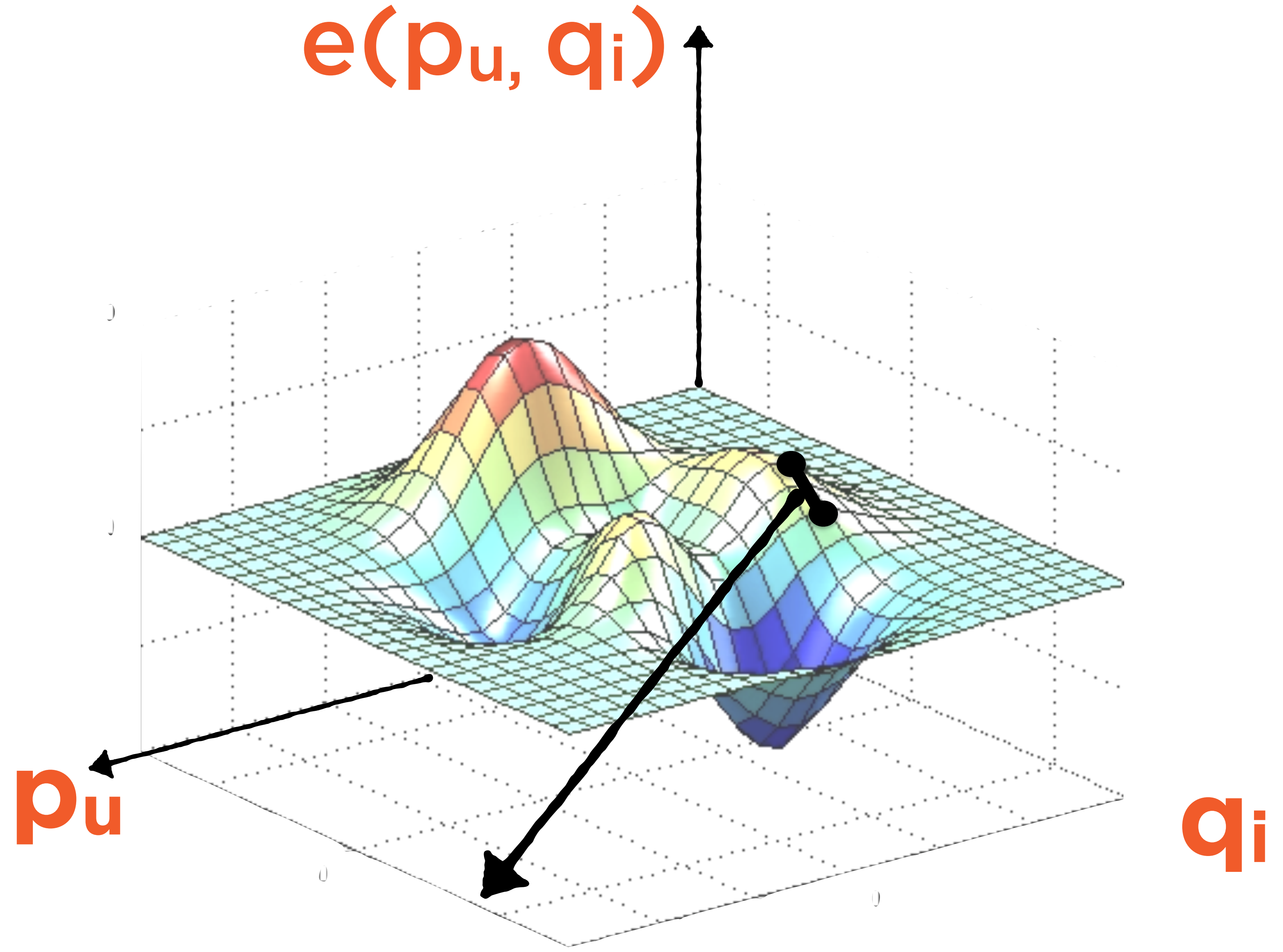
$$q_i = q_i + \gamma \left(- \frac{\partial E(p_u, q_i)}{\partial q_i} \right)$$

Stochastic Gradient Descent

While max # allowed steps is not reached :

- For each rating :
  - Update p_u and q_i by a small value
- If RMSE <threshold
  - Break

Stochastic Gradient Descent



$$p_u = p_u + \gamma \left(- \frac{\partial E(p_u, q_i)}{\partial p_u} \right)$$

$$q_i = q_i + \gamma \left(- \frac{\partial E(p_u, q_i)}{\partial q_i} \right)$$

```
eui=rui-np.dot(P[u,:],Q[:,i])
P[u,:]=P[u,:]+gamma*2*(eui*Q[:,i]-lamda*P[u,:])
Q[:,i]=Q[:,i]+gamma*2*(eui*P[u,:]-lamda*Q[:,i])
```

Updating p_u and q_i

$\varepsilon_{ui} = r_{ui} - p_u . q_i$

```
eui=rui-np.dot(P[u,:],Q[:,i])
P[u,:]=P[u,:]+gamma*2*(eui*Q[:,i]-lamda*P[u,:])
Q[:,i]=Q[:,i]+gamma*2*(eui*P[u,:]-lamda*Q[:,i])
```

Updating p_u and q_i

$$p_u = p_u + \gamma \left(-\frac{\partial E(p_u, q_i)}{\partial p_u} \right)$$

```
eui=rui-np.dot(P[u,:],Q[:,i])
P[u,:]=P[u,:]+gamma*2*(eui*Q[:,i]-lamda*P[u,:])
Q[:,i]=Q[:,i]+gamma*2*(eui*P[u,:]-lamda*Q[:,i])
```

Updating p_u and q_i

$$p_u = p_u + \gamma\ (\ 2(\varepsilon_{ui} q_i - \lambda p_u)\ )$$

```
eui=rui-np.dot(P[u,:],Q[:,i])
P[u,:]=P[u,:]+gamma*2*(eui*Q[:,i]-lamda*P[u,:])
Q[:,i]=Q[:,i]+gamma*2*(eui*P[u,:]-lamda*Q[:,i])
```

Updating p_u and q_i

$$q_i = q_i + \gamma \left(- \frac{\partial E(p_u, q_i)}{\partial q_i} \right)$$

```
eui=rui-np.dot(P[u,:],Q[:,i])
P[u,:]=P[u,:]+gamma*2*(eui*Q[:,i]-lamda*P[u,:])
Q[:,i]=Q[:,i]+gamma*2*(eui*P[u,:]-lamda*Q[:,i])
```

Updating p_u and q_i

$$q_i = q_i + \gamma\,(\,2(\varepsilon_{ui}p_u - \lambda q_i)\,)$$

```
eui=rui-np.dot(P[u,:],Q[:,i])
P[u,:]=P[u,:]+gamma*2*(eui*Q[:,i]-lamda*P[u,:])
Q[:,i]=Q[:,i]+gamma*2*(eui*P[u,:]-lamda*Q[:,i])
```

Updating p_u and q_i

Summary

Understand the latent factors model for collaborative filtering

Contrast the latent factors model and the nearest neighbors model

Use optimization techniques to solve for latent factors

- Stochastic gradient descent