Nightout

Matt Zoufaly, Stephen Beard, Deep Ghosh, Joey Dodds

The Problem

- “What do I do when I’m bored?”
- We want to recommend things to do from previous user ratings
- We will provide useful recommendations for a user with only a few ratings
- Tell users why they will like a business
  - “We recommend Hoagie Haven because you liked George’s”

NMF

- We will give ratings by Nonnegative Matrix Factorization (NMF)
- NMF discovers latent factors that affect ratings
- It relates both users and businesses to these factors
- If user *x* likes “burgers” and restaurant *y* is a “burgers” restaurant user *x* will like restaurant *y*

Businesses

Users

	Pizza Palace	Sushi Store	Wrap Hut	Beer Bar	…
Bob	--	--	5	--	…
Jane	2	5	--	3	…
⋮	⋮	⋮	⋮	⋮	

+ K (number of latent factors)

Input

Output

Latent Factors

Users

	F1 (italian)	F2 (japanese)	…
Bob	.8	.2	…
Jane	.1	.5	…
⋮	⋮	⋮	

Businesses

Latent Factors

	Piz.	Sush.	Wrap	Beer	…
F1	.9	.2	.3	.5	…
F2	.1	.9	.2	.4	…
⋮	⋮	⋮	⋮	⋮	

	F1 (italian)	F2 (japanese)	
Bob	.8	.2	…
Jane	.1	.5	…
⋮	⋮	⋮	

	Piz.	Sush.	Wrap	Beer	…
F1	.9	.2	.3	.5	…
F2	.1	.9	.2	.4	…
⋮	⋮	⋮	⋮	⋮	

Dot Product!

Predicted Ratings

Businesses

	Pizza Palace		Sushi Store		Wrap Hut		Beer Bar		…
	Actual	Predicted	Act.	Pred.	Act.	Pred.	Act.	Pred.	…
Bob	--	5	--	4	5	4	--	2	…
Jane	2	1	5	4	--	5	3	2	
⋮	⋮		⋮		⋮		⋮		

Algorithm

- We use an iterative EM algorithm to factorize
- Inputs are ratings, K, α, and β
- K is the number of latent factors we wish to find
- α scales the amount we change relationships each iteration
- β scales the regularization factor to avoid over-fitting

The Algorithm

We calculate the new Latent Factor Matrices P and Q by

$$p'_{ik} = p_{ik} + \alpha\left(2e_{ij}q_{kj} \ - \beta p_{ik}\right)$$
$$q'_{ik} = q_{kj} + \alpha(2e_{ij}p_{ik} \ - \beta q_{kj})$$

Where e is predicted by P and Q

Then we get predicted ratings:

$$R \ \approx P \times Q^T = \hat{R}$$

Dataset

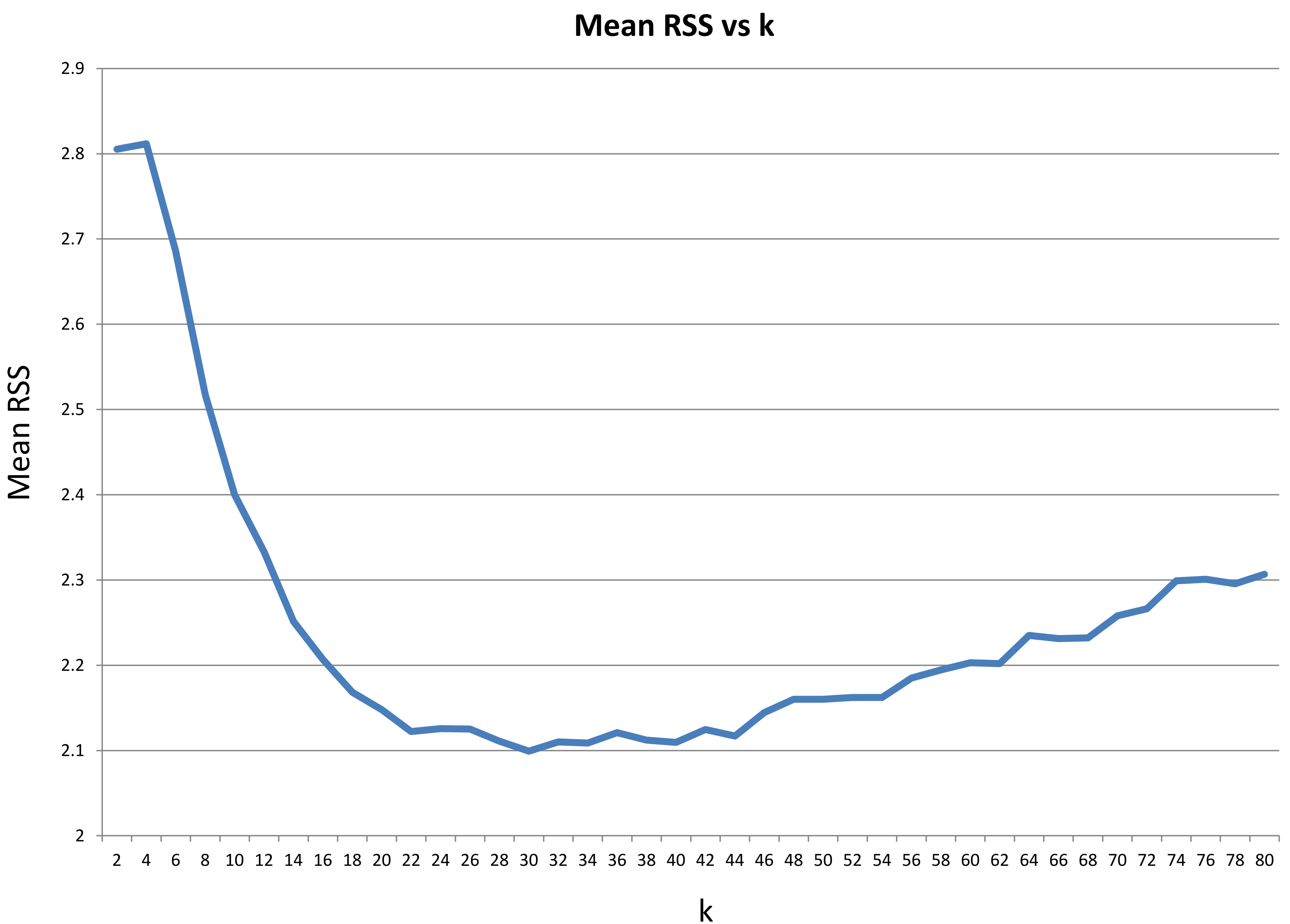
- Yelp academic datasets
- 30k Users with 25 ratings or more each
- 1400 Businesses with 25 ratings or more each
- Many other users and business have fewer ratings
- We are using 2 datasets
  - Michigan dataset: For testing and building our model
  - Princeton dataset: Unused for now, we will test the quality of our model on this when it is complete

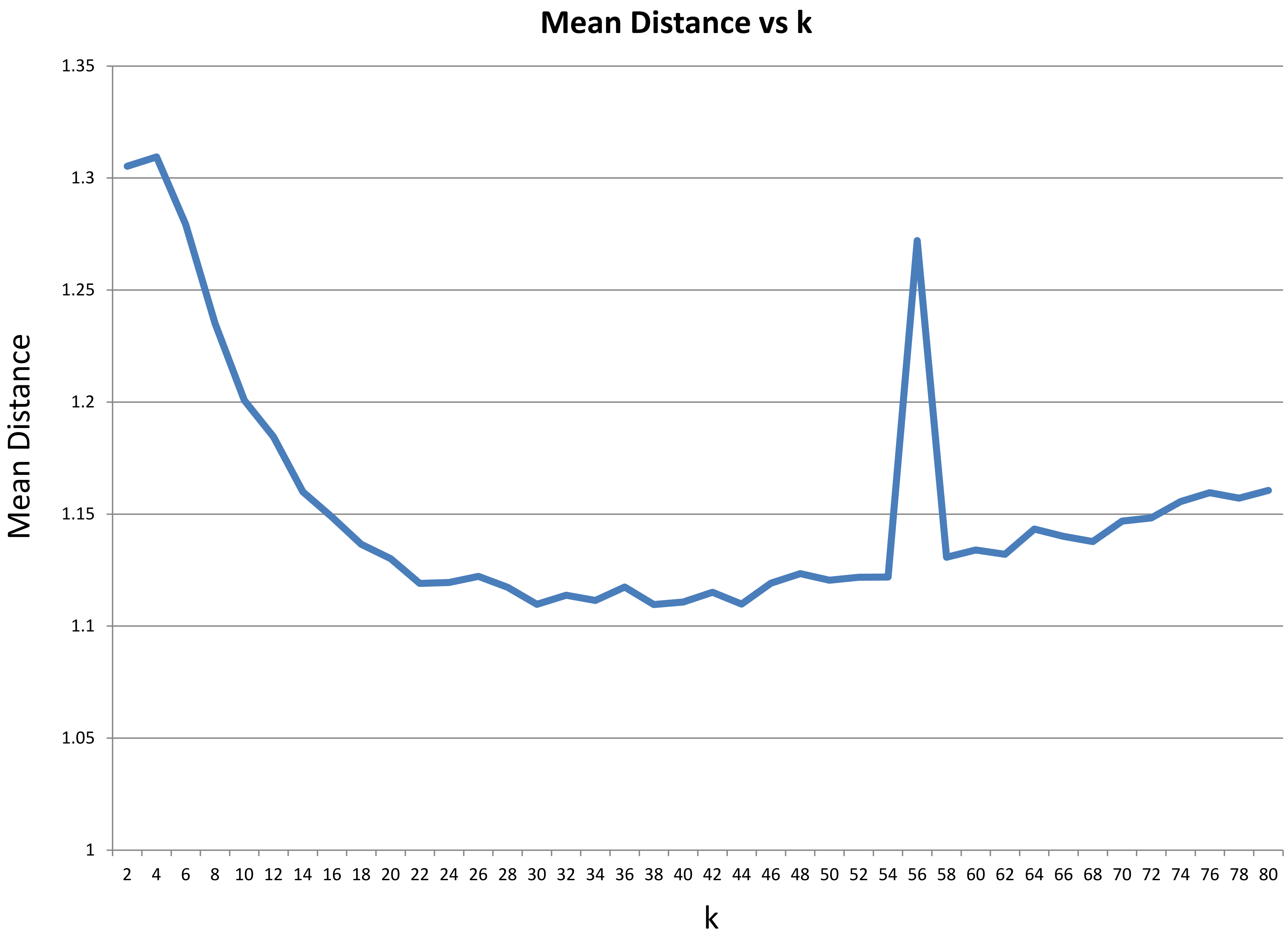
Challenges

- The NMF algorithm takes a while to run on large datasets
- Switched to a sparse matrix representation
- Ported python code to C and parallelized
- Played with the α input to approach convergence quickly without overshooting
- Dealing with inconsistencies in user voting (future work)

Distance Vs. K

- Graphs “Mean RSS Vs. K” and “Distance Vs. K” show results of cross-validation on the Michigan data
- Around 30 seems like the best value for K
- We hope to find a relationship so we can discover an approximate good K from the data
- We would expect a good value of K to be related to the number of “Tags” that are often applied to businesses
  - We might want to modify tags to include things like pricing and quality of service


Mean RSS vs k
Mean RSS
k
2.9
2.8
2.7
2.6
2.5
2.4
2.3
2.2
2.1
2
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80


Mean Distance vs k
Mean Distance
k
1.35
1.3
1.25
1.2
1.15
1.1
1.05
1
2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62 64 66 68 70 72 74 76 78 80

Distance Per Rating

- We are interested in building a recommendation engine
- This means that we wish to predict high ratings correctly, and are less concerned about low
- We compare distance from each rating from 1 to 5
- These results will be especially interesting when we attempt to normalize global effects (correct for rater inconsistency)

Mean Distance and RSS for each Rating

Actual Rating	# Records	Mean Distance	Mean RSS
1	3870	2.319897	6.403101
2	8322	1.370704	2.714852
3	18756	0.576989	0.801663
4	29944	0.339701	0.411234
5	19182	0.823115	1.371546

k = 30 (best detected in terms of average RSS)
Thresholds = 25
# steps = 1000

Percentage of Businesses with Correct Predicted Ratings

Actual Rating	# Samples	# Correct	% Correct
1	3870	40	1.03
2	8322	608	7.31
3	18756	5311	28.32
4	29944	11737	39.20
5	19182	4438	23.14

k = 30 (best detected in terms of average RSS)
Thresholds = 25
# steps = 1000

Normalization of Global Effects

- Users and businesses tend to develop a bias. By normalizing, you are able to find the true interaction between a user and a business.
- The idea is to break a rating into different effects
  - Global effect: This is the average of all ratings
  - Business effect: The bias in the businesses ratings
  - User effect: The bias in the user’s ratings
  - Interaction: The specific interaction between the user and the business.
  - Other effects could make the users rating depend upon the businesses average, time, number of ratings, etc

Example of Normalization

- For example, imagine user HarvardHippie rates Hoagie Haven 1 star
  - We break this up into four effects
  - Global: The sites average rating is 3.2 stars
  - Hoagie Haven: Average rating is 4.5 stars, thus Hoagie Haven has a +1.3 effect
  - HarvardHippie: Average rating is 2.1 stars, thus has a -1.1 effect
  - Specific Interaction: This 1 star rating is thus broken into
    - 1 = 3.2 (global) + 1.3 (Hoagie Haven) – 1.1 (HarvardHippie) – 2.4 (Hoagie Haven to HarvardHippie specific)
    - Thus the specific interaction between Hoagie Haven and HarvardHippie is -2.4

Goal is Recommendation

- Our end goal is to recommend businesses to users, not necessarily predict a users specific rating.
- Final evaluation will be to see if we can accurately guess what someone will really like