Using SVD to Predict Movie Ratings

Serdar Sali

sali@soe.ucsc.edu

Collaborative filtering

- Similar tastes in the past : similar tastes in the future

Our domain: Movie Ratings

- We would like to predict how a user would rate a given movie
- A multi-label classification problem
  - Ratings : 1 to 5
  - Only available data : User – movie – rating triplets

Problem Formulation

- R : uxm matrix of ratings
  - u : number of users
  - m : number of movies
- Ideally, we would have:
  - A set of features for each movie: f_j
  - A set of preference multipliers for each user : call p_i
  - Then rating of user i for movie j becomes $r_{ij} = p_i f_j^T$

Problems

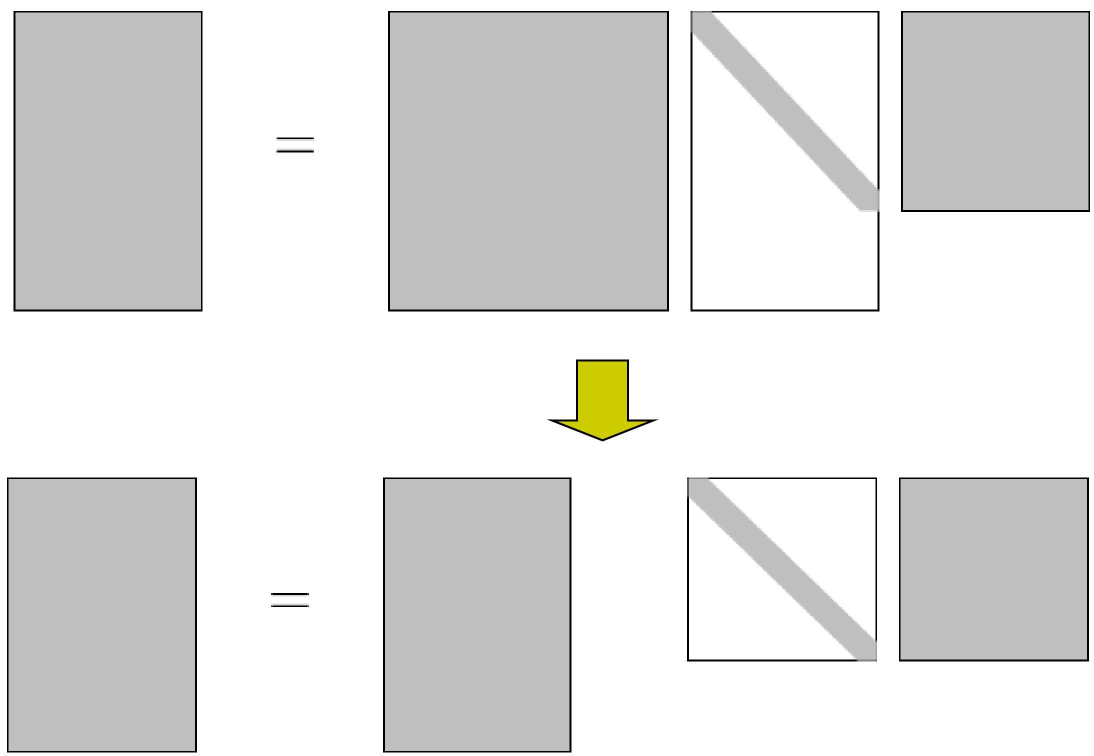
- Feature list for movies are hard to obtain
  - Task is inherently subjective and difficult, classification is hard
  - Dependence on external data resources
  - Tremendous effort required to clean-up data
- Notice that R already contains this data, but it is lumped together in a sum.
  - Can we retrieve it somehow?

Singular Value Decomposition

- SVD states that every mxn matrix A can be written as $A = USV^T$ where
  - *U is an mxm orthogonal matrix,*
  - *S is an mxn diagonal matrix with singular values of A along the diagonal,*
  - *V is an mxn orthogonal matrix.*

Full vs Reduced SVD

- Since S is diagonal, we can obtain a more compact representation:



SVD for Matrix Approximation

- Instead of using all the singular values of S, use only the most significant r
- Compute a rank-r approximation A' to A such that $A' = U'S'V'^T$ where U' is mxr, S' is rxr, and V' is mxr
- This approximation minimizes the Frobenius form: $||A-A'||_F = \mathrm{sqrt}(\sum(a_{ij}-a'_{ij})^2)$

SVD for Movie Rating Prediction

- Given a matrix of ratings R, we want to compute an approximate matrix R_{app} such that RMSE is minimized.
- But RMSE = $||R-R_{app}||_F$
- So, SVD is a perfect fit to our problem

SVD for Movie Rating Prediction

- Recall R_{uxm} : ratings matrix
- Compute an SVD for R and just lump the singular value matrix in the sum:
  - $R = P_{uxf} F_{mxf}^T$
  - P : Preference matrix for f features for u users
  - F : f-features matrix for m movies

But...

- SVD is not defined for sparse matrices
  - Netflix data: 8.5B possible entries, 8.4B empty
- Fill in with averages, some clever combinations
  - Perturbs the data too much
  - And even if we fill in the missing values...
- Computing SVD for large matrices is computationally very expensive

Incremental SVD Method

- Devised by Simon Funk
- Only consider existing values
- Do a gradient descent to minimize the error:
  - $E = (R-R_{app})_{ij}^2$
  - Take the derivative wrt p_{ij} and f_{jk}, and the updates become
    - $p_{ik}^{(t+1)} = p_{ik}^{(t)} + learning\_rate*(R-R_{app})_{ij}*f_{jk}^{(t)}$
    - $f_{jk}^{(t+1)} = f_{jk}^{(t)} + learning\_rate*(R-R_{app})_{ij}*p_{ik}^{(t)}$

Implementation

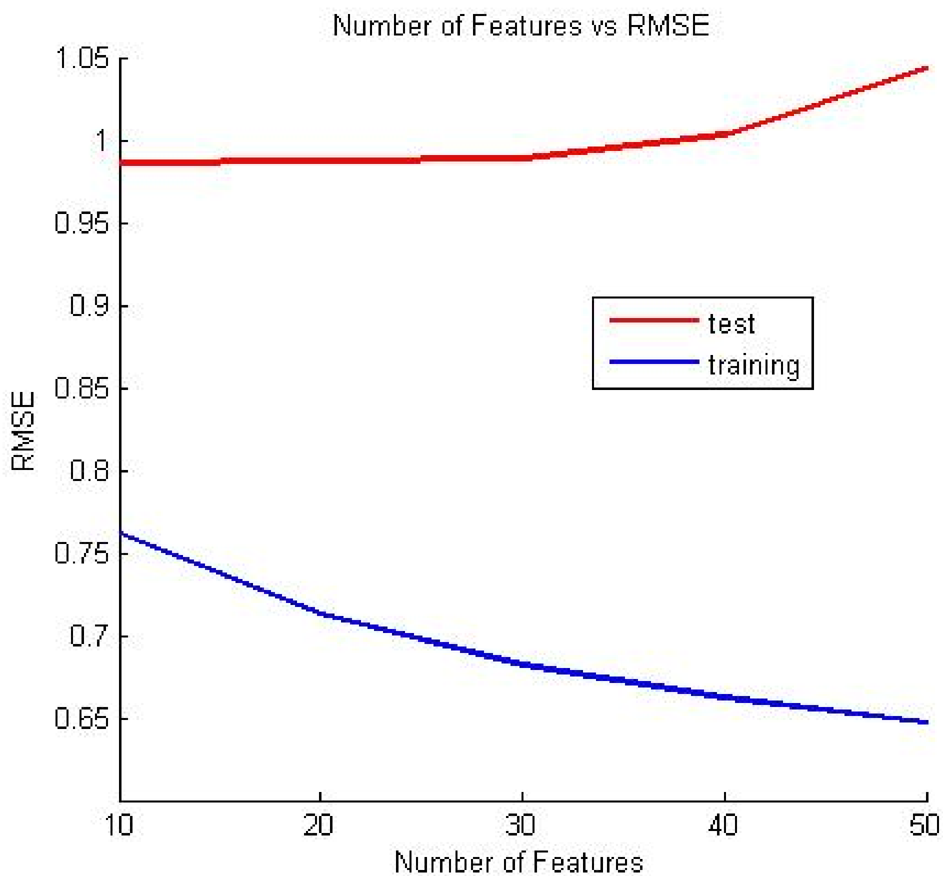
- Online update
- Set each feature & each multiplier to 0.1
- Train the most significant feature first, and then the second, etc.
- Parameters:
  - Number of features
  - Learning rate
  - Regularization : very simple : -K*(update target)
  - Different starting values

Experiments

- Smaller dataset
  - 2000 movies, 480189 users
  - 10314269 ratings
  - Just blind downsampling
- Test: predict 3000 ratings
- Does not perform as well as it does on whole dataset

Experiments

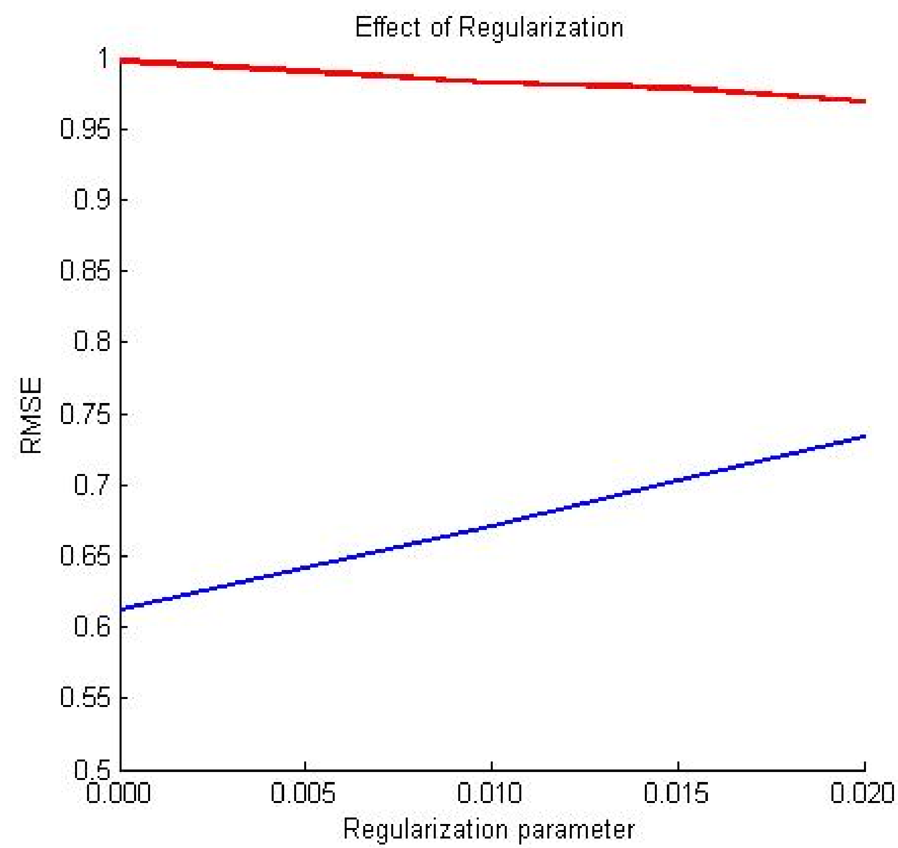
- Number of Features



#feats	Training	Test
10	0.7623	0.9858
20	0.7128	0.9869
30	0.6820	0.9882
40	0.6619	1.0028
50	0.6468	1.0438
Combined : 0.9895		

Experiments

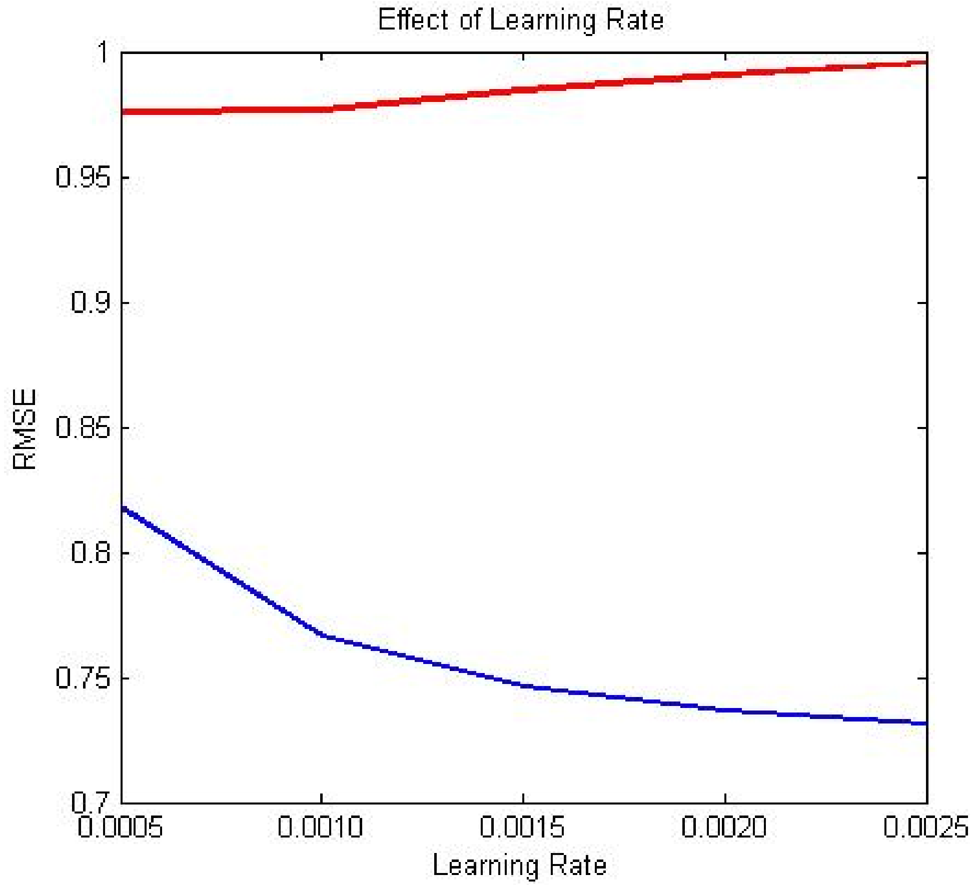
- Regularization



Reg Rate	Training	Test
0.0000	0.6118	0.9973
0.005	0.6418	0.9899
0.010	0.6705	0.9823
0.015	0.7027	0.9787
0.020	0.7333	0.9686
Combined : 0.9741		

Experiments

- Different learning rates



Lrate	Training	Test
0.0005	0.8180	0.9755
0.0010	0.7666	0.9765
0.0015	0.7464	0.9843
0.0020	0.7365	0.9904
0.0025	0.7314	0.9955
Combined : 0.9756		

Experiments

- Different starting values

	Base = 1	Base = Average	Base = Average + Offset
Training	0.7651	0.7638	0.7638
Test data	1.0144	1.0156	1.0158

- Combined : 0.9741

Conclusion

- Overfitting is a problem
  - Especially with my smaller dataset, models tend to overfit the training data very easily
- Even a blind combination of results give surprisingly good results
  - This implies that different models work good for different cases
  - A combination of different models is the way to go

Future Work

- Many parameters to adjust
- More clever downsampling of the dataset
- Use the computed features as input to another algorithm?
  - May help fine-tune the results at least
- Different regularization methods

Questions?