A STATISTICAL LEARNING MODEL FOR EMBEDDING RATING SCALE DATA

JORDAN E. TURLEY, ZEYANG HUANG, AND Dr. MICHAEL T. LAMAR

CENTRE COLLEGE

INTRODUCTION

The statistical learning model we present is adapted from the CODE model and the observation that constraining the embedding to a sphere can lead to improved performance on some data. We show that working on a torus rather than a sphere may be a more advantageous strategy in general.

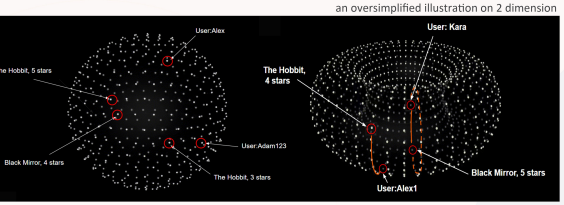
NETFLIX PRIZE

ABOUT THE DATATSET
- 100 million data points
- <user id, movie id, movie rating>
- Ratings from 1 to 5 stars
- 500,000 users, 20,000 movies

- A competition hosted by Netflix with $1,000,000 prize
- From 2006 to 2009
- Develop algorithm to predict movie ratings by users

MODEL

an oversimplified illustration on 2 dimension



DATA EMBEDDING ON A UNIT SPHERE

DATA EMBEDDING ON A TORUS

$$p(u,m,x) = \frac{1}{Z}\bar{p}(u)\bar{p}(m,x)\exp(-d^2(u,m,x))$$

$$Z = \sum_{u,m,x} \bar{p}(u)\bar{p}(m,x)\exp(-d^2(u,m,x))$$

NOTATION:

u - Individuals correspond to Netflix users.
m - Objects correspond to movies.
x - Integer ratings range from 1 to 5.
ϕ - Individuals' embedding function that maps the set of users to a vector in Euclidean space.
ψ - Embeding function that maps the set of movies and a rating to a vector in Euclidean space. Each movie has five corresponding rating vectors, one for each rating.

The p functions are empirical marginal probabilities measuring the relative frequency of each user and each movie-rating pair.

Z should be interpreted as the expected value of the negative exponential of the distance between embeddings chosen independently according to their empirical distributions.

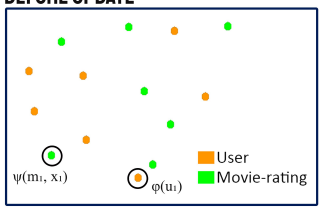
d(u,m,x) is the Euclidean distance between the embedding ϕ(u) and the embedding ψ(m,x).

ALGORITHM

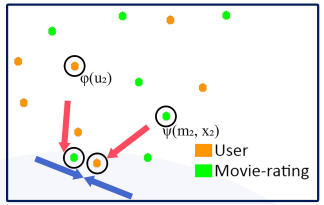
MOVE CLOSER TO ONE ANOTHER THE POINTS THAT OCCUR TOGETHER IN THE DATA SET. MOVE RANDOM PAIRS OF POINTS FURTHER APART. SEEN APART.

To optimize our likelihood function using this stochastic gradient ascent approach, we create an update rule that moves the embedding vectors for the current data point, ϕ(u1) and ψ(m1,x1).

BEFORE UPDATE



AFTER UPDATE



ATTRACTION

$$\phi(u_1) := [(1-\eta)\phi(u_1) + \eta\psi(m_1,x_1)] + \frac{\eta\exp(u_1,m_1,x_1)}{Z}[\phi(u_1) - \psi(m_2,x_2)]$$

These gradient update rules have a simple and intuitive interpretation. To update the embedding of the user, the first term in is a weighted average that moves the embedding of the user for the current data point in the direction of the embedding of the movie-rating for the current data point – a user's embedding vector moves toward the movie-rating vector for the user's chosen rating.

REPULSION

$$\psi(m_1,x_1) := [(1-\eta)\psi(m_1,x_1) + \eta\phi(u_1)] + \frac{\eta\exp(u_2,m_1,x_1)}{Z}[\psi(m_1,x_1) - \phi(u_2)]$$

This term moves the embedding for the user in the current data point away from the embedding of an independently chosen random movie-rating pair by an amount given by the ratio of their e^{-d^2} value to the current value of Z (which we recall is expected value of e^{-d^2}).

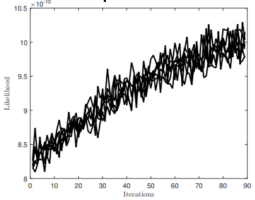
An additional repulsion term is included in the algorithm to accelerate the convergence and improve the predictive power of the embeddings.

PERFORMANCE

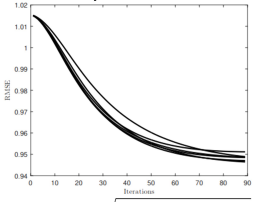
- Runtime per iteration = 20 minutes

Because the initializations are random, each run is slightly different. However, we observe that because of the sheer amount of data that we have, the RMSE of the runs is very similar. We observe the same thing with the likelihood; it is noisier as the likelihood is estimated through an average each iteration.

Likelihood convergence on a sphere



RMSE convergence on a sphere



$$RMSE = \sqrt{\frac{1}{N}\sum(Y_i - y_i)^2}$$

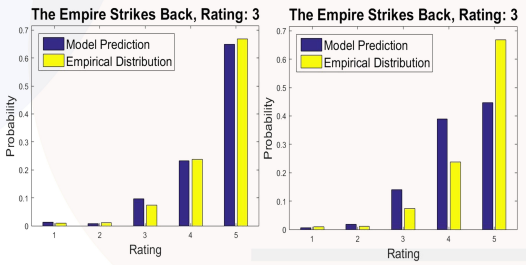
COMPARISON TO OTHER MODELS

- **Intuitive**: vectors closer together are more likely
- **Simple**: move likely data close together, unlikely farther apart
- **Fast**: good results after only a few iterations
- **Online**: new movies or users can be easily added

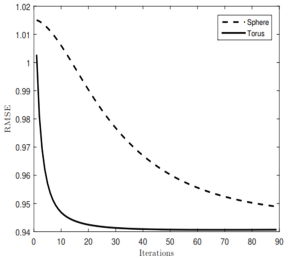
LEARNING

Here is an example of our model learning the preferences of one user.

We begin very close to the empirical distribution, which is simply the ratings this movie actually received, but after several iterations, we see the weight of the five rating goes down and the weight of the two, three, and four ratings go up. We can see that our model is learning the preferences of users in a meaningful way.



EXPERIMENTAL RESULTS



SPHERE vs TORUS

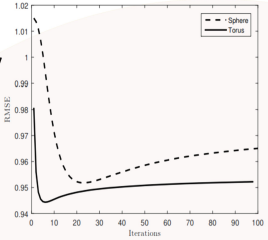
Here we compare the sphere and the torus. We see that convergence on the torus is significantly faster than convergence on the sphere.

The RMSE is also consistently lower on the torus than the sphere throughout training. After just eight iterations, the RMSE on the torus is within 1% of its final value. In contrast, the sphere requires over fifty iterations to be within 1% of its final value.

FAST CONVERGENCE

Attempting to force very rapid convergence can result in non-monotonic RMSE both on the sphere and the torus.

Again, we see the torus converges much more quickly than the sphere.



We can increase the step size to force convergence much faster. The torus reaches its lowest value after only six iterations, where the sphere reaches it after twenty-four. The minimum RMSE value of the torus is significantly lower than the minimum value of the sphere, and the value achieved is only marginally worse than the value achieved in the slower run.

Although not depicted, the RMSE performance is stable across a wide range of embedding dimensions slowly improving as the dimension grows.

CONCLUSION

Our work extends the model to allow for rating scale data rather than co-occurence data, and it demonstrates that working on a torus rather than a sphere can significantly improve performance for the rating scale data of the Netflix Prize.

The resulting embedding is able to successfully make predictions about the preferences of the individuals making it a valuable tool for enhancing customer experience for companies like Netflix.

FUTURE WORK
- Other datasets
- Clustering with the embedding

REFERENCE

[1] F. R. Bach and M. I. Jordan, Kernel independent component analysis, Journal of Machine Learning Research, (2002), pp. 1–48,
[2] R. M. Bell, Y. Koren, and C. Volinsky, The bellkor solution to the netflix prize. Netflix Prize Documentation, 2007,
[3] T. F. Cox and M. A. Cox, Multidimensional Scaling, Chapman and Hall/CRC, London, 2nd ed., 2000.285
[4] A. Globerson, G. Chechik, F. Pereira, and N. Tishby, Euclidean embedding of co-occurrence data, Journal of Machine Learning Research, (2007), pp. 2265–2295,
[5] J. H. Ham, D. D. Lee, and L. K. Saul, Learning high dimensional correspondences from low dimensional manifolds, Aug 2003,
[6] T. Iwata, K. Saito, N. Ueda, S. Stromsten, T. L. Griffiths, , and J. B. Tenenbaum, Parametric embedding for class visualization, Advances in Neural Information Processing Systems, (2005), pp.
[7] P. L. Lai and C. Fyfe, Kernel and nonlinear canonical correlation analysis, International Journal of Neural Systems, (2000), pp. 365–377,
[8] Y. Maron, M. Lamar, and E. Bienenstock, Sphere embedding: An application to part-of-speech induction, Advances in Neural Information Processing Systems, (2010), pp. 1567–1575
[9] S. T. Roweis and L. K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science, (2000), p. 23232326,
[10] J. B. Tenenbaum, V. D. Silva, and J. C. Langford, A global geometric framework for non-linear dimensionality reduction, Science, (2000), p.