Out-of-sample Extension for Latent Position Graphs

questions

- why do we use the square root of eigenvalues for ASE instead of the eigenvalues themselves?
- what is significant about the rows of Z? e.g. each latent position corresponds to a vector created by the i_{th}ith element of all eigenvectors. Why, how is that significant?
- what is a laplace-beltrami operator?
- does adding some new node zz change the latent positions of the other nodes if you re-embed?
- I keep reading about these kernel functions, how do they relate to ASE?
- what is a hollow matrix?
- how does the notation in (4) work?

Abstract

graph embedding works great
out-of-sample embedding presents a problem
we can show that we have a solution and it works better the higher your n

Introduction

ASE involves eigendecomposing some matrix AA, grabbing the top dd eigenvalues/vectors,
and creating Z=U_AS_A^{1/2}Z = UASA1/2.
Z is called the ASE of A.

what else can we talk about? well, out-of-sample embeddings are cool, but computationally expensive to redo every time you get a new node.
so we can do out-of-sample embedding. We can show that as the number of nodes nn increases, oos embeddings converge to the true latent position.

Framework

a lot of math that I don’t understand and hopefully isn’t super important

OOS extension

define some adjacency matrix AA and its ASE ZZ. Then,

(Z^TZ)^{-1}Z^T(ZT Z)−1ZT is the moore-penrose pseudoinverse \bold{Z^{\dagger}}Z†,
and T_n(X):=\bold{Z^{\dagger}\xi}Tn(X) := Z†ξ extends the embedding as best it can.

Importantly, note that T_n(X)Tn(X) is a random transformation, whose randomness comes from the fact that AA was random. So it won't actually be the true out-of-sample embedding, it's just the best we can do given the sample data.

OOS extension and Nystrom approximation

if AA is symmetric and SS has a lot more rows than columns, C=ASC = AS and A_s=S^TASAs = ST AS.
Then, CA_s^{\dagger}C^TCAs†CT is a low-rank approximation to AA.

Estimation of feature map

we have a pretty good bound on the error for T_nTn, the out-of-sample map.
<lots of heavy-duty theorems I don't fully understand>

Experimental Results

Experiment 1

sample points from a 2D gaussian. Then generate a latent position graph and classify with a least-squares regression. Then look at out-of-sample classification performance.

results

less than 2% performance degradation for oos

Experiment 2

uses abalone dataset from UCI ML repo.
we first make a graph using a latent position model, where X_i \in R^7Xi ∈ R7 represents the physical measurements of the i_{th}ith abalone observation. Performance degradation for oos was great.

Experiment 3

uses CharityNet dataset, which is 2 years of anonymized donations between donors and charities.
clustered, validated with ARI (adjusted rand index) between clustering labels and true labels
ARI obtained by clustering oos embedded charities is significantly better than chance

Conclusions

our paper is good and we talked about a lot of stuff