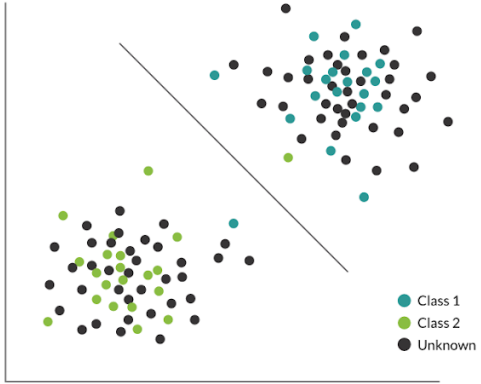
Embeddings: learning representations for unsupervised learning

Marc Lelarge Kevin Scaman Jill-Jênn Vie

Oct 28, 2022

Supervised learning

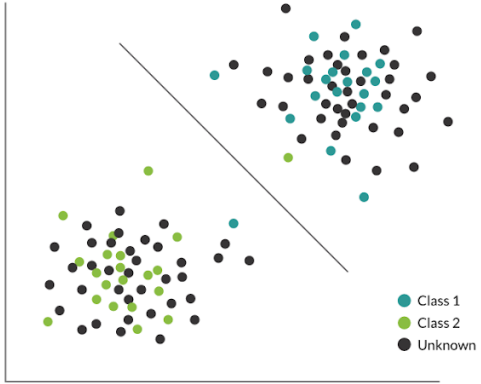
We observe $\boldsymbol{x}, y$
i.e. classification, regression



Unsupervised learning

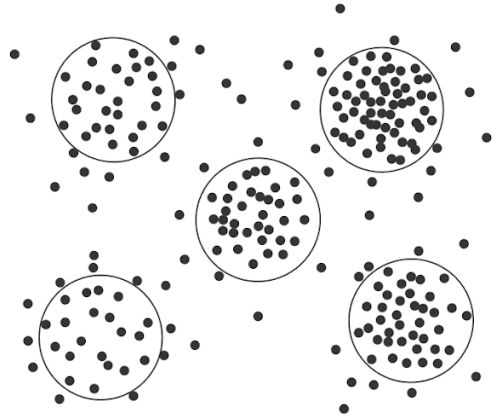
Supervised learning

We observe $\boldsymbol{x}, y$
i.e. classification, regression



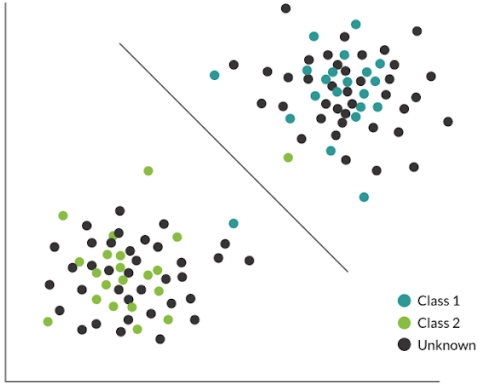
Unsupervised learning

We just observe $\boldsymbol{x}$
i.e. raw text, music, ratings



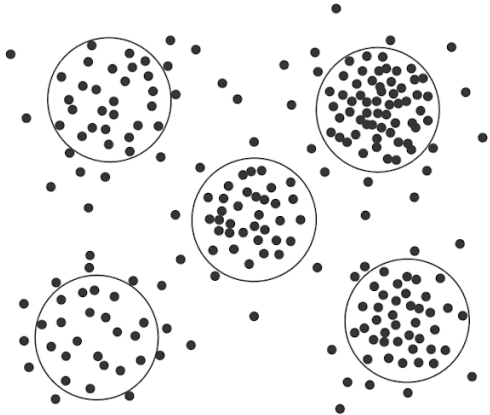
Supervised learning

We observe $\boldsymbol{x}, y$
i.e. classification, regression



Unsupervised learning

We just observe $\boldsymbol{x}$
i.e. raw text, music, ratings



It all deals with learning a representation of the data: embedding entities into $\mathbf{R}^d$

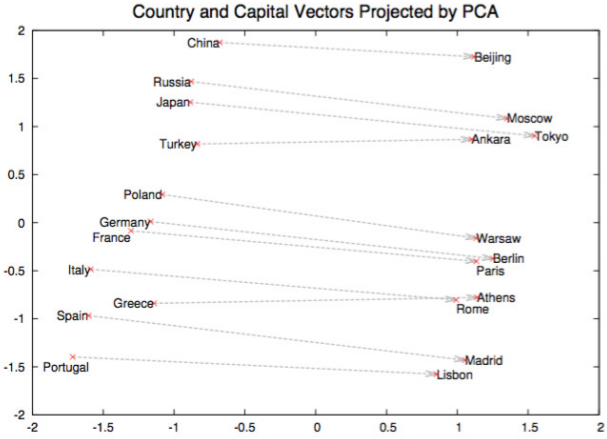
Context

- Dealing with unlabeled data, like raw text
- Dealing with tabular data, which mixes categorical and continuous variables
- Detecting outliers in the dataset
- Or mistakes in labels

Natural language processing

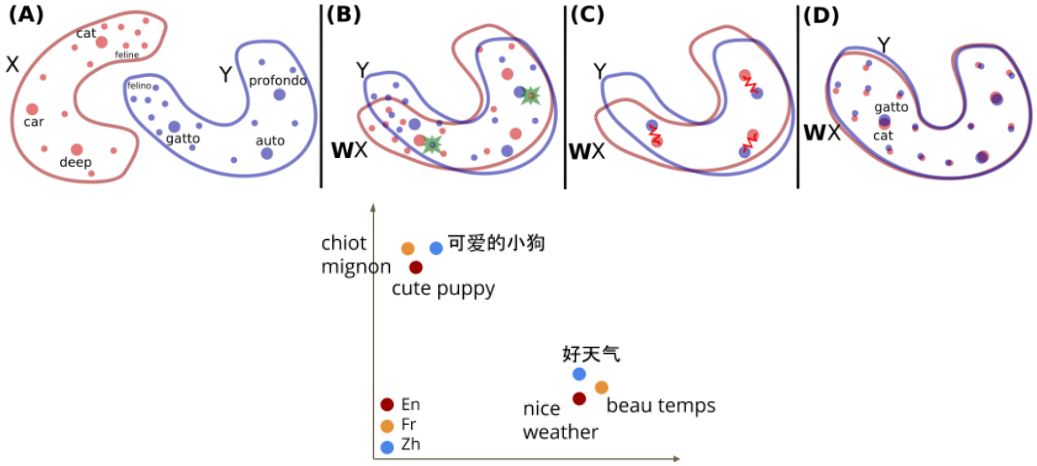
- A document can be represented as a sparse vector $\boldsymbol{x}$ of size 60,000 where $\boldsymbol{x}_i$ is the number of occurrences of the ith word.
- Could we guess meaning from context like humans?

Learned embeddings have interesting properties: word2vec



Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf

Multilingual word embeddings (unsupervised)



Guillaume Lample et al. "Word translation without parallel data". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=H196sainb

Application: Recommender Systems

Problem

- Every user rates few items (1 %)
- How to infer missing ratings?

Example

Sacha	?	5	2	?
Ondine	4	1	?	5
Pierre	3	3	1	4
Joëlle	5	?	2	?

Application: Recommender Systems

Problem

- Every user rates few items (1 %)
- How to infer missing ratings?

Example

Sacha	3	5	2	2
Ondine	4	1	4	5
Pierre	3	3	1	4
Joëlle	5	2	2	5

Encoding the problem

Matrix completion (unsupervised)

Given sparse entries M_{ij} find the other entries M_{ij}

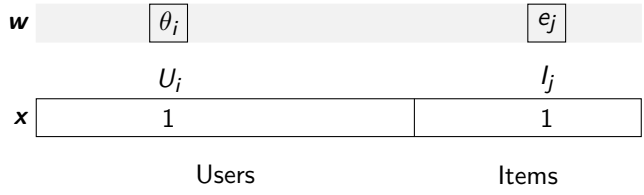
Supervised problem

- $\boldsymbol{X} = \{(i, j)\}$ are the categorical observed indices (ex. user i and item j)
- $\boldsymbol{y} = \{R_{ij}\}$ are the continuous (or categorical) ratings

Learning embeddings

Embedding n entities into $\mathbf{R}^d$ is equivalent to learning a $n \times d$ matrix $\boldsymbol{W}$ s.t. embedding of entity i is W_i.

One-hot encoding $\boldsymbol{e}_j = (0, \ldots, 0, \underbrace{1}_{j}, 0, \ldots, 0)$ so that $\boldsymbol{W}\boldsymbol{e}_j = W_j$



Dimensionality reduction, or best approximation of rank k

Input: $n \times m$ matrix $\boldsymbol{X}$

Minimize: $||\boldsymbol{X} - M||_2^2 = \sum_{i,j}(X_{ij} - M_{ij})^2$ where $rank(M) = k$

Dimensionality reduction, or best approximation of rank k

Input: $n \times m$ matrix $\boldsymbol{X}$

Minimize: $||\boldsymbol{X} - M||_2^2 = \sum_{i,j}(X_{ij} - M_{ij})^2$ where $rank(M) = k$

Solution: compute the singular value decomposition (SVD) $\boldsymbol{X} = U\Sigma V^T$ where

- U $(n \times r)$ is orthogonal: $U^T U = I_r$ where $r = rank(\boldsymbol{X})$
- Σ $(r \times r)$ is diagonal: its values $\lambda_1 \geq \ldots \geq \lambda_r > 0$
- V $(m \times r)$ is orthogonal: $V^T V = I_r$

k-SVD is truncated to k biggest singular values. $M = U_k \Sigma_k V_k^T$ is solution. And $W = U_k \Sigma_k$ is a $n \times k$ matrix of embeddings (`pca.transform`).

KNN $\rightarrow$ measure similarity between users (or items)

K-nearest neighbors

- R_u represents the row vector of user u in the rating matrix (users $\times$ items).
- Similarity score between users (cosine):

$$score(u, v) = \frac{R_u \cdot R_v}{||R_u|| \cdot ||R_v||}.$$

- Let's identify the k-nearest neighbors of user u
- And recommend to user u what u's neighbors liked, but u didn't watch

KNN $\rightarrow$ measure similarity between users (or items)

K-nearest neighbors

- R_u represents the row vector of user u in the rating matrix (users $\times$ items).
- Similarity score between users (cosine):

$$score(u, v) = \frac{R_u \cdot R_v}{||R_u|| \cdot ||R_v||}.$$

- Let's identify the k-nearest neighbors of user u
- And recommend to user u what u's neighbors liked, but u didn't watch

Hint

If R' the $N \times M$ matrix of rows $\frac{R_u}{||R_u||}$, we can get the $N \times N$ score matrix by computing $R'R'^T$. May not fit in memory.

Matrix factorization $\rightarrow$ reduce dimension to generalize

$$R = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{pmatrix} = \boxed{} = \boxed{U} \boxed{P}$$

R: 600 users $\times$ 9k works $\iff$ $\begin{cases} U\text{: 600 users} \times \text{20 profiles} \\ P\text{: 20 profiles} \times \text{9k works} \end{cases}$

$\mathbf{R}_{\text{Bob}}$ is a linear combination of profiles P_1, P_2, etc.

Matrix factorization $\rightarrow$ reduce dimension to generalize

$$R = \begin{pmatrix} R_1 \\ R_2 \\ \vdots \\ R_n \end{pmatrix} = \square = U \; P$$

R: 600 users $\times$ 9k works $\iff$ $\begin{cases} U\text{: 600 users} \times \text{20 profiles} \\ P\text{: 20 profiles} \times \text{9k works} \end{cases}$

$\mathbf{R}_{\text{Bob}}$ is a linear combination of profiles P_1, P_2, etc.

Interpreting Key Profiles

If P	P_1: adventure	P_2: romance	P_3: plot twist
And U_{Bob}	0.2	-0.5	0.6

$\Rightarrow$ Bob likes a bit adventure, hates romance, loves plot twists.

Matrix factorization $\rightarrow$ reduce dimension to generalize

Train

- R ratings, U user embeddings, V item embeddings.

$$R = UV^T \qquad \hat{r}_{ij} = \boldsymbol{u}_i^T \boldsymbol{v}_j$$

Matrix factorization $\rightarrow$ reduce dimension to generalize

Train

- R ratings, U user embeddings, V item embeddings.

$$R = UV^T \qquad \hat{r}_{ij} = \boldsymbol{u}_i^T \boldsymbol{v}_j$$

Test: Will user i' like item j'?

- Just compute $\boldsymbol{u}_{i'}^T \boldsymbol{v}_{j'}$

Objective: regularized squared loss

$$\mathcal{L} = \sum_{i,j} (\underbrace{\boldsymbol{u}_i^T \boldsymbol{v}_j}_{\text{pred}} - \underbrace{r_{ij}}_{\text{real}})^2 + \underbrace{\lambda||\boldsymbol{u}_i||_2^2 + \lambda||\boldsymbol{v}_j||_2^2}_{\text{regularization}}$$

Variants

SVD: $\sum_{\text{all } i,j} (\boldsymbol{u}_i^T \boldsymbol{v}_j - r_{ij})^2$ (missing pairs are imputed to zero)

ALS-WR : $\sum_{i,j \text{ known}} (\boldsymbol{u}_i^T \boldsymbol{v}_j - r_{ij})^2 + \lambda(\sum_i N_i||\boldsymbol{u}_i||^2 + \sum_j M_j||\boldsymbol{v}_j||^2)$

where N_i (resp. M_j): how many times user i rated items (resp. item j was rated)

Computing the gradients

$$\mathcal{L} = \sum_{i,j} (\underbrace{\boldsymbol{u}_i^T \boldsymbol{v}_j}_{\text{pred}} - \underbrace{r_{ij}}_{\text{real}})^2 + \underbrace{\lambda ||\boldsymbol{u}_i||_2^2 + \lambda ||\boldsymbol{v}_j||_2^2}_{\text{regularization}}$$

Computing the gradients

$$\mathcal{L} = \sum_{i,j} (\underbrace{\boldsymbol{u}_i^T \boldsymbol{v}_j}_{\text{pred}} - \underbrace{r_{ij}}_{\text{real}})^2 + \underbrace{\lambda ||\boldsymbol{u}_i||_2^2 + \lambda ||\boldsymbol{v}_j||_2^2}_{\text{regularization}}$$

$\mathcal{L}$ is (polynomial and) convex in each $\boldsymbol{u}_k$ or $\boldsymbol{v}_\ell$.

Finding $\boldsymbol{u}_k$ that minimizes $\mathcal{L} \Rightarrow$ Finding the zeroes of

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_k} = \sum_{j \text{ rated by } k} 2(\boldsymbol{u}_k{}^T \boldsymbol{v}_j - r_{kj})\boldsymbol{v}_j + 2\lambda \boldsymbol{u}_k = 0$$

can be rewritten $A\boldsymbol{u}_k = B$ so $\boldsymbol{u}_k = A^{-1}B$ (closed form)

Complexity: $O(d^3)$ where d is the (embedding) size of A (but can be parallelized)

Computing the gradients

$$\mathcal{L} = \sum_{i,j} (\underbrace{\boldsymbol{u}_i^T \boldsymbol{v}_j}_{\text{pred}} - \underbrace{r_{ij}}_{\text{real}})^2 + \underbrace{\lambda ||\boldsymbol{u}_i||_2^2 + \lambda ||\boldsymbol{v}_j||_2^2}_{\text{regularization}}$$

$\mathcal{L}$ is (polynomial and) convex in each $\boldsymbol{u}_k$ or $\boldsymbol{v}_\ell$.

Finding $\boldsymbol{u}_k$ that minimizes $\mathcal{L} \Rightarrow$ Finding the zeroes of

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{u}_k} = \sum_{j \text{ rated by } k} 2(\boldsymbol{u}_k{}^T \boldsymbol{v}_j - r_{kj})\boldsymbol{v}_j + 2\lambda \boldsymbol{u}_k = 0$$

can be rewritten $A\boldsymbol{u}_k = B$ so $\boldsymbol{u}_k = A^{-1}B$ (closed form)

Complexity: $O(d^3)$ where d is the (embedding) size of A (but can be parallelized)

Algorithm ALS: Alternating Least Squares (Zhou, 2008)

- Until convergence (few iterations):
  - Fix U (users) learn V (items) in order to minimize $\mathcal{L}$
  - Fix V find U

Illustration of Alternating Least Squares

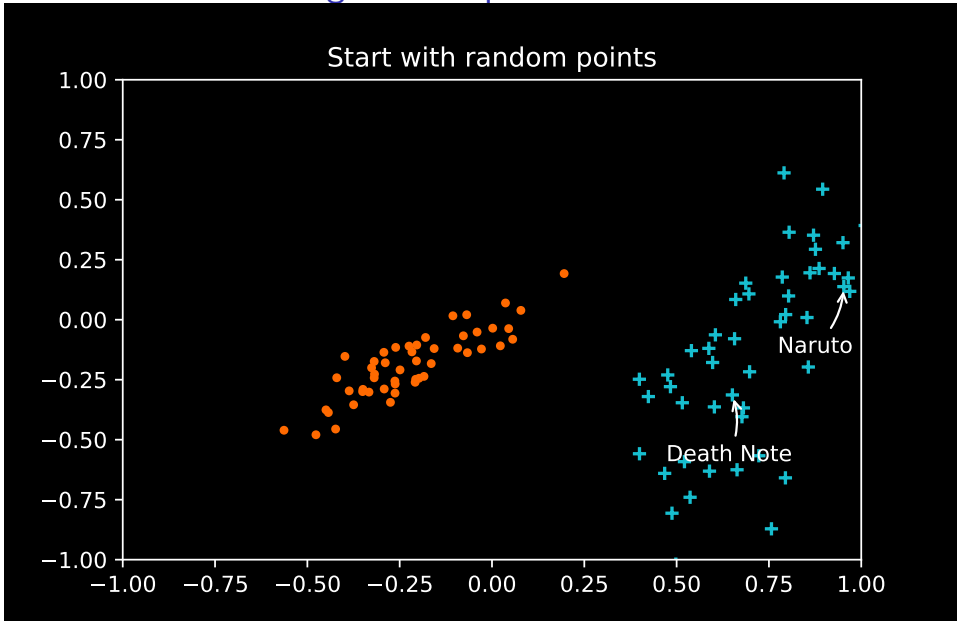


Illustration of Alternating Least Squares

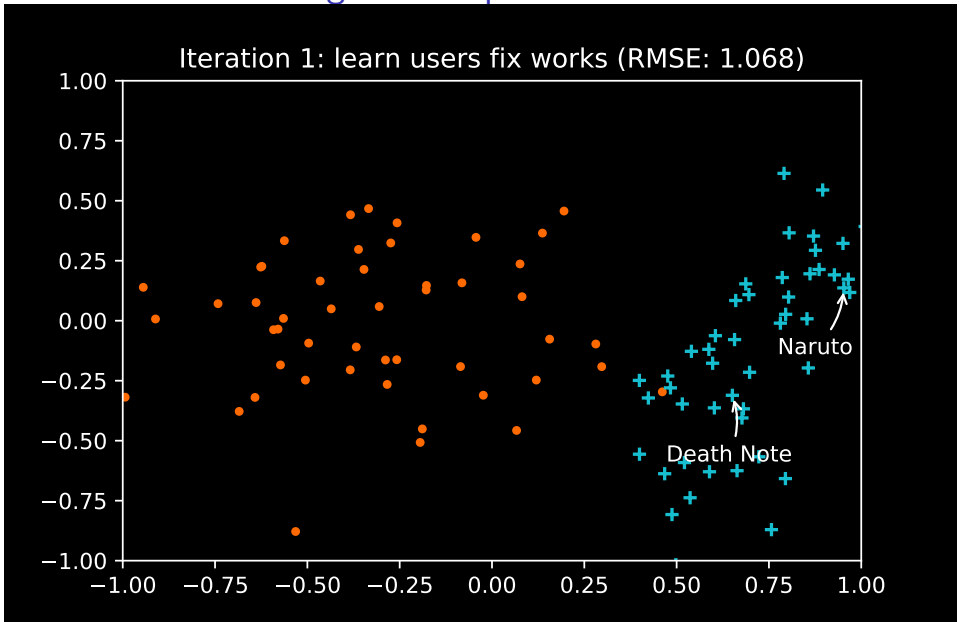


Illustration of Alternating Least Squares

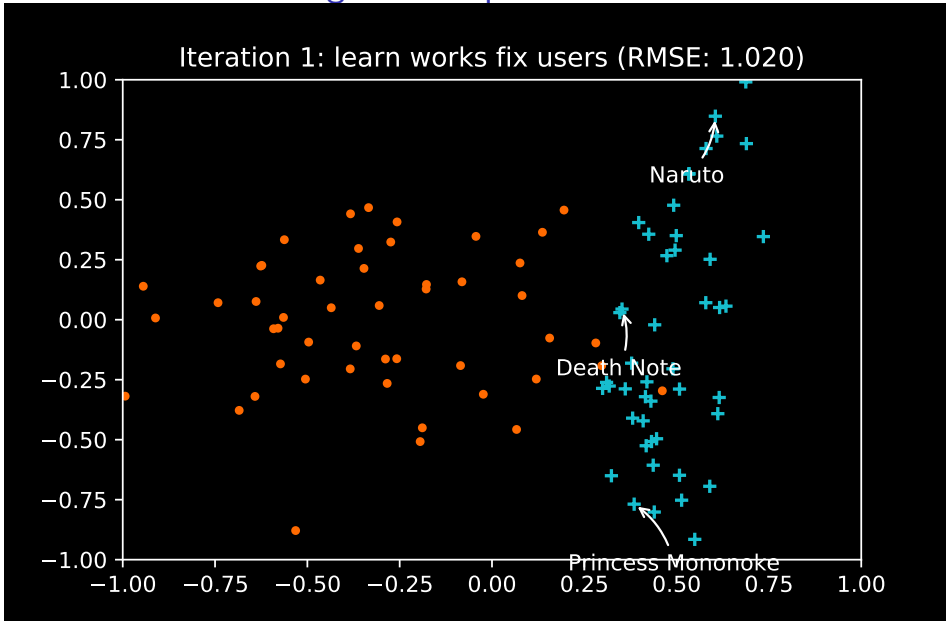


Illustration of Alternating Least Squares

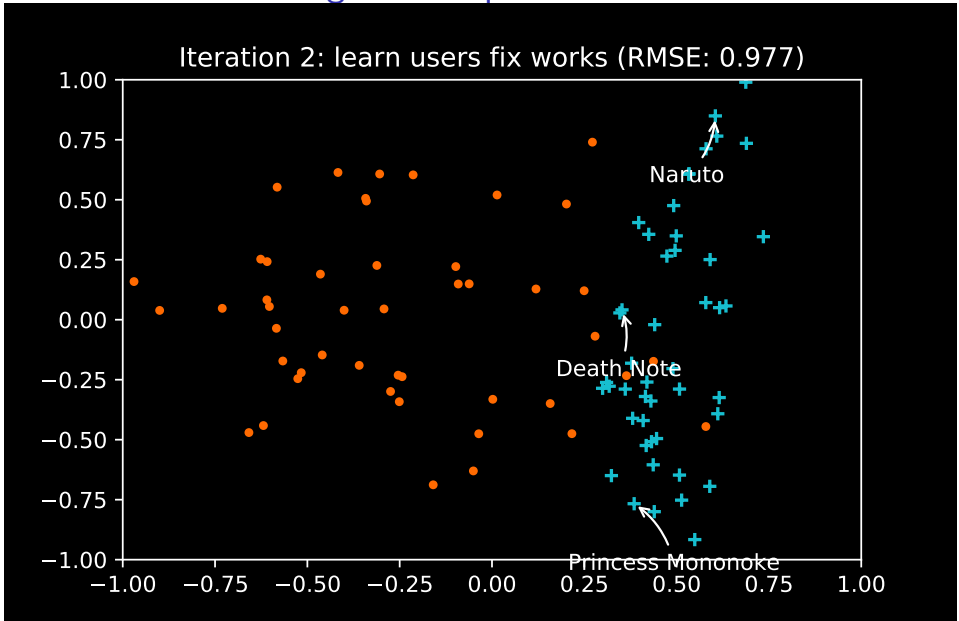


Illustration of Alternating Least Squares

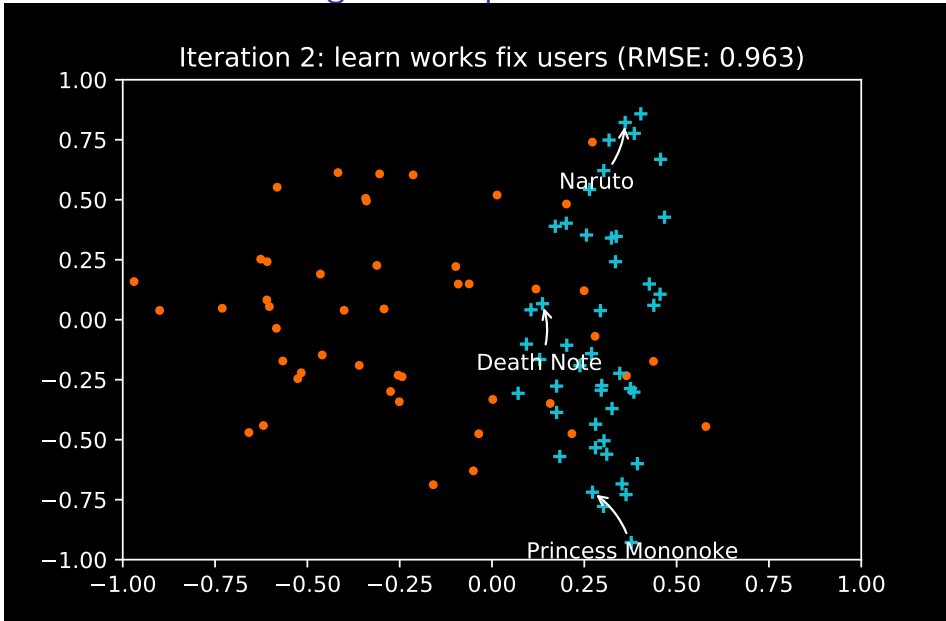


Illustration of Alternating Least Squares

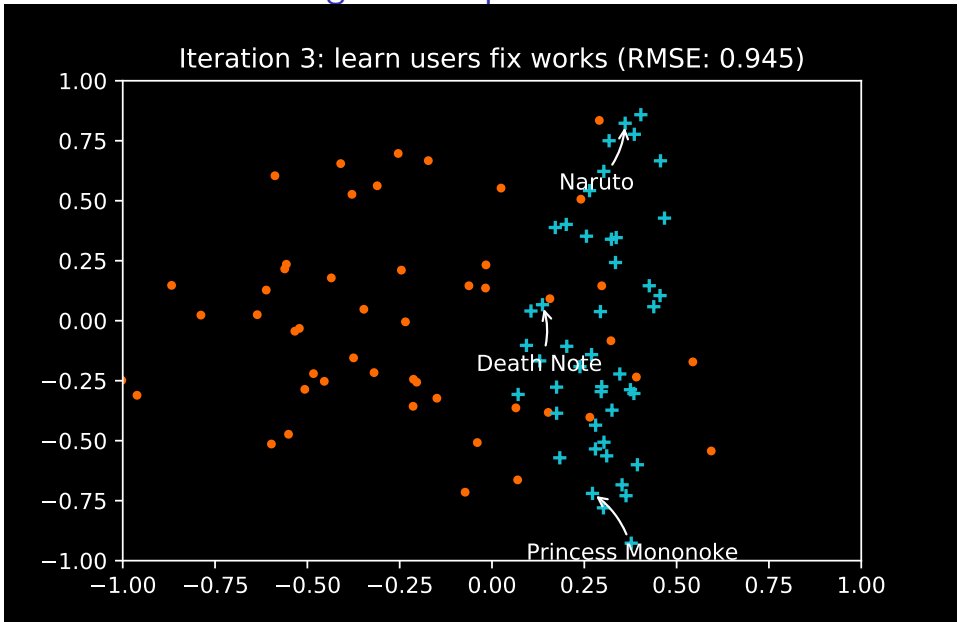


Illustration of Alternating Least Squares

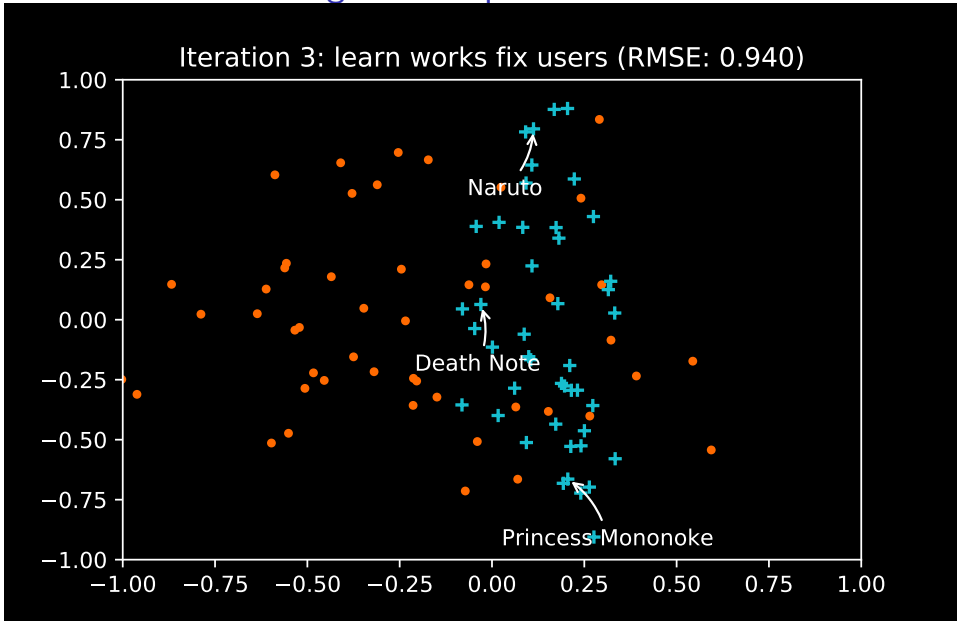


Illustration of Alternating Least Squares

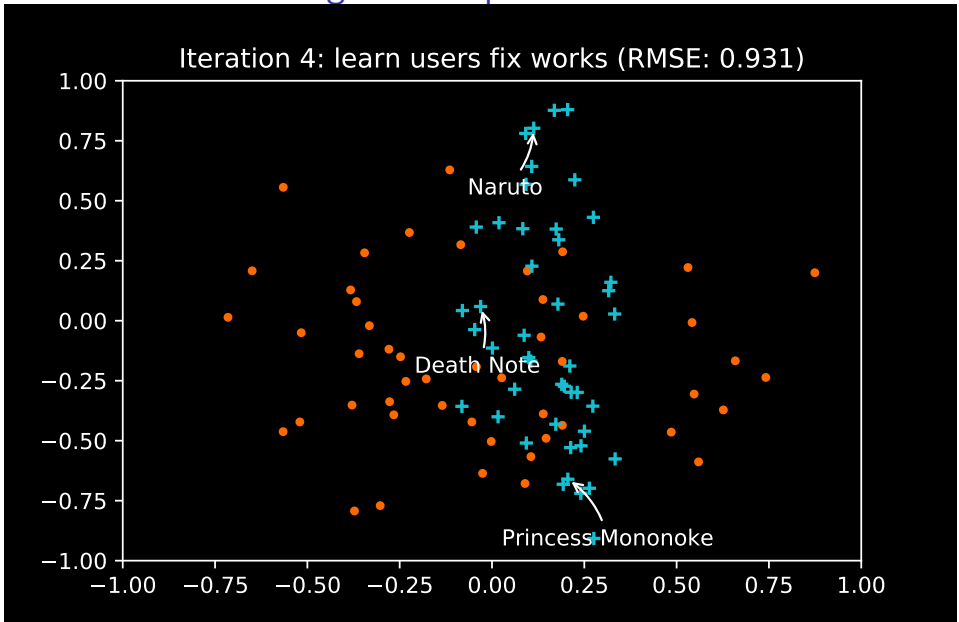


Illustration of Alternating Least Squares

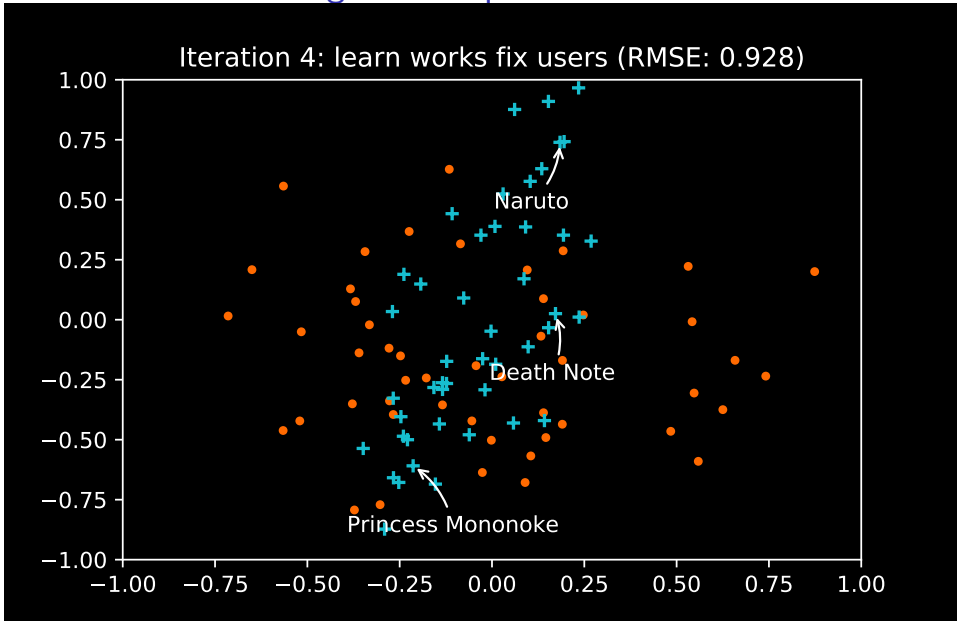


Illustration of Alternating Least Squares

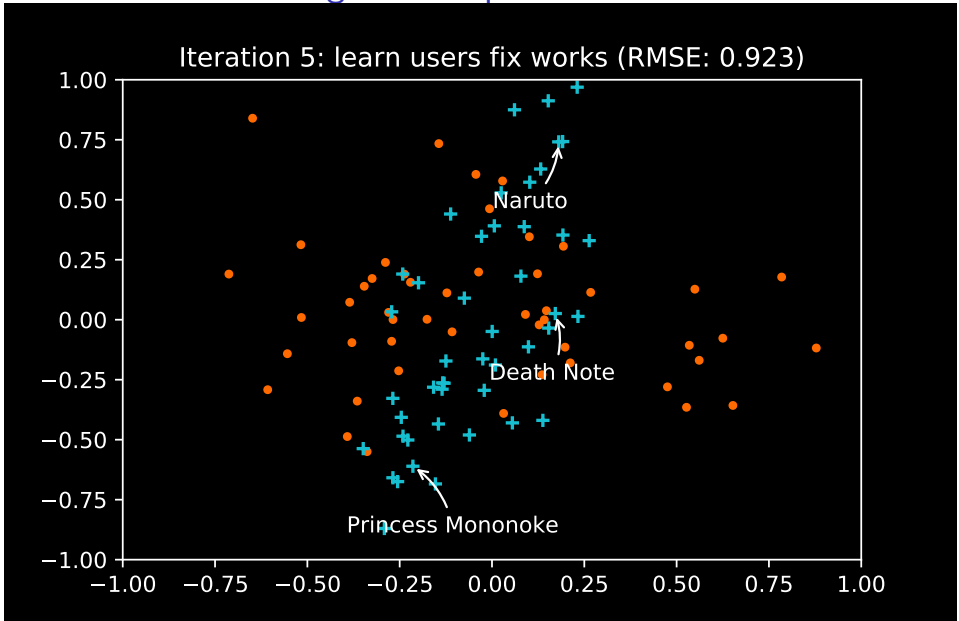


Illustration of Alternating Least Squares

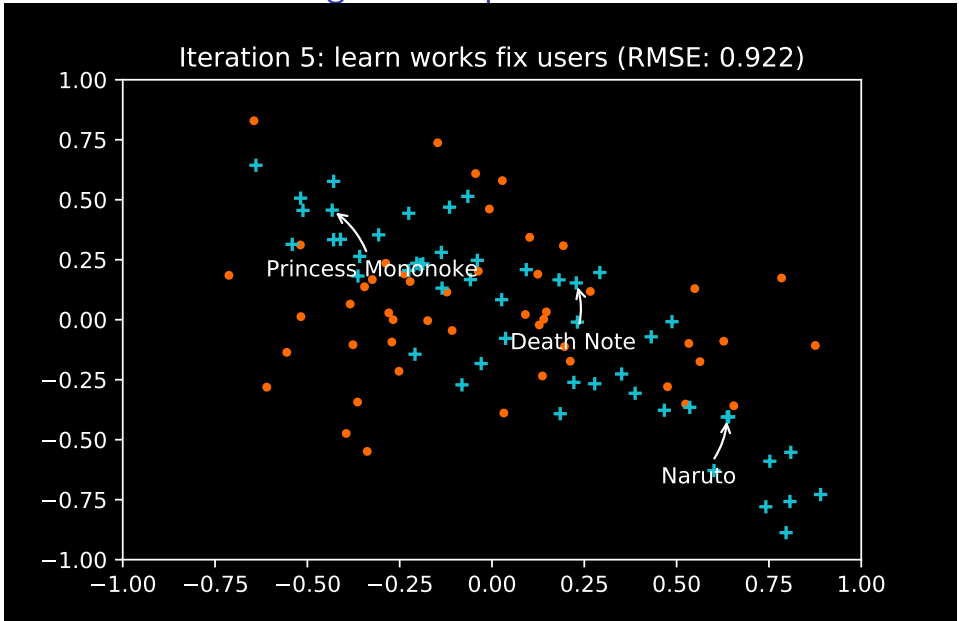


Illustration of Alternating Least Squares

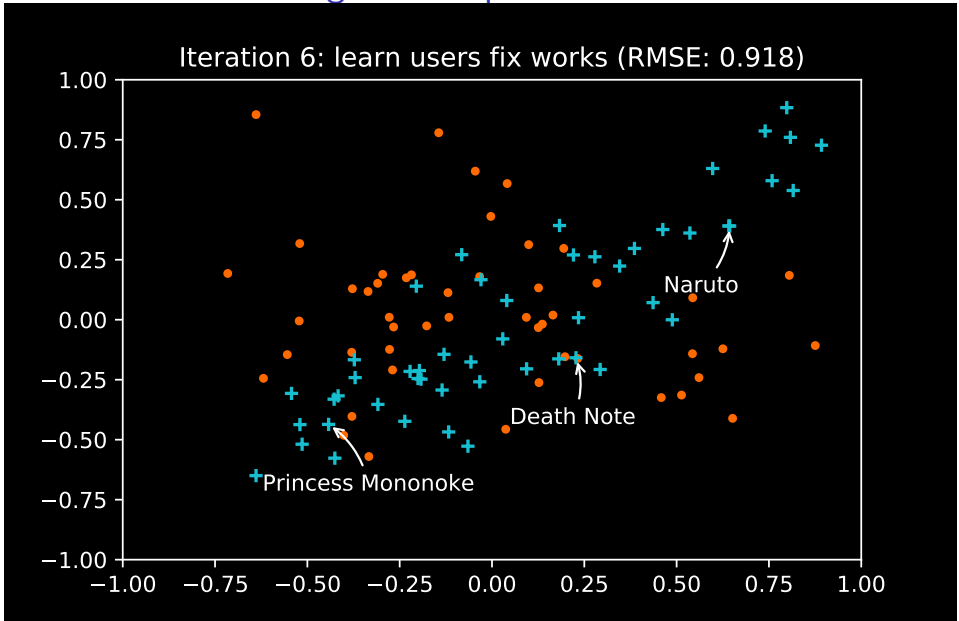


Illustration of Alternating Least Squares

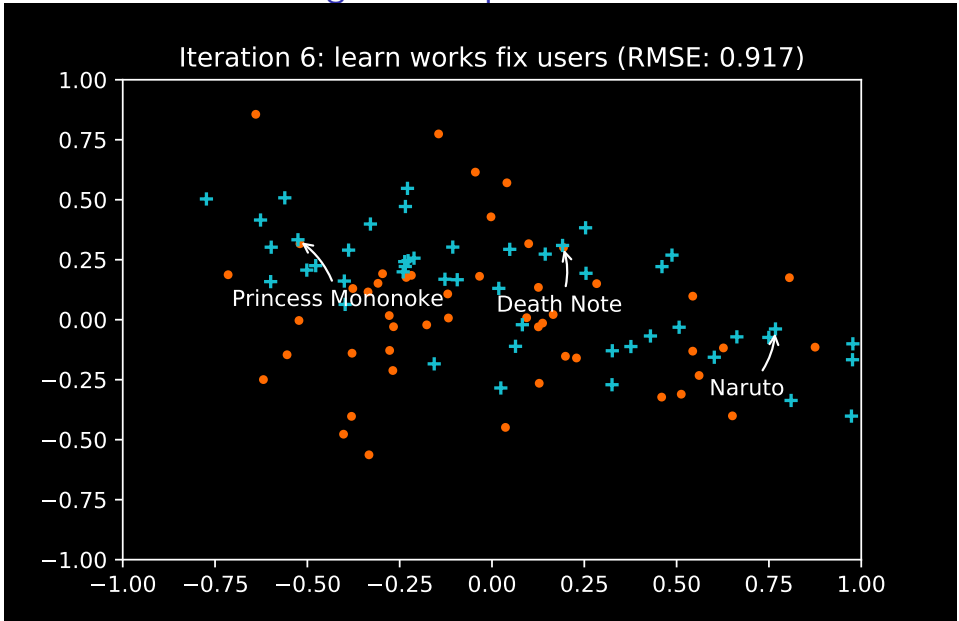


Illustration of Alternating Least Squares

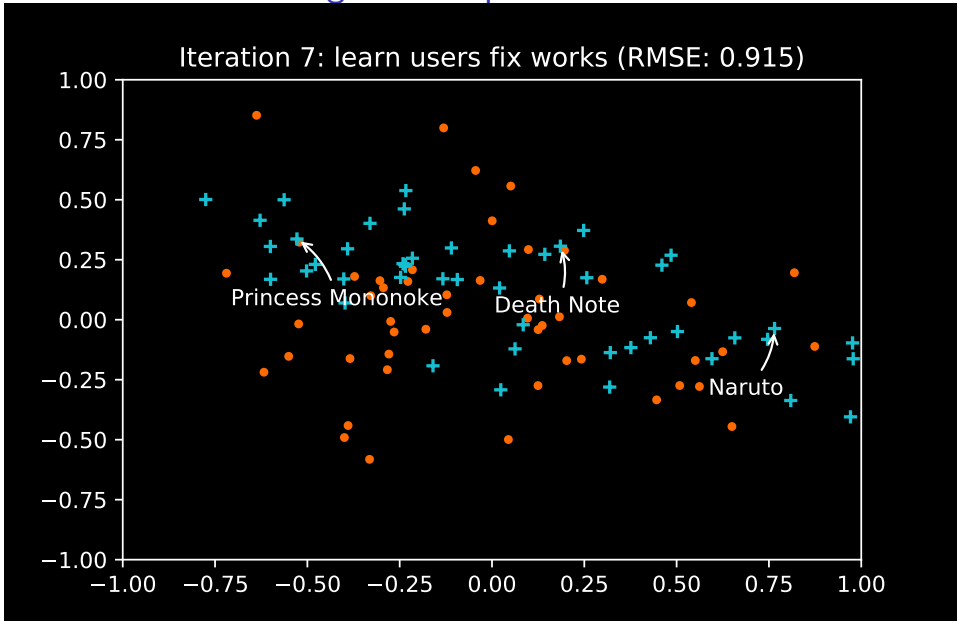


Illustration of Alternating Least Squares

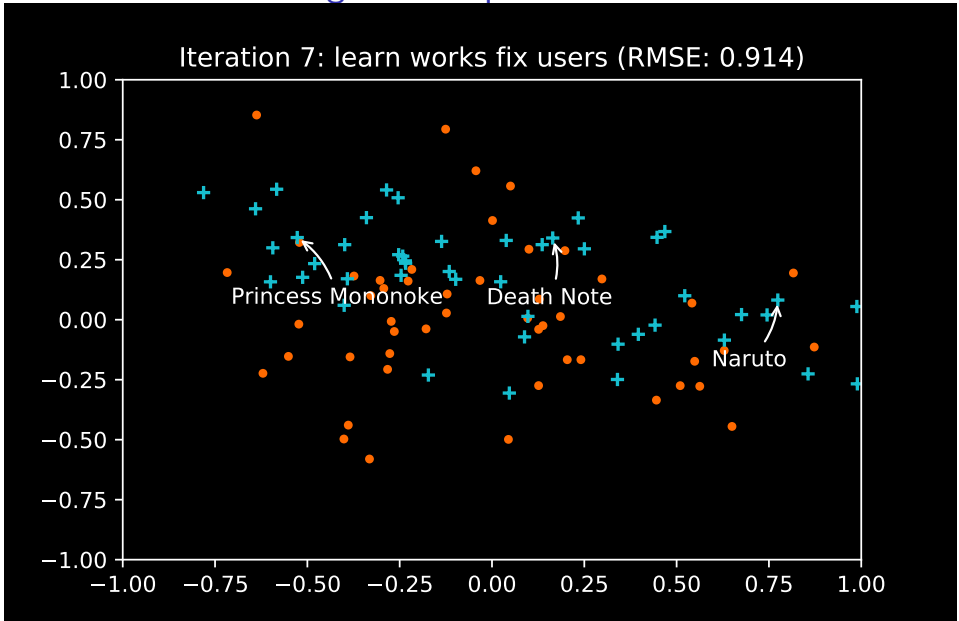


Illustration of Alternating Least Squares

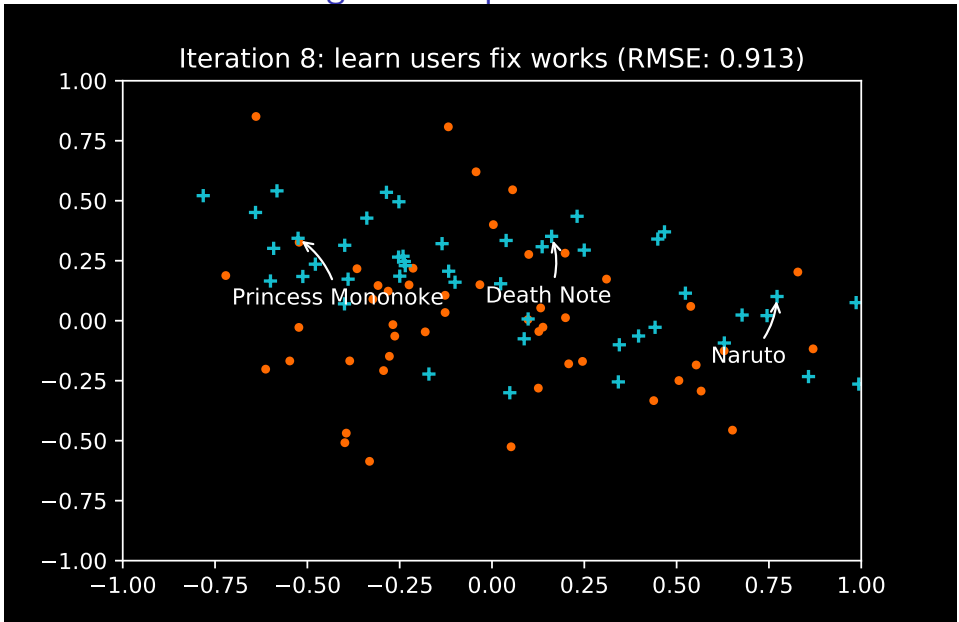


Illustration of Alternating Least Squares

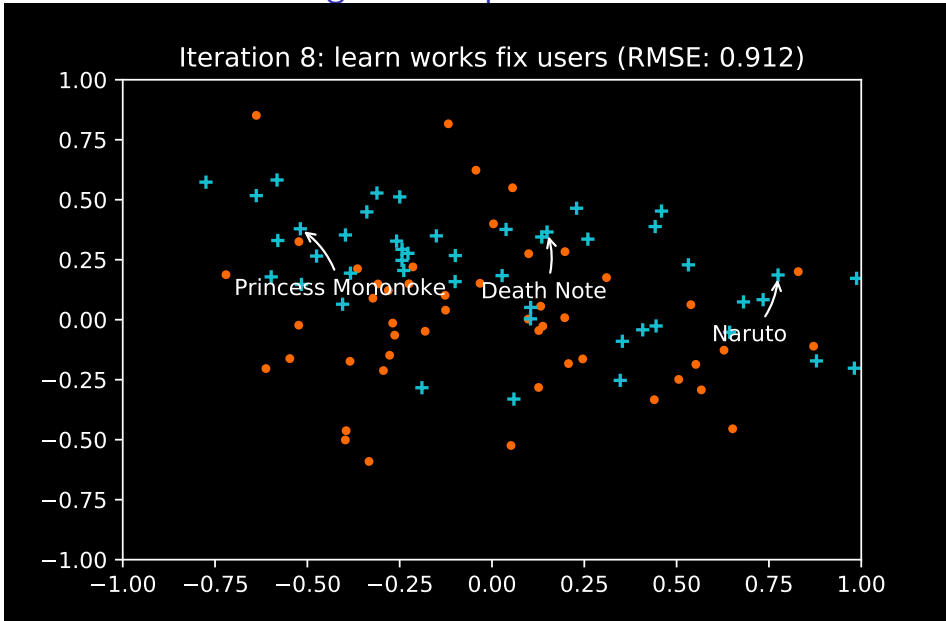


Illustration of Alternating Least Squares

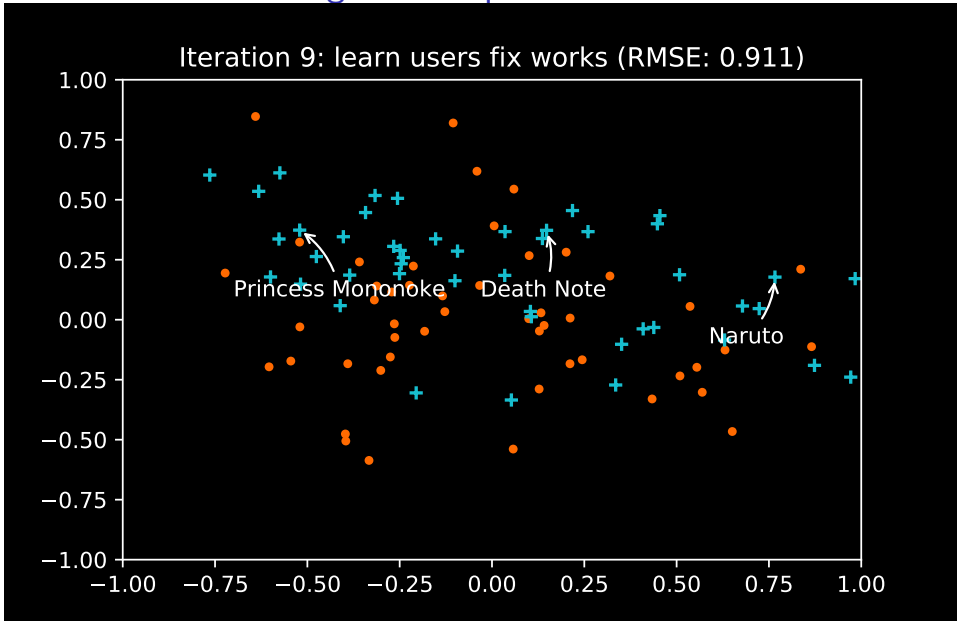


Illustration of Alternating Least Squares

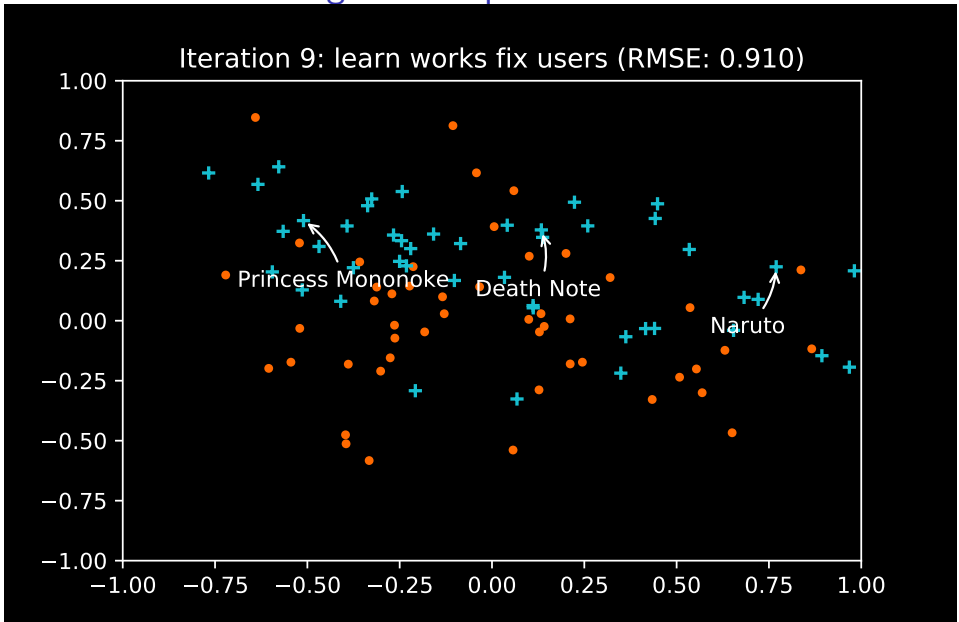


Illustration of Alternating Least Squares

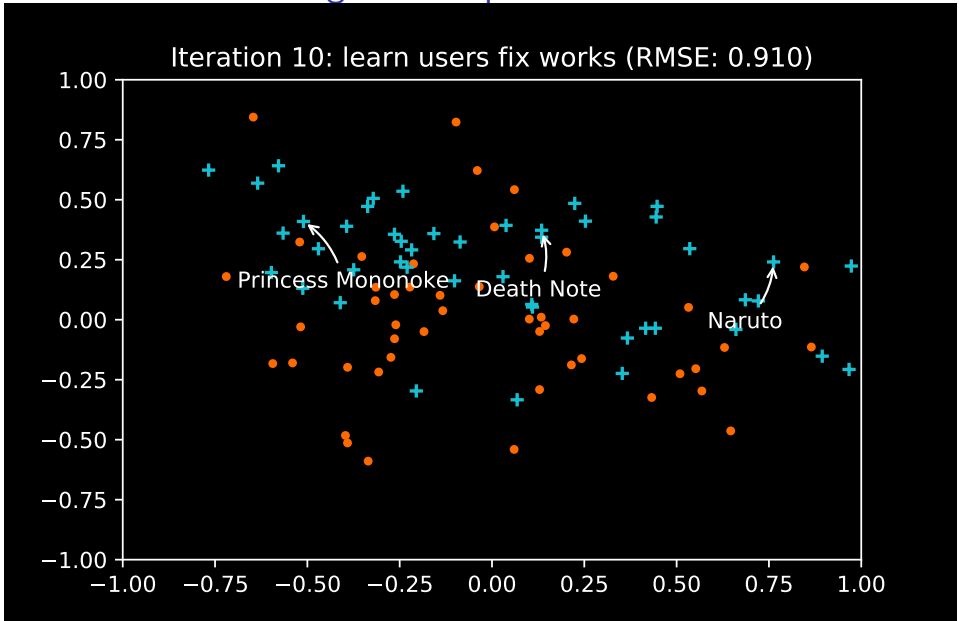


Illustration of Alternating Least Squares

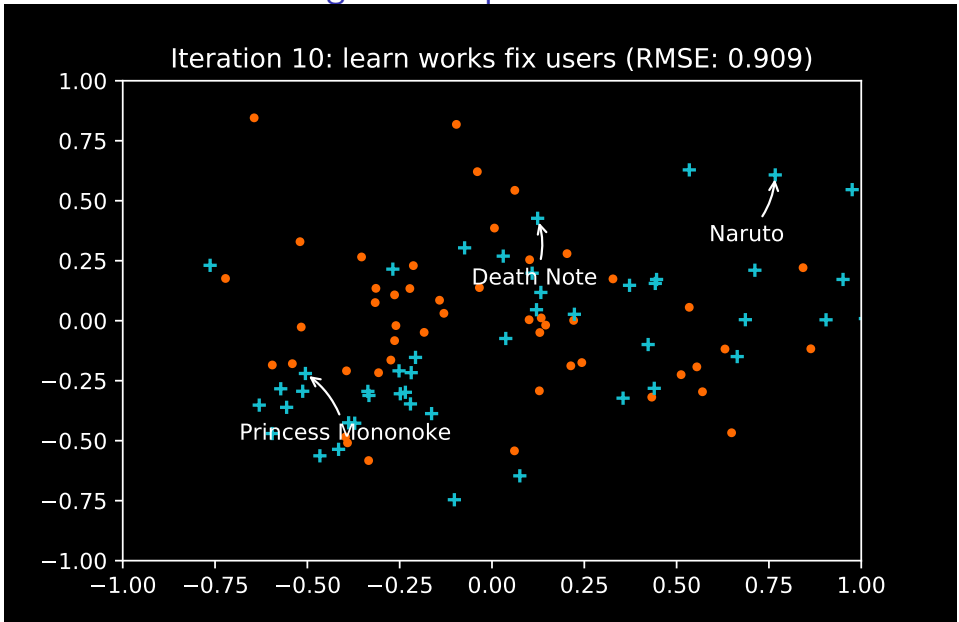


Illustration of Alternating Least Squares

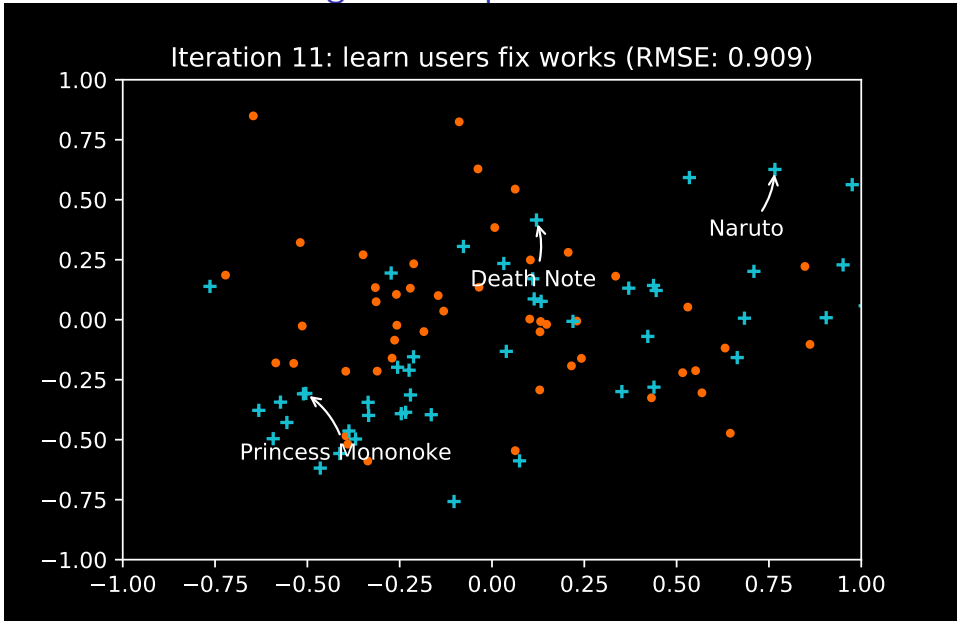


Illustration of Alternating Least Squares

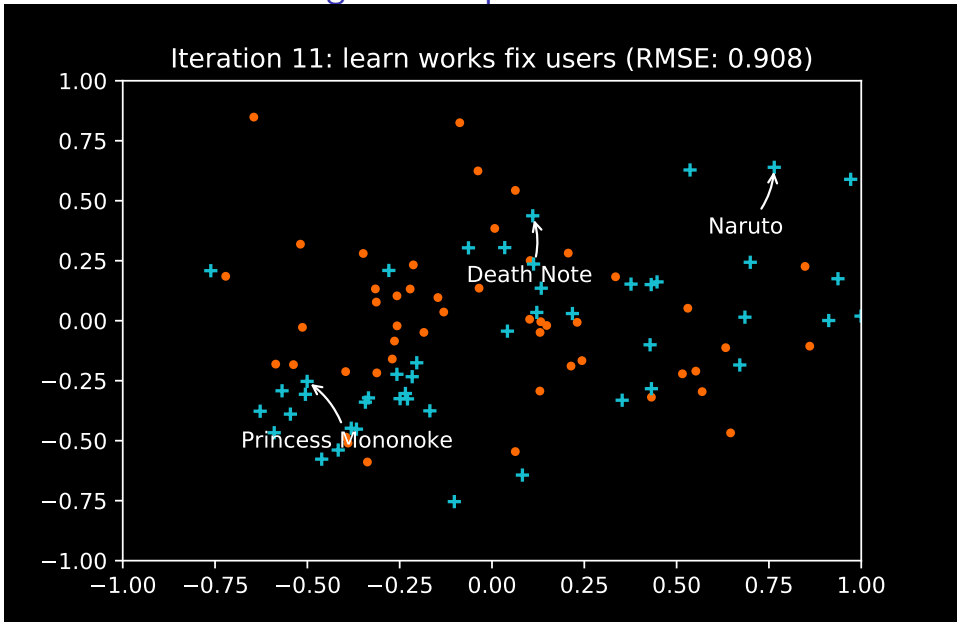


Illustration of Alternating Least Squares

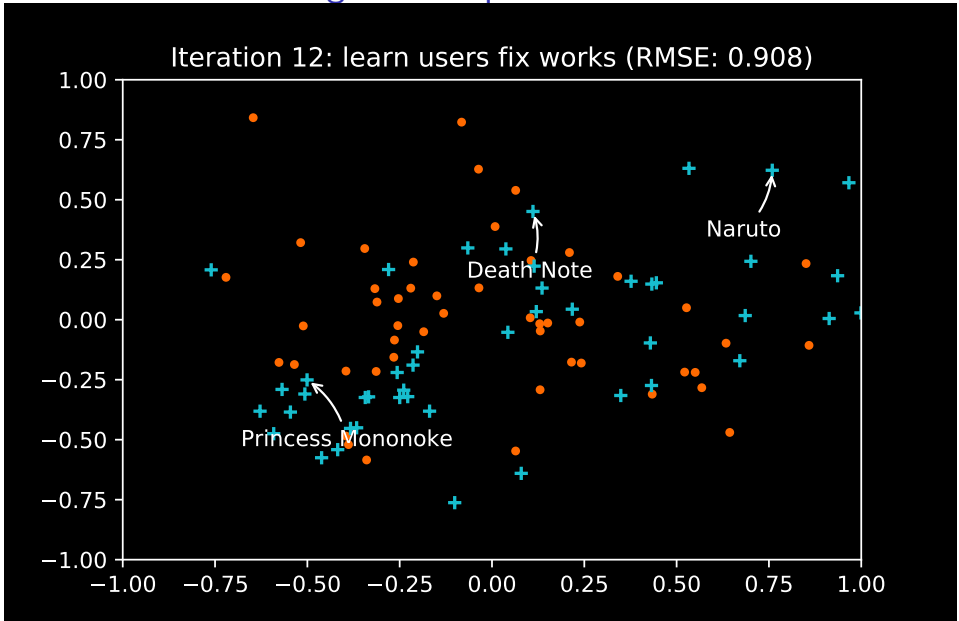


Illustration of Alternating Least Squares

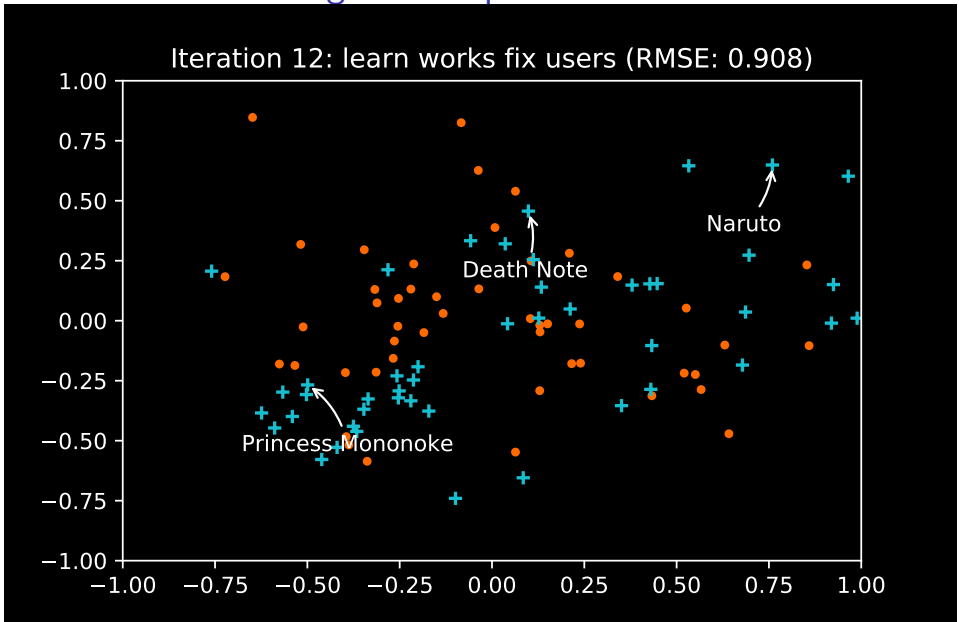


Illustration of Alternating Least Squares

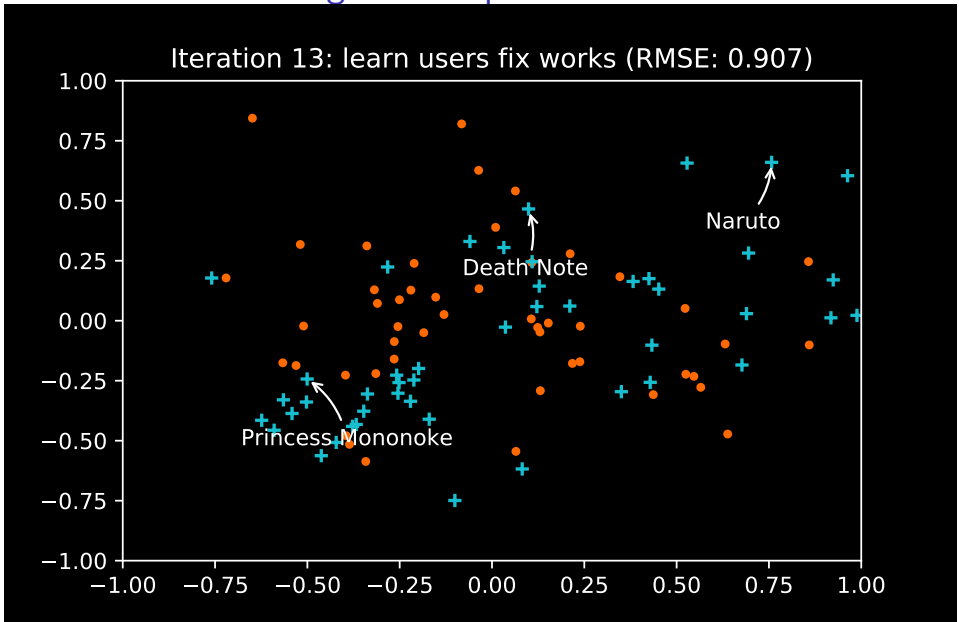


Illustration of Alternating Least Squares

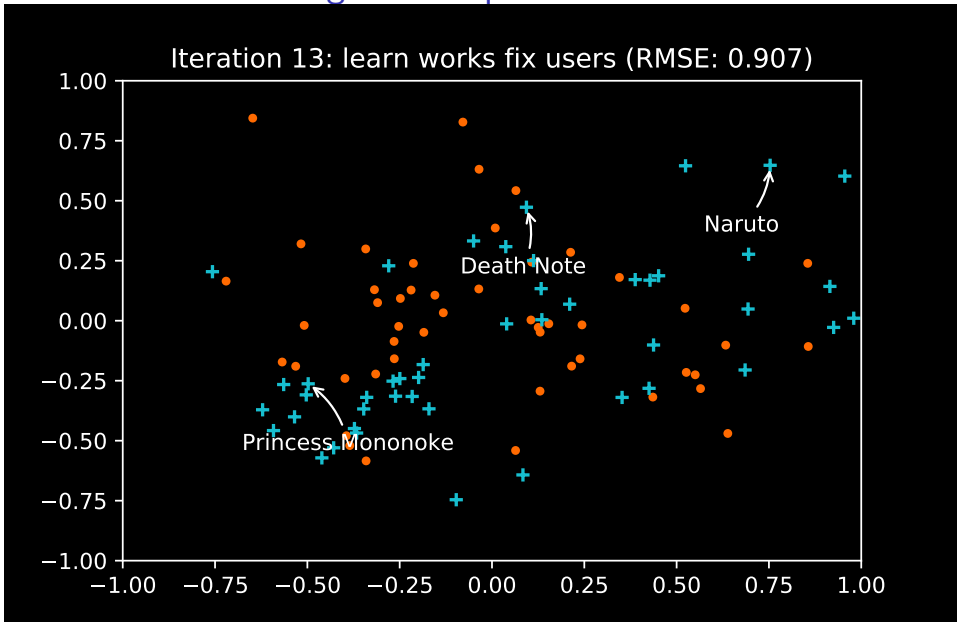


Illustration of Alternating Least Squares

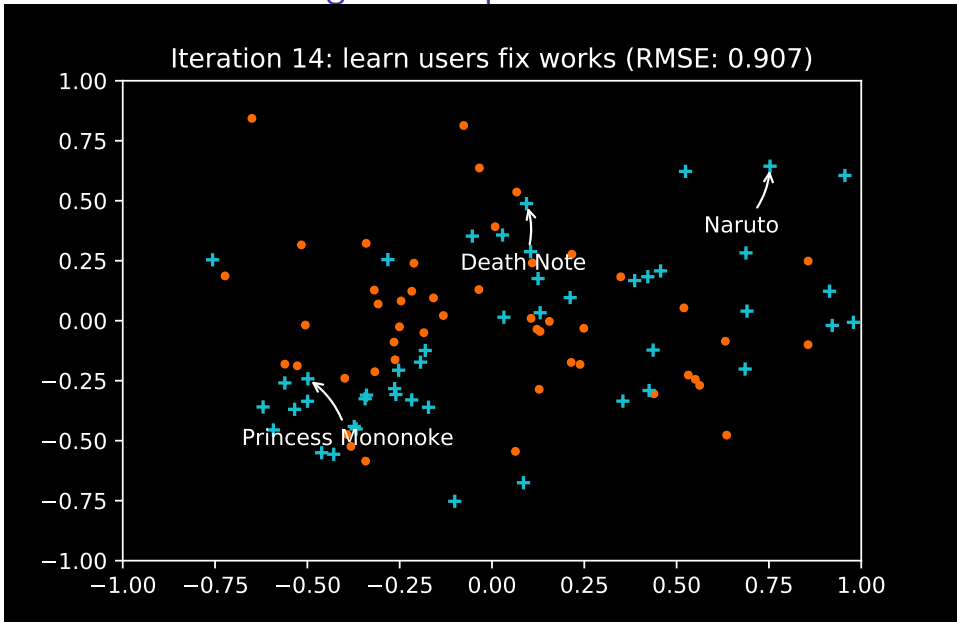


Illustration of Alternating Least Squares

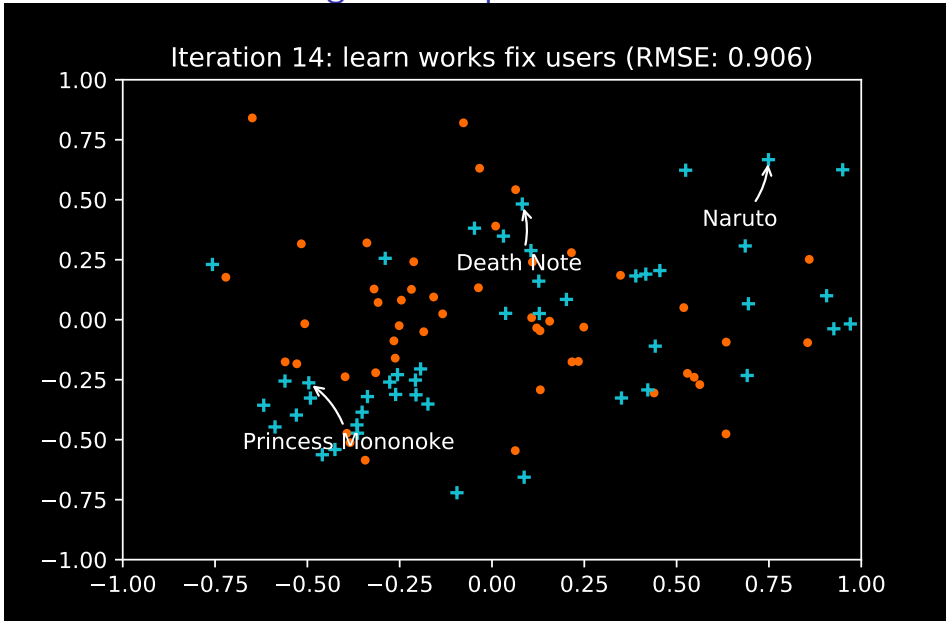


Illustration of Alternating Least Squares

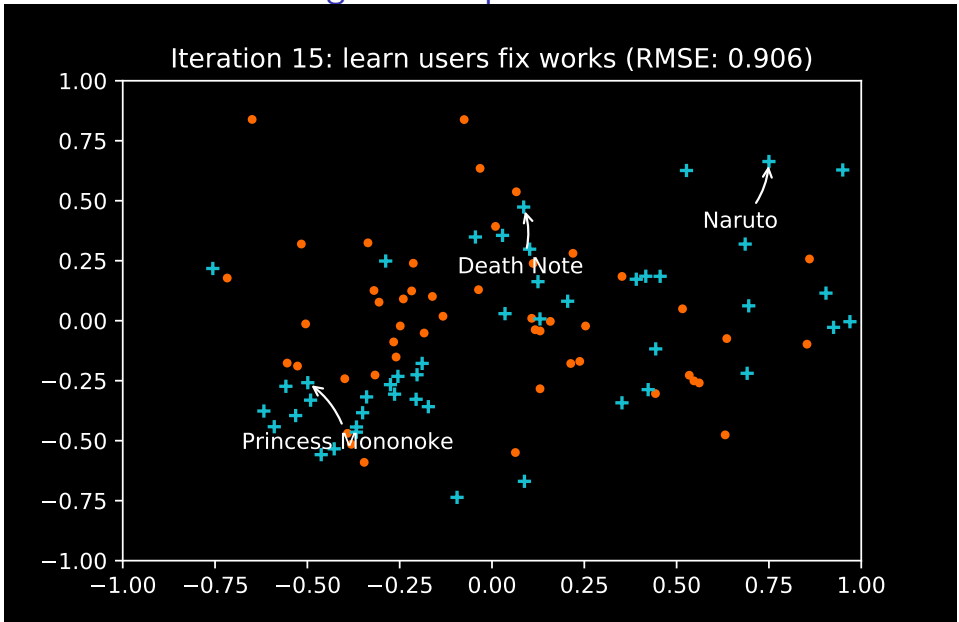


Illustration of Alternating Least Squares

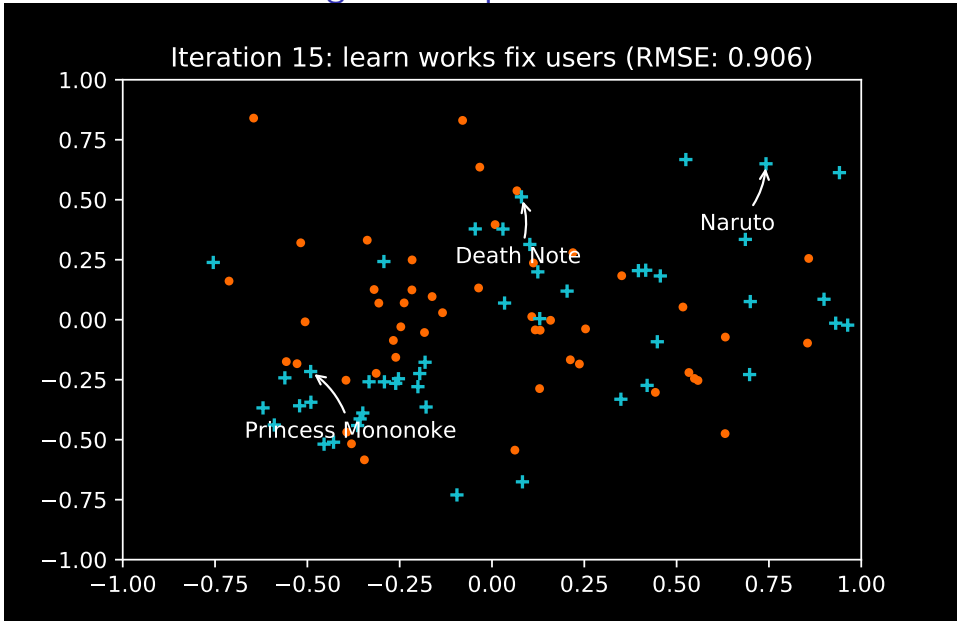


Illustration of Alternating Least Squares

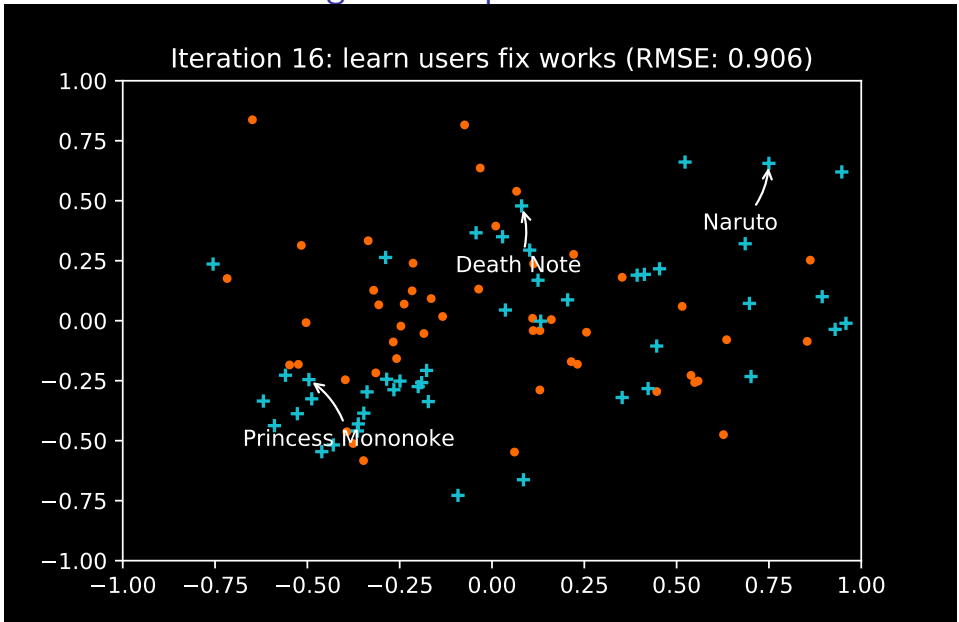


Illustration of Alternating Least Squares

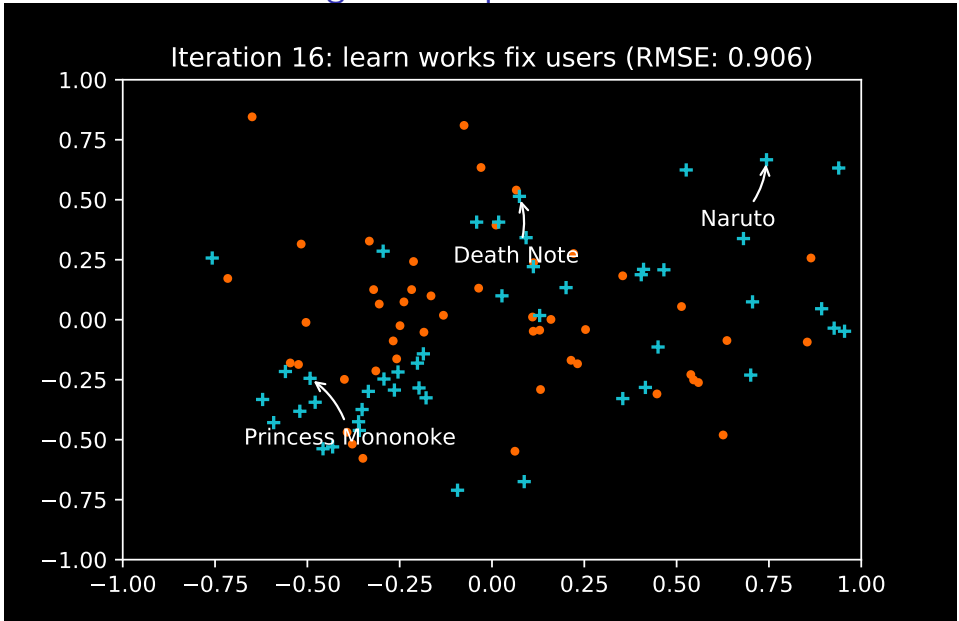


Illustration of Alternating Least Squares

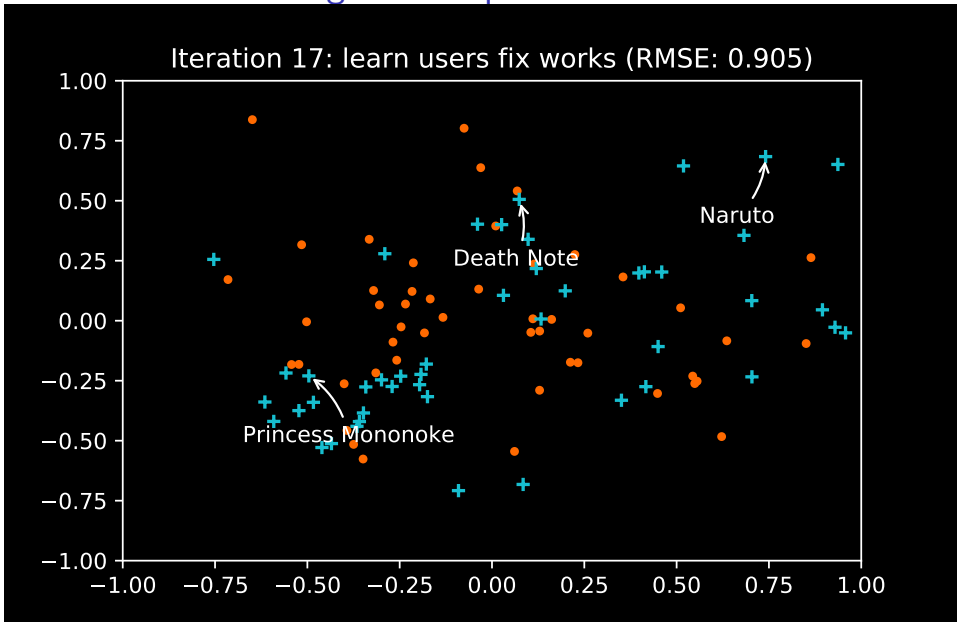


Illustration of Alternating Least Squares

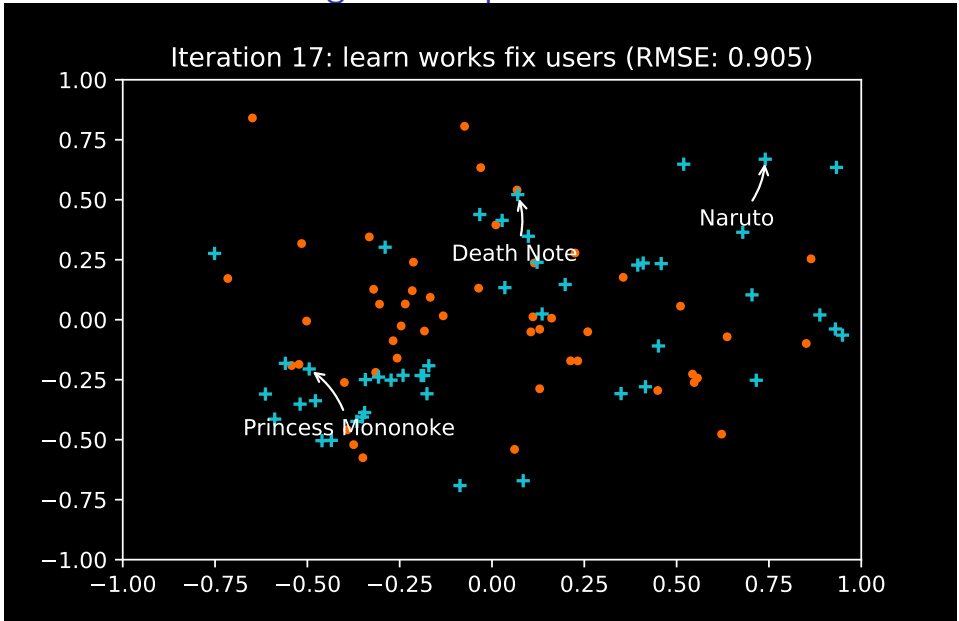


Illustration of Alternating Least Squares

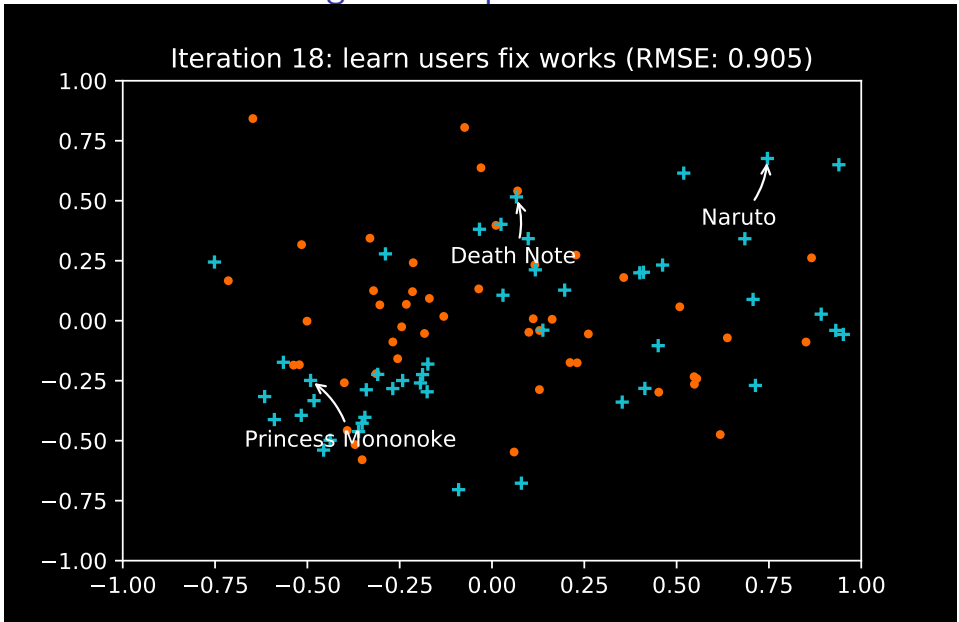


Illustration of Alternating Least Squares

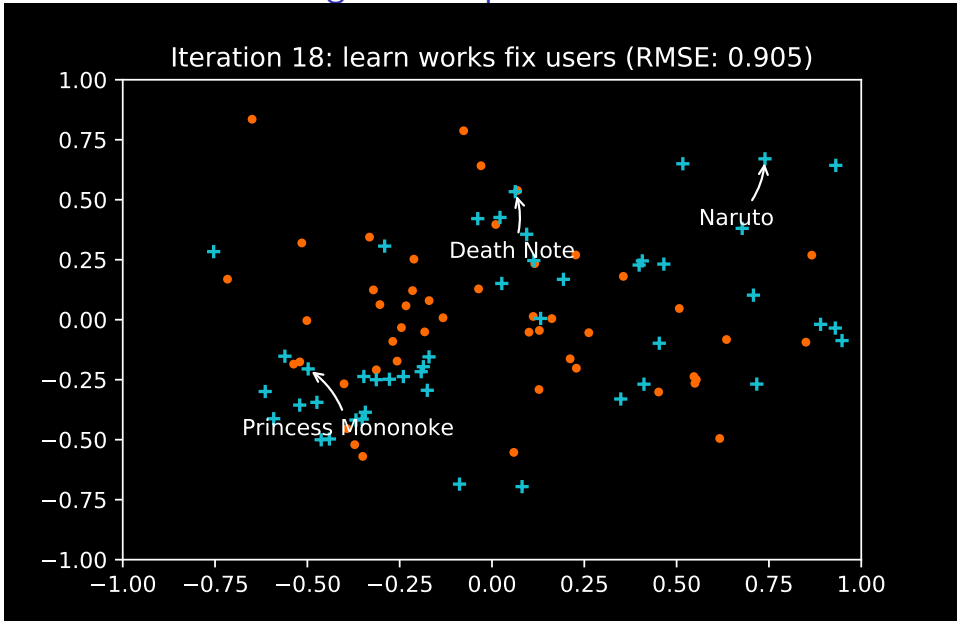


Illustration of Alternating Least Squares

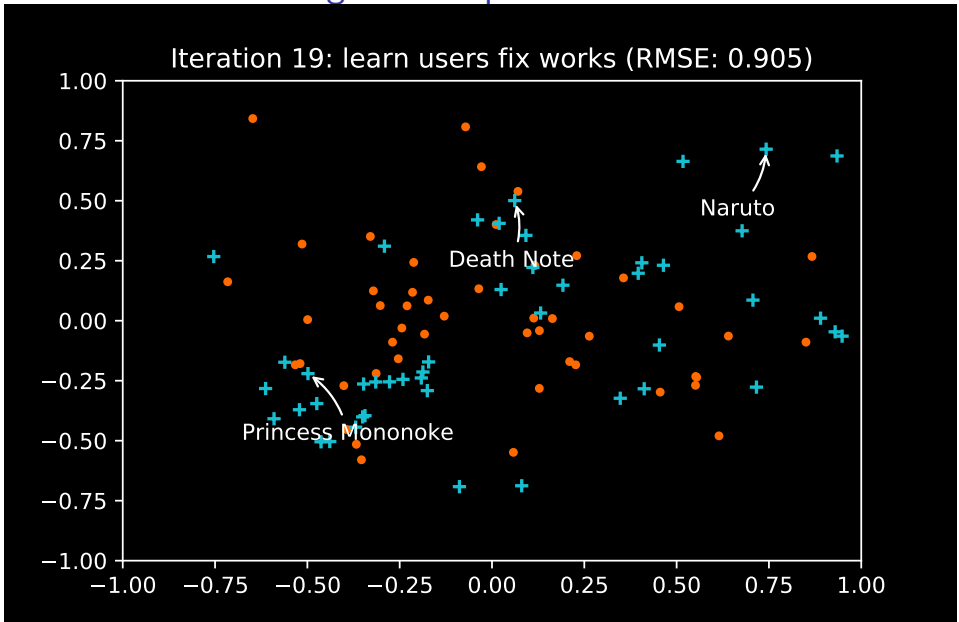
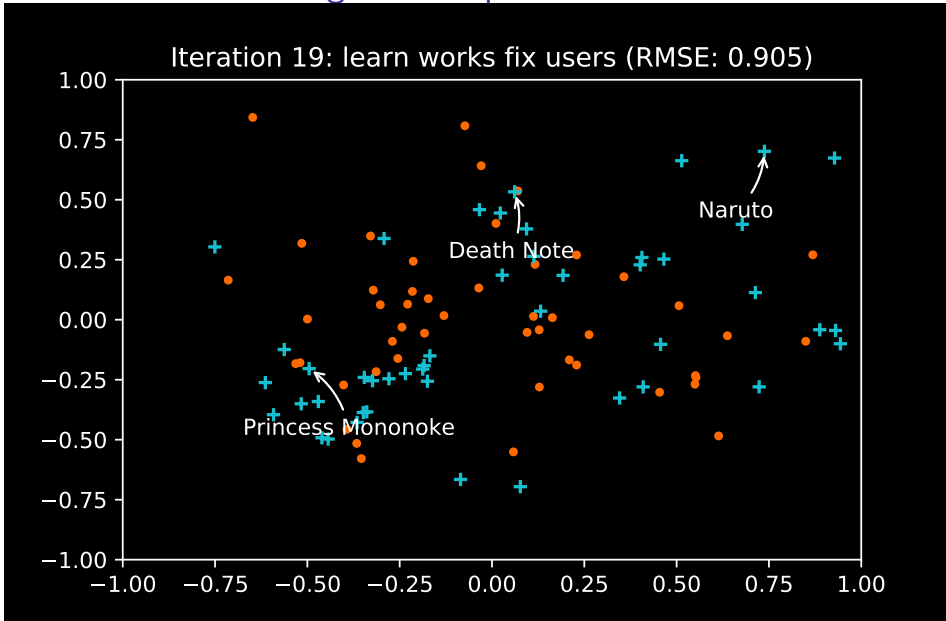
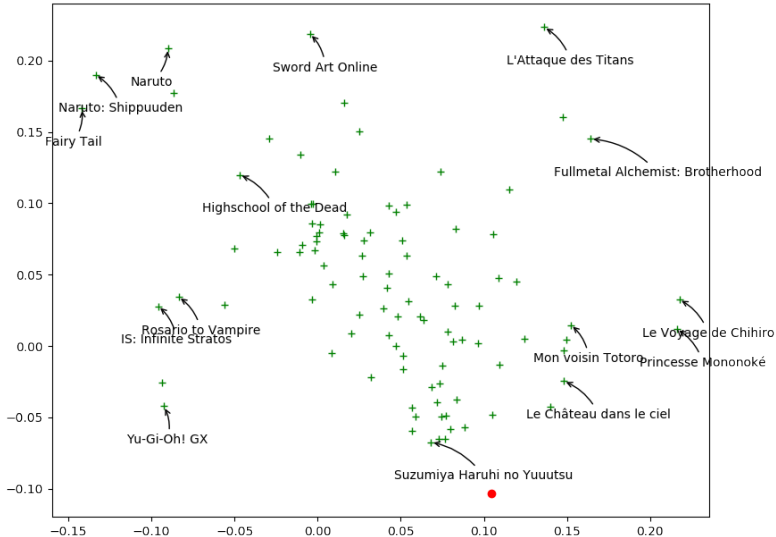


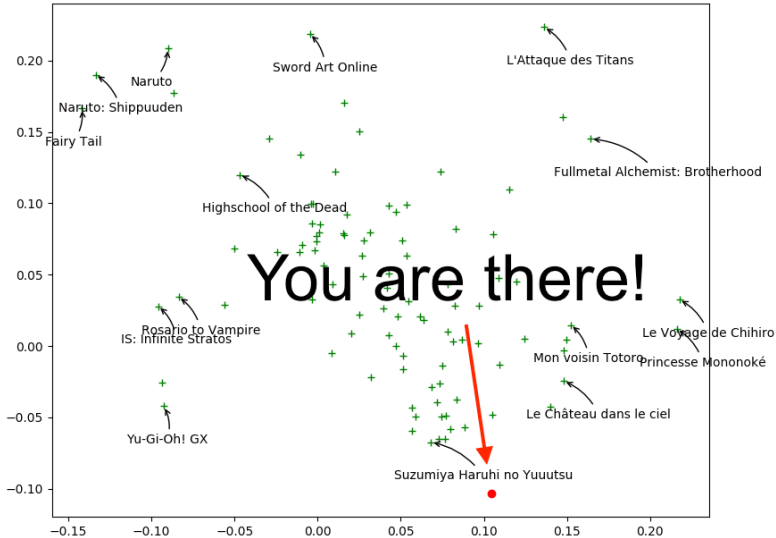
Illustration of Alternating Least Squares



Close $\boldsymbol{v}_j$ mean similar movies



u_i will like movies in its direction



word2vec: the Skip-Gram model

We want, given a word $w \sim P(W)$, to predict its context (i.e. the words around it).

Oh, it's a classification problem! With 60,000 classes.

$$P(c|w) = \text{softmax}(\boldsymbol{u}_w^T \boldsymbol{v}_c) \propto \exp(\boldsymbol{u}_w^T \boldsymbol{v}_c)$$

Cross-entropy would look like:

$$\mathcal{L} = \log \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_c) + \sum_{\substack{i=1 \\ c_i \neq c}}^{60000} \log(1 - \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_{c_i}))$$

So instead let's just sample k words from vocabulary proportionally to their occurrence:

$$\mathcal{L} = \sum_{w,c} \log \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_c) + \sum_{\substack{i=1 \\ c_i \sim P(W)}}^{k} \log(1 - \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_{c_i}))$$

Neural Word Embedding as Implicit Matrix Factorization

Noise contrastive estimation

$$\mathcal{L} = \sum_{w,c} \log \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_c) + \sum_{\substack{i=1 \\ c_i \sim P(W)}}^{k} \log(1 - \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_{c_i}))$$

Negative sampling (similar objective)

$$\mathcal{L} = \sum_{w,c} \log \sigma(\boldsymbol{u}_w^T \boldsymbol{v}_c) + \sum_{\substack{i=1 \\ c_i \sim P(W)}}^{k} \log \sigma(-\boldsymbol{u}_w^T \boldsymbol{v}_{c_i}))$$

This objective is equivalent to factorizing the pointwise mutual information matrix:

$$\log \frac{\#(w,c)N}{\#w\#c} - \log k = \boldsymbol{u}_w^T \boldsymbol{v}_c \text{ where } N \text{ is the number of pairs.}$$

Omer Levy and Yoav Goldberg. "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf

Practical datasets

A toy dataset of matrix completion

You are given some entries of $M = UV$ where the shapes are $(10, 3) \cdot (3, 5)$
The goal is to recover the other entries

Movielens

100,000 ratings (1–5 stars) of 600 users on 9,000 movies.
https://files.grouplens.org/datasets/movielens/ml-latest-small.zip

The Adventures of Sherlock Holmes

By Arthur Conan Doyle
https://github.com/theeluwin/pytorch-sgns

[1] Guillaume Lample et al. "Word translation without parallel data". In: *International Conference on Learning Representations*. 2018. URL: https://openreview.net/forum?id=H196sainb.

[2] Omer Levy and Yoav Goldberg. "Neural Word Embedding as Implicit Matrix Factorization". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/feab05aa91085b7a8012516bc3533958-Paper.pdf.

[3] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and their Compositionality". In: *Advances in Neural Information Processing Systems*. Ed. by C.J. Burges et al. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper/2013/file/9aa42b31882ec039965f3c4923ce901b-Paper.pdf.