Statistics and distance based features

Some data for CTR task

	User_id	Page_id	Ad_price	Ad_position
0	4	6	165.977125	Bottom_right
1	4	6	34.5395640	Bottom_right
2	4	6	29.1963786	Bottom_left
3	4	6	79.4373729	Bottom_left
4	4	6	290.534595	Bottom_right
5	4	6	314.412660	Bottom_right
6	4	6	138.9007639	Bottom_right
7	4	6	107.4711914	Bottom_right
8	4	6	242.1089786	Bottom_left
9	4	7	27.16719836	Bottom_left
10	4	7	413.5421978	Bottom_right

Useful features

	User_id	Page_id	Ad_price	Ad_position	Max_price	min_price	Min_price _position
0	4	6	95.874252	Bottom_right	474.63772	73.711548	Bottom_left
1	4	6	215.751007	Bottom_right	474.63772	73.711548	Bottom_left
2	4	6	474.637726	Bottom_left	474.63772	73.711548	Bottom_left
3	4	6	73.711548	Bottom_left	474.63772	73.711548	Bottom_left
4	4	6	79.288841	Bottom_right	474.63772	73.711548	Bottom_left
5	4	6	271.391785	Bottom_right	474.63772	73.711548	Bottom_left
6	4	6	296.529053	Bottom_right	474.63772	73.711548	Bottom_left
7	4	6	96.030029	Bottom_right	474.63772	73.711548	Bottom_left
8	4	6	130.175064	Bottom_left	474.63772	73.711548	Bottom_left
9	4	7	35.465202	Bottom_left	121.54219	35.465202	Bottom_left
10	4	7	121.542191	Bottom_right	121.54219	35.465202	Bottom_left

Useful features: implementation

In [22]:
```
gb = df.groupby(['user_id','page_id'],as_index=False).agg(
    {'ad_price':{'max_price':np.max, 'min_price':np.min}})
gb.columns = ['user_id','page_id','min_price','max_price']
gb
```

Out[22]:

	user_id	page_id	min_price	max_price
0	4	6	73.711548	474.637726
1	4	7	35.465202	121.542191

In [23]:
```
df = pd.merge(df,gb,how='left',on=['user_id','page_id'])
```

More features

- How many pages user visited
- Standard deviation of prices
- Most visited page
- Many, many more

Neighbors

- Explicit group is not needed
- More flexible
- Much harder to implement

Neighbors

- Number of houses in 500m, 1000m,..
- Average price per square meter in 500m, 1000m,..
- Number of schools/supermarkets/parking lots in 500m, 1000m,..
- Distance to closest subway station

Neighbors

- Explicit group is not needed
- More flexible
- Much harder to implement

KNN features in Springleaf

- Mean encode all the variables
- For every point, find 2000 nearest neighbors using

Bray-Curtis metric

$$\sum |u_i - v_i| / \sum |u_i + v_i|$$

- Calculate various features from those 2000 neighbors

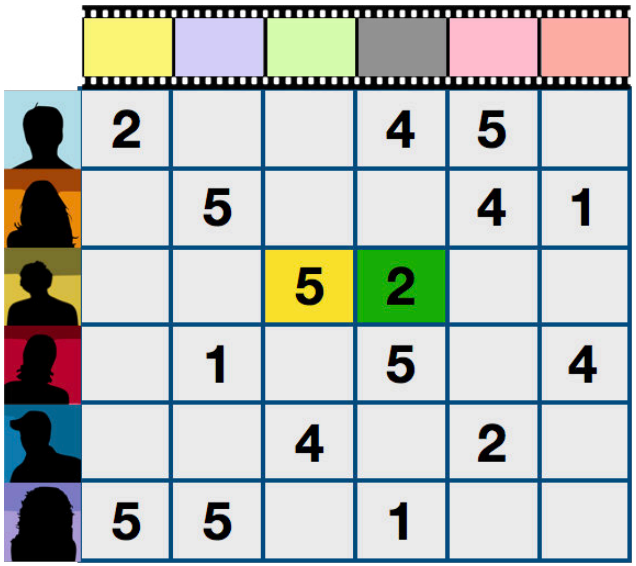
KNN features in Springleaf

- Mean target of nearest 5,10,15,500, 2000 neighbors
- Mean distance to 10 closest neighbors
- Mean distance to 10 closest neighbors with target 1
- Mean distance to 10 closest neighbors with target 0

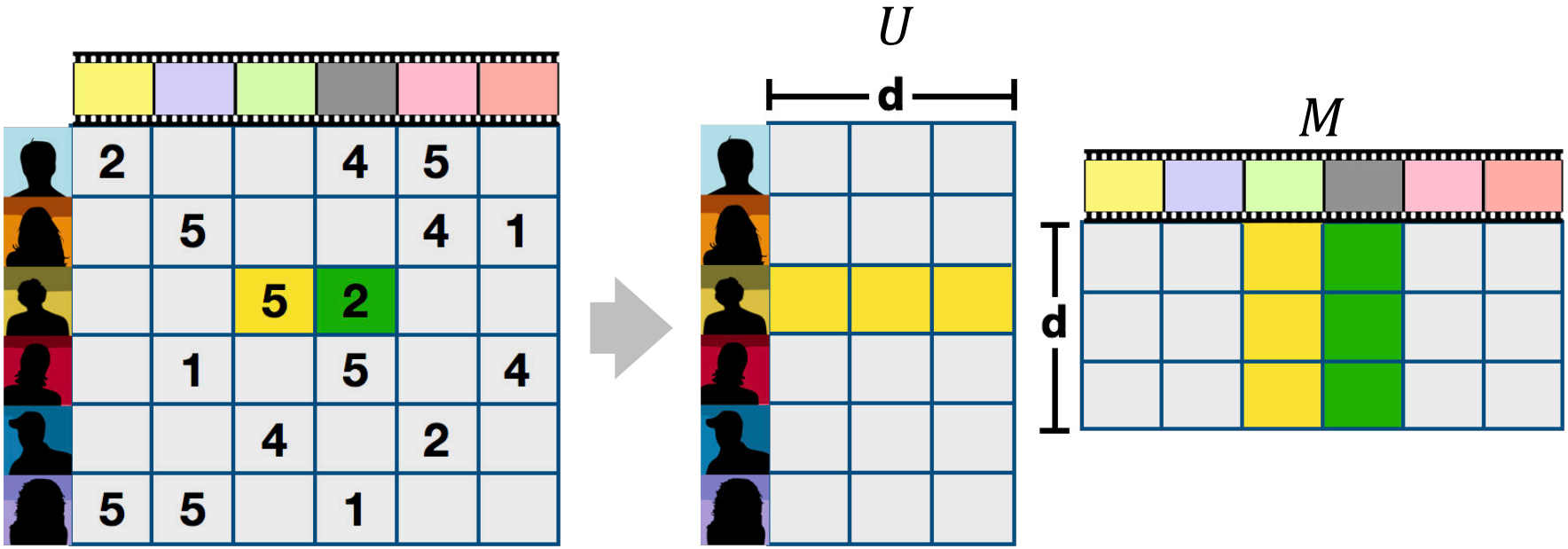
Thank you

Matrix Factorizations for Feature Extraction

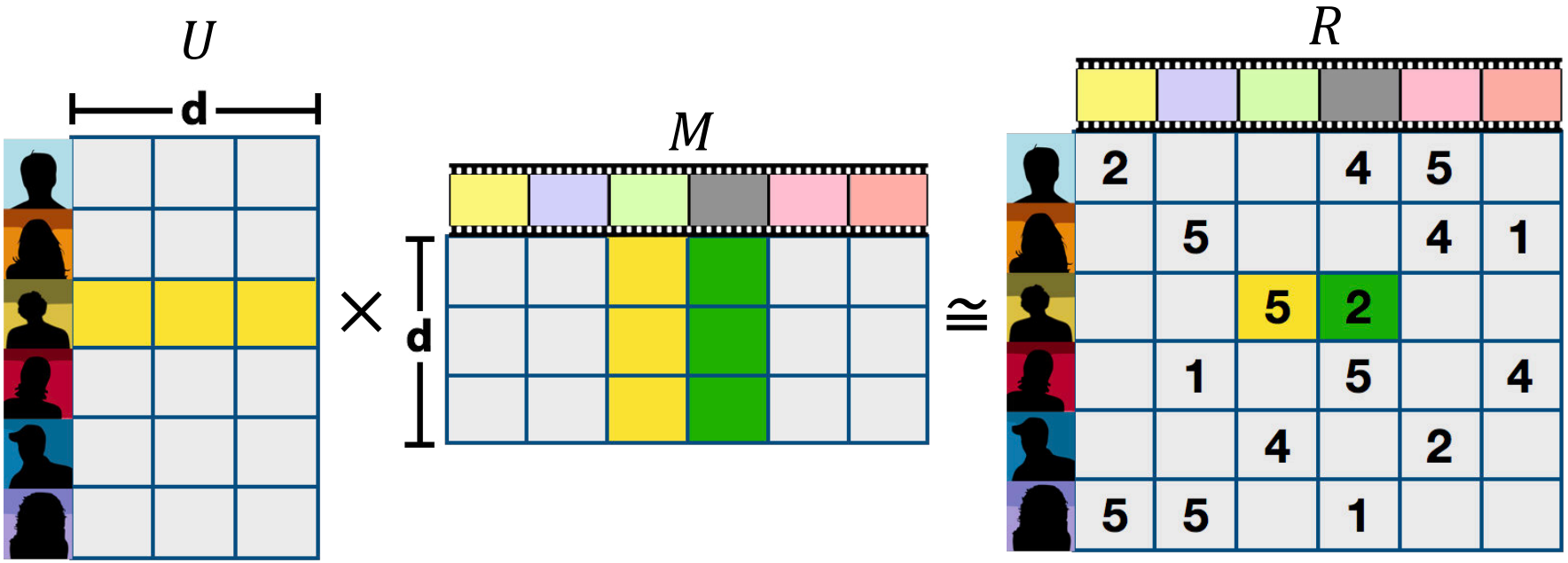
Example



Example



What is Matrix Factorization?



Documents\words example

	cat	is	dog	animal	nature
Cat is animal	1	0	0	1	0
Dog is animal	0	1	1	1	0
Cat is dog	1	1	1	0	0

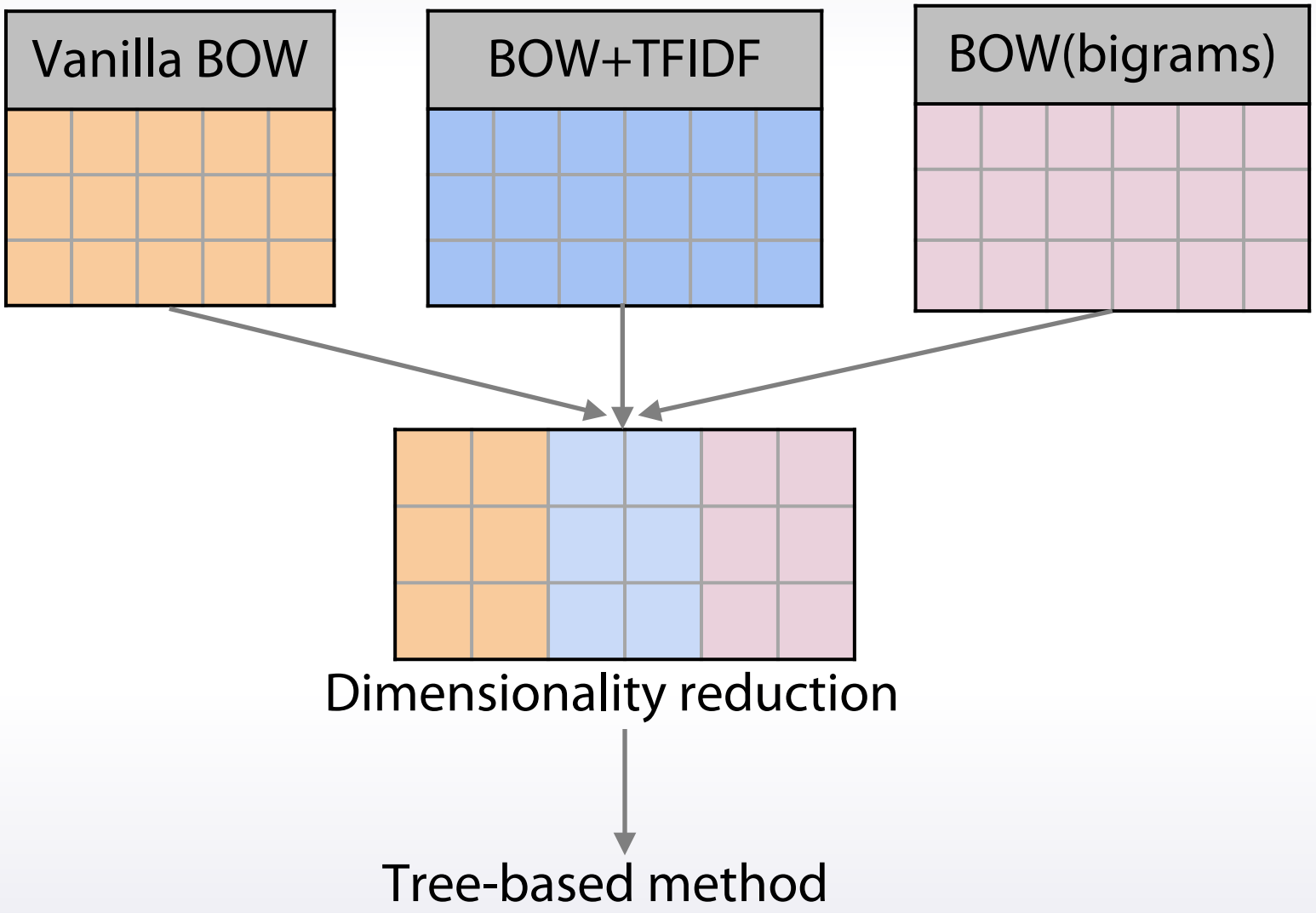
Documents\words example

	cat	is	dog	animal	nature
Cat is animal	1	0	0	1	0
Dog is animal	0	1	1	1	0
Cat is dog	1	1	1	0	0

Cat is animal			
Dog is animal			
Cat is dog			

cat			
is			
dog			
animal			
nature			

Example of feature fusion



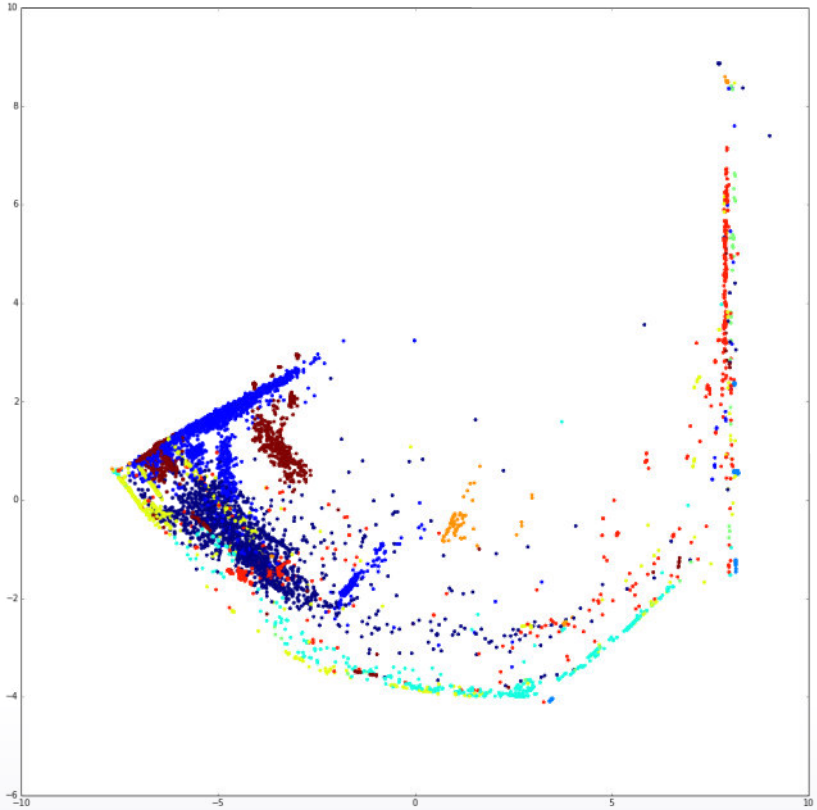
Notes about Matrix Factorization

- Can be apply only for some columns
- Can provide additional diversity
  - Good for ensembles
- It is a **lossy** transformation. Its' efficiency depends on:
  - Particular task
  - Number of latent factors
    - Usually 5-100

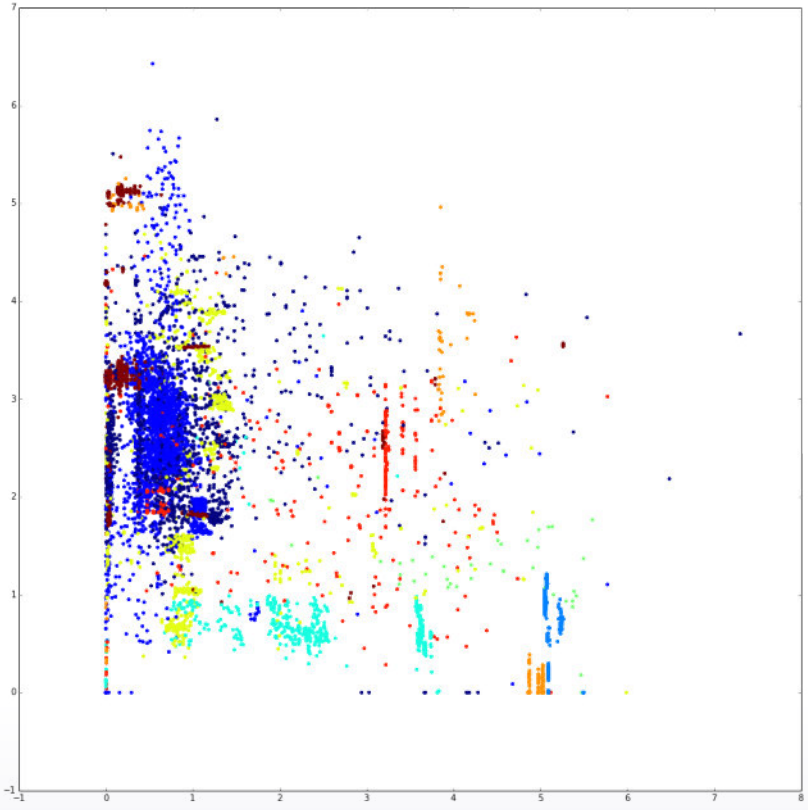
Implementation

- Several MF methods you can find in sklearn
- SVD and PCA
  - Standart tools for Matrix Factorization
- TruncatedSVD
  - Works with sparse matrices
- Non-negative Matrix Factorization (NMF)
  - Ensures that all latent factors are non-negative
  - Good for counts-like data

NMF for tree-based methods



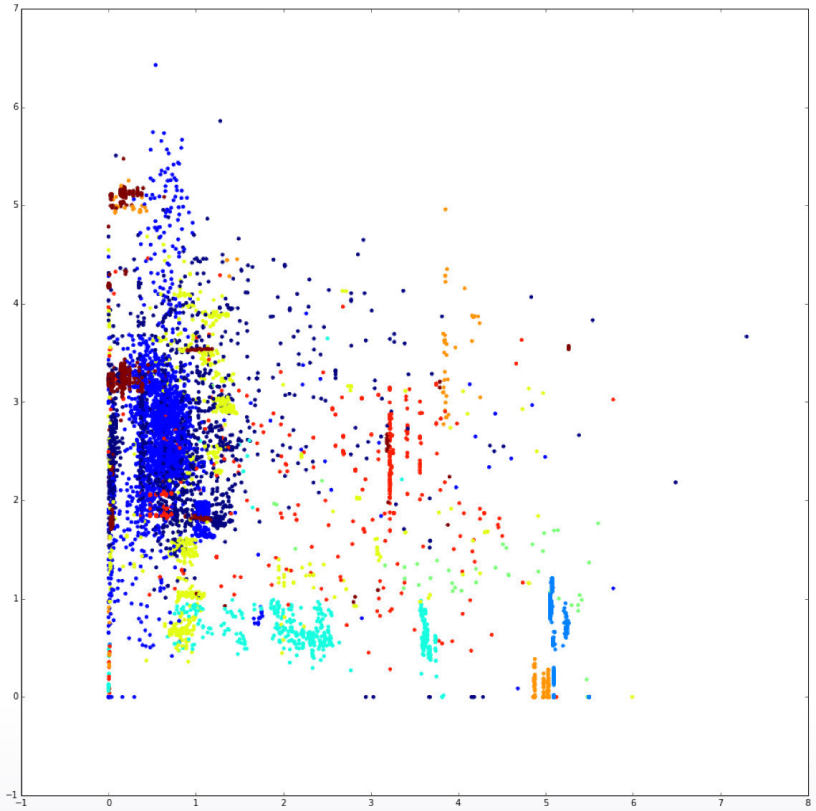
PCA



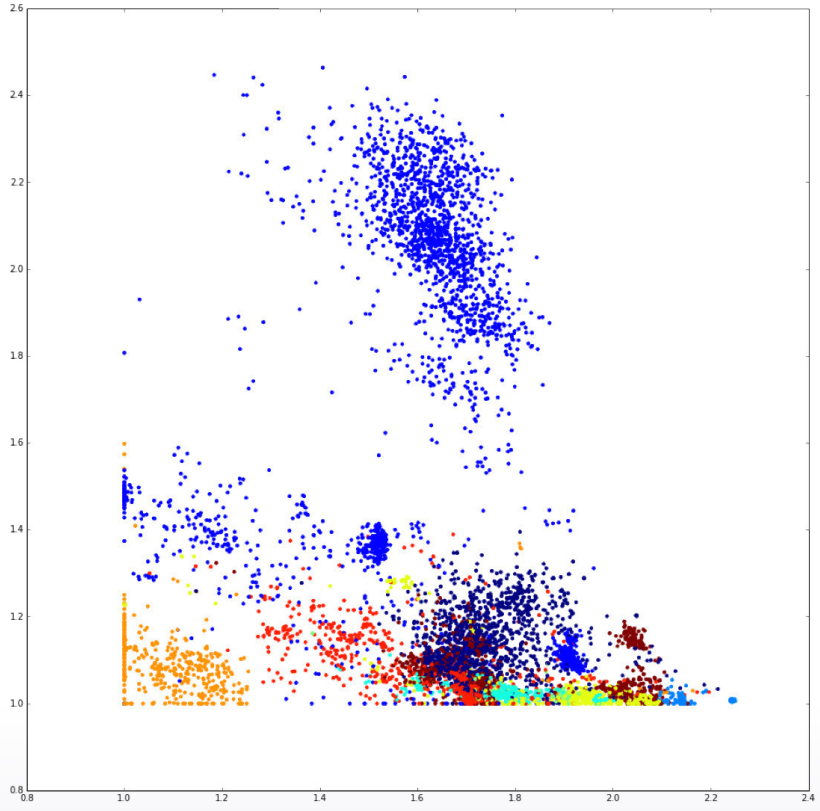
NMF

Notes about MF

Matrix factorization is similar in spirit to linear models.
You can use the same transformation tricks.



NMF(X)



NMF(log(X+1))

Gochas

Wrong way:

```
pca = PCA(n_components=5)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.fit_transform(X_test)
```

Right way:

```
X_all = np.concatenate([X_train,X_test])
pca.fit(X_all)
X_train_pca = pca.transform(X_train)
X_test_pca = pca.transform(X_test)
```

Conclusion

- Matrix Factorization is a very general approach for dimensionality reduction and feature extraction

Conclusion

- Matrix Factorization is a very general approach for dimensionality reduction and feature extraction
- It can be applied for transforming categorical features into real-valued

Conclusion

- Matrix Factorization is a very general approach for dimensionality reduction and feature extraction
- It can be applied for transforming categorical features into real-valued
- Many of tricks trick suitable for linear models can be useful for MF

Feature interactions

Example: banner selection

...	category_ad	category_site	...	is_clicked
...	auto_part	game_news	...	0
...	music_tickets	music_news	..	1
...	mobile_phones	auto_blog	...	0

Example: banner selection

...	category_ad	category_site	...	is_clicked
...	auto_part	game_news	...	0
...	music_tickets	music_news	..	1
...	mobile_phones	auto_blog	...	0

...	ad_site	...	is_clicked
...	auto_part|game_news	...	0
...	music_tikets|music_news	..	1
...	mobile_phones|auto_blog	...	0

Example of interactions

f1	f2
A	X
B	Y
B	Z
A	Z

Example of interactions

f1	f2
A	X
B	Y
B	Z
A	Z

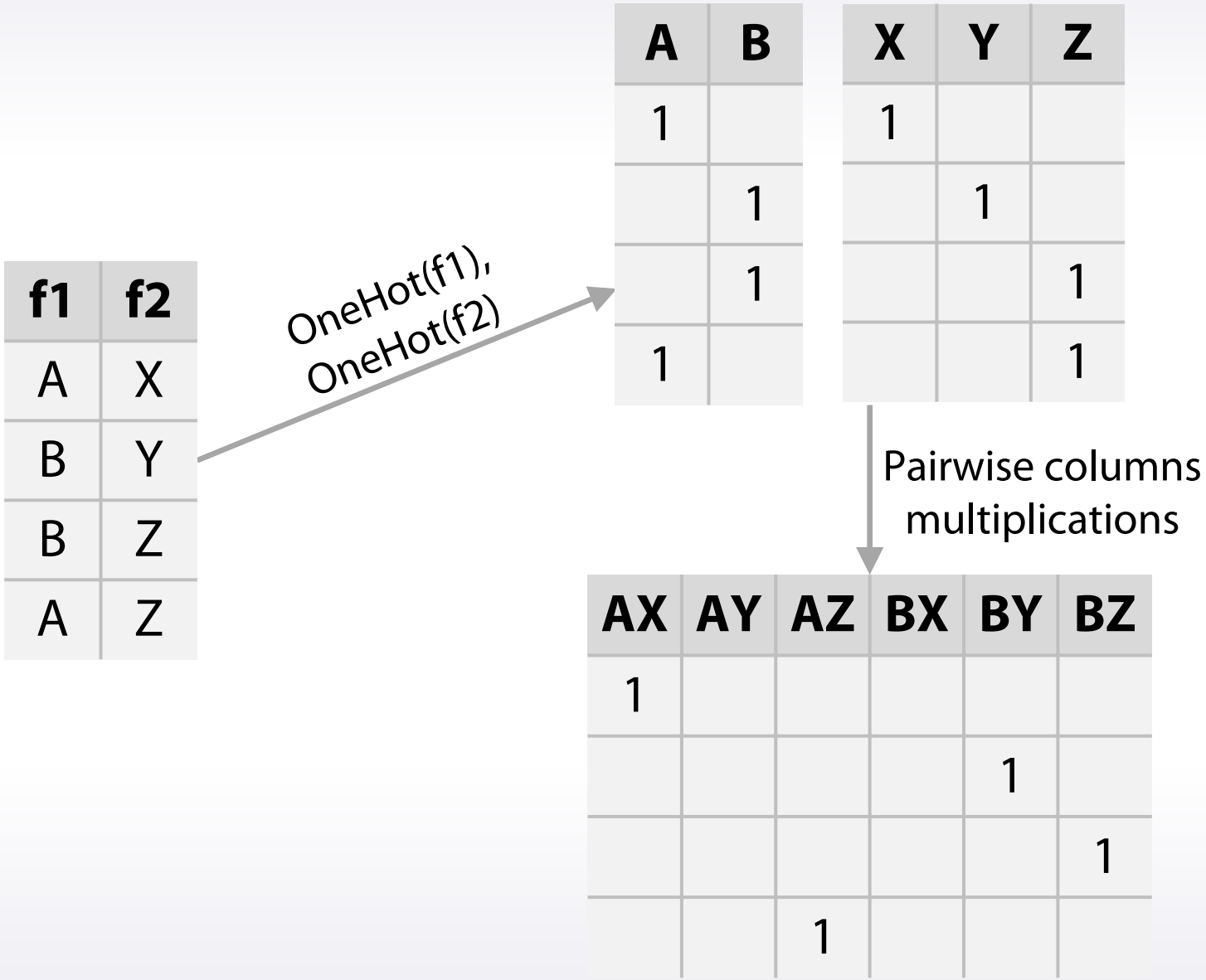
Join

f_join
A|X
B|Y
B|Z
A|Z

OneHot (f_join)

A|X	B|Y	B|Z	A|Z
1			
	1		
		1	
			1

Example of interactions



Example of interactions

f1	f2
1.2	0.0
3.4	0.1
5.6	1.0
7.8	-1.0

Mul →

f_join
0.0
0.34
5.6
-7.8

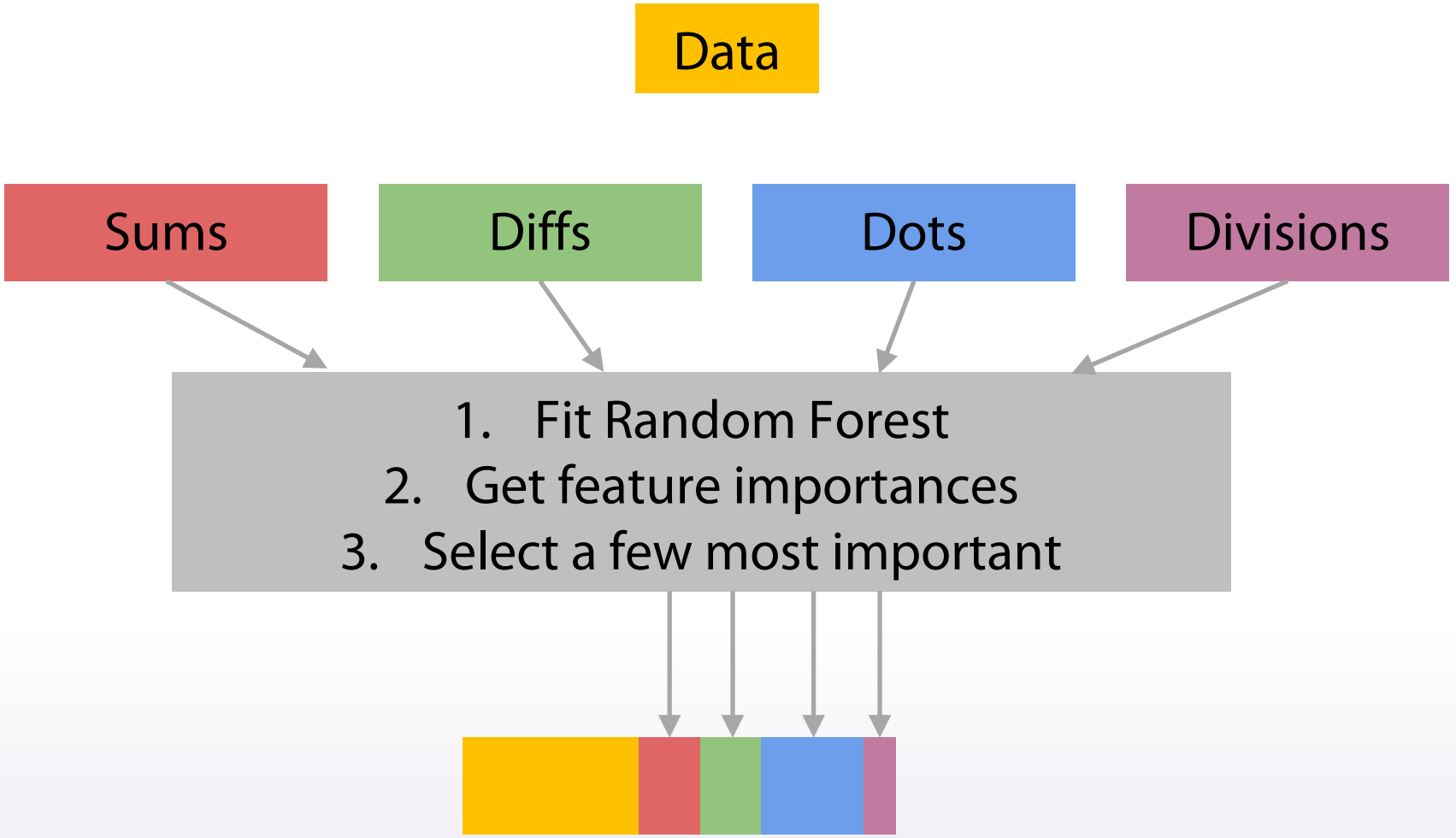
Frequent operations for feature interaction

- Multiplication
- Sum
- Diff
- Division

Practical Notes

- We have a lot of possible interactions – N*N for N features.
    a. Even more if use several types in interactions
- Need to reduce its' number
    a. Dimensionality reduction
    b. Feature selection

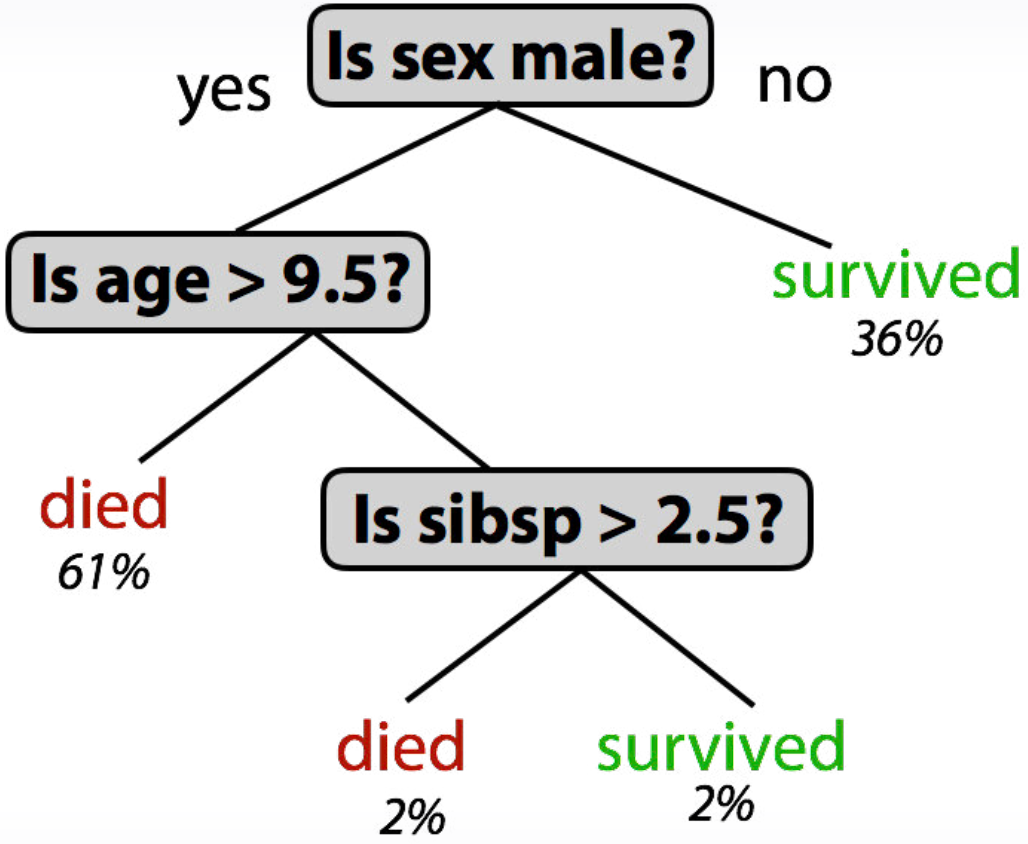
Example of interaction generation pipeline



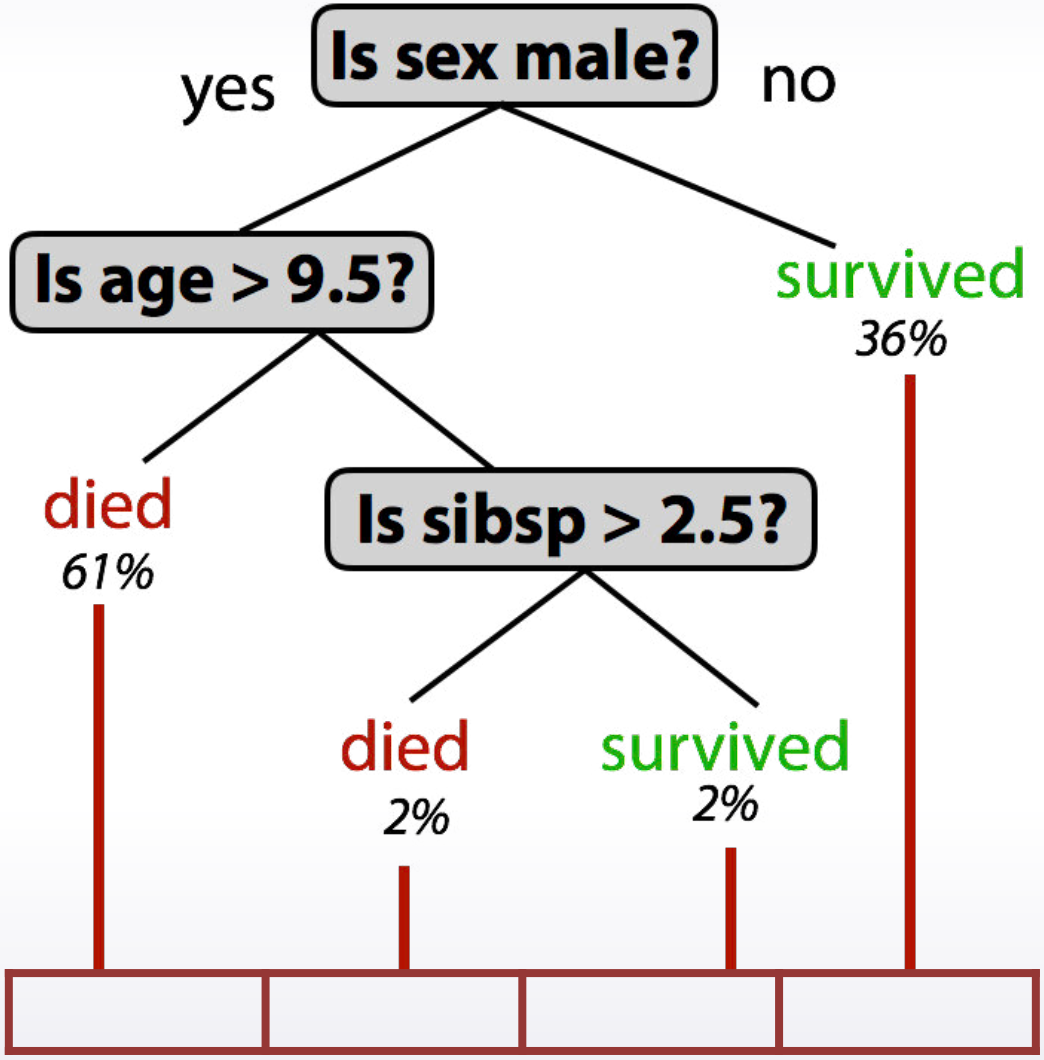
Interactions' order

- We looked at 2nd order interactions.
- Such approach can be generalized for higher orders.
- It is hard to do generation and selection automatically.
- Manual building of high-order interactions is some kind of art.

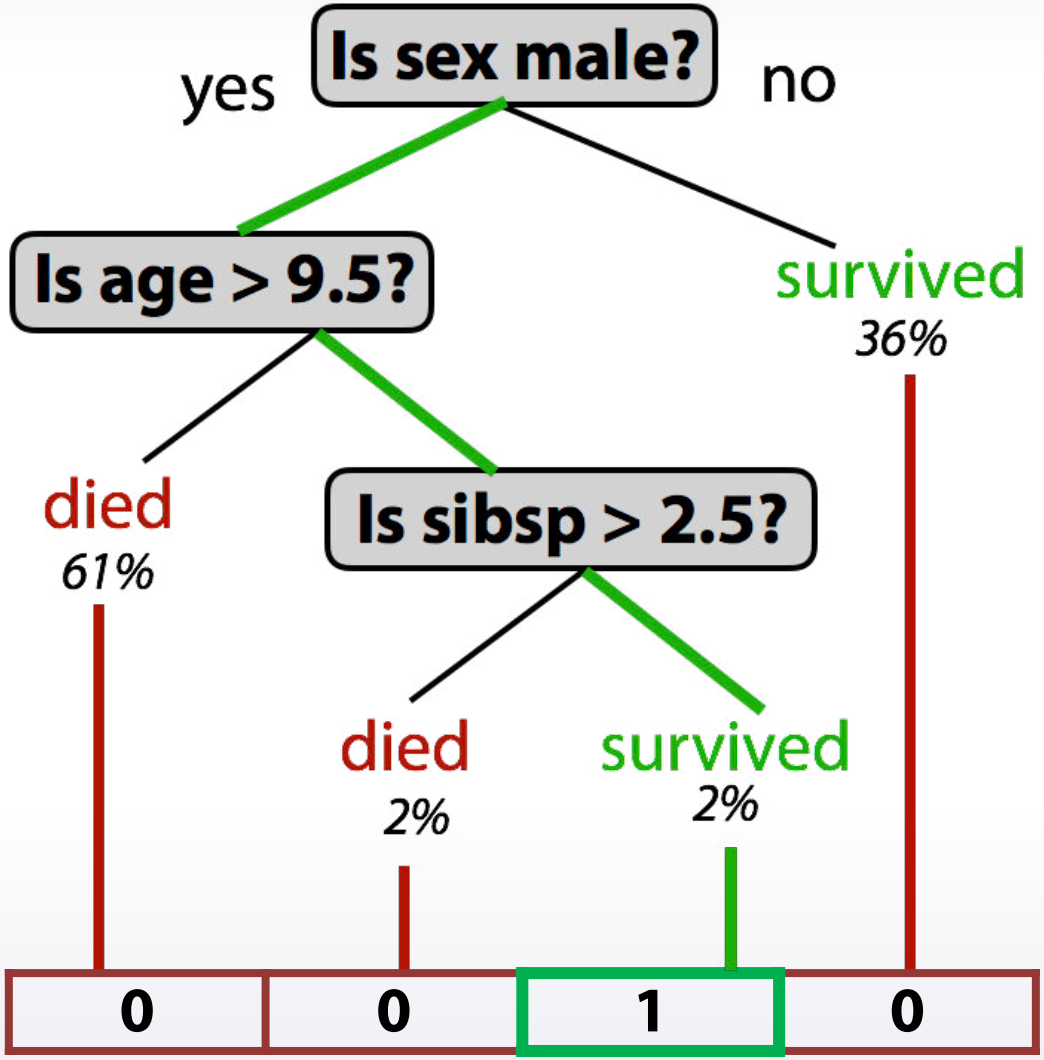
Extract features from DT



Extract features from DT



Extract features from DT



How to use it

In sklearn:

```
tree_model.apply()
```

In xgboost:

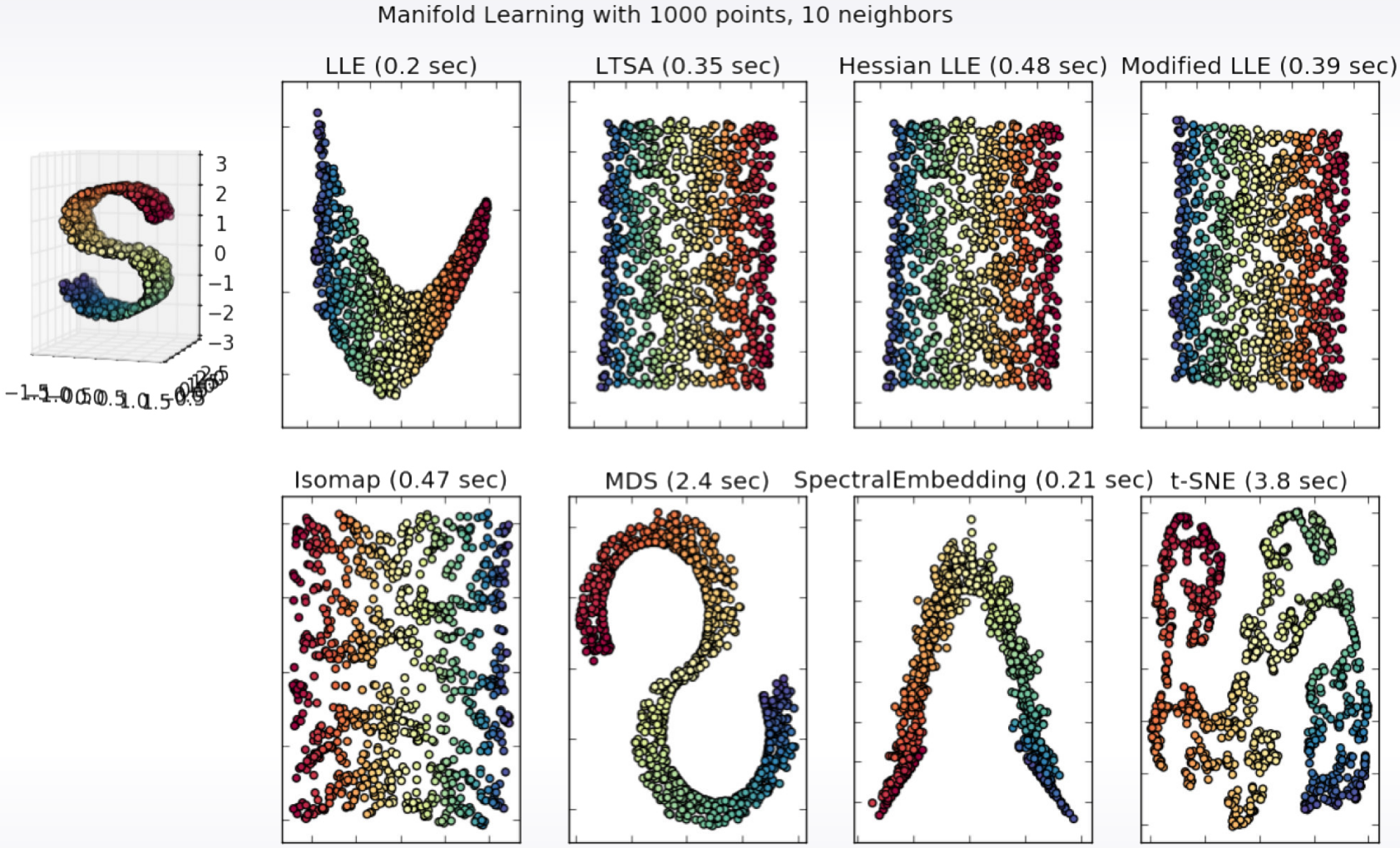
```
booster.predict(pred_leaf=True)
```

Conclusion

- We looked at ways to build an interaction of categorical attributes
- Extended this approach to real-valued features
- Learn how to extract features via decision trees

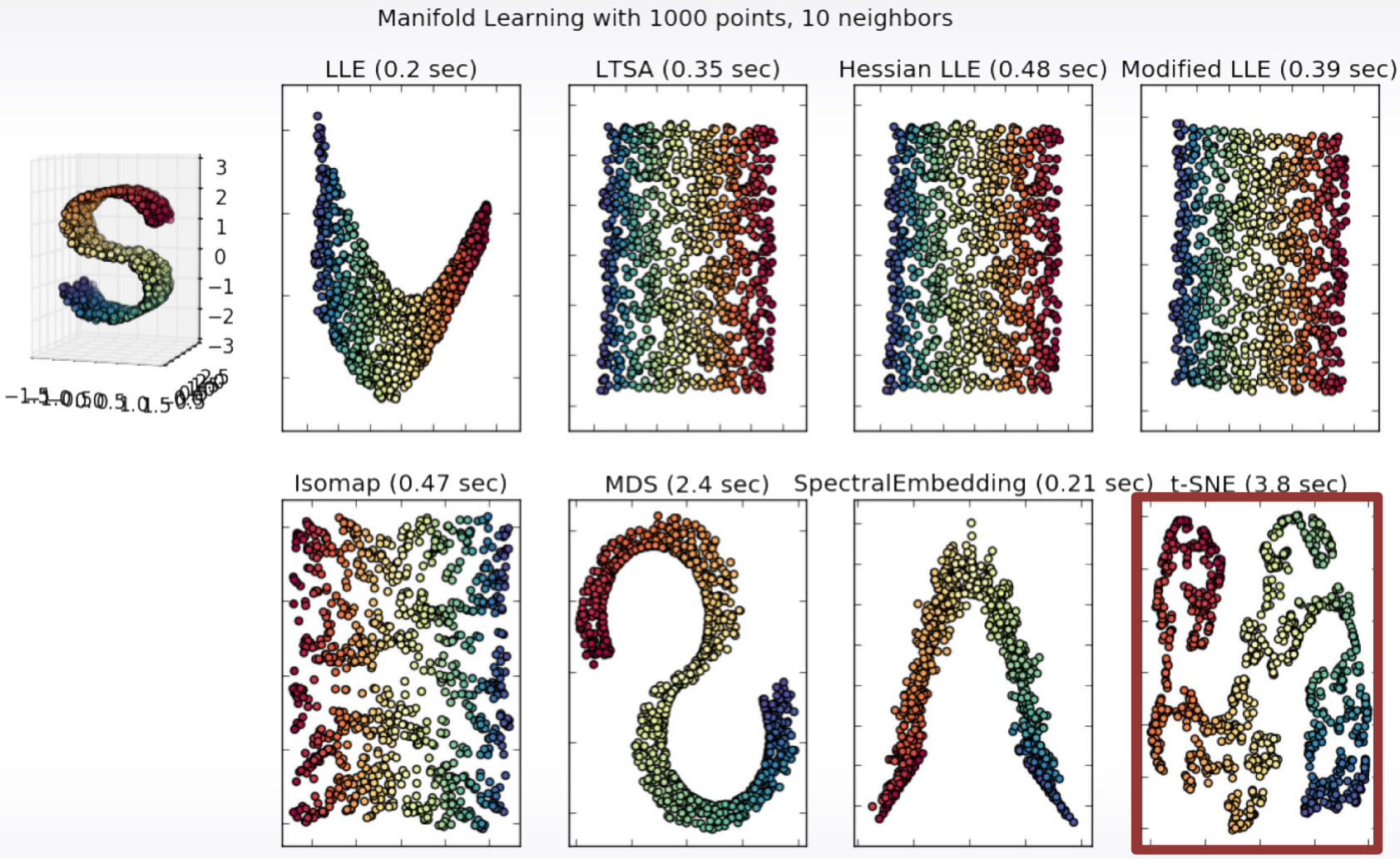
tSNE

Manifold learning methods



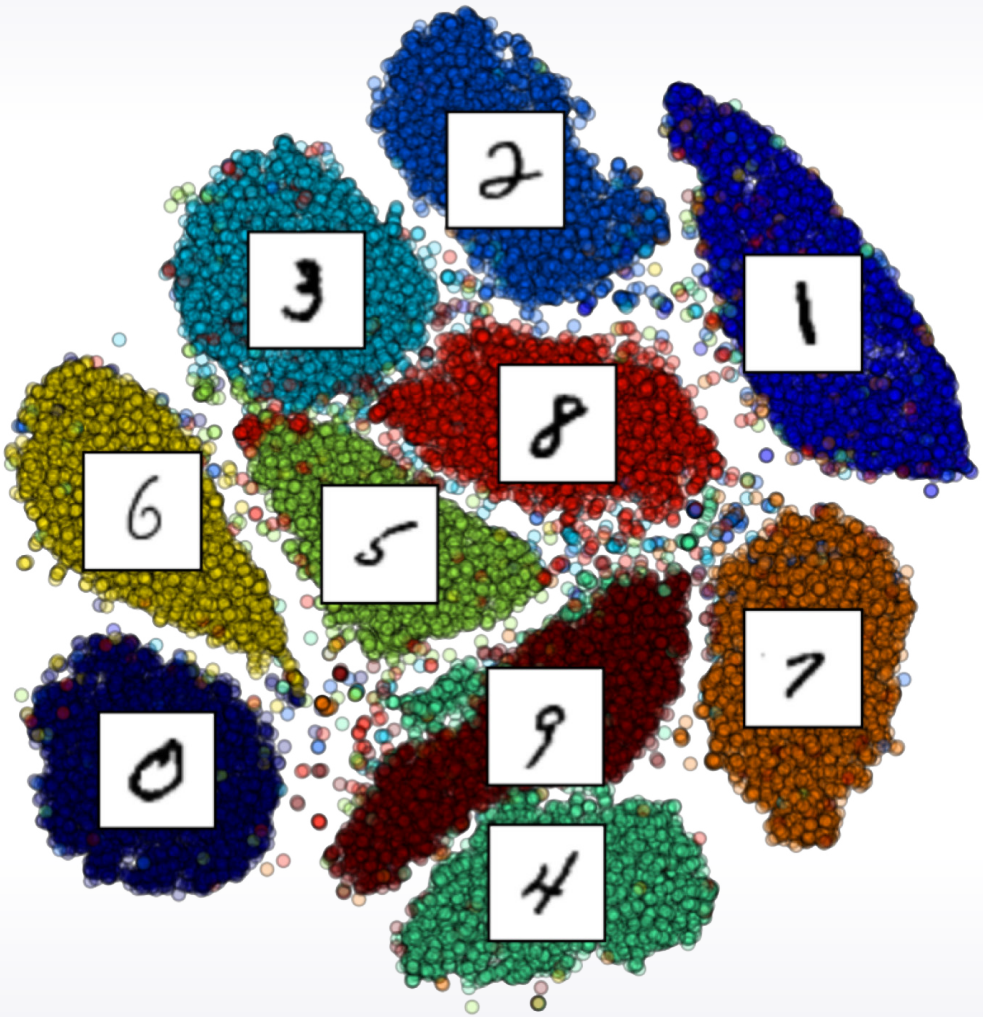
Comparison of Manifold Learning methods, http://scikitlearn.org/stable/auto_examples/manifold/plot_compare_methods.html#sphx-glr-auto-examples-manifold-plot-compare-methods-py

Manifold learning methods

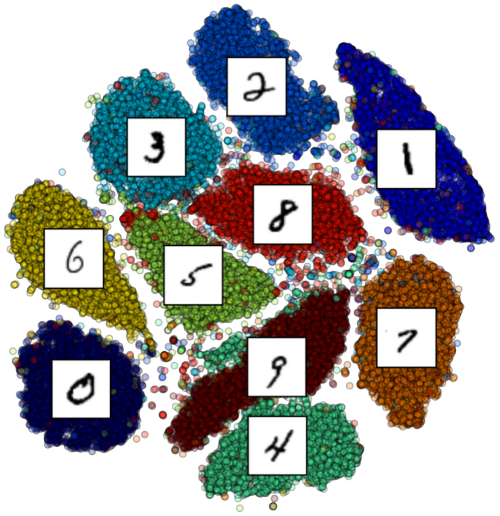


Comparison of Manifold Learning methods, http://scikitlearn.org/stable/auto_examples/manifold/plot_compare_methods.html#sphx-glr-auto-examples-manifold-plot-compare-methods-py

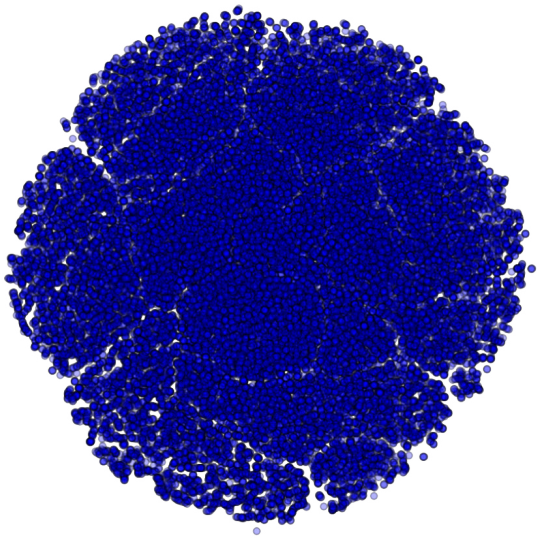
Example: tSNE on MNIST



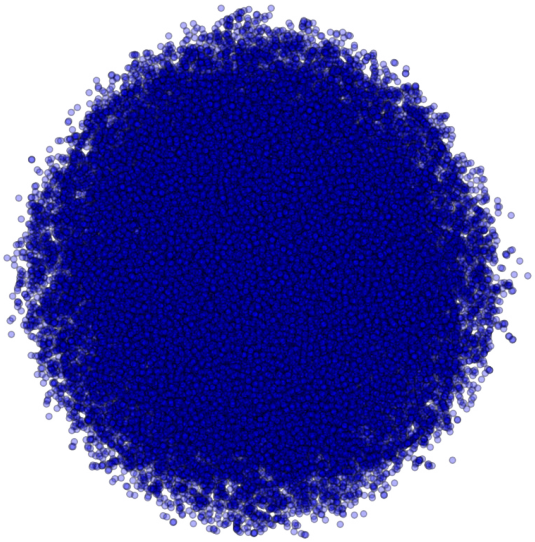
Interpretation of tSNE



MNIST

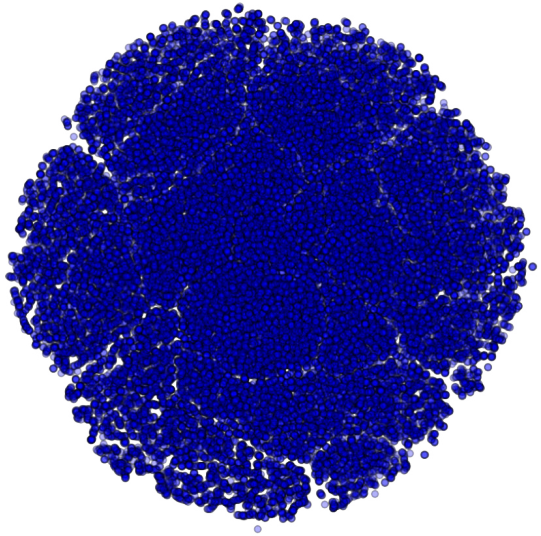


MNIST

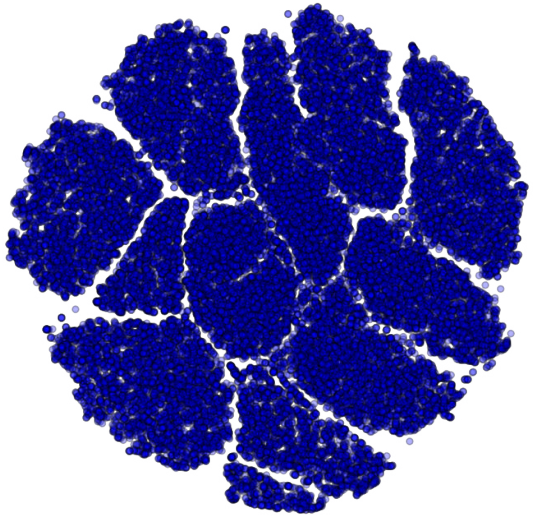


Random Values

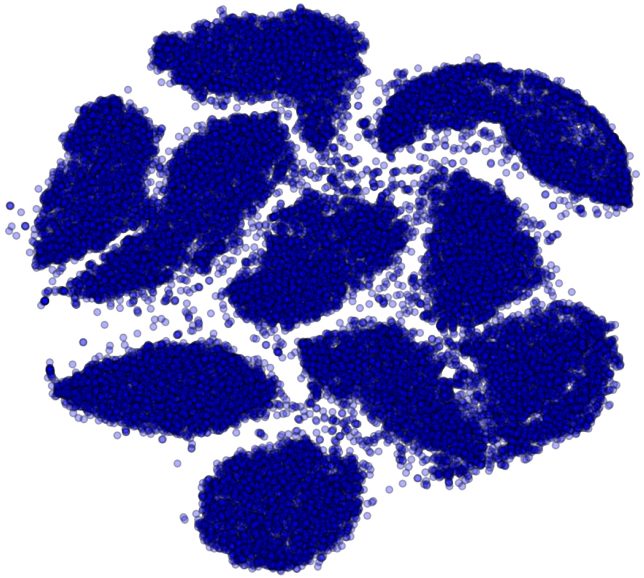
MNIST: different perplexities



Perplexity=3

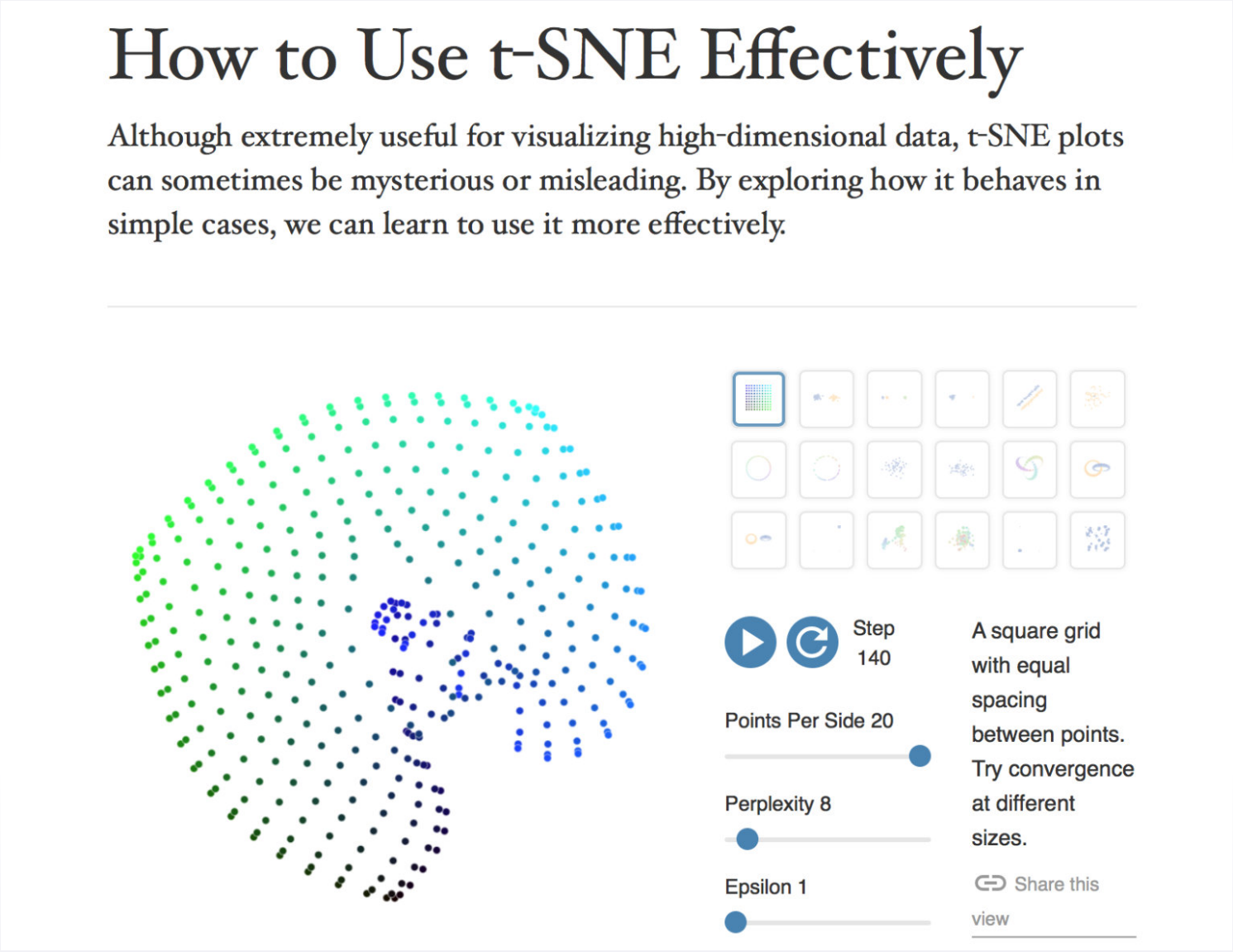


Perplexity=10



Perplexity=150

Interactive Example



Martin Wattenberg, https://distill.pub/2016/misread-tsne/

Practical Notes

- Result heavily depends on hyperparameters (perplexity)
  - Good practice is to use several projections with different perplexities (5-100)
- Due to stochastic nature, tSNE provides different projections even for the same data\hyperparams
  - Train and test should be projected together
- tSNE runs for a long time with a big number of features
  - it is common to do dimensionality reduction before projection.

Practical Notes

- Implementation of tSNE can be found in sklearn library.
- But personally I prefer you use stand-alone implementation python package tsne due to its' faster speed.

Conclusion

- tSNE is a great tool for visualization
- It can be used as feature as well
- Be careful with interpretation of results
- Try different perplexities