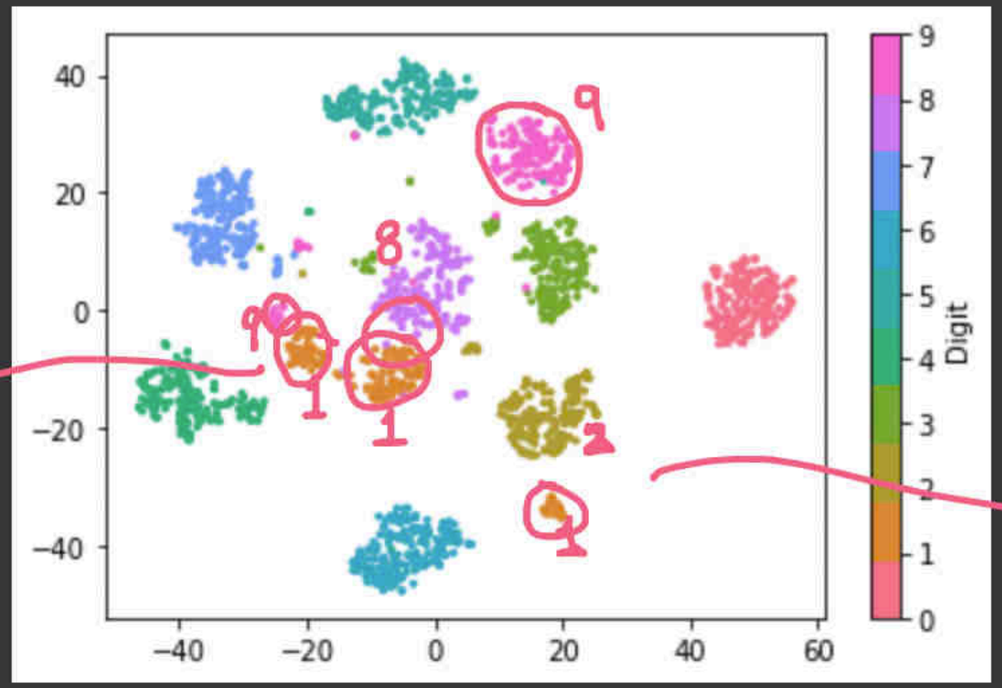
Topics

1. tSNE → practical aspects
2. UMAP → Math + practical aspects

+ code

3. Recommender-Systems : Content-based
4. RecSys : Collaborative
5. RecSys : Matrix factorization

```
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



81 8x8 pixels

-Unclosed

$\mathcal{D} = \{x_i\}$

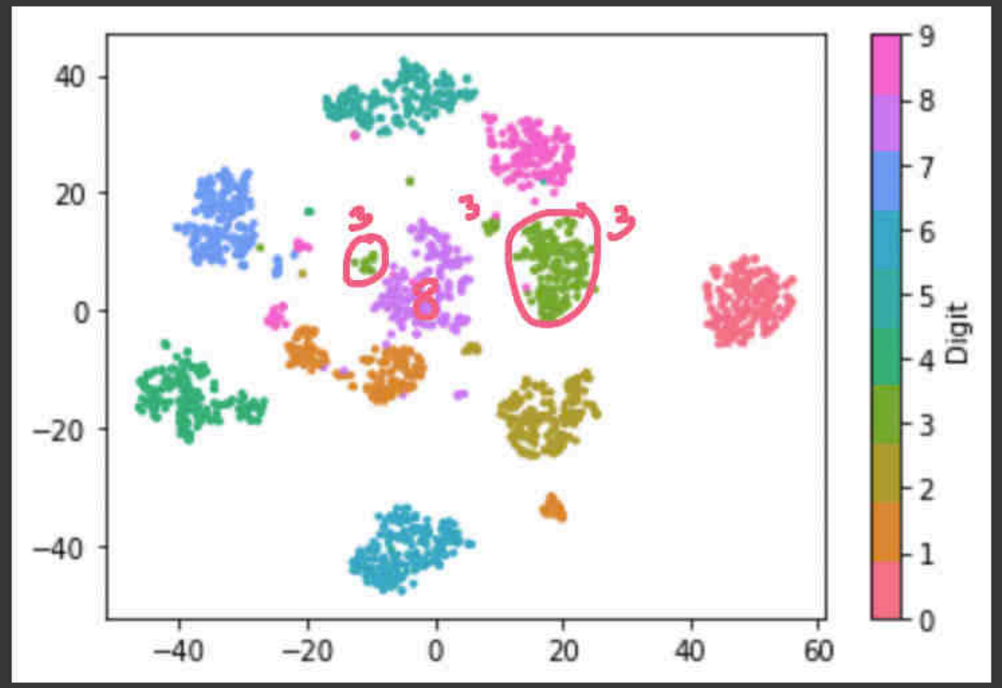
9 1

2 1

```
[ ] %%time
    Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

```
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```

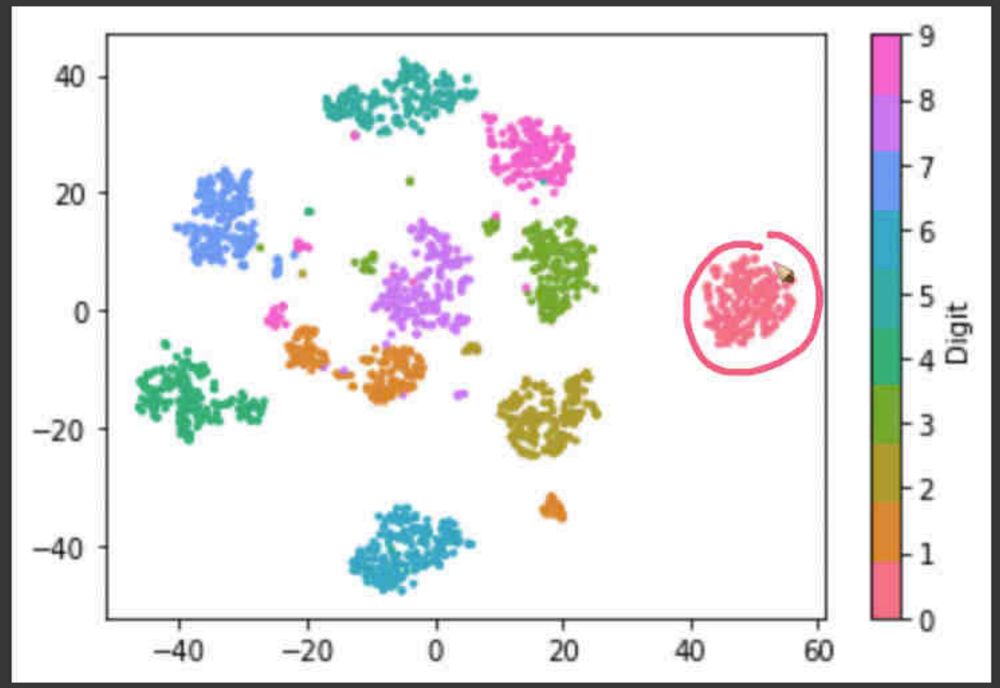


8 3

```
[ ] %%time
    Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

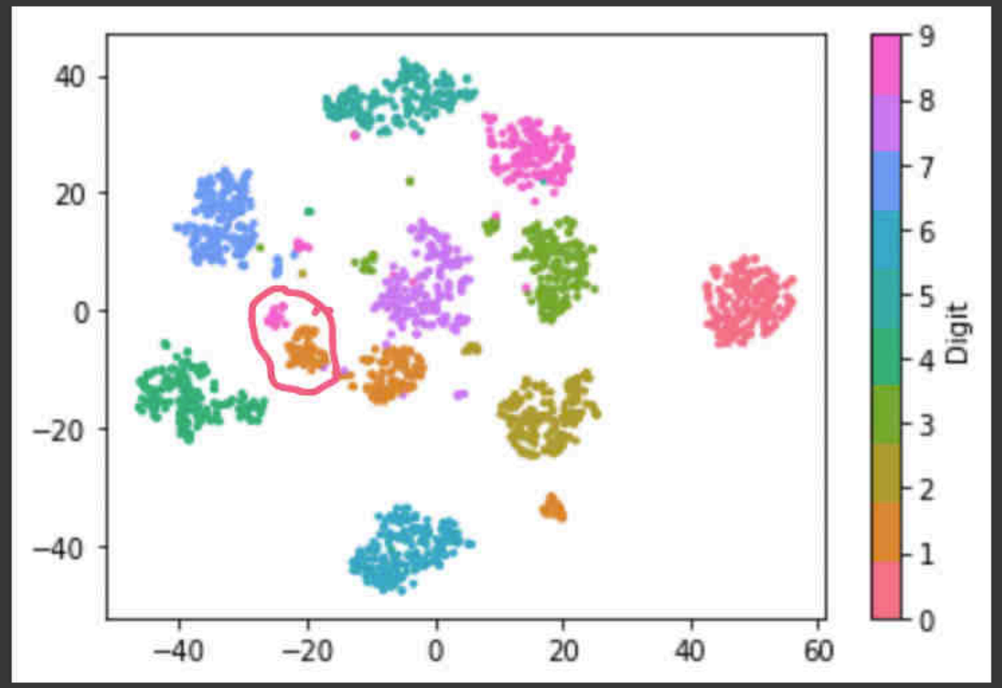
```
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```




```
[ ]  %%time
     Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

```
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



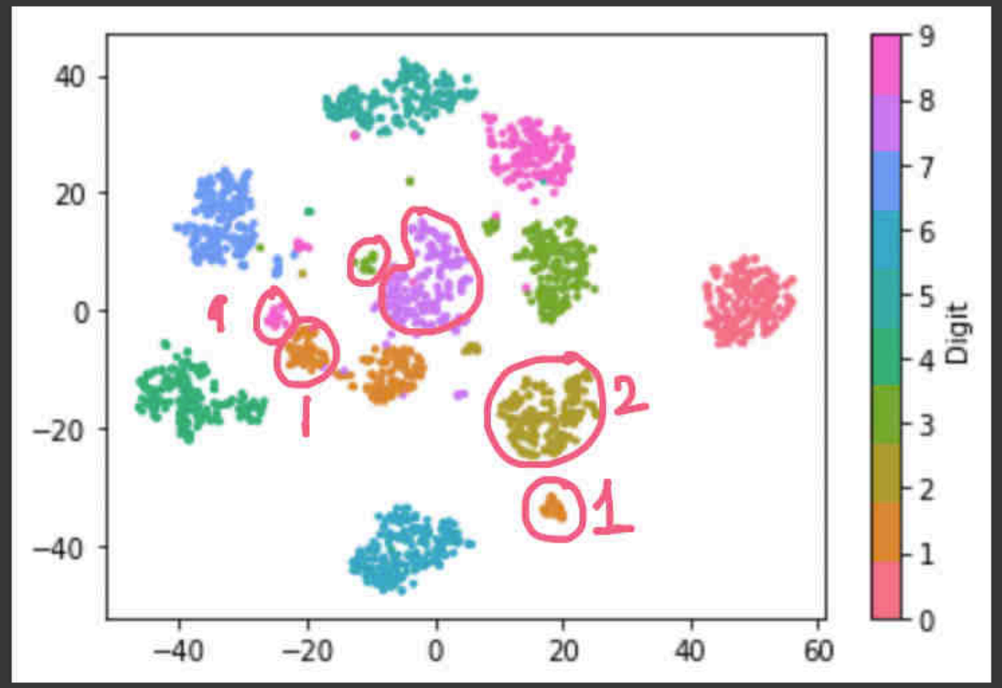
→ not colored & no labels

→ colored

```
%%time
Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

```
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



8x8 pixels → 64 dim-vec

```
%%time
Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

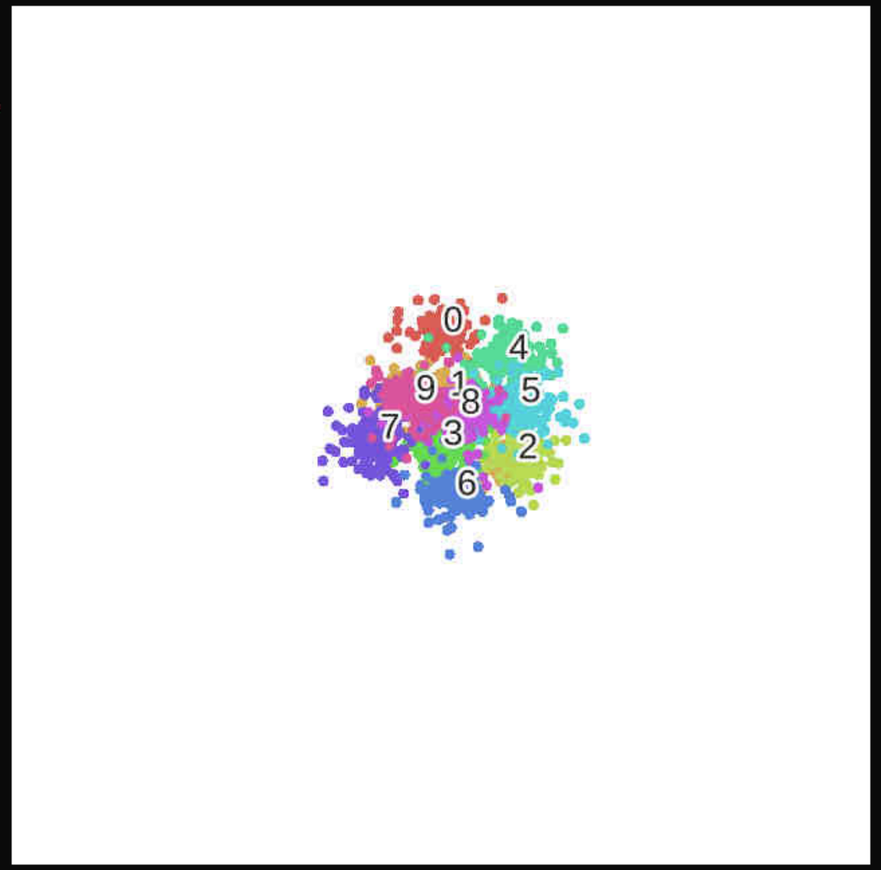
Chrome File Edit View History Bookmarks Profiles Tab Window Help

PCA_tSNE_UMAP.ipynb - Colab | animation-94a2c1ff.gif (640×6 | How to Use t-SNE Effectively | 1802.03426.pdf | UMAP Dimensionality Reductio | Understanding UMAP

oreilly.com/content/wp-content/uploads/sites/2/2019/06/animation-94a2c1ff.gif

t-SNE → opt problem (SGD)

min KL div (P, Q)

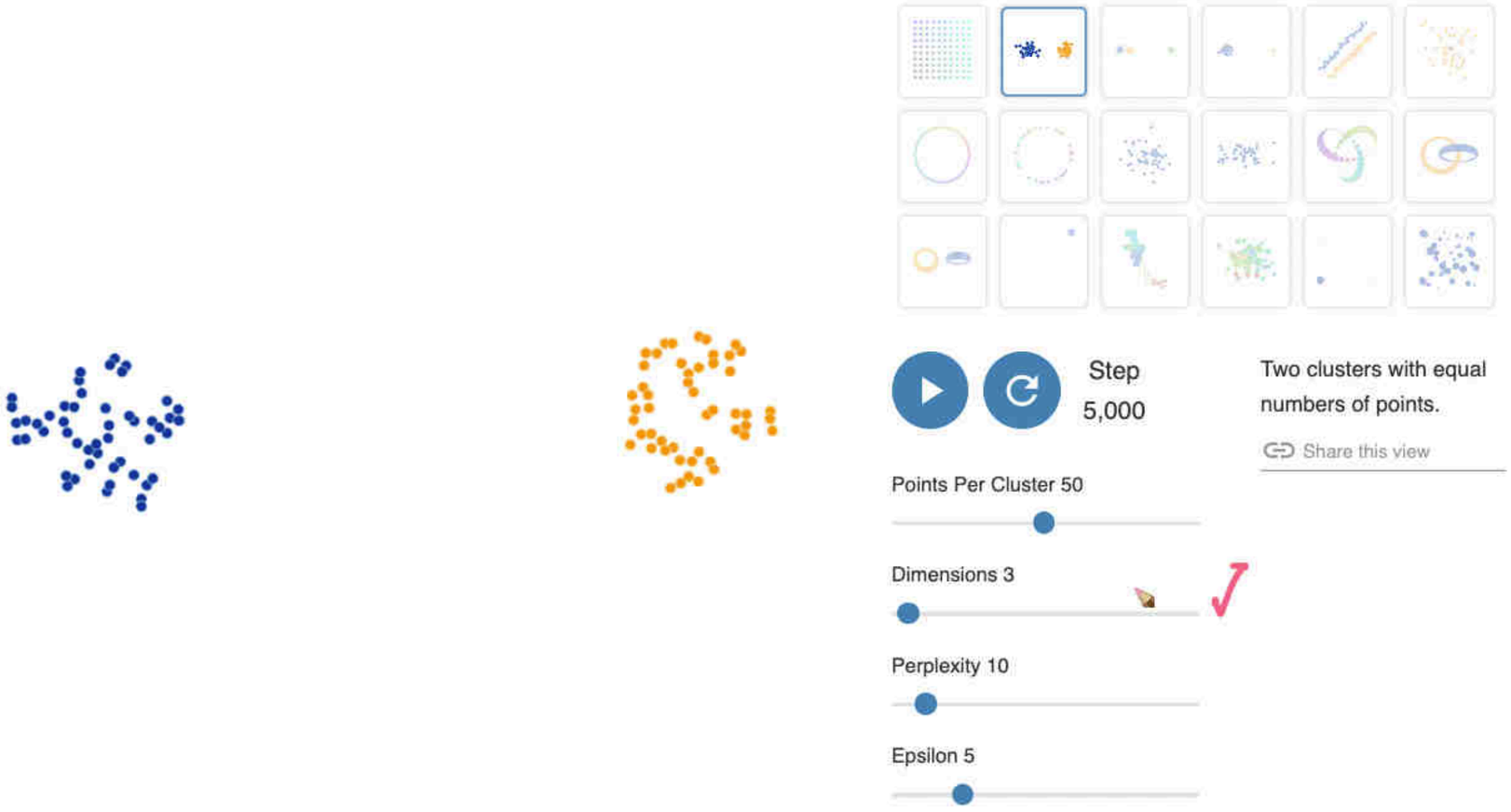


animation

{ 1st iteration

7 / 7

sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.

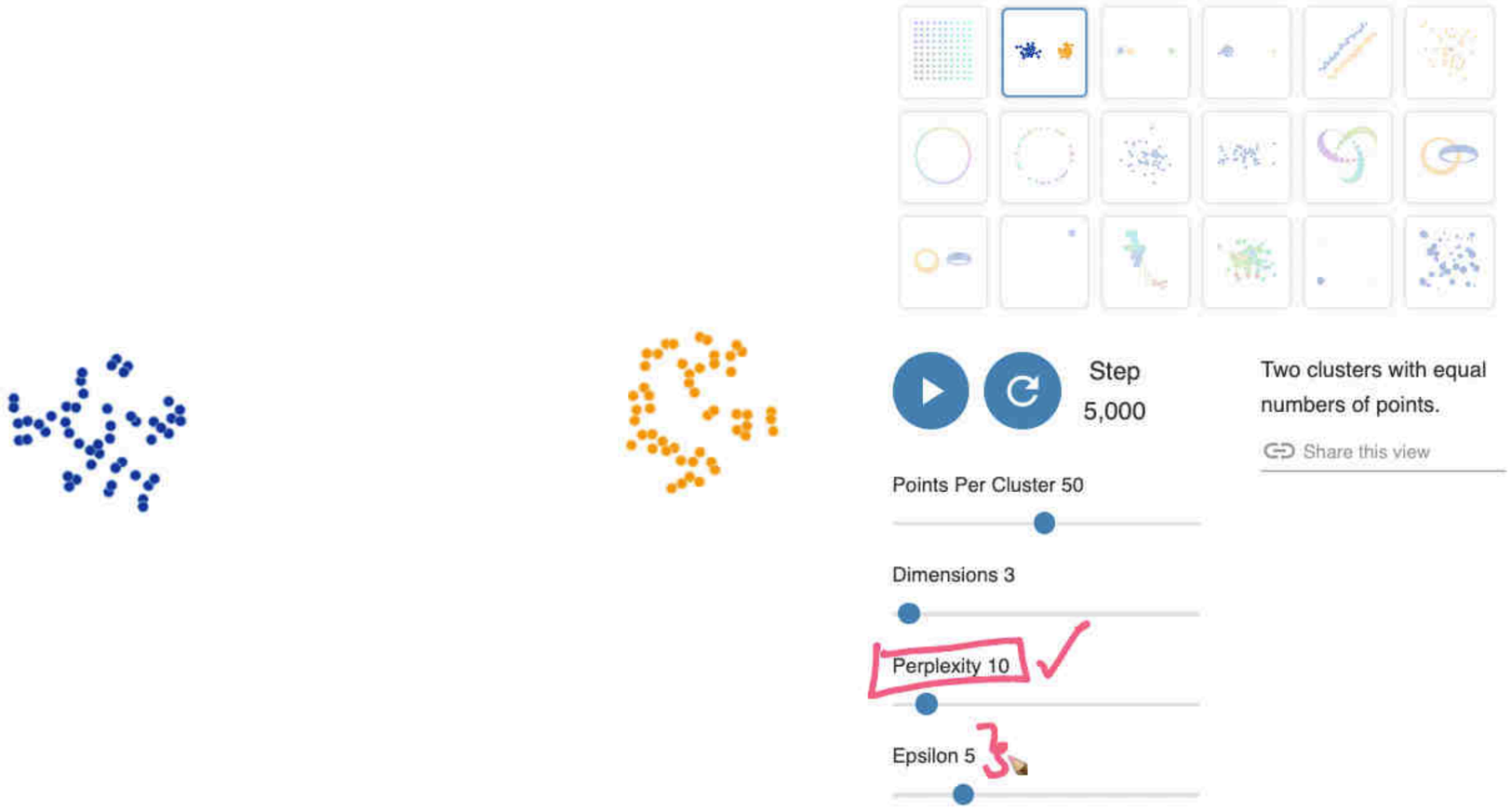


MARTIN WATTENBERG
Google Brain
FERNANDA VIÉGAS
Google Brain
IAN JOHNSON
Google Cloud

Oct. 13
2016

Citation:
Wattenberg, et al., 2016

sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.



MARTIN WATTENBERG
Google Brain
FERNANDA VIÉGAS
Google Brain
IAN JOHNSON
Google Cloud

Oct. 13
2016

Citation:
Wattenberg, et al., 2016

sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.

Crowding

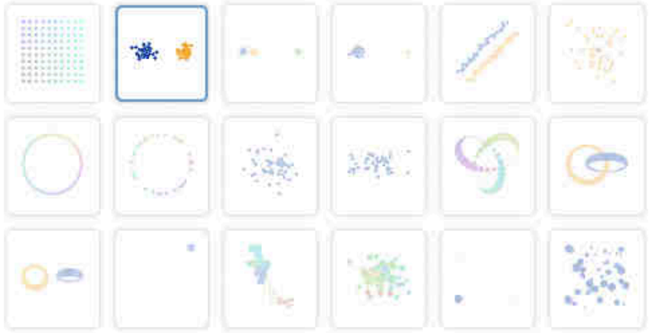
Perp ↑

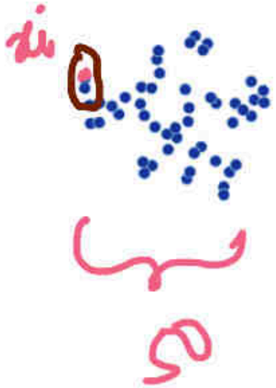
(preserve global structure)

Global

{ local-structure

Perp = 90





Step 5,000

Two clusters with equal numbers of points.

Share this view

Points Per Cluster 50

Dimensions 3

Perplexity 10

Epsilon 5

(5-50)

≈ effective # NN whose dist we want to preserve

MARTIN WATTENBERG Google Brain
FERNANDA VIÉGAS Google Brain
IAN JOHNSON Google Cloud

Oct. 13 2016
Citation: Wattenberg, et al., 2016

10 / 10

sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.

Step 5,000

A dense, tight cluster inside of a wide, sparse cluster. Perplexity makes a big difference here.

Share this view

Points Per Cluster 50

Dimensions 2

Perplexity 10

Epsilon 5

MARTIN WATTENBERG
Google Brain

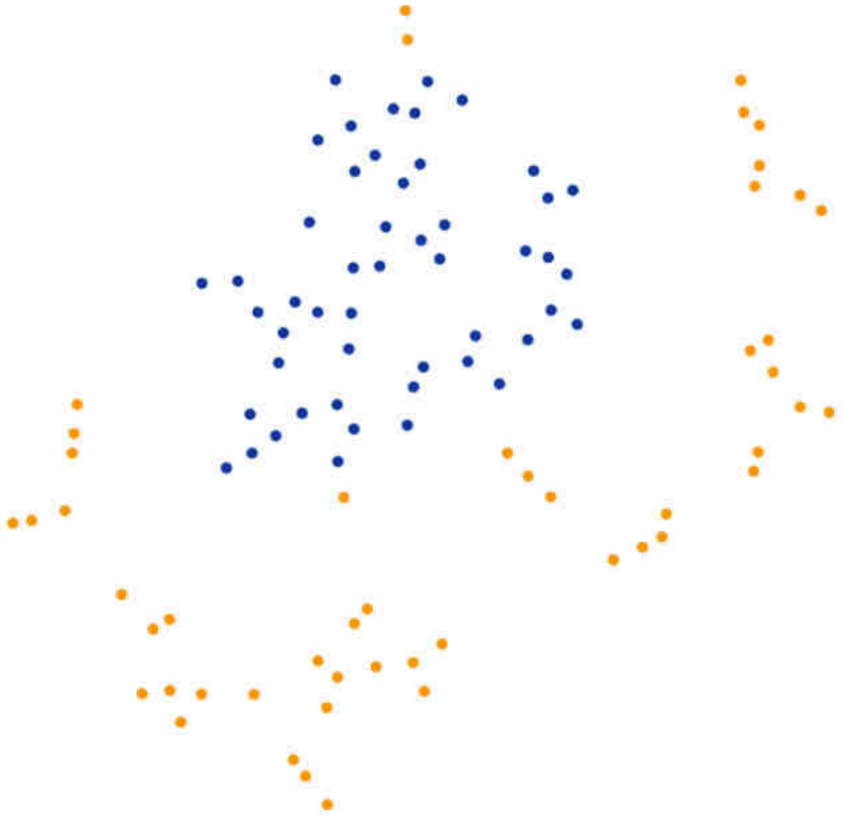
FERNANDA VIÉGAS
Google Brain

IAN JOHNSON
Google Cloud

Oct. 13
2016

Citation:
Wattenberg, et al., 2016

sometimes be mysterious or misleading. By exploring how it behaves in simple cases, we can learn to use it more effectively.



Step
5,000

A dense, tight cluster inside of a wide, sparse cluster. Perplexity makes a big difference here.

Share this view

Points Per Cluster 50

Dimensions 4

Perplexity 10

Epsilon 5

MARTIN WATTENBERG
Google Brain

FERNANDA VIÉGAS
Google Brain

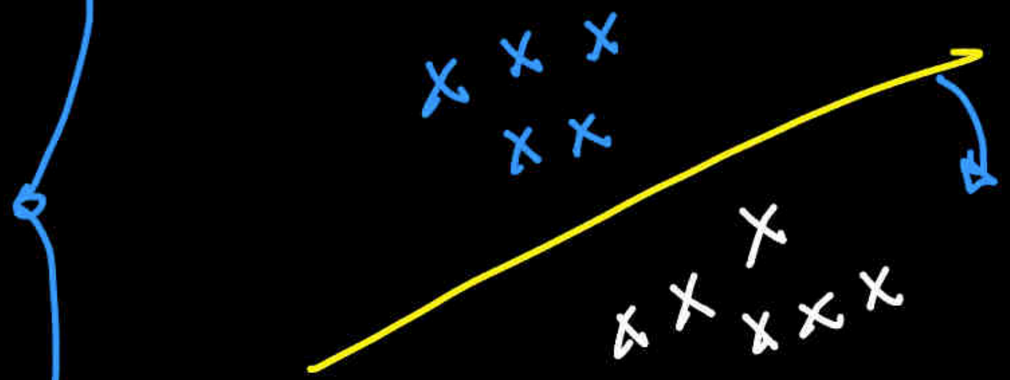
IAN JOHNSON
Google Cloud

Oct. 13
2016

Citation:
Wattenberg, et al., 2016

logistic regression

2D & 3D



Lesson

1. Those hyperparameters really matter

Let's start with the "hello world" of t-SNE: a data set of two widely separated clusters. To make things as simple as possible, we'll consider clusters in a 2D plane, as shown in the lefthand diagram. (For clarity, the two clusters are color coded.) The diagrams at right show t-SNE plots for five different perplexity values.

5 - 50

Original

avoid perp = n

With perplexity values in the range (5 - 50) suggested by van der Maaten & Hinton, the diagrams do show these clusters, although with very different shapes. Outside that range, things get a little weird. With perplexity 2, local variations dominate. The image for perplexity 100, with merged clusters, illustrates a pitfall: for the algorithm to operate properly, the perplexity really should be smaller than the number of points. Implementatio

points. Implementations can give unexpected behavior otherwise.

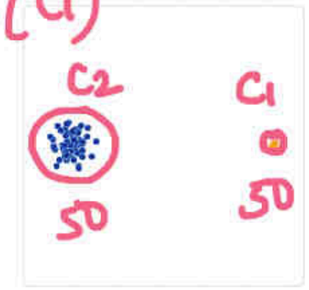
Each of the plots above was made with 5,000 iterations with a learning rate (often called "epsilon") of 10, and had reached a point of stability by step 5,000. How much of a difference do those values make? In our experience, the most important thing is to iterate until reaching a stable configuration.

The images above show five different runs at perplexity 30. The first four were stopped before stability. After 10, 20, 60, and 120 steps you can see layouts with seeming 1-dimensional and even pointlike images of the clusters. If you see a t-SNE plot with strange "pinched" shapes, chances are the process was stopped too early. Unfortunately, there's no fixed number of steps that yields a stable result. Different data sets can require different numbers of iterations to converge.

Another natural quest
hyperparameters produce the same results. In this simple two-cluster

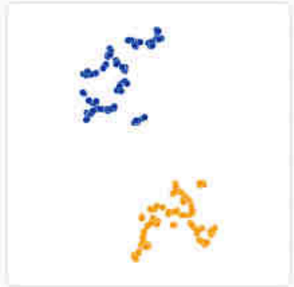
2. Cluster sizes in a t-SNE plot mean nothing

So far, so good. But what if the two clusters have different standard deviations, and so different sizes? (By size we mean bounding box measurements, not number of points.) Below are t-SNE plots for a mixture of Gaussians in plane, where one is 10 times as dispersed as the other.



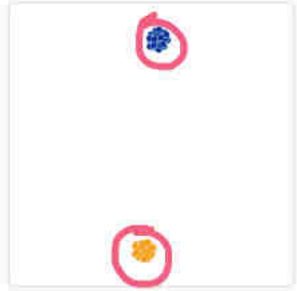
Original

Perplexity: 2
Step: 5,000

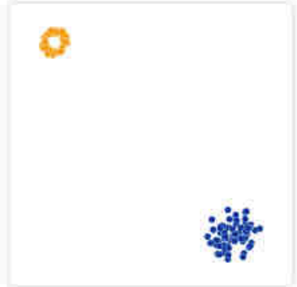


Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000



Perplexity: 50
Step: 5,000



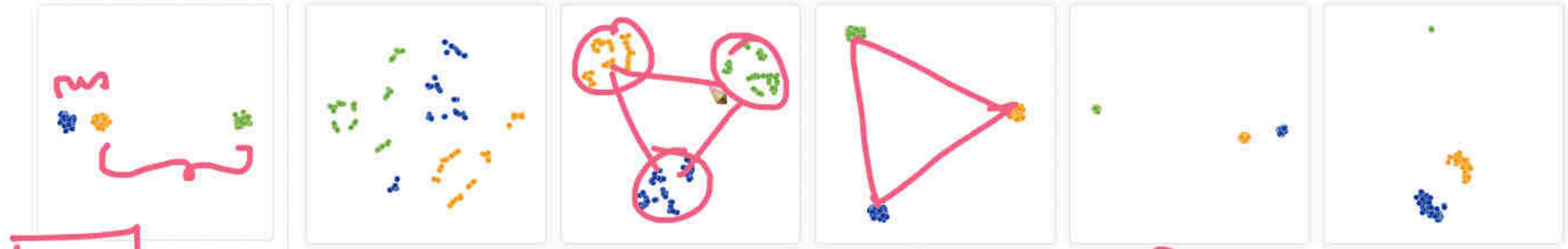
Perplexity: 100
Step: 5,000

Surprisingly, the two clusters look about same size in the t-SNE plots. What's going on? The t-SNE algorithm adapts its notion of "distance" to regional density variations in the data set. As a result, it naturally expands dense clusters, and contracts sparse ones, evening out cluster sizes. To be clear, this is a different effect than the run-of-the-mill fact that any dimensionality reduction technique will distort distances. (After all, in this example all

Lesson

3. Distances between clusters might not mean anything

What about distances *between* clusters? The next diagrams show three Gaussians of 50 points each, one pair being 5 times as far apart as another pair.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity 30
Step: 5,000

Perplexity 50
Step: 5,000

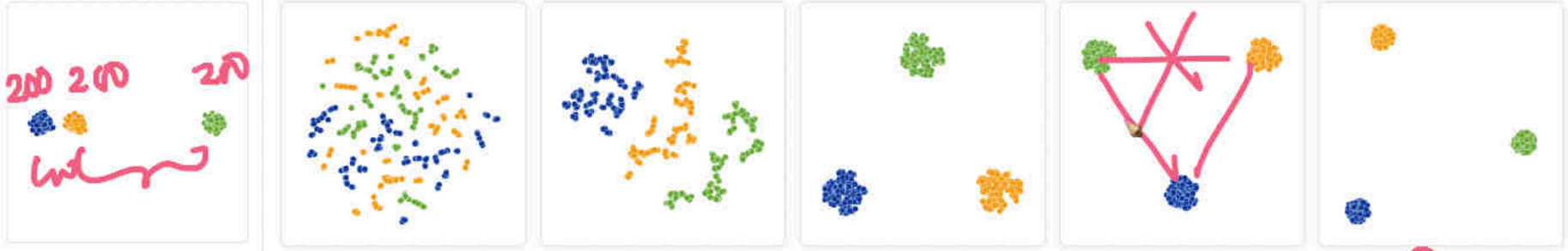
Perplexity: 100
Step: 5,000

At perplexity 50, the diagram gives a good sense of the global geometry. For lower perplexity values the clusters look equidistant. When the perplexity is 100, we see the global geometry fine, but one of the cluster appears, falsely, much smaller than the others. Since perplexity 50 gave us a good picture in this example, can we always set perplexity to 50 if we want to see global geometry?

Sadly, no. If we add m

appears, falsely, much smaller than the others. Since perplexity 50 gave us a good picture in this example, can we always set perplexity to 50 if we want to see global geometry?

Sadly, no. If we add more points to each cluster, the perplexity has to increase to compensate. Here are the t-SNE diagrams for three Gaussian clusters with 200 points each, instead of 50. Now none of the trial perplexity values gives a good result.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

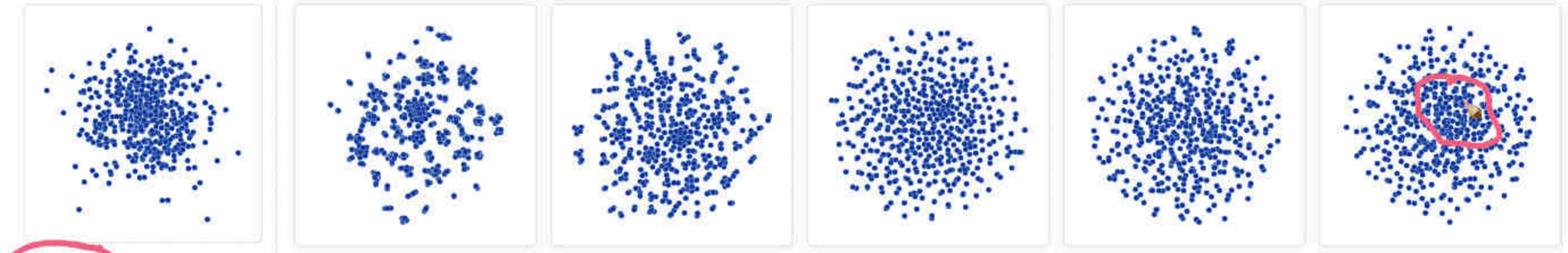
It's bad news that seeing global geometry requires fine-tuning perplexity. Real-world data would probably have multiple clusters with different numbers of elements. There may not be one perplexity value that will capture distances across all clusters—and sadly perplexity is a global parameter. Fixing this problem might be an interesting area for future research.

The basic message is t

t-SNE plot may mean nothing

lesson 4. Random noise doesn't always look random.

A classic pitfall is thinking you see patterns in what is really just random data. Recognizing noise when you see it is a critical skill, but it takes time to build up the right intuitions. A tricky thing about t-SNE is that it throws a lot of existing intuition out the window. The next diagrams show genuinely random data, 500 points drawn from a unit Gaussian distribution in 100 dimensions. The left image is a projection onto the first two coordinates.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

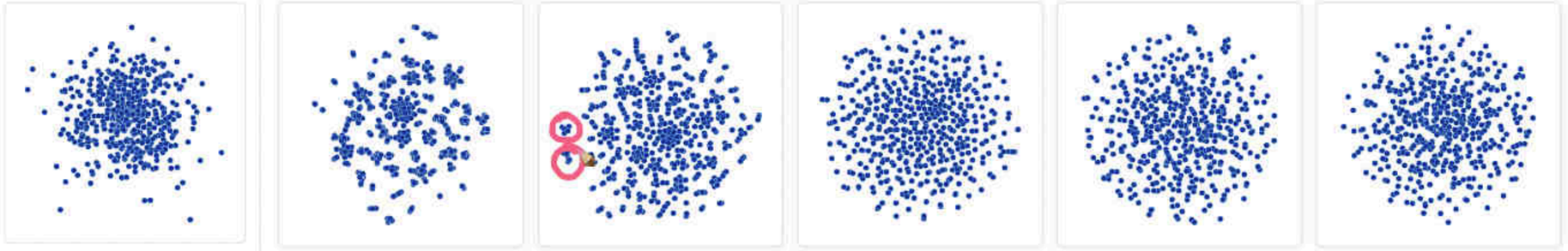
Perplexity: 100
Step: 5,000

The plot with perplexity 2 seems to show dramatic clusters. If you were tuning perplexity to bring out structure in the data, you might think you'd hit the jackpot.

Of course, since we kn

4. Random noise doesn't always look random.

A classic pitfall is thinking you see patterns in what is really just random data. Recognizing noise when you see it is a critical skill, but it takes time to build up the right intuitions. A tricky thing about t-SNE is that it throws a lot of existing intuition out the window. The next diagrams show genuinely random data, 500 points drawn from a unit Gaussian distribution in 100 dimensions. The left image is a projection onto the first two coordinates.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

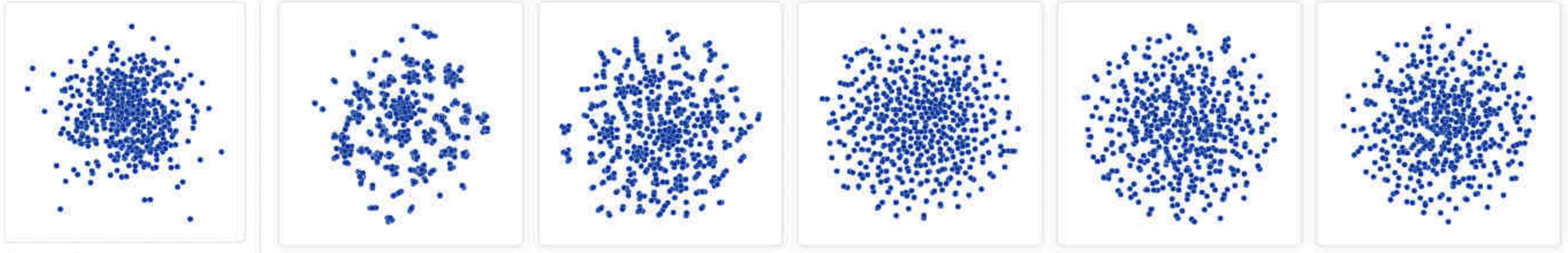
Perplexity: 100
Step: 5,000

The plot with perplexity 2 seems to show dramatic clusters. If you were tuning perplexity to bring out structure in the data, you might think you'd hit the jackpot.

Of course, since we kn

4. Random noise doesn't always look random.

A classic pitfall is thinking you see patterns in what is really just random data. Recognizing noise when you see it is a critical skill, but it takes time to build up the right intuitions. A tricky thing about t-SNE is that it throws a lot of existing intuition out the window. The next diagrams show genuinely random data, 500 points drawn from a unit Gaussian distribution in 100 dimensions. The left image is a projection onto the first two coordinates.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

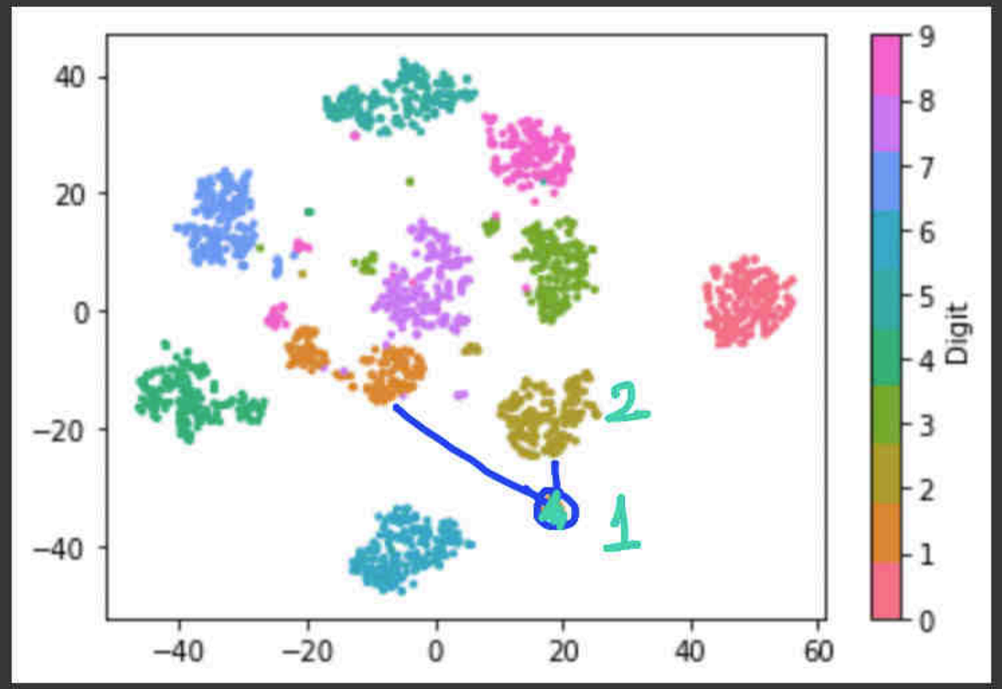
Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

The plot with perplexity 2 seems to show dramatic clusters. If you were tuning perplexity to bring out structure in the data, you might think you'd hit the jackpot.

Of course, since we kn

```
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



→ tool → hypothesis/idea
→ look @ the data + domain knowl

```
[ ] %%time
    Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

tSNE in ML

$$\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{n}$$

y_i

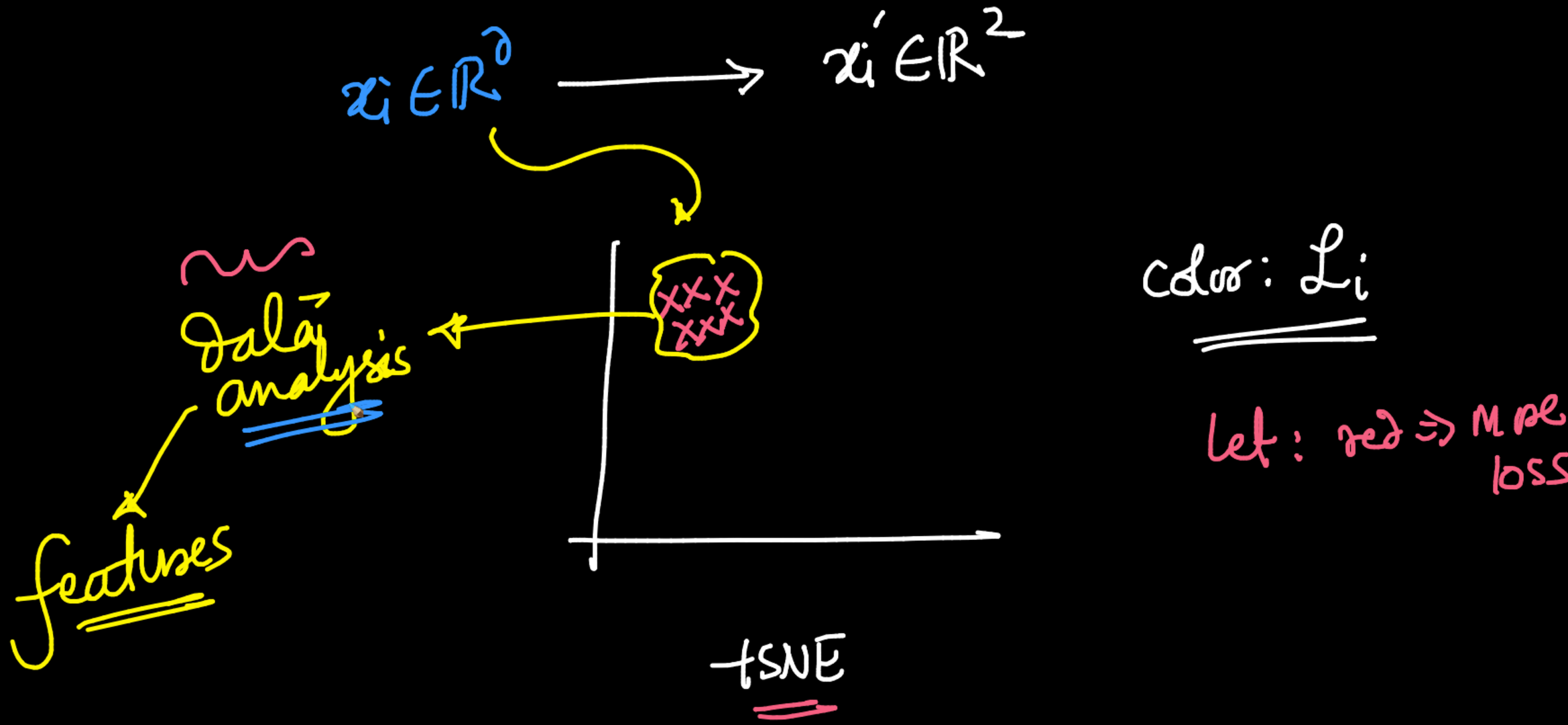
Classification

→ outliers

→ clustering

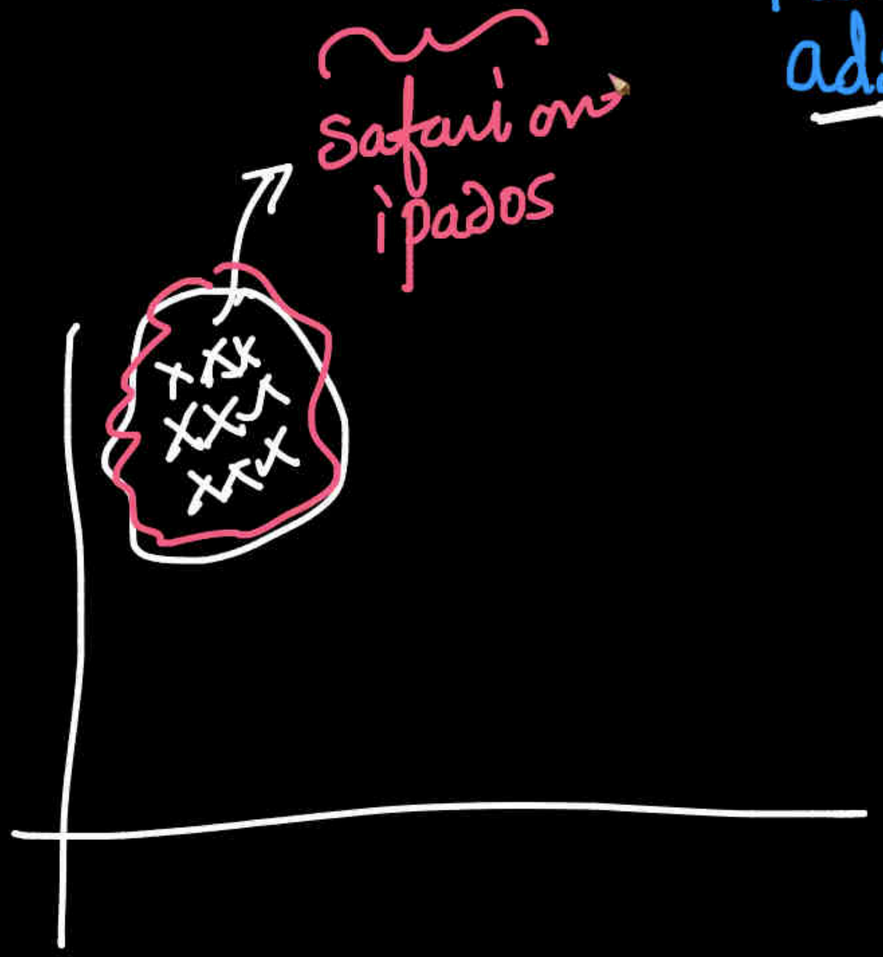
$$\mathcal{L}_i = \text{diff}(y_i, \hat{y}_i)$$

↑ classfn.loss



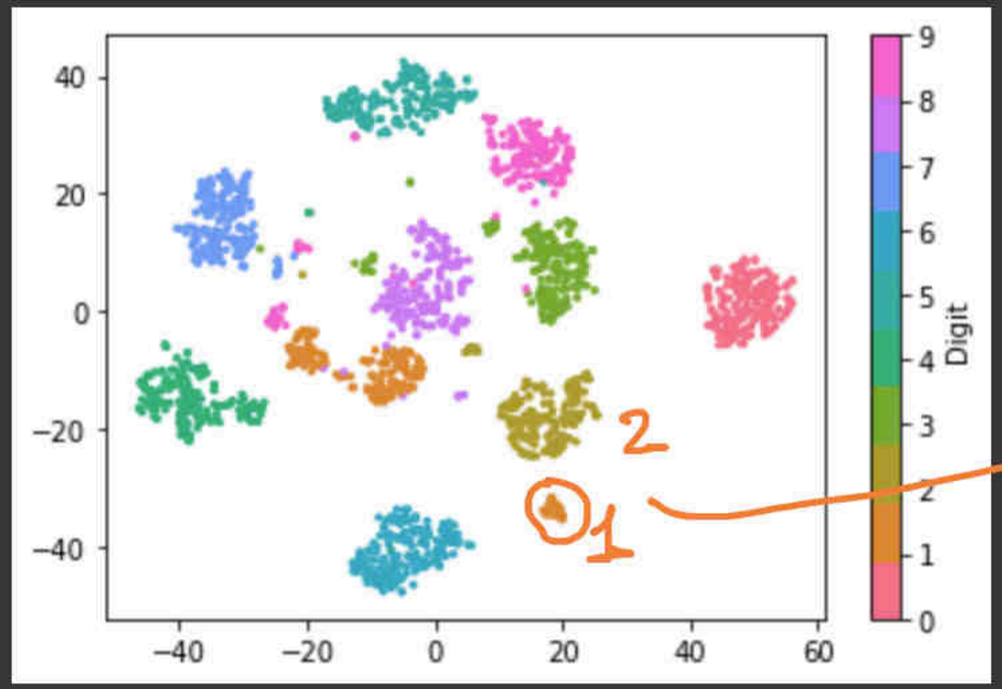
Chrome File Edit View History Bookmarks Profiles Tab Window Help 9:44

Concretely:



ads on web-browser/ app/ Mobile/ kp

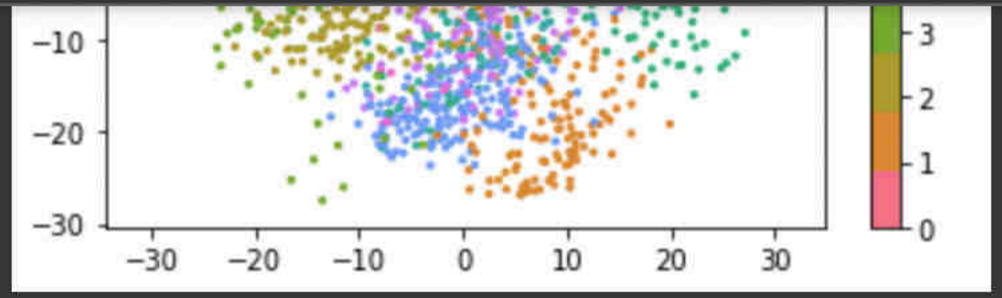
```
fig, ax = plt.subplots()
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



- plot
- underlying data

```
[ ]  %%time
     Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

[ ]



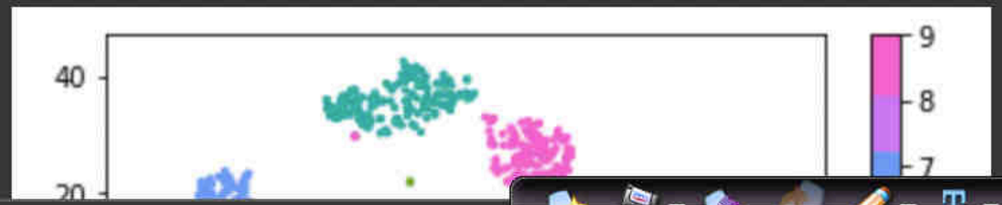
1797x64

[ ]
```
# tSNE
%%time
Z2 = TSNE(n_jobs=-1, initialization='random', random_state=42, negative_gradient_method='bh').fit(X
```

1797x2

```
CPU times: user 14.6 s, sys: 179 ms, total: 14.8 s
Wall time: 8.32 s
```

```
fig, ax = plt.subplots()
im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



27 / 27


```
[ ] # tSNE
    %%time
    Z2 = TSNE(n_jobs=-1, initialization='random', random_state=42, negative_gradient_method='bh').fit(X
```

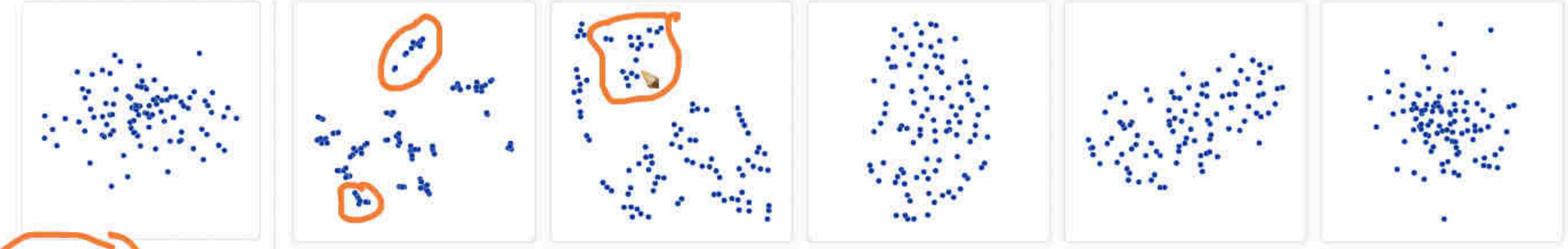
```
CPU times: user 14.6 s, sys: 179 ms, total: 14.8 s
Wall time: 8.32 s
```

```
[ ] fig, ax = plt.subplots()
    im = ax.scatter(Z2[:,0], Z2[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
    cbar = fig.colorbar(im, ax=ax,label='Digit')
```

Lesson

5. You can see some shapes, sometimes

It's rare for data to be distributed in a perfectly symmetric way. Let's take a look at an axis-aligned Gaussian distribution in 50 dimensions, where the standard deviation in coordinate i is 1/i. That is, we're looking at a long-ish ellipsoidal cloud of points.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

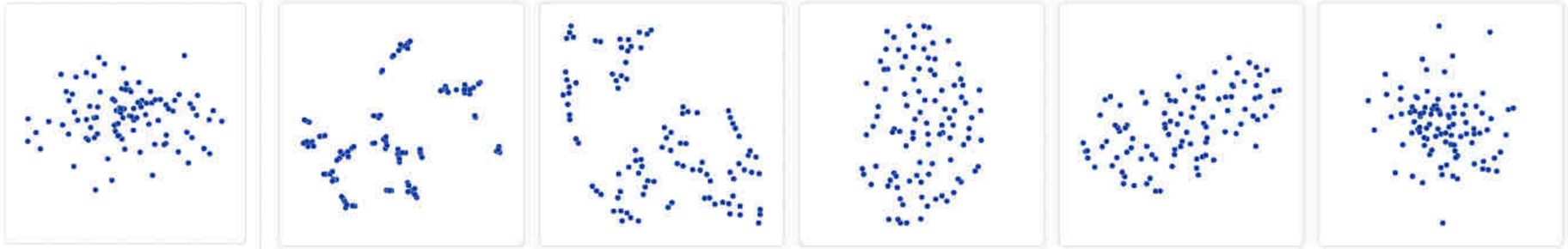
Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

For high enough perplexity values, the elongated shapes are easy to read. On the other hand, at low perplexity, local effects and meaningless "clumping" take center stage. More extreme shapes also come through, but again only at the right perplexity. For example, here are two clusters of 75 points each in 2D, arranged in parallel lines with a bit of noise.

5. You can see some shapes, sometimes

It's rare for data to be distributed in a perfectly symmetric way. Let's take a look at an axis-aligned Gaussian distribution in 50 dimensions, where the standard deviation in coordinate i is 1/i. That is, we're looking at a long-ish ellipsoidal cloud of points.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

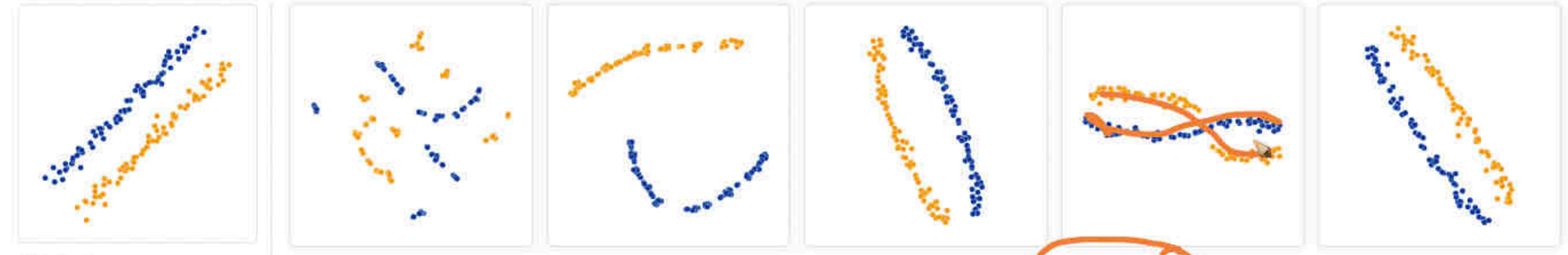
Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

For high enough perplexity values, the elongated shapes are easy to read. On the other hand, at low perplexity, local effects and meaningless "clumping" take center stage. More extreme shapes also come through, but again only at the right perplexity. For example, here are two clusters of 75 points each in 2D, arranged in parallel lines with a bit of noise.

On the other hand, at low perplexity, local effects and meaningless "clumping" take center stage. More extreme shapes also come through, but again only at the right perplexity. For example, here are two clusters of 75 points each in 2D, arranged in parallel lines with a bit of noise.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

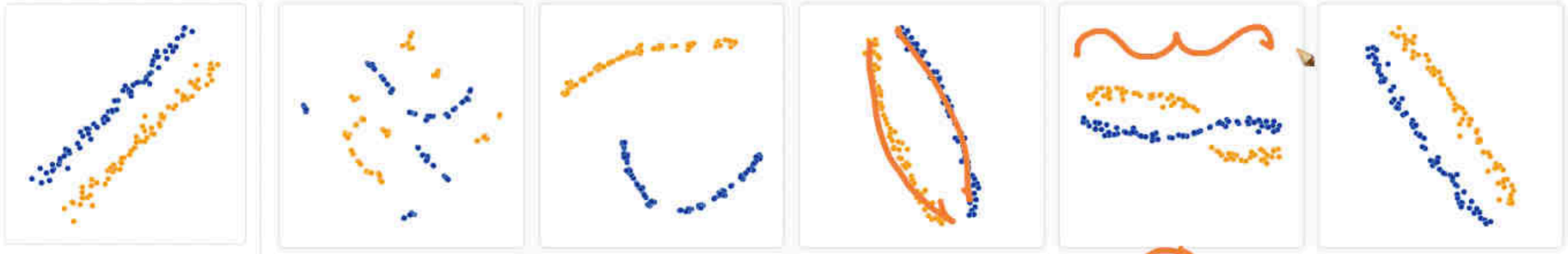
Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

For a certain range of perplexity the long clusters look close to correct, which is reassuring.

Even in the best cases, though, there's a subtle distortion: the lines are slightly curved outwards in the t-SNE diagram. The reason is that, as usual, t-SNE tends to expand denser regions of data. Since the middles of the clusters have less empty space around them than the ends, the algorithm magnifies them.

On the other hand, at low perplexity, local effects and meaningless "clumping" take center stage. More extreme shapes also come through, but again only at the right perplexity. For example, here are two clusters of 75 points each in 2D, arranged in parallel lines with a bit of noise.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

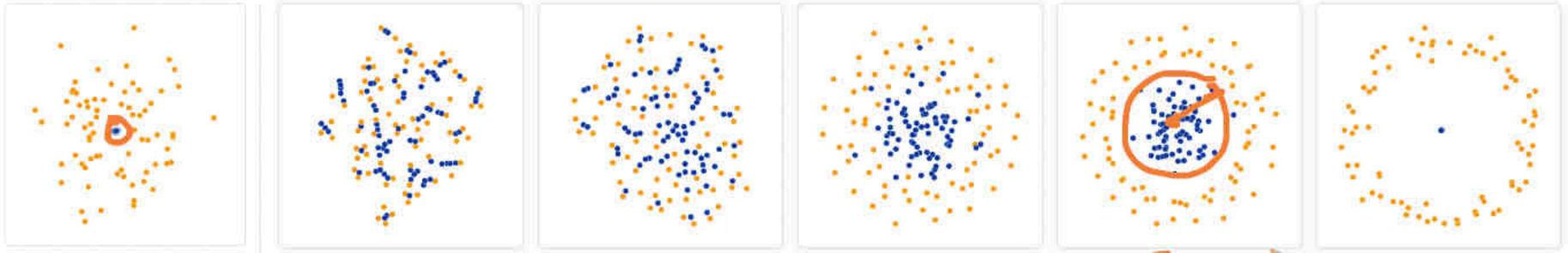
Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

For a certain range of perplexity the long clusters look close to correct, which is reassuring.

Even in the best cases, though, there's a subtle distortion: the lines are slightly curved outwards in the t-SNE diagram. The reason is that, as usual, t-SNE tends to expand denser regions of data. Since the middles of the clusters have less empty space around them than the ends, the algorithm magnifies them.

Sometimes you can read topological information off a t-SNE plot, but that typically requires views at multiple perplexities. One of the simplest topological properties is containment. The plots below show two groups of 75 points in 50 dimensional space. Both are sampled from symmetric Gaussian distributions centered at the origin, but one is 50 times more tightly dispersed than the other. The "small" distribution is in effect contained in the large one.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

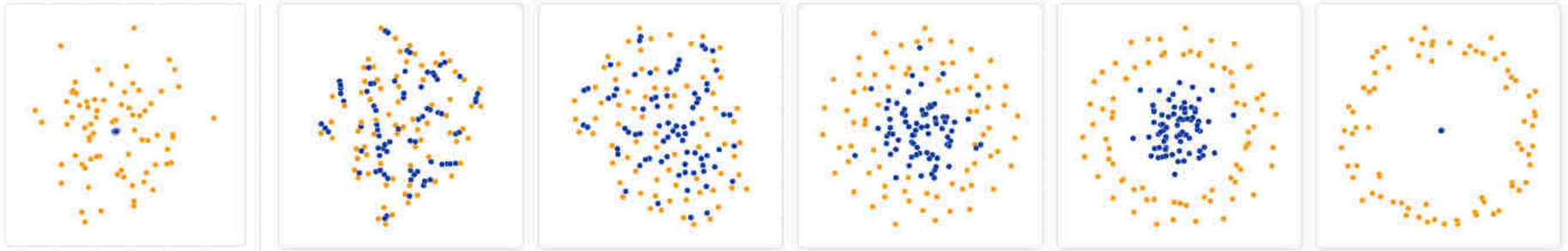
Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

The perplexity 30 view shows the basic topology correctly, but again t-SNE greatly exaggerates the size of the smaller group of points. At perplexity 50, there's a new phenomenon: the outer group becomes a circle, as the plot tries to depict the fact that all its points are about the same distance from the inner group. If you looked at this image alone, it would be easy to misread these outer points as a one-dimensional structure.

Sometimes you can read topological information off a t-SNE plot, but that typically requires views at multiple perplexities. One of the simplest topological properties is containment. The plots below show two groups of 75 points in 50 dimensional space. Both are sampled from symmetric Gaussian distributions centered at the origin, but one is 50 times more tightly dispersed than the other. The "small" distribution is in effect contained in the large one.



Original

Perplexity: 2
Step: 5,000

Perplexity: 5
Step: 5,000

Perplexity: 30
Step: 5,000

Perplexity: 50
Step: 5,000

Perplexity: 100
Step: 5,000

The perplexity 30 view shows the basic topology correctly, but again t-SNE greatly exaggerates the size of the smaller group of points. At perplexity 50, there's a new phenomenon: the outer group becomes a circle, as the plot tries to depict the fact that all its points are about the same distance from the inner group. If you looked at this image alone, it would be easy to misread these outer points as a one-dimensional structure.

Conclusion

There's a reason that t-SNE has become so popular: it's incredibly flexible, and can often find structure where other dimensionality-reduction algorithms cannot. Unfortunately, that very flexibility makes it tricky to interpret. Out of sight from the user, the algorithm makes all sorts of adjustments that tidy up its visualizations. Don't let the hidden "magic" scare you away from the whole technique, though. The good news is that by studying how t-SNE behaves in simple cases, it's possible to develop an intuition for what's going on.

Acknowledgments

We are grateful to Chris Olah and Shan Carter for creating this platform, and for excellent design and editorial help from Shan Carter. Daniel Smilkov, James Wexler, and Chi Zeng provided many helpful comments. We also thank Andrej Karpathy for creating the tsnejs library used in the interactive diagrams.

This work was made possible

UMAP (2018)

assumption

- ✓ d-dim (high) → 2-dim
- ✓ manifold (surface)
- ✓ Uniformity assumption

Uniform manifold Approx & Proj

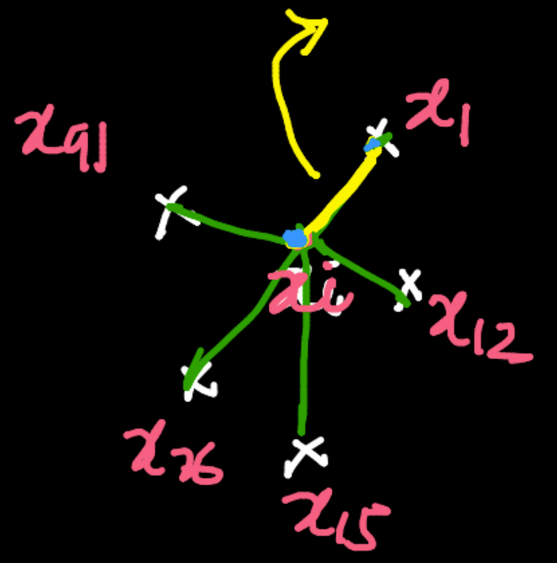
{ Topology } X

{ Algo + optmzn }

Algo + opt

d-dim

1. Finds neasest neighbors
   n-neighbors = 5 ✓



2. Builds a graph (d-dim) (weighted)

$G^{(d)}$

$W_{ij}^{d} = \frac{1}{dist^{d}(x_i, x_j)}$

- nodes : places
- edges : roads
- weights :

(3) for each x_i $\xrightarrow{\text{Proj}}$ y_i

x_i: d-dim

y_i: 2-dim

s.t $G^{(d)} \approx G^{(2)}$

$w_h(e)$ - known weights of edges from **high-dimensional** manifold approximation (found in step 1)

Cross Entropy

$$\min CE = \sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right)$$

For edges (e) in a set E (i.e. all edges found in step 1)

$w_l(e)$ - weights to be discovered for **low-dimensional** representation (to be found in this step [step 2])

The first term acts as an **"attractive force"** whenever there is a large weight associated to the high-dimensional case. This is because this term will be minimized when $w_l(e)$ is as large as possible, which will occur when the distance between the poinns is as small as possible.

The second term acts as a **"repulsive force"** whenever high-dimensional weight $w_h(e)$ is small. This is because the term will be minimized by making $w_l(e)$ as small as possible.

On balance the push and pull of the two "forces" will let the low-dimensional representation settle into a state that ... original data.

$w_h(e)$ - known weights of edges from **high-dimensional** manifold approximation (found in step 1)

$w_h(e)$ small!

Cross-Entropy

$$CE = \sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right)$$

0.000

0.9999

For edges (e) in a set E (i.e. all edges found in step 1)

$w_l(e)$ - weights to be discovered for **low-dimensional** representation (to be found in this step [step 2])

The first term acts as an **"attractive force"** whenever there is a large weight associated to the high-dimensional case. This is because this term will be minimized when $w_l(e)$ is as large as possible, which will occur when the distance between the poinns is as small as possible.

The second term acts as a **"repulsive force"** whenever high-dimensional weight $w_h(e)$ is small. This is because the term will be minimized by making $w_l(e)$ as small as possible.

On balance the push and pull of the two "forces" will let the low-dimensional representation settle into a state that r… original data.

$w_h(e)$ - known weights of edges from **high-dimensional** manifold approximation (found in step 1)

UMAP → preserve global

Cross-Entropy

$$CE = \sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right)$$

For edges (e) in a set E (i.e. all edges found in step 1)

$w_l(e)$ - weights to be discovered for **low-dimensional** representation (to be found in this step [step 2])

The first term acts as an **"attractive force"** whenever there is a large weight associated to the high-dimensional case. This is because this term will be minimized when $w_l(e)$ is as large as possible, which will occur when the distance between the poinns is as small as possible.

The second term acts as a **"repulsive force"** whenever high-dimensional weight $w_h(e)$ is small. This is because the term will be minimized by making $w_l(e)$ as small as possible.

On balance the push and pull of the two "forces" will let the low-dimensional representation settle into a state that r original data.

$w_h(e)$ - known weights of edges from **high-dimensional** manifold approximation (found in step 1)

Cross-Entropy

$$CE = \sum_{e \in E} w_h(e) \log\left(\frac{w_h(e)}{w_l(e)}\right) + (1 - w_h(e)) \log\left(\frac{1 - w_h(e)}{1 - w_l(e)}\right)$$

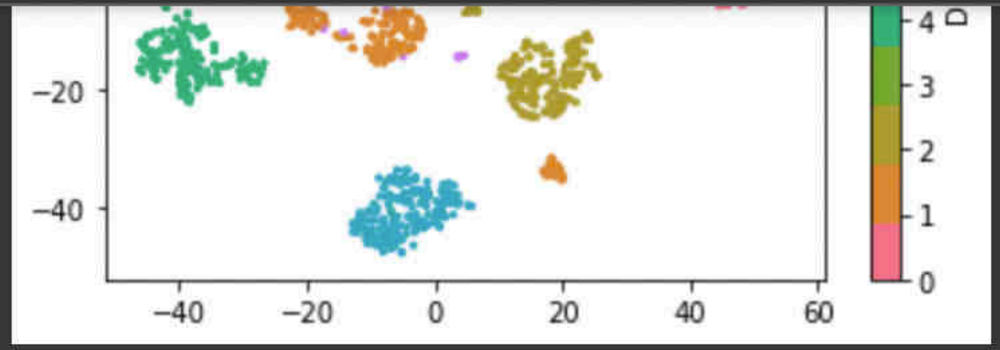
For edges (e) in a set E (i.e. all edges found in step 1)

$w_l(e)$ - weights to be discovered for **low-dimensional** representation (to be found in this step [step 2])

The first term acts as an **"attractive force"** whenever there is a large weight associated to the high-dimensional case. This is because this term will be minimized when $w_l(e)$ is as large as possible, which will occur when the distance between the poinns is as small as possible.

The second term acts as a **"repulsive force"** whenever high-dimensional weight $w_h(e)$ is small. This is because the term will be minimized by making $w_l(e)$ as small as possible.

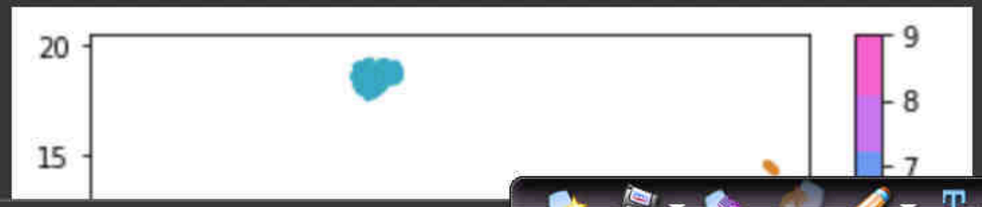
On balance the push and pull of the two "forces" will let the low-dimensional representation settle into a state that r original data.



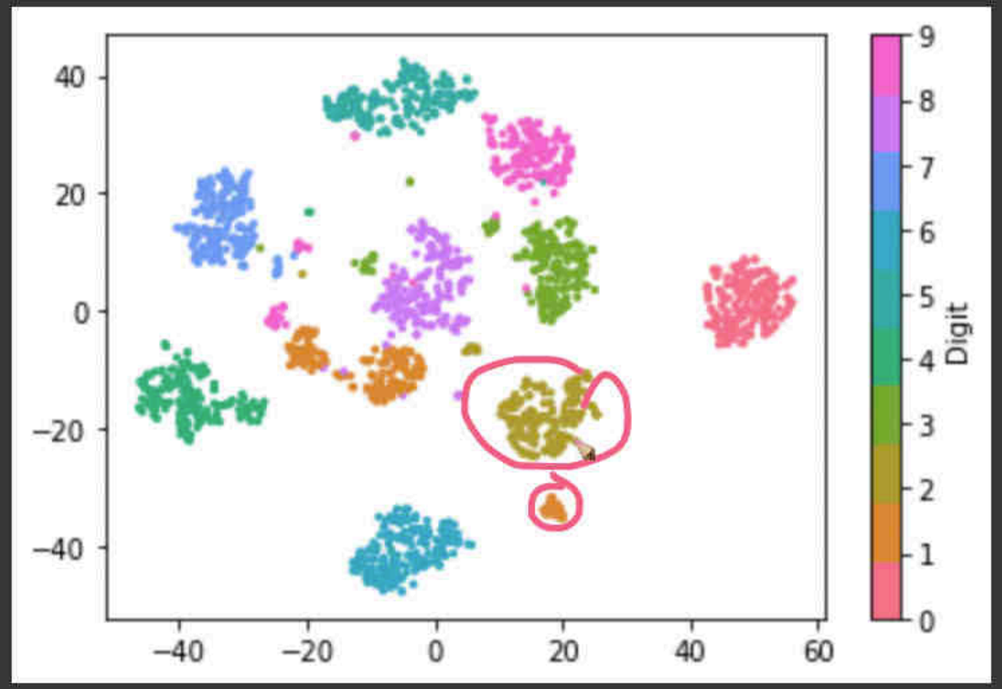

```
%%time
Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

```
fig, ax = plt.subplots()
im = ax.scatter(Z3[:,0], Z3[:,1], s=10, c=Y, cmap=cmap, edgecolor='none')
cbar = fig.colorbar(im, ax=ax,label='Digit')
```



```
cbar = fig.colorbar(im, ax=ax,label='Digit')
```




```
[ ]  %%time
     Z3 = UMAP(init='random', random_state=42).fit_transform(X)
```

```
CPU times: user 8.39 s, sys: 30 ms, total: 8.42 s
Wall time: 8.27 s
```

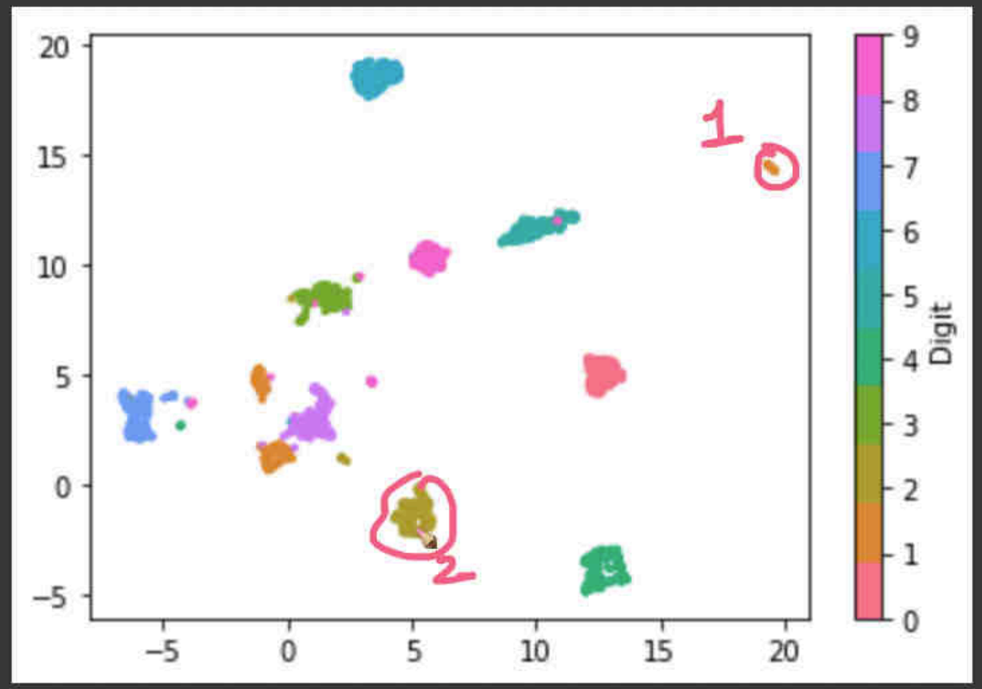
Chrome File Edit View History Bookmarks Profiles Tab Window Help 10:15

PCA_tSNE_UMAP.ipynb - Colab | animation-94a2c1ff.gif (640×6 | How to Use t-SNE Effectively | 1802.03426.pdf | UMAP Dimensionality Reductio | Understanding UMAP | Manifold - Wikipedia

colab.research.google.com/drive/13olY9wlU97WrxgEwKa9XIjJr0RF_CbjH#scrollTo=cpws4FmMDLYZ

+ Code + Text Connect

```
[ ]  cbar = fig.colorbar(im, ax=ax,label='Digit')
```



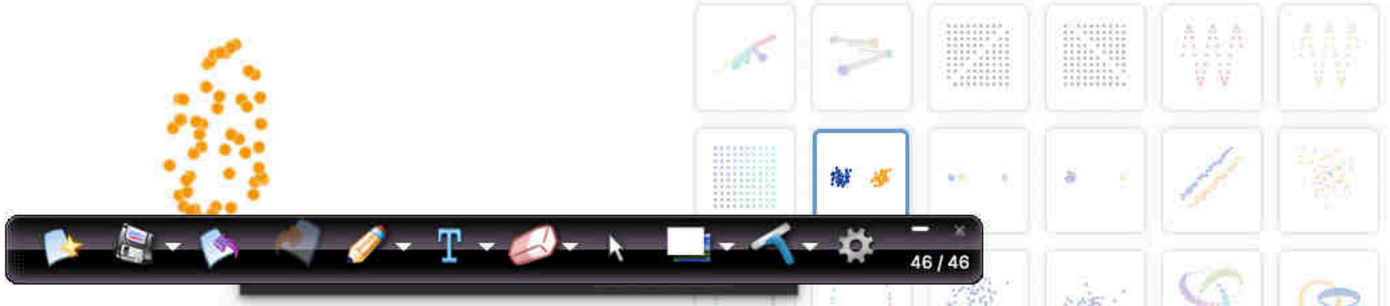
45 / 45

Understanding UMAP

Andy Coenen, Adam Pearce | *Google PAIR*

Dimensionality reduction is a powerful tool for machine learning practitioners to visualize and understand large, high dimensional datasets. One of the most widely used techniques for visualization is t-SNE, but its performance suffers with large datasets and using it correctly can be challenging.

UMAP is a new technique by McInnes et al. that offers a number of advantages over t-SNE, most notably increased speed and better preservation of the data's global structure. In this article, we'll take a look at the theory behind UMAP in order to better understand how the algorithm works, how to use it effectively, and how its performance compares with t-SNE.



algorithm works, how to use it effectively, and how its performance compares with t-SNE.

Figure 1: *Apply UMAP projection to various toy datasets, powered by umap-js.*

So what does UM[illegible]ling well in terms of both dataset size and dimensionality. For example, UMAP can project the 784-dimensional,

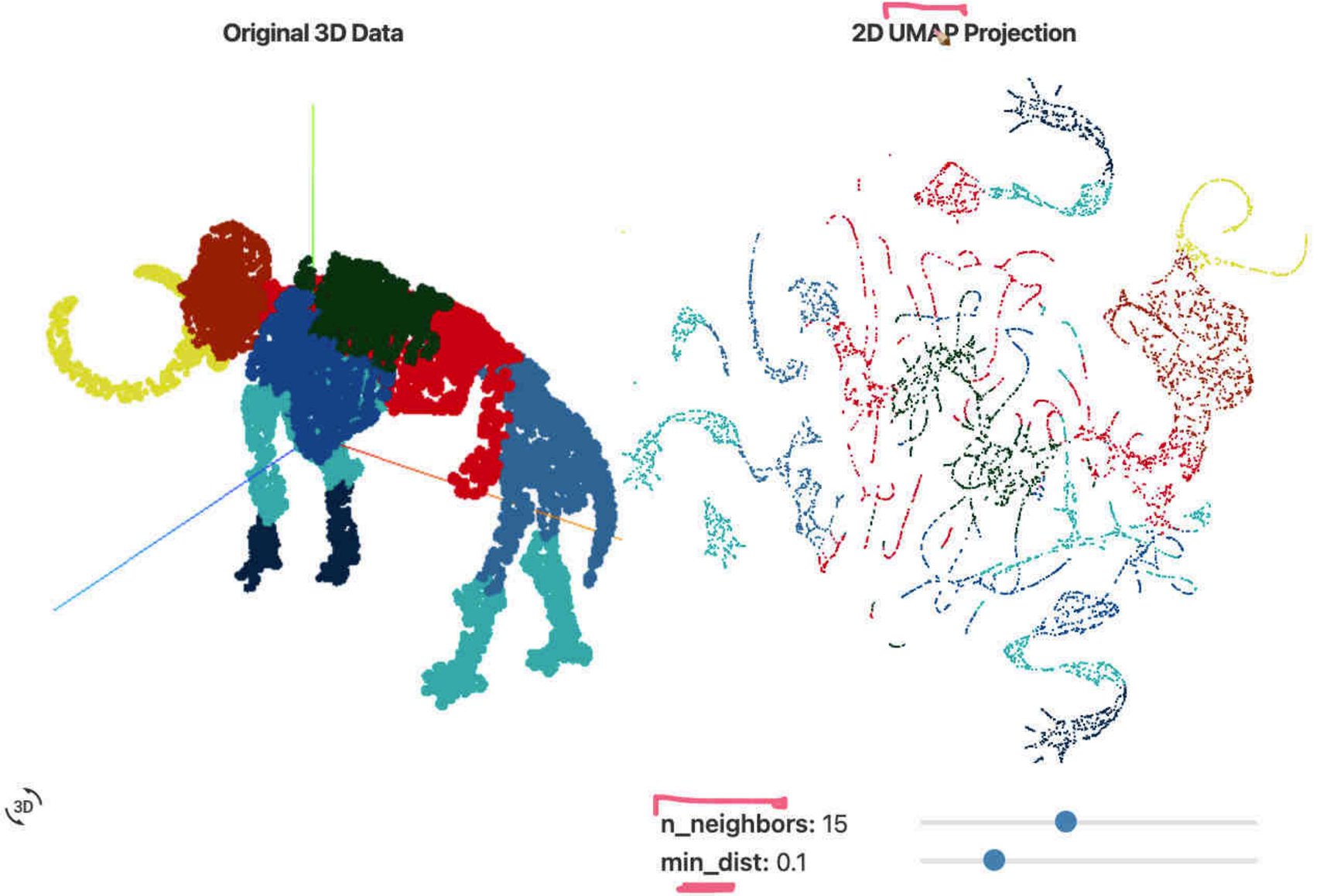


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with

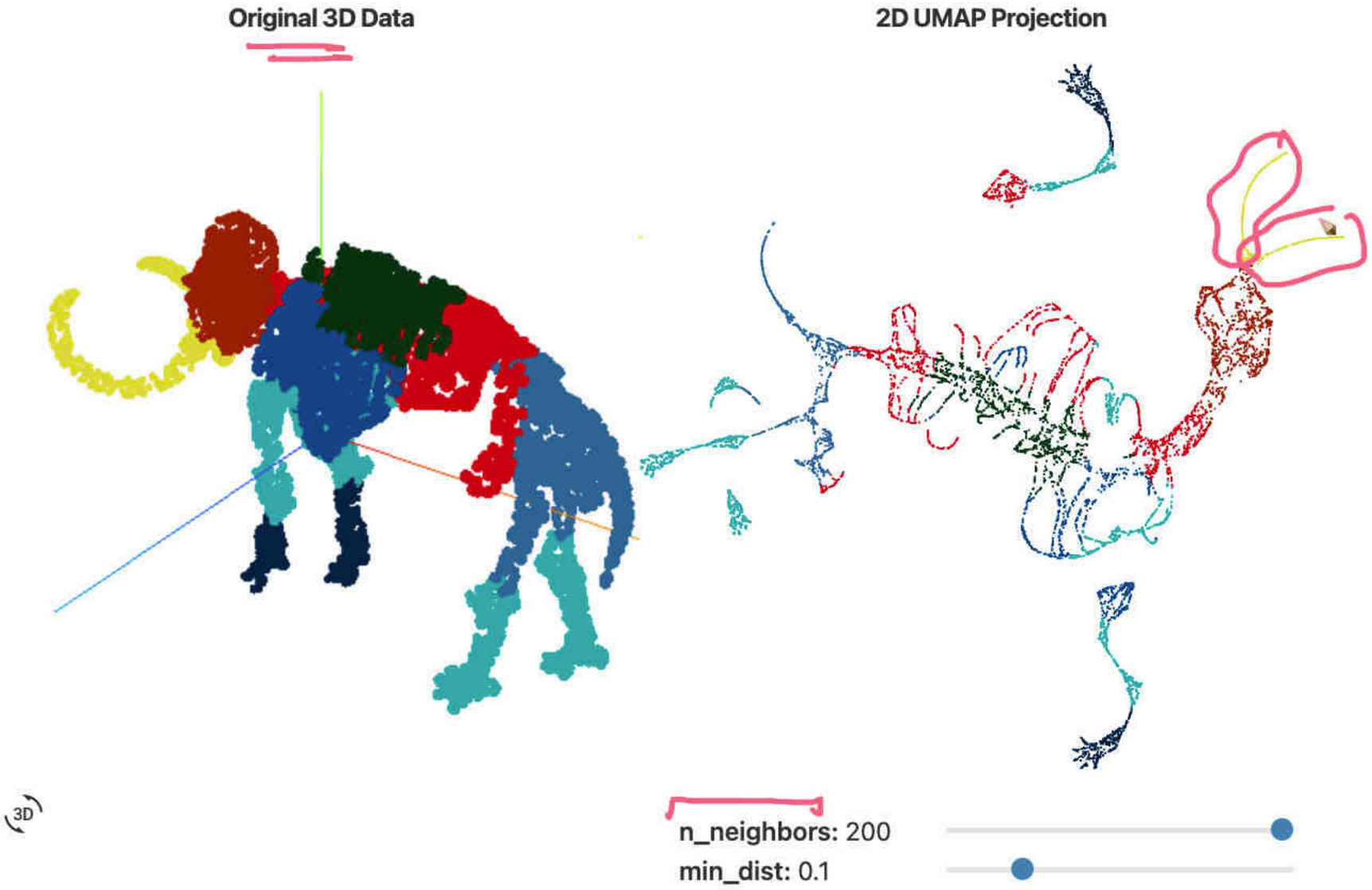


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with

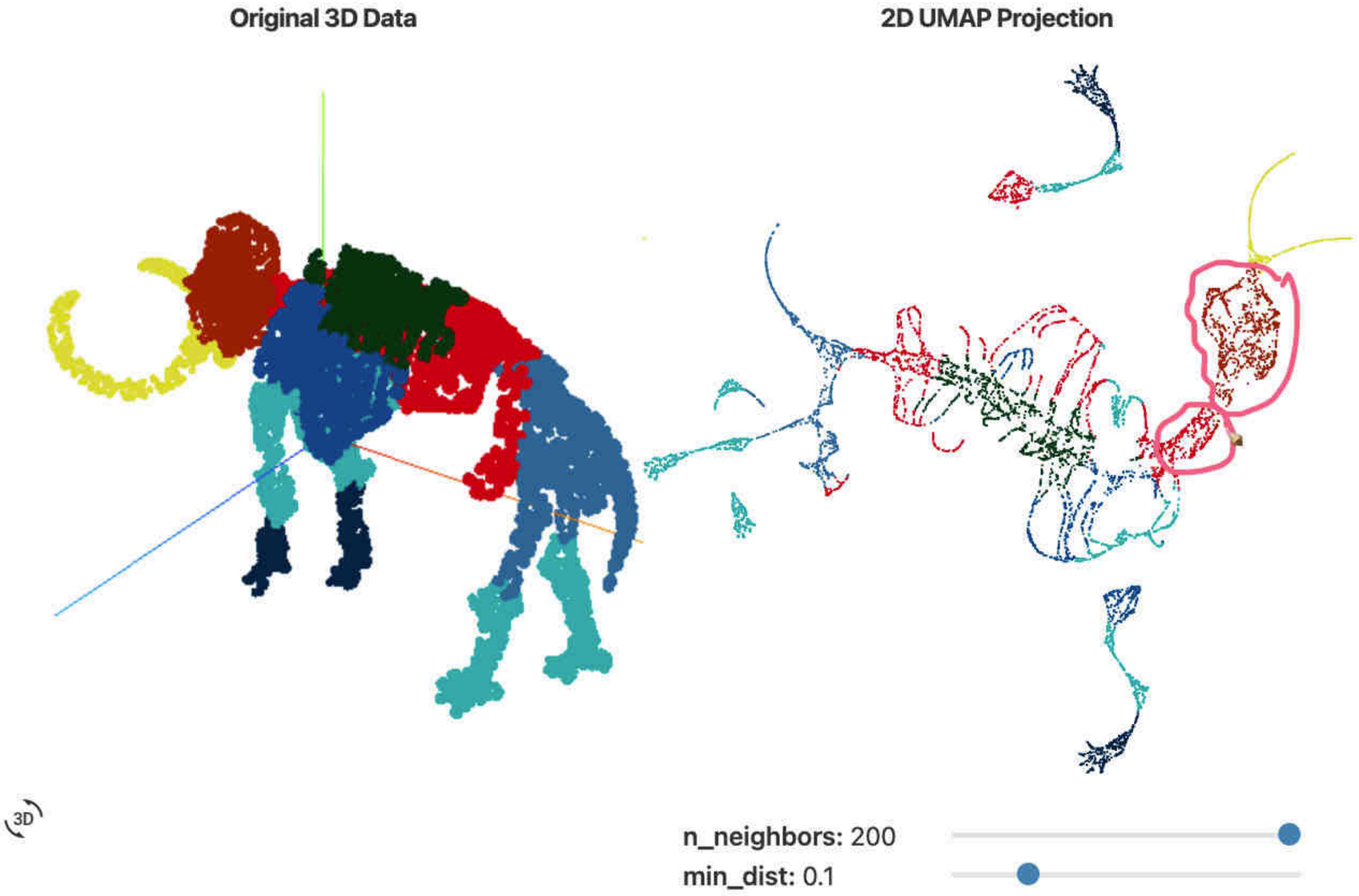


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with

would distract from our intent.

4.3 Hyper-parameters

As described in Algorithm 1, the UMAP algorithm takes four hyper-parameters:

1. n, the number of neighbors to consider when approximating the local metric;
2. d, the target embedding dimension;
3. min-dist, the desired separation between close points in the embedding space; and

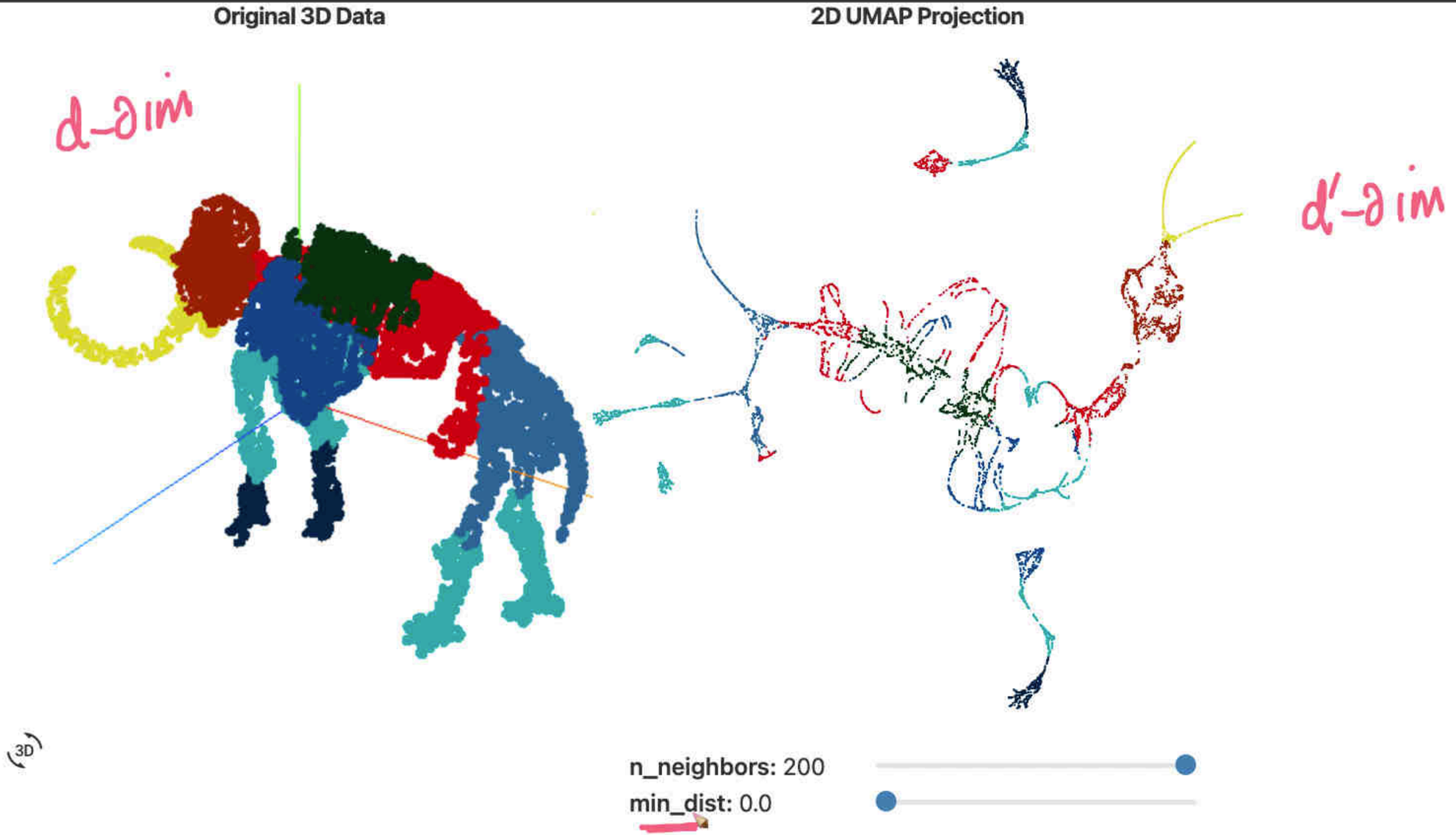


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` *and* `min_dist` *parameters.*

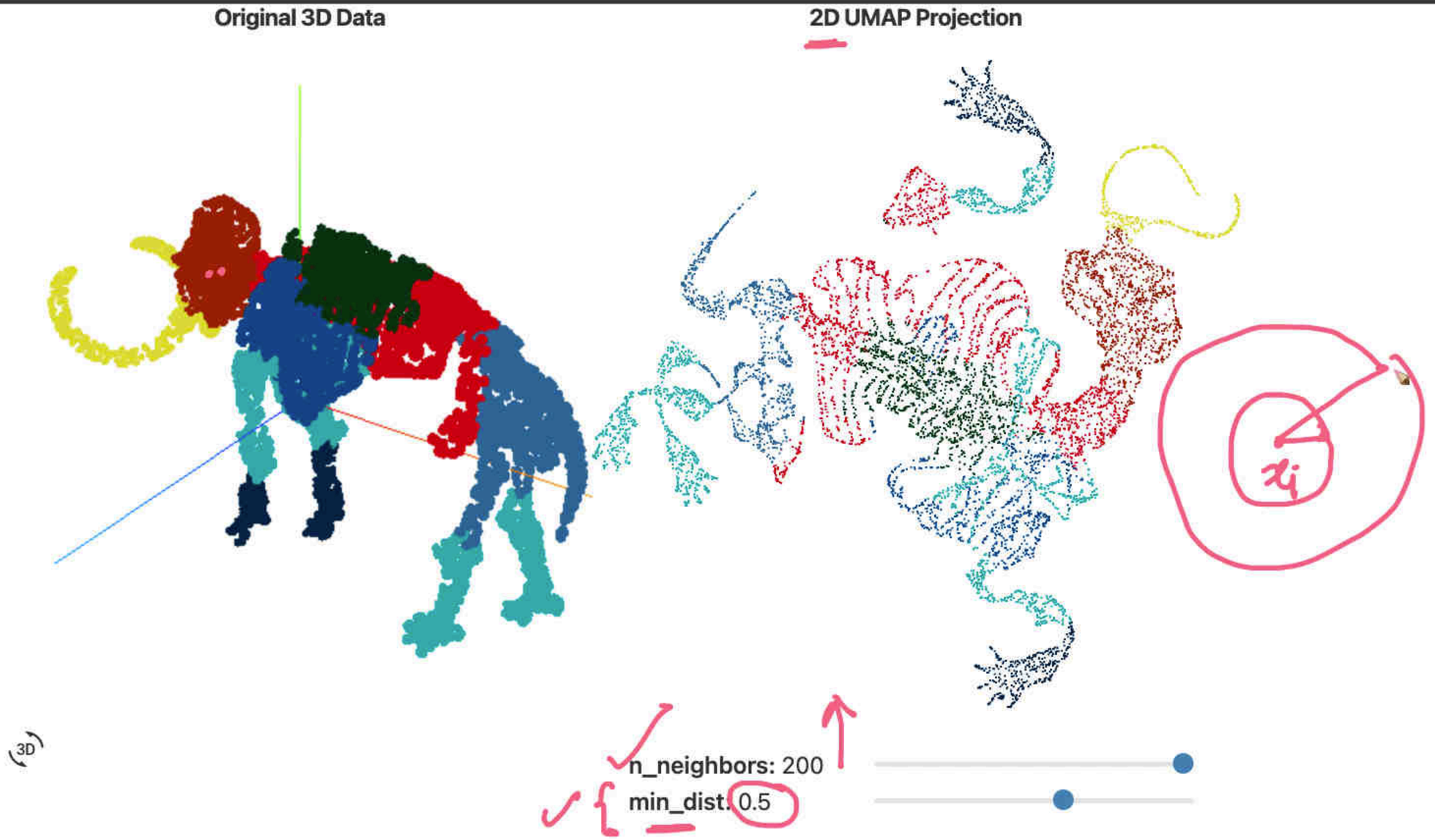


Figure 5: UMAP projections of a 3D woolly mammoth skeleton (50k points, 10k shown) into 2 dimensions, with various settings for the `n_neighbors` and `min_dist` parameters.

Q

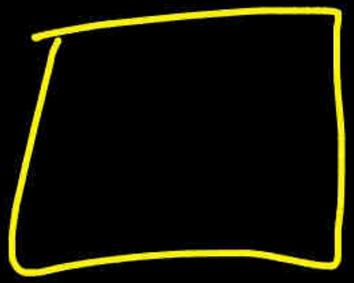
obj: Viz

d-3in

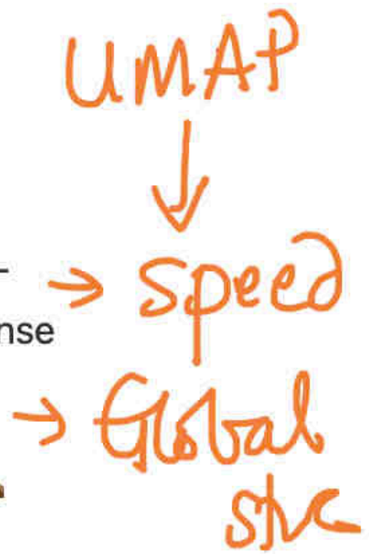
(d')

2in

3in



reading and understanding its results requires some care. It's worth revisiting our previous work on (mis)reading t-SNE, since many of the same takeaways apply to UMAP:



1. Hyperparameters really matter

Choosing good values isn't easy, and depends on both the data and your goals (eg, how tightly packed the projection ought to be). This is where UMAP's speed is a big advantage - By running UMAP multiple times with a variety of hyperparameters, you can get a better sense of how the projection is affected by its parameters.

2. Cluster sizes in a UMAP plot mean nothing

Just as in t-SNE, the size of clusters relative to each other is essentially meaningless. This is because UMAP uses local notions of distance to construct its high-dimensional graph representation.

3. Distances between clusters might not mean anything

Likewise, the distances between clusters is likely to be meaningless. While it's true that the global positions of clusters are better preserved in UMAP, the distances between them are not meaningful. Again, this is due to using local distances when constructing the graph.

4. Random noise doesn't always look random.

Especially at low v ved.

any given axis or distance in lower dimensions still isn't directly interpretable in the way of techniques such as PCA.

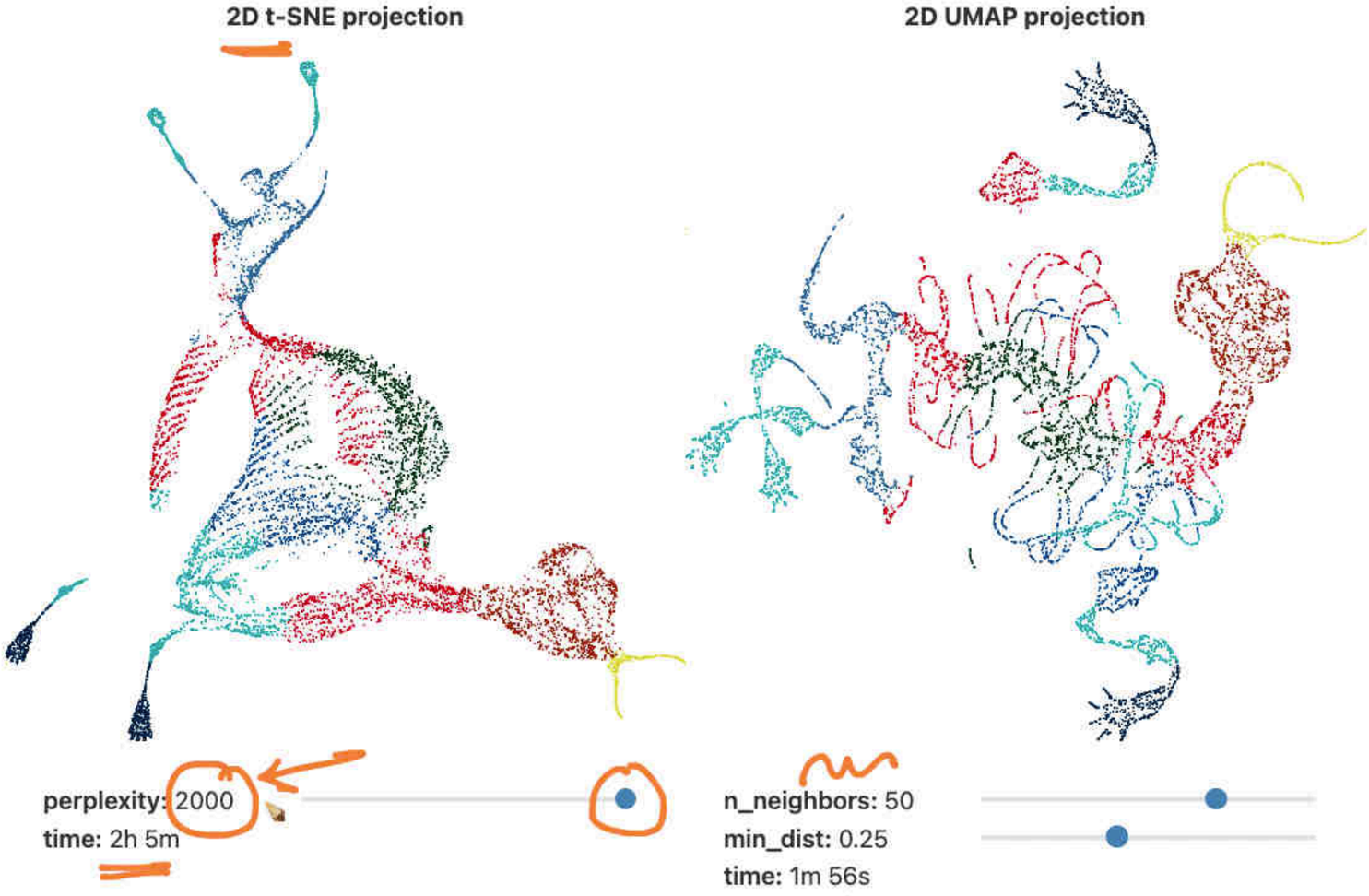


Figure 6: *A comparison between UMAP and t-SNE projections of a 3D woolly mammoth skeleton (50,000 points) into 2 dimensions, with various settings for parameters. Notice how much more global structure is preserved with UMAP, particularly with larger values of n_neighbors.*

How to (mis)read UMAP

While UMAP offers a number of advantages over t-SNE, it's by no means a silver bullet - and reading and understanding its results requires some care. It's worth revisiting our previous work on (mis)reading t-SNE, since many of the same takeaways apply to UMAP:

Speed & global-structure

1. Hyperparameters really matter

Choosing good values isn't easy, and depends on both the data and your goals (eg, how tightly packed the projection ought to be). This is where UMAP's speed is a big advantage - By running UMAP multiple times with a variety of hyperparameters, you can get a better sense of how the projection is affected by its parameters.

2. Cluster sizes in a UMAP plot mean nothing

Just as in t-SNE, the size of clusters relative to each other is essentially meaningless. This is because UMAP uses local notions of distance to construct its high-dimensional graph representation.

3. Distances between clusters might not mean anything

Likewise, the distances between clusters is likely to be meaningless. While it's true that the global positions of clusters are better preserved in UMAP, the distances between them are not meaningful. Again, this is due to using local distances when constructing the graph.

4. Random noise doesn't always look random.

Especially at low values of `n_neighbors`, spurious clustering can be observed.

5. You may n

tSNE

↓

large dist

let

2000

$\forall i,j \quad dist(x_i, x_j) \rightarrow P_{ij}$

UMAP

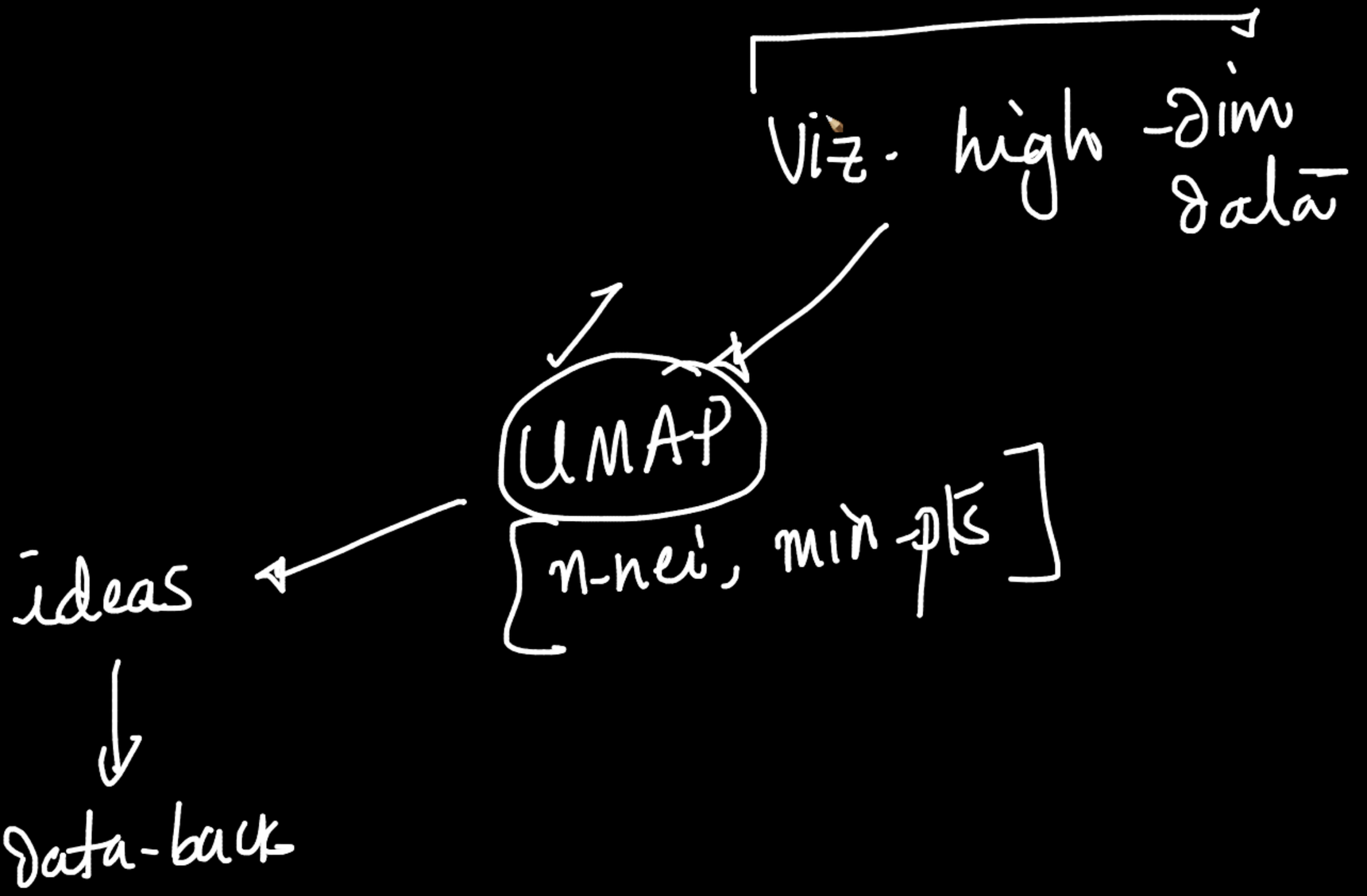
↓

opt { Small dist & large dist }

$dist(x_i, x_j) \quad x_j \in N(x_i)$

n_neigh 50

Takeaways:



local-str

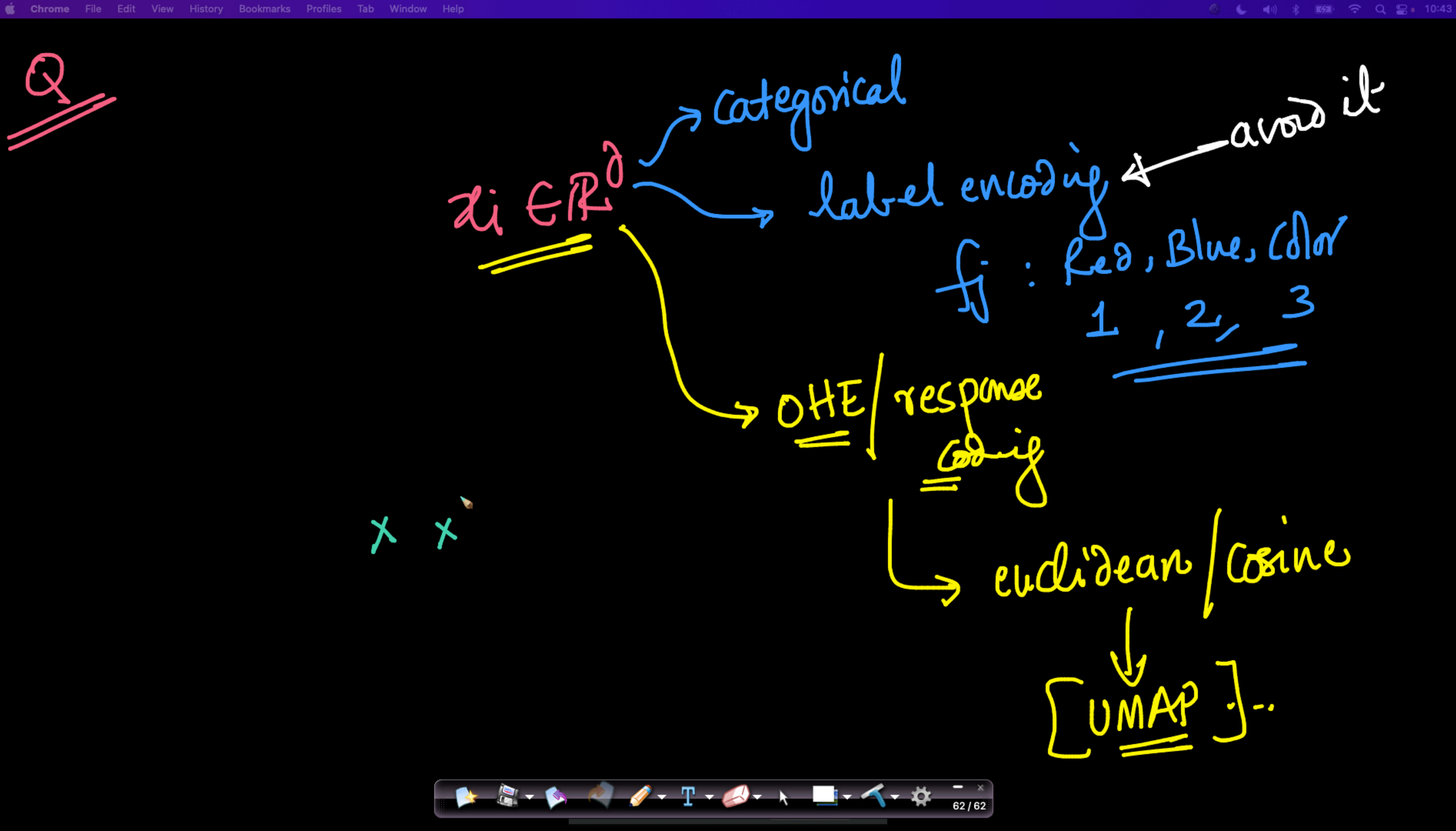
small-list
↳ large (rule)

global str

xi ---------- large list / small (rule)

xj

10:40


Q
$x_i \in R^d$
categorical
label encoding
avoid it
fj : Red, Blue, Color
1, 2, 3
OHE / response coding
euclidean / cosine
[UMAP]..
62 / 62

exercise

Text 1 : x_1 → politics

BoW

Text 2 : x_2 → politics

BoW

BoW

x_3 → Football

Cosine-Sim → UMAP

x_2

[RecSys]

Netflix; TikTok; Amzn; Youtube }

U-U, I-I
1. Sim-based
2. Content-based
3. Coll. Filtering

← Pre 2006

← 2007 - 2015

← 2015 +

MF

DL

Clustering
PCA
SVD
...

[Netflix $1M]

Formulation

User: u_i $\quad i = 1 \rightarrow n$ → Billions

Items: I_j (movie/product) $\quad j = 1 \rightarrow m$ → 100s of millions

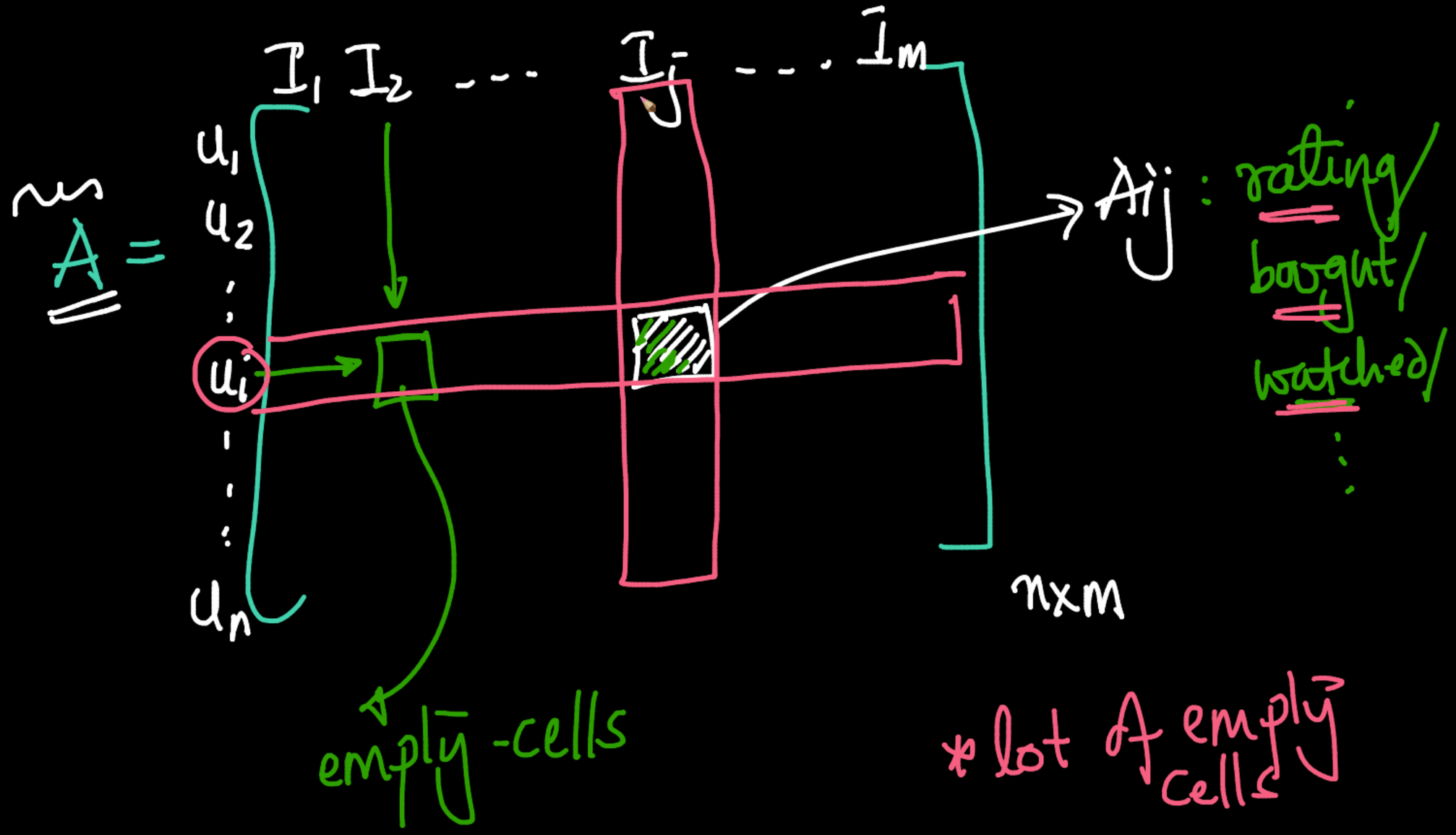
→ V. V. large

Task:

$u_i \rightarrow \checkmark I_{10}, I_{12}, I_{16}, I_{18}$

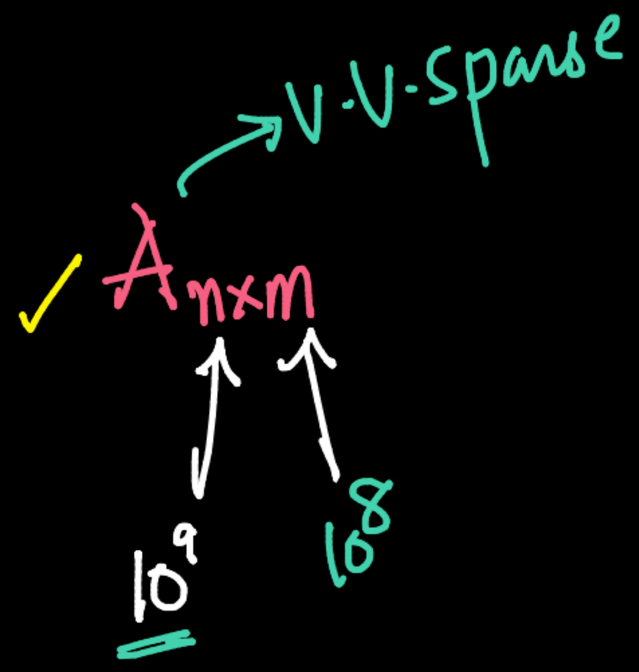
based on historical data

Dataset



historical data

* lot of empty cells



cells: V·V·huge

Sparsity: # non empty cells / total

$$\frac{10^9 \times 1000}{10^{17}} = \frac{10^{12}}{10^{17}}$$

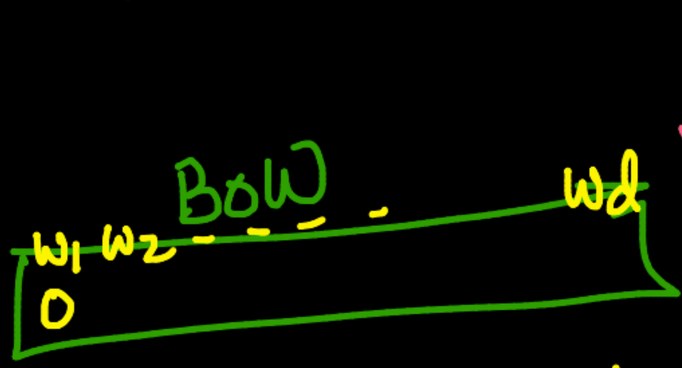
$$= \frac{1}{10^5} = 10^{-5}$$

Already learnt technique

Task: $u_i \rightarrow \in R^m$

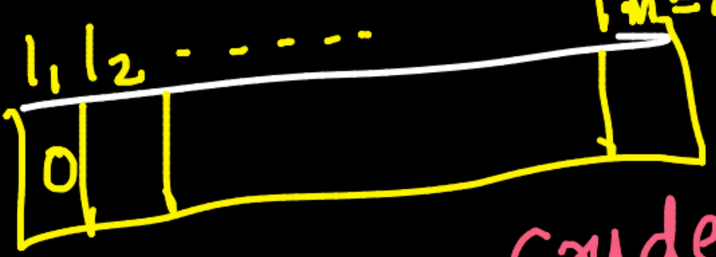
$u_i \rightarrow$ some items

Text: BoW



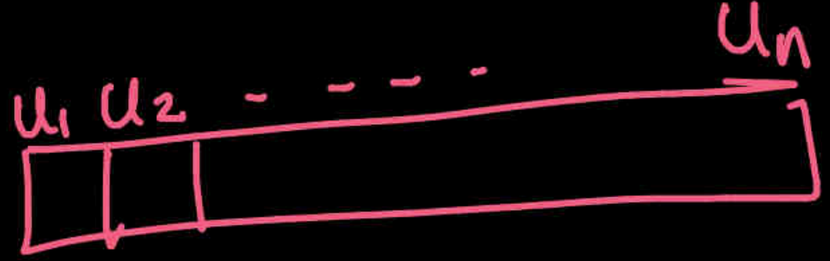
Represent u_i → ?

user:



Code

Represent I_j

I_j : 

$\in \mathbb{R}^n$

Very Crude

Given $A_{n \times m}$

Task: $u_i \longrightarrow$ recommend some items

✓ **idea1:** $u_i \rightarrow \widetilde{I_{10}, I_{12}}$ (already bought)

Item-Item sim → $\forall j$

[Nearest Neigh]

$sim(I_j, I_{10})$; $sim(I_j, I_{12})$ is large

↳ cosine-sim

Idea 2:

u_i → already bought I_{10}, I_{18}

u_j $sim(u_j, u_i)$ is large

Nearest neigh ⇒ u_j

User-User

(let) u_0, u_{26}, u_{58} (most similar)

u_0	u_{26}	u_{58}
~~I_{10}~~	~~I_{10}~~	
I_{12}	~~I_{18}~~	
~~I_{18}~~	I_{24}	
I_{20}	I_{12}	

frequency bow

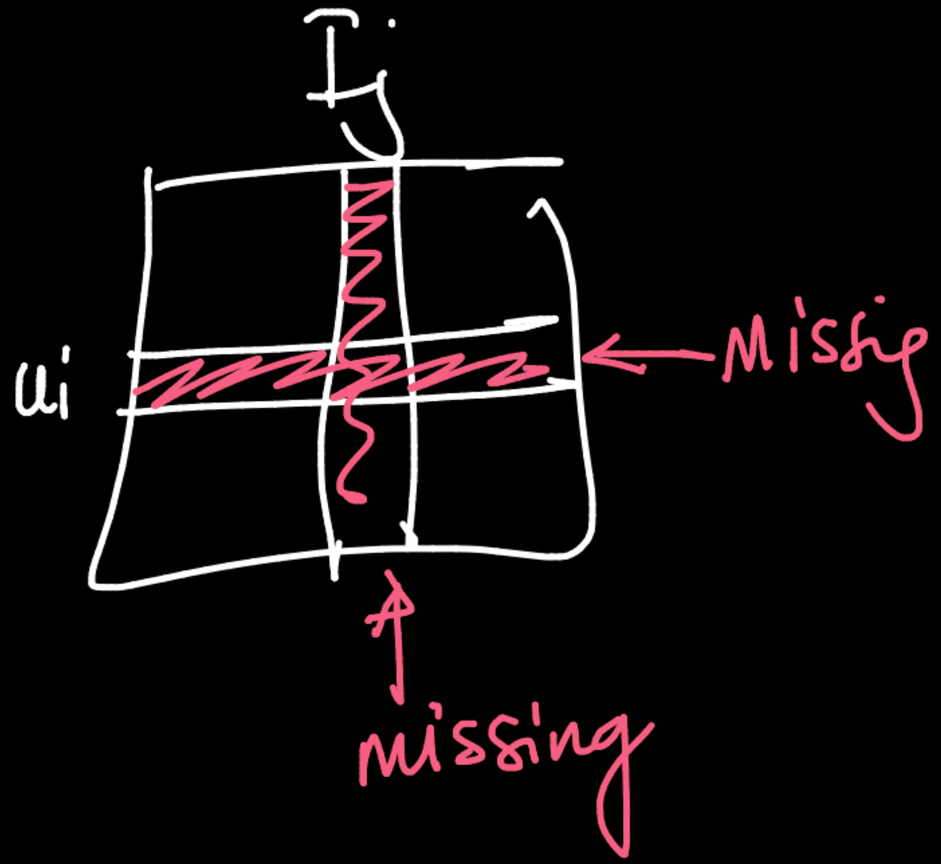
I_{12}

COLD-START

recommend items

u_i : new-user

I_j : new-item



idea 1: frequently bought items (or) popular

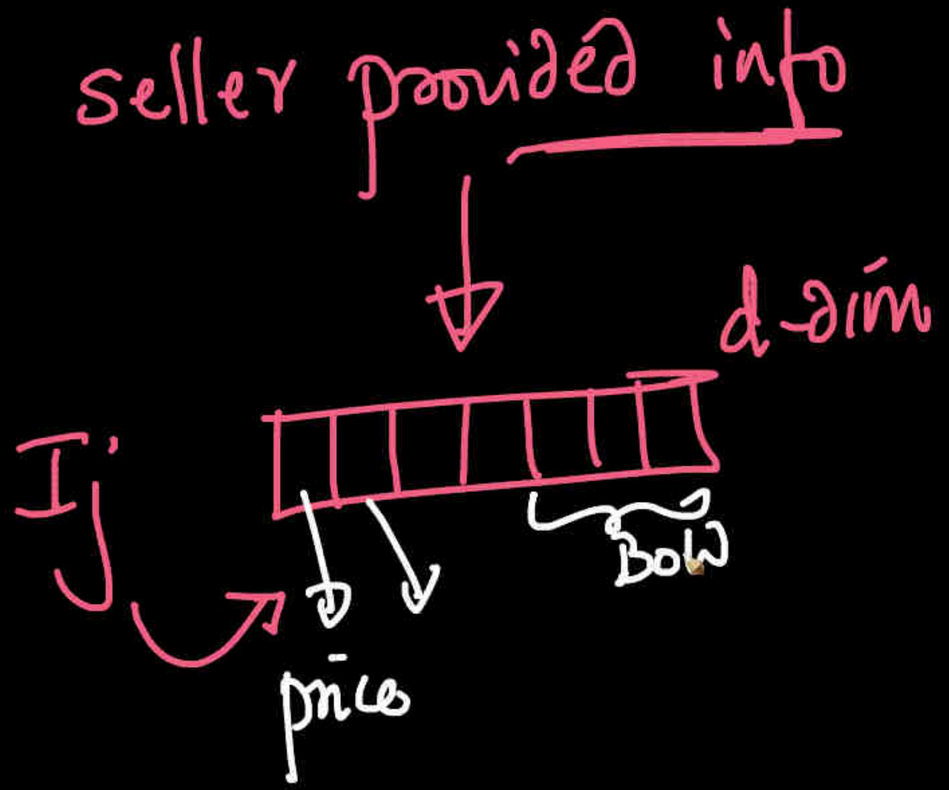
additional data u_i → location (GPS/IP...)

popular by location; gender; age

credit-card

$u_i \in \mathbb{R}^d$

Item j is new ⟶ seller provided info



Find U-U or I-I similarities

Using additional features

Content-based RecSys

$u_i \in \mathbb{R}^{d'}$

$I_j \in \mathbb{R}^{d}$

provided/extracted features

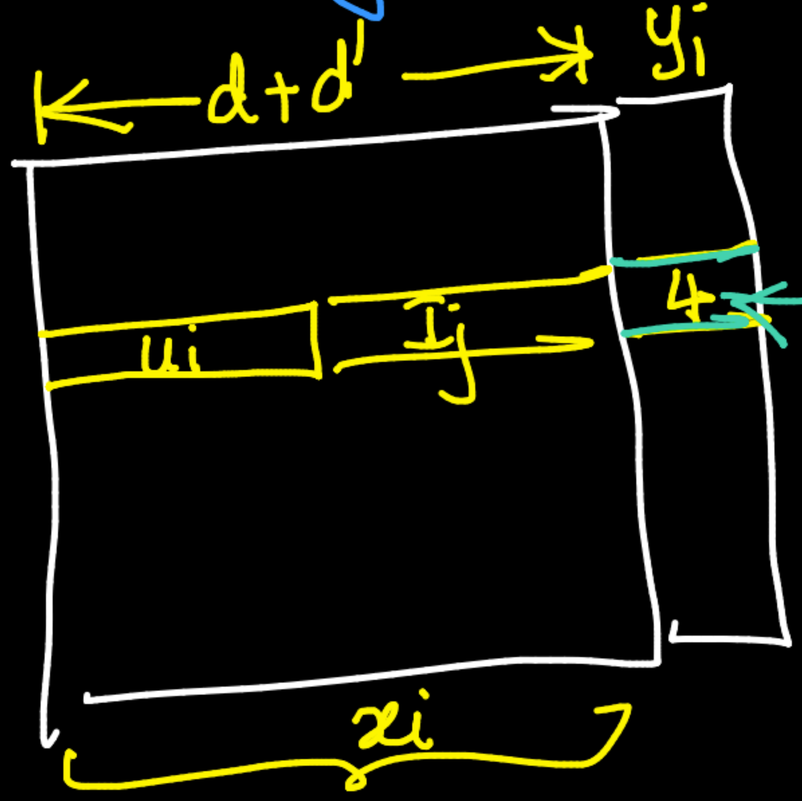
Q

Idea:

content-based provided features

$u_i \in R^{d'}$; $I_j \in R^d$

Regr / Classfn

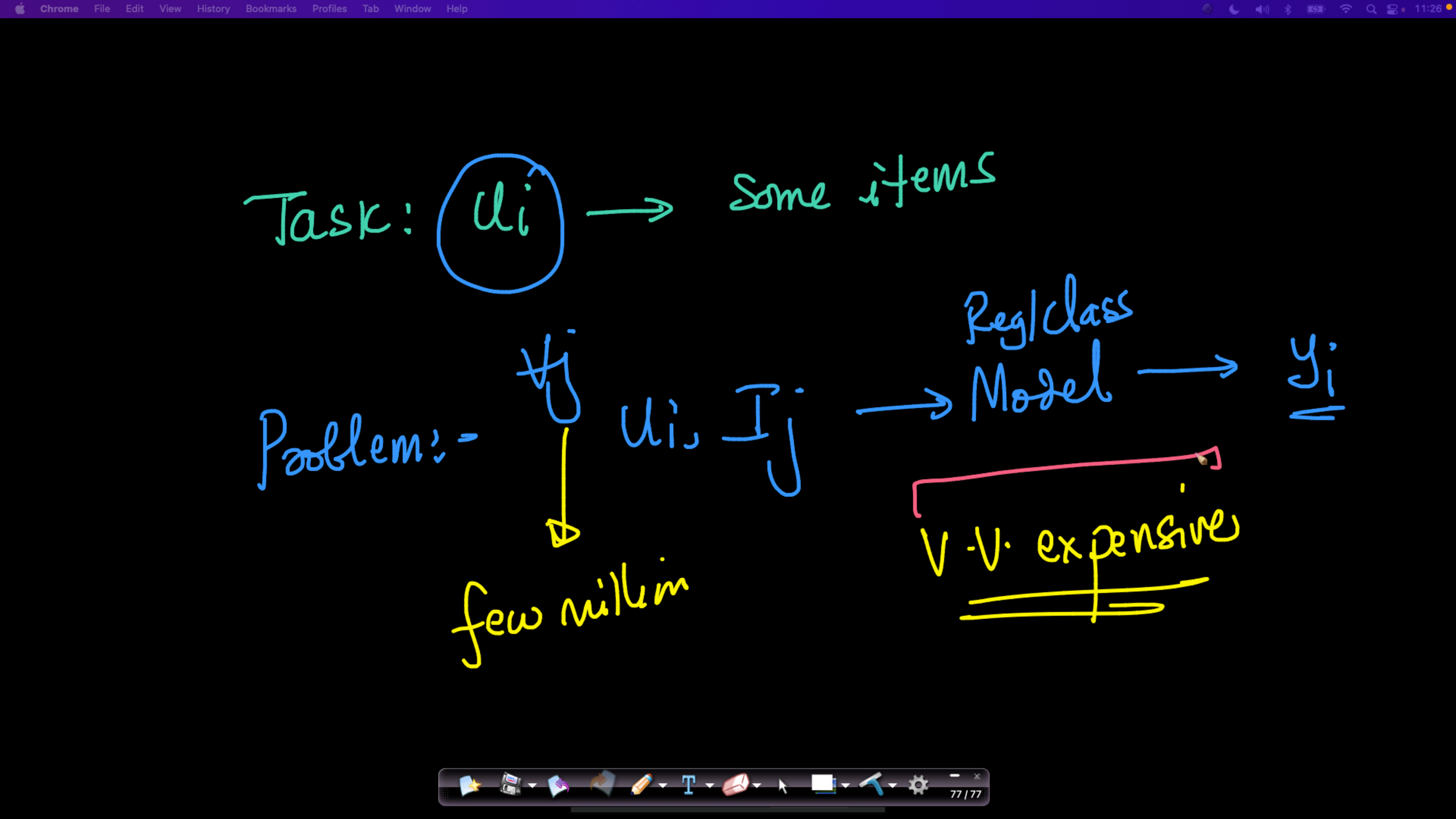


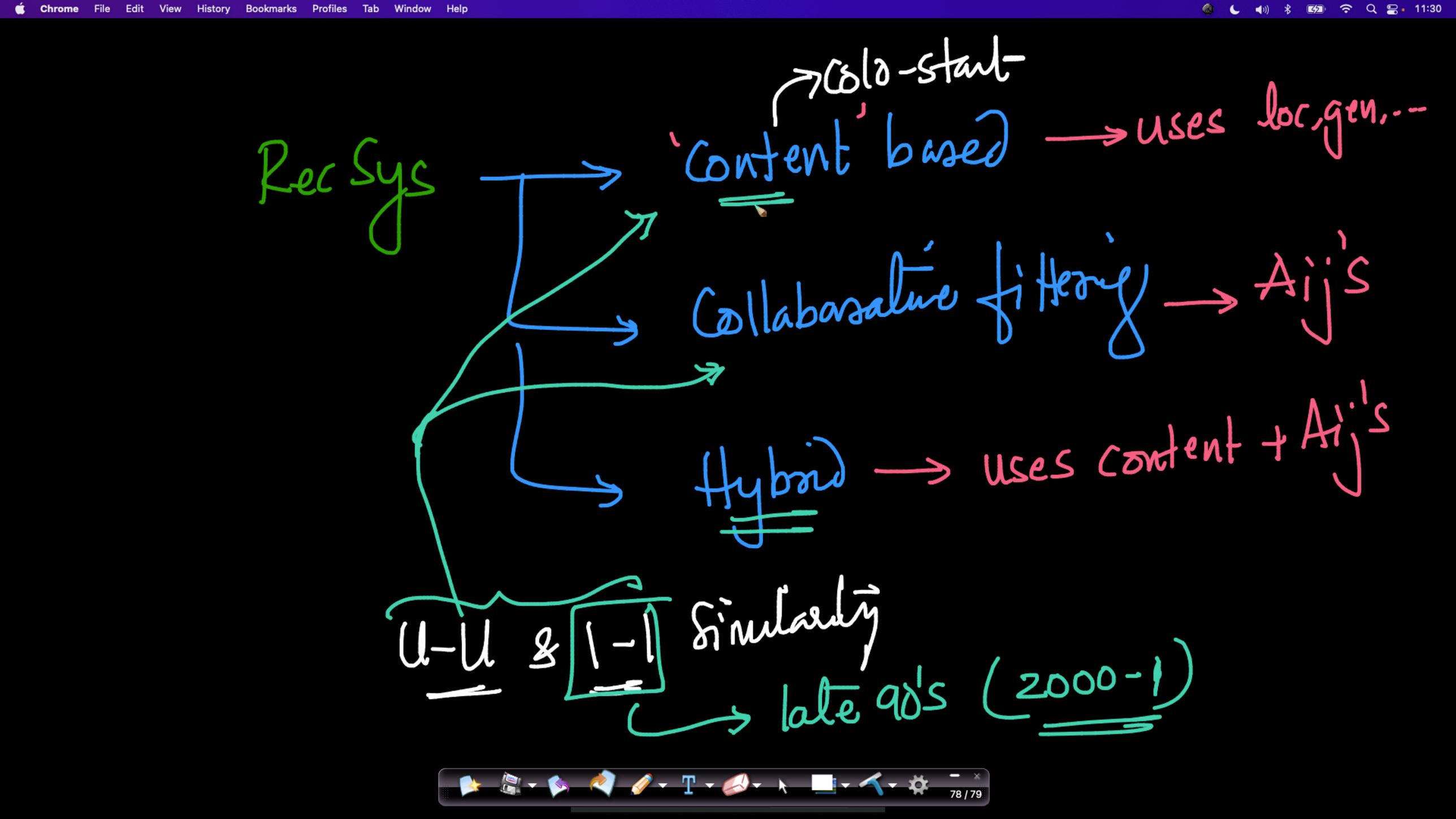
list-data

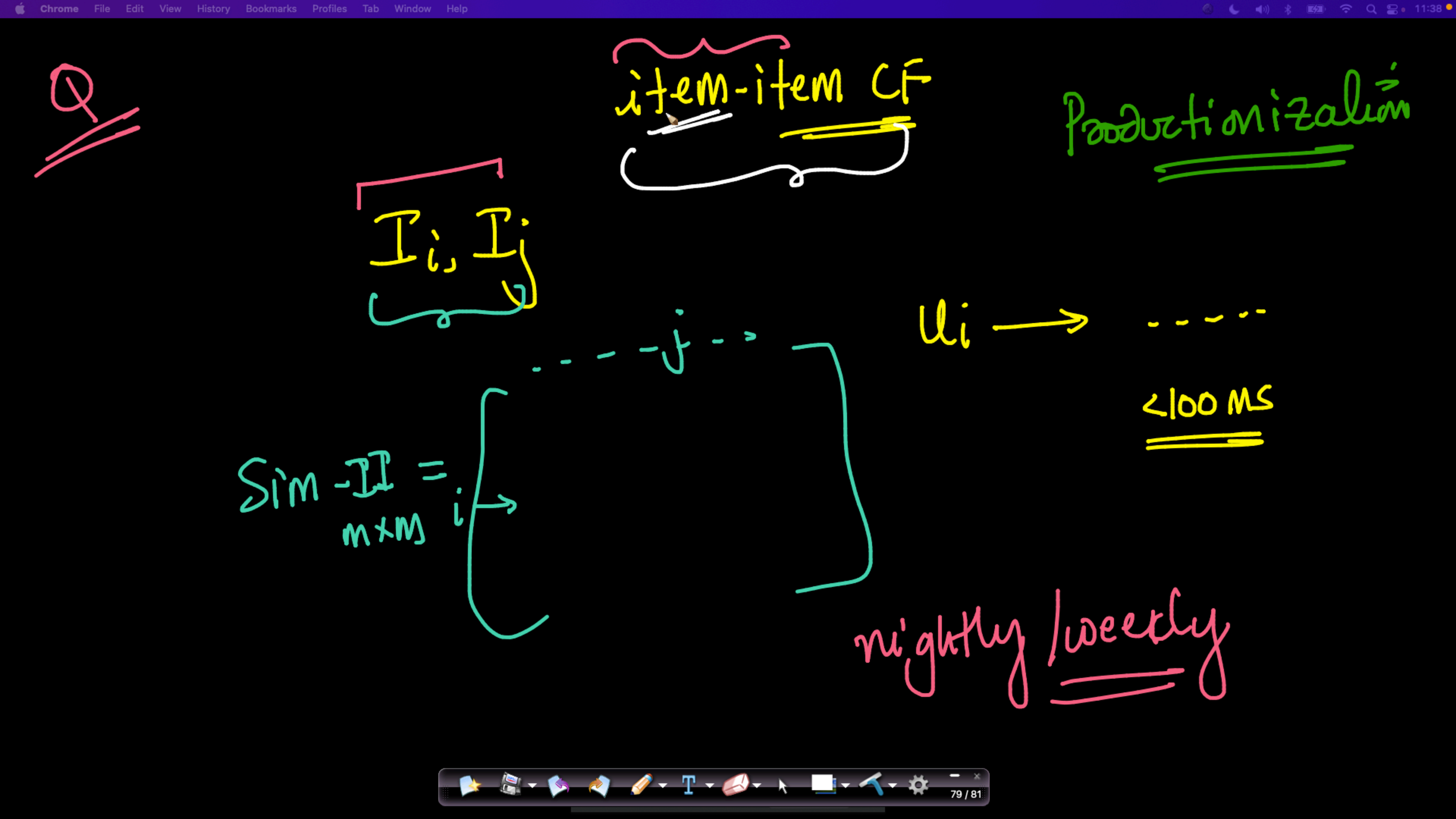
$A_{ij} = 4$

$u_i, I_j \rightarrow 4$

u_{1000}, I_{111} y_i


Task: Ui → Some items
Problem:- ∀j Ui, Ij → Reg/class Model → yi
few millions
V.V. expensive

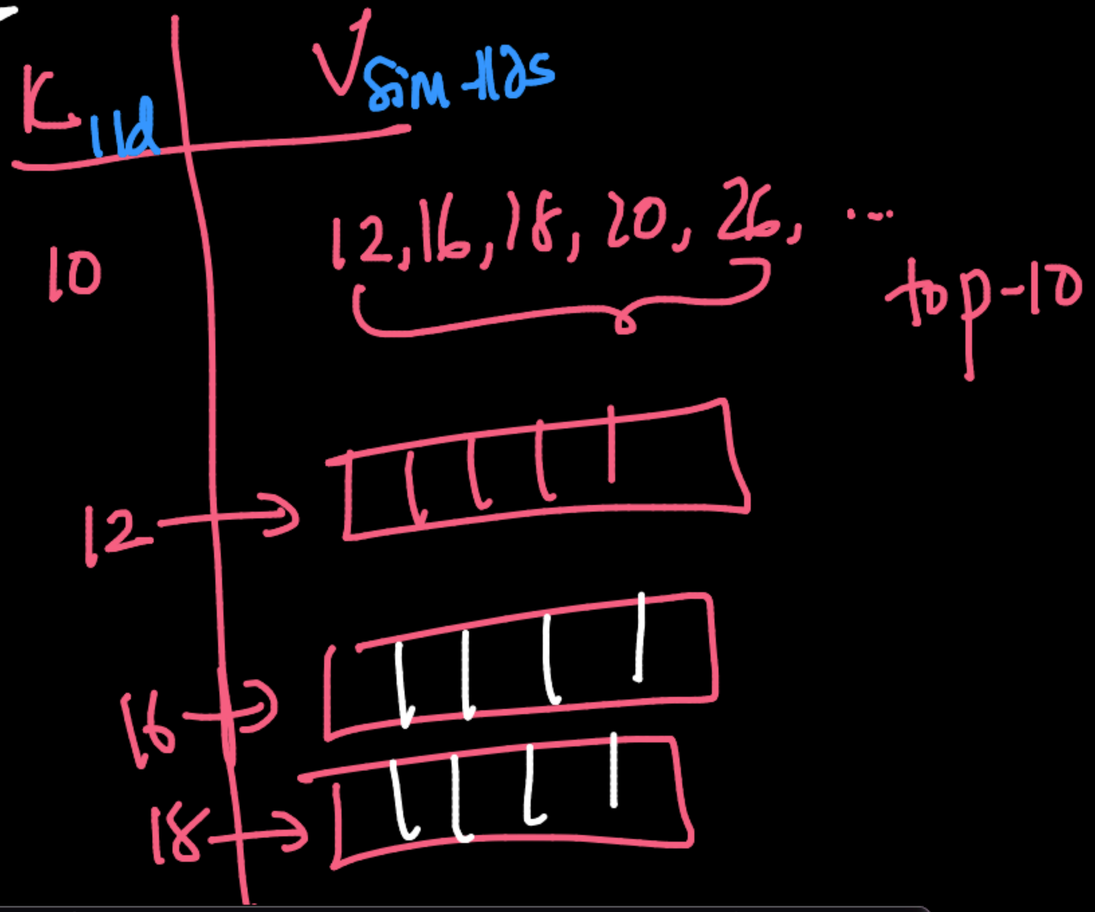

Rec Sys
'Content' based
→ Cold-start
→ uses loc, gen, ...
Collaborative filtering
→ Aij's
Hybrid
→ uses content + Aij's
U-U & I-I
Similarity
→ late 90's (2000-1)


item-item CF
Productionization
I_i, I_j
Sim-II
m x m
i
j
Ui →
<100 MS
nightly/weekly

Redis (disb-dict) & Memcache

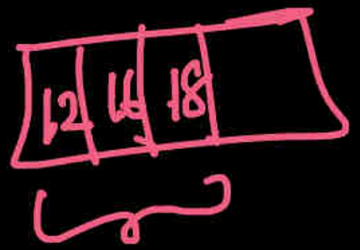
→ dict - 11 sim

→ FAANG



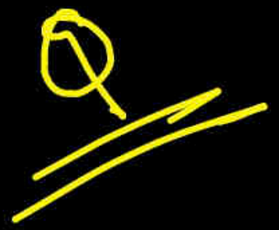
O(1)

dict - u-1

K uid	V list of likes
u_{10}	

u_{10}

O(1)



Netflix

PCA_tSNE_UMAP.ipynb - Colab | Amazon.com recommendation | LIVE: Republic Day Parade - 2

youtube.com/watch?v=eUWfAyNqTI0

YouTube IN

Search

#RepublicDay #RepublicDayParade2022 #RepublicDay2022

LIVE: Republic Day Parade - 2022

6,303,585 views • Streamed live on Jan 26, 2022

94K DISLIKE SHARE CLIP SAVE ...

President of India
2.18M subscribers

SUBSCRIBE

83 / 83

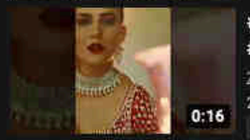
Live chat replay was turned off for this video.

All | Republic Day | Indian Army | Troops

What is Indira Gandhi's connection with Jaigarh Fort's...
StudyIQ IAS
2.9M views • 1 year ago

जलवायु परिवर्तन अब वाद-विवाद का विषय नहीं रह गया है
President of India
12K views • 13 days ago

Maa Tujhe Salaam | Daredevil Bike Stunts | Stunning Air...
Music India Entertainment
11M views • 2 years ago



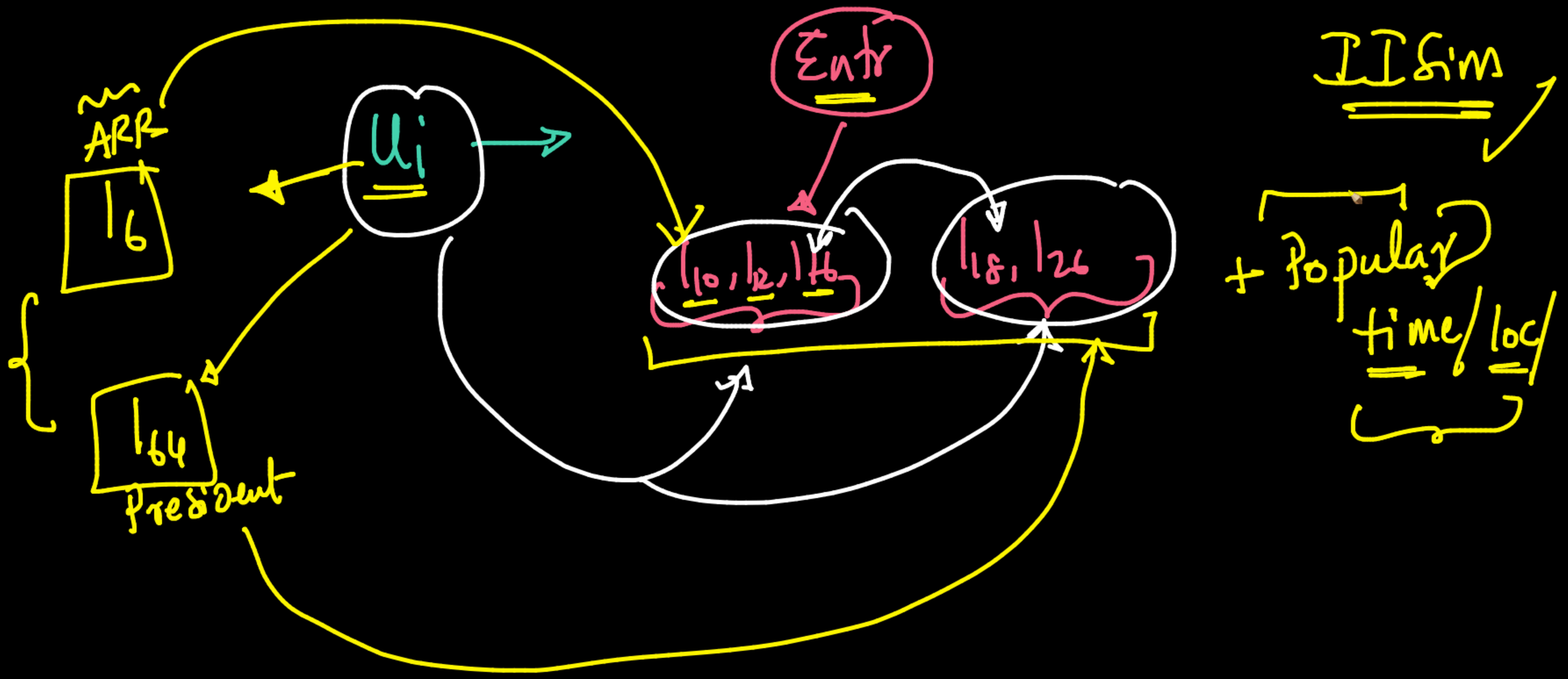
#EkThumkeSe #LoveYouLoktantra |...
Zee Music Company
28K views • 13 hours ago
New

कारगिल: वेलर & विक्ट्री हिंदी एपिसोड | Kargil: Valour & Victory Hindi...
HISTORY TV18
468K views • 10 days ago

Namo Namo - Lyrical | Kedarnath | Sushant Rajput |...
Zee Music Company
358M views • 2 years ago

The Pursuit Of Happyness: Job interview
Binge Society
10M views • 1 year ago

Munna vs Pandits | Mirzapur |


ARR
16
Ui
Entr
I10, I12, I16
I18, I24
I64
President
IISim
+ Popular
time/loc

Chrome File Edit View History Bookmarks Profiles Tab Window Help

PCA_tSNE_UMAP.ipynb - | Amazon.com recommend | LIVE: Republic Day Parade | Amazon.in : tshirt | Buy CHKOKKO Men's Gym | Amazon.in : gift | Amazon.in : rakhi gift | Buy Collectible India Rakh

amazon.in/CHKOKKO-Regular-Sleeves-T-Shirts-Seagreen/dp/B0856V6J1G/ref=sr_1_9?crid=2G0FEI6TC20N1&keywords=t-shirt&qid=1659723434&sprefix=tshir%2Caps%2C226&sr=8-9

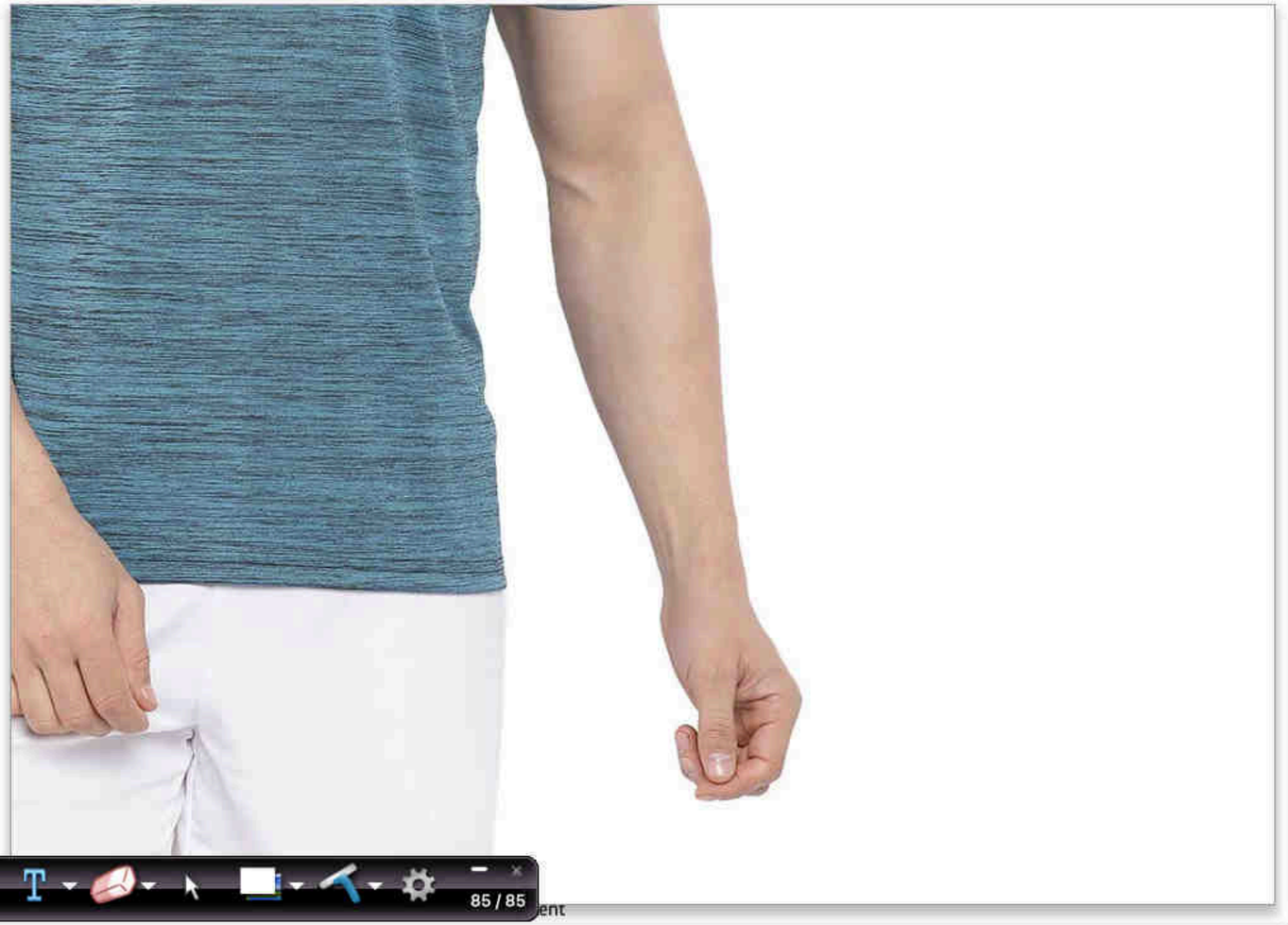
amazon.in Hello Select your address | All t-shirt | Hello, Sign in Account & Lists | Returns & Orders | 0 Cart

All Best Sellers Mobiles Today's Deals Customer Service Books Electronics Prime Fashion New Releases Home & Kitchen Amazon Pay Computers Coupons Toys & Games Sell PRIME EARLY ACCESS LIVE NOW

Amazon Fashion Women Men Kids Bags & Luggage Sportswear Sales & Deals 30 DAY RETURNS Restrictions Apply

Amazon Brand - Symbol Men's Regular Polo Shirt 6,536 ₹299.00

‹ Back to results Sponsored



85 / 85