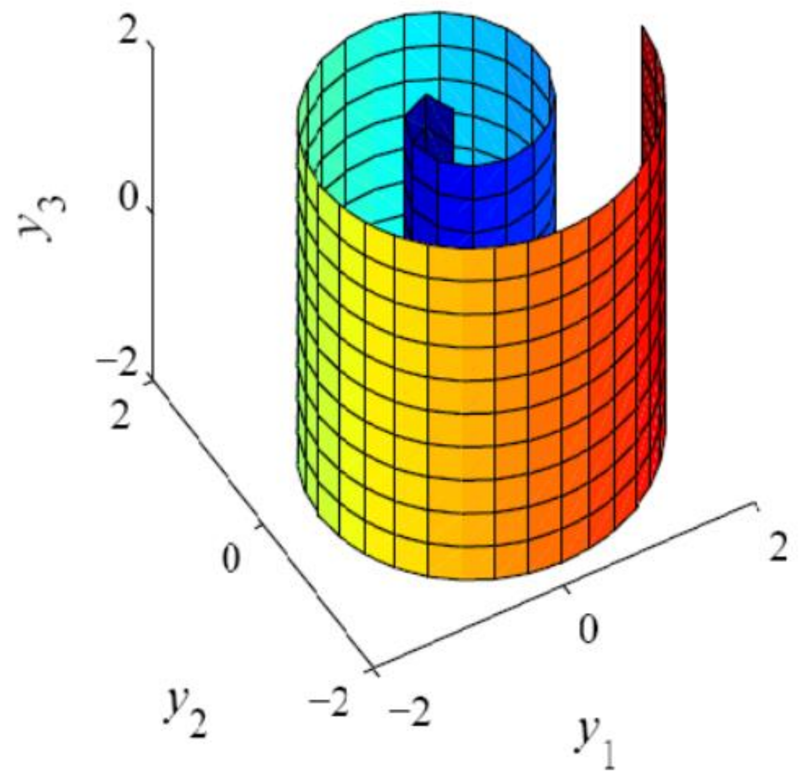
Dimensionality Reduction

Yu-Shuen Wang, CS, NYCU

Dimensionality Reduction is a helpful tool for visualization

- Dimensionality reduction algorithms
  - Map high-dimensional data to a lower dimension
  - While preserving structure
- They are used for
  - Visualization
  - Performance
  - Curse of dimensionality
- A ton of algorithms exist
  - PCA is a linear algorithm that works well in many applications
  - t-SNE is specialised for visualization and has gained a lot of popularity

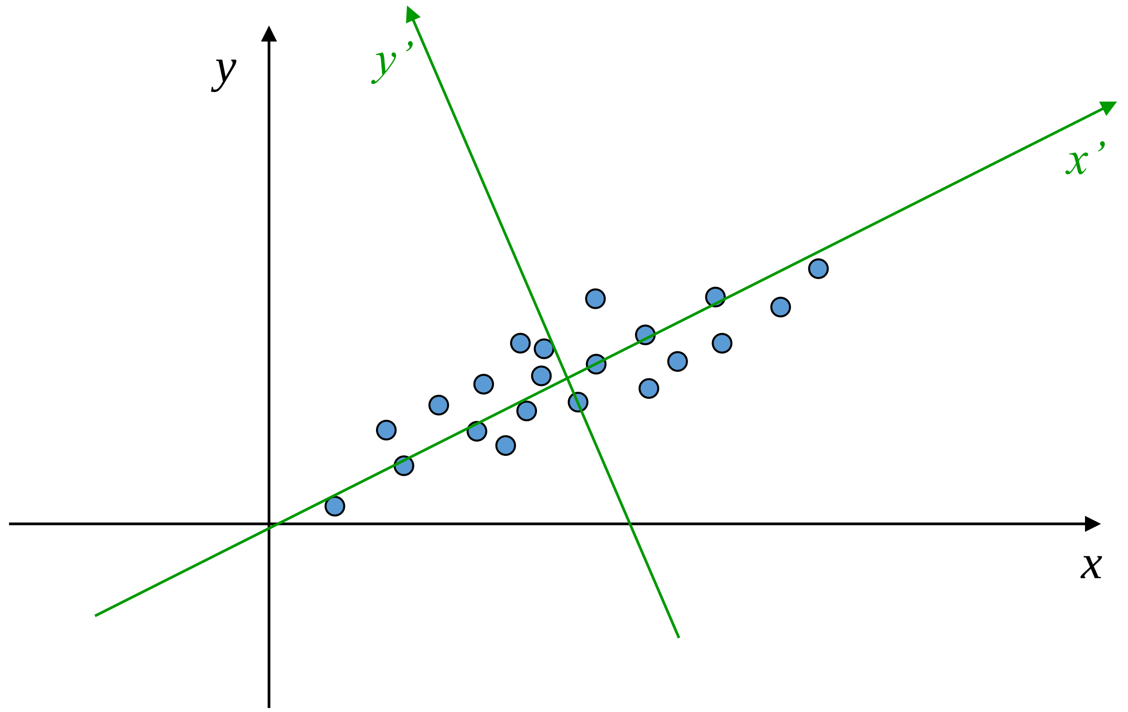


Principal component analysis

The slides are from Prof. Olga Sorkine, ETH Zurich.

PCA – the general idea

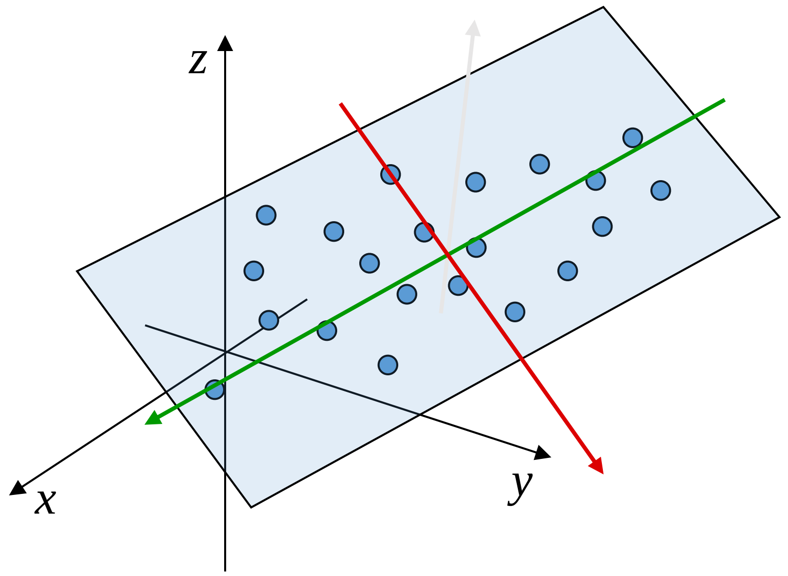
- PCA finds an orthogonal basis that best represents given data set.



- The sum of distances^2 from the x' axis is minimized.

PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.

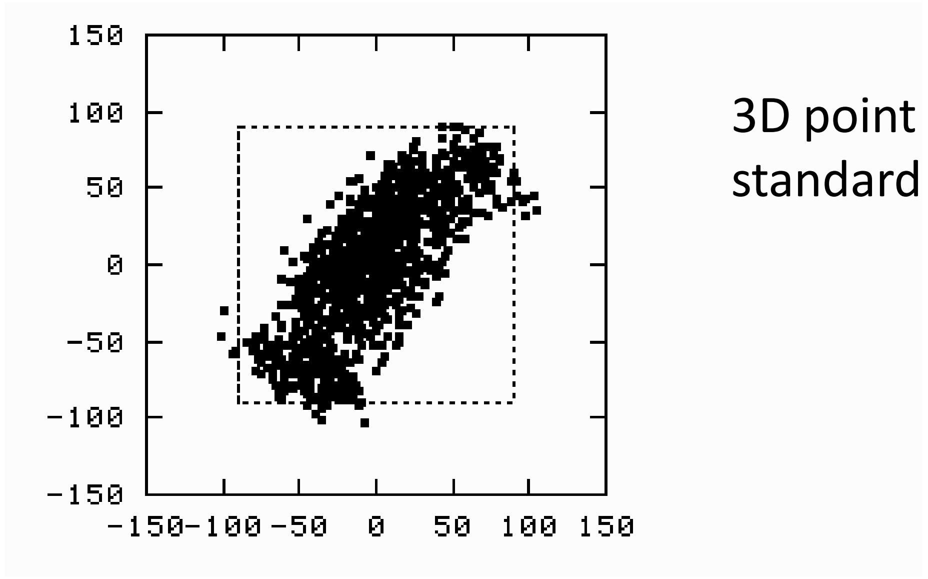


3D point set in standard basis

- PCA finds a best approximating plane (again, in terms of $\Sigma distances^2$)

PCA – the general idea

- PCA finds an orthogonal basis that best represents given data set.

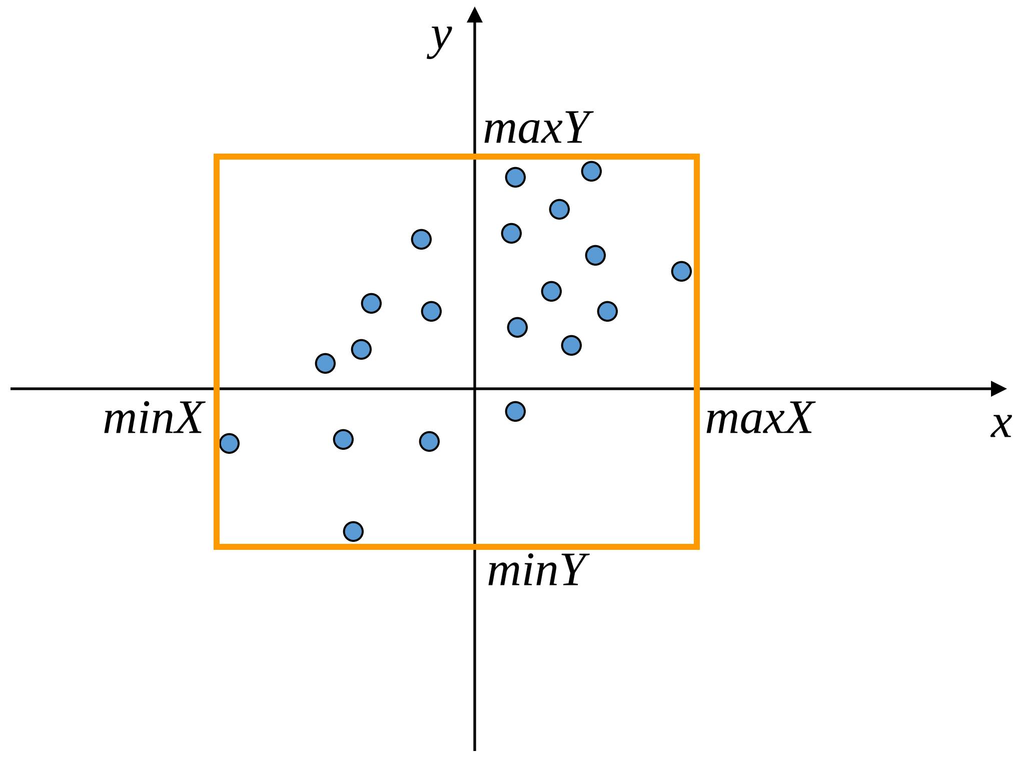


3D point set in standard basis

- PCA finds a best approximating plane (again, in terms of $\Sigma distances^2$)

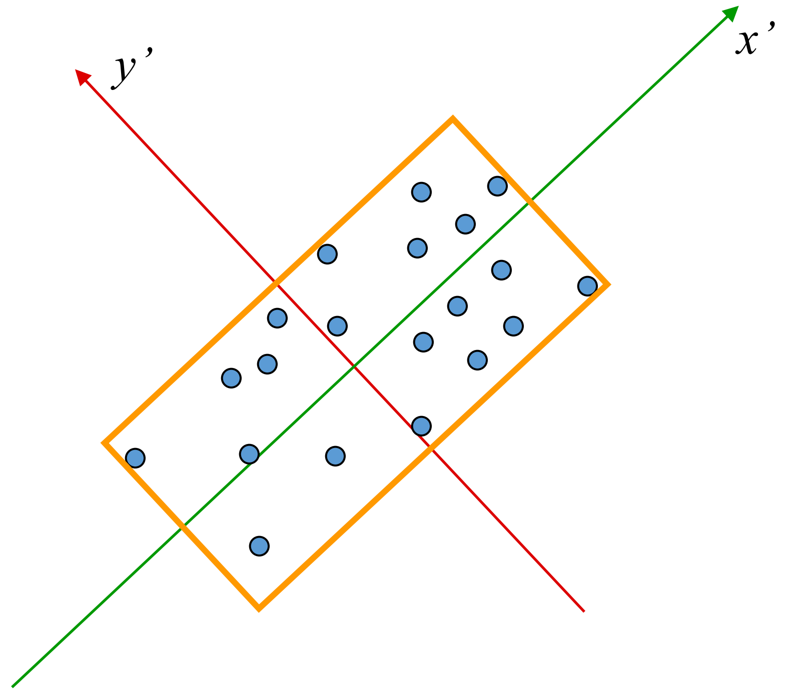
Application: finding tight bounding box

- An axis-aligned bounding box: agrees with the axes



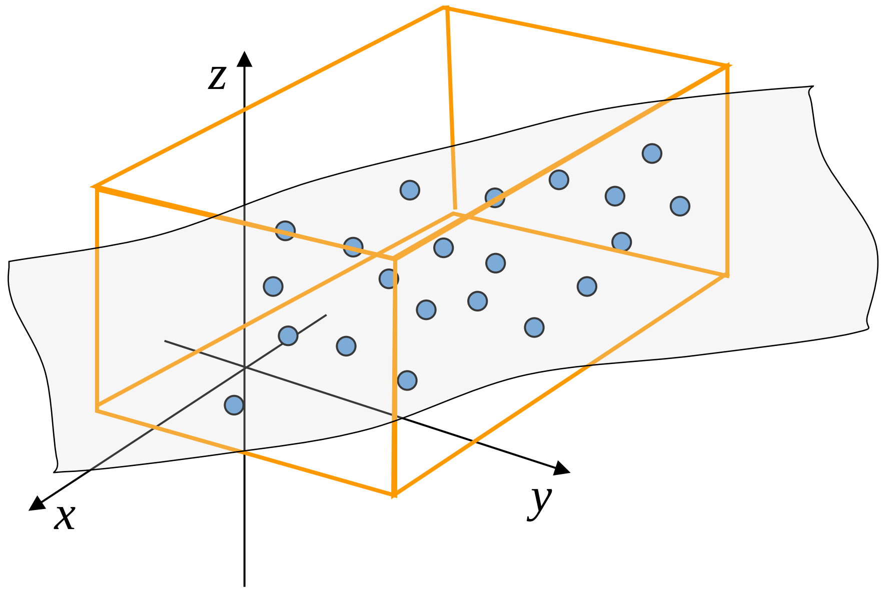
Application: finding tight bounding box

- Oriented bounding box: we find better axes!



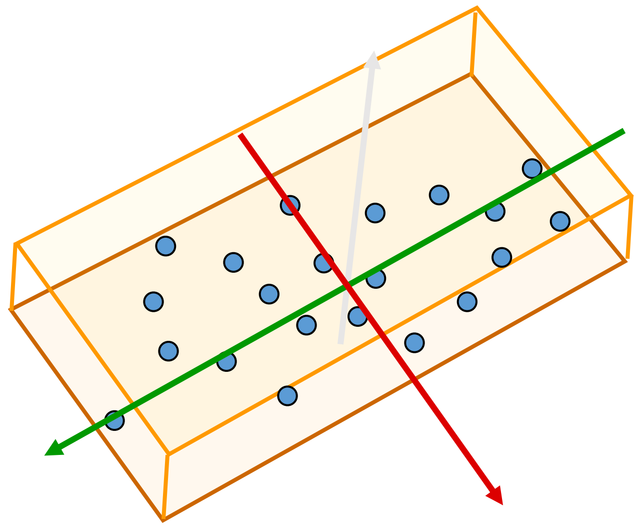
Application: finding tight bounding box

- This is not the optimal bounding box



Application: finding tight bounding box

- Oriented bounding box: we find better axes!



Notations

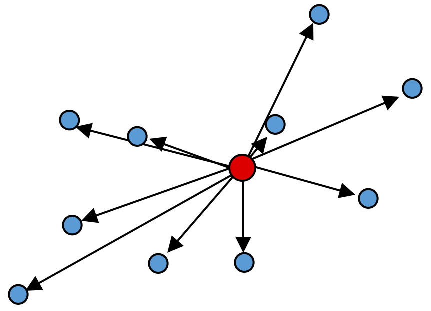
- Denote our data points by $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \in R^d$

$$\mathbf{x_1} = \begin{pmatrix} x_1^1 \\ x_1^2 \\ \vdots \\ x_1^d \end{pmatrix}, \ \mathbf{x_2} = \begin{pmatrix} x_2^1 \\ x_2^2 \\ \vdots \\ x_2^d \end{pmatrix}, \ \ldots, \ \mathbf{x_n} = \begin{pmatrix} x_n^1 \\ x_n^2 \\ \vdots \\ x_n^d \end{pmatrix}$$

The origin of the new axes

- The origin is zero-order approximation of our data set (a point)
- It will be the center of mass:

$$\boxed{\mathbf{m} = \tfrac{1}{n}\sum_{i=1}^{n}\mathbf{x}_i}$$



- It can be shown that:

$$\boxed{\mathbf{m} = \underset{\mathbf{x}}{\operatorname{argmin}}\sum_{i=1}^{n}\left\|\mathbf{x}_i - \mathbf{x}\right\|^2}$$

Scatter matrix

- Denote $\mathbf{y}_i = \mathbf{x}_i - \mathbf{m}$, $\quad i = 1, 2, \ldots, n$

$$\boxed{S = YY^T}$$

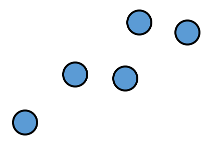
where Y is $d \times n$ matrix with $\mathbf{y}_k$ as columns ($k = 1, 2, \ldots, n$)

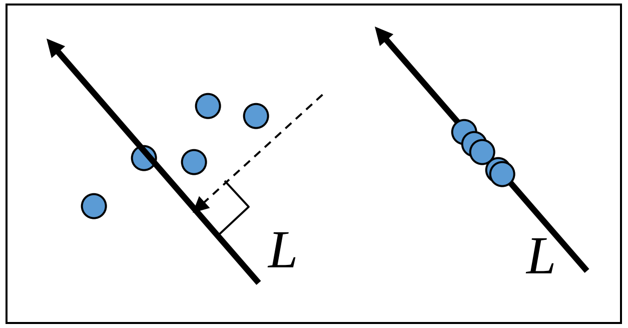
$$S = \underbrace{\begin{pmatrix} y_1^1 & y_2^1 & \cdots & y_n^1 \\ y_1^2 & y_2^2 & & y_n^2 \\ \vdots & \vdots & & \vdots \\ y_1^d & y_2^d & \cdots & y_n^d \end{pmatrix}}_{Y} \underbrace{\begin{pmatrix} y_1^1 & y_1^2 & \cdots & y_1^d \\ y_2^1 & y_2^2 & \cdots & y_2^d \\ \vdots & & & \vdots \\ y_n^1 & y_n^2 & \cdots & y_n^d \end{pmatrix}}_{Y^T}$$

Variance of projected points

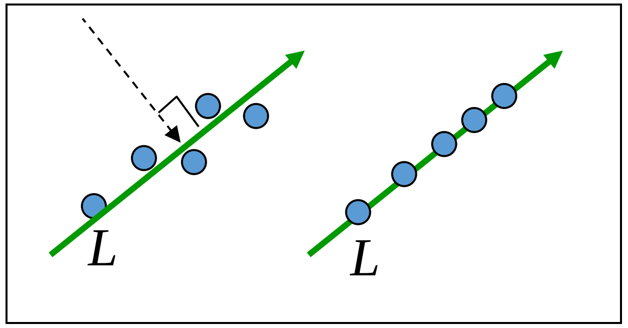
- In a way, S measures variance (= scatterness) of the data in different directions.
- Let’s look at a line L through the center of mass $\mathbf{m}$, and project our points $\mathbf{x}_i$ onto it. The variance of the projected points $\mathbf{x'}_i$ is:

$$\boxed{\operatorname{var}(L) = \tfrac{1}{n}\sum_{i=1}^{n} \| \mathbf{x}'_i - \mathbf{m} \|^2}$$


Original set



Small variance

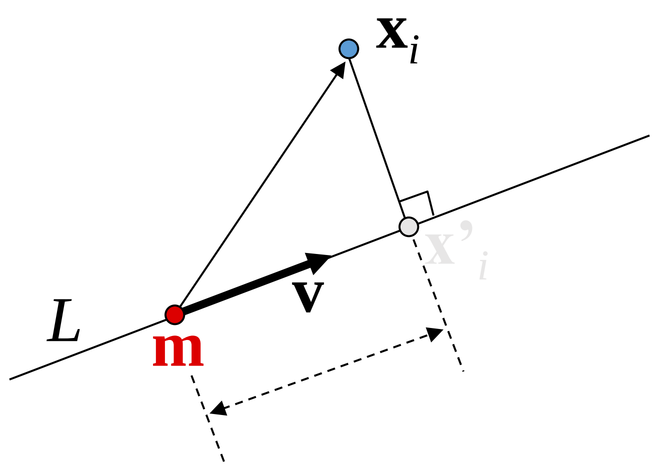


Large variance

Variance of projected points

- Given a direction $\mathbf{v}$, $\|\mathbf{v}\| = 1$, the projection of $\mathbf{x}_i$ onto $L = \mathbf{m} + \mathbf{v}t$ is:

$$\boxed{\| \mathbf{x}'_i - \mathbf{m} \| = < \mathbf{v},\ \mathbf{x}_i - \mathbf{m} > / \| \mathbf{v} \| = < \mathbf{v},\ \mathbf{y}_i > = \mathbf{v}^T \mathbf{y}_i}$$



Variance of projected points

- So,

$$\operatorname{var}(L) = \frac{1}{n}\sum_{i=1}^{n} \| \mathbf{x}_i' - \mathbf{m} \|^2 = \frac{1}{n}\sum_{i=1}^{n} (\mathbf{v}^{\mathbf{T}}\mathbf{y}_i)^2 = \frac{1}{n} \| \mathbf{v}^{\mathbf{T}} Y \|^2 =$$

$$= \frac{1}{n} \| Y^T \mathbf{v} \|^2 = \frac{1}{n} < Y^T\mathbf{v},\ Y^{\mathbf{T}}\mathbf{v} > = \frac{1}{n}\mathbf{v}^{\mathbf{T}} Y Y^T \mathbf{v} = \frac{1}{n}\mathbf{v}^{\mathbf{T}} S \mathbf{v} = \frac{1}{n} < S\mathbf{v},\ \mathbf{v} >$$

$$\sum_{i=1}^{n} (\mathbf{v}^T \mathbf{y}_{\mathbf{i}})^2 = \sum_{i=1}^{n} \left(\begin{pmatrix} v^1 & v^2 & \cdots & v^d \end{pmatrix} \begin{pmatrix} y_i^1 \\ y_i^2 \\ \vdots \\ y_i^d \end{pmatrix} \right)^2 = \left\| \begin{pmatrix} v^1 & v^2 & \cdots & v^d \end{pmatrix} \begin{pmatrix} y_1^1 & y_2^1 & \cdots & y_n^1 \\ y_1^2 & y_2^2 & & y_n^2 \\ \vdots & \vdots & & \vdots \\ y_1^d & y_2^d & \cdots & y_n^d \end{pmatrix} \right\|^2 = \| \mathbf{v}^{\mathbf{T}} Y \|^2$$

Directions of maximal variance

- So, we have: $\text{var}(L) = \langle S\mathbf{v}, \mathbf{v}\rangle$
- Theorem:

  Let $f : \{\mathbf{v} \in R^d \;|\; ||\mathbf{v}|| = 1\} \rightarrow R,$

  $f(\mathbf{v}) = \langle S\mathbf{v}, \mathbf{v}\rangle$ (and S is a symmetric matrix).

  Then, the extrema of f are attained at the eigenvectors of S.

- So, eigenvectors of S are directions of maximal/minimal variance!

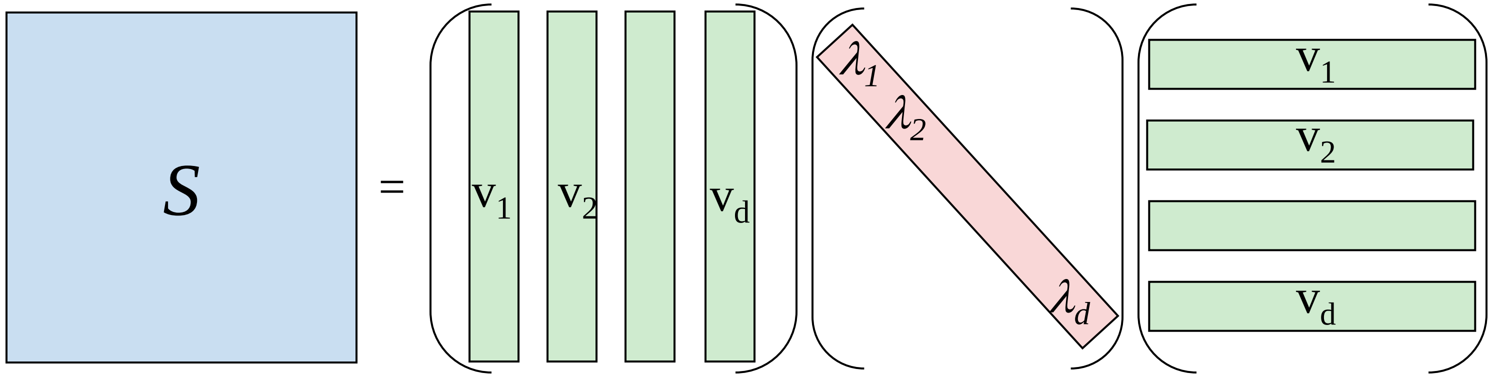
Summary so far

- We take the centered data vectors $\mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_\mathrm{n} \in R^d$
- Construct the scatter matrix $S = YY^T$
- S measures the variance of the data points
- Eigenvectors of S are directions of maximal variance.

Scatter matrix - eigendecomposition

- $\boldsymbol{S}$ is symmetric

$\Rightarrow$ $\boldsymbol{S}$ has eigendecomposition: $\boldsymbol{S} = V\Lambda V^{\mathrm{T}}$



The eigenvectors form orthogonal basis

Principal components

- Eigenvectors that correspond to big eigenvalues are the directions in which the data has strong components (= large variance).
- If the eigenvalues are more or less the same – there is no preferable direction.

- Note: the eigenvalues are always non-negative. Think why…

Technical remarks:

- $\lambda_i \geq 0, i = 1, \ldots, d$ (such matrices are called positive semi-definite). So we can indeed sort by the magnitude of λ_i
- Theorem: $\lambda_i \geq 0 \quad \Leftrightarrow \quad <S\mathbf{v}, \mathbf{v}> \geq 0 \quad \forall \mathbf{v}$

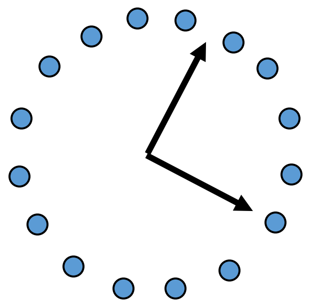
Proof:

$$S = V\Lambda V^T \quad \Rightarrow \quad < S\boldsymbol{v}, \boldsymbol{v} > = \boldsymbol{v}^T S \boldsymbol{v} = \boldsymbol{v}^T V \Lambda V^T \boldsymbol{v}$$
$$= (V^T \boldsymbol{v})^T \Lambda (\boldsymbol{v}^T V) = \boldsymbol{u}^T \Lambda \boldsymbol{u} = < \Lambda \boldsymbol{u}, \boldsymbol{u} >$$

$$\boxed{< S\mathbf{v}, \mathbf{v} > = \lambda_1 \mathbf{u_1}^2 + \lambda_2 \mathbf{u_2}^2 + \ldots + \lambda_d \mathbf{u_d}^2}$$

Therefore, $\lambda_i \geq 0 \Leftrightarrow <S\mathbf{v}, \mathbf{v}> \geq 0 \quad \forall \mathbf{v}$

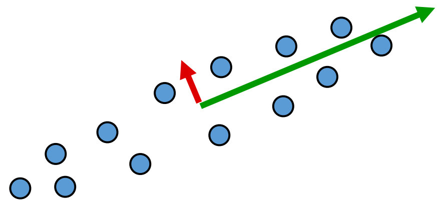
Principal components



- There's no preferable direction
- ***S*** looks like this:

$$\mathbf{V}\begin{pmatrix} \lambda & \\ & \lambda \end{pmatrix}\mathbf{V}^T$$

- Any vector is an eigenvector



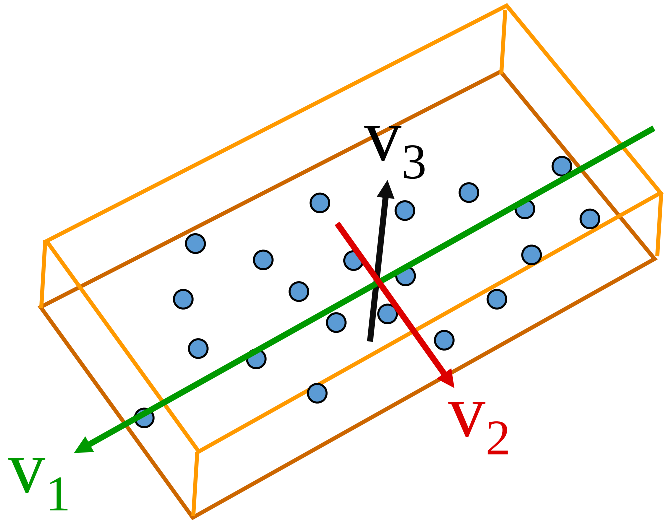
- There is a clear preferable direction
- ***S*** looks like this:

$$\mathbf{V}\begin{pmatrix} \lambda & \\ & \mu \end{pmatrix}\mathbf{V}^T$$

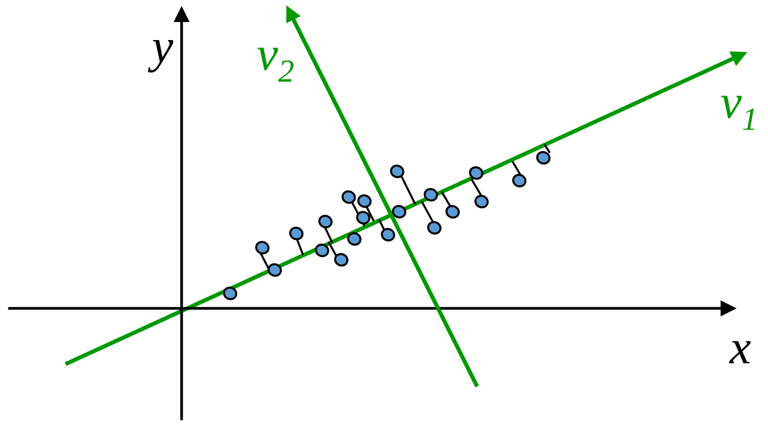
- μ is close to zero, much smaller than λ.

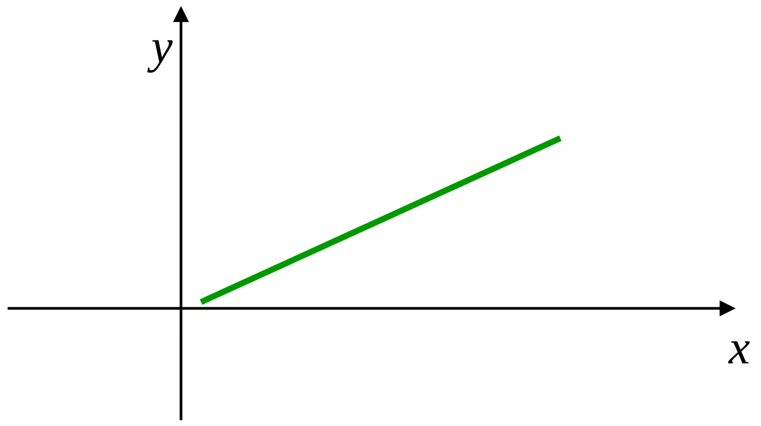
How to use what we got

- For finding oriented bounding box – we simply compute the bounding box with respect to the axes defined by the eigenvectors. The origin is at the mean point $\mathbf{m}$.

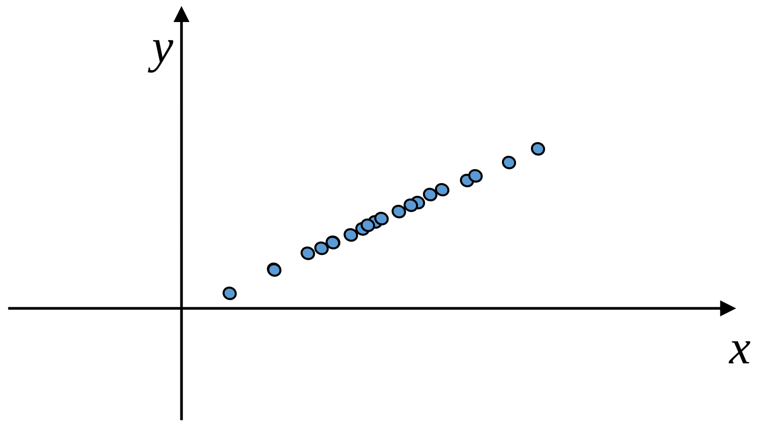


For approximation





This line segment approximates the original data set



The projected data set approximates the original data set

For approximation

- In general dimension d, the eigenvalues are sorted in descending order:

$$\lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_d$$

- The eigenvectors are sorted accordingly.
- To get an approximation of dimension $d' < d$, we take the d' first eigenvectors and look at the subspace they span ($d' = 1$ is a line, $d' = 2$ is a plane…)

For approximation

- To get an approximating set, we project the original data points onto the chosen subspace:

$$\mathbf{x}_i = \mathbf{m} + \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_{d'}\mathbf{v}_{d'} + \ldots + \alpha_d\mathbf{v}_d$$

Projection:

$$\mathbf{x}_i' = \mathbf{m} + \underbrace{\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_{d'}\mathbf{v}_{d'}} + 0\cdot\mathbf{v}_{d'+1} + \ldots + 0\cdot\mathbf{v}_d$$

Contrastive PCA

When to use contrastive PCA?

- If you want to compare two datasets with the same data attributes.

- Suppose we have gene-expression measurements from individuals of different ethnicities and sexes. This data includes gene expression levels of ***cancer patients*** $\{\boldsymbol{X_i}\}$, which we are interested in analyzing. We also have control data, which corresponds to the gene-expression levels of ***healthy patients*** $\{\boldsymbol{Y_i}\}$ from a similar demographic background.

Contrastive PCA

- Idea: enriched variations are (by definition) found only in target points, not in background

- Find directions that have high variance in target dataset, low variance in background dataset

Contrastive PCA

- Target dataset: $\boldsymbol{x_i} \in \mathbb{R}^{\boldsymbol{d}}$
- Background dataset: $\boldsymbol{y_i} \in \mathbb{R}^{\boldsymbol{d}}$
- Target/background covariance matrices: $\boldsymbol{S_X}/\boldsymbol{S_Y}$
- Target Variance: $\boldsymbol{\lambda_X(v) = v^T S_Y v}$
- Background Variance: $\boldsymbol{\lambda_Y(v) = v^T S_Y v}$

Contrastive PCA

$$\boldsymbol{v}^* = \mathbf{argmax}_{\boldsymbol{v}}\ \boldsymbol{\lambda}_{\boldsymbol{X}}(\boldsymbol{v}) - \boldsymbol{\alpha}\boldsymbol{\lambda}_{\boldsymbol{Y}}(\boldsymbol{v})$$

$$\Big\downarrow$$

$$\boldsymbol{v}^* = \mathbf{argmax}_{\boldsymbol{v}}\ \boldsymbol{v}^{\boldsymbol{T}}(\boldsymbol{S}_{\boldsymbol{X}} - \underset{\uparrow}{\boldsymbol{\alpha}}\boldsymbol{S}_{\boldsymbol{Y}})\boldsymbol{v}$$

How to choose?

Handwritten Digits on Complex Backgrounds

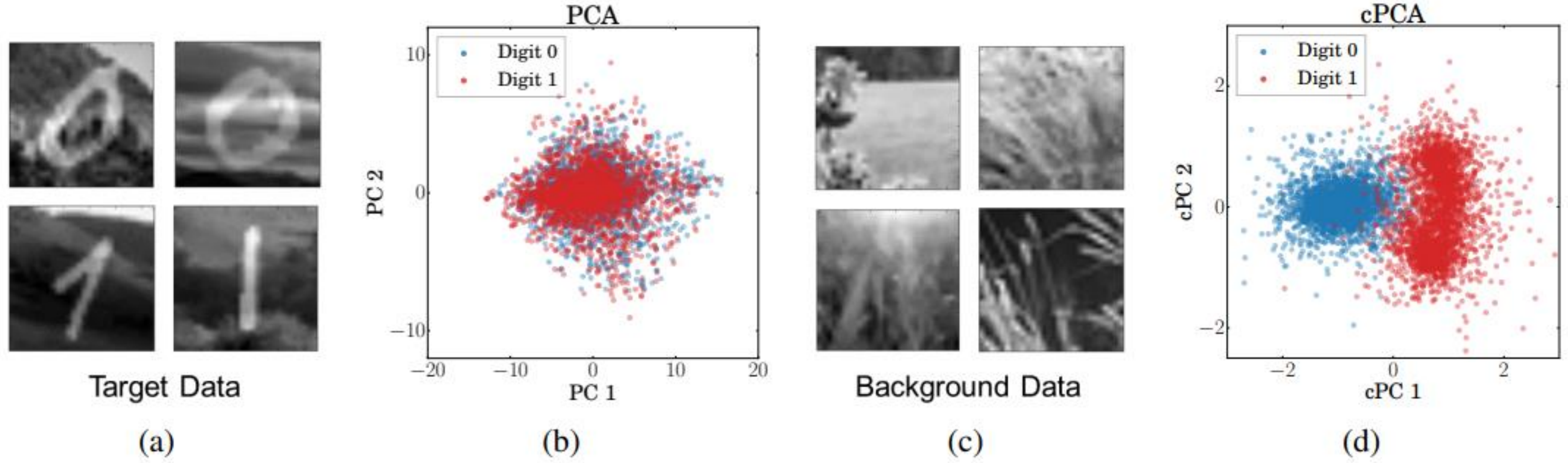
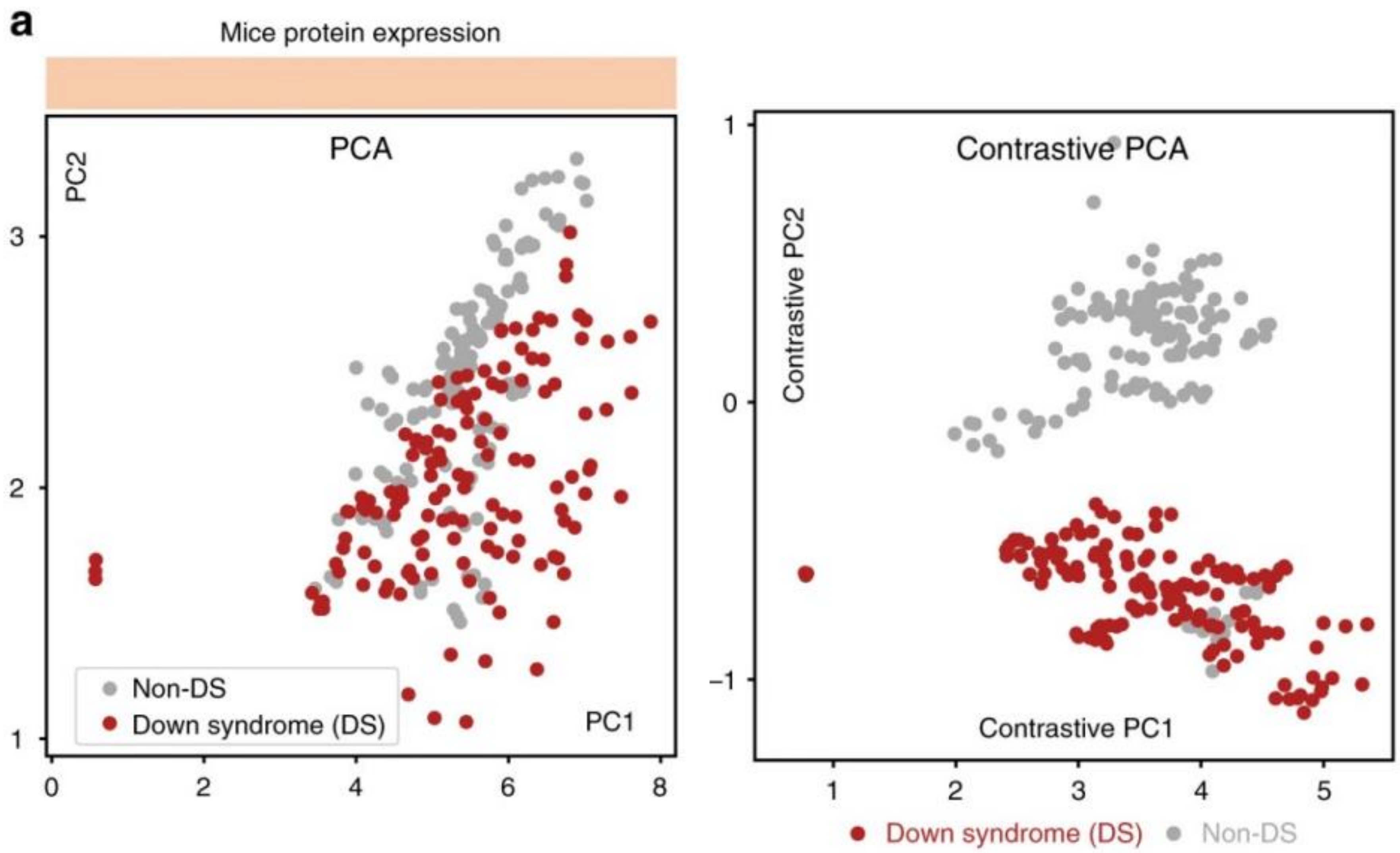


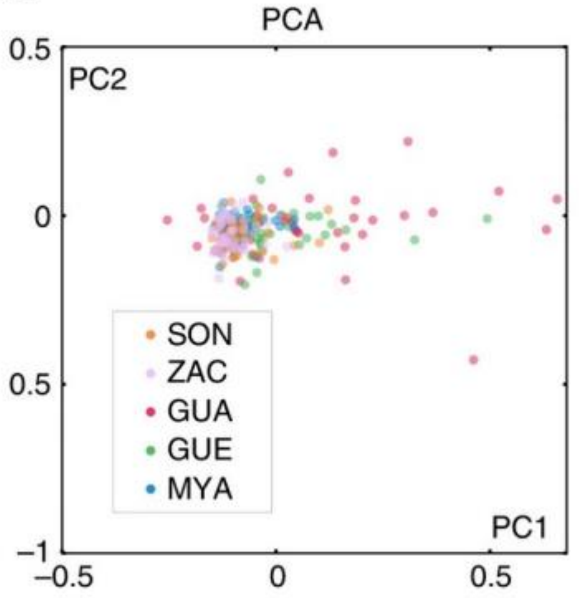
Figure 1. **Contrastive PCA on Synthetic Images.** (a) We create a target dataset of 5,000 synthetic images by randomly superimposing images of handwritten digits 0 and 1 from the MNIST dataset (LeCun et al., 1998) on top of images of grass taken from ImageNet dataset (Russakovsky et al., 2015) belonging to the synset *grass*. The images of grass are converted to grayscale, resized to be 100x100, and then randomly cropped to be the same size as the MNIST digits, 28x28. (b) Here, we plot the result of embedding the synthetic images onto their first two principal components using standard PCA. We see that the lower-dimensional embeddings of the images with 0s and images with 1s are hard to distinguish. (c) A background dataset is then introduced consisting solely of images of grass belonging to the same synset, but we use images that are different than those used to create the target dataset. (d) Using cPCA on the target and background datasets, (with a value of the contrast parameter α [see section 2] set to 2.0), two clusters emerge in the lower-dimensional representation of the target dataset, one consisting of images with the digit 0 and the other of images with the digit 1.

Mice Protein

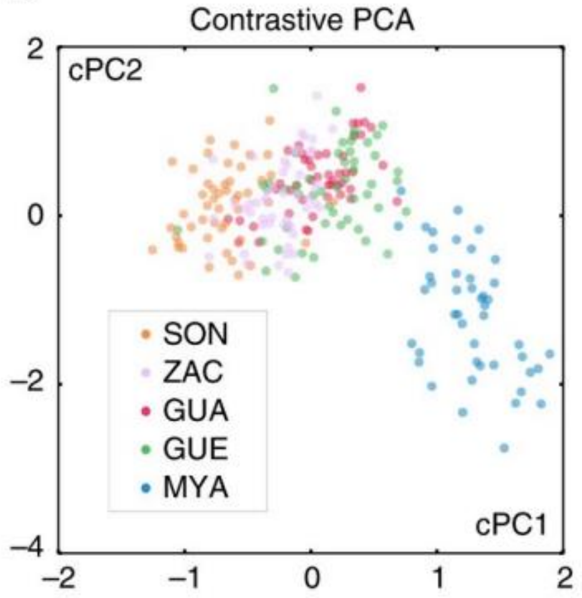


Mexican Ancestry

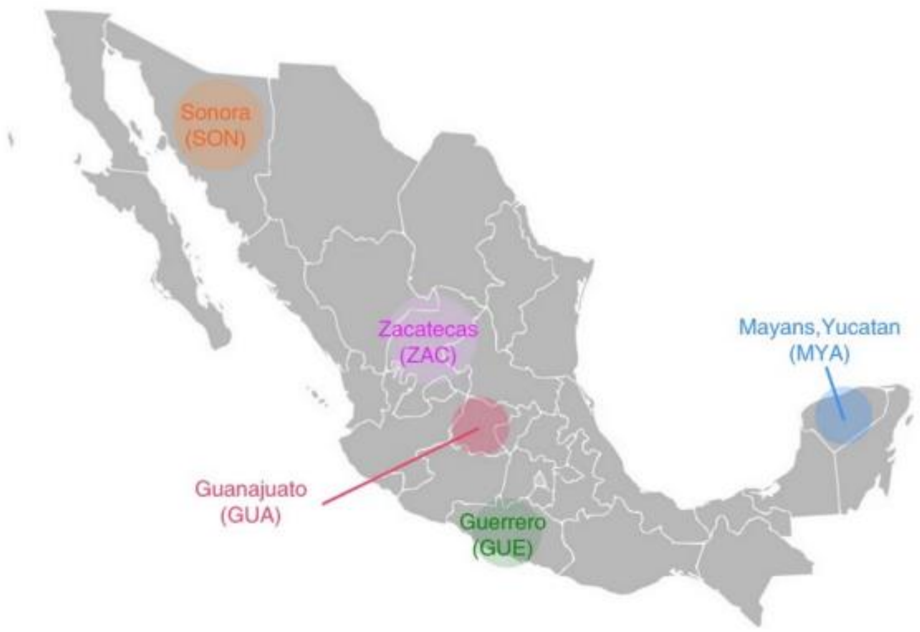
a



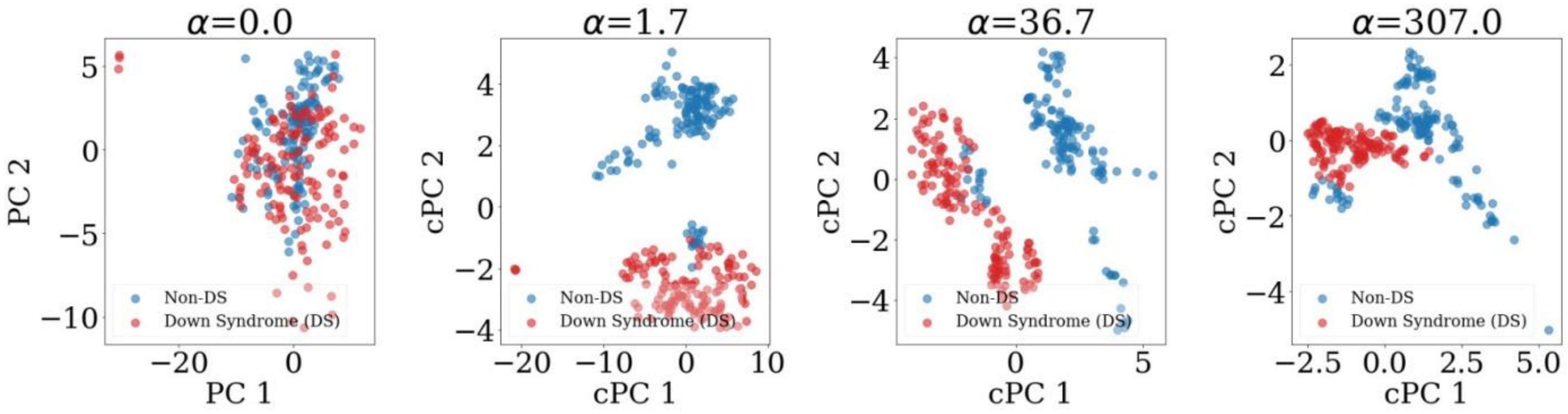
b



c



The role of $\boldsymbol{\alpha}$

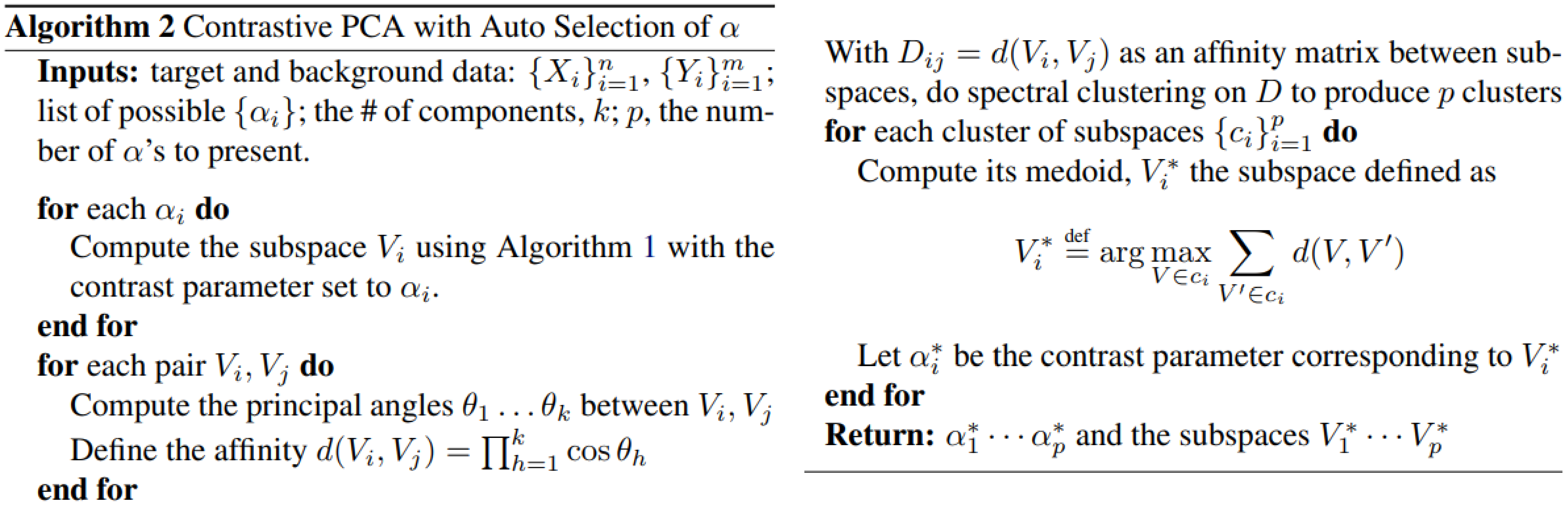


$\boldsymbol{\alpha}$ choosing procedure

- Try a bunch of alphas and get the resulting subspaces
- Measure how similar each pair of subspaces is
- Cluster subspaces based on their similarities
- Pick a representative subspace/alpha from these clusters
- Return list of potential alphas

- Choosing $\boldsymbol{\alpha}$ manually and visualize the results immediately?

Alpha choosing procedure



Algorithm 2 Contrastive PCA with Auto Selection of α

```
Inputs: target and background data: $\{X_i\}_{i=1}^n$, $\{Y_i\}_{i=1}^m$; list of possible $\{\alpha_i\}$; the # of components, $k$; $p$, the number of $\alpha$'s to present.

for each $\alpha_i$ do
    Compute the subspace $V_i$ using Algorithm 1 with the contrast parameter set to $\alpha_i$.
end for
for each pair $V_i, V_j$ do
    Compute the principal angles $\theta_1 \dots \theta_k$ between $V_i, V_j$
    Define the affinity $d(V_i, V_j) = \prod_{h=1}^{k} \cos\theta_h$
end for
With $D_{ij} = d(V_i, V_j)$ as an affinity matrix between subspaces, do spectral clustering on $D$ to produce $p$ clusters
for each cluster of subspaces $\{c_i\}_{i=1}^p$ do
    Compute its medoid, $V_i^*$ the subspace defined as
        $$V_i^* \stackrel{\text{def}}{=} \arg\max_{V \in c_i} \sum_{V' \in c_i} d(V, V')$$
    Let $\alpha_i^*$ be the contrast parameter corresponding to $V_i^*$
end for
Return: $\alpha_1^* \cdots \alpha_p^*$ and the subspaces $V_1^* \cdots V_p^*$
```

t-distributed Stochastic Neighboring Embedding (t-SNE)

The slides are from https://sites.uclouvain.be/ispgroup/uploads/ISPS/slides_120516.pdf

Dimensionality Reduction techniques solve optimization problems

$$X = \{x_1, x_2, \dots, x_n \in R^h\} \Rightarrow Y = \{y_1, y_2, \dots, y_n \in R^l\}$$

$$\min_{y} C(X, Y)$$

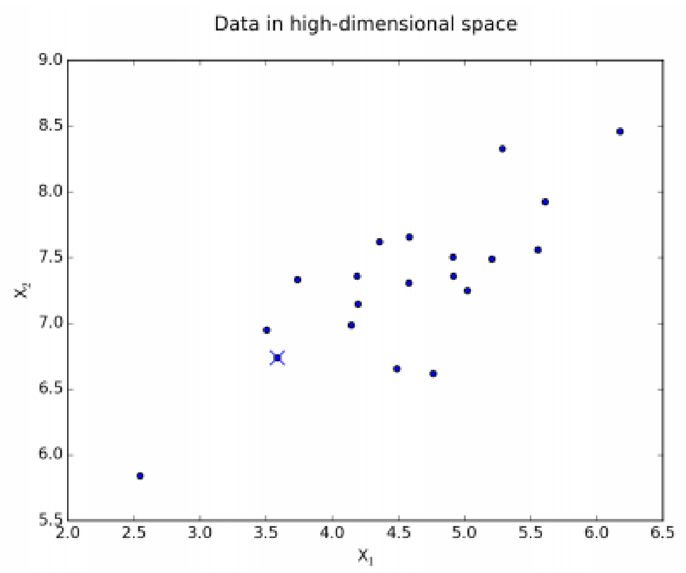
- Three approaches for Dimensionality Reduction
  - Distance preserving
  - Topology preserving
  - Information preserving
- t-SNE is distance-based but tends to preserve topology

SNE computes pair-wise similarities

$$p_{j|i} = \frac{exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k!=i} exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}, \quad q_{j|i} = \frac{exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k!=i} exp(-\|y_i - y_k\|^2)}$$

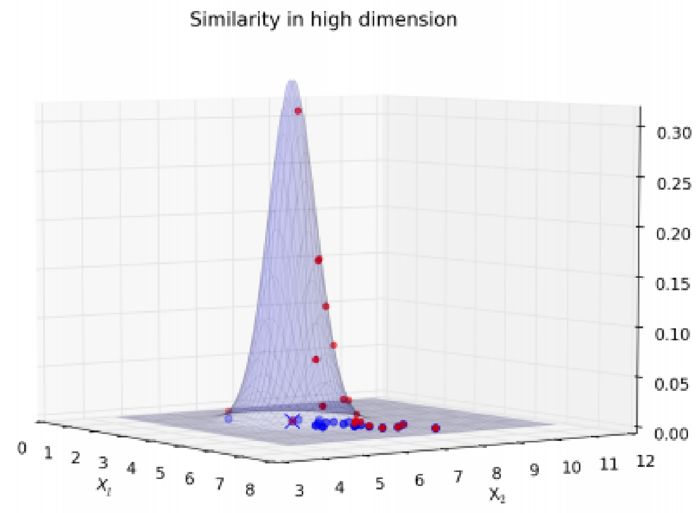
$$p_{i|i} = 0, \quad q_{i|i} = 0$$

- SNE converts euclidean distances to similarities, that can be interpreted as probabilities



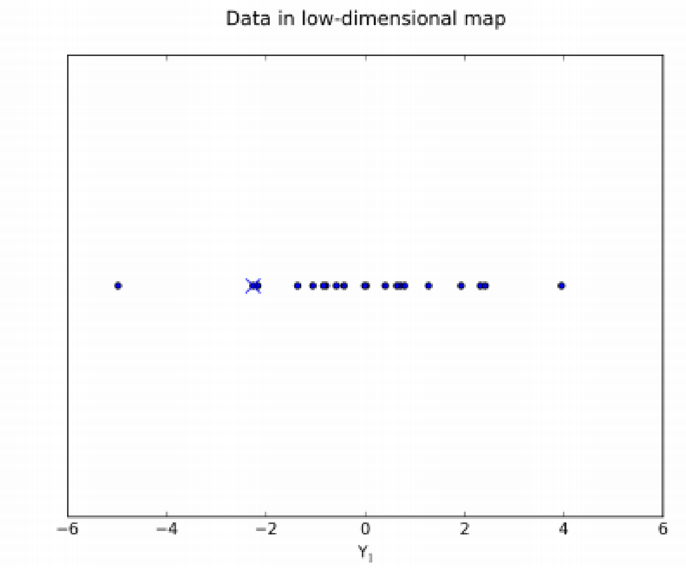
$p_{j|i}$

$\Leftrightarrow$



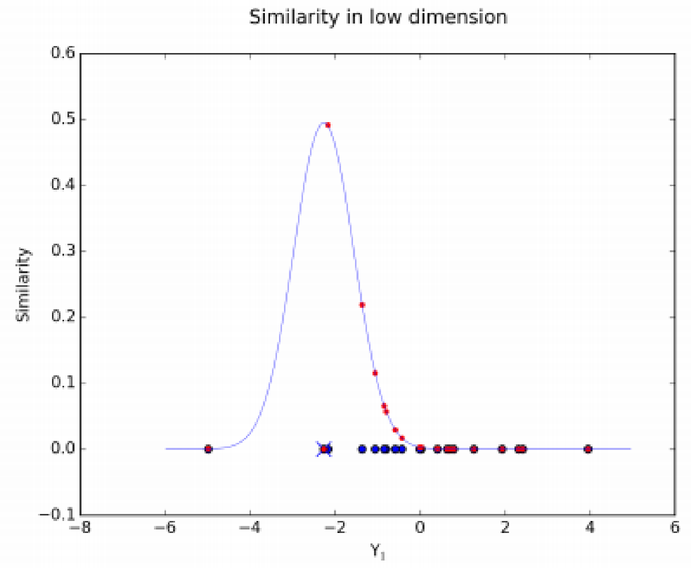
Pair-wise similarities should stay the same

$\Downarrow$



$q_{j|i}$

$\Leftrightarrow$



Kullback-Leiber Divergence measures the faithfulness with wich $\boldsymbol{q_{j|i}}$ models $\boldsymbol{p_{j|i}}$

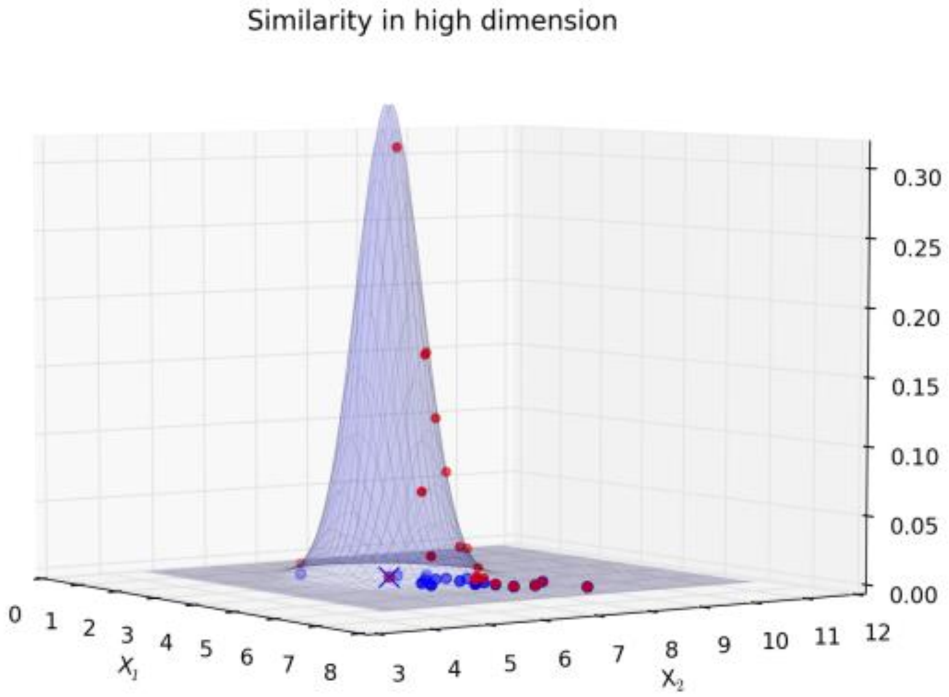
- $P_i = \{p_{1|i}, p_{2|i}, \dots, p_{n|i}\}$ and $Q_i = \{q_{1|i}, q_{2|i}, \dots, q_{n|i}\}$ are the distribution on the neighbors of point i.
- Kullback-Leiber Divergence (KL) compares two distributions.

$$C = \sum_i KL(P_i||Q_i) = \sum_i \sum_j p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

- KL divergence is asymmetric
- KL divergence is always positive.
- We have our minimization problem: $\min_y C(X, Y)$

Why radial basis function (exponential)?

$$p_{j|i} = \frac{exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k!=i} exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}, \quad q_{j|i} = \frac{exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k!=i} exp\left(-\|y_i - y_k\|^2\right)}$$



Focus on local geometry.

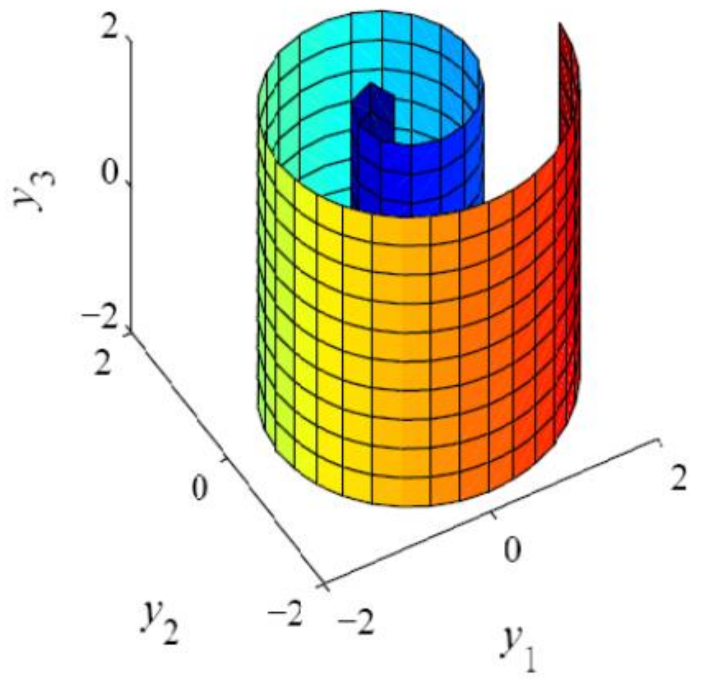
This is why t-SNE can be interpreted as topology-based

Why probabilities?

$$p_{j|i} = \frac{exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k!=i} exp\left(-\|x_i - x_k\|^2 / 2\sigma_i^2\right)}, \quad q_{j|i} = \frac{exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k!=i} exp(-\|y_i - y_k\|^2)}$$

Small distance does not mean proximity on manifold.

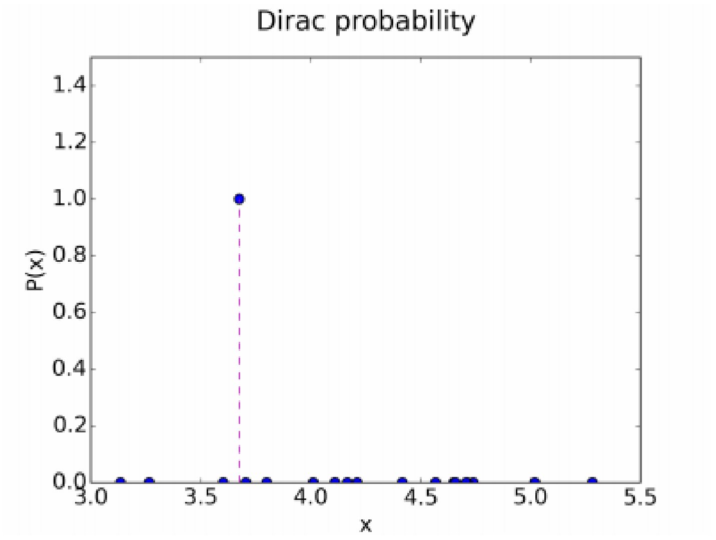
Probabilities are appropriate to model this uncertainty.

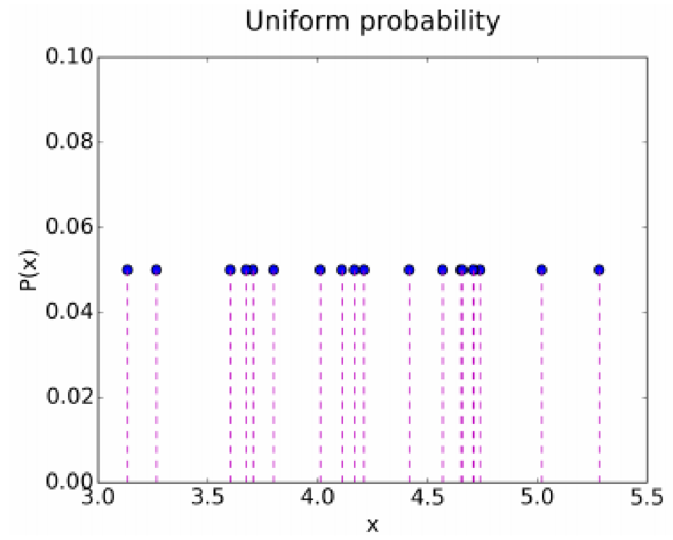


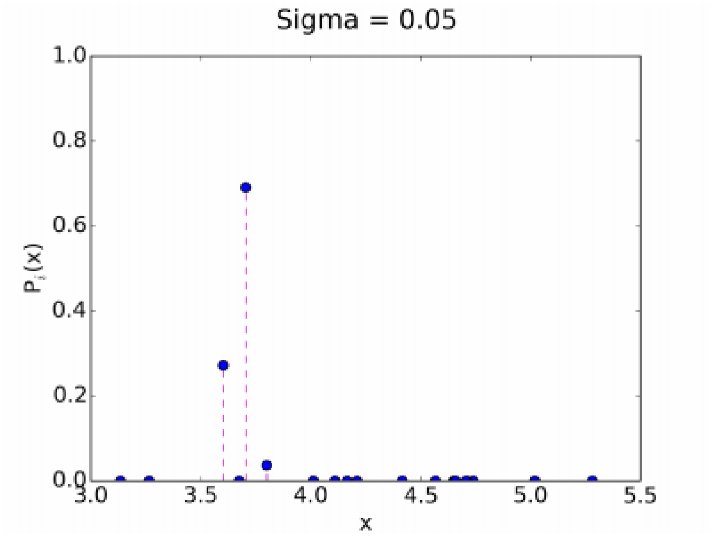
How do you choose σ_i ?

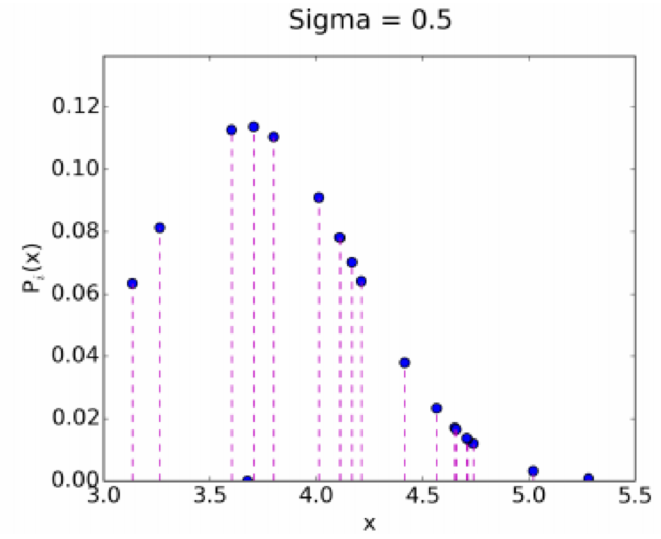
$$H(P) = -\sum_i p_i \log p_i$$

The entropy of P_i increases with σ_i

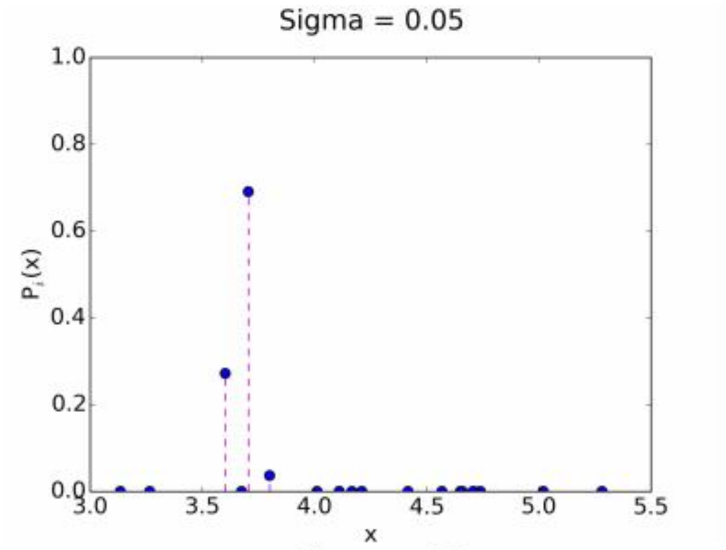




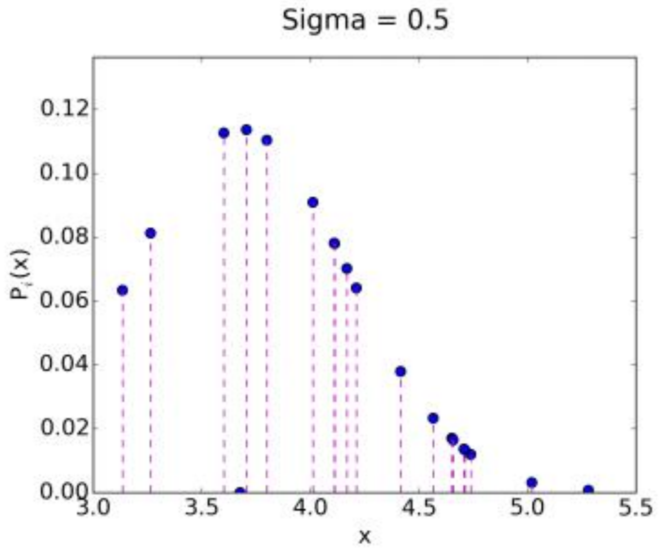




Perplexity, a smooth measure of the # of neighbors.



$\Rightarrow$ Entropy of 1.055
Perplexity of 2.078



$\Rightarrow$ Entropy of 3.800
Perplexity of 13.929

From SNE to t-SNE

SNE

Modelisation

$$p_{j|i} = \frac{exp\left(-\left\|x_i - x_j\right\|^2 / 2\sigma_i^2\right)}{\sum_{k!=i} exp(-\|x_i - x_k\|^2 / 2\sigma_i^2)},$$

$$q_{j|i} = \frac{exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k!=i} exp(-\|y_i - y_k\|^2)}$$

Cost Function

$$C = \sum_i KL(P_i||Q_i)$$

Derivatives

$$\frac{dC}{dy_i} = 2\sum_j (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

Symmetric SNE

Modelisation

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2},$$

$$q_{ij} = \frac{exp\left(-\left\|y_i - y_j\right\|^2\right)}{\sum_{k!=i} exp(-\|y_i - y_k\|^2)}$$

Cost Function

$$C = \sum KL(P||Q)$$

Derivatives

$$\frac{dC}{dy_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)$$

t-SNE

Modelisation

$$p_{ij} = \frac{p_{j|i} + p_{i|j}}{2},$$

$$q_{ij} = \frac{\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}}{\sum_{k!=i} (1 + \|y_i - y_k\|^2)^{-1}}$$

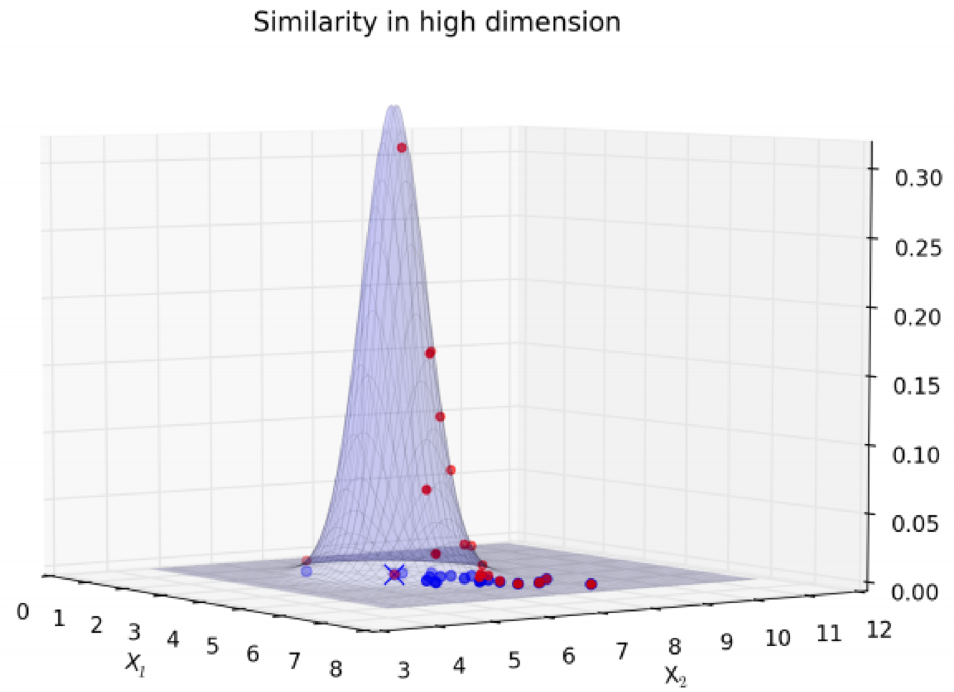
Cost Function

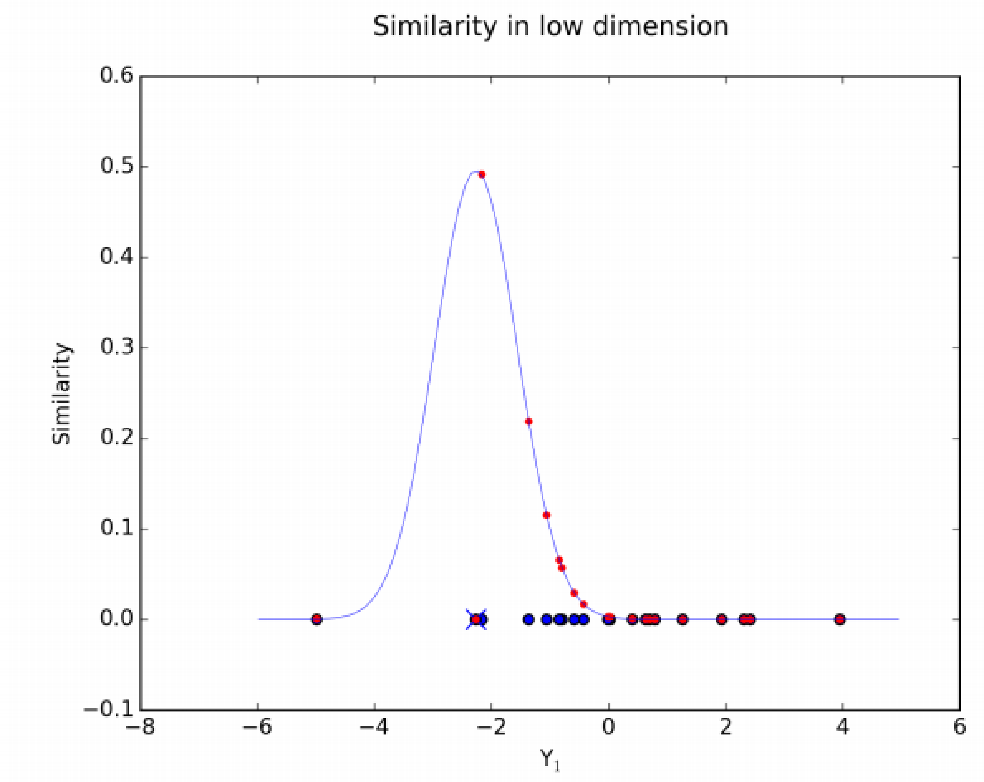
$$C = \sum KL(P||Q)$$

Derivatives

$$\frac{dC}{dy_i} = 4\sum_j (p_{ij} - q_{ij})(y_i - y_j)\left(1 + \left\|y_i - y_j\right\|^2\right)^{-1}$$

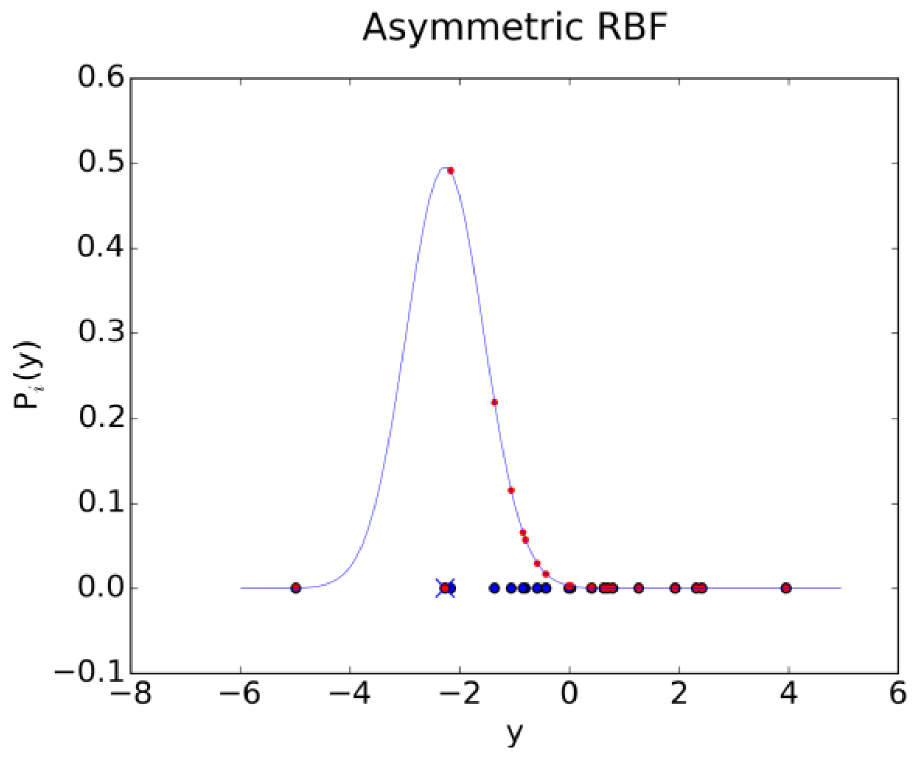
The ”Crowding problem”

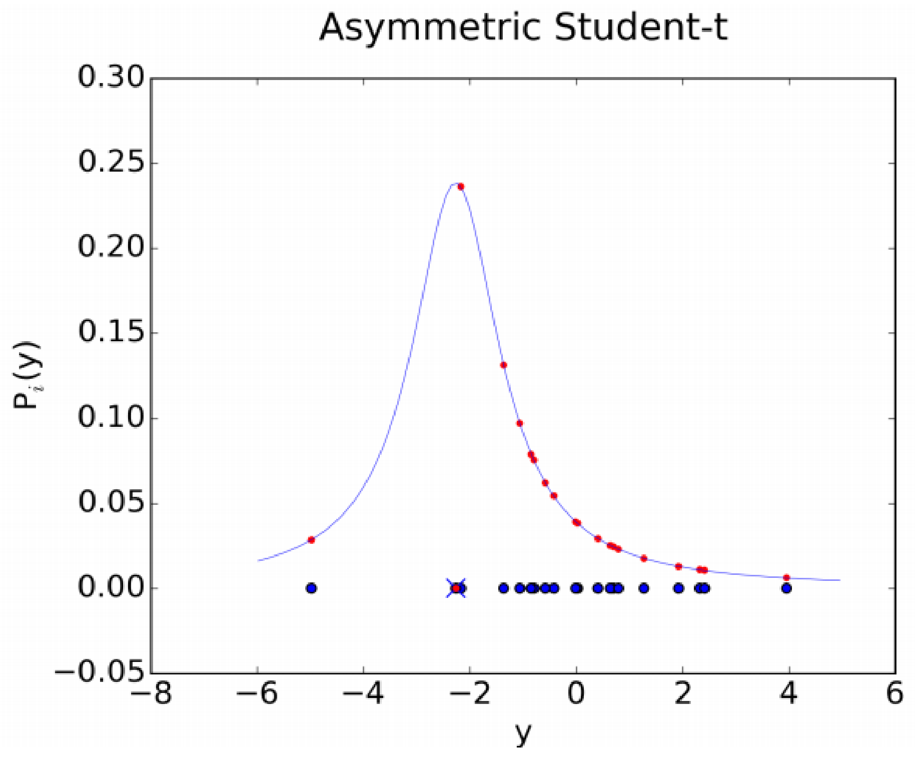




- There is much more space in high dimensions.

Mismatched Tails can Compensate for Mismatched Dimensionalities





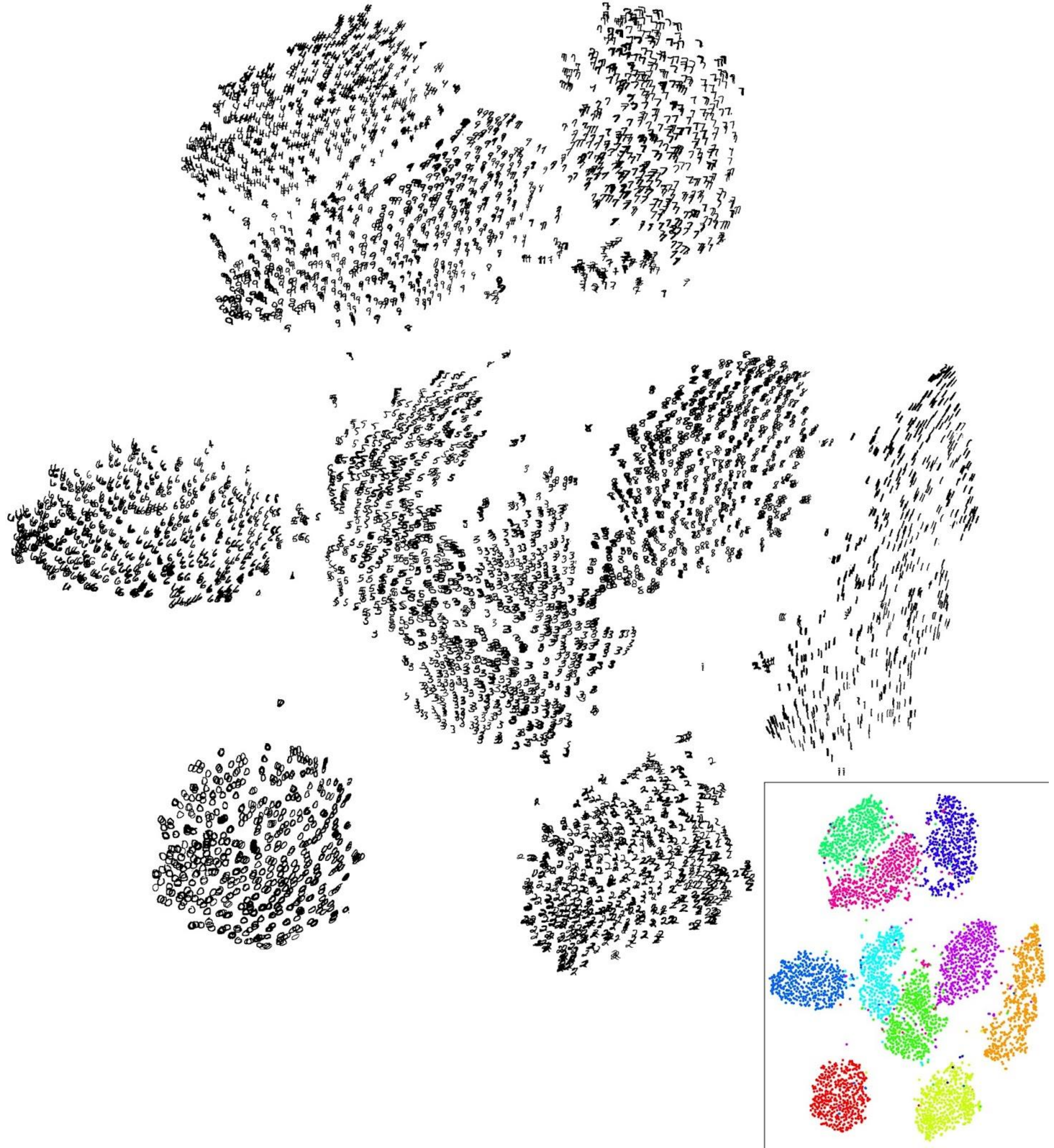
- Student-t distribution has heavier tails.

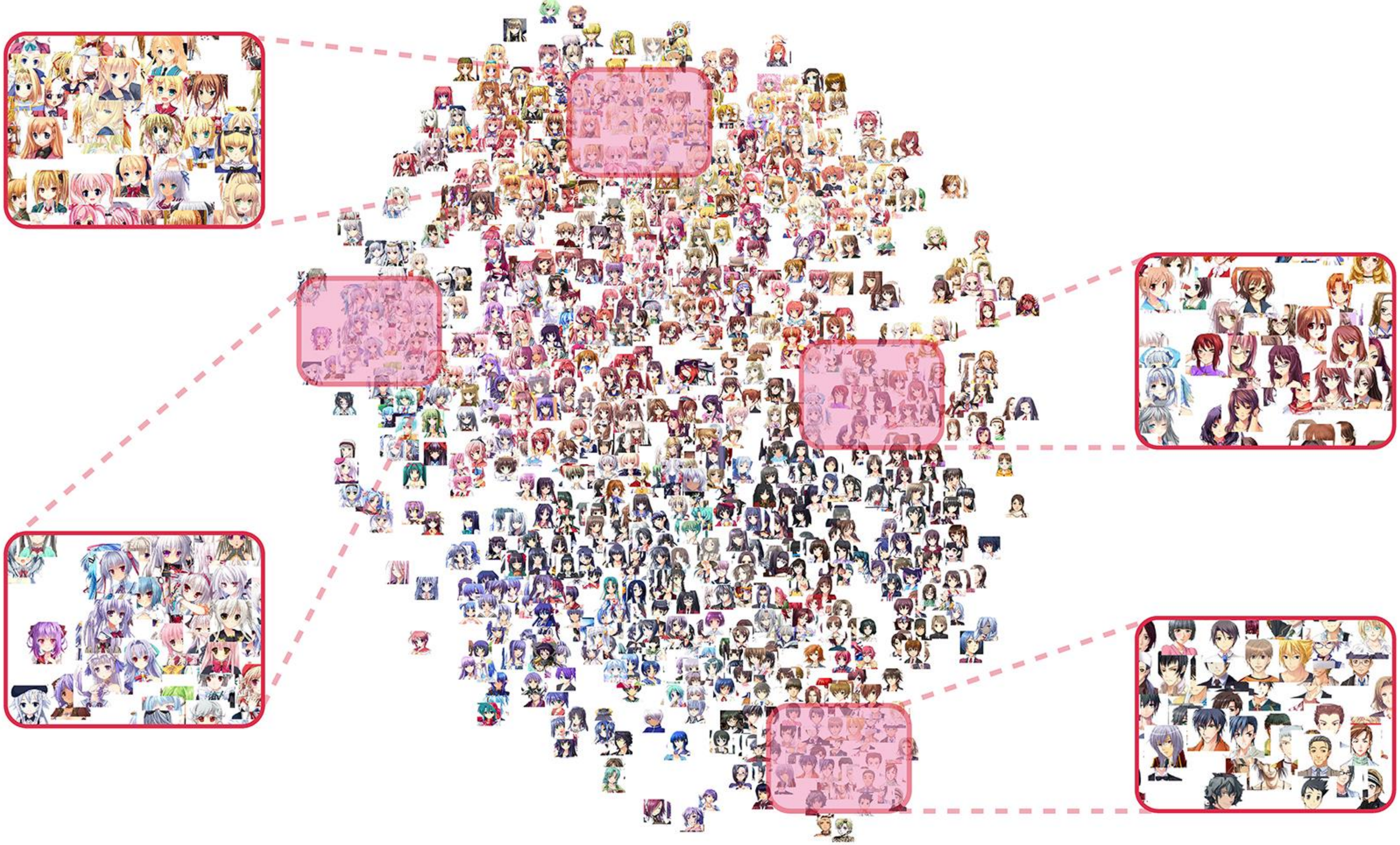
Last but not least: Optimization

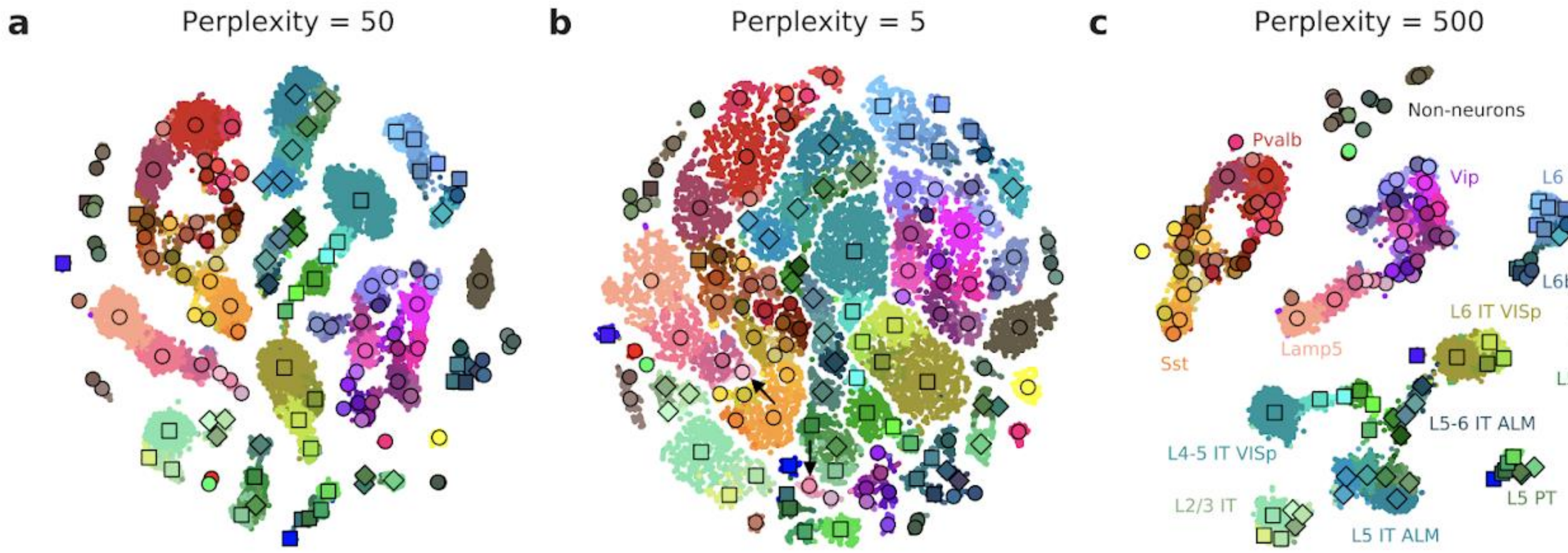
$$\min_{y} C(X,Y)$$

$$C = \sum_{i} KL(P_i||Q_i) = \sum_{i}\sum_{j} p_{j|i} log \frac{p_{j|i}}{q_{j|i}}$$

- Non-convex
- Gradient descent + Adaptive learning rate + Momentum
- $y^t = y^{t-1} + \eta \frac{\partial C}{\partial Y} + \alpha(y^{t-1} - y^{t-2})$






a
Perplexity = 50
b
Perplexity = 5
c
Perplexity = 500
Non-neurons
Pvalb
Vip
L6 CT
L6b
Sst
Lamp5
L6 IT VISp
L5 NP
L5-6 IT ALM
L4-5 IT VISp
L2/3 IT
L5 IT ALM
L5 PT