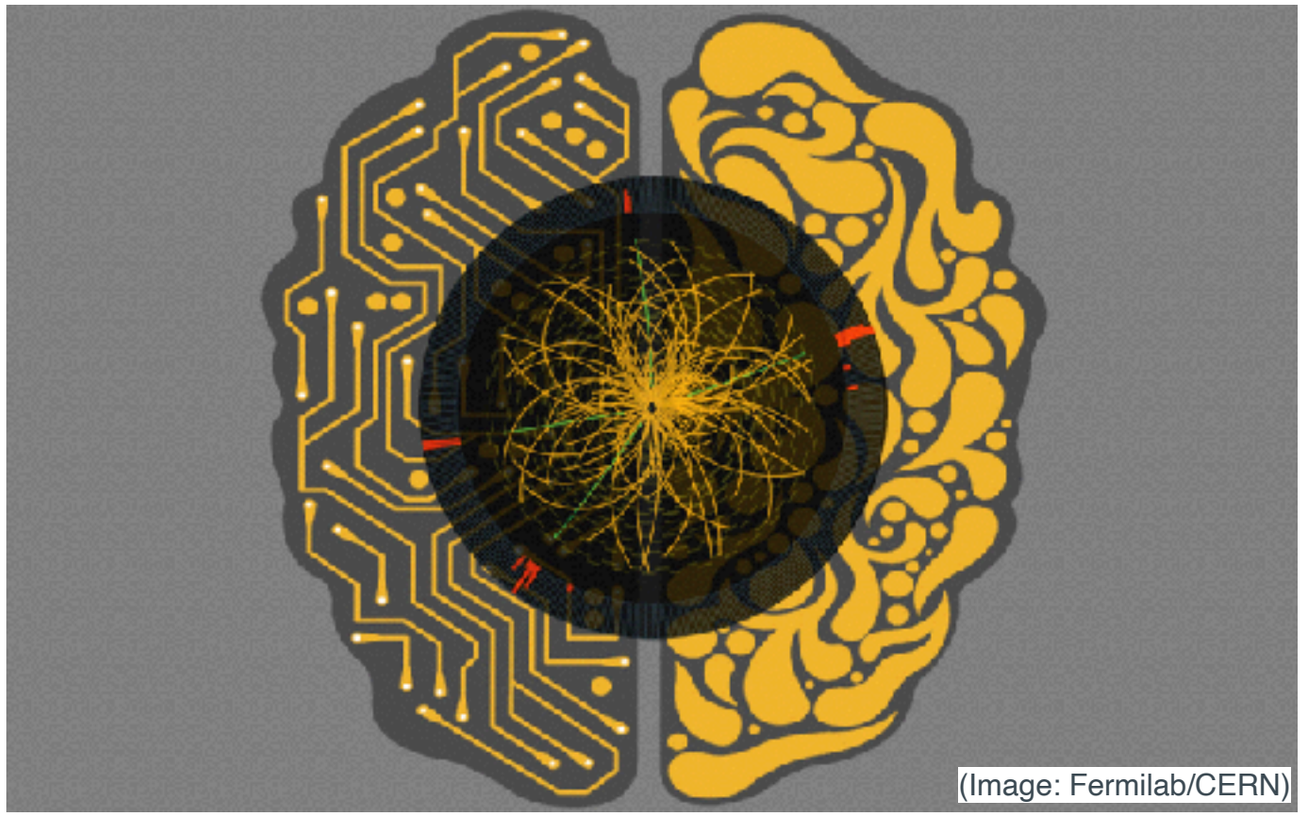
PHY 835: Collider Physics Phenomenology

Machine Learning in Fundamental Physics

Gary Shiu, UW-Madison



Lecture 13: Unsupervised Learning

Recap of Lecture 12

- Recurrent Neural Networks (RNNs)
- Teacher forcing
- Bi-directional RNNs
- Deep RNNs
- Long term dependencies and gated recurrent units

Outline for today

- Unsupervised learning
- Challenges of High-dimensional data
- Principal component analysis (PCA)
- Multi-dimensional scaling (MDS)
- t-stochastic neighbor embedding (t-SNE)

References: 1803.08823, Deep Learning Book

Unsupervised Learning

- Discovering structure in unlabelled data.
- Two ways: 1) some appropriate numerical measure (e.g. distance in some representation space). 2) with visualizations.
- Need to dimensionally reduce data as it is impractical for datasets involving large number of measured features (e.g. images)
- We call the dimensionally reduced space **latent space**.
- By dimensional reduction we often loose information. This is not necessarily bad. By loosing only irrelevant information, we can find good representations.

Challenges of High-dimensional Data

- **High-dimensional data lives near the edge of sample space.**

- Consider data distributed uniformly at random in a D-dimensional hypercube $C = [-e/2, e/2]^D$. Probably of a data point inside a D-dimensional hypersphere S of radius $e/2$ centered at the origin:

$$p(\|\boldsymbol{x}\|_2 < e/2) \sim (1/2)^D \quad \to 0 \text{ exponentially as } D \to \infty$$

- Most of the data will concentrate outside the hypersphere, in the corners of the hypercube.

- Recall this property underlies properties of statistical systems such as the Maxwell distribution.

Challenges of High-dimensional Data

- **Real-world data is usually not random or uniformly distributed** (data lives in a lower-dim. space compared with original space).

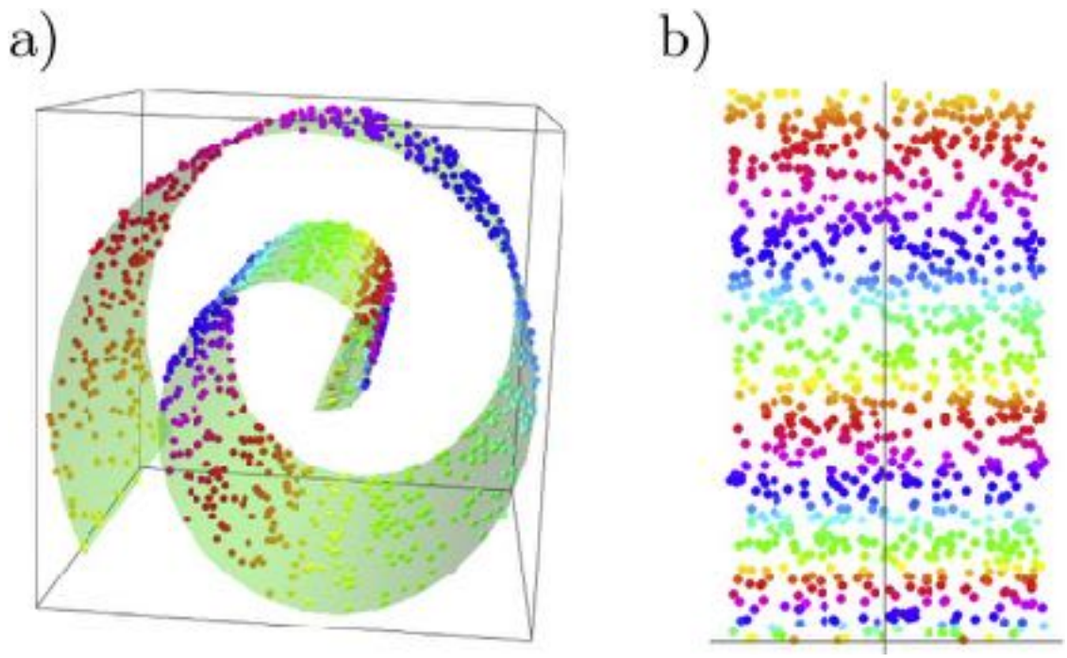
- "Blessing of non-uniformity": Data will typically be locally smooth (local variation will not incur a change in the target variable).

- Example: thermodynamics variables (temperature, pressure, etc) are not sensitive to variations of the dynamical variables (position and momentum of individual particles).

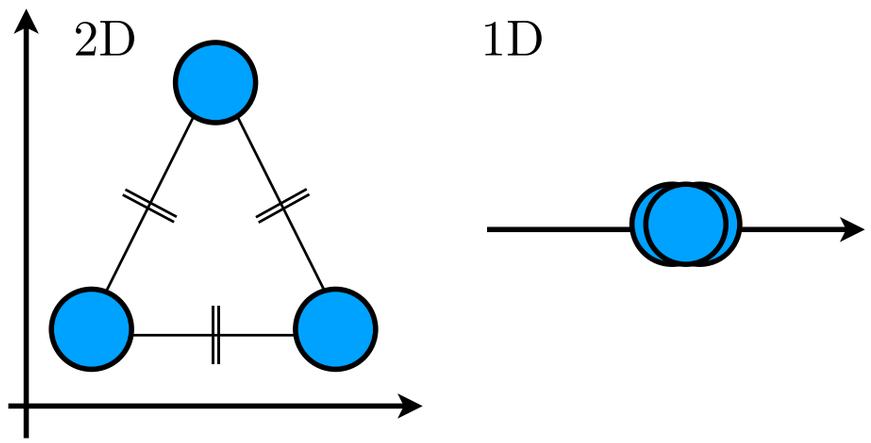
- Objective: preserve relative pairwise distances between data points when going to latent space.

Challenges of High-dimensional Data

- **Intrinsic dimensionality and the crowding problem:**



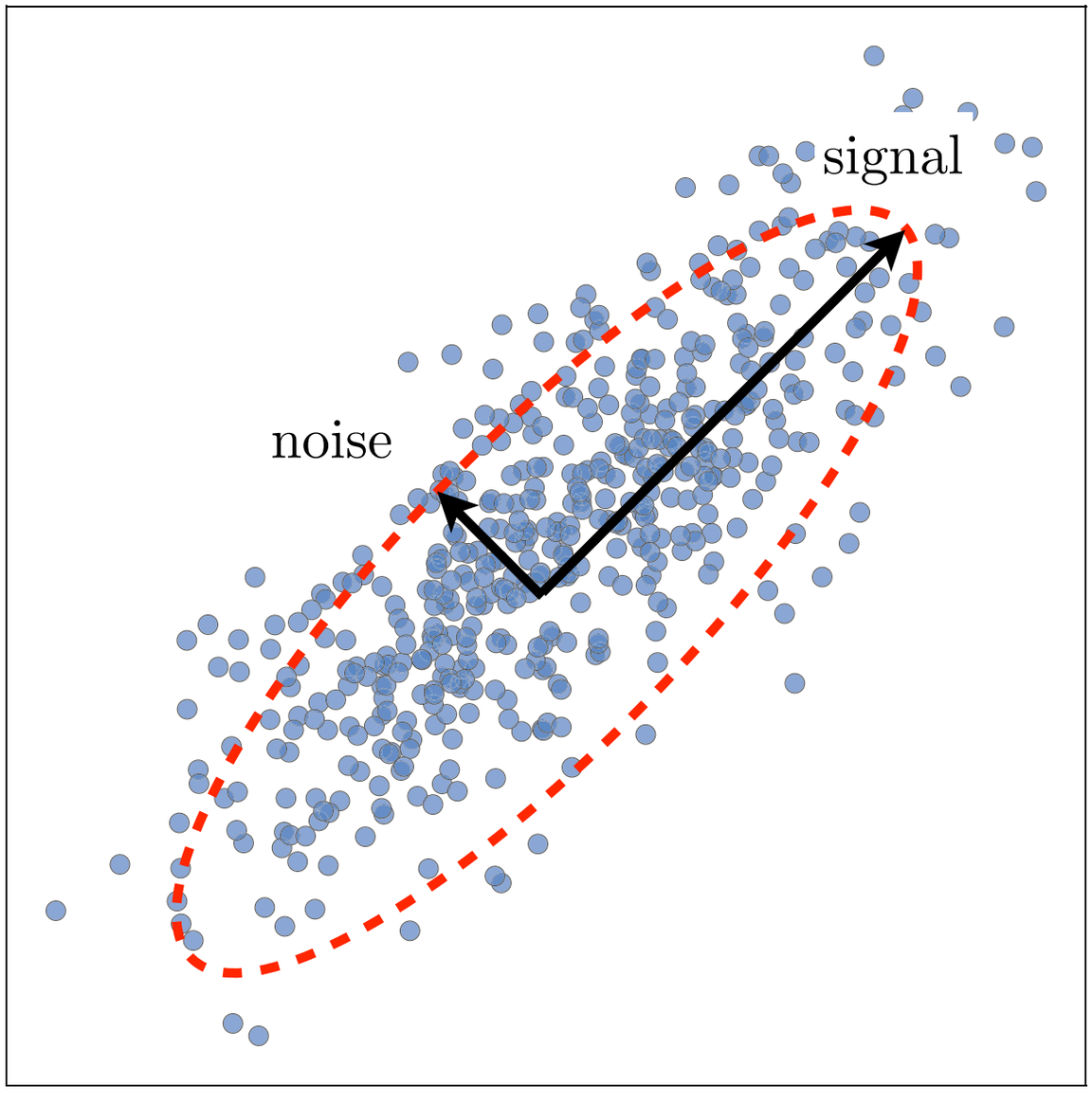
Intrinsic dim = min. # parameters
to parametrize the data.

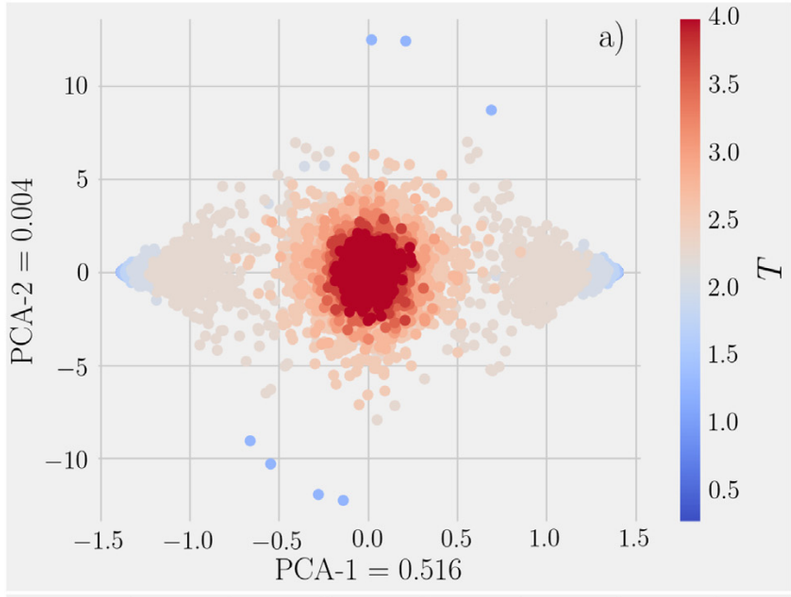


Attempts to represent data in a space
with dim < intrinsic dimensionality
lead to a “crowding” problem.

Principal Component Analysis (PCA)

- **Perform an orthogonal transformation of the data to find the high variance directions ⇔ minimizing the error in projection.**





PCA — Minimizing Decoding Error

- Suppose we have a collection of N points $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(N)}\}$ in $\mathbb{R}^n$.
- Compress them into code vectors $\{\mathbf{c}^{(1)}, \ldots, \mathbf{c}^{(N)}\}$ in $\mathbb{R}^l$ with $l < n$
- Encoding function: $f(\mathbf{x}) = \mathbf{c}$; Decoding function: $\mathbf{x} = g(\mathbf{c})$
- Encoding + decoding: $\tilde{\mathbf{x}} = g(f(\mathbf{x}))$
- A measure of goodness for your compression is how accurate is this encoding+decoding:

$$||\mathbf{x} - \tilde{\mathbf{x}}|| \ll 1 \quad \text{(good commpression)}$$

PCA — Minimizing Decoding Error

- Let $g(\mathbf{c}) = \mathbf{D}\mathbf{c}$ where $\mathbf{D} \in \mathbb{R}^{n \times l}$ is a matrix defining the decoding.
- Columns of $\mathbf{D}$ are orthogonal to each other and have unit norm.
- Minimizing the loss: $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} ||\boldsymbol{x} - g(\boldsymbol{c})||_2$

  or equivalently (and more conveniently): $\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} ||\boldsymbol{x} - g(\boldsymbol{c})||_2^2$
- The function to be minimized: $(\boldsymbol{x} - g(\boldsymbol{c}))^\top (\boldsymbol{x} - g(\boldsymbol{c}))$

$$= \boldsymbol{x}^\top \boldsymbol{x} - \boldsymbol{x}^\top g(\boldsymbol{c}) - g(\boldsymbol{c})^\top \boldsymbol{x} + g(\boldsymbol{c})^\top g(\boldsymbol{c}) \;\; = \boldsymbol{x}^\top \boldsymbol{x} - 2\boldsymbol{x}^\top g(\boldsymbol{c}) + g(\boldsymbol{c})^\top g(\boldsymbol{c})$$

- Omit the first term which does not depend on $\mathbf{c}$:

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top g(\boldsymbol{c}) + g(\boldsymbol{c})^\top g(\boldsymbol{c})$$

PCA — Minimizing Decoding Error

- Using the definition of the decoding function:

$$\boldsymbol{c}^* = \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{D}^\top \boldsymbol{D}\boldsymbol{c}$$

$$= \arg\min_{\boldsymbol{c}} -2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{I}_l \boldsymbol{c}$$

because the columns of D are orthogonal and have unit norm.

- The optimization problem has the solution:

$$\nabla_{\boldsymbol{c}}(-2\boldsymbol{x}^\top \boldsymbol{D}\boldsymbol{c} + \boldsymbol{c}^\top \boldsymbol{c}) = \boldsymbol{0}$$

$$-2\boldsymbol{D}^\top \boldsymbol{x} + 2\boldsymbol{c} = \boldsymbol{0}$$

$$\boldsymbol{c} = \boldsymbol{D}^\top \boldsymbol{x}.$$

- The encoding function: $f(\boldsymbol{x}) = \boldsymbol{D}^\top \boldsymbol{x}$
- PCA reconstruction operation: $r(\boldsymbol{x}) = g\left(f\left(\boldsymbol{x}\right)\right) = \boldsymbol{D}\boldsymbol{D}^\top \boldsymbol{x}$

PCA — Minimizing Decoding Error

- Since we use the same matrix $\mathbf{D}$ to decode all the points, we minimize the Frobenius norm of the matrix of errors computed over all dimensions and all points:

$$\boldsymbol{D}^* = \arg\min_{\boldsymbol{D}} \sqrt{\sum_{i,j} \left(x_j^{(i)} - r(\boldsymbol{x}^{(i)})_j \right)^2} \text{ subject to } \boldsymbol{D}^\top \boldsymbol{D} = \boldsymbol{I}_l.$$

- Consider $l = 1$ (generalization to other l, Ex 6), then $\mathbf{D} = \mathbf{d}$

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i ||\boldsymbol{x}^{(i)} - \boldsymbol{d}\boldsymbol{d}^\top \boldsymbol{x}^{(i)}||_2^2 \text{ subject to } ||\boldsymbol{d}||_2 = 1.$$

- Some cosmetic changes (noting $\mathbf{d}^T\mathbf{x}^{(i)}$ is a scalar, and so its transpose is equal to itself) give:

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} \sum_i ||\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(i)\top} \boldsymbol{d}\boldsymbol{d}||_2^2 \text{ subject to } ||\boldsymbol{d}||_2 = 1.$$

PCA — Minimizing Decoding Error

- Introduce compact notation by defining the matrix $\mathbf{X}$:

$$\boldsymbol{X} \in \mathbb{R}^{m \times n} \qquad\qquad \boldsymbol{X}_{i,:} = \boldsymbol{x}^{(i)\top}$$

- The decoding error is minimized when:

$$\boldsymbol{d}^* = \arg\min_{\boldsymbol{d}} ||\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top||_F^2 \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1.$$

- The Frobenius norm part:

$$\begin{aligned}
\arg\min_{\boldsymbol{d}} ||\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top||_F^2 &= \arg\min_{\boldsymbol{d}} \operatorname{Tr}\left(\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\right)^\top\left(\boldsymbol{X} - \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top\right)\right) \\
&= \arg\min_{\boldsymbol{d}} \operatorname{Tr}(\boldsymbol{X}^\top\boldsymbol{X} - \boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top - \boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X} + \boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \\
&= \arg\min_{\boldsymbol{d}} \cancel{\operatorname{Tr}(\boldsymbol{X}^\top\boldsymbol{X})} - \operatorname{Tr}(\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) - \operatorname{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}) + \operatorname{Tr}(\boldsymbol{d}\boldsymbol{d}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top)
\end{aligned}$$

does not depend on d

PCA — Minimizing Decoding Error

- Cycle the order of the matrices inside a trace, the Frobenius norm:

$$= \arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{d}\boldsymbol{d}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top)$$

$$= \arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top \boldsymbol{d}\boldsymbol{d}^\top)$$

- The constraint $\mathbf{d}^T\mathbf{d} = 1$ gives:

$$= \arg\min_{\boldsymbol{d}} -2\,\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) + \mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1$$

- Thus minimizing decoding error is the same as maximizing variance:

$$= \arg\min_{\boldsymbol{d}} -\mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1$$

$$= \arg\max_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}\boldsymbol{d}^\top) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1$$

$$= \arg\max_{\boldsymbol{d}} \mathrm{Tr}(\boldsymbol{d}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{d}) \text{ subject to } \boldsymbol{d}^\top\boldsymbol{d} = 1.$$

PCA — Maximizing Variance

- The covariance matrix of data matrix $\mathbf{X}$ is defined as:

$$\boldsymbol{\Sigma}(\boldsymbol{X}) = \frac{1}{N-1}\boldsymbol{X}^T\boldsymbol{X}$$

- $\Sigma(\mathbf{X})_{jj}$ corresponds to the variance of the j-th feature while $\Sigma(\mathbf{X})_{ij}$ measures the covariance (correlation) between feature i & feature j.

- Find a new basis that emphasizes highly variable directions while reducing redundancy between basis vectors. Perform SVD:

$$\boldsymbol{X} = \boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T,$$

$$\begin{aligned}\boldsymbol{\Sigma}(\boldsymbol{X}) &= \frac{1}{N-1}\boldsymbol{V}\boldsymbol{S}\boldsymbol{U}^T\boldsymbol{U}\boldsymbol{S}\boldsymbol{V}^T \\ &= \boldsymbol{V}\left(\frac{\boldsymbol{S}^2}{N-1}\right)\boldsymbol{V}^T \\ &\equiv \boldsymbol{V}\boldsymbol{\Lambda}\boldsymbol{V}^T.\end{aligned}$$

- The eigenvalues λ_i of Λ are given by $\lambda_i = s_i^2/(N-1)$.

PCA — Maximizing Variance

- To reduce the dimensionality of data from n to l, construct the $n \times l$ projection matrix $\mathbf{V}_l$ by selecting the singular components with the l largest singular values. The projection is then

$$\mathbf{Y} = \mathbf{X}\mathbf{V}_l$$

- The singular vector with the largest singular value (largest variance) is the first principal component; the singular vector with the second largest variance is the second principal component, etc.

- Common in data visualization is to project on the first few principal components (as long as a large part of the variance is explained in those components, e.g., Ising Model).

- Low explained variance may imply that the intrinsic dimensionality of the data is high, or it cannot be captured by a linear representation.

Multidimensional Scaling (MDS)

- Non-linear dimensional reduction technique which preserves the pairwise distance (or dissimilarity) d_{ij} between data points.
- **Metric MDS:** the latent coordinates are obtained by minimizing:

$$\tilde{\boldsymbol{Y}} = \arg\min_{\boldsymbol{Y}} \sum_{i<j} w_{ij} |d_{ij}(\boldsymbol{X}) - d_{ij}(\boldsymbol{Y})|,$$

- w_{ij} specifies the level of confidence (precision) in the value of $d_{ij}(\mathbf{X})$. If Euclidean metric is used, MDS=PCA; known as **classical scaling**.
- MDS (metric or non-metric) is a generalization of PCA.
- **Non-metric MDS:** d_{ij} can be any distance matrix that preserves the ordination, i.e., if $d_{12}(\mathbf{X}) < d_{13}(\mathbf{X})$ then $d_{12}(\mathbf{Y}) < d_{13}(\mathbf{Y})$.

Multidimensional Scaling (MDS)

- Both MDS and PCA can be implemented using standard Python packages such as Scikit.
- **MDS** algorithms have a scaling of $\mathcal{O}(N^3)$ where $N = \#$ data points.
- Sample-based methods can reduce this scaling to $\mathcal{O}(N \log N)$.
- **PCA** has a scaling of $\mathcal{O}(Np^2 + p^3)$ for a complete decomposition.

computation of covariance matrix

eigenvalue decomposition where $p = \#$ features

- Can be improved to give a $\mathcal{O}(Np^2 + p)$ scaling for PCA if only a few principal components are desired.

t-SNE

- t-stochastic neighbor embedding: non-parametric method that constructs non-linear embeddings, optimized to preserve the local data structure.
- **Idea:** associate a probability distribution to the neighborhood of each data:

$$p_{i|j} = \frac{\exp(-||x_i - x_j||^2/2\sigma_i^2)}{\sum_{k\neq i}\exp(-||x_i - x_k||^2/2\sigma_i^2)}, \qquad p_{i|i} = 0$$

- σ_i are free bandwidth parameters determined by the local entropy:

$$H(p_i) \equiv -\sum_j p_{j|i}\log_2 p_{j|i}.$$

- Setting $H(p_i)$= constant, $\Sigma = 2^{H(p_i)}$ = perplexity determines σ_i. Points in regions of high density will have small σ_i.

t-SNE

- **Gaussian likelihoods: only nearby points contribute**
  - Similarity of nearby points well represented
  - Problem of outliers (exponentially vanishing contributions to the distribution): embedding coordinates are ambiguous.

- The outliner problem can be avoided by symmetrization:

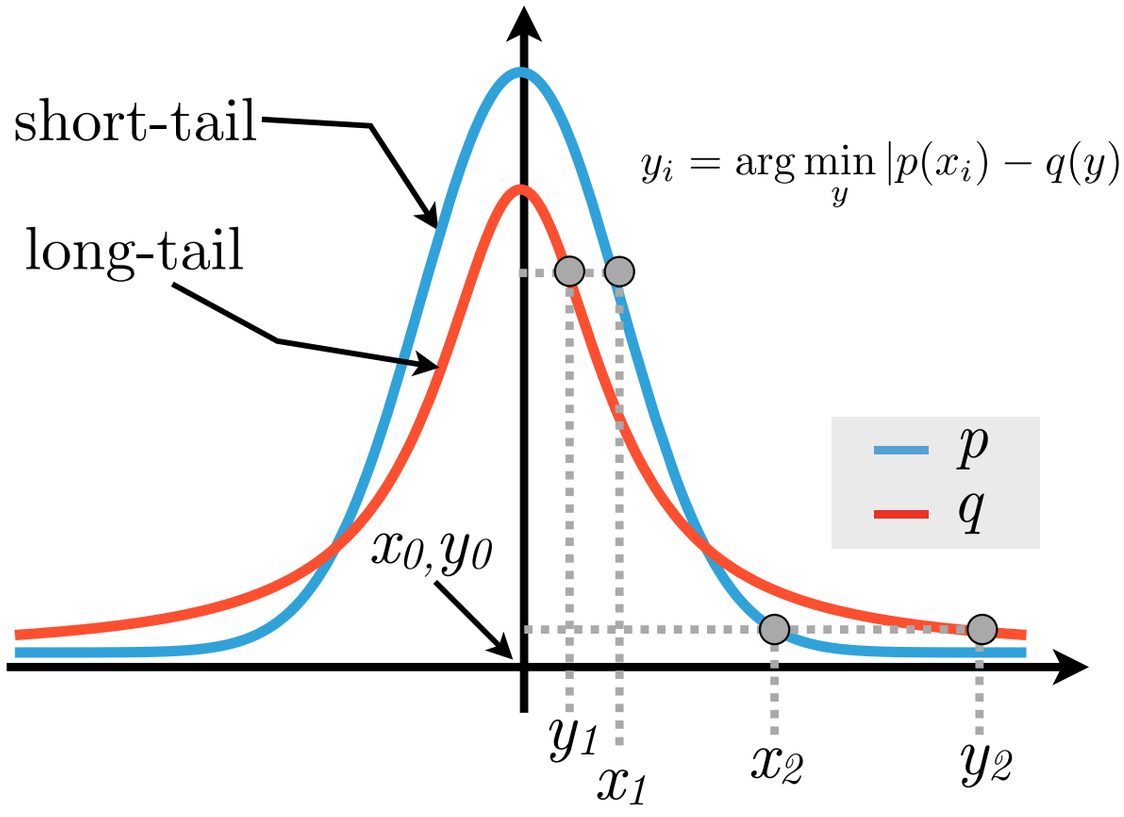
$$p_{ij} \equiv (p_{i|j} + p_{j|i})/(2N). \qquad \Rightarrow \qquad \sum_j p_{ij} > 1/(2N)$$

- t-SNE constructs a similar probability distribution in a lower dimensional latent space:

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

t-SNE

- Long tail distribution: preserves short distance information while strongly repelling two points that are far apart in the original space.



- Latent space coordinates are found by minimizing the KL divergence:

$$C(Y) = D_{KL}(p \parallel q) \equiv \sum_{ij} p_{ij} \log\left(\frac{p_{ij}}{q_{ij}}\right)$$

- Equivalent to finding equilibrium configuration of particles:

$$\partial_{y_i} C = \sum_{j \neq i} 4 p_{ij} q_{ij} Z_i (y_i - y_j) - \sum_{j \neq i} 4 q_{ij}^2 Z_i (y_i - y_j),$$
$$= F_{\text{attractive},i} - F_{\text{repulsive},i},$$

where $Z_i = 1/(\sum_{k \neq i}(1 + ||y_k - y_i||^2)^{-1})$.

attractive force comes only between nearby points in the original space

Properties of t-SNE

- Can rotate data: KL divergence is invariant under rotations in latent space.

- Results are stochastic: will depend on initial seed for gradient descent.

- Generally preserves short-distance information (preserves ordination but not actual distance between points).

- Deforms scales (not too much emphasis on the latent space)

- Computationally expensive with a $\mathcal{O}(N^2)$ scaling (can be improved to $\mathcal{O}(N \log N)$ using the Barnes-Hut method.

Performance

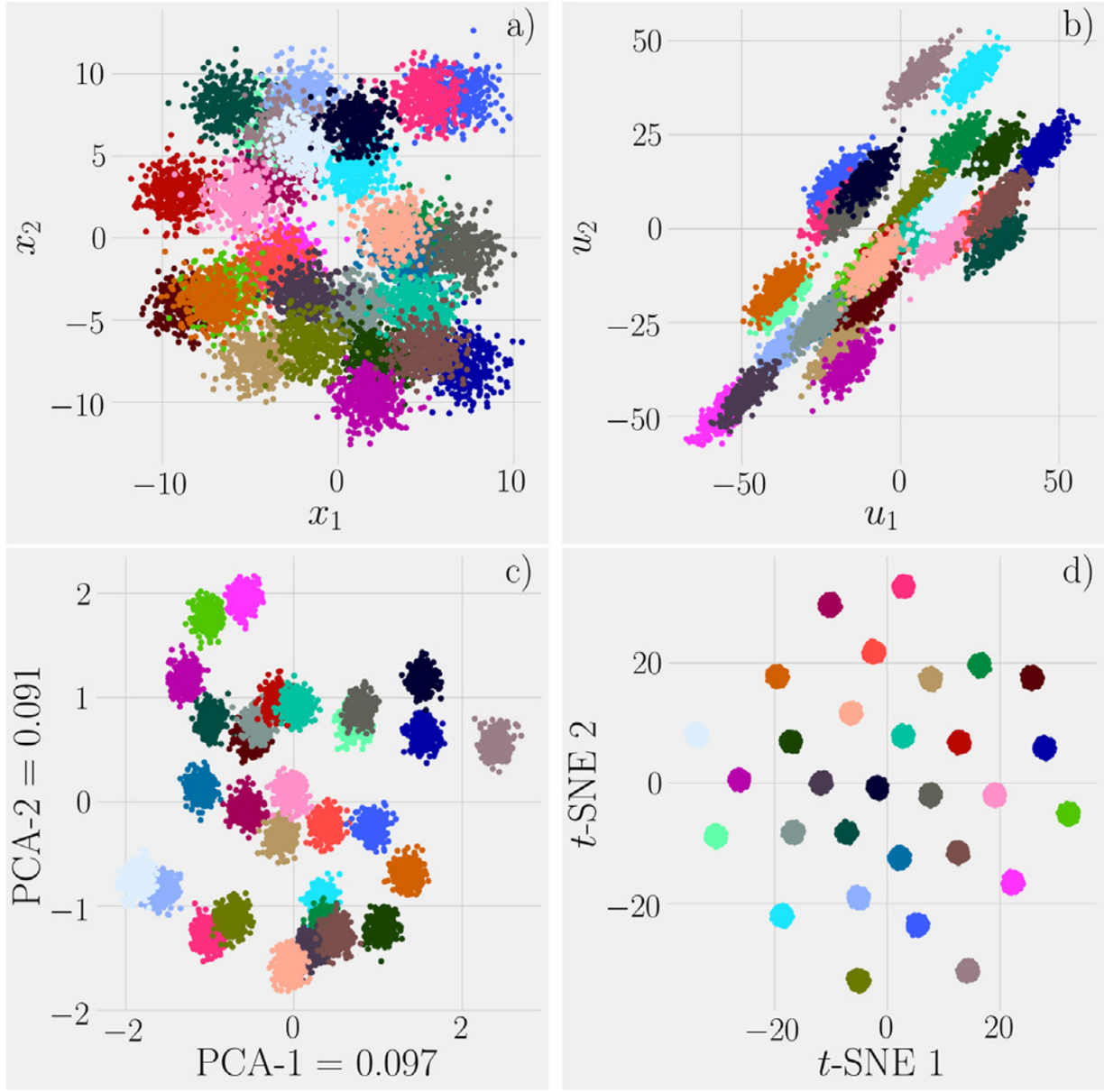


Fig. 53. Different visualizations of a Gaussian mixture formed of $K = 30$ mixtures in a $D = 40$ dimensional space. The Gaussians have the same covariance but have means drawn uniformly at random in the space $[-10, 10]^{40}$. (a) Plot of the first two coordinates. The labels of the different Gaussian are indicated by the different colors. Note that in a realistic setting, label information is of course not available, thus making it very hard to distinguish the different clusters. (b) Random projection of the data onto a 2 dimensional space. (c) Projection onto the first 2 principal components. Only a small fraction of the variance is explained by those components (the ratio is indicated along the axis). (d) t-SNE embedding (perplexity = 60, # iteration = 1000) in a 2 dimensional latent space. t-SNE captures correctly the local structure of the data.

Performance

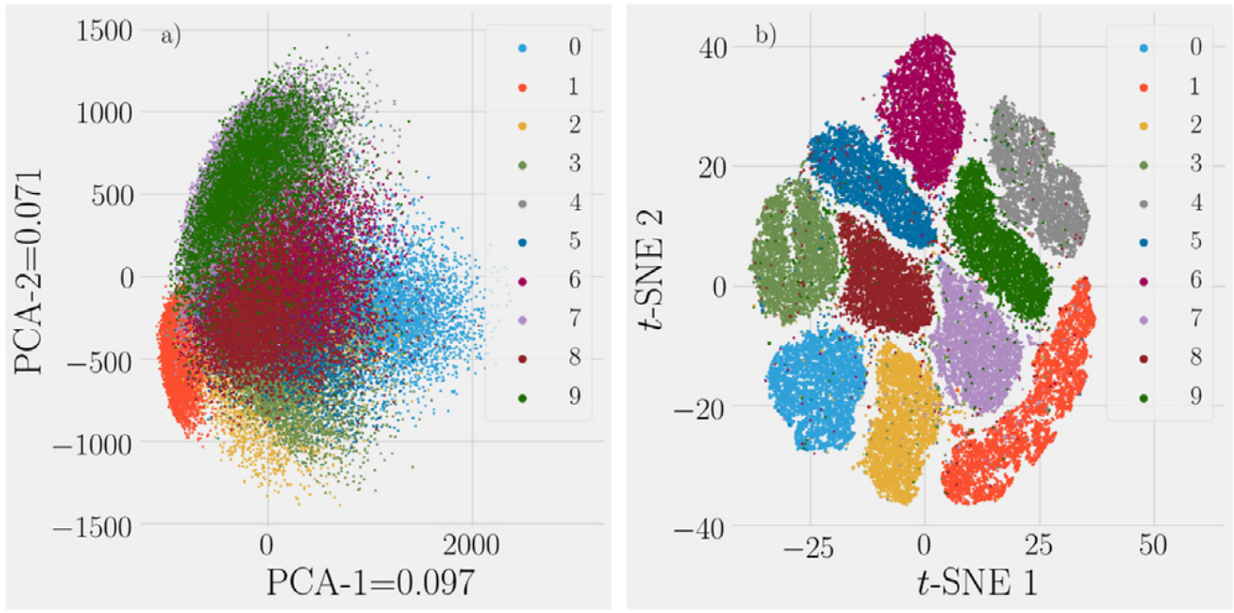


Fig. 54. Visualization of the MNIST handwritten digits training dataset (here $N = 60\ 000$). (a) First two principal components. (b) t-SNE applied with a perplexity of 30, a Barnes–Hut angle of 0.5 and 1000 gradient descent iterations. In order to reduce the noise and speed-up computation, PCA was first applied to the dataset to project it down to 40 dimensions. We used an open-source implementation to produce the results (Linderman et al., 2017), see https://github.com/KlugerLab/FIt-SNE.

t-SNE on GPU

- T-SNE is a great tool but quickly becomes slow to operate with the sklearn implementation.

- Making T-SNE fast by putting it on the GPU: https://medium.com/rapids-ai/tsne-with-gpus-hours-to-seconds-9d9c17c941db

Applications

- How much power is in your dimensions? MNIST: decay of power in components of PCA.

- Interpretability of first components: 2D Ising (magnetization)

- Visualize which variables your neural network is using: apply PCA (or other visualization methods) to different layers. Remember, deeper layers use more abstract variables.

- Disclaimer: this is a subset of visualizing techniques. If you face a visualization problem which cannot be dealt with these methods, take a more detailed look on available algorithms.

Summary

- Unsupervised learning
- Challenges of High-dimensional data
- Principal component analysis (PCA)
- Multi-dimensional scaling (MDS)
- t-stochastic neighbor embedding (t-SNE)