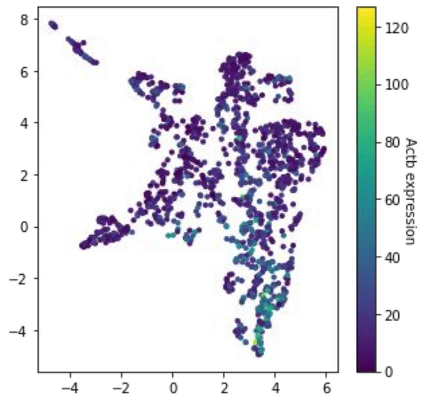
Representation learning

Deep learning overview, representation learning methods in detail (sammons map, t-sne), the backprop algorithm in detail, and regularization and its impact on optimization.

Dimensionality reduction
Word2vec
PCA
Sammon's map
Regularization
t-SNE
Factorized Embeddings
Latent variable models
ALI/BiGAN

Slides by Joseph Paul Cohen 2020



Dim reduction overview

We would like a mapping from R^{100} (or anything) to R^2 to visualize it: aka: encoding, code, embedding, representation

Invertible vs non-invertible:
If we allow for the loss of information we cannot have a lossless reconstruction.
Some methods just preserve distances, no mapping learned.

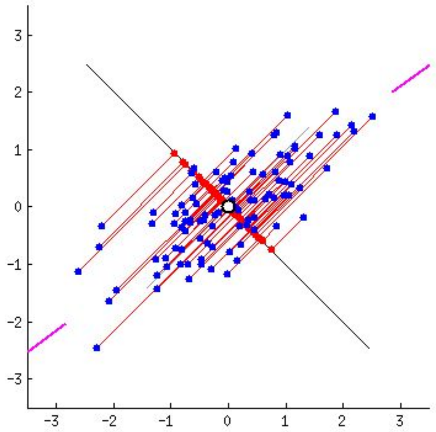
PCA for dim reduction (quick recap)

$$X \in R^{n \times k}, A \in R^{m \times n}$$

$$z = A^T x$$

$$z = E(x) = [a_0, a_1]^T x$$

A is the eigenvectors of X such that $x = AA^T x$, A's columns are orthogonal, and the columns of A form a basis which encodes the most variance.



In 2d data this vector captures the most variance

word2vec

Presented at NeurIPS 2013

Strategy to learn representations for word tokens given their context.

Relevant to problems where context defines concepts (with redundancy)

Tomas Mikolov

Distributed Representations of Words and Phrases and their Compositionality

Tomas Mikolov
Google Inc.
Mountain View
mikolov@google.com

Ilya Sutskever
Google Inc.
Mountain View
ilyasu@google.com

Kai Chen
Google Inc.
Mountain View
kai@google.com

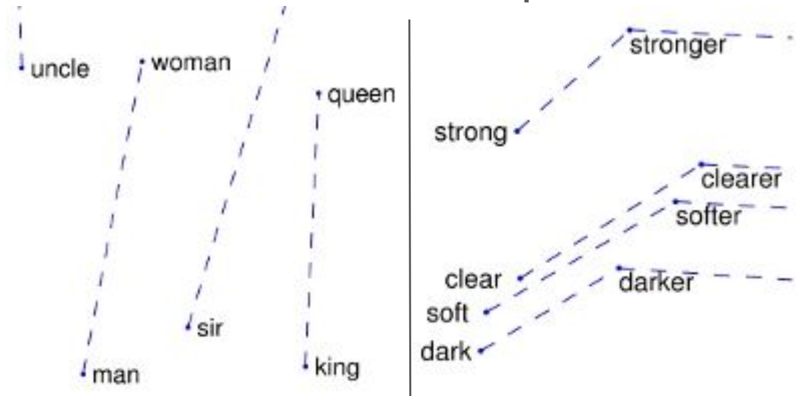
Greg Corrado

Jeffrey Dean

What to do with word embeddings?

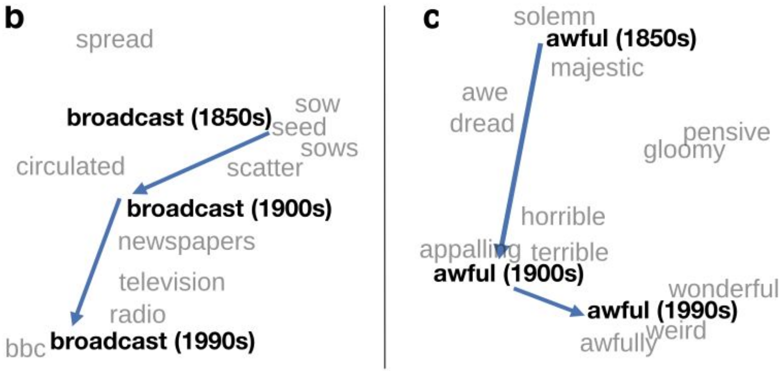
- We can compose them to create paragraph embeddings (bag of embeddings).
- Use in place of words for an RNN
- Augment learned representations on small datasets

Study the compositionality of the learned latent space



[Mikolov, 2013] [Pennington, 2014]

Study how the meaning between two texts varies (or hospitals, or doctors)?



[Cultural Shift or Linguistic Drift, Hamilton, 2016]

Token representations

One-hot encoding: binary vector per token

Example:

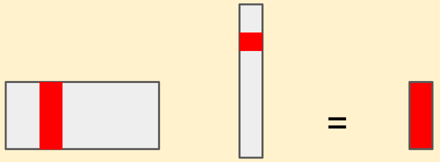
cat = [0 0 1 0 0 0 0 0 0 0 0 0 0 0 … 0]

dog = [0 0 0 0 1 0 0 0 0 0 0 0 0 0 … 0]

house = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 … 0]

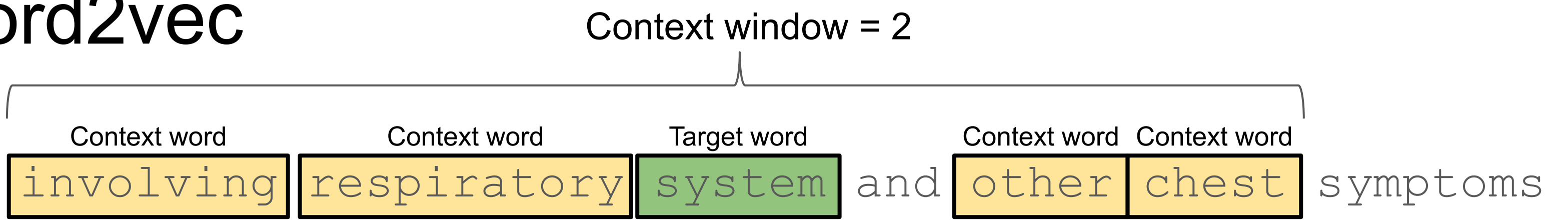
Note!

If x is one hot
The dot product of Wx
= a Single column of W

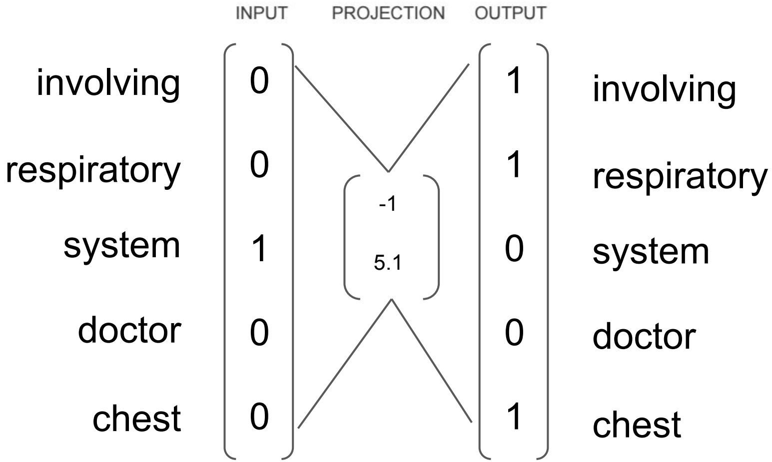


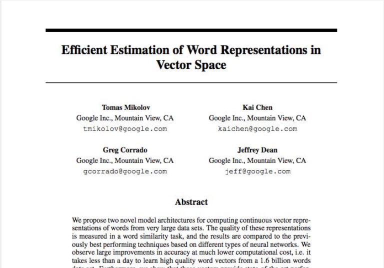
M x N   N x 1   M x 1

word2vec



1. Each word is a training example
2. Each word is used in many contexts
3. The context defines each word




Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

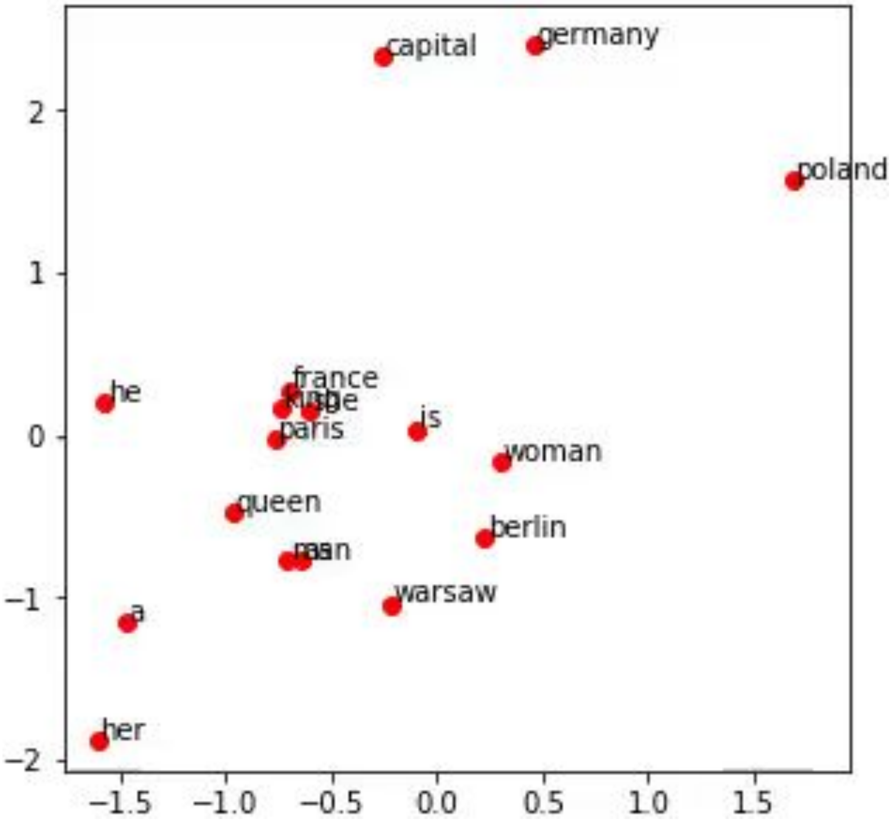
Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

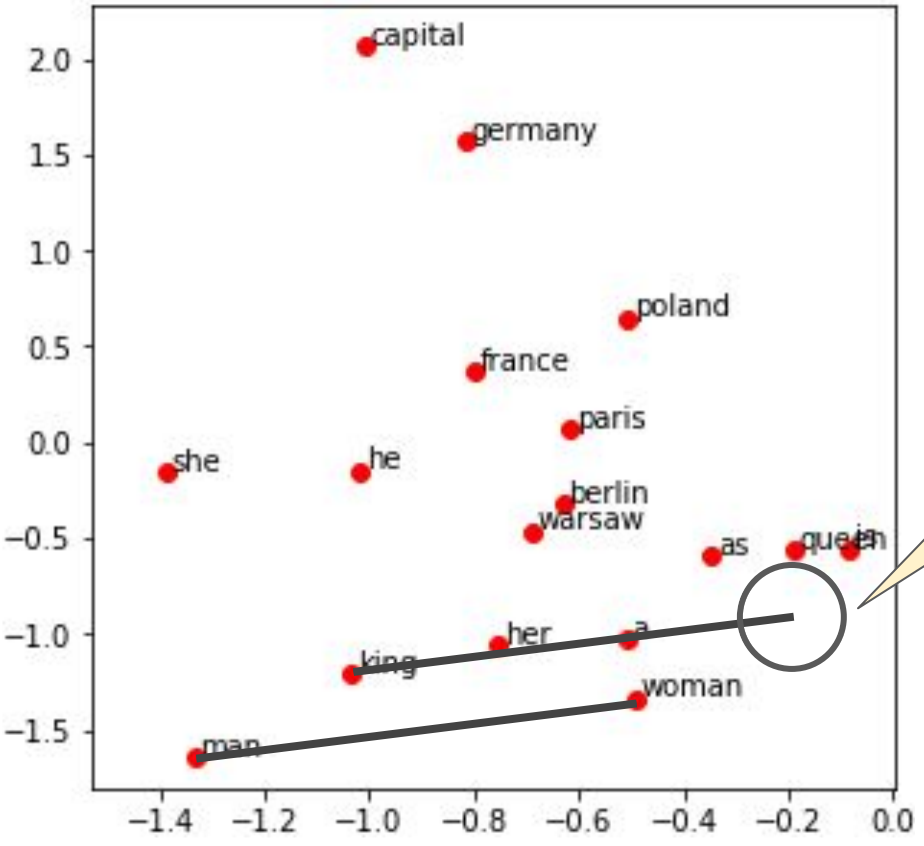
Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words

Mikolov, Efficient Estimation of Word Representations in Vector Space, 2013



Learning in progress

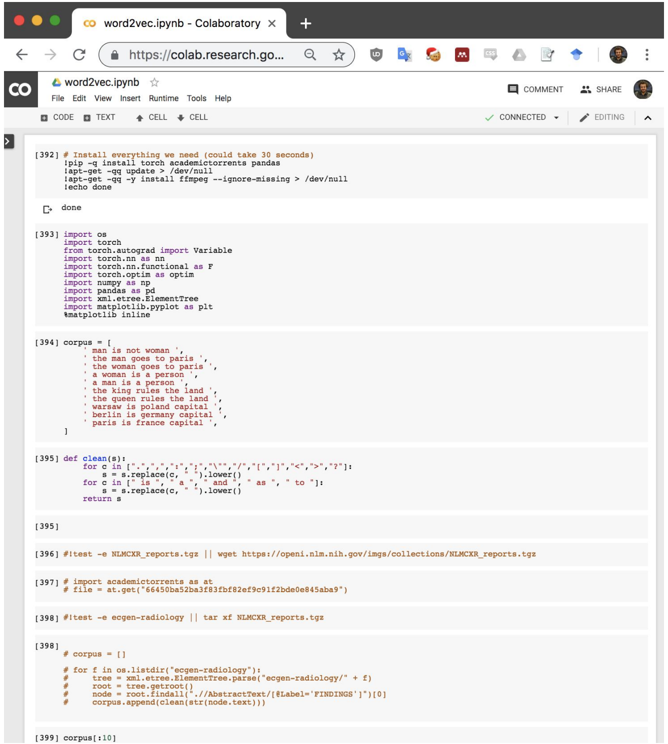


The point that is closest is queen!

king + (woman - man) = ?

Try it yourself!

https://colab.research.google.com/drive/1VU4mm_DThBaQc9t0Cf6ajjHQDEw-Q1H2



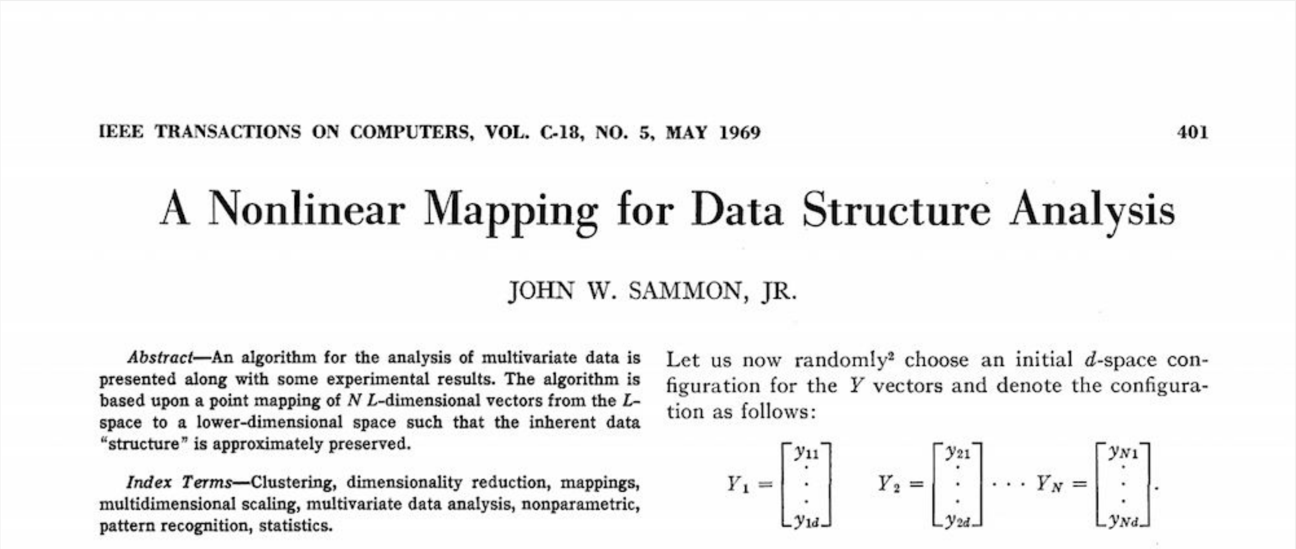
Sammon's map

Described by John W. Sammon in 1969

Method of non-linear dim reduction based on gradient descent.

Basic method of preserving distances in a low dim space.

John W. Sammon



IEEE TRANSACTIONS ON COMPUTERS, VOL. C-18, NO. 5, MAY 1969 401

A Nonlinear Mapping for Data Structure Analysis

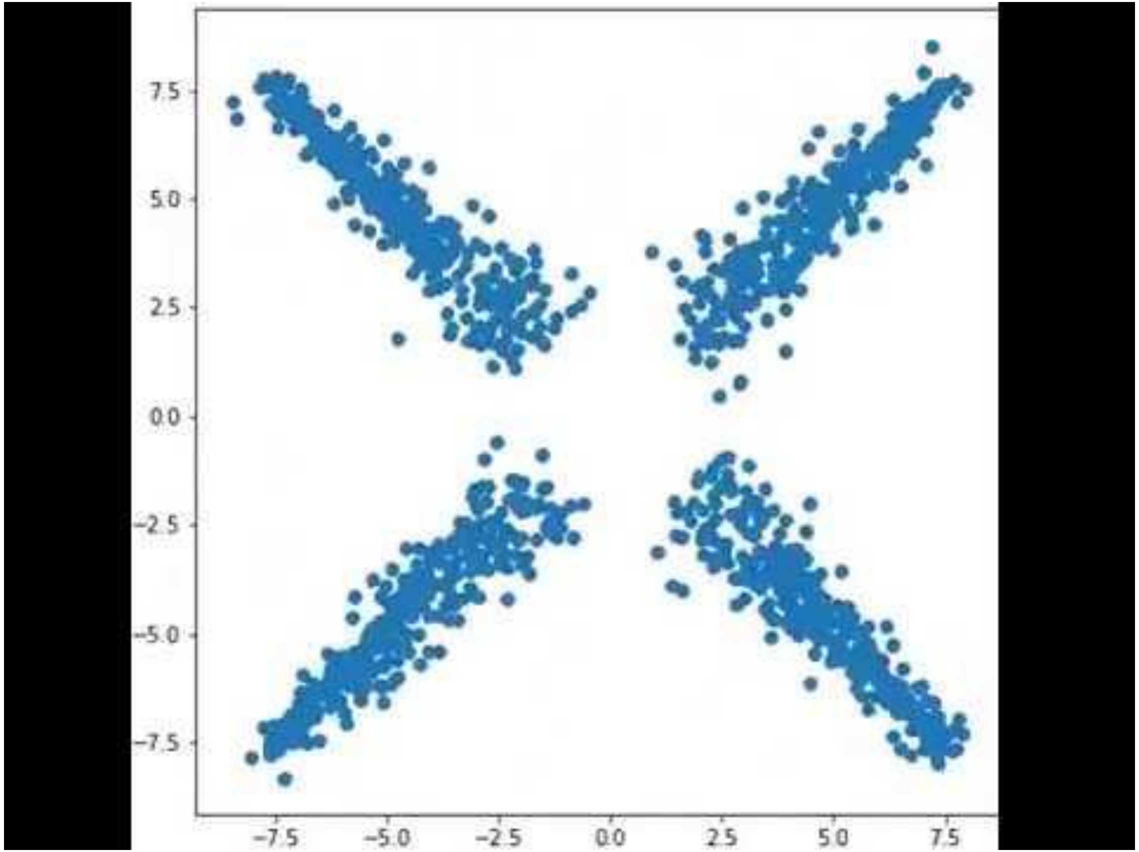
JOHN W. SAMMON, JR.

Abstract—An algorithm for the analysis of multivariate data is presented along with some experimental results. The algorithm is based upon a point mapping of N L-dimensional vectors from the L-space to a lower-dimensional space such that the inherent data "structure" is approximately preserved.

Index Terms—Clustering, dimensionality reduction, mappings, multidimensional scaling, multivariate data analysis, nonparametric, pattern recognition, statistics.

Let us now randomly[2] choose an initial d-space configuration for the Y vectors and denote the configuration as follows:

$$Y_1 = \begin{bmatrix} y_{11} \\ \vdots \\ y_{1d} \end{bmatrix} \qquad Y_2 = \begin{bmatrix} y_{21} \\ \vdots \\ y_{2d} \end{bmatrix} \cdots Y_N = \begin{bmatrix} y_{N1} \\ \vdots \\ y_{Nd} \end{bmatrix}.$$



Colab Notebook

https://colab.research.google.com/drive/1FDJ2FlVfN5PYYrNKEW2w48_BuSknhKif

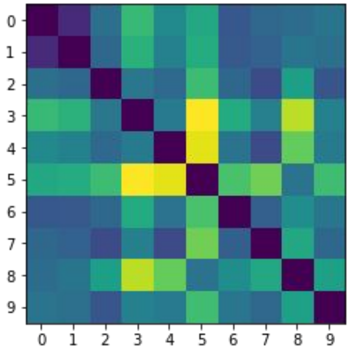
First: a basic non-linear dimensionality reduction

Learn a representation that maintains pairwise distances.

$$C = \sum_{i<j} (d^*_{ij} - d_{ij})^2$$

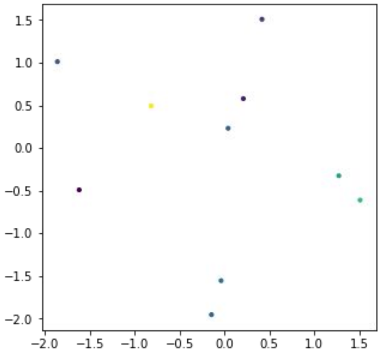
$$d^*_{ij}$$

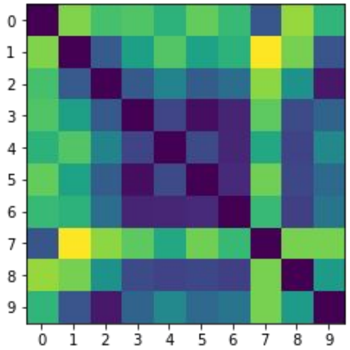
= distance function (or matrix) that you want to represent



$$d_{ij}$$

= distance computed between each learned representation





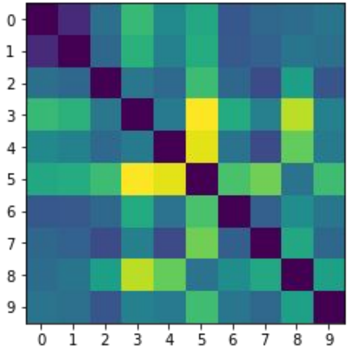
$$\text{Sammon's stress} = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}}$$

Constant!

Scale discrepancy by true distance. Small distance = more important

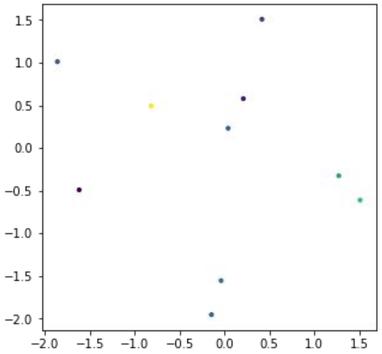
$$d^*_{ij}$$

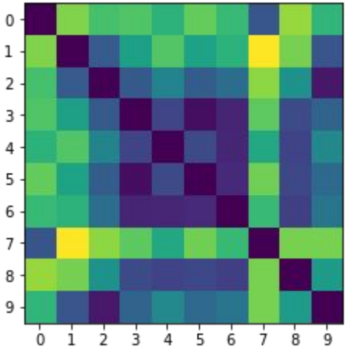
= distance function (or matrix) that you want to represent



$$d_{ij}$$

= distance computed between each learned representation





$$\text{Sammon's stress} = \frac{1}{\sum_{i<j} d^*_{ij}} \sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}}$$

d^*_{ij} = distance function (or matrix) that you want to represent

d_{ij} = distance computed between each learned representation

```
source_d = torch.pdist(source)
target_d = torch.pdist(target)

stress = (((target_d - source_d)**2)/(source_d+1)).sum()
```

pdist

`torch.nn.functional.pdist(`*input*, *p=2*`)` → Tensor

Computes the p-norm distance between every pair of row vectors in the input. This is identical to the upper triangular portion, excluding the diagonal, of *torch.norm(input[:, None] - input, dim=2, p=p)*. This function will be faster if the rows are contiguous.

If input has shape $N \times M$ then the output will have shape $\frac{1}{2}N(N-1)$.

This function is equivalent to *scipy.spatial.distance.pdist(input, 'minkowski', p=p)* if $p \in (0, \infty)$. When $p = 0$ it is equivalent to *scipy.spatial.distance.pdist(input, 'hamming') * M*. When $p = \infty$, the closest scipy function is *scipy.spatial.distance.pdist(xn, lambda x, y: np.abs(x - y).max())*.

Parameters

- **input** – input tensor of shape $N \times M$.
- **p** – p value for the p-norm distance to calculate between each vector pair $\in [0, \infty]$.

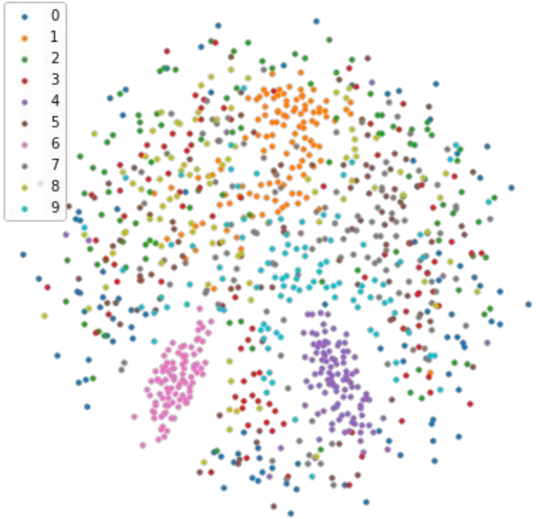
Output is a non-square (non redundant) distance matrix between vectors

```
source = torch.Tensor(data.values)
target = torch.randn(source.shape[0],2, requires_grad=True)

optimizer = torch.optim.SGD([target], lr=0.5)
optimizer.zero_grad() # get ready for new gradients

source_d = torch.pdist(source) # compute distances
target_d = torch.pdist(target) # compute distances

stress = (((target_d - source_d)**2)/(source_d+1)).sum()

stress.backward() # compute gradients for target

optimizer.step() # adjust the target tensor
```

> and MF is the "magic factor" which was determined empirically to be $MF \approx 0.3$ or 0.4. The partial derivatives are given by

This paper calls the learning rate the "magic factor" !

Exercise (regularization)



How to control the representation learned?

Adjust the objective function so that the minimum has the property you want.

$$\sum_{i<j} \frac{(d^*_{ij} - d_{ij})^2}{d^*_{ij}} + \sum_{i<j} \mathbb{I}(y_i = 6 \text{ and } y_j = 6) d_{ij}$$

```
dloss += 0.01*torch.pdist(target[label==6]).mean()
```

Discussion

Simple cases? hard cases?

What does a learning rate over 1 mean?

What are the drawbacks of this sammon's map?

What does regularization change about training?

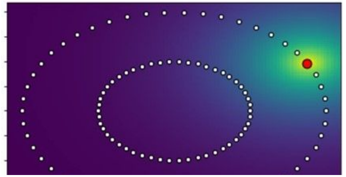
t-SNE

TL;DR: Sammon's map but distances delay exponentially

$p_{j|i}$ = conditional probability that x_i is next to x_j given a Gaussian centered at x_i

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

Ratio between distances weighted by source data distance.
Drives p and q to be equal but only for nearby points.



data space

embedding space

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

$$q_{j|i} = \frac{\exp(-||x_i - x_j||^2))}{\sum_{k\neq i}\exp(-||x_i - x_k||^2))}$$

Setting the σ (perplexity)

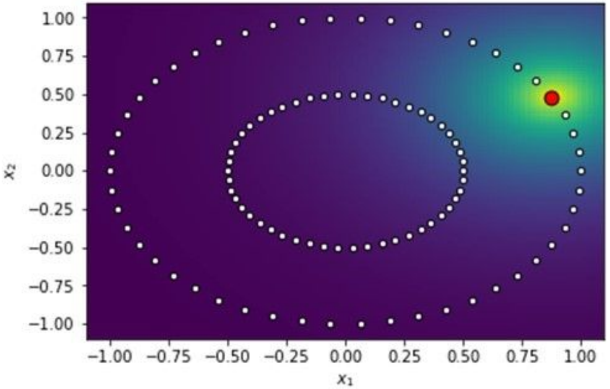
t-SNE performs a binary search for the value of σ_i that produces a P_i with a fixed perplexity.

$$\mathrm{Perp}(P_i) = 2^{H(P_i)}$$

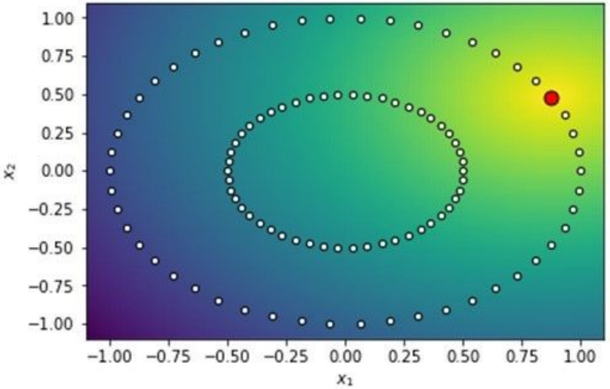
Perp=30
-> H = ~4.9

$$H(P_i(\sigma_i)) = -\sum_j p_{j|i}(\sigma_i) \log_2 p_{i|j}(\sigma_i)$$

Data space
(but in 2d)



Small σ



Large σ

Image from: https://www.dataminingapps.com/2019/11/a-refresher-on-t-sne/

Discussion

How does t-SNE differ from sammon's map?

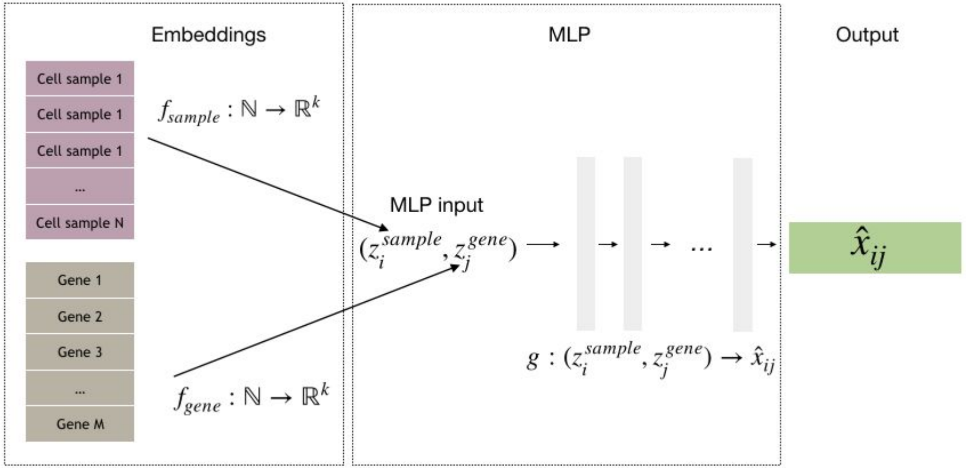
Which distances are meaningful?

Factorized Embeddings

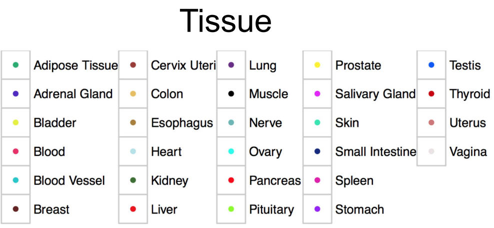
TL;DR: two spaces of non-linear embeddings.
Conditioned on each other to predict data.

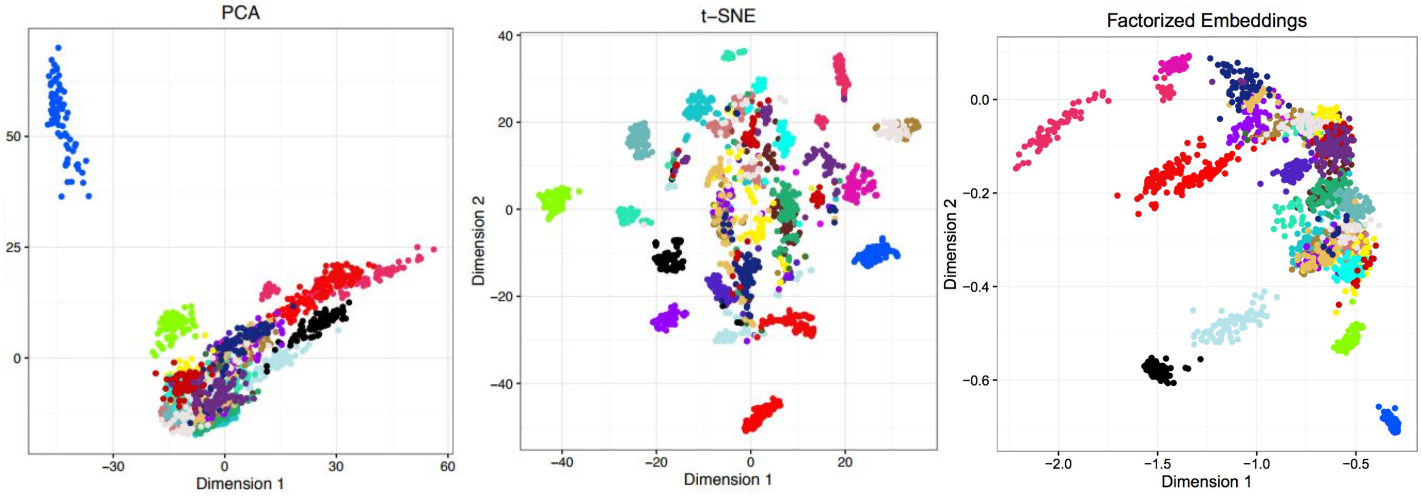
$$\hat{x}_{ij} = g(z_i^{\text{sample}}, z_j^{\text{gene}})$$

$$\min_z (\hat{x}_{ij} - x_{ij})^2$$



GTEx Dataset: 8,910 samples, 56,000 genes



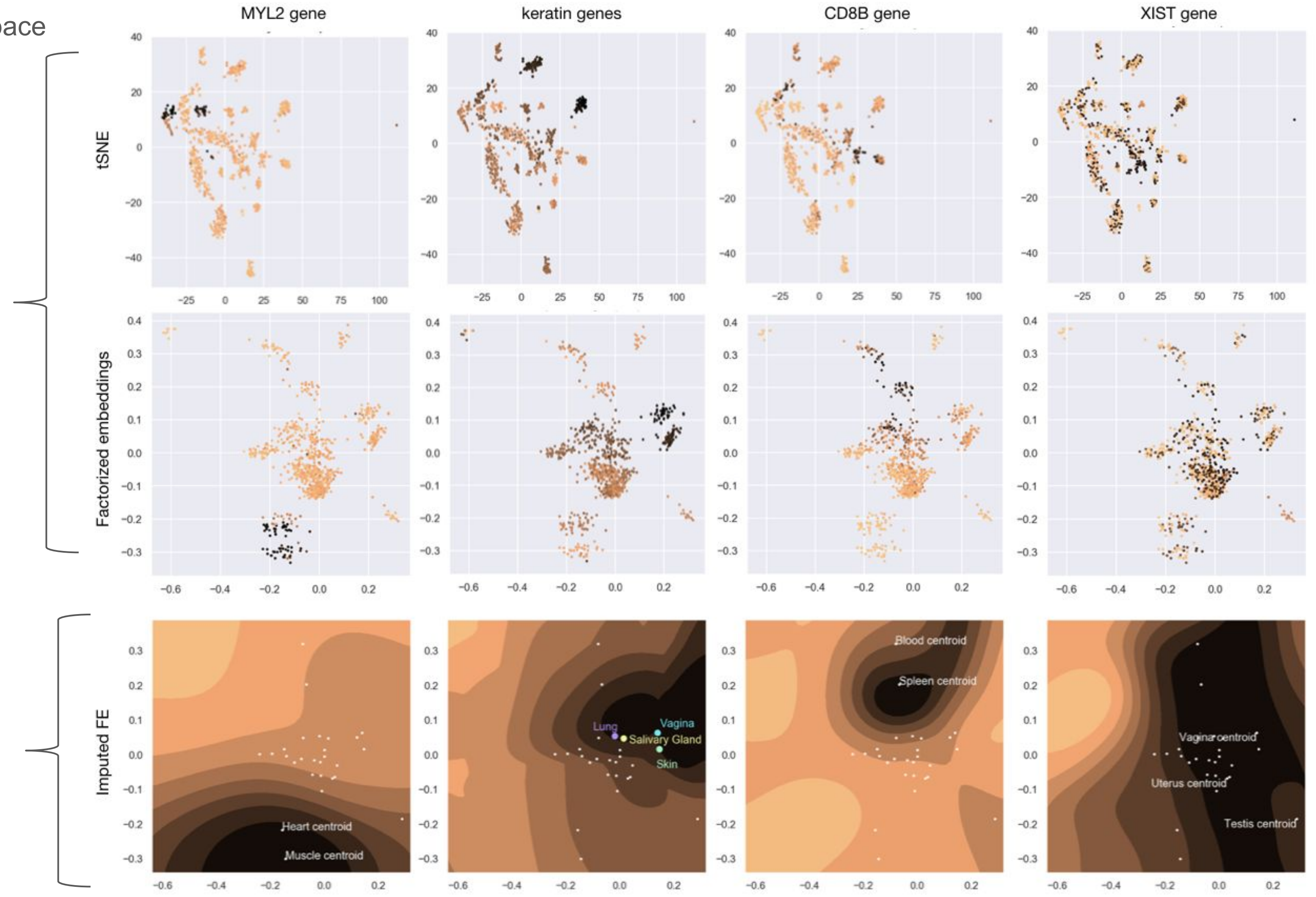


[Trofimov et al. ICML WCB 2017]

Evaluating sample space

Color represents expression predicted when conditioned on a gene

Value predicted for entire space



Latent variable models

We learn a mapping from a latent variable z to a complicated x

$$p(x) = \int p(x, z) dz$$

where

$$p(x, z) = p(x|z)p(z)$$

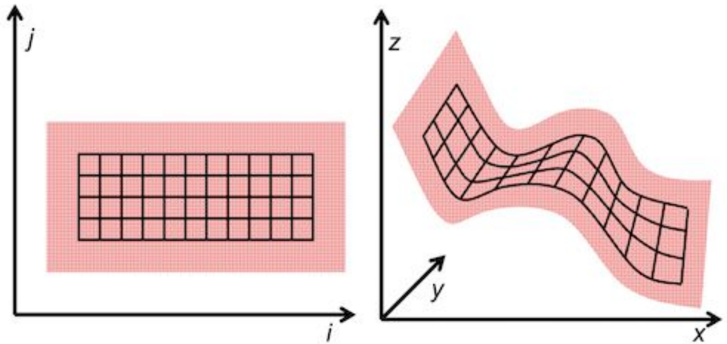


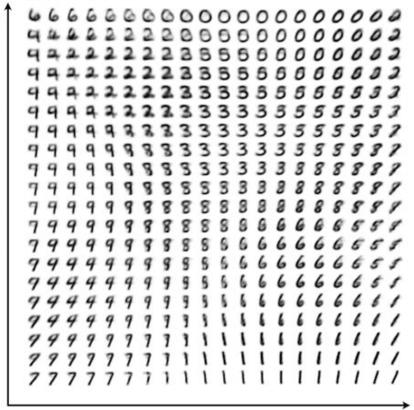
Image from Ward, A. D, 3D Surface Parameterization Using Manifold Learning for Medial Shape Representation

$$p(x|z) = g_\theta(z)$$

$p(z)$ = Something simple like a Gaussian

The conditional prob is modeled by a neural network and the latent space is a distribution we understand.

MNIST:



ALI/BiGAN

ICLR 2017. Two papers, same idea.

Matches joint distribution p(x,z).

Trains an encoder and decoder.

Vincent Dumoulin

Jeff Donahue

ADVERSARIALLY LEARNED INFERENCE

Vincent Dumoulin[1], **Ishmael Belghazi**[1], **Ben Poole**[2]
Olivier Mastropietro[1], **Alex Lamb**[1], **Martin Arjovsky**[3]
Aaron Courville[1†]
[1] MILA, Université de Montréal, firstname.lastname@umontreal.ca.
[2] Neural Dynamics and Computation Lab, Stanford, poole@cs.stanford.edu.
[3] New York University, martinarjovsky@gmail.com.
[†]CIFAR Fellow.

ABSTRACT

We introduce the adversarially learned inference (ALI) model, which jointly learns a generation network and an inference network using an adversarial process. The generation network maps samples from stochastic latent variables to the data space while the inference network maps training examples in data space to the space of latent variables. An adversarial game is cast between these two networks and a discriminative network is trained to distinguish between joint latent/data-space

ADVERSARIAL FEATURE LEARNING

Jeff Donahue
jdonahue@cs.berkeley.edu
Computer Science Division
University of California, Berkeley

Philipp Krähenbühl
philkr@utexas.edu
Department of Computer Science
University of Texas, Austin

Trevor Darrell
trevor@eecs.berkeley.edu
Computer Science Division
University of California, Berkeley

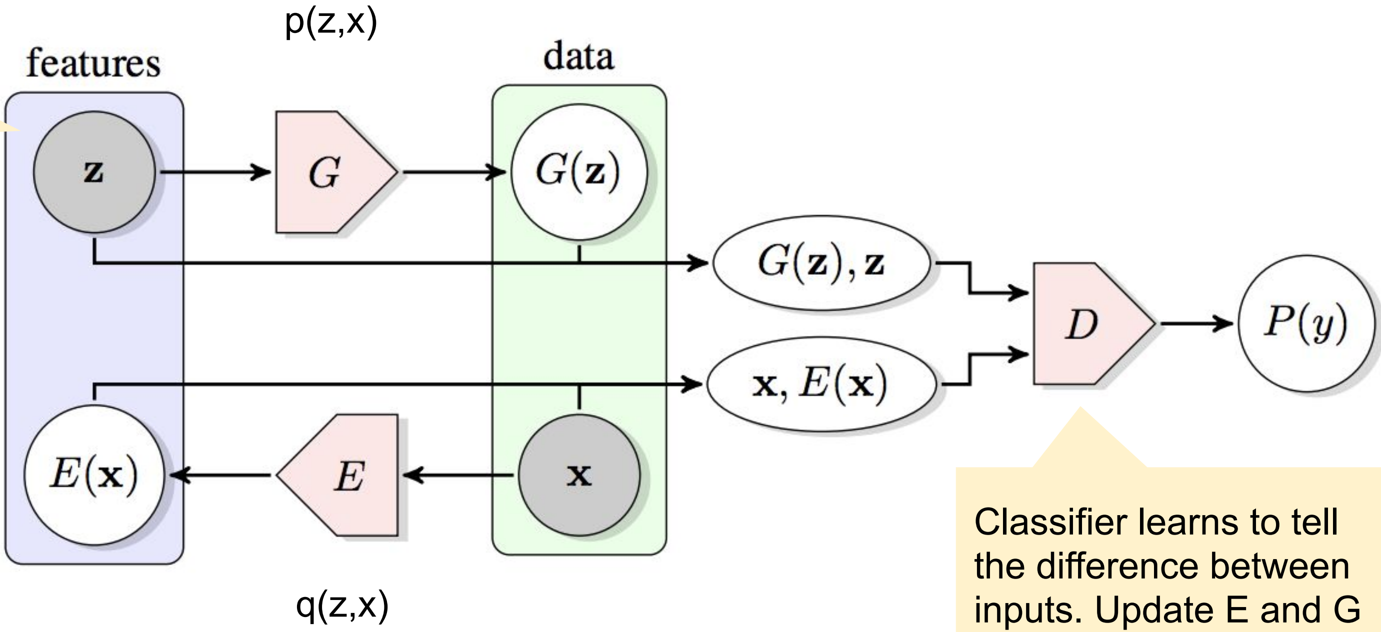
ABSTRACT

The ability of the Generative Adversarial Networks (GANs) framework to learn generative models mapping from simple latent distributions to arbitrarily complex data distributions has been demonstrated empirically, with compelling results showing that the latent space of such generators captures semantic variation in the data distribution. Intuitively, models trained to predict these semantic latent

ALI/BiGAN

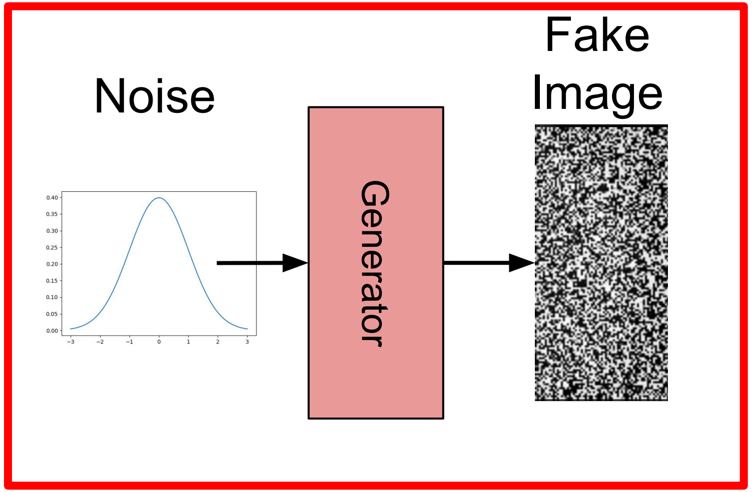
Matches the joint distribution p(z,x) with q(z,x) using an adversarial loss.

Typically a Gaussian generates points here.

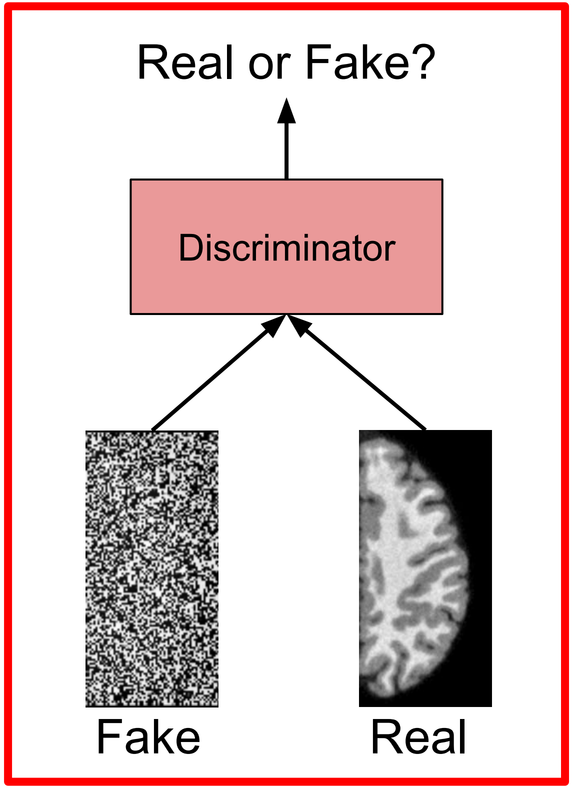


Classifier learns to tell the difference between inputs. Update E and G so D cannot distinguish.

Quick intro to adversarial distribution matching



The generator learns to match the target distribution to fool the discriminator



Homework

1) Find a single/multi cell RNA-seq dataset compute a PCA, Sammon's Map, and t-SNE. Color points by some relevant value.

2) What are the challenges for representation learning?

3)