UiO : Institutt for informatikk
Det matematisk-naturvitenskapelige fakultet

INF5860 - Maskinlæring for bildeanalyse

Unsupervised Learning

Themes

- Supervised vs unsupervised learning
- t-SNE
- Autoencoders
- Variational Autoencoders

Material

t-SNE: https://youtu.be/EMD106bB2vY

Unsupervised learning / autoencoders Lecture 14 (32 mins): https://youtu.be/I-i1KBuShCc?t=32m37s

Fun and educational:

http://distill.pub/2016/misread-tsne/

https://www.oreilly.com/learning/an-illustrated-introduction-to-the-t-sne-algorithm

http://colah.github.io/posts/2014-10-Visualizing-MNIST/

Supervised learning

- We have a set of pairs **(x, y)** and we want to learn a function that maps x -> y **y=f(x)**

Y is often a class label, a number etc. some extracted/compressed information about **x**.

Supervised learning vs Unsupervised learning

- We have a set of pairs **(x, y)** and we want to learn a function that maps x -> y **y=f(x)**

Y is often a class label, a number etc. some extracted/compressed information about **x**.

- We have only **x** and we want to extract some information.

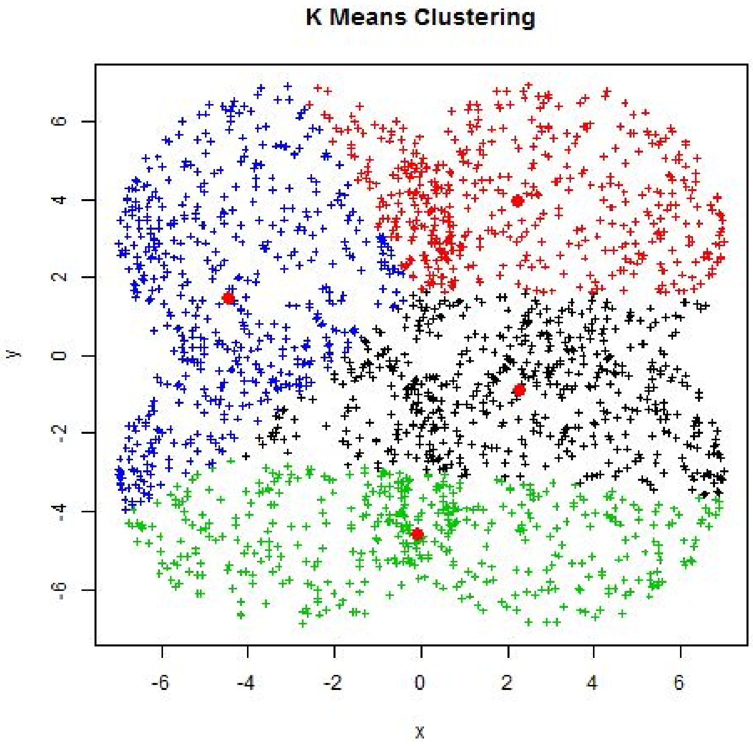
We want some compression or grouping of the data.

In many examples we want to do some grouping (classifications without “true” labels.)

Clustering by distances (k-means)

1. Start with random initial "means"
2. Label each example to the closest **mean**
3. Recalculate the means according to the means of the examples

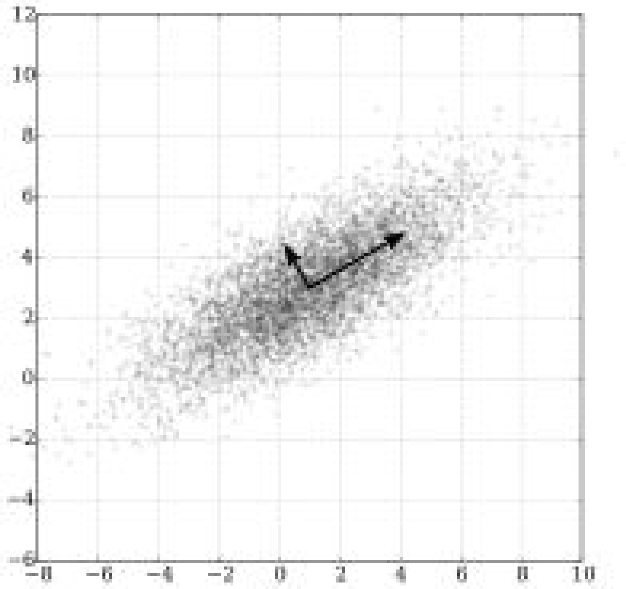
Iterate through 2. and 3. until convergence.



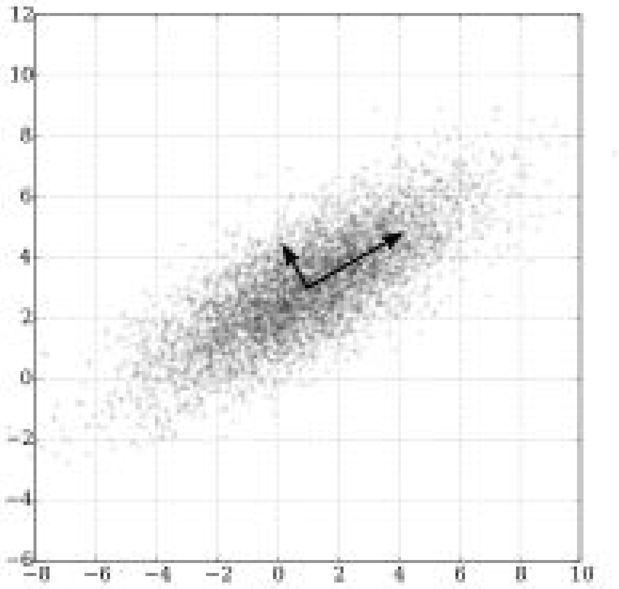
PCA can be used for compression

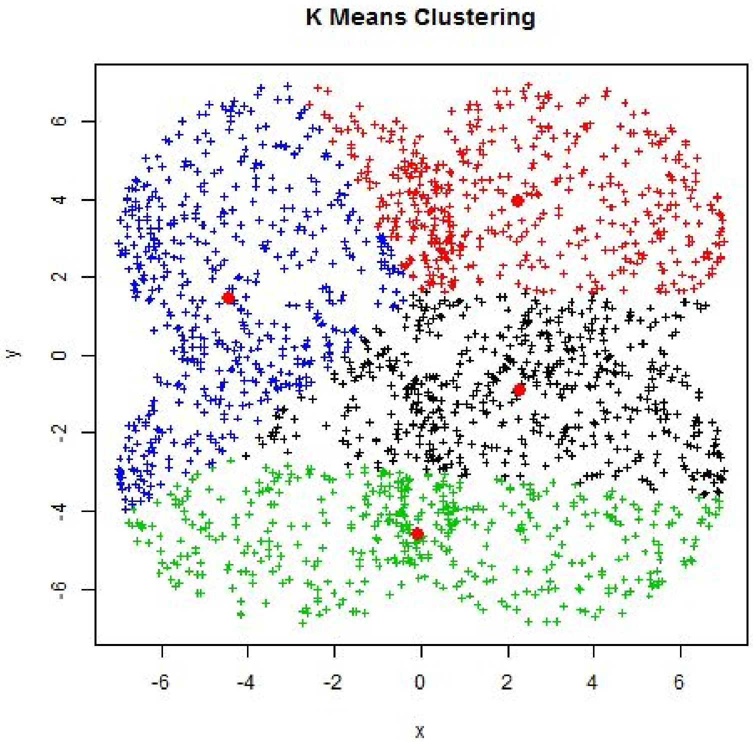
- Find the eigen-values ad eigen-vectors of the covariance matrix
- Use the k-highest eigen-values and vectors to transform the data to a lower dimension

- Keeps as much of the variance as possible



Extraction based on linear distances work well for many applications

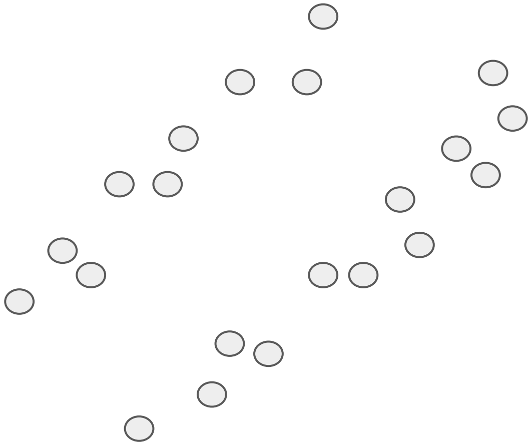




But what if distances is not the only factor

- In this settings we may want to label the samples primarily with local distances
- With gaps being an indications of a cluster

- Both PCA and k-means work bad in this case

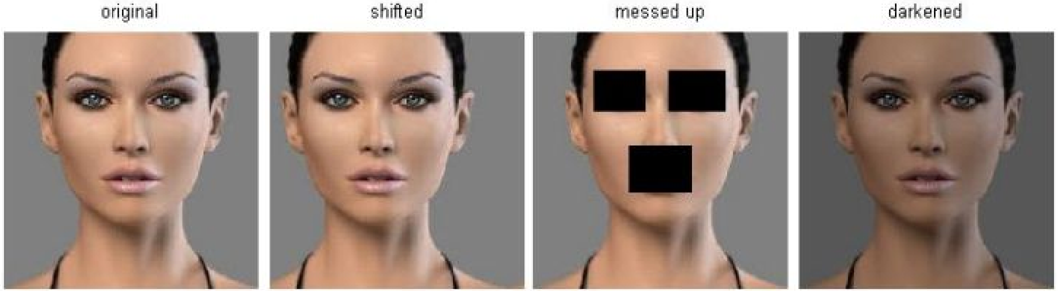


PCA can fail for compression

-

Images is often non-linear

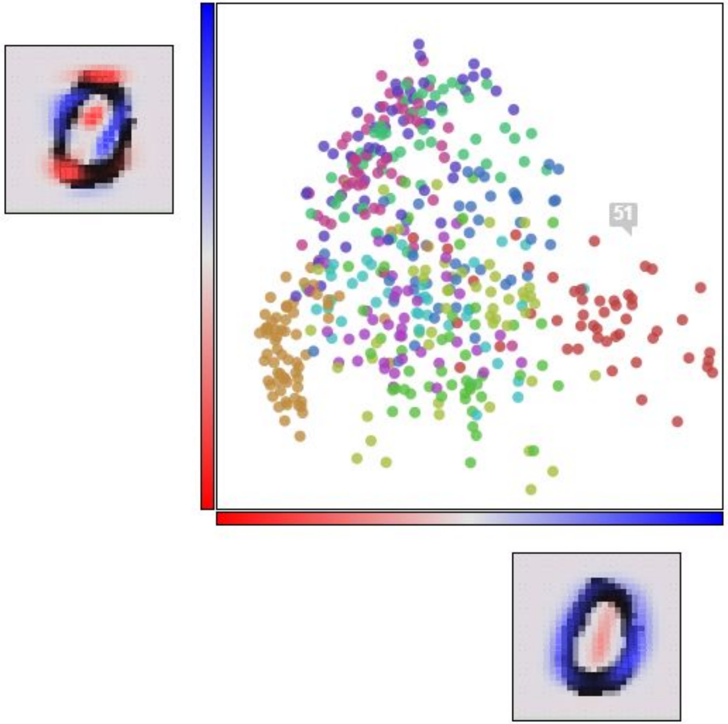
- In this settings we may want to label the samples primarily with local distances
- With gaps being an indications of a cluster



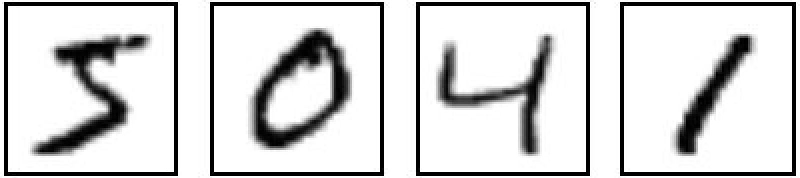
(all 3 images have same L2 distance to the one on the left)

PCA on handwritten digits

- Even on a highly standardized dataset PCA can have problems with images
- I got 22% accuracy with k-means alone
- 30% accuracy with PCA first.

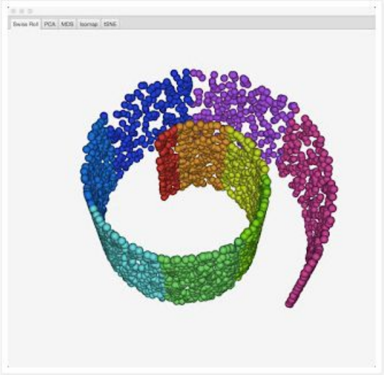


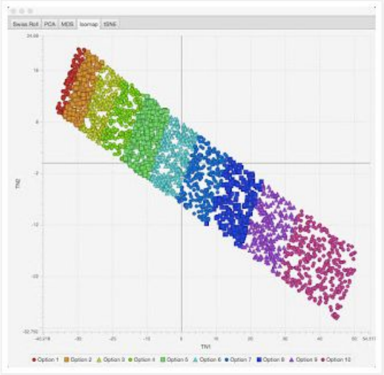
Visualizing MNIST with PCA



Stochastic Neighbor Embedding (SNE)

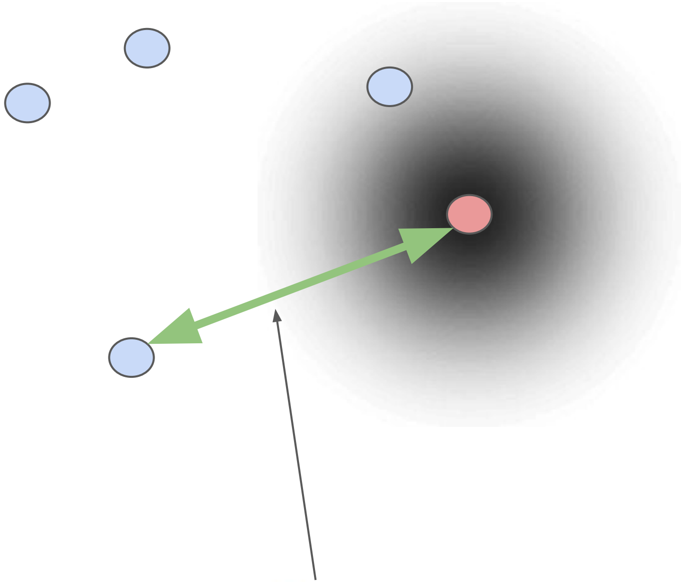
- Preserving local distances through dimensionality reduction





Probability of one point being connected to another

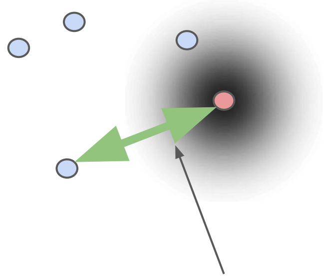
- We model distances as gaussian probability density functions

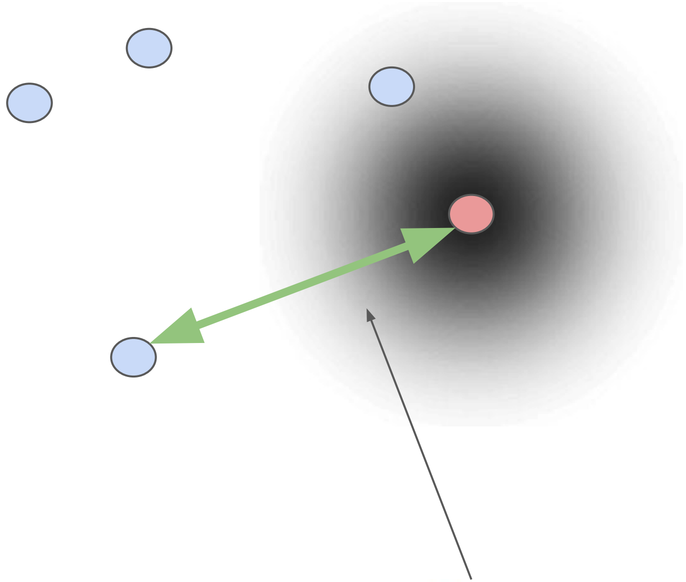


$$p_{j|i} = \frac{\exp\left(-|x_i - x_j|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-|x_i - x_k|^2 / 2\sigma_i^2\right)}$$

Probability of one point being connected to another

- We model distances as gaussian probability density functions
- We measure the distance both in the high dimension and the low dimension



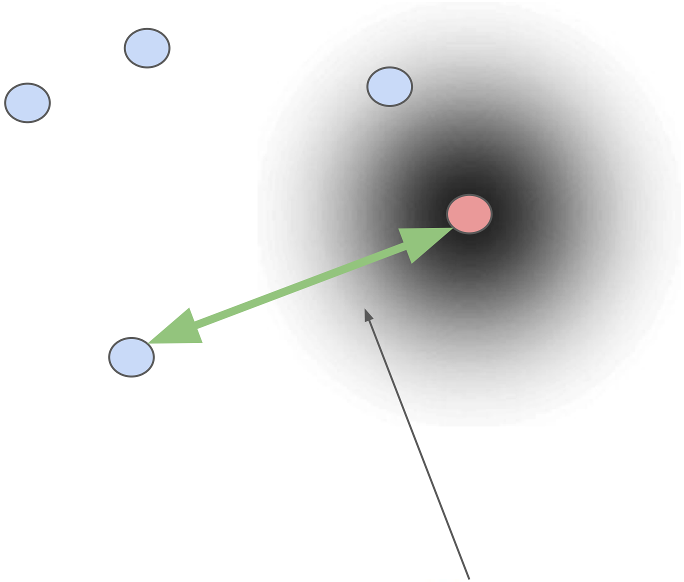


$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

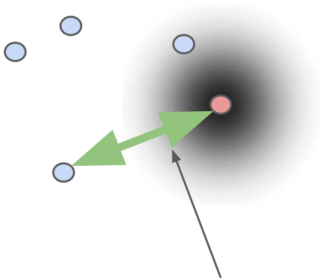
$$p_{j|i} = \frac{\exp\left(-|x_i - x_j|^2 / 2\sigma_i^2\right)}{\sum_{k \neq i} \exp\left(-|x_i - x_k|^2 / 2\sigma_i^2\right)}$$

We want the probabilities to be similar in both spaces

- Connected pairs should be connected in both spaces
- We can measure this with KL-divergence



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



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

We want the probabilities to be similar in both spaces

- Connected pairs should be connected in both spaces
- We can measure this with KL-divergence

$$KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

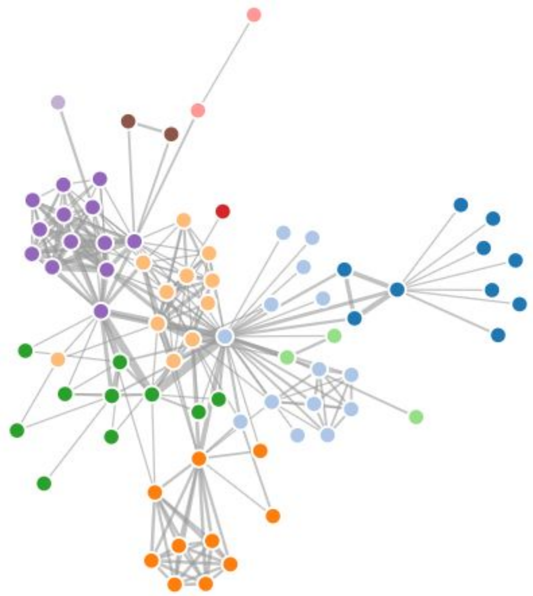
$$q_{j|i} = \frac{exp(-||y_i - y_j||^2)}{\sum_{k \neq i} exp(-||y_i - y_k||^2)}$$

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

Minimizing KL-Divergence preserves distributions

- Connected pairs should be connected in both spaces
- We can measure this with KL-divergence
- Minimizing KL-Divergence, means the distributions are as overlapping as possible

$$KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$



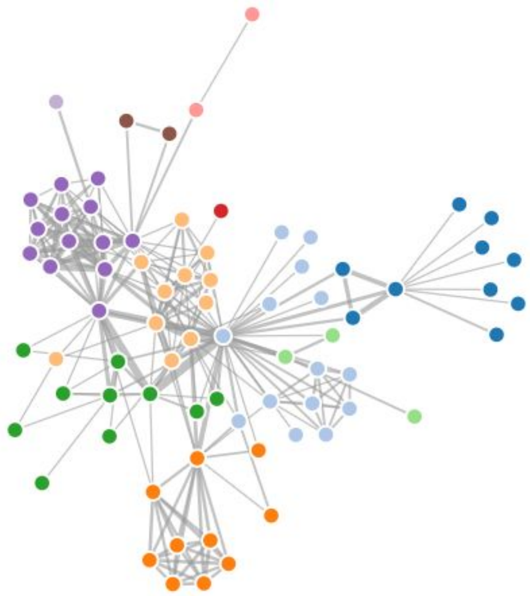
We want the probabilities to be similar in both spaces

- KL-divergence is not the same in both directions
- This means that we care more about the distances that are close in high-dimension

$$KL(P||Q) = \sum_{i,j} p_{ij} \, \log \frac{p_{ij}}{q_{ij}}$$

Minimizing KL-Divergence

- Optimizing the position in the low-dimensional space directly
- Simple gradient
- Optimizing the local positions are a non-linear problem, but solving it iteratively still works well.



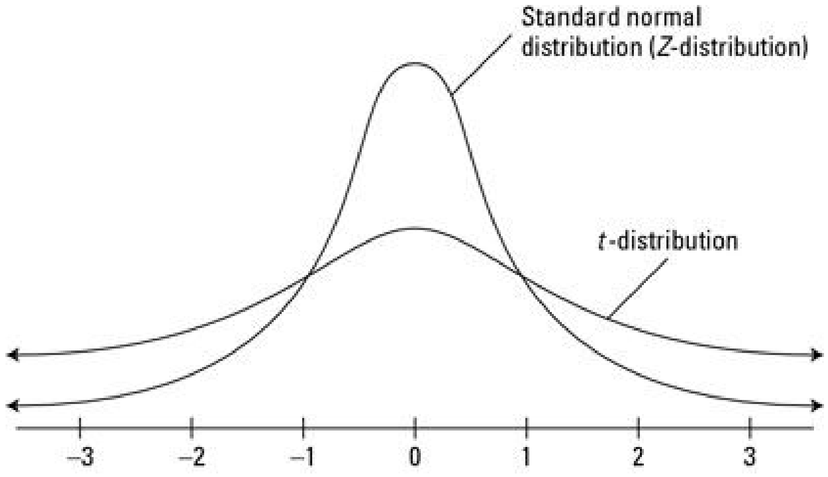
$$KL(P||Q) = \sum_{i,j} p_{ij} \log \frac{p_{ij}}{q_{ij}}$$

$$\frac{\partial KL}{\partial y_i} = \sum_{j \neq i} (p_{j|i} - q_{j|i} + p_{i|j} - q_{i|j})(y_i - y_j)$$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

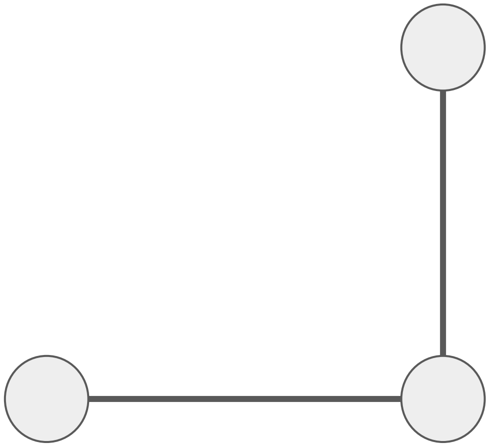
- Still using Gaussian distribution for high-dimensional space
- Using t-Distribution in low-dimensional space

- Enlarging long distances in high dimensional space...



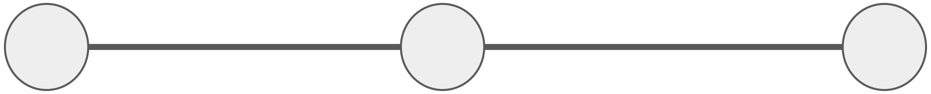
t-SNE - Crowding problem

- Keeping the local distances constant



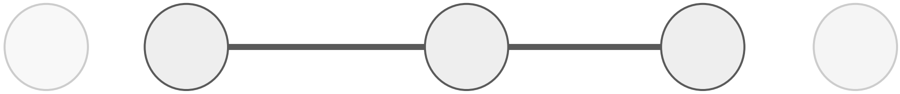
t-SNE - Crowding problem

- Keeping the local distances constant
- With normal distribution this is a problem, since also the far apart nodes wants to keep close



t-SNE - Crowding problem

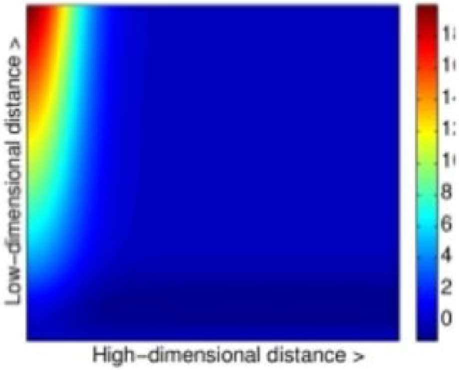
- Keeping the local distances constant
- With normal distribution this is a problem, since also the far apart nodes wants to keep close
- Squeezing everything together

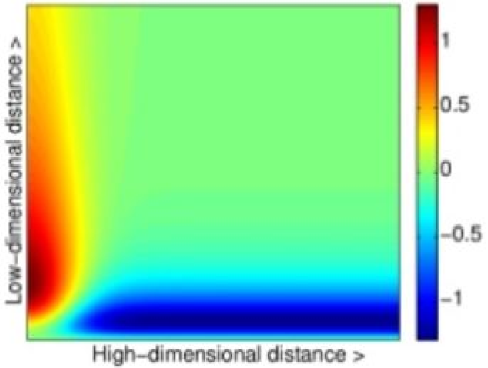


- t-SNE “allow” them to stay apart

t-SNE - Crowding problem

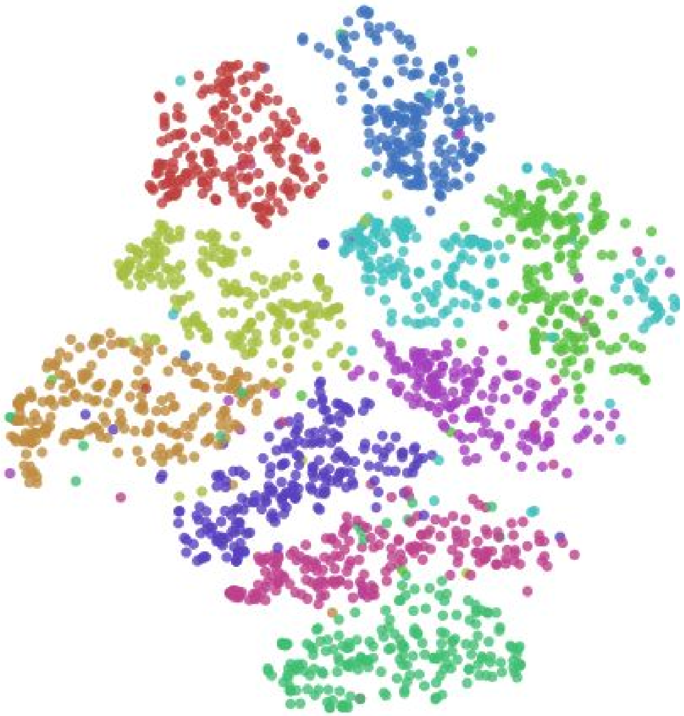
- Visualizing gradient of t-SNE and SNE
- SNE ignores points with large high-dimensional distance
- t-SNE increase those distances
- t-SNE effect those with low high-dimensional distance and large low-dimensional distance most





Visualizing MNIST with t-SNE

- We can see that t-SNE give better separation of handwritten digits than PCA



A t-SNE plot of MNIST

t-SNE for unsupervised learning and visualization

- t-SNE are perhaps most commonly used for visualizing the effect of a deep network on a data-distribution
- Embedding of last layer of VGG show that similar concepts are grouped together

t-SNE for unsupervised learning and visualization

- t-SNE are perhaps most commonly used for visualizing the effect of a deep network on a data-distribution
- Embedding of last layer of VGG show that similar concepts are grouped together
- It is also used for unsupervised learning, e.g. in combination with k-mean

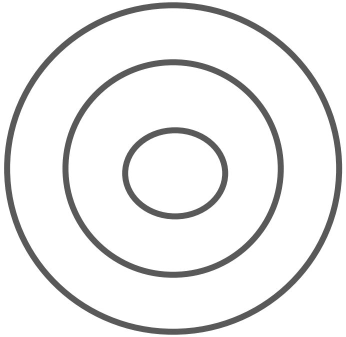
[Multifaceted Feature Visualization]

Hyper-parameter of t-SNE - Perplexity

- Searching for a sigma to give the right perplexity
- Increasing sigma in sparse areas
- Decreasing sigma in dense areas
- This means that **perplexity** is close to a measure of how many points should influence each point
- How many connected neighbors do you expect

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

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

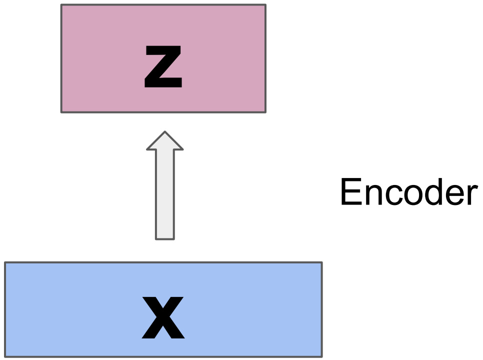


Deep learning for compression

Autoencoders

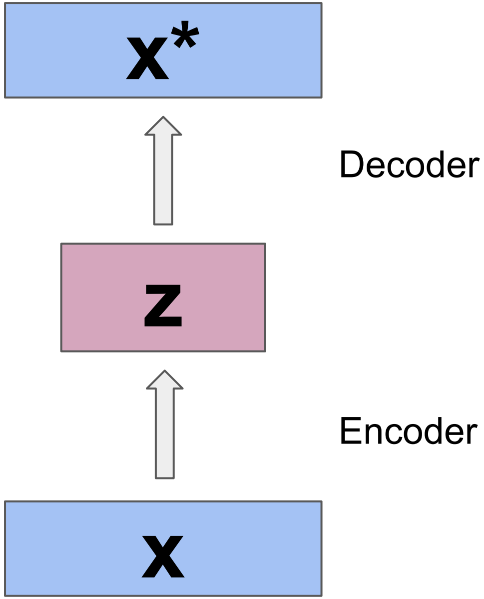
Autoencoders

- A neural network transforming the input
- Often into a smaller dimension



Autoencoders

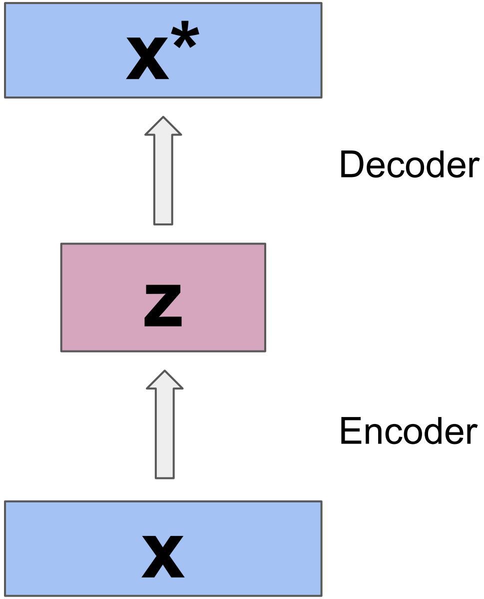
- A neural network transforming the input
- Often into a smaller dimension
- Then a decoder network reconstructs the input



Autoencoders

- A neural network transforming the input
- Often into a smaller dimension
- Then a decoder network reconstructs the input
- Restrictions are put on **z** either through loss functions, or **size**

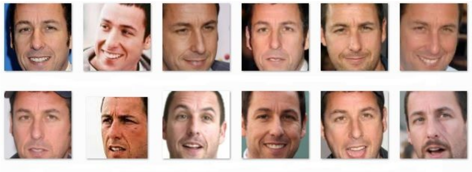
- Often used with convolutional architectures for images

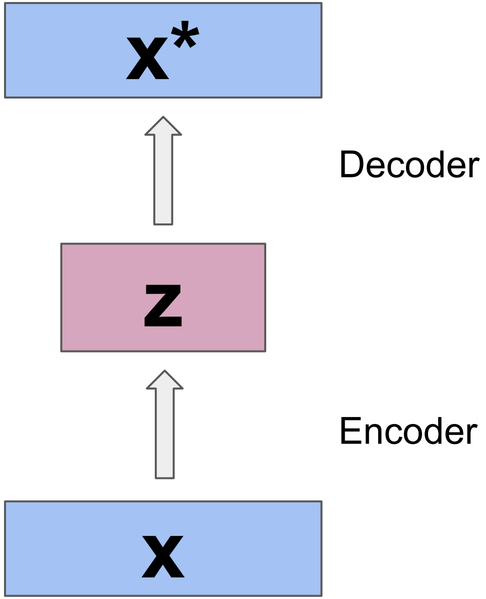


Autoencoders

- Restrictions are put on **z** either through loss functions, or **size**
- Often minimizing l2 loss:

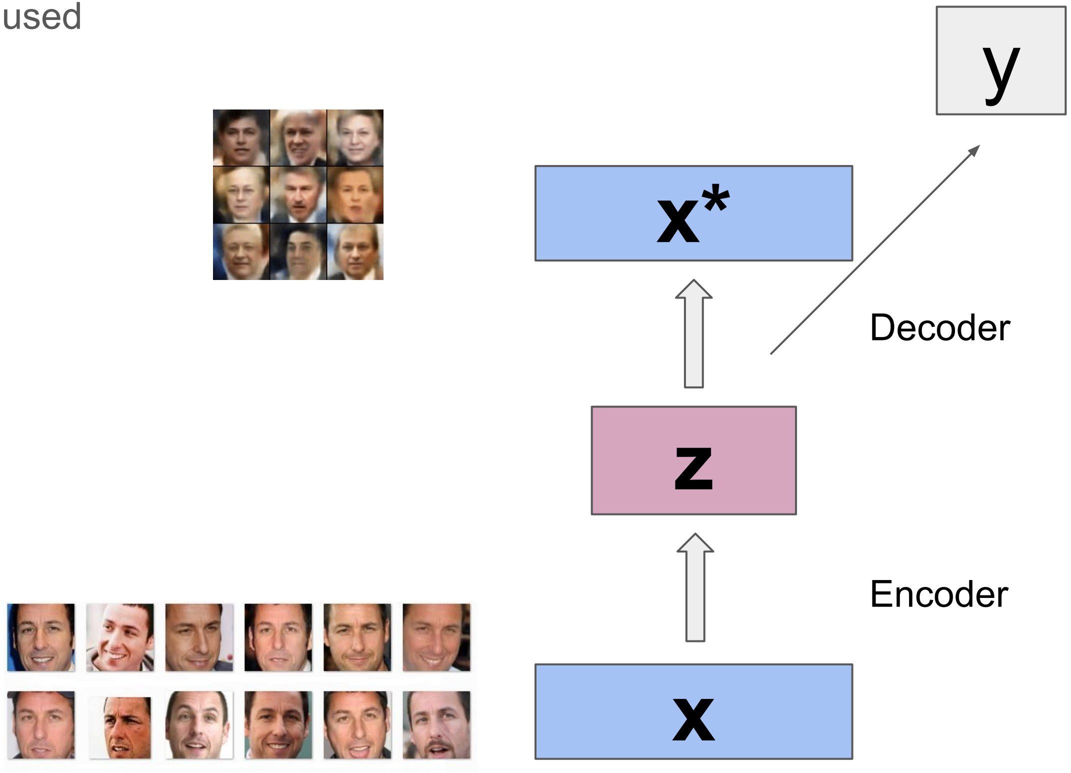
$$L(x) = (x - x*)^2$$





Autoencoders - Semi-supervised learning

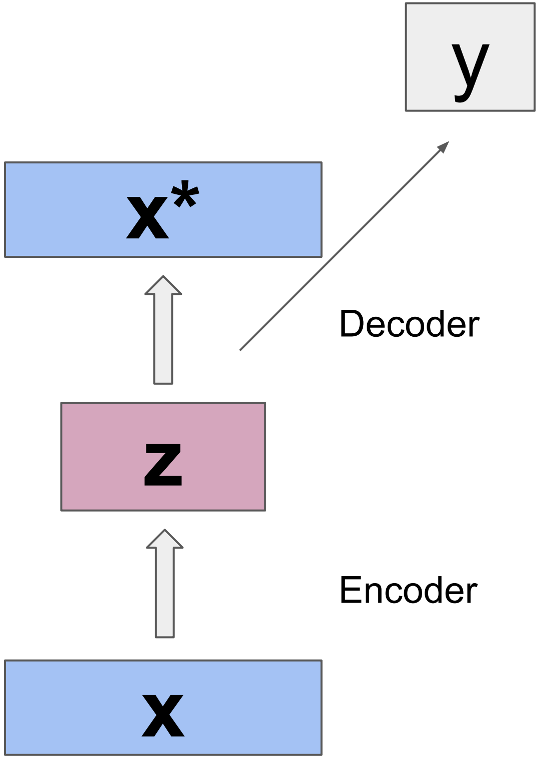
- The encoded feature is sometimes used as features for supervised-learning



Autoencoders - Some challenges

You don’t have control over the features learned:

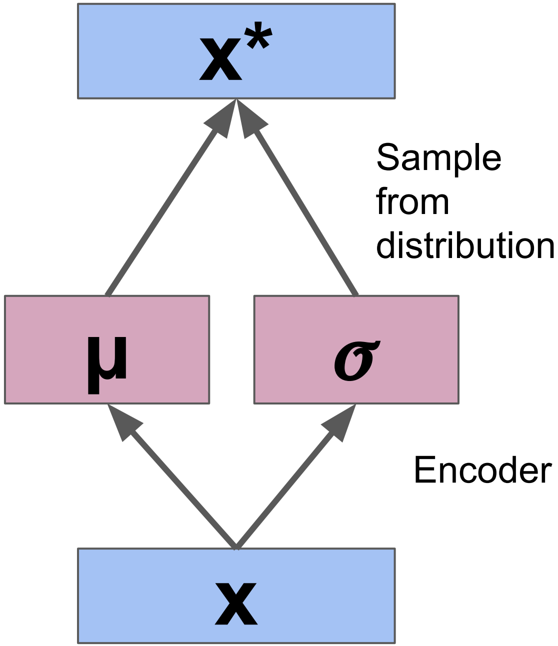
- Even though the features compress the data, they may not be good for categorization.
- For l2-loss it is more important whether an animal is black or white, than which animal it is.
- You can learn the data, but it does not mean that similar data gives similar results (**z**)



Variational Autoencoder

Find the data distribution instead of reconstructing simple images

- Assume some prior distribution
- Use the encoder to estimate distribution parameters
- Sample a **z** from the distribution and try to reconstruct

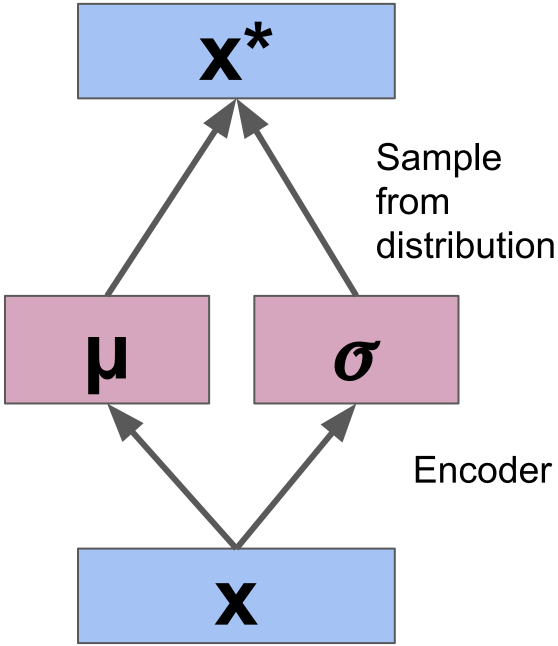


Variational Autoencoder - loss function

Find the data distribution instead of reconstructing simple images

Often

- L2 loss between images
- KL-divergence between estimated distribution and prior distribution
    - Typically unit gaussian



$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$

Variational Autoencoder - loss function

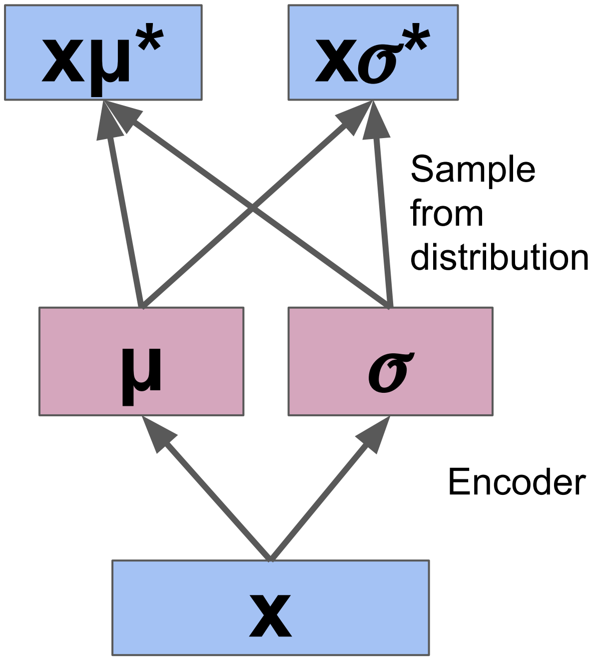
Find the data distribution instead of reconstructing simple images

Often

- L2 loss between images
- KL-divergence between estimated distribution and prior distribution
  - Typically unit gaussian

Alternatively:

- Decode image distribution
- Loss is then the log likelyhood of the inputed image, given the outputted distribution.

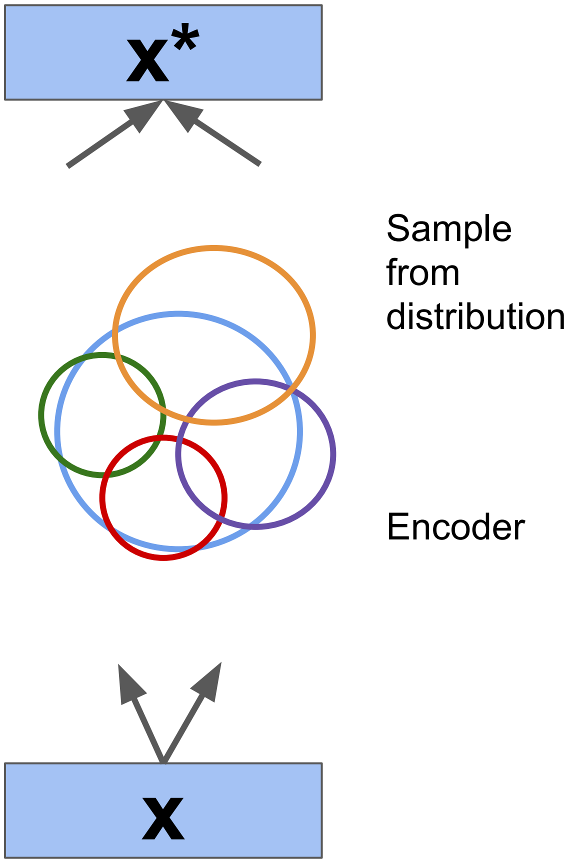


$$\mathbf{E}_z\left[\log p_\theta(x^{(i)} \mid z)\right] - D_{KL}(q_\phi(z \mid x^{(i)}) \,||\, p_\theta(z))$$

Variational Autoencoder - loss function

Find the data distribution instead of reconstructing simple images

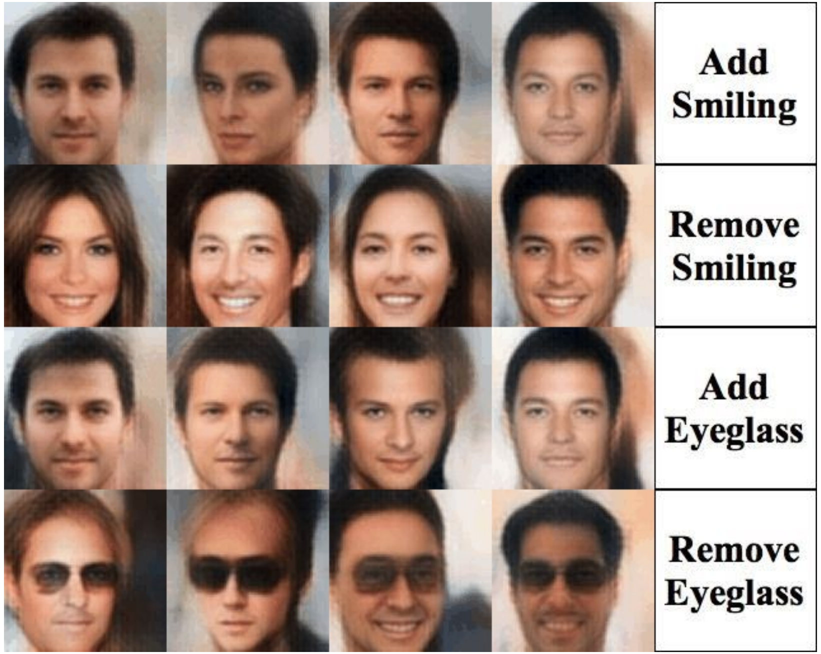
- Force similar data into overlapping distribution
- To really separate some data, you need small variance
    - You pay a cost for lowering variance
    - Have to be weighted by gain in reconstruction
- You train the network to reconstruct “any” input
- Interpolating between samples should give viable results



Variational Autoencoder

Interpolating between samples should give viable results

Similar effects as adversarial autoencoders, where feature directions can have semantic meaning.

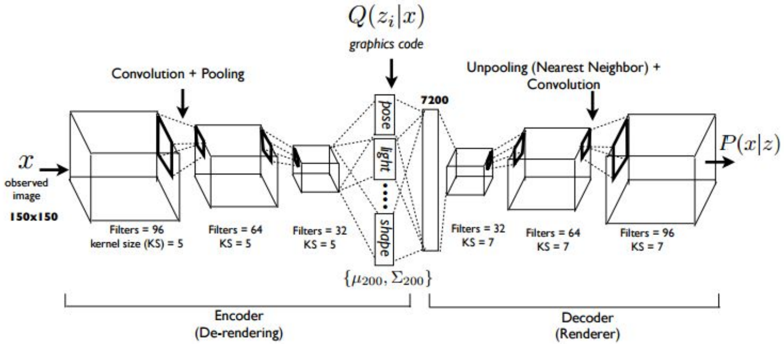


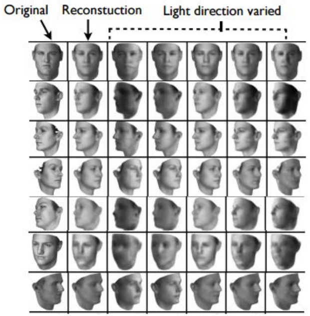
[Deep Feature Consistent Variational Autoencoder]()

Variational Autoencoder - forcing sematics

Interpolating between samples should give viable results

As with GANs we can insert specific information to do semi-supervised learning, and force the embedding to be what we want.



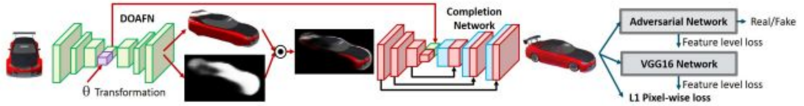


[Deep Convolutional Inverse Graphics Network]

Variational Autoencoder - forcing sematics

Interpolating between samples should give viable results

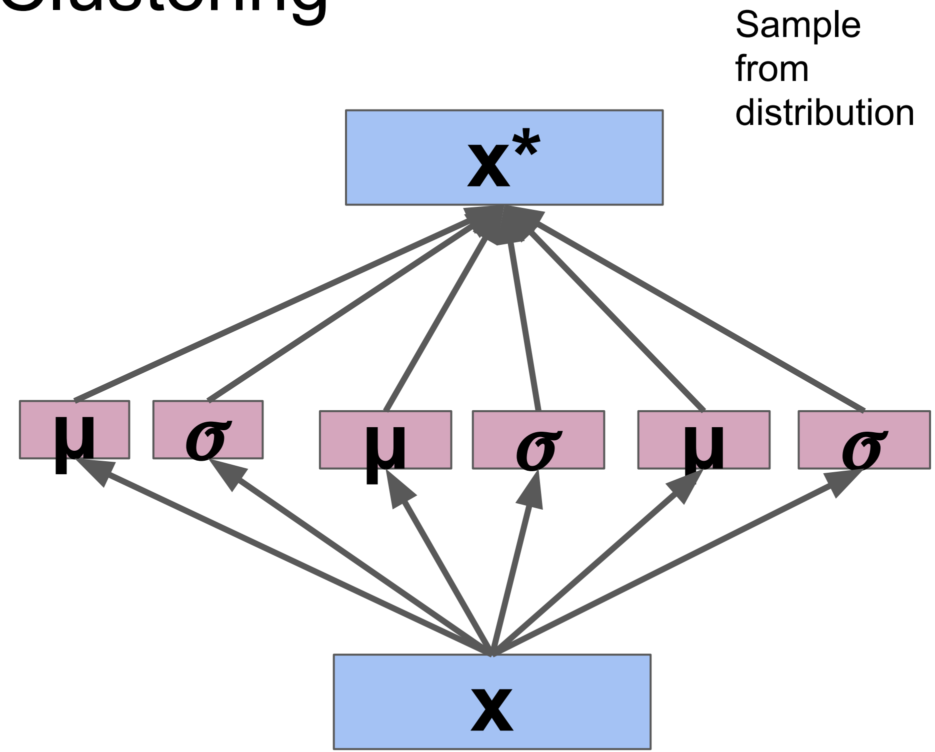
As with GANs we can insert specific information to do semi-supervised learning, and force the embedding to be what we want.



[Transformation-Grounded Image Generation Network for Novel 3D View Synthesis]

Variational Autoencoder - Clustering

- One option is to use k-means clustering on the reduced dimension
- An alternative is to make your prior distribution multimodal
- So your encoder has to put the encoding close to one of the K predefined modes.



[DEEP UNSUPERVISED CLUSTERING WITH GAUSSIAN MIXTURE VARIATIONAL AUTOENCODERS]

Autoencoder for anomaly detection

- Autoencoders can be used for anomaly detection
- Often reconstruction error is a good measure of an anomaly.

Table 4
CCAD-SW performance comparison.

Model	Threshold	TPR (%)	FPR (%)	AUC
CCAD-SW[b]	0.63	52.1	50.4	0.513
CCAD-17[a]	0.07	68.6	12.7	0.842
CCAD-26[a]	0.05	80.2	21.1	0.862
CCAD-SW[a]	0.001	94.5	4.7	0.981

[a] Autoencoder.
[b] PCA.

An ensemble learning framework for anomaly detection in building energy consumption