10417-617
Deep Learning: Fall 2019

Andrej Risteski

Machine Learning Department

Lecture 10:
Intro to unsupervised learning

Unsupervised learning

Learning from data **without** labels.

What can we hope to do:

Task A: Fit a parametrized **structure** (e.g. clustering, low-dimensional subspace, manifold) to data to reveal something meaningful about data. (**Structure learning**)

Task B: Learn a (parametrized) **distribution** *close* to data generating distribution. (**Distribution learning**)

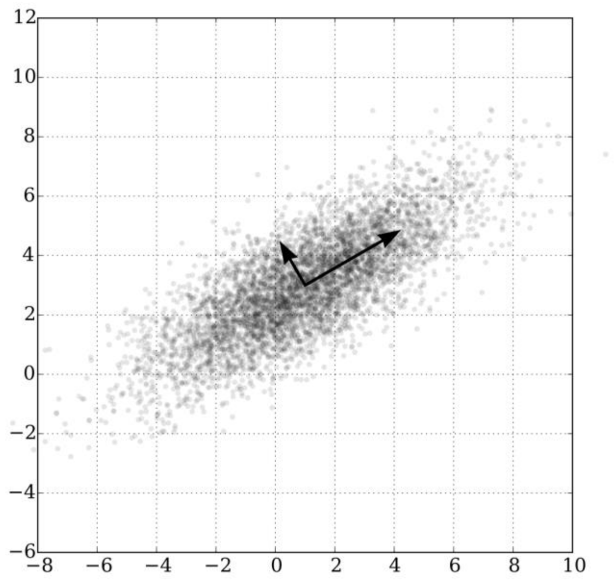
Task C: Learn a (parametrized) distribution that implicitly reveals an **"embedding"/"representation"** of data for downstream tasks. (**Representation/feature learning**)

Entangled! The "structure" and "distribution" often reveals an embedding.

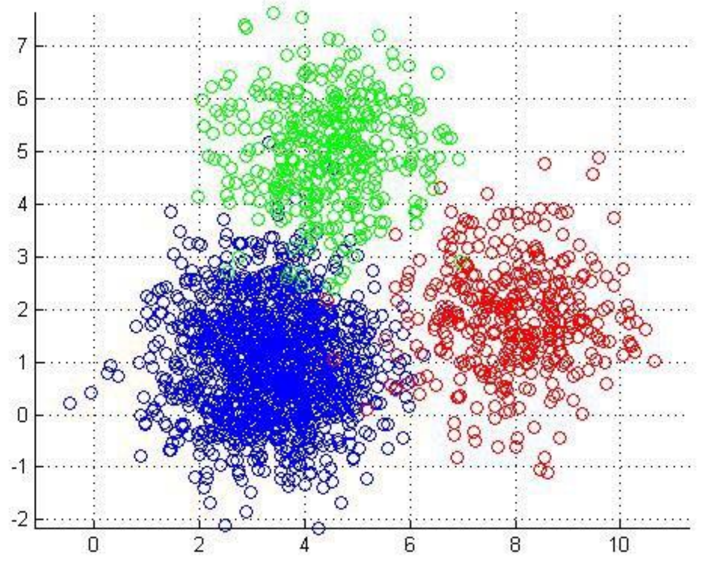
Structure learning

Fit a parametrized **structure** (e.g. clustering, low-dimensional subspace) to data to reveal something meaningful about data.

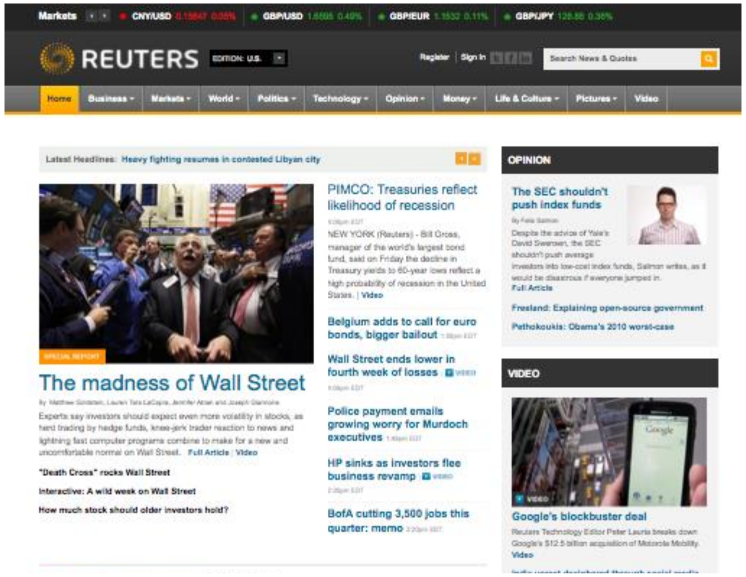
PCA(principal component analysis), direction of highest variance



Clustering



Structure learning



Markets CNY/USD 0.15847 0.05% GBP/USD 1.6505 0.49% GBP/EUR 1.1532 0.11% GBP/JPY 128.88 0.36%

REUTERS EDITION: U.S. Register | Sign In Search News & Quotes

Home Business Markets World Politics Technology Opinion Money Life & Culture Pictures Video

Latest Headlines: Heavy fighting resumes in contested Libyan city

SPECIAL REPORT

The madness of Wall Street

By Matthew Goldstein, Lauren Tara LaCapra, Jennifer Ablan and Joseph Giannone

Experts say investors should expect even more volatility in stocks, as hard trading by hedge funds, knee-jerk trader reaction to news and lightning fast computer programs combine to make for a new and uncomfortable normal on Wall Street. Full Article | Video

"Death Cross" rocks Wall Street

Interactive: A wild week on Wall Street

How much stock should older investors hold?

PIMCO: Treasuries reflect likelihood of recession

NEW YORK (Reuters) - Bill Gross, manager of the world's largest bond fund, said on Friday the decline in Treasury yields to 60-year lows reflect a high probability of recession in the United States. | Video

Belgium adds to call for euro bonds, bigger bailout

Wall Street ends lower in fourth week of losses VIDEO

Police payment emails growing worry for Murdoch executives

HP sinks as investors flee business revamp VIDEO

BofA cutting 3,500 jobs this quarter: memo

OPINION

The SEC shouldn't push index funds

By Felix Salmon

Despite the advice of Yale's David Swensen, the SEC shouldn't push average investors into low-cost index funds, Salmon writes, as it would be disastrous if everyone jumped in.
Full Article

Freeland: Explaining open-source government

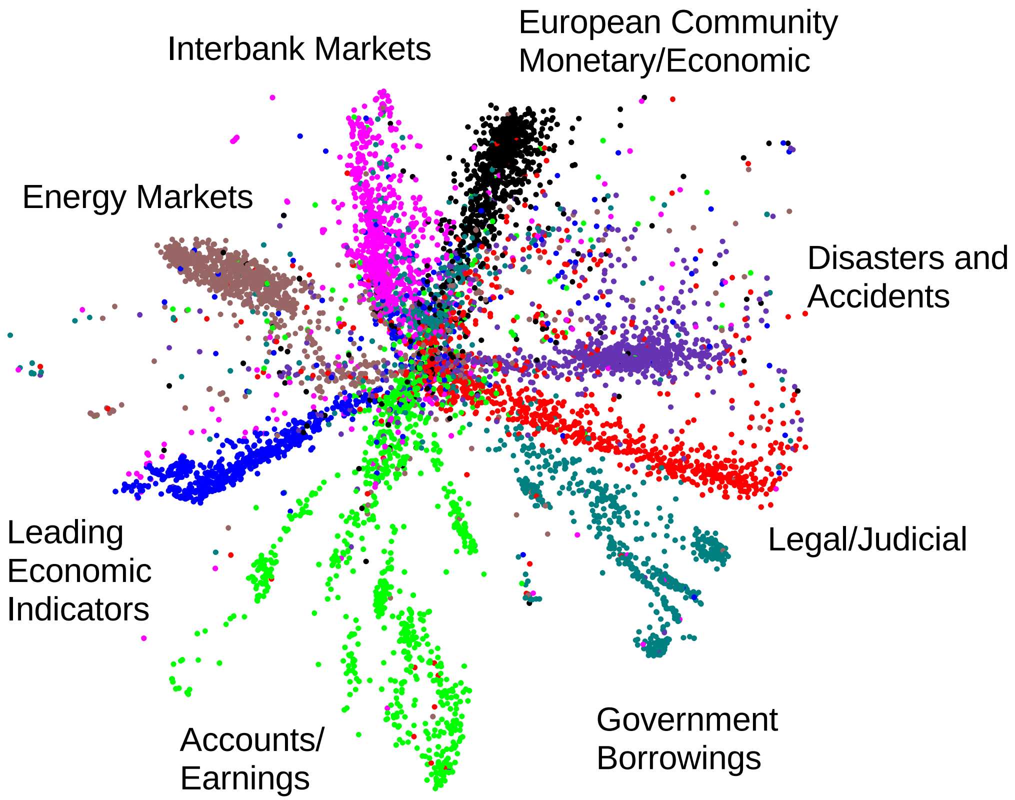
Pethokoukis: Obama's 2010 worst-case

VIDEO

Google's blockbuster deal

Reuters Technology Editor Peter Lauria breaks down Google's $12.5 billion acquisition of Motorola Mobility.
Video

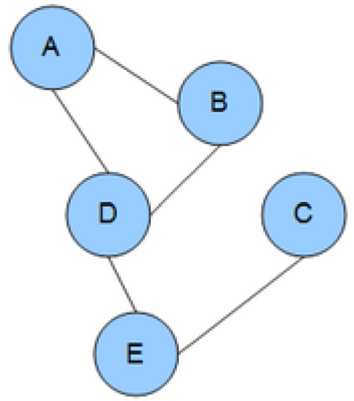
804,414 newswire stories



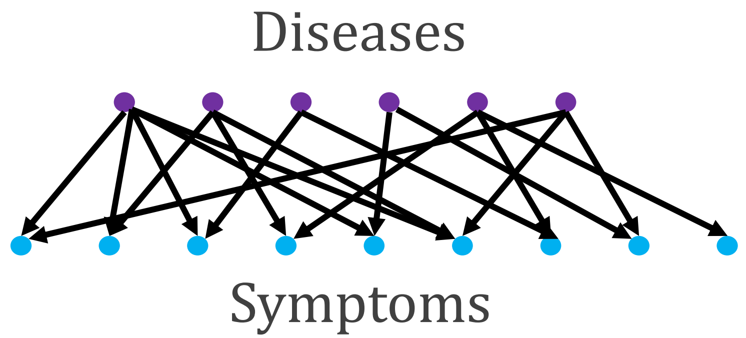
Distribution learning

Some typical choices of parametrized distributions:

Classical choices: fully-observed graphical models (undirected and directed), latent-variable graphical models (mixture models, sparse coding, topic models).



Markov Random Fields: sparse independence structure: "A is independent of other vars, given B, D"

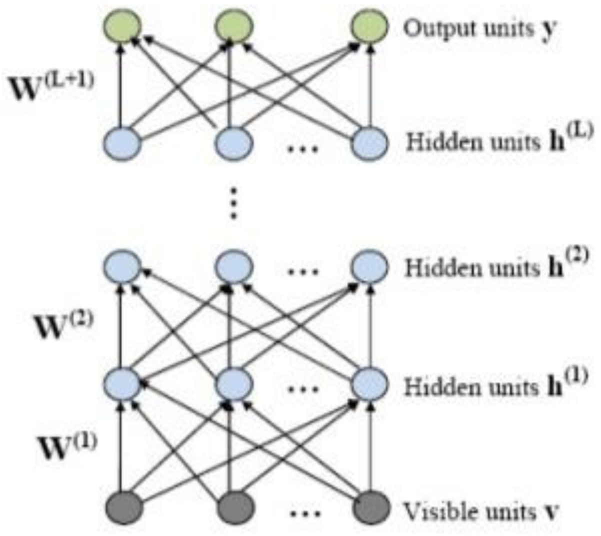


Latent variable models: data is "simple" conditioned on some unobserved (latent) variables

Distribution learning

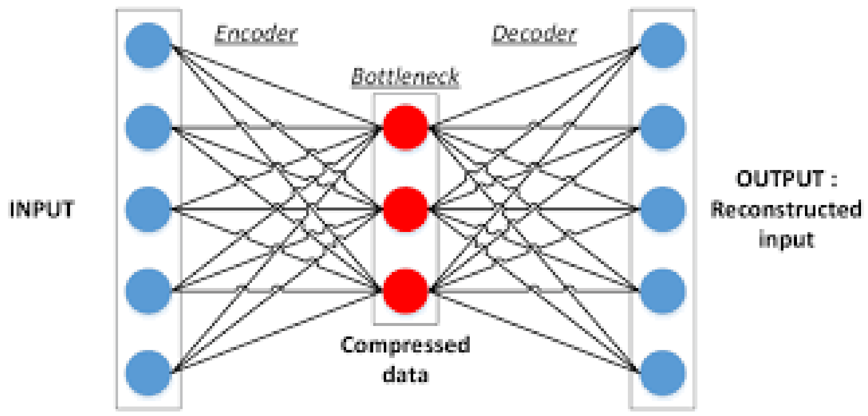
Some typical choices of parametrized distributions:

Semi-modern choices: deep Boltzmann machines, deep belief networks, (variational) auto-encoders, energy models.



Output units $\mathbf{y}$
$\mathbf{W}^{(L+1)}$
Hidden units $\mathbf{h}^{(L)}$
Hidden units $\mathbf{h}^{(2)}$
$\mathbf{W}^{(2)}$
Hidden units $\mathbf{h}^{(1)}$
$\mathbf{W}^{(1)}$
Visible units $\mathbf{v}$

Deep Boltzmann machines, belief networks: graphical model analogues of deep neural networks.

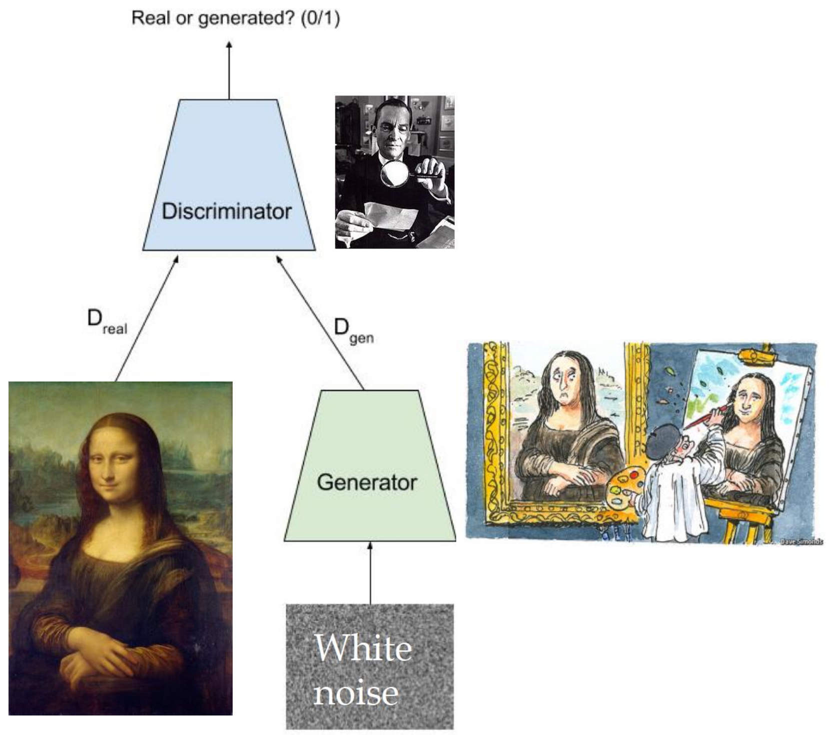


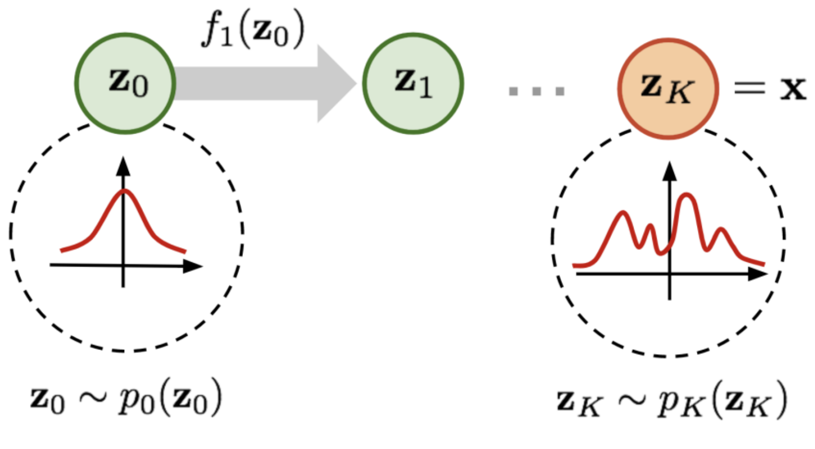
(Variational) autoencoders: model data by enforcing a latent space "bottleneck:

Distribution learning

Some typical choices of parametrized distributions:

Modern choices: generative adversarial networks, autoregressive models (pixelRNN, pixelCNN), flow models, etc.




$f_1(\mathbf{z}_0)$
$\mathbf{z}_0$
$\mathbf{z}_1$
$\cdots$
$\mathbf{z}_K = \mathbf{x}$
$\mathbf{z}_0 \sim p_0(\mathbf{z}_0)$
$\mathbf{z}_K \sim p_K(\mathbf{z}_K)$

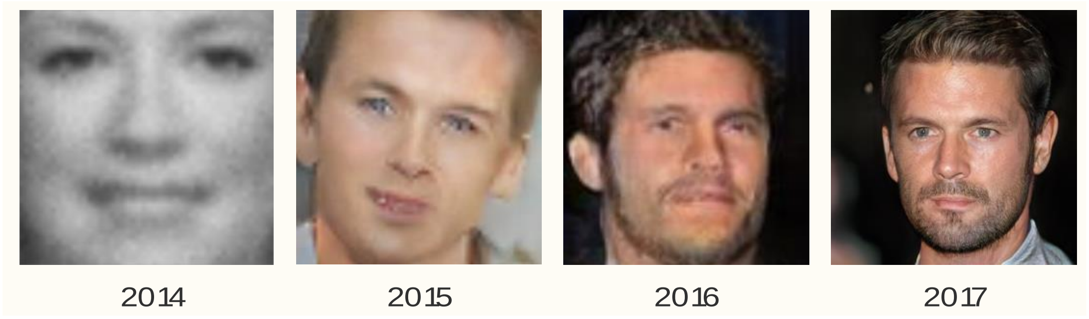
Distribution learning

$\longrightarrow$

Training Data(CelebA)

Model Samples (Karras et.al., 2018)

4 years of progression on Faces



Brundage et al., 2017

Distribution learning

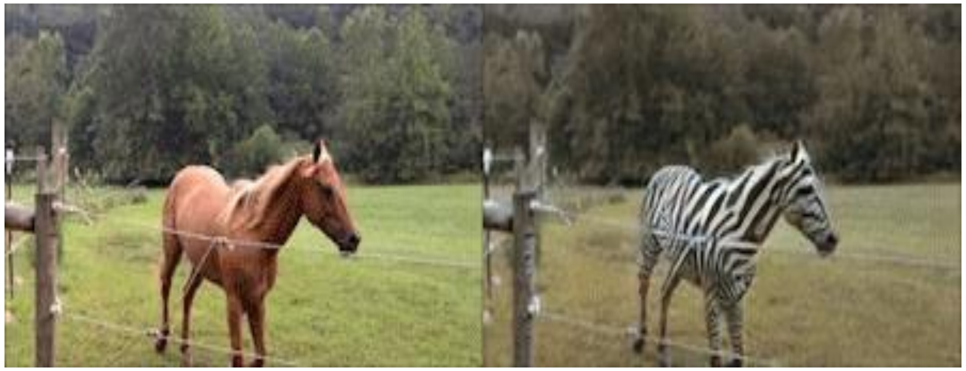
Whichfaceisreal.com

Distribution learning

BigGAN, Brock et al ‘18

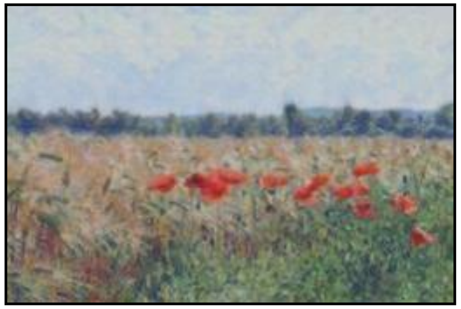
Distribution learning

Conditional generative model P(zebra images| horse images)

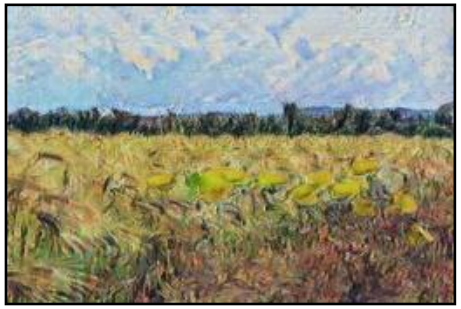


Style Transfer

Input Image



Monet

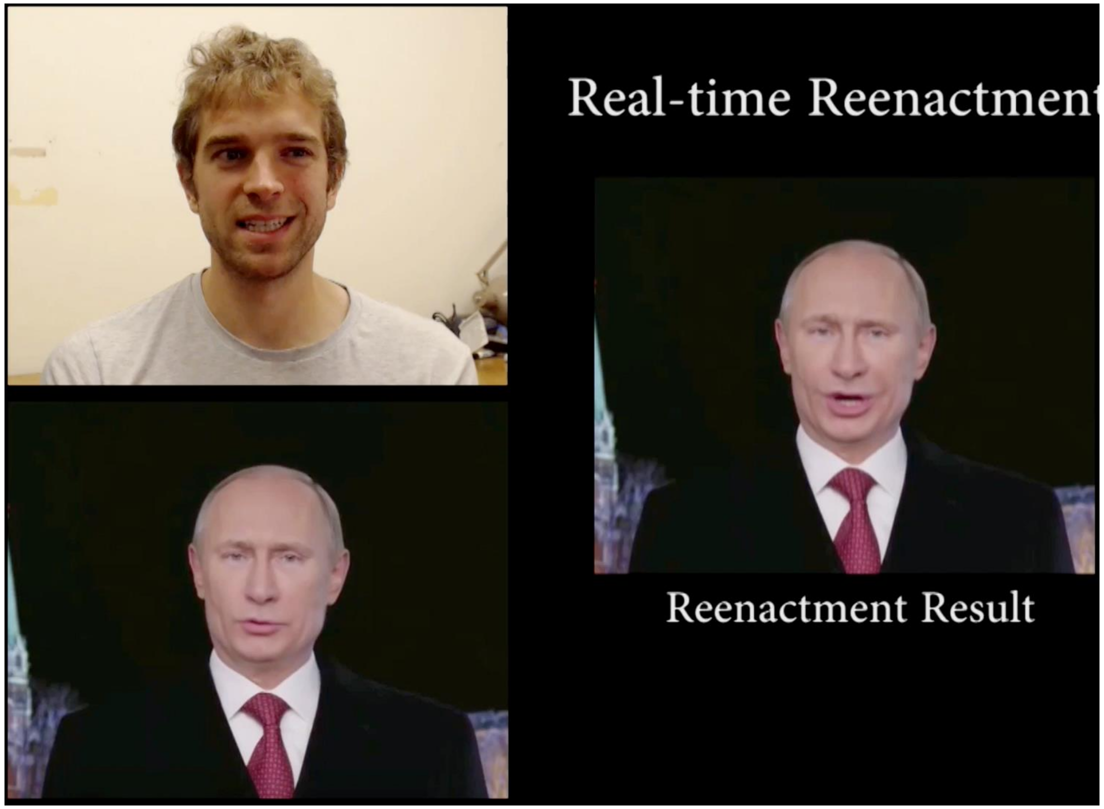


Van Gogh

Zhou el al., Cycle GAN 2017

Distribution learning

Source actor



Real-time reenactment

Target actor

[Face2Face, Thies et al, CVPR 2016]

Representation learning and self-supervised learning

Given unlabeled data, design supervised tasks that induce a good representation for downstream tasks.

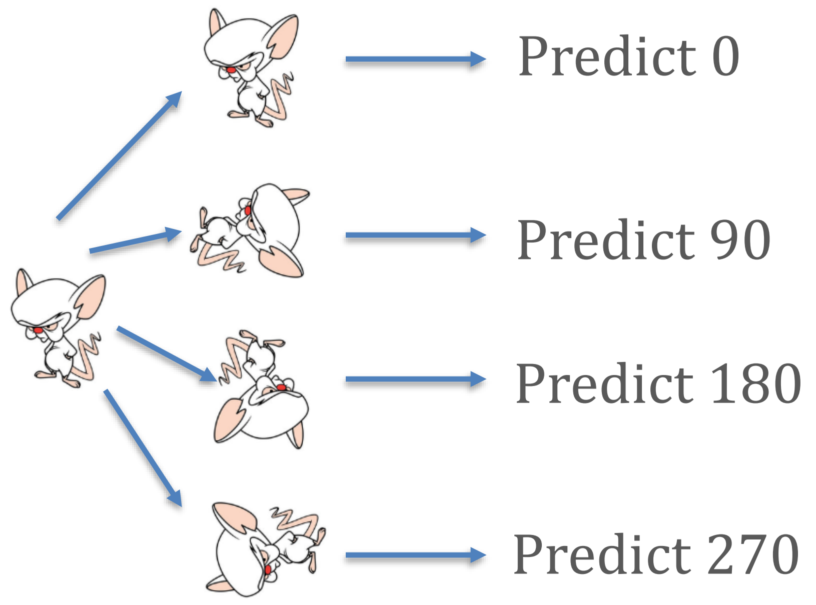
No good mathematical formalization, but the intuition is to “force” the predictor used in the task to learn something “semantically meaningful” about the data.

Examples in NLP: predict next word, given previous 5 words; predict middle word, given surrounding 5 words; etc.

Representation learning and self-supervised learning

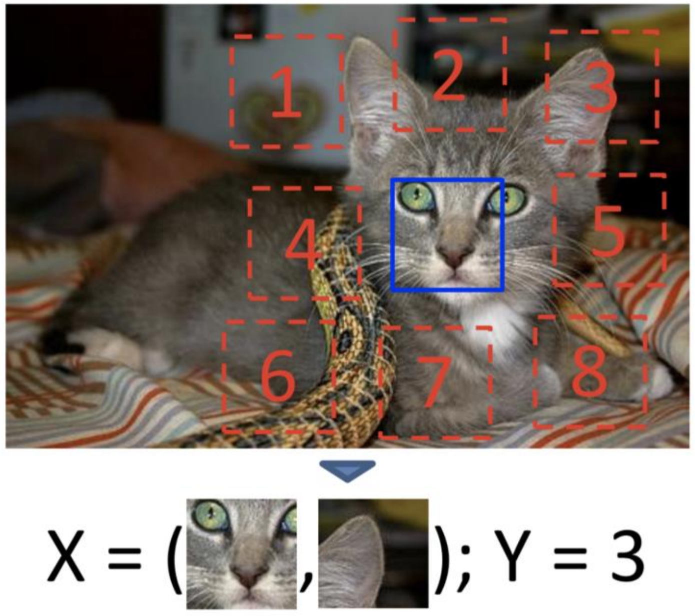
Examples in vision: a lot, and quite different in nature.

Rotation prediction



Predict one of four angles an image is rotated by

Jigsaw puzzles



Predict position of second piece wrt to first

Relationships between the tasks

Structure learning and feature learning often blend
E.g. low dimensional features in PCA or cluster a point belongs to in clustering can be viewed as features.

Structure learning/feature learning is in general weaker than distribution learning:
E.g. methods like PCA/clustering can't be used to generate new samples.

Feature, not a bug: it has been argued that self-supervised learning works because the task we are solving is easier (both computationally and statistically)

Relationships between the tasks

Distribution learning often implies representation learning:
Many distributions we fit are **latent-variable** models (i.e. model the joint distribution between some latent variables **h** and the observed data **x**)

$$P_\theta(x, h) = P_\theta(h) P_\theta(x|h)$$

The latent variables are often viewed as a "***representation***".

The **posterior** distribution $P_\theta(h|x)$ captures a distribution over representations, given some values of the observed data.

However, a-priori, distribution $P_\theta(h|x)$ is a-priori not an easy distribution to **approximate/sample** from!

$$P_\theta(h|x) = \frac{P_\theta(x|h)p(h)}{\int_{h'} p(h')p(x|h')}$$

Hard high-dimensional sum/integral

Training techniques: Likelihood-based vs likelihood-free

Two typical families of training algorithms:

Likelihood-based: maximize the likelihood of the data under the model (possibly with some approximations)

$$\max_\theta \sum_{\text{samples } x_i} \log p_\theta(x_i)$$

In the limit of infinite number of samples:

$$\begin{aligned}\max_\theta \mathbb{E}_{x\sim p} \log p_\theta(x) &= -(\mathbb{E}_{x\sim p} \log \frac{1}{p_\theta(x)} + \mathbb{E}_{x\sim p} \log p(x) - \mathbb{E}_{x\sim p} \log p(x)) \\ &= -(\qquad KL(p||p_\theta) \qquad + \quad H(p))\end{aligned}$$

Hence, we are minimizing KL divergence ($H(p)$ is a constant).

Training techniques: Likelihood-based vs likelihood-free

Two typical families of training algorithms:

Likelihood-based: maximize the likelihood of the data under the model (possibly with some approximations)

Pros

Easy training: can just maximize via gradient descent.

Evaluation: evaluating the fit of the model can be done by evaluating the likelihood (on test data)

Cons

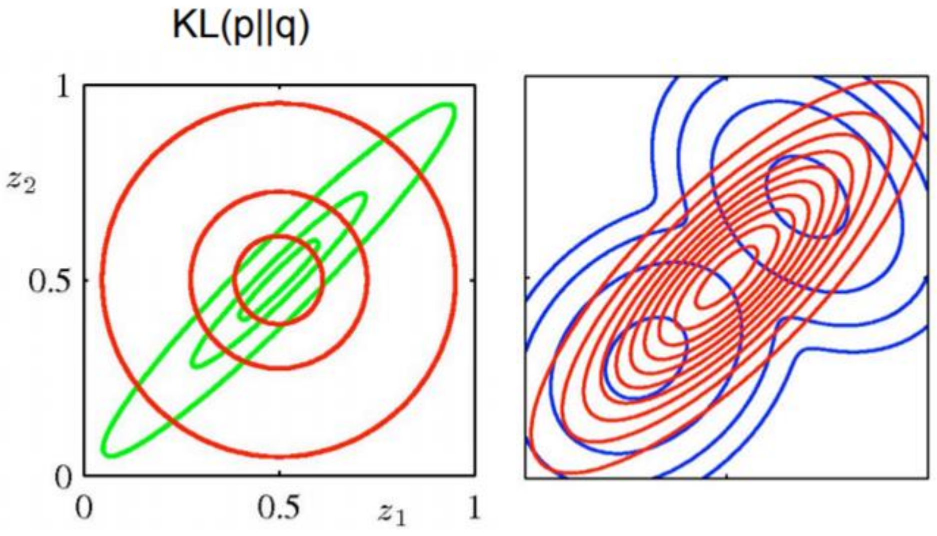
Larger models needed: likelihood objective is hard, to fit well need very big model

Likelihood encourages averaging: produced samples tend to be blurrier, as likelihood encourages "coverage" of training data.

Training techniques: Likelihood-based vs likelihood-free

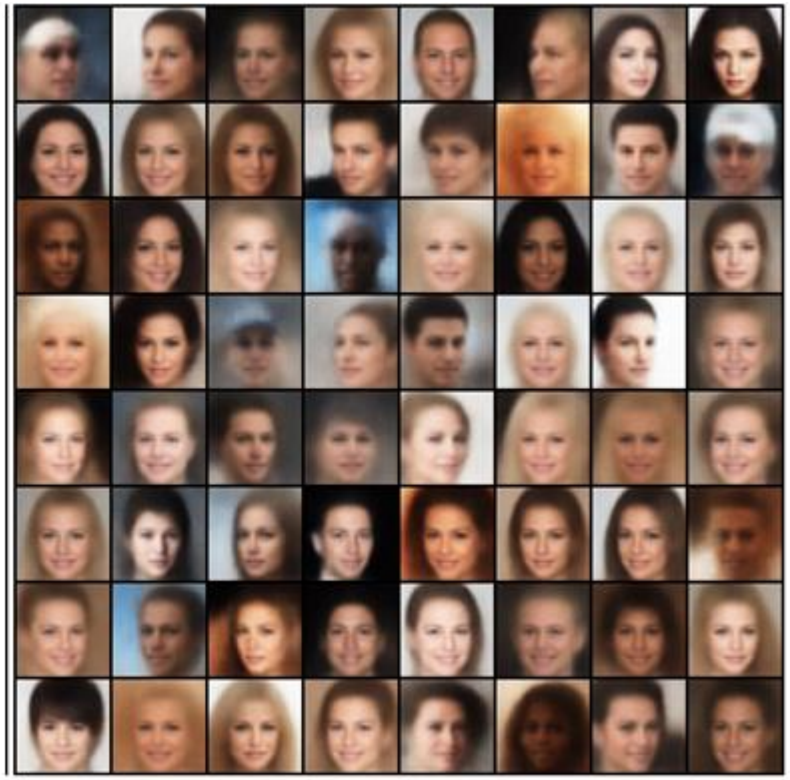
$$\mathrm{KL}(p\|q) = -\int p(\mathbf{Z}) \ln \frac{q(\mathbf{Z})}{p(\mathbf{Z})} \mathrm{d}\mathbf{Z}.$$

There is a large positive contribution to the KL divergence from regions of Z space in which:
- q(Z) is near zero,
- unless p(Z) is also close to zero.



Minimizing KL(p||q) leads to distributions q(Z) that are nonzero in regions where p(Z) is nonzero.

Training techniques: Likelihood-based vs likelihood-free



Faces generated using a trained VAE

Training techniques: Likelihood-based vs likelihood-free

Two typical families of training algorithms:

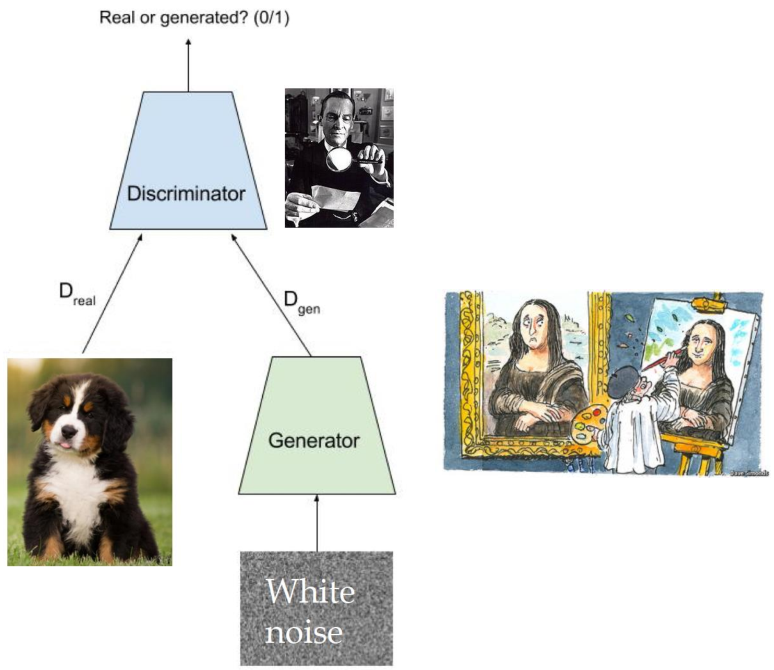
Likelihood-based: maximize the likelihood of the data under the model (possibly with some approximations)

Typical approximations used: variational inference (optimize tractable deterministic approximation of posteriors), MCMC methods (idea: approximate difficult quantities like posteriors with sampling)

Likelihood-free: use a surrogate loss – e.g. in GANs, train a discriminator to tell real and generated samples apart; noise-contrastive training: encourage model to put probability mass away from “fake” samples.

Training techniques: Likelihood-based vs likelihood-free

Likelihood-free: use a surrogate loss – e.g. in GANs, train a discriminator to tell real and generated samples apart; noise-contrastive training: encourage model to put probability mass away from “fake” samples.



Training techniques: Likelihood-based vs likelihood-free

Likelihood-free: use a surrogate loss – e.g. in GANs, train a discriminator to tell real and generated samples apart; noise-contrastive training: encourage model to put probability mass away from "fake" samples.

Pros

Better objective, smaller models needed: objective itself (i.e. discriminator) is "learned" – can result in visually better images w/ smaller models.

Cons

Unstable training: typically min-max (saddle point) problems.

Evaluation: no way to evaluate likelihood, so no way to evaluate quality of fit.

The classics: PCA

Figure 1: Population structure within Europe.

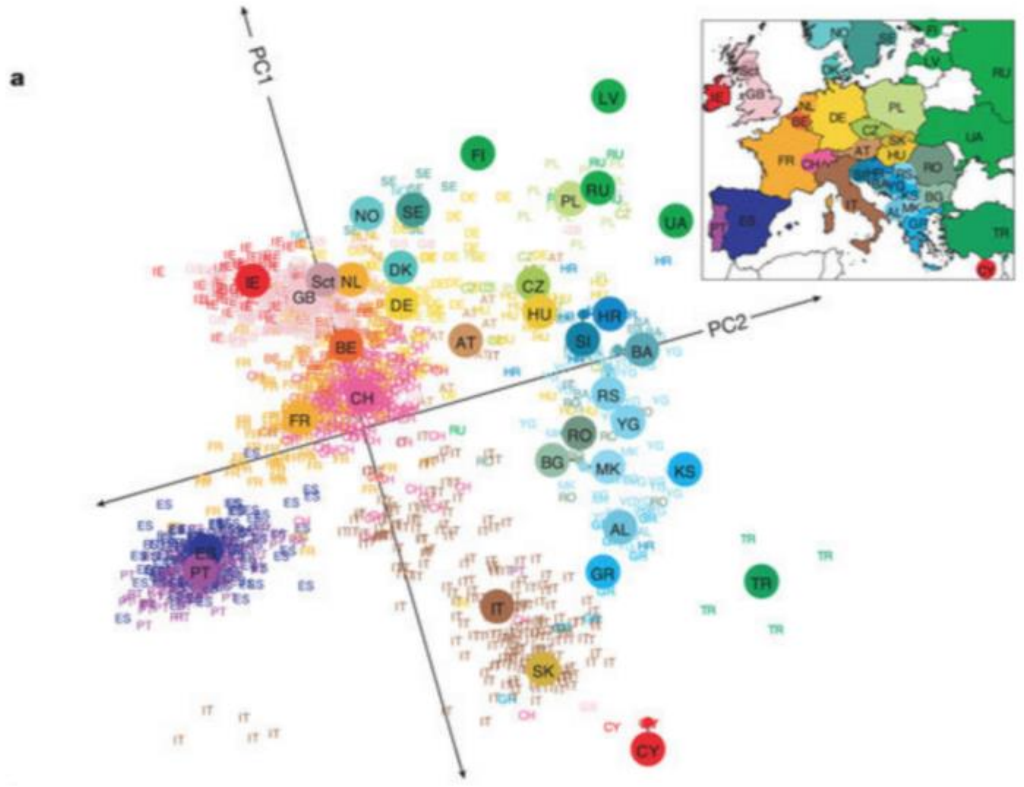


Figure 6: Plot from [1], depicting genomes for 1387 Europeans projected onto top 2 principal components. Colors/labels of datapoints correspond to geographic location of the individuals. Map of Europe (with same coloring) included in upper right for reference.

Novembre et al '08

The classics: PCA

Goal: find a k-dimensional (linear) subspace explaining most of the variance in the data.

Assume the data is centered, that is $\mathbb{E}[x] = 0$

Warmup: let k=1.

$$\underset{\{v:\ ||v||=1\}}{\text{argmax}} \frac{1}{m} \sum_{\text{samples } x_i} \langle v, x_i \rangle^2$$

As $m \to \infty$, we get $\mathbb{E}[\langle v, x \rangle^2]$,
i.e. variance.

The classics: PCA

Goal: find a k-dimensional (linear) subspace explaining most of the variance in the data.

$$\underset{\{v_1,v_2,\ldots,v_k\}}{\operatorname{argmax}} \frac{1}{m} \sum_{\text{samples } x_i} \left(\text{length of } \mathrm{x_i} \text{ on span}(v_1, v_2, \ldots, v_k)\right)^2$$

$$= \underset{\text{orthonormal } \{v_1,v_2,\ldots,v_k\}}{\operatorname{argmax}} \frac{1}{m} \sum_{\text{samples } x_i} \sum_{j=1}^{k} \langle x_i, v_j \rangle^2$$

As $m \to \infty$, we get $\mathbb{E}[\sum_j \langle v_j, x \rangle^2]$

The classics: PCA

How to do this efficiently?

$$\underset{\text{orthonormal}\ \{v_1,v_2,\dots,v_k\}}{\text{argmax}} \frac{1}{m} \sum_{\text{samples}\ x_i} \sum_{j=1}^{k} \langle x_i, v_j \rangle^2$$

A convenient rewrite:

$$= \underset{\text{orthonormal}\ \{v_1,v_2,\dots,v_k\}}{\text{argmax}} \frac{1}{m} \sum_{\text{samples}\ x_i} \sum_{j=1}^{k} (v_j^T x_i)(x_i^T v_j)$$

$$= \underset{\text{orthonormal}\ \{v_1,v_2,\dots,v_k\}}{\text{argmax}} \frac{1}{m} \sum_{\text{samples}\ x_i} \sum_{j=1}^{k} v_j^T (x_i x_i^T) v_j$$

$$= \underset{\text{orthonormal}\ \{v_1,v_2,\dots,v_k\}}{\text{argmax}} \sum_{j=1}^{k} v_j^T \underbrace{\left(\frac{1}{m} \sum_{\text{samples}\ x_i} (x_i x_i^T)\right)}_{:=\ \text{D, (covariance matrix)}} v_j$$

The classics: PCA

*How to do this efficiently? – **Singular Value Decomposition**!!*

$$\underset{\text{orthonormal } \{v_1, v_2, \dots, v_k\}}{\operatorname{argmax}} \frac{1}{m} \sum_{\text{samples } x_i} \sum_{j=1}^{k} \langle x_i, v_j \rangle^2$$

A convenient rewrite: $$\underset{\text{orthonormal } \{v_1, v_2, \dots, v_k\}}{\operatorname{argmax}} \sum_{j=1}^{k} v_j^T \mathrm{D} v_j$$

If D is **diagonal** (w/ positive entries), and we sort the entries s.t. $D_{11} \geq D_{22} \dots \geq D_{dd}$, it's easy to see max is $\sum_{j=1}^{k} D_{jj}$. Namely:

$$\sum_{j=1}^{k} v_j^T \mathrm{D} v_j = \sum_j \sum_i \left(v_j\right)_i^2 D_{ii} \leq \sum_{j=1}^{k} D_{jj}, \text{ as } \left(v_j\right)_i^2 = 1$$

The corresponding argmax is $e_1, e_2, \dots, e_k$.

The classics: PCA

*How to do this efficiently? – **Singular Value Decomposition**!!*

$$\underset{\text{orthonormal }\{v_1,v_2,\dots,v_k\}}{\operatorname{argmax}} \frac{1}{m} \sum_{\text{samples } x_i} \sum_{j=1}^{k} \langle x_i, v_j \rangle^2$$

A convenient rewrite: $$\underset{\text{orthonormal }\{v_1,v_2,\dots,v_k\}}{\operatorname{argmax}} \sum_{j=1}^{k} v_j^T \mathrm{D} v_j$$

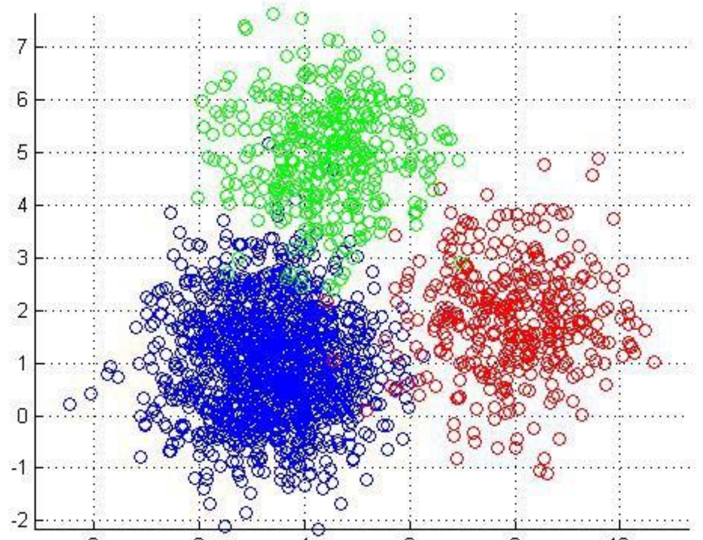
If D is not diagonal, write $\mathrm{D} = \mathrm{U}\, \widetilde{D} U^T$ to reduce to diagonal case:

$$\underset{\text{orthonormal }\{v_1,v_2,\dots,v_k\}}{\operatorname{argmax}} \sum_{j=1}^{k} v_j^T (U\, \widetilde{D} U^T) v_j = \underset{\text{orthonormal }\{v_1,v_2,\dots,v_k\}}{\operatorname{argmax}} \sum_{j=1}^{k} \left(U^T v_j\right)^T \widetilde{D} (U^T v_j)$$

Since U^T is orthogonal, $\{U^T v_1, U^T v_2, \dots, U^T v_k\}$ also are orthonormal!

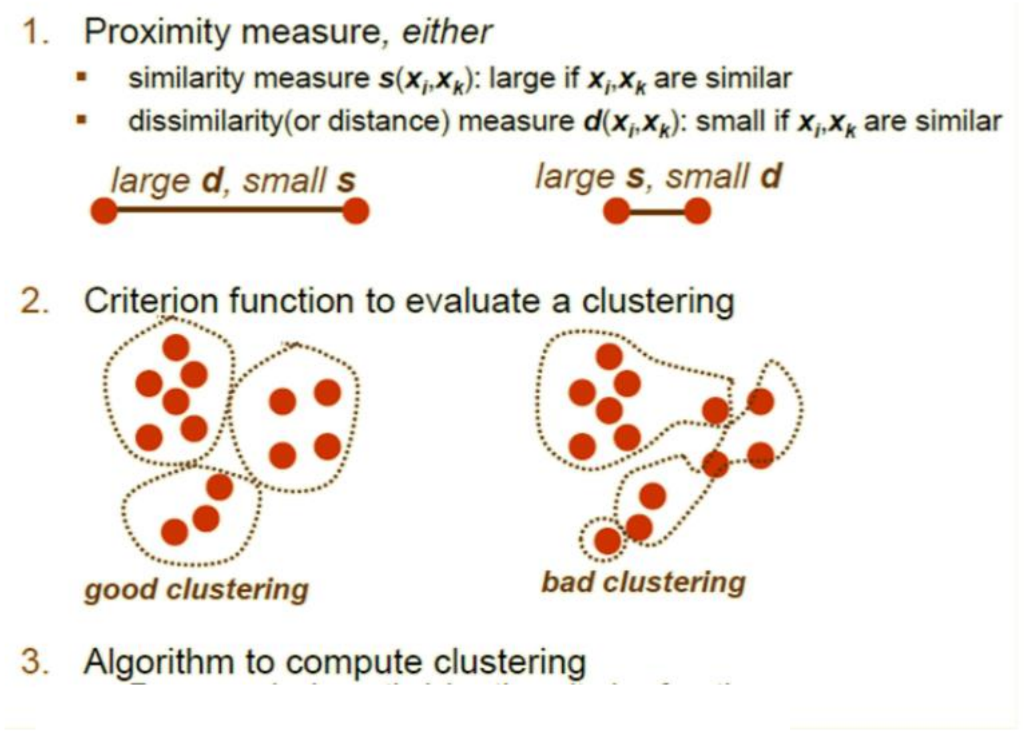
So, the max is $\sum_{j=1}^{k} \widetilde{D}_{jj} = \sum_{j=1}^{k} \lambda_j(D)$ and the argmax are the **top k eigenvectors** of D

The classics: clustering



Goal: group the data into clusters of nearby points.

What's needed for clustering?



1. Proximity measure, *either*
   - similarity measure $s(\mathbf{x}_i,\mathbf{x}_k)$: large if $\mathbf{x}_i,\mathbf{x}_k$ are similar
   - dissimilarity(or distance) measure $d(\mathbf{x}_i,\mathbf{x}_k)$: small if $\mathbf{x}_i,\mathbf{x}_k$ are similar

   large ***d****, small* ***s*** *large* ***s****, small* ***d***

2. Criterion function to evaluate a clustering

   good clustering ***bad clustering***

3. Algorithm to compute clustering

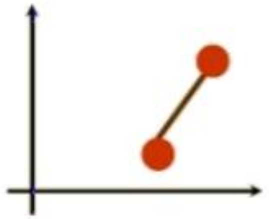
The classics: clustering

Popular distance metrics:

- Euclidean distance

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^{d} \left(x_i^{(k)} - x_j^{(k)}\right)^2}$$

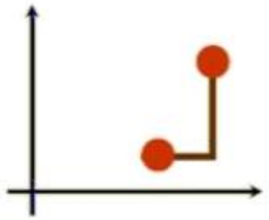
  - translation invariant



- Manhattan (city block) distance

$$d(x_i, x_j) = \sum_{k=1}^{d} \left| x_i^{(k)} - x_j^{(k)} \right|$$

  - approximation to Euclidean distance, cheaper to compute



- They are special cases of **Minkowski distance**:

$$d_p(\mathbf{x}_i, \mathbf{x}_j) = \left(\sum_{k=1}^{m} \left| x_{ik} - x_{jk} \right|^p \right)^{\frac{1}{p}}$$

(p is a positive integer)

The classics: clustering

Criterion functions:

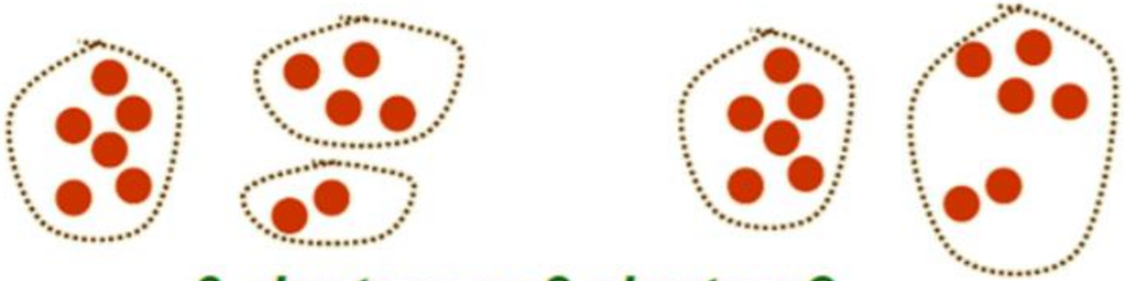
Intra-cluster cohesion
– Cohesion measures how near the data points in a cluster are to the cluster “center”.

Inter-cluster separation
– Separation means that different cluster centroids should be far away from one another.

In most applications, expert judgments are still the key

The classics: clustering

How many clusters?



3 clusters or 2 clusters?

- Possible approaches
  1. fix the number of clusters to ***k***
  2. find the best clustering according to the criterion function (number of clusters may vary)

K-means clustering

If the distance metric is the Euclidean distance, and the measure of cohesion is the average distance from the centroid: we get the **k-means objective.**

$$argmin_{\{r_{nk},\boldsymbol{\mu}_\mathbf{k}\}} \sum_{n=1}^{N} \sum_{k=1}^{K} \underbrace{r_{nk}}_{\textit{Is point n in cluster k?}} \|\mathbf{x_n} - \underbrace{\boldsymbol{\mu}_\mathbf{k}}_{\textit{Centroid of k-th cluster}}\|^2$$

K-means clustering

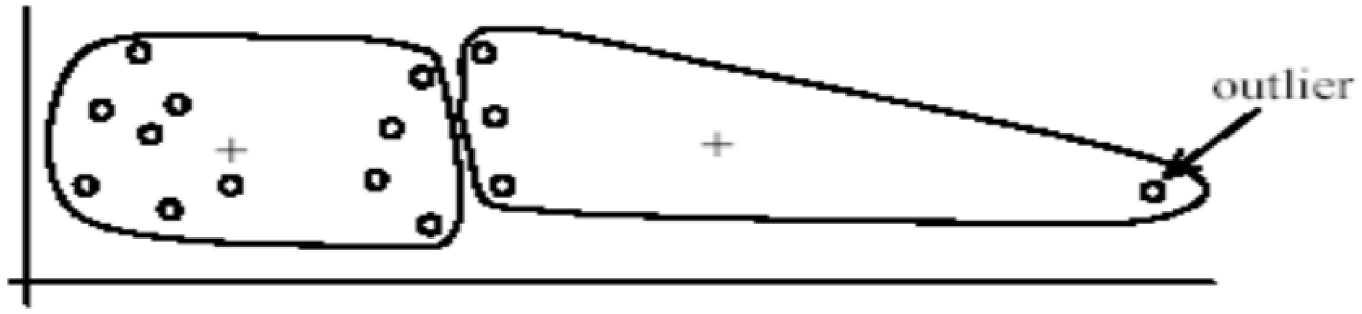
A natural iterative algorithm, which as we will see later is a variant of the **EM (expectation-maximization)** algorithm:

```
Input: Data set $X = \{x^{(1)}, x^{(2)}, \cdots, x^{(m)} | x^{(i)} \in \mathbb{R}^n\}$
Output: Cluster centroids $\mu_{i=1,\cdots,k} \in \mathbb{R}^n$;Cluster assignments $c \in \mathbb{Z}$
Initialize $k$ cluster centroids $\mu_1, \cdots, \mu_k \in \mathbb{R}^n$ randomly from $X$;
repeat
    for $i = 1, \cdots, m$ do // Update cluster assignments
        set $c^{(i)} = \underset{j}{arg\ min} ||x^{(i)} - \mu_j||^2$;
    end
    for $j = 1, \cdots, k$ do // Update cluster centroids
        set $\mu_j = \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$;
    end
until $Convergence$;
return $\mu$ and $c$;
```

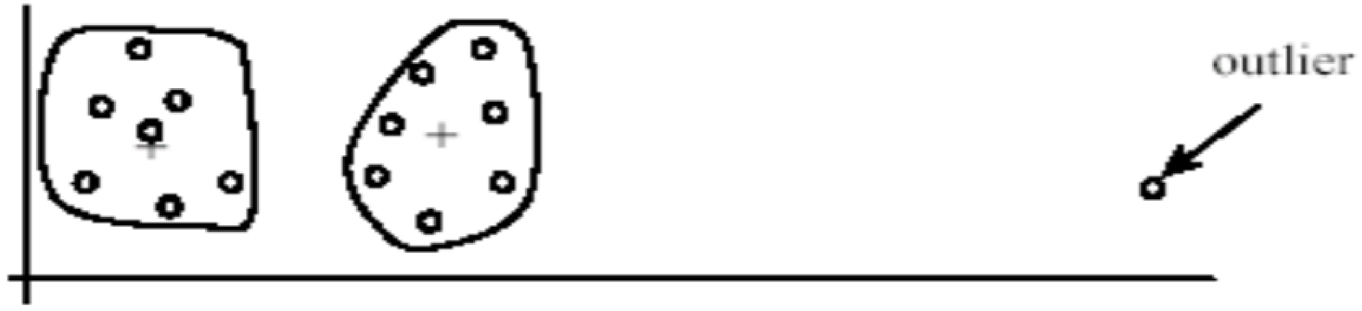
Algorithm 1: Algorithm of batch-version for K-means

Some weaknesses

Very sensitive to outliers:



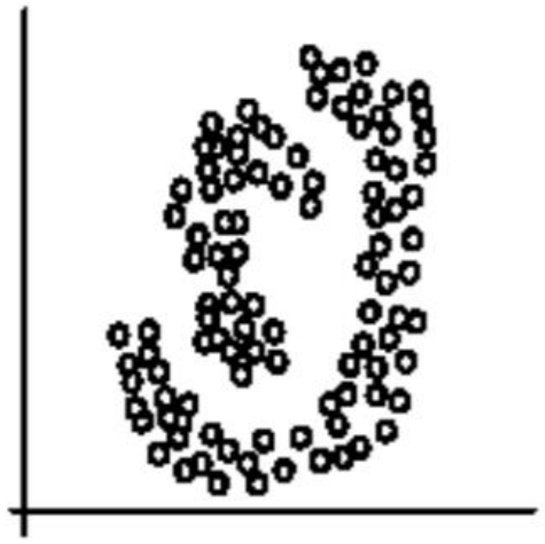
(A): Undesirable clusters

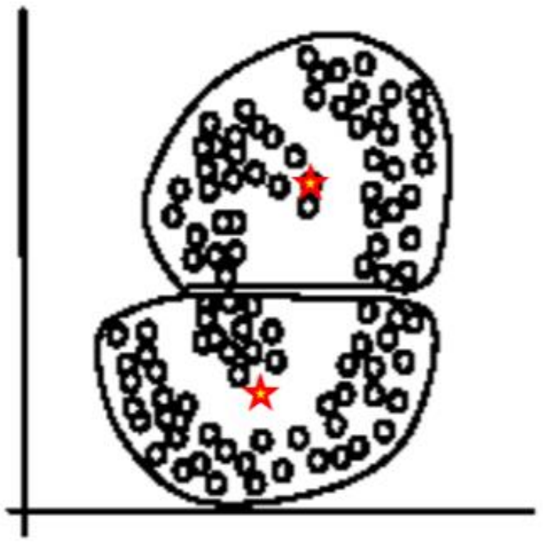


(B): Ideal clusters

Some weaknesses

Not suitable for non-spherical clusters:


(A): Two natural clusters


(B): *k*-means clusters