KSBi-BIML 2021

Bioinformatics & Machine Learning (BIML)
Workshop for Life Scientists

생물정보학 & 머쉰러닝 워크샵(온라인)

Dimensionality Reduction

이제근

SBi 한국생명정보학회 Korean Society for Bioinformatics
서울대학교 생물정보연구소 SNU BIOINFORMATICS INSTITUTE
Gene·2·us

Bioinformatics & Machine Learning for Life Scientists

BIML-2021

안녕하십니까?

한국생명정보학회의 동계 워크샵인 BIML-2021을 2월 15부터 2월 19일까지 개최합니다. 생명정보학 분야의 융합이론 보급과 실무역량 강화를 위해 도입한 전문 교육 프로그램인 BIML 워크샵은 2015년에 시작하였으며 올해로 7차를 맞이하게 되었습니다. 유례가 없는 코로나 대유행으로 인해 올해의 BIML 워크숍은 온라인으로 준비했습니다. 생생한 현장 강의에서만 느낄 수 있는 강의자와 수강생 사이의 상호교감을 가질 수 없다는 단점이 있지만, 온라인 강의의 여러 장점을 살려서 최근 생명정보학에서 주목받고 있는 거의 모든 분야를 망라한 강의를 준비했습니다. 또한 온라인 강의의 한계를 극복하기 위해서 실시간 Q&A 세션 또한 마련했습니다.

BIML 워크샵은 전통적으로 크게 생명정보학과 AI, 두 개의 분야로 구성되어오고 있으며 올해 역시 유사한 방식을 채택했습니다. AI 분야는 Probabilistic Modeling, Dimensionality Reduction, SVM 등과 같은 전통적인 Machine Learning부터 Deep Learning을 이용한 신약개발 및 유전체 연구까지 다양한 내용을 다루고 있습니다. 생명정보학 분야로는, Proteomics, Chemoinformatics, Single Cell Genomics, Cancer Genomics, Network Biology, 3D Epigenomics, RNA Biology, Microbiome 등 거의 모든 분야가 포함되어 있습니다. 연사들은 각 분야 최고의 전문가들이라 자부합니다.

이번 BIML-2021을 준비하기까지 너무나 많은 수고를 해주신 BIML-2021 운영위원회의 김태민 교수님, 류성호 교수님, 남진우 교수님, 백대현 교수님께 커다란 감사를 드립니다. 또한 재정적 도움을 주신, 김선 교수님 (AI-based Drug Discovery), 류성호 교수님, 남진우 교수님께 감사를 표시하고 싶습니다. 마지막으로 부족한 시간에도 불구하고 강의 부탁을 흔쾌히 허락하시고 훌륭한 강의자료를 만드는데 노력하셨을 뿐만 아니라 실시간 온라인 Q&A 세션까지 참여해 수고해 주시는 모든 연사분들께 깊이 감사드립니다.

2021년 2월

한국생명정보학회장 김동섭

강의개요

Dimensionality Reduction

바이오 데이터는 고차원의 데이터로 구성되어 있는 경우가 많다. 일례로 유전자 발현 데이터를 생각하면, 일반적으로 유전자가 변수(feature)가 되기에 인간의 경우 20,000 차원의 이상으로 구성된 데이터라 할 수 있으며 이러한 고차원의 데이터를 이용하여 분석을 진행하는 것은 계산학적으로도 간단한 일이 아니다.

본 강의에서는 이러한 고차원 데이터를 분석할 때 활용할 수 있는 dimensionality reduction (차원 축소) 개념과 대표적인 방법들, 활용 방안에 대해 설명한다. 가장 대표적인 방법인 PCA (principal component analysis)를 비롯하여 ICA (independent component analysis), NMF (non-negative matrix factorization)등 여러 방법론에 대해 간단히 소개한다. 또한 deep neural networks을 이용하여 dimensionality reduction 목적으로 사용할 수 있는 autoencoder의 개념과 활용에 대해서도 설명한다. 각 방법들에 대한 기본 개념 소개와 함께 실제 데이터를 이용한 활용 예시를 통해 이러한 방법들이 바이오 데이터 분석에 어떻게 활용될 수 있는지 보인다.

본 강의에서는 수학적인 부분을 아예 제외할 수는 없으나, 각 방법의 특성을 비롯한 주요 개념 이해를 목적으로 설명한다. 또한 이론 위주의 강의이나 R이나 python을 이용하여 해당 방법들을 어떻게 활용할 수 있는지에 대해서도 간단히 소개한다. 직접적인 R이나 python 실습이 진행되지는 않지만, 강의 중 소개되는 코드 이해를 위해서는 R과 python 기초 문법을 사전에 알고 있으면 도움이 될 수 있다.

***참고강의교재:**
강의 자료 참조

***교육생준비물:**
강의 수강을 위한 개인 컴퓨터

*** 강의: 이제근 교수 (숭실대학교 의생명시스템학부)**

Curriculum Vitae

Speaker Name: Je-Keun Rhee, Ph.D.

▶Personal Info

Name Je-Keun Rhee

Title Assistant Professor

Affiliation Soongsil University

▶Contact Information

369 Sang-doro, Sangdo-dong, Dongjak-gu, Seoul

Email jkrhee@ssu.ac.kr

Phone Number 02-828-7038

Research interest : Cancer Genomics, Machine Learning

Educational Experience

2004	B.S. in Life Science, Korea University, Korea
2004	B.S. in Computer Science & Engineering, Korea University, Korea (Double Major)
2014	Ph.D. in Bioinformatics, Seoul National University, Korea

Professional Experience

2011	Visiting Scholar, School of Informatics and Computing (SoIC), Indiana University, USA
2014-2018	Research Professor, Cancer Research Institute / Department of Medical Informatics, The Catholic University of Korea, Korea
2018-2019	Assistant Professor, School of Dentistry, Pusan National University, Korea
2019-	Assistant Professor, School of Systems Biomedical Sciences, Soongsil University, Korea

Selected Publications (5 maximum)

1. Joong Chae Na, Inbok Lee, Je-Keun Rhee, and Soo-Yong Shin, Fast single individual haplotyping method using GPGPU, Computers in Biology and Medicine, 113:103421, 2019. (Co-coresponding author)
2. Je-Keun Rhee, Soo-Jin Kim, Byoung-Tak Zhang, Identifying DNA Methylation Modules Associated with a Cancer by Probabilistic Evolutionary Learning , IEEE Computational Intelligence Magazine, 13(3): 12-19, 2018.
3. Joong Chae Na, Jong-Chan Lee, Je-Keun Rhee, and Soo-Yong Shin, PEATH: Single Individual Haplotyping by Probabilistic Evolutionary Algorithm with Toggling, Bioinformtics, 34(11): 1801-1807, 2018. (Co-coresponding author)
4. Je-Keun Rhee*, Yu Chae Jung*, Kyu Ryung Kim, Jinseon Yoo, Jeeyoon Kim, Yong-Jae Lee, Yoon Ho Ko, Han Hong Lee, Byoung Chul Cho, and Tae-Min Kim, Impact of tumor purity on immune gene expression and clustering analyses across multiple cancer types, Cancer Immunology Research, 6(1):87-97, 2018.
5. Je-Keun Rhee, Sejoon Lee, Woong-Yang Park, Young-Ho Kim, Tae-Min Kim., Allelic imbalance of somatic mutations in cancer genomes and transcriptomes, Scientific Reports, 7: 1653, 2017.

KSBi-BIML 2021

Dimensionality Reduction

Rhee, Je-Keun

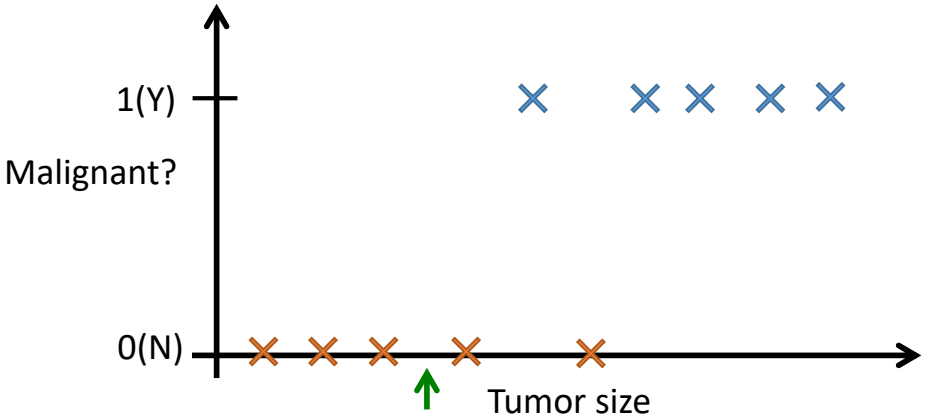
Soongsil University (숭실대 이제근)

본 강의 자료는 한국생명정보학회가 주관하는 KSBi-BIML 2021 워크샵 온라인 수업을 목적으로 제작된것으로 해당 목적 이외의 다른 용도로 사용할 수 없음을 분명하게 알립니다. 수업 목적으로 배포 및 전송 받은 경우에도 이를 다른 사람과 공유하거나 복제, 배포, 전송할 수 없습니다.

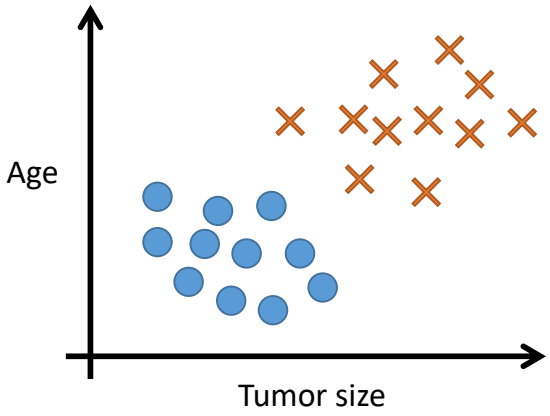
만약 이러한 사항을 위반할 경우 발생하는 모든 법적 책임은 전적으로 불법 행위자 본인에게 있음을 경고합니다.

2

Cancer (Malignant, Benign)

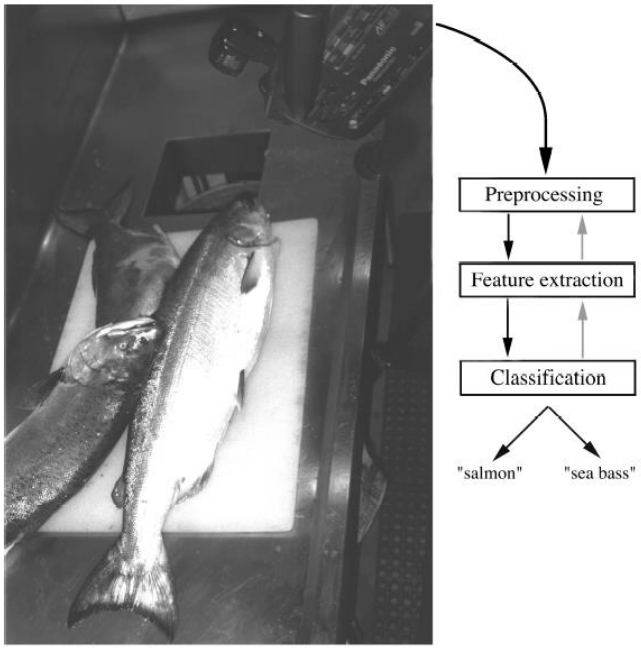


Cancer Classification



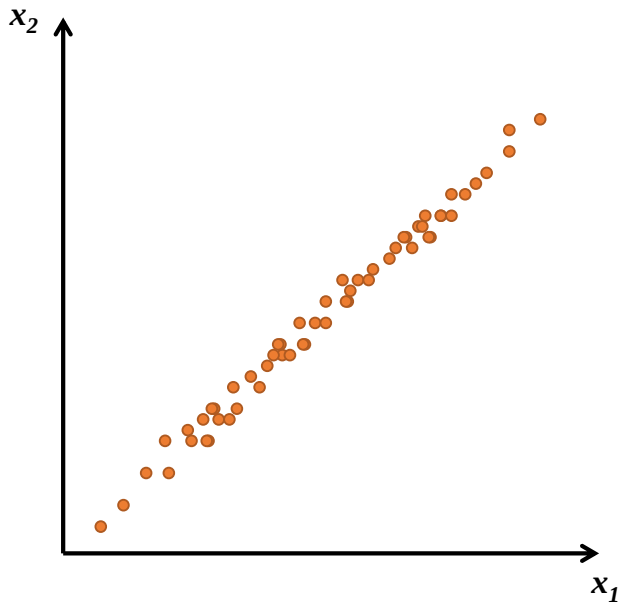
- Clump Thickness
- Uniformity of Cell Size
- Uniformity of Cell Shape
- …

Dimension Reduction / Feature Selection

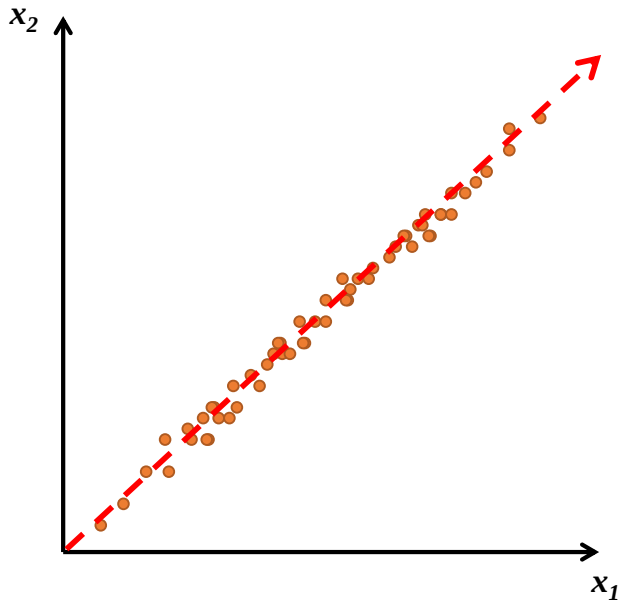


from Pattern Classification (book)

Principal component analysis (PCA)



(0.9, 1.0), (1.3, 1.4), (1.9, 1.8), (2.2, 2.2), (2.4, 2.3), ….



From k original variables: $x_1,x_2,\ldots,x_k$:

Produce k new variables: $y_1,y_2,\ldots,y_k$:

$$y_1 = a_{11}x_1 + a_{12}x_2 + \ldots + a_{1k}x_k$$
$$y_2 = a_{21}x_1 + a_{22}x_2 + \ldots + a_{2k}x_k$$
$$\ldots$$
$$y_k = a_{k1}x_1 + a_{k2}x_2 + \ldots + a_{kk}x_k$$

y_k's are **Principal Components**

such that:

y_k's are orthogonal

y_1 explains as much as possible of original variance in data set

y_2 explains as much as possible of remaining variance etc.

$\{a_{11},a_{12},\ldots,a_{1k}\}$ is 1st **Eigenvector** of covariance matrix, and **coefficients** of first principal component

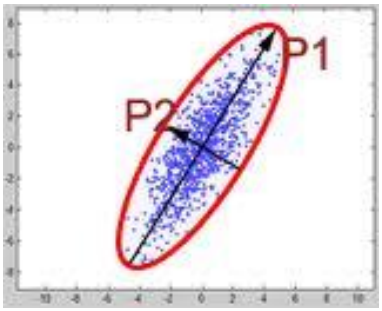
$\{a_{21},a_{22},\ldots,a_{2k}\}$ is 2nd **Eigenvector** of covariance matrix, and **coefficients** of 2nd principal component

…

$\{a_{k1},a_{k2},\ldots,a_{kk}\}$ is kth **Eigenvector** of covariance matrix, and **coefficients** of kth principal component

Computation of PCA

- The direction is given by the eigenvector γ_1 correponding to the largest eigenvalue of covariance matrix **C**
- The second vector that is orthogonal (uncorrelated) to the first is the one that has the second highest variance which comes to be the eignevector corresponding to the second eigenvalue
- And so on …



Calculating eigenvalues and eigenvectors

- The eigenvalues λ_i are found by solving the equation

  $Cx = \lambda x$

  $det(C-\lambda I)=0$

- Eigenvectors are columns of the matrix *A* such that

  $C=A\ D\ A^T$

  where D is a diagonal matrix

$$D = \begin{pmatrix} \lambda_1 & 0 & \ldots\ldots & 0 \\ 0 & \lambda_2 & \ldots\ldots & 0 \\ 0 & & & \\ 0 & \ldots\ldots\ldots & & \lambda_p \end{pmatrix}$$

- In practice, the eigenvectors are computed by SVD (singular value decomposition).

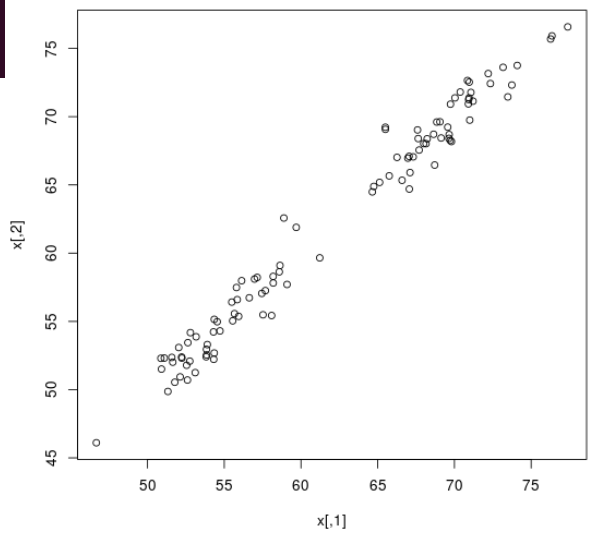
```
> X <- matrix(rnorm(32000), 1000, 32)
> dim(X)
[1] 1000   32
> pca_X <- prcomp(X)
> pca_X
Standard deviations (1, .., p=32):
 [1] 1.1611268 1.1515482 1.1477628 1.1316116 1.1165251 1.1021387 1.0869687
 [8] 1.0770949 1.0669926 1.0638087 1.0539471 1.0525123 1.0368192 1.0296589
[15] 1.0204239 1.0106765 1.0014669 0.9868647 0.9819980 0.9618256 0.9587249
[22] 0.9491361 0.9463531 0.9391354 0.9229802 0.9129457 0.9069524 0.8888694
[29] 0.8832288 0.8669869 0.8599976 0.8498946

Rotation (n x k) = (32 x 32):
               PC1          PC2         PC3          PC4          PC5
 [1,]  0.1273681775 -0.141384568  0.10253371 -0.025551064  0.063805435
 [2,] -0.0586362373 -0.097460978 -0.23289460  0.033280613 -0.057664530
 [3,] -0.0568348643 -0.145480526 -0.03000443 -0.364971502 -0.165182630
 [4,] -0.0267425081  0.217378536 -0.16041250  0.014201023 -0.256952950
 [5,] -0.4712193258 -0.166847283 -0.09229190 -0.059443440 -0.046960081
 [6,] -0.1614406885 -0.166437088 -0.21686270  0.168164903 -0.242285265
 [7,] -0.0014116763 -0.191471560  0.03982241 -0.050303873 -0.003665553
 [8,]  0.1099764928  0.400860439 -0.38249763  0.024063732  0.068599711
```

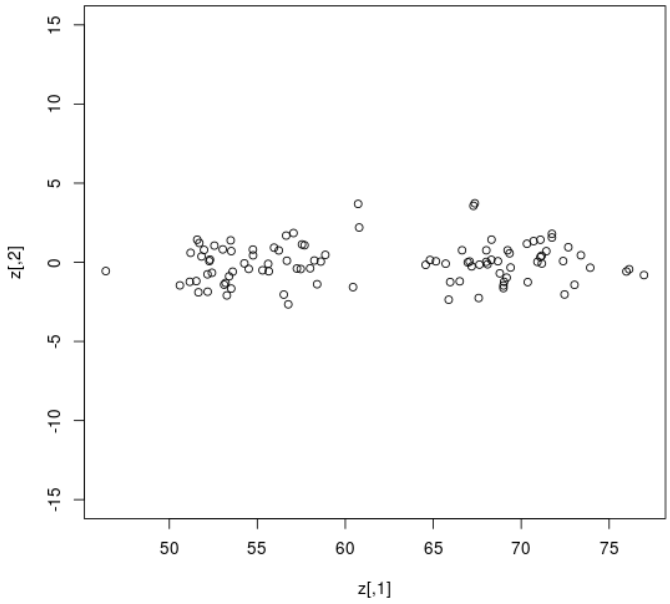


```
> summary(pca_X)
Importance of components:
                          PC1     PC2     PC3     PC4     PC5     PC6     PC7
Standard deviation     1.16113 1.15155 1.14776 1.13161 1.11653 1.10214 1.08697
Proportion of Variance 0.04146 0.04078 0.04051 0.03938 0.03834 0.03736 0.03633
Cumulative Proportion  0.04146 0.08224 0.12275 0.16213 0.20047 0.23783 0.27416
                          PC8     PC9    PC10    PC11    PC12    PC13   PC14
Standard deviation     1.07709 1.06699 1.0638 1.05395 1.05251 1.03682 1.0297
Proportion of Variance 0.03568 0.03501 0.0348 0.03416 0.03407 0.03306 0.0326
Cumulative Proportion  0.30984 0.34485 0.3796 0.41381 0.44788 0.48094 0.5135
                         PC15    PC16    PC17    PC18    PC19    PC20    PC21
Standard deviation     1.02042 1.01068 1.00147 0.98686 0.98200 0.96183 0.95872
Proportion of Variance 0.03202 0.03141 0.03084 0.02995 0.02966 0.02845 0.02827
Cumulative Proportion  0.54557 0.57698 0.60782 0.63777 0.66743 0.69588 0.72414
                         PC22    PC23    PC24   PC25    PC26   PC27   PC28
Standard deviation     0.9491 0.94635 0.93914 0.9230 0.91295 0.9070 0.8889
Proportion of Variance 0.0277 0.02754 0.02712 0.0262 0.02563 0.0253 0.0243
Cumulative Proportion  0.7519 0.77939 0.80651 0.8327 0.85834 0.8836 0.9079
                         PC29    PC30    PC31    PC32
Standard deviation     0.88323 0.86699 0.86000 0.84989
Proportion of Variance 0.02399 0.02312 0.02274 0.02221
Cumulative Proportion  0.93193 0.95504 0.97779 1.00000
>
```

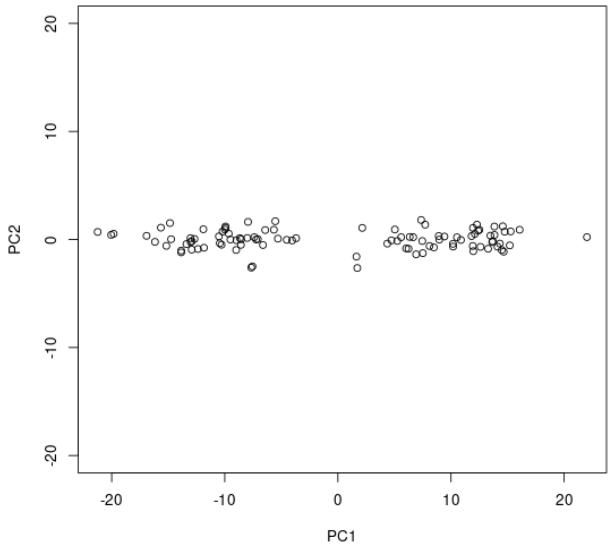
```
> set.seed(1988)
> library(MASS)
> n <- 100
> Sigma <- matrix(c(9, 9 * 0.9, 9 * 0.92, 9 * 1), 2, 2)
> x <- rbind(mvrnorm(n / 2, c(69, 69), Sigma),
+            mvrnorm(n / 2, c(55, 55), Sigma))
>
> dim(x)
[1] 100   2
```




```
> z  <- cbind((x[,2] + x[,1])/2,  x[,2] - x[,1])
```



```
> summary(prcomp(x))
Importance of components:
                          PC1     PC2
Standard deviation     11.357 0.88114
Proportion of Variance  0.994 0.00598
Cumulative Proportion   0.994 1.00000
```



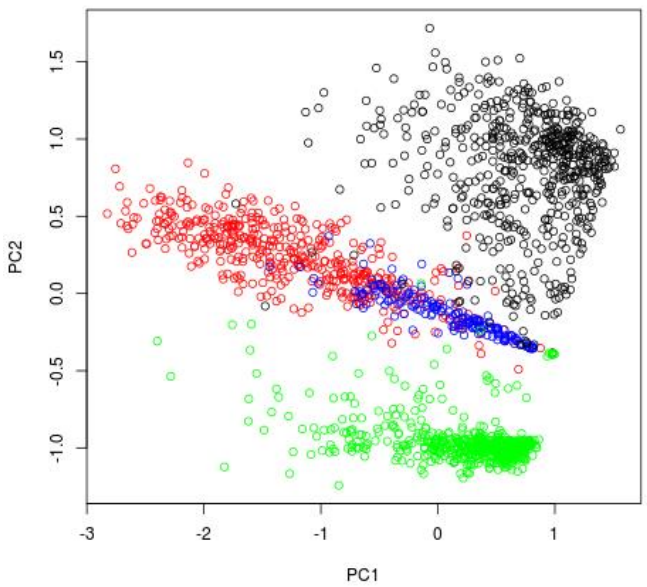

```
> head(USArrests)
           Murder Assault UrbanPop Rape
Alabama      13.2     236       58 21.2
Alaska       10.0     263       48 44.5
Arizona       8.1     294       80 31.0
Arkansas      8.8     190       50 19.5
California    9.0     276       91 40.6
Colorado      7.9     204       78 38.7
>
> pca_usarrests <- prcomp(USArrests)
> summary(pca_usarrests)
Importance of components:
                          PC1      PC2    PC3     PC4
Standard deviation     83.7324 14.21240 6.4894 2.48279
Proportion of Variance  0.9655  0.02782 0.0058 0.00085
Cumulative Proportion   0.9655  0.99335 0.9991 1.00000
>
> pca_usarrests$rotation
                PC1         PC2         PC3         PC4
Murder   0.04170432 -0.04482166  0.07989066 -0.99492173
Assault  0.99522128 -0.05876003 -0.06756974  0.03893830
UrbanPop 0.04633575  0.97685748 -0.20054629 -0.05816914
Rape     0.07515550  0.20071807  0.97408059  0.07232502
```

sklearn.decomposition.PCA

class `sklearn.decomposition.` **PCA**(*n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', random_state=None*) [source]

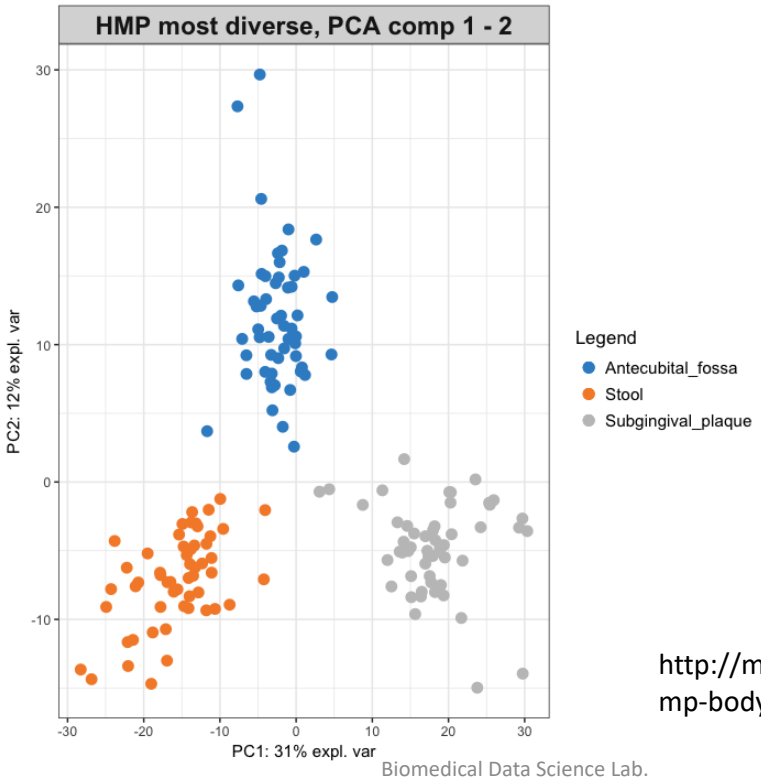
https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html

PCA example for TCGA expression data



Red: COADREAD
Blue: GBM
Black: HNSC
Green: PRAD

PCA example for microbiome RNAseq



http://mixomics.org/mixmc/case-study-hmp-bodysites-repeated-measures/

Independent Component Analysis (ICA)

ICA

- Independent component analysis (ICA) is a method for finding underlying factors or components from multivariate (multi-dimensional) statistical data.
- It looks for components that are both *statistically independent*, and *nonGaussian*.

- The two broadest definitions of independence for ICA are

  1) Minimization of Mutual Information
  2) Maximization of non-Gaussianity

ICA

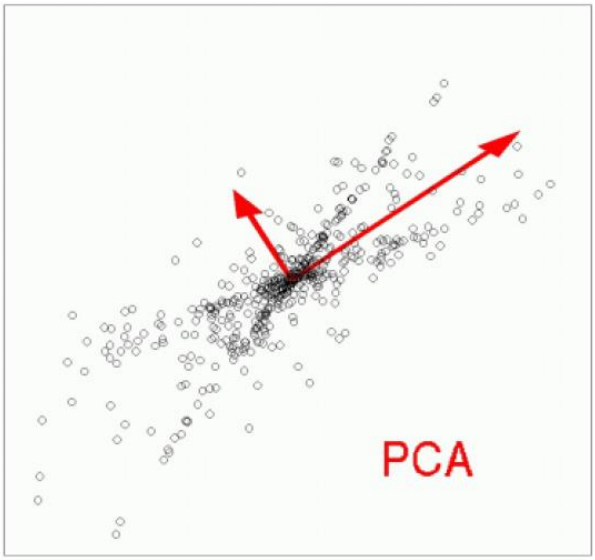
- For example, given two-dimensional vector , $\boldsymbol{x} = [\, x_1 \, x_2 \,]^T$, ICA aims at finding the following decomposition

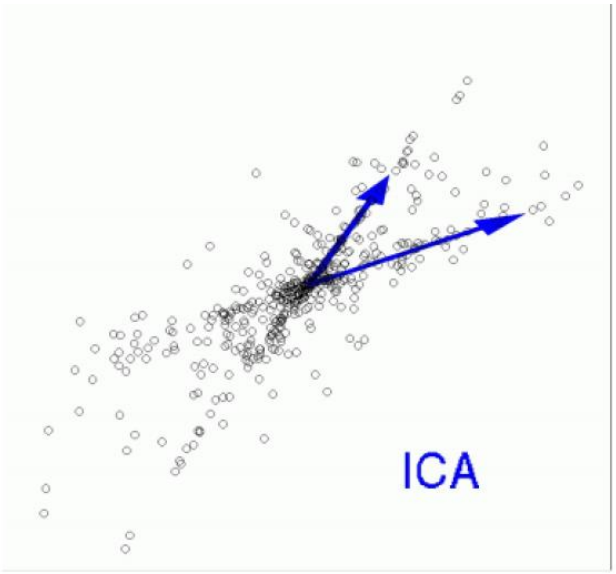
$$\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} a_{11} \\ a_{21} \end{bmatrix} s_1 + \begin{bmatrix} a_{12} \\ a_{22} \end{bmatrix} s_2$$

$$\boldsymbol{x} = \boldsymbol{a}_1 s_1 + \boldsymbol{a}_2 s_2$$

where $\boldsymbol{a}_1$, $\boldsymbol{a}_2$ are basis vectors and s_1, s_2 are basis coefficients.

- Constraint: Basis coefficients s_1 and s_2 are statistically independent





sklearn.decomposition.FastICA

class `sklearn.decomposition.` **`FastICA`**(*n_components=None*, ***, *algorithm='parallel'*, *whiten=True*, *fun='logcosh'*, *fun_args=None*, *max_iter=200*, *tol=0.0001*, *w_init=None*, *random_state=None*) [source]

FastICA: a fast algorithm for Independent Component Analysis.

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.FastICA.html

Non-negative Matrix Factorization (NMF)

nature
International journal of science

Letter | Published: 21 October 1999

Learning the parts of objects by non-negative matrix factorization

Daniel D. Lee & H. Sebastian Seung

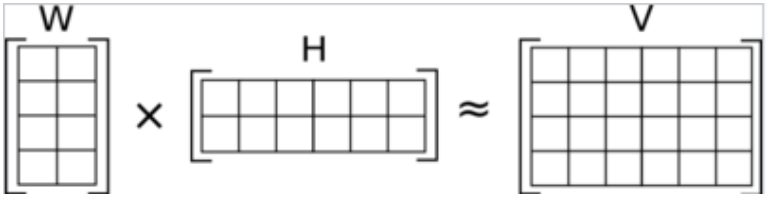
Nature **401**, 788–791 (1999) | Download Citation

Algorithms for Non-negative Matrix Factorization

Daniel D. Lee*
*Bell Laboratories
Lucent Technologies
Murray Hill, NJ 07974

H. Sebastian Seung*†
†Dept. of Brain and Cog. Sci.
Massachusetts Institute of Technology
Cambridge, MA 02138

NMF



From Wikipedia

- **Non-negative matrix factorization** (**NMF** or **NNMF**), also **non-negative matrix approximation** is a group of algorithms in multivariate analysis a nd linear algebra where a matrix **V** is factorized into (usually) two matric es **W** and **H**, with the property that all three matrices have no negative e lements.
- The matrix **V** is represented by the two smaller matrices **W** and **H**, which , when multiplied, approximately reconstruct **V**.

PNAS Proceedings of the National Academy of Sciences of the United States of America

Keyword

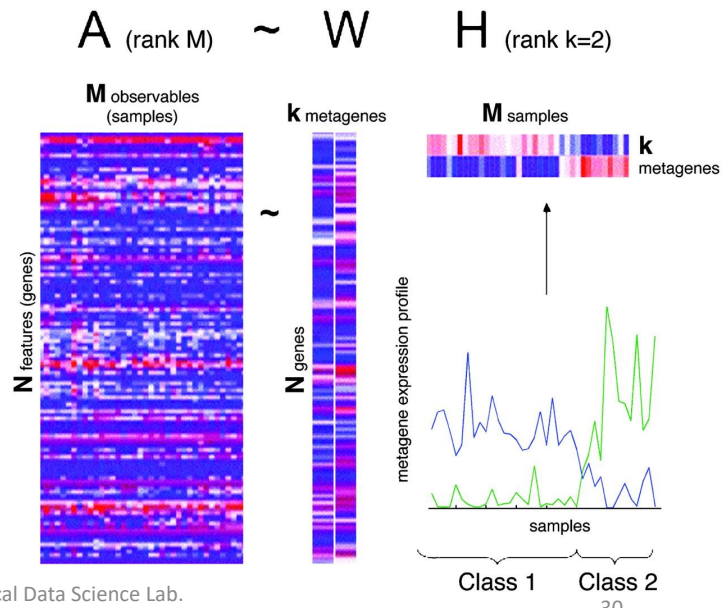
Home | Articles | Front Matter | News | Podcasts | Auth

NEW RESEARCH IN Physical Sciences | Social Sciences

Metagenes and molecular pattern discovery using matrix factorization

Jean-Philippe Brunet, Pablo Tamayo, Todd R. Golub, and Jill P. Mesirov

PNAS March 23, 2004 101 (12) 4164-4169; https://doi.org/10.1073/pnas.0308531101



nature International weekly journal of science

Home | News & Comment | Research | Careers & Jobs | Current Issue | Archive | Audio & Video

Archive > Volume 500 > Issue 7463 > Articles > Article

NATURE | ARTICLE

日本語要約

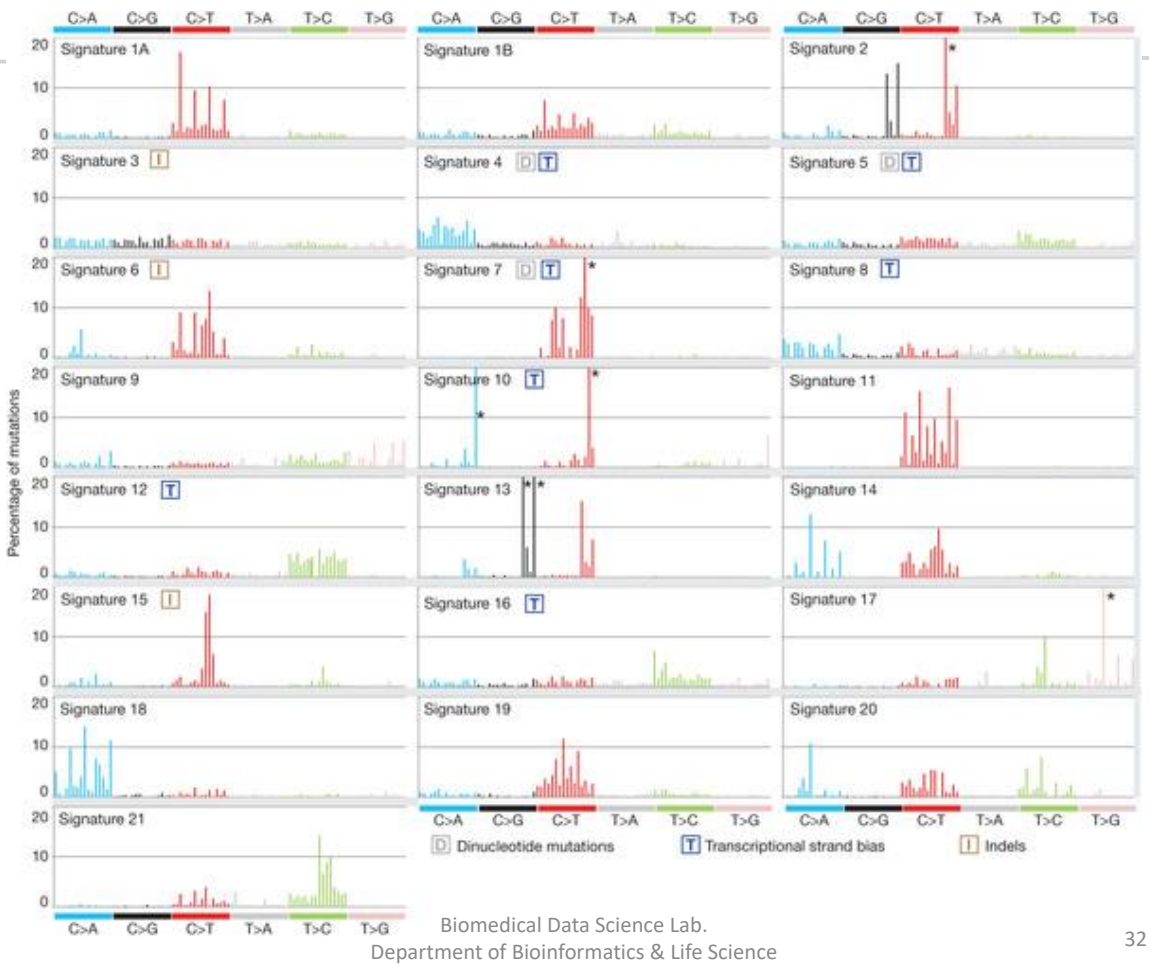
Signatures of mutational processes in human cancer

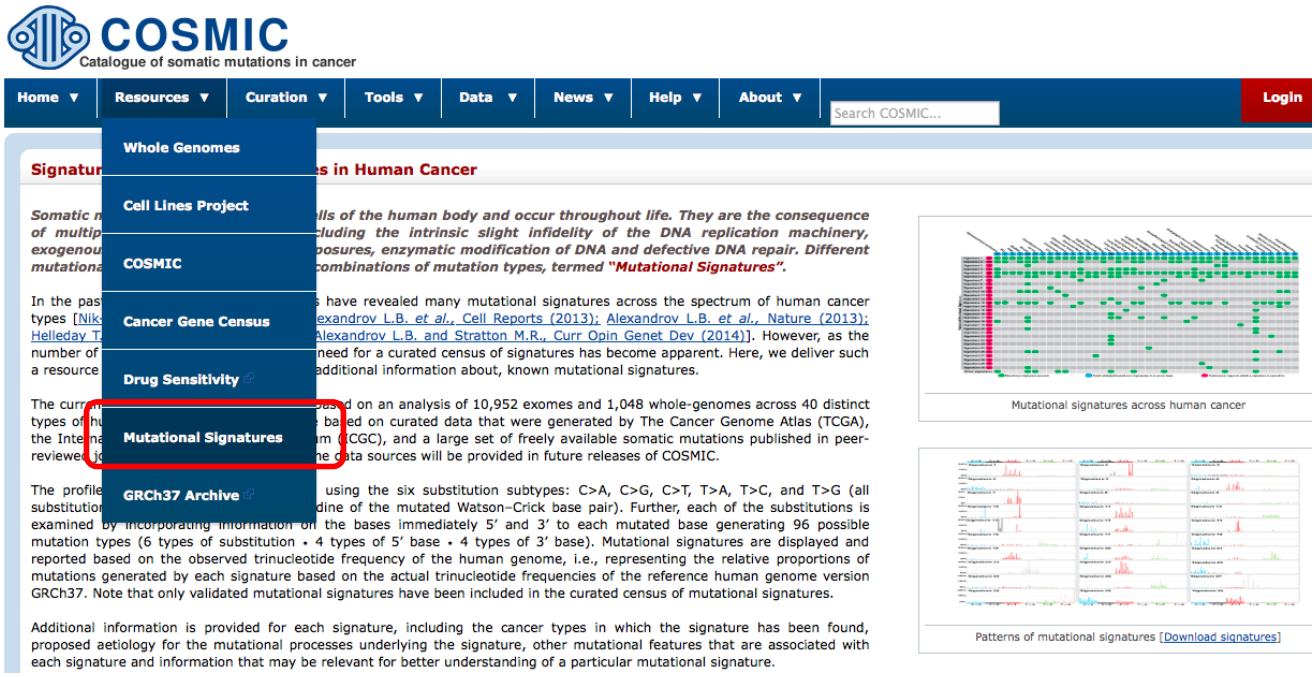
Ludmil B. Alexandrov, Serena Nik-Zainal, David C. Wedge, Samuel A. J. R. Aparicio, Sam Behjati, Andrew V. Biankin, Graham R. Bignell, Niccolò Bolli, Ake Borg, Anne-Lise Børresen-Dale, Sandrine Boyault, Birgit Burkhardt, Adam P. Butler, Carlos Caldas, Helen R. Davies, Christine Desmedt, Roland Eils, Jórunn Erla Eyfjörd, John A. Foekens, Mel Greaves, Fumie Hosoda, Barbara Hutter, Tomislav Ilicic, Sandrine Imbeaud, Marcin Imielinsk + *et al.*

Affiliations | Contributions | Corresponding author

Nature **500**, 415–421 (22 August 2013) | doi:10.1038/nature12477

Received 24 March 2013 | Accepted 19 July 2013 | Published online 14 August 2013






```
# Install
install.packages('NMF')
# Load
library(NMF)
```

```
mat <- matrix(runif(n= 200, min= 0, max= 100), nrow= 30, ncol= 10)

res<- nmf(mat, rank= 2)
w<- basis(res)
h<- coef(res)

w %*% h # ~ mat
dim(w) # 30 2
```


sklearn.decomposition.NMF

class sklearn.decomposition.**NMF**(*n_components=None*, ***, *init='warn'*, *solver='cd'*, *beta_loss='frobenius'*, *tol=0.0001*, *max_iter=200*, *random_state=None*, *alpha=0.0*, *l1_ratio=0.0*, *verbose=0*, *shuffle=False*, *regularization='both'*) [source]

Non-Negative Matrix Factorization (NMF).

Examples

```
>>> import numpy as np
>>> X = np.array([[1, 1], [2, 1], [3, 1.2], [4, 1], [5, 0.8], [6, 1]])
>>> from sklearn.decomposition import NMF
>>> model = NMF(n_components=2, init='random', random_state=0)
>>> W = model.fit_transform(X)
>>> H = model.components_
```

Methods

fit(X[, y])	Learn a NMF model for the data X.
fit_transform(X[, y, W, H])	Learn a NMF model for the data X and returns the transformed data.
get_params([deep])	Get parameters for this estimator.
inverse_transform(W)	Transform data back to its original space.
set_params(**params)	Set the parameters of this estimator.
transform(X)	Transform the data X according to the fitted NMF model.

https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.NMF.html

t-SNE

t-distributed Stochastic Neighbor Embedding (t-SNE)

- T-distributed Stochastic Neighbor Embedding (t-SNE) is a ma chine learning algorithm for visualization developed by Laur ens van der Maaten and Geoffrey Hinton.
- It is a nonlinear dimensionality reduction technique well-sui ted for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions.

```
> install.packages("Rtsne")
```

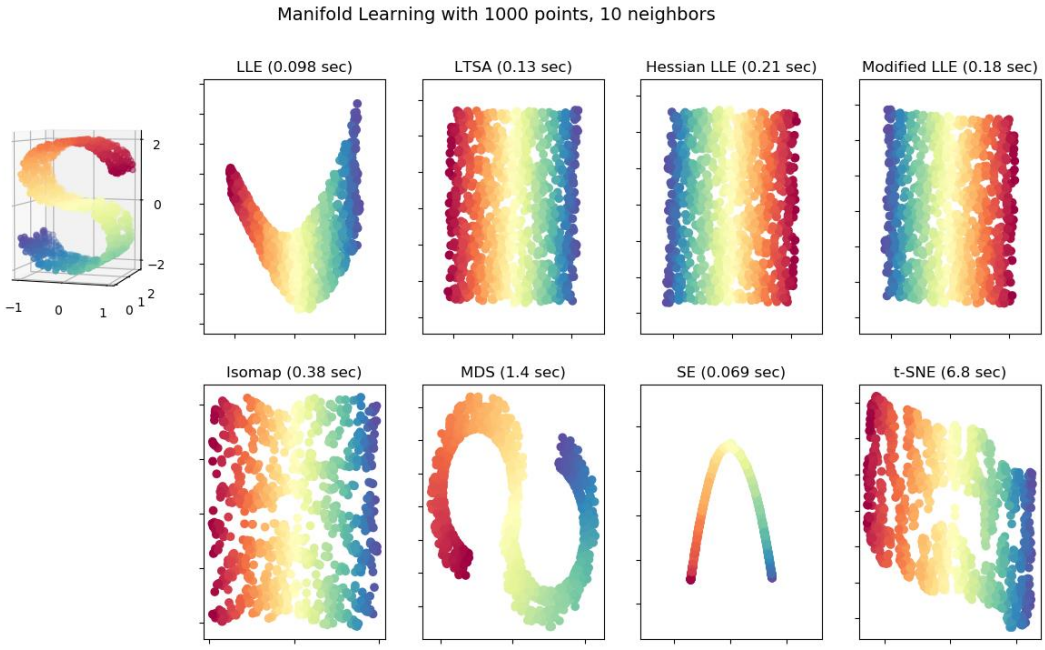
sklearn.manifold.TSNE

class `sklearn.manifold.` **TSNE**(*n_components=2, *, perplexity=30.0, early_exaggeration=12.0, learning_rate=200.0, n_iter=1000, n_iter_without_progress=300, min_grad_norm=1e-07, metric='euclidean', init='random', verbose=0, random_state=None, method='barnes_hut', angle=0.5, n_jobs=None, square_distances='legacy'*) [source]

t-distributed Stochastic Neighbor Embedding.

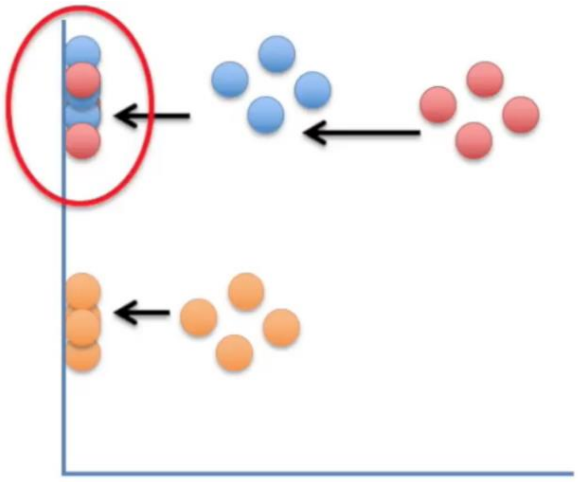
https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

Manifold learning



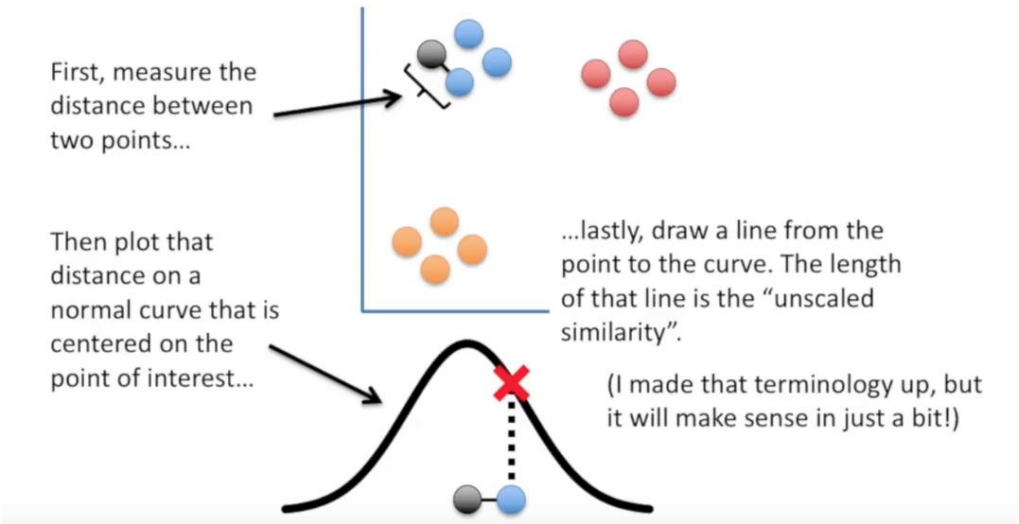
https://scikit-learn.org/stable/modules/manifold.html

PCA



https://www.youtube.com/watch?v=NEaUSP4YerM

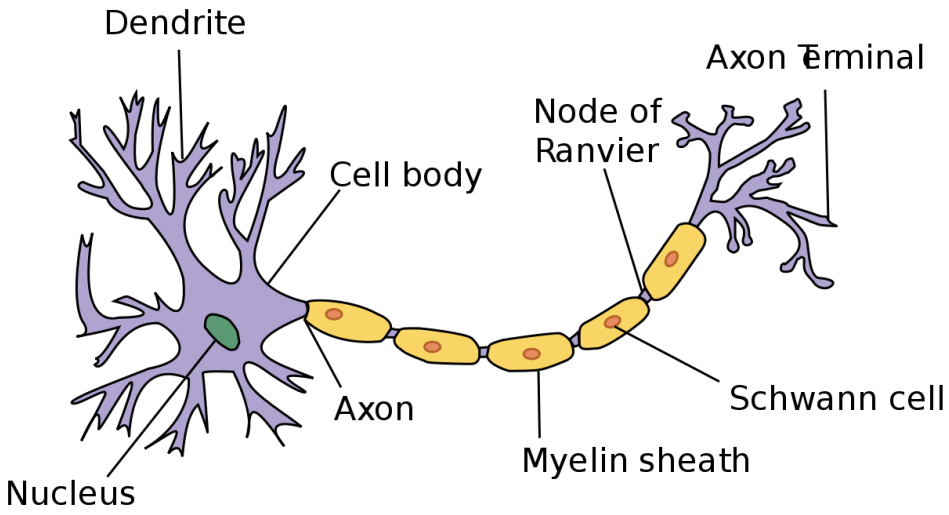
t-SNE



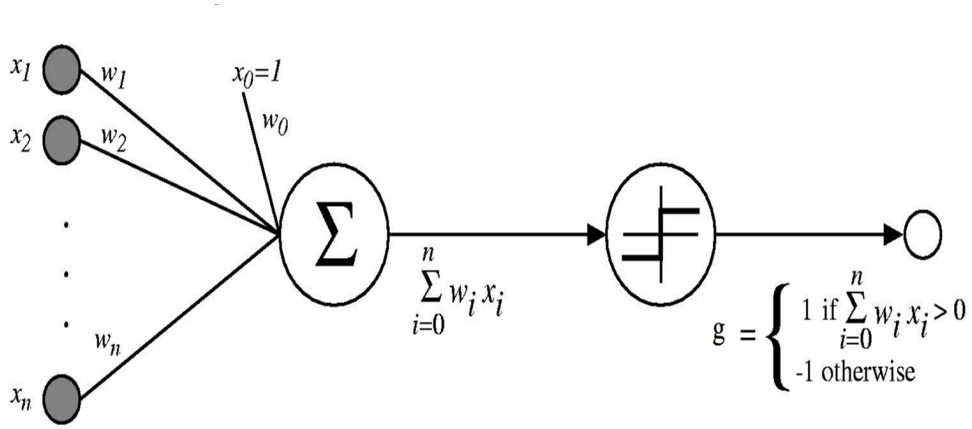
https://www.youtube.com/watch?v=NEaUSP4YerM

Deep Learning & Autoencoder

Human neuron



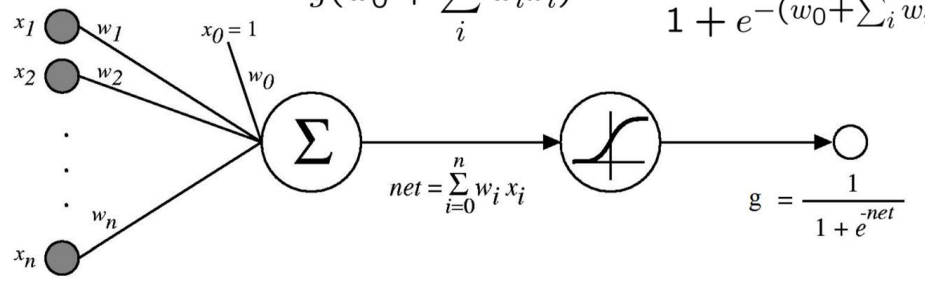
Perceptron as a neuron



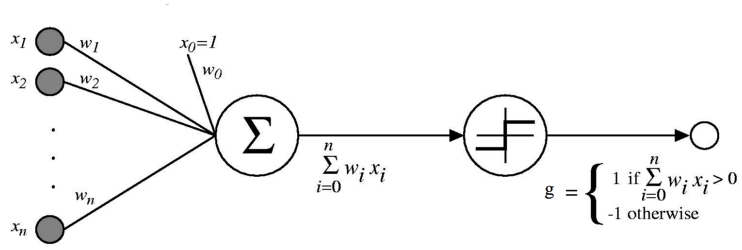
- x_1, x_2, ... , x_n as inputs
- Sum is passed through **activation function** g()

Sigmoid neuron

$$g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + e^{-(w_0 + \sum_i w_i x_i)}}$$



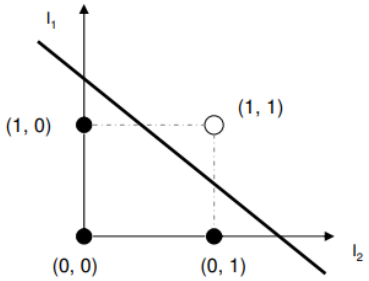
XOR problem



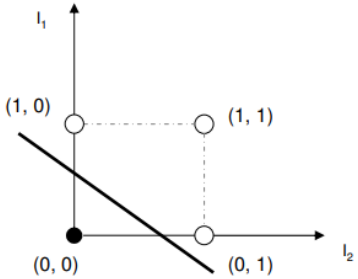
XOR truth table

Input		Output
A	B	
0	0	0
0	1	1
1	0	1
1	1	0

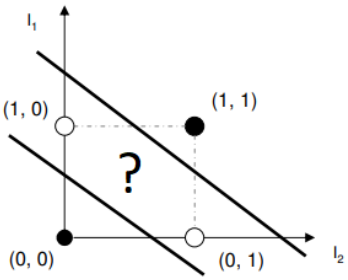
AND		
I_1	I_2	out
0	0	0
0	1	0
1	0	0
1	1	1



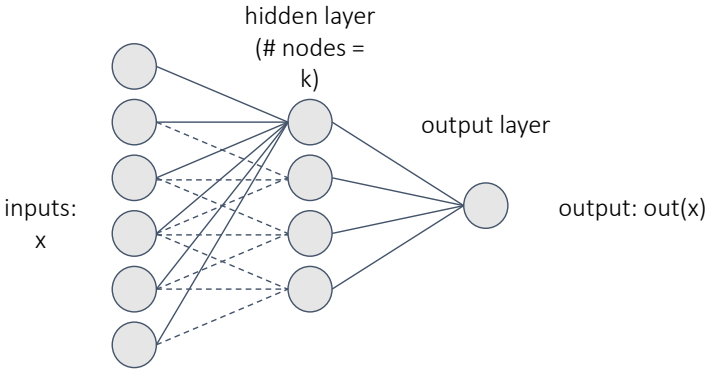
OR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	1



XOR		
I_1	I_2	out
0	0	0
0	1	1
1	0	1
1	1	0



Multilayer percepton



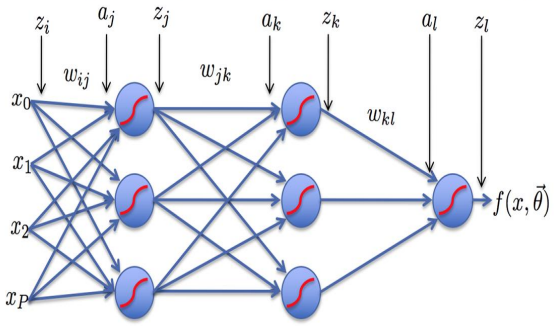
$$out(x) = g(w_0 + \Sigma w_k g(\Sigma w_i^k x_i))$$

Backpropagation of errors

$$R(\theta) = \frac{1}{N}\sum_{n=0}^{N} L(y_n - f(x_n)) \quad \boxed{\text{Empirical Risk Function}}$$

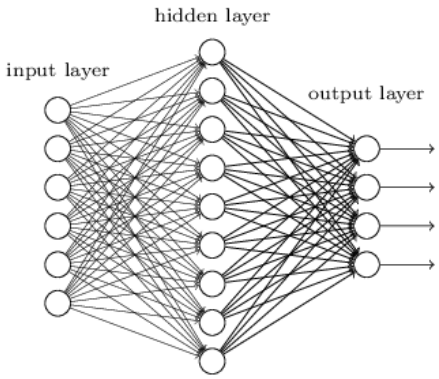
$$= \frac{1}{N}\sum_{n=0}^{N} \frac{1}{2}(y_n - f(x_n))^2$$

$$= \frac{1}{N}\sum_{n=0}^{N} \frac{1}{2}\left(y_n - g\left(\sum_k w_{kl} g\left(\sum_j w_{jk} g\left(\sum_i w_{ij} x_{n,i}\right)\right)\right)\right)^2$$

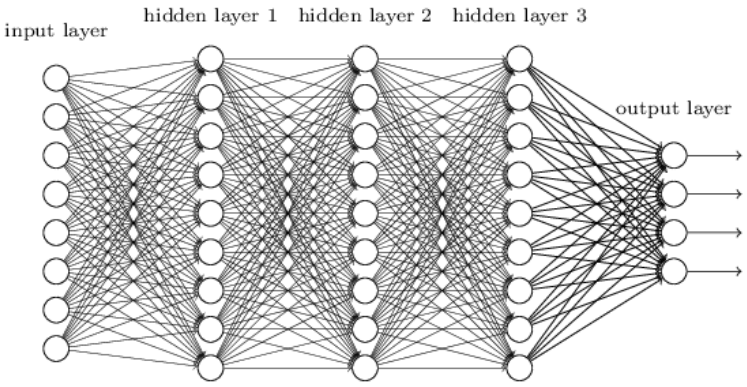


Deep neural networks

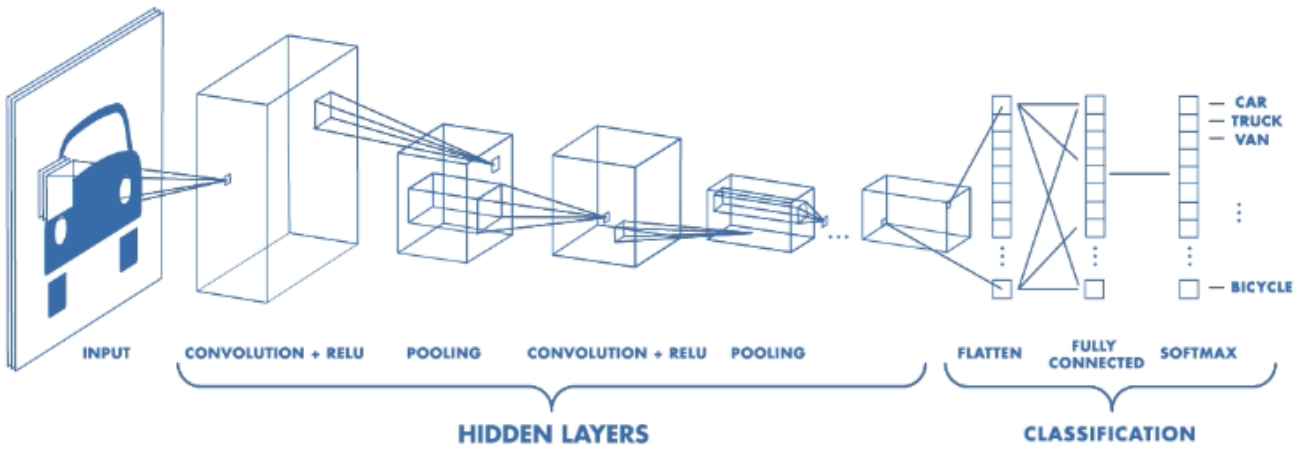
"Non-deep" feedforward neural network



Deep neural network



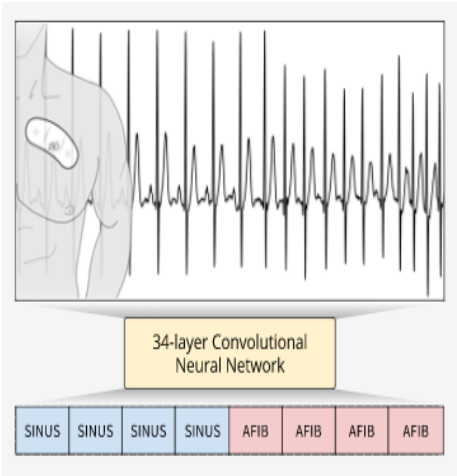
Convolutional Neural Networks

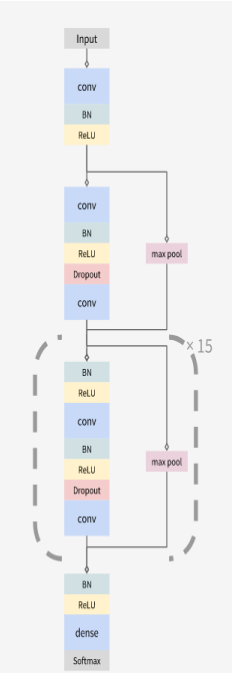


Cardiologist-Level Arrhythmia Detection With Convolutional Neural Networks

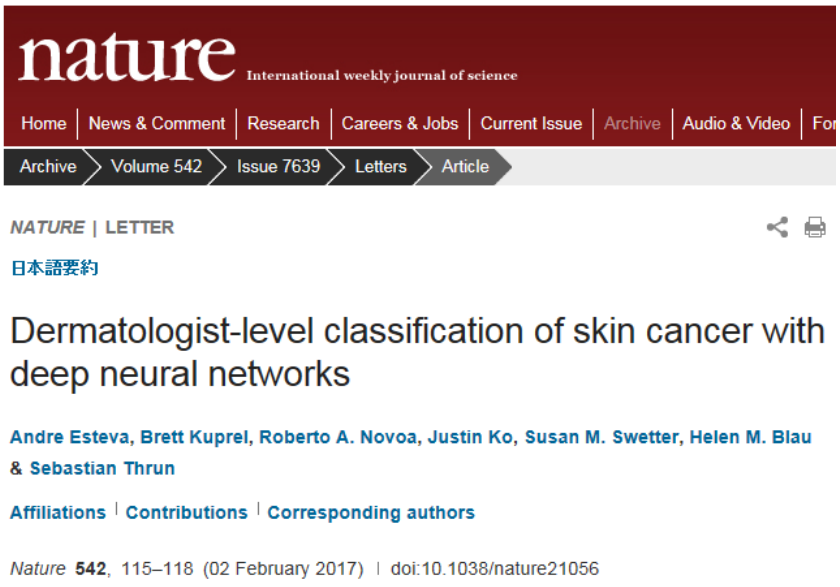
- They have a dataset of 64,121 ECG records from 29,163 patients.
- The model outperforms the cardiologist average on both the Sequence and Set F1 metrics

https://stanfordmlgroup.github.io/projects/ecg/





Skin Cancer Detection



- 5.4 million new cases of skin cancer in USA every year.
- Melanomas represent fewer than 5% but they account for 75% of all skin cancer deaths
- Early detection is critical (survival rate would be 99% at early stage, but 14% at latest stage)

MENU

nature biotechnology

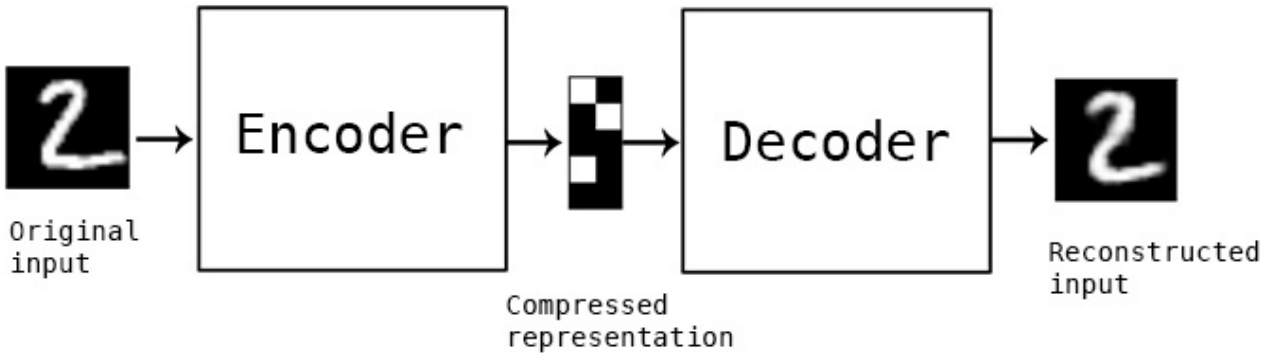
Brief Communication | Published: 29 January 2018

Deep learning improves prediction of CRISPR–Cpf1 guide RNA activity

Hui Kwon Kim, Seonwoo Min, Myungjae Song, Soobin Jung, Jae Woo Choi, Younggwang Kim, Sangeun Lee, Sungroh Yoon ✉ & Hyongbum (Henry) Kim ✉

Nature Biotechnology **36**, 239–241 (2018) | Download Citation ⬇

Autoencoder



Keras example

```
from keras import layers

# This is the size of our encoded representations
encoding_dim = 32

# This is our input image
input_img = keras.Input(shape=(784,))

# "encoded" is the encoded representation of the input
encoded = layers.Dense(encoding_dim, activation='relu')(input_img)

# "decoded" is the lossy reconstruction of the input
decoded = layers.Dense(784, activation='sigmoid')(encoded)

# This model maps an input to its reconstruction
autoencoder = keras.Model(input_img, decoded)
```

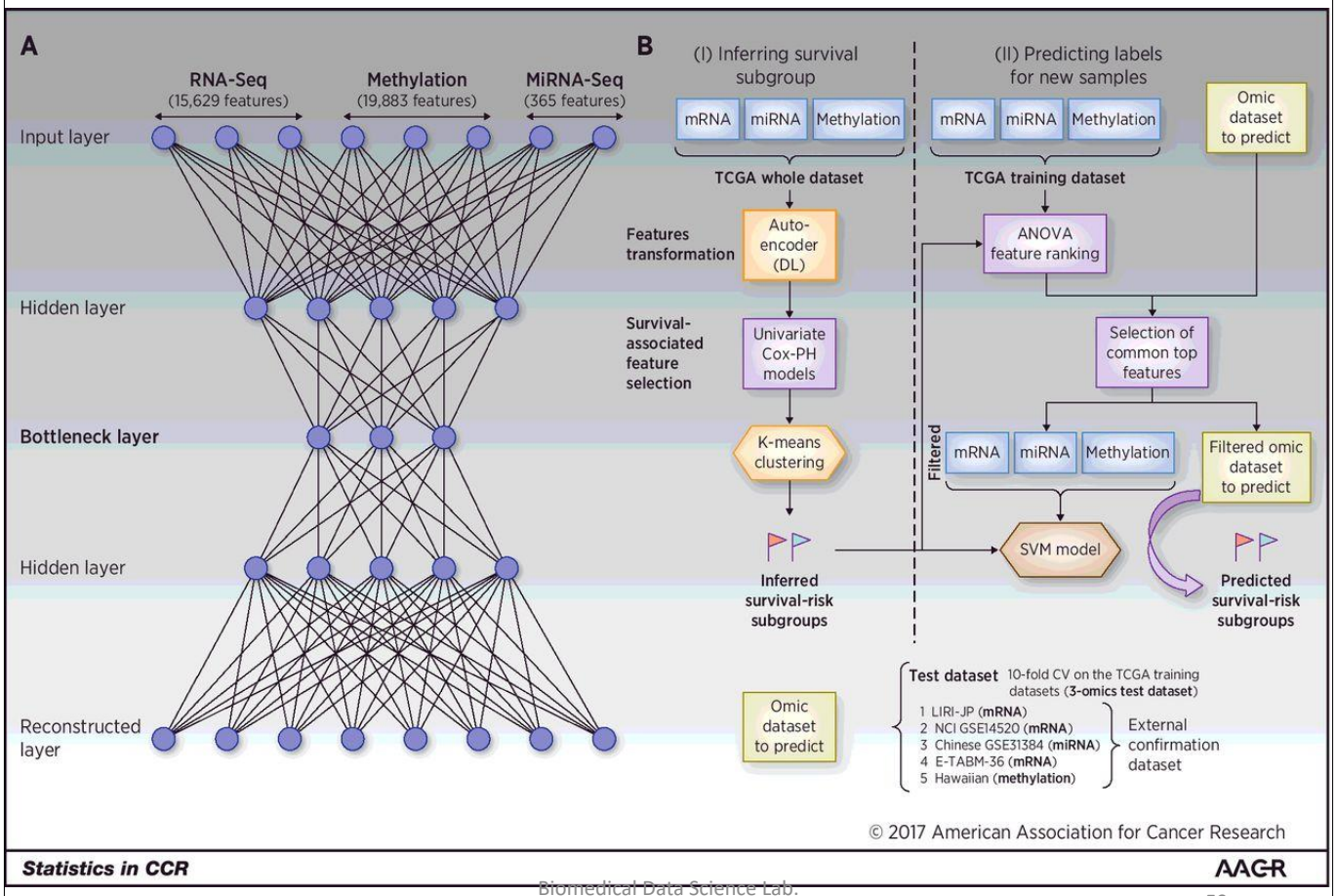

Deep autoencoder

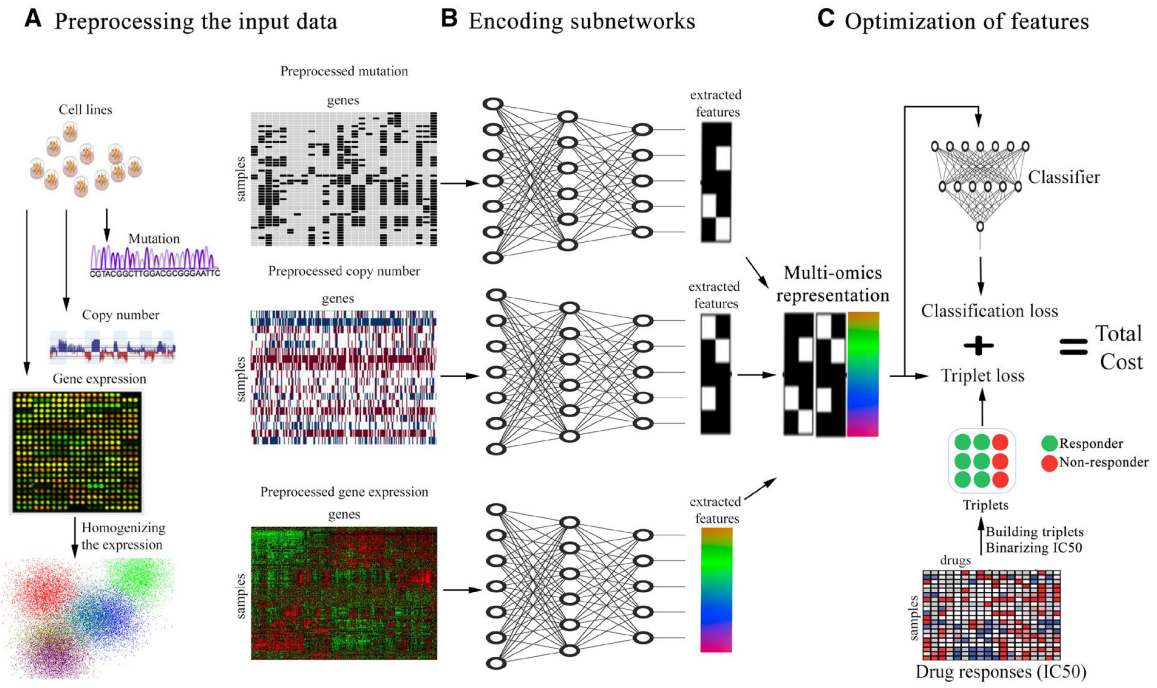
```
input_img = keras.Input(shape=(784,))
encoded = layers.Dense(128, activation='relu')(input_img)
encoded = layers.Dense(64, activation='relu')(encoded)
encoded = layers.Dense(32, activation='relu')(encoded)

decoded = layers.Dense(64, activation='relu')(encoded)
decoded = layers.Dense(128, activation='relu')(decoded)
decoded = layers.Dense(784, activation='sigmoid')(decoded)

autoencoder = keras.Model(input_img, decoded)
autoencoder.compile(optimizer='adam', loss='binary_crossentropy')

autoencoder.fit(x_train, x_train,
                epochs=100,
                batch_size=256,
                shuffle=True,
                validation_data=(x_test, x_test))
```

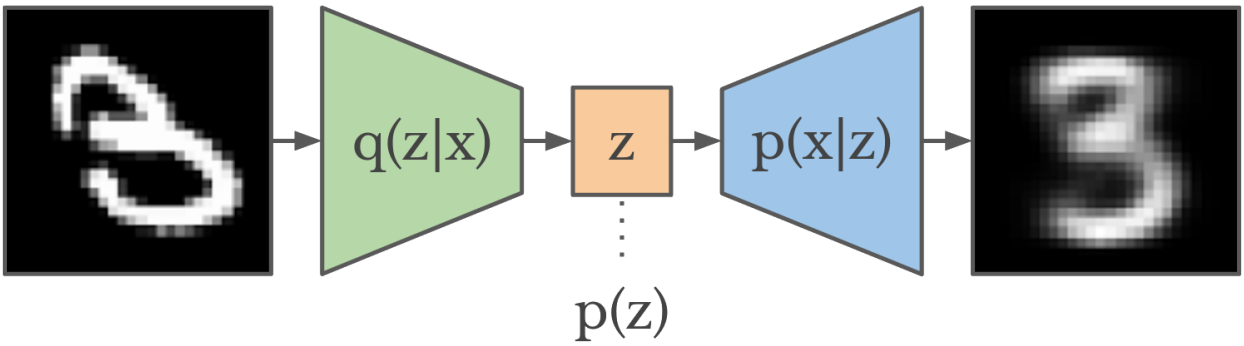




https://academic.oup.com/bioinformatics/article/35/14/i501/5529255

Variational Autoencoder

A VAE consist of three components: an encoder q(z|x), a prior p(z), and a decoder p(x|z).

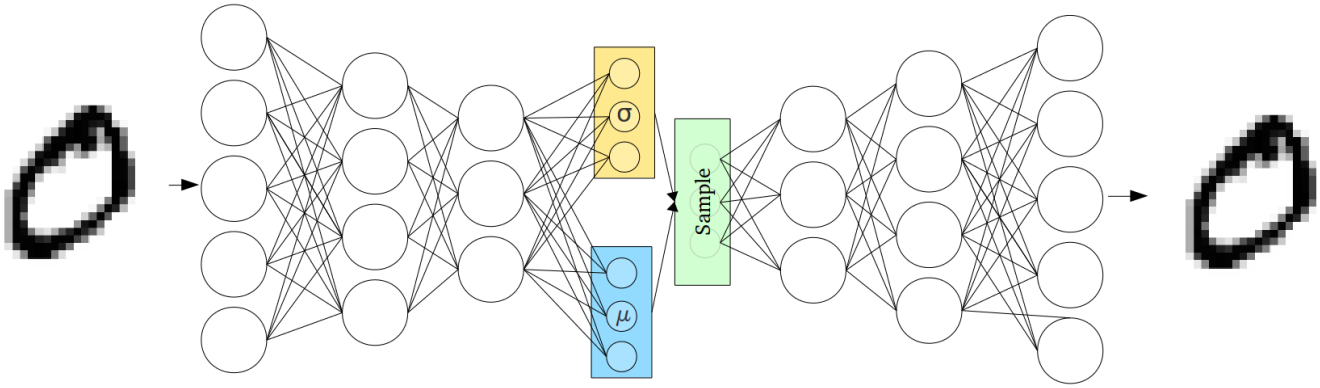


https://danijar.com/building-variational-auto-encoders-in-tensorflow/

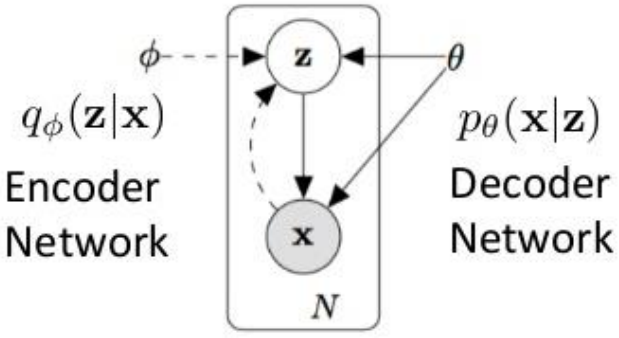
Variational Autoencoder

- Autoencoders learn a “compressed representation” of input (*could be image,text sequence etc.)* automatically by first co mpressing the input (*encoder*) and decompressing it back (*d ecoder*) to match the original input.

- Variational autoencoders learn the parameters of a probabili ty distribution representing the data. Since it learns to mode l the data, we can sample from the distribution and generat e new input data samples.



Variational autoencoder

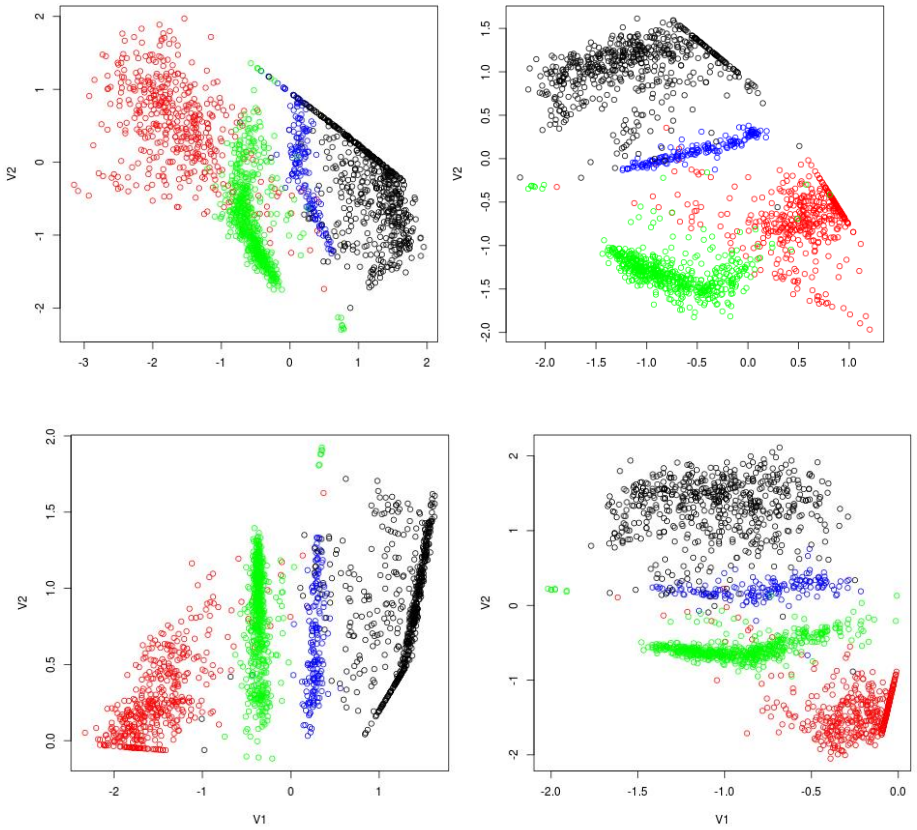


Minimize: $D_{KL}[q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})]$

Intractable: $p_\theta(\mathbf{z}|\mathbf{x}) = \dfrac{p_\theta(\mathbf{x}|\mathbf{z})p_\theta(\mathbf{z})}{p_\theta(\mathbf{x})}$

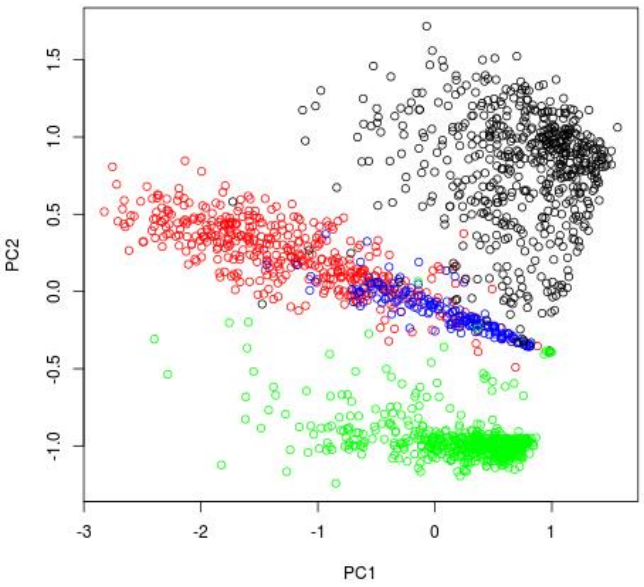
https://www.slideshare.net/ckmarkohchang/variational-autoencoder

VAE – TCGA expression



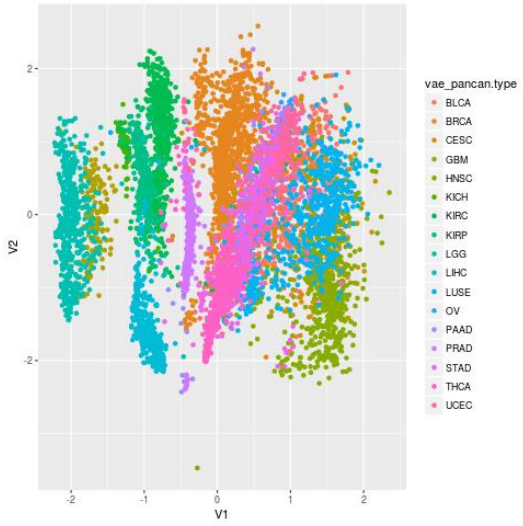
Red: COADREAD
Blue: GBM
Black: HNSC
Green: PRAD

PCA

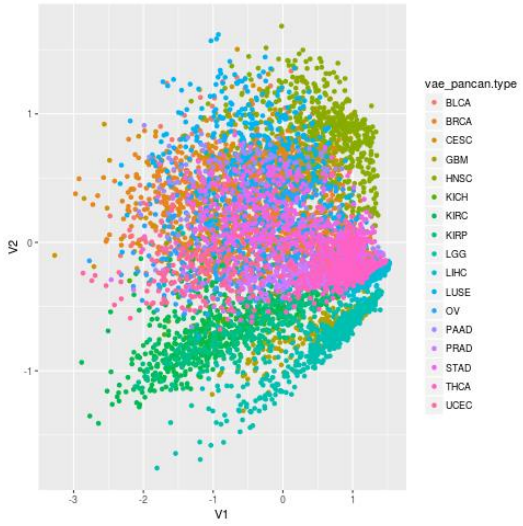


Red: COADREAD
Blue: GBM
Black: HNSC
Green: PRAD

VAE



PCA



Thank you