Genome Informatics

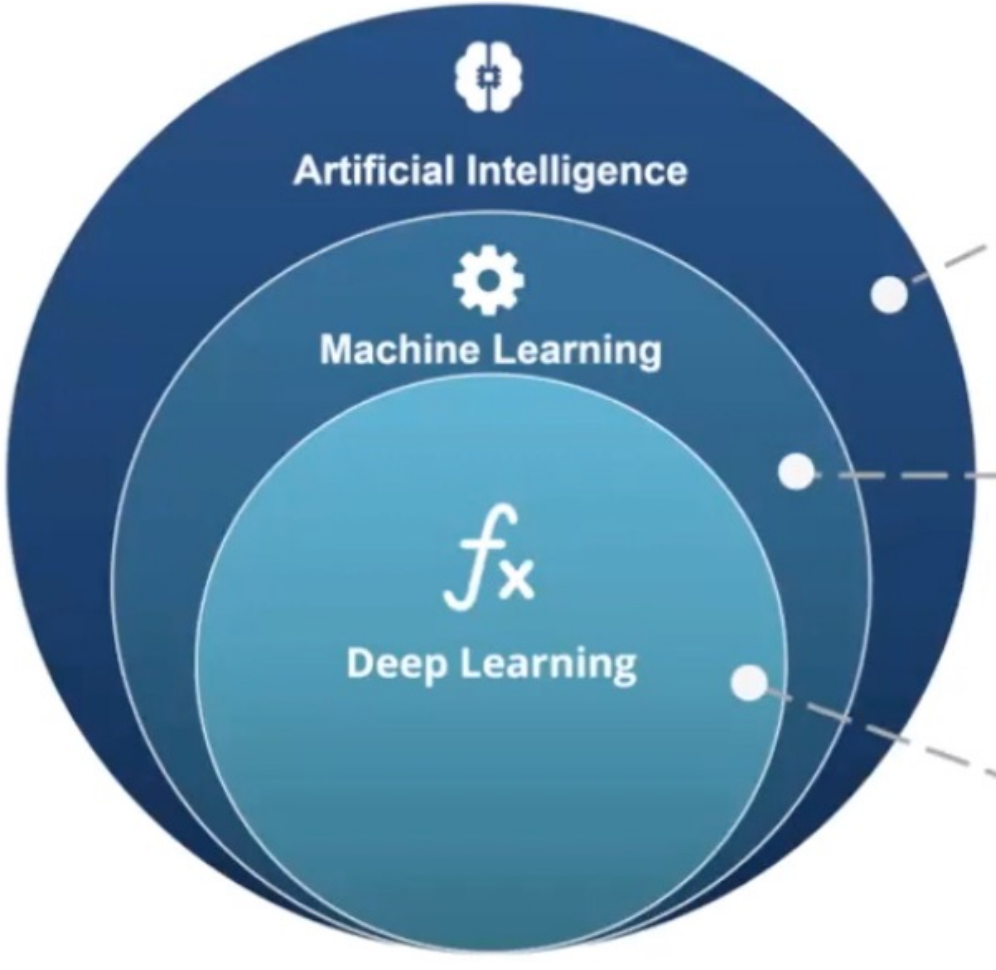
Machine learning II

Guest lecture: Tiam Heydari

Today we will cover:

- What is a Neural Network
- How they work and why they are useful
- What are some challenges to training NNs
- The concept of convolutional neural network and it’s usage examples
- Unsupervised learning: Clustering, dimensionality reduction

Terminology:



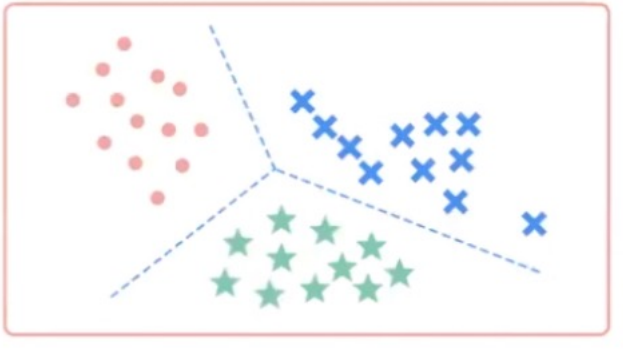
A technique that enables machines to **mimic human behavior**
Ex: Computer plays chess

Statistical methods that allow computer to **optimize models from data**
Ex: Given DNA sequence, predict that chance of type x cancer

Specific class of machine learning models **that uses multiple layers of processing**
Ex: Predicting gene expression from DNA sequence

Reminder: Two classes of machine learning:

Supervised ML



Have a labeled target

Aim: to learn outputs depend inputs

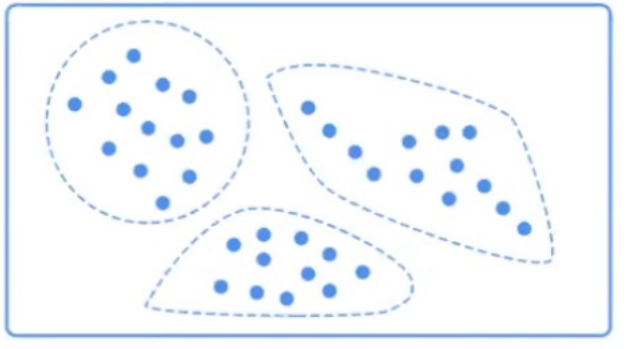
Regression: numeric target

-predict efficiency percentage of a drug

Classification: Categorical target:

-Predict if protein binds to a target or not

Unsupervised ML



No known target variables, unlabeled training data

Aim: to find and learn structure within data

Clustering

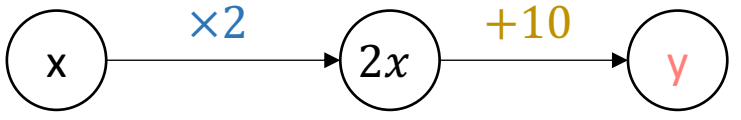
PCA

Autoencoders

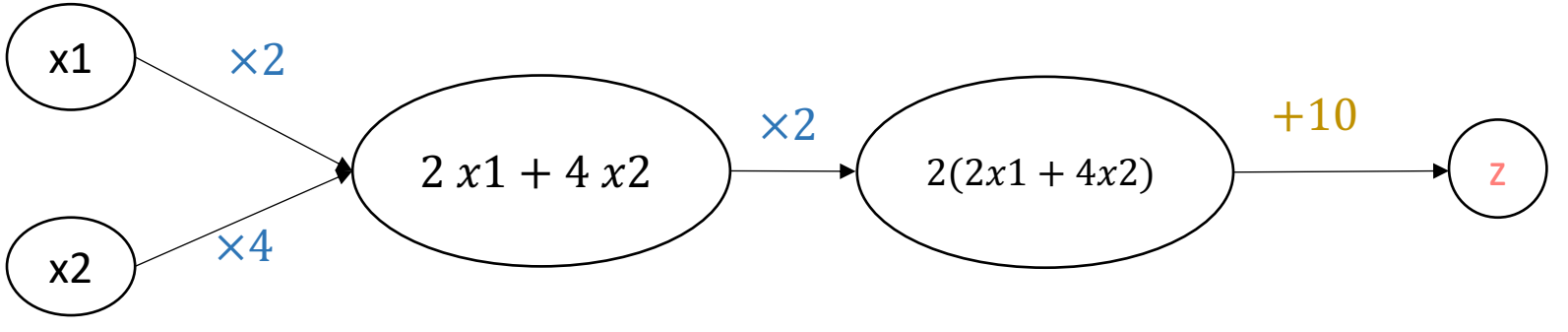
…

Computational graphs:

y=2x+10

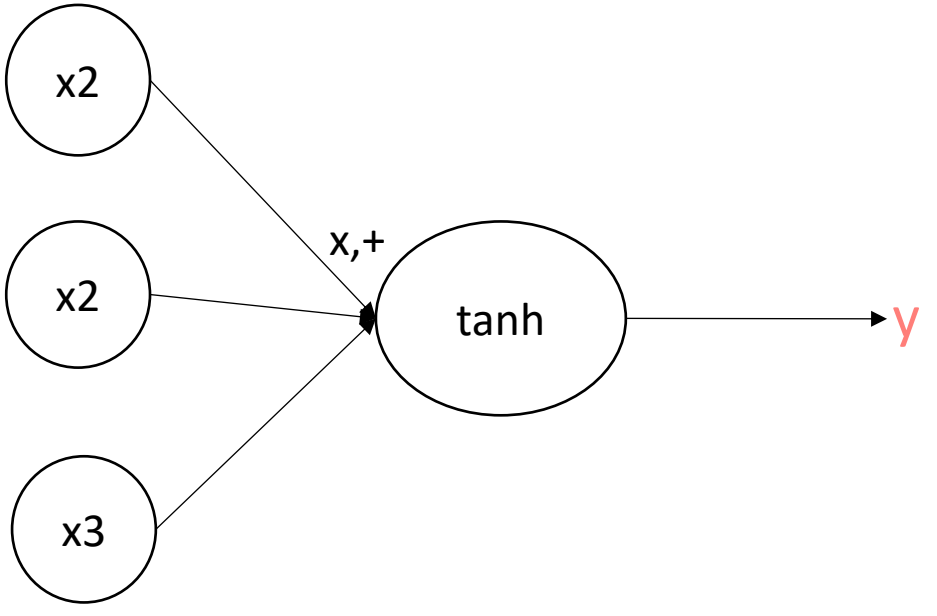


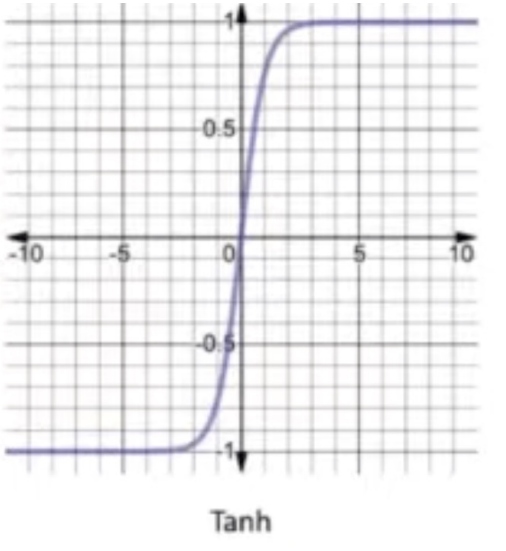
y=2(2 x1 +4 x2)+10



Computational graphs:

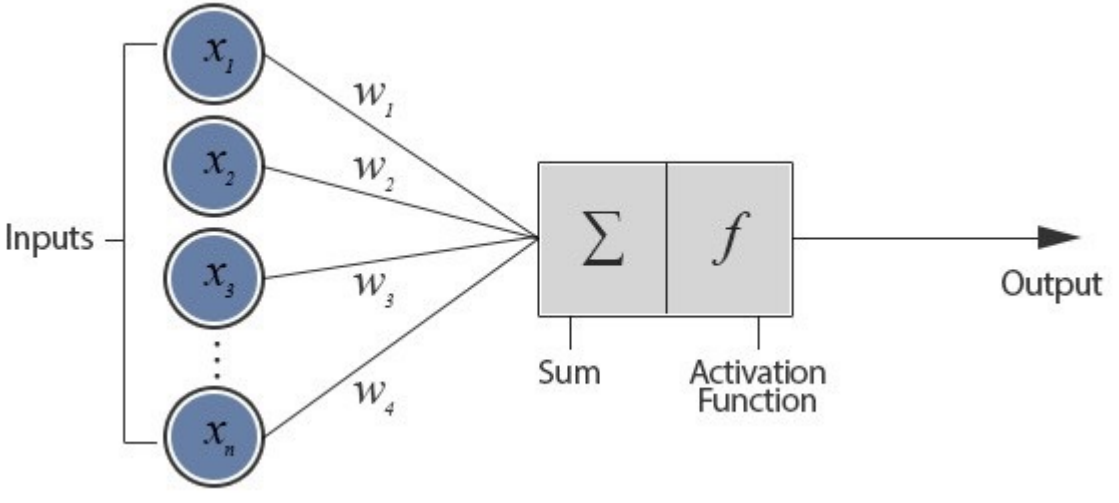
$$y=\tanh(2\ x1+2\ x2+5\ x3+20)$$

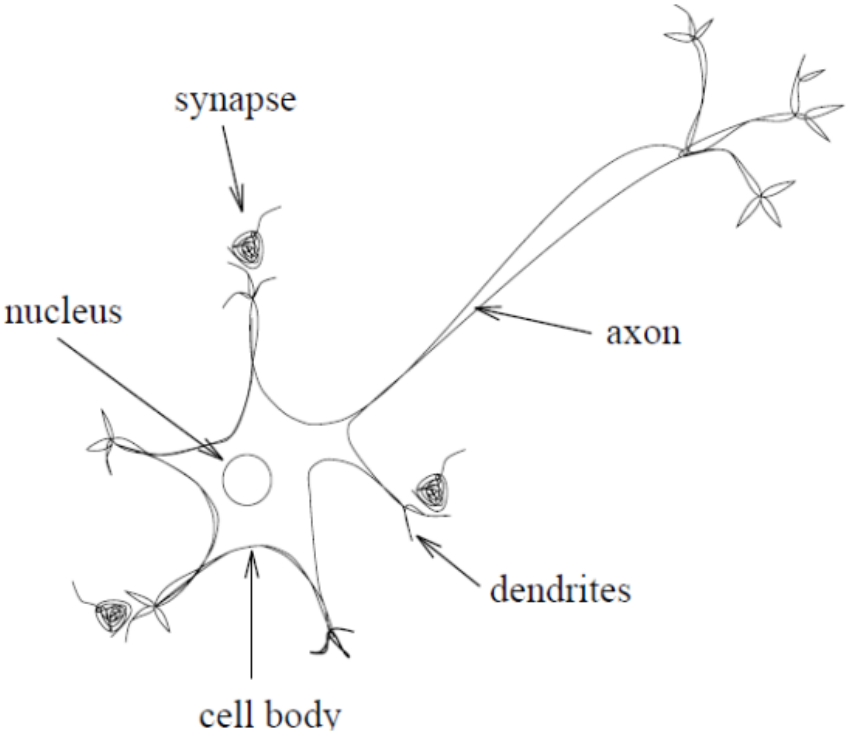




Tanh

THE PERCEPTRON:

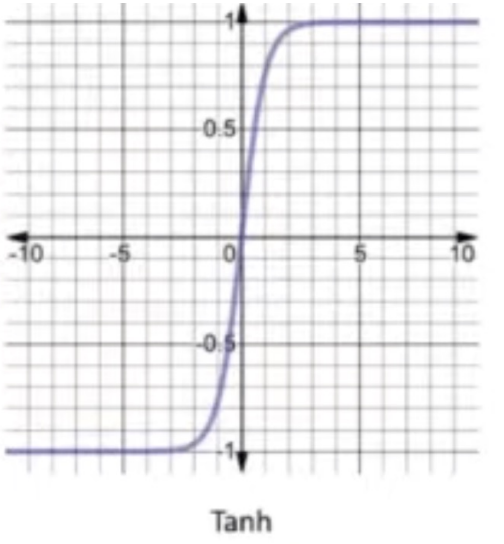




- Perceptrons were developed in the 1950s-60s by the Frank Rosenblatt,
- As you can see, the network of nodes sends signals in one direction. This is called **a feed-forward network**.
- The figure depicts a neuron connected with *n* other neurons and thus receives *n* inputs (x_1, x_2, ….. x_n). This configuration is called a **Perceptron**.

example:

$$y=\tanh(1\ x1+2\ x2+5\ x3+2)$$



Tanh

We have a model of a neuron as above. If for this neuron the inputs are:
x1=-1
x2=-1
X3=1
This neuron will be:

(A)active (>0)
(B)not active (0<)

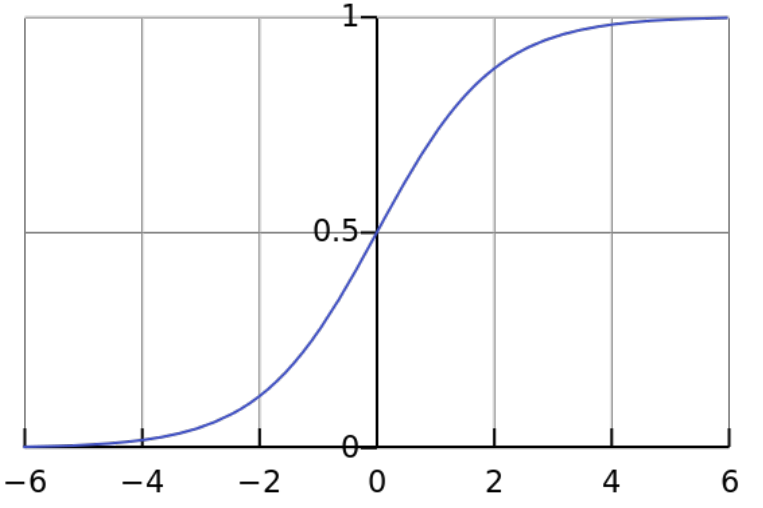
ACTIVATION FUNCTION

- ***Activation*** *functions* are really important for a Artificial Neural Network to learn and make sense of something really complicated and
- Their purpose is to convert input signals of a node in a to an output signal.

TYPES OF ACTIVATION FUNCTIONS

Sigmoid function

$$A = \frac{1}{1+e^{-x}}$$



This looks smooth and “step function like”. What are the benefits of this? It is nonlinear in nature. Combinations of this function are also nonlinear! Great. Now we can stack layers. What about non binary activations? Yes, that too! It will give an analog activation unlike step function. It has a smooth gradient too.

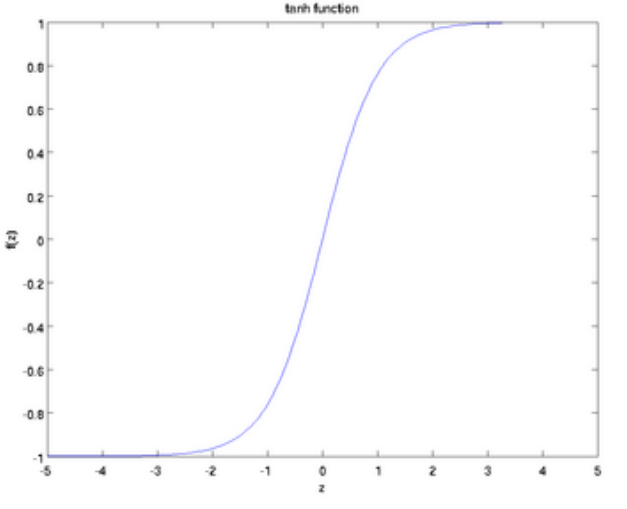
And if you notice, between X values -2 to 2, Y values are very steep. Which means, any small changes in the values of X in that region will cause values of Y to change significantly. That means this function has a tendency to bring the Y values to either end of the curve.

TYPES OF ACTIVATION FUNCTIONS

Tanh Function

- Another activation function that is used is the tanh function.

$$f(x) = tanh(x) = \frac{2}{1+e^{-2x}} - 1$$



This looks very similar to sigmoid. In fact, it is a scaled sigmoid function!

$$tanh(x) = 2\,sigmoid(2x) - 1$$

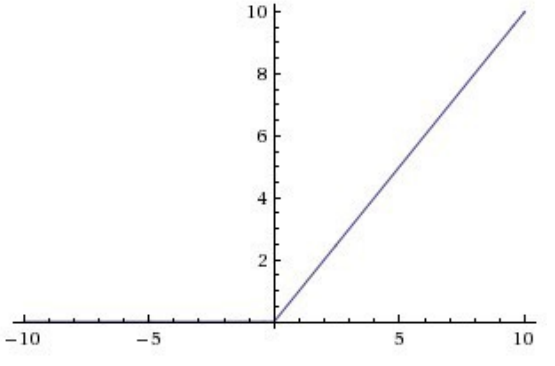
TYPES OF ACTIVATION FUNCTIONS

ReLu

- Later, comes the ReLu function,

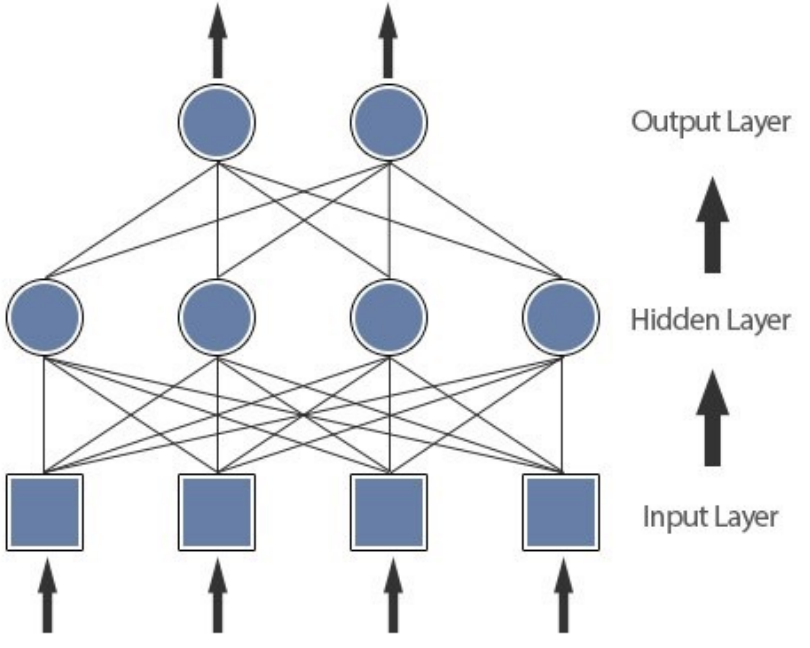
$$A(x) = \max(0,x)$$

The ReLu function is as shown above. It gives an output x if x is positive and 0 otherwise.



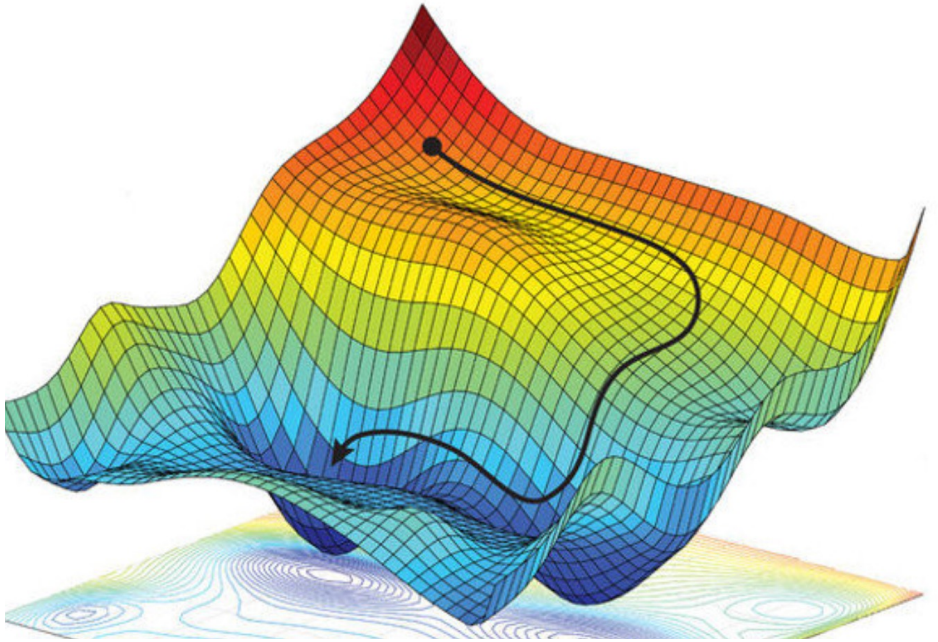
MULTI-LAYERED NEURAL NETWORKS

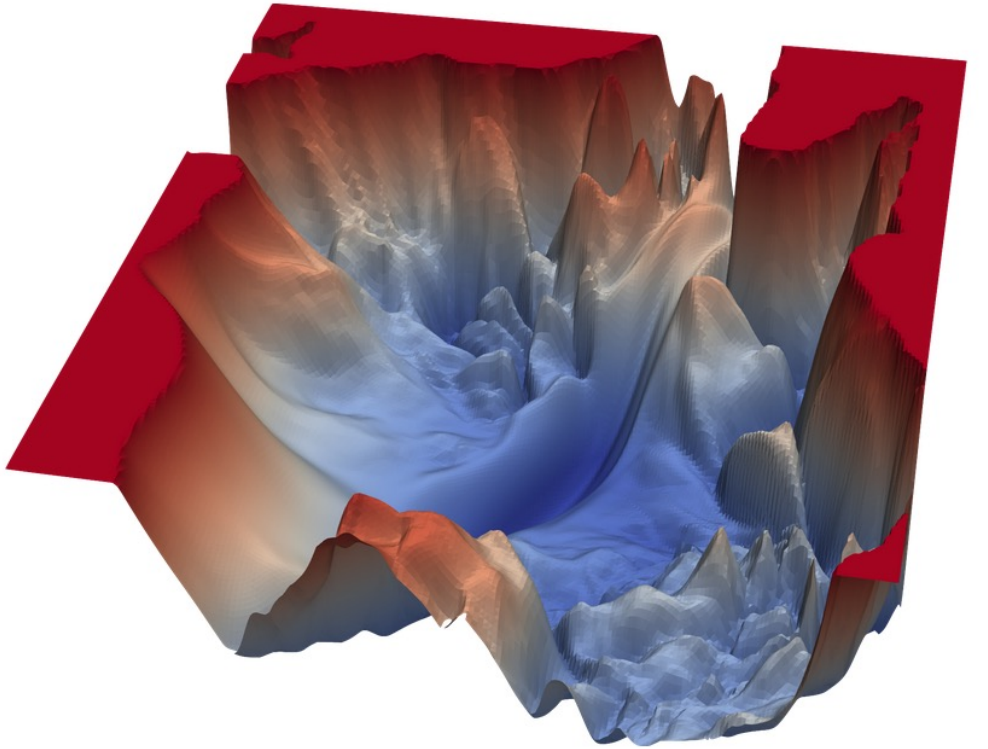
- This process is repeated by feeding the whole training set several times until the network responds with a correct output for all the samples. The training is possible only for inputs that are linearly separable. This is where multi-layered neural networks come into picture.



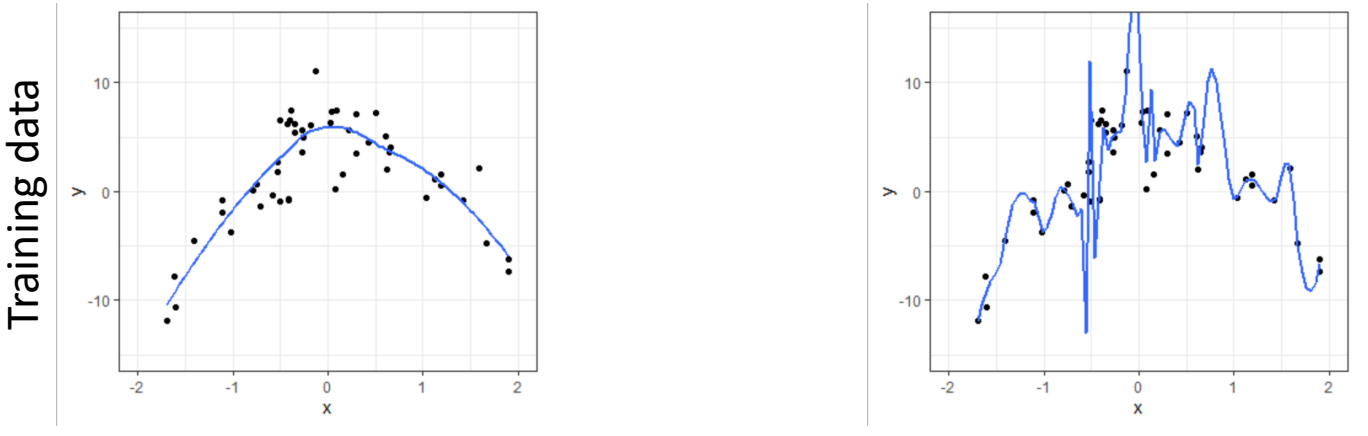
BACK PROPAGATION (BACKWARD PROPAGATION OF ERRORS)

- Backpropagation is a common method for training a neural network.
- The goal of backpropagation is to optimize the weights so that the neural network can learn how to correctly map arbitrary inputs to outputs.
- To tune the weights between the hidden layer and the input layer, we need to know the error at the hidden layer, but we know the error only at the output layer (*We know the correct output from the training sample and we also know the output predicted by the network.).*
- So, the method that was suggested was to take the errors at the output layer and proportionally propagate them backwards to the hidden layer.





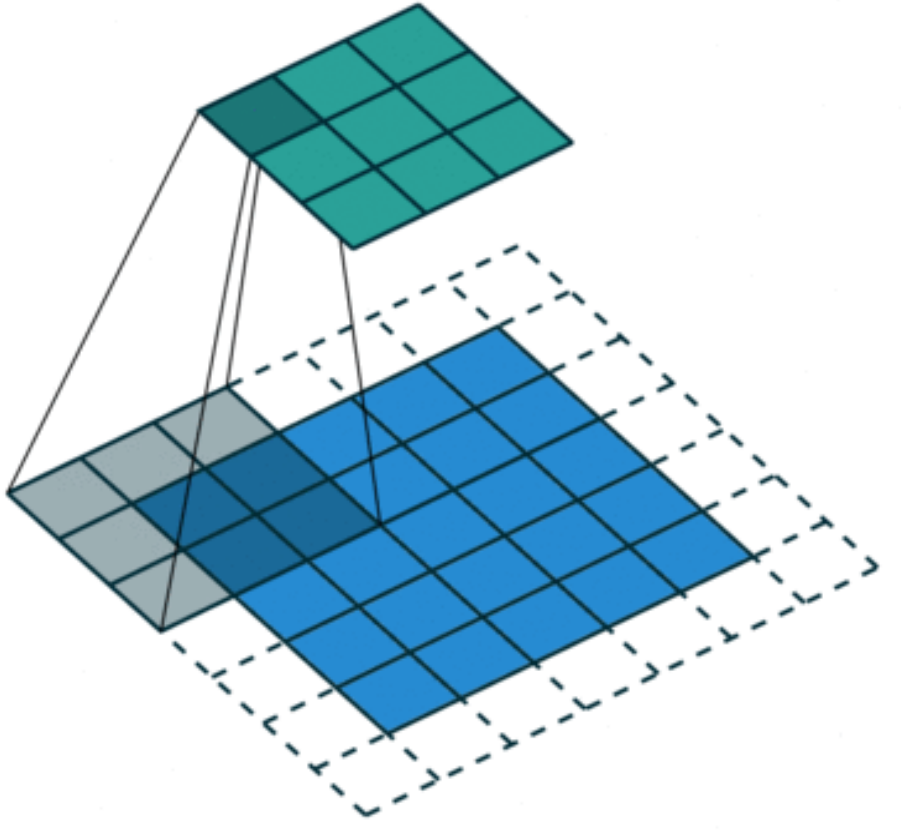
Challenge: Overfitting



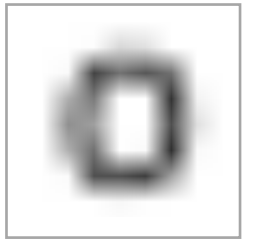
- **Cross-validation**: Split your data into multiple subsets for training and testing
- **Regularization**: Techniques like L1 and L2 regularization penalize large weights in the network
- **Dropout**: Randomly drop a proportion of neurons during training, forcing the network to learn redundant representations and reducing its reliance on specific neurons
- …

Convolutions:

Spatial structure, local computation, shared parameters



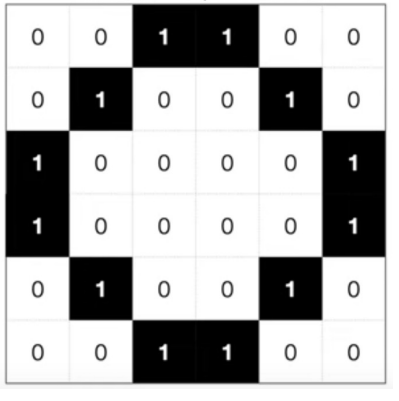
Convolutional NN, example:



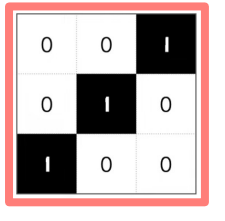
or

Steps:

Convert into pixel values



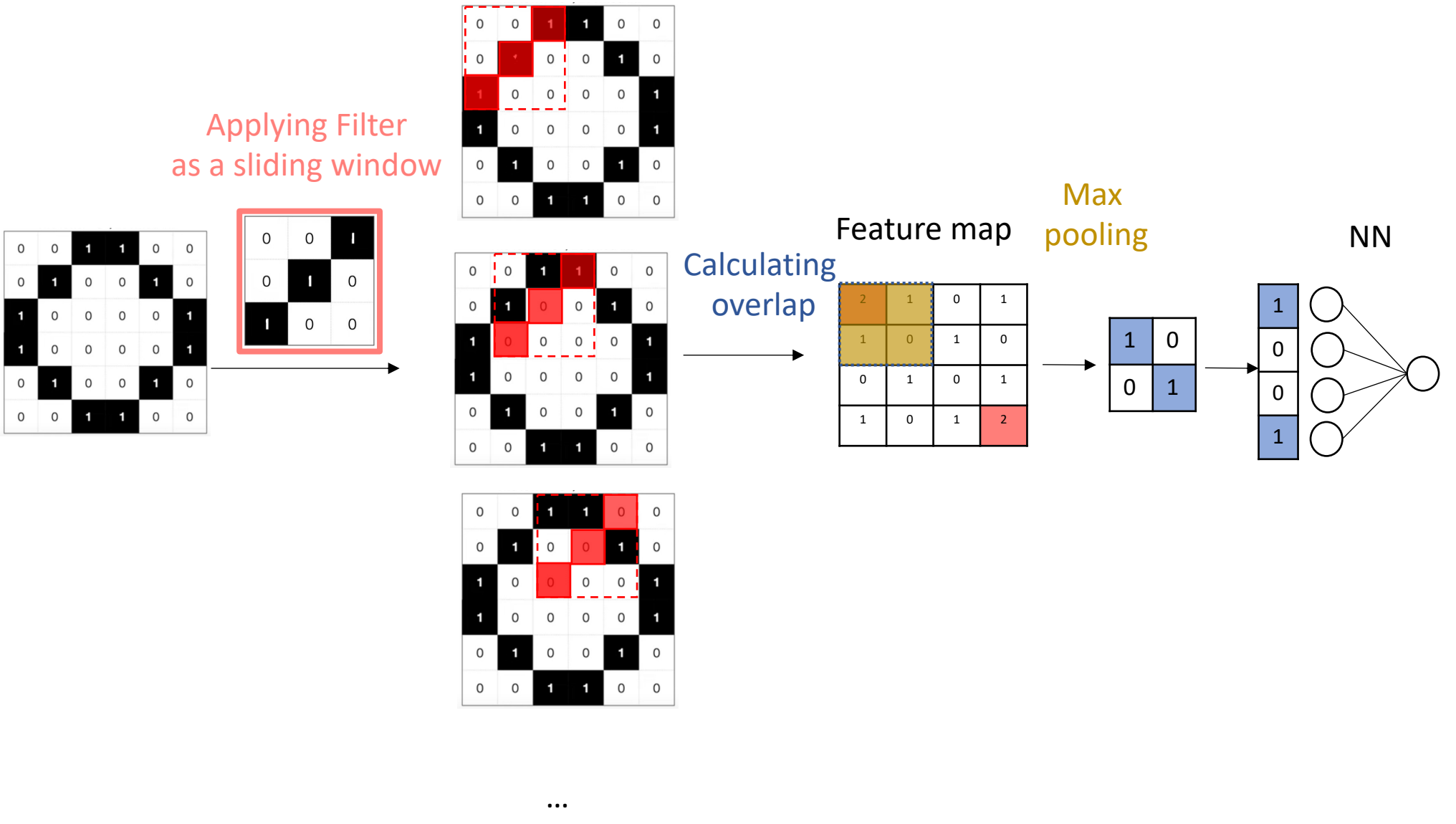
Apply Filter



Pooling

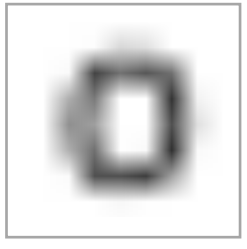
NN

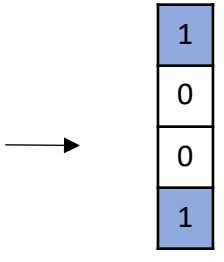
Convolutional NN, example:

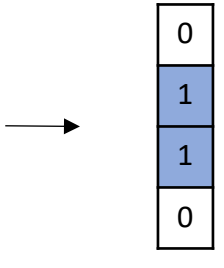


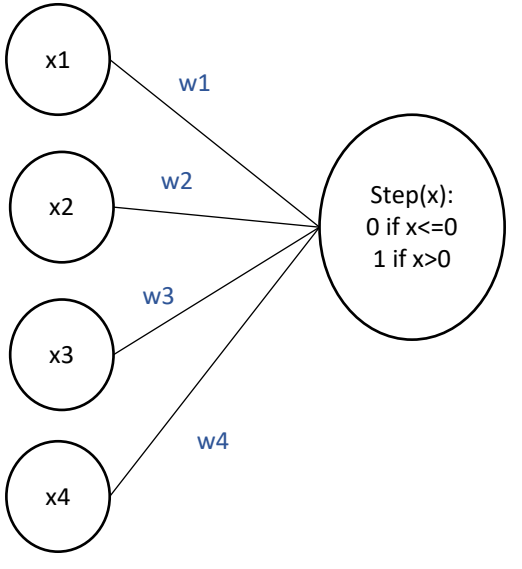
Quiz:

Given this feature map, and the given activation function, which set of weights for the NN correctly predicts the letter ‘O’?



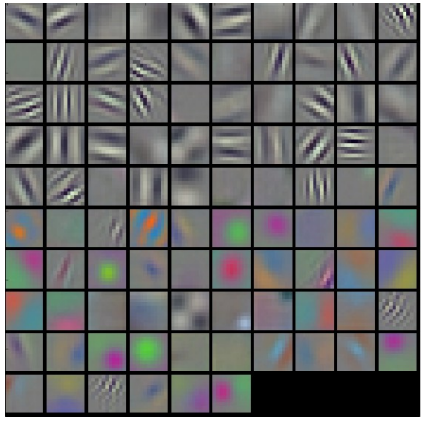


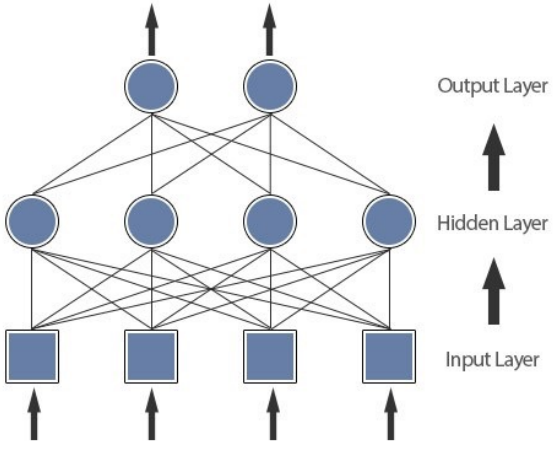


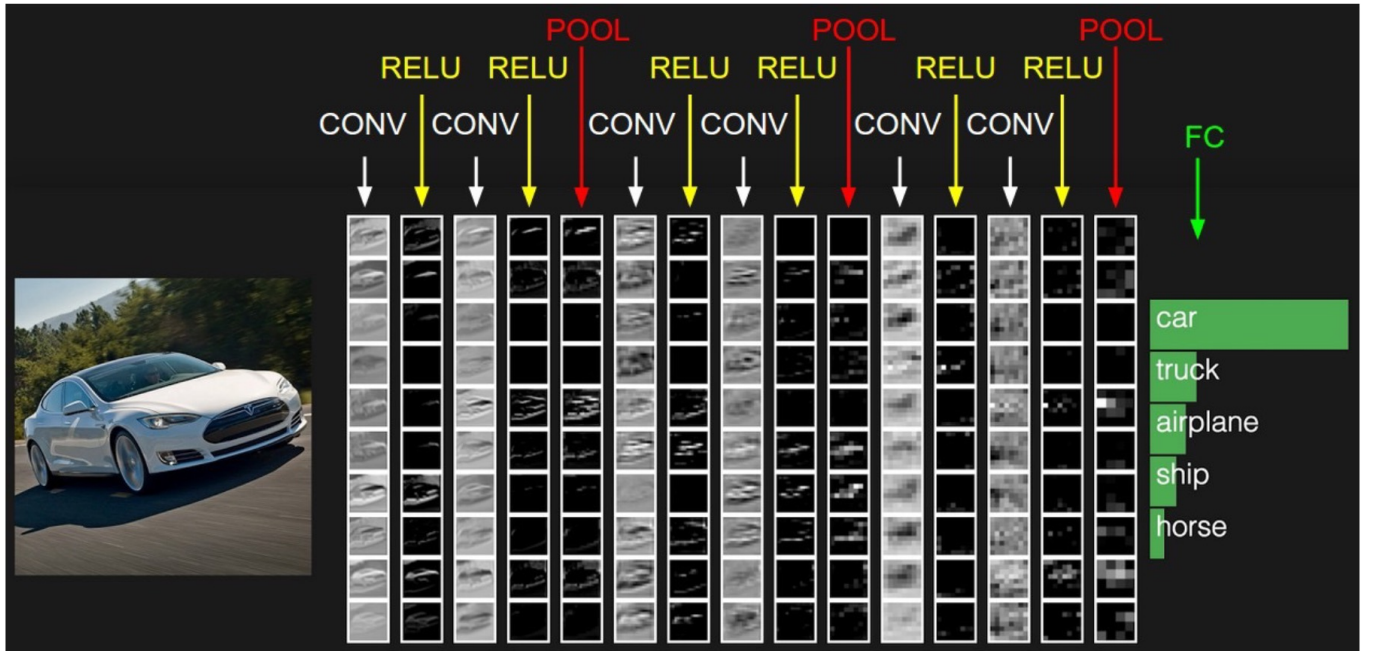


A) w1=0, w2=-1, w3=-1, w4=0
B) w1=0, w2=1, w3=-1, w4=0
C) w1=1, w2=-1, w3=-1, w4=1
D) w1=1, w2=1, w3=0, w4=0

ALEXNET




Output Layer
Hidden Layer
Input Layer


POOL
POOL
POOL
RELU
RELU
RELU
RELU
RELU
RELU
CONV
CONV
CONV
CONV
CONV
CONV
FC
car
truck
airplane
ship
horse

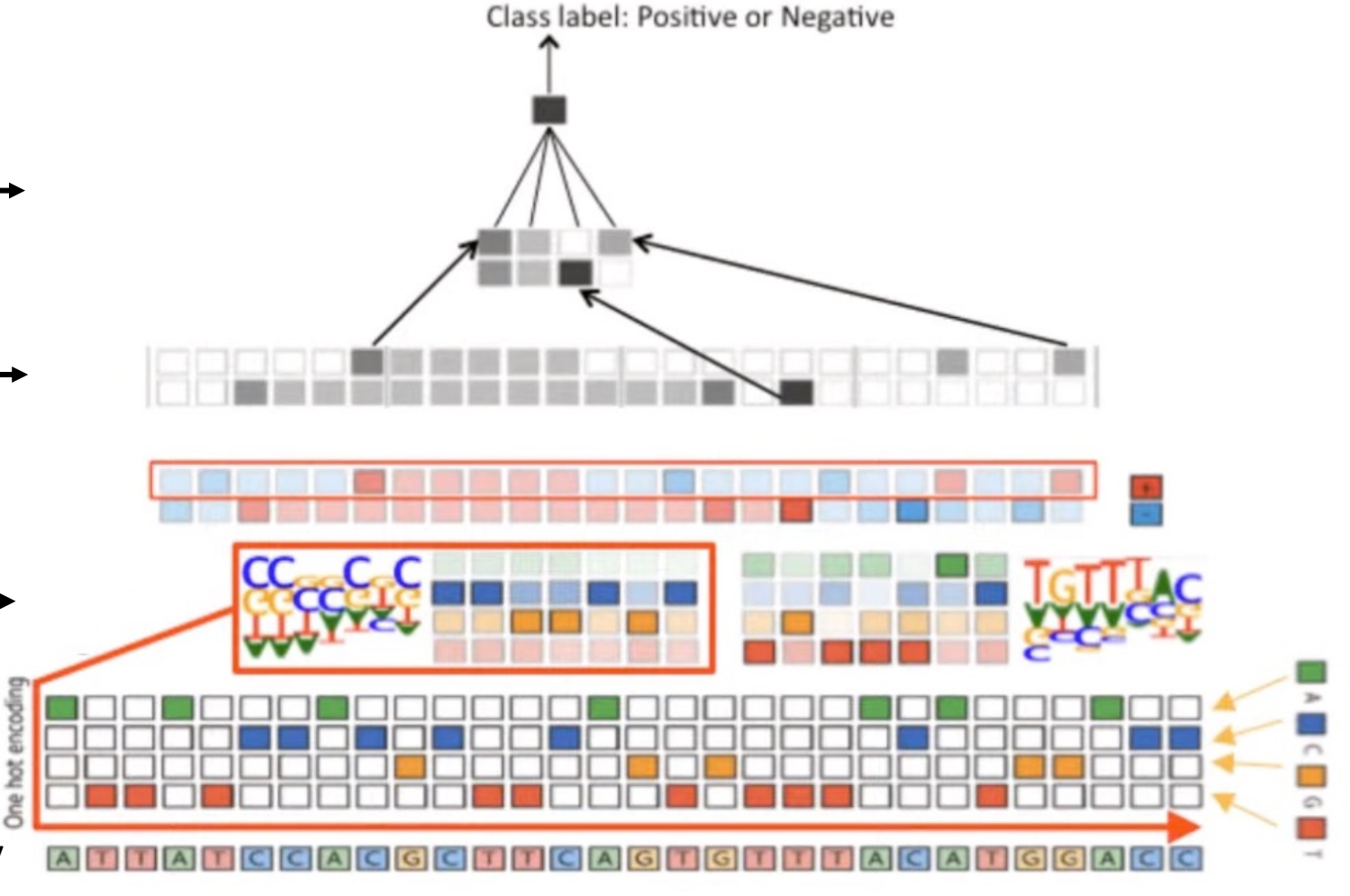
CNN in genomics (DragoNN):

DragoNN is a toolkit to teach and learn about deep learning for genomics

NN →

pooling →

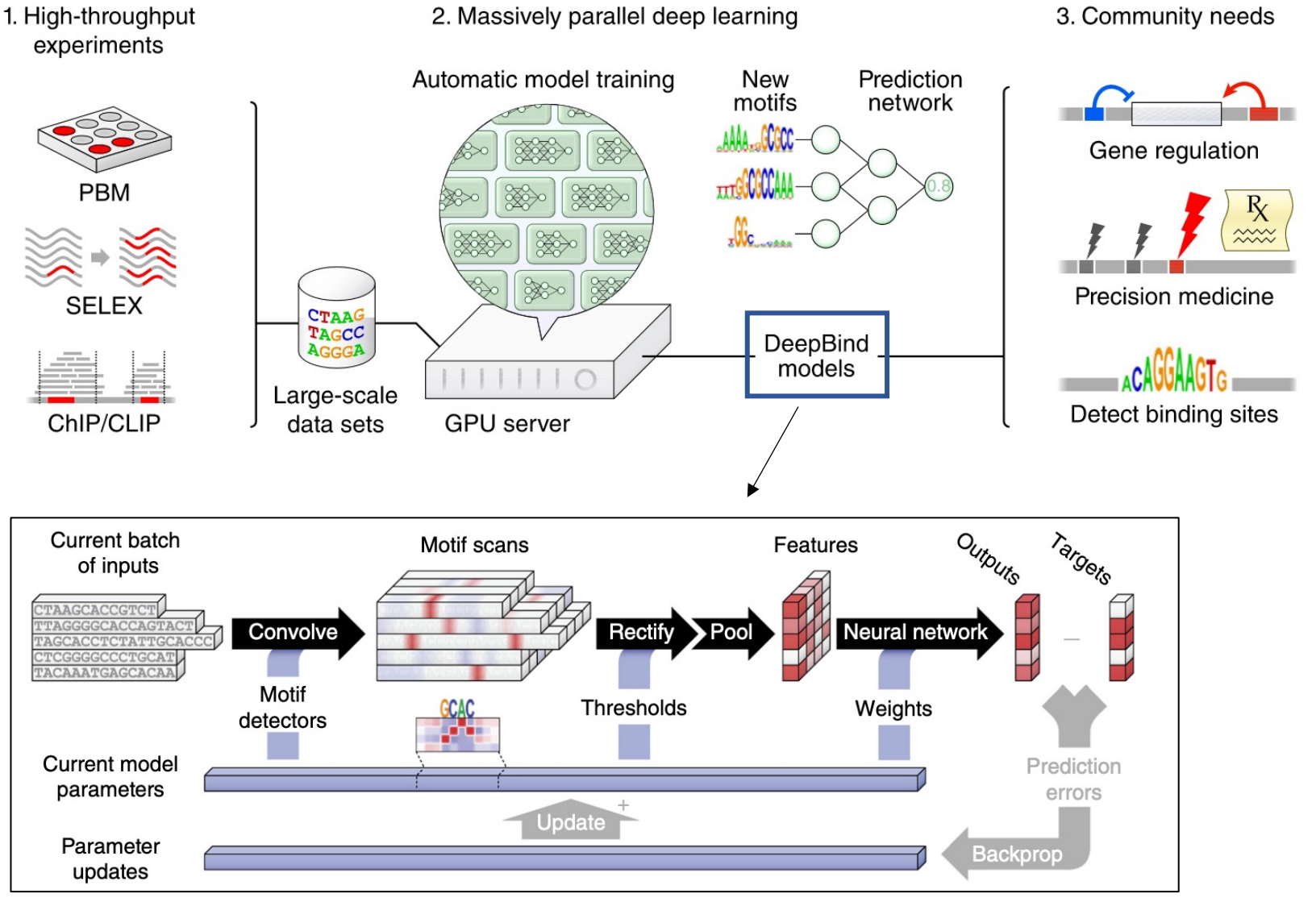
Filter →



https://kundajelab.github.io/dragonn/

CNN in genomics (Deep bind):

Predicting the sequence specificities of dna and rna binding proteins by deep learning

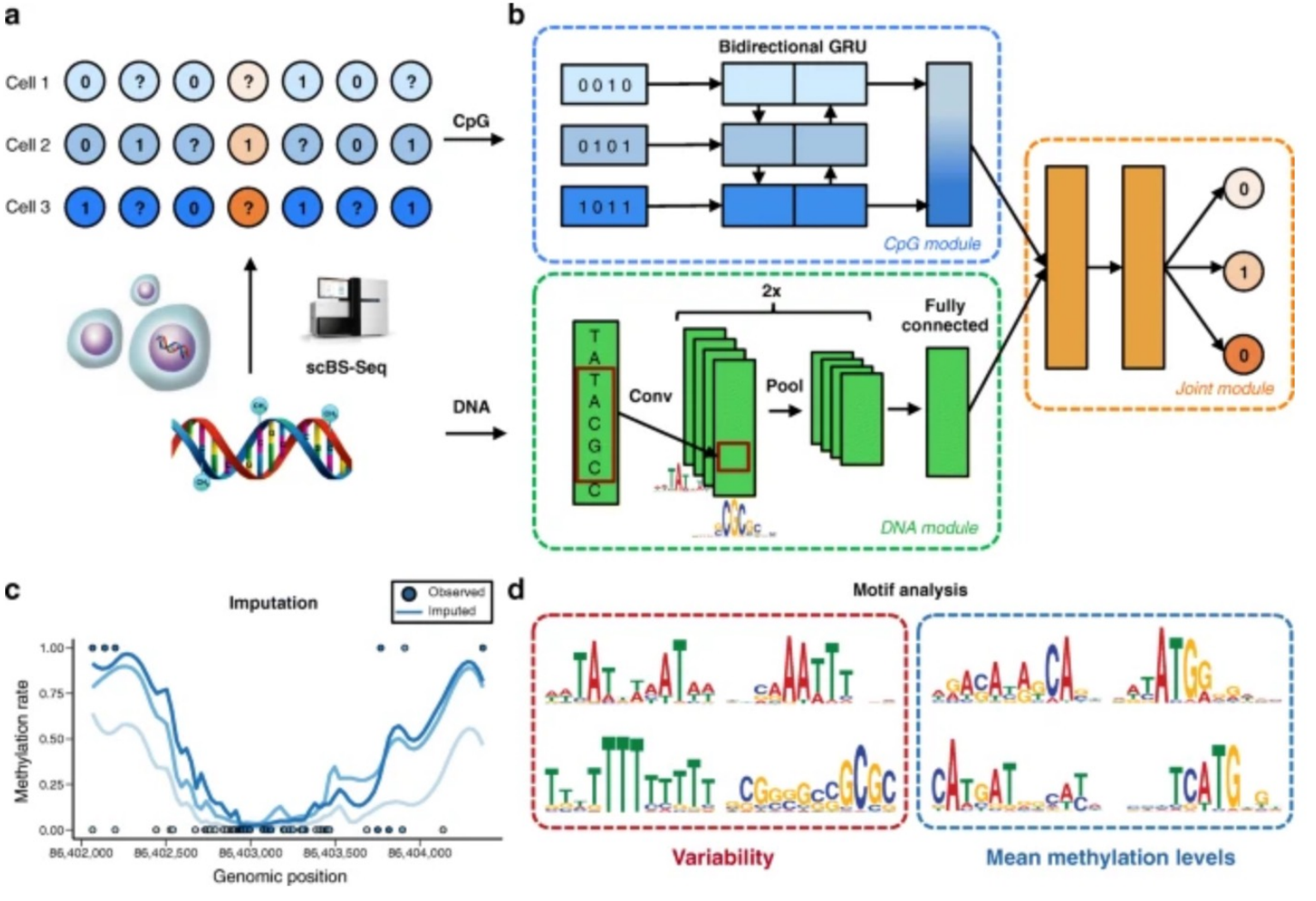


Alipanahi et. Al. 2015

CNN in genomics (Deep CpG):

prediction of single-cell DNA methylation states using deep learning

It is a fusion model (CNN + RNN)



Angermueller, Christof, et al

Unsupervised learning:

- Clustering
- Dimensionality reduction
- ….

clustering algorithms

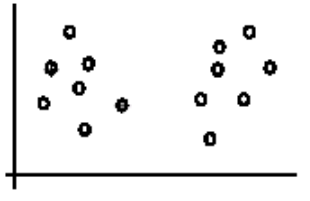
- Partitional clustering
- Hierarchical clustering
- …

distance (similarity, or dissimilarity) function

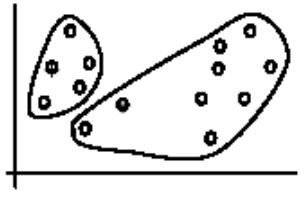
- Clustering quality
  - Inter-clusters distance ⇒ maximized
  - Intra-clusters distance ⇒ minimized
- The quality of a clustering result depends on the algorithm, the distance function, and the application.

K-means clustering:

K-means is a partitional clustering algorithm



(A). Random selection of k centers



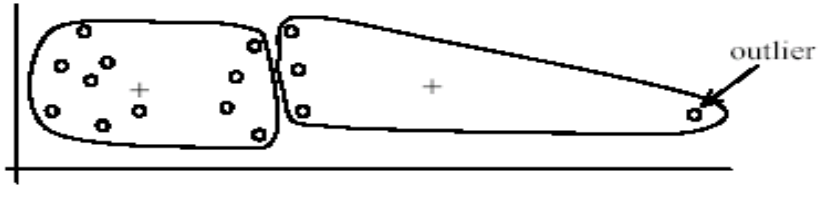
Iteration 1: (B). Cluster assignment

(C). Re-compute centroids

minimizing the **sum of squared error** (SSE)

$$SSE = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} dist(\mathbf{x}, \mathbf{m}_j)^2$$

Weaknesses of k-means: Problems with outliers



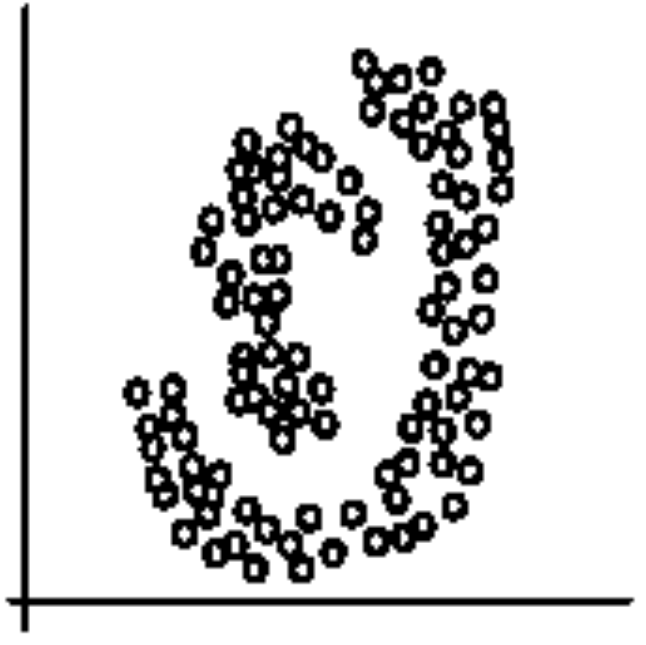
(A): Undesirable clusters

(B): Ideal clusters

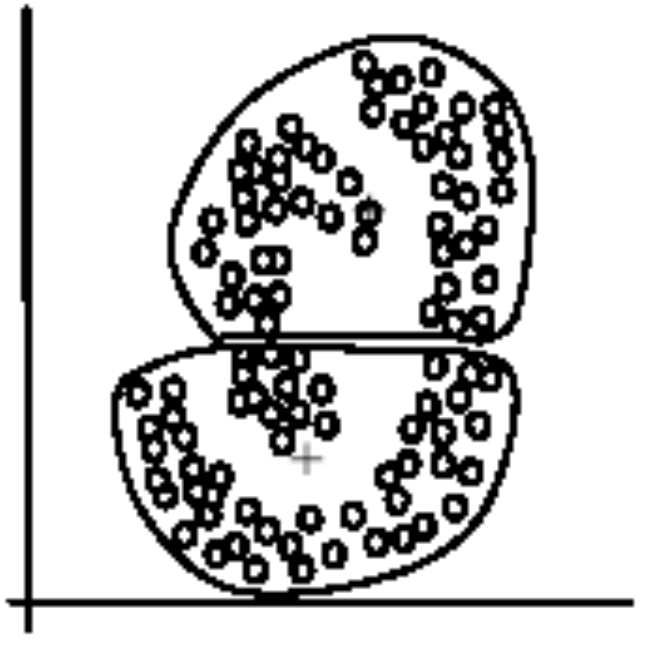
Weaknesses of k-means: To deal with outliers

- One method is to remove some data points in the clustering process that are much further away from the centroids than other data points.
  - To be safe, we may want to monitor these possible outliers over a few iterations and then decide to remove them.
- Another method is to perform random sampling. Since in sampling we only choose a small subset of the data points, the chance of selecting an outlier is very small.
  - Assign the rest of the data points to the clusters by distance or similarity comparison, or classification

- The algorithm is sensitive to initial seeds.
- The *k*-means algorithm is not suitable for discovering clusters that are not hyper-ellipsoids (or hyper-spheres).

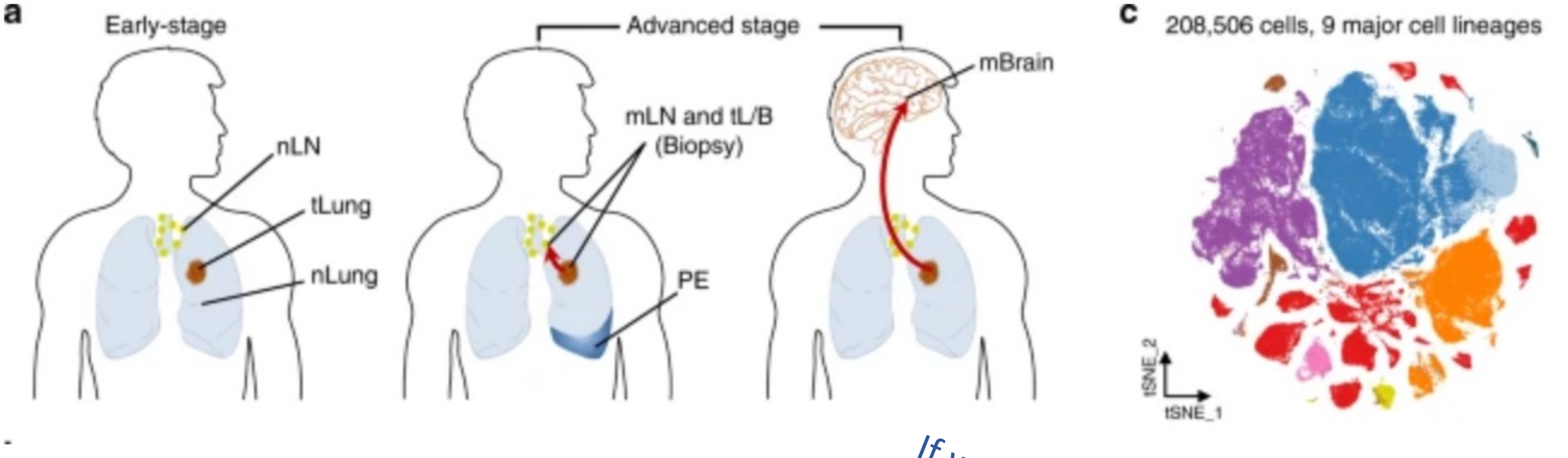


(A): Two natural clusters



(B): *k*-means clusters

Example usage of clustering in cancer research:



If we are measuring 200k cells and 25k genes, how we can visualize it in a 2D plot?

Nayoung, et al. 2020

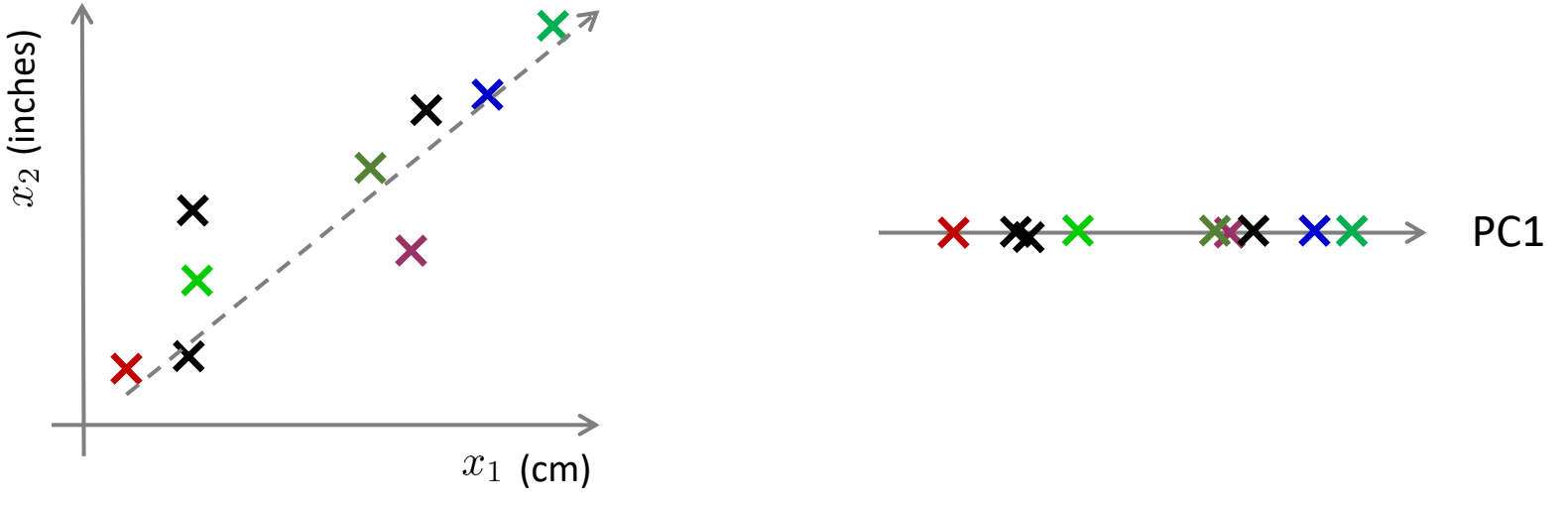
Unsupervised learning:

- Clustering
- Dimensionality reduction
- ….

Motivation:

- A way to simplify complex high-dimensional data
- Summarize data with a lower dimensional real valued vector

Example: Reduce data from 2D to 1D



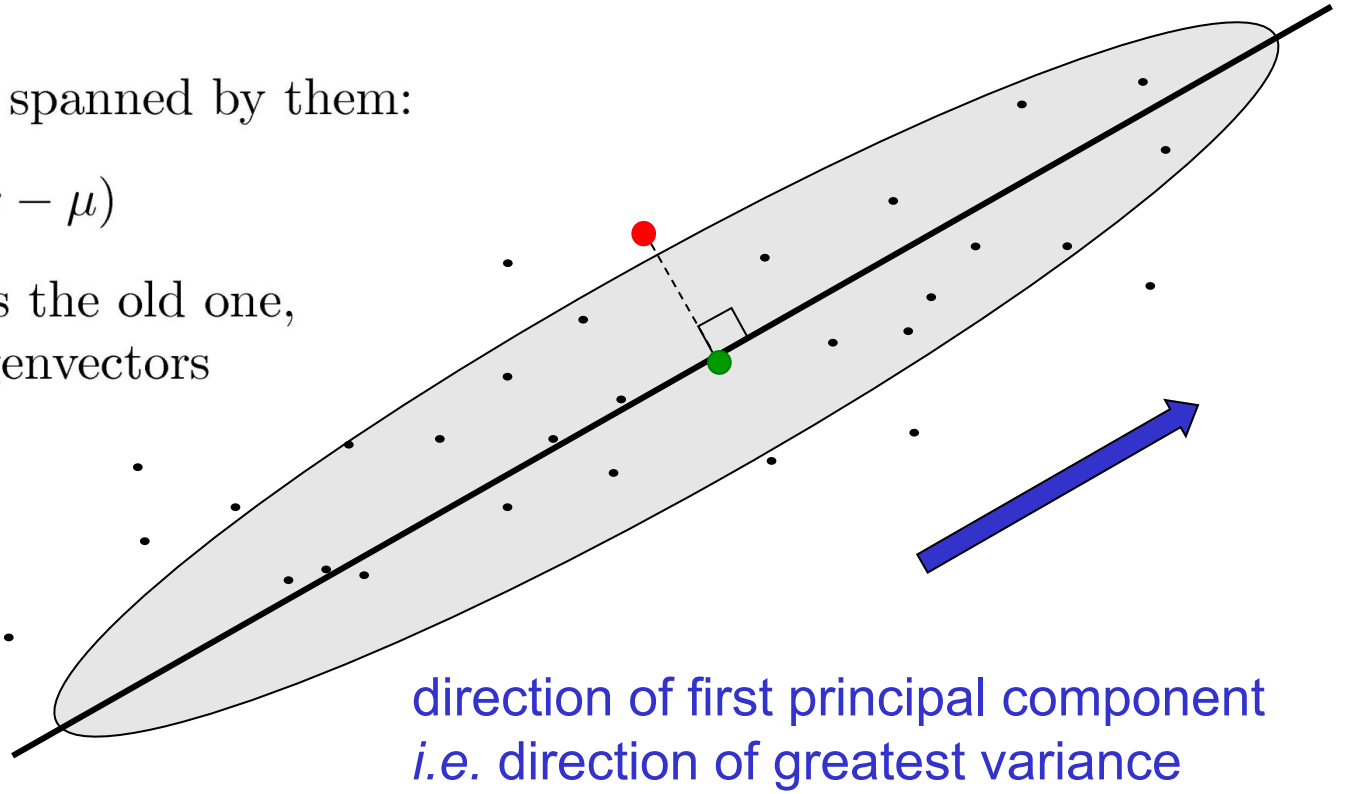
Principal Component Analysis

Goal: Find r-dim projection that best preserves variance

1. Compute mean vector μ and covariance matrix Σ of original points
2. Compute eigenvectors and eigenvalues of Σ
3. Select top r eigenvectors
4. Project points onto subspace spanned by them:

$$y = A(x - \mu)$$

where y is the new point, x is the old one, and the rows of A are the eigenvectors



direction of first principal component
i.e. direction of greatest variance

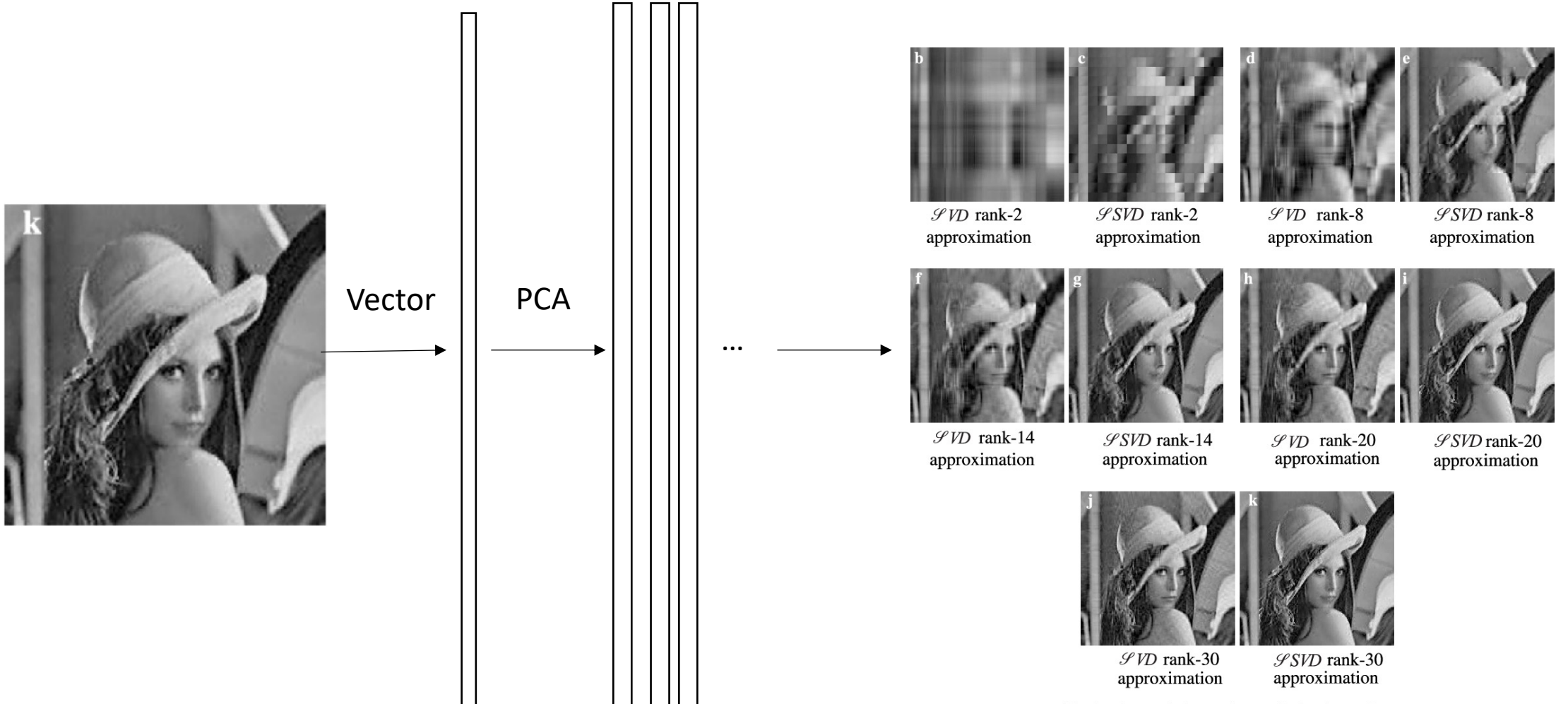


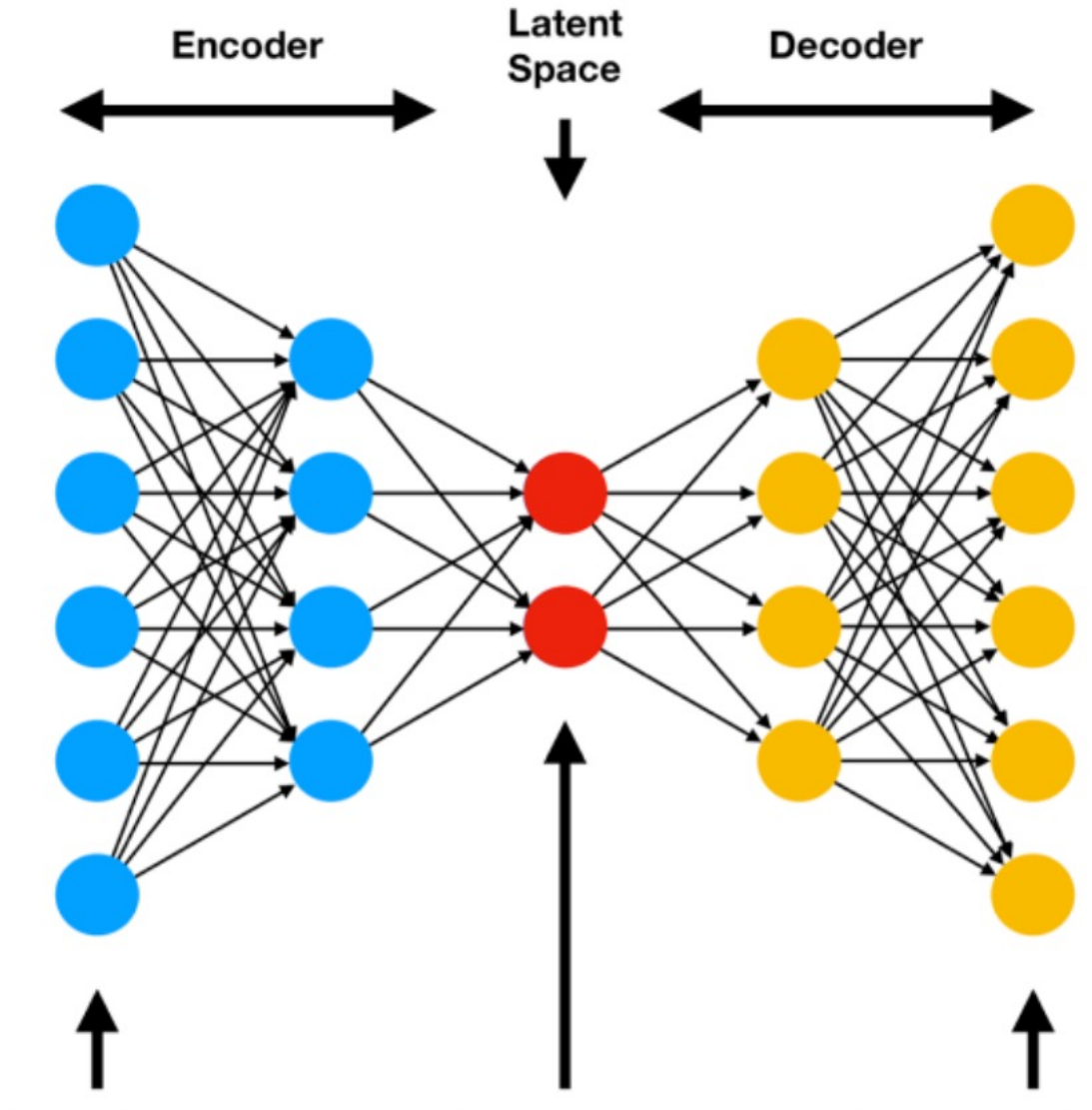
Fig. 1. $\mathcal{SVD}$ and $\mathcal{SSVD}$ when applied on image Lena.

Compare:

- PCA/SVD
  - PCA takes a collection of vectors (images) and produces a usually smaller set of vectors that can be used to approximate the input vectors via linear combination.
  - Very efficient for certain applications.
  - Fourier and wavelet compression is similar.

- **Neural network autoencoders**
  - Can learn nonlinear dependencies
  - Can use convolutional layers

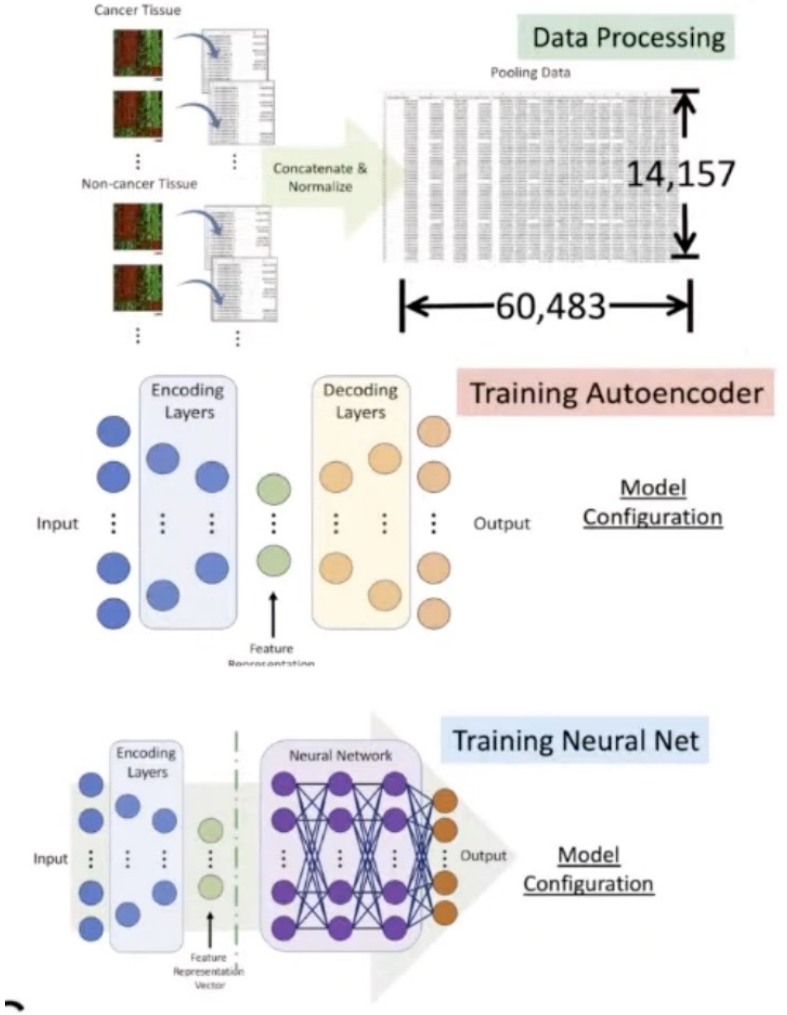
Neural network autoencoders

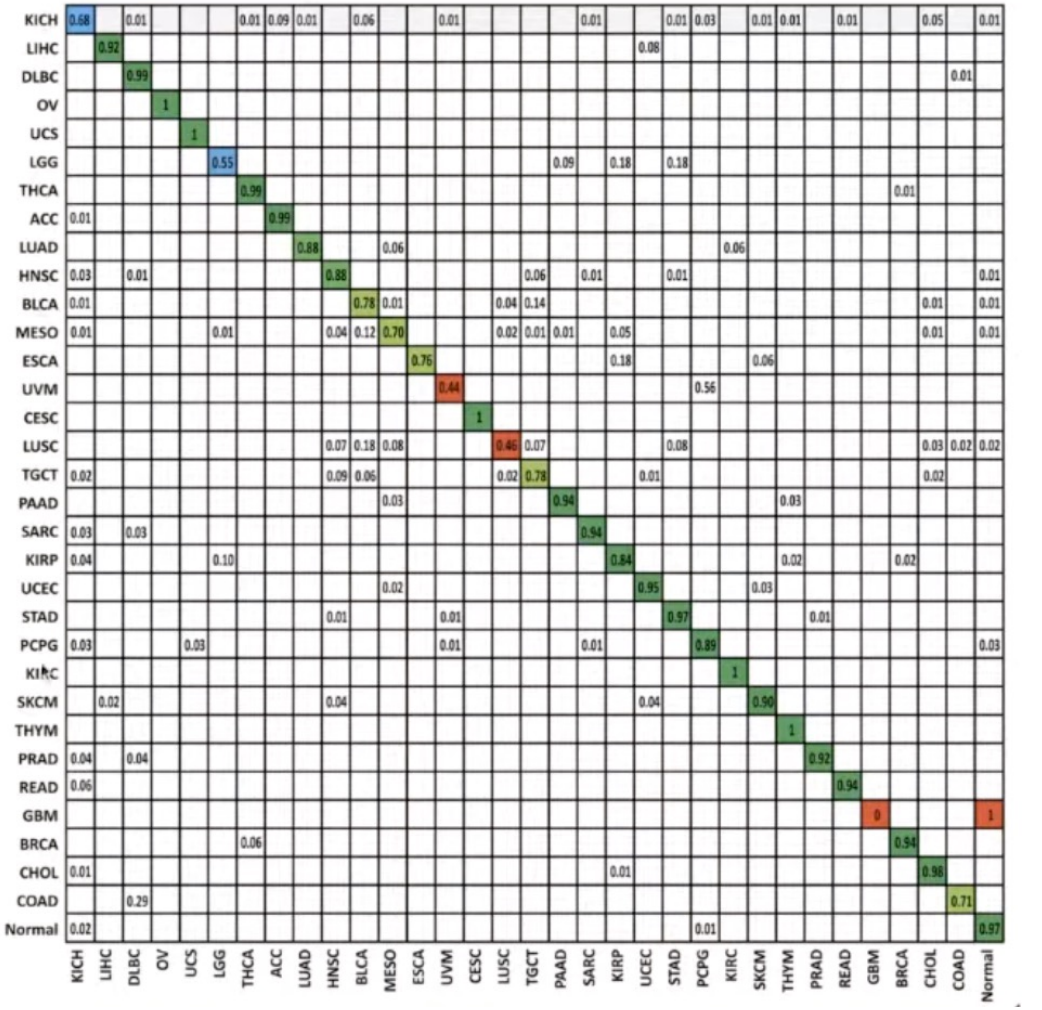


An auto encoder with linear transfer function is nearly equivalent to PCA:

Plaut, Elad. "From principal subspaces to principal components with linear autoencoders." *arXiv preprint arXiv:1804.10253* (2018).

Example: Facilitate Cancer Detection and Type Classification from Gene Expression





chen et al 2019

Tiam.heydari@ubc.ca