

# Introduction To Artificial Intelligence For Drug Development & Medical Science

Mark Chang, PhD

FDA-Industry Workshop, Washington DC, September, 2020

# Topics to Cover

- AI/Machine Learning Overview
- Supervised Learning
  - Deep Learning Neural Networks
  - Similarity-Based Learning/Kernel Method/Support Vector Machine
  - Decision Tree Methods
- Non-Supervised Learning
  - Unsupervised Learning
  - Reinforcement Learning
  - Swarm Intelligent Learning
  - Evolutionary Learning

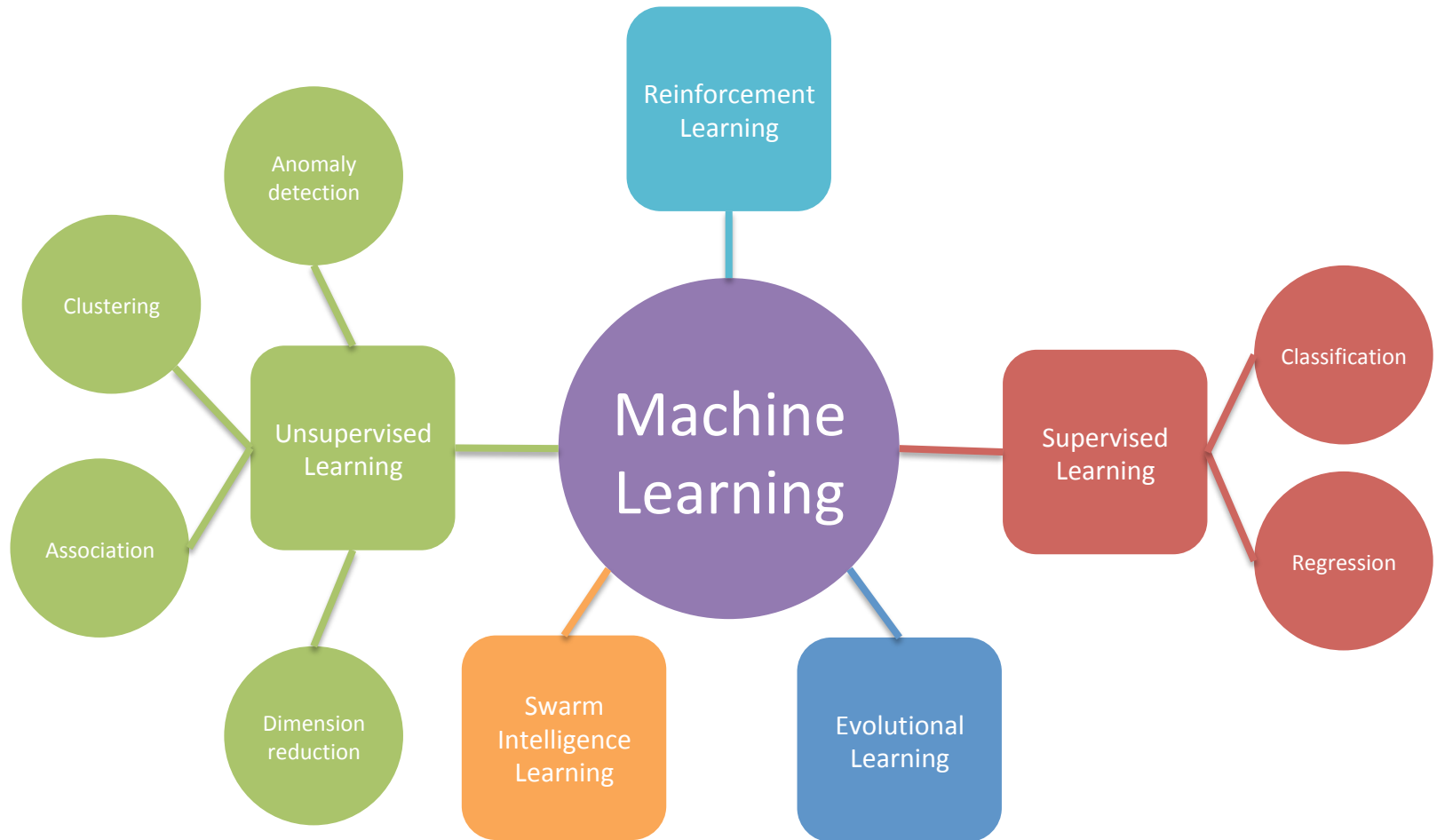
# Artificial Intelligence (AI)

- **Intelligence** = Ability of Performing Mental Work
- **AI** = Software with the Ability of Performing Human Mental Work
- **Robot** = Machine (hardware) with the Ability Perform Physical and/or Mental Work that Usually Requires AI
- **Examples of AI Technologies and Robotics**
  - Calculator, computer, iphone, ipad,
  - ELIZA, Apps, search engine, Google map, ecommerce system,.. Alpha0
  - Machine-learning, data mining, statistical modeling.
  - Humanoid robots, iRobot, robots in assembly lines, self-drive cars
- **Medicine AI**
  - Quantitative Structure–Activity Relationship (QSAR) and Molecular Design in Drug Discovery
  - Cancer Prediction Using Microarray Data
  - Disease Diagnosis and Prognosis Based on Medical Image
  - Natural Language Processing for Medical Records
  - Similarity-Based AI for Clinical Trials

# Artificial General Intelligence (AGI)

- **AGI Is Machine With Full Scope Of Human Intelligence**
  - Unlike Narrow AI that accomplish specific tasks for human
  - Capability of experiencing consciousness, emotion, discovery, creativity, self-awareness, collaboration, and evolution
  - Maybe be capable of creating robots like himself and human that can give births
- **AGI Is Another Race Of Human Beings**
  - As medical AI develops, we gradually replace our malfunctioned organs with ones (artificial or not), making everyone a mixture of human and machine.
  - As robots reach full human intelligence and intensively interact with us, we will develop strong feeling with them and will not discriminate them just because of their originalities.
- **Challenges To Overcome**
  - Computing power required - possibly powered by Quantum Computing
  - Physical (muscular) power required
  - Man-machine mixture might be a solution

# Machine Learning = Backbone of Narrow AI



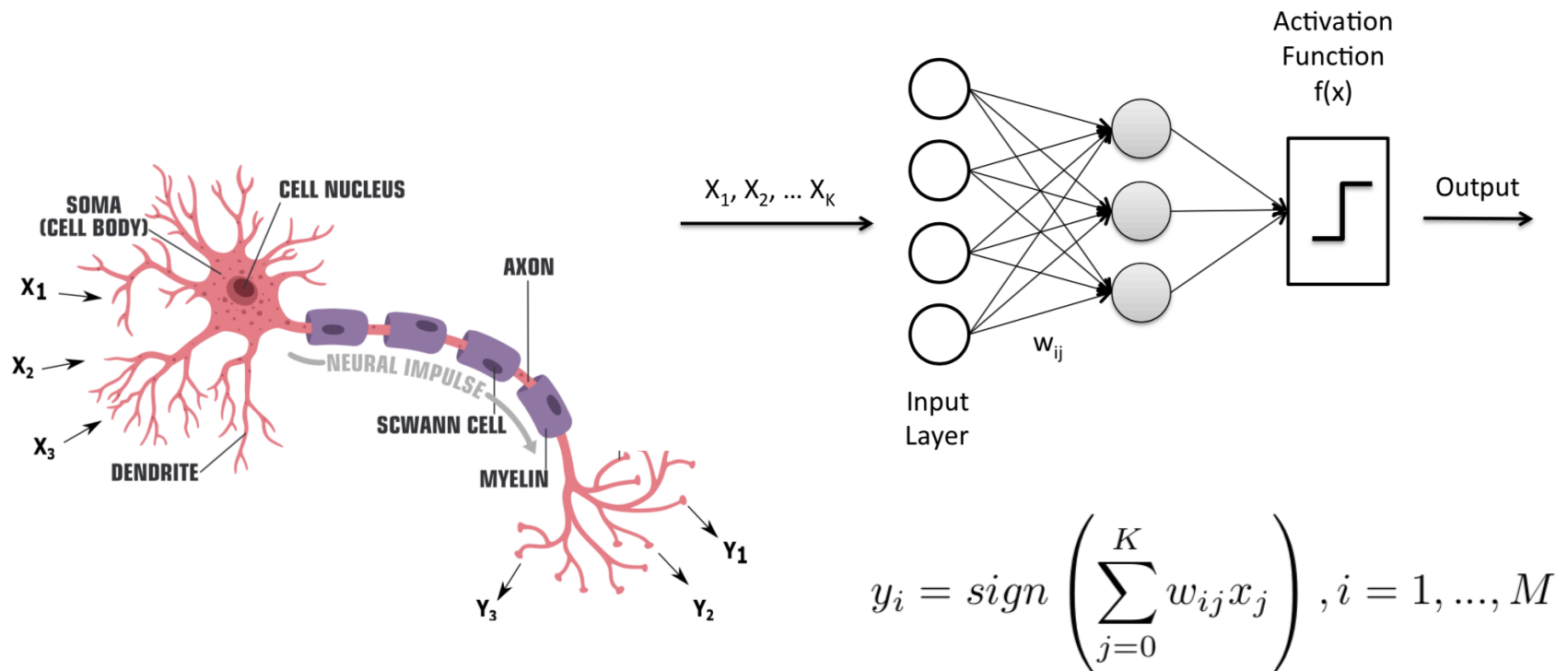
# Types Of Machine Learning

- **Supervised Learning**
  - Learner gives a response  $y$  based on input  $x$ , and compare the target  $y$
  - Trained learner will be able to predict future outcome
  - Ex: Digital imaging recognition for cancer diagnosis
- **Unsupervised Learning**
  - Based on the similarities to re-represent the inputs in a more efficient way
  - Find hidden structure without the help of a target answer
  - Ex: Documentation organization, customer segmentation, patient clustering
- **Reinforcement Learning**
  - Concerns how a learner should take actions in an environment so as to maximize some notion of long-term reward
  - Ex: iRobot, driverless car
- **Evolutionary Intelligence/Learning**
  - Emerges through evolutions over generations of AI agents – become better in some sense
  - Ex: Genetic algorithm for pharmacovigilance and Healthcare management
- **Swarm Intelligence/Learning**
  - Systems operation based on micro (individual) motivations, without a centralized controller or leader
  - Ex: Ant algorithm for optimal customer boarding strategy at airport

# Deep Learning

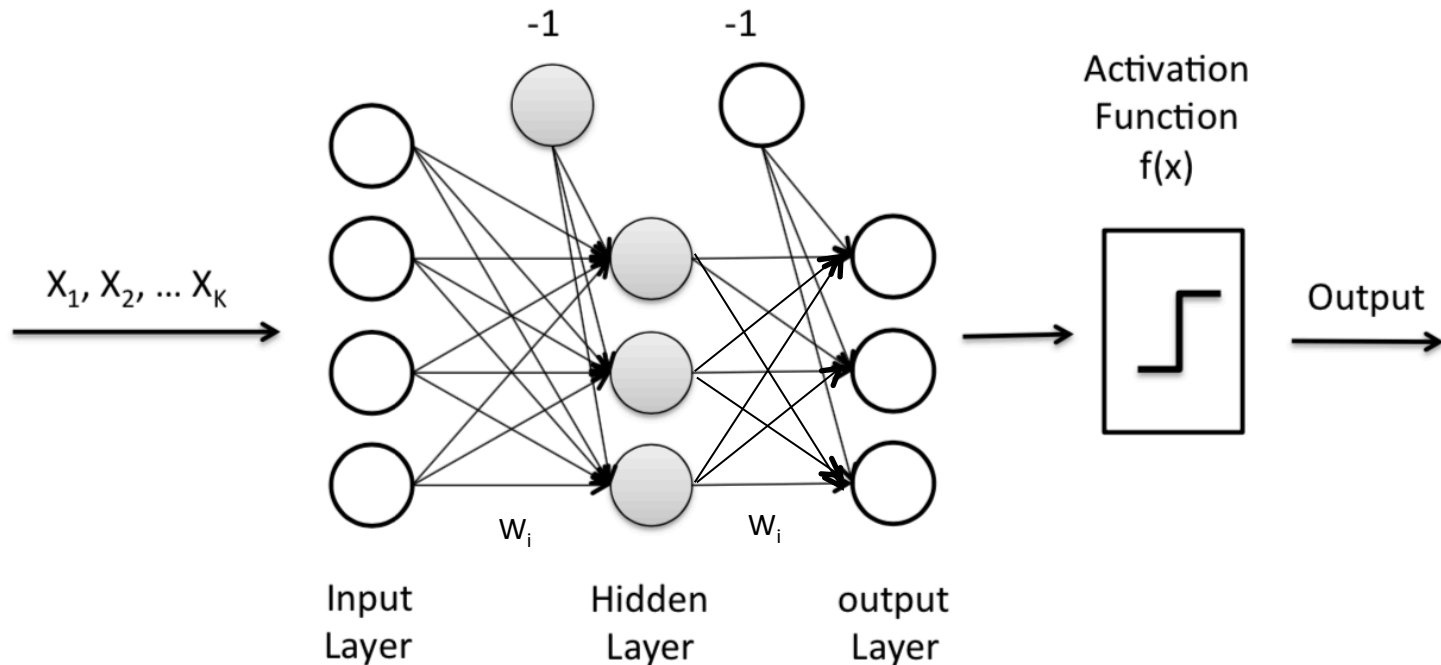
- Deep Learning Neural Network
  - Multiple-layer artificial neural network
  - Progressively extract higher-level features from raw input
- Four Major Network Architectures
  1. Feed-forward Neural Network (FNN) for General Classification and Regression
  2. Convolution Neural Network (CNN) for Image and Facial Recognition
  3. Recurrent Neural Network (RNN) for Speech Recognition, Natural Language Processing
  4. Deep Belief Network (DBN) for Disease Diagnosis and Prognosis.

# Artificial Neural Network (ANN)



Activation Function: identity, softmax, logistic, sigmoidal, tanh, and sgn functions, ...

# Multiple Perceptron (Forward Neural Network)



- With a identity activation function, the ANN = linear function  $X_i$  and linear in  $W_i$
- Classic linear model is linear in parameter  $W_i$  but can nonlinear in  $X_i$
- Learning = updating weight  $W_i$

# Learning - Backpropagation Algorithm

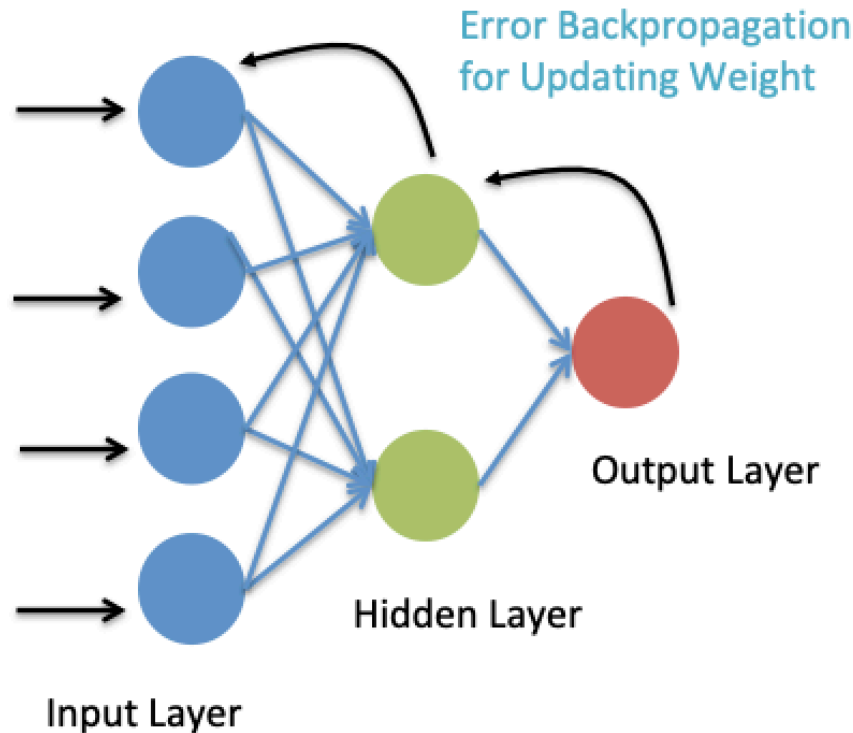
Learning (training) = updating weight  $w_i$  in the network.

Error or loss function:

$$E = \frac{1}{N} \sum_i (\hat{Y}_i - Y_i)^2$$

Predicted (red arrow) Targeted (blue arrow)

$$\hat{Y} = f(w_i, Y_i)$$



$$\Delta w_i = -\alpha \frac{\partial E}{\partial w_i}$$

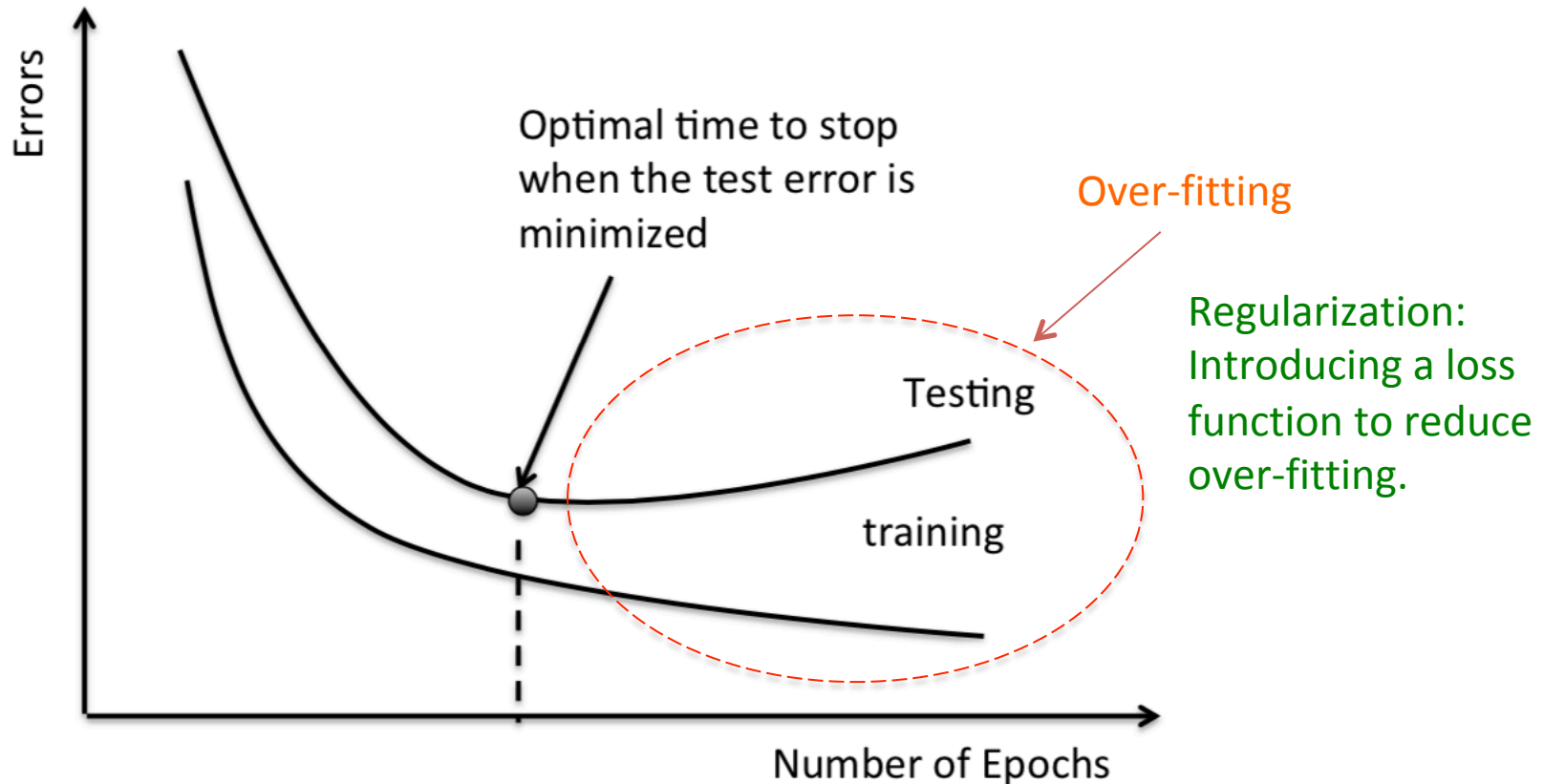
Weight increment (blue arrow)

Update weight  $w_i$  backwards layer by layer starting from the output layer, hidden layers, to the input layer.  
 $\alpha$  = learning rate, a small constant to ensure convergence of the algorithm,

# Training And Testing

- Train AI Model to Determine Its Parameters Before Use
- Test the Trained Model for Its Performance
- Define Target Population
- Obtain Training and Testing datasets
  - Resampling with replacement (bootstrap) and without replacement (split)
  - For larger sample size problems, use resampling without replacement because each sample likely reflects the distribution of the target population.
  - For small sample size problems, use resampling with replacement because each sample often does not reflect the population and sample is too small for resampling without replacement (**Empirical distribution  $\approx$  population distribution**).

# Training Error Versus Testing Error



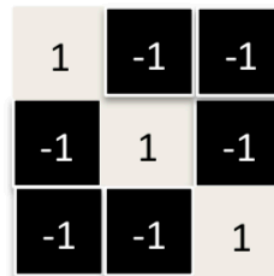
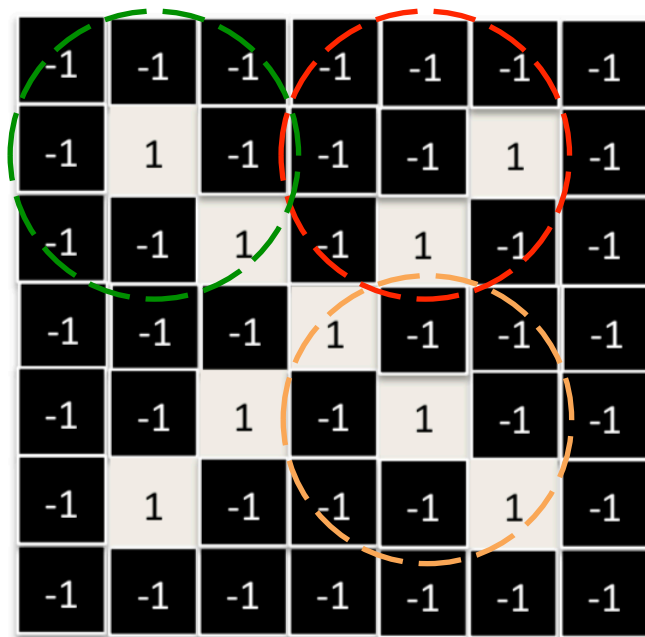
# Convolution Neural Network (CNN)

- Main Ideas In CNNs
  - Different filters in the convolution layers
  - Location invariance and compositionality: make sense for image processing but not for NLP or time-series events
- Filter Functions
  - Each identifies particular features or image elements
  - Overlapped local features to formulate the “overall picture.”
- CNN for Data Other Than Images, If
  - (1) the data can be transformed to look like image data in matrix form, and
  - (2) the data is just as useful after swapping any two columns

# Convolution Illustrated

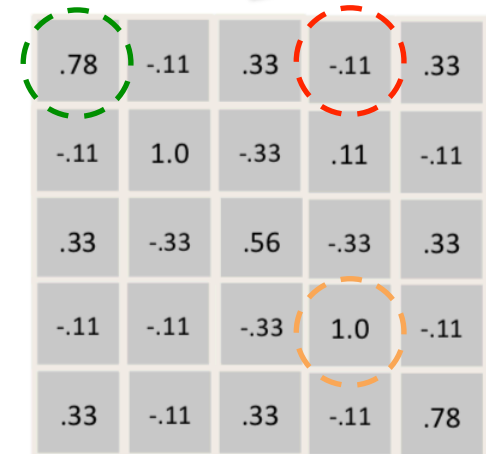
A convolution (function) of two function  $f(x)$  and  $g(y)$  is defined as the sum of the products of one image  $f(\cdot)$  at location  $x$  and another image,  $g(\cdot)$  (called the filter) at a different location  $y-x$ . At pix level:

$$\int_{\infty} f(x)g(y-x)dx = \frac{1}{N} \sum_v f(x)g(y-x) = \text{Net Proportion of pixels in agreement}$$



Filter size N=9

=

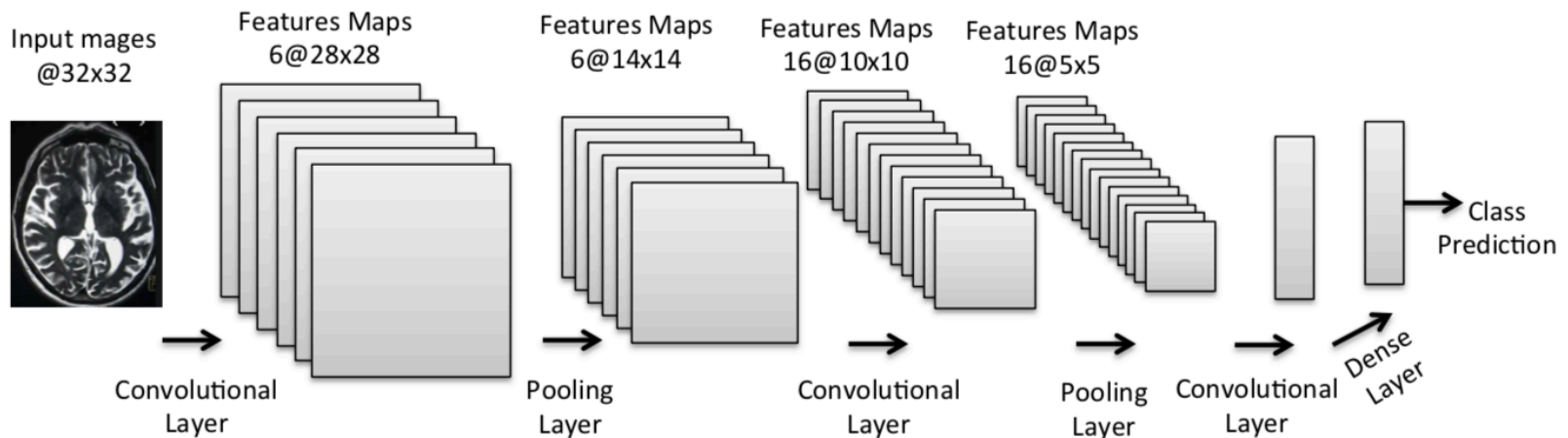


A high value indicates good match between filter and original images

High values on diagonal elements indicate the backslash image

# CNN For Disease Diagnosis

- **Pooling Layer** => Reduce Resolution or Size of Images
- **Convolutional Layer** => Filtering Specific Features
- **Dense Layer** => Fully Connected Layer



# Sample Of Mnist Handwritten Digits



# CNN For Written Digits Recognition 98.7% Accuracy Using keras In R

```
> table(test.predY,mnistTest[,1])
```

test.predY	0	1	2	3	4	5	6	7	8	9
0	975	0	2	0	0	2	6	0	4	2
1	0	1133	4	0	0	0	3	4	1	3
2	0	2	1019	3	1	1	0	11	3	2
3	0	0	0	998	0	3	0	0	1	1
4	0	0	1	0	968	0	2	0	1	4
5	0	0	0	5	0	881	2	1	0	3
6	3	0	0	0	4	4	944	0	1	0
7	1	0	6	2	0	1	0	1008	1	6
8	1	0	0	1	1	0	1	1	958	3
9	0	0	0	1	8	0	0	2	4	985

```
> mean(mnistTest[,1] == test.predY)
```

```
[1] 0.9869987
```

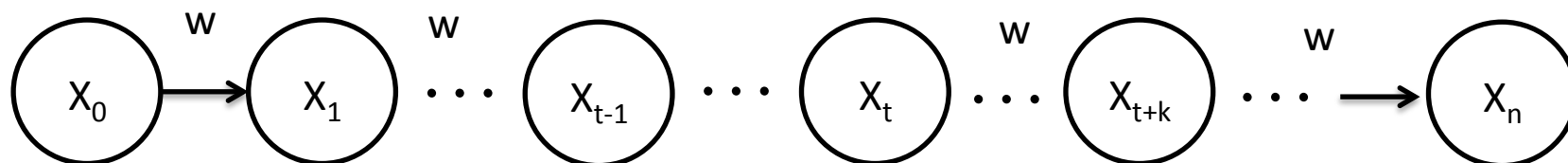
# Applications Of CNN

- Hussain (2017), Cascaded deep CNN for a **brain tumor segmentation**.
- Farooq (2017), CNN-based method for the classification of **Alzheimer's disease in MRI images**
- Moeskops et. al (2016), a multiscale CNN-based approach for automatic segmentation of MR Images for classifying voxel into **brain tissue classes**
- Prasoon et. al (2013), a tri-planar CNN used for segmentation of tibial cartilage in **knee MRI images**
- Zhang et al (2015), segmentation of isointense brain tissue presented through a CNN using a multimodal MRI dataset by training the network on three patches that are extracted from the images
- Anthimopoulos et al (2016), **lung pattern classification** for interstitial lung diseases using a deep convolutional neural network
- Cole et al (2016), **Predicting brain age** with deep learning from raw imaging data results in a reliable and heritable biomarker
- Esteva et al (2017), Dermatologist-level classification of **skin cancer** with deep neural networks

# Architecture Of Recurrent Neural Network (RNN)

- Recurrent Neural Network (RNN)
  - ANN for modeling temporal dynamic behavior
  - Unlike FNN, RNNs can use their internal state as memory to process sequences of inputs.
- Basic RNN
  - Suffers from a short-term memory problem
  - Difficult to carry information from earlier time steps to later ones when the sequence is long as such English text
- Long Short-Term Memory Networks (LSTMs)
  - Designed to deal with long-term dependency problems
  - Used for LNP, stock market prediction, voice recognition, motion picture captioning, and poem and music generation.

# Vanishing And Exploding Gradient Problem & Solutions



$$W_n X_0 \rightarrow \begin{cases} 0, & W < 1 \\ \infty, & W > 1 \end{cases}$$

$$\frac{\partial}{\partial W} W_n X_0 \rightarrow \begin{cases} 0, & W < 1 \\ \infty, & W > 1 \end{cases}$$

## Problem :

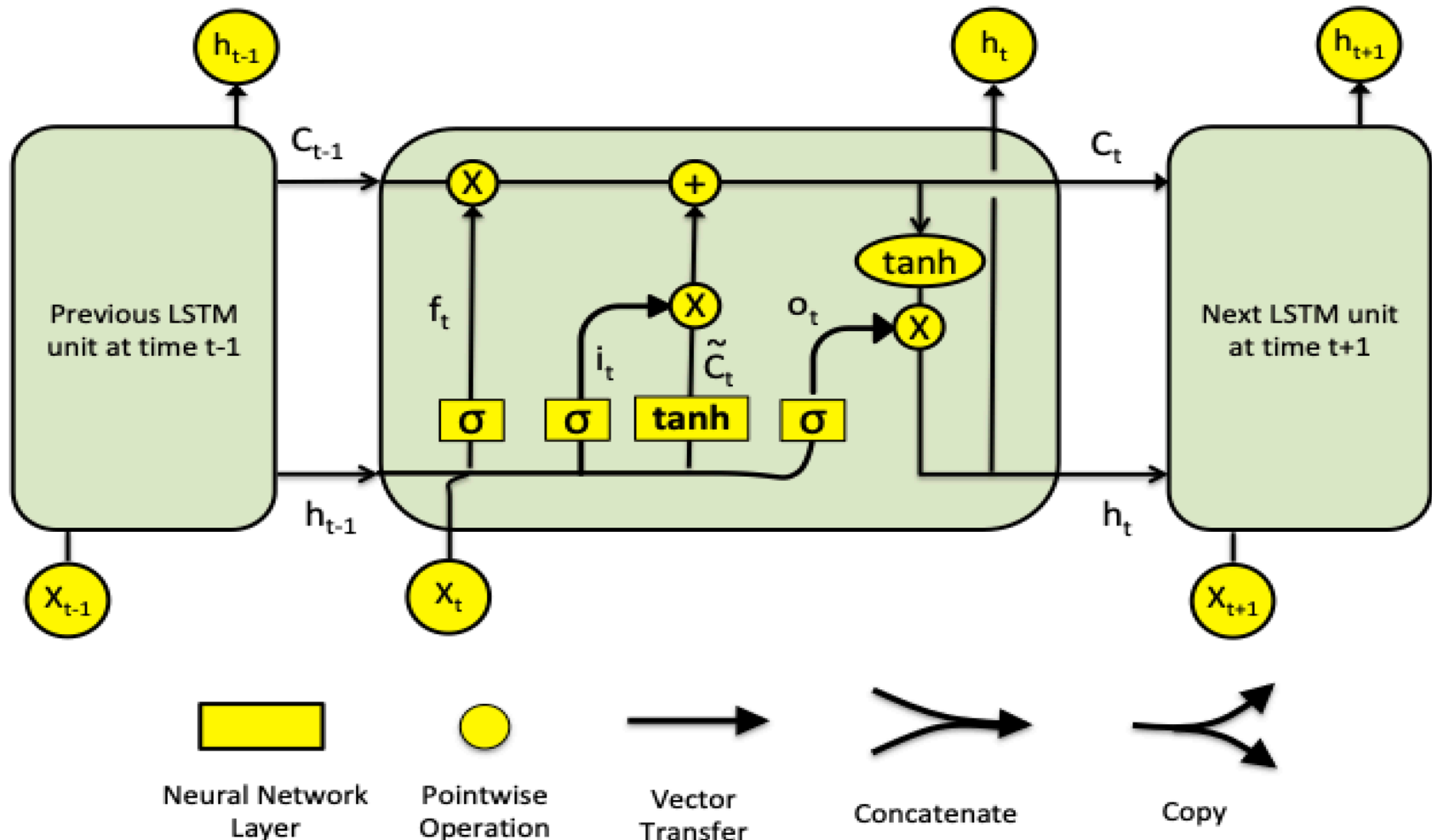
When the number of layers gets very large, the gradients at early layers vanish and changes of the corresponding weights have minimal effect on the outcome

## Solutions:

- Introduce Skip Connections (e.g., if... then...)
- Leaky Recurrent Units
- Gate Recurrent Network
- Long Short-Term Memory Network (LSTM)
- Deep Belief Network

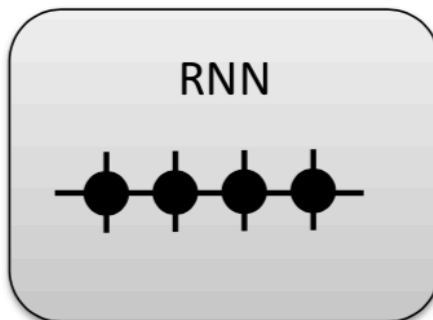
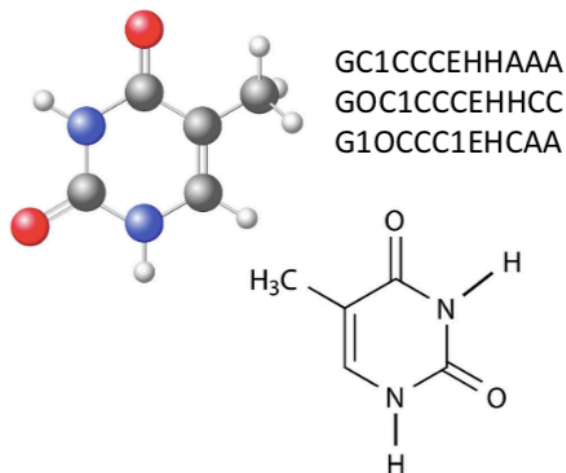
# Architecture Of Long Short-Term Memory Network

The Repeating Module in an LSTM with 4 Interacting Layers in Each Module

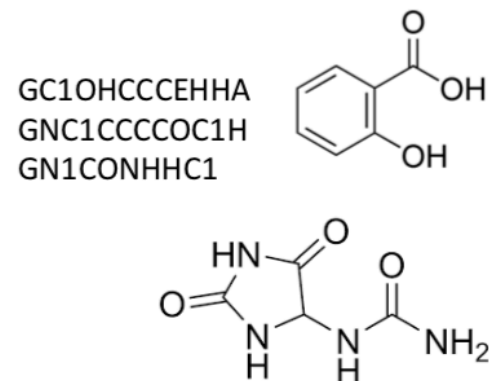


# Application Of RNN To Molecular Design

**Training set:** various drugable molecules expressed as one-dimensional sequences of letters



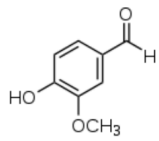
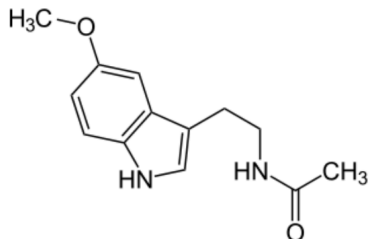
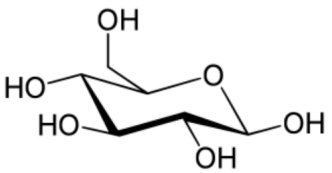
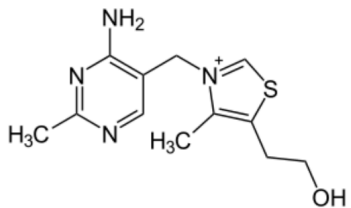
**Output:** nova molecules generated by sampling from the RNN or the conditional distributions



Gupta, et al. (2018) trained their RNN on 541,555 SMILES strings, with lengths from 34 to 74 SMILES characters (tokens). The RNN model can be used to generate sequences one token at a time, as these models can output a probability distribution over all possible tokens at each time step. Typically, the RNN aims to predict the next token of a given input. Sampling from this distribution would then allow generating novel molecular structures.

# Simplified Molecular-Input Line-Entry System (SMILES)

SMILES describe the structure of chemical species using short ASCII strings. In 2007, an open standard called OpenSMILES was developed in the open-source chemistry community

Vanillin		<chem>O=Cc1ccc(OC)c(OC)c1</chem> <chem>COCc1cc(C=O)ccc1O</chem>
Melatonin (C <sub>13</sub> H <sub>16</sub> N <sub>2</sub> O <sub>2</sub> )		<chem>CC(=O)NCCC1=CNc2c1cc(OC)cc2</chem> <chem>CC(=O)NCCc1c[nH]c2ccc(OC)cc12</chem>
Glucose (glucopyranose) (C <sub>6</sub> H <sub>12</sub> O <sub>6</sub> )		<chem>OC[C@H](O)[C@H](O)[C@@H](O)[C@@H](O)O</chem> <chem>[C@H]1O[C@H](O)[C@H](O)[C@@H](O)[C@@H]1O</chem>
Thiamine (vitamin B <sub>1</sub> , C <sub>12</sub> H <sub>17</sub> N <sub>4</sub> OS <sup>+</sup> )		<chem>OCCc1c(C)[n+](cs1)Cc2cnc(C)nc2N</chem>

# Applications Of LSTM In NLP

A trained LSTM is to provide the probability of string  $S$  via a conditional probability:

$$P_{\theta}(S) = P_{\theta}(s_1) \prod_{t=2} P_{\theta}(s_t | s_{t-1}, \dots, s_1)$$

- **Sentiment Analysis**: identify text as “positive” or “negative,” filtering spam or classifying email text as spam, classifying the language of the source text
- **Language Modeling**: predicting the probabilistic relationships between words, enabling one to predict the next words or phrases
- **Speech Recognition**: translate speech to text readable by humans
- **Machine Translation**: translate text between languages
- **Document Summarization**: a heading, abstract for a document
- **Question Answering System**: Process Questions and Provide Answers in a natural language

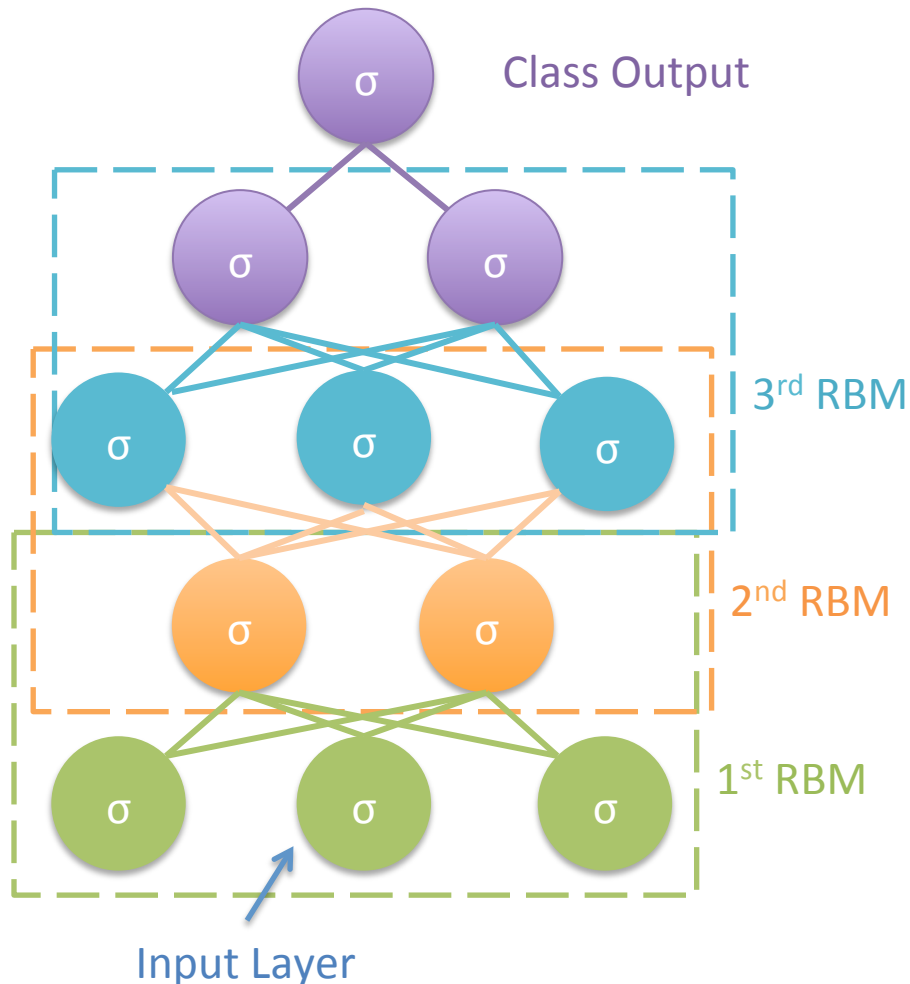
# Natural Language Processing (NLP) Versus Molecular Design

- **Features In NLP**
  - **Words** at difference locations or sequence of words in a sentence
- **Word Embedding**
  - Semantic modeling requires mapping words to numbers or word embedding.
  - Word embedding involves a mathematical embedding from a space with many dimensions per word to a continuous vector space with a lower dimension.
- **Features In Molecular Design**
  - Molecular models exhibit tree-structures, which can be transferred into one-dimensional sequences of different **substructures**
- **Word Embedding In Biological sequences**
  - Word Embedding for n-gram for **biological sequences** (e.g. DNA, RNA, and Proteins) - Asgari and Mofrad (2015).
  - The representation can be used in deep learning in **proteomics and genomics**.
- **Long Short-Term Memory Nets**
  - NLP deals with sequences of words. Drug discovery deals with gene sequences or molecular structures represented by a sequence of biological substructures
  - LSTMs can be used for compound screening and molecular design as in NLP.

# Deep Belief Network (DBN)

- Two Major Challenges In QSAR Studies and Drug Design
  - (1) the large number of descriptors may have autocorrelations
  - (2) proper parameter initialization in model prediction to avoid an over-fitting problem.
- DBN Combines Unsupervised and Supervised Learning
  - For efficient learning
  - Each DBN layer is trained independently via unsupervised portion
  - The unsupervised learning is used for dimension reduction or selecting subsets of descriptors.
  - After the completion of unsupervised learning, the output from the layers is refined with supervised logistic regression.

# Architecture Of Deep Belief Net With Straggled Restricted Boltzmann Machine (RBM)



- RBMs are two-layer neural nets that constitute the building blocks of deep-belief networks.
- The first layer of RBM is called the visible, or input layer, and the second is the hidden layer.
- Activation function  $\sigma$  = normalized exponential energy function.
- Two-Stage Learning: Unsupervised learning for feature selection = training RBM in parallel, followed by supervised learning using backpropagation

# Single-Step Contrastive Divergence (CD-1)

## Procedure For RBM And DBN

- Take a training set  $v$  and initial  $w$ , compute the probabilities of the hidden units
- Sample a hidden activation vector  $h$  from this probability distribution.
- Compute **positive gradient** = outer product of  $v$  and  $h$
- From  $h$ , sample a reconstruction  $v'$  of the visible units, then resample the hidden activations  $h'$  from this (Gibbs sampling step).
- Compute **negative gradient** = outer product of  $v'$  and  $h'$
- Update to weight  $W$  with increment = Learning\_rate \* (positive gradient - negative gradient):

$$\Delta W = \epsilon(vh^T - v'h'^T).$$

- Update the biases  $a$  and  $b$ :

$$\Delta a = \epsilon(v - v'), \Delta b = \epsilon(h - h').$$

- The above procedure starts from the input layer or the first RBM later and move gradually to the output layer.
- When the unsupervised training is finished for all RBM, the supervised learning (classification) starts to fine tuning the weights.

# Application Of Deep Belief Network

- Hinton et al (2006) proposed a fast learning algorithm used for deep belief networks.
- Zhen et al (2016) proposed a convolutional deep belief network is for direct estimation of a **ventricular volume from images** without performing segmentation at all
- Jaekwon, et al (2017) compared DBNs with other methods in cardiovascular risk prediction. They proposed a cardiovascular disease prediction model using the sixth Korea National Health and Nutrition Examination Survey (KNHANES-VI) 2013 dataset to analyze **cardiovascular-related health data**. They show that statistical DBN-based prediction model has an 83.9% accuracy.
- Ghasemi et al (2018) used a deep belief network to evaluate the DBN's performance using Kaggle datasets with fifteen targets containing more than 70k **molecules**. The results revealed that an optimization in parameter initialization will improve the ability of deep neural networks to provide high quality model predictions.

# Software Packages For Deep Learning

## (TensorFlow, Keras, keras and kerasR)

- TensorFlow
  - Multiple levels of abstraction powered by Microsoft
  - An open source artificial intelligence library, using data flow graphs to build models.
  - For creating large-scale neural networks with many layers
  - Scale -> Vector -> Matrix -> Tensor.
  - TensorFlow = Multidimensional Data Flow from Layer to Layer in Artificial Neural Networks
- Keras
  - High-level neural networks API written in Python
  - Capable of running on top of TensorFlow, CNTK, or Theano
  - Developed with a focus on enabling fast experimentation
- The R packages, keras and kerasR
  - Two R version of Keras for the statistical community
  - keras package uses the pipe operator (%>%) to connect functions or operations together, but you won't find this in kerasR
  - kerasR uses the \$ operator to make your model
  - kerasR contains functions that are named in a similar, but not identical way as the original Keras package.

# Machine Learning Packages In R

---

<b>R Package</b>	<b>Description</b>
neuralnet	Multilayer neural networks using backpropagation
keras	interface with tensorflow for deep learning
kerasR	interface with tensorflow for deep learning
nnet	Feedforward neural networks
deepnet	Deep learning toolkit in R
h2o	R scripting functionality for H2O
RSNNS	Interface to the Stuttgart Neural Network Simulator (SNNS)
tensorflow	Interface to TensorFlow
darch	Deep Architectures and Restricted Boltzmann Machines
rnn	Package to implement Recurrent Neural Networks (RRNs)
FCNN4R	Interface to the FCNN library that allows user-extensible ANNs
rcppNL	Deep Belief Nets and other deep learning methods

---

# Public Data Sources For AI

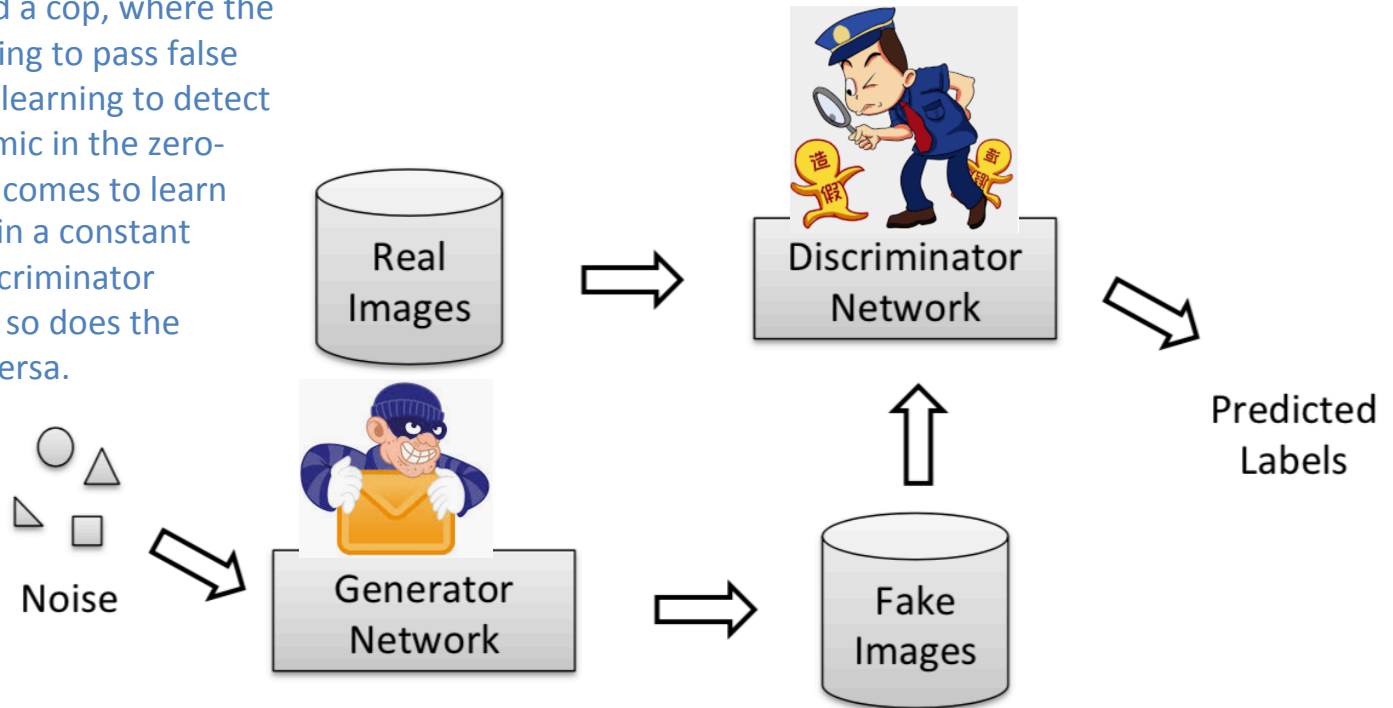
- The commonly used public data repositories for AI applications in cancer prediction and clustering include the TCGA, UCI, NCBI Gene Expression Omnibus (GEO) and Kentrige biomedical databases.
- Kaggle
  - Inside Kaggle you'll find all the code & data you need to do your data science work. Use over 19,000 public datasets and 200,000 public notebooks are ready to conquer any analysis in no time.
  - [www.kaggle.com](http://www.kaggle.com)
- UCI Center for Machine Learning and Intelligent Systems
  - <http://archive.ics.uci.edu/ml/datasets.php>

# Two Special Artificial Neural Networks:

- Generative Adversarial Networks (GANs)
- Autoencoder (Autoassociative Network)

# Generative Adversarial Networks (GANs)

GAN can be viewed as the combination of a counterfeiter and a cop, where the counterfeiter is learning to pass false notes, and the cop is learning to detect them. Both are dynamic in the zero-sum game, each side comes to learn the other's methods in a constant escalation. As the discriminator changes its behavior, so does the generator, and vice versa.

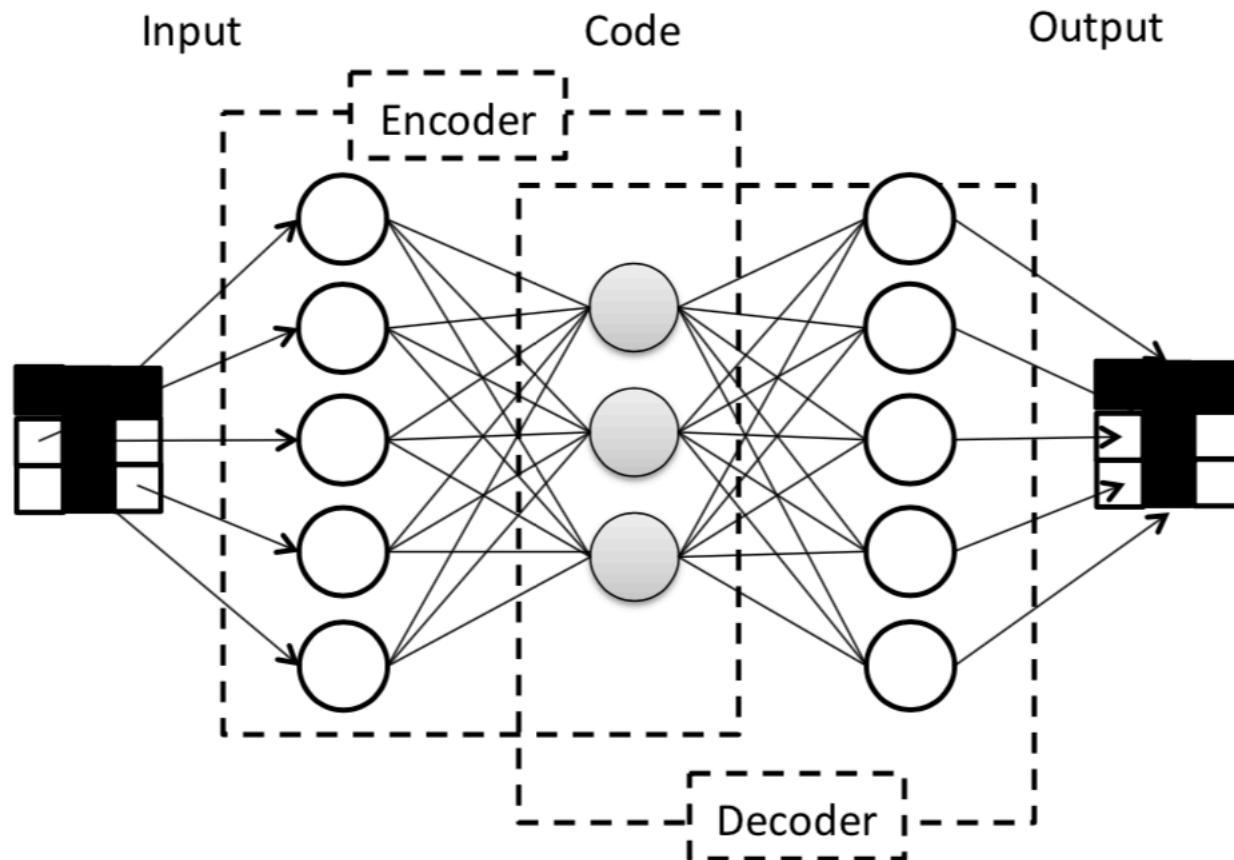


1. Generator takes in random numbers and returns an image.
2. The generated image is fed into the discriminator along with a stream of images taken from the actual dataset.
3. Discriminator takes in both real and fake images and returns probabilities of authenticity.

# Applications Of Generative Adversarial Networks (GAN)

- Imaging markers can be used for monitoring disease progression with or without treatment. Models are typically based on large amounts of data with annotated examples of known markers aiming at automating detection. Christian Doppler et al (2017) developed a GAN that can learn a manifold of normal anatomical variability. Applied to new data such as images containing retinal fluid or hyperreflective foci, the model labels anomalies and scores image patches indicating their fit into the learned distribution.
- Deep generative adversarial networks are the emerging technology in drug discovery and biomarker development. Kadurin et al. (2017) demonstrated a proof-of-concept in implementing a deep GAN to identify new molecular fingerprints with predefined anticancer properties. They also developed a new GAN model for molecular feature extraction problems, and showed that the model significantly enhances the capacity and efficiency of development of the new molecules with specific anticancer properties using the deep generative models.
- Yahi, et al. (2017) propose a framework for exploring the value of GANs in the context of continuous laboratory time series data. The authors devise an unsupervised evaluation method that measures the predictive power of synthetic laboratory test time series and show that when it comes to predicting the impact of drug exposure on laboratory test data, incorporating representation learning of the training cohorts prior to training the GAN models is beneficial.
- Putin, et al. (2018) proposed a Reinforced Adversarial Neural Computer (RANC) for the de novo design of novel small-molecule organic structures based on the GAN paradigm and reinforcement learning. The study shows RANCs can be reasonably regarded as a promising starting point to develop novel molecules with activity against different biological targets or pathways. This approach allows scientists to save time and covers a broad chemical space populated with novel and diverse compounds.

# Autoencoder (Autoassociative Network)



1. Autoassociative: Output = input
2. Dimension Reduction: Hidden layer smaller than input layer
3. Mixture of unsupervised and supervised learning

# Applications of Autoencoder

- Kadurin, et al. (2017) presented an application of autoencoders for generating novel molecular fingerprints with a defined set of parameters.
  - 7-layer architecture with the latent middle layer serving as a discriminator
  - Input and output use a vector of binary fingerprints and concentration of the molecule.
  - NCI-60 cell line assay data for 6252 compounds
  - Screened 72 million compounds in PubChem and select candidate molecules with potential anti-cancer properties.
- Cancer prediction using AI includes predicting the existence of cancer, cancer type and survivability risk. Different types of autoencoders have been used for filtering microarray gene expressions:
  - Stacked denoising autoencoders (Jie, et al., 2015),
  - Contractive autoencoders (Macias-Garcia, et al., 2017)
  - Sparse autoencoders (Rasool, 2013),
  - Regularized autoencoders (Kumardeep, et al., 2017)
  - Variational autoencoders (Way and Greene, 2017)
  - Deep architecture of four layers with 15,000, 10,000, 2000, and 500 neurons (Danaee et al, 2016)

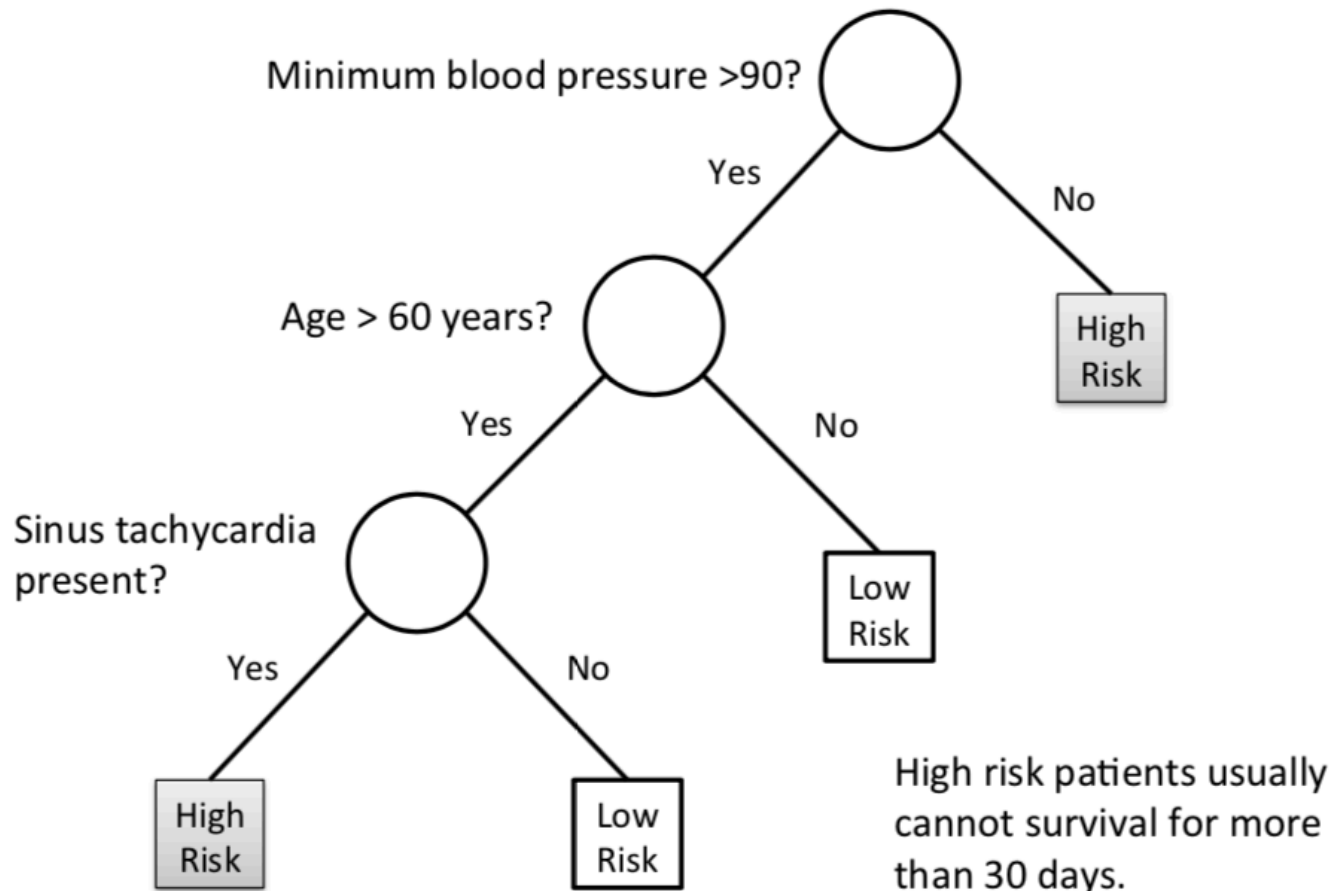
# Decision Tree

- **A Tree Method (TM)**
  - A commonly used supervised learning method
  - Classification & Regression Tree
  - Features are dichotomized for a binary tree
- **Impurity Measures For Optimization**
  - Gini index (CART)
  - Entropy (ID3, C4.5)
  - Misclassification error
- **Overfitting and Error Propagation**
  - For larger trees, an early misclassification error can propagate downstream, eventually leading to poor predictions.

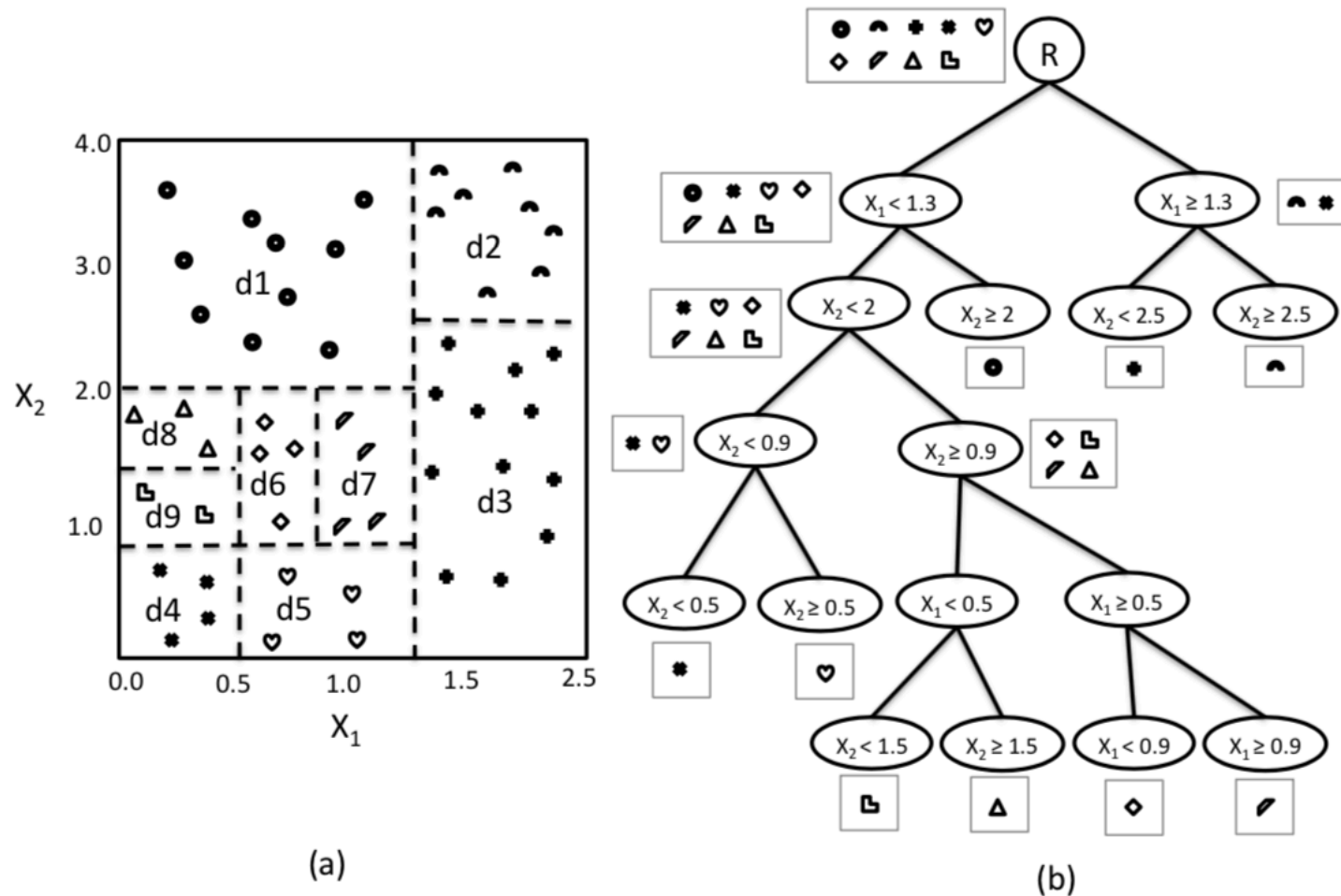
**Solutions:**

  - Tree pruning
  - Bagging & Boosting
  - Random Forest
- **Applications in Medicine**

# Example Of Binary Decision Tree



# The Decision Tree Method For Categorical Outcome



# Common Impurity Measures

For Node  $m$  of A Classification Tree ( predicted probability  $p_{mk}$ ):

## Misclassification Error

$$ME = \frac{1}{n_m} \sum_{x_i \in R_m} I(y_i \neq k) = 1 - \hat{p}_{mk}$$

## Gini Index

$$GI = \sum_{k=1}^K \hat{p}_{mk}(1 - \hat{p}_{mk})$$

## Gross-Entropy or Deviance

$$GE = - \sum_{k=1}^K \hat{p}_{mk} \ln(\hat{p}_{mk})$$

# Threshold Determination And Tree Pruning

- **Determining An Optimal Tree Model**
  - Two competing factors: the accuracy of the tree method and computational efficacy.
  - For a given tree depth, optimal threshold for each parameter is obtained using the greedy method: try a range of different thresholds
  - Any of the three impurities can be used, but often misclassification rate.
  - Tree size is a tuning parameter governing the model's complexity
- **Cost-Complexity Pruning**
  - Split tree nodes only when the decrease in error due to the split exceeds some threshold. This strategy is too short-sighted since a seemingly worthless split might lead to a very good split below it.
  - The preferred strategy is to grow a large tree, stopping the splitting process only when some minimum node size or tree depth is reached. Then this large tree is pruned using cost-complexity pruning

# Bagging, Boosting, And Random Forest

- A single big tree can propagate classification errors to the leaves, making the prediction very unstable.
  - Solutions: Bagging, Boosting, and Random Forest
- **Bagging** (Bootstrap Aggregation)
  - Forming an average of many different trees.
- **Boosting**
  - Output the class:  $G(\mathbf{x}) = \text{sgn} \left( \sum_{i=1}^{n_t} \alpha_i G_i(\mathbf{x}) \right)$
  - Weak classifiers  $G_i(x)$  are only slightly better than random guessing.
  - $\alpha_i$  computed by the boosting algorithm (AdaBoost Algorithm) to weight more on better classifiers  $G_i(x)$ .
- **Random Forests**
  - an ensemble classifier that consists of many decision trees with randomly selected features
  - the final class is the mode of the class's output by individual trees.

# Applications Of Decision Tree Methods

- Singh, et al (2015) have used RFs for **bioactivity classification**
- Wang et al. (2015) have used the RF method to model the **protein-ligand binding** affinity between 170 complexes of HIV-1 proteases, 110 complexes of trypsin, and 126 complexes of carbonic anhydrase.
- Kumari et al (2015) have constructed an improved RF by integrating bootstrap and rotation feature matrix components for **drug targets identification**.
- Mistry, et al. (2016) have used RF and DTs to model the **drug-vehicle toxicity relationship**. Their data set included 227,093 potential drug candidates and 39 potential vehicles. The resulting model predicted the toxicity relief of drugs by specific vehicles.

# Similarity-Based and Similarity- Principle-Based Machine Learning

# New Scientific Paradigm

- Controversies in Statistical Evidence and Scientific Discovery
- Call for New Paradigms – AI/Machine Learning
- The Similarity Principle (SP)
  - Human intelligence in daily life
  - Role of SP in scientific discovery
  - Similarity-Based Machine Learning (SBML)

# Simpson's Paradox

Drug Responses in Males and Females

	Drug A	Drug B
Males	200/500	380/1000
Females	300/1000	140/500
Total	500/1500	520/1500

**All patients:** B better than A

**Males:** A better than B

**Females:** A better than B

**Question:** Should a patient take A or B?

Drug Responses in Young and Old Females

	Drug A	Drug B
Young Females	20/200	40/300
Old Females	280/800	100/200
Total	300/1000	140/500

**All Females:** A better than B

**Young Females:** B is better than A

**Old Females:** B is better than A

**Question:** Should a patient take A or B?

- **How to interpret the results if these patients + you = the entire patient population (One response from you would not change the direction of the drug effect)?**
- **Controversial Question: How Specific Is Too Specific?**
- It is philosophical question or statistical question: Don't put cart before the horse
- **Fundamental Solution: The Similarity Principle**

# Controversies In Prediction Of Drug Effect

- Longevity Prediction – Paradox of Traveling
  - Does the prediction of person's longevity changes as soon as he arrived at the new country because of its different life-expectance?
- Bias In Predictive Effect
  - Stratified randomized and un-randomness in clinic site selection lead to a different patient distribution (e.g. race distribution) in the clinical trial from the target population
  - If race has effect, the mean drug effect will be usually biased.
- Drug Effect Estimation for Multiregional Trial
  - How small should regions to be, country, state, city?

# Similarity Principle

- All science, and learning itself, is based on a fundamental principle – the similarity principle (Chang, 2012, 2014).
- **The Similarity Principle:**
  - Similar things or individuals will likely behave similarly, and the more similar they are the more similarly they behave.
  - Ex., people with the same (or a similar) disease, gender, and age will likely have similar responses to a medical intervention. If they are similar in more aspects they will have more similar responses.
  - Predicted outcome = similarity-weighted the observed outcomes:

$$\hat{Y}_i \propto \sum_j S_{ij} Y_j$$

- Now do you know how to resolve the Simpson's Paradox?

# Similarity Principle Illustrated

## Similarity Principle

## Illustrations

Similar things behave similarly.

1. QSAR: Compounds with similar structures will have similar activities.
2. Terrorist attack: Sept 11 Every year is a high risky.

Things more similar behave more similarly.

1. Use results of drug test in animals to inform a small first-in-man clinical trial, and
2. Use results from clinical trials to decide whether the drug can be used for large patient population

Roles of outcome and features in similarity

1. Similarity relies on by both the features and outcome variable.
2. The relative importance of different features can be objectively determined using feature-scaling factors

Features selection and grouping.

1. Define the target population and used for model evaluation
2. Applied to scientific discovery and causal relationships

# Two Approaches To Learning From Data

- **Causality Approach**
  - Seeking the relationship between the outcome and the independent variables (attributes)
  - $Y(x) = f(x; a(x'))$ , parameters = function of observed data
- **Similarity Approach**
  - Seeking the relationship between the outcomes between different subjects with different attributes
  - $Y(x) \sim S(x, x')Y(x')$ , similarity = function of observed data
- **Similarity Principle (SP) versus Causality**
  - SP is the foundation of any causal relationships
  - Unavoidable similarity grouping for repetitions of events and causality
- **AI Learning and Scientific Discovery**
  - Show human brains incredible ability
  - Imply human incredible inability to handle everything individually

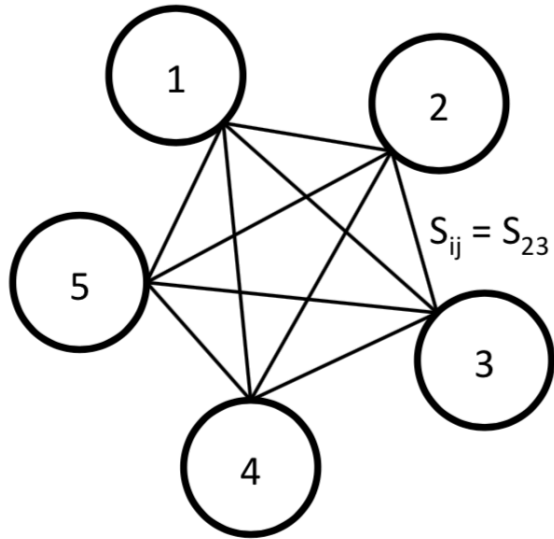
# Subjectivity Of Causality & Controversies In Interpretations Of Cause Effects

- Subjectivity in Potential Cause Selection
- Causes Are Generally Not Independent
- Models Are Not Unique
  - Body Weight =  $f(\text{waist, height})$
  - Body Weight =  $f(\text{waist, height, shoulder height})$
  - Body Weight =  $f(\text{waist, height, shoulder height, foot length})$
  - In fact, all factors are correlated and statistically significant
- Different Models Provide Different Interpretations of Effect (association or causality) of Waist and Height
- Similar Controversies In Drug Effect =  $f(\text{drug, race, gene x, gene y})$ 
  - Simply stating drug effect without specifying other factors is misleading.
  - Interpretation of a factor effect must in the context of all other factors in the model
  - Given a larger but finite population, many factors (and infinite possible combinations of these factors) can be significant. The factors included in the model is subjective and consequently the interpretation drug effect is arbitrary

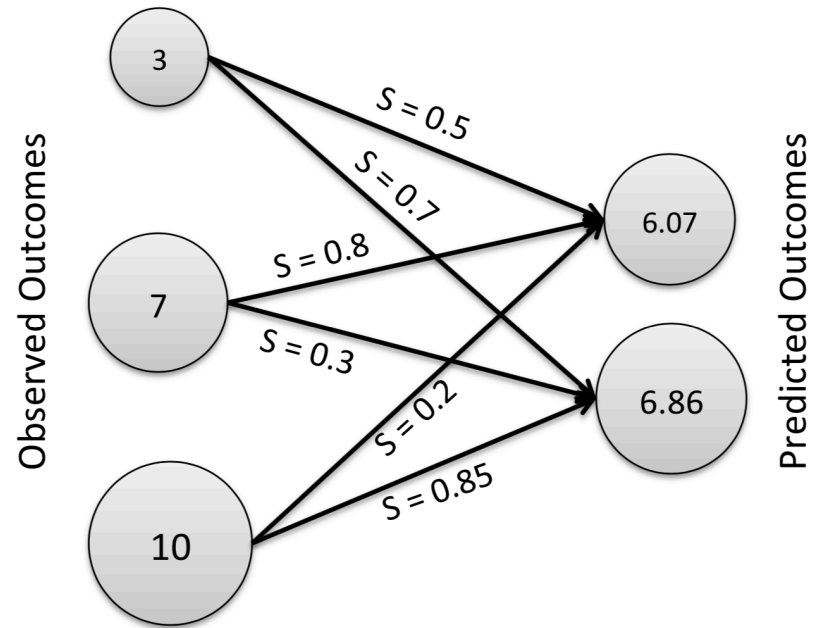
# Dose Analysis of Covariance Make Sense?

- **Model Options:**
  - Model 1:  $\text{Weight} = a_1 * \text{height}$
  - Model 2:  $\text{Weight} = a_2 * \text{height} + b_2 * (\text{shoulder Height})$
  - Model 3:  $\text{Weight} = a_3 * \text{height} + b_3 * (\text{shoulder Height}) + c_3 * (\text{leg length})$
  - Model 4:  $\text{Weight} = d_4 * (\text{head height}) + b_4 * (\text{shoulder height})$
- **Given big data**
  - All models are significant
- **How to interpret the parameters?**
  - What is the effect of height,  $a_1$ ,  $a_2$ , or  $a_3$ ?
  - What is the effect of shoulder height,  $b_1$ ,  $b_2$  or  $b_4$ ?
  - $a_1$ ,  $a_2$ , and  $a_3$  can be very different since height mainly consists of shoulder-height, and leg is a part of shoulder-height and height
  - $c_3$  can be negative since  $a_3$  and  $b_3$  have already include the leg effect, would you say: “longer leg will reduce the weight?”
- **Conclusion:**
  - Many attributes or covariates (including genes) in modeling exhibit strong correlations, the common interpretation of parameters often do not make much sense to me.

# Similarity Principle In Action - Similarix



$$S = \begin{bmatrix} S_{11} & S_{12} & S_{13} & S_{14} & S_{15} \\ S_{21} & S_{22} & S_{23} & S_{24} & S_{25} \\ S_{31} & S_{32} & S_{33} & S_{34} & S_{35} \\ S_{41} & S_{42} & S_{43} & S_{44} & S_{45} \\ S_{51} & S_{52} & S_{53} & S_{54} & S_{55} \end{bmatrix} \cdot$$



Bounded Similarity:  $0 \leq S_{ij} \leq 1$   
 Completely different = 0  
 Identical = 1

# Feature-Scaling Factors (FSF)

Exponential Similarity Function:

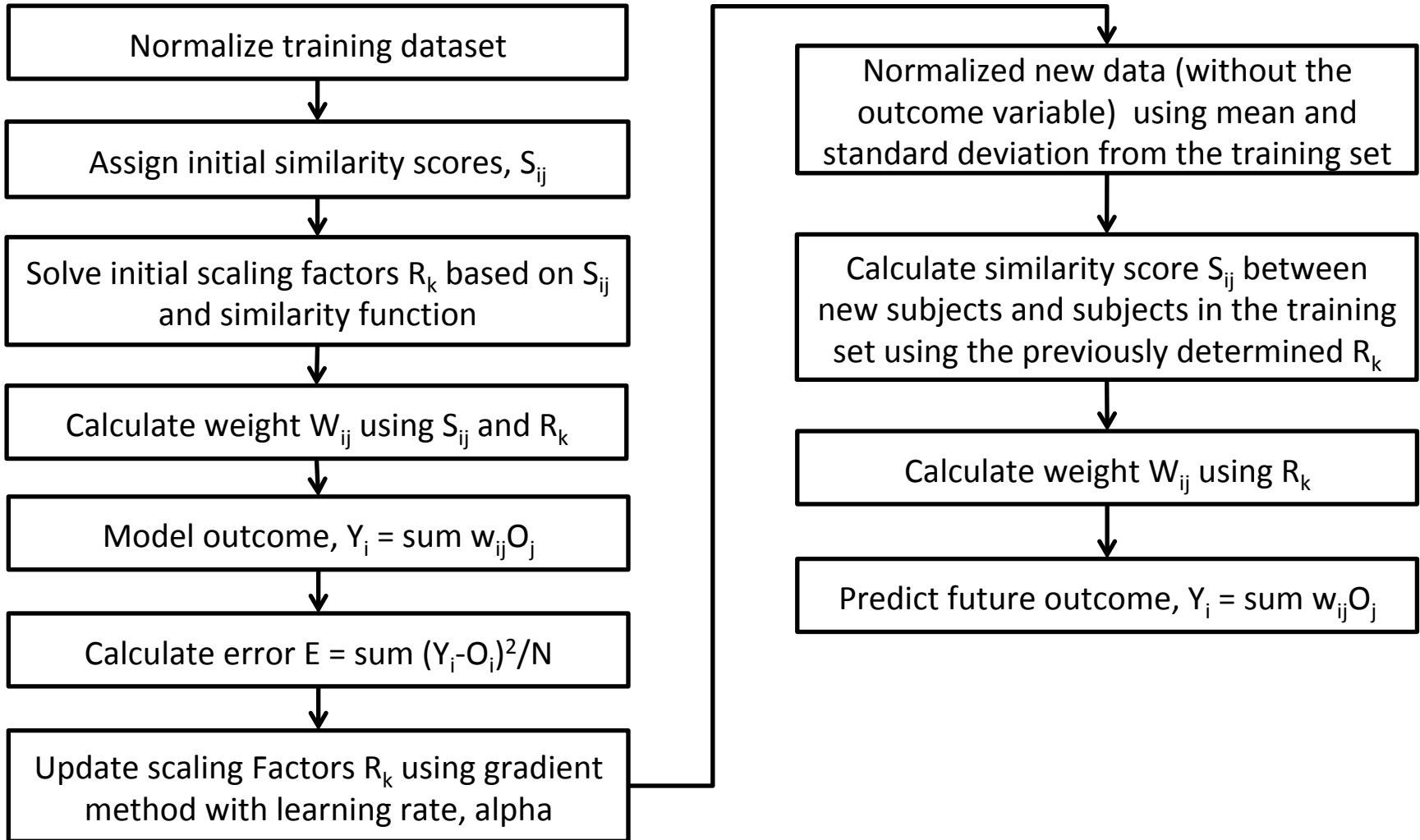
$$S_{ij} = \exp(-d_{ij}^2)$$

Distance Function:

$$d_{ij} = \left( \sum_{k=1}^K (R_k |X_{jk} - X_{ik}|)^2 \right)^{1/2}$$

- $R_k$  depend on Selected Features and Outcome
- Including More Common Features will
  - reduce the difference and increase the similarity among subjects
  - lead to larger  $R_k$  differences to balance out the effect of common features
- Why Exponential Similarity Function
  - Human only can pay attention to limited “local events” and distant events quickly disappear at the horizon
  - Human organ sensibility is on log-scale (e.g., loudness, brightness)

# SBML - Algorithm



# Learning With Regularization

- Loss Functions

- Lasso:

$$\varrho_1(\mathbf{R}; \boldsymbol{\lambda}) = MAD + \lambda \|\mathbf{R}\|_1$$

- Ridge:

$$\varrho_2(\mathbf{R}; \boldsymbol{\lambda}) = MSE + \lambda \|\mathbf{R}\|_2^2$$

- Elastic Net:

$$\varrho_{12}(\mathbf{R}; \boldsymbol{\lambda}, \alpha) = E + \lambda_1((1 - \omega) \|\mathbf{R}\|_2 + \omega \lambda_2 \|\mathbf{R}\|_1), \alpha \in [0, 1]$$

- Minimization with Gradient Method

$$\frac{\partial \varrho_i(\mathbf{R}; \lambda)}{\partial R_m} = \frac{\partial E}{\partial R_m} + \frac{\lambda \partial \|\mathbf{R}\|^2}{\partial R_m}$$

- Learning with rate  $\alpha$

$$R_k^{(t+1)} = R_k^{(t)} - \alpha \frac{\partial \varrho_2}{\partial R_k}$$

# Clinical Trial Example: Rare Disease

## Mean Squared Error (MSE) of Different Machine Learning Methods Based on Bootstraps

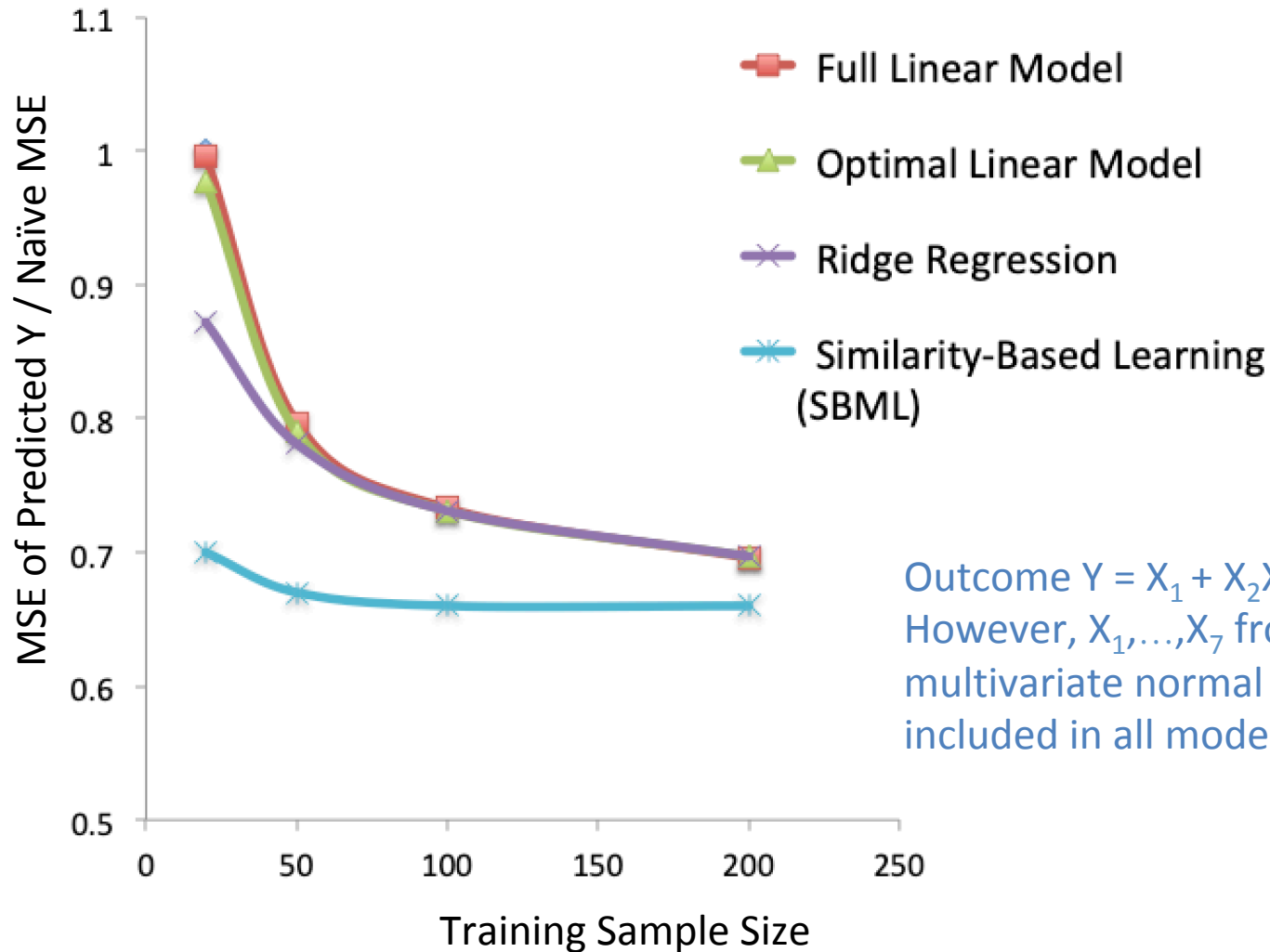
Training Set Size	Full Linear Model	Optimal Linear Model	Ridge Regression	SBML
25%	0.774	0.788	0.796	0.616
50%	0.803	0.802	0.804	0.655
75%	0.806	0.809	0.810	0.668

Note: MSE normalized by variance of the outcomes

# Training, Validation & Evaluation Of AI Methods

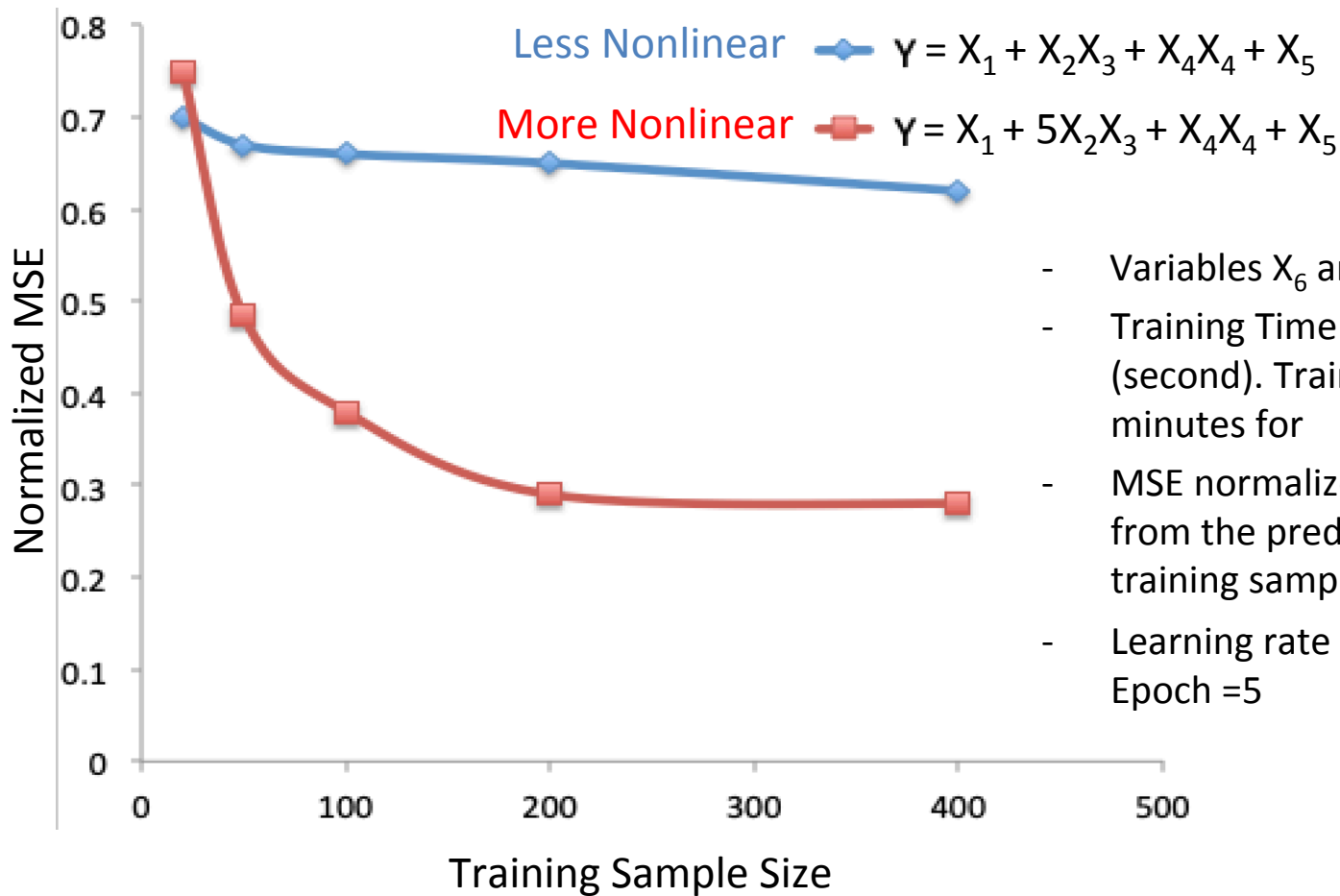
- **Cross-Validation for Tuning Parameters**
  - Exhaustive cross-validation methods: all possible ways of training-validation splits.
  - Leave- $p$ -out cross-validation (LpO CV): exhaustively splits with  $p$  observations as the validation set, the rest for training.
  - Bootstrapping: the random selection of size  $p$  as training set and size  $q$  as test set, with replacement
- **Target Population (TP)**
  - Well-defined TP (WDTP) is needed for model evaluation, but in AI world, WDTP is often missing.
  - Big data: simply split the sample into trainset and testset
  - Larger data: trainsets and testsets obtained from resampling without replacement
  - Small data: use empirical distribution – trainsets and testsets obtained from bootstrapping

# SBML Performance Of Regression Problem



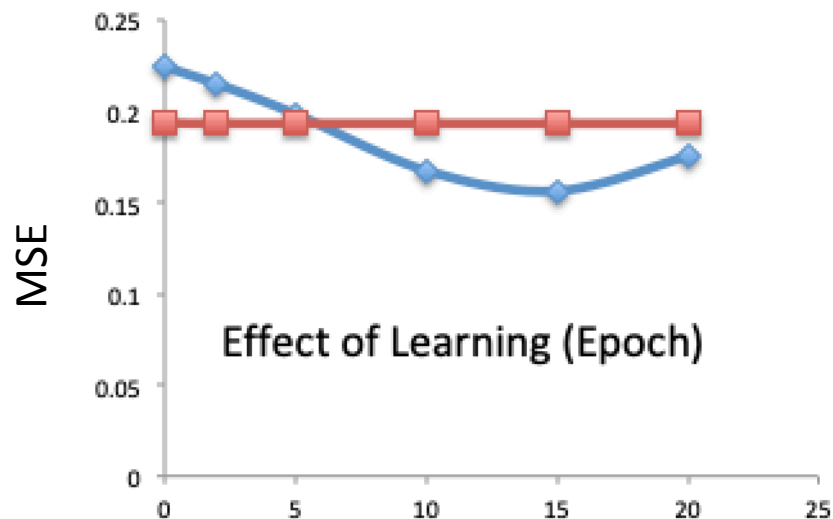
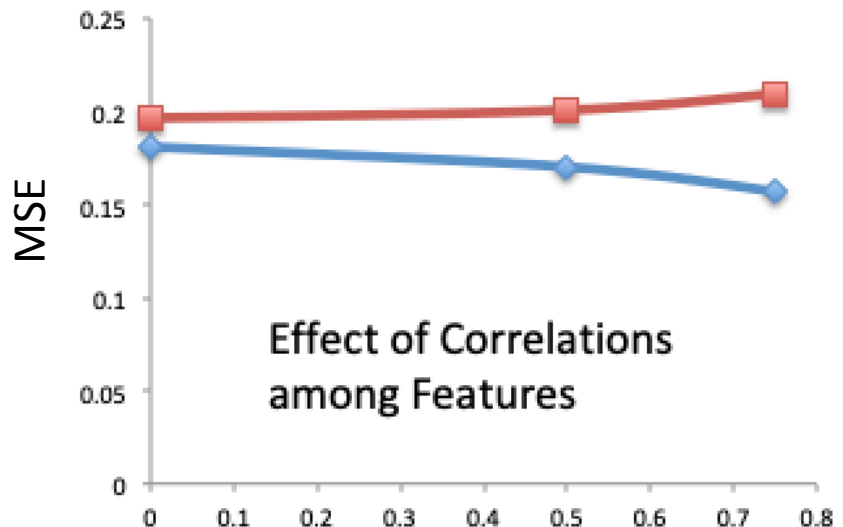
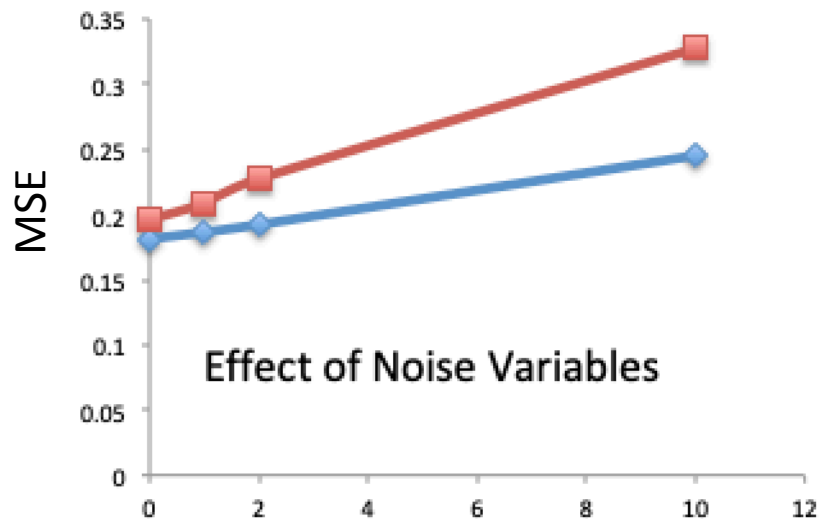
Outcome  $Y = X_1 + X_2X_3 + X_4X_4 + X_5$ .  
However,  $X_1, \dots, X_7$  from standard  
multivariate normal distribution are  
included in all models ( $X_6, X_7$  are noise).

# Precision Versus Complexity Of Data Relationship, Sample Size, And Computational Efficiency



- Variables  $X_6$  and  $X_7$  are noise
- Training Time =  $O(N^2) = 0.09 N^2$  (second). Training Time = 2.4 minutes for trainset size  $N = 400$ .
- MSE normalized by the naïve MSE from the prediction using the training sample mean
- Learning rate  $\alpha = 0.125$ ,  $\lambda = -0.5$  Epoch = 5

# SBML Performance For Classification



- Logistic Model
- SBML
- $Y = \text{sign}(X_1 + X_2 X_3 + X_4 X_5 + 1) / 2$
- $X_i = \text{multivariate normal}$
- Loss function  $\lambda = -0.5$
- Learning Rate  $\alpha = 0.125$
- MSE is based on probability of  $Y=1$ .

# Paradigms Of Drug Development

- Challenges

- Multi-Regional Trial and Subgroup Analysis
- Rare Disease and Precision Medicine

- Solutions

- Smaller Clinical Trials
- Stagewise Marketing-Authorization
- Enhanced Pharmacovigilance & Postmarketing Monitoring

# Kernel Method

- Kernel Method:
  - Kernel As smooth function with normalization factor (Hastie et al, 2001)
  - Non-linear kernel regression without normalization factor (Hastie et al, 2001)
  - Kernel as attributes in learning (training weights) (Scholkopf, etc., 2005)

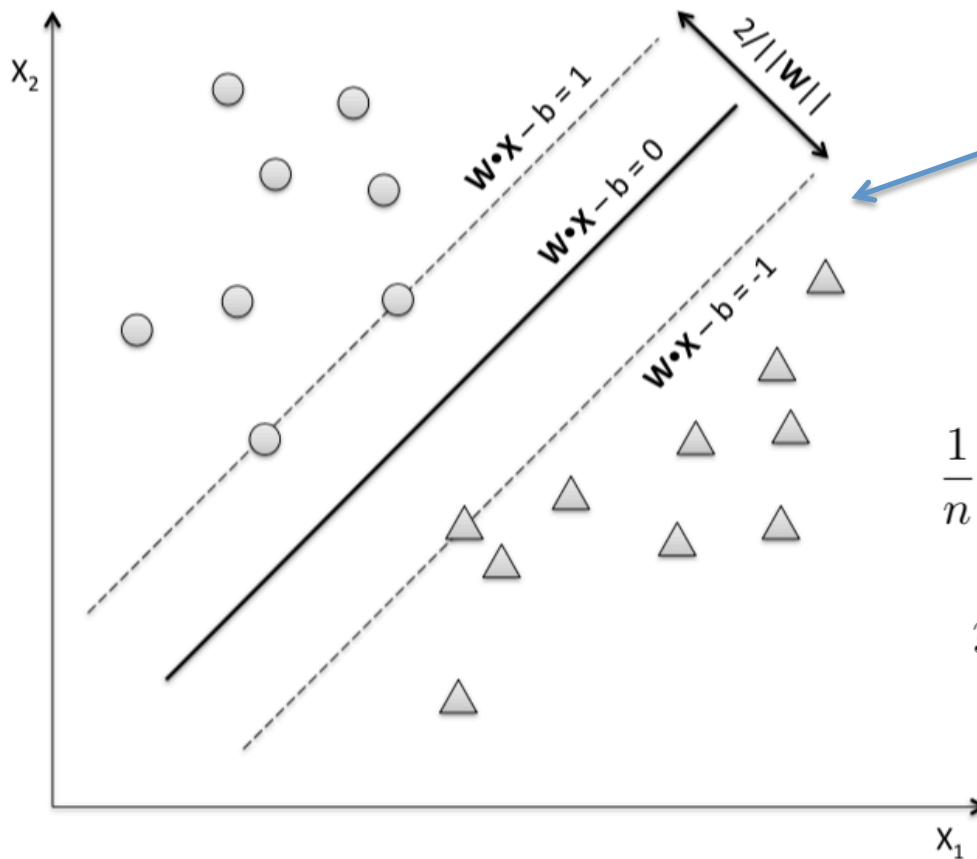
$$\hat{y} = \text{sgn} \sum_{i=1}^n w_i y_i k(\mathbf{x}_i, \mathbf{x}')$$

- Kernel
  - Kernel,  $k(\bullet, \bullet)$ , is the dot product of feature vectors
  - Kernel is a similarity score function
- Feature Selection for Different Objects
  - Sequencing (text, sound, music)
  - Images (black-white, color)
  - Trees (chemical compounds)
  - Networks (metabolic net, disease-symptom net, drug-AE net)

# The Kernel Trick

- Any algorithm for vectorial data that can be expressed only in terms of dot products between vectors can be performed implicitly in the feature space associated with any kernel, by replacing each dot product by a kernel evaluation (Scholkopf, etc., 2005).
- Transfer linear methods (e.g., PCA) into nonlinear methods by simply replacing the classic dot product with a more general kernel, such as the Gaussian RBF kernel.
- Nonlinearity via the new kernel is then obtained at minimal extra computational cost, as the algorithm remains exactly the same.

# Support Vector Machine As A Kernel Method



Classifier with Hard Margin Model:

$$\hat{y}(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^n \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} - b\right).$$

Classifier with Soft Margin Model:

$$\frac{1}{n} \sum_{i=1}^n \max(0, 1 - y_i (\mathbf{w} \cdot \mathbf{x}_i - b)) + \lambda \|\mathbf{w}\|^2$$

$$\hat{y} = \text{sign}\left(\sum_{i=1}^n y_i \alpha_i k(\mathbf{x}_i, \mathbf{x}_j) + b\right)$$

# Support Vector Machine In R

## For Predicting Cognitive Impairment

```
library(ISLR)
library(e1071) # Lib for support vector machine
library(AppliedPredictiveModeling)
# Prepare training and test datasets
data("AlzheimerDisease", package = "AppliedPredictiveModeling")
summary(diagnosis) # labels: either "Impaired" or "Control".
summary(predictors) # predictors
ALZ <- cbind(predictors, diagnosis)
n = nrow(ALZ)
trainIndex = sample(1:n, size = round(0.7*n), replace=FALSE)
ALZtrain=ALZ[trainIndex, ]
ALZtest=ALZ[-trainIndex, ]
# Train SVM
ALZ.svm <- svm(diagnosis~., data=ALZtrain, kernel="linear", cost=10)
summary(ALZ.svm)
pred.test=predict (ALZ.svm , newdata =ALZtest ) # prediction for new data
table(pred.test, ALZtest$diagnosis)
# Study the effect of cost
set.seed (1)
ALZtune=tune.svm(diagnosis~., data=ALZtrain, kernel="linear",
cost=c(0.001, 0.1, 1, 10, 100) )
summary (ALZtune)
```

- Alzheimer's disease is the most common cause of dementia in the elderly.
- Predict the cognitive outcome using the demographics and 130 biomarkers (predictors).

The output confusion matrix is

pred.te	Impaired	Control
Impaired	14	11
Control	9	66

# Application Of Kernel Methods

<<Kernel Methods in Computational Biology>>, Scholkopf et al (2004) collect different applications of kernel methods and support vector machines:

- Inexact Matching String Kernels for Protein Classification
- Fast Kernels for String and Tree Matching
- Local Alignment Kernels for Biological Sequences
- Kernels for Graphs
- Diffusion Kernels
- A Kernel for Protein Secondary Structure Prediction
- Heterogeneous Data Comparison and Gene Selection with Kernel Canonical Correlation Analysis
- Kernel-Based Integration of Genomic Data Using Semi-definite Programming
- Protein Classification via Kernel Matrix Completion
- Accurate Splice Site Detection for Caenorhabditis elegans
- Gene Expression Analysis: Joint Feature Selection and Classifier Design
- Gene Selection for Microarray Data

# Application Of SVM

- SVM-based **protein fold recognition** methods (Shamim et al., 2007, 2013; Damoulas and Girolami, 2008; Dong et al., 2009; Yang and Chen, 2011). The main difference among these SVM-based methods is their feature representation algorithms.
- Poorinmohammad et al. (2015) combined the SVM approach with pseudo amino acid composition descriptors **to classify anti-HIV peptides**, with a prediction accuracy of 96.76%.
- Wei and Zou (2016) provided a review of the recent progress in machine learning-based methods for **protein fold recognition** prior to 2011.

# Unsupervised Learning

# Unsupervised Learning (USL)

- Unsupervised Learning Has No Clear, Correct Answers
- USL = Clustering, Association, and Anomaly Detection
  - Clustering = discovering the inherent groupings in the data. **Ex:** grouping patients by their baseline disease severity and demographic characteristics
  - An association rule learning = discovering rules that describe large portions of the data. **Ex:** people buying product A also tending to buy product B.
  - Anomaly (outlier) and novelty detections = identifying items, events or observations that do not conform to an expected pattern. Anomaly detection can also be a supervised learning. **Ex:** structural defects, medical problems, text errors, or bank fraud
- Applications
  - Grouping breast cancer patients by their genetic markers
  - Grouping Movie viewers by the ratings assigned by movie viewers
  - Finding sale items that are highly related can be very helpful when stocking shelves or doing cross-marketing in sales promotions, catalog design, and consumer segmentation based on buying patterns.

# Common Unsupervised Learning Algorithms

- Apriori Algorithm for Association Rule

- Identify the frequent individual items in the relational database and extends them to larger and larger item sets as long as those item sets appear sufficiently often in the database

- Principal Component Analysis (PCA)

- Find a low-dimensional representation of the observations
- Sequentially find a set of linear combinations of the predictors that have maximal variance and are orthogonal with each other.

- K-Means Clustering for Dimension Reduction

- Partition the  $N$  observations into a pre-specified  $K$  clusters so as to minimize the within-cluster sum of squares:

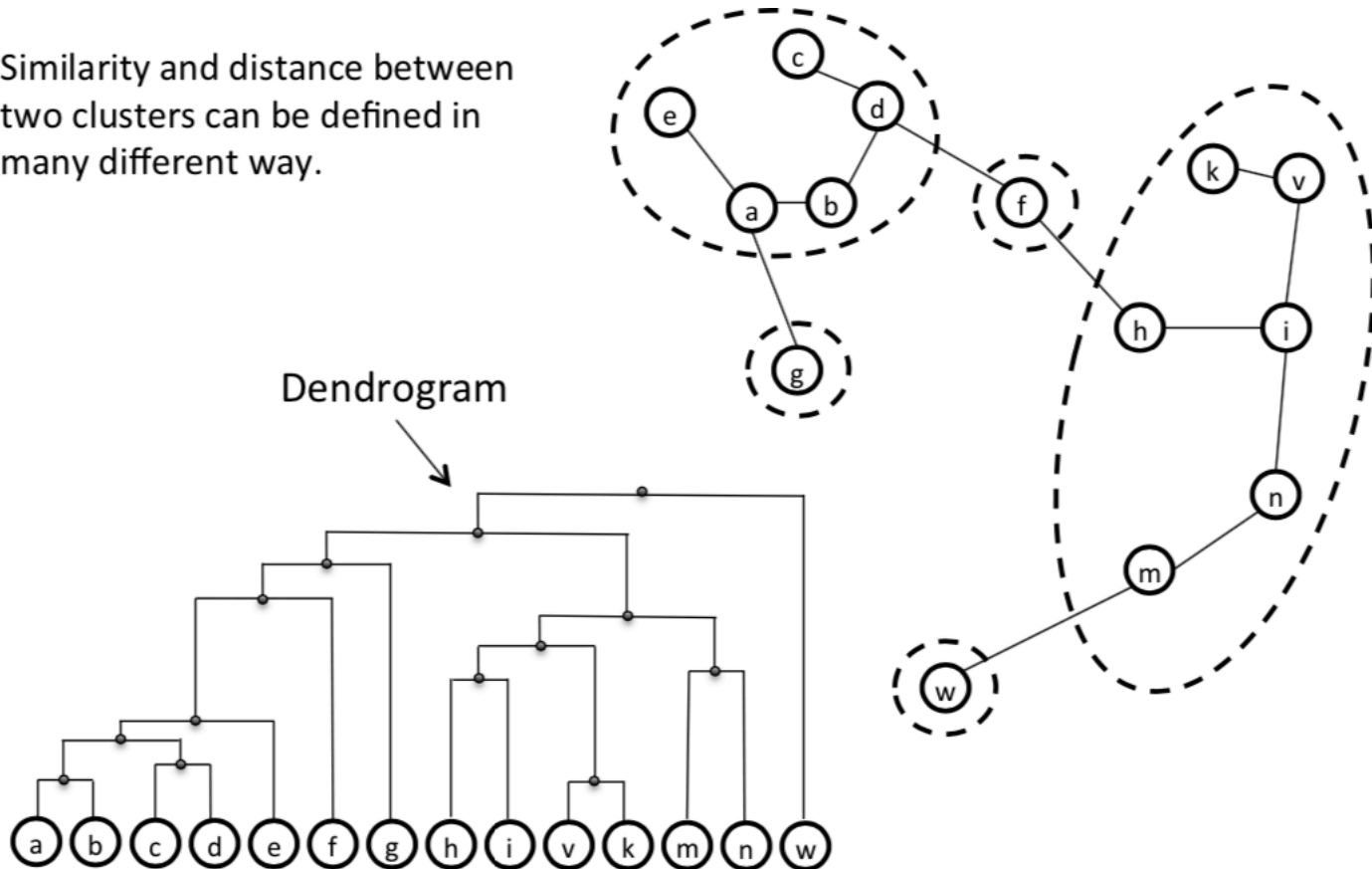
$$\arg \min_{\mathbf{S}} \sum_{i=1}^k \sum_{\mathbf{x} \in S_i} \|\mathbf{x} - \boldsymbol{\mu}_i\|^2$$

- Hierarchical Clustering for Dimension Reduction

- Yields a tree-like visual representation (Dendrogram) of the observations without pre-specified number of clusters

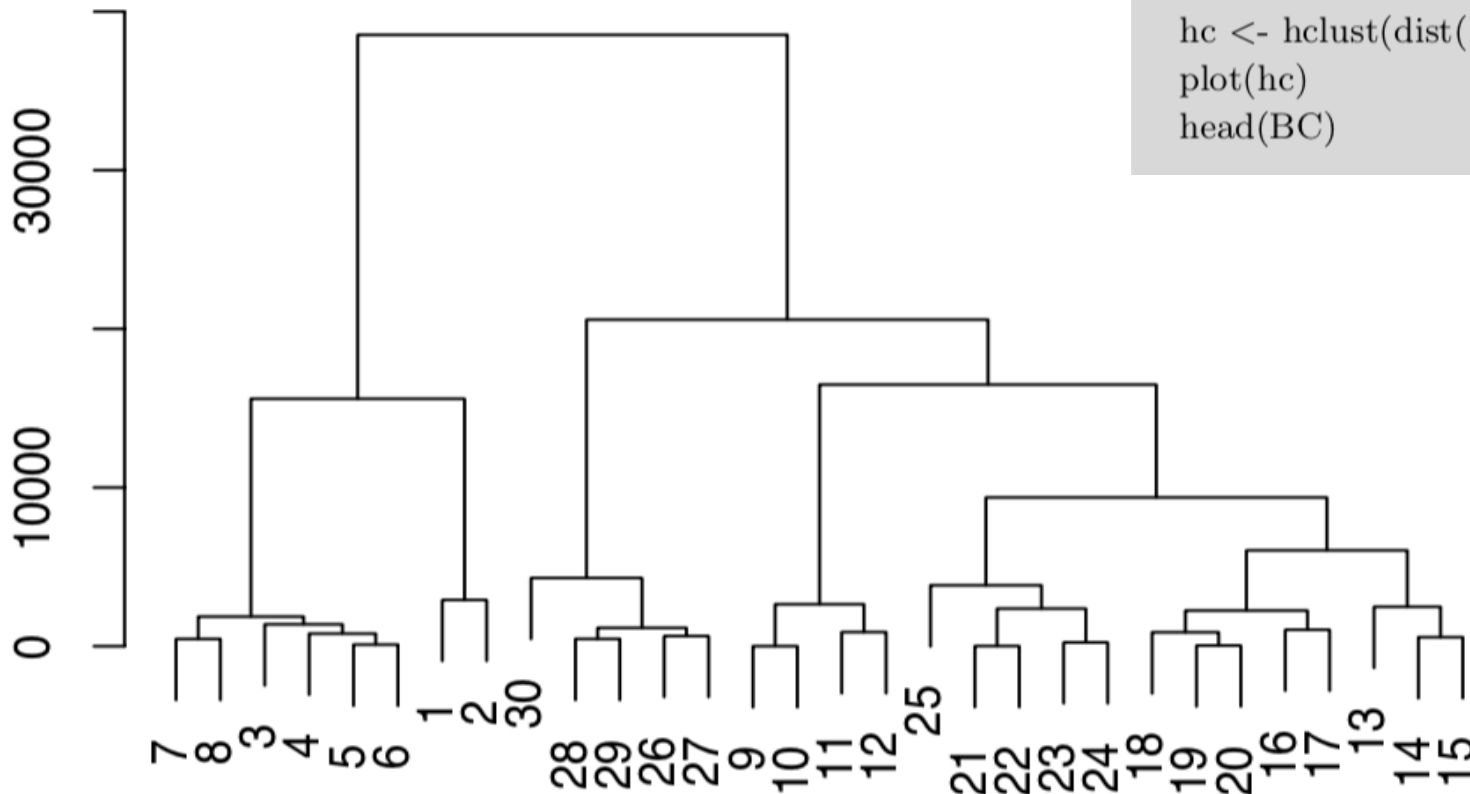
# Hierarchical Clustering Algorithm

Similarity and distance between two clusters can be defined in many different way.



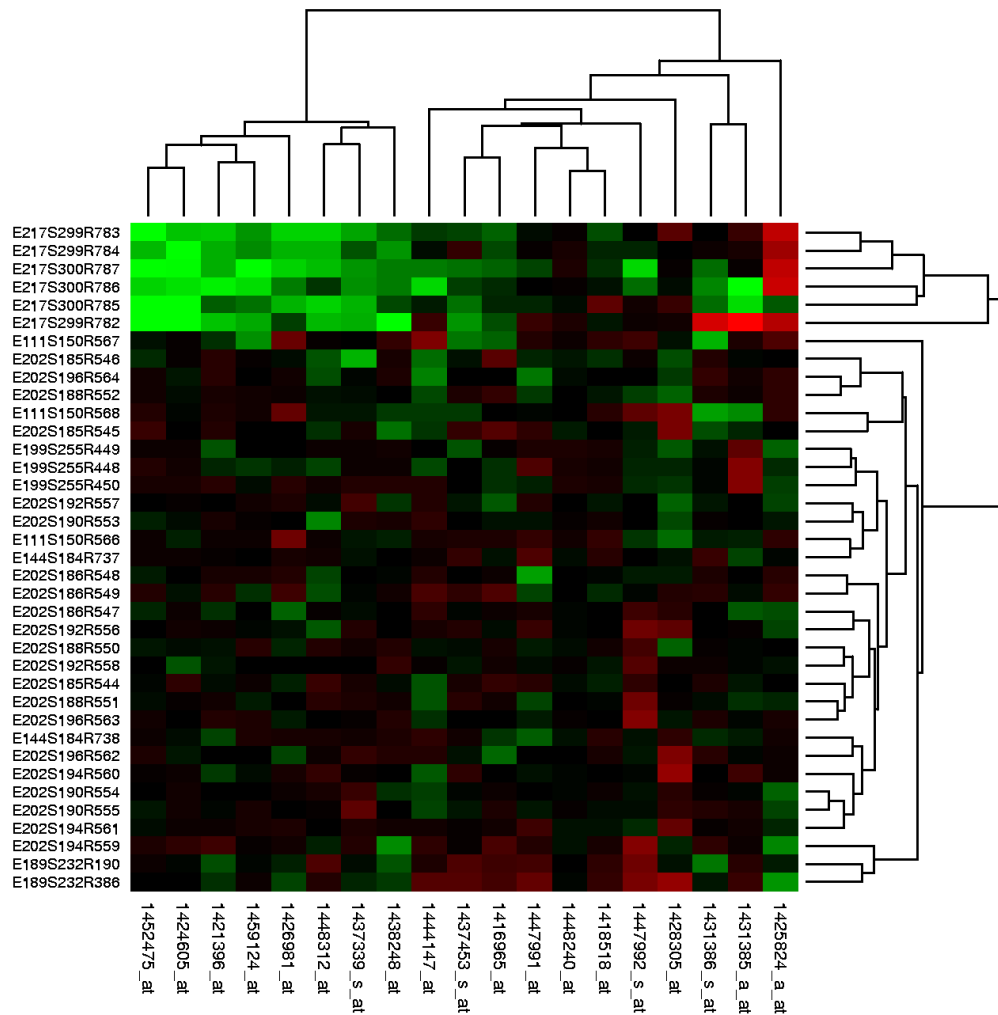
# Hierarchical Clustering For Breast Cancer Patients

**Cluster Dendrogram**



```
library(stats) # for hclust() function
require(graphics)
library(AppliedPredictiveModeling)
BC=BreastCancer[1:30,-11] # Remove
hc <- hclust(dist(BC), "ave")
plot(hc)
head(BC)
```

# Heatmap of Gene Samples



# Applications Of Clustering Methods

- Böcke et al (2005), a hierarchical clustering approach for large compound libraries
- Wallner et al (2010) studied correlation and cluster analysis of immunomodulatory drugs based on cytokine profiles.
- Tan et al (2015) and Rasool et al (2013) use neural network filtering methods are used for extracting representations that best describe the gene expressions.
- Haraty et al (2015) studied an enhanced k-mean clustering algorithm for pattern discovery in health data.
- Yildirim et al (2014) use k-means algorithm to discover hidden knowledge in health records of children and data mining in bioinformatics. They conclude that medical professionals can investigate the clusters which our study revealed, thus gaining useful knowledge and insight into this data for their clinical studies.
- Hameed ert al (2018) proposed a two-tiered unsupervised clustering approach for drug repositioning through heterogeneous data integration.
- MacCuish et al (book, 2019), clustering in bioinformatics and drug discovery
- Ma et al (2019), a comparative study of cluster detection algorithms in protein–protein interaction for drug target discovery and drug repurposing.

# Reinforcement Learning (RL)

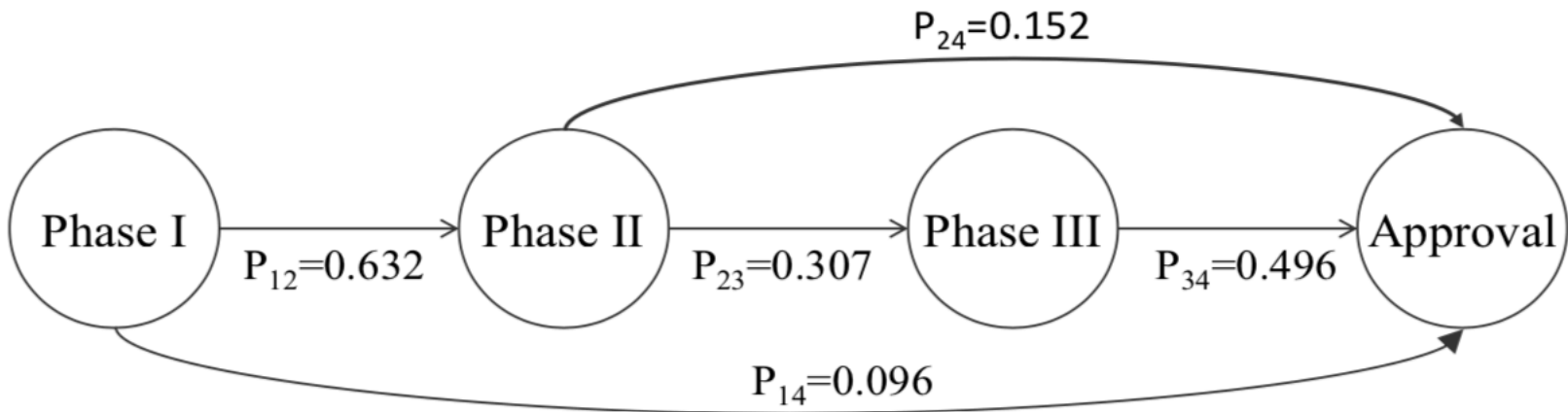
- **Learning**
  - Optimize a utility Function
  - Feedback from one's environment is essential for RL.
  - For the situation when the correct answer is difficult to define or there are too many possible paths to complete the task.
- **Example -AlphaGo Zero**
  - A computer program that plays the board game Go
  - At the 2017 Future of Go Summit, beat the world No.1 ranked player
  - AlphaGo and its successors use a Monte Carlo tree search algorithm to find its moves based on knowledge learned by an ANN from both human and computer play
- **Methods**
  - Stochastic Decision Process
  - Q-Learning
- **Applications**
  - Markov Decision Process for clinical development programs (Chang, 2010)
  - Value iteration and policy iteration algorithms

# Phase Transition Probabilities Of Clinical Trials

	Test Drug	Phase 1 to Ph 2	Phase 2 to Ph 3	Phase 3 to NDA*	NDA* to Appr	Phase 3 to Appr
Hematology	86	73.3%	56.6%	75.0%	84.0%	63.0%
Infectious disease	347	69.5%	42.7%	72.7%	88.7%	64.5%
Ophthalmology	66	84.8%	44.6%	58.3%	77.5%	45.2%
Other	96	66.7%	39.7%	69.6%	88.4%	61.5%
Metabolic	95	61.1%	45.2%	71.4%	77.8%	55.5%
Gastroenterology*	41	75.6%	35.7%	60.6%	92.3%	55.9%
Allergy	37	67.6%	32.5%	71.4%	93.8%	67.0%
Endocrine	299	58.9%	40.1%	65.0%	86.0%	55.9%
Respiratory	150	65.3%	29.1%	71.1%	94.6%	67.3%
Urology	21	57.1%	32.7%	71.4%	85.7%	61.2%
Autoimmune	297	65.7%	31.7%	62.2%	86.0%	53.5%
Neurology	462	59.1%	29.7%	57.4%	83.2%	47.8%
Cardiovascular	209	58.9%	24.1%	55.5%	84.2%	46.7%
Psychiatry	154	53.9%	23.7%	55.7%	87.9%	49.0%
Oncology	1222	62.8%	24.6%	40.1%	82.4%	33.0%
All Indications	3582	63.2%	30.7%	58.1%	85.3%	49.6%

\* Include both new drug and biologic license Applications (NDA & BLA)  
Source: Chang et al. (2019)

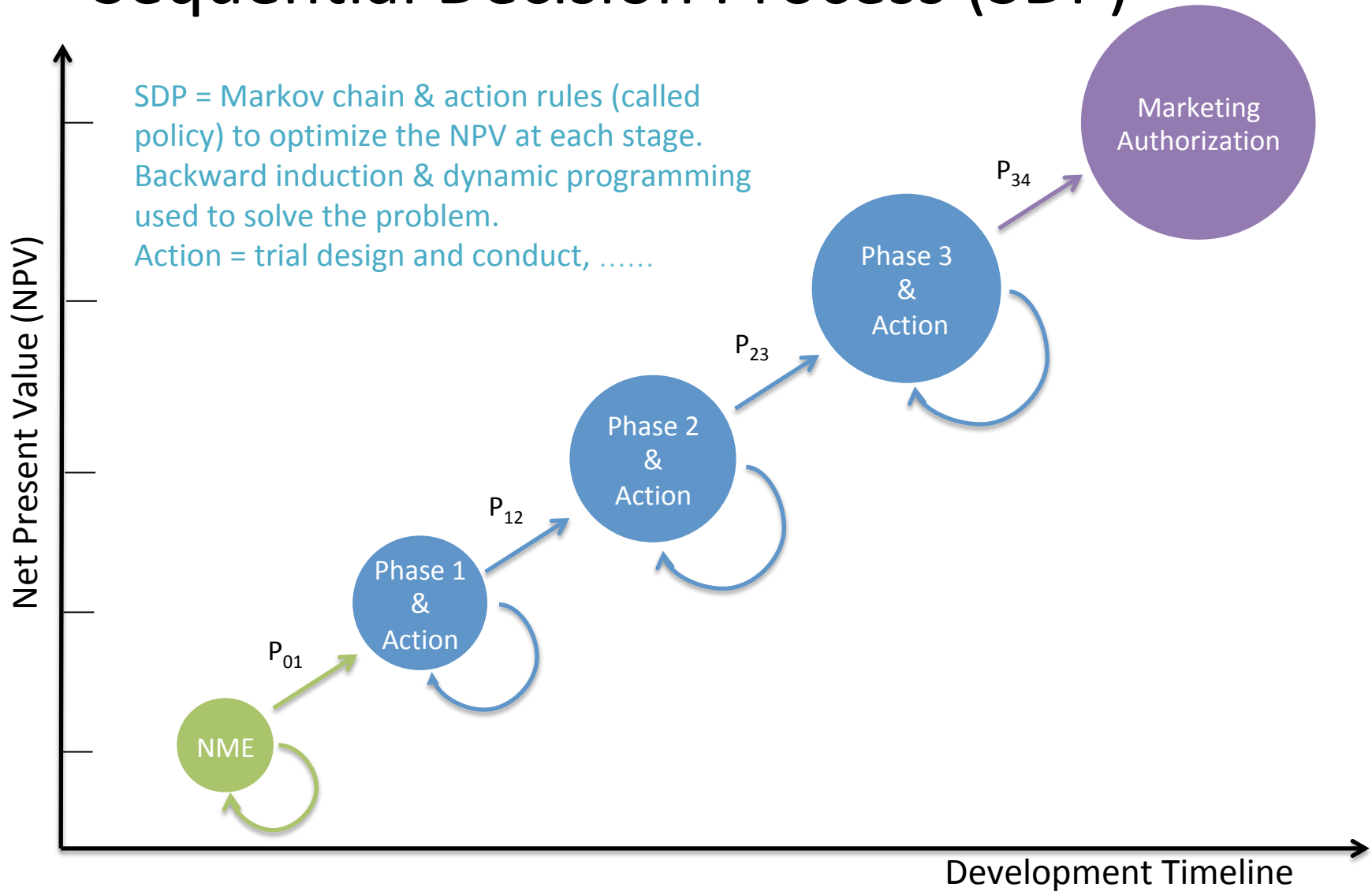
# Markov Chain For Clinical Development Program



$P_{ij}$  = Actual Phase Transition Probability from Phase i to Phase j for all disease indications

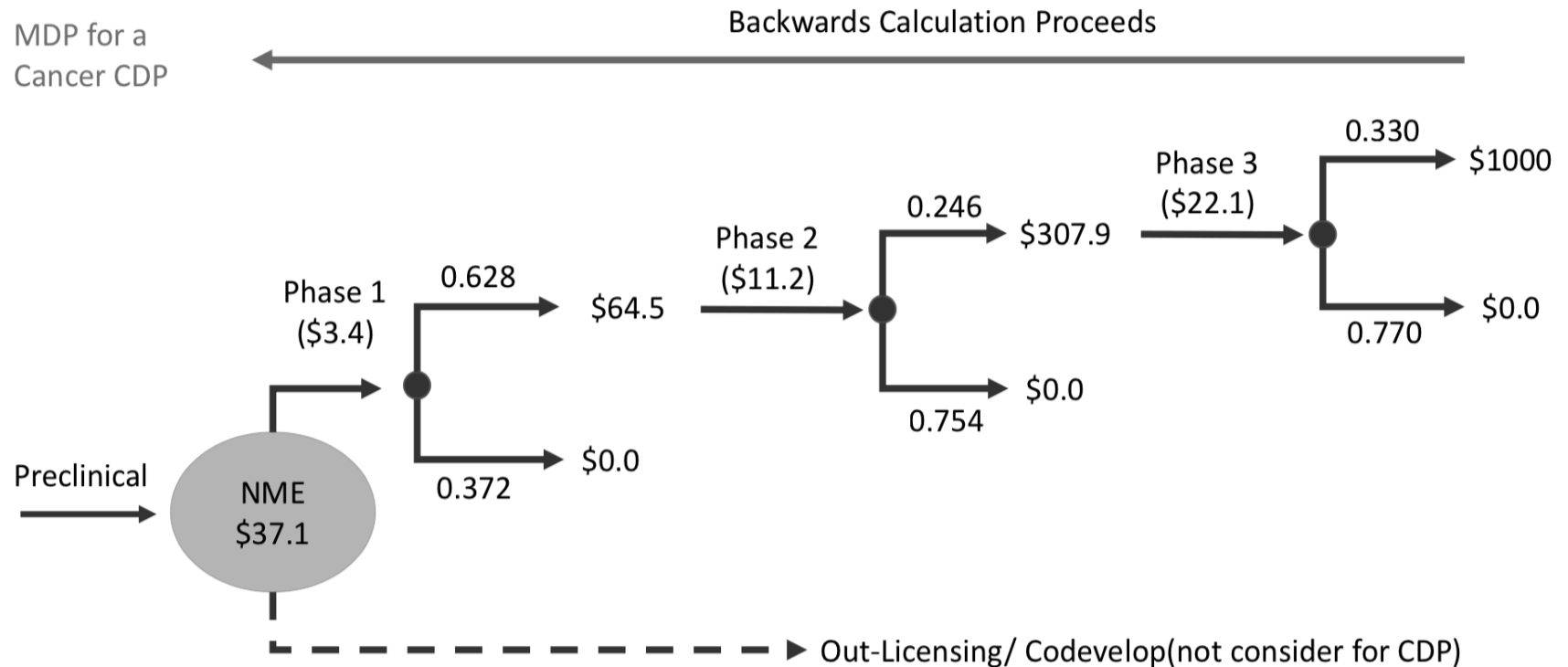
Source: Chang, 2010; Chang et al. 2019

# Sequential Decision Process (SDP)



Source: Chang, 2010; Chang et al. 2019

# Stochastic Decision Process For A Cancer Clinical Development Program (CDP)

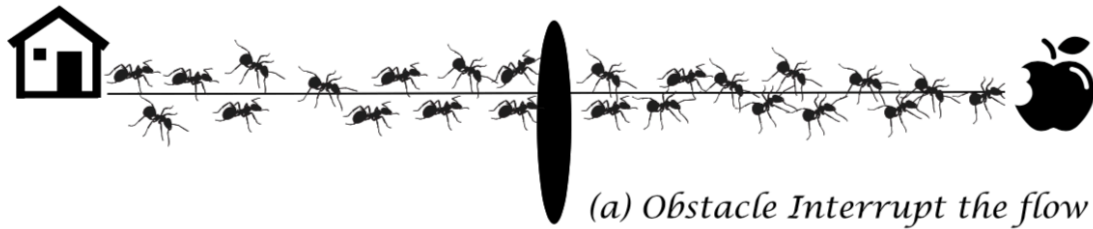


Source: Chang et al. 2019

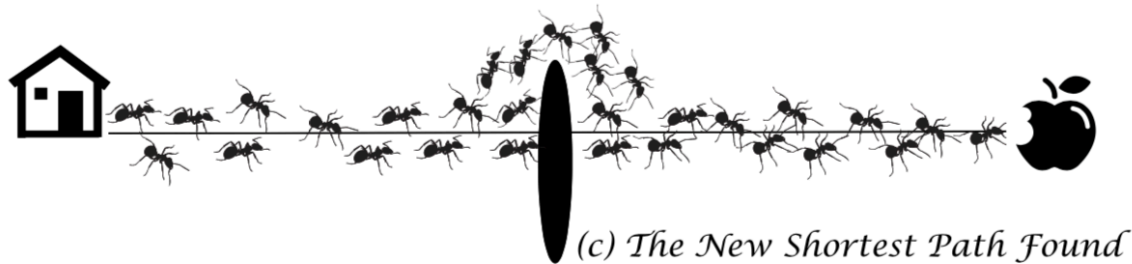
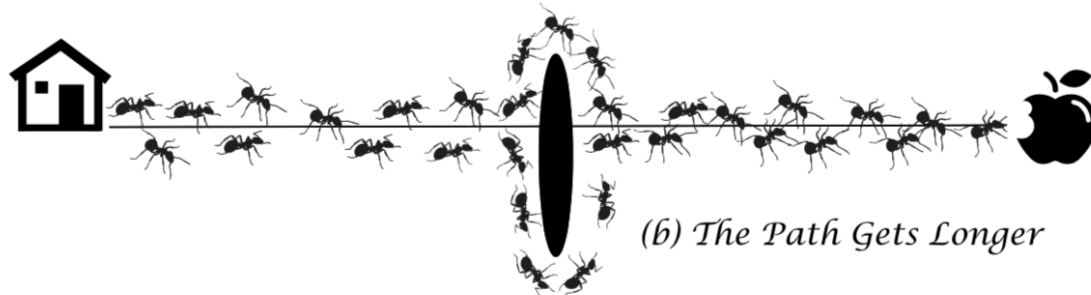
# Swarm Intelligence (SI)

- Self-Organized System (SOS)
  - SOS = Systems in which organized behavior arises without a centralized controller or leader
  - **Stigmergy** = a mechanism of indirect coordination between agents: the trace left in the environment by an action stimulates the performance of a next action, by a different agent.
- Swarm Intelligence (SI)
  - SI = intelligence possessed by SOS without common goal and leader
- Characteristics of SI
  - Micro motivated and macro consequence (genotype versus phenotype)
  - Each individual in SOS has no intelligence and follow a simple rule.
  - Ex: each ant simply follows the hormone, but the colony behaves intelligently in finding the shortest path to the food source
  - SI is not own by any individual
- SI Examples in Nature
  - Ant colonies, bird flocking, animal herding, bacterial growth, fish schooling, and microbial intelligence
- Artificial Swarm Intelligence Algorithms
  - Particle swarm, Ant colony, Artificial bee colony, Artificial immune systems,, Gravitational search, Glowworm swarm, and Bat algorithms

# How Swarm Intelligence Works



**Swarm Intelligence:**  
Every ant follow the  
smell, collectively the  
ant colony quickly find  
the shortest path  
without the guidance of  
any central intelligence



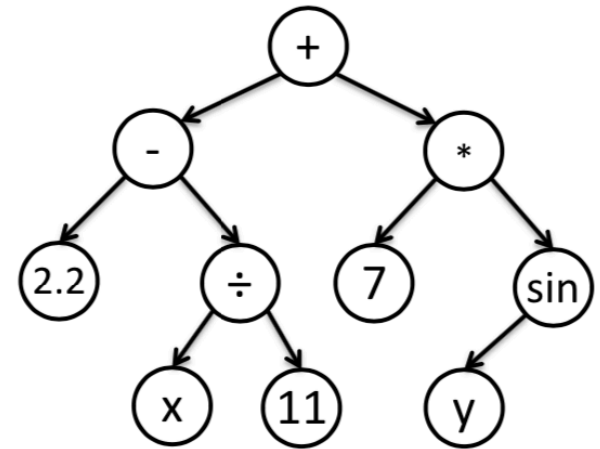
# Swarm Intelligence Applications

- A central part of the rational drug development process is the prediction of the complex structure of a small ligand with a protein, the so-called protein-ligand docking problem, used in virtual screening of large databases and lead optimization.
- Korb et al (2006) developed a docking algorithm based on ant colony optimization, to **structure-based drug design**.
- Fu, et al. (2015) studied a new approach for flexible **molecular docking** based on swarm intelligence. They compute the interactions of 23 protein-ligand complexes.
- Rajeshkumar and Kousalya (2017) present a view on applications of swarm-based intelligence algorithms in pharmaceutical industry, including **drug design**, **pharmacovigilance**, and **alignment of sequence**.
- Soulami, et. Al. (2017) used a particle swarm optimization (PSO) based algorithm for **detection and classification of abnormalities in mammographic images** using texture features and support vector machine (SVM) classifiers.
- Fang et. al. (2018) presented a novel feature selection called the elite search mechanism-based flower pollination algorithm (ESFPA) used to determine protein essentiality. ESFPA uses an improved SI algorithm for feature selection and selects optimal features for **protein essentiality prediction**.
- The metaheuristicOpt package in R includes many optimization algorithms

# Evolutionary Intelligence (EI)

- **Inspiration by Natural Evolution Process**
  - Inspired by our understanding of biological evolution
- **Minimal or No Specification of Solution Structure**
  - Automatically solves problems without requiring the user to know or specify the form or structure of the solution in advance.
- **Solution Improvement Via Evolution**
  - Generation by generation, stochastically transform populations of computer programs into new populations of programs that will effectively solve problems.
- **Genetic Algorithm (GA) versus Genetic Programming (GP)**
  - Both use evolution algorithms
  - GA is represented as a list of actions and values, often a string
  - GP is represented as a tree structure of actions and values, usually a nested data structure.

# Genetic Programming



$$\left(2.2 - \left(\frac{x}{11}\right)\right) + (7 * \sin(y))$$

## Three Key Elements in GP:

### (1) Representation: Syntax Tree

Mathematical or compound structure

### (2) Reproductive Mechanism: Crossovers and Mutations

Crossover = combination of two randomly selected data structures to produce one new structure.

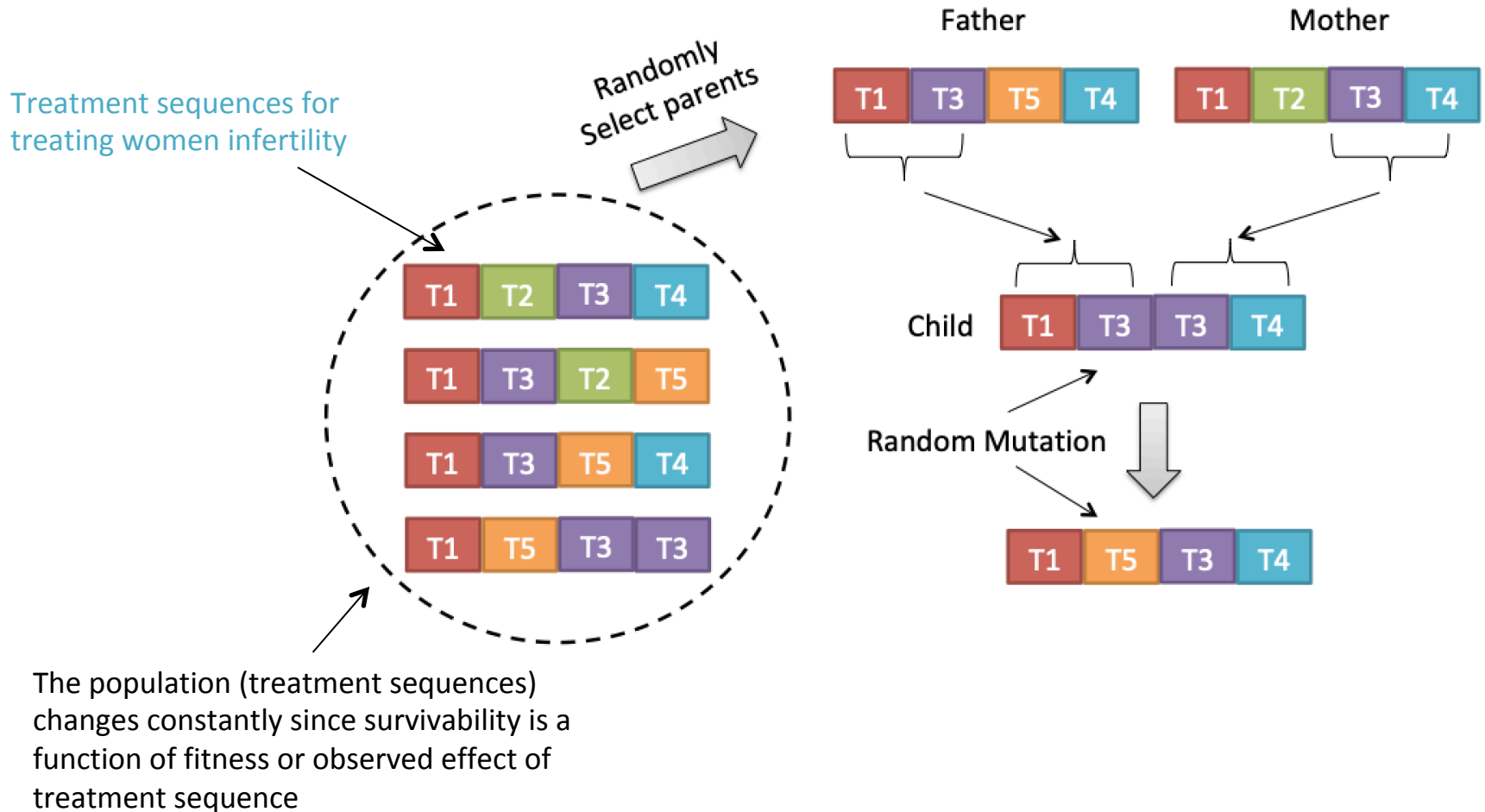
A mutation = randomly generates a subtree to replace a randomly selected subtree from a randomly selected individual.

### (3) Survival Fitness

For finding a function  $g(x)$  to approximate the target function  $f(x)$ , the fitness = the mean square error between the two functions.

For finding a optimal treatment sequence for infertility, the fitness function = treatment success (1 for success and 0 for failure).

# Genetic Algorithm For Treatment Strategy Optimization



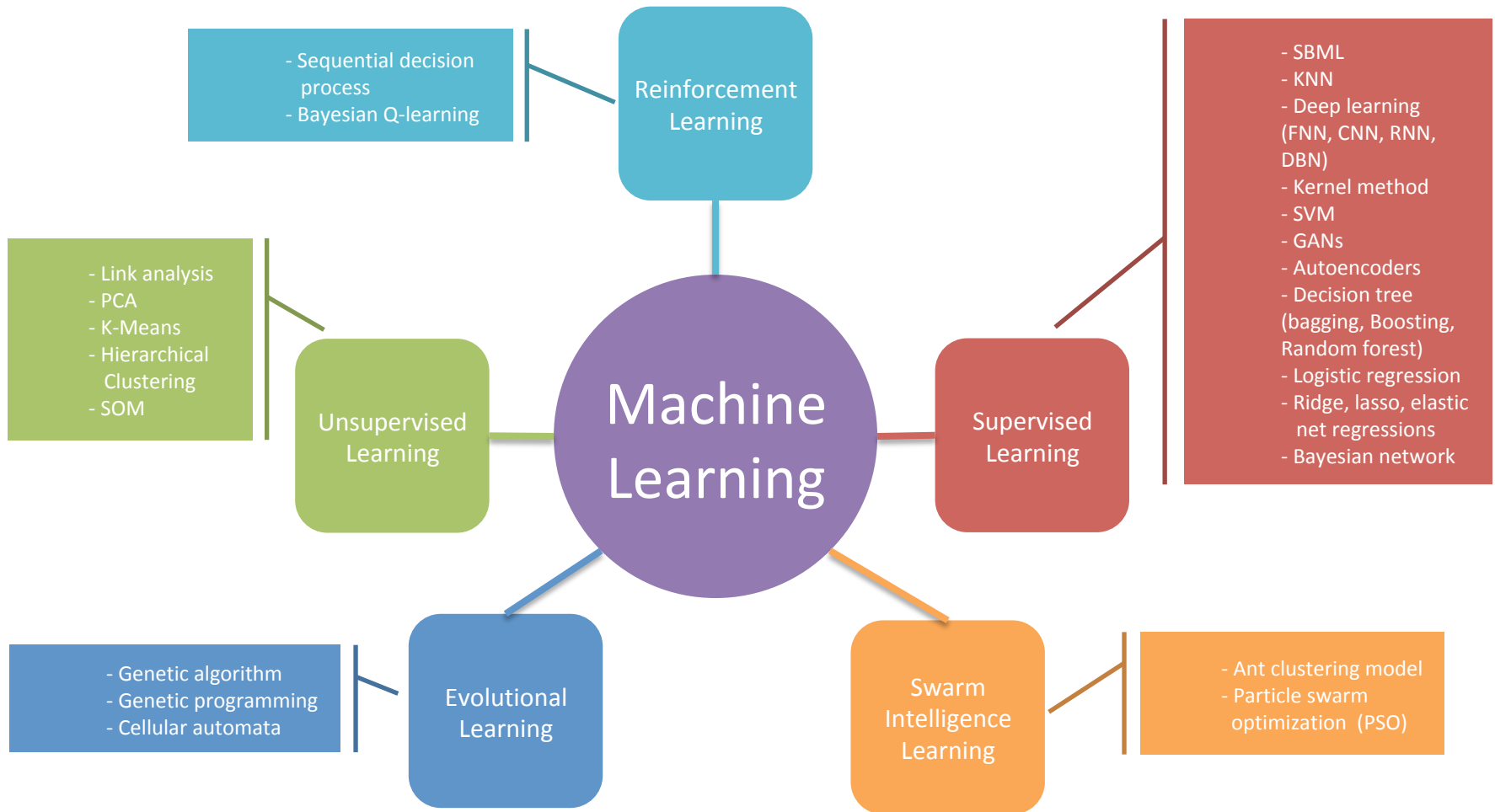
# Application Of Evolutionary Intelligence

- RNA structure prediction (Batenburg et al,1995)
- Molecular structure optimization (Wong, et al. 2011)
- Ghosh and Jain (2005) collected articles across a broad range of topics on the applications of evolutionary AI in drug discovery, GP in data mining for drug discovery
- Barmpalex, et. al. (2011) has used symbolic regression via genetic programming in the optimization of a controlled release pharmaceutical formulation and compared its predictive performance to ANNs.
- Ghaheri,et. al. (2015) reviewed the applications of the genetic algorithm in disease screening, diagnosis, treatment planning, pharmacovigilance, prognosis, and health care management.
- Ghaheri et al (2019) provided an comprehensive review of the applications of genetic algorithms in medicine, in 14 different areas:
  - radiology, oncology, cardiology, endocrinology, obstetrics and gynecology, pediatrics, surgery, infectious diseases, pulmonology, radiotherapy, rehabilitation medicine, orthopedics, neurology, pharmacotherapy, and health care management.

# Review & Summary

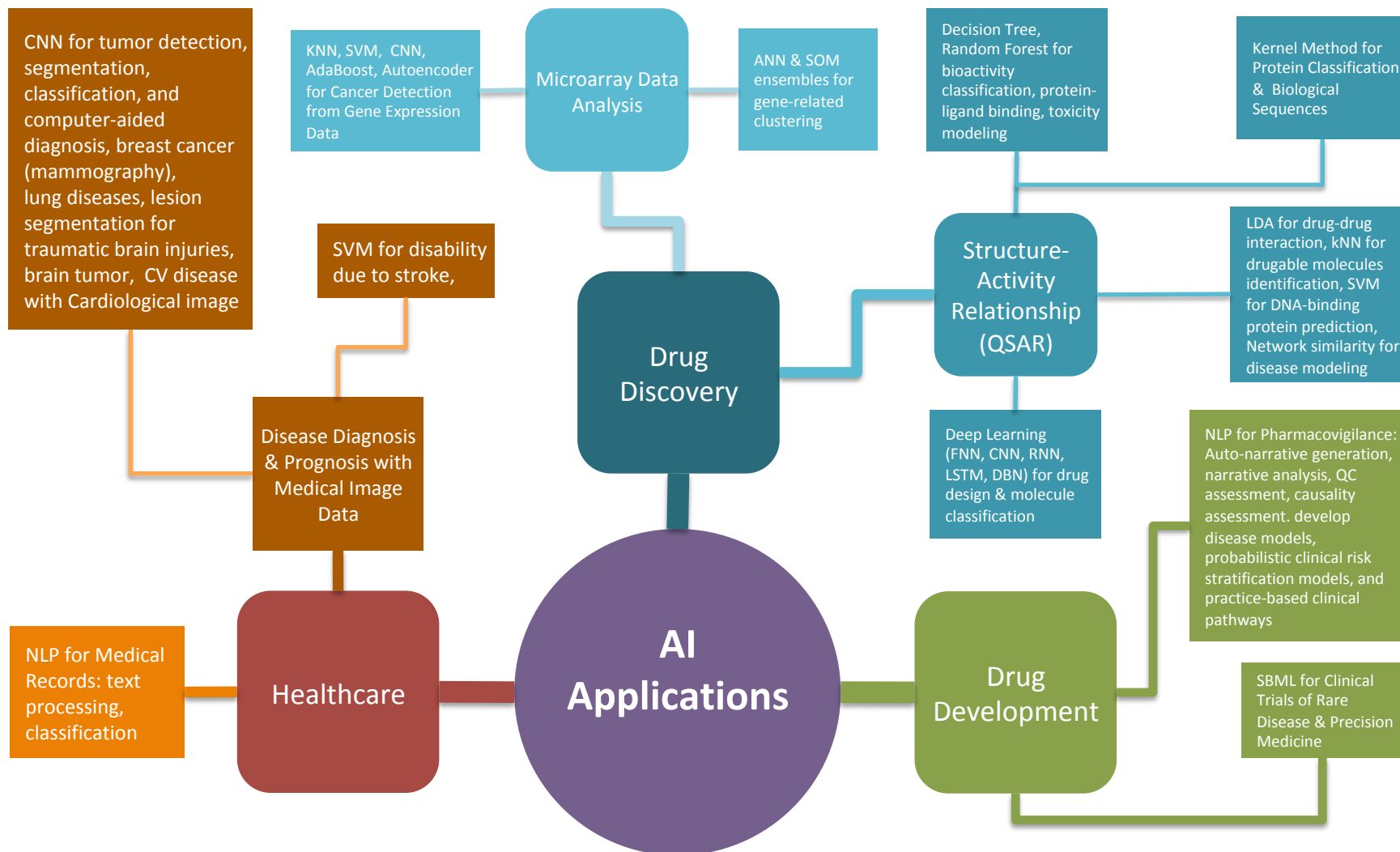
# List Of Machine Learning Methods

Details and References Available from the Upcoming Book (Chang, 2020)



# List Of AI Applications

Details and References Available from Upcoming Book (Chang, Feb 2020)



# References

## Books:

- Chang, M. (2019). Artificial Intelligence in Drug Development, Precision Medicine, and Healthcare. CRC
- Chang, M. (2010). Monte Carlo Simulation for the pharmaceutical industry. CRC
- Chang, M. (2013). Principles of Scientific Methods. CRC
- Chang, M.(2007, 2014). Adaptive Design Theory and Implementation Using SAS and R. CRC
- Chang, M. (2011). Modern Issues and Methods in Biostatistics, Springer.
- Chang, M. (2012). Paradoxes in Scientific Inference. Chang, M. (2014). Principles of Scientific Methods. CRC
- Chang M. et al. (2019). Innovative Strategies, Statistical Solutions and Simulations for Modern Clinical Trials. CRC
- Chow and Chang (2011). Adaptive Design Methods for Clinical Trials. CRC
- Hastie, T., Tibshirani, R., Friedman, J. (2001, 2nd Edition, 2009). The Elements of Statistical Learning. Springer
- Koza, J. et al (2005). Genetic Programming VI. Routine human-competitive machine intelligence. Springer
- Ben, G & Pennachin, C. (1998). Artificial General Intelligence. Springer
- Russell S & Norving (2003). Artificial intelligence – a modern approach. Prentice Hall.
- Bishop. C. (2006). Pattern recognition and Machine learning. Springer

## Review Papers:

- Daouda, M. and Mayo, M. (2019). A Survey Of Neural Network-based Cancer Prediction Models From Microarray Data. Artificial Intelligence In Medicine 97, 204—214
- Jiang, F. et al (2017). Artificial intelligence in healthcare: past, present and future. Stroke and Vascular Neurology
- Long, E., Lin, H., et al (2017). An artificial intelligence platform for the multihospital collaborative management of congenital cataracts.
- Qayyuma, A. et al. (2018.) Medical Image Analysis using Convolutional Neural Networks: A Review, Journal of Medical Systems November 2018, 42:226
- Vanhaelen, Q. et al. (2017). Design of efficient computational workflows for in silico drug repurposing. Drug Discovery Today Volume 22, Number 2 February 2017
- Schmider , J. Artificial Intelligence in Adverse Event Case Processing. Clin. Pharmco & Therapeutics. April 2019
- Ghasemi, F. (2018). Neural network and deep-learning algorithms used in QSAR studies: merits and drawbacks. Drug Discovery Today Volume 23, Number 10 October 2018

# Similarity-Based Machine Learning in R (1)

```
library(class)
# Normalize data (by subtracting Mean and Dividing by SD of the RefData)
normalizeData = function(DataToNormalize, RefData){
  normalizedData = DataToNormalize
  for (n in 1:ncol(DataToNormalize)){
    normalizedData[,n] = (DataToNormalize[,n] - mean(RefData[,n])) / sd(RefData[,n])
  }
  return(normalizedData)
}
# Calculate initial scaling factors R0s # S0s = p-values
InitialRs = function(X, eta, S0s) {
  K = length(S0s); R0s = rep(0, K)
  for (k in 1:K) {
    # force <12 to avoid s=exp(d) overflow
    R0s[k] = min((-log(S0s[k])) ^ (1/eta) / max(IQR(X[,k]), 0.0000001), 12)
  }
  return(R0s);
}
Distance = function(Rs, x1, x2) {
  # 3 vectors: Rs = scaling factors, x1 = point 1, x2 = point 2
  K = length(Rs)
  d = 0; for (k in 1:K) { d = d + (Rs[k]* (x2[k]-x1[k]))^2 }
  return (d^0.5)
}
# Calculate Similarity Score.
SimilarityTrain = function(eta, Rs, Xtrain) {
  N1 = nrow(Xtrain); K = length(Rs);
  S = matrix(0, nrow=N1, ncol=N1)
  for (i in 1:N1) { for (j in i:N1) {
    d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k]* (Xtrain[i,k]-Xtrain[j,k]))^2 }
    S[i,j] = exp(-d2^0.5) # avoid 0 for i=j
    S[j,i]=S[i,j] # Use symmetry to reduce 50% CPU time.
  } } # End j and i loops
  return(S)
}
# Calculate Similarity Scores between training and test subjects.
Similarity = function(eta, Rs, Xtrain, Xtest) {
  N1 = nrow(Xtrain); N2=nrow(Xtest); K = length(Rs)
  S = matrix(0, nrow=N2, ncol=N1)
  for (i in 1:N2) { for (j in 1:N1) {
    d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k]* (Xtest[i,k]-Xtrain[j,k]))^2 }
    S[i,j] = exp(-d2^0.5)
  } } # End j and i loops
  return(S)
}
```

```
Weight = function(S) {
  N1= nrow(S); N2=ncol(S);
  W = matrix(0, nrow=N1, ncol=N2)
  for (i in 1:N1) {
    sum_S_row = sum(S[i, ])
    for (j in 1:N2) { W[i,j] = S[i,j] / max(sum_S_row,0.0000000001) }
  }
  return(W)
} # End of Weight
PredictedY = function(W, Ytrain, Ytest) {
  # Calculate predicted outcome and Error
  OutObj = list()
  OutObj$pred_Y = W %%% Ytrain # For binary outcome, pred_y = prob of 1.
  OutObj$MSE = mean((OutObj$pred_Y - Ytest)^2)
  return(OutObj)
} # End of PredictionY
DerivativeE = function(eta, pred_Y, Rs, X, S, O) {
  # Derivatives if the loss function
  N = nrow(X); K=length(Rs)
  der_S = matrix(0, nrow=K*N, ncol=N); der_W = matrix(0, nrow=K*N, ncol=N)
  dist = matrix(0, nrow=N, ncol=N); der_E = rep(0, K)
  for (i in 1:N) { for (j in i:N) {
    d2 = 0; for (k in 1:K) { d2 = d2 + (Rs[k]* (X[i,k]-X[j,k]))^2 }
    dist[i,j] = max(d2^0.5, 0.0000001) # avoid overflow in der_d below
    dist[j,i]=dist[i,j]
  } } # End of i and j loops
  for (m in 1:K) { for (i in 1:N) { for (j in i:N) {
    der_d = (Rs[m] / dist[i,j]) * (X[i,m]-X[j,m])^2
    der_S[(m-1)*N+i, j] = -1 * S[i,j] * eta * (dist[i,j])^(eta-1) * der_d
    der_S[(m-1)*N+j, i] = der_S[(m-1)*N+i, j]
  } } } # End of j, i, and m loops
  # Weight Derivative
  for (m in 1:K) { for (i in 1:N) {
    sum_der_S = sum(der_S[(m-1)*N+i, :]); sumSi = sum(S[i, :])
    for (j in i:N) {
      der_W[(m-1)*N+i, j] = der_S[(m-1)*N+i, j] / sumSi - S[i,j]* sum_der_S / sumSi^2
      der_W[(m-1)*N+j, i] = der_W[(m-1)*N+i, j]
    } } } # End of j, i, and m loops
  # Derivatives of E
  for (m in 1:K) { for (i in 1:N) {
    err = 2/N * (pred_Y[i] - O[i]) # For mean squared error
    for (j in 1:N) { der_E[m] = der_E[m] + err * O[j] * der_W[(m-1)*N+i, j] }
  } } # End of i and m loops
  return(der_E)
} # End of DerivativeE
```

## Similarity-Based Machine Learning in R (2)

```
Learning = function (LearningRate, Lamda, Rs, der_E) {
  # Update scaling factors, Rs.
  K=length(Rs)
  der_lossFun = der_E+2*Lamda*Rs # derivatives of loss function
  Rs = Rs - LearningRate * der_lossFun
  # force Rs<25 to avoid S=exp(d) overflow
  for (m in 1:length(Rs)) { Rs[m] = min(max(0, Rs[m]), 25) }
  return (Rs)
} # End of Learning

# Training AI to obtain scaling factors, Rs ###
SBMLtrain = function(Epoch, S0, Lamda, LearningRate, eta, Xtrain, Ytrain) {
  TrainObj=list(); OutObj0= list(); OutObj= list()
  R0 = InitialRs(Xtrain, eta, S0);
  S = SimilarityTrain(eta, R0, Xtrain)
  OutObj0 = PredictedY (Weight(S), Ytrain, Ytrain)
  Rs=R0; OutObj =OutObj0 # in case Epoch=0 for no learning
  TrainMSE0 = OutObj0$MSE
  preLoss = OutObj0$MSE+Lamda*sum(Rs^2)
  # Learning
  iter=0;
  while (iter<Epoch) {
    preRs=Rs
    Rs = Learning(LearningRate, Lamda, Rs, DerivativeE(eta, OutObj0$pred_Y, Rs, Xtrain, S, Ytrain))
    OutObj = PredictedY (Weight(SimilarityTrain (eta, Rs, Xtrain)), Ytrain, Ytrain)
    iter=iter+1
    Loss = OutObj$MSE+Lamda*sum(Rs^2)
    if (Loss>preLoss) {Rs=preRs; iter=Epoch+1}
    preLoss=Loss
  }
  TrainObj$Rs = Rs; TrainObj$Y = OutObj$pred_Y; TrainObj$MSE = OutObj$MSE
  return( TrainObj)
}

# Linear regression
# outcome = "gaussian" or "binomial", ...
GLMPv = function(Y, X, outcome) {
  LMfull=glm(Y ~., data=X, family=outcome)
  sumLM=coef(summary(LMfull))
  pVals=sumLM[,4][1:ncol(X)+1]
  return (pVals)
}
```

```
# Example with Simulated Multivariate Normal Data
# covariance matrix for creating correlated X.
library(mvtnorm)
sigma = matrix(c(1, 0.0, 0.6, 0.2, 0.0, 0.0, 0.0,
                 0.0, 1, 0.0, 0.0, 0.0, 0.0, 0.0,
                 0.6, 0.0, 1, 0.5, 0.0, 0.0, 0.0,
                 0.2, 0.0, 0.5, 1, 0.0, 0.0, 0.0,
                 0.0, 0.0, 0.0, 0.0, 1, 0.0, 0.0,
                 0.0, 0.0, 0.0, 0.0, 0.0, 1, 0.0,
                 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1), ncol=7)

mu=c(0,0,0,0,0,0,0) # 7 multivariate normal means
initS = rep(0.5, 7) # 7 initial similarities.
xTrain = rmvnorm(n=40, mean=mu, sigma=sigma)
xTest = rmvnorm(n=40, mean=mu, sigma=sigma)
# Construct y that are not linearly related to x
Ytrain=xTrain[,1]+xTrain[,2]*xTrain[,3]+xTrain[,4]^2+xTrain[,5]
Ytest=xTest[,1]+xTest[,2]*xTest[,3]+xTest[,4]^2+xTest[,5]
# Data normalization
Xtrain = as.data.frame(normalizeData(xTrain, xTrain))
Xtest = as.data.frame(normalizeData(xTest, xTrain))
# Assign p-value from LM to initial similarities
# initS = GLMPv(Y=Ytrain, X=Xtrain, outcome = "gaussian")

mySBMLtrain=SBMLtrain(Epoch=5, S0=initS, Lamda=-0.5, LearningRate=0.125,
eta=1, Xtrain, Ytrain)
myPred=PredictedY(Weight(Similarity(eta=1, mySBMLtrain$Rs, Xtrain,
Xtest)),Ytrain, Ytest)
# MSE using training mean for prediction
plot(myPred$pred_Y, Ytest)
mse1=mean((Ytest-mean(Ytrain))^2) # naive mse
mse2=myPred$MSE # Probability error
avemse1=avemse1+mse1/50
avemse2=avemse2+mse2/50
}
avemse1; avemse2
```

# Acknowledgment

- Dr. Susan Hwang, Boston University