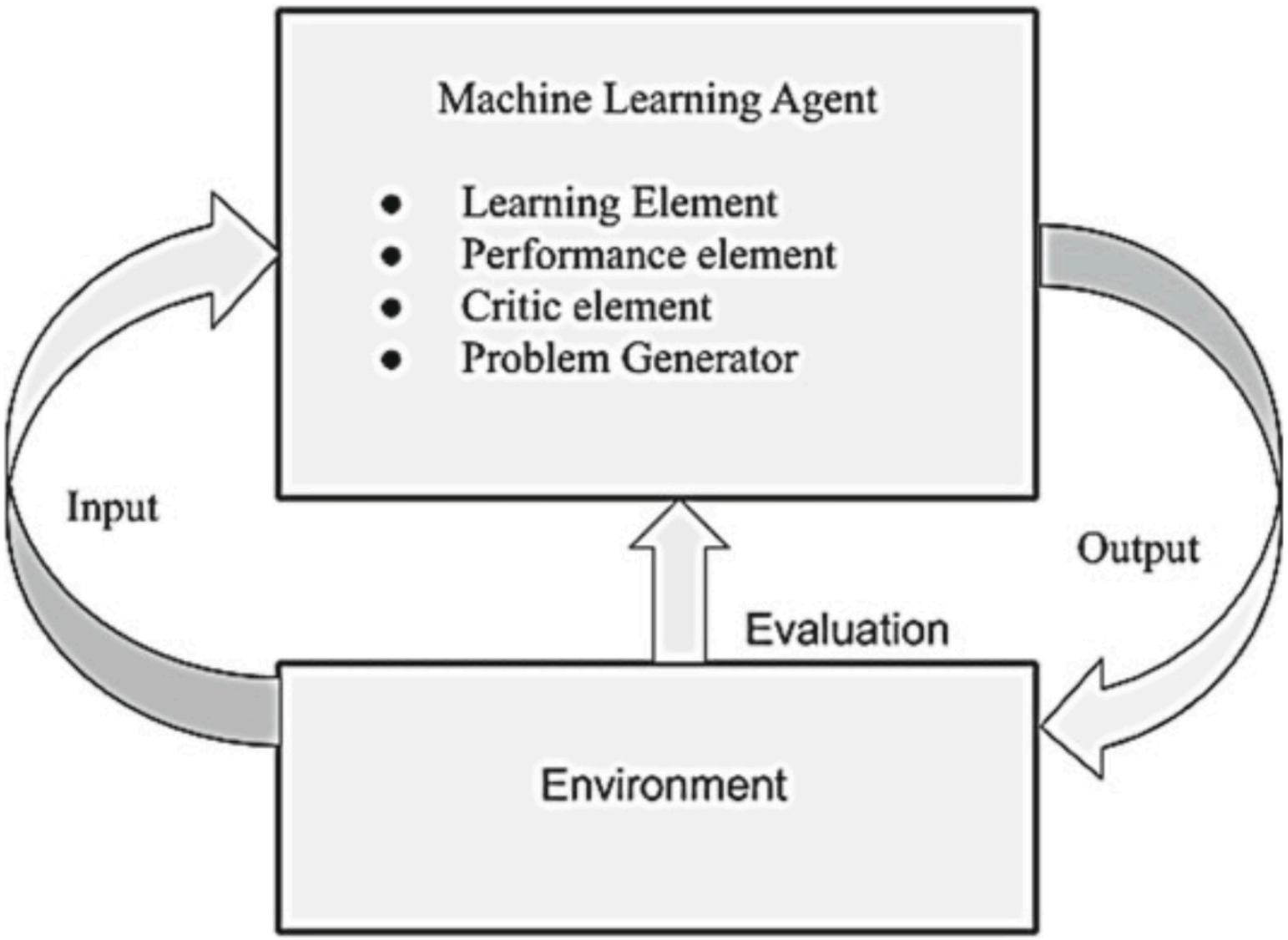
# Machine Learning Algorithms in Bioinformatics

Bahman Moraffah
School of Electrical, Computer, and Energy Engineering
Arizona State University

*https://bmoraffa.github.io/*

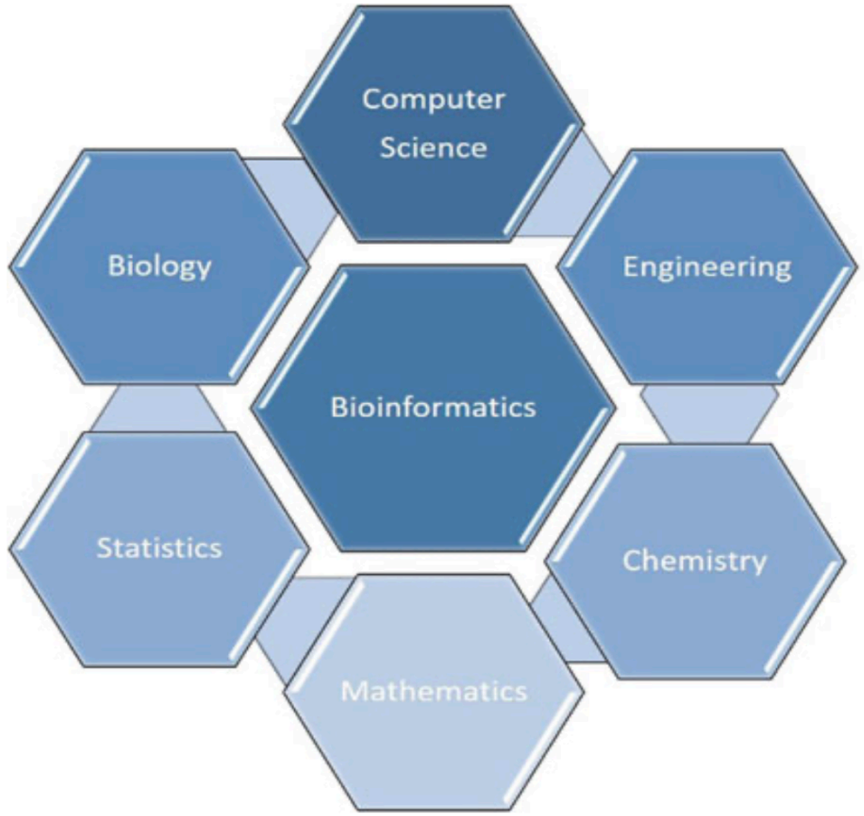
# What Is Machine Learning?



# What Is Bioinformatics?

**Role of Machine Learning in Bioinformatics?**

- Bioinformatics as a hybrid science that connects biological data with analytical advanced technique to withdraw meaningful information for various scientific research including biomedicine.
- In Bioinformatics, data is collected, stored, manipulated, in addition, this includes modeling of data for analysis, data visualization and foresight by the deployment of algorithms and software.
- Bioinformatics uses computation to get relevant information from biological data through different methods to explore, analyze, manage and store data.
- It is mainly used for the identification of genes and nucleotides for a better understanding of disease based on genes.
- Classification of gene sequence has an important role to understand the principle within nucleic acid and protein sequence.



# Some Problems In Bioinformatics

## DNA Structure and Biological Sequence:

- Predicting the protein structure
- Analysis of structures of protein for marks of a functioning
- Alignment of structures
- Biological networks such as gene regulatory network, protein-protein interaction network requires an extensive amount of care
- Prediction and classifications of genes
- Gene finding: Finding introns and exons in DNA-sequence segment
- Genome annotation: analyzing the repetitive DNA which is copied from the same or nearly the same sequence within the genome

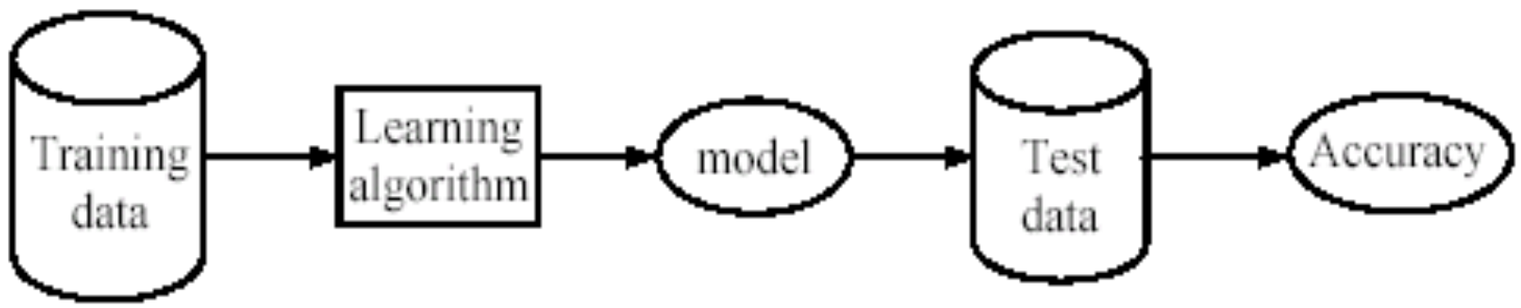
# Supervised Learning Vs Unsupervised Learning

- Supervised learning: classification is seen as supervised learning from examples.
  - Supervision: The data (observations, measurements, etc.) are labeled with pre-defined classes. It is like that a “teacher” gives the classes (supervision).
  - Test data are classified into these classes too.

- Unsupervised learning (clustering)
  - Class labels of the data are unknown
  - Given a set of data, the task is to establish the existence of classes or clusters in the data

# Supervised Learning

- Data: A set of data records (also called examples, instances or cases) described by
  - k attributes: $A_1$, $A_2$, … $A_k$.
  - a class: Each example is labelled with a pre-defined class.
- Goal: To learn a classification model from the data that can be used to predict the classes of new (future, or test) cases/instances.
- Learn: Learn a classification model from the data



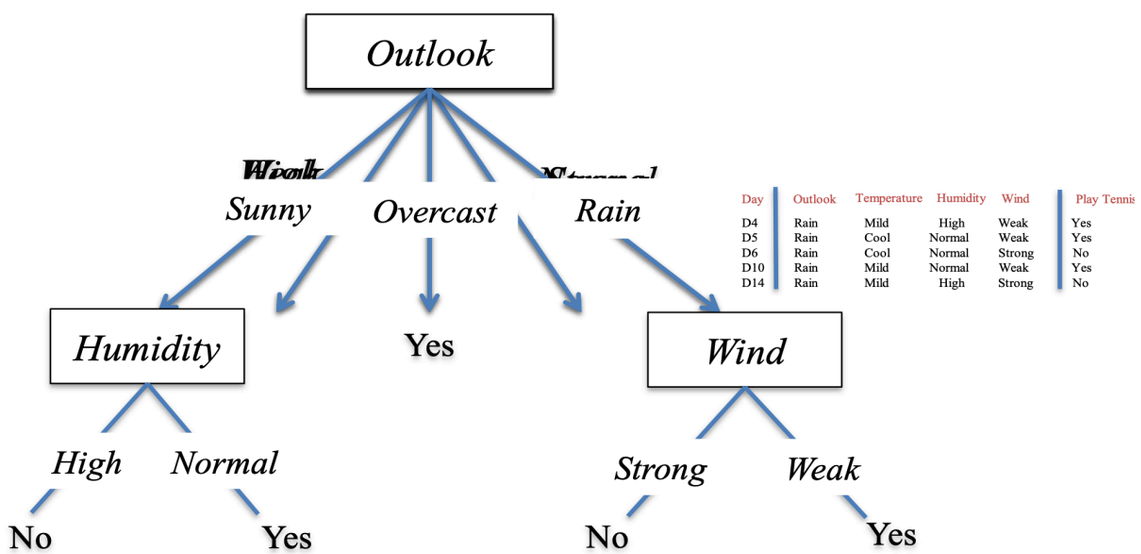
# Supervised Learning Methods

- Decision tree
- Naïve Bayesian classification
- Support vector machines (SVD)
- K-nearest neighbor
- Ensemble Methods (Boosting and Bagging)

# Supervised Classification: Decision Tree Learning

- Decision tree learning is one of the most widely used techniques for classification.
  - It is very efficient
- Algorithm:
  - Pick "best" attribute to split at the root based on training data.
  - Recurse on children that are impure (e.g. have both Yes and No).
  - If training examples perfectly classified, Then STOP, Else iterate over new leaf nodes.

What is the best? Information gain/ entropy




| Day | Outlook | Temperature | Humidity | Wind | Play Tennis |
|---|---|---|---|---|---|
| D4 | Rain | Mild | High | Weak | Yes |
| D5 | Rain | Cool | Normal | Weak | Yes |
| D6 | Rain | Cool | Normal | Strong | No |
| D10 | Rain | Mild | Normal | Weak | Yes |
| D14 | Rain | Mild | High | Strong | No |

# Top-Down Decision Tree Learning

- Basic algorithm (a greedy divide-and-conquer algorithm)
  - Assume attributes are categorical now (continuous attributes can be handled too)
  - Tree is constructed in a top-down recursive manner
  - At start, all the training examples are at the root
  - Examples are partitioned recursively based on selected attributes
  - Attributes are selected on the basis of an impurity function (e.g., information gain, Gini index)
- Conditions for stopping partitioning
  - All examples for a given node belong to the same class
  - There are no remaining attributes for further partitioning – majority class is the leaf
  - There are no examples left

# How to Avoid Overfitting in Decision Tree

- Overfitting: A tree may overfit the training data
  - Good accuracy on training data but poor on test data
  - Symptoms: tree too deep and too many branches, some may reflect anomalies due to noise or outliers
- Approaches to avoid overfitting
  - Pre-pruning: Halt tree construction early
    - Difficult to decide because we do not know what may happen subsequently if we keep growing the tree.
  - Post-pruning: Remove branches or sub-trees from a “fully grown” tree.
    - A validation set may be used for pruning as well.

# Naïve Bayesian classification

- Bayesian Method in classification

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}$$

- Posterior distribution $p(Y|X)$ provides the decision rule. Given a test point

$$y^* = argmax_y p(y|X)$$

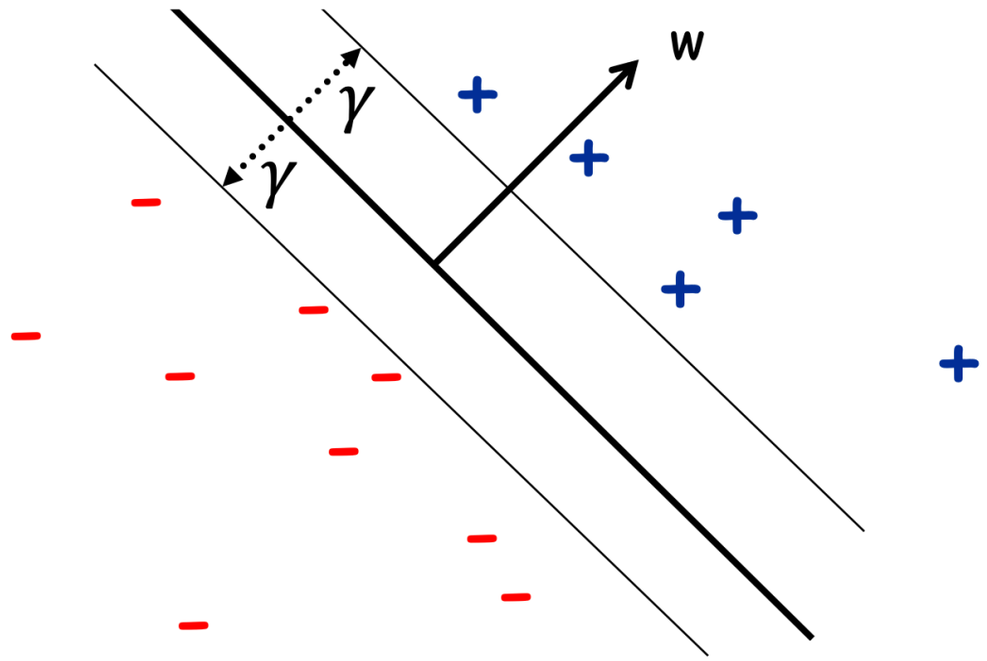
- Naïve Bayes assumption:

$$p(x_1, \ldots, x_n|Y) = \prod_i p(x_i|Y)$$

**Bayes, Thomas (1763)** An essay towards solving a problem in the doctrine of chances. *Philosophical Transactions of the Royal Society of London,* **53:370-418**

# Support Vector Machine (SVM)

- SVMs are linear classifiers that find a hyperplane to separate two class of data, positive and negative.
- Kernel tricks are used for nonlinear separation.
- SVM not only has a rigorous theoretical foundation, but also performs classification more accurately than most other methods in applications, especially for high dimensional data.




[Vapnik 1990]

# Support Vector Machines (SVMs)

- Looking to optimize for the maximum margin separator

Algorithm:

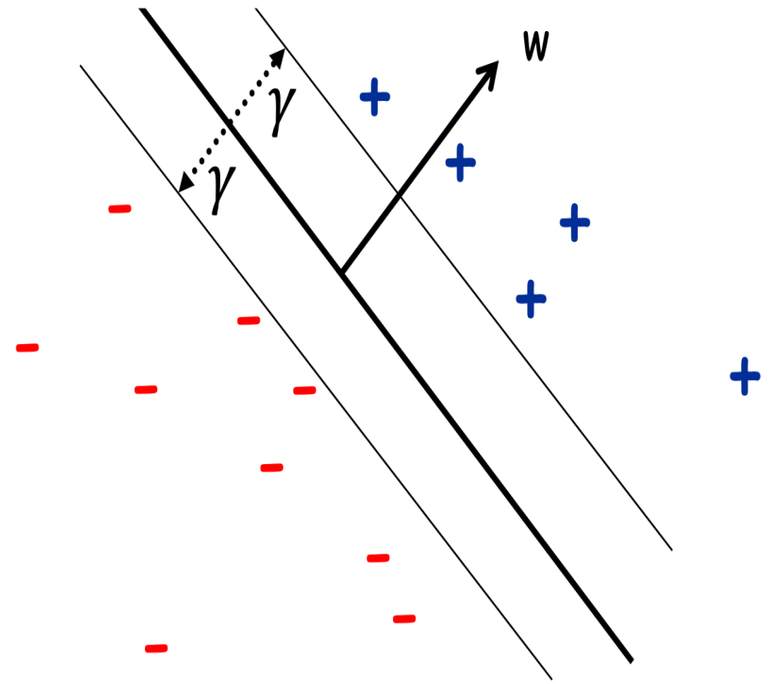
Input: $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^{N}$

Maximize $\gamma$ under the constraints:

$$||w||^2 = 1$$

$$y_i w.x_i \geq 0 \quad \forall i$$

Non-convex constraint




$$\min_{w} ||w||^2$$

$$\forall i, y_i w.x_i \geq 1$$

(quadratic programming)

[Vapnik 1990]

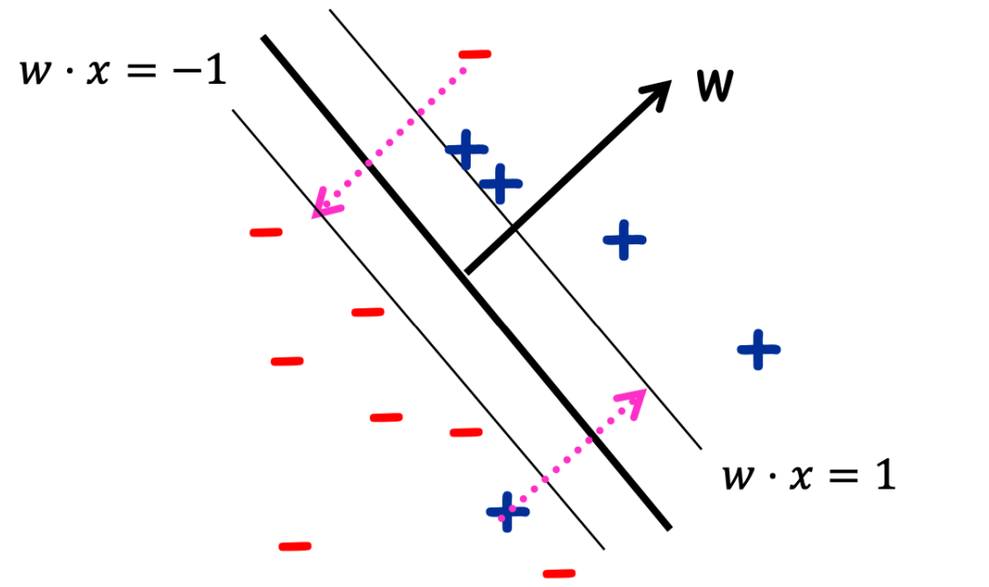
# Support Vector Machines (SVMs)

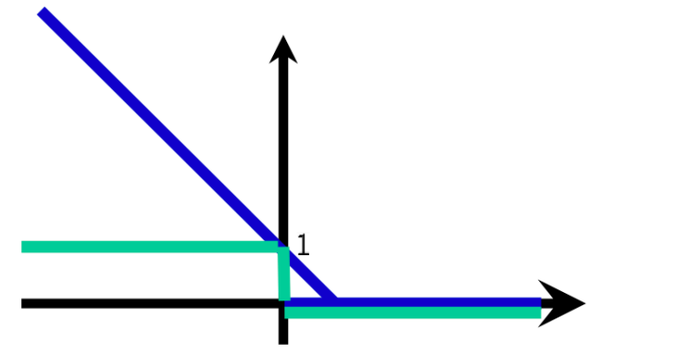
- Question: what if data isn't perfectly linearly separable? Replace "# mistakes" with upper bound called "hinge loss"

$$\min_{w,\xi_1,\ldots,\xi_N} ||w||^2 + C\sum \xi_i$$

$$\forall i, y_i w.x_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$




C controls the relative weighting between the goals of making $||w||^2$ small (margin is large) and ensuring that most examples have functional margin ≥ 1.

Idea: Maximize the margin and minimize # of misclassification

# SVMs and Kernelization

$$\min_{w,\xi_1,\ldots,\xi_N} ||w||^2 + C\sum \xi_i$$

(Primal Form)

$$\forall i, y_i w.x_i \geq 1 - \xi_i$$

$$\xi_i \geq 0$$

Can be replaced by any kernel $K(x_i, x_j)$

$$\text{argmin}_\alpha \frac{1}{2}\sum_i\sum_j y_i y_j \alpha_i \alpha_j x_i.x_j - \sum_i \alpha_i$$

(Lagrangian Dual Form)

$$\forall i,\ 0 \leq \alpha_i \leq C_i$$

$$\sum_i y_i \alpha_i = 0$$

Final Classifier is $w = \sum_i \alpha_i y_i x_i$

The point $x_i$ for which $\alpha_i \neq 0$ are called support vectors.

# Applications of SVMs in Bioinformatics

Table I. *Summary of typical applications of SVM in cancer genomics*.

| Applications | SVM model* | Data | Cancer type | Ref |
|---|---|---|---|---|
| Classification/subtyping | Linear SVM | mRNA | Soft tissue sarcomas | 14 |
| | Linear SVM | Methylation | Leukemia | 17 |
| | SVM-RFE | Methylation | Multiple | 20 |
| | Meta-SVM | Multi-omics | Breast cancer | 28 |
| | Linear SVM | Protein | Multiple | 22 |
| | Linear SVM | Proteomics | Breast cancer | 23 |
| | Linear SVM | CNV | Bladder cancer | 24 |
| | Linear SVM | SNP | Breast cancer | 25 |
| Biomarker/signature | SVM-RFE | mRNA | Multiple | 38 |
| | NetSVM | Expression & interaction | Breast cancer | 41 |
| Drug discovery | | | | |
| Screen radiation protection | RBF-SVM | Normal cell culture | All | 46 |
| Identify novel drug targets | Linear SVM | Druggability data set | All | 48 |
| Assess target-ligand interactions | Reg-SVM | Structure-activity data sets | All | 49 |
| Identify drug target proteins | Biased SVM | A collected protein dataset | All | 51 |
| Anti/non-anticancer molecule classification | Linear SVM | Anti-, non-anticancer molecules | NCI-60 cells | 53 |
| Anticancer drug sensitivity prediction | Ensemble SVM | Cell multi omics | Cell lines | 55 |
| Predicting substrates of the cancer resistance | Linear SVM | BCRP substrates | Breast cancer | 56 |
| Driver gene discovery | | | | |
| Kinase mutation activation | Linear SVM | Kinase data set | All | 59 |
| Drivers *versus* passengers | Linear SVM | COSMIC | All | 62 |
| Gene interaction | | | | |
| | RBF-SVM | Interacting proteins (DIP) | All | 68 |

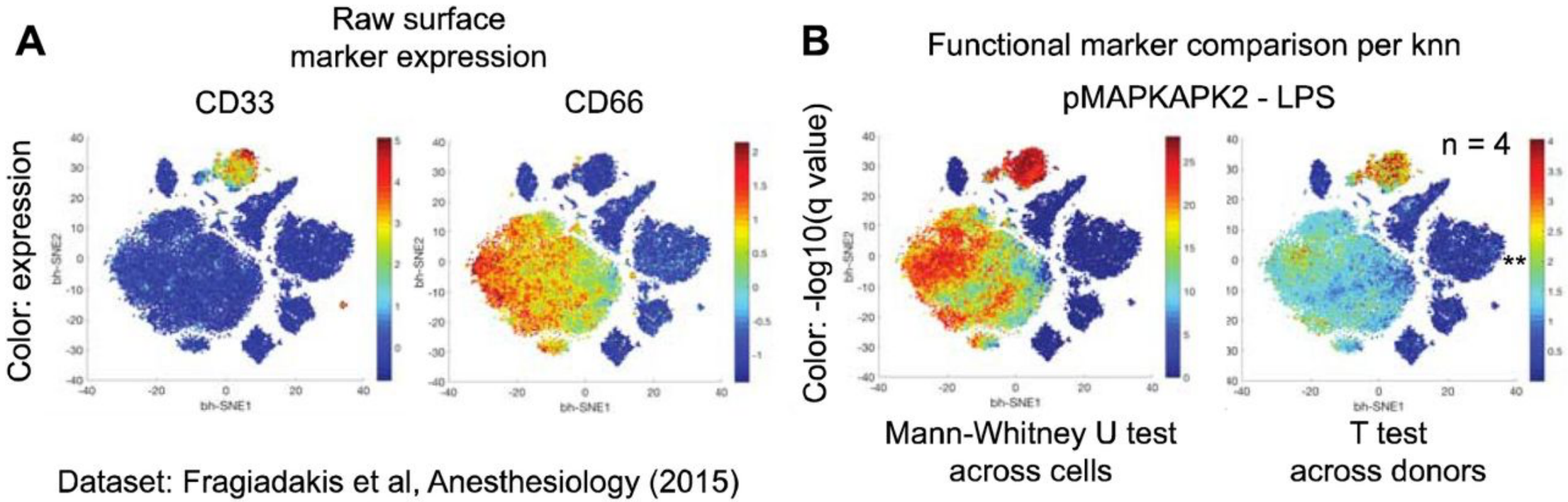
*SVM-RFE: SVM recursive feature elimination; RBF-SVM: radial basis function SVM; netSVM: network-constrained SVM; Reg-SVM: regression SVM; CNV: copy number variation; SNP: single nucleotide polymorphism; COSMIC: the catalogue of somatic mutations in cancer; BCRP: human breast cancer resistance protein.

[Huang 2018]

# K-Nearest Neighbor (KNN)

- Unlike all the previous learning methods, KNN does not build model from the training data.
- To classify a test instance x, define K-neighborhood P as k nearest neighbors of x
- Count number n of training instances in P that belong to class $c_j$
- Estimate $Pr(c_j|d)$ as n/k
- No training is needed. Classification time is linear in training set size for each test case
- This method suffers from curse of dimensionality
- K is usually chosen through validation set or cross-validation by trying a range of K values.
- Distance function is crucial but depends on applications.

# KNN to Visualize Differences between Biological Conditions




Dataset: Fragiadakis et al, Anesthesiology (2015)
Donors: healthy human
Cells: whole blood
Cell number: 40,000
Patient number: 4

Per-KNN statistical tests can reveal differences both across cells and across donors or replicates. (A) t-SNE maps colored by surface marker expression. (B) Per-cell Mann-Whitney U test or per-donor t test (n = 4) median values of pMAPKAPK2 expression between untreated and LPS-treated cells. **, $q < 0.01$.

[Burns 2020]

# Ensemble Methods

- Combining classifiers to get a better classifier
  - Boosting [Schapire, 1996]
    - Take weak classifiers and results in a strong classifier
    - Useful when there is high bias
    - Training for AdaBoost
      1. Produce a sequence of classifiers (weak)
      2. Each classifier is dependent on the previous one, and focuses on the previous one's errors
      3. Examples that are incorrectly predicted in previous classifiers are given higher weights
    - Test
      1. For a test case, the results of the series of classifiers are combined to determine the final class of the test case.

  - Bagging (Bootstrap Aggregating) [Breiman 1996]
    - Based on the idea of bootstrap (sampling with replacement)
    - Useful when high variance
    - Training
      1. Create K bootstrap samples from data D
      2. Train each data set and get the corresponding classifier
    - Test
      1. Classify each new instance by voting of the k classifiers (equal weights)

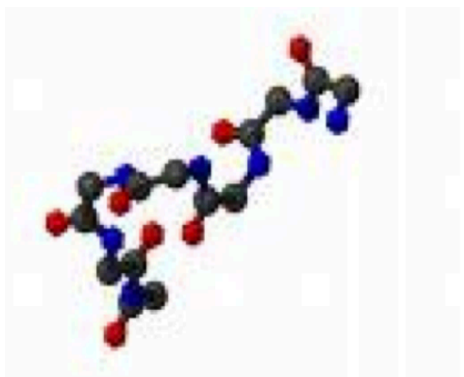
# Unsupervised Learning Methods

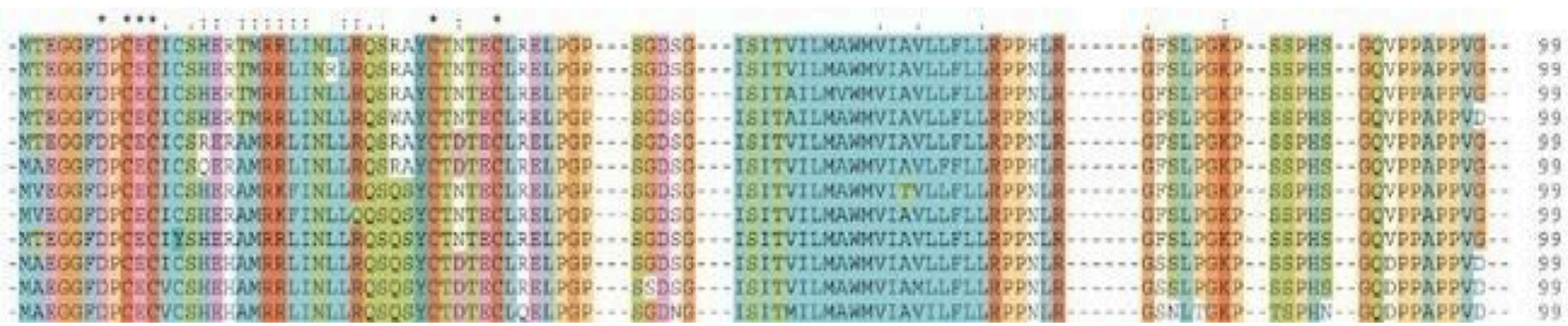
- Clustering
- Objective-based clustering
- Gaussian mixture model
- Hierarchical clustering
- Dimension reduction
- Principal component analysis (PCA)
- Factor analysis
- Probabilistic PCA (pPCA)

# Clustering

- Clustering is one main approach to unsupervised learning.
  - Automatically partitions unlabeled data into groups of similar datapoints
  - Useful for:
    1. Automatically organizing data.
    2. Understanding hidden structure in data.
    3. Preprocessing for further analysis e.g. dimension reduction

- Applications:
  - Cluster protein sequences by function or genes according to expression profile.





  - And many more applications

# Objective-based Clustering

- **Input:** Data and a distance/ dissimilarity function
- **Output:** A partition of data
- Objective-based clustering
  1. K-means
     - Find center points $c_1, \ldots, c_k$ to minimize

$$\sum_{i=1}^{N} \min_{j \in \{1,\ldots,k\}} d^2(x_i, c_j)$$

(Euclidean distance)
(d = 1, k=1 only easy to solve)

  2. K-median
     - Find center points $c_1, \ldots, c_k$ to minimize

$$\sum_{i=1}^{N} \min_{j \in \{1,\ldots,k\}} d(x_i, c_j)$$

  3. K-center
     - Find partitions to minimize the maximum radius.

# Lloyd's Method(Heuristic Method)

**Input:** Data points $x_1, \ldots, x_N \in \mathbb{R}^d$

**Initialize**: Center points: $c_1, \ldots, c_k \in \mathbb{R}^d$

Clusters: $C_1, \ldots, C_k$

**Repeat** till there is no drastic change in the cost

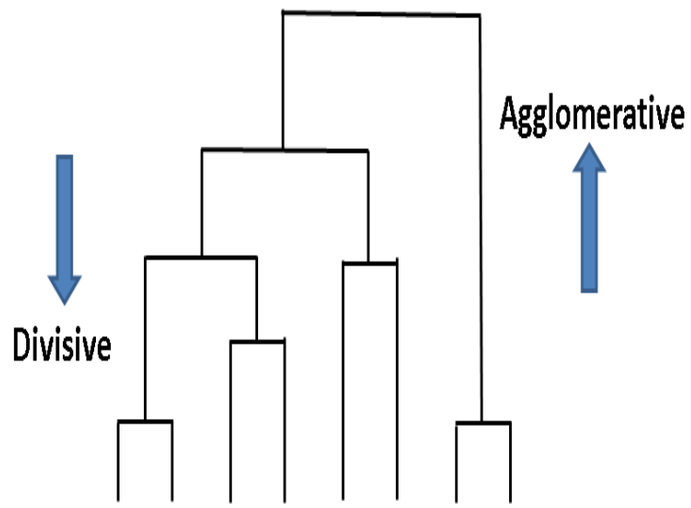
- For each $j : C_j \leftarrow \{x \in \mathcal{D} \text{ whose closet center is } c_j\}$
- For each $j : c_j \leftarrow \text{mean}(C_j)$

- Initialization is crucial:
  - Random centers from the datapoints
  - Furthest traversal
  - K-means ++ (works well and has mathematical guarantee)

[Lloyd 1982 ]

# Hierarchical Clustering

- Top-down (divisive, fragmentation)
  - Partition data into 2-groups (e.g., 2-means)
  - Recursively cluster each group.




- Bottom-Up (agglomerative, coalescent)
  - Start with every point in its own cluster.
  - Repeatedly merge the “closest” two clusters.
  - Different definitions of “closest” give different algorithms.
  - Have a distance measure on pairs of objects.
    - Single linkage: $dist(C, \tilde{C}) = \min_{x \in C, \tilde{x} \in \tilde{C}} dist(x, \tilde{x})$
    - Complete linkage: $dist(C, \tilde{C}) = \max_{x \in C, \tilde{x} \in \tilde{C}} dist(x, \tilde{x})$
    - Average linkage: $dist(C, \tilde{C}) = \text{average}_{x \in C, \tilde{x} \in \tilde{C}} dist(x, \tilde{x})$

# Dimension Reduction

- Powerful unsupervised learning techniques for extracting hidden (potentially lower dimensional) structure from high dimensional datasets
- Useful for:
  - Visualization
  - More efficient use of resources (e.g., time, memory, communication)
  - Statistical: fewer dimensions → better generalization Noise removal (improving data quality)
  - Further processing by machine learning algorithms

# Principal component analysis (PCA)

- Unsupervised technique for extracting variance structure from high dimensional datasets.
- PCA is an orthogonal projection or transformation of the data into a (possibly lower dimensional) subspace so that the variance of the projected data is maximized.
- PCA provides the directions along which data are most spread.

# Interpretation of PCA

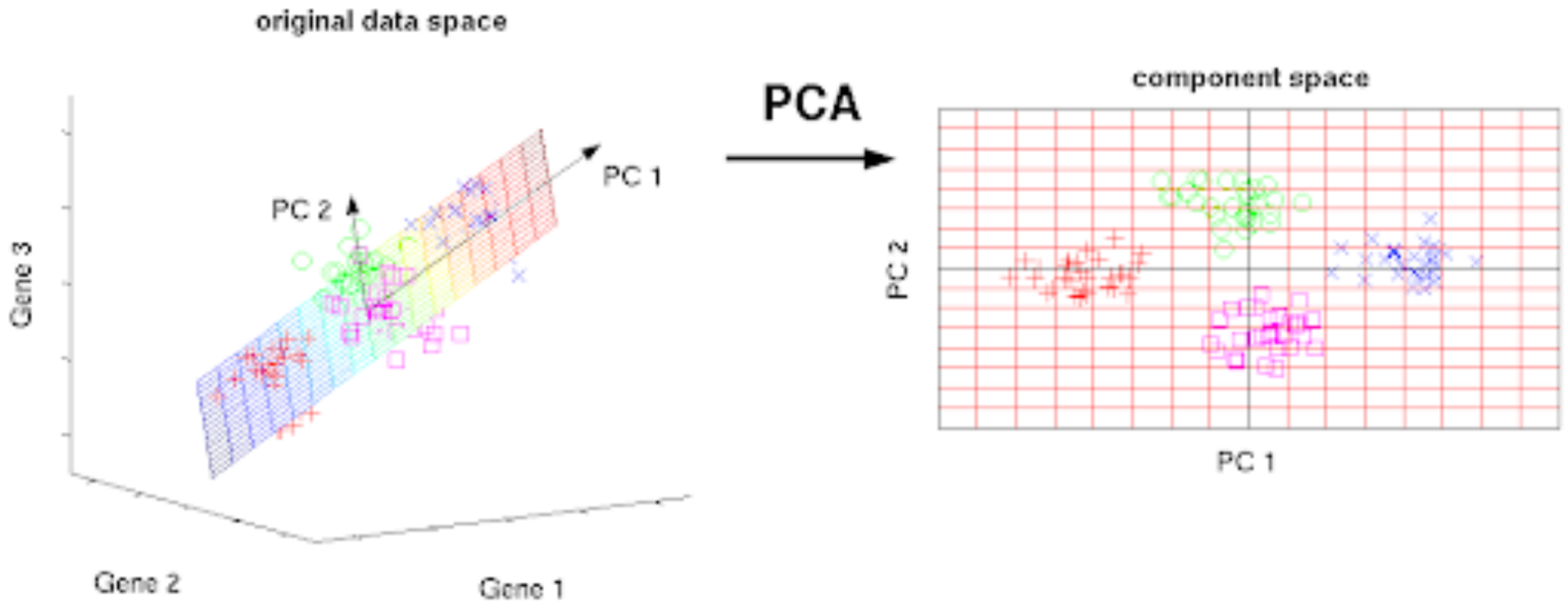
- Interpretation 1: Maximum Variance Subspace. PCA finds vectors v such that projections on to the vectors capture maximum variance in the data

$$\max_{v:||v||=1} v^T X X^T v$$

- Interpretation 2: Minimum Reconstruction Error. PCA finds vectors v such that projection on to the vectors yields minimum MSE reconstruction

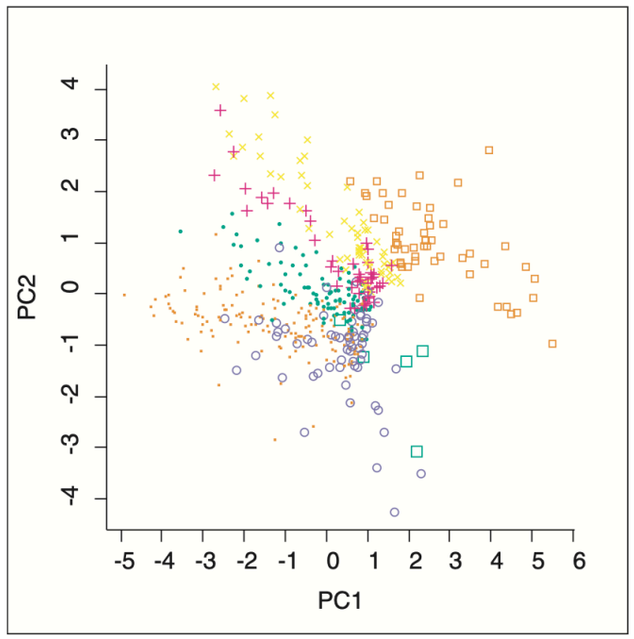
$$\min_{v:||v||=1} \sum_{i=1}^{N} ||x_i - (v^T x_i)v||^2$$

# PCA analysis for clustering gene expression data

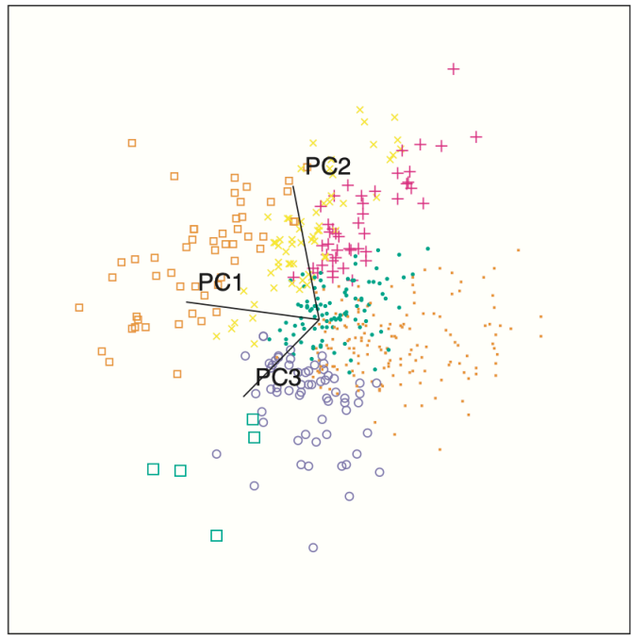



PCA Weakness:

- PCA only considers the second statistics
- PCA is only limited to linear projection




(a) In the subspace of the first 2 PC's




(b) In the subspace of the first 3 PC's

[Yeung, 2001]

# Factor Analysis (FA)

- Data Reduction: Factor analysis is a technique that is used to reduce a large number of variables into fewer numbers of factors (explain covariation between d variables using r < d latent factors), i.e., (exploratory FA)
- Maximum Common Variance: This technique extracts maximum common variance from all variables and puts them into a common score
- Theory Testing: Determine if hypothesized factor structure fits observed data (confirmatory FA)
- FA and PCA have the same theme however, they are different:
  - FA assumes a statistical model that describes covariation in observed variables via linear combinations of latent variables
  - PCA finds uncorrelated linear combinations of observed variables that explain maximal variance (no latent variables here)

FA refers to a statistical model, whereas PCA refers to the eigenvalue decomposition of a covariance (or correlation) matrix

# Factor Analysis

- Data $x_1, \ldots, x_N \in \mathbb{R}^d$
- Latent Structure $z_1, \ldots, z_N \in \mathbb{R}^r \ (r < d)$
- Probabilistic Modelling

$$z_i \sim \mathcal{N}(z_i | \mu_0, \Sigma_0)$$

$$x_i | z_i, W, \mu, \Psi \sim \mathcal{N}(W z_i + \mu, \Psi),$$

$$\text{where } W \in \mathbb{R}^{d \times r}, \Psi \in \mathbb{R}^{d \times d}, \mu \in \mathbb{R}^d$$

- Marginalizing out z,

$$x_i | W, \mu, \Psi \sim \mathcal{N}(W \mu_0 + \mu, \Psi + W \Sigma_0 W^T),$$

- Inference

$$argmax_{\{W, \mu, \Psi\}} \log p(X | W, \mu, \Psi)$$

EM Algorithm to Do Inference

# Probabilistic PCA (pPCA)

- Without loss of generality we can assume $z_i \sim \mathcal{N}(0, I)$ , then

$$x_i | W, \Psi \sim \mathcal{N}(\mu, \Psi + WW^T)$$

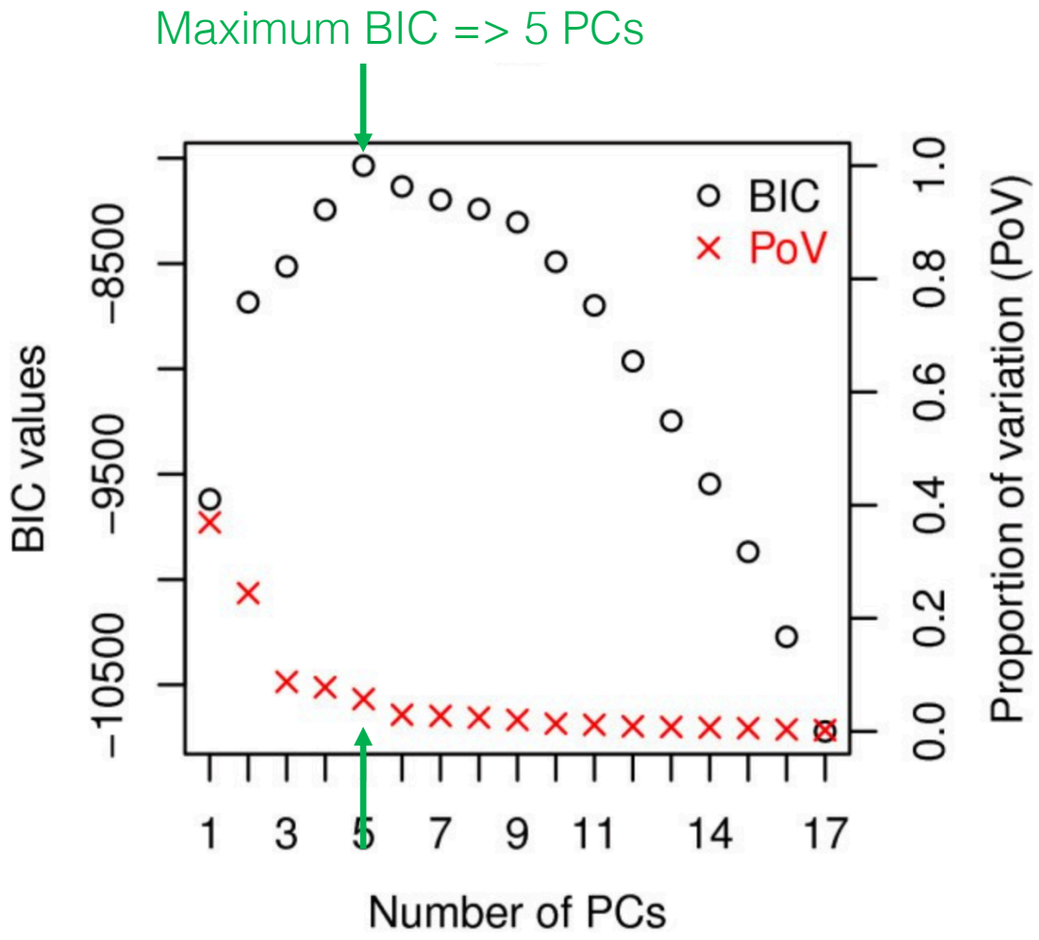
- pPCA is special case of Factor analysis
  - Assume $\Psi = \sigma^2 I$
  - Assume data is centered $\mu = 0$
- Inference can be done though Expectation-Maximization algorithm
- Observation: Assume centered data, If $\sigma \to 0$ , the solution for W gives back the same solution as PCA.

# pPCA for Metabolic Data

- To explore the effect of treatment with PTZ, a pPCA model was fitted to the urinary metabolomic data.

- Parameter estimation was achieved via the EM algorithm.

- A number of pPCA models with varying numbers of principal components was fitted; a modified Bayesian Information Criterion (BIC) is used to aid selection of the optimal model.

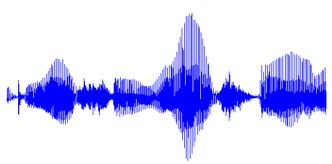



Fitting a pPCA model to the urine dataset. A. Plot of the modified BIC values and the proportion of variation explained by each model (different number of PCs)

[Nyamundanda 2010]

# What Do We Want a Machine to Do?

Label: “Motorcycle”
Suggest tags
Image search
…



Speech recognition
Speaker identification
Music classification
…

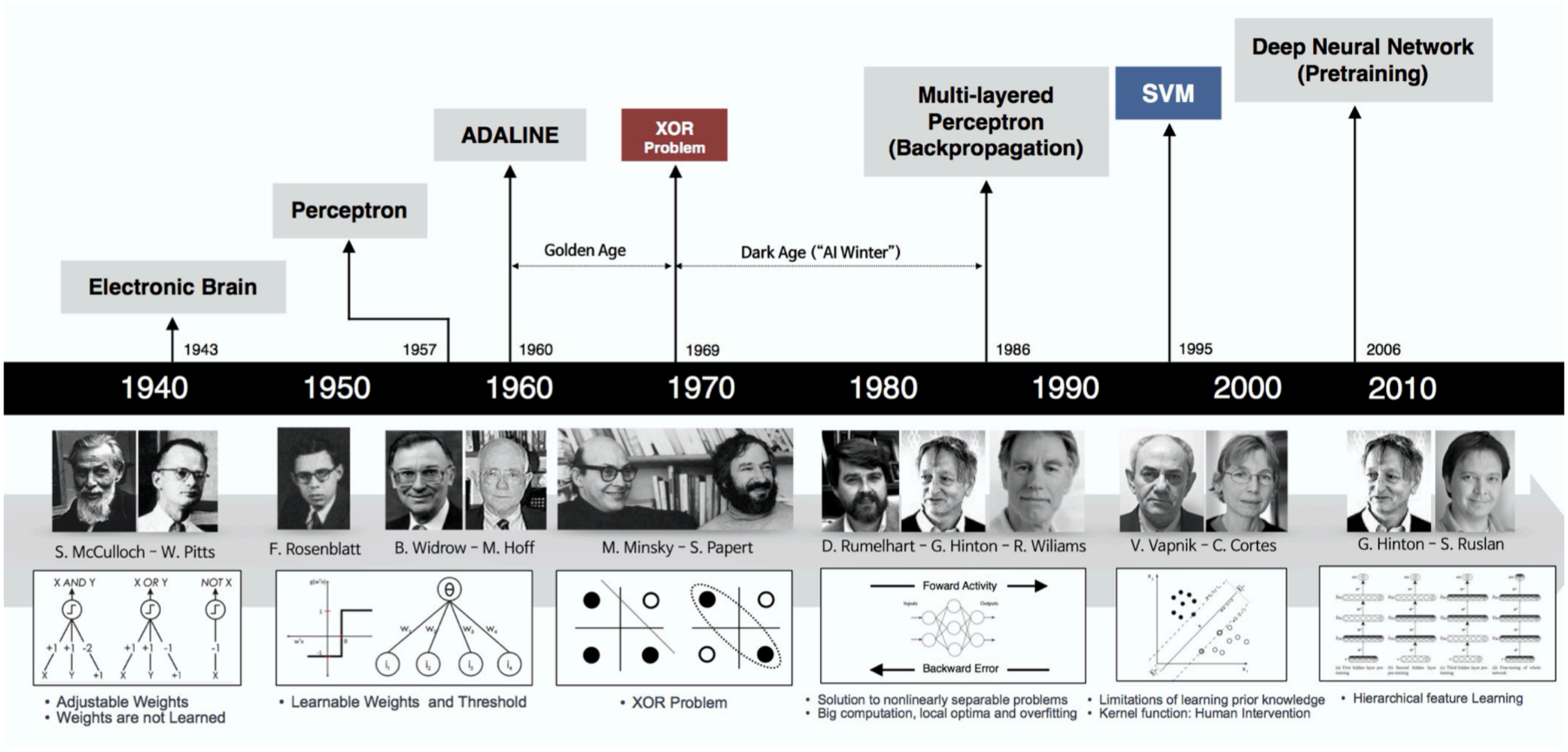

Web search
Anti-spam
Machine translation
…

Why are machine learning algorithms not enough for these problems?

# Deep Learning

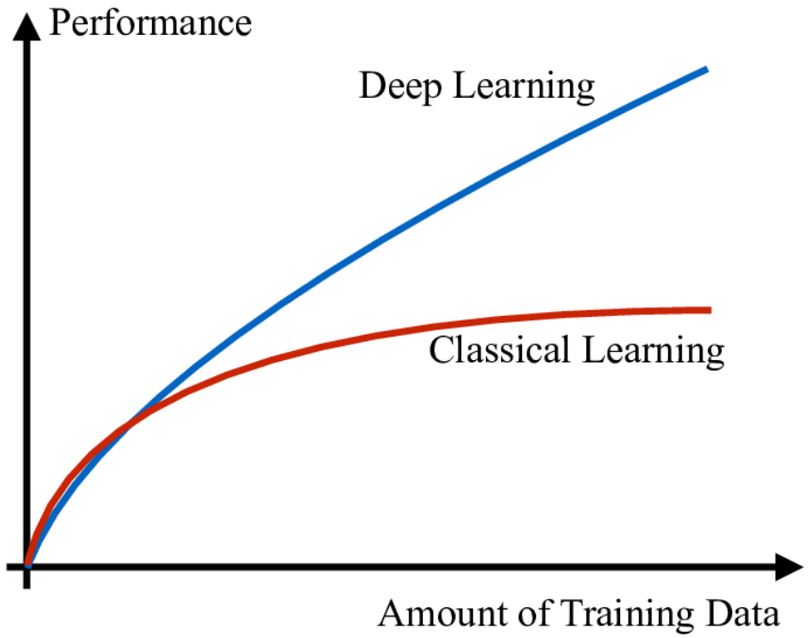
- History
- Deep learning in a nutshell
  - Neural networks (NNs)
  - Convolutional neural networks (CNNs)
  - Recurrent neural networks (RNNs)
- Deep learning in bioinformatics

# Neural Network




Slide credit: Mert Pilanci

# Classical Learning vs Deep Learning

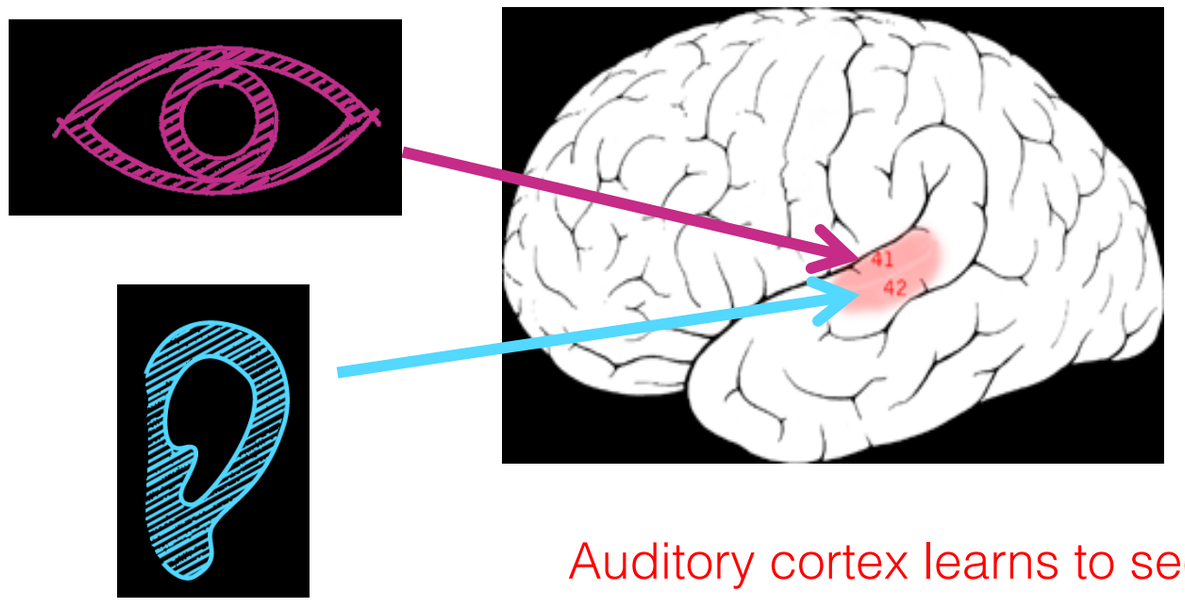



- Deep learning scales much better with more data than classical learning algorithms
- Deep learning have achieved accuracies that are far beyond that of classical learning methods in speech processing, natural language processing, computer vision, and reinforcement learning
- Classical learning algorithms often require feature engineering where there is no need for feature engineering in deep learning: features are earned through optimization

Is deep learning the solution for all the problems?

# Intro to Deep Learning

- Build learning algorithms that mimic the brain.
  - Biological systems built of very complex webs of interconnected neurons.
  - Highly connected to other neurons, and performs computations by combining signals from other neurons.
- Using brain simulations:
  - Make learning algorithms much better and easier to use.
  - Make advances in machine learning and AI.




Auditory cortex learns to see

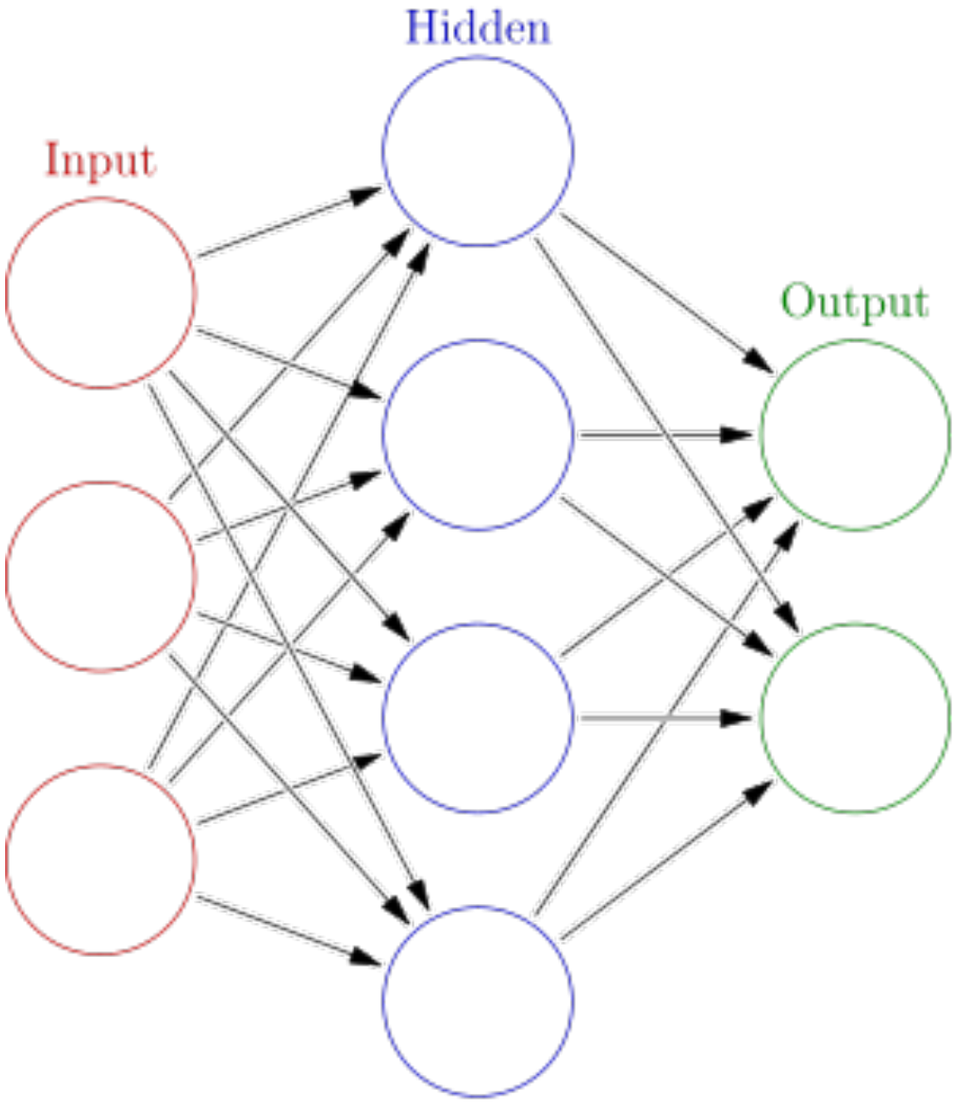
[Roe 1992]

# Main Idea of Deep Learning

- Linear classifier: $h(x) = W^T x$
- Kernelized classifiers: $h(x) = W^T \phi(x)$
- Idea: Use other functions instead of those well-crafted kernels, e.g. $\phi(x) = \sigma(Ax + b)$
- What is the benefit over kernel?
  - Scales linearly with data but for kernel is quadratic in number of data points
  - With more data kernel must be approximated

There is no function that can be learned by deep network but cannot be learned by shallow learners

# Neural Network




Weights

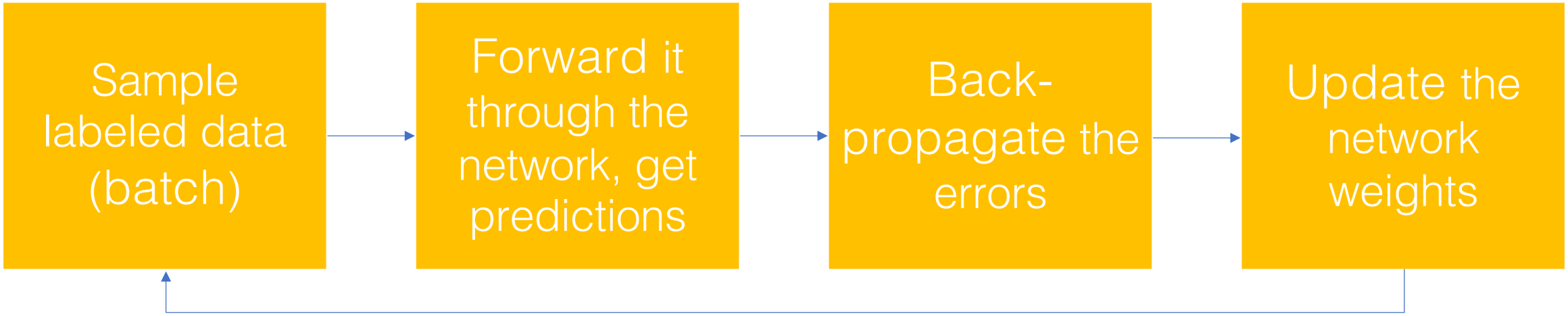
$$h = \sigma(W_1 x + b_1)$$

$$y = \sigma(W_2 h + b_2)$$

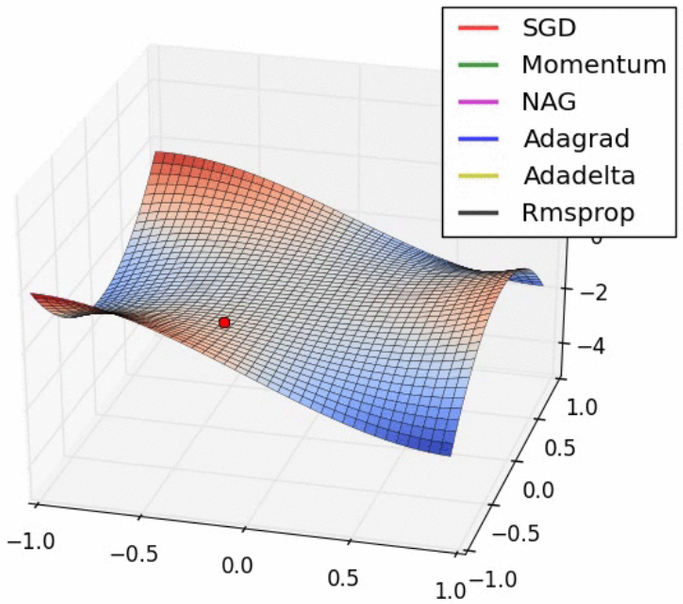
Activation functions

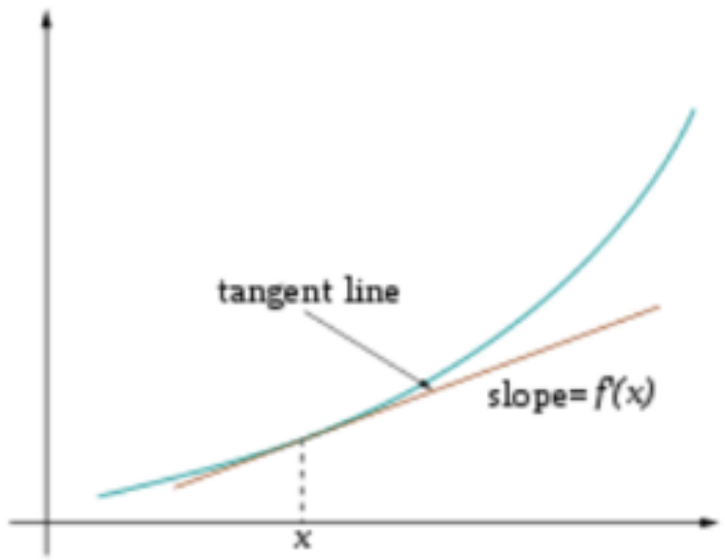
How do we train?

# Training




Optimize (min. or max.) objective/cost function $J(\theta)$
Generate error signal that measures difference between predictions and target values

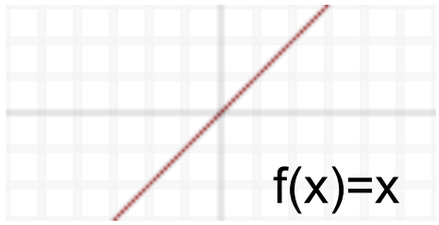


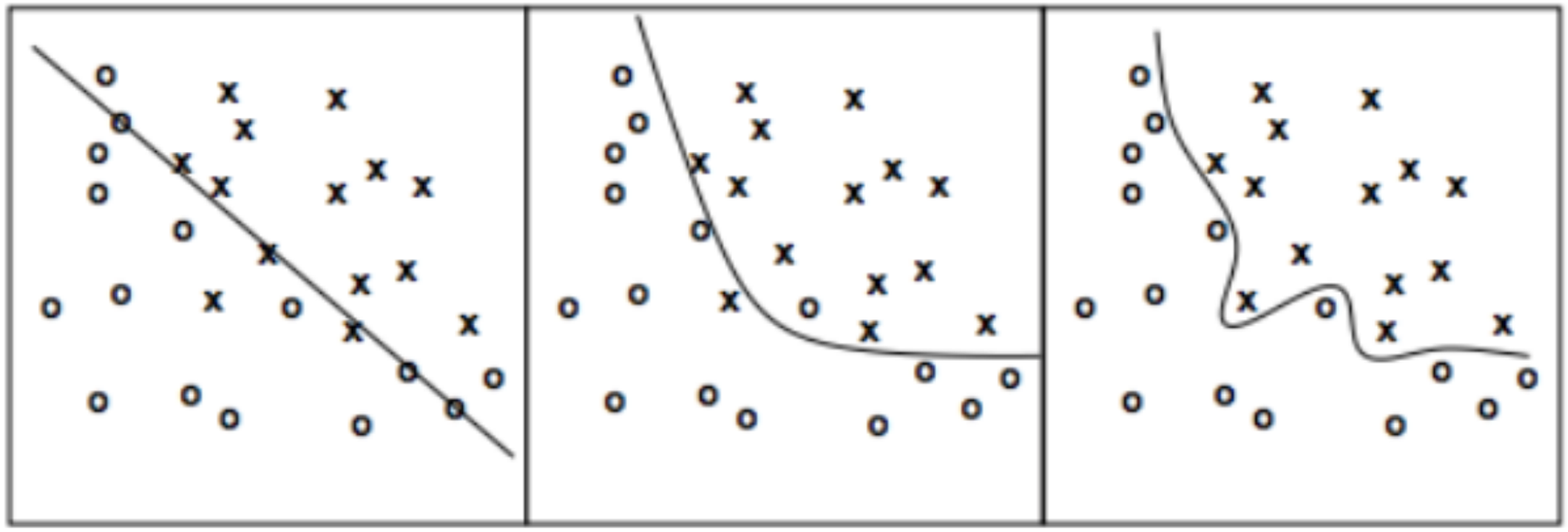

Use error signal to change the weights and get more accurate predictions
Subtracting a fraction of the gradient moves you towards the (local) minimum of the cost function

[https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39]

# Loss Functions

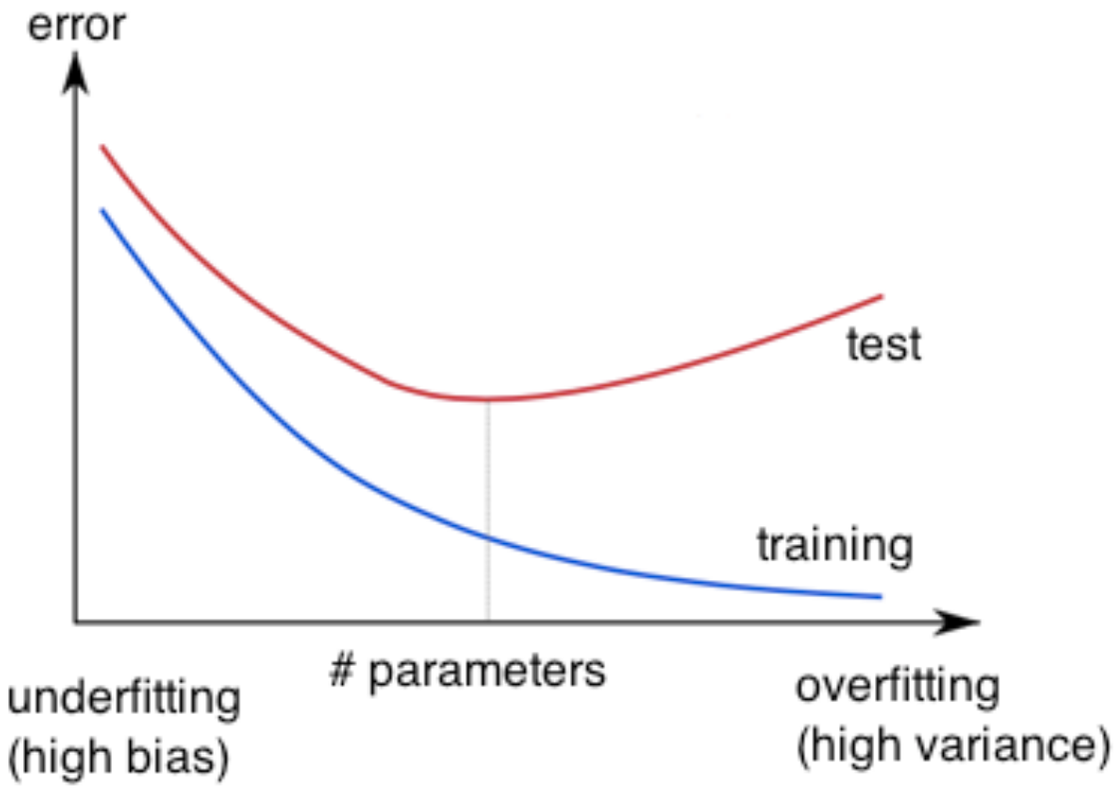
| | Classification | Regression |
|---|---|---|
| Training examples | $R^n$ x {class_1, ..., class_n} (one-hot encoding) | $R^n$ x $R^m$ |
| Output Layer | Soft-max [map $R^n$ to a probability distribution] $$P(y = j \mid \mathbf{x}) = \frac{e^{\mathbf{x}^\mathsf{T}\mathbf{w}_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\mathsf{T}\mathbf{w}_k}}$$ | Linear (Identity) or Sigmoid   |
| Cost (loss) function | Cross-entropy $$J(\theta) = -\frac{1}{n}\sum_{i=1}^{n}\sum_{k=1}^{K}\left[y_k^{(i)} \log \hat{y}_k^{(i)} + \left(1 - y_k^{(i)}\right) \log\left(1 - \hat{y}_k^{(i)}\right)\right]$$ | Mean Squared Error $$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - \hat{y}^{(i)}\right)^2$$ Mean Absolute Error $$J(\theta) = \frac{1}{n}\sum_{i=1}^{n}\left\lvert y^{(i)} - \hat{y}^{(i)}\right\rvert$$ |

# Overfitting



inadequate | good compromise | over-fitting

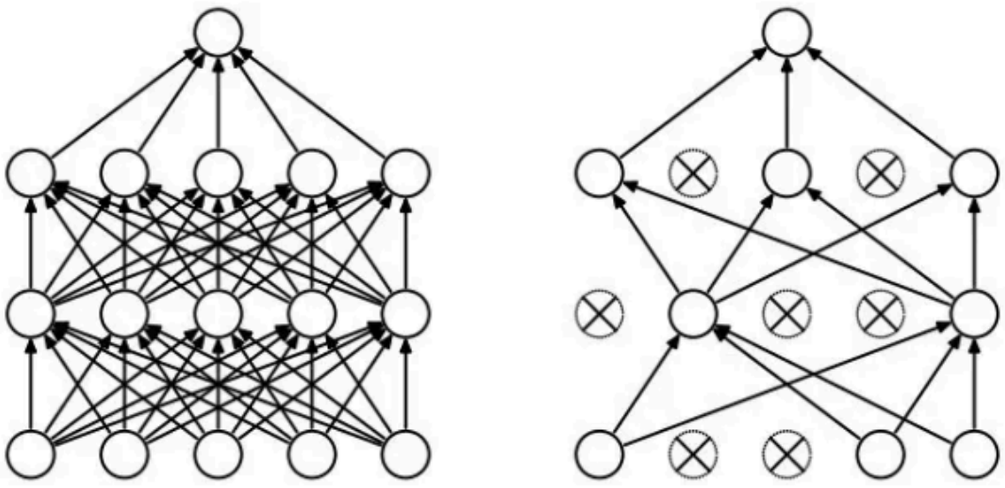
[http://wiki.bethanycrane.com/overfitting-of-data]




[https://www.neuraldesigner.com/images/learning/selection_error.svg]

Learned hypothesis may fit the training data very well, even outliers (noise) but fail to generalize to new examples (test data)

# Regularization



## Dropout

- Randomly drop units (along with their connections) during training
- Each unit retained with fixed probability p, independent of other units
- Hyper-parameter p to be chosen (tuned)

## L2 = weight decay

- Regularization term that penalizes big weights, added to the objective
- Weight decay value determines how dominant regularization is during gradient computation
- Big weight decay coefficient → big penalty for big weights

$$J_{reg}(\theta) = J(\theta) + \lambda \sum_k \theta_k^2$$

## Early-stopping

- Use validation error to decide when to stop training
- Stop when monitored quantity has not improved after n subsequent epochs
- n is called patience



[Srivastava 2014]

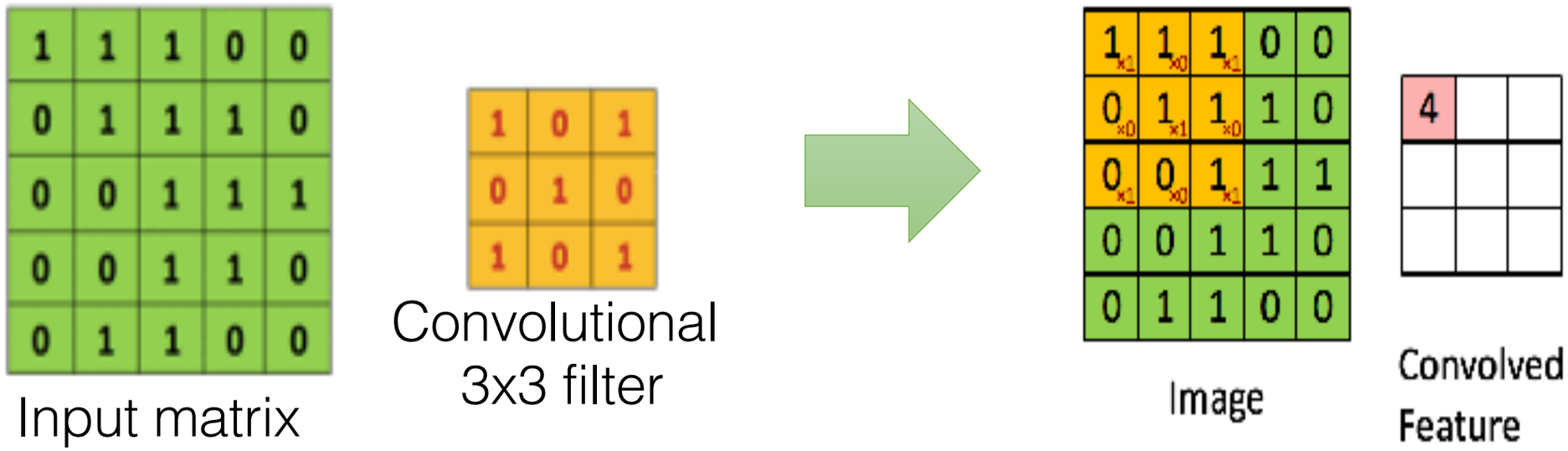
# Convolutional Neural Networks (CNNs)

- A specialized kind of neural network for processing data that has a known grid-like topology.
  - E.g., time-series data, which can be thought of as a 1-D grid taking samples at regular time intervals, and image data, which can be thought of as a 2-D grid of pixels
- The name “convolutional neural network” indicates that the network employs a mathematical operation called convolution . Convolution is a specialized kind of linear operation.
- Convolutional networks are neural networks that use convolution in place of general matrix multiplication in at least one of their layers.

# CNNs: Main Idea

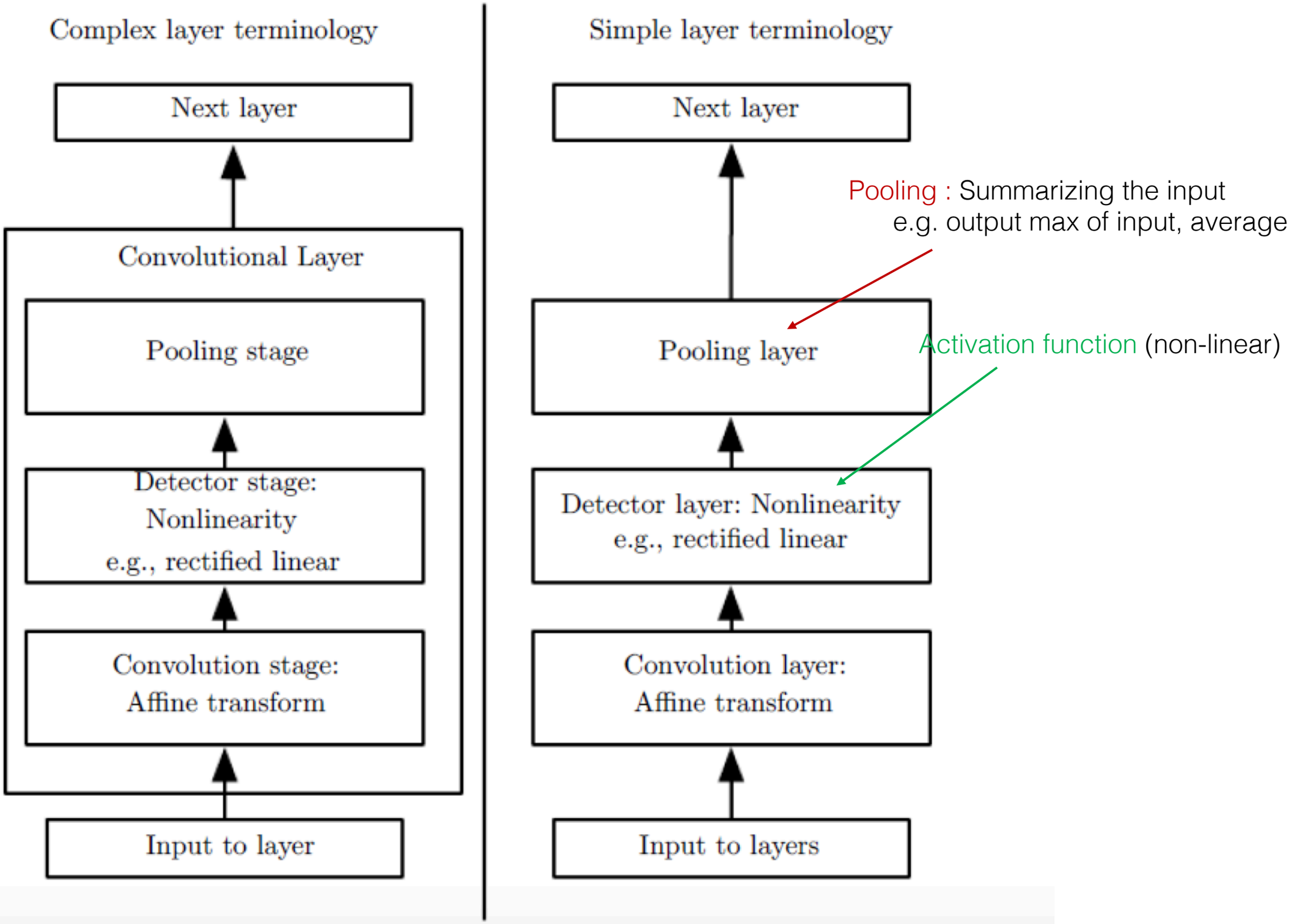
- Main CNN idea for text:
  - Compute vectors for n-grams and group them afterwards

Example: “this takes too long” compute vectors for:
This takes, takes too, too long, this takes too, takes too long, this takes too long



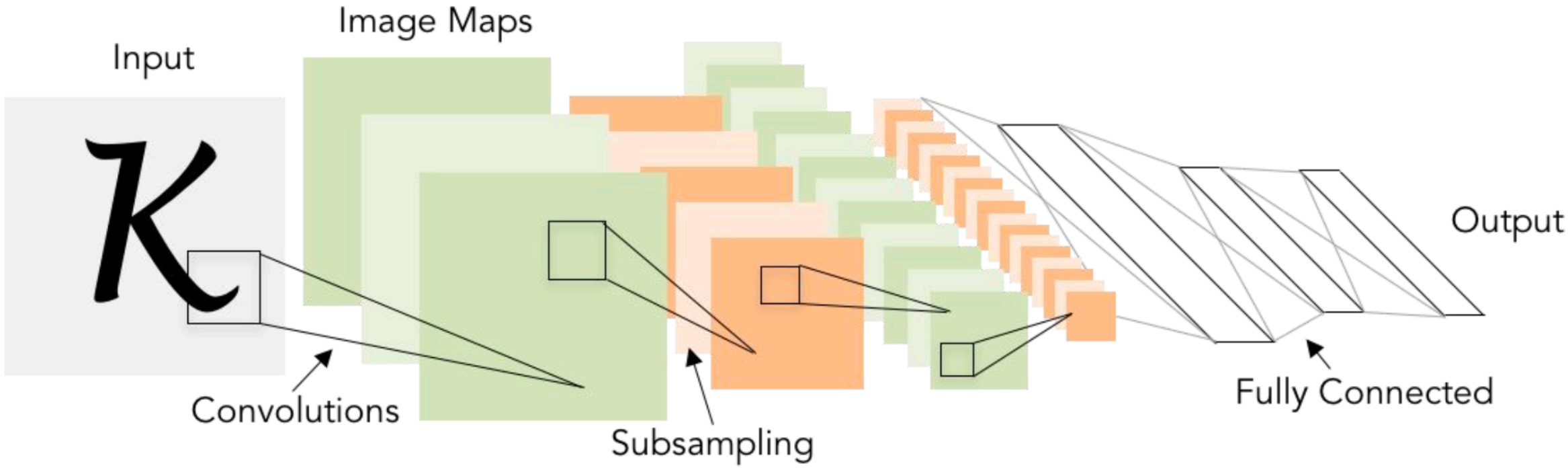

[http://deeplearning.stanford.edu/wiki/index.php/Feature_extraction_using_convolution]

# Terminology




[figure from Goodflow 2017]

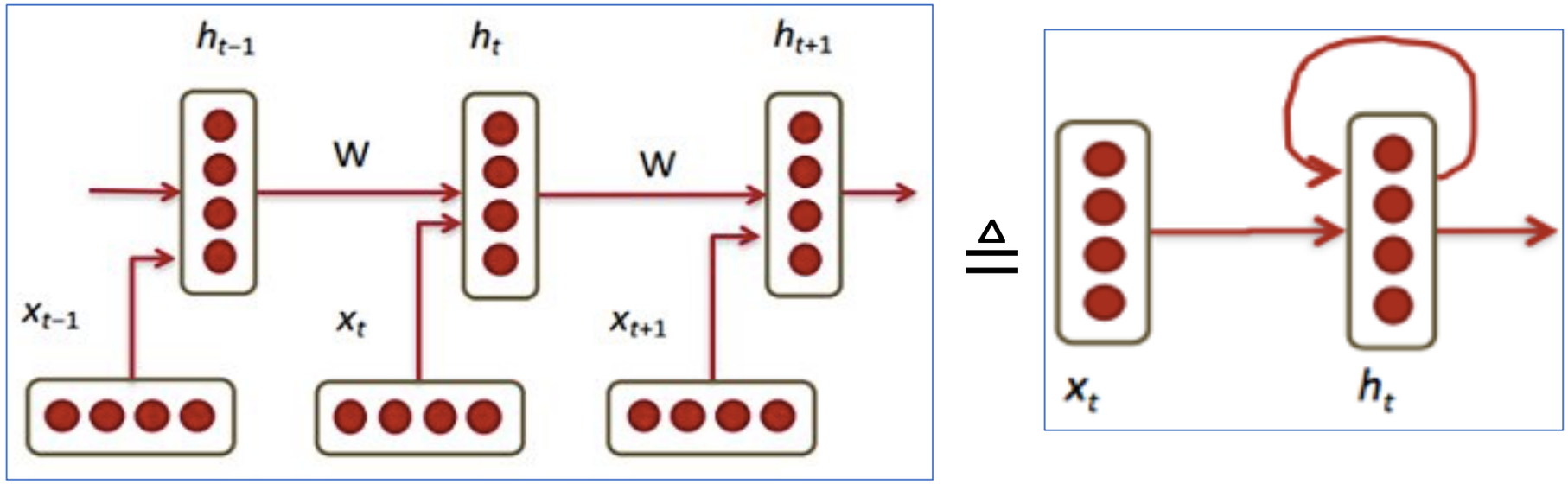
# How Do CNNs Work?




[LeCun 1998, Goodfeloow 2017]

# Recurrent Neural Networks (RNNs)

- Main RNN idea : Condition on all previous features
- Use same set of weights at all time steps

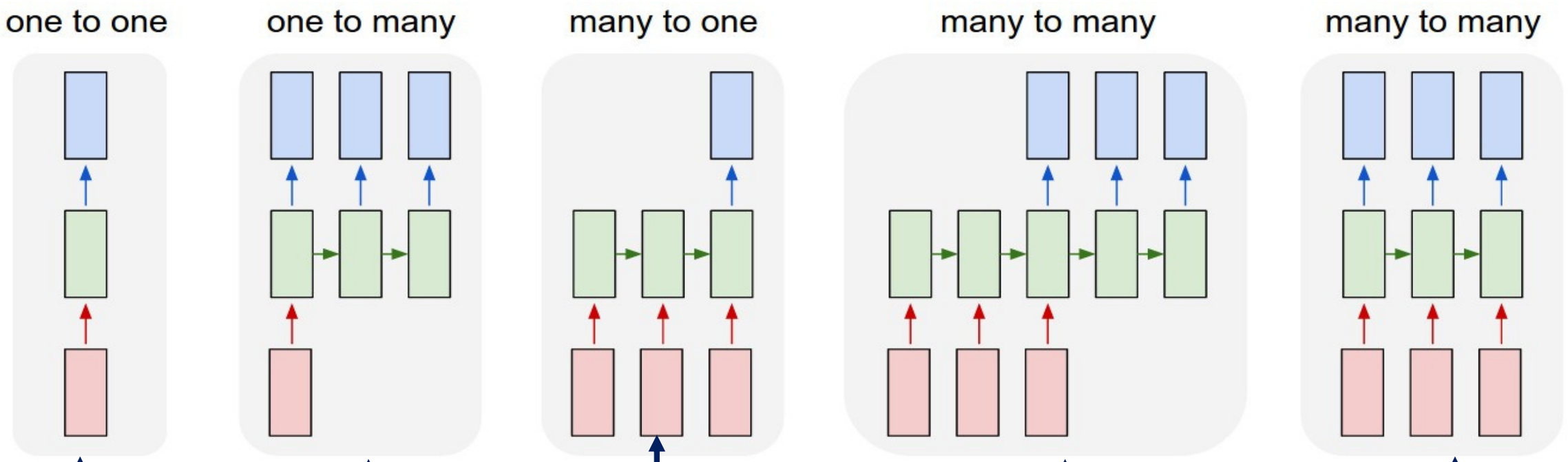



$$h_t = \sigma(W^{(hh)} h_{t-1} + W^{(hx)} x_t)$$

[https://discuss.pytorch.org/uploads/default/original/1X/6415da0424dd66f2f5b134709b92baa59e604c55.jpg]

e.g. **Sentiment Classification** sequence of words -> sentiment

# RNNs




Vanilla NNs

e.g., image captioning
Image → sequence of words

e.g. Sentiment Classification
sequence of words -> sentiment

e.g. Machine Translation
sequence of words -> sequence of words

e.g. Video classification on frame level

$$h_t = f_W(h_{t-1}, x_t)$$

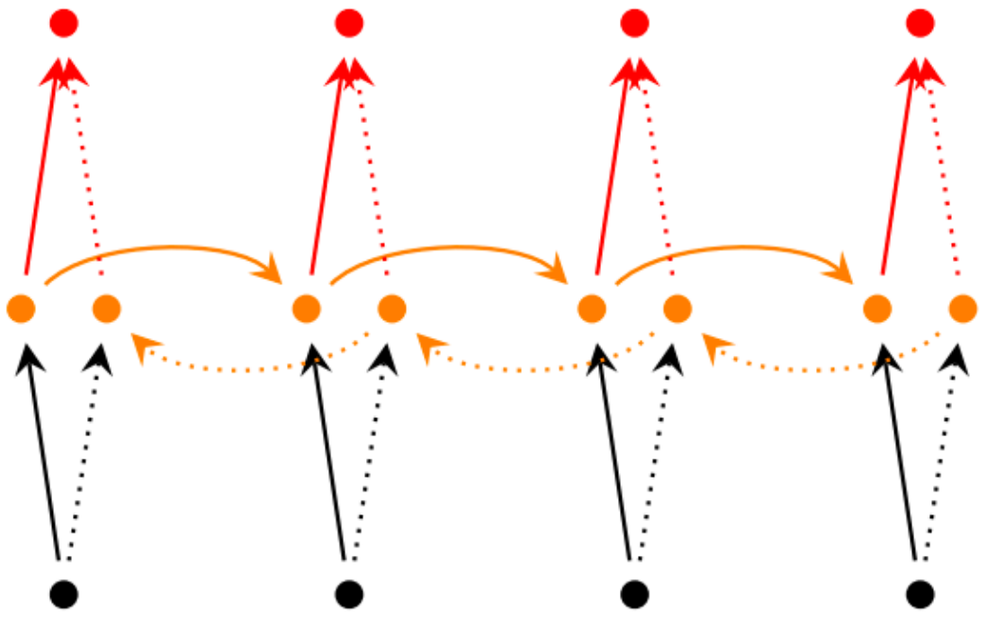
$$h_t = \tanh(W_{hh}h_{t-1} + W_{xh}x_t)$$

$$y_t = W_{hy}h_t$$

[https://discuss.pytorch.org/uploads/default/original/1X/6415da0424dd66f2f5b134709b92baa59e604c55.jpg]

# Bidirectional RNNs

- **Main idea:** incorporate both left and right context
- Output may not only depend on the previous elements in the sequence,  but also future elements.



$$\vec{h}_t = \sigma(\vec{W}^{(hh)}\vec{h}_{t-1} + \vec{W}^{(hx)}x_t)$$

$$\overleftarrow{h}_t = \sigma(\overleftarrow{W}^{(hh)}\overleftarrow{h}_{t+1} + \overleftarrow{W}^{(hx)}x_t)$$

$$y_t = f(\,[\vec{h}_t; \overleftarrow{h}_t\,])$$

past and future around a single token

- Two RNNs stacked on top of each other
- Output is computed based on the hidden state of both RNNs $[\vec{h}_t; \overleftarrow{h}_t\,]$



[Figure from http://www.wildml.com/2015/09/recurrent-neural-networks-tutorial-part-1-introduction-to-rnns/]

# Deep Learning: Applications in Bioinformatics

- CNN to better learn imaging features from CT scans and MRI images from head trauma, stroke diagnosis and brain (enlarged perivascular space ) EPV [Chilumkurthy 2018]
- CNN has been adopted rapidly in biomedical imaging studies [Ravi 2017]
- RNN is suitable to deal with long and sequential data, such as DNA array and genomics sequence [Xu 2016, Alipanahi 2015]
- RNN cannot interact with hidden neurons far from the current one → construct an efficient framework of recalling deep memory
  - MD-RNN in analyzing electron microscopy and MRIs of breast cancer samples [Kim 2018]
  - BRNN in protein secondary structure prediction [Baldi 1999]

# Deep Learning: Applications in Bioinformatics

- CNN can be combined with other deep learning models, such as RNN to predict imaging content, where CNN encodes an image and RNN generates the corresponding image description [[Angermueller 2016 ]
- Ensembled RNN-CNN architecture, DeepCpG, on single-cell DNA methylation data [Angermueller 2017 ]
  - better predict missing CpG status for genome-wide analysis
  - model's interpretable parameters shed light on the connection between sequence composition and methylation variability

- Sparse autoencoder (SAE), denoising autoencoder (DAE). Stacked sparse autoencoder (SSAE) was proposed to analyze high-resolution histopathological images in breast cancer [Xu 2016]