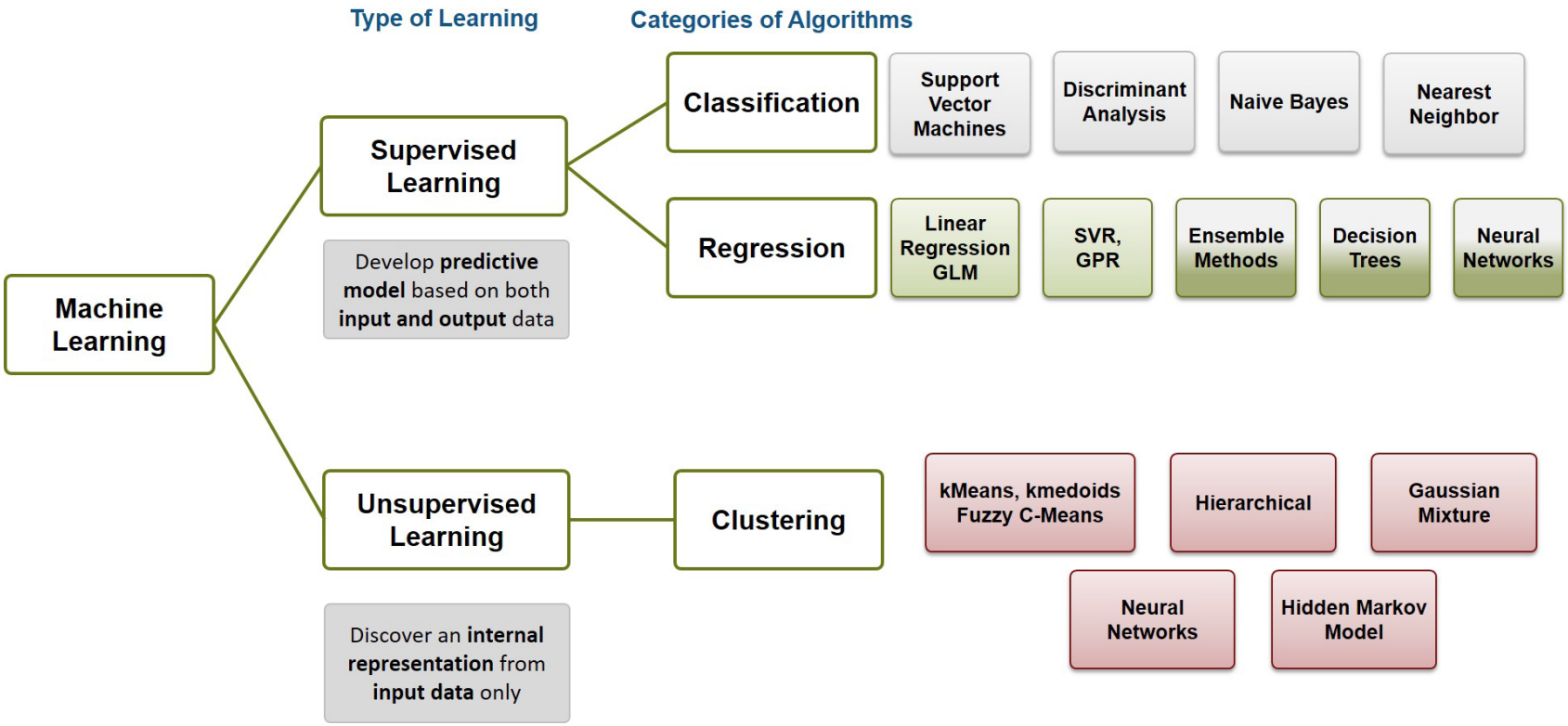
Session 15: Neural Networks

Data Analytics Course at IIFT

Dr. Tanujit Chakraborty

Machine Learning

A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E. - A definition by Tom Mitchell.



Slide Credit: Erwan Scornet, Ecole Polytechnique

Development of Statistical Machine Learning Models

- Linear Regression (Galton, 1875).
- Linear Discriminant Analysis (R.A. Fisher, 1936).
- Logistic Regression (Berkson, JASA, 1944).
- k-Nearest Neighbor (Fix and Hodges, 1951).
- Parzen's Density Estimation (E Parzen, AMS, 1962)
- Classification and Regression Tree (Breiman et al., 1984).
- Artificial Neural Network (Rumelhart et al., 1985).
- Perceptron Trees (Paul Utgoff, 1989, Connection Science).
- MARS (Friedman, 1991, Annals of Statistics).
- SVM (Cortes and Vapnik, Machine learning, 1995)
- Random forest (Breiman, 2001).
- Deep Convolutional Neural Nets (Krizhevsky, Sutskever, Hinton, NIPS 2012).
- Generative Adversarial Nets (Ian Goodfellow et al., NIPS 2014).
- Deep Learning (LeCun, Bengio, Hinton, Nature 2015).
- Bayesian Deep Neural Network (Yarin Gal, Islam, Zoubin Ghahramani, ICML 2017).

Developments of Neural Nets

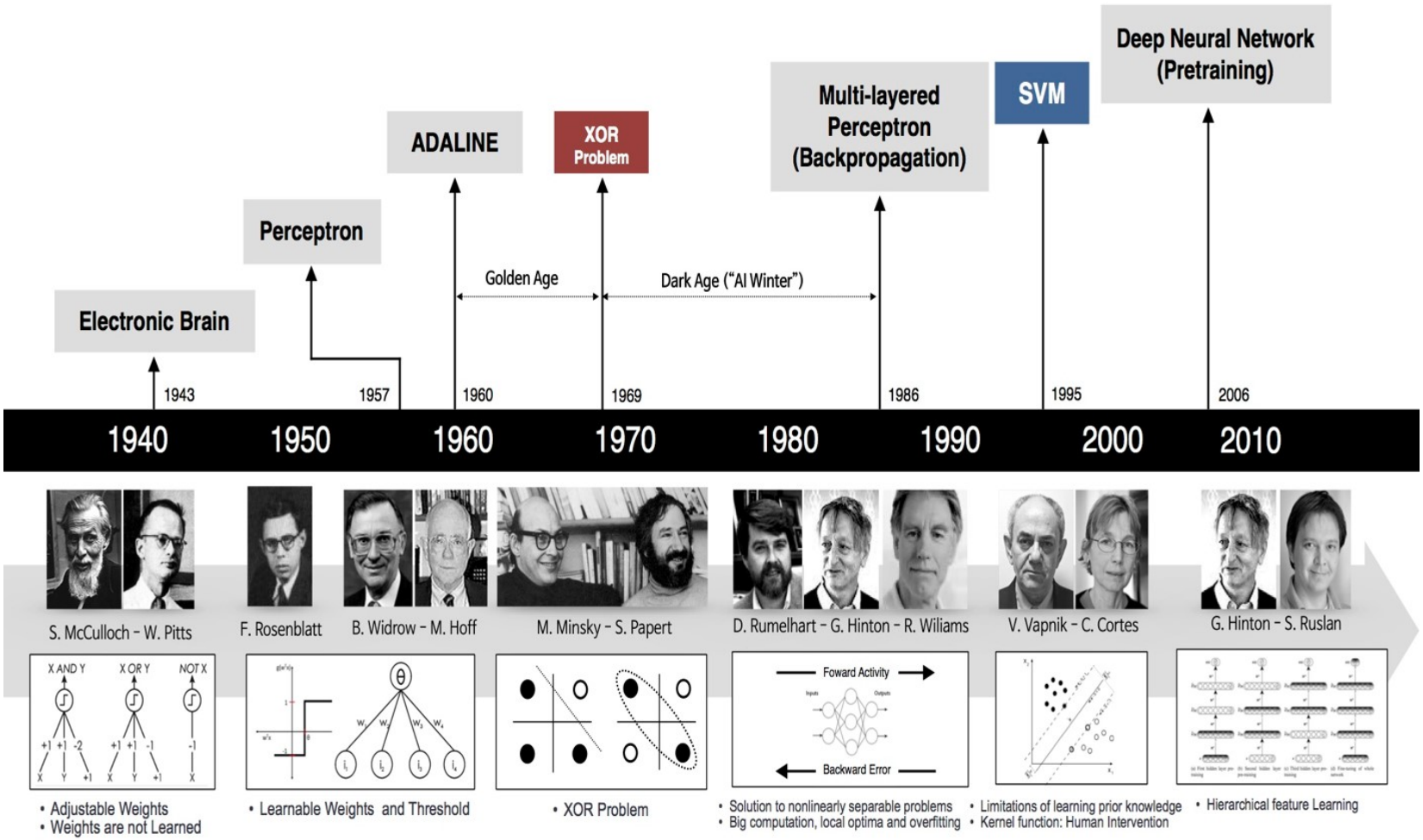
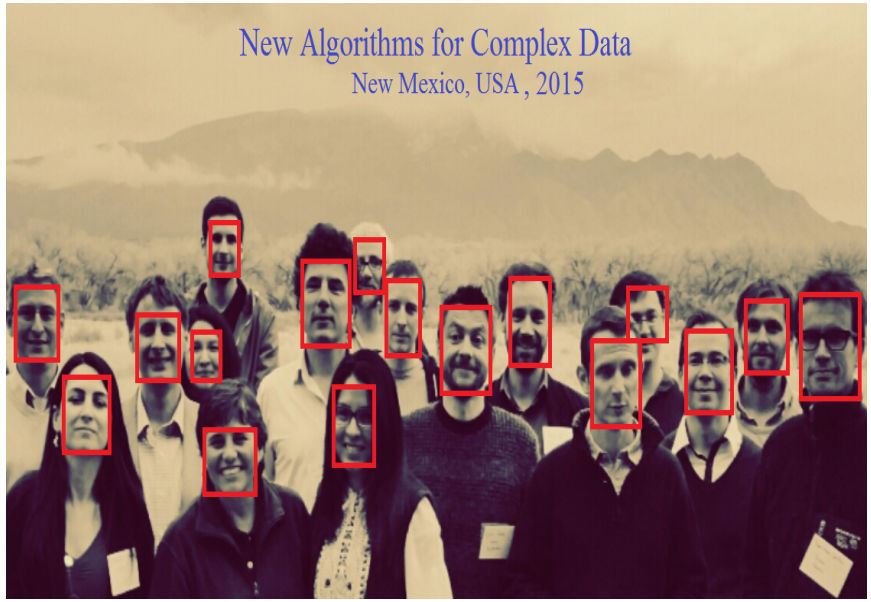


Fig: Developments of Neural Network Models

Motivating Examples

Spam detection (Text classification)
Data: email collection
Input: email
Output : Spam or No Spam.



Face Recognition
Data: Annotated database of images
Input : Sub window in the image
Output : Presence or no of a face...

Biology and Perceptron

- The idea of neural networks began as a model of how neurons in the brain function, termed 'connectionism' and used connected circuits to simulate intelligent behaviour.
- In 1943, portrayed with a simple electrical circuit by neurophysiologist Warren McCulloch and mathematician Walter Pitts.
- Donald Hebb took the idea further by proposing that neural pathways strengthen over each successive use, especially between neurons that tend to fire at the same time.

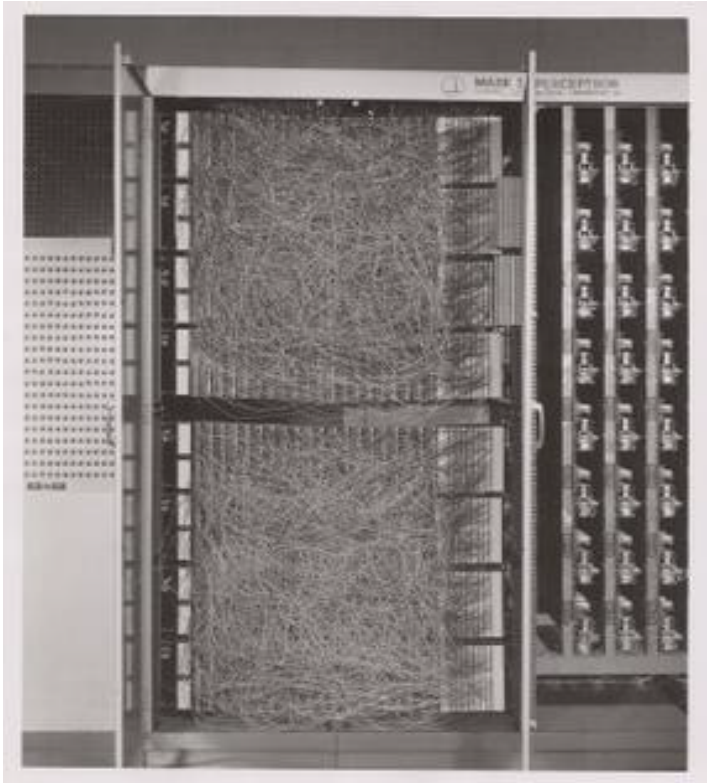


Fig: The Mark I Perceptron machine. The machine was connected to a camera that used 20x20 photocells to produce a 400-pixel image. The main visible feature is a patchboard that allowed experimentation with different combinations of input features. On the right, you can see potentiometers that implement the adaptive weights.

First Perceptron

- Around 1950's, Frank Rosenblatt, a psychologist at Cornell, was working on understanding the comparatively simpler decision systems present in the eye of a fly, which underlie and determine its flee response.
- In an attempt to understand and quantify this process, he proposed the idea of a Perceptron in 1958, calling it Mark I Perceptron.
- It was a system with a simple input output relationship, modeled on a McCulloch-Pitts neuron to explain the complex decision processes in a brain using a linear threshold gate.

Perceptron Machine

- A McCulloch-Pitts neuron takes in inputs, takes a weighted sum and returns 0 if the result is below threshold and 1 otherwise.
- The Mark I Perceptron machine was the first implementation of the perceptron algorithm.
- Limitations:
  1. Can't solve non linearly separable problems... (by design).
  2. Algorithm not robust to non linear separability!
- Solution: Artificial Neural Networks, e.g., Feedforward Neural Net, Radial Basis Function Network, Bayesian Neural Net).

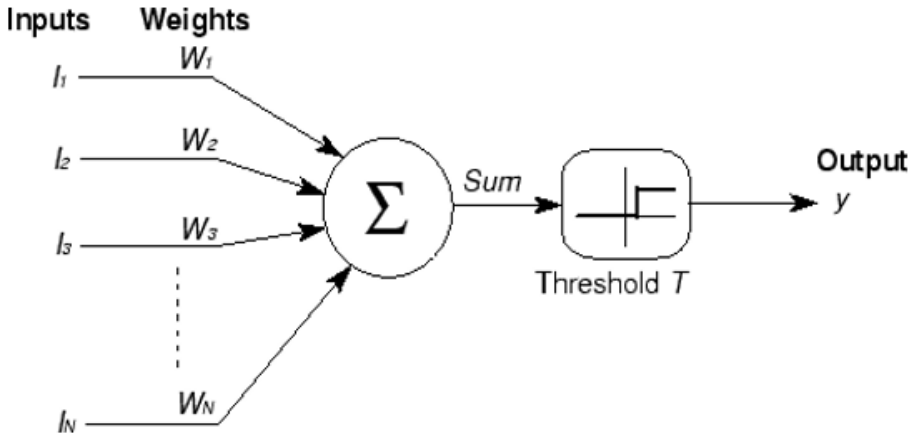


Fig: Most common representation of perceptron

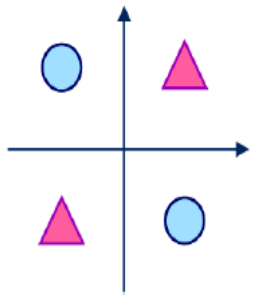


Fig: XOR Problem

Artificial Neural Networks

- ANN is composed of several perceptron-like units arranged in multiple layers.
- Consists of an input layer, one or more hidden layer, and an output layer.
- Nodes in the hidden layers compute a nonlinear transform of the inputs.
- Also called a Feedforward Neural Network (since there is no backward connections between layers, viz., no loops).
- Note: All nodes between layers are assumed connected with each other.

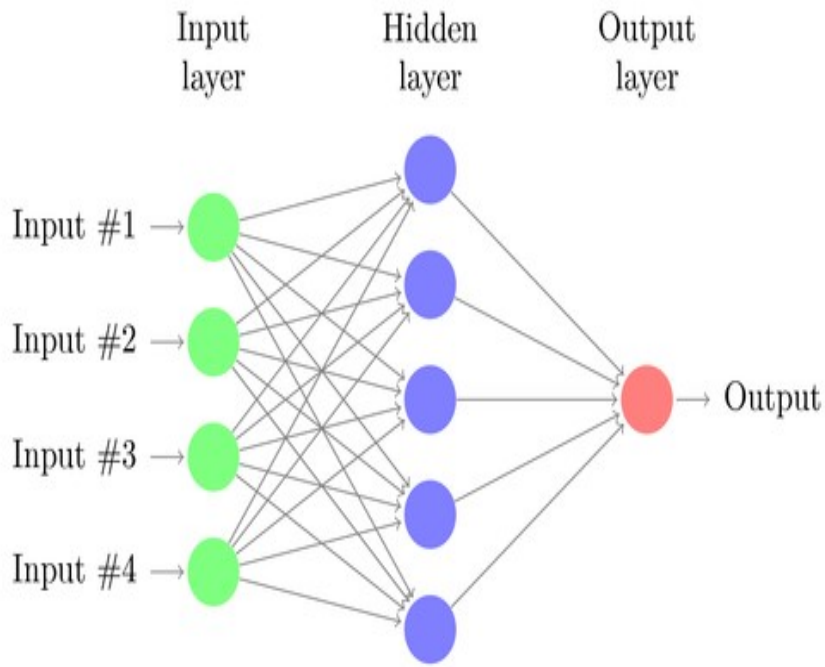
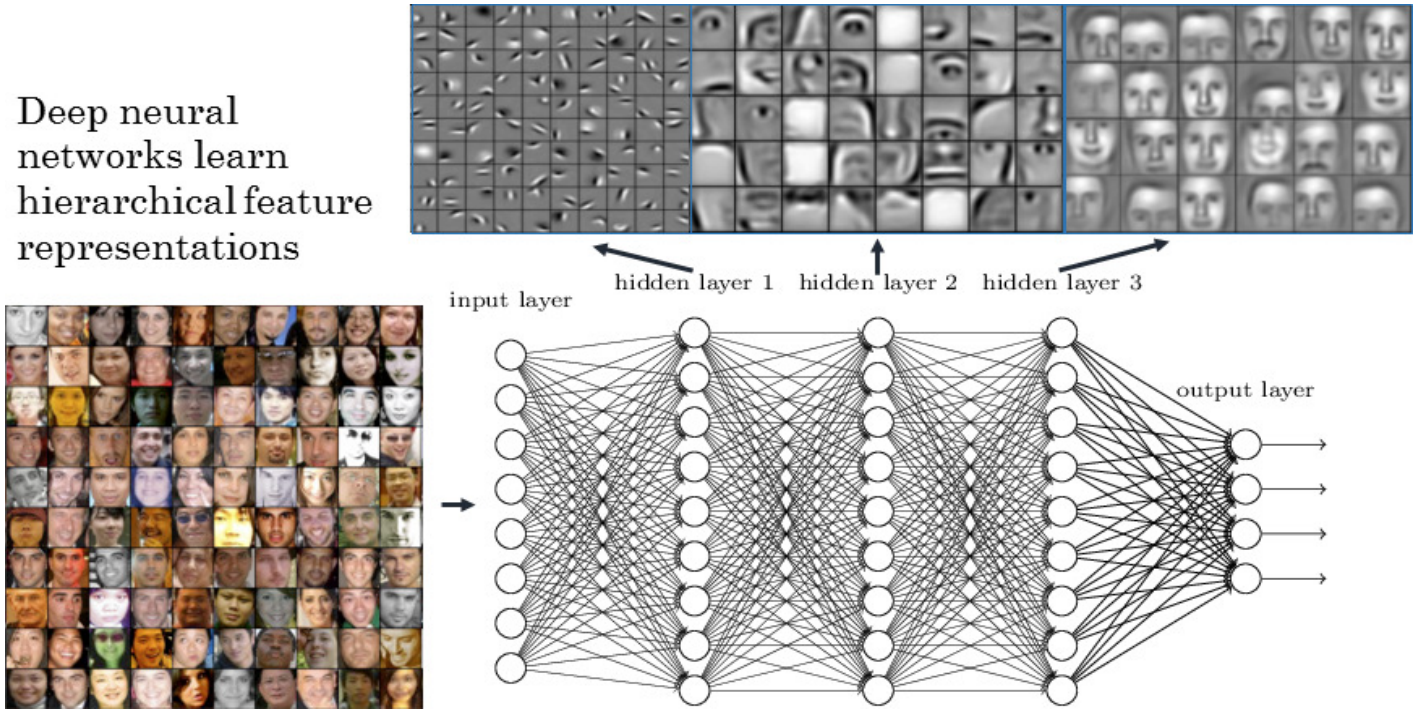


Fig: FFNN Model with one hidden layer with 5 hidden units

What do Hidden Layers Learn?

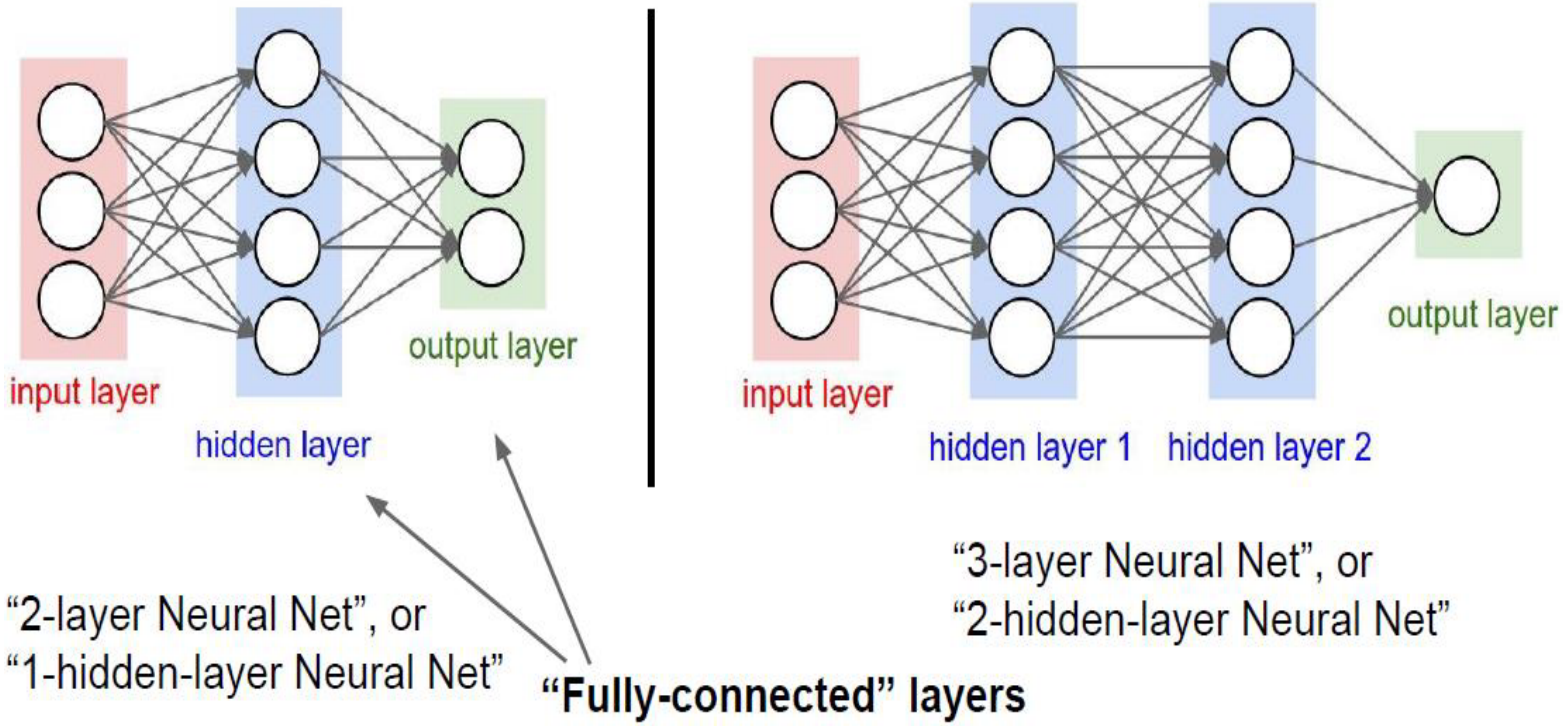
- Hidden layers can automatically extract features from data.



Slide Credit : Piyush Rai, IIT Kanpur

- The bottom-most hidden layer captures very low level features (e.g., edges).
- Subsequent hidden layers learn progressively more high-level features (e.g., parts of objects) that are composed of previous layer's features.

Neural Networks: Architectures



Slide Credit: Fei-Fei Li et al., Stanford University

Will I pass this class?

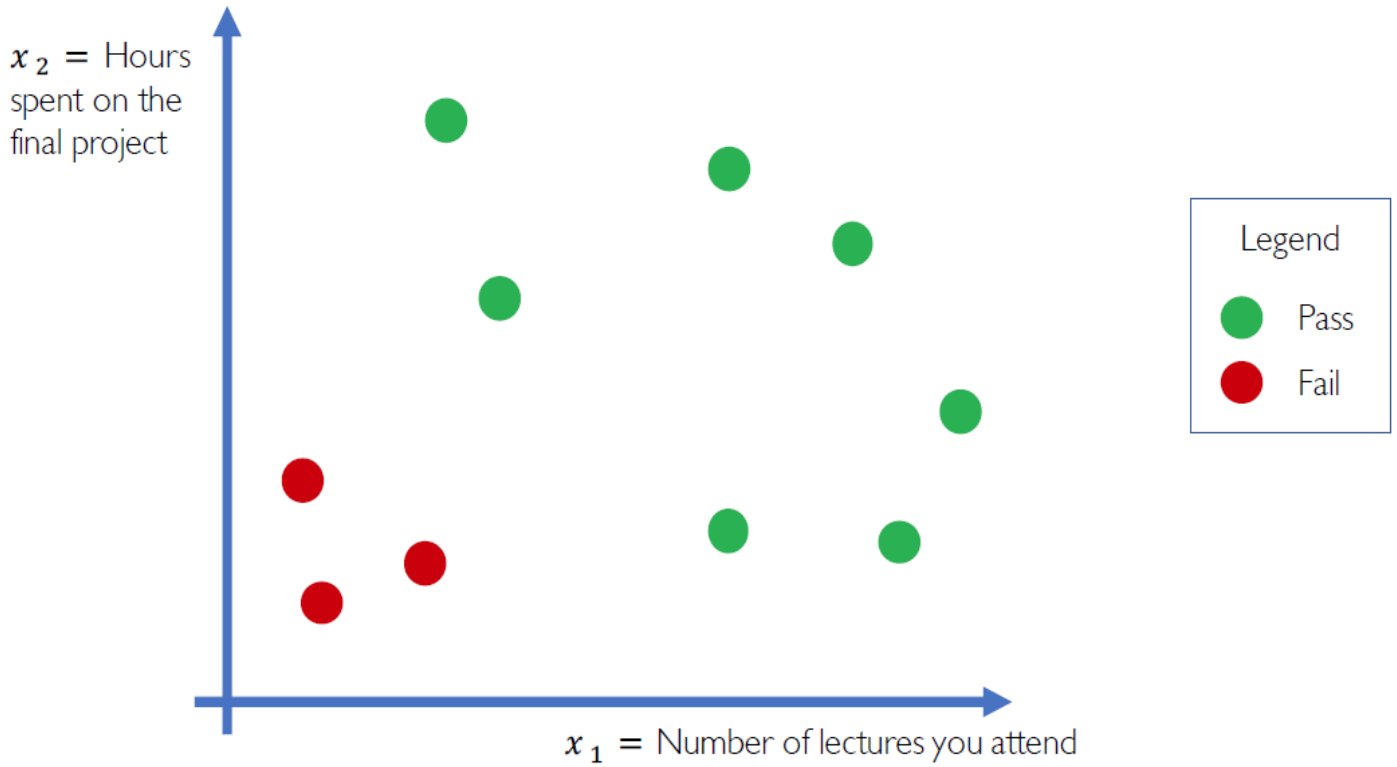
Let's start with a simple two feature model

x_1 = Number of lectures you attend

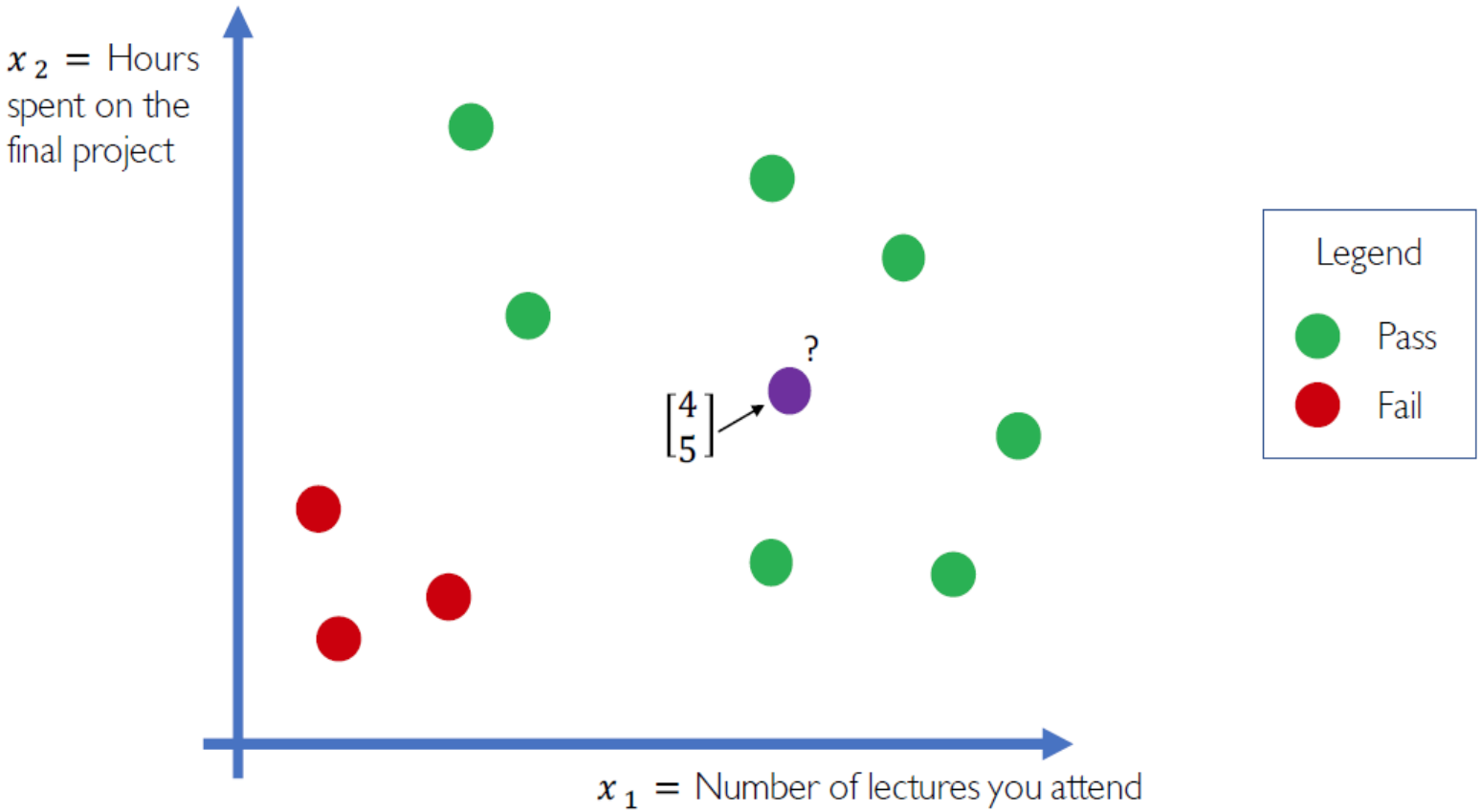
x_2 = Hours spent on the final project

Example Slide(s) Credit: Alexander Amini, MIT

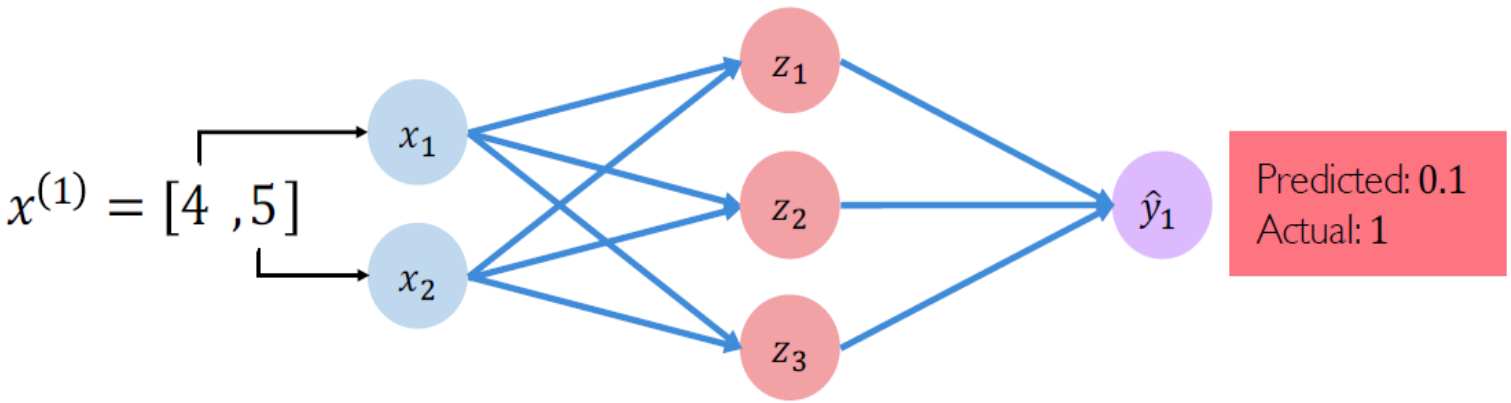
Example Problem: Will I pass this class?



Example Problem: Will I pass this class?

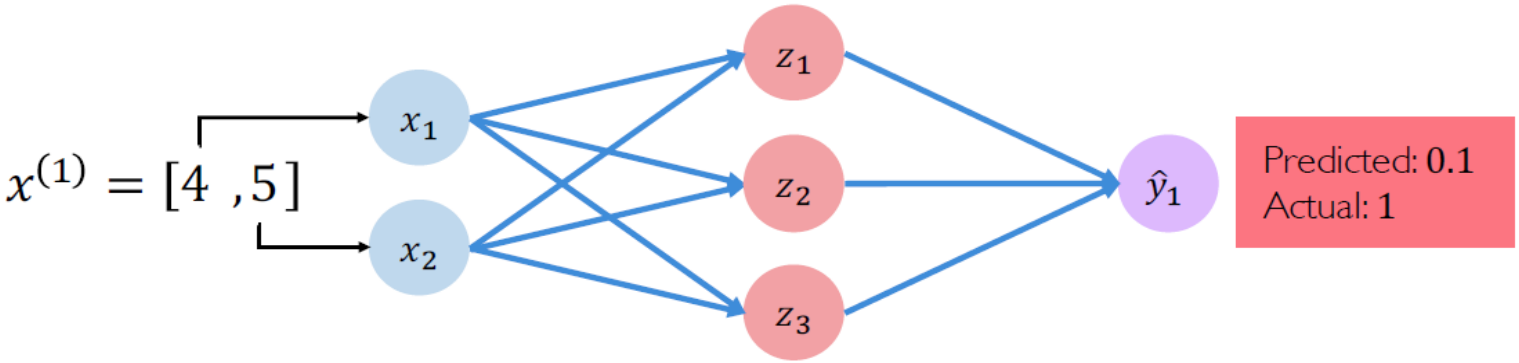


Example Problem: Will I pass this class?



Quantifying Loss

*The **loss** of our network measures the cost incurred from incorrect predictions*

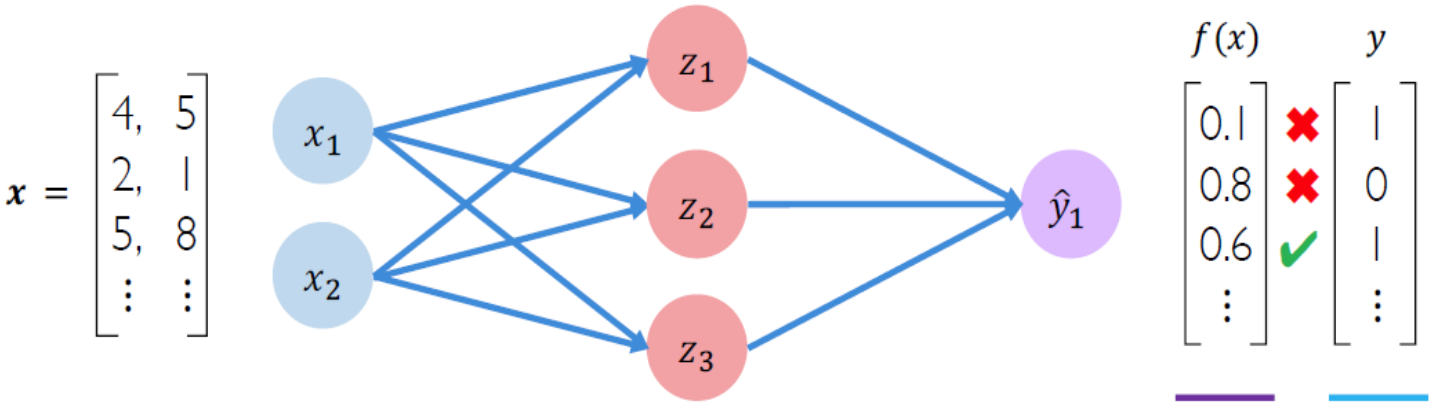


$$\mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{\theta}\right), y^{(i)}\right)$$

Predicted Actual

Empirical Loss

*The **empirical loss** measures the total loss over our entire dataset*



Also known as:

- Objective function
- Cost function
- Empirical Risk

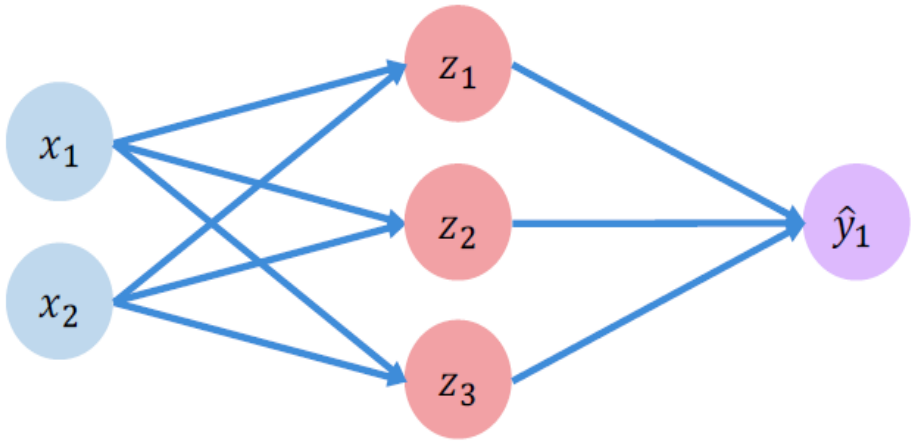
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)};\theta\right), y^{(i)}\right)$$

Predicted Actual

Binary Cross Entropy Loss

Cross entropy loss *can be used with models that output a probability between 0 and 1*

$$x = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$



$f(x)$ y

$$\begin{bmatrix} 0.1 \\ 0.8 \\ 0.6 \\ \vdots \end{bmatrix} \quad \begin{bmatrix} 1 \\ 0 \\ 1 \\ \vdots \end{bmatrix}$$

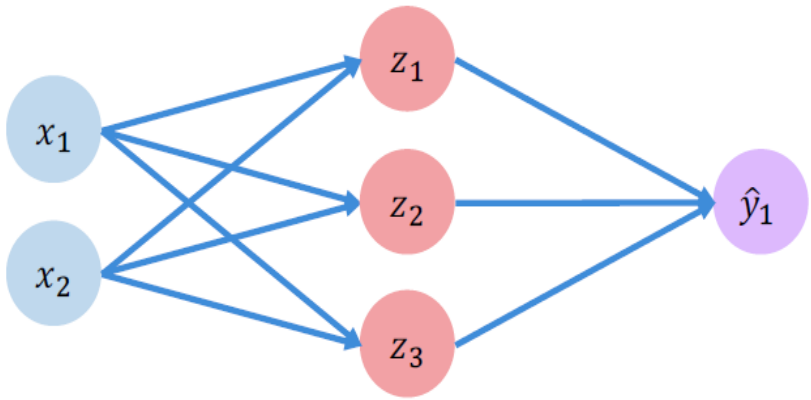
$$J(\theta) = \frac{1}{n}\sum_{i=1}^{n} y^{(i)} \log\left(f\left(x^{(i)};\theta\right)\right) + \left(1 - y^{(i)}\right) \log\left(1 - f\left(x^{(i)};\theta\right)\right)$$

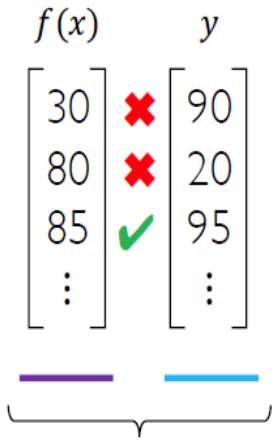
Actual Predicted Actual Predicted

Mean Squared Error Loss

Mean squared error loss *can be used with regression models that output continuous real numbers*

$$\boldsymbol{x} = \begin{bmatrix} 4, & 5 \\ 2, & 1 \\ 5, & 8 \\ \vdots & \vdots \end{bmatrix}$$





$$\boldsymbol{J(\theta)} = \frac{1}{n}\sum_{i=1}^{n}\left(y^{(i)} - f\left(x^{(i)};\theta\right)\right)^2$$

Actual Predicted

Loss Optimization

We want to find the network weights that **achieve the lowest loss**

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^{n} \mathcal{L}\left(f\left(x^{(i)}; \boldsymbol{\theta}\right), y^{(i)}\right)$$

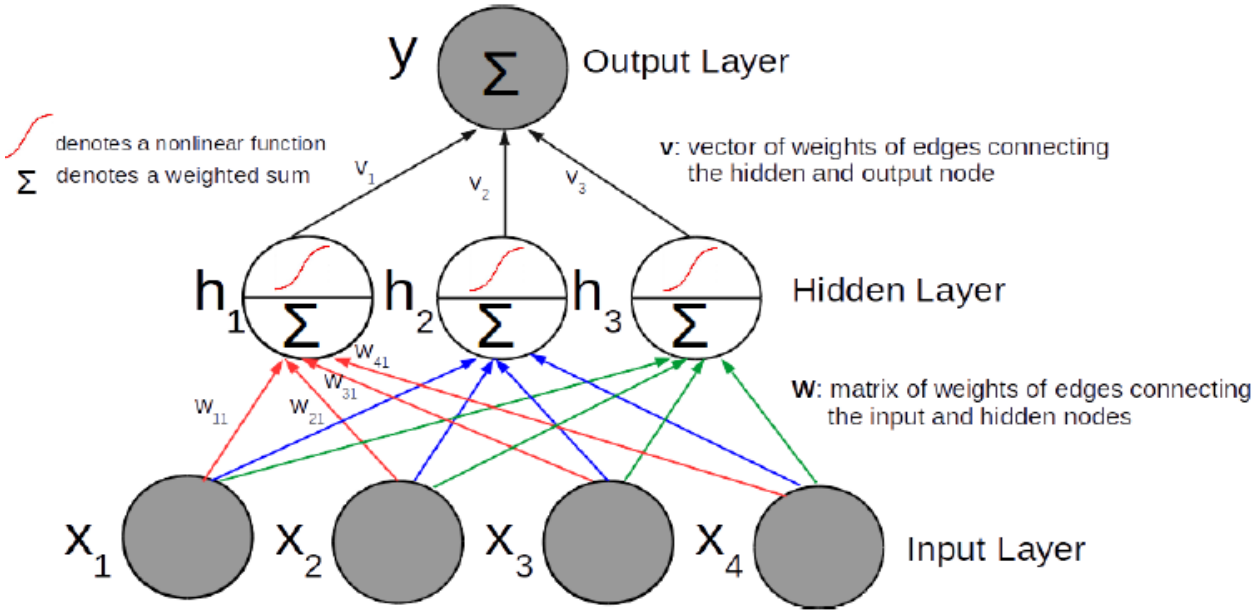
$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} J(\boldsymbol{\theta})$$

Remember:
$\boldsymbol{\theta} = \{\theta^{(0)}, \theta^{(1)}, \cdots\}$

Can be solved using Gradient Descent Algorithm.

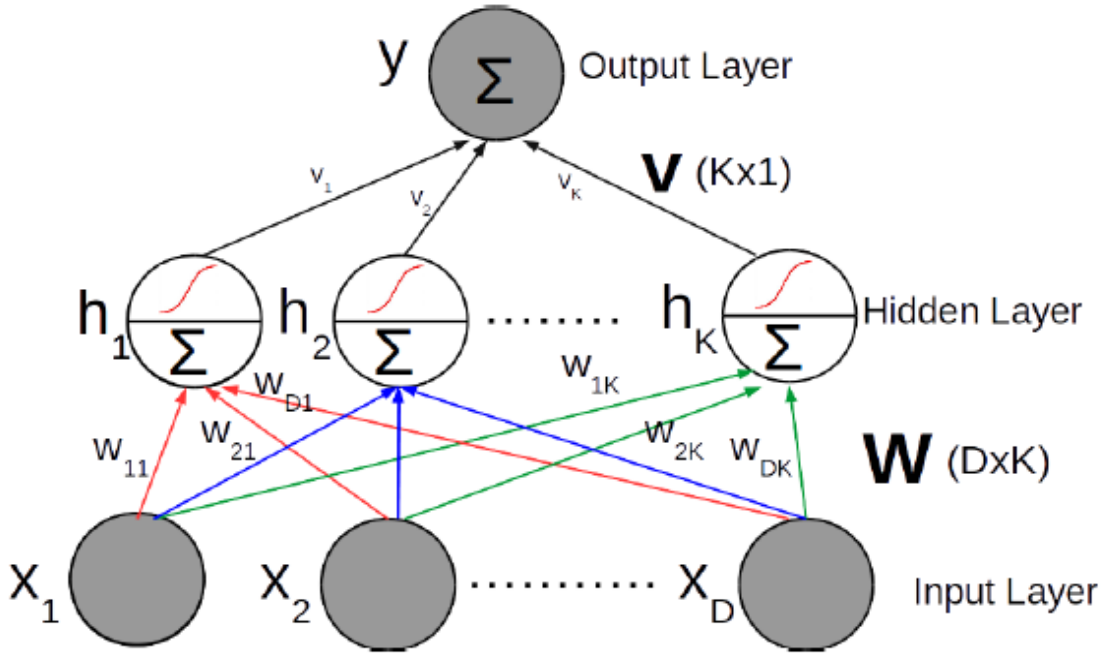
Feedforward Neural Net : Mathematical Formulation

- Below: FFNN with 4 inputs, one hidden layer with 3 nodes, and 1 output.



- Each hidden node computes a nonlinear transformation of its incoming inputs: Weighted linear combination followed by a nonlinear activation function.
- Output y is a weighted combination of the preceding layer's hidden nodes (followed by another transform if y isn't real valued, e.g., binary/multiclass label).

Feedforward Neural Net

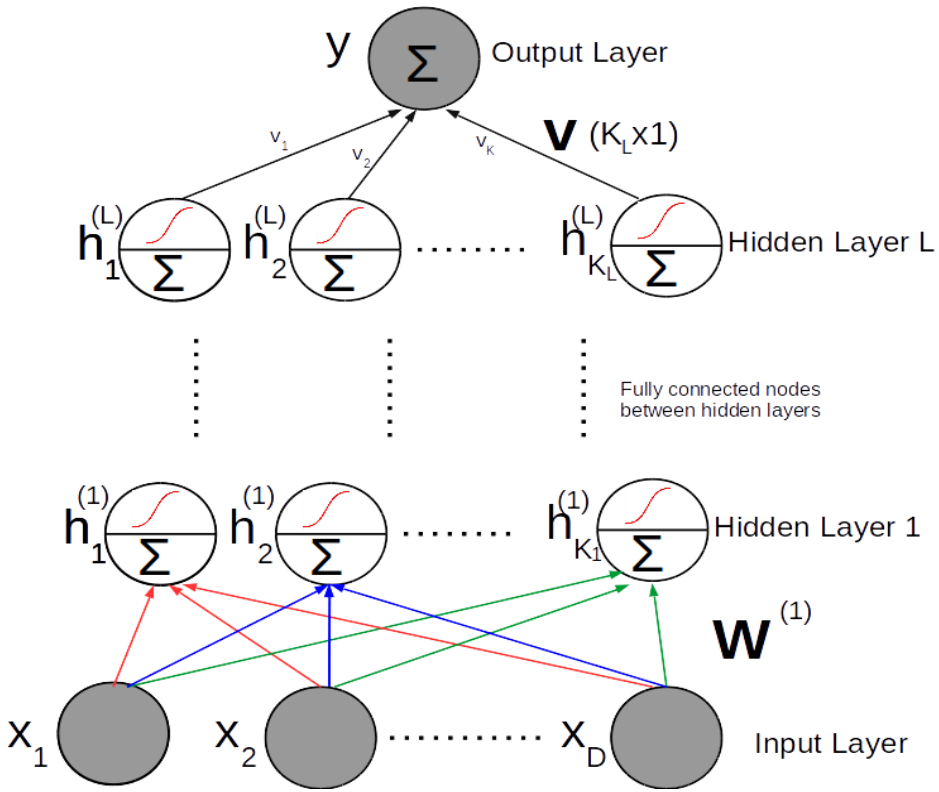


- For an FFNN with D inputs $\mathbf{x} = [x_1, ..., x_D]$, a single hidden layer with K hidden nodes $\mathbf{h} = [h_1, ..., h_K]$, and a scalar-valued output node y, each hidden node's value is computed as: $h_k = f(\mathbf{w}_k^T \mathbf{x}) = f\left(\sum_{d=1}^{D} w_{dk} x_d\right)$ and $\mathbf{h} = f(\mathbf{W}^T \mathbf{x})$.
- Output $y = \mathbf{v}^T \mathbf{h}$, where $\mathbf{v} = [v_1, ..., v_K] \in \mathbb{R}^K$, $\mathbf{W} = [\mathbf{w}_1, ..., \mathbf{w}_K] \in \mathbb{R}^{D \times K}$, and f is the nonlinear activation function.

(Deep) Feedforward Neural Net

- Feedforward neural net with L hidden layers $\boldsymbol{h}^{(1)}, \boldsymbol{h}^{(2)}, \ldots, \boldsymbol{h}^{(L)}$ where

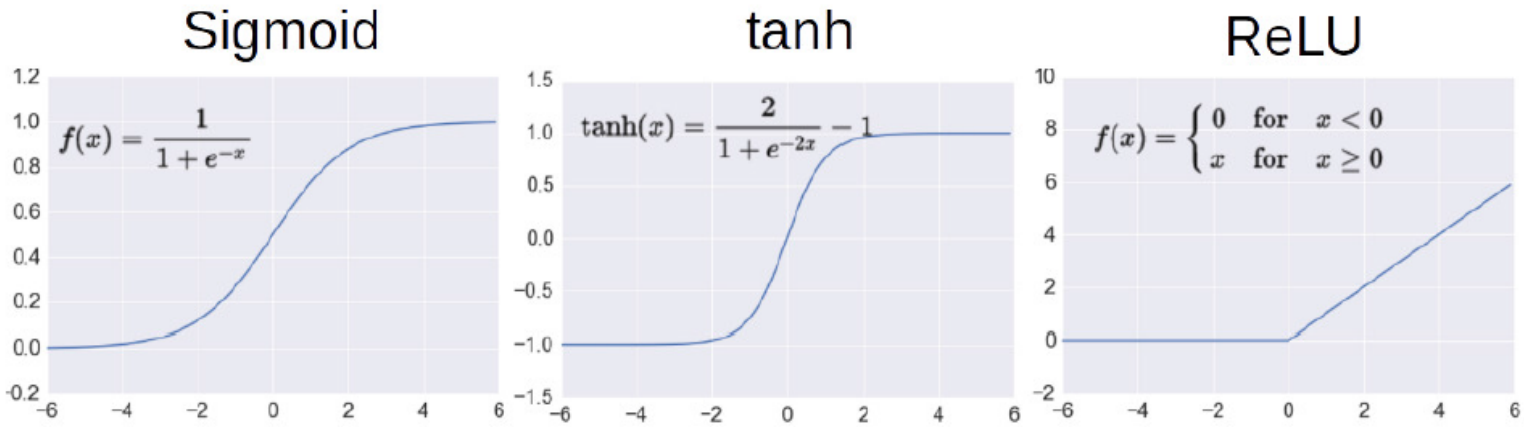
$$\boldsymbol{h}^{(1)} = f\left(\mathrm{W}^{(1)^\top} \boldsymbol{x}\right) \quad \text{and} \quad \boldsymbol{h}^{(\ell)} = f\left(\boldsymbol{W}^{(\ell)^\top} \boldsymbol{h}^{(\ell-1)}\right), \ell \geq 2.$$



- Note: The hidden layer ℓ contains K_ℓ hidden nodes, $\boldsymbol{W}^{(1)}$ is of size $D \times K_1$, $\boldsymbol{W}^{(\ell)}$ for $\ell \geq 2$ is of size $K_{\ell-1} \times K_\ell$, $\boldsymbol{v}$ is of size $K_L \times 1$.

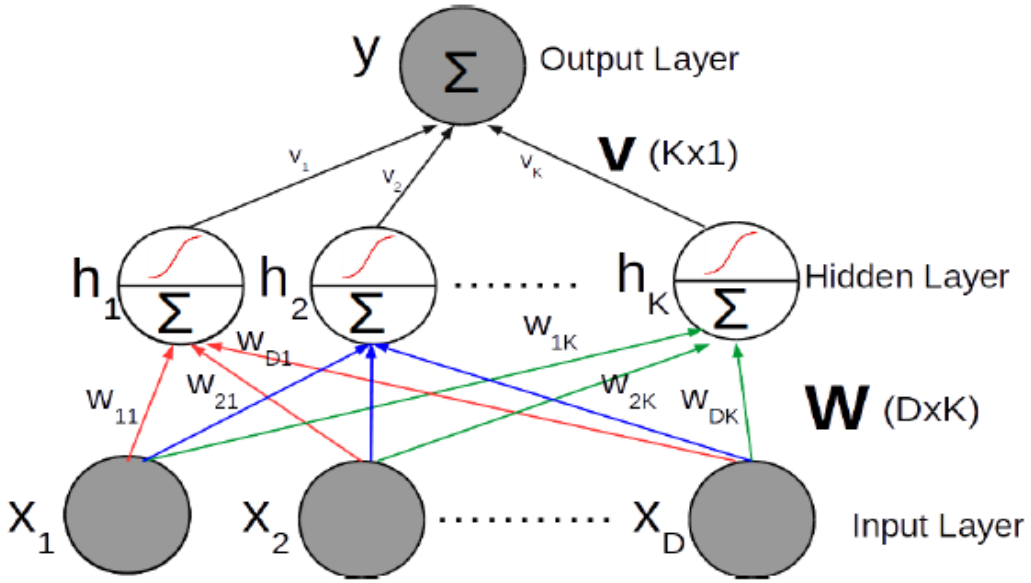
Nonlinear Activation Functions

- Some popular choices for the nonlinear activation functions:



- Sigmoid saturates and can kill gradients. Also not "zero-centered".
- tanh also saturates but is zero-centered (thus preferred over sigmoid).
- Rectified Linear Unit (ReLU): $f(x) = max(0; x)$ is currently the most popular (also cheap to compute).
- Learn the parameters by minimizing some loss function (e.g., squared loss).
- Backpropagation (gradient descent + chain rule for derivatives) is commonly used to do this efficiently (Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986)).

- Consider the feedforward neural net with one hidden layer:



- Assuming a regression problem, the optimization problem would be

$$\min_{\mathrm{W,v}} \frac{1}{2} \sum_{n=1}^{N} \left(y_n - \boldsymbol{v}^\top f\left(\boldsymbol{W}^\top \boldsymbol{x}_n \right) \right)^2 = \min_{\mathrm{W,v}} \frac{1}{2} \sum_{n=1}^{N} \left(y_n - \sum_{k=1}^{K} v_k f\left(\boldsymbol{w}_k^\top \mathrm{x}_n \right) \right)^2,$$

where $\boldsymbol{w}_k$ is the k -th column of the $D \times K$ matrix $\boldsymbol{W}$.

- We can learn the parameters by doing gradient descent (or stochastic gradient descent) on the objective function (loss function)

$$\mathcal{L} = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \sum_{k=1}^{K} v_k f\left(\boldsymbol{w}_k^\top \boldsymbol{x}_n\right)\right)^2 = \frac{1}{2}\sum_{n=1}^{N}\left(y_n - \boldsymbol{v}^\top \boldsymbol{h}_n\right)^2.$$

- Gradient w.r.t. $\boldsymbol{v} = [v_1, v_2, \ldots, v_K]$ is straightforward

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{v}} = -\sum_{n=1}^{N}\left(y_n - \sum_{k=1}^{K} v_k f\left(\boldsymbol{w}_k^\top \boldsymbol{x}_n\right)\right)\boldsymbol{h}_n = -\sum_{n=1}^{N}\boldsymbol{e}_n \boldsymbol{h}_n.$$

- Gradient w.r.t. the weights $\mathrm{W} = [\boldsymbol{w}_1, \boldsymbol{w}_2, \ldots, \boldsymbol{w}_K]$ is a bit more involved due to the presence of f but can be computed using chain rule

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{w}_k} = \frac{\partial \mathcal{L}}{\partial f_k}\frac{\partial f_k}{\partial \boldsymbol{w}_k} \quad \text{(note: } f_k = f(\boldsymbol{w}_k^\top \boldsymbol{x})\text{)}.$$

Learning Feedforward Neural Nets

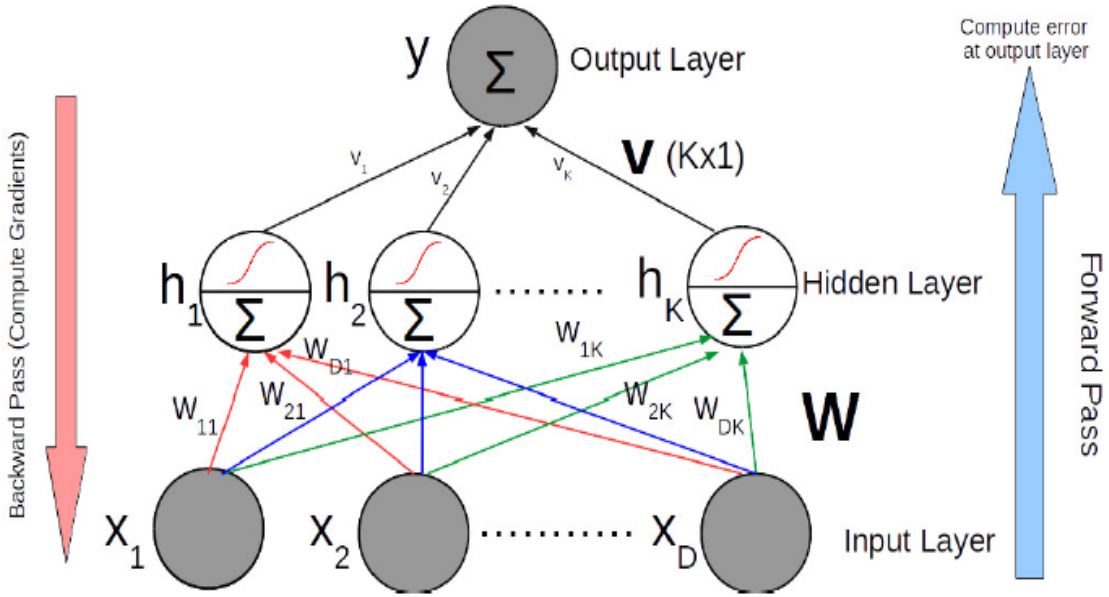
- We have: $\frac{\partial \mathcal{L}}{\partial f_k} = -\sum_{n=1}^{N} \left(y_n - \sum_{k=1}^{K} v_k f\left(\boldsymbol{w}_k^\top \boldsymbol{x}_n\right)\right) v_k = -\sum_{n=1}^{N} \mathrm{e}_n v_k$
- We have: $\frac{\partial f_k}{\partial w_k} = \sum_{n=1}^{N} f'\left(\mathrm{w}_k^\top \boldsymbol{x}_n\right) \mathrm{x}_n$, where $f'\left(\mathrm{w}_k^\top \boldsymbol{x}_n\right)$ is f's derivative at $\boldsymbol{w}_k^\top \boldsymbol{x}_n$
- These calculations can be done efficiently using backpropagation.
- The prediction rule for a deep neural network

$$y = \sum_{k=1}^{K} v_k h_k.$$

- In this case, the h_k 's are learned from data (possibly after multiple layers of nonlinear transformations).
- Note : Kernel SVM and deep FFNN be seen as using nonlinear basis functions for making predictions. Kernel methods use fixed basis functions (defined by the kernel) whereas NN learns the basis functions adaptively from data.

Backpropagation

- Basically consists of a forward pass and a backward pass:



- Forward pass computes the errors e_n using the current parameters.
- Backward pass computes the gradients and updates the parameters, starting from the parameters at the top layer and then moving backwards.
- Also good at reusing previous computations (updates of parameters at any layer depends on parameters at the layer above).

Remember: Artificial Neural Networks

- Universal Approximation Theorem (Hornik, 1989): A one hidden layer FFNN with sufficiently large number of hidden nodes can approximate any function.
- Caution: This result is only in terms of theoretical feasibility. Learning the model can be very difficult in practice (e.g., due to optimization difficulties).
- In Deep Neural Networks, hidden layer can automatically extract features from data. (Hinton 2006, Nature)
- Stochastic gradient descent backpropagation is most popular method for weight optimization.

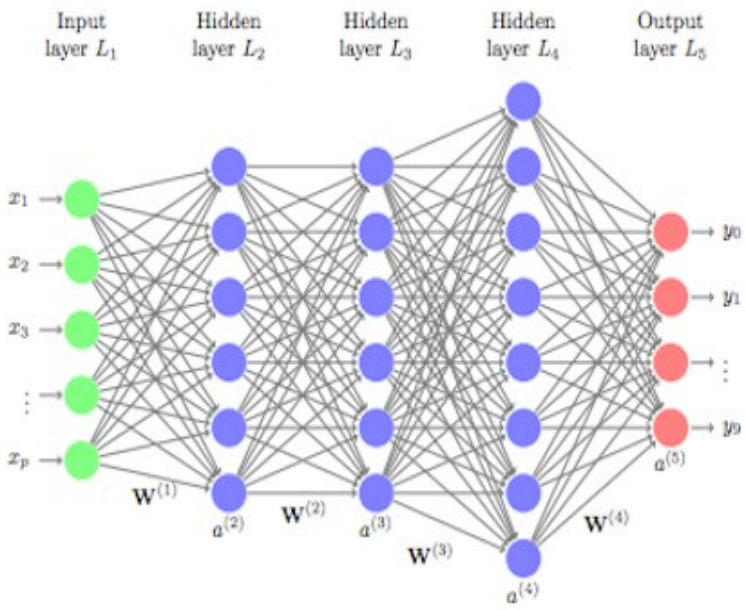


Fig: An example of ANN model with 3 hidden layers

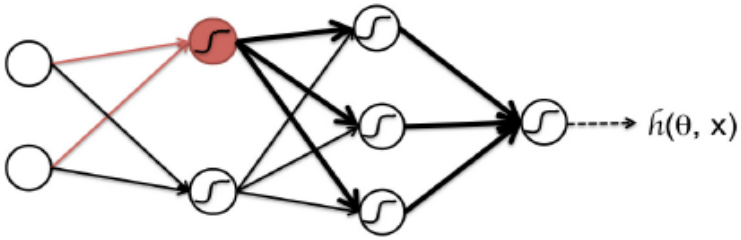
Pros & Cons of Artificial Neural Network

NO FREE LUNCH !

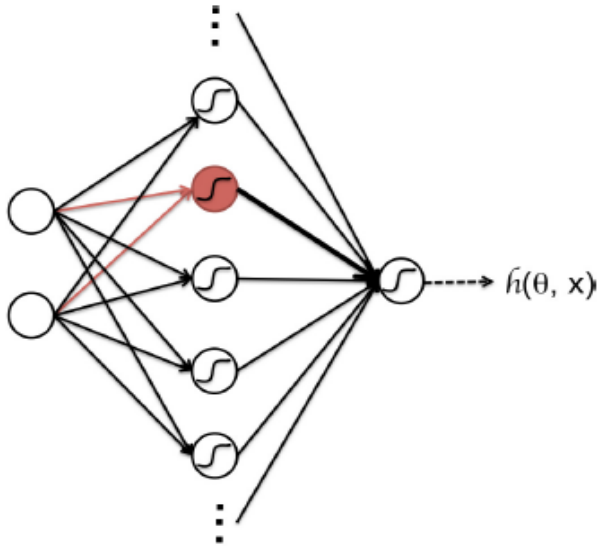
- Able to learn any complex nonlinear mapping or approximate any continuous function.
- As a nonparametric model, it doesn't make any prior assumption about the data distribution or input-output mapping function.
- ANN are very flexible with respect to incomplete, missing and noisy data. ANN are "fault tolerant".
- Neural network when applied to limited data can overfit the training data and lose generalization capability.
- ANNs can't deal with sequential or "temporal" data.
- The selection of the network topology and its parameter lack theoretical background, it is often a "trial and error" matter.
- ANNs lack memory. Solution: Recurrent Neural Networks, deep learning.

Deep Vs. Shallow Networks

- Why would one prefer a deep model over a wide and shallow model for large data sets?
  – For small / moderate data people still uses FFNN, RBFN, and BNN due to their interpretability and accuracy.



Deep Network



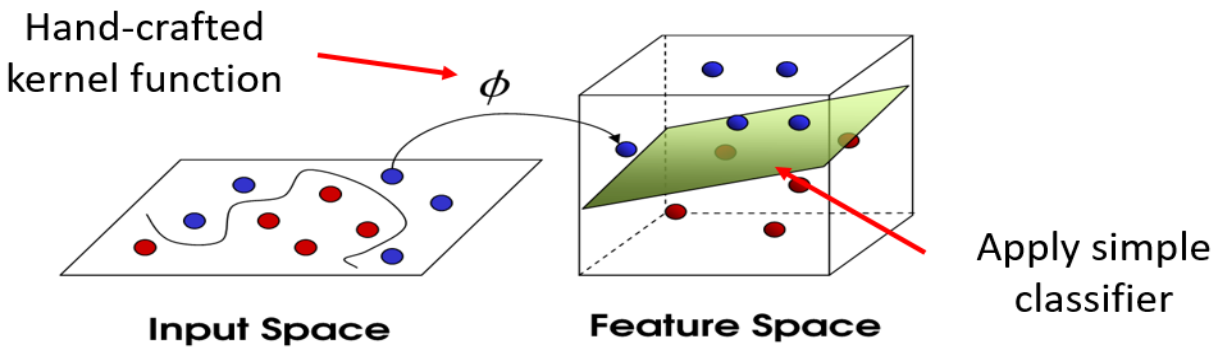
Shallow Network

An informal justifications:

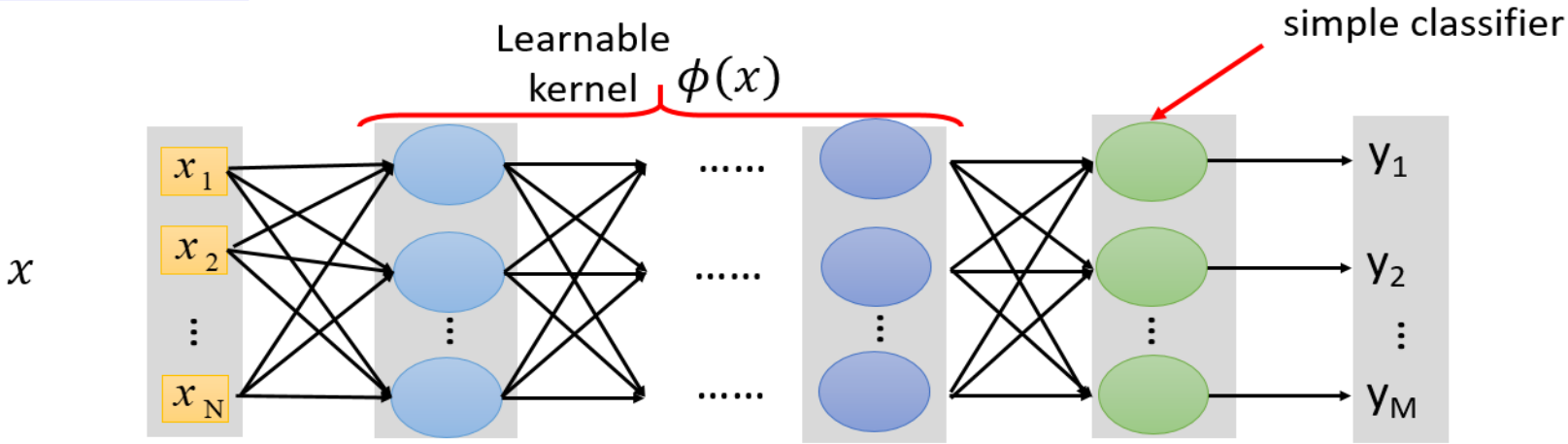
- Deep models can reuse computational subroutines (and are more compact)
- Learning Certain functions may require a huge number of units in a shallow model.

SVM Vs. Deep Learning

SVM



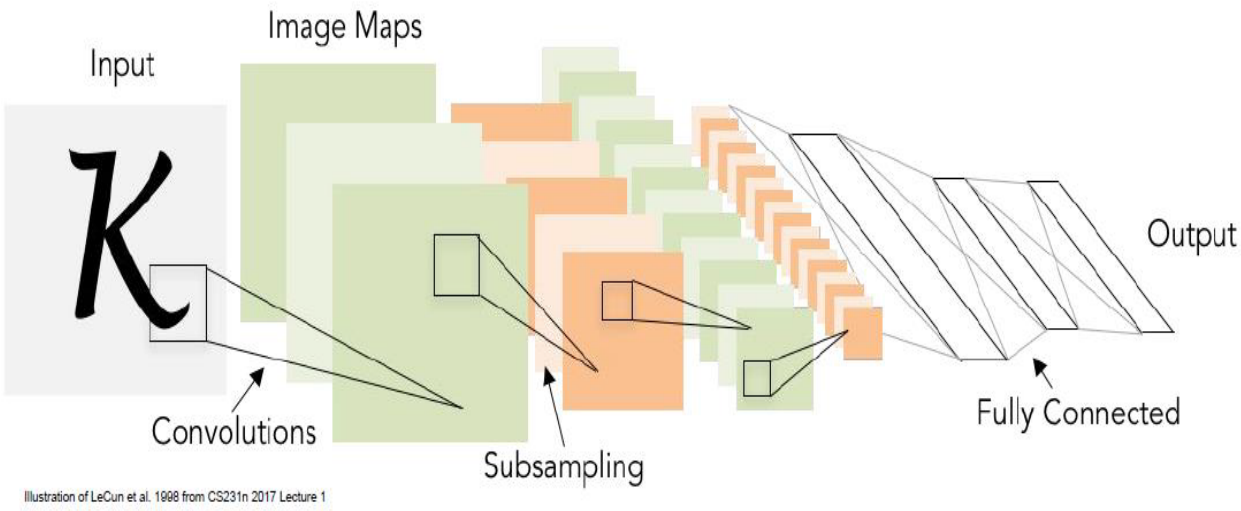
Deep Learning



Slide Credit: Swagatam Das, ISI Kolkata

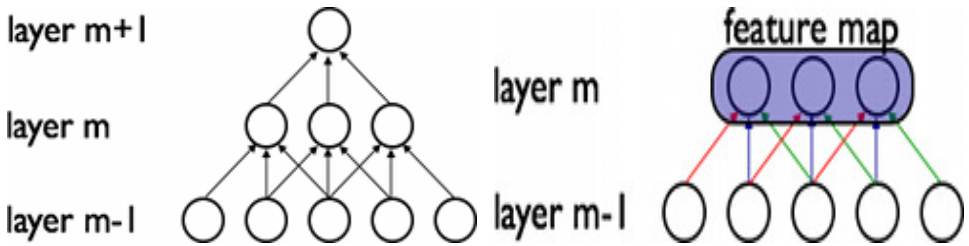
More about Deep Learning : ConvNet / CNN

- Convolutional Neural Networks (CNN) evolved mainly to deal with images.
- The 2012 Breakthrough : A Krizhevsky, I Sutskever, and GE Hinton, "Imagenet classification with deep convolutional neural networks", NeurIPS (2012).



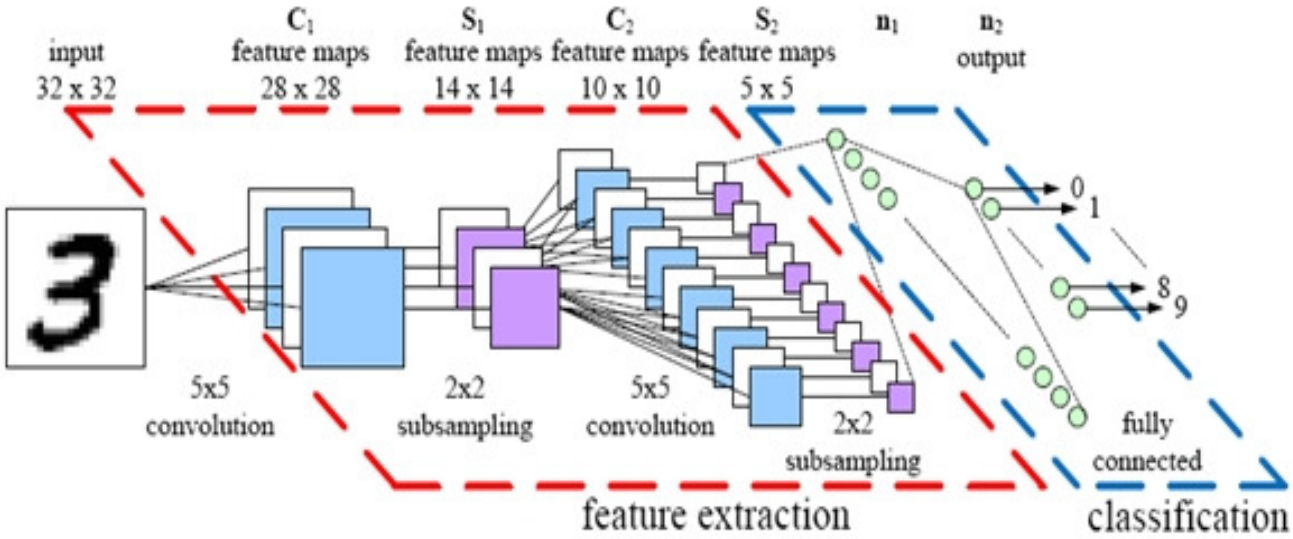
Convolutional Neural Networks

- A feedforward neural network with a special structure.
- Sparse "local" connectivity between layers (except the last output layer). Reduces the number of parameters to be learned.



- Shared weights (like a "global" filter). Helps capture the local properties of the signal (useful for data such as images or time-series)
- Uses a sequence of 2 operations, convolution and pooling (subsampling), applied repeatedly on the input data.

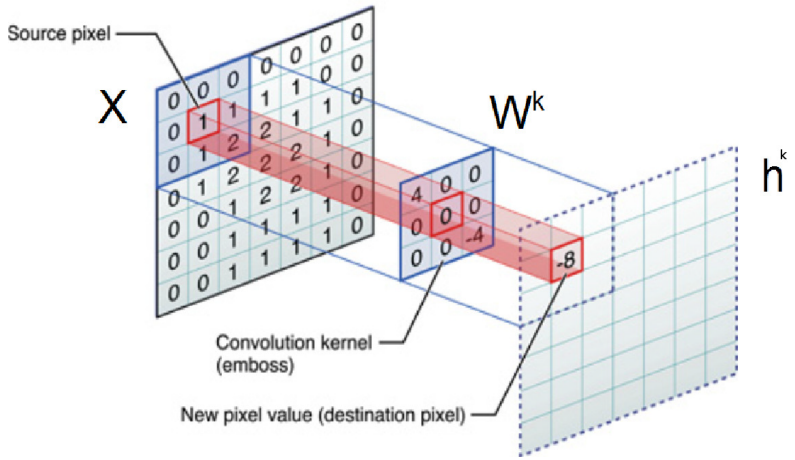
- Convolution : Extract "local" properties of the signal. Uses a set of "filters" that have to be learned (these are the "weights" $\boldsymbol{W}$ between layers).
- Pooling : Downsamples the outputs to reduce the size of representation.



- Note : A nonlinearity is also introduced after the convolution layer.

Convolution

- An operation that captures local (e.g., spatial) properties of a signal:



- Mathematically, the operation is defined as

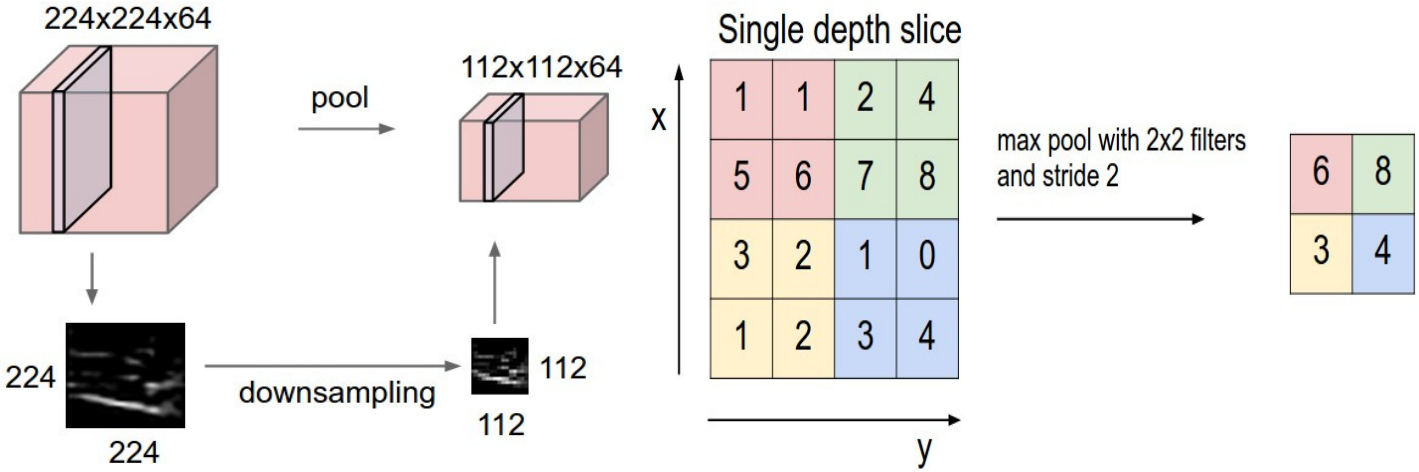
$$h_{ij}^k = f\left(\left(W^k * \mathrm{X}\right)_{ij} + b_k\right),$$

where W^k is a filter, $*$ is the convolution operator, and f is a nonlinearity.

- Usually a number of filters $\left\{W^k\right\}_{k=1}^K$ are applied (each will produce a separate "feature map"). These filters need to be learned.
- Size of these filters also need to be specified.

Pooling / Subsampling

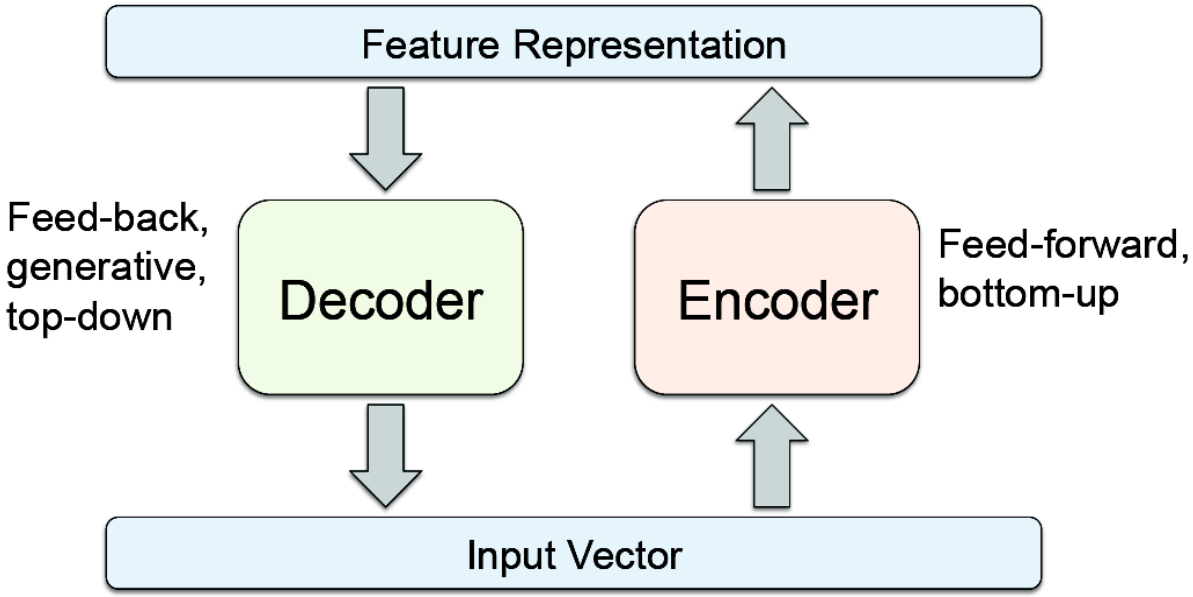
- This operation is used to reduce the size of the representation.



- Max-pooling is commonly used (replacing a block of values by the max value).

Neural Nets for Unsupervised Learning

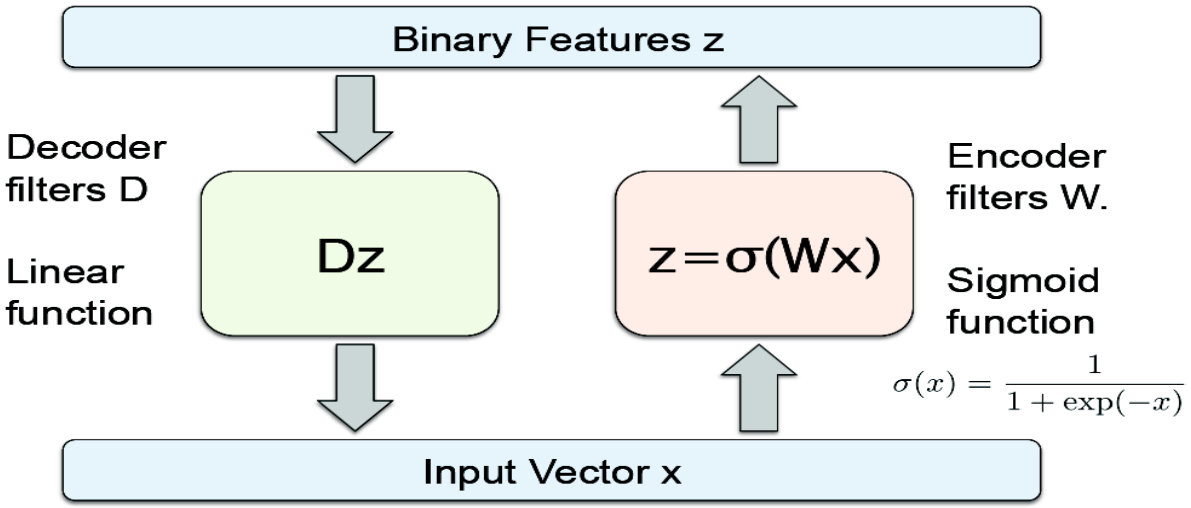
- Autoencoder: A neural net for unsupervised feature extraction.
- Basic principle: Learns an encoding of the inputs so as to recover the original input from the encodings as much as possible.



- Also used to initialize deep learning models (layer-by-layer pre-training).

Autoencoder: An Example

- Real-valued inputs, binary-valued encodings:



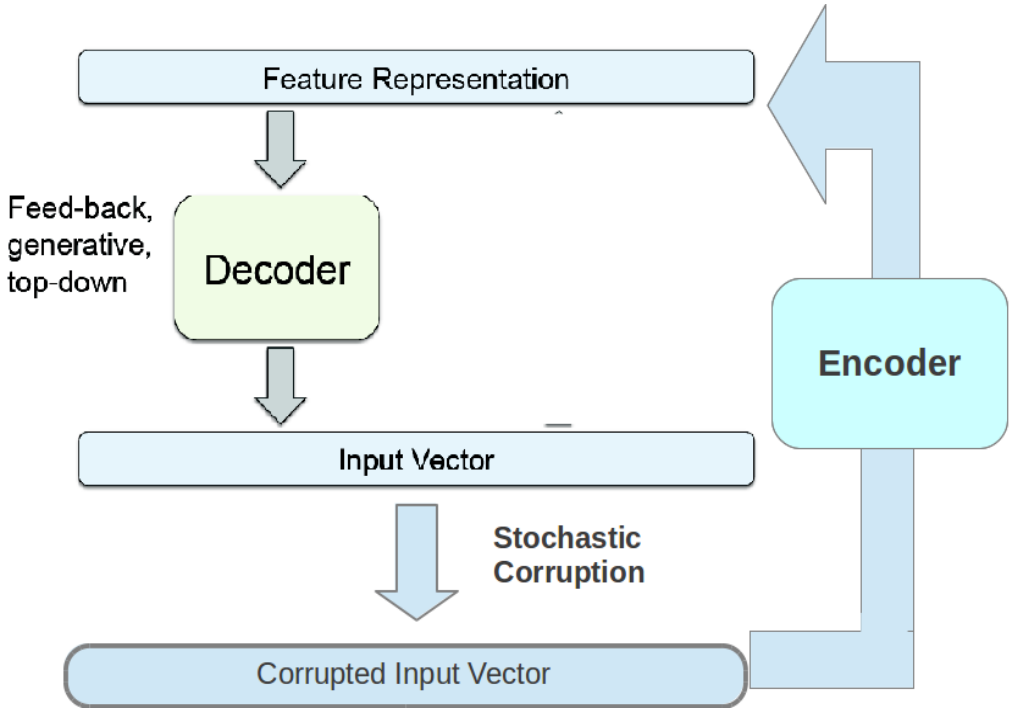
- Sigmoid encoder (parameter matrix W), linear decoder (parameter matrix D), learned via:

$$\arg\min_{D,W} E(D,W) = \sum_{n=1}^{N} \|Dz_n - x_n\|^2 = \sum_{n=1}^{N} \|D\sigma(Wx_n) - x_n\|^2.$$

- If encoder is linear, then autoencoder is equivalent to PCA.

Denoising Autoencoders

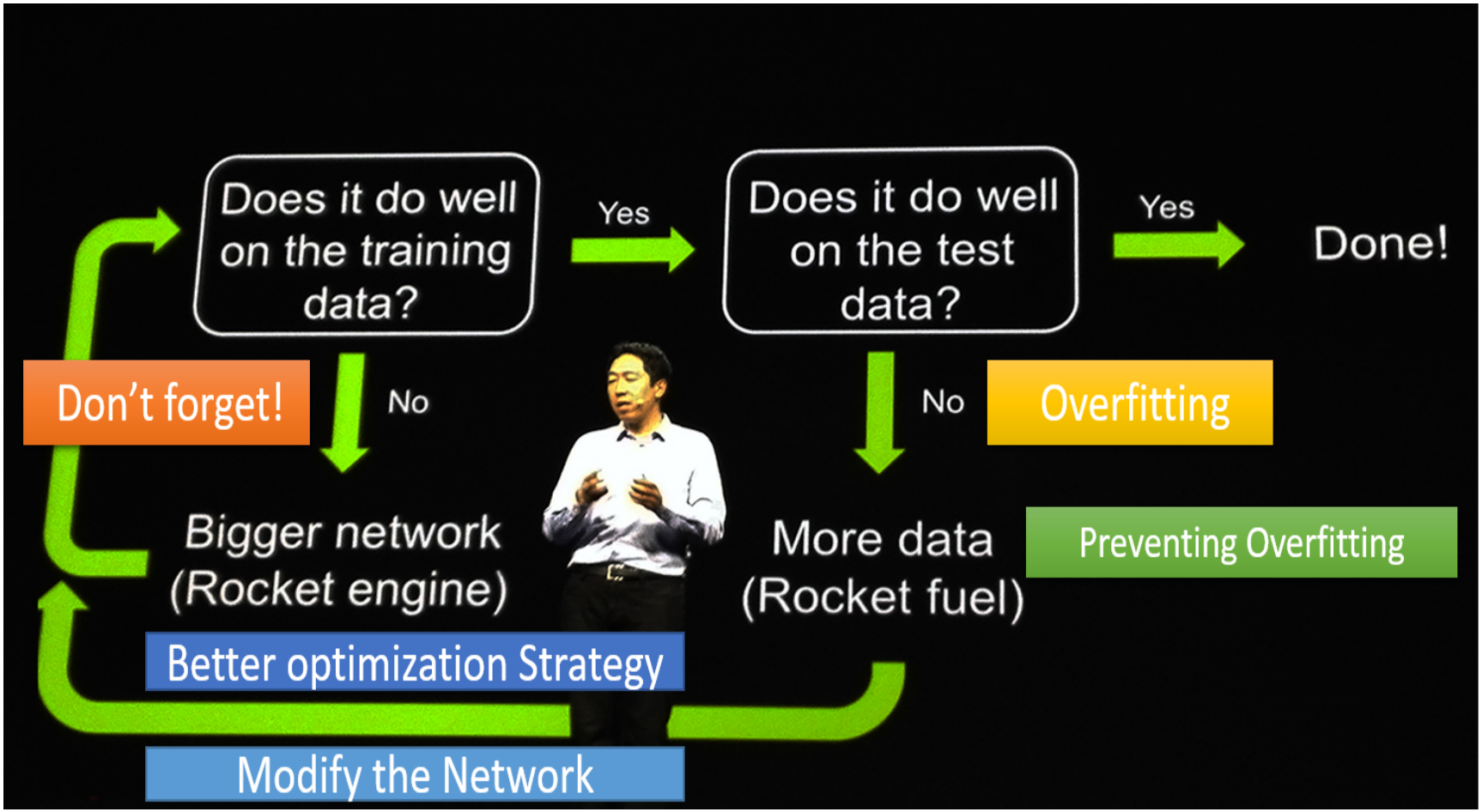
- Idea: introduce stochastic corruption to the input; e.g. hide some features or add Gaussian noise.



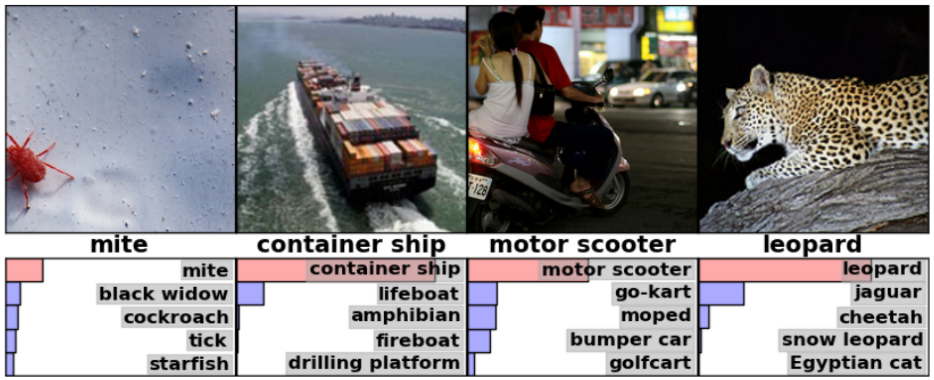
Deep Neural Nets : Some Comments

- Highly effective in learning good feature representations from data in an "end-to-end" manner.
- The objective functions of these models are highly non-convex.
- Training these models is computationally very expensive (GPUs required).
- Training these models can be tricky, need a proper initialization (Soln.: unsupervised layer-wise pre-training).
- Deep learning models can also be probabilistic and generative, e.g., deep belief networks.
- Looked at Autoencoder - Neural network for unsupervised feature extraction.

Recipe for Learning

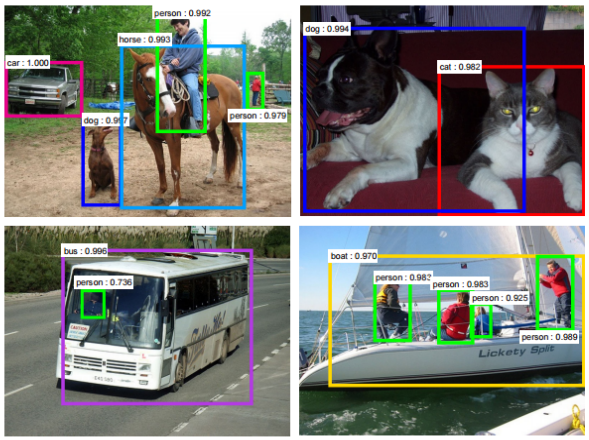


Applications: Deep Learning in Computer Vision



[Krizhevsky 2012]

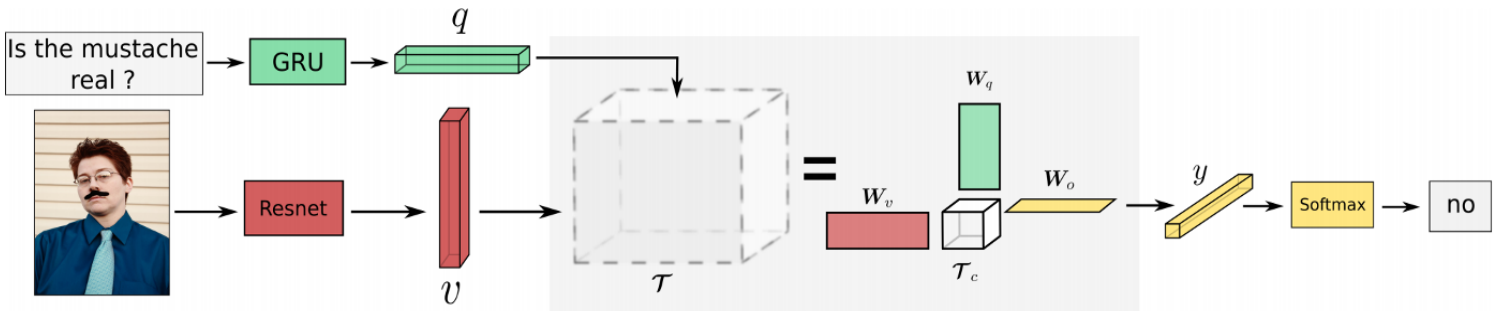
[Ciresan et al. 2013]



[Faster R-CNN - Ren 2015]

[NVIDIA dev blog]

Applications: Deep Learning in Natural Language Processing



[VQA - Mutan 2017]

"man in black shirt is playing guitar."

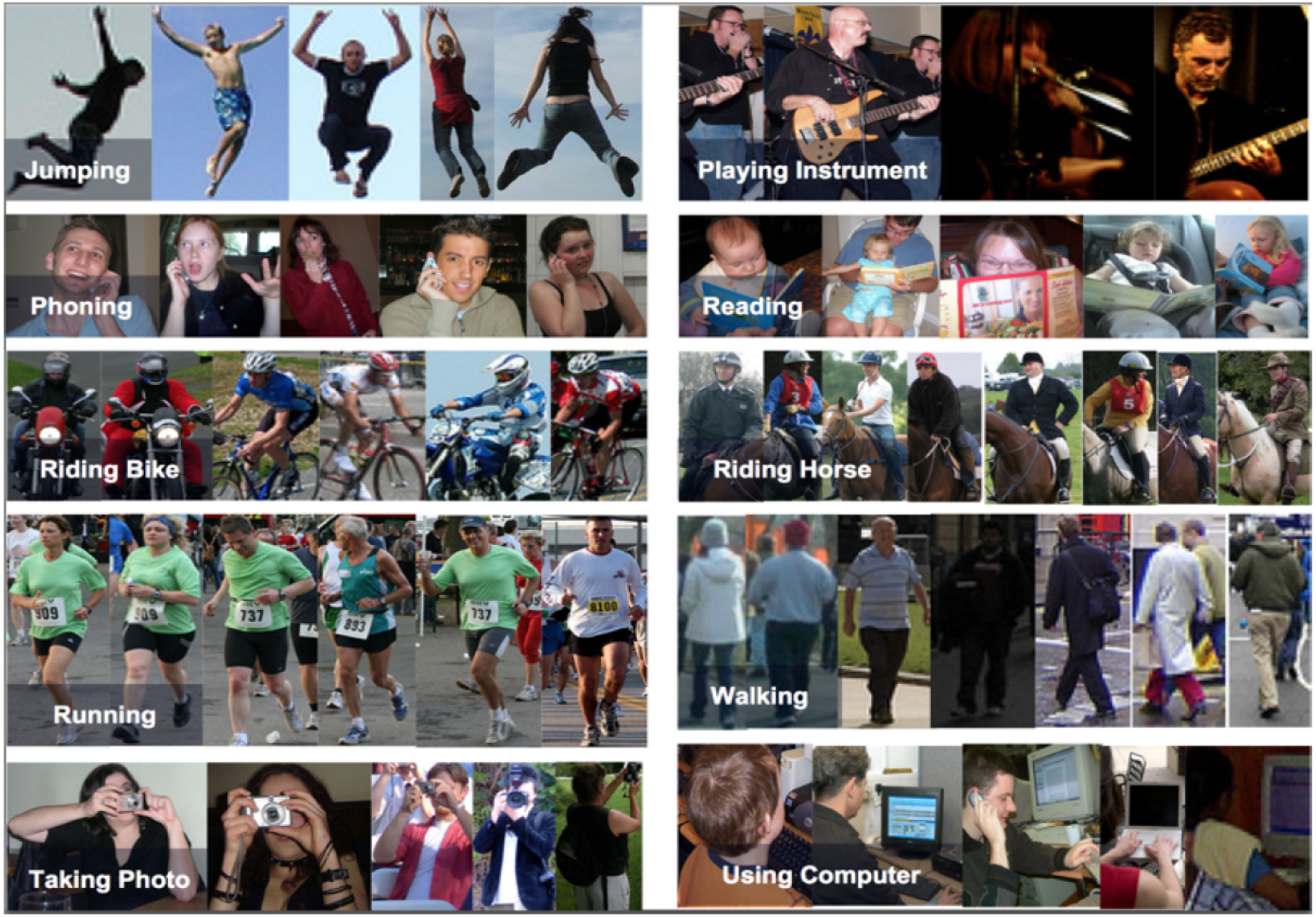
"construction worker in orange safety vest is working on road."

"two young girls are playing with lego toy."

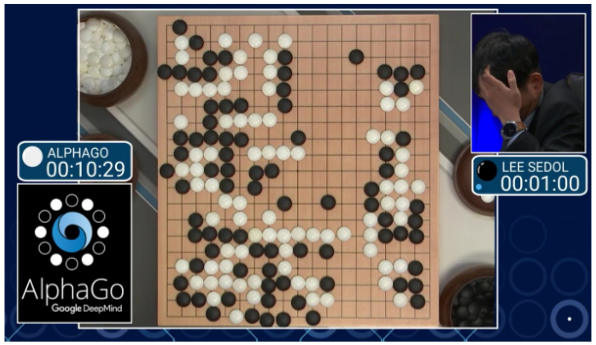
"boy is doing backflip on wakeboard."

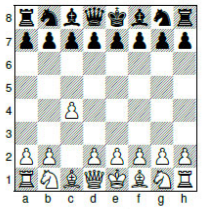
[Karpathy 2015]

Applications: Action recognition



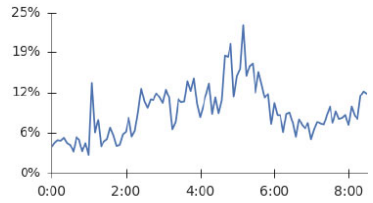
Applications: Deep Learning – Generative Models





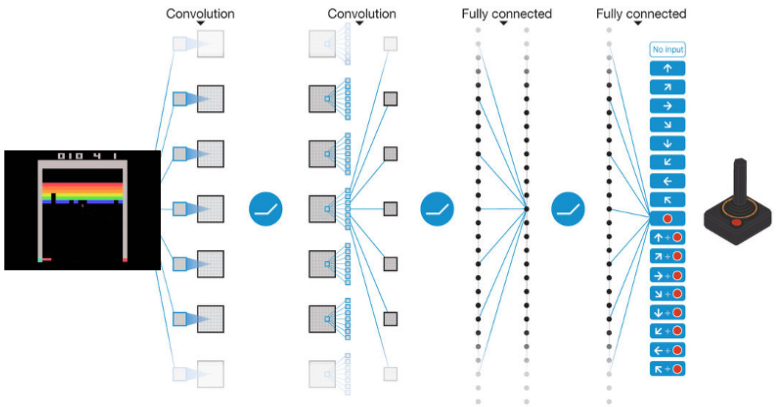
w 20/30/0, b 8/40/2

A10: English Opening

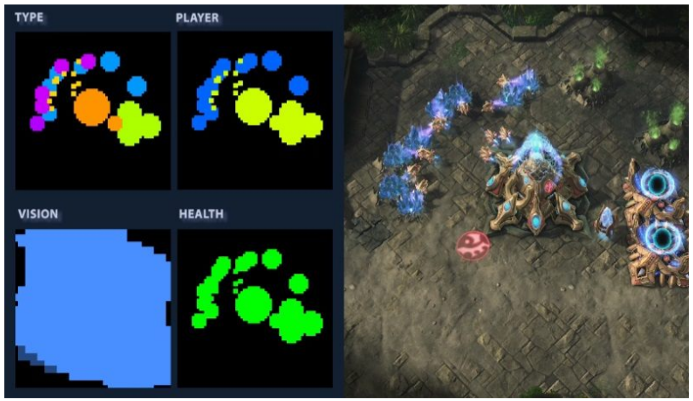


1...e5 g3 d5 cxd5 ♘f6 ♗g2 ♘xd5 ♘f3

[Deepmind AlphaGo / Zero 2017]



[Atari Games - DeepMind 2016]

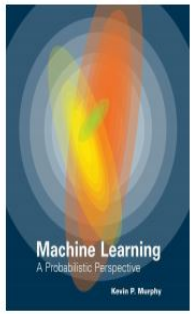


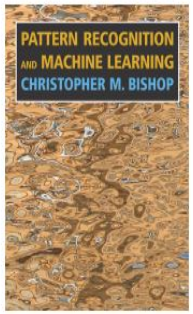
[Starcraft 2 for AI research]

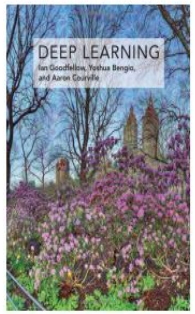
Textbook and References

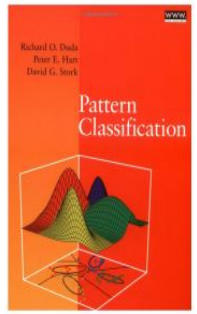
Start with a book of 130 Pages...

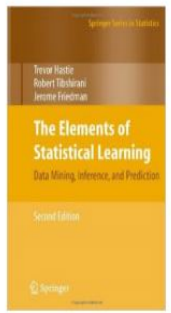
Then you can start reading these books...

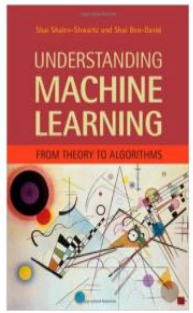


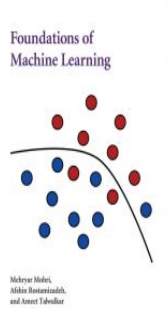












Deep Learning Resources

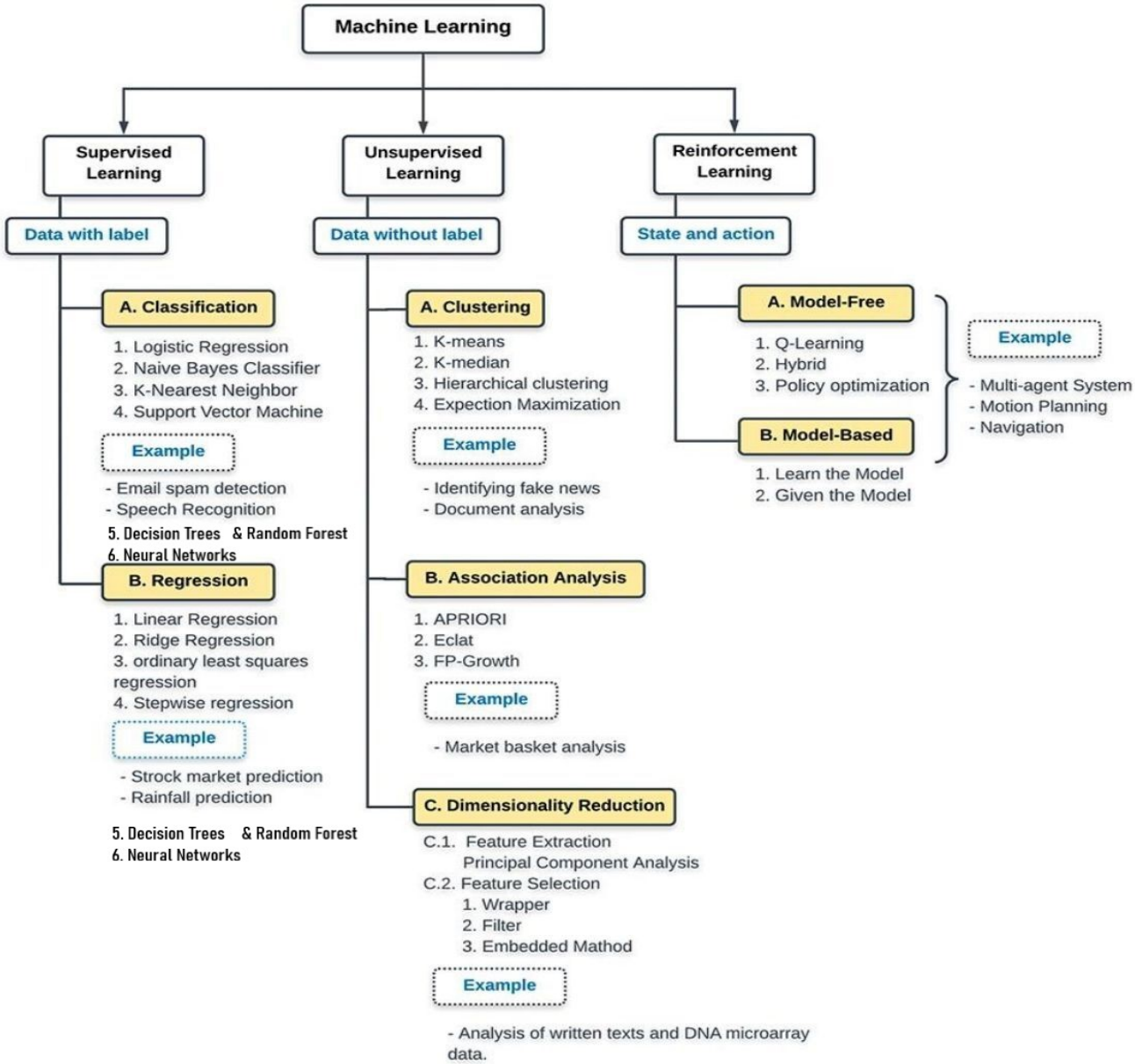
Name	Language	Link	Note
Pylearn2	Python	http://deeplearning.net/software/pylearn2/	A machine learning library built on Theano
Theano	Python	http://deeplearning.net/software/theano/	A python deep learning library
Keras	Python	https://keras.io/	A python deep learning library
Caffe	C++	http://caffe.berkeleyvision.org/	A deep learning framework by Berkeley
Torch	Lua	http://torch.ch/	An open source machine learning framework
Overfeat	Lua	http://cilvr.nyu.edu/doku.php?id=code:start	A convolutional network image processor
Deeplearning4j	Java	http://deeplearning4j.org/	A commercial grade deep learning library
Word2vec	**C**	**https://code.google.com/p/word2vec/**	**Word embedding framework**
GloVe	C	http://nlp.stanford.edu/projects/glove/	Word embedding framework
Doc2vec	C	https://radimrehurek.com/gensim/models/doc2vec.html	Language model for paragraphs and documents
StanfordNLP	**Java**	**http://nlp.stanford.edu/**	**A deep learning-based NLP package**
TensorFlow	**Python**	**http://www.tensorflow.org**	**A deep learning based python library**
PyTorch	**Python**	**https://pytorch.org/**	**A deep learning based python library**

Slide Credit: Bikash Santra, ISI Kolkata.

Other Useful Links

1. Free Datasets: Link
2. Deep Learning Book: Link
3. Deep Learning Course: Link
4. CA Murthy's Lectures: Link
5. Andrew Ng's Lectures: Link
6. Patrick Winston's Lectures: Link
7. Probabilistic ML Course: Link
8. Sebastian Raschka's Blog: Link
9. Radford M Neal's Blog: Link
10. Dive into Deep Learning: Link

Cheatsheet


Machine Learning
Supervised Learning
Unsupervised Learning
Reinforcement Learning
Data with label
Data without label
State and action
A. Classification
1. Logistic Regression
2. Naive Bayes Classifier
3. K-Nearest Neighbor
4. Support Vector Machine
Example
- Email spam detection
- Speech Recognition
5. Decision Trees & Random Forest
6. Neural Networks
B. Regression
1. Linear Regression
2. Ridge Regression
3. ordinary least squares regression
4. Stepwise regression
Example
- Strock market prediction
- Rainfall prediction
5. Decision Trees & Random Forest
6. Neural Networks
A. Clustering
1. K-means
2. K-median
3. Hierarchical clustering
4. Expection Maximization
Example
- Identifying fake news
- Document analysis
B. Association Analysis
1. APRIORI
2. Eclat
3. FP-Growth
Example
- Market basket analysis
C. Dimensionality Reduction
C.1. Feature Extraction
Principal Component Analysis
C.2. Feature Selection
1. Wrapper
2. Filter
3. Embedded Mathod
Example
- Analysis of written texts and DNA microarray data.
A. Model-Free
1. Q-Learning
2. Hybrid
3. Policy optimization
B. Model-Based
1. Learn the Model
2. Given the Model
Example
- Multi-agent System
- Motion Planning
- Navigation