# DS 4400

## Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

March 12 2019

# Logistics

- Feedback on project proposal in Gradescope
- Please start working on your projects!
- Project Milestone due on March 25
  - 2-3 pages describing progress so far
  - Any encountered challenges
- Project Presentations will be last 2 classes
  - April 11 and 16
- Exam on Thursday, March 28
  - Review on Tuesday, March 26
  - Office hours on Wednesday, March 27
- HW4 will be out at the end of the week

# Outline

- Review SVM
- Review traditional classifiers
- Deep Learning
  - Motivation
  - Goals
- Deep Learning as representation learning
- Categories of neural networks
- Feed-Forward Neural Networks

# Review Naïve Bayes

- Density Estimators can estimate joint probability distribution from data
- Risk of overfitting and curse of dimensionality
- Naïve Bayes assumes that features are independent given labels
  - Reduces the complexity of density estimation
  - Even though the assumption is not always true, Naïve Bayes works well in practice
- Applications: text classification with bag-of-words representation
  - Naïve Bayes becomes a linear classifier

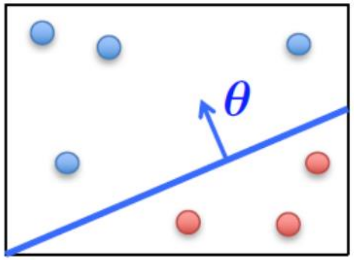
Generative model

# Recall:

## Linear classifiers

- **Linear classifiers**: represent decision boundary by hyperplane

$$\boldsymbol{\theta} = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_d \end{bmatrix} \quad \boldsymbol{x}^\mathsf{T} = \begin{bmatrix} 1 & x_1 & \dots & x_d \end{bmatrix}$$



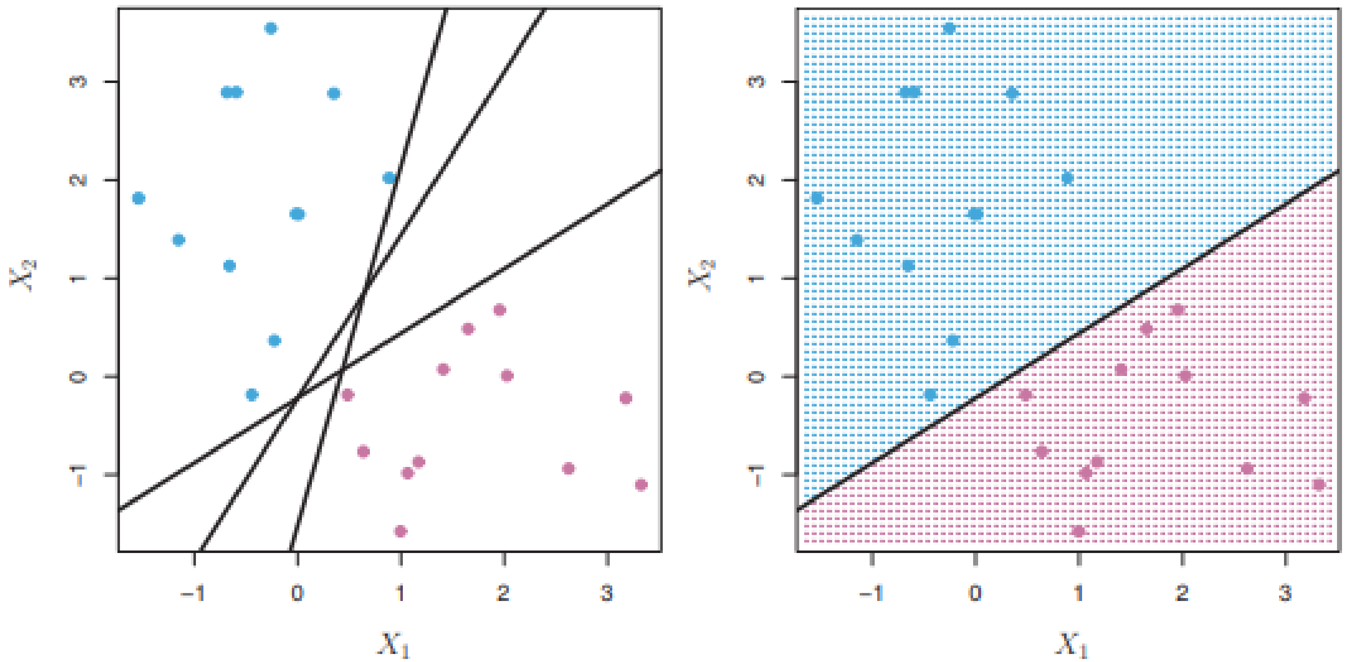

All the points x on the hyperplane satisfy: $\theta^T x = 0$

$$h(\boldsymbol{x}) = \text{sign}(\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x}) \quad \text{where} \quad \text{sign}(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ -1 & \text{if } z < 0 \end{cases}$$

– Note that: $\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x} > 0 \implies y = +1$

$\boldsymbol{\theta}^\mathsf{T}\boldsymbol{x} < 0 \implies y = -1$

# Separating hyperplane




$$\boxed{y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0}$$

For all training data $x^{(i)}, y^{(i)}$, $i \in \{1, \ldots, n\}$

# From separating hyperplane to classifier

- Training data $x^{(1)}, \dots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \dots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y^{(i)} \in \{-1,1\}$
- Let $\theta_0, \dots, \theta_d$ (will be learned) such that:

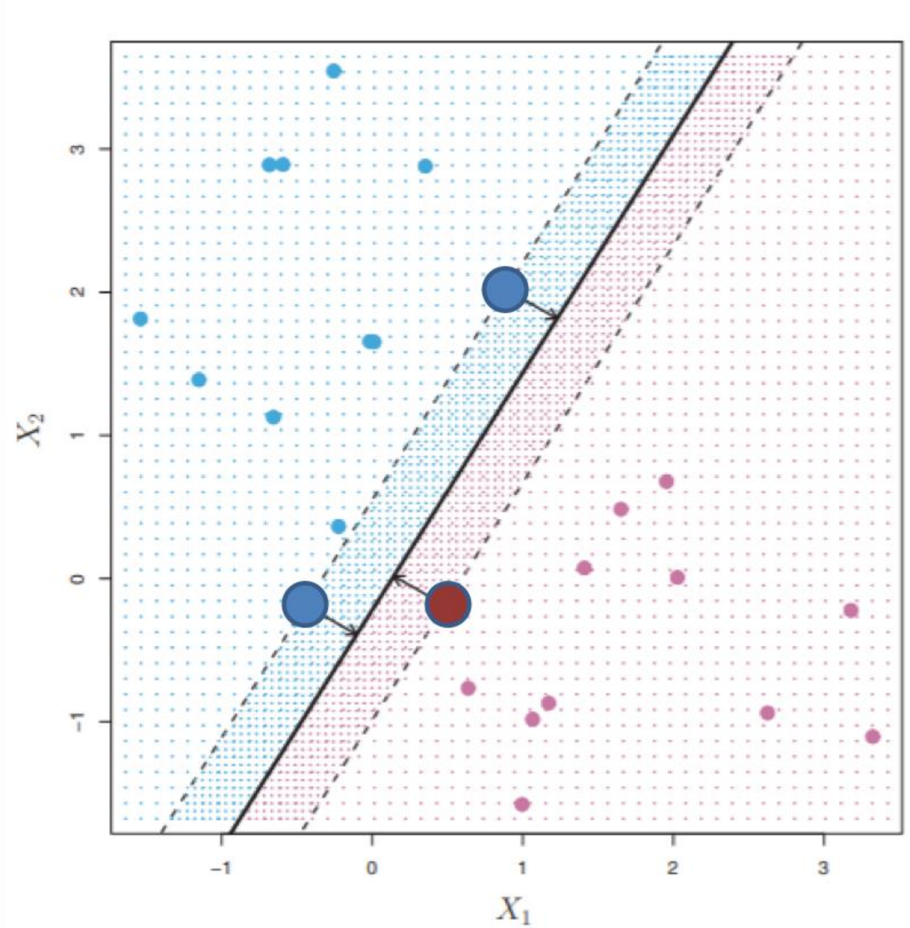
$$y^{(i)}(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}) > 0$$

- Classifier

$$f(z) = \mathrm{sign}(\theta_0 + \theta_1 z_1 + \cdots \theta_d z_d) = \mathrm{sign}(\theta^T \mathrm{z})$$

- Classify new test point $x'$
  - If $f(x') > 0$ predict y’$= 1$
  - Otherwise predict y’$= -1$

# Support Vectors (informally)




- Support vectors = points "closest" to hyperplane
- If support vectors change, classifier changes
- If other points change, no effect on classifier

# Maximum margin classifier

- Training data $x^{(1)}, \ldots, x^{(n)}$ with $x^{(i)} = \left(x_1^{(i)}, \ldots, x_d^{(i)}\right)^{\mathrm{T}}$
- Labels are from 2 classes: $y_i \in \{-1,1\}$

maximize M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M \ \forall i$$

$$\left|\left|\theta\right|\right|_2 = 1$$

Normalization constraint

Each point is on the right side of hyper-plane at distance $\geq M$

# Support vector classifier

- Allow for small number of mistakes on training data
- Obtain a more robust model

max M

$$y^{(i)}\left(\theta_0 + \theta_1 x_1^{(i)} + \cdots \theta_d x_d^{(i)}\right) \geq M(1 - \epsilon_i)\forall i$$

$$\left||\theta|\right|_2 = 1$$

$$\epsilon_i \geq 0, \sum_i \epsilon_i = C$$

Slack

Error Budget (Hyper-parameter)

# Properties

- Maximum margin classifier
  - Classifier of maximum margin
  - For linearly separable data
- Support vector classifier
  - Allows some slack and sets a total error budget (hyper-parameter)
- For both, final classifier on a point is a linear combination of inner product of point with support vectors
  - Efficient to evaluate

# Kernels

- Support vector classifier
  - $h(z) = \theta_0 + \sum_{i \in S} \alpha_i < z, x^{(i)} >$
    $= \theta_0 + \sum_{i \in S} \alpha_i \sum_{j=1} z_j x_j^{(i)}$
  - S is set of support vectors
  - Replace with $h(z) = \theta_0 + \sum_{i \in S} \alpha_i K(z, x^{(i)})$

Any kernel function!

- What is a kernel?
  - Function that characterizes similarity between 2 observations
  - $K(a, b) = < a, b > = \sum_j a_j b_j$ linear kernel!
  - The closer the points, the larger the kernel
- Intuition
  - The closest support vectors to the point play larger role in classification

# Kernels

- Linear kernels
  - $K(a,b) =< a,b > = \sum_i a_i b_i$
- Polynomial kernel of degree *m*
  - $K(a,b) = \left(1 + \sum_{i=0}^{d} a_i b_i\ \right)^m$
- Radial Basis Function (RBF) kernel (or Gaussian)
  - $K(a,b) = \exp\left(-\gamma \sum_{i=0}^{d} (a_i - b_i)^2\right)$
- Support vector machine classifier
  - $\mathrm{h}(z) = \theta_0 + \sum_{i \in S} \alpha_i K(z, x^{(i)})$

# Kernel Example

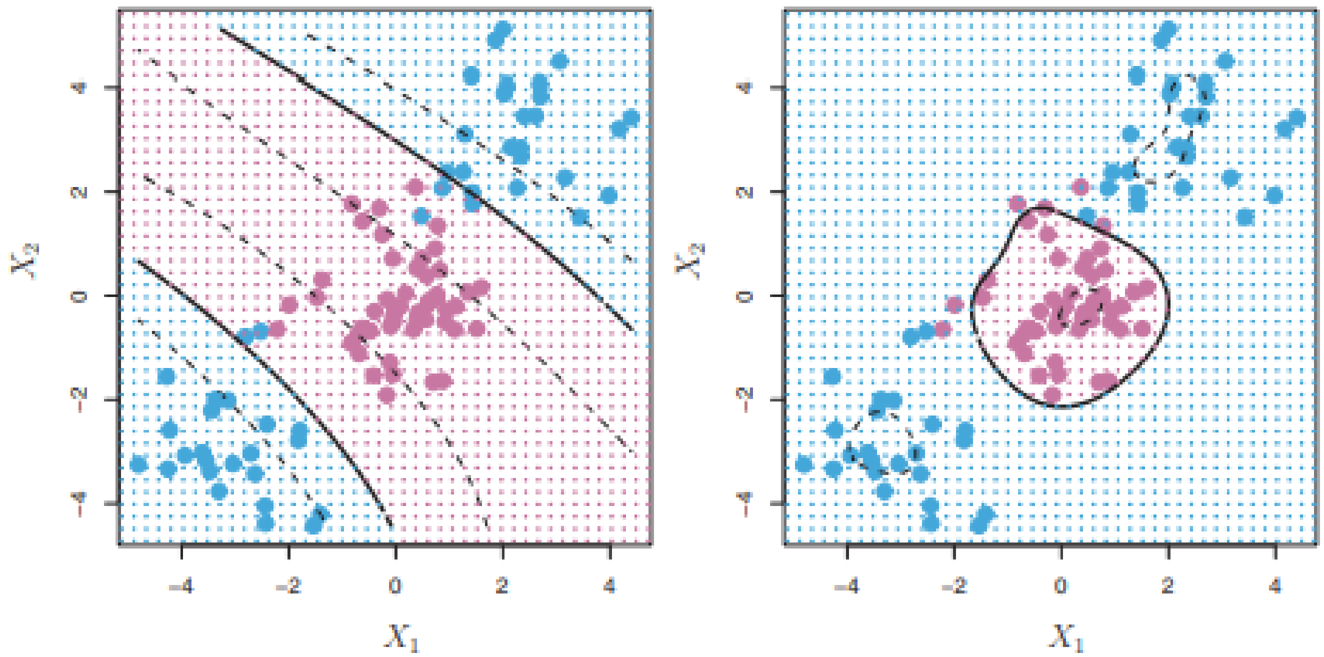



**FIGURE 9.9.** Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

# Classification axes

- Linearity of decision boundary
  - Linear models: logistic regression, perceptron, LDA, SVM
  - Non-linear models: kNN, decision trees, ensembles
- Generative models
  - Discriminative: logistic regression, perceptron, SVM, kNN, decision trees, ensembles
  - Generative: LDA, Naïve Bayes
- Training procedure
  - Gradient descent: logistic regression, SVM, perceptron
  - Probabilistic: LDA, Naïve Bayes
  - Greedy: decision trees, random forests

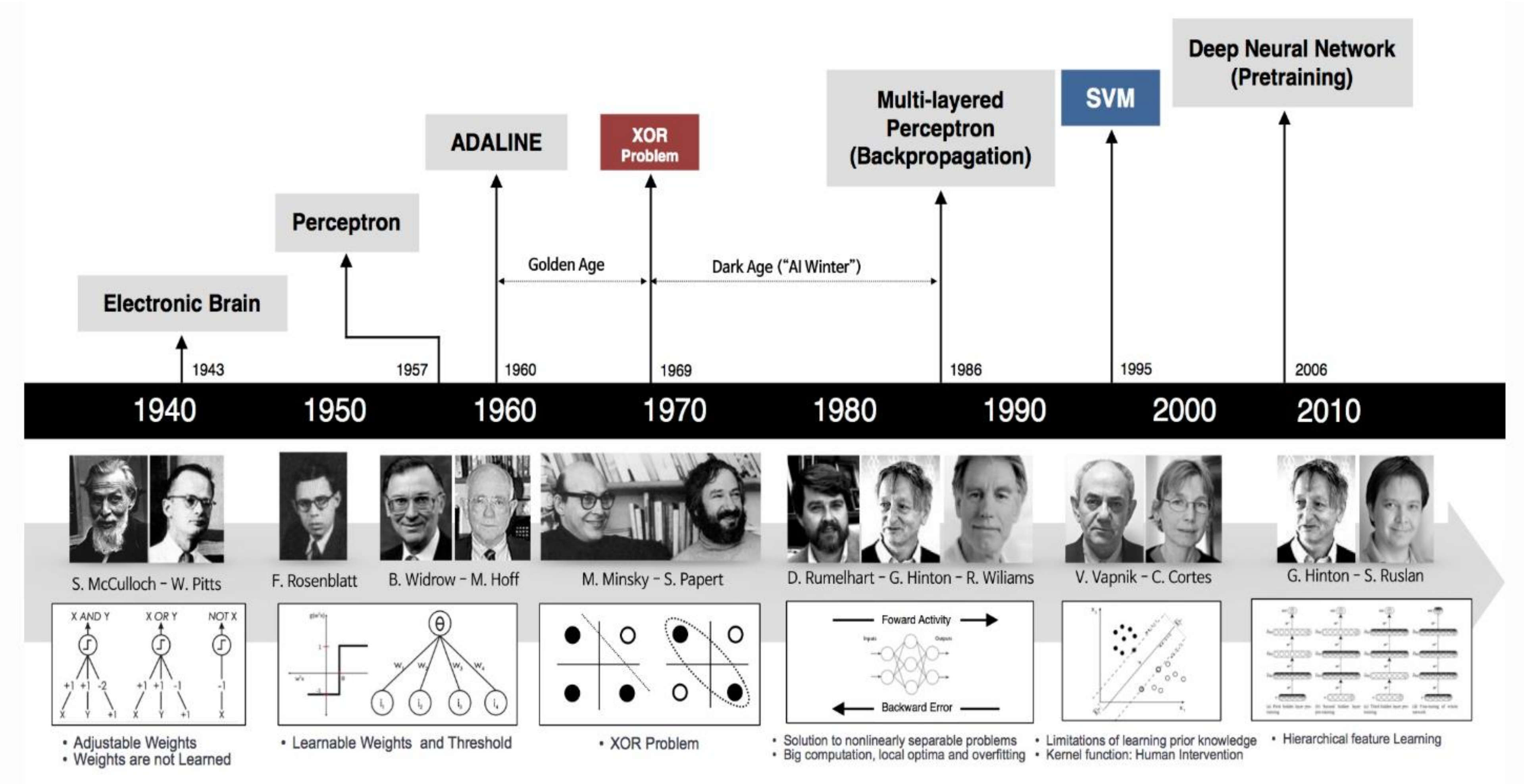
# Comparing classifiers

| Algorithm | Interpretable | Model size | Predictive accuracy | Training time | Testing time |
|---|---|---|---|---|---|
| Logistic regression | High | Small | Lower | Low | Low |
| kNN | Medium | Large | Lower | No training | High |
| LDA | Medium | Small | Lower | Low | Low |
| Decision trees | High | Medium | Lower | Medium | Low |
| Ensembles | Low | Large | High | High | High |
| Naïve Bayes | Medium | Small | Lower | Medium | Low |
| SVM | Medium | Small | High | High | Low |
| Neural Networks | Low | Large | High | High | Low |

# Roadmap to End-of-Semester

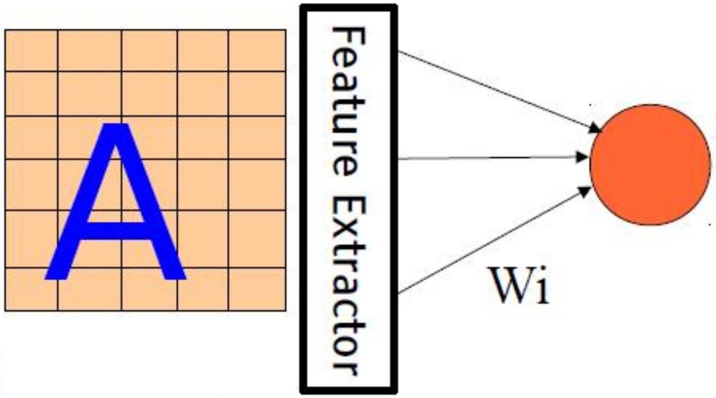
- Deep Learning
  - Motivation
  - Feed-Forward Neural Networks
  - Training by backpropagation
  - Convolutional and Recurrent Neural Networks
- Unsupervised learning
  - Principal Component Analysis (PCA)
  - Feature representation (Autoencoders)
  - Clustering (k-means, Hierarchical Clustering)
- Adversarial learning

# History of Deep Learning

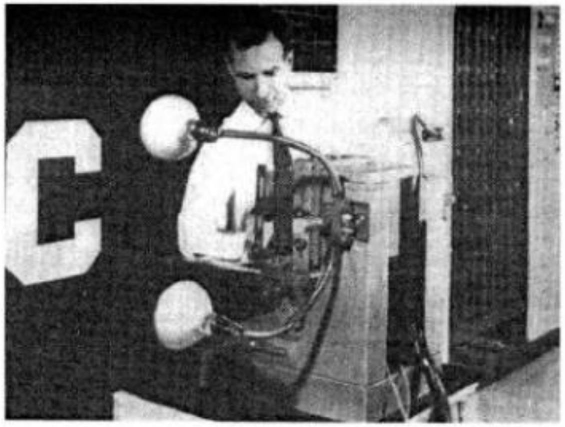


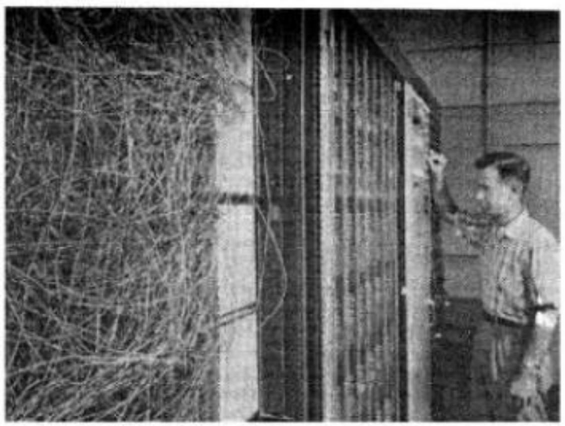
# Before 2013

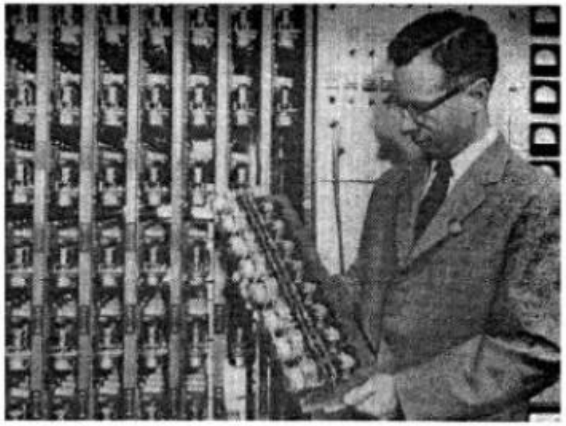
- The first learning machine: the Perceptron
  - Built at Cornell in 1960
- The Perceptron was a linear classifier on top of a simple feature extractor
- The vast majority of practical applications of ML today use glorified linear classifiers or glorified template matching.
- Designing a feature extractor requires considerable efforts by experts.




$$y = sign\left(\sum_{i=1}^{N} W_i F_i(X) + b\right)$$

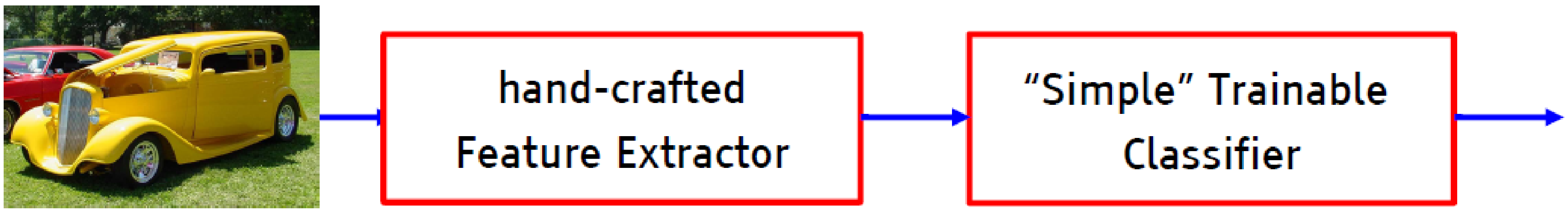




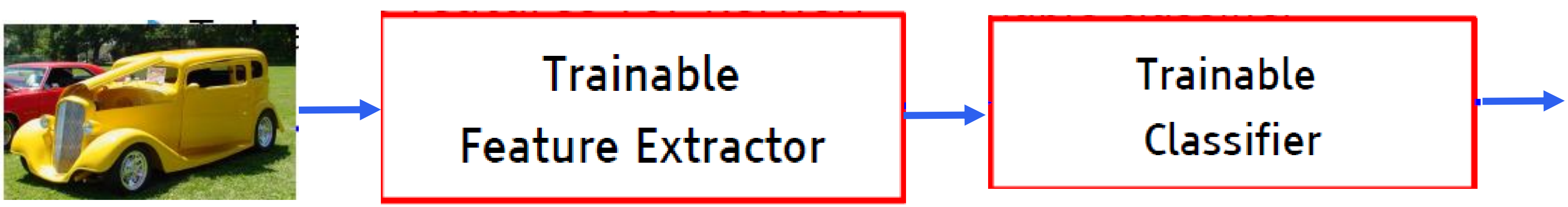


# Deep Learning

- **The traditional model of pattern recognition (since the late 50's)**
  - Fixed/engineered features (or fixed kernel) + trainable classifier

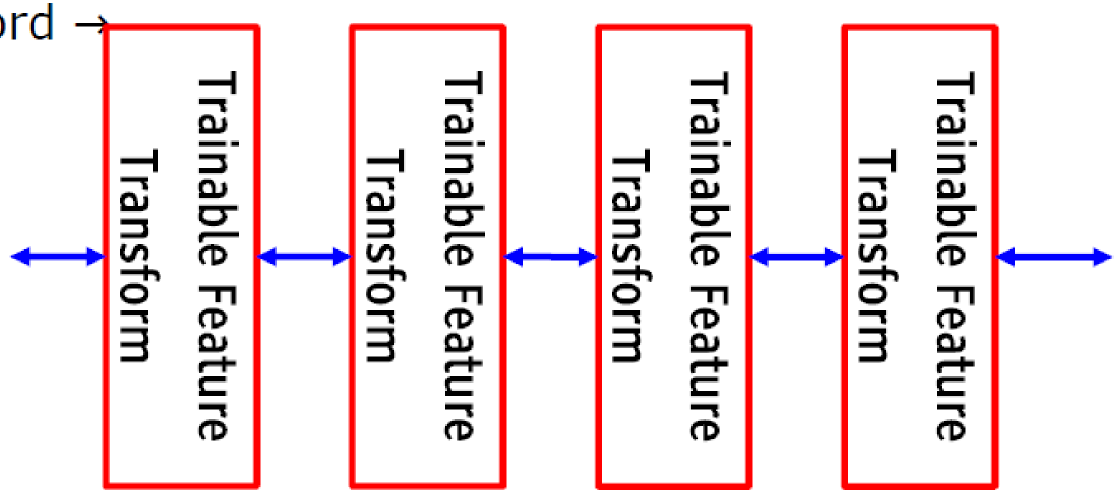



- End-to-end learning / Feature learning / Deep learning

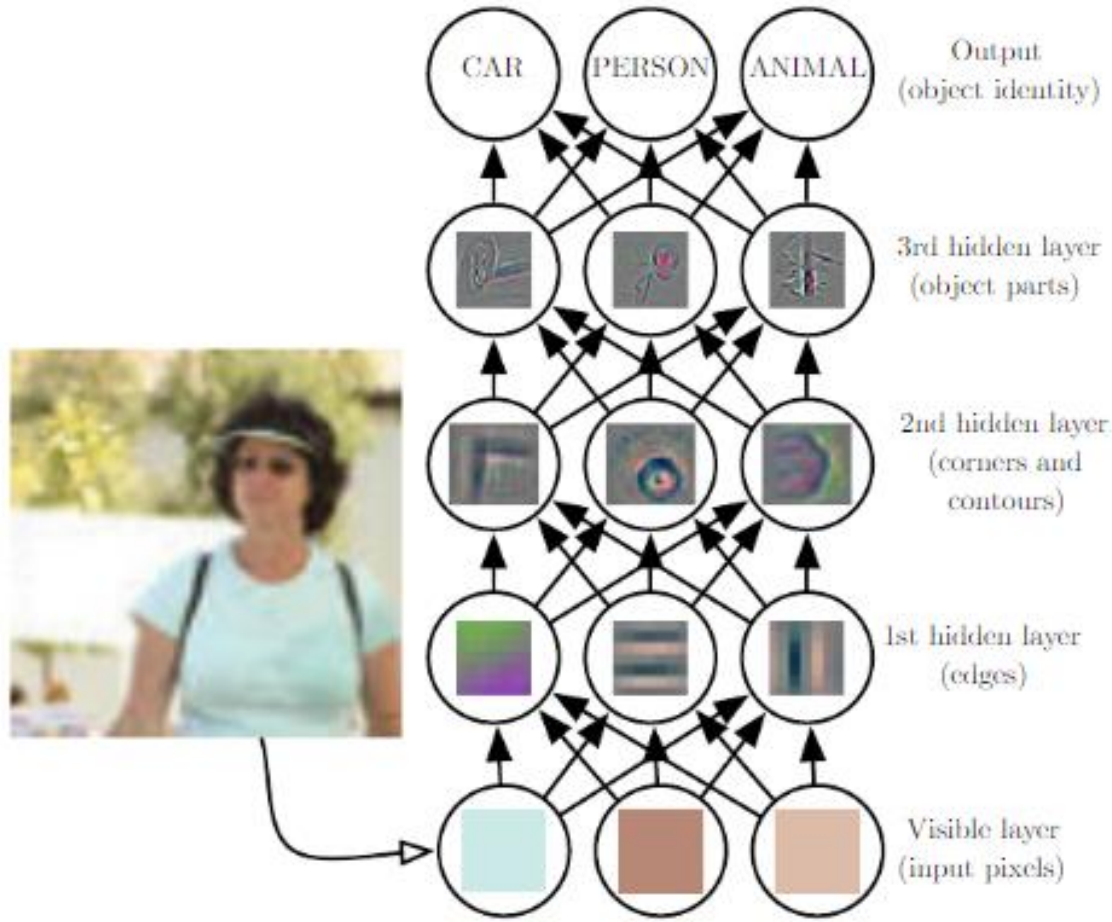


# Trainable Feature Hierarchy

- Hierarchy of representations with increasing level of abstraction
- Each stage is a kind of trainable feature transform
- Image recognition
  - Pixel → edge → texton → motif → part → object
- Text
  - Character → word → word group → clause → sentence → story
- Speech
  - Sample → spectral band → sound → ... → phone → phoneme → word →



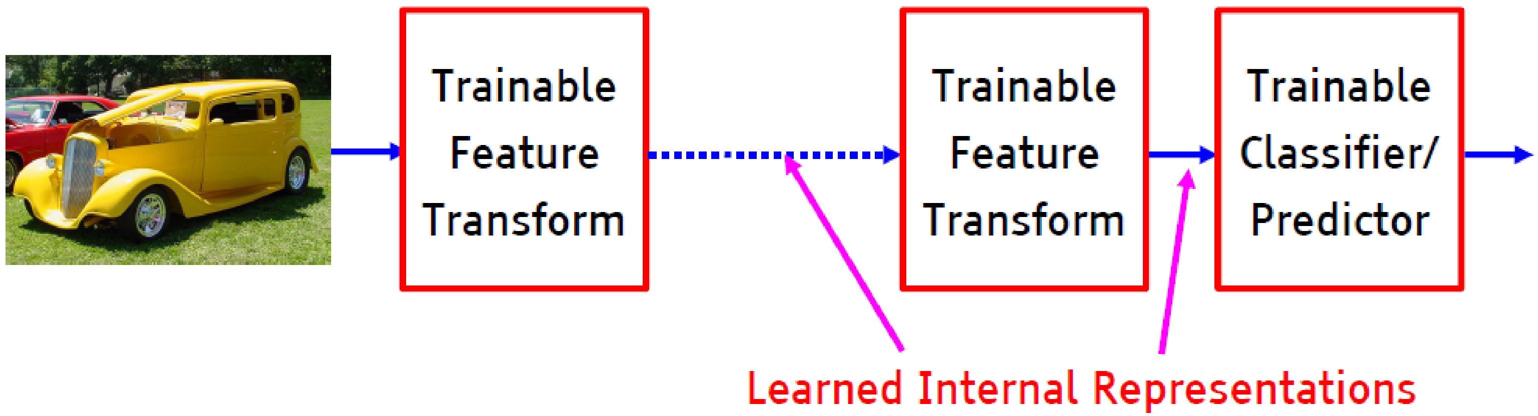
# Learning Representations




Deep Learning addresses the problem of learning hierarchical representations

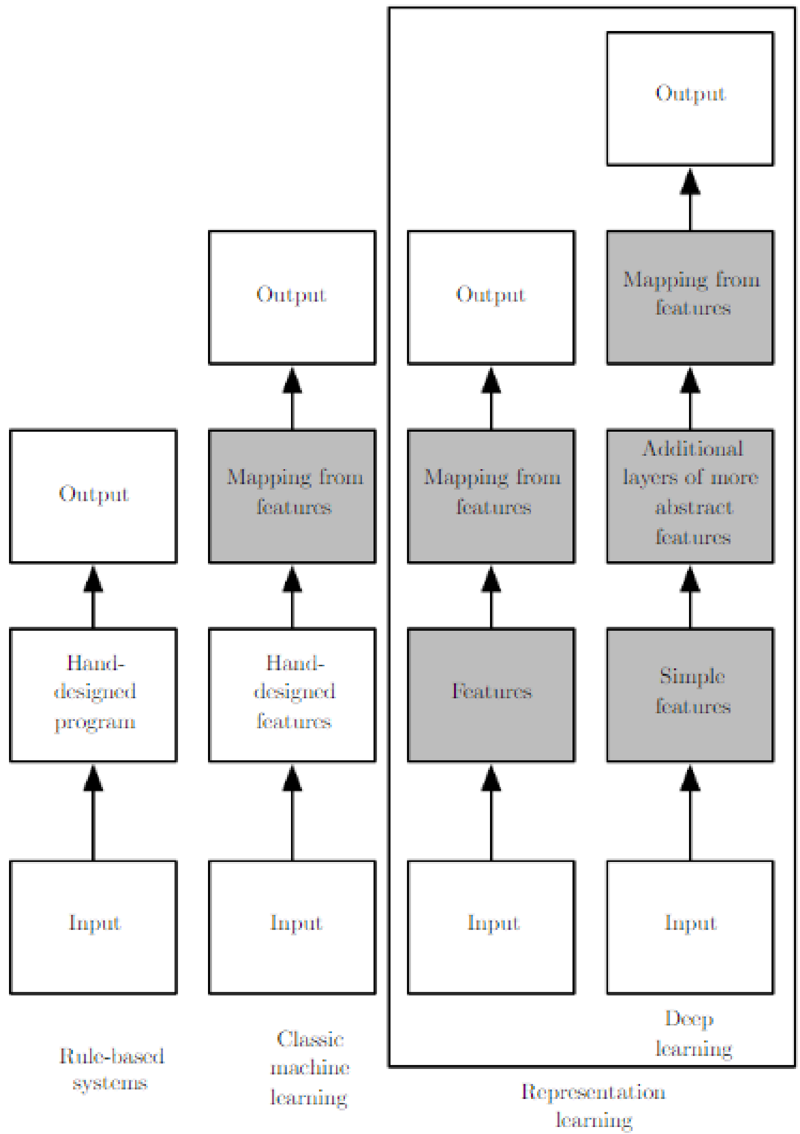
# End-to-end learning

- **A hierarchy of trainable feature transforms**
  - Each module transforms its input representation into a higher-level one.
  - High-level features are more global and more invariant
  - Low-level features are shared among categories



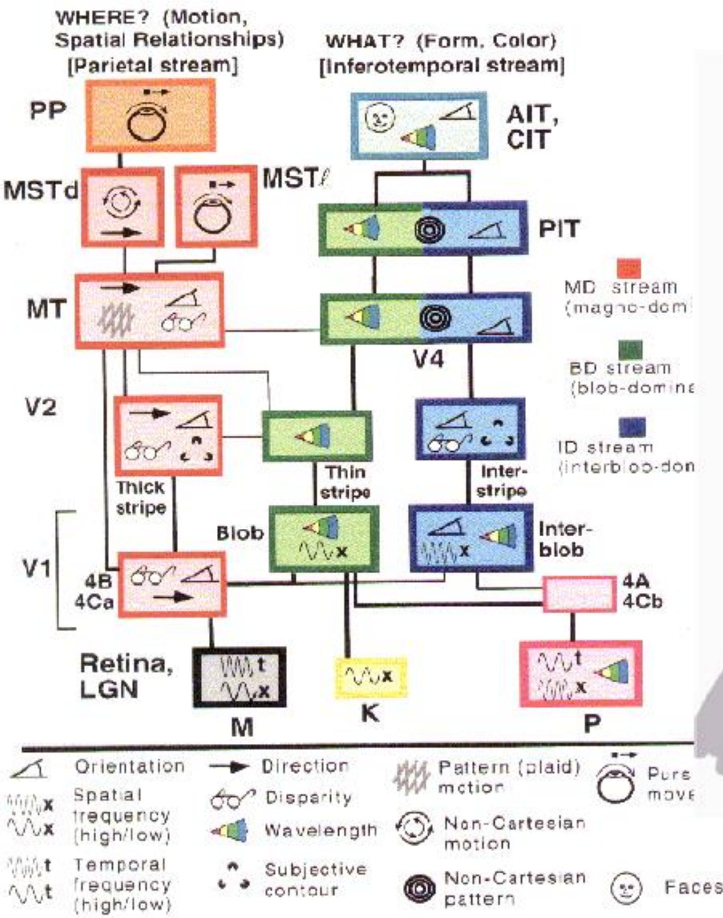

- **How can we make all the modules trainable and get them to learn appropriate representations?**

# Deep Learning vs Traditional Learning

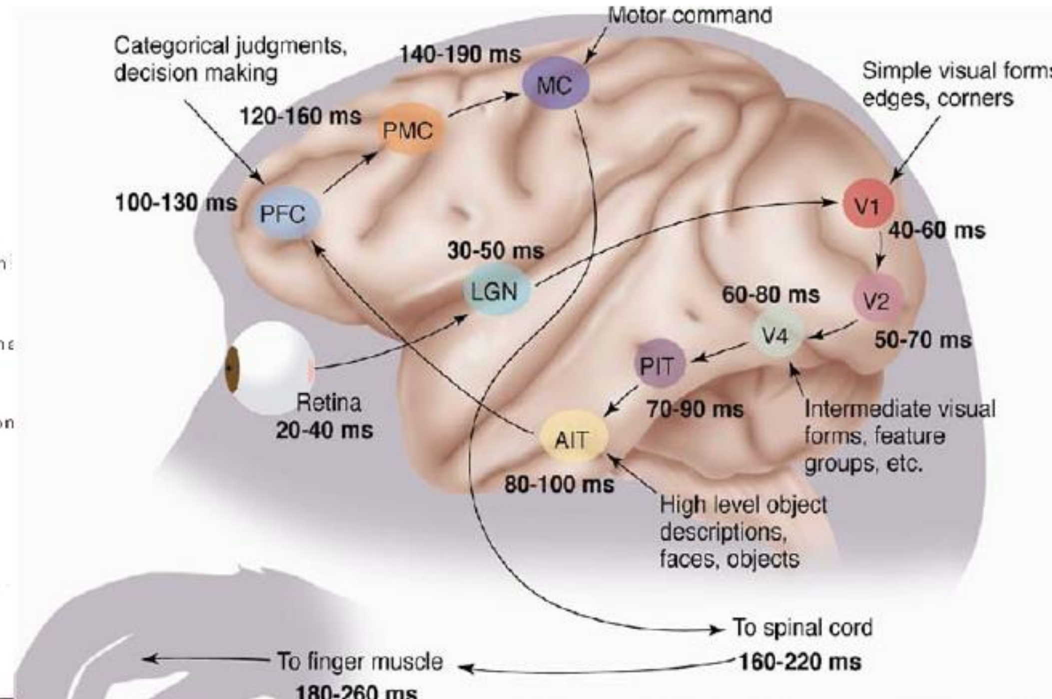


# The Visual Cortex is Hierarchical

- The ventral (recognition) pathway in the visual cortex has multiple stages
- Retina - LGN - V1 - V2 - V4 - PIT - AIT ....
- Lots of intermediate representations




[Gallant & Van Essen]

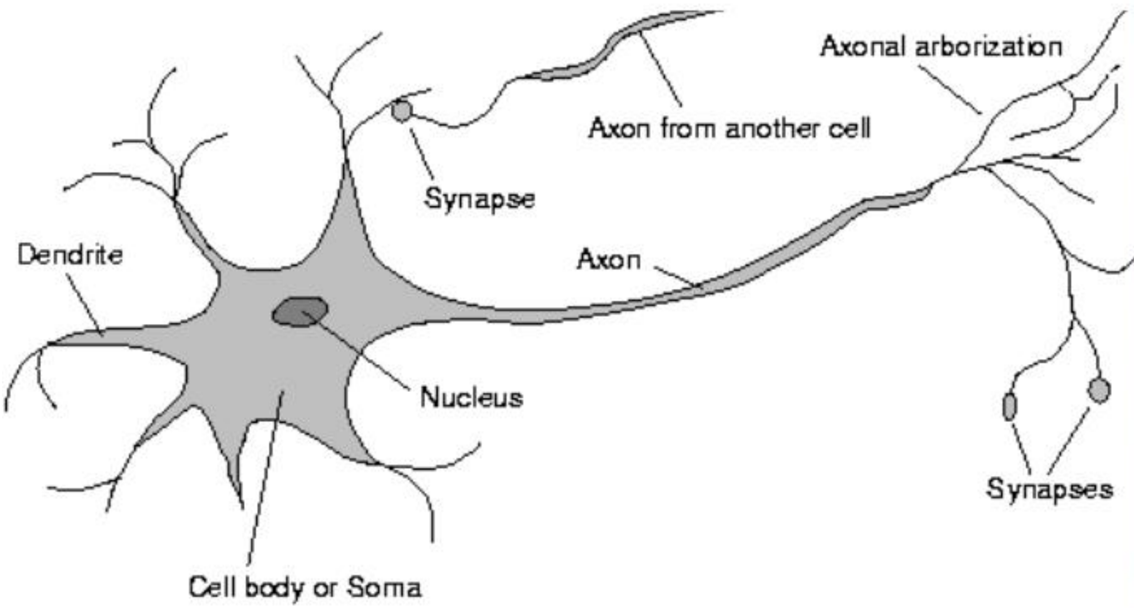


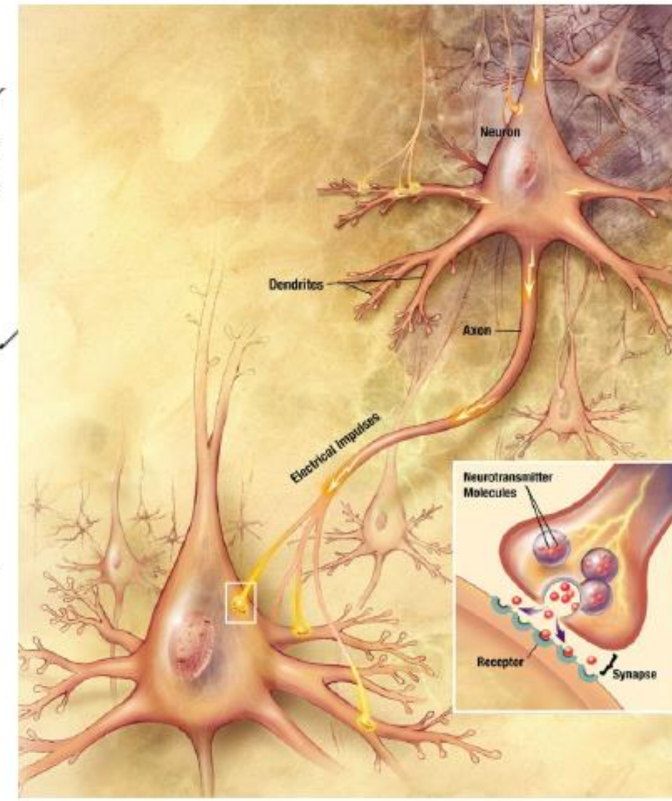

[picture from Simon Thorpe]

# Neural Function

- Brain function (thought) occurs as the result of the firing of **neurons**
- Neurons connect to each other through **synapses**, which propagate **action potential** (electrical impulses) by releasing **neurotransmitters**
  - Synapses can be **excitatory** (potential-increasing) or **inhibitory** (potential-decreasing), and have varying **activation thresholds**
  - Learning occurs as a result of the synapses' **plasticicity**: They exhibit long-term changes in connection strength
- There are about $10^{11}$ neurons and about $10^{14}$ synapses in the human brain!

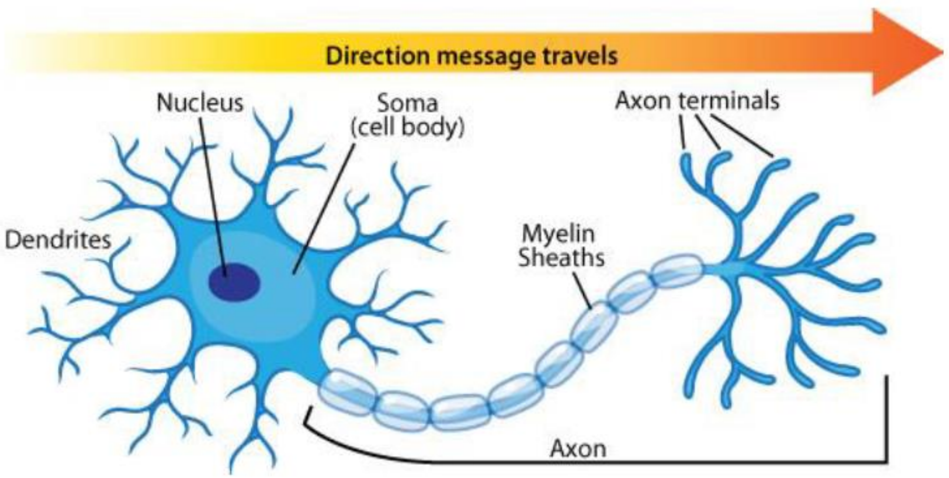
# Biology of a Neuron



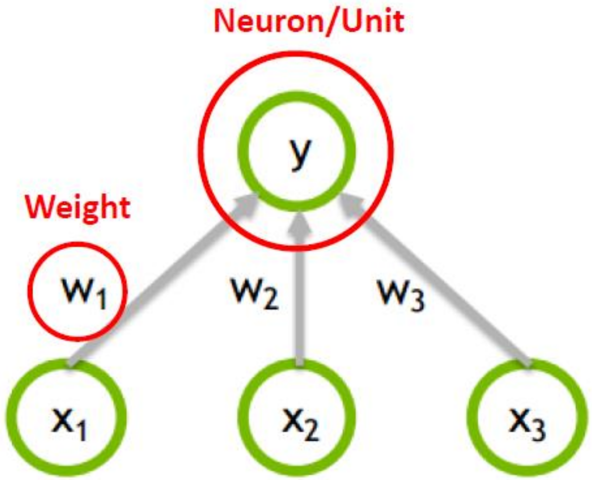



# Analogy to Human Brain

## Human Brain




**Biological Neuron**




$$y=F(w_1x_1+w_2x_2+w_3x_3)$$

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

**Artificial Neuron**

# Comparison of computing power

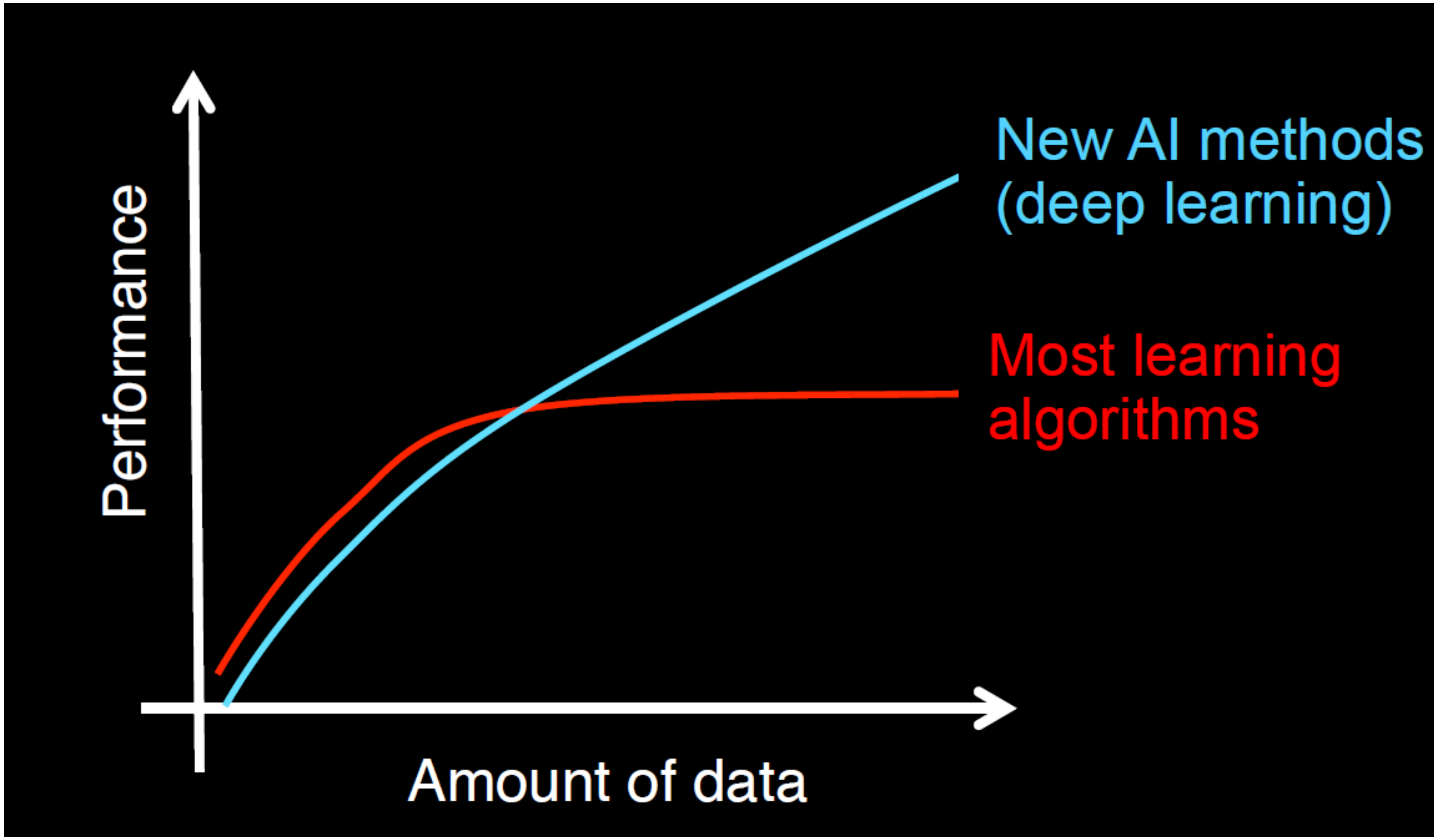
| INFORMATION CIRCA 2012 | Computer | Human Brain |
|---|---|---|
| **Computation Units** | 10-core Xeon: $10^9$ Gates | $10^{11}$ Neurons |
| **Storage Units** | $10^9$ bits RAM, $10^{12}$ bits disk | $10^{11}$ neurons, $10^{14}$ synapses |
| **Cycle time** | $10^{-9}$ sec | $10^{-3}$ sec |
| **Bandwidth** | $10^9$ bits/sec | $10^{14}$ bits/sec |

- Computers are way faster than neurons…
- But there are a lot more neurons than we can reasonably model in modern digital computers, and they all fire in parallel
- Neural networks are designed to be massively parallel
- The brain is effectively a billion times faster

# Neural Networks

- Origins: Algorithms that try to mimic the brain.
- Very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications
- Artificial neural networks are not nearly as complex or intricate as the actual brain structure

# Performance of Deep Learning



# References

- Deep Learning book
  - https://www.deeplearningbook.org/
- Stanford notes on deep learning
  - http://cs229.stanford.edu/notes/cs229-notes-deep_learning.pdf
- History of Deep Learning
  - https://beamandrew.github.io/deeplearning/2017/02/23/deep_learning_101_part1.html

# Acknowledgements


- Slides made using resources from:
  - Andrew Ng
  - Eric Eaton
  - David Sontag
  - Andrew Moore
- Thanks!