전통적인 접근법

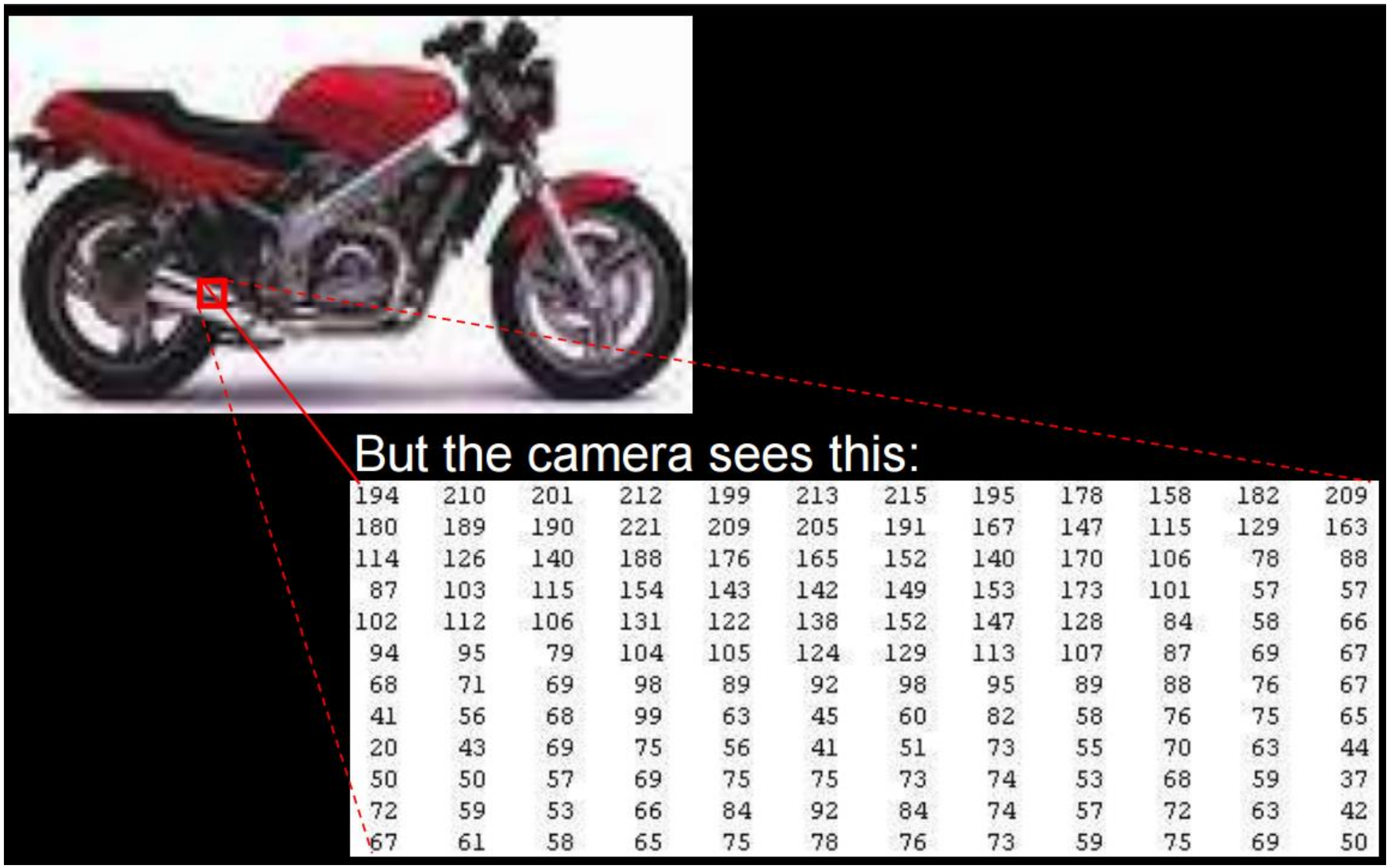
Conventional approach

- Image classification

“Motocycle”

Slides from “Andrew Ng”

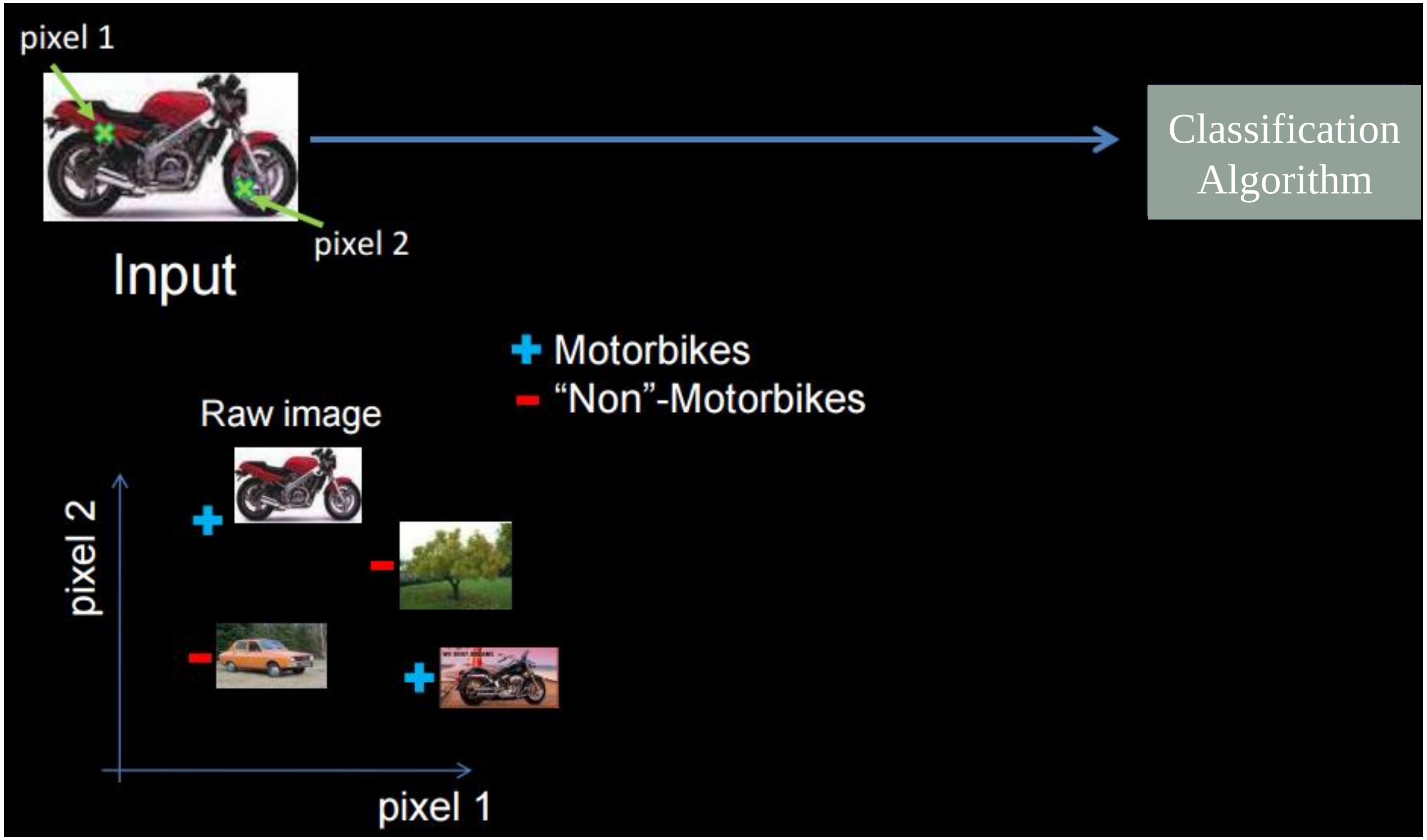
Why is this hard?



194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	75	65
20	43	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

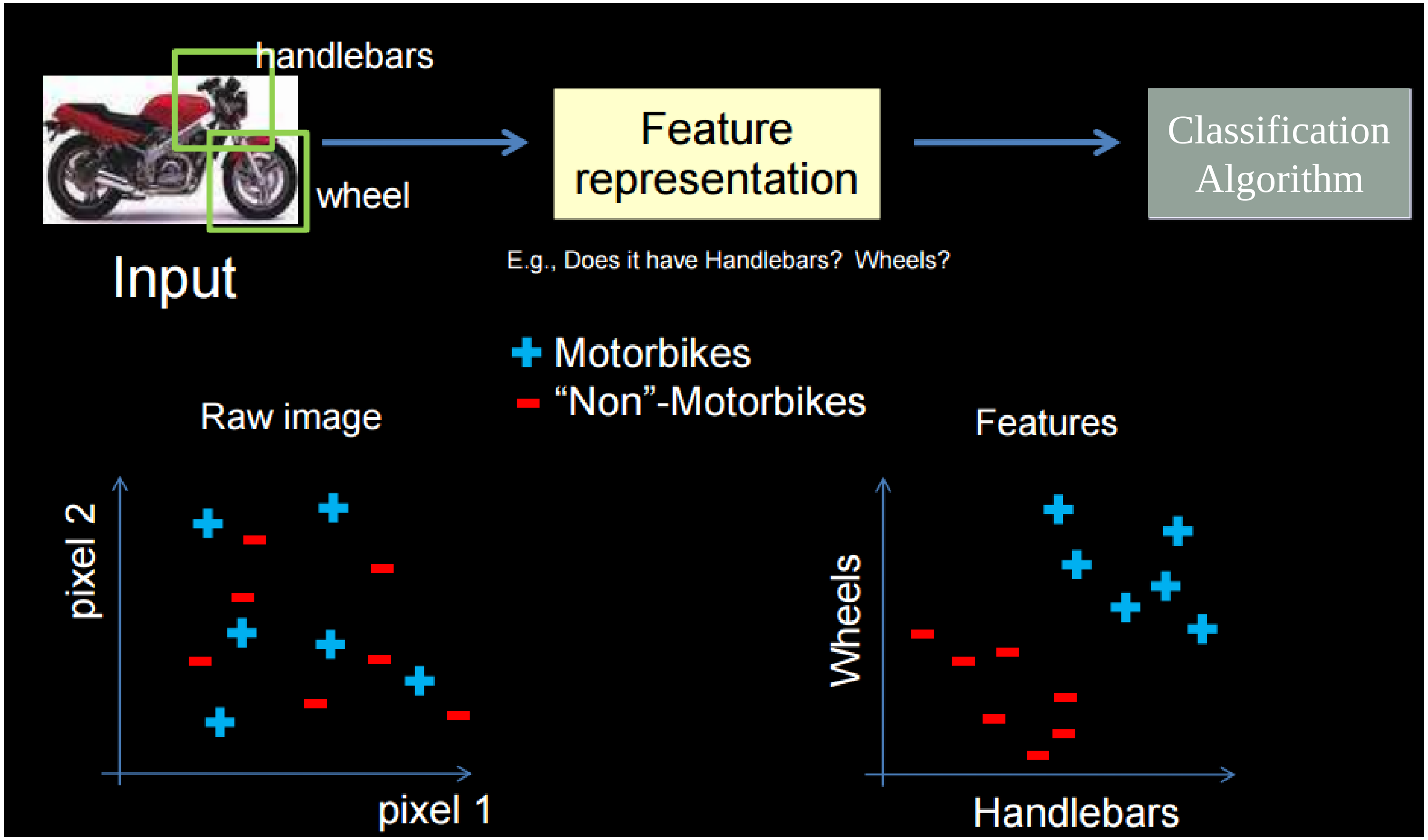
Slides from "Andrew Ng"

Feature representation



Slides from "Andrew Ng"

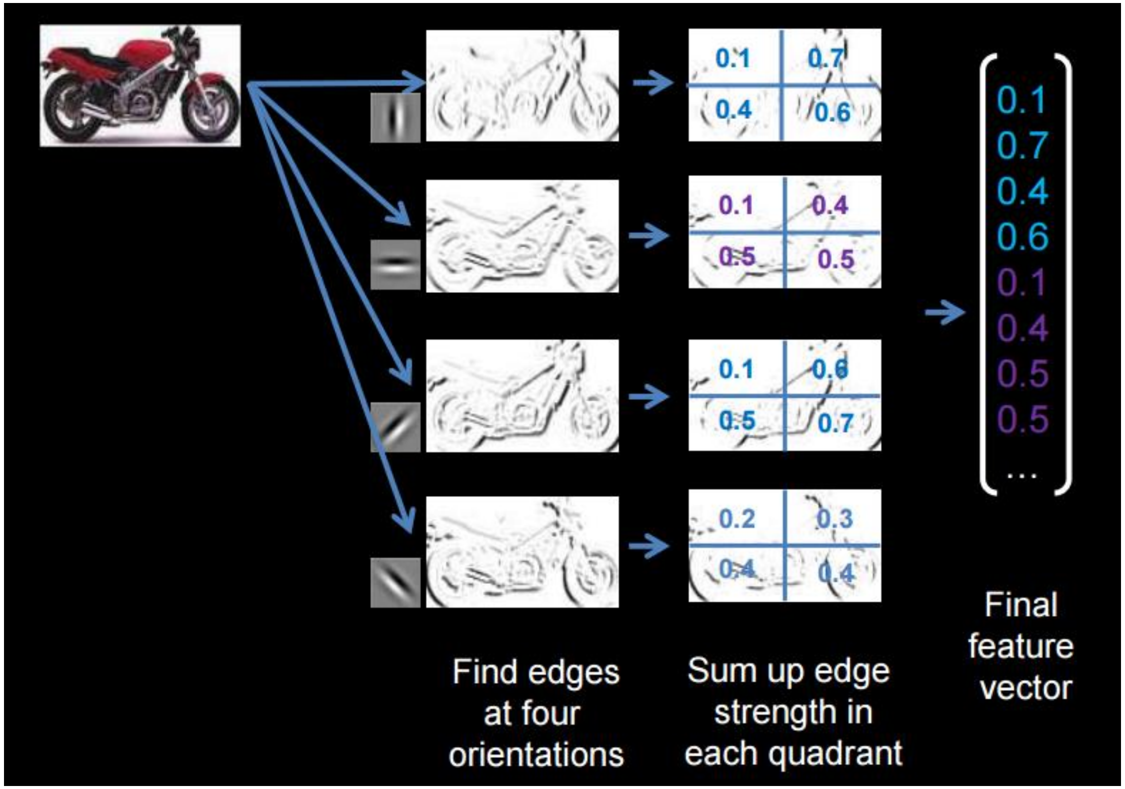
Feature representation



Slides from “Andrew Ng”

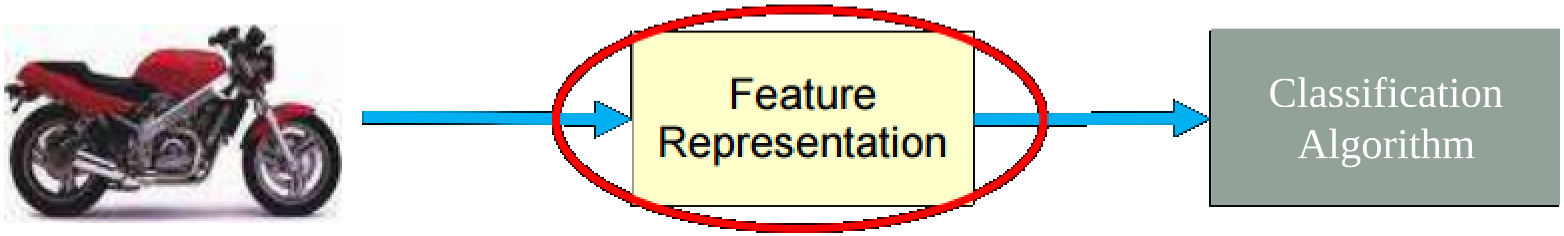
Example of Feature Representation

- But, … we don't have a handlebars detector. So, researchers try to hand-design features to capture various statistical properties of the image



Slides from "Andrew Ng"

Feature representation

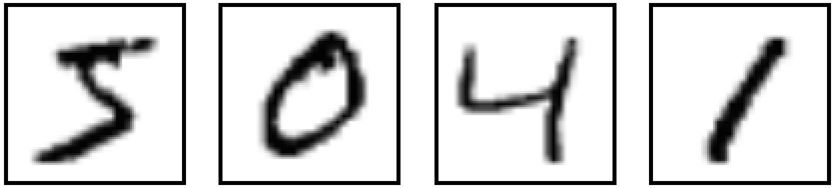


Slides from "Andrew Ng"

MNIST EXAMPLE

MNIST dataset

- Simple computer vision dataset
  - 28x28 pixel images of handwritten digits

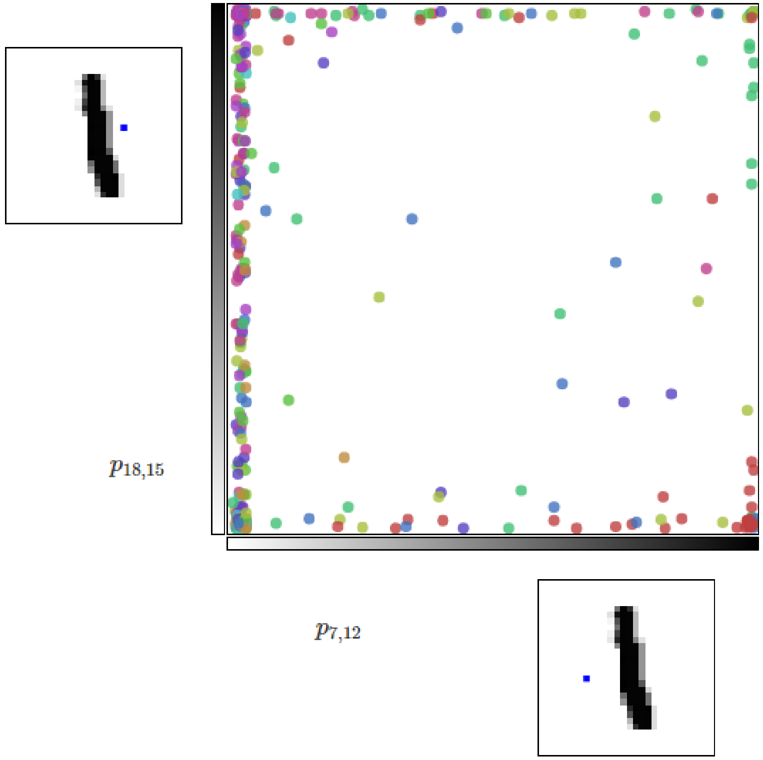


  - 28x28 array or 784 dimensional vector

$$\cong \begin{bmatrix}
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&.6&.8&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.7&1&0&0&0&0&0&0\\
0&0&0&0&0&0&.5&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.4&0&0&0&0&0\\
0&0&0&0&0&0&0&1&.7&0&0&0&0&0\\
0&0&0&0&0&0&0&1&1&0&0&0&0&0\\
0&0&0&0&0&0&0&.9&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&.3&1&.1&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0\\
0&0&0&0&0&0&0&0&0&0&0&0&0&0
\end{bmatrix}$$

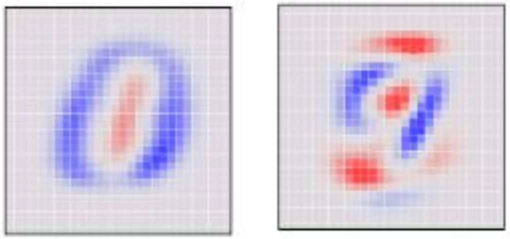
Dimension Reduction – 1

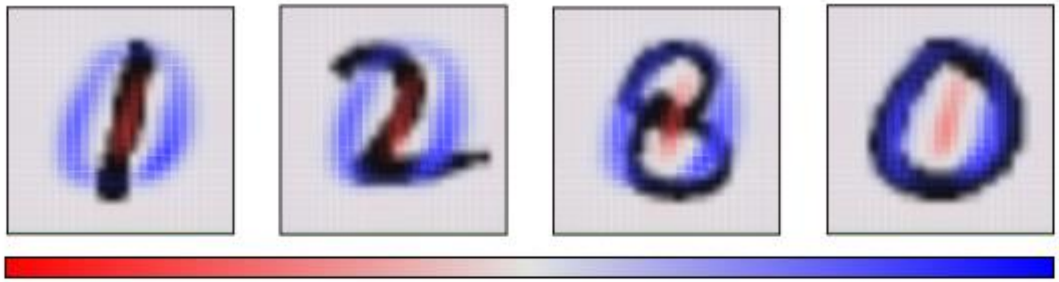
- Dots are colored based on which class of digit the data point belongs to.



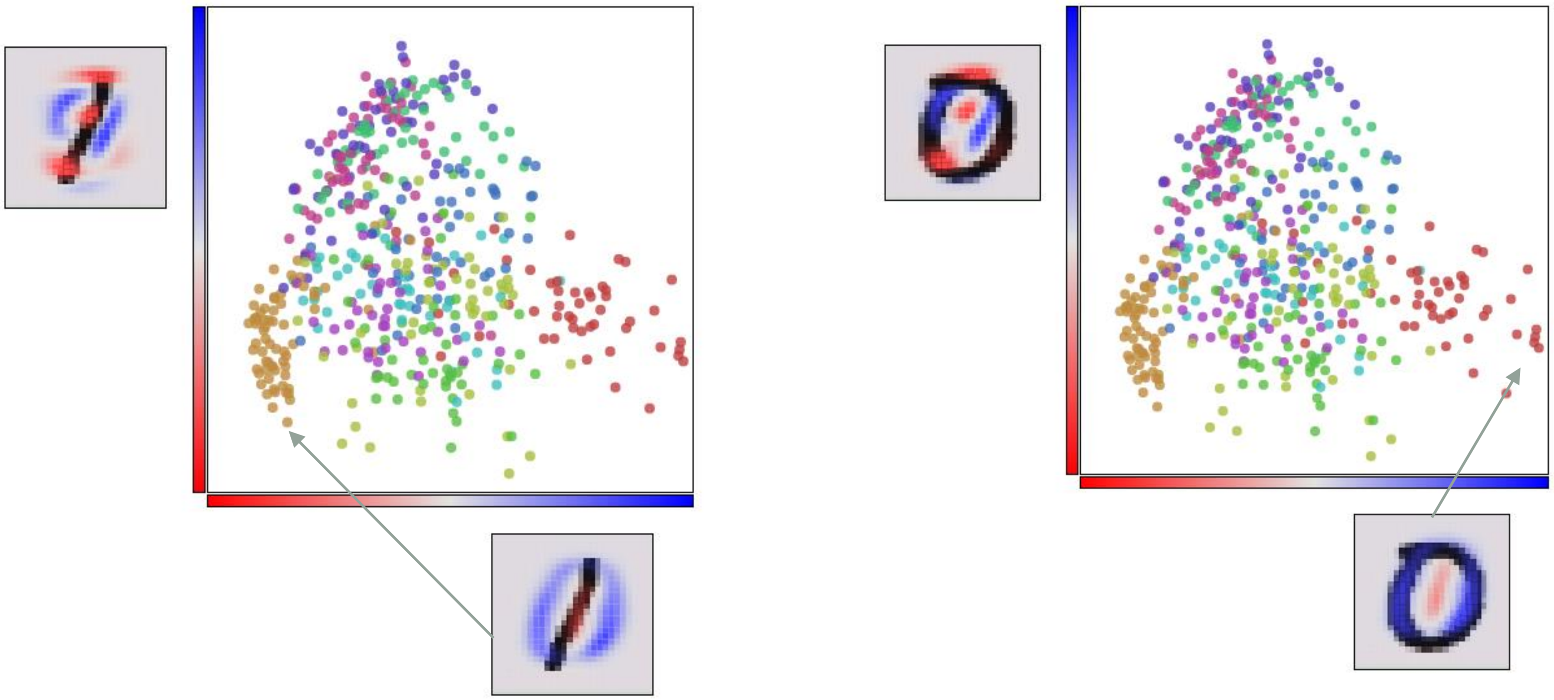
Dimension Reduction – 2 (PCA)

- Filter

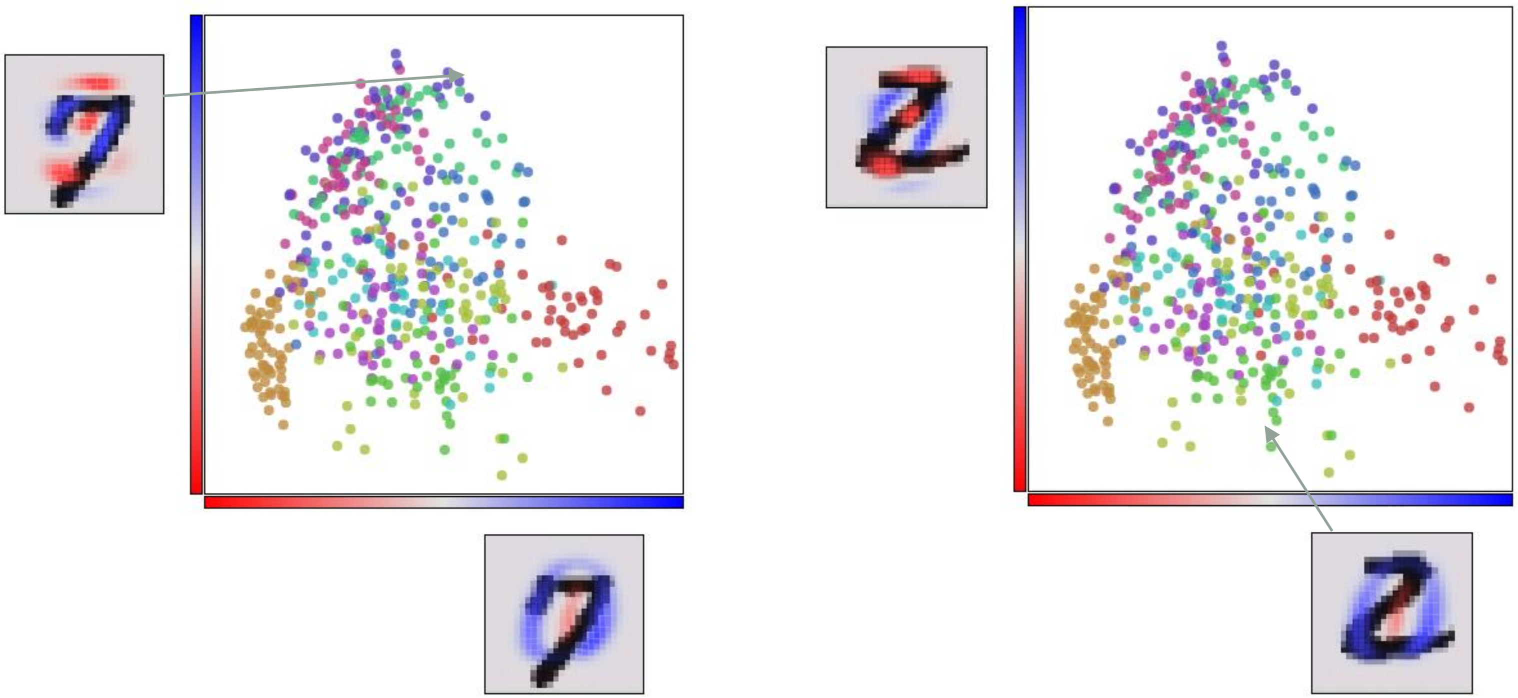




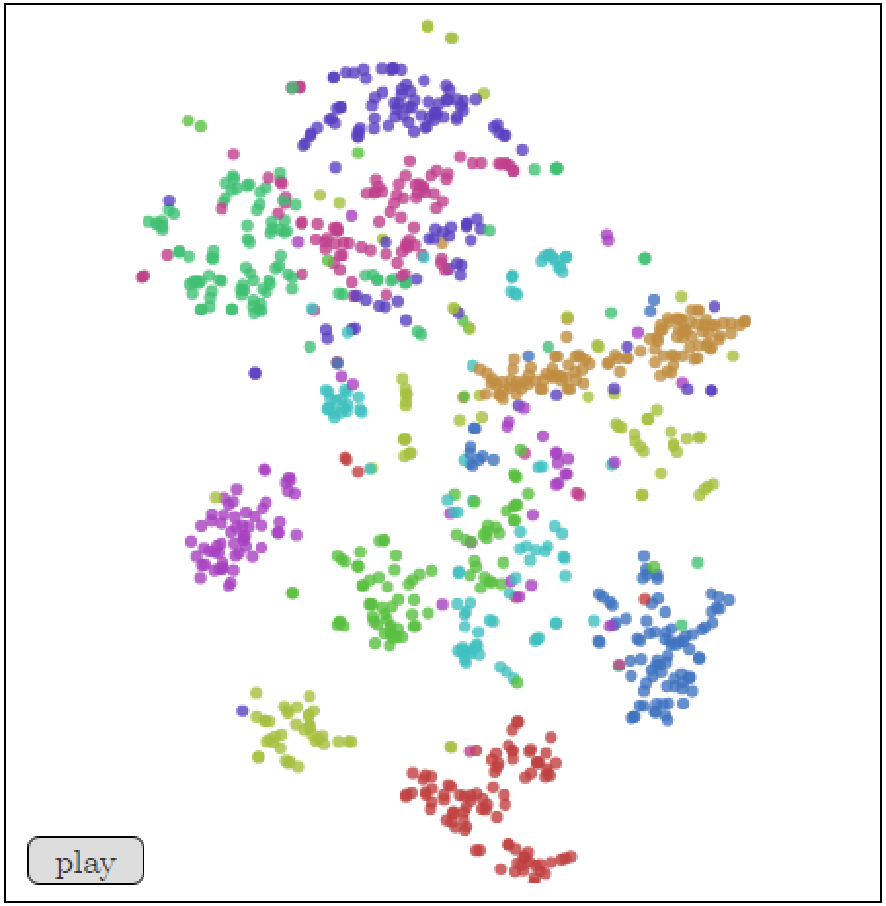
Dimension Reduction – 2 (PCA)



Dimension Reduction – 2 (PCA)



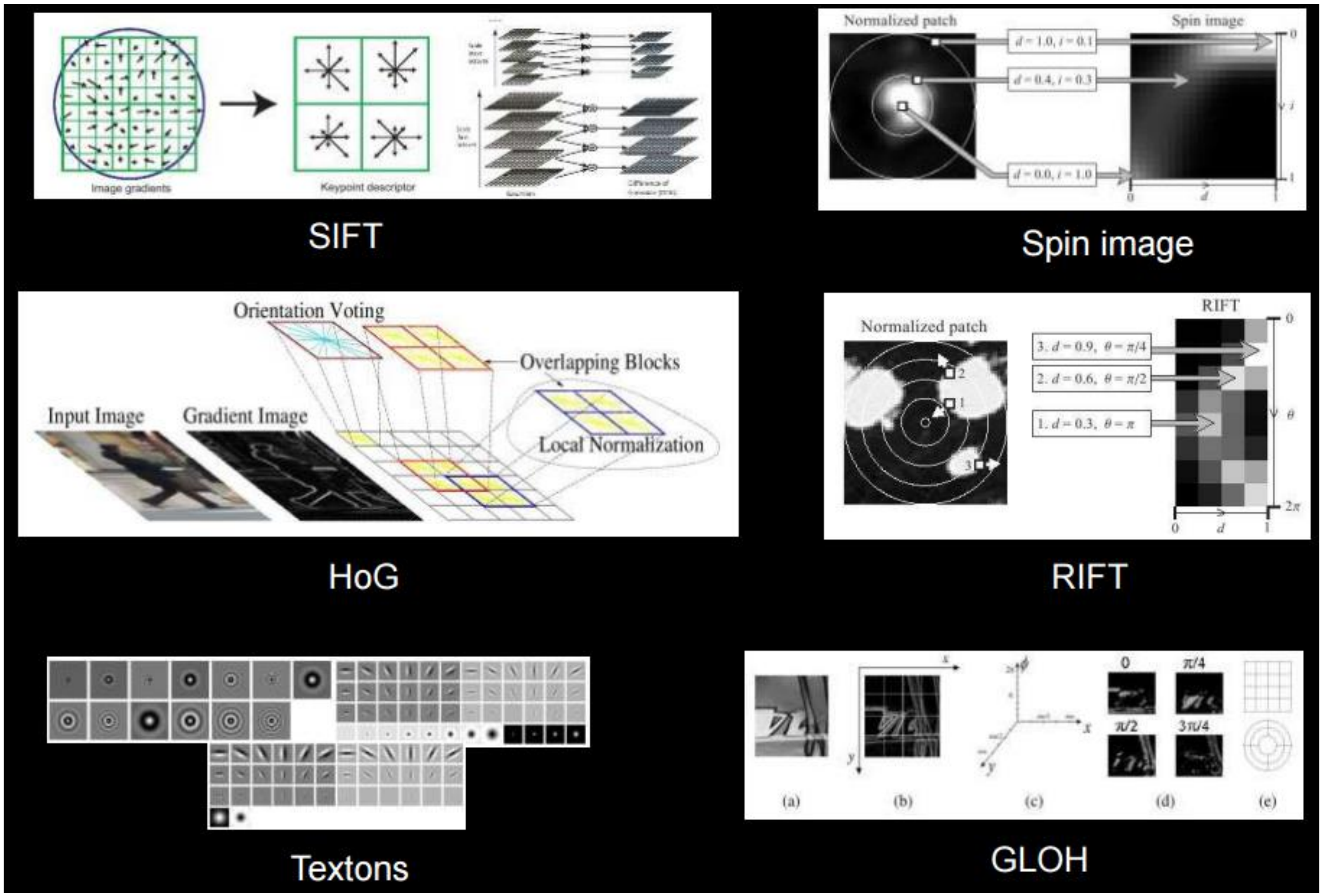
Visualization MNIST with t-SNE



Visualizing MNIST with t-SNE

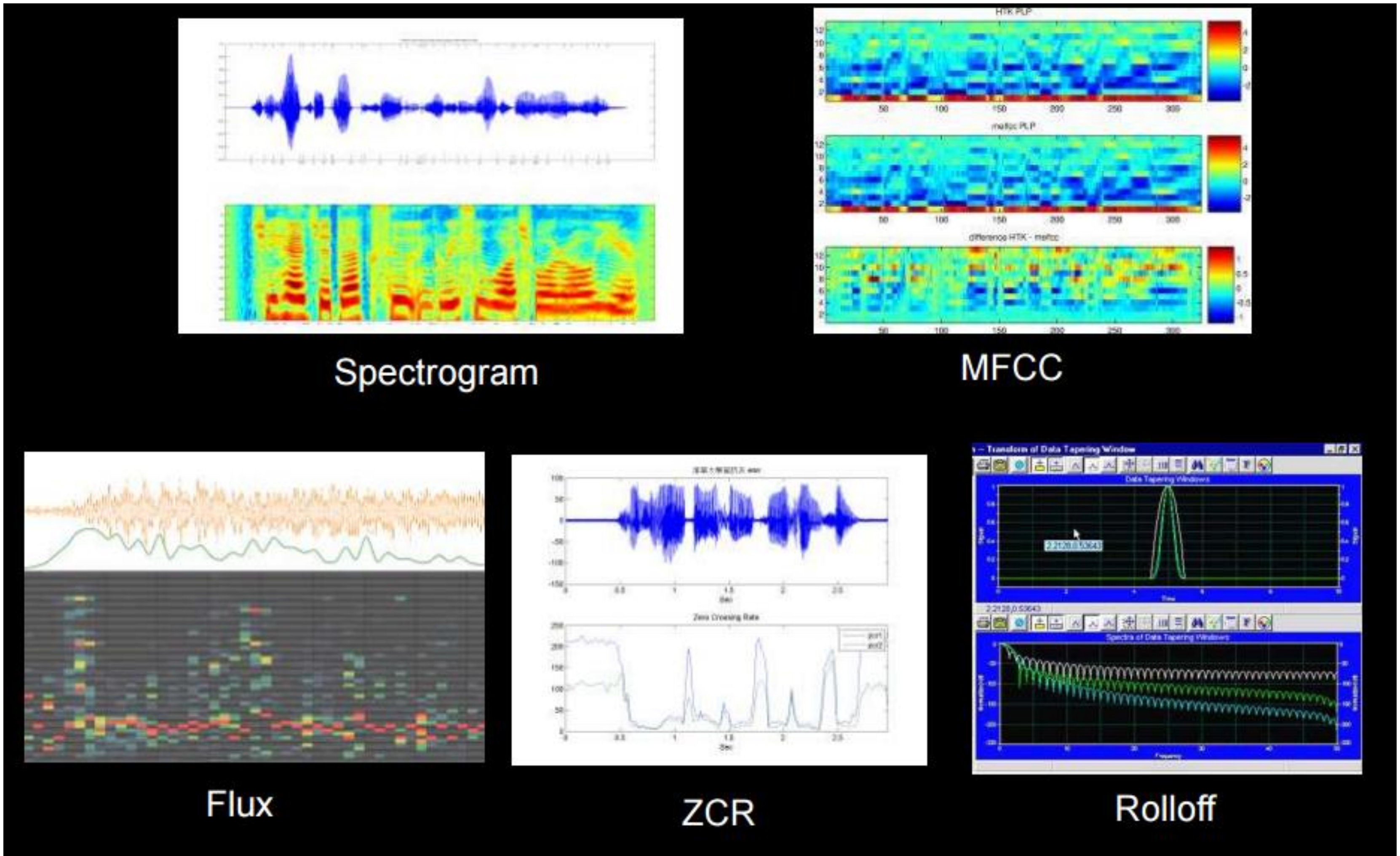
FEATURE EXAMPLES

Computer vision features

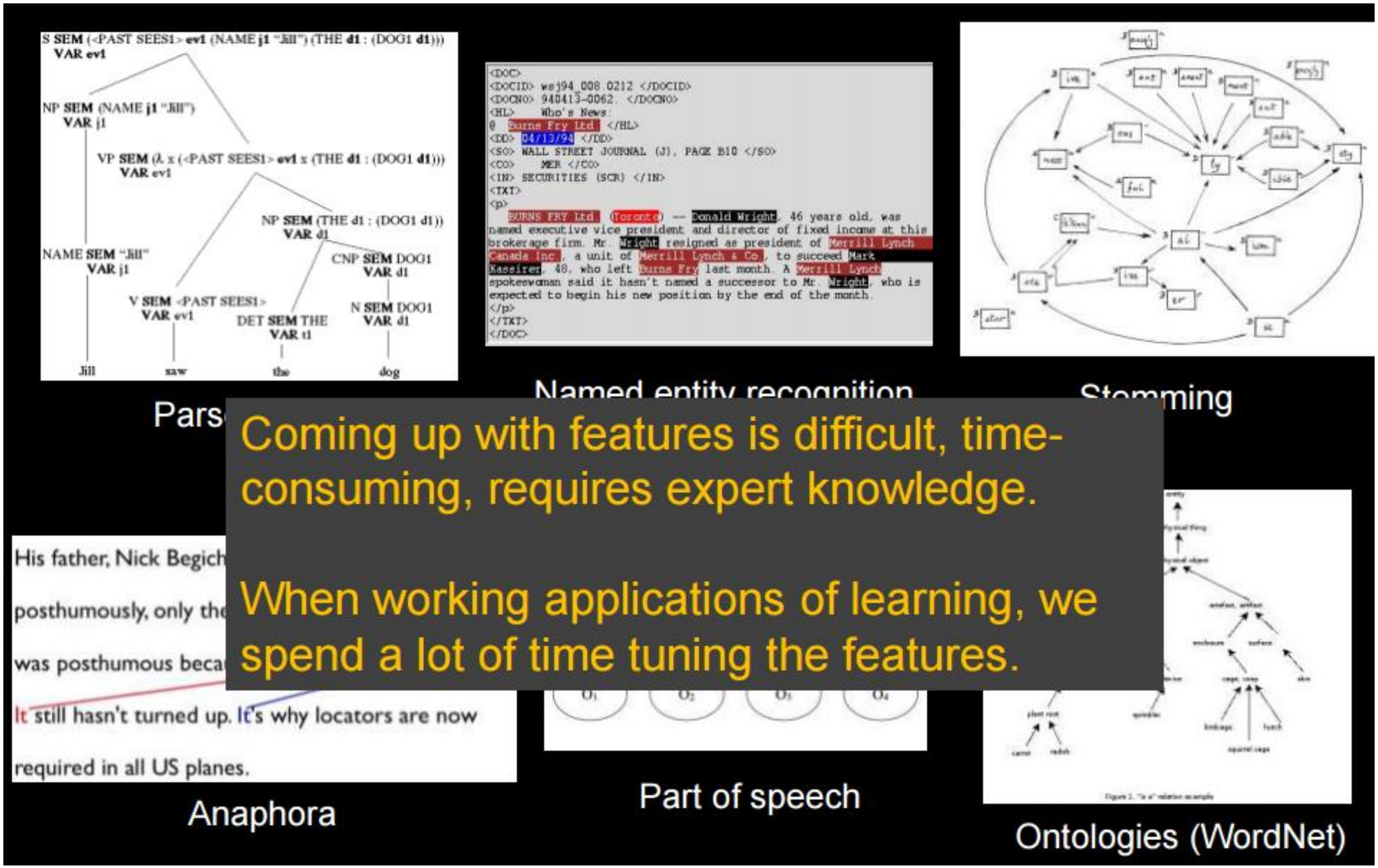


Slides from “Andrew Ng”

Audio features



Natural Language Processing Features



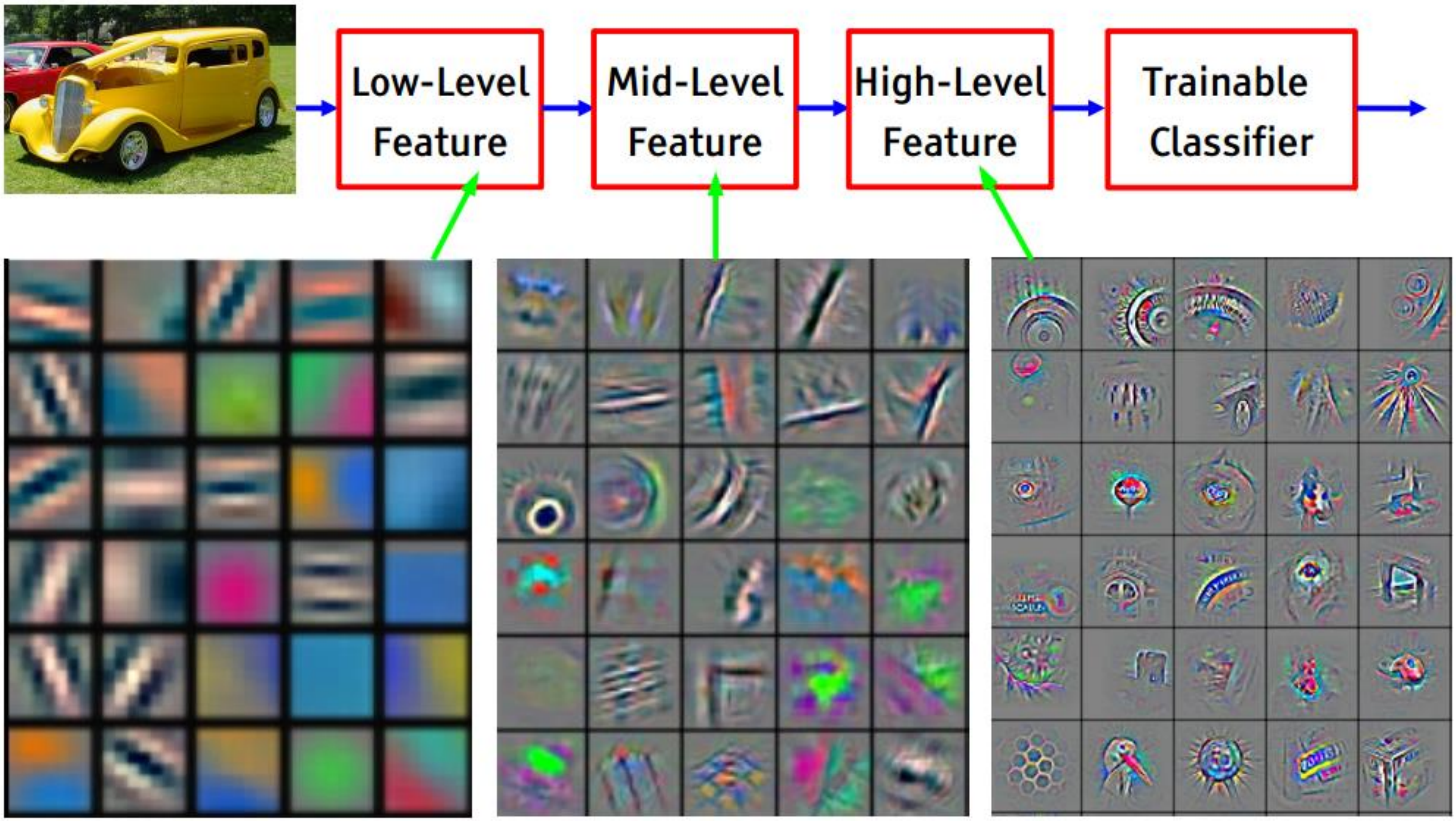
“SIMPLE” TRAINABLE CLASSIFIERS

Models of pattern recognition

- Traditional Pattern Recognition
  - Fixed/engineered features (or fixed kernel) + trainable classifier

Deep Learning

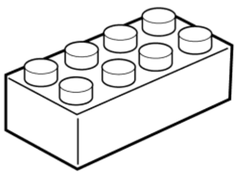
A family of methods that uses deep architectures to learn high-level feature representations

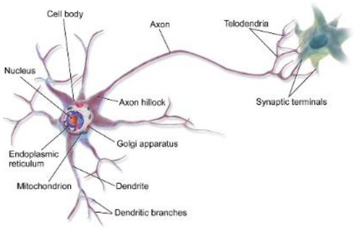


Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

LINEAR PERCEPTRON

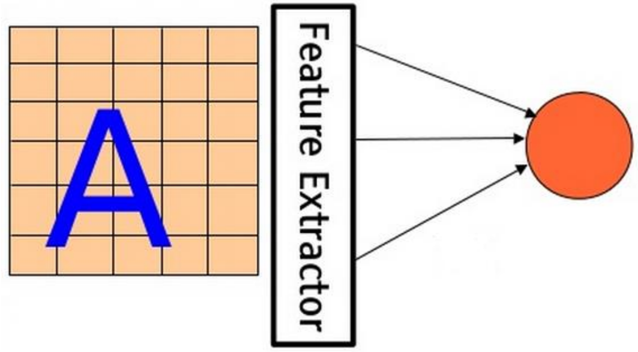
뉴런: 신경망의 기본 단위



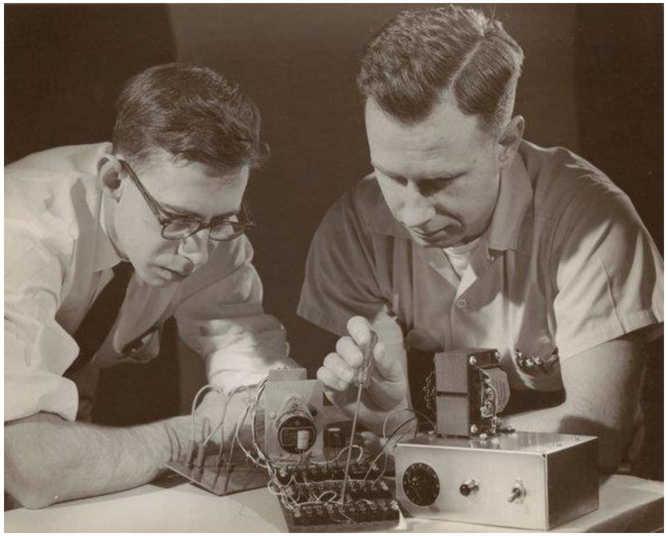

Cell body
Axon
Telodendria
Nucleus
Axon hillock
Synaptic terminals
Golgi apparatus
Endoplasmic reticulum
Mitochondrion
Dendrite
Dendritic branches

Basic model

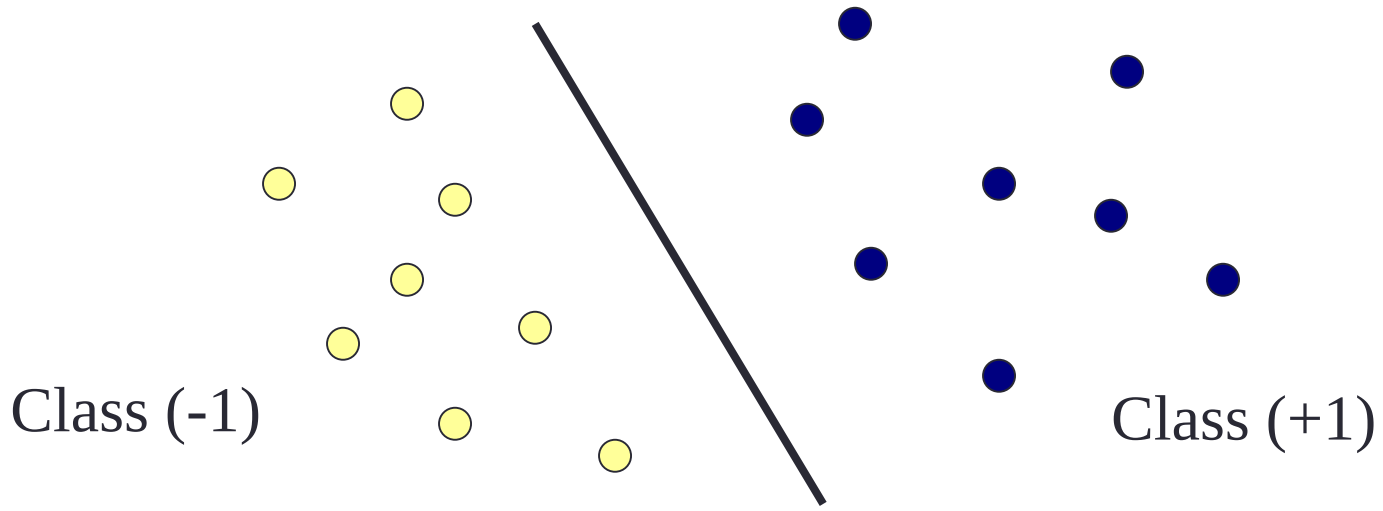
- The first learning machine: the Perceptron (built in 1960)
- The perceptron was a linear classifier on top of a simple feature extractor
- The vast majority of practical applications of machine learning used linear classifiers.



$$y = \text{sign}(W^T F(X) + b)$$



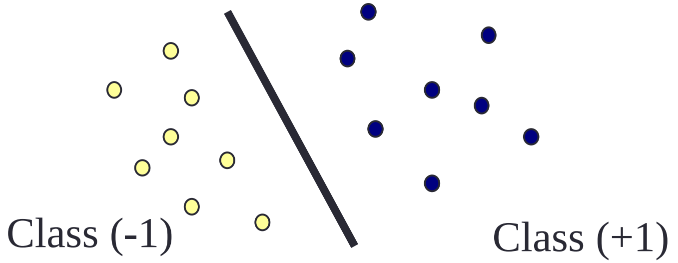
Simple Linear Perceptron



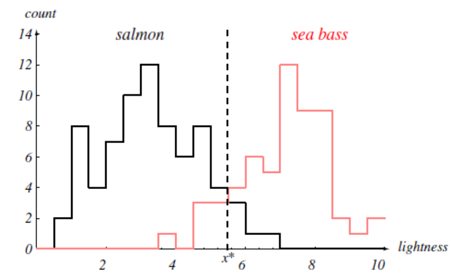
- The goal: Find the best line (or hyper-plane) to separate the training data. How to formalize it?
  - In two dimensions, the equation of the line is given by:
    - $w_1 x + w_2 y = b$
  - A better notation for *n* dimensions: treat each data point and the coefficients as vectors. Then the equation is given by:
    - $w^T x = b$

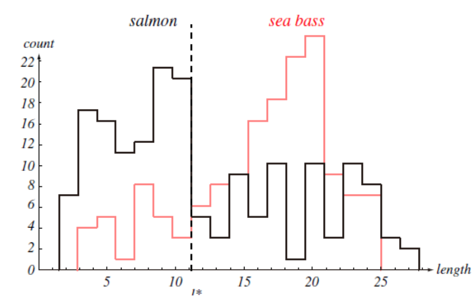
Simple Linear Perceptron

- The Simple Linear Perceptron is a classifier as shown in the picture
  - Points that fall on the right are classified as "+1"
  - Points that fall on the left are classified as "-1"
- Therefore: using the training set, find a hyperplane (line) so that
  - $w^T x > b$ for positive samples
  - $w^T x < b$ for negative samples



예시: 연어와 농어의 구별


count
14
12
10
8
6
4
2
0
salmon
sea bass
lightness
2
4
x*
6
8
10


salmon
sea bass
count
22
20
18
16
12
10
8
6
4
2
0
length
5
10
l*
15
20
25

예시: 연어와 농어의 구별

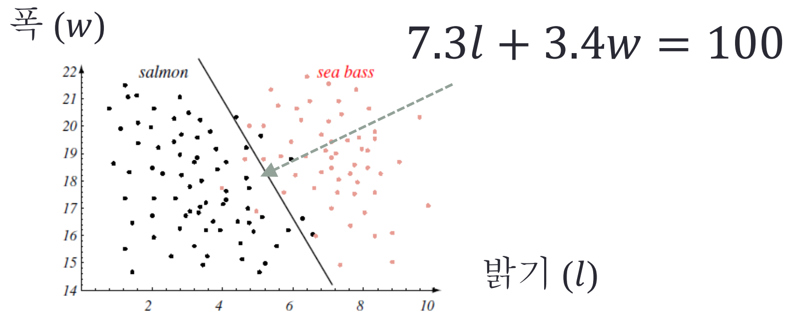
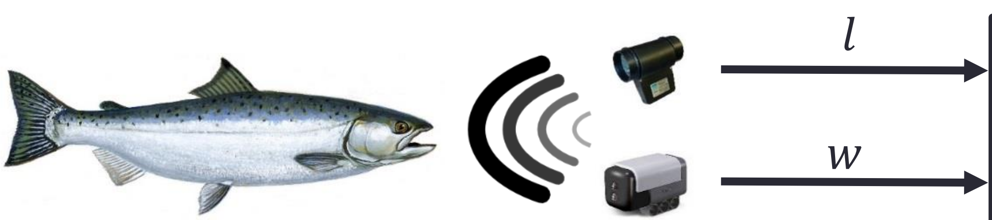
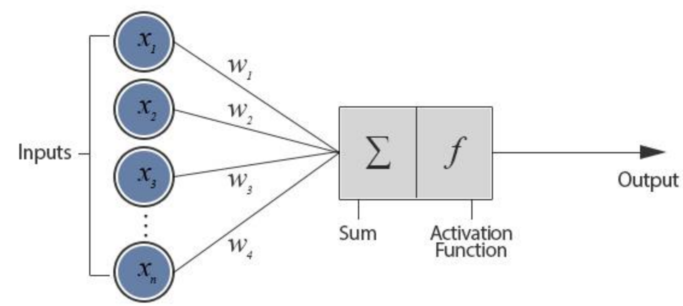


FIGURE 1.4. The two features of lightness and width for sea bass and salmon. The dark line could serve as a decision boundary of our classifier. Overall classification error on the data shown is lower than if we use only one feature as in Fig. 1.3, but there will still be some errors. From: Richard O. Duda, Peter E. Hart, and David G. Stork, *Pattern Classification*. Copyright © 2001 by John Wiley & Sons, Inc.

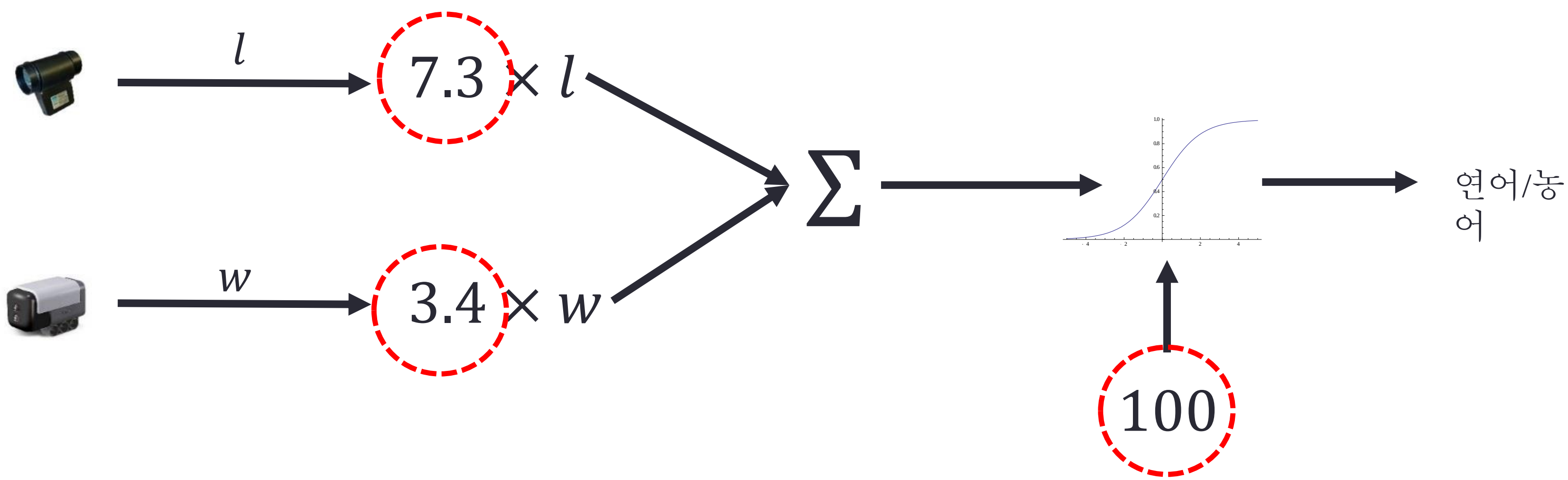


농어	$7.3l + 3.4w \geq 100$
연어	$7.3l + 3.4w < 100$

예시: 연어와 농어의 구별



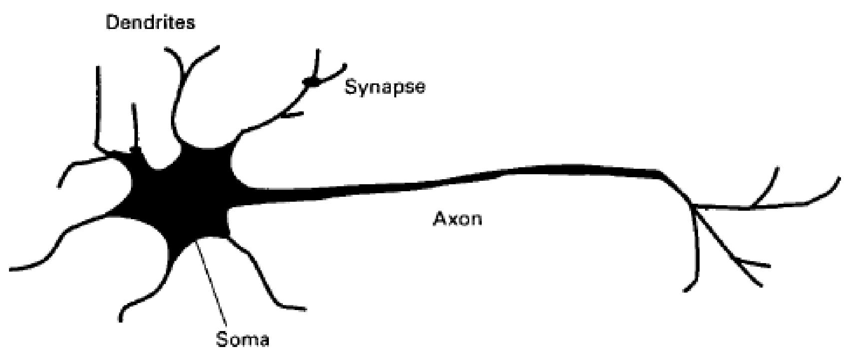
Inputs
x_1
x_2
x_3
x_n
w_1
w_2
w_3
w_4
Σ
f
Sum
Activation Function
Output

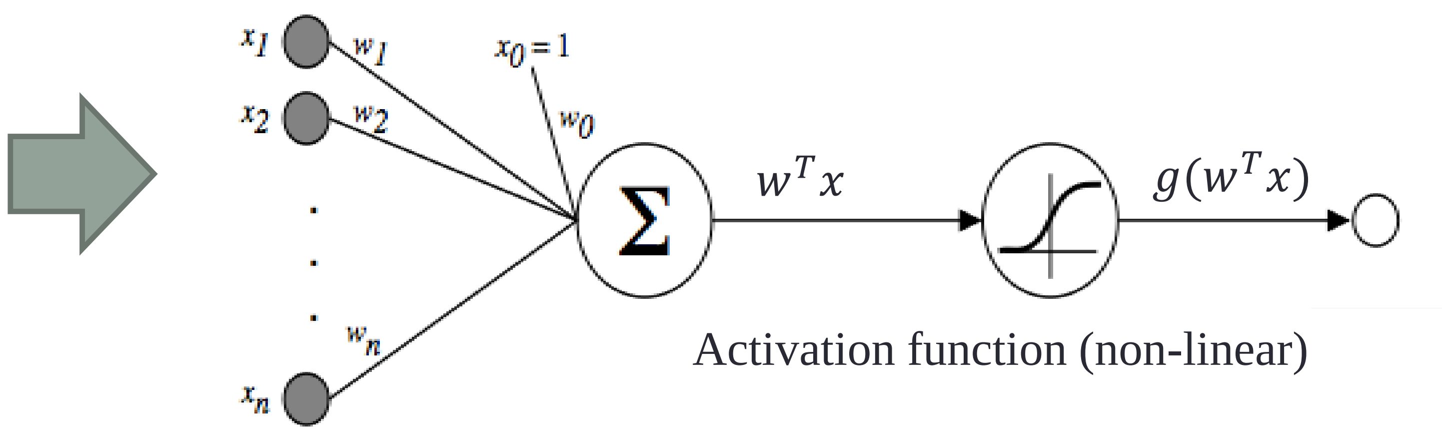


l
w
$7.3 \times l$
$3.4 \times w$
Σ
100
연어/농어

MULTI-LAYER PERCEPTRON – BACK-PROPAGATION ALGORITHM

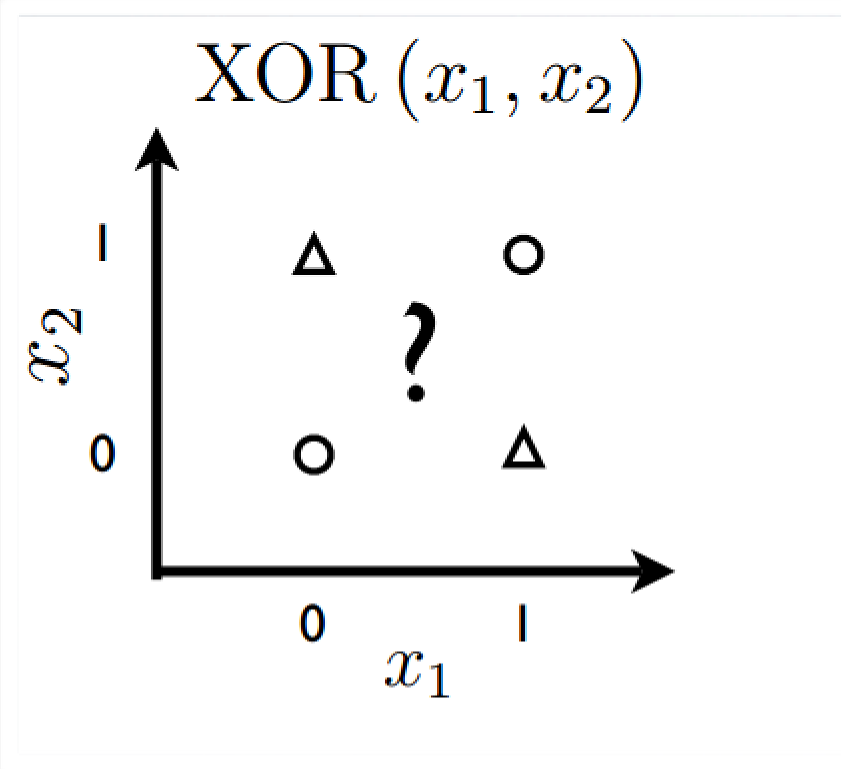
Artificial Neuron





Artificial Neuron

- Can't solve non-linearly-separable problems

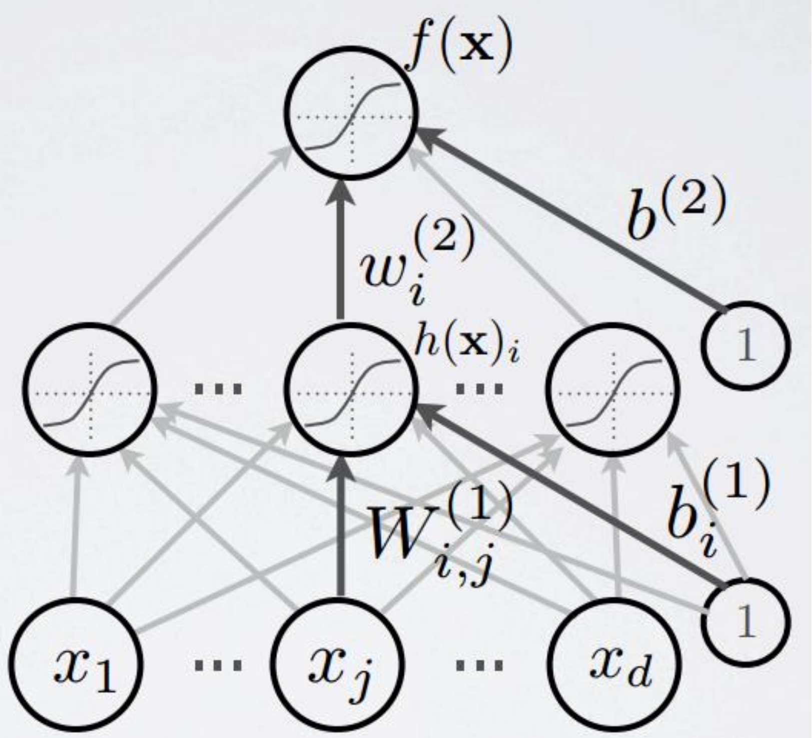


Multi-layer Neural Network

- Hidden layer pre-activation:
  - $\mathbf{a}(\mathbf{x}) = \mathbf{b}^1 + \mathbf{W}^1\mathbf{x}$

- Hidden layer activation
  - $\mathbf{h}(\mathbf{x}) = \mathbf{g}(\mathbf{a}(\mathbf{x}))$

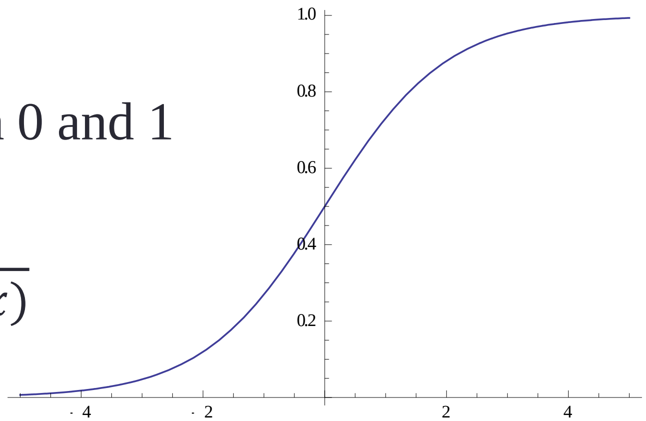
- Output layer activation
  - $\mathbf{f}(\mathbf{x}) = \mathbf{o}(\mathbf{b}^2 + (\mathbf{w}^2)^\mathbf{T}\mathbf{h}^1\mathbf{x})$



Activation function $g(\cdot)$

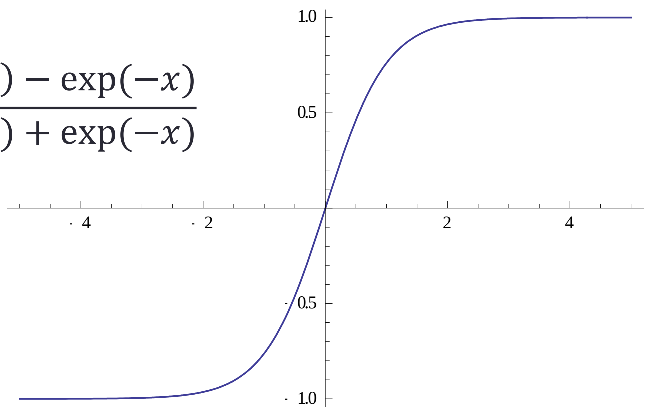
- Sigmoid activation function
  - Squashes the neuron's pre-activation between 0 and 1
  - Always positive
  - Bounded
  - Strictly increasing

$$g(x) = \frac{1}{1 + \exp(-x)}$$



- Hyperbolic tangent ("tanh") activation function
  - Squashes the neuron's pre-activation between -1 and 1
  - Can be positive or negative
  - Bounded
  - Strictly increasing

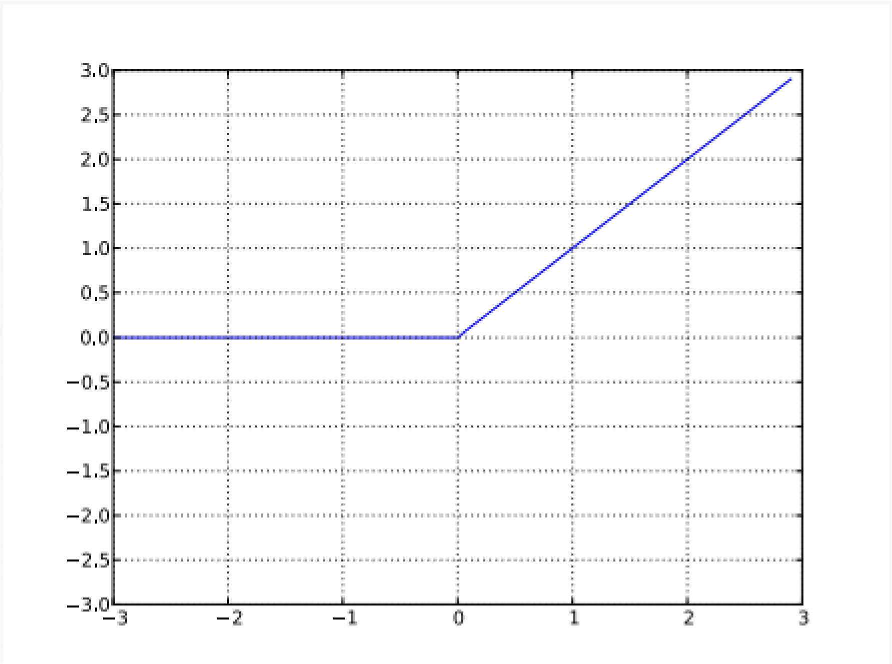
$$g(x) = \tanh(x) = \frac{\exp(x) - \exp(-x)}{\exp(x) + \exp(-x)}$$



Activation function $g(\cdot)$

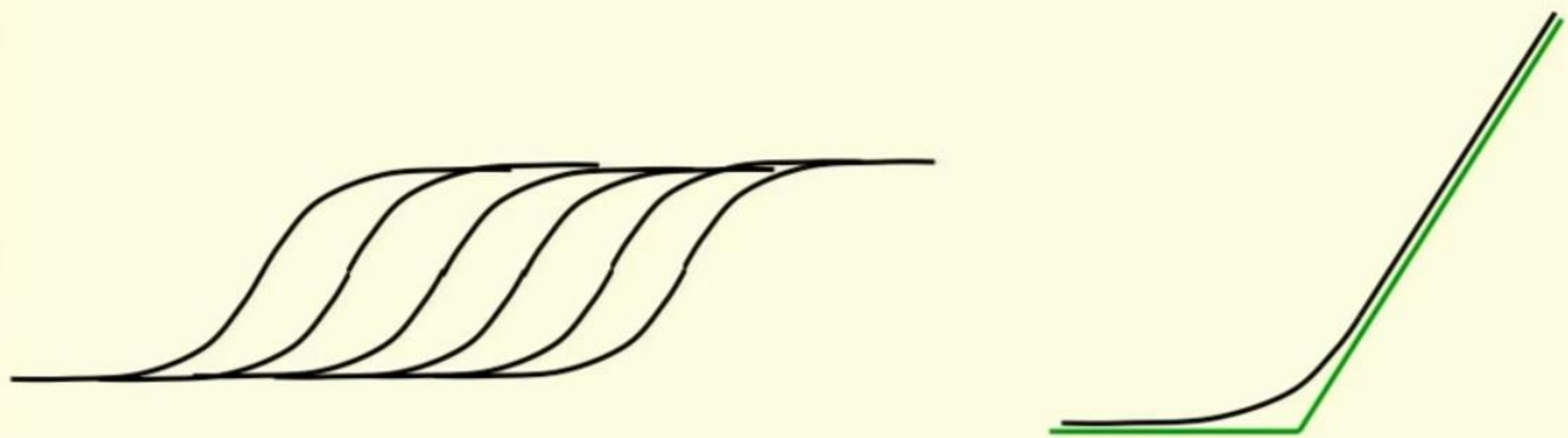
- Rectified linear activation function
  - Bounded below by 0
  - Not upper bounded
  - Strictly increasing
  - Tends to give neurons with sparse activities

$$g(a) = rectlin(a) = \max(0, a)$$



Rectified linear activation function

- Rectified linear units are much faster to compute than the sum of many logistic units.



$$\sum_{n=1}^{n=\infty} \text{logistic}(x + 0.5 - n) \quad \approx \quad \log(1 + e^{x})$$

Output = max(0, input)

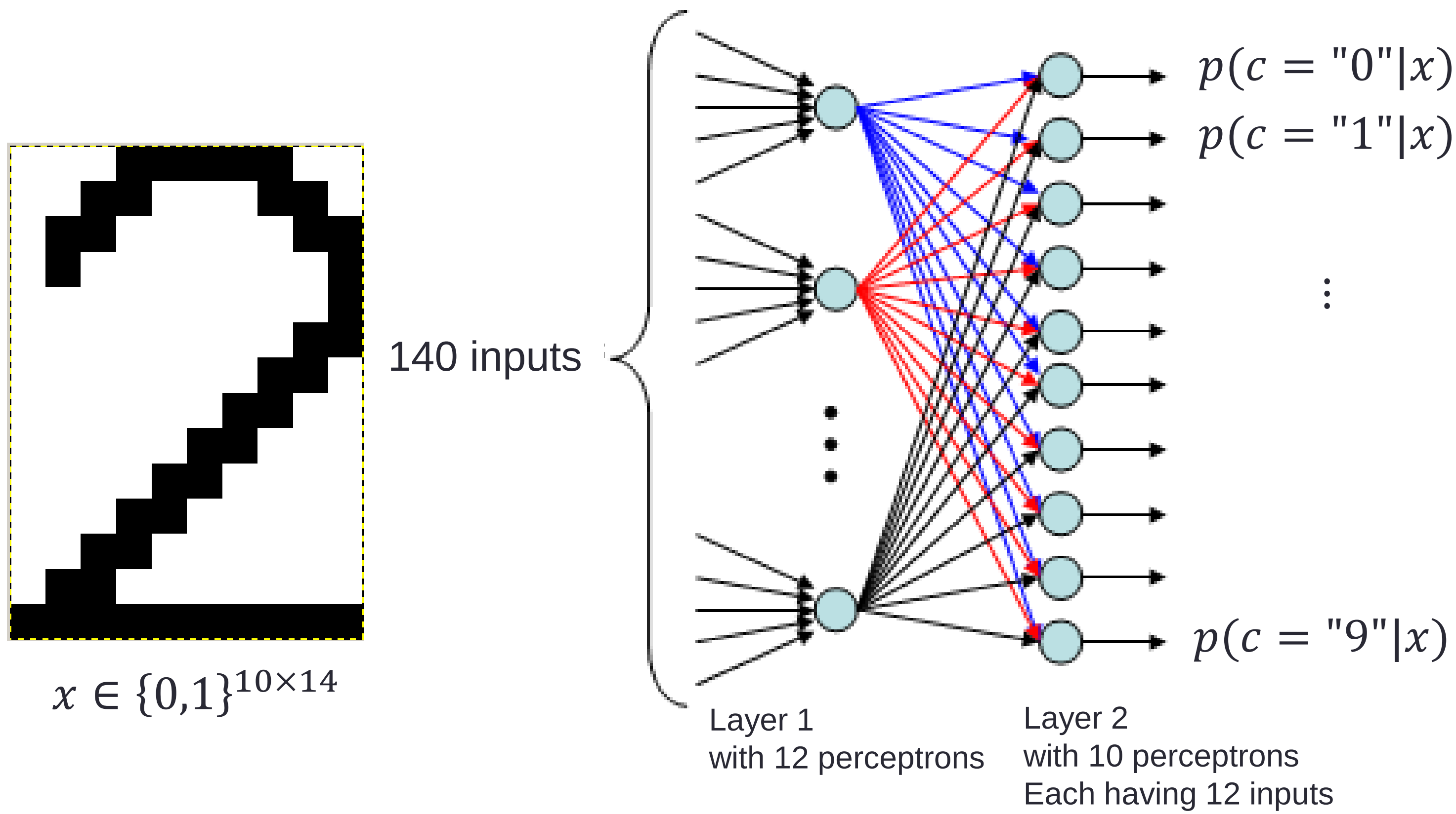
Softmax activation function at the output

- For multi-class classification
  - We need multiple outputs (1 output per class)
  - We would like to estimate the conditional probability $p(y = c|x)$
- We use the softmax activation function at the output

$$O(\mathbf{a}) = softmax(\mathbf{a}) = \begin{bmatrix} \frac{\exp(a_1)}{\sum_c \exp(a_c)} \\ \frac{\exp(a_2)}{\sum_c \exp(a_c)} \\ \vdots \\ \vdots \\ \frac{\exp(a_c)}{\sum_c \exp(a_c)} \end{bmatrix}$$

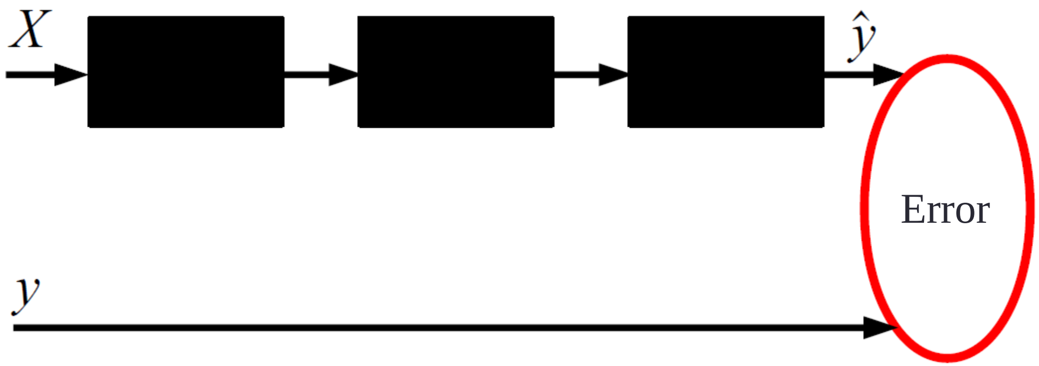
  - strictly positive
  - sums to one

Example (character recognition example)



TRAINING OF MULTI-LAYER PERCEPTRON

Training: Loss function

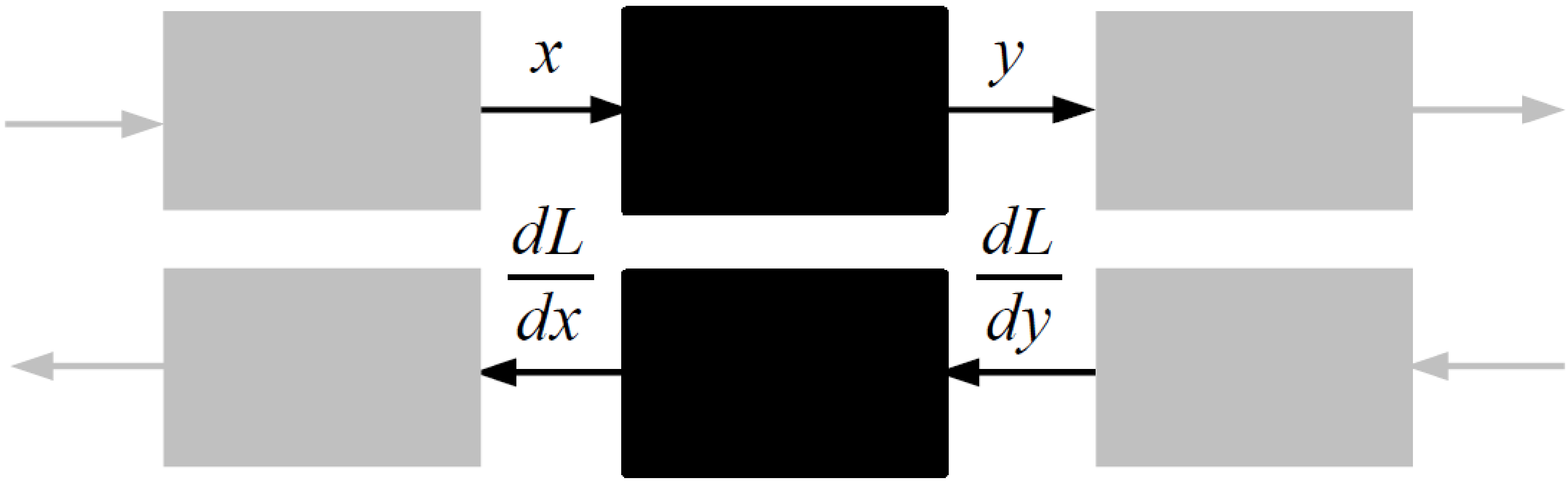


- Square Euclidean distance (regression)
  - $y, \hat{y} \in \Re^N$
  - $L = \frac{1}{2}\sum(y_i - \hat{y}_i)^2$

- Cross entropy (classification)
  - $y, \hat{y} \in [0,1]^N, \sum_{i=1} y_i = 1, \sum_{i=1} \hat{y}_i = 1$
  - $L = -\sum y_i \log \hat{y}_i$

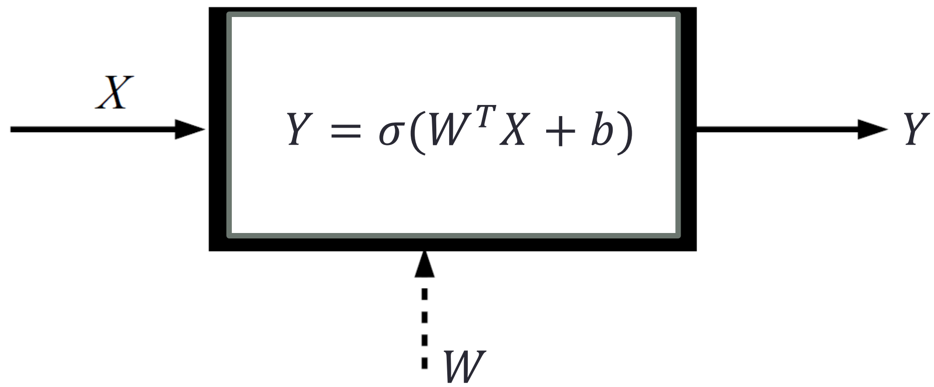
Forward/Backward propagation

- Chain rule



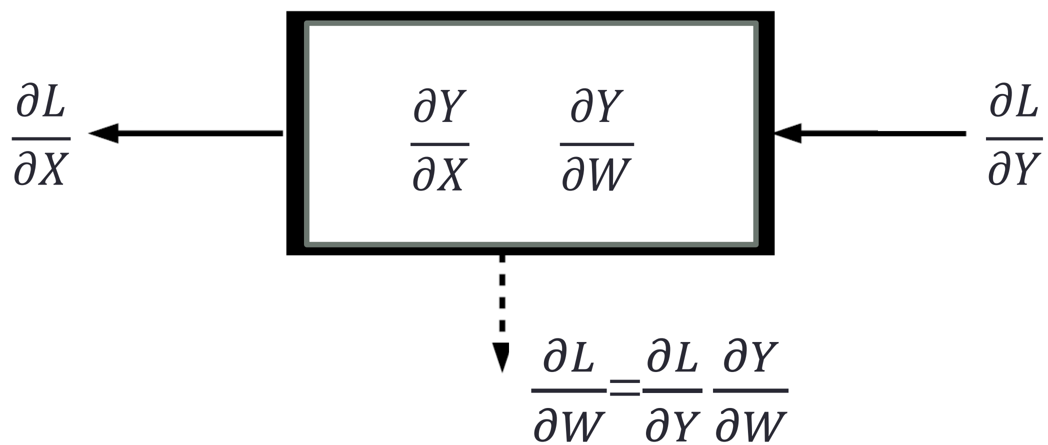
$$W^{new} = W^{old} - \eta \frac{dL}{dW}$$

Forward/Backward propagation

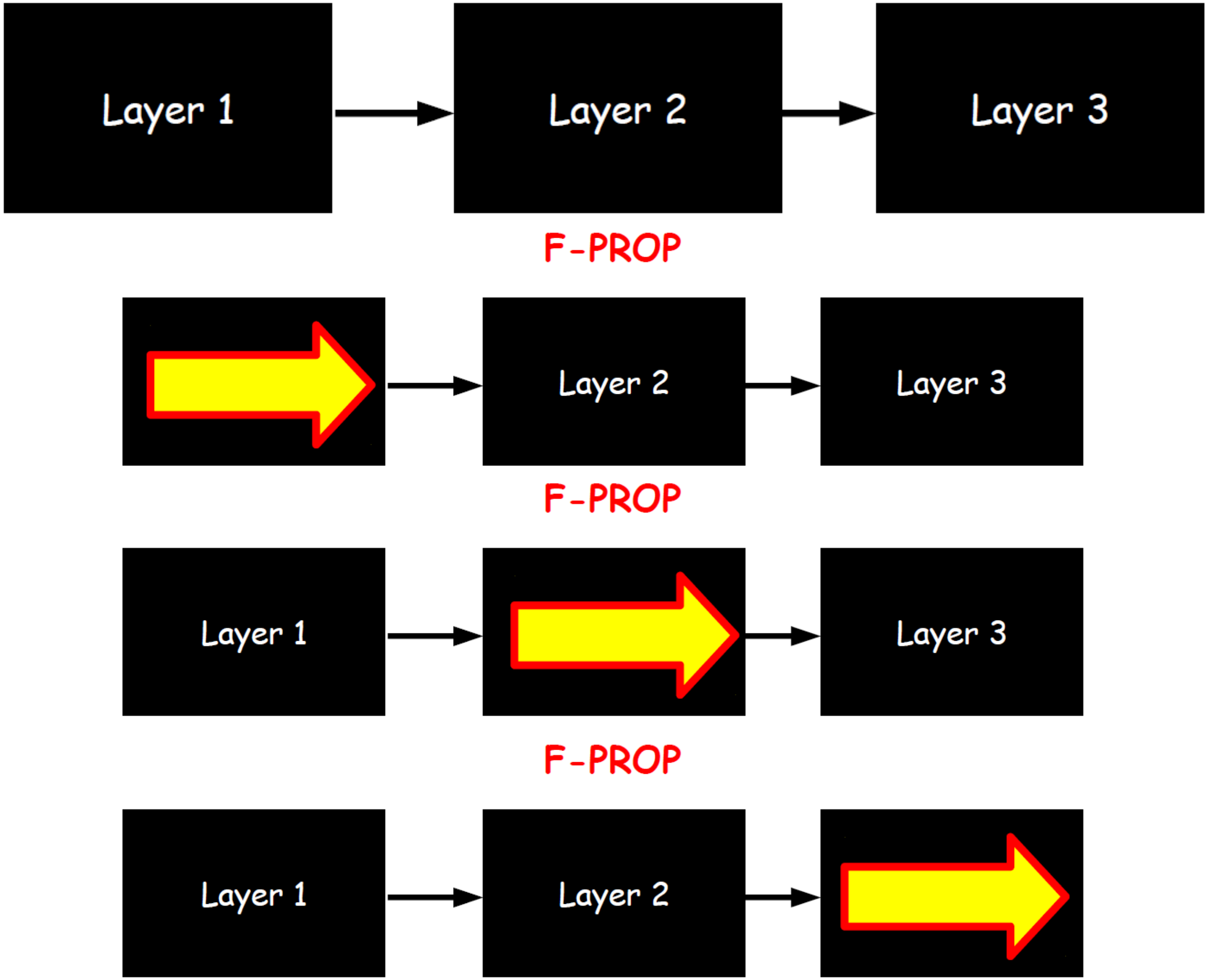


Forward propagation

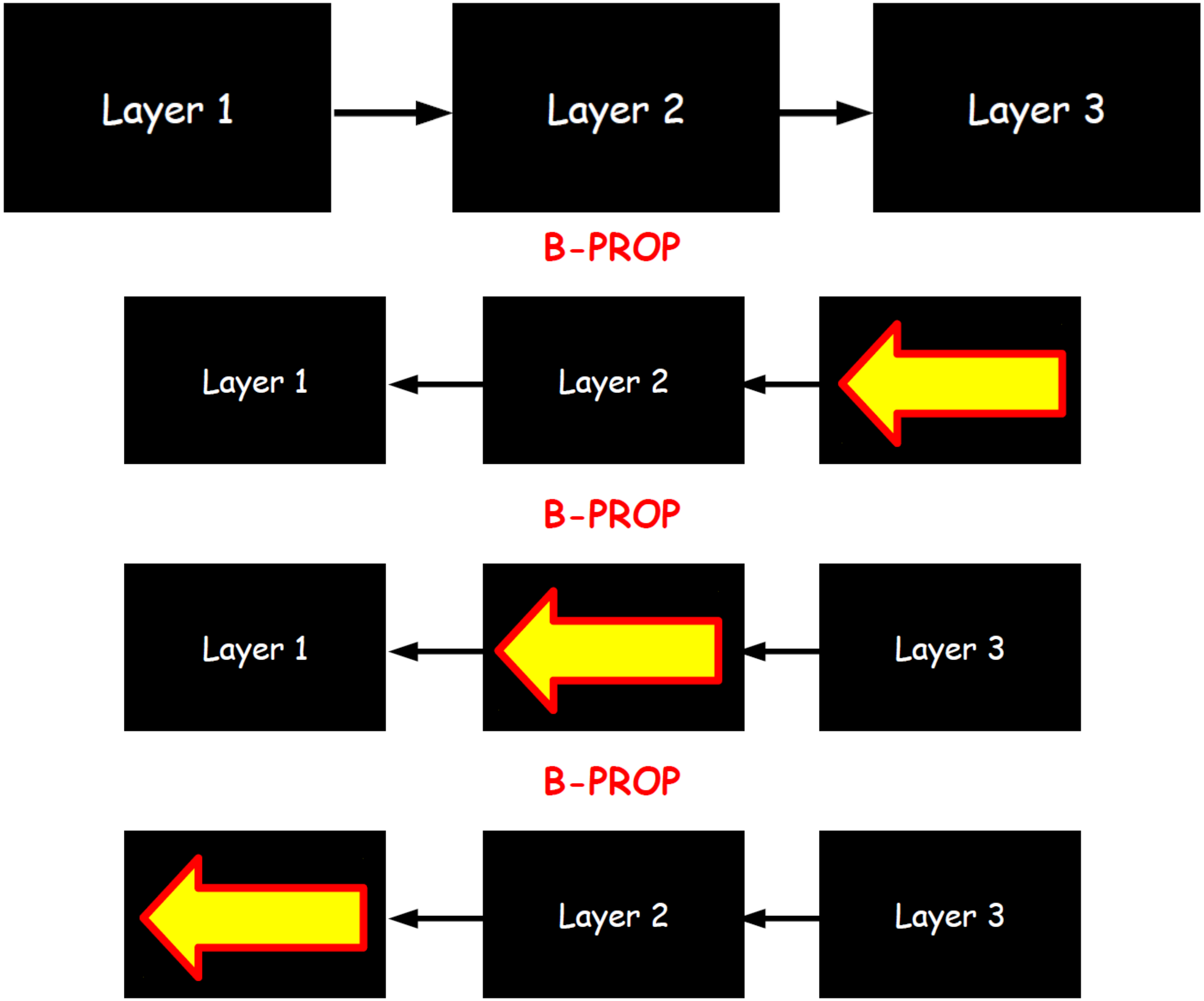
$$\frac{dL}{dX} = \frac{dL}{dY} \cdot \frac{\partial Y}{\partial X}$$



Backward propagation



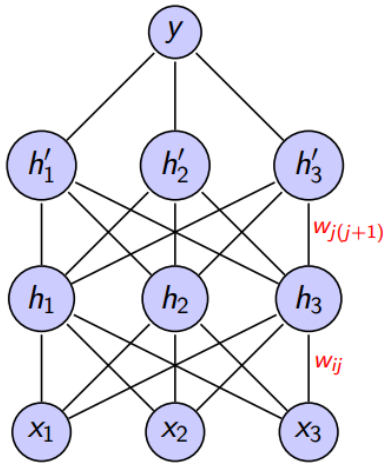
Layer 1
Layer 2
Layer 3
F-PROP
Layer 2
Layer 3
F-PROP
Layer 1
Layer 3
F-PROP
Layer 1
Layer 2



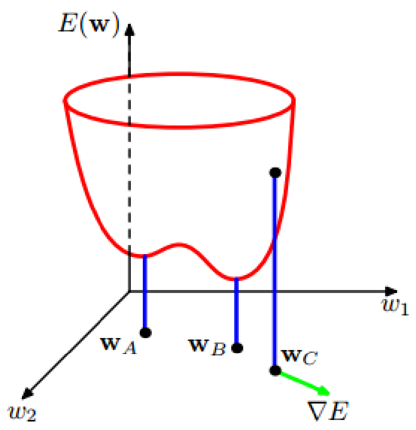
Compute gradient w.r.t. parameters and update parameters by using the gradients.

Why are Deep Architectures hard to train?

- Vanishing gradient problem in back-propagation.



- Local Optimum (saddle points?) Issue in Neural Nets
  - For Deep Architectures, back-propagation is apparently getting a local optimum (saddle points?) that does not generalize well



Empirical Results: Poor performance of Backpropagation on Deep Neural Nets [Erhan et al., 2009]

- MNIST digit classification task; 400 trials (random seed)
- Each layer: initialize weights with random numbers
- Although $L + 1$ layers is more expressive, worse error than L layers.