CS6220: DATA MINING TECHNIQUES

Image Data: Classification via Neural Networks

Instructor: Yizhou Sun

yzsun@ccs.neu.edu

November 19, 2015

Methods to Learn

	Matrix Data	Text Data	Set Data	Sequence Data	Time Series	Graph & Network	Images
Classification	Decision Tree; Naïve Bayes; Logistic Regression SVM; kNN			HMM		Label Propagation*	**Neural Network**
Clustering	K-means; hierarchical clustering; DBSCAN; Mixture Models; kernel k-means*	PLSA				SCAN*; Spectral Clustering*	
Frequent Pattern Mining			Apriori; FP-growth	GSP; PrefixSpan			
Prediction	Linear Regression				Autoregression		
Similarity Search					DTW	P-PageRank	
Ranking						PageRank	

Mining Image Data

- Image Data
- Neural Networks as a Classifier
- Summary

Images

- Images can be found everywhere
  - Social Networks, e.g. Instagram, Facebook, etc.
  - World Wide Web
  - All kinds of cameras

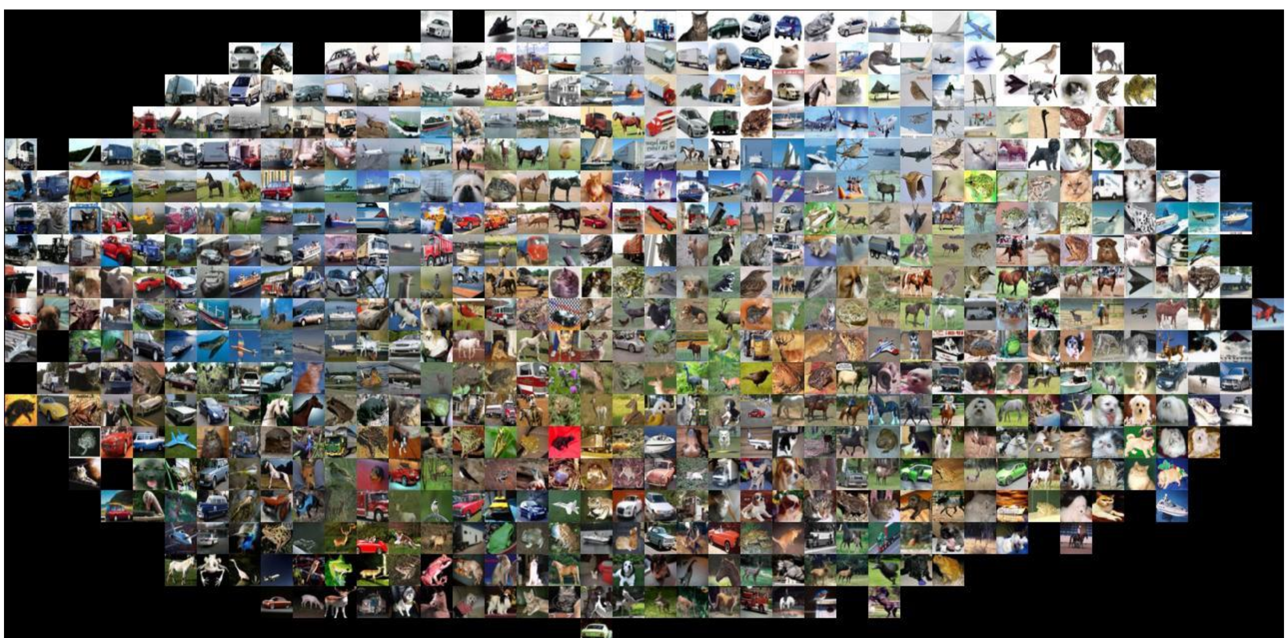
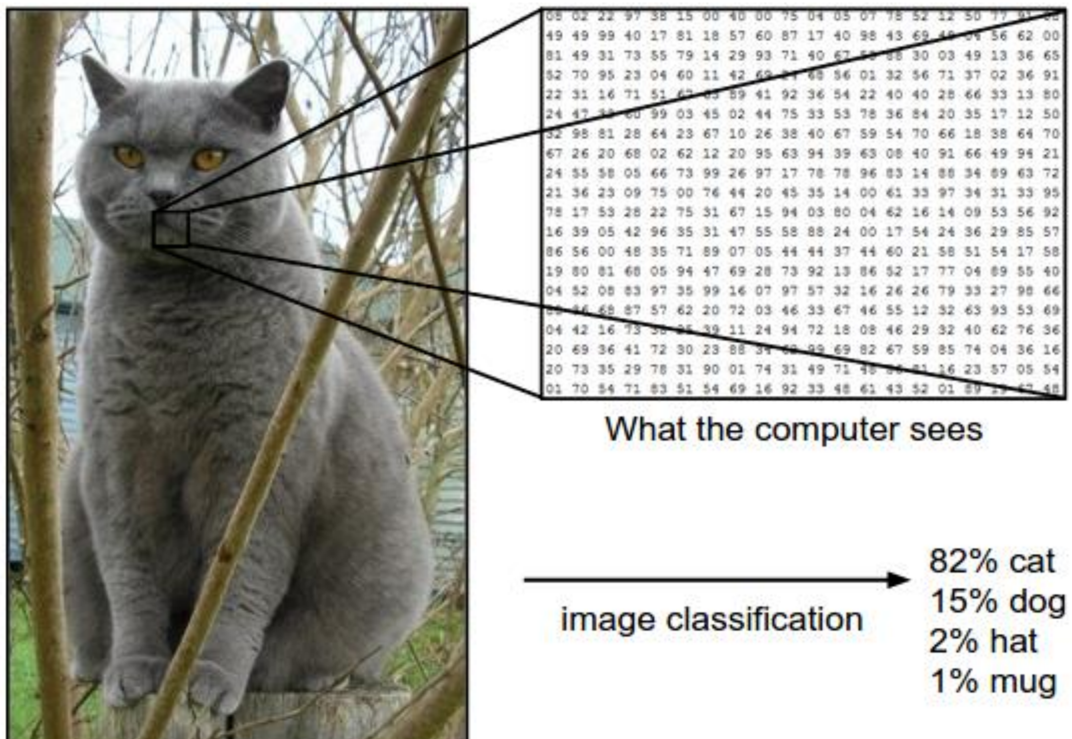


Image Representation

- Image represented as matrix

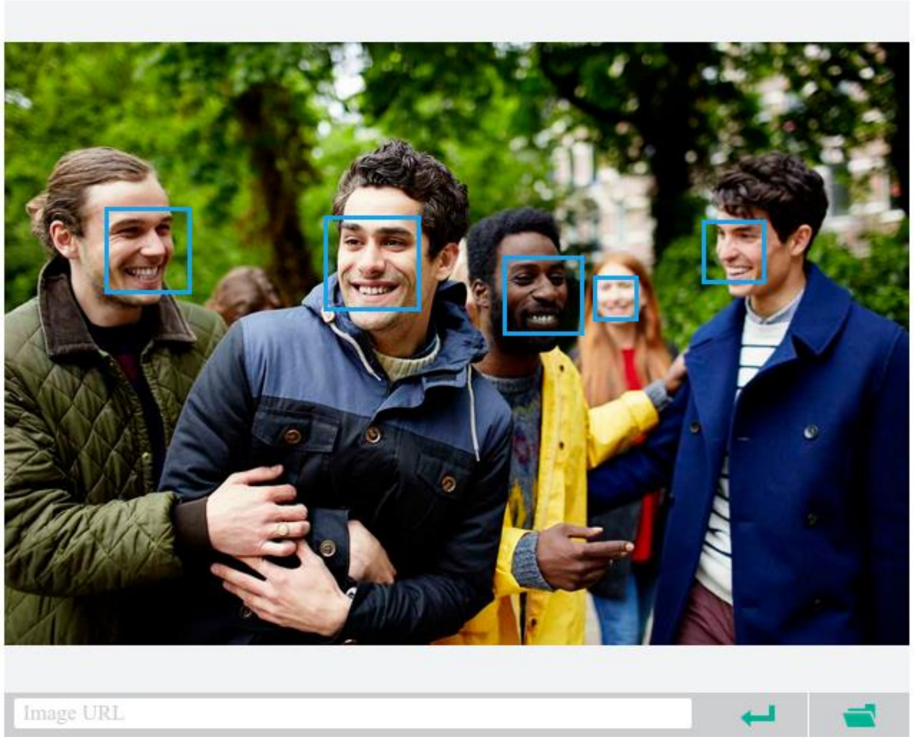


Applications: Face Recognition

- Recognize human face in images

Applications: Face Recognition

- Can also recognize emotions!



Detection Result:
5 faces detected

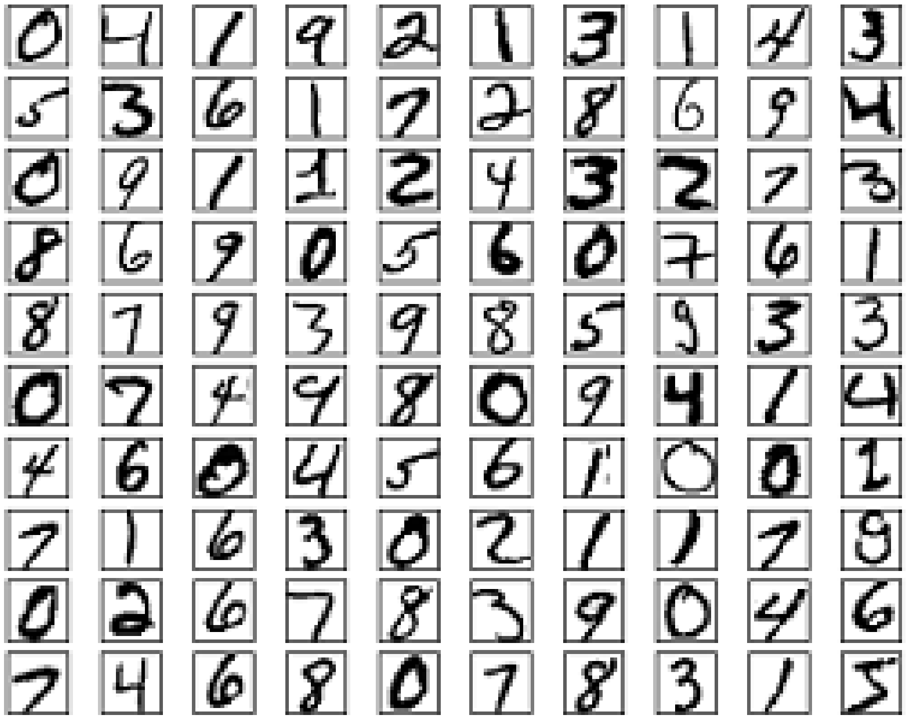
JSON:

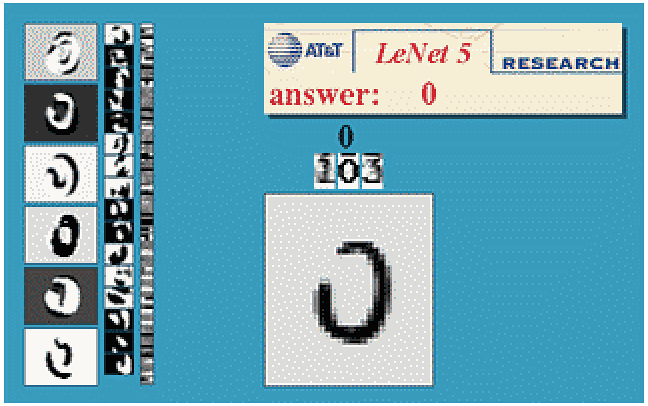

```
[
  {
    "FaceRectangle": {
      "Left": 488,
      "Top": 263,
      "Width": 148,
      "Height": 148
    },
    "Scores": {
      "Anger": 9.075572e-13,
      "Contempt": 7.048959e-9,
      "Disgust": 1.02152783e-11,
      "Fear": 1.778957e-14,
      "Happiness": 0.9999999,
      "Neutral": 1.31694478e-7,
      "Sadness": 6.04054263e-12,
      "Surprise": 3.92249462e-11
    }
  },
  {
    "FaceRectangle": {
      "Left": 153,
      "Top": 251,
      "Width": 133,
```


- Try it yourself @ https://www.projectoxford.ai/demo/emotion

Applications: Hand Written Digits Recognition

- What are the numbers?





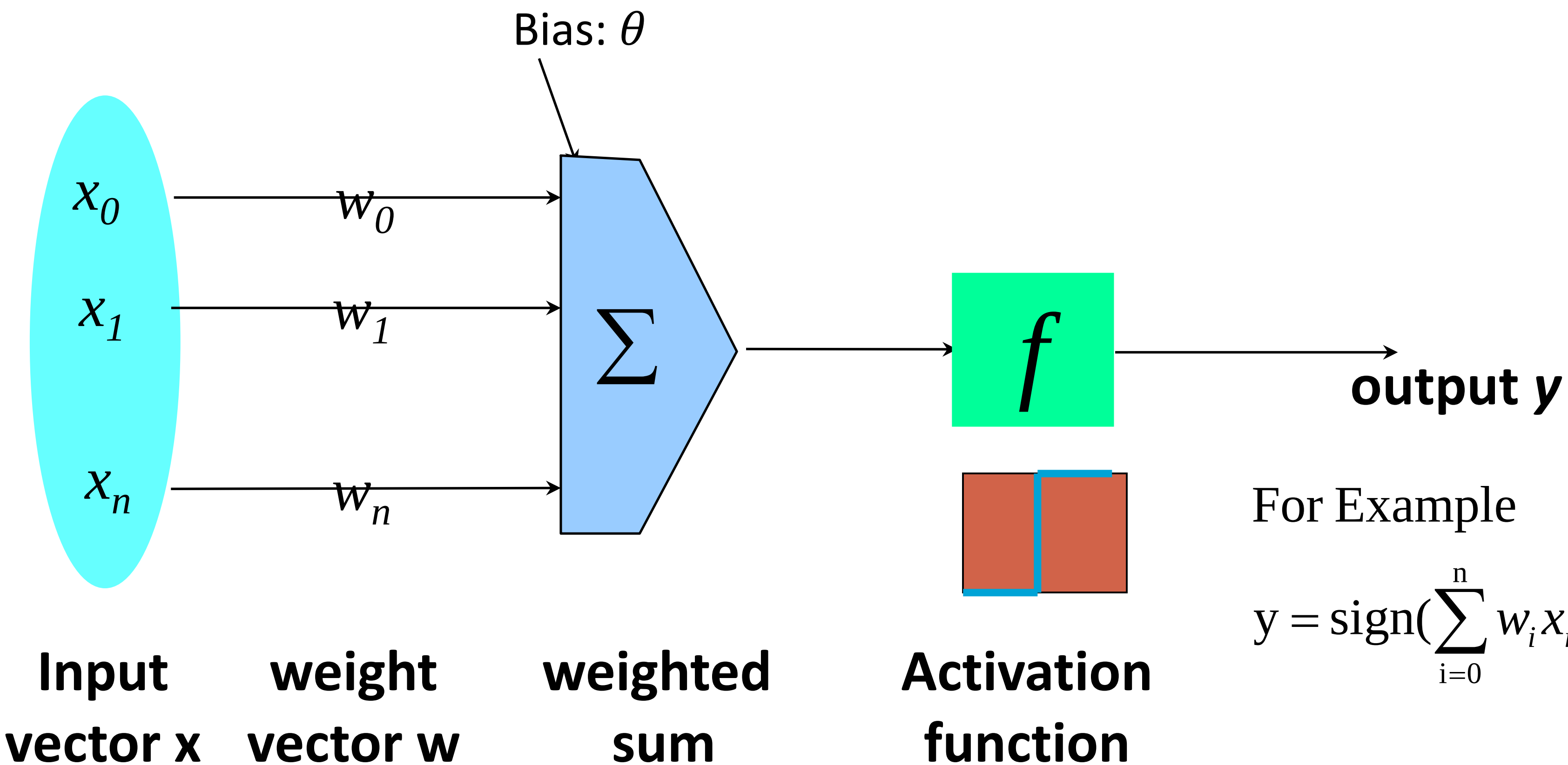
Mining Image Data

- Image Data
- Neural Networks as a Classifier
- Summary

Artificial Neural Networks

- Consider humans:
  - Neuron switching time ~.001 second
  - Number of neurons ~10^{10}
  - Connections per neuron ~10^{4-5}
  - Scene recognition time ~.1 second
  - 100 inference steps doesn't seem like enough -> parallel computation
- Artificial neural networks
  - Many neuron-like threshold switching units
  - Many weighted interconnections among units
  - Highly parallel, distributed process
  - Emphasis on tuning weights automatically

Single Unit: Perceptron



For Example

$$y = \text{sign}\left(\sum_{i=0}^{n} w_i x_i + \theta\right)$$

- An *n*-dimensional input vector **x** is mapped into variable y by means of the scalar product and a nonlinear function mapping

Perceptron Training Rule

For each training data point:

$$w_i \leftarrow w_i + \Delta w_i$$

where

$$\Delta w_i = \eta(t - o)x_i$$

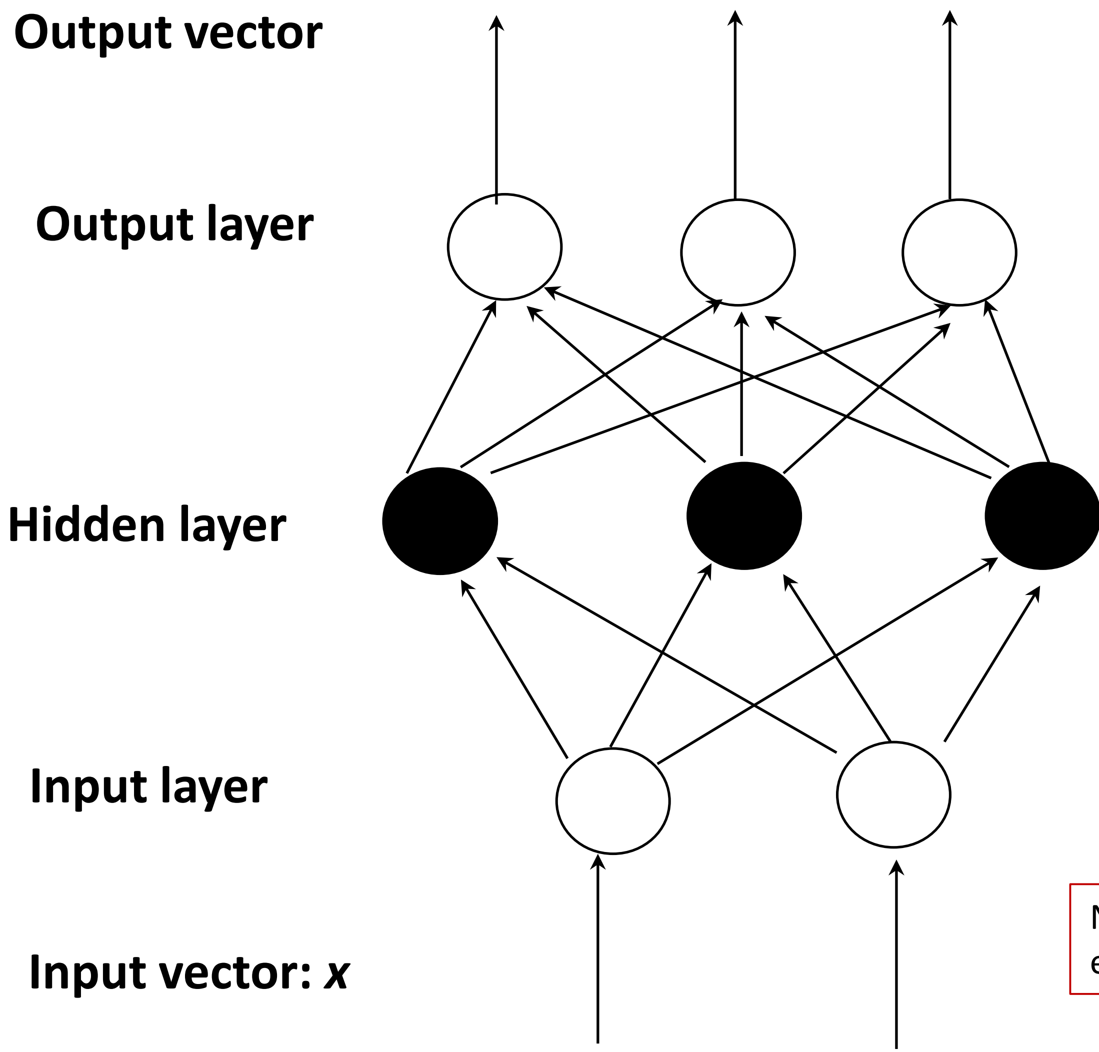
- t: target value (true value)
- o: output value
- η: learning rate (small constant)

- Derived using Gradient Descent method by minimizing the squared error:

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

A Multi-Layer Feed-Forward Neural Network

A **two-layer** network



$$\boldsymbol{y} = g(W^{(2)}\boldsymbol{h} + b^{(2)})$$

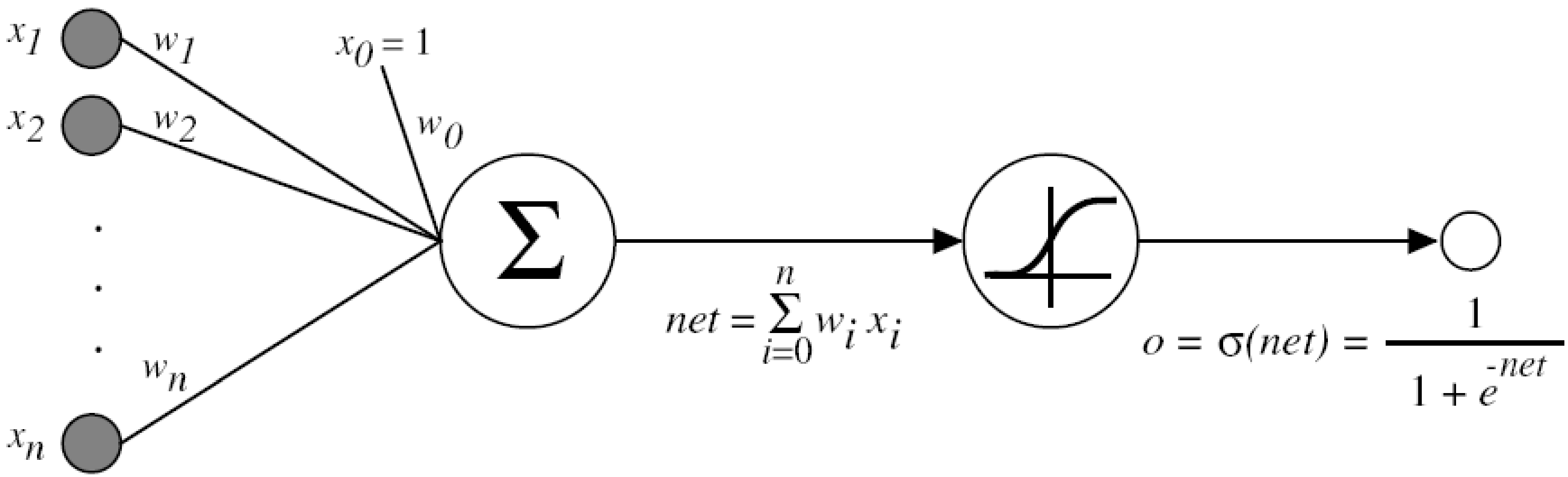
$$\boldsymbol{h} = f(W^{(1)}\boldsymbol{x} + b^{(1)})$$

Bias term

Weight matrix

Nonlinear transformation,
e.g. sigmoid transformation

Sigmoid Unit



- $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function
  - Property: $\frac{d\sigma(x)}{dx} = \sigma(x)(1 - \sigma(x))$
  - Will be used in learning

How A Multi-Layer Neural Network Works

- The **inputs** to the network correspond to the attributes measured for each training tuple
- Inputs are fed simultaneously into the units making up the **input layer**
- They are then weighted and fed simultaneously to a **hidden layer**
- The number of hidden layers is arbitrary, although usually only one
- The weighted outputs of the last hidden layer are input to units making up the **output layer**, which emits the network's prediction
- The network is **feed-forward**: None of the weights cycles back to an input unit or to an output unit of a previous layer
- From a math point of view, networks perform **nonlinear regression**: Given enough hidden units and enough training samples, they can closely approximate any continuous function

Defining a Network Topology

- Decide the **network topology:** Specify # of units in the *input layer*, # of *hidden layers* (if > 1), # of units in *each hidden layer*, and # of units in the *output layer*
- Normalize the **input** values for each attribute measured in the training tuples to [0.0—1.0]
- **Output**, if for classification and more than two classes, one output unit per class is used
- Once a network has been trained and its accuracy is **unacceptable**, repeat the training process with a different network topology or a different set of initial weights

Learning by Backpropagation

- Backpropagation: A **neural network** learning algorithm
- Started by psychologists and neurobiologists to develop and test computational analogues of neurons
- During the learning phase, the **network learns by adjusting the weights** so as to be able to predict the correct class label of the input tuples
- Also referred to as **connectionist learning** due to the connections between units

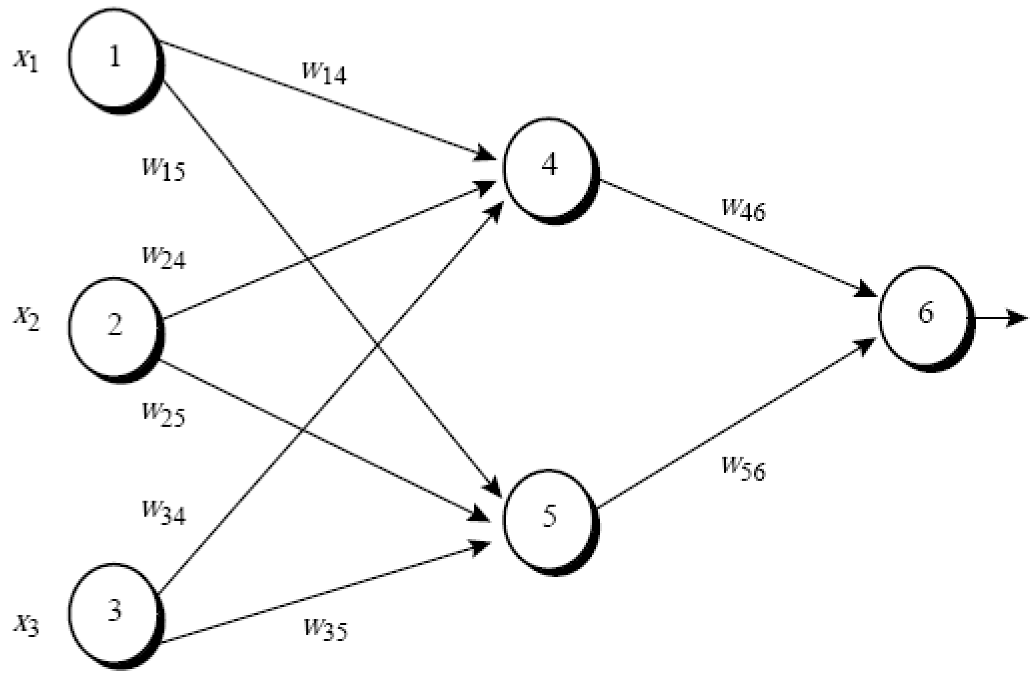
Backpropagation

- Iteratively process a set of training tuples & compare the network's prediction with the actual known target value
- For each training tuple, the weights are modified to **minimize the mean squared error** between the network's prediction and the actual target value
- Modifications are made in the "**backwards**" direction: from the output layer, through each hidden layer down to the first hidden layer, hence "**backpropagation**"

Backpropagation Steps to Learn Weights

- Initialize weights to small random numbers, associated with biases
- Repeat until terminating condition meets
  - For each training example
    - **Propagate the inputs forward** (by applying activation function)
      - For a hidden or output layer unit j
        - Calculate net input: $I_j = \sum_i w_{ij} O_i + \theta_j$
        - Calculate output of unit j: $O_j = \frac{1}{1+e^{-I_j}}$
    - **Backpropagate the error** (by updating weights and biases)
      - For unit j in output layer: $Err_j = O_j(1 - O_j)(T_j - O_j)$
      - For unit j in a hidden layer: : $Err_j = O_j(1 - O_j) \sum_k Err_k w_{jk}$
      - Update weights: $w_{ij} = w_{ij} + \eta Err_j O_i$
- Terminating condition (when error is very small, etc.)

Example



A multilayer feed-forward neural network

x_1	x_2	x_3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}	θ_4	θ_5	θ_6
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2	-0.4	0.2	0.1

Initial Input, weight, and bias values

Example

- Input forward:

Table 9.2: The net input and output calculations.

Unit j	*Net input*, I_j	*Output*, O_j
4	$0.2 + 0 - 0.5 - 0.4 = -0.7$	$1/(1 + e^{0.7}) = 0.332$
5	$-0.3 + 0 + 0.2 + 0.2 = 0.1$	$1/(1 + e^{-0.1}) = 0.525$
6	$(-0.3)(0.332) - (0.2)(0.525) + 0.1 = -0.105$	$1/(1 + e^{0.105}) = 0.474$

- Error backpropagation and weight update:

Table 9.3: Calculation of the error at each node.

Unit j	Err_j
6	$(0.474)(1 - 0.474)(1 - 0.474) = 0.1311$
5	$(0.525)(1 - 0.525)(0.1311)(-0.2) = -0.0065$
4	$(0.332)(1 - 0.332)(0.1311)(-0.3) = -0.0087$

Table 9.4: Calculations for weight and bias updating.

Weight or bias	*New value*
w_{46}	$-0.3 + (0.9)(0.1311)(0.332) = -0.261$
w_{56}	$-0.2 + (0.9)(0.1311)(0.525) = -0.138$
w_{14}	$0.2 + (0.9)(-0.0087)(1) = 0.192$
w_{15}	$-0.3 + (0.9)(-0.0065)(1) = -0.306$
w_{24}	$0.4 + (0.9)(-0.0087)(0) = 0.4$
w_{25}	$0.1 + (0.9)(-0.0065)(0) = 0.1$
w_{34}	$-0.5 + (0.9)(-0.0087)(1) = -0.508$
w_{35}	$0.2 + (0.9)(-0.0065)(1) = 0.194$
θ_6	$0.1 + (0.9)(0.1311) = 0.218$
θ_5	$0.2 + (0.9)(-0.0065) = 0.194$
θ_4	$-0.4 + (0.9)(-0.0087) = -0.408$

Efficiency and Interpretability

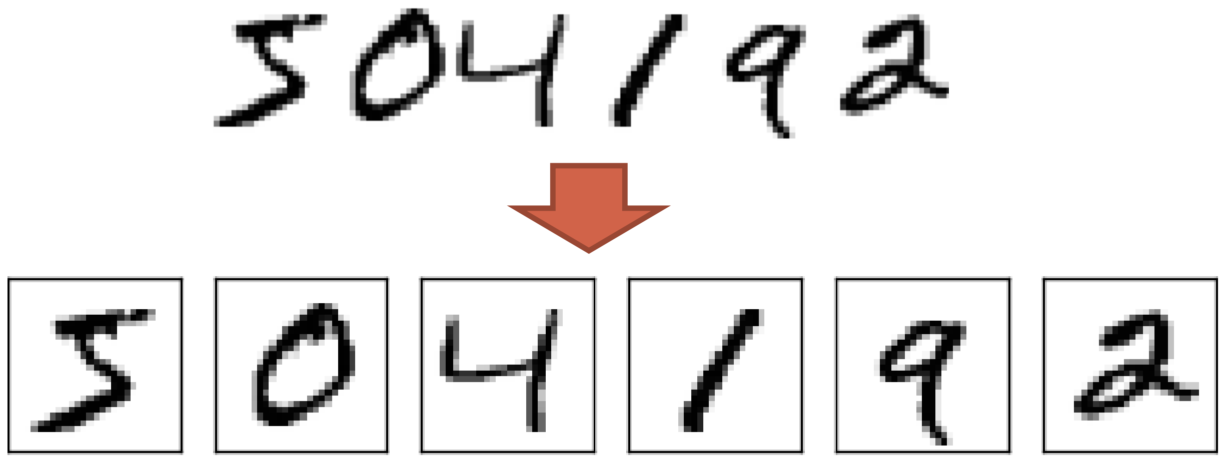
- **Efficiency** of backpropagation: Each iteration through the training set takes O(|D| * *w*), with |D| tuples and *w* weights, but # of iterations can be exponential to n, the number of inputs, in worst case
- For easier comprehension: **Rule extraction** by network pruning
  - Simplify the network structure by removing weighted links that have the least effect on the trained network
  - Then perform link, unit, or activation value clustering
  - The set of input and activation values are studied to derive rules describing the relationship between the input and hidden unit layers
- **Sensitivity analysis**: assess the impact that a given input variable has on a network output. The knowledge gained from this analysis can be represented in rules
  - E.g., If x decreases 5% then y increases 8%

Neural Network as a Classifier

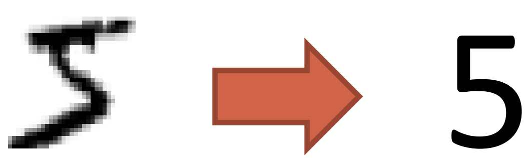
- Weakness
  - Long training time
  - Require a number of parameters typically best determined empirically, e.g., the network topology or "structure."
  - Poor interpretability: Difficult to interpret the symbolic meaning behind the learned weights and of "hidden units" in the network
- Strength
  - High tolerance to noisy data
  - Well-suited for continuous-valued inputs *and outputs*
  - Successful on an array of real-world data, e.g., hand-written letters
  - Algorithms are inherently parallel
  - Techniques have recently been developed for the extraction of rules from trained neural networks

Digits Recognition Example

- Obtain sequence of digits by segmentation

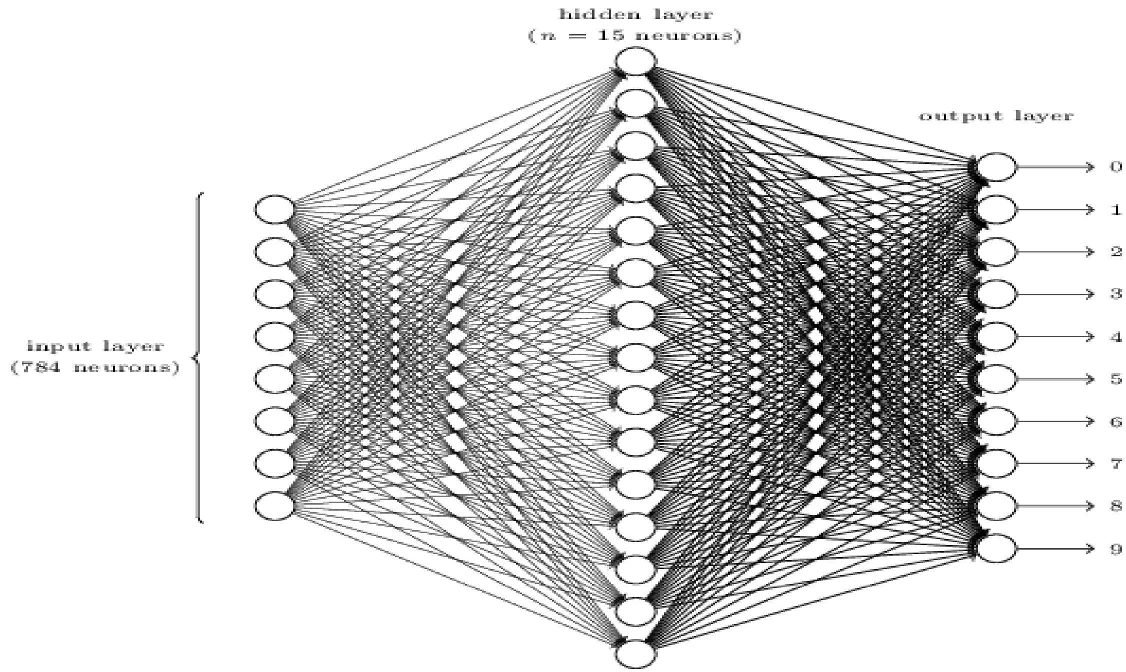


- Recognition (our focus)

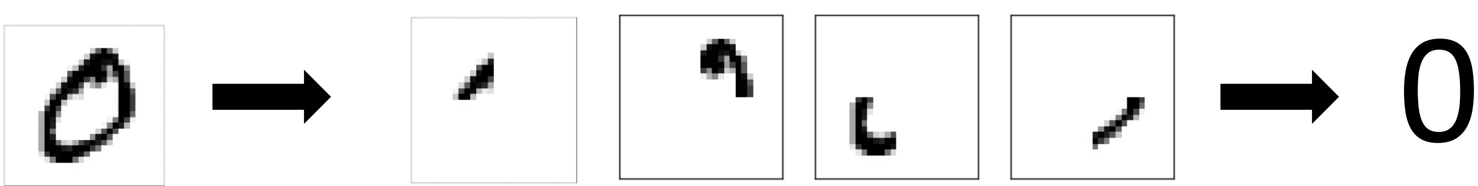


Digits Recognition Example

- The architecture of the used neural network



- What each neurons are doing?

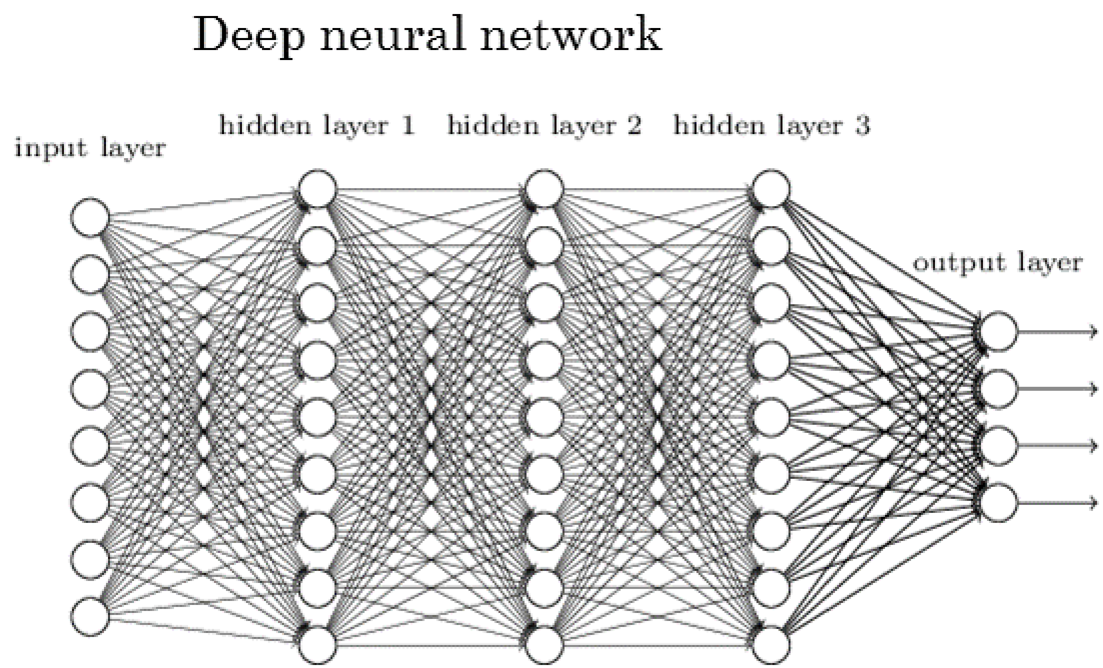


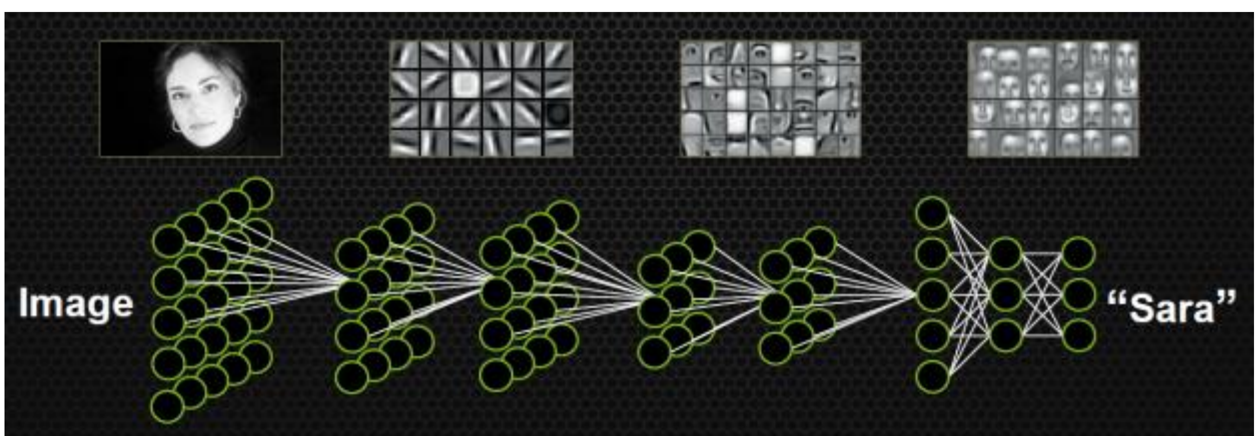
Input image

Activated neurons detecting image parts

Predicted number

Towards Deep Learning





Mining Image Data

- Image Data
- Neural Networks as a Classifier
- Summary

Summary

- Image data representation
- Image classification via neural networks
  - The structure of neural networks
  - Learning by backpropagation