Machine Learning

Machine Learning

Machine? Learning?

- A computer. Improve through experience

Build computer programs

- Improve itself at some task
- Through experience

Experience

- Through training examples

Machine Learning

Subfield of computer science

Give computers ability to learn

- Without being explicitly programmed

Study and construction of algorithms

- Learn from and make predictions on data

Employed in a range of computing tasks

- Programming explicit algorithms is infeasible
- Spam filtering, detection of network intruders, optical character recognition (OCR) and search engines

Relations to Other Subjects

Artificial Intelligence

- Provide a way for implementing A.I.
- A.I. – methods for human-level cognitive tasks
  - Most cognitive tasks – classification / prediction

Statistics

- Learning with mathematics
- Mostly aimed for prediction tasks
- Most ML methods are from statistics
  - Training data / experience E

Relations to Other Subjects

Pattern Recognition

- One of applications of A.I.
  - An application of Machine Learning Methods
  - For signals, graphics, and multimedia
- Mostly for classification tasks

Data Mining

- Machine learning with very large or distributed data sets
  - Traditional methods only work for limited-size data set
  - New efficient algorithms are necessary

Classification of Machine Learning Tasks

Classification of Machine Learning Tasks

Supervised learning

- Present computer with examples
  - Inputs and their desired outputs
- Given by a “teacher”
- Learn a general rule that maps inputs to outputs

Unsupervised learning

- No labels are given
- Find structure of input its own
- Discover hidden patterns in data

Classification of Machine Learning Tasks

Semi-supervised learning

- Between supervised and unsupervised learning
- Teacher gives an incomplete training signal
  - Training set with some or many missing target outputs

Reinforcement learning

- Computer program interacts with a dynamic environment
- Perform a certain goal
  - Drive vehicle or play game against an opponent
- Feedback are provided
  - In terms of rewards and punishments
  - When it navigates its problem space

Supervised Learning

Infer a function from labeled training data

- Analyze training data
- Produce an inferred function
- Use for mapping new examples

Training data

- A set of training examples
- Each is a pair
  - Input object (typically a vector)
  - Desired output value (also called supervisory signal)

Optimal scenario

- Determine class labels for unseen instances correctly

Supervised Learning

Given data $\{(x_1, y_1), \ldots, (x_n, y_n)\}$

Seek a function that explains relationship between

- Input attribute x
- Output attribute y
- $y = f(x) + \epsilon$

Algorithms

- Naive Bayes classifier
- Neural network
- Support vector machines
- Nearest Neighbor Algorithm

Unsupervised Learning

Infer a function

- Describe hidden structure from unlabeled data

Distinguishable from other learning schemes

- Examples given are unlabeled
- No error or reward signal
- Cannot evaluate a potential solution
- No objective evaluation
  - Accuracy of the structure output

Approaches

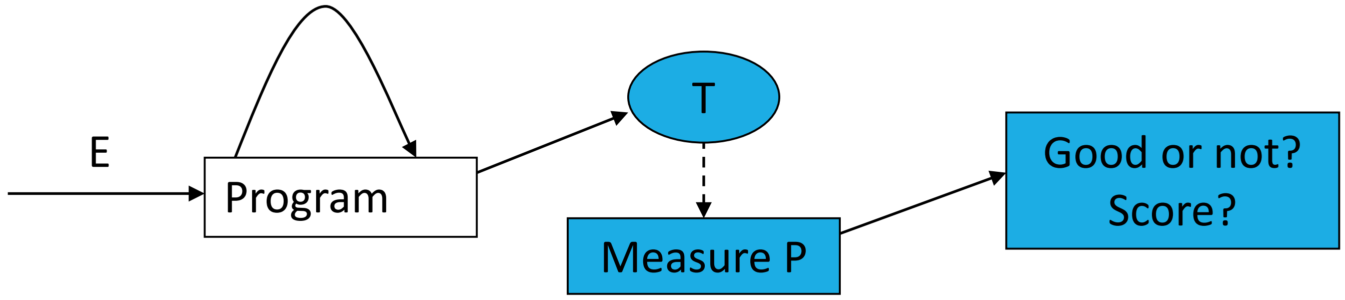
- Clustering

Supervised Learning

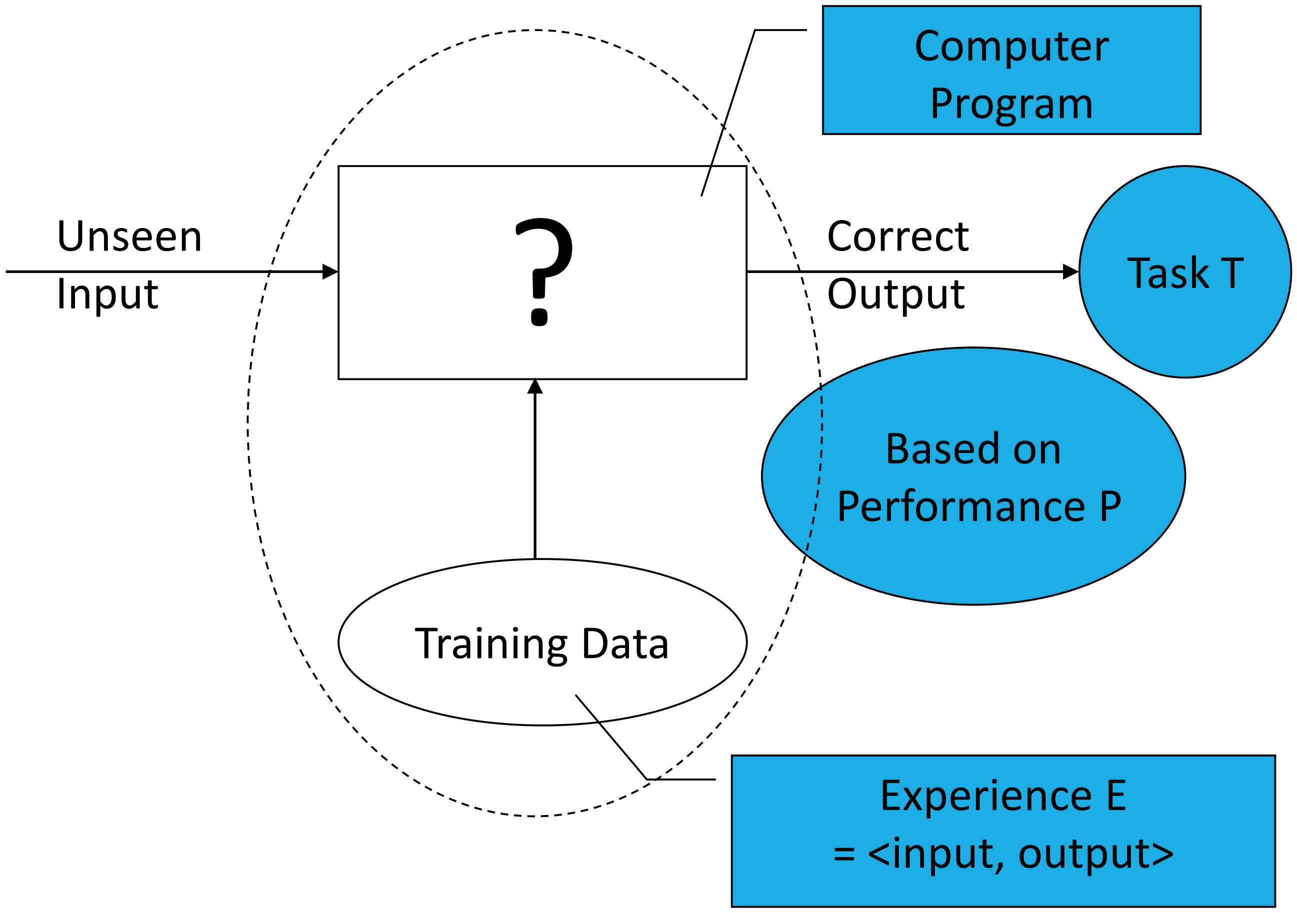
Supervised Learning

A computer program

- Learn from experience E
  - In some class of tasks T with performance P
- Its performance at tasks in T improves with experience E
  - Performance is measured by P



Supervised Learning



Tasks for Supervised Learning

Classification / Prediction

By making a target function

- Estimate a mathematical model
- Through a set of training examples

Only outputs are different

- Classification – Discrete values
  - True or false, Yes or no, Class A to class F
- Prediction – Continuous values

Examples of Learning Tasks

Classification

- Recognize spoken words
- Classify new astronomical structures

Prediction

- Stock trend
- Robot control

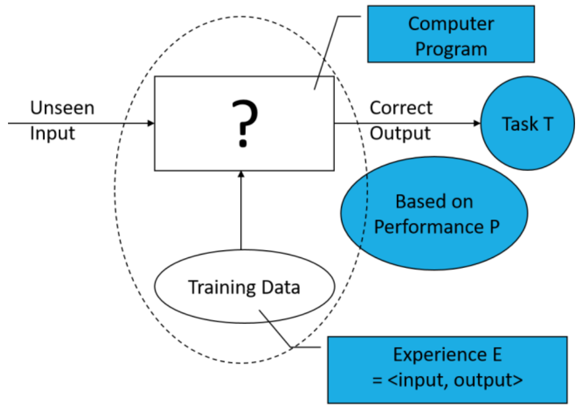
Summary of Supervised Learning

A job of estimating a function (program, ?)

- Take E (training data) as inputs
- Output as accurate as possible
  - Solve problems in task T

Learning job may be iterative

- Improve the function accuracy
  - Not just one time but many times



Designing a Supervised Learning System

Designing a Supervised Learning System

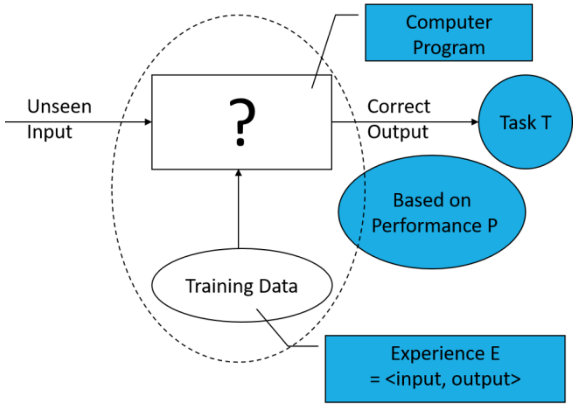
1. Exact type of knowledge to be learned, i.e., the '?'
   - A program? A decision tree? Logical rules? A function?
   - Usually a mathematical function
2. Representation for this target knowledge, or function
   - Format of f(x)? Sine wave? Exponential?
   - Usually f(x) = wx + b or polynomial
3. A function approximation algorithm

Example Problem Domain

Chess playing program
- Input: Any board state
- Output: Legal moves

Experience E = <input, output>
- E = (State, Legal moves)

Learn = Make the function '?'
- Under any state
  - Return the *best* move among legal moves
- *Best* is judged by performance *P*
  - Rules of the game



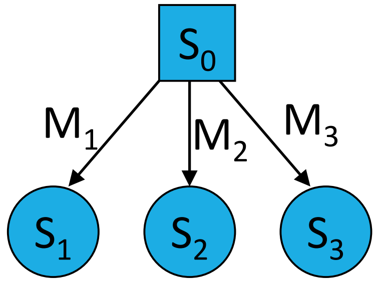
Example Problem Domain

Function '?' = *ChooseMove*

- *ChooseMove: State → Move*
- How to compare resulting *Move*?

Easier alternative

- *y*: State → R
  - A function evaluating any state into a real number
  - The higher the score, the better the state
- State S_0
  - Under legal moves M_1, M_2, M_3, …
  - Generates S_1, S_2, S_3, …
  - *y*: S_1 → R_1, *y*:S_2 → R_2, …
  - Choose the best state (i.e. the move)



$$\begin{tikzcd} & S_0 \arrow[dl, "M_1"'] \arrow[d, "M_2"] \arrow[dr, "M_3"] & \\ S_1 & S_2 & S_3 \end{tikzcd}$$

Example Problem Domain

Learned function is changed

- From *symbolic* to *numerical*

Function y is usually not efficiently computable

- Hard to find it exactly

Our target

- Estimate an approximate function $\hat{y}$ to replace y
  - Representation of $\hat{y}$

Example Problem Domain

Representations for description of $\hat{y}$

- A look up table
- A collection of logical rules
- A polynomial function
- A simple linear combination

$$\hat{y}(S) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$
$$\text{where } S = < x_1, x_2, \ldots, x_n >$$

A Basic Function Approximation Algorithm

Data for Training Algorithm

A set of training examples

- Each is a pair
  - <S, $Value_S$> = <input, output>
  - S = State, $Value_S$ = Value of State S
- E.g. <(x_1=1, x_2=3, x_3=0, …), 50>
  - Record in a database
  - Many records in the database

1	(x_1=1, x_2=3, x_3=0, …)	50
2	…	40
⋮	⋮	⋮
m	…	35

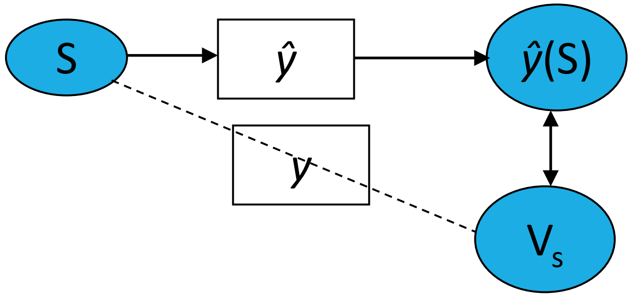
Evaluating Function Estimation

Approximating $\hat{y}$ using training data

- E = (S, Value$_S$)
- A score $\hat{y}$(S) can be obtained for state S
  - Evaluation of S

$$\hat{y}(S) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$\text{where } S = < x_1, x_2, \ldots, x_n >$$



Error can be measured

- Between True and Estimated values
- $\text{Error}_S = (\text{Value}_S - \hat{y}(S))^2$

$$\text{Error} = \sum_{\substack{<S,\, \text{Value}_S> \\ \in \text{Training Data}}} (\text{Value}_S - \hat{y}(S))^2$$

1	(x_1=1, x_2=3, x_3=0, …)	50
2	…	40
… m	…	35

Evaluating Function Estimation

Find $\hat{y}$

- Such that *error* can be minimized

Since $\hat{y} = \Sigma w_i x_i$, $i = 0$ *to* n

- x_i are constants (input data)
- Adjust $w_i \rightarrow$ Adjust $\hat{y}$
  - Minimize the error
- Algorithm is called LMS (Least Mean Square)
  - A very rational and easy training rule

$$\hat{y}(S) = w_0 + w_1 x_1 + w_2 x_2 + \ldots + w_n x_n$$

$$\text{where } S = < x_1, x_2, \ldots, x_n >$$

Adjust Weights

For each training example <S, Value$_S$>

- Use current weights w_i
  - Calculate $\hat{y}(S)$
- For each weight w_i, update it as

$$\hat{y} = \Sigma w_i x_i$$

$$w_i \leftarrow w_i + \eta\ (Value_S - \hat{y}(S))\ x_i$$

$$w_2 \leftarrow 1 + 0.05\ (50 - 48)\ 3 = 1.3$$

A small constant controlling learning rate, “eta”

Neural Network

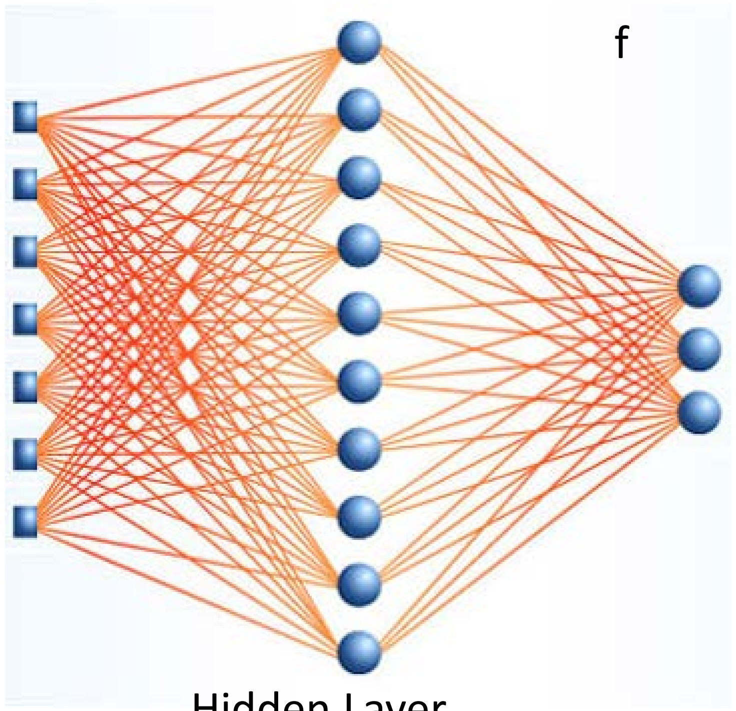
Neural Network

A famous and powerful learning method

- Linear and nonlinear target function
- Single output
  - Real-valued, discrete-valued
- Multiple output
  - Vector-valued
- Other features
  - Robust (insensitive to noise)
    - Noise = some misleading / incorrect input values
  - Easy to implement
  - Fast and efficient
  - But hard to interpret

Neural Network Representation

Consist of at least three layers



x

Input Layer

y

Output Layer

Hidden Layer

y=f(**x**)

Neural Network Representation

Input layer

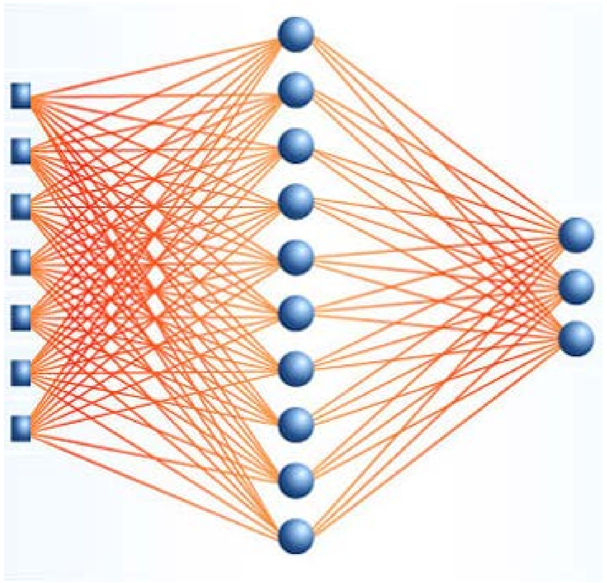
- Accept values of training examples <**x**, **y**>
  - If **x** is 7-tuple, then 7 input neurons

Output layer

- Similar to input layer
  - If **y** is triple, then 3 output neurons

Hidden layer

- Handle unknowns
- Handle nonlinearity within data
- Number of hidden neurons
  - Usually > input neurons



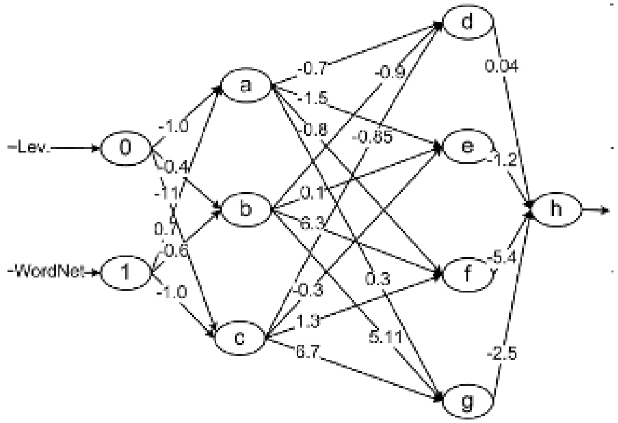
Neural Network Representation

Between any two layers

- Connected by some arcs
- Associated with a weight

Train neural network

- Update the weights in layers
- Fixed representation for target function



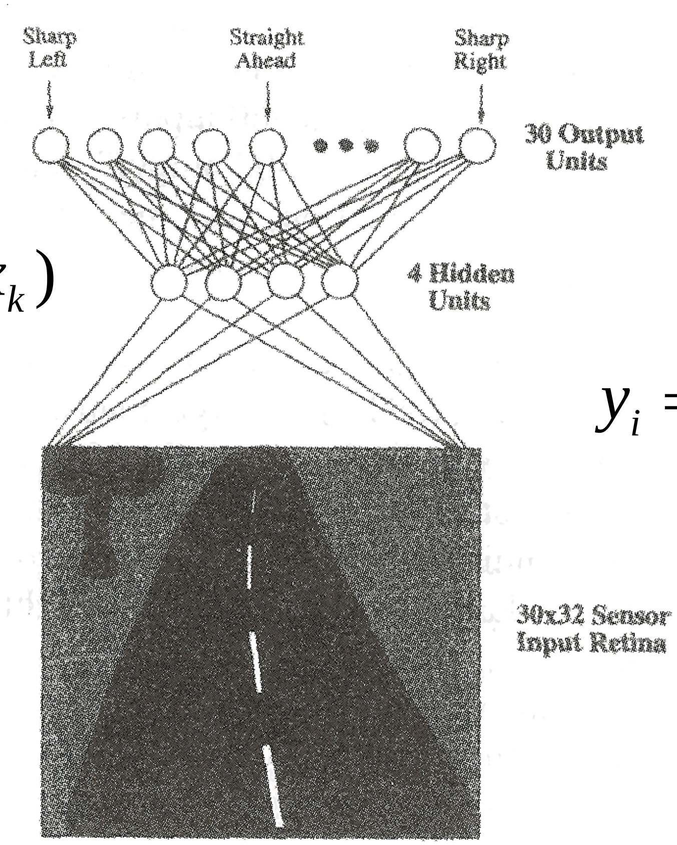
$$y = \sum_i w_{3i}\varphi_i \left\{ \sum_j w_{2ij}[\sigma_j(\sum_k w_{1jk}x_k)] \right\}$$

Example of Steering Control

y_i

$\sigma_j(\sum_k w_{1jk} x_k)$

x_k



Output Layer

A binary-vector

[0 0 0 1 0 … 0]

$$y_i = \sum_j w_{2ij}[\sigma_j(\sum_k w_{1jk} x_k)]$$

Input Layer

= 30 x 32

= 960 neurons

Appropriate Problem

Problems have

- Instances (data records)
  - Many attribute-value pairs
  - Input = matrix or vector
- Target function may be any data type
  - Discrete-valued, real-valued, or a vector of attributes
- Noisy training data

Properties of NN learning

Training takes long time

- From seconds to hours
- Or even days or weeks

Trained target function

- Execution takes very short time
  - Several seconds
- NN structure
  - Linear combination of weights and inputs
  - $y = wx+b = \mathbf{wx}$

Support Vector Machines

Support Vector Machines

Abbreviated as SVM

Main application

- Classification
- Regression
  - Model/Function estimation
  - Approximation

Support Vector Machines

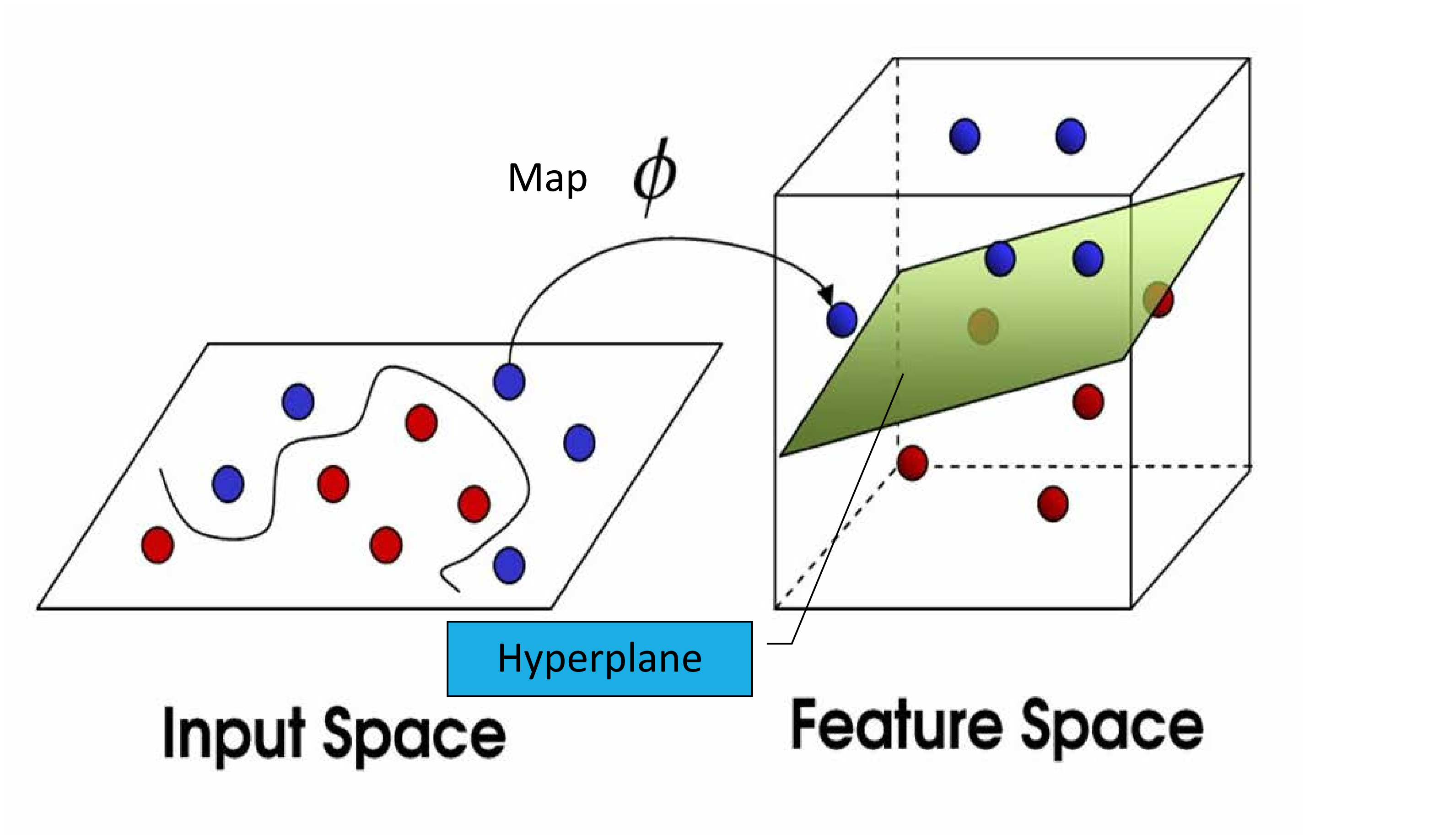
Can only build linear functions

- Goal: nonlinear functions
  - Done by some nonlinear transformation ϕ
  - Nonlinear input spaces → High dimensional linear feature spaces
  - Build linear functions

Special property

- Required hidden units are automatically determined

Support Vector Machines

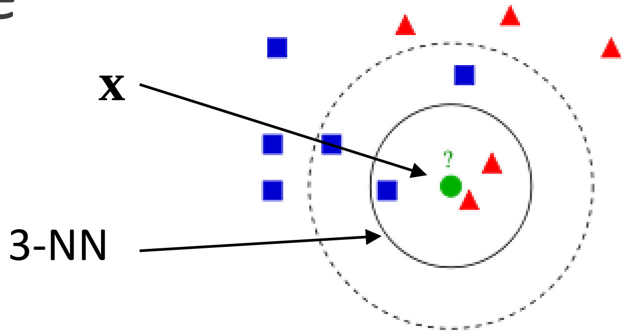


k-Nearest Neighbor Method

k-Nearest Neighbor

Nearest neighbors of an instance

- Defined on Euclidean distance



Euclidean distance

- An instance **x** = <x_1, x_2, …, x_n>
- Distance between two instances **x, y**

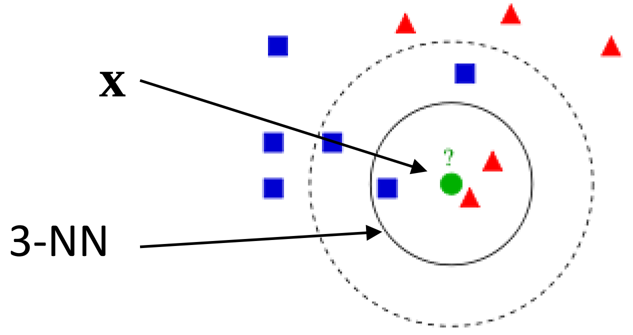
$$d(\mathbf{x},\mathbf{y}) \equiv \sqrt{\sum_{r=1}^{n} (x_r - y_r)^2}$$

k-NN for Discrete Values (Classes)

Compare query $\mathbf{x}$

- With every training example $<\mathbf{y}, f(\mathbf{y})>$
- For $k = 1$
  - Return most similar object $\hat{\mathbf{y}}$
    - Based on Euclidean equation
  - Assign $f(\hat{\mathbf{y}})$ to $f(\mathbf{x})$
- For $k > 1$
  - Retrieve a set of k similar instances $\hat{\mathbf{y}}_\mathbf{i}$

Class of $\mathbf{y}$



$$f(\mathbf{x}) \leftarrow \underset{v \in \mathrm{V}}{\operatorname{argmax}} \sum_{i=1}^{k} \delta(v, f(\hat{\mathbf{y}}_i))$$

$$\delta(\mathrm{a}, \mathrm{b}) = 1 \text{ if } \mathrm{a} = \mathrm{b}$$
$$\delta(\mathrm{a}, \mathrm{b}) = 0 \text{ if } \mathrm{a} \neq \mathrm{b}$$

$\mathrm{V} = \text{all possible classes}$ {square, tri}

k-NN for Continuous Target Values

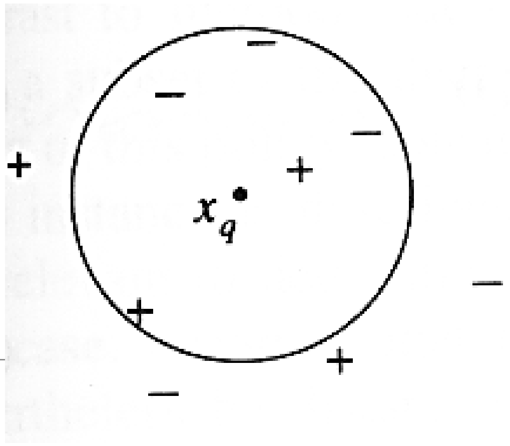
Replace function of estimating class

$$f(\mathbf{x}) \leftarrow \frac{1}{k} \sum_{i=1}^{\mathrm{k}} f(\hat{\mathbf{y}}_i)$$

Continuous target values

- Return the mean of
  - Target values of k similar nearest neighbors

Distance-Weighted NN



Refinement to kNN

- k similar instances are weighted
  - Based on their distance to $\mathbf{x}$

Suggested weights for instance

$$w_i \equiv \frac{1}{d(\mathbf{x}, \mathbf{y}_i)^2}$$

If $d(\mathbf{x}, \mathbf{y}_i)$ is zero $f(\mathbf{x}) = f(\mathbf{y}_i)$

For discrete value: target class

$$f(\mathbf{x}) \leftarrow \underset{v \in V}{\operatorname{argmax}} \sum_{i=1}^{k} w_i \delta(v, f(\hat{\mathbf{y}}_i))$$

Distance-Weighted NN

For continuous value: target value

$$f(\mathbf{x}) \leftarrow \frac{\sum_{i=1}^{k} w_i \, f(\hat{\mathbf{y}}_i)}{\sum_{i=1}^{k} w_i} \qquad w_i \equiv \frac{1}{d(\mathbf{x}, \mathbf{y}_i)^2}$$

If $w_i = 1$ for all i, this term $= k$
i.e. kNN

Remarks on kNN

Pros

- Distance-weighted kNN is highly effective
- Robust to noisy training data
  - Provided a large set of training examples

Cons

- Similarity metric depends on all attributes
  - Some attributes may be irrelevant = noise
    - Misleading / wrong
- Distance metric = Euclidean space
  - Do not guarantee that it can represent similarity

Unsupervised Learning – Clustering

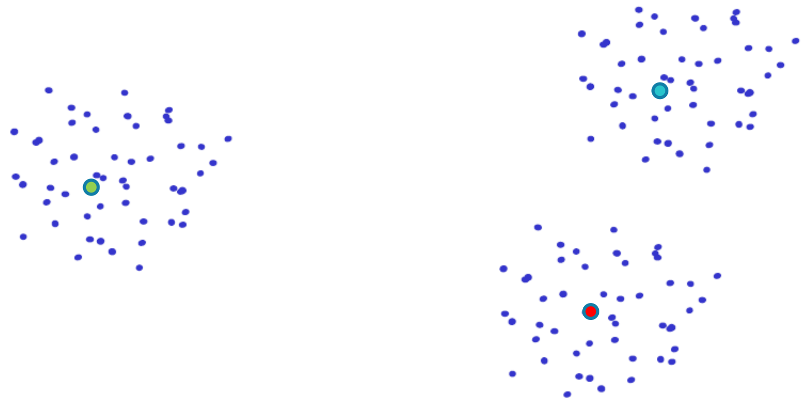
Clustering

Divide a set of objects into groups

- Objects in same group are similar
- Objects in different groups are not similar

Input

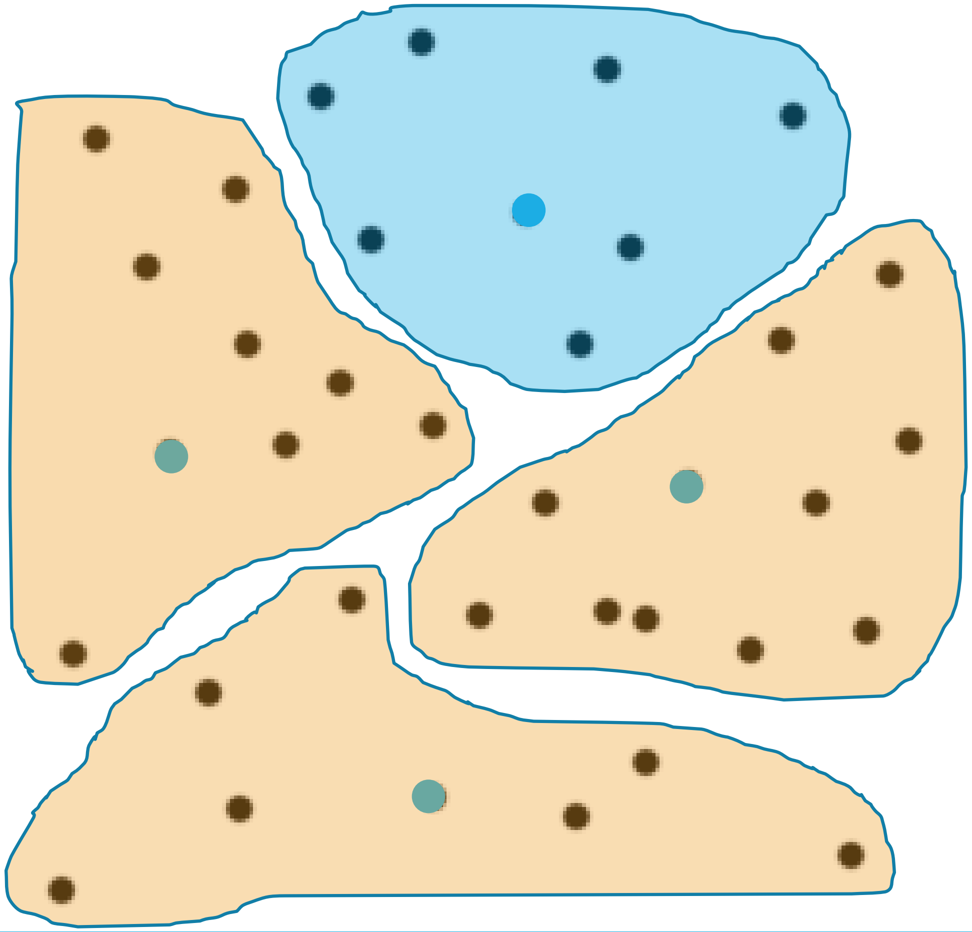
- A set of points P
- A set of centers C (optional)



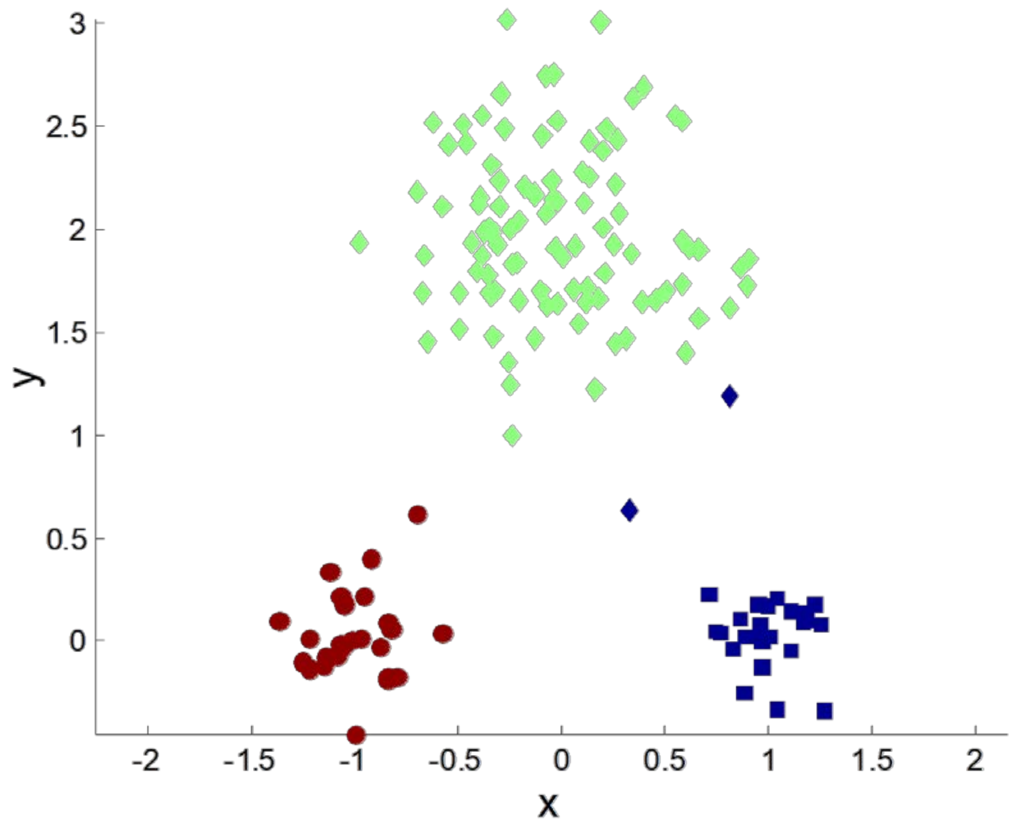
Clustering

- Assign every point of P to the nearest center in C

Clustering Problem



Clustering Problem



Applications of Clustering

Image Processing

- Cluster images
  - Based on visual content

Web

- Cluster groups of users
  - Based on access patterns on webpages
- Cluster webpages
  - Based on content

Bioinformatics

- Cluster similar proteins together
  - Similarity in chemical structure or functionality

Clustering Problem

Discrete vs continuous clustering

Discrete clustering

- Restrict centers of clustering to a subset of input

Continuous clustering

- Centers might be placed anywhere in given metric space

Clustering Algorithm

K-Centers clustering

K-Median clustering

K-Means clustering

Hierarchical clustering

K-Means Clustering

Minimize function:

$$E(\Gamma, V) = \sum_{i=1}^{k} \sum_{j=1}^{n} \gamma_{ij} \left\| \overline{x}_j - \overline{v}_i \right\|^2$$

Data points: $X = \{\overline{x}_1, \overline{x}_2, \cdots, \overline{x}_n\}$

Clusters: $C_1, C_2, \cdots C_k$

Centers : $V = \{\overline{v}_1, \overline{v}_2, \cdots, \overline{v}_k\}$

Partition matrix: $\Gamma = \{\gamma_{ij}\}$

$$\gamma_{ij} = \begin{cases} 1 & \text{if } \overline{x}_j \in C_i \\ 0 & \text{otherwise} \end{cases}$$

Iterative algorithm

- Initialize the centers *V* (the position of k centers)
  - By randomly picking points from *X*
- Assign each data point to the nearest center
  - Recalculate partition matrix and *E*(*Γ*, *V*)
- Adjust the position of each center
- Repeat above two steps until convergence

K-Means Clustering

Disadvantages

- Dependent on initialization