인공지능

22년 삼성 AI 전문가과정
6월 8일 수요일 6교시
장병탁

15차시 : Machine Learning

서울대학교 컴퓨터공학부
담당 교수: 장병탁

Seoul National University
Byoung-Tak Zhang




Lecture Overview

인공지능

15차시 : Machine Learning

서울대학교 컴퓨터공학부
담당 교수: 장병탁

Seoul National University
Byoung-Tak Zhang



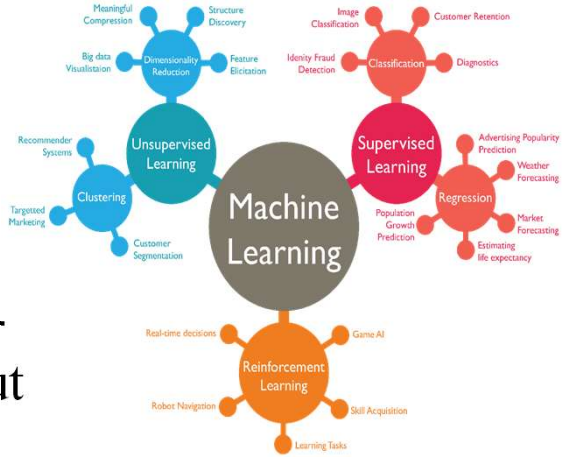

Introduction: Machine Learning

- **Knowledge Representation and Reasoning (Previous lectures)**
  - How to build a model of the world (or to represent the knowledge about the world).
  - How to make inferences (using the model) to predict the world
  - The models (or representations) were usually designed or built manually (not automatically).
- **Machine Learning (This lecture)**
  - We describe agents that can improve their behavior through diligent study of past experiences and predictions about the future.
  - Machine learning builds the models (or representations) automatically from data.
  - Decision trees, linear models, nonparametric models, ensemble models, etc.
- **Why we care about machine learning?**
  - The designers cannot anticipate all possible future situations
  - Sometimes the designers have no idea how to program a solution themselves

Learning Tasks

Forms of learning: Feedbacks to learn from Text

- **Unsupervised learning:** learns patterns in the input without explicit feedback, such as clustering
- **Reinforcement learning:** learns from a series of reinforcements – rewards or punishments
- **Supervised learning:** observes some example input-output pairs and learn a function that maps from input to output
- **Semisupervised learning:** given a few labeled examples and a large collection of unlabeled examples



Google. (n.d.). Google search. Retrieved February 25, 2022, from https://www.google.com/search?q=machine%2Blearning&source=lnms&tbm=isch&sa=X&ved=2ahUKEwj2ht_t8Zr2AhXLDt4KHcSYCBQQ_AUoAXoECAIQAw#imgrc=AS58sENCemFFvM

Learning Models

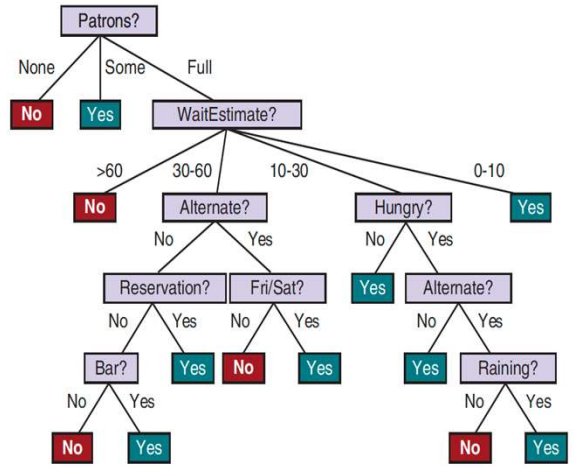
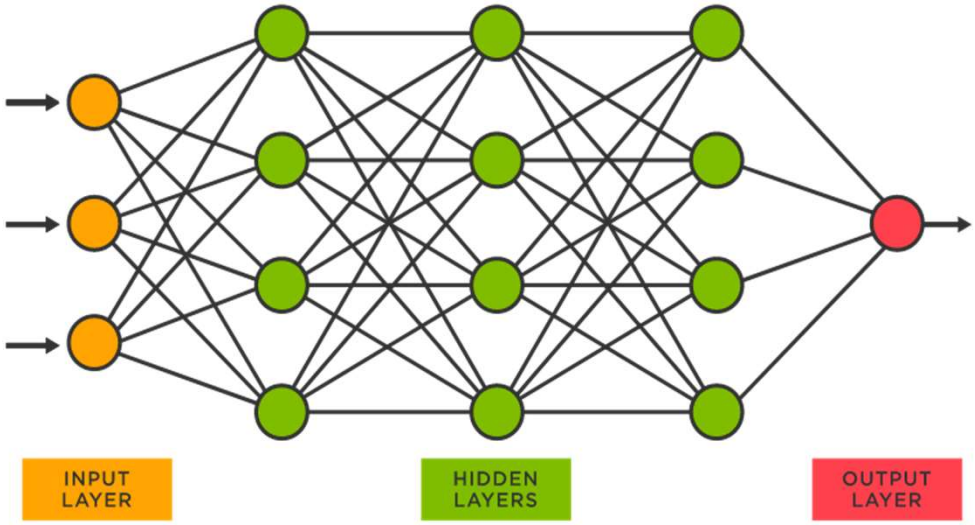


Figure 19.3 A decision tree for deciding whether to wait for a table.



Decision Trees (DT)

Neural Networks (NN)

Learning Algorithms

➢ Entropy to Choose Attributes in DT Learning

$$H(V) = \sum_k P(v_k)\log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k)\log_2 P(v_k)$$

➢ Weight Update in NN Learning

- $h_w(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i}$
- $\mathbf{w}^* = \underset{\mathbf{w}}{\text{argmin}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j)$

The algorithm is as follows:

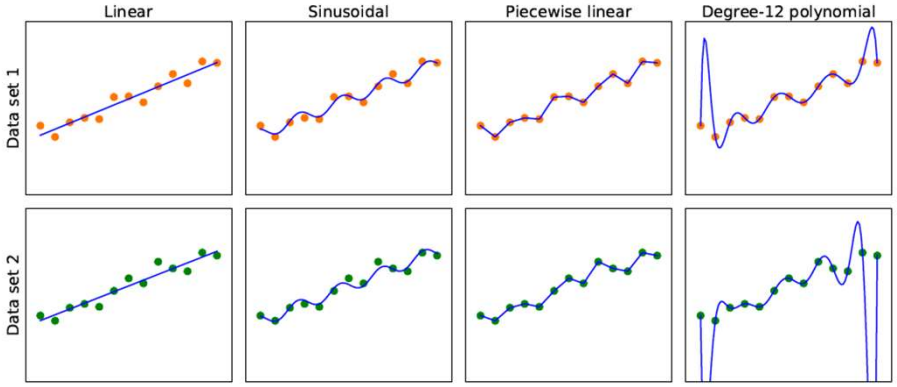
w ← any point in the parameter space
while not converged **do**
  for each w_i **in w do**
    $w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$

- $Loss(h_{\boldsymbol{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\boldsymbol{w}}(\mathrm{x}_j)) = \sum_{j=1}^{N} (y_j - h_{\boldsymbol{w}}(\mathrm{x}_j))^2 =$

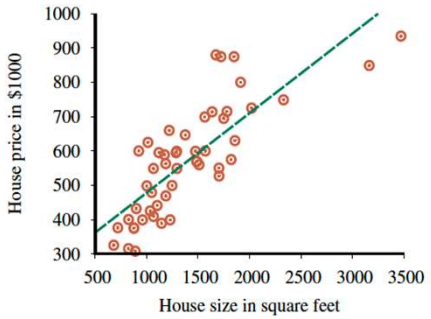
  $\sum_{j=1}^{N} (y_j - (w_1 \mathrm{x}_j + w_0))^2$

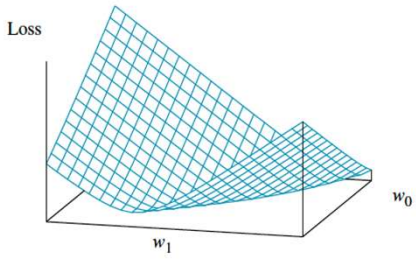
Lecture 15. Machine Learning

- **Forms of Learning**
  - Agent components, Representation and prior knowledge
- **Supervised Learning**
  - Classification, Regression
- **Learning Decision Trees**
  - Boolean decision tree, Entropy
  - Information gain
- **The Theory of Learning**
  - Probably approximately correct
- **Linear Regression and Classification**
  - Univariate, Multivariable linear regression, Gradient descent
- **Nonparametric Models**
  - Nearest-neighbor models, SVM
- **Ensemble Learning**
  - Bagging, Random forests, Stacking, Boosting



<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson





Outline (Lecture 15)

Stuart Russell & Peter Norvig (2021), Artificial Intelligence: A Modern Approach (4th Edition)

15.1 Forms of Learning

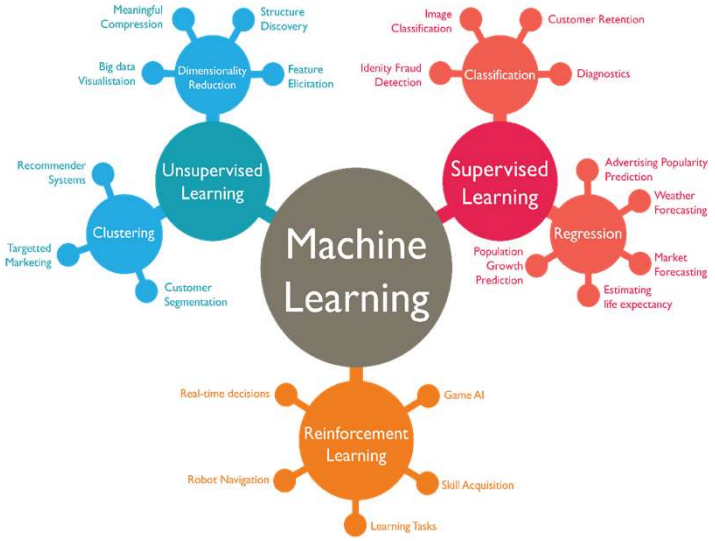
15.1 Forms of Learning (1/2)

- This chapter assumes little prior knowledge on the part of the agent.
  - Later we will consider transfer learning, in which knowledge from one domain is transferred to a new domain so that learning can proceed faster with less data.
- This chapter concentrates on the problems with
  - Inputs: factored representation – a vector of attribute values
  - Outputs: discrete values (classification) or continuous numerical values (regression)
    - Classification: the output is one of a finite set of values (sunny/cloudy/rainy or true/false)
    - Regression: When the output is a number (such as tomorrow's temperature, measured either as an integer or a real number)
- Inductive learning: learning a general function or rule from specific input-output pairs
- Analytical or deductive learning: going from a known general rule to a new rule that is logically entailed

15.1 Forms of Learning (2/2)

Feedback to learn from

- **Unsupervised learning:** learns patterns in the input without explicit feedback, such as clustering
- **Reinforcement learning:** learns from a series of reinforcements – rewards or punishments
- **Supervised learning:** observes some example input-output pairs and learn a function that maps from input to output
- **Semisupervised learning:** given a few labeled examples and a large collection of unlabeled examples



Google. (n.d.). Google search. Retrieved February 25, 2022

15.2 Supervised Learning

15.2 Supervised Learning (1/4)

Task of supervised learning:

- Given a training set of N example input-output pairs

$$(x_1, y_1), (x_2, y_2), \ldots, (x_N, y_N)$$

where each y_j was generated by an unknown function

$$y = f(x)$$

discover a function h that approximates the true function f.

Classification: discrete y
Regression: continuous y

- A **model** h is drawn from a model class $\mathcal{H}$. Or a function is drawn from a function class.
- A **consistent hypothesis** is an h s.t. $h(x_i) = y_i$

15.2 Supervised Learning (2/4)

Task of supervised learning:

- Supervised learning can be done by choosing the hypothesis h^* that is most probable given the data:

$$h^* = \arg\max_{h \in \mathcal{H}} P(h \mid data) \qquad P(h \mid data) = \frac{P(data \mid h)P(h)}{P(data)}$$

- By Bayes' rule, this is equivalent to

$$h^* = \arg\max_{h \in \mathcal{H}} P(data \mid h)P(h)$$ (MAP, Maximum A Posteriori Estimation)

- Assuming that all models are equiprobable

$$h^* = \arg\max_{h \in \mathcal{H}} P(data \mid h)$$ (MLE, Maximum Likelihood Estimation)

15.2 Supervised Learning (3/4)

Task of supervised learning:

- It shows that the function h that a learning algorithm discovers depends on the hypothesis space $\mathcal{H}$ it considers and on the training set it is given.

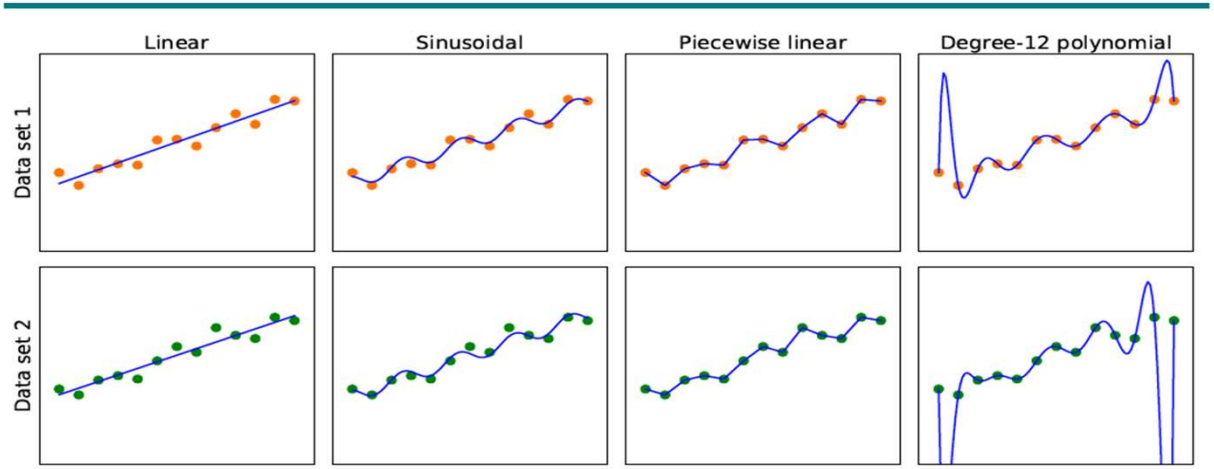


Figure 19.1 Finding hypotheses to fit data. **Top row**: four plots of best-fit functions from four different hypothesis spaces trained on data set 1. **Bottom row**: the same four functions, but trained on a slightly different data set (sampled from the same $f(x)$ function).

15.2 Supervised Learning (4/4)

Example problem: Restaurant waiting

- The problem of deciding whether to wait for a table at a restaurant based on the supervised learning

Example	Input Attributes										Output
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 = Yes$
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 = No$
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 = Yes$
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 = Yes$
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 = No$
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 = Yes$
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 = No$
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 = Yes$
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 = No$
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} = No$
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} = No$
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} = Yes$

Figure 19.2 Examples for the restaurant domain.

15.3 Learning Decision Trees

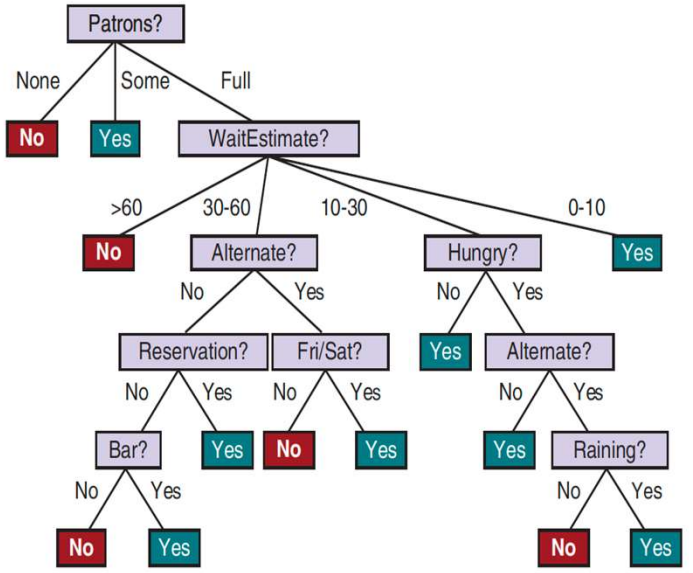
15.3 Learning Decision Trees (1/6)

Expressiveness of decision trees

- A Boolean decision tree is equivalent to a logical statement of the form:

  $Output \Leftrightarrow (Path_1 \vee Path_2 \vee \cdots)$,

- Where each $Path_i$ is a conjunction of the form $(A_m = v_x \wedge A_n = v_y \wedge \cdots)$ of attribute-value tests corresponding to a path from the root to a true leaf.



<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approa

Figure 19.3 A decision tree for deciding whether to wait for a table.

15.3 Learning Decision Trees (2/6)

- An algorithm for learning decision trees

function LEARN-DECISION-TREE(*examples*, *attributes*, *parent_examples*) **returns** a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree $\leftarrow$ a new decision tree with root test A
  for each value v of A **do**
    $exs \leftarrow \{e : e \in examples \text{ and } e.A = v\}$
    subtree $\leftarrow$ LEARN-DECISION-TREE(*exs*, *attributes* $- A$, *examples*)
    add a branch to *tree* with label $(A = v)$ and subtree *subtree*
  return *tree*

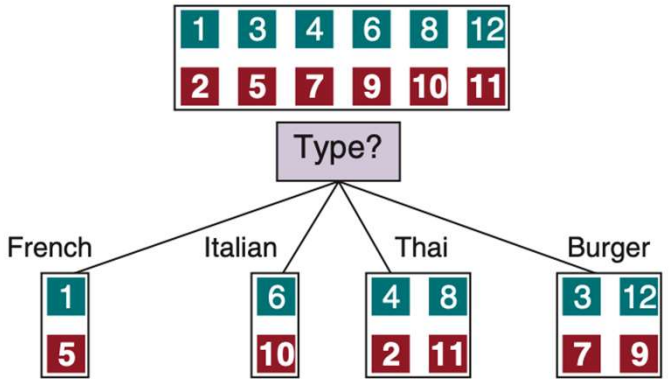


Figure 19.5 The decision tree learning algorithm. The function IMPORTANCE is described in Section 19.3.3. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.3 Learning Decision Trees (3/6)

- We can evaluate the performance of a learning algorithm with a learning curve, as shown below
- As the training set size grows, the accuracy increases (happy graphs)

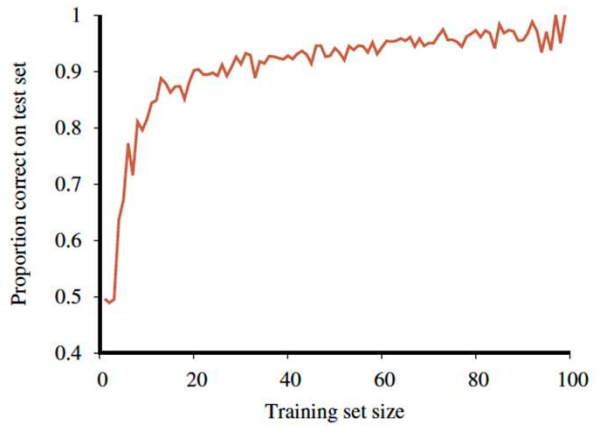


Figure 19.7 A learning curve for the decision tree learning algorithm on 100 randomly generated examples in the restaurant domain. Each data point is the average of 20 trials.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.3 Learning Decision Trees (4/6)

Choosing Attributes

- **Entropy** is a measure of the uncertainty of a random variable;
- Entropy of random variable V with values v_k, each with probability $P(v_k)$:

$$H(V) = \sum_k P(v_k)\log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k)\log_2 P(v_k)$$

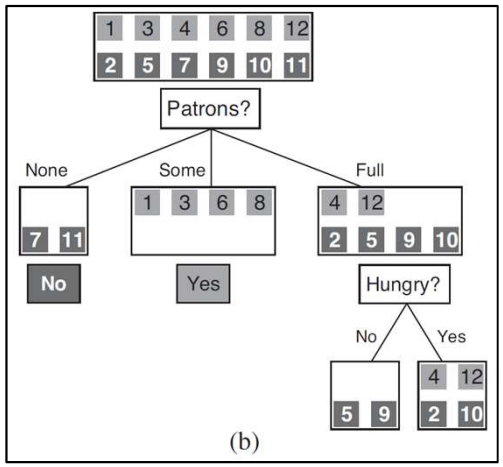
- Entropy of a Boolean random variable that is true with probability q:

$$B(q) = -(q \log_2 q + (1-q)\log_2(1-q))$$

- Entropy of goal attribute with p positive and n negative examples:

$$H(Goal) = B\left(\frac{p}{p+n}\right)$$

- An attribute A with d distinct values divides the training set E into subsets $E_1, \ldots, E_d$. Each subset E_k has p_k positive examples and n_k negative examples. To answer questions, we will need an additional $B(p_k/(p_k+n_k))$ bits of information.



<출처> Stuart J. Russell and Peter Norvig (2016). Artificial Intelligence: A Modern Approach (3rd Edition). Pearson

15.3 Learning Decision Trees (5/6)

Choosing Attributes

- Information gain from the attribute test on A is the expected reduction in entropy

$$Gain(A) = B\left(\frac{p}{p+n}\right) - Remainder(A)$$

$$Gain(Patrons)$$

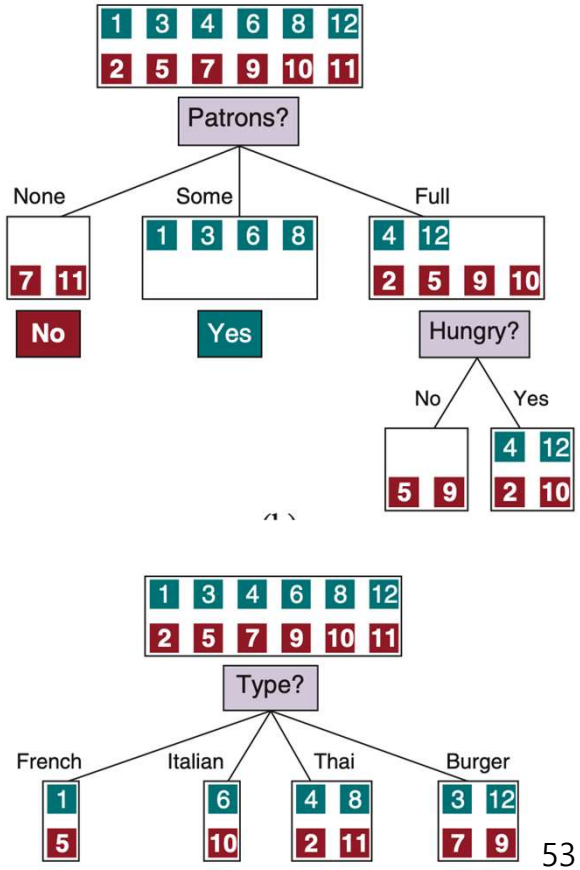
$$= 1 - \left[\frac{2}{12}B\left(\frac{0}{2}\right) + \left[\frac{4}{12}B\left(\frac{4}{4}\right)\right] + \left[\frac{6}{12}B\left(\frac{2}{6}\right)\right]\right]$$

$$\approx 0.541 \text{ bits}$$

$$Gain(Type)$$

$$= 1$$

$$- \left[\frac{2}{12}B\left(\frac{1}{2}\right) + \frac{2}{12}B\left(\frac{1}{2}\right) + \left[\frac{4}{12}B\left(\frac{2}{4}\right)\right] + \left[\frac{4}{12}B\left(\frac{2}{4}\right)\right]\right]$$

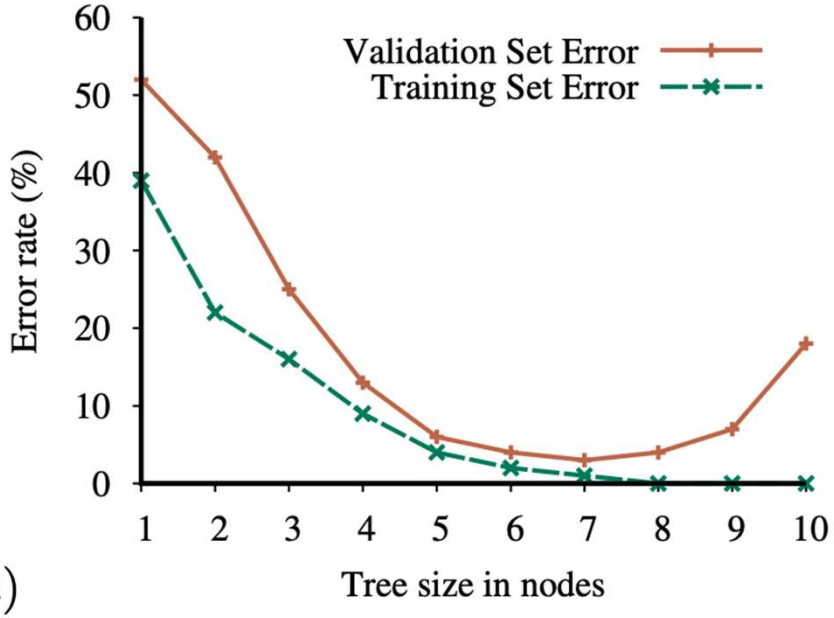


15.3 Learning Decision Trees (6/6)

Overfitting and Regularization

- **Overfitting** becomes more likely as the hypothesis space and the number of input attributes grows, and less likely as we increase the number of training examples
- Decision tree pruning combats overfitting
- **Regularization:**

$$Cost(h) = EmpLoss(h) + \lambda Complexity(h)$$
$$\hat{h}^* = \operatorname*{argmin}_{h \in \mathcal{H}} Cost(h) .$$



15.4 The Theory of Learning




15.4 The Theory of Learning (1/3)

- Learning curves are useful, but they are specific to a particular learning algorithm on a particular problem.
- Are there some more general principles governing the number of examples needed?
  - Computational learning theory

Probably approximately correct (PAC)

- Any hypothesis that is consistent with a sufficiently large set of training examples is unlikely to be seriously wrong: that is, it must be *probably approximately correct* (PAC).
- Any learning algorithm that returns hypotheses that are probably approximately correct is called a PAC learning algorithm;
  - We can use this approach to provide bounds on the PAC learning performance of various learning algorithms.

15.4 The Theory of Learning (2/3)

Probably approximately correct (PAC)

- The simplest PAC theorems deal with Boolean functions. The error rate of a hypothesis *h* is defined formally as the expected generalization error drawn from the stationary distribution:

$$error(h) = GenLoss_{L_{0/1}}(h) = \sum_{x,y} L_{0/1}\big(y, h(x)\big)P(x,y)$$

- A hypothesis h is called *approximately correct* if $error(h) \leq e$, where e is a small constant.
- Applying PAC learning to a new hypothesis space: decision lists (= a conjunction of literals).

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.4 The Theory of Learning (3/3)

Probably approximately correct (PAC)

- Learning decision lists

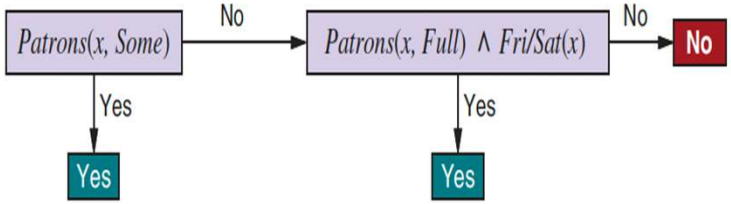


Figure 19.10 A decision list for the restaurant problem.

Figure 19.12 Learning curve for DECISION-LIST-LEARNING algorithm on the restaurant data. The curve for LEARN-DECISION-TREE is shown for comparison; decision trees do slightly better on this particular problem.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.5 Linear Regression and Classification

15.5 Linear Regression and Classification (1/10)

Univariate linear regression

- A univariate linear function (a straight line) with input x and output y has the form

$$y = w_1 x + w_0$$

- We'll define w to be the vector $< w_0, w_1 >$, and define the linear function with those weights as:

$$h_{\mathbf{w}}(x) = w_1 x + w_0$$

- The task of finding the $h_{\mathbf{w}}$ that best fits these data is called **linear regression.**

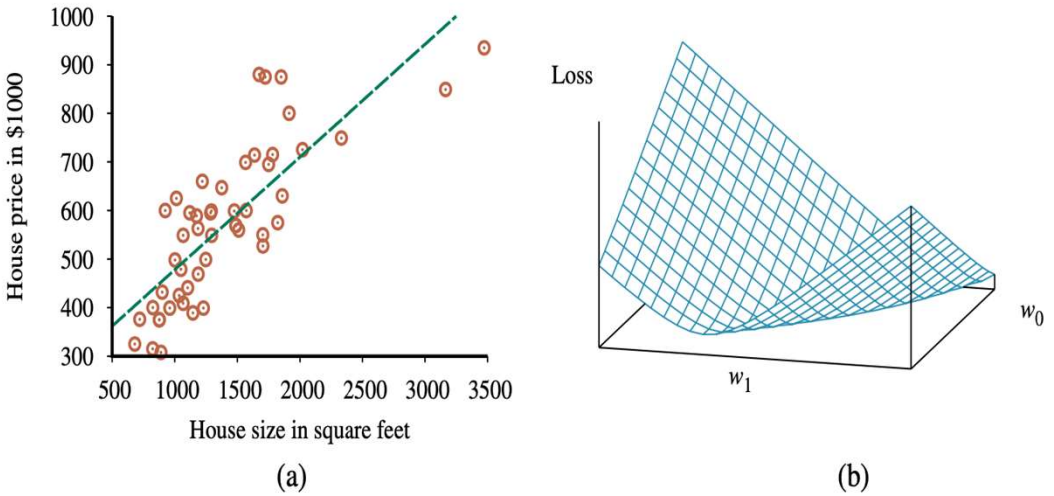


Figure 19.13 (a) Data points of price versus floor space of houses for sale in Berkeley, CA, in July 2009, along with the linear function hypothesis that minimizes squared-error loss: $y = 0.232x + 246$. (b) Plot of the loss function $\sum_j (y_j - w_1 x_j + w_0)^2$ for various values of w_0, w_1. Note that the loss function is convex, with a single global minimum.

15.5 Linear Regression and Classification (2/10)

Univariate linear regression

- To fit a line to the data, all we have to do is find the values of the weights $< w_0, w_1 >$ that minimize the empirical loss.
- It is traditional to use the squared-error loss function, L_2:

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 .$$

We would like to find $\mathbf{w}^* = \text{argmin}_{\mathbf{w}} Loss(h_{\mathbf{w}})$. The sum $\sum_{j=1}^{N}(y_j - (w_1 x_j + w_0))^2$ is minimized when its partial derivatives with respect to w_0 and w_1 are zero:

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 . \qquad (19.2)$$

These equations have a unique solution:

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N . \qquad (19.3)$$

15.5 Linear Regression and Classification (3/10)

Univariate linear regression

- (Example) The solution is $w_1 = 0.232$, $w_0 = 246$, and the line with those weights is shown as a dashed line in the figure.

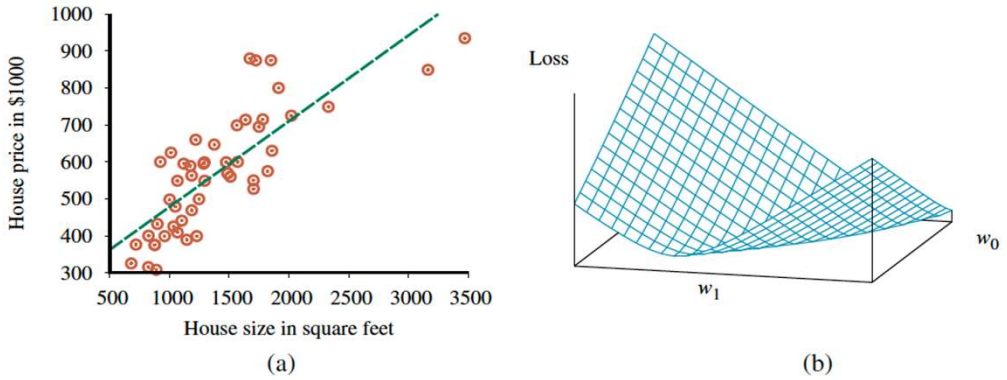


Figure 19.13 (a) Data points of price versus floor space of houses for sale in Berkeley, CA, in July 2009, along with the linear function hypothesis that minimizes squared-error loss: $y = 0.232x + 246$. (b) Plot of the loss function $\sum_j (y_j - w_1 x_j + w_0)^2$ for various values of w_0, w_1. Note that the loss function is convex, with a single global minimum.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.5 Linear Regression and Classification (4/10)

Gradient descent

- We choose any starting point in weight space—here, a point in the $< w_0, w_1 >$ plane—and then compute an estimate of the gradient and move a small amount in the steepest downhill direction, repeating until we converge on a point in weight space with (local) minimum loss.

The algorithm is as follows:

w $\leftarrow$ any point in the parameter space
while not converged **do**
  for each w_i **in w do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

The parameter α is the step size (learning rate)

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 = 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))$$

$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) .$$

15.5 Linear Regression and Classification (5/10)

Gradient descent (cont'd)

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2 = 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))$$

$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) . \qquad (19.5)$$

Applying this to both w_0 and w_1 we get:

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) ; \qquad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x .$$

Plugging this into Equation (19.4), and folding the 2 into the unspecified learning rate α, we get the following learning rule for the weights:

$$w_0 \leftarrow w_0 + \alpha (y - h_{\mathbf{w}}(x)) ; \quad w_1 \leftarrow w_1 + \alpha (y - h_{\mathbf{w}}(x)) \times x .$$

15.5 Linear Regression and Classification (6/10)

Multivariable linear regression

- Each example x_j is an *n-element* vector. Our hypothesis space is the set of functions of the form:

$$h_{\mathbf{w}}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i}$$

- Then h is simply the dot product of the weights and the input vector:

$$h_w(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i}$$

- The best vector of weights, w^*, minimizes squared-error loss L_2 over the examples:

$$\mathbf{w}^* = \underset{\mathbf{w}}{\operatorname{argmin}} \sum_j L_2(y_j, \mathbf{w} \cdot \mathbf{x}_j)$$

$$w_i \leftarrow w_i + \alpha \sum_j \left(y_j - h_{\mathbf{w}(\mathbf{x}_j)}\right) \mathrm{x}\, x_{j,i}$$

15.5 Linear Regression and Classification (7/10)

Overfitting and regularization

- With multivariable linear regression in high-dimensional spaces, it is possible that some dimension that is actually irrelevant appears by chance to be useful, resulting in overfitting
- Thus, it is common to use *regularization* on multivariable linear functions to avoid overfitting
- For linear functions the complexity can be specified as a function of the weights. We can consider a family of regularization functions:

$$Complexity(h_{\mathbf{w}}) = L_q(\mathbf{w}) = \sum_i |w_i|^q$$

15.5 Linear Regression and Classification (8/10)

Regularization

- L_1 regularization leads to weights of zero, while L_2 regularization does not.

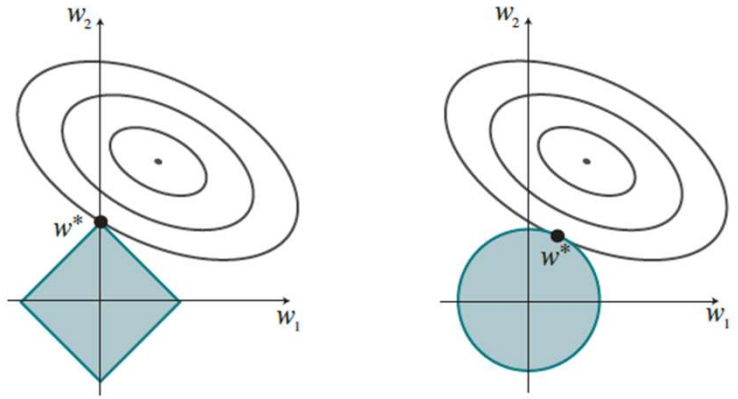


Figure 19.14 Why L_1 regularization tends to produce a sparse model. Left: With L_1 regularization (box), the minimal achievable loss (concentric contours) often occurs on an axis, meaning a weight of zero. Right: With L_2 regularization (circle), the minimal loss is likely to occur anywhere on the circle, giving no preference to zero weights.

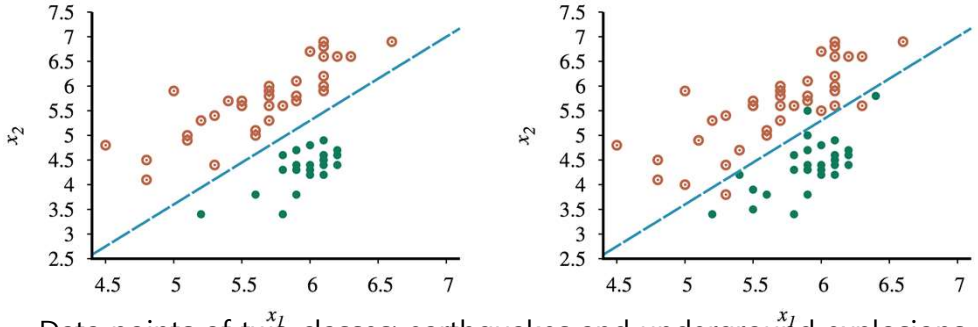
<출처> Stuart J. Russell and Peter Norvig (2021) . Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.5 Linear Regression and Classification (9/10)

Linear classifiers with a hard threshold

- Linear functions can be used to do classification as well as regression.

  $h_w(\mathbf{x}) = 1$ if $\mathbf{w} \cdot \mathbf{x} \geq 0$ and 0 otherwise



Data points of two classes: earthquakes and underground explosions

- Alternatively, we can think of *h* as the result of passing the linear function $\mathbf{w} \cdot \mathbf{x}$ through a threshold function:

  $h_w(\mathbf{x}) = Threshold(\mathbf{w} \cdot \mathbf{x})$
  where $Threshold(z) = 1$ if $z \geq 0$ and 0 otherwise

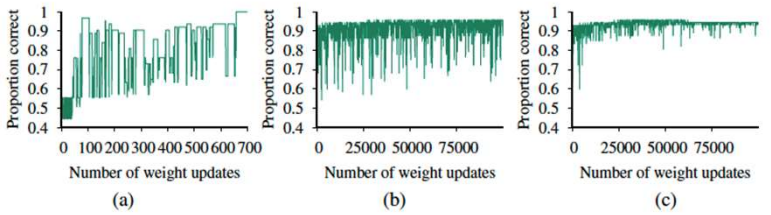


Figure 19.16 (a) Plot of total training-set accuracy vs. number of iterations through the training set for the perceptron learning rule, given the earthquake/explosion data in Figure 19.15(a). (b) The same plot for the noisy, nonseparable data in Figure 19.15(b); note the change in scale of the *x*-axis. (c) The same plot as in (b), with a learning rate schedule $\alpha(t) = 1000/(1000 + t)$.

15.5 Linear Regression and Classification (10/10)

Linear classifiers with logistic regression

$$h_w(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + \exp(-\mathbf{w} \cdot \mathbf{x})}$$

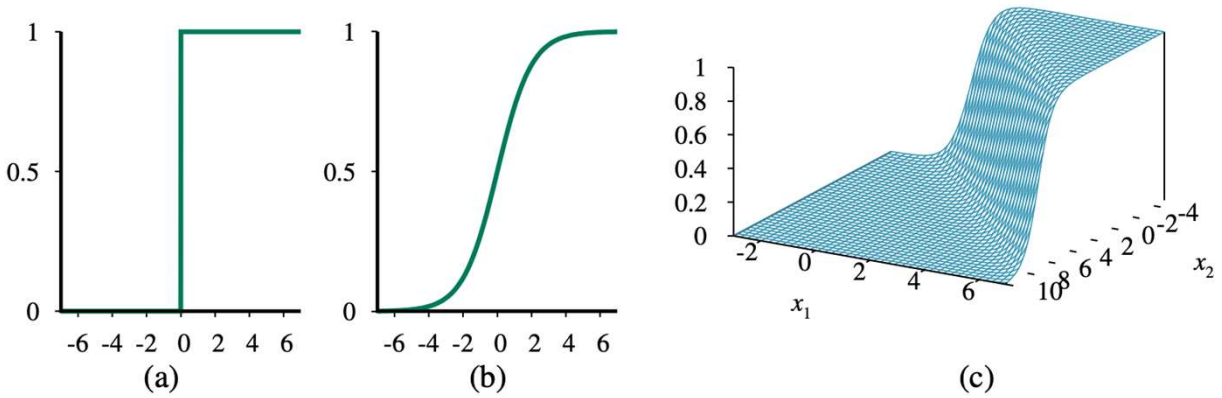


Figure 19.17 (a) The hard threshold function $Threshold(z)$ with 0/1 output. Note that the function is nondifferentiable at $z=0$. (b) The logistic function, $Logistic(z) = \frac{1}{1+e^{-z}}$, also known as the sigmoid function. (c) Plot of a logistic regression hypothesis $h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x})$ for the data shown in Figure 19.15(b).

15.6 Nonparametric Models

15.6 Nonparametric Models (1/5)

Nonparametric models

- A nonparametric model is one that cannot be characterized by a bounded set of parameters
  - The problem with this method is that it does not generalize well

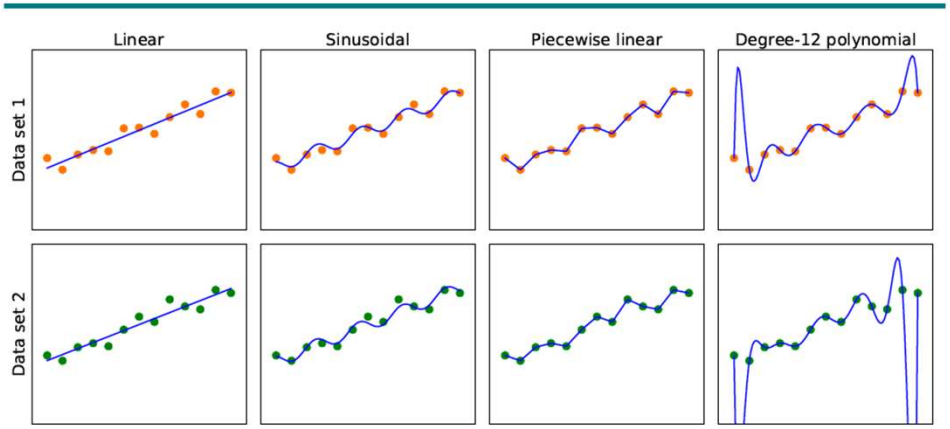


Figure 19.1 Finding hypotheses to fit data. **Top row**: four plots of best-fit functions from four different hypothesis spaces trained on data set 1. **Bottom row**: the same four functions, but trained on a slightly different data set (sampled from the same $f(x)$ function).

<출처> Stuart J. Russell and Peter Norvig (2021) . Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.6 Nonparametric Models (2/5)

Nearest-neighbor models

- Given a query $\mathbf{x}_q$, instead of finding an example that is equal to $\mathbf{x}_q$, find the k examples that are nearest to $\mathbf{x}_q$. This is called *k*-nearest-neighbors lookup.
- Distances are measured with a *Minkowski distance* or L^p norm, defined as:

$$L^p(\mathbf{x}_j, \mathbf{x}_q) = (\sum_i |\mathbf{x}_{j,i} - \mathbf{x}_{q,i}|^p)^{1/p}$$

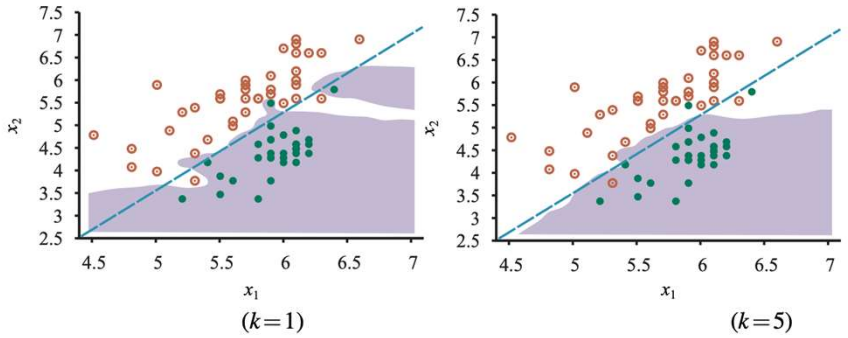


Figure 19.19 (a) A *k*-nearest-neighbors model showing the extent of the explosion class for the data in Figure 19.15, with $k=1$. Overfitting is apparent. (b) With $k=5$, the overfitting problem goes away for this data set.

- In low-dimensional spaces with plenty of data, nearest neighbors works very well: we are likely to have enough nearby data points to get a good answer. But as the number of dimensions rises we encounter a problem (*curse of dimensionality*).

15.6 Nonparametric Models (3/5)

Nonparametric regression

- *K*-nearest-neighbors regression improves on connect-the-dots.

- Locally weighted regression (Figure 19.20(d)) gives us the advantages of nearest neighbors, without the discontinuities.

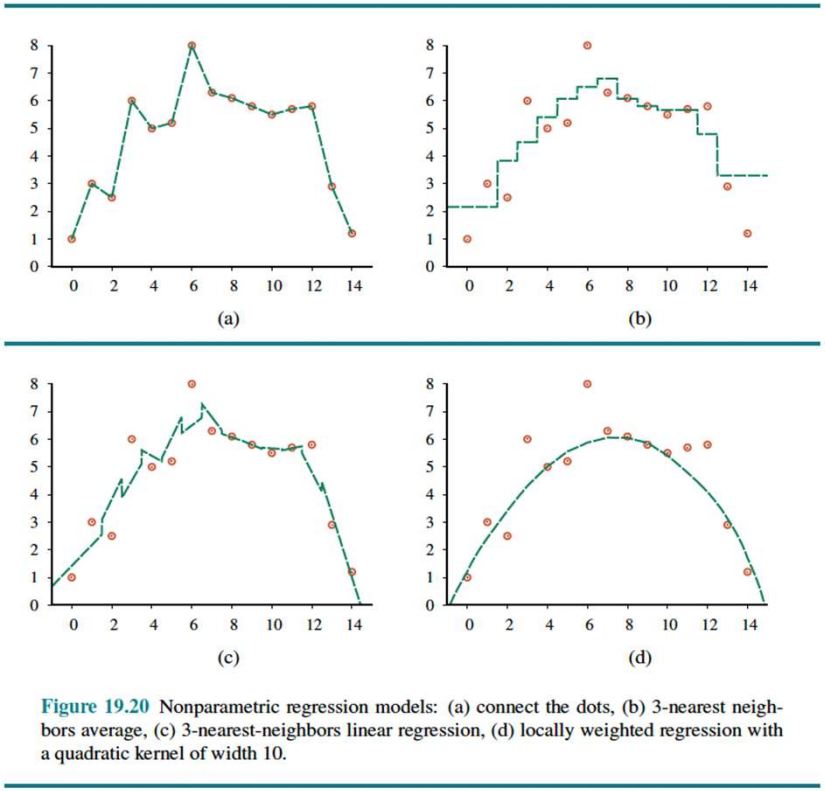


Figure 19.20 Nonparametric regression models: (a) connect the dots, (b) 3-nearest neighbors average, (c) 3-nearest-neighbors linear regression, (d) locally weighted regression with a quadratic kernel of width 10.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.6 Nonparametric Models (4/5)

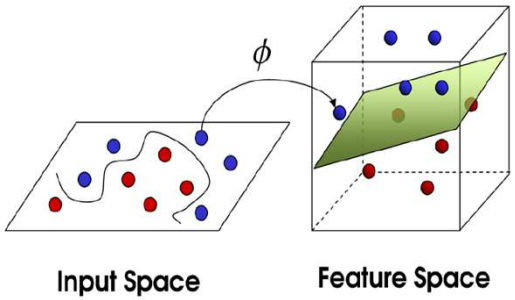
Support vector machines

- SVMs construct a maximum margin separator—a decision boundary with the largest possible distance to example points. This helps them generalize well.
- SVMs create a linear separating hyperplane, but they have the ability to embed the data into a higher-dimensional space, using the so-called kernel trick.

$$\underset{\alpha}{\text{argmax}} \sum_j \alpha_j - \frac{1}{2} \sum_{j,k} \alpha_j \alpha_k y_j y_k (\mathbf{x}_j \cdot \mathbf{x}_k)$$

subject to the constraints $\alpha_j \geq 0$ and $\sum_j \alpha_j y_j = 0$.

$$h(\mathbf{x}) = \text{sign}\left(\sum_j \alpha_j y_j (\mathbf{x} \cdot \mathbf{x}_j) - b \right)$$



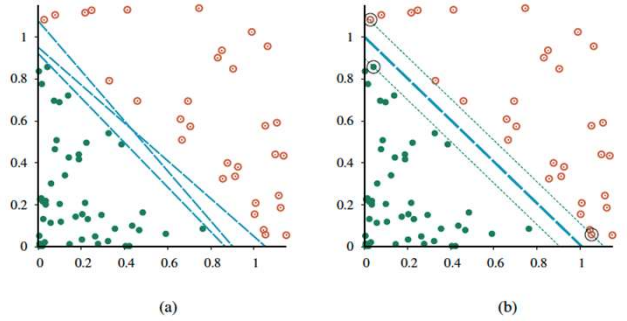


Figure 19.21 Support vector machine classification: (a) Two classes of points (orange open and green filled circles) and three candidate linear separators. (b) The maximum margin separator (heavy line), is at the midpoint of the **margin** (area between dashed lines). The **support vectors** (points with large black circles) are the examples closest to the separator; here there are three.

15.6 Nonparametric Models (5/5)

The kernel trick

- The kernel method implicitly transform the input data into a high-dimensional space where a linear separator may exist, even if the original data are nonseparable.
- The dot product is replaced by a kernel function and we have a kernelized version of the algorithm (kernelization)

$$F(\mathbf{x}_j) \cdot F(\mathbf{x}_k) = (\mathbf{x}_j \cdot \mathbf{x}_k)^2$$

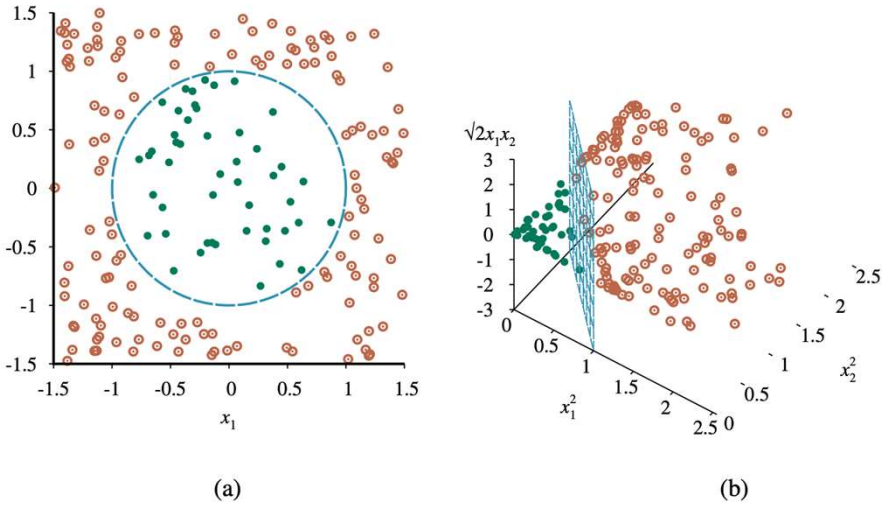


Figure 19.22 (a) A two-dimensional training set with positive examples as green filled circles and negative examples as orange open circles. The true decision boundary, $x_1^2 + x_2^2 \leq 1$, is also shown. (b) The same data after mapping into a three-dimensional input space $(x_1^2, x_2^2, \sqrt{2}x_1x_2)$. The circular decision boundary in (a) becomes a linear decision boundary in three dimensions. Figure 19.21(b) gives a closeup of the separator in (b).

15.7 Ensemble Learning

15.7 Ensemble Learning (1/7)

What is ensemble learning?

- The idea of ensemble learning is to select a collection, or ensemble, of hypotheses, $h_1, h_2, \ldots, h_n$, and combine their predictions by averaging, voting, or by another level of machine learning. We call the individual hypotheses base models and their combination **an ensemble model**.

Two reasons of ensemble learning

1) The first is to reduce *bias*.
2) The second reason is to reduce *variance*.

15.7 Ensemble Learning (2/7)

Ensemble of linear classifiers

- An ensemble of three linear classifiers (Figure 19.23) can represent a triangular region that could not be represented by a single linear classifier.
- An ensemble of *n* linear classifiers allows more functions to be realizable.

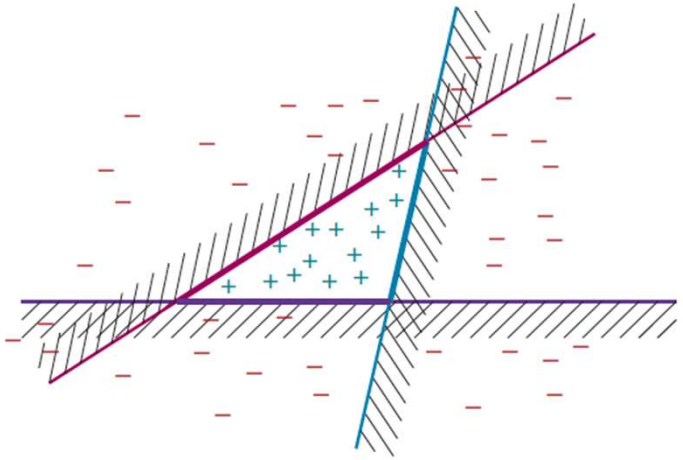


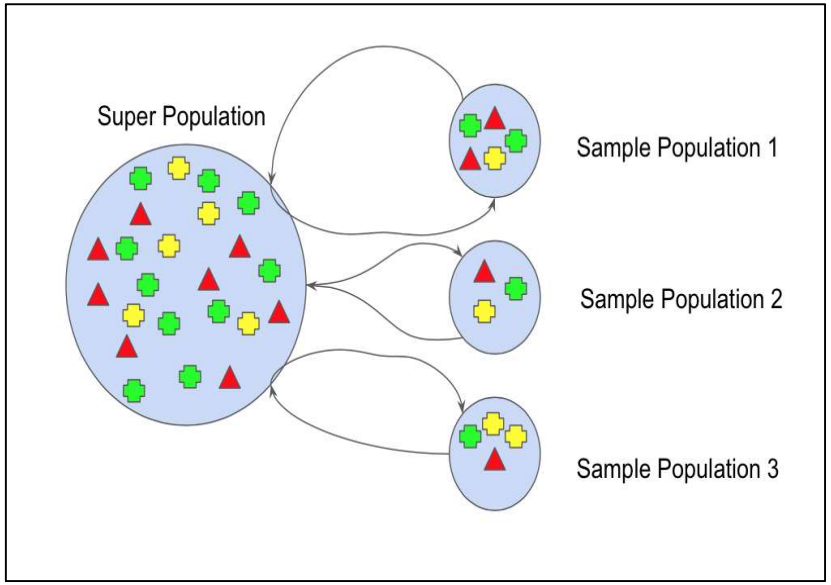
Figure 19.23 Illustration of the increased expressive power obtained by ensemble learning. We take three linear threshold hypotheses, each of which classifies positively on the unshaded side, and classify as positive any example classified positively by all three. The resulting triangular region is a hypothesis not expressible in the original hypothesis space.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.7 Ensemble Learning (3/7)

Bagging

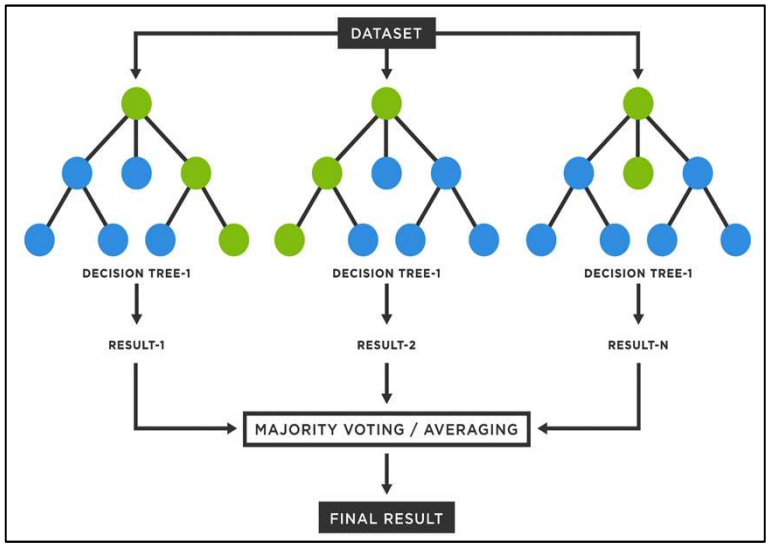
- In bagging, we generate *K* distinct training sets by sampling with replacement from the original training set.
- Bagging tends to reduce variance and is a standard approach when there is limited data or when the base model is seen to be overfitting.



15.7 Ensemble Learning (4/7)

Random forests

- The random forest model is a form of decision tree bagging in which we take extra steps to make the ensemble of *K* trees more diverse, to reduce variance.
- Random forests can be used for classification or regression.
- The key idea is to randomly vary the attribute choices (rather than the training examples).



15.7 Ensemble Learning (5/7)

Stacking

- Whereas **bagging** combines multiple base models of the *same model class* trained on *different data,*
- **Stacking** (or stacked generalization) combines multiple base models from *different model classes* trained on the *same data.*

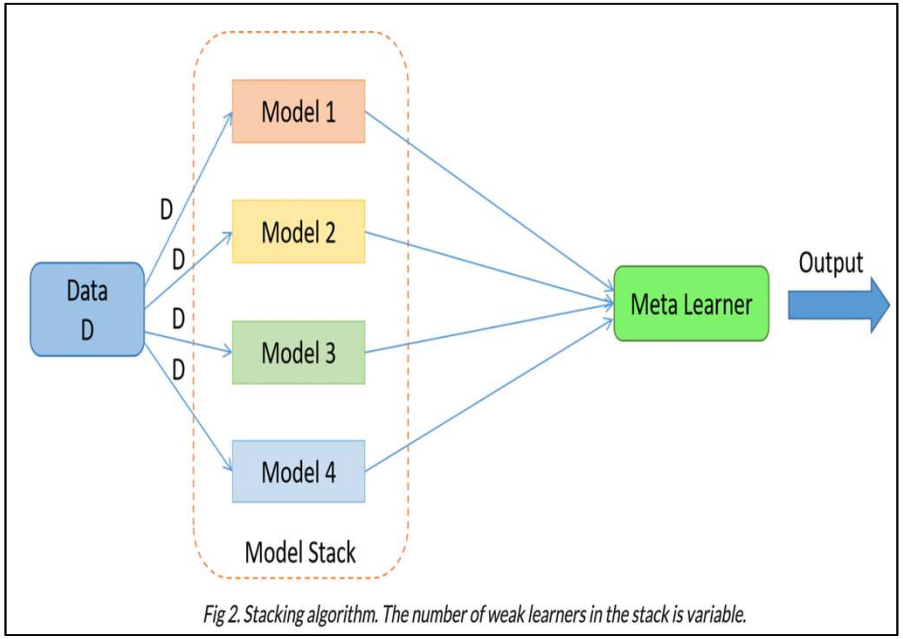


Fig 2. Stacking algorithm. The number of weak learners in the stack is variable.

15.7 Ensemble Learning (6/7)

Boosting

- The most popular ensemble method is called boosting.
  - Weighted training set, in which each example has an associated weight $W_j \geq 0$ that describes how much the example should count during training.

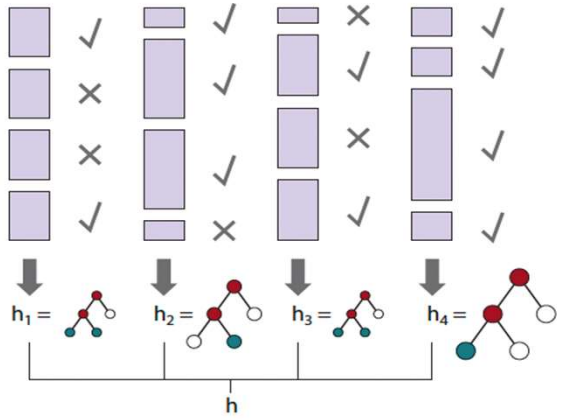


Figure 19.24 How the boosting algorithm works. Each shaded rectangle corresponds to an example; the height of the rectangle corresponds to the weight. The checks and crosses indicate whether the example was classified correctly by the current hypothesis. The size of the decision tree indicates the weight of that hypothesis in the final ensemble.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

15.7 Ensemble Learning (7/7)

Boosting

- An interesting thing happens as the ensemble size K increases.
- Figure 19.26(b) shows the training set performance (on 100 examples) as a function of K

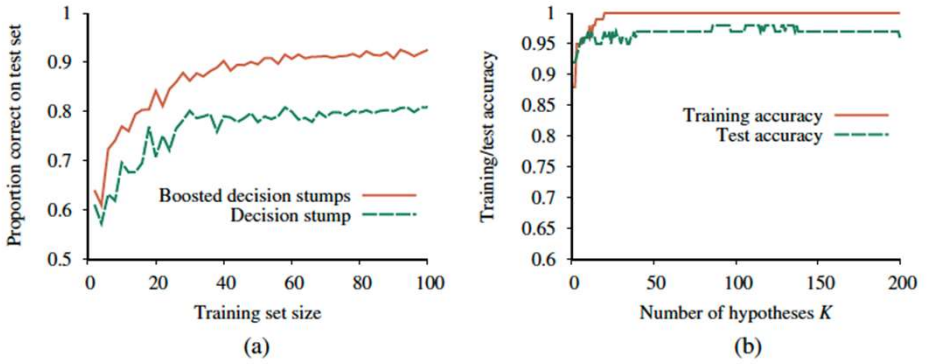


Figure 19.26 (a) Graph showing the performance of boosted decision stumps with $K = 5$ versus unboosted decision stumps on the restaurant data. (b) The proportion correct on the training set and the test set as a function of K, the number of hypotheses in the ensemble. Notice that the test set accuracy improves slightly even after the training accuracy reaches 1, i.e., after the ensemble fits the data exactly.

<출처> Stuart J. Russell and Peter Norvig (2021). Artificial Intelligence: A Modern Approach (4th Edition). Pearson

Summary

1. If the available feedback provides the correct answer for example inputs, then the learning problem is called **supervised learning**. The task is to learn a function $y = h(x)$.
2. Learning a discrete-valued function is called classification; learning a continuous function is called regression.
3. Inductive learning involves finding a hypothesis that agrees well with the examples.
4. **Decision trees** can represent all Boolean functions. The Information-gain heuristic provides an efficient method for finding a simple, consistent decision tree.
5. The performance of a learning algorithm is measured by the learning curve, which shows the prediction accuracy on the test set as a function of the **training-set size**. When there are multiple models to choose from, cross-validation can be used to select a model that will generalize well.
6. Sometimes not all errors are equal. A loss function tells us how bad each error is; the goal is then to minimize loss over a validation set.
7. **Computational learning theory** analyzes the sample complexity and computational complexity of inductive learning. There is a tradeoff between the expressiveness of the hypothesis space and the ease of learning.

Summary

1. Linear regression is a widely used model. The optimal parameters of a linear regression model can be calculated exactly, or can be found by gradient descent search, which is a technique that can be applied to models that do not have a closed-form solution.
2. A linear classifier with a hard threshold—also known as a perceptron—can be trained by a simple weight update rule to fit data that are linearly separable. In other cases, the rule fails to converge.
3. Logistic regression replaces the perceptron's hard threshold with a soft threshold defined by a logistic function. Gradient descent works well even for noisy data that are not linearly separable.
4. Nonparametric models use all the data to make each prediction, rather than trying to summarize the data with a few parameters. Examples include nearest neighbors and locally weighted regression.
5. **Support vector machines** find linear separators with maximum margin to improve the generalization performance of the classifier. **Kernel methods** implicitly transform the input data into a high-dimensional space where a linear separator may exist, even if the original data are nonseparable.
6. **Ensemble methods** such as bagging and boosting often perform better than individual methods. In online learning we can aggregate the opinions of experts to come arbitrarily close to the best expert's performance, even when the distribution of the data are constantly shifting.