Artificial Intelligence

Hai Thi Tuyet Nguyen

Outline

CHAPTER 1: INTRODUCTION (CHAPTER 1)

CHAPTER 2: INTELLIGENT AGENTS (CHAPTER 2)

CHAPTER 3: SOLVING PROBLEMS BY SEARCHING (CHAPTER 3)

CHAPTER 4: INFORMED SEARCH (CHAPTER 3)

CHAPTER 5: LOGICAL AGENT (CHAPTER 7)

CHAPTER 6: FIRST-ORDER LOGIC (CHAPTER 8, 9)

CHAPTER 7: QUANTIFYING UNCERTAINTY(CHAPTER 13)

CHAPTER 8: PROBABILISTIC REASONING (CHAPTER 14)

CHAPTER 9: LEARNING FROM EXAMPLES (CHAPTER 18)

CHAPTER 9: LEARNING FROM EXAMPLES

9.1 Forms Of Learning

9.1 Forms Of Learning

- Any component of an agent can be improved by learning from data.
- The improvements depend on four major factors:
  - Which component is to be improved.
    Information about the way the world evolves and about the results of possible actions the agent can take.
    E.g., By trying actions and observing the results of braking hard on a wet road, the driver agent can learn the effects of its actions
  - What prior knowledge the agent already has + What representation is used for the data and the component.
    - propositional and first-order logical sentences for the components in a logical agent
    - a factored representation - a vector of attribute values - and outputs that can be either a continuous numerical value or a discrete value.
  - What feedback is available to learn from: it depends on three types of feedback

9.1 Forms Of Learning

- The improvements depend on four major factors:
  - What feedback is available to learn from: it depends on three types of feedback
    - In **unsupervised learning** the agent learns patterns in the input even though no explicit feedback is supplied.
      - The most common task is clustering
      - E.g.: a taxi agent might gradually develop a concept of “good traffic days” and “bad traffic days” without ever being given labeled examples of each by a teacher.
    - In **reinforcement learning** the agent learns from a series of reinforcements - rewards or punishments.
      - E.g.: the points for a win at the end of a chess game tells the agent it did something right.
    - In **supervised learning** the agent observes some example input–output pairs and learns a function that maps from input to output.

9.2 Supervised Learning

9.2 Supervised Learning

- The task of supervised learning:
  Given a training set of N example input–output pairs
  $(x_1,y_1), (x_2,y_2),...(x_N,y_N)$,
  where each y_j was generated by an unknown function $y = f(x)$,
  discover a function h (as hypothesis) that approximates the true function f .

- To measure the accuracy of a hypothesis h, we give it a test set of examples that are distinct from the training set.
- A hypothesis generalizes well if it correctly predicts the value of y for novel examples.
- When the output y is one of a finite set of values (such as sunny, cloudy or rainy), the learning problem is called classification.
- When y is a number, the learning problem is called regression.

9.3 Learning Decision Trees

9.3 The decision tree representation

A decision tree represents a function that takes as input a vector of attribute values and returns a "decision"—a single output value.

A decision tree reaches its decision by performing a sequence of tests.

- Each internal node: a test of the values of the input attribute, A_i
- The branches from the node: the values of the attribute, $A_i = v_{ik}$.
- Each leaf node: a value to be returned by the function.

Expressiveness of decision trees: $\text{Goal} \Leftrightarrow (\text{Path}_1 \vee \text{Path}_2 \vee \cdots)$

9.3 Inducing decision trees from examples

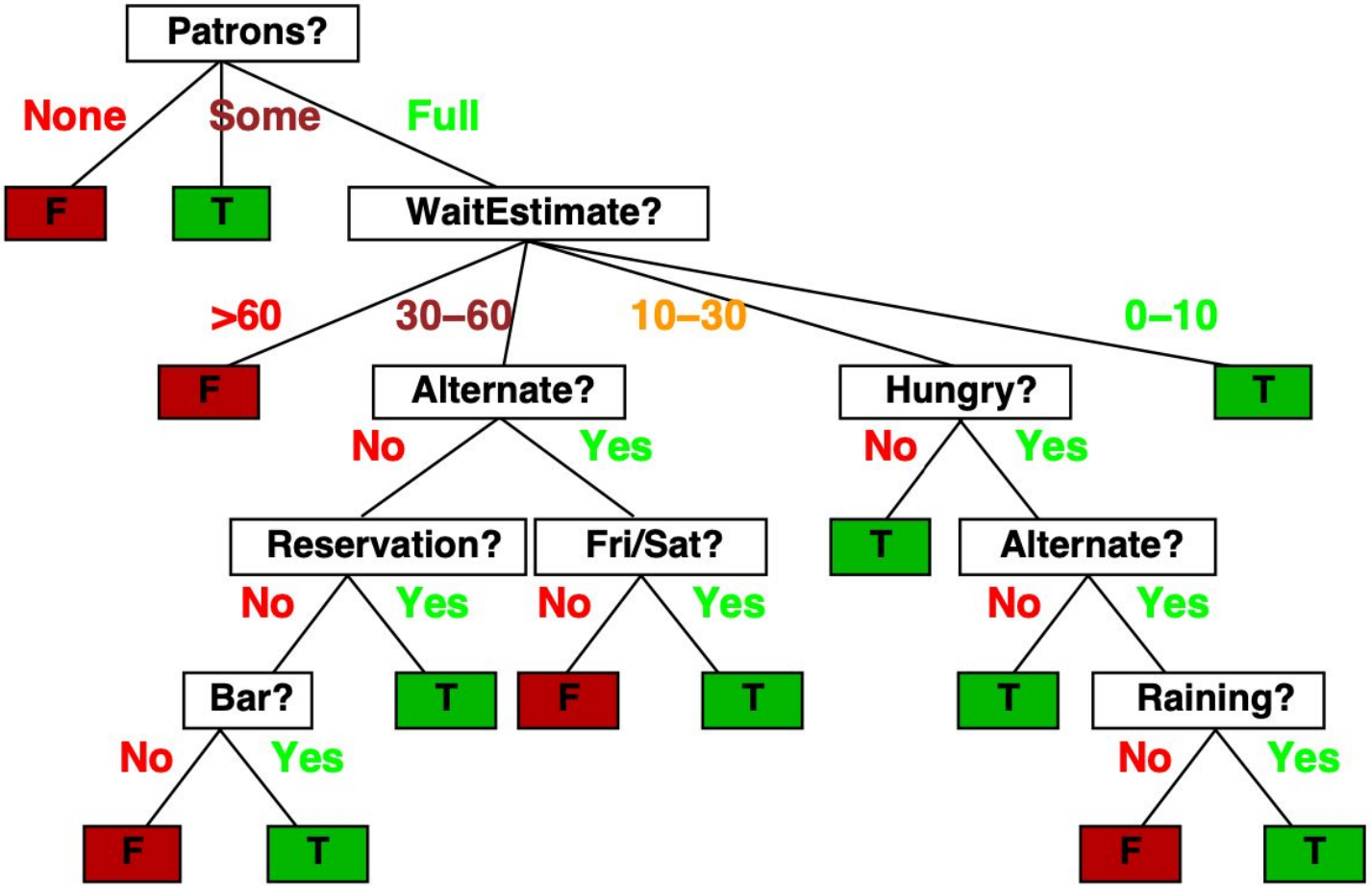
Build a decision tree to decide whether to wait for a table at a restaurant with the following attributes:

1. Alternate: whether there is a suitable alternative restaurant nearby.
2. Bar: whether the restaurant has a comfortable bar area to wait in.
3. Fri/Sat: true on Fridays and Saturdays.
4. Hungry: whether we are hungry.
5. Patrons: how many people are in the restaurant (values are None, Some, and Full).
6. Price: the restaurant's price range ($, $$, $$$).
7. Raining: whether it is raining outside.
8. Reservation: whether we made a reservation.
9. Type: the kind of restaurant (French, Italian, Thai, or burger).
10. WaitEstimate: the wait estimated by the host (0–10 minutes, 10–30, 30–60, or >60).

9.3 Attribute-based representations

Example	Attributes										Target
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
X_1	T	F	F	T	Some	$$$	F	T	French	0–10	T
X_2	T	F	F	T	Full	$	F	F	Thai	30–60	F
X_3	F	T	F	F	Some	$	F	F	Burger	0–10	T
X_4	T	F	T	T	Full	$	F	F	Thai	10–30	T
X_5	T	F	T	F	Full	$$$	F	T	French	>60	F
X_6	F	T	F	T	Some	$$	T	T	Italian	0–10	T
X_7	F	T	F	F	None	$	T	F	Burger	0–10	F
X_8	F	F	F	T	Some	$$	T	T	Thai	0–10	T
X_9	F	T	T	F	Full	$	T	F	Burger	>60	F
X_{10}	T	T	T	T	Full	$$$	F	T	Italian	10–30	F
X_{11}	F	F	F	F	None	$	F	F	Thai	0–10	F
X_{12}	T	T	T	T	Full	$	F	F	Burger	30–60	T

9.3 Inducing decision trees from examples



9.3 Decision tree learning

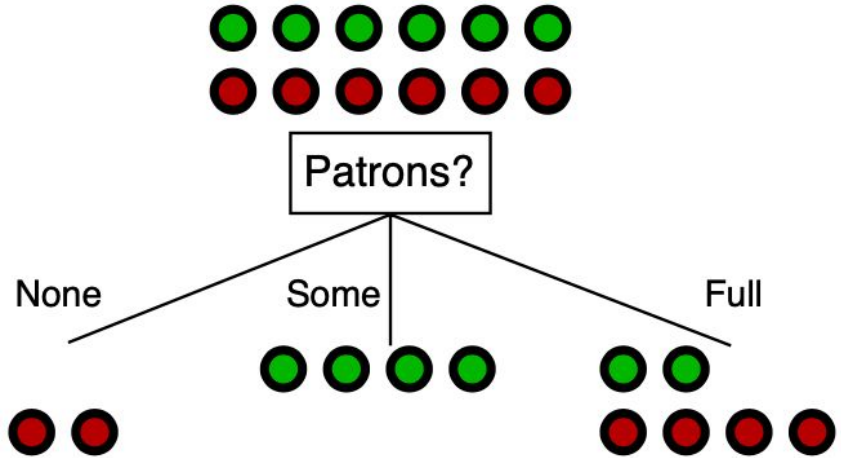
- Aim: find a small tree consistent with the training examples
- Idea: (recursively) choose "most significant" attribute as root of (sub)tree
- PLURALITY-VALUE selects the most common output value among a set of examples.

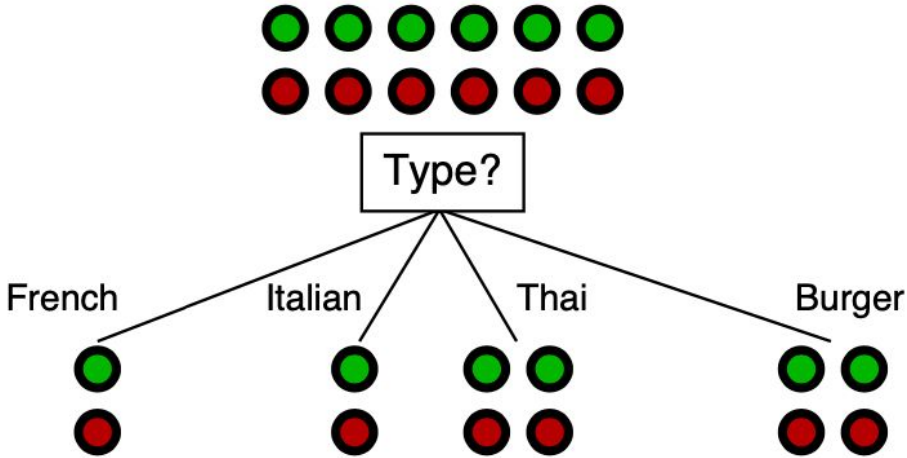
function DECISION-TREE-LEARNING($examples$, $attributes$, $parent\_examples$) **returns** a tree

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_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
    add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
  return $tree$

9.3 Choosing attribute tests

Idea: a good attribute splits the examples into subsets that are “all positive” or “all negative”
Patrons? is a better choice—gives information about the classification





9.3 Choosing attribute tests

- Entropy is a measure of the uncertainty of a random variable; information gain corresponds to a reduction in entropy.
- Scale: 1 bit = an answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$
- The entropy of the prior or information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is $H(\langle P_1,\ldots,P_n \rangle)=\Sigma -P_i \log_2 P_i$

9.3 Choosing attribute tests

An attribute splits the examples E into subsets E_i, each E_i has p_i positive and n_i negative examples
⇒ $H(\langle p_i/(p_i+n_i), n_i/(p_i+n_i)\rangle)$ bits needed to classify a new example
⇒ expected number of bits per example over all branches:

$$\sum_i \frac{p_i + n_i}{p + n} H(\langle p_i/(p_i + n_i), n_i/(p_i + n_i)\rangle)$$

For Patrons?, this is 0.459 bits, for Type this is (still) 1 bit
⇒ choose the attribute that minimizes the remaining information needed

$$\frac{2}{12}H(<0,1>) + \frac{4}{12}H(<1,0>) + \frac{6}{12}H(<\frac{2}{6},\frac{4}{6}>)$$
$$= -\frac{1}{2}(\frac{1}{3}log_2\frac{1}{3} + \frac{2}{3}log_2\frac{2}{3})$$
$$= 0.4591$$

9.3 Broadening the applicability of decision trees

- Overfitting: the model corresponds too closely or exactly to a particular set of data, and may therefore fail to fit to additional data or predict future observations reliably.
- Missing data: not all the attribute values will be known for every example
- Multivalued attributes: an attribute has many possible values
- Continuous and integer-valued input attributes: find the split point that gives the highest information gain

9.4 Evaluating And Choosing The Best Hypothesis

9.4 Evaluating And Choosing The Best Hypothesis

- We want to learn a hypothesis that fits the future data best.
- "Best fit": the error rate of a hypothesis as the proportion of mistakes it makes—the proportion of times that h(x) <> y for an (x, y) example.
- Small number of examples: k-fold cross-validation

  split the data into k equal subsets

  perform k rounds of learning

  on each round: 1/k of the data is held out as a test set,

  the remaining examples are used as training data

- Rather large number of examples: split examples into 3 sets:
  - training set: train h
  - validation set: choose the best h
  - testing set: evaluate h on the unseen data

9.4 Model selection

An algorithm to select the model that has the lowest error rate on validation data by

- building models of increasing complexity
- choosing the one with best empirical error rate on validation data.

9.4 From error rates to loss

It is traditional to express utilities by a loss function.
The loss function $L(x, y_1, y_2)$ is defined as the amount of utility lost by predicting $h(x) = y_2$ when the correct answer is $f(x) = y_1$:

$L(x, y_1, y_2) =$ Utility(result of using y_1 given an input x)
$-$ Utility(result of using y_2 given an input x)

Absolute value loss: $L_1(y_1,y_2)=|y_1-y_2|$
Squared error loss: $L_2(y_1, y_2) = (y_1 - y_2)^2$

9.5 Regression And Classification With Linear Models

9.5 Univariate linear regression

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

$$y = w_1 x + w_0,$$

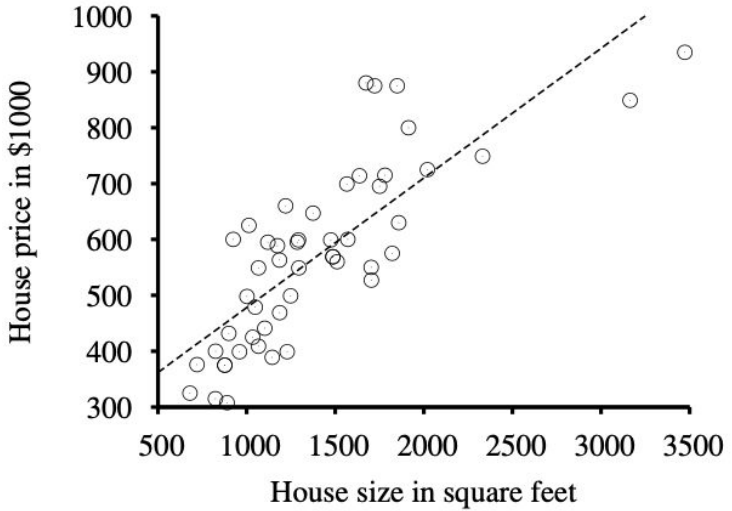
where w_0 and w_1 are weights to be learned.

We'll define **w** to be the vector $[w_0, w_1]$, and define

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

9.5 Univariate linear regression

- An example of a training set of n points in the x, y plane: data points of price versus floor space of houses for sale
- Aim: find h_w that best fits these data or find the values of weights $[w_0, w_1]$ to minimize the empirical loss



(a) Data points of price versus floor space of houses for sale along with the linear function hypothesis that minimizes squared error loss: $y = 0.232x + 246$.

9.5 Univariate linear regression

- Hill-climbing algorithm follows the gradient of the function to be optimized. Minimize the loss, we will use gradient descent.
  - Start from a point in weight space
  - Move to a neighboring point that is downhill, repeating until we converge on the minimum possible loss
- α: the step size or the learning rate when we are trying to minimize loss in a learning problem.

$\mathbf{w} \leftarrow$ any point in the parameter space

loop until convergence **do**

  for each w_i **in w do**

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

9.5 Linear classifiers with a hard threshold

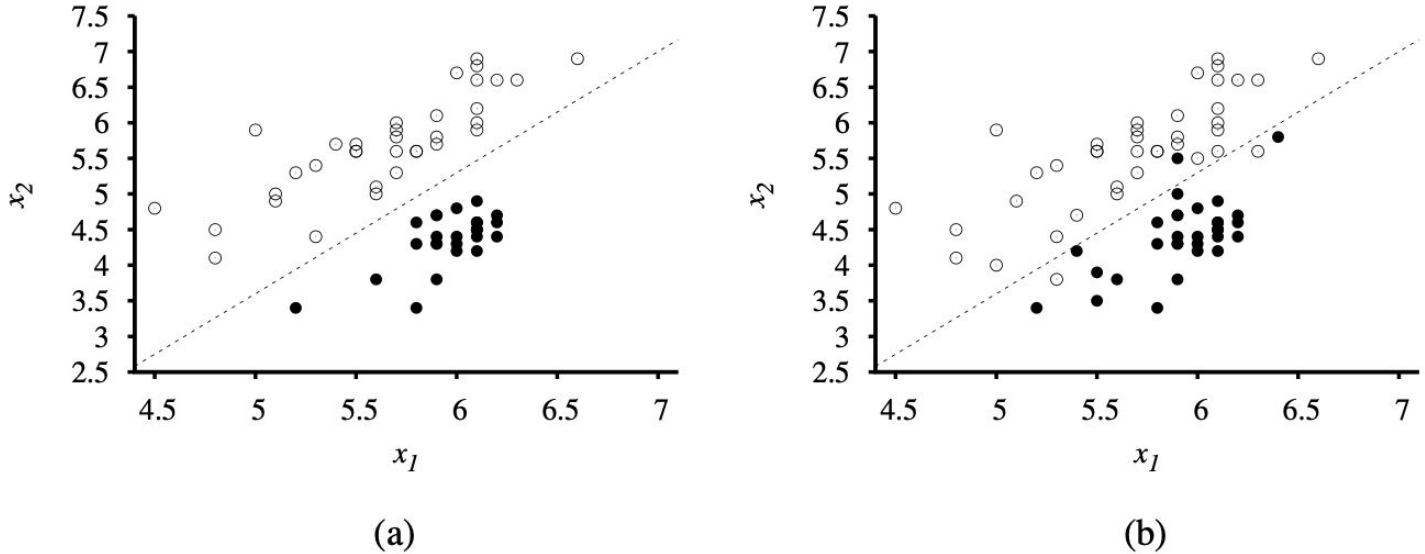
- Linear functions can be used to do classification and regression.
- The task of classification is to learn a hypothesis h that will take new (x_1 , x_2) points and return either 0 for earthquakes or 1 for explosions.
- A decision boundary is a line (or a surface, in higher dimensions) that separates the two classes.

9.5 Linear classifiers with a hard threshold

Example: the decision boundary is a straight line $1.7x_1 - 4.9 - x2 = 0$

(a) Plot of two data parameters, body wave magnitude x_1 and surface wave magnitude x_2, for earthquakes (white circles) and nuclear explosions (black circles). Also shown is a decision boundary between the classes.

(b) The same domain with more data points. The earthquakes and explosions are no longer linearly separable.



9.5 Linear classifiers with a hard threshold

The explosions are points for which $-4.9 + 1.7x1 - x2 > 0$

while earthquakes have $-4.9 + 1.7x1 - x2 < 0$

=> the classification hypothesis

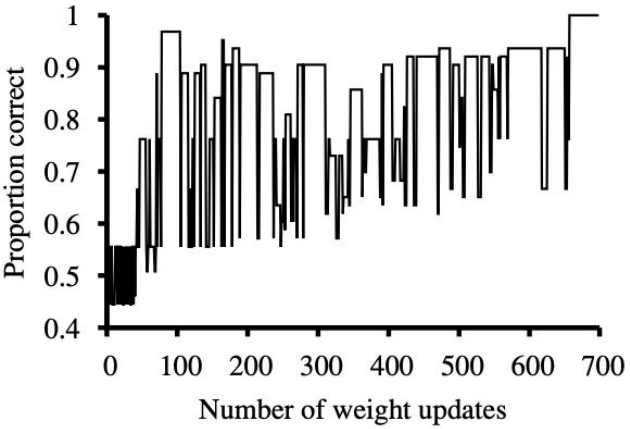
$h_{\mathbf{w}}(x) = 1$ if $\mathbf{w} \cdot x \geq 0$ and 0 otherwise.

OR $h_{\mathbf{w}}(x) = \text{Threshold}(w \cdot x)$ where $\text{Threshold}(w \cdot x) = 1$ if $w \cdot x \geq 0$ and 0 otherwise.

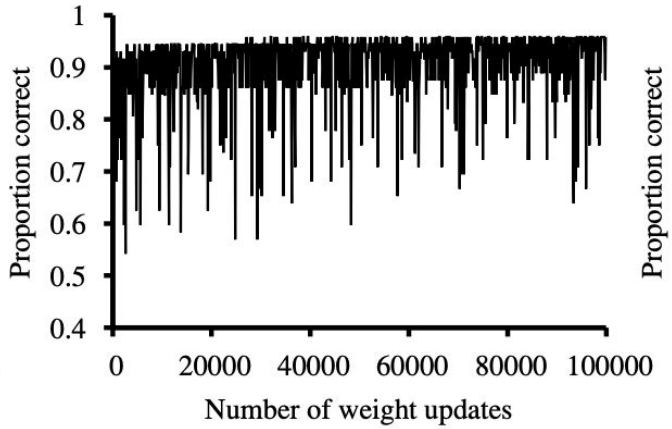
9.5 Linear classifiers with a hard threshold

A training curve measures the classifier performance on a fixed training set as the learning process proceeds on that same training set.

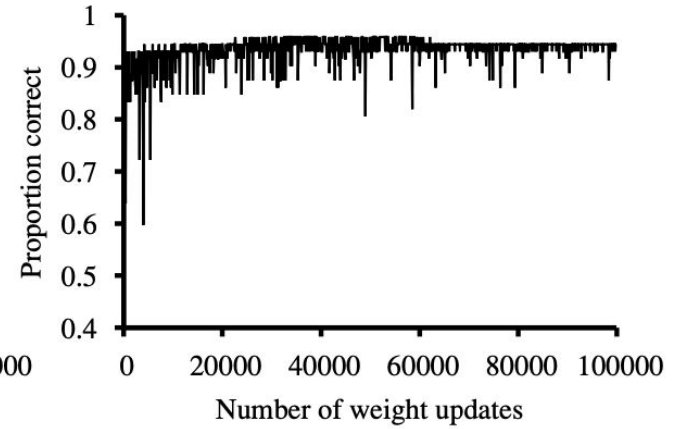
(a) Plot of total training-set accuracy vs. number of iterations through the training set, given the earthquake/explosion data in Figure 18.15(a).
(b) The same plot for the noisy, non-separable data in Figure 18.15(b); note the change in scale of the x-axis.
(c) The same plot as in (b), with a learning rate schedule $\alpha(t) = 1000/(1000 + t)$.



(a)



(b)



(c)

9.3 Linear classification with logistic regression

- Linear classifier: passing the output of a linear function through the threshold function
- The linear classifier always announces a completely confident prediction of 1 or 0, even for examples that are very close to the boundary
- Soften the threshold function by approximating the hard threshold with a continuous, differentiable function.

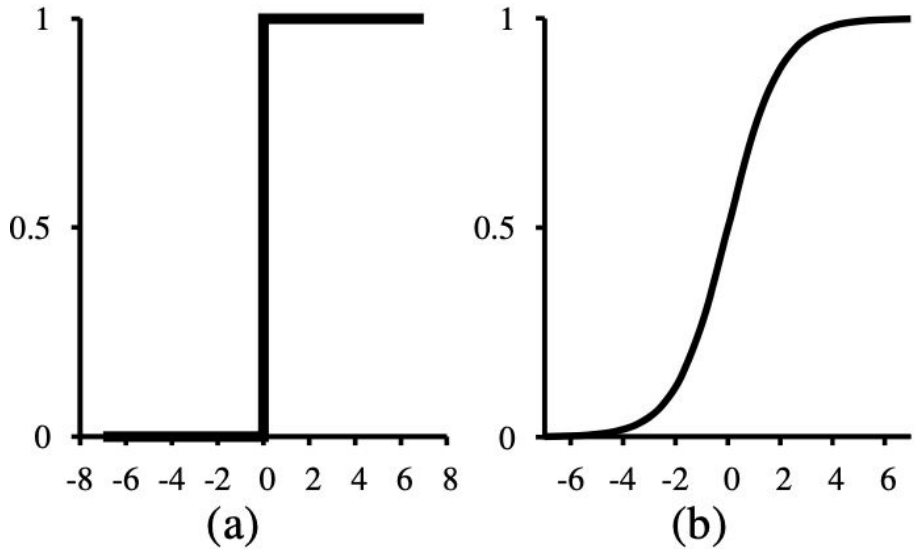
$$Logistic(z) = \frac{1}{1 + e^{-z}} \qquad h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}}$$

- The process of fitting the weights of this model to minimize loss on a data set is called logistic regression
  The output, being a number between 0 and 1, can be interpreted as a probability of belonging to the class labeled 1

9.3 Linear classification with logistic regression

(a) The hard threshold function Threshold(z) with 0/1 output. Note that the function is nondifferentiable at z=0.

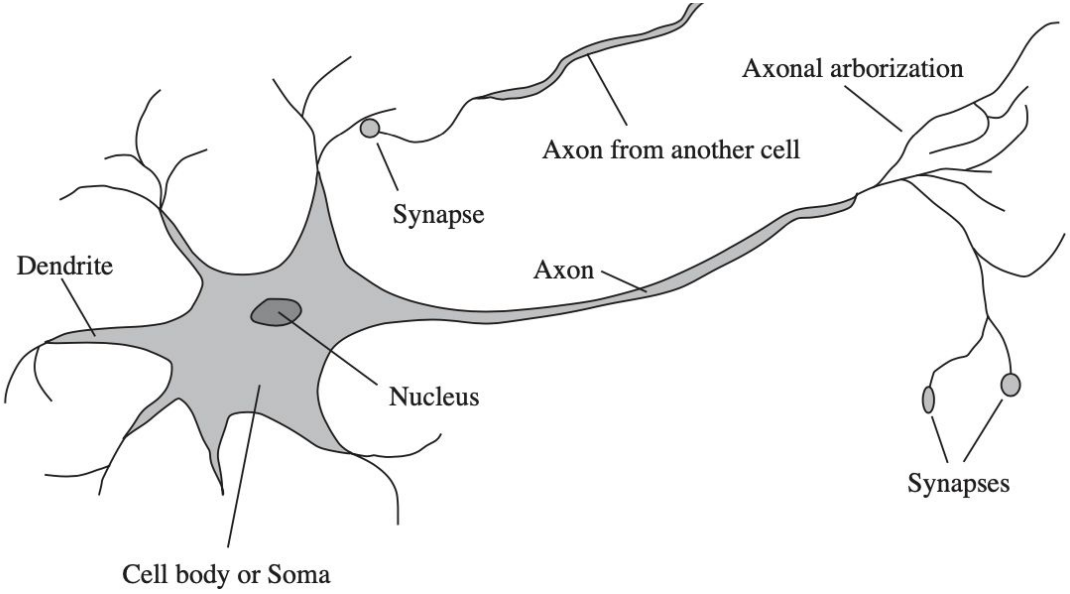
(b) The logistic function, it is also known as the sigmoid function.



9.6 Artificial Neural Networks

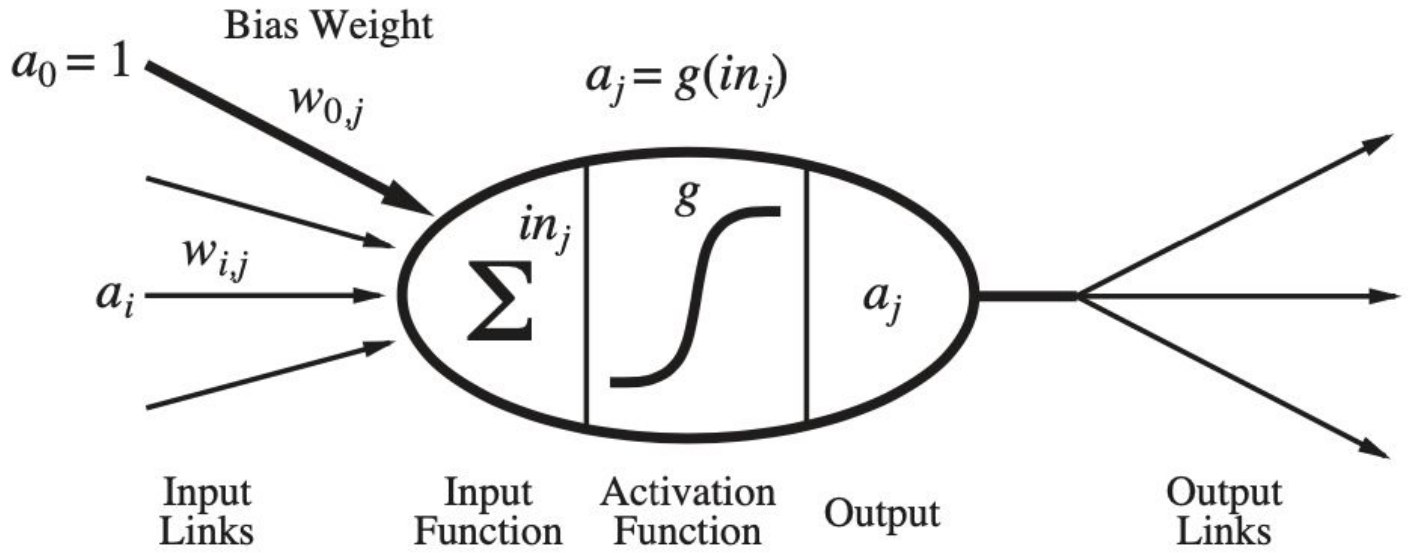
9.6 Brain

- Each neuron consists of a cell body, or soma, that contains a cell nucleus.
- Branching out from the cell body are a number of fibers called dendrites and a single long fiber called the axon.
- A neuron makes connections with 10 to 100,000 other neurons at junctions called synapses.
- Signals are propagated from neuron to neuron.
- 10^{11} neurons of > 20 types, 1014 synapses



9.6 Neural network structures

- A simple mathematical model for a neuron.



9.6 Neural network structures

- Neural networks are composed of nodes or units connected by directed links.
- A link from unit i to unit j serves to propagate the activation a_i from i to j
- Each link also has a numeric weight $w_{i,j}$ associated with it, which determines the strength and sign of the connection.
- Each unit has a dummy input $a_0 = 1$ with an associated weight $w_{0,j}$.
- Each unit j first computes a weighted sum of its inputs:

$$in_j = \sum_{i=0}^{n} w_{i,j} a_i$$

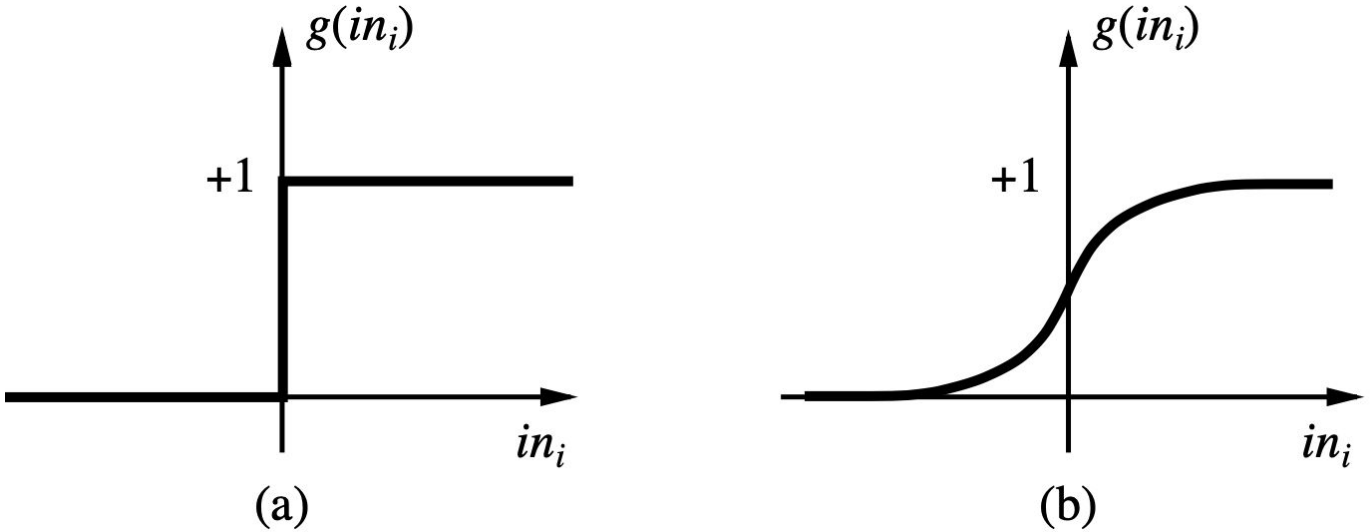
- It applies an activation function g to this sum to derive the output:

$a_j = g(in_j) = g(\sum_{i=0}^{n} w_{i,j} a_i)$,

9.6 Activation functions

(a) is a step function or threshold function

(b) is a sigmoid function $1/(1 + e^{-x})$



9.6 Neural network structures

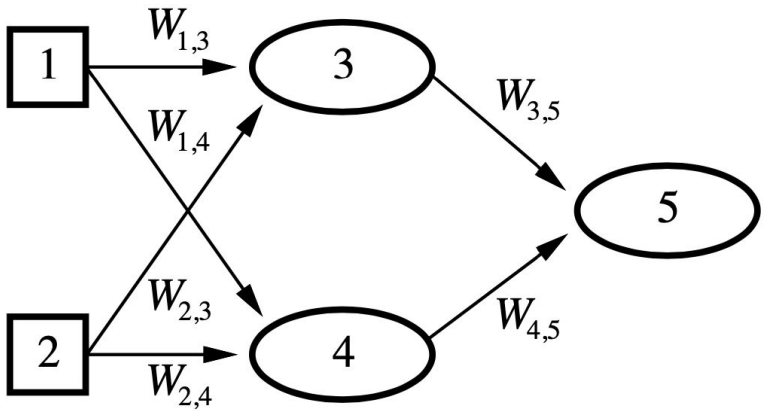
- A feed-forward network has connections only in one direction.
  - Every node receives input from “upstream” nodes and delivers output to “downstream” nodes; there are no loops.
  - The networks are usually arranged in layers, such that each unit receives input only from units in the immediately preceding layer.
- A recurrent network feeds its outputs back into its own inputs.

9.6 Single-layer feed-forward neural networks

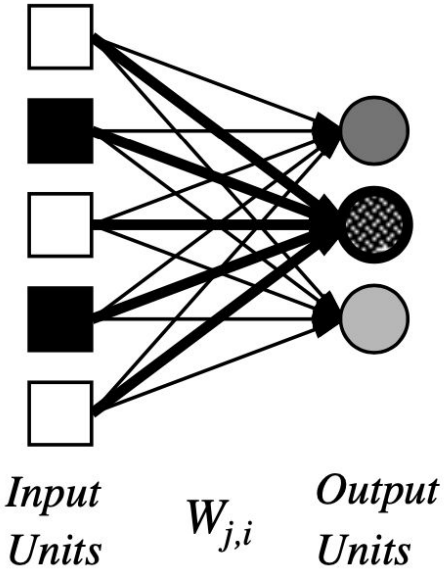
- A network with all the inputs connected directly to the outputs is called a single-layer neural network, or a perceptron network.
- Example: a perceptron network with two inputs and two output units.

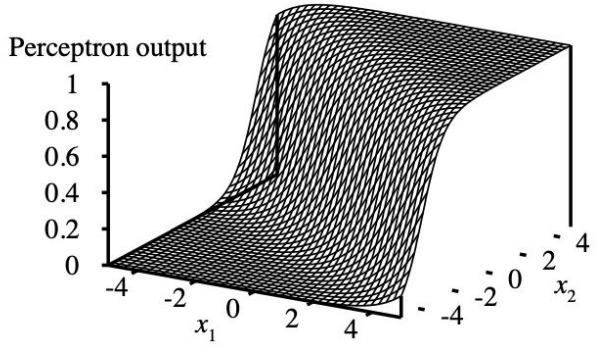
$$a_5 = g(W_{3,5} \cdot a_3 + W_{4,5} \cdot a_4)$$
$$= g(W_{3,5} \cdot g(W_{1,3} \cdot a1 + W_{2,3} \cdot a_2) + W_{4,5} \cdot g(W_{1,4} \cdot a_1 + W_{2,4} \cdot a_2))$$

  - adjusting weights changes the function $g_w(x)$: do learning this way!



9.6 Single-layer feed-forward neural networks



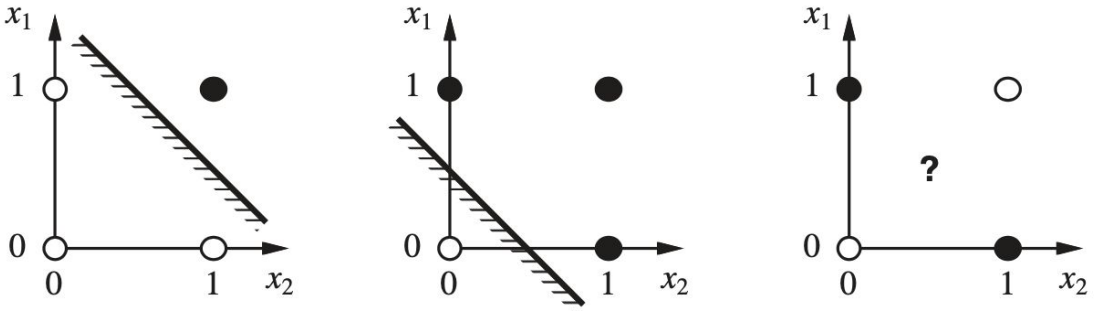


- Output units all operate separately - no shared weights
- Adjusting weights moves the location, orientation, and steepness of cliff

9.6 Single-layer feed-forward neural networks

Expressiveness of perceptrons

- Consider a perceptron with g = step function
- Can represent AND, OR, NOT, majority, etc., but not XOR
- Represents a linear separator in input space



Linear separability in threshold perceptrons.
Black dots indicate a point in the input space where the value of the function is 1, and white dots indicate a point where the value is 0.
The perceptron returns 1 on the region on the non-shaded side of the line.
In (c), no such line exists that correctly classifies the inputs.

9.6 Perceptron learning

Learn by adjusting weights to reduce error on training set
The squared error for an example with input **x** and true output y is

$$E = \frac{1}{2}Err^2 \equiv \frac{1}{2}(y - h_{\mathbf{W}}(\mathbf{x}))^2$$

Perform optimization search by gradient descent:

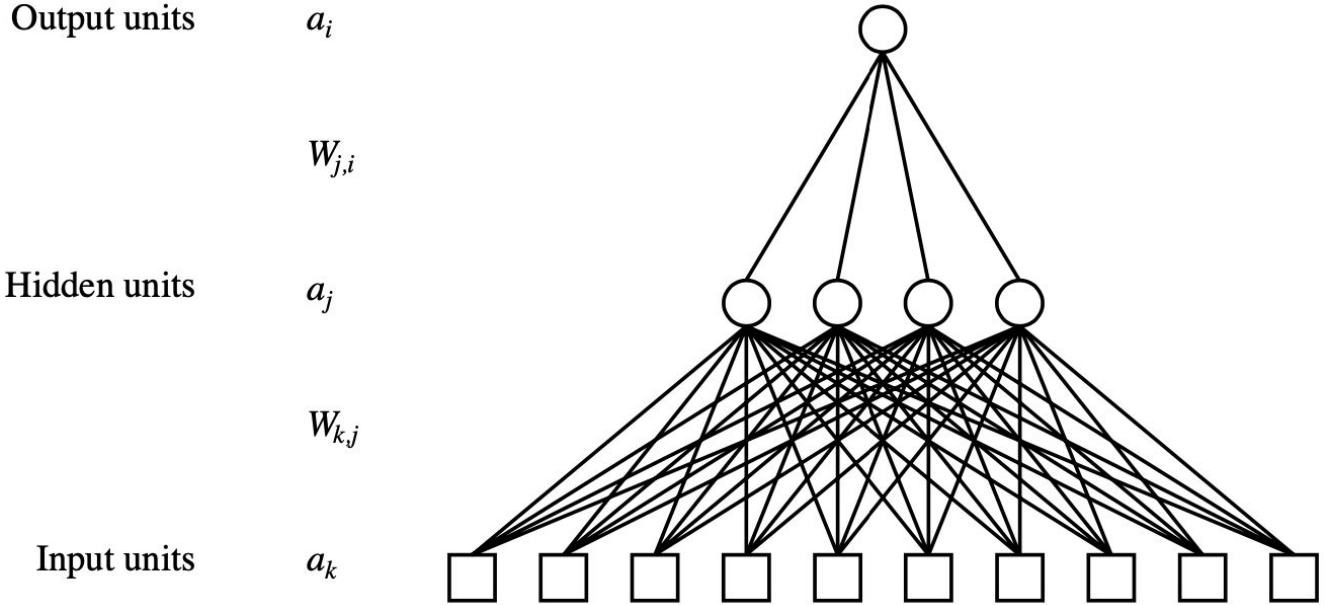
$$\begin{aligned}\frac{\partial E}{\partial W_j} &= Err \times \frac{\partial Err}{\partial W_j} = Err \times \frac{\partial}{\partial W_j}\left(y - g(\Sigma_{j=0}^{n} W_j x_j)\right) \\ &= -Err \times g'(in) \times x_j\end{aligned}$$

Simple weight update rule:

$$W_j \leftarrow W_j + \alpha \times Err \times g'(in) \times x_j$$

9.6 Multilayer feed-forward neural networks

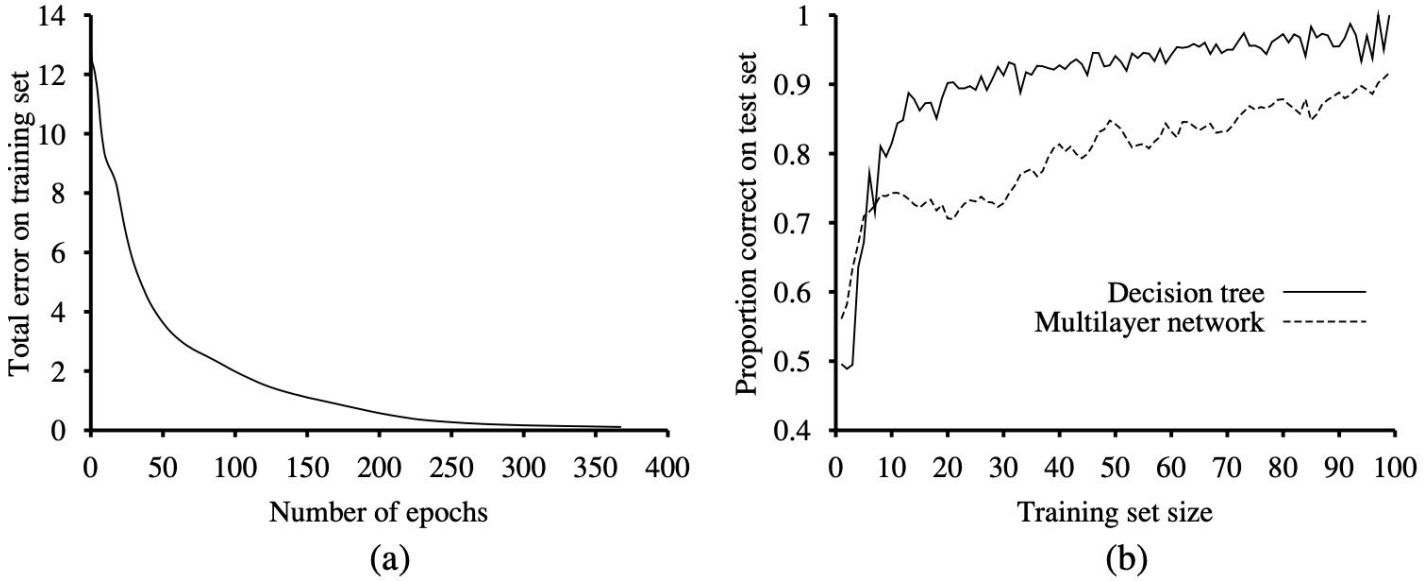
Layers are usually fully connected; numbers of hidden units typically chosen by hand



function BACK-PROP-LEARNING($examples$, $network$) **returns** a neural network
inputs: $examples$, a set of examples, each with input vector **x** and output vector **y**
$network$, a multilayer network with L layers, weights $w_{i,j}$, activation function g
local variables: Δ, a vector of errors, indexed by network node

repeat
for each weight $w_{i,j}$ in $network$ **do**
$w_{i,j} \leftarrow$ a small random number
for each example $(\mathbf{x}, \mathbf{y})$ **in** $examples$ **do**
/* *Propagate the inputs forward to compute the outputs* */
for each node i in the input layer **do**
$a_i \leftarrow x_i$
for $\ell = 2$ **to** L **do**
for each node j in layer ℓ **do**
$in_j \leftarrow \sum_i w_{i,j} a_i$
$a_j \leftarrow g(in_j)$
/* *Propagate deltas backward from output layer to input layer* */
for each node j in the output layer **do**
$\Delta[j] \leftarrow g'(in_j) \times (y_j - a_j)$
for $\ell = L - 1$ **to** 1 **do**
for each node i in layer ℓ **do**
$\Delta[i] \leftarrow g'(in_i) \sum_j w_{i,j} \Delta[j]$
/* *Update every weight in network using deltas* */
for each weight $w_{i,j}$ in $network$ **do**
$w_{i,j} \leftarrow w_{i,j} + \alpha \times a_i \times \Delta[j]$
until some stopping criterion is satisfied
return $network$

9.6 Multilayer feed-forward neural networks



(a) Training curve showing the gradual reduction in error as weights are modified over several epochs, for a given set of examples in the restaurant domain.
(b) Comparative learning curves showing that decision-tree learning does slightly better on the restaurant problem than back-propagation in a multilayer network.

9.6 Learning neural network structures

- Neural networks are subject to overfitting when there are too many parameters in the model.
- If we stick to fully connected networks, the only choices to be made concern the number of hidden layers and their sizes.
- The usual approach is to try several and keep the best which gives the highest prediction accuracy on the validation sets.