Neural Networks - A comprehensive foundation

Simon Haykin

Prentice-Hall, 1998

2nd edition

1. Preface

- **Artificial Neural Network**:
  - consists of simple, adaptive processing units, called often **neurons**
  - the neurons are **interconnected**, forming a large network
  - **parallel computation**, often in layers
  - **nonlinearities** are used in computations

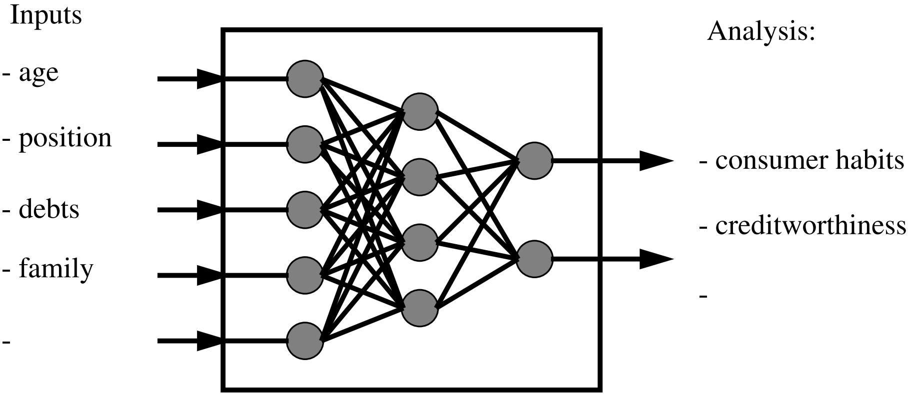
- Important property of neural networks: **learning from input data**.
  - with teacher (supervised learning)
  - without teacher (unsupervised learning)

- **Artificial neural networks** have their roots in:
  - neuroscience
  - mathematics and statistics
  - computer science
  - engineering

- Neural computing was inspired by computing in human brains

- **Application areas of neural networks**:
  - modeling
  - time series processing
  - pattern recognition
  - signal processing
  - automatic control
- **Computational intelligence**
  - Neural networks
  - Fuzzy systems
  - Evolutionary computing
    - Genetic algorithms

Neural computing has many application areas in economics and management, because a lot of data which can be used in training of the neural network have been saved in databases.



Principle of neural modeling. The inputs are known or they can be measured. The behavior of outputs is investigated when input varies.

All information has to be converted into vector form.

2. Contents of Haykin's book

1. **Introduction**
2. **Learning Processes**
3. **Single Layer Perceptrons**
4. **Multilayer Perceptrons**
5. **Radial-Basis Function Networks**
6. Support Vector Machines
7. Committee Machines
8. Principal Components Analysis
9. **Self-Organizing Maps**
10. Information-theoretic Models
11. Stochastic Machines and Their Approximates Rooted in Statistical Mechanics
12. Neurodynamic Programming
13. Temporal Processing Using Feedforward Networks
14. Neurodynamics
15. Dynamically Driven Recurrent Networks

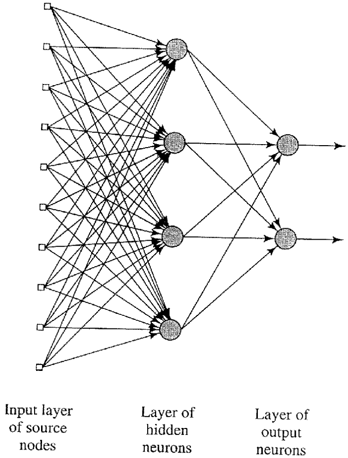
The **boldfaced** chapters will be discussed in this course

1. Introduction

Neural networks resemble the brain in two respects:

1. The network acquires knowledge from its environment using a **learning process (algorithm)**

2. **Synaptic weights**, which are interneuron connection strenghts, are used to store the learned information.



Fully connected 10-4-2 feedforward network with 10 source (input) nodes, 4 hidden neurons, and 2 output neurons.

1.1 Benefits of neural networks

1. **Nonlinearity**
   - Allows modeling of nonlinear functions and processes.
   - Nonlinearity is distributed through the network.
   - Each neuron typically has a nonlinear output.
   - Using nonlinearities has **drawbacks**, too: local minima, difficult analysis, no closed-form easy linear solutions.

2. **Input-Output Mapping**
   - In supervised learning, the input-output mapping is learned from training data.
   - For example known prototypes in classification.
   - Typically, some statistical criterion is used.
   - The synaptic weights (free parameters) are modified to optimize the criterion.

3. **Adaptivity**
   - Weights (parameters) can be retrained with new data.
   - The network can adapt to nonstationary environment.
   - However, the changes must be slow enough.

4. Evidential Response

5. Contextual Information

6. Fault Tolerance

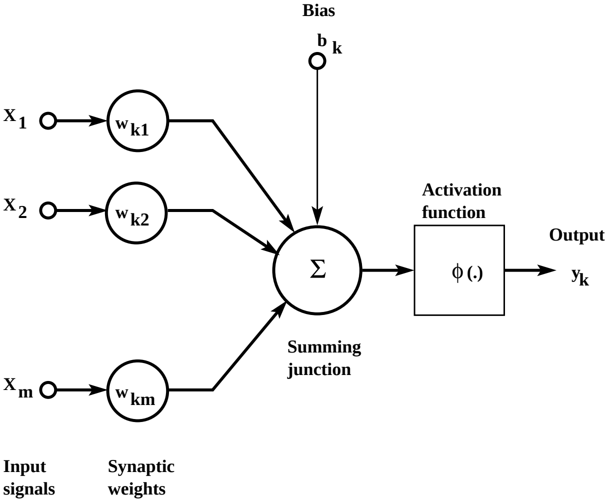
7. VLSI Implementability

8. Uniformity of Analysis and Design

9. **Neurobiological Analogy**
   - Human brains are fast, powerful, fault tolerant, and use massively parallel computing.
   - **Neurobiologists** try to explain the operation of human brains using artificial neural networks.
   - **Engineers** use neural computation principles for solving complex problems.

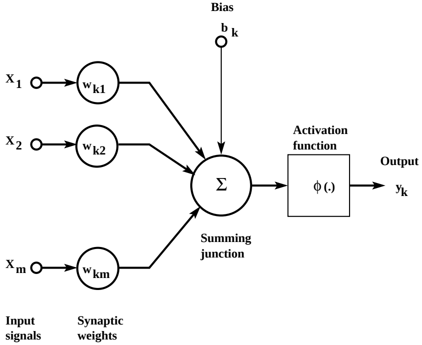
1.3 Models of a neuron

A **neuron** is the fundamental information processing unit of a neural network.



The **neuron model** consists of three (or four) basic elements

1. A set of **synapses** or **connecting links**:
   - Characterized by **weights** (strengths).
   - Let x_j denote a signal at the input of synapse j.
   - When connected to neuron k, x_j is multiplied by the synaptic weight w_{kj}.
   - weights are usually real numbers.



2. An **adder** (linear combiner):
   - Sums the weighted inputs $w_{kj}x_j$.

3. An **activation function**:
   - Applied to the output of a neuron, limiting its value.
   - Typically a nonlinear function.
   - Called also **squashing function**.

4. Sometimes a neuron includes an externally applied **bias** term b_k.

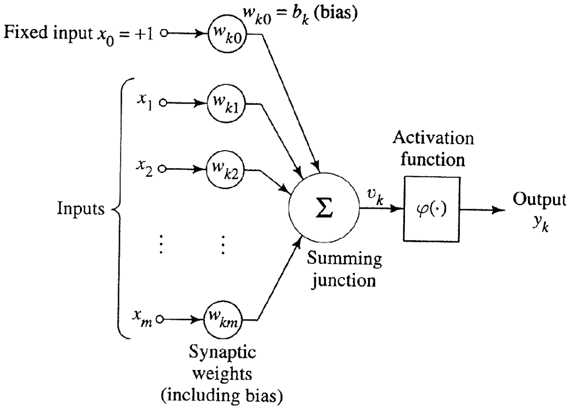
Mathematical equations describing neuron k:

$$u_k = \sum_{j=1}^{m} w_{kj} x_j, \tag{1}$$

$$y_k = \varphi(u_k + b_k). \tag{2}$$

Here:
- u_k is the linear combiner output;
- $\varphi(.)$ is the activation function;
- y_k is the output signal of the neuron;
- $x_1, x_2, \ldots, x_m$ are the m input signals;
- $w_{k1}, w_{k2}, \ldots, w_{km}$ are the respective m synaptic weights.

A mathematically equivalent representation:
- Add an extra synapse with input $x_0 = +1$ and weight $w_{k0} = b_k$.



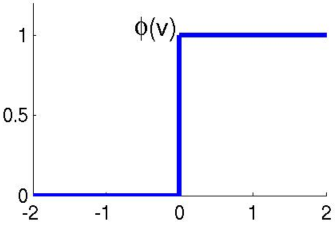
- The equations are now slightly simpler:

$$v_k = \sum_{j=0}^{m} w_{kj} x_j, \tag{3}$$

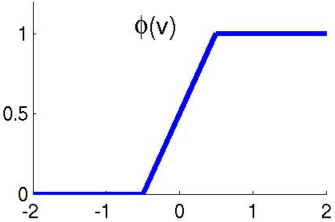
$$y_k = \varphi(v_k). \tag{4}$$

Typical activation functions

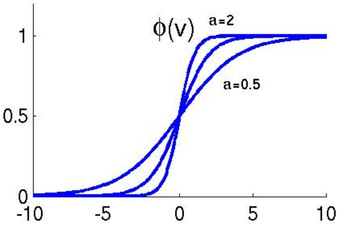
1. Threshold function $\phi(v) = 1$, $v \geq 0$; $\phi(v) = 0$, if $v < 0$



2. Piecewise-linear function: Saturates at 1 and 0



3. Sigmoid function



- Most commonly used in neural networks
- The figure shows the logistic function defined by $\phi(v) = \frac{1}{1+e^{-av}}$
- The slope parameter a is important
- When $a \to \infty$, the logistic sigmoid approaches the threshold function (1.)
- Continuous, balance between linearity and nonlinearity
- $\phi(v) = tanh(av)$ allows the activation function to have negative values

Stochastic model of a neuron

- The activation function of the **McCulloch-Pitts** early neuronal model (1943) is the threshold function.
- The neuron is permitted to reside in only two states, say $x = +1$ and $x = -1$.
- In the stochastic model, a neuron **fires** (switches its state x) according to a probability.
- The state is $x = 1$ with probability $P(v)$
  The state is $x = -1$ with probability $1 - P(v)$
- A standard choice for the probability is the sigmoid type function

$$P(v) = \frac{1}{1 + \exp(-v/T)}$$

- Here T is a parameter controlling the uncertainty in firing, called pseudotemperature.

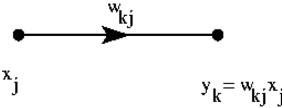
1.4 Neural networks as directed graphs

- Neural networks can be represented in terms of **signal-flow graphs**.
- Nonlinearities appearing in a neural network cause that two different types of **links (branches)** can appear:
  1. **Synaptic links** having a linear input-output relation: $y_k = w_{kj} x_j$.
  2. **Activation links** with a nonlinear input-output relation: $y_k = \varphi(x_j)$.

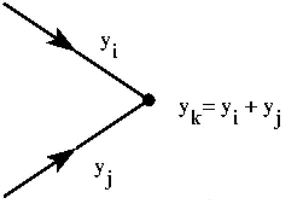
Signal-flow graphs

- Signal-flow graph consists of directed branches
- The branches sum up in nodes
- Each node j has a signal x_j
- Branch kj starts from node j and ends at node k; w_{kj} is the synaptic weight corresponding the strengthening or damping of signal

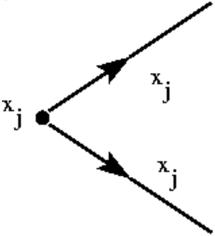
- Three basic rules:
  - Rule 1.
    Signal flows only to the direction of arrow. Signal strength will be multiplied with strengthening factor w_{kj}.



  - Rule 2.
    Node signal $=$ Sum of incoming signals from branches

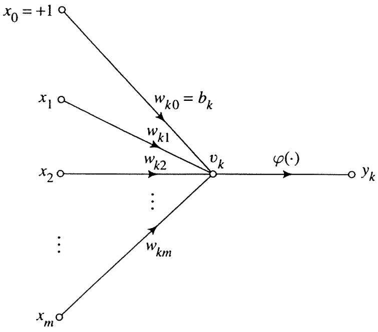


  - Rule 3.
    Node signal will be transmitted to each outgoing branch; strengthening factors are independent of node signal



Example: Signal flow graph of linear combination

$$v_k = \sum_{j=0}^{M} w_{kj} x_j \qquad (5)$$

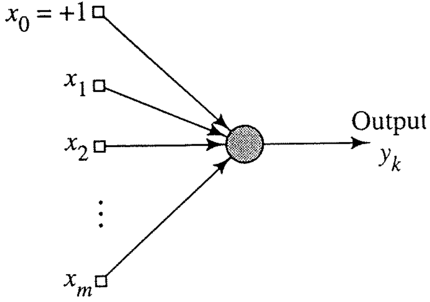


- coefficients w_{k0}, w_{k1} $\ldots w_{kM}$ are weights
- x_0, $x_1 \ldots x_M$ are input signals
- by defining $\mathbf{w}_k = [w_{k0}, w_{k1} \ldots w_{kM}]^T$ and $\mathbf{x} = [x_0, x_1 \ldots x_M]^T$

$$v_k = \mathbf{w}_k{}^T \mathbf{x} = \mathbf{x}^T \mathbf{w}_k \qquad (6)$$

- Thus rule 1 is divided into 2 parts, while the basic rules 2 and 3 for handling signal-flow graphs remain unchanged.
- In Haykin's book, a mathematical definition of a neural network as a directed graph is represented on page 17.

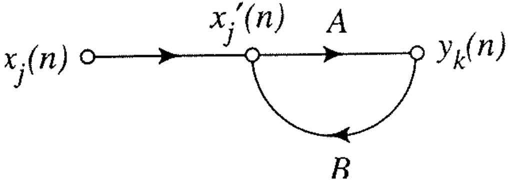
- Often the signal flow inside a neuron is not considered.
- This leads to so-called **architectural graph**, which describes the layout of a neural network.

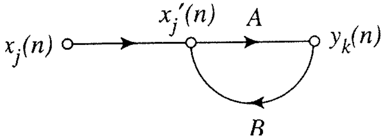


- This is the typical representation showing the structure of a neural network.

1.5 Feedback

- Feedback: Output of an element of a dynamic system affects to the input of this element.
- Thus in a feedback system there are closed paths.
- Feedback appears almost everywhere in natural nervous systems.
- Important in recurrent networks (Chapter 15 in Haykin).
- Signal-flow graph of a **single-loop feedback system**





- The system is discrete-time and linear.

- Relationships between the input signal $x_j(n)$, internal signal $x_j'(n)$, and output signal $y_k(n)$:

$$y_k(n) = A[x_j'(n)],$$

$$x_j'(n) = x_j(n) + B[y_k(n)]$$

where A and B are linear operators.

- Eliminating the internal signal $x_j'(n)$ yields

$$y_k(n) = \frac{A}{1 - AB}[x_j(n)].$$

Here $A/(1-AB)$ is called the closed-loop operator of the system, and AB the open-loop operator.

- **Stability** is a major issue in feedback systems.

- If the feedback terms are too strong, the system may diverge or become unstable.

- An example is given in Haykin, pp. 19-20.

- Stability of linear feedback systems (IIR filters) is studied in digital signal processing.

- Feedback systems have usually a **fading, infinite memory**.

- The output depends on all the previous samples, but usually the less the older the samples are.

- Studying the stability and dynamic behavior of feedback (recurrent) neural networks is complicated because of nonlinearities.

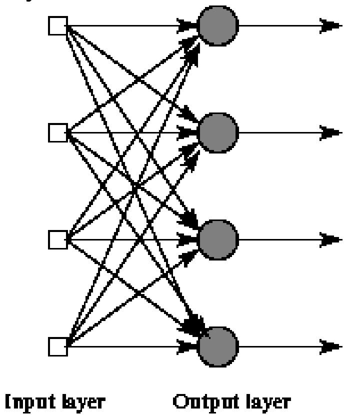
1.6 Network Architectures

- The structure of a neural network is closely related with the learning algorithm used to train the network.
- Learning algorithms are classified in chapter 2 of Haykin.
- Different learning algorithms are discussed in subsequent chapters.
- There are three fundamentally different classes of network architectures.

Single-Layer Feedforward Networks

- The simplest form of neural networks.
- The **input layer** of source nodes projects onto an **output layer** of neurons (computation nodes).
- The network is strictly a **feedforward** or acyclic type, because there is no feedback.
- Such a network is called a **single-layer network**.

- A single-layer network with four nodes in both the input and output layers.

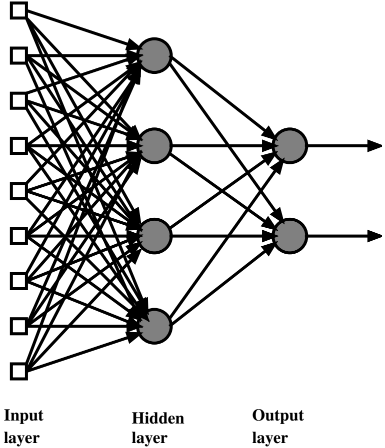


- The input layer is not counted as a layer because no computation is performed there.

Multilayer Feedforward Networks

- In a multilayer network, there is one or more **hidden layers**.
- Their computation nodes are called **hidden neurons** or **hidden units**.
- The hidden neurons can extract higher-order statistics and acquire more global information.
- Typically the input signals of a layer consist of the output signals of the preceding layer only.

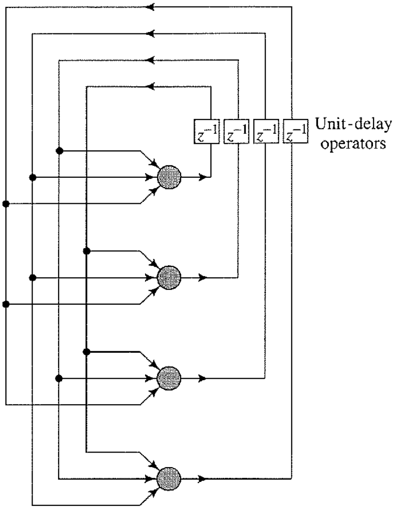
- a 9-4-2 feedforward network with 9 source (input) nodes, 4 hidden neurons, and 2 output neurons.



- The network is **fully connected**: all the nodes between subsequent layers are connected.

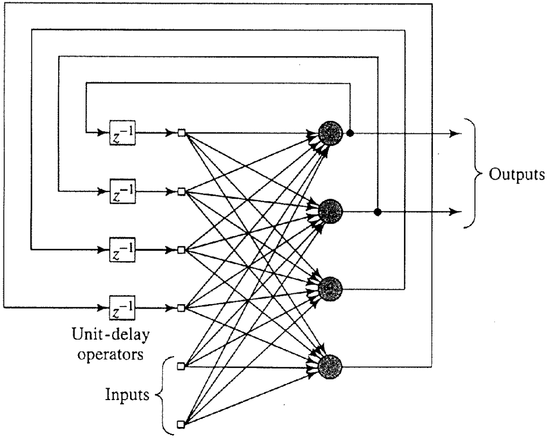
Recurrent Networks

- A **recurrent neural network** has at least one **feedback loop**.
- In a feedforward network there are no feedback loops.
- Recurrent network with:
  - No self-feedback loops to the "own" neuron.
  - No hidden neurons.



- Another recurrent network which has hidden neurons.
- The feedback loops have a profound impact on the learning capability and performance of the network.

- The unit-delay elements result in a nonlinear dynamical behavior if the network contains nonlinear elements.



1.7 Knowledge Representation

- **Definition:** Knowledge refers to stored information or models used by a person or machine to interpret, predict, and appropriately respond to the outside world.

- In **knowledge representation** one must consider:
  1. What information is actually made explicit;
  2. How the information is physically encoded for subsequent use.

- A well performing neural network must represent the knowledge in an appropriate way.

- A real design challenge, because there are highly diverse ways of representing information.

- A major task for a neural network: learn a model of the world (environment) where it is working.

- Two kinds of information about the environment:

    1. **Prior information** = the known facts.
    2. Observation (measurements). Usually noisy, but give **examples** (prototypes) for training the neural network.

- The examples can be:
  \- **labeled**, with a known **desired response** (target output) to an input signal.
  \- **unlabeled**, consisting of different realizations of the input signal.

- A set of pairs, consisting of an input and the corresponding desired response, form a **set of training data** or **training sample**.

An example: Handwritten digit recognition

- Input signal: a digital image with black and white pixels.
- Each image represents one of the 10 possible digits.
- The training sample consists of a large variety of hand-written digits from a real-world situation.
- An appropriate architecture in this case:
  - Input signals consist of image pixel values.
  - 10 outputs, each corresponding to a digit class.
- **Learning:** The network is trained using a suitable algorithm with a subset of examples.
- **Generalization:** After this, the recognition performance of the network is tested with data not used in learning.

Rules for knowledge representation

- The free parameters (synaptic weights and biases) represent knowledge of the surrounding environment.

- Four general rules for knowledge representation.

- **Rule 1.** Similar inputs from similar classes should produce similar representations inside the network, and they should be classified to the same category.

- Let $\mathbf{x}_i$ denote the column vector

$$\mathbf{x}_i = [x_{i1}, x_{i2}, \ldots, x_{im}]^T$$

- Typical similarity measures:

  1. Reciprocal $1/d(\mathbf{x}_i, \mathbf{x}_j)$ of the **Euclidean distance**

     $$d(\mathbf{x}_i, \mathbf{x}_j) = \parallel \mathbf{x}_i - \mathbf{x}_j \parallel$$

     between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$.

  2. The **inner product** $\mathbf{x}_i^T \mathbf{x}_j$ between the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$.

     - If $\mathbf{x}_i$ and $\mathbf{x}_j$ are normalized to unit length, then one can easily see that

     $$d^2(\mathbf{x}_i, \mathbf{x}_j) = 2 - 2\mathbf{x}_i^T \mathbf{x}_j.$$

3. A statistical measure: **Mahalanobis distance**

$$d_{ij}^2 = (\mathbf{x}_i - \mathbf{m}_i)^T \mathbf{C}^{-1} (\mathbf{x}_j - \mathbf{m}_j)$$

Here $\mathbf{m}_i = E[\mathbf{x}_i]$ is the expectation (mean) of the vector (class) $\mathbf{x}_i$, $\mathbf{m}_j$ is the mean of $\mathbf{x}_j$, and

$$\mathbf{C} = E[(\mathbf{x}_i - \mathbf{m}_i)(\mathbf{x}_i - \mathbf{m}_i)^T] = E[(\mathbf{x}_j - \mathbf{m}_j)(\mathbf{x}_j - \mathbf{m}_j)^T]$$

is the common covariance matrix of the classes represented by the vectors $\mathbf{x}_i$ and $\mathbf{x}_j$.

Assumption: difference of the classes is only in their means.

- **Rule 2:** Items to be categorized as separate classes should be given widely different representations in the network.

- **Rule 3:** If a particular feature is important, there should be a large number of neurons involved in representing it in the network.

- **Rule 4:** Prior information and invariances should be built into the design of a neural network.

- Rule 4 leads to neural networks with a **specialized (restricted) structure.**

- Such networks are highly desirable for several reasons:

  1. Biological visual and auditory networks are known to be very specialized.
  2. A specialized network has a smaller number of free parameters.
     - Easier to train, requires less data, generalizes often better.
  3. The rate of information transmission is higher.
  4. Cheaper to build than a more general network because of smaller size.

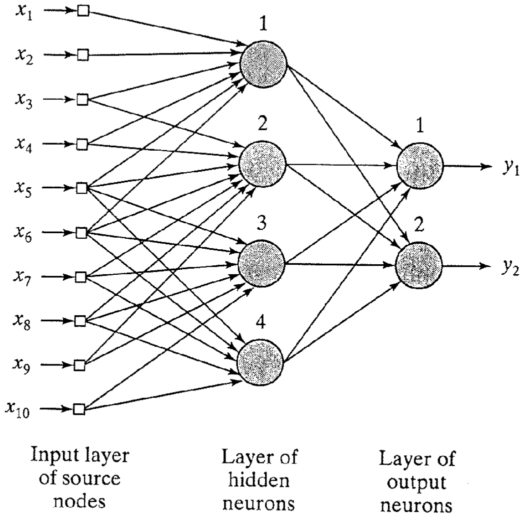
How to Build Prior Information into Neural Network Design

- No general technique exists: ad-hoc procedures which are known to yield useful results are applied instead.

- Two such ad-hoc procedures:

  1. *Restricting the network architecture* through the use of local connections known as *receptive fields*.
  2. *Constraining the choice of synaptic weights* through the use of *weight-sharing*.

- These procedures reduce the number of free parameters to be learned.



Illustrating the combined use of a receptive field and weight sharing. All four hidden neurons share the same set of weights for their synaptic connections.

Bayesian probability theory

- Can be used for incorporating useful prior information
- Usually the data x is assumed to be generated by some model
- A **generative model approach**
- **Prior information** on the model parameters is represented by their **prior probability density** $p(\theta)$
- **Bayes' rule** is then used to compute posterior probabilities:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)} \tag{7}$$

  where $p(x)$ is the unconditional density function used for normalization and $p(x|\theta)$ is the conditional probability

- Somewhere between **classical estimation theory** and **neural networks**

- No simple, adaptive processing in each computational neuron
- Differs from classical estimation theory in that **distributed nonlinear network structures** are used
- Mathematical analysis is often impossible
- **Local minima** may be a problem
- But such nonlinear distributed systems may lead to powerful representations
- Can be used for teaching MLP (multilayer perceptron) or RBF (radial basis function) networks
- Also in **unsupervised manner**

How to Build Invariances into Neural Network Design

- Classification systems must be **invariant** to certain transformations depending on the problem.
- For example, a system recognizing objects from images must be invariant to rotations and translations.
- At least three techniques exist for making classifier-type neural networks invariant to transformations.
  1. **Invariance by Structure**
     - Synaptic connections between the neurons are created so that transformed versions of the same input are forced to produce the same output.
     - **Drawback:** the number of synaptic connections tends to grow very large.

2. **Invariance by Training**
   - The network is trained using different examples of the same object corresponding to different transformations (for example rotations).
   - **Drawbacks:** computational load, generalization ability for other objects.

3. **Invariant feature space**
   - Try to extract *features* of the data invariant to transformations.
   - Use these instead of the original input data.
   - Probably the most suitable technique to be used for neural classifiers.
   - Requires prior knowledge on the problem.

- In Haykin's book, two examples of knowledge representation are briefly described:
  1. A radar system for air surveillance.
  2. Biological sonar system of echo-locating bats.

- Optimization of the structure of a neural network is difficult.

- Generally, a neural network acquires knowledge about the problem through training.

- The knowledge is represented by in a distributed and compact form by the synaptic connection weights.

- Neural networks lack an *explanation capability*.

- A possible solution: integrate a neural network and artificial intelligence into a hybrid system.