Data Mining II: Advanced Methods and Techniques

Lecture 1

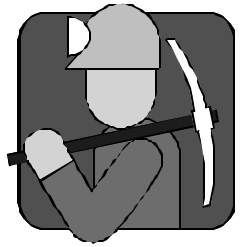
Natasha Balac, Ph.D.

Introduction

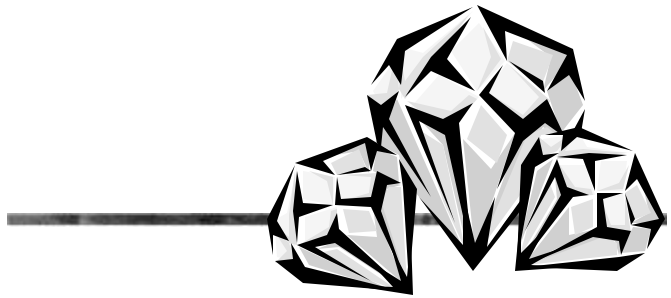
What is DATA MINING?

- *Extracting or "mining" knowledge from large amounts of data*
- Data -driven discovery and modeling of hidden patterns (we never new existed) in large volumes of data
- Extraction of implicit, previously unknown and unexpected, potentially extremely useful information from data

What Is Data Mining?



- Data mining (knowledge discovery in databases):
  - Extraction of interesting **(non-trivial, implicit, previously unknown** and **potentially useful**) information or patterns from data in large databases



Data Mining is NOT

- Data Warehousing
- (Deductive) query processing
  - SQL/ Reporting
- Software Agents
- Expert Systems
- Online Analytical Processing (OLAP)
- Statistical Analysis Tool
- Data visualization

Data Mining

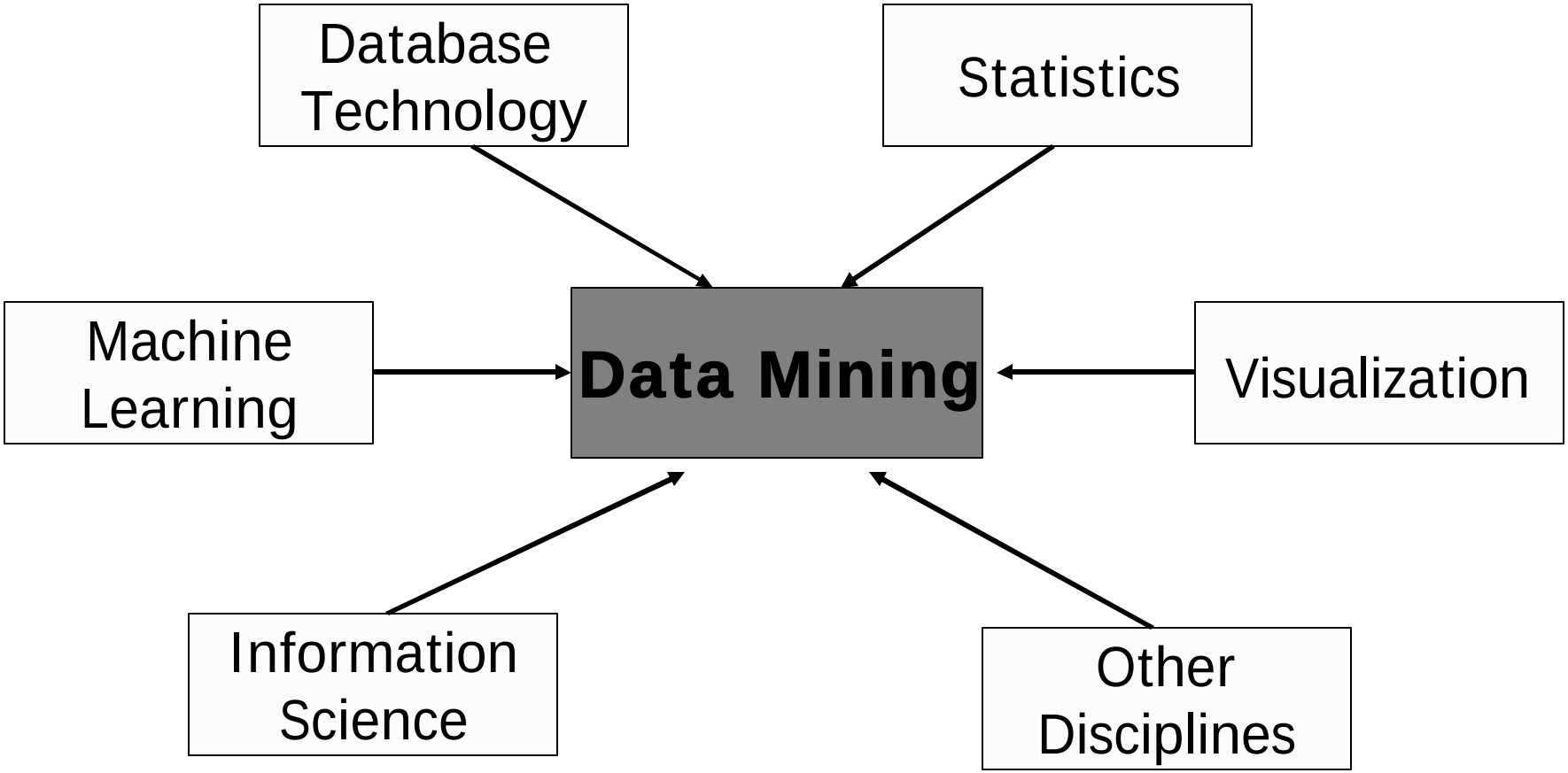
- Programs that detect patterns and regularities in the data
- Strong patterns can be used to make non-trivial predictions on new data

- **Problem 1:** most patterns are not interesting
- **Problem 2**: patterns may be inexact (or even completely spurious) if data is garbled or missing

MACHINE LEARNING TECHNIQUES

- Technical basis for data mining: algorithms for acquiring structural descriptions from examples
- Structural descriptions represent patterns explicitly
- Can be used to predict outcome in new situation
- Can be used to understand and explain how prediction is derived (maybe even more important)
- Methods originate from artificial intelligence, statistics, and research on databases

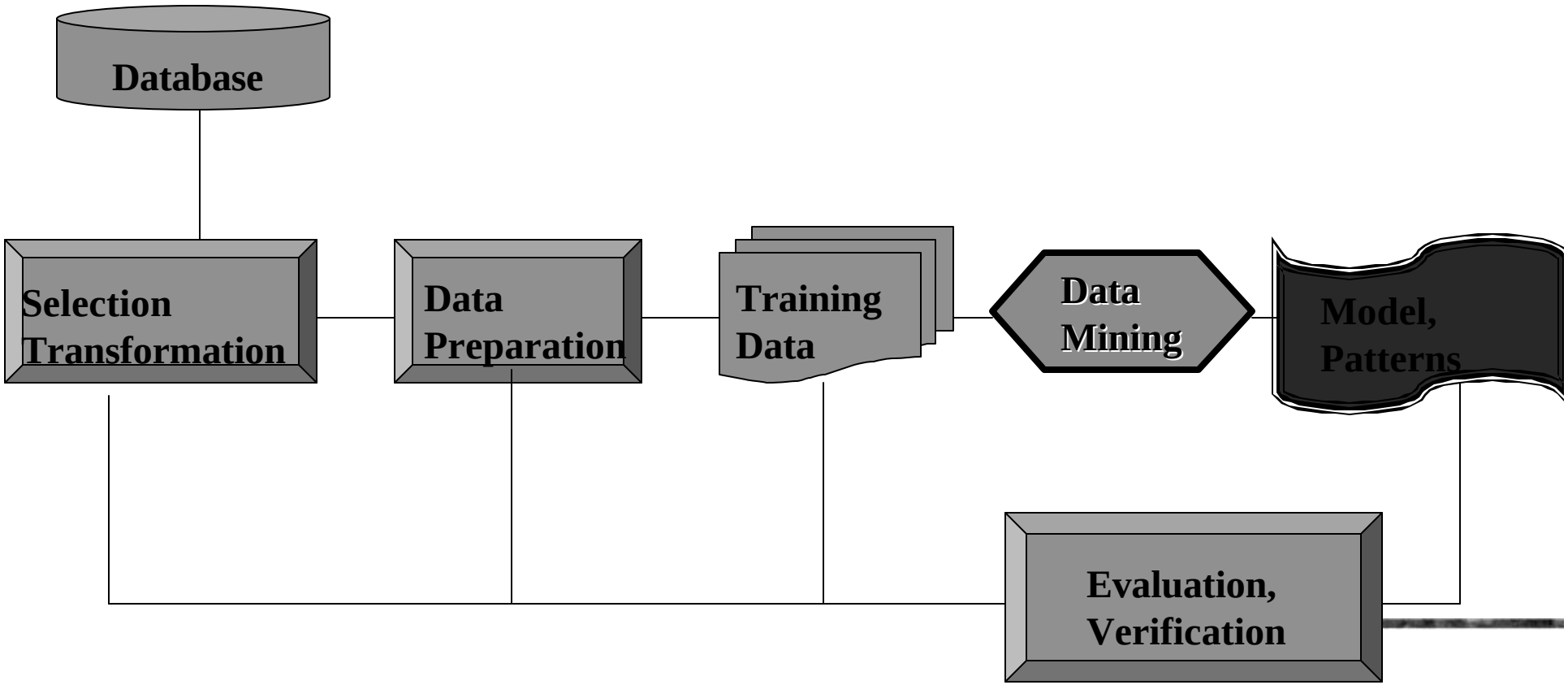
Multidisciplinary Field



Multidisciplinary Field

- Database technology
- Artificial Intelligence
- Machine Learning
- Neural Networks
- Statistics
- Pattern recognition
- Knowledge-based systems/acquisition
- High-performance computing
- Data visualization

KDD Process


Database
Selection
Transformation
Data
Preparation
Training
Data
Data
Mining
Model,
Patterns
Evaluation,
Verification

Steps of a KDD Process

- Learning the application domain:
  - relevant prior knowledge and goals of application
- Creating a target data set: data selection
- Data cleaning and preprocessing: (may take 60% of effort!)
- Data reduction and transformation:
  - Find useful features, dimensionality/variable reduction, representation
- Choosing functions of data mining
  - summarization, classification, regression, association, clustering
- Choosing the mining algorithm(s)
- Data mining: search for patterns of interest
- Pattern evaluation and knowledge presentation
  - visualization, transformation, removing redundant patterns, etc.
- Use of discovered knowledge

LEARNING ALGORITHMS

- Fundamental idea:

learn rules/patterns/relationships automatically from the data

Data Mining Tasks

- Exploratory Data Analysis
- Predictive Modeling: Classification and Regression
- Descriptive Modeling
  - Cluster analysis/segmentation
- Discovering Patterns and Rules
  - Association/Dependency rules
  - Sequential patterns
  - Temporal sequences
- Deviation detection

TAXONOMY

- **Predictive Methods**
  - *Use some variables to predict some unknown or future values of other variables*
- **Descriptive Methods**
  - *Find human –interpretable patterns that describe the data*

TAXONOMY

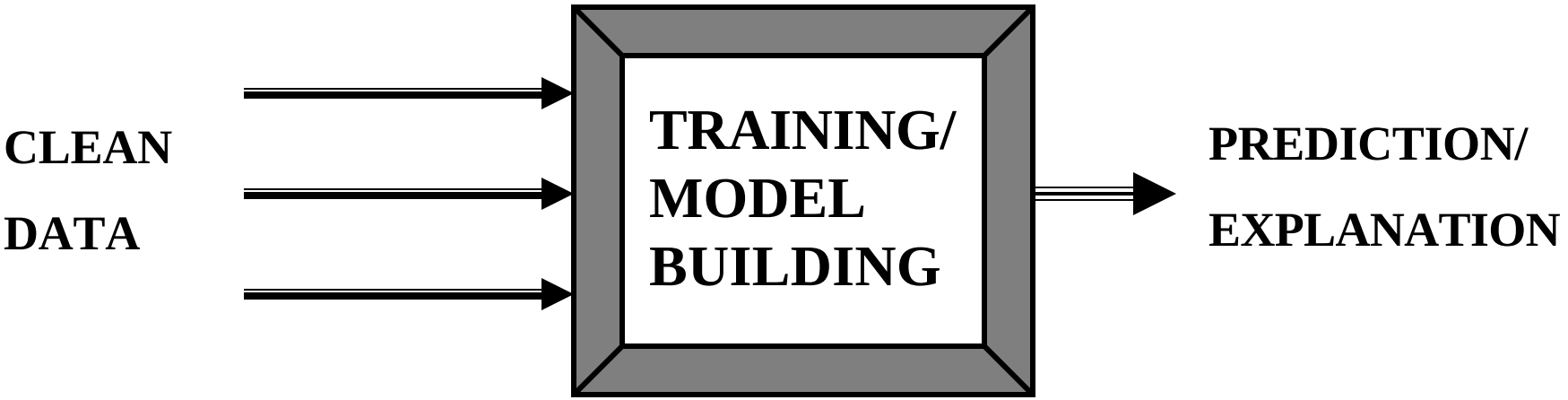
- **Supervised vs. Unsupervised**
  - Labeled vs. unlabeled data

- Weka book page 9: The weather data set
- Weka book page 15: The CPU performance data

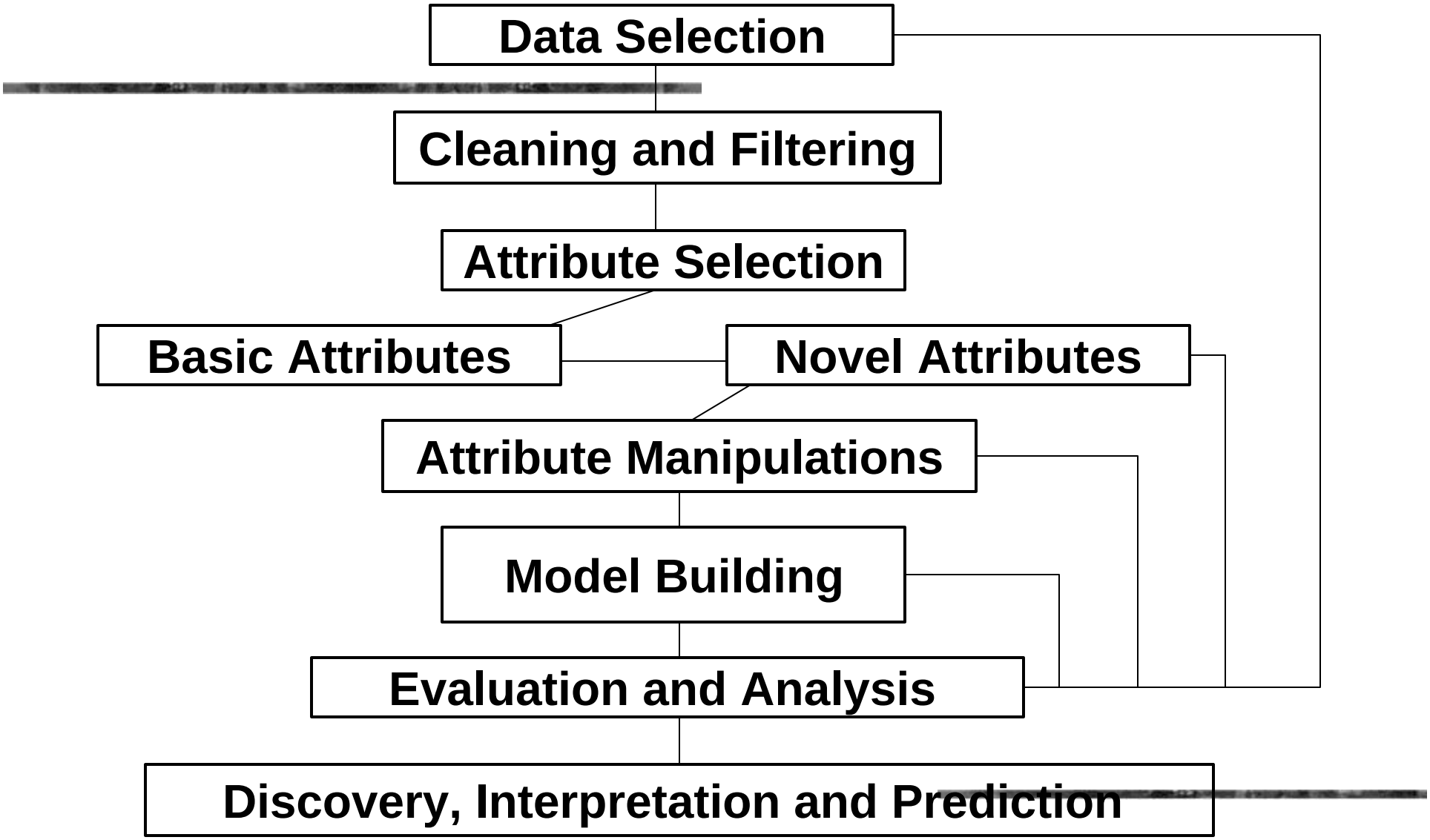
PREDICTIVE MODELING OVERVIEW

- Data Preparation:
  - Data Selection
  - Cleaning
  - Filtering
  - Attribute Selection
  - Attribute Manipulations
- Model Building:
  - classification, clustering, connectionist models
- Model Analysis and Interpretation
- Pattern Discovery and Data Prediction

MODELING PROCESS



PREDICTIVE MODELING



DATA MINING CHALLENGES

- Computationally expensive to investigate all possibilities
- Dealing with noise/missing information and errors in data
- Choosing appropriate attributes/input representation
- Finding the minimal attribute space
- Finding adequate evaluation function(s)
- Extracting meaningful information
- Not overfitting

LEARNING AND MODELING METHODS

- Decision Tree Induction
- Regression Tree Induction
- Multi-variate Regression Tree
- Clustering
  - K-means, EM, Cobweb
- Neural network
  - Backpropagation
  - Recurrent
- Various other models

Jargon

- Collection of records – **training set**
  - Each record contains a set of attributes (class)
- **Model** for the class attribute is a function of the values of other attributes
- Previously unseen data – assigned to the appropriate class
- **Test set** – determines the accuracy of the model on unseen data

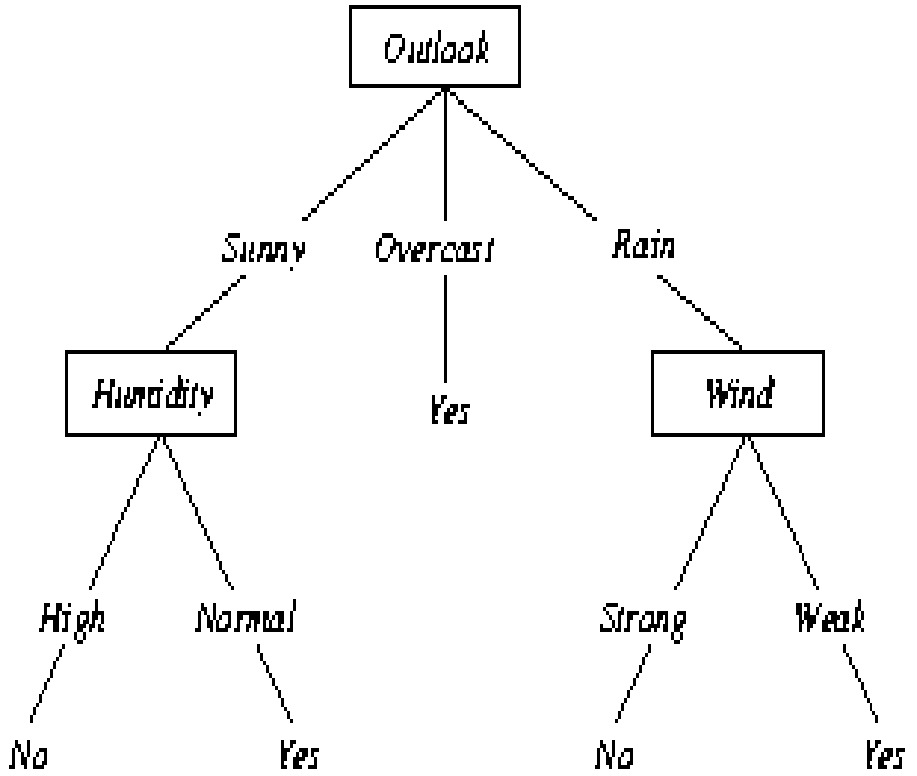
DECISION TREE INDUCTION

- Method for approximating discrete-valued functions
  - robust to noisy/missing data
  - can learn non-linear relationships
  - inductive bias towards shorter trees

DECISION TREE FOR THE CONCEPT

"Play Tennis"

Day	Outlook	Temp	Humidity	Wind	PlayTennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcas	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcas	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcas	Mild	High	Strong	Yes
D13	Overcas	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

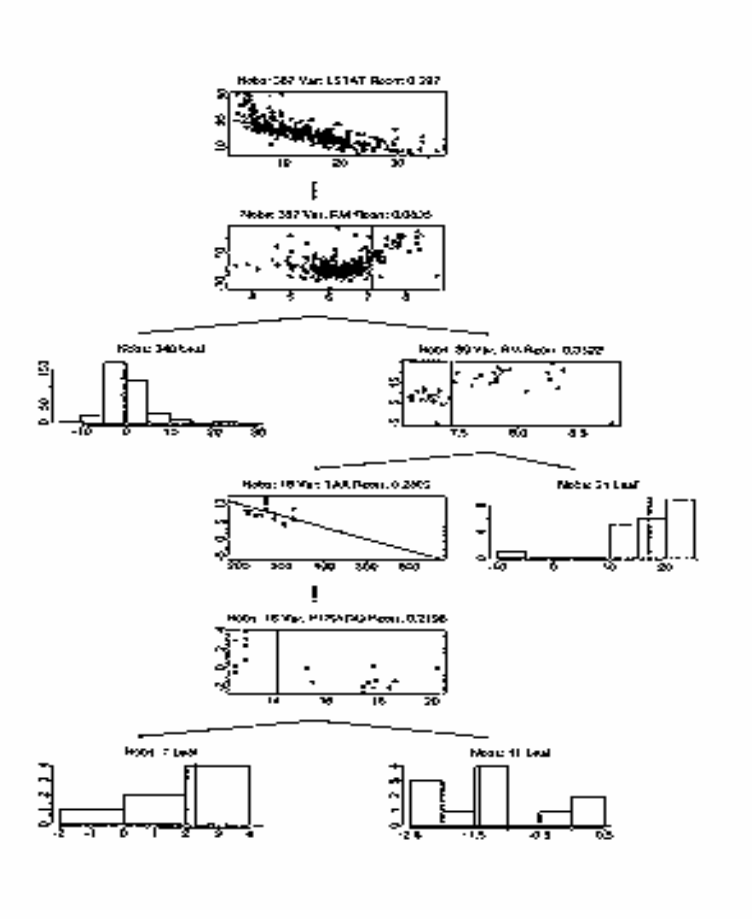


[Mitchell,1997]

REGRESSION TREE INDUCTION

- Why Regression tree?
  - Ability to:
    - Predict continuous variable
    - Model conditional effects
    - Model uncertainty

Regression Trees



- Continuous goal variables
- Induction by means of an efficient recursive partitioning algorithm
- Uses linear regression to select internal nodes

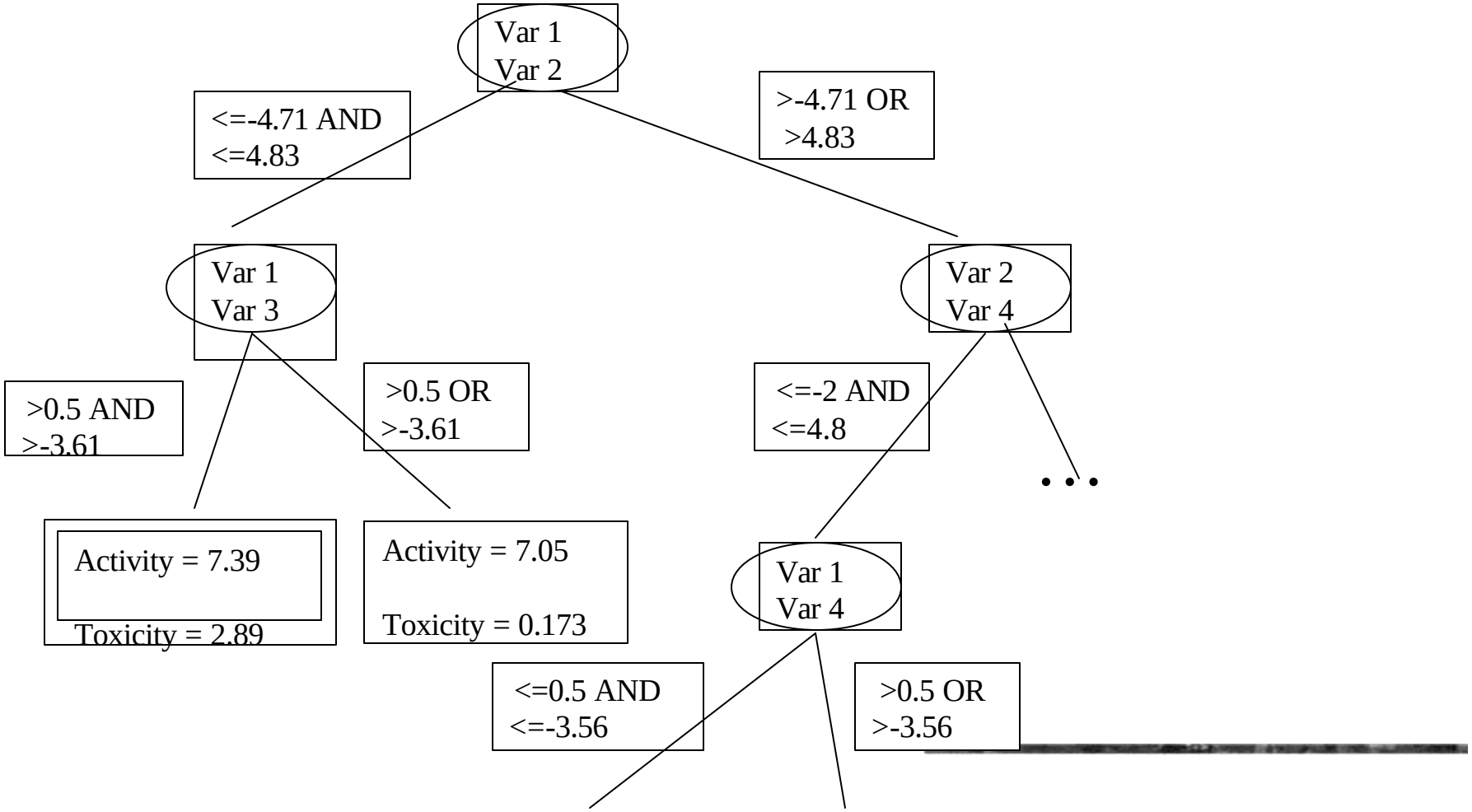
Quinlan, 1992

MULTI-VARIATE REGRESSION TREES*

- All the characteristics of a regression tree
- Capable of predicting two or more outcomes
- Example:
  - Activity and toxicity, monetary gain and time

Balac, Gaines ICML 2001

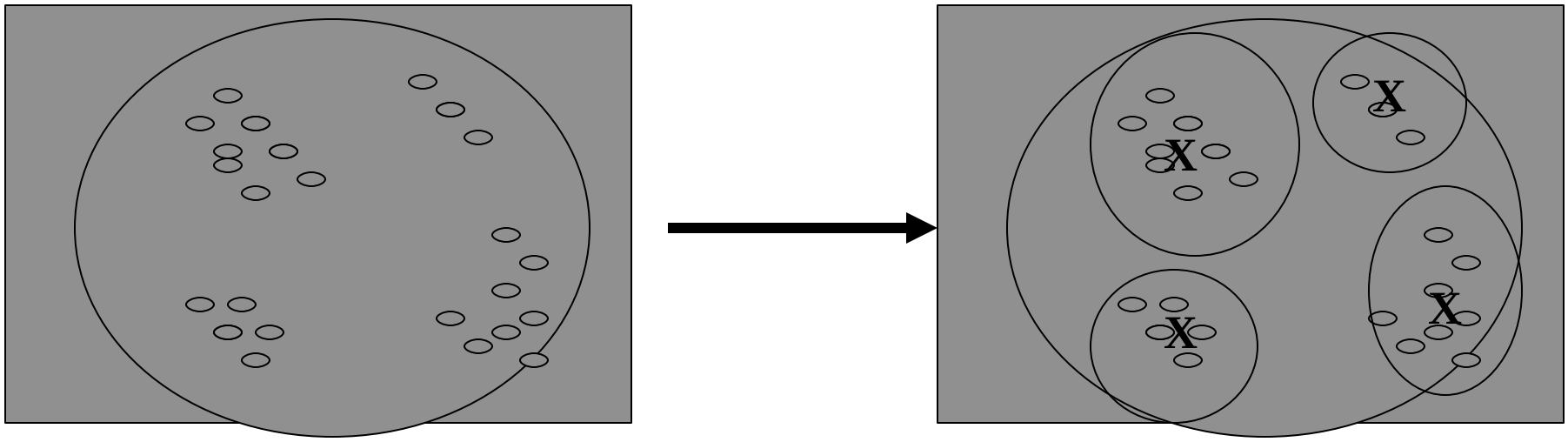
MULTI-VARIATE REGRESSION TREE INDUCTION



CLUSTERING

- **Basic idea: Group similar things together**
- **Unsupervised Learning – Useful when no other info is available**
- **K-means**
  - Partitioning instances into *k* disjoint clusters
  - Measure of similarity

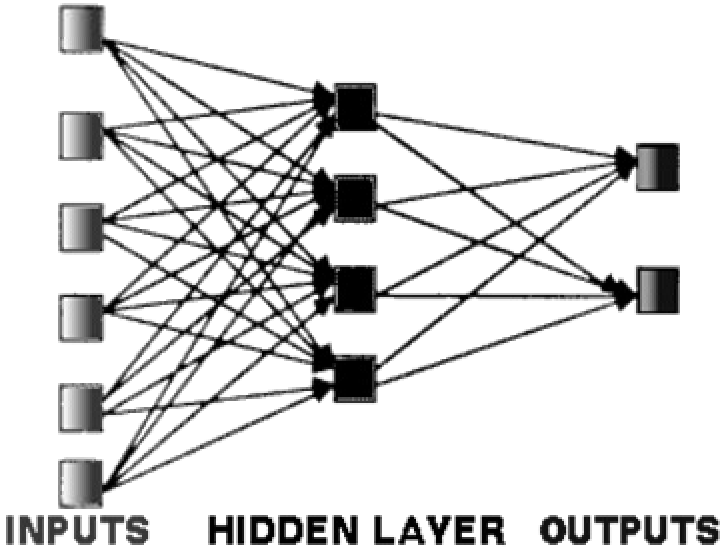
CLUSTERING



Clustering Techniques

- **Hierarchical clustering**
- **Incremental clustering**
- **Bayesian clustering**

ARTIFICIAL NEURAL NETWORKS (ANNs)



- Network of many simple units
- Main Components
  - Inputs
  - Hidden layers
  - Outputs
- Adjusting weights of connections
- Backpropagation

ANN Example

- Train data: 64 different triplets
- Input: 4 nucleotide for each 3 position of codon
- Output: corresponding 20 amino acids

Are Our Brains Computers?

- Human brain is merely a very complex computer?
- The human brain is definitely not the type of computer with which most people are familiar
- Computers found in our homes and offices are *serial*
  - they only do one computation at a time
  - execute sequences of computations very quickly

Are Our Brains Computers?

- To perform even the simplest functions a serial computer requires hundreds or thousands of computations
  - Can be very fast (trillions of computations per second)
- They have limitations
- Very complex activity such as driving a car through a large city during rush hour

Brain vs. Serial Computer

- Our neurons do not transmit information very fast
- Even for a reflex that only involves three neurons, it may take several tenths of a second
- Somehow, we are able to process the information that it takes to drive a car using our relatively slow neurons

Human Brain

- Our nervous system does not process serially
- Human brain is an example of a *massively-parallel system*
  - rather than executing computations one-by-one in series
  - the human brain makes numerous computations simultaneously when performing a neural process

A Little Bit Of History

- At the beginning of the computer age
  - problems that men couldn't solve
  - men were too slow in solving
- Computer was invented
  - to automate some solutions of problems that were causing trouble to humans
- Calculating equations to resolve important physical problems, etc.

Limitations

- But now computers and standard programming became limiting
- Can't easily/quickly perform:
  - Data mining
  - Pattern recognition
  - Voice recognition
  - Image processing

Standard Computers Are

Good at

- Fast arithmetic
- Doing precisely what the programmer programs them to do

Not so good at

- Interacting with noisy data or data from the environment
- Massive parallelism
- Fault tolerance
- Adapting to circumstances

Can neural network systems help?

- Where?
  - we can't formulate an algorithmic solution
  - we *can* get lots of examples of the behavior we require
  - we need to discover the structure from existing data

Neural Network (NN)

- Neuronal network - Intellectual abstraction which would work mostly like human brain works
- Famous NN topology called BPN - Backward propagation network
- The BPN succeeded in solving very complicated problems such as pattern recognition in a very noisy environment

What is a neural network (NN)?

- ”Artificial Neural Network" (ANN) vs. Neural Network
- Biological neural networks are much more complicated than the mathematical models we use for ANNs
  - Customary to drop the "A" or the "artificial”
- There is no universally accepted definition of an NN

NN Definition

- NN is a network of many simple processors ("units"), each possibly having a small amount of local memory
- The units are connected by communication channels ("connections") which usually carry numeric data of various kinds
- The units operate only on their local data and on the inputs they receive via the connections

About Neural Networks

- Some NNs are models of biological neural networks and some are not
- Historically, much of the inspiration for the field of NNs came from the desire to produce artificial systems capable of sophisticated, "intelligent", computations similar to those that the human brain routinely performs
- Possibly to enhance our understanding of the human brain

About Neural Networks

- Most NNs have some sort of "training" rule
  - weights of connections are adjusted based on data
- NNs "learn" from examples
  - as children learn to distinguish dogs from cats based on examples of dogs and cats
- NNs are capable of exhibiting capability for generalization beyond the training data
  - to produce approximately correct results for new, unseen cases that were not used for training

About Neural Networks

- NNs normally have great potential for parallelism
  - Computations of the components are largely independent of each other
- Are massive parallelism and high connectivity defining characteristics of NNs?
- Such requirements rule out various simple models
  - simple linear regression - a minimal feed-forward net with only two units plus bias

Sample of the definitions

- According to the *DARPA Neural Network Study* (1988, AFCEA International Press, p. 60):
  - ... a neural network is a system composed of many simple processing elements operating in parallel whose function is determined by network structure, connection strengths, and the processing performed at computing elements or nodes.

Sample of the definitions

- According to Haykin (1994), p. 2:
  - A neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
    - Knowledge is acquired by the network through a learning process.
    - Interneuron connection strengths known as synaptic weights are used to store the knowledge.

Sample of the definitions

- According to Nigrin (1993), p. 11:
  - A neural network is a circuit composed of a very large number of simple processing elements that are neurally based. Each element operates only on local information. Furthermore each element operates asynchronously; thus there is no overall system clock.

Sample of the definitions

- According to Zurada (1992), p. xv:
  - Artificial neural systems, or neural networks, are physical cellular systems which can acquire, store, and utilize experiential knowledge.

Sample of the definitions

- Jochen Fröhlich
  - A neural net is an artificial representation of the human brain that tries to simulate its learning process
  - The term "artificial" means that neural nets are implemented in computer programs that are able to handle the large number of necessary calculations during the learning process

Neural nets origin - The human brain

- Consists of more than a billion neural cells that process information
- Each cell works like a simple processor
  - the massive interaction between all cells and their parallel processing makes the brain's abilities possible
- On average neuron is connected to other neurons through about 10 000 **synapses**

Brain as an Information Processing System

- The brain's network of neurons forms a massively parallel information processing system vs. conventional computers
  - single processor executes a single series of instructions
- Time taken for each elementary operation:
  - neurons typically operate at a maximum rate of about 100 Hz
  - conventional CPU carries out several hundred million machine level operations per second

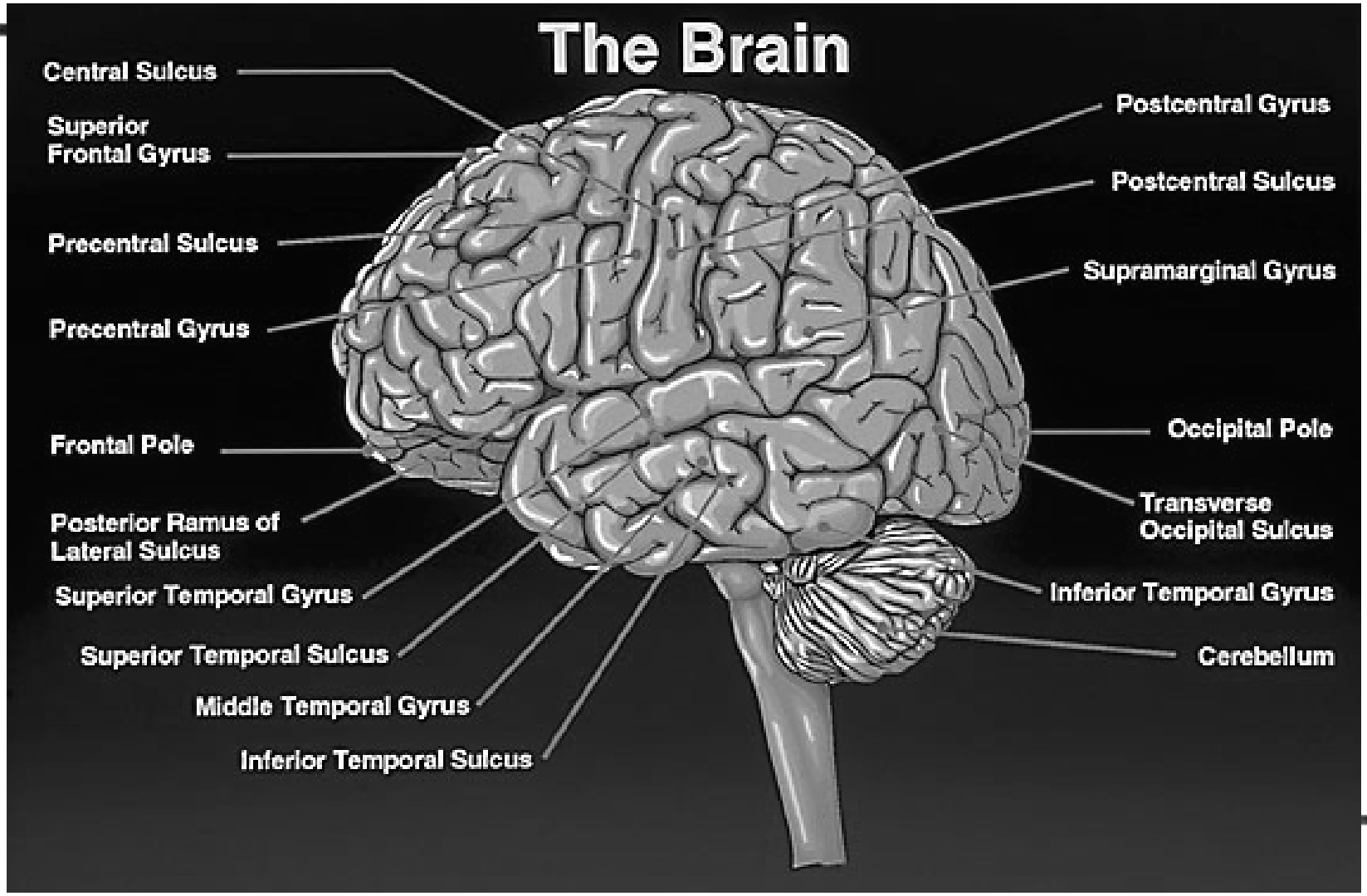
Brain as an Information Processing System

- Despite of being built with very slow hardware, the brain has quite remarkable capabilities
  - its performance tends to degrade gracefully under partial damage
- In contrast most programs and engineered systems are brittle
  - if you remove some arbitrary parts, very likely the whole will cease to function previously carried out by the damaged areas

Brain as an Information Processing System

- It can learn (reorganize itself) from experience
- Partial recovery from damage is possible if healthy units can learn to take over the functions previously carried out by the damaged areas
- Performs massively parallel computations extremely efficiently
- It supports our intelligence and self-awareness
  - How? Nobody knows yet

Human Brain Is Not Homogeneous



Human Brain

- The overall pattern of **projections** (bundles of neural connections) between areas is extremely complex and only partially known
- The best mapped (and largest) system in the human brain is the visual system
  - where the first 10 or 11 processing stages have been identified

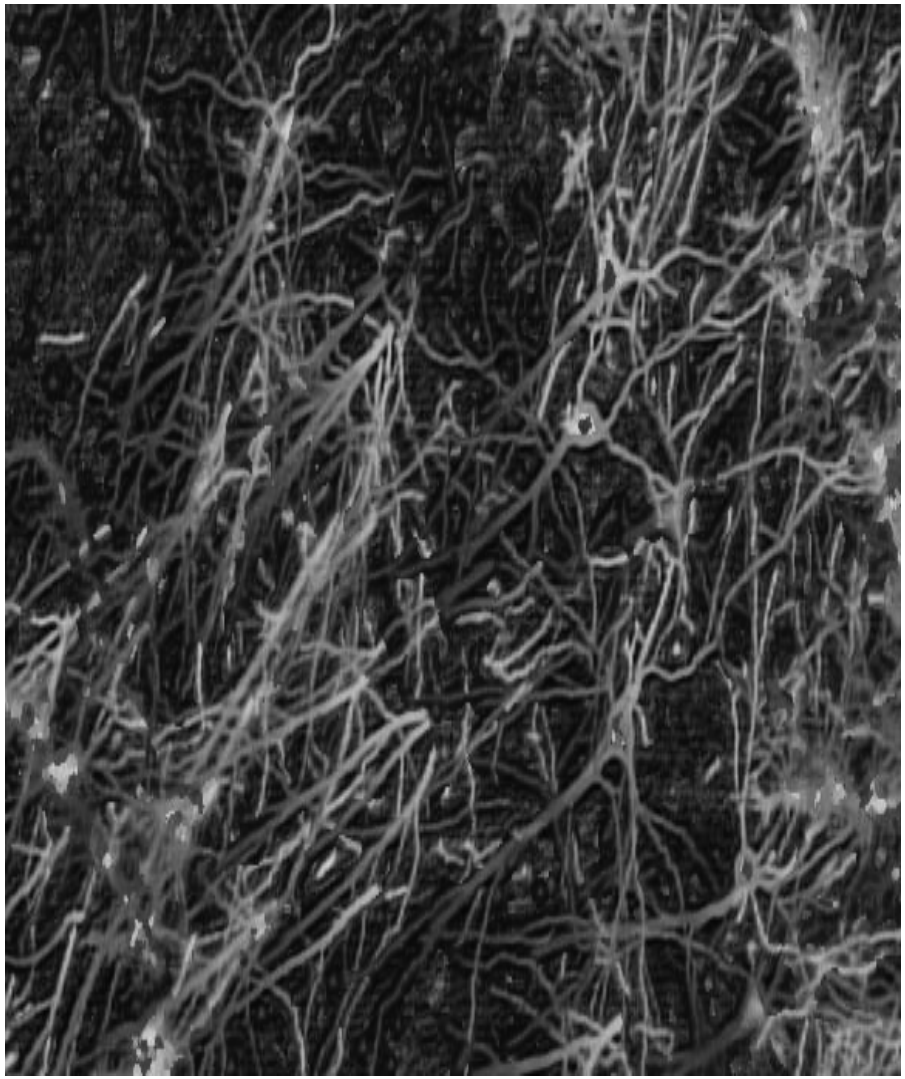
Connections

We distinguish:

- **feed-forward** projections that go from earlier processing stages (near the sensory input) to later ones (near the motor output)
- **feedback** connections that go in the opposite direction.

Neurons

- In addition to these long-range connections, neurons also link up with many thousands of their neighbors
- To form very dense, complex local networks



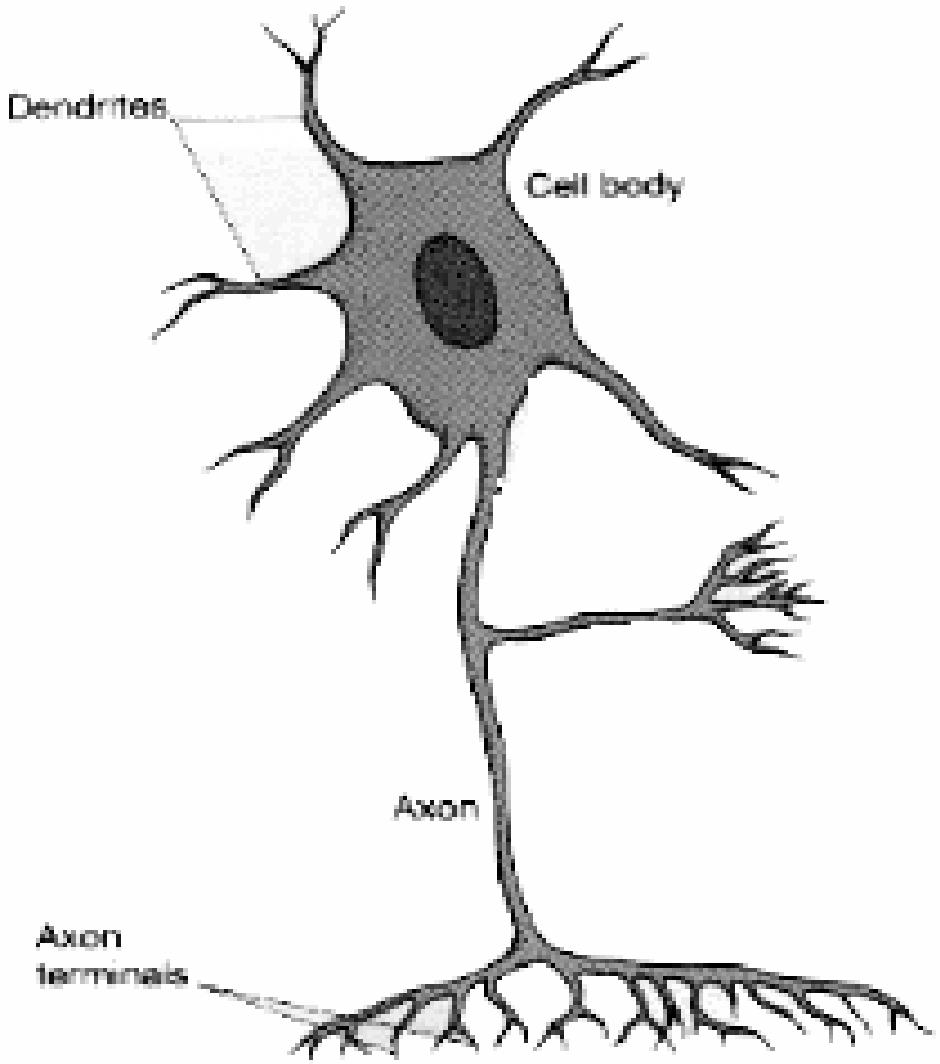
Human Brain

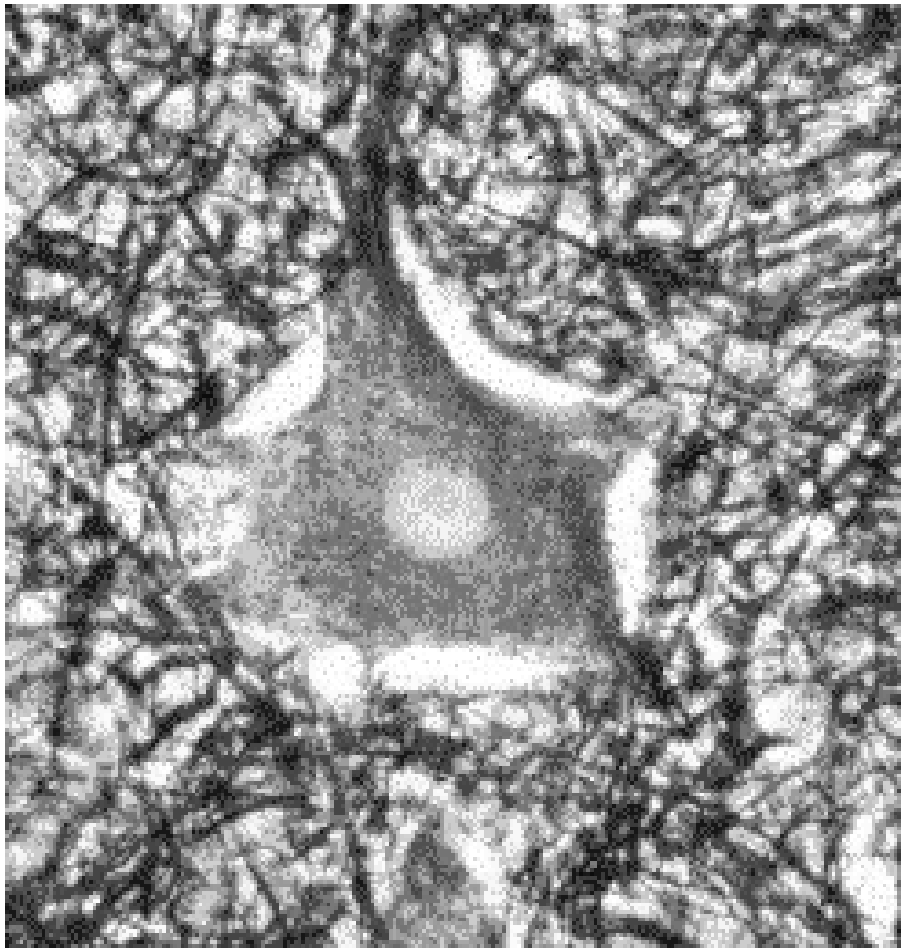
- Our brains are made up of about 100 billion tiny units called *neurons*
- Each neuron is connected to thousands of other neurons and communicates with them via electrochemical signals
- Signals coming into the neuron are received via junctions called *synapses*
- Synapses are located at the end of branches of the neuron cell called *dendrites*

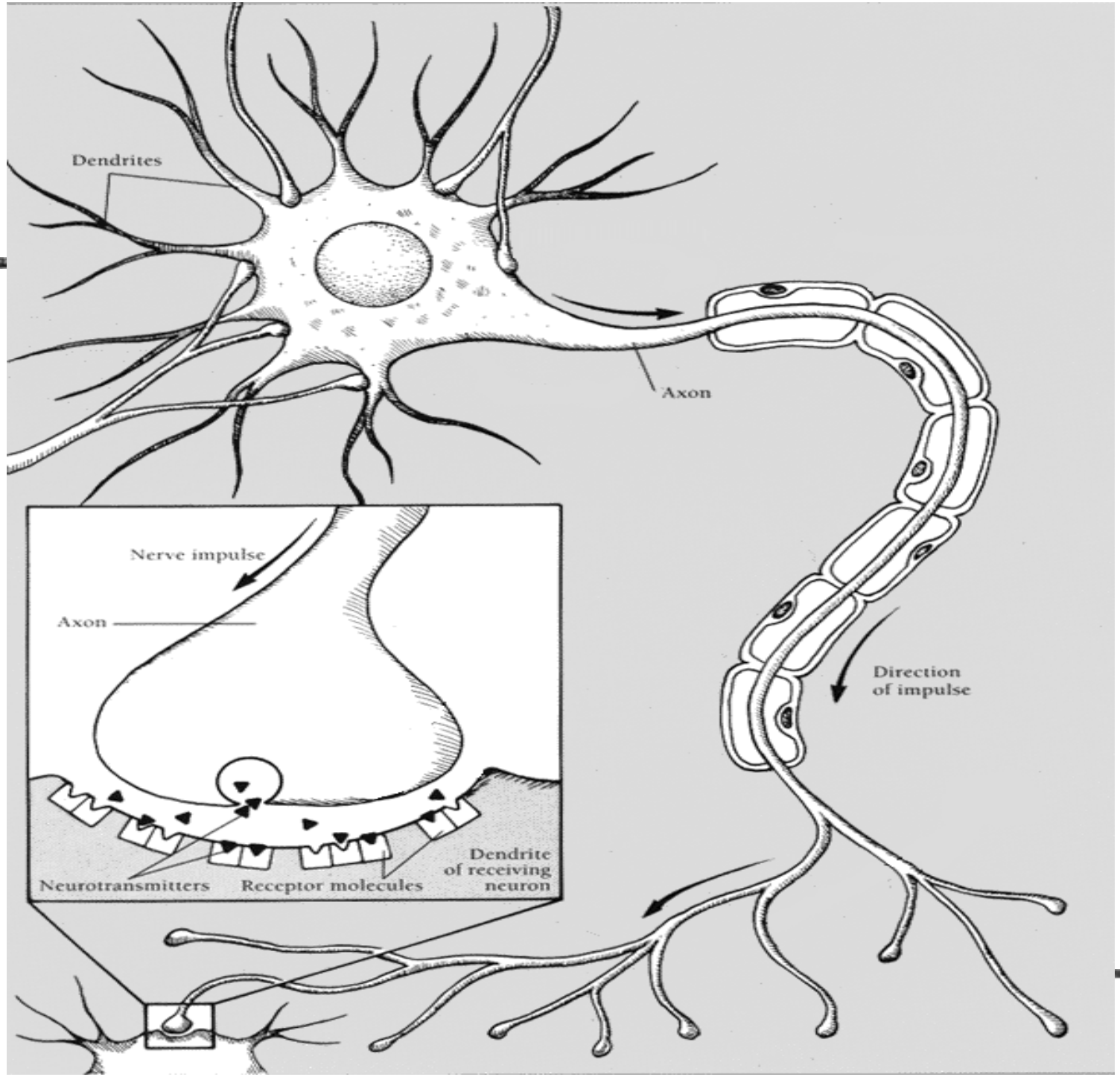
Human Brain

- The neuron continuously receives signals from inputs and then performs a little bit of “magic”
- Neuron sums up the inputs to itself (in some way) and then, if the end result is greater than some threshold value, the neuron fires
- It generates a voltage and outputs a signal along an *axon*

A sketch a neural cell or neuron



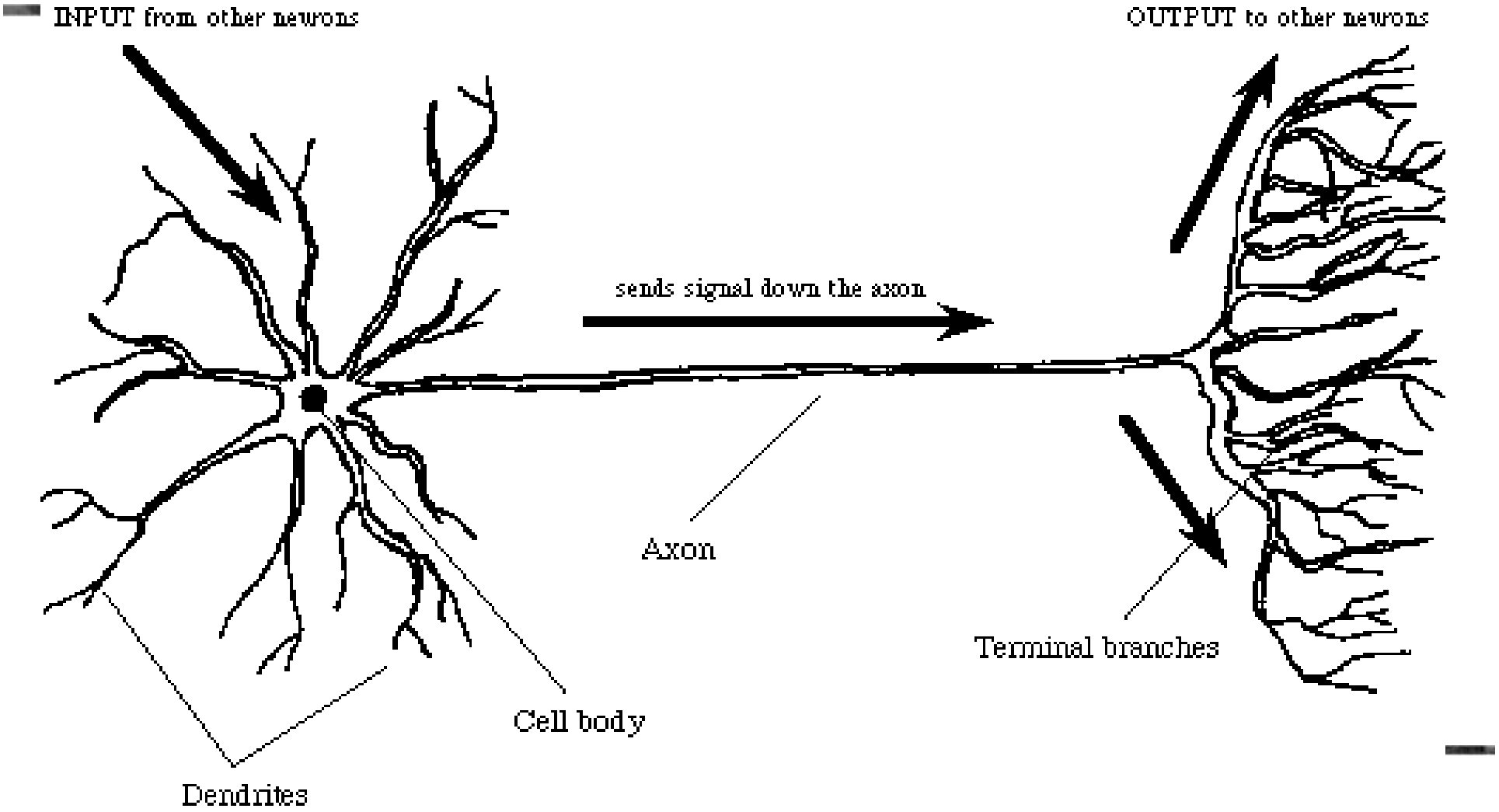



Dendrites
Axon
Direction
of impulse
Nerve impulse
Axon
Neurotransmitters
Receptor molecules
Dendrite
of receiving
neuron

The neuron

- Two important parts
  - **synapse** and the **dentrites**
- Dentrites are extensions of a neuron which connect to other neurons to form a neural network
- Synapses are a gateway which connects to dentrites that come from other neurons

Neuron


INPUT from other neurons
OUTPUT to other neurons
sends signal down the axon
Axon
Terminal branches
Cell body
Dendrites

Biological Neuron

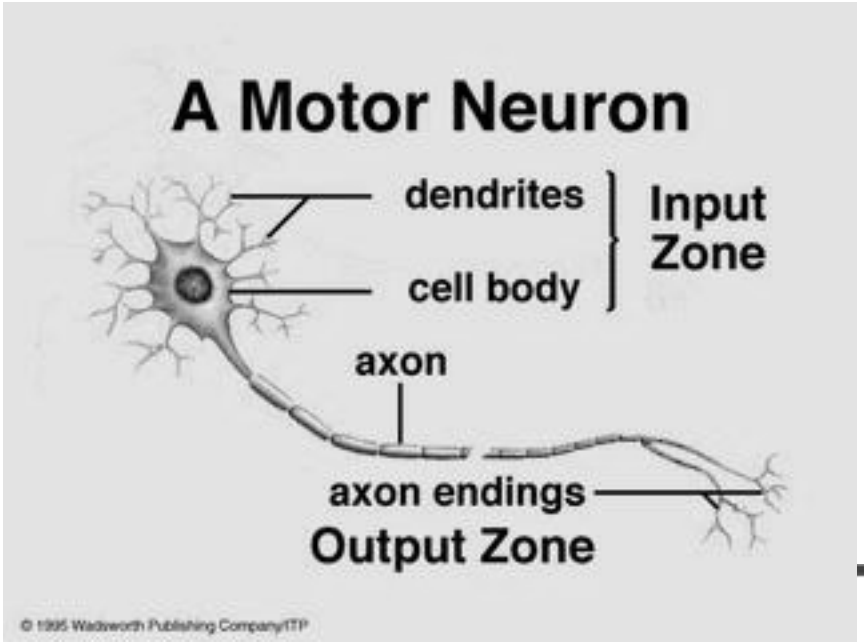
- A biological neuron may be
  - connected to other neurons
  - accepting connections from other neurons
- Through those connections
  - electrical pulses are transmitted
  - information is carried in the timing and the frequency with which these pulses are emitted

More About Neurons

- Neuron receives information from other neurons, processes it and then relays this information to other neurons
- What form does this processing take?
- Clearly the neuron must generate some kind of output based on the cumulative input
- Neuron integrates the pulses that arrive and when this integration exceeds a certain limit, neuron in turn emits a pulse

Neurons and Synapses

- The basic computational unit in the nervous system is the nerve cell, or **neuron**. A neuron has:
- Dendrites (inputs)
- Cell body
- Axon (output)



Neurons and Synapses

- A neuron receives input from other neurons (typically many thousands)
- Inputs sum (approximately)
- Once input exceeds a critical level, the neuron discharges a **spike** - an electrical pulse that travels from the body, down the axon, to the next neuron(s) (or other receptors)

Neurons and Synapses

- This spiking event is also called **depolarization** and is followed by a **refractory period**
  - during which the neuron is unable to fire
- The axon endings (Output Zone) almost touch the dendrites or cell body of the next neuron
- Transmission of an electrical signal from one neuron to the next is effected by

Neurons And Synapses

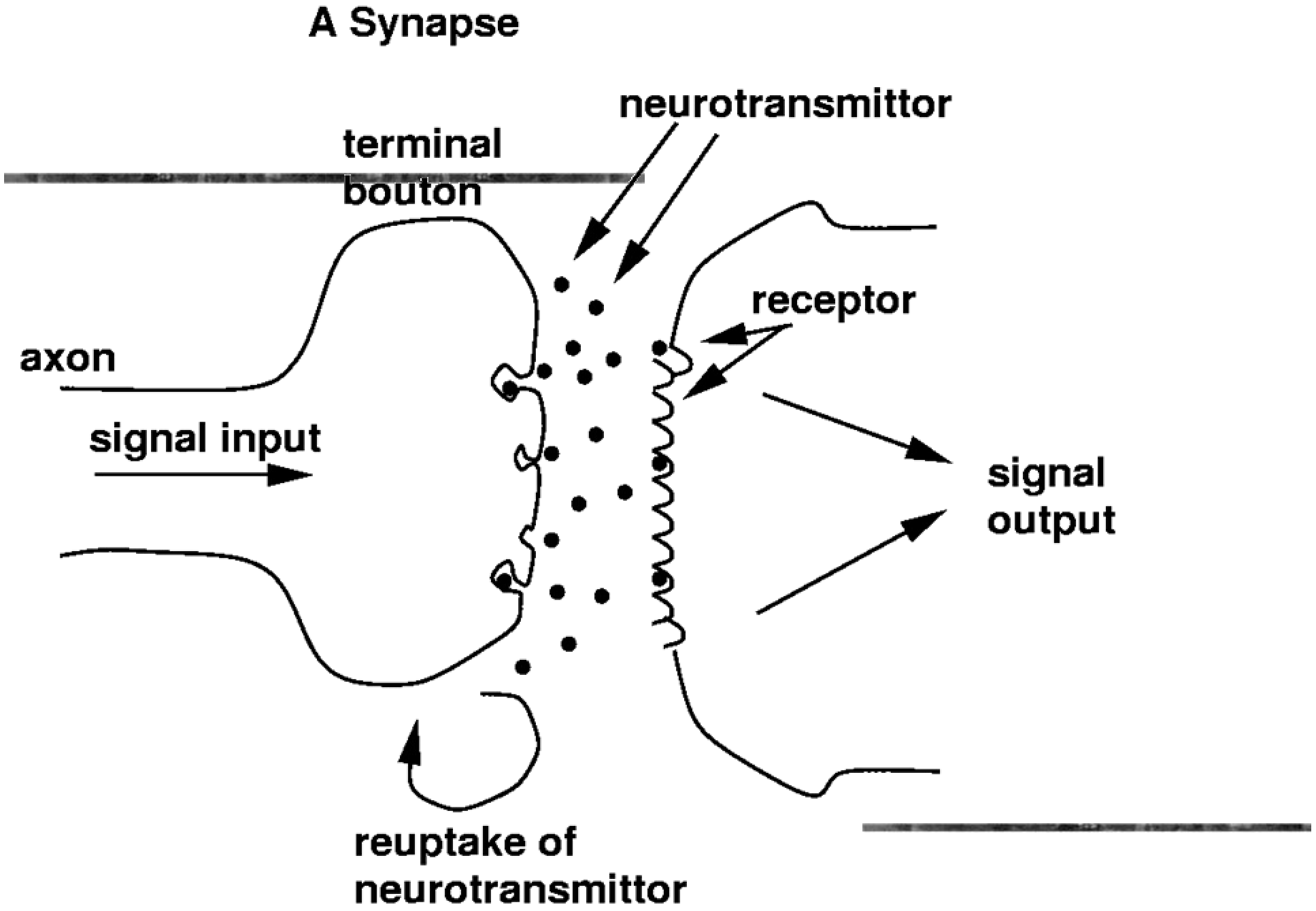
- **Neurotransmitters** are chemicals which are released from the first neuron and which bind to receptors in the second
- This link is called a **synapse**
- The extent to which the signal from one neuron is passed on to the next depends on many factors
  - e.g. the amount of neurotransmitter available, the number and arrangement of receptors, amount of neurotransmitter reabsorbed, etc.

Brains Learn - Of Course How?

- One way brains learn is by altering the strengths of connections between neurons and by adding or deleting connections between neurons
- They learn "on-line”
  - based on experience
  - and typically without the benefit of a teacher

Synaptic Learning

- The efficacy of a synapse can change as a result of experience
- Providing both memory and learning through **long-term potentiation**
- One way this happens is through release of more neurotransmitter
- Many other changes may also be involved


A Synapse
neurotransmittor
terminal
bouton
receptor
axon
signal input
signal
output
reuptake of
neurotransmittor

Important Properties of Nervous System

- Parallel, distributed information processing
- High degree of connectivity among basic units
- Connections are modifiable based on experience
- Learning is a constant process, and usually unsupervised
- Learning is based only on local information
- Performance degrades gracefully if some units are removed

Artificial Neuron Models

- Computational neurobiologists have constructed very elaborate computer models of neurons in order to run detailed simulations of particular circuits in the brain
- We are more interested in the general properties of neural networks, independent of how they are actually "implemented" in the brain

Artificial Neuron Models

- This means that we can use much simpler, abstract "neurons", which (hopefully) capture the essence of neural computation even if they leave out much of the details of how biological neurons work
- People have implemented model neurons in hardware as electronic circuits, often integrated on VLSI chips

Artificial Neuron Models

- Remember that computers run much faster than brains!
- We can therefore run fairly large networks of simple model neurons as software simulations in reasonable time
- This has obvious advantages over having to use special "neural" computer hardware

A Simple Artificial Neuron

- Basic computational element (model neuron) is often called a **node** or **unit**
- It receives input from some other units, or perhaps from an external source
- Each input has an associated **weight** *w*, which can be modified so as to model synaptic learning

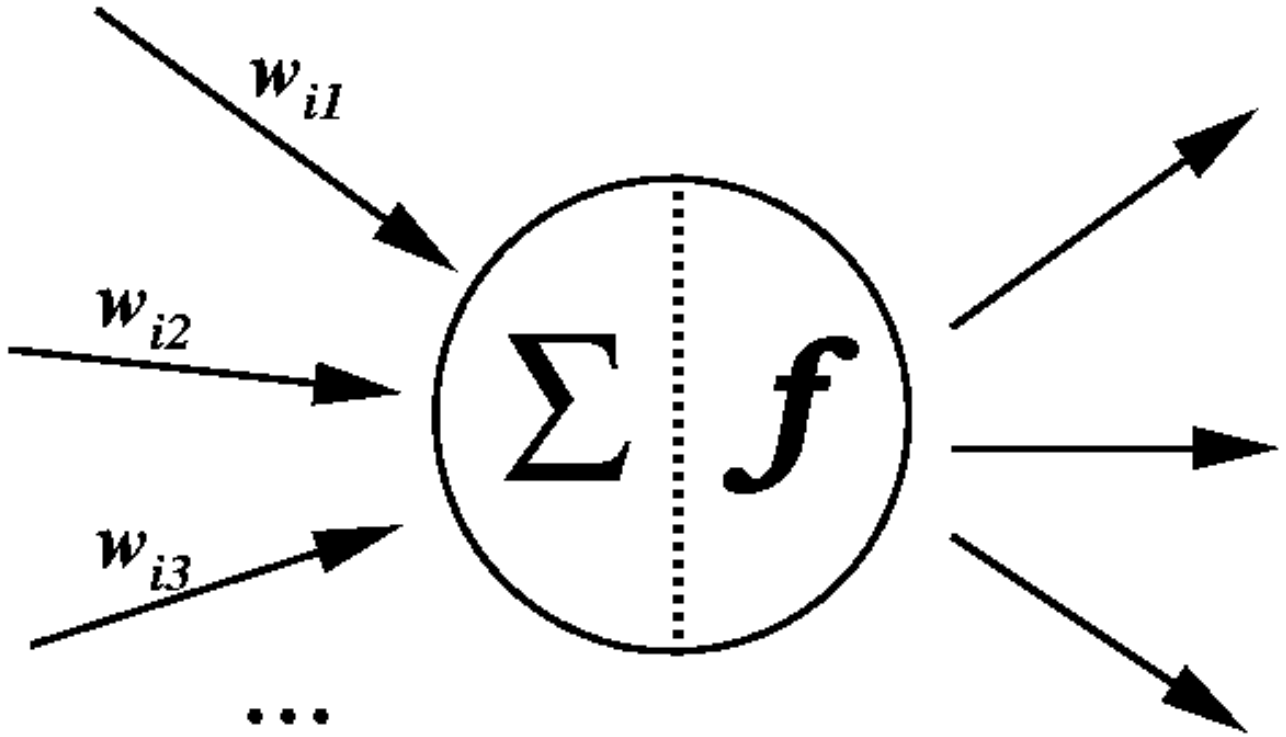
A Simple Artificial Neuron

- The unit computes some function *f* of the weighted sum of its inputs:

$$y_i = f\left(\sum_j w_{ij}\, y_j\right)$$

- Its output, in turn, can serve as input to other units

Linear unit

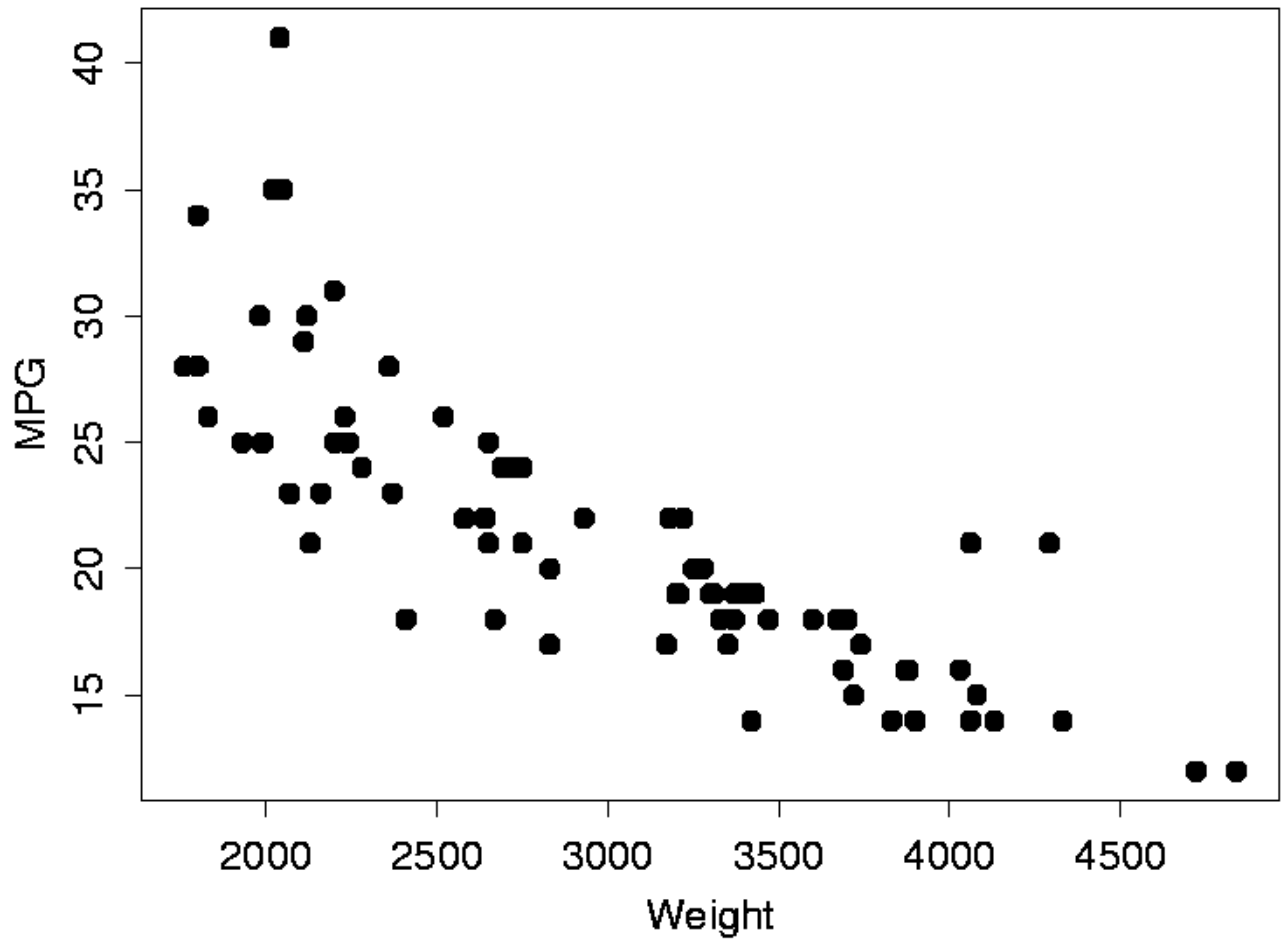


$$y_i = f(net_i)$$

Activation Function

- Identity Function
- Step Function
- Logistic Function (Sigmoid)
- Symmetric Sigmoid
- Radial Basis Functions
- Derivatives

Linear Regression



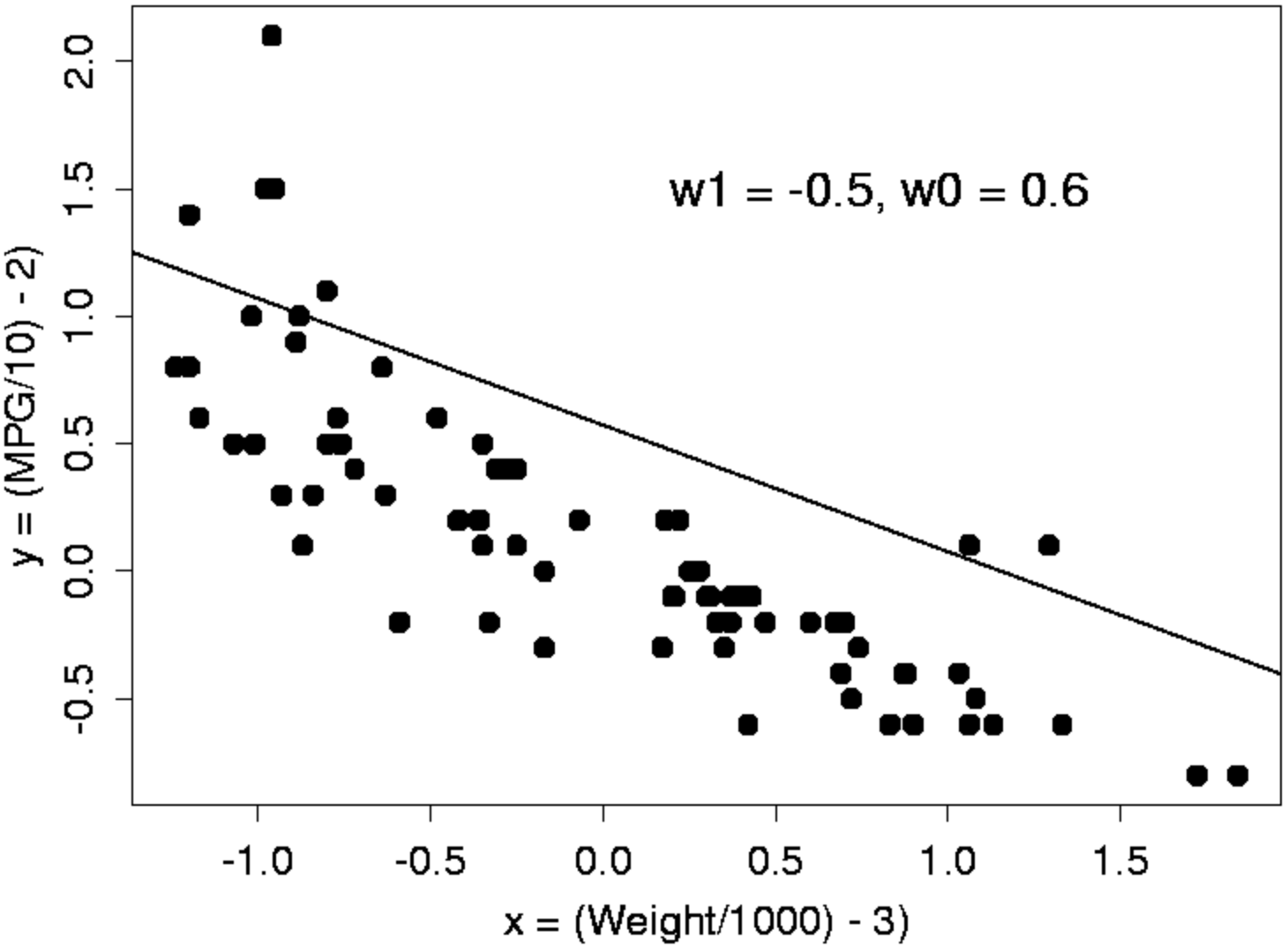
Example

- Each dot in the figure provides information about the weight
  - x-axis, units: U.S. pounds and fuel consumption
  - y-axis, units: miles per gallon for one of 74 cars
  - Clearly weight and fuel consumption are linked
    - heavier cars use more fuel.

Example

- We are given the weight of a 75th car and asked to predict how much fuel it will use based on the given data
- Such questions can be answered by using a **model**
  - a short mathematical description of the data
- The simplest useful model here is of the form

$$y = w_1 x + w_0$$


w1 = -0.5, w0 = 0.6
y = (MPG/10) - 2)
x = (Weight/1000) - 3)
2.0
1.5
1.0
0.5
0.0
-0.5
-1.0
-0.5
0.0
0.5
1.0
1.5

Regression Example

- How do we choose the two parameters w_0 and w_1 of our model?
- Any straight line drawn somehow through the data could be used as a predictor
- Some lines will do a better job than others
- Is the line in previous slide a good model?
  - for most cars it will predict too much fuel consumption for a given weight

The Loss Function

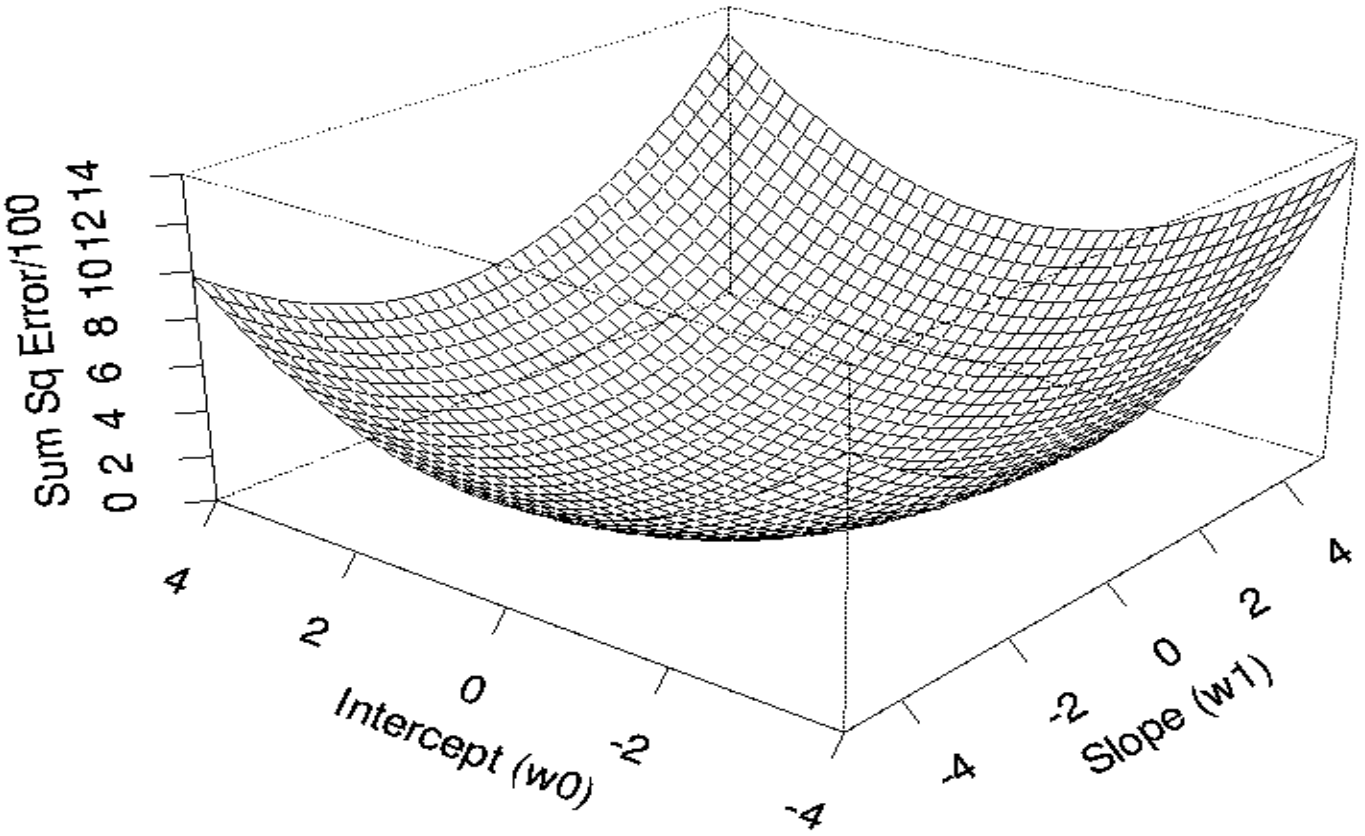
- In order to make precise what we mean by being a "good predictor”
- Define a **loss** (**objective** or **error**) function *E* over the model parameters
- Popular choice for *E* is the **sum-squared error**

$$E = \frac{1}{2}\sum_{p}(t_p - y_p)^2$$

$$E = \frac{1}{2} \sum_p (t_p - y_p)^2$$

- Sum over all points *i* in our data set of the squared difference between
  - **target** value t_i (actual fuel consumption) and
  - the model's prediction y_i, calculated from the input value x_i (weight of the car)
    - $y = w_1 x + w_0$
  - For a linear model, the sum-sqaured error is a quadratic function of the model parameters

Figure showing E for a range of values of w_0 and w_1



Minimizing the Loss

- The loss function *E* provides an objective measure of predictive error for a specific choice of model parameters
- We can thus restate our goal of finding the best (linear) model as finding the values for the model parameters that minimize *E*

Gradient Descent

- For linear models
  - **linear regression** provides a direct way to compute these optimal model parameters
- Does not generalize to **nonlinear** models
- Even though the solution cannot be calculated explicitly in that case
  - the problem can still be solved by an iterative numerical technique called **gradient descent**

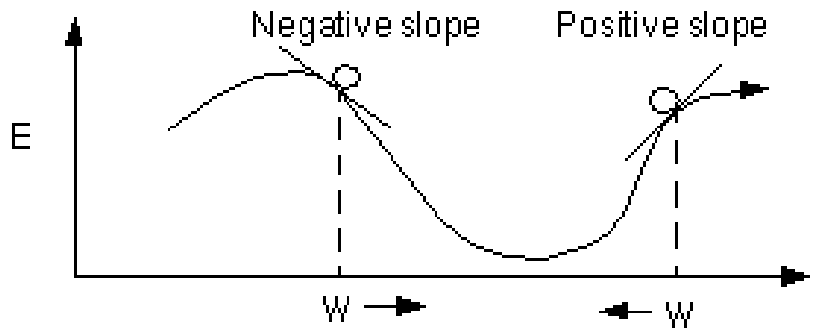
Gradient Descent

- Choose some (random) initial values for the model parameters
- Calculate the gradient G of the error function with respect to each model parameter
- Change the model parameters so that we move a short distance in the direction of the greatest rate of decrease of the error, i.e., in the direction of -G
- Repeat steps 2 and 3 until G gets close to zero

How does this work?

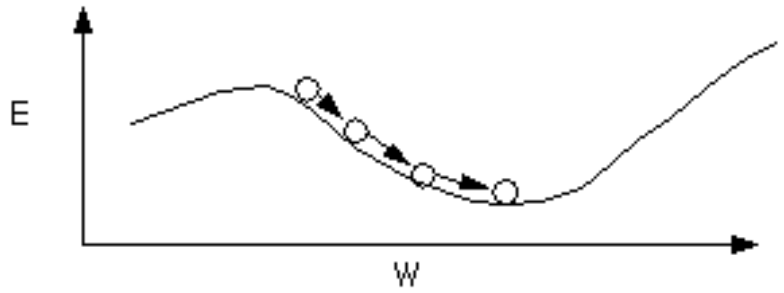
- The gradient of E gives us the direction in which the loss function at the current setting of the *w* has the steepest **slope**
- In order to decrease *E* take a small step in the opposite direction or *-G*

Slope of E positive
=> decrease W
Slope of E negative
=> increase W

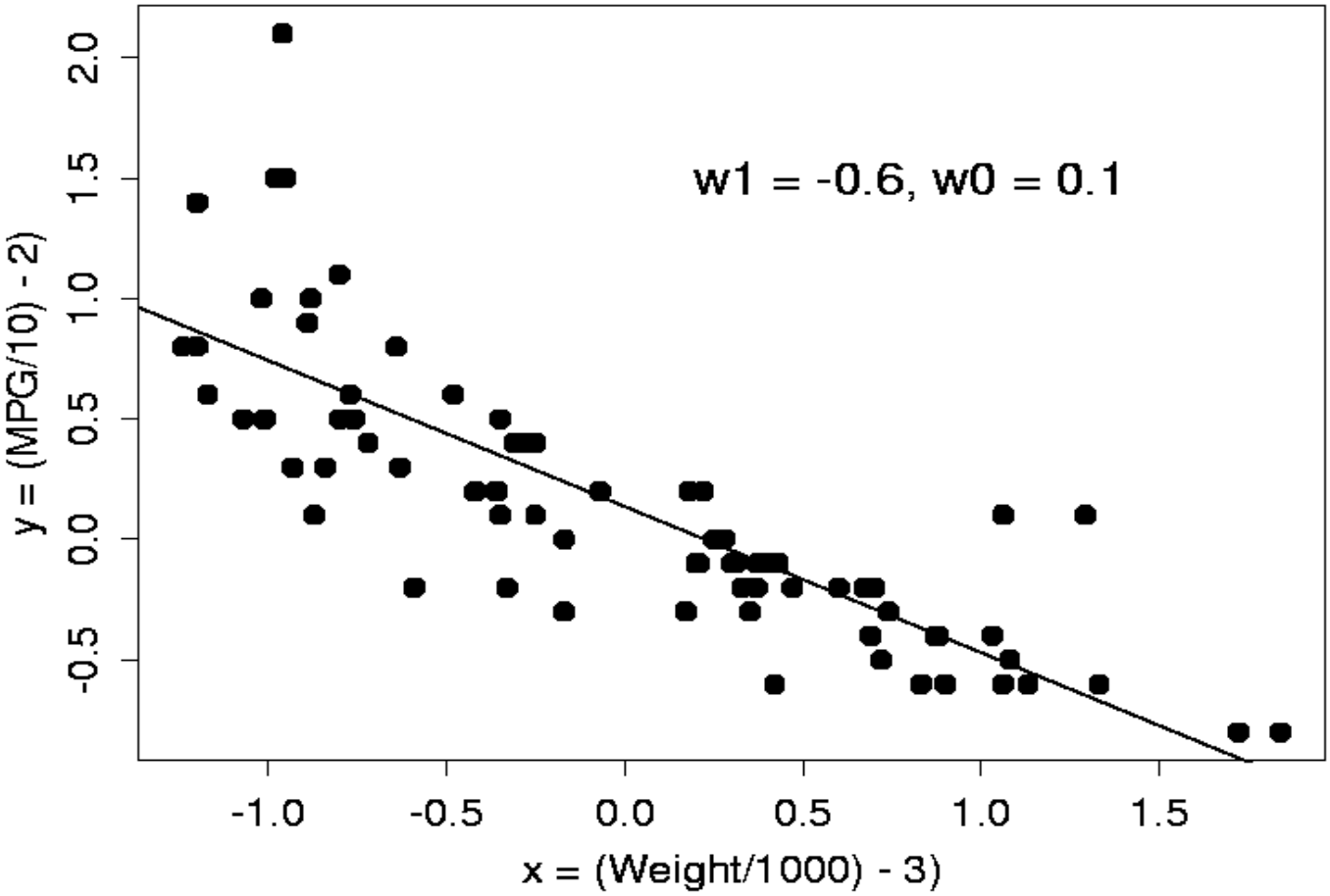


Gradient Descent

- By repeating this over and over, we move "downhill" in *E* until reach a minimum where $G = 0$
- so that no further progress is possible



The best linear model for car data found by gradient decent



It's a neural network!

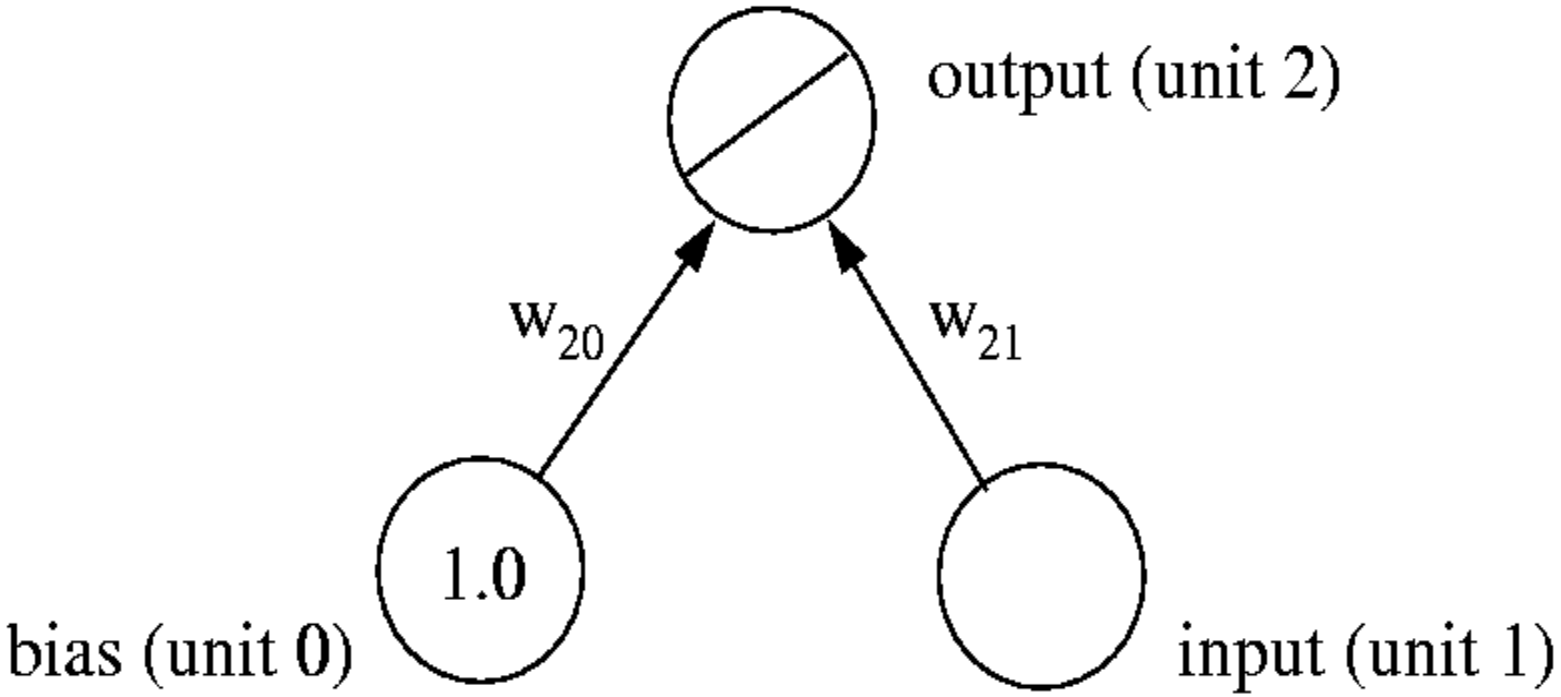
- Linear model of equation $y = w_1 x + w_0$ can be implemented by the simple neural network
  - It consists of a
    - **bias** unit
    - an **input** unit
    - a linear **output** unit
- The input unit makes external input x (the weight of a car) available to the network
- While the bias unit always has a constant output of 1

It's a neural network

- The output unit computes the sum
  - $y_2 = y_1 w_{21} + 1.0 w_{20}$
- This is equivalent to the previous equation

  $y = w_1 x + w_0$
  - with w_{21} implementing the slope of the straight line
  - and w_{20} its intercept with the y-axis

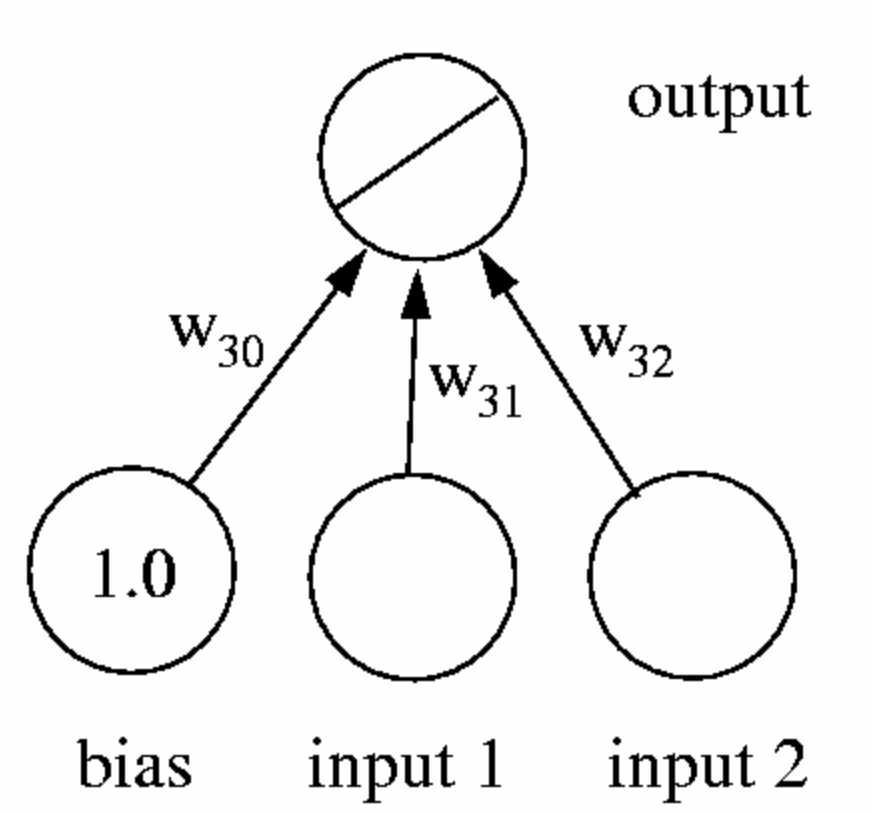
Simple Neural Network



Multiple regression

- Car example showed how we could discover an optimal linear function for predicting one variable (fuel consumption) from one other (weight)
- What if we are also given one or more additional variables which could be useful as predictors?
- Simple neural network model can be extended by adding more input units

Multiple inputs

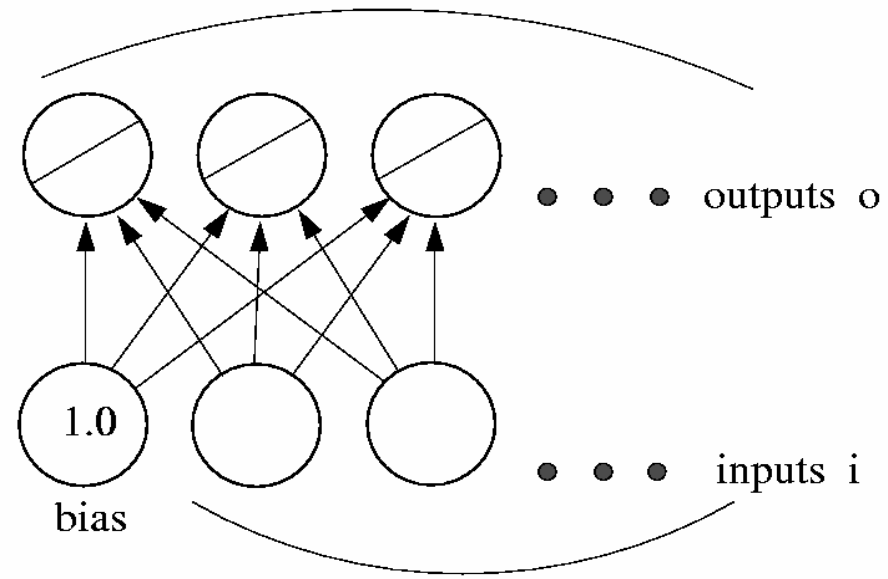


Multiple Outputs

- To predict more than one variable from the given data
  - Adding more output units
- The loss function for a network with multiple outputs is obtained simply by adding the loss for each output unit together

Network Structure

- The network now has a typical layered structure
  - a layer of input units (and the bias)
  - connected by a layer of weights to
  - a layer of output units



Computing the gradient

- Compute the gradient *G* of the loss function with respect to each weight w_{ij} of the network
- How a small change in that weight will affect the overall error *E*
- We begin by splitting the loss function into separate terms for each point *p* in the training data:

$$E = \sum_p E^p, \qquad E^p = \frac{1}{2} \sum_o (t_o^p - y_o^p)^2$$

Computing the gradient

- Since differentiation and summation are interchangeable, we can split the gradient into separate components for each training point:

$$G = \frac{\partial E}{\partial w_{ij}} = \frac{\partial}{\partial w_{ij}} \sum_p E^p = \sum_p \frac{\partial E^p}{\partial w_{ij}}$$

Computing the gradient

- Computation of the gradient for a single data point
  - omitting the superscript *p* in order to make the notation easier to follow
- First use the **chain rule** to decompose the gradient into two factors:

$$\frac{\partial E}{\partial w_{oi}} = \frac{\partial E}{\partial y_o} \frac{\partial y_o}{\partial w_{oi}}$$

Computing the gradient

- The first factor can be obtained by differentiating $E = \sum_p E^p, \qquad E^p = \frac{1}{2}\sum_o (t_o^p - y_o^p)^2$

$$\frac{\partial E}{\partial y_o} = -(t_o - y_o)$$

- Using $y_o = \sum_j w_{oj} y_j$ the second factor becomes

$$\frac{\partial y_o}{\partial w_{oi}} = \frac{\partial}{\partial w_{oi}} \sum_j w_{oj} y_j = y_i$$

- Putting the pieces back together, we obtain

$$\frac{\partial E}{\partial w_{oi}} = -(t_o - y_o)\, y_i$$

Computing the gradient

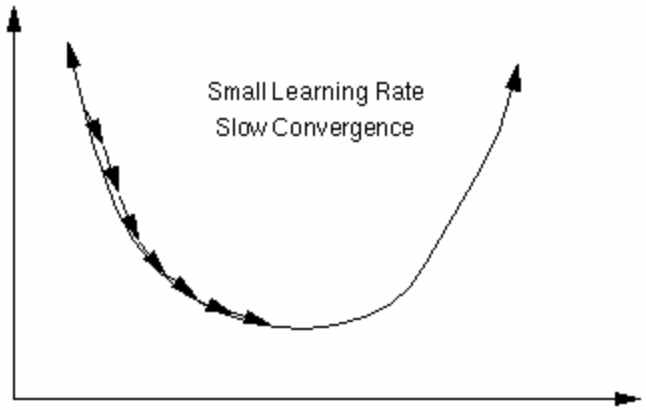
- To find the gradient *G* for the entire data set
- Sum at each weight the contribution given by equation $\frac{\partial E}{\partial w_{oi}} = -(t_o - y_o)\, y_i$ over all the data points
- We can then subtract a small proportion μ (called the **learning rate**) of *G* from the weights to perform gradient descent

The Gradient Descent Algorithm

- Initialize all weights with small random values
- REPEAT until done
  - For each weight w_{ij} set $\Delta w_{ij} := 0$
  - For each data point $(\mathbf{x}, \mathbf{t})^p$
    - set input units to **x**
    - compute value of output units
    - For each weight w_{ij} set $\Delta w_{ij} := \Delta w_{ij} + (t_i - y_i)\, y_j$
  - For each weight w_{ij} set $w_{ij} := w_{ij} + \mu\, \Delta w_{ij}$

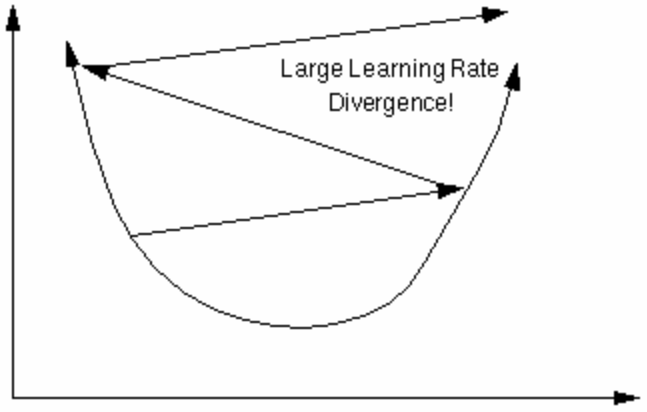
The Learning Rate

- Important consideration is the learning rate μ
  - determines by how much we change the weights w at each step
- If μ is too small the algorithm will take a long time to converge



The Learning Rate

- If μ is too large
  - we may end up bouncing around the error surface out of control - the algorithm **diverges**
  - This usually ends with an overflow error in the computer's floating-point arithmetic

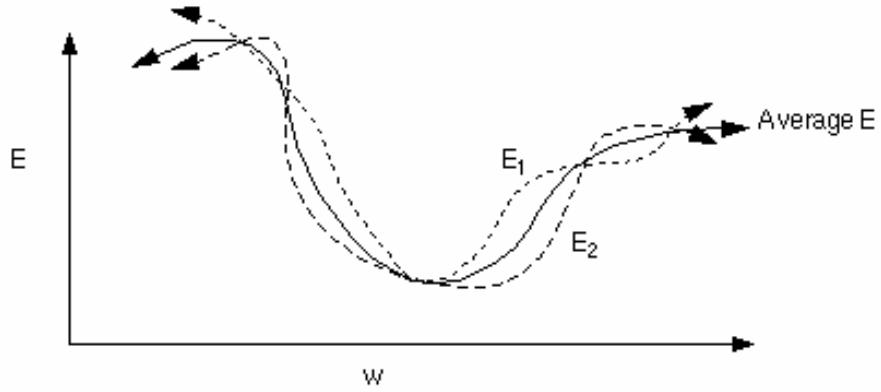


Batch vs. Online Learning

- **Batch** learning
  - Accumulate the gradient contributions for all data points in the training set before updating the weights
- **Online** learning
  - weights are updated immediately after seeing each data point

Stochastic Gradient Descent

- Since the gradient for a single data point can be considered a noisy approximation to the overall gradient *G*
  - this is also called **stochastic** (noisy) gradient descent



Online Learning Advantages

- Often much faster, especially when the training set is **redundant**
- Can be used when there is no fixed training set
- Better at tracking **non-stationary** environments
- Noise in the gradient can help to escape from **local minimum**

Online Learning Disadvantages

- Many powerful optimization techniques (conjugate and second-order gradient methods, support vector machines, Bayesian methods) are batch methods that cannot be used online
- A compromise between batch and online learning is the use of "mini-batches"
  - the weights are updated after every *n* data points
    - where *n* is greater than 1 but smaller than the training set size

Neural Networks Application

- Financial forecasting
- Business decision making
- Pattern recognition
- Behavior modeling
- Mechanical controls
- Character recognition
- Medical diagnosis
- Robotics

Neural Networks Application

- Machine learning:
  - Having a computer program itself from a set of examples so you don't have to program it yourself
    - Neural networks that learn from a set of examples
  - Optimization: given a set of constraints and a cost function, how do you find an optimal solution?
    - E.g. traveling salesman problem
  - Classification: grouping patterns into classes: i.e. handwritten characters into letters
  - Associative memory: recalling a memory based on a partial match
  - Regression: function mapping

Neural Networks Application

- in investment analysis:
  - to attempt to predict the movement of stocks currencies from previous data
- in signature analysis:
  - as a mechanism for comparing signatures made (e.g. in a bank) with those stored
  - one of the first large-scale applications of neural networks in the USA
  - one of the first to use a neural network chip

Neural Networks Application

- in process control:
  - most processes cannot be determined as computable algorithms
- in monitoring:
  - the state of aircraft engines
    - By monitoring vibration levels and sound, early warning of engine problems can be given
    - British Rail have also been testing a similar application monitoring diesel engines

Neural Networks Application

- in marketing:
  - used to improve marketing mailing
  - one technique is to run a test mailing, and look at the pattern of the returned responses
  - the idea is to find a predictive mapping from the data known about the clients to how they have responded
  - this mapping is then used to direct further mailings

Neural Networks Application

- Robotics
  - robot limbs movement
  - object recognition and movement
- Object identification
  - underwater listening
  - aircraft and small object identification

Neural Networks Application

- Speech recognition
  - NetTalk
  - Kohonen: Japanese and Finish in real time
- Flight simulators
  - for training new pilots on the ground
- Opening and closing valves on the rockets
- Ford Motor Company
  - NN that reads sensor data from automobile engines and determines probable cause of existing problems

Neural Networks Application

- Detection of medical phenomena
- Stock market prediction
- Credit assignment
- Monitoring the condition of machinery
- Engine management

Next Week

- Multi-layered Neural Networks
- Error
- Backpropagation
- **Classification**
- Perceptron learning
- Delta Learning
- Lab 1