Probability Distributions

Random Variable

- A random variable *X* takes on a defined set of values with different probabilities.
  - For example, if you roll a die, the outcome is random (not fixed) and there are 6 possible outcomes, each of which occur with probability one-sixth.
  - For example, if you poll people about their voting preferences, the percentage of the sample that responds "Yes on Proposition 100" is a also a random variable (the percentage will be slightly different every time you poll).
- Roughly, probability is how frequently we expect different outcomes to occur if we repeat the experiment over and over ("frequentist" view)

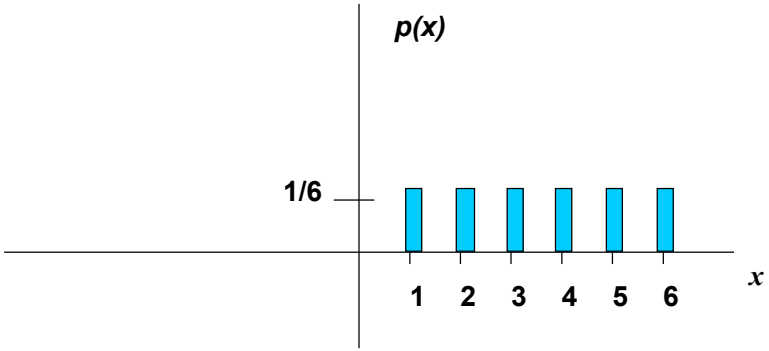
Random variables can be discrete or continuous

- **Discrete** random variables have a countable number of outcomes
  - Examples: Dead/alive, treatment/placebo, dice, counts, etc.
- **Continuous** random variables have an infinite continuum of possible values.
  - Examples: blood pressure, weight, the speed of a car, the real numbers from 1 to 6.

Probability functions

- A probability function maps the possible values of *x* against their respective probabilities of occurrence, *p(x)*
- *p(x)* is a number from 0 to 1.0.
- The area under a probability function is always 1.

Discrete example: roll of a die

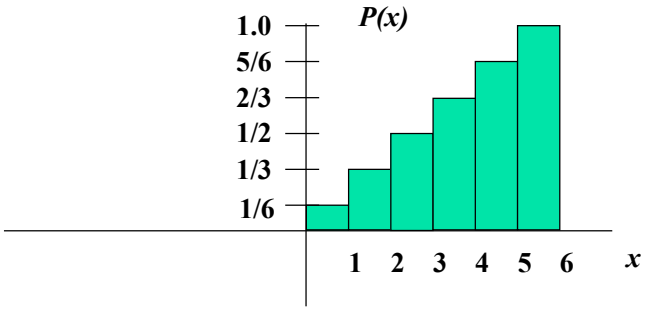


$$\sum_{\text{all } x} P(x) = 1$$

Probability mass function (pmf)

x	p(x)
1	p(x=1)=1/6
2	p(x=2)=1/6
3	p(x=3)=1/6
4	p(x=4)=1/6
5	p(x=5)=1/6
6	p(x=6)=1/6
	1.0

Cumulative distribution function (CDF)



Cumulative distribution function

x	P(x≤A)
1	P(x≤1)=1/6
2	P(x≤2)=2/6
3	P(x≤3)=3/6
4	P(x≤4)=4/6
5	P(x≤5)=5/6
6	P(x≤6)=6/6

Examples

1. What's the probability that you roll a 3 or less?
$P(x \leq 3) = 1/2$

2. What's the probability that you roll a 5 or higher?
$P(x \geq 5) = 1 - P(x \leq 4) = 1 - 2/3 = 1/3$

Practice Problem

Which of the following are probability functions?

a. $f(x) = .25$ for x=9,10,11,12

b. $f(x) = (3-x)/2$ for x=1,2,3,4

c. $f(x) = (x^2+x+1)/25$ for x=0,1,2,3

Answer (a)

a. $f(x) = .25$ for x=9,10,11,12

x	*f(x)*
9	.25
10	.25
11	.25
12	.25
	1.0

Yes, probability function!

Answer (b)

b. $f(x) = (3-x)/2$ for x=1,2,3,4

x	*f(x)*
1	(3-1)/2=1.0
2	(3-2)/2=.5
3	(3-3)/2=0
4	(3-4)/2=-.5

Though this sums to 1, you can't have a negative probability; therefore, it's not a probability function.

Answer (c)

c. $f(x) = (x^2+x+1)/25$ for x=0,1,2,3

x	f(x)
0	1/25
1	3/25
2	7/25
3	13/25
	24/25

Doesn't sum to 1. Thus, it's not a probability function.

Practice Problem:

- The number of times that Rohan wakes up in the night is a random variable represented by *x*. The probability distribution for *x* is:

x	1	2	3	4	5
P(x)	.1	.1	.4	.3	.1

Find the probability that on a given night:

a. He wakes exactly 3 times $p(x=3) = .4$

b. He wakes at least 3 times $p(x \geq 3) = (.4 + .3 + .1) = .8$

c. He wakes less than 3 times $p(x<3) = (.1 + .1) = .2$

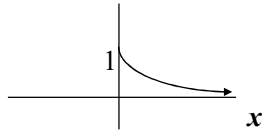
Important discrete distributions in epidemiology…

- Binomial (coming soon…)
  - Yes/no outcomes (dead/alive, treated/untreated, smoker/non-smoker, sick/well, etc.)
- Poisson
  - Counts (e.g., how many cases of disease in a given area)

Continuous case

- The probability function that accompanies a continuous random variable is a continuous mathematical function that integrates to 1.
  - For example, recall the negative exponential function (in probability, this is called an "exponential distribution"): $f(x) = e^{-x}$
  - This function integrates to 1:

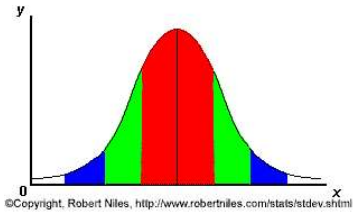
$$\int_0^{+\infty} e^{-x} = -e^{-x} \Big|_0^{+\infty} = 0 + 1 = 1$$



Review: Continuous case

- The normal distribution function also integrates to 1 (i.e., the area under a bell curve is always 1):

$$\int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = 1$$



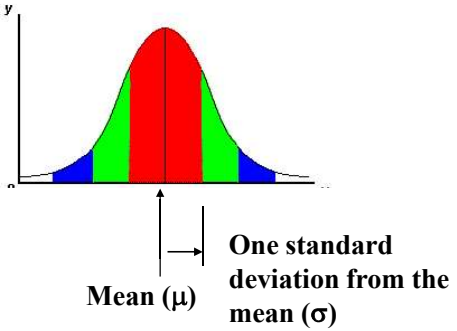
Review: Continuous case

- The probabilities associated with continuous functions are just areas under the curve (integrals!).
- Probabilities are given for a range of values, rather than a particular value (e.g., the probability of getting a math SAT score between 700 and 800 is 2%).

Expected Value and Variance

- All probability distributions are characterized by an expected value (=mean!) and a variance (standard deviation squared).

For example, bell-curve (normal) distribution:



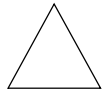
Expected value, or mean

- If we understand the underlying probability function of a certain phenomenon, then we can make informed decisions based on how we expect *x* to behave on-average over the long-run…(so called "frequentist" theory of probability).
- Expected value is just the weighted average or mean (μ) of random variable *x*. Imagine placing the masses *p*(*x*) at the points *X* on a beam; the balance point of the beam is the expected value of *x*.

Example: expected value

- Recall the following probability distribution of Rohan's waking pattern:

x	1	2	3	4	5
P(x)	.1	.1	.4	.3	.1



$$\sum_{i=1}^{5} x_i p(x) = 1(.1) + 2(.1) + 3(.4) + 4(.3) + 5(.1) = 3.2$$

Expected value, formally

Discrete case:

$$E(X) = \mu = \sum_{\text{all x}} x_i p(x_i)$$

Continuous case:

$$E(X) = \mu = \int_{\text{all x}} x_i p(x_i) dx$$

Sample Mean is a special case of Expected Value…

Sample mean, for a sample of n subjects: =

$$\overline{X} = \frac{\sum_{i=1}^{n} x_i}{n} = \sum_{i=1}^{n} x_i \left(\frac{1}{n}\right)$$

The probability (frequency) of each person in the sample is 1/n.

Variance/standard deviation

"The average (expected) squared distance (or deviation) from the mean"

$$\sigma^2 = Var(x) = E[(x-\mu)^2] = \sum_{\text{all x}} (x_i - \mu)^2 p(x_i)$$

***We square because squaring has better properties than absolute value. Take square root to get back linear average distance from the mean (="standard deviation").*

Variance, formally

Discrete case:

$$Var(X) = \sigma^2 = \sum_{\text{all x}} (x_i - \mu)^2 p(x_i)$$

Continuous case:

$$Var(X) = \sigma^2 = \int_{-\infty}^{\infty} (x_i - \mu)^2 p(x_i) dx$$

Sample variance is a special case…

The variance of a sample: s^2 =

$$\frac{\sum_{i=1}^{N} (x_i - \bar{x})^2}{n-1} = \sum_{i=1}^{N} (x_i - \bar{x})^2 \left(\frac{1}{n-1}\right)$$

Division by n-1 reflects the fact that we have lost a "degree of freedom" (piece of information) because we had to estimate the sample mean before we could estimate the sample variance.

Practice Problem

A roulette wheel has the numbers 1 through 36, as well as 0 and 00. If you bet $1.00 that an odd number comes up, you win or lose $1.00 according to whether or not that event occurs. If *X* denotes your net gain, *X*=1 with probability 18/38 and *X*= -1 with probability 20/38.

We already calculated the mean to be = -$.053. What's the variance of *X*?

Answer

$$\sigma^2 = \sum_{\text{all x}} (x_i - \mu)^2 p(x_i)$$

$$= (+1 - -.053)^2 (18/38) + (-1 - -.053)^2 (20/38)$$

$$= (1.053)^2 (18/38) + (-1 + .053)^2 (20/38)$$

$$= (1.053)^2 (18/38) + (-.947)^2 (20/38)$$

$$= .997$$

$$\sigma = \sqrt{.997} = .99$$

Standard deviation is $.99. Interpretation: On average, you're either 1 dollar above or 1 dollar below the mean, which is just under zero. Makes sense!

calculation formula!

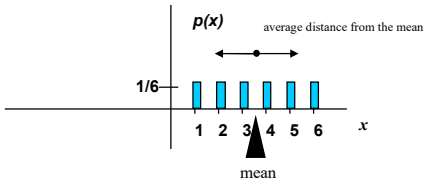
$$Var(X) = \sum_{\text{all x}} (x_i - \mu)^2 p(x_i) = \sum_{\text{all x}} x_i^2 p(x_i) - (\mu)^2$$

Intervening algebra!

$$= E(x^2) - [E(x)]^2$$

For example, what are the mean and standard deviation of the roll of a die?

x	p(x)
1	p(x=1)=1/6
2	p(x=2)=1/6
3	p(x=3)=1/6
4	p(x=4)=1/6
5	p(x=5)=1/6
6	p(x=6)=1/6
	1.0



$$E(x) = \sum_{\text{all x}} x_i p(x_i) = (1)(\frac{1}{6}) + 2(\frac{1}{6}) + 3(\frac{1}{6}) + 4(\frac{1}{6}) + 5(\frac{1}{6}) + 6(\frac{1}{6}) = \frac{21}{6} = 3.5$$

$$E(x^2) = \sum_{\text{all x}} x_i^2 p(x_i) = (1)(\frac{1}{6}) + 4(\frac{1}{6}) + 9(\frac{1}{6}) + 16(\frac{1}{6}) + 25(\frac{1}{6}) + 36(\frac{1}{6}) = 15.17$$

$$\sigma_x^2 = Var(x) = E(x^2) - [E(x)]^2 = 15.17 - 3.5^2 = 2.92$$

$$\sigma_x = \sqrt{2.92} = 1.71$$

Practice Problem

Find the variance and standard deviation for Rohan's night wakings (recall that we already calculated the mean to be 3.2):

x	**1**	**2**	**3**	**4**	**5**
P(x)	**.1**	**.1**	**.4**	**.3**	**.1**

Answer:

x^2	1	4	9	16	25
P(x)	.1	.1	.4	.3	.1

$$E(x^2)=\sum_{i=1}^{5} x_i^{\,2}\,p(x_i)=(1)(.1)+(4)(.1)+9(.4)+16(.3)+25(.1)=11.4$$

$$Var(x)=E(x^2)-[E(x)]^2=11.4-3.2^2=1.16$$

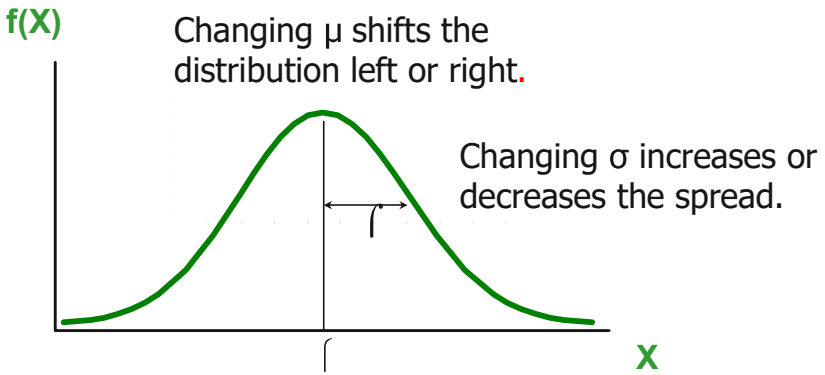
$$stddev(x)=\sqrt{1.16}=1.08$$

Interpretation: On an average night, we expect Rohan to awaken 3 times, plus or minus 1.08. This gives you a feel for what would be considered an unusual night!

continuous probability(Gaussian) distributions:

The normal and standard normal

The Normal Distribution



The Normal Distribution: as mathematical function (pdf)

$$f(x)=\frac{1}{\sigma\sqrt{2\pi}}\cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2}$$

Note constants:
π=3.14159
e=2.71828

This is a bell shaped curve with different centers and spreads depending on μ and σ

The Normal PDF

It's a probability function, so no matter what the values of μ and σ, must integrate to 1!

$$\int_{-\infty}^{+\infty} \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx = 1$$

Normal distribution is defined by its mean and standard dev.

$E(X)=\mu =$ $\int_{-\infty}^{+\infty} x \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx$

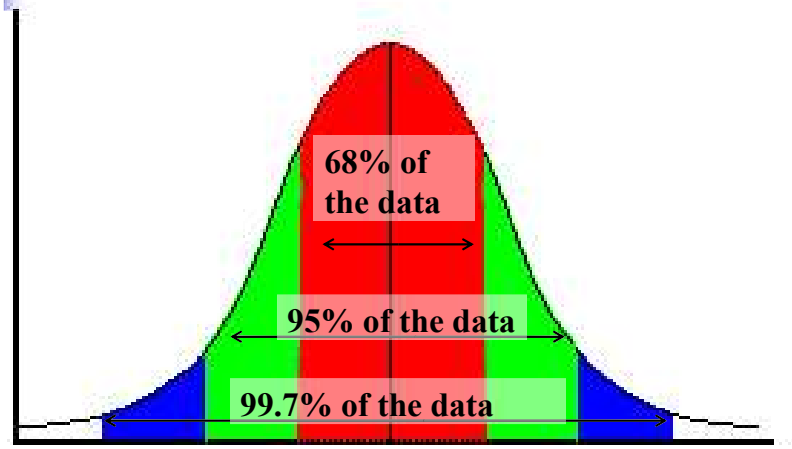
$Var(X)=\sigma^2 =$ $\int_{-\infty}^{+\infty} x^2 \frac{1}{\sigma\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} dx) - \mu^2$

Standard Deviation(X)=σ

**The beauty of the normal curve:

68-95-99.7 Rule

No matter what μ and σ are, the area between μ-σ and μ+σ is about 68%; the area between μ-2σ and μ+2σ is about 95%; and the area between μ-3σ and μ+3σ is about 99.7%. Almost all values fall within 3 standard deviations.



68-95-99.7 Rule in Math terms…

$$\int_{\mu-\sigma}^{\mu+\sigma} \frac{1}{\sigma\sqrt{2\pi}} \bullet e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = .68$$

$$\int_{\mu-2\sigma}^{\mu+2\sigma} \frac{1}{\sigma\sqrt{2\pi}} \bullet e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = .95$$

$$\int_{\mu-3\sigma}^{\mu+3\sigma} \frac{1}{\sigma\sqrt{2\pi}} \bullet e^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2} dx = .997$$

How good is rule for real data?

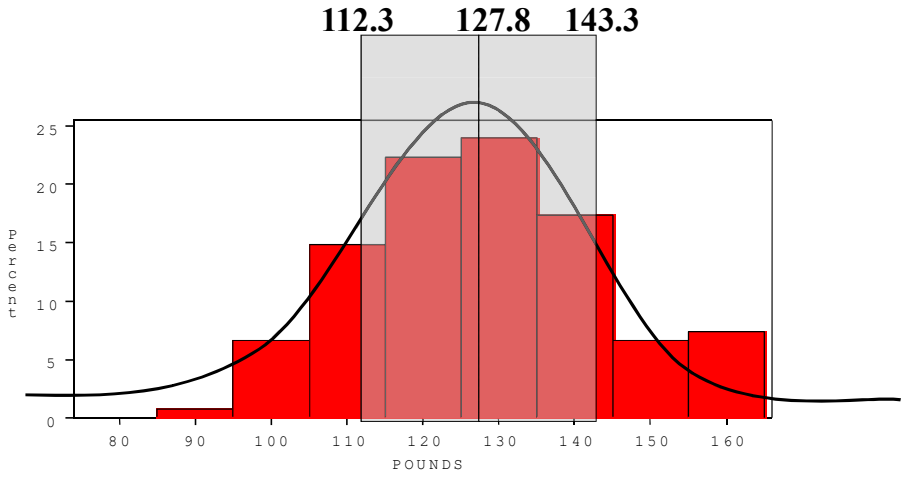
Check some example data:

The mean of the weight of the women = 127.8

The standard deviation (SD) = 15.5

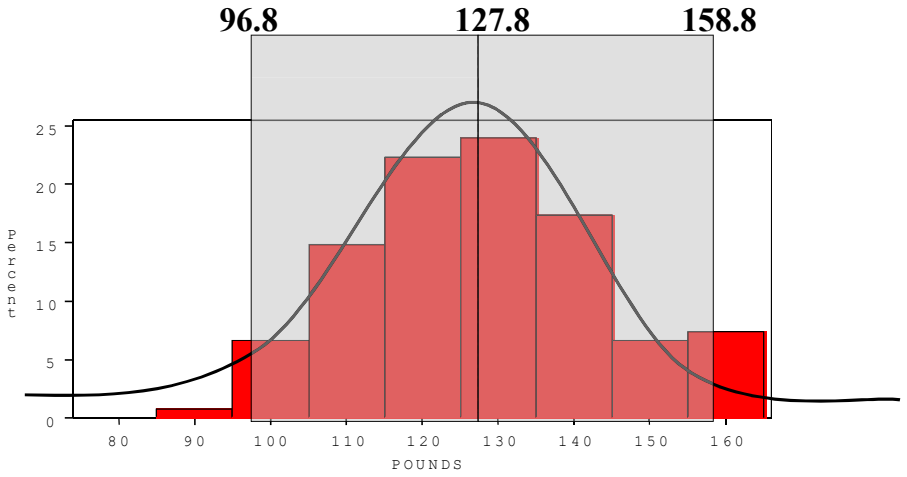
68% of 120 = .68x120 = ~ 82 runners

In fact, 79 runners fall within 1-SD (15.5 lbs) of the mean.



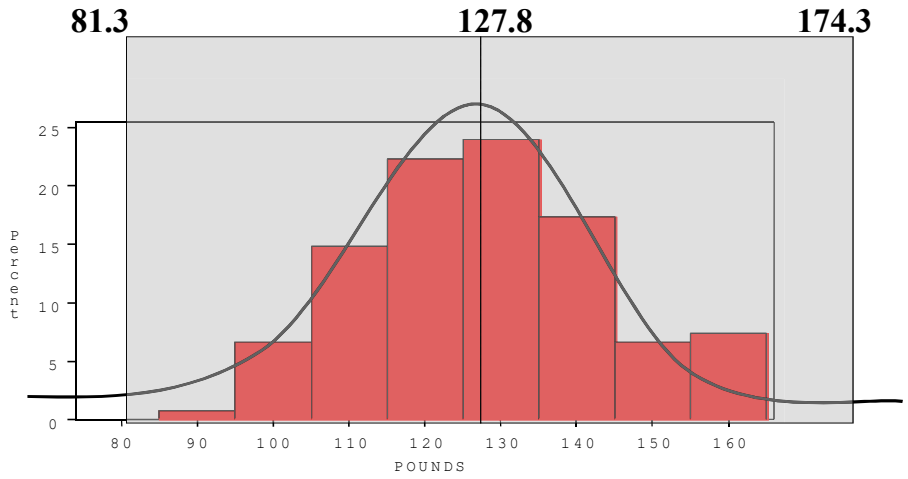
95% of 120 = .95 x 120 = ~ 114 runners

In fact, 115 runners fall within 2-SD's of the mean.



99.7% of 120 = .997 x 120 = 119.6 runners

In fact, all 120 runners fall within 3-SD's of the mean.



Example

- Suppose SAT scores roughly follows a normal distribution in the U.S. population of college-bound students (with range restricted to 200-800), and the average math SAT is 500 with a standard deviation of 50, then:
  - 68% of students will have scores between 450 and 550
  - 95% will be between 400 and 600
  - 99.7% will be between 350 and 650

Example

- BUT…
- What if you wanted to know the math SAT score corresponding to the 90th percentile (=90% of students are lower)?

$P(X \leq Q) = .90 \rightarrow$

$$\int_{200}^{Q} \frac{1}{(50)\sqrt{2\pi}} \bullet e^{-\frac{1}{2}\left(\frac{x-500}{50}\right)^2} dx = .90$$

The Standard Normal (Z): "Universal Currency"

The formula for the standardized normal probability density function is

$$p(Z) = \frac{1}{(1)\sqrt{2\pi}} \cdot e^{-\frac{1}{2}\left(\frac{Z-0}{1}\right)^2} = \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(Z)^2}$$

The Standard Normal Distribution (Z)

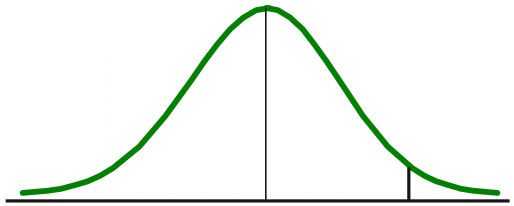
All normal distributions can be converted into the standard normal curve by subtracting the mean and dividing by the standard deviation:

$$Z = \frac{X - \mu}{\sigma}$$

Somebody calculated all the integrals for the standard normal and put them in a table! So we never have to integrate!

Even better, computers now do all the integration.

Comparing X and Z units



100 200 X

0 2.0 Z

($\mu = 100, \sigma = 50$)

($\mu = 0, \sigma = 1$)

Example

- For example: What's the probability of getting a math SAT score of 575 or less, μ=500 and σ=50?

$$Z = \frac{575 - 500}{50} = 1.5$$

- i.e., A score of 575 is 1.5 standard deviations above the mean

$$\therefore P(X \le 575) = \int_{200}^{575} \frac{1}{(50)\sqrt{2\pi}} \cdot e^{-\frac{1}{2}(\frac{x-500}{50})^2} dx \longrightarrow \int_{-\infty}^{1.5} \frac{1}{\sqrt{2\pi}} \cdot e^{-\frac{1}{2}Z^2} dz$$

But to look up Z= 1.5 in standard normal chart (or enter into SAS)→ no problem! = .9332

Answer

a. What is the chance of obtaining a birth weight of 141 oz *or heavier* when sampling birth records at random?

$$Z = \frac{141 - 109}{13} = 2.46$$

From the chart or SAS → Z of 2.46 corresponds to a right tail (greater than) area of: P(Z≥2.46) = 1-(.9931)= .0069 or .69 %

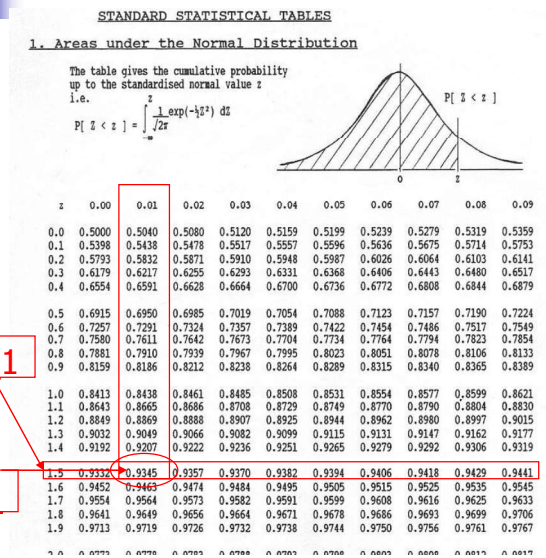
Answer

b. What is the chance of obtaining a birth weight of 120 *or lighter*?

$$Z = \frac{120 - 109}{13} = .85$$

From the chart or SAS → Z of .85 corresponds to a left tail area of:
P(Z≤.85) = .8023= 80.23%

Looking up probabilities in the standard normal table



STANDARD STATISTICAL TABLES

1. Areas under the Normal Distribution

The table gives the cumulative probability up to the standardised normal value z

i.e. $P[\,Z < z\,] = \int_{-\infty}^{z} \frac{1}{\sqrt{2\pi}} \exp(-\frac{1}{2}Z^2)\, dZ$

z	0.00	0.01	0.02	0.03	0.04	0.05	0.06	0.07	0.08	0.09
0.0	0.5000	0.5040	0.5080	0.5120	0.5159	0.5199	0.5239	0.5279	0.5319	0.5359
0.1	0.5398	0.5438	0.5478	0.5517	0.5557	0.5596	0.5636	0.5675	0.5714	0.5753
0.2	0.5793	0.5832	0.5871	0.5910	0.5948	0.5987	0.6026	0.6064	0.6103	0.6141
0.3	0.6179	0.6217	0.6255	0.6293	0.6331	0.6368	0.6406	0.6443	0.6480	0.6517
0.4	0.6554	0.6591	0.6628	0.6664	0.6700	0.6736	0.6772	0.6808	0.6844	0.6879
0.5	0.6915	0.6950	0.6985	0.7019	0.7054	0.7088	0.7123	0.7157	0.7190	0.7224
0.6	0.7257	0.7291	0.7324	0.7357	0.7389	0.7422	0.7454	0.7486	0.7517	0.7549
0.7	0.7580	0.7611	0.7642	0.7673	0.7704	0.7734	0.7764	0.7794	0.7823	0.7854
0.8	0.7881	0.7910	0.7939	0.7967	0.7995	0.8023	0.8051	0.8078	0.8106	0.8133
0.9	0.8159	0.8186	0.8212	0.8238	0.8264	0.8289	0.8315	0.8340	0.8365	0.8389
1.0	0.8413	0.8438	0.8461	0.8485	0.8508	0.8531	0.8554	0.8577	0.8599	0.8621
1.1	0.8643	0.8665	0.8686	0.8708	0.8729	0.8749	0.8770	0.8790	0.8804	0.8830
1.2	0.8849	0.8869	0.8888	0.8907	0.8925	0.8944	0.8962	0.8980	0.8997	0.9015
1.3	0.9032	0.9049	0.9066	0.9082	0.9099	0.9115	0.9131	0.9147	0.9162	0.9177
1.4	0.9192	0.9207	0.9222	0.9236	0.9251	0.9265	0.9279	0.9292	0.9306	0.9319
1.5	0.9332	0.9345	0.9357	0.9370	0.9382	0.9394	0.9406	0.9418	0.9429	0.9441
1.6	0.9452	0.9463	0.9474	0.9484	0.9495	0.9505	0.9515	0.9525	0.9535	0.9545
1.7	0.9554	0.9564	0.9573	0.9582	0.9591	0.9599	0.9608	0.9616	0.9625	0.9633
1.8	0.9641	0.9649	0.9656	0.9664	0.9671	0.9678	0.9686	0.9693	0.9699	0.9706
1.9	0.9713	0.9719	0.9726	0.9732	0.9738	0.9744	0.9750	0.9756	0.9761	0.9767

Z=1.51

Z=1.51

What is the area to the left of Z=1.51 in a standard normal curve?

Area is 93.45%

Bayesian Networks

Introduction

Suppose you are trying to determine if a patient has inhalational anthrax. You observe the following symptoms:

- The patient has a cough
- The patient has a fever
- The patient has difficulty breathing

Introduction

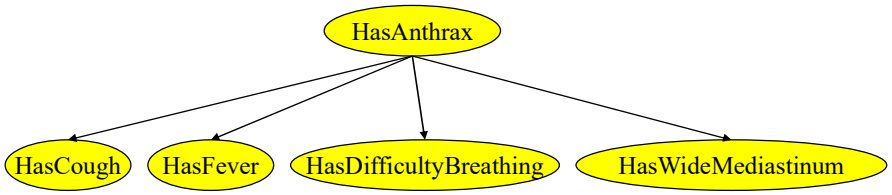
You would like to determine how likely the patient is infected with inhalational anthrax given that the patient has a cough, a fever, and difficulty breathing

We are not 100% certain that the patient has anthrax because of these symptoms. We are dealing with uncertainty!

Introduction

- In the previous slides, what you observed affected your belief that the patient is infected with anthrax
- This is called reasoning with uncertainty
- Wouldn't it be nice if we had some methodology for reasoning with uncertainty? Well in fact, we do…

Bayesian Networks



- In the opinion of many AI researchers, Bayesian networks are the most significant contribution in AI in the last 10 years
- They are used in many applications eg. spam filtering, speech recognition, robotics, diagnostic systems and even syndromic surveillance

Probability Primer: Random Variables

- A random variable is the basic element of probability
- Refers to an event and there is some degree of uncertainty as to the outcome of the event
- For example, the random variable *A* could be the event of getting a head on a coin flip

Boolean Random Variables

- We will start with the simplest type of random variables – Boolean ones
- Take the values *true* or *false*
- Think of the event as occurring or not occurring
- Examples (Let *A* be a Boolean random variable):

  A = Getting a head on a coin flip

  A = It will rain today

The Joint Probability Distribution

- Joint probabilities can be between any number of variables

  eg. *P(A = true, B = true, C = true)*
- For each combination of variables, we need to say how probable that combination is
- The probabilities of these combinations need to sum to 1

A	B	C	P(A,B,C)
false	false	false	0.1
false	false	true	0.2
false	true	false	0.05
false	true	true	0.05
true	false	false	0.3
true	false	true	0.1
true	true	false	0.05
true	true	true	0.15

Sums to 1

The Joint Probability Distribution

- Once you have the joint probability distribution, you can calculate any probability involving *A*, *B*, and *C*
- Note: May need to use marginalization and Bayes rule, (both of which are not discussed in these slides)

A	B	C	P(A,B,C)
false	false	false	0.1
false	false	true	0.2
false	true	false	0.05
false	true	true	0.05
true	false	false	0.3
true	false	true	0.1
true	true	false	0.05
true	true	true	0.15

Examples of things you can compute:

- *P(A=true)* = sum of *P(A,B,C)* in rows with *A=true*
- *P(A=true, B = true | C=true)* =

  P(A = true, B = true, C = true) / P(C = true)

The Problem with the Joint Distribution

- Lots of entries in the table to fill up!
- For *k* Boolean random variables, you need a table of size 2^k
- How do we use fewer numbers? Need the concept of independence

A	B	C	P(A,B,C)
false	false	false	0.1
false	false	true	0.2
false	true	false	0.05
false	true	true	0.05
true	false	false	0.3
true	false	true	0.1
true	true	false	0.05
true	true	true	0.15

Independence

Variables *A* and *B* are independent if any of the following hold:

- *P(A,B) = P(A) P(B)*
- *P(A | B) = P(A)*
- *P(B | A) = P(B)*

Independence

How is independence useful?

- Suppose you have n coin flips and you want to calculate the joint distribution $\boldsymbol{P}(C_1, \ldots, C_n)$
- If the coin flips are not independent, you need 2^n values in the table
- If the coin flips are independent, then

$$P(C_1,\ldots,C_n) = \prod_{i=1}^{n} P(C_i)$$

Each $P(C_i)$ table has 2 entries and there are *n* of them for a total of $2n$ values

Conditional Independence

Variables A and B are conditionally independent given C if any of the following hold:

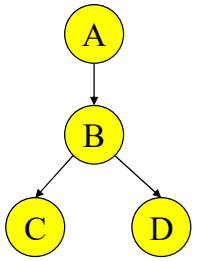
- $P(A, B \mid C) = P(A \mid C)\, P(B \mid C)$
- $P(A \mid B, C) = P(A \mid C)$
- $P(B \mid A, C) = P(B \mid C)$

Knowing C tells me everything about B. I don't gain anything by knowing A (either because A doesn't influence B or because knowing C provides all the information knowing A would give)

A Bayesian Network

A Bayesian network is made up of:

1. A Directed Acyclic Graph



2. A set of tables for each node in the graph

A	P(A)
false	0.6
true	0.4

A	B	P(B|A)
false	false	0.01
false	true	0.99
true	false	0.7
true	true	0.3

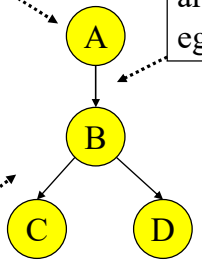
B	D	P(D|B)
false	false	0.02
false	true	0.98
true	false	0.05
true	true	0.95

B	C	P(C|B)
false	false	0.4
false	true	0.6
true	false	0.9
true	true	0.1

A Directed Acyclic Graph

Each node in the graph is a random variable

A node X is a parent of another node Y if there is an arrow from node X to node Y eg. A is a parent of B



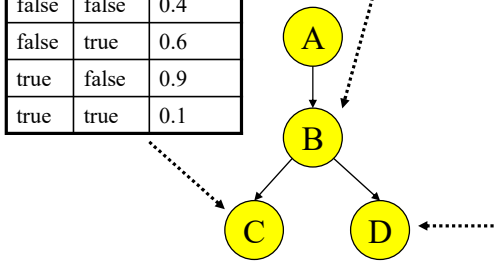
Informally, an arrow from node X to node Y means X has a direct influence on Y

A Set of Tables for Each Node

A	P(A)
false	0.6
true	0.4

A	B	P(B|A)
false	false	0.01
false	true	0.99
true	false	0.7
true	true	0.3

B	C	P(C|B)
false	false	0.4
false	true	0.6
true	false	0.9
true	true	0.1



B	D	P(D|B)
false	false	0.02
false	true	0.98
true	false	0.05
true	true	0.95

Each node X_i has a conditional probability distribution $P(X_i \mid \text{Parents}(X_i))$ that quantifies the effect of the parents on the node

The parameters are the probabilities in these conditional probability tables (CPTs)

A Set of Tables for Each Node

Conditional Probability
Distribution for C given B

B	C	P(C|B)
false	false	0.4
false	true	0.6
true	false	0.9
true	true	0.1

For a given combination of values of the parents (B in this example), the entries for P(C=true | B) and P(C=false | B) must add up to 1
eg. P(C=true | B=false) + P(C=false |B=false)=1

If you have a Boolean variable with k Boolean parents, this table has 2^{k+1} probabilities (but only 2^k need to be stored)

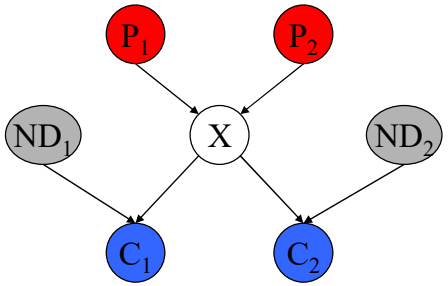
Bayesian Networks

Two important properties:

1. Encodes the conditional independence relationships between the variables in the graph structure
2. Is a compact representation of the joint probability distribution over the variables

Conditional Independence

The Markov condition: given its parents (P_1, P_2), a node (X) is conditionally independent of its non-descendants (ND_1, ND_2)



The Joint Probability Distribution

Due to the Markov condition, we can compute the joint probability distribution over all the variables X_1, …, X_n in the Bayesian net using the formula:

$$P(X_1 = x_1,..., X_n = x_n) = \prod_{i=1}^{n} P(X_i = x_i \mid Parents(X_i))$$

Where Parents(X_i) means the values of the Parents of the node X_i with respect to the graph

Using a Bayesian Network Example

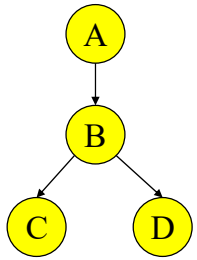
Using the network in the example, suppose you want to calculate:

P(A = true, B = true, C = true, D = true)

= P(A = true) * P(B = true | A = true) *

P(C = true | B = true)* P(D = true | B = true)

= (0.4)*(0.3)*(0.1)*(0.95)



Using a Bayesian Network Example

Using the network in the example, suppose you want to calculate:

P(A = true, B = true, C = true, D = true)

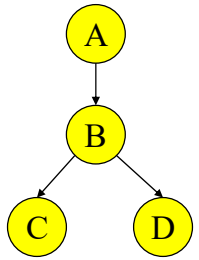
= P(A = true) * P(B = true | A = true) *

P(C = true | B = true) P(D = true | B = true)

= (0.4)*(0.3)*(0.1)*(0.95)

This is from the graph structure

These numbers are from the conditional probability tables



Joint Probability Factorization

For any joint distribution of random variables the following factorization is always true:

$$P(A,B,C,D) = P(A)P(B \mid A)P(C \mid A,B)P(D \mid A,B,C)$$

We derive it by repeatedly applying the Bayes' Rule P(X,Y)=P(X|Y)P(Y):

$$P(A,B,C,D) = P(B,C,D \mid A)P(A)$$
$$= P(C,D \mid B,A)P(B \mid A)P(A)$$
$$= P(D \mid C,B,A)P(C \mid B,A)P(B \mid A)P(A)$$
$$P(A)P(B \mid A)P(C \mid A,B)P(D \mid A,B,C)$$

Joint Probability Factorization

Our example graph carries additional independence information, which simplifies the joint distribution:

$$P(A,B,C,D) = P(A)P(B \mid A)P(C \mid A,B)P(D \mid A,B,C)$$
$$= P(A)P(B \mid A)P(C \mid B)P(D \mid B)$$

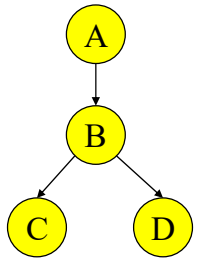
This is why, we only need the tables for P(A), P(B|A), P(C|B), and P(D|B) and why we computed

P(A = true, B = true, C = true, D = true)

= P(A = true) * P(B = true | A = true) *

P(C = true | B = true) P(D = true | B = true)

= (0.4)*(0.3)*(0.1)*(0.95)



Inference

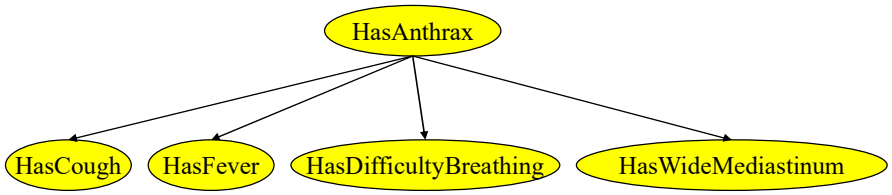
- Using a Bayesian network to compute probabilities is called inference
- In general, inference involves queries of the form:

  P(X | E)

  E = The evidence variable(s)

  X = The query variable(s)

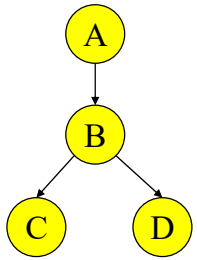
Inference



- An example of a query would be:

  P(*HasAnthrax* = *true* | *HasFever* = *true*, *HasCough* = *true*)
- Note: Even though *HasDifficultyBreathing* and *HasWideMediastinum* are in the Bayesian network, they are not given values in the query (ie. they do not appear either as query variables or evidence variables)
- They are treated as unobserved variables and summed out.

Inference Example



Supposed we know that A=true.
What is more probable C=true or D=true?
For this we need to compute
P(C=t | A =t) and P(D=t | A =t).
Let us compute the first one.

$$P(C=t \mid A=t) = \frac{P(A=t, C=t)}{P(A=t)} = \frac{\sum_{b,d} P(A=t, B=b, C=t, D=d)}{P(A=t)}$$

A	P(A)
false	0.6
true	0.4

A	B	P(B|A)
false	false	0.01
false	true	0.99
true	false	0.7
true	true	0.3

B	D	P(D|B)
false	false	0.02
false	true	0.98
true	false	0.05
true	true	0.95

B	C	P(C|B)
false	false	0.4
false	true	0.6
true	false	0.9
true	true	0.1

Introduction to Machine Learning

Why "Learn"?

- Machine learning is programming computers to optimize a performance criterion using example data or past experience.
- There is no need to "learn" to calculate payroll
- Learning is used when:
  - Human expertise does not exist (navigating on Mars),
  - Humans are unable to explain their expertise (speech recognition)
  - Solution changes in time (routing on a computer network)
  - Solution needs to be adapted to particular cases (user biometrics)

What We Talk About When We Talk About"Learning"

- Learning general models from a data of particular examples
- Data is cheap and abundant (data warehouses, data marts); knowledge is expensive and scarce.
- Example in retail: Customer transactions to consumer behavior:

  People who bought "Da Vinci Code" also bought "The Five People You Meet in Heaven" (www.amazon.com)

- Build a model that is *a good and useful approximation* to the data.

What is Machine Learning?

- Machine Learning
  - Study of algorithms that
  - improve their performance
  - at some task
  - with experience
- Optimize a performance criterion using example data or past experience.
- Role of Statistics: Inference from a sample
- Role of Computer science: Efficient algorithms to
  - Solve the optimization problem
  - Representing and evaluating the model for inference

Growth of Machine Learning

- Machine learning is preferred approach to
  - Speech recognition, Natural language processing
  - Computer vision
  - Medical outcomes analysis
  - Robot control
  - Computational biology
- This trend is accelerating
  - Improved machine learning algorithms
  - Improved data capture, networking, faster computers
  - Software too complex to write by hand
  - New sensors / IO devices
  - Demand for self-customization to user, environment
  - It turns out to be difficult to extract knowledge from human experts→*failure of expert systems in the 1980's.*

Applications

- Association Analysis
- Supervised Learning
  - Classification
  - Regression/Prediction
- Unsupervised Learning
- Reinforcement Learning

Learning Associations

- Basket analysis:

  $P(Y \mid X)$ probability that somebody who buys X also buys Y where X and Y are products/services.

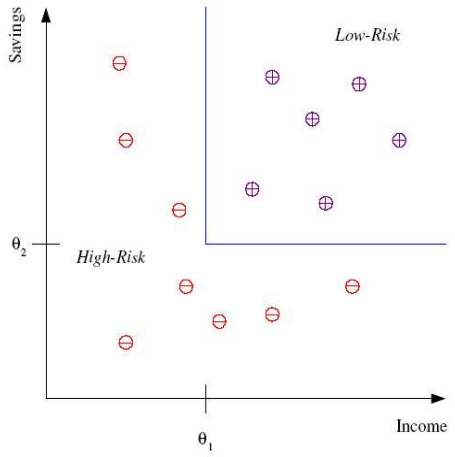
  Example: P (chips | beer) = 0.7

Market-Basket transactions

TID	Items
1	Bread, Milk
2	Bread, Diaper, Beer, Eggs
3	Milk, Diaper, Beer, Coke
4	Bread, Milk, Diaper, Beer
5	Bread, Milk, Diaper, Coke

Classification

- Example: Credit scoring
- Differentiating between low-risk and high-risk customers from their *income* and *savings*



Discriminant: IF *income* > θ_1 AND *savings* > θ_2 THEN low-risk ELSE high-risk

Model

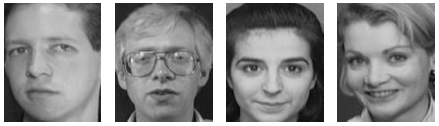
Classification: Applications

- Aka Pattern recognition
- Face recognition: Pose, lighting, occlusion (glasses, beard), make-up, hair style
- Character recognition: Different handwriting styles.
- Speech recognition: Temporal dependency.
  - Use of a dictionary or the syntax of the language.
  - Sensor fusion: Combine multiple modalities; eg, visual (lip image) and acoustic for speech
- Medical diagnosis: From symptoms to illnesses
- Web Advertizing: Predict if a user clicks on an ad on the Internet.

Face Recognition

Training examples of a person

Test images



AT&T Laboratories, Cambridge UK
http://www.uk.research.att.com/facedatabase.html

Prediction: Regression

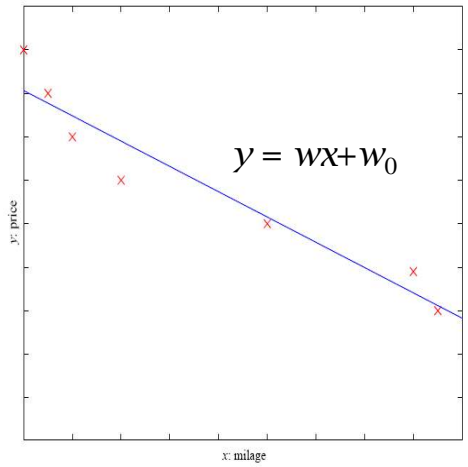
- Example: Price of a used car
- x : car attributes

  y : price

  $y = g(x \mid \theta)$

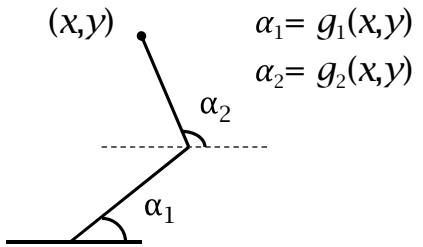
  $g(\ )$ model,

  θ parameters



Regression Applications

- Navigating a car: Angle of the steering wheel (CMU NavLab)
- Kinematics of a robot arm



Supervised Learning: Uses

Example: decision trees tools that create rules

- Prediction of future cases: Use the rule to predict the output for future inputs
- Knowledge extraction: The rule is easy to understand
- Compression: The rule is simpler than the data it explains
- Outlier detection: Exceptions that are not covered by the rule, e.g., fraud

Unsupervised Learning

- Learning "what normally happens"
- No output
- Clustering: Grouping similar instances
- Other applications: Summarization, Association Analysis
- Example applications
  - Customer segmentation in CRM
  - Image compression: Color quantization
  - Bioinformatics: Learning motifs

Reinforcement Learning

- Topics:
  - Policies: what actions should an agent take in a particular situation
  - Utility estimation: how good is a state (→used by policy)
- No supervised output but delayed reward
- Credit assignment problem (what was responsible for the outcome)
- Applications:
  - Game playing
  - Robot in a maze
  - Multiple agents, partial observability, ...

Resources: Datasets

- UCI Repository: http://www.ics.uci.edu/~mlearn/MLRepository.html
- UCI KDD Archive: http://kdd.ics.uci.edu/summary.data.application.html
- Statlib: http://lib.stat.cmu.edu/
- Delve: http://www.cs.utoronto.ca/~delve/

Resources: Journals

- Journal of Machine Learning Research www.jmlr.org
- Machine Learning
- IEEE Transactions on Neural Networks
- IEEE Transactions on Pattern Analysis and Machine Intelligence
- Annals of Statistics
- Journal of the American Statistical Association
- ...

Resources: Conferences

- International Conference on Machine Learning (ICML)
- European Conference on Machine Learning (ECML)
- Neural Information Processing Systems (NIPS)
- Computational Learning
- International Joint Conference on Artificial Intelligence (IJCAI)
- ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)
- IEEE Int. Conf. on Data Mining (ICDM)

Naive Bayesian classifier

Bayesian Classification: Why?

- A statistical classifier: performs *probabilistic prediction, i.e.,* predicts class membership probabilities
- Foundation: Based on Bayes' Theorem.
- Performance: A simple Bayesian classifier, *naïve Bayesian classifier*, has comparable performance with decision tree and selected neural network classifiers
- Incremental: Each training example can incrementally increase/decrease the probability that a hypothesis is correct — prior knowledge can be combined with observed data

Bayes' Rule

$$p(h \mid d) = \frac{P(d \mid h)P(h)}{P(d)}$$

Understanding Bayes' rule

d = data

h = hypothesis (model)

- rearranging

$p(h \mid d)P(d) = P(d \mid h)P(h)$

$P(d,h) = P(d,h)$

the same joint probability on both sides

Who is who in Bayes' rule

$P(h)$: prior belief (probability of hypothesis *h* before seeing any data)

$P(d \mid h)$: likelihood (probability of the data if the hypothesis *h* is true)

$P(d) = \sum_h P(d \mid h)P(h)$: data evidence (marginal probability of the data)

$P(h \mid d)$: posterior (probability of hypothesis *h* after having seen the data *d*)

Example of Bayes Theorem

- Given:
  - A doctor knows that meningitis causes stiff neck 50% of the time
  - Prior probability of any patient having meningitis is 1/50,000
  - Prior probability of any patient having stiff neck is 1/20
- If a patient has stiff neck, what's the probability he/she has meningitis?

$$P(M \mid S) = \frac{P(S \mid M)P(M)}{P(S)} = \frac{0.5 \times 1/50000}{1/20} = 0.0002$$

Choosing Hypotheses

- *Maximum Likelihood* hypothesis:

$$h_{ML} = \arg\max_{h \in H} P(d \mid h)$$

$$h_{MAP} = \arg\max_{h \in H} P(h \mid d)$$

- Generally we want the most probable hypothesis given training data.This is the *maximum a posteriori* hypothesis:
  - Useful observation: it does not depend on the denominator P(d)

Bayesian Classifiers

- Consider each attribute and class label as random variables
- Given a record with attributes (A_1, A_2,…,A_n)
  - Goal is to predict class C
  - Specifically, we want to find the value of C that maximizes P(C| A_1, A_2,…,A_n)
- Can we estimate P(C| A_1, A_2,…,A_n) directly from data?

Bayesian Classifiers

- Approach:
  - compute the posterior probability P(C | A_1, A_2, …, A_n) for all values of C using the Bayes theorem

$$P(C \mid A_1 A_2 \ldots A_n) = \frac{P(A_1 A_2 \ldots A_n \mid C) P(C)}{P(A_1 A_2 \ldots A_n)}$$

  - Choose value of C that maximizes
    P(C | A_1, A_2, …, A_n)
  - Equivalent to choosing value of C that maximizes
    P(A_1, A_2, …, A_n|C) P(C)
- How to estimate P(A_1, A_2, …, A_n | C)?

Naïve Bayes Classifier

- Assume independence among attributes A_i when class is given:
  - P(A_1, A_2, …, A_n |C) = P(A_1| C_j) P(A_2| C_j)… P(A_n| C_j)
  - Can estimate P(A_i| C_j) for all A_i and C_j.
  - This is a simplifying assumption which may be violated in reality
- The Bayesian classifier that uses the Naïve Bayes assumption and computes the MAP hypothesis is called Naïve Bayes classifier

$$c_{Naive\ Bayes} = \arg\max_{c} P(c) P(\mathbf{x} \mid c) = \arg\max_{c} P(c) \prod_{i} P(a_i \mid c)$$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Class: $P(C) = N_c/N$
  - e.g., P(No) = 7/10, P(Yes) = 3/10
- For discrete attributes:

$$P(A_i \mid C_k) = |A_{ik}| / {}_k N_c$$

  - where $|A_{ik}|$ is number of instances having attribute A_i and belongs to class C_k
  - Examples:

    P(Status=Married|No) = 4/7
    P(Refund=Yes|Yes)=0

How to Estimate Probabilities from Data?

- For continuous attributes:
  - Discretize the range into bins
    - one ordinal attribute per bin
    - violates independence assumption
  - Two-way split: (A < v) or (A > v)
    - choose only one of the two splits as new attribute
  - Probability density estimation:
    - Assume attribute follows a normal distribution
    - Use data to estimate parameters of distribution (e.g., mean and standard deviation)
    - Once probability distribution is known, can use it to estimate the conditional probability $P(A_i|c)$

How to Estimate Probabilities from Data?

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

- Normal distribution:

$$P(A_i \mid c_j) = \frac{1}{\sqrt{2\pi\sigma_{ij}^2}} e^{\frac{-(A_i-\mu_{ij})^2}{2\sigma_{ij}^2}}$$

  - One for each (A_i, c_i) pair
- For (Income, Class=No):
  - If Class=No
    - sample mean = 110
    - sample variance = 2975

$$P(Income = 120 \mid No) = \frac{1}{\sqrt{2\pi}(54.54)} e^{\frac{-(120-110)^2}{2(2975)}} = 0.0072$$

Naïve Bayesian Classifier: Training Dataset

Class:
C1:buys_computer = 'yes'
C2:buys_computer = 'no'

New Data:
X = (age <=30,
Income = medium,
Student = yes
Credit_rating = Fair)

age	income	student	credit_rating	comp
<=30	high	no	fair	no
<=30	high	no	excellent	no
31…40	high	no	fair	yes
>40	medium	no	fair	yes
>40	low	yes	fair	yes
>40	low	yes	excellent	no
31…40	low	yes	excellent	yes
<=30	medium	no	fair	no
<=30	low	yes	fair	yes
>40	medium	yes	fair	yes
<=30	medium	yes	excellent	yes
31…40	medium	no	excellent	yes
31…40	high	yes	fair	yes
>40	medium	no	excellent	no

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)
Maximize P(X|Ci)P(Ci), for i=1,2

First step: Compute P(C) The prior probability of each class can be computed based on the training tuples:

P(buys_computer=yes)=9/14=0.643
P(buys_computer=no)=5/14=0.357

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)
Maximize P(X|Ci)P(Ci), for i=1,2

Second step: compute P(X|Ci)
P(**X|buys_computer=yes**)= P(age=youth|buys_computer=yes)x
P(income=medium|buys_computer=yes) x
P(student=yes|buys_computer=yes)x
P(credit_rating=fair|buys_computer=yes)
= 0.044
P(age=youth|buys_computer=yes)=0.222
P(income=medium|buys_computer=yes)=0.444
P(student=yes|buys_computer=yes)=6/9=0.667
P(credit_rating=fair|buys_computer=yes)=6/9=0.667

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)
Maximize P(X|Ci)P(Ci), for i=1,2

Second step: compute P(X|Ci)
P(**X|buys_computer=no**)= P(age=youth|buys_computer=no)x
P(income=medium|buys_computer=no) x
P(student=yes|buys_computer=no) x
P(credit_rating=fair|buys_computer=no)
= 0.019
P(age=youth|buys_computer=no)=3/5=0.666
P(income=medium|buys_computer=no)=2/5=0.400
P(student=yes|buys_computer=no)=1/5=0.200
P(credit_rating=fair|buys_computer=no)=2/5=0.400

Naïve Bayesian Classifier: An Example

Given X (age=youth, income=medium, student=yes, credit=fair)
Maximize P(X|Ci)P(Ci), for i=1,2

We have computed in the first and second steps:
P(buys_computer=yes)=9/14=0.643
P(buys_computer=no)=5/14=0.357
P(X|buys_computer=yes)= 0.044
P(X|buys_computer=no)= 0.019

Third step: compute **P(X|Ci)P(Ci)** for each class
P(X|buys_computer=yes)P(buys_computer=yes)=0.044 x 0.643=0.028
P(X|buys_computer=no)P(buys_computer=no)=0.019 x 0.357=0.007
The naïve Bayesian Classifier predicts **X belongs to class ("buys_computer = yes")**

Example

Training set :
(Öğrenme Kümesi)

Tid	Refund	Marital Status	Taxable Income	Evade
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

k

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

Example of Naïve Bayes Classifier

Given a Test Record:

$$X = (\text{Refund} = \text{No}, \text{Married}, \text{Income} = 120\text{K})$$

naive Bayes Classifier:

P(Refund=Yes|No) = 3/7
P(Refund=No|No) = 4/7
P(Refund=Yes|Yes) = 0
P(Refund=No|Yes) = 1
P(Marital Status=Single|No) = 2/7
P(Marital Status=Divorced|No)=1/7
P(Marital Status=Married|No) = 4/7
P(Marital Status=Single|Yes) = 2/7
P(Marital Status=Divorced|Yes)=1/7
P(Marital Status=Married|Yes) = 0

For taxable income:
If class=No: sample mean=110
sample variance=2975
If class=Yes: sample mean=90
sample variance=25

- P(X|Class=No) = P(Refund=No|Class=No)
  × P(Married| Class=No)
  × P(Income=120K| Class=No)
  = 4/7 × 4/7 × 0.0072 = 0.0024

- P(X|Class=Yes) = P(Refund=No| Class=Yes)
  × P(Married| Class=Yes)
  × P(Income=120K| Class=Yes)
  = 1 × 0 × 1.2 × 10^{-9} = 0

Since P(X|No)P(No) > P(X|Yes)P(Yes)
Therefore P(No|X) > P(Yes|X)
=> Class = No

Naïve Bayes (Summary)

- *Advantage*
  - Robust to isolated noise points
  - Handle missing values by ignoring the instance during probability estimate calculations
  - Robust to irrelevant attributes
- *Disadvantage*
  - Assumption: class conditional independence, which may cause loss of accuracy
  - Independence assumption may not hold for some attribute. Practically, dependencies exist among variables
    - Use other techniques such as Bayesian Belief Networks (BBN)

Remember

- Bayes' rule can be turned into a classifier
- Maximum A Posteriori (MAP) hypothesis estimation incorporates prior knowledge; Max Likelihood (ML) doesn't
- Naive Bayes Classifier is a simple but effective Bayesian classifier for vector data (i.e. data with several attributes) that assumes that attributes are independent given the class.
- Bayesian classification is a generative approach to classification

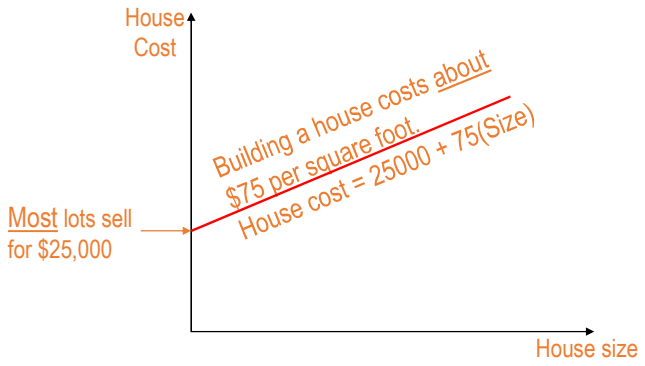
Linear Regression

Introduction

- The motivation for using the technique:
  - Forecast the value of a dependent variable (Y) from the value of independent variables (X_1, X_2,…X_k.).
  - Analyze the specific relationships between the independent variables and the dependent variable.

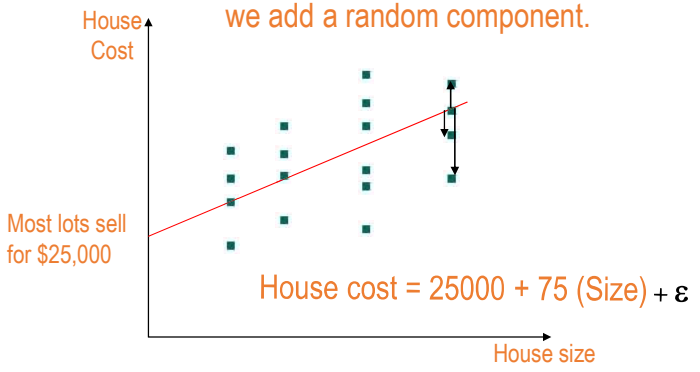
The Model

The model has a deterministic and a probabilistic components



However, house cost vary even among same size houses!

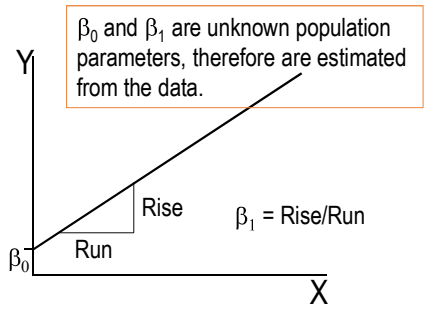
Since cost behave unpredictably, we add a random component.



- The first order linear model

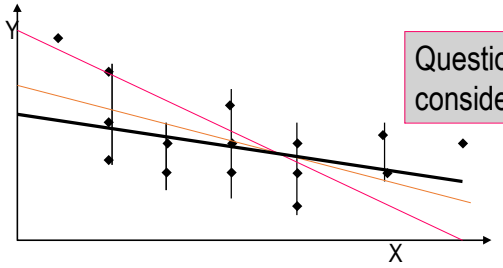
$$Y = \beta_0 + \beta_1 X + \varepsilon$$

Y = dependent variable
X = independent variable
β_0 = Y-intercept
β_1 = slope of the line
ε = error variable



Estimating the Coefficients

- The estimates are determined by
  - drawing a sample from the population of interest,
  - calculating sample statistics.
  - producing a straight line that cuts into the data.



Question: What should be considered a good line?

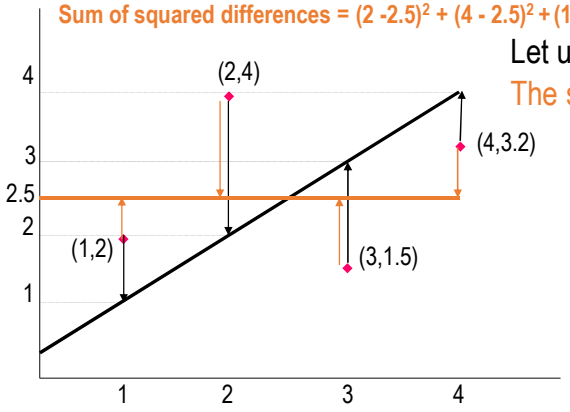
The Least Squares (Regression) Line

A good line is one that minimizes the sum of squared differences between the points and the line.

Sum of squared differences = $(2 - 1)^2 + (4 - 2)^2 + (1.5 - 3)^2 + (3.2 - 4)^2 = 6.89$

Sum of squared differences = $(2 - 2.5)^2 + (4 - 2.5)^2 + (1.5 - 2.5)^2 + (3.2 - 2.5)^2 = 3.99$

Let us compare two lines
The second line is horizontal



The smaller the sum of squared differences the better the fit of the line to the data.

The Estimated Coefficients

To calculate the estimates of the line coefficients, that minimize the differences between the data points and the line, use the formulas:

$$b_1 = \frac{\text{cov}(X,Y)}{s_X^2}\left(= \frac{s_{XY}}{s_X^2}\right)$$

$$b_0 = \overline{Y} - b_1\overline{X}$$

The regression equation that estimates the equation of the first order linear model is:

$$\hat{Y} = b_0 + b_1 X$$

The Simple Linear Regression Line

- Example 1
  - A car dealer wants to find the relationship between the odometer reading and the selling price of used cars.
  - A random sample of 100 cars is selected, and the data recorded.
  - Find the regression line.

Car	Odometer	Price
1	37388	14636
2	44758	14122
3	45833	14016
4	30862	15590
5	31705	15568
6	34010	14718
.	Independent variable X	Dependent variable Y
.	.	.

- Solution
  - Solving by hand: Calculate a number of statistics

$$\overline{X} = 36{,}009.45; \quad s_X^2 = \frac{\sum (X_i - \overline{X})^2}{n-1} = 43{,}528{,}690$$

$$\overline{Y} = 14{,}822.823; \quad \text{cov}(X,Y) = \frac{\sum (X_i - \overline{X})(Y_i - \overline{Y})}{n-1} = -2{,}712{,}511$$

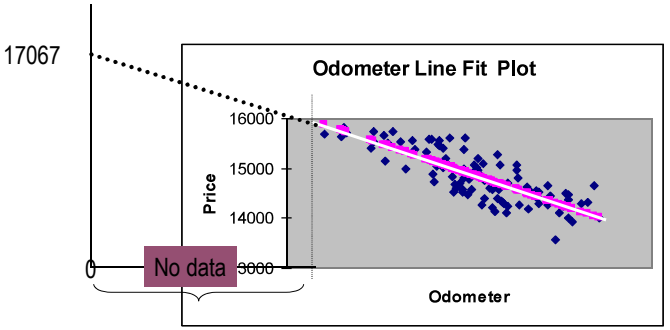
where n = 100.

$$b_1 = \frac{\text{cov}(X,Y)}{s_X^2} = \frac{-1{,}712{,}511}{43{,}528{,}690} = -.06232$$

$$b_0 = \overline{Y} - b_1\overline{X} = 14{,}822.82 - (-.06232)(36{,}009.45) = 17{,}067$$

$$\hat{Y} = b_0 + b_1 X = 17{,}067 - .0623X$$

Interpreting the Linear Regression -Equation



$$\hat{Y} = 17{,}067 - .0623X$$

The intercept is b_0 = $17067.

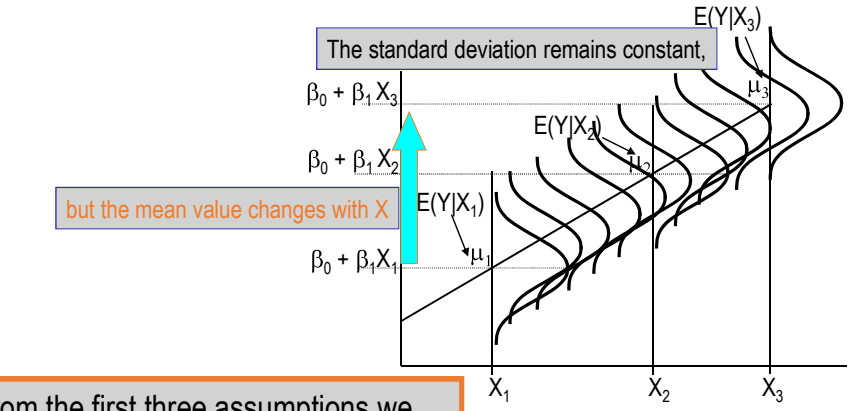
Do not interpret the intercept as the "Price of cars that have not been driven"

This is the slope of the line.
For each additional mile on the odometer, the price decreases by an average of $0.0623

Error Variable: Required Conditions

- The error ε is a critical part of the regression model.
- Four requirements involving the distribution of ε must be satisfied.
  - The probability distribution of ε is normal.
  - The mean of ε is zero: E(ε) = 0.
  - The standard deviation of ε is σ_ε for all values of X.
  - The set of errors associated with different values of Y are all independent.

The Normality of ε



From the first three assumptions we have: Y is normally distributed with mean $E(Y) = \beta_0 + \beta_1 X$, and a constant standard deviation σ_ε

Assessing the Model

- The least squares method will produces a regression line whether or not there are linear relationship between X and Y.
- Consequently, it is important to assess how well the linear model fits the data.
- Several methods are used to assess the model. All are based on the sum of squares for errors, SSE.

Sum of Squares for Errors

- This is the sum of differences between the points and the regression line.
- It can serve as a measure of how well the line fits the data. SSE is defined by

$$\text{SSE} = \sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2.$$

– A shortcut formula

$$\text{SSE} = (n-1)s_Y^2 - \frac{[\text{cov}(X,Y)]^2}{s_X^2}$$

Standard Error of Estimate

The mean error is equal to zero.

If se is small the errors tend to be close to zero (close to the mean error). Then, the model fits the data well.

Therefore, we can, use se as a measure of the suitability of using a linear model.

An estimator of se is given by se

Standard Error of Estimate

$$s_\varepsilon = \sqrt{\frac{SSE}{n-2}}$$

- Example 2
  - Calculate the standard error of estimate for Example 1, and describe what does it tell you about the model fit?
- Solution

$$s_Y^2 = \frac{\sum (Y_i - \hat{Y}_i)^2}{n-1} = 259{,}996$$

$$SSE = (n-1)s_Y^2 - \frac{[\text{cov}(X,Y)]^2}{s_X^2} = 99(259{,}996) - \frac{(-2{,}712{,}511)^2}{43{,}528{,}690} = 9{,}005{,}450$$

Calculated before

$$s_\varepsilon = \sqrt{\frac{SSE}{n-2}} = \sqrt{\frac{9{,}005{,}450}{98}} = 303.13$$

It is hard to assess the model based on s_ε even when compared with the mean value of Y.

$s_\varepsilon = 303.1$ $\bar{y} = 14{,}823$

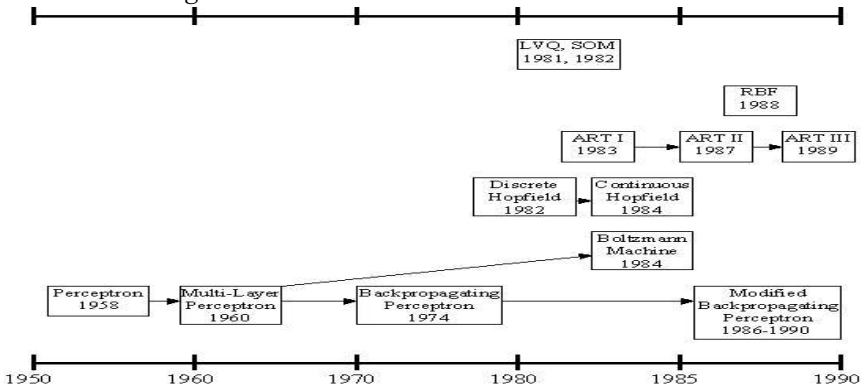
Neural Networks

Part 1

5/12/2024

History of the Artificial Neural Networks

- history of the ANNs stems from the 1940s, the decade of the first electronic computer.
- However, the first important step took place in 1957 when Rosenblatt introduced the first concrete neural model, the perceptron. Rosenblatt also took part in constructing the first successful neurocomputer, the Mark I Perceptron. After this, the development of ANNs has proceeded as described in *Figure*.



Artificial Neural Network

- An artificial neural network consists of a pool of simple processing units which communicate by sending signals to each other over a large number of weighted connections.

Artificial Neural Network

- A set of major aspects of a parallel distributed model include:
  - a set of processing units (cells).
  - a state of activation for every unit, which equivalent to the output of the unit.
  - connections between the units. Generally each connection is defined by a weight.
  - a propagation rule, which determines the effective input of a unit from its external inputs.
  - an activation function, which determines the new level of activation based on the effective input and the current activation.
  - an external input for each unit.
  - a method for information gathering (the learning rule).
  - an environment within which the system must operate, providing input signals and _ if necessary _ error signals.

Computers vs. Neural Networks

"Standard" Computers	Neural Networks
• one CPU	highly parallel processing
•fast processing units	slow processing units
•reliable units	unreliable units
•static infrastructure	dynamic infrastructure

Why Artificial Neural Networks?

•There are two basic reasons why we are interested in building artificial neural networks (ANNs):

- **Technical viewpoint:** Some problems such as character recognition or the prediction of future states of a system require massively parallel and adaptive processing.
- **Biological viewpoint:** ANNs can be used to replicate and simulate components of the human (or animal) brain, thereby giving us insight into natural information processing.

Artificial Neural Networks

- The "building blocks" of neural networks are the **neurons**.
  - In technical systems, we also refer to them as **units** or **nodes**.
- Basically, each neuron
  - receives **input** from many other neurons.
  - changes its internal state (**activation**) based on the current input.
  - sends **one output signal** to many other neurons, possibly including its input neurons (recurrent network).

Artificial Neural Networks

- Information is transmitted as a series of electric impulses, so-called **spikes**.
- The **frequency** and **phase** of these spikes encodes the information.
- In biological systems, one neuron can be connected to as many as **10,000** other neurons.
- Usually, a neuron receives its information from other neurons in a confined area, its so-called **receptive field**.

How do ANNs work?

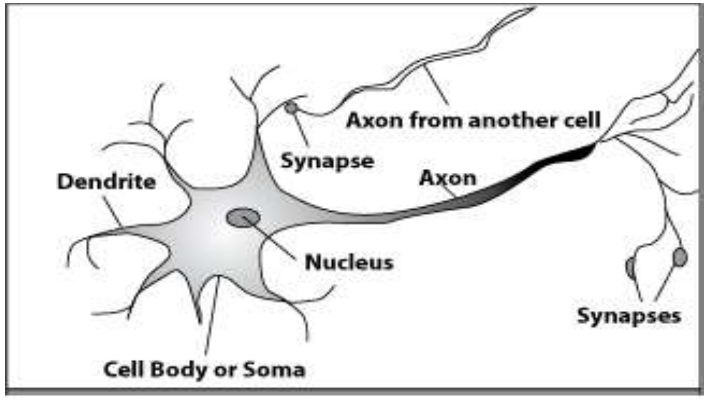
- An artificial neural network (ANN) is either a hardware implementation or a computer program which strives to simulate the information processing capabilities of its biological exemplar. ANNs are typically composed of a great number of interconnected artificial neurons. The artificial neurons are simplified models of their biological counterparts.
- ANN is a technique for solving problems by constructing software that works like our brains.

How do our brains work?

- The Brain is A massively parallel information processing system.
- Our brains are a huge network of processing elements. A typical brain contains a network of 10 billion neurons.

How do our brains work?

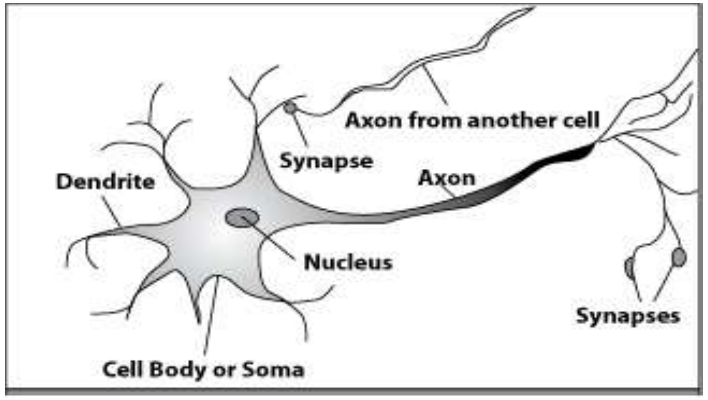
- A processing element



Dendrites: Input
Cell body: Processor
Synaptic: Link
Axon: Output

How do our brains work?

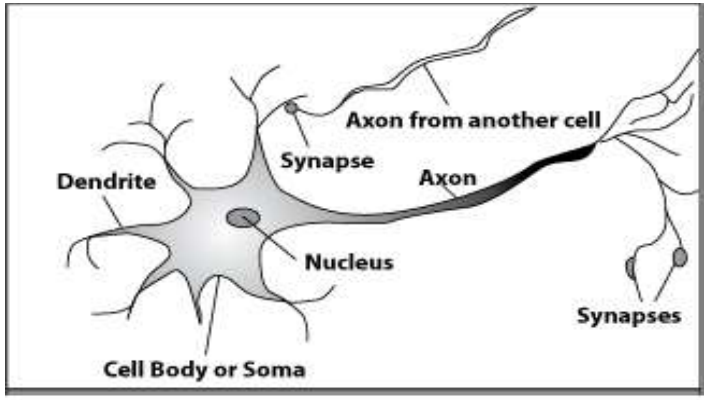
- A processing element



A neuron is connected to other neurons through about *10,000 synapses*

How do our brains work?

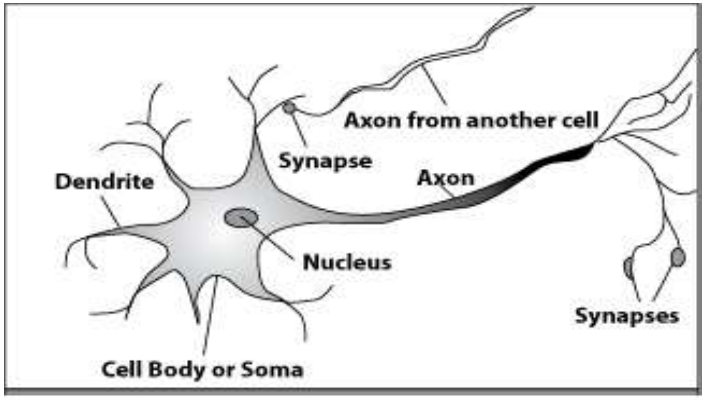
- A processing element



A neuron receives input from other neurons. Inputs are combined.

How do our brains work?

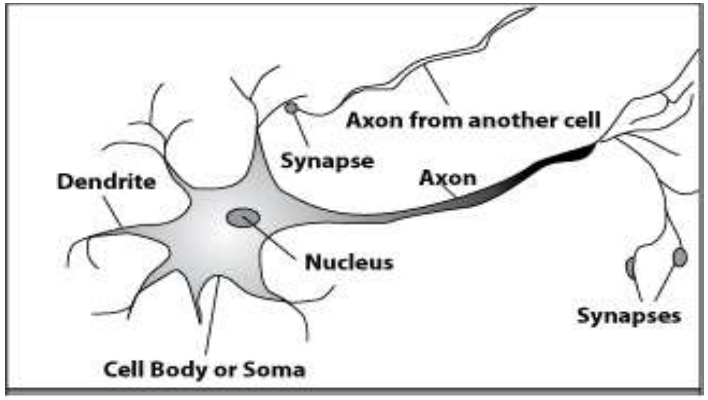
- A processing element



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)

How do our brains work?

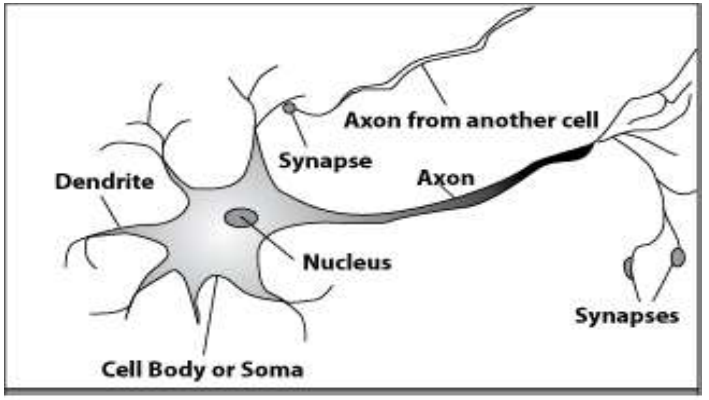
- A processing element



The axon endings almost touch the dendrites or cell body of the next neuron.

How do our brains work?

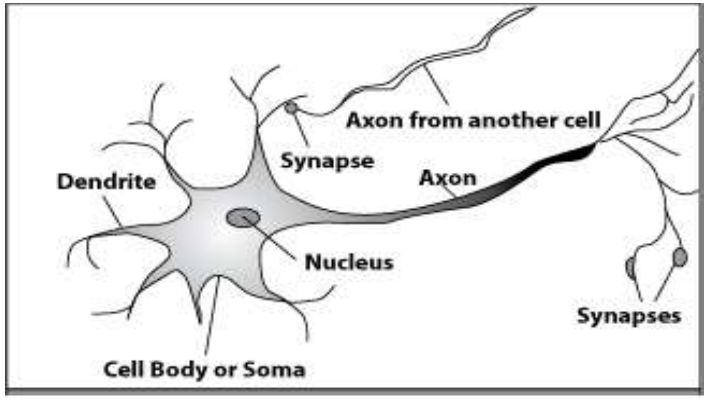
- A processing element



Transmission of an electrical signal from one neuron to the next is effected by neurotransmitters.

How do our brains work?

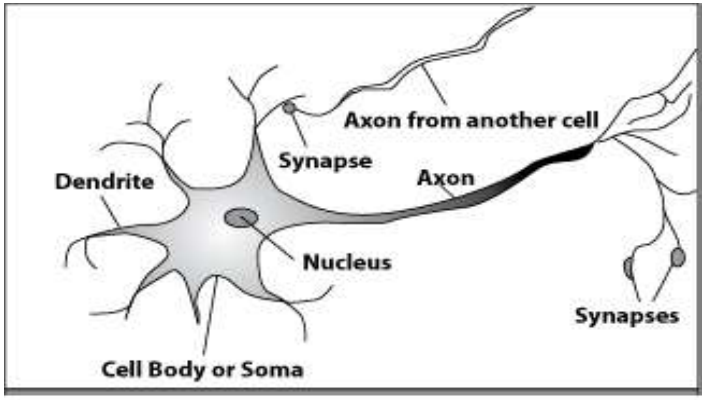
- A processing element



Neurotransmitters are chemicals which are released from the first neuron and which bind to the
Second.

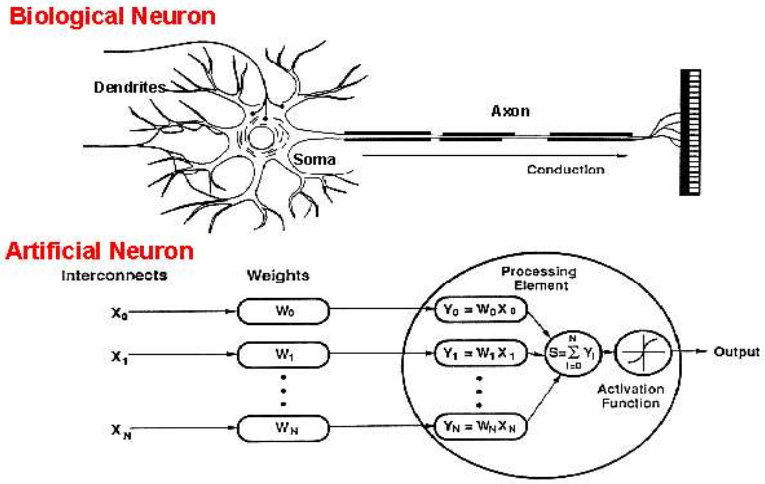
How do our brains work?

- A processing element



This link is called a synapse. The strength of the signal that reaches the next neuron depends on factors such as the amount of neurotransmitter available.

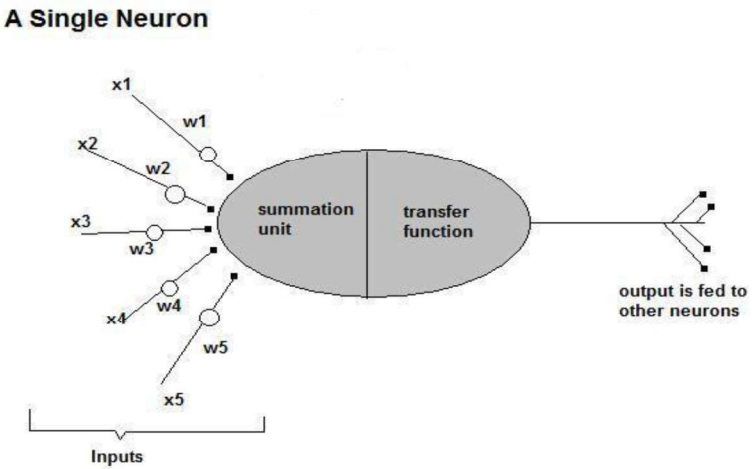
How do ANNs work?



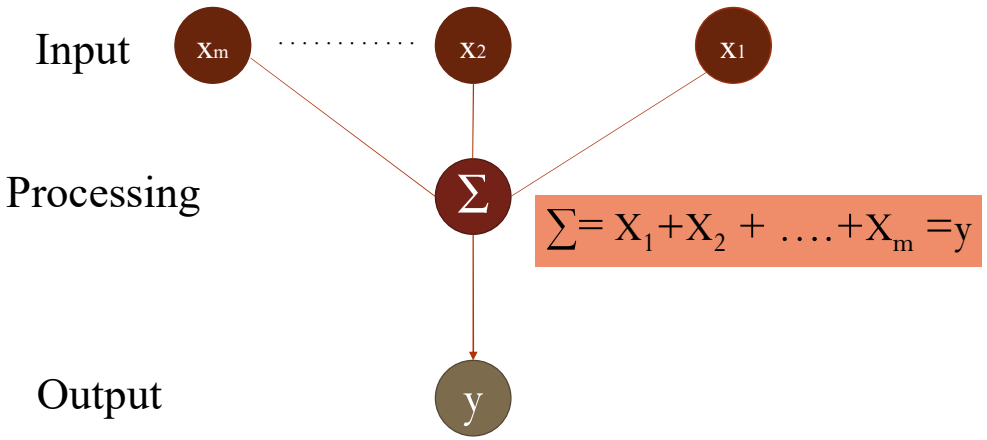
An artificial neuron is an imitation of a human neuron

How do ANNs work?

• Now, let us have a look at the model of an artificial neuron.

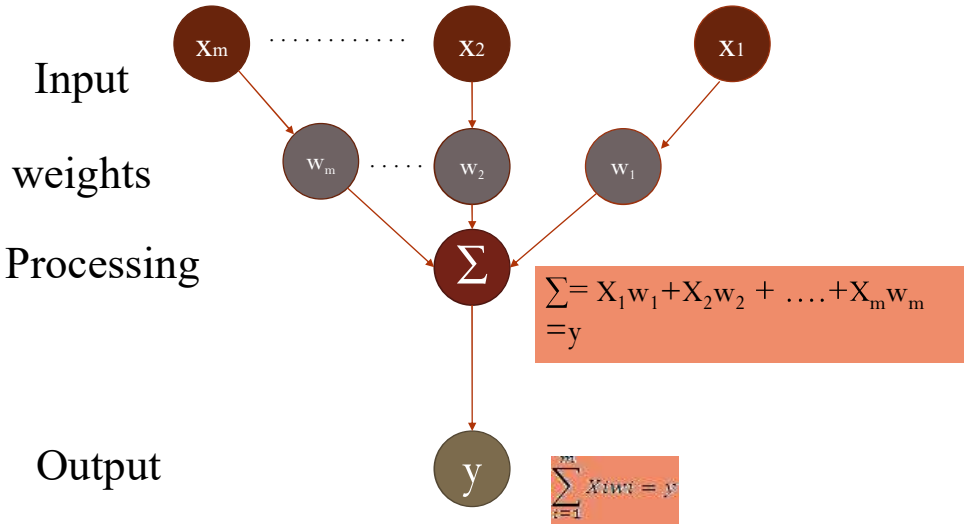


How do ANNs work?



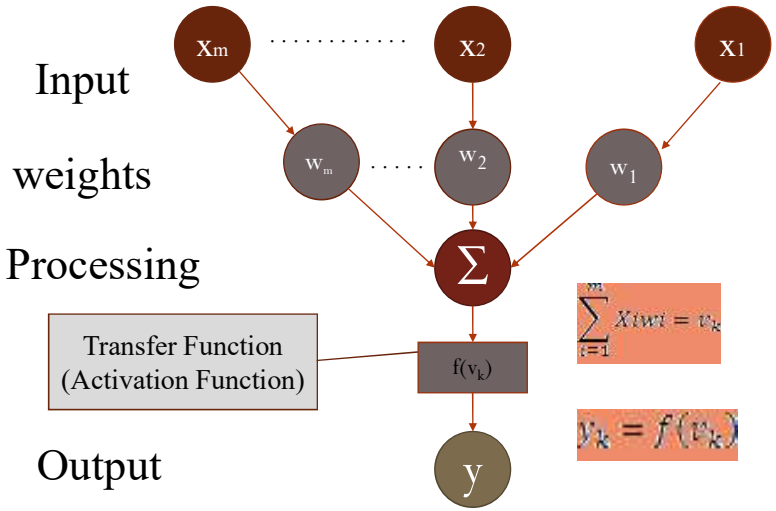
How do ANNs work?

Not all inputs are equal

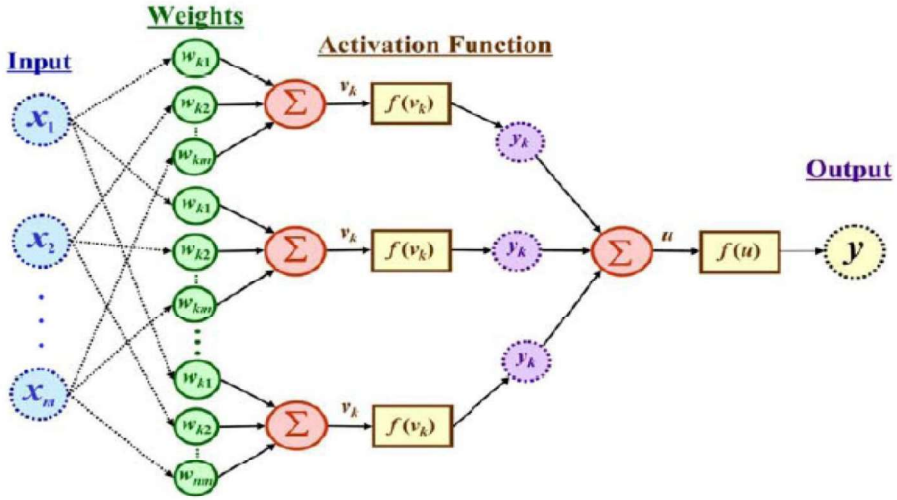


How do ANNs work?

The signal is not passed down to the next neuron verbatim



The output is a function of the input, that is affected by the weights, and the transfer functions



Artificial Neural Networks

- An ANN can:
  1. compute *any computable* function, by the appropriate selection of the network topology and weights values.
  2. learn from experience!
     - Specifically, by trial-and-error

Learning by trial-and-error

Continuous process of:

- Trial:

  Processing an input to produce an output (In terms of ANN: Compute the output function of a given input)

- Evaluate:

Evaluating this output by comparing the actual output with the expected output.

- Adjust:

  Adjust the *weights*.

How it works?

- Set initial values of the weights randomly.
- Input: truth table of the XOR
- Do
  - Read input (e.g. 0, and 0)
  - Compute an output (e.g. 0.60543)
  - Compare it to the expected output. (Diff= 0.60543)
  - Modify the weights *accordingly.*
- Loop until a condition is met
  - Condition: certain number of iterations
  - Condition: error threshold

Design Issues

- Initial weights (small random values $\in[-1,1]$)
- Transfer function (How the inputs and the weights are combined to produce output?)
- Error estimation
- Weights adjusting
- Number of neurons
- Data representation
- Size of training set

Transfer Functions

- **Linear:** The output is proportional to the total weighted input.
- **Threshold:** The output is set at one of two values, depending on whether the total weighted input is greater than or less than some threshold value.
- **Non-linear:** The output varies continuously but not linearly as the input changes.

Error Estimation

- The **root mean square error (RMSE)** is a frequently-used measure of the differences between values predicted by a model or an estimator and the values actually observed from the thing being modeled or estimated

Weights Adjusting

- After each iteration, weights should be adjusted to minimize the error.
  - All possible weights
  - Back propagation

Back Propagation

- Back-propagation is an example of supervised learning is used at each layer to minimize the error between the layer's response and the actual data
- The error at each hidden layer is an average of the evaluated error
- Hidden layer networks are trained this way

Back Propagation

- N is a neuron.
- N_w is one of N's inputs weights
- N_{out} is N's output.
- $N_w = N_w + \Delta N_w$
- $\Delta N_w = N_{out} * (1 - N_{out}) * N_{ErrorFactor}$
- $N_{ErrorFactor} = N_{ExpectedOutput} - N_{ActualOutput}$
- This works only for the last layer, as we can know the actual output, and the expected output.

Number of neurons

- Many neurons:
  - Higher accuracy
  - Slower
  - Risk of over-fitting
    - Memorizing, rather than understanding
    - The network will be useless with new problems.
- Few neurons:
  - Lower accuracy
  - Inability to learn at all
- Optimal number.

Data representation

- Usually input/output data needs pre-processing
- Pictures
  - Pixel intensity
- Text:
  - A pattern

Size of training set

- No one-fits-all formula
- Over fitting can occur if a "good" training set is not chosen
- What constitutes a "good" training set?
  - Samples must represent the general population.
  - Samples must contain members of each class.
  - Samples in each class must contain a wide range of variations or noise effect.
- The size of the training set is related to the number of hidden neurons

Learning Paradigms

- Supervised learning
- Unsupervised learning

Supervised learning

- This is what we have seen so far!
- A network is fed with a set of training samples (inputs and corresponding output), and it uses these samples to learn the general relationship between the inputs and the outputs.
- This relationship is represented by the values of the weights of the trained network.

Unsupervised learning

- No desired output is associated with the training data!
- Faster than supervised learning
- Used to find out *structures within data:*
  - Clustering
  - Compression

Applications Areas

- Function approximation
  - including time series prediction and modeling.
- Classification
  - including patterns and sequences recognition, novelty detection and sequential decision making.
    - (radar systems, face identification, handwritten text recognition)
- Data processing
  - including filtering, clustering blinds source separation and compression.
    - (data mining, e-mail Spam filtering)

Advantages / Disadvantages

- Advantages
  - Adapt to unknown situations
  - Powerful, it can model complex functions.
  - Ease of use, learns by example, and very little user domain-specific expertise needed
- Disadvantages
  - Forgets
  - Not exact
  - Large complexity of the network structure

Conclusion

- Artificial Neural Networks are an imitation of the biological neural networks, but much simpler ones.
- The computing would have a lot to gain from neural networks. Their ability to learn by example makes them very flexible and powerful furthermore there is need to device an algorithm in order to perform a specific task.

Conclusion

- Neural networks also contributes to area of research such a neurology and psychology. They are regularly used to model parts of living organizations and to investigate the internal mechanisms of the brain.
- Many factors affect the performance of ANNs, such as the transfer functions, size of training sample, network topology, weights adjusting algorithm, …

References

- Introduction to Artificial Neural Networks, Nicolas Galoppo von Borries
- Tom M. Mitchell, *Machine Learning, WCB McGraw-Hill, Boston, 1997.*

Neural Network

- Neural Network is a set of connected INPUT/OUTPUT UNITS, where each connection has a WEIGHT associated with it.
- Neural Network learning is also called CONNECTIONIST learning due to the connections between units.
- It is a case of SUPERVISED, INDUCTIVE or CLASSIFICATION learning.

Neural Network

- Neural Network learns by adjusting the weights so as to be able to correctly classify the training data and hence, after testing phase, to classify unknown data.
- Neural Network needs long time for training.
- Neural Network has a high tolerance to noisy and incomplete data

Neural Network Classifier

- **Input: Classification data**
  It contains classification attribute
- Data is divided, as in any classification problem.
  [Training data and Testing data]
- **All data must be normalized.**
  (i.e. all values of attributes in the database are changed to contain values in the internal [0,1] or[-1,1])
  Neural Network can work with data in the range of (0,1) or (-1,1)
- **Two basic normalization techniques**
  [1] Max-Min normalization
  [2] Decimal Scaling normalization

Data Normalization

[1] Max- Min normalization formula is as follows:

$$v' = \frac{v - \min A}{\max A - \min A}(new\_\max A - new\_\min A) + new\_\min A$$

[minA, maxA , the minimun and maximum values of the attribute A max-min normalization maps a value v of A to v' in the range {new_minA, new_maxA}]

Example of Max-Min Normalization

Max- Min normalization formula

$$v' = \frac{v - \min A}{\max A - \min A}(new\_\max A - new\_\min A) + new\_\min A$$

Example: We want to normalize data to range of the interval [0,1].
We put: new_max A= 1, new_minA =0.

Say, max A was 100 and min A was 20 (That means maximum and minimum values for the attribute).

Now, if v = 40 (If for this particular pattern , attribute value is 40), v' will be calculated as , v' = (40-20) x (1-0) / (100-20) + 0
=> v' = 20 x 1/80
=> v' = 0.4

Decimal Scaling Normalization

[2]Decimal Scaling Normalization

Normalization by decimal scaling normalizes by moving the decimal point of values of attribute A.

$$v' = \frac{v}{10^j}$$

Here j is the smallest integer such that max|v'|<1.

Example :

A – values range from -986 to 917. Max |v| = 986.

v = -986 normalize to v' = -986/1000 = -0.986

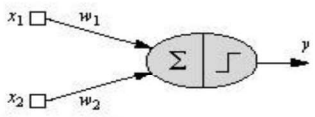


Fig1: an artificial neuron

One Neuron as a Network

- Here x1 and x2 are normalized attribute value of data.
- y is the output of the neuron , i.e the class label.
- x1 and x2 values multiplied by weight values w1 and w2 are input to the neuron x.
- Value of x1 is multiplied by a weight w1 and values of x2 is multiplied by a weight w2.
- Given that
  - w1 = 0.5 and w2 = 0.5
  - Say value of x1 is 0.3 and value of x2 is 0.8,
  - So, weighted sum is :
  - sum= w1 x x1 + w2 x x2 = 0.5 x 0.3 + 0.5 x 0.8 = 0.55

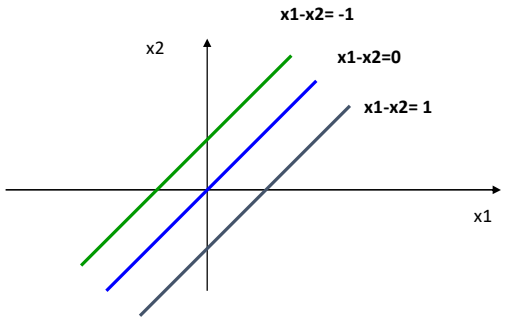
One Neuron as a Network

- The neuron receives the weighted sum as input and calculates the output as a function of input as follows :
- y = f(x) , where f(x) is defined as
- f(x) = 0 { when x< 0.5 }
- f(x) = 1 { when x >= 0.5 }
- For our example, x (weighted sum) is 0.55, so y = 1 ,
- That means corresponding input attribute values are classified in class 1.
- If for another input values , x = 0.45 , then f(x) = 0,
- so we could conclude that input values are classified to class 0.

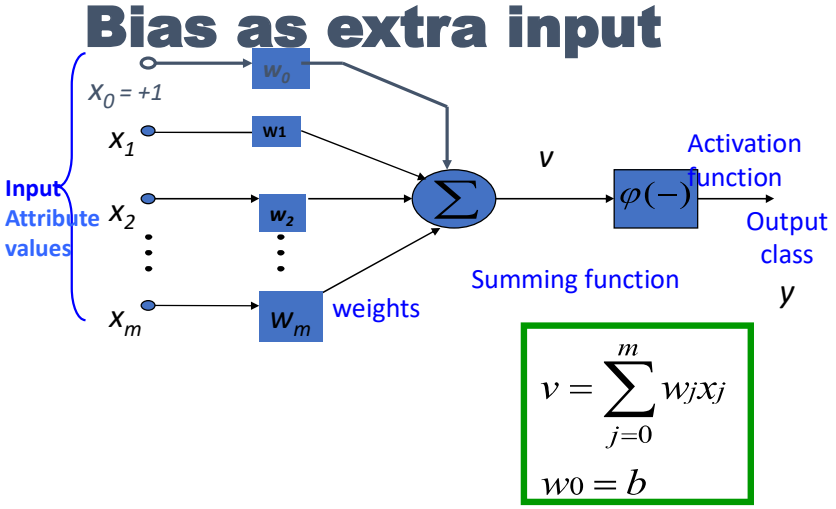
Bias of a Neuron

- We need the bias value to be added to the weighted sum ∑wixi so that we can transform it from the origin.

v = ∑wixi + b, here b is the bias



Bias as extra input



Neuron with Activation

- The neuron is the basic information processing unit of a NN. It consists of:

1 A set of links, describing the neuron inputs, with weights W_1, W_2, …, W_m

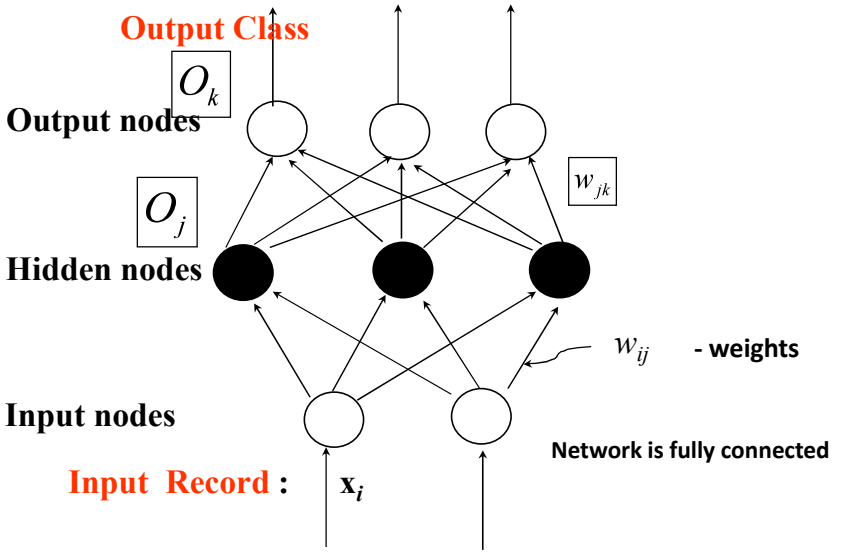
2. An adder function (linear combiner) for computing the weighted sum of the inputs (real numbers):

$$u = \sum_{j=1}^{m} w_j x_j$$

3 Activation function : for limiting the amplitude of the neuron output.

$$y = \varphi(u + b)$$

A Multilayer Feed-Forward Neural Network



Neural Network Learning

- The inputs are fed simultaneously into the input layer.
- The weighted outputs of these units are fed into hidden layer.
- The weighted outputs of the last hidden layer are inputs to units making up the output layer.

A Multilayer Feed Forward Network

- The units in the hidden layers and output layer are sometimes referred to as neurodes, due to their symbolic biological basis, or as output units.
- A network containing two hidden layers is called a three-layer neural network, and so on.
- The network is feed-forward in that none of the weights cycles back to an input unit or to an output unit of a previous layer.

A Multilayered Feed – Forward Network

- INPUT: records without class attribute with normalized attributes values.
- INPUT VECTOR: X = { x1, x2, …. xn}

  where n is the number of (non class) attributes.
- INPUT LAYER – there are as many nodes as non-class attributes i.e. as the length of the input vector.
- HIDDEN LAYER – the number of nodes in the hidden layer and the number of hidden layers depends on implementation.

A Multilayered Feed–Forward Network

- **OUTPUT LAYER** – corresponds to the class attribute.
- There are as many nodes as classes (values of the class attribute).

O_k k= 1, 2,.. #classes

- Network is fully connected, i.e. each unit provides input to each unit in the next forward layer.

Classification by Back propagation

- *Back Propagation learns by iteratively processing a set of training data (samples).*
- For each sample, weights are modified to minimize the error between network's classification and actual classification.

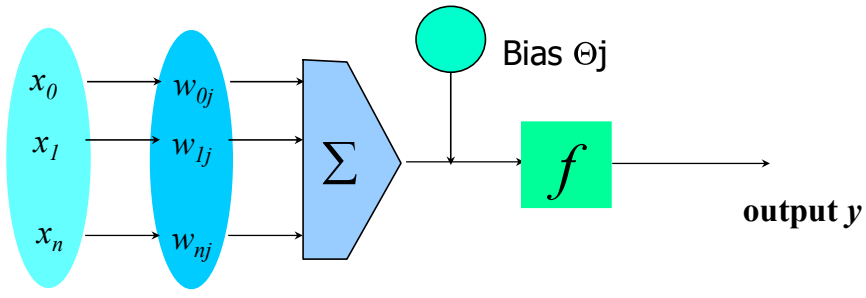
Steps in Back propagation Algorithm

- STEP ONE: initialize the weights and biases.
- The weights in the network are initialized to random numbers from the interval [-1,1].
- Each unit has a BIAS associated with it
- The biases are similarly initialized to random numbers from the interval [-1,1].
- STEP TWO: feed the training sample.

Steps in Back propagation Algorithm (cont..)

- STEP THREE: Propagate the inputs forward; we compute the net input and output of each unit in the hidden and output layers.
- STEP FOUR: back propagate the error.
- STEP FIVE: update weights and biases to reflect the propagated errors.
- STEP SIX: terminating conditions.

Propagation through Hidden Layer (One Node)



- The inputs to unit j are outputs from the previous layer. These are multiplied by their corresponding weights in order to form a weighted sum, which is added to the bias associated with unit j.
- A nonlinear activation function f is applied to the net input.

Propagate the inputs forward

- For unit j in the input layer, its output is equal to its input, that is,

$$O_j = I_j$$

for input unit j.

- The net input to each unit in the hidden and output layers is computed as follows.
- Given a unit j in a hidden or output layer, the net input is

$$I_j = \sum_i w_{ij} O_i + \theta_j$$

where wij is the weight of the connection from unit i in the previous layer to unit j;
Oi is the output of unit I from the previous layer;

θ_j is the bias of the unit

Propagate the inputs forward

- Each unit in the hidden and output layers takes its net input and then applies an activation function. The function symbolizes the activation of the neuron represented by the unit. It is also called a logistic, sigmoid, or squashing function.
- Given a net input Ij to unit j, then Oj = f(Ij), the output of unit j, is computed as

$$O_j = \frac{1}{1+e^{-I_j}}$$

Back propagate the error

- When reaching the Output layer, the error is computed and propagated backwards.
- For a unit k in the output layer the error is computed by a formula:

$$Err_k = O_k(1-O_k)(T_k - O_k)$$

Where O k – actual output of unit k (computed by activation function.

$$O_k = \frac{1}{1+e^{-I_k}}$$

Tk – True output based of known class label; classification of training sample

Ok(1-Ok) – is a Derivative (rate of change) of activation function.

Back propagate the error

- The error is propagated backwards by updating weights and biases to reflect the error of the network classification .
- For a unit j in the hidden layer the error is computed by a formula:

$$Err_j = O_j(1-O_j)\sum_k Err_k w_{jk}$$

where wjk is the weight of the connection from unit j to unit k in the next higher layer, and Errk is the error of unit k.

Update weights and biases

- Weights are updated by the following equations, where l is a constant between 0.0 and 1.0 reflecting the learning rate, this learning rate is fixed for implementation.

$$\Delta w_{ij} = (l) Err_j O_i$$

$$w_{ij} = w_{ij} + \Delta w_{ij}$$

- Biases are updated by the following equations

$$\Delta \theta_j = (l) Err_j$$

$$\theta_j = \theta_j + \Delta \theta_j$$

Update weights and biases

- We are updating weights and biases after the presentation of each sample.
- This is called case updating.

- Epoch --- One iteration through the training set is called an epoch.

- **Epoch updating ----------**
- **Alternatively, the weight and bias increments could be accumulated in variables and the weights and biases updated after all of the samples of the training set have been presented.**

- **Case updating is more accurate**

Terminating Conditions

- Training stops

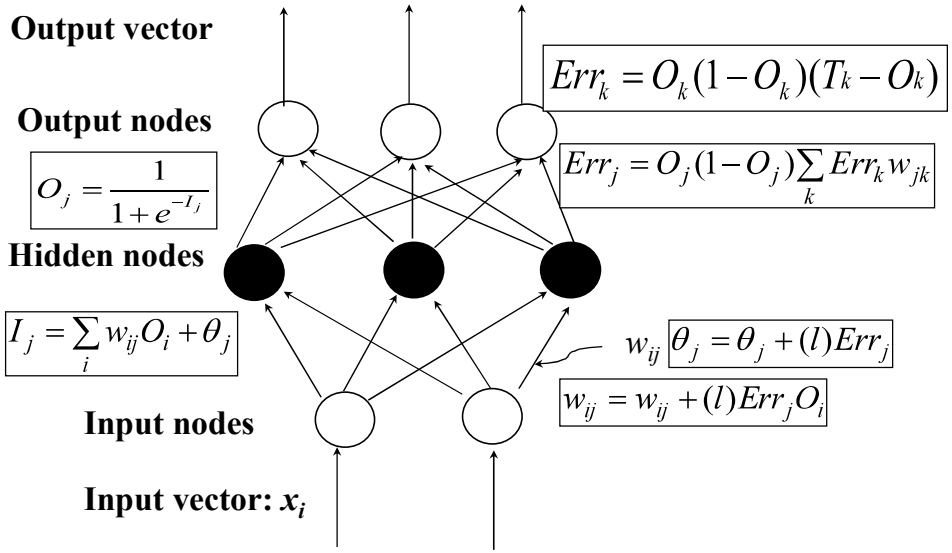
- All Δw_{ij} in the previous epoch are below some threshold, or

- The percentage of samples misclassified in the previous epoch is below some threshold, or

- a pre specified number of epochs has expired.

- In practice, several hundreds of thousands of epochs may be required before the weights will converge.

Backpropagation Formulas



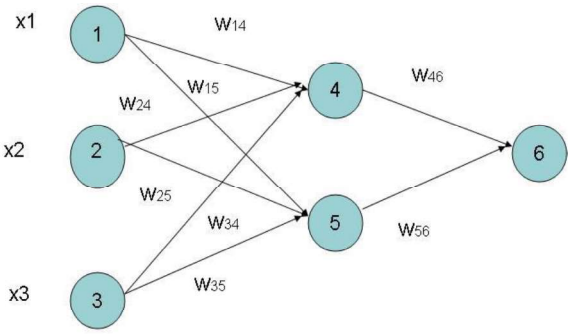
Example of Back propagation

Input = 3, Hidden Neuron = 2 Output =1

Initialize weights :

Random Numbers from -1.0 to 1.0

Initial Input and weight

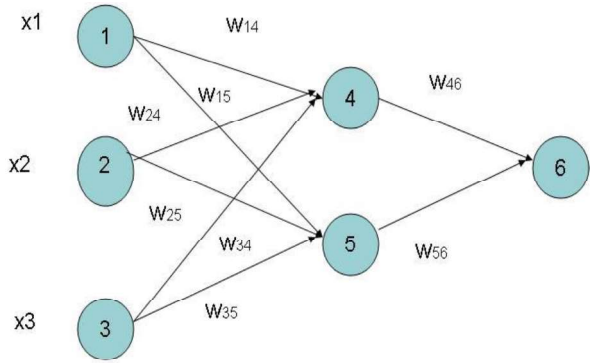


x1	x2	x3	w_{14}	w_{15}	w_{24}	w_{25}	w_{34}	w_{35}	w_{46}	w_{56}
1	0	1	0.2	-0.3	0.4	0.1	-0.5	0.2	-0.3	-0.2

Example (cont..)

- **Bias added to Hidden**
- **+ Output nodes**
- **Initialize Bias**
- **Random Values from**
- **-1.0 to 1.0**

- **Bias (Random)**



θ_4	θ_5	θ_6
-0.4	0.2	0.1

Net Input and Output Calculation

Unit j	Net Input I_j	Output O_j
4	**0.2 + 0 + 0.5 -0.4 = -0.7**	$O_j = \frac{1}{1+e^{0.7}}$ = 0.332
5	**-0.3 + 0 + 0.2 + 0.2 =0.1**	$O_j = \frac{1}{1+e^{-0.1}}$ = 0.525
6	**(-0.3)0.332-(0.2)(0.525)+0.1= -0.105**	$O_j = \frac{1}{1+e^{0.105}}$ = 0.475

Calculation of Error at Each Node

Unit j	Error j
6	**0.475(1-0.475)(1-0.475) =0.1311** **We assume $T_6 = 1$**
5	**0.525 x (1- 0.525)x 0.1311x (-0.2) = 0.0065**
4	**0.332 x (1-0.332) x 0.1311 x (-0.3) = -0.0087**

Calculation of weights and Bias Updating

Learning Rate l =0.9

Weight	New Values
w_{46}	-0.3 + 0.9(0.1311)(0.332) = -0.261
w_{56}	-0.2 + (0.9)(0.1311)(0.525) = -0.138
w_{14}	0.2 + 0.9(-0.0087)(1) = 0.192
w_{15}	-0.3 + (0.9)(-0.0065)(1) = -0.306
……..similarly	……similarly
θ_6	0.1 +(0.9)(0.1311)=0.218
……..similarly	……similarly

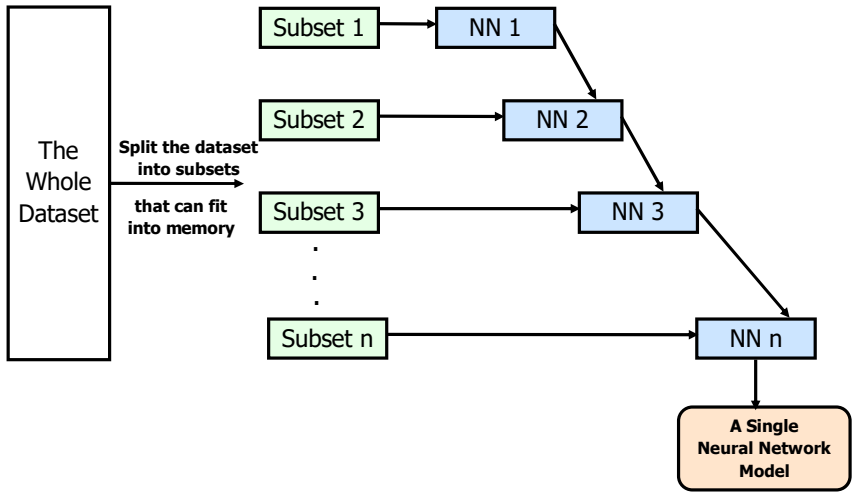
Variants of Neural Networks Learning

- Supervised learning/Classification
  - Control
  - Function approximation
  - Associative memory
- Unsupervised learning or Clustering

Training with Subsets

- Select subsets of data
- Build new classifier on subset
- Aggregate with previous classifiers
- Compare error after adding classifier
- Repeat as long as error decreases

Training with subsets



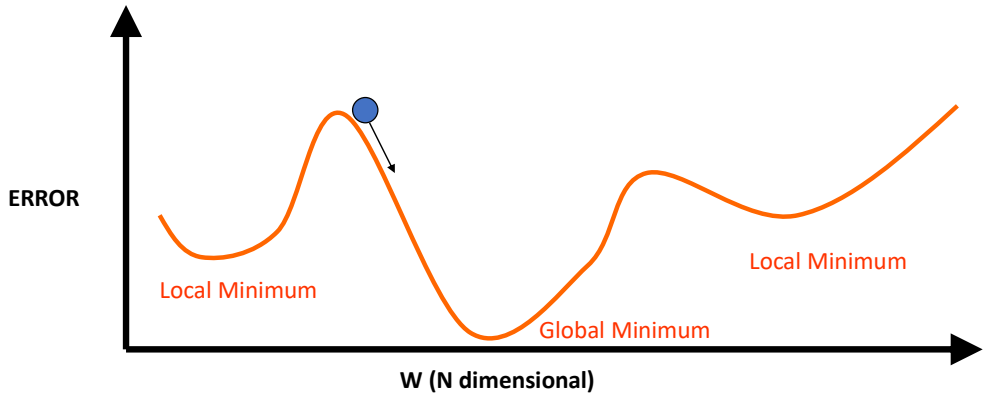
Modular Neural Network

- Modular Neural Network
  - Made up of a combination of several neural networks.

The idea is to reduce the load for each neural network as opposed to trying to solve the problem on a single neural network.

Training Process of the MLP

- The training will be continued until the RMS is minimized.



Applications-I

- Handwritten Digit Recognition
- Face recognition
- Time series prediction
- Process identification
- Process control
- Optical character recognition

Application-II

- Forecasting/Market Prediction: finance and banking
- Manufacturing: quality control, fault diagnosis
- Medicine: analysis of electrocardiogram data, RNA & DNA sequencing, drug development without animal testing

- Control: process, robotics

Summary

- We presented mainly the followings-------
- Basic building block of Artificial Neural Network.
- Construction , working and limitation of single layer neural network (Single Layer Neural Network).
- Back propagation algorithm for multi layer feed forward NN.
- Some Advanced Features like training with subsets, Quicker convergence, Modular Neural Network, Evolution of NN.
- Application of Neural Network.

Deep Learning-11

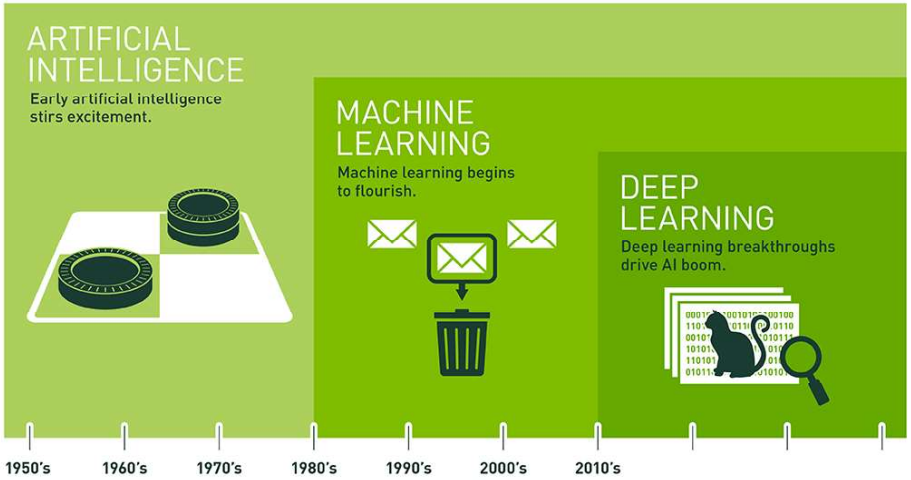


Image from https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

"Traditional" machine learning:

Deep, "end-to-end" learning:

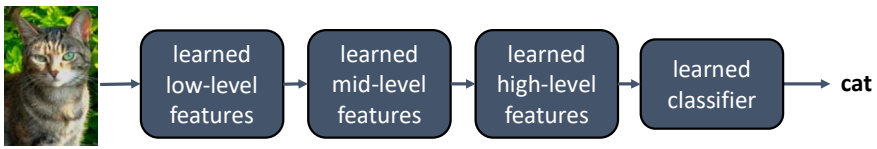


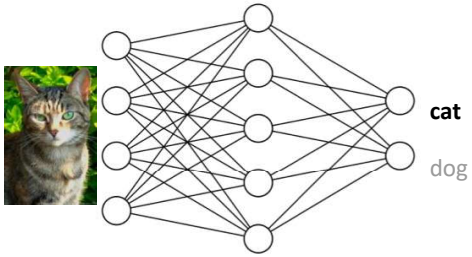
Table 1: Major milestones that will be covered in this paper

Year	Contributer	Contribution
300 BC	Aristotle	introduced Associationism, started the history of human's attempt to understand brain.
1873	Alexander Bain	introduced Neural Groupings as the earliest models of neural network, inspired Hebbian Learning Rule.
1943	McCulloch & Pitts	introduced MCP Model, which is considered as the ancestor of Artificial Neural Model.
1949	Donald Hebb	considered as the father of neural networks, introduced Hebbian Learning Rule, which lays the foundation of modern neural network.
1958	Frank Rosenblatt	introduced the first perceptron, which highly resembles modern perceptron.
1974	Paul Werbos	introduced Backpropagation
1980	Teuvo Kohonen	introduced Self Organizing Map
	Kunihiko Fukushima	introduced Neocogitron, which inspired Convolutional Neural Network
1982	John Hopfield	introduced Hopfield Network
1985	Hilton & Sejnowski	introduced Boltzmann Machine
1986	Paul Smolensky	introduced Harmonium, which is later known as Restricted Boltzmann Machine
	Michael I. Jordan	defined and introduced Recurrent Neural Network
1990	Yann LeCun	introduced LeNet, showed the possibility of deep neural networks in practice
1997	Schuster & Paliwal	introduced Bidirectional Recurrent Neural Network
	Hochreiter & Schmidhuber	introduced LSTM, solved the problem of vanishing gradient in recurrent neural networks
2006	Geoffrey Hinton	introduced Deep Belief Networks, also introduced layer-wise pretraining technique, opened current deep learning era.
2009	Salakhutdinov & Hinton	introduced Deep Boltzmann Machines
2012	Geoffrey Hinton	introduced Dropout, an efficient way of training neural networks

From: Wang & Raj: On the Origin of Deep Learning (2017)

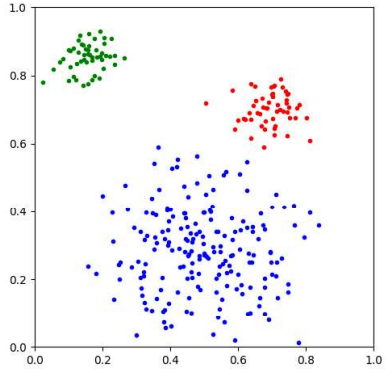
Main types of machine learning

- **Supervised learning**
- Unsupervised learning
- Self-supervised learning
- Reinforcement learning



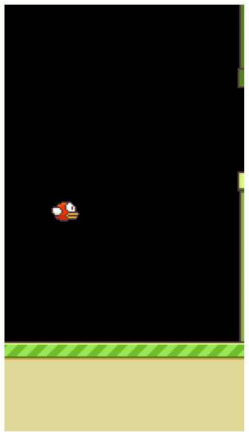
Main types of machine learning

- Supervised learning
- **Unsupervised learning**
- Self-supervised learning
- Reinforcement learning



Main types of machine learning

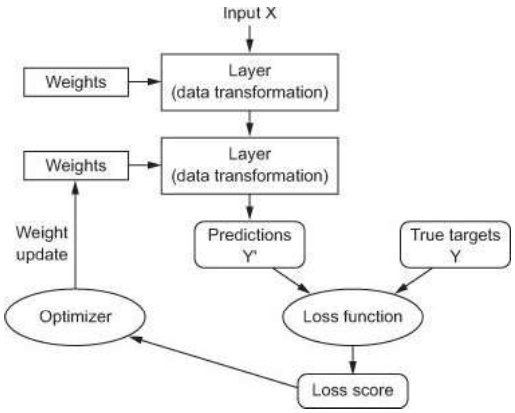
- Supervised learning
- Unsupervised learning
- Self-supervised learning
- **Reinforcement learning**



Animation from https://yanpanlau.github.io/2016/07/10/FlappyBird-Keras.html

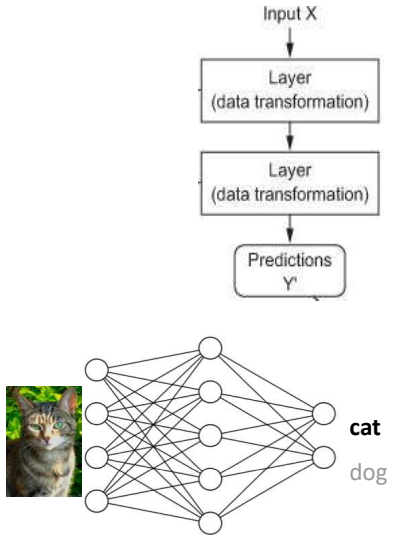
Anatomy of a deep neural network

- Layers
- Input data and targets
- Loss function
- Optimizer



Input data and targets

- The network maps the input data X to predictions Y'
- During training, the predictions Y' are compared to true targets Y using the loss function

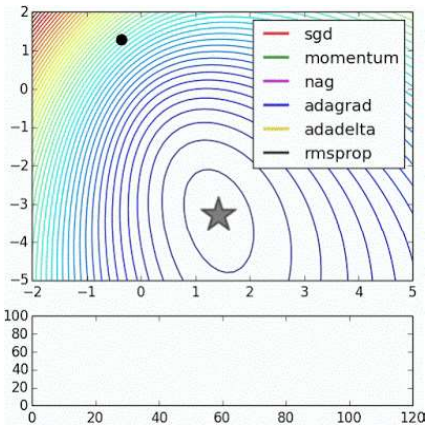


Loss function

- The quantity to be minimized (optimized) during training
  - the only thing **the network** cares about
  - there might also be other metrics **you** care about
- Common tasks have "standard" loss functions:
  - *mean squared error* for regression
  - *binary cross-entropy* for two-class classification
  - *categorical cross-entropy* for multi-class classification
  - etc.
- https://lossfunctions.tumblr.com/

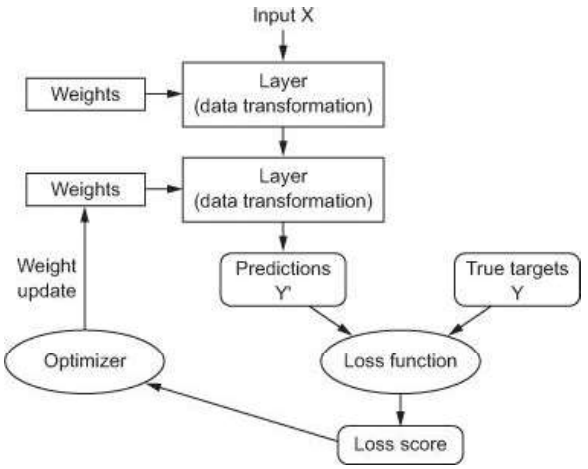
Optimizer

- How to update the weights based on the loss function
- *Learning rate (+scheduling)*
- Stochastic gradient descent, momentum, and their variants
  - RMSProp is usually a good first choice
  - more info: http://ruder.io/optimizing-gradient-descent/



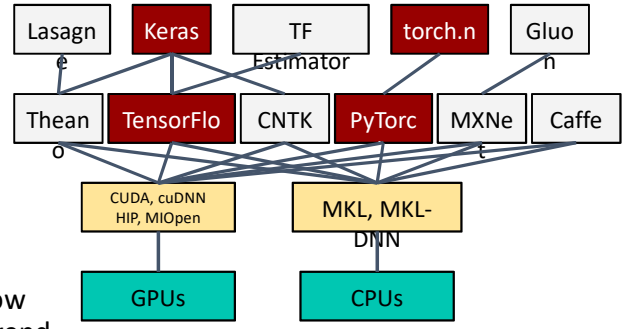
Animation from: https://imgur.com/s25RsOr

Anatomy of a deep neural network



Deep learning frameworks

Deep learning frameworks

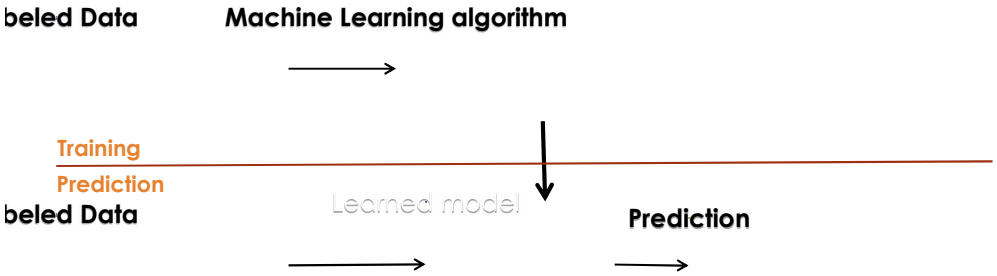


- Keras is a high-level neural networks API
  - we will use TensorFlow as the compute backend
  - included in TensorFlow 2 as `tf.keras`
  - https://keras.io/ , https://www.tensorflow.org/guide/keras
- PyTorch is:
  - a GPU-based tensor library
  - an efficient library for dynamic neural networks
  - https://pytorch.org/

Deep Learning-1

Machine Learning Basics

Machine learning is a field of computer science that gives computers the ability to **learn without being explicitly programmed**



Methods that can learn from and make predictions on data

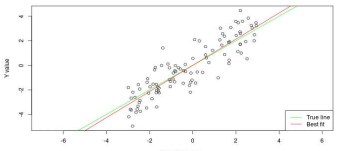
Types of Learning

Supervised: Learning with a **labeled training** set
Example: email *classification* with already labeled emails

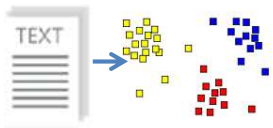
Unsupervised: Discover **patterns** in **unlabeled** data
Example: *cluster* similar documents based on text

Reinforcement learning: learn to **act** based on **feedback/reward**
Example: learn to play Go, reward: *win or lose*

Classification



Regression



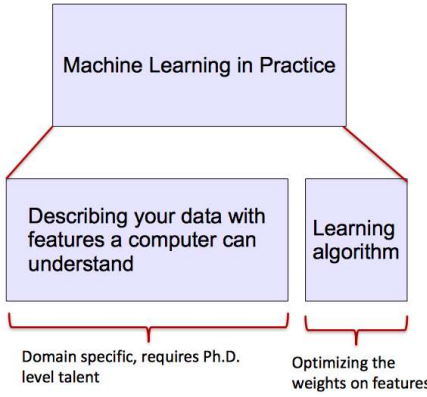
Clustering

Anomaly Detection
Sequence labeling
…

http://mbjoseph.github.io/2013/11/27/measure.html

ML vs. Deep Learning

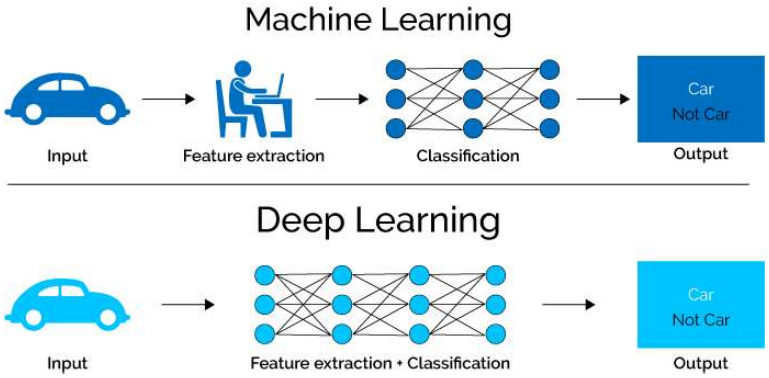
Most machine learning methods work well because of **human-designed representations** and **input features**
ML becomes just **optimizing weights** to best make a final prediction



Feature	NER
Current Word	✓
Previous Word	✓
Next Word	✓
Current Word Character n-gram	all
Current POS Tag	✓
Surrounding POS Tag Sequence	✓
Current Word Shape	✓
Surrounding Word Shape Sequence	✓
Presence of Word in Left Window	size 4
Presence of Word in Right Window	size 4

What is Deep Learning (DL) ?

A machine learning subfield of learning **representations** of data.
Exceptional effective at **learning patterns**.
Deep learning algorithms attempt to learn (multiple levels of) representation by using a **hierarchy of multiple layers**
If you provide the system **tons of information**, it begins to understand it and respond in useful ways.

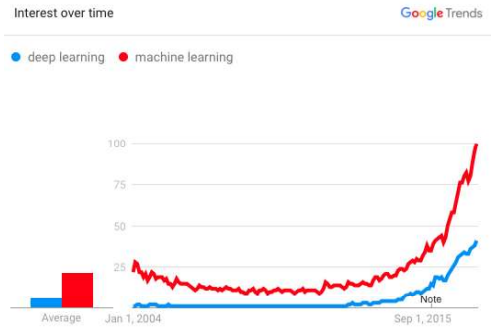


https://www.xenonstack.com/blog/static/public/uploads/media/machine-learning-vs-deep-learning.png

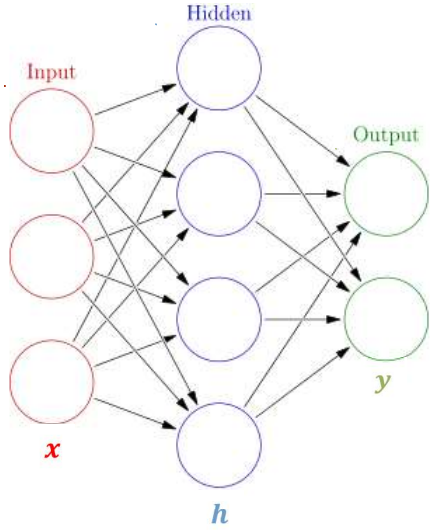
Why is DL useful?

- Manually designed features are often **over-specified**, **incomplete** and take a **long time to design** and validate
- Learned Features are **easy to adapt**, **fast** to learn
- Deep learning provides a very **flexible**, (almost?) **universal**, learnable framework for representing world, visual and linguistic information.
- Can learn both unsupervised and supervised
- Effective **end-to-end** joint system learning
- Utilize large amounts of training data

In ~2010 DL started outperforming other ML techniques
first in speech and vision, then NLP



Neural Network Intro



Weights

$$y = \sigma(W_2 h + b_2)$$

Activation functions

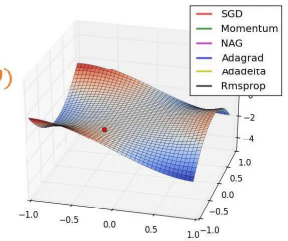
How do we train?

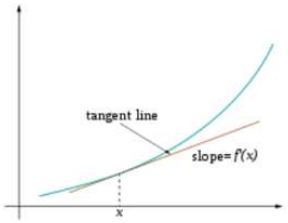
4 + 2 = 6 neurons (not counting inputs)
[3 x 4] + [4 x 2] = 20 weights
4 + 2 = 6 biases
26 learnable **parameters**

Training

Sample labeled data (**batch**) → **Forward** it through the network, get predictions → **Back-propagate** the errors → **Update** the network weights

Optimize (min. or max.) **objective/cost function** $J(\theta)$
Generate **error signal** that measures difference between predictions and target values





Use error signal to change the **weights** and get more accurate predictions
Subtracting a fraction of the **gradient** moves you towards the **(local) minimum of the cost function**

https://medium.com/@ramrajchandradevan/the-evolution-of-gradient-descend-optimization-algorithm-4106a6702d39

One forward pass

Text (input) representation
TFIDF
Word embeddings
….

0.2	-0.5	0.1
2.0	1.5	1.3
0.5	0.0	0.25
-0.3	2.0	0.0

$\mathbf{W}$

0.1
0.2
0.3

x_i

1.0
3.0
0.025
0.0

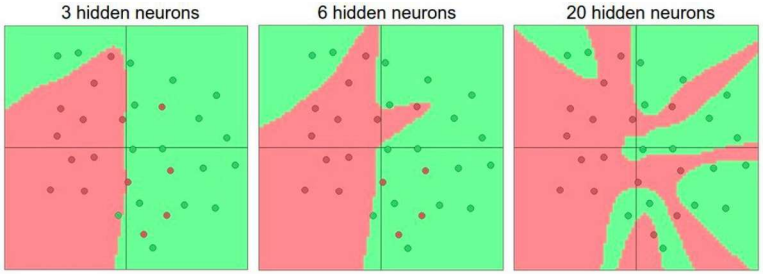
b

0.95	very positive
3.89	positive
0.15	negative
0.37	very negative

$\sigma(x_i; W, b)$

Activation functions

Non-linearities needed to learn complex (non-linear) representations of data, otherwise the NN would be just a linear function $W_1 W_2 x = Wx$

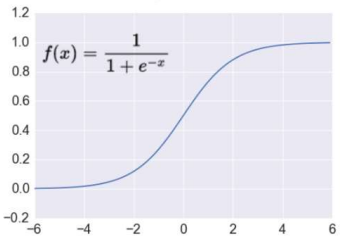


http://cs231n.github.io/assets/nn1/layer_sizes.jpeg

More layers and neurons can approximate more complex functions

Full list:

Activation: Sigmoid



http://adilmoujahid.com/images/activation.png

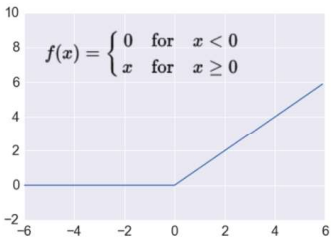
Takes a real-valued number and "squashes" it into range between 0 and 1.

$$R^n \rightarrow [0,1]$$

\+ Nice interpretation as the **firing rate** of a neuron
- 0 = not firing at all
- 1 = fully firing

\- Sigmoid neurons **saturate** and **kill gradients**, thus NN will barely learn
- when the neuron's activation are 0 or 1 (saturate)
  - gradient at these regions almost zero
  - almost no signal will flow to its weights
  - if initial weights are too large then most neurons would saturate

Activation: ReLU



http://adilmoujahid.com/images/activation.png

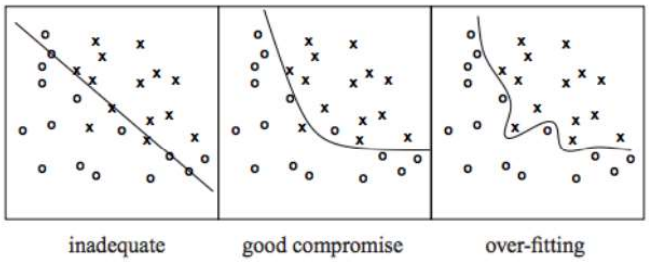
Takes a real-valued number and thresholds it at zero $f(x) = \max(0, x)$

$$R^n \rightarrow R_+^n$$

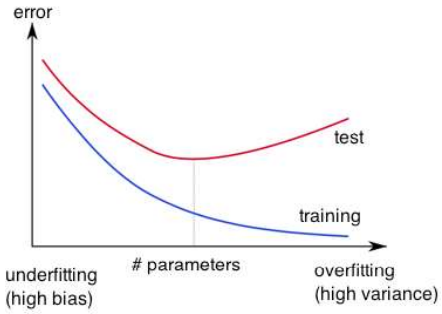
Most Deep Networks use ReLU nowadays

- Trains much **faster**
  - accelerates the convergence of SGD
  - due to linear, non-saturating form
- Less expensive operations
  - compared to sigmoid/tanh (exponentials etc.)
  - implemented by simply thresholding a matrix at zero
- More **expressive**
- Prevents the **gradient vanishing problem**

Overfitting

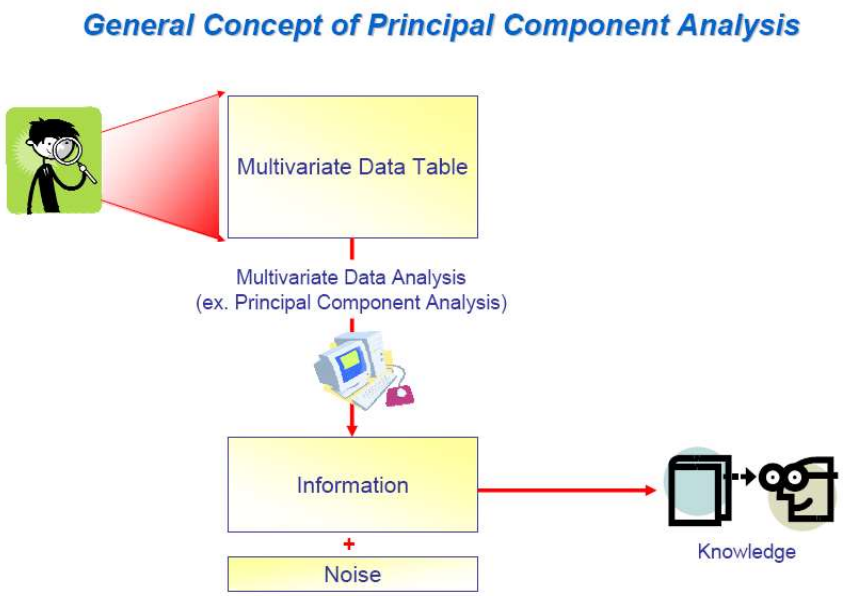


http://wiki.bethanycrane.com/overfitting-of-data



https://www.neuraldesigner.com/images/learning/selection_error.svg

Learned hypothesis may **fit** the training data very well, even outliers (**noise**) but fail to **generalize** to new examples (test data)



Basics of PCA

- PCA is useful when we need to extract useful information from multivariate data sets.
- This technique is based on the reduced dimensionality.
- Therefore, trends in multivariate data are easily visualized.

Variable Reduction Procedure

- Principal component analysis is a variable reduction procedure. It is useful when you have obtained data on a number of variables (possibly a large number of variables), and believe that there is some redundancy in those variables
- Redundancy means that some of the variables are correlated with one another, possibly because they are measuring the same construct.
- Because of this redundancy, you believe that it should be possible to reduce the observed variables into a smaller number of principal components (artificial variables) that will account for most of the variance in the observed variables.

What is Principal Component

- A **principal component** can be defined as a linear combination of optimally-weighted observed variables.
- Based on how subject scores on a principal component are computed.

7 Item measure of Job Satisfaction

Please respond to each of the following statements by placing a rating in the space to the left of the statement. In making your ratings, use any number from 1 to 7 in which 1="strongly disagree" and 7="strongly agree."

_____ 1. My supervisor treats me with consideration.
_____ 2. My supervisor consults me concerning important decisions that affect my work.
_____ 3. My supervisors give me recognition when I do a good job.
_____ 4. My supervisor gives me the support I need to do my job well.
_____ 5. My pay is fair.
_____ 6. My pay is appropriate, given the amount of responsibility that comes with my job.
_____ 7. My pay is comparable to the pay earned by other employees whose jobs are similar to mine.

Correlations among Seven Job Satisfaction Items

	Correlations						
Variable	1	2	3	4	5	6	7
1	1.00						
2	.75	1.00					
3	.83	.82	1.00				
4	.68	.92	.88	1.00			
5	.03	.01	.04	.01	1.00		
6	.05	.02	.05	.07	.89	1.00	
7	.02	.06	.00	.03	.91	.76	1.00

General Formula

- Below is the general form for the formula to compute scores on the first component extracted (created) in a principal component analysis:

C1 = b 11(X1) + b12(X 2) + ... b1p(Xp)

where

- C1 = the subject's score on principal component 1 (the first component extracted)
- b1p = the regression coefficient (or weight) for observed variable p, as used in
- creating principal component 1
- Xp = the subject's score on observed variable p.

- For example, assume that component 1 in the present study was the "satisfaction with supervision" component. You could determine each subject's score on principal component 1 by using the following fictitious formula:

- C1 = .44 (X1) + .40 (X2) + .47 (X3) + .32 (X4)+ .02 (X5) + .01 (X6) + .03 (X7)

- Obviously, a different equation, with different regression weights, would be used to compute subject scores on component 2 (the satisfaction with pay component). Below is a fictitious illustration of this formula:

- C2 = .01 (X1) + .04 (X2) + .02 (X3) + .02 (X4)+ .48 (X5) + .31 (X6) + .39 (X7)

Number of components Extracted

- If a principal component analysis were performed on data from the 7-item job satisfaction questionnaire, only two components was created. However, such an impression would not be entirely correct.
- In reality, the number of components extracted in a principal component analysis is equal to the number of observed variables being analyzed.
- However, in most analyses, only the first few components account for meaningful amounts of variance, so only these first few components are retained, interpreted, and used in subsequent analyses (such as in multiple regression analyses).

Characteristics of principal components

- The first component extracted in a principal component analysis accounts for a maximal amount of total variance in the observed variables.

- Under typical conditions, this means that the first component will be correlated with at least some of the observed variables. It may be correlated with many.

- The second component extracted will have two important characteristics. First, this component will account for a maximal amount of variance in the data set that was not accounted for by the first component.

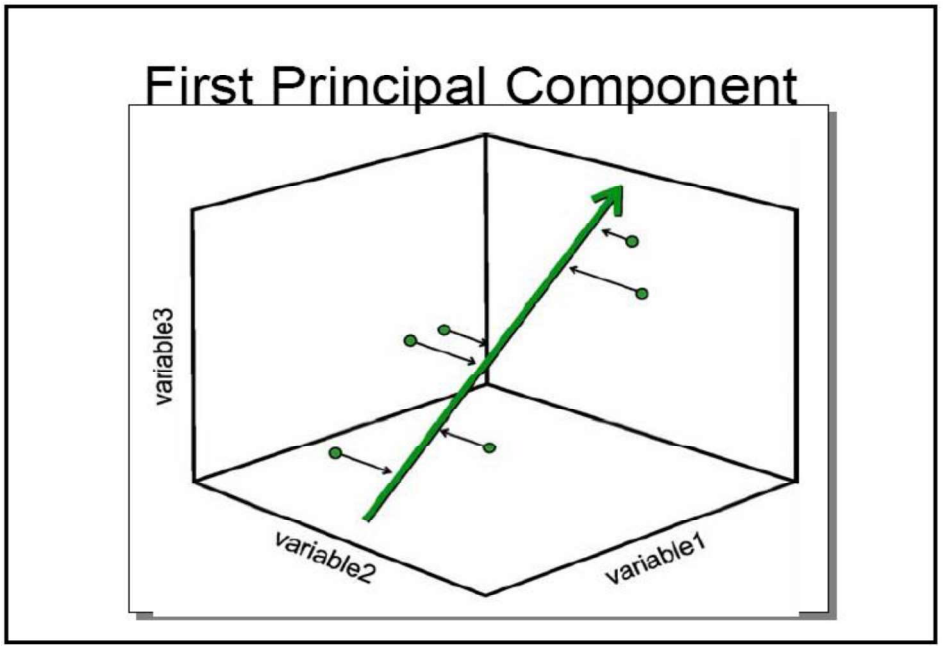
- Under typical conditions, this means that the second component will be correlated with some of the observed variables that did not display strong correlations with component 1.

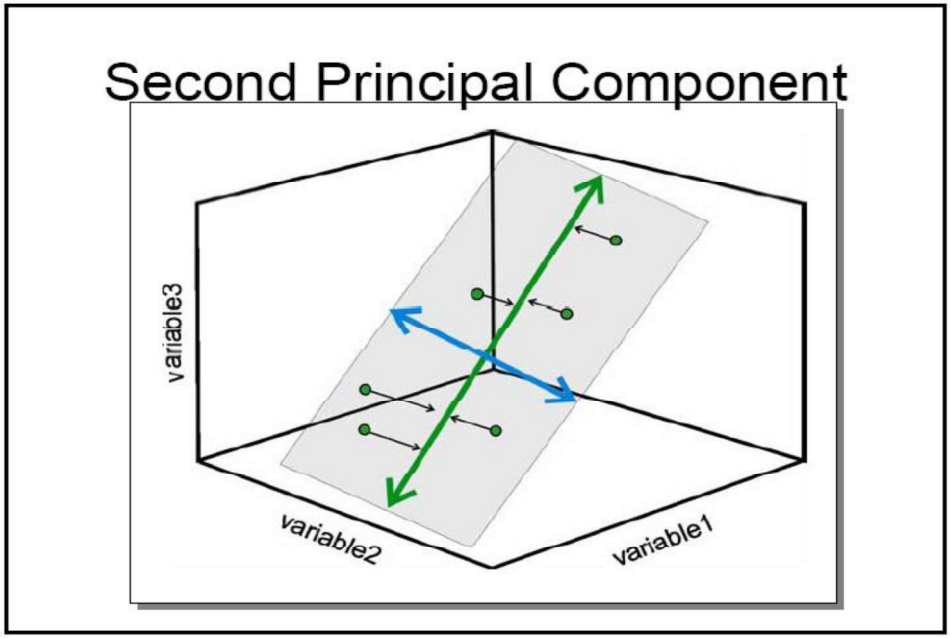
- The second characteristic of the second component is that it will be *uncorrelated* with the first component. Literally, if you were to compute the correlation between components 1 and 2, that correlation would be zero.

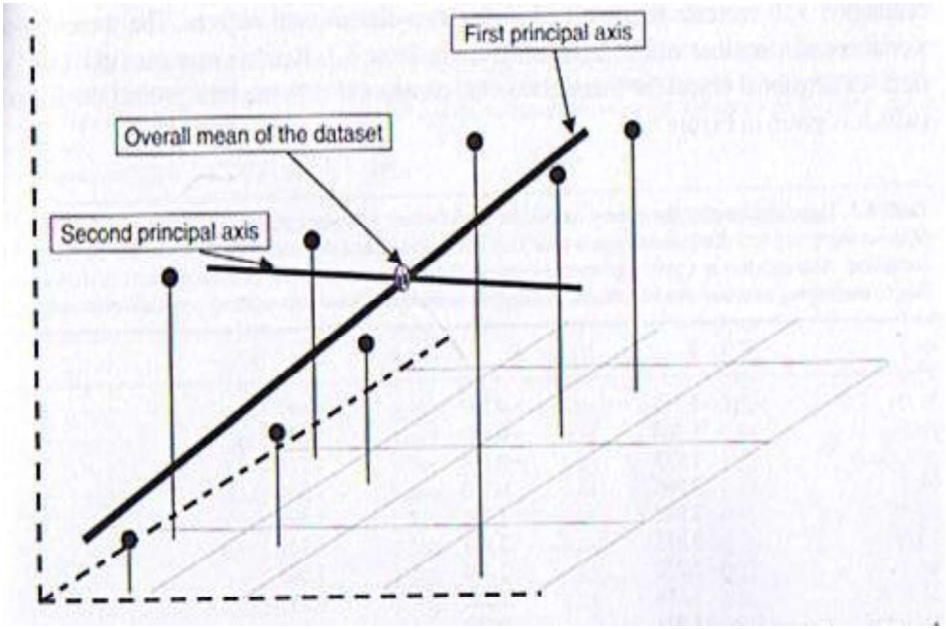
- The remaining components that are extracted in the analysis display the same two characteristics: each component accounts for a maximal amount of variance in the observed variables that was not accounted for by the preceding components, and is uncorrelated with all of the preceding components.

Generalization

- A principal component analysis proceeds in this fashion, with each new component accounting for progressively smaller and smaller amounts of variance (this is why only the first few components are usually retained and interpreted).
- When the analysis is complete, the resulting components will display varying degrees of correlation with the observed variables, but are completely uncorrelated with one another.







Data mining techniques

Outline

- Definition, motivation & application
- Branches of data mining
- Classification, clustering, Association rule mining
- Some classification techniques

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
- Alternative names and their "inside stories":
  - Data mining: a misnomer?
  - Knowledge discovery(mining) in databases (KDD), knowledge extraction, data/pattern analysis, data archeology, business intelligence, etc.

Data Mining Definition

- Finding hidden information in a database
- Fit data to a model
- Similar terms
  - Exploratory data analysis
  - Data driven discovery
  - Deductive learning

Motivation:

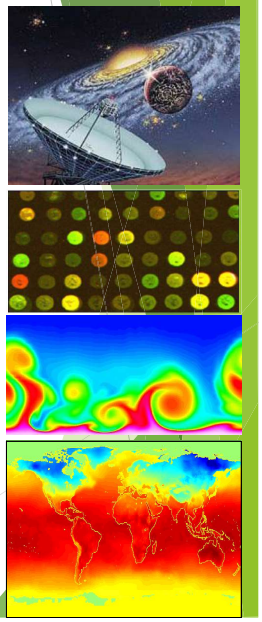
- Data explosion problem
  - Automated data collection tools and mature database technology lead to tremendous amounts of data stored in databases, data warehouses and other information repositories
- We are drowning in data, but starving for knowledge!
- Solution: Data warehousing and data mining
  - Data warehousing and on-line analytical processing
  - Extraction of interesting knowledge (rules, regularities, patterns, constraints) from data in large databases

Why Mine Data? Commercial Viewpoint

- Lots of data is being collected and warehoused
  - Web data, e-commerce
  - purchases at department/ grocery stores
  - Bank/Credit Card transactions
- Computers have become cheaper and more powerful
- Competitive Pressure is Strong
  - Provide better, customized services for an *edge* (e.g. in Customer Relationship Management)

Why Mine Data? Scientific Viewpoint

- Data collected and stored at enormous speeds (GB/hour)
  - remote sensors on a satellite
  - telescopes scanning the skies
  - microarrays generating gene expression data
  - scientific simulations generating terabytes of data
- Traditional techniques infeasible for raw data
- Data mining may help scientists
  - in classifying and segmenting data
  - in Hypothesis Formation



Examples: What is (not) Data Mining?

What is not Data Mining?

– Look up phone number in phone directory

– Query a Web search engine for information about "Amazon"

What is Data Mining?

– Certain names are more prevalent in certain US locations (O'Brien, O'Rurke, O'Reilly… in Boston area)

– Group together similar documents returned by search engine according to their context (e.g. Amazon rainforest, Amazon.com,)

Database Processing vs. Data Mining Processing

- Query
  - Well defined
  - SQL

Data
- Operational data

Output
- Precise
- Subset of database

- Query
  - Poorly defined
  - No precise query language

Data
- Not operational data

Output
- Fuzzy
- Not a subset of database

Query Examples

- Database
  - Find all credit applicants with last name of Smith.
  - Identify customers who have purchased more than $10,000 in the last month.
  - Find all customers who have purchased milk
- Data Mining
  - Find all credit applicants who are poor credit risks. (classification)
  - Identify customers with similar buying habits. (Clustering)
  - Find all items which are frequently purchased with milk. (association rules)

Data Mining: Classification Schemes

- Decisions in data mining
  - Kinds of databases to be mined
  - Kinds of knowledge to be discovered
  - Kinds of techniques utilized
  - Kinds of applications adapted
- Data mining tasks
  - Descriptive data mining
  - Predictive data mining

Decisions in Data Mining

- Databases to be mined
  - Relational, transactional, object-oriented, object-relational, active, spatial, time-series, text, multi-media, heterogeneous, legacy, WWW, etc.
- Knowledge to be mined
  - Characterization, discrimination, association, classification, clustering, trend, deviation and outlier analysis, etc.
  - Multiple/integrated functions and mining at multiple levels
- Techniques utilized
  - Database-oriented, data warehouse (OLAP), machine learning, statistics, visualization, neural network, etc.
- Applications adapted
  - Retail, telecommunication, banking, fraud analysis, DNA mining, stock market analysis, Web mining, Weblog analysis, etc.

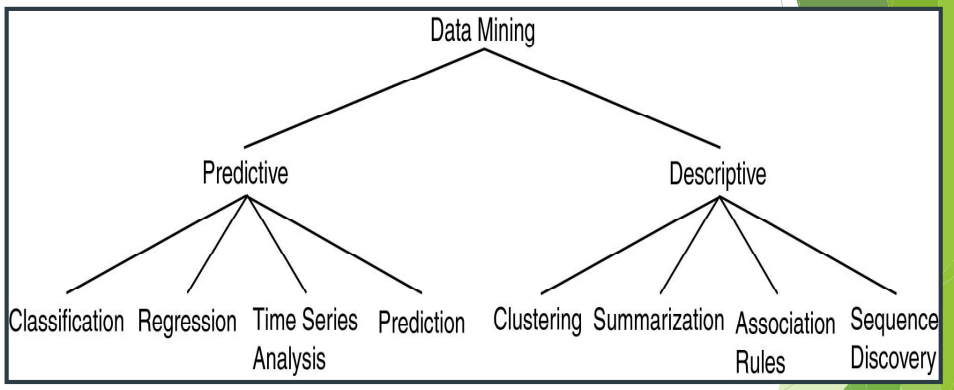
Data Mining Tasks

- Prediction Tasks
  - Use some variables to predict unknown or future values of other variables
- Description Tasks
  - Find human-interpretable patterns that describe the data.

Common data mining tasks

- Classification [Predictive]
- Clustering [Descriptive]
- Association Rule Discovery [Descriptive]
- Sequential Pattern Discovery [Descriptive]
- Regression [Predictive]
- Deviation Detection [Predictive]

Data Mining Models and Tasks



Classification

Classification: Definition

- Given a collection of records (*training set*)
  - Each record contains a set of *attributes*, one of the attributes is the *class*.
- Find a *model* for class attribute as a function of the values of other attributes.
- Goal: previously unseen records should be assigned a class as accurately as possible.
  - A *test set* is used to determine the accuracy of the model. Usually, the given data set is divided into training and test sets, with training set used to build the model and test set used to validate it.

An Example

(from *Pattern Classification by* Duda & Hart & Stork - Second Edition, 2001)

- A fish-packing plant wants to automate the process of sorting incoming fish according to species
- As a pilot project, it is decided to try to separate sea bass from salmon using optical sensing

An Example (continued)

Features (to distinguish):

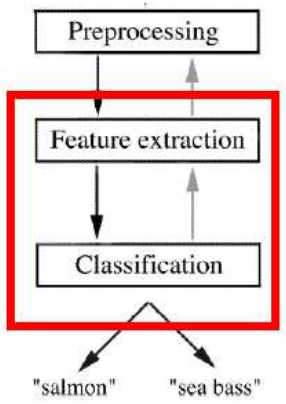
Length

Lightness

Width

Position of mouth

An Example (continued)



- **Preprocessing:** Images of different fishes are isolated from one another and from background;
- **Feature extraction:** The information of a single fish is then sent to a feature extractor, that measure certain "features" or "properties";
- **Classification:** The values of these features are passed to a classifier that evaluates the evidence presented, and build a model to discriminate between the two species

An Example (continued)

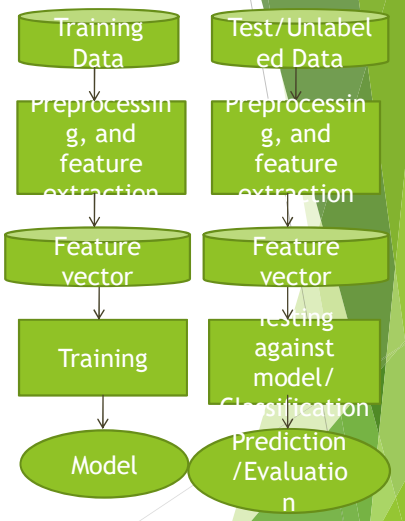
- Domain knowledge:
  - A sea bass is generally longer than a salmon
- Related feature: (or attribute)
  - Length
- Training the classifier:
  - Some examples are provided to the classifier in this form: <fish_length, fish_name>
  - These examples are called training examples
  - The classifier *learns* itself from the training examples, how to distinguish Salmon from Bass based on the *fish_length*

An Example (continued)

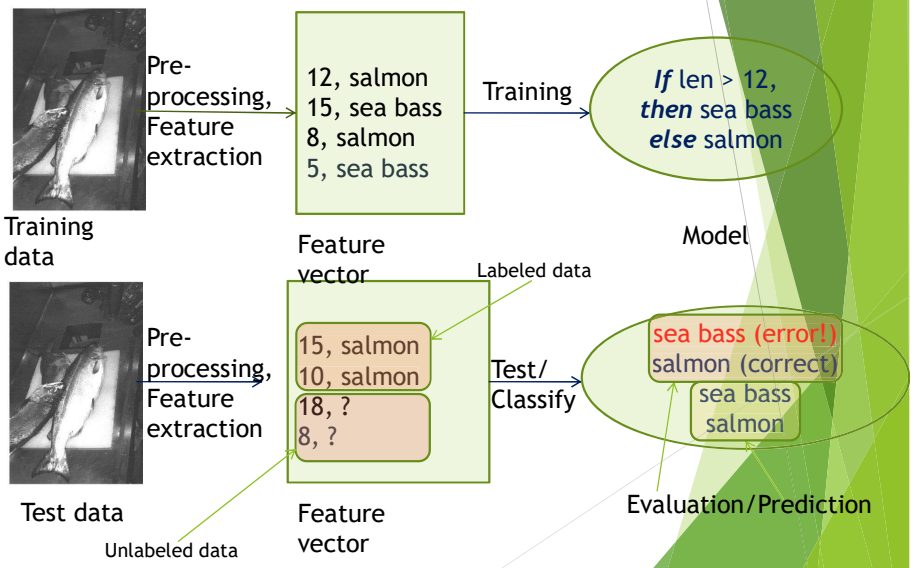
- Classification model (hypothesis):
  - The classifier generates a model from the training data to classify future examples (test examples)
  - An example of the model is a rule like this:
  - If *Length* >= *l* then sea bass* otherwise *salmon*
  - Here the value of *l** determined by the classifier
- Testing the model
  - Once we get a model out of the classifier, we may use the classifier to test future examples
  - The test data is provided in the form <fish_length>
  - The classifier outputs <fish_type> by checking *fish_length* against the model

An Example (continued)

- So the overall classification process goes like this →



An Example (continued)

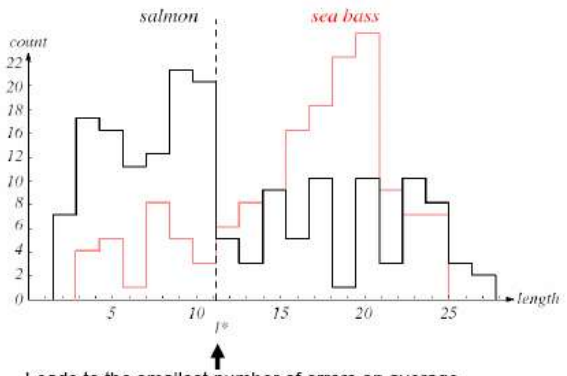


An Example (continued)

- Why error?
  - Insufficient training data
  - Too few features
  - Too many/irrelevant features
  - Overfitting / specialization

An Example (continued)

Histograms of the length feature for the two categories



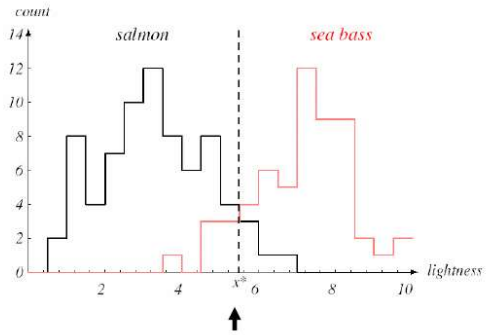
Leads to the smallest number of errors on average

We cannot reliably separate sea bass from salmon by length alone!

An Example (continued)

- New Feature:
  - *Average lightness of the fish scales*

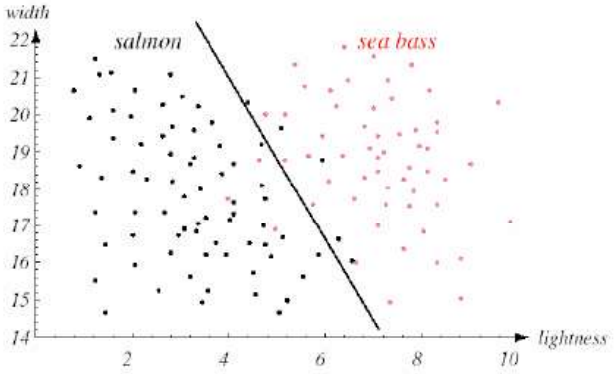
Histograms of the lightness feature for the two categories



Leads to the smallest number of errors on average

The two classes are much better separated!

An Example (continued)



Decision rule: Classify the fish as a sea bass if its feature vector falls above the decision boundary shown, and as salmon otherwise

An Example (continued)

Training data

Pre-processing, Feature extraction

12, 4, salmon
15, 8, sea bass
8, 2, salmon
5, 10, sea bass

Feature vector

Training

If ltns > 6 **or** len*5+ltns*2>100 *then* sea bass *else* salmon

Model

Test data

Pre-processing, Feature extraction

15, 2, salmon
10, 7, salmon
18, 7, ?
8, 5, ?

Feature vector

Test/ Classify

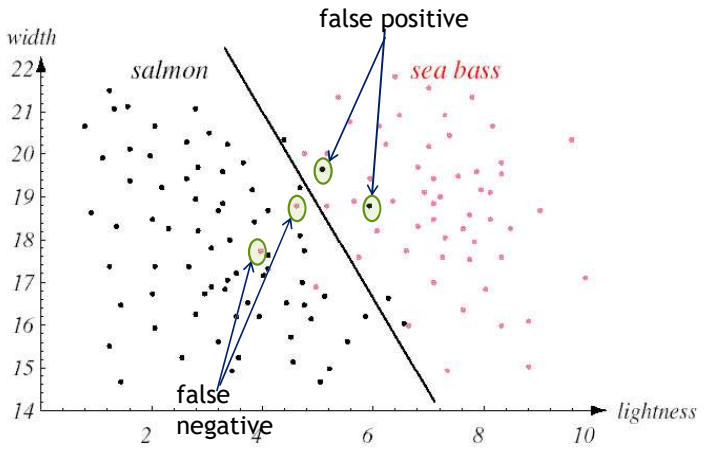
salmon (correct)
salmon (correct)
sea bass
salmon

Evaluation/Prediction

Terms

- Accuracy:
  - % of test data correctly classified
  - In our first example, accuracy was 3 out 4 = 75%
  - In our second example, accuracy was 4 out 4 = 100%
- False positive:
  - Negative class incorrectly classified as positive
  - Usually, the larger class is the negative class
  - Suppose
    - **salmon is negative** class
    - **sea bass is positive class**

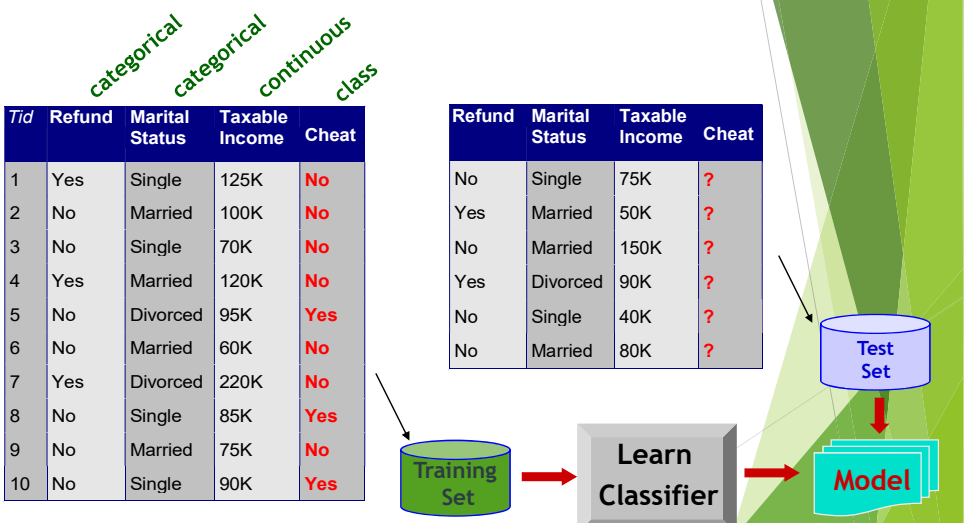
Terms



Terms

- Cross validation (3 fold)

Classification Example 2



Classification: Application 1

- Direct Marketing
  - Goal: Reduce cost of mailing by *targeting* a set of consumers likely to buy a new cell-phone product.
  - Approach:
    - Use the data for a similar product introduced before.
    - We know which customers decided to buy and which decided otherwise. This *{buy, don't buy}* decision forms the *class attribute*.
    - Collect various demographic, lifestyle, and company-interaction related information about all such customers.
      - Type of business, where they stay, how much they earn, etc.
    - Use this information as input attributes to learn a classifier model.

Classification: Application 2

- Fraud Detection
  - Goal: Predict fraudulent cases in credit card transactions.
  - Approach:
    - Use credit card transactions and the information on its account-holder as attributes.
      - When does a customer buy, what does he buy, how often he pays on time, etc
    - Label past transactions as fraud or fair transactions. This forms the class attribute.
    - Learn a model for the class of the transactions.
    - Use this model to detect fraud by observing credit card transactions on an account.

Classification: Application 3

- Customer Attrition/Churn:
  - Goal: To predict whether a customer is likely to be lost to a competitor.
  - Approach:
    - Use detailed record of transactions with each of the past and present customers, to find attributes.
      - How often the customer calls, where he calls, what time-of-the day he calls most, his financial status, marital status, etc.
    - Label the customers as loyal or disloyal.
    - Find a model for loyalty.

Classification: Application 4

- Sky Survey Cataloging
  - Goal: To predict class (star or galaxy) of sky objects, especially visually faint ones, based on the telescopic survey images (from Palomar Observatory).
    - 3000 images with 23,040 x 23,040 pixels per image.
  - Approach:
    - Segment the image.
    - Measure image attributes (features) - 40 of them per object.
    - Model the class based on these features.
    - Success Story: Could find 16 new high red-shift quasars, some of the farthest objects that are difficult to find!

Clustering

Clustering Definition

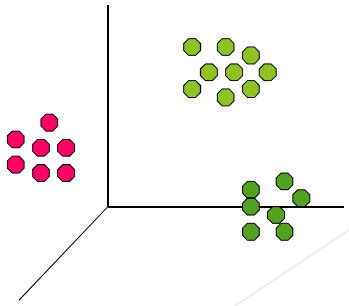
- Given a set of data points, each having a set of attributes, and a similarity measure among them, find clusters such that
  - Data points in one cluster are more similar to one another.
  - Data points in separate clusters are less similar to one another.
- Similarity Measures:
  - Euclidean Distance if attributes are continuous.
  - Other Problem-specific Measures.

Illustrating Clustering

Euclidean Distance Based Clustering in 3-D space.

Intracluster distances are minimized

Intercluster distances are maximized



Clustering: Application 1

- Market Segmentation:
  - Goal: subdivide a market into distinct subsets of customers where any subset may conceivably be selected as a market target to be reached with a distinct marketing mix.
  - Approach:
    - Collect different attributes of customers based on their geographical and lifestyle related information.
    - Find clusters of similar customers.
    - Measure the clustering quality by observing buying patterns of customers in same cluster vs. those from different clusters.

Clustering: Application 2

- Document Clustering:
  - Goal: To find groups of documents that are similar to each other based on the important terms appearing in them.
  - Approach: To identify frequently occurring terms in each document. Form a similarity measure based on the frequencies of different terms. Use it to cluster.
  - Gain: Information Retrieval can utilize the clusters to relate a new document or search term to clustered documents.

Association rule mining

Association Rule Discovery: Definition

- Given a set of records each of which contain some number of items from a given collection;
  - Produce dependency rules which will predict occurrence of an item based on occurrences of other items.

TID	Items
1	Bread, Coke, Milk
2	Beer, Bread
3	Beer, Coke, Diaper, Milk
4	Beer, Bread, Diaper, Milk
5	Coke, Diaper, Milk

Rules Discovered:
{Milk} --> {Coke}
{Diaper, Milk} --> {Beer}

Association Rule Discovery: Application 1

- Marketing and Sales Promotion:
  - Let the rule discovered be
    {Bagels, … } --> {Potato Chips}
  - Potato Chips as consequent => Can be used to determine what should be done to boost its sales.
  - Bagels in the antecedent => Can be used to see which products would be affected if the store discontinues selling bagels.
  - Bagels in antecedent *and* Potato chips in consequent => Can be used to see what products should be sold with Bagels to promote sale of Potato chips!

Association Rule Discovery: Application 2

- Supermarket shelf management.
  - Goal: To identify items that are bought together by sufficiently many customers.
  - Approach: Process the point-of-sale data collected with barcode scanners to find dependencies among items.
  - A classic rule --
    - If a customer buys diaper and milk, then he is very likely to buy beer:

$$Diapers \rightarrow Beer,\ support = 20\%,\ confidence = 85\%$$

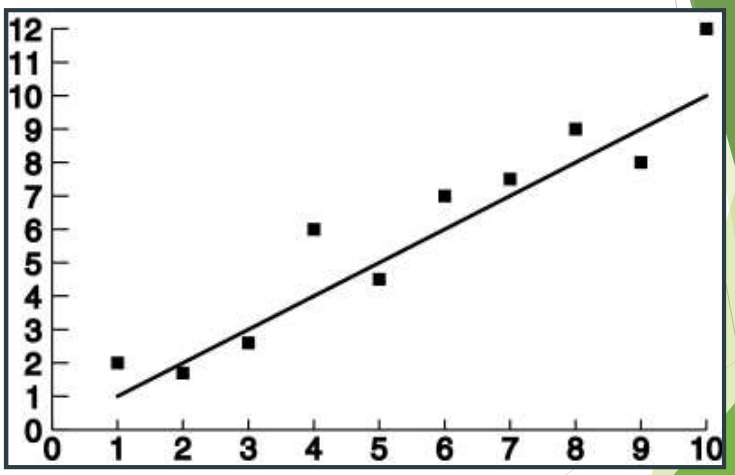
Regression

- Predict future values based on past values
- ***Linear Regression*** assumes linear relationship exists.

$$y = c_0 + c_1 x_1 + \ldots + c_n x_n$$

- Find values to best fit the data

Linear Regression



Correlation

- Examine the degree to which the values for two variables behave similarly.
- Correlation coefficient r:
  - 1 = perfect correlation
  - -1 = perfect but opposite correlation
  - 0 = no correlation

$$r = \frac{\sum(x_i - \bar{X})(y_i - \bar{Y})}{\sqrt{\sum(x_i - \bar{X})^2 \sum(y_i - \bar{Y})^2}}$$

Ensemble Learning

Introduction

When you want to purchase a new car, will you walk up to the first car shop and purchase one based on the advice of the dealer? It's highly unlikely.

You would likely browser a few web portals where people have posted their reviews and compare different car models, checking for their features and prices. You will also probably ask your friends and colleagues for their opinion. In short, you wouldn't directly reach a conclusion, but will instead make a decision considering the opinions of other people as well.

Ensemble Techniques in Machine Learning

Ensemble models in machine learning operate on a similar idea. They combine the decisions from multiple models to improve the overall performance.

What is Ensemble Learning with example?

Ensemble learning is a machine learning technique that enhances accuracy and resilience in forecasting by merging predictions from multiple models. It aims to mitigate errors or biases that may exist in individual models by leveraging the collective intelligence of the ensemble.

The underlying concept behind ensemble learning is to combine the outputs of diverse models to create a more precise prediction. By considering multiple perspectives and utilizing the strengths of different models, ensemble learning improves the overall performance of the learning system. This approach not only enhances accuracy but also provides resilience against uncertainties in the data. By effectively merging predictions from multiple models, ensemble learning has proven to be a powerful tool in various domains, offering more robust and reliable forecasts.

Let's understand the concept of ensemble learning with an example. Suppose you are a movie director and you have created a short movie on a very important and interesting topic. Now, you want to take preliminary feedback (ratings) on the movie before making it public. What are the possible ways by which you can do that?

A: *You may ask one of your friends to rate the movie for you.* Now it's entirely possible that the person you have chosen loves you very much and doesn't want to break your heart by providing a 1-star rating to the horrible work you have created.

B: *Another way could be by asking 5 colleagues of yours to rate the movie.* This should provide a better idea of the movie. This method may provide honest ratings for your movie. But a problem still exists. These 5 people may not be "Subject Matter Experts" on the topic of your movie. Sure, they might understand the cinematography, the shots, or the audio, but at the same time may not be the best judges of dark humour.

C: *How about asking 50 people to rate the movie?*
Some of which can be your friends, some of them can be your colleagues and some may even be total strangers.

The responses, in this case, would be more generalized and diversified since now you have people with different sets of skills. And as it turns out – this is a better approach to get honest ratings than the previous cases we saw.

With these examples, you can infer that a diverse group of people are likely to make better decisions as compared to individuals. Similar is true for a diverse set of models in comparison to single models. This diversification in Machine Learning is achieved by a technique called Ensemble Learning.

Now that you have got a gist of what ensemble learning is – let us look at the various techniques in ensemble learning along with their implementations.

Ensemble Techniques

In this section, we will look at a few techniques, namely:

1. **Max Voting**

The max voting method is generally used for classification problems. In this technique, multiple models are used to make predictions for each data point. The predictions by each model are considered as a 'vote'. The predictions which we get from the majority of the models are used as the final prediction.

For example, when you asked 5 of your colleagues to rate your movie (out of 5); we'll assume three of them rated it as 4 while two of them gave it a 5. Since the majority gave a rating of 4, the final rating will be taken as 4. **You can consider this as taking the mode of all the predictions.**

The result of max voting would be something like this:

Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5
5	4	5	4	4

2. **Averaging**

Similar to the max voting technique, multiple predictions are made for each data point in averaging. In this method, we take an average of predictions from all the models and use it to make the final prediction. Averaging can be used for making predictions in regression problems or while calculating probabilities for classification problems.

For example, in the below case, the averaging method would take the average of all the values.

i.e. (5+4+5+4+4)/5 = 4.4

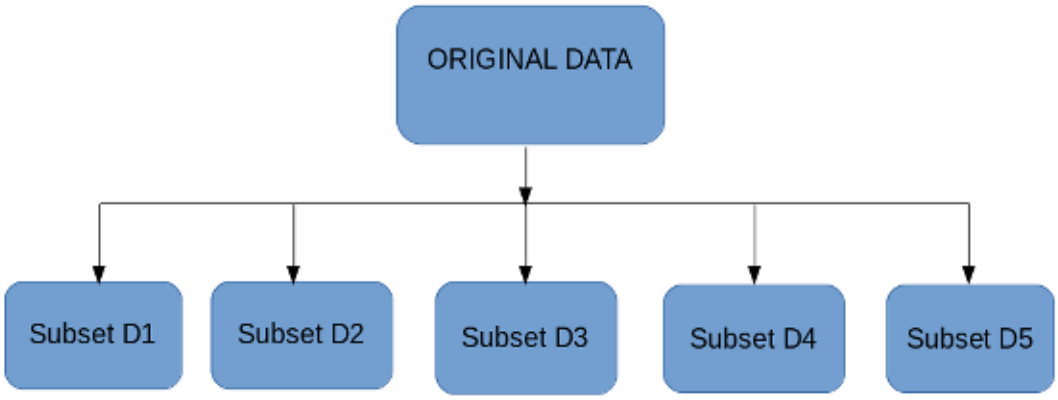
Colleague 1	Colleague 2	Colleague 3	Colleague 4	Colleague 5
5	4	5	4	4

3. Bagging

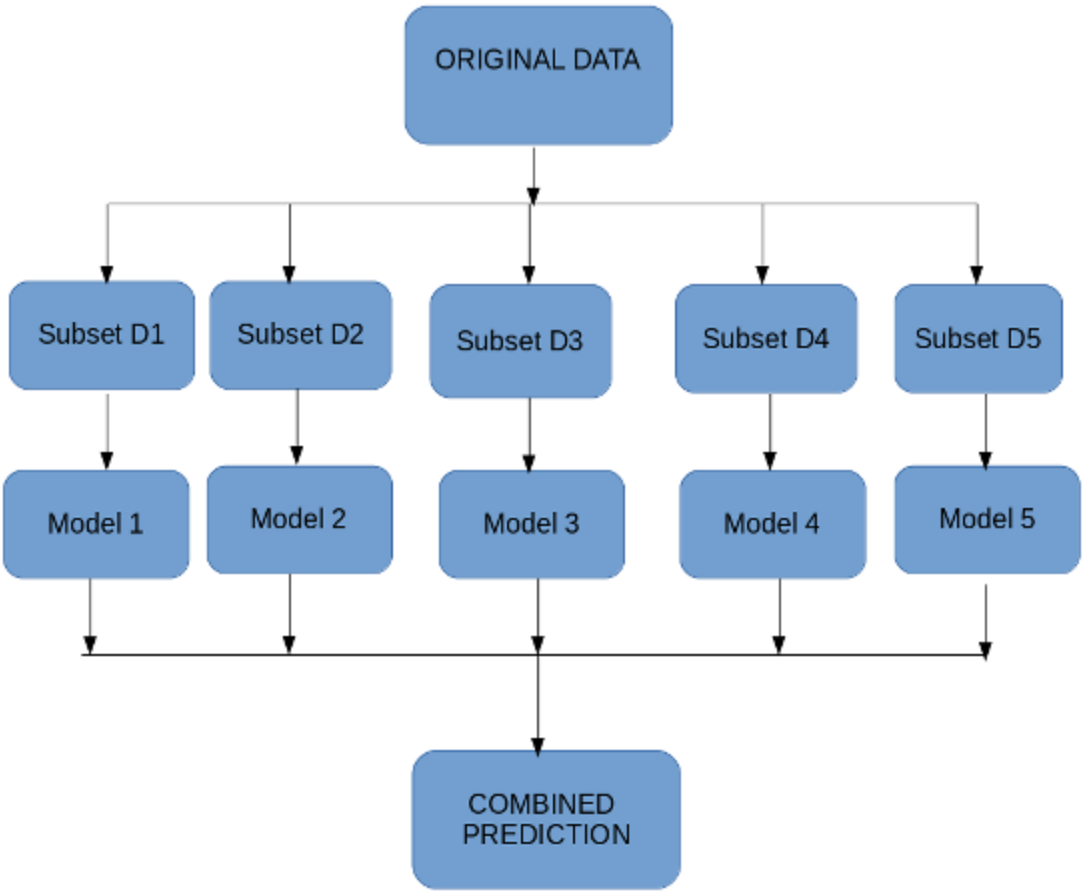
The idea behind bagging is combining the results of multiple models (for instance, all decision trees) to get a generalized result. Here's a question: If you create all the models on the same set of data and combine it, will it be useful? There is a high chance that these models will give the same result since they are getting the same input. So how can we solve this problem? One of the techniques is bootstrapping.

Bootstrapping is a sampling technique in which we create subsets of observations from the original dataset, **with replacement**. The size of the subsets is the same as the size of the original set.

Bagging (or Bootstrap Aggregating) technique uses these subsets (bags) to get a fair idea of the distribution (complete set). The size of subsets created for bagging may be less than the original set.



1. Multiple subsets are created from the original dataset, selecting observations with replacement.
2. A base model (weak model) is created on each of these subsets.
3. The models run in parallel and are independent of each other.



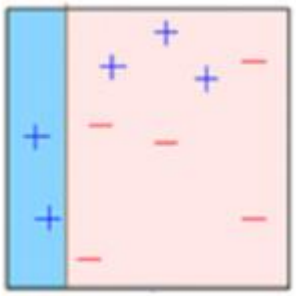
4. Boosting

Before we go further, here's another question for you: If a data point is incorrectly predicted by the first model, and then the next (probably all models), will combining the predictions provide better results? Such situations are taken care of by boosting.

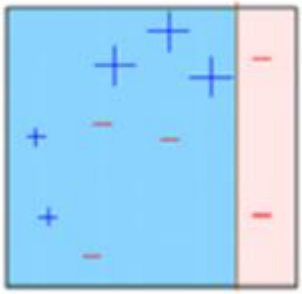
Boosting is a sequential process, where each subsequent model attempts to correct the errors of the previous model. The succeeding models are dependent on the previous model. Let's understand the way boosting works in the below steps.

1. A subset is created from the original dataset.
2. Initially, all data points are given equal weights.
3. A base model is created on this subset.

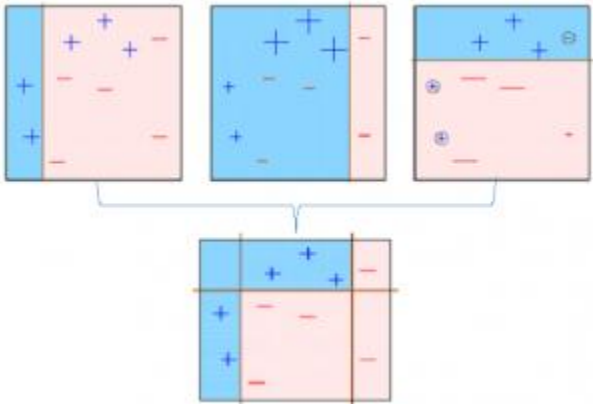
4. This model is used to make predictions on the whole dataset.



5. Errors are calculated using the actual values and predicted values.
6. The observations which are incorrectly predicted, are given higher weights. (Here, the three misclassified blue-plus points will be given higher weights)
7. Another model is created and predictions are made on the dataset. (This model tries to correct the errors from the previous model)



8. Similarly, multiple models are created, each correcting the errors of the previous model.
9. The final model (strong learner) is the weighted mean of all the models (weak learners).



Thus, the boosting algorithm combines a number of weak learners to form a strong learner. The individual models would not perform well on the entire dataset, but they work well for some part of the dataset. Thus, each model actually boosts the performance of the ensemble.

Decision Tree

Decision trees are a popular machine learning algorithm that can be used for both regression and classification tasks. A decision tree is **a non-parametric supervised learning algorithm for classification and regression tasks**. It has a hierarchical tree structure consisting of a root node, branches, internal nodes, and leaf nodes. Decision trees are used for classification and regression tasks, providing easy-to-understand models.

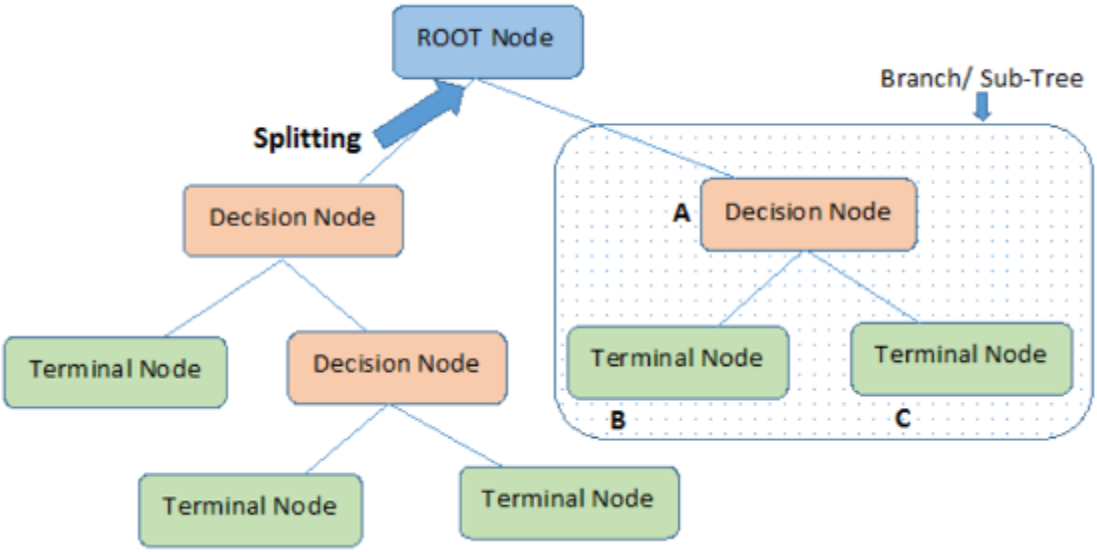
A decision tree is a hierarchical model used in decision support that depicts decisions and their potential outcomes, incorporating chance events, resource expenses, and utility. This algorithmic model utilizes conditional control statements and is non-parametric, supervised learning, useful for both classification and regression tasks. The tree structure is comprised of a root node, branches, internal nodes, and leaf nodes, forming a hierarchical, tree-like structure.

It is a tool that has applications spanning several different areas. Decision trees can be used for classification as well as regression problems. The name itself suggests that it uses a flowchart like a tree structure to show the predictions that result from a series of feature-based splits. It starts with a root node and ends with a decision made by leaves.

Decision Tree Terminologies

Before learning more about decision trees let's get familiar with some of the terminologies:

- ***Root Node****:* The initial node at the beginning of a decision tree, where the entire population or dataset starts dividing based on various features or conditions.
- ***Decision Nodes****:* Nodes resulting from the splitting of root nodes are known as decision nodes. These nodes represent intermediate decisions or conditions within the tree.
- ***Leaf Nodes****:* Nodes where further splitting is not possible, often indicating the final classification or outcome. Leaf nodes are also referred to as terminal nodes.
- ***Sub-Tree****:* Similar to a subsection of a graph being called a sub-graph, a sub-section of a decision tree is referred to as a sub-tree. It represents a specific portion of the decision tree.
- ***Pruning***: The process of removing or cutting down specific nodes in a decision tree to prevent overfitting and simplify the model.
- ***Branch / Sub-Tree****:* A subsection of the entire decision tree is referred to as a branch or sub-tree. It represents a specific path of decisions and outcomes within the tree.
- ***Parent and Child Node****:* In a decision tree, a node that is divided into sub-nodes is known as a parent node, and the sub-nodes emerging from it are referred to as child nodes. The parent node represents a decision or condition, while the child nodes represent the potential outcomes or further decisions based on that condition.


ROOT Node
Branch/ Sub-Tree
Splitting
Decision Node
A
Decision Node
Terminal Node
Decision Node
Terminal Node
Terminal Node
B
C
Terminal Node
Terminal Node

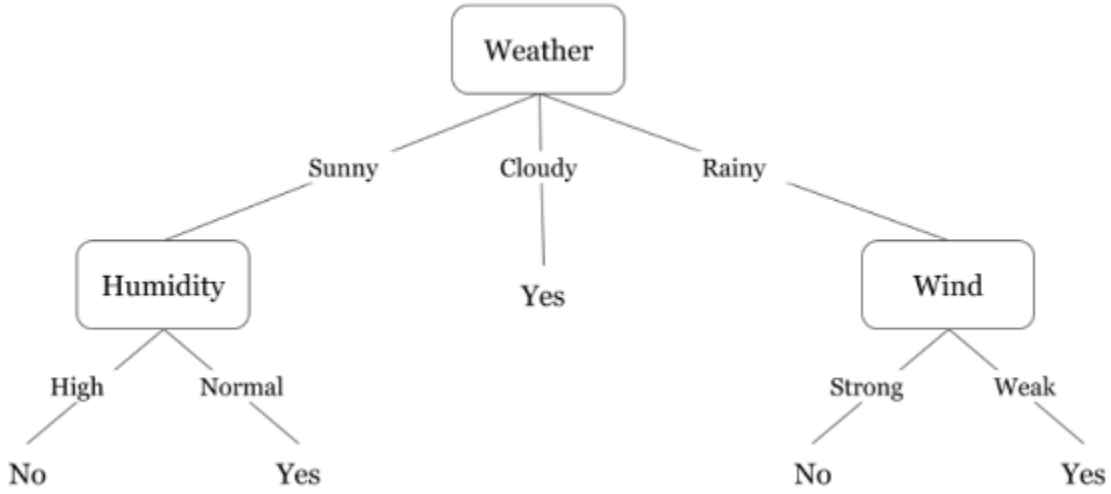
Example of Decision Tree

Let’s understand decision trees with the help of an example:

Day	Weather	Temperature	Humidity	Wind	Play?
1	Sunny	Hot	High	Weak	No
2	Cloudy	Hot	High	Weak	Yes
3	Sunny	Mild	Normal	Strong	Yes
4	Cloudy	Mild	High	Strong	Yes
5	Rainy	Mild	High	Strong	No
6	Rainy	Cool	Normal	Strong	No
7	Rainy	Mild	High	Weak	Yes
8	Sunny	Hot	High	Strong	No
9	Cloudy	Hot	Normal	Weak	Yes
10	Rainy	Mild	High	Strong	No

Decision trees are upside down which means the root is at the top and then this root is split into various several nodes. Decision trees are nothing but a bunch of if-else statements in layman terms. It checks if the condition is true and if it is then it goes to the next node attached to that decision.

In the below diagram the tree will first ask what is the weather? Is it sunny, cloudy, or rainy? If yes then it will go to the next feature which is humidity and wind. It will again check if there is a strong wind or weak, if it’s a weak wind and it’s rainy then the person may go and play.



Did you notice anything in the above flowchart? We see that if the *weather is cloudy* then we must go to play. Why didn't it split more? Why did it stop there?

To answer this question, we need to know about few more concepts like entropy, information gain, and Gini index. But in simple terms, I can say here that the output for the training dataset is always yes for cloudy weather, since there is no disorderliness here we don't need to split the node further.

The goal of machine learning is to decrease uncertainty or disorders from the dataset and for this, we use decision trees.

Now you must be thinking how do I know what should be the root node? what should be the decision node? when should I stop splitting? To decide this, there is a metric called "Entropy" which is the amount of uncertainty in the dataset.

How decision tree algorithms work?

Decision Tree algorithm works in simpler steps

- **Starting at the Root**: The algorithm begins at the top, called the "root node," representing the entire dataset.
- **Asking the Best Questions:** It looks for the most important feature or question that splits the data into the most distinct groups. This is like asking a question at a fork in the tree.
- **Branching Out**: Based on the answer to that question, it divides the data into smaller subsets, creating new branches. Each branch represents a possible route through the tree.
- **Repeating the Process**: The algorithm continues asking questions and splitting the data at each branch until it reaches the final "leaf nodes," representing the predicted outcomes or classifications.

Decision Tree Assumptions

Several assumptions are made to build effective models when creating decision trees. These assumptions help guide the tree's construction and impact its performance. Here are some common assumptions and considerations when creating decision trees:

Binary Splits

Decision trees typically make binary splits, meaning each node divides the data into two subsets based on a single feature or condition. This assumes that each decision can be represented as a binary choice.

Recursive Partitioning

Decision trees use a recursive partitioning process, where each node is divided into child nodes, and this process continues until a stopping criterion is met. This assumes that data can be effectively subdivided into smaller, more manageable subsets.

Feature Independence

Decision trees often assume that the features used for splitting nodes are independent. In practice, feature independence may not hold, but decision trees can still perform well if features are correlated.

Homogeneity

Decision trees aim to create homogeneous subgroups in each node, meaning that the samples within a node are as similar as possible regarding the target variable. This assumption helps in achieving clear decision boundaries.

Top-Down Greedy Approach

Decision trees are constructed using a top-down, greedy approach, where each split is chosen to maximize information gain or minimize impurity at the current node. This may not always result in the globally optimal tree.

Categorical and Numerical Features

Decision trees can handle both categorical and numerical features. However, they may require different splitting strategies for each type.

Overfitting

Decision trees are prone to overfitting when they capture noise in the data. Pruning and setting appropriate stopping criteria are used to address this assumption.

Impurity Measures

Decision trees use impurity measures such as Gini impurity or entropy to evaluate how well a split separates classes. The choice of impurity measure can impact tree construction.

No Missing Values

Decision trees assume that there are no missing values in the dataset or that missing values have been appropriately handled through imputation or other methods.

Equal Importance of Features

Decision trees may assume equal importance for all features unless feature scaling or weighting is applied to emphasize certain features.

No Outliers

Decision trees are sensitive to outliers, and extreme values can influence their construction. Preprocessing or robust methods may be needed to handle outliers effectively.

Sensitivity to Sample Size

Small datasets may lead to overfitting, and large datasets may result in overly complex trees. The sample size and tree depth should be balanced.