ARTIFICIAL INTELLIGENCE

The Very Idea

Vasant G. Honavar
Dorothy Foehr Huck and J. Lloyd Huck Chair in Biomedical Data Sciences and Artificial Intelligence
Professor of Data Sciences, Informatics, Computer Science, Bioinformatics & Genomics and Neuroscience
Director, Artificial Intelligence Research Laboratory
Director, Center for Artificial Intelligence Foundations and Scientific Applications
Associate Director, Institute for Computational and Data Sciences
Pennsylvania State University
Co-PI, Northeast Big Data Hub
Informatics Lead, Penn State Clinical and Translational Sciences Institute

vhonavar@psu.edu
http://faculty.ist.psu.edu/vhonavar
http://ailab.ist.psu.edu

Why should machines learn? – Science of learning

Machine learning offers algorithmic models of learning that can provide useful insights into

- How humans and animals learn
- Information requirements of learning tasks
- The precise conditions under which learning is possible
- Inherent difficulty of learning tasks
- How to improve learning – e.g. value of active versus passive learning
- Computational architectures for learning

Why do we need machines to learn? – Practical

- Intelligent behavior requires knowledge
- Explicitly specifying the knowledge needed for specific tasks is hard, and often infeasible
  - Lesson from expert systems
- If we can program computers to learn from experience, we can
  - Dramatically enhance the usability of software
  - Dramatically reduce the cost of software development
  - Automate aspects of scientific discovery

Machine Learning is most useful when

- the structure of the task is not well understood but a representative data or interactions with the environment is available
- task (or its parameters) change dynamically

Why should machines learn?

Machines that learn have many applications

- Diagnosing diseases from symptoms
- Detecting SPAM
- Determining credit-worthiness
- Recommending products, movies, web pages..
- Targeting advertisements
- Predicting stock prices
- Detecting malware
- Driving cars
- Predicting molecular function from sequence
- Predicting health risks
- Detecting fraud
- Precision farming
- Translating between languages
- Conversational agents, e.g., ChatGPT

What is Machine Learning?

- A machine *M* is said to learn from experience *E* with respect to some class of tasks *T* and performance measure *P* if its performance as measured by *P* on tasks in *T* in the environment *Z* improves with experience *E*.

Example 1

T – cancer diagnosis

E – a set of diagnosed cases

P – accuracy of diagnosis on new cases

Z – noisy measurements, occasionally misdiagnosed training cases

M – a program that runs on a general purpose computer

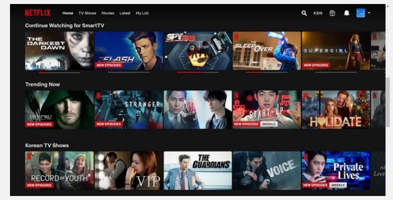
What is Machine Learning?

Example 2

T – personalized movie recommendation, e.g., on Netflix

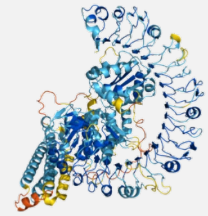
E – movie ratings data from individuals

P – accuracy of predicted movie ratings



10% improvement in prediction accuracy – $1 million prize

What is Machine Learning?

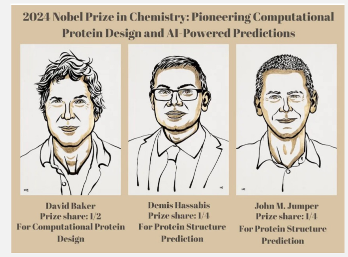


Example 3

T – Predicting protein structures from amino acid sequence

E – A data set experimentally determined sequence structure pairs

P – accuracy of predicted structures



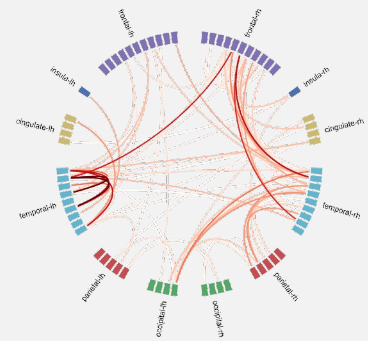
What is Machine Learning?

Example 4

T – Reconstructing functional connectivity of brains from brain activity (e.g., fMRI) data

E – fMRI data

P – accuracy of the reconstructed network



What is Machine Learning?

Example 5

T – solving integral calculus problems, given rules of integral calculus

E – a set of solved problems

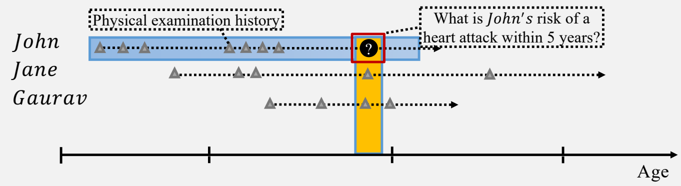
P – score on test consisting of problems not in *E*

$$\int x\sqrt{x}\,dx \qquad \int \sqrt{ax+b}\,dx$$

$$\int \frac{dx}{\sqrt[5]{x}} \qquad \int \frac{3x^2-2\sqrt{x}}{x}\,dx$$

$$\int \frac{(x^2+2x+1)\,dx}{(x^3+6x^2+6x+7)^{\frac{1}{3}}}$$

What is Machine Learning?



Example 6

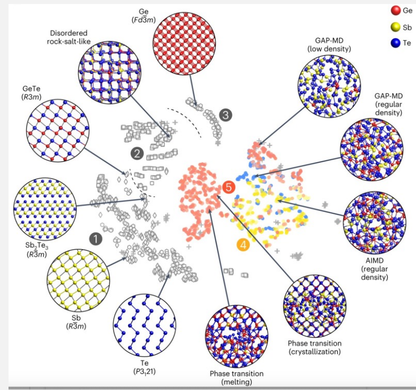
T – predicting the risk of diabetes before the onset of clinical symptoms

E – longitudinal electronic health records data

P – accuracy of predictions

What is Machine Learning?

- Example 7
- *T* – Generating novel material structures with desired properties
- *E* – Databases of materials – composition,
  structure, properties, literature
- *P* – success in generating novel materials with the desired properties



PennState Institute for Computational and Data Sciences

Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory

PennState Clinical and Translational Science Institute

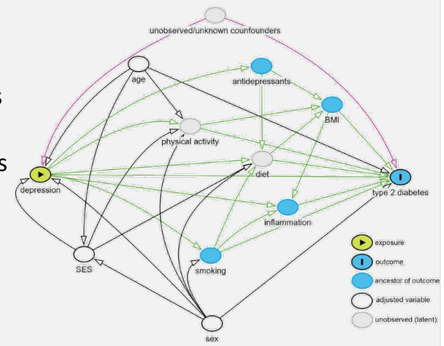
What is Machine Learning?

Example 8

T – Uncovering the causal relationship between exercise, diet and diabetes

E – Data from observations and interventions (changes in diet, exercise)

P – accuracy of causal predictions



PennState College of Information Sciences And Technology

AI 100 Fall 2024

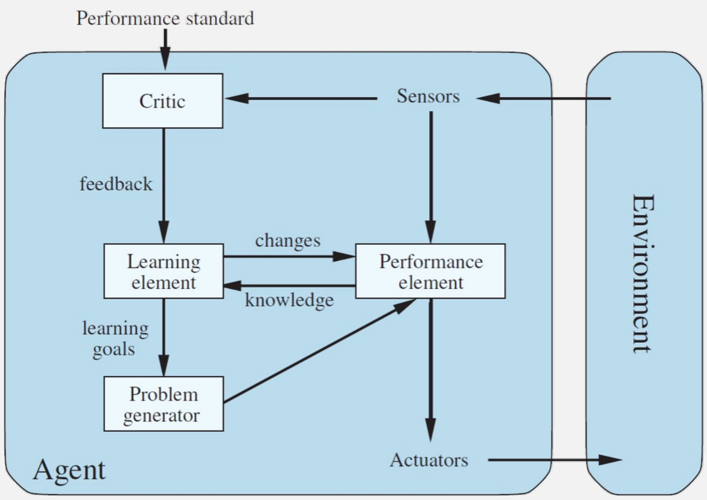
Vasant G Honavar

What is Machine Learning?

Example 9

- *T* – driving a car
- *E* – Observations of driver actions under a broad range of conditions
- *P* – suitable measure of good driving – safety, efficiency, ...

Agents that learn



Canonical Types of Learning

Supervised Learning: given examples of inputs and corresponding desired outputs, predict outputs on future inputs

- Classification
- Regression
- Time series prediction
- Multi-task learning
- Domain adaptation

Unsupervised Learning: given only inputs, automatically discover representations, features, structure, etc.

- Clustering
- Outlier detection
- Generative modeling

Reinforcement Learning

…

Key requirements for learning

- There are patterns to be learned
- There are data to learn from

Applicant information:

age	23 years
gender	male
annual salary	\$30,000
years in residence	1 year
years in job	1 year
current debt	\$15,000
···	···

Approve credit?

Learning to approve credit

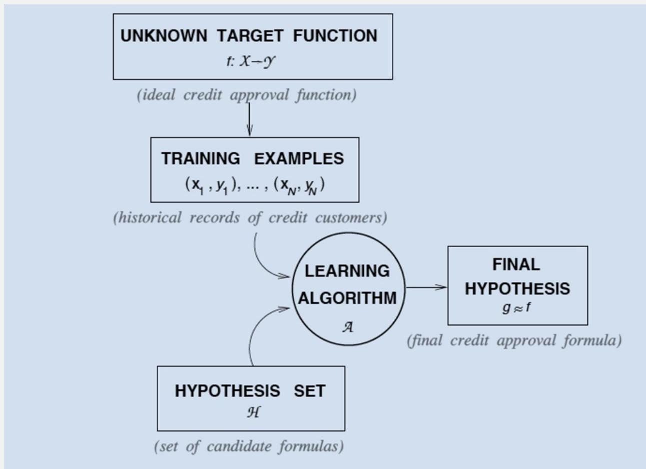
Formalization:

- Input: $\mathbf{x}$ (*customer application*)
- Output: y (*good/bad customer?*)
- Target function: $f : \mathcal{X} \to \mathcal{Y}$ (*ideal credit approval formula*)
- Data: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \cdots, (\mathbf{x}_N, y_N)$ (*historical records*)

$\downarrow \quad \downarrow \quad \downarrow$

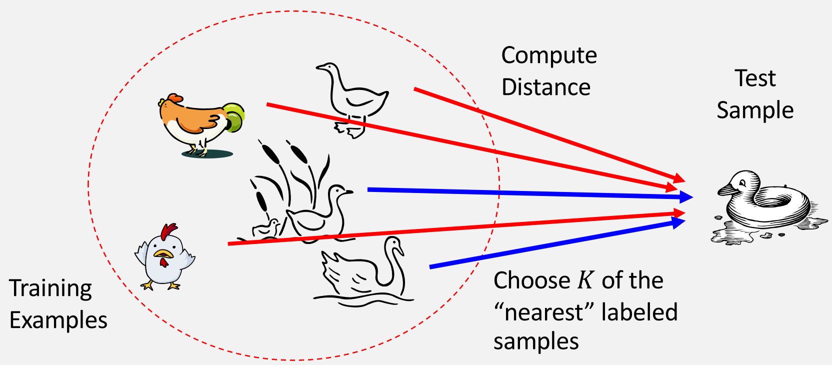
- Hypothesis: $g : \mathcal{X} \to \mathcal{Y}$ (*formula to be used*)

Learning to approve to credit

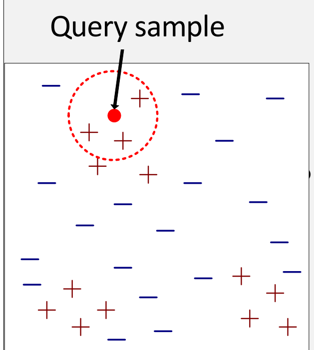


Learning to classify using nearest neighbors

- Basic idea:
  - If it walks like a duck, quacks like a duck, then it's probably a duck



Nearest-Neighbor Classifiers



Require

- The set of stored labeled training examples
- Distance measure to compute distance between samples
- The value of K, an odd number of nearest neighbors to retrieve

To classify a query sample:

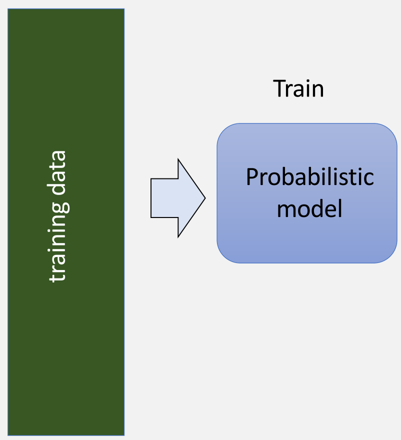
- Compute distance to training samples
- Identify K nearest neighbors
- Use class labels of the K nearest neighbors to determine the class label of the query sample (e.g., by taking majority vote)

Nearest neighbor classifier

- How to represent data samples
  - Binary features
  - Images
  - Text
  - Molecular sequences
  - Networks

- How to compute similarity between samples
  - Number of shared features
  - Number of shared features
  - Words in common
  - Number of shared subsequences
  - Number of shared subnetworks of different sizes

Probabilistic Modeling



Model the data with a probabilistic model

specifically, learn $P(features, label)$

$P(features, label)$ is the joint probability of features and label

Training the classifier

Examples	Label
$red, round, leaf, 3oz, \ldots$	$apple$
$green, round, no\ leaf, 4oz, \ldots$	$apple$
$yellow, curved, no\ leaf, 4oz, \ldots$	$banana$
green, curved, no leaf, 5oz, …	$banana$

Train →

probabilistic model: $p(features, label)$

Performing classification on novel data

Sample

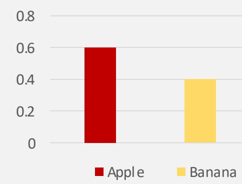
Classify

label

$red, round, leaf, 3oz, \ldots$

Probabilistic model:
$P(features, label)$
$P(label|features)$

$P(label|features)$



$P(label|features) = \frac{P(features, label)P(label)}{P(features)}$ by Bayes rule

$P(label|features) \propto P(features, label)P(label)$

Assign sample the most probable label given the features
It is provably the optimal way to label – minimizes classification error

What is the joint distribution of features and labels?

From product rule we have:

$$P(features, label) = P(x_1, x_2, \cdots x_n, y)$$
$$= P(y)\, P(x_1, x_2, \cdots x_n |\, y)$$
$$= P(y) P(x_1|y) p(x_2, \cdots x_n |\, y)$$
$$= P(y) P(x_1|y) P\left(x_{2|} \middle| y, x_1\right) P(x_3 \cdots x_n |\, y)$$
$$= P(y) \prod_{i=1}^{n} P(\, x_i \,|\, y, x_1 \cdots x_{i-1})$$

- chain rule!

PennState Institute for Computational and Data Sciences
Center for Artificial Intelligence Foundations & Scientific Applications
Artificial Intelligence Research Laboratory
PennState Clinical and Translational Science Institute

What is the joint distribution of features and labels?

- In the absence of some assumptions about the form of the distribution, there are too many probabilities to estimate
  - If the features and label are binary, we need to estimate 2^{N+1} probabilities
  - We seldom have enough labeled training data
- What can we do?
- Assume that the features are independent of each other given the label
- The resulting classifier is called the Naïve Bayes Classifier

$$P(features, label) = P(x_1, x_2, \cdots x_n, y)$$
$$= P(y)\, P(x_1, x_2, \cdots x_n |\, y)$$
$$= P(y) \prod_{i=1}^{n} P(\, x_i \,| y)$$

PennState College of Information Sciences And Technology

AI 100 Fall 2024 Vasant G Honavar

- chain rule!

Naïve Bayes classifier

- During training, for each label y and each value of each feature (random variable) estimate from the training data
  - $P(y)$
  - $P(x_1|\, y), P(x_2|\, y),\cdots, P(x_n|\, y)$

- During classification, for each label y, read off the corresponding probabilities from the learned distribution for the sample to be classified
  - $P(features, label) = P(y)\prod_{i=1}^{n} P(\, x_i\,|y)$
  - Assign the sample to the label for which $P(features, label)$ is maximum

- chain rule!

Example: Simple SPAM detector

- Suppose we have a small dataset of emails with their labels and some words that frequently appear in “Spam” and “Not Spam” emails.
- To keep things simple, let’s focus on just three words: “buy,” “cheap,” and “hello.”
- We encode each email according to the presence/absence of each of the three words along with the corresponding label
- We summarize the data using counts of each value of each feature in each type of email

Label	Buy	$Cheap$	$Hello$	#emails
$Spam$	30	25	5	40
$\neg Spam$	5	10	45	60

Example: Training a simple SPAM detector

Label	buy	$cheap$	$hello$	#emails
$spam$	30	25	5	40
$\neg spam$	5	10	45	60

$$P(spam) = \frac{40}{100} = 0.4$$

$$P(buy|spam) = \frac{30}{40} = 0.75 \qquad P(\neg buy|spam) = 1 - 0.75 = 0.25$$

$$P(cheap|spam) = \frac{25}{40} = 0.625 \qquad P(\neg cheap|spam) = \frac{25}{40} = 0.375$$

$$P(hello|spam) = \frac{5}{40} = 0.125 \qquad P(\neg hello|spam) = 0.875$$

Example: Training a simple SPAM detector

Label	*buy*	*cheap*	*hello*	#emails
spam	30	25	5	40
$\neg spam$	5	10	45	60

$$P(\neg spam) = \frac{60}{100} = 0.6$$

$$P(buy|\neg spam) = \frac{5}{60} \approx 0.0833 \qquad P(\neg buy|\neg spam) = 0.9177$$

$$P(cheap|\neg spam) = \frac{10}{60} \approx 0.167 \qquad P(\neg cheap|\neg spam) \approx 0.833$$

$$P(hello|\neg spam) = \frac{45}{60} = 0.75 \qquad P(\neg hello|\neg spam) = 0.25$$

Example: Classifying email using the SPAM detector

- Suppose we now receive an email that contains the words $buy, cheap$ but not $hello$. We need to calculate
  - $P(spam \mid buy, cheap, \neg\, hello)$
    $\propto P(spam)P(buy|spam)P(cheap|spam)P(\neg\, hello\,|\,spam)$
    $= \ (0.4)(0.75)(0.625)(0.875)$
    ≈ 0.1641
  - $P(\neg spam \mid buy, cheap, \neg\, hello)$
    $\propto P(\neg spam)P(buy|\neg spam)P(cheap|\neg spam)P(\neg\, hello\,|\neg spam)$
    $\approx (0.6)(0.0833)(0.167)(0.25)$
    ≈ 0.0021
  - $P(spam \mid buy, cheap, \neg\, hello) > P(\neg spam \mid buy, cheap, \neg\, hello)$
  - Hence, the email is more likely spam than not.

What if the Naïve Bayes assumption does not hold?

- We can learn Bayesian networks from data (not covered in this course)
- We can learn decision tree classifier

Learning Decision Trees – Playing 20 questions

Learning to predict whether Joe will enjoy AI major

- **You:** Are you interested in AI?
- **Me:** Yes
- **You:** Do you like math?
- **Me:** Yes
- **You:** Do you enjoy programming?
- **Me:** Yes
- **You:** I predict Joe will like AI major.

- How do you know what questions to ask, and in what order?
- How do you use the answers to the questions to make the right prediction?

Decision tree representation

In the simplest case,

- Each internal node tests on an attribute
- Each branch corresponds to an attribute value
- Each leaf node corresponds to a class label

In general,

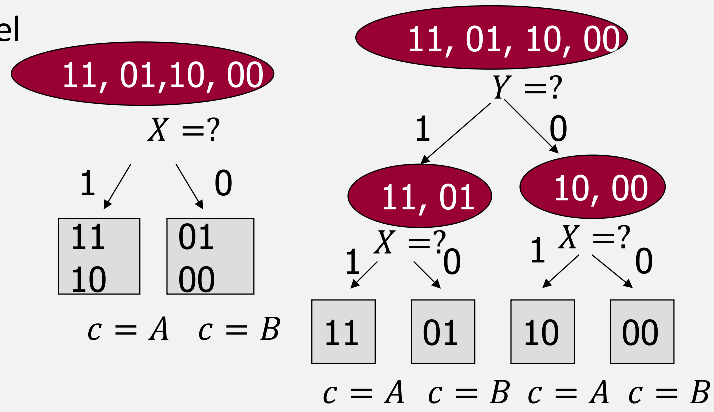
- Each internal node corresponds to a test (on input data sample)
  - Test outcomes are mutually exclusive and exhaustive
  - Each branch corresponds to an outcome of a test
  - Each leaf node corresponds to a class label

Learning is data compression

Attributes (x, y) — Label (c)

	x	y	c
1	1	1	A
2	0	1	B
3	1	0	A
4	0	0	B

Data set



Tree 1 Tree 2

Should we choose Tree 1 or Tree 2? Why?

Learning Decision Tree Classifiers

- Decision trees are especially well suited for representing simple rules for classifying data samples described by discrete feature values
- Decision tree learning algorithms
  - Implement Ockham's razor as a preference bias (simpler decision trees are preferred over more complex trees)
  - Are relatively efficient
  - Produce easy-to-understand classifiers
  - Are often among the first to be tried on a new data set

Learning Decision Tree Classifiers

- Ockham's razor recommends that we pick the simplest decision tree that is consistent with the training set
- Simplest tree is one that takes the fewest bits to encode (we will see why in a bit)
- There are far too many trees that are consistent with training data
- Searching for the simplest tree that is consistent with the training set is not typically computationally feasible

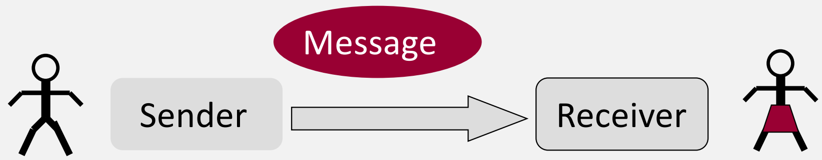
Solution

- Use a greedy algorithm – not guaranteed to find the simplest tree – but works well in practice
- Or restrict the space of hypothesis to a subset of simple trees

Information – Some intuitions

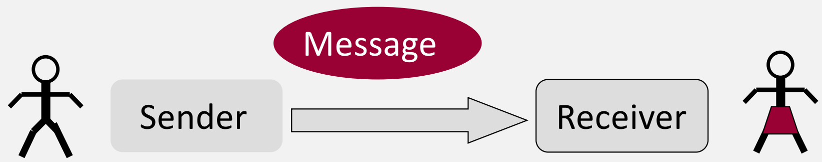
- Information reduces uncertainty
- Information is relative – to what you already know
- Information in a message is related to how surprising the message is
- Information depends on context

Digression: Information and Uncertainty



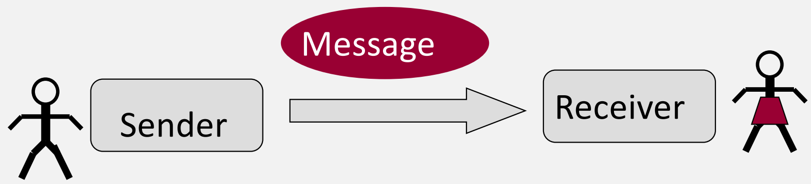
- You are stuck inside.
- You and I are both generally familiar with the weather in Pennsylvania.
- I do not lie, and you trust me.
- You send me out to report back to you on what the weather is like.
  - On a July afternoon in Pennsylvania, I walk into the room and tell you it is hot outside
  - On a January afternoon in Pennsylvania, I walk into the room and tell you it is hot outside

Digression: Information and Uncertainty



- On a July afternoon in Pennsylvania, I walk into the room and tell you it is hot outside
- On a January afternoon in Pennsylvania, I walk into the room and tell you it is hot outside
- Which of the above two messages is more informative?
- Whichever is more surprising given what you know about what the Pennsylvania weather is like in different months of the year!
  - “That it is hot outside on a January afternoon”
- It is this intuition that Claude Shannon formalized in information theory

Digression: Information and Uncertainty



- How much information does a message contain?
- If my message to you describes a scenario that you expect with certainty, the information content of the message for you is zero
- The more surprising the message to the receiver, the greater the amount of information conveyed by the message
- What does it mean for a message to be surprising?

Digression: Information and Uncertainty

- Suppose I have a coin with heads on both sides and you know that I have a coin with heads on both sides.
- I toss the coin, and without showing you the outcome, tell you that it came up heads.
- How much information did I give you? Zero!
- Suppose I have a fair coin and you know that I have a fair coin.
- I toss the coin, and without showing you the outcome, tell you that it came up heads.
- How much information did I give you? 1 bit

Measuring Information

- Without loss of generality, assume that messages are binary – made of 0s and 1s.
- Conveying the outcome of a fair coin toss requires 1 bit of information
  - Must specify which one out of two equally likely outcomes occurred
- Conveying the outcome of a random experiment (the value of a random variable) with 8 equally likely outcomes requires 3 bits
- Conveying an outcome of that is certain takes 0 bits
- In general, if an outcome has a probability *p*, the information content of the corresponding message is

$I(p) = -\log_2 p \qquad I(0) = 0$

Information is Subjective

- Suppose there are 3 agents – Sahar, Neil, Abhishek, in a world where a fair dice has been tossed.
- Sahar observes the outcome is a "6" and whispers to Neil that the outcome is "even" but Abhishek knows nothing about the outcome.
- Information gained by Sahar by looking at the outcome of the dice $= -\log_2\left(\frac{1}{6}\right) = \log_2 6$ bits.
- Sahar's uncertainty about the outcome $= 0$
- Information conveyed by Sahar to Abhishek = 0 bits.
- Abhishek's uncertainty about the outcome $= \log_2 6$ bits.
- Information gained by Neil from Sahar $= \log_2 3$ bits.
- Neil's uncertainty about the outcome after hearing from Sahar $= \log_2 6 - \log_2 3$ bits

Entropy of a random variable

- Suppose a random variable X that one of n values from its domain
- Suppose the p_i is the probability that X takes value x_i , that is, $P(X = x_i)$
- $I(p_i)$, information conveyed by the outcome x_i is $-\log_2 p_i$ (assuming $p_i \neq 0$. When $p_i = 0, I(p_i)$ is defined as 0)
- Then the expected information content of the message conveying the observation of the random variable is the entropy of X is given by

$$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

The concept extends naturally to sets of random variables if we replace probabilities by joint probabilities over the random variables in the set

Shannon's entropy as a measure expected information

- Let $P = (p_1, \cdots p_n)$ be a discrete probability distribution over the n disjoint values of a random variable
- Then the Shannon Entropy $H(X)$ of the random variable with distribution P is given by

$$H(X) = -\sum_{p=1}^{n} p_i \, I\,(p_i)$$

$$H(X) = -\sum_{p=1}^{n} p_i \log_2 (p_i)$$

Shannon entropy offers a measure of expected information supplied by observing a random variable

Shannon's entropy as a measure expected information

- Let $P = (p_1, \cdots p_n)$ be a discrete probability distribution over the n disjoint values of a random variable
- Then the Shannon Entropy $H(X)$ of the random variable with distribution P is given by

$$H(X) = -\sum_{p=1}^{n} p_i \log_2 (p_i)$$

When X denotes the outcome of a fair coin toss,

$$H(X) = -\left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right)\log_2\left(\frac{1}{2}\right) = 1 \text{ bit}$$

When X denotes the outcome of a rigged (two-headed) coin toss,

$$H(1,0) = -(1)\log_2(1) - 0(0) = 0 \text{ bit}$$

Coding Theory Perspective

- Suppose you and I both know the distribution P over outcomes of X
- I choose an outcome according to P
- Suppose I want to send you a message about the outcome observed
- You and I could agree in advance on the questions
- I can simply send you the answers
- Optimal message length on average is $H(X)$

Conditional entropy

Conditional entropy of Y given X is the residual uncertainty of Y after we have observed X

$$H(Y|X) = -\sum_{x} P(x) \sum_{y} P(y|x) \log_2 P(y|x)$$

Example:

Rain and Cloudy are two binary random variables

$P(cloudy) = 0.5;$ $P(\neg cloudy) = 0.5$

$P(rain|cloudy) = 0.8;$ $P(\neg rain|cloudy) = 0.8$

$P(rain|\neg cloudy) = 0.1;$ $P(\neg rain|\neg cloudy) = 0.9$

Calculate $H(Rain|Cloudy)$

Conditional entropy

Conditional entropy of Y given X is the residual uncertainty of Y after we have observed X

$$H(Y|X) = -\sum_x P(x) \sum_y P(y|x) \log_2 P(y|x)$$

Example:

$$H(Rain|Cloudy) = -P(cloudy)H(Rain|cloudy) + P(\neg cloudy)\, H(Rain|\neg cloudy)$$

$$= -(0.5)[0.8 \log 0.8 + 0.2 \log 0.2] -(0.5)\,[0.1 \log 0.1 + 0.9 \log 0.9]$$

$$= -0.5[(0.8)(0.3219 + (0.2)(-2.3219)] -0.5[(0.1)(-3.3219) + (0.9)(0.152)]$$

Joint Entropy and Conditional Entropy

Some Useful results :

$$\left. \begin{aligned} H(X,Y) &\le H(X) + H(Y) \\ H(Y \mid X) &\le H(Y) \end{aligned} \right\}$$

(When do we have equality?) ← When Y is independent of X

Chain rule for Entropy

$$\begin{aligned} H(X,Y) &= H(X) + H(Y|X) \\ &= H(Y) + H(X|Y) \end{aligned}$$

These results follow the definition of entropy and the laws of probability

Example of entropy calculations

$$P(X = H; Y = H) = 0.2.\ P(X = H; Y = T) = 0.4$$

$$P(X = T; Y = H) = 0.3.\ P(X = T; Y = T) = 0.1$$

$$H(X,Y) = -0.2\log_2 0.2 - 0.3\log_2 0.3 - 0.4\log_2 0.4 - 0.1\log_2 0.1 \approx 1.85$$

$$P(X = H) = 0.6.\ H(X) = 0.97$$

$$P(Y = H) = 0.5.\ H(Y) = 1.0$$

$$P(Y = H | X = H) = 0.2/0.6 = 0.333$$

$$P(Y = T | X = H) = 1 - 0.333 = 0.667$$

$$P(Y = H | X = T) = 0.3/0.4 = 0.75$$

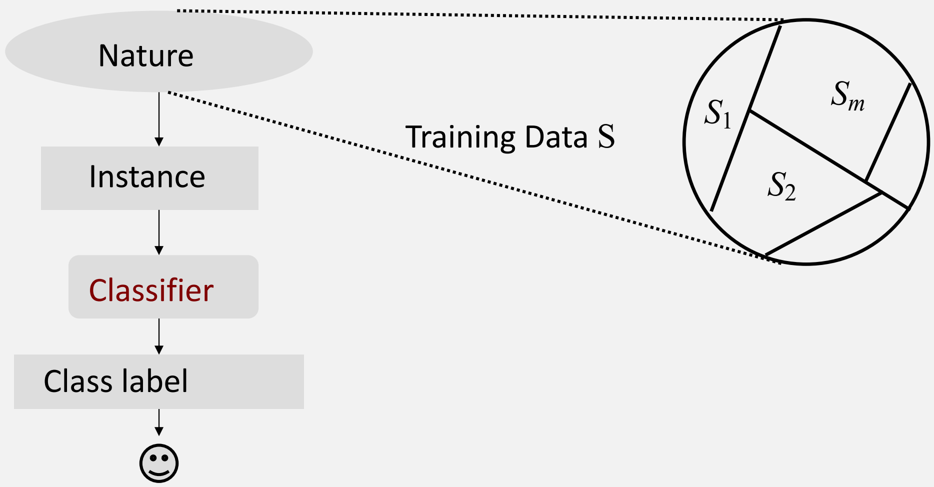
$$P(Y = T | X = T) = 0.1/0.4 = 0.25$$

$$H(Y|X) = H(X,Y) - H(X) = 0.88$$

Decision Tree Classifiers

- Think of class labels as random variables.
- Then the entropy of the data is simply the entropy of the label distribution $H(Y)$
- To classify a random sample from nature, we need $H(Y)$ bits of information
- The learner's task is to figure out the questions to ask about the features of the data and the order in which to ask them to gain the information necessary to classify the sample

Decision Tree Classifiers



S_i is the multi-set of data samples with label y_i

Learning Decision Tree Classifiers

- The amount of information present in the training data is equal to the entropy of the distribution
- The task of the learner then is to identify a series of questions that optimally extract the information needed to classify samples from the distribution
- The learned model is stored in the form of a decision tree

Learning Decision Tree Classifiers

Information gain based decision tree learner

- Start with the entire training set at the root
- Recursively add nodes to the tree
  - The nodes correspond to the questions that yield the greatest expected reduction in entropy
  - That is, the greatest reduction in uncertainty about the class label for samples to be classified
    - until some termination criterion is met (e.g., the training data at every leaf node has zero entropy)

Information gain based decision tree learner

- Base case: If all data belong to the same class, create a leaf node with that label
- Otherwise, recursively
  - Calculate the entropy reduction for each feature if we use it to split the data based on the values of the feature
  - Pick the feature that yields the greatest reduction in entropy, partition the data based on values of the feature

Learning Decision Tree Classifiers - Example

Samples: ordered 3-tuples of attribute values corresponding to

Height ∈{*tall*, *short*}

Hair ∈ {*dark*, *blonde*, *red*}

Eye ∈ {*blue*, *brown*}

Training Data

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	–
I_4	t, r, l	–
I_5	s, b, l	–
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

Learning Decision Tree Classifiers - Example

Training Data

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	–
I_4	t, r, l	–
I_5	s, b, l	–
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

$$H(Class) = -\left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right) - \left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right) = 0.954\text{bits}$$

$$H(Class|Height = t) = -\left(\frac{3}{5}\right)\log_2\left(\frac{3}{5}\right) - \left(\frac{2}{5}\right)\log_2\left(\frac{2}{5}\right)$$

$$H(Class|Height = s) = -\left(\frac{2}{3}\right)\log_2\left(\frac{2}{3}\right) - \left(\frac{1}{3}\right)\log_2\left(\frac{1}{3}\right)$$

$$\begin{aligned} H(Class|Height) &= P(Height = t)\, H(Class|Height = t) \\ &\quad + P(Height = s)\, H(Class|Height = s) \\ &= \left(\frac{5}{8}\right)(0.971) + \left(\frac{3}{8}\right)(0.918)\text{bits} \\ &= 0.95 \text{ bits} \end{aligned}$$

Learning Decision Tree Classifiers - Example

Training Data

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	–
I_4	t, r, l	–
I_5	s, b, l	–
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

$$H(Class) = -\left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right) - \left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right) = 0.954\text{bits}$$

$$H(Class|Hair = d) = 0 \text{ bits}$$

$$H(Class|Hair = b) = 1 \text{ bit}$$

$$H(Class|Hair = r) = 0 \text{ bit}$$

$$H(Class|Hair) = \left(\frac{4}{8}\right)(1) + 0 + 0 = 0.5 \text{ bits}$$

Learning Decision Tree Classifiers - Example

Training Data

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	–
I_4	t, r, l	–
I_5	s, b, l	–
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

$$H(Class) = -\left(\frac{5}{8}\right)\log_2\left(\frac{5}{8}\right) - \left(\frac{3}{8}\right)\log_2\left(\frac{3}{8}\right) = 0.954\text{bits}$$

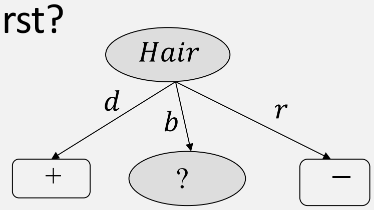
$$H(Class|Eye = l) = 0.971 \text{ bits}$$

$$H(Class|Eye = w) = 0 \text{ bit}$$

$$H(Class|Eye) = \left(\frac{5}{8}\right)(0.971) + 0 = 0.607 \text{ bit}$$

Which feature to ask about first?

$H(Class|Height) = 0.95$ bits
$H(Class|Hair) = 0.5$ bits
$H(Class|Eye) = 0.607$ bit

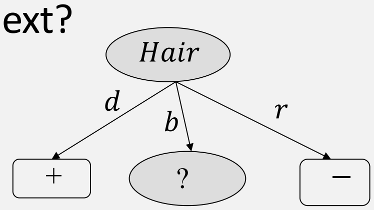


- Which feature is most informative about the class?
- Whichever has the lowest conditional entropy!
- Which in this case is $Hair$
- We pick $Hair$ at the root of the decision tree
- Are we done?
- No because $H(Class|Hair) \neq 0$
- We focus on the node(s) with non-zero entropy

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	−
I_4	t, r, l	−
I_5	s, b, l	−
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

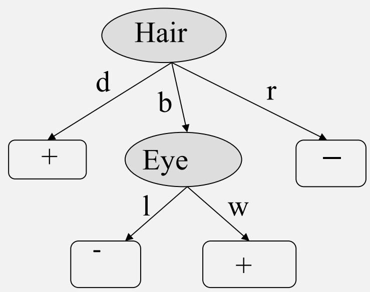
Which feature to ask about next?

- Among $Height$ and Eye, which is most informative about the $Class$ for the remaining data?
- We estimate at any node with non-zero entropy we calculate
  - $H(Class|Height)$
  - $H(Class|Eye)$
- In this example, there is only one such node with the data shown
  - $H(Class|Height)$=1 bit
  - $H(Class|Eye)$= 0 bit
- Which feature do we pick?
- Eye



	Height, Hair, Eye	Class
I_3	t, b, l	−
I_5	s, b, l	−
I_6	t, b, w	+
I_8	s, b, w	+

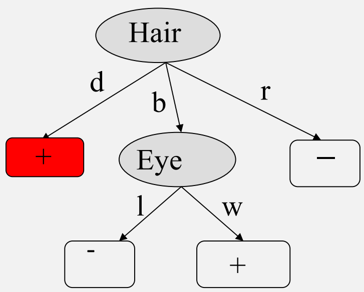
Now we have the decision tree



Training Data

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	−
I_4	t, r, l	−
I_5	s, b, l	−
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

How to classify data samples?

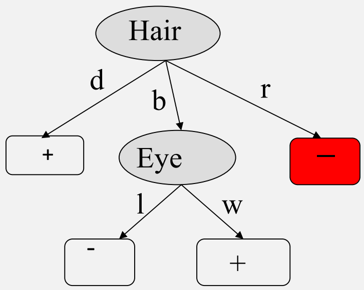


Training Data

	Height, Hair, Eye	Class
I_1	t, d, l	+
I_2	s, d, l	+
I_3	t, b, l	−
I_4	t, r, l	−
I_5	s, b, l	−
I_6	t, b, w	+
I_7	t, d, w	+
I_8	s, b, w	+

- Suppose we have a novel sample
  Height is short, *Hair* is dark, and *Eye* is brown
- What is the label?
- Just ask about the features based on the tree and follow the branches until you end up at a leaf node and read off the label

How to classify samples



Training Data

Height, Hair, Eye	Class
I_1 t, d, l	+
I_2 s, d, l	+
I_3 t, b, l	−
I_4 t, r, l	−
I_5 s, b, l	−
I_6 t, b, w	+
I_7 t, d, w	+
I_8 s, b, w	+

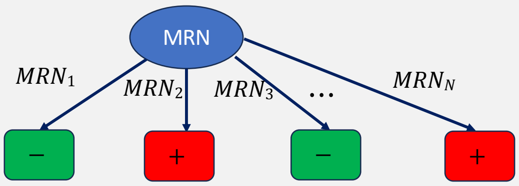
- Suppose we have a novel sample
  Height is short, *Hair* is red, and *Eye* is blue?
- What is the label?

There is one problem

- Consider a clinical database of patient records
- You want to learn a decision tree for diagnosing cancer patients
- Suppose one of the columns of the database records the medical record number (MRN)
- Suppose the hospital gives you the data, but fails to tell you that one of the columns is the MRN
- You build a decision tree that has perfect accuracy on training data
- Now you are given a new patient to diagnose
- What do you expect will happen?

There is one problem

- The tree learned has MRN at the root of the tree
- MRN is unique for each training sample
- What kind of tree will you get?



- What happens when you get a new patient?
- The MRN has never been seen before
- The decision tree fails to classify the patient
- In fact, the decision tree fails on any patients not in the training data

Fixing the problem with entropy

- Method favors features with many values as compared to features with fewer values – extreme case occurs when the feature has unique values (MRN, SSN etc.)
- How can we avoid this problem?
- Normalize entropy by a factor that depends on the value distribution of the feature to obtain entropy ratio
- Use entropy ratio instead of entropy to decide on the feature to choose at each step in constructing the decision tree

$$EntropyRatio(S \mid A) \equiv \frac{Entropy(S \mid A)}{SplitEntropy(S \mid A)}$$

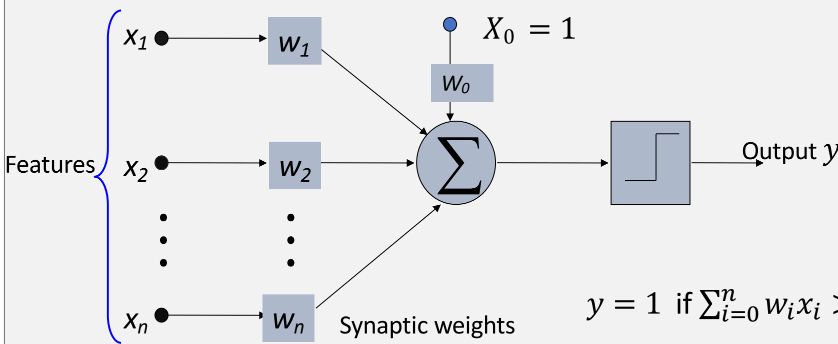
$$SplitEntropy(S \mid A) \equiv -\sum_{i=1}^{|Values(A)|} \frac{|S_i|}{|S|} \log_2 \frac{|S_i|}{|S|}$$

Avoiding overfitting on the training data

- At each successive level, the number of training data samples available to estimate entropy decreases
- Entropy estimates become less and less reliable
- The decision tree overfits the training data
- We should stop before all leaf nodes have zero entropy to avoid overfitting
- This is achieved by
  - Checking if the split suggested by the algorithm is statistically significantly different from a random split
  - Pruning the fully built tree based on accuracy on validation data

Simple neural network

- Remember the McCulloch-Pitts neuron?



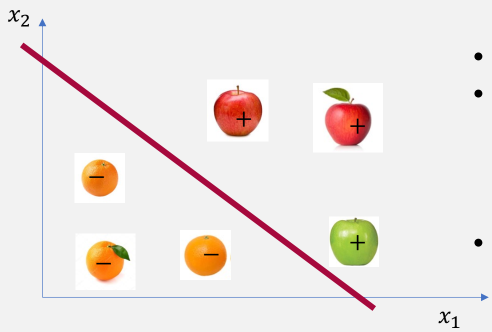
$y = 1$ if $\sum_{i=0}^{n} w_i x_i > 0$

$y = -1$ otherwise

The neuron acts like a classifier

McCulloch-Pitts Neuron as a Classifier

- If the output of the neuron for input pattern X_p is +1 then X_p is assigned to class C_1
- If the output is -1 then the pattern X_p is assigned to C_2

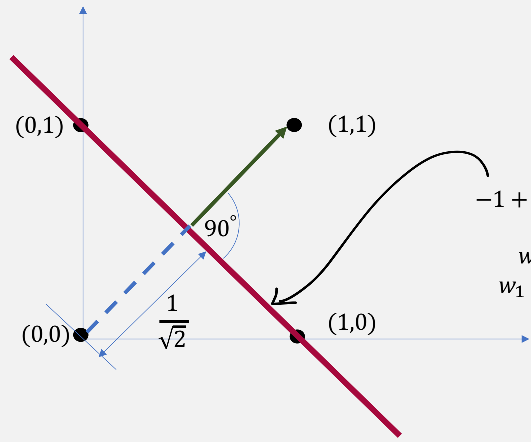


- Feature are numeric
- Data samples live in a Euclidian space of as many dimensions as there are features
- The neuron with the ``right'' weights implements a separating hyperplane

The weights of the neuron specify the hyperplane

Hyperplane is defined by

$$\sum_{i=0}^{n} w_i x_i = 0$$

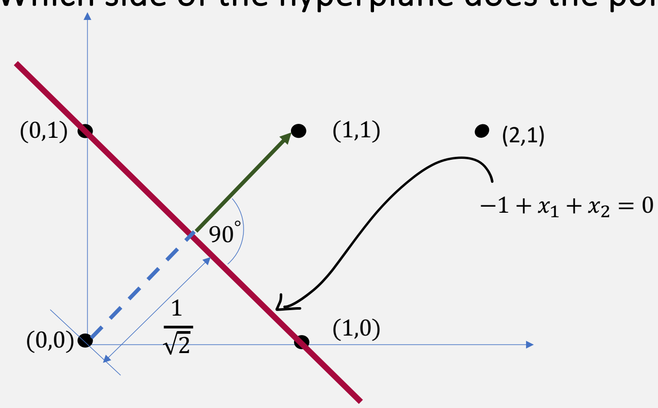


$-1 + x_1 + x_2 = 0$

$w_0 = -1$
$w_1 = w_2 = 1$

-1

Which side of the hyperplane does the point (2,1) lie?

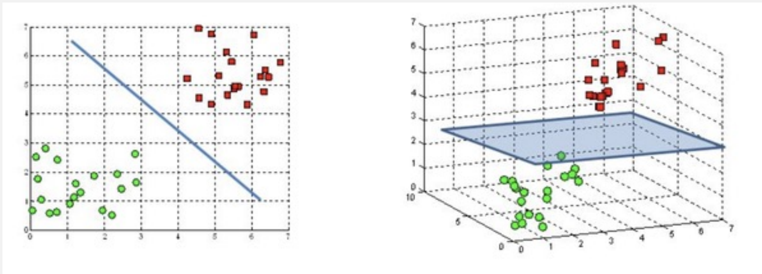


- Check the sign of $w_0 + \sum_{i=1}^{2} w_i x_i = -1 + 2 + 1 = 2$
- The sign is positive, so on the “positive side” pointed by the direction of the positive normal

-1

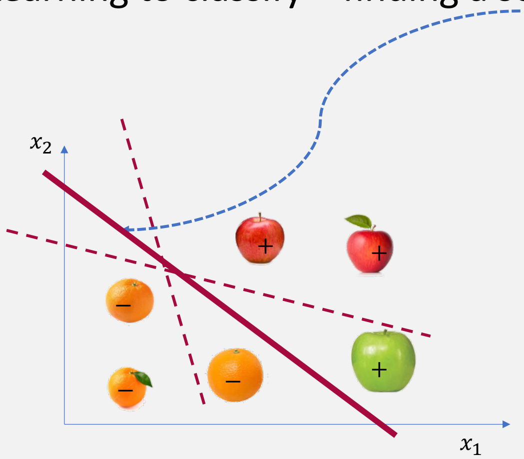
so

McCulloch-Pitts Neuron - Connection to geometry



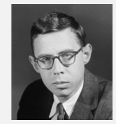
- A perceptron with 3 weights $[w_0, w_1, w_2]$implements a line in 2-D
  A perceptron with 4 weights $[w_0, w_1, w_2, w_4]$ implements a plane in 3-D
- A perceptron with $n+1$ weights $[w_0, \cdots, w_n]$ implements an $(n-1)$ dimensional hyperplane in n-D
  - Dividing the n-D space into two half spaces

Learning to classify = finding a separating hyperplane

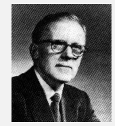


- Given a training set of labeled samples where the labels specify one of two classes (say $+1$ and -1, e.g., corresponding to apple and orange)
  - the goal is to learn the weights such that the resulting hyperplane correctly classifies the training samples

Rosenblatt's Perceptron Learning Algorithm



Donald Hebb: Neurons that fire together wire together



- Initialize all weights to 0
- Choose a learning rate $\eta > 0$
- Repeat until a complete pass through the training set results in no changes to the weights
  - For each training example X_p, d_p (X_p is the input, $d_p = +1$ or -1 is the desired output for that input)
    - Compute the output y_p:
      - $y_p = 1$ if $\sum_{i=0}^{n} w_i x_{ip} > 0$; $y_p = -1$ otherwise
      - If $d_p \neq y_p$, update each weight:
        - $w_i \leftarrow w_i + (d_p - y_p)\, x_{ip}$

Perceptron learning algorithm – Example

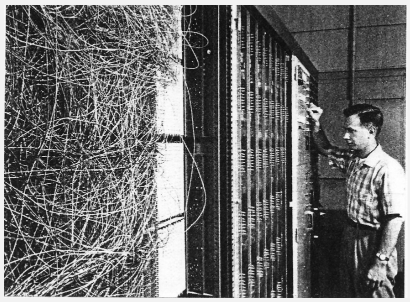
$\eta = \frac{1}{2}$

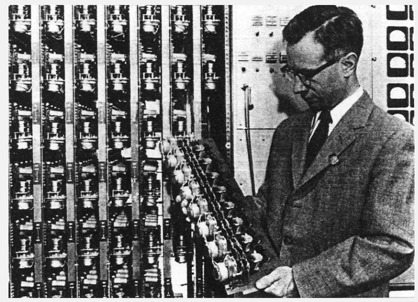
Let $\{(1,1,1),(1,1,-1),(1,0,-1)\}$ be samples with desired output $= +1$

And $\{(1,-1,-1),(1,-1,1),(1{,}0,1)\}$ be samples with desired output $= -1$

$\boldsymbol{X}_k$	d_k	w_0, w_1, w_2	$\sum_{i=0}^{2} w_i\, x_{kp}$	y_k	**Update?**	Updated w_0, w_1, w_2
(1, 1, 1)	1	(0, 0, 0)	0	-1	Yes	(1, 1, 1)
(1, 1, -1)	1	(1, 1, 1)	1	1	No	(1, 1, 1)
(1,0, -1)	1	(1, 1, 1)	0	-1	Yes	(2, 1, 0)
(1, -1, -1)	-1	(2, 1, 0)	1	1	Yes	(1, 2, 1)
(1,-1, 1)	-1	(1, 2, 1)	0	-1	No	(1, 2, 1)
(1,0, 1)	-1	(1, 2, 1)	2	1	Yes	(0, 2, 0)
(1, 1, 1)	1	(0, 2, 0)	2	1	No	(0, 2, 0)

Perceptron – the first learning machine

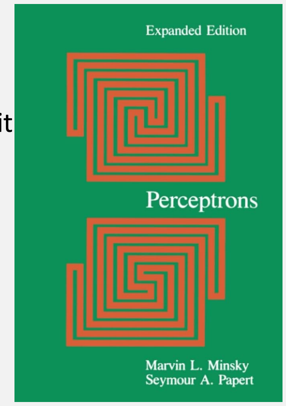


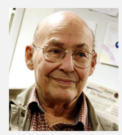


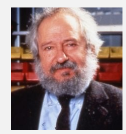
Frank Rosenblatt with his neural network circa 1957-1960

Limitations of linear machines

- Linear machines can deal with only simple learning tasks where the boundary between classes is linear
- If there is a linear boundary, the algorithm will find it
- If there is no linear boundary that separates the data, the algorithm will run for ever
- Easy to generalize to many classes instead of only two classes
- More complex tasks need multi-layer networks
- Minsky in 1969 argued that attempts to develop them were futile
- This shifted the focus away from machine learning Decades later they were proven wrong!
- Machine learning is now the primary means of building AI systems for many practical applications







Deep neural networks

- More complex tasks need multi-layer networks
- Training them requires more sophisticated methods
- Their development took decades
- Modern deep neural networks are generalizations of linear classifiers
- Variants of deep neural networks currently offer state-of-the-art methods for many problems
    - Image classification
    - Speech recognition
    - Protein structure prediction