An Introduction of Machine Learning

2024

Textbooks

- David Freeman, Clarence Chio*, Machine Learning and Security*, O'Reilly Media, Inc., 2018
- Emmanuel Tsukerman, *Machine Learning for Cybersecurity Cookbook*, Packt Publishing Ltd, 2019.

- Mitchell, T.M., *Machine learning*. McGraw-Hill, New York, 1997.
- Sumeet Dua, Xian Du, *Data Mining and Machine Learning in Cybersecurity*, Auerbach Publications, 2011

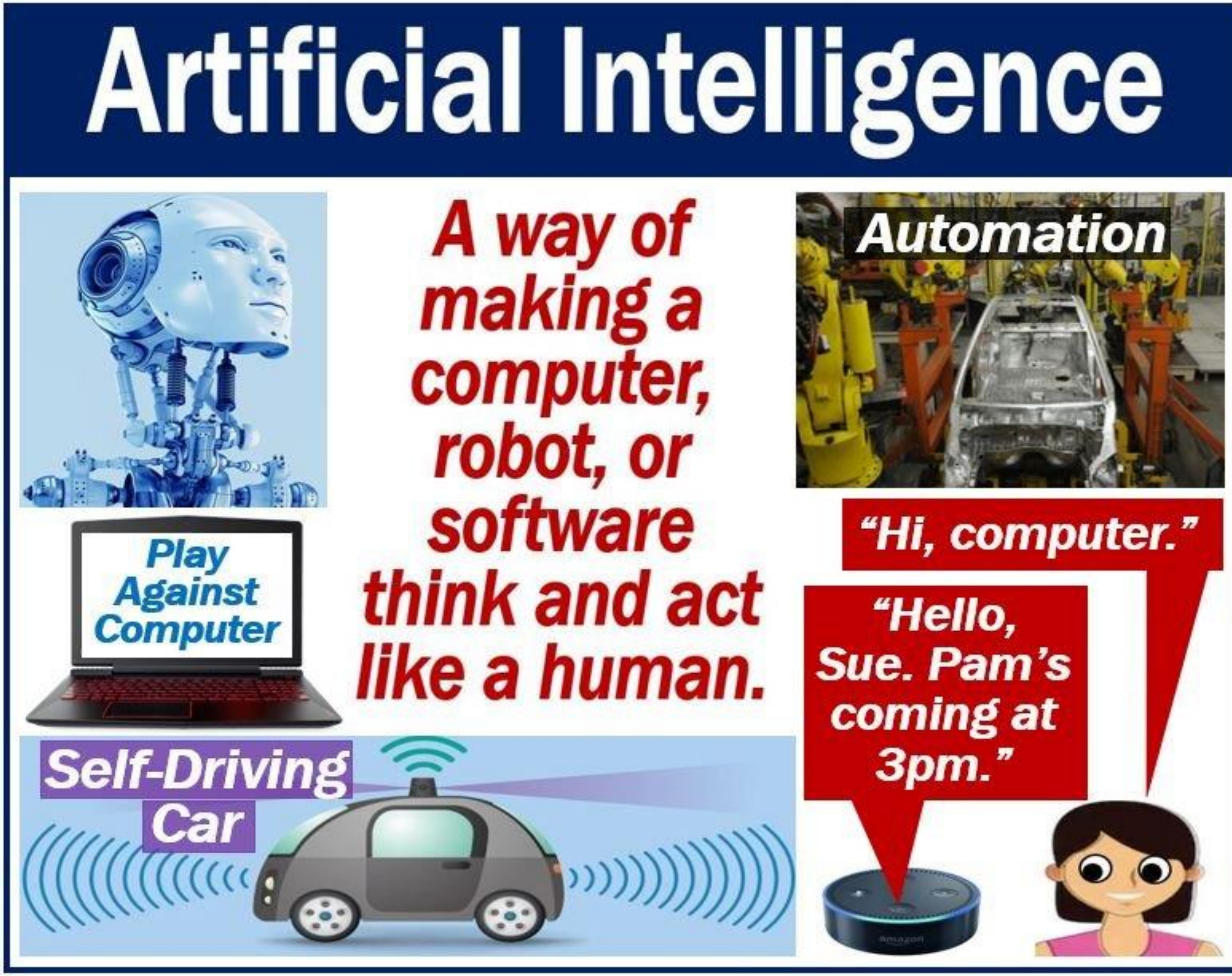
Introduction

How to make computers to learn?

Learn - improve automatically with experience

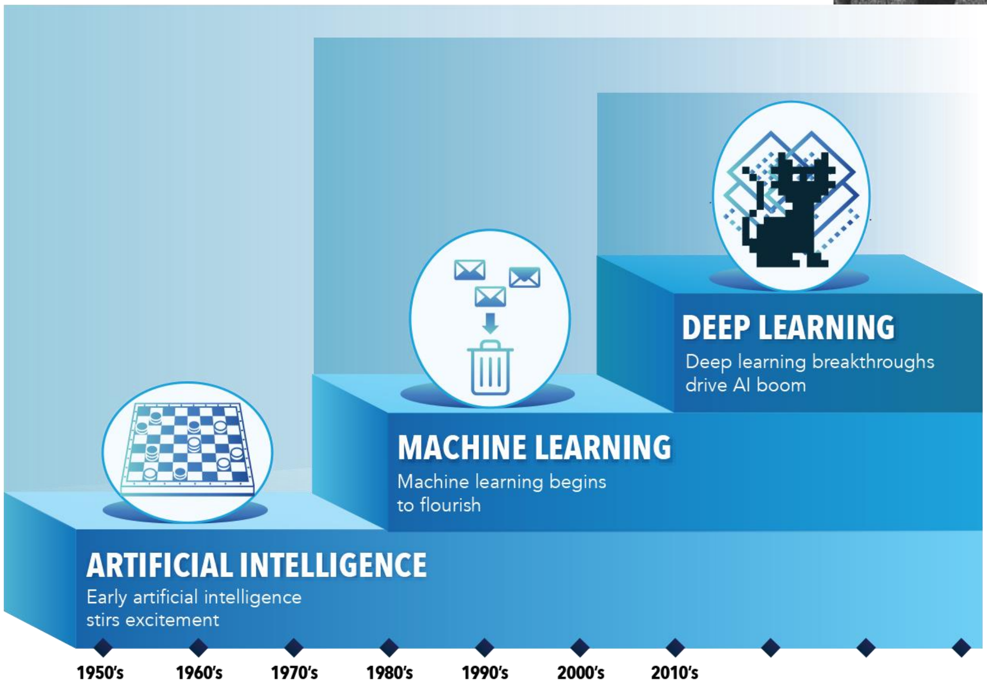
- *Data Mining* – using historical data to improve decisions
  - learning from medical records which treatments are most effective
- *Self customizing programs*
  - houses learning to optimize energy costs based on particular usage patterns of their occupants
  - personal software assistants learning the evolving interests of their users in order to highlight relevant stories from online newspapers
- *Software applications:* we can't program by hand
  - autonomous driving,
  - speech recognition

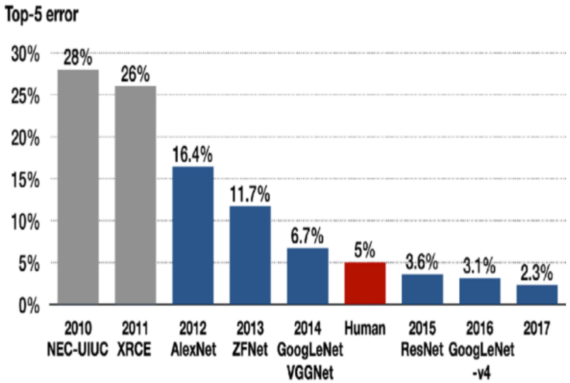
Machine learning might lead to a better understanding of human learning abilities (and disabilities)



https://marketbusinessnews.com/financial-glossary/artificial-intelligence/

AI





https://www.image-net.org/challenges/LSVRC/

RECENT AI TRENDS

AI has greatly contributed to supporting human:

- *Improving human's ability to solve problems* in decision-making, production line control (through intelligent robots in industrial production lines), big data analysis (analysis of medical and biological data);
- *Improving human's living and working conditions:* automation simplifies workers' work, intelligent entertainment technologies, ...
- *Building capacity for new human abilities:* forecasting situations, supporting decision making

AI systems are classified into two main categories:

- *Narrow AI (weak AI, specialized AI)*: is a concept that refers to AI systems designed for a specific problem or limited in scope such as automatic translation, voice identification, or human face recognition in videos. Most of the current AI applications fall into this category.
- *General AI (strong AI):* is a concept that refers to AI systems capable of being similar to or better than human when solving a wide class of intelligence-demanding problems. Building a system with strong AI is a difficult and not feasible in the near future

RECENT AI TRENDS

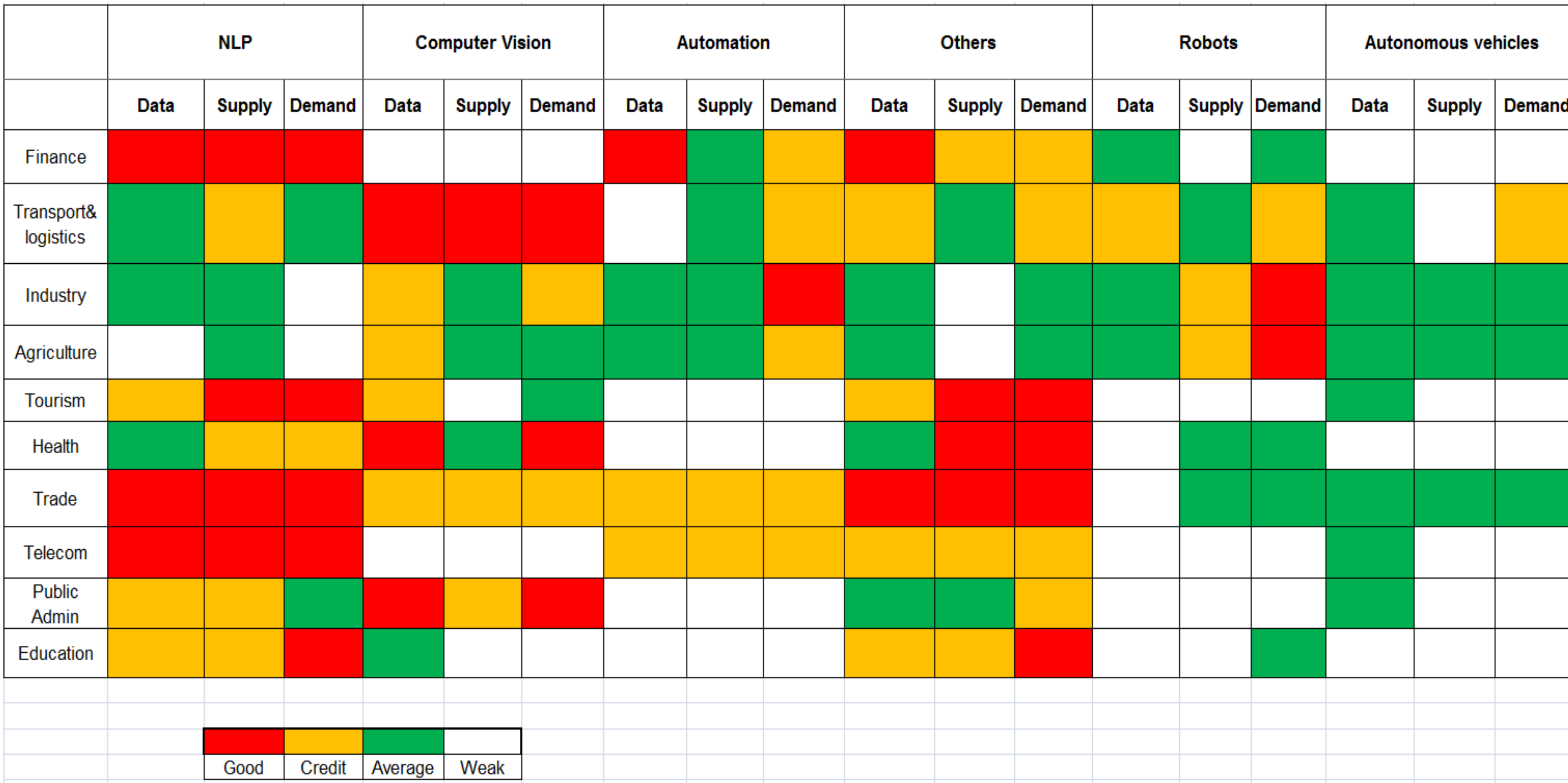
PATENTS 1960-2018

1960- 2018: 340.000 applications (US and China the top list)

	Machine learning	Computer vision	Natural language processing	Speech processing	Control methods	Planning and scheduling	Robotics
Telecommunications	16,201	22,871	7,553	12,549	3,496	2,601	2,476
Transportation	13,741	21,744	2,330	3,997	14,030	3,614	5,080
Personal devices, computing and HCI	11,585	17,164	7,920	6,678	1,625	1,663	1,416
Life and medical sciences	18,772	17,098	3,818	2,504	1,494	1,617	1,988
Security	8,813	17,235	3,033	3,075	1,162	1,401	793
ocument management and publishing	6,841	11,530	9,526	3,291	163	517	221
Business	9,709	7,968	5,850	2,422	271	1,381	350
Industry and manufacturing	9,569	5,573	3,031	798	1,262	2,404	1,073
Physical sciences and engineering	8,330	5,397	1,284	1,183	1,540	721	679
Networks	5,296	3,659	2,350	1,498	343	789	380
Arts and humanities	2,489	4,852	2,669	2,615	237	273	371
Education	3,914	3,767	1,642	1,951	284	365	372
Cartography	3,276	3,334	1,610	759	697	697	257
Energy management	3,766	1,056	397	309	734	944	336
Entertainment	1,822	2,890	737	1,087	309	199	528

Source: WIPO Technology Trends 2019: Artificial Intelligence

AI ADOPTION SECTORS IN VIETNAM



Source: MOST

Relevant disciplines

Machine Learning is involved many diverse disciplines:

Artificial Intelligence

Probability and Statistics

Computational Complexity

Information Theory

Psychology and Neurobiology

Control Theory

Philosophy

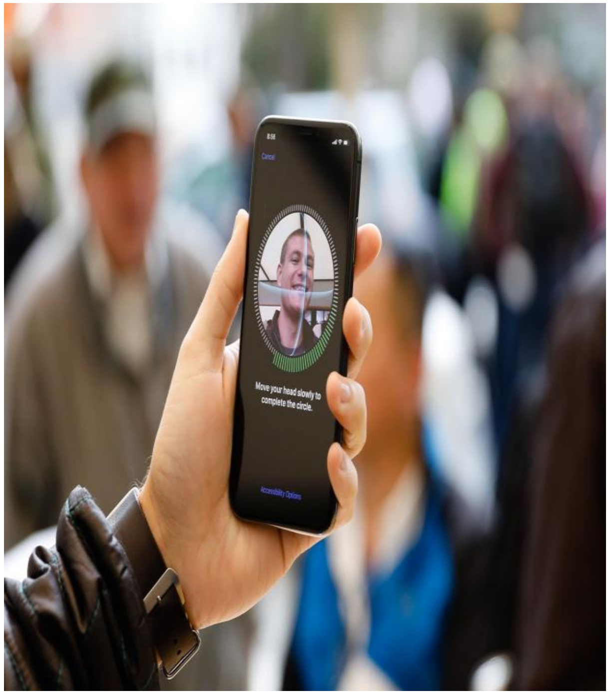
Successful applications of ML

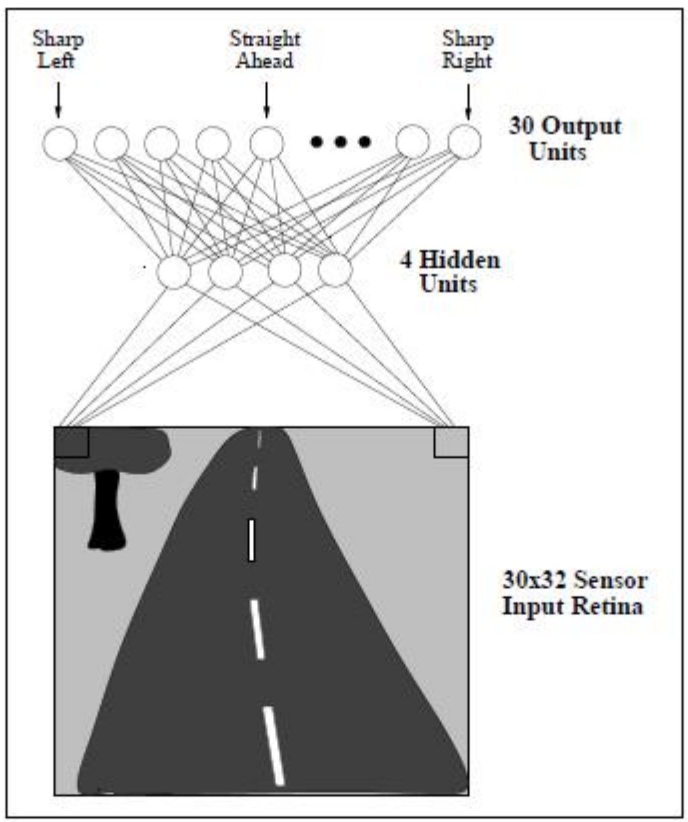
Learn to recognize written & spoken words

Learn to detect and recognize faces

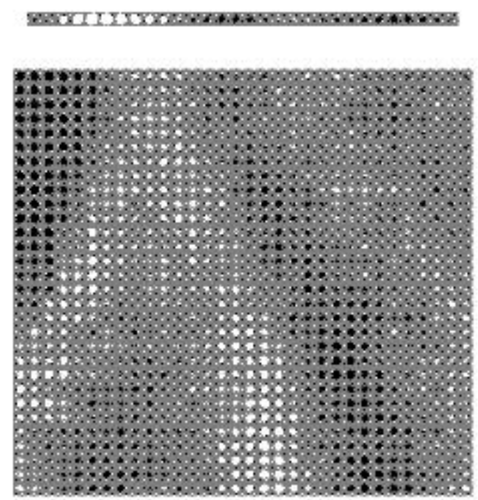
Learn to drive autonomous vehicles

Learn to play games (chess games,…)





Sharp Left
Straight Ahead
Sharp Right
30 Output Units
4 Hidden Units
30x32 Sensor Input Retina



I. Learning problems

Definition: A computer program is said to learn from experience **E**

with respect to some class of tasks **T**

and performance measure **P**,

if its performance at tasks in **T**,

as measured by **P**,

improves with experience **E**.

Example Learning Problems

Handwriting Recognition:

T: recognising and classifying handwritten words with images

P: percent of words correctly classified

E: a database of handwritten words with given classifications

Robot driving:

T: driving on public four lane highways using vision sensors

P: average distance traveled before an error (as judged by human overseer)

E: a sequence of images and steering commands recorded while observing a human driver

Example Learning Problems

Learn to play checkers

T: play checkers

E: Oppotunity to play against self

P: % Game won in the world tournament

Choice of P very important

Expert system or human comprehension? Data-mining!!

Definition Continued

This definition of learning is broad enough to include most tasks that we would call "learning tasks"

It is also broad enough to encompass computer programs that improve from experience in quite straightforward ways.

Example: A database system that allows users to update data entries - it improves its performance at answering database queries, based on the experience gained from databases updates

II. Designing a Learning System

For a specific problem

What experience?

What exactly should be learned?

How shall it be represented?

What specific algorithm to learn it?

Choose Training Experience (1) *Direct versus Indirect Learning*

Training experience provide direct or indirect feedback?

1. Individual checkers board states and correct move for each
2. Move sequences and final outcomes of various games played

Credit assignment problem - the degree to which each move in the sequence deserves credit or blame for the final outcome - game can be lost even when early moves are optimal, if these are followed later by poor moves or vice versa

Choose Training Experience (2) *Teacher or not?*

Degree to which learner controls the sequence of training examples

1. Teacher selects informative board states & provides the correct moves

2. For each proposed board state the learner finds particularly confusing it asks the teacher for correct move

3. Learner may have complete control as it does when it learns by playing itself with no teacher - learner may choose between experimenting with novel board states or honing its skill by playing minor variations of promising lines of play

Choose Training Experience (3)

How well training experience represents the distribution of examples over which the final system performance P must be measured

In general, learning is the most reliable when training examples follows a distribution similar to that of future test examples.

For checkers learning problem: P is percent of games in the world tournament, ->obvious danger when E consists of only games played against itself (probably can't get world champion to teach computer!)

Most current theories of machine learning assume that the distribution of training examples is identical to the distribution of test examples

It is IMPORTANT to keep in mind that this assumption must often by violated in practice.

Choose a Target Function

ChooseMove: B -> M where B is any legal board state and M is a legal move (hopefully the "best" legal move)

Alternatively, function *V: B -> $\Re$* which maps from B to some real value where higher scores are assigned to better board states

Now use the legal moves to generate every subsequent board state and use V to choose the best one and therefore the best legal move

Choose a Target Function II

Let us define the target value V(b) for an arbitrary board state b in B:

V(b) = 100, if b is a final board state that is won
V(b) = -100, if b is a final board state that is lost
V(b) = 0, if b is a final board state that is a draw
V(b) = V(b´), if b is not a final state where b´ is the best final board state starting from b assuming both players play optimally

This recursive style is not efficiently computable!! - non-operational definition (changes over time!!!)

Need Operational V - What are Realistic Time Bounds??

May be difficult to learn an operational form of V perfectly
-> We need function approximation for V

Choose Representation for Target Function

Use a large table with an entry specifying a value for each distinct board state

Collection of rules that match against features of the board state

Quadratic polynomial function of predefined board features

Artificial neural network

NOTICE - choice of representation is closely tied to algorithm choice!!

Expressability Tradeoff

Very expressive representations allow close approximations to the ideal target function V, but the more expressive the representation the more training data the program will require in order to choose among the alternative hypothesis

Also depending on the purpose, a more expressive representation might make it more or less easy for people to understand!

Choose SIMPLE Representation

We choose: a linear combination of

- *X_1 the number of black pieces on the board*
- *X_2 the number of red pieces on the board*
- *X_3 the number of black kings on the board*
- *X_4 the number of red kings on the board*
- *X_5 the number of black pieces threatened by red (which can be captured on red's next turn)*
- *X_6 the number of red pieces threatened by black*

$V'(b) = w_0+w_1x_1+w_2x_2+w_3x_3+w_4x_4+w_5x_5+w_6x_6$

where w_0 through w_6 are numerical coefficients or weights to be chosen by the learning algorithm

Design So Far

T: Checkers

P: percent of games won in world tournament

E: games played against self

V: Board -> $\Re$

Target Function Representation:

$$V'(b) = w_0 + w_1x_1 + w_2x_2 + w_3x_3 + w_4x_4 + w_5x_5 + w_6x_6$$

Choose Function Approximation Algorithm

First, need a set of training examples

<b,V_{train}(b)>

i.e:

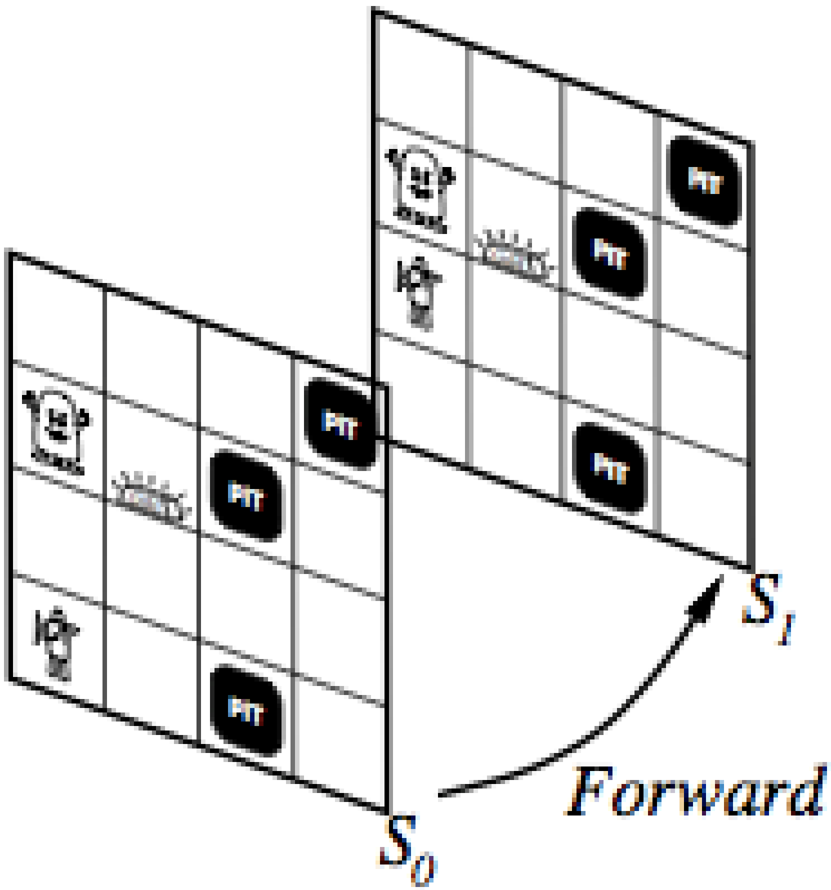
<(x_1=3,x_2=0,x_3=1,x_4=0,x_5=0,x_6=0),+100>

Rule for estimating training values:

$$V_{train}(b) \leftarrow V'(successor(b))$$

Good if V´ tends to be more accurate for board positions closer to game's end

Successor



Choose Learning Algorithm

Learning Algorithm for choosing weights (*hypothesis*) w_i to best fit the set of training examples

$\{<b, V_{train}(b)>\} \equiv \{<b, V'(Successor(b))>\}$

"Best fit" could be defined as minimizes the squared error Er

$$E \equiv \sum_{<b, V_{train}(b)> \in \, training-examples} (V_{train}(b) - V'(b))^2$$

Choose learning Algorithm II

We seek the weights that minimise E for the observed training examples

We need an algorithm that incrementally refines the weights as new training examples become available & is robust to errors in estimated training values

One such algorithm is LMS (basis of Neural Network algorithms)

Least Mean Squares

LMS adjusts the weights a small amount in the direction that reduces the error on this training example

Stochastic gradient-descent search through the space of possible hypothesis to minimize the squared error

Why stochastic ??? *(using estimation of gradient)*

LMS Algorithm

For each $<b, V_{train}(b)>$,

1. Use current weights to calculate $V'(b)$.
2. For each weight

$$w_i \leftarrow w_i + \eta \left(V_{train}(b) - V'(b) \right) x_i$$

Where η is a small constant (i.e .01) that moderates the size of the weight update

LMS Intuition

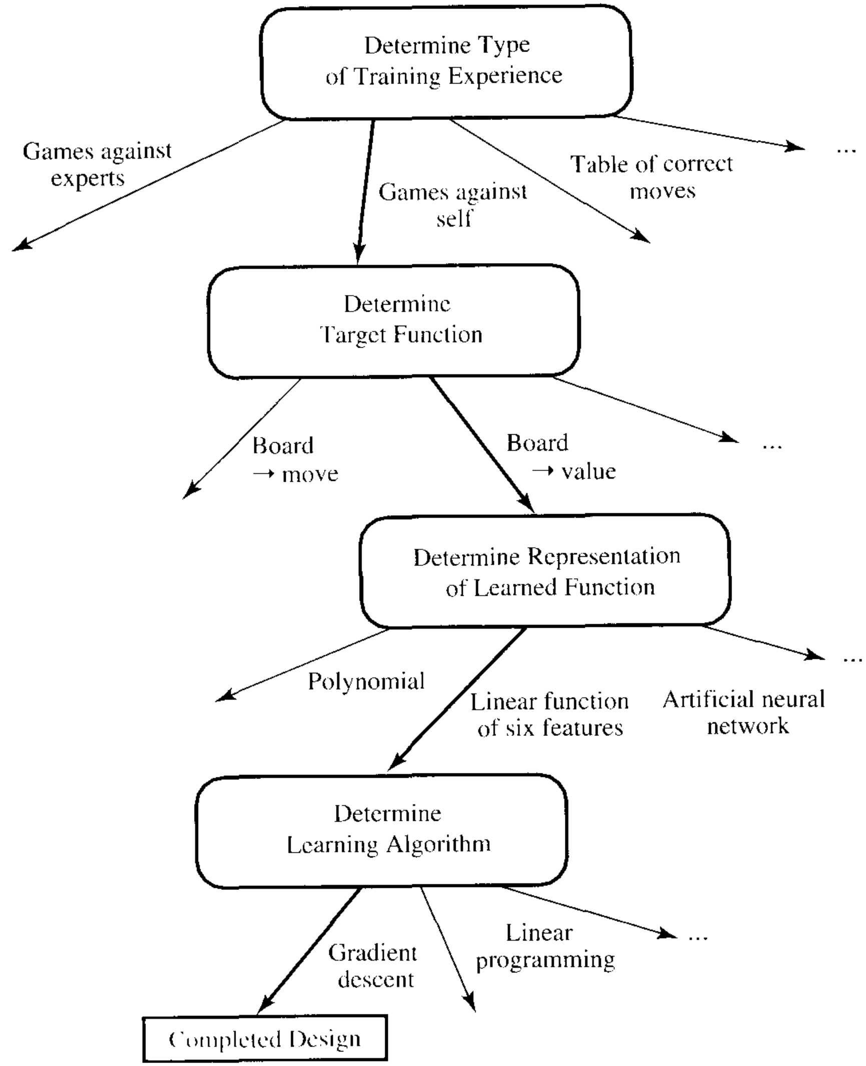
To get an intuitive understanding notice that

- when the error is 0 no weights are changed,
- when it is positive then each weight is increased in proportion to the value of its corresponding feature

Surprisingly, in certain settings this simple method can be proven to converge to the least squared approximation to V_{train}.

- In how many training instances?
- How understandable is the result? (Datamining)

Design Choices



Summary of Design Choices

- Constrained the learning task
- Single linear evaluation function
- Six specific board features

If the true function can be represented this way we are golden, otherwise sunk

Even if it can be represented, our learning algorithm might miss it!!!!

Very few guarantees but pretty good empirically (like Quicksort)

Our approach probably not good enough, but a similar approach worked for backgammon with a whole board representation and training on over 1 million games

Learning as Search

Search a very large space of possible hypothesis to find one that best fits the observed data

For example, hypothesis space consists of all evaluation functions that can be represented by some choice of values for w0…w6

The learner searches through this space to locate the hypothesis which is most consistent with the available training examples

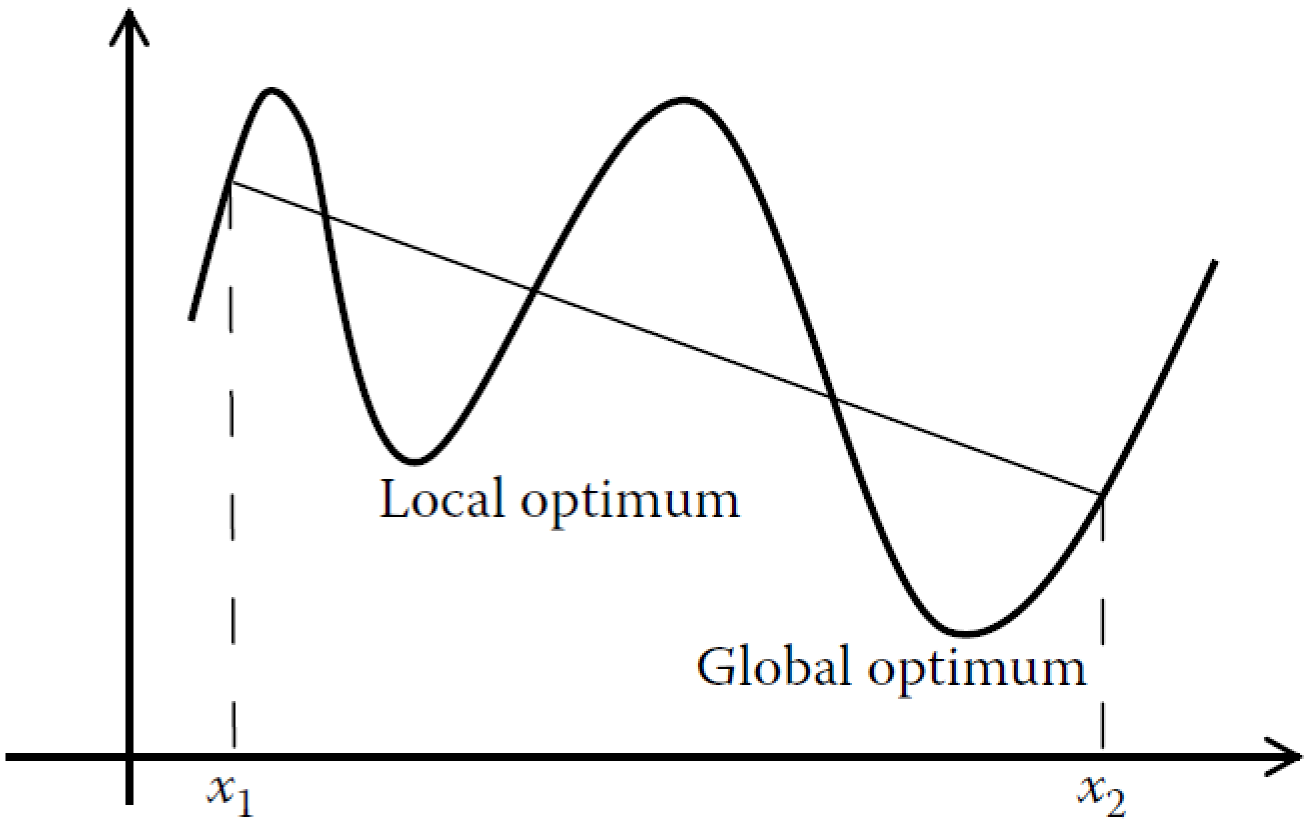
Choice of target function defines hypothesis space and therefore the algorithms which can be used.

As soon as space is small enough just test them all chess -> tic-tac-toe

What is the hypothesis Space?

- Space of vectors of 7 real numbers w_0 thru w_6

Global or Local Search?



Global or Local Search?

- Local search moves from the current state to another close state (does not save a path)

- Global search systematically searches the space from an “initial state” (saves a path)

Global or Local Search?

- So is LMS more like a global search or a local search?
- Global Search: depth-first, breadth-first, best-first search
- Local Search: hill-climbing, simulated annealing

Global or Local Search?

- So is LMS more like a global search or a local search?
- Global Search: depth-first, breadth-first, best-first search
- Local Search: hill-climbing, simulated annealing, LMS

III. Evaluation

- **Evaluation**
  - **Precision/Recall**
  - **Accuracy + weighted loss**
  - **ROC and AUC**

Confusion matrix

Actual\predicted	P	N
P	TP	FN
N	FP	TN

Precision and Recall

When evaluating a search tool or a classifier, we are interested in at least two performance measures:

Precision: Within a given set of positively-labeled results, the fraction that were true positives = TP/(TP + FP)

Recall: Given a set of positively-labeled results, the fraction of all positives that were retrieved = TP/(TP + FN)

Positively-labeled means judged "relevant" by the search engine or labeled as in the class by a classifier. TP = true positive, FP = false positive etc.

Be careful of “Accuracy”

The simplest measure of performance would be the fraction of items that are correctly classified, or the “accuracy” which is:

$$\frac{TP + TN}{TP + TN + FP + FN}$$

But this measure is dominated by the larger set (of positives or negatives) and favors trivial classifiers:

For example, we have a data set that has a distribution in which 95% of samples are negative and 5% of samples are positive. If 5 of a given test data set of 100 samples are positive and 95 samples are negative, then, even if all test results are classified as negative, the accuracy is 95%.

Weighted loss

We can instead try to minimize a weight sum:

$$w_1 \text{ fn} + w_2 \text{ fp}$$

And typically $w_1 \gg w_2$, since positives are often much rarer (clicks or purchases or viewing a movie).

The weighted "F" measure

A measure that naturally combines precision and recall is the β-weighted F-measure:

$$F = \frac{(\beta^2+1)PR}{\beta^2 P + R}$$

Which is the weighted harmonic mean of precision and recall. Setting $\beta = 1$ gives us the F_1 – measure. It can also be computed as:

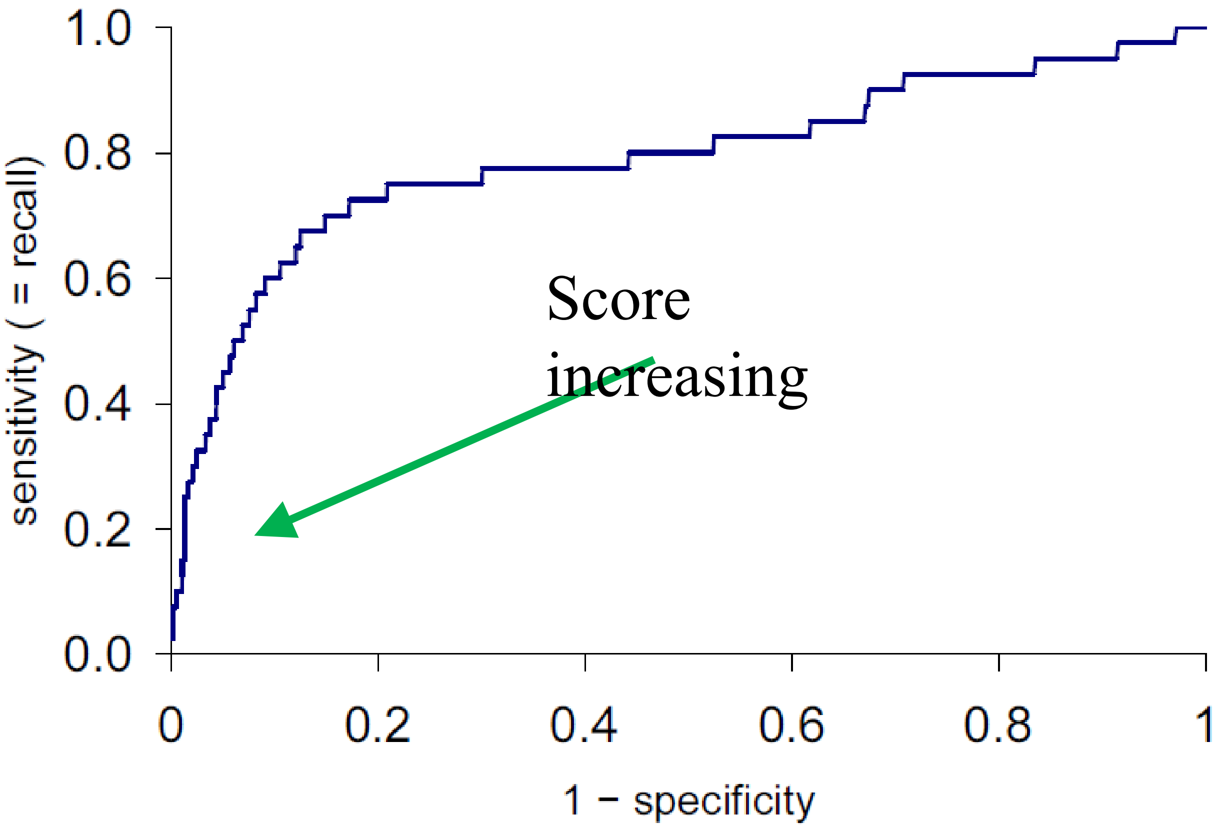
$$F_{\beta=1} = \frac{2PR}{P+R}$$

ROC plots

ROC is Receiver-Operating Characteristic. ROC plots

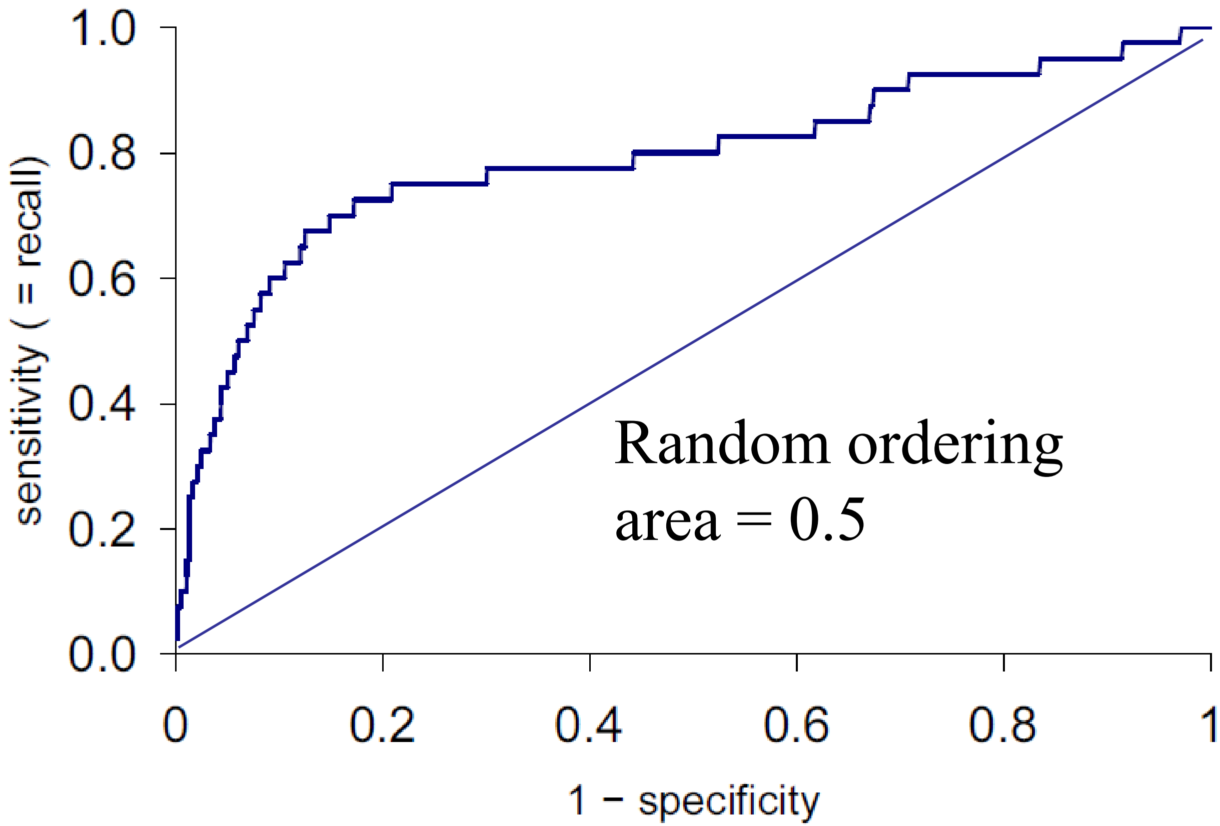
Y-axis: true positive rate = tp/(tp + fn), same as recall(sensitivity)

X-axis: false positive rate = fp/(fp + tn) = 1 – TNR(specificity)



ROC AUC

ROC AUC is the “Area Under the Curve” – a single number that captures the overall quality of the classifier. It should be between 0.5 (random classifier) and 1.0 (perfect).



Resampling

- When the data set is very large and all cases are well represented:
  - then no resampling method is needed, and
  - it is possible to use the holdout method where a portion of the data set is reserved for training while the rest of the data set is used for testing.
- Resampling is divided into two categories:
  - *simple resampling* (where each data point is used for testing only once) and
  - *multiple resampling* (which allows the use of the same data point more than once for testing)

K-fold cross-validation

Fig. 4.6 The *k*-fold cross-validation process

IV. Machine learning models

- Supervised & Unsupervised learning
  - Both families of methods can be applied to problems of classification (assigning observations to categories) or regression (predicting numerical properties of an observation).
- Reinforcement learning

Supervised vs. Unsupervised Learning

- Supervised learning (classification)
  - Supervision: The training data (observations, measurements, etc.) are accompanied by **labels** indicating the class of the observations
  - New data is classified based on the training set
- Unsupervised learning (clustering)
  - The class labels of training data is unknown
  - Given a set of measurements, observations, etc. with the aim of establishing the existence of classes or clusters in the data

V. Research Issues

What algorithms perform best for which type of problems and representations?

How much training data is sufficient?

How can prior knowledge be used?

How can you choose a useful next training experience?

How does noisy data influence accuracy?

How do you reduce a learning problem to a set of function approximations?

How can the learner automatically alter its representation to improve its ability to represent and learn the target function?

Summary

Machine Learning is useful for

- Datamining (credit worthiness)
- Poorly understood domains (face recognition)
- Programs that must dynamically adapt to changing conditions (Internet)

Summary II

Learning problem needs **well-specified** task, performing metric, and source of training experience.

Machine Learning approach involves a number of design choices:

- type of training experience,
- target function,
- representation of target function,
- an algorithm for learning the target function from the training data.

Summary III

Learning involves **searching** the space of possible hypothesis.

Different learning methods search different hypothesis spaces (numerical functions, neural networks, decision trees, symbolic rules).

There are some theoretical results which characterize conditions under which these search methods converge toward an optimal hypothesis.

Key to success

- Advances of algorithms
- Power of computing facilities
- Available of Data