1. Intro

05/10/2020

Machine learning is programming computers to improve a performance criterion using example data or past experience.
Machine learning (or data mining) is the task of producing knowledge from data.

1.1 Well-posed learning problems

Def: A computer program is said to learn from experience E with respect t some class of tasks T and performance measure P, if its performance at tasks in T, as measure by P, improves wit experience E

To have a well-defined problem we have to identify:

1. class of tasks
2. measure of performance
3. source of experience

-The chosen algorithm is the last thing to focus on

e.g.:

Task T: playing checkers

Performance measure P: percent of games won

Training experience E: playing practice games against itself

1.2 Designing a Learning system

1.2.1 Choosing the training experience

One key attribute is whether the training experience provides direct or indirect feedback regarding the choices made by the performance system.

- **Direct training:** individual checkers board states and the correct move for each. (easier)

- **Indirect training: c**onsists of the move sequences and final outcomes of various games played.

Problem of credit assignment: importance of a move in the entire game.

A **second important attribute of the training experience** is the degree to which the learner controls the sequence of training examples. For example, the learner might rely on the teacher to select informative board states and to provide the correct move for each. Alternatively, the learner might itself propose board states that it finds particularly confusing and ask the teacher for the correct move.

A **third important attribute of the training experience** is how well it represents the distribution of examples over which the final system performance P must be measured. In general, learning is most reliable when the training examples follow a distribution similar to that of future test examples.

e.g: If its training experience E consists only of games played against itself, there is an obvious danger that this training experience might not be fully representative of the distribution of situations over which it will later be tested.

To complete the design of the learning system, we must now choose:

1. the exact type of knowledge to be learned
2. a representation for this target knowledge
3. a learning mechanism

1.2.2 Choosing the target function

What type of knowledge will be learned and how this will be used by the performance program. In our case, the program just need to learn hiw ti chiise the best move:

- ChooseMove: B → M → output: best move
- V: B → R (Real numbers) → output: board states with higher scores:
    - value V(b) for an arbitrary board state b in B, as follows:
        1. if b is a final board state that is won, then V(b) = 100
        2. if b is a final board state that is lost, then V(b) = -100
        3. if b is a final board state that is drawn, then V(b) = 0

4. if b is a not a final state in the game, then V(b) = V(b'), where b' is the best final board state that can be achieved starting from b and playing optimally until the end of the game (assuming the opponent plays optimally, as well).

1.2.3 Choosing a Representation for the Target Function

V will be calculated as a linear combination of the following board features:

x1: the number of black pieces on the board
x2: the number of red pieces on the board
x3: the number of black kings on the board
x4: the number of red kings on the board

x5: the number of black pieces threatened by red (i.e., which can be captured on red's next turn)
X6: the number of red pieces threatened by black

Our elaborated learning task is now:
Task T: playing checkers
Performance measure P: percent of games won in the world tournament
Training experience E: games played against itself
Targetfunction: V:Board + 8
Targetfunction representation

1.2.4 Choosing a function approximation algorithm

We require a set of **training examples, each describing a specific board state *b* and the training value Vtrain(b) for b.**

In other words: a training example is an ordered pair of the form (b,Vtrain(b))

e.g.: the following describes a board state b in which black has won the game

(note x2 = 0 indicates that red has no remaining pieces) and for which the target function value Vtrain(b) is therefore +100.

1.2.4.1 Estimating training values

We require training examples that assign specific scores to specific board states.

Q: How to assign training values to the more numerous intermediate board states that occur before the game's end?

Assign the training value of Vtrain(b) for b to be V'(Successor(b)), where V' is the learner's current approximation to V and where Successor(b), this is:

Rule for estimating training values: Vtrain(b) ←V'(Successor (b))

we can see this will make sense if V' tends to be more accurate for board states closer to game's end.

1.2.4.2 Adjusting the weights

Last step: specify the learning algorithm for choosing the weights *Wi* to best fit the set of training examples {<b,Vtrain(b)>}.

Define the set of weights or **best hypothesis,** which minimizes the squared error between the training values and the values prediced by the hypothesis V'

We require an algorithm that will **incrementally refine the weights as new training examples become available** and that will be robust to errors in these estimated training values.

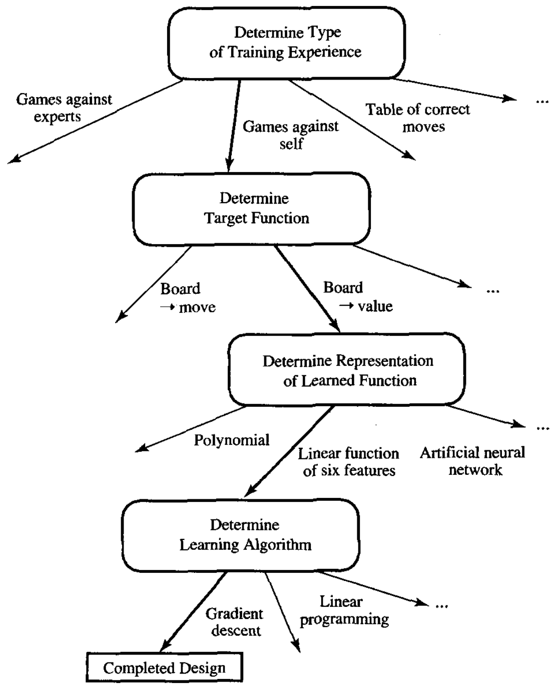
This algorithm can be viewed as performing a stochastic gradient-descent search through the space of possible hypotheses (weight values) to minimize the squared error E.
LMS (least mean square) weight update rule.
- Use the current weights to calculate V'(b)
- For each weight *wi*, update it as: *wi ← wi + n (*Vtrain(b) - V'(b)*) xi*

n~0.1 | *When Vtrain(b) - V'(b) = 0 no weights are changed.*

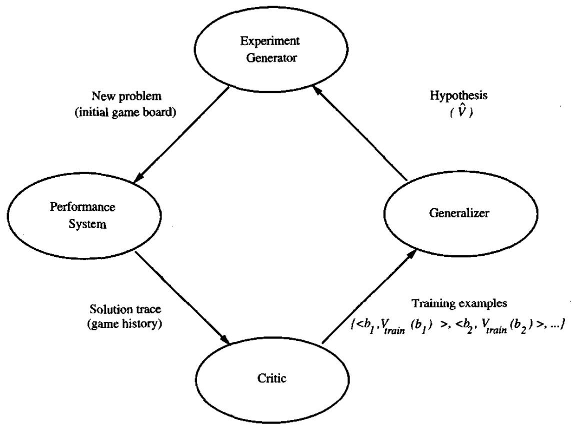
1.2.5 The final design

4 program modules:

1. Prformance system: must solve the given performance task
2. Critic: as input history of the game an produce a set of training examples
3. Generalizer: produce an hypothesis that estimate the target function.
4. Experiment Generator: produce a new problem, to maximize the learning rate of the overall system.



Determine Type of Training Experience
Games against experts
Games against self
Table of correct moves
...
Determine Target Function
Board → move
Board → value
...
Determine Representation of Learned Function
Polynomial
Linear function of six features
Artificial neural network
...
Determine Learning Algorithm
Gradient descent
Linear programming
...
Completed Design



Experiment Generator
New problem (initial game board)
Hypothesis ($\hat{V}$)
Performance System
Generalizer
Solution trace (game history)
Training examples
$\{<b_1, V_{train}(b_1)>, <b_2, V_{train}(b_2)>, ...\}$
Critic

1.3 Perspectives and issues in machine learning

One useful perspective on machine learning is that it involves searching a very large space of possible hypotheses to determine one that best fits. For example, consider the space of hypotheses that could in principle be output by the above checkers learner. This hypothesis space consists of all evaluation functions that can be represented by some choice of values for the weights w0 through w6. **The learner's task is thus to search through this vast space to locate the hypothesis that is most consistent with the available training examples.**

1.4 Machine Learning Problems

Learning a function f : X → Y , given a training set D containing sampled information about f.
Learning a function f means computing an approximated function ˆf that returns values as close as possible to f , specially for samples x not present in the training set D.
Xd = {x|x in D} in X and |Xd|<<|X|

1.5 Classification

Classification based on input

1.5.1 Supervised Learning

D = {<xi,yi>}

$$X \equiv \begin{cases} A_1 \times \ldots \times A_n,\ A_i \text{ finite sets } \textbf{(Discrete)} \\ \Re^n \ \textbf{(Continuous)} \end{cases}$$

$$Y \equiv \begin{cases} \Re^k \ \ \textbf{(Regression)} \\ \{C_1, \ldots, C_k\},\ \textbf{(Classification)} \end{cases}$$

Special case:
$X \equiv A_1 \times \ldots A_n$ (A_i finite) and $Y \equiv \{0, 1\}$ **(Concept Learning)**

1.5.2 Unsupervised Learning

D = {xi}

- Learning what normally happens
- No output available
- Clustering: grouping similar instances

1.5.3 Reinforcment Learning

D = {<Ai(1) , . . . , Ai(n), Ri>}

- Learning a policy: what to do in every state
- No supervised output available, only sparse and time-delayed rewards

1.6 Learning task

Given a training set D = {<Xi , c(Xi)>}, find the best approximation h*in H of the target function c : X → {0, 1}.
Steps:
1. Define the hypothesis space H (i.e., a representation of the hypotheses)
2. Define a performance metric to determine the best approximation
3. Define an appropriate algorithm

A hypothesis h is consistent with a set of training examples D of target concept c if and only if h(x) = c(x) for each training example hx, c(x)i in D.

Consistent(h,D) = (For every x in D) h(x) = c(x)

The real goal of a machine learning system is to find the best hypothesis h that predicts correct values of h(x') for instances x' not in D with respect to the unknown values c(x').

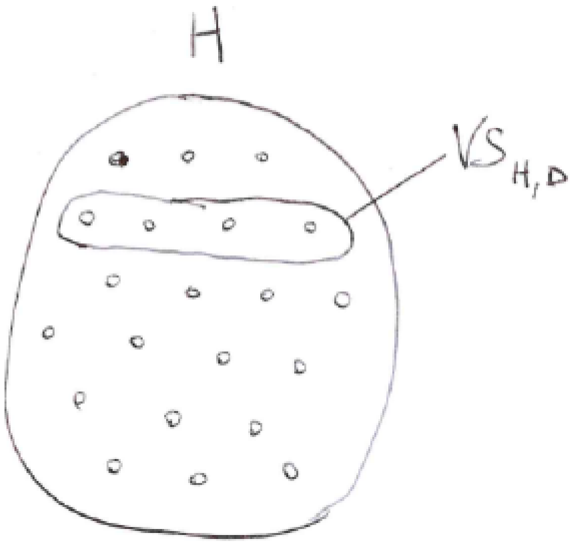
A performance measure is based on evaluating c(xi) = h(xi) for all xi in D.

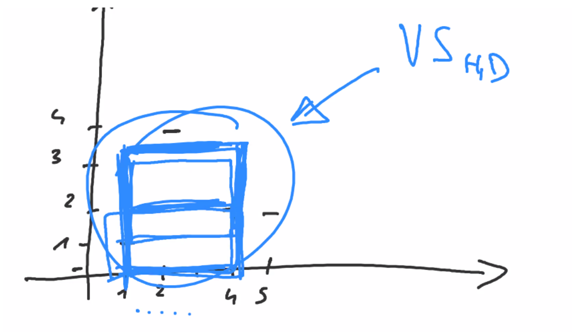
Inductive learning hypothesis: Any hypothesis that approximates the target function well over a sufficiently large set of training and unobserved examples.

1.6.1 Representation of hypothesis space

The version space, VSH,D, is the subset of hypotheses from H consistent with all training examples in D.

VSH,D {h in H| Consistent(h,D)}





1.6.2 LIST-THEN-ELIMINATE ALGORITHM

1. VS ← a list containing every hypothesis in H
2. For each training example, <x, c(x)>: remove from VS any hypothesis h for which h(x) ≠ c(x)

3. Output the list of hypotheses in VS

1.6.3 Output of a learning system

1. VSH',D (H' can represent all the subsets of X) → cannot predict instances not in D
2. VSH,D (H can not represent all the subsets of X) → limited prediction instances not in D
3. h* in VSH,D (h is one hypothesis chosen within the VS) → maximum prediction power on values not in D

In case of **noisy data set**, statistical methods must be used to implement robust algorithms and to avoid the possibility of having no consistent hypothesis.