

Intro. to machine learning (CSI 5325) Lecture 1: introduction

Greg Hamerly

Some content from Tom Mitchell.

- 1 About this class
- 2 What is machine learning?
- 3 Example applications
- 4 Learning and data
- 5 Why machine learning?
- 6 Specifying the learning task
- 7 Machine learning perspectives

Class stuff

Basics...

- textbook: Machine Learning by Mitchell
- syllabus: cs.baylor.edu/~hamerly/courses/5325_09s
- several assignments – writeups in \LaTeX
- midterm, final
- feel free to discuss with each other, but do your own work
- my office hours

Things that may change...

- topics covered and their order
- paper presentation format
- larger project topic

What is machine learning?

- building algorithms...
- which improve their performance...
- with experience.

Applications in speech recognition, planning, image analysis, vision, forecasting and prediction, etc.

Example: identifying handwritten digits



→ 0



→ 1



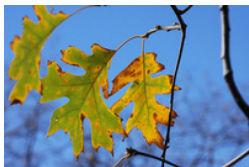
→ 2



→ 8

- Goal: post office wants to route mail quickly
- Input: grayscale picture (matrix of numbers?)
- Desired output: 0-9 (one of 10 classes), or 'DON'T KNOW'
- Available training data: labeled images

Example: finding faces



-



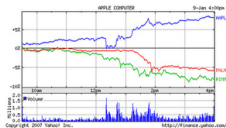
+



?

- Goal: identify if a picture contains a person's face
- Input: any image (matrix of numbers?)
- Desired output: there is/is not a face in the picture (+/-)
- Available training data: labeled images

Example: stock market predictions



- Goal: predict stock price changes
- Input:
 - yesterday's price (real number)
 - prices from last 5 days (vector of real numbers)
 - news articles (text)
 - ...
- Desired output:
 - stock will go up today (+/-)
 - stock value in 5 days (real number)
 - ...
- Available training data: inputs with corresponding outputs

Experience = data

Humans learn to improve their basketball skills through practice.

The machine learning goal is to write algorithms that will improve with experience.

- Experience is encoded as **examples**, or **data**.
- Data is usually cheaper than a programmer...
 - ...but it can be expensive!
- Autonomous learning must gain from experience.

Gathering training data

Where does training data come from?

- humans who gather and label data
- automatic collection

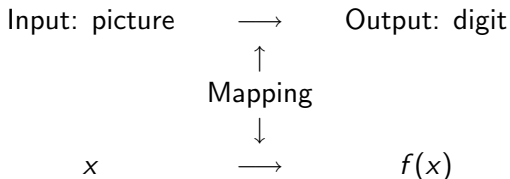
Where does machine learning come in?

Back to the handwritten digits:

Input: picture \longrightarrow Output: digit

Where does machine learning come in?

Back to the handwritten digits:



ML is learning the mapping from one concept to another, based on observed inputs and outputs.

Think of this as function approximation – learning f above.

- But f could be anything!
- So generally, we must assume some restrictions on f .

Why machine learning?

- lots of data
- lots of computational power
- human time is valuable
- industry interest

One view of three niches for machine learning:

- data mining: using historical data to improve decisions
 - medical records → medical knowledge
- software applications we can't program by hand
 - autonomous driving
 - speech recognition
- self-customizing software
 - newsreader that learns user interests

Relevant Disciplines

- Artificial intelligence
- Statistics and probability
- Bayesian statistics
- Computational complexity theory
- Control theory
- Information theory
- Optimization theory
- Philosophy
- Psychology and neurobiology
- ...

What is the learning problem?

Learning = improving with experience at some task

- improve over task T ,
- with respect to performance measure P ,
- based on experience E .

E.g., learn to play checkers

- T : play checkers
- P : % of games won in world tournament
- E : opportunity to play against self (!)

Learning to play checkers

- T : Play checkers
- P : Percent of games won in world tournament

- What experience?
- What exactly should be learned?
- How shall it be represented?
- What specific algorithm to learn it?

Type of Training Experience

- Direct or indirect?
 - for checkers, direct feedback might correspond to some reward/penalty after each move, while indirect feedback might be receiving a reward/penalty only at the end of the game
- Teacher or not?
 - in some tasks (probably not checkers), there might not be any relevant feedback

Something to consider: is training experience representative of performance goal?

Choose the Target Function

What target function should we choose for learning to play checkers?

- $ChooseMove : Board \rightarrow Move$
 - given a board, choose a move directly
- $V : Board \rightarrow \mathbb{R}$
 - given a board, assign the board a value
 - the player module chooses the next possible board with the highest valuation
- others?

For now, let's consider V .

Possible Definition for Target Function V

- if b is a final board state that is won, then $V(b) = 100$
- if b is a final board state that is lost, then $V(b) = -100$
- if b is a final board state that is drawn, then $V(b) = 0$
- if b is 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.

This gives correct values, but is not operational (efficiently computable)

Functions in learning

- $V(b)$: the true target function (fixed, but impractical)
- $\hat{V}(b)$: the learned function (what is being learned/modified with experience)
- $V_{train}(b)$: the training value (what we are learning from)

Comments...

- V is unavailable/intractable, so we use V_{train} .
 - What if V was available?
- V_{train} usually comes in the form of example data, rather than an explicit function.
- \hat{V} is what is being learned to approximate V .

Choose Representation for Target Function \hat{V}

We want to approximate V (the true optimal strategy) with a learned function \hat{V} .

How should we represent \hat{V} ?

- collection of rules?
- neural network?
- polynomial function of board features?
- ...

A representation for the learned function \hat{V}

Here's a linear function that assigns a value to a board state:

$$\hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

- $bp(b)$: number of black pieces on board b
- $rp(b)$: number of red pieces on b
- $bk(b)$: number of black kings on b
- $rk(b)$: number of red kings on b
- $bt(b)$: number of red pieces threatened by black (i.e., which can be taken on black's next turn)
- $rt(b)$: number of black pieces threatened by red

Learning \hat{V}

We've assumed that we can approximate V with:

$$\hat{V}(b) = w_0 + w_1 \cdot bp(b) + w_2 \cdot rp(b) + w_3 \cdot bk(b) + w_4 \cdot rk(b) + w_5 \cdot bt(b) + w_6 \cdot rt(b)$$

So the things that we want to learn are w_0, w_1, \dots, w_6 .

- we'll see how to do that in a minute.
- note that the function (e.g. $bp(b)$) values are fixed for any given board
- also, we're learning from only one player's perspective...

Obtaining Training Examples

Where can we get $V_{train}(b)$?

- we know the value of a finished game: +100, -100, or 0
- what about in the middle of a game?
- manually label the value of other board states (how?)
- use $V(b)$ to calculate the value of the state (impractical!)

Obtaining Training Examples

One rule for estimating training values of mid-game boards:

$$V_{train}(b) \leftarrow \hat{V}(Successor(b))$$

where $Successor(b)$ is the state before the same player makes the next move.

This is clearly not perfect, but will be appropriate for the end-game.

Our training data will potentially change over time, since we are estimating its value.

(In some ways, this example is a bit weird.)

Choose Weight Tuning Rule

Least mean squares (LMS) weight update rule:

Do repeatedly:

- Select a training example b at random

- 1 Compute $error(b)$:

$$error(b) = V_{train}(b) - \hat{V}(b)$$

- 2 For each board feature f_i , update weight w_i :

$$w_i \leftarrow w_i + \eta \cdot f_i \cdot error(b)$$

η is some small constant, say 0.1, to moderate the rate of learning

Whole learning system

- play checkers using \hat{V} → game history
- game history → training examples (board/value pairs)
- training examples → learned hypothesis \hat{V} (using LMS)
- hypothesis → new game (any valid starting board)
- go back to the beginning

Design choices

1. Type of training experience
 - games against experts
 - **games against self**
 - table of correct moves
 - ...
2. Target function
 - board \rightarrow move
 - **board \rightarrow value**
 - ...
3. Representation of learned function
 - polynomial
 - **linear function of six features**
 - artificial neural network
 - ...
4. Learning algorithm
 - **gradient descent**
 - linear programming
 - ...

A perspective on machine learning

Machine learning can be thought of as function approximation.

We have examples of some process, encoded as (input, output) pairs, and we want to train an algorithm to perform that process.

The task is then figuring out which is the best function to use for the task at hand.

A perspective on machine learning

Lots of questions to answer; no panacea:

- how should the training examples be formatted?
- what class of functions should you use?
- how should you pick a single function out of that class?
(i.e. how to train the model)
- how should you evaluate its performance?

Often we'll talk about a *hypothesis space* – which is really a broader term similar to 'class of functions.'

Some Issues in Machine Learning

- What algorithms can approximate functions well (and when)?
- How does number of training examples influence accuracy?
- How does complexity of hypothesis representation impact it?
- How does noisy data influence accuracy?
- What are the theoretical limits of learnability?
- How can prior knowledge of learner help?
- What clues can we get from biological learning systems?
- How can systems alter their own representations?