Introduction to Machine Learning

CS 586
Machine Learning

Prepared by Jugal Kalita

With help from Alpaydin’s *Introduction to Machine Learning* and Mitchell’s *Machine Learning*


**Algorithms**

- **Algorithm**: To solve a problem on a computer, we need an algorithm.

- An algorithm is a sequence of instructions that when carried out transforms input to output.

- **Example**: An algorithm for sorting; Input: a set of numbers; Output: An ordered list of the same numbers.

- **What if** we gave a program a number of examples of unsorted lists and corresponding sorted lists, and wanted the program to learn (or, come up with an algorithm) to sort?
Tasks with No Algorithms

• There are tasks for which we don’t have algorithms.
• *Example:* To distinguish between spam and legitimate emails.

• Issues:
  
  – In such a case, we usually have examples of spam emails and legitimate emails.
  – We cannot transform an input to an output in any sense.
  – We know we want a yes/no answer when we see an email, or a yes/no answer with a probability.
  – The nature of spam changes from time to time and from person to person.
Characteristics of Tasks with No Algorithms

- What we lack in algorithm or knowledge, we make up in data.
- We can collect thousands of example messages some of which we know to be spam and others that we know to be legitimate.
- **Task**: Our new task is to “learn” what differentiates spam from non-spam. In other words, we would like the computer to automatically extract the algorithm for this task.
Sources of Data and Their Analysis

- Supermarkets or chain stores generate gigabytes of data every day on customers, goods bought, total money spent, etc. They analyze data for advertising, marketing, ordering.
- Banks generate huge amounts of data on their customers. Banks analyze data to offer credit cards, analyze mortgage applicants, detect fraud.
- Telephone companies analyze call data for optimizing paths for calls, to maximize quality of service.
- Researchers or practitioners in healthcare informatics analyze large amounts of genomic data, proteomic data, microarray data, patient health records data.
• A company like Google analyzes content of emails and other documents one may have under various Google services to determine what advertisement to serve.

• *Do you have any datasets at work or you can think of, from which you would like to learn something?*
Machine Learning: An Example

• Consider consumer shopping behavior: We do not know what people buy when exactly, but we know it’s not random. E.g., they usually buy chips and soda together; they buy turkey before Thanksgiving.

• Certainly, there are patterns in the data. There is a process to how people decide what to buy when in what quantity.

• Our goal is to discover the process. We may not be able to do it exactly and completely.

• A more realistic goal is to construct a *good and useful approximation* of the process or the corresponding function.
Another Example of Machine Learning: Face Recognition

• This is a task we do effortlessly. Every day, we recognize family members, friends and others by looking at their faces or photographs, despite differences in pose, lighting, hair style, beard/no beard, makeup/no makeup, etc.

• We do it subconsciously and cannot explain how we do it. Because we can't explain how we do it, we can't write an algorithm.

• We also know that a face is not a random collection of pixels; a face has a structure; it is symmetric; it has pre-defined components: eyes, nose, mouth, located appropriately.
Another Example of Machine Learning: Face Recognition (continued...)

- Each person’s face is a pattern composed of a particular combination of these features: nose, eyes, mouth, eyelashes, hair, colors, etc.
- By analyzing sample face images of a person, a learning program captures the pattern specific to that person and uses it to recognize if a new real face or new image belongs to this specific person or not.
Uses of Machine Learning

- Machine Learning creates an *optimized* model of the concept being learned based on data or past experience. The model is *parameterized*.
- Learning is the execution of a computer program to optimize the parameter values so that the model fits data or past experience well.
- Uses of learning: Predictive and/or Descriptive.
- *Predictive*: Use the model to predict things about an unseen example.
- *Descriptive*: Use the model to describe the examples seen or experiences had. This model can be used in some problem-solving situation.
AI Perspective: Impact of Machine Learning on System Design

- Machine Learning is not just a database problem.
- It is also a part of artificial intelligence.
- To be intelligent, a system that is in a changing environment *should have the ability to learn*.
- If a system can learn and adapt to such changes, the system designer need not foresee and provide solutions for all possible situations.
- An example: Since the nature of spam keeps on changing, a spam detector should be able to adapt.
Machine Learning: Definition

- **Mitchell 1997**: A computer program $R$ 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.

- **Alpaydin 2010**: Machine learning is programming computers to optimize a performance criterion using example data or past experience.

- **E.g.**, a program that learns to play poker. It plays random games against itself, and when it loses, it analyzes the game and tries not to make the same mistakes.
Examples of Machine Learning Techniques and Applications

- Supervised Learning: Regression Analysis, Classification
- Unsupervised Learning: Association Learning, Clustering, Outlier finding
- Reinforcement Learning: Learning Actions
CLASSIFICATION (Supervised Learning)

• In a classification problem, we have to learn how to classify an unseen example into two or more classes based on examples we have seen earlier.

• *Binary Classification*: For example, we can learn how to classify an example into automobile or not automobile by observing a lot of examples. Babies learn how to classify a face as mom or non-mom.

• *Multi-class Classification*: We can learn how to classify an example into a sports car, a sedan, an SUV, a truck, a bus, etc. Babies learn how classify a face as mom, dad, brother, sister, grandpa, grandma, family, friend, unknown.
Algorithms for Classification

• Given a set of labeled data, learn how to classify unseen data into two or more classes.

• Many different algorithms have been used for classification. Here are some examples:
  
  – Decision Trees
  – Artificial Neural Networks
  – Kernel Methods such as Support Vector Machines
  – Bayesian classifiers
Example Training Dataset of Classification

- We need to find the boundary between the data representing classes: low-risk and high-risk for lending.
Essentially, we need to find a point $P(\theta_1, \theta_2)$ in 2-D to separate the two classes. We can consider $\theta_1, \theta_2$ to be two parameters we need to learn.
Applications of Classification

• Credit Scoring: Classify customers into high-risk and low-risk classes given amount of credit and customer information.

• Handwriting Recognition: Different handwriting styles, different writing instruments. 26 *2 = 52 classes for simple Roman alphabet.

• Determining car license plates for violation or for billing.

• Printed Character Recognition: OCR. Issues are fonts, spots or smudges, figures for OCR, weather conditions, occlusions, etc. Language models may be necessary.
Handwriting Recognition on a PDA

Judging by your completely uninformed opinion, I'm going to go ahead and venture to guess you've never actually USED any of these technologies for any extended period of time, if at all.

License plate Recognition

Taken from http://www.platerecognition.info/. This may be an image when a car enters a parking garage.
Applications of Classification (Continued)

- **Face Recognition:** Given an image of an individual, classify it into one of the people known. Each person is a class. Issues include poses, lighting conditions, occlusions, occlusion with glasses, makeup, beards, etc.

- **Medical Diagnosis:** The inputs are relevant information about the patient and the classes are the illnesses. Features include patient’s personal information, medical history, results of tests, etc.

- **Speech Recognition:** The input consists of sound waves and the classes are the words that can be spoken. Issues include accents, age, gender, etc. Language models may be necessary in addition to the acoustic input.
Face Recognition

Training examples of a person

Test images

Taken from http://www.uk.research.att.com.
Applications of Classification (Continued)

- Natural Language Processing: Parts-of-speech tagging, spam filtering, named entity recognition.
- Biometrics: Recognition or authentication of people using their physical and/or behavioral characteristics. Examples of characteristics: Images of face, iris and palm; signature, voice, gait, etc. Machine learning has been used for each of the modalities as well as to integrate information from different modalities.
POS tagging

Taken from

Named Entity Recognition

Taken from http://www.dcs.shef.ac.uk/hamish/IE/userrguide/ne.jpg.
• Regression analysis includes any techniques for modeling and analyzing several variables, when the focus is on the relationship between a dependent variable and one or more independent variables. Regression analysis helps us understand how the typical value of the dependent variable changes when any one of the independent variables is varied, while the other independent variables are held fixed.

• Regression can be linear, quadratic, higher polynomial, log-based and exponential functions, etc.
Regression

![Least Square Regression](http://plot.micw.eu/uploads/Main/regression.png)

*linreg(2, A=128.75, B=-1.03, r=0.92)*

*logreg(1, A=0.49, B=27.54, r=0.88)*

Taken from http://plot.micw.eu/uploads/Main/regression.png.
Applications of Regression

- Navigation of a mobile robot, an autonomous car: The output is the angle by which the steering wheel should be turned each time to advance without hitting obstacles and deviating from the route. The inputs are obtained from sensors on the car: video camera, GPS, etc. Training data is collected by monitoring the actions of a human driver.
UNSUPERVISED LEARNING

- In supervised learning, we learn a mapping from input to output by analyzing examples for which correct values are given by a supervisor or a teacher or a human being.
- Examples of supervised learning: Classification, regression that we have already seen.
- In unsupervised learning, there is no supervisor. We have the input data only. The aim is to find regularities in the input.
- Examples: Association Rule Learning, Clustering, Outlier mining
Learning Association (Unsupervised)

• This is also called (Market) Basket Analysis.

• If people who buy $X$ typically also buy $Y$, and if there is a customer who buys $X$ and does not buy $Y$, he or she is a potential customer for $Y$.

• Learn Association Rules: Learn a conditional probability of the form $P(Y \mid X)$ where $Y$ is the product we would like to condition on $X$, which is a product or a set of products the customer has already purchased.
Algorithms for Learning Association

• The challenge is how to find good associations fast when we have millions or even billions of records.

• Researchers have come up with many algorithms such as
  
  – Apriori: The best-known algorithm using breadth-first search strategy along with a strategy to generate candidates.
  – FP-growth: uses an extended prefix-tree to store the database in a compressed form. Uses a divide-and-conquer approach to decompose both the mining tasks and the databases.
Applications of Association Rule Mining

- Market-basket analysis helps in cross-selling products in a sales environment.

- **Web usage mining**: Items can be links on Web pages; Is a person more likely to click on a link having clicked on another link? We may download page corresponding to the next likely link to be clicked.

- If a customer has bought certain products on Amazon.com, what else could he/she be coaxed to buy?

- **Intrusion detection**: Each (or chosen) action is an item. If one action or a sequence of actions has been performed, is the person more likely to perform a few other instructions that may lead to intrusion?
• *Bioinformatics:* Find genes that are co-expressed for a disease to afflict a person.
Unsupervised Learning: Clustering

- Clustering: Given a set of data instances, organize them into a certain number of groups such that instances within a group are similar and instances in different groups are dissimilar.
- Bioinformatics: Clustering genes according to gene array expression data.
- Finance: Clustering stocks or mutual based on characteristics of company or companies involved.
- Document clustering: Cluster documents based on the words that are contained in them.
- Customer segmentation: Cluster customers based on demographic information, buying habits, credit
information, etc. Companies advertise differently to different customer segments. Outliers may form niche markets.
Clustering

Taken from a paper by Das, Bhattacharyya and Kalita, 2009.
Applications of Clustering (continued)

- Image Compression using color clustering: Pixels in the image are represented as RGB values. A clustering program groups pixels with similar colors in the same group; such groups correspond to colors occurring frequently. Colors in a cluster are represented by a single average color. We can decide how many clusters we want to obtain the level of compression we want.

- High level image compression: Find clusters in higher level objects such as textures, object shapes, whole object colors, etc.
• Finding outliers usually go with clustering, although the mathematical computations may be a bit different.
- Here $C_1$, $C_2$ and $C_3$ are clusters. $O_1$ and $O_2$ belong to singleton or very small clusters. Based on some additional computation, we may be able to call $O_1$ and $O_2$ outliers.
REINFORCEMENT LEARNING (A kind of supervised learning)

- In some applications, the output of the system is a sequence of *actions*.
- A single action is not important alone.
- What is important is the *policy* or the sequence of correct actions to reach the goal.
- In reinforcement learning, reward or punishment comes usually at the very end or infrequent intervals.
- The machine learning program should be able to assess the goodness of “policies”; learn from past good action sequences to generate a “policy”.
Reinforcement Learning...

- Taken from *Artificial Intelligence by Russell and Norvig*.

- The workspace over which agent is trying to learn what to do is a 3x4 grid. In any cell, the agent can perform at most 4 actions (up, down, left, right). The agent gets a reward of +1 in cell (4,3) and -1 in cell (4,2). We need to find the best course of action for the agent in any cell.
Reinforcement Learning...

- The agent has learned a “policy”, which tells the agent what action to perform in what state.
The problem can be reduced to learning a value for each of the states. In a particular cell, go to the neighboring cell with the highest value.
Applications of Reinforcement Learning

• Game Playing: Games usually have simple rules and environments although the game space is usually very large. A single move is not of paramount importance; a sequence of good moves is needed. We need to learn good game playing policy.

• Example: Playing world class backgammon or checkers, Playing soccer.
Applications of Reinforcement Learning (continued)

• Imagine we have a robot that cleans the engineering building at night.

• It knows how to perform actions such as

  – charge itself
  – open a room’s door
  – look for the trash can in a room
  – pick up trash can
  – empty trash can
  – lock a door,
  – go up on the elevator, etc.
• Imagine also that each floor has several charging stations in various locations.

• At any time, the robot can move in many in one of a number of directions, or perform one of several actions.

• After a number of trial runs (not by solving a bunch of complex equations it sets up in the beginning, with complete knowledge of the environment), it should learn the correct sequence of actions to pick up most amount of trash, making sure it is charged all the time.

• If it is taken to another building, it should be able to learn an optimized sequence of actions for that building.
• If one of the charging stations is not working, it should be able to adapt and find a new sequence of actions quickly.
Relevant Disciplines

- Artificial intelligence
- Computational complexity theory
- Control theory
- Information theory
- Philosophy
- Psychology and neurobiology
- Statistics
- Bayesian methods
- ...
What is the Learning Problem?: A Specific Example in Details

• Reiterating our definition: Learning = Improving with experience at some task
  
  – Improve at task $T$
  – with respect to performance measure $P$
  – based on experience $E$.

• Example: Learn to play checkers
  
  – $T$: Play checkers
  – $P$: % of games won in world tournament
  – $E$: opportunity to play against self
Checkers

• 64 squares on board. 12 checkers for each player.
• Flip a coin to determine black or white.
• Use only black squares.
• Move forward one space diagonally and forward. No landing on an occupied square.

Taken from http://www.learnplaywin.net/checkers/checkers-rules.htm.
Checkers: Standard rules

- Players alternate turns, making one move per turn.
- A checker reaching last row of board is "crowned". A king moves the same way as a regular checker, except he can move forward or backward.
- One must jump if its possible. Jumping over opponent’s checker removes it from board. Continue jumping if possible as part of the same turn.
- You can jump and capture a king the same way as you jump and capture a regular checker.
- A player wins the game when all of opponent’s checkers are captured, or when opponent is completely blocked.
Steps in Designing a Learning System

• Choosing the Training Experience
• Choosing the Target Function: What should be learned?
• Choosing a Representation for the Target Function
• Choosing a Learning Algorithm
Type of Training Experience

- **Direct or Indirect?**
  - *Direct*: Individual board states and correct move for each board state are given.
  - *Indirect*: Move sequences for a game and the final result (win, loss or draw) are given for a number of games. How to assign credit or blame to individual moves is the *credit assignment* problem.
Type of Training Experience (continued)

- **Teacher or Not?**
  - Supervised: Teacher provides examples of board states and correct move for each.
  - Unsupervised: Learner generates random games and plays against itself with no teacher involvement.
  - Semi-supervised: Learner generates game states and asks the teacher for help in finding the correct move if the board state is difficult or confusing.
Type of Training Experience (continued)

- Is the training experience good?
  - Do the training examples represent the distribution of examples over which the final system performance will be measured?
  - Performance is best when training examples and test examples are from the same/a similar distribution.
  - Our checker player learns by playing against oneself. Its experience is indirect. It may not encounter moves that are common in human expert play.
Choose the Target Function

- Assume that we have written a program that can generate all legal moves from a board position.

- We need to find a target function \( \text{ChooseMove} \) that will help us choose the best move among alternatives. This is the learning task.

- \( \text{ChooseMove} : \text{Board} \rightarrow \text{Move} \). Given a board position, find the best move. Such a function is difficult to learn from indirect experience.

- Alternatively, we want to learn \( V : \text{Board} \rightarrow \mathbb{R} \). Given a board position, learn a numeric score for it such that higher score means a better board position. Our goal is to learn \( V \).
A Possible (Ideal) Definition for Target Function

- 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 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.

This gives correct values, but is not operational because it is not efficiently computable since it requires searching till the end of the game. We need an operational definition of \( V \).
Choose Representation for Target Function

We need to choose a way to represent the ideal target function in a program.

• A table specifying values for each possible board state?
• collection of rules?
• neural network?
• polynomial function of board features?
• ...

We use $\hat{V}$ to represent the actual function our program will learn. We distinguish $\hat{V}$ from the ideal target function $V$. $\hat{V}$ is a function approximation for $V$. 
A Representation for Learned Function

\[ \hat{\mathbf{V}} = w_0 + w_1 \cdot x_1(b) + w_2 \cdot x_2(b) + w_3 \cdot x_3(b) + w_4 \cdot x_4(b) + w_5 \cdot x_5(b) + w_6 \cdot x_6(b) \]

- \( x_1(b) \): number of black pieces on board \( b \)
- \( x_2(b) \): number of red pieces on \( b \)
- \( x_3(b) \): number of black kings on \( b \)
- \( x_4(b) \): number of red kings on \( b \)
- \( x_5(b) \): number of red pieces threatened by black (i.e., which can be taken on black’s next turn)
- \( x_6(b) \): number of black pieces threatened by red

It is a simple equation. Note a more complex representation requires more training experience to learn.
Specification of the Machine Learning Problem at this time

- **Task** $T$: Play checkers
- **Performance Measure** $P$: % of games won in world tournament
- **Training Experience** $E$: opportunity to play against self
- **Target Function**: $V : Board \rightarrow \mathbb{R}$
- **Target Function Representation**:

$$\hat{V} = w_0 + w_1 \cdot x_1(b) + w_2 \cdot x_2(b) + w_3 \cdot x_3(b) + w_4 \cdot x_4(b)$$
$$+ w_5 \cdot x_5(b) + w_6 \cdot x_6(b)$$

The last two items are design choices regarding how to implement the learning program. The first three specify the learning problem.
Generating Training Data

• To train our learning program, we need a set of training data, each describing a specific board state $b$ and the training value $V_{\text{train}}(b)$ for $b$.
• Each training example is an ordered pair $\langle b, V_{\text{train}}(b) \rangle$.
• For example, a training example may be

$$\langle \langle x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0 \rangle, +100 \rangle$$

This is an example where black has won the game since $x_2 = 0$ or red has no remaining pieces.

• However, such clean values of $V_{\text{train}}(b)$ can be obtained only for board value $b$ that are clear win, loss or draw.

• For other board values, we have to estimate the value of $V_{\text{train}}(b)$.
Generating Training Data (continued)

- According to our set-up, the player learns indirectly by playing against itself and getting a result at the very end of a game: win, loss or draw.
- Board values at the end of the game can be assigned values. How do we assign values to the numerous intermediate board states before the game ends?
- A win or loss at the end does not mean that every board state along the path of the game is necessarily good or bad.
- However, a very simple formulation for assigning values to board states works under certain situations.
Generating Training Data (continued)

- The approach is to assign the training value $V_{\text{train}}(b)$ for any intermediate board state $b$ to be $\hat{V}(\text{Successor}(b))$ where $\hat{V}$ is the learner’s current approximation to $V$ (i.e., it uses the current weights $w_i$) and $\text{Successor}(b)$ is the next board state for which it’s again the program’s turn to move.

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

- It may look a bit strange that we use the current version of $\hat{V}$ to estimate training values to refine the very same function, but note that we use the value of $\text{Successor}(b)$ to estimate the value of $b$. 
Training the Learner: Choose Weight Training Rule

LMS Weight update rule:

For each training example \( b \) do
   Compute \( \text{error}(b) \):
      \[
      \text{error}(b) = V_{\text{train}}(b) - \hat{V}(b)
      \]
   For each board feature \( x_i \), update weight \( w_i \):
      \[
      w_i \leftarrow w_i + \eta \cdot x_i \cdot \text{error}(b)
      \]
   Endfor
Endfor

Here \( \eta \) is a small constant, say 0.1, to moderate the rate of learning.
Checkers Design Choices

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

Taken from Page 13, Machine Learning by Tom Mitchell, 1997.
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?