Learning in Intelligent Systems

Artificial Intelligence @ Allegheny College

Janyl Jumadinova

September 25, 2018
Overview of Learning
Learning in Humans

- The act / process of acquiring, modify or reinforcing knowledge or skills through synthesizing different types of new or existed information.
Learning in Humans

- The act / process of acquiring, modify or reinforcing knowledge or skills through synthesizing different types of new or existed information.

- Key to human survival.
Learning in Humans

- The act / process of acquiring, modify or reinforcing knowledge or skills through synthesizing different types of new or existed information.
- Key to human survival.
- Progress over time tends to follow learning curves (relatively permanent).
Computational methods using “experience” to improve performance.
Learning in Computing Systems

- Computational methods using “experience” to improve performance.
- Experience — data driven task
Computational methods using “experience” to improve performance.

Experience — data driven task

Computer science — involves learning algorithms, analysis of complexity, and theoretical guarantees.
Artificial intelligence | Machine learning
Artificial intelligence | Machine learning

- Computer program(s) with adaptive mechanisms that enable computer / machine to learn from experience / example / analogy / rewards.
Artificial intelligence | Machine learning

- Computer program(s) with adaptive mechanisms that enable computer / machine to learn from experience / example / analogy / rewards.
- It improves the performance of an intelligent system over time (e.g., reducing error rate, improving rewards).
Why Learning in Computing Systems?

- Understand and improve efficiency of human learning / understanding.
Why Learning in Computing Systems?

- Understand and improve efficiency of human learning / understanding.
- Discover new things or structure that is unknown to humans.
Why Learning in Computing Systems?

- Understand and improve efficiency of human learning / understanding.
- Discover new things or structure that is unknown to humans.
- Fill in skeletal or incomplete knowledge / expert specifications about a domain.
Applications of Learning

Mainly in decision making / pattern recognition / intelligent systems.
Applications of Learning

Mainly in decision making / pattern recognition / intelligent systems.

- Robot navigation.
- Automatic speech recognition (Siri in iPhone, Google speech-to-text search).
- Search and recommendation (Google, Amazon, eBay).
- Financial prediction, fraud detection, medical diagnosis.
- Video games, data visualization.
Black-box Learning

Experiences/Data

Problem/Task

Background knowledge/Bias

Answer/Performance
Learning Architecture

Experiences/ Data

Problem/ Task

Background knowledge/ Bias

Learner

Reasoner

Model(s)

Answer/ Performance
Learning Paradigms

- **Supervised learning**
  - input-output relationships
Learning Paradigms

- **Supervised learning**
  - input-output relationships

- **Unsupervised learning**
  - relationship among inputs
Learning Paradigms

- **Supervised learning**
  - input-output relationships

- **Unsupervised learning**
  - relationship among inputs

- **Reinforcement learning**
  - input-action relates to rewards / punishment
Learning Paradigms

- **Supervised learning**
  - input-output relationships

- **Unsupervised learning**
  - relationship among inputs

- **Reinforcement learning**
  - input-action relates to rewards / punishment

- **Rule learning**
  - discovering common relationship to develop rules
Supervised Learning

Given examples of inputs and corresponding desired outputs.
Supervised Learning

Given examples of inputs and corresponding desired outputs.

Tasks:
- **Classification** (categorizing output: correct class)
- **Regression** (continuous output to predict output based for new inputs)
- **Prediction** (classify / regression on new input sequences)
Supervised Learning
Unsupervised Learning

Given only inputs and automatically discover representations, features, structure etc.
Unsupervised Learning

Given only inputs and automatically discover representations, features, structure etc.

**Tasks:**

- **Clustering** (to group similar data into a finite number of clusters / groups)
- **Vector Quantization** (compress / decode dataset into a new representation but maintaining internal information)
- **Outlier Detection** (select highly unusual cases)
Unsupervised Learning

Diagram:
- Training Text, Documents, Images, etc.
- Feature Vectors
- Machine Learning Algorithm
- New Text, Document, Image, etc.
- Feature Vector
- Predictive Model
- Likelihood or Cluster ID or Better Representation
Reinforcement Learning

- Learning approach of getting a computer system to act in the world so as to maximize its rewards.
Reinforcement Learning

- Learning approach of getting a computer system to act in the world so as to maximize its rewards.
- Consider teaching a domestic animal. We cannot tell it what to do, but we can reward / punish if it does the right/ wrong thing.
Reinforcement Learning

- Learning approach of getting a computer system to act in the world so as to maximize its rewards.
- Consider teaching a domestic animal. We cannot tell it what to do, but we can reward / punish if it does the right/ wrong thing.
- Process to determine what it did that made it get the reward / punishment – “credit assignment problem.”
Reinforcement Learning
Rule Learning

Given multiple measurements to discover very common settings in term of causal-effect.
Rule Learning

Given multiple measurements to discover very common settings in term of causal-effect.

**Tasks:**

- Association rules (to group similar data into a finite number of clusters / groups)
Rule Learning

Given multiple measurements to discover very common settings in term of causal-effect.

**Tasks:**

- Association rules (to group similar data into a finite number of clusters / groups)
- Classification rules (compress / decode dataset into a new representation but maintaining internal information)
Rule Learning
Learning Paradigms and Some Techniques

- **Supervised**
  - Neural networks (backpropagation)
  - Support vector machine
  - Decision tree
  - K-nearest neighbour
  - Case-based reasoning

- **Unsupervised**
  - Neural networks (self-organizing map)
  - Hidden markov model
  - K-Means
  - Principle component analysis

- **Reinforcement**
  - Q-learning
  - Tabular lambda (TD)
  - Tabular Q-learning
  - Sarsa (lambda)

- **Rule**
  - Apriori algorithm
  - Equivalence class transformation
  - Frequent pattern (FP) – growth algorithm
  - Generalized sequential pattern (GSP) algorithm
Goal:
Find an object of a pre-defined class in a static image or video frame.
Object/Face Detection/Recognition

Goal:
Find an object of a pre-defined class in a static image or video frame.

Approach:
- Extract certain image features, such as edges, color regions, textures, contours, etc.
- Use some heuristics to find configurations and/or combinations of those features specific to the object of interest.
Training Set (Positive Samples/Negative Samples)

Different features are extracted from the training samples and distinctive features that can be used to classify the object are selected.
Statistical Model Training

- Training Set (Positive Samples/Negative Samples)
- Different features are extracted from the training samples and distinctive features that can be used to classify the object are selected.
- Each time the trained classifier does not detect an object (misses the object) or mistakenly detects the absent object (gives a false alarm), model is adjusted.
Decision Tree

1.65% of data
536 total instances

Decision path for income relationship?

equals Husband

education?

equals Bachelors

occupation?

equals Prof-specialty

age?

>= 28.72

predict: >50K
Decision Tree

- **Root node**
  - Entry point to a collection of data

- **Inner nodes (among which the root node)**
  - A question is asked about data
  - One child node per possible answer

- **Leaf nodes**
  - Correspond to the decision to take (or conclusion to make) if reached
Decision Tree

- Represented by a series of binary splits.
- Each internal node represents a value query on one of the variables e.g. “Is $X_3 > 0.4$”. If the answer is “Yes”, go right, else go left.
Decision Tree

- Represented by a series of binary splits.
- Each internal node represents a value query on one of the variables e.g. “Is $X_3 > 0.4$”. If the answer is “Yes”, go right, else go left.
- The terminal nodes are the decision nodes.
- New observations are classified by passing their $X$ down to a terminal node of the tree, and then using majority vote.
Decision Tree

- Can handle huge datasets
- Can handle mixed predictors—quantitative and qualitative
- Easily ignore redundant variables
- Handle missing data elegantly
- Small trees are easy to interpret
- Large trees are hard to interpret
- Often prediction performance is poor
Model Averaging

Classification trees can be simple, but often produce noisy and weak classifiers.

- **Bagging**: Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.
- **Boosting**: Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.
- **Random Forests**: Fancier version of bagging.
Model Averaging

Classification trees can be simple, but often produce noisy and weak classifiers.

- **Bagging**: Fit many large trees to bootstrap-resampled versions of the training data, and classify by majority vote.
- **Boosting**: Fit many large or small trees to reweighted versions of the training data. Classify by weighted majority vote.
- **Random Forests**: Fancier version of bagging.

In general,

\[
\text{Boosting} \succ \text{Random Forests} \succ \text{Bagging} \succ \text{Single Tree}.
\]
Weak Classifier

- Computed feature value is used as input to a very simple decision tree classifier with 2 terminal nodes

\[
\begin{cases}
1 & x_i \geq t_i \\
-1 & x_i \leq t_i
\end{cases}
\]
Boosted Classifier

- Complex and robust classifier is built out of multiple weak classifiers using a procedure called boosting.
- The boosted classifier is built iteratively as a weighted sum of weak classifiers.
Boosted Classifier

- Complex and robust classifier is built out of multiple weak classifiers using a procedure called boosting.
- The boosted classifier is built iteratively as a weighted sum of weak classifiers.
- On each iteration, a new weak classifier $f_i$ is trained and added to the sum.
Boosted Classifier

- Complex and robust classifier is built out of multiple weak classifiers using a procedure called boosting.
- The boosted classifier is built iteratively as a weighted sum of weak classifiers.
- On each iteration, a new weak classifier $f_i$ is trained and added to the sum.
- The smaller the error $f_i$ gives on the training set, the larger is the coefficient/weight that is assigned to it.
Cascade of Boosted Classifiers

- Sequence of boosted classifiers with constantly increasing complexity.
- Chained into a cascade with the simpler classifiers going first.
OpenCV: Cascade Classifier

- Uses simple features and a cascade of boosted tree classifiers as a statistical model.
OpenCV: Cascade Classifier

- Classifier is trained on image of fixed size (Viola uses 24x24)
- Detection is done by sliding a search window of that size through the image and checking whether an image region at a certain location looks like our object or not.
Feature’s value is a weighted sum of two components:
- Pixel sum over the black rectangle
- Sum over the whole feature area
Cascade Classifier

- Instead of applying all the 6000 features on a window, group the features into different stages of classifiers and apply one-by-one.
- If a window fails the first stage, discard it. We don’t consider remaining features on it.
- If it passes, apply the second stage of features and continue the process. The window which passes all stages is a face region.
OpenCV: Cascade Classifier

OpenCV already contains many pre-trained classifiers for face, eyes, smile etc. Those XML files are stored in opencv/data/haarcascades/
cv2.CascadeClassifier.detectMultiScale(image[, scaleFactor[, minNeighbors[, flags[, minSize[, maxSize]]]]]])

- **scaleFactor** : Parameter specifying how much the image size is reduced at each image scale.
- **minNeighbors** : Parameter specifying how many neighbors each candidate rectangle should have to retain it. This parameter will affect the quality of the detected objects: higher value results in less detections but higher quality.