Lecture 16
Machine Learning Intro

CS 520: Intro AI
Jingjin Yu | Rutgers
Why do we approximate

- Exact inference is intractable
- Satisfiability \( \leq \) propositional logic inference \( \leq \) exact inference on Bayes net

Methods for doing approx. inference

- Simplest: sample the joint distribution and count, e.g.,
  - To compute \( P(e|j) \)
  - Sample from \( P(B), P(E) \), then \( P(A|B,E) \), then \( P(J|A) \)
  - Count \( \hat{P}(e|j) = \frac{\# \text{events with } e,j}{\# \text{events with } j} \)

- Rejection sampling: discard irrelevant samples
  - E.g., events with outcome \( \neg j \) can be discarded for computing \( P(e|j) \)

- Likelihood weighting
  - Combines sampling (non-evidence variable) and weighting (evidence variable)

- Markov chain Monte Carlo (MCMC)
  - Start with a random assignment of values to non-evidence variables
  - In each iteration, sample non-evidence variables one by one using its Markov blanket
  - Count


Review: Markov Networks

Bayesian networks and Markov networks are both \textbf{graphical models}.

\textbf{Markov networks} model \textbf{correlation} on \textbf{undirected graphs}.

- Cliques and factor potentials

- Joint probability: product of factor potentials
  \[ P(X_1, \ldots, X_n) = \frac{1}{Z} \prod_{i=1}^{m} \phi_i(C_i) \]

- In associative Markov network (only 1- and 2-cliques),
  \[ P(X) = \frac{1}{Z} \exp \left( \sum_{X_i} \sum_k \text{sign}(X_i) w_{node}[k] f_i[k] + \sum_{(X_i, X_j) \text{ s.t. } X_i = X_j \text{ and } X_i, X_j \text{ adjacent}} \sum_k w_{edge}[k] f_{i,j}[k] \right) \]

- Data give us $f_i, f_{i,j}$ → compute $w_{node}$ and $w_{edge}$ that maximize $P(X)$

- Make predictions over new data

---

Graphical representation of Markov networks with nodes $X_1$ to $X_{21}$. Each node represents a variable in the network, and the edges between nodes represent the dependencies or correlations between the variables.
This Lecture

⇒ Introduction to machine learning
  ⇒ Definition and scope of machine learning
  ⇒ Application scenarios
  ⇒ Typical approaches

⇒ Decision tree

⇒ Linear classifiers (e.g., perceptron)
What does Machine Learning do?

Arthur Samuel: a field of study that gives computers the ability to learn without being explicitly programmed.

Tom M Mitchell: 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 $E$.

- Emphasizes the **operational** perspective
- Not the **cognitive** perspective (i.e., can machines think)
  - We do not know how we think, so it is premature to study whether machines can think
General (Supervised Learning) Pipeline

Learning phase

Training data → Feature extraction → Features → Training → Learned model → Prediction

Inference phase

Test data → Feature extraction → Prediction
Applications: Spam Filter

Dear Sir,

I know that this mail will come to you as a surprise as we have never met before, but need not to worry as I am contacting you independently of my investigation and no one is informed of this communication. I need your urgent Cooperation in transferring the sum of $11.3 Million dollars immediately ...

Hello Professor Yu:

I am a CS junior at Rutgers and found your robotics related research very interesting. In particular, I am intrigued by the application of the ludicrous learning model toward robotic dominance in your paper “How Robot Can Take Over the World”. I wonder whether you have projects in your lab on this that I can help with during the summer ...

⇒ As mentioned, we can use naïve Bayes for this purpose
⇒ What is the data? Feature?
Applications: Playing Competitive Games

DeepBlue defeated Kasparov, 1997

Watson winning Jeopardy, 2011

AlphaGo defeated Lee Sedol, 2016

Sophisticated techniques, with deep learning showing advantage
Applications: Voice Recognition & Assistant

Again, deep learning is taking over here...
Applications: Robot Grasping

L. Pinto and A. Gupta, Supersizing self-supervision: Learning to grasp from 50K tries and 700 robot hours, arXiv.org/abs/1509.06825

⇒ Some more deep learning...
Applications: Distilling Natural Laws

Michael Schmidt and Hod Lipson, Distilling Free-Form Natural Laws from Experimental Data, Science, 2009

⇒ In this case, evolutionary method (e.g., genetic algorithms)
Supervised Learning

Learning: learn $f$ based on a training set $(X_1, y_1), (X_2, y_2), \ldots, (X_N, y_N)$

Inference: given new data $X$, compute $f(X)$
Unsupervised Learning

⇒ No labels!
⇒ The goal is to learn hidden structures from data
⇒ E.g., clustering, dimension reduction, ...
Generative and Discriminative Models

- A generative model can simulate actual data.
- A discriminative model does not have enough information to reproduce expected data.

\[ N(x_1, \sigma_1^2) \quad N(x_2, \sigma_2^2) \]

Why not in this case?

Naïve Bayes?
Approaches (an Incomplete List)

- Naïve Bayes
- Bayes net
- Markov network
- Decision tree
- Linear classifier
- SVM
- Clustering
- Dimension reduction
- Reinforcement learning
- Artificial neural network
- Deep learning / DeepNet
- Genetic algorithms
Decision Tree

- Decide the outcome based on a set of attributes.
- E.g., whether to wait for table given *patrons, waitEstimate, ...*

```
Patrons?
  None  Some  Full
  No    Yes  WaitEstimate?
    >60  30-60  10-30
    No  Alternate?  Yes
        No  Fri/Sat?  Yes
        Yes  Reservation?  No
            Bar?  yes  No
                No  Yes
                    No  Yes
            No  Yes
                Yes  Raining?
```
Decision Tree: the Algorithm

```
function DECISION-TREE-LEARNING(examples, attributes, parent_examples) returns a tree

    if examples is empty then return PLURALITY-VALUE(parent_examples)
    else if all examples have the same classification then return the classification
    else if attributes is empty then return PLURALITY-VALUE(examples)
    else
        \[ A \leftarrow \arg\max_{a \in \text{attributes}} \text{IMPORTANCE}(a, \text{examples}) \]
        tree \leftarrow a new decision tree with root test \( A \)
        for each value \( v_k \) of \( A \) do
            \[ \text{exs} \leftarrow \{ e : e \in \text{examples} \text{ and } e.A = v_k \} \]
            \[ \text{subtree} \leftarrow \text{DECISION-TREE-LEARNING}(	ext{exs}, \text{attributes} - A, \text{examples}) \]
            add a branch to \( \text{tree} \) with label \( (A = v_k) \) and subtree \( \text{subtree} \)
        return tree
```

⇒ The **Importance** function estimates how good an attribute is for splitting – more on this latter

⇒ The **Plurality-Value** function basically does majority vote
Example: Playing Tennis?

Labeled data

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>

**Outlook:** S(unny), O(vercast), R(ainy)

**Temperature:** H(ot), M(ild), C(ool)

**Humidity:** H(igh), N(ormal), L(ow)

**Wind:** S(trong), W(ek)
Which Attribute?

⇒ Need to split over an attribute to start the tree

⇒ We can decide this using **entropy**

  ⇒ Entropy measures how chaotic something is
  
  ⇒ More entropy → less information

⇒ Given a probability distribution $P = p_1, ..., p_k$ with $p_i = P(X = x_i)$

⇒ Its entropy is computed as

$$ H = - \sum_i p_i \log_2 p_i $$

⇒ E.g., $k = 2, p_1 = 0.5, p_2 = 0.5, H = 1$

⇒ $k = 2, p_1 = 0, p_2 = 1, H = 0$

⇒ Entropy is a fundamental concept in information theory
Splitting Based on Information Gain

Given an attribute $A$ with values $a_1, \ldots, a_k$, suppose the number of positive labels is $n^+$ and the number of negative labels is $n^-$, the **initial entropy** is

$$H_A = -\left(\frac{n^+}{n^- + n^+} \log_2 \frac{n^+}{n^- + n^+} + \frac{n^-}{n^- + n^+} \log_2 \frac{n^-}{n^- + n^+}\right)$$

Suppose for $a_i$, #positive label is $n_i^+$, #negative label is $n_i^-$, then the leftover **entropy after** splitting using attribute $A$ is

$$H_A' = -\sum_{i=1}^{k} \frac{n_i^- + n_i^+}{n^+ + n^-} \left(\frac{n_i^+}{n_i^- + n_i^+} \log_2 \frac{n_i^+}{n_i^- + n_i^+} + \frac{n_i^-}{n_i^- + n_i^+} \log_2 \frac{n_i^-}{n_i^- + n_i^+}\right)$$

The information gain from the splitting is

$$\Delta H(A) = H_A - H_A'$$

We choose $A$ that maximizes $\Delta H(A)$
Example: Playing Tennis?

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>T</th>
<th>H</th>
<th>W</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>2</td>
<td>S</td>
<td>H</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>3</td>
<td>O</td>
<td>H</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>4</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>5</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>6</td>
<td>R</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>-</td>
</tr>
<tr>
<td>7</td>
<td>O</td>
<td>C</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>8</td>
<td>S</td>
<td>M</td>
<td>H</td>
<td>W</td>
<td>-</td>
</tr>
<tr>
<td>9</td>
<td>S</td>
<td>C</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>10</td>
<td>R</td>
<td>M</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>11</td>
<td>S</td>
<td>M</td>
<td>N</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>12</td>
<td>O</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>+</td>
</tr>
<tr>
<td>13</td>
<td>O</td>
<td>H</td>
<td>N</td>
<td>W</td>
<td>+</td>
</tr>
<tr>
<td>14</td>
<td>R</td>
<td>M</td>
<td>H</td>
<td>S</td>
<td>-</td>
</tr>
</tbody>
</table>

**Outlook:** S(unny), O(vercast), R(ainy)

**Temperature:** H(ot), M(ild), C(ool)

**Humidity:** H(igh), N(ormal), L(ow)

**Wind:** S(trong), W(weak)
The Issue of Overfitting

We do not want to **overfit** the data, e.g., using shoe sizes to infer gender. Overall males have larger shoes but not always true. E.g.,

8.5(M), 6(F), 9(F), 11(M), 13(M), 7.5(F)

Which tree is better?

- **Decision tree A**
  - Shoe size
    - > 8.5: Male
    - ≤ 8.5: Female

- **Decision tree B**
  - Shoe size
    - > 9.25: Male
    - 8.25 – 8.75: Female
    - ≤ 8.25, 8.75 – 9.25: Female

Tree A is likely better: tree B implies a much more complex, not well-supported hypothesis, indicating overfitting.
Linear Classifier

- A linear classifier for $n$ features (an $n$-dimensional space) is an $(n - 1)$-dimensional plane that separates the positive/negative labels.

- E.g., differentiate earthquake (white dots) and nuclear explosion (black dots) using features body wave magnitude ($x_1$) and surface wave magnitude ($x_2$).
Linear Classifier Model

- A linear classifier takes in \( n \) features and outputs a label, i.e. true/false

- Or simply \( y = h_w(x) = \text{sign} \ (w_0 + w_1 x_1 + \cdots + w_n x_n) \)

- Also known as perceptron – perhaps this is how neurons work

- Learning a linear classifier is to find appropriate weights \( w \)
The Perceptron Learning Rule

⇒ The **perceptron learning rule** is an iterative update rule for obtaining the appropriate \( w \)

⇒ Start with \( w(t = 0) = (w_0(0), w_1(0), \ldots, w_n(0)) = 0 \)

⇒ To update using data point \((x^j = (x_1^j, \ldots, x_n^j), y^j)\)

  ⇒ Compute \( \widehat{y}^j = h_w(t)(x^j) \)

  ⇒ Update, for all \( 0 \leq i \leq n, w_i(t + 1) = w_i(t) + \alpha(y^j - \widehat{y}^j)x_i^j \)

⇒ What is happening here?

  ⇒ \( y^j = \widehat{y}^j \)

  ⇒ \( y^j = 0, \widehat{y}^j = 1 \)

  ⇒ \( y^j = 1, \widehat{y}^j = 0 \)

⇒ Converges when the data is linearly separable

⇒ If not, letting \( \alpha \) decay slowly \((O \left(\frac{1}{t}\right))\) converges to a minimum error solution for randomly ordered data
The Course Project: Image Classification

You need to implement three classifiers

- Naïve Bayes classifier
- Linear classifier (perceptron)
- A classifier of your choice

Which digit? Which ones are faces?
The Course Project: Image Classification

⇒ Data (see project PDF, to be released later today, for details)

⇒ A digit or a face is in an ASCII file
The Course Project: Image Classification

- You should start with designing the features
- Implement the algorithms – do not use existing classification algorithms
- Compare the algorithms and report the prediction error (including standard deviation) as a function of the number of data points used for training
- Write a report (2+ pages) describing your implementation and results
- Submit the code and report by May 4\textsuperscript{th}, 11:55 pm
- Setup an appointment with TA to demonstrate your program before the final exam period is over