Welcome back!

Reminder: In the News Call #2 due tomorrow

Reminder: Midterm #2 is on March 13

Project proposals are all marked. You can resubmit your proposal after you have addressed the concerns raised by your TA. We will take the highest mark out of the project proposal and the project proposal resubmission.

Each group has a TA assigned to their project— they will be your project manager for the rest of the term. They are also the ones marking most of your project work. Make sure everyone is on the same page! Communicate often!
Frozen!

- Now that you have had some real life experience with snow, I thought I’d show you the video I wanted to show last class....
In the News

Paul Manafort left an electronic paper trail incriminating himself because he couldn’t convert a PDF document to a Word document

Data Mining
Learning Goals

- [CT Building Block] Students will be able to create English language descriptions of algorithms to analyze data and show how their algorithms would work on an input data set.
- [CT Application] Students will be able to use computing to examine datasets and facilitate exploration in order to gain insight and knowledge (data and information).
- [CT Impact] Students will be able to give examples of privacy and security issues that arise as a result of data mining.
- [CT Building Block] Students will be able to describe what a greedy algorithm is.
- [CT Building Block] Students will be able to describe what the general decisions are in building a decision tree.
- [CT Building Block] Students will be able to build a simple decision tree.
- [CT Building Block] Students will be able to describe what considerations are important in building a decision tree.
- [CT Building Block] Students will be able to give examples showing why clustering is useful, describe how clustering tasks can be formulated in terms of high-dimensional numerical data, and infer what is the output of the k-means clustering algorithm on a small input.
Why data mining?

• The world is awash with digital data; trillions of gigabytes and growing
• How many bytes in a gigabyte?  

Clicker question

A. 1 000 000
B. 1 000 000 000
C. 1 000 000 000 000
Why data mining?

- The world is awash with digital data; trillions of gigabytes and growing
- A trillion gigabytes is a zettabyte, or
  1 000 000 000 000 000 000 000 000 bytes
Why data mining?

More and more, businesses and institutions are using data mining to make decisions, classifications, diagnoses, and recommendations that affect our lives.
“We have the capacity to send every customer an ad booklet, specifically designed for them, that says, ‘Here’s everything you bought last week and a coupon for it,’ ” one Target executive told me. ‘We do that for grocery products all the time.’ But for pregnant women, Target’s goal was selling them baby items they didn’t even know they needed yet.”
Target can identify pregnant women and send them individual mailings

In a group of 3-4 discuss whether you think this is cool, creepy, or both

A. Cool
B. Creepy
C. Both
D. Neither
Target: Cool, creepy or both

**Cool**
- You might be a victim of food borne disease so you might benefit
- Pay off student loans by suing them for invasion of privacy
- Saves time

**Creepy**
- Ads that aren’t applicable to me are more irritating. It doesn’t feel creepy if you can figure out why they are giving you those coupons but if you pay for something in cash but you get a coupon, that’s creepy
Okay. What if I told you…

People pay different insurance (life or medical) rates based on being pregnant. Insurance companies may now want to pay Target for this information. Does this change your opinion?

A. It was already creepy
B. It wasn’t creepy before, but this is
C. No, still not creepy
Clicker question:
Loyalty cards and credit cards

After reading these articles, are you more or less likely to use a credit card/loyalty card for purchases:
A. More likely
B. Less likely
C. The same
As we discussed, cookies tell information about you. But how do pages that you’ve visited predict the future?
Data Mining

- Data mining is the process of looking for patterns in large data sets
- There are many different kinds for many different purposes
- We’ll do an in depth exploration of two of them
Data mining for classification
Recall our loan application example

Goal: given colours, credit ratings, and past rates of successfully paying back loans, decide to grant a loan or not.

Total profit = 32400
Data mining for classification

• In the loan strategy example, we focused on fairness of different classifiers, but we didn’t focus much on how to build a classifier
• Today you’ll learn how to build decision tree classifiers for simple data mining scenarios
Before we get to decision trees, we need to define a tree.
A rooted tree in computer science

A tree is a collection of nodes such that

• one node is the designated *root*

• a node can have zero or more *children*; a node with zero children is a *leaf*

• all non-root nodes have a single *parent*

• *edges* denote parent-child relationships

• nodes and/or edges may be labeled by data
A rooted tree in computer science
Often but not always drawn with root on top
Is this a rooted tree?
Clicker question

A. Yes
B. No
C. I’m not sure

Node 5 has two parents

http://jerome.boulinguez.free.fr/english/file/hotpotatoes/familytree.htm
Decision trees: trees whose node labels are attributes, edge labels are conditions

Decision Tree:
Should I accept a new job offer?

- Salary at least $50,000
  - commute more than 1 hour
    - offers free coffee
      - yes
        - accept offer
      - no
        - decline offer
    - no
      - decline offer
  - yes
    - accept offer
Decision trees: trees whose node labels are attributes, edge labels are conditions

Decision tree for Lyme Disease diagnosis

1. Enzyme Immunoassay
   - No: Consider Alternative Diagnosis
   - Yes: Test Symptom Length
     - ≤ 30 days: IgM and IgG Western Blot
     - > 30 days: IgG Western Blot ONLY
How Gerber Used a Decision Tree in Strategic Decision-Making

Possible outcomes explored in an in Products Safety Commission.

By JAY BUCKLEY and THOMAS J. DUDLEY, DBA
1999 Volume 2 Issue 3

Decision trees can assist executives in making strat

What's the Right Thing to Do?

- Is it ethical? (To answer, weigh the effect on customers, employees, the community, the environment, and suppliers against the benefit to the shareholders.)
  - yes: Do it.
  - no: Don't do it.

- Does it maximize shareholder value?
  - yes: Move on to the next question.
  - no: Is the proposed action legal?
    - yes: Move on to the next question.
    - no: Don't do it.

- Would it be ethical not to take the action? (To answer, weigh the harm or cost that would be imposed on shareholders against the costs or benefits to other stakeholders.)
  - yes: Don't do it.
  - no: Do it but disclose the effect of the action to shareholders.
Back to our example. We may want to make a tree saying when to approve or deny a loan.

Goal: given colours, credit ratings, and past rates of successfully paying back loans, decide to grant a loan or not.
Decision trees: trees whose node labels are attributes, edge labels are conditions

A decision tree for max profit loan strategy

(Note that some worthy applicants are denied loans, while other unworthy ones get loans)
Exercise: Construct the decision tree for the “Group Unaware” loan strategy

Goal: given colours, credit ratings, and past rates of successfully paying back loans, decide to grant a loan or not.

Total profit = 25600
Sample Decision Tree for “Group Unaware” strategy

A decision tree for max profit loan strategy

(Note that some worthy applicants are denied loans, while other unworthy ones get loans)
Building decision trees from training data

• Should you get an ice cream?
• You might start out with the following data
• You might build a decision tree that looks like this:
Shall we play a game? Help a soccer league decide whether to cancel games

Build a decision tree to help officials decide

Assume that decisions are the same given the same information

The leaf nodes should be whether or not to play

The non-leaf nodes should be attributes (e.g., Outlook, Windy)

The edges should be conditions (e.g., sunny, hot, normal)

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>No</td>
</tr>
</tbody>
</table>

Example adapted from http://www.kdnuggets.com/data_mining_course/index.html#materials
Some example potential starts to the decision tree

- Outlook?
  - Overcast
  - Rainy
  - Sunny
  - Temperature?
    - Humidity?
      - Windy?
        - Humidity?
        - Windy?
          - true
          - false
How did you split up your tree and why?

- Minimize the number of branches required in the decision tree
Deciding which nodes go where: A decision tree construction algorithm

- Top-down tree construction
  - At start, all examples are at the root.
  - Partition the examples recursively by choosing one attribute each time.
- In deciding which attribute to split on, one common method is to try to reduce entropy – i.e., each time you split, you should make the resulting groups more homogenous. The more you reduce entropy, the higher the information gain.
Let’s go back to our example

Intuitively, our goal is to get to having as few mixed “Yes” and “No” answers together in groups as possible.

So in the initial case, we have 14 mixed Yes’s and No’s.

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>No</td>
</tr>
</tbody>
</table>
What happens if we split on Temperature?

Overall entropy = 4 + 4 + 6 = 14
What’s the entropy if you split on Outlook?

Group exercise

A. 0  
B. 5  
C. 10  
D. 14  
E. None of the above

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Windy</th>
<th>Play?</th>
</tr>
</thead>
<tbody>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>hot</td>
<td>high</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>high</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>No</td>
</tr>
<tr>
<td>overcast</td>
<td>cool</td>
<td>normal</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>high</td>
<td>false</td>
<td>No</td>
</tr>
<tr>
<td>sunny</td>
<td>cool</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>sunny</td>
<td>mild</td>
<td>normal</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>overcast</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>Yes</td>
</tr>
<tr>
<td>overcast</td>
<td>hot</td>
<td>normal</td>
<td>false</td>
<td>Yes</td>
</tr>
<tr>
<td>rain</td>
<td>mild</td>
<td>high</td>
<td>true</td>
<td>No</td>
</tr>
</tbody>
</table>
What’s the entropy if you split on Outlook?

Group exercise results

Overall entropy = 5 + 0 + 5 = 10
What if you split on Windy?

Overall entropy = 8+6=14
What if you split on Humidity?

7 mixed

Humidity

high

normal

Yes
Yes
Yes
No
No
No
No

Yes
Yes
Yes
Yes
Yes
Yes
No

7 mixed

Overall entropy = 7 + 7 = 14
To summarize

- Entropy if we split on Outlook = 10
- Entropy if we split on Windy = 14
- Entropy if we split on Humidity = 14

→ The best option to split on is “Outlook”
It does the best job of reducing entropy
This example suggests why a more complex entropy definition might be better.

Humidity is better, even though both have “entropy” 14.
Great! Now we do the same thing again

Here’s what we have so far:

```
Outlook
  └── sunny
  │
  ├── overcast
  │
  │  └── rainy
  │
  │
```

For each option, we have to decide which attribute to split on next: Temperature, Windy, or Humidity.
Great! Now we do the same thing again

Clicker question

What’s the best attribute to split on for Outlook = sunny?

A. Temperature
   - hot
     - No
     - Yes
     - Yes
   - mild
     - Yes
   - cool
     - Yes

B. Windy
   - false
     - Yes
     - Yes
     - No
   - true
     - No

C. Humidity
   - high
     - No
   - normal
     - Yes
     - Yes
We don’t need to split for Outlook = overcast

The answer was yes each time. So we’re done there.
What’s the best attribute to split on for Outlook = rain? Clicker question

A. Temperature

<table>
<thead>
<tr>
<th>Temperature</th>
<th>hot</th>
<th>mild</th>
<th>cool</th>
</tr>
</thead>
<tbody>
<tr>
<td>N/A</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>No</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

B. Windy

<table>
<thead>
<tr>
<th>Windy</th>
<th>true</th>
<th>false</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

C. Humidity

<table>
<thead>
<tr>
<th>Humidity</th>
<th>high</th>
<th>normal</th>
</tr>
</thead>
<tbody>
<tr>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
This was, of course, a simple example

- In this example, the algorithm found the tree with the smallest number of nodes
- We were given the attributes and conditions
- A simplistic notion of entropy worked (a more sophisticated notion of entropy is typically used to determine which attribute to split on)
This was, of course, a simple example

- In more complex examples, like the loan application example
  - We may not know which conditions or attributes are best to use
  - The final decision may not be correct in every case (e.g., given two loan applicants with the same colour and credit rating, one may be credit worthy while the other is not)
  - Even if the final decision is always correct, the tree may not be of minimum size
Coding up a decision tree classifier

- **Outlook**
  - sunny
  - overcast
  - rainy

- **Humidity**
  - high
  - normal
  - No
  - Yes

- **Windy**
  - false
  - true
  - Yes
  - No
Coding up a decision tree classifier

Can you see the relationship between the hierarchical tree structure and the hierarchical nesting of “if” statements?
Coding up a decision tree classifier

Outlook

- sunny
  - high
    - normal
      - yes
      - no
  - low
    - yes
    - no

Windy

- sunny
  - high
    - normal
      - yes
      - no
  - low
    - yes
    - no

Can you extend the code to handle the "rainy" case?