ECE 5984: Introduction to Machine Learning

Topics:
- Decision/Classification Trees

Readings: Murphy 16.1-16.2; Hastie 9.2

Dhruv Batra
Virginia Tech
Project Proposals Graded

• Mean 3.6/5 = 72%
Administrativia

• Project Mid-Sem Spotlight Presentations
  – Friday: 5-7pm, 3-5pm Whittemore 654 457A
    – 5 slides (recommended)
    – 4 minute time (STRICT) + 1-2 min Q&A
    – Tell the class what you’re working on
    – Any results yet?
    – Problems faced?
    – Upload slides on Scholar
Recap of Last Time
Convolution Explained

- [https://github.com/bruckner/deepViz](https://github.com/bruckner/deepViz)
Fully Connected Layer

Example: 200x200 image
40K hidden units
~2B parameters!!!

- Spatial correlation is local
- Waste of resources + we have not enough training samples anyway..
Locally Connected Layer

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

Note: This parameterization is good when input image is registered (e.g., face recognition).

Slide Credit: Marc'Aurelio Ranzato
Locally Connected Layer

**STATIONARITY?** Statistics is similar at different locations

Example: 200x200 image
40K hidden units
Filter size: 10x10
4M parameters

**Note:** This parameterization is good when input image is registered (e.g., face recognition).
Convolutional Layer

Share the same parameters across different locations (assuming input is stationary):

Convolutions with learned kernels

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer
Convolutional Layer

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
-1 & 0 & 1 \\
\end{bmatrix}
\]

Slide Credit: Marc'Aurelio Ranzato
Convolutional Layer

Learn multiple filters.

E.g.: 200x200 image
100 Filters
Filter size: 10x10
10K parameters

Slide Credit: Marc'Aurelio Ranzato

Ranzato
Pooling Layer

Let us assume filter is an “eye” detector.

Q.: how can we make the detection robust to the exact location of the eye?
Pooling Layer

By “pooling” (e.g., taking max) filter responses at different locations we gain robustness to the exact spatial location of features.
Convolutional Nets

• Example:
Visualizing Learned Filters

Layer 1

Layer 2

Figure Credit: [Zeiler & Fergus ECCV14]
Visualizing Learned Filters

Layer 3

Figure Credit: [Zeiler & Fergus ECCV14]
Visualizing Learned Filters

Layer 4

Layer 5

Figure Credit: [Zeiler & Fergus ECCV14]
Addressing non-linearly separable data – Option 1, non-linear features

• Choose non-linear features, e.g.,
  – Typical linear features: \( w_0 + \sum_i w_i x_i \)
  – Example of non-linear features:
    • Degree 2 polynomials, \( w_0 + \sum_i w_i x_i + \sum_{ij} w_{ij} x_i x_j \)

• Classifier \( h_w(x) \) still linear in parameters \( w \)
  – As easy to learn
  – Data is linearly separable in higher dimensional spaces
  – Express via kernels
Addressing non-linearly separable data – Option 2, non-linear classifier

• Choose a classifier $h_w(x)$ that is non-linear in parameters $w$, e.g.,
  – Decision trees, neural networks, …

• More general than linear classifiers
• But, can often be harder to learn (non-convex/concave optimization required)
• Often very useful (outperforms linear classifiers)
• In a way, both ideas are related
New Topic: Decision Trees

You Dropped Food on the Floor
Do You Eat It?

Was it sticky?  No.  Did anyone see you?  Yes.

No.

Is it an Emausaurus?

Yes.

Is it a raw steak?

Yes.

Was it a boss/lover/parent?  No.

Yes.

Was it expensive?  Yes.

No.

Can you cut off the part that touched the floor?

Yes.

EAT IT.

No.

Are you a puma?

Yes.

Is it bacon?

Yes.

EAT IT.

No.

Are you a Megalosaurus?

Yes.

Did the cat lick it?

Yes.

EAT IT.

No.

Don't EAT IT

Is your cat healthy?

Yes.

EAT IT.

No.

Your Call

EAT IT.
Synonyms

• Decision Trees

• Classification and Regression Trees (CART)

• Algorithms for learning decision trees:
  – ID3
  – C4.5

• Random Forests
  – Multiple decision trees
Decision Trees

• Demo
  – http://www.cs.technion.ac.il/~rani/LocBoost/
Pose Estimation

• Random Forests!
  – Multiple decision trees
  – http://youtu.be/HNkbG3KsY84
Learning Decision Trees

Decision trees provide a very popular and efficient hypothesis space.

- **Variable Size.** Any boolean function can be represented.
- **Deterministic.**
- **Discrete and Continuous Parameters.**
Decision Tree Decision Boundaries

Decision trees divide the feature space into axis-parallel rectangles, and label each rectangle with one of the $K$ classes.
Decision Trees Can Represent Any Boolean Function

The tree will in the worst case require exponentially many nodes, however.
Decision Trees Provide Variable-Size Hypothesis Space

As the number of nodes (or depth) of tree increases, the hypothesis space grows

- **depth 1** ("decision stump") can represent any boolean function of one feature.

- **depth 2** Any boolean function of two features; some boolean functions involving three features (e.g., \((x_1 \land x_2) \lor (\neg x_1 \land \neg x_3)\))

- etc.
# A small dataset: Miles Per Gallon

<table>
<thead>
<tr>
<th>mpg</th>
<th>cylinders</th>
<th>displacement</th>
<th>horsepower</th>
<th>weight</th>
<th>acceleration</th>
<th>modelyear</th>
<th>maker</th>
</tr>
</thead>
<tbody>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>75to78</td>
<td>asia</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>75to78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td>:</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>8</td>
<td>high</td>
<td>medium</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>75to78</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>high</td>
<td>75to78</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>medium</td>
<td>low</td>
<td>low</td>
<td>79to83</td>
<td>america</td>
<td></td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>low</td>
<td>medium</td>
<td>high</td>
<td>79to83</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>high</td>
<td>high</td>
<td>high</td>
<td>low</td>
<td>70to74</td>
<td>america</td>
</tr>
<tr>
<td>good</td>
<td>4</td>
<td>low</td>
<td>medium</td>
<td>low</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>medium</td>
<td>75to78</td>
<td>europe</td>
</tr>
</tbody>
</table>

40 Records

Suppose we want to predict MPG

From the UCI repository (thanks to Ross Quinlan)
A Decision Stump

mpg values: bad good

root

22 18

cylinders = 3

0 0

Predict bad

cylinders = 4

4 17

Predict good

cylinders = 5

1 0

Predict bad

cylinders = 6

8 0

Predict bad

cylinders = 8

9 1

Predict bad
mpg values:  bad  good

The final tree

Slide Credit: Carlos Guestrin

(C) Dhruv Batra
Comments

• Not all features/attributes need to appear in the tree.

• A features/attribute $X_i$ may appear in multiple branches.

• On a path, no feature may appear more than once.
  – Not true for continuous features. We’ll see later.

• Many trees can represent the same concept
• But, not all trees will have the same size!
  – e.g., $Y = (A^B) \lor (\neg A^C)$ (A and B) or (not A and C)
Learning decision trees is hard!!!

- Learning the simplest (smallest) decision tree is an NP-complete problem [Hyafil & Rivest ’76]

- Resort to a greedy heuristic:
  - Start from empty decision tree
  - Split on next best attribute (feature)
  - Recurse
    - “Iterative Dichotomizer” (ID3)
    - C4.5 (ID3+improvements)
Recursion Step

Take the original dataset and partition it according to the value of the attribute we split on.

- Records in which cylinders = 4
  - mpg values: bad good
  - root
    - Predict bad: 22
    - Predict good: 18

- Records in which cylinders = 5
  - cylinders = 3: 0 0
  - cylinders = 4: 4 17
  - cylinders = 5: 1 0

- Records in which cylinders = 6
  - cylinders = 6: 8 0

- Records in which cylinders = 8
  - cylinders = 8: 9 1

And partition it according to the value of the attribute we split on.
Recursion Step

mpg values: bad good

Build tree from These records..
Build tree from These records..
Build tree from These records..
Build tree from These records..

Records in which cylinders = 4
Records in which cylinders = 5
Records in which cylinders = 6
Records in which cylinders = 8
Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia
The final tree

mpg values: bad good

root
22 18

- cylinders = 3
  - Predict bad

- cylinders = 4
  - Predict bad

- cylinders = 5
  - Predict bad

- cylinders = 6
  - Predict good

- cylinders = 8
  - Predict bad

- maker = america
  - Predict good

- maker = asia
  - Predict bad

- maker = europe
  - Predict bad

- horsepower = low
  - Predict good

- horsepower = medium
  - Predict bad

- horsepower = high
  - Predict bad

- acceleration = low
  - Predict bad

- acceleration = medium
  - Predict good

- acceleration = high
  - Predict bad

- modelyear = 70to74
  - Predict bad

- modelyear = 75to78
  - Predict bad

- modelyear = 79to83
  - Predict bad

(C) Dhruv Batra

Slide Credit: Carlos Guestrin
Choosing a good attribute

<table>
<thead>
<tr>
<th>X₁</th>
<th>X₂</th>
<th>Y</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>
Measuring uncertainty

- Good split if we are more certain about classification after split
  - Deterministic good (all true or all false)
  - Uniform distribution bad

\[
\begin{align*}
P(Y=F \mid X_1 = T) &= 0 \\
P(Y=T \mid X_1 = T) &= 1 \\
P(Y=F \mid X_2 = F) &= \frac{1}{2} \\
P(Y=T \mid X_2 = F) &= \frac{1}{2}
\end{align*}
\]
Entropy $H(X)$ of a random variable $Y$

$$H(Y) = - \sum_{i=1}^{k} P(Y = y_i) \log_2 P(Y = y_i)$$

*More uncertainty, more entropy!*

*Information Theory interpretation: $H(Y)$ is the expected number of bits needed to encode a randomly drawn value of $Y$ (under most efficient code)*
Information gain

• Advantage of attribute – decrease in uncertainty
  – Entropy of Y before you split
  – Entropy after split
    • Weight by probability of following each branch, i.e., normalized number of records

\[
H(Y | X) = - \sum_{j=1}^{v} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i | X = x_j) \log_2 P(Y = y_i | X = x_j)
\]

• Information gain is difference \( IG(X) = H(Y) - H(Y | X) \)
  – (Technically it’s mutual information; but in this context also referred to as information gain)
Learning decision trees

- Start from empty decision tree
- Split on **next best attribute (feature)**
  - Use, for example, information gain to select attribute
  - Split on $\arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y | X_i)$
- Recurse
Suppose we want to predict MPG

Look at all the information gains...
When do we stop?
Don’t split a node if all matching records have the same output value.
Base Case Two: No attributes can distinguish

Don’t split a node if none of the attributes can create multiple non-empty children.
Base Cases

- Base Case One: If all records in current data subset have the same output then don’t recurse
- Base Case Two: If all records have exactly the same set of input attributes then don’t recurse
Base Cases: An idea

- Base Case One: If all records in current data subset have the same output then don’t recurse
- Base Case Two: If all records have exactly the same set of input attributes then don’t recurse

Proposed Base Case 3:

If all attributes have zero information gain then don’t recurse

• Is this a good idea?
The problem with Base Case 3

<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

\[ y = a \text{ XOR } b \]

The information gains:

<table>
<thead>
<tr>
<th>Input</th>
<th>Value</th>
<th>Distribution</th>
<th>Info Gain</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>0</td>
<td>Blue/Red</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Blue/Red</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>Blue/Red</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>Blue/Red</td>
<td></td>
</tr>
</tbody>
</table>

The resulting decision tree:

```
  y values:  0  1
    root
      2  2
      Predict 0
```
If we omit Base Case 3:

\[
\begin{array}{ccc}
  a & b & y \\
  0 & 0 & 0 \\
  0 & 1 & 1 \\
  1 & 0 & 1 \\
  1 & 1 & 0 \\
\end{array}
\]

\( y = a \text{ XOR } b \)

The resulting decision tree: