Generative v. Discriminative classifiers – Intuition

- **Want to Learn**: h: X \(\mapsto\) Y
  - X – features
  - Y – target classes
- **Bayes optimal classifier** – P(Y|X)
- **Generative classifier**, e.g., Naïve Bayes:
  - Assume some **functional form for P(X|Y), P(Y)**
  - Estimate parameters of P(X|Y), P(Y) directly from training data
  - Use Bayes rule to calculate P(Y|X=x)
  - This is a ‘**generative**’ model
    - Indirect computation of P(Y|X) through Bayes rule
      - But, **can generate a sample of the data**, P(X) = \(\sum_y P(y) P(X|y)\)
- **Discriminative classifiers**, e.g., Logistic Regression:
  - Assume some **functional form for P(Y|X)**
  - Estimate parameters of P(Y|X) directly from training data
  - This is the ‘**discriminative**’ model
    - Directly learn P(Y|X)
    - But **cannot obtain a sample of the data**, because P(X) is not available
Logistic Regression

- Learn $P(Y|X)$ directly!
  - Assume a particular functional form
  - Sigmoid applied to a linear function of the data:
    \[
    P(Y = 1|X) = \frac{1}{1 + \exp(w_0 + \sum_{i=1}^{n} w_i x_i)}
    \]
    \[
    P(Y = 0|X) = 1 - P(Y = 1|X)
    \]

Features can be discrete or continuous!

Logistic Regression – a Linear classifier

- $g(w_0 + \sum_i w_i x_i) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)}$
- $P(Y = 1|X) > 0.5$ when $-w_0 + \sum_i w_i x_i > 0$
  $\Rightarrow w_0 + \sum_i w_i x_i < 0$
  $\Rightarrow P(Y = 0|X) > 0.5$
  $\Rightarrow w_0 + \sum_i w_i x_i > 0$
Very convenient!

\[ P(Y = 1 | X = < X_1, ..., X_n >) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

implies

\[ P(Y = 0 | X = < X_1, ..., X_n >) = \frac{\exp(w_0 + \sum_i w_i X_i)}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

implies

\[ \frac{P(Y = 0 | X)}{P(Y = 1 | X)} = \exp(w_0 + \sum_i w_i X_i) \]

implies

\[ \ln \frac{P(Y = 0 | X)}{P(Y = 1 | X)} = w_0 + \sum_i w_i X_i \]

Logistic regression v. Naïve Bayes

- Consider learning \( f: X \rightarrow Y \), where
  - \( X \) is a vector of real-valued features, \(< X_1 \ldots X_n >\)
  - \( Y \) is boolean

- Could use a Gaussian Naïve Bayes classifier
  - assume all \( X_i \) are conditionally independent given \( Y \)
  - model \( P(X_i | Y = y_k) \) as Gaussian \( N(\mu_{ik}, \sigma_i) \)
  - model \( P(Y) \) as Bernoulli(\( \theta, 1-\theta \))

- What does that imply about the form of \( P(Y|X) \)?

\[ P(Y = 1 | X = < X_1, ..., X_n >) = \frac{1}{1 + \exp(w_0 + \sum_i w_i X_i)} \]

Cool!!!!
Derive form for $P(Y|X)$ for continuous $X_i$

$$P(Y = 1|X) = \frac{P(Y = 1)P(X|Y = 1)}{P(Y = 1)P(X|Y = 1) + P(Y = 0)P(X|Y = 0)}$$

$$= \frac{1}{1 + \frac{P(Y = 0)P(X|Y = 0)}{P(Y = 1)P(X|Y = 1)}}$$

$$= \frac{1}{1 + \exp\left(\ln \frac{P(Y = 0)P(X|Y = 0)}{P(Y = 1)P(X|Y = 1)}\right)}$$

$$= \frac{1}{1 + \exp\left(\ln \frac{1}{\theta} + \sum_i \ln \frac{P(X_i|Y = 0)}{P(X_i|Y = 1)}\right)}$$

---

Ratio of class-conditional probabilities

$$\ln \frac{P(X_i|Y = 0)}{P(X_i|Y = 1)}$$

$p(X_i = x | Y = y_k) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{(x-y_k)^2}{2\sigma^2}}$
Derive form for \( P(Y|X) \) for continuous \( X_i \)

\[
P(Y = 1|X) = \frac{P(Y = 1)P(X|Y = 1)}{P(Y = 1)P(X|Y = 1) + P(Y = 0)P(X|Y = 0)}
\]

\[
= \frac{1}{1 + \exp(-\ln \frac{1-\theta}{\theta} + \sum_i \ln \frac{P(X_i|Y=0)}{P(X_i|Y=1)})}
\]

\[
= \frac{\sum \left( \frac{\mu_{i0} - \mu_{i1}}{\sigma_i^2} X_i + \frac{\mu_{i1}^2 - \mu_{i0}^2}{2\sigma_i^2} \right)}{1 + \exp\left(w_0 + \sum_{i=1}^n w_i X_i\right)}
\]

Gaussian Naïve Bayes v. Logistic Regression

- Representation equivalence
  - But only in a special case!!! (GNB with class-independent variances)
- But what's the difference???
- LR makes no assumptions about \( P(X|Y) \) in learning!!!
- Loss function!!!
  - Optimize different functions \(\rightarrow\) Obtain different solutions
Logistic regression for more than 2 classes

- Logistic regression in more general case, where \( Y \in \{Y_1 \ldots Y_R\} \) : learn \( R-1 \) sets of weights

Logistic regression more generally

- Logistic regression in more general case, where \( Y \in \{Y_1 \ldots Y_R\} \) : learn \( R-1 \) sets of weights

For \( k < R \)

\[
P(Y = y_k | X) = \frac{\exp(w_{k0} + \sum_{i=1}^{n} w_{ki}X_i)}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}
\]

For \( k = R \) (normalization, so no weights for this class)

\[
P(Y = y_R | X) = \frac{1}{1 + \sum_{j=1}^{R-1} \exp(w_{j0} + \sum_{i=1}^{n} w_{ji}X_i)}
\]

Features can be discrete or continuous!
Announcements

- Don’t forget recitation tomorrow
- And start the homework early

Loss functions: Likelihood v. Conditional Likelihood

- Generative (Naïve Bayes) Loss function:
  Data likelihood
  \[ \ln P(D \mid w) = \sum_{j=1}^{N} \ln P(x^j, y^j \mid w) \]
  \[ = \sum_{j=1}^{N} \ln P(y^j \mid x^j, w) + \sum_{j=1}^{N} \ln P(x^j \mid w) \]

- Discriminative models cannot compute \( P(x \mid w) \)!
- But, discriminative (logistic regression) loss function:
  Conditional Data Likelihood
  \[ \ln P(D_Y \mid D_X, w) = \sum_{j=1}^{N} \ln P(y^j \mid x^j, w) \]
  - Doesn’t waste effort learning \( P(X) \) – focuses on \( P(Y \mid X) \) all that matters for classification
Expressing Conditional Log Likelihood

\[ l(w) \equiv \sum_j \ln P(y^j|x^j, w) \]

\[ P(Y = 0|X, w) = \frac{1}{1 + \exp(w_0 + \sum_i w_ix_i)} \]

\[ P(Y = 1|X, w) = \frac{\exp(w_0 + \sum_i w_ix_i)}{1 + \exp(w_0 + \sum_i w_ix_i)} \]

\[ l(w) = \sum_j y^j \ln P(y^j = 1|x^j, w) + (1 - y^j) \ln P(y^j = 0|x^j, w) \]

Maximizing Conditional Log Likelihood

\[ l(w) \equiv \ln \prod_j P(y^j|x^j, w) \]

\[ P(Y = 0|X, W) = \frac{1}{1 + \exp(w_0 + \sum_i w_i x_i)} \]

\[ P(Y = 1|X, W) = \frac{\exp(w_0 + \sum_i w_i x_i)}{1 + \exp(w_0 + \sum_i w_i x_i)} \]

\[ l(w) = \sum_j y^j(w_0 + \sum_i w_ix_i^j) - \ln(1 + \exp(w_0 + \sum_i w_ix_i^j)) \]

**Good news:** \( l(w) \) is concave function of \( w \) → no locally optimal solutions

**Bad news:** no closed-form solution to maximize \( l(w) \)

**Good news:** concave functions easy to optimize
Optimizing concave function – Gradient ascent

- Conditional likelihood for Logistic Regression is concave → Find optimum with gradient ascent
  
  **Gradient:** \( \nabla_w \ell(w) = \left[ \frac{\partial \ell(w)}{\partial w_0}, \ldots, \frac{\partial \ell(w)}{\partial w_n} \right]' \)

  **Update rule:**
  \[
  \Delta w = \eta \nabla_w \ell(w) \\
  w^{(t+1)}_i = w^{(t)}_i + \eta \frac{\partial \ell(w)}{\partial w_i}
  \]

- Gradient ascent is simplest of optimization approaches
  - e.g., Conjugate gradient ascent much better (see reading)

Maximize Conditional Log Likelihood: Gradient ascent

\[
\ell(w) = \sum_j y^j (w_0 + \sum_i^n w_i x_i^j) - \ln(1 + \exp(w_0 + \sum_i^n w_i x_i^j))
\]
Gradient Descent for LR

Gradient ascent algorithm: iterate until change < $\epsilon$

$$w_0^{(t+1)} \leftarrow w_0^{(t)} + \eta \sum_j [y^j - \hat{P}(Y^j = 1 \mid x^j, w)]$$

For $i = 1 \ldots n$, 

$$w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_{ij}^j [y^j - \hat{P}(Y^j = 1 \mid x^j, w)]$$

repeat

That’s all M(C)LE. How about MAP?

$$p(w \mid Y, X) \propto P(Y \mid X, w)p(w)$$

- One common approach is to define priors on $w$
  - Normal distribution, zero mean, identity covariance
  - "Pushes" parameters towards zero
- Corresponds to **Regularization**
  - Helps avoid very large weights and overfitting
  - More on this later in the semester

- MAP estimate

$$w^* = \arg \max_w \ln \left[ p(w) \prod_{j=1}^N P(y^j \mid x^j, w) \right]$$

©2005-2007 Carlos Guestrin
M(C)AP as Regularization

\[ \ln \left[ p(w) \prod_{j=1}^{N} P(y^j | x^j, w) \right] \]

\[ p(w) = \prod_{i} \frac{1}{\kappa \sqrt{2\pi}} \frac{-w_i^2}{e^{2\kappa^2}} \]

Penalizes high weights, also applicable in linear regression

Gradient of M(C)AP

\[ \frac{\partial}{\partial w_i} \ln \left[ p(w) \prod_{j=1}^{N} P(y^j | x^j, w) \right] \]

\[ p(w) = \prod_{i} \frac{1}{\kappa \sqrt{2\pi}} \frac{-w_i^2}{e^{2\kappa^2}} \]
MLE vs MAP

- Maximum conditional likelihood estimate
  \[ w^* = \arg \max_w \ln \left( \prod_{j=1}^{N} P(y^j | x^j, w) \right) \]
  \[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 | x^j, w)] \]

- Maximum conditional a posteriori estimate
  \[ w^* = \arg \max_w \ln \left[ p(w) \prod_{j=1}^{N} P(y^j | x^j, w) \right] \]
  \[ w_i^{(t+1)} \leftarrow w_i^{(t)} + \eta \left\{ -\lambda w_i^{(t)} + \sum_j x_i^j [y^j - \hat{P}(Y^j = 1 | x^j, w)] \right\} \]

Naïve Bayes vs Logistic Regression

Consider Y boolean, X_i continuous, X=<X_1 ... X_n>

Number of parameters:
- NB: 4n +1
- LR: n+1

Estimation method:
- NB parameter estimates are uncoupled
- LR parameter estimates are coupled
G. Naïve Bayes vs. Logistic Regression 1

- Generative and Discriminative classifiers

- Asymptotic comparison (# training examples → infinity)
  - when model correct
    - GNB, LR produce identical classifiers
  - when model incorrect
    - LR is less biased – does not assume conditional independence
      - therefore LR expected to outperform GNB

[Ng & Jordan, 2002]

G. Naïve Bayes vs. Logistic Regression 2

- Generative and Discriminative classifiers

- Non-asymptotic analysis
  - convergence rate of parameter estimates, \( n = \# \) of attributes in \( X \)
    - Size of training data to get close to infinite data solution
    - GNB needs \( O(\log n) \) samples
    - LR needs \( O(n) \) samples

  - GNB converges more quickly to its (perhaps less helpful) asymptotic estimates

[Ng & Jordan, 2002]
Some experiments from UCI data sets

**What you should know about Logistic Regression (LR)**

- Gaussian Naïve Bayes with class-independent variances representationally equivalent to LR
  - Solution differs because of objective (loss) function
- In general, NB and LR make different assumptions
  - NB: Features independent given class → assumption on $P(X|Y)$
  - LR: Functional form of $P(Y|X)$, no assumption on $P(X|Y)$
- LR is a linear classifier
  - decision rule is a hyperplane
- LR optimized by conditional likelihood
  - no closed-form solution
  - concave → global optimum with gradient ascent
  - Maximum conditional a posteriori corresponds to regularization
- Convergence rates
  - GNB (usually) needs less data
  - LR (usually) gets to better solutions in the limit

©2005-2007 Carlos Guestrin
Linear separability

- A dataset is **linearly separable** iff ∃ a separating hyperplane:
  - ∃ \( w \), such that:
    - \( w_0 + \sum_i w_i x_i > 0 \); if \( x = \{x_1, \ldots, x_n\} \) is a positive example
    - \( w_0 + \sum_i w_i x_i < 0 \); if \( x = \{x_1, \ldots, x_n\} \) is a negative example

Not linearly separable data

- Some datasets are **not linearly separable**!
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 $w$
  - Usually easy to learn (closed-form or convex/concave optimization)
  - Data is linearly separable in higher dimensional spaces
  - More discussion later this semester

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, nearest neighbor,…
- More general than linear classifiers
- But, can often be harder to learn (non-convex/concave optimization required)
- But, but, often very useful
- (BTW. Later this semester, we’ll see that these options are not that different)
A small dataset: Miles Per Gallon

Suppose we want to predict MPG

From the UCI repository (thanks to Ross Quinlan)

©2005-2007 Carlos Guestrin

A Decision Stump

©2005-2007 Carlos Guestrin
Recursion Step

Take the Original Dataset...

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

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
Second level of tree

Recursively build a tree from the seven records in which there are four cylinders and the maker was based in Asia

(Similar recursion in the other cases)

The final tree
Classification of a new example

- Classifying a test example – traverse tree and report leaf label

Are all decision trees equal?

- Many trees can represent the same concept
- But, not all trees will have the same size!
  - e.g., $\phi = A \land B \lor \neg A \land C \equiv ((A \land B) \lor (\neg A \land C))$
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

Choosing a good attribute

<table>
<thead>
<tr>
<th>$X_1$</th>
<th>$X_2$</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

<table>
<thead>
<tr>
<th>P(Y=A) = 1/2</th>
<th>P(Y=B) = 1/4</th>
<th>P(Y=C) = 1/8</th>
<th>P(Y=D) = 1/8</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>P(Y=A) = 1/4</th>
<th>P(Y=B) = 1/4</th>
<th>P(Y=C) = 1/4</th>
<th>P(Y=D) = 1/4</th>
</tr>
</thead>
</table>

Entropy

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)
Andrew Moore’s Entropy in a nutshell

Low Entropy  High Entropy

..the values (locations of soup) sampled entirely from within the soup bowl

..the values (locations of soup) unpredictable... almost uniformly sampled throughout our dining room
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
      \[ H(Y \mid X) = -\sum_{i=1}^{k} P(X = x_j) \sum_{i=1}^{k} P(Y = y_i \mid X = x_j) \log_2 P(Y = y_i \mid X = x_j) \]

- Information gain is difference
  \[ IG(X) = H(Y) - H(Y \mid X) \]

Learning decision trees

- Start from empty decision tree
- Split on next best attribute (feature)
  - Use, for example, information gain to select attribute
  - Split on \( \arg \max_i IG(X_i) = \arg \max_i H(Y) - H(Y \mid X_i) \)
- Recurse
Information Gain Example

Suppose we want to predict MPG

Look at all the information gains...
A Decision Stump

mpg values: bad good

root
22 18
pchance = 0.001

- 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

Base Case

Don’t split a node if all matching records have the same output value
Don’t split a node if none of the attributes can create multiple non-empty children.

Base Case Two: No attributes can distinguish.
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

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

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

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>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>b</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td></td>
<td>1</td>
<td>1</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:

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

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

The resulting decision tree:
Basic Decision Tree Building
Summarized

BuildTree(DataSet, Output)

- If all output values are the same in DataSet, return a leaf node that says "predict this unique output"
- If all input values are the same, return a leaf node that says "predict the majority output"
- Else find attribute $X$ with highest Info Gain
  - Suppose $X$ has $n_X$ distinct values (i.e. $X$ has arity $n_X$).
    - Create and return a non-leaf node with $n_X$ children.
    - The $i$th child should be built by calling BuildTree($DS_i$, Output)
      Where $DS_i$ built consists of all those records in DataSet for which $X = i$th distinct value of $X$.

Real-Valued inputs

- What should we do if some of the inputs are real-valued?

<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>97</td>
<td>75</td>
<td>2265</td>
<td>18.2</td>
<td>77</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>158</td>
<td>90</td>
<td>2648</td>
<td>16</td>
<td>70</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>123</td>
<td>110</td>
<td>2880</td>
<td>12.8</td>
<td>77</td>
<td>europe</td>
</tr>
<tr>
<td>bad</td>
<td>8</td>
<td>350</td>
<td>175</td>
<td>4100</td>
<td>13</td>
<td>73</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>6</td>
<td>198</td>
<td>95</td>
<td>3100</td>
<td>16.5</td>
<td>74</td>
<td>america</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>108</td>
<td>94</td>
<td>2379</td>
<td>16.5</td>
<td>73</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>4</td>
<td>113</td>
<td>95</td>
<td>2226</td>
<td>14</td>
<td>71</td>
<td>asia</td>
</tr>
<tr>
<td>bad</td>
<td>5</td>
<td>302</td>
<td>139</td>
<td>3657</td>
<td>12.8</td>
<td>78</td>
<td>america</td>
</tr>
</tbody>
</table>

Infinite number of possible split values!!!

Finite dataset, only finite number of relevant splits!

Idea One: Branch on each possible real value
“One branch for each numeric value” idea:

Hopeless: with such high branching factor will shatter the dataset and overfit

Threshold splits

- Binary tree, split on attribute $X$
  - One branch: $X < t$
  - Other branch: $X \geq t$
Choosing threshold split

- Binary tree, split on attribute $X$
  - One branch: $X < t$
  - Other branch: $X \geq t$
- Search through possible values of $t$
  - Seems hard!!
- But only finite number of $t$'s are important
  - Sort data according to $X$ into $\{x_1, \ldots, x_m\}$
  - Consider split points of the form $x_i + (x_{i+1} - x_i)/2$

A better idea: thresholded splits

- Suppose $X$ is real valued
- Define $IG(Y|X:t)$ as $H(Y) - H(Y|X:t)$
- Define $H(Y|X:t) = H(Y|X < t) P(X < t) + H(Y|X \geq t) P(X \geq t)$
  - $IG(Y|X:t)$ is the information gain for predicting $Y$ if all you know is whether $X$ is greater than or less than $t$
- Then define $IG^*(Y|X) = \max_t IG(Y|X:t)$
- For each real-valued attribute, use $IG^*(Y|X)$ for assessing its suitability as a split
Example with MPG

Example tree using reals
What you need to know about decision trees

- Decision trees are one of the most popular data mining tools
  - Easy to understand
  - Easy to implement
  - Easy to use
  - Computationally cheap (to solve heuristically)
- Information gain to select attributes (ID3, C4.5,…)
- Presented for classification, can be used for regression and density estimation too
- It’s possible to get in trouble with overfitting (more next lecture)

Acknowledgements

- Some of the material in the presentation is courtesy of Tom Mitchell, and of Andrew Moore, from his excellent collection of ML tutorials:
  - [http://www.cs.cmu.edu/~awm/tutorials](http://www.cs.cmu.edu/~awm/tutorials)