DS 4400

Machine Learning and Data Mining I

Alina Oprea
Associate Professor, CCIS
Northeastern University

February 7 2019
Outline

• Linear Discriminant Analysis (LDA)
• Lab (logistic regression, LDA, kNN)
• Feature selection
  – Wrapper
  – Filter
  – Embedded methods
• Decision trees
  – Information Gain
Cross Validation

- CV can be used for
  - Hyper-parameter selection
  - Comparing different models and features

- 1. k-fold Cross-Validation
  - Split data into k partitions of equal size

Compute error metrics in each fold
Average error across folds
LDA

• Classify to one of k classes
• Logistic regression computes directly
  – \( P[Y = 1|X = x] \)
  – Assume sigmoid function
• LDA uses Bayes Theorem to estimate it
  \[
  P[Y = k|X = x] = \frac{P[X = x|Y = k]P[Y=k]}{P[X=x]}
  \]
  – Let \( \pi_k = P[Y = k] \) be the prior probability of class \( k \) and \( f_k(x) = P[X = x|Y = k] \)
LDA

Assume \( f_k(x) \) is Gaussian!

Unidimensional case (d=1)

\[
Pr(Y = k | X = x) = \frac{\pi_k f_k(x)}{\sum_{l=1}^{K} \pi_l f_l(x)}.
\]

\[
f_k(x) = \frac{1}{\sqrt{2\pi\sigma_k}} \exp \left( -\frac{1}{2\sigma_k^2} (x - \mu_k)^2 \right)
\]

\[
p_k(x) = \frac{\pi_k}{\sum_{l=1}^{K} \pi_l} \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{1}{2\sigma^2} (x - \mu_k)^2 \right) \cdot \frac{\pi_l}{\sum_{l=1}^{K} \pi_l} \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{1}{2\sigma^2} (x - \mu_l)^2 \right).
\]

Assumption: \( \sigma_1 = \ldots \sigma_k = \sigma \)
LDA decision boundary

Pick class $k$ to maximize

$$\delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k)$$

Example: $k = 2$, $\pi_1 = \pi_2$

Classify as class 1 if $x > \frac{\mu_1 + \mu_2}{2\sigma}$

True decision boundary  Estimated decision boundary
LDA in practice

Given training data \((x^{(i)}, y^{(i)}), i = 1, \ldots, n, y^{(i)} \in \{1, \ldots, K\}\)

1. Estimate mean and variance

\[
\hat{\mu}_k = \frac{1}{n_k} \sum_{i: y_i = k} x^{(i)}
\]

\[
\hat{\sigma}^2 = \frac{1}{n - K} \sum_{k=1}^{K} \sum_{i: y_i = k} (x^{(i)} - \hat{\mu}_k)^2
\]

2. Estimate prior

\[
\hat{\pi}_k = \frac{n_k}{n}.
\]

Given testing point \(x\), predict \(k\) that maximizes:

\[
\hat{\delta}_k(x) = x \cdot \frac{\hat{\mu}_k}{\hat{\sigma}^2} - \frac{\hat{\mu}_k^2}{2\hat{\sigma}^2} + \log(\hat{\pi}_k)
\]
LDA vs Logistic Regression

• Logistic regression computes directly $\Pr[Y = 1|X = x]$ by assuming sigmoid function
  – Uses Maximum Likelihood Estimation
  – Discriminative Model
• LDA uses Bayes Theorem to estimate it
  – Estimates mean, co-variance, and prior from training data
  – Generative model
  – Assumes Gaussian distribution for $f_k(x) = \Pr[X = x|Y = k]$
• Which one is better?
  – LDA can be sensitive to outliers
  – LDA works well for Gaussian distribution
  – Logistic regression is more complex to solve, but more expressive
Lab

> library(ISLR)
> fix(Smarket)

<table>
<thead>
<tr>
<th>Year</th>
<th>Lag1</th>
<th>Lag2</th>
<th>Lag3</th>
<th>Lag4</th>
<th>Lag5</th>
<th>Volume</th>
<th>Today</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.381</td>
<td>-0.192</td>
<td>-2.624</td>
<td>-1.055</td>
<td>5.01</td>
<td>1.1913</td>
<td>0.959</td>
<td>Up</td>
</tr>
<tr>
<td>2</td>
<td>0.959</td>
<td>0.381</td>
<td>-0.192</td>
<td>-2.624</td>
<td>-1.055</td>
<td>1.2965</td>
<td>1.032</td>
<td>Up</td>
</tr>
<tr>
<td>3</td>
<td>1.032</td>
<td>0.959</td>
<td>0.381</td>
<td>-0.192</td>
<td>-2.624</td>
<td>1.4112</td>
<td>-0.623</td>
<td>Down</td>
</tr>
<tr>
<td>4</td>
<td>-0.623</td>
<td>1.032</td>
<td>0.959</td>
<td>0.381</td>
<td>-0.192</td>
<td>1.276</td>
<td>0.614</td>
<td>Up</td>
</tr>
<tr>
<td>5</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.032</td>
<td>0.959</td>
<td>0.381</td>
<td>1.2057</td>
<td>0.213</td>
<td>Up</td>
</tr>
<tr>
<td>6</td>
<td>0.213</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.032</td>
<td>0.959</td>
<td>1.3491</td>
<td>1.392</td>
<td>Up</td>
</tr>
<tr>
<td>7</td>
<td>1.392</td>
<td>0.213</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.032</td>
<td>1.445</td>
<td>-0.403</td>
<td>Down</td>
</tr>
<tr>
<td>8</td>
<td>-0.403</td>
<td>1.392</td>
<td>0.213</td>
<td>0.614</td>
<td>-0.623</td>
<td>1.4078</td>
<td>0.027</td>
<td>Up</td>
</tr>
<tr>
<td>9</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.392</td>
<td>0.213</td>
<td>0.614</td>
<td>1.164</td>
<td>1.303</td>
<td>Up</td>
</tr>
<tr>
<td>10</td>
<td>1.303</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.392</td>
<td>0.213</td>
<td>1.2326</td>
<td>0.287</td>
<td>Up</td>
</tr>
<tr>
<td>11</td>
<td>0.287</td>
<td>1.303</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.392</td>
<td>1.309</td>
<td>-0.498</td>
<td>Down</td>
</tr>
<tr>
<td>12</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.303</td>
<td>0.027</td>
<td>-0.403</td>
<td>1.258</td>
<td>-0.189</td>
<td>Down</td>
</tr>
<tr>
<td>13</td>
<td>-0.189</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.303</td>
<td>0.027</td>
<td>1.098</td>
<td>0.68</td>
<td>Up</td>
</tr>
<tr>
<td>14</td>
<td>0.68</td>
<td>-0.189</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.303</td>
<td>1.0531</td>
<td>0.701</td>
<td>Up</td>
</tr>
<tr>
<td>15</td>
<td>0.701</td>
<td>0.68</td>
<td>-0.189</td>
<td>-0.498</td>
<td>0.287</td>
<td>1.1498</td>
<td>-0.562</td>
<td>Down</td>
</tr>
</tbody>
</table>
Lab Logistic Regression

Train on data before 2005

```r
> train=(Year<2005)
> Smarket.2005=Smarket[!train,]
> dim(Smarket.2005)
[1] 252 9
> Direction.2005=Direction[!train]
> glm.fits=glm(Direction~Lag1+Lag2+Lag3+Lag4+Lag5+Volume, data=Smarket, family=binomial, subset=train)
> summary(glm.fits)

Call:
glm(formula = Direction ~ Lag1 + Lag2 + Lag3 + Lag4 + Lag5 +
    Volume, family = binomial, data = Smarket, subset = train)

Deviance Residuals:
     Min      1Q  Median      3Q     Max
-1.302 -1.190  1.079  1.160  1.350

Coefficients:
                     Estimate  Std. Error   z value     Pr(>|z|)
(Intercept)  0.191213   0.333690    0.573 0.567
Lag1         -0.054178   0.051785   -1.046 0.295
Lag2         -0.045805   0.051797   -0.884 0.377
Lag3          0.007200   0.051644    0.139 0.889
Lag4          0.006441   0.051706    0.125 0.901
Lag5         -0.004223   0.051138   -0.083 0.934
Volume      -0.116257   0.239618   -0.485 0.628
```
Lab Logistic Regression

Test on data in 2005

```r
> glm.probs=predict(glm.fits, Smarket.2005, type="response")
> glm.pred=rep("Down", nrow(Smarket.2005))
> glm.pred[glm.probs>.5]="Up"
> head(Smarket.2005)
     Year Lag1  Lag2  Lag3  Lag4  Lag5 Volume Today Direction
999 2005  -0.134  0.008 -0.007  0.715  -0.431  0.7869   -0.812    Down
1000 2005  -0.812 -0.134  0.008  -0.007  0.715  1.5108   -1.167    Down
1001 2005  -1.167 -0.812 -0.134  0.008  -0.007  1.7210   -0.363    Down
1002 2005  -0.363 -1.167 -0.812 -0.134  0.008  1.7389    0.351     Up
1003 2005   0.351 -0.363 -1.167 -0.812 -0.134  1.5691  -0.143    Down
1004 2005  -0.143  0.351 -0.363 -1.167 -0.812  1.4779    0.342     Up
> head(glm.probs)
   999 1000 1001 1002 1003 1004
0.5282195 0.5156688 0.5226521 0.5138543 0.4983345 0.5010912
> head(glm.pred)
[1] "Up"  "Up"  "Up"  "Up"  "Down" "Up"
> table(glm.pred, Direction.2005)
      Direction.2005
glm.pred     Down Up
  Down     77  97
  Up       34  44
> mean(glm.pred==Direction.2005)
[1] 0.4801587
```
Lab LDA

```r
> library(MASS)
> lda.fit=lda(Direction~Lag1+Lag2, data=Smarket, subset=train)
> lda.fit
Call:
lda(Direction ~ Lag1 + Lag2, data = Smarket, subset = train)

Prior probabilities of groups:
       Down    Up
0.491984 0.508016

Group means:
   Lag1     Lag2
Down 0.04279022 0.03389409
Up   -0.03954635 -0.03132544

Coefficients of linear discriminants:

LD1
Lag1 -0.6420190
Lag2 -0.5135293

> lda.pred=predict(lda.fit, Smarket.2005)
> lda.class=lda.pred$class
> table(lda.class, Direction.2005)
Direction.2005
lda.class Down    Up
    Down   35   35
    Up     76  106

> mean(lda.class==Direction.2005)
[1] 0.5595238
```
Lab kNN

```r
> library(class)
> train.X=cbind(Lag1,Lag2)[train,]
> test.X=cbind(Lag1,Lag2)[!train,]
> train.Direction=Direction[train]
> set.seed(1)
> knn.pred=knn(train.X,test.X,train.Direction,k=1)
> table(knn.pred,Direction.2005)
      Direction.2005
   knn.pred Down Up
     Down   43  58
     Up     68  83
> mean(knn.pred==Direction.2005)
[1] 0.5
> knn.pred=knn(train.X,test.X,train.Direction,k=3)
> table(knn.pred,Direction.2005)
      Direction.2005
   knn.pred Down Up
     Down   48  54
     Up     63  87
> mean(knn.pred==Direction.2005)
[1] 0.5357143
> knn.pred=knn(train.X,test.X,train.Direction,k=7)
> table(knn.pred,Direction.2005)
      Direction.2005
   knn.pred Down Up
     Down   41  65
     Up     70  76
> mean(knn.pred==Direction.2005)
[1] 0.4642857
```
Linear models

- Perceptron

\[ h(x) = \text{sign}(\theta^T x) \]

- Logistic regression

\[ h_\theta(x) = \frac{1}{1 + e^{-\theta^T x}} \]

- LDA

\[ \max_k \delta_k(x) = x \cdot \frac{\mu_k}{\sigma^2} - \frac{\mu_k^2}{2\sigma^2} + \log(\pi_k) \]
Supervised Learning

Training

Data → Pre-processing → Feature extraction → Learning model
- Labeled
- Normalization
- Feature Selection
- Classification Regression

Testing

New data → Learning model → Predictions
- Unlabeled
- Healthy Sick Classification
- Price Risk score Regression

15
Example: Email Classification

- **Input:** a email message
- **Output:** is the email...
  - spam,
  - work-related,
  - personal, ...
Bag-of-Words

- Input: $\mathbf{x}$ (email-valued)
- Feature vector:

$$f(\mathbf{x}) = \begin{bmatrix} f_1(\mathbf{x}) \\ f_2(\mathbf{x}) \\ \vdots \\ f_n(\mathbf{x}) \end{bmatrix}, \quad \text{e.g. } f_1(\mathbf{x}) = \begin{cases} 1 & \text{if the email contains} \\ 0 & \text{otherwise} \end{cases}$$

- Learn one weight vector for each class:

$$w_y \in \mathbb{R}^n, \quad y \in \{\text{SPAM, WORK, PERS}\}$$

Could also use frequency
- $f_i(\mathbf{x})$ is the number of times word $i$ appears in $\mathbf{x}$
• Large number of words in dictionary (>50,000)
• Very sparse representation (many features set at 0)
Feature selection

• *Feature Selection*
  • Process for choosing an optimal subset of features according to a certain criteria

• Why we need Feature Selection:
  1. To improve performance (in terms of speed, predictive power, simplicity of the model).
  2. To visualize the data for model selection.
  3. To reduce dimensionality and remove noise.
Feature Search Space

Exponentially large!
Methods for Feature Selection

• **Wrappers**
  – Select subset of features that gives best prediction accuracy (using cross-validation)
  – Model-specific

• **Filters**
  – Compute some statistical metrics (correlation coefficient, mutual information)
  – Select features with statistics higher than threshold

• **Embedded methods**
  – Feature selection done as part of training
  – Example: Regularization (Lasso, L1 regularization)
Feature Engineering

• Feature engineering is crucial to getting good results
• Strategy: overshoot and regularize
  – Define as many features as you can
  – Use regularization for models that support it
  – Use other feature selection methods (e.g., filters) otherwise
• Do cross-validation to evaluate selected features on multiple runs
• When feature selection is frozen, evaluate on test set
Wrappers: Search Strategy

- With an **exhaustive search**

  \[101110000001000100001000000000100101010\]

  With \(d\) features \(\rightarrow \ 2^d\) possible feature subsets.

  20 features … 1 million feature sets to check
  25 features … 33.5 million sets
  30 features … 1.1 billion sets

- Need for a **search strategy**
  - Sequential forward selection
  - Recursive backward elimination
  - Genetic algorithms
  - Simulated annealing
  - ...

23
Wrappers: Sequential Forward Selection

Start with the empty set $S = \emptyset$

While stopping criteria not met

For each feature $X_f$ not in $S$

- Define $S' = S \cup \{X_f\}$
- Train model using the features in $S'$
- Compute the accuracy on validation set

End

$S = S'$ where $S'$ is the feature set with the greatest accuracy

End

Backward feature selection starts with all features and eliminates backward
Search complexity for sequential forward selection

- Evaluates $\frac{d(d+1)}{2}$ features sets instead of $2^d$
Cross Validation

- **k-fold CV**
  - Split data into \( k \) partitions of equal size

- **Leave-one-out CV (LOOCV)**
  - \( k=n \) (validation set only one point)
Filters

**Principle**: replace evaluation of model with quick to compute statistics $J(X_f)$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$J(X_k)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>35</td>
<td>0.846</td>
</tr>
<tr>
<td>42</td>
<td>0.811</td>
</tr>
<tr>
<td>10</td>
<td>0.810</td>
</tr>
<tr>
<td>654</td>
<td>0.611</td>
</tr>
<tr>
<td>22</td>
<td>0.443</td>
</tr>
<tr>
<td>59</td>
<td>0.388</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>212</td>
<td>0.09</td>
</tr>
<tr>
<td>39</td>
<td>0.05</td>
</tr>
</tbody>
</table>

For each feature $X_f$
- Compute $J(X_f)$

End
Rank features according to $J(X_f)$
Choose manual cut-off point

Examples of filtering criterion
- The mutual information with the target variable $J(X_f) = I(X_f; Y)$
- The correlation with the target variable
- $\chi^2$ - statistic
Search Complexity for Filter Methods

Pros:
- A lot less expensive!

Cons:
- Not model-oriented
Embedded methods: Regularization

Lasso regression

\[ J(\theta) = \sum_{i=1}^{n} (h_\theta(x^{(i)}) - y^{(i)})^2 + \lambda \sum_{j=1}^{d} |\theta_j| \]

- L1 norm for regularization
- No closed form solution
- Algorithms based on gradient descent or quadratic programming
Embedded methods: Regularization

**Principle:** the classifier performs feature selection as part of the learning procedure

**Example:** the logistic LASSO (Tibshirani, 1996)

\[
f(x) = \frac{1}{1 + e^{-(w^T x)}} = P(Y = 1 | x)
\]

With Error Function:

\[
E = - \sum_{i=1}^{N} \left\{ y_i \log f(x_i) + (1 - y_i) \log(1 - f(x_i)) \right\} + \lambda \sum_{f=1}^{d} |w_f|
\]

- Cross-entropy error
- Regularizing term

**Pros:**
- Performs feature selection as part of learning the procedure

**Cons:**
- Computationally demanding
Summary: Feature Selection

- Filtering
- $L_1$ regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

Computational cost
Summary: Feature Selection

- **Filtering**
  - $L_1$ regularization (embedded methods)
- Wrappers
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Good preprocessing step
- Fails to capture relationship between features

Computational cost
Summary: Feature Selection

- Can add regularization in optimization objective
- Can be solved with Gradient Descent
- Can be applied to many models (e.g., linear or logistic regression)
- Can not be applied to all methods (e.g., kNN)
Summary: Feature Selection

- Filtering
- $L_1$ regularization (embedded methods)
- *Wrappers*
  - Forward selection
  - Backward selection
  - Other search
  - Exhaustive

- Most directly optimize prediction performance
- Can be very expensive, even with greedy search methods
- Cross-validation is a good objective function to start with
Outline

• Linear Discriminant Analysis (LDA)
• Lab (logistic regression, LDA, kNN)
• Feature selection
  – Wrapper
  – Filter
  – Embedded methods
• Decision trees
  – Information Gain
Sample Dataset

- Columns denote features $X_i$
- Rows denote labeled instances $\langle x^{(i)}, y^{(i)} \rangle$
- Class label denotes whether a tennis game was played

<table>
<thead>
<tr>
<th>Outlook</th>
<th>Temperature</th>
<th>Humidity</th>
<th>Wind</th>
<th>Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Hot</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>No</td>
</tr>
<tr>
<td>Overcast</td>
<td>Cool</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>High</td>
<td>Weak</td>
<td>No</td>
</tr>
<tr>
<td>Sunny</td>
<td>Cool</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Sunny</td>
<td>Mild</td>
<td>Normal</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>Yes</td>
</tr>
<tr>
<td>Overcast</td>
<td>Hot</td>
<td>Normal</td>
<td>Weak</td>
<td>Yes</td>
</tr>
<tr>
<td>Rain</td>
<td>Mild</td>
<td>High</td>
<td>Strong</td>
<td>No</td>
</tr>
</tbody>
</table>

Categorical data
Decision Tree

- A possible decision tree for the data:

  - Each internal node: test one attribute $X_i$
  - Each branch from a node: selects one value for $X_i$
  - Each leaf node: predict $Y$ (or $p(Y \mid x \in \text{leaf})$)
• A possible decision tree for the data:

```
Decision Tree

• What prediction would we make for
  <outlook=sunny, temperature=hot, humidity=high, wind=weak>?
```

Decision Tree:
- Outlook
  - Sunny
  - Overcast
  - Rain
    - Wind
      - Strong
      - Weak
    - Humidity
      - High
      - Normal
    - No
      - Yes
Decision Tree Learning

Problem Setting:

- Set of possible instances $X$
  - each instance $x$ in $X$ is a feature vector
  - e.g., $\langle \text{Humidity}=\text{low}, \text{Wind}=\text{weak}, \text{Outlook}=\text{rain}, \text{Temp}=\text{hot} \rangle$
- Unknown target function $f : X \rightarrow Y$
  - $Y$ is discrete valued
- Set of function hypotheses $H=\{ h \mid h : X \rightarrow Y \}$
  - each hypothesis $h$ is a decision tree
  - trees sorts $x$ to leaf, which assigns $y$
Expressiveness

- Decision trees can represent any boolean function of the input attributes.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>A xor B</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</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>F</td>
</tr>
</tbody>
</table>

- In the worst case, the tree will require exponentially many nodes.

XOR cannot be learned with linear classifiers.
Occam’s Razor

• Principle stated by William of Ockham (1285-1347)
  – “non sunt multiplicanda entia praeter necessitatem”
  – entities are not to be multiplied beyond necessity
  – AKA Occam’s Razor, Law of Economy, or Law of Parsimony

Idea: The simplest consistent explanation is the best

• Therefore, the smallest decision tree that correctly classifies all of the training examples is best
  • Finding the provably smallest decision tree is NP-hard
  • ...So instead of constructing the absolute smallest tree consistent with the training examples, construct one that is pretty small
Learning Decision Trees

- 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
Key Idea: Use Recursion Greedily

mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
0 0
Predict bad

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

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)
Full Tree

A full tree

mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
cylinders = 4
cylinders = 5
cylinders = 6
cylinders = 8

0 0
4 17
1 0
8 0
9 1

Predict bad
Predict bad
Predict bad
Predict bad

pchance = 0.135
pchance = 0.085

maker = america
maker = asia
maker = europe
horsepower = low
horsepower = medium
horsepower = high

0 10
2 5
2 2
0 0
0 1
9 0

Predict good
Predict good
Predict bad

pchance = 0.317
pchance = 0.717

horsepower = low
horsepower = medium
horsepower = high
acceleration = low
acceleration = medium
acceleration = high

0 4
2 1
0 0
1 0
1 1
0 1

Predict good
Predict bad
Predict bad

pchance = 0.894
pchance = 0.717

acceleration = low
acceleration = medium
acceleration = high

1 0
1 1
0 0

Predict bad
Predict bad
Predict bad

Predict bad

Predict good

modelyear = 70to74
modelyear = 75to78
modelyear = 79to83

0 0
0 1
1 0

Predict bad
Predict bad
Predict bad
Predict bad
Splitting

Would we prefer to split on $X_1$ or $X_2$?

Idea: use counts at leaves to define probability distributions, so we can measure uncertainty!

Use entropy-based measure (Information Gain)
Acknowledgements

• Slides made using resources from:
  – Andrew Ng
  – Eric Eaton
  – David Sontag

• Thanks!