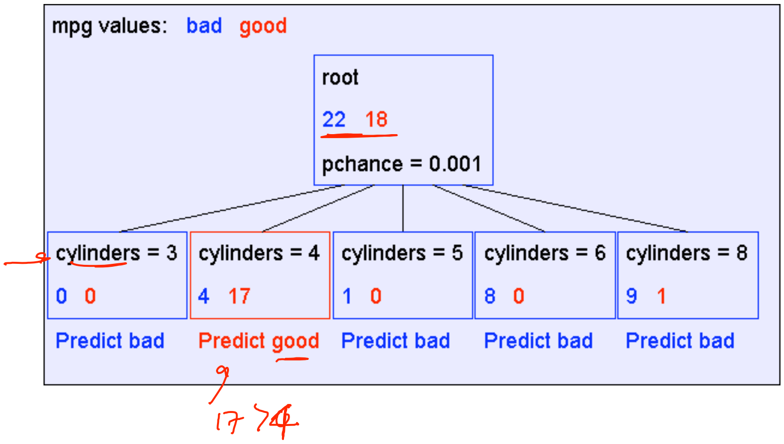
Decision Trees, cont.

Boosting

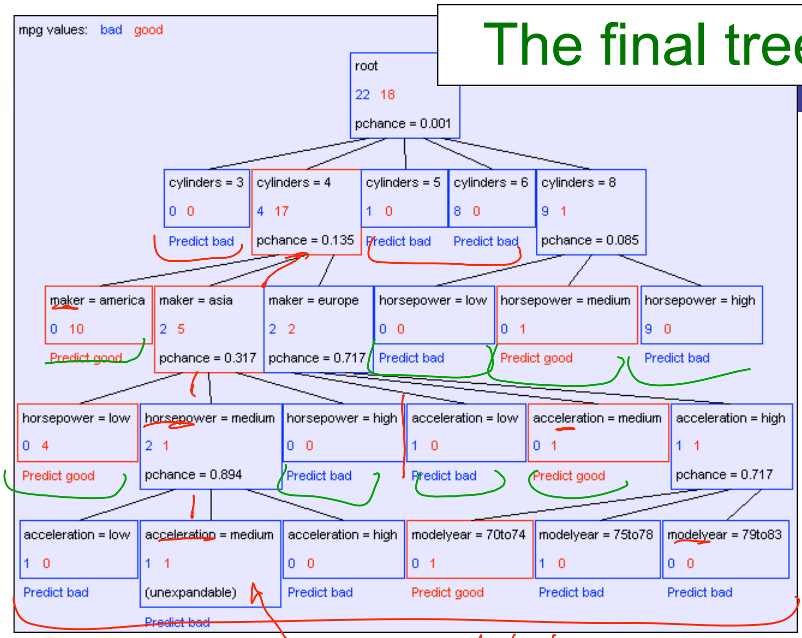
Machine Learning – 10701/15781
Carlos Guestrin
Carnegie Mellon University

October 1st, 2007

A Decision Stump



The final tree



mpg values: bad good

root
22 18
pchance = 0.001

cylinders = 3
0 0
Predict bad

cylinders = 4
4 17
pchance = 0.135

cylinders = 5
1 0
Predict bad

cylinders = 6
8 0
Predict bad

cylinders = 8
9 1
pchance = 0.085

maker = america
0 10
Predict good

maker = asia
2 5
pchance = 0.317

maker = europe
2 2
pchance = 0.717

horsepower = low
0 0
Predict bad

horsepower = medium
0 1
Predict good

horsepower = high
9 0
Predict bad

horsepower = low
0 4
Predict good

horsepower = medium
2 1
pchance = 0.894

horsepower = high
0 0
Predict bad

acceleration = low
1 0
Predict bad

acceleration = medium
0 1
Predict good

acceleration = high
1 1
pchance = 0.717

acceleration = low
1 0
Predict bad

acceleration = medium
1 1
(unexpandable)
Predict bad

acceleration = high
0 0
Predict bad

modelyear = 70to74
0 1
Predict good

modelyear = 75to78
1 0
Predict bad

modelyear = 79to83
0 0
Predict bad

no more distinctions

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"

base cases

- 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

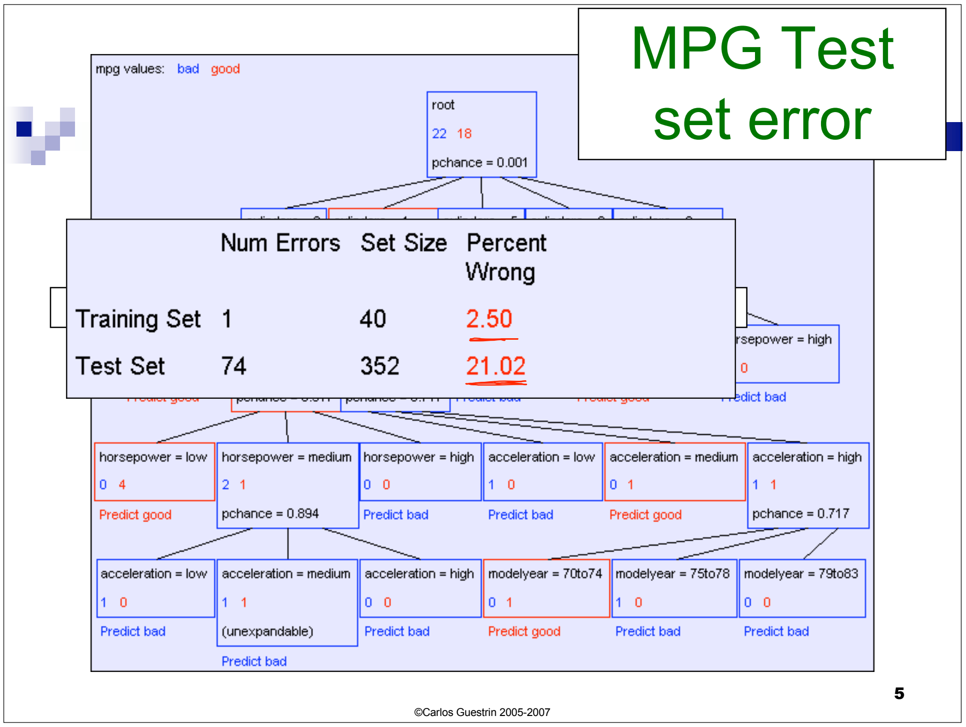
    BuildTree(DS_i,*Output*)

  Where DS_i built consists of all those records in DataSet for which X = *i*th distinct value of X.

recurse

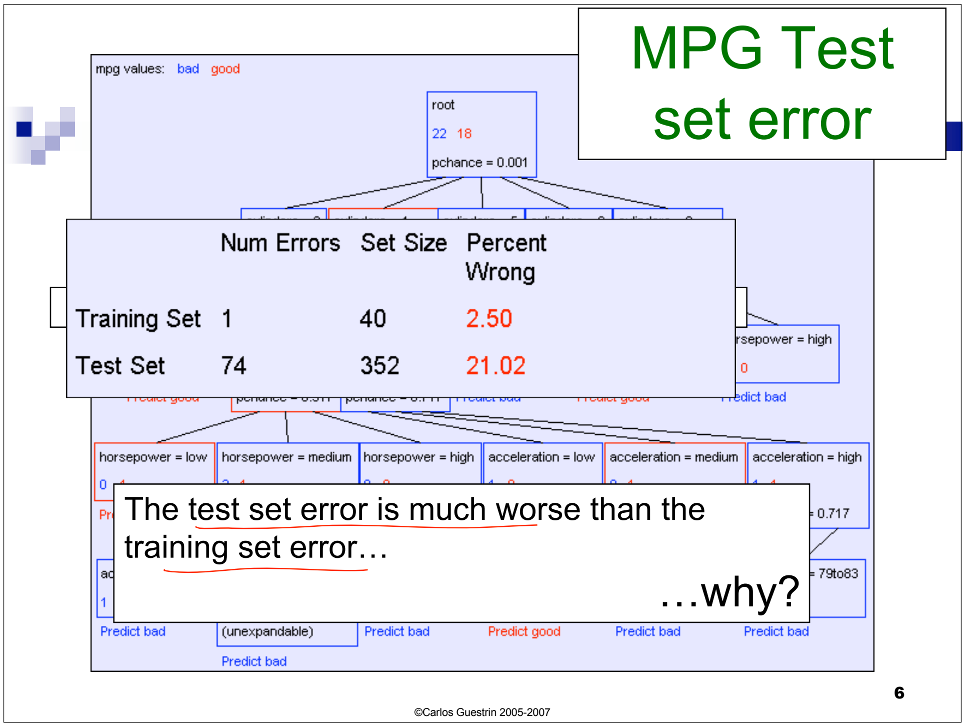
recurse on children

pick part of dataset consistent with child



MPG Test set error

	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02



MPG Test set error

	Num Errors	Set Size	Percent Wrong
Training Set	1	40	2.50
Test Set	74	352	21.02

The test set error is much worse than the training set error…

…why?

Decision trees & Learning Bias

mpg	cylinders	displacement	horsepower	weight	acceleration	modelyear	maker
good	4	low	low	low	high	75to78	asia
bad	6	medium	medium	medium	medium	70to74	america
bad	4	medium	medium	medium	low	75to78	europe
bad	8	high	high	high	low	70to74	america
bad	6	medium	medium	medium	medium	70to74	america
bad	4	low	medium	low	medium	70to74	asia
bad	4	low	medium	low	low	70to74	asia
bad	8	high	high	high	low	75to78	america
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
:	:	:	:	:	:	:	:
bad	8	high	high	high	low	70to74	america
good	8	high	medium	high	high	79to83	america
bad	8	high	high	high	low	75to78	america
good	4	low	low	low	low	79to83	america
bad	6	medium	medium	medium	high	75to78	america
good	4	medium	low	low	low	79to83	america
good	4	low	low	medium	high	79to83	america
bad	8	high	high	high	low	70to74	america
good	4	low	medium	low	medium	75to78	europe
bad	5	medium	medium	medium	medium	75to78	europe

Can fit training set perfectly

i.e., $\ell_{train} = 0$

no label noise:

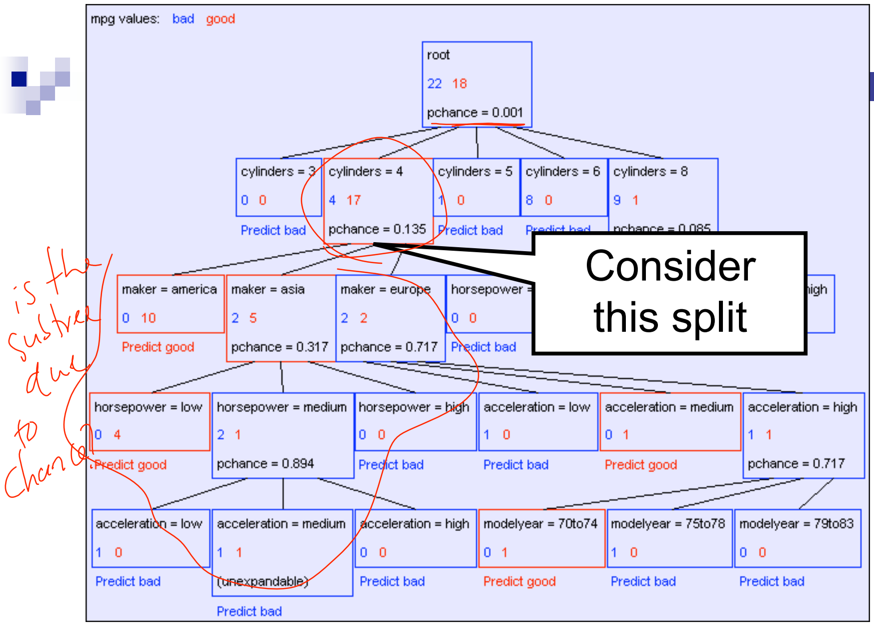
if x^1 & x^2

$\therefore \underbrace{x^1 = x^2}_{\text{agree on features}} \Rightarrow \underbrace{y^1 = y^2}_{\text{agree on labels}}$

very complex tree

Decision trees will overfit

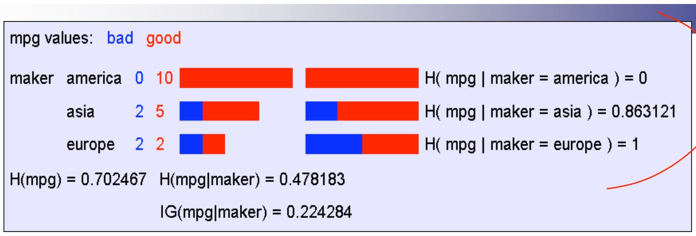
- Standard decision trees are have no learning bias~~ed~~
  - □ Training set error is always zero!
    - (If there is no label noise)
  - □ Lots of variance
  - □ Will definitely overfit!!!
  - □ Must bias towards simpler trees
- Many strategies for picking simpler trees:
  - □ Fixed depth
  - □ Fixed number of leaves
  - □ Or something smarter…



Consider this split

is the subtree due to chance

A chi-square test



mpg values: bad good

maker america 0 10 H(mpg | maker = america) = 0

asia 2 5 H(mpg | maker = asia) = 0.863121

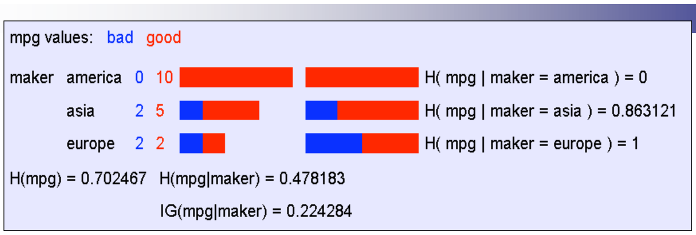
europe 2 2 H(mpg | maker = europe) = 1

H(mpg) = 0.702467 H(mpg|maker) = 0.478183

IG(mpg|maker) = 0.224284

- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

A chi-square test



mpg values: bad good

maker america 0 10 H(mpg | maker = america) = 0

asia 2 5 H(mpg | maker = asia) = 0.863121

europe 2 2 H(mpg | maker = europe) = 1

H(mpg) = 0.702467 H(mpg|maker) = 0.478183

IG(mpg|maker) = 0.224284

- Suppose that mpg was completely uncorrelated with maker.
- What is the chance we'd have seen data of at least this apparent level of association anyway?

By using a particular kind of chi-square test, the answer is 7.2%

(Such simple hypothesis tests are very easy to compute, unfortunately, not enough time to cover in the lecture,
but in your homework, you'll have fun! :))

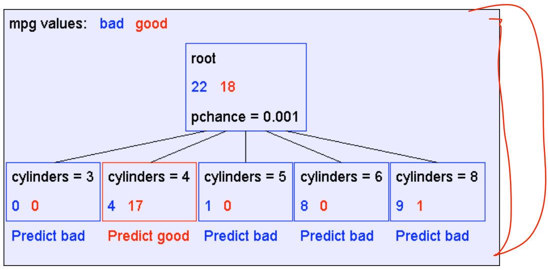
Using Chi-squared to avoid overfitting

- Build the full decision tree as before
- But when you can grow it no more, start to prune:
  - □ Beginning at the bottom of the tree, delete splits in which $p_{chance} > MaxPchance$
  - □ Continue working you way up until there are no more prunable nodes

MaxPchance is a magic parameter you must specify to the decision tree, indicating your willingness to risk fitting noise

Pruning example

- With MaxPchance = 0.1, you will see the following MPG decision tree:



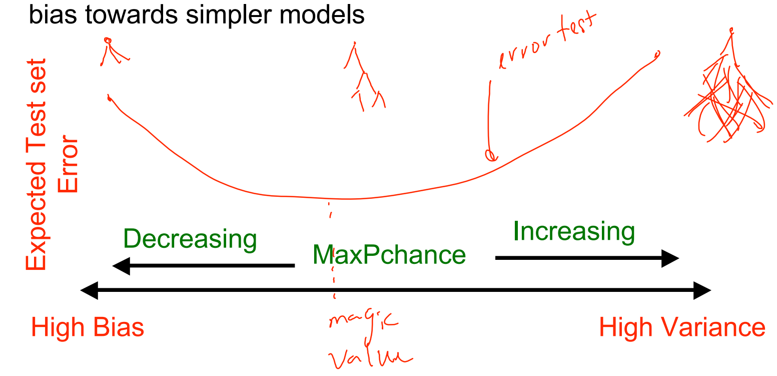
decision stump

Note the improved test set accuracy compared with the unpruned tree

	Num Errors	Set Size	Percent Wrong
Training Set	5	40	12.50
Test Set	56	352	15.91

MaxPchance

- Technical note MaxPchance is a regularization parameter that helps us bias towards simpler models



Expected Test set Error

Decreasing

MaxPchance

Increasing

High Bias

High Variance

We'll learn to choose the value of these magic parameters soon!

Real-Valued inputs

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

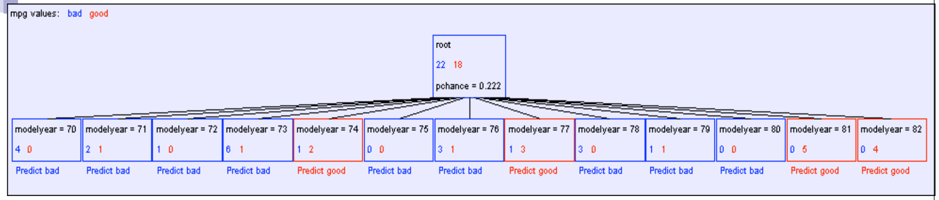
mpg	cylinders	displacemen	horsepower	weight	acceleration	modelyear	maker
good	4	97	75	2265	18.2	77	asia
bad	6	199	90	2648	15	70	america
bad	4	121	110	2600	12.8	77	europe
bad	8	350	175	4100	13	73	america
bad	6	198	95	3102	16.5	74	america
bad	4	108	94	2379	16.5	73	asia
bad	4	113	95	2228	14	71	asia
bad	8	302	139	3570	12.8	78	america
:	:	:	:	:	:		:
:	:	:	:	:	:		:
:	:	:	:	:	:		:
good	4	120	79	2625	18.6	82	america
bad	8	455	225	4425	10	70	america
good	4	107	86	2464	15.5	76	europe
bad	5	131	103	2830	15.9	78	europe

Infinite number of possible split values!!!

Finite dataset, only finite number of relevant splits!

Idea One: Branch on each possible real value

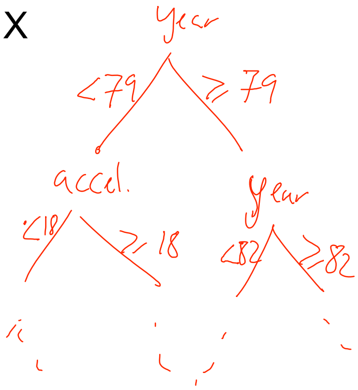
“One branch for each numeric value” idea:



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 ≥ t



Choosing threshold split

- Binary tree, split on attribute X
  - □ One branch: X < t
  - □ Other branch: X ≥ 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)*

  ← threshold

- Define *H(Y|X:t)* =

  year < t, year, year

  H(Y|X < t) P(X < t) + H(Y|X >= t) P(X >= 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*

naive implementation O(m^2) for m data points

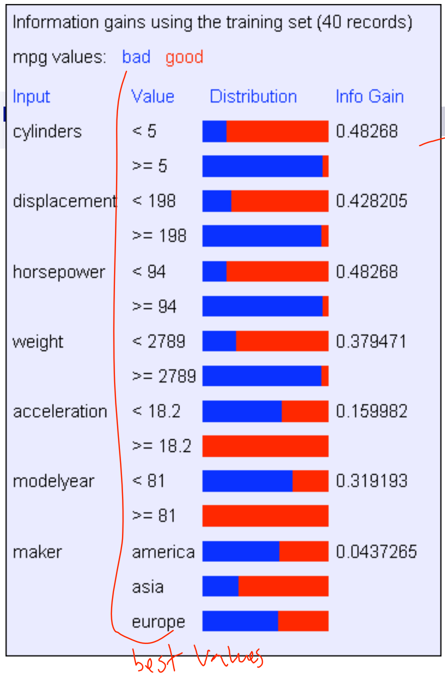
Dynamic program O(m)

- 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

  then pick best $X_i = \arg\max_i IG^*(Y|X_i)$

- Note, may split on an attribute multiple times, with different thresholds

Example with MPG

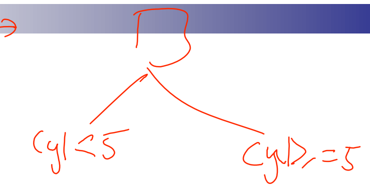


Information gains using the training set (40 records)

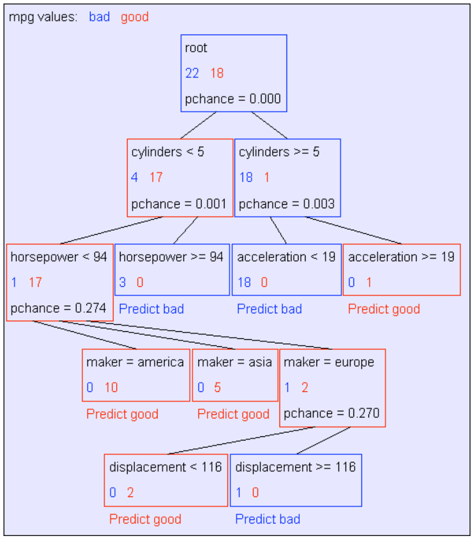
mpg values: bad good

Input	Value	Distribution	Info Gain
cylinders	< 5	🟦🟥	0.48268
	>= 5	🟦🟥	
displacement	< 198	🟦🟥	0.428205
	>= 198	🟦🟥	
horsepower	< 94	🟦🟥	0.48268
	>= 94	🟦🟥	
weight	< 2789	🟦🟥	0.379471
	>= 2789	🟦🟥	
acceleration	< 18.2	🟦🟥	0.159982
	>= 18.2	🟥	
modelyear	< 81	🟦🟥	0.319193
	>= 81	🟥	
maker	america	🟦🟥	0.0437265
	asia	🟦🟥	
	europe	🟦🟥	

best values



Example tree using reals



mpg values: bad good

root
22 18
pchance = 0.000

cylinders < 5
4 17
pchance = 0.001

cylinders >= 5
18 1
pchance = 0.003

horsepower < 94
1 17
pchance = 0.274

horsepower >= 94
3 0
Predict bad

acceleration < 19
18 0
Predict bad

acceleration >= 19
0 1
Predict good

maker = america
0 10
Predict good

maker = asia
0 5
Predict good

maker = europe
1 2
pchance = 0.270

displacement < 116
0 2
Predict good

displacement >= 116
1 0
Predict bad

What you need to know about decision trees

- Decision trees are one of the most popular data mining tools
  - □ Easy to understand “interpretable”
  - □ 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
- Decision trees will overfit!!!
  - □ Zero bias classifier → Lots of variance
  - □ Must use tricks to find “simple trees”, e.g.,
    - Fixed depth/Early stopping
    - Pruning
    - Hypothesis testing

Acknowledgements

- Some of the material in the decision trees presentation is courtesy of Andrew Moore, from his excellent collection of ML tutorials:
  - □ http://www.cs.cmu.edu/~awm/tutorials

Announcements

- Homework 1 due Wednesday beginning of class
  - □ started early, started early, started early, started early, started early, started early, started early, started early

- Exam dates set:
  - □ Midterm: Thursday, Oct. 25th, 5-6:30pm, MM A14 this room
  - □ Final: Tuesday, Dec. 11, 05:30PM-08:30PM somewhere

Fighting the bias-variance tradeoff

- **Simple (a.k.a. weak) learners are good**
  - □ e.g., naïve Bayes, logistic regression, decision stumps (or shallow decision trees)
  - □ Low variance, don't usually overfit
- **Simple (a.k.a. weak) learners are bad**
  - □ High bias, can't solve hard learning problems
- Can we make weak learners always good???
  - □ **No!!!**
  - □ **But often yes…**

Voting (Ensemble Methods)

- Instead of learning a single (weak) classifier, learn **many weak classifiers** that are **good at different parts of the input space**
- **Output class:** (Weighted) vote of each classifier
  - □ Classifiers that are most "sure" will vote with more conviction
  - □ Classifiers will be most "sure" about a particular part of the space
  - □ On average, do better than single classifier!

$H(X): X \mapsto Y$

$Y \in \{-1, +1\}$

weight

$$H(X) = \text{Sign}\left\{ \sum_{t=1}^{T} \alpha_t h_t(x) \right\}$$

Simple learners:

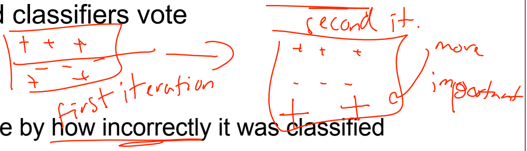
$h_t(x): X \to \{-1, +1\}$

$\to [-1, +1]$

- **But how do you ???**
  - □ force classifiers to learn about different parts of the input space?
  - □ weigh the votes of different classifiers?

Boosting [Schapire, 1989]

- Idea: given a weak learner, run it multiple times on (reweighted) training data, then let learned classifiers vote



- On each iteration *t*:
  - weight each training example by how incorrectly it was classified
  - Learn a hypothesis – h_t
  - A strength for this hypothesis – α_t
- Final classifier:

$$H(x) = \text{sign}\left\{ \sum_{t=1}^{T} \alpha_t h_t(x) \right\}$$

- **Practically useful**
- **Theoretically interesting**

Learning from weighted data

- **Sometimes not all data points are equal**
  - Some data points are more equal than others
- **Consider a weighted dataset**
  - D(i) – weight of *i* th training example ($\mathbf{x}^i$,y^i)
  - Interpretations:
    - *i* th training example counts as D(i) examples
    - If I were to “resample” data, I would get more samples of “heavier” data points
- **Now, in all calculations, whenever used, *i* th training example counts as D(i) “examples”**
  - e.g., MLE for Naïve Bayes, redefine *Count(Y=y)* to be weighted count

Prior

$$\hat{P}(Y=y) = \frac{Count(Y=y)}{m}$$

weighted :

indicator

$$\hat{P}(Y=y) = \frac{\sum_{i=1}^{m} D(i)\, \mathbb{I}(y^i=y)}{\sum_{i=1}^{m} D(i)}$$

AdaBoost

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$ data

$e^{-a} < 1$ if $a > 0$

$e^{-a} > 1$ if $a < 0$

Initialize $D_1(i) = 1/m$. ← uniform

For $t = 1, \ldots, T$: ← iteration

- Train weak base learner using distribution D_t. (weights) ← as in previous slide
- Get weak base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$.
- Update:

weight before

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

normalizer $\sum_i D_{t+1}(i) = 1$

where Z_t is a normalization factor

$$Z_t = \sum_{i=1}^{m} D_t(i) \exp(-\alpha_t y_i h_t(x_i))$$

Output the final classifier:

$$H(x) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right).$$

Figure 1: The boosting algorithm AdaBoost.

if normalizer $Z_t = 1$

if $y_i = +1$

$\alpha_t > 0$

if $h_t(x_i)$ is correct:

$\Rightarrow h_t(x_i) > 0$

$\Rightarrow -\alpha_t y_i h_t(x_i) < 0$

$\Rightarrow D_{t+1}(i)$ reduced

if $h_t(x_i)$ is incorrect

$\Rightarrow -\alpha_t y_i h_t(x_i) > 0$

$\Rightarrow D_{t+1}(i)$ increase

Given: $(x_1, y_1), \ldots, (x_m, y_m)$ where $x_i \in X, y_i \in Y = \{-1, +1\}$

Initialize $D_1(i) = 1/m$.

For $t = 1, \ldots, T$:

- Train base learner using distribution D_t.
- Get base classifier $h_t : X \to \mathbb{R}$.
- Choose $\alpha_t \in \mathbb{R}$. ←

$$\boxed{\alpha_t = \frac{1}{2} \ln\left(\frac{1 - \epsilon_t}{\epsilon_t}\right)}$$

- Update:

$$D_{t+1}(i) = \frac{D_t(i) \exp(-\alpha_t y_i h_t(x_i))}{Z_t}$$

ε_t is weighted error of $h_t(x)$ at iteration t:

$$\boxed{\epsilon_t = P_{i \sim D_i}[\mathbf{x}^i \neq y^i]}$$

$$\boxed{\epsilon_t = \frac{1}{\sum_{i=1}^{m} D_t(i)} \sum_{i=1}^{m} D_t(i) \delta(h_t(x_i) \neq y_i)}$$

as $\varepsilon_t \to 0$, $\alpha_t \to +\infty$ really trust

as $\varepsilon_t \to 1$, $\alpha_t \to -\infty$ ← really trust opposite

if $\varepsilon_t = 0.5$, $\alpha_t = 0$, random classifiers are bad ⇒ zero weight

What α_t to choose for hypothesis h_t?

[Schapire, 1989]

Training error of final classifier is bounded by:

bound (count # mistakes in training)

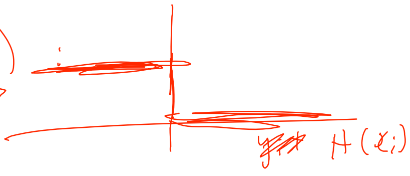
$$\text{error}_{train}(H) = \frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i))$$

Where $f(x) = \sum_t \alpha_t h_t(x);\ H(x) = sign(f(x))$

if $y_i = +1$

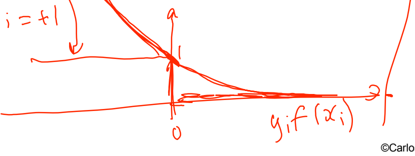
implies

$\delta(H(x_i) \neq y_i)$:



$\forall i \quad \delta(H(x_i) \neq y_i) \leq e^{-y_i f(x_i)}$

$y_i = +1$



What α_t to choose for hypothesis h_t?

[Schapire, 1989]

[magic of telescopic sums]

your homework answer

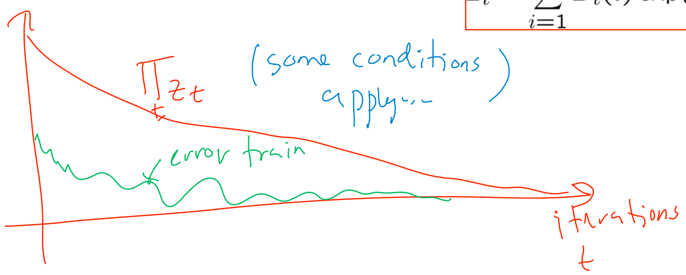
Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \frac{1}{m}\sum_{i=1}^{m}\exp(-y_i f(x_i)) = \prod_{t=1}^{T} Z_t$$

upper bound on train error

Where $f(x) = \sum_{t=1}^{T} \alpha_t h_t(x);\ H(x) = sign(f(x))$

$$\boxed{Z_t = \sum_{i=1}^{m} D_t(i)\exp(-\alpha_t y_i h_t(x_i))}$$



iterations t

What α_t to choose for hypothesis h_t?

[Schapire, 1989]

Training error of final classifier is bounded by:

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i)\leq\frac{1}{m}\sum_i \exp(-y_i f(x_i)) = \prod_t Z_t$$

Where $f(x)=\sum_t \alpha_t h_t(x);\ H(x)=sign(f(x))$

If we minimize $\prod_t Z_t$, we minimize our training error

Z_{t-1} doesn't depend on α_t, h_t

We can tighten this bound greedily, by choosing α_t and h_t on each iteration to minimize Z_t.

$$Z_t=\sum_{i=1}^{m}D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

What α_t to choose for hypothesis h_t?

[Schapire, 1989]

We can minimize this bound by choosing α_t on each iteration to minimize Z_t.

$$Z_t=\sum_{i=1}^{m}D_t(i)\exp(-\alpha_t y_i h_t(x_i))$$

For boolean target function, this is accomplished by [Freund & Schapire '97]:

$$\boxed{\alpha_t=\frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)}$$

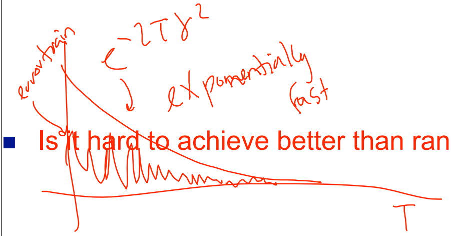
You'll prove this in your homework! ☺

Strong, weak classifiers

- If each classifier is (at least slightly) better than random
  - □ $\varepsilon_t < 0.5$
- AdaBoost will achieve zero *training error* (exponentially fast):

$$\frac{1}{m}\sum_{i=1}^{m}\delta(H(x_i)\neq y_i) \leq \prod_t Z_t \leq \underbrace{\exp\left(-2\sum_{t=1}^{T}(1/2-\epsilon_t)^2\right)}_{} \leq e^{-2T\gamma^2}$$

$(1/2-\varepsilon_t)^2$ ← how much better is ε_t than random

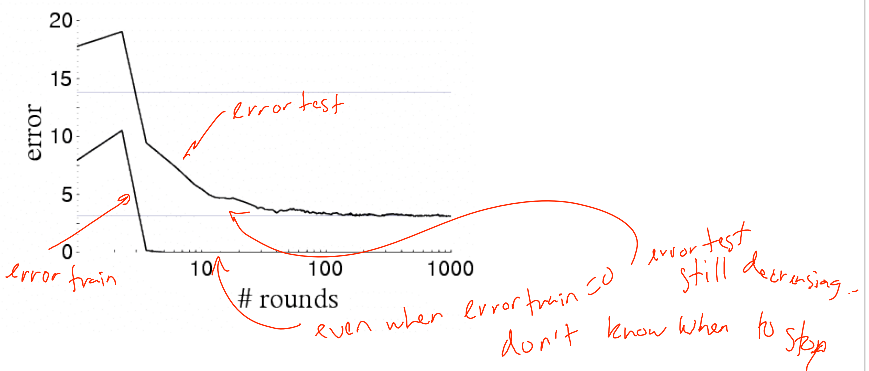


- Is it hard to achieve better than random training error? ε_t

$|1/2-\varepsilon_t| \geq \gamma$

Boosting results – Digit recognition

[Schapire, 1989]



even when error train =0 error test still decreasing.

don't know when to stop

- Boosting often
  - □ Robust to overfitting
  - □ Test set error decreases even after training error is zero