Supervised Learning: Summary So Far

learning as function approximation

k-**NN:** distance function (any attributes), any labels
Neural network: numeric attributes, numeric or binary labels

Regression: incremental training with LMS
3-Layer ANN: train with BackProp

what about discrete attributes and labels?

Decision Trees

Example: WillWait

Decision Trees
- Example
- Construction
- Break

Naive Bayes

Boosting

Example	Alt	Bar	Fri	Hun	Pat	Price	Rain	Res	Type	Est	Goal WillWait
	Attributes										
X_1	Yes	No	No	Yes	Some	$$$	No	Yes	French	0–10	Yes
X_2	Yes	No	No	Yes	Full	$	No	No	Thai	30–60	No
X_3	No	Yes	No	No	Some	$	No	No	Burger	0–10	Yes
X_4	Yes	No	Yes	Yes	Full	$	Yes	No	Thai	10–30	Yes
X_5	Yes	No	Yes	No	Full	$$$	No	Yes	French	>60	No
X_6	No	Yes	No	Yes	Some	$$	Yes	Yes	Italian	0–10	Yes
X_7	No	Yes	No	No	None	$	Yes	No	Burger	0–10	No
X_8	No	No	No	Yes	Some	$$	Yes	Yes	Thai	0–10	Yes
X_9	No	Yes	Yes	No	Full	$	Yes	No	Burger	>60	No
X_{10}	Yes	Yes	Yes	Yes	Full	$$$	No	Yes	Italian	10–30	No
X_{11}	No	No	No	No	None	$	No	No	Thai	0–10	No
X_{12}	Yes	Yes	Yes	Yes	Full	$	No	No	Burger	30–60	Yes

Figure 18.3 Examples for the restaurant domain.

Building a Decision Tree

DTLearn(examples, attributes, default)
if no examples, return default
if all same label, return it
$m \leftarrow$ majority label
if no attributes, return m
else
    $a \leftarrow$ choose attribute
    make node that branches on a
    remove a from attributes
    for each value v of a
        subtree $\leftarrow$ DTLearn(examples with $a = v$, attributes, m)
        add branch to subtree for v at node
    return node

Branching

want attribute that reduces uncertainity

Branching

want attribute that reduces uncertainity = entropy =

$$H(X) = -\sum_i P(x_i) \log_2 P(x_i)$$

where X is random var that takes value x_i with prob $P(x_i)$

Branching

want attribute that reduces uncertainity = entropy =

$$H(X) = -\sum_i P(x_i) \log_2 P(x_i)$$

where X is random var that takes value x_i with prob $P(x_i)$

information gain of atttribute A:

$$H(X) - \sum_{a \in A} P(a) H(X_a)$$

where X_a contains only examples with $A = a$

Branching

want attribute that reduces uncertainity = entropy =

$$H(X) = -\sum_i P(x_i)\log_2 P(x_i)$$

where X is random var that takes value x_i with prob $P(x_i)$

information gain of atttribute A:

$$H(X) - \sum_{a \in A} P(a) H(X_a)$$

where X_a contains only examples with $A = a$

prune branches when gain is small (χ^2 test, see p.705) or cross-validate

Break

- asst 10

Naive Bayes

Supervised Learning: Summary So Far

learning as function approximation

k-**NN:** distance function (any attributes), any labels
Neural network: numeric attributes, numeric or binary labels

Regression: incremental training with LMS
3-Layer ANN: train with BackProp

Decision Trees: easier with discrete attributes and labels

learning as density estimation

Bayes' Theorem

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

Bayes' Theorem

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

$$\begin{aligned} P(H) &= 0.0001 \\ P(D|H) &= 0.99 \\ P(D) &= 0.01 \end{aligned}$$

$$P(H|D) =$$

Bayes' Theorem

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

$$\begin{aligned} P(H) &= 0.0001 \\ P(D|H) &= 0.99 \\ P(D) &= 0.01 \end{aligned}$$

$$P(H|D) =$$

If you don't have P(D),

Bayes' Theorem

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

$$\begin{aligned} P(H) &= 0.0001 \\ P(D|H) &= 0.99 \\ P(D) &= 0.01 \end{aligned}$$

$$P(H|D) =$$

If you don't have P(D), somtimes it helps to note that

$$P(D) = P(D|H)P(H) + P(D|\neg H)P(\neg H)$$

A Naive Bayesian Model

Bayes' Theorem:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

A Naive Bayesian Model

Bayes' Theorem:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

naive model:

$$P(D|H) = P(x_i, \ldots, x_n|H) = \prod_i P(x_i|H)$$

A Naive Bayesian Model

Bayes' Theorem:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

naive model:

$$P(D|H) = P(x_i, \ldots, x_n|H) = \prod_i P(x_i|H)$$

attributes independent, given class

A Naive Bayesian Model

Bayes' Theorem:

$$P(H|D) = \frac{P(H)P(D|H)}{P(D)}$$

naive model:

$$P(D|H) = P(x_i, \ldots, x_n|H) = \prod_i P(x_i|H)$$

attributes independent, given class

$$P(H|x_1, \ldots, x_n) = \alpha P(H) \prod_i P(x_i|H)$$

The 'Naive Bayes' Classifier

$$P(H|x_1, \ldots, x_n) = \alpha P(H) \prod_i P(x_i|H)$$

attributes independent, given class

maximum *a posteriori* = pick highest
maximum likelihood = ignore prior

watch for sparse data when learning!

learning as density estimation

Boosting

Ensemble Learning

committees, ensembles

weak vs strong learners

reduce variance, expand hypothesis space (eg, half-spaces)

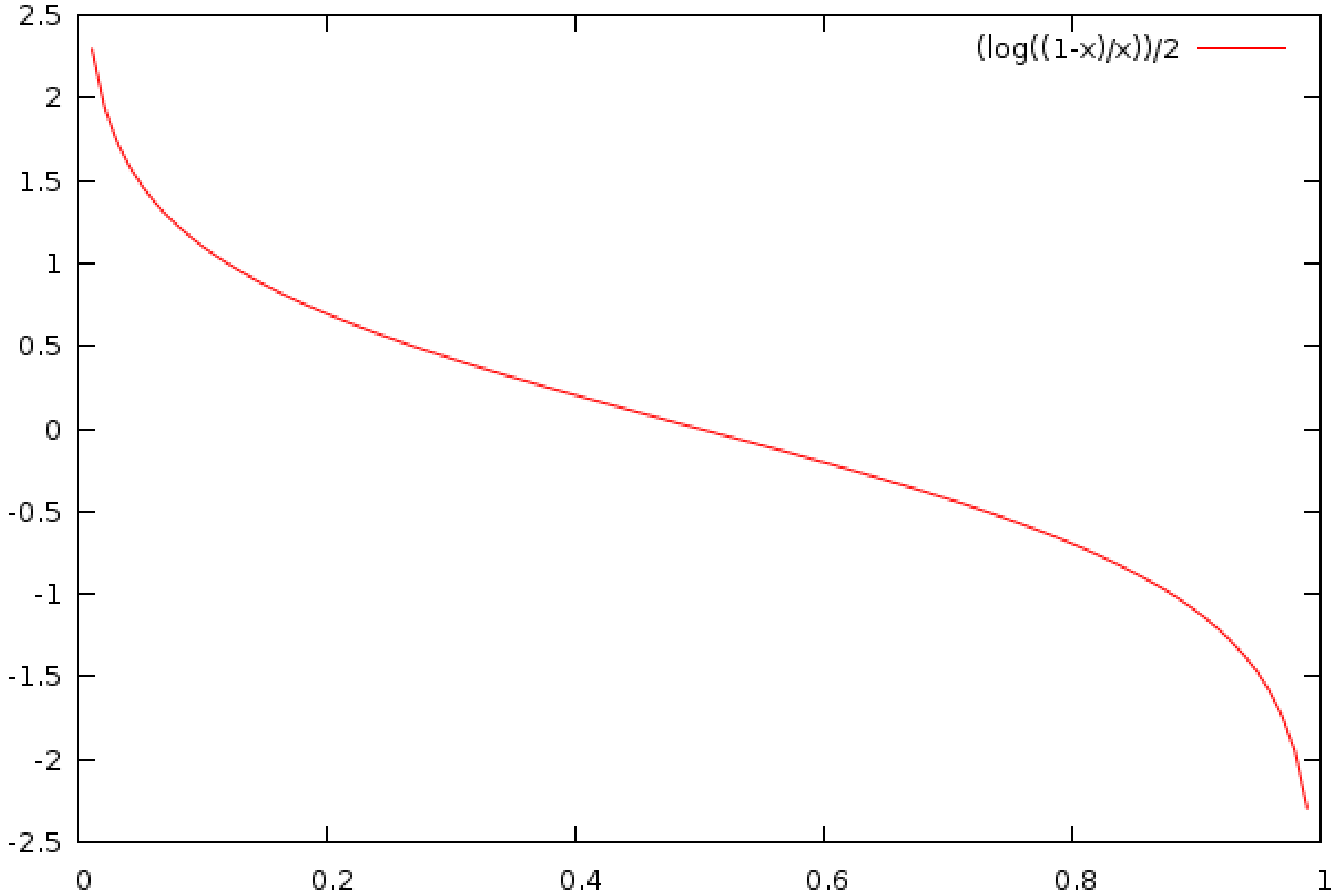
AdaBoost

N examples, T rounds, L a weak learner on weighted examples

$p \leftarrow$ uniform distribution over the N examples
for $t = 1$ to T do
  $h_t \leftarrow$ call L with weights p
  $\epsilon_t \leftarrow h_t$'s weighted misclassification probability
  if $\epsilon_t = 0$, return h_t
  $\alpha_t \leftarrow \frac{1}{2}\ln(\frac{1-\epsilon_t}{\epsilon_t})$
  for each example i
    if $h_t(i)$ is correct, $p_i \leftarrow p_i e^{-\alpha_t}$
    else, $p_i \leftarrow p_i e^{\alpha_t}$
  normalize p to sum to 1
return the h weighted by the α

to classify, choose label with highest sum of weighted votes

Boosting Function



Behavior

doesn't overfit (maximizes margin even when no error)
outliers get high weight, can be inspected
problems:

- not enough data
- hypothesis class too small
- boosting: learner too weak, too strong

Supervised Learning: Summary

k-NN: distance function (any attributes), any labels
Neural network: numeric attributes, numeric or binary labels

Regression: incremental training with LMS
3-Layer ANN: BackProp learning

Decision Trees: easier with discrete attributes and labels
Naive Bayes: easier with discrete attributes and labels
Boosting: general wrapper to improve performance

Didn't cover: RBFs, SVMs, deep learning, structured output learning

EOLQs

- What question didn't you get to ask today?
- What's still confusing?
- What would you like to hear more about?

Please write down your most pressing question about AI and put it in the box on your way out.
Thanks!