More Classifiers

Justin Silverman

Penn State University

Table of Contents

1 Naive Bayes

2 Decision Trees

3 Ensemble Methods

4 Random Forests

Sources

In this lecture I adopt slides from the following sources:

- David Sontag, NYU, Naive Bayes
- Seyoung Kim, CMU, Naive Bayes Classifier
- Stuart Russel, Berkley, Decision Trees

Section 1

Naive Bayes

Recomended Reading

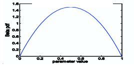
- Murphy, Section 3.5

Bayesian Learning

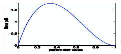
- Use Bayes' rule!

Data Likelihood

Prior



Posterior



$$P(\theta \mid \mathcal{D}) = \frac{P(\mathcal{D} \mid \theta) P(\theta)}{P(\mathcal{D})}$$

Normalization

- Or equivalently: $P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta) P(\theta)$
- For *uniform* priors, this reduces to maximum likelihood estimation!

$$P(\theta) \propto 1 \qquad P(\theta \mid \mathcal{D}) \propto P(\mathcal{D} \mid \theta)$$

Conditional Independence

- X is **conditionally independent** of Y given Z, if the probability distribution for X is independent of the value of Y, given the value of Z

$$(\forall i, j, k) P(X = i | Y = j, Z = k) = P(X = i | Z = k)$$

- e.g.,

$$P(Thunder | Rain, Lightning) = P(Thunder | Lightning)$$

- Equivalent to:

$$P(X, Y \mid Z) = P(X \mid Z) P(Y \mid Z)$$

Naïve Bayes

- Naïve Bayes assumption:
  - Features are independent given class:

$$\begin{aligned} P(X_1, X_2|Y) &= P(X_1|X_2, Y)P(X_2|Y) \\ &= P(X_1|Y)P(X_2|Y) \end{aligned}$$

  - More generally:

$$\boxed{P(X_1 \ldots X_n|Y) = \prod_i P(X_i|Y)}$$

- How many parameters now?
    - Suppose **X** is composed of *n* binary features

Reducing the number of parameters to estimate

$$P(Y|X_1,\ldots,X_n) = \frac{P(X_1,\ldots,X_n \mid Y)P(Y)}{P(X_1,\ldots,X_n)}$$

To make this tractable we naively assume conditional independence of the features given the class: ie

$$P(X_1\ldots,X_n \mid Y) = P(X_1 \mid Y)\cdot P(X_2 \mid Y)\cdots P(X_n \mid Y)$$

Now: I only need to estimate … parameters:

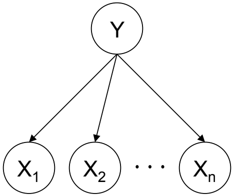
$$P(X_1 \mid Y), P(X_2 \mid Y), \cdots, P(X_n \mid Y), P(Y)$$

How many parameters to describe $P(X_1 \ldots X_n|Y)$? $P(Y)$?

- Without conditional indep assumption? $(2^n-1)\text{x}2+1$
- With conditional indep assumption? $2n+1$

The Naïve Bayes Classifier

- Given:
  - Prior P(Y)
  - *n* conditionally independent features **X** given the class Y
  - For each X_i, we have likelihood $P(X_i|Y)$



- Decision rule:

$$\begin{aligned} y^* = h_{NB}(\mathbf{x}) &= \arg\max_y P(y)P(x_1,\ldots,x_n \mid y) \\ &= \arg\max_y P(y)\prod_i P(x_i|y) \end{aligned}$$

If certain assumption holds, NB is optimal classifier!
(they typically don't)

Model Specification Example

My dog's name is Gauss. Predict whether Gauss is in the Kitchen ($y = 1$), or not ($y = 0$). Given whether I am in the kitchen ($x_1 = 1$) and or my wife is in the kitchen ($x_2 = 1$).

Model Specification Example

My dog's name is Gauss. Predict whether Gauss is in the Kitchen ($y = 1$), or not ($y = 0$). Given whether I am in the kitchen ($x_1 = 1$) and or my wife is in the kitchen ($x_2 = 1$). Also consider the covariate of the time since he last slept x_3.

Naïve Bayes Algorithm – discrete X_i

- Train Naïve Bayes (given data for X and Y)

```
for each* value $y_k$
    estimate    $\pi_k \equiv P(Y = y_k)$
for each* value $x_{ij}$ of each attribute $X_i$
    estimate    $\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k)$
```

Training Naïve Bayes Classifier Using MLE

- From the data D, estimate *class priors*.
  - For each possible value of Y, estimate *Pr(Y=y_1), Pr(Y=y_2),…. Pr(Y=y_k)*
  - An MLE estimate: $\widehat{\pi}_k = \widehat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$
- From the data, estimate the conditional probabilities
  - If every X_i has values *x_{i1},…,x_{ik}*
    - for each y_i and each X_i estimate *q(i,j,k)=Pr(X_i=x_{ij}|Y=y_i)*

$$\widehat{\theta}_{ijk} = \widehat{P}(X_i = x_{ij}|Y = y_k) = \frac{\#D\{X_i = x_{ij} \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

Number of items in dataset D for which Y=y_k

Naïve Bayes Algorithm – discrete X_i

- Train Naïve Bayes (given data for X and Y)

  for each* value y_k

  estimate $\pi_k \equiv P(Y = y_k)$

  for each* value x_{ij} of each attribute X_i

  estimate $\theta_{ijk} \equiv P(X_i = x_{ij} | Y = y_k)$

- Classify (X^{new})

$$Y^{new} \leftarrow \arg\max_{y_k} \; P(Y = y_k) \prod_i P(X_i^{new} | Y = y_k)$$

$$Y^{new} \leftarrow \arg\max_{y_k} \; \pi_k \prod_i \theta_{ijk}$$

* probabilities must sum to 1, so need estimate only n-1 of these...

Example

My dog's name is Gauss. Predict whether Gauss is in the Kitchen ($y = 1$), or not ($y = 0$). Given whether I am in the kitchen ($x_1 = 1$) and or my wife is in the kitchen ($x_2 = 1$).

y	x_1	x_2
1	1	0
0	0	1
0	0	0
1	1	1
0	0	1

Naïve Bayes: Subtlety #1

If unlucky, our MLE estimate for $P(X_i \mid Y)$ might be zero. (e.g., nobody in your sample has X_i = <40.5 and Y=rich)

- Why worry about just one parameter out of many?

$$P(X_1 \ldots X_n|Y) = \prod_i \underbrace{P(X_i|Y)}_{\text{If one of these terms is 0...}}$$

- What can be done to avoid this?

Estimating Parameters

- Maximum Likelihood Estimate (MLE): choose θ that maximizes probability of observed data $\mathcal{D}$

$$\widehat{\theta} \;=\; \arg\max_{\theta} \quad P(\mathcal{D} \mid \theta)$$

- Maximum a Posteriori (MAP) estimate: choose θ that is most probable given prior probability and the data

$$\widehat{\theta} \;=\; \arg\max_{\theta} \quad P(\theta \mid \mathcal{D})$$

$$= \arg\max_{\theta} \;=\; \frac{P(\mathcal{D} \mid \theta) P(\theta)}{P(\mathcal{D})}$$

Estimating Parameters: Y, X_i discrete-valued

Maximum likelihood estimates:

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\}}{|D|}$$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_j | Y = y_k) = \frac{\#D\{X_i = x_j \wedge Y = y_k\}}{\#D\{Y = y_k\}}$$

MAP estimates (Beta, Dirichlet priors):

$$\hat{\pi}_k = \hat{P}(Y = y_k) = \frac{\#D\{Y = y_k\} + (\beta_k - 1)}{|D| + \sum_m (\beta_m - 1)}$$

$$\hat{\theta}_{ijk} = \hat{P}(X_i = x_j | Y = y_k) = \frac{\#D\{X_i = x_j \wedge Y = y_k\} + (\beta_k - 1)}{\#D\{Y = y_k\} + \sum_m (\beta_m - 1)}$$

Only difference:
“imaginary” examples

Naïve Bayes: Subtlety #2

Often the X_i are not really conditionally independent

- We use Naïve Bayes in many cases anyway, and it often works pretty well
  - often the right classification, even when not the right probability (see [Domingos&Pazzani, 1996])
- What is effect on estimated P(Y|X)?
  - Special case: what if we add two copies: $X_i = X_k$

Special case: what if we add two copies: $X_i = X_k$

$$P(X_1 \ldots X_n|Y) = \prod_i \underbrace{P(X_i|Y)}_{\text{Redundant terms}}$$

About Naïve Bayes

- Naïve Bayes is blazingly fast and quite robust!

Section 2

Decision Trees

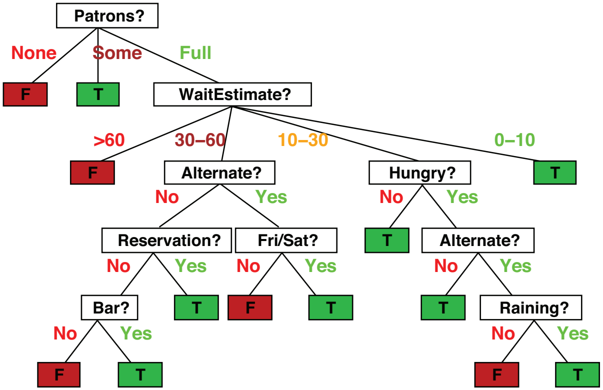
Restaurant example

Discrete inputs (some have static discretizations), Boolean output

Example	Input Attributes										Goal
	Alt	*Bar*	*Fri*	*Hun*	*Pat*	*Price*	*Rain*	*Res*	*Type*	*Est*	*WillWait*
$\mathbf{x}_1$	*Yes*	*No*	*No*	*Yes*	*Some*	*$$$*	*No*	*Yes*	*French*	*0–10*	$y_1 =$ *Yes*
$\mathbf{x}_2$	*Yes*	*No*	*No*	*Yes*	*Full*	*$*	*No*	*No*	*Thai*	*30–60*	$y_2 =$ *No*
$\mathbf{x}_3$	*No*	*Yes*	*No*	*No*	*Some*	*$*	*No*	*No*	*Burger*	*0–10*	$y_3 =$ *Yes*
$\mathbf{x}_4$	*Yes*	*No*	*Yes*	*Yes*	*Full*	*$*	*Yes*	*No*	*Thai*	*10–30*	$y_4 =$ *Yes*
$\mathbf{x}_5$	*Yes*	*No*	*Yes*	*No*	*Full*	*$$$*	*No*	*Yes*	*French*	*>60*	$y_5 =$ *No*
$\mathbf{x}_6$	*No*	*Yes*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Italian*	*0–10*	$y_6 =$ *Yes*
$\mathbf{x}_7$	*No*	*Yes*	*No*	*No*	*None*	*$*	*Yes*	*No*	*Burger*	*0–10*	$y_7 =$ *No*
$\mathbf{x}_8$	*No*	*No*	*No*	*Yes*	*Some*	*$$*	*Yes*	*Yes*	*Thai*	*0–10*	$y_8 =$ *Yes*
$\mathbf{x}_9$	*No*	*Yes*	*Yes*	*No*	*Full*	*$*	*Yes*	*No*	*Burger*	*>60*	$y_9 =$ *No*
$\mathbf{x}_{10}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$$$*	*No*	*Yes*	*Italian*	*10–30*	$y_{10} =$ *No*
$\mathbf{x}_{11}$	*No*	*No*	*No*	*No*	*None*	*$*	*No*	*No*	*Thai*	*0–10*	$y_{11} =$ *No*
$\mathbf{x}_{12}$	*Yes*	*Yes*	*Yes*	*Yes*	*Full*	*$*	*No*	*No*	*Burger*	*30–60*	$y_{12} =$ *Yes*

Decision trees

Popular representation for hypotheses, even among humans!
E.g., here is the "true" tree for deciding whether to wait:



Geometric Interpretation of a Decision Tree

Classification and regression

Each path from root to a leaf defines a region R_m of input space

Let $\mathbf{X}_m$ be the training examples that fall into R_m

Classification tree:

- discrete output
- leaf value $\hat{y}_m$ typically set to the most common value in $\mathbf{X}_m$

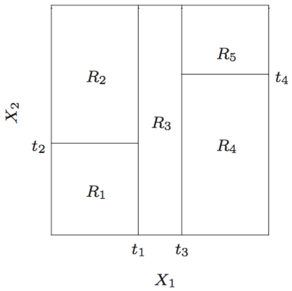
Regression tree:

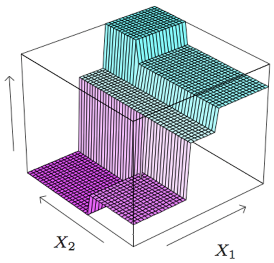
- continuous output
- leaf value $\hat{y}_m$ typically set to the mean value in $\mathbf{X}_m$ (L_2)

Discrete and continuous inputs

Simplest case: discrete inputs with small ranges (e.g., Boolean)
$\Rightarrow$ one branch for each value; attribute is “used up” (“complete split”)

For continuous attribute, test is $X_j > c$ for some split point c
$\Rightarrow$ two branches, attribute may be split further in each subtree





Also split large discrete ranges into two or more subsets

Example

My dog's name is Gauss. Predict whether Gauss is in the Kitchen ($y = 1$), or not ($y = 0$). Given whether I am in the kitchen ($x_1 = 1$) and or my wife is in the kitchen ($x_2 = 1$).

y	x_1	x_2
1	1	0
0	0	1
0	0	0
1	1	1
0	0	1

Oblique Splits

Very useful, much harder to do.

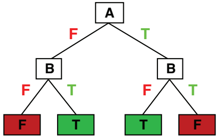
Expressiveness

Discrete-input, discrete-output case:

– Decision trees can express any function of the input attributes.

– E.g., for Boolean functions, truth table row $\rightarrow$ path to leaf:

A	B	A xor B
F	F	F
F	T	T
T	F	T
T	T	F



Continuous-input, continuous-output case:

– Can approximate any function arbitrarily closely

Trivially, there is a consistent decision tree for any training set w/ one path to leaf for each example (unless f nondeterministic in x) but it probably won't generalize to new examples

Need some kind of regularization to ensure more **compact** decision trees

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with 2^n rows

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions
$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions
$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

= number of Boolean functions
= number of distinct truth tables with 2^n rows = 2^{2^n}

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \wedge \neg Rain$)??

Digression: Hypothesis spaces

How many distinct decision trees with n Boolean attributes??

$=$ number of Boolean functions
$=$ number of distinct truth tables with 2^n rows $= 2^{2^n}$

E.g., with 6 Boolean attributes, there are 18,446,744,073,709,551,616 trees

How many purely conjunctive hypotheses (e.g., $Hungry \land \neg Rain$)??

Each attribute can be in (positive), in (negative), or out
$\Rightarrow$ 3^n distinct conjunctive hypotheses

Digression: Hypothesis spaces

More expressive hypothesis space
- increases chance that target function can be expressed
- increases number of hypotheses consistent w/ training set
  - $\Rightarrow$ many consistent hypotheses have large test error
  - $\Rightarrow$ may get worse predictions

2^{2^n} hypotheses $\Rightarrow$ all but an exponentially small fraction will require $O(2^n)$ bits to describe in **any** representation (decision trees, SVMs, English, brains)

Conversely, any given hypothesis representation can provide compact descriptions for only an exponentially small fraction of functions

E.g., decision trees are bad for "additive threshold"-type functions such as "at least 6 of the 10 inputs must be true," but linear separators are good

Decision tree learning

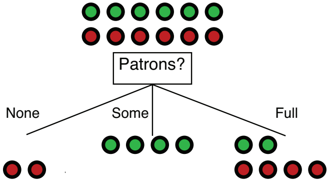
Aim: find a small tree consistent with the training examples

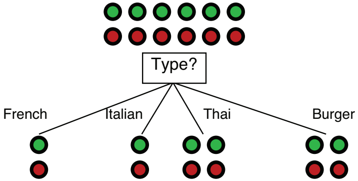
Idea: (recursively) choose "most significant" attribute as root of (sub)tree

```
function DECISION-TREE-LEARNING($examples, attributes, parent\_examples$)
returns a tree
  if $examples$ is empty then return PLURALITY-VALUE($parent\_examples$)
  else if all $examples$ have the same classification then return the classification
  else if $attributes$ is empty then return PLURALITY-VALUE($examples$)
  else
      $A \leftarrow \mathrm{argmax}_{a \in attributes}$ IMPORTANCE($a, examples$)
      $tree \leftarrow$ a new decision tree with root test $A$
      for each value $v_k$ of $A$ do
          $exs \leftarrow \{e : e \in examples$ and $e.A = v_k\}$
          $subtree \leftarrow$ DECISION-TREE-LEARNING($exs, attributes - A, examples$)
          add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
      return $tree$
```

Choosing an attribute: Minimizing loss

Idea: if we can test only one more attribute, which one results in the smallest empirical loss on examples that come through this branch?



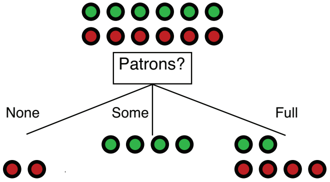


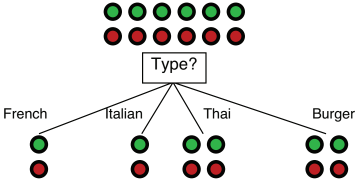
Assuming we pick the majority value in each new leaf, 0/1 loss is $0 + 0 + 2 = 2$ for $Patrons$ and $1 + 1 + 2 + 2 = 6$ for $Type$

For continuous output, assume least-squares fit (mean value) in each leaf, pick the attribute that minimizes total squared error across new leaves

Choosing an attribute: Information gain

Idea: measure the contribution the attribute makes to increasing the "purity" of the Y-values in each subset of examples





$Patrons$ is a better choice—gives information about the classification (also known as reducing entropy of distribution of output values)

Information

Information answers questions

The more clueless I am about the answer initially, the more information is contained in the answer

Scale: 1 bit = answer to Boolean question with prior $\langle 0.5, 0.5 \rangle$

Information in an answer when prior is $\langle P_1, \ldots, P_n \rangle$ is

$$H(\langle P_1, \ldots, P_n \rangle) = \sum_{i=1}^{n} - P_i \log_2 P_i$$

(also called entropy of the prior)

Convenient notation: $B(p) = H(\langle p, 1-p \rangle)$

Information contd.

Suppose we have p positive and n negative examples at the root
$\Rightarrow$ $B(p/(p+n)$ bits needed to classify a new example
E.g., for 12 restaurant examples, $p=n=6$ so we need 1 bit

An attribute splits the examples E into subsets E_k, each of which (we hope) needs less information to complete the classification

Let E_k have p_k positive and n_k negative examples
$\Rightarrow$ $B(p_k/(p_k+n_k)$ bits needed to classify a new example
$\Rightarrow$ **expected** number of bits per example over all branches is

$$\Sigma_i \ \frac{p_i+n_i}{p+n} B(p_k/(p_k+n_k))$$

For $Patrons$, this is 0.459 bits, for $Type$ this is (still) 1 bit

$\Rightarrow$ choose the attribute that minimizes the remaining information needed

Empirical loss vs. information gain

Information gain tends to be less greedy:

– Suppose X_j splits a 10/6 parent into 2/0 and 8/6 leaves

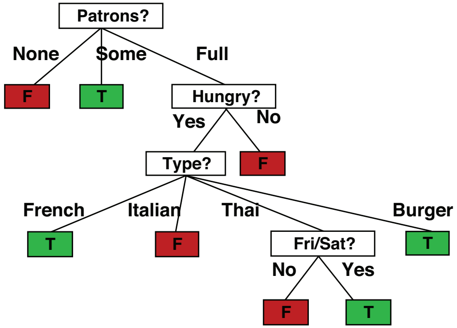
– 0/1 loss is 6 for parent, 6 for children: “no progress”

– 0/1 loss is still 6 with 4/0 and 6/6 leaves! – Information gain is 0.2044 bits: we have “peeled off” an easy category

Most experimental investigations indicate that information gain leads to better results (smaller trees, better predictive accuracy)

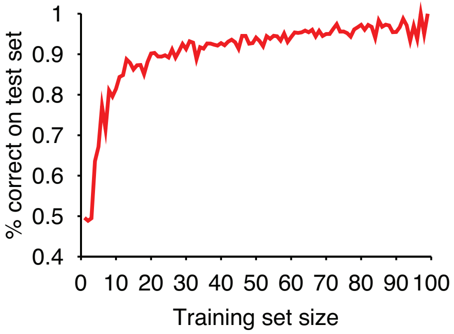
Results on restaurant data

Decision tree learned from the 12 examples:



Substantially simpler than "true" tree—a more complex hypothesis isn't justified by small amount of data

Learning curve for restaurant data



Pruning

Tree grows until it fits the data perfectly or runs out of attributes

In the non-realizable case, will severely overfit

Pruning methods trim tree back, removing “weak” tests

A test is weak if its ability to separate +ve and -ve is not
significantly different from that of a completely irrelevant attribute

E.g., X_j splits a 4/8 parent into 2/2 and 2/6 leaves; gain is 0.04411 bits.
How likely is it that randomly assigning the examples to leaves of the same
size leads to a gain of at least this amount?

Test the deviation from 0-gain split using χ^2-squared statistic,
prune leaves of nodes that fail the test, repeat until done.
Regularization parameter is the χ^2 threshold (10%, 5%, 1%, etc.)

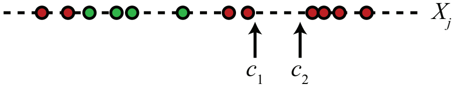
Optimal splits for continuous attributes

Infinitely many possible split points c to define node test $X_j > c$?

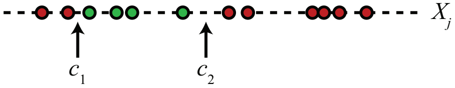
Optimal splits for continuous attributes

Infinitely many possible split points c to define node test $X_j > c$?

No! Moving split point along the empty space between two observed values has no effect on information gain or empirical loss; so just use midpoint

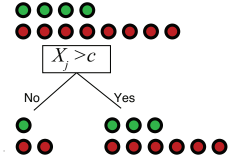


Moreover, only splits **between examples from different classes** can be optimal for information gain or empirical loss reduction



Optimal splits contd.

Lemma: Information gain is minimized (zero) when each child has the same output label distribution as the parent



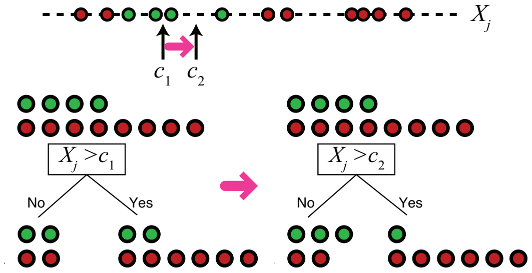
Optimal splits contd.

Lemma: if c splits X_j between two +ve examples, and examples to the left of c are **more** +ve than the parent distribution, then

– examples to the right are are **more** -ve than the parent distribution,

– moving c right by one takes children **further** from parent distribution,

– and information gain necessarily increases



Gain for c_1: $B(\frac{4}{12}) - (\frac{4}{12}B(\frac{2}{4}) + \frac{8}{12}B(\frac{2}{8})) = 0.04411$ bits

Gain for c_2: $B(\frac{4}{12}) - (\frac{5}{12}B(\frac{3}{5}) + \frac{7}{12}B(\frac{1}{7})) = 0.16859$ bits

Optimal splits for discrete attributes

A discrete attribute may have many possible values
– e.g., $ResidenceState$ has 50 values

Intuitively, this is likely to result in overfitting;
extreme case is an identifier such as SS$\#$.

Solutions:
– don't use such attributes
– use them and hope for the best
– group values by hand (e.g., $NorthEast$, $South$, $MidWest$, etc.)
– find best split into two subsets–but there are 2^K subsets!

Optimal splits for discrete attributes contd.

The following algorithm gives an optimal binary split:

Let p_{jk} be proportion of $+$ve examples
in the set of examples that have $X_j = x_{jk}$

Order the values x_{jk} by p_{jk}

Choose the best split point just as for continuous attributes

E.g., $p_{j,CA} = 7/10$, $p_{j,NY} = 4/10$, $p_{j,TX} = 5/20$, $p_{j,IL} = 3/20$;
ascending order is IL, TX, NY, CA; 3 possible split points.

Additional tricks

Blended selection criterion: info gain near root, empirical loss near leaves

"Oblique splits": node test is a linear inequality $\mathbf{w}^T\mathbf{x} + b > 0$

Full regressions at leaves: $\hat{\mathbf{y}}(\mathbf{x}) = (\mathbf{X}_\ell^T \mathbf{X}_\ell)^{-1} \mathbf{X}_\ell^T \mathbf{Y}_\ell$

Cost-sensitive classification

Lopsided outputs: one class very rare (e.g., direct mail targeting)

Scaling up: disk-resident data

Section 3

Ensemble Methods

Ensemble Methods

- Bagging
- Boosting
- Bayesian Model Averaging
- Stacking

Bagging (Bootstrap Aggregating)

Train K models in such a way as to promote variance (*i.e.*, diversity). Do this by training each learning on a random subset (sampling with replacement) of the data.

Have each learner "vote" on the prediction (each learner is weighted equally).

- Random Forests

Boosting

Incrementally build an ensemble by training each new model to correct the errors of the prior model.

e.g., increasing weight of misclassified data points from learner to learner.

- Adaboost
- XGBoost

Bayesian Model Averaging

- Take K models, calculate the marginal likelihood of each model $p(\mathcal{D}|\mathcal{M}_k)$.
- Specify a prior over models $p(\mathcal{M}_1, \ldots, \mathcal{M}_K)$
- Use Bayes rule to calculate posterior over models $p(\mathcal{M}_1, \ldots, \mathcal{M}_K|\mathcal{D})$
- Predict using weighted combination of models (weights picked by posterior):

$$p(\mathcal{D}^*|\mathcal{D}) = \int p(\mathcal{D}^*|\mathcal{M}_1, \ldots, \mathcal{M}_K)p(\mathcal{M}_1, \ldots, \mathcal{M}_K|\mathcal{D})d\mathcal{M}_1 \cdots d\mathcal{M}_K$$

Stacking

Train a learner to combine the predictions of several other learners.

Section 4

Random Forests

Random Forests

Bagging with Decision Trees.

They work Remarkably Well.

This is often my model of choice for classification when logistic regression fails and I am flying completely blind (e.g., poorly annotated ugly dataset and I just want to find something that works reasonably well).