Learning Agents

Vasant Honavar
Artificial Intelligence Research Laboratory
Department of Computer Science
Bioinformatics and Computational Biology Program
Center for Computational Intelligence, Learning, & Discovery
Iowa State University
honavar@cs.iastate.edu
www.cs.iastate.edu/~honavar/
www.cild.iastate.edu/
www.bcb.iastate.edu/
www.igert.iastate.edu

Copyright Vasant Honavar, 2006.

Learning agents

• All previous agent-programs use knowledge (e.g., model of the environment)
• Knowledge engineering is expensive and tedious
• How to acquire knowledge?
  – Learning

Why should Machines Learn?

• Some tasks are best specified by example (e.g., medical diagnosis)
• Buried in large volume of data are useful predictive relationships (data mining)
• The operating environment of certain types of software (user characteristics, distribution of problem instances) may not be completely known at design time
• Environment changes over time – ability of software to adapt to changes would enhance usability
Why study machine learning?

Practical
- Intelligent behavior requires knowledge
- Explicitly specifying the knowledge needed for specific tasks is hard, and often infeasible

Machine Learning is most useful when
- the structure of the task is not well understood but a representative dataset is available
- task (or its parameters) change dynamically

Why study machine learning?

If we can program computers to learn from experience, we can
- Dramatically enhance the usability of software
- Dramatically reduce the cost of software development
- Automate aspects of scientific discovery in emerging data rich domains

Why study machine learning? – Applications

- Medical diagnosis/image analysis (e.g. pneumonia, pap smears)
- Scientific discovery – identifying sequence correlates of protein function
- Spam Filtering, fraud detection (e.g. credit cards, phone calls)
- Search and recommendation (e.g. google, amazon)
- Automatic speech recognition & speaker verification
- Locating/tracking/identifying objects in images & video (e.g. faces)
Why study machine learning? – Science of learning

Information processing models can provide useful insights into
- How humans and animals learn
- Information requirements of learning tasks
- The precise conditions under which certain learning goals are achievable
- Inherent difficulty of learning tasks
- How to improve learning – e.g. value of active versus passive learning
- Computational architectures for learning

Machine Learning – related disciplines
- Applied Statistics – applied almost always to small data sets, manually by a statistician sometimes assisted by a computer
- Data mining – emphasis on large data sets, computational and memory considerations
- Machine learning – emphasis on automating the discovery of regularities from data, characterizing what can be learned and under what conditions, obtaining guarantees regarding quality of learned models

Machine Learning = (Statistical) Inference + Data Structures + Algorithms

What is Machine Learning?

A program $M$ is said to learn from experience $E$ with respect to some class of tasks $T$ and performance measure $P$ if its performance as measured by $P$ on tasks in $T$ in an environment $Z$ improves with experience $E$.

Example 1
- $T$ – cancer diagnosis
- $E$ – a set of diagnosed cases
- $P$ – accuracy of diagnosis on new cases
- $Z$ – noisy measurements, occasionally misdiagnosed training cases
- $M$ – a program that runs on a general purpose computer
What is Machine Learning?

Example 2

- $T$ – solving integral calculus problems, given rules of integral calculus
- $E$ – a set of solved problems
- $P$ – score on test consisting of problems not in $E$

Learning

Data

Learning = Inference + Memorization

Knowledge

Canonical Learning Problems

**Supervised Learning:** given examples of inputs and corresponding desired outputs, predict outputs on future inputs.
- Classification
- Regression
- Time series prediction

**Unsupervised Learning:** given only inputs, automatically discover representations, features, structure, etc.
- Clustering
- Outlier detection
- Compression

**Reinforcement Learning**
Learning input – output functions

Target function \( f \) – unknown to the learner – \( h \in H \)
Learner’s hypothesis about what \( f \) might be – \( f \in F \)

\( H \) – hypothesis space

Instance space – \( X \) – domain of \( f, h \)
Output space – \( Y \) – range of \( f, h \)

Example – an ordered pair \((x, y)\) where 
\[ x \in X \quad \text{and} \quad f(x) = y \in Y \]

\( F \) and \( H \) \textbf{may or may not} be the same!

Training set \( E \) – a multi set of examples

Learning algorithm \( L \) – a procedure which given some \( E \), outputs an \( h \in H \)

Learning from Examples

- **Premise** – A hypothesis (e.g., a classifier) that is consistent with a sufficiently large number of representative training examples is likely to accurately classify novel instances drawn from the same universe
- We can prove that this is an optimal approach (under appropriate assumptions)
- When the number of examples is limited, the learner needs to be smarter (e.g., find a concise hypothesis that is consistent with the data)
Learning as Bayesian Inference

Probability is the logic of Science (Jaynes)

- Bayesian (subjective) probability provides a basis for updating beliefs based on evidence
- By updating beliefs about hypotheses based on data, we can learn about the world.
- Bayesian framework provides a sound probabilistic basis for understanding many learning algorithms and designing new algorithms
- Bayesian framework provides several practical reasoning and learning algorithms

Classification using Bayesian Decision Theory

Consider the problem of classifying an instance $X$ into one of two mutually exclusive classes $o_1$ or $o_2$

$P(o_1 | X)$ = probability of class $o_1$ given the evidence $X$

$P(o_2 | X)$ = probability of class $o_2$ given the evidence $X$

What is the probability of error?

$P(\text{error} | X) = P(o_1 | X)$ if we choose $o_2$

$= P(o_2 | X)$ if we choose $o_1$

Minimum Error Classification

To minimize classification error

Choose $o_1$ if $P(o_1 | X) > P(o_2 | X)$

Choose $o_2$ if $P(o_2 | X) > P(o_1 | X)$

which yields

$P(\text{error} | X) = \min[P(o_1 | X), P(o_2 | X)]$

We have:

$P(o_1 | X) = P(X | o_1)P(o_1)$;

$P(o_2 | X) = P(X | o_2)P(o_2)$
Classification using Bayesian decision theory

Choose $\omega_1$ if $P(\omega_1|X) > P(\omega_2|X)$ i.e. $X \in R_1$
Choose $\omega_2$ if $P(\omega_1|X) > P(\omega_2|X)$ i.e. $X \in R_2$

Optimality of Bayesian Decision Rule
We can show that the Bayesian classifier is optimal in that it is guaranteed to minimize the probability of misclassification
Optimality of the Bayes Decision Rule

\[ P_e = P(x \in R_e, x \in \omega_1) + P(x \in R_e, x \in \omega_2) \]
\[ = P(x \in R_e | \omega_1)P(\omega_1) + P(x \in R_e | \omega_2)P(\omega_2) \]
\[ = P(\omega_1) \int_{R_e} p(x | \omega_1) dx + P(\omega_2) \int_{R_e} p(x | \omega_2) dx \]

Applying Bayes Rule:
\[ p(x | \omega) P(\omega) = P(\omega | x) p(x) = p(x, \omega) \]
\[ P_e = \int_{R_1} P(\omega_1 | x) p(x) dx + \int_{R_2} P(\omega_2 | x) p(x) dx \]

Because \( R_1 \cup R_2 \) covers the entire input space,
\[ \int_{R_1} P(\omega_1 | x) p(x) dx + \int_{R_2} P(\omega_2 | x) p(x) dx = P(\omega_1) \]
\[ P_e = P(\omega_1) - \left[ \int_{R_1} P(\omega_1 | x) - P(\omega_2 | x) \right] p(x) dx \]

\( P_e \) is minimized by choosing
\( R_1 \) such that \( P(\omega_1 | x) > P(\omega_2 | x) \)
and
\( R_2 \) such that \( P(\omega_1 | x) > P(\omega_2 | x) \)

The proof generalizes to multivariate input spaces
Similar result can be proved in the case of discrete (as opposed to continuous) input spaces – replace integration over the input space by summation
Bayes Decision Rule yields Minimum Error Classification

To minimize classification error
Choose $o_1$ if $P(o_1|X) > P(o_2|X)$
Choose $o_2$ if $P(o_1|X) > P(o_2|X)$
which yields
$P(error|X) = \min[P(o_1|X), P(o_2|X)]$

Bayes Decision Rule

Behavior of Bayes decision rule as a function of prior probability of classes

Bayes Optimal Classifier

Classification rule that guarantees minimum error:
Choose $o_1$ if $P(X|o_1)P(o_1) > P(X|o_2)P(o_2)$
Choose $o_2$ if $P(X|o_2)P(o_2) > P(X|o_1)P(o_1)$
If $P(X|o_1) = P(X|o_2)$
classification depends entirely on $P(o_1)$ and $P(o_2)$
If $P(o_1) = P(o_2)$,
classification depends entirely on $P(X|o_1)$ and $P(X|o_2)$
Bayes classification rule combines the effect of the two terms optimally - so as to yield minimum error classification.
Generalization to multiple classes
$c(x) = \arg \max_{o_i} P(o_i|X)$
Minimum Risk Classification

Let \( \lambda_i \) = risk or cost associated with assigning an instance to class \( \omega_i \) when the correct classification is \( \omega_i \).

\[
R(\omega_i | X) = \text{expected loss incurred in assigning } X \text{ to class } \omega_i
\]

\[
R(\omega_i | X) = \lambda_{i1}P(\omega_i | X) + \lambda_{i2}P(\omega_i | X)
\]

Classification rule that guarantees minimum risk:

Choose \( \omega_i \) if \( R(\omega_i | X) < R(\omega_j | X) \)

Choose \( \omega_j \) if \( R(\omega_j | X) < R(\omega_i | X) \)

Flip a coin otherwise.

This classification rule can be shown to be optimal in that it is guaranteed to minimize the risk of misclassification.

Summary of Bayesian recipe for classification

\( \lambda_i \) = risk or cost associated with assigning an instance to class \( \omega_i \) when the correct classification is \( \omega_i \).

Choose \( \omega_i \) if

\[
\frac{P(X|\omega_i)}{P(X|\omega_j)} > \frac{\lambda_{i1} - \lambda_{i2}}{\lambda_{j1} - \lambda_{j2}} P(\omega_i)
\]

Choose \( \omega_j \) if

\[
\frac{P(X|\omega_j)}{P(X|\omega_i)} > \frac{\lambda_{j1} - \lambda_{j2}}{\lambda_{i1} - \lambda_{i2}} P(\omega_i)
\]

Minimum error classification rule is a special case:

Choose \( \omega_i \) if

\[
\frac{P(X|\omega_i)}{P(X|\omega_j)} > \frac{\lambda_{i1}}{\lambda_{j1}} P(\omega_i)
\]

Otherwise choose \( \omega_j \).
Bayesian recipe for classification

Note that $P(\omega_2 | x) = \frac{P(x | \omega_2) P(\omega_2)}{P(x)}$
Model $P(x | \omega_1), P(x | \omega_2), P(\omega_1)$, and $P(\omega_2)$
Using Bayes rule, choose $\omega_1$ if $P(x | \omega_1) P(\omega_1) > P(x | \omega_2) P(\omega_2)$
Otherwise choose $\omega_2$

Summary of Bayesian recipe for classification

• The Bayesian recipe is simple, optimal, and in principle, straightforward to apply
• To use this recipe in practice, we need to know $P(X | \omega_i)$ – the generative model for data for each class and $P(\omega_i)$ – the prior probabilities of classes
• Because these probabilities are unknown, we need to estimate them from data – or learn them!
• $X$ is typically high-dimensional
• Need to estimate $P(X | \omega_i)$ from limited data

Naïve Bayes Classifier

• We can classify $X$ if we know $P(X | \omega_i)$
• How to learn $P(X | \omega_i)$?
One solution: Assume that the random variables in $X$ are conditionally independent given the class.
• Result: Naïve Bayes classifier which performs optimally under certain assumptions
• A simple, practical learning algorithm grounded in Probability Theory
When to use
• Attributes that describe instances are likely to be conditionally independent given classification
• The data is insufficient to estimate all the probabilities reliably if we do not assume independence
Conditional Independence

Let $Z_1, \ldots, Z_n$ and $W$ be random variables on a given event space. $Z_1, \ldots, Z_n$ are mutually independent given $W$ if

$$P(Z_1, Z_2, \ldots, Z_n | W) = \prod_{i=1}^n P(Z_i | W)$$

Note that these represent sets of equations, for all possible value assignments to random variables.

Implications of Independence

- Suppose we have 5 Binary attributes and a binary class label.
- Without independence, in order to specify the joint distribution, we need to specify a probability for each possible assignment of values to each variable resulting in a table of size $2^{5\times 2} = 64$.
- Suppose the features are independent given the class label – we only need $5(2\times 2) = 20$ entries.
- The reduction in the number of probabilities to be estimated is even more striking when $N$, the number of attributes is large – from $O(2^N)$ to $O(N)$.

Naive Bayes Classifier

Consider a discrete valued target function $f : \mathcal{X} \rightarrow \Omega$ where an instance $X = (X_1, X_2, \ldots, X_n) \in \mathcal{X}$ is described in terms of attribute values $X_i = x_i$, $X_j = x_j$, ... $X_n = x_n$ where $x_i \in \text{Domain}(X_i)$.

$$o_{\text{MAP}} = \arg \max_{o \in \Omega} P(o | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$$

$$= \arg \max_{o \in \Omega} \frac{P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n | o)P(o)}{P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)}$$

$$= \arg \max_{o \in \Omega} P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n | o)P(o)$$

$o_{\text{MAP}}$ is called the maximum a posteriori classification.
Naive Bayes Classifier

\[ \omega_{\text{MAP}} = \arg \max_{\omega_j \in \Omega} P(\omega_j \mid X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n) \]

\[ = \arg \max_{\omega_j \in \Omega} P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n \mid \omega_j) P(\omega_j) \]

If the attributes are independent given the class, we have

\[ \omega_{\text{MAP}} = \arg \max_{\omega_j \in \Omega} \prod_{i=1}^{n} P(X_i = x_i \mid \omega_j) P(\omega_j) \]

\[ = \omega_{\text{MAP}} \]

\[ = \arg \max_{\omega_j \in \Omega} P(\omega_j) \prod_{i=1}^{n} P(X_i = x_i \mid \omega_j) \]

Naive Bayes Learner

For each possible value \( \omega \) of \( \Omega \),

\[ \hat{P}(\Omega = \omega) \leftarrow \text{Estimate}(P(\Omega = \omega), D) \]

For each possible value \( a_i \) of \( X_i \),

\[ \hat{P}(X_i = a_i \mid \omega) \leftarrow \text{Estimate}(P(X_i = a_i \mid \Omega = \omega), D) \]

Classify a new instance \( X = (x_1, x_2, \ldots, x_n) \)

\[ c(X) = \arg \max_{\omega_j \in \Omega} P(\omega_j) \prod_{i=1}^{n} P(X_i = x_i \mid \omega_j) \]

Estimate is a procedure for estimating the relevant probabilities from set of training examples.

Learning Dating Preferences

Instances – ordered 3-tuples of attribute values corresponding to

- Height (all, short)
- Hair (dark, blonde, red)
- Eye (blue, brown)

Classes – +, -

<table>
<thead>
<tr>
<th>Training Data</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>Instance</td>
<td></td>
</tr>
<tr>
<td>I_1 (t, d, l)</td>
<td>+</td>
</tr>
<tr>
<td>I_2 (s, d, l)</td>
<td>+</td>
</tr>
<tr>
<td>I_3 (t, b, l)</td>
<td>-</td>
</tr>
<tr>
<td>I_4 (t, r, l)</td>
<td>-</td>
</tr>
<tr>
<td>I_5 (s, b, l)</td>
<td>-</td>
</tr>
<tr>
<td>I_6 (t, b, w)</td>
<td>+</td>
</tr>
<tr>
<td>I_7 (t, d, w)</td>
<td>+</td>
</tr>
<tr>
<td>I_8 (s, b, w)</td>
<td>+</td>
</tr>
</tbody>
</table>
Probabilities to estimate

\[ P(+) = \frac{5}{8}, \quad P(-) = \frac{3}{8} \]

|                | \( \text{Height} | c \) | \( \text{Hair} | c \) | \( \text{Eye} | c \) |
|----------------|-----------------|-----------------|-----------------|
| \(+\)          | 3/5             | 2/5             | 2/5             |
| \(-\)          | 2/3             | 1/3             | 0               |

Classify \( \text{Height}=t, \text{Hair}=b, \text{eye}=l \)
\[ P(X | +) = \left( \frac{3}{5} \right) \left( \frac{2}{5} \right) \left( \frac{2}{5} \right) = \frac{12}{125} \]
\[ P(X | -) = \left( \frac{2}{3} \right) \left( \frac{2}{3} \right) \left( 1 \right) = \frac{4}{9} \]
Classification = ?

Classify \( \text{Height}=t, \text{Hair}=r, \text{eye}=w \)

Note the problem with zero probabilities
Solution – Use Laplacian estimates

Estimation of Probabilities from Small Samples

\[ \hat{P}(X_i = a_k \mid o_j) = \frac{n_{ak} + mp}{n_j + m} \]
- \( n \) is the number of training examples of class \( o_j \)
- \( n_{ak} \) is the number of training examples of class \( o_j \) which have attribute value \( a_k \) for attribute \( X_i \)
- \( p \) is the prior estimate for \( \hat{P}(X_i = a_k \mid o_j) \)
- \( m \) is the weight given to the prior

As \( n \to \infty \), \( \hat{P}(X_i = a_k \mid o_j) \to \frac{n_{ak}}{n_j} \)

This is effectively the same as using Dirichlet priors

Sample Applications of Naïve Bayes Classifier

- Learning dating preferences
- Learn which news articles are of interest.
- Learn to classify web pages by topic.
- Learn to classify SPAM
- Learn to assign proteins to functional families based on amino acid composition

Naïve Bayes is among the most useful algorithms
What attributes shall we use to represent text?
Learning to Classify Text

• Target concept Interesting? : Documents $\rightarrow \{+,-\}$
• Learning: Use training examples to estimate
  $P(+), P(-), P(d|+), P(d|-)$

Alternative generative models for documents:
• Represent each document by sequence of words
  - In the most general case, we need a probability for each word
    occurrence in each position in the document, for each possible
    document length
  - Too many probabilities to estimate!
• Represent each document by tuples of word counts

Bag of Words Representation

So we estimate one position-independent class-conditional probability $P(w_j | c_i)$ for each word instead of the set of
probabilities $P(X_1 = w_j | c_i) \ldots P(X_{\ell(d,c)} = w_j | c_i)$
The number of probabilities to be estimated drops to
$|\text{Vocabulary}| \cdot |L|$

The result is a generative model for documents that treats
each document as an ordered tuple of word frequencies

More sophisticated models can consider dependencies
between adjacent word positions
Learning to Classify Text

With the bag of words representation, we have

$$P(d | \omega_1) \propto \left( \sum_{w} n_{w} \right) \prod_{w} \left( P(w | \omega_1) \right)^{n_w}$$

where \( n_w \) is the number of occurrences of \( w \) in document \( d \)
(ignoring dependence on length of the document)

We can estimate \( P(w | \omega_1) \) from the labeled bags of words we have.

---

Naïve Bayes Text Classifier

- Given 1000 training documents from each group, learn to classify new documents according to the newsgroup where it belongs
- Naïve Bayes achieves 89% classification accuracy

comp.graphics
comp.os.ms-windows.misc
comp.sys.ibm.pc.hardware
comp.sys.mac.hardware
comp.windows.x
talk.religion.misc
talk.politics.mideast
talk.politics.misc
talk.politics.guns
misc.forsale
rec.autos
rec.motorcycles
rec.sport.baseball
rec.sport.hockey
sci.space
sci.crypt
sci.electronics
sci.med

---

Representative article from rec.sport.hockey

Path: cantaloupe.arc.cs.cmu.edu/dausnews.harvard.edu/logicse@umn.edu
From: xxx@yyy.zzz.edu (John Doe)
Subject: Re: This year’s biggest and worst (opinion)...
Date: 5 Apr 93 09:53:39 GMT

I can only comment on the Kings, but the most obvious candidate for pleasant surprise is Alex Zhitnik. He came highly touted as a defensive defenseman, but he’s clearly much more than that. Great skater and hard shot (though wish he were more accurate). In fact, he pretty much allowed the Kings to trade away that huge defensive liability Paul Coffey. Kelly Hrudey is only the biggest disappointment if you thought he was any good to begin with. But, at best, he’s only a mediocre goaltender. A better choice would be Tomas Sandstrom, though not through any fault of his own, but because some thugs in Toronto decided ---.
Naïve Bayes Learner – Summary

- Produces minimum error classifier if attributes are conditionally independent given the class

**When to use**

- Attributes that describe instances are likely to be conditionally independent given classification
- There is not enough data to estimate all the probabilities reliably if we do not assume independence
- Often works well even if when independence assumption is violated (Domigos and Pazzani, 1996)
- Can be used iteratively – Kang et al., 2006

Extra slides on Estimation

- Maximum likelihood estimation
- Bayesian estimation
- Maximum a posteriori estimation
Example: Binomial Experiment

- When tossed, the thumbtack can land in one of two positions: **Head** or **Tail**
- We denote by $\theta$ the (unknown) probability $P(H)$.
- Estimation task—
- Given a sequence of toss samples $x[1], x[2], \ldots, x[M]$ we want to estimate the probabilities $P(H) = \theta$ and $P(T) = 1 - \theta$.

Statistical parameter fitting

Consider samples $x[1], x[2], \ldots, x[M]$ such that

- The set of values that $X$ can take is known
- Each is sampled from the same distribution
- Each is sampled independently of the rest

The task is to find a parameter $\Theta$ so that the data can be summarized by a probability $P(x[j] | \Theta)$.

- The parameters depend on the given family of probability distributions: multinomial, Gaussian, Poisson, etc.
- We will focus first on binomial and then on multinomial distributions
- The main ideas generalize to other distribution families

The Likelihood Function

How good is a particular $\theta$?

It depends on how likely it is to generate the observed data $\mathcal{D}$:

$$L(\theta; \mathcal{D}) = P(D | \theta) = \prod P(x[m] | \theta)$$

The likelihood for the sequence $H, T, T, H, H$ is

$$L(\theta; \mathcal{D}) = \theta \cdot (1 - \theta) \cdot (1 - \theta) \cdot \theta \cdot \theta$$
Likelihood function

- The likelihood function $L(\theta : D)$ provides a measure of relative preferences for various values of the parameter $\theta$ given a collection of observations $D$ drawn from a distribution that is parameterized by fixed but unknown $\theta$.
- $L(\theta : D)$ is the probability of the observed data $D$ considered as a function of $\theta$.
- Suppose data $D$ is 5 heads out of 8 tosses. What is the likelihood function that the observations were generated by a binomial distribution with an fixed parameter $\theta$?

$$L(\theta : D) = \theta^5(1-\theta)^3$$

Sufficient Statistics

- To compute the likelihood in the thumbtack example we only require $N_H$ and $N_T$ (the number of heads and the number of tails)
- $N_H$ and $N_T$ are sufficient statistics for the parameter $\theta$ that specifies the binomial distribution
- A statistic is simply a function of the data
- A sufficient statistic $s$ for a parameter $\theta$ is a function that summarizes from the data $D$, the relevant information $s(D)$ needed to compute the likelihood $L(\theta : D)$.
- If $s$ is a sufficient statistic for $s(D) = s(D')$, then $L(\theta : D) = L(\theta : D')$

Maximum Likelihood Estimation

- **Main Idea:** Learn parameters that maximize the likelihood function
- Maximum likelihood estimation is
  - Intuitively appealing
  - One of the most commonly used estimators in statistics
  - Assumes that the parameter to be estimated is fixed, but unknown
Example: MLE for Binomial Data

- Applying the MLE principle we get

\[ \hat{\theta} = \frac{N_H}{N_H + N_T} \]

Example:

\( (N_H, N_T) = (3, 2) \)

ML estimate is \( 3/5 = 0.6 \)

MLE for Binomial Data

The likelihood is positive for all legitimate values of \( \theta \)

So maximizing the likelihood is equivalent to maximizing its logarithm i.e. log likelihood

\[ \frac{\partial}{\partial \theta} \log L(\theta; D) = 0 \text{ at extrema of } L(\theta; D) \]

\[ \frac{\partial}{\partial \theta} \log L(\theta; D) = \frac{N_H - N_H (1 - \theta)}{N_H + N_T (1 - \theta)} = 0 \]

\( N_H + N_T \theta = N_H \)

\[ \hat{\theta}_{ML} = \frac{N_H}{N_H + N_T} \]

Note that the likelihood is indeed maximized at \( \theta = \hat{\theta}_{ML} \) because in the neighborhood of \( \hat{\theta}_{ML} \), the value of the likelihood is smaller than it is at \( \theta = \hat{\theta}_{ML} \)

Maximum and curvature of likelihood around the maximum

- At the maximum, the derivative of the log likelihood is zero
- At the maximum, the second derivative is negative.
- The curvature of the log likelihood is defined as

\[ I(\theta) = -\frac{\partial^2}{\partial \theta^2} \log L(\theta; D) \]

- Large observed curvature \( I(\hat{\theta}_{ML}) \) at \( \theta = \hat{\theta}_{ML} \)
  - is associated with a sharp peak, intuitively indicating less uncertainty about the maximum likelihood estimate
  - \( I(\hat{\theta}_{ML}) \) is called the Fisher information
Maximum Likelihood Estimate

ML estimate can be shown to be

- Asymptotically unbiased
  \[ \lim_{N \to \infty} E(\hat{\theta}_{ML}) = \theta_{True} \]

- Asymptotically consistent - converges to the true value as the number of examples approaches infinity
  \[ \lim_{N \to \infty} \Pr(\|\hat{\theta}_{ML} - \theta_{True}\| \leq \epsilon) = 1 \]
  \[ \lim_{N \to \infty} E(\|\hat{\theta}_{ML} - \theta_{True}\|) = 0 \]

- Asymptotically efficient – achieves the lowest variance that any estimate can achieve for a training set of a certain size (satisfies the Cramer-Rao bound)

From Binomial to Multinomial

- Suppose a random variable \( X \) can take the values \( 1, 2, \ldots, K \)
- We want to learn the parameters \( \theta_1, \theta_2, \ldots, \theta_K \)
- Sufficient statistics: \( N_1, N_2, \ldots, N_K \) - the number of times each outcome is observed
- Likelihood function
  \[ L(\theta : D) = \prod_{k=1}^{K} \theta_{k}^{N_k} \]
- ML estimate
  \[ \hat{\theta}_k = \frac{N_k}{\sum N_i} \]
Summary of Maximum Likelihood estimation

• Define a likelihood function which is a measure of how likely it is that the observed data were generated from a probability distribution with a particular choice of parameters
• Select the parameters that maximize the likelihood
• In simple cases, ML estimate has a closed form solution
• In other cases, ML estimation may require numerical optimization

• Problem with ML estimate – assigns zero probability to unobserved values – can lead to difficulties when estimating from small samples
• Question – How would Naive Bayes classifier behave if some of the class conditional probability estimates are zero?

Bayesian Estimation

• MLE commits to a specific value of the unknown parameter(s)
• MLE is the same in both cases shown

Of course, in general, one cannot summarize a function by a single number!

Intuitively, the confidence in the estimates should be different

Bayesian Estimation

Maximum Likelihood approach is Frequentist at its core
• Assumes there is an unknown but fixed parameter \( \theta \)
• Estimates \( \theta \) with some confidence
• Prediction of probabilities using the estimated parameter value

Bayesian Approach
• Represents uncertainty about the unknown parameter
• Uses probability to quantify this uncertainty:
  – Unknown parameters as random variables
• Prediction follows from the rules of probability:
  – Expectation over the unknown parameters
Example: Binomial Data Revisited

- Suppose that we choose a uniform prior $p(\theta) = 1$ for $\theta$ in $[0,1]$.
- $p(\theta | D)$ is proportional to the likelihood $L(\theta | D)$

$$p(\theta | D) = \frac{p(D | \theta) p(\theta)}{{\int p(D | \theta) p(\theta) d\theta}}$$

In this case, $p(D | \theta) = \theta^5 (1 - \theta)$ and $\forall \theta \in [0,1], p(\theta) = \frac{1}{1 - 0} = 1$

$$\int p(D | \theta) p(\theta) = \left[ \theta^5 (1 - \theta) \right]_{\theta = 0}^{\theta = 1} = \frac{1}{10}$$

$$p(\theta | D) = \frac{\theta^5 (1 - \theta)}{\frac{1}{10}} = 10 \theta^5 (1 - \theta)$$

$$P(X = 1 | H | D) = \int p(\theta | D) d\theta = \int 10 \theta^5 (1 - \theta) d\theta = \frac{10 \theta^6}{6} - \frac{5 \theta^7}{7} | \theta = 0.7142$$

- In practice we might want to express priors that allow us to express our beliefs regarding the parameter to be estimated.
- For example, we might want a prior that assigns a higher probability to parameter values that describe a fair coin than it does to an unfair coin.
- The beta distribution allows us to capture such prior beliefs.

Bayesian estimate is $P(X = 1 | H | D) = \int \theta p(\theta | D) d\theta = \frac{5}{7} = 0.7142$

In this example, MLE and Bayesian prediction differ.

It can be proved that if the prior is well-behaved – i.e. does not assign 0 density to any feasible parameter value – then both MLE and Bayesian estimate converge to the same value in the limit.

Both almost surely converge to the underlying distribution $P(X)$, but the ML and Bayesian approaches behave differently when the number of samples is small.

All relative frequencies are not equi-probable.

- In practice we might want to express priors that allow us to express our beliefs regarding the parameter to be estimated.
- For example, we might want a prior that assigns a higher probability to parameter values that describe a fair coin than it does to an unfair coin.
- The beta distribution allows us to capture such prior beliefs.
Gamma Function:
\[ \Gamma(x) = \int_0^\infty t^{x-1} e^{-t} dt \]

The integral converges if and only if \( x > 0 \).
If \( x \) is an integer that is greater than 0, it can be shown that
\[ \Gamma(x) = (x-1)! \]
So
\[ \Gamma(x+1) = x \]

The beta density function with parameters \( a, b \), \( N = a + b \), where \( a, b \) are real numbers \( > 0 \), beta(\( \theta; a, b \)) is:
\[ p(\theta) = \frac{\Gamma(N)}{\Gamma(a)\Gamma(b)} \theta^{a-1}(1-\theta)^{b-1} \]
where \( 0 \leq \theta \leq 1 \)

If \( \theta \) has distribution given by \( \text{beta}(\theta; a, b) \), then \( E(\theta) = \frac{a}{N} \).

Let \( D = \{ X[1], \ldots, X[M] \} \) be a sequence of iid samples from a binomial distribution;
Let \( N_y = s \), \( N_x = t \), and \( p(\theta) = \text{beta}(\theta; a, b) \)
Then we can show that \( p(\theta|D) = \text{beta}(\theta; a+s, b+t) \)
Update of the parameter with a beta prior based on data yields a beta posterior

Conjugate Families
• The property that the posterior distribution follows the same
  parametric form as the prior distribution is called conjugacy

• Conjugate families are useful because:
  – For many distributions we can represent them with
    hyper parameters
  – They allow for sequential update to obtain the posterior
  – In many cases we have closed-form solution for
    prediction

• Beta prior is a conjugate family for the binomial likelihood
Bayesian prediction

prior: \( \text{beta}(\theta; a, b) \)
Data: \( D = \{X[1], \ldots, X[M]\} \)
posterior: \( p(\theta \mid D) = \text{beta}(\theta; a + N_y, b + N_{\neg y}) \)
prediction: \( P(X[M+1] = H \mid D) = \frac{a + N_y}{N + M} = \frac{(a + b) + (N_y + N_{\neg y})}{(a + b)} \)

Dirichlet Priors

- Recall that the likelihood function is
- A Dirichlet prior with hyperparameters \( \alpha_1, \ldots, \alpha_K \) is defined as
  \[
  P(\theta) = \frac{\Gamma(\alpha)}{\prod \Gamma(\alpha_k)} \prod \theta_k^{\alpha_k-1}; \quad 0 \leq \theta_k \leq 1; \quad \sum_k \theta_k = 1
  \]
  where \( \theta = (\theta_1, \ldots, \theta_K) \)
- Then the posterior has the same form, with hyperparameters \( \alpha_1 + N_{\neg y}, \ldots, \alpha_K + N_K \)
  \[
  P(\theta \mid D) \propto P(\theta)P(D \mid \theta) \\
  \propto \prod \theta_k^{\alpha_k-1} \prod \theta_k^{N_y} = \prod \theta_k^{\alpha_k + N_{\neg y} - 1}
  \]

Dirichlet Priors

- Dirichlet priors enable closed form prediction based on multinomial samples:
  - If \( P(\theta) \) is Dirichlet with hyperparameters \( \alpha_1, \ldots, \alpha_K \) then
    \[
    P(X[1] = k) = \int \theta_k P(\theta) d\theta = \frac{\alpha_k}{\sum_i \alpha_i}
    \]
  - Since the posterior is also Dirichlet, we get
    \[
    P(X[M+1] = k \mid D) = \int \theta_k P(\theta \mid D) d\theta = \frac{\alpha_k + N_{\neg y}}{\sum_i (\alpha_i + N_i)}
    \]
Intuition behind priors

• The hyperparameters $\alpha_1, \ldots, \alpha_K$ can be thought of as imaginary counts from our prior experience.

• Equivalent sample size $= \alpha_1 + \cdots + \alpha_K$.

• The larger the equivalent sample size the more confident we are in our prior.

Effect of Priors

Prediction of $P(X = H)$ after seeing data with $N_H = 0.25N_T$ for different sample sizes.

Different strength $\alpha_H \neq \alpha_T$

Fixed ratio $\alpha_H / \alpha_T$

Different ratio $\alpha_H / \alpha_T$

Effect of Priors

• In real data, Bayesian estimates are less sensitive to noise in the data.
Conjugate Families

• The property that the posterior distribution follows the same parametric form as the prior distribution is called conjugacy.
  – Dirichlet prior is a conjugate family for the multinomial likelihood.

• Conjugate families are useful because:
  – For many distributions we can represent them with hyperparameters.
  – They allow for sequential update within the same representation.
  – In many cases we have closed-form solution for prediction.

Bayesian Estimation

\[
P(x[M+1] | x[1],\ldots,x[M]) = \pi(\theta) x[1],\ldots,x[M] d\theta = \int P(x[M+1] | \theta) x[1],\ldots,x[M] d\theta\]

where

\[
P(\theta | x[1],\ldots,x[M]) = \frac{P(x[1],\ldots,x[M] | \theta) \pi(\theta)}{P(x[1],\ldots,x[M])}
\]

Summary of Bayesian estimation

• Treat the unknown parameters as random variables.
• Assume a prior distribution for the unknown parameters.
• Update the distribution of the parameters based on data.
• Use Bayes rule to make prediction.
Maximum a posteriori (MAP) estimates –
Reconciling ML and Bayesian approaches

\[ P(\Theta|D) = \frac{P(D|\Theta)P(\Theta)}{P(D)} \]

\[ \Theta_{MAP} = \arg \max_{\Theta} P(\Theta|D) \]
\[ = \arg \max_{\Theta} P(D|\Theta)P(\Theta) \]
\[ = \arg \max_{\Theta} P(\Theta)L(\Theta:D) \]

Like in Bayesian estimation, we treat the unknown parameters as random variables.

But we estimate a single value for the parameter – the maximum a posteriori estimate that corresponds to the most probable value of the parameter given the data for a given choice of the prior.

Back to Naive Bayes Classifier

\[ \hat{P}(X_i = a_i | \omega_j) = 0 \rightarrow \hat{P}(\omega_j) \prod \hat{P}(X_i = a_i | \omega_j) = 0 \]

If one of the attribute values has estimated class conditional probability of 0, it dominates all other attribute values.

When we have few examples, this is more likely.

Solution – use priors e.g., assume each value to be equally likely unless data indicates otherwise.
Decision Tree Classifiers

- Modeling dependencies among input variables
- Elements of information theory
- How to learn decision trees from data
- Over-fitting and how to minimize it
- How to deal with missing values in the data
- Learning decision trees from distributed data

Decision tree representation

In the simplest case,
- each internal node tests on an attribute
- each branch corresponds to an attribute value
- each leaf node corresponds to a class label

In general,
- each internal node corresponds to a test (on input instances) with mutually exclusive and exhaustive outcomes – tests may be univariate or multivariate
- each branch corresponds to an outcome of a test
- each leaf node corresponds to a class label
Decision tree representation

- Any Boolean function can be represented by a decision tree

- Any function

\[ f: A_1 \times A_2 \times \cdots \times A_n \rightarrow C \]

where each \( A_i \) is the domain of the \( i \)th attribute and \( C \) is a discrete set of values (class labels) can be represented by a decision tree

- In general, the inputs need not be discrete valued

Learning Decision Tree Classifiers

- Decision trees are especially well suited for representing simple rules for classifying instances that are described by discrete attribute values

- Decision tree learning algorithms
  - Implement Ockham’s razor as a preference bias (simpler decision trees are preferred over more complex trees)
  - Are relatively efficient – linear in the size of the decision tree and the size of the data set
  - Produce comprehensible results
  - Are often among the first to be tried on a new data set
Learning Decision Tree Classifiers

- Ockham’s razor recommends that we pick the simplest decision tree that is consistent with the training set
- Simplest tree is one that takes the fewest bits to encode (why? – information theory)
- There are far too many trees that are consistent with a training set
- Searching for the simplest tree that is consistent with the training set is not typically computationally feasible

Solution

- Use a greedy algorithm – not guaranteed to find the simplest tree – but works well in practice
- Or restrict the space of hypothesis to a subset of simple trees

Information – Some intuitions

- Information reduces uncertainty
- Information is relative – to what you already know
- Information content of a message is related to how surprising the message is
- Information is related to information depends on context

Digression: Information and Uncertainty

You are stuck inside. You send me out to report back to you on what the weather is like. I do not lie, so you trust me. You and I are both generally familiar with the weather in Iowa.

On a July afternoon in Iowa, I walk into the room and tell you it is hot outside.

On a January afternoon in Iowa, I walk into the room and tell you it is hot outside.
Digression: Information and Uncertainty

How much information does a message contain?
If my message to you describes a scenario that you expect with certainty, the information content of the message for you is zero.
The more surprising the message to the receiver, the greater the amount of information conveyed by the message.
What does it mean for a message to be surprising?

Suppose I have a coin with heads on both sides and you know that I have a coin with heads on both sides. I toss the coin, and without showing you the outcome, tell you that it came up heads. How much information did I give you?

Suppose I have a fair coin and you know that I have a fair coin. I toss the coin, and without showing you the outcome, tell you that it came up heads. How much information did I give you?

Information
• Without loss of generality, assume that messages are binary – made of 0s and 1s.
• Conveying the outcome of a fair coin toss requires 1 bit of information – need to identify one out of two equally likely outcomes.
• Conveying the outcome one of an experiment with 8 equally likely outcomes requires 3 bits.
• Conveying an outcome that is certain takes 0 bits.
• In general, if an outcome has a probability \( p \), the information content of the corresponding message is

\[
I(p) = -\log_2 p \quad I(0) = 0
\]
Information is Subjective

• Suppose there are 3 agents – Adrian, Oksana, Carson, in a world where a dice has been tossed. Adrian observes the outcome is a “6” and whispers to Oksana that the outcome is “even” but Carson knows nothing about the outcome.
• Probability assigned by Oksana to the event “6” is a subjective measure of Oksana’s belief about the state of the world.
• Information gained by Adrian by looking at the outcome of the dice =log\(_2\)6 bits.
• Information conveyed by Adrian to Oksana = \log\(_2\)6 – \log\(_2\)3 bits
• Information conveyed by Adrian to Carson = 0 bits

Information and Shannon Entropy

• Suppose we have a message that conveys the result of a random experiment with \(m\) possible discrete outcomes, with probabilities \(p_1, p_2, \ldots, p_m\).

The expected information content of such a message is called the entropy of the probability distribution

\[
H(p_1, p_2, \ldots, p_m) = \sum_{i=1}^{m} p_i I(p_i) = -\sum_{i=1}^{m} p_i \log(p_i) \\
I(p_i) = -\log(p_i) \text{ provided } p_i \neq 0 \\
I(p_i) = 0 \text{ otherwise}
\]

Shannon’s entropy as a measure of information

Let \(\hat{P} = (p_1, \ldots, p_m)\) be a discrete probability distribution. The entropy of the distribution \(\hat{P}\) is given by

\[
H(\hat{P}) = \sum_{i=1}^{m} p_i \log\left(\frac{1}{p_i}\right) = -\sum_{i=1}^{m} p_i \log(p_i)
\]

\[
H\left(\frac{1}{2}, \frac{1}{2}\right) = -\sum_{i=1}^{m} p_i \log\left(p_i\right) = -\left(\frac{1}{2}\right) \log\left(\frac{1}{2}\right) - \left(\frac{1}{2}\right) \log\left(\frac{1}{2}\right) = 1 \text{ bit}
\]

\[
H(0,1) = -\sum_{i=1}^{m} p_i \log(p_i) = -1 \log(1) - 0 \log(0) = 0 \text{ bit}
\]
Properties of Shannon’s entropy

- $\forall \tilde{P}, H(\tilde{P}) \geq 0$
- If there are $N$ possible outcomes, $H(\tilde{P}) \leq \log_2 N$
- If $\forall i, p_i = \frac{1}{N}, H(\tilde{P}) = \log_2 N$
- If $\exists i$ such that $p_i = 1, H(\tilde{P}) = 0$
- $H(\tilde{P})$ is a continuous function of $\tilde{P}$

Shannon’s entropy as a measure of information

- For any distribution $\tilde{P}$, $H(\tilde{P})$ is the optimal number of binary questions required on average to determine an outcome drawn from $P$.
- We can extend these ideas to talk about how much information is conveyed by the observation of the outcome of one experiment about the possible outcomes of another (mutual information).
- We can also quantify the difference between two probability distributions (Kullback-Leibler divergence or relative entropy).

Coding Theory Perspective

- Suppose you and I both know the distribution $\tilde{P}$
- I choose an outcome according to $\tilde{P}$
- Suppose I want to send you a message about the outcome
- You and I could agree in advance on the questions
- I can simply send you the answers
- Optimal message length on average is $H(\tilde{P})$
- This generalizes to noisy communication
Entropy of random variables and sets of random variables

For a random variable $X$ taking values $a_1, \ldots, a_n$,

$$H(X) = - \sum_{i=1}^{n} P(X = a_i) \log_2 P(X = a_i)$$

If $X$ is a set of random variables, $H(X)$ is defined as

$$H(X) = - \sum_{X} P(X) \log_2 P(X)$$

Joint Entropy and Conditional Entropy

For random variables $X$ and $Y$, the joint entropy

$$H(X, Y) = - \sum_{X,Y} P(X,Y) \log_2 P(X,Y)$$

Conditional entropy of $X$ given $Y$

$$H(X \mid Y) = - \sum_{X,Y} P(X,Y) \log_2 P(X \mid Y)$$

$$= \sum_{Y} P(Y) H(X \mid Y = a)$$

$$H(X \mid Y = a) = - \sum_{X} P(X,Y = a) \log_2 P(X \mid Y = a)$$

Some Useful results:

$$H(X, Y) \leq H(X) + H(Y)$$

$$H(Y \mid X) \leq H(Y)$$

(When do we have equality?)

Chain rule for Entropy

$$H(X, Y) = H(X) + H(Y \mid X)$$

$$= H(Y) + H(X \mid Y)$$
Example of entropy calculations

\[ P(X = H; Y = H) = 0.2. \quad P(X = H; Y = T) = 0.4 \]
\[ P(X = T; Y = H) = 0.3. \quad P(X = T; Y = T) = 0.1 \]
\[ H(X, Y) = -0.2 \log_2 0.2 + \ldots \approx 1.85 \]
\[ P(X = H) = 0.6. \quad H(X) = 0.97 \]
\[ P(Y = H) = 0.5. \quad H(Y) = 1.0 \]
\[ P(Y = H|X = H) = 0.2/0.6 = 0.333 \]
\[ P(Y = T|X = H) = 1 - 0.333 = 0.667 \]
\[ P(Y = H|X = T) = 0.3/0.4 = 0.75 \]
\[ P(Y = T|X = T) = 0.1/0.4 = 0.25 \]
\[ H(Y|X) = 0.88 \]

Mutual Information
For a random variable \( X \) and \( Y \), the average mutual information between \( X \) and \( Y \)

\[ M(X, Y) = H(X) + H(Y) - H(X, Y) \]

Or by using chain rule
\[ H(X, Y) = H(X) + H(Y | X) = H(Y) + H(X | Y) \]
\[ M(X, Y) = H(X) - H(X | Y) \]
\[ M(X, Y) = H(Y) - H(Y | X) \]

In terms of probability distributions,
\[ M(X, Y) = \sum_{x,y} P(X = a, Y = b) \log_2 \frac{P(X = a, Y = b)}{P(X = a)P(Y = b)} \]

Question: When is \( I(X, Y) = 0 \)?

Relative Entropy
Let \( P \) and \( Q \) be two distributions over random variable \( X \).
The relative entropy (Kullback-Liebler distance) is a measure of "distance" from \( P \) to \( Q \).
\[ D(P \parallel Q) = \sum_x P(X) \log_2 \frac{P(X)}{Q(X)} \]

Note \( D(P \parallel Q) \neq D(Q \parallel P) \)
\[ D(P \parallel Q) \geq 0 \]
\[ D(P \parallel P) = 0 \]
Learning Decision Tree Classifiers

On average, the information needed to convey the class membership of a random instance drawn from nature is

\[ H(\hat{P}) = -\sum_{i=1}^{m} (\hat{P}_i \log \hat{P}_i) = H(X) \]

where \( P \) is an estimate of \( P \) and \( X \) is a random variable with distribution \( P \).

\( S_i \) is the multi-set of examples belonging to class \( C_i \).

The task of the learner then is to extract the needed information from the training set and store it in the form of a decision tree for classification.

Information gain based decision tree learner

Start with the entire training set at the root.

Recursively add nodes to the tree corresponding to tests that yield the greatest expected reduction in entropy (or the largest expected information gain) until some termination criterion is met (e.g., the training data at every leaf node has zero entropy).
Learning Decision Tree Classifiers - Example

Instances –
ordered 3-tuples of attribute values corresponding to

Height (tall, short)
Hair (dark, blonde, red)
Eye (blue, brown)

Training Data

<table>
<thead>
<tr>
<th>Instance</th>
<th>Class label</th>
</tr>
</thead>
<tbody>
<tr>
<td>I₁ (t, d, l)</td>
<td>+</td>
</tr>
<tr>
<td>I₂ (s, d, l)</td>
<td>+</td>
</tr>
<tr>
<td>I₃ (t, b, l)</td>
<td>−</td>
</tr>
<tr>
<td>I₄ (t, r, l)</td>
<td>−</td>
</tr>
<tr>
<td>I₅ (s, b, l)</td>
<td>−</td>
</tr>
<tr>
<td>I₆ (t, b, w)</td>
<td>+</td>
</tr>
<tr>
<td>I₇ (t, d, w)</td>
<td>+</td>
</tr>
<tr>
<td>I₈ (s, b, w)</td>
<td>+</td>
</tr>
</tbody>
</table>

Similarly, Hair is the most informative because it yields the largest reduction in entropy. Test on the value of Hair is chosen to correspond to the root of the decision tree.

In practice, we need some way to prune the tree to avoid overfitting the training data – more on this later.
Minimizing over fitting

- Use roughly the same size sample at every node to estimate entropy – when there is a large data set from which we can sample
- Stop when further split fails to yield statistically significant information gain (estimated from validation set)
- Grow full tree, then prune
- minimize size (tree) + size (exceptions (tree))

Reduced error pruning

Each decision node in the tree is considered as a candidate for pruning

Pruning a decision node consists of
- removing the sub tree rooted at that node,
- making it a leaf node, and
- assigning it the most common label at that node

Reduced error pruning – Example

<table>
<thead>
<tr>
<th>Node</th>
<th>Accuracy gain by Pruning</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>-20%</td>
</tr>
<tr>
<td>B</td>
<td>+10%</td>
</tr>
</tbody>
</table>

Before Pruning  After Pruning
Reduced error pruning

Do until further pruning is harmful:

- Evaluate impact on validation set of pruning each candidate node
- Greedily select a node which most improves the performance on the validation set when the sub tree rooted at that node is pruned

Drawback – holding back the validation set limits the amount of training data available – not desirable when data set is small

Pruning based on whether information gain is significantly greater than zero

Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level

\[
\chi^2 = \frac{(n_{12} - n_{1} \cdot \hat{\gamma})^2}{n_{1}} + \frac{(n_{12} - n_{1} \cdot \hat{\gamma})^2}{n_{2}} = 25 + 25 = 50
\]

This split is significantly better than random with confidence > 99% because \( \chi^2 > 6.64 \)
Pruning based on whether information gain is significantly greater than zero

Evaluate Candidate split to decide if the resulting information gain is significantly greater than zero as determined using a suitable hypothesis testing method at a desired significance level

Example: \( \chi^2 \) statistic
In the 2-class, binary (L,R) split case,
\[ \chi^2 = \sum_{i=1}^{2} \frac{(n_{iL} - n_{iR})^2}{n_{ic}} \]
n\_i of class 1, n\_i of class 2, N=n\_i+n\_2
Split sends p\_i to L and (1-p)\_i to R

Random split would send p\_i of class 1 to L and p\_i of class 2 to L.
The critical value of \( \chi^2 \) depends on the degrees of freedom which is 1 in this case (for a given \( p, n\_i \) fully specifies \( n\_iL, n\_iR \) and \( n\_iR \)).
In general, the number of degrees of freedom can be > 1

The greater the value of \( \chi^2 \) the less likely it is that the split is random. For a sufficiently high value of \( \chi^2 \), the difference between the expected (random) split is statistically significant and we reject the null hypothesis that the split is random.

Degrees of freedom = (Classes -1)(Branches -1)

Rule post-pruning
Convert tree to equivalent set of rules

IF (Outlook = Sunny) ∧ (Humidity = High)
THEN PlayTennis = No
IF (Outlook = Sunny)
THEN PlayTennis = Yes
...
Classification of instances

- Unique classification – possible when each leaf has zero entropy and there are no missing attribute values
- Most likely classification – based on distribution of classes at a node when there are no missing attribute values
- Probabilistic classification – based on distribution of classes at a node when there are no missing attribute values

Handling different types of attribute values

Types of attributes
- Nominal – values are names
- Ordinal – values are ordered
- Cardinal (Numeric) – values are numbers (hence ordered)

Handling numeric attributes

<table>
<thead>
<tr>
<th>Attribute T</th>
<th>40</th>
<th>48</th>
<th>50</th>
<th>54</th>
<th>60</th>
<th>70</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>N</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
</tr>
</tbody>
</table>

Candidate splits

\[ T > \frac{(48+50)}{2}, \quad T > \frac{(60+70)}{2} \]

\[ E(S|T > 49) = \frac{2}{6}(0) + \frac{4}{6}(\left\{\frac{3}{4}\right\} \log_4 \left\{\frac{3}{4}\right\} - \left\{\frac{1}{4}\right\} \log_4 \left\{\frac{1}{4}\right\}) \]

- Sort instances by value of numeric attribute under consideration
- For each attribute, find the test which yields the lowest entropy
- Greedily choose the best test across all attributes
Handling numeric attributes

**Axis-parallel split**

**Oblique split**

Oblique splits cannot be realized by univariate tests

---

Two-way versus multi-way splits

Entropy criterion favors many-valued attributes

Pathological behavior – what if in a medical diagnosis data set, social security number is one of the candidate attributes? $A = \text{value}$ versus $A = \text{--value}$

Solutions

Only two-way splits (CART)

Entropy ratio (C4.5)

\[
\text{EntropyRatio}(S \mid A) = \frac{\text{Entropy}(S \mid A)}{\text{SplitEntropy}(S \mid A)}
\]

\[
\text{SplitEntropy}(S \mid A) = - \sum_{i=1}^{V} \left( \frac{V_i}{|S|} \log_2 \left( \frac{V_i}{|S|} \right) \right)
\]

---

Alternative split criteria

Consider split of set $S$ based on attribute $A$

\[
\text{Impurity}(S \mid A) = \sum_{j=1}^{V} \text{Impurity}(S_j)
\]

Entropy \[\text{Impurity}(Z) = \sum_{i=1}^{C} \left( \frac{|Z_i|}{|Z|} \log_2 \left( \frac{|Z_i|}{|Z|} \right) \right)\]

Gini \[\text{Impurity}(Z) = \sum_{i=1}^{C} \left( \frac{|Z_i|}{|Z|} \right) \left( 1 - \frac{|Z_i|}{|Z|} \right)^2 \]

(Expected rate of error if class label is picked randomly according to distribution of instances in a set)
Alternative split criteria

One-sided split criteria – often useful for exploratory analysis of data

\[ \text{Impurity}(S | A) = \min_{i \in \text{Values}(A)} \{\text{Impurity}(S_i)\} \]

Incorporating Attribute costs

Not all attribute measurements are equally costly or risky in Medical diagnosis

- Blood-Test has cost $150
- Exploratory-Surgery may have a cost of $3000

Goal: Learn a Decision Tree Classifier which minimizes cost of classification

- Tan and Schlimmer (1990) \[ \frac{\text{Gain}(S, A)}{\text{Cost}(A)} \]
- Nunez (1988) \[ \frac{2^{\text{Gain}(S, A)} - 1}{(\text{Cost}(A) + 1)^2} \]

where \( w \in [0, 1] \) determines importance of cost

Incorporating Different Misclassification Costs for different classes

Not all misclassifications are equally costly

- An occasional false alarm about a nuclear power plant meltdown is less costly than the failure to alert when there is a chance of a meltdown

Weighted Gini Impurity

\[ \text{Impurity}(S) = \sum_i \frac{|S_i|}{|S|} \left( \frac{|S_j|}{|S_i|} \right) \]

\( \lambda_{i,j} \) is the cost of wrongly assigning an instance belonging to class \( i \) to class \( j \)
Dealing with Missing Attribute Values (Solution 1)

Sometimes, the fact that an attribute value is missing might itself be informative –
Missing blood sugar level might imply that the physician had reason not to measure it
Introduce a new value (one per attribute) to denote a missing value
Decision tree construction and use of tree for classification proceed as before

Dealing with Missing Attribute Values (Solution 2)

During decision tree construction
Replace a missing attribute value in a training example with the most frequent value found among the instances at the node that have the same class label as the training example

During use of tree for classification
Assign to a missing attribute the most frequent value found at the node (based on the training sample)
Sort the instance through the tree to generate the class label

Dealing with Missing Attribute Values

During decision tree construction
Generate several fractionally weighted training examples based on the distribution of values for the corresponding attribute at the node
During use of tree for classification
Generate multiple instances by assigning candidate values for the missing attribute based on the distribution of instances at the node
Sort each such instance through the tree to generate candidate labels and assign the most probable class label or probabilistically assign class label
Dealing with Missing Attribute Values

\[ n_+=60, n_-=40 \]

\[ (n|A=1)=50 \]
\[ (n|A=0)=10; (n|A=0) \geq 40 \]
\[ (n|A=0, B=1)=40 \]
\[ (n|A=0, B=0)=10 \]

Suppose B is missing
Replaced with most frequent value at the node \( \Rightarrow B=1 \)
Replaced with most frequent value if class is + \( \Rightarrow B=0 \)

Fractional instance based on distribution at the node .. 4/5 for \( B=1 \), 1/5 for \( B=0 \)
Fractional instance based on distribution at the node for class + ..
\[ 1/5 \text{ for } B=0, 0 \text{ for } B=1 \]