Machine Learning (CS 567)

Fall 2008

Time: T-Th 5:00pm - 6:20pm
Location: GFS 118

Instructor: Sofus A. Macskassy (macskass@usc.edu)
Office: SAL 216
Office hours: by appointment

Teaching assistant: Cheol Han (cheolhan@usc.edu)
Office: SAL 229
Office hours: M 2-3pm, W 11-12

Class web page:
http://www-scf.usc.edu/~csci567/index.html
Lecture 11 Outline

• Bayesian Learning
  – Probability theory
  – Bayesian Classification
So far: Discriminative Learning

- We want to distinguish between different classes based on examples of each class.
- Linear classifiers and decision trees generate separating planes between the classes.
- When a new instance needs to be classified, we check on which side of the decision boundaries it falls, and classify accordingly (deterministic).
- This is called discriminative learning, because we defined an explicit boundary that discriminates between the different classes.
- The classification algorithms studied so far (except logistic regression) fall into this category.
Next: *Generative* Learning

- A different idea: use the data to build a model for each of the different classes.
- To classify a new instance, we match it against the different models, then decide which model it resembles more.
- This is called *generative learning*.
- Note that we now categorize whether an instance is more or less likely to come from a given model.
- Also note that you can use these models to generate new data.
Learning Bayesian Networks: Naïve and non-Naïve Bayes

• Hypothesis Space
  – fixed size
  – stochastic
  – continuous parameters

• Learning Algorithm
  – direct computation
  – eager
  – batch
But first... basic probability theory

- Random variables
- Distributions
- Statistical formulae
Random Variables

- A random variable is a random number (or value) determined by chance, or more formally, drawn according to a probability distribution
  - The probability distribution can be estimated from observed data (e.g., throwing dice)
  - The probability distribution can be synthetic
  - Discrete & continuous variables

- Typical random variables in Machine Learning Problems
  - The input data
  - The output data
  - Noise

- Important concept in learning: The data generating model
  - E.g., what is the data generating model for:
    i) Throwing dice
    ii) Regression
    iii) Classification
    iv) For visual perception
Why have Random Variables?

- Our goal is to predict our target variable
- We are not given the true (presumably deterministic) function
- We are only given observations
  - Can observe the number of times a dice lands on 4
  - Can estimate the probability, given the input, that the dice will land on 4
  - But we don’t know where the dice will land
  - Can only make a guess to the most likely value of the dice, given the input.
**Distributions**

- The random variables only take on discrete values
  - E.g., throwing dice: possible values: \( v_i \in \{1,2,3,4,5,6\} \)
- The probabilities sum to 1
  \[
  \sum_i P(v_i) = 1
  \]
- Discrete distributions are particularly important in classification
- Probability Mass Function or Frequency Function (normalized histogram)

A “non fair” dice
Classic Discrete Distributions (I)

**Bernoulli Distribution**

- A Bernoulli random variable takes on only two values, i.e., 0 and 1.
- \( P(1)=p \) and \( P(0)=1-p \) or in compact notation:

\[
P(x) = \begin{cases} 
p^x (1 - p)^{1-x}, & \text{if } x = 0 \text{ or } x = 1 \\
0, & \text{otherwise}
\end{cases}
\]

- The performance of a fixed number of trials with fixed probability of success (\( p \)) on each trial is known as a *Bernoulli trial*.

\[P(x)\text{ for } p=0.3\]
Binomial Distribution

- Like Bernoulli distribution: binary input variables: 0 or 1, and probability $P(1)=p$ and $P(0)=1-p$
- What is the probability of $k$ successes, $P(k)$, in a series of $n$ independent trials? ($n \geq k$)
- $P(k)$ is a binomial random variable:

$$P(k) = \binom{n}{k} p^k (1-p)^{n-k}$$

where

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

- Bernoulli distribution is a subset of the binomial distribution (i.e., $n=1$)
Binomial Distribution

Binomial p=0.5

Binomial p=0.25

Binomial p=0.1
Classic Discrete Distributions (III)

Multinomial Distribution

- A generalization of the binomial distribution to multiple outputs (i.e., multiple classes can be categorized instead of just one class)
- \( n \) independent trials can result in one of \( r \) types of outcomes, where each outcome \( c_r \) has a probability \( P(c_r) = p_r (p_r = 1) \).
- What is the probability \( P(n_1, n_2, \ldots, n_r) \), i.e., the probability that in \( n \) trials, the frequency of the \( r \) classes is \( (n_1, n_2, \ldots, n_r) \)? This is a multinomial random variable:

\[
P(n_1, n_2, \ldots, n_r) = \binom{n}{n_1 n_2 \cdots n_r} p_1^{n_1} p_2^{n_2} \cdots p_r^{n_r}
\]

where

\[
\binom{n}{n_1 n_2 \cdots n_r} = \frac{n!}{n_1! n_2! \cdots n_r!}
\]
Continuous Probability Distributions

- The random variables take on real values.
- Continuous distributions are discrete distributions where the number of discrete values goes to infinity while the probability of each discrete value goes to zero.
- Probabilities become densities.
- Probability density integrates to 1.

\[
\int_{-\infty}^{+\infty} p(x) \, dx = 1
\]
Continuous Probability Distributions (cont’d)

- Probability Density Function $p(x)$

- Probability of an event:

$$P(a < x < b) = \int_a^b p(x)dx = 1$$
Normal Distribution

- The most used distribution
  \[ P(x) = \frac{1}{\sqrt{(2\pi)^d \|\Sigma\|}} \exp \left( -\frac{1}{2} (x - \mu)^T \Sigma^{-1} (x - \mu) \right) \]
  
- Also called Gaussian distribution after C.F. Gauss who proposed it

- Justified by the Central Limit Theorem:
  - Roughly: “if a random variable is the sum of a large number of independent random variables it is approximately normally distributed”
  - Many observed variables are the sum of several random variables

- Shorthand:
  \[ x \sim N(\mu, \Sigma) \]
Classic Continuous Distributions (II)

**Uniform Distribution**

- All data is equally probably within a bounded region \( R \), \( p(x) = 1/R \)
The expected value, mean or average of the random variable \( x \) is defined by:

\[
E[x] = \mu = \sum_{x \in \mathcal{X}} xP(x) = \sum_{i=1}^{m} v_i p_i.
\]

More generally, if \( f(x) \) is any function of \( x \), the expected value of \( f \) is defined by:

\[
E[f(x)] = \sum_{x \in \mathcal{X}} f(x)P(x).
\]

This is also called the center of mass.

Note that forming an expected values is linear, in that if \( \alpha_1 \) and \( \alpha_2 \) are arbitrary constants, then we have

\[
E[\alpha_1 f_1(x) + \alpha_2 f_2(x)] = \alpha_1 E[f_1(x)] + \alpha_2 E[f_2(x)].
\]
Expected Value

- **General rules of thumb:**

\[ E[g(x)] \neq g(E[x]) \]
\[ E[\alpha x] = \alpha E[x] \]
\[ E[x + y] = E[x] + E[y] \]
\[ E \left[ \sum_i \alpha_i x_i \right] = \sum_i \alpha_i E[x_i] \]

- **In general:**

\[ E[x \cdot y] \neq E[x] \cdot E[y] \]

- **Given a FINITE sample data, the Expectation is:**

\[ E[x] = \frac{1}{N} \sum_{i=1}^{N} x_i \]
Variance and Standard Deviation

- **Variance**: \( Var[x] = \sigma^2 = E[(x - \mu)^2] = \sum_{x \in \mathcal{X}} (x - \mu)^2 P(x) \)

- \( \sigma \) is the *standard deviation* of \( x \). The variance is never negative and approaches 0 as the probability mass is centered at one point.

- The standard deviation is a simple measure of how far values of \( x \) are likely to depart from the mean.
  
  - i.e., the standard or typical amount one should expect a randomly drawn value of \( x \) to deviate or differ from \( \mu \).
Sample variance and covariance

- **Sample Variance**
  \[ Var[x] = \frac{1}{N - 1} \sum_{i=1}^{N} (x_i - E[x])^2 \]
  - Why division by (N-1)? This is to obtain an unbiased estimate of the variance. (unbiased estimate: \( E[\hat{x}] = x \))

- **Covariance**
  \[ Cov[x, y] = E \left[ (x - E[x])(y - E[y]) \right] \]

- **Sample Covariance**
  \[ Cov[x, y] = \frac{1}{N - 1} \sum_{i=1}^{N} (x_i - E[x])(y_i - E[y]) \]
  \[ Cov[x] = \frac{1}{N - 1} \sum_{i=1}^{N} (x_i - E[x])(x_i - E[x])^T \]
Biased vs. Unbiased variance

- **Biased variance:** 
  \[ V = \frac{1}{n} \sum_{i=1}^{n} (x_i - X')^2 \]

- **“Anti-biased” variance:** 
  \[ V^* = \frac{1}{n} \sum_{i=1}^{n} (x_i - X_i')^2 \]

\[ \Rightarrow (n - 1)^2 V^* = n^2 V \]

\[ \Rightarrow \sqrt{V^*} \cdot V = \frac{1}{n-1} \sum_{i=1}^{n} (x_i - X')^2 \]
Sample variance

\[ s^2 \text{ is sample variance, } \sigma^2 \text{ is true variance, } \mu \text{ is true mean} \]

\[
E[s^2] = E \left[ \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2 \right] \quad \text{(where } \bar{x} = E[x] \text{ is the sample mean)}
\]

\[
= \frac{1}{n-1} \sum_{i=1}^{n} (x_i - \bar{x})^2
\]

\[
= \frac{1}{n-1} \sum_{i=1}^{n} ((x_i - \mu) - (\bar{x} - \mu))^2
\]

\[
= \frac{1}{n-1} \sum_{i=1}^{n} E [(x_i - \mu)^2] - 2E [(x_i - \mu) - (\bar{x} - \mu)] + E [(\bar{x} - \mu)^2]
\]

\[
= \frac{1}{n-1} \sum_{i=1}^{n} \left[ \sigma^2 - 2 \left( \frac{1}{n} \sum_{j=1}^{n} E [(x_i - \mu)(x_j - \mu)] \right) + \frac{1}{n^2} \sum_{j=1}^{n} \sum_{k=1}^{n} E [(x_j - \mu)(x_k - \mu)] \right]
\]

\[
= \frac{1}{n-1} \sum_{i=1}^{n} \left[ \sigma^2 - 2 \frac{\sigma^2}{n} + \frac{\sigma^2}{n} \right]
\]

\[
= \frac{1}{n-1} \sum_{i=1}^{n} \frac{(n-1)\sigma^2}{n}
\]

\[
= \frac{(n-1)\sigma^2}{n - 1}
\]

\[
= \sigma^2
\]
Conditional Probability

- \( P(x|y) \) is the probability of the occurrence of event \( x \) given that \( y \) occurred and is given as:

\[
P(x|y) = \frac{P(x \cap y)}{P(y)}
\]

- Knowing that \( y \) occurred reduces the sample space to \( y \), and the part of it where \( x \) also occurred is \((x,y)\).
Conditional Probability

- \( P(x|y) \) is the probability of the occurrence of event \( x \) given that \( y \) occurred and is given as:

\[
P(x|y) = \frac{P(x \cap y)}{P(y)}
\]

- Knowing that \( y \) occurred reduces the sample space to \( y \), and the part of it where \( x \) also occurred is \((x,y)\).

- This is only defined if \( P(y) > 0 \). Also, because \( \cap \) is commutative, we have:

\[
P(x \cap y) = P(x|y)P(y) = P(y|x)P(x)
\]
Statistical Independence

- If $x$ and $y$ are independent then we have
  \[ P(x|y) = P(x) \]
- From there it follows that
  \[ P(x \cap y) = P(x)P(y) \]
- In other words, knowing that $y$ occurred does not change the probability that $x$ occurs (and vice versa).
Bayes Rule

- **Remember:**
  \[ P(x \cap y) = P(x|y)P(y) = P(y|x)P(x) \]

- **Bayes Rule:**
  \[ P(y|x) = \frac{P(x|y)P(y)}{P(x)} \]

- **Interpretation**
  - \( P(y) \) is the **PRIOR** knowledge about \( y \)
  - \( X \) is new evidence to be incorporated to update my belief about \( y \).
  - \( P(x|y) \) is the **LIKELIHOOD** of \( x \) given that \( y \) was observed.
  - Both prior and likelihood can often be generated beforehand, e.g., by histogram statistics.
  - \( P(x) \) is a normalizing factor, corresponding to the *marginal distribution* of \( x \). Often it need not be evaluated explicitly, but it can become a great computational burden. “\( P(x) \) is an enumeration of all possible combinations of \( x \), and the probability of their occurrence.”
  - \( P(y|x) \) is the **POSTERIOR** probability of \( y \), i.e., the belief in \( y \) after one discovers \( x \).
Learning Bayesian Networks: Naïve and non-Naïve Bayes

• Hypothesis Space
  – fixed size
  – stochastic
  – continuous parameters

• Learning Algorithm
  – direct computation
  – eager
  – batch
Roles for Bayesian Methods

Provides practical learning algorithms:
- Naive Bayes learning
- Bayesian belief network learning
- Combine prior knowledge (prior probabilities) with observed data
- Requires prior probabilities

Provides useful conceptual framework
- Provides “gold standard” for evaluating other learning algorithms
Bayesian Learning

Bayes Theorem

- Consider hypothesis space $H$

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

- $P(h) = \text{prior prob. of hypothesis } h \in H$
- $P(D) = \text{prior prob. of training data } D$
- $P(h|D) = \text{probability of } h \text{ given } D$
- $P(D|h) = \text{probability of } D \text{ given } h$
Choosing Hypotheses

Natural choice is most probable hypothesis given the training data, or *maximum a posteriori* hypothesis $h_{MAP}$:

$$h_{MAP} = \arg\max_{h \in H} P(h|D)$$

$$= \arg\max_{h \in H} \frac{P(D|h)P(h)}{P(D)}$$

$$= \arg\max_{h \in H} P(D|h)P(h)$$

If we assume $P(h_i) = P(h_j)$ then can further simplify, and choose the *maximum likelihood* (ML) hypothesis

$$h_{ML} = \arg\max_{h \in H} P(D|h)$$
Bayes Theorem: Example

Does patient have cancer or not?

- A patient takes a lab test and the result comes back positive. The test returns a correct positive result in 98% of the cases in which the disease is actually present, and a correct negative result in 97% of the cases in which the disease is not present. Furthermore, .008 of the entire population have this cancer.

\[
\begin{align*}
P (\text{cancer}) &= 0.008 \\
P (\neg \text{cancer}) &= 0.992 \\
P (+|\text{cancer}) &= 0.98 \\
P (+|\neg \text{cancer}) &= 0.03 \\
P (-|\text{cancer}) &= 0.02 \\
P (-|\neg \text{cancer}) &= 0.97
\end{align*}
\]
Bayes Theorem: Example

P (cancer) = 0.008  P (¬cancer) = 0.992
P (+|cancer) = 0.98  P (-|cancer) = 0.02
P (+|¬cancer) = 0.03  P (-|¬cancer) = 0.97

A positive test result comes in for a patient.
What is the $h_{\text{MAP}}$?

$$P(+|\text{cancer})P(\text{cancer}) = (0.98)(0.008) = 0.0078$$
$$P(+|\neg \text{cancer})P(\neg \text{cancer}) = (0.03)(0.992) = 0.0298$$

$h_{\text{MAP}} = \neg \text{cancer}$
Brute Force MAP Learner

1. For each hypothesis $h$ in $H$, calculate the posterior probability

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

2. Output the hypothesis $h_{\text{MAP}}$ with the highest posterior probability

$$h_{\text{MAP}} = \arg\max_{h \in H} P(h|D)$$
Evolution of Posterior Probs

- As data is added, certainty of hypotheses increases.
Classifying New Instances

So far we've sought the most probable hypothesis given the data $D$ (i.e., $h_{\text{MAP}}$)

Given new instance $\mathbf{x}$, what is its most probable classification?

$h_{\text{MAP}}(\mathbf{x})$ is not the most probable classification!
Classification Example

Consider:

• Three possible hypotheses:
  \[ P(h_1|D) = 0.4, \ P(h_2|D) = 0.3, \ P(h_3|D) = 0.3 \]

• Given new instance \( x \),
  \[ h_1(x) = +, \ h_2(x) = -, \ h_3(x) = - \]

• What’s \( h_{\text{MAP}}(x) \) ?

• What's most probable classification of \( x \)?
Bayes Optimal Classifier

Bayes optimal classification:

\[
\arg\max_{v \in V} \sum_{h \in H} P(v|h) P(h|D)
\]

Example:

- \(P(h_1|D) = .4\), \(P(-|h_1) = 0\), \(P(+|h_2) = 1\)
- \(P(h_2|D) = .3\), \(P(-|h_2) = 1\), \(P(+|h_3) = 0\)
- \(P(h_3|D) = .3\), \(P(-|h_3) = 1\), \(P(+|h_3) = 0\),

therefore

\[
\sum_{h \in H} P(+|h) P(h|D) = 0.4
\]
\[
\sum_{h \in H} P(-|h) P(h|D) = 0.6
\]

MAP class
Gibbs Classifier

Bayes optimal classifier provides best result, but can be expensive if many hypotheses.

Gibbs algorithm:
1. Choose one hypothesis at random, according to \( P(h|D) \)
2. Use this one to classify new instance
Error of Gibbs

Noteworthy fact [Haussler 1994]: Assume target concepts are drawn at random from $H$ according to priors on $H$. Then:

$$E[\text{error}_{\text{Gibbs}}] \leq 2E[\text{error}_{\text{BayesOptimal}}]$$

Suppose correctly, uniform prior distribution over $H$, then

• Pick any hypothesis consistent with the data, with uniform probability

• Its expected error no worse than twice Bayes optimal
Naive Bayes Classifier

Along with decision trees, neural networks, $k$NN, one of the most practical and most used learning methods.

When to use:
- Moderate or large training set available
- Attributes that describe instances are conditionally independent given classification

Successful applications:
- Diagnosis
- Classifying text documents
Naïve Bayes Assumption

- Suppose the features $x_i$ are discrete
- Assume the $x_i$ are conditionally independent given $y$.
- In other words, assume that:

  $$P(x_i|y) = P(x_i|y, x_j), \forall i, j$$

- Then we have:

  $$P(x_1, x_2, \cdots, x_n|y) = P(x_1|y)P(x_2|y)\cdots P(x_n|y)$$

- For binary features, instead of $O(2^n)$ numbers to describe a model, we only need $O(n)$!
Graphical Representation of Naïve Bayes Model

• Each node contains a probability table
  – $y$: $P(y = k)$
  – $x_j$: $P(x_j = v | y = k)$ “class conditional probability”

• Interpret as a generative model
  – Choose the class $k$ according to $P(y = k)$
  – Generate each feature independently according to $P(x_j = v | y = k)$
  – The feature values are conditionally independent
    $P(x_i, x_j | y) = P(x_i | y) \cdot P(x_j | y)$
Naïve Bayes Algorithm

Naïve_Bayes_Learn(examples)

For each target value $y_j$

$P(\hat{y}_j) \leftarrow \text{estimate } P(y_j)$

For each attribute value $x_i$

$P(\hat{x}_i|y_j) \leftarrow \text{estimate } P(x_i|y_j)$

Classify_New_Instance($\mathbf{x}$)

$y_{NB} = \arg\max_{y_j \in Y} P(y_j) \prod_i P(x_i|y_j)$
Naïve Bayes: Example

- Consider the \emph{PlayTennis} problem and new instance \(<\text{Outlook} = \text{sun}, \text{Temp} = \text{cool}, \text{Humid} = \text{high}, \text{Wind} = \text{strong}>\)

Want to compute:

\[ y_{\text{NB}} = \arg\max_{y_j \in Y} P(y_j) \prod_i P(x_i | y_j) \]

\[ P(y) P(\text{sun} | y) P(\text{cool} | y) P(\text{high} | y) P(\text{strong} | y) = .005 \]

\[ P(n) P(\text{sun} | n) P(\text{cool} | n) P(\text{high} | n) P(\text{strong} | n) = .021 \]

- So, \( y_{\text{NB}} = n \)
Naïve Bayes: Subtleties

• Conditional independence assumption is often violated

\[ P(x_1, x_2 \ldots x_n, | y_j) = \prod_i P(x_i | y_j) \]

• ...but it works surprisingly well anyway. Note don't need estimated posteriors \( P(y_j|x) \) to be correct; need only that

\[ \arg\max_{y_j \in Y} P(y_j|x) = \arg\max_{y_j \in Y} P(y_j) \prod_i P(x_i | y_j) \]

• See Domingos & Pazzani [1996] for analysis

• Naïve Bayes posteriors often unrealistically close to 1 or 0
Decision Boundary of naïve Bayes with binary features

• The parameters of the model are \( \theta_{i,1} = P(x_i = 1|y=1), \theta_{i,0} = P(x_i = 1|y=0), \theta_1 = P(y=1) \)

• What is the decision surface?

\[
\frac{P(y = 1|x)}{P(y = 0|x)} = \frac{P(y = 1) \prod_{i=1}^{n} P(x_i|y = 1)}{P(y = 0) \prod_{i=1}^{n} P(x_i|y = 0)}
\]

• Using the log trick, we get:

\[
\log \frac{P(y = 1|x)}{P(y = 0|x)} = \log \frac{P(y = 1)}{P(y = 0)} + \sum_{i=1}^{n} \log \frac{P(x_i|y = 1)}{P(x_i|y = 0)}
\]

• Note that in the equation above, the \( x_i \) would be 1 or 0, depending on the values were present in the instance.
Decision Boundary of naïve Bayes with binary features

\begin{align*}
\text{let:} \quad w_0 &= \log \frac{P(y = 1)}{P(y = 0)} \\
w_{i,1} &= \log \frac{P(x_i = 1 | y = 1)}{P(x_i = 1 | y = 0)} \\
&\quad \quad \quad \quad \frac{P(x_i = 0 | y = 0)}{P(x_i = 0 | y = 0)} \\
\end{align*}

- We can re-write the decision boundary as:

\begin{align*}
\log \frac{P(y = 1 | x)}{P(y = 0 | x)} &= w_0 + \sum_{i=0}^{n} (w_{i,1}x_i + w_{i,0}(1 - x_i)) \\
&= w_0 + \sum_{i=1}^{n} w_{i,0} + \sum_{i=1}^{n} (w_{i,1} - w_{i,0})x_i
\end{align*}

- This is a \textit{linear decision boundary}!
Representing $P(x_j | y)$

Many representations are possible

– Univariate Gaussian
  • if $x_j$ is a continuous random variable, then we can use a normal distribution and learn the mean $\mu$ and variance $\sigma^2$

– Multinomial/Binomial
  • if $x_j$ is a discrete random variable, $x_j \in \{v_1, ..., v_m\}$, then we construct a conditional probability table

– Discretization
  • convert continuous $x_j$ into a discrete variable

– Kernel Density Estimates
  • apply a kind of nearest-neighbor algorithm to compute $P(x_j | y)$ in neighborhood of query point
Representing $P(x_j | y)$ – Discrete Values

- Multinomial/Binomial

  • if $x_j$ is a discrete random variable, $x_j \in \{v_1, ..., v_m\}$, then we construct the conditional probability table

<table>
<thead>
<tr>
<th></th>
<th>$y = 1$</th>
<th>$y = 2$</th>
<th>...</th>
<th>$y = K$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_j = v_1$</td>
<td>$P(x_j = v_1</td>
<td>y = 1)$</td>
<td>$P(x_j = v_1</td>
<td>y = 2)$</td>
</tr>
<tr>
<td>$x_j = v_2$</td>
<td>$P(x_j = v_2</td>
<td>y = 1)$</td>
<td>$P(x_j = v_2</td>
<td>y = 2)$</td>
</tr>
<tr>
<td>$x_j = v_m$</td>
<td>$P(x_j = v_m</td>
<td>y = 1)$</td>
<td>$P(x_j = v_m</td>
<td>y = 2)$</td>
</tr>
</tbody>
</table>

$$P(x_j = v_l | y = k) = \frac{\text{number of instances for which } x_j = v_l \text{ and } y = k}{\text{number of instances for which } y = k}$$
Many discretization algorithms have been studied. One of the best is based on mutual information [Fayyad & Irani 93].

- To discretize feature $x_j$, grow a decision tree considering only splits on $x_j$. Each leaf of the resulting tree will correspond to a single value of the discretized $x_j$. 

---

![Diagram of discretization via mutual information](image-url)
Discretization via Mutual Information

- Many discretization algorithms have been studied. One of the best is based on mutual information [Fayyad & Irani 93].
  - To discretize feature $x_j$, grow a decision tree considering only splits on $x_j$. Each leaf of the resulting tree will correspond to a single value of the discretized $x_j$.
  - Stopping rule (applied at each node). Stop when

$$I(x_j; y) < \frac{\log_2(N - 1)}{N} + \frac{\Delta}{N}$$

$$\Delta = \log_2(3^K - 2) - [K \cdot H(S) - K_l \cdot H(S_l) - K_r \cdot H(S_r)]$$

- where $S$ is the training data in the parent node; $S_l$ and $S_r$ are the examples in the left and right child. $K$, $K_l$, and $K_r$ are the corresponding number of classes present in these examples. $I$ is the mutual information, $H$ is the entropy, and $N$ is the number of examples in the node.
Many discretization algorithms have been studied. One of the best is based on mutual information [Fayyad & Irani 93].

- An alternative encoding [Macskassy et al. 02] is similar to thermometer coding as used in neural networks [Gallant 93].
- Rather than encode continuous values into one value, encode into \(2^v\) values, where \(v\) is the number of split points created.
- Each encoded binary value represents whether the continuous value was greater or less-than-or-equal to one of the split points.

Encoding: \{\text{>split1,} \leq \text{split1,} \text{>split2,} \leq \text{split2,} \text{>split3,} \leq \text{split3,} \text{>split4,} \leq \text{split4}\}

\[ p: \{1, 0, 0, 1, 0, 1, 0, 0, 1\} \]
Kernel Density Estimators

- Define \( K(x_j, x_{i,j}) = \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\left(\frac{x_j - x_{i,j}}{\sigma}\right)^2\right) \) to be the Gaussian Kernel with parameter \( \sigma \).

- Estimate

\[
P(x_j | y = k) = \frac{\sum\{i | y = k\} K(x_j, x_{i,j})}{N_k}
\]

where \( N_k \) is the number of training examples in class \( k \).
Kernel Density Estimators (2)

- This is equivalent to placing a Gaussian “bump” of height $1/N_k$ on each training data point from class $k$ and then adding them up.
Kernel Density Estimators (3)

- Resulting probability density
The value chosen for $\sigma$ is critical

$\sigma=0.15$???

$\sigma=0.50$
Learning the Probability Distributions by Direct Computation

- \( P(y=k) \) is just the fraction of training examples belonging to class \( k \).
- For multinomial variables, \( P(x_j = v \mid y = k) \) is the fraction of training examples in class \( k \) where \( x_j = v \).
- For Gaussian variables, \( \hat{\mu}_{jk} \) is the average value of \( x_j \) for training examples in class \( k \). \( \hat{\sigma}_{jk} \) is the sample standard deviation of those points:

\[
\hat{\sigma}_{jk} = \sqrt{\frac{1}{N_k} \sum_{i \mid y_i = k} (x_{i,j} - \hat{\mu}_{jk})^2}
\]
Improved Probability Estimates via Laplace Corrections

- When we have very little training data, direct probability computation can give probabilities of 0 or 1. Such extreme probabilities are “too strong” and cause problems.
- Suppose we are estimating a probability $P(z)$ and we have $n_0$ examples where $z$ is false and $n_1$ examples where $z$ is true. Our direct estimate is:

$$P(z = 1) = \frac{n_1}{n_0 + n_1}$$

- Laplace Estimate. Add 1 to the numerator and 2 to the denominator:

$$P(z = 1) = \frac{n_1 + 1}{n_0 + n_1 + 2}$$

This says that in the absence of any evidence, we expect $P(z) = 0.5$, but our belief is weak (equivalent to 1 example for each outcome).
- Generalized Laplace Estimate. If $z$ has $K$ different outcomes, then we estimate it as:

$$P(z = 1) = \frac{n_1 + 1}{n_0 + \cdots + n_{K-1} + K}$$
Naïve Bayes Applied to Diabetes Diagnosis

- Bayes nets and causality
  - Bayes nets work best when arrows follow the direction of causality
    - two things with a common cause are likely to be conditionally independent given the cause; arrows in the causal direction capture this independence
  - In a Naïve Bayes network, arrows are often not in the causal direction
    - diabetes does not cause pregnancies
    - diabetes does not cause age
  - But some arrows are correct
    - diabetes does cause the level of blood insulin and blood glucose
Non-Naïve Bayes

- Manually construct a graph in which all arcs are causal
- Learning the probability tables is still easy. For example, \( P(\text{Mass} \mid \text{Age, Preg}) \) involves counting the number of patients of a given age and number of pregnancies that have a given body mass
- Classification:

\[
P(D = d \mid A, P, M, I, G) = \frac{P(I \mid D = d) P(G \mid I, D = d) P(D = d \mid A, M, P)}{P(I, G)}
\]
Bayesian Belief Network

Network represents a set of conditional ind. assertions:

- Each node is asserted to be conditionally ind. of its non-descendants, given its immediate predecessors.
- Directed acyclic graph
Bayesian Belief Network

Represents joint probability distribution over all variables

- e.g., \( P(\text{Storm}, \text{BusTourGroup}, \ldots, \text{ForestFire}) \)
- in general, \( P(y_1, \ldots, y_n) = \prod_{i=1}^{n} P(y_i | \text{Parents}(Y_i)) \) where \( \text{Parents}(Y_i) \) denotes immediate predecessors of \( Y_i \) in the graph
- Therefore, the joint distribution is fully defined by graph, plus the CPTS: \( P(y_i | \text{Parents}(Y_i)) \)
Inference in Bayesian Nets

How can one infer the (probabilities of) values of one or more network variables, given observed values of others?

• Bayes net contains all information needed for this inference
• If only one variable with unknown value, easy to infer it
• Easy if BN is a “polytree”
• In general case, problem is NP hard (#P-complete, Roth 1996).
In practice, can succeed in many cases

- Exact inference methods work well for some network structures (small “induced width”)
- Monte Carlo methods “simulate” the network randomly to calculate approximate solutions
- Now used as a primitive in more advanced learning and reasoning scenarios. (e.g., in relational learning)
Learning Bayes Nets

Suppose structure known, variables partially observable

e.g., observe *ForestFire, Storm, BusTourGroup, Thunder*, but not *Lightning, Campfire*...

Similar to training neural network with hidden units

• In fact, can learn network conditional probability tables using gradient ascent!

• Converge to network \( h \) that (locally) maximizes \( P(D|h) \)
Gradient Ascent for BNs

Let $w_{ijk}$ denote one entry in the conditional probability table for variable $Y_i$ in the network

$$w_{ijk} = P(Y_i = y_{ij} | \text{Parents}(Y_i) = u_{jk} \text{ values})$$

e.g., if $Y_i = \text{Campfire}$, then $u_{ik}$ might be

$$<\text{Storm} = T, \text{BusTourGroup} = F>$$

Perform gradient ascent by repeatedly:

1. Update all $w_{ijk}$ using training data $D$

$$w_{ijk} \leftarrow w_{ijk} + \eta \frac{\sum_{d \in D} P_h(y_{ij}, u_{jk} | d)}{w_{ijk}}$$

2. Then, renormalize the $w_{ijk}$ to assure

$$\sum_j w_{ijk} = 1, \ 0 \leq w_{ijk} \leq 1$$
Unknown Structure

When structure unknown...

- Algorithms use greedy search to add/subtract edges and nodes
- Active research topic

Somewhat like decision trees: searching for a discrete graph structure
Belief Networks

- Combine prior knowledge with observed data
- Impact of prior knowledge (when correct!) is to lower the sample complexity
- Active research area (UAI)
  - Extend from Boolean to real-valued variables
  - Parameterized distributions instead of tables
  - Extend to first-order systems
  - More effective inference methods
  - ...
Naïve Bayes Summary

- **Advantages of Bayesian networks**
  - Produces stochastic classifiers
    - can be combined with utility functions to make optimal decisions
  - Easy to incorporate causal knowledge
    - resulting probabilities are easy to interpret
  - Very simple learning algorithms
    - if all variables are observed in training data

- **Disadvantages of Bayesian networks**
  - Fixed sized hypothesis space
    - may underfit or overfit the data
    - may not contain any good classifiers if prior knowledge is wrong
  - Harder to handle continuous features
Evaluation of Naïve Bayes

<table>
<thead>
<tr>
<th>Criterion</th>
<th>LMS</th>
<th>Logistic</th>
<th>LDA</th>
<th>Trees</th>
<th>NNbr</th>
<th>Nets</th>
<th>NB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mixed data</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Missing values</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>some</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Outliers</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>disc</td>
</tr>
<tr>
<td>Monotone transformations</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>no</td>
<td>some</td>
<td>disc</td>
</tr>
<tr>
<td>Scalability</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Irrelevant inputs</td>
<td>no</td>
<td>no</td>
<td>no</td>
<td>some</td>
<td>no</td>
<td>no</td>
<td>some</td>
</tr>
<tr>
<td>Linear combinations</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>some</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Interpretable</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>Accurate</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
<td>no</td>
<td>no</td>
<td>yes</td>
<td>yes</td>
</tr>
</tbody>
</table>

- Naïve Bayes is very popular, particularly in natural language processing and information retrieval where there are many features compared to the number of examples.
- In applications with lots of data, Naïve Bayes does not usually perform as well as more sophisticated methods.