

Bayesian Learning

Michael van Lent

The Big Picture

- Problem
 - Classification
 - Stochastic Modeling
- Feedback
 - Supervised learning
- Knowledge Representation
 - Bayesian classifiers
 - Bayesian Networks
- Knowledge Source
 - Examples

Background

- Most learning approaches learn a single best guess
 - Learning algorithm selects a single hypothesis
 - Hypothesis = Decision tree, rule set, neural network...
- Probabilistic learning
 - Learn the probability that a hypothesis is correct
 - Identify the most probable hypothesis
 - Competitive with other learning techniques
 - A single example doesn't eliminate any hypothesis
- Notation
 - $P(h)$: probability that hypothesis h is correct
 - $P(D)$: probability of seeing data set D
 - $P(D|h)$: probability of seeing data set D given that h is correct
 - $P(h|D)$: probability that h is correct given that D is seen

Bayes Rule

- Bayes rule is the foundation of Bayesian learning

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

- As $P(D|h)$ increases, so does $P(h|D)$
- As $P(h)$ increases, so does $P(h|D)$
- As $P(D)$ increases, probability of $P(h|D)$ decreases

Example

- A monster has two attacks, A and B:
 - Attack A does 11-20 damage and is used 10% of the time
 - Attack B does 16–115 damage and is used 90% of the time
 - You have counters A' (for attack A) and B' (for attack B)
- If an attack does 16-20 damage, which counter to use?
 - $P(A|\text{damage}=16-20)$ greater or less than 50%?
- We don't know $P(A|16-20)$
 - We do know $P(A)$, $P(B)$, $P(16-20|A)$, $P(16-20|B)$
 - We only need $P(16-20)$
 - $P(16-20) = P(A) P(16-20|A) + P(B) P(16-20|B)$

Example (cont'd)

- Some probabilities
 - $P(A) = 10\%$
 - $P(B) = 90\%$
 - $P(16-20|A) = 50\%$
 - $P(16-20|B) = 5\%$

$$P(A|16-20) = \frac{P(16-20|A)P(A)}{P(16-20)}$$

$$P(A|16-20) = \frac{0.5(0.1)}{(0.1)(0.5) + (0.9)(0.05)}$$

$$P(A|16-20) = \frac{0.05}{0.05 + 0.045} = \frac{0.05}{0.095} = 0.5263 = 52.63\%$$

- So counter A' is the slightly better choice

Bayes Optimal Classifier

- Given data D , what's the probability that a new example falls into category c
- $P(\text{example}=c|D)$ or $P(c|D)$
- Best classification is highest $P(c|D)$

$$\max_{c_i \in C} P(c_i|D) = \max_{c_i \in C} \sum_{h_j \in H} P(c_i|h_j)P(c_j|D)$$

- This approach tends to be computationally expensive
 - Space of hypothesis is generally very large

Example Problem

Classify how I should react to an object in the world

- Facts about any given object include:
 - Allegiance = < friendly, neutral, enemy >
 - Health = < low, medium, full >
 - Animate = < true, false >
 - RelativeHealth = < weaker, same, stronger >
- Output categories include:
 - Reaction = Attack
 - Reaction = Ignore
 - Reaction = Heal
 - Reaction = Eat
 - Reaction = Run



- <friendly, low, true, weaker> => Heal
- <neutral, low, true, same> => Heal
- <enemy, low, true, stronger> => Attack
- <enemy, medium, true, weaker> => Attack

Naïve Bayes Classifier

- Each example is a set of feature values
 - friendly, low, true, weaker
- Given a set of feature values, find the most probable category
- Which is highest:
 - $P(\text{Attack} \mid \text{friendly, low, true, weaker})$
 - $P(\text{Ignore} \mid \text{friendly, low, true, weaker})$
 - $P(\text{Heal} \mid \text{friendly, low, true, weaker})$
 - $P(\text{Eat} \mid \text{friendly, low, true, weaker})$
 - $P(\text{Run} \mid \text{friendly, low, true, weaker})$

$$c_{nb} = \max_{c_i \in C} P(c_i \mid f_1, f_2, f_3, f_4)$$

Calculating Naïve Bayes Classifier

$$C_{nb} = \max_{c_i \in C} P(c_i | f_1, f_2, f_3, f_4)$$

$$C_{nb} = \max_{c_i \in C} \frac{P(f_1, f_2, f_3, f_4 | c_i)P(c_i)}{P(f_1, f_2, f_3, f_4)}$$

$$C_{nb} = \max_{c_i \in C} P(f_1, f_2, f_3, f_4 | c_i)P(c_i)$$

- Simplifying assumption: each feature in the example is independent
 - Value of Allegiance doesn't affect value of Health, Animate, or RelativeHealth

$$P(f_1, f_2, f_3, f_4 | c_i) = \prod_j P(f_j | c_i)$$

$$C_{nb} = \max_{c_i \in C} P(c_i) \prod_j P(f_j | c_i)$$

Example

- Slightly modified 13 examples:
 - <friendly, low, true, weaker> => Heal
 - <neutral, full, false, stronger> => Eat
 - <enemy, low, true, weaker> => Eat
 - <enemy, low, true, same> => Attack
 - <neutral, low, true, weaker> => Heal
 - <enemy, medium, true, stronger> => Run
 - <friendly, full, true, same> => Ignore
 - <neutral, full, true, stronger> => Ignore
 - <enemy, full, true, same> => Run
 - <enemy, medium, true, weaker> => Attack
 - <enemy, low, true, weaker> => Ignore
 - <neutral, full, false, stronger> => Ignore
 - <friendly, medium, true, stronger> => Heal
- Estimate the most likely classification of:
 - <enemy, full, true, stronger>

Example

- Need to calculate:
 - $P(\text{Attack} | \langle \text{enemy}, \text{full}, \text{true}, \text{stronger} \rangle)$
 $= P(\text{Attack}) P(\text{enemy} | \text{Attack}) P(\text{full} | \text{Attack}) P(\text{true} | \text{Attack}) P(\text{stronger} | \text{Attack})$
 - $P(\text{Ignore} | \langle \text{enemy}, \text{full}, \text{true}, \text{stronger} \rangle)$
 $= P(\text{Ignore}) P(\text{enemy} | \text{Ignore}) P(\text{full} | \text{Ignore}) P(\text{true} | \text{Ignore}) P(\text{stronger} | \text{Ignore})$
 - $P(\text{Heal} | \langle \text{enemy}, \text{full}, \text{true}, \text{stronger} \rangle)$
 $= P(\text{Heal}) P(\text{enemy} | \text{Heal}) P(\text{full} | \text{Heal}) P(\text{true} | \text{Heal}) P(\text{stronger} | \text{Heal})$
 - $P(\text{Eat} | \langle \text{enemy}, \text{full}, \text{true}, \text{stronger} \rangle)$
 $= P(\text{Eat}) P(\text{enemy} | \text{Eat}) P(\text{full} | \text{Eat}) P(\text{true} | \text{Eat}) P(\text{stronger} | \text{Eat})$
 - $P(\text{Run} | \langle \text{enemy}, \text{full}, \text{true}, \text{stronger} \rangle)$
 $= P(\text{Run}) P(\text{enemy} | \text{Run}) P(\text{full} | \text{Run}) P(\text{true} | \text{Run}) P(\text{stronger} | \text{Run})$

Example (cont'd)

- $P(\text{Ignore} | \langle \text{enemy, full, true, stronger} \rangle)$
= $P(\text{Ignore}) P(\text{enemy} | \text{Ignore}) P(\text{full} | \text{Ignore}) P(\text{true} | \text{Ignore}) P(\text{stronger} | \text{Ignore})$

$P(\text{Ignore}) = 4 \text{ of } 13 \text{ examples} = 4/13 = 31\%$

$P(\text{enemy} | \text{Ignore}) = 1 \text{ of } 4 \text{ examples} = 1/4 = 25\%$

$P(\text{full} | \text{Ignore}) = 3 \text{ of } 4 \text{ examples} = 3/4 = 75\%$

$P(\text{true} | \text{Ignore}) = 3 \text{ of } 4 \text{ examples} = 3/4 = 75\%$

$P(\text{stronger} | \text{Ignore}) = 2 \text{ of } 4 \text{ examples} = 2/4 = 50\%$

$P(\text{Ignore} | \langle \text{enemy, full, true, stronger} \rangle) = 2.2\%$

Example (cont'd)

- $P(\text{Run} | \langle \text{enemy, full, true, stronger} \rangle)$
= $P(\text{Run}) P(\text{enemy} | \text{Run}) P(\text{full} | \text{Run}) P(\text{true} | \text{Run}) P(\text{stronger} | \text{Run})$

$P(\text{Run}) = 2 \text{ of } 13 \text{ examples} = 2/13 = 15\%$

$P(\text{enemy} | \text{Run}) = 2 \text{ of } 2 \text{ examples} = 100\%$

$P(\text{full} | \text{Run}) = 1 \text{ of } 2 \text{ examples} = 50\%$

$P(\text{true} | \text{Run}) = 2 \text{ of } 2 \text{ examples} = 100\%$

$P(\text{stronger} | \text{Run}) = 1 \text{ of } 2 \text{ examples} = 50\%$

$P(\text{Run} | \langle \text{enemy, full, true, stronger} \rangle) = 3.8\%$

Result

- $P(\text{Ignore} | \langle \text{enemy, full, true, stronger} \rangle) = 2.2\%$
- $P(\text{Run} | \langle \text{enemy, full, true, stronger} \rangle) = 3.8\%$
- $P(\text{Eat} | \langle \text{enemy, full, true, stronger} \rangle) = 0.1\%$
- $P(\text{Heal} | \langle \text{enemy, full, true, stronger} \rangle) = 0\%$
- $P(\text{Attack} | \langle \text{enemy, full, true, stronger} \rangle) = 0\%$

- So Naïve Bayes Classification says Run is most probably
 - 63% of Run being correct
 - 36% of Ignore being correct
 - 1% of Eat being correct

Estimating Probabilities

- Need lots of examples for accurate estimates
- With only 13 examples:
 - No example of:
 - Health=full for Attack category
 - RelativeHealth=Stronger for Attack
 - Allegiance=enemy for Heal
 - Health=full for Heal
 - Only two examples of Run
 - $P(f_1|\text{Run})$ can only be 0%, 50%, or 100%
 - What if the true probability is 16.2%?
- Need to add a factor to probability estimates that:
 - Prevents missing examples from dominating
 - Estimates what might happen with more examples

m-estimate

- Solution: m-estimate
 - Establish a prior estimate p
 - Expert input
 - Assume uniform distribution
 - Estimate the probability as:

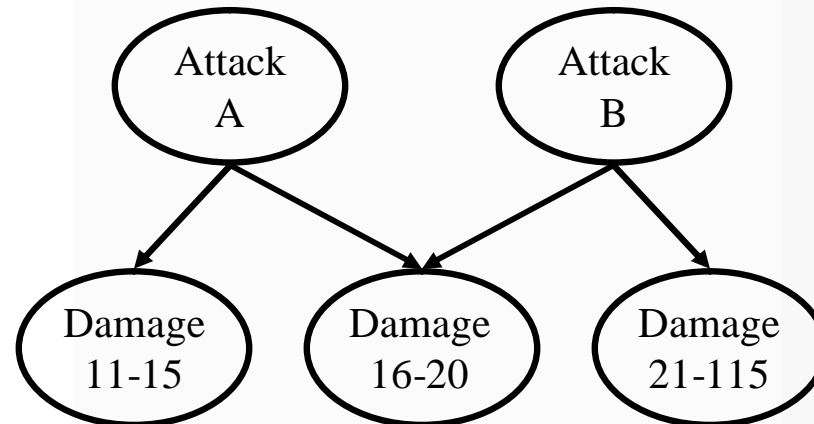
$$\frac{n_c + mp}{n + m}$$

- m is the equivalent sample size
- Augment n observed samples with m virtual samples
- If there are no examples ($n_c = 0$) estimate is still $> 0\%$
- If $p(\text{run}) = 20\%$ and $m = 10$ then $P(\text{full}|\text{Run})$:
 - Goes from 50% (1 of 2 examples)
 - to 25%

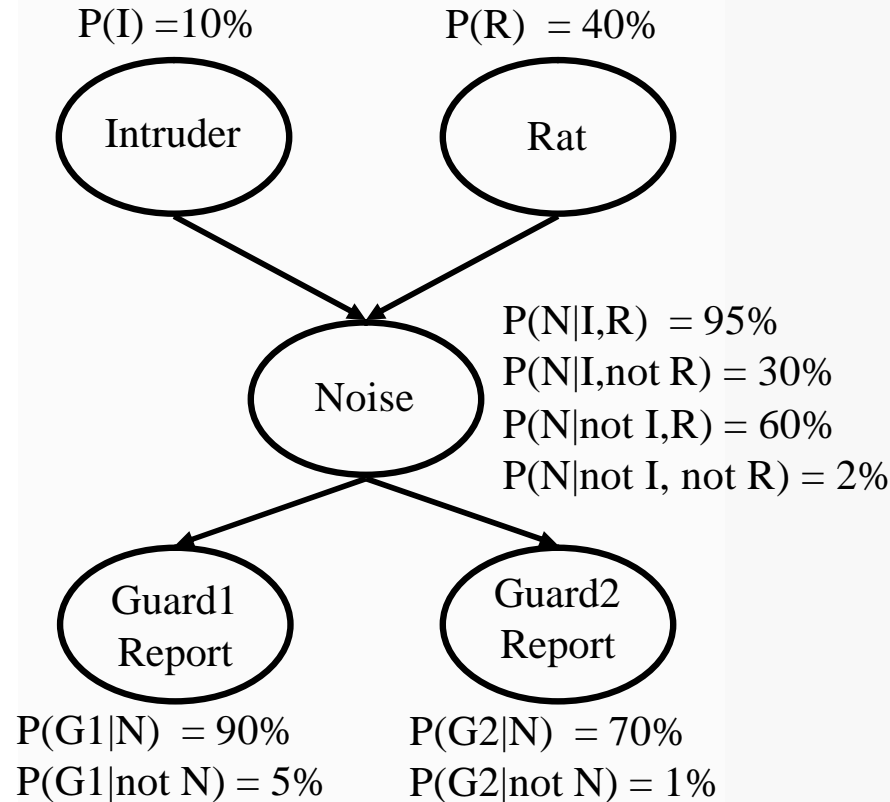
$$\frac{1 + 10(.2)}{2 + 10} = \frac{3}{12} = 0.25$$

Bayesian Networks

- Graph structure encoding causality between variables
 - Directed, acyclic graph
 - A → B indicates that A directly influences B
 - Positive or negative influence



Another Bayesian Network



- Inference on Bayesian Networks can determine probability of unknown nodes (Intruder) given some known values
 - If Guard2 reports but Guard 1 doesn't, what's the probability of Intruder?

Learning Bayesian Networks

- Learning the topology of Bayesian networks
 - Search the space of network topologies
 - Adding arcs, deleting arcs, reversing arcs
 - Are independent nodes in the network independent in the data?
 - Does the network explain the data?
 - Need to weight towards fewer arcs
- Learning the probabilities of Bayesian networks
 - Experts are good at constructing networks
 - Experts aren't as good at filling in probabilities
 - Expectation Management (EM) algorithm
 - Gibbs Sampling

Bayesian Learning Evaluation

- Pros
 - Takes advantage of prior knowledge
 - Probabilistic predictions (prediction confidence)
 - Handles noise well
 - Incremental learning
- Cons
 - Less effective with low number of examples
 - Can be computationally expensive
- Challenges
 - Identifying the right features
 - Getting a large number of good examples

References

- Mitchell: Machine Learning, McGraw Hill, 1997.
- Russell and Norvig: Artificial Intelligence: A Modern Approach, Prentice Hall, 1995.
- AI Game Programming Wisdom.

Reinforcement Learning

John Laird

Thanks for online reference material to: Satinder Singh, Yijue Hou & Patrick Doyle