LOGICAL AND BAYESIAN LEARNING

Outline

- A Logical Formulation of Learning
- Bayesian Learning

Based on the textbook by Stuart Russell & Peter Norvig:

Artificial Intelligene, A Modern Approach (2nd Edition)

Sections 19.1 and 20.1

1. A LOGICAL FORMULATION OF LEARNING

- Inductive learning was previously defined as a process of searching for a hypothesis that agrees with the observed examples.
- For now we concentrate on the case where hypotheses, examples, classifications are **represented** in terms of *logical sentences*.
- This form of learning is more general and complex compared to learning decision trees or lists.
- This approach allows for *incremental construction* of hypotheses, one sentence at a time.
- The full power of logical inference can be utilized in learning.

Examples and Hypotheses

- In the logical representation, attributes become unary predicates.
- The i^{th} example is generically denoted by X_i.

  Example. The first example in the restaurant domain is described by the following sentence:

  $$Alternate(X_1) \wedge \neg Bar(X_1) \wedge \neg Fri/Sat(X_1) \wedge Hungry(X_1) \wedge \ldots$$

- The classification of the object is given by $WillWait(X_1)$.
- The generic notations $Q(X_i)$ and $\neg Q(X_i)$ are used for *positive* and *negative* examples, respectively.
- The complete training set corresponds to the conjunction of the respective description and classifications sentences.

Candidate Definitions

- The aim is to find an equivalent logical expression for the goal predicate Q that can be used to classify examples correctly.
- Each hypothesis H_i proposes a **candidate definition** $C_i(x)$ for the goal predicate Q, i.e. H_i takes the form $\forall x(Q(x) \leftrightarrow C_i(x))$.
- The **extension** of a hypothesis $H_i = \forall x(Q(x) \leftrightarrow C_i(x))$ is the set of examples X for which $Q(X)$ evaluates to true.

Example. For the decision tree learned in the restaurant example:

$$\begin{aligned} H_1 = \forall r(WillWait(r) \leftrightarrow & Patrons(r,Some) \vee \\ & (Patrons(r,Full) \wedge \neg Hungry(r) \wedge Type(r,French)) \vee \\ & (Patrons(r,Full) \wedge \neg Hungry(r) \wedge Type(r,Thai) \wedge Fri/Sat(r)) \vee \\ & (Patrons(r,Full) \wedge \neg Hungry(r) \wedge Type(r,Burger))\ ) \end{aligned}$$

Hypothesis Space

- Logically equivalent hypotheses have equal extensions.
- Two hypotheses with different extensions are logically inconsistent with each other, as they differ on at least one example X_i.

  Example. The conjunction of $H_2 = \forall r(WillWait(r) \leftrightarrow Hungry(r))$ and $H_3 = \forall r(WillWait(r) \leftrightarrow \neg Hungry(r))$ imply a contradiction.
- The hypothesis space $\{H_1, H_2, \ldots, H_n\}$ is denoted by $\mathbf{H}$.
- It is usually believed that one of the hypotheses in $\mathbf{H}$ is correct, i.e. the disjunction $H_1 \vee H_2 \vee \ldots \vee H_n$ evaluates to true.

Example. In decision tree learning, the hypothesis space consists of all decision trees that can be defined in terms of the attributes provided.

Classifying Examples with Hypotheses

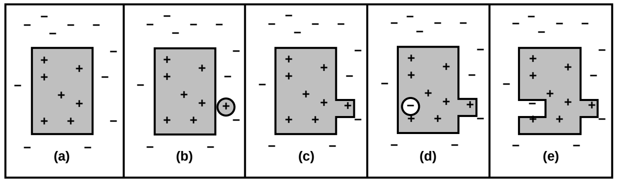
- Given a hypothesis $H_i = \forall x(Q(x) \leftrightarrow C_i(x))$, an example X is **positive/negative** if $Q(X)/\neg Q(X)$ evaluates to true.
- A **false** positive/negative example X for a hypothesis $H_i = \forall x(Q(x) \leftrightarrow C_i(x))$ gets an incorrect classification by H_i.
- Inductive learning can be seen as a process of gradually eliminating hypotheses that are inconsistent with examples.

Example. For H_1 in the restaurant domain, the first example X_1 is a positive one, as $WillWait(X_1)$ evaluates to true.

On the other hand, X_1 is a false negative example for $H_3 = \forall r(WillWait(r) \leftrightarrow \neg Hungry(r))$, as $Hungry(X_1)$ holds.

Current-Best-Hypothesis Search

- The idea is to maintain a single hypothesis H, and to adjust it as new examples arrive in order to maintain consistency.
- The current hypothesis H is ilustrated in the figure (a) below.
- A false negative example (b) can be removed by a **generalization** (c) that extends the extension of the current hypothesis H_i.
- A false positive example (d) can be removed by a **specialization** (e) that narrows the extension of the current hypothesis H_i.



Skeletal Algorithm

Current-best-hypothesis search is captured by the following algorithm:

```
function CURRENT-BEST-LEARNING(examples) returns a hypothesis

  H ← any hypothesis consistent with the first example in examples
  for each remaining example in examples do
     if e is false positive for H then
        H ← choose a specialization of H consistent with examples
     else if e is false negative for H then
        H ← choose a generalization of H consistent with examples
     if no consistent specialization/generalization can be found then fail
  end
  return H
```

- Generalizations and specializations imply *logical relationships*:

  E.g., if $H_1 = \forall x(Q(x) \leftrightarrow C_1(x))$ is a generalization of $H_2 = \forall x(Q(x) \leftrightarrow C_2(x))$, then $\forall x(C_2(x) \rightarrow C_1(x))$ holds.
- Note that H_2 is a specialization of H_1 in the setting above.

Examples

Example. Recall the training set used in the restaurant domain.

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

Example. A way to generalize is to **drop conditions** from hypotheses. For instance, $\forall x(WillWait(x) \leftrightarrow Patrons(x, Some))$ generalizes the hypothesis $\forall x(WillWait(x) \leftrightarrow Alternate(x) \wedge Patrons(x, Some))$.

Example. Hypotheses are formed in the restaurant example as follows:

H_1: $\forall x(WillWait(x) \leftrightarrow Alternate(x))$

H_2: $\forall x(WillWait(x) \leftrightarrow Alternate(x) \wedge Patrons(x, Some))$

H_3: $\forall x(WillWait(x) \leftrightarrow Patrons(x, Some))$

H_4: $\forall x(WillWait(x) \leftrightarrow Patrons(x, Some) \vee (Patrons(x, Full) \wedge Fri/Sat(x))\ )$

There are also other hypotheses conforming to the first four examples:

H_4': $\forall x(WillWait(x) \leftrightarrow \neg WaitEstimate(x, 30\text{-}60))$

H_4'': $\forall x(WillWait(x) \leftrightarrow Patrons(x, Some) \vee$
$(Patrons(x, Full) \wedge WaitEstimate(x, 10\text{-}30))\ )$

Discussion

- The CURRENT-BEST-LEARNING algorithm is *non-deterministic*: there may be several possible specializations or generalizations that can be applied at any point.
- The choices made might not lead to the simplest hypothesis.
- If a dead-end (unrecoverable inconsistency) is encountered, the algorithm must backtrack to a previous choice point.
- Checking the consistency of all the previous examples over again for each choice is very expensive.

Least-Commitment Search

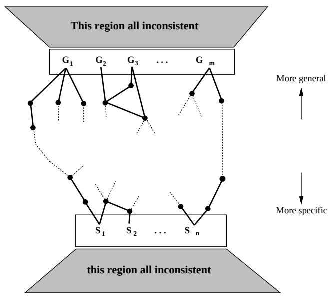
- The original hypothesis space can be seen as a huge disjunction
$$H_1 \vee H_2 \vee \ldots \vee H_n.$$
- Hypotheses which are consistent with all examples encountered so far form a set of hypotheses called the **version space** V.
- Version space is shrunk by the **candidate elimination** algorithm:

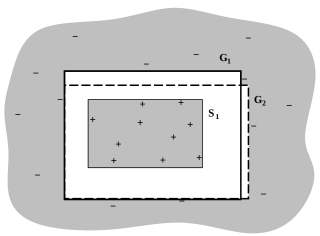
```
function VERSION-SPACE-LEARNING(examples) returns a version space
  local variables: V, the version space: the set of all hypotheses

  V ← the set of all hypotheses
  for each example e in examples do
     if V is not empty then V ← VERSION-SPACE-UPDATE(V, e)
  end
  return V
---------------------------------------------------------------
function VERSION-SPACE-UPDATE(V, e) returns an updated version space
  V ← {h ∈ V : h is consistent with e}
```

Boundary Sets

- The algorithm finds a subset of the version space V that is consistent with all examples in an *incremental* way.
- Candidate elimination is an example of a **least-commitment** algorithm, as no arbitrary choices are made among hypotheses.
- Since the hypothesis space V is possibly enormous, it cannot be represented directly as a set of hypotheses or a disjunction.
- The problem can be alleviated by **boundary sets** $\{S_1, \ldots, S_n\}$ (**S-set**) and $\{G_1, \ldots, G_m\}$ (**G-set**) and a partial ordering among hypotheses induced by specialization/generalization.
- Any hypothesis H between a most specific boundary S_i and a most general boundary G_j is consistent with the examples seen.

- Boundary sets for the version space are illustrated below:





- Initially, the S-set contains a single hypothesis $\forall x(Q(x) \leftrightarrow False)$ while the G-set contains $\forall x(Q(x) \leftrightarrow True)$ only.
- The remaining problem is how to update S-sets and G-sets for a new example (the job of the VERSION-SPACE-UPDATE function).

Updating Version Space

- Upon a false negative/positive example, a most specific boundary S is replaced by all its immediate generalizations / deleted.
- Upon a false positive/negative example, a most general boundary G is replaced by all its immediate specializations / deleted.

These operations on S-sets and G-sets are continued until:

1. There is exactly one hypothesis left in the version space.
2. The version space *collapses* (i.e., the S-set or G-set becomes empty): there are no consistent hypotheses for the training set.
3. We run out of examples with several hypotheses remaining in the version space: a solution is to take the majority vote.

Discussion

- If the domain contains noise or insufficient attributes for exact classification, the version space will always collapse.
- If unlimited disjunction is allowed in the hypothesis space, the S-set will always contain a single most-specific hypothesis (disjunction of positive examples seen to date).
- Analogously for the G-set and negative examples.
- A solution is to allow only limited forms of disjunction.
- For certain hypothesis spaces, the number of elements in the S-set and G-set may grow exponentially in the number of attributes.

2. BAYESIAN LEARNING

- The **data**, i.e. instantiations of some or all random variables describing the domain, serve as evidence.
- **Hypotheses** are probabilistic theories of how the domain works.
- The aim is to make a prediction concerning an unknown quantity X given some data and hypotheses.
- In **Bayesian learning**, the probability of each hypothesis is calculated, given the data, and predictions are made on that basis.
- Predictions are made by using *all* the hypotheses, weighted by their probabilities, rather than by using a single "best" hypothesis.

Example. Our favorite *Surprise* candy comes in two flavors, cherry and lime, but they are wrapped in an indistinguishable way.

The candy is sold in large (indistinguishable) bags containing various mixtures of the two flavors:

1. 100% cherry
2. 75% cherry and 25% lime
3. 50% cherry and 50% lime
4. 25% cherry and 75% lime
5. 100% lime

Given a new bag of candy, the random variable H (for *hypothesis*) denotes the type of the bag, with possible values h_1 through h_5.

☞ The agent needs to infer a probabilistic model of the world.

Bayesian learning

- Let $\mathbf{D}$ represent all the data with observed value $\mathbf{d}$.
- The probability of each hypothesis h_i is obtained by Bayes' rule:

$$P(h_i \mid \mathbf{d}) = \alpha P(\mathbf{d} \mid h_i) P(h_i).$$

- Assuming that each h_i specifies a complete distribution for an unknown quantity X, Bayesian learning is characterized by

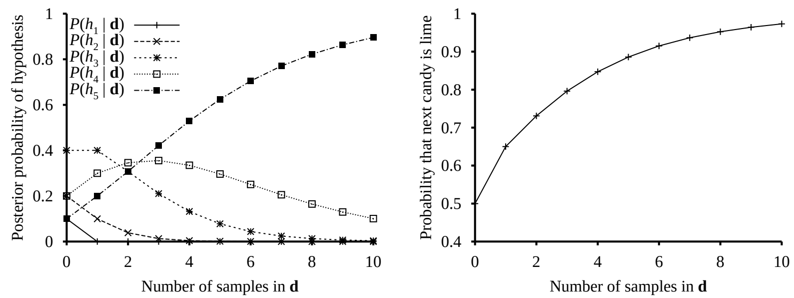
$$\mathbf{P}(X \mid \mathbf{d}) = \sum_i \mathbf{P}(X \mid \mathbf{d}, h_i)\mathbf{P}(h_i \mid \mathbf{d}) = \sum_i \mathbf{P}(X \mid h_i) P(h_i \mid \mathbf{d}).$$

- The key quantities are the **hypothesis** prior $P(h_i)$ and the **likelihood** of the data under each hypothesis $P(\mathbf{d} \mid h_i)$.
- If the observations are independently and identically distributed (**i.i.d.** for short), then $P(\mathbf{d} \mid h_i) = \prod_j P(d_j \mid h_i)$.

Example. For the candy example, the prior distribution over $h_1, \ldots, h_5$ is given by $\langle 0.1, 0.2, 0.4, 0.2, 0.1 \rangle$, as advertised by the manufacturer.

If the bag is really an all-lime bag (h_5) and the first 10 candies are consequently all lime, then $P(\mathbf{d} \mid h_3) = 0.5^{10}$.

The posterior probabilities of the five hypotheses change as the sequence of 10 lime candies is observed:



MAP and ML hypotheses

- The true hypothesis eventually dominates Bayesian prediction.
- Unfortunately, the hypothesis space is usually very large or infinite which makes the Bayesian approach intractable.
- A common approximation is to use **maximum a posteriori** (MAP) **hypothesis** h_{MAP} – a hypothesis h_i that maximizes $P(h_i \mid \mathbf{d})$:

$$\mathbf{P}(X \mid \mathbf{d}) \approx \mathbf{P}(X \mid h_{\mathrm{MAP}}).$$

- To determine h_{MAP}, it is sufficient to maximize $P(\mathbf{d} \mid h_i)P(h_i)$.
- In some cases, the prior probabilities $P(h_i)$ can be assumed to be **uniformly** distributed.
- Then maximizing $P(\mathbf{d} \mid h_i)$ produces a **maximum-likelihood** (ML) **hypothesis** h_{ML} – a special case of h_{MAP}.

Bayesian Network Learning Problems

The learning problem for Bayesian networks comes in several varieties:

1. **Known structure, fully observable:** only CPTs are learned and the statistics of the set of examples can be used.
2. **Unknown structure, fully observable:** this involves heuristic search through the space of structures – guided by the ability of modeling data correctly (MAP or ML probability value).
3. **Known structure, hidden variables:** analogy to neural networks.
4. **Unknown structure, hidden variables:** no good/general algorithms are known for learning in this setting.