Foundations of AI

15. Statistical Machine Learning

Bayesian Learning and Why Learning Works

Wolfram Burgard and Bernhard Nebel
Contents

- Statistical learning
- Why learning works
Statistical Learning Methods

- In MDPs probability and utility theory allow agents to deal with uncertainty.
- To apply these techniques, however, the agents must first learn their probabilistic theories of the world from experience.
- We will discuss statistical learning methods as robust ways to learn probabilistic models.
An Example for Statistical Learning

- The key concepts are data (evidence) and hypotheses.
- A candy manufacturer sells five kinds of bags that are indistinguishable from the outside:
  - $h_1$: 100% cherry
  - $h_2$: 75% cherry and 25% lime
  - $h_3$: 50% cherry and 50% lime
  - $h_4$: 25% cherry and 75% lime
  - $h_5$: 100% lime
- Given a sequence $d_1, \ldots, d_N$ of candies observed, what is the most likely flavor of the next piece of candy?
Bayesian Learning

- Calculates the probability of each hypothesis, given the data.
- It then makes predictions using all hypotheses weighted by their probabilities (instead of a single best hypothesis).
- Learning is reduced to probabilistic inference.
Application of Bayes Rule

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

$$P(h_i \mid d) = \alpha P(d \mid h_i) P(h_i)$$

- The manufacturer tells us that the prior distribution over $h_1, \ldots, h_5$ is given by \langle .1, .2, .4, .2, .1 \rangle
- We compute the likelihood of the data under the assumption that the observations are independently and identically distributed (i.i.d.):

$$P(d \mid h_i) = \prod_{j} P(d_j \mid h_i)$$
How to Make Predictions?

- Suppose we want to make predictions about an unknown quantity \( X \) given the data \( \mathbf{d} \).

\[
P(X \mid \mathbf{d}) = \sum_i P(X \mid h_i, \mathbf{d}) P(h_i \mid \mathbf{d})
\]

\[
= \sum_i P(X \mid h_i) P(h_i \mid \mathbf{d})
\]

- Predictions are weighted averages over the predictions of the individual hypotheses.

- The key quantities are the hypothesis prior \( P(h_i) \) and the likelihood \( P(d \mid h_i) \) of the data under each hypothesis.
Example

- Suppose the bag is an all-lime bag ($h_5$)
- The first 10 candies are all lime.
- Then $P(d|h_3)$ is $0.5^{10}$ because half the candies in an $h_3$ bag are lime.
- Evolution of the five hypotheses given 10 lime candies were observed (the values start at the prior!).
Observations

- The true hypothesis often dominates the Bayesian prediction.

- For any fixed prior that does not rule out the true hypothesis, the posterior of any false hypothesis will eventually vanish.

- The Bayesian prediction is optimal and, given the hypothesis prior, any other prediction will be correct less often.

- It comes at a price that the hypothesis space can be very large or infinite.
Maximum a Posteriori (MAP)

- A common approximation is to make predictions based on a single most probable hypothesis.
- The maximum a posteriori (MAP) hypothesis is the one that maximizes $P(h_i | d)$.
  \[ P(X | d) \approx P(X | h_{MAP}) \]
- In the candy example, $h_{MAP} = h_5$ after three lime candies in a row.
- The MAP learner predicts that the fourth candy is lime with probability 1.0, whereas the Bayesian prediction is still 0.8.
- As more data arrive, MAP and Bayesian predictions become closer.
- Finding MAP hypotheses is often much easier than Bayesian learning.
Maximum-Likelihood Hypothesis (ML)

- A final simplification is to assume a uniform prior over the hypothesis space.
- In that case MAP-learning reduces to choosing the hypothesis that maximizes $P(d|h_i)$.
- This hypothesis is called the maximum-likelihood hypothesis (ML).
- ML-learning is a good approximation to MAP learning and Bayesian learning when there is a uniform prior and when the data set is large.
Why Learning Works

How can we decide that $h$ is close to $f$ when $f$ is unknown?

→ Probably approximately correct

Stationarity as the basic assumption of PAC-Learning: training and test sets are selected from the same population of examples with the same probability distribution.

Key question: how many examples do we need?

$X$  Set of examples
$D$  Distribution from which the examples are drawn
$H$  Hypothesis space ($f \in H$)
$m$  Number of examples in the training set

$error(h) = P(h(x) \neq f(x) \mid x \text{ drawn from } D) \leq \epsilon$
**PAC-Learning**

A hypothesis $h$ is **approximately correct** if $\text{error}(h) \leq \epsilon$.

To show: After the training period with $m$ examples, with high probability, all consistent hypotheses are approximately correct.

How high is the probability that a wrong hypothesis $h_b \in H_{\text{bad}}$ is consistent with the first $m$ examples?
Sample Complexity

Assumption: \( \text{error}(h) > \epsilon \implies \)

\[ P(h_b \text{ is consistent with 1 example}) \leq (1 - \epsilon) \]

\[ P(h_b \text{ is consistent with } N \text{ examples}) \leq (1 - \epsilon)^N \]

\[ P(H_{bad} \text{ contains a consistent } h) \leq |H_{bad}|(1 - \epsilon)^N \]

Since \( |H_{bad}| \leq |H| \)

\[ P(H_{bad} \text{ contains a consistent } h) \leq |H|(1 - \epsilon)^N \]

We want to limit this probability by some small number \( \delta \):

\[ |H|(1 - \epsilon)^N < \delta \]

Since \( (1 - \epsilon) \leq e^{-\epsilon} \), we derive

\[ N \geq \frac{1}{\epsilon} \left( \log \left( \frac{1}{\delta} \right) + \log |H| \right) \]

Sample Complexity: Number of required examples, as a function of \( \epsilon \) and \( \delta \).
Sample Complexity (2)

Example: Boolean functions

The number of Boolean functions over $n$ attributes is $|H| = 2^{2^n}$.

The sample complexity therefore grows as $2^n$.

Since the number of possible examples is also $2^n$, any learning algorithm for the space of all Boolean functions will do no better than a lookup table, if it merely returns a hypothesis that is consistent with all known examples.
Learning from Decision Lists

In comparison to decision trees:

- The overall structure is simpler
- The individual tests are more complex

This represents the hypothesis

\[ H_4 : \forall x \text{ WillWait}(x) \Leftrightarrow \text{Patrons}(x, \text{some}) \lor [\text{Patrons}(x, \text{full}) \land \text{Fri/Sat}(x)] \]

If we allow tests of arbitrary size, then any Boolean function can be represented.

**k-DL**: Language with tests of length \( \leq k \).
Learnability of k-DL

\[
\text{function} \quad \text{DECISION-LIST-LEARNING}(\text{examples}) \quad \text{returns} \quad \text{a decision list, No or failure}
\]

if \text{examples} \text{ is empty then return the value No}

\[ t \leftarrow \text{a test that matches a nonempty subset \text{examples}, of \text{examples} such that the members of \text{examples}, are all positive or all negative} \]

if there is no such \( t \) then return failure

if the examples in \text{examples}, are positive then \( o \leftarrow \text{Yes} \)

else \( o \leftarrow \text{No} \)

return a decision list with initial test \( t \) and outcome \( o \) and remaining elements given by \text{DECISION-LIST-LEARNING}(\text{examples} - \text{examples},)\)

\[ |k-\text{DL}(n)| \leq 3|\text{Conj}(n, k)| \leq |\text{Conj}(n, k)!! | \quad \text{(Yes, No, no-Test, all permutations)} \]

\[ |\text{Conj}(n, k)| = \sum_{i=0}^{k} \left( \binom{2n}{i} \right) = O(n^k) \]

(Combination without repeating pos/neg attributes)

\[ |k-\text{DL}(n)| = 2^{O(n^{k\log(n^k)})} \quad \text{(with Euler’s summation formula)} \]

\[ m \geq \frac{1}{\epsilon} (ln(\frac{1}{\delta}) + O(n^{k\log(n^k)})) \]
Summary
(Statistical Learning Methods)

- Bayesian learning techniques formulate learning as a form of probabilistic inference.
- Maximum a posteriori (MAP) learning selects the most likely hypothesis given the data.
- Maximum likelihood learning selects the hypothesis that maximizes the likelihood of the data.
Summary
(Statistical Learning Theory)

Inductive learning as learning the representation of a function from example input/output pairs.

- **Decision trees** learn deterministic Boolean functions.
- **PAC learning** deals with the complexity of learning.
- **Decision lists** as functions that are easy to learn.