Chapter 3. Learning Algorithms

I. Initialize initial hypothesis \( h \rightarrow \) false 

2. For each positive example \( e \in S \) do:

3. Output \( h \) as the hypothesis that best approximates the target.

\( h \) from \( h \)' otherwise remove \( h \).

If the \( h \) boolean attribute \( h = 0 \) in the example, remove

A Learning Algorithm

<table>
<thead>
<tr>
<th>Example</th>
<th>1</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>1</th>
<th>1</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Training Examples
An analysis of the learning algorithm

\[
\left( \frac{9}{1} u \eta + (\frac{9}{1}) \eta u \right)^2 = u
\]

Examples where

Find high-accuracy approximations in polynomial time, given in ...

Surprisingly, we can then show that this algorithm can reliably ...

from a hard (unknown) distribution.

Additional assumptions are made (e.g., examples are drawn

This algorithm will never converge quickly unless some ...

```
How many examples are needed for the algorithm to learn the
```

Let \( u = 100 \). Then \( P \approx 1 \).

Size of hypothesis space \( P = 3^n \) (exponential).

Analysis of this learning algorithm

- Hypothesis consistent with the data.
- Algorithmic bias: keeps track of only the most specific
- Confounding expressions.
- Representational bias: concepts are describable by purely

Properties of this learning algorithm
Design of a Learning System

Mistake-Bounded Model of Concept Learning

\[ \text{Answer: } n + 1 \quad \text{where } n \text{ is the number of attributes} \]

*What algorithm makes, before converging, to the right hypotheses?*

*Problem: How many mistakes will our concept learner determine?*

- Learner is evaluated in terms of the number of mistakes it makes before converging to the right hypotheses.
- Learner is evaluated in terms of the number of mistakes before giving the right answer.
- Learner is evaluated in terms of the number of mistakes before predicting the label (positive or negative) before example is seen.
- Learner is evaluated in terms of the number of mistakes before receiving a training example.
Learning Method

\[ (t) x(t) y(t) \prod_{u} = (t)(m) \nabla \]

Generalized delta rule:

\[ \Delta_{\theta} = \nabla \]

Let \( f \) be the prediction on day \( t \) and \( z \) be the final outcome on

Learning experience:

Database of measurements and final

\[ \text{prob} \text{(snow)} \]

Task: Predict weather in East Lansing next Saturday

Example: Weather prediction

Machine Learning
Sequential Prediction/Decision Problems

- Machine learning/scheduling
- Robot navigation
- Game playing
- Stock market
- Weather prediction

Weather prediction: Supervised learning

Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
Temporal Difference Learning

**Key Ideas**

- Temporal difference learning (TD)
- How can an agent learn online experience?
- Problem: Can we learn useful value functions even if we don’t know the full model?
- Error $\epsilon$: Saturdays’ outcome – Monday’s prediction

---

**Weather Prediction**

- Successful predictions
- Reexpress error as sum of differences between temporally

---

**Temporal Difference Learning**

\[ d_m^a \Delta_{t-1} (t^a - t^{a+1}) = (t^a)m \nabla : \nabla (y) \]
Reinforcement Learning

- Model-based (real-time dynamic programming)
  - Model-free (TD(0) or Q-learning)

Key ideas:

- Learn the optimal value function $V^*$

Model:

- What happens if I do this action?

Policy:

- What do I do in this state?

Value function:

- How good is this state (assuming I follow a fixed policy)?

Reward:

- Scalar feedback

Weather Prediction using TD Learning
Instances

Assgin new vector the class label of the majority of the closest instances
Given a new feature vector, determine closest instances using some distance metric
Store all instances

Nearest Neighbor

Repeat until class imputy is minimal
Partition all instances into \( \geq 80\% \) and \(< 80\% \)
Choose some value to split on (e.g. 80%) (Choose some feature to split on (e.g. humidity))

Decision Trees

Other Function Approximators
Issues in Choosing Approximators