Generalization in Machine Learning:
Concepts, Algorithms, and Examples

Mehran Sahami

Robotics Laboratory
Computer Science Department
Stanford University
Stanford, CA 94305

sahami@cs.Stanford.EDU

Overview

- Machine Learning
- Conceptual Definitions
- Generalization Examined
- Algorithms (with examples)
  - decision trees
  - lattice-based rule systems
  - nearest neighbor methods
  - neural networks

Machine Learning: Supervised Learning

- Given: a set Ξ of N examples <**x**, c>, where $\mathbf{x} \in X^m$ denotes a vector of attributes and $c \in C$ denotes a classification.
- Output: a classifier, $F(\mathbf{x}): X^m \to C$, which maximizes some objective function on Ξ. (i.e. $F(\mathbf{x})$ minimizes error on some portion of Ξ.)

• Example: Learning the function $(x_1 \wedge x_2)$

- Attribute space: $[0, 1]^3$
- Class space: [0, 1]
- exemplars:

<0, 0, 0, 0>	<1, 1, 0, 1>
<0, 1, 0, 0>	<1, 0, 1, 0>
<1, 1, 1, 1>	<0, 1, 1, 0>

Machine Learning: Unsupervised Learning

- Given: a set Ξ of N examples <**x**>, where $\mathbf{x} \in X^m$ denotes a vector of attributes. (Note: no class information is provided)
- Output: a clusterer, $F(\mathbf{x}): X^m \to C$, which maximizes some objective function on Ξ. (i.e. $F(\mathbf{x})$ maximizes within cluster similarity and minimizes between cluster similarity.)

• Example: clustering cars (i.e. sports/family cars)

- Attribute space: [{red, blue}, {2-dr, 4-dr}, price]
- exemplars:

<red, 2-dr, 30K>	<blue, 4-dr, 15K>
<red, 4-dr, 40K>	<blue, 2-dr, 45K>
<red, 4-dr, 2K>	<blue, 4-dr, 7K>

Concepts:

- Bias: Preference on the type of function to fit the data.
- Variance: The amount by which the classification function will vary given perturbations in the data.
- Bias-Variance Trade-off: The weaker the bias in our algorithm the greater the potential for change in the induced classification function.
- Occam's Razor: A generally employed bias to select the "simplest" classification function from a number of equally likely candidates.
- Example: Consider the function $(x_1 \wedge x_2)$
  - Given the data in the previous example we could induce: $(x_1 \wedge x_2) \vee (\neg x_1 \wedge \neg x_2 \wedge x_3)$
  - This is not the simplest function consistent with X.
  - Adding <0, 0, 1, 0> changes the function induced.

Generalization: Why do it?

- To perform well on *unseen* data.
- To deal with *noise* in the data (*overfitting*).
- To produce more *compact representations* that humans can understand.
- Generalization is <u>the</u> goal of learning.
  - Often in learning tasks, the given data set X is split into a training set and test set. The learning algorithm is run on the training set and its performance is measured using the testing set.
  - This disadvantages the learner for the sake of testing generalization ability.

Algorithm: Decision Trees

The ID3 algorithm [Quinlan, 1986]

```
ID3(X)
{
    If X contains only positive examples, return 1
    Else if X contains only negative examples, return 0
    Else {
        choose "best" attribute x_i to be root of the tree
        partition:  X_0i = {x ∈ X | x_i = 0}
                    X_1i = {x ∈ X | x_i = 1}
        return tree with root x_i, left subtree ID3(X_0i) and
            right subtree ID3(X_1i)
    }
}
```


Algorithm: Decision Trees continued

Determine the "best" attribute by selecting the attribute which maximizes the information gain criterion:

$X_{1i} = \{\mathbf{x} \in X \mid x_i = 1\}$ $\qquad X_{0i} = \{\mathbf{x} \in X \mid x_i = 0\}$

$p = |\{\mathbf{x} \in X \mid c = 1\}|$ $\qquad n = |\{\mathbf{x} \in X \mid c = 0\}|$

$p_{ij} = |\{\mathbf{x} \in X_{ji} \mid c = 1\}|$ $\qquad n_{ij} = |\{\mathbf{x} \in X_{ji} \mid c = 0\}|$

Then the information of this attribute is given by:

$$I(p_{ij}, n_{ij}) = -\frac{p_{ij}}{p_{ij}+n_{ij}} \log_2\left(\frac{p_{ij}}{p_{ij}+n_{ij}}\right) - \frac{n_{ij}}{p_{ij}+n_{ij}} \log_2\left(\frac{n_{ij}}{p_{ij}+n_{ij}}\right)$$

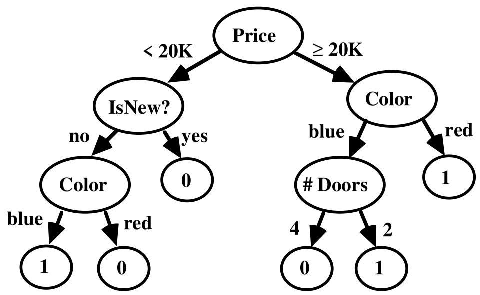
$$\text{Information Gain} = I(p, n) - \sum_{j=0}^{1} \frac{p_{ij}+n_{ij}}{p+n} I(p_{ij}, n_{ij})$$

Algorithm: Decision Trees example

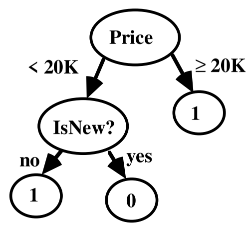
Consider the set of exemplars for “sports cars”:

<red, 2-dr, 30K, new, 1>	<blue, 4-dr, 18K, new, 0>
<red, 4-dr, 40K, new, 1>	<blue, 2-dr, 15K, new, 0>
<red, 4-dr, 20K, used, 1>	<blue, 2-dr, 12K, used, 1>
<red, 4-dr, 60K, new, 1>	<blue, 4-dr, 15K, used, 1>
<blue, 4-dr, 22K, new, 0>	<red, 2-dr, 4K, used, 0>
<blue, 2-dr, 45K, used, 1>	<red, 4-dr, 2K, used, 0>

What we get



What we want

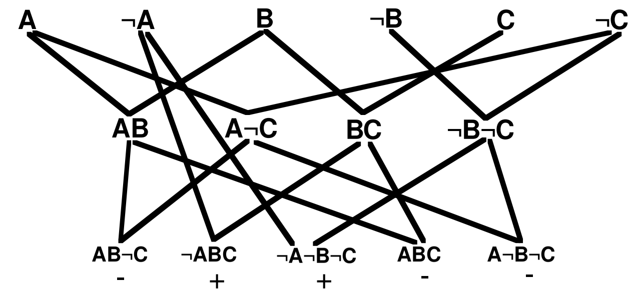


Algorithm: Lattice-based rule system

The Rulearner algorithm [Sahami, 1995]

- Build a lattice using the set of instance vectors.
- Consider the training set:

<¬A, ¬B, ¬C, +> <A, B, ¬C, –> <A, B, C, –>
<¬A, B, C, +> <A, ¬B, C, –>



Algorithm: Lattice-based rule system

- Consider the Karnaugh map of this space:

C \ AB	0 0	0 1	1 1	1 0
0	+	?	–	–
1	?	+	–	?

- We wish to find the rules that cover the greatest number of instances while still maintaining classification “purity”.
- Do a top-down search through the lattice to find nodes which *cover* the greatest number of instances of the same class. *Generalize by allowing for mistakes!*

Algorithm: Lattice-based rule system example

CD \ AB	0 0	0 1	1 1	1 0
0 0	?	?	–	?
0 1	?	+	?	?
1 1	–	+	+	–
1 0	?	–	?	?

Unordered rules
B^D → +
¬B → –
¬D → –

Decision List
B^D → +
→ –

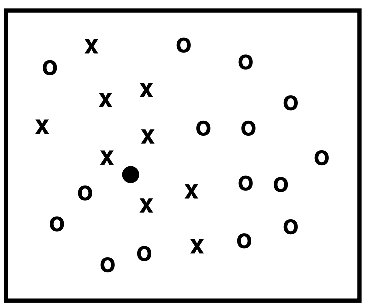
Algorithm: Lattice-based rule system example

- Individual rules can be further generalized by pruning antecedents that imply a classification.
- The whole rule set can also be pruning (i.e. throw away some rules completely) to promote generalization.
- A *default rule* can also be employed which is used to classify all instances that are not covered by some other rule (i.e. predict the majority class).
- Such methods are applicable to any rule-based system.
- Decision trees can be converted to a set of rules and further generalized using these pruning methods.

Algorithm: Nearest Neighbor

- Consider each training instance as a data point in *m*-dimensional space.
- For each new instance that we want to classify, we simply measure its “distance” away for all the points we have previously seen.
- The class predicted for the new instance is the class of the nearest previously seen node.
- Can use a variety of “distance” metrics:
  - Euclidean distance for real-valued attributes
  - Hamming distance for Boolean attributes
- Very easy algorithm to implement.
- Memory intensive.

Algorithm: Nearest Neighbor example



- The new instance (the dot) is classified as X since it’s nearest neighbor is also of class X.
- Generalization is achieved by deciding how many instances to keep in memory to make your classifications based on.
- Want large regions of similar classification.

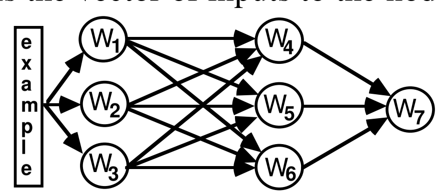
Algorithm: Neural Networks

Back-Propagation Algorithm [Rumelhart *et al*, 1986]

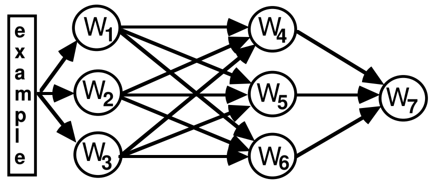
- Construct a network topology of nodes and connections between them.
- Each node contains a vector of weights, W. The output of each node is given by the function f(X, W):

$$f(\vec{X}, \vec{W}) = \frac{1}{1 + e^{-(X \cdot W)}}$$

where X is the vector of inputs to the node.



Algorithm: Neural Networks

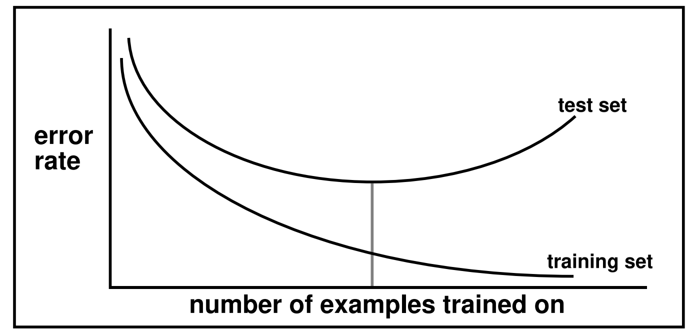


- The learning task is to appropriately update the weights, W*i*, in each node so that the network implements the function from instance attributes to instance class.
- Weight updates can be computed using a gradient-decent method where we attempt to minimize the squared error of the network with respect to the training set.
- The complexity of the network is measured by the number of weights/nodes it contains.

17

Algorithm: Neural Networks

- Large enough neural networks can learn functions of arbitrary complexity since they have so many degrees of freedom (i.e. weights).
- Training for too long can cause the network to overfit the training data and thus do poorly on unseen data.



18

Algorithm: Neural Networks

- To prevent overfitting and thus promote generalization, a *validation* set is used during training.
- We test the partially trained network at various points during learning on the validation set.
- If performance on the validation set begins to deteriorate, we have usually begun to overfit the training data and should stop training.
- Generalization is also achieve by limiting the number of weights/nodes in the network. *This is tricky!*
- Methods have also been suggested to prune weights after training as a means of achieving generalization.

19

Conclusions

- Generalization is the primary goal of machine learning.
- Different algorithms lend themselves to different biases.
- Generalization is achieved in a variety of ways among a myriad of algorithms.
- This is also an active area of research in the statistics community. (Regression is just generalization)
- There are no magic cures and no fool-proof answers.
- Over-generalization is also often a problem as it can cause you to miss the underlying patterns in your data.

“Don’t throw out the baby with the bath water... but make sure you throw out as much bath water as you can.”

20