ADVANCES IN MACHINE LEARNING APPLICATIONS IN SOFTWARE ENGINEERING

DU ZHANG & JEFFREY J.P. TSAI

Advances in Machine Learning Applications in Software Engineering

Du Zhang
California State University, USA

Jeffrey J.P. Tsai
University of Illinois at Chicago, USA

IDEA GROUP PUBLISHING
Hershey • London • Melbourne • Singapore

Acquisitions Editor: Kristin Klinger
Development Editor: Kristin Roth
Senior Managing Editor: Jennifer Neidig
Managing Editor: Sara Reed
Assistant Managing Editor: Sharon Berger
Copy Editor: Amanda Appicello
Typesetter: Amanda Appicello
Cover Design: Lisa Tosheff
Printed at: Integrated Book Technology

Published in the United States of America by
Idea Group Publishing (an imprint of Idea Group Inc.)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@idea-group.com
Web site: http://www.idea-group.com

and in the United Kingdom by
Idea Group Publishing (an imprint of Idea Group Inc.)
3 Henrietta Street
Covent Garden
London WC2E 8LU
Tel: 44 20 7240 0856
Fax: 44 20 7379 0609
Web site: http://www.eurospanonline.com

Library of Congress Cataloging-in-Publication Data

Advances in machine learning applications in software engineering / Du Zhang and Jeffrey J.P. Tsai, editors.
p. cm.
Summary: "This book provides analysis, characterization and refinement of software engineering data in terms of machine learning methods. It depicts applications of several machine learning approaches in software systems development and deployment, and the use of machine learning methods to establish predictive models for software quality while offering readers suggestions by proposing future work in this emerging research field"--Provided by publisher.
Includes bibliographical references and index.
ISBN 1-59140-941-1 (hardcover) -- ISBN 1-59140-942-X (softcover) -- ISBN 1-59140-943-8 (ebook)
1. Software engineering. 2. Self-adaptive software. 3. Application software. 4. Machine learning. I. Zhang, Du. II. Tsai, Jeffrey J.-P.
QA76.758.A375 2007
005.1--dc22
2006031366

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

Advances in Machine Learning Applications in Software Engineering

Table of Contents

Preface

Machine learning is the study of how to build computer programs that improve their performance at some task through experience. The hallmark of machine learning is that it results in an improved ability to make better decisions. Machine learning algorithms have proven to be of great practical value in a variety of application domains. Not surprisingly, the field of software engineering turns out to be a fertile ground where many software development and maintenance tasks could be formulated as learning problems and approached in terms of learning algorithms.

To meet the challenge of developing and maintaining large and complex software systems in a dynamic and changing environment, machine learning methods have been playing an increasingly important role in many software development and maintenance tasks. The past two decades have witnessed an increasing interest, and some encouraging results and publications in machine learning application to software engineering. As a result, a crosscutting niche area emerges. Currently, there are efforts to raise the awareness and profile of this crosscutting, emerging area, and to systematically study various issues in it. It is our intention to capture, in this book, some of the latest advances in this emerging niche area.

Machine Learning Methods

Machine learning methods fall into the following broad categories: *supervised* learning, *unsupervised* learning, *semi-supervised* learning, *analytical* learning, and *reinforcement* learning. Supervised learning deals with learning a target function from labeled examples. Unsupervised learning attempts to learn patterns and associations from a set of objects that do not have attached class labels. Semi-supervised learning is learning from a combination of labeled and unlabeled examples. Analytical learning relies on domain theory or background knowledge, instead of labeled examples, to learn a target function. Reinforcement learning is concerned with learning a control policy through reinforcement from an environment.

There are a number of important issues in machine learning:

- How is a target function represented and specified (based on the formalism used to represent a target function, there are different machine learning approaches)? What are the interpretability, complexity, and properties of a target function? How does it generalize?
- What is the hypothesis space (the search space)? What are its properties?
- What are the issues in the search process for a target function? What are heuristics and bias utilized in searching for a target function?
- Is there any background knowledge or domain theory available for the learning process?
- What properties do the training data have?
- What are the theoretical underpinnings and practical issues in the learning process?

The following are some frequently-used machine learning methods in the aforementioned categories.

In concept learning, a target function is represented as a conjunction of constraints on attributes. The hypothesis space H consists of a lattice of possible conjunctions of attribute constraints for a given problem domain. A least-commitment search strategy is adopted to eliminate hypotheses in H that are not consistent with the training set D. This will result in a structure called the version space, the subset of hypotheses that are consistent with the training data. The algorithm, called the candidate elimination, utilizes the generalization and specialization operations to produce the version space with regard to H and D. It relies on a language (or restriction) bias that states that the target function is contained in H. This is an eager and supervised learning method. It is not robust to noise in data and does not have support for prior knowledge accommodation.

In decision tree learning, a target function is defined as a decision tree. Search in decision tree learning is often guided by an entropy-based information gain measure that indicates how much information a test on an attribute yields. Learning algorithms often have a bias for small trees. It is an eager, supervised, and unstable learning method, and is susceptible to noisy data, a cause for overfitting. It cannot accommodate prior knowledge during the learning process. However, it scales up well with large data in several different ways.

In neural network learning, given a fixed network structure, learning a target function amounts to finding weights for the network such that the network outputs are the same as (or within an acceptable range of) the expected outcomes as specified in the training data. A vector of weights in essence defines a target function. This makes the target function very difficult for human to read and interpret. This is an eager, supervised, and unstable learning approach and cannot accommodate prior knowledge. A popular algorithm for feed-forward networks is backpropagation, which adopts a gradient descent search and sanctions an inductive bias of smooth interpolation between data points.

Bayesian learning offers a probabilistic approach to inference, which is based on the assumption that the quantities of interest are dictated by probability distributions, and that optimal decisions or classifications can be reached by reasoning about these probabilities along with observed data. Bayesian learning methods can be divided into two groups based

on the outcome of the learner: the ones that produce the most probable hypothesis given the training data, and the ones that produce the most probable classification of a new instance given the training data. A target function is thus explicitly represented in the first group, but implicitly defined in the second group. One of the main advantages is that it accommodates prior knowledge (in the form of Bayesian belief networks, prior probabilities for candidate hypotheses, or a probability distribution over observed data for a possible hypothesis). The classification of an unseen case is obtained through combined predictions of multiple hypotheses. It also scales up well with large data. It is an eager and supervised learning method and does not require search during learning process. Though it has no problem with noisy data, Bayesian learning has difficulty with small data sets. Bayesian learning adopts a bias that is based on the minimum description length principle.

Genetic algorithms and genetic programming are both biologically-inspired learning methods. A target function is represented as bit strings in genetic algorithms, or as programs in genetic programming. The search process starts with a population of initial hypotheses. Through the crossover and mutation operations, members of current population give rise to the next generation of population. During each step of the iteration, hypotheses in the current population are evaluated with regard to a given measure of fitness, with the fittest hypotheses being selected as members of the next generation. The search process terminates when some hypothesis h has a fitness value above some threshold. Thus, the learning process is essentially embodied in the generate-and-test beam search. The bias is fitness-driven. There are generational and steady-state algorithms.

Instance-based learning is a typical lazy learning approach in the sense that generalizing beyond the training data is deferred until an unseen case needs to be classified. In addition, a target function is not explicitly defined; instead, the learner returns a target function value when classifying a given unseen case. The target function value is generated based on a subset of the training data that is considered to be local to the unseen example, rather than on the entire training data. This amounts to approximating a different target function for a distinct unseen example. This is a significant departure from the eager learning methods where a single target function is obtained as a result of the learner generalizing from the entire training data. The search process is based on statistical reasoning, and consists in identifying training data that are close to the given unseen case and producing the target function value based on its neighbors. Popular algorithms include: K-nearest neighbors, case-based reasoning, and locally weighted regression.

Because a target function in inductive logic programming is defined by a set of (propositional or first-order) rules, it is highly amenable to human readability and interpretability. It lends itself to incorporation of background knowledge during learning process, and is an eager and supervised learning. The bias sanctioned by ILP includes rule accuracy, FOIL-gain, or preference of shorter clauses. There are a number of algorithms: SCA, FOIL, PROGOL, and inverted resolution.

Instead of learning a non-linear target function from data in the input space directly, support vector machines use a kernel function (defined in the form of inner product of training data) to transform the training data from the input space into a high dimensional feature space F first, and then learn the optimal linear separator (a hyperplane) in F. A decision function, defined based on the linear separator, can be used to classify unseen cases. Kernel functions play a pivotal role in support vector machines. A kernel function relies only on a subset of the training data called support vectors.

In ensemble learning, a target function is essentially the result of combining, through weighted or unweighted voting, a set of component or base-level functions called an ensemble. An ensemble can have a better predictive accuracy than its component function if (1) individual functions disagree with each other, (2) individual functions have a predictive accuracy that is slightly better than random classification (e.g., error rates below 0.5 for binary classification), and (3) individual functions' errors are at least somewhat uncorrelated. ensemble learning can be seen as a learning strategy that addresses inadequacies in training data (insufficient information in training data to help select a single best $h \in H$), in search algorithms (deployment of multiple hypotheses amounts to compensating for less than perfect search algorithms), and in the representation of H (weighted combination of individual functions makes it possible to represent a true function $f \notin H$). Ultimately, an ensemble is less likely to misclassify than just a single component function.

Two main issues exist in ensemble learning: ensemble construction and classification combination. There are bagging, cross-validation, and boosting methods for constructing ensembles, and weighted vote and unweighted vote for combining classifications. The AdaBoost algorithm is one of the best methods for constructing ensembles of decision trees.

There are two approaches to ensemble construction. One is to combine component functions that are homogeneous (derived using the same learning algorithm and being defined in the same representation formalism, for example, an ensemble of functions derived by decision tree method) and weak (slightly better than random guessing). Another approach is to combine component functions that are heterogeneous (derived by different learning algorithms and being represented in different formalisms, for example, an ensemble of functions derived by decision trees, instance-based learning, Bayesian learning, and neural networks) and strong (each of the component functions performs relatively well in its own right).

Multiple instance learning deals with the situation in which each training example may have several variant instances. If we use a bag to indicate the set of all variant instances for a training example, then for a Boolean class the label for the bag is positive if there is at least one variant instance in the bag that has a positive label. A bag has a negative label if all variant instances in the bag have a negative label. The learning algorithm is to approximate a target function that can classify every variant instance of an unseen negative example as negative, and at least one variant instance of an unseen positive example as positive.

In unsupervised learning, a learner is to analyze a set of objects that do not have their class labels, and discern the categories to which objects belong. Given a set of objects as input, there are two groups of approaches in unsupervised learning: density estimation methods that can be used in creating statistical models to capture or explain underlying patterns or interesting structures behind the input, and feature extraction methods that can be used to glean statistical features (regularities or irregularities) directly from the input. Unlike supervised learning, there is no direct measure of success for unsupervised learning. In general, it is difficult to establish the validity of inferences from the output unsupervised learning algorithms produce. Most frequently utilized methods under unsupervised learning include: association rules, cluster analysis, self-organizing maps, and principal component analysis.

Semi-supervised learning relies on a collection of labeled and unlabeled examples. The learning starts with using the labeled examples to obtain an initial target function, which is then used to classify the unlabeled examples, thus generating additional labeled examples. The learning process will be iterated on the augmented training set. Some semi-supervised learning methods include: expectation-maximization with generative mixture models, self-training, co-training, transductive support vector machines, and graph-based methods.