A Brief Introduction into Machine Learning

Gunnar Rätsch
Friedrich Miescher Laboratory of the Max Planck Society,
Spemannstraße 37, 72076 Tübingen, Germany
http://www.tuebingen.mpg.de/~raetsch

1 Introduction

The Machine Learning field evolved from the broad field of *Artificial Intelligence*, which aims to mimic intelligent abilities of humans by machines. In the field of *Machine Learning* one considers the important question of how to make machines able to "learn". Learning in this context is understood as *inductive inference*, where one observes *examples* that represent incomplete information about some "statistical phenomenon". In *unsupervised* learning one typically tries to uncover hidden regularities (e.g. clusters) or to detect anomalies in the data (for instance some unusual machine function or a network intrusion). In *supervised learning*, there is a *label* associated with each example. It is supposed to be the answer to a question about the example. If the label is discrete, then the task is called *classification problem* – otherwise, for real-valued labels we speak of a *regression problem*. Based on these examples (including the labels), one is particularly interested to *predict* the answer for other cases before they are explicitly observed. Hence, learning is not only a question of remembering but also of *generalization to unseen cases*.

2 Supervised Classification

An important task in Machine Learning is *classification*, also referred to as pattern recognition, where one attempts to build algorithms capable of automatically constructing methods for distinguishing between different exemplars, based on their differentiating patterns.

Watanabe [1985] described a pattern as "the opposite of chaos; it is an entity, vaguely defined, that could be given a name." Examples of patterns are human faces, text documents, handwritten letters or digits, EEG signals, and the DNA sequences that may cause a certain disease. More formally, the goal of a (supervised) classification task is to find a functional mapping between the input data X, describing the input pattern, to a class label Y (e.g. -1 or $+1$), such that $Y = f(X)$. The construction of the mapping is based on so-called *training data* supplied to the classification algorithm. The aim is to accurately predict the correct label on unseen data.

A pattern (also: "example") is described by its *features*. These are the characteristics of the examples for a given problem. For instance, in a face recognition task some features could be the color of the eyes or the distance between the eyes. Thus, the input

to a pattern recognition task can be viewed as a two-dimensional matrix, whose axes are the examples and the features.

Pattern classification tasks are often divided into several sub-tasks:

1. Data collection and representation.
2. Feature selection and/or feature reduction.
3. Classification.

Data collection and representation are mostly problem-specific. Therefore it is difficult to give general statements about this step of the process. In broad terms, one should try to find invariant features, that describe the differences in classes as best as possible.

Feature selection and feature reduction attempt to reduce the dimensionality (i.e. the number of features) for the remaining steps of the task. Finally, the classification phase of the process finds the actual mapping between patterns and labels (or targets). In many applications the second step is not essential or is implicitly performed in the third step.

3 Classification Algorithms

Although Machine Learning is a relatively young field of research, there exist more learning algorithms than I can mention in this introduction. I chose to describe six methods that I am frequently using when solving data analysis tasks (usually classification). The first four methods are traditional techniques that have been widely used in the past and work reasonably well when analyzing low dimensional data sets with not too few labeled training examples. In the second part I will briefly outline two methods (Support Vector Machines & Boosting) that have received a lot of attention in the Machine Learning community recently. They are able to solve high-dimensional problems with very few examples (e.g. fifty) quite accurately and also work efficiently when examples are abundant (for instance several hundred thousands of examples).

3.1 Traditional Techniques

k-Nearest Neighbor Classification Arguably the simplest method is the *k-Nearest Neighbor* classifier [Cover and Hart, 1967]. Here the k points of the training data closest to the test point are found, and a label is given to the test point by a majority vote between the k points. This method is highly intuitive and attains – given its simplicity – remarkably low classification errors, but it is computationally expensive and requires a large memory to store the training data.

Linear Discriminant Analysis computes a hyperplane in the input space that minimizes the within-class variance and maximizes the between class distance [Fisher, 1936]. It can be efficiently computed in the linear case even with large data sets. However, often a linear separation is not sufficient. Nonlinear extensions by using kernels exist [Mika et al., 2003], however, making it difficult to apply it to problems with large training sets.

Decision Trees Another intuitive class of classification algorithms are *decision trees*. These algorithms solve the classification problem by repeatedly partitioning the in-

put space, so as to build a tree whose nodes are as pure as possible (that is, they contain points of a single class). Classification of a new test point is achieved by moving from top to bottom along the branches of the tree, starting from the root node, until a terminal node is reached. Decision trees are simple yet effective classification schemes for small datasets. The computational complexity scales unfavorably with the number of dimensions of the data. Large datasets tend to result in complicated trees, which in turn require a large memory for storage. The C4.5 implementation by Quinlan [1992] is frequently used and can be downloaded at `http://www.rulequest.com/Personal`.

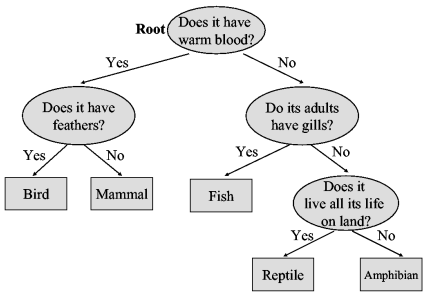

Figure 1: An example a decision tree (Figure taken from Yom-Tov [2004]).

Neural Networks are perhaps one of the most commonly used approaches to classification. Neural networks (suggested first by Turing [1992]) are a computational model inspired by the connectivity of neurons in animate nervous systems. A further boost to their popularity came with the proof that they can approximate any function mapping via the Universal Approximation Theorem [Haykin, 1999]. A simple scheme for a neural network is shown in Figure 2. Each circle denotes a computational element referred to as a *neuron*, which computes a weighted sum of its inputs, and possibly performs a nonlinear function on this sum. If certain classes of nonlinear functions are used, the function computed by the network can approximate any function (specifically a mapping from the training patterns to the training targets), provided enough neurons exist in the network and enough training examples are provided.

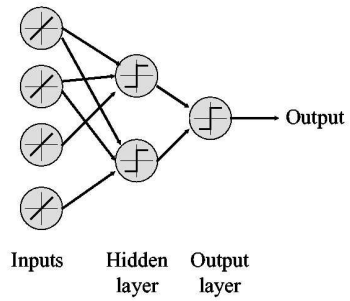

Figure 2: A schematic diagram of a neural network. Each circle in the hidden and output layer is a computational element known as a neuron. (Figure taken from Yom-Tov [2004])

3.2 Large Margin Algorithms

Machine learning rests upon the theoretical foundation of *Statistical Learning Theory* [e.g. Vapnik, 1995] which provides conditions and guarantees for good generalization of learning algorithms. Within the last decade, *large margin classification techniques* have emerged as a practical result of the theory of generalization. Roughly speaking, the margin is the distance of the example to the separation boundary and a large margin classifier generates decision boundaries with large margins to almost all training examples. The two most widely studied classes of large margin classifiers are *Support Vector Machines* (SVMs) [Boser et al., 1992, Cortes and Vapnik, 1995] and *Boosting* [Valiant, 1984, Schapire, 1992]:

Support Vector Machines work by mapping the training data into a feature space by the aid of a so-called kernel function and then separating the data using a *large margin hyperplane* (cf. Algorithm 1). Intuitively, the kernel computes a similarity

between two given examples. Most commonly used kernel functions are *RBF kernels* $k(\mathbf{x}, \mathbf{x}') = \exp\left(\frac{\|\mathbf{x}-\mathbf{x}'\|^2}{\sigma^2}\right)$ and *polynomial kernels* $k(\mathbf{x}, \mathbf{x}') = (\mathbf{x} \cdot \mathbf{x}')^d$.

The SVM finds a large margin separation between the training examples and previously unseen examples will often be close to the training examples. Hence, the large margin then ensures that these examples are correctly classified as well, i.e., the decision rule *generalizes*. For so-called *positive definite* kernels, the optimization problem can be solved efficiently and SVMs have an interpretation as a hyperplane separation in a high dimensional feature space [Vapnik, 1995, Müller et al., 2001, Schölkopf and Smola, 2002]. Support Vector Machines have been used on million dimensional data sets and in other cases with more than a million examples [Mangasarian and Musicant, 2001]. Research papers and implementations can be downloaded from the kernel machines web-site `http://www.kernel-machines.org`.

Algorithm 1 The Support Vector Machine with regularization parameter C.

Given labeled sequences $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_m, y_m)$ ($\mathbf{x} \in X$ and $y \in \{-1, +1\}$) and a kernel k, the SVM computes a function

$$f(s) = \sum_{i=1}^{m} \alpha_i k(\mathbf{x}_i, \mathbf{x}) + b,$$

where the coefficients α_i and b are found by solving the optimization problem

$$\begin{array}{ll} \text{minimize} & \sum_{i,j=1}^{m} \alpha_i \alpha_j k(s_i, s_j) + C \sum_{i=1}^{m} \xi_i \\ \text{subject to} & y_i f(\mathbf{x}_i) \geq 1 - \xi_i \end{array}$$

Boosting The basic idea of boosting and *ensemble learning algorithms* in general is to iteratively combine relatively simple *base hypotheses* – sometimes called *rules of thumb* – for the final prediction. One uses a so-called *base learner* that generates the base hypotheses. In boosting the base hypotheses are linearly combined. In the case of *two-class classification*, the final prediction is the weighted majority of the votes. The combination of these simple rules can boost the performance drastically. It has been shown that Boosting has strong ties to support vector machines and large margin classification [Rätsch, 2001, Meir and Rätsch, 2003]. Boosting techniques have been used on very high dimensional data sets and can quite easily deal with than hundred thousands of examples. Research papers and implementations can be downloaded from `http://www.boosting.org`.

4 Summary

Machine Learning research has been extremely active the last few years. The result is a large number of very accurate and efficient algorithms that are quite easy to use for a practitioner. It seems rewarding and almost mandatory for (computer) scientist and engineers to learn how and where Machine Learning can help to automate tasks

or provide predictions where humans have difficulties to comprehend large amounts of data. The long list of examples where Machine Learning techniques were successfully applied includes: Text classification and categorization [e.g. Joachims, 2001] (for instance spam filtering), network intrusion detection [e.g. Laskov et al., 2004], Bioinformatics (e.g. cancer tissue classification, gene finding; e.g. Furey et al. [2000], Zien et al. [2000], Sonnenburg et al. [2002]), brain computer interfacing [e.g. Blankertz et al., 2003], monitoring of electric appliances [e.g. Onoda et al., 2000], optimization of hard disk caching strategies [e.g. Gramacy et al., 2003] and disk spin-down prediction [e.g. Helmbold et al., 2000]), drug discovery [e.g. Warmuth et al., 2003]), high-energy physics particle classification, recognition of hand writing, natural scene analysis etc.

Obviously, in this brief summary I have to be far from being complete. I did not mention regression algorithms (e.g. ridge regression, regression trees), unsupervised learning algorithms (such as clustering, principle component analysis), reinforcement learning, online learning algorithms or model-selection issues. Some of these techniques extend the applicability of Machine Learning algorithms drastically and would each require an introduction for them self. I would like to refer the interested reader to two introductory books [Mendelson and Smola, 2003, von Luxburg et al., 2004] which are the result of the last annual Summer Schools on Machine Learning (cf. `http://mlss.cc`).

Acknowledgments I would like to thank Julia Lüning for encouraging me to write this paper. Furthermore, thanks to Sören Sonnenburg and Nora Toussaint for proofreading the manuscript. For the preparation of this work I used and modified some parts of Müller et al. [2001], Rätsch [2001], Meir and Rätsch [2003], Yom-Tov [2004]. This work was partly funded by the PASCAL Network of Excellence project (EU #506778).

References

B.. Blankertz, G. Dornhege, C. Schäfer, R. Krepki, J. Kohlmorgen, K.-R. Müller, V. Kunzmann, F. Losch, and G. Curio. BCI bit rates and error detection for fast-pace motor commands based on single-trial EEG analysis. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, 11:127–131, 2003.

B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In D. Haussler, editor, *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152, 1992.

C. Cortes and V.N. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.

T.M. Cover and P.E. Hart. Nearest neighbor pattern classifications. *IEEE transaction on information theory*, 13(1):21—27, 1967.

R.A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7:179–188, 1936.

T. Furey, N. Cristianini, N. Duffy, D. Bednarski, M. Schummer, and D. Haussler. Support vector machine classification and validation of cancer tissue samples using microarray expression data. *Bioinformatics*, 10:906–914, 2000.

R.B. Gramacy, M.K. Warmuth, S.A. Brandt, and I. Ari. Adaptive caching by refetching. In *In Advances in Neural Information Processing Systems 15*, pages 1465–1472. MIT Press, 2003.

S. Haykin. *Neural Networks: A comprehensive foundation, 2nd Ed.* Prentice-Hall, 1999.

D.P. Helmbold, D.D.E. Long, T.L. Sconyers, and B. Sherrod. Adaptive disk spin-down for mobile computers. *Mobile Networks and Applications*, 5(4):285–297, 2000.

T. Joachims. *The Maximum-Margin Approach to Learning Text Classifiers*. PhD thesis, Computer Science Dept., University of Dortmund, 2001.

P. Laskov, C. Schäfer, and I. Kotenko. Intrusion detection in unlabeled data with quarter-sphere support vector machines. In *Proc. DIMVA*, pages 71–82, 2004.

O.L. Mangasarian and D.R. Musicant. Lagrangian support vector machines. *JMLR*, 1:161–177, March 2001.

R. Meir and G. Rätsch. An introduction to boosting and leveraging. In S. Mendelson and A. Smola, editors, *Advanced Lectures on Machine Learning*, LNAI, pages 119–184. Springer, 2003.

S. Mendelson and A. Smola, editors. *Advanced Lectures on Machine Learning*, volume 2600 of *LNAI*. Springer, 2003.

S. Mika, G. Rätsch, J. Weston, B. Schölkopf, A.J. Smola, and K.-R. Müller. Constructing descriptive and discriminative nonlinear features: Rayleigh coefficients in kernel feature spaces. *IEEE Transactions on Patterns Analysis and Machine Intelligence*, 25(5):623–627, May 2003.

K.-R. Müller, S. Mika, G. Rätsch, K. Tsuda, and B. Schölkopf. An introduction to kernel-based learning algorithms. *IEEE Transactions on Neural Networks*, 12(2):181–201, 2001.

T. Onoda, G. Rätsch, and K.-R. Müller. A non-intrusive monitoring system for household electric appliances with inverters. In H. Bothe and R. Rojas, editors, *Proc. of NC'2000*, Berlin, 2000. ICSC Academic Press Canada/Switzerland.

J.R. Quinlan. *C4.5: Programs for Machine Learning.* Morgan Kaufmann, 1992.

G. Rätsch. *Robust Boosting via Convex Optimization*. PhD thesis, University of Potsdam, Neues Palais 10, 14469 Potsdam, Germany, October 2001.

R.E. Schapire. *The Design and Analysis of Efficient Learning Algorithms*. PhD thesis, MIT Press, 1992.

B. Schölkopf and A.J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.

S. Sonnenburg, G. Rätsch, A. Jagota, and K.-R. Müller. New methods for splice-site recognition. In JR. Dorronsoro, editor, *Proc. International conference on artificial Neural Networks – ICANN'02*, pages 329–336. LNCS 2415, Springer Berlin, 2002.

A.M. Turing. Intelligent machinery. In D.C. Ince, editor, *Collected works of A.M. Turing: Mechanical Intelligence*, Amsterdam, The Netherlands, 1992. Elsevier Science Publishers.

L.G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, November 1984.

V.N. Vapnik. *The nature of statistical learning theory*. Springer Verlag, New York, 1995.

U. von Luxburg, O. Bousquet, and G. Rätsch, editors. *Advanced Lectures on Machine Learning*, volume 3176 of *LNAI*. Springer, 2004.

M. K Warmuth, J. Liao, G. Rätsch, Mathieson. M., S. Putta, and C. Lemmem. Support Vector Machines for active learning in the drug discovery process. *Journal of Chemical Information Sciences*, 43(2):667–673, 2003.

W. Watanabe. *Pattern recognition: Human and mechanical*. Wiley, 1985.

E. Yom-Tov. An introduction to pattern classification. In U. von Luxburg, O. Bousquet, and G. Rätsch, editors, *Advanced Lectures on Machine Learning*, volume 3176 of *LNAI*, pages 1–23. Springer, 2004.

A. Zien, G. Rätsch, S. Mika, B. Schölkopf, T. Lengauer, and K.-R. Müller. Engineering Support Vector Machine Kernels That Recognize Translation Initiation Sites. *BioInformatics*, 16(9):799–807, September 2000.