4 Discovering Informative Patterns and Data Cleaning

I. Guyon, N. Matić, and V. Vapnik
AT&T Bell Laboratories
isabelle@neural.att.com

Abstract

We present a method for discovering informative patterns from data. With this method, large databases can be reduced to only a few representative data entries. Our framework also encompasses methods for cleaning databases containing corrupted data. Both on-line and off-line algorithms are proposed and experimentally checked on databases of handwritten images. The generality of the framework makes it an attractive candidate for new applications in knowledge discovery.

Keywords: knowledge discovery, machine learning, informative patterns, data cleaning, information gain.

4.1 Introduction

Databases often contain redundant data. It would be convenient if large databases could be replaced by only a subset of informative patterns. A difficult, yet important, problem is to define what informative patterns are. We use the learning theoretic definition [Rissanen 1989, Haussler 1991, MacKay 1992]: given a model trained on a sequence of patterns, a new pattern is informative if it is difficult to predict by a model trained on previously seen data. With that definition, we derive on-line and batch algorithms for discovering informative patterns. The techniques are developed for classification problems, but are also applicable to regression and density estimation problems.

Informative patterns are often intermixed with other "bad" outliers which correspond to errors introduced non-intentionally in the database. For databases containing errors,

our algorithms can be used for computer-aided data cleaning, with or without supervision. We review results of several experiments in handwriting recognition [Boser 1992, Matić 1992, Guyon 1993, Matić 1993] which demonstrate the usefulness of our data cleaning techniques.

4.2 Discovering Informative Patterns

In this section, we assume that the data is perfectly clean. First, we give an intuition of what informative patterns ought to be. Then, we show that this intuition coincides with the information theoretic definition. Finally, we derive algorithms to discover informative patterns.

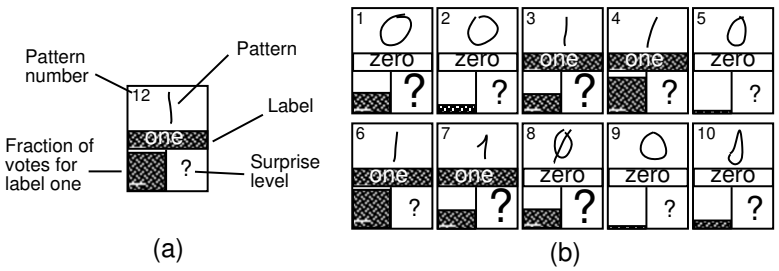

Figure 4.1
Small example database containing "zeros" and "ones". (a) A data entry. (b) A sequence of data entries during a training session showing the variation of the surprise level. The patterns which are most surprising are most informative.

4.2.1 Informative Patterns are most Surprising

In figure 4.1, we constructed a small example database containing only handwritten zeros and ones. Most patterns of a given category look similar. A typical zero is a circle and a typical one is a vertical bar. However, there exist other shapes of zeros and ones. If we wanted to keep only a few data representatives, we would probably keep at least one example of each basic shape.

To choose the best data representatives, we run an imaginary experiment. Imagine that we find 100 people who do not know what the shape of a zero and that of a one are. We teach these people to recognize zeros and ones by letting them examine the patterns of our database in sequence. Every time we show them a new image, we first hide the label and let them make a guess.

We represent in figure 4.1 the fraction of votes for label "one" for a particular sequence of data. Since we assume that our subjects have never seen a zero nor a one before the

experiment, about 50% guess "zero" and 50% guess "one" when they are shown the first pattern. But, for the second example of the same shape, the majority makes the correct guess. As learning goes on, familiar shapes are guessed more and more accurately and the percentage of wrong guesses raises only occasionally when a new shape appears.

We represent with the size of a question mark the average amount of surprise generated among our subjects when the true label is uncovered. People who guess the correct label are not surprised while people who make the wrong guess are surprised. We see from figure 4.1 that a large average surprise level coincides with the appearance of a new shape. Therefore, the level of surprise is a good indication of how informative a pattern is.

More formally, let x_k be a pattern, $y_k \in \{0, 1\}$ its associated label, $k-1$ the number of data entries seen thus far and $\hat{y}_k$ the label predicted for pattern x_k. The level of surprise is inversely related to the probability of guessing the correct label $P_k(\hat{y}_k = y_k) = P(\hat{y}_k = y_k | x_k, (x_0, y_0), (x_1, y_1), ...(x_{k-1}, y_{k-1}))$. This probability of making the correct guess is precisely what is involved in Shannon's information gain:

$$I(k) = -\log P_k(\hat{y}_k = y_k) = -y_k \log P_k(\hat{y}_k = 1) - (1 - y_k) \log(1 - P_k(\hat{y}_k = 1)) \qquad (4.2.1)$$

In the information theoretic sense, the data entries that are most informative are those that are most surprising.

4.2.2 Machine Learning Techniques to Estimate the Information Gain

It is somewhat unrealistic to hire 100 ignorant people to estimate the information gain. Let us now replace people with machines.

In the machine learning framework, patterns drawn from a database are presented to the learning machine which makes predictions. The prediction error is evaluated and used to improve the accuracy of further predictions by adjusting the learning machine parameters.

Assume first that we trained 100 different learning machines (substituting our 100 people), each one predicting its own value of $\hat{y}_k$. This is similar to a "Bayesian" approach [MacKay 1992] for which predictions are made according to the vote of an ensemble of machines. Formula 4.2.1 can be readily used to determine the information gain. Although this is a perfectly valid method, we propose here a more practical one: we train a single learning machine to provide an estimate $\hat{P}_k(y_k = 1)$ of the probability that the correct label is "one" (figure 4.2). Our prediction $\hat{y}_k$ will be the most likely category according to $\hat{P}_k(y_k = 1)$. In formula 4.2.1, we substitute $\hat{P}_k(y_k = 1)$ for $P_k(\hat{y}_k = 1)$.

Many machine learning techniques can be applied to discover informative patterns. For example, the learning machine can be a simple K-nearest-neighbor classifier [Duda 1973].

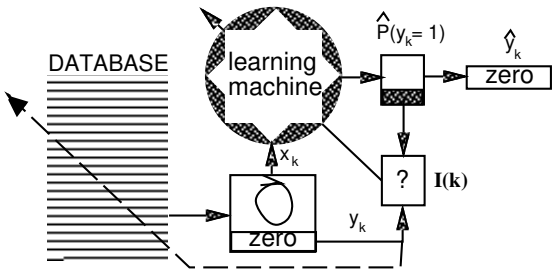

Figure 4.2
A learning machine used to predict the information gain (or surprise level). In the classical machine learning framework, the goal is to train the learning machine, either to provide a model of the data or to make predictions. In our framework the goal is to discover the informative patterns of the database (dashed line).

All patterns presented to the classifier are stored. $\hat{P}_k(y_k = 1)$ is given by the fraction of the K training patterns that are nearest to x_k which have label "one".

Another example is a neural network trained with a "cross-entropy" cost function which is the average over the training patterns of the Shannon information gain. The squared loss $(y_k - \hat{P}_k(y_k = 1))^2$, which also is often used, provides an *information criterion* which ranks patterns in the same order as the Shannon information gain. This is due to the fact that on the $[0, 1]$ interval, $-\log P(y_k = 1)$ and $(1 - P(y_k = 1))^2$ are both convex functions, one bounding the other one. Therefore, $-\log P(y_1 = 1) < -\log P(y_2 = 1)$ if and only if $(1 - y_1)^2 < (1 - y_2)^2$.

In the following, we use the size of the symbol "question mark", in the figures, and the notation $I(k)$, in the text, to represent the *information criteria* used to measure the "surprise level."

4.2.3 On-line Algorithms and Batch Algorithms

In an on-line algorithm, patterns are presented in sequence and the learning machine adjusts its parameters at each presentation of a new pattern. This is a situation similar to that of our example of section 4.2.1. In that case, we could say that a pattern is informative if the information criterion exceeds a pre-defined threshold. The disadvantage of this method is that informative patterns depend on the sequence in which patterns are presented. Nonetheless, there may be practical situations where the data is only available on-line.

In a batch algorithm, conversely, all data entries are available at once and the information criterion is independent of pattern ordering. This implies that if there are p data entries in the database we need to train p machines, each one on all the data but one

pattern, and then try to predict that last pattern. In practice, this is feasible only if training is inexpensive, as for the K-nearest-neighbor algorithm.

For other batch algorithms, we prefer to train the learning machine only once on all of the data. We then approximate the information criterion of each pattern with an estimate of how much the *cumulative information criterion* would decrease if we removed that pattern from the training set. For instance, on can use as cumulative information criterion the value of the cost function on the training set at the solution; the information criterion of each pattern is then its contribution to the overall cost.

For batch algorithms all the patterns of the database are uniquely ranked according to their information criterion. The m most informative patterns can be selected to represent the entire database.

4.2.4 Minimax Algorithms

Minimax algorithms are batch algorithms that are particularly well suited to discovering informative patterns. Most algorithms train the learning machine to minimize the average loss (e.g., the mean-square-error). Minimax algorithms minimize the maximum loss:

$$\min_{w} \max_{k} \ell(k) \tag{4.2.2}$$

where w represents the parameters of the learning machine and k runs over all data entries, and $\ell(.)$ is some loss function. Minimax algorithms are extreme cases of "active learning" methods which emphasize patterns with large information gain [MacKay 1992]. The solution of a minimax algorithm is a function of only a small subset of the training patterns, precisely called "informative patterns" [Vapnik 1982]. These are the patterns that have maximum loss.

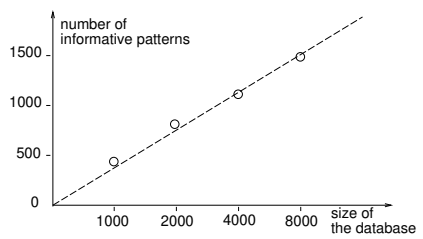

Figure 4.3
Variation of the number of informative patterns as a function of the number of patterns in the database. Experiments are carried out on a database of handwritten digits, encoded with 16 real-valued global features derived from the pen trajectory information. A polynomial classifier of degree 2 is trained with a minimax algorithm (Optimal Margin Classifier). The informative patterns are the patterns for which $I(k) = \alpha_k^* \neq 0$.

In Appendix A, we describe a minimax algorithm for classification problems: the Optimum Margin Classifier [Vapnik 1982, Boser 1992, Guyon 1993]. The algorithm maximizes the minimum distance of the training patterns to the decision boundary. It can be shown that the solution w^* is a linear combination of basis functions of the informative patterns:

$$w^* = \sum_{k=1}^{p} (2y_k - 1)\alpha_k^* \varphi(x_k), \quad \alpha_k^* \geq 0, \tag{4.2.3}$$

where $y_k \in \{0, 1\}$ indicates the class membership, the $\varphi(.)$ are basis functions and α_k^* are coefficients (all null except for the informative patterns). We use the value of the cost function at the solution as the *cumulative information criterion*:

$$I = \sum_{k=1}^{p} \alpha_k^*, \tag{4.2.4}$$

from which we derive an estimate of the information loss incurred by removing the informative pattern k:

$$I(k) = \alpha_k^*. \tag{4.2.5}$$

One important question is: what is the rate of growth of the number of informative patterns as a function of the size of the database? The results of experiments carried out on a database of handwritten digits using the Optimal Margin Classifier suggest a logarithmic growth, in the regime when the number of patterns is small compared to the total number of distinct patterns (figure 4.3).

Other minimax algorithms have been proposed for classification, regression and density estimation [Demyanov 1972, Cortes 1993].

4.3 Data Cleaning

In this section we tackle the problem of real world databases which may contain corrupted data entries. We propose data cleaning algorithms and analyze experimental results.

4.3.1 Garbage Patterns are also Surprising

The information theoretic definition of an informative pattern does not always coincide with the common sense definition. In figure 4.4, we show examples of patterns drawn from our database of "zeros" and "ones" which have a large information gain. We see two kinds of patterns:

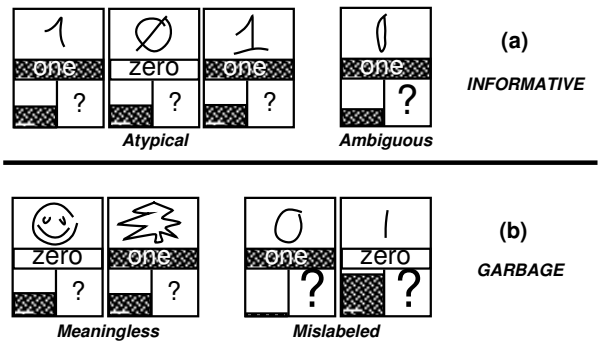

Figure 4.4
(a) Informative patterns versus (b) garbage patterns. Informative patterns are intermixed with garbage patterns which also generate a lot of surprise (i.e. have a large information gain).

- Patterns that are actually informative: atypical shapes or ambiguous shapes.
- Garbage patterns: meaningless or mislabeled patterns.

Truly informative patterns should be kept in the database while garbage patterns should be eliminated.

Purely automatic cleaning could be performed by eliminating systematically all patterns with suspiciously large information criterion. However, this is dangerous since valuable informative patterns may also be eliminated. Purely manual cleaning, by examining all patterns in the database, is tedious and impractical for very large databases. We propose a computer-aided cleaning method where a human operator must check only those patterns that have largest information criterion and are therefore most suspicious.

4.3.2 On-line Algorithms and Batch Algorithms

In figure 4.5 we present our on-line version of data cleaning. It combines cleaning and training in one single session. The learning machine is initially trained with a few clean examples. At step k of the cleaning process, a new pattern x_k is presented to the learning machine. The prediction of the learning machine and the desired value y_k are used to compute the information criterion $I(k)$. If $I(k)$ is below a given threshold θ, the pattern is directly sent to the learning machine for adaptation. Otherwise, the pattern is sent to the human operator for checking. Depending on the decision of the operator, the pattern is either trashed or sent to the learning machine for adaptation.

When all the data has been processed, both training and cleaning are completed, since the learning machine is trained only on clean data. This is an advantage if further use is made of the learning machine. But on-line cleaning has several disadvantages:

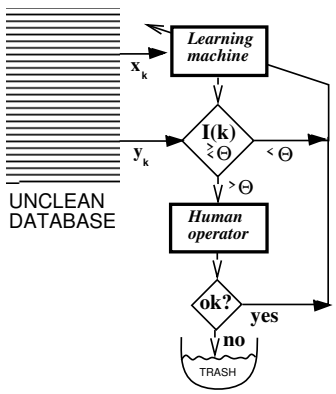

Figure 4.5
Flow diagram of on-line cleaning. In the process of learning, patterns for which the information criterion exceeds a threshold θ are examined by a human operator. Good patterns are kept in the database and sent to the recognizer for adaptation. Bad patterns are removed from the database.

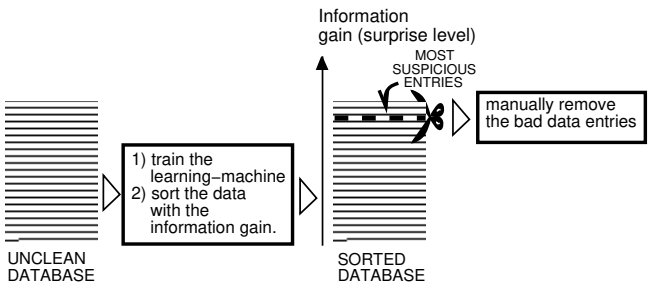

Figure 4.6
Block diagram of batch cleaning. The learning machine is trained on unclean data. The information criterion is evaluated for each pattern by estimating how much information would be lost if that pattern are removed. The database is sorted according to the information criterion. Only the top ranked patterns are examined for cleaning by the human operator. After cleaning, the learning machine needs to be retrained to obtain a good model and/or make good predictions.

- One has to find an appropriate threshold θ.
- Training the learning machine may be slower than checking the patterns which may result in wasting the time of the operator.
- The method depends on the order of presentation of the patterns; it is not possible to reverse a decision on whether a pattern is good or bad.

When possible, batch methods are preferred for cleaning (figure 4.6). Training is performed on all patterns, including garbage patterns, with any training algorithm. After training, the information criterion of the patterns is computed as explained in sections 4.2.3 and 4.2.4. Typically, one uses the contribution of the training patterns to the value of the cost function at the solution. The data entries are then sorted in decreasing order of information criterion. The patterns are examined by the operator, starting from the top (most suspicious) until the number of consecutive "good" patterns exceeds a given threshold. The classifier is then retrained on the clean data. If the database contains correlated errors, it may be necessary to iterate this procedure several times to remove all "bad" patterns.

The combination of batch cleaning and minimax algorithms is particularly attractive. In that case, only informative patterns (with non zero information criterion) need to be examined. In figure 4.7 we show the first few informative patterns obtained with a minimax classifier (the Optimum Margin Classifier [Boser 1992]). The classifier is trained to discriminate between the digit "two" and all the other digits. The patterns with largest α_k^* (our information criterion from equation 4.2.5) is a garbage pattern.

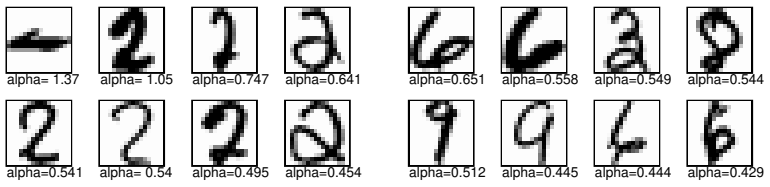

Figure 4.7
The informative patterns obtained by a minimax algorithm (the Optimum Margin Algorithm), for the separation of handwritten digit "two" against all other digit categories [Boser 1992]. Patterns are represented by a 16x16 grey-level pixel-map. The informative patterns are shown in order of decreasing information criterion.(a) Informative patterns for class 2: a garbage pattern comes first. (b) Informative patterns for all other classes: several ambiguous or atypical shapes are among the patterns with the largest information criterion.

4.3.3 Point of Optimal Cleaning

In the experiments described in this section [Matić 1992], we use the following experimental setup. We train a neural network to recognize handwritten lowercase letters. The representation is 630 local features of the pen trajectory. A training data set of 9513 characters and a validation data set of 2000 characters from disjoint set of writers is used for cleaning (the validation set does not get cleaned). The network is a convolutional neural network with local connections and shared weights. It is trained by minimizing a mean-square-error cost function with the backpropagation algorithm. Weights are initialized to small random values. Training is stopped after 30 iterations through the training set. No weight decay is used.

We may wonder how reliable our cleaning techniques are: did we examine all the patterns that should be candidates for cleaning? Among the patterns that were candidates for cleaning, did we remove too many or too few patterns?

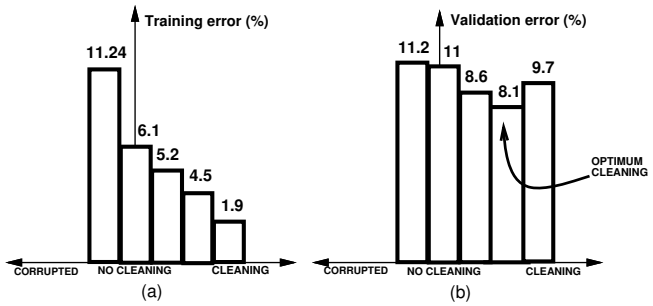

Figure 4.8
Data cleaning results showing a point of optimal cleaning [Matić 1992]. These results are obtained with an on-line cleaning method on a handwriting recognition task using a neural network classifier (see text).

We can again use our learning machine to provide us with an answer. In the experiments of figure 4.8, we vary the amount of cleaning [Matić 1992]. Starting with the "uncleaned" database, several levels of cleaning are applied by the human operator. Each stage is more strict, i.e. he lowers the tolerance for marginal-quality characters. To test the power of the cleaning technique, we also corrupt the initial, uncleaned database, by artificially mislabeling 5% of the characters in the training set (left most column). A version of on-line cleaning called "bootstrap" cleaning is used: small batches of data are processed at a time instead of a single pattern. The classification error is used as information criterion.

We observe that, as more patterns are removed through the cleaning process, the

error rate on the training set decreases. This is understandable since the learning task becomes easier and easier to the learning machine. The validation error however goes through a minimum. This is because we remove first only really bad patterns, then we start removing valuable informative patterns that we mistook for garbage patterns. As explained in the next paragraph, this behavior of the validation error is general and can be predicted theoretically. The minimum of the validation error is the point of optimal cleaning. If our validation error does not go through a minimum, more patterns should be examined and considered for cleaning.

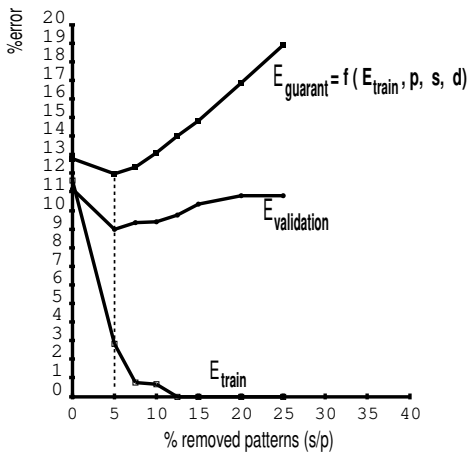

Figure 4.9
Detection of the point of optimal cleaning with the Vapnik-Chervonenkis prediction. These results are obtained with a batch cleaning method. The experimental setup is the same as in figure 4.8. $E_{guarant}$ is rescaled to fit on the figure.

It is not always possible nor desirable to split the database into a training and a validation set. Another way of obtaining the point of optimum cleaning is to use the predictions of the Vapnik-Chervonenkis (VC) theory [Vapnik 1982]. According to the VC-theory (see Appendix B), the point of optimum cleaning is the minimum of the so-called *guaranteed risk*, which is a bound on $E_{validation}$, the validation set frequency of errors:

$$E_{guarant} = E_{train}(s) + \beta \, \frac{d(\ln\frac{2(p-s)}{d} + 1) + s(\ln\frac{p}{s} + 1)}{p - s} \tag{4.3.6}$$

where β is a constant, $E_{train}(m)$ is the frequency of training errors when s suspicious patterns have been removed from the training set, p is the number of training patterns

Table 4.1
Results obtained on various databases of handwritten patterns. The cleaning time is the time spent by the operator browsing through the informative patterns (not including the learning machine training time). Only the training data is cleaned. The test errors are computed on separate unclean test sets of more than 2000 examples from a disjoint set of writers. Without cleaning, the last database is completely worthless.

References	Database	Db. size	Num. of inf. pat.	Cleaning time (h)	% test error unclean train	% test error clean train
Boser-92	OCR digits	7,300	690	1	15	10.5
Matić-93	on-line lower	16,000	3,130	1	11	6.5
Guyon-92	on-line ASCII	100,000	10,983	5	30	6

before cleaning and d is the VC-dimension. β and d are estimated as explained in Appendix B.

In figure 4.9 [Matić 1992] we compare the prediction of the VC-theory and that of the validation set. The experimental setup is the same as in the previous experiment, except that training is performed with an "active learning" technique [Guyon 1992]. The information criterion is the total time each pattern spent in the active set. The number of patterns before cleaning is $p = 9513$. We use a constant $\beta = 0.5$ and our estimation of the VC-dimension of the neural network is $d = 2000$ (approximately one third of the number of free parameters).

We obtain good agreement between the prediction of the VC-theory and that of the validation set, even with a very rough estimate of the VC-dimension. In fact, for the particular network architecture that we use, the position of the point of optimum cleaning is insensitive to a change in the estimate of the VC-dimension in the range $300 < d < 5000$.

4.4 Conclusions

We presented a computer-aided method for detecting informative patterns and cleaning data. We used this method on various databases of handwritten characters. The results are summarized in table 4.1. It is important to stress that the number of informative patterns varies sub-linearly with the number of data entries and that therefore the cleaning time varies sub-linearly with the number of data entries. The point of optimum cleaning can be predicted by the VC-theory which does not require splitting the data between a training set and a validation set. Our cleaning method clearly improves the quality of the data, as measured by the improvement in classification performance on a test set independent from the training set. Our framework is general and applies to other

problems than classification problems (regression or density estimation) which makes it attractive for new applications in knowledge discovery.

Acknowledgements Discussions with colleagues AT&T Bell Labs and at UC Berkeley are gratefully acknowledged. Special thanks to B. Boser who contributed many ideas.

Appendix 4.A: Optimal Margin Classifier (OMC)

This appendix summarizes the OMC algorithm described in [Vapnik 1982, Boser 1992, Guyon 1993].

We seek a decision function for pattern vectors $\mathbf{x}$ of dimension n belonging to one of two classes A and B. The input to the training algorithm is a set of p examples $\mathbf{x}_k$ with labels $g_k \in \{-1, 1\}$ indicating the class membership ($g_k = 2y_k - 1$). From these training examples the algorithm finds the parameters of the decision function $D(\mathbf{x})$ during a learning phase. After training, the classification of unknown patterns is predicted according to the following rules: $\mathbf{x} \in$ A, if $D(\mathbf{x}) > 0$ and $\mathbf{x} \in$ B otherwise. The OMC algorithm applies to classifiers linear in their parameters, but not restricted to linear dependencies in their input components, such as perceptrons and kernel-based classifiers.

Duality

Perceptrons and kernel-based classifiers are often considered two very distinct approaches to classification. However, for a number of training algorithms, they constitute *dual* representations of the same decision function. Perceptrons [Duda 1973] have a decision function defined as:

$$D(\mathbf{x}) = \mathbf{w} \cdot \varphi(\mathbf{x}) + b = \sum_{i=1}^{N} w_i \varphi_i(\mathbf{x}) + b, \tag{4.4.7}$$

where the φ_i are predefined functions of $\mathbf{x}$, and the w_i and b are the adjustable parameters of the decision function. This definition encompasses that of polynomial classifiers. In that particular case, the φ_i are products of components of vector $\mathbf{x}$. Kernel-based classifiers, have a decision function defined as:

$$D(\mathbf{x}) = \sum_{k=1}^{p} \alpha_k K(\mathbf{x}_k, \mathbf{x}) + b, \tag{4.4.8}$$

The coefficients α_k and the bias b are the parameters to be adjusted and the $\mathbf{x}_k$ are the training patterns. The function K is a predefined kernel, for example a potential function [Aizerman 1964] or any *radial basis function* (RBF).

A perceptron classifier admits a dual kernel representation if its weight vector is a linear combination of the training patterns in φ-space:

$$\mathbf{w} = \sum_{k=1}^{p} \alpha_k \varphi(\mathbf{x}_k) \ . \tag{4.4.9}$$

Reciprocally, a kernel classifier admits a dual perceptron representation if the kernel function possesses a finite (or infinite) expansion of the form:

$$K(\mathbf{x}, \mathbf{x}') = \sum_i \varphi_i(\mathbf{x}) \, \varphi_i(\mathbf{x}') \ . \tag{4.4.10}$$

Such is the case for instance for some symmetric kernels. Examples of kernels that we have been using include

$$\begin{array}{lcll} K(\mathbf{x}, \mathbf{x}') & = & (\mathbf{x} \cdot \mathbf{x}' + 1)^q & \text{(polynomial of order q)}, \\ K(\mathbf{x}, \mathbf{x}') & = & \tanh\left(\gamma \, \mathbf{x} \cdot \mathbf{x}'\right) & \text{(neural units)}, \\ K(\mathbf{x}, \mathbf{x}') & = & \exp\left(\gamma \, \mathbf{x} \cdot \mathbf{x}'\right) \ - \ 1 & \text{(exponential)}, \\ K(\mathbf{x}, \mathbf{x}') & = & \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / \gamma\right) & \text{(gaussian RBF)}, \\ K(\mathbf{x}, \mathbf{x}') & = & \exp\left(-\|\mathbf{x} - \mathbf{x}'\| / \gamma\right) & \text{(exponential RBF)}, \\ K(\mathbf{x}, \mathbf{x}') & = & (\mathbf{x} \cdot \mathbf{x}' + 1)^q \exp\left(-\|\mathbf{x} - \mathbf{x}'\| / \gamma\right) & \text{(mixed polynomial \& RBF)}. \end{array} \tag{4.4.11}$$

These kernels have positive parameters (the integer q or the real number γ) which can be determined with a *structural risk minimization* or *cross-validation* procedure (see Appendix B).

The OMC algorithm computes the maximum margin solution in the kernel representation. This is crucial for making the computation tractable when training very large VC-dimension classifiers (see Appendix B). Training a classifier in the kernel representation is computationally advantageous when the dimension N of vectors $\mathbf{w}$ (or the VC-dimension $N + 1$) is large compared to the number of parameters α_k, which equals the number of training patterns p. This is always true if the kernel function possesses an infinite expansion (4.4.10). Our experimental results [Boser 1992, Guyon 1993] indicate that this argument holds in practice even for low order polynomial expansions when the dimension n of the input space is sufficiently large.

Minimizing the Maximum Loss

The margin, defined as the Euclidean distance between the decision boundary and the closest training patterns in φ-space can be computed as

$$M = \min_k \frac{g_k D(\mathbf{x}_k)}{\|\mathbf{w}\|} \ . \tag{4.4.12}$$

The goal of the maximum margin training algorithm is to find the decision function $D(\mathbf{x})$ which maximizes M, that is the solution of the optimization problem

$$\max_{\mathbf{w}} \min_{k} \frac{g_k D(\mathbf{x}_k)}{\|\mathbf{w}\|} \ . \tag{4.4.13}$$

In the classical framework of loss minimization, Problem 4.4.13 is equivalent to minimizing (over $\mathbf{w}$) the maximum loss. The loss function is defined as $\ell(\mathbf{x}_k) = -g_k D(\mathbf{x}_k)/\|\mathbf{w}\|$.

One of the benefits of minimax algorithms is that the solution is a function only of a restricted number of training patterns, namely the informative patterns. This results in high computational efficiency in those cases when the number m of informative patterns is small compared to both the total number of training patterns p and the dimension N of φ-space.

Formulation as a Quadratic Programming Problem

The Problem 4.4.13 is reformulated in the dual α-space as the quadratic programming problem of maximizing the cost function [Vapnik 1982]:

$$J(\alpha, b) = \sum_{k=1}^{p} \alpha_k \, (1 - b g_k) - \frac{1}{2} \alpha \cdot H \cdot \alpha, \tag{4.4.14}$$

under the constrains:

$$\begin{cases} \alpha_k \geq 0 \\ \sum_k g_k \alpha_k = 0 \ . \end{cases} \tag{4.4.15}$$

The $p \times p$ square matrix H has elements $H_{kl} = g_k g_l K(\mathbf{x}_k, \mathbf{x}_l)$, where $K(\mathbf{x}, \mathbf{x}')$ is a kernel, such as the ones proposed in (4.4.11), which can be expanded as in (4.4.10). A unique solution exists if H is positive semi-definite.

The quadratic programming problem thus defined can be solved efficiently by standard numerical methods [Luenberger 1984]. Numerical computation can be further reduced by processing iteratively small chunks of data [Vapnik 1982]. The computational time is linear the dimension n of $\mathbf{x}$-space (not the dimension N of φ-space) and in the number p of training examples and polynomial in the number $m < \min(N + 1, p)$ of informative patterns. It can be theoretically proven that it is a polynomial in m of order lower than 10, but experimentally an order 2 is observed.

Properties of the OMC solution

This formulation of the problem in dual space is obtained with the formalism of Lagrange multipliers and using the scaling constraint that the margin is at a distance $1/\|\mathbf{w}\|$ of

the decision boundary. This imposes:

$$\begin{cases} g_k D(\mathbf{x}_k) = 1 \quad \text{for all informative patterns,} \\ g_k D(\mathbf{x}_k) > 1 \text{ and } \alpha_k^* = 0 \quad \text{for the other training patterns,} \end{cases} \tag{4.4.16}$$

where α_k are the Lagrange multipliers.

Only the informative patterns appear in the solution with non-zero weight α_k^*:

$$\begin{aligned} D(\mathbf{x}) &= \sum_k g_k \alpha_k^* K(\mathbf{x}_k, \mathbf{x}) + b^*, \qquad \alpha_k^* \geq 0, \\ &= \mathbf{w}^* \cdot \varphi(\mathbf{x}) + b, \\ \mathbf{w}^* &= \sum_k g_k \alpha_k^* \varphi(x_k), \\ b^* &= -\frac{1}{2} \left(\mathbf{w}^* \cdot \varphi(\mathbf{x}_A) + \mathbf{w}^* \cdot \varphi(\mathbf{x}_B) \right), \end{aligned}$$

where $\mathbf{x}_A$ and $\mathbf{x}_B$ are two arbitrary informative patterns of class A and B. Using the kernel representation, with a factorized kernel (such as 4.4.11), the classification time is linear in n (not N) and in m (not p).

Using (4.4.16), (4.4.14) and (4.4.15), it can be shown that the value of the cost function at the solution is:

$$J(\alpha^*, b^*) = \frac{1}{2} \|\mathbf{w}\|^2 = \frac{1}{2} \sum_k \alpha_k^* \tag{4.4.17}$$

In the text, we use as *cumulative information criterion* $I = 2\ J(\alpha^*, b^*)$.

Another property of the OMC algorithm is that its prediction error on a test set is bounded by m/p [Boser 1992]. Therefore, maximum margin solutions obtain good generalization even when the problem is grossly underdetermined, i.e. the number of training patterns p is much smaller than the number of adjustable parameters, $N + 1$.

Appendix 4.B: Vapnik-Chervonenkis Dimension

In this appendix, we review the notion of Vapnik-Chervonenkis Dimension or VC-dimension [Vapnik 1982].

Capacity and Guaranteed Risk

A common way of training a given classifier is to adjust the parameters $\mathbf{w}$ in the classification function $D(\mathbf{x}, \mathbf{w})$ to minimize the *training error* E_{train}, i.e., the frequency of errors on a set of p training examples. E_{train} estimates the expected risk based on the empirical data provided by the p available examples. The method is thus called *empirical*

risk minimization. But the classification function $D(\mathbf{x}, \mathbf{w}^*)$ which minimizes the empirical risk does not necessarily minimize the *generalization error*, i.e., the expected value of the risk over the full distribution of possible inputs and their corresponding outputs. Such generalization error E_{gen} cannot in general be computed, but it can be estimated on a separate test set (E_{test}).

Any family of classification functions $\{D(\mathbf{x}, \mathbf{w})\}$ can be characterized by its capacity. The Vapnik-Chervonenkis dimension (or VC-dimension) [Vapnik 1982] is such a capacity, defined as the maximum number d of training examples which can be learned without error, *for all possible binary labelings*. In some simple cases, the VC-dimension is given by the number of free parameters of the classifier (e.g., for classifiers linear in their parameters, the VC-dimension is $N + 1$, where N is the dimension of φ-space, the space spanned by the $\varphi(\mathbf{x})$ vectors, see Appendix A). In most cases it is quite difficult to determine it analytically.

The VC-theory provides bounds. Let $\{D(\mathbf{x}, \mathbf{w})\}$ be a set of classification functions of capacity d. For p training examples that are independently and identically distributed, $p > d$, with probability $(1 - \eta)$, simultaneously for all classification functions $D(\mathbf{x}, \mathbf{w})$, the generalization error E_{gen} is lower than a *guaranteed risk* defined by:

$$E_{guarant} = E_{train} + \beta\ \varepsilon(p, d, E_{train}, \eta)\ , \tag{4.4.18}$$

where $\varepsilon(p, d, E_{train}, \eta)$ is equal to:

$$\varepsilon_0 = \frac{d(\ln\frac{2p}{d} + 1) - \ln\eta}{p} \tag{4.4.19}$$

for small E_{train}, and equal to $\sqrt{\varepsilon_0}$ for E_{train} close to one [Vapnik 1982, Le Cun 1994]. The smaller ε, the closer the *guaranteed risk* is to the training error, therefore the more we can trust the method of *empirical risk minimization*. The first term in ε roughly behaves like $(d/p)^\alpha$, where $1/2 < \alpha < 1$. It says that generalization is likely to be better if $p \geq d$. The second term, $\ln\eta/p$, says that the surer we want to be of the inequality $E_{gen} \leq E_{guarant}$, the looser the bound. This second term is usually ignored in the experiments.

For a fixed number of training examples p, the training error decreases monotonically as the capacity d increases, while the ε term increases monotonically. As a result, the *guaranteed risk* goes through a minimum which usually coincides with a minimum in the generalization error. Before the minimum, the problem is *overdetermined*: the capacity is too small for the amount of training data. Beyond the minimum the problem is *underdetermined*. The key issue is therefore to match the capacity of the classifier to the amount of training data in order to get best generalization performance. The method

of *structural risk minimization* (SRM) [Vapnik 1982, Le Cun 1994] provides a way of achieving this goal.

Let us choose a family of classifiers $\{D(\mathbf{x}, \mathbf{w})\}$, and define a structure consisting of nested subsets of elements of the family: $S_1 \subset S_2 \subset ... \subset S_r \subset ....$ By defining such a structure, we ensure that the capacity d_r of the subset of classifiers S_r is less than d_{r+1} of subset S_{r+1}. The method of SRM amounts to finding the subset S^{opt} for which the classifier $D(\mathbf{x}, \mathbf{w}^*)$ which minimizes the empirical risk within such subset yields the best overall generalization performance.

Two problems arise in implementing SRM: (i) How to find a good structure? (ii) How to select S^{opt}?

A good structure reflects the *a priori* knowledge of the designer, and only a few guidelines can be provided from the theory to solve problem (i). The designer must find the best compromise between two competing terms: E_{train} and ε. Reducing d causes ε to decrease, but E_{train} to increase. A good structure should be such that decreasing the VC-dimension happens at the expense of the smallest possible increase in training error. Examples of structures include varying the number of hidden units in a neural network and varying the order of a polynomial in a polynomial classifier.

Problem (ii) arises because we have no direct access to E_{gen}. Several methods can be used, including *guaranteed risk minimization* (GRM) which consists of using $E_{guarant}$ instead of E_{gen}, *cross-validation* which consists of using the error $E_{validation}$ calculated on a validation set, distinct from the training set and the test set and the *leave-one-out* method which consists of calculating $E_{validation}$ by averaging the results of rotating through the training set, training on all patterns but one, and testing on that pattern.

Removing patterns from the training data, as performed during data cleaning, is a special case of SRM. The number s of suspicious patterns removed establish a structure. The expression of the *guaranteed risk* is slightly different for this particular problem because the pattern selection process adds degrees of freedom:

$$E_{guarant} = E_{train}(s) + \beta \, \frac{d(\ln\frac{2(p-s)}{d} + 1) + s(\ln\frac{p}{s} + 1) - \ln\eta}{p - s} \tag{4.4.20}$$

It is similar to equation 4.4.18 (in the small training error case), where p has been replaced by $p - s$, except for the additional term $s(\ln\frac{p}{s} + 1)/(p - s)$, which shows that number of patterns removed s behaves like an additional capacity.

For the *optimal margin classifier* the GRM method does not apply because the training error is always zero when a solution exists. The method of *cross-validation* is then used to determine the point of optimum cleaning.

Effective VC-dimension

Using SRM with *cross-validation* does not require knowing the VC-dimension and is therefore the simplest and most commonly used method.

Using SRM with GRM is presumably better because it does not rely on reserving patterns for the validation set, but it is more involved. First, the constant β predicted by the theory is always too pessimistic and it must be empirically determined. Second, except in very trivial case, the VC-dimension d is not known analytically. Moreover, the VC-dimension *per se* is not a very good measure of capacity because it does not take into account the properties of the loss function and the training algorithm. It is prefereable to substitute it an empirical value, the *effective* VC-dimension, which is generally smaller than the VC-dimension predicted analytically.

The functional form of the *guaranteed risk* provides nevertheless useful theoretical insights to predict the behavior of the generalization error. Several methods have been proposed to determine empirical values of d and β [Le Cun 1994]. We recommend the following simple protocol:

- Choose a classification task defined by a training and validation set.
- Choose a classifier $D(\mathbf{x}, \mathbf{w})$.
- Calculate the training error E_{train} and the validation error $E_{validation}$ of the classifier for various number of training examples p.
- Determine the unknowns β and the *effective* VC-dimension d by fitting the expression of $[E_{guarant} - E_{train}]$ (ignoring the $\ln\eta$ term) to the experimental points given by $[E_{validation} - E_{train}]$.

More accurate estimates are obtained by splitting the data into a training set and test set in different ways and averaging the results.

The GRM method is advantageous compared to *cross-validation* mainly if β and d are invariant with respect to the classification task. Otherwise, the same data needs to be used to train, to determine β and d and to compute $E_{guarant}$. The only benefit is then that the functional form of $E_{guarant}$ is used to smooth $E_{validation}$.

Confidence in the invariance of β and d with respect to the classification task is built by repeating the experiment on various data sets. Our experiments indicate that for the neural networks that we use, trained with the backpropagation algorithm with small weight random initialization, but no weight decay, we have $\beta \simeq 0.5$ and $d \simeq f/3$, where f is the number of free parameters [Matić 1992].

Bibliography

[Aizerman 1964] Aizerman, M., Braverman, E., and Rozonoer, L. 1964. Theoretical foundations of the potential function method in pattern recognition learning. *Automation and Remote Control*, 25 (6):821–837.

[Boser 1992] Boser, B., Guyon, I., and Vapnik, V. 1992. A training algorithm for optimal margin classifiers. In *Fifth Annual Workshop on Computational Learning Theory*, pages 144–152, Baltimore, MD: ACM Press.

[Cortes 1993] Cortes, C. and Vapnik, V. 1993. The soft margin classifier. Technical memorandum 11359-931209-18TM, AT&T Bell Labs, Holmdel, NJ.

[Demyanov 1972] Demyanov, V. F. and Malozemov, V. N. 1972. *Introduction to Minimax*. New York, NY: Dover.

[Duda 1973] Duda, R. and Hart, P. 1973. *Pattern Classification And Scene Analysis*. New York: Wiley and Son.

[Guyon 1992] Guyon, I., Henderson, D., Albrecht, P., Le Cun, Y., and Denker, J. 1992. Writer independent and writer adaptive neural network for on-line character recognition. In Impedovo, S., editor, *From pixels to features III*, pages 493–506, Amsterdam: Elsevier.

[Guyon 1993] Guyon, I., Boser, B., and Vapnik, V. 1992. Automatic capacity tuning of very large VC dimension classifiers. In S. Hanson et al., editors, *Advances in Neural Information Processing Systems 5*, pages 147–155, San Francisco, CA: Morgan Kaufmann.

[Haussler 1991] Haussler, D., Kearns, M., and Shapire, R. 1991. Bounds on the sample complexity of Bayesian learning using information theory and the VC dimension. In *Fourth Annual Workshop on Computational Learning Theory*, Baltimore, MD: ACM Press. Also *Machine Learning*, volume 14 (1):83–113.

[Le Cun 1994] Le Cun, Y., Levin, E., and Vapnik, V. 1994. Measuring the VC-dimension of a learning machine. *Neural Computation*, 6 (5):851–884.

[Luenberger 1984] Luenberger, D. 1984. *Linear and Non-linear Programming*. Reading, MA: Addison Wesley.

[MacKay 1992] MacKay, D. 1992. Information-based objective functions for active data selection. *Neural Computation*, 4 (4):590–604.

[Matić 1992] Matić, N., Guyon, I., Bottou, L., Denker, J., and Vapnik, V. 1992. Computer aided cleaning of large databases for character recognition. In *11th International Conference on Pattern Recognition*, volume II, pages 330–333, Los Alamitos, CA: IEEE Computer Society Press.

[Matić 1993] Matić, N., Guyon, I., Denker, J., and Vapnik, V. 1993. Writer adaptation for on-line handwritten character recognition. In *Second International Conference on Pattern Recognition and Document Analysis*, pages 187–191, Los Alamitos, CA: IEEE Computer Society Press.

[Rissanen 1989] Rissanen, J. 1989. *Stochastic Complexity in Statistical Inquiry*. Singapore: World Scientific.

[Vapnik 1982] Vapnik, V. 1982. *Estimation of dependences based on empirical data*. New York: Springer.