Max–Planck–Institut für biologische Kybernetik
Max Planck Institute for Biological Cybernetics

Technical Report No. TR-155

Minimal Logical Constraint Covering Sets

Fabian Sinz,[1] Bernhard Schölkopf[1]

December 2006

[1] Department Schölkopf, email: fabian.sinz@tuebingen.mpg.de

Minimal Logical Constraint Covering Sets

Fabian Sinz, Bernhard Schölkopf

Abstract. We propose a general framework for computing minimal set covers under class of certain logical constraints. The underlying idea is to transform the problem into a mathematical programm under linear constraints. In this sense it can be seen as a natural extension of the vector quantization algorithm proposed by [5]. We show which class of logical constraints can be cast and relaxed into linear constraints and give an algorithm for the transformation.

We provide two examples and two applications show the practical relevance of our approach.

1 Introduction

Vector quantization is a well known algorithm for a variety of applications like density estimation and compression. The idea is to represent a set of vectors by a smaller set of so called codebook vectors that cover the input domain with respect to the distribution of the data points. In most cases the resulting codebook is then used for further tasks. In compression, for instance, the codebook is used to encode new data points by using the index of the closest codebook vector instead of the data point itself. Since an index can be encoded much more efficiently than an entire data vector, this compression method yields high compression rates. Nevertheless, this compression scheme is afflicted with loss since the example and the representing codebook vector will coincide very rarely.

In standard vector quantization the number of codebook vectors is fixed in advance. The optimization step places them optimally in the domain such that the distribution of the codebook reflects the distribution of the data. For some training points, especially for those lying in a low density region, the distance to the closest codebook vector might be very large. This normally results in a high compression loss for those data points. Tipping and Schölkopf [5] use a a different approach by computing a cover of the training set using training vectors as centers for the covering balls. The maximal distance ε of a training example to a codebook vector is fixed in advance while the size of the codebook is determined at the training stage. Since they are using training examples as codebook vectors the optimization problem is selecting a minimal subset from the training set such that no example has a distance more than ε to one of the codebook vectors. They relax this combinatorial problem into the following linear program:

$$\texttt{minimize}: \quad ||\boldsymbol{w}||_1 \tag{1}$$

$$\texttt{s.t.} \quad \sum_{i=1}^{m} w_i \, I_{\leq\varepsilon}(x_i, x_j) \geq 1 \;\; \texttt{for all} \;\; x_j \in \mathcal{X}, \tag{2}$$

whereas $I_{\leq\varepsilon}(x_i, \cdot)$ is the indicator function for the ball around x_i with radius ε. Intuitively, this objective minimizes the total number of nonzero entries of $\boldsymbol{w}$ while enforcing that every example lies in at least one of the ε-balls around the codebook vectors. Since 1-norm minimization does not directly minimize the sparseness of the solution, (1) leads to redundant (i.e. not minimally sized) codebook sets. Tipping and Schölkopf overcome this problem by a pruning step after the optimization. Another possibility to directly get sparse solutions is the approach of [6]. The authors approximate the 0-norm (i.e. the number of nonzero entries) minimization by the following optimization problem:

$$\texttt{minimize} \quad \sum_{j=1}^{n} \ln(\varepsilon + |w_j|) \tag{3}$$

$$\texttt{s.t.} \quad \sum_{i=1}^{m} w_i \, I_{\leq\varepsilon}(x_i, x_j) \geq 1 \;\; \texttt{for all} \;\; x_j \in \mathcal{X}. \tag{4}$$

The program (3) can be minimized by solving a series of linear programs of the same form as (1).

In this paper we consider a natural generalization of the idea of Tipping and Schölkopf [5]. Given a binary logical predicate $\Phi(\cdot,\cdot)$ defined on a finite set $\mathcal{X}_1$ and the power set $\wp(\mathcal{X}_2)$ of another finite set which is true for $\mathcal{X}_1$ and at least one element of $\tilde{\mathcal{X}} = \wp(\mathcal{X}_2)$, i.e. $\Phi(\mathcal{X}_1, \tilde{\mathcal{X}}) = \top$, find a minimal subset $\mathcal{X}^* \in \wp(\mathcal{X}_2)$ such that $\Phi(\mathcal{X}_1, \mathcal{X}^*)$ is still fulfilled. Intuitively, Φ is the covering condition which is true if $\tilde{\mathcal{X}}$ covers $\mathcal{X}_1$ and false otherwise. Therefore it describes the feasible set on $\wp(\mathcal{X}_2)$. Under certain conditions that will be specified in section 2 this problem can be relaxed into an optimization problem with linear integer constraints. The idea is then to approximate the linear integer constraints by standard linear constraints and solve a standard optimization problem. The only nonlinearity that might occur in the optimization problem is the function controlling the minimality of the covering set $\mathcal{X}^*$. If this function is linear or can be approximated by a series of linear programs then the problem can be solved by programs similar to those in [5, 6].

The paper is structured as follows. In section 2 we state the conditions that are necessary to formulate the described problem as an ILP and state the algorithm for solving it. In section 3 we give a few examples for choosing Φ. In particular, we show that the vector quantization of [5] is a special case of our algorithm. We also derive two supervised variations of this VQ. In section 4 we report experiments and conclude in section 5.

2 Algorithm

The goal is to transform the combinatorial problem of finding a minimal set $\tilde{\mathcal{X}} \in \wp(\mathcal{X}_2)$ into an optimization problem of the form:

$$\texttt{minimize} \quad g(\boldsymbol{w}) \tag{5}$$

$$\texttt{s.t.} \quad \boldsymbol{A}\boldsymbol{w} \geq \boldsymbol{b} \tag{6}$$

such that the set $\tilde{\mathcal{X}}$ corresponding to a feasible solution $\tilde{\boldsymbol{w}}$ of (5) fulfills $\Phi(\mathcal{X}_1, \cdot)$ and vice versa. $g(\boldsymbol{w})$ is the function controlling the minimality of $|\tilde{\mathcal{X}}|$ by controlling the sparseness of $\boldsymbol{w}$. After optimization every element of $\mathcal{X}_2$ with a nonzero corresponding entry in $\tilde{\boldsymbol{w}}$ will belong to $\tilde{\mathcal{X}}$. The inequality constraints assure that the solution $\tilde{\mathcal{X}}$ still fulfills Φ.

In order to be able to represent a logical predicate as linear constraints in an optimization problem like (5) we first need to transform the predicate into an arithmetic formula. This kind of transformation is well known and used for example in complexity theory to reduce 3SAT to ILP or to show that IP=PSPACE. For now, assume that Φ is complex, i.e. it is a logical formula composed of several atoms ϕ, quantors $\forall$ and $\exists$ as well as the connectives $\wedge$ and $\vee$. Furthermore, assume that the negation $\neg$ only occurs in front of the atoms. This is no restriction since this can always be accomplished by the application of DeMorgan's laws. Since we only deal with the finite case, $\forall$ and $\exists$ can be replaced by a finite conjunctions or a finite disjuntion respectively. "$\forall x \in \mathcal{X}$" transforms into "$\bigwedge_{x\in\mathcal{X}}$" and "$\exists x \in \mathcal{X}$" transforms into "$\bigvee_{x\in\mathcal{X}}$". Now the remaining logical formula is transformed into an arithmetic formula by replacing "$\wedge$" by "$\cdot$", "$\vee$" by "$+$", true atoms by a positive integers and false atoms by zero. This arithmetic formula is further relaxed into a linear constraint over $\mathbb{R}^n$ by allowing real numbers instead of integers.

Let us now turn to the question which predicates Φ are accessible to be transformed into a sets of linear constraints. Intuitively, it is those predicates that can be factored into set of smaller predicates that each depend only on one variable of $\mathcal{X}_2$. By this means it is possible to establish a one to one correspondence between a single entry of $\boldsymbol{w}$ and a element of $\mathcal{X}_2$. The following lemma makes this notion precise.

Lemma *Let Φ be a binary predicate defined a finite set $\mathcal{X}_1$ and the power set $\wp(\mathcal{X}_2)$ of $\mathcal{X}_2$. Furthermore, let Φ be satisfiable for $\mathcal{X}_1$ and at least one element $\tilde{\mathcal{X}} \in \wp(\mathcal{X}_2)$. If $\Phi(\mathcal{X}_1, \mathcal{X}_2)$ can be written as*

$$\Phi(\mathcal{X}_1, \mathcal{X}_2) = \bigodot_{i=1}^{n} \bigodot_{x_2 \in \mathcal{X}_2} \phi_i(\mathcal{X}_1, x_2) \tag{7}$$

where each $\odot$ denotes an arbitrary connective from $\{\vee, \wedge\}$, then $\Phi(\mathcal{X}_1, \cdot)$ can be relaxed into a set of linear inequalities.

Proof: Given $\Phi(\mathcal{X}_1, \mathcal{X}_2)$ in the form of (7), the first step is to convert it into conjunctive normal form (CNF), i.e. write it as

$$\Phi(\mathcal{X}_1, \mathcal{X}_2) = \bigwedge_{j=1}^{k} \bigvee_{x_2 \in \mathcal{X}_2} \psi_j(\mathcal{X}_1, x_2)\,, \tag{8}$$

where $\psi_j(\mathcal{X}_1, x_2) = \bigvee_{l \in I_j} \phi_l(\mathcal{X}_1, x_2)$ for appropriate index sets I_j. Intuitively, I_j groups together all ϕ_i in one clause that have the same x_2 as second argument. Writing Φ in the form of (8) puts no restriction on the accessible formulae of (7), since every logical formula can be converted into CNF (cf. [4] or any other textbook on logic). Now, every clause $\bigvee_{x_2 \in \mathcal{X}_2} \psi_j(\mathcal{X}_1, x_2)$ is converted into an arithmetic formula as described above. Since it consists merely of disjunctions, the arithmetic formula will be a sum and the relaxed form can be written as $\boldsymbol{a}_j^\top w$ where $a_{j,x_2} = 1$ if $\psi_j(\mathcal{X}_1, x_2)$ is true and $a_{j,x_2} = 0$ otherwise. To require that $\bigvee_{x_2 \in \mathcal{X}_2} \psi_j(\mathcal{X}_1, x_2)$ is fulfilled is now the same as requiring $a_j^\top \boldsymbol{w} \geq 1$. Converting every clause into a linear constraint in this way yields the required set of linear constraints. The conjunction is enforced by the fact that in a feasible region of an optimization problem all constraints are met. Since we required, that $\Phi(\mathcal{X}_1, \cdot)$ is fulfilled by at least one $\tilde{\mathcal{X}}$, the feasible region is nonempty. $\square$

Extensions The proof of the lemma presents a method to construct a set of linear inequality constraints $\boldsymbol{a}_j^\top \boldsymbol{w} \geq 1$ given a predicate Φ with the required properties. This method admits at least three extensions. First, the predicate Φ does not need to be restricted to a single set $\mathcal{X}_1$. In principle, Φ can be a predicate over the cartesian product of n sets and $\mathcal{X}_2$ as long as it still factors over $\mathcal{X}_2$. The discriminative and the shared VQ described in section 3 are examples for this extension.

Second, expanding Φ into the single ψ_j might lead to constant terms, i.e. terms that do not depend on $\mathcal{X}_2$. When constructing the linear constraints, those terms enter the right hand side of the inequality, leading to a new expression ν_i that might in turn be interpreted as a logical formula that specifies wether this constraint has to be considered ($\nu_i = 1$) or not ($\nu_i = 0$). In case of $\nu_i = 0$ the constraint can be excluded from the optimization. The shared and the discriminative VQ in section 3 are examples for the second extension.

Third, when choosing the size controlling function g to be $g(\boldsymbol{w}) = ||\boldsymbol{w}||_1$ and restricting every coefficient of $\boldsymbol{w}$ to the intervall $0 \leq w_i \leq 1$, ν_i can also be set to a number that specifies how many elements $x_2 \in \mathcal{X}_2$ have to fulfill the ith constraint since there is no other way to fulfill $\boldsymbol{a}_i^\top \boldsymbol{w} \geq \nu_i$ than having at least ν_i coefficients of $\boldsymbol{w}$ greater than zero, since $a_{ij} \in \{0, 1\}$ by construction.

Algorithm 1 : Algorithm for Logical Constrained Set Compression

Input: Two sets $\mathcal{X}_1$, $\mathcal{X}_2$, a function g controlling the minimality of subsets $\tilde{\mathcal{X}} \subseteq X_2$ and a logical predicate Φ describing the feasible set of covering sets for $\mathcal{X}_1$

Returns: A subset $\mathcal{X}^* \subseteq \mathcal{X}_2$ that is minimal according to g but still fulfills $\Phi(\mathcal{X}_1, \cdot)$

Initialization:

- Construct a set $\boldsymbol{A} = (\boldsymbol{a}_1, ..., \boldsymbol{a}_m)^\top$ of linear inequality constraints from Φ
- Initialize the vector $\boldsymbol{\nu}$ specifying the number of elements in $\mathcal{X}_2$ that need to fulfill the constraints in $\boldsymbol{A}$

Optimization: Optimize

$$\begin{aligned} \texttt{minimize}_{\boldsymbol{w}} \quad & g(\boldsymbol{w}) \\ \texttt{s.t.} \quad & \boldsymbol{A}\boldsymbol{w} \geq \nu \\ & \boldsymbol{w} \geq 0 \end{aligned}$$

Return: $\mathcal{X}^* = \{x_i \in \mathcal{X}_2 | w_i \neq 0\}$

Choice of g Putting everything together yields the algorithm shown in algorithm figure 1. The optimization step itself depends on the actual choice of g. As already mentioned in the introduction, possible choices of g are $g(\boldsymbol{w}) =$

$||\boldsymbol{w}||_1$ or the minimal 0-norm approximation $g(\boldsymbol{w}) = \sum_{j=1}^{n} \ln(\varepsilon + |w_j|)$ respectively. The first choice just leads to the optimization of a single linear program, but since it is likely to give redundant, i.e. non-minimal solutions, $g(\boldsymbol{w}) = \sum_{j=1}^{n} \ln(\varepsilon + |w_j|)$ is the preferable choice. The authors of [6] show that in this case the optimization can be accomplished by solving a series of linear programs. Each single optimization step simply consists of a 1-norm minimization of $\boldsymbol{w}$ like in equation (1). After every optimization the columns of $\boldsymbol{A}$ are rescaled by the previous solution. Intuitively, this assigns low weights to unimportant features while upscaling important ones. Algorithm 2 shows the optimization procedure.

Algorithm 2 : Optimization Procedure for $g(\boldsymbol{w}) = \sum_{j=1}^{n} \ln(\varepsilon + |w_j|)$

```
Input: A set of linear inequality constraints specified by $\boldsymbol{A}$ and $\boldsymbol{\nu}$ and a convergence tolerance $\varepsilon$
Returns: A local minimum of $g(\boldsymbol{w}) = \sum_{j=1}^{n} \ln(\varepsilon + |w_j|)$
Initialization: Set $t = 0$, $\boldsymbol{z}_0 \leftarrow (1, ..., 1)$, $\boldsymbol{A}_0 = \boldsymbol{A}$
Optimization:
repeat
    Optimize
        $\texttt{minimize}_{\boldsymbol{w}} \quad ||\boldsymbol{w}||_1$
        $\texttt{s.t.} \quad \boldsymbol{A}\boldsymbol{w} \geq \nu$
        $\boldsymbol{w} \geq 0$
    Let $\boldsymbol{w}_t^*$ be solution at step $t$ and $\boldsymbol{a}_t^i$ the $i$th column of $\boldsymbol{A}_t$
    Set $z_{(t+1)i} \leftarrow z_{ti} \cdot \boldsymbol{w}_{ti}^*$ for all $1 \leq i \leq n$
    Set $\boldsymbol{A}_{t+1} \leftarrow (z_{(t+1)1} \cdot \boldsymbol{a}_t^1, ..., z_{(t+1)n} \cdot \boldsymbol{a}_t^n)$
until $|\boldsymbol{w}_t^* - \boldsymbol{w}_{t-1}^*| < \varepsilon$

Return: $\boldsymbol{w}_t^*$
```

Optimization In most cases using a standard LP optimizer is sufficient. But there might be situations in which the set $\mathcal{X}_2$ is very large, leading to a large number of variables and columns of $\boldsymbol{A}$ to optimize over. But usually the solution $\mathcal{X}^*$ is expected to contain a much smaller number of elements. All other elements of $\mathcal{X}_2$ and their respective columns are irrelevant to the optimization problem since the optimizer is expected to push the corresponding values of $\boldsymbol{w}$ to zero. In situations where solutions of the linear program are expected to be sparse, one can use *column generation* [3, 2] to solve the optimization problem. In column generation the problem (called the *global master problem*) is decomposed into a *restricted master problem* that uses only a subset $(\boldsymbol{a}^i)_{i \in I}$ of the columns of $\boldsymbol{A}$ and entries $(w_i)_{i \in I}$ of $\boldsymbol{w}$ and an *oracle problem* that adds new columns $\boldsymbol{a}^j,\ j \notin I$ to $(\boldsymbol{a}^i)_{i \in I}$ if the restricted master problem was not feasible or did not yield the optimal solution. Column generation makes use of the fact that columns of a primal linear program become constraints in the dual linear program. Solving a primal program with only a subset $(\boldsymbol{a}^i)_{i \in I}$ of the columns of $\boldsymbol{A}$ is the same as solving the dual problem with only a subset of the constraints. If the dual is unbounded or infeasible the primal problem is also unbounded or infeasible. In this case the oracle has to choose an additional column from $\boldsymbol{A}$ and add it to the restricted master problem. The method how to choose this column may vary with the specific application, but a common way is to choose the most violated dual constraint. If the dual restricted master problem has an optimal solution $\boldsymbol{y}^*$ the corresponding primal solution $\boldsymbol{x}^*$ is also optimal. The question is now whether $\boldsymbol{x}^*$ is also optimal for the global master problem. The question can be solved by looking again at the dual solution $\boldsymbol{y}^*$. Assuming $y_j^* = 0$ for $j \notin I$ one can check if $\boldsymbol{y}^*$ meets all constraints of the global master problem. If this is the case then $\boldsymbol{y}^*$ is the optimal solution for the dual global master problem and so is $\boldsymbol{x}^*$ for the primal. If $\boldsymbol{y}^*$ still violates some constraints, the most violated one is added to the column set of the primal problem and optimization is repeated.

All in all, obtaining a set $\mathcal{X}^* \subseteq \mathcal{X}_2$ that still fulfills $\Phi(\mathcal{X}_1, \cdot)$ involves converting $\Phi(\mathcal{X}_1, \mathcal{X}_2)$ into a set of linear constraints and optimizing a function g subject to this set of linear constraints with an appropriate optimization method. In the next section we give a few examples how to apply this method.

Algorithm 3 : Optimization with Column Generation

```
Input: Linear minimization problem: minimize$_{\boldsymbol{x}}$ $\boldsymbol{w}^\top \boldsymbol{x}$ subject to $\boldsymbol{A}\boldsymbol{x} \geq \boldsymbol{\nu},\ \boldsymbol{x} \geq 0$
Returns: A minimal solution $\boldsymbol{x}^*$
Initialization: Select a subset $\tilde{\boldsymbol{A}} := (\boldsymbol{a}^i)_{i \in I}$ of columns from $\boldsymbol{A}$ and coefficients $\tilde{\boldsymbol{w}} := (w_i)_{i \in I}$ from $\boldsymbol{w}$
Optimization:
repeat
    $\texttt{maximize}_{\boldsymbol{y}}\ \boldsymbol{\nu}^\top \boldsymbol{y}$  s.t. $\tilde{\boldsymbol{A}}\boldsymbol{y} \leq \tilde{\boldsymbol{w}}$, $\boldsymbol{y} \geq 0$
    Let $\tilde{\boldsymbol{y}}^*$ be the optimal solution. Set
    $y_i^* = \begin{cases} \tilde{y}_i & \text{if } i \in I \\ 0 & \text{else} \end{cases}$
    if $\min_i(w_i - \boldsymbol{y}^{*\top}\boldsymbol{a}^i) < 0$ then
        $I \leftarrow I \cup \operatorname{argmin}_i(w_i - \boldsymbol{y}^{*\top}\boldsymbol{a}^i)$
    end if
until $\min_i(w_i - \boldsymbol{y}^{*\top}\boldsymbol{a}^i) \geq 0$
Return: Primal solution $\boldsymbol{x}^*$ corresponding to $\boldsymbol{y}^*$
```

3 Examples

The algorithm was originally motivated by the linear programming approach to VQ by [5]. We now state two modifications of the original algorithm and show how they fit into our framework.

Discriminative VQ Assume we are given a labeled set of points $(\boldsymbol{x}_i, y_i) \in \mathbb{R}^r \times \{-1, 1\},\ 1 \leq i \leq m$ and the goal is to represent this set by a set of codebook vectors like in [5]. The only difference to [5] is that we only want to cover those points by codebook vectors that have a distance greater than δ to any point of the opposite class. In some sense those are the discriminative points for both classes. We use d to denote the distance function in order to indicate that $\mathbb{R}^r$ could be any metric space in this example. Let I^+ and I^- denote the indices of the positive and the negative examples, respectively, and $\mathcal{X}_I$ the set of points specified by the index set I. Then one can express the requirements for the points of the positive class in the following logical formula:

$$\begin{aligned}\Phi_+(\mathcal{X}, \mathcal{X}_{I^+}) &= \forall i \in I^+ : [\min_{k \in I^-} d(x_i, x_k) > \delta \rightarrow (\exists j \in I^+ : d(x_i, x_j) \leq \varepsilon)] \\ &= \bigwedge_{i \in I^+} \underbrace{[\neg \min_{k \in I^-} d(x_i, x_k) > \delta \vee \bigvee_{j \in I^+} d(x_i, x_j) \leq \varepsilon]}_{=:\phi_i}\end{aligned}$$

This formula can now be relaxed to a set of linear constraints, where each single term ϕ_i becomes a linear constraint. Let $\varrho[\phi] = 1$ if ϕ is true and $\varrho[\phi] = 0$ else, then

$$\begin{aligned}\phi_i &= \neg\left(\min_{k \in I^-} d(x_i, x_k) > \delta\right) \vee \bigvee_{j \in I^+} (d(x_i, x_j) \leq \varepsilon) \\ &\Rightarrow \quad \varrho[\min_{k \in I^-} d(x_i, x_k) \leq \delta] + \left(\sum_{j \in I^+} w_j \varrho[d(x_i, x_j) \leq \varepsilon]\right) \geq 1 \\ &\Leftrightarrow \quad \sum_{j \in I^+} w_j \underbrace{\varrho[d(x_i, x_j) \leq \varepsilon]}_{=a_i^j} \geq \underbrace{1 - \varrho[\min_{k \in I^-} d(x_i, x_k) \leq \delta]}_{=\nu_i}\end{aligned}$$

Since the constraints of the negative class can be transformed exactly the same way, running a linear program with the constraints for $\Phi_+(\mathcal{X}, \mathcal{X}_{I^+})$ and $\Phi_-(\mathcal{X}, \mathcal{X}_{I^-})$ is the same as relaxing the formula $\Phi(\mathcal{X}, \mathcal{X}) = \Phi_+(\mathcal{X}, \mathcal{X}_{I^+}) \wedge \Phi_-(\mathcal{X}, \mathcal{X}_{I^-})$ into linear constraints.

Looking at the relaxed linear constraint we can see that it enforces exactly what we wanted to have. In order to get $\sum_{j \in I^+} w_j \varrho[d(x_i, x_j) \leq \varepsilon] \geq 1$ at least one coefficient w_j corresponding to a x_j for which $\varrho[d(x_i, x_j) \leq \varepsilon] = 1$ must be sufficiently positive. This x_j will be the codebook vector for x_i. If, however, $\varrho[\min_{k \in I^-} d(x_i, x_k) \leq \delta] = 1$, which corresponds to $\nu_i = 0$, then the constraint just needs to be positive which is trivially true, since it merely consists of positive terms. Thus this constraint can be left out, which means that a x_i for which $\varrho[\min_{k \in I^-} d(x_i, x_k) \leq \delta]$ holds true will not be represented by a codebook vector. But that was exactly our goal since the distance from x_i to a point of the other class is less than δ. The corresponding linear programm has a feasible solution independent of the choice of ε and δ since every vector can always be its own codebook vector.

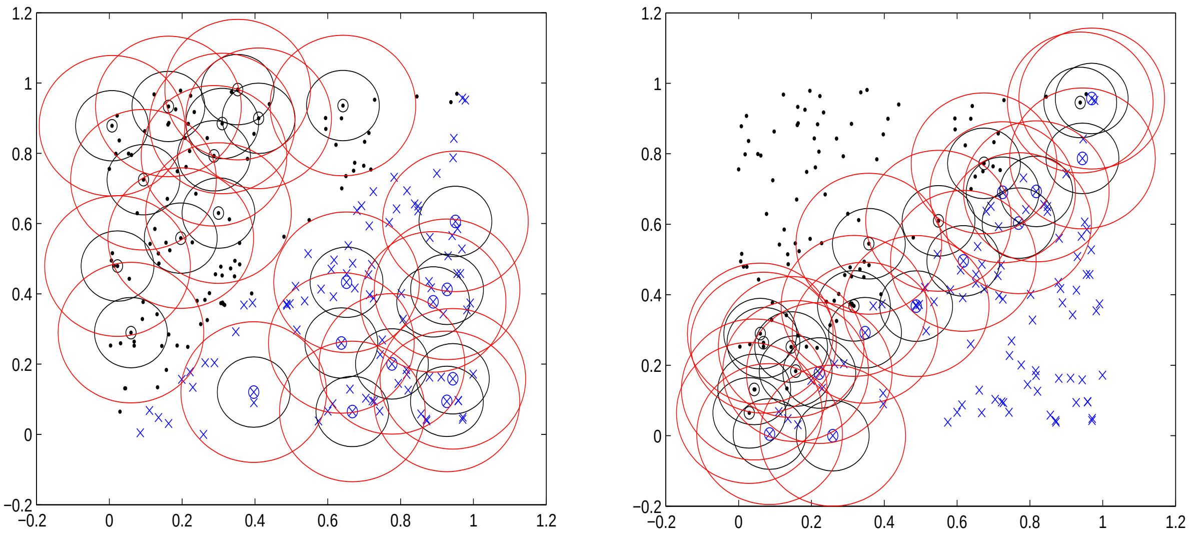

Figure 1: Discriminative (left) and Shared VQ (right) with $\varepsilon = 0.1$ (depicted by the black circles) and $\delta = 0.2$ (depicted by the blue circles) on a toy example, where each point $\boldsymbol{x}$ from 200 uniformly distributed examples in $[0, 1] \times [0, 1]$ was assigned the label $y = 1$ if $x_1 > x_2$ and $y = -1$ otherwise. A small circle around a point indicates that this point is an element of the codebook.

Shared VQ The shared VQ is in some sense the exact opposite of the discriminative VQ. Instead of not covering examples that are closer than δ to any example of the other class, a point now does not need to be represented by a codebook vector if the distance to any point of the other class is larger than δ. This simply means turning "$\min_{k \in I^-} d(x_i, x_k) \leq \delta$" to turn it into "$\min_{k \in I^-} d(x_i, x_k) \geq \delta$". Figure 1 shows the effect of discriminative and shared VQ on a toy example.

4 Experiments

Automatic Summary Extraction Since the establishment of the internet as a common information source, the number of digitally available text has increased enormously and the search for relevant information has become an issue. In this experiment we address the problem of extracting a subset of sentences from a given text that summarize the text well according to a chosen measure. Imagine a database of texts like newspaper arcticle or other text information. Searching the whole database for a certain query posed by a user might be time consuming due to the mere amount of text. An alternative search strategy would be to search a set of summaries for the user's query instead. Everytime a new text is entered into the database, also its summary is computed and stored. Since the procedure of summary extraction is supposed to leave out irrelevant information and yields much shorter texts, searching for relevant texts can be carried out more robustly and efficiently.

We characterize a summarizing set of sentences $\mathcal{X}_s$ by the condition that each sentence x in the orginal text has a sentence $x_s \in \mathcal{X}_s$ that summarizes it well, i.e.

$$\forall x \in \mathcal{X}\ \exists x_s \in \mathcal{X}_s : \Phi(x, x_s).$$

The features which x_s must have in order to summarize x are all put into $\Phi(x, \cdot)$, which can involve arbitrarily complex features of x and x_s. In this experiment we used the paragraph distance d_p, i.e. the number of paragraphs between two sentences, and the L-ROUGE [1] score as features for Φ. Two sentences were considered to summarize each other if they were not more than n paragraphs away from each other and had a L-ROUGE score greater than some threshold σ:

$$\Phi(x, x_s) = (d_p(x, x_s) \leq n) \wedge (L\text{–}ROUGE(x, x_s) \geq \sigma). \tag{9}$$

The L-ROUGE score is one of various measures proposed in [1] to evaluate the quality of a summary according to a reference summary that serves as ground truth. L-ROUGE uses the normalized lengths $r = |lcs(R, P)|/|R|$ and $p = |lcs(R, P)|/|P|$ of the longest common subsequence $|lcs(R, P)|$ between a reference summary R and a proposed summary P to compute an F-measure $F_\beta(R, P) = \frac{(1+\beta^2)rp}{r+\beta^2 p}$. $F_\beta(R, P)$ has values between zero and one, where one indicates a perfect match and zero means that R and P do not share a single word. Because the author of [1] recommends L-ROUGE for short summaries, we use it in our experiments, since we compute the ”summarizing power” between single sentences.

Converting the conditions of (9) into linear constraints in exactly the same manner as shown in section 3 yields

$$\sum_{x_s \in \mathcal{X}} w_s \varrho[\Phi(x, x_s)] \geq 1 \quad \text{for all } x \in \mathcal{X}.$$

Experiments were run on 20 texts from a the German public news website *http://www.tagesschau.de*, using the abstract provided on the website as reference summary. The whole texts were split manually into single sentences. We used $g(w) = \sum_{j=1}^{n} \ln(\varepsilon + |w_j|)$ to control the minimality of the summaries. The parameter β of the L-ROUGE score was set to 1. Before computing the score, all letters were converted to lower case. The algorithm was trained on each text for each parameter pair (n, σ) with $n \in \{1, 2, 3, \infty\}$ and $\sigma \in \{0.1, 0.2, ..., 0.9\}$. We used a validation set of nine texts to find the best set of parameters in terms of L-ROUGE score between the summary proposed by the algorithm and the abstracts offered by the website. Then we used the parameters with best average performance on those nine texts to run the algorithm on the test set.

Figure (2) shows the results on the test texts. Note that selecting a subset of sentences from the original text limits the maximal achievable score, because the maximal score can only be achieved if the exact sentences of the abstract are contained in the full text. This will happen only in very rare cases. To take this into account, we also compute the maximal achievable score, i.e. the maximum score over the power set of sentences, and normalize the test score of our algorithm by it. The normalized scores are also shown in figure (2). Most of the normalized scores have values around 0.5. This means that this simple automatic summary extraction can already capture around 50% of the best score that can be achieved by only considering summaries consisting of sentence subsets from the original text. Since $\Phi(x, x_s)$ can incorporate arbitrarily complex linguistic features, the scores can probably be improved by considering a more sophisticated predicate describing the ”summarization power” of $\mathcal{X}_s$. Note however, that Φ must be defined such that it can be expanded over $\mathcal{X}_s$. Another enhancement would be to allow $\mathcal{X}_s$ to contain additional sentences that are not in the original text, but may be more representative. In fact, $\mathcal{X}_s$ does not even need to be restricted to sentences. It could e.g. already contain structures more suitable for a search engine.

Robust feature selection with object background discrimination In this experiment we consider the following problem. Assume we are given a set $\mathcal{X}$ of image patches from several images of two objects from different views in front of a static background. For each patch we have the information which object is displayed on the image the patch is taken from. Additionally we know if two patches are from the same image or not. The task is now to extract patches from $\mathcal{X}$ that represent robust features to detect the objects in previously unseen images taken from the same setting. Note that it is not known if the patch depicts the object or the background of an image. Therefore the algorithm has to accomplish two tasks at once. On the one hand, it must separate the object patches from the background patches and on the other hand it must make sure that the extracted patches are in some sense close to other object patches of

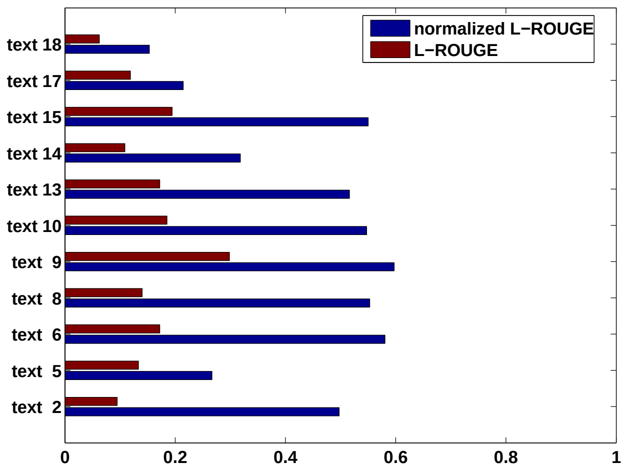

Figure 2: Test results for automatic summary with vector quantization for parameters $\sigma = 0.1$ and $n = \infty$. The two scores shown are the absolute L-ROUGE score and the L-ROUGE score normalized by the maximal achievable score.

the different images of the same object. One way to meet that constraint is to run a discriminative and a shared vector quantization at the same time by requiring that only those patches need to be covered that have distance larger than δ to patches from the image (discriminative) of the other object but are closer than σ to patches from different images of the same object (shared). Since the background is similar in all images, the discriminative part will hopefully exclude the background patches while the shared part ensures sufficient closeness to object patches of the other images. In order to formulate this condition as a logical constraint let $\mathcal{X}_i^+$ be the set of patches from image i of object "+", $\mathcal{X}^+ = \bigcup_i \mathcal{X}_i^+$ and $\mathcal{X}_{I\setminus i}^+ = \bigcup_{j\in I\setminus i} \mathcal{X}_j^+$. Then the constraints for patches of image i depicting object "+" are

$$\forall x^+ \in \mathcal{X}_i^+ : (\min_{x^-\in\mathcal{X}^-} ||x^- - x^+||_2 \geq \delta \wedge \min_{\tilde{x}^+\in\mathcal{X}_{I\setminus i}^+} ||x^+ - \tilde{x}^+||_2 \leq \sigma) \rightarrow \exists x \in \mathcal{X}^+ : ||x^+ - x||_2 \leq \varepsilon.$$

The constraints for the other images and for object "$-$" are defined analogously.

We carried out this experiment on pictures of a toy cow and a toy car from the ETH80 image dataset. For each object we selected all pictures for which the camera is on the same plane as the object and selected six pictures for training, three for model selection and four for testing. In order to find a good parameter setting we selected a set of ten representative values for each of the parameters δ, σ and ε and ran the algorithm on all possible combinations. To get reasonable ranges for the parameters we used the $\frac{n}{20}$-quantile ($1 \leq n \leq 10$) of the empirical distribution of pairwise distances between all patches for δ and σ and set $\varepsilon = \frac{m}{10} \cdot \sigma$ ($1 \leq m \leq 10$) to ensure that ε is always at most σ. The codebook corresponding to the best parameter setting was computed on the training set and used for testing.

Since the algorithm yields three patch classes, "background" for the non-covered patches, "+" for patches covered by vectors from $\mathcal{X}^+$ and "-" for patches covered by vectors from $\mathcal{X}^-$, a simple $0-1$-loss cannot be applied to evaluate the performance. Instead another measure has to be used to access the quality of the performance. In order to capture exactly what we consider a "good" solution, various measures have been used to compute a final score. Here, "positives" means all patches really depicting object "+". Clearly we want the algorithm to cover the true object patches. Therefore we used the *recall* $r^+ = \frac{|\text{true positives}|}{|\text{positives}|}$ on the positive object patches and analogously r^- for the negative object patches. On the other hand the algorithm should not declare everything to be an object patch. Therefore we included the *precisions* $p^+ = \frac{|\text{true positives}|}{|\text{predicted positives}|}$ and p^- for the positive and negative object patches in our final measure. For the same reason the *fall-out* $f^+ = \frac{|\text{false positives}|}{\mathcal{X}\setminus|\text{positives}|}$ should be small. The single scores are simply added, which yields the final score $s = r^+ + p^+ - f^+ + r^- + p^- - f^-$.

Figure 3 shows the results for the highest scoring parameters on the training set. As one can see, the object-background separation works well at the training stage. Naturally there are no incorrect patches sinces the constraints do not allow this.

In the test phase we used the codebook with highest score on the validation set to predict the patch classes on the test set. Figure 4 shows the results on the test set. Clearly, the algorithm fails in terms of object background separation,

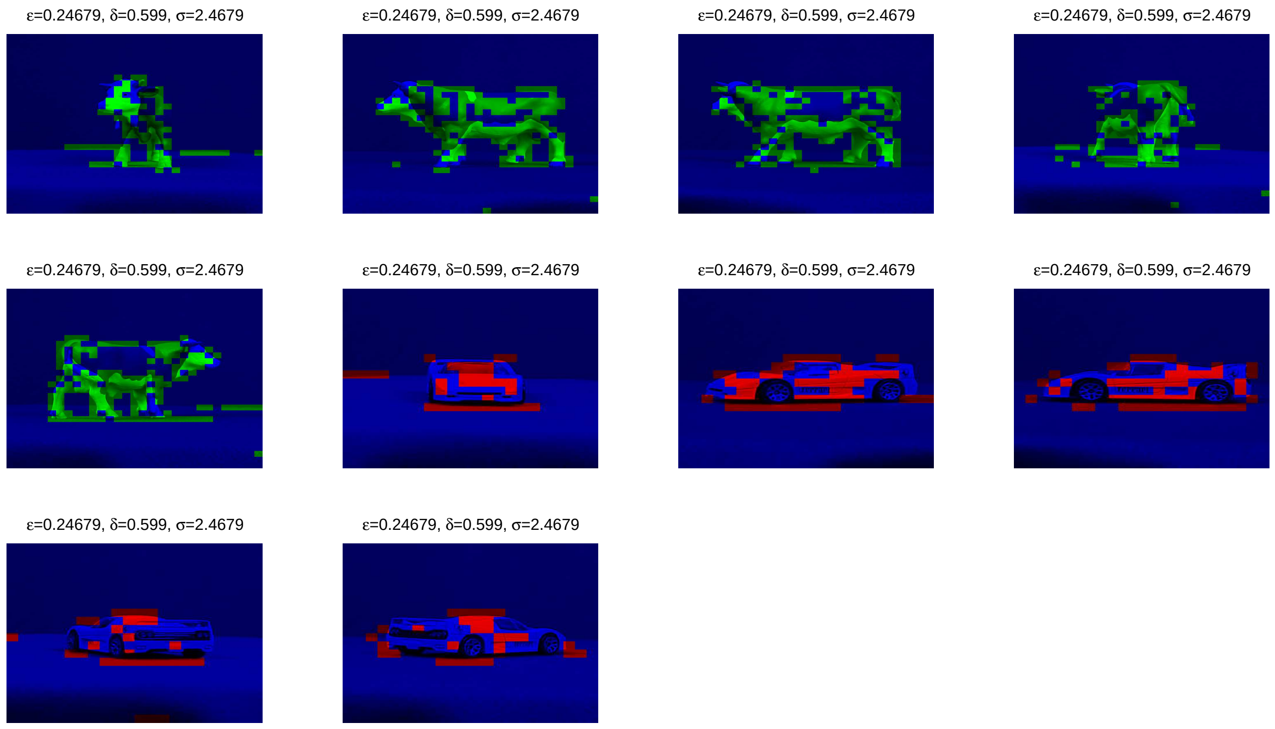
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679
ε=0.24679, δ=0.599, σ=2.4679

Figure 3: Result on the training set for the best set of parameters: Patches that have been assigned to ”car” are colored in red, patches that have been assigned ”cow” are colored in green and patches that are not covered by any codebook vector are colored in blue.

since a lot of cow or car patches are labeled as background, especially for view angles that are very dissimilar to view angles of the training set (i.e. the car and the cow from behind).

This effect is probably due to the high pairwise similarity of all training patches, which leads to a small cover radius ε in order to keep the background patches outside the cover radius since covering them would decrease the score s. However, rotating an object seems to change the structure of most of the object patches in such a way that they are placed outside the cover radius of the codebook obtained on the training set, leading to the observed sparse cover of the test object patches. Another indication for that is the fact that once object patches on the training set are covered, the class assignment seems to be very precise. Seemingly taking the euclidean norm is not the best way to measure similarities between image patches for this kind of problem.

5 Summary and Conclusion

We introduced a natural extension to the vector quantization algorithm by [5] and proposed a general framework to compute covering sets under a certain class of logical constraints. We showed that as long as the logical constraint can be expressed as a compound formula of constraints, each of them only depending on a single element of the covering set, it can be relaxed into linear constraints. The task of finding a minimal set cover can then be accomplished by minimizing a function controlling the size of the cover under these linear constraints.We proposed two functions for that purpose, one being the L1-norm already used by [5] and the other being an approximation to the L0-norm introduced by [6]. We showed two examples how to use this approach for more sophisticated versions of vector quantizations and showed two applications.

One drawback of this approach is that the number of constraints can potentially grow very fast, leading to a large optimization problem. If the solution is expected to be sparse, this problem can be overcome by using column generation to compute a minimal cover. But since column generation is an optimization technique from the domain of linear

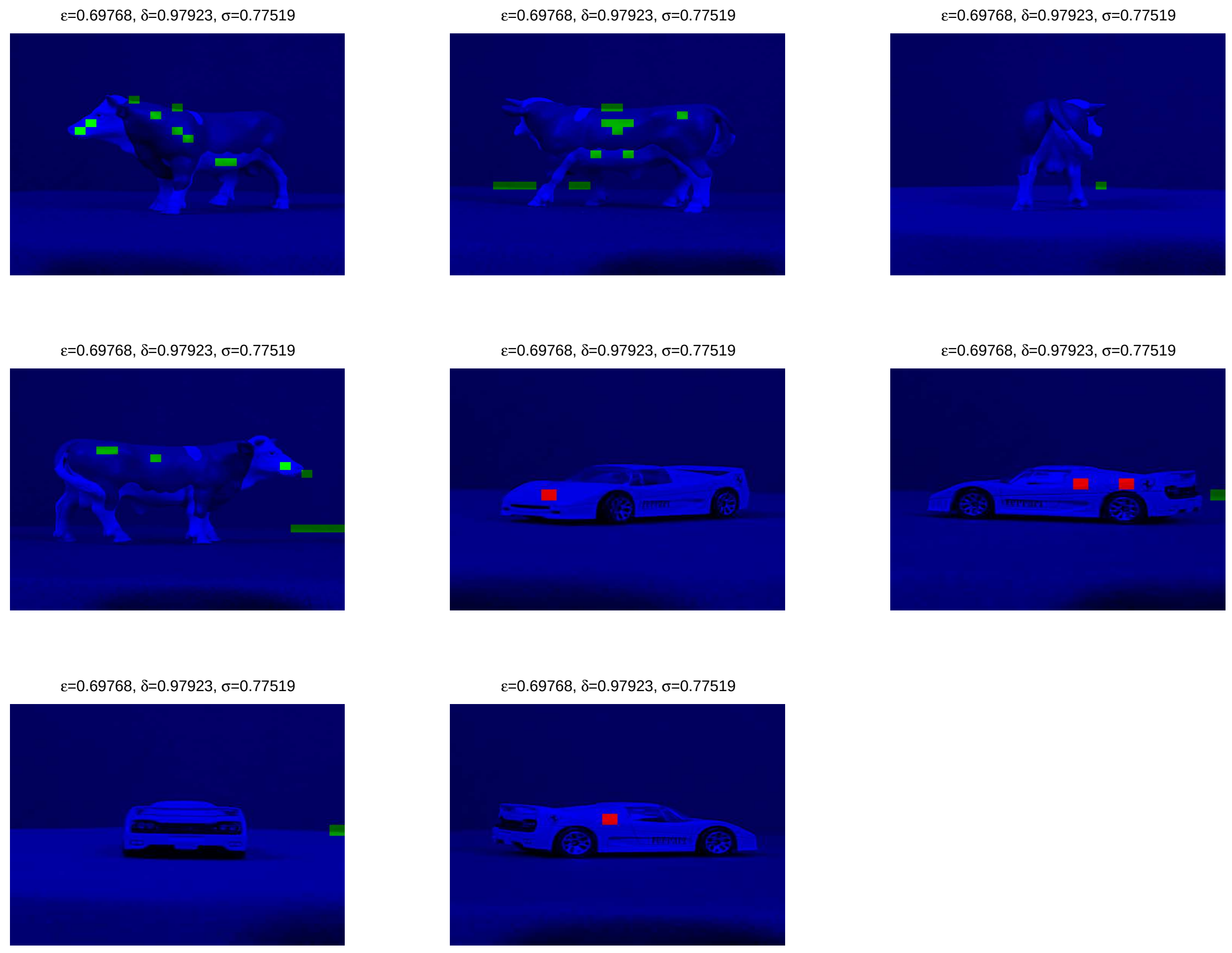
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519
ε=0.69768, δ=0.97923, σ=0.77519

Figure 4: Result on the test set for the best set of parameters on the validation set: Patches that have been assigned to ”car” are colored in red, patches that have been assigned ”cow” are colored in green and patches that are not covered by any codebook vector are colored in blue.

programming, this approach is limited to functions that can be optimized by a linear program. Additionaly, as long as this problem is solved by a linear program, the resulting solution must be approximate[1], since it is possible to reduce NP-complete problems like 3SAT to this kind of linear optimization problem. If we obtained an exact solution, we could solve 3SAT in polynomial time because linear programming is in P. However, in most practical applications it is not essential to obtain global minimum, but only a solution that is reasonable small, especially if computing the real global optimum amounts to solving an NP-hard problem. In this case the proposed framework is a principled and easy way to relax to problem into a real valued optimization problem that can be solved with standard optimization techniques.

Acknowledgements

The authors would like to thanks Frank Jäkel, Stefanie Jegelka and Sebastian Nowozin for useful hints and discussions.

[1]Given that $P \neq NP$

References

[1] *ROUGE: A package for Automatic Evaluation of Summaries*, 2004.

[2] George B. Dantzig. *Linear Programming and Extensions*. Princeton Universtiy Press, 1st edition, 1963.

[3] Lester R. Jr. Ford and Delbert R. Fulkerson. Constructing maximal dynamic flows from static flows. *Operations Research*, 6:419–433, May 1958.

[4] Uwe Schöning. *Logik für Informatiker*. Spektrum Akademischer Verlag, 5 edition, 2000.

[5] M. Tipping and B. Schölkopf. A kernel approach for vector quantization with guaranteed distortion bounds. pages 129–134, San Francisco, CA, 2001. Morgan Kaufmann.

[6] J. Weston, A. Elisseeff, B. Schölkopf, and M. Tipping. Use of the zero-norm with linear models and kernel methods. *Journal of Machine Learning Research*, 3(7-8):1439–1461, 2003. The data, as well as an extended version of the paper, can be obtained from http://www.kyb.tuebingen.mpg.de/bs/people/weston/l0/.