1 Active Constraint Identification

We handle design problems with modeling and optimization. Sometimes this procedure is under question due to the lack of knowledge about the design constraints, especially when human evaluation is involved. In this work we look into the problem of indentifying constraints when they are not physically available and can only be identified by trial.

The solution is indeed an application of active learning. Let $D$ be the design space and a constraint be a manifold $H = 0$ in $D$ such that

\[
\begin{align*}
D^+ \cup D^- &= D, \text{ and } D^+ \cap D^- = \emptyset, \\
D^+ &= \{x \mid x \in D, \ H(x) \geq 0\}, \\
D^- &= \{x \mid x \in D, \ H(x) < 0\}.
\end{align*}
\]

Let $D^+$ be the feasible region in a design problem. The goal of constraint identification is to approximate $D^+$ by sampling and labeling $x \in D$. A naive approach would be to mesh $D$, label all the mesh points and classify this data set. However, such an approach may not be feasible, let alone inefficient, since the required data size is proportional to $m^n$, where $n$ is the dimensionality of $D$ and $m$ is the mesh size. Labeling all the the points will be costly when $n$ is large and human evaluation is involved. Further, the computational cost of a classification routine will be $O(m^{3n})$.

Below we introduce three query algorithms that do not require mesh. The first two algorithms have been reported in active learning researches on discrete and finite $D$ but are now extended to deal with a continuous compact space. We propose the last algorithm combining the benefits of the first two.
2 Algorithms

2.1 Algorithm SIMPLE

An existing solution to derive a classifier from a large set of unlabeled data is the query by committee concept [1] which is adopted in Tong et al. [2], where the algorithm is named SIMPLE. In general, we need a classification and a query scheme. To begin with, we label a sub-set from $U$ and build a classifier. The query scheme then identifies from $U$ the next point to query. The classifier is rebuilt after the label of the new point is obtained, and the procedure continues. Tong et al. proved that the optimal point to query is the one that halves the version space of the current classifier. To elaborate, we first define a version space $V$ as follows:

$$V = \{ w \in \mathcal{F} ||| w ||| = 1, y_i(w \cdot \Phi(x_i)) > 0, \ i = 1,...,n \}, \quad (2)$$

where $\mathcal{F}$ is the feature space, $\Phi(x_i)$ are the sample points in $\mathcal{F}$, or in other words, the version space $V$ is the feasible set of parameter $w$ of the classifier. Geometrically, $V$ is part of a hypersphere in $\mathcal{F}$, where the hypersphere has radius of 1 and is sliced by hyperplanes $\Phi(x_i)$ as visualized in [2].

Since SVM essentially maximize the minimum distance of training data to the decision boundary in the feature space, the optimal solution $w^*$ corresponds to the center of the largest hypersphere not intersecting with the hyperplanes $\Phi(x_i)$. One shall also see that during the query process, the version space shrinks and converges to the real one in finite steps if training data has finite size.

However, the last assumption is not valid in the constraint identification task, i.e., we have a vector space $D$ rather than the finite set $U$, and as mentioned earlier, meshing $D$ will be costly due to its high dimensionality. Tong et al. suggests that one approximation approach to half the version space is to find $\Phi(x)$ closest to $w$, since $w$ is usually close to the center of the version space $^1$. We show below that following this spirit, the optimal query points will lie on the decision boundary. First, as to minimize the distance between the current $w$ and potential $\Phi(x)$, we need to solve:

$$\min_{\Phi(x) \in \mathcal{F}} ||w \cdot \Phi(x)||. \quad (3)$$

Obviously, such $\Phi(x)$ lies on the decision plane. Or, since $w$ is derived from

$^1$This is not guaranteed though.
the previous SVM training, it has a form of
\[ w = \sum_{i=1}^{n} \alpha_i \Phi(x_i), \]  
(4)
and thus (3) becomes
\[ \min_{x \in D} \sum_{i=1}^{n} \alpha_i |K(x, x_i)|, \]  
(5)
meaning that the optimal query points are on the manifold \( \sum_{i=1}^{n} \alpha_i K(x, x_i) = 0 \), which is the decision boundary in \( D \). The physical meaning of this strategy is clear: the most information we gain is by testing most uncertain points, which are on the decision boundary.

The implementation of this algorithm requires simply a Newton-Raphson step to project a random point to the decision boundary.

2.2 Algorithm Kernel Farthest First

Unlike SIMPLE, the kernel farthest first (KFF) algorithm tries to explore the space by maximizing the minimum kernel distance from the current query point to all labeled points. When \( D \) is discrete and finite, the search can be done in \( O(||D||) \) time. If \( D \) is continuous, the query point \( x \) is the solution to the max-min problem:
\[ \max_{x \in D} \min_i ||\Phi(x_i) - \Phi(x)||^2. \]  
(6)
Recall that
\[ ||\Phi(x_i) - \Phi(x)||^2 = ||\Phi(x_i)||^2 + ||\Phi(x)||^2 - 2 < \Phi(x_i), \Phi(x) > \]  
\[ = K(x_i, x_i) + K(x, x) - 2K(x_i, x), \]  
(7)
and assume RBF kernel so that
\[ K(x, x) = 1 \]  
(8)
\[ K(x_i, x) = \exp(-\lambda||x - x_i||^2), \]  
(9)
the max-min problem becomes
\[ \max_{x \in D} \min_i -\exp(-\lambda||x - x_i||^2). \]  
(10)
Due to monotonicity of the function \( f(x) = \exp(x) \), the problem is further simplified as
\[ \max_{x \in D} \min_i ||x - x_i||^2, \]  
(11)
which is in fact the formulation of the max-min sampling algorithm. It is understood that solving this problem blindly is costly and a special algorithm is needed. In brief, one shall first derive all bisecting hyperplanes between each pair of labeled points. Use these hyperplanes as constraints to form a set of maximization problems and find the overall maximum solution \(^2\).

One thing we shall note here in case the reader may question is that using the sum of kernel distances instead of the minimum distance as the objective will not serve the purpose of KFF. This is because the sum of distances is always a convex function, and thus the maximum solution will always lie on the boundary of \(D\).

### 2.3 Algorithm SIMPLE KFF

Algorithm SIMPLE KFF is a hybrid method. The query point is generated on the decision boundary and it is kernel farthest away from all labeled points. Or formally, the next \(x\) to label solves the following problem:

\[
\max_{x \in D} \min_i \|x - x_i\|^2 \quad (12)
\]

s.t. \( \sum_{i=1}^{n} \alpha_i K(x, x_i) = 0. \)

However, in the experiment we used a similar but simpler model:

\[
\max_{x \in D} \sum_i ||x - x_i||^2 \quad (13)
\]

s.t. \( \sum_{i=1}^{n} \alpha_i K(x, x_i) = 0, \)

where \(||L||\) is the cardinality of the labeled set.

### 3 Results

We setup an experiment as follows:

\[
D = \{(x, y) \mid x \in [0, 1], \ y \in [0, 1]\},
\]

\[
D^+ = \{(x, y) \mid x - y > 0, \ (x - 0.5)^2 + (y - 0.5)^2 < 0.3^2\}.
\]

\(^2\)This is not yet implemented. fmincon is used.
A fixed labeled set $L$ is given at the beginning. 40 query points are generated separately using the three algorithms. A demonstration of how the query process looks like is shown in Figure 1 and its final decision boundary in Figure 2. We record the resulting SVM models and compare them with the constraint $D^+$. We use the miss-classification error as an error measure, which can be formulated as:

$$\text{error rate} = \frac{\text{measure}\left(\{x \mid \text{SVM}(x) > 0\} \triangle D^+\right)}{\text{measure}(D)} \times 100\%.$$  \hspace{1cm} (14)

The measure function here simply calculates the area of a set. Since the algorithms are stochastic, we run five experiments for each one and the statistics are shown in Figure 3 through 5. Observation from these results shows directions for future improvements. First of all, the large variation in the KFF plot shows that fmincon performs poorly on the max-min distance problem. Secondly, the convergence rate in all algorithms are less than encouraging. In fact, we observed a fact that as the number of sample points goes up, the refinement of the decision boundary becomes less and less significant. Recall that in the SVM objective we have two conflicting sub-objectives: the minimization of over-fit rate $^3$ and the minimization of miss-classification on the training data. The SVM parameter $C$ on the latter part determines how much importance we put on each of the sub-objectives. In all experiments, $C$ is set to $1e10$, meaning that the model complexity (an indicator of the over-fit rate) is of high priority, and thus local modification to the decision boundary is made. To test if this idea is correct, we perform another 5 experiments using SIMPLE-KFF and $C = 1e5$. The results are shown in Figure 6. Observing a clear improvement, the question of what value of $C$ shall we use arises and will be looked into.

4 Remaining Issues

A big remaining question is: when shall we stop the query? Or how much confidence do we have if we stop at some query step? As far as I know, there is no answer to this. However, one reference we shall look at is the VC theory $^4$, which gives the upper bound of the test error of a classifier if the samples are drawn i.i.d.:

$$\text{test error} = \text{training error} + \sqrt{\frac{h(\log(2N/h) + 1) - \log(\eta/4)}{N}}.$$  \hspace{1cm} (15)

$^3$Or the minimization of the margin in the geometric sense

$^4$Vapnik - Chervonenkis theory
Figure 1: SIMPLE-KFF query demonstration. The circled points are from the initial data set. Query points are numbered and starred.

Here $h$ is the VC dimension (complexity of the classifier, test error is monotonically increasing along $h$), $N$ is the sample size and $\eta$ is the confidence.

References


Figure 2: SIMPLE-KFF query result.

Figure 3: Algorithm SIMPLE results.
Figure 4: Algorithm KFF results.

Figure 5: Algorithm SIMPLE-KFF results.
Figure 6: Algorithm SIMPLE-KFF results with $C = 1e5$. 