Support vector vector comparison machines
Toby HOCKING†, Supaporn SPANURATTANA†, and Masashi SUGIYAMA†
† Department of Computer Science, Tokyo Institute of Technology
2-12-1-W8-74, O-okayama, Meguro-ku, Tokyo, 152-8552, Japan

Abstract In ranking problems, the goal is to learn a ranking function \( r(x) \in \mathbb{R} \) from labeled pairs \( x, x' \) of input points. In this paper, we consider the related comparison problem, where the label \( y \in \{-1,0,1\} \) indicates which element of the pair is better, or if there is no significant difference. We cast the learning problem as a margin maximization, and show that it can be solved by converting it to a standard SVM. We compare our algorithm to SVMrank using simulated data.

Key words support vector machine, quadratic program, linear program, ranking, margin, comparison

1. Introduction

In this paper we consider the supervised comparison problem. Assume that we have \( n \) labeled training pairs and for each pair \( i \in \{1, \ldots, n\} \) we have input features \( x_i, x'_i \in \mathbb{R}^p \) and a label \( y_i \in \{-1,0,1\} \) that indicates which element is better:

\[
y_i = \begin{cases} 
-1 & \text{if } x_i \text{ is better than } x'_i, \\
0 & \text{if } x_i \text{ is as good as } x'_i, \\
1 & \text{if } x'_i \text{ is better than } x_i.
\end{cases}
\]

These data are geometrically represented by segments and arrows in the top panel of Figure 1.

Comparison data naturally arise when considering subjective human evaluations of pairs of items. For example, if I were to compare some pairs of movies I have watched, I would say Les Misérables is better than Star Wars, and The Empire Strikes Back is as good as Star Wars. Features \( x_i, x'_i \) of the movies can be length in minutes, year of theatrical release, indicators for genre, actors/actresses, directors, etc.

The goal of learning is to find a comparison function \( c : \mathbb{R}^p \times \mathbb{R}^p \to \{-1,0,1\} \) which generalizes to a test set of data, as measured by the zero-one loss:

\[
\text{minimize } \sum_{i \in \text{test}} I[c(x_i, x'_i) \neq y_i],
\]

where \( I \) is the indicator function.

The rest of this article is organized as follows. In Section 2, we discuss links with related work on classification and ranking, then in Section 3, we propose a new algorithm: SVMcompare. We show results on simulated data in Section 4, and discuss future work in Section 5.

2. Related work

First we discuss connections with several existing methods in the machine learning literature, and then we discuss how ranking algorithms can be applied to the comparison problem.

2.1 Rank, reject, and rate

Comparison is similar to ranking and classification with a reject option (Table 1). The classification with reject option is a version of binary classification with outputs \( y \in \{-1,0,1\} \), where 0 signifies “rejection” or “no guess” [1]. There are many algorithms for the supervised learning to rank problem [2], which is similar to the supervised comparison problem we consider in this paper. The main idea of learning to rank is to train on labeled pairs of possible documents \( x_i, x'_i \) for the same search query. The labels are \( y_i \in \{-1,1\} \), where \( y_i = 1 \) means document \( x'_i \) is more relevant than \( x_i \) and \( y_i = -1 \) means the opposite. The SVM-rank algorithm was proposed for this problem [3], and the algorithm we propose here is similar. The difference is that we also consider the case where both items/documents are judged to be equally good (\( y_i = 0 \)). A boosting algorithm has been proposed for this “ranking with ties” problem [4], and the authors observed that modeling ties is more effective

<table>
<thead>
<tr>
<th>Outputs</th>
<th>Inputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>( y \in {-1,1} )</td>
<td>single items ( x )</td>
</tr>
<tr>
<td>( y \in {-1,0,1} )</td>
<td>pairs ( x, x' )</td>
</tr>
</tbody>
</table>

Table 1 Comparison is similar to ranking and classification with reject option.
when there are more output values. A Bayesian model which can be applied to these data is TrueSkill [5], a generalization of the Elo chess rating system. When the inputs are discrete \( x_i, x'_i \in \{1, \ldots, k\} \), then the problem is known as learning a relation [6]. In this article we consider the case when inputs are continuous \( x_i, x'_i \in \mathbb{R}^p \).

2.2 SVMrank for comparing

In this section we explain how to apply the existing SVMrank algorithm to a comparison data set.

The goal of the SVMrank algorithm is to learn a ranking function \( r : \mathbb{R}^p \to \mathbb{R} \). When \( r(x) = w^\top x \) is linear, the primal problem for some cost parameter \( C \in \mathbb{R}^+ \) is the following quadratic program (QP):

\[
\begin{align*}
\text{minimize} \quad & \frac{1}{2} w^\top w + C \sum_{i \in I_1 \cup I_2} \xi_i \\
\text{subject to} \quad & \forall i \in I_1 \cup I_2, \xi_i \geq 0, \\
& \text{and } \xi_i \geq 1 - w^\top (x'_i - x_i)y_i,
\end{align*}
\]

where \( I_y = \{i \mid y_i = y\} \) are the sets of indices for the different labels. Note that the equality pairs \( i \in I_0 \) are not used in the optimization problem.

After obtaining a weight vector \( w \in \mathbb{R}^p \) and ranking function \( r \) by solving (3), we define a threshold \( \tau \in \mathbb{R}^+ \) and a thresholding function \( t_\tau : \mathbb{R} \to \{-1, 0, 1\} \)

\[
t_\tau(z) = \begin{cases} -1 & \text{if } z < -\tau, \\ 0 & \text{if } |z| \leq \tau, \\ 1 & \text{if } z > \tau. \end{cases}
\]

A comparison function \( c_\tau : \mathbb{R}^p \times \mathbb{R}^p \to \{-1, 0, 1\} \) is defined as the thresholded difference of predicted ranks

\[
c_\tau(x, x') = t_\tau(r(x') - r(x)).
\]

We can then use grid search to estimate an optimal threshold \( \hat{\tau} \), by minimizing the zero-one loss with respect to all the training pairs:

\[
\hat{\tau} = \arg\min_\tau \sum_{i=1}^n \mathbb{I} \left[ c_\tau(x_i, x'_i) \neq y_i \right].
\]

However, there are two potential problems with the learned comparison function \( c_\tau \). First, the equality pairs \( i \in I_0 \) are not used to learn the weight vector \( w \) in (3). Second, the threshold \( \tau \) is learned in a separate optimization step, which may be suboptimal. In the next section, we propose a new algorithm that fixes these potential problems.

3. Support vector comparison machines

In this section we discuss new learning algorithms for comparison problems. In all cases, we will first learn a ranking function \( r : \mathbb{R}^p \to \mathbb{R} \) and then a comparison function

**Figure 1** Geometric interpretation. Top: input feature pairs \( x_i, x'_i \in \mathbb{R}^p \) are segments and arrows, colored using the labels \( y_i \in \{-1, 0, 1\} \). The level curves of the ranking function \( r(x) = \|x\|^2 \) are grey, and differences \( |r(x) - r(x')| \leq 1 \) are considered insignificant (\( y_i = 0 \)). Middle: in the enlarged feature space, the ranking function is linear: \( r(x) = w^\top \Phi(x) \). Bottom: two symmetric hyperplanes \( w^\top [\Phi(x') - \Phi(x_i)] \in \{-1, 1\} \) are used to classify the difference vectors.
\( c : \mathbb{R}^p \times \mathbb{R}^p \rightarrow \{-1, 0, 1\} : \)
\[ c(x, x') = t_1(r(x') - r(x)), \quad (7) \]

using the threshold function \( t_1 : \mathbb{R} \rightarrow \{-1, 0, 1\} \) (4). In other words, any small difference in ranks \( |r(x') - r(x)| \leq 1 \) is considered insignificant, and there are two decision boundaries \( r(x') - r(x) \in \{-1, 1\} \).

3.1 LP and QP for separable data

To illustrate the nature of the max-margin comparison problem, in this section we assume that the training data are linearly separable. Later in Section 3.2, we propose an algorithm for learning a nonlinear function from non-separable data.

Using the following linear program (LP), we learn a linear ranking function \( r(x) = w^\top x \) that maximizes the geometric margin \( \mu \). As shown in the left panel of Figure 2, the geometric margin \( \mu \) is the smallest distance from any difference vector \( x' - x \), to a decision boundary \( r(x) \in \{-1, 1\} \). The max margin LP is

\[
\begin{align*}
\text{maximize} & \quad \mu \\
\text{subject to} & \quad \mu \leq 1 - |w^\top(x'_i - x_i)|, \quad \forall i \in I_0, \\
& \quad \mu \leq -1 + w^\top(x'_i - x_i)y_i, \quad \forall i \in I_1 \cup I_{-1}.
\end{align*}
\]

(8)

Note that finding a feasible point for this LP is a test of linear separability. If there are no feasible points then the data are not linearly separable.

Another way to formulate the comparison problem is by first performing a change of variables, and then solving a binary SVM QP. The idea is to maximize the margin between significant differences \( y_i \in \{-1, 1\} \) and equality pairs \( y_i = 0 \). Let \( X_y, X'_y \) be the \(|I_y| \times p\) matrices formed by all the pairs \( i \in I_y \). We define a new “flipped” data set with \( m = |I_1| + |I_{-1}| + 2|I_0| \) pairs suitable for training a binary SVM:

\[
\tilde{X} = \begin{bmatrix} X_1 \\ X'_{-1} \\ X_0 \end{bmatrix}, \quad \tilde{X}' = \begin{bmatrix} X'_1 \\ X_{-1} \\ X'_0 \end{bmatrix}, \quad \tilde{y} = \begin{bmatrix} 1_{|I_1|} \\ -1_{|I_{-1}|} \\ -1_{|I_0|} \end{bmatrix}, \quad (9)
\]

where \( 1_n \) is an \( n \)-vector of ones, \( \tilde{X}, \tilde{X}' \in \mathbb{R}^{m \times p} \) and \( \tilde{y} \in \{-1, 1\}^m \). Note that \( \tilde{y}_i = -1 \) implies no significant difference between \( \tilde{x}_i \) and \( \tilde{x}'_i \), and \( \tilde{y}_i = 1 \) implies that \( \tilde{x}'_i \) is better than \( \tilde{x}_i \). We then learn an affine function \( f(x) = \beta + u^\top x \) using a binary SVM QP (black lines in middle and right panels of Figure 2):

\[
\begin{align*}
\text{minimize} & \quad u^\top u \\
\text{subject to} & \quad \tilde{y}_i(\beta + u^\top(\tilde{x}'_i - \tilde{x}_i)) \geq 1, \quad \forall i \in \{1, \ldots, m\}.
\end{align*}
\]

(10)

Intuitively, the SVM QP learns a separator \( f(x) = 0 \) between significant difference pairs \( \tilde{y}_i = 1 \) and insignificant difference pairs \( \tilde{y}_i = -1 \). However, we want a comparison function that predicts \( c(x, x') \in \{-1, 0, 1\} \). So we use the next Lemma to construct a ranking function \( r(x) = \tilde{w}^\top x \) that is feasible for the original max margin comparison LP (8), and can be used with the comparison function (7).

Lemma 1. Let \( u \in \mathbb{R}^p, \beta \in \mathbb{R} \) be a solution of (10). Then \( \tilde{\mu} = -1/\beta \) and \( \tilde{\omega} = -u/\beta \) are feasible for (8).

Proof. Begin by assuming that we want to find a ranking function \( r(x) = \tilde{\omega}^\top x = \gamma u^\top x \), where \( \gamma \in \mathbb{R} \) is a scaling constant. Then consider that for all \( x \) on the decision boundary, we have

\[
\begin{align*}
\text{point} & \quad \bullet \quad \text{LP constraint active} \\
& \quad \bigcirc \quad \text{LP constraint inactive} \\
& \quad \blacklozenge \quad \text{QP support vector} \\
\text{line} & \quad \text{decision } r(x) = \pm 1 \\
& \quad \text{margin } r(x) = \pm 1 \pm \mu
\end{align*}
\]

Figure 2 The separable LP and QP comparison problems. Left: the difference vectors \( x' - x \) of the original data and the optimal solution to the LP (8). Middle: for the unscaled flipped data \( \tilde{x}' - \tilde{x} \) (9), the LP is not the same as the QP (13). Right: in these scaled data, the QP is equivalent to the LP.
\[ r(x) = \tilde{w}^T x + 1 \quad \text{and} \quad f(x) = u^T x + \beta = 0. \]  
\[ r(x) = \tilde{w}^T x = 1 + \tilde{\mu} \quad \text{and} \quad f(x) = u^T x + \beta = 1. \]

Taken together, it is clear that \( \gamma = -1/\beta \) and thus \( \tilde{w} = -u/\beta \). Consider for all \( x \) on the margin we have

\[ r(x) = \tilde{w}^T x = 1 + \tilde{\mu} \quad \text{and} \quad f(x) = u^T x + \beta = 1. \]

Taken together, these imply \( \tilde{\mu} = -1/\beta \). Now, by definition of the flipped data (9), we can re-write the max margin QP (10) as

\[
\begin{align*}
\text{minimize} & \quad u^T u \\
\text{subject to} & \quad \beta + |u^T (x'_i - x_i)| \leq -1, \ \forall \ i \in I_0, \\
& \quad \beta + u^T (x'_i - x_i) y_i \geq 1, \ \forall \ i \in I_1 \cup \bar{I}_0.
\end{align*}
\]

By re-writing the constraints of (13) in terms of \( \tilde{\mu} \) and \( \tilde{w} \), we recover the same constraints as the max margin comparison LP (8). Thus \( \tilde{\mu}, \tilde{w} \) are feasible for (8). \( \square \)

One may also wonder: are \( \tilde{\mu}, \tilde{w} \) optimal for the max margin comparison LP? In general, the answer is no, and we give one counterexample in the middle panel of Figure 2. This is because the LP defines the margin in terms of the size of the normal vector \( r \) values, which implicitly cause the LP to define the margin in terms of ranking function coefficients such that \( f(x) = \tilde{\mu} + \tilde{\beta} \). However, when the input variables are scaled in a pre-processing step, we have observed that the solutions to the LP and QP are equivalent (right panel of Figure 2).

Lemma 1 is very useful in practice. It means that a ranking function \( r \) can be learned on comparison data by first solving a standard binary SVM, and then transforming the solution. We use this result in the next section to build a general support vector algorithm for comparison data.

### 3.2 Kernelized QP for non-separable data

In this section, we assume the data are not separable, and want to learn a nonlinear ranking function. We define a positive definite kernel \( \kappa : \mathbb{R} \times \mathbb{R} \rightarrow \mathbb{R} \), which implicitly defines an enlarged set of features \( \Phi(x) \) (middle panel of Figure 1). As in (10), we learn a function \( f(x) = \beta + u^T \Phi(x) \) which is affine in the feature space. Let \( \alpha, \alpha' \in \mathbb{R}^m \) be coefficients such that \( u = \sum_{i=1}^m \alpha_i \Phi(\tilde{x}_i) + \alpha'_i \Phi(\tilde{x}'_i) \), and so we have \( f(x) = \beta + \sum_{i=1}^m \alpha_i \kappa(\tilde{x}_i, x) + \alpha'_i \kappa(\tilde{x}'_i, x) \). We then use Lemma 1 to define the ranking function

\[
r(x) = \frac{u^T \Phi(x)}{-\beta} = \sum_{i=1}^m \frac{\alpha_i \kappa(\tilde{x}_i, x) + \alpha'_i \kappa(\tilde{x}'_i, x)}{-\beta}.
\]

Let \( K = [K_1 \ldots K_m] K'_1 \cdots K'_m] \in \mathbb{R}^{2m \times 2m} \) be the kernel matrix, where for all pairs \( i \in \{1, \ldots, m\} \), the kernel vectors \( K_i, K'_i \in \mathbb{R}^m \) are defined as

\[
K_i = \begin{bmatrix} \kappa(\tilde{x}_1, \tilde{x}_i) \\ \vdots \\ \kappa(\tilde{x}_m, \tilde{x}_i) \end{bmatrix}, \quad K'_i = \begin{bmatrix} \kappa(\tilde{x}_1, \tilde{x}'_i) \\ \vdots \\ \kappa(\tilde{x}_m, \tilde{x}'_i) \end{bmatrix}.
\]

Letting \( a = [a^T \alpha'^T]^T \in \mathbb{R}^{2m} \), the norm of the linear function in the feature space is \( u^T a = a^T K a \), and we can write the primal soft-margin comparison QP for some \( C \in \mathbb{R}^+ \) as

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} a^T K a + C \sum_{i=1}^m \xi_i \\
\text{subject to} & \quad \forall i \in \{1, \ldots, m\}, \ \xi_i \geq 0, \\
& \quad \xi_i \geq 1 - y_i (\beta + a^T (K'_i - K_i)).
\end{align*}
\]

Let \( \lambda, v \in \mathbb{R}^m \) be the dual variables, let \( Y = \text{Diag}(\hat{y}) \) be the diagonal matrix of \( m \) labels. Then the Lagrangian can be written as

\[
L = \frac{1}{2} a^T K a + C \sum_{i=1}^m \xi_i - \lambda^T (1 + y \hat{y})^T M^T K a - \xi_i),
\]

where \( M = [-I_m I_m]^T \in \{-1, 0, 1\}^{2m \times m} \). Solving \( \nabla_a L = 0 \) results in the following stationary condition:

\[
a = M Y v.
\]

The rest of the derivation of the dual comparison problem is the same as for the standard binary SVM. The resulting dual QP is

\[
\begin{align*}
\text{minimize} & \quad \frac{1}{2} v^T M^T K M Y v - v^T 1_m \\
\text{subject to} & \quad \sum_{i=1}^m v_i y_i = 0, \\
& \quad \forall i \in \{1, \ldots, m\}, \ 0 \leq v_i \leq C,
\end{align*}
\]

which is equivalent to the dual problem of a standard binary SVM with kernel \( \tilde{K} = M^T K M \in \mathbb{R}^{m \times m} \) and labels \( \hat{y} \in \{-1, 1\}^m \).

So we can solve the dual comparison problem (19) using any efficient SVM solver, such as libsvm [7]. We used the R interface in the kernlab package [8], and our code is available in the tdock/rankSWcompare package on Github.

After obtaining optimal dual variables \( v \in \mathbb{R}^m \) as the solution of (19), the SVM solver also gives us the optimal bias \( \beta \) by analyzing the complementary slackness conditions. The learned ranking function can be quickly evaluated since the optimal \( v \) is sparse. Let \( \text{sv} = \{i \mid v_i > 0\} \) be the indices of the support vectors. Since we need only \( 2|\text{sv}| \) kernel evaluations, the ranking function (14) becomes
\[ r(x) = \sum_{i \in \mathcal{AV}} \tilde{y}_i v_i \left[ \kappa(\tilde{x}_i, x) - \kappa(\tilde{z}_i, x) \right] / \beta. \] (20)

Note that for all \( i \in \{1, \ldots, m\} \), the optimal primal variables \( \alpha_i = -\tilde{y}_i v_i \) and \( \alpha'_i = \tilde{y}_i v_i \) are recovered using the stationary condition (18). The learned comparison function remains the same (7).

The training procedure is summarized as Algorithm 1. There are two sub-routines: \textsc{KernelMatrix} computes the \( 2m \times 2m \) kernel matrix, and \textsc{SVMdual} solves the SVM dual QP (19). There are two hyper-parameters to tune: the cost \( C \) and the kernel \( \kappa \). As with standard SVM for binary classification, these parameters can be tuned by minimizing the prediction error on a held-out validation set.

4. Results

The goal of our learning algorithm is to accurately predict a test set of labeled pairs (2). We use the zero-one loss for evaluating the learned comparison function on the test set (Table 2).

4.1 Simulation: norms in 2d

We used a simulation to test the proposed SVMcompare model and a baseline ranking model that ignores the equality \( y_i = 0 \) pairs, SVMrank [3]. The goal of our simulation is to demonstrate that our model can perform better by learning from the equality \( y_i = 0 \) pairs, when there are few inequality \( y_i \in \{-1, 1\} \) pairs. We generated pairs \( x_i, x'_i \in [-3, 3]^2 \) and noisy labels \( y_i = t_i [r(x_i) - r(x_i) + \epsilon_i] \), where \( t_i \) is the threshold function (4), \( r \) is the latent ranking function, and \( \epsilon_i \sim N(0, 1/4) \) is noise. We picked train, validation, and test sets, each with \( n/2 \) equality pairs and \( n/2 \) inequality pairs, for \( n \in \{50, \ldots, 800\} \). We fit a \( 10 \times 10 \) grid of models to the training set (cost parameter \( C = 10^{-3}, \ldots, 10^3 \), Gaussian kernel width \( 2^{-7}, \ldots, 2^4 \)), select the model with minimal zero-one loss on the validation set, and then use the test set to estimate the generalization ability of the selected model. For the SVMrank model explained in Section 2.2, equality pairs are ignored when learning the ranking function, but used to learn a threshold \( \hat{r} \) via grid search for when to predict \( c(x, x') = 0 \).

In Figure 3 we show the training set for \( n = 100 \) pairs, and the level curves of the ranking functions learned by the SVMrank and SVMcompare models. It is clear that SVMcompare recovers a ranking function that is closer to the latent \( r \), especially in the case of the squared \( \ell_2 \) norm \( r(x) = ||x||_2^2 \).

In Figure 4 the test error of the proposed SVMcompare model is clearly lower than that of the baseline SVMrank model, especially with more labeled pairs \( n \).

5. Conclusions and future work

We discussed an extension of SVM to comparison problems. Our results highlighted the importance of directly modeling the equality pairs (\( y_i = 0 \)), and it will be interesting to see if the same results are observed in learning to rank data sets. For scaling to very large data sets, it will be interesting to try algorithms based on smooth discriminative loss functions, such as stochastic gradient descent with a logistic loss.

Acknowledgements: TDH was funded by KAKENHI 23120004, SS by a MEXT scholarship, and MS by KAKENHI 26700022. Thanks to Simon Lacoste-Julien and Hang Li for helpful discussions.

References


Figure 3  Application to simulated patterns $x, x' \in \mathbb{R}^2$, where the ranking functions $r(x) = ||x||^2$ are squared norms (panels from left to right). Top: the training data are $n = 100$ pairs, half equality (segments indicate two points of equal rank), and half inequality (arrows point to the higher rank). Middle and bottom: level curves of the ranking functions learned by SVMrank and SVMcompare. SVMrank ignores the equality pairs, so in general SVMcompare recovers the latent pattern better.

Figure 4  Test error of SVMrank and SVMcompare (the Bayes error rate of the latent $r$ is shown for comparison). We plot mean and standard deviation of the prediction error across 4 randomly chosen data sets, as a function of training set size $n$ (a vertical line shows the data set size $n = 100$ which was used in Figure 3). It is clear that in general SVMcompare makes better predictions on the test data.