A MILP Formulation for Setwise Max-Margin Learning

Paolo Viappiani[1]

Abstract. In max-margin learning, the system aims at establishing a solution as robust as possible. In this paper we extend the idea of max-margin learning to cases where the goal is to produce a set of solutions, instead of a single one. In particular, we focus on the problem of preference elicitation, but we believe our idea could be adapted to other situations. We present a MILP formulation for our "setwise" max margin learner and discuss current ongoing works.

1 Max-margin Learning for Preference Optimization

The notion of margin is important in several machine learning algorithms. By maximizing the margin, the learner ensures robustness in the solution. In particular, a max-margin approach has been proposed for optimization with a partially specified preference profile and elicitation of further preferences [2]. Automatic assessment of preferences is an important component of many artificial intelligence systems [3].

In this paper, we show how to extend the idea of max-margin for optimizing a set and we provide a MILP formulation. The result of the computation can be used to generate a set of recommendations and to generate a "diverse" set of feasible utility functions. This can be useful for further elicitation as we may ask the user a choice query based on this set ("*Among these items, which one do you prefer ?*").

First of all, we review the standard (singleton) max-margin learning problem. We assume a multi attribute feature space $\mathbf{X}$ (we assume possible options are specified by configuration constraints) and that the user preferences are encoded by a vector of weights $\mathbf{w} = (w_1, ..., w_m)$; the utility of an option $\mathbf{y}$ is then $\mathbf{w} \cdot \mathbf{y}$.

The weight vector $\mathbf{w}$ is unknown but we are given constraints (that encode a "feasible region"). A pairwise comparison between a more preferred product $\mathbf{y}_+$ and a less preferred product $\mathbf{y}_-$ is encoded by a constraint $\mathbf{w} \cdot (\mathbf{y}_+ - \mathbf{y}_-) \geq 0$. Given a number of such pairwise comparisons, represented by a set D (the "learning set"), and, possibly, other additional constraints (representing some kind of "prior" knowledge, as for example $w_2 > 0.3$) we want to find a vector $\mathbf{w}$ of parameters such that all constraints are satisfied and does that by the largest margin.

As shown by [2] the problem can be easily formulated as a linear program. The shared margin is represented by the decision variable M, that we want to maximize. We require, for each pairwise statement in D, to have $\mathbf{w} \cdot (\mathbf{y}^h_+ - \mathbf{y}^h_-) \geq M$.

$$\max_{M,\mathbf{w}} \; M$$
$$s.t. \;\; \mathbf{w} \cdot (\mathbf{y}^h_+ - \mathbf{y}^h_-) \geq M \qquad \forall (\mathbf{y}^h_+, \mathbf{y}^h_-) \in D$$

The optimization can be made tolerant to contradictory or hard-to-satisfy constraints; this can be done by adding explicit slack variables ε_h.

$$\max_{M,\mathbf{w}} \; M - \alpha \sum_h \varepsilon_h \tag{1}$$
$$s.t. \;\; \mathbf{w} \cdot (\mathbf{y}^h_+ - \mathbf{y}^h_-) \geq M - \varepsilon_h \qquad \forall (\mathbf{y}^h_+, \mathbf{y}^h_-) \in D \tag{2}$$

Where α is a parameter that controls the tolerance to a small number of violated constraints or constraints satisfied by a margin smaller than M. If $\alpha = 0$, then all original constraints must be strictly satisfied.

Once the optimization has been solved, the learned $\mathbf{w}$ can be used to choose a recommendation $\mathbf{x}^*$ by maximizing $\mathbf{w} \cdot \mathbf{x}$. If the space of feasible options (or products) is a configuration space encoded by linear constraints, the recommendation can be computed by solving a linear program. We refer to $\mathbf{x} \in \mathit{Feasible}(\mathbf{X})$ to specify that an option must satisfy the configuration constraints.

In interactive settings, one should decide the query to ask next. [2] suggests asking queries whose associated hyperplane divide the feasible region in roughly equal part, but to do so they require considering all possible pairs of options; this approach can only be feasible in small datasets and do not apply in configuration deomains. Instead, a viable method for large configuration spaces should be able to retrieve a set of options in a single optimization.

2 Setwise Max-Margin

We extend *maxmargin* optimization of preference weights to sets. Given, as before, a number of pairwise statements D and possibly additional constraints (the "learning set") our goal is to find a number of utility weight vectors $\mathbf{w}^1, ..., \mathbf{w}^k$ (with k given), consistent with out knowledge about the user, but that are representative of different parts of the weights' feasible region.

In order to do that, we require, as before, a shared margin M to hold for all constraints associated with prior knowledge and responses. We also use the same margin to enforce each vector $\mathbf{w}^i$ to be as "farther away" from each other as possible. This insight is realized by introducing a set of options $\{\mathbf{x}^1, ..., \mathbf{x}^k\}$ as decision variables (of the same cardinality k). For each i, we impose $\mathbf{x}^i$ to be the best option among them

[1] Aalborg University, Denmark, email: paolo@cs.aau.dk

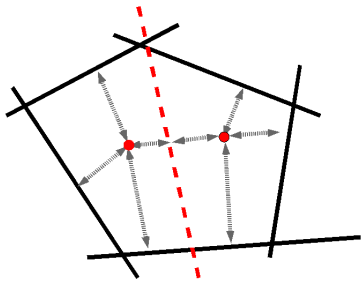

Figure 1. Pairwise max-margin ($k = 2$). The feasible region in the vector space (black constraints) is divided in two parts by the red hyperplane. The two weight vectors w_1 and w_2 (red circles), together with the hyperplane, are chosen maximize the minimum of the margins (gray arrows).

(offering higher utility) when evaluated according to $\mathbf{w}^i$ - and we require these constraints to hold by at least a margin M. The optimization is the following:

$$\max_{M,\mathbf{w}^i,\mathbf{x}^i} M - \alpha_1 \sum_h \varepsilon_h - \alpha_2 \sum_{i,j} \varepsilon'_{i,j} \tag{3}$$

$$s.t.\ \ \mathbf{w}^i \cdot (\mathbf{y}^h_+ - \mathbf{y}^h_-) \geq M - \varepsilon_h \qquad \forall (\mathbf{y}^h_+, \mathbf{y}^h_-) \in D, \forall i \in [1,k] \tag{4}$$

$$\mathbf{w}^i \cdot (\mathbf{x}^i - \mathbf{x}^j) \geq M - \varepsilon'_{i,j} \qquad \forall j \neq i;\ i,j \in [1,k] \tag{5}$$

$$\mathbf{x}^i \in Feasible(\mathbf{X}) \qquad \forall i \in [1,k]$$

Note that, we are choosing the options $\mathbf{x}^1, ..., \mathbf{x}^k$ and the weights $\mathbf{w}^1, ..., \mathbf{w}^k$ simultaneously: since we want to maximize M, the optimizer will be better off by choosing a set of outcomes $\mathbf{x}^i$ that divide the weight space roughly equally, and the utility functions such each $\mathbf{w}^i$ should lie (intuitively) near the centre of each subregion.

This initial formulation is problematic, as we have quadratic terms. However, there is a solution: in fact, by using integer programming tricks, the problem can be formulated as a mixed integer linear program (MILP).

$$\max\ \ M - \alpha_1 \sum_h \varepsilon_h - \alpha_2 \sum_{i,j} \varepsilon'_{i,j} \tag{6}$$

$$s.t.\ \ \mathbf{w}^i \cdot (\mathbf{y}^h_+ - \mathbf{y}^h_-) \geq M - \varepsilon_h \qquad \forall (\mathbf{y}^h_+, \mathbf{y}^h_-) \in D, \forall i \in [1,k] \tag{7}$$

$$\sum_z A^i_z - B^{i,j}_z \geq M - \varepsilon'_{i,j} \qquad \forall j \neq i;\ i,j \in [1,k] \tag{8}$$

$$A^i_z \leq w{\uparrow}\, x^i_z \qquad \forall i \in [1,k], z \in [1,m] \tag{9}$$

$$A^i_z \leq w^i_z \qquad \forall i \in [1,k], z \in [1,m] \tag{10}$$

$$B^{i,j}_z \geq w^i_z - C \cdot (1 - x^j_z) \qquad \forall j \neq i \in [1,k], z \in [1,m] \tag{11}$$

$$B^{i,j}_z \geq 0 \qquad \forall j \neq i \in [1,k], z \in [1,m] \tag{12}$$

$$\mathbf{x} \in Feasible(\mathbf{X}) \qquad \forall i \in [1,k]$$

In the optimization, the decision variables are the following:

- M is the size of the shared margin
- $\mathbf{w}^1, ..., \mathbf{w}^k$ is a set of utility vectors; each vector defined over m attributes (features): $\mathbf{w}^i = (w^i_1, ..., w^i_m)$
- $\mathbf{x}^1, ..., \mathbf{x}^k$ is a set of configurations (options) with each configuration $\mathbf{x}^i = (x^i_1, ..., x^i_m)$; each element x^i_z is binary
- ε slack variables to represent cost of unsatisfied constraints in D; ε' slack variables to represent cost of not respecting the margin M when choosing the hyperplanes
- $\mathbf{A}^i$ encodes the vector $(w^i_1 x^i_1, ..., w^i_m x^i_m)$, the element-by-element product of $\mathbf{w}^i$ and $\mathbf{x}^i$: $A^i_z = w^i_z x^i_z$
- $\mathbf{B}^{i,j}$ encodes $(w^i_1 x^j_1, ..., w^i_m x^j_m)$, the element-by-element product of $\mathbf{w}^i$ and $\mathbf{x}^j$

In addition to the decision variables, we make use of the following parameters:

- k is the size of the set
- α_1, α_2 control the tolerance with respect to violated constraints (we can differentiate the tolerance for the two different types of constraints).
- D is the *learning set*: a set of pairwise comparisons known to the system
- $w{\uparrow}$ is any upper bound of the value of the utility weights
- C is an arbitrary large number
- $\mathbf{y}^h_+, \mathbf{y}^h_-$ are pairs of configurations (options) in the learning set D, with $\mathbf{y}^h_+$ preferred to $\mathbf{y}^h_-$

The solver aims at setting M, that is a decision variable, as large as possible. Constraint 5 of the original program is replaced by constraint 8, enforcing M to be smaller than $\sum_z A^i_z - B^{i,j}_z$ for any i, j. In order to maximize M, the solver will try to keep the A^i_z as large as possible and the $B^{i,j}_z$ as small as possible. The fact that $\sum_z A^i_z - B^{i,j}_z$ evaluates to $\mathbf{w}^i \cdot (\mathbf{x}^i - \mathbf{x}^j)$ is achieved by setting additional constraints. Constraints 9 and 10 together force each $\mathbf{A}^i$ to be the element-by-element product of $\mathbf{w}^i$ and $\mathbf{x}^i$ ($A^i_z = w^i_z x^i_z$): if $x^i_z = 0$ then (constraint 9) also A^i_z must be 0; otherwise (if $x^i_z = 1$) $A^i_z \leq w^i_z$ (constraint 10) but because of the objective function maximizing M, and each A constrain M, the solver will set A^i_z to w^i_z. Similarly constraints 11 and 12 together make it so that $\mathbf{B}^{i,j}$ encodes $(w^i_1 x^j_1, ..., w^i_m x^j_m)$, the element-by-element product of $\mathbf{w}^i$ and $\mathbf{x}^j$ (as C is an arbitrary large constant, constraint 11 is binding only when x^j_z is 1, otherwise it is always satisfied).

3 Discussion

This paper describes an extension of maximum margin learner to sets. We are currently planning to perform experiments to evaluate the efficiency of this approach. We will compare to baseline methods, maximizing product diversity for a single learned weight vector. We are interested in computation time, but also in the quality of the elicitation using our method in an interactive setting. In particular, we are interested in comparing this approach for generating recommendation sets with others based on Bayesian inference [5] and minimax regret [4]. We are also interested in learning approaches ensuring sparsity in the parameter space [1].

REFERENCES

[1] Paolo Campigotto, Andrea Passerini, and Roberto Battiti. Active learning of combinatorial features for interactive optimization. In Carlos A. Coello Coello, editor, *LION*, volume 6683 of *Lecture Notes in Computer Science*, pages 336–350. Springer, 2011.

[2] Krzysztof Gajos and Daniel S. Weld. Preference elicitation for interface optimization. In Patrick Baudisch, Mary Czerwinski, and Dan R. Olsen, editors, *UIST*, pages 173–182. ACM, 2005.

[3] Bart Peintner, Paolo Viappiani, and Neil Yorke-Smith. Preferences in interactive systems: technical challenges and case studies. *AI Magazine*, 29(4):13–24, 2008.

[4] Paolo Viappiani and Craig Boutilier. Regret-based optimal recommendation sets in conversational recommender systems. In *Proceedings of the 3rd ACM Conference on Recommender Systems (RecSys09)*, pages 101–108, New York, 2009.

[5] Paolo Viappiani and Craig Boutilier. Optimal bayesian recommendation sets and myopically optimal choice query sets. In *Advances in Neural Information Processing Systems 23 (NIPS)*, Vancouver, 2010.