A Universal Machine Learning Optimization Framework For Arbitrary Outputs

Sandor Szedmak
School of Electronics and Computer Science
University of Southampton
United Kingdom

Zakria Hussain
Department of Computer Science
University College London
United Kingdom

Abstract

We derive a general Convex Linearly Constrained Program (CLCP) parameterized by a matrix **G**, constructed from the information given by the input-output pairs. The CLCP then chooses a set of regularization and loss functions in order to impose constraints for the learning task. We show that several algorithms, including the SVM, LPBoost, Ridge Regression etc., can be solved using the same optimization framework when the appropriate choice of **G**, regularization and loss function are made. Due to this unification we show that if **G** is constructed from more complex input-output paired information then we can solve more difficult problems such as structured output learning, with the same complexity as a regression/classification problem. We discuss various different forms of **G** and then show on some real world enzyme prediction tasks, requiring structured outputs, that our method performs as well as the state-of-the-art.

1 INTRODUCTION

In this paper we show that many optimization based algorithms in machine learning are connected and can be solved using the same Convex Linearly Constrained Program (CLCP).[1] The main difference between these algorithms is characterized by a matrix **G**, which contains information specific to each learning task, such as for example Support Vector Machines (SVM) [Vapnik, 1998], Linear Program Boosting (LPBoost) [Demiriz et al., 2001] or Ridge Regression [Duda et al., 2001]. The main change required in the CLCP to solve one of these alternative algorithms is a redefinition of the matrix **G**.

[1] sometimes we relax the objective to be Linear, giving us a Linear Program (LP).

One may ask the question, if all the algorithms exist then why bother with this generalization? Our aim is to show that if we would like to solve a more complex (or different) problem then a redefinition of matrix **G** and the use of the correct regularization and loss function is all that is required. This gives us a standard algorithm for which we would only need to evaluate these quantities each time a different learning problem was presented. This may be far easier and efficient than constructing a different optimization problem each time a new machine learning problem is encountered. We show that indeed with the same framework we can solve arbitrary output learning problems such as structured learning with the same ease and efficiency as classification/regression problems, simply by redefining **G**.

We proceed by walking the reader through an example of an CLCP (or Linear Program (LP)) using game theory and then follow this up with the closely related Linear Program Boosting formulation. Next we delve into a more complex scenario by mapping outputs into higher dimensional spaces called the *label* space and use this abstraction to present a more general definition of **G** in terms of input-output mappings into higher dimensional spaces, allowing us to tackle structural learning scenarios. In section 3 we set out our general optimization framework and describe examples using SVMs, learning when distances are given, and a table characterizing several different learning algorithms before finishing with experiments on a real world enzyme prediction task. All notations and definitions will be given as and when they appear.

2 MOTIVATION: TWO-PLAYER ZERO-SUM GAME

The two-player zero-sum game is one of the most elementary problems in game theory, and can be solved using the *minimax* strategy. In this type of game, two players $\mathscr{P}_1$ and $\mathscr{P}_2$, and a *pay-off* matrix $\mathbf{G} \in \mathbb{R}^{m\times n}$ are given where $\mathbf{G} = G_{ij},\ i = 1,\dots,m,\ j = 1,\dots,n$. The following rules are applied in the game:

- Player $\mathscr{P}_1$ chooses one row of $\mathbf{G}$, indexed by i;
- Player $\mathscr{P}_2$ chooses one column of $\mathbf{G}$, indexed by j;
- Player $\mathscr{P}_1$ loses the amount of G_{ij};
- Player $\mathscr{P}_2$ gains the amount of G_{ij}.

Assume that both players know the elements of the pay-off matrix, but their decisions are independent. The game is called a zero-sum game because the sum of the loss of $\mathscr{P}_1$ together with the gain of $\mathscr{P}_2$ equates to zero.

The game can be repeated and each player can change their strategy on the rows and columns. To model their varying strategies one can introduce *mixed strategies*, where both players choose rows and columns with a certain probability; $\mathscr{P}_1$ chooses row i with probability α_i and $\mathscr{P}_2$ chooses column j with probability β_j. Expressing these strategies in a more linear algebraic form, we have two vectors of nonnegative weights $\boldsymbol{\alpha} = (\alpha_i),\ i = 1,\dots,m$ and $\boldsymbol{\beta} = (\beta_j),\ j = 1,\dots,n$ satisfying the property $\sum_i^m \alpha_i = 1$ and $\sum_j^n \beta_j = 1$. We refer to $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ as strategies of the corresponding players. The aim of both players is to find optimal strategies in the minimax sense:

- Player $\mathscr{P}_1$ would like to minimise the loss given by strategy $\boldsymbol{\beta}$ of player $\mathscr{P}_2$:
$$\min_{\alpha} \sum_{i,j}^{m,n} \alpha_i G_{ij} \beta_j,$$
- Player $\mathscr{P}_2$ would like to maximise the gain given by strategy $\boldsymbol{\alpha}$ of $\mathscr{P}_1$:
$$\max_{\beta} \sum_{i,j}^{m,n} \alpha_i G_{ij} \beta_j.$$

Since the strategies are chosen independently the expression $\sum_{i,j}^{m,n} \alpha_i G_{ij} \beta_j$ can be interpreted as the expected value of the loss and gain respectively.

The game above can be formulated as the following optimisation problem:

$$\begin{array}{ll} \max_{\boldsymbol{\beta}} \min_{\boldsymbol{\alpha}} & \sum_{i,j}^{m,n} \alpha_i G_{ij} \beta_j \\ \text{s.t.} & \sum_i^m \alpha_i = 1, \\ & \sum_j^n \beta_j = 1, \\ & \alpha_i \geq 0,\ i = 1,\dots,m \\ & \beta_j \geq 0,\ j = 1,\dots,n. \end{array} \tag{1}$$

This optimization problem can be reformulated and interpreted as a conditional Lagrangian functional (*i.e.*, see Chvatal [1983] for details) of the following primal and dual linear programming problems (expressed in matrix form):

$$\begin{array}{llll} & \textbf{Primal} & & \textbf{Dual} \\ \max & \lambda & \min & \gamma \\ \text{w.r.t.} & \boldsymbol{\beta},\ \lambda & \text{w.r.t.} & \boldsymbol{\alpha},\ \gamma \\ \text{s.t.} & \mathbf{G}\boldsymbol{\beta} \geq \lambda\mathbf{1}, & \text{s.t.} & \mathbf{G}'\boldsymbol{\alpha} \leq \gamma\mathbf{1}, \\ & \mathbf{1}'\boldsymbol{\beta} = 1, & & \mathbf{1}'\boldsymbol{\alpha} = 1, \\ & \boldsymbol{\beta} \geq \mathbf{0}, & & \boldsymbol{\alpha} \geq \mathbf{0}, \end{array} \tag{2}$$

where $\lambda, \gamma > 0$, $\mathbf{1}$ is the all-one vector, $\mathbf{0}$ the all-zero vector and prime $'$ denotes the transpose of a vector (or matrix). The primal expresses the strategy of player $\mathscr{P}_2$ and the dual corresponds to the strategy of player $\mathscr{P}_1$.

The primal and dual formulations that will form the basis of our work follow the formulations of the work of Demiriz et al. [2001]. We state them here without further elaboration.

$$\begin{array}{llll} & \textbf{Primal} & & \textbf{Dual} \\ \min & \mathbf{1}'\mathbf{w} & \max & \mathbf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} & \mathbf{w}, & \text{w.r.t.} & \boldsymbol{\alpha}, \\ \text{s.t.} & \mathbf{G}\mathbf{w} \geq \mathbf{1}, & \text{s.t.} & \mathbf{G}'\boldsymbol{\alpha} \leq \mathbf{1}, \\ & \mathbf{w} \geq \mathbf{0}. & & \boldsymbol{\alpha} \geq \mathbf{0}. \end{array} \tag{3}$$

The key step in the reformulation from LP (2) to LP (3) is the substitution $\mathbf{w} = \boldsymbol{\beta}/\lambda$, where $\lambda > 0$ is assumed to be true.

2.1 BEYOND THE TWO-PLAYER ZERO-SUM GAME: LEARNING AS A GAME

We start by working through an example of a learning algorithm, Linear Program Boosting (LPB), to help demonstrate the connections between game theory and machine learning [Schapire, 2002, Demiriz et al., 2001]. In a machine learning context player $\mathscr{P}_2$ is assumed to be the "Learner" and player $\mathscr{P}_1$ plays on behalf of "Nature".

Given a set of observations (sample) $\mathcal{S} = \{(y_i, x_i)\}$, $i = 1,\dots,m$, $y_i \in \mathcal{Y}$, $x_i \in \mathcal{X}$ where $\mathcal{Y}$ is the output space and $\mathcal{X}$ is the input space, where the first components $\{y_i\}$ are outputs and the second components $\{x_i\}$ are inputs, the Learner tries to find the "best" *prediction* function $f : \mathcal{X} \to \mathcal{Y}$ with respect to a given loss functional $\mathscr{L}$ from a given set of functions $\mathcal{F}$. The loss function is real-valued and measures the quality of the prediction, defined on the set of functions $f \in \mathcal{F}$ considered during learning and on the set of possible samples $\mathcal{S}$.

In the boosting approach [Schapire, 2002] we look for a prediction function f given by a convex combination of other known functions $\mathcal{F}_0 = \{f_1, \ldots, f_n\}$. More precisely the prediction function is $f(x) = \sum_{j=1}^{n} \beta_j f_j(x)$, $\sum_{j=1}^{n} \beta_j = 1$, $\beta_j \geq 0$, $j = 1, \ldots, n$ for all $x \in \mathcal{X}$. These subscripted functions in $\mathcal{F}_0$ are known as weak learners.

In this section we deal with problems where the set of outputs represent a binary classification task, namely $\mathcal{Y} = \{-1, +1\}$. The function f of the learner outputs some positive or negative value and the prediction is equal to the sign of $f(x)$. Now we define the two-player zero-sum game for this learning task. Recall that player $\mathscr{P}_1$ has a strategy $\boldsymbol{\alpha}$, $\sum_{i=1}^{m} \alpha_i = 1$ which can be considered as probabilities of the samples. Player $\mathscr{P}_1$ or Nature wants to find the most "tricky" distribution against the Learner (player $\mathscr{P}_2$) who has the ability to choose the best convex combination of weak learners. The elements of the pay-off matrix G are given by,

$$\boxed{G_{ij} = y_i f_j(x_i)},\ i, j \in 1, \ldots, m, \tag{4}$$

where they are positive when the prediction is correct and negative when it is wrong. Therefore this matrix provides a real pay-off with respect to the Learner.

As stated earlier to guarantee a positive gain for the Learner, we require $\lambda > 0$. We can extend the pay-off matrix and the strategy of $\mathscr{P}_2$ like so (for details see Demiriz et al. [2001]):

$$\begin{array}{llll} & \textbf{Primal} & & \textbf{Dual} \\ \text{min} & \mathbf{1}'\mathbf{w} + C\mathbf{1}'\boldsymbol{\xi} & \text{max} & \mathbf{1}'\boldsymbol{\alpha} \\ \text{w.r.t.} & \mathbf{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^m & \text{w.r.t.} & \boldsymbol{\alpha} \in \mathbb{R}^m, \\ \text{s.t.} & \mathbf{G}\mathbf{w} + \boldsymbol{\xi} \geq \mathbf{1}, & \text{s.t.} & \mathbf{G}'\boldsymbol{\alpha} \leq \mathbf{1}, \\ & \mathbf{w} \geq \mathbf{0}, \boldsymbol{\xi} \geq \mathbf{0}, & & C\mathbf{1} \geq \boldsymbol{\alpha} \geq \mathbf{0}, \end{array} \tag{5}$$

where $C > 0$ is a penalty parameter. This modification consists of a concatenation of the pay-off matrix with the identity matrix $\mathbf{I}$, i.e. $\mathbf{G} = [\mathbf{G}, \mathbf{I}]$, and the extension of the Learners strategy into $[\mathbf{w}, \boldsymbol{\xi}]$, together with a penalty term reducing the possible range of the extended strategy. What does this mean? The Learner, Player $\mathscr{P}_2$, can choose a component of $\boldsymbol{\xi}$ corresponding to the identity part of the new pay-off matrix which turns out to be given by any choice of player $\mathscr{P}_1$'s non-negative outcomes. It may not be an optimal strategy but guarantees a lower bound on the yield of player $\mathscr{P}_2$. A machine learning interpretation of this modification is that it allows the inclusion of misclassifications within the original model given by (3).

We can conclude that in the game based interpretation of the LPB the relationship between the data points and the input and output views can be embedded into the pay-off matrix $\mathbf{G}$ defined by Equation (4). The optimization problem then follows a solution schema for deriving the best strategies for the Learner (and in turn for the Nature) as well as conditions expressed by the constraints imposed on these strategies. We now move onto a more abstract definition of $\mathbf{G}$ that includes classification, regression, structured output learning etc., as special cases.

2.2 BEYOND BINARY CLASSIFICATION: LEARNING ARBITRARY OUTPUTS

In the description of the LPB the outputs were taken from the binary space labeled by $\{-1, +1\}$. We show that this restriction can be eliminated without changing the base form of the optimization problem and consequently the complexity of the underlying optimization task. The idea stems from the interpretation of the pay-off matrix $\mathbf{G}$. First recall the definition of the elements of the pay-off matrix for LPB given by Equation (4). Looking at the primal and dual formulations of the optimization problem (5) we can conclude that:

- the relationship between the inputs $\{x\}$ and outputs $\{y\}$ are completely captured by the pay-off matrix $\mathbf{G}$,
- the remaining parts of the optimization problem (*i.e.*, $\mathbf{w}, \xi$ in the primal and $\boldsymbol{\alpha}$ in the dual) are independent of the sample data, they are only required for the constraints on the players strategies.

Before extending the capability of the Learner we make the following assumptions:

- the input objects (examples) are mapped by the function ϕ into a linear vector space $\mathcal{L}_\phi$,
- the output space $\mathcal{Y}$ is an arbitrary set, and there is a function ψ that maps every output into a linear vector space $\mathcal{L}_\psi$,
- also, there is a vector valued function $\mathbf{\Gamma} : \mathcal{L}_\phi \times \mathcal{L}_\psi \rightarrow \mathcal{H}$ mapping the images of the input and outputs into a *Hilbert space* $\mathcal{H}$ assumed to be of finite dimension.

We call the image space of ϕ the *feature space* and the image space of ψ the *label space.* If the inputs and (or) outputs are originally given as vectors of a Hilbert space $\mathcal{H}$ then ϕ and (or) ψ might be identity mappings. Using this new representation we can define the elements of a *similarity* pay-off matrix $\mathbf{G}$ by

$$\boxed{G_{ij} = \left(\mathbf{\Gamma}(\psi(y_i), \phi(x_i))\right)_j}, \tag{6}$$

where the notation $\mathbf{\Gamma}(\psi(y), \phi(x))$ denotes a "similarity" between the image of a label $\psi(y)$ and the image of an example $\phi(x)$ *e.g.*, an inner product in an inner product space. Furthermore $\left(\mathbf{\Gamma}(\cdot,\cdot)\right)_j$ denotes the jth component of this similarity. We will often refer to this abstract description of $\mathbf{G}$ in the remainder of the paper.

After choosing the pay-off matrix $\mathbf{G}$ we can solve the corresponding learning problem. For example in the primal of LPB we would solve for $\mathbf{w}$ to find the optimal $\mathbf{w}^*$ and make predictions. The prediction can be made with $\mathbf{w}^*$ and is based on the following conjecture.

Conjecture 1. *If the relationship among the data objects is expressed as a monotonically increasing function of similarity measures then the best candidate of the output space is the one that maximizes the margin.*

The prediction $f(x)$ is then expressible (in general) for an arbitrary $x \in \mathcal{X}$ by

$$f(x) = \arg\max_{\forall u \in \tilde{\mathcal{Y}}} \langle \mathbf{w}^*, \mathbf{\Gamma}(\psi(u), \phi(x)) \rangle$$

where $\langle \cdot, \cdot \rangle$ denotes an inner product (for instance) and $\widetilde{\mathcal{Y}} \subseteq \mathcal{Y}$ is a properly chosen subset of the output space, *e.g.*, in binary classification u takes values $\{-1, +1\} = \tilde{\mathcal{Y}}$. Essentially, the prediction function means that we choose a set of outputs and compute $\langle \mathbf{w}^*, \mathbf{\Gamma}(\psi(u), \phi(x)) \rangle$ for each possible output $u \in \tilde{\mathcal{Y}}$ and choose the label that maximizes the margin. This follows an argument closely related to the approach of Tsochantaridis et al. [2005]. The following list gives a set of outputs we would use for classification, regression and structured output.

- Classification: $u \in \tilde{\mathcal{Y}} = \{1, \ldots, k\}$ when we have k classes.
- Regression: $u \in \tilde{\mathcal{Y}} = [a, b]$, when we have a lower bound $a \in \mathbb{R}$ and upper bound $b \in \mathbb{R}$.
- Structured output: $u \in \tilde{\mathcal{Y}}$ where u can take on all possible output structures such as particular types of trees, graphs, objects, etc.

Now we give an example of the similarity function $\mathbf{\Gamma}$. Let the components of $\mathbf{\Gamma}$ be expressed as:

$$\mathbf{\Gamma}(\psi(y_i), \phi(x_i))_j = \langle \psi(y_i), \psi(y_j) \rangle \langle \phi(x_i), \phi(x_j) \rangle, \quad i, j = 1, \ldots, m. \tag{7}$$

The components of $\mathbf{\Gamma}\left(\psi(y_i), \phi(x_i)\right)$ are the products of the coordinates of the vector represented input and output points, $\langle \psi(y_i), \psi(y_j) \rangle$ and $\langle \phi(x_i), \phi(x_j) \rangle$, with respect to the vector system spanned by the vector representation of the sample. Splitting the constraint $\mathbf{G}\mathbf{w}$ of the primal LP (5) row-wise we obtain for each sample,

$$\sum_{j=1}^{m} \langle \psi(y_i), \psi(y_j) \rangle \langle \phi(x_i), \phi(x_j) \rangle w_j \geq 1 - \xi_i, \quad i = 1, \ldots, m, \tag{8}$$

where w_j is an element of $\mathbf{w}$ and ξ_i is an element of $\boldsymbol{\xi}$. Furthermore the product $\langle \psi(y_i), \psi(y_j) \rangle \langle \phi(x_i), \phi(x_j) \rangle$ can be rewritten as inner products of the outer products of the input-output pairs giving us $\langle \psi(y_i) \otimes \phi(x_i), \psi(y_j) \otimes \phi(x_j) \rangle$ where $\otimes$ denotes an *outer product.* Let us introduce a new matrix $\mathbf{W} = \sum_{j=1}^{m} w_j \psi(y_j) \otimes \phi(x_j)$, then the lhs of Equation (8) is expressible in the form

$$\langle \mathbf{W}, \phi(x_i) \otimes \psi(y_i) \rangle,$$

which becomes

$$\langle \psi(y_i), \mathbf{W}\phi(x_i) \rangle. \tag{9}$$

Hence, $\mathbf{W}$ is a matrix with the dimensionality of the label space by the feature space.

For example, if $\psi : \mathcal{Y} \to \{-1, +1\}$ is the mapping of the output space into -1 and $+1$ then $\mathbf{W}$ collapses into a row vector, and the functions f_j taken from the LPB subsume $(\phi(x_i))_j = f_j(x_i)$. Therefore the LPB becomes a special case of this new abstract pay-off representation described by Equation (6).

We are now in a position to describe our general CLCP optimization problem.

3 GENERAL CONVEX LINEARLY CONSTRAINED OPTIMIZATION PROBLEM

Let there be two functions $\mathscr{R}$ and $\mathscr{L}$ where $\mathscr{R}$ plays the role of regularization and $\mathscr{L}$ expresses the loss. The regularization $\mathscr{R}$ tries to reduce the range of strategies for $\mathscr{P}_2$ the Learner and assumed to be a convex, monotonically increasing function of a given norm, depending on $\mathscr{P}_2$'s strategy vector. The loss function $\mathscr{L}$ measures a learning method specific approximation error, and is a convex, monotonically increasing function in the absolute value of all components of the approximation error vector.

Given these two functions we can now present our general optimization framework that can be parameterized by a matrix $\mathbf{G}$:

Definition 1 (Main optimization problem)**.**

$$\begin{array}{ll} \min & \mathscr{R}(\mathbf{w}) + C\mathscr{L}(\boldsymbol{\xi}) \\ w.r.t. & \mathbf{w} \in \mathbb{R}^n, \boldsymbol{\xi} \in \mathbb{R}^m \\ s.t. & \mathbf{G}\mathbf{w} = \mathbf{g} - \boldsymbol{\xi} \\ & \mathbf{w} \in \mathcal{W}, \end{array} \tag{10}$$

where $C > 0$ is a penalty parameter balancing between regularization and loss, $\mathbf{g}$ is a constant vector and $\mathcal{W}$ expresses general constraints independent of $\mathbf{G}$ on $\mathbf{w}$, e.g. nonnegativity. One may interpret this problem as regression, where the vector $\mathbf{g}$ is approximated based on the known matrix $\mathbf{G}$, and the possible error expressed by $\boldsymbol{\xi}$.

Take Linear Program Boosting (LPB) as an example, in which case we have an L_1 regularization on $\mathbf{w}$ giving us $\mathscr{R}(\mathbf{w}) = \|\mathbf{w}\|_1$, an L_1 norm constraint on the positive elements of the loss vector giving $\mathscr{L}(\boldsymbol{\xi}) = \|\boldsymbol{\xi}_+\|_1$, a $\mathbf{g}$ vector equal to $\mathbf{1}$, and a pay-off matrix $\mathbf{G}$ defined as $G_{ij} = y_i f_j(x_i)$. In this way, by simply redefining $\mathbf{G}$, $\mathscr{R}(\mathbf{w})$, $\mathscr{L}(\boldsymbol{\xi})$ and $\mathbf{g}$ we can arrive at different learning algorithms. We move onto an example using SVMs and structured output learning before presenting a table of different algorithms under the same optimization framework. Before proceeding we would like to make the following definitions. Let $\boldsymbol{\xi}_+ = \max(\mathbf{0}, \boldsymbol{\xi})$ be the positive elements of $\boldsymbol{\xi}$, and similarly $\boldsymbol{\xi}_- = \max(\mathbf{0}, -\boldsymbol{\xi})$ the negative elements of $\boldsymbol{\xi}$. Therefore, $\boldsymbol{\xi} = \boldsymbol{\xi}_+ - \boldsymbol{\xi}_-$ and $\|\boldsymbol{\xi}\|_1 = \|\boldsymbol{\xi}_+ + \boldsymbol{\xi}_-\| = \mathbf{1}'(\boldsymbol{\xi}_+ + \boldsymbol{\xi}_-)$.

3.1 ANOTHER BINARY CLASSIFICATION EXAMPLE: SUPPORT VECTOR MACHINES

The learning algorithm of Support Vector Machines (SVMs) is similar to the case of LPB except that it applies quadratic regularization as opposed to linear. Assume we have a binary classification problem. Furthermore given a function $\phi : \mathcal{X} \to \mathcal{H}_\phi$ that embeds the inputs into a Hilbert space $\mathcal{H}_X$ (similar to the embedding given in sub-section 2.2), the primal form of the SVM reads:

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{w}\|_2^2 + C\mathbf{1}'\boldsymbol{\xi} \\ \text{w.r.t.} & \mathbf{w} \in \mathcal{H}_\phi^*, \boldsymbol{\xi} \in \mathbb{R}^m \\ \text{s.t.} & y_i\phi(x_i)'\mathbf{w} \geq \mathbf{1} - \boldsymbol{\xi}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{array} \tag{11}$$

where $\mathcal{H}_X^*$ is the dual space of the Hilbert space $\mathcal{H}_X$. Placing this optimization problem in the form of our general linearly constrained optimization problem (10) we have regularization $\mathscr{R}(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|_2^2$ and loss $\mathscr{L}(\boldsymbol{\xi}) = \|\boldsymbol{\xi}\|_1$. The simplest pay-off $\mathbf{G}$ for this schema is $G_{ij} = y_i(\phi(x_i))_j$. We can reconstruct the constraint $y_i\phi(x_i)'\mathbf{w}$ into the general form of $\mathbf{G}$ we gave in sub-section 2.2 by embedding the outputs into the label space using the same idea behind Equation (8) and (9):

$$y_i\phi(x_i)'\mathbf{w} \Rightarrow \boldsymbol{\Gamma}(\psi(y_i), \mathbf{W}\phi(x_i)).$$

Applying the inner product based representation of $\boldsymbol{\Gamma}$ from Equation (7) we can derive a "vector" learning version of SVMs, called Maximum Margin Regression (MMR), see in [Szedmak et al., 2005] and [Astikainen et al., 2008]. By vector learning we mean the outputs are vectors as opposed to individual scalar values. This vector learning CLCP is given by:

$$\begin{array}{ll} \min & \frac{1}{2}\|\mathbf{W}\|_{\text{Frob}}^2 + C\mathbf{1}'\boldsymbol{\xi} \\ \text{w.r.t.} & \mathbf{W} : \mathcal{H}_\phi \to \mathcal{H}_\psi, \text{linear operator}, \\ & \boldsymbol{\xi} \in \mathbb{R}^m \\ \text{s.t.} & \langle\psi(y_i), \mathbf{W}\phi(x_i)\rangle \geq \mathbf{1} - \boldsymbol{\xi}, \\ & \boldsymbol{\xi} \geq \mathbf{0}, \end{array} \tag{12}$$

where $\|\cdot\|_{\text{Frob}}$ denotes the Frobenius norm of a matrix. We call this MMR_{base} and will use this CLCP in our experiments.

3.2 EXCHANGING THE ROLES OF LEARNER AND NATURE: LEARNING VIA DISTANCES

The linear programming primal and dual of the two-player zero-sum game (section 2) gives us a straightforward way of learning when the relationship between the input and the output objects are given by distances (dissimilarities) [Pekalska and Duin, 2005]. Looking at the next pair of primal and dual problems, a simple case of exchanging the roles of the Learner $\mathscr{P}_2$ to the primal, and Nature $\mathscr{P}_1$ to the dual retrieves the following "distance" learning scenario.

Learning similarity
G is gain

$\swarrow$ **Learner** $\downarrow$ Primal	$\searrow$ **Nature** $\downarrow$ Dual
min $\mathbf{1}'\mathbf{w} + C\mathbf{1}'\boldsymbol{\xi}$	max $\mathbf{1}'\boldsymbol{\alpha}$
w.r.t. $\mathbf{w} \in \mathbb{R}^n,\ \boldsymbol{\xi} \in \mathbb{R}^m,$	w.r.t. $\boldsymbol{\alpha} \in \mathbb{R}^m$
s.t. $\mathbf{G}\mathbf{w} + \boldsymbol{\xi} \geq \mathbf{1},$	s.t. $\mathbf{G}'\boldsymbol{\alpha} \leq \mathbf{1},$
$\mathbf{w} \geq \mathbf{0}, \boldsymbol{\xi} \geq \mathbf{0}$	$C\mathbf{1} \geq \boldsymbol{\alpha} \geq \mathbf{0},$
$\uparrow$ **Nature** $\nwarrow$	$\uparrow$ **Learner** $\nearrow$

G is loss
Learning dissimilarity

Reading of these primal and dual problems can be done like so: from top to bottom we are learning similarity and $\mathbf{G}$ is the payoff of the Learner $\mathscr{P}_2$. This is the scenario we have used throughout the paper (see section 2.1). Furthermore, we define *reference sets* $\hat{\mathcal{X}}$ and $\hat{\mathcal{Y}}$ that may be contained in or equal to $\mathcal{X}$ and $\mathcal{Y}$, respectively, such that there exists a pair of real valued functions $s_x : \mathcal{X} \times \hat{\mathcal{X}} \to \mathbb{R}$ and $s_y : \mathcal{Y} \times \hat{\mathcal{Y}} \to \mathbb{R}$ expressing similarity measures between the objects and there references, giving us the following $\mathbf{G}$:

$$\boxed{G_{ij} = s_\psi\left(\psi(y_i), \psi(y_j)\right) s_\phi\left(\phi(x_i), \phi(x_j)\right)}.$$

However, reading from bottom to top the role of the Learner is switched and solves the dual problem, with the pay-off $\mathbf{G}$ in this case changed to loss and the learning scenario being of a dissimilarity (distance) measure. This simple procedure of switching gives us a new learning algorithm in the framework of the MMR algorithm which we call $\text{MMR}_{\text{dissimilarity}}$. For example, in the dissimilarity case we can define $\mathbf{G}$ like so:

$$\boxed{G_{ij} = d_\psi\left(\psi(y_i), \psi(y_j)\right) d_\phi\left(\phi(x_i), \phi(x_j)\right)},$$

where d_ψ is a distance measure *e.g.*, Euclidean, in the label space and d_ϕ is some distance measure in the feature space.

For $\text{MMR}_{\text{dissimilarity}}$ we can predict the output $f(x)$ of an arbitrary $x \in \mathcal{X}$ by assuming that $\boldsymbol{\alpha}^*$ is an optimum solution of the dual problem and reversing the maximum margin approach suggested by Conjecture 1 into a minimum one:

$$f(x) = \arg\min_{\forall u \in \widetilde{\mathcal{Y}}} \sum_{j=1}^{m} \alpha_j^* d_\psi\left(\psi(y), \psi(y_j)\right) d_\phi\left(\phi(x_j), \phi(x)\right).$$

where, as before, u takes all possible output values from a properly chosen subset $\widetilde{\mathcal{Y}} \subseteq \mathcal{Y}$ (see section 2.2 for examples).

A list of other learning algorithms that can be posed under our framework have been described in Table 1. We give examples for classification, regression and structural learning. As mentioned earlier the pay-off matrices $\mathbf{G}$ all have the same dimensionality irrespective of the type of problem to be learnt.

4 EXPERIMENTS

The task of enzyme function prediction is a complex hierarchical learning problem where both the inputs and the outputs are structured objects; the inputs are proteins expressed via amino-acid sequences, and the outputs are paths in a directed acyclic graph – see a detailed description and the biological relevance of the problem in Astikainen et al. [2008]. The following real-world data sets were used:

- **EC** dataset is a sample of 5934 enzymes from the KEGG LIGAND database [Goto et al., 2002]. The EC hierarchy to be predicted has four levels plus root and is of size 1634 (1376 leaves, 258 internal nodes). In this version of the data, only a single function per enzyme is reported.
- **Gold Standard** dataset contains 3090 proteins which are classified into superfamily and family classes by their function [Brown et al., 2006]. The hierarchy to be predicted has two levels plus root and is of size 493 (487 families, 5 super-families).

Throughout this paper we have delayed the introduction of kernels [Schölkopf and Smola, 2002, Shawe-Taylor and Cristianini, 2004] as our work has primarily been based on feature representations and similarity measures. Similarity measures do not require certain restrictive properties such as positive semi-definiteness etc, as is the case with kernels. However, kernels can of course be applied in our framework and are an excellent choice for similarity due to the large depth of literature available on kernels for structured feature representations such as graphs, trees, texts, etc,. They also help avoid the need to explicitly compute the feature and label space representations. We now briefly describe the kernels we used for the input and output space.

Input feature representation: substring spectrum kernels (STR) [Lodhi et al., 2002], gap or mismatch kernels (GAP) [Leslie et al., 2003] and the so-called Alignment Trace Graph (GTG) kernels [Heger et al., 2007, 2003].

Output feature representation: The output is given by a rooted directed acyclic graph, where the leaves are related to types of chemical reactions catalyzed by the enzyme. The non-leave nodes represent hierarchical classes of the chemical reactions. All of the nodes can be identified by the shortest path, assuming equal edge weights, connecting them to the root node. The output labels are the indicators of the paths in the set of all nodes, so they can be given by a subset of the nodes. In this way the prediction provides a set of indicators of a possible path, these indicators are called microlabels in the sequel. The label vectors are given by

$$\psi_v(y) = \gamma^{d-1} [\![\text{node } v \text{ in the path to node } y]\!],$$

where $\gamma > 1$ and d denotes the depth of the node.

In our experiments we use this embedding with $\gamma = 10$.

We use two measures to characterize the performance of the learning approaches. The first is the usual 0/1 loss. The second measure is the microlabel F_1 score which is defined like so:

$$F_1 = \frac{2Prec}{Prec + Rec}$$

where $Prec = \frac{TP}{TP+FP}$ is *Precision* and $Rec = \frac{TP}{TP+FN}$ is *Recall* where TP, FP and FN denote the True Positive, False Positive and False Negative prediction rate.

Type of Learning	Learning Algorithm	$\mathscr{R}$	$\mathscr{L}$	$\mathbf{G}$	$\mathbf{g}$	$\mathcal{W}$
Classification	SVM	$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\|\boldsymbol{\xi}_+\|_1$	$G_{ij} = y_i(\phi(x_i))_j$	$g_i = 1$	
	Least-square SVM	$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\|\boldsymbol{\xi}\|_2^2$	$G_{ij} = y_i(\phi(x_i))_j$	$g_i = 1$	
	LPBoost	$\|\mathbf{w}\|_1$	$\|\boldsymbol{\xi}_+\|_1$	$G_{ij} = y_i f_j(x_i)$	$g_i = 1$	$\mathbf{w} \geq \mathbf{0}$
	LP Machine	$\|\mathbf{w}\|_1$	$\|\boldsymbol{\xi}_+\|_1$	$G_{ij} = y_i(\phi(x_i))_j$	$g_i = 1$	
Regression	Ridge Regression	$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\|\boldsymbol{\xi}\|_2^2$	$G_{ij} = (x_i)_j$	$g_i = y_i$	
	Lasso	$\|\mathbf{w}\|_1$	$\|\boldsymbol{\xi}\|_2^2$	$G_{ij} = (x_i)_j$	$g_i = y_i$	
Structured	MMR_{base}	$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\|\boldsymbol{\xi}_+\|_1$	$G_{ij} = (\phi(x_i) \otimes \psi(y_i))_j$	$g_i = 1$	
	$\text{MMR}_{\text{similarity}}$	$\frac{1}{2}\|\mathbf{w}\|_2^2$	$\|\boldsymbol{\xi}_+\|_1$	$G_{ij} = s_y(\psi(y_i), \psi(y_j)) s_x(\phi(x_i), \phi(x_j))$	$g_i = 1$	
	$\text{MMR}_{\text{dissimilarity}}$	$-\|\mathbf{w}\|_1$	$\|\boldsymbol{\xi}_-\|_1$	$G_{ij} = d_y(\psi(y_i), \psi(y_j)) d_x(\phi(x_i), \phi(x_j))$	$g_i = 1$	

Table 1: Several different algorithms that can be described under our framework.

The data sets we test against have been formed using a Nearest Neighbor approach and hence Nearest Neighbor forms a good baseline. Furthermore we also compare against a recently proposed algorithm for structural learning problems of this sort called the Hierarchical Max-Margin Markov algorithm (HM^3) [Rousu et al., 2006] which is closely related to the work of Taskar et al. [2003] and Tsochantaridis et al. [2005].

Table 2 presents the results using the MMR_{base} algorithm we presented earlier, Nearest Neighbor and HM^3. As we can see the MMR_{base} always works as well as or better than the Nearest Neighbor and HM^3 methods. We should point out that MMR_{base} took minutes to train whereas HM^3 took from several hours to days. This is because the complexity of the HM^3 algorithm, measured in terms of the number of constraints, grows exponentially with the number of nodes in the hierarchies. However, due to the formulation we gave with the pay-off matrix $\mathbf{G}$, we can achieve results in a time equivalent to solving a simple classification/regression problem.

5 EXTENSIONS AND DISCUSSION

We derived a general Convex Linearly Constrained Program (CLCP) and showed that several machine learning algorithms fit into this optimization framework parameterized by a matrix $\mathbf{G}$ containing all the information from the input-output paired samples. All other information required for the optimization problem is independent of the sample and can in some sense be viewed as our prior belief of the best way of tackling the learning problem. We showed that classification and regression algorithms could be placed under these frameworks, and furthermore gave detailed examples of structured output learning algorithms using this CLCP. By exchanging the role of the Learner and Nature we showed that we could learn using dissimilarity in the structured learning framework – hopefully convincing the reader that many different forms of learning algorithm could be created by changing the definition of matrix $\mathbf{G}$. We used the MMR_{base} formulation to help demonstrate the benefits of our new framework and gave experimental results on a real world enzyme prediction task – showing that we can achieve excellent results with an optimization problem utilizing the equivalent complexity of a classification/regression problem.

An open source implementation for the cases of the linear and quadratic regularization is available on the web.

Also, the structured learning algorithms of the MMR seem to work well and so we would like to bound the generalization error of these algorithms to prove their learnability under any distribution and hopefully show that the algorithms minimize such bounds.

References

K. Astikainen, L. Holm, E. Pitkänen, S. Szedmak, and J. Rousu. Towards structured output prediction of enzyme function. In *BMC Proceedings, 2(Suppl 4):S2*. 2008.

S.D. Brown, J.A. Gerlt, J.L. Seffernick, and P.C. Babbitt. A gold standard set of mechanistically diverse enzyme superfamilies. *Genome Biol*, 7(1):R8, 2006.

V. Chvatal. *Linear Programming.* W.H.Freeman, New York, 1983.

A. Demiriz, K. P. Bennett, and J. Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46:1:225–254, 2001.

R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification.* Wiley, 2001.

S. Goto, Y. Okuno, M. Hattori, T. Nishioka, and M. Kanehisa. LIGAND: database of chemical com-

EC: F1 score over all microlabel predictions with different kernel combinations combined with linear or degree 51 polynomial kernel

Sequence Kernel	Nearest neighbor (std)	MMR_{base} linear (std)	MMR_{base} poly-51 (std)	HM^3 poly-51 (std)
GTG	89.3	88.3 (0.9)	**89.4 (0.8)**	89.3 (0.8)
GTG+STR	**91.7**	90.0 (0.5)	**91.7 (0.4)**	**91.7 (0.4)**
GTG+GAP	**90.9**	86.0 (0.6)	**90.9 (0.3)**	**90.9 (0.3)**

EC: 0/1-loss over all microlabel predictions with different kernel combinations combined with linear or degree 51 polynomial kernel

Sequence Kernel	Nearest neighbor (std)	MMR_{base} linear (std)	MMR_{base} poly-51 (std)	HM^3 poly-51 (std)
GTG	16.8 (0.9)	18.6 (0.8)	**16.7 (0.9)**	**16.7 (0.9)**
GTG+STR	**14.2 (0.5)**	16.9 (0.5)	**14.2 (0.5)**	**14.2 (0.5)**
GTG+GAP	14.8 (0.6)	19.7 (0.6)	**14.8 (0.5)**	**14.8 (0.5)**

Gold Standard: F1 score and standard deviation over all microlabel predictions with GTG kernel

Sequence Kernel	Nearest neighbor (std)	MMR_{base} linear (std)	MMR_{base} poly-51 (std)	HM^3 linear (std)	HM^3 poly-51
GTG	88.0 (1.0)	81.9 (1.4)	89.3 (0.9)	**90.2 (0.8)**	89.6 (0.8)

Gold Standard: 0/1-loss over all microlabel predictions with GTG kernel

Sequence Kernel	Nearest neighbor (std)	MMR_{base} linear (std)	MMR_{base} poly-51 (std)	HM^3 linear	HM^3 poly-51
GTG	24.1 (1.9)	36.3 (2.8)	**21.4 (1.8)**	23.3 (2.0)	21.6 (1.7)

Table 2: Results of the experiments with best results in bold face.

pounds and reactions in biological pathways. *Nucleic Acids Research*, 30(1):402, 2002.

A. Heger, S. Mallick, C. Wilton, and L. Holm. The global trace graph, a novel paradigm for searching protein sequence databases. *Bioinformatics*, 23(18): 2361, 2007.

Andreas Heger, Michael Lappe, and Liisa Holm. Accurate detection of very sparse sequence motifs. In *RECOMB '03: Proceedings of the seventh annual international conference on Research in computational molecular biology*, pages 139–147, New York, NY, USA, 2003. ACM. ISBN 1-58113-635-8. doi: http://doi.acm.org/10.1145/640075.640094.

C. Leslie, E. Eskin, J. Weston, and W.S. Noble. Mismatch sring kernels for svm protein prediction. *Advances in Neural Information Processing Systems*, volume 15, 2003.

H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2: 419–444, 2002.

E. Pekalska and R.P.W. Duin. *The Dissimilarity Representation for Pattern Recognition. Foundations and Applications.* World Scientific, Singapore, 2005.

J. Rousu, C.J. Saunders, S. Szedmak, and J. Shawe-Taylor. Kernel-based learning of hierarchical multilabel classification models. *Journal of Machine Learning Research*, Special issue on Machine Learning and Large Scale Optimization, 2006.

R. Schapire. The boosting approach to machine learning: an overview. In *MRSI Workshop on Nonlinear Estimation and Control.* 2002.

B. Schölkopf and A. J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond.* MIT Press, Cambridge, MA, USA, 2002.

J. Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis.* Cambridge University Press, 2004.

S. Szedmak, J. Shawe-Taylor, and E. Parado-Hernandez. Learning via linear operators: Maximum margin regression. In *PASCAL Research Reports, http://eprints.pascal-network.org/.* 2005.

B. Taskar, C. Guestrin, and D. Koller. Max-margin markov networks. In *NIPS 2003.* 2003.

I. Tsochantaridis, T. Joachims, T. Hofmann, and Y. Altun. Large margin methods for structured and interdependent output variables. *Journal of Machine Learning Research (JMLR)*, 6(Sep):1453–1484, 2005.

V. Vapnik. *Statistical Learning Theory.* Wiley, 1998.