Proof and Implementation of Algorithmic Realization of Learning Using Privileged Information (LUPI) Paradigm: SVM+

Z. Berkay Celik*, Rauf Izmailov†, and Patrick McDaniel*

*Department of Computer Science and Engineering
The Pennsylvania State University
Email: {zbc102,mcdaniel}@cse.psu.edu

†Applied Communication Sciences, Basking Ridge, NJ, USA
Email: rizmailov@appcomsci.com

Abstract

Vapnik et al. recently introduced a new learning paradigm called Learning Using Privileged Information (LUPI). In this paradigm, along with standard training data, the teacher provides the student privileged (additional) information, yet not available at test time. The paradigm is realized by implementation of SVM+ algorithm. In this report, we give the proof of the SVM+ algorithm and show implementation details in MATLAB quadratic programming (`quadprog()`) function provided by the optimization toolbox. This allows us to obtain a vector $\vec{x}$ that minimizes the quadratic function.

I. Introduction

Vapnik et al. recently introduced Learning Using Privileged Information (LUPI) paradigm that aims at improving the predictive performance of learning algorithms and reducing the amount of required training data [1], [2], [3]. The algorithmic realization of LUPI paradigm is named as SVM+ algorithm. SVM+ allows generating a function $f(\cdot)$ that is learned from both standard training data and privileged information but requires only standard training data to make predictions at test time. The privileged information available at training time is utilized to reduce the error measure on the prediction.

SVM+ algorithm has recently attracted a lot of attention in machine learning community and started using in a few research areas. Recently, Heng et al. [4] applied the paradigm to

facial feature detection where head poses or gender available only in training time; Sharmanska et al. [5] employed to the image classification by providing additional description of images in training time. Hernandez-Lobato et al. treated learning with privileged information paradigm under the Gaussian process classification (GPC) framework [6]. Finally, Lopez-Paz et al. generalized the idea of distillation [7] by using privileged information and presented applications in semisupervised, unsupervised and multitask learning case studies [8].

We describe the goal of LUPI paradigm as follows. We can formally split the feature space into two spaces at training time: given L standard vectors $\mathcal{X}^{std} = \vec{x_1}, \ldots, \vec{x_L}$ and L privileged vectors $\mathcal{X}^* = \vec{x_1^*}, \ldots, \vec{x_L^*}$ with a target class $y = \{+1, -1\}$, where $\vec{x_i} \in \mathbb{R}^N$ and $\vec{x_i^*} \in \mathbb{R}^M$ for all $i = 1, \ldots, L$. Then, the goal is finding the best function $f \to \mathbf{x}^{std} : y$ such that the expected loss is:

$$R(\theta) = \int_{X^{std}} \int_{Y} L((\mathbf{x}^{std}, \theta), y) p(\mathbf{x}^{std}, \mathbf{x}^*, y) \, d\mathbf{x}^{std} \, dy \tag{1}$$

where $L(f(\mathbf{x}, \theta), y)$ is the loss function and θ is the parameters.

In this report, we give the proof of the SVM+ algorithm with two spaces in Section II and provide the implementation in MATLAB `quadprog()` function [9] in Section III.

II. Formulation of SVM+ Algorithm With Two Spaces

Given L standard vectors $\vec{x_1}, \ldots, \vec{x_L}$ and L privileged vectors $\vec{x_1^*}, \ldots, \vec{x_L^*}$ with a target class $y = \{+1, -1\}$, where $\vec{x_i} \in \mathbb{R}^N$ and $\vec{x_i^*} \in \mathbb{R}^M$ for all $i = 1, \ldots, L$. The kernels $K(\vec{x_i}, \vec{x_j})$ and $K^*(\vec{x_i^*}, \vec{x_j^*})$ are selected, along with positive parameters κ, γ. The two-spaces SVM+ optimization problem is formulated as

$$\begin{cases} \sum_{i=1}^{L} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{L} y_i y_j \alpha_i \alpha_j K(\vec{x_i}, \vec{x_j}) - \frac{\gamma}{2} \sum_{i,j=1}^{L} y_i y_j (\alpha_i - \delta_i)(\alpha_j - \delta_j) K^*(\vec{x_i^*}, \vec{x_j^*}) \to \max \\ \sum_{i=1}^{L} \alpha_i y_i = 0, \ \sum_{i=1}^{L} \delta_i y_i = 0 \\ 0 \le \alpha_i \le \kappa C_i, \ 0 \le \delta_i \le C_i, \ i = 1, \ldots, L \end{cases} \tag{2}$$

The decision rule f for vector $\vec{z}$ is as follows:

$$f(\vec{z}) = \text{sign} \left(\sum_{i=1}^{L} y_i \alpha_i K(\vec{x_i}, \vec{z}) + B \right), \tag{3}$$

where in order to compute B we first have to compute the Lagrangian of (2).

The Lagrangian of (2) has the form

$$\mathcal{L}(\vec{\alpha},\vec{\beta},\vec{\phi},\vec{\lambda},\vec{\mu},\vec{\nu},\vec{\rho}) = \sum_{i=1}^{L}\alpha_i - \frac{1}{2}\sum_{i,j=1}^{L} y_i y_j \alpha_i \alpha_j K(\vec{x_i},\vec{x_j}) - \\ \frac{\gamma}{2}\sum_{i,j=1}^{L} y_i y_j (\alpha_i-\delta_i)(\alpha_j-\delta_j) K^*(\vec{x_i^*},\vec{x_j^*}) + \\ \phi_1\sum_{i=1}^{L}\alpha_i y_i + \phi_2\sum_{i=1}^{L}\delta_i y_i + \sum_{i=1}^{L}\lambda_i\alpha_i + \sum_{i=1}^{L}\mu_i(\kappa C_i-\alpha_i) + \sum_{i=1}^{L}\nu_i\delta_i + \sum_{i=1}^{L}\rho_i(C_i-\delta_i) \tag{4}$$

with Karush-Kuhn-Tucker (KKT) conditions (for each $i = 1,\ldots,L$)

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial\alpha_i} = -K(\vec{x_i},\vec{x_i})\alpha_i - \gamma K^*(\vec{x_i^*},\vec{x_i}^*)\alpha_i + K^*(\vec{x_i^*},\vec{x_i^*})\gamma\delta_i - \sum\limits_{k\neq i} K(\vec{x_i},\vec{x_k}) y_i y_k \alpha_k - \\ \qquad \gamma\sum\limits_{k\neq i} K^*(\vec{x_i^*},\vec{x_k^*}) y_i y_k (\alpha_k-\delta_k)) + 1 + \phi_1 y_i + \lambda_i - \mu_i = 0 \\ \dfrac{\partial\mathcal{L}}{\partial\delta_i} = -K^*(\vec{x_i^*},\vec{x_i^*})\gamma\delta_i + K^*(\vec{x_i^*},\vec{x_i^*})\gamma\alpha_i + \sum\limits_{k\neq i} K(\vec{x_i},\vec{x_k}) y_i y_k \gamma(\alpha_k-\delta_k) + \phi_2 y_i + \nu_i - \rho_i = 0 \\ \lambda_i \geq 0, \quad \mu_i \geq 0, \quad \nu_i \geq 0, \quad \rho_i \geq 0, \\ \lambda_i\alpha_i = 0, \quad \mu_i(C_i-\alpha_i) = 0, \quad \nu_i\delta_i = 0, \quad \rho_i(C_i-\delta_i) = 0 \\ \sum\limits_{i=1}^{L}\alpha_i y_i = 0, \quad \sum\limits_{i=1}^{L}\delta_i y_i = 0 \end{cases} \tag{5}$$

We denote

$$F_i = \sum_{k=1}^{L} K(\vec{x_i},\vec{x_k}) y_k \alpha_k, \quad f_i = \sum_{k=1}^{L} K^*(\vec{x_i^*},\vec{x_k^*}) y_k (\alpha_k-\delta_k) \quad i = 1,\ldots,L \tag{6}$$

and rewrite (5) in the form

$$\begin{cases} \dfrac{\partial\mathcal{L}}{\partial\alpha_i} = -y_i F_i - \gamma y_i f_i + 1 + \phi_1 y_i + \lambda_i - \mu_i = 0 \\ \dfrac{\partial\mathcal{L}}{\partial\delta_i} = \gamma y_i f_i + \phi_2 y_y + \nu_i - \rho_i = 0 \\ \lambda_i \geq 0, \quad \mu_i \geq 0, \quad \nu_i \geq 0, \quad \rho_i \geq 0, \\ \lambda_i\alpha_i = 0, \quad \mu_i(C_i-\alpha_i) = 0, \quad \nu_i\delta_i = 0, \quad \rho_i(C_i-\delta_i) = 0 \\ \sum\limits_{i=1}^{L}\alpha_i y_i = 0, \quad \sum\limits_{i=1}^{L}\delta_i y_i = 0 \end{cases} \tag{7}$$

The first equation in (7) implies

$$\phi_1 = -y_j(1 - y_j F_j - \gamma y_j f_j + \lambda_j - \mu_j) \tag{8}$$

for all j. If j is selected such that $0 < \alpha_j < \kappa C_j$ and $0 < \delta_j < C_j$, then (7) implies $\lambda_j = \mu_j = \nu_j = \rho_j = 0$ and (8) has the form

$$\phi_1 = -y_j(1 - y_j F_j - \gamma y_j f_j) = -y_j \left(1 - \sum_{i=1}^{L} y_i y_j K(\vec{x}_i, \vec{x}_j)(\alpha_i) - \gamma \sum_{i=1}^{L} y_i y_j K^*(\vec{x_i^*}, \vec{x_j^*})(\alpha_i - \delta_i) \right) \tag{9}$$

Therefore, B is computed as $B = -\phi_1$:

$$B = y_j \left(1 - \sum_{i=1}^{L} y_i y_j K(\vec{x}_i, \vec{x}_j)(\alpha_i) - \gamma \sum_{i=1}^{L} y_i y_j K^*(\vec{x_i^*}, \vec{x_j^*})(\alpha_i - \delta_i) \right) \tag{10}$$

where j is such that $0 < \alpha_j < \kappa C_j$ and $0 < \delta_j < C_j$.

III. Matlab Quadratic Programming for SVM+ Algorithm With Two Spaces

Matlab function `quadprog`$(H, \vec{f}, A, \vec{b}, Aeq, \vec{beq}, \vec{lb}, \vec{ub})$ solves the quadratic programming (QP) problem in the form as follows:

$$\begin{cases} \dfrac{1}{2}\vec{z}^T H \vec{z} + f^T \vec{z} \to \min \\ A \cdot \vec{z} \le \vec{b} \\ Aeq \cdot \vec{z} = \vec{beq} \\ \vec{lb} \le \vec{z} \le \vec{ub} \end{cases} \tag{11}$$

Here, H, A, Aeq are matrices, and $\vec{f}, \vec{b}, \vec{beq}, \vec{lb}, \vec{ub}$ are vectors.

where $\vec{z} = (\alpha_1, \ldots, \alpha_L, \delta_1, \ldots, \delta_L) \in R^{2L}$. We now rewrite (2) in the form (11).
The first line of (11) corresponds to the first line of (2) when written as

$$\frac{1}{2}\vec{z}^T H \vec{z} + f^T \vec{z} \to \min$$

where

$$f = (-1_1, -1_2, \ldots, -1_L, 0_{L+1}, \ldots, 0_{2L})$$

and

$$H_{ij} = \begin{pmatrix} H^{11} & H^{12} \\ H^{12} & H^{22} \end{pmatrix}$$

where, for each pair $i, j = 1, \ldots, L$,

$$H_{ij}^{11} = K(\vec{x}_i, \vec{x}_j) y_i y_j + \gamma K^*(\vec{x}_i^*, \vec{x}_j^*) y_i y_j, \;\; H_{ij}^{12} = -\gamma K^*(\vec{x}_i^*, \vec{x}_j^*) y_i y_j, \;\; H_{ij}^{22} = +\gamma K^*(\vec{x}_i^*, \vec{x}_j^*) y_i y_j.$$

The second line of (11) is absent.

The third line of (11) corresponds to the second line of (2) when written as

$$Aeq \cdot \vec{z} = \vec{beq}$$

where

$$Aeq = \begin{pmatrix} y_1 & y_2 & \cdots & y_L & 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 & y_1 & y_2 & \cdots & y_L \end{pmatrix}, \quad \vec{beq} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

The fourth line of (11) corresponds to the third line of (2) when written as

$$\vec{lb} \leq \vec{z} \leq \vec{ub}$$

where

$$\vec{lb} = (0_1, 0_2, \ldots, 0_L, 0_{L+1}, \ldots, 0_{2L}),$$
$$\vec{ub} = (\kappa C_1, \kappa C_2, \ldots, \kappa C_L, C_1, C_2, \ldots, C_L)$$

After all variables of `quadprog`$(H, \vec{f}, A, \vec{b}, Aeq, \vec{beq}, \vec{lb}, \vec{ub})$ are defined as above, optimization toolbox guide [9] can be used to select `quadprog()` function options such as an optimization algorithm. Then, output of the function can be used in decision rule f for a new sample $\vec{z}$ to make predictions as presented in (3) as follows:

$$f(\vec{z}) = \text{sign}\left(\sum_{i=1}^{L} y_i \alpha_i K(\vec{x}_i, \vec{z}) + B\right) \tag{12}$$

IV. Conclusion

In this report, we give the proof of SVM+ algorithm and describe the implementation in MATLAB `quadprog()` function. Given standard training data and privileged information along with the selected kernel and parameters, the function returns the value of the objective function that can be used to make predictions on unseen samples by using standard training data.

REFERENCES

[1] Vladimir Vapnik and Akshay Vashist. A new learning paradigm: Learning using privileged information. *Neural Networks*, 2009.

[2] Dmitry Pechyony, Rauf Izmailov, Akshay Vashist, and Vladimir Vapnik. Smo-style algorithms for learning using privileged information. In *Proc. International Conference on Data Mining (DMIN)*, 2010.

[3] Vladimir Vapnik and Rauf Izmailov. Learning with intelligent teacher: similarity control and knowledge transfer. In *Statistical Learning and Data Sciences*. Springer, 2015.

[4] Heng Yang and Ioannis Patras. Privileged information-based conditional regression forest for facial feature detection. In *Proc. International Conference on Automatic Face and Gesture Recognition*. IEEE, 2013.

[5] V. Sharmanska, N. Quadrianto, and C. H. Lampert. Learning to rank using privileged information. In *Proc. International Conference on Computer Vision (ICCV)*. IEEE, 2013.

[6] Daniel Hernández-Lobato, Viktoriia Sharmanska, Kristian Kersting, Christoph H Lampert, and Novi Quadrianto. Mind the nuisance: Gaussian process classification using privileged noise. In *Advances in Neural Information Processing Systems*, pages 837–845, 2014.

[7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[8] David Lopez-Paz, Léon Bottou, Bernhard Schölkopf, and Vladimir Vapnik. Unifying distillation and privileged information. *arXiv preprint arXiv:1511.03643*, 2015.

[9] MATLAB documentation of quadprog() function. http://www.mathworks.com/help/optim/ug/quadprog.html. [Online; accessed 19-December-2015].