Template-SVM Machine Learning toolkit: Documentation and Usage

Shantanu Chakrabartty SHANTANU@WUSTL.EDU
Department of Electrical and Systems Engineering
Washington University in St. Louis
Saint Louis, MO 63130, USA

Version: 1.3

Abstract

This document briefly describes the template-SVM training and inference model that is implemented using a MATLAB© based *GiniSVMMicro* toolkit. The theory and details of the template-SVM and GiniSVM formulations can be found in the references and we briefly summarize the formulation in this manuscript for the sake of completeness. The key advantage of template-SVM compared to traditional SVMs and other shallow-learning methods are: (a) The number of template/support vectors is fixed and specified by the user; (b) learning is based on a maximal-margin training procedure which inherits the stability and generalization properties of traditional SVMs; (c) the training complexity in terms of memory usage is significantly lower than that of traditional SVM; (d) the framework can incorporate non-positive definite kernels and functions while maintaining the convexity in training; (e) the formulation can find solutions using user specified hardware-friendly kernels, as a result the toolkit could be used for design space explorations; and (f) the formulation can incorporate functions that cannot be described in closed form, for instance functions described by the solution to an ordinary differential equation (ODE).

Keywords: Support vector machines, Template-SVM, *Gini*-SVM, Shallow networks, Random projections, Basis pursuit methods.

1. Template-SVM Model: Inference and Training

The mathematical model for a template-SVM is similar to other shallow-learning networks like random projection networks using a set of M inference functions $f_k : \mathbb{R}^D \rightarrow \mathbb{R}$, each corresponding to $k = 1, .., M$ classes/categories. The inference functions $f_k(\mathbf{x})$ produce an output based on the input $\mathbf{x} \in \mathbb{R}^D$ according to

$$f_k(\mathbf{x}) = \sum_{p=1}^{P} W_{pk}\Phi_p(\mathbf{x}) + b_k \tag{1}$$

where $W_{pk} \in \mathbb{R}; p = 1, .., P$ are the weights corresponding to the k^{th} class and the P basis functions $\Phi_p : \mathbb{R}^D \rightarrow \mathbb{R}$. $b_k \in \mathbb{R}$ corresponds to the bias for the k^{th} decision function.

A template-SVM learning framework seeks to find W_{pk} that is a linear superposition of the basis functions defined over N data points $\mathbf{x}_i, i = 1, .., N$ as

$$W_{pk} = \sum_{i=1}^{N} \alpha_{ik}\Phi_p(\mathbf{x}_i) \tag{2}$$

but the coefficients α_{ik} have a balanced and bounded mathematical structure as

$$-1 \leq \alpha_{ik} \leq 1 \tag{3}$$

and

$$\sum_{i=1}^{N} \alpha_{ik} = 0 \tag{4}$$

$$\sum_{k=1}^{M} \alpha_{ik} = 0 \tag{5}$$

Under the constraint 2 the decision function 1 becomes

$$f_k(\mathbf{x}) = \sum_{i=1}^{N} \alpha_{ik} \sum_{p=1}^{P} \Phi_p(\mathbf{x}_i)\Phi_p(\mathbf{x}) + b_k \tag{6}$$

$$= \sum_{i=1}^{N} \alpha_{ik} K(\mathbf{x}_i, \mathbf{x}) + b_k \tag{7}$$

where $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}$ is a positive-definite kernel function satisfying the Mercer's criterion. Note the decision function given by 7 resembles a traditional SVM decision function. However, unlike the traditional SVMs template-SVMs uses the basis functions $\Phi_p(.)$ instead of the kernel functions $K(.,.)$ for training.

For training the template-SVM, each of the N data points $\mathbf{x}_i, i = 1, .., N$ are associated with a label probability vector $Y_i \in \mathbb{R}_+^M$ such that

$$0 \leq Y_{ik} \leq 1 \tag{8}$$

$$\sum_{k=1}^{M} Y_{ik} = 1. \tag{9}$$

Thus, Y_{ik} signifies a-priori conditional probability estimates that the i^{th} data-point belongs to the k^{th} class/category.

In this case, one way to satisfy the constraints 5 is by using another conditional probability variable P_{ik} as

$$\alpha_{ik} = Y_{ik} - P_{ik} \tag{10}$$

with

$$0 \leq P_{ik} \leq 1 \tag{11}$$

$$\sum_{k=1}^{M} P_{ik} = 1. \tag{12}$$

The variable P_{ik} signifies the conditional probability generated by template-SVM that the i^{th} data-point belongs to the k^{th} class/category.

To estimate P_{ik} based on the training data, the template-SVM follows the multi-class *Gini*-SVM training procedure Chakrabartty (2007) and involves optimizing a maximum-entropy function (comprising of a quadratic kernel divergence function and a *Gini* entropy function) as

$$\min_{P_{ik}} \frac{1}{2} \sum_{k=1}^{M} \left[\sum_{i=1}^{N} \sum_{j=1}^{N} K(\mathbf{x}_i, \mathbf{x}_j) \left(Y_{ik} - P_{ik}\right) \left(Y_{jk} - P_{jk}\right) + \gamma \sum_{i=1}^{N} P_{ik}^2 \right] \tag{13}$$

subject to

$$\frac{1}{N} \sum_{i=1}^{N} P_{ik} = \frac{1}{N} \sum_{i=1}^{N} Y_{ik}. \tag{14}$$

Note that the constraint is similar to the linear constraints used in conventional maximum-entropy function which equates empirical expectations with respect to two different probability distributions.

Instead of storing the entire kernel matrix, the memory requirements for the template-SVM training can be significantly reduced by exploiting the constraint 2 as

$$\min_{W_{pk}, P_{ik}} \frac{1}{2} \sum_{k=1}^{M} \left[\sum_{p=1}^{P} W_{pk}^2 + \gamma \sum_{i=1}^{N} P_{ik}^2 \right] \tag{15}$$

subject to

$$W_{pk} = \sum_{i}^{N} \left(Y_{ik} - P_{ik}\right) \Phi_p(\mathbf{x}_i) \tag{16}$$

$$\frac{1}{N} \sum_{i=1}^{N} P_{ik} = \frac{1}{N} \sum_{i=1}^{N} Y_{ik}. \tag{17}$$

Note that instead of storing $N \times N$ kernel evaluations, template-SVM stores only $P \times N$ basis function evaluations. The size of the quadratic program (QP) in (15) has increased to $(P+N) \times (P+N)$ compared to $N \times N$ size QP(13), however, the QP in (15) is separable for which several efficient algorithms have been reported.

Like the *Gini*-SVM formulation, template-SVM also directly produces conditional probability estimates according to a margin-propagation (MP) based normalization Chakrabartty (2007) as

$$P_k(\mathbf{x}) = \frac{1}{\gamma} \left[f_k(\mathbf{x}) - z(\mathbf{x}) \right]_+ \tag{18}$$

where

$$z(\mathbf{x}) : \sum_{k=1}^{M} P_k(\mathbf{x}) = 1. \tag{19}$$

The mathematical and robustness properties of MP-based normalization is described in detail in Chakrabartty (2007). Extensions of the *Gini*-SVM training that could be applied to template-SVM can be found in Gangopadhyay (2017) and Chakrabartty (2004).

1.1 Basis function selection

One of the key steps in designing a template-SVM is choosing an appropriate set of basis functions $\Psi_p(.)$. The basis functions not only incorporates our a-priori knowledge about the machine learning problem but a careful choice of $\Psi_p(.)$ can also reduce the computational complexity for template-SVM inference. For instance, in Kumar (2019) the basis functions were chosen to be a hardware friendly mapping implemented by an array or memristors. In other applications $\Psi_p(\mathbf{x})$ could be a set of P filters that acts on the input vector $\mathbf{x}$. However, a most general form for basis functions (implemented in the default version of the toolkit) is of the type

$$\Psi_p(\mathbf{x}) = \Phi(\mathbf{z}_p, \mathbf{x}) \tag{20}$$

where $\Phi : \mathbb{R}^D \times \mathbb{R}^D \rightarrow \mathbb{R}$ is a potential function that measures the similarity between the vector $\mathbf{x}$ with P template vectors $\mathbf{z}_p, p = 1, .., P$. One can view the template vectors as support vectors, except that $\Phi(., .)$ need not be a positive-definite kernel function.

2. Template-SVM Toolkit

The basic version of template-SVM training and inference has been implemented in MATLAB© and provides the ability to the user to optimize the architecture to different learning problems and to different hardware constraints. One of the attractive aspect of template-SVM is that the training and the inference hardware are similar (in terms of storage and precision), as a result the framework can be used for online training on computationally limited platforms.

2.1 Installation and Directory Structure

The MATLAB version of the template-SVM toolkit is a zipped file *GiniSVMMicrov1.3.zip*. In the directory of your choice unzip the the zipped file. This should create the following directories *templateSVM, data, docs* under the folder *GiniSVMMicrov1.3*. The scripts will use the *data* directory to store all the intermediate training and cross-validation files. The folder *GiniSVMMicrov1.3/templateSVM* should have the following scripts that can be modified by the user:

- *Training.m* : This is the main script that the user can specify different template-SVM training parameters (M,N,P) and the basis function parameters. The user can also specify the location of the training/test data and the location where the performance data will be stored.

- *Inference.m* : This is the main script that shows the user how to use templateSVM for inference. The script loads the stored the trained parameter and the test data from a user-defined location and then generates the performance data.

- *Potential.m* : The script defines different types of potential functions.

Other scripts in the *templateSVM* directory should be modified by advanced user. The documentation for each of the script is embedded in the script itself.

2.2 Input Data formats

The toolkit assumes that all the training and test data are stored in the *data* directory. The scripts *Training.m* and *Inference.m*, loads the data from the default locations *../data/Trainingdata.mat* and *../data/Testdata.mat*. Please change these locations to load data from different repositories.

- trainx - input data matrix (number of training data $\times$ Dimension)
- Ytrain - input label matrix (number of training data $\times$ total classes)
- testx - test data matrix (number of test data $\times$ Dimension)
- Ytest - test label matrix (number of test data $\times$ Dimension)

The training and cross-validation labels (Ytrain and Ytest) should be prior probabilities. An example for a three class label is [0 0 1] to indicate that the training label belongs to class 3. Or it could be [0.1 0.3 0.6] to indicate prior confidence There are a few sample data repositories that have been provided in the zipped folder. The naming convention for these repositories are *data(X)class(Y)D.mat* where X denotes the number of categories/classes and Y denotes the number of dimension for the input data. The training and inference scripts stores the training/cross-validation results along with the confusion matrices in *data/TrainingresulttSVM.mat*. The trained parameters are stored in the file *data/TrainingparamtSVM.mat*

2.3 Hyperparameters

There are three parameters defined in the script *Training.m* that users can use to control the properties of the template-SVM.

- GAMMA - This parameter (same as γ in the mathematical model) controls the generalization ability of the template-SVM. If a smaller value of GAMMA is chosen, it will try to overfit to the training data. Note that choosing a smaller GAMMA increases the training time. See the reference Chakrabartty (2007) for details.
- KTYPE - This specifies the type of Potential function as defined in *Potential.m*. Currently, there are three types defined: KTYPE = 1 or Cauchy potential - $\Psi_p(\mathbf{x}) = \left(1 + \frac{||\mathbf{z}_p - \mathbf{x}||^2}{\sigma}\right)^{-1}$; KTYPE = 2 or Gaussian potential - $\Psi_p(\mathbf{x}) = \exp\left(-\frac{||\mathbf{z}_p - \mathbf{x}||^2}{\sigma}\right)$; and KTYPE = 3 or User defined potential.
- KSCALE - This specifies the parameter (σ) of the Potential function as defined in *Potential.m*. For the Cauchy and Gaussian potential functions, SCALE determines the area of influence of around the template vectors $\mathbf{z}_p$.
- Ntem - This specifies the number P of template vectors or equivalently basis functions that the template-SVM should use. This parameter is important because it determines the memory usage and computational complexity of the template-SVM. Therefore Ntem should be chosen either based on resource constraints or should be selected after a cross-validation procedure.

- PerTrain - This specifies the percentage of the data used for training; and the remaining data is used for cross-validation. If the number of training points is limited, then consider using a leave-one-out procedure. Use the script Training.m to train a template-SVM and to evaluate the performance on the cross-validation set. You could use a 3-fold cross-validation procedure and adjust the hyper-parameters (GAMMA,KSCALE,Ntem) to balance the training and the cross-validation errors.

References

A. Gangopadhyay, O. Chatterjee, and S. Chakrabartty, Extended polynomial growth transforms for design and training of generalized support vector machines. IEEE transactions on neural networks and learning systems, 29(5):1961–1974, 2017.

S. Chakrabartty and G. Cauwenberghs, "Gini-Support Vector Machine: Quadratic Entropy Based Multi-class Probability Regression", Journal of Machine Learning Research, Volume 8, pp. 813-839, April 2007.

P. Kumar, A. R. Nair, Oindrila Chatterjee, T. Paul, A. Ghosh, S. Chakrabartty, and C. S. Thakur, "Neuromorphic in-memory computing framework using memtransistor cross-bar based support vector machines", CoRR, abs/1903.12330, 2019.

S. Chakrabartty, "Design and Implementation of Ultra-low-power Pattern Recognizers and Sequence Decoders", Ph.D. Dissertation, Electrical and Computer Engineering, The Johns Hopkins University, 2004.