A Combined Semi-supervised and Transfer Learning Support Vector Machine for Mental Health Data

Hongzhe Zhang
hongzhe.zhang@pennmedicine.upenn.edu

Samprit Banerjee
sab2028@med.cornell.edu

September 2022

Abstract

Semi-supervised learning methods aim to improve the generalization accuracy of supervised models with unlabelled data. However, they still assume the unlabelled samples are from the same distribution as the labelled ones. In this paper, we present a novel supervised algorithm Semi Transfer Support Vector Machines (ST-SVM) which can learn from labelled and unlabelled data from multiple problem domains. We also show the non-convex optimization of ST-SVM can be solved by a finite iterations of disciplined convex optimization (DCP) with concave-convex procedure. The optimization problem is further simplified as a quadratic programming problem in the appendix. The performance of ST-SVM is compared to traditional SVMs on a smartphone dataset collected by Weill Cornell ALACRITY Center.

1 Introduction

Supervised learning models refer to machine learning models that extract dependencies and interactions from the training data, and use them to infer future outcomes. As the name ”supervised” implies, labels serve as teachers, and they needs to be present for the models to learn. Traditional supervised learning methods are trained on pairs of labels and features. Labelled observations, however, are often difficult, expensive or sometimes impossible to obtain as they typically require efforts of human annotators or simply immeasurable. Unlabelled data on the other hand, is relative easier to collect. Unsupervised methods try to find regular patterns in the unlabelled data. Without external teachers, it is their sole responsibility to define and locate the patterns. The patterns can serve as descriptions or even new features of the data, but they are reflective only of the training data. When the training data of a prediction problem is only partially labelled, semi-supervised learning methods can improve upon traditional supervised learning methods by assuming various patterns for the label and unlabelled data.

On the other hand, both supervised and semi-supervised methods assumes that the training and future testing data are drown from the same distribution. When the distribution changes, new data needs to be collected, and the model needs to be rebuilt. When the tasks on the new and the old data are related, it is desirable to transfer the knowledge from the original supervised or semi-supervised model. This type of "transfer learning" can be useful when training data in certain problem domains is scarce. Consider a movie recommendation system that is attempting to predict whether its users will like an unseen movie. As each user has different preference, an user-specific model is desirable. Transfer learning methods can aid the predictions for inactive user with few liked/disliked movie with knowledge learnt from all other users. However, the unlabelled cases which are the movies users have seen but did not rate can not be utilized.

We propose a novel semi-supervised method for transfer learning called semi-transfer support vector machine (ST-SVM). ST-SVM can be trained on multiple tasks simultaneously using labelled and unlabelled data drawn from different distributions. It can learn task-specific hyperplanes using task–coupling parameters as in [1], and its non-linear generalizations can be easily derived with reproducing kernel Hilbert spaces (RKHS)[2]. Although the ST-SVM optimization problem is not convex, we show that it can be solved by the concave-convex procedure (CCCP) by adapting the optimization procedures in [3]. The procedures are guaranteed to find at least a local minimum in a finite number of iteration steps. We demonstrate the performance of our algorithm in an inductive transfer learning fashion with mobile health data collected by the app HealthRhythm redreference needed.

1.1 Related Methods

Suppose we have several predictions problems, each with labelled or unlabelled data. We can categorize candidate methods by which part of the data they can learn from. To better define the categories, following [14], we define "domain" as a pair contains the feature space and a marginal probability distribution, $\{X, P(X)\}$. Its "task" is defined as a pair consists of the label space and the conditional distribution $\{Y, P(Y|X)\}$. Each of the problems can then be characterized by a domain and a task.

Inductive Transfer Learning If we label each of the problems by either "target" or "source", transfer learning then aims to improve the learning of target tasks with the knowledge from both target and source problems. Out problem falls in inductive transfer learning(T-TL) where both target and source labels are available. When no labelled data is present for the source problems, one can utilize self-taught learning firstly proposed by [4]. Its idea is to learn basis vectors using the source domain data, and then use them to represent the target domain data. Similarly, [5] proposes to find a low-dimensional feature representation using labelled source domain data. Researchers have also generalized traditional statistical learning methods into the T-TL setting. T. Evgeniou and M. Pontil [1] borrows ideas from A. Schwaighofer et al.[6] and proposes an SVM which can learn from multiple domains simultaneously by separating model weights into shared terms and task-specific terms. This gives a different decision boundary to each task, but all share a certain amount of common knowledge. Dai et al.[7] extended the AdaBoost [8] algorithm to address the inductive transfer learning problems. To the best of our knowledge, no method has been proposed to learn from partially-labelled source domains.

Semi-supervised Learning Semi-supervised learning focuses on learning from unlabelled data separately for each task. They make model assumptions on relationships between unlabelled and labelled data, decision boundary, etc. [9]. When the assumptions are met, semi-supervised methods can yield great improvement. Frameworks such as self-training [9] can be applied to most supervised learning methods. It assumes that the predictions of the model, at least the confident ones, tend to be correct. In each iteration of self-training, the unlabelled observations with the most confident unlabeled are appended to the training set. The process continues until there is no unlabelled observations. Traditional supervised algorithms are also modified to learn from unlablled data. Semi-supvervised (transductive) support vector machine (S3VM)[2] is a method of improving the generalization accuracy of SVM [10] by using unlabeled data. It assumes that the true decision boundary lies in a region of low observation density (so-called cluster assumption [11]), so that it forces the estimated decision boundary to be far away from unlabelled data. However, unlike self-taught learning, semi-supervised models assume the domains and tasks are the same for label and unlabelled data.

2 Method

The method section is divided into two part. In the first part, we motivate the objective function of ST-SVM by presenting to readers the single-task SVM, semi-supervised (transductive) SVM, transfer learning SVM (we call this TSVM, and it was originally named as regularized multi–task learning in [1]).

In this paper, it is assumed that the prediction problem consists of one source and one target domain and task. We denote the source and target by S, T, and we denote the partially labelled data in each of the domains by $(x_1, y_1), ..., (x_{l_t}, y_{l_t}), x_{l_t+1}, ..., x_{l_t+u_t}$, where $x_i = (x_{i1}, x_{i2}, ..., x_{ip})$, $y_i \in \{-1, 1\}$ for $t \in \{S, T\}$. Without loss of generality, the objective is set to achieving high performance for T.

2.1 Support Vector Machine

2.1.1 Formulation

One way to accomplish that is to train a single task SVM in T. In binary classification problems, when two classes are linearly separable, optimal separating hyperplane performs classification by finding the hyperplane that maximizes the distance to the closest point from either class. SVM generalizes it to the problems when two classes overlap. While the SVM is still maximizing the distance, SVM allows for some point to "slack" which are also minimized. The optimization problem can be written as

$$\begin{aligned} &\underset{\beta,\beta_0}{\text{minimize}}\frac{1}{2}\left\|\beta\right\|^2 + \lambda\ \Sigma_{i=1}^{l_T}\xi_i \\ &\text{subject to } \xi_i \geq 0,\ y_i(x_i^T\beta+\beta_0) \geq 1-\xi_i, i=1,...,l_T \end{aligned} \tag{1}$$

Where β and β_0 are p and 1 dimensional vector which define the hyperplane $\{x : f(x) = x^T\beta + \beta_0\}$, and the classification rule induced by $f(x)$ is $\hat{y} = sign(x^T\beta + \beta_0)$. The length of the margin between the hyperplane and the closest points is equal to $\frac{1}{||\beta||}$, which is set to be 1 when deriving (1). The slack variables ξ_i can then be interpreted as the overlap for observation i from the margin in distance.

2.1.2 Generalization to Non-linear Classification

If the classes are not naturally separable in the original feature space, one can enlarge it by transformations. For SVMs, one can perform the 'kernel trick' and avoid direct computations with high dimensional vectors in transformed feature space. This techniques is introduced in this section, and its connections with reproducing kernel Hilbert space will be pointed out. Although the derivations are carried out in the regular SVM environment for simplicity, all conclusions are generalizable to all methods introduced in the following sections.

Computing the SVM We look further into the optimization problem of SVM for the purpose of this section. By introducing the Lagrange variables $\boldsymbol{\alpha}$ and $\boldsymbol{\mu}$, the Lagrange primal problem of (1) can be written as

$$\begin{aligned} \mathcal{L}_p(\beta,\beta_0,\boldsymbol{\xi}) =&\frac{1}{2}\left\|\beta\right\|^2 + \lambda\ \Sigma_{i=1}^{l_T}\xi_i - \Sigma_{i=1}^{l_T}\alpha_i[y_i(x_i^T\beta+\beta_0)-(1-\xi_i)] \\ &-\Sigma_{i=1}^{l_T}\mu_i\xi_i \end{aligned} \tag{2}$$

Setting the derivative of the respective Lagrange variables to 0, we get

$$\beta = \Sigma_{i=1}^{l_T}\alpha_i y_i x_i \tag{3}$$

$$0 = \Sigma_{i=1}^{l_T}\alpha_i y_i \tag{4}$$

$$\alpha_i = \lambda - \mu_i \tag{5}$$

By substituting equations (3-5) into (2), we obtain the Lagrangian Wolfe dual objective function and the solution to function $f(x)$

$$\mathcal{L}_D(\boldsymbol{\alpha}) = \Sigma_{i=1}^{l_T}\alpha_i - \frac{1}{2}\Sigma_{i=1}^{l_T}\Sigma_{i'=1}^{l_T}\alpha_i\alpha_{i'}y_iy_{i'}x_ix_{i'} \tag{6}$$

$$f(x) = \Sigma_{i=1}^{l_T}\alpha_iy_ix_ix \tag{7}$$

Reproducing kernel Hilbert spaces Given any kernel $k(x, x^{'})$, we can construct a Hilbert space where the kernel k is a dot product. We then associates a function $k(\cdot, x)$ to every point in this space, with the reproducing kernel map $\Phi = x \to k(\cdot, x)$. This gives us a space which contains all linear combinations of functions that can be written as $f(\cdot) = \Sigma_{i=1}^{n}k(\cdot\ , x_i)$, and this is our RKHS. We further define the function g as $g(\cdot) = \Sigma_{i'=1}^{m}k(\cdot\ , x_{i'})$. The inner product of this space is then defined by

$$\langle f, g\rangle = \Sigma_{i=1}^{n}\Sigma_{i'=1}^{m}\alpha_i\alpha_{i'}k(x_i, x_{i'})$$

The kernel trick and its link to RKHS If we write (7) directly in terms of transformed feature vectors $h(x)$, we can get

$$f(\cdot) = \Sigma_{i=1}^{n}\alpha_iy_ik(\cdot\ , x_i) \tag{8}$$

Since kernels $k(\cdot\ , x_i)$ spans the feature space, and $f(x)$ is a linear combination of x_i, we can conclude that $f(x)$ belongs to the Hilbert space. The dot product we defined in this space (also known as the reproducing property of kernels) implies that

$$f(x) = \Sigma_{i=1}^{l_T}\alpha_iy_i\langle h(x), h(x_i)\rangle \tag{9}$$

$$= \Sigma_{i=1}^{l_T}\alpha_iy_iK(x, x_i) \tag{10}$$

This means we need not specify the transformations $h(\cdot)$ at all, and it requires only the knowledge of the kernel function K. This is the "kernel trick".

2.2 Semi-supervised Support Vector Machine

One can also complete this task in a semi-supervised fashion with S3VM by making use of the unlabelled data in target domain. S3VM explicitly searches the hyperplane in feature space where the observation density is low by adding one extra term into the loss function of regular SVM.

Consider an unlabelled instance x, let the prediction of this instance be $\hat{y} = sign(f(x))$, we write the unlabelled "slack" of x as:

$$
\begin{aligned}
\xi &= max(1-\hat{y}(x^T\beta+\beta_0),0)\\
&= max(1-sign(x^T\beta+\beta_0)(x^T\beta+\beta_0),0)\\
&= max(1-|x^T\beta+\beta_0|,0)
\end{aligned}
\tag{11}
$$

If we add the slacks of all unlabelled observations to the optimization problem (1), we get (11). Note that adding $max(1-|\cdot|,0)$ to the objective is the same as adding ξ to the objective function and adding $|\cdot| \geq 1-\xi$, $\xi \geq 0$ to the constraints.

$$
\begin{aligned}
&\underset{\beta,\beta_0}{\text{minimize}}\frac{1}{2}\left\|\beta\right\|^2+\lambda_1\Sigma_{i=1}^{l_T}\xi_i+\lambda_2\Sigma_{j=l_T+1}^{l_T+u_T}\xi_j \qquad (12)\\
&\text{subject to } \xi_i \geq 0,\\
&y_i(x_i^T\beta+\beta_0) \geq 1-\xi_i, i=1,...l_T\\
&|(x_i^T\beta+\beta_0)| \geq 1-\xi_i, i=l_T+1,...,l_T+u_T
\end{aligned}
$$

This minimization objective increases when the separating hyperplane approaches the unlabelled observations from either side. Therefore, it encourages the hyperplane to go through regions in feature space where there is fewer unlabelled observations.

Empirically, to minimize the unlabelled term, S3VM can converge to trivial solutions where hyperplanes are in regions with no observations at all. In this case, all observations would be predicted as -1 or 1. To avoid this, the optimization is often constrained by restricting class proportions for the unlabelled and labelled instances to be relatively the same . The optimization problem (12) becomes

$$
\begin{aligned}
&\underset{\beta,\beta_0}{\text{minimize}}\frac{1}{2}\left\|\beta\right\|^2+\lambda_1\Sigma_{i=1}^{l_T}\xi_i+\lambda_2\Sigma_{j=l_T+1}^{l_T+u_T}\xi_j \qquad (13)\\
&\text{subject to } \xi_i \geq 0,\\
&y_i(x_i^T\beta+\beta_0) \geq 1-\xi_i, i=1,...l_T\\
&|(x_i^T\beta+\beta_0)| \geq 1-\xi_i, i=l+1,...,l_T+u_T\\
&\frac{1}{u_T}\Sigma_{j=l_T+1}^{l_T+u_T}\beta^T x_j+\beta_0=\frac{1}{l_T}\Sigma_{j=1}^{l_T}y_j
\end{aligned}
$$

The trade off between λ_1 and λ_2 stands for the trade off between the contribution of the loss between the labelled data and unlabelled data to the loss. The higher the ratio $\frac{\lambda_1}{\lambda_2}$, the more (13) leans towards the slacks for the labelled observations. When $\frac{\lambda_1}{\lambda_2}$ tends to infinity, (13) reduces to solving a regular SVM problem. Practically speaking, one can choose λ_1 and λ_2 with cross-validation or held-out validation data.

On the other hand, this is not a convex problem. To see this, we rewrite problem (13) in to (14) shown below.

$$
\begin{aligned}
&\underset{\beta,\beta_0}{\text{minimize}}\frac{1}{2}\left\|\beta\right\|^2 + \lambda_1\Sigma_{i=1}^{l_T} max(1 - y_i(x_i^T\beta + \beta_0), 0)\\
&+ \lambda_2\Sigma_{j=l_T+1}^{l_T+u_T} max(1 - |x_j^T\beta + \beta_0|, 0)\\
&\text{subject to } \frac{1}{u_T}\Sigma_{j=l_T+1}^{l_T+u_T}\beta^T x_j + \beta_0 = \frac{1}{l_T}\Sigma_{j=1}^{l_T} y_j
\end{aligned}
\tag{14}
$$

Consider a function composition $f(x) = h(g_1(x_1), ..., g_k(x_k))$. Assuming h and g are twice differentiable, it is easy to derive that f is concave if h is convex, h is non-decreasing in each argument, and g_i are concave. This tells us the third term of (14) is concave. In [3], the authors proposed to use concave convex procedure to solve this problem. We will go into the details of this algorithm in the ST-SVM section.

2.3 Transfer Learning Support Vector Machine

Up to now, only the data in the target domain has been used. One can apply TSVM to incorporate the labelled data from source domain into model fitting. Before TSVM, Schwaighofer et al.[9] proposed to use a hierarchical Bayesian framework which tries to learn parameters of multiple tasks simultaneously by letting them share a Gaussian process prior distribution. TSVM borrows this idea and decomposes the hyperplane slope in a way such that

$$\boldsymbol{\beta}^S = \beta^0 + \beta^S \quad and \quad \boldsymbol{\beta}^T = \beta^0 + \beta^T$$

Intuitively, this implies $\beta^t, t \in \{S, T\}$ comes from a particular probability distribution and are close to the mean β^0. We follow the same logic and define $\beta_0^t, t \in \{S, T\}$. This gives us the new optimization problem outlined below.

$$
\begin{aligned}
&\underset{\beta^t,\beta_0^t}{\text{minimize}}\ \lambda_1\left\|\beta^0\right\|^2 + \Sigma_{t\in\{S,T\}}\frac{\lambda_2}{2}\left\|\beta^t\right\|^2 + \Sigma_{i=1}^{l_t}\xi_i\\
&\text{subject to } y_{t_i}((\beta^0 + \beta^t)x_t + \beta_0^t) >= 1 - \xi_{i_t}\\
&\qquad\qquad \xi_{i_t} >= 0, i \in \{1, 2, ..., n_t\}, t \in \{S, T\}
\end{aligned}
\tag{15}
$$

Similar to S3VM, The trade off between λ_1 and λ_2 reflects the trade off between the magnitudes of $\boldsymbol{\beta}^S$ and $\boldsymbol{\beta}^T$, therefore how much the tasks in source and target are expected to differ from each other. Large $\frac{\lambda_2}{\lambda_1}$ forces the models in source and target domains to be similar, and near-zero $\frac{\lambda_2}{\lambda_1}$ tends to make the models unrelated. When $\frac{\lambda_2}{\lambda_1}$ is zero, (15) is equivalent to separately fit regular SVM on source and target task. Note that the unlabelled observations in source and target domains are not utilized. The lambdas can again be chosen by standard techniques such as cross-validations.

2.4 Semi-Transfer Support Vector Machine

In this section, we introduce the novel algorithm "Semi Transfer SVM", which can utilize both labelled and unlabelled data in two domains. We firstly introduce the ST-SVM and give intuitions about how the algorithm utilizes partially labelled observations from both domains, then we will go over its estimation techniques generalized from [3].

2.4.1 ST-SVM

As introduced in previous sections, both transferring knowledge from other domains and utilizing unlabelled observations can both be achieved by modifying the objective of the support vector machines. ST-SVM combines the modifications made by both S3VM and TSVM, and its optimization problem is outlined below.

$$\begin{aligned}\underset{\beta^t,\beta_0^t}{\text{minimize}}\ &\Sigma_{t\in\{S,T\}}\Big(\Sigma_{i=1}^{l_t}\xi_{t_i}+\lambda_1\Sigma_{t\in\{S,T\}}\Sigma_{j=l_t+1}^{l_t+u_t}\xi_j+\\&\frac{\lambda_2}{2}\Sigma_{t\in\{S,T\}}||\beta^t||^2\Big)+\lambda_3||\beta^0||^2\end{aligned} \tag{16}$$

$$\begin{aligned}\text{subject to}\ &y_{i_t}(\beta^t x_{i_t}+\beta_0^t)>=1-\xi_{i_t}\ \ \xi_{i_t}>=0, i\in\{1,2,...,l_t\}, t\in\{S,T\}\\&|\beta^t x_{i_t}+\beta_0^t|>=1-\xi_{i_t}\ \ \xi_{i_t}>=0, i\in\{l_t,l_t+1,...,l_t+u_t\}, t\in\{S,T\}\\&\frac{1}{u_t}\Sigma_{j=l_t+1}^{l_t+u_t}(\beta^t+\beta^0)x_j+\beta_0^t=\frac{1}{l_t}\Sigma_{i=1}^{l_t}y_i, t\in\{S,T\}\end{aligned} \tag{17}$$

In each of the problem domains, slacks measured by labelled observations are minimized, while the hyperplanes are driven away from high unlabelled density regions. At the same time, the hyperplanes are task-specific but share the common knowledge though β^0. The trade offs between labelled and unlabelled observations, the difference between the hyperplanes can be manipulated via the lambdas. The algorithm reduces to S3VM on all data when $\frac{\lambda_2}{\lambda_3}$ tends to infinity, and the algorithm reduces to TSVM when λ_1 equals to zero.

2.4.2 ST-SVM Estimation Via The Concave-Convex Procedure

Loss for Unlabelled Cases Define the hinge loss as $H_s(x)=\max(0,s-|x|)$, as discussed in previous section, ST-SVM assigns a hinge loss $H_1(\cdot)$ on the labelled cases and a "symmetric hinge loss" $H_1(|\cdot|)$ on the unlabelled cases (shown by the left plot of Figure 1). We follow [3] and use a slightly more general form of the hinge loss shown on the right of Figure 1. Given an unlabelled case x, this "non-peaky" loss can be written as $R_s(\beta x+\beta^0)+R_s(-(\beta x+\beta^0))+C$ where C is a constant that does not affect the optimization results and s is a hyper-parameter that controls the wideness of the flat part of the loss function.

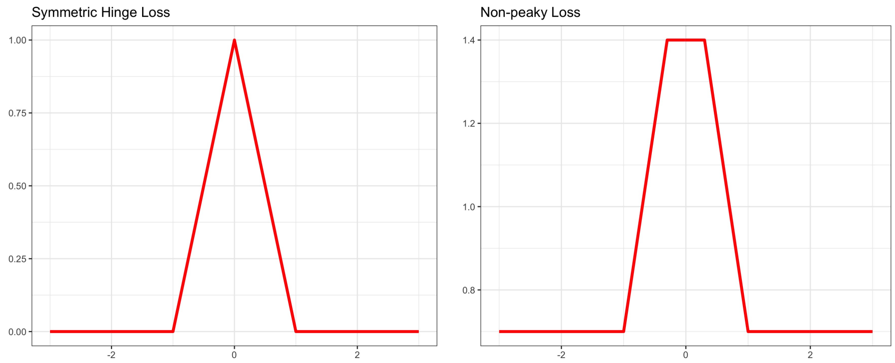

Figure 1

Notations and Definitions To simplify the notations and the optimization problem (this will become more obvious later), we redefine the observations from source and target domains as

$$y_i = 1 \;\text{ for }\; i \in \{l_t, ..., l_t + u_t\} \tag{18}$$
$$y_i = -1 \;\text{ for }\; i \in \{l_t + u_t + 1, ..., l_t + 2u_t\} \tag{19}$$
$$x_i = x_{i-u_t} \;\text{ for }\; i \in \{l_t + u_t + 1, ..., l_t + 2u_t\} \qquad t \in \{S, T\} \tag{20}$$

We further define $f^t(x_i) = \beta_0^t + \boldsymbol{\beta}^t x_i$. Recall that the non-peaky loss for an unlablled observation x can be written as $R_s(f^t(x)) + R_s(-x)$. Given the definition of y for unlabelled observations and (20), the objective (16) can be equivalently written as (21)

$$\lambda_3||\beta^0||^2 + \Sigma_{t\in\{S,T\}}\Big(\frac{\lambda_2}{2}||\beta^t||^2 + \Sigma_{i=1}^{l_t} H_1(y_i f^t(x_i)) + \lambda_1 \Sigma_{i=l_t+1}^{l_t+2u_t} R_s(y_i f^t(x_i))\Big) \tag{21}$$

The CCCP The concave convex procedures is firstly proposed by [12]. Assume we have a cost function $J(\theta)$ which can be written as the sum of a convex part $J_{vex}(\theta)$ and a concave part $J_{cav}(\theta)$. At each step of CCCP, it approximates the $J_{cav}(\theta)$ by its first order derivative and minimizes the resulting function. The procedures of CCCP are given in Algorithm 1.

Algorithm 1 The concave-convex procedure (CCCP)

```
Input: The Best Guess $\theta^0$
Repeat
    $\theta^{t+1} = \arg\min_{\theta} J_{vex}(\theta) + J_{cav}^{'}(\theta^t) \cdot \theta$
Until Convergence of $\theta^t$;
```

One can see that the cost function $J(\theta)$ is guaranteed to decrease at each iteration with the cancavity of $J_{cav}(\theta)$. The proof is given in (22). The convergence of CCCP was also shown by the authors with similar arguments.

$$\begin{aligned} J_{vex}(\theta^{t+1}) + J^{'}_{cav}(\theta^{t}) \cdot \theta^{t+1} \leq J_{vex}(\theta^{t}) + J^{'}_{cav}(\theta^{t}) \cdot \theta^{t} \\ J_{cav}(\theta^{t+1}) \leq J_{vex}(\theta^{t}) + J^{'}_{cav}(\theta^{t}) \cdot (\theta^{t+1} - \theta^{t}) \end{aligned} \tag{22}$$

The CCCP for ST-SVM Pointed out by [3], we notice the ramp loss can be rewritten as the difference between two hinge losses such that $R_s(z) = H_1(z) - H_s(z)$. Incorporating this, the objective function of ST-SVM can then be written as the sum of a convex and concave function. This allows us to apply CCCP.

$$J^s(\theta) = \underbrace{\lambda_3 ||\beta^0||^2 + \Sigma_{t\in\{S,T\}} \frac{\lambda_2}{2} ||\beta^t||^2 + \Sigma_{i=1}^{l_t} H_1(y_i f^t(x_i)) + \lambda_1 \Sigma_{i=l_t+1}^{l_t+2u_t} H_1(y_i f^t(x_i))}_{J_{vex}} \tag{23}$$

$$\underbrace{-\Sigma_{t\in\{S,T\}} \lambda_1 \Sigma_{i=l_t+1}^{l_t+2u_t} H_s(y_i f^t(x_i))}_{J_{cav}} \tag{24}$$

It is easy to derive that $\frac{\mathrm{d}J_{cav}}{\mathrm{d}\theta} = \lambda_1 \Sigma_{t\in\{S,T\}} \Sigma_{i=l_t+1}^{l_t+2u_t} \frac{\mathrm{d}J_{cav}}{\mathrm{d}f^t(x_j)} \times \frac{\mathrm{d}f^t(x_j)}{\mathrm{d}\theta}$, and we introduce the notation $w_i^t = y_i \frac{\mathrm{d}J_{cav}}{\mathrm{d}f^t(x_i)} = \begin{cases} \lambda_1 & \text{if } y_i f^t(x_i) < s \text{ and } i > l_t, \\ 0 & \text{Otherwise} \end{cases}, t \in \{S,T\}$ Note that $f^t(x_i) = \beta_0^t + \beta^t x_i$, we define $\theta = (\beta^t, \beta_0^t)$, then we know that $\frac{\mathrm{d}f^t(x_i)}{\mathrm{d}\theta} = (x_i, 1)$ for $t \in \{S,T\}$ Now we can write the CCCP loss as

$$J_{vex}(\Theta) + \frac{\mathrm{d}J_{cav}(\Theta)}{\mathrm{d}\Theta} \cdot \Theta = J_{vex}(\Theta) + \lambda_1 \Sigma_{t\in\{S,T\}} \Sigma_{i=l_t+1}^{l_t+2u_t} w_i^t y_i f(x_i) \tag{25}$$

For each iteration of CCCP, the optimization task is

$$\underset{\Theta,\xi}{\text{minimize}}\ \lambda_3 ||\beta^0||^2 + \Sigma_{t\in\{S,T\}} \Big(\frac{\lambda_2}{2} ||\beta^t||^2 + \Sigma_{i=1}^{l_t} \xi_i^t + \lambda_1 \Sigma_{i=l_t+1}^{l_t+2u_t} \xi_i^t + \Sigma_{i=l_t+1}^{l_t+2u_t} w_i^t y_i f^t(x_i) \Big) \tag{26}$$

subject to $y_i f^t(x_i) >= 1 - \xi_i^t$

$$\xi_{t_i} >= 0, i \in \{1, 2, ..., l_t + 2u_t\}, t \in \{S,T\}$$

$$\frac{1}{u_t} \Sigma_{i=l_t+1}^{l_t+u_t} f^t(x_i) = \frac{1}{l_t} \Sigma_{i=1}^{l_t} y_i, t \in \{S,T\}$$

This is a disciplined convex optimization (DCP)[13]. Standard solvers for this type of problems are available in many major programming languages

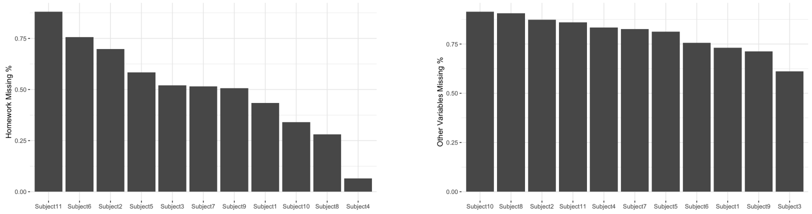

Figure 2

(Python, R, Matlab, etc.). The full optimization procedures should initiate the weights with estimations of TSVM. As the only set of parameter altered in each iterations is $w_i^t, t \in \{S, T\}$, together with (22), we know the algorithm is guaranteed to converge in a finite number of iterations to at least a local minimum. One can therefore repeat (26) until $w_i^t, t \in \{S, T\}$ do not change. This optimization problem is further simplified as a quadratic programming problem in the appendix.

3 Application

3.1 Background

We demonstrate ST-SVM with smartphone data collected by Weill Cornell ALACRITY Center. This data set records passively collected variables such as ”step counts”, ”time away from home”, and self-reported variables by subjects such as ”stress”. Each subject is asked everyday whether they complete the ”homework” assigned by doctors to aid their depressions. Our objective is to predict homework status using passive and self-reported variables prior to the time subjects answer the ”homework” question.

Practically speaking, it is a challenging problem for traditional supervised algorithms. The subjects come from a variety of social-economic background with vastly different behavioural patterns, so individualized prediction models are required. This means the size of the training data for each model is limited to be the length of follow-up for that one subject. Furthermore, subjects oftentimes do not provide an answer to the ”homework” question, which leads to a large proportion of missing in the predicted variable (left of Figure 1). When they do provide the answers to ”homework”, other variables may still be missing(right of Figure 1). This results a low signal-to-noise ratio for each subject.

ST-SVM can be applied in this setting. Eleven subjects who have more than four ”Yes” and ”No” responses are selected. For each subject, we built a ST-SVM with the current subject as T and all the other subjects as S. One can also build a single ST-SVM and assign a domain to each of the subjects. We do not consider this option here as this would greatly diminishes the shared knowledge between subjects. redneed to be expanded

3.2 Numerical Results

To show the advantages of ST-SVM, four types of SVM namely regular SVM, S3VM, TSVM and ST-SVM are fitted on each of the subjects. Each of the cost parameters λ is cross-validated with $\{0.1, 0, 1\}$, and the tuned model are tested on the 20% held-out data. This procedure is repeated for three times, and the averaged prediction accuracies are shown by Table 1.

	ST-SVM	S3VM	TSVM	SVM
Subject 1	0.87	0.73	0.85	0.86
Subject 2	0.73	0.68	0.63	0.65
Subject 3	0.72	0.59	0.67	0.64
Subject 4	0.69	0.67	0.62	0.62
Subject 5	0.69	0.50	0.62	0.59
Subject 6	0.66	0.42	0.56	0.56
Subject 7	0.62	0.55	0.54	0.52
Subject 8	0.71	0.67	0.58	0.55
Subject 9	0.69	0.65	0.69	0.64
Subject 10	0.77	0.71	0.69	0.60
Subject 11	0.84	0.82	0.74	0.64

Table 1

In this example, ST-SVM consistently outperform all other SVMs, which is in line with our expectation as the tuning parameters are well selected. It is worth noticing that when the signals are strong (Subject 1), other methods can hardly improve upon the the performance of regular SVM. This suggests the domain knowledge of labelled data only is sufficient. however when the local signals are weak (Subject 2-11), knowledge from unlabelled data and other subjects can greatly boost the prediction accuracy.

4 Concluding Remarks

We have proposed an algorithm which can learn from multiple problem domains with only partially labelled data. It reduces to S3VM, TSVM or regular SVM when corresponding cost parameters are set to be zero. Its non-convex optimization problem can be simplified to a sequence of disciplined convex optimization and solved by standard programming languages. When multiple problem domains are present, our results show its performance is significantly higher than traditional SVMs.

Appendix

We further transform the problem (26) in to a simpler quadratic programming (QP) problem. Introduce Lagrangian variables α_0^t, $\boldsymbol{\alpha}^t$, $\boldsymbol{\mu}^t$ where $t \in \{S, T\}$, with respective to the constraints, we can write its Lagrange primal problem as (27). Please note that we neglect neglect $t \in \{S, T\}$ from now on.

$$\mathcal{L}_p(\Theta, \boldsymbol{\xi}, \boldsymbol{\alpha}^t, \boldsymbol{\mu}^t) = \lambda_3 ||\beta^0||^2 \tag{27}$$

$$+ \Sigma_{t\in\{S,T\}} \Big(\frac{\lambda_2}{2} ||\beta^t||^2 + \Sigma_{i=1}^{l_t} \xi_i^t + \lambda_1 \Sigma_{i=l_t+1}^{l_t+2u_t} \xi_i^t + \Sigma_{i=l_t+1}^{l_t+2u_t} w_i^t y_i f^t(x_i) \Big) \tag{28}$$

$$- \Sigma_{t\in\{S,T\}} \alpha_0^t \Big(\Sigma_{i=l_t+1}^{l_t+u_t} \frac{1}{u_t} f^t(x_i) - \frac{1}{l_t} \Sigma_{i=1}^{l_t} y_i \Big) \tag{29}$$

$$- \Sigma_{t\in\{S,T\}} \Sigma_{i=1}^{l_t+2u_t} \alpha_i^t (y_i f^t(x_i) - 1 + \xi_i^t) \tag{30}$$

$$- \Sigma_{t\in\{S,T\}} \Sigma_{i=1}^{l_t+2u_t} \mu_i^t \xi_i^t \tag{31}$$

Taking the derivatives with respect to the primal variables yields the followings. Note that $w_i^t = 0$ when $i \leq l_t$.

$$\frac{\mathrm{d}\mathcal{L}_p}{\mathrm{d}\beta^0} = 2\lambda_3 \beta^0 + \Sigma_{t\in\{S,T\}} \Big(\Sigma_{i=1}^{l_t+2u_t} w_i^t y_i x_i - \Sigma_{i=1}^{l_t+2u_t} \alpha_i^t y_i x_i - \Sigma_{i=1}^{l_t+u_t} \frac{\alpha_0^t}{u_t} x_i \Big) \tag{32}$$

$$= 2\lambda_3 \beta^0 + \Sigma_{t\in\{S,T\}} \Big(- \Sigma_{i=1}^{l_t+2u_t} (\alpha_i^t - w_i^t) y_i x_i - \Sigma_{i=1}^{l_t+u_t} \frac{\alpha_0^t}{u_t} x_i \Big) \tag{33}$$

$$\frac{\mathrm{d}\mathcal{L}_p}{\mathrm{d}\beta^t} = \lambda_2 \beta^t - \Sigma_{i=1}^{l_t+2u_t} (\alpha_i^t - w_i^t) y_i x_i - \Sigma_{i=1}^{l_t+u_t} \frac{\alpha_0^t}{u_t} x_i \tag{34}$$

$$\frac{\mathrm{d}\mathcal{L}_p}{\mathrm{d}\beta_0^t} = -\Sigma_{i=1}^{l_t+2u_t} (\alpha_i^t - w_i^t) y_i - \alpha_0^t \tag{35}$$

$$\frac{\mathrm{d}\mathcal{L}_p}{\mathrm{d}\xi_i^t} = \begin{cases} 1 & \text{if } 1< i < l_t, \\ \lambda_1 & \text{if } l_t + 1< i < l_t + 2u_t \end{cases} - \alpha_i^t - \mu_i^t \tag{36}$$

To simplify the notations, we introduce $x_{0^t} = \Sigma_{i=l_t+1}^{l_t+u_t} \frac{1}{u_t} x_i$, $y_{0^t} = 1$ and $w_{0^t} = 1$. Setting the derivatives to zero, we can get

$$\beta^0 = \frac{1}{2 \times \lambda_3} \Sigma_{t\in\{S,T\}} \Sigma_{i=0}^{l_t+2u_t} (\alpha_i^t - w_i^t) y_i x_i \tag{37}$$

$$\beta^t = \frac{1}{\lambda_2} \Sigma_{i=0}^{l_t+2u_t} (\alpha_i^t - w_i^t) y_i x_i \tag{38}$$

$$\Sigma_{i=0}^{l_t+2u_t} (\alpha_i^t - w_i^t) y_i = 0 \tag{39}$$

$$\begin{cases} 1 & \text{if } 1< i < l_t, \\ \lambda_1 & \text{if } l_t + 1< i < l_t + 2u_t \end{cases} - \alpha_i^t - \mu_i^t = 0 \tag{40}$$

If we substitute (32 - 36) back to the problem (26), we can get

$$\begin{aligned} \underset{\Theta,\xi}{\text{maximize}} \ & -\frac{1}{4 \times \lambda_3} \Sigma_{t,t' \in \{S,T\}} \Big(\Sigma_{i,j=0}^{l_t+2u_t} (\alpha_i^t - w_i^t)(\alpha_j^{t^{'}} - w_j^{t^{'}}) y_i y_j x_i x_j \Big) \\ & - \Sigma_{t \in \{S,T\}} \Big(\frac{1}{2 \times \lambda_2} \Sigma_{i,j=0}^{l_t+2u_t} (\alpha_i^t - w_i^t)(\alpha_j^t - w_j^t) y_i y_j x_i x_j \\ & \qquad + \Sigma_{i=1}^{l_t+2u_t} \alpha_i^t + \alpha_0 (\frac{1}{l_t} \Sigma_{i=1}^{l_t} y_i) \Big) \\ \text{subject to } & 0 \le \alpha_i \le \begin{cases} 1 & \text{if } 1< i < l_t, \\ \lambda_1 & \text{if } l_t + 1< i < l_t + 2u_t \end{cases} \\ & \Sigma_{i=0}^{l_t+2u_t} (\alpha_i^t - \lambda_1 w_i^t) y_i = 0 \\ & t \in \{S,T\} \end{aligned} \tag{41}$$

The weight vectors β^t, β^0 are given respectively by (38) and (37). The offset terms β_0^t can be obtained from the the Karush-Kuhn- Tucker (KKT) conditions (42).

$$\alpha_0^t \neq 0 \implies \frac{1}{u_t} \Sigma_{i=l_t}^{l_t+u_t} x_i^T \beta^t + \beta_0^t = \frac{1}{l_t} \Sigma_{i=1}^{l_t} y_i \tag{42}$$

We re-define $\xi_i^t = y_i$ for $i \in 1, ..., l_t + 2u_t$ and $\xi_0 = \frac{1}{l_t} \Sigma_{i=1}^{l_t} y_i$, and define matrix $\Sigma^{t,t^{'}}$ such that $\Sigma_{ij}^{t,t^{'}} = x_i x_j, \ x_i \in t, x_j \in t^{'}$. We further introduce a variable $\gamma_i^t = y_i(\alpha_i^t - w_i^t)$. The optimization problem (41) can then be written as

$$\begin{aligned} \underset{\Theta,\xi}{\text{maximize}} \ & -\frac{1}{4 \times \lambda_3} \Sigma_{t,t' \in \{S,T\}} \boldsymbol{\gamma}^t \Sigma^{t,t^{'}} \boldsymbol{\gamma}^{t^{'}} \\ & - \Sigma_{t \in \{S,T\}} \Big(\frac{1}{2 \times \lambda_2} \boldsymbol{\gamma}^t \Sigma^{t,t} \boldsymbol{\gamma}^t \\ & \qquad + \boldsymbol{\xi}^t \boldsymbol{\gamma}^t \Big) \\ \text{subject to } & \begin{cases} 0 \\ -w_i^t \end{cases} \le \gamma_i^t y_i \le \begin{cases} 1 & \text{if } 1< i < l_t, \\ \lambda_1 - w_i^t & \text{if } l_t + 1< i < l_t + 2u_t \end{cases} \\ & \Sigma_{i=0}^{l_t+2u_t} \gamma_i^t = 0 \\ & t \in \{S,T\} \end{aligned} \tag{43}$$

Thus, for each iteration of CCCP, we only need to solve a QP problem which involves parameters $\boldsymbol{\gamma}^t, \boldsymbol{\xi}^t, t \in \{S,T\}$.

References

[1] T. Evgeniou and M. Pontil, "Regularized multi-task learning," in Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Dis-

covery and Data Mining. Seattle, Washington, USA: ACM, August 2004, pp. 109–117.

[2] V. Vapnik. The Nature of Statistical Learning Theory. Springer, second edition, 1995.

[3] Ronan Collobert, Fabian Sinz, Jason Weston, and Léon Bottou. 2006. Large Scale Transductive SVMs. J. Mach. Learn. Res. 7 (12/1/2006), 1687–1712.

[4] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. 2007. Self-taught learning: transfer learning from unlabeled data. In Proceedings of the 24th international conference on Machine learning (ICML '07). Association for Computing Machinery, New York, NY, USA, 759–766. DOI:https://doi.org/10.1145/1273496.1273592

[5] A. Argyriou, T. Evgeniou, and M. Pontil, "Multi-task feature learning," in Proceedings of the 19th Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 2007, pp. 41–48.

[6] A. Schwaighofer, V. Tresp, and K. Yu, "Learning gaussian process kernels via hierarchical bayes," in Proceedings of the 17th Annual Conference on Neural Information Processing Systems. Cambridge, MA: MIT Press, 2005, pp. 1209–1216.

[7] W. Dai, Q. Yang, G. Xue, and Y. Yu, "Boosting for transfer learning," in Proceedings of the 24th International Conference on Machine Learning, Corvalis, Oregon, USA, June 2007, pp. 193–200.

[8] Freund, Y., Schapire, R. E. (1997). A decisiontheoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences, 55(1), 119–139.

[9] Zhu, Xiaojin, Semi-Supervised Learning Literature Survey http://digital.library.wisc.edu/1793/60444

[10] B. E. Boser, I. M. Guyon, and V. N. Vapnik. A training algorithm for optimal margin classifiers. In Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pages 144–152, Pittsburgh, PA, 1992. ACM Press.

[11] Chapelle, O., Zien, A. (2005). Semi-Supervised Classification by Low Density Separation. AISTATS.

[12] A. L. Yuille and Anand Rangarajan. 2001. The Concave-Convex procedure (CCCP). In Proceedings of the 14th International Conference on Neural Information Processing Systems: Natural and Synthetic (NIPS'01). MIT Press, Cambridge, MA, USA, 1033–1040.

[13] Grant M., Boyd S., Ye Y. (2006) Disciplined Convex Programming. In: Liberti L., Maculan N. (eds) Global Optimization. Nonconvex Optimization and Its Applications, vol 84. Springer, Boston, MA. https://doi.org/10.1007/0-387-30528-9-7

[14] S. J. Pan and Q. Yang, "A Survey on Transfer Learning," in IEEE Transactions on Knowledge and Data Engineering, vol. 22, no. 10, pp. 1345-1359, Oct. 2010, doi: 10.1109/TKDE.2009.191.