Abstract—Multitask Learning has been proven to be more effective than the traditional single task learning on many real-world problems by simultaneously transferring knowledge among different tasks which may suffer from limited labeled data. However, in order to build a reliable multitask learning model, nontrivial effort to construct the relatedness between different tasks is critical. When the number of tasks is not large, the learning outcome may suffer if there exists outlier tasks that inappropriately bias majority. Rather than identifying or discarding such outlier tasks, we present a weighted regularized multitask learning framework based on regularized multitask learning, which uses statistical metrics, such as Kullback-Leibler divergence, to assign weights prior to regularization process that robustly reduces the impact of outlier tasks and results in better learned models for all tasks. We then show that this formulation can be solved using dual form like optimizing a standard support vector machine with varied kernels. We perform experiments using both synthetic dataset and real-world dataset from petroleum industry which shows that our methodology outperforms existing methods.

Index Terms—multitask learning; anomaly detection; weighted regularization; svm; outlier task;

I. INTRODUCTION

For problems that can be divided into separate, yet related, tasks, where each task has their own feature distributions, multitask learning has been proven to perform better than the traditional single task learning. Traditional single-task learning approaches tend to fail when number of training examples are small for each task. The small training sample size under constrains the search in hypothesis space for appropriate models resulting in high generalization errors. Under the setting of multitask learning related tasks can be used to introduce biases to fine tune the model search. In order to capture this bias information, many approaches in multitask learning have been proposed including sharing distance metric, sharing common feature set [7], sharing a common latent representation [8], etc. Regularization has been introduced to solve multitask learning problems that produces better performance than traditional solution using single-task learning. [1] [2]. The approach described in these works takes full advantage of support vector machines by constructing novel kernels that reveal task relations.

Multitask learning is a learning framework which falls into transfer learning category which requires the same feature representation for all tasks. It tries to learn multiple tasks simultaneously even when they are different. As an active field in machine learning, multitask learning has been adapted to discover the common (latent) features that benefits each individual task.

In this paper, we develop methods by properly generalizing the formulation of regularization multitask learning model. Especially, we construct each learning task as a support vector machine classification with different weights that represents their divergence with other tasks. We then prove that the combined form of multiple SVM still satisfies the multitask learning property that enables smoother and more flexible task relation assumptions. This MTL can be reduced to a dual form optimization similar to how traditional SVM works. In both synthetic dataset and real-world experiments where task assignment is easily known, we show that this generalized form works better when the observations are rare for each task while all tasks have noisy data. Our methodology outperforms existing single task learning and regularized multitask learning.

A. Related Work

The intrinsic task relatedness is critical to multitask learning. Various regularization metrics have been developed to handle different assumptions on how tasks are related. Tasks can be fully related [9], containing outliers [10], in form of groups [11], graphs [12] or trees [13]. Zhou et. al. apply MTL to predict disease progression for different patients [14], [15] by doing multiple ridge regression with temporal smoothing priors in its learning process. But optimizing the corresponding multitask learning objectives are always difficult because rather than the form of original single task learning, multitask learning introduces more complexity by combining multiple
objectives together, as well as additional task relation regularization terms.

[1] develops methods for solving multitask learning problems by naturally extending existing kernel based learning methods for single task learning. A complex multitask learning can be formulated as a single task dual optimization and furthermore, just by varying the kernel by inheriting support vector machines implementations would yield optimal learning outcome. Their experiments show the benefits of multitask learning which are superior to their single task counterpart.

B. Notation

Given $T$ binary classification tasks, for each task $t$ there are $m_t$ examples $\{(x_{it}, y_{it} : t \in \mathbb{N})\}$ sampled from a distribution $P_t$ on $X_t \times Y_t$ where $X_t \subset \mathbb{R}^d$, $Y = \{+1, -1\}$. The total labeled data is:

$$\{(x_{i1}, y_{i1}), \ldots, (x_{im_t}, y_{im_t})\}$$

The goal is to learn $T$ functions

$$f_1, f_2, \ldots, f_T$$

such that

$$f_t(x_{it}) \approx y_{it}$$

For support vector machines, the functions can be described by a set of hyperplanes so that for each task, the hyperplane separates the positively-labeled points with the negatively-labeled points by maximum margin in linear (dot product) space or in Reproducing Kernel Hilbert Space (RKHS) by kernel tricks.

II. METHODS FORMULATION

A. Problem Definition

For simplicity, we assume that each task contains same number of examples, and the functions $f_t$ is a hyperplane in linear space which is $f_t(x) = w_t \cdot x$, for $t \in \{1, 2, \ldots, T\}$. $w_t$ represents the hyperplane and in linear space, $w_t \cdot x$ is the dot product of the two vectors. In a support vector machine classifier, $w_t$ is the maximum margin hyperplane. In [1], the task relation assumption is Gaussian so that the parameters for each task are independent and identically distributed (i.i.d.). There are two assumptions: 1) i.i.d. and 2) number of tasks is large, under which the central limit theorem can be applied to the task parameters as random variables to estimate the distribution. But in practice the number of tasks may be small, and each task may not have equal weights, e.g. prioritizations for some tasks. When used to analyze the observed correlations, this lack of examples and lack of i.i.d. characteristics may introduce the effect called Simpson’s Paradox, or Yule-Simpson Effect [5].

One common approach to handle the Simpson’s Paradox is to weight each task differently. Thus rather than the uniform mean which is used for estimating Gaussian distribution parameters, we use a weighted mean.

$$w_t = w_0 + \alpha_t v_t \quad (1)$$

where $w_t$ is the parameter for task $t$, $v_t$ is the bias of the task, $\alpha_t$ is the weight for task $t$’s bias, and $w_0$ is the weighted mean of all task parameters. If $\alpha_t$ are all 1.0, this assumption is reduced to Gaussian.

Due to the fact that each task may not have same level of priority, we can assign the values for $\alpha_t$ but instead, we solve it by forming the following optimization problem by allowing weights assignment for each task indicated via prior knowledge:

PROBLEM 1.

$$\min_{w_0, v_t, \xi_{it}} \mathcal{J}(w_0, v_t, \xi_{it}) = \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it}$$

$$+ \frac{1}{T} \sum_{t=1}^{T} \lambda_t ||\alpha_t v_t||^2 + \lambda_0 ||w_0||^2$$

subject to $\xi_{it} \geq 0$

$$y_{it} w_t \cdot x_{it} \geq 1 - \xi_{it}$$

where $i \in \{1, 2, \ldots, m\}, t \in \{1, 2, \ldots, T\}$

$\lambda_0$ is regularization parameter for the mean parameter, $\lambda_t$ is regularization parameter for each biased parameter, while $\xi_{it}$ is the slack variable for task $t$’s $i$th example which measures the hinge loss of the model on this data point. The regularization parameters control the parameters out of optimizations. From this formulation, the larger $\lambda_t$ is, the closer the corresponding regularization term tend to be zero. Thus, if $\lambda_0 \to \infty$, then each task would be independent because $w_0 \to 0$. If task $t$’s parameters diverge from other tasks, we can expect its $\lambda_t$ to be small because it requires greater bias to deviate itself to other tasks. Similar tasks are expected to have greater $\lambda_t$ because they are likely to be used to reveal the average parameters.

B. Problem Equivalent to Multitask Learning

LEMMA 1. The optimal solution to the Problem 1 is:

$$w_0 = \frac{\sum_{t=1}^{T} \lambda_t w_t}{T \lambda_0 + \sum_{t=1}^{T} \lambda_t} \quad (2)$$

Proof: Using Lagrange multiplier, we can easily have the Lagrange function as:

$\ell(w_0, v_t, \alpha_t, \gamma_{it}) = \mathcal{J}(w_0, v_t, \xi_{it})$

$$- \sum_{t=1}^{T} \sum_{i=1}^{m} \alpha_t (y_{it}(w_0 + \alpha_t v_t) \cdot x_{it} - 1 + \xi_{it}) - \sum_{t=1}^{T} \sum_{i=1}^{m} \gamma_{it} \xi_{it} \quad (3)$$

where $\alpha_t, \gamma_{it}$ are the Lagrange multipliers for Problem 1. Take partial derivative for $w_0$, we can have:

$$w_0 = \frac{1}{2 \lambda_0} \sum_{t=1}^{T} \sum_{i=1}^{m} \alpha_t y_{it} x_{it} \quad (4)$$
Similarly, for $\alpha_t v_t$:

$$\alpha_t v_t = \frac{T}{2\lambda_t} \sum_{i=1}^{m} \alpha_{it} y_{it} \tilde{x}_{it}$$  \hspace{1cm} (5)$$

Therefore

$$w_0 = \frac{1}{\lambda_0 T} \sum_{t=1}^{T} \lambda_t \alpha_t v_t$$

Therefore given Equation 1, we have:

$$w_t = w_0 + \alpha_t v_t$$

Replacing $\alpha_t v_t$ we have:

$$w_0 = \frac{\sum_{t=1}^{T} \lambda_t w_t}{T \lambda_0 + \sum_{t=1}^{T} \lambda_t}$$

From this expression of relation between $w_0$ and $w_t$, we can conclude that $w_0$ is the weighted mean of $w_t$ only when $\lambda_0 = 0$, otherwise it is not.

Now we can prove that optimizing the single formulation of Problem 1 is equivalent to a multitask learning problem.

**LEMMA 2.** Solving the optimization problem defined in Problem 1 is equivalent to solving the following multitask learning problem:

**PROBLEM 2.**

\[
\min_{w_t, \xi_t} \sum_{t=1}^{T} \sum_{i=1}^{m} \xi_{it} + \sum_{t=1}^{T} \rho_{1,t} \alpha_t ||w_t||^2
\]

\[+ \rho_2 \sum_{t=1}^{T} \left|\alpha_t v_t - \frac{1}{T} \sum_{s=1}^{T} \alpha_s w_s\right|^2\]

\[\text{such that } i \in \{1, 2, \ldots, m\}, t \in \{1, 2, \ldots, T\}\]

\[\xi_{it} \geq 0\]

\[y_{it} w_t \cdot x_{it} \geq 1 - \xi_{it}\]

The values for $\rho_{1,t}$ and $\rho_2$ are

\[\rho_{1,t} = \frac{1}{T} \left(1 - \frac{\lambda_t T}{\lambda_0 T + \sum_{t=1}^{T} \lambda_t}\right)\]  \hspace{1cm} (7)

\[\rho_2 = \frac{1}{\lambda_0 T + \sum_{t=1}^{T} \lambda_t}\]  \hspace{1cm} (8)

Proof: Based on the form from Problem 1, taking in Equation 1, 2 and 4 we have:

\[\frac{1}{T} \sum_{t=1}^{T} \lambda_t ||w_t||^2 + \lambda_0 ||w_0||^2\]

\[= \frac{1}{T} \sum_{t=1}^{T} \lambda_t ||w_t||^2 - \left(\frac{\sum_{t=1}^{T} \lambda_t}{T}\right) ||w_0||^2\]

\[= \frac{1}{T} \sum_{t=1}^{T} \lambda_t ||w_t||^2 - \frac{1}{\lambda_0 T + \sum_{t=1}^{T} \lambda_t} \frac{1}{T} \left(\sum_{t=1}^{T} \lambda_t w_t\right)^2\]

For the form from Problem 2:

\[\sum_{t=1}^{T} \rho_{1,t} \alpha_t ||w_t||^2 + \rho_2 \sum_{t=1}^{T} \left|\alpha_t v_t - \frac{1}{T} \sum_{s=1}^{T} \alpha_s w_s\right|^2\]

\[= \sum_{t=1}^{T} (\rho_{1,t} \alpha_t + \rho_2 \alpha_t^2) ||w_t||^2 - \frac{\rho_2}{T} \sum_{t=1}^{T} \alpha_t w_t^2\]

\[\Leftrightarrow \frac{1}{T} \sum_{t=1}^{T} \lambda_t ||w_t||^2 - \frac{1}{\lambda_0 T + \sum_{t=1}^{T} \lambda_t} \frac{1}{T} \left(\sum_{t=1}^{T} \lambda_t w_t\right)^2\]

Let $\alpha_t = \lambda_t$, we have:

\[\rho_2 = \frac{1}{\lambda_0 T + \sum_{t=1}^{T} \lambda_t}\]

\[\rho_{1,t} = \frac{1}{T} \left(1 - \frac{\lambda_t T}{\lambda_0 T + \sum_{t=1}^{T} \lambda_t}\right)\]

Now the form of $\rho_{1(t)}$ and $\rho_2$ are found, and the equivalence is proved.

C. Kernels

1) Linear case: In this section, we introduce Dual optimization to solve the problem and derive the kernel in order to solve Problem 1.

The dual optimization is standard and is similar to solving common SVM problems. Based on the Lagrange function described in Equation 3, we take derivatives for different variables, replace the terms from the function and finally get:

\[\ell(w_0, v_t, \alpha_t, \gamma_{it}) = \sum_{t=1}^{T} \sum_{i=1}^{m} \alpha_{it} - \frac{1}{2} \lambda_0 T + \sum_{t=1}^{T} \lambda_t\]

\[\frac{1}{T} \sum_{t=1}^{T} \sum_{s=1}^{T} \sum_{j=1}^{m} \left(\frac{T}{2\lambda_t} \delta_{st} + \frac{1}{2\lambda_0}\right) \alpha_{it} \alpha_{js} y_{it} y_{js} \tilde{x}_{it} \cdot \tilde{x}_{js}\]

Thus if we consider $C = \frac{1}{2\lambda_0}$ and define $\mu_t = \frac{T \lambda_0}{\lambda_t}$, we can then define the kernel as:

\[K_{st}(x_{it}, x_{js}) = (\delta_{st} \mu_t + 1) \tilde{x}_{it} \cdot \tilde{x}_{js}\]  \hspace{1cm} (9)

Once the kernel is defined, the optimization can be done by fitting in any existing SVM solutions.

2) Nonlinear case: We can easily generalize the linear kernel into nonlinear case using kernel tricks so that the original features are mapped into higher dimensional Hilbert space $\mathcal{H}$.

\[\Phi : \mathcal{X} \rightarrow \mathcal{H}\]

Then the new kernel function would be the inner product of the mapped features:

\[K'_st(x_{it}, x_{js}) = (\delta_{st} \mu_t + 1) \langle \Phi(\tilde{x}_{it}), \Phi(\tilde{x}_{js}) \rangle\]  \hspace{1cm} (10)

If we are mapping it into Hilbert space with infinite dimensions, we can replace Equation 10 into an RBF function, and then we apply common SVM in practice.
D. Generalization of uniform regularized multitask learning

For a uniform multitask regularization kernel (RMTL), each task is assigned with identical weights, thus the ratio between \( \lambda_2 \) (for \( w_0 \)) and \( \lambda_1 \) (for \( w_1 \)) determines the task similarities, which can be represented as \( \mu = \frac{T\lambda_2}{\lambda_1} \).

\[
K_{\text{rmtl}}(x_{it}, x_{js}) = \delta_{st} + \frac{1}{\mu} \vec{x}_{it} \cdot \vec{x}_{js} \\
= \left( \delta_{st} + \frac{1}{T\lambda_2} \right) \vec{x}_{it} \cdot \vec{x}_{js}
\]

For generalized regularized multitask learning kernel (WRMTL), the task parameters are:

\[
K_{\text{wrmtl}}(x_{it}, x_{js}) = (\delta_{st}\mu_t + 1)\vec{x}_{it} \cdot \vec{x}_{js} \\
= \left( \delta_{st} + \frac{1}{T\lambda_0} \right) \vec{x}_{it} \cdot \vec{x}_{js}
\]

It is clear that our derivation is a superset representation for the uniformed version. By setting each \( \lambda_t \) to be uniform, we are expecting the same output. Rather than \( \mu \) in RMTL represents general task similarity, in WRMTL, each \( \mu_t \) actually represents how each task diverges from the other tasks. Larger \( \mu_t \) introduces higher penalties if any of its example is misclassified during training stage. For a task that exhibits small divergence from other tasks, it {melts} itself into the optimization process so that each task would learn from it. On the contrary, if we set a \( \mu_t \) to be infinite, then the task will learn nothing from other tasks. Therefore this framework has a very robust characteristics that lessens the effect of outlier tasks.

III. EXPERIMENTS

We divide this section into two parts. We first demonstrate the effectiveness of our algorithm using synthetic data. Then we test it using real-world data that is from petroleum industry. Our implementation is based on the implementation of LibSVM [16].

A. Synthetic Data

We tested the proposed method using data with task affinity and dissimilarity. We generated two-variated Gaussian distributions for three tasks, each task consists of two Gaussian shaped classes and each dimension corresponds to one attribute. We enforce two of the tasks (i.e. \( T_1 \) and \( T_2 \)) to be distributively similar, while the other one (i.e. \( T_3 \)) possesses different inner relations and class distributions, which could be deemed as an outlier. In such a setting, we can show how our weighted method could effectively utilize the affinity among tasks to adjust the hyperplane so as to improve classification accuracy.

We use the same Gaussian to generate hyperplanes for \( T_1 \) and \( T_2 \), which are straight lines in two-dimensional space, and use another Gaussian for \( T_3 \). Given each hyperplane (weight vector), we generate two Gaussian classes to each side with large variance so that classes heavily mix up together. The number of samples for each class and each task is 300, thus 1800 in total. Fig. 1 shows the synthetic dataset and the optimal classification hyperplanes for all three tasks. Given the basic synthetic dataset, we only randomly pick extremely limited labeled observations as the training set, say 3 out of 300, and the rest is for testing. Our rod pumps data set has the characteristic of being very noisy and of having limited number of labeled data. Normal SVM classifier is not able to learn an optimal hyperplane from the small number of training points for each task. The resulting hyperspace tends to be badly deviated or even be orthogonal to the optimal hyperplane.

![Fig. 1. Synthetic dataset: Task 1 and Task 2 are two similar tasks while Task 3 is very different from their optimal classification boundaries - red for Task 1, blue for Task 2 and pink for Task 3.](image-url)
make impacts; and vice versa with larger $\mu_t$. $\mu_t$ is constant across three tasks for RMTL, but varies according to task similarity and dissimilarity for WRMTL. We tested RMTL with $\mu = (0.0001, 0.001, 0.01, 0.1, 1, 10, 100, 1000, 10000)$ and tested WRMTL with $\mu_t = (0.001, 0.001, 1000) \ast \mu$, which means for each scale of $\mu$, we don’t retain $\delta_s = 1$ for each task anymore, instead, we add a weight for each task. Because of the similarity between $T_1$ and $T_2$, we predict that sharing each other’s training set would improve the result. While $T_3$ is relatively independent and dissimilar using the training points of $T_1$, $T_2$ should not lead to better performance. $\mu_t$ is changeable within some range, as long as keeping $\mu_t$ small enough for similar tasks and large for outliers, then we could derive some satisfying results.

For each of the four methods, Fig. 2 shows the classification accuracy averaged over 1000 iterations. For each iteration, we randomly pick 3 training points for each class in each task. The rest of the points are reserved for the test set. Within each iteration, sSVM and oSVM are run once. RMTL and WRMTL are run 9 times by varying $\mu$ and $\mu_t$, then picking the highest one as the accuracy representing for each iteration.

In Fig. 3, the top row represents the optimal classification boundaries for three tasks as in three columns. The second row represents the results for sSVM. The third row shows the results using oSVM, and the bottom two rows are results for RMTL and WRMTL individually. All the observed data that are used for training are circled. We can observe that WRMTL outperforms all others with the smallest deviation, and second comes RMTL, then sSVM, finally oSVM.

The reason for WRMTL outperforming the other three algorithms could be illustrated by considering task affinity between $T_1$ and $T_2$, we enlarge both training set by combining them together, which leads to an adjustment of both hyperplanes. The second row shows that sSVM’s hyperplanes are completely influenced by the sparse training set. The sSVM’s hyperplanes for task $T_1$ and $T_2$ substantially deviate from optimal hyperplanes, while the hyperplane for $T_3$ seems reasonable. The third row shows oSVM hyperplanes which is significantly influenced by $T_1$ and $T_2$, which makes the hyperplane for $T_3$ almost orthogonal to its optimal one. The oSVM accuracy is the worst of all; thus the importance of task relation is highlighted here. By using RMTL, the best result happens when $\mu = 0.1$, which implies that task relations play an important role for classification. But because of the limitation of RMTL’s assumption that each task shares equal weight during estimating the task parameter distribution, the fourth row shows the hyperplanes for all three tasks that are skewed towards to wrong way whose performance is as poor as oSVM. The fifth row shows that the skew is successfully prevented by WRMTL that enforces a task weight for Task 3 as large as 100, which diverges itself from the first two tasks. At the same time Task 1 and Task 2, as similar tasks, are sharing their samples properly so that the final results almost reveal the true classification boundaries for all three tasks. WRMTL gives the highest accuracy, especially when $\mu_t = (1, 1, 100)$. In this setting, even though $T_3$ is deemed as the outlier task, due to the random selection of training set, it could be slightly related with other tasks, but usually remain independence of it could be a wise choice.

![Fig. 2. Overall performance for separated SVM, one SVM, regularized MTL and weighted tasks RMTL.](image)

### B. Rod Pump Failure Prediction

We also tested our algorithm on a real-world equipment failure prediction problem on data gathered from instrumented artificial lift systems in the petroleum industry. Artificial lift techniques are widely used in petroleum industry to enhance production of reservoirs with energy levels too low to directly lift fluids to the surface. Successful early failure predictions can dramatically improve oil recovery performance. This allows the petroleum engineer to adjust operating parameters to forestall failures or by scheduling maintenance to reduce unplanned repairs and to minimize downtime. Pump-off controllers (POCs) are used to monitor daily down-hole performance of each oil production well. Databases are used to store multi-dimensional sensor time series as well as event logs. The database includes data from multiple oil fields accumulated across the enterprise. Depending on the geology and geometry of the oil reservoir, as well as on the difference in artificial lift equipment, the equipment failure signals may be different. This makes this data set idea for multitask learning.

The reasons for rod pump failures can be broadly classified into two main categories: mechanical and chemical. Mechanical failures are caused by improper design, by improper manufacturing, or by wear and tear during operations. Well conditions may contribute to excessive wear and tear, such as sand intrusions, gas pounding, rod cutting and asphalting. Chemical failures are caused by the corrosive nature of the fluid being pumped through the systems. For example, the fluid may contain H2S or bacteria. For rod pumps one of the major mechanical failures is called Tubing Failure, where its tubing is leaking because of accumulated mechanical frictions and cutting events. A cutting event happens when the rod frequently cuts the tubing during its movement which may ultimately cause a tubing hole. The cutting event may cause the monitored load to change. But because of uncertainties of real-world cases, e.g. the non-straightness of downhole tubings, the strength of abrasions, it may impact several attributes in different ways. If a tubing hole appears, fluid inside the tubing will leak out that causes a severe reduce of load, and may
experience longer run time because the pump is not producing enough fluid. Trends of the attributes are commonly used as features which reveal the “increase” or “decrease” scale for each parameter after a certain events. Besides tubing failure, another common downhole failure type called Rod Pump Failure is also included because of the pump malfunction.

The goal is to predict down-hole tubing leaks of sucker rod pump production wells on a daily basis across heterogeneous oil fields. A dataset has been constructed by the SMEs who picked 50 well studied failure cases across 4 fields and extracted the trends from different attributes monitored from sensors. The attributes include the pump load, runtime, cycles and several relevant metrics which are commonly used to identify failures in petroleum industry. The extracted trends serve as features - both long-term trends and short-term trends for each attribute [4]. Besides failures, there are 300 normal wells in the dataset which may contain noise that looks like failures. Each well has at least 100 records. For each well failure, it is labeled according to the progress of the development of the failure - such as a rod cutting event happens consistently which lead to a tubing leak, or a pump friction happens that lead to a rod pump failure. Failure wells also contains normal examples. Among this dataset, all sudden failures, which happens because of accumulated mechanical strain that would not exhibit any anomalous signals before the events, are excluded. Fig. 4 shows an example of what the data looks like for a tubing failure.

It is commonly understood that different oil fields would bias the data distribution - different geographic locations, well depth, geological formations, productions and even failure handling actions because different fields are managed by different teams. Therefore for the 4 fields from the dataset, it can be naturally divided into 4 different tasks. However, failures are generally rare events. Similar failure patterns, even if they are in different fields, are very possible to mean the same failure types, e.g. for tubing leaks, it always follows a significant decline in pump load. It is also intuitive that similar failure patterns can be shared among different fields.

The data set is very noisy. The noise comes from multiple sources, which include both natural and man-made causes. The wells operate in rough physical environments which tend to cause equipment to break down. For example, lightning strikes
Fig. 4. Sample data from a single oil production well which experiences a failure. Four attributes are shown including Card Area, Peak Load, Min Load and Daily Run Time.

can sometimes disrupt the wireless communication networks on which the data collected by POC sensors are not sent to the centralized logging database. This would result in missing values in the data. Also, petroleum engineering field workers would sometimes perform maintenance and make calibration adjustments to the equipment. These maintenance activities and adjustments would cause the sensor measurements to change. Workers typically diligently log their work in downtime and workover database tables. Occasionally a log entry would be delayed, or would be not logged at all. For example, the workers occasionally would recalibrate the POC sensors. The results of the recalibration may introduce drastic changes to sensor readings. Nevertheless, it is not standard practice to record such recalibration.

Previous works have been done on similar dataset and it is known that the state-of-the-art algorithm is based on a semi-supervised support vector machines. We tested our algorithm together with the state-of-the-art algorithm and other algorithms using cross-validation. To demonstrate how fields differ for common classification algorithms, we employ leave-one-field-out cross-validation. Fig. 5 shows the results for all failure classes precision and recall using leave-one-field-out, original semi-supervised support vector machine, regularized multitask support vector machines, as well as weighted regularized multitask support vector machines. RBF kernel and same $C = 1, \gamma = 0.4$ values are used for all algorithms. For the two multitask learning, we set $\mu = 1.0$, while for WRMTL, we set its $\lambda$ vector as the symmetric Kullback-Liebler divergence. We consider each task as a single domain and calculate its distribution for each feature, then based on the sum of the KL-divergence of the corresponding features, we collect the $\lambda_t$ values. We take $\lambda_0 = 1.0$.

From the results, we can clearly observe that leave-one-field-out yields the worst performance, which proves our hypothesis that different fields are very different for the learning task. The state-of-the-art solution yields one global model using a single SVM. But because it generalizes over all fields, the results are “over-smoothed”. While we enable the multitask learning setting, RMTL and WRMTL shows consistently better precision and recall from each failure classes. Since WRMTL adapts KL-divergence for fine-tuning the task regularization parameters, it produces $1-5\%$ better than MTL from both evaluation metrics.

IV. DISCUSSION

We have presented our weighted regularized multitask learning framework which is a generalized framework of RMTL. Like its predecessor, this framework employs a single task learning, which reuses a common support vector machine with self-defined kernels.

It is able to learn from multiple tasks simultaneously and transfer knowledge in among tasks that potentially alleviate the lack of observations in noisy environments. We tested our algorithm on both synthetic data and a real-world dataset from petroleum industry. The result on the real-world data is better than the state-of-the-art solution, as well as regularized multitask learning. Our algorithm shows a robust learning capability that handles outlier tasks without complex metrics using only KL-divergence as the task weight indicator. This is effective especially when there are few tasks, and when some prior knowledge is available.

Besides KL-divergence, other distance metrics may also be used. We leave that as future work.

V. CONCLUSION

We introduced a generalized regularized multitask learning method that encodes finer-grained task relationships. We
proved multitask learning property holds over this method, and the updated kernels of dual form exhibit the generalized form. By experiments on both synthetic data and real-world data, we showed that our methodology is superior to some traditional regularized multitask learning, as well as to single-task learning approaches. In our future work, we will compare with other extensions of MTL framework, such as GO-MTL [17] and online MTL [18], etc.

ACKNOWLEDGMENT

We would like to thank Chevron for providing us the data and their domain expertise. This work is supported by Chevron Corp. under the joint project, Center for Interactive Smart Oileld Technologies (CiSoft), at the University of Southern California.

REFERENCES