Robust Multi-Task Learning with t-Processes

Shipeng Yu[†] SHIPENG.YU@SIEMENS.COM
CAD and Knowledge Solutions, Siemens Medical Solutions, Malvern, PA 19355, USA

Volker Tresp VOLKER.TRESP@SIEMENS.COM
Corporate Technology, Siemens AG, Munich 81730, Germany

Kai Yu[†] KYU@SV.NEC-LABS.COM
NEC Laboratories America, Cupertino, CA 95014, USA

Abstract

Most current multi-task learning frameworks ignore the *robustness* issue, which means that the presence of "outlier" tasks may greatly reduce overall system performance. We introduce a robust framework for Bayesian multi-task learning, t-processes (TP), which are a generalization of Gaussian processes (GP) for multi-task learning. TP allows the system to effectively distinguish good tasks from noisy or outlier tasks. Experiments show that TP not only improves overall system performance, but can also serve as an indicator for the "informativeness" of different tasks.

1. Introduction

Multi-task learning is based on the assumption that multiple tasks *share certain structures*, e.g., hidden units in neural networks (Caruana, 1997), common feature mappings (Ando & Zhang, 2005; Zhang et al., 2005; Argyriou et al., 2006), regularizations (Evgeniou & Pontil, 2004), or covariance structure in a hierarchical Bayesian perspective (Bakker & Heskes, 2003; Yu et al., 2005). Therefore, tasks can mutually benefit from these shared structures.

Most multi-task learning systems implicitly or explicitly assume that all the tasks are *equally important*, for instance, by equally weighting them in a regularization framework. This assumption does not hold for many real world problems, due to different measurement noises, different task functionalities or intentions. One example is in movie rating systems where the goal is to predict a user's preferences for a set of movies. Multi-task learning is well-suited for this scenario because users often share interests (cf. Yu et al., 2006). However, "malicious" or "careless" users who intentionally or unintentionally rate movies poorly may contaminate the system to such an extent that the overall performance is degraded. Therefore, it is important to distinguish good tasks from noisy or *outlier* tasks (the malicious or careless users) and thus improve the *robustness* of multi-task learning system. We call this the *robust multi-task learning* problem.

In this paper we introduce t-processes (TP), a generalization of Gaussian processes (GP), for robust multi-task learning. TP defines a nonparametric Bayesian prior over functions, and extends GP in the sense that given any fixed number of data points, the function values are sampled not from a Gaussian, but from a multivariate t distribution. It is well-known that multivariate t is implicitly an infinite Gaussian mixture and is more robust than the Gaussian. We study some basic properties of TP in Section 2. We show that TP is a robust version of GP for multi-task learning (Section 3), and that learning and inference can be done effectively using variational Bayes (Section 4). After further discussions (Section 5), we present empirical results on synthetic data and two real world problems (movie rating and temperature prediction) in which TP outperforms GP for robust multi-task learning and also recovers "informativeness" of each task (Section 6).

2. The t-Processes

We begin by reviewing the multivariate Gaussian, t distributions and the Gaussian processes.

Appearing in *Proceedings of the 24th International Conference on Machine Learning*, Corvallis, OR, 2007.

[†] This work was done when both the first and third authors were with Corporate Technology, Siemens AG, Germany.

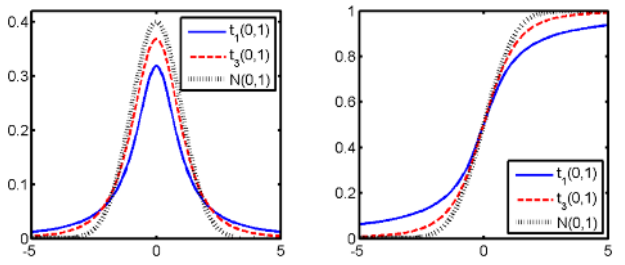


Figure 1. P.d.f. (left) and c.d.f. (right) of one dimensional t distribution $t_1(0,1)$, $t_3(0,1)$ and $\mathcal{N}(0,1) = t_{+\infty}(0,1)$.

2.1. Multivariate Gaussian and t Distributions

A random variable $\mathbf{x} \in \mathbb{R}^d$ is said to follow a multivariate Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ if the p.d.f. is

$$P(\mathbf{x}) = (2\pi)^{-\frac{d}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right).$$

Here $\boldsymbol{\mu} \in \mathbb{R}^d$ is the mean vector, and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is the positive definite covariance matrix. A well-known limitation of a Gaussian distribution is that it is not robust, since if the observations are contaminated by outliers, the accuracy of estimated $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ can significantly be compromised (see, e.g., Gelman et al., 1996). A more robust alternative is the multivariate t distribution, with the p.d.f. defined as

$$\pi^{-\frac{d}{2}} |\boldsymbol{\Sigma}|^{-\frac{1}{2}} \nu^{\frac{\nu}{2}} \frac{\Gamma(\frac{\nu+d}{2})}{\Gamma(\frac{\nu}{2})} \Big(\nu + (\mathbf{x}-\boldsymbol{\mu})^\top \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big)^{-\frac{\nu+d}{2}},$$

where $\Gamma(\cdot)$ is the Gamma function. We conventionally write $\mathbf{x} \sim t_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, with $\nu > 0$ being the degrees of freedom. The univariate Student-t is a special case with $d = 1$ and $\boldsymbol{\mu} = \mathbf{0}$. The t distribution is known to have "heavy tails" in its p.d.f. compared to a Gaussian distribution (see Figure 1).

It is well-known that samples from $t_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ can be obtained by repeatedly sampling the *latent variable* τ and $\mathbf{x}$ following

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \tfrac{1}{\tau}\boldsymbol{\Sigma}), \qquad \tau \sim \text{Gamma}(\tfrac{\nu}{2}, \tfrac{\nu}{2}),$$

where $\text{Gamma}(\alpha, \beta)$ is the Gamma distribution with density $\beta^\alpha \tau^{\alpha-1} \exp(-\beta\tau)/\Gamma(\alpha)$. This indicates that multivariate t can be realized as an infinite mixture of Gaussians, where all the Gaussian components have the same mean but different scales of the covariance.

The following propositions summarize some useful results for the multivariate t distribution. These can be easily proved using the latent variable interpretation (see Liu & Rubin, 1995; Kotz & Nadarajah, 2004).

Proposition 2.1. $\lim_{\nu\to+\infty} t_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$.

Proposition 2.2. *If* $\mathbf{w} \in \mathbb{R}^d \sim t_\nu(\boldsymbol{\mu}_w, \boldsymbol{\Sigma}_w)$, *then for any matrix* $\mathbf{X} \in \mathbb{R}^{n\times d}$, $\mathbf{X}\mathbf{w} \sim t_\nu(\mathbf{X}\boldsymbol{\mu}_w, \mathbf{X}\boldsymbol{\Sigma}_w\mathbf{X}^\top)$.

Proposition 2.3. *Let* $\mathbf{x} \sim t_\nu(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, *and let* $\mathbf{x} = \binom{\mathbf{x}_1}{\mathbf{x}_2}$, $\boldsymbol{\mu} = \binom{\boldsymbol{\mu}_1}{\boldsymbol{\mu}_2}$ *and* $\boldsymbol{\Sigma} = \left(\begin{smallmatrix}\boldsymbol{\Sigma}_{11} & \boldsymbol{\Sigma}_{12}\\ \boldsymbol{\Sigma}_{21} & \boldsymbol{\Sigma}_{22}\end{smallmatrix}\right)$ *be the* $[d_1, d-d_1]$ *partition of corresponding vectors and matrix. Then* $\mathbf{x}_1$ *and* $\mathbf{x}_2|\mathbf{x}_1$ *are independently distributed, with*

$$\mathbf{x}_1 \sim t_\nu(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_{11}), \quad \mathbf{x}_2|\mathbf{x}_1 \sim t_{\nu+d_1}(\boldsymbol{\mu}_{\mathbf{x}_2|\mathbf{x}_1}, \boldsymbol{\Sigma}_{\mathbf{x}_2|\mathbf{x}_1}),$$

where

$$\boldsymbol{\mu}_{\mathbf{x}_2|\mathbf{x}_1} = \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_1 - \boldsymbol{\mu}_1) + \boldsymbol{\mu}_2,$$
$$\boldsymbol{\Sigma}_{\mathbf{x}_2|\mathbf{x}_1} = \frac{\nu + (\mathbf{x}_1-\boldsymbol{\mu}_1)^\top \boldsymbol{\Sigma}_{11}^{-1}(\mathbf{x}_1-\boldsymbol{\mu}_1)}{\nu+d_1}\Big(\boldsymbol{\Sigma}_{22} - \boldsymbol{\Sigma}_{21}\boldsymbol{\Sigma}_{11}^{-1}\boldsymbol{\Sigma}_{12}\Big).$$

2.2. Gaussian Processes

A Gaussian process (GP) is a stochastic process that defines a nonparametric prior over functions in Bayesian statistics (Rasmussen & Williams, 2006). A random real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ follows a GP, denoted by $\mathcal{GP}(h, \kappa)$, if for every finite number of data points $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$, $\mathbf{f} = \{f(\mathbf{x}_i)\}_{i=1}^n$ follows a multivariate Gaussian $\mathcal{N}(\mathbf{h}, \mathbf{K})$ with mean $\mathbf{h} = \{h(\mathbf{x}_i)\}_{i=1}^n$ and covariance $\mathbf{K} = \{\kappa(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^n$. $h(\cdot)$ and $\kappa(\cdot,\cdot)$ are called the *mean function* and the *covariance function*, respectively, and κ needs to satisfy the Mercer's condition and is also called the *kernel* (Schölkopf & Smola, 2002). GP is widely applied to many learning problems. Please refer to (Rasmussen & Williams, 2006) for a comprehensive overview of GP models.

A GP framework for multi-task learning was introduced in (Schwaighofer et al., 2005; Yu et al., 2005). Each task is fully specified by a latent function f (with corrupted Gaussian noises), and all these functions share a common GP prior in the hierarchical Bayesian framework. In this way the multiple tasks can *collaborate* with each other. An EM algorithm was derived for learning the GP prior $\mathbf{h}$ and $\mathbf{K}$.

2.3. t-Processes

As shown, Gaussian is not robust and can suffer from "outlier" samples. Extending this to the functional space, GP for multi-task learning may also lack robustness if there are "outlier" tasks. We now define t-processes that improve the robustness of GP.

Definition 2.4 (t-Process)**.** *A random, real-valued function* $f : \mathbb{R}^d \to \mathbb{R}$ *is said to follow a* t-process (TP) *with* degrees of freedom $\nu > 0$, mean function $h(\cdot)$ *and* covariance function $\kappa(\cdot,\cdot)$, *if for any positive integer* n *and any* $\mathbf{x}_1, \ldots, \mathbf{x}_n \in \mathbb{R}^d$,

$$\mathbf{f} = [f(\mathbf{x}_1), \ldots, f(\mathbf{x}_n)]^\top \sim t_\nu(\mathbf{h}, \mathbf{K}),$$

with $\mathbf{h} = [h(\mathbf{x}_1), \ldots, h(\mathbf{x}_n)]^\top$, $\mathbf{K} = \{\kappa(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^n$.

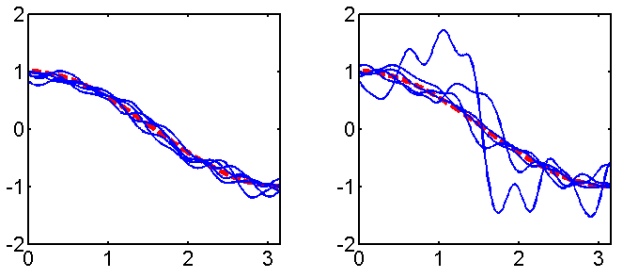


Figure 2. Five samples (blue solid) from $\mathcal{GP}(h,\kappa)$ (left) and $\mathcal{TP}_\nu(h,\kappa)$ (right), with $h(x)=\cos(x)$ (red dashed), $\kappa(x_i,x_j)=0.01\exp(-20(x_i-x_j)^2)$ and $\nu=5$.

This definition needs some justification, since it implicitly assumes that multivariate t distributions *keep marginals*, i.e., any marginal distribution of $t_\nu(\mathbf{h},\mathbf{K})$ is still multivariate t, with the same degrees of freedom ν and the corresponding part of $\mathbf{h}$ and $\mathbf{K}$ as mean and covariance. This is non-trivial for general distributions, but can be proved for multivariate t (see Prop. 2.3). We denote a sample from a TP as $f\sim\mathcal{TP}_\nu(h,\kappa)$.

The following results show some basic properties of TP and follow easily from Prop. 2.1 to 2.3.

Proposition 2.5 (Mixture Interpretation)**.** *Sampling $f\sim\mathcal{TP}_\nu(h,\kappa)$ is equivalent to two-step sampling*

$$f\sim\mathcal{GP}(h,\tfrac{1}{\tau}\kappa),\qquad \tau\sim\text{Gamma}(\tfrac{\nu}{2},\tfrac{\nu}{2}).$$

And moreover, $\lim_{\nu\to+\infty}\mathcal{TP}_\nu(h,\kappa)=\mathcal{GP}(h,\kappa)$.

Proposition 2.6 (Linear Interpretation)**.** *In a linear system $f(\mathbf{x})=\mathbf{w}^\top\mathbf{x}$, if $\mathbf{w}\sim t_\nu(\boldsymbol{\mu}_w,\boldsymbol{\Sigma}_w)$, then $f\sim\mathcal{TP}_\nu(h,\kappa)$ with $h(\mathbf{x})=\boldsymbol{\mu}_w^\top\mathbf{x}$, $\kappa(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^\top\boldsymbol{\Sigma}_w\mathbf{x}_j$.*

Proposition 2.7 (Posterior Process)**.** *If the prior process $f\sim\mathcal{TP}_\nu(h,\kappa)$, then conditioned on a length-n vector $\mathbf{f}_n=[f(\mathbf{x}_1),\ldots,f(\mathbf{x}_n)]^\top$, the posterior process $f^*|\mathbf{f}_n\sim\mathcal{TP}_{\nu+n}(h^*,\kappa^*)$, where*

$$h^*(\mathbf{x})=\mathbf{k}_\mathbf{x}^\top\mathbf{K}_n^{-1}(\mathbf{f}_n-\mathbf{h}_n)+h(\mathbf{x})$$
$$\kappa^*(\mathbf{x}_i,\mathbf{x}_j)=\frac{\nu+(\mathbf{f}_n-\mathbf{h}_n)^\top\mathbf{K}_n^{-1}(\mathbf{f}_n-\mathbf{h}_n)}{\nu+n}\left(\kappa_{ij}-\mathbf{k}_{\mathbf{x}_i}^\top\mathbf{K}_n^{-1}\mathbf{k}_{\mathbf{x}_j}\right)$$

with $\mathbf{h}_n=[h(\mathbf{x}_1),\ldots,h(\mathbf{x}_n)]^\top$, $\mathbf{K}_n=\{\kappa(\mathbf{x}_s,\mathbf{x}_t)\}_{s,t=1}^n$, $\mathbf{k}_\mathbf{x}=[\kappa(\mathbf{x},\mathbf{x}_1),\ldots,\kappa(\mathbf{x},\mathbf{x}_n)]^\top$, *and* $\kappa_{ij}=\kappa(\mathbf{x}_i,\mathbf{x}_j)$.

Given a mean function h and a covariance function κ, $\mathcal{TP}_\nu(h,\kappa)$ defines a set of *robust processes*, in which each process has a desired *robustness level* denoted by the (inverse of) degrees of freedom ν. In general the bigger the ν, the smaller the robustness control of the process. Prop. 2.5 shows that TP defines an *infinite mixture* of GPs, and also converges to a GP as ν goes to infinity. This also means that GP is a special case of TP without any robustness control. If the Gamma variable τ happens to be small, the sampled function will look "noisy" (see examples in Figure 2).

Similar to the linear interpretation of GP, Prop. 2.6 shows that a linear predictive model has a TP interpretation if its linear weights follow a multivariate t prior. Another important result is given in Prop. 2.7, which says the posterior process of a TP is another TP. Note that the new TP has not only new mean and covariance functions, but also new degrees of freedom $\nu+n$ which depend on the sample size n. It is easy to check that when ν goes to infinity, Prop. 2.7 recovers the posterior process of $\mathcal{GP}(h,\kappa)$. It provides additional insights to compare the posterior process of $\mathcal{TP}_\nu(h,\kappa)$ and that of $\mathcal{GP}(h,\kappa)$. They have the same posterior mean, but the posterior covariance of TP is a *rescaling* of that of the corresponding GP. The square of the Mahalanobis distance, $d^2_{\mathbf{K}_n}(\mathbf{f}_n,\mathbf{h}_n)=(\mathbf{f}_n-\mathbf{h}_n)^\top\mathbf{K}_n^{-1}(\mathbf{f}_n-\mathbf{h}_n)$, measures the *explainability* of current parameter (h,κ) to the observation $\mathbf{f}_n$. The scale of the posterior covariance decreases with decreasing Mahalanobis distance. ν acts as a *smoothing factor* for this adjustment. As more observations are available, the posterior degrees of freedom increase, and the posterior process tends to have less robustness control. So when observations are sufficient, the posterior covariance is able to reflect the uncertainty and no robustness control is necessary.

2.4. t-Processes with Noisy Observations

In real world applications, we normally cannot observe the function values $f(\mathbf{x}_i)$ directly, but only y_i which introduces additional noise or transformation. Since TP only replaces the nonparametric prior for f, all noise models for GP can be potentially used for TP. For instance, in regression tasks we can assume $f(\mathbf{x}_i)$ is corrupted with Gaussian noise, i.e., $P(y_i|f(\mathbf{x}_i))\sim\mathcal{N}(f(\mathbf{x}_i),\sigma^2)$; in binary classification tasks we can take any sigmoid function $\lambda(\cdot)$ such that $P(y_i|f(\mathbf{x}_i))=\lambda(y_if(\mathbf{x}_i))$. See Section 5 for more discussions on this.

3. Multi-Task Learning with TP

We follow the notations in (Yu et al., 2005) to describe our robust multi-task learning with TP. For simplicity we only consider regression tasks with Gaussian noise, but the entire framework is easily extended to other noise models. Let there be m tasks, and task ℓ has outputs/labels $\mathbf{y}_\ell$ on an item set $\mathbf{X}_\ell$ of size n_ℓ. Let $\mathbf{X}=\{\mathbf{x}_1,\ldots,\mathbf{x}_n\}\supset\bigcup_{\ell=1}^m\mathbf{X}_\ell$ be the total item set, and $\mathbf{Y}=\{\mathbf{y}_1,\ldots,\mathbf{y}_m\}$ be the outputs of all tasks. The indices of $\mathbf{X}$ that $\mathbf{X}_\ell$ contains are denoted in $\mathbb{I}_\ell$. Suppose there is a latent function f_ℓ underlying each task ℓ, which generates the label $\mathbf{y}_\ell$ independent of

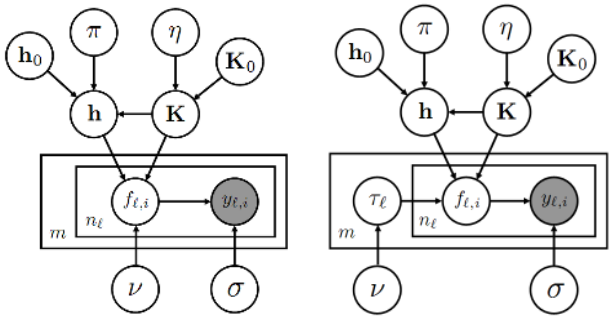


Figure 3. Graphical models for TP multi-task learning (left) and the infinite mixture interpretation (right).

other tasks. To allow that the multiple tasks *share some information* with each other, we assume all these m latent functions share the same TP prior $\mathcal{T}P_\nu(h, \kappa)$. The sampling process is as follows:

$$\mathbf{y}_\ell|\mathbf{f}_\ell, \sigma^2 \overset{\text{ind}}{\sim} \mathcal{N}(\mathbf{f}_\ell, \sigma^2\mathbf{I}), \quad \ell = 1, \ldots, m;$$
$$\mathbf{f}_\ell|\nu, h, \kappa \overset{\text{iid}}{\sim} t_\nu(\mathbf{h}_\ell, \mathbf{K}_{\ell,\ell}).$$

Here we denote $\mathbf{f}_\ell = \{f_\ell(\mathbf{x}_i)\}_{i\in\mathbb{I}_\ell}$, $\mathbf{h}_\ell = \{h(\mathbf{x}_i)\}_{i\in\mathbb{I}_\ell}$, $\mathbf{K}_{\ell,\ell} = \{\kappa(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j\in\mathbb{I}_\ell}$, and $\mathbf{I}$ the identity matrix. Since multivariate t has an interpretation of infinite Gaussian mixture, the equivalent sampling process is:

$$\mathbf{y}_\ell|\mathbf{f}_\ell, \sigma^2 \overset{\text{ind}}{\sim} \mathcal{N}(\mathbf{f}_\ell, \sigma^2\mathbf{I}), \quad \ell = 1, \ldots, m;$$
$$\mathbf{f}_\ell|\tau_\ell, h, \kappa \overset{\text{ind}}{\sim} \mathcal{N}(\mathbf{h}_\ell, \tfrac{1}{\tau_\ell}\mathbf{K}_{\ell,\ell}), \quad \tau_\ell|\nu \overset{\text{iid}}{\sim} \text{Gamma}(\tfrac{\nu}{2}, \tfrac{\nu}{2}).$$

This formulation turns out to be useful in later sections for learning and inference. Let $\mathbf{h} = \{h(\mathbf{x}_i)\}_{i=1}^n$, $\mathbf{K} = \{\kappa(\mathbf{x}_i, \mathbf{x}_j)\}_{i,j=1}^n$, then the conditional log-likelihood of the data can be written as

$$\log P(\mathbf{Y}|\mathbf{X}) = \sum_\ell \log \int P_N(\mathbf{y}_\ell|\mathbf{f}_\ell, \sigma^2) P_t(\mathbf{f}_\ell|\nu, \mathbf{h}, \mathbf{K})\, d\mathbf{f}_\ell$$

with

$$P_t(\mathbf{f}_\ell|\nu, \mathbf{h}, \mathbf{K}) = \int P_N\left(\mathbf{f}_\ell \middle| \mathbf{h}, \tfrac{1}{\tau_\ell}\mathbf{K}\right) P_G\left(\tau_\ell \middle| \tfrac{\nu}{2}, \tfrac{\nu}{2}\right) d\tau_\ell.$$

Here P_N, P_t and P_G denote Gaussian, multivariate t and Gamma distributions, respectively. Note that we write $\mathbf{h}$, $\mathbf{K}$ instead of $\mathbf{h}_\ell$, $\mathbf{K}_{\ell,\ell}$ for the ℓ-th task since all the tasks share the same TP parameters.

A standard way of fitting h and κ is to assume $h \equiv 0$ and fit a parametric form for κ (e.g., Gaussian kernel), but as pointed out by (Yu et al., 2005), parameterizing kernel in this way limits the flexibility of multi-task learning, and the learned kernel may not be able to reflect the covariance structure shared among all the tasks. Therefore, we instead assign a conjugate prior to the (finite) TP prior $(\mathbf{h}, \mathbf{K})$, which takes a Normal-Inverse-Wishart distribution: $P(\mathbf{h}, \mathbf{K}) = \mathcal{N}(\mathbf{h}; \mathbf{h}_0, \frac{1}{\pi}\mathbf{K})\, \mathcal{IW}(\mathbf{K}; \eta, \mathbf{K}_0)$, where the parameters $\mathbf{h}_0$ and $\mathbf{K}_0$ are respectively the *prior mean* and *base kernel*, and π, η correspond to the equivalent sample sizes before we observe any data (Schwaighofer et al., 2005; Yu et al., 2005). For the *maximum a posteriori* (MAP) estimate of $\mathbf{h}$ and $\mathbf{K}$, they correspond to a smooth term in the learning process (cf. Section 4). The final graphical model is shown in Figure 3, with model parameters $\mathbf{\Theta} = \{\sigma^2, \nu, \mathbf{h}, \mathbf{K}\}$ and hyperparameters $\{\mathbf{h}_0, \mathbf{K}_0, \pi, \eta\}$.

4. Learning and Inference

Learning a multi-task TP is more complicated than learning a multi-task GP due to the multivariate t prior for each latent function $\mathbf{f}_\ell$. To simplify learning we treat both $\mathbf{f}_\ell$ and τ_ℓ as latent variables and apply *variational Bayes* (VB) learning (Jordan et al., 1999). In the following we first discuss learning without missing labels (i.e., $\mathbf{y}_\ell$ is fully labeled for each task ℓ, $n_\ell = n$), and then turn to the general (and more realistic) case with missing labels.

4.1. Learning without Missing Labels

Given input data $\mathbf{X}$, fully observed labels $\mathbf{Y}$ and model parameters $\mathbf{\Theta}$, the joint posterior $P(\{\mathbf{f}_\ell, \tau_\ell\})$ is

$$\frac{1}{Z}\prod_\ell P_N(\mathbf{y}_\ell|\mathbf{f}_\ell, \sigma^2) P_N\left(\mathbf{f}_\ell \middle| \mathbf{h}, \tfrac{1}{\tau_\ell}\mathbf{K}\right) P_G\left(\tau_\ell \middle| \tfrac{\nu}{2}, \tfrac{\nu}{2}\right)$$

which is intractable due to the intractability of the normalization term Z. In the VB setting, we approximate this posterior with a *factorized* form

$$Q(\{\mathbf{f}_\ell, \tau_\ell\}) = \prod_\ell P_N(\mathbf{f}_\ell|\boldsymbol{\mu}_\ell, \mathbf{C}_\ell) P_G(\tau_\ell|\alpha_\ell, \beta_\ell) \qquad (1)$$

in which $\{\boldsymbol{\mu}_\ell, \mathbf{C}_\ell, \alpha_\ell, \beta_\ell\}$ are *variational parameters*, with $\boldsymbol{\mu}_\ell \in \mathbb{R}^{n_\ell}$, $\mathbf{C}_\ell \in \mathbb{R}^{n_\ell \times n_\ell}$, $\alpha_\ell, \beta_\ell > 0$. Then in the E-step of the VB learning, we minimize the Kullback-Leibler (KL) divergence of Q and P, $\int Q \log \frac{Q}{P}\, d\mathbf{f}_\ell\, d\tau_\ell$, w.r.t. these variational parameters by setting the corresponding derivatives to 0. This is equivalent to maximizing a lower bound of the data log-likelihood. This leads to the following iterative updates:

$$\boldsymbol{\mu}_\ell = \mathbf{C}_\ell\left(\tfrac{1}{\sigma^2}\mathbf{y}_\ell + \tfrac{\alpha_\ell}{\beta_\ell}\mathbf{K}^{-1}\mathbf{h}\right), \quad \mathbf{C}_\ell = \left(\tfrac{1}{\sigma^2}\mathbf{I} + \tfrac{\alpha_\ell}{\beta_\ell}\mathbf{K}^{-1}\right)^{-1}$$
$$\alpha_\ell = \tfrac{\nu+n}{2}, \quad \beta_\ell = \tfrac{\nu + (\boldsymbol{\mu}_\ell - \mathbf{h})^\top \mathbf{K}^{-1}(\boldsymbol{\mu}_\ell - \mathbf{h}) + \text{tr}(\mathbf{K}^{-1}\mathbf{C}_\ell)}{2}$$

where $\text{tr}(\cdot)$ denotes matrix trace. In the M-step, the KL divergence is minimized w.r.t. model parameters $\mathbf{\Theta}$. Here we fix the degrees of freedom ν (see Section 5

for numerical update and discussions), and update σ^2 with ML estimate and $\mathbf{h}$, $\mathbf{K}$ with MAP estimates. The update equations are as follows:

$$\mathbf{h} = \frac{1}{\pi + \sum_\ell \frac{\alpha_\ell}{\beta_\ell}} \left(\pi \mathbf{h}_0 + \sum_\ell \frac{\alpha_\ell}{\beta_\ell} \boldsymbol{\mu}_\ell \right), \quad (2)$$

$$\mathbf{K} = \frac{1}{\eta + m} \Big(\pi (\mathbf{h} - \mathbf{h}_0)(\mathbf{h} - \mathbf{h}_0)^\top + \eta \mathbf{K}_0 + \sum_\ell \frac{\alpha_\ell}{\beta_\ell} \left[\mathbf{C}_\ell + (\boldsymbol{\mu}_\ell - \mathbf{h})(\boldsymbol{\mu}_\ell - \mathbf{h})^\top \right] \Big), \quad (3)$$

$$\sigma^2 = \frac{1}{mn} \sum_\ell [\|\mathbf{y}_\ell - \boldsymbol{\mu}_\ell\|^2 + \mathrm{tr}(\mathbf{C}_\ell)], \quad (4)$$

where $\|\cdot\|$ denote vector 2-norm. The whole VB algorithm iterates E-step and M-step until convergence.

These update equations are similar to those for multi-task GP model (cf. Yu et al., 2005), but the differences provide additional insights to TP multi-task learning:

- In E-step, updates for $\boldsymbol{\mu}_\ell$ and $\mathbf{C}_\ell$ take into account the weight $\frac{\alpha_\ell}{\beta_\ell}$ which is *the expected scale variable* τ_ℓ for the ℓ-th task.
- The expected τ_ℓ, i.e., $\frac{\alpha_\ell}{\beta_\ell}$, gets smaller if the ℓ-th task is *poorly explained* by the shared common structure $\mathbf{h}$ and $\mathbf{K}$, i.e., if $(\boldsymbol{\mu}_\ell - \mathbf{h})^\top \mathbf{K}^{-1} (\boldsymbol{\mu}_\ell - \mathbf{h})$ is big (the task mean is "far away" from shared mean) or $\mathrm{tr}(\mathbf{K}^{-1}\mathbf{C}_\ell)$ is big (the task covariance is "far away" from shared covariance) or both. This means outlier tasks are automatically penalized.
- ν acts as a *smoothing term* for updates of τ_ℓ. The bigger the ν, the smaller the penalties for outlier tasks.
- In M-step, the shared mean $\mathbf{h}$ and covariance $\mathbf{K}$ are *weighted averages* (with weights given in $\frac{\alpha_\ell}{\beta_\ell}$) of all tasks and the hyperprior.
- With $\nu \to +\infty$, all the update equations reduce to those for multi-task GP model (Yu et al., 2005).

4.2. Learning with Missing Labels

When there are missing labels, standard VB solution is to treat them as missing data and estimate them as well in the E-step. This leads to:

$$\boldsymbol{\mu}_\ell = \mathbf{K}_{n,\ell} \mathbf{R}_\ell (\mathbf{y}_\ell - \mathbf{h}_\ell) + \mathbf{h},$$
$$\mathbf{C}_\ell = \frac{\beta_\ell}{\alpha_\ell} \left(\mathbf{K} - \mathbf{K}_{n,\ell} \mathbf{R}_\ell \mathbf{K}_{n,\ell}^\top \right), \quad (5)$$

$$\alpha_\ell = \frac{\nu + n_\ell}{2}, \quad \beta_\ell = \frac{\nu + (\mathbf{y}_\ell - \boldsymbol{\mu}_\ell)^\top \mathbf{R}_\ell \mathbf{K}_{\ell,\ell} \mathbf{R}_\ell (\mathbf{y}_\ell - \boldsymbol{\mu}_\ell) + \sigma^2 \,\mathrm{tr}(\mathbf{R}_\ell)}{2}$$

with $\mathbf{R}_\ell = (\mathbf{K}_{\ell,\ell} + \sigma^2 \frac{\alpha_\ell}{\beta_\ell} \mathbf{I})^{-1}$ and $\mathbf{K}_{n,\ell} = \mathbf{K}(:, \mathbb{I}_\ell)$ the $n \times n_\ell$ sub-matrix of $\mathbf{K}$. Note that we only need to inverse an $n_\ell \times n_\ell$ matrix for the ℓ-th task. In the M-step $\mathbf{h}$ and $\mathbf{K}$ are updated as before, and the noise level is now

$$\sigma^2 = \frac{1}{m \sum_\ell n_\ell} \sum_\ell \Big[\|\mathbf{y}_\ell - \boldsymbol{\mu}_\ell(\mathbb{I}_\ell)\|^2 + \mathrm{tr}(\mathbf{C}_\ell(\mathbb{I}_\ell, \mathbb{I}_\ell)) \Big]. \quad (6)$$

Only the sub-vector $\boldsymbol{\mu}_\ell(\mathbb{I}_\ell)$ and sub-matrix $\mathbf{C}_\ell(\mathbb{I}_\ell, \mathbb{I}_\ell)$ enter the calculation here since only these n_ℓ labels are observed in $\mathbf{y}_\ell$. The final algorithm is shown in Algorithm 1. The time complexity is $\mathcal{O}(m(n\hat{n}^2 + \hat{n}^3))$ where $\hat{n} = \max\{n_\ell\}$, similar to that of a GP model.

Algorithm 1 Robust Multi-Task Learning

Require: A size-n item set with input features $\mathbf{X} \in \mathbb{R}^{n \times d}$.
Require: m tasks of partial labels $\mathbf{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$, in which task ℓ labels a subset of $n_\ell \le n$ items.
1: Choose prior mean $\mathbf{h}_0$ (e.g., zero function), base kernel $\mathbf{K}_0$ (e.g., a Gaussian kernel), degrees of freedom $\nu > 0$, noise level $\sigma^2 > 0$, and hyperparameter $\pi > 0$, $\eta > 0$.
2: Initialize $\mathbf{h} = \mathbf{h}_0$ and $\mathbf{K} = \mathbf{K}_0$.
3: **repeat**
4: **for** $\ell = 1, \ldots, m$ **do**
5: Iterate (5) to obtain $\boldsymbol{\mu}_\ell$, $\mathbf{C}_\ell$, α_ℓ, β_ℓ for ℓ-th task.
6: **end for**
7: Update shared parameter $\mathbf{h}$, $\mathbf{K}$, σ^2 via (2), (3), (6).
8: **until** the improvement is smaller than a threshold.

4.3. Label Prediction

For label prediction, we wish to infer for a test point $\mathbf{x}^*$ the probability of its label y_ℓ^* for the ℓ-th task. After observing training data $\mathcal{D} = \{\mathbf{X}, \mathbf{Y}\}$, we have

$$P(y_\ell^* | \mathcal{D}, \boldsymbol{\Theta}) = \int P(y_\ell^* | f_\ell^*, \boldsymbol{\Theta}) P(f_\ell^* | \mathcal{D}, \boldsymbol{\Theta}) \, df_\ell^* \quad (7)$$

with $f_\ell^* = f_\ell(\mathbf{x}^*)$. Here $P(y_\ell^* | f_\ell^*, \boldsymbol{\Theta})$ is the noise model $\mathcal{N}(f_\ell^*, \sigma^2)$, and $P(f_\ell^* | \mathcal{D}, \boldsymbol{\Theta})$ can be calculated as

$$P(f_\ell^* | \mathcal{D}, \boldsymbol{\Theta}) = \int P(f_\ell^* | \mathbf{f}_\ell, \boldsymbol{\Theta}) P(\mathbf{f}_\ell | \mathcal{D}, \boldsymbol{\Theta}) \, d\mathbf{f}_\ell. \quad (8)$$

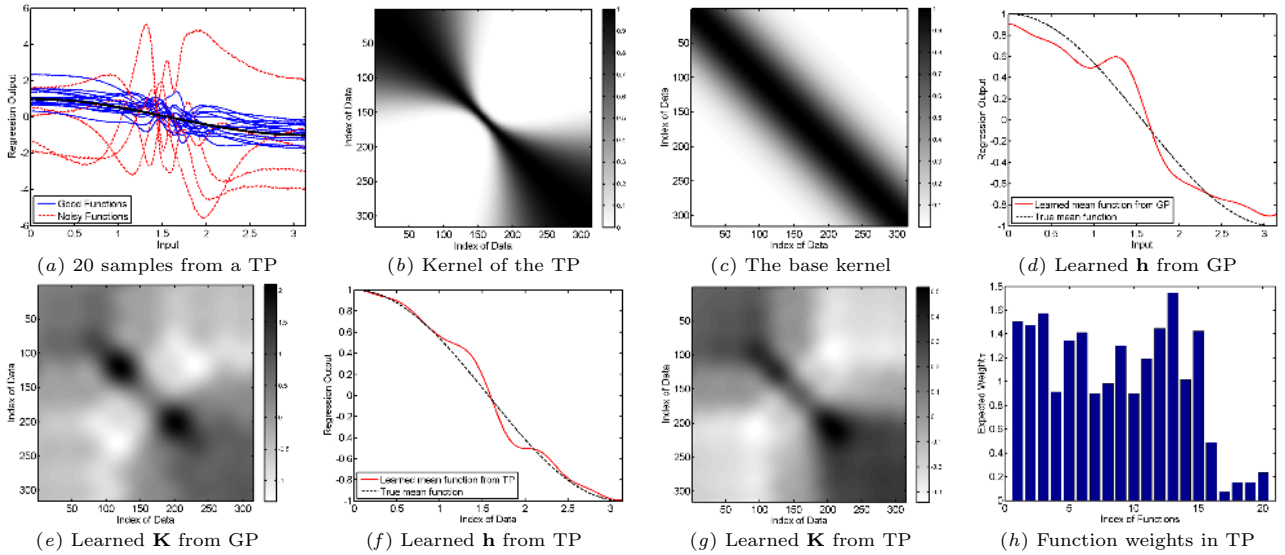
Prop. 2.3 says that $P(f_\ell^* | \mathbf{f}_\ell, \boldsymbol{\Theta})$ is $t_{\nu + n_\ell}(\mu_\ell^*, \sigma_\ell^{*2})$, with

$$\mu_\ell^* = \mathbf{k}^\top \mathbf{K}_{\ell,\ell}^{-1} (\mathbf{f}_\ell - \mathbf{h}_\ell) + h(\mathbf{x}^*),$$
$$\sigma_\ell^{*2} = \frac{\nu + (\mathbf{f}_\ell - \mathbf{h}_\ell)^\top \mathbf{K}_{\ell,\ell}^{-1} (\mathbf{f}_\ell - \mathbf{h}_\ell)}{\nu + n_\ell} [\kappa(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^\top \mathbf{K}_{\ell,\ell}^{-1} \mathbf{k}].$$

The problems are: (i) integral (8) is difficult to calculate; (ii) $h(\mathbf{x}^*)$ and $\kappa(\mathbf{x}^*, \mathbf{x}^*)$ might not be available for test data $\mathbf{x}^*$, since we are learning a finite TP posterior $(\mathbf{h}, \mathbf{K})$ which could be substantially different from the prior (h, κ).

To address the first problem, we can rewrite (7) as a *mixture of GP regression problems*, with the help of latent variable τ_ℓ:

$$P(y_\ell^* | \mathcal{D}, \boldsymbol{\Theta}) = \int P(\tau_\ell | \mathcal{D}, \boldsymbol{\Theta}) P(y_\ell^* | \tau_\ell, \mathcal{D}, \boldsymbol{\Theta}) \, d\tau_\ell. \quad (9)$$



(a) 20 samples from a TP (b) Kernel of the TP (c) The base kernel (d) Learned $\mathbf{h}$ from GP

(e) Learned $\mathbf{K}$ from GP (f) Learned $\mathbf{h}$ from TP (g) Learned $\mathbf{K}$ from TP (h) Function weights in TP

Figure 4. Multi-task learning on a 1D toy data using GP (d,e) and TP (f,g). See descriptions in Section 6.

Conditioned on τ_ℓ, $P(y^*_\ell|\tau_\ell, \mathcal{D}, \mathbf{\Theta})$ here is simply a GP regression problem with mean $\mathbf{h}$ and kernel $\frac{1}{\tau_\ell}\mathbf{K}$, which is a Gaussian $\mathcal{N}(\hat{\mu}^*_\ell, \hat{\sigma}^{*2}_\ell)$ with

$$\hat{\mu}^*_\ell = \mathbf{k}^\top(\mathbf{K}_{\ell,\ell} + \sigma^2\tau_\ell\mathbf{I})^{-1}(\mathbf{y}_\ell - \mathbf{h}_\ell) + h(\mathbf{x}^*),$$
$$\hat{\sigma}^{*2}_\ell = \tfrac{1}{\tau_\ell}\big[\kappa(\mathbf{x}^*, \mathbf{x}^*) - \mathbf{k}^\top(\mathbf{K}_{\ell,\ell} + \sigma^2\tau_\ell\mathbf{I})^{-1}\mathbf{k}\big].$$

$P(\tau_\ell|\mathcal{D}, \mathbf{\Theta})$ is our posterior belief of τ_ℓ, for which we can use the variational posterior Gamma$(\alpha_\ell, \beta_\ell)$. When the number of labeled data n_ℓ is large, the posterior Gamma will peak at mean $\frac{\alpha_\ell}{\beta_\ell}$ with small variance $\frac{\alpha_\ell}{\beta^2_\ell}$, and it suffices to use the mode $\frac{\alpha_\ell - 1}{\beta}$ for τ_ℓ and remove the integral in (9). Otherwise one may need to compute a one-dimensional integral numerically.

For the second problem, we distinguish two settings. In *transductive setting* where all the test data are available before learning, we can put them all into the training item set $\mathbf{X}$ and obtain $h(\mathbf{x}^*)$ and $\kappa(\mathbf{x}^*, \mathbf{x}^*)$ directly. In *inductive setting* where test data are unknown at learning phase or if there are too many such that the previous solution is unfeasible, Yu et al., 2005, Theorem 5.2 suggests that we can define $\hat{\boldsymbol{\alpha}}_\ell = \mathbf{K}_0^{-1}\mathbf{f}_\ell$ and assign a hyperprior to $\hat{\boldsymbol{\alpha}}_\ell$ instead of to $\mathbf{f}_\ell$ in the multi-task GP framework. In this way we can still recover $h(\mathbf{x}^*)$ and $\kappa(\mathbf{x}^*, \mathbf{x}^*)$ by means of the posterior structure of $\hat{\boldsymbol{\alpha}}_\ell$. This whole solution can be seamlessly transferred into the proposed TP framework, and we refer interested readers to (Yu et al., 2005) for details.

5. Discussion

Learning the Degrees of Freedom ν: In this paper we suggest fixing the degrees of freedom ν in the learning process, due to possible pitfalls of empirically estimating ν reported in (Fernandez & Steel, 1999). However if an empirical estimate of ν is desired, one can set the derivative of the log-likelihood w.r.t. ν to 0 in M-step and numerically solve for ν the equation

$$\tfrac{1}{m}\sum_\ell \Big(\ln\tfrac{\alpha_\ell}{\beta_\ell} - \tfrac{\alpha_\ell}{\beta_\ell} + \psi(\alpha_\ell) - \ln\alpha_\ell\Big) - \psi(\tfrac{\nu}{2}) + \ln\tfrac{\nu}{2} + 1 = 0$$

with $\psi(\cdot)$ the digamma function. Empirically we found estimating ν in this way leads to small ν (less than 1) in first VB steps and normally leads to bad local minima.

TP for Robust Linear Function Learning: Sometimes the functions we are interested are linear functions, i.e., $f_\ell(\mathbf{x}) = \mathbf{w}_\ell^\top\mathbf{x}$ with $\mathbf{w}_\ell \in \mathbb{R}^d$. Prop. 2.6 allows us to easily adapt the TP framework for linear functions. A similar VB algorithm can be derived for learning with time complexity $\mathcal{O}(m(nd^2 + d^3))$.

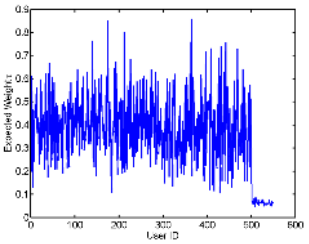
Double Robustness Control: We mainly discuss TP with Gaussian noise in this paper, but it is straightforward to consider other noise models. An interesting case is the t noise model, where $P(\mathbf{y}_\ell|\mathbf{f}_\ell)$ also follows a multivariate t distribution. This is the canonical robust regression for each function f_ℓ. Therefore, having a TP with a t noise model achieves *double robustness control*, in both task level and item level. Take a movie rating system as an example. The t noise model helps to uncover the "outlier" movies, which have large prediction variances w.r.t. a certain user; the TP helps to find the "outlier" users, who (intentionally or unintentionally) give ratings far away from the majorities and are not helpful to others. Empirically evaluating this joint model will be part of the future work.

6. Empirical Study

In Figure 4 we show a toy multi-task learning problem with TP on a 1D data set of 315 points (0 to π with 0.01 space). 15 "good" functions and 5 "noisy" functions are sampled from a given TP, with $\nu = 5$, mean $h(x) = \cos(x)$ (thick black line in (a)) and kernel $\mathbf{K}$ shown in (b). Gaussian noise model is used with $\sigma = 0.01$. This kernel is non-stationary such that every sampled function has a highly non-smooth part in the middle. For multi-task learning, we assume only 20% randomly selected data points are observed for each function, and our goal is to learn the common structure $\mathbf{h}$ and $\mathbf{K}$ for better prediction. A Gaussian kernel (c) is chosen as the base kernel, and the learned $\mathbf{h}$ and $\mathbf{K}$ are shown in (d), (e) for GP multi-task learning and (f), (g) for TP multi-task learning, respectively. It is seen that TP does a good job of recovering both $\mathbf{h}$ and $\mathbf{K}$, but GP is clearly biased by the noisy functions. In (h) we show the learned expected weight τ_ℓ for each function ℓ. Noisy functions 16 to 20 have lower weights compared to the others as expected.

6.1. Robust Collaborative Filtering

We apply TP on the MovieLens data set for robust collaborative filtering.[1] MovieLens contains 100,000 ratings for 1682 movies from 943 users, and as a preprocessing we select the 500 users with the most ratings (i.e., more likely to be "good" users) and end up with 927 movies by removing those rated less than 20 times. The "genres" information of the movies is used as features, from which a linear kernel is computed as the base kernel. Since we do not know which users are "noisy" users, we manually create 50 such users and see if TP can uncover them. In collaborative filtering a user can be "noisy" in many ways, and we tested the following three well-known possibilities: 1) He rates movies totally randomly (i.e., he is an uninformative user); 2) He intentionally rates a subset of movies high or low (i.e., he tries to increase or decrease the popularity of some movies); 3) He always gives high or low ratings (i.e., he is always satisfied or critical). TP can uncover all three types of users, and due to lack of space we only show results for the second scenario. Figure 5 (left) shows the learned weights τ (with 50 randomly sampled movie ratings observed for each user and TP parameter $\nu = 5$, $\pi = 1$, $\eta = 1$, $\sigma = 0.1$) for all users, and it is clear that the weights for users $501 \sim 550$ are low compared to the others. Some of the 500 users also have low weights, and we suspect they are also "noisy". A user ranking based on this weight might also be interesting for certain applications. Figure 5 (right) shows the rating prediction performance for the rest of ratings each user makes, and TP outperforms GP in all the evaluation metrics (averaged over 20 independent repeats).



	GP	TP
RMSE	1.0562	**1.0331**
MAE	0.7915	**0.7693**
MZOE	0.6308	**0.6196**

Figure 5. The learned user weights on MovieLens (left) including "noisy" users 501∼550, and rating prediction for GP and TP (right). The metrics are RMSE (root mean-square error), MAE (mean absolute error) and MZOE (mean zero-one error). All improvements are statistically significant (p-value 0.01 in Wilcoxon rank sum test).

6.2. Indoor Temperature Prediction

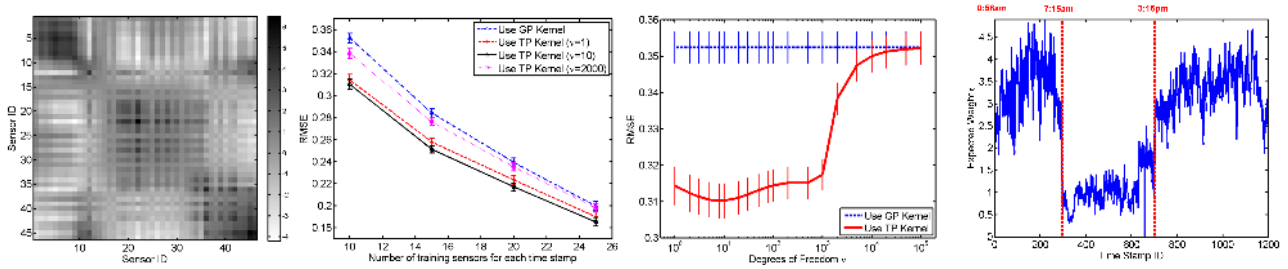
We also apply TP to a sensor network problem. Suppose a fixed number of sensors are placed at fixed indoor locations, and the temperature is read at a certain time interval for each of these sensors. The problem is to predict the temperature at certain sensors from the other sensor readings. This can be regarded as a multi-task learning problem, where each "data point" is a sensor and each "task" is a time stamp.

It is well-known that the *temperature correlation* w.r.t. these sensor locations plays a crucial role for temperature prediction. This can be represented via the kernel $\mathbf{K}$ in the multi-task learning view, where a GP with covariance matrix $\mathbf{K}$ is used to model the temperature readings. In reality, however, there might be certain time stamps at which temperature readings are noisy, due to abnormal weather conditions, unexpected human activities, mistakes by careless readers, etc. We need a robust model which can automatically penalize these "noisy" time-stamp readings and learn a clean kernel $\mathbf{K}$. The TP model is ideal for this purpose.

We consider the temperature data collected in Berkeley since 00:58:15, Feb. 28, 2004.[2] There were 54 sensors in the building, and temperature measurements were read at 30 seconds intervals. We use the data in the first 24 hours and end up with 46 sensors with 1730 time stamps after removing bad sensors and ignoring those time stamps with less than 30 readings. The first 1200 time stamps are used for training the

[1] The data is available at `http://www.grouplens.org/`.

[2] The data and the description are available at `http://www.cs.cmu.edu/~guestrin/Research/Data/`.



(a) Learned kernel from TP (b) RMSE vs no. training sensors k (c) RMSE vs ν with $k = 10$ (d) Expected τ for training tasks

Figure 6. Indoor temperature prediction results using TP models.

kernel $\mathbf{K}$, and the last 530 time stamps are left out for prediction. The learned kernel $\mathbf{K}$ (with $\nu = 10$, $\mathbf{h}_0 = \mathbf{0}$, $\mathbf{K}_0 = \mathbf{I}$, $\pi = 1$, $\eta = 1$ and $\sigma = 0.1$) is shown in Figure 6(a), in which strong correlations are found for sensors $1 \sim 10$, $15 \sim 35$ and $36 \sim 46$. We then fix the kernel and randomly pick $k = [10, 15, 20, 25]$ sensor readings as observations and predict the readings for each of the remaining 530 time stamps. (b) shows the performance measured in RMSE over 50 trials, with kernels trained using different degrees of freedom ν. TP kernels yield significantly smaller errors than the kernel trained with GP. In (c) we fix $k = 10$ and draw RMSE versus ν in log scale with error bars. TP error is much lower for small ν, and it goes down but then increases and approaches the GP error as ν goes from 1 to $+\infty$. Thus a good trade-off for ν can yield the best performance, and the prediction error is not sensitive w.r.t. degrees of freedom ν.

Finally (d) shows the expected weights of the training time stamps. We note that the time stamps in the middle have much lower weights than the others, namely from 7:15 a.m. to 3:15 p.m. While a final explanation is still missing, one reason might be that there were human activities (e.g., 8 hours regular working time) within the building which made the measurements noisy. Alternately, sensor correlation during day time might be quite different from that at night, which happens to be the correlation TP is learning. In this case TP suggests how to learn correlations using separated time stamps (or using a mixture of GPs) and provides a hint on how to separate the tasks. We also note that a time stamp (683) has a tiny weight (0.0001), and upon examination this may be due to reading errors (some sensors read 40, whereas the average temperature is 20). This indicates that TP can indeed detect outlier tasks.

Acknowledgement

The authors would like to thank Bharat Rao and Zoubin Ghahramani for valuable discussions.

References

Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, *6*.

Argyriou, A., Evgeniou, T., & Pontil, M. (2006). Multi-task feature learning. *NIPS'06*.

Bakker, B., & Heskes, T. (2003). Task clustering and gating for Bayesian multitask learning. *JMLR*, *4*, 83–89.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*, 41–75.

Evgeniou, T., & Pontil, M. (2004). Regularized multi-task learning. *Proceedings SIGKDD*.

Fernandez, C., & Steel, M. F. J. (1999). Multivariate Student-t Regression Models: Pitfalls and Inference. *Biometrika*, *86*, 153–167.

Gelman, A., Carlin, J. B., Stern, H., & Rubin, D. B. (1996). *Bayesian data analysis*. Chapman and Hall-CRC.

Jordan, M. I., Ghahramani, Z., Jaakkola, T., & Saul, L. K. (1999). An introduction to variational methods for graphical models. *Machine Learning*, *37*, 183–233.

Kotz, S., & Nadarajah, S. (2004). *Multivariate t-distributions and their applications*. Cambridge University Press.

Liu, C., & Rubin, D. B. (1995). ML Estimation of the t Distribution using EM and its Extensions, ECM and ECME. *Statistica Sinica*, *5*, 19–39.

Rasmussen, C. E., & Williams, C. K. I. (2006). *Gaussian processes for machine learning*. MIT Press.

Schölkopf, B., & Smola, A. J. (2002). *Learning with kernels*. MIT Press.

Schwaighofer, A., Tresp, V., & Yu, K. (2005). Hierarchical bayesian modelling with Gaussian processes. *NIPS'05*.

Yu, K., Tresp, V., & Schwaighofer, A. (2005). Learning Gaussian processes from multiple tasks. *ICML'05*.

Yu, S., Yu, K., Tresp, V., & Kriegel, H.-P. (2006). Collaborative ordinal regression. *ICML'06*.

Zhang, J., Ghahramani, Z., & Yang, Y. (2005). Learning multiple related tasks using latent independent component analysis. *NIPS'05*.