Parametric Task Learning

Authored by:

Masashi Sugiyama
Ichiro Takeuchi
Shinichi Nakajima
Tatsuya Hongo

Abstract

We introduce a novel formulation of multi-task learning (MTL) called parametric task learning (PTL) that can systematically handle infinitely many tasks parameterized by a continuous parameter. Our key finding is that, for a certain class of PTL problems, the path of optimal task-wise solutions can be represented as piecewise-linear functions of the continuous task parameter. Based on this fact, we employ a parametric programming technique to obtain the common shared representation across all the continuously parameterized tasks efficiently. We show that our PTL formulation is useful in various scenarios such as learning under non-stationarity, cost-sensitive learning, and quantile regression, and demonstrate the usefulness of the proposed method experimentally in these scenarios.

1 Paper Body

Multi-task learning (MTL) has been studied for learning multiple related tasks simultaneously. A key assumption behind MTL is that there exists a common shared representation across the tasks. Many MTL algorithms attempt to find such a common representation and at the same time to learn multiple tasks under that shared representation. For example, we can enforce all the tasks to share a common feature subspace or a common set of variables by using an algorithm introduced in [1, 2] that alternately optimizes the shared representation and the task-wise solutions. Although the standard MTL formulation can handle only a finite number of tasks, it is sometimes more natural to consider infinitely many tasks parameterized by a continuous parameter, e.g., in learning under non-stationarity [3] where learning problems change over continuous time, costsensitive learning [4] where loss functions are asymmetric with continuous cost balance, and quantile regression [5] where the quantile is a continuous variable between zero and one. In order to handle these infinitely many parametrized tasks, we propose in this paper an extended formulation of MTL called parametric-task learning (PTL). The key contribution of this paper is to show that, for a certain class of PTL problems, the optimal common representation shared across infinitely many parameterized tasks can be obtainable. Specifically, we develop an alternating minimization algorithm a' la [1, 2] for finding the entire continuum of solutions and the common feature subspace (or the common set of variables) among infinitely many parameterized tasks. Our algorithm exploits the fact that, for those classes of PTL problems, the path of task-wise solutions is piecewise-linear in the task parameter. We use the parametric programming technique [6, 7, 8, 9] for computing those piecewise linear solutions. 1

Notations: Let us denote by R, R+ , and R++ the set of real, nonnegative, and positive numbers, d respectively, while we define Nn := {1, . . . , n} for every natural number n. We denote by S++ the set of d ? d positive definite matrices, and let I(?) be the indicator function.

2

Review of Multi-Task Learning (MTL)

In this section, we review an MTL method developed in [1, 2]. Let {(xi , yi)}i?Nn be the set of n training instances, where xi ? X ? Rd is the input and yi ? Y is the output. We define wi (t) ? [0, 1], t ? NT as the weight of the ith instance for the tth task, where T is the number of tasks. We consider an affine model ft (x) = ?t,0 + ?t? x for each task, where ?t,0 ? R and ?t ? Rd . For notational simplicity, we define augmented vectors ?? := (?0 , ?1 , . . . , ?d)? ? Rd+1 ? := (1, x1 , . . . , xd)? ? Rd+1 , and write the affine model as ft (x) = ??t? x. ? and x The multi-task feature learning method discussed in [1] is formulated as ? ? ? ? ? ?1 ? i)) + ?t D ?t , min wi (t)?t (r(yi , ??t? x ?t }t?N T {? T d D?S++ ,tr(D)?1

t?NT t?NT

(1)

t?NT

where tr(D) is the trace of D, ?t : R ? R+ is the loss function for the tth task incurred on the ? i)1 , and ? ¿ 0 is the regularization parameter2 . It was shown [1] that the problem residual r(yi , ??t? x (1) is equivalent to ? ? ? ? i)) + ——B——2tr , min wi (t)?t (r(yi , ??t? x ?t }t?N T {? T t?NT i?NN

where B is the d ? T matrix whose tth column is given by the vector ?t , and ——B——tr := tr((BB ?)1/2) is the trace norm of B. As shown in [10], the trace norm is the convex upper envelope of the rank of B, and (1) can be interpreted as the problem of finding a common feature subspace across T tasks. This problem is often referred to as multi-task feature learning. If the matrix D is restricted to be diagonal, the formulation (1) is reduced to multi-task variable selection [11, 12]. In order to solve the problem (1), the alternating minimization algorithm was suggested in [1] (see Algorithm 1). This algorithm alternately optimizes the task-wise solutions {??t }t?NT and the common representation matrix D. It is worth noting that, when D is fixed, each ??t can be independently optimized (Step 1). On the other hand, when {??t }t?NT are fixed, the optimization of the matrix D can be reduced to the minimization over d eigenvalues ?1 , . . . , ?d of the matrix C := BB ? , and the optimal D can be analytically computed (Step 2).

3

Parametric-Task Learning (PTL)

We consider the case where we have infinitely many tasks parametrized by a single continuous parameter. Let ? ? [?L , ?U] be a continuous task parameter. Instead of the set of weights wi (t), t ? NT , we consider a weight function wi : [?L , ?U] ? [0, 1] for each instance i ? Nn . In PTL, we learn a parameter vector ??? ? Rd+1 as a continuous function of the task parameter ?: ? ?U ? ? ?U ? ? ? i)) d? + ? min wi (?) ?? (r(yi , ?? x ??? D?1 ?? d?, (2) ?? }??[? ,?] {? L U

d D?S++ ,tr(D)?1

?L

?L

i?Nn

where, note that, the loss function ?? possibly depends on ?. As we will explain in the next section, the above PTL formulation is useful in various important machine learning scenarios including learning under non-stationarity, cost-sensitive learning, and ? i) = (yi ? ??? x ? i)2 for regression problems with yi ? R, while r(yi , ??t? x ?i) = For example, r(yi , ??t? x ? ? ? i for binary classification problems with yi ? {?1, 1}. 1 ? y i ?t x 2 In [1], wi (t) takes either 1 or 0. It takes 1 only if the ith instance is used in the tth task. We slightly generalize the setup so that each instance can be used in multiple tasks with different weights. 1

2

Algorithm 1 A LTERNATING M INIMIZATION A LGORITHM FOR MTL [1] 1: Input: Data {(xi , yi)}i?Nn and weights {wi (t)}i?Nn ,t?NT ; 2: Initialize: D ? Id /d (Id is d ? d identity matrix) 3: while convergence condition is not true do 4: Step 1: For t = 1, . . . , T do ? ? ? i)) + ? ? D?1 ? wi (t)?t (r(yi , ??? x ??t ? arg min ? T ? i?Nn

5:

Step 2: D ?

? C 1/2 ?t? D?1 ?t , = arg min 1/2 d tr(C) D?S++ ,tr(D)?1 t?N T

where C := BB ? whose (j, k)th element is defined as Cj,k := 6: end while ?t }t?N and D; 7: Output: {? T

? t?NT

?tj ?tk .

quantile regression. However, at first glance, the PTL optimization problem (2) seems computationally intractable since we need to find infinitely many task-wise solutions as well as the common feature subspace (or the common set of variables if D is restricted to be diagonal) shared by infinitely many tasks. Our key finding is that, for a certain class of PTL problems, when D is fixed, the optimal path of the task-wise solutions ??? is shown to be piecewise-linear in ?. By exploiting this piecewise-linearity, we can efficiently handle infinitely many parameterized tasks, and the optimal solutions of those class of PTL problems can be exactly computed. In the following theorem, we prove that the task-wise solutions ??? is piecewise-linear in ? if the weight functions and the loss function satisfy certain conditions. d Theorem 1 For any d ? d positive-definite matrix D ? S++ , the optimal solution path of ? ? i)) + ?? ? D?1 ? ??? ? arg min wi (?)?? (r(yi , ??? x ? ?

(3)

i?Nn

? can be for ? ? [?L , ?U] is written as a piecewise-linear function of ? if the residual r(y, ??? x) ? and the weight functions wi : [?L , ?U] ? [0, 1], i ? Nn and the written as an affine function of ?, loss function ? : R ? R+ satisfy either of the following conditions (a) or (b): (a) All the weight functions are piecewise-linear functions, and the loss function is a convex piecewise-linear function which does not depend on ?; (b) All the weight functions are piecewise-constant functions, and the loss function is a convex piecewise-linear function which depends on ? in the following form: ? ?? (r) = max{(ah + bh r)(ch + dh ?), 0}, (4) h?NH

where H is a positive integer, and ah , bh , ch , dh ? R are constants such that ch + dh ? ? 0 for all ? ? [?L , ?U]. In the proof in Appendix A, we show that, if the weight functions and the loss function satisfy the conditions (a) or (b), the problem (3) is reformulated as a parametric quadratic program (parametric QP), where the parameter ? only appears in the linear term of the objective function. As shown, for example, in [9], the optimal solution path of this class of parametric QP has a piecewise-linear form. If ??? is piecewise-linear in ?, we can exactly compute the entire solution path by using parametric programming. In machine learning literature, parametric programming is often used in the context 3

Algorithm 2 A LTERNATING M INIMIZATION A LGORITHM FOR PTL 1: Input: Data {(xi , yi)}i?Nn and weight functions wi : [?L , ?U] :? [0, 1] for all i ? Nn ; 2: Initialize: D ? Id /d (Id is d ? d identity matrix) 3: while convergence condition is not true do 4: Step 1: For all the continuum of ? ? [?L , ?U] do ? ? i)) + ?? ? D?1 ? ??? ? arg min wi (?)?? (r(yi , ??? x ? ?

5:

i?Nn

by using parametric programming; Step 2: C 1/2 D ? = arg min d ,tr(D)?1 tr(C)1/2 D?S++

?

where (j, k)th element of C ? Rd?d is defined as Cj,k := 6: end while ?? } for ? ? [?L , ?U] and D; 7: Output: {?

?U

??? D?1 ?? d?,

(5)

?L

? ?U ?L

??,j ??,k d?;

of regularization path-following [13, 14, 15]3 . We start from the solution at ? = ?L , and follow the path of the optimal solutions while ? is continuously increased. This is efficiently conducted by exploiting the piecewise-linearity. Our proposed algorithm for solving the PTL problem (2) is described in Algorithm 2, which is essentially a continuous version of the MTL algorithm shown in Algorithm 1. Note that, by exploiting the piecewise linearity of ?? , we can compute the integral at Step 2 (Eq. (5)) in Algorithm 2. Algorithm 2 can be changed to parametric-task variable selection if Step 2 is replaced with ?? ?U 2 ??,j d? ?L ?? D ? diag(?1 , . . . , ?d) where ?j = ? for all j ? Nd , ?U 2 ? ? d? j ? ?Nd ?,j ?L which can also be computed efficiently by exploiting the piecewise linearity of ?? .

4 Examples of PTL Problems In this section, we present three examples where our PTL formulation (2) is useful. Binary Classification Under Non-Stationarity Suppose that we observe n training instances sequentially, and denote them as {(xi , yi , ?i)}i?Nn , where xi ? Rd , yi ? {?1, 1}, and ?i is the time when the ith instance is observed. Without loss of generality, we assume that ?1 ¡ . . . ¡ ?n . Under non-stationarity, if we are requested to learn a classifier to predict the output for a test input x observed at time ? , the training instances observed around time ? should have more influence on the classifier than others. Let wi (?) denote the weight of the ith instance when training a classifier for a test point at time ? . We can for example use the following triangular weight function (see Figure1): ? ? 1 + s?1 (?i ? ?) if ? ? s ? ?i ¡ ?, wi (?) = (6) 1 ? s?1 (?i ? ?) if ? ? ?i ¡ ? + s, ? 0 otherwise, where s ¿ 0 determines the width of the triangular time windows. The problem of training a classifier for time ? is then formulated as ? ? i) + ?——?——22 , min wi (?) max(0, 1 ? yi ??? x ? ?

i?Nn

where we used the hinge loss. 3 In regularization path-following, one computes the optimal solution path w.r.t. the regularization parameter, whereas we compute the optimal solution path w.r.t. the task parameter ?.

4

Figure 1: Examples of weight functions {wi (?)}i?Nn in non-stationary time-series learning. Given a training instances (xi , yi) at time ?i for i = 1, . . . , n under non-stationary condition, it is reasonable to use the weights {wi (?)}i?Nn as shown here when we learn a classifier to predict the output of a test input at time ? . If we have the belief that a set of classifiers for different time should have some common structure, we can apply our PTL approach to this problem. If we consider a time interval ? ? [?L , ?U], the parametric-task feature learning problem is formulated as ? ?U ? ? ?U ? i) d? + ? min wi (?) max(0, 1 ? yi ???? x ??? D?1 ?? d?. (7) ?)}? ?[? ,?] {?(? L U

?L

d D?S++ ,tr(D)?1

?L

i?Nn

Note that the problem (7) satisfies the condition (a) in Theorem 1. Joint Cost-Sensitive Learning Next, let us consider cost-sensitive binary classification. When the costs of false positives and false negatives are unequal, or when the numbers of positive and negative training instances are highly imbalanced, it is effective to use the cost-sensitive learning approach [16]. Suppose that we are given a set of training instances {(xi , yi)}i?Nn with xi ? Rd and yi ? {?1, 1}. If we know that the ratio of the false positive and false negative costs is approximately ? : (1 ? ?), it is reasonable to solve the following cost-sensitive SVM [17]: ? ? i) + ?——?——22 , min wi (?) max(0, 1 ? yi ??? x ? ?

i?Nn

where the weight wi (?) is defined as wi (?) =

{

? 1??

if yi = ?1, if yi = +1.

When the exact false positive and false negative costs in the test scenario are unknown [4], it is often desirable to train several cost-sensitive SVMs with different values of ?. If we have the belief that a set of classifiers for different cost ratios should have some common structure, we can apply our PTL approach to this problem. If we consider an interval ? ? [?L , ?U], 0 ¡ ?L ¡ ?U ¡ 1, the parametric-task feature learning problem is formulated as ? ?U ? ? ?U ? ? ? i) d? + ? min wi (?) max(0, 1 ? yi ?? x ??? D?1 ?? d?. (8) ?? }??[? ,?] {? L U

d D?S++ ,tr(D)?1

?L

?L

i?Nn

The problem (8) also satisfies the condition (a) in Theorem 1. Figure 2 shows an example of joint cost-sensitive learning applied to a toy 2D binary classification problem. Joint Quantile Regression Given a set of training instances {(xi , yi)}i?Nn with xi ? Rd and yi ? R drawn from a joint distribution P (X, Y), quantile regression [19] is used to estimate the conditional ? th quantile FY?1 —X=x (?) as a function of x, where ? ? (0, 1) and FY —X=x is the cumulative distribution function of the conditional distribution P (Y —X = x). Jointly estimating multiple conditional quantile functions is often useful for exploring the stochastic relationship between X and Y (see Section 5 for an example of joint quantile regression problems). Linear quantile regression along with L2 regularization [20] at order ? ? (0, 1) is formulated as { ? (1 ? ?)—r— if r ? 0, ? i) + ?——?——22 , ?? (r) := min ?? (yi ? ??? x ? —r— if r ¿ 0. ? ? i?Nn

5

4

2

2

0

0

x2

x2

4

-2

-2

-4

-4

-4

-2

0
2
4
6
-4
-2
x1
0
2
4
6
x1
(a) Independent cost-sensitive learning
(b) Joint cost-sensitive learning
Figure 2: An example of joint cost-sensitive learning on 2D toy dataset (2D input x is expanded to n-dimension by radial basis functions centered on each xi). In each plot, the decision boundaries of five cost-sensitive SVMs (? = 0.1, 0.25, 0.5, 0.75, 0.9) are shown. (a) Left plot is the results obtained by independently training each cost-sensitive SVMs. (b) Right plot is the results obtained by jointly training infinitely many cost-sensitive SVMs for all the continuum of ? ? [0.05, 0.95] using the methodology we present in this paper (both are trained with the same regularization parameter ?). When independently trained, the inter-relationship among different cost-sensitive SVMs looks inconsistent (c.f., [18]). If we have the belief that a family of quantile regressions at various ? ? (0, 1) have some common structure, we can apply our PTL framework to joint estimation of the family of quantile regressions This PTL problem satisfies the condition (b) in Theorem 1, and is written as ? 1 ? 1 ? ??? D?1 ?? d?, ?? (yi ? ??? xi)d? + ? min {?? }? ?(0,1) d D?S++ ,tr(D)?1
0 i?N n
0
where we do not need any weighting and omit wi (?) = 1 for all i ? Nn and ? ? [0, 1].

5 Numerical Illustrations

In this section, we illustrate various aspects of PTL with the three examples discussed in the previous section. Artificial Example for Learning under Non-stationarity We first consider a simple artificial problem with non-stationarity, where the data generating mechanism gradually changes. We assume that our data generating mechanism produces the training set {(xi , yi , ?i)}i?Nn with n = 100 as 2? 2? follows. For each ?i ? {0, 1 2? n , 2 n , . . . , (n ? 1) n }, the output yi is first determined as yi = 1 if i d is odd, while yi = ?1 if i is even. Then, xi ? R is generated as xi1 ? N (yi cos ?i , 12), xi2 ? N (yi sin ?i , 12), xij ? N (0, 12), ?j ? {3, . . . , d}, (9) where N (?, ? 2) is the normal distribution with mean ? and variance ? 2 . Namely, only the first two dimensions of x differ in two classes, and the remaining d ? 2 dimensions are considered as noise. In addition, according to the value of ?i , the means of the class-wise distributions in the first two dimensions gradually change. The data distributions of the first two dimensions for ? = 0, 0.5?, ?, 1.5? are illustrated in Figure 3. Here, we applied our PT feature learning approach with triangular time windows in (6) with s = 0.25?. Figure 4 shows the mis-classification rate of PT feature learning (PTFL) and ordinary independent learning (IND) on a similarly generated test sample with size 1000. When the input dimension d = 2, there is no advantage for learning common features since these two input dimensions are important for classification. On the other hand, as d increases, PT feature learning becomes more and more advantageous. Especially when the regularization parameter ? is large, the independent learning approach is completely deteriorated as d increases, while PTFL works reasonably well in all the setups. 6

Figure 3: The first 2 input dimensions of artificial example at ? = 0, 0.5?, ?, 1.5?. The class-wise distributions in these two dimensions gradually change with ? ? [0, 2?]. 0.5

PTL IND

0.3 0.2 0.1 0

0.4

0.5

PTL IND

Mis-classification Rate

0.4

Mis-classification Rate

Mis-classification Rate

0.5

0.3 0.2 0.1 0

2

5 10 20 50 Input Dimension

100

0.4

PTL IND

0.3 0.2 0.1 0

2

5 10 20 50 Input Dimension

100

2

5 10 20 50 Input Dimension

100

Figure 4: Experimental results on artificial example under non-stationarity. Mis-classification rate on test sample with size 1000 for various setups d ? {2, 5, 10, 20, 50, 100} and ? ? {0.1, 1, 10} are shown. The red symbols indicate the results of our PT feature learning (PTFL) whereas the blue symbols indicate ordinary independent learning (IND). The plotted are average (and standard deviation) over 100 replications with different random seeds. All the differences except d = 2 are statistically significant (p ¡ 0.01). Joint Cost-Sensitive SVM Learning on Benchmark Datasets Here, we report the experimental results on

joint cost-sensitive SVM learning discussed in Section 4. Although our main contribution is not just claiming favorable generalization properties of parametric task learning solutions, we compared, as an illustration, the generalization performances of PT feature learning (PTFL) and PT variable selection (PTVS) with the ordinary independent learning approach (IND). In PTFL and PTVS, we learned common feature subspaces and common sets of variables shared across the continuum of cost-sensitive SVM for ? ? [0.05, 0.95] for 10 benchmark datasets (see Table 1). In each data set, we divided the entire sample into training, validation, and test sets with almost equal size. The average test errors (and the standard deviation) of 10 different data splits are reported in ? Table 1. The total (?test errors for cost-sensitive SVMs ?with ? = 0.1, 0.2, . .). , 0.9 are defined as ??{0.1,...,0.9} ? i:yi =?1 I(f? (xi) ¿ 0) + (1 ? ?) i:yi =1 I(f? (xi) ? 0) , where f? is the trained SVM with the cost ratio ?. Model selection was conducted by using the same criterion on validation sets. We see that, in most cases, PTFL or PTVS had better generalization performance than IND. Joint Quantile Regression Finally, we applied PT feature learning to joint quantile regression problems. Here, we took a slightly different approach from what was described in the previous section. Given a training set {(xi , yi)}i?Nn , we first estimated conditional mean function E[Y —X = ? —X = xi], where E ? is the x] by least-square regression, and computed the residual ri := yi ? E[Y estimated conditional mean function. Then, we applied PT feature learning to {(xi , ri)}i?Nn , and ? ? estimated the conditional ? th quantile function as F?Y?1 —X=x (?) := E[Y —X = xi] + fres (x—?), where f?res (?—?) is the estimated ? th quantile regression fitted to the residuals. When multiple quantile regressions with different ? s are independently learned, we often encounter a notorious problem known as quantile crossing (see Section 2.5 in [5]). For example, in Figure 5(a), some of the estimated conditional quantile functions cross each other (which never happens in the true conditional quantile functions). One possible approach to mitigate this problem is to assume a model on the heteroscedastic structure. In the simplest case, if we assume that the data is homoscedastic (i.e., the conditional distribution P (Y —x) does not depend on x except its location), 7

Table 1: Average (and standard deviation) of test errors obtained by joint cost-sensitive SVMs on benchmark datasets. n is the sample size, d is the input dimension, Ind indicates the results when each cost-sensitive SVM was trained independently, while PTFL and PTVS indicate the results from PT feature learning and PT feature selection, respectively. The bold numbers in the table indicate the best performance among three methods. n 195 569 194 690 768 862 1000 1000 300 528

Data Name Parkinson Breast Cancer Diagnostic Breast Cancer Prognostic Australian Diabetes Fourclass Germen Splice SVM Guide DVowel

d 20 30 33 14 8 2 24 60 10 10

Ind 32.30 (10.60) 20.36 (7.77) 48.97 (12.92) 117.97 (22.97) 185.90 (21.13) 181.69 (22.13) 242.21 (18.35) 179.80 (24.22) 175.70 (15.55) 175.16 (13.78)

PTFL 30.21 (9.09) 18.49 (6.15) 49.28 (9.83) 106.25 (12.66) 179.89 (16.31) 179.30 (14.25) 219.66 (16.22) 151.69 (18.02) 170.16 (9.99) 175.74 (9.37)

PTVS 30.25 (8.53) 19.46 (5.89) 48.68 (5.89) 111.22 (15.95) 175.95 (16.26) 178.67 (19.24) 237.20 (15.78) 183.54 (21.27) 179.76 (14.76) 175.50 (7.38)

quantile regressions at different ? s can be obtained by just vertically shifting other quantile regression function (see Figure 5(f)). Our PT feature learning approach, when applied to the joint quantile regression problem, allows us to interpolate these two extreme cases. Figure 5 shows a joint QR example on the bone mineral density (BMD) data [21]. We applied our approach after expanding univariate input x to a d = 5 dimensional vector by using evenly allocated RBFs. When (a) ? ? 0, our approach is identical with independently estimating each quantile regression, while it coincides with homoscedastic case when (f) ? ? ?. In our experience, the best solution is usually found somewhere between these two extremes: in this example, (d) ? = 5 was chosen as the best model by 10-fold cross-validation. 4

2 1 0 -1 -2

2 1 0 -1

-1.5

-1

-0.5

0 0.5 (Standardized) Age

1

1.5

-2

2

-1.5

-1

-0.5

0

0.5

1

1.5

2 1 0 -1

-0.5

0 0.5 (Standardized) Age

(d) ? = 5

-2

-1.5

-1

-0.5

1

1.5

2

0

0.5

1

1.5

2

(Standardized) Age
(c) ? = 1 4
0.05, 0.10, ..., 0.95 conditional quantile functions
3 2 1 0 -1 -2
-2 -1
0 -1
2
(Standardized) Relative BMD Change
(Standardized) Relative BMD Change
3
-1.5
1
(b) ? = 0.1 4
0.05, 0.10, ..., 0.95 conditional quantile functions
-2
2
(Standardized) Age
(a) ? ? 0 4
0.05, 0.10, ..., 0.95 conditional quantile functions
3
-2
-2 -2
(Standardized) Relative BMD Change
4
0.05, 0.10, ..., 0.95 conditional quantile functions
3
(Standardized) Relative BMD Change
0.05, 0.10, ..., 0.95 conditional quantile functions
3
(Standardized) Relative BMD Change
(Standardized) Relative BMD Change
4
0.05, 0.10, ..., 0.95 conditional quantile functions
3 2 1 0 -1 -2
-2
-1.5
-1
-0.5
0 0.5 (Standardized) Age
1
(e) ? = 10
1.5
2
-2
-1.5
-1

-0.5
0 0.5 (Standardized) Age
1
1.5
2
(f) ? ? ?
Figure 5: Joint quantile regression examples on BMD data [21] for six different ?s.
6
Conclusions
In this paper, we introduced parametric-task learning (PTL) approach that can systematically handle infinitely many tasks parameterized by a continuous parameter. We illustrated the usefulness of this approach by providing three examples that can be naturally formulated as PTL. We believe that there are many other practical problems that falls into this PTL framework. Acknowledgments The authors thank the reviewers for fruitful comments. IT, MS, and SN thank the support from MEXT Kakenhi 23700165, JST CREST Program, MEXT Kakenhi 23120004, respectively. 8

2 References

[1] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In Advances in Neural Information Processing Systems, volume 19, pages 41?48. 2007. [2] A. Argyriou, C. A. Micchelli, M. Pontil, and Y. Ying. A spectral regularization framework for multi-task structure learning. In Advances in Neural Information Processing Systems, volume 20, pages 25?32. 2008. [3] L. Cao and F. Tay. Support vector machine with adaptive parameters in finantial time series forecasting. IEEE Transactions on Neural Networks, 14(6):1506?1518, 2003. [4] F. R. Bach, D. Heckerman, and E. Horvits. Considering cost asymmetry in learning classifiers. Journal of Machine Learning Research, 7:1713?41, 2006. [5] R. Koenker. Quantile Regression. Cambridge University Press, 2005. [6] K. Ritter. On parametric linear and quadratic programming problems. mathematical Programming: Proceedings of the International Congress on Mathematical Programming, pages 307?335, 1984. [7] E. L. Allgower and K. George. Continuation and path following. Acta Numerica, 2:1?63, 1993. [8] T. Gal. Postoptimal Analysis, Parametric Programming, and Related Topics. Walter de Gruyter, 1995. [9] M. J. Best. An algorithm for the solution of the parametric quadratic programming problem. Applied Mathemetics and Parallel Computing, pages 57?76, 1996. [10] M. Fazel, H. Hindi, and S. P. Boyd. A rank minimization heuristic with application to minimum order system approximation. In Proceedings of the American Control Conference, volume 6, pages 4734?4739, 2001. [11] B. A. Turlach, W. N. Venables, and S. J. Wright. Simultaneous variable selection. Technometrics, 47:349?363, 2005. [12] G. Obozinski, B. Taskar, and M. Jordan. Joint covariate selection and joint sbspace selection for multiple classification problems. Statistics and Computing, 20(2):231?252, 2010. [13] M. R.

Osborne, B. Presnell, and B. A. Turlach. A new approach to variable selection in least squares problems. IMA Journal of Numerical Analysis, 20(20):389?404, 2000. [14] B. Efron and R. Tibshirani. Least angle regression. Annals of Statistics, 32(2):407?499, 2004. [15] T. Hastie, S. Rosset, R. Tibshirani, and J. Zhu. The entire regularization path for the support vector machine. Journal of Machine Learning Research, 5:1391?415, 2004. [16] Y. Lin, Y. Lee, and G. Wahba. Support vector machines for classification in nonstandard situations. Machine Learning, 46:191?202, 2002. [17] M. A. Davenport, R. G. Baraniuk, and C. D. Scott. Tuning support vector machine for minimax and Neyman-Pearson classification. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2010. [18] G. Lee and C. Scott. Nested support vector machines. IEEE Transactions on Signal Processing, 58(3):1648?1660, 2010. [19] R. Koenker. Quantile Regression. Cambridge University Press, 2005. [20] I. Takeuchi, Q. V. Le, T. Sears, and A. J. Smola. Nonparametric quantile estimation. Journal of Machine Learning Research, 7:1231?1264, 2006. [21] L. K. Bachrach, T. Hastie, M. C. Wang, B. Narasimhan, and R. Marcus. Acquisition in healthy Asian, hispanic, black and caucasian youth. a longitudinal study. The Journal of Clinical Endocrinology and Metabolism, 84:4702?4712, 1999. [22] S. Boyd and L. Vandenberghe. Convex Optimization. Cambridge University Press, 2004.

9