Heterogeneous-Neighborhood-based Multi-Task Local Learning Algorithms

Authored by:

Yu Zhang

Abstract

All the existing multi-task local learning methods are defined on homogeneous neighborhood which consists of all data points from only one task. In this paper, different from existing methods, we propose local learning methods for multi-task classification and regression problems based on heterogeneous neighborhood which is defined on data points from all tasks. Specifically, we extend the k-nearest-neighbor classifier by formulating the decision function for each data point as a weighted voting among the neighbors from all tasks where the weights are task-specific. By defining a regularizer to enforce the task-specific weight matrix to approach a symmetric one, a regularized objective function is proposed and an efficient coordinate descent method is developed to solve it. For regression problems, we extend the kernel regression to multi-task setting in a similar way to the classification case. Experiments on some toy data and real-world datasets demonstrate the effectiveness of our proposed methods.

1 Paper Body

For single-task learning, besides global learning methods there are local learning methods [7], e.g., k-nearest-neighbor (KNN) classifier and kernel regression. Different from the global learning methods, the local learning methods make use of locality structure in different regions of the feature space and are complementary to the global learning algorithms. In many applications, the single-task local learning methods have shown comparable performance with the global counterparts. Moreover, besides classification and regression problems, the local learning methods are also applied to some other learning problems, e.g., clustering [18] and dimensionality reduction [19]. When the number of labeled data is not very large, the performance of the local learning methods is limited due to sparse local density [14]. In this case, we can leverage the useful information from other related tasks to help improve the performance which matches the philosophy of multi-task learning [8, 4, 16]. Multi-task learning utilizes supervised information from some related tasks to improve the performance of one task at hand and during the past decades many advanced methods have been proposed for multi-task learning, e.g., [17, 3, 9, 1, 2, 6, 12, 20, 14, 13]. Among those methods, [17, 14] are two representative multi-task local learning methods. Even though both methods in [17, 14] use KNN as the base learner for each task, Thrun and O?Sullivan [17] focus on learning cluster structure among different tasks while Parameswaran and Weinberger [14] learn different distance metrics for different tasks. The KNN classifiers use in both two methods are defined on the homogeneous neighborhood which is the set of nearest data points from the same task the query point belongs to. In some situation, it is better to use a heterogeneous neighborhood which is defined as the set of nearest data points from all tasks. For example, suppose we have two similar tasks marked with two colors as shown in Figure 1. For a test data point marked with ??? from one task, we obtain an estimation with low confidence or even a wrong one based on the homogeneous neighborhood. However, if we can use the data points from both two tasks to define the neighborhood (i.e., heterogeneous neighborhood), we can obtain a more confident estimation. 1

In this paper, we propose novel local learning models for multi-task learning based on the heterogeneous neighborhood. For multi-task classification problems, we extend the KNN classifier by formulating the decision function on each data point as weighted voting of its neighbors from all tasks where the weights are task-specific. Since multi-task learning usually considers that the contribution of one task to another one equals that in the reverse direc- Figure 1: Data points with one color tion, we define a regularizer to enforce the task-specific (i.e., black or red) are from the same weight matrix to approach a symmetric matrix and then task and those with one type of marker based on this regularizer, a regularized objective function (i.e., ?+? or ?-?) are from the same class. is proposed. We develop an efficient coordinate descent A test data point is represented by ???. method to solve it. Moreover, we also propose a local method for multi-task regression problems. Specifically, we extend the kernel regression method to multi-task setting in a similar way to the classification case. Experiments on some toy data and real-world datasets demonstrate the effectiveness of our proposed methods.

2

A Multi-Task Local Classifier based on Heterogeneous Neighborhood

In this section, we propose a local classifier for multi-task learning by generalizing the KNN algorithm, which is one of the most widely used local classifiers for single-task learning. Suppose we are given m learning tasks {Ti }m i=1 . The training set consists of n triples (xi , yi , ti) with the ith data point as xi ? RD , its label yi ? {?1, 1} and its task indicator ti ? {1, . . . , m}. So each task is a binary classification problem with ni = —{j—tj = i}— data points belonging to the ith task Ti . For the ith data point xi , we use Nk (i) to denote the set of the indices of its k nearest neighbors. If Nk (i) is a homogeneous neighborhood which only P contains data points from the task that xi belongs to, we can use d(xi) = sgn j?Nk (i) s(i, j)yj to make a decision for xi where sgn(?) denotes the sign function and s(i, j) denotes a similarity function between xi and xj . Here, by defining Nk (i) as a heterogeneous neighborhood which contains data points from all tasks, we cannot directly utilize this decision function and instead we introduce a weighted decision function by using task-specific weights as ? ? d(xi) = sgn ?

X

wti ,tj s(i, j)yj ?

j?Nk (i)

where wqr represents the contribution of the rth task Tr to the qth one Tq when Tr has some data points to be neighbors of a data point from Tq . Of course, the contribution from one task to itself should be positive and also the largest, i.e., wii ? 0 and ?wii ? wij ? wii for j 6= i. When wqr (q 6= r) approaches wqq , it means Tr is very similar to Tq in local regions. At another extreme where wqr (q 6= r) approaches ?wqq , if we flip the labels of data points in Tr , Tr can have a positive contribution ?wqr to Tq which indicates that Tr is negatively correlated to Tq . Moreover, when wqr /wqq (q 6= r) is close to 0 which implies there is no contribution from Tr to Tq , Tr is likely to be unrelated to Tq . So the utilization of {wqr } can model three task relationships: positive task correlation, negative task correlation and task unrelatedness as in [6, 20]. P We use f (xi) to define the estimation function as f (xi) = j?Nk (i) wti ,tj s(i, j)yj . Then similar to support vector machine (SVM), we use hinge loss l(y, y 0) = max(0, 1 ? yy 0) to measure empirical performance on the training data. Moreover, recall that wqr represents the contribution of Tr to Tq and wrq is the contribution of Tq to Tr . Since multi-task learning usually considers that the contribution of Tr to Tq almost equals that of Tq to Tr , we expect wqr to be close to wrq . To encode this priori information into our model, we can either formulate it as wqr = wrq , a hard constraint, or a soft regularizer, i.e., minimizing (wqr ? wrq)2 to enforce wqr ? wrq , which is more preferred. Combining all the above considerations, we can construct a objective function for our proposed method MT-KNN as min W

n X i=1

l(yi , f (xi)) +

?1 ?2 kW ? WT k2F + kWk2F 4 2

2

s.t. wqq ? 0, wqq ? wqr ? ?wqq (q 6= r)

(1)

where W is a m ? m matrix with wqr as its (q, r)th element and k ? kF denotes Frobenius norm of a matrix. The first term in the objective function of problem (1) measures the training loss, the second one enforces W to be a symmetric matrix which implies wqr ? wrq , and the last one penalizes the complexity of W. The regularization parameters ?1 and ?2 balance the trade-off between these three terms. 2.1

Optimization Procedure

In this section, P we discuss how to solve problem (1). We first rewrite f (xi) as f (xi) = Pm ?i where Nkj (i) denotes the set of the indices of xi ?s nearest j=1 wti j l?N j (i) s(i, l)yl = wti x k

neighbors from the jth task in Nk (i),Pwti = (wti 1 , . . . , wti m) is the ti th row of W, and x ?i is a m ? 1 vector with the jth element as l?N j (i) s(i, l)yl . Then we can reformulate problem (1) as k

min W
m X X
l(yj , wi x ?j) +
i=1 j?Ti
?1 ?2 kW ? WT k2F + kWk2F 4 2
s.t. wqq ? 0, wqq ? wqr ? ?wqq (q 6= r). (2)

To solve problem (2), we use a coordinate descent method, which is also named as an alternating optimization method in some literatures. By adopting the hinge loss in problem (2), the optimization problem for wik (k 6= i) is formulated as min wik

X ? 2 wik ? ?ik wik + max(0, ajik wik + bjik) 2 j?T
s.t. cik ? wik ? eik
(3)
i

where ?jk is the kth element of x ?j , ajik = ?yj x ?jk , bjik = 1 ? P ? = ?1 + ?2 , ?ik = ?1 wki , x yj t6=k wit x ?jt , cik = ?wii , and eik = wii . If the objective function of problem (3) only has the first two terms, this problem will become a univariate quadratic programming (QP) problem with a linear inequality constraint, leading to an analytical solution. Moreover, similar to SVM we can introduce some slack variables for the third term in the objective function of problem (3) and then that problem will become a QP problem with ni + 1 variables and 2ni + 1 linear constraints. We can use off-the-shelf softwares to solve this problem in polynomial time. However, the whole optimization procedure may not be very efficient since we need to solve problem (3) and call QP solvers for multiple times. Here we utilize the piecewise linear structure of the last term in the objective function of problem (3) and propose a more efficient solution. We assume all aj are non-zero and otherwise we can discard them without affecting the solution since the corresponding losses are constants. We define six index sets as C1 = {j—ajik ¿ 0, ?

bj bj bjik ¡ cik }, C2 = {j—ajik ¿ 0, cik ? ? ik ? eik }, C3 = {j—ajik ¿ 0, ? ik ¿ eik } j j aik aik ajik

C4 = {j—ajik ¡ 0, ?

bjik bjik bjik j j ¡ c }, C = {j—a ¡ 0, c ? ? ? e }, C = {j—a ¡ 0, ? ¿ eik }. 5 6 ik ik ik ik ik ajik ajik ajik

It is easy to show that when j ? C1 ?C6 where the operator ? denotes the union of sets, ajik w+bjik ¿ 0 holds for w ? [cik , eik], corresponding to the set of data points with non-zero loss. Oppositely when j ? C3 ? C4 , the values of the corresponding losses become zero since ajik w + bjik ? 0 holds for w ? [cik , eik]. The variation lies in the data points with indices j ? C2 ? C5 . We sort sequence {?bjik /ajik —j ? C2 } and record it in a vector u of length du with u1 ? . . . ? udu . Moreover, we also keep a index mapping qu with its rth element qru defined as qru = j if ur = ?bjik /ajik . Similarly, for sequence {?bjik /ajik —j ? C5 }, we define a sorted vector v of length dv and the corresponding index mapping qv . By using the merge-sort algorithm, we merge u and v into a sorted vector s and then we add cik and eik into s as the minimum and maximum elements if they are not contained in s. Obviously, in range [sl , sl+1] where sl is the lth element of s and ds is the length of s, problem (3) becomes a univariate QP problem which has an analytical solution. So we can compute local minimums in successive regions [sl , sl+1] (l = 1, . . . , ds ? 1) and get the global minimum over region [cik , eik] by comparing all local optima. The key operation is to compute the coefficients of quadratic function over each region [sl , sl+1] and we devise an algorithm in Table 1 which only needs to scan s once, leading to an efficient solution for problem (3). 3

The first step of the algorithm in Table 1 needs O(ni) time complexity to construct the six sets C1 to C6 . In step 2, we need to sort two sequences to obtain u and v in O(du ln du + dv ln dv) time and merge two sequences to get s in O(du + dv). Then it costs O(ni) to calculate coefficients c0 and c1 by scanning C1 , C2 and C6 in step 4 and 5. Then from step 6 to step 13, we need to scan vector s once which costs O(du + dv) time. The overall complexity of the algorithm in Table 1 is O(du ln du + dv ln dv + ni) which is at most O(ni ln ni) due to du + dv ? ni .

wii

X ?2 2 max(0, aji wii + bji) wii + 2 j?T

Construct four sets C1 , C2 , C3 , C4 , C5 and C6 ; Construct u, qu , v, qv and s; Insert cik and eik into s if needed; P c0 := bj ; Pj?C1 ?C2 ?C6 ik j c1 := j?C1 ?C2 ?C6 aik ? ?ik ; w := sds ; o := c0 + c1 w + ?w2 /2; for l = ds ? 1 to 1 if sl+1 = ur for some r

08:

c0 := c0 ? bikr ; c1 := c1 ? aikr ; end if if sl+1 = vr for some r

s.t. wii ? ci ,

(4)

i

qu

qu

qv

qv

c0 := c0 + bikr ; c1 := c1 + aikr ; end if c 10: w0 := min(sl+1 , max(sl , ? ?1)); 2 11: o0 := c0 + c1 w0 + ?w0 /2; if o0 ¡ o 12: w := w0 ; o := o0 ; end if 13: l := l ? 1; end for 09:

For wii , the optimization problem is formulated as min

Table 1: Algorithm for problem (3) 01: 02: 03: 04: 05: 06: 07:

P ?jt , ci = ?ji , bji = 1 ? yj t6=i wit x where aji = ?yj x max(0, maxj6=i (—wij —)), and — ? — denotes the absolute value of a scalar. The main difference between problem (3) and (4) is that there exist a box constraint for wik in problem (3) but in problem (4) wii is only bj

lower-bounded. We define ei as ei = maxj {? aij } for all aji 6= 0. For wii ? [ei , +?), the objective i P 2 function of problem (4) can be reformulated as ?22 wii + j?S (aji wii + bji) where S = {j—aji ¿ 0} (1)

and the minimum value in [ei , +?) will take at wii = max{ei , ?

P

j?S

?2

aji

}. Then we can use the

(2) wii

algorithm in Table 1 to find the minimizor in the interval [ci , ei] for problem (4). Finally we (1) (2) can choose the optimal solution to problem (4) from {wii , wii } by comparing the corresponding values of the objective function. Since the complexity to solve both Pmproblem (3) and (4) is O(ni ln ni), the complexity of one update for the whole matrix W is O(m i=1 ni ln ni). Usually the coordinate descent algorithm converges very fast in a small number of iterations and hence the whole algorithm to solve problem (2) or (1) is very efficient. We can use other loss functions for problem (2) instead of hinge loss, e.g., square loss l(s, t) = (s ? t)2 as in the least square show that problem (3) has an analytical to SVM [10]. It is easy P j j ?ik ?2 j?T aik bik i P j 2 ?+2 j?T (aik) i

P j j ?2 j?T ai bi i max ci , P j ?2 +2 j?T (ai)2

solution as wik = min max cik , computed as wii =

, eik

and the solution to problem (4) can be

. Then the computational complexity of the whole

i

algorithm to solve problem (2) by adopting square loss is O(mn).

3

A Multi-Task Local Regressor based on Heterogeneous Neighborhood

In this section, we consider the situation that each task is a regression problem with each label yi ? R. Similar to the classification case in the previous section, one candidate for multi-task local regressor is a generalization of kernel regression, a counterpart of KNN classifier for regression problems, and the estimation function can be formulated as P

j?N (i) f (xi) = P k

wti ,tj s(i, j)yj

j?Nk (i)

wti ,tj s(i, j)

(5)

where wqr also represents the contribution of Tr to Tq . Since the denominator of f (xi) is a linear combination of elements in each row of W with data-dependent combination coefficients, if we utilize a similar formulation to problem (1) with square loss, we need to solve a complex and nonconvex fractional programming problem. For computational consideration, we resort to another way to construct the multi-task local regressor. 4

Recall that the estimation function for the classification case is formulated as f (xi) = P Pm s(i, l)yl . We can see that the expression in the brackets on the right-hand j j=1 wti j l?Nk (i) side can be viewed as a prediction for xi based on its neighbors in the jth task. Inspired by this observation, we can construct a prediction y?ji for xi based on its neighbors from the jth task by utilizing any regressor, e.g., kernel regression and support vector regression. Here due to the local nature of our proposed method, we choose the kernel regression method, which is a local regression P method, as a good candidate and hence y?ji is formulated as y?ji =

s(i,l)yl j l?N (i) k s(i,l) j l?N (i) k

P

.

When j equals

ti which means we use neighbored data points from the task that xi belongs to, we can use this prediction in confidence. However, if j 6= ti , we cannot totally trust the prediction and need to add some weight wti ,j as a confidence. Then by using the square loss, we formulate an optimization problem to get the estimation function f (xi) based on {? yji } as f (xi) = arg min y

m X

Pm

j=1 wti ,j (y ? y?ji)2 = Pm

wti ,j y?ji

j=1

j=1

wti ,j

.

(6)

Compared with the regression function of the direct extension of kernel regression to multi-task learning in Eq. (5), the denominator of our proposed regressor in Eq. (6) only includes the row summation of W, making the optimization problem easier to solve as we will see later. Since the scale of wij does not matter theP value of the estimation function in Eq. (6), we constrain the row m summation of W to be 1, i.e., j=1 wij = 1 for i = 1, . . . , m. Moreover, the estimation y?tii by using data from the same task as xi is more trustful than the estimations based on other tasks, which suggestsP wii should be the largest among elements in the ith row. Then this constraint implies 1 1 that wii ? m k wik = m ¿ 0. To capture the negative task correlations, wij (i 6= j) is only required to be a real scalar and wij ? ?wii . Combining the above consideration, we formulate an optimization problem as min W

m X X

(wi y ?j ? yj)2 +

i=1 j?Ti

?2 ?1 kW ? WT k2F + kWk2F s.t. W1 = 1, wii ? wij ? ?wii , 4 2

(7)

j T) . In the following where 1 denotes a vector of all ones with appropriate size and y ?j = (? y1j , . . . , y?m section, we discuss how to optimize problem (7).

3.1

Optimization Procedure

Due to the linear equality constraints in problem (7), we cannot apply a coordinate descent method to update variables one by one in a similar way to problem (2). However, similar to the SMO algorithm [15] for SVM, we can update two variables in one row of W at one time to keep the linear equality constraints valid. We update each row one by one and the optimization problem with respect to wi is formulated as min wi

1 wi AwiT + wi bT 2

s.t.

m X

wij = 1, ?wii ? wij ? wii ?j 6= i,

(8)

j=1

P where A = 2 j?Ti y ?j y ?jT + ?1 Iim + ?2 Im , Im is an m ? m identity matrix, Iim is a copy of Im P by setting the (i, i)th element to be 0, b = ?2 j?Ti yj y ?jT ? ?1 cTi , and ci is the ith column of W by setting its ith element to 0. We define the Lagrangian as m X X X 1 J = wi AwiT + wi bT ? ?(wij ? 1) ? (wii ? wij)?j ? (wii + wij)?j . 2 j=1 j6=i

j6=i

The Karush-Kuhn-Tucker (KKT) optimality condition is formulated as ?J = wi aj + bj ? ? + ?j ? ?j = 0, for j 6= i ?wij X ?J = w i a i + bi ? ? ? (?k + ?k) = 0 ?wii

(9) (10)

k6=i

?j ? 0, (wii ? wij)?j = 0 ?j 6= i ?j ? 0, (wii + wij)?j = 0 ?j 6= i,

5

(11) (12)

where aj is the jth column of A and bj is the jth element of b. It is easy to show that ?j ?j = 0 for all j 6= i. When wij satisfies wij = wii , according to Eq. (12) we have ?j = 0 and further wi aj + bj = ? ? ?j ? ? according to Eq. (9). When wij = ?wii , based on Eq. (11) we can get ?j = 0 and then wi aj + bj = ? + ?j ? ?. For wij between those two extremes (i.e., ?wii ¡ wij ¡ wii), ?j = ?j = 0 according to Eqs. (11) P and (12), which implies that wi aj + bj = ?. Moreover, Eq. (10) implies that wi ai + bi = ? + k6=i (?k + ?k) ? ?. We define sets as S1 = {j—wij = wii , j 6= i}, S2 = {j— ? wii ¡ wij ¡ wii }, S3 = {j—wij = ?wii }, and S4 = {i}. Then a feasible wi is a stationary point of problem (8) if and only if maxj?S1 ?S2 {wi aj + bj } ? mink?S2 ?S3 ?S4 {wi ak + bk }. If there exist a pair of indices (j, k), where j ? S1 ? S2 and k ? S2 ? S3 ? S4 , satisfying wi aj + bj ¿ wi ak + bk , {j, k} is called a violating pair. If the current estimation wi is not an optimal solution, there should exist some violating pairs. Our SMO algorithm updates a violating pair at one step by choosing the most violating pair {j, k} with j and k defined as j = arg maxl?S1 ?S2 {wi al + bl } and k = arg minl?S2 ?S3 ?S4 {wi al + bl }. We define the update rule for wij and wik as w ?ij = wij + t and w ?ik = wik ? t. By noting that j cannot be i, t should satisfy the following constraints to make the updated solution feasible: when k = i, t ? wik ? wij + t ? wik ? t, t ? wik ? wil ? wik ? t ?l 6= j&l 6= k when k 6= i, ?wii ? wij + t ? wii , ?wii ? wik ? t ? wii .

w ?w When k = i, there will be a constraint on t as t ? e ? min ik 2 ij , minl6=j&l6=k (wik ? —wil —) and otherwise t will satisfy c ? t ? e where c

= max(wik ? wii , ?wij ? wii) and e = min(wii ? wij , wii + wik). Then the optimization problem for t can be unified as min t

ajj + aii ? 2aji 2 t + (wi aj + bj ? wi ai ? bi)t 2

s.t. c ? t ? e,

where for the case that k = i, c is set to be ??. This problem has an analytical solution as w ai +bi ?wi aj ?bj t = min e, max c, iajj . We update each row of W one by one until convergence. +aii ?2aji

4

Experiments

In this section, we test the empirical performance of our proposed methods in some toy data and real-world problems. 4.1

Toy Problems

We first use one UCI dataset, i.e., diabetes data, to analyze the learned W matrix. The diabetes data consist of 768 data points from two classes. We randomly select p percent of data points to form the training set of two learning tasks respectively. The regularization parameters ?1 and ?2 are fixed as 1 and the number of nearest neighbors is set to 5. When p = 20 and p = 40, the means of the 0.1025 0.1011 0.1014 0.1004 estimated W over 10 trials are and . This result shows 0.0980 0.1056 0.1010 0.1010 wij (j 6= i) is very close to wii for i = 1, 2. This observation implies our method can find that these two tasks are positive correlated which matches our expectation since those two tasks are from the same distribution. For the second experiment, we randomly select p percent of data points to form the training set of two learning tasks respectively but differently we flip the labels of one task so that those two tasks should be negatively The matrices W?s learned for p = 20 and p = 40 are

correlated.

0.1019 ?0.1017 0.1019 ?0.0999 and . We can see that wij (j 6= i) is very close ?0.1007 0.1012 ?0.0997 0.1038 to ?wii for i = 1, 2, which is what we expect. As the third problem, we construct two learning tasks as in the first one but flip 50% percent of the class labels in each class of those two tasks. Here those two tasks can be viewed as unrelated tasks since the label assignment matrices W?s for p = 20 and p = 40 are

is random. The estimated

0.1575 0.0398

0.0144 0.1281

and

0.1015 0.0081

?0.0003 0.1077

, where wij (i 6= j) is much smaller than wii . From

the structure of the estimations, we can see that those two tasks are more likely to be unrelated, matching our expectation. In summary, our method can learn the positive correlations, negative correlations and task unrelatedness for those toy problems. 6

4.2

Experiments on Classification Problems

Two multi-task classification problems are used in our experiments. The first problem we investigate is Table 2: Comparison of classification errors of different a handwritten letter classification ap- methods on the two classification problems in the form of plication consisting of seven tasks mean?std. Letter USPS each of which is to distinguish tKNN 0.0775?0.0053 0.0445?0.0131 wo letters. The corresponding lettermtLMNN 0.0511?0.0053 0.0141?0.0038 s for each task to classify are: c/e, MTFL 0.0505?0.0038 0.0140?0.0025 g/y, m/n, a/g, a/o, f/t and h/n. Each MT-KNN(hinge) 0.0466?0.0023 0.0114?0.0013 class in each task has about 1000 data MT-KNN(square) 0.0494?0.0028 0.0124?0.0014 points which have 128 features corresponding to the pixel values of handwritten letter images. The second one is the USPS digit classification problem and it consists of nine binary classification tasks each of which is to classify two digits. Each task contains about 1000 data points with 255 features for each class.

Running Time (in second)

Here the similarity function we use is a heat kx ?x k2 0.8 kernel s(i, j) = exp{? i2?2j 2 } where ? Our Method CVX Solver 0.7 is set to the mean pairwise Euclidean dis0.6 tance among training data. We use 5-fold cross validation to determine the optimal ?1 0.5 and ?2 whose candidate values are chosen 0.4 from n ? {0.01, 0.1, 0.5, 1, 5, 10, 100} and the 0.3 optimal number of nearest neighbors from 0.2 {5, 10, 15, 20}. The classification error is used 0.1 as the performance measure. We compare our 0 Letter USPS Robot method, which is denoted as MT-KNN, with Dataset the KNN classifier which is a single-task learning method, the multi-task large margin nearest neighbor (mtLMNN) method [14]1 which is a Figure 2: Comparison on average running time multi-task local learning method based on the over 100 trials between our proposed coordinate homogeneous neighborhood, and the multi-task descent methods and the CVX solver on classifeature learning (MTFL) method [2] which is a cation and regression problems. global method for multi-task learning. By utilizing hinge and square losses, we also consider two variants of our MT-KNN method. To mimic the real-world situation where the training data are usually limited, we randomly select 20% of the whole data as training data and the rest to form the test set. The random selection is repeated for 10 times and we record the results in Table 2. From the results, we can see that our method MT-KNN is better than KNN, mtLMNN and MTFL methods, which demonstrates that the introduction of the heterogeneous neighborhood is helpful to improve the performance. For different loss functions utilized by our method, MT-KNN with hinge loss is better than that with square loss due to the robustness of the hinge loss against the square loss. For those two problems, we also compare our proposed coordinate descent method described in Table 1 with some off-the-shelf solvers such as the CVX solver [11] with respect to the running time. The platform to run the experiments is a desktop with Intel i7 CPU 2.7GHz and 8GB RAM and we use Matlab 2009b for implementation and experiments. We record the average running time over 100 trials in Figure 2 and from the results we can see that on the classification problems above, our proposed coordinate descent method is much faster than the CVX solver which demonstrates the efficiency of our proposed method. 4.3

Experiments on Regression Problems

Here we study a multi-task regression problem to learn the inverse dynamics of a seven degree-offreedom SARCOS anthropomorphic robot arm.2 The objective is to predict seven joint torques based 1 2

http://www.cse.wustl.edu/?kilian/code/files/mtLMNN.zip http://www.gaussianprocess.org/gpml/data

7

on 21 input features, corresponding to seven joint positions, seven joint velocities and seven joint accelerations. So each task corresponds to the prediction of one torque and can be formulated as a regression problem. Each task has 2000 data points. The similarity function used here is also the heat kernel and 5-fold cross validation is used to determine the hyperparameters, i.e., ?1 , ?2 and k. The performance measure used is normalized mean squared error (nMSE), which is mean squared error on the test data divided by the variance of the ground truth. We compare our method denoted by MTKR with single-task kernel regression (KR), the multi-task feature learning (MTFL) under different configurations on the size of the training set. Compared with KR and MTFL methods, our method achieves better performance over different sizes of the training sets. Moreover, for our proposed coordinate descent method introduced in section 3.1, we compare it with CVX solver and record the results in the last two columns of Figure 2. We find the running time of our proposed method is much smaller than that of the CVX solver which demonstrates that the proposed coordinate descent method can speed up the computation of our MT-KR method. 0.08 KR MTFL MT?KR

nMSE

0.06

0.04

0.02

0

0.1

0.2 The size of training set

0.3

Figure 3: Comparison of different methods on the robot arm application when varying the size of the training set. 4.4

Sensitivity Analysis

Here we test the sensitivity of the performance with respect to the number of nearest neighbors. By changing the number of nearest neighbors from 5 to 40 at an interval of 5, we record the mean of the performance of our method over 10 trials in Figure 4. From the results, we can see our method is not very sensitive to the number of nearest neighbors, which makes the setting of k not very difficult.

0.06

Error

0.05 0.04 0.03 0.02 0.01

5

Conclusion

Letter USPS Robot

5
10
15 20 25 30 Number of Neighbors
35
40
Figure 4: Sensitivity analysis of the performance of our method with respect to the number of nearest neighbors at different data sets.

In this paper, we develop local learning methods for multi-task classification and regression problems. Based on an assumption that all task pairs contributes to each other almost equally, we propose regularized objective functions and develop efficient coordinate descent methods to solve them. Up to here, each task in our studies is a binary classification problem. In some applications, there may be more than two classes in each task. So we are interested in an extension of our method to multi-task multi-class problems. Currently the task-specific weights are shared by all data points from one task. One interesting research direction is to investigate a localized variant where different data points have different task-specific weights based on their locality structure.

Acknowledgment Yu Zhang is supported by HKBU ?Start Up Grant for New Academics?. 8

2 References

[1] R. K. Ando and T. Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. Journal of Machine Learning Research, 6:1817?1853, 2005. [2] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Sch?olkopf, J. C. Platt, and T. Hoffman, editors, Advances in Neural Information Processing Systems 19, pages 41?48, Vancouver, British Columbia, Canada, 2006. [3] B. Bakker and T. Heskes. Task clustering and gating for bayesian multitask learning. Journal of Machine Learning Research, 4:83?99, 2003. [4] J. Baxter. A Bayesian/information theoretic model of learning to learn via multiple task sampling. Machine Learning, 28(1):7?39, 1997. [5] J. C. Bezdek and R. J. Hathaway. Convergence of alternating optimization. Neural, Parallel & Scientific Computations, 11(4):351?368, 2003. [6] E. Bonilla, K. M. A. Chai, and C. Williams. Multi-task Gaussian process prediction. In J.C. Platt, D. Koller, Y. Singer, and S. Roweis, editors, Advances in Neural Information Processing Systems 20, pages 153?160, Vancouver, British Columbia, Canada, 2007. [7] L. Bottou and V. Vapnik. Local learning algorithms. Neural Computation, 4(6):888?900, 1992. [8] R. Caruana. Multitask learning. Machine Learning, 28(1):41?75, 1997. [9] T. Evgeniou and M. Pontil. Regularized multi-task learning. In Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pages 109?117, Seattle, Washington, USA, 2004. [10] T. V. Gestel, J. A. K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. Machine Learning, 54(1):5?32, 2004. [11] M. Grant and S. Boyd. CVX: Matlab software for disciplined convex

programming, 2011. [12] L. Jacob, F. Bach, and J.-P. Vert. Clustered multi-task learning: a convex formulation. In D. Koller, D. Schuurmans, Y. Bengio, and L. Bottou, editors, Advances in Neural Information Processing Systems 21, pages 745?752, Vancouver, British Columbia, Canada, 2008. [13] A. Kumar and H. Daum?e III. Learning task grouping and overlap in multi-task learning. In Proceedings of the 29 th International Conference on Machine Learning, Edinburgh, Scotland, UK, 2012. [14] S. Parameswaran and K. Weinberger. Large margin multi-task metric learning. In J. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, Advances in Neural Information Processing Systems 23, pages 1867?1875, 2010. [15] J. C. Platt. Fast training of support vector machines using sequential minimal optimization. In B. Sch?olkopf, C. J. C. Burges, and A. J. Smola, editors, Advances in Kernel Methods: Support Vector Learning. MIT Press, 1998. [16] S. Thrun. Is learning the n-th thing any easier than learning the first? In D. S. Touretzky, M. Mozer, and M. E. Hasselmo, editors, Advances in Neural Information Processing Systems 8, pages 640?646, Denver, CO, 1995. [17] S. Thrun and J. O?Sullivan. Discovering structure in multiple learning tasks: The TC algorithm. In Proceedings of the Thirteenth International Conference on Machine Learning, pages 489?497, Bari, Italy, 1996. [18] M. Wu and B. Sch?olkopf. A local learning approach for clustering. In B. Sch?olkopf, J. C. Platt, and T. Hoffman, editors, Advances in Neural Information Processing Systems 19, pages 1529?1536, Vancouver, British Columbia, Canada, 2006. [19] M. Wu, K. Yu, S. Yu, and B. Sch?olkopf. Local learning projections. In Proceedings of the Twenty-Fourth International Conference on Machine Learning, pages 1039?1046, Corvallis, Oregon, USA, 2007. [20] Y. Zhang and D.-Y. Yeung. A convex formulation for learning task relationships in multi-task learning. In Proceedings of the 26th Conference on Uncertainty in Artificial Intelligence, pages 733?742, Catalina Island, California, 2010.

9