Title: Multi-task learning

URL: https://en.wikipedia.org/wiki/Multi-task_learning

PageID: 938663

Categories: Category:Machine learning

Multi-task learning (MTL) is a subfield of machine learning in which multiple learning tasks are solved at the same time, while exploiting commonalities and differences across tasks. This can result in improved learning efficiency and prediction accuracy for the task-specific models, when compared to training the models separately. [1] [2] [3] Inherently, Multi-task learning is a multi-objective optimization problem having trade-offs between different tasks. [4] Early versions of MTL were called "hints". [5] [6]

In a widely cited 1997 paper, Rich Caruana gave the following characterization:

Multitask Learning is an approach to inductive transfer that improves generalization by using the domain information contained in the training signals of related tasks as an inductive bias . It does this by learning tasks in parallel while using a shared representation ; what is learned for each task can help other tasks be learned better. [3]

In the classification context, MTL aims to improve the performance of multiple classification tasks by learning them jointly. One example is a spam-filter, which can be treated as distinct but related classification tasks across different users. To make this more concrete, consider that different people have different distributions of features which distinguish spam emails from legitimate ones, for example an English speaker may find that all emails in Russian are spam, not so for Russian speakers. Yet there is a definite commonality in this classification task across users, for example one common feature might be text related to money transfer. Solving each user's spam classification problem jointly via MTL can let the solutions inform each other and improve performance. [citation needed] Further examples of settings for MTL include multiclass classification and multi-label classification . [7]

Multi-task learning works because regularization induced by requiring an algorithm to perform well on a related task can be superior to regularization that prevents overfitting by penalizing all complexity uniformly. One situation where MTL may be particularly helpful is if the tasks share significant commonalities and are generally slightly under sampled. [8] However, as discussed below, MTL has also been shown to be beneficial for learning unrelated tasks. [8] [9]

Methods

The key challenge in multi-task learning, is how to combine learning signals from multiple tasks into a single model. This may strongly depend on how well different task agree with each other, or contradict each other. There are several ways to address this challenge:

Task grouping and overlap

Within the MTL paradigm, information can be shared across some or all of the tasks. Depending on the structure of task relatedness, one may want to share information selectively across the tasks. For example, tasks may be grouped or exist in a hierarchy, or be related according to some general metric. Suppose, as developed more formally below, that the parameter vector modeling each task is a linear combination of some underlying basis. Similarity in terms of this basis can indicate the relatedness of the tasks. For example, with sparsity , overlap of nonzero coefficients across tasks indicates commonality. A task grouping then corresponds to those tasks lying in a subspace generated by some subset of basis elements, where tasks in different groups may be disjoint or overlap arbitrarily in terms of their bases. [10] Task relatedness can be imposed a priori or learned from the data. [7] [11] Hierarchical task relatedness can also be exploited implicitly without assuming a priori knowledge or learning relations explicitly. [8] [12] For example, the explicit learning of sample relevance across tasks can be done to guarantee the effectiveness of joint learning across multiple domains. [8]

Exploiting unrelated tasks

One can attempt learning a group of principal tasks using a group of auxiliary tasks, unrelated to the principal ones. In many applications, joint learning of unrelated tasks which use the same input data can be beneficial. The reason is that prior knowledge about task relatedness can lead to sparser and more informative representations for each task grouping, essentially by screening out idiosyncrasies of the data distribution. Novel methods which builds on a prior multitask methodology by favoring a shared low-dimensional representation within each task grouping have been proposed. The programmer can impose a penalty on tasks from different groups which encourages the two representations to be orthogonal . Experiments on synthetic and real data have indicated that incorporating unrelated tasks can result in significant improvements over standard multi-task learning methods. [9]

Transfer of knowledge

Related to multi-task learning is the concept of knowledge transfer. Whereas traditional multi-task learning implies that a shared representation is developed concurrently across tasks, transfer of knowledge implies a sequentially shared representation. Large scale machine learning projects such as the deep convolutional neural network GoogLeNet , [13] an image-based object classifier, can develop robust representations which may be useful to further algorithms learning related tasks. For example, the pre-trained model can be used as a feature extractor to perform pre-processing for another learning algorithm. Or the pre-trained model can be used to initialize a model with similar architecture which is then fine-tuned to learn a different classification task. [14]

Multiple non-stationary tasks

Traditionally Multi-task learning and transfer of knowledge are applied to stationary learning settings. Their extension to non-stationary environments is termed Group online adaptive learning (GOAL). [15] Sharing information could be particularly useful if learners operate in continuously changing environments, because a learner could benefit from previous experience of another learner to quickly adapt to their new environment. Such group-adaptive learning has numerous applications, from predicting financial time-series , through content recommendation systems, to visual understanding for adaptive autonomous agents.

Multi-task optimization

Multi-task optimization focuses on solving optimizing the whole process. [16] [17] The paradigm has been inspired by the well-established concepts of transfer learning [18] and multi-task learning in predictive analytics . [19]

The key motivation behind multi-task optimization is that if optimization tasks are related to each other in terms of their optimal solutions or the general characteristics of their function landscapes, [20] the search progress can be transferred to substantially accelerate the search on the other.

The success of the paradigm is not necessarily limited to one-way knowledge transfers from simpler to more complex tasks. In practice an attempt is to intentionally solve a more difficult task that may unintentionally solve several smaller problems. [21]

There is a direct relationship between multitask optimization and multi-objective optimization . [22]

In some cases, the simultaneous training of seemingly related tasks may hinder performance compared to single-task models. [23] Commonly, MTL models employ task-specific modules on top of a joint feature representation obtained using a shared module. Since this joint representation must capture useful features across all tasks, MTL may hinder individual task performance if the different tasks seek conflicting representation, i.e., the gradients of different tasks point to opposing directions or differ significantly in magnitude. This phenomenon is commonly referred to as negative transfer. To mitigate this issue, various MTL optimization methods have been proposed. Commonly, the per-task gradients are combined into a joint update direction through various aggregation algorithms or heuristics.

There are several common approaches for multi-task optimization: Bayesian optimization , evolutionary computation , and approaches based on Game theory . [16]

Multi-task Bayesian optimization

Multi-task Bayesian optimization is a modern model-based approach that leverages the concept of knowledge transfer to speed up the automatic hyperparameter optimization process of machine learning algorithms. [24] The method builds a multi-task Gaussian process model on the data originating from different searches progressing in tandem. [25] The captured inter-task dependencies are thereafter utilized to better inform the subsequent sampling of candidate solutions in respective search spaces.

Evolutionary multi-tasking

Evolutionary multi-tasking has been explored as a means of exploiting the implicit parallelism of population-based search algorithms to simultaneously progress multiple distinct optimization tasks. By mapping all tasks to a unified search space, the evolving population of candidate solutions can harness the hidden relationships between them through continuous genetic transfer. This is induced when solutions associated with different tasks crossover. [17] [26] Recently, modes of knowledge transfer that are different from direct solution crossover have been explored. [27] [28]

Game-theoretic optimization

Game-theoretic approaches to multi-task optimization propose to view the optimization problem as a game, where each task is a player. All players compete through the reward matrix of the game, and try to reach a solution that satisfies all players (all tasks). This view provide insight about how to build efficient algorithms based on gradient descent optimization (GD), which is particularly important for training deep neural networks . [29] In GD for MTL, the problem is that each task provides its own loss, and it is not clear how to combine all losses and create a single unified gradient, leading to several different aggregation strategies. [30] [31] [32] This aggregation problem can be solved by defining a game matrix where the reward of each player is the agreement of its own gradient with the common gradient, and then setting the common gradient to be the Nash Cooperative bargaining [33] of that system.

Applications

Algorithms for multi-task optimization span a wide array of real-world applications. Recent studies highlight the potential for speed-ups in the optimization of engineering design parameters by conducting related designs jointly in a multi-task manner. [26] In machine learning , the transfer of optimized features across related data sets can enhance the efficiency of the training process as well as improve the generalization capability of learned models. [34] [35] In addition, the concept of multi-tasking has led to advances in automatic hyperparameter optimization of machine learning models and ensemble learning . [36] [37]

Applications have also been reported in cloud computing, [38] with future developments geared towards cloud-based on-demand optimization services that can cater to multiple customers simultaneously. [17] [39] Recent work has additionally shown applications in chemistry. [40] In addition, some recent works have applied multi-task optimization algorithms in industrial manufacturing. [41] [42]

Mathematics

Reproducing Hilbert space of vector valued functions (RKHSvv)

The MTL problem can be cast within the context of RKHSvv (a complete inner product space of vector-valued functions equipped with a reproducing kernel). In particular, recent focus has been on cases where task structure can be identified via a separable kernel, described below. The presentation here derives from Ciliberto et al., 2015. [7]

RKHSvv concepts

Suppose the training data set is S t = { (x i t , y i t) } i = 1 n t {\displaystyle {\mathcal {S}}_{t}=\{(x_{i}^{t},y_{i}^{t})\}_{i=1}^{n_{t}}} , with x i t ∈ X {\displaystyle x_{i}^{t}\in {\mathcal {X}}} , y i t ∈ Y {\displaystyle y_{i}^{t}\in {\mathcal {Y}}} , where t indexes task, and t ∈ 1 , . . . , T

{\displaystyle t\in 1,...,T} . Let n = ∑ t = 1 T n t {\displaystyle n=\sum _{t=1}^{T}n_{t}} . In this setting there is a consistent input and output space and the same loss function L : R × R → R + {\displaystyle {\mathcal {L}}:\mathbb {R} \times \mathbb {R} \rightarrow \mathbb {R} _{+}} for each task: . This results in the regularized machine learning problem:

where H {\displaystyle {\mathcal {H}}} is a vector valued reproducing kernel Hilbert space with functions f : X → Y T {\displaystyle f:{\mathcal {X}}\rightarrow {\mathcal {Y}}^{T}} having components f t : X → Y {\displaystyle f_{t}:{\mathcal {X}}\rightarrow {\mathcal {Y}}} .

The reproducing kernel for the space H {\displaystyle {\mathcal {H}}} of functions f : X → R T {\displaystyle f:{\mathcal {X}}\rightarrow \mathbb {R} ^{T}} is a symmetric matrix-valued function Γ : X × X → R T × T {\displaystyle \Gamma :{\mathcal {X}}\times {\mathcal {X}}\rightarrow \mathbb {R} ^{T\times T}} , such that Γ (⋅ , x) c ∈ H {\displaystyle \Gamma (\cdot ,x)c\in {\mathcal {H}}} and the following reproducing property holds:

The reproducing kernel gives rise to a representer theorem showing that any solution to equation 1 has the form:

Separable kernels

The form of the kernel Γ induces both the representation of the feature space and structures the output across tasks. A natural simplification is to choose a separable kernel, which factors into separate kernels on the input space X and on the tasks { 1 , . . . , T } {\displaystyle \{1,...,T\}} . In this case the kernel relating scalar components f t {\displaystyle f_{t}} and f s {\displaystyle f_{s}} is given by γ ((x i , t) , (x j , s)) = k (x i , x j) k T (s , t) = k (x i , x j) A s , t {\textstyle \gamma ((x_{i},t),(x_{j},s))=k(x_{i},x_{j})k_{T}(s,t)=k(x_{i},x_{j})A_{s,t}} . For vector valued functions f ∈ H {\displaystyle f\in {\mathcal {H}}} we can write Γ (x i , x j) = k (x i , x j) A {\displaystyle \Gamma (x_{i},x_{j})=k(x_{i},x_{j})A} , where k is a scalar reproducing kernel, and A is a symmetric positive semi-definite T × T {\displaystyle T\times T} matrix. Henceforth denote S + T = { PSD matrices } ⊂ R T × T {\displaystyle S_{+}^{T}=\{{\text{PSD matrices}}\}\subset \mathbb {R} ^{T\times T}} .

This factorization property, separability, implies the input feature space representation does not vary by task. That is, there is no interaction between the input kernel and the task kernel. The structure on tasks is represented solely by A . Methods for non-separable kernels Γ is a current field of research.

For the separable case, the representation theorem is reduced to f (x) = ∑ i = 1 N k (x , x i) A c i {\textstyle f(x)=\sum _{i=1}^{N}k(x,x_{i})Ac_{i}} . The model output on the training data is then KCA , where K is the n × n {\displaystyle n\times n} empirical kernel matrix with entries K i , j = k (x i , x j) {\textstyle K_{i,j}=k(x_{i},x_{j})} , and C is the n × T {\displaystyle n\times T} matrix of rows c i {\displaystyle c_{i}} .

With the separable kernel, equation 1 can be rewritten as

where V is a (weighted) average of L applied entry-wise to Y and KCA . (The weight is zero if Y i t {\displaystyle Y_{i}^{t}} is a missing observation).

Note the second term in P can be derived as follows:

Known task structure

There are three largely equivalent ways to represent task structure: through a regularizer; through an output metric, and through an output mapping.

Regularizer — With the separable kernel, it can be shown (below) that | | f | | H 2 = ∑ s , t = 1 T A t , s † ■ f s , f t ■ H k {\textstyle ||f||_{\mathcal {H}}^{2}=\sum _{s,t=1}^{T}A_{t,s}^{\dagger }\langle f_{s},f_{t}\rangle _{{\mathcal {H}}_{k}}} , where A t , s † {\displaystyle A_{t,s}^{\dagger }} is the t , s {\displaystyle t,s} element of the pseudoinverse of A {\displaystyle A} , and H k {\displaystyle {\mathcal {H}}_{k}} is the RKHS based on the scalar kernel k {\displaystyle k} , and f t (x) = ∑ i = 1 n k (x , x i) A t ■ c i {\textstyle f_{t}(x)=\sum _{i=1}^{n}k(x,x_{i})A_{t}^{\top }c_{i}} . This formulation shows that A t , s † {\displaystyle A_{t,s}^{\dagger }} controls the weight of the penalty associated with ■ f s , f t ■ H k {\textstyle \langle f_{s},f_{t}\rangle _{{\mathcal {H}}_{k}}} . (Note that ■ f s , f t ■ H k {\textstyle \langle f_{s},f_{t}\rangle _{{\mathcal {H}}_{k}}} arises from | | f t | | H k = ■ f t , f t ■ H k

{\textstyle ||f_{t}||_{{\mathcal {H}}_{k}}=\langle f_{t},f_{t}\rangle _{{\mathcal {H}}_{k}}} .)

■ f ■ H 2 = ■ ∑ i = 1 n γ ((x i , t i) , ·) c i t i , ∑ j = 1 n γ ((x j , t j) , ·) c j t j ■ H = ∑ i , j = 1 n c i t i c j t j γ ((x i , t i) , (x j , t j)) = ∑ i , j = 1 n ∑ s , t = 1 T c i t c j s k (x i , x j) A s , t = ∑ i , j = 1 n k (x i , x j) ■ c i , A c j ■ R T = ∑ i , j = 1 n k (x i , x j) ■ c i , A A † A c j ■ R T = ∑ i , j = 1 n k (x i , x j) ■ A c i , A † A c j ■ R T = ∑ i , j = 1 n ∑ s , t = 1 T (A c i) t (A c j) s k (x i , x j) A s , t † = ∑ s , t = 1 T A s , t † ■ ∑ i = 1 n k (x i , ·) (A c i) t , ∑ j = 1 n k (x j , ·) (A c j) s ■ H k = ∑ s , t = 1 T A s , t † ■ f t , f s ■ H k {\displaystyle {\begin{aligned}\|f\|_{\mathcal {H}}^{2}&=\left\langle \sum _{i=1}^{n}\gamma ((x_{i},t_{i}),\cdot)c_{i}^{t_{i}},\sum _{j=1}^{n}\gamma ((x_{j},t_{j}),\cdot)c_{j}^{t_{j}}\right\rangle _{\mathcal {H}}\\&=\sum _{i,j=1}^{n}c_{i}^{t_{i}}c_{j}^{t_{j}}\gamma ((x_{i},t_{i}),(x_{j},t_{j}))\\&=\sum _{i,j=1}^{n}\sum _{s,t=1}^{T}c_{i}^{t}c_{j}^{s}k(x_{i},x_{j})A_{s,t}\\&=\sum _{i,j=1}^{n}k(x_{i},x_{j})\langle c_{i},Ac_{j}\rangle _{\mathbb {R} ^{T}}\\&=\sum _{i,j=1}^{n}k(x_{i},x_{j})\langle c_{i},AA^{\dagger }Ac_{j}\rangle _{\mathbb {R} ^{T}}\\&=\sum _{i,j=1}^{n}k(x_{i},x_{j})\langle Ac_{i},A^{\dagger }Ac_{j}\rangle _{\mathbb {R} ^{T}}\\&=\sum _{i,j=1}^{n}\sum _{s,t=1}^{T}(Ac_{i})^{t}(Ac_{j})^{s}k(x_{i},x_{j})A_{s,t}^{\dagger }\\&=\sum _{s,t=1}^{T}A_{s,t}^{\dagger }\langle \sum _{i=1}^{n}k(x_{i},\cdot)(Ac_{i})^{t},\sum _{j=1}^{n}k(x_{j},\cdot)(Ac_{j})^{s}\rangle _{{\mathcal {H}}_{k}}\\&=\sum _{s,t=1}^{T}A_{s,t}^{\dagger }\langle f_{t},f_{s}\rangle _{{\mathcal {H}}_{k}}\end{aligned}}}

Output metric — an alternative output metric on Y T {\displaystyle {\mathcal {Y}}^{T}} can be induced by the inner product ■ y 1 , y 2 ■ Θ = ■ y 1 , Θ y 2 ■ R T {\displaystyle \langle y_{1},y_{2}\rangle _{\Theta }=\langle y_{1},\Theta y_{2}\rangle _{\mathbb {R} ^{T}}} . With the squared loss there is an equivalence between the separable kernels k (· , ·) I T {\displaystyle k(\cdot ,\cdot)I_{T}} under the alternative metric, and k (· , ·) Θ {\displaystyle k(\cdot ,\cdot)\Theta } , under the canonical metric.

Output mapping — Outputs can be mapped as L : Y T → Y ~ {\displaystyle L:{\mathcal {Y}}^{T}\rightarrow {\mathcal {\tilde {Y}}}} to a higher dimensional space to encode complex structures such as trees, graphs and strings. For linear maps L , with appropriate choice of separable kernel, it can be shown that A = L ■ L {\displaystyle A=L^{\top }L} .

Via the regularizer formulation, one can represent a variety of task structures easily.

Letting A † = γ I T + (γ − λ) 1 T 1 1 ■ {\textstyle A^{\dagger }=\gamma I_{T}+(\gamma -\lambda){\frac {1}{T}}\mathbf {1} \mathbf {1} ^{\top }} (where I T {\displaystyle I_{T}} is the T x T identity matrix, and 1 1 ■ {\textstyle \mathbf {1} \mathbf {1} ^{\top }} is the T x T matrix of ones) is equivalent to letting Γ control the variance ∑ t | | f t − f ¯ | | H k {\textstyle \sum _{t}||f_{t}-{\bar {f}}||_{{\mathcal {H}}_{k}}} of tasks from their mean 1 T ∑ t f t {\textstyle {\frac {1}{T}}\sum _{t}f_{t}} . For example, blood levels of some biomarker may be taken on T patients at n t {\displaystyle n_{t}} time points during the course of a day and interest may lie in regularizing the variance of the predictions across patients.

Letting A † = α I T + (α − λ) M {\displaystyle A^{\dagger }=\alpha I_{T}+(\alpha -\lambda)M} , where M t , s = 1 | G r | I (t , s ∈ G r) {\displaystyle M_{t,s}={\frac {1}{|G_{r}|}}\mathbb {I} (t,s\in G_{r})} is equivalent to letting α {\displaystyle \alpha } control the variance measured with respect to a group mean: ∑ r ∑ t ∈ G r | | f t − 1 | G r | ∑ s ∈ G r) f s | | {\displaystyle \sum _{r}\sum _{t\in G_{r}}||f_{t}-{\frac {1}{|G_{r}|}}\sum _{s\in G_{r})}f_{s}||} . (Here | G r | {\displaystyle |G_{r}|} the cardinality of group r, and I {\displaystyle \mathbb {I} } is the indicator function). For example, people in different political parties (groups) might be regularized together with respect to predicting the favorability rating of a politician. Note that this penalty reduces to the first when all tasks are in the same group.

Letting A † = δ I T + (δ − λ) L {\displaystyle A^{\dagger }=\delta I_{T}+(\delta -\lambda)L} , where L = D − M {\displaystyle L=D-M} is the Laplacian for the graph with adjacency matrix M giving pairwise similarities of tasks. This is equivalent to giving a larger penalty to the distance separating tasks t and s when they are more similar (according to the weight M t , s {\displaystyle M_{t,s}} ,) i.e. δ {\displaystyle \delta } regularizes ∑ t , s | | f t − f s | | H k 2 M t , s {\displaystyle \sum _{t,s}||f_{t}-f_{s}||_{{\mathcal {H}}_{k}}^{2}M_{t,s}} .

All of the above choices of A also induce the additional regularization term λ ∑ t | | f | | H k 2 {\textstyle \lambda \sum _{t}||f||_{{\mathcal {H}}_{k}}^{2}} which penalizes complexity in f more broadly.

Learning tasks together with their structure

Learning problem P can be generalized to admit learning task matrix A as follows:

Choice of F : S + T → R + {\displaystyle F:S_{+}^{T}\rightarrow \mathbb {R} _{+}} must be designed to learn matrices A of a given type. See "Special cases" below.

Restricting to the case of convex losses and coercive penalties Ciliberto et al. have shown that although Q is not convex jointly in C and A, a related problem is jointly convex.

Specifically on the convex set C = { (C , A) ∈ R n × T × S + T | R a n g e (C ■ K C) ⊆ R a n g e (A) } {\displaystyle {\mathcal {C}}=\{(C,A)\in \mathbb {R} ^{n\times T}\times S_{+}^{T}|Range(C^{\top }KC)\subseteq Range(A)\}} , the equivalent problem

is convex with the same minimum value. And if (C R , A R) {\displaystyle (C_{R},A_{R})} is a minimizer for R then (C R A R † , A R) {\displaystyle (C_{R}A_{R}^{\dagger },A_{R})} is a minimizer for Q .

R may be solved by a barrier method on a closed set by introducing the following perturbation:

The perturbation via the barrier δ 2 t r (A †) {\displaystyle \delta ^{2}tr(A^{\dagger })} forces the objective functions to be equal to + ∞ {\displaystyle +\infty } on the boundary of R n × T × S + T {\displaystyle R^{n\times T}\times S_{+}^{T}} .

S can be solved with a block coordinate descent method, alternating in C and A. This results in a sequence of minimizers (C m , A m) {\displaystyle (C_{m},A_{m})} in S that converges to the solution in R as δ m → 0 {\displaystyle \delta _{m}\rightarrow 0} , and hence gives the solution to Q .

Spectral penalties - Dinnuzo et al [43] suggested setting F as the Frobenius norm t r (A ■ A) {\displaystyle {\sqrt {tr(A^{\top }A)}}} . They optimized Q directly using block coordinate descent, not accounting for difficulties at the boundary of R n × T × S + T {\displaystyle \mathbb {R} ^{n\times T}\times S_{+}^{T}} .

Clustered tasks learning - Jacob et al [44] suggested to learn A in the setting where T tasks are organized in R disjoint clusters. In this case let E ∈ { 0 , 1 } T × R {\displaystyle E\in \{0,1\}^{T\times R}} be the matrix with E t , r = I (task t ∈ group r) {\displaystyle E_{t,r}=\mathbb {I} ({\text{task }}t\in {\text{group }}r)} . Setting M = I − E † E T {\displaystyle M=I-E^{\dagger }E^{T}} , and U = 1 T 11 ■ {\displaystyle U={\frac {1}{T}}\mathbf {11} ^{\top }} , the task matrix A † {\displaystyle A^{\dagger }} can be parameterized as a function of M {\displaystyle M} : A † (M) = ■ M U + ■ B (M − U) + ■ (I − M) {\displaystyle A^{\dagger }(M)=\epsilon _{M}U+\epsilon _{B}(M-U)+\epsilon (I-M)} , with terms that penalize the average, between clusters variance and within clusters variance respectively of the task predictions. M is not convex, but there is a convex relaxation S c = { M ∈ S + T : I − M ∈ S + T ∧ t r (M) = r } {\displaystyle {\mathcal {S}}_{c}=\{M\in S_{+}^{T}:I-M\in S_{+}^{T}\land tr(M)=r\}} . In this formulation, F (A) = I (A (M) ∈ { A : M ∈ S C }) {\displaystyle F(A)=\mathbb {I} (A(M)\in \{A:M\in {\mathcal {S}}_{C}\})} .

Non-convex penalties - Penalties can be constructed such that A is constrained to be a graph Laplacian, or that A has low rank factorization. However these penalties are not convex, and the analysis of the barrier method proposed by Ciliberto et al. does not go through in these cases.

Non-separable kernels - Separable kernels are limited, in particular they do not account for structures in the interaction space between the input and output domains jointly. Future work is needed to develop models for these kernels.

Software package

A Matlab package called Multi-Task Learning via StructurAl Regularization (MALSAR) [45] implements the following multi-task learning algorithms: Mean-Regularized Multi-Task Learning, [46] [47] Multi-Task Learning with Joint Feature Selection, [48] Robust Multi-Task Feature Learning, [49] Trace-Norm Regularized Multi-Task Learning, [50] Alternating Structural Optimization, [51] [52] Incoherent Low-Rank and Sparse Learning, [53] Robust Low-Rank Multi-Task Learning, Clustered Multi-Task Learning, [54] [55] Multi-Task Learning with Graph

Structures.

Literature

Multi-Target Prediction: A Unifying View on Problems and Methods Willem Waegeman, Krzysztof Dembczynski, Eyke Huellermeier https://arxiv.org/abs/1809.02352v1

See also

Artificial intelligence

Artificial neural network

Automated machine learning (AutoML)

Evolutionary computation

Foundation model

General game playing

Human-based genetic algorithm

Kernel methods for vector output

Multiple-criteria decision analysis

Multi-objective optimization

Multicriteria classification

Robot learning

Transfer learning

James–Stein estimator

References

External links

The Biosignals Intelligence Group at UIUC

Washington University in St. Louis Department of Computer Science

Software

The Multi-Task Learning via Structural Regularization Package

Online Multi-Task Learning Toolkit (OMT) A general-purpose online multi-task learning toolkit based on conditional random field models and stochastic gradient descent training (C# , .NET)

v

t

e

Golden-section search

Powell's method

Line search

Nelder–Mead method

Successive parabolic interpolation

Trust region

Wolfe conditions

Berndt–Hall–Hall–Hausman

Broyden–Fletcher–Goldfarb–Shanno and L-BFGS

Davidon–Fletcher–Powell

Symmetric rank-one (SR1)

Conjugate gradient

Gauss–Newton

Gradient

Mirror

Levenberg–Marquardt

Powell's dog leg method

Truncated Newton

Newton's method

Barrier methods

Penalty methods

Augmented Lagrangian methods

Sequential quadratic programming

Successive linear programming

Cutting-plane method

Reduced gradient (Frank–Wolfe)

Subgradient method

Affine scaling

Ellipsoid algorithm of Khachiyan

Projective algorithm of Karmarkar

Simplex algorithm of Dantzig

Revised simplex algorithm

Criss-cross algorithm

Principal pivoting algorithm of Lemke

Active-set method

Approximation algorithm

Dynamic programming

Greedy algorithm

Integer programming Branch and bound / cut

Branch and bound / cut

Bor■vka

Prim

Kruskal

Bellman–Ford SPFA

SPFA

Dijkstra

Floyd–Warshall

Dinic

Edmonds–Karp

Ford–Fulkerson

Push–relabel maximum flow

Evolutionary algorithm

Hill climbing

Local search

Parallel metaheuristics

Simulated annealing

Spiral optimization algorithm

Tabu search

Software