WikipediA

Multi-task 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] Early versions of MTL were called "hints"[4][5]

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.[6] 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.[6] However, as discussed below, MTL has also been shown to be beneficial for learning unrelated tasks.[8]

Contents

Methods

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.[9] Task relatedness can be imposed a priori or learned from the data.[7][10]. Hierarchical task relatedness can also be exploited implicitly without assuming a priori knowledge or learning relations explicitly.[11]

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.[8]

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,[12] 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.[13]

Group online adaptive learning

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).[14] 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.

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 $\mathcal{S}_t = \{(x_i^t, y_i^t)\}_{i=1}^{n_t}$, with $x_i^t \in \mathcal{X}$, $y_i^t \in \mathcal{Y}$, where t indexes task, and $t \in 1, \ldots, T$. Let $n = \sum_{t=1}^{T} n_t$. In this setting there is a consistent input and output space and the same loss function $\mathcal{L} : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_+$ for each task: . This results in the regularized machine learning problem:

$$\min_{f \in \mathcal{H}} \sum_{t=1}^{T} \frac{1}{n_t} \sum_{i=1}^{n_t} \mathcal{L}(y_i^t, f_t(x_i^t)) + \lambda ||f||_{\mathcal{H}}^2 \qquad (1)$$

where $\mathcal{H}$ is a vector valued reproducing kernel Hilbert space with functions $f : \mathcal{X} \to \mathcal{Y}^T$ having components $f_t : \mathcal{X} \to \mathcal{Y}$.

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

$$\langle f(x), c \rangle_{\mathbb{R}^T} = \langle f, \Gamma(x, \cdot)c \rangle_{\mathcal{H}} \qquad (2)$$

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

$$f(x) = \sum_{t=1}^{T} \sum_{i=1}^{n_t} \Gamma(x, x_i^t) c_i^t \qquad (3)$$

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 $\mathcal{X}$ and on the tasks $\{1, \ldots, T\}$. In this case the kernel relating scalar components f_t and f_s is given by $\gamma((x_i, t), (x_j, s)) = k(x_i, x_j) k_T(s, t) = k(x_i, x_j) A_{s,t}$. For vector valued functions $f \in \mathcal{H}$ we can write $\Gamma(x_i, x_j) = k(x_i, x_j)A$, where k is a scalar reproducing kernel, and A is a symmetric positive semi-definite $T \times T$ matrix. Henceforth denote $S_+^T = \{\textbf{PSD matrices}\} \subset \mathbb{R}^{T \times T}$.

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 an current field of research.

For the separable case, the representation theorem is reduced to $f(x) = \sum_{i=1}^{N} k(x, x_i) A c_i$. The model output on the training data is then KCA , where K is the $n \times n$ empirical kernel matrix with entries $K_{i,j} = k(x_i, x_j)$, and C is the $n \times T$ matrix of rows c_i.

With the separable kernel, equation 1 can be rewritten as

$$\min_{C \in \mathbb{R}^{n \times T}} V(Y, KCA) + \lambda tr(KCAC^\top) \qquad (P)$$

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

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

$$||f||_{\mathcal{H}}^2 = \langle \sum_{i=1}^{n} k(\cdot, x_i) A c_i, \sum_{j=1}^{n} k(\cdot, x_j) A c_j \rangle_{\mathcal{H}}$$

$$= \sum_{i,j=1}^{n} \langle k(\cdot, x_i) A c_i, k(\cdot, x_j) A c_j \rangle_{\mathcal{H}} \text{ (bilinearity)}$$

$$= \sum_{i,j=1}^{n} \langle k(x_i, x_j) A c_i, c_j \rangle_{\mathbb{R}^T} \text{ (reproducing property)}$$

$$= \sum_{i,j=1}^{n} k(x_i, x_j) c_i^\top A c_j = tr(KCAC^\top)$$

Known task structure

Task structure representations

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 $||f||^2_{\mathcal{H}} = \sum_{s,t=1}^{T} A^{\dagger}_{t,s} \langle f_s, f_t \rangle_{\mathcal{H}_k}$, where $A^{\dagger}_{t,s}$ is the t, s element of the pseudoinverse of A, and $\mathcal{H}_k$ is the RKHS based on the scalar kernel k, and $f_t(x) = \sum_{i=1}^{n} k(x, x_i) A_t^{\top} c_i$. This formulation shows that $A^{\dagger}_{t,s}$ controls the weight of the penalty associated with $\langle f_s, f_t \rangle_{\mathcal{H}_k}$. (Note that $\langle f_s, f_t \rangle_{\mathcal{H}_k}$ arises from $||f_t||_{\mathcal{H}_k} = \langle f_t, f_t \rangle_{\mathcal{H}_k}$.)

Proof:

$$||f||^2_{\mathcal{H}} = \langle \sum_{i=1}^{n} \gamma((x_i, t_i), \cdot) c_i^{t_i}, \sum_{j=1}^{n} \gamma((x_j, t_j), \cdot) c_j^{t_j} \rangle_{\mathcal{H}}$$

$$= \sum_{i,j=1}^{n} c_i^{t_i} c_j^{t_j} \gamma((x_i, t_i), (x_j, t_j))$$

$$= \sum_{i,j=1}^{n} \sum_{s,t=1}^{T} c_i^t c_j^s k(x_i, x_j) A_{s,t}$$

$$= \sum_{i,j=1}^{n} k(x_i, x_j) \langle c_i, A c_j \rangle_{\mathbb{R}^T}$$

$$= \sum_{i,j=1}^{n} k(x_i, x_j) \langle c_i, A A^{\dagger} A c_j \rangle_{\mathbb{R}^T}$$

$$= \sum_{i,j=1}^{n} k(x_i, x_j) \langle A c_i, A^{\dagger} A c_j \rangle_{\mathbb{R}^T}$$

$$= \sum_{i,j=1}^{n} \sum_{s,t=1}^{T} (A c_i)^t (A c_j)^s k(x_i, x_j) A^{\dagger}_{s,t}$$

$$= \sum_{s,t=1}^{T} A^{\dagger}_{s,t} \langle \sum_{i=1}^{n} k(x_i, \cdot)(A c_i)^t, \sum_{j=1}^{n} k(x_j, \cdot)(A c_j)^s \rangle_{\mathcal{H}_k}$$

$$= \sum_{s,t=1}^{T} A^{\dagger}_{s,t} \langle f_t, f_s \rangle_{\mathcal{H}_k}$$

Output metric - an alternative output metric on $\mathcal{Y}^T$ can be induced by the inner product $\langle y_1, y_2 \rangle_{\Theta} = \langle y_1, \Theta y_2 \rangle_{\mathbb{R}^T}$. With the squared loss there is an equivalence between the separable kernels $k(\cdot, \cdot) I_T$ under the alternative metric, and $k(\cdot, \cdot)\Theta$, under the canonical metric.

Output mapping - Outputs can be mapped as $L : \mathcal{Y}^T \to \tilde{\mathcal{Y}}$ 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 $A = L^{\top} L$.

Task structure examples

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

- Letting $A^{\dagger} = \gamma I_T + (\gamma - \lambda) \frac{1}{T} \mathbf{1}\mathbf{1}^{\top}$ (where I_T is the *T*x*T* identity matrix, and $\mathbf{1}\mathbf{1}^{\top}$ is the *T*x*T* matrix of ones) is equivalent to letting γ control the variance $\sum_t ||f_t - \bar{f}||_{\mathcal{H}_k}$ of tasks from their mean $\frac{1}{T} \sum_t f_t$. For example, blood levels of some biomarker may be taken on T patients at n_t time points during the course of a day and interest may lie in regularizing the variance of the predictions across patients.
- Letting $A^{\dagger} = \alpha I_T + (\alpha - \lambda) M$, where $M_{t,s} = \frac{1}{|G_r|} \mathbb{I}(t, s \in G_r)$ is equivalent to letting α control the variance measured with respect to a group mean: $\sum_r \sum_{t \in G_r} ||f_t - \frac{1}{|G_r|} \sum_{s \in G_r)} f_s||$. (Here $|G_r|$ the cardinality of group r, and $\mathbb{I}$ 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 $A^{\dagger} = \delta I_T + (\delta - \lambda) L$, where $L = D - M$ 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 $M_{t,s}$,) i.e. δ regularizes $\sum_{t,s} ||f_t - f_s||^2_{\mathcal{H}_k} M_{t,s}$.
- All of the above choices of A also induce the additional regularization term $\lambda \sum_t ||f||^2_{\mathcal{H}_k}$ 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:

$$\min_{C \in \mathbb{R}^{n \times T}, A \in S_+^T} V(Y, KCA) + \lambda tr(KCAC^{\top}) + F(A) \qquad (Q)$$

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

Optimization of Q

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 $\mathcal{C} = \{(C, A) \in \mathbb{R}^{n \times T} \times S_+^T | Range(C^{\top} KC) \subseteq Range(A)\}$, the equivalent problem

$$\min_{C, A \in \mathcal{C}} V(Y, KC) + \lambda tr(A^{\dagger} C^{\top} KC) + F(A) \qquad (R)$$

is convex with the same minimum value. And if (C_R, A_R) is a minimizer for **R** then $(C_R A_R^{\dagger}, A_R)$ is a minimizer for **Q**.

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

$$\min_{C \in \mathbb{R}^{n \times T}, A \in S_+^T} V(Y, KC) + \lambda tr(A^\dagger (C^\top KC + \delta^2 I_T)) + F(A) \qquad (S)$$

The perturbation via the barrier $\delta^2 tr(A^\dagger)$ forces the objective functions to be equal to $+\infty$ on the boundary of $R^{n \times T} \times S_+^T$.

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) in **S** that converges to the solution in **R** as $\delta_m \to 0$, and hence gives the solution to **Q**.

Special cases

Spectral penalties - Dinnuzo *et al*[15] suggested setting *F* as the Frobenius norm $\sqrt{tr(A^\top A)}$. They optimized **Q** directly using block coordinate descent, not accounting for difficulties at the boundary of $\mathbb{R}^{n \times T} \times S_+^T$.

Clustered tasks learning - Jacob *et al*[16] suggested to learn *A* in the setting where *T* tasks are organized in *R* disjoint clusters. In this case let $E \in \{0,1\}^{T \times R}$ be the matrix with $E_{t,r} = \mathbb{I}(\textbf{task } t \in \textbf{group } r)$. Setting $M = I - E^\dagger E^T$, and $U = \frac{1}{T}\mathbf{1}\mathbf{1}^\top$, the task matrix $A^\dagger$ can be parameterized as a function of M: $A^\dagger(M) = \epsilon_M U + \epsilon_B(M - U) + \epsilon(I - M)$, 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 $\mathcal{S}_c = \{M \in S_+^T : I - M \in S_+^T \wedge tr(M) = r\}$. In this formulation, $F(A) = \mathbb{I}(A(M) \in \{A : M \in \mathcal{S}_C\})$.

Generalizations

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.

Applications

Spam filtering

Using the principles of MTL, techniques for collaborative spam filtering that facilitates personalization have been proposed. In large scale open membership email systems, most users do not label enough messages for an individual local classifier to be effective, while the data is too noisy to be used for a global filter across all users. A hybrid global/individual classifier can be effective at absorbing the influence of users who label emails very diligently from the general public. This can be accomplished while still providing sufficient quality to users with few labeled instances.[17]

Web search

Using boosted decision trees, one can enable implicit data sharing and regularization. This learning method can be used on web-search ranking data sets. One example is to use ranking data sets from several countries. Here, multitask learning is particularly helpful as data sets from different countries vary largely in size because of the cost of editorial judgments. It has been demonstrated that learning various tasks jointly can lead to significant improvements in performance with surprising reliability.[18]

RoboEarth

In order to facilitate transfer of knowledge, IT infrastructure is being developed. One such project, RoboEarth, aims to set up an open source internet database that can be accessed and continually updated from around the world. The goal is to facilitate a cloud-based interactive knowledge base, accessible to technology companies and academic institutions, which can enhance the sensing, acting and learning capabilities of robots and other artificial intelligence agents.[19]

Software package

The Multi-Task Learning via StructurAl Regularization (MALSAR) Matlab package[20] implements the following multi-task learning algorithms:

- Mean-Regularized Multi-Task Learning[21][22]
- Multi-Task Learning with Joint Feature Selection[23]
- Robust Multi-Task Feature Learning[24]
- Trace-Norm Regularized Multi-Task Learning[25]
- Alternating Structural Optimization[26][27]
- Incoherent Low-Rank and Sparse Learning[28]
- Robust Low-Rank Multi-Task Learning
- Clustered Multi-Task Learning[29][30]
- Multi-Task Learning with Graph Structures

See also

- Artificial Intelligence
- Artificial neural network
- Evolutionary computation
- Human-based genetic algorithm
- Kernel methods for vector output
- Machine Learning
- Robotics

References

1. Baxter, J. (2000). A model of inductive bias learning" *Journal of Artificial Intelligence Research* 12:149--198, On-line paper (http://www-2.cs.cmu.edu/afs/cs/project/jair/pub/volume12/baxter00a.pdf)
2. Thrun, S. (1996). Is learning the n-th thing any easier than learning the first?. In Advances in Neural Information Processing Systems 8, pp. 640--646. MIT Press. Paper at Citeseer (http://citeseer.ist.psu.edu/thrun96is.html)

3. Caruana, R. (1997). "Multi-task learning" (http://www.cs.cornell.edu/~caruana/mlj97.pdf) (PDF). *Machine Learning*. 28: 41–75. doi:10.1023/A:1007379606734 (https://doi.org/10.1023%2FA%3A1007379606734).
4. Suddarth, S., Kergosien, Y. (1990). Rule-injection hints as a means of improving network performance and learning time. EURASIP Workshop. Neural Networks pp. 120-129. Lecture Notes in Computer Science. Springer.
5. Abu-Mostafa, Y. S. (1990). "Learning from hints in neural networks". *Journal of Complexity*. 6: 192–198. doi:10.1016/0885-064x(90)90006-y (https://doi.org/10.1016%2F0885-064x%2890%2990006-y).
6. Weinberger, Kilian. "Multi-task Learning" (http://www.cs.cornell.edu/~kilian/research/multitasklearning/multitasklearning.html).
7. Ciliberto, C. (2015). "Convex Learning of Multiple Tasks and their Structure". arXiv:1504.03101 (https://arxiv.org/abs/1504.03101).
8. Romera-Paredes, B., Argyriou, A., Bianchi-Berthouze, N., & Pontil, M., (2012) Exploiting Unrelated Tasks in Multi-Task Learning. http://jmlr.csail.mit.edu/proceedings/papers/v22/romera12/romera12.pdf
9. Kumar, A., & Daume III, H., (2012) Learning Task Grouping and Overlap in Multi-Task Learning. http://icml.cc/2012/papers/690.pdf
10. Jawanpuria, P., & Saketha Nath, J., (2012) A Convex Feature Learning Formulation for Latent Task Structure Discovery. http://icml.cc/2012/papers/90.pdf
11. Zweig, A. & Weinshall, D. Hierarchical Regularization Cascade for Joint Learning. Proceedings: of 30th International Conference on Machine Learning (ICML), Atlanta GA, June 2013. http://www.cs.huji.ac.il/~daphna/papers/Zweig_ICML2013.pdf
12. Szegedy, C. (2014). "Going Deeper with Convolutions". *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*. arXiv:1409.4842 (https://arxiv.org/abs/1409.4842). doi:10.1109/CVPR.2015.7298594 (https://doi.org/10.1109%2FCVPR.2015.7298594).
13. Roig, Gemma. "Deep Learning Overview" (http://www.mit.edu/~9.520/fall15/slides/class24/deep_learning_overview.pdf) (PDF).
14. Zweig, A. & Chechik, G. Group online adaptive learning. Machine Learning, DOI 10.1007/s10994-017- 5661-5, August 2017. http://rdcu.be/uFSv
15. Dinuzzo, Francesco (2011). "Learning output kernels with block coordinate descent.". *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*.
16. Jacob, Laurent (2009). "Clustered multi-task learning: A convex formulation". *Advances in neural information processing systems*.
17. Attenberg, J., Weinberger, K., & Dasgupta, A. Collaborative Email-Spam Filtering with the Hashing-Trick. http://www.cse.wustl.edu/~kilian/papers/ceas2009-paper-11.pdf
18. Chappelle, O., Shivaswamy, P., & Vadrevu, S. Multi-Task Learning for Boosting with Application to Web Search Ranking. http://www.cse.wustl.edu/~kilian/papers/multiboost2010.pdf
19. Description of RoboEarth Project (http://www.roboearth.org/motivation)
20. Zhou, J., Chen, J. and Ye, J. MALSAR: Multi-tAsk Learning via StructurAl Regularization. Arizona State University, 2012. http://www.public.asu.edu/~jye02/Software/MALSAR. On-line manual (http://www.public.asu.edu/~jye02/Software/MALSAR/Manual.pdf)
21. Evgeniou, T., & Pontil, M. (2004). Regularized multi–task learning. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 109–117).
22. Evgeniou, T.; Micchelli, C.; Pontil, M. (2005). "Learning multiple tasks with kernel methods" (http://jmlr.org/papers/volume6/evgeniou05a/evgeniou05a.pdf) (PDF). *Journal of Machine Learning Research*. 6: 615.
23. Argyriou, A.; Evgeniou, T.; Pontil, M. (2008a). "Convex multi-task feature learning". *Machine Learning*. 73: 243–272. doi:10.1007/s10994-007-5040-8 (https://doi.org/10.1007%2Fs10994-007-5040-8).
24. Chen, J., Zhou, J., & Ye, J. (2011). Integrating low-rank and group-sparse structures for robust multi-task learning. Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining.
25. Ji, S., & Ye, J. (2009). An accelerated gradient method for trace norm minimization. Proceedings of the 26th Annual International Conference on Machine Learning (pp. 457–464).
26. Ando, R.; Zhang, T. (2005). "A framework for learning predictive structures from multiple tasks and unlabeled data". *The Journal of Machine Learning Research*. 6: 1817–1853.
27. Chen, J., Tang, L., Liu, J., & Ye, J. (2009). A convex formulation for learning shared structures from multiple tasks. Proceedings of the 26th Annual International Conference on Machine Learning (pp. 137–144).
28. Chen, J., Liu, J., & Ye, J. (2010). Learning incoherent sparse and low-rank patterns from multiple tasks. Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining (pp. 1179–1188).
29. Jacob, L., Bach, F., & Vert, J. (2008). Clustered multi-task learning: A convex formulation. Advances in Neural Information Processing Systems， 2008
30. Zhou, J., Chen, J., & Ye, J. (2011). Clustered multi-task learning via alternating structure optimization. Advances in Neural Information Processing Systems.

External links

- The Biosignals Intelligence Group at UIUC (http://big.cs.uiuc.edu/webpage/cumulativeLearning/cumulativeLearning.html)
- Washington University at St. Louis Depart. of Computer Science (http://www.cse.wustl.edu/~kilian/research/multitasklearning/multitasklearning.html)

Software

- The Multi-Task Learning via Structural Regularization Package (http://www.public.asu.edu/~jye02/Software/MALSAR/index.html)
- Online Multi-Task Learning Toolkit (OMT) (http://klcl.pku.edu.cn/member/sunxu/code.htm) A general-purpose online multi-task learning toolkit based on conditional random field models and stochastic gradient descent training (C#, .NET)

Retrieved from "https://en.wikipedia.org/w/index.php?title=Multi-task_learning&oldid=807764458"

This page was last edited on 30 October 2017, at 00:44.

Text is available under the Creative Commons Attribution-ShareAlike License; additional terms may apply. By using this site, you agree to the Terms of Use and Privacy Policy. Wikipedia® is a registered trademark of the Wikimedia Foundation, Inc., a non-profit organization.