Convex Optimization for Multitask Feature Learning

Priya Venkateshan

Learning Multiple Tasks Simultaneously

- Learning multiple related tasks vs. learning independently.
- Few data per task; pooling data across related tasks.
- Examples:
 - user preferences (movies, products etc.)
 - computer vision (recognizing faces, objects etc.)
 - text classification
 - etc.

Multi-Task Feature Learning

- Assumption: common underlying representation across tasks.
- A *small* set of *shared* features ([Baxter 1995], [Torralba et al. 2004], [Ando & Zhang 2005] etc.).

MULTITASK FEATURE LEARNING

Learning Paradigm

- Tasks $t = 1, \dots, T$.
- m examples per task: $(x_{t1}, y_{t1}), \dots, (x_{tm}, y_{tm}) \in \mathbb{R}^d \times \mathbb{R}$.
- Estimate $f_t : \mathbb{R}^d \rightarrow \mathbb{R}$, $t = 1, \dots, T$.
- Consider features

$$h_1(x), \dots, h_d(x)$$

- Predict using functions

$$f_t(x) = \sum_{i=1}^d a_{it} h_i(x)$$

Weighting Features

- *Feature importance vs. tasks* is described by the matrix

$$A = \begin{pmatrix} a_{11} & \dots & a_{1T} \\ \vdots & \ddots & \vdots \\ a_{d1} & \dots & a_{dT} \end{pmatrix} = \begin{pmatrix} -a^1- \\ \vdots \\ -a^d- \end{pmatrix} = \begin{pmatrix} | & & | \\ a_1 & \dots & a_T \\ | & & | \end{pmatrix}$$

where

$$a^i = (a_{i1}, \dots, a_{iT})$$

$$a_t = \begin{pmatrix} a_{1t} \\ \vdots \\ a_{dt} \end{pmatrix}$$

Sharing Features Across Tasks

- Desiderata:
 1. a *low-dimensional data representation* shared across the tasks
 2. the importance of each feature is *preserved across the tasks*
 3. *convex* formulation

Sharing Features Across Tasks

- In terms of matrix A :
 1. *most a^i should equal zero*
 2. *for each i , the $|a_{it}|$ should be similar*

(2, 1)-Norm

- Approximate desiderata 1, 2 using the norm

$$\|A\|_{2,1} := \sum_{i=1}^d \sqrt{\sum_{t=1}^T a_{it}^2}$$

- First compute the 2-norms of the rows: $\|a^1\|_2, \dots, \|a^d\|_2$
- Then compute the 1-norm of the resulting vector: $\sum_{i=1}^d \|a^i\|_2$.

(2, 1)-Norm

- Want the $(2, 1)$ -norm to be *small*.
- Small 1-norm favors sparsity and small 2-norm favors uniformity.
- Hence, small $(2, 1)$ -norm means
 - many rows a^i are ≈ 0
 - for each i , the $|a_{it}|$ are similar.

(2, 1)-Norm Regularization

$$\min \left\{ \sum_{t=1}^T \sum_{j=1}^m L(y_{tj}, \sum_{i=1}^d a_{it} h_i(x_{tj})) + \gamma \|A\|_{2,1}^2 : A \in \mathbb{R}^{d \times T} \right\}$$

- This is a *convex* problem.
- The number of features in the solution decreases with γ

L_1 Regularization

- For *one task*, this is simply L_1 regularization:

$$\min \left\{ \sum_{j=1}^m L(y_j, \sum_{i=1}^d a_i h_i(x_j)) + \gamma \|a\|_1^2 : a \in \mathbb{R}^d \right\}$$

- $\|a\|_1$ approximates $\#\{\text{nonzero entries of } a\}$.
- Many components of the solution will be ≈ 0 .

Learning the Features

- How about learning the *features* as well?
- Focus on *linear, orthonormal* features

$$h_i(x) = \langle u_i, x \rangle$$

$$\min \left\{ \sum_{t=1}^T \sum_{j=1}^m L(y_{tj}, \langle a_t, U^\top x_{tj} \rangle) + \gamma \|A\|_{2,1}^2 : U^\top U = I, A \in \mathbb{R}^{d \times T} \right\}$$

- *Non-convex, nonsmooth* problem.

Convex Reformulation

- Variable transformation

$$W = \begin{pmatrix} | & & | \\ w_1 & \dots & w_T \\ | & & | \end{pmatrix} = \begin{matrix} U & A \\ d \times d & d \times T \end{matrix}$$

$$D = U \operatorname{Diag} \left(\frac{\|a^i\|_2}{\|A\|_{2,1}} \right) U^\top$$

- Optimal W will be *low-rank*.
- D combines features U and feature weights A .

Convex Reformulation (cont.)

$$\inf \left\{ \sum_{t=1}^T \sum_{j=1}^m L(y_{tj}, \langle w_t, x_{tj} \rangle) + \gamma \sum_{t=1}^T \langle w_t, D^{-1} w_t \rangle \right. \\ \left. : W \in \mathbb{R}^{d \times T}, D \succ 0, \text{trace}(D) \leq 1 \right\}$$

- $\sum_{t=1}^T \langle w_t, D^{-1} w_t \rangle$ induces relations between the tasks.
- *Jointly convex in W and D !*

Alternating Algorithm

- Alternate between W (supervised learning) and D (unsupervised “correlating” of tasks).

Initialization: set $D = \frac{I_{d \times d}}{d}$

while convergence condition is not true **do**

for $t = 1, \dots, T$, learn w_t *independently*

 by minimizing $\sum_{j=1}^m L(y_{tj}, \langle w_t, x_{tj} \rangle) + \gamma \langle w_t, D^+ w_t \rangle$

end for

Find the D that best “relates” the tasks:

$$D = \frac{(W W^\top)^{\frac{1}{2}}}{\text{trace}(W W^\top)^{\frac{1}{2}}} \quad (\text{using SVD})$$

end while

Experiment 1 (toy data)

- $T = 200$ tasks.
- $h_i(x) = x, \quad i = 1, \dots, d.$
- $a_{it} = \begin{cases} \mathcal{N}(0, \sigma_i) & i = 1, \dots, 5 \\ 0 & i = 6, \dots, d \end{cases}$
- 5 training examples per task. Inputs uniformly drawn from $[0, 1]^d$.
- Outputs $y_{tj} = \langle a_t, x_{tj} \rangle + \text{noise}.$

Experiment 1 (toy data)

- Learning multiple tasks together improves performance.
- *Improvement is large*, even when most features are irrelevant.
- More tasks lead to better estimates of the features.

Experiment 2 (real data)

- Consumers' ratings of products [Lenk et al. 1996].
- 180 persons (tasks).
- 8 PC models (training examples); 4 PC models (test examples).
- 13 binary input attributes (RAM, CPU, price etc.).
- Integer output in $\{0, \dots, 10\}$ (likelihood of purchase).

Experiment 2 (real data)

Test error

#tasks

Eig(D)

- Performance improves with more tasks (for independent, error = 16.53).
- A single most important feature shared by all persons.

Experiment 2 (real data)

- The most important feature weighs *technical characteristics* (RAM, CPU, CD-ROM) vs. *price*.

Summary

- Multi-task feature learning
 - *low-dimensional data representation* shared by a pool of tasks
 - feature importance *preserved across tasks*.
- *Convex problem*. Converges to global solution.
- Alternating algorithm.
- Solution is *low-rank*. Algorithm *selects the salient features*. Additional tasks enhance prediction.

MULTITASK FEATURE LEARNING VIA EFFICIENT L2,1 NORM MINIMIZATION

A probabilistic framework for MTL

- k tasks, data of type $\{(a_i^j, y_i^j)\}_{i=1}^{m_j}, j = 1, 2, \dots, k,$
- Data matrix $A_j = [a_1^j, \dots, a_{m_j}^j]^T \in \mathbb{R}^{m_j \times n}$ and $A = [A_1^T, \dots, A_k^T]^T$
- Linear model: $f_j(a) = w_j^T a, j = 1, \dots, k,$
- Weight matrix $W = [w_1, \dots, w_k] \in \mathbb{R}^{n \times k}$ estimated from data

A probabilistic framework for MTLFL

- Assume y has a gaussian distribution with mean $f_j(\mathbf{a}^j)$ and variance $\sigma^j > 0$

- $$p(y^j | \mathbf{w}_j, \mathbf{a}^j, \sigma^j) = \sqrt{\frac{\sigma^j}{2\pi}} \exp\left(\frac{-\sigma^j (y^j - \mathbf{w}_j^T \mathbf{a}^j)^2}{2}\right)$$

- Likelihood:
$$p(y|W, A, \sigma) = \prod_{j=1}^k \prod_{i=1}^{m_j} p(y_i^j | \mathbf{w}_j, \mathbf{a}_i^j, \sigma^j).$$

- Define a prior on W to capture task interrelatedness

$$p(\mathbf{w}^i | \delta^i) \propto \exp(-\|\mathbf{w}^i\| \delta^i), \quad i = 1, 2, \dots, n,$$

$$p(W | \delta) = \prod_{i=1}^n p(\mathbf{w}^i | \delta^i).$$

A probabilistic framework for MTFL

- Posterior : $p(W|A, y, \sigma, \delta) \propto p(y|W, A, \sigma)p(W|\delta)$.
- Plug in value from equations, take negative log of posterior.
- Optimal value of W can be computed by minimizing
$$\frac{1}{2} \sum_{j=1}^k \sigma^j \|y_j - A_j w_j\|^2 + \sum_{i=1}^n \delta^i \|w^i\|.$$
- Equivalent:
$$\min_W \frac{1}{2} \sum_{j=1}^k \|y_j - A_j w_j\|^2 + \rho \|W\|_{2,1},$$
- Generalize:
$$\min_{W \in \mathbb{R}^{n \times k}} \text{loss}(W) + \rho \|W\|_{2,1},$$

Two Smooth Reformulations

- Above optimization is nonsmooth.
- Reformulate it as an equivalent smooth convex problem

First Smooth Reformulation

- Introduce additional variable $\mathbf{t} = [t_1, t_2, \dots, t_n]^T$ which upperbounds $\|\mathbf{w}^i\|$.

Theorem 1. *Let $\text{loss}(W)$ be a smooth convex loss function. Then the problem in Eq. (9) is equivalent to the following constrained smooth convex optimization problem:*

$$\min_{(\mathbf{t}, W) \in D} \text{loss}(W) + \rho \sum_{i=1}^n t_i,$$

where $\mathbf{t} = [t_1, t_2, \dots, t_n]^T$ and

$$D = \{(\mathbf{t}, W) \mid \|\mathbf{w}^i\| \leq t_i, \forall i = 1, 2, \dots, n\}$$

is close and convex.

Second Smooth Reformulation

Theorem 2. *Let $\text{loss}(W)$ be a smooth convex loss function. Then the problem in Eq. (9) is equivalent to the following $\ell_{2,1}$ -ball constrained smooth convex optimization problem:*

$$\min_{W \in Z} \text{loss}(W),$$

where

$$Z = \{W \in \mathbb{R}^{n \times k} \mid \|W\|_{2,1} \leq z\},$$

$z \geq 0$ ¹ is the radius of the $\ell_{2,1}$ -ball, and there is a one-to-one correspondence between ρ and z .

Nesterov's Method

- Converges faster than most traditional methods: $O(1/d^2)$, while gradient descent = $O(1/d)$
- Based on two sequences: $\{x_i\}$ sequence of approximate solutions, $\{s_i\}$ sequence of search points.

Nesterov's Method

- Each sequence point is affine combination of previous solution points $s_i = x_i + \alpha_i(x_i - x_{i-1})$
- Approximate solution is calculated as gradient step of sequence point $x_{i+1} = \pi_G(s_i - \frac{1}{\gamma_i}g'(s_i))$,
where $\pi_G(v) = \min_{x \in G} \frac{1}{2} \|x - v\|^2$, is Euclidean projection of v onto convex set G .

Figure 1: Illustration of Nesterov's method. We set $x_1 = x_0$ and $\alpha_2 = 0$, thus $s_1 = x_1$ and $s_2 = x_2$. The search point s_i is the affine combination of x_{i-1} and x_i (the dashed lines), and the next approximate solution is obtained by $x_{i+1} = \pi_G(s_i - \frac{1}{\gamma_i} g'(s_i))$ (solid lines). We assume that $x_6 = x_7 = x^*$, and x^* is an optimal solution.

Algorithm

Algorithm 1 Nesterov's Method for Constrained Smooth Convex Optimization

Input: $g(\cdot), g'(\cdot), G, L_0 > 0, \mathbf{x}_0$

Output: $\mathbf{x}$

```
1: Initialize  $\mathbf{x}_1 = \mathbf{x}_0, t_{-1} = 0, t_0 = 1, \gamma_0 = L_0$ 
2: for  $i = 1$  to  $\dots$  do
3:   Set  $\alpha_i = \frac{t_{i-2}-1}{t_{i-1}}, \mathbf{s}_i = \mathbf{x}_i + \alpha_i(\mathbf{x}_i - \mathbf{x}_{i-1})$ 
4:   for  $j = 0$  to  $\dots$  do
5:     Set  $\gamma = 2^j \gamma_{i-1}$ 
6:     Compute  $\mathbf{x}_{i+1} = \pi_G(\mathbf{s}_i - \frac{1}{\gamma} g'(\mathbf{s}_i))$ 
7:     if  $g(\mathbf{x}_{i+1}) \leq g_{\gamma, \mathbf{s}_i}(\mathbf{x}_{i+1})$  then
8:        $\gamma_i = \gamma$ , break
9:     end if
10:  end for
11:  Set  $t_i = (1 + \sqrt{1 + 4t_{i-1}^2})/2$ 
12:  if convergence then
13:     $\mathbf{x} = \mathbf{x}_i$ , terminate
14:  end if
15: end for
```

Complexity

- For both smooth formulations, the time complexity of the algorithm turns out to be $O(\frac{1}{\sqrt{\epsilon}}(mn + nk))$, where m = number of samples, n = number of features, k = number of tasks.

Experiments

- Datasets:
 - School dataset
 - Scores of 15,342 students from 139 schools from 1985, 1986 and 1987. 28 attributes in each sample
 - 139 tasks of predicting performance of students in each school
 - Letter dataset
 - 8 tasks of 2-class classification problems for letters by 180 writers. `c/e, g/y, m/n, a/g, i/j, a/o, f/t, h/n`
 - 45,679 samples

Results

Results

γ	0.1	0.01	0.001	0.0001
MTL-FEAT	17.90	18.01	18.02	18.26
aMTFL ₁	0.04	0.13	0.76	2.89
aMTFL ₂	0.06	0.22	0.60	1.32
GD	0.07	0.62	4.60	6.63

Table 1: Comparison of the computational time (in seconds) among MTL-FEAT, the proposed two reformulations, and GD for the $\ell_{2,1}$ -norm regularized least squares regression on the School data set.

m	916	1832	2748	3664	4580
MTL-FEAT	1.11	5.84	16.84	36.79	67.99
aMTFL ₁	0.03	0.03	0.12	0.18	0.32
aMTFL ₂	0.07	0.18	0.36	0.44	0.82
GD	0.41	0.91	1.97	1.98	2.41

Table 2: Comparison of the computational time (in seconds) among MTL-FEAT, the proposed two reformulations, and GD for the $\ell_{2,1}$ -norm regularized least squares regression on the Letter data set.

References

- Multitask Feature Learning - Andreas Argyriou, Theodoros Evgeniou, Massimiliano Pontil, NIPS-06
- Multitask Feature Learning via efficient $L_{2,1}$ Norm Minimization - Jun Liu, Shuiwang Ji, Jieping Ye, UAI-09.
- Slides of first approach from authors' website