



Aalto University  
School of Science  
and Technology

# Newton update in $L_2$ -norm random tree approximation

Hongyu Su

Helsinki Institute for Information Technology HIIT  
Department of Computer Science  
Aalto University

May 22, 2015

# Preliminaries

- ▶  $\mathcal{X}$  is an arbitrary input space,  $\mathbf{x} \in \mathcal{X}$ .
- ▶  $\mathcal{Y}$  is an output space of a set of  $\ell$ -dimensional *multilabels*

$$\mathbf{y} = (y_1, \dots, y_\ell) \in \mathcal{Y}.$$

- ▶  $y_i$  is a *microlabel* and  $y_i \in \{1, \dots, r_i\}$ ,  $r_i \in \mathbb{Z}$ .
- ▶ For example, multilabel binary classification  $y_i \in \{-1, +1\}$ .
- ▶ Training examples are sampled from  $(\mathbf{x}, \mathbf{y}) \in \mathcal{X} \times \mathcal{Y}$ .
- ▶ Each example  $(\mathbf{x}, \mathbf{y})$  is mapped into a joint feature space  $\phi(\mathbf{x}, \mathbf{y})$ .
- ▶  $\mathbf{w}$  is the weight vector in the joint feature space.
- ▶ Define a linear score function  $F(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle$ .
- ▶ The prediction  $\mathbf{y}_{\mathbf{w}}(\mathbf{x})$  of an input  $\mathbf{x}$  is the multilabel  $\mathbf{y}$  that maximizes the score function

$$\mathbf{y}_{\mathbf{w}}(\mathbf{x}) = \underset{\mathbf{y} \in \mathcal{Y}}{\operatorname{argmax}} \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle. \quad (1)$$

- ▶ (1) is called *inference* problem which is  $\mathcal{NP}$ -hard for most output feature maps.

# Markov network

- ▶ We assume that the joint feature map  $\phi$  is a potential function on a Markov network  $G = (E, V)$ .
- ▶  $G$  is a complete graph with  $|V| = \ell$  nodes and  $|E| = \frac{\ell(\ell-1)}{2}$  undirected edges.
- ▶  $G$  models all pairwise correlations.
- ▶  $\varphi(\mathbf{x})$  is the input feature map, e.g., bag-of-words feature of an example  $\mathbf{x}$ .
- ▶  $\psi(\mathbf{y})$  is the output feature map which is a collection of edges and labels

$$\varphi(\mathbf{y}) = (u_e)_{e \in E}, u_e \in \{-1, +1\}^2.$$

- ▶ The joint feature is the Kronecker product of  $\varphi(\mathbf{x})$  and  $\psi(\mathbf{y})$

$$\phi(\mathbf{x}, \mathbf{y}) = (\phi_e(\mathbf{x}, \mathbf{y}))_{e \in E} = (\varphi(\mathbf{x}) \otimes \psi_e(\mathbf{y}_e))_{e \in E}.$$

- ▶ The score function can be factorized by the complete graph  $G$

$$F(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \langle \mathbf{w}, \phi(\mathbf{x}, \mathbf{y}) \rangle = \sum_{e \in E} \langle \mathbf{w}_e, \phi_e(\mathbf{x}, \mathbf{y}_e) \rangle.$$

# Inference in terms of all spanning trees

- ▶ Solving the following inference problem on a complete graph is  $\mathcal{NP}$ -hard

$$\mathbf{y}_{\mathbf{w}}(\mathbf{x}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} F(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \operatorname{argmax}_{\mathbf{y} \in \mathcal{Y}} \sum_{e \in E} \langle \mathbf{w}_e, \phi_e(\mathbf{x}, \mathbf{y}_e) \rangle.$$

- ▶ For a complete graph, there are  $\ell^{\ell-2}$  unique spanning trees.
- ▶ We can write  $F(\mathbf{w}, \mathbf{x}, \mathbf{y})$  as a conic combination of all spanning trees

$$F(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \mathbf{E}_{T \in U(G)} a_T \langle \mathbf{w}_T, \phi_T(\mathbf{x}, \mathbf{y}) \rangle$$
$$\mathbf{E}_{T \in U(G)} a_T^2 = 1, \quad \mathbf{E}_{T \in U(G)} a_T < 1.$$

- ▶  $U(G)$  is the uniform distribution over  $\ell^{\ell-2}$  spanning trees.
- ▶ The number of spanning trees is exponentially dependent on the number of nodes  $\ell$ .

# A sample of $n$ spanning trees

- ▶ Instead of using all spanning trees, we can just use  $n$  spanning trees

$$F_{\mathcal{T}}(\mathbf{w}, \mathbf{x}, \mathbf{y}) = \frac{1}{n} \sum_{i=1}^n a_{T_i} \langle \mathbf{w}_{T_i}, \phi_{T_i}(\mathbf{x}, \mathbf{y}) \rangle$$
$$\frac{1}{n} \sum_{i=1}^n a_{T_i}^2 = 1, \quad \frac{1}{n} \sum_{i=1}^n a_{T_i} < 1.$$

- ▶ When

$$n \geq \frac{\ell^2}{\epsilon^2} \left( \frac{1}{16} + \frac{1}{2} \ln \frac{8\sqrt{n}}{\delta} \right),$$

we have  $|F_{\mathcal{T}}(\mathbf{w}, \mathbf{x}, \mathbf{y}) - F(\mathbf{w}, \mathbf{x}, \mathbf{y})| \leq \epsilon$ , with high probability.

- ▶ A sample of  $n \in \Theta(\ell^2/\delta^2)$  random spanning tree is sufficient to estimate the score function.
- ▶ Margin achieved by  $F(\mathbf{w}, \mathbf{x}, \mathbf{y})$  is also preserved by the sample of  $n$  random spanning trees  $F_{\mathcal{T}}(\mathbf{w}, \mathbf{x}, \mathbf{y})$ .

# Optimization problem

- ▶ The primal optimization problem is defined as

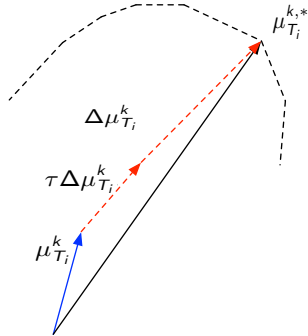
$$\begin{aligned} \min_{\mathbf{w}_{T_i}, \xi_i} \quad & \frac{1}{2} \sum_{i=1}^n \|\mathbf{w}_{T_i}\|^2 + C \sum_{k=1}^m \xi_k \\ \text{s.t.} \quad & \sum_{i=1}^n \langle \mathbf{w}_{T_i}, \Delta \phi_{T_t}(\mathbf{x}_k, \mathbf{y}_k) \rangle \geq \ell_{T_i, k} - \xi_k, \\ & \xi_k \geq 0, \forall k \in \{1, \dots, m\}. \end{aligned}$$

- ▶ The marginalized dual problem is defined as

$$\begin{aligned} \max_{\mu \in \mathcal{M}} \quad & \sum_{i=1}^n \left( \mu_{T_i} \ell_{T_i} - \frac{1}{2} \mu_{T_i} K_{T_i}^{\Delta \phi} \mu_{T_i} \right) \\ \text{s.t.} \quad & \sum_{u_e} \mu_{T_i, e}(u_e) \leq C. \end{aligned}$$

- ▶ How to optimize a collection of  $n$  spanning trees jointly?

# Optimization on a single random spanning tree $T_i$



# Optimization on a single random spanning tree $T_i$

- ▶ The optimization problem can be solved efficiently on a single tree  $T_i$ .
- ▶ The algorithm iterates over all training examples until convergence.
- ▶ We drop the index momentarily  $\mu \leftarrow \mu(j)$ ,  $g \leftarrow g(j)$ ,  $\ell \leftarrow \ell(j)$ .
- ▶ For the  $k$ th iteration:
  1. Obtain the current solution of the  $j$ th example  $\mu_{T_i}^k$ .
  2. Compute the current gradient on  $\mu_{T_i}^k$ ,  $g_{T_i}^k = \ell_{T_i} - K_{T_i}^{\Delta\phi} \mu_{T_i}^k$ .
  3. Compute a feasible solution  $\mu_{T_i}^{k,*}$  as an update direction (**efficiently**)

$$\mu_{T_i}^{k,*} = \underset{\mu \in \mathcal{M}}{\operatorname{argmax}} \mu^\top g_{T_i}^k. \quad (2)$$

4. Compute an update direction from the current solution  $\mu_{T_i}^k$  towards the feasible solution  $\mu_{T_i}^{k,*}$

$$\Delta\mu_{T_i}^k = \mu_{T_i}^{k,*} - \mu_{T_i}^k.$$

5. Compute a stationary point ( $\tau$ ) and perform the update

$$\mu_{T_i}^{k+1} = \mu_{T_i}^k + \tau \Delta\mu_{T_i}^k.$$



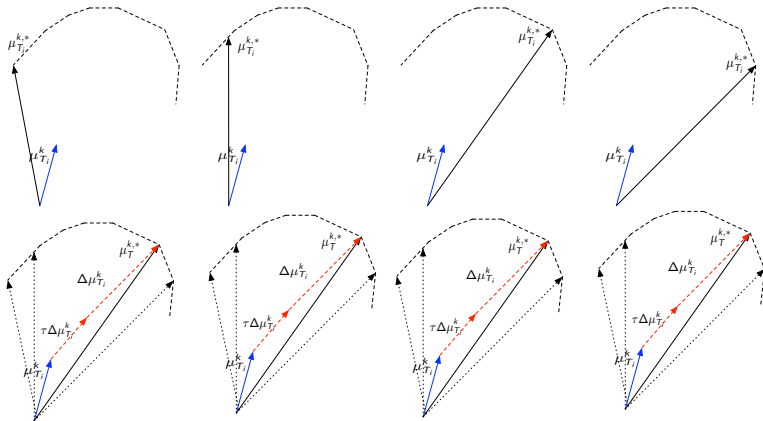
# Exact line search for a single tree

- ▶ Line search gives the optimal feasible solution as a stationary point ( $\tau$ )

$$\begin{aligned} \max_{\tau} \quad & f(\mu_{T_i}^k + \tau \Delta \mu_{T_i}^k) \\ \text{s.t.} \quad & 0 \leq \tau \leq 1. \end{aligned} \tag{3}$$

- ▶  $\tau = 0$  corresponds to no update.
- ▶ Feasible maximum update is achieved at  $\tau = 1$ .
- ▶ The cost of (3) is significantly smaller than the cost of (2).

# Optimization on a collection of $n$ spanning trees



# Optimization on a collection of $n$ spanning trees

- ▶ The algorithm iterates over all training examples until convergence.
- ▶ We drop the index momentarily  $\mu \leftarrow \mu(j)$ ,  $g \leftarrow g(j)$ ,  $\ell \leftarrow \ell(j)$ .
- ▶ For the  $k$ th iteration:

1. Obtain the current solutions over all spanning trees  $(\mu_{T_i}^k)_{i=1}^n$ .
2. Compute the gradients over all trees  $(g_{T_i}^k)_{i=1}^n$ .
3. Compute a feasible solution for each individual spanning tree

$$\mu_{T_i}^{k,*} = \underset{\mu \in \mathcal{M}}{\operatorname{argmax}} \mu^\top g_{T_i}^k, \quad \forall i.$$

4. Compute the best feasible solution over the collection

$$\mu_T^{k,*} = \underset{\mu \in (\mu_{T_i}^{k,*})_{i=1}^n}{\operatorname{argmax}} \sum_{i=1}^n \mu^\top g_{T_i}^k$$

5. Compute the update direction

$$\Delta \mu_{T_i}^k = \mu_{T_i}^k - \mu_T^{k,*}, \quad \forall i.$$

6. Perform the update to the optimal feasible solution

$$\mu_{T_i}^{k+1} = \mu_{T_i}^k + \tau \Delta \mu_{T_i}^k, \quad \forall i.$$

# Exact line search for the collection of trees

- The step size along the update direction  $\tau$  is given by the exact line search

$$\begin{aligned} \max_{\tau} \quad & \sum_{i=1}^n f(\mu_{T_i}^k + \tau \Delta \mu_{T_i}^k) \\ \text{s.t.} \quad & 0 \leq \tau \leq 1. \end{aligned}$$

- Problems with the *best update*

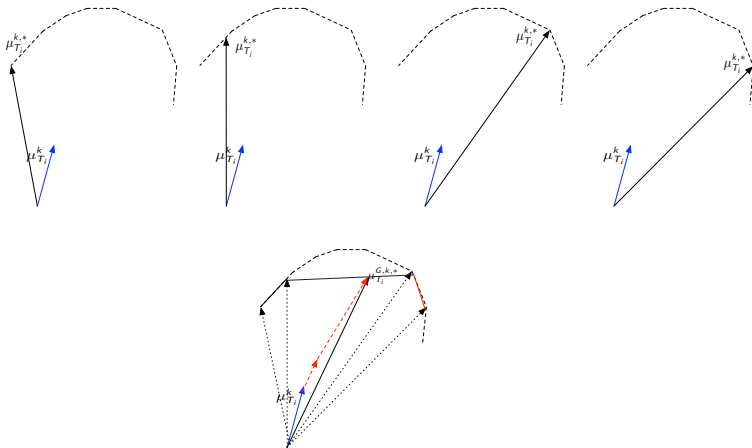
1. The best feasible solution on a single tree might not be the best feasible solution on a collection of trees

$$\mu_T^{k,*} \notin (\mu_{T_i}^{k,*})_{i=1}^n.$$

2.  $\kappa$ -best inference algorithm

$$\begin{aligned} (\mu_{T_i}^{k,*h})_{h=1}^{\kappa} &= \operatorname{argmax}_{\mu \in \mathcal{M}} \mu^T g_{T_i}^k, \quad \forall i \\ \mu_T^{k,*} &\in (\mu_{T_i}^{k,*h})_{i=\{1,\dots,n\}, h \in \{1,\dots,\kappa\}}. \end{aligned}$$

# Update with multiple directions



# Update with multiple directions

- ▶ The algorithm iterates over all training examples until convergence.
- ▶ We drop the index momentarily  $\mu \leftarrow \mu(j)$ ,  $g \leftarrow g(j)$ ,  $\ell \leftarrow \ell(j)$ .
- ▶ For the  $k$ th iteration:
  1. Obtain the current solutions over all spanning trees  $(\mu_{T_i}^k)_{i=1}^n$ .
  2. Compute the gradients over all trees  $(g_{T_i}^k)_{i=1}^n$ .
  3. Compute a feasible solution for each individual spanning tree

$$\mu_{T_i}^{k,*} = \underset{\mu \in \mathcal{M}}{\operatorname{argmax}} \mu^\top g_{T_i}^k, \forall i.$$

4. Project each local feasible solution to a global feasible solution

$$\mu_{T_i}^{G,k,*} \leftarrow \mu_{T_i}^{k,*}, \forall i.$$

5. Define a conic combination of update directions

$$\Delta \mu^{G,k} = \sum_{i=1}^n \tau_i \Delta \mu_{T_i}^{G,k,*} = \sum_{i=1}^n \tau_i \left( \mu^{G,k} - \mu_{T_i}^{G,k,*} \right), 0 \leq \tau_i \leq 1, \forall i.$$

6. Perform the update  $\mu^{G,k+1} = \mu^{G,k} + \Delta \mu^{G,k+1}$ .
7. Project the global solution on spanning trees  $(\mu_{T_i}^{k+1})_{i=1}^n \leftarrow \mu^{G,k+1}$ .

# Newton method to compute $\tau$

- ▶ We want to find  $\tau$  that maximize the objective function given the update

$$\begin{aligned} \max_{\tau} \quad & f(\mu^{G,k} + \Delta\mu^{G,k+1}) \\ \text{s.t.} \quad & 0 \leq \tau_i \leq 1. \end{aligned}$$

- ▶ The objective is quadratic with respect to  $\tau$ .
- ▶ We use Newton method to find  $\tau$  that maximize the objective.
- ▶  $\tau$  is projected into the feasible region.

# Compute duality gap

- ▶ We use duality gap to measure the progress of the optimization.
- ▶ Primal and dual objective function

$$f(\mathbf{w}) = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum_{i=1}^m (\ell_i - \langle \mathbf{w}, \Delta\phi(\mathbf{x}_i, \mathbf{y}_i) \rangle)$$

$$g(\boldsymbol{\alpha}) = \sum_{i=1}^m \alpha_i \ell_i - \frac{1}{2} \sum_{i=1}^m \sum_{j=1}^m \alpha_i K^{\Delta\phi}(\mathbf{x}_i, \mathbf{y}_i; \mathbf{x}_j, \mathbf{y}_j) \alpha_j$$

- ▶  $\max_{\boldsymbol{\alpha}} g(\boldsymbol{\alpha}) \leq \min_{\mathbf{w}} f(\mathbf{w})$ , minimum gap when optimal.
- ▶ Duality gap at  $\boldsymbol{\alpha}^k$

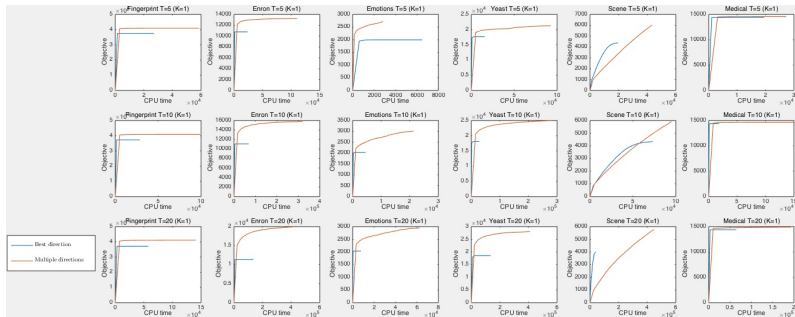
$$\begin{aligned} f(\mathbf{w}^k) - g(\boldsymbol{\alpha}^k) &= C \left( \ell - K^{\Delta\phi} \boldsymbol{\alpha}^k \right) - \boldsymbol{\alpha}^k \left( \ell - K^{\Delta\phi} \boldsymbol{\alpha}^k \right) \\ &= C^T \nabla g(\boldsymbol{\alpha}^k) - \boldsymbol{\alpha}^{kT} \nabla g(\boldsymbol{\alpha}^k) \end{aligned}$$

1. Estimate the dual objective function using a linear approximation  $\nabla g$ .
2. Dual objective value at  $\boldsymbol{\alpha}^k$  is computed by  $\boldsymbol{\alpha}^{kT} \nabla g(\boldsymbol{\alpha}^k)$ .
3. Primal objective value is estimate by  $C^T \nabla g(\boldsymbol{\alpha}^k)$ .



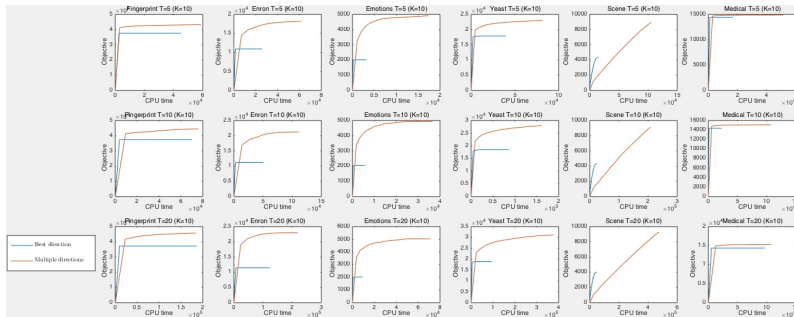
# Experimental results - Objective value

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 1$  best direction.



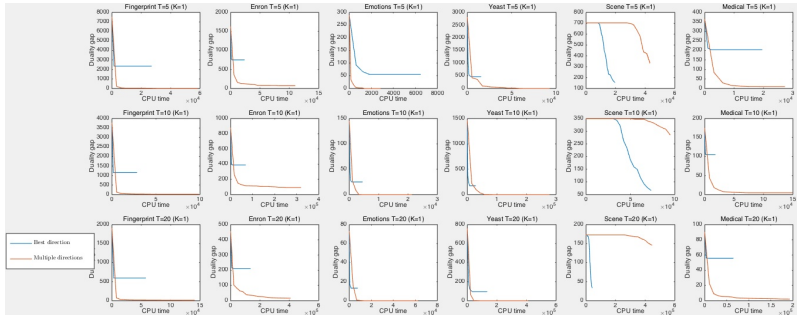
# Experimental results - Objective value

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 10$  best directions.



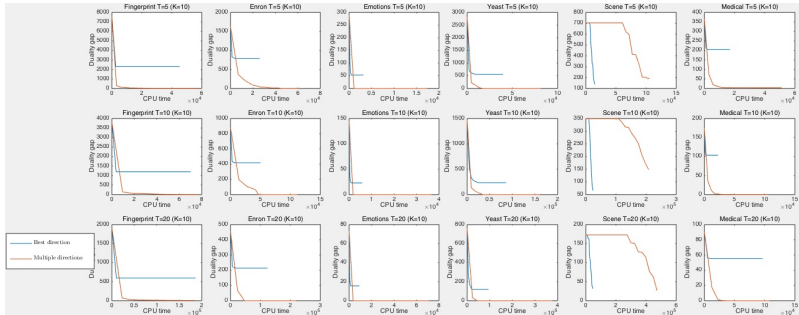
# Experimental results - Duality gap

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 1$  best direction.



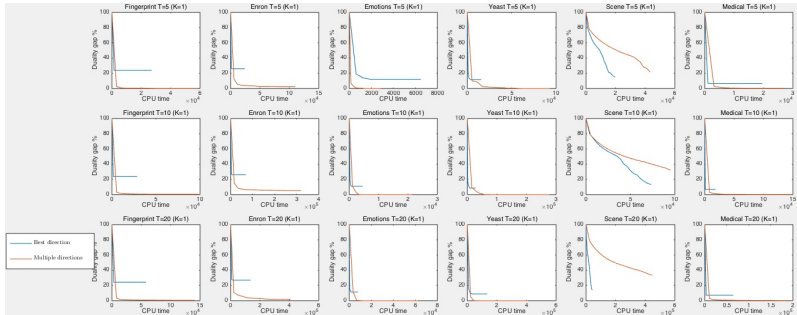
# Experimental results - Duality gap

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 10$  best directions.



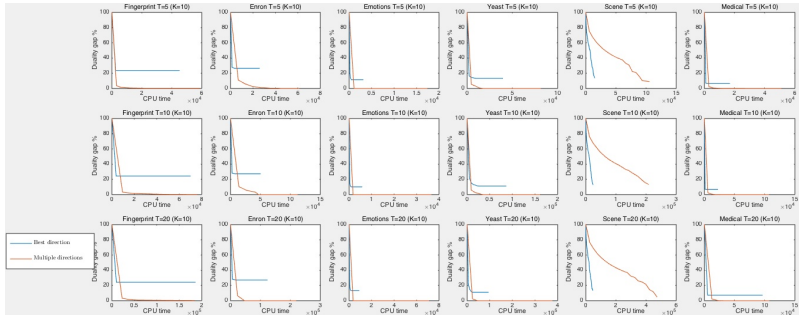
# Experimental results - Relative duality gap

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 1$  best direction.



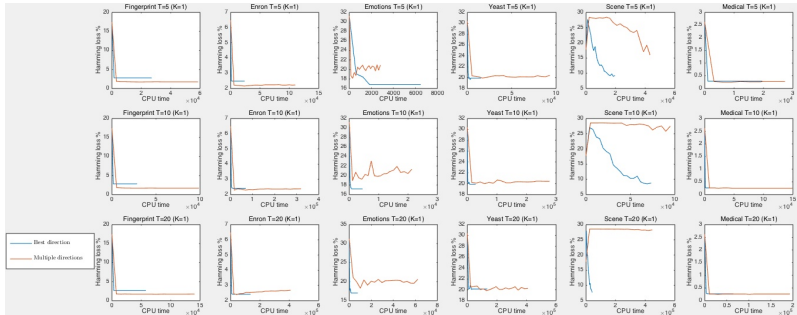
# Experimental results - Relative duality gap

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 10$  best directions.



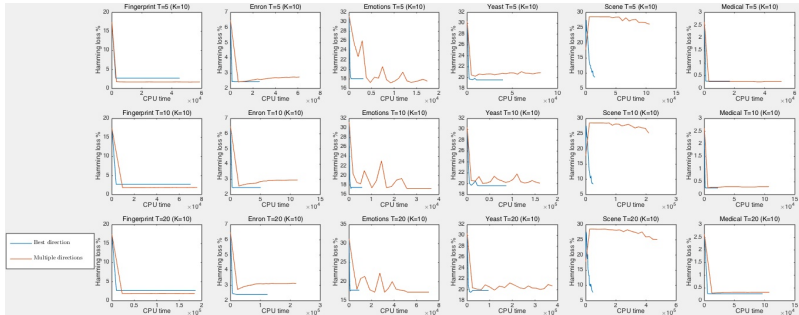
# Experimental results - Hamming loss training

- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 1$  best direction.



# Experimental results - Hamming loss training

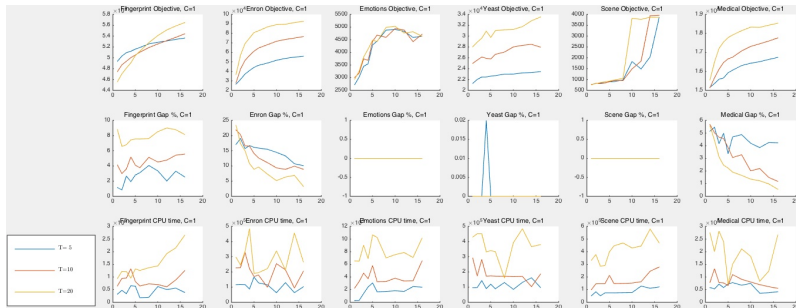
- ▶ Compare update with the best direction v.s. update with multiple directions
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs top  $\kappa = 10$  best directions.





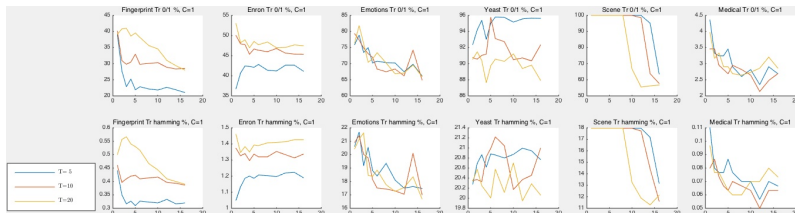
# Experimental results

- ▶ Multiple update.
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs  $\kappa = \{1, \dots, 16\}$  best directions.
- ▶ Objective value, duality gap, CPU time versus iteration.



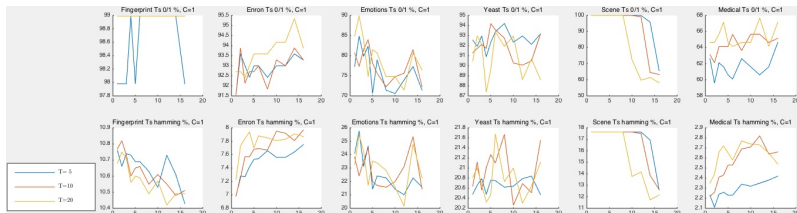
# Experimental results

- ▶ Multiple update.
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs  $\kappa = \{1, \dots, 16\}$  best directions.
- ▶ Training 0/1 loss, hamming loss versus iteration.



# Experimental results

- ▶ Multiple update.
- ▶ Number of spanning trees  $|T| = \{5, 10, 20\}$ .
- ▶ Each spanning tree outputs  $\kappa = \{1, \dots, 16\}$  best directions.
- ▶ Test 0/1 loss, hamming loss versus iteration.



# Conclusions

► *Pros:*

1. Achieves much better objectives, duality gaps.
2. The quality of the optimization increases as more update directions are used.
3. Improves training and test error in many datasets.

► *Cons:*

1. Consumes more CPU times.
2. The improvements are not dramatic compare to the *best update*.