Multi-Task Learning for Improved Discriminative Training in SMT

Patrick Simianer and Stefan Riezler

Department of Computational Linguistics,
Heidelberg University, Germany
Learning from Big Data in SMT

- Machine learning theory and practice suggests benefits from using **expressive feature representations** and from **tuning on large training samples**.
- Discriminative training in SMT has mostly been content with tuning **small sets of dense features** on **small development data** (Och NAACL’03).
- Notable exceptions and recent success stories using **larger feature and training sets**:
  - Liang et al. ACL’06: 1.5M features, 67K parallel sentences.
  - Tillmann and Zhang ACL’06: 35M feats, 230K sents.
  - Blunsom et al. ACL’08: 7.8M feats, 100K sents.
  - Simianer, Riezler, Dyer ACL’12: 4.7M feats, 1.6M sents.
Learning from Big Data in SMT

- Machine learning theory and practice suggests benefits from using **expressive feature representations** and from **tuning on large training samples**.

- Discriminative training in SMT has mostly been content with tuning **small sets of dense features on small development data** (Och NAACL’03).

- Notable exceptions and recent success stories using **larger feature and training sets**:
  - Liang et al. ACL’06: 1.5M features, 67K parallel sentences.
  - Tillmann and Zhang ACL’06: 35M feats, 230K sents.
  - Blunsom et al. ACL’08: 7.8M feats, 100K sents.
  - Simianer, Riezler, Dyer ACL’12: 4.7M feats, 1.6M sents.
Learning from Big Data in SMT

- Machine learning theory and practice suggests benefits from using expressive feature representations and from tuning on large training samples.

- Discriminative training in SMT has mostly been content with tuning small sets of dense features on small development data (Och NAACL’03).

- Notable exceptions and recent success stories using larger feature and training sets:
  - Liang et al. ACL’06: 1.5M features, 67K parallel sentences.
  - Tillmann and Zhang ACL’06: 35M feats, 230K sents.
  - Blunsom et al. ACL’08: 7.8M feats, 100K sents.
  - Simianer, Riezler, Dyer ACL’12: 4.7M feats, 1.6M sents.
Framework: Multi-Task Learning

- **Goal:** A number of statistical models need to be estimated simultaneously from data belonging to different tasks.

- **Examples:**
  - OCR of handwritten characters from different writers: Exploit commonalities on pixel- or stroke-level shared between writers.
  - LTR from search engine query logs from different countries: Some queries are country-specific (“football”), most preference rankings are shared across countries.

- **Idea:**
  - Learn a shared model that takes advantage of commonalities among tasks, without neglecting individual knowledge.
  - Problem of simultaneous learning is harder, but it also offers possibility of knowledge sharing.
Framework: Multi-Task Learning

- **Goal:** A number of statistical models need to be estimated simultaneously from data belonging to different tasks.
- **Examples:**
  - OCR of handwritten characters from different writers: Exploit commonalities on pixel- or stroke-level shared between writers.
  - LTR from search engine query logs from different countries: Some queries are country-specific ("football"), most preference rankings are shared across countries.
- **Idea:**
  - Learn a shared model that takes advantage of commonalities among tasks, without neglecting individual knowledge.
  - Problem of simultaneous learning is harder, but it also offers possibility of knowledge sharing.
Framework: Multi-Task Learning

- **Goal:** A number of statistical models need to be estimated simultaneously from data belonging to different tasks.

- **Examples:**
  - OCR of handwritten characters from different writers: Exploit commonalities on pixel- or stroke-level shared between writers.
  - LTR from search engine query logs from different countries: Some queries are country-specific (“football”), most preference rankings are shared across countries.

- **Idea:**
  - Learn a shared model that takes advantage of commonalities among tasks, without neglecting individual knowledge.
  - Problem of simultaneous learning is harder, but it also offers possibility of knowledge sharing.
Multi-Task Distributed SGD for Discriminative SMT

**Idea:** Take advantage of algorithms designed for hard problems to ease discriminative SMT on big data.

- Distribute work,
- learn efficiently on each example,
- share information.

**Method:**

- **Distributed learning** using Hadoop/MapReduce or Sun Grid Engine.
- **Online learning** via Stochastic Gradient Descent optimization.
- **Feature selection** via $\ell_1/\ell_2$ block norm regularization on features across multiple tasks.
Multi-Task Distributed SGD for Discriminative SMT

**Idea:** Take advantage of algorithms designed for hard problems to ease discriminative SMT on big data.
- Distribute work,
- learn efficiently on each example,
- share information.

**Method:**
- **Distributed learning** using Hadoop/MapReduce or Sun Grid Engine.
- **Online learning** via Stochastic Gradient Descent optimization.
- **Feature selection** via $\ell_1/\ell_2$ block norm regularization on features across multiple tasks.
Related Work

- **Online learning:**
  - We deploy pairwise ranking perceptron (Shen & Joshi JMLR’05)
  - and margin perceptron (Collobert & Bengio ICML’04).

- **Distributed learning:**
  - Without feature selection, our algorithm reduces to Iterative Mixing (McDonald et al. NAACL’10),
  - which itself is related to Bagging (Breiman JMLR’96) if shards are treated as random samples.
Related Work

- **Online learning:**
  - We deploy pairwise ranking perceptron (Shen & Joshi JMLR’05)
  - and margin perceptron (Collobert & Bengio ICML’04).

- **Distributed learning:**
  - Without feature selection, our algorithm reduces to Iterative Mixing (McDonald et al. NAACL’10),
  - which itself is related to Bagging (Breiman JMLR’96) if shards are treated as random samples.
Related Work

• $\ell_1/\ell_2$ regularization:
  • Related to group-Lasso approaches which use mixed norms (Yuan & Lin JRSS’06), hierarchical norms (Zhao et al. Annals Stats’09), structured norms (Martins et al. EMNLP’11).
  • Difference: Norms and proximity operators are applied to groups of features in single regression or classification task – multi-task learning groups features orthogonally by tasks.
  • Closest relation to Obozinski et al. StatComput’10: Our algorithm is weight-based backward feature elimination variant of their gradient-based forward feature selection algorithm.
OL Framework: Pairwise Ranking Perceptron

- Preference pairs \( \mathbf{x}_j = (\mathbf{x}_j^{(1)}, \mathbf{x}_j^{(2)}) \) where \( \mathbf{x}_j^{(1)} \) is ordered above \( \mathbf{x}_j^{(2)} \) w.r.t. sentence-wise BLEU (Nakov et al. COLING'12).

- Hinge loss-type objective

\[
l_j(\mathbf{w}) = (\langle \mathbf{w}, \tilde{\mathbf{x}}_j \rangle)_+ \]

where \( \tilde{\mathbf{x}}_j = \mathbf{x}_j^{(1)} - \mathbf{x}_j^{(2)} \), \((a)_+ = \max(0, a)\), \( \mathbf{w} \in \mathbb{R}^D \) is a weight vector, and \( \langle \cdot, \cdot \rangle \) denotes the standard vector dot product.

- Ranking perceptron by stochastic subgradient descent:

\[
\nabla l_j(\mathbf{w}) = \begin{cases} 
-\tilde{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \tilde{\mathbf{x}}_j \rangle \leq 0, \\
0 & \text{else.}
\end{cases}
\]
OL Framework: Pairwise Ranking Perceptron

- Preference pairs $x_j = (x_j^{(1)}, x_j^{(2)})$ where $x_j^{(1)}$ is ordered above $x_j^{(2)}$ w.r.t. sentence-wise BLEU (Nakov et al. COLING’12).

- Hinge loss-type objective

$$l_j(w) = (−⟨w, \tilde{x}_j⟩)_+$$

where $\tilde{x}_j = x_j^{(1)} - x_j^{(2)}$, $(a)_+ = \max(0, a)$, $w \in \mathbb{R}^D$ is a weight vector, and $⟨·, ·⟩$ denotes the standard vector dot product.

- Ranking perceptron by stochastic subgradient descent:

$$\nabla l_j(w) = \begin{cases} -\tilde{x}_j & \text{if } ⟨w, \tilde{x}_j⟩ \leq 0, \\ 0 & \text{else.} \end{cases}$$
OL Framework: Pairwise Ranking Perceptron

- Preference pairs \( x_j = (x_j^{(1)}, x_j^{(2)}) \) where \( x_j^{(1)} \) is ordered above \( x_j^{(2)} \) w.r.t. sentence-wise BLEU (Nakov et al. COLING’12).
- Hinge loss-type objective

\[
l_j(w) = (- \langle w, \tilde{x}_j \rangle)_+\]

where \( \tilde{x}_j = x_j^{(1)} - x_j^{(2)} \), \( (a)_+ = \max(0, a) \), \( w \in \mathbb{R}^D \) is a weight vector, and \( \langle \cdot, \cdot \rangle \) denotes the standard vector dot product.

- Ranking perceptron by stochastic subgradient descent:

\[
\nabla l_j(w) = \begin{cases} 
-\tilde{x}_j & \text{if } \langle w, \tilde{x}_j \rangle \leq 0, \\
0 & \text{else.}
\end{cases}
\]
OL framework: Margin Perceptron

- Hinge loss-type objective

\[ l_j(w) = (1 - \langle w, \tilde{x}_j \rangle)_+ \]

- Stochastic subgradient descent:

\[ \nabla l_j(w) = \begin{cases} -\tilde{x}_j & \text{if } \langle w, \tilde{x}_j \rangle < 1, \\ 0 & \text{else.} \end{cases} \]

- Margin term controls capacity, but results in more updates.
- Collobert & Bengio (ICML'04) argue that this justifies not using an explicit regularization (as for example in an SGD version of the SVM (Shalev-Shwartz et al. ICML'07)).
OL framework: Margin Perceptron

- Hinge loss-type objective

\[ l_j(w) = (1 - \langle w, \tilde{x}_j \rangle)_+ \]

- Stochastic subgradient descent:

\[ \nabla l_j(w) = \begin{cases} 
-\tilde{x}_j & \text{if } \langle w, \tilde{x}_j \rangle < 1, \\
0 & \text{else.}
\end{cases} \]

- Margin term controls capacity, but results in more updates.
- Collobert & Bengio (ICML’04) argue that this justifies not using an explicit regularization (as for example in an SGD version of the SVM (Shalev-Shwartz et al. ICML’07)).
OL framework: Margin Perceptron

• Hinge loss-type objective

\[ l_j(w) = (1 - \langle w, \tilde{x}_j \rangle)_+ \]

• Stochastic subgradient descent:

\[ \nabla l_j(w) = \begin{cases} -\tilde{x}_j & \text{if } \langle w, \tilde{x}_j \rangle < 1, \\ 0 & \text{else.} \end{cases} \]

• Margin term controls capacity, but results in more updates.

• Collobert & Bengio (ICML’04) argue that this justifies not using an explicit regularization (as for example in an SGD version of the SVM (Shalev-Shwartz et al. ICML’07)).
MTL Framework: $\ell_1/\ell_2$ Block Norm Regularization

- Data points $\{(x_{zn}, y_{zn}), z = 1, \ldots, Z, \ n = 1, \ldots, N_z\}$, sampled from $P_z$ on $X \times Y$ ($z =$ task; $n =$ data point).

- Objective:

$$\min_W \sum_{z,n} l_n(w_z) + \lambda \|W\|_{1,2}$$

- where $W = (w^d_z)_{z,d}$ is a $Z$-by-$D$ matrix $W = (w^d_z)_{z,d}$ of $D$-dimensional row vectors $w_z$ and $Z$-dimensional column vectors $w^d$ of weights associated with feature $d$ across tasks.

- Weighted $\ell_1/\ell_2$ norm:

$$\lambda \|W\|_{1,2} = \lambda \sum_{d=1}^{D} \|w^d\|_2$$

- Each $\ell_2$ norm of a weight column $w^d$ represents the relevance of the corresponding feature across tasks.
MTL Framework: $\ell_1/\ell_2$ Block Norm Regularization

- Data points $\{(x_{zn}, y_{zn}), z = 1, \ldots, Z, n = 1, \ldots, N_z\}$, sampled from $P_z$ on $X \times Y$ ($z =$ task; $n =$ data point).

- Objective:

$$\min_{W} \sum_{z,n} l_n(w_z) + \lambda \|W\|_{1,2}$$

- where $W = (w_{zd})_{z,d}$ is a $Z$-by-$D$ matrix $W = (w_{zd})_{z,d}$ of $D$-dimensional row vectors $w_z$ and $Z$-dimensional column vectors $w^d$ of weights associated with feature $d$ across tasks.

- Weighted $\ell_1/\ell_2$ norm:

$$\lambda \|W\|_{1,2} = \lambda \sum_{d=1}^{D} \|w^d\|_2$$

- Each $\ell_2$ norm of a weight column $w^d$ represents the relevance of the corresponding feature across tasks.
MTL Framework: $\ell_1 / \ell_2$ Block Norm Regularization

- Data points $\{(x_{zn}, y_{zn}), z = 1, \ldots, Z, n = 1, \ldots, N_z\}$, sampled from $P_z$ on $X \times Y$ ($z =$ task; $n =$ data point).
- Objective:

$$\min_W \sum_{z,n} l_n(w_z) + \lambda \| W \|_{1,2}$$

- where $W = (w^d_z)_{z,d}$ is a $Z$-by-$D$ matrix $W = (w^d_z)_{z,d}$ of $D$-dimensional row vectors $w_z$ and $Z$-dimensional column vectors $w^d$ of weights associated with feature $d$ across tasks.
- Weighted $\ell_1 / \ell_2$ norm:

$$\lambda \| W \|_{1,2} = \lambda \sum_{d=1}^{D} \| w^d \|_2$$

- Each $\ell_2$ norm of a weight column $w^d$ represents the relevance of the corresponding feature across tasks.
\(\ell_1/\ell_2\) Regularization Explained

\[
\begin{array}{cccccc}
\text{column } \ell_2 \text{ norm:} & \text{\(w_1\)} & \text{\(w_2\)} & \text{\(w_3\)} & \text{\(w_4\)} & \text{\(w_5\)} \\
6 & 4 & 3 & 2 & 3 & 7 & 5 & 0 & 0 & 0
\end{array}
\]

\[
\ell_1 \text{ sum:} \Rightarrow 18 \Rightarrow 12
\]

- \(\ell_1\) sum of \(\ell_2\) norms encourages several feature columns \(w^d\) to be 0 and others to have high weights across tasks.

- **Algorithm idea:**
  - Contribution to loss reduction must outweigh regularizer penalty in order to activate feature by non-zero weight.
  - Weight-based feature elimination criterion:
    \[
    \text{If } \|w^d\|_2 \leq \lambda, \text{ set } W[z][d] = 0, \forall z.
    \]
  - Implementation by threshold on \(K\) features or by threshold \(\lambda\).
\( \ell_1 / \ell_2 \) Regularization Explained

\[
\begin{align*}
  w_1 & \begin{bmatrix} 6 & 4 & 0 & 0 & 0 \end{bmatrix} & w_1 & \begin{bmatrix} 6 & 4 & 0 & 0 & 0 \end{bmatrix} \\
  w_2 & \begin{bmatrix} 0 & 0 & 3 & 0 & 0 \end{bmatrix} & w_2 & \begin{bmatrix} 3 & 0 & 0 & 0 & 0 \end{bmatrix} \\
  w_3 & \begin{bmatrix} 0 & 0 & 0 & 2 & 3 \end{bmatrix} & w_3 & \begin{bmatrix} 2 & 3 & 0 & 0 & 0 \end{bmatrix}
\end{align*}
\]

column \( \ell_2 \) norm: \( 6 \, 4 \, 3 \, 2 \, 3 \) \( 7 \, 5 \, 0 \, 0 \, 0 \)
\( \ell_1 \) sum: \( \Rightarrow 18 \) \( \Rightarrow 12 \)

- \( \ell_1 \) sum of \( \ell_2 \) norms encourages several feature columns \( w^d \) to be \( 0 \) and others to have high weights across tasks.

- **Algorithm idea:**
  - Contribution to loss reduction must outweigh regularizer penalty in order to activate feature by non-zero weight.
  - Weight-based feature elimination criterion:
    
    \[
    \text{If } \| w^d \|_2 \leq \lambda, \text{ set } W[z][d] = 0, \forall z. \]

- Implementation by threshold on \( K \) features or by threshold \( \lambda \).
Implementation as Feature Selection Algorithm

Algorithm 1 Multi-task Distributed SGD

Get data for $Z$ tasks, each including $S$ sentences; distribute to machines.
Initialize $v \leftarrow 0$.

for epochs $t \leftarrow 0 \ldots T - 1$: do

for all tasks $z \in \{1 \ldots Z\}$: parallel do

$w_{z,t,0,0} \leftarrow v$

for all sentences $i \in \{0 \ldots S - 1\}$: do

Decode $i^{th}$ input with $w_{z,t,i,0}$.

for all pairs $j \in \{0 \ldots P - 1\}$: do

$w_{z,t,i,j+1} \leftarrow w_{z,t,i,j} - \eta \nabla l_j(w_{z,t,i,j})$

end for

$w_{z,t,i,P} \leftarrow w_{z,t,i,P}$

end for

end for

Stack weights $W \leftarrow [w_{1,t,s,0} \ldots w_{Z,t,s,0}]^T$

Select top $K$ feature columns of $W$ by $\ell_2$ norm

for $k \leftarrow 1 \ldots K$ do

$v[k] = \frac{1}{Z} \sum_{z=1}^{Z} W[z][k]$

end for

end for

return $v$
Experiments: Random vs. Natural Tasks

- **Research Question:**
  - As shown in earlier work (Simianer, Riezler, Dyer ACL’12), multi-task learning can be used as general regularization technique on *random shards*.
  
  - Can multi-task learning benefit from *natural task structure* in the data, where shared and individual knowledge is properly balanced?
Experiments: Random vs. Natural Tasks

- Research Question:
  - As shown in earlier work (Simianer, Riezler, Dyer ACL’12), multi-task learning can be used as general regularization technique on random shards.
  - Can multi-task learning benefit from natural task structure in the data, where shared and individual knowledge is properly balanced?
International Patent Classification (IPC) categorizes patents hierarchically into eight sections, 120 classes, 600 subclasses, down to 70,000 subgroups at the leaf level.

Typically, a patent belongs to more than one section, with one section chosen as main classification.

Eight top classes/sections used to define natural tasks.
SMT Setup

(1) \( X \rightarrow X_1 \text{ hat } X_2 \text{ versprochen} \); \( X_1 \text{ promised } X_2 \)
(2) \( X \rightarrow X_1 \text{ hat mir } X_2 \text{ versprochen} \);
\hspace{1cm} X_1 \text{ promised me } X_2
(3) \( X \rightarrow X_1 \text{ versprach } X_2 \); \( X_1 \text{ promised } X_2 \)

- Hierarchical phrase-based translation (Chiang CL’07), formalizes translation rules as productions of synchronous context-free grammar (SCFG).
- Features in discriminative training:
  - **Rule identifiers** for SCFG productions
    Examples: rule (1), (2) and (3)
  - **Rule n-gram** features in source and target
    Examples: “\( X \text{ hat} \)”, “\( \text{hat } X \)”,”\( X \text{ versprochen} \)”
  - **Rule shape** features
    Examples: (\( NT, \text{term}^*, NT, \text{term}^*; NT, \text{term}^*, NT \)) for (1), (2);
    (\( NT, \text{term}^*, NT; NT, \text{term}^*, NT \)) for rule (3).
SMT Setup

(1) \( X \rightarrow X_1 \) hat \( X_2 \) versprochen; \( X_1 \) promised \( X_2 \)
(2) \( X \rightarrow X_1 \) hat mir \( X_2 \) versprochen;
    \( X_1 \) promised me \( X_2 \)
(3) \( X \rightarrow X_1 \) versprach \( X_2 \); \( X_1 \) promised \( X_2 \)

- Hierarchical phrase-based translation (Chiang CL’07), formalizes translation rules as productions of synchronous context-free grammar (SCFG).
- Features in discriminative training:
  - **Rule identifiers** for SCFG productions
    Examples: rule (1), (2) and (3)
  - **Rule n-gram** features in source and target
    Examples: “\( X \) hat”, “hat \( X \)”, “\( X \) versprochen”
  - **Rule shape** features
    Examples: (\( NT, \) term**, \( NT, \) term**; \( NT, \) term**, \( NT \)) for (1), (2);
    (\( NT, \) term**, \( NT; \) \( NT, \) term**, \( NT \)) for rule (3).
MERT Baseline w/ 12 Dense Features

<table>
<thead>
<tr>
<th></th>
<th>single-task tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>indep. 0 pooled 1 pooled-cat 2</td>
</tr>
<tr>
<td>pooled test</td>
<td>– 51.18 51.22</td>
</tr>
<tr>
<td>A</td>
<td>54.92 0255.27 055.17</td>
</tr>
<tr>
<td>B</td>
<td>51.53 51.48 0151.69</td>
</tr>
<tr>
<td>C</td>
<td>1256.31 255.90 55.74</td>
</tr>
<tr>
<td>D</td>
<td>49.94 050.33 050.26</td>
</tr>
<tr>
<td>E</td>
<td>149.19 48.97 149.13</td>
</tr>
<tr>
<td>F</td>
<td>1251.26 51.02 51.12</td>
</tr>
<tr>
<td>G</td>
<td>149.61 49.44 49.55</td>
</tr>
<tr>
<td>H</td>
<td>49.38 49.50 0149.67</td>
</tr>
<tr>
<td>average test</td>
<td>51.52 51.49 51.54</td>
</tr>
</tbody>
</table>

- Neither tuning on pooled or pooled-cat improves over indep..
- \( x \subset \{0, 1, 2\} \) BLEU denotes statistical significance of pairwise test.
- Tuning was repeated 3 times and BLEU scores averaged.
### Single-Task Perceptron w/ $\ell_1$ Regularization

<table>
<thead>
<tr>
<th>single-task tuning</th>
<th>indep. $^0$</th>
<th>pooled $^1$</th>
<th>pooled-cat $^2$</th>
</tr>
</thead>
<tbody>
<tr>
<td>pooled test</td>
<td>–</td>
<td>50.75</td>
<td>$^1$ 52.08</td>
</tr>
<tr>
<td>A</td>
<td>$^1$ 55.11</td>
<td>54.32</td>
<td>$^01$ 55.94</td>
</tr>
<tr>
<td>B</td>
<td>$^1$ 52.61</td>
<td>50.84</td>
<td>$^1$ 52.57</td>
</tr>
<tr>
<td>C</td>
<td>56.18</td>
<td>56.11</td>
<td>$^01$ 56.75</td>
</tr>
<tr>
<td>D</td>
<td>$^1$ 50.68</td>
<td>49.48</td>
<td>$^01$ 51.22</td>
</tr>
<tr>
<td>E</td>
<td>$^1$ 50.27</td>
<td>48.69</td>
<td>$^1$ 50.01</td>
</tr>
<tr>
<td>F</td>
<td>$^1$ 51.68</td>
<td>50.71</td>
<td>$^1$ 51.95</td>
</tr>
<tr>
<td>G</td>
<td>$^1$ 49.90</td>
<td>49.06</td>
<td>$^01$ 50.51</td>
</tr>
<tr>
<td>H</td>
<td>$^1$ 50.48</td>
<td>49.16</td>
<td>$^1$ 50.53</td>
</tr>
<tr>
<td><strong>average test</strong></td>
<td><strong>52.11</strong></td>
<td><strong>51.05</strong></td>
<td><strong>52.44</strong></td>
</tr>
<tr>
<td>model size</td>
<td>430,092.5</td>
<td>457,428</td>
<td>1,574,259</td>
</tr>
</tbody>
</table>

- Improvements over MERT, mostly on pooled-cat tuning set.
- 1.5M features make serial tuning on pooled-cat infeasible.
- Overfitting effect on small pooled data.
## Single- and Multi-Task Perceptron

<table>
<thead>
<tr>
<th></th>
<th>single-task tuning</th>
<th>multi-task tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>indep. 0</td>
<td>pooled 1</td>
</tr>
<tr>
<td>pooled test</td>
<td>–</td>
<td>51.33</td>
</tr>
<tr>
<td>A</td>
<td>54.79</td>
<td>54.76</td>
</tr>
<tr>
<td>B</td>
<td>12 52.45</td>
<td>51.30</td>
</tr>
<tr>
<td>C</td>
<td>2 56.62</td>
<td>56.65</td>
</tr>
<tr>
<td>D</td>
<td>1 50.75</td>
<td>49.88</td>
</tr>
<tr>
<td>E</td>
<td>1 49.70</td>
<td>49.23</td>
</tr>
<tr>
<td>F</td>
<td>1 51.60</td>
<td>51.09</td>
</tr>
<tr>
<td>G</td>
<td>1 49.50</td>
<td>49.06</td>
</tr>
<tr>
<td>H</td>
<td>1 49.77</td>
<td>49.50</td>
</tr>
<tr>
<td>average test</td>
<td>51.90</td>
<td>51.42</td>
</tr>
<tr>
<td>model size</td>
<td>366,869.4</td>
<td>448,359</td>
</tr>
</tbody>
</table>

- Multi-task tuning improves BLEU over all single-task runs.
- Also more efficient due to iterative feature selection.
- Difference between natural and random tasks inconclusive.
### Single- and Multi-Task Margin Perceptron

<table>
<thead>
<tr>
<th></th>
<th>single-task tuning</th>
<th>multi-task tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>indep. 0</td>
<td>pooled 1</td>
</tr>
<tr>
<td>pooled test</td>
<td>–</td>
<td>51.33</td>
</tr>
<tr>
<td>A</td>
<td>1 56.09</td>
<td>55.33</td>
</tr>
<tr>
<td>B</td>
<td>1 52.45</td>
<td>51.59</td>
</tr>
<tr>
<td>C</td>
<td>1 57.20</td>
<td>56.85</td>
</tr>
<tr>
<td>D</td>
<td>1 50.51</td>
<td>50.18</td>
</tr>
<tr>
<td>E</td>
<td>1 50.27</td>
<td>49.36</td>
</tr>
<tr>
<td>F</td>
<td>1 52.06</td>
<td>51.20</td>
</tr>
<tr>
<td>G</td>
<td>1 50.00</td>
<td>49.58</td>
</tr>
<tr>
<td>H</td>
<td>1 50.57</td>
<td>49.80</td>
</tr>
<tr>
<td><strong>average test</strong></td>
<td>52.39</td>
<td>51.74</td>
</tr>
<tr>
<td>model size</td>
<td>423,731.5</td>
<td>484,483</td>
</tr>
</tbody>
</table>

- Single-task runs beat standard perceptron w/ and w/o $\ell_1$.
- Regularization by margin and multi-task learning adds up.
- Best result is nearly 2 BLEU points better than MERT.
Conclusion

- Multi-task learning for SMT is **efficient** due to online learning, parallelization and feature selection,
- but also **effective** in terms of BLEU improvements over single-task learning.
- Multi-task distributed learning is **easy to implement as wrapper** around perceptron.
Future Work: Task Adaption

- *Natural* tasks are slightly advantageous over *random* tasks.
- Goal: Adapt task definition to SMT problem.
  - Explore various similarity metrics on IPC subclasses,
  - jointly optimize task partitioning and SMT performance.
Future Work: Adaptive Regularization

**Algorithm 2** Path-Following Multi-task Distributed SGD

- Get data for $Z$ tasks, each including $S$ sentences; distribute to machines.
- Initialize $v \leftarrow 0$; $\lambda_0$, $\lambda_{\text{min}}$, $\epsilon$.
- for epochs $t \leftarrow 0 \ldots T - 1$: do
  - for all tasks $z \in \{1 \ldots Z\}$: parallel do
    - Perform task-specific learning
  - end for
- Stack weights $W \leftarrow [w_{1,t,s,0} \mid \ldots \mid w_{Z,t,s,0}]^T$
- for feature columns $d \in \{1 \ldots D\}$ in $W$: do
  - if $\|w^d\|_2 \leq \lambda_t$ then
    - $v[d] = 0$
  - else
    - $v[d] = \frac{1}{Z} \sum_{z=1}^{Z} W[z][d]$
  - end if
- end for
- Set $\lambda_{t+1} = \min\{\lambda_t, \frac{\sum_{z,i,j}(l_{z,i,j}(v_{t-1}) - l_{z,i,j}(v_t))}{\epsilon}\}$
- if $\lambda_{t+1} < \lambda_{\text{min}}$ then
  - break
- end if
- end for
- return $v$
Thanks for your attention!

dtrain codebase is part of cdec:
https://github.com/redpony/cdec.