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.
  - Flanigan, Dyer, Carbonell NAACL'13: 28.8M feats, 1M 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.
  - Flanigan, Dyer, Carbonell NAACL'13: 28.8M feats, 1M 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.
  - Flanigan, Dyer, Carbonell NAACL'13: 28.8M feats, 1M 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 ℓ_1/ℓ_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 ℓ_1/ℓ_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

- ℓ_1/ℓ_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

$$l_j(\mathbf{w}) = (-\langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

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

- **Ranking perceptron** by stochastic subgradient descent:

$$\nabla l_j(\mathbf{w}) = \begin{cases} -\bar{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle \leq 0, \\ 0 & \text{else.} \end{cases}$$

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

$$l_j(\mathbf{w}) = (- \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

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

- **Ranking perceptron** by stochastic subgradient descent:

$$\nabla l_j(\mathbf{w}) = \begin{cases} -\bar{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle \leq 0, \\ 0 & \text{else.} \end{cases}$$

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

$$l_j(\mathbf{w}) = (-\langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

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

- **Ranking perceptron** by stochastic subgradient descent:

$$\nabla l_j(\mathbf{w}) = \begin{cases} -\bar{\mathbf{x}}_j & \text{if } \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle \leq 0, \\ 0 & \text{else.} \end{cases}$$

OL framework: Margin Perceptron

- Hinge loss-type objective

$$l_j(\mathbf{w}) = (1 - \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

- Stochastic subgradient descent:

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

- 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

$$l_j(\mathbf{w}) = (1 - \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

- Stochastic subgradient descent:

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

- 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

$$l_j(\mathbf{w}) = (1 - \langle \mathbf{w}, \bar{\mathbf{x}}_j \rangle)_+$$

- Stochastic subgradient descent:

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

- 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: ℓ_1/ℓ_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:

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

  - where $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ is a Z-by-D matrix $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ of D-dimensional row vectors $\mathbf{w}_z$ and Z-dimensional column vectors $\mathbf{w}^d$ of weights associated with feature d across tasks.
- Weighted ℓ_1/ℓ_2 norm:

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

  - Each ℓ_2 norm of a weight column $\mathbf{w}^d$ represents the relevance of the corresponding feature across tasks.

MTL Framework: ℓ_1/ℓ_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:

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

  - where $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ is a Z-by-D matrix $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ of D-dimensional row vectors $\mathbf{w}_z$ and Z-dimensional column vectors $\mathbf{w}^d$ of weights associated with feature d across tasks.
- Weighted ℓ_1/ℓ_2 norm:

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

  - Each ℓ_2 norm of a weight column $\mathbf{w}^d$ represents the relevance of the corresponding feature across tasks.

MTL Framework: ℓ_1/ℓ_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:

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

  - where $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ is a Z-by-D matrix $\mathbf{W} = (\mathbf{w}_z^d)_{z,d}$ of D-dimensional row vectors $\mathbf{w}_z$ and Z-dimensional column vectors $\mathbf{w}^d$ of weights associated with feature d across tasks.
- Weighted ℓ_1/ℓ_2 norm:

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

  - Each ℓ_2 norm of a weight column $\mathbf{w}^d$ represents the relevance of the corresponding feature across tasks.

ℓ_1/ℓ_2 Regularization Explained

	$\mathbf{w}^1$	$\mathbf{w}^2$	$\mathbf{w}^3$	$\mathbf{w}^4$	$\mathbf{w}^5$		$\mathbf{w}^1$	$\mathbf{w}^2$	$\mathbf{w}^3$	$\mathbf{w}^4$	$\mathbf{w}^5$	
$\mathbf{w}_{z_1}$	6	4	0	0	0		6	4	0	0	0	
$\mathbf{w}_{z_2}$	0	0	3	0	0		3	0	0	0	0	
$\mathbf{w}_{z_3}$	0	0	0	2	3		2	3	0	0	0	
column ℓ_2 norm:	6	4	3	2	3		7	5	0	0	0	
ℓ_1 sum:					$\Rightarrow$	18					$\Rightarrow$	12

- ℓ_1 sum of ℓ_2 norms encourages several feature columns $\mathbf{w}^d$ 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:

$$\text{If } \|\mathbf{w}^d\|_2 \leq \lambda, \text{ set } \mathbf{W}[z][d] = 0, \forall z.$$

  - Implementation by threshold on K features or by threshold λ.

ℓ_1/ℓ_2 Regularization Explained

$$
\begin{array}{r|c}
\begin{array}{rcccccc}
 & \mathbf{w}^1 & \mathbf{w}^2 & \mathbf{w}^3 & \mathbf{w}^4 & \mathbf{w}^5 & \\
\mathbf{w}_{z_1} & [\ 6 & 4 & 0 & 0 & 0\ ] & \\
\mathbf{w}_{z_2} & [\ 0 & 0 & 3 & 0 & 0\ ] & \\
\mathbf{w}_{z_3} & [\ 0 & 0 & 0 & 2 & 3\ ] & \\
\text{column } \ell_2 \text{ norm:} & 6 & 4 & 3 & 2 & 3 & \\
\ell_1 \text{ sum:} & & & & & \Rightarrow & 18
\end{array}
&
\begin{array}{cccccc}
\mathbf{w}^1 & \mathbf{w}^2 & \mathbf{w}^3 & \mathbf{w}^4 & \mathbf{w}^5 & \\
[\ 6 & 4 & 0 & 0 & 0\ ] & \\
[\ 3 & 0 & 0 & 0 & 0\ ] & \\
[\ 2 & 3 & 0 & 0 & 0\ ] & \\
7 & 5 & 0 & 0 & 0 & \\
 & & & & \Rightarrow & 12
\end{array}
\end{array}
$$

- ℓ_1 sum of ℓ_2 norms encourages several feature columns $\mathbf{w}^d$ 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:

$$\text{If } \|\mathbf{w}^d\|_2 \leq \lambda, \text{ set } \mathbf{W}[z][d] = 0, \forall z.$$

  - Implementation by threshold on K features or by threshold λ.

Implementation as Feature Selection Algorithm

Algorithm 1 Multi-task Distributed SGD

Get data for Z tasks, each including S sentences;
distribute to machines.
Initialize $\mathbf{v} \leftarrow \mathbf{0}$.
for epochs $t \leftarrow 0 \ldots T-1$: **do**
  for all tasks $z \in \{1 \ldots Z\}$: **parallel do**
    $\mathbf{w}_{z,t,0,0} \leftarrow \mathbf{v}$
    for all sentences $i \in \{0 \ldots S-1\}$: **do**
      Decode i^{th} input with $\mathbf{w}_{z,t,i,0}$.
      for all pairs $j \in \{0 \ldots P-1\}$: **do**
        $\mathbf{w}_{z,t,i,j+1} \leftarrow \mathbf{w}_{z,t,i,j} - \eta \nabla l_j(\mathbf{w}_{z,t,i,j})$
      end for
      $\mathbf{w}_{z,t,i+1,0} \leftarrow \mathbf{w}_{z,t,i,P}$
    end for
  end for
  Stack weights $\mathbf{W} \leftarrow [\mathbf{w}_{1,t,S,0}| \ldots |\mathbf{w}_{Z,t,S,0}]^T$
  Select top K feature columns of $\mathbf{W}$ by ℓ_2 norm
  for $k \leftarrow 1 \ldots K$ **do**
    $$\mathbf{v}[k] = \frac{1}{Z} \sum_{z=1}^{Z} \mathbf{W}[z][k]$$
  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?

Data

A	Human Necessities
B	Performing Operations, Transporting
C	Chemistry, Metallurgy
D	Textiles, Paper
E	Fixed Constructions
F	Mechanical Engineering, Lighting, Heating, Weapons
G	Physics
H	Electricity

- 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

$$
\begin{aligned}
&(1)\ X \rightarrow\ X_1 \text{ hat } X_2 \text{ versprochen};\ X_1 \text{ promised } X_2\\
&(2)\ X \rightarrow\ X_1 \text{ hat mir } X_2 \text{ versprochen};\\
&\qquad\qquad X_1 \text{ promised me } X_2\\
&(3)\ X \rightarrow\ X_1 \text{ versprach } X_2;\ X_1 \text{ promised } X_2
\end{aligned}
$$

- 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).

SMT Setup

$$
\begin{aligned}
&(1)\ X \rightarrow\ X_1 \text{ hat } X_2 \text{ versprochen};\ X_1 \text{ promised } X_2\\
&(2)\ X \rightarrow\ X_1 \text{ hat mir } X_2 \text{ versprochen};\\
&\qquad\qquad X_1 \text{ promised me } X_2\\
&(3)\ X \rightarrow\ X_1 \text{ versprach } X_2;\ X_1 \text{ promised } X_2
\end{aligned}
$$

- 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

	single-task tuning		
	indep. ^{0}	pooled ^{1}	pooled-cat ^{2}
pooled test	–	51.18	51.22
A	54.92	^{02}55.27	^{0}55.17
B	51.53	51.48	^{01}51.69
C	^{12}56.31	^{2}55.90	55.74
D	49.94	^{0}50.33	^{0}50.26
E	^{1}49.19	48.97	^{1}49.13
F	^{12}51.26	51.02	51.12
G	^{1}49.61	49.44	49.55
H	49.38	49.50	^{01}49.67
average test	**51.52**	**51.49**	**51.54**

- 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/ ℓ_1 Regularization

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

- 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

	single-task tuning			multi-task tuning		
	indep. [0]	pooled [1]	pooled-cat [2]	IPC [3]	sharding [4]	resharding [5]
pooled test	–	51.33	1 51.77	[12] 52.56	[12] 52.54	[12] 52.60
A	54.79	54.76	[01] 55.31	[012] 56.35	[012] 56.22	[012] 56.21
B	[12] 52.45	51.30	[1] 52.19	[012] 52.78	[0123] 52.98	[012] 52.96
C	[2] 56.62	56.65	[1] 56.12	[01245] 57.76	[012] 57.30	[012] 57.44
D	[1] 50.75	49.88	[1] 50.63	[01245] 51.54	[012] 51.33	[012] 51.20
E	[1] 49.70	49.23	[01] 49.92	[012] 50.51	[012] 50.52	[012] 50.38
F	[1] 51.60	51.09	[1] 51.71	[012] 52.28	[012] 52.43	[012] 52.32
G	[1] 49.50	49.06	[01] 49.97	[012] 50.84	[012] 50.88	[012] 50.74
H	[1] 49.77	49.50	[01] 50.64	[012] 51.16	[012] 51.07	[012] 51.10
average test	**51.90**	**51.42**	**52.06**	**52.90**	**52.84**	**52.79**
model size	366,869.4	448,359	1,478,049	100,000	100,000	100,000

- 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

	single-task tuning			multi-task tuning		
	indep. ^{0}	pooled ^{1}	pooled-cat ^{2}	IPC ^{3}	sharding ^{4}	resharding ^{5}
pooled test	–	51.33	^{1} 52.58	^{12} 52.98	^{12} 52.95	^{12} 52.99
A	^{1} 56.09	55.33	^{1} 55.92	^{01245} 56.78	^{012} 56.62	^{012} 56.53
B	^{1} 52.45	51.59	^{1} 52.44	^{012} 53.31	^{012} 53.35	^{012} 53.21
C	^{1} 57.20	56.85	^{01} 57.54	^{01} 57.46	^{1} 57.42	^{1} 57.43
D	^{1} 50.51	50.18	^{01} 51.38	^{01245} 52.14	^{0125} 51.82	^{012} 51.66
E	^{1} 50.27	49.36	^{01} 50.72	^{0124} 51.13	^{012} 50.89	^{012} 51.02
F	^{1} 52.06	51.20	^{01} 52.61	^{01245} 53.07	^{012} 52.80	^{012} 52.87
G	^{1} 50.00	49.58	^{01} 50.90	^{01245} 51.36	^{012} 51.19	^{012} 51.11
H	^{1} 50.57	49.80	^{01} 51.32	^{012} 51.57	^{012} 51.62	^{01} 51.47
average test	**52.39**	**51.74**	**52.85**	**53.35**	**53.21**	**53.16**
model size	423,731.5	484,483	1,697,398	100,000	100,000	100,000

- Single-task runs beat standard perceptron w/ and w/o ℓ_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 $\mathbf{v} \leftarrow \mathbf{0}$; λ_0, λ_{min}, ϵ.
for epochs $t \leftarrow 0 \dots T-1$: **do**
  for all **tasks** $z \in \{1 \dots Z\}$: **parallel do**
    Perform task-specific learning
  end for
  Stack weights $\mathbf{W} \leftarrow [\mathbf{w}_{1,t,S,0}| \dots |\mathbf{w}_{Z,t,S,0}]^T$
  for feature columns $d \in \{1 \dots D\}$ in **W**: **do**
    if $\|\mathbf{w}^d\|_2 \leq \lambda_t$ **then**
      $\mathbf{v}[d] = 0$
    else
      $\mathbf{v}[d] = \frac{1}{Z} \sum_{z=1}^{Z} \mathbf{W}[z][d]$
    end if
  end for
  Set $\lambda_{t+1} = \min\{\lambda_t, \frac{\sum_{z,i,j}(l_{z,i,j}(\mathbf{v}_{t-1}) - l_{z,i,j}(\mathbf{v}_t))}{\epsilon}\}$
  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`.