Fast and Adaptive Online Training of Feature-Rich Translation Models

Spence Green **Sida Wang**

Daniel Cer **Christopher D. Manning**

Stanford University

ACL 2013

Feature-Rich Research	**Industry/Evaluations**
Liang et al. 2006	
Tillmann and Zhang 2006	
Arun and Koehn 2007	
Ittycheriah and Roukos 2007	
Watanabe et al. 2007	
Chiang et al. 2008; Chiang et al. 2009	*n*-best/lattice MERT
Haddow et al. 2011	
Hopkins and May 2011	MIRA (ISI)
Xiang and Ittycheriah 2011	
Cherry and Foster 2012	
Chiang 2012	
Gimpel 2012	
Simianer et al. 2012	
Watanabe 2012	

Feature-Rich Research	**Industry/Evaluations**
Liang et al. 2006	
Tillmann and Zhang 2006	
Arun and Koehn 2007	
Ittycheriah and Roukos 2007	
Watanabe et al. 2007	
Chiang et al. 2008; Chiang et al. 2009	*n*-best/lattice MERT
Haddow et al. 2011	
Hopkins and May 2011	MIRA (ISI)
Xiang and Ittycheriah 2011	
Cherry and Foster 2012	
Chiang 2012	
Gimpel 2012	
Simianer et al. 2012	
Watanabe 2012	

Feature-rich Shared Task Submissions

		# Feature-rich
2012	WMT	0
	IWSLT	1
2013	WMT	2 ?
	IWSLT	TBD

Speculation: Entrenchment Of MERT

Feature-rich on small tuning sets?

Implementation complexity

Open source availability

Speculation: Entrenchment Of MERT

Feature-rich on small tuning sets?

Implementation complexity

Open source availability

Top-selling phone of 2003

Motivation: Why Feature-Rich MT?

Make MT more like other machine learning settings

Features for specific errors

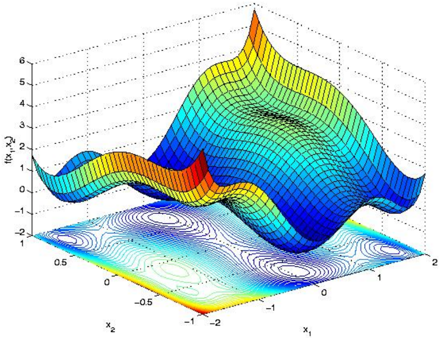
Domain adaptation

Motivation: Why Online MT Tuning?

Search: decode more often

Better solutions
   See: [Liang and Klein 2009]

Computer-aided translation:
incremental updating



Benefits Of Our Method

Fast and scalable

Adapts to dense/sparse feature mix

Not complicated

Online Algorithm Overview

Updating with an adaptive learning rate

Automatic feature selection via L_1 regularization

Loss function: Pairwise ranking

Notation

t time/update step

Notation

t	time/update step
w_t	weight vector in $\mathbb{R}^n$

Notation

t	time/update step
w_t	weight vector in $\mathbb{R}^n$
η	learning rate

Notation

t	time/update step
w_t	weight vector in $\mathbb{R}^n$
η	learning rate
$\ell_t(w)$	loss of t'th example

Notation

t	time/update step
w_t	weight vector in $\mathbb{R}^n$
η	learning rate
$\ell_t(w)$	loss of t'th example
$z_{t-1} \in \partial \ell_t(w_{t-1})$	subgradient set (*subdifferential*)

Notation

t	time/update step
w_t	weight vector in $\mathbb{R}^n$
η	learning rate
$\ell_t(w)$	loss of t'th example
$z_{t-1} \in \partial \ell_t(w_{t-1})$	subgradient set (*subdifferential*)
$z_{t-1} = \nabla \ell_t(w_{t-1})$	for differentiable loss functions

Notation

t	time/update step
w_t	weight vector in $\mathbb{R}^n$
η	learning rate
$\ell_t(w)$	loss of t'th example
$z_{t-1} \in \partial \ell_t(w_{t-1})$	subgradient set (*subdifferential*)
$z_{t-1} = \nabla \ell_t(w_{t-1})$	for differentiable loss functions
$r(w)$	regularization function

Warm-up: Stochastic Gradient Descent

Per-instance update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Warm-up: Stochastic Gradient Descent

Per-instance update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Issue #1: learning rate schedule

η / t ?

Warm-up: Stochastic Gradient Descent

Per-instance update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Issue #1: learning rate schedule

$\eta \,/\, t$?

$\eta \,/\, \sqrt{t}$?

Warm-up: Stochastic Gradient Descent

Per-instance update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Issue #1: learning rate schedule

$\eta \,/\, t$?

$\eta \,/\, \sqrt{t}$?

$\eta \,/\, (1 + \gamma t)$? Yuck.

Warm-up: Stochastic Gradient Descent

SGD update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Issue #2: same step size for every coordinate

Warm-up: Stochastic Gradient Descent

SGD update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Issue #2: same step size for every coordinate

Intuitively, we might want:

- Frequent feature: small steps e.g. $\eta \,/\, t$
- Rare feature: large steps e.g. $\eta \,/\, \sqrt{t}$

SGD: Learning Rate Adaptation

SGD update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Scale learning rate with $A^{-1} \in \mathbb{R}^{n \times n}$:

$$w_t = w_{t-1} - \eta A^{-1} z_{t-1}$$

SGD: Learning Rate Adaptation

SGD update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Scale learning rate with $A^{-1} \in \mathbb{R}^{n \times n}$:

$$w_t = w_{t-1} - \eta A^{-1} z_{t-1}$$

Choices:

$A^{-1} = I$ (SGD)

SGD: Learning Rate Adaptation

SGD update:

$$w_t = w_{t-1} - \eta z_{t-1}$$

Scale learning rate with $A^{-1} \in \mathbb{R}^{n \times n}$:

$$w_t = w_{t-1} - \eta A^{-1} z_{t-1}$$

Choices:

$A^{-1} = I$ (SGD)

$A^{-1} = H^{-1}$ (Batch: Newton step)

AdaGrad

Duchi et al. 2011

Update:

$$w_t = w_{t-1} - \eta A^{-1} z_{t-1}$$

Set $A^{-1} = G_t^{-1/2}$:

$$G_t = G_{t-1} + z_{t-1} \cdot z_{t-1}^{\top}$$

AdaGrad: Approximations and Intuition

For high-dimensional w_t, use diagonal G_t

$$w_t = w_{t-1} - \eta G_t^{-1/2} z_{t-1}$$

AdaGrad: Approximations and Intuition

For high-dimensional w_t, use diagonal G_t

$$w_t = w_{t-1} - \eta G_t^{-1/2} z_{t-1}$$

Intuition:

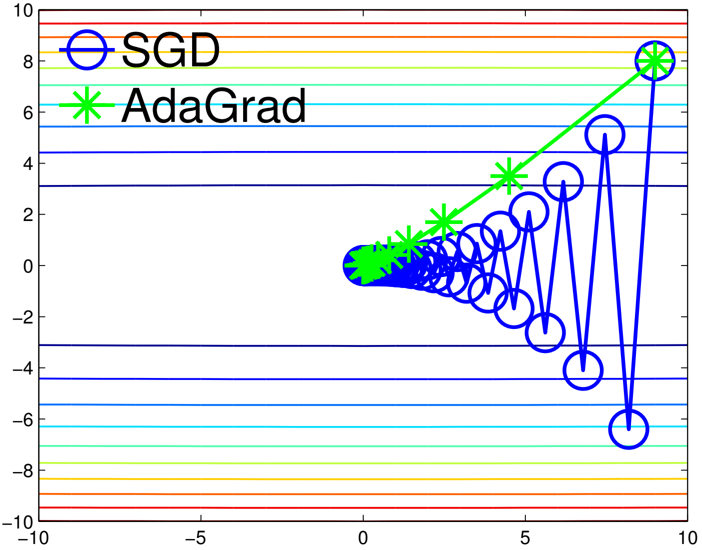
$1/\sqrt{t}$ schedule on constant gradient

Small steps for frequent features

Big steps for rare features

[Duchi et al. 2011]

AdaGrad vs. SGD: 2D Illustration



Feature Selection

Traditional approach: frequency cutoffs

Unattractive for large tuning sets (e.g. bitext)

Feature Selection

Traditional approach: frequency cutoffs

Unattractive for large tuning sets (e.g. bitext)

More principled: L_1 regularization

$$r(w) = \sum_i |w_i|$$

Feature Selection: FOBOS

Two-step update:

$$w_{t-\frac{1}{2}} = w_{t-1} - \eta z_{t-1} \tag{1}$$

$$w_t = \arg\min_w \left(\underbrace{\frac{1}{2}\left\| w - w_{t-\frac{1}{2}} \right\|^2}_{\text{proximal term}} + \underbrace{\lambda \cdot r(w)}_{\text{regularization}} \right) \tag{2}$$

[Duchi and Singer 2009]

Feature Selection: FOBOS

Two-step update:

$$w_{t-\frac{1}{2}} = w_{t-1} - \eta z_{t-1} \tag{1}$$

$$w_t = \arg\min_{w} \left(\underbrace{\frac{1}{2} \left\| w - w_{t-\frac{1}{2}} \right\|^2}_{\text{proximal term}} + \underbrace{\lambda \cdot r(w)}_{\text{regularization}} \right) \tag{2}$$

[Duchi and Singer 2009]

Extension: AdaGrad update in step (1)

Feature Selection: FOBOS

For L_1, FOBOS becomes soft thresholding:

$$w_t = \text{sign}(w_{t-\frac{1}{2}}) \left[\left| w_{t-\frac{1}{2}} \right| - \lambda \right]_+$$

Feature Selection: FOBOS

For L_1, FOBOS becomes soft thresholding:

$$w_t = \text{sign}(w_{t-\frac{1}{2}})\left[\,\left|w_{t-\frac{1}{2}}\right| - \lambda\,\right]_+$$

Squared-L_2 also has a simple form

Feature Selection: Lazy Regularization

Lazy updating: only update active coordinates

Big speedup in MT setting

Feature Selection: Lazy Regularization

Lazy updating: only update active coordinates

Big speedup in MT setting

Easy with FOBOS:

t'_j : last update of dimension j

Use $\lambda(t - t'_j)$

AdaGrad+FOBOS: Full Algorithm

1. Additive update: G_t

AdaGrad+FOBOS: Full Algorithm

1. Additive update: G_t
2. Additive update: $w_{t-\frac{1}{2}}$

AdaGrad+FOBOS: Full Algorithm

1. Additive update: G_t
2. Additive update: $w_{t-\frac{1}{2}}$
3. Closed-form regularization: w_t

AdaGrad+FOBOS: Full Algorithm

1. Additive update: G_t
2. Additive update: $w_{t-\frac{1}{2}}$
3. Closed-form regularization: w_t

AdaGrad+FOBOS: Full Algorithm

1. Additive update: G_t
2. Additive update: $w_{t-\frac{1}{2}}$
3. Closed-form regularization: w_t

Not complicated

Very fast

Recap: Pairwise Ranking

For derivation d, feature map $\phi(d)$, references $e^{1:k}$

Metric: $B(d, e^{1:k})$ (e.g. BLEU+1)

Model score: $M(d) = w \cdot \phi(d)$

Recap: Pairwise Ranking

For derivation d, feature map $\phi(d)$, references $e^{1:k}$

Metric: $B(d, e^{1:k})$ (e.g. BLEU+1)

Model score: $M(d) = w \cdot \phi(d)$

Pairwise consistency:

$$M(d_+) > M(d_-) \iff B\left(d_+, e^{1:k}\right) > B\left(d_-, e^{1:k}\right)$$

[Hopkins and May 2011]

Loss Function: Pairwise Ranking

$$M(d_+) > M(d_-) \iff w \cdot (\phi(d_+) - \phi(d_-)) > 0$$

Loss Function: Pairwise Ranking

$$M(d_+) > M(d_-) \iff w \cdot (\phi(d_+) - \phi(d_-)) > 0$$

Loss formulation:

Difference vector: $v = \phi(d_+) - \phi(d_-)$

Find w so that $w \cdot v > 0$

Binary classification problem between v and $-v$

Loss Function: Pairwise Ranking

$$M(d_+) > M(d_-) \iff w \cdot (\phi(d_+) - \phi(d_-)) > 0$$

Loss formulation:

Difference vector: $v = \phi(d_+) - \phi(d_-)$

Find w so that $w \cdot v > 0$

Binary classification problem between v and $-v$

Logistic loss: convex, differentiable

[Hopkins and May 2011]

Parallelization

Online algorithms are inherently sequential

Parallelization

Online algorithms are inherently sequential

Out-of-order updating:

$$w_7 = w_6 - \eta z_4$$
$$w_8 = w_7 - \eta z_6$$
$$w_9 = w_8 - \eta z_5$$

Parallelization

Online algorithms are inherently sequential

Out-of-order updating:

$$
\begin{aligned}
w_7 &= w_6 - \eta z_4 \\
w_8 &= w_7 - \eta z_6 \\
w_9 &= w_8 - \eta z_5
\end{aligned}
$$

Low-latency regret bound: $O(\sqrt{T})$ [Langford et al. 2009]

Translation Quality Experiments

Arabic-English (Ar–En) and Chinese-English (Zh–En)

Newswire and mixed-genre experiments

BOLT bitexts: data up to 2012

	Bilingual		Monolingual
	Sentences	*Tokens*	*Tokens*
Ar–En	6.6M	375M	990M
Zh–En	9.3M	538M	

MT System

Phrase-based MT: Phrasal

[Cer et al. 2010]

Dense baseline: MERT

Cer et al. 2008 line search

Accumulates *n*-best lists

Random starting points, etc.

Feature-Rich Baseline: PRO

Pairwise Ranking Optimization (PRO)

- Batch log loss minimization

Phrasal implementation:

- L-BFGS with L_2 regularization

[Hopkins and May 2011]

Feature-Rich Baseline: PRO

Pairwise Ranking Optimization (PRO)

- Batch log loss minimization

Phrasal implementation:

- L-BFGS with L_2 regularization

[Hopkins and May 2011]

Sanity check: Moses PRO and kb-MIRA (batch) implementations

Dense Features

8	Hierarchical lex. reordering

Dense Features

8	Hierarchical lex. reordering
5	Moses phrase table features
1	Rule bitext count
1	Unique rule indicator

Dense Features

8	Hierarchical lex. reordering
5	Moses phrase table features
1	Rule bitext count
1	Unique rule indicator
1	Word penalty
1	Linear distortion
1	LM
1	Unknown word
19	

Sparse Feature Templates

Discriminative Phrase Table (PT)

Rule indicator: $\mathbb{1}\left(\text{برنامج الفضاء} \Rightarrow \text{space program}\right)$

Sparse Feature Templates

Discriminative Phrase Table (PT)

Rule indicator: $\mathbb{1}\left(\text{برنامج الفضاء} \Rightarrow \text{space program}\right)$

Discriminative Alignments (AL)

Source word deletion: $\mathbb{1}\left(\text{الفضاء} \Rightarrow\right)$

Word alignments: $\mathbb{1}\left(\text{الفضاء} \Rightarrow \text{space}\right)$

Sparse Feature Templates

Discriminative Phrase Table (PT)

Rule indicator: $\mathbb{1}\left(\text{برنامج الفضاء} \Rightarrow \text{space program}\right)$

Discriminative Alignments (AL)

Source word deletion: $\mathbb{1}\left(\text{الفضاء} \Rightarrow\right)$

Word alignments: $\mathbb{1}\left(\text{الفضاء} \Rightarrow \text{space}\right)$

Discriminative Lex. Reordering (LO)

Phrase orientation: $\mathbb{1}\left(\text{swap}(\text{الفضاء} \Rightarrow \text{space})\right)$

Evaluation: NIST OpenMT

Small tuning set: MT06

“Large” tuning set: MT0568 (≈4200 segments)

BLEU-4 uncased, Four references

Evaluation: NIST OpenMT

Small tuning set: MT06

“Large” tuning set: MT0568 (≈4200 segments)

BLEU-4 uncased, Four references

Paper: mixed genre (bitext) experiments

Results: Small Tuning Set (Dense)

	Ar–En		Zh–En	
	Tune	**Test Avg.**	Tune	**Test Avg.**
MERT	45.08	50.51	33.73	34.49
This paper	43.16	50.11	32.20	35.25

Results: Add More Features

	Ar–En		Zh–En	
	Tune	**Test Avg.**	Tune	**Test Avg.**
MERT—Dense	45.08	50.51	33.73	34.49
This paper +PT	50.61	50.52	34.92	35.12

Results: Add More Features

	Ar–En		Zh–En	
	Tune	**Test Avg.**	Tune	**Test Avg.**
MERT—Dense	45.08	50.51	33.73	34.49
This paper +PT	50.61	50.52	34.92	35.12
This paper +All	60.85	50.97	39.43	35.31

(MT06 tuning set)

Results: Add More Data

	Ar–En	Zh–En
	Test Avg.	**Test Avg.**
MERT—mt06	50.51	34.49
MERT—mt0568	50.74	34.55

Results: Add More Data

	Ar–En	Zh–En
	Test Avg.	**Test Avg.**
MERT—mt06	50.51	34.49
MERT—mt0568	50.74	34.55
This paper		
+All—mt06	50.97	35.31

Results: Add More Data

	Ar–En		Zh–En	
	Test Avg.		**Test Avg.**	
MERT—mt06	50.51		34.49	
MERT—mt0568	50.74		34.55	
This paper				
+All—mt06	50.97		35.31	
+All—mt0568	**52.34**	+1.60	**36.61**	+2.06

Results: Add More Data

	Ar–En Test Avg.		Zh–En Test Avg.	
MERT—mt06	50.51		34.49	
MERT—mt0568	50.74		34.55	
This paper				
+All—mt06	50.97		35.31	
+All—mt0568	**52.34**	+1.60	**36.61**	+2.06

PRO+All worse than MERT—mt0568

Analysis: Zh–En MT06 Tuning

(16 threads)		Epochs	Min/epoch
MERT	Dense	22	180

Analysis: Zh–En MT06 Tuning

(16 threads)		**Epochs**	**Min/epoch**
MERT	Dense	22	180
PRO	+PT	25	35
kb-MIRA*	+PT	26	25
This paper	+PT	10	10

Analysis: Zh–En MT06 Tuning

(16 threads)		Epochs	Min/epoch
MERT	Dense	22	180
PRO	+PT	25	35
kb-MIRA*	+PT	26	25
This paper	+PT	10	10
PRO	+All	13	100
This paper	+All	5	15

Analysis: Zh–En MT06 Tuning

(16 threads)		**Epochs**	**Min/epoch**
MERT	Dense	22	180
PRO	+PT	25	35
kb-MIRA*	+PT	26	25
This paper	+PT	10	10
PRO	+All	13	100
This paper	+All	5	15

MERT—mt0568 tuning takes about 5 days

Analysis: Runtime

Online regret bounds depend on # updates

- Large datasets: more updates per epoch
- Fewer epochs to converge

Analysis: Runtime

Online regret bounds depend on # updates

- Large datasets: more updates per epoch
- Fewer epochs to converge

Lazy updating helps:

- $w_t \approx 100k$ features
- $z_{t-1} \approx 500$ features

Analysis: Reordering

Arabic matrix clauses often verb-initial

Analysis: Reordering

Arabic matrix clauses often verb-initial

Manually selected 208 verb-initial segments (MT09)

Analysis: Reordering

Arabic matrix clauses often verb-initial

Manually selected 208 verb-initial segments (MT09)

32 differed for MERT–Dense vs. +All

Analysis: Reordering

+All correct	18	56.3%
MERT–Dense correct	4	12.5%
Both wrong	10	31.3%
	32	

Analysis: Reordering

+All correct	18	56.3%
MERT–Dense correct	4	12.5%
Both wrong	10	31.3%
	32	

ref: `the newspaper and television reported`

MERT she said the newspaper and television

+All television and newspaper said

Analysis: Domain Adaptation

برنامج ⇒ *program, programme*

Analysis: Domain Adaptation

برنامج ⇒ *program, programme*

	# bitext–5k	# MT0568
programme	185	0
program	19	449

Analysis: Domain Adaptation

برنامج ⇒ *program, programme*

	# bitext–5k	# MT0568
programme	185	0
program	19	449
+PT rules: *programme*	353	79
+PT rules: *program*	9	31

Caveats and Next Steps

- Single-reference setting
  - BLEU+1 is unreliable
  - Lexicalized features cause overfitting

Caveats and Next Steps

- Single-reference setting
  - BLEU+1 is unreliable
  - Lexicalized features cause overfitting
- Current work
  - Bitext tuning
  - Different loss function

Conclusion

Fast, adaptive, online tuning for MT

Conclusion

Fast, adaptive, online tuning for MT

Easy to implement

Conclusion

Fast, adaptive, online tuning for MT

Easy to implement

Works as well as MERT for Dense

Conclusion

Fast, adaptive, online tuning for MT

Easy to implement

Works as well as MERT for Dense

Sane feature engineering

Fast and Adaptive Online Training of Feature-Rich Translation Models

Spence Green **Sida Wang**

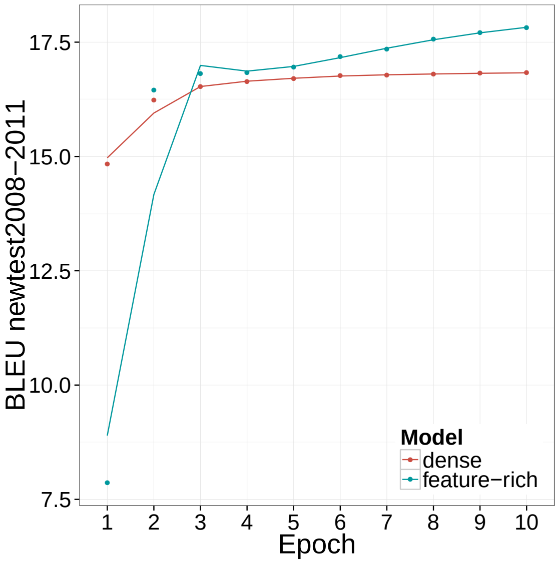
Daniel Cer **Christopher D. Manning**

Stanford University

Try the code in Phrasal:

nlp.stanford.edu/software/phrasal/

En–De Learning Curve



Sparse Features: Negative Results

Discriminative LM	Jane **called Sally**
Phrase boundary features	**Jane** || **called** Sally
Alignment constellation	**1-0 0-1**
Target word insertion	Jane called **the** Sally