Tuning Machine Translation Parameters with SPSA

Patrik Lambert and Rafael E. Banchs

TALP Research Center
Jordi Girona Salgado, 1-3
08034 Barcelona, Spain

IWSLT 2006, Kyoto

Motivation

- Nowadays most SMT systems consist of a log-linear combination of various models.
- Model weight values have a critical impact on translation quality. This impact can be greater than the impact of making some improvement in the system.
- In spite of this, model weights optimisation process is highly dependent on starting values.

Motivation

- Nowadays most SMT systems consist of a log-linear combination of various models.
- Model weight values have a critical impact on translation quality. This impact can be greater than the impact of making some improvement in the system.
- In spite of this, model weights optimisation process is highly dependent on starting values.

$\Rightarrow$ problem of result significance. Is the increase of translation quality score due to system improvement or to optimisation variability ?

Motivation

- Nowadays most SMT systems consist of a log-linear combination of various models.
- Model weight values have a critical impact on translation quality. This impact can be greater than the impact of making some improvement in the system.
- In spite of this, model weights optimisation process is highly dependent on starting values.

$\Rightarrow$ problem of result significance. Is the increase of translation quality score due to system improvement or to optimisation variability ?

Objective

Try to achieve more reproducible outcome of the optimisation process

Optimisation problem

- MT system implementing log-linear combination of models
- Problem: maximising score of translation quality measure adjusting the model scaling factors over development data
- Characteristics of objective function:
  - no analytic representation, so the gradient cannot be calculated
  - many local minima
  - its evaluation has a significant computational cost

  $\Rightarrow$ discard: algorithms based on derivatives, algorithms such as simulated annealing or genetic algorithms
- popular methods: Direction set (or Powell's) method, downhill simplex (or Nelder and Mead's) Method

Simultaneous Perturbation Stochastic Approximation

- The SPSA method [J. Spall, 1992] is based on a gradient approximation which requires only two evaluations of the objective function, regardless of the dimension of the optimisation problem.
- SPSA procedure is in the general recursive stochastic approximation form:

$$\hat{\lambda}_{k+1} = \hat{\lambda}_k - \mathbf{a}_k \hat{\mathbf{g}}_k(\hat{\lambda}_k)$$

$\hat{\mathbf{g}}_k(\hat{\lambda}_k)$: estimate of the gradient $\mathbf{g}(\lambda) \equiv \partial E / \partial \lambda$ at iterate k

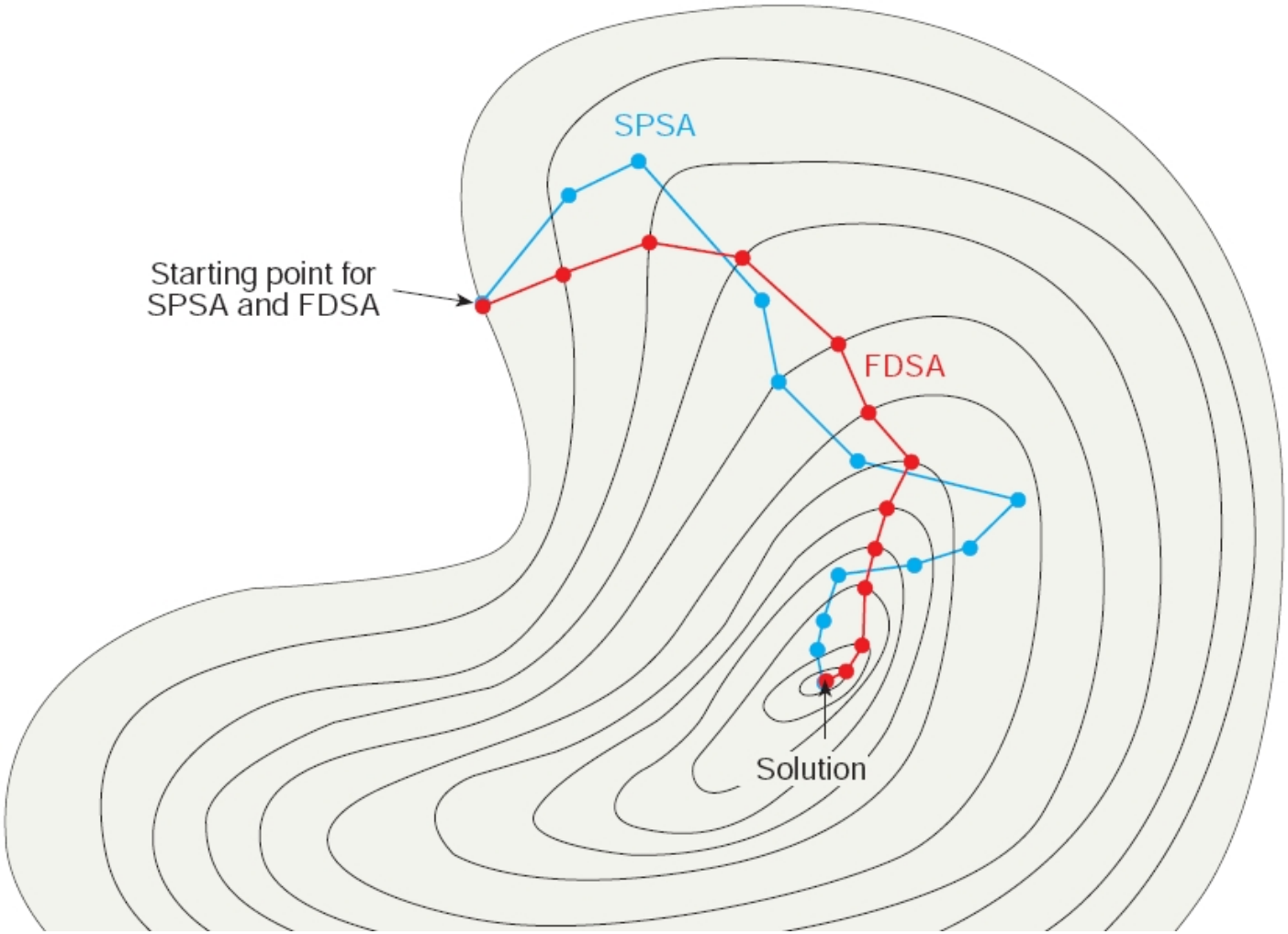
Simultaneous Perturbation Stochastic Approximation

- The SPSA method [J. Spall, 1992] is based on a gradient approximation which requires only two evaluations of the objective function, regardless of the dimension of the optimisation problem.
- SPSA procedure is in the general recursive stochastic approximation form:

$$\hat{\lambda}_{k+1} = \hat{\lambda}_k - \mathbf{a}_k \hat{\mathbf{g}}_k(\hat{\lambda}_k)$$

  $\hat{\mathbf{g}}_k(\hat{\lambda}_k)$: estimate of the gradient $\mathbf{g}(\lambda) \equiv \partial E / \partial \lambda$ at iterate k
- simultaneous perturbation approximation of the gradient:

$$\hat{\mathbf{g}}_k(\hat{\lambda}_k) = \frac{E(\hat{\lambda}_k + c_k \mathbf{\Delta_k}) - E(\hat{\lambda}_k - c_k \mathbf{\Delta_k})}{2c_k} \begin{bmatrix} 1/\Delta_{k1} \\ 1/\Delta_{k2} \\ \vdots \\ 1/\Delta_{kN} \end{bmatrix}$$



The simultaneous approximation causes deviations of the search path. These deviations are averaged out in reaching a solution.

Algorithm

It is very simple to implement. Our (slightly modified) implementation:

1. Calculate gain sequences a_k and c_k.
2. Generate the simultaneous perturbation vector $\mathbf{\Delta}_k$. For example, take for each component of $\mathbf{\Delta}_k$ a Bernoulli ± 1 distribution with probability of $1/2$ for each ± 1 outcome
3. Evaluate $E(\hat{\lambda}_k + c_k \mathbf{\Delta_k})$
4. Approximate the gradient as seen above, but replacing $E(\hat{\lambda}_k - c_k \mathbf{\Delta_k})$ by $E(\hat{\lambda}_k)$ and dividing by c_k instead of $2c_k$ (one-sided approximation)
5. Update λ estimate ; evaluate the objective function with this new set of parameters. Accept the new parameters according to the following probability: $\exp\left[-|E(\hat{\lambda}_{k+1}) - E(\hat{\lambda}_k)|/T(k+1)\right]$
6. Iteration or termination

Procedure

Experiment goal: maximise $E(\lambda)$ (here E is the BLEU score), with downhill simplex and SPSA methods, and compare the consistency of the results over changes in initial settings.

- ran the algorithms from 7 different initial points and for each point, for 10 slightly different realisations
- we compared the number of objective function evaluations necessary to reach some value
- Note: in our optimisation scheme, each function evaluation requires translating the development corpus (no rescoring)
- For SPSA method, each of the 10 realisations of the algorithm corresponded to a different seed used to randomly generate the simultaneous perturbation vector $\mathbf{\Delta}_k$.

Downhill simplex algorithm

- The method uses a geometrical figure called a *simplex* (in N dimensions, consists of $N+1$ points and all their interconnecting line segments, polygonal faces, etc.)
- Starting point: initial simplex ($N+1$ points in parameter space)
- At each step, the simplex performs geometrical operations (reflexions, contractions and expansions) until a local minimum is reached
- Given a starting point $\mathbf{P}_0$, the other N points of the initial simplex were taken to be $\mathbf{P}_i = \mathbf{P}_0 + \alpha_i \mathbf{e}_i$, where the $\mathbf{e}_i$ are unit vectors
- The N constants α_i were chosen randomly
- The 10 different realisations of the algorithm were obtained by varying the seed of the random generator used to compute the α_i constants

Translation system and data

We used the TALP N-gram system, with the following feature functions:

- a translation model (based on a 4-grams language model of bilingual units, called tuples, which are extracted from Viterbi alignments)
- a 4-gram language model of the target language
- a 4-gram language model of target POS-tags
- a word bonus feature
- two lexicon models

Train and Development data were those of Chinese-English IWSLT 2006 task. Test data was a set of 500 sentences among dev1, dev2, dev3.

Results on development data

We calculated, for each of the 7 starting points, the average BLEU score and standard deviation after running 20, 40, 60 and 90 function evaluations of SPSA and simplex algorithms.
(For each starting point, average and standard deviation are calculated over the 10 slightly different realisations controlled with the random seeds)

- Both algorithm exhibit similar performance (optimum value reached after N evaluations)
- From 60 evaluations on, standard deviation always smaller for SPSA
- Example: smallest and highest standard deviation difference after 60 function evaluations:

	Starting points	
	2	7
simplex	19.8±0.14	19.2±0.41
SPSA	19.7±0.11	19.5±0.11

Results on development data

Now, fix the seed and see algorithm behaviour across several initial points.

- Average and standard deviation (over the 7 starting points) of the averages (over different realisations):

	Function Evaluations			
	20	40	60	90
simplex	19.0±0.58	19.3±0.45	19.4±0.37	19.5±0.33
SPSA	18.9±0.64	19.4±0.22	19.5±0.09	19.7±0.08

Results on development data

Now, fix the seed and see algorithm behaviour across several initial points.

- Average and standard deviation (over the 7 starting points) of the averages (over different realisations):

	Function Evaluations			
	20	40	60	90
simplex	19.0±0.58	19.3±0.45	19.4±0.37	19.5±0.33
SPSA	18.9±0.64	19.4±0.22	19.5±0.09	19.7±0.08

Results on development data

Now, fix the seed and see algorithm behaviour across several initial points.

- Average and standard deviation (over the 7 starting points) of the averages (over different realisations):

	Function Evaluations			
	20	40	60	90
simplex	19.0±0.58	19.3±0.45	19.4±0.37	19.5±0.33
SPSA	18.9±0.64	19.4±0.22	19.5±0.09	19.7±0.08

Results on development data

Now, fix the seed and see algorithm behaviour across several initial points.

- Average and standard deviation (over the 7 starting points) of the averages (over different realisations):

	Function Evaluations			
	20	40	60	90
simplex	19.0±0.58	19.3±0.45	19.4±0.37	19.5±0.33
SPSA	18.9±0.64	19.4±0.22	19.5±0.09	19.7±0.08

Results on development data

Now, fix the seed and see algorithm behaviour across several initial points.

- Average and standard deviation (over the 7 starting points) of the averages (over different realisations):

	Function Evaluations			
	20	40	60	90
simplex	19.0±0.58	19.3±0.45	19.4±0.37	19.5±0.33
SPSA	18.9±0.64	19.4±0.22	19.5±0.09	19.7±0.08

- For each seed, calculate average BLEU score and standard deviation over 7 starting points, after 20, 40, 60, and 90 function evaluations
  $\Rightarrow$ from 60 evaluations on, stand. deviation always smaller for SPSA

Results on Development Data

Conclusion: the optimum value obtained with SPSA is less sensitive to the choice of initial parameters, which should lead to more consistent results.

Possible explanations:

- SPSA search path follows in average the direction of the gradient, whereas the simplex orientation is blind
- SPSA has always a probability to go away from current hill, whereas the simplex gets stuck there

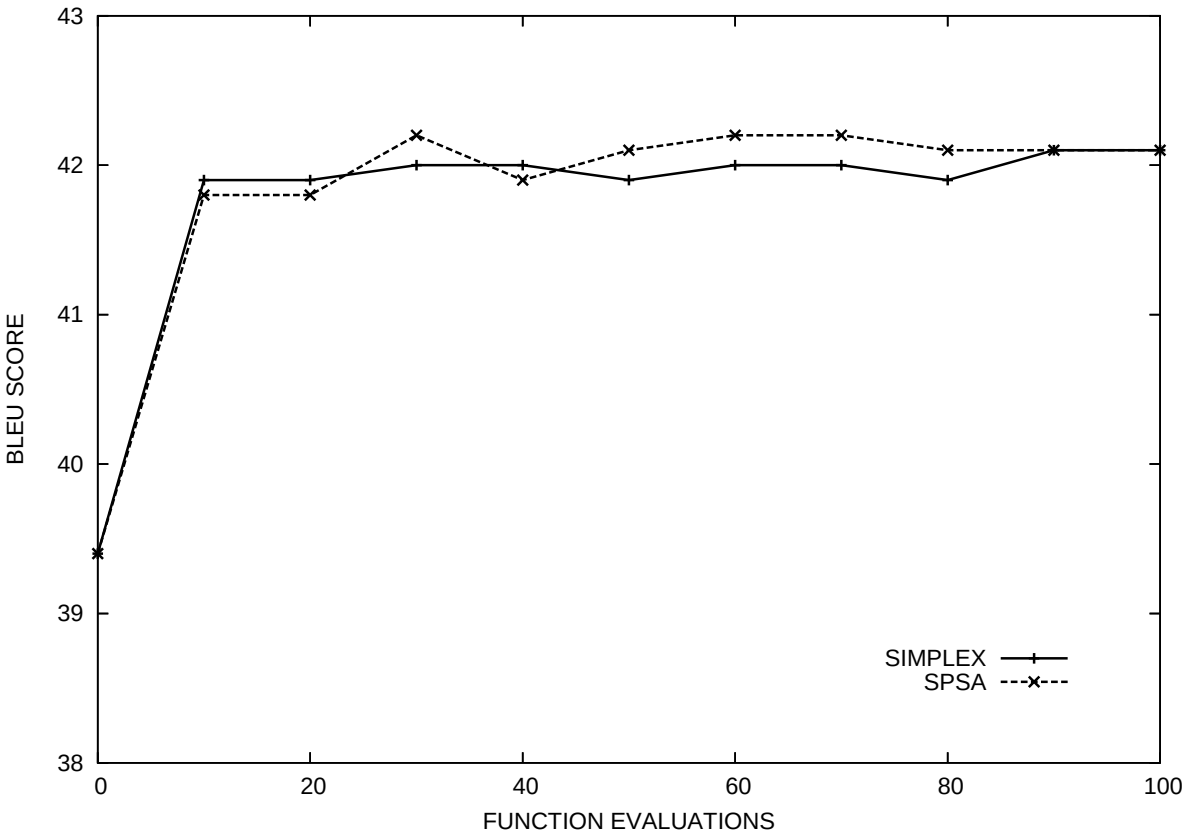
Results on Test Data

For each initial point and seed, and after a given number of function evaluations, we collected the optimum parameter set over the development corpus, and translated the test corpus with these parameters.

- as expected, dispersion of scores is higher in test than in development
- standard deviation in test is similar for both algorithms
  $\Rightarrow$ stability gain observed in development for SPSA not conserved with new data
- average BLEU score in test is similar for both algorithms

Results on Test Data

We selected the parameters corresponding to the best score out of the 10 realisations, and translated the test corpus with these parameters. Plot of the average over starting points of these scores:



Conclusions

- SPSA and simplex algorithm seem to have similar performance (SPSA expected to perform better in systems with more models)
- SPSA was more robust with respect to the choice of initial parameters and with respect to different realisations of the algorithm
- This SPSA advantage was not conserved when using the optimal parameters to translate new data
- Whatever the algorithm: very high BLEU score dispersion for test data (0.3 to 1.1 absolute)
  - dispersion over-evaluated because in this task development, test and training corpus are small ? $\Rightarrow$ repeat these experiments with more data
  - significance problem still unresolved
  - to alleviate the problem: always perform a bunch of optimisations, translate new text with each optimal set of parameters, and average.