# An Empirical Analysis of Optimization for Max-Margin NLP

Jonathan K. Kummerfeld, Taylor Berg-Kirkpatrick, and Dan Klein
{jkk,tberg,klein}@cs.berkeley.edu

## Some margin optimizers are better than others... Use this one!

There is substantial variation in the effectiveness of optimization methods for structured max-margin objectives. We investigated the behavior of a range of optimizers (dual and primal margin, likelihood, perceptron, and MIRA) for training high-performance systems for NER, coreference, parsing and summarization.

### Symbol Definitions

| | |
|---|---|
| w | Weights |
| q | Sum of squared subgradients |
| u | Last update for each weight |
| n | Number of updates |
| g | Subgradient |
| f | Feature index |
| eta | Learning rate |
| delta | Small value to prevent q[f] = 0 |

```
function optimize-margin(data, iters):
  w[] = 0
  q[] = delta
  u[] = 0
  n = 0
  for iter in [1, iters]:
    for (x*, y*) in data:
      y' =   argmax   (score(x*, y) + L(y, y*))
        (over y in Y(x*))
      g = features(y*) - features(y')
      q += g^2
      n += 1
      for f in (nonzero features in g):
        w[f] = update-active(w[f], g[f], q[f])
        w[f] = n

function update-active(w, g, q):
  d = |w - g*eta / sqrt(q)| - C*eta / sqrt(q)
  return sign(w - g*eta / sqrt(q)) * max(0, d)

function update-catchup(w, q, t):
  return sign(w)*max(0, |w| - eta*C*t / sqrt(q))

function score(x, y):
  s = 0
  for f in features(x, y):
    w[f] = update-catchup(w[f], q[f], n - u[f])
    u[f] = n
    s += w[f]
  return s
```

### Main Loop

Loss augmented decoding, with tracking of the sum of squared subgradients. A range of methods can be implemented by varying L, `update-active`, and `score`. For example, the perceptron is: `L = 0`, `score = f dot w`, and `update-active = w + g`

### AdaGrad Update

The AdaGrad update with L1 regularization. See the paper for L2.

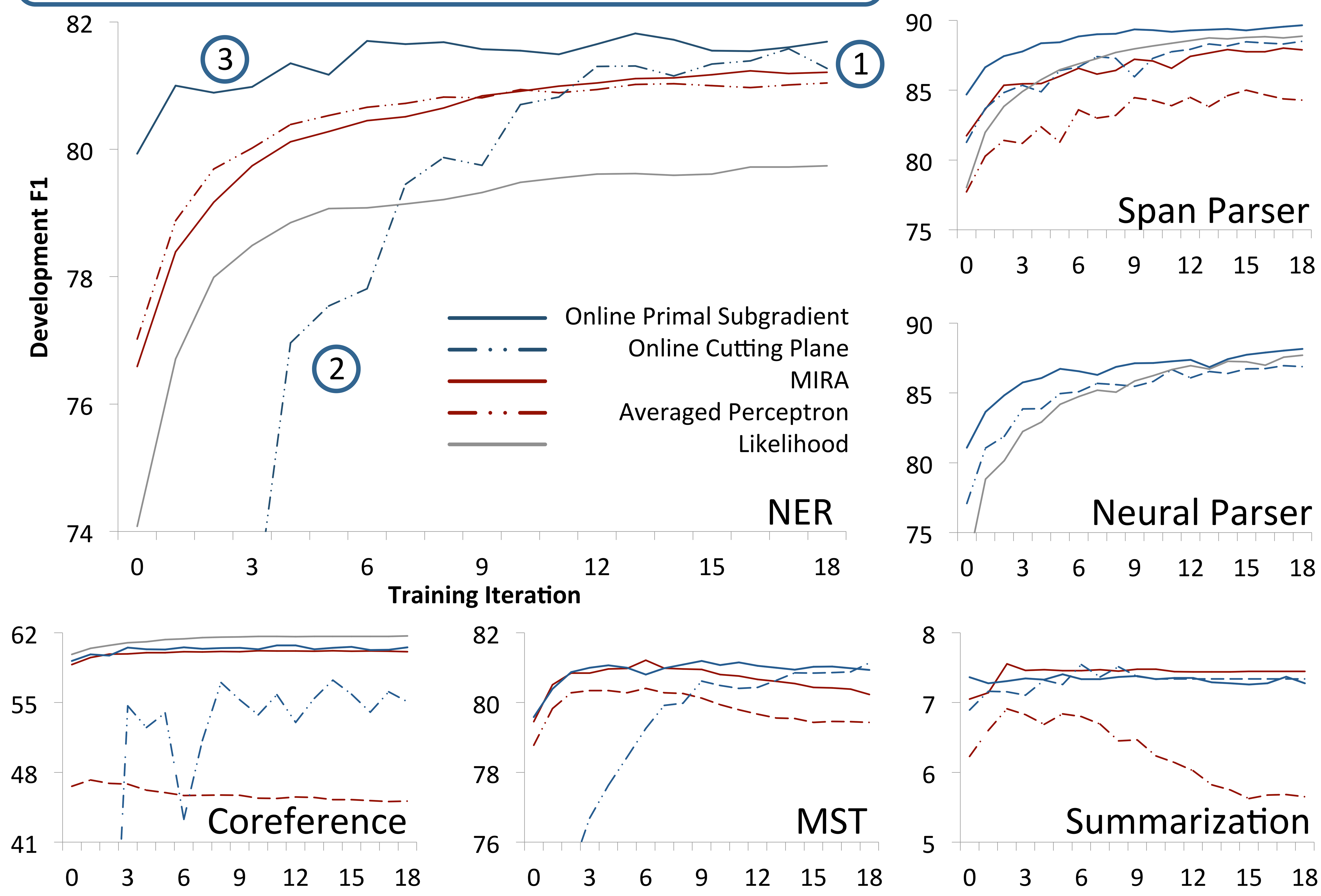
### Sparse updates

AdaGrad modifies every weight on every update. A fast way to implement this is to only update weights for nonzero features at first (in the main loop, above), and update other weights the next time they are needed(here).

### Observations

1. Max-margin generally outperformed MIRA, perceptron, likelihood.
2. Cutting plane methods learned more slowly.
3. OPS (Online Primal Subgradient) was easy, robust, and effective.

\+ Decoding dominated time for each iteration on most tasks. Sparse updates were crucial for OPS on some tasks.




### Code Release

See nlp.cs.berkeley.edu for our learning library!