Learning a global ranker: Perceptron training

[Collins, 2002]

- Presents a Perceptron learning algorithm for discriminatively training a Markov-based tagger
- Global ranking approach: all candidate solutions are considered
- Instantiation on sequential labeling problems using Viterbi-like inference
- Improved results, on POS tagging and NP chunking, compared to a MaxEnt tagger
- Best paper award at EMNLP-2002

(recall) Reranking: perceptron learning

[Collins and Duffy, 2002]

$S = \{(x_i, y_i)\}_{i=1}^{n}$; assume $GEN(x_i) = \{y_{i1}, \ldots, y_{in_i}\}$

Initialization: set parameters $\mathbf{w} = 0$
repeat for N epochs
  for $i = 1 \ldots n$
    $j = \arg\max_{j=1\ldots n_i} \Phi(x_i, y_{ij}) \cdot \mathbf{w}$
    if $(y_i \neq y_{ij})$ **then** $\mathbf{w} = \mathbf{w} + \Phi(x_i, y) - \Phi(x_i, y_{ij})$
  end-for
end-repeat
output: **w**

- A simple extension to dual perceptron exists: kernels and voted perceptron can be used

Learning a global ranker: Perceptron training

[Collins, 2002]

Initialization: set parameters $\mathbf{w} = 0$
repeat for N epochs
  for $i = 1 \ldots n$
    $\hat{y}_i = \arg\max_{y \in GEN(x_i)} \Phi(x_i, y) \cdot \mathbf{w}$
    if $(y_i \neq \hat{y}_i)$ **then** $\mathbf{w} = \mathbf{w} + \Phi(x_i, y) - \Phi(x_i, \hat{y}_i)$
  end-for
end-repeat
output: **w**

- The general algorithm is the same, but now GEN generates all possible solutions
- $|\text{GEN}(x)|$ is exponential in the length of x
- The implementation cannot list all $y \in GEN(x_i)$ explicitely

Decomposition: Local/Global

[Collins, 2002]

- Restricted to sequential labeling problems:
  - $x = x_1, x_2, \ldots, x_n = x_{[1:n]} = \mathbf{x}$
  - $y = t_1, t_2, \ldots, t_n = t_{[1:n]} = \mathbf{t}$
- "$\arg\max$" inference is performed using Viterbi search
- Each example $(\mathbf{x}, \mathbf{t})$ is decomposed into a finite set of parts $R(\mathbf{x}, \mathbf{t}) \subseteq \mathcal{R}$.
- In a trigram based tagger (linear number of parts): $R(\mathbf{x}, \mathbf{t}) = \bigcup_{i=1}^{n} r_i = \bigcup_{i=1}^{n} \langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i \rangle$

Decomposition: Feature Vectors

[Collins, 2002]

- Local feature-vector representation:
  $\phi(r_i) = \phi(\langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i \rangle) = (\phi_1(r_i), \phi_2(r_i), \ldots, \phi_d(r_i))$
  with local feature functions $\phi_j(r_i)$, one for each dimension

Decomposition: Feature Vectors

[Collins, 2002]

- Local feature-vector representation:
  $\phi(r_i) = \phi(\langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i \rangle) = (\phi_1(r_i), \phi_2(r_i), \ldots, \phi_d(r_i))$
  with local feature functions $\phi_j(r_i)$, one for each dimension
- The global feature-vector representation is the sum of local representations: $\boxed{\Phi(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi(r_i)}$

Decomposition: Feature Vectors

[Collins, 2002]

- Local feature-vector representation:
  $\phi(r_i) = \phi(\langle t_i, t_{i-1}, t_{i-2}, \mathbf{x}, i \rangle) = (\phi_1(r_i), \phi_2(r_i), \ldots, \phi_d(r_i))$
  with local feature functions $\phi_j(r_i)$, one for each dimension
- The global feature-vector representation is the sum of local representations: $\boxed{\Phi(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi(r_i)}$
- If local features are indicator functions, then the global features will be "counts", e.g:

$$\phi_{1000}(r_i) = \begin{cases} 1 & \text{if current word } x_i \text{ is "the" and } t_i = \text{DT} \\ 0 & \text{otherwise} \end{cases}$$

  Then, $\Phi_{1000}(x_{[1:n]}, t_{[1:n]}) = \sum_{i=1}^{n} \phi_{1000}(r_i)$ is the number of times "the" is seen tagged as DT in the example $(x_{[1:n]}, t_{[1:n]})$

Learning a global ranker: objective function

[Collins, 2002]

- $$\boxed{F(x_{[1:n]}) = \hat{t}_{[1:n]} = \arg\max_{t_{[1:n]}} \text{score}(x_{[1:n]}, t_{[1:n]})}$$

Learning a global ranker: objective function

[Collins, 2002]

- $$\boxed{F(x_{[1:n]}) = \hat{t}_{[1:n]} = \arg\max_{t_{[1:n]}} \text{score}(x_{[1:n]}, t_{[1:n]})}$$

- $\text{score}(x_{[1:n]}, t_{[1:n]}) = \sum_{r \in R(\mathbf{x},\mathbf{t})} \text{score}(r) =$

  $\sum_{i=1}^{n} \text{score}(r_i) = \sum_{i=1}^{n} \mathbf{w} \cdot \phi(r_i)$

  $\sum_{i=1}^{n} \sum_{j=1}^{d} w_j \cdot \phi_j(r_i) = \sum_{j=1}^{d} \sum_{i=1}^{n} w_j \cdot \phi_j(r_i) =$

  $\sum_{j=1}^{d} w_j \cdot \Phi_j(x_{[1:n]}, t_{[1:n]}) = \mathbf{w} \cdot \Phi(x_{[1:n]}, t_{[1:n]})$

- $\mathbf{w} = (w_1, \ldots, w_d)$ is the weight vector

Learning a global ranker: Perceptron training

[Collins, 2002]

Training set: $S = \{(x^i_{[1:n_i]}, t^i_{[1:n_i]})\}_{i=1}^{n}$

Initialization: set parameters $\mathbf{w} = \mathbf{0}$

repeat for N epochs

for $i = 1 \ldots n$

Use the Viterbi algorithm to compute:

$\hat{t}_{[1:n_i]} = \arg\max_{t_{[1:n_i]}} \mathbf{w} \cdot \Phi(x^i_{[1:n_i]}, t_{[1:n_i]})$

This search makes use of the decomposition of the problem: $r_i, \phi()$

if $(\hat{t}_{[1:n_i]} \neq t^i_{[1:n_i]})$ **then**

$\mathbf{w} = \mathbf{w} + \Phi(x^i_{[1:n_i]}, t^i_{[1:n_i]}) - \Phi(x^i_{[1:n_i]}, \hat{t}_{[1:n_i]})$

end-for

end-repeat

output: $\mathbf{w}$

Learning a global ranker: Perceptron training

[Collins, 2002]

- Use of the voted perceptron with "averaged" parameters
- Application to POS tagging and NP chunking
- Comparison to a MaxEnt-based tagger (MEMM with exactly the same features)
- Simple approach with 11.9% and 5.1% relative error reduction, respectively
- Best paper award at EMNLP-2002