Learning to Search

Artem Sokolov

Institute for Computational Linguistics, Heidelberg University

15 October 2018

Learning to Search

structured prediction as search problem

A structured learning problem consists of

- an input space $\mathcal{X}$
- an output space $\mathcal{Y}$
- a fixed but unknown data distribution $\mathcal{D}$ over $\mathcal{X} \times \mathcal{Y}$
- a loss function $\ell(y^*, \hat{y}) \to \mathbb{R}^+$, which measures the distance between the true (y^*) and predicted $(\hat{y})$ outputs.

The goal of structured learning is to use N samples, $\{x_i, y_i\}_{i=1}^N \sim \mathcal{D}$ and learn a mapping $f : \mathcal{X} \to \mathcal{Y}$,
that minimizes the expected loss $\mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, f(x))]$

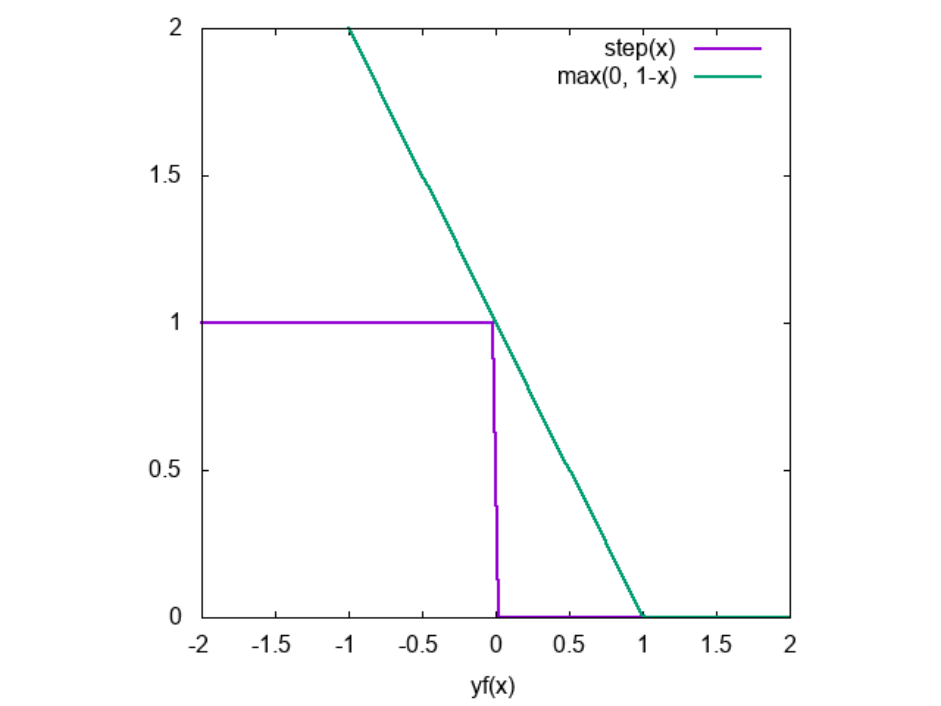
Perceptron [Rosenblatt, 1958]

init: $w_0 = 0$
for $t = 0, \ldots$ **do**
    observe x_t
    predict $\hat{y}_t = sign(w_t^\top x_t)$ (values: 0 or 1)
    recieve true label y_t
    if $\hat{y}_t \neq y_t$ **then**
        update $w_{t+1} = w_t + x_t$
    else
        keep $w_{t+1} = w_t$
    end if

- the $0/1$ loss, $\mathbb{I}[y \cdot w^\top x \leq 0]$, is unfortunately non-convex
- we already know that we should build some convex surrogate:

$$\ell(w, x) = \max(0, 1 - y \cdot w^\top x)$$

- this is called 'hinge-loss'
- it's convex and upper-bounds the $0/1$ loss, $\ell_t(w, x) \geq \mathbb{I}[y \cdot w^\top x \leq 0], \forall w$



Now we have an online convex optimization task:

1. receive x_t
2. predict $p_t = sign(w_t^\top x_t)$
3. suffer loss $\ell_t(w, x_t) = \max(0, 1 - y \cdot w^\top x_t)$
4. update w_t

Now we have an online convex optimization task:

1. receive x_t
2. predict $p_t = sign(w_t^\top x_t)$
3. suffer loss $\ell_t(w, x_t) = \max(0, 1 - y \cdot w^\top x_t)$
4. update w_t (how?)

Now we have an online convex optimization task:

1. receive x_t
2. predict $p_t = sign(w_t^\top x_t)$
3. suffer loss $\ell_t(w, x_t) = \max(0, 1 - y \cdot w^\top x_t)$
4. update w_t

We know already Follow-The-Regularized-Leader as a way to no-regret:

$$w_t = \arg\min \sum_{i=1}^{t} \ell_t(w, x_i) + R(w)$$

Now we have an online convex optimization task:

1. receive x_t
2. predict $p_t = sign(w_t^\top x_t)$
3. suffer loss $\ell_t(w, x_t) = \max(0, 1 - y \cdot w^\top x_t)$
4. update w_t

We know already Follow-The-Regularized-Leader as a way to no-regret:

$$w_t = \arg\min \sum_{i=1}^{t} \ell_t(w, x_i) + \frac{1}{2\nu} ||w||_2^2$$

We also know that for convex functions it is sufficient to work with linearized losses, setting $z_t = \nabla_w \ell_t(w)$:

Theorem

Consider FTRL, linear losses $\ell_t(w) = w^\top z_t$, and regularization $R(w) = \frac{1}{2\nu} ||w||_2^2$ and $w, u \in S = \mathbb{R}^d$, then

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t=1}^{T} ||z_t||_2^2 .$$

We also know that for convex functions it is sufficient to work with linearized losses, setting $z_t = \nabla_w \ell_t(w)$:

Theorem

Consider FTRL, linear losses $\ell_t(w) = w^\top z_t$, and regularization $R(w) = \frac{1}{2\nu} ||w||_2^2$ and $w, u \in S = \mathbb{R}^d$, then

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||z_t||_2^2 .$$

- when there is no error the gradient is zero

We also know that for convex functions it is sufficient to work with linearized losses, setting $z_t = \nabla_w \ell_t(w)$:

Theorem

Consider FTRL, linear losses $\ell_t(w) = w^\top z_t$, and regularization $R(w) = \frac{1}{2\nu} ||w||_2^2$ and $w, u \in S = \mathbb{R}^d$, then

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||\nabla \ell_t(w_t)||^2 .$$

- when there is no error the gradient is zero
- z_t is the gradient

We also know that for convex functions it is sufficient to work with linearized losses, setting $z_t = \nabla_w \ell_t(w)$:

Theorem

Consider FTRL, linear losses $\ell_t(w) = w^\top z_t$, and regularization $R(w) = \frac{1}{2\nu} ||w||_2^2$ and $w, u \in S = \mathbb{R}^d$, then

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||x_t||^2 .$$

- when there is no error the gradient is zero
- z_t is the gradient
- gradient of $w^\top x_t$ is just x_t

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||x_t||^2 .$$

- and let $R = \max ||x_t||$ (bounded features)

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||x_t||^2 .$$

- and let $R = \max ||x_t||$ (bounded features)
- use the definition of regret and the surrogate upper-bounding property

$$\#errors \leq \sum_t \ell(w_t) - \sum_t \ell(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} R^2 \cdot \#errors$$

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||x_t||^2 .$$

- and let $R = \max ||x_t||$ (bounded features)
- use the definition of regret and the surrogate upper-bounding property

$$\#errors \leq \sum_t \ell(w_t) - \sum_t \ell(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} R^2 \cdot \#errors$$

- set $\nu = \frac{||u||}{R\sqrt{\#errors}}$ and rearrange

$$\#errors - R\,||u|| \sqrt{\#errors} - \sum_{t=1}^{T} \ell_t(u) \leq 0$$

$$R_T(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} \sum_{t \in \{errors\}} ||x_t||^2 .$$

- and let $R = \max ||x_t||$ (bounded features)
- use the definition of regret and the surrogate upper-bounding property

$$\#errors \leq \sum_t \ell(w_t) - \sum_t \ell(u) \leq \frac{1}{2\nu} ||u||_2^2 + \frac{\nu}{2} R^2 \cdot \#errors$$

- set $\nu = \frac{||u||}{R\sqrt{\#errors}}$ and rearrange

$$\#errors - R ||u|| \sqrt{\#errors} - \sum_{t=1}^{T} \ell_t(u) \leq 0$$

- let's assume realizability, $\exists u^*$, s.t. $y_t = sign((u^*)^\top x_t), \forall t$

$$\#errors - R ||u^*|| \sqrt{\#errors} \leq 0$$

- if we additionaly assume, that $||u^*|| \leq 1/\gamma$ we get

$$\#errors \leq R^2/\gamma^2$$

[Novikoff, 1962]

Structured Perceptron [Collins and Roark, 2004]

input: training data $\{(x_t, y_t)\}_{t=1}^{N}$
init: $w_0 = 0$
for $t = 0, \ldots$ **do**
    observe x_t, y_t
    predict $\hat{y}_t = \arg\max_{y \in \mathcal{Y}(x)} w_t^\top \phi(x_t, y)$
    if $\hat{y}_t \neq y_t$ **then**
        update $w_{t+1} = w_t + \phi(x_t, y_t) - \phi(x_t, \hat{y}_t)$
    end if

Thm

If $||\phi(x_t, y_t)|| \leq R$ and $\exists u, ||u|| = 1$ such that $\forall t, y \in \mathcal{Y}(x_t)$:
$u^\top \phi(x_t, y_t) \geq u^\top \phi(x_t, y) + \gamma$, then

$$\# \text{ errors} \leq \frac{R^2}{\gamma^2}$$

Structured Perceptron [Collins and Roark, 2004]

input: training data $\{(x_t, y_t)\}_{t=1}^N$
init: $w_0 = 0$
for $t = 0, \ldots$ **do**
    observe x_t, y_t
    predict $\hat{y}_t = \arg\max_{y \in \mathcal{Y}(x)} w_t^\top \phi(x_t, y)$ ← this is expensive
    if $\hat{y}_t \neq y_t$ **then**
        update $w_{t+1} = w_t + \phi(x_t, y_t) - \phi(x_t, \hat{y}_t)$
    end if

Thm

If $||\phi(x_t, y_t)|| \leq R$ and $\exists u, ||u|| = 1$ such that $\forall t, y \in \mathcal{Y}(x_t)$:
$u^\top \phi(x_t, y_t) \geq u^\top \phi(x_t, y) + \gamma$, then

$$\# \text{ errors} \leq \frac{R^2}{\gamma^2}$$

Let's look again at the structured SVM model from the previous lecture

StructSVM Objective

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} \left(\max_{\mu} (w^\top F \cdot \mu + l_i^\top \mu) - w^\top F_i \cdot \mu_i \right) + \frac{\lambda}{2} ||w||^2$$

Rewriting in the same notation as perceptron:

StructSVM Objective

$$R(w) = \frac{1}{N} \sum_{i=1}^{N} \left(\max_{y \in \mathcal{Y}} (w^\top \phi(x_t, y) + \ell(y_t, y)) - w^\top \phi(x_t, y_t) \right) + \frac{\lambda}{2} ||w||^2$$

Rewriting in the same notation as perceptron:

StructSVM Objective

$$R(w) = \frac{1}{N}\sum_{i=1}^{N}\Big(\max_{y\in\mathcal{Y}}(w^\top\phi(x_t,y)+\ell(y_t,y)) - w^\top\phi(x_t,y_t)\Big) + \frac{\lambda}{2}\,||w||^2$$

- $R(w)$ is convex (sum of affine & convex functions)
- denote $\tilde{y} = \arg\max_y(w^\top\phi(x_t,y)+\ell(y_t,y))$
- can be minimized using batch subgradient method

$$\frac{\partial R}{\partial w} = \frac{1}{N}\sum_{i=1}^{N}\Big(\phi(x_t,y_t) - \phi(x_t,\hat{y})\Big) + \lambda w$$

$$w_{t+1} = w_t - \alpha_t\frac{\partial R}{\partial w}$$

$$w_{t+1} = w_t - \alpha_t(\frac{1}{N}\sum_{i=1}^{N}\Big(\phi(x_i,y_i) - \phi(x_i,\hat{y})\Big) + \lambda w_t)$$

Rewriting in the same notation as perceptron:

StructSVM Objective

$$R(w) = \frac{1}{N}\sum_{i=1}^{N}\Big(\max_{y\in\mathcal{Y}}(w^\top\phi(x_t,y)+\ell(y_t,y)) - w^\top\phi(x_t,y_t)\Big) + \frac{\lambda}{2}||w||^2$$

- $R(w)$ is convex (sum of affine & convex functions)
- denote $\tilde{y} = \arg\max_y(w^\top\phi(x_t,y)+\ell(y_t,y))$ ← this is expensive
- can be minimized using batch subgradient method

$$\frac{\partial R}{\partial w} = \frac{1}{N}\sum_{i=1}^{N}\Big(\phi(x_t,y_t) - \phi(x_t,\hat{y})\Big) + \lambda w$$

$$w_{t+1} = w_t - \alpha_t\frac{\partial R}{\partial w}$$

$$w_{t+1} = w_t - \alpha_t(\frac{1}{N}\sum_{i=1}^{N}\Big(\phi(x_i,y_i) - \phi(x_i,\hat{y})\Big) + \lambda w_t)$$

Where do we get $\tilde{y}$?

$$\tilde{y}_i = \arg\min_{y \in \mathcal{Y}} (w^\top \phi(x_i, y) - \ell(x_i, y))$$

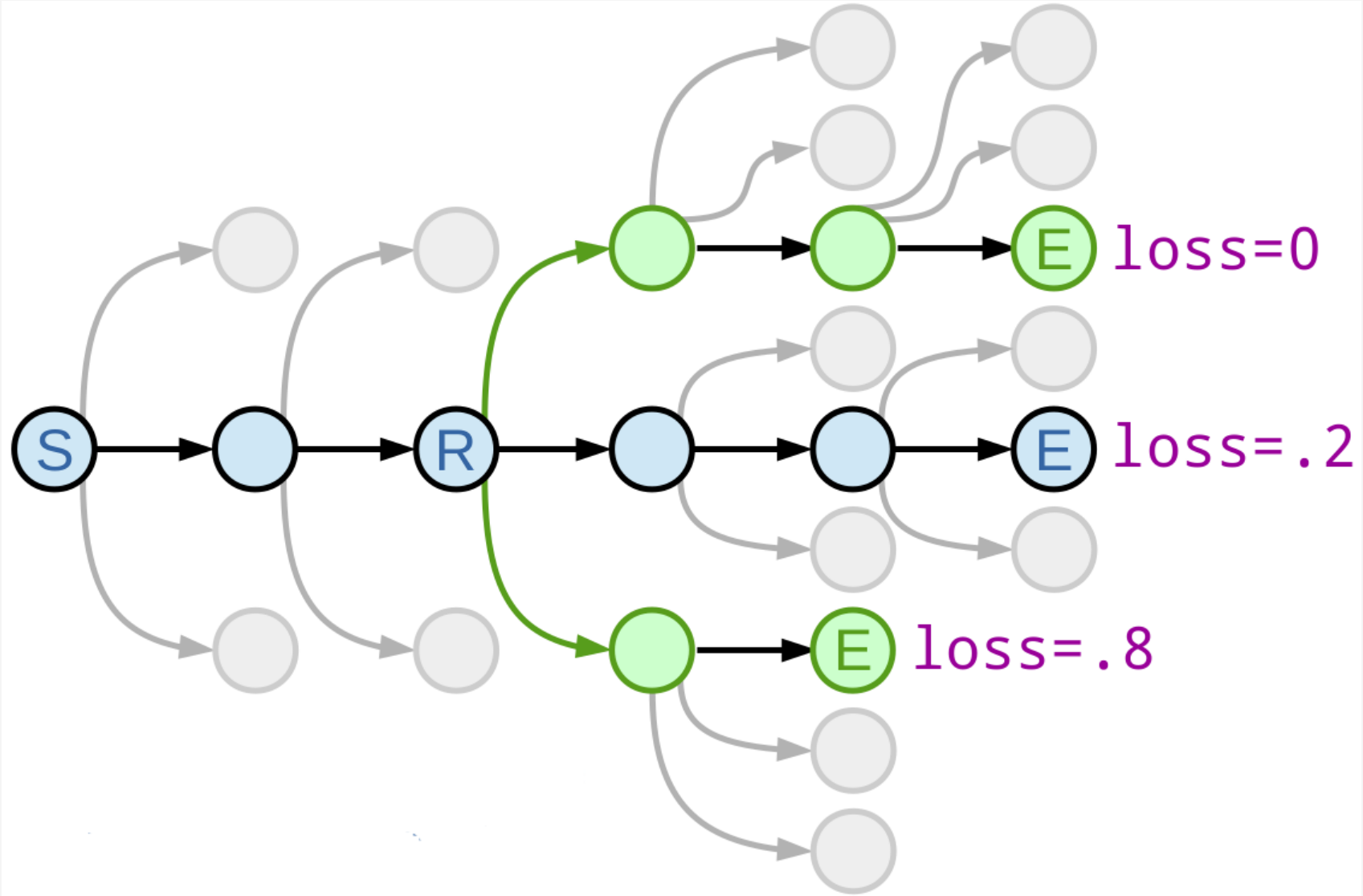
- this is the loss-augmented inference task
- requires a solution to a search problem in the underlying space
- has to be solved for every input instance (big data is a problem)
  - ➡ solvable in $O(T \cdot K)$ on chains and trees (long sequences are a problem)
  - ➡ much harder or intractable for general graphs (e.g. MRFs)

So we have two problems:

- when there is relation to the inference, the learning loop is expensive
  - ➡ the non-probabilistic models above (Perceptron or SVM)
- or inference and learning maybe not related at all
  - ➡ e.g. CRF: the learning objective knows nothing about inference
  - ➡ same for many deep learning-based models
- often finding $\tilde{y}$ alone is not enough
  - ➡ CRFs require feature expectations
  - ➡ large-margin methods require n-best lists
- Learning as Search Optimization (LaSo) is tackling these problems
  - get rid of the expensive $\arg\max$ (global decision)
  - decompose the structure building into local decisions

Setup:

- input $x \in \mathcal{X}$ induces a search space $\mathcal{Y}(x)$
- initial state b (also encodes x)
- set of states $\mathcal{S}$
- transition function $P(s_{t+1}|s_t, a_t)$ (here deterministic)
- for each (valid) sequence of states and actions, there is a corresponding output $y(e)$
- loss $\ell(e) = \ell(y^*, y(e))$, where y^* is the ground truth structure
- feature generating function $\phi : \mathcal{S} \to \mathbb{R}^d$, that expresses both the input x and previous actions
- agent follows a policy $\pi(a_t|s_t)$, which chooses an action a_t in state s_t
- trajectory $\tau_\pi = (a_1, s_1, \ldots, s_{T-1}, a_T, s_T)$, where $P(s_{t+1}|s_t, \pi(s_t))$
- T is maximum lengths of τ
- goal: $\min_\pi \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(y, \tau_\pi))]$



[Daume'15]

- for any $s \in \mathcal{S}$ and $y \in \mathcal{Y}$ we can say if s can lead to y
- in this case we call it 'y-good', otherwise 'y-bad'
- goal:
  - ➡ first node can lead to any structure
  - ➡ the queue always contains at least one y-good node

Algo Learn($problem$, $initial$, $enqueue$, $\boldsymbol{w}$, x, y)
$nodes \leftarrow$ MakeQueue(MakeNode($problem$,$initial$))
while $nodes$ is not empty **do**
  $node \leftarrow$ RemoveFront($nodes$)
  if none of $nodes \cup \{node\}$ is y-good **or**
    GoalTest($node$) and $node$ is not y-good **then**
    $sibs \leftarrow siblings(node, y)$
    $\boldsymbol{w} \leftarrow update(\boldsymbol{w}, x, sibs, node \cup nodes)$
    $nodes \leftarrow$ MakeQueue($sibs$)
  else
    if GoalTest($node$) **then return** $\boldsymbol{w}$
    $next \leftarrow$ Operators($node$)
    $nodes \leftarrow enqueue(problem, nodes, next, \boldsymbol{w})$
  end if
end while

- online learning algorithm
- search and learning are tightly intervened
- usually needs a size restriction on the queue
- similar to structured perceptron, except for early updates
- updates are done on errors (if the current beam cannot lead to the right answer)

$$w_{t+1} = w_t + \sum_{n \in sibs} \frac{\phi(x, n)}{|sibs|} - \sum_{n \in nodes} \frac{\phi(x, n)}{|nodes|}$$

Analysis:

- basically, follows the analysis for structured perceptron
- result: for separable problems with margin γ, number of errors is $\leq \frac{R^2}{\gamma^2}$
- actually it's not exactly true for subtle reasons, see [Xu and Fern, 2007]

- natural only for sequence labelling problems
- hard to apply for tasks with production in arbitrary order

Literature


Collins, M. and Roark, B. (2004).
Incremental parsing with the perceptron algorithm.
In ACL.


Novikoff, A. (1962).
On convergence proofs on perceptrons.
12:615–622.


Rosenblatt, F. (1958).
The perceptron: A probabilistic model for information storage and organization in the brain.
Psychological Review, pages 65–386.


Xu, Y. and Fern, A. (2007).
On learning linear ranking functions for beam search.
In ICML.