Imitation learning

Imitation learning for part-of-speech tagging

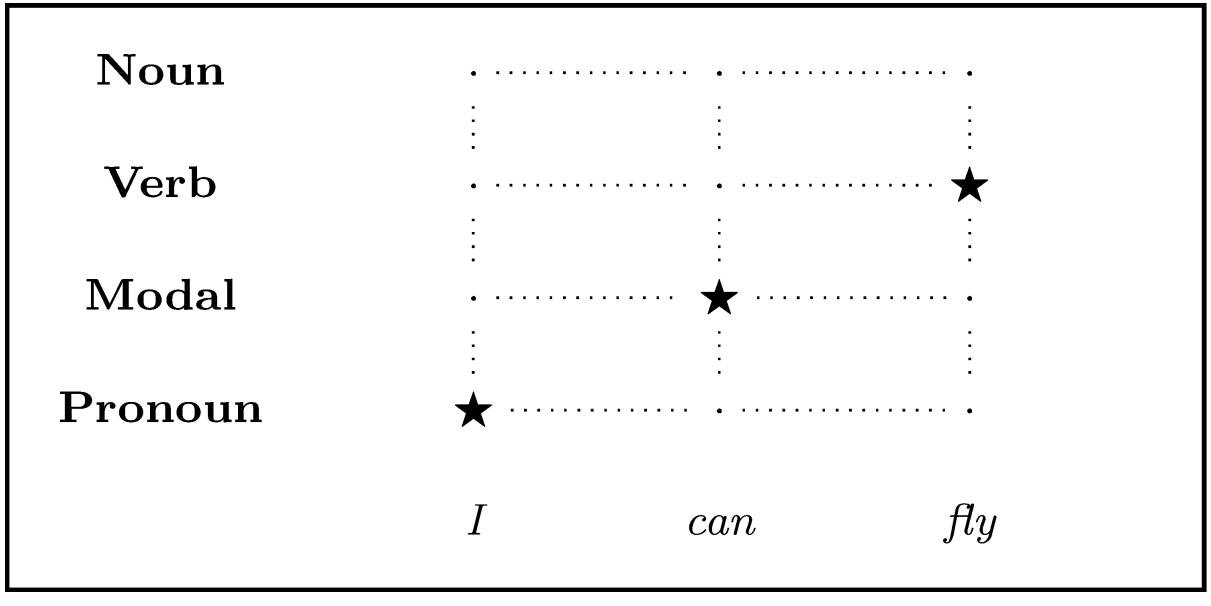
I	**can**	**fly**
Pronoun	Modal	Verb

Task loss: Hamming loss: number of incorrectly predicted tags

Transition system: Tag each token left-to-right

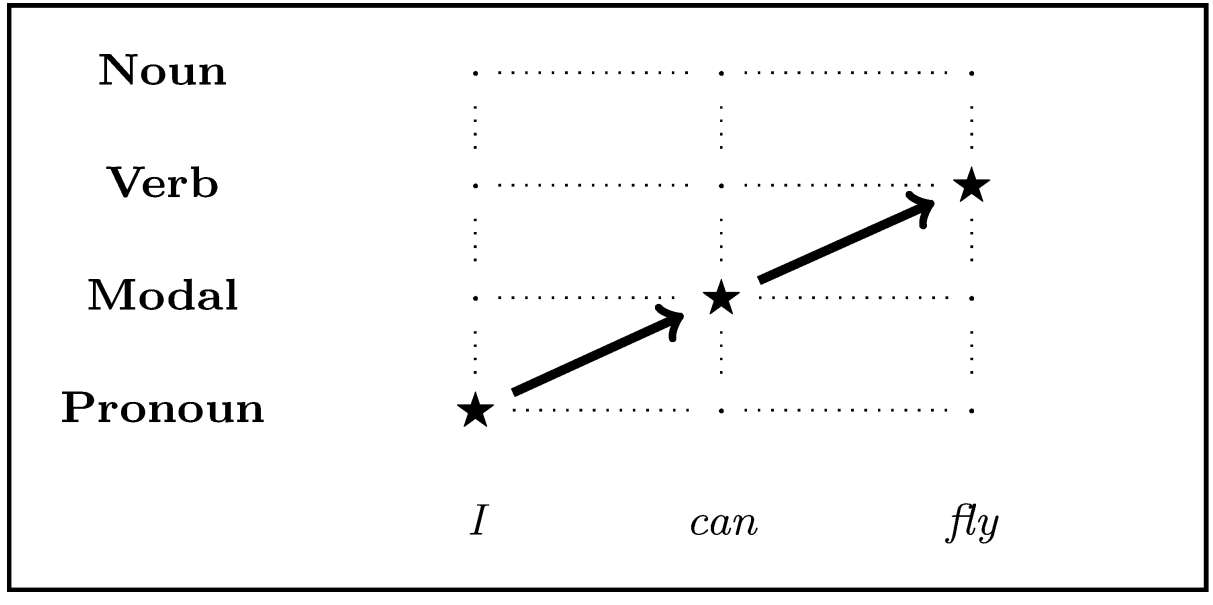
Expert policy: Return the next tag from the gold standard

Gold standard in search space



- Three actions to complete the output
- Expert policy replicates the gold standard

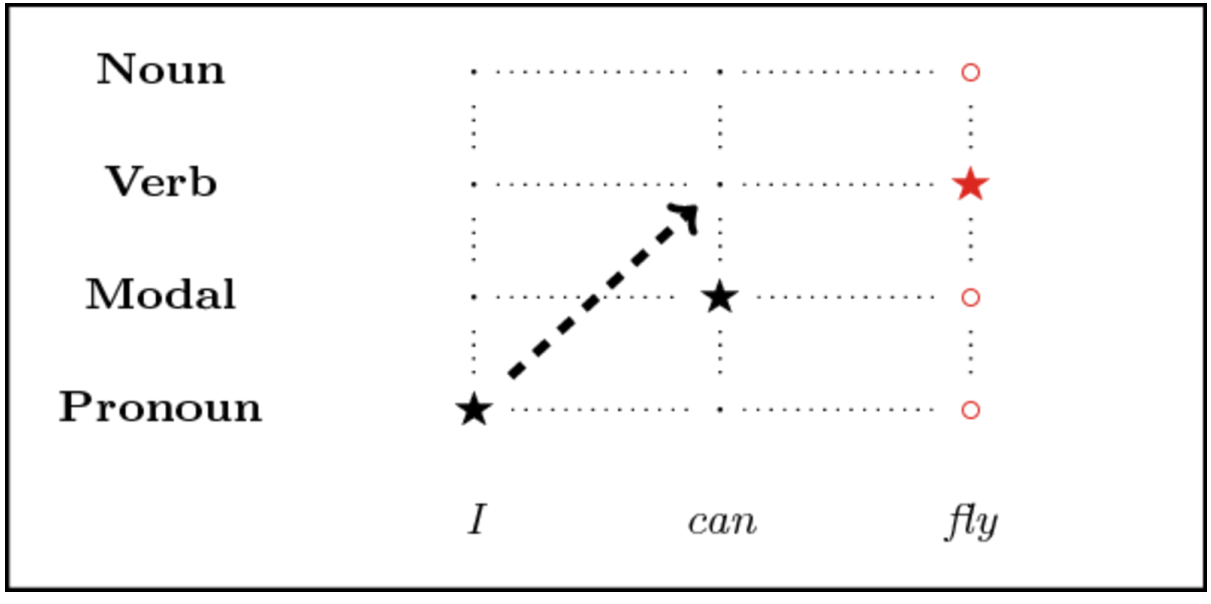
Training a classifier with structure features



label	features
Pronoun	token=I, ..., prev=**NULL**
Modal	token=can, ..., prev=**Pronoun**
Verb	token=fly, ..., prev=**Modal**

With logistic regression and k previous tags: training a kth-order Maximum Entropy Markov Model (McCallum et al., 2000 (http://people.csail.mit.edu/mcollins/6864/slides/memm.pdf))

Exposure bias



We had seen:

label	features
Verb	token=fly,..., prev=**Modal**

but not:

label	features
Verb	token=fly,..., prev=**Verb**

Addressing exposure

Allow the classifier to guide the learning

(https://www.pinterest.com/explore/affordable-driving-school/)

- 1st iteration: **roll-in** through the data with the expert
- 2nd onwards: mix expert and classifier to expose the classifier to its own actions

DAgger algorithm

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$, *expert* $\pi^\star$, *loss function* L
Output: *classifier* H
training examples $\mathcal{E} = \emptyset$, *expert probability* $\beta = 1$
while termination condition not reached **do**
  set *rollin policy* $\pi^{in} = \beta H + (1 - \beta)\pi^\star$
  for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
    rollin to predict $\hat{\alpha}_1 \ldots \hat{\alpha}_T = \pi^{in}(\mathbf{x}, \mathbf{y})$
    for $\hat{\alpha}_t \in \hat{\alpha}_1 \ldots \hat{\alpha}_T$ **do**
      ask expert for best action $\alpha^\star = \pi^\star(\mathbf{x}, S_{t-1})$
      extract $features = \phi(\mathbf{x}, S_{t-1})$
      $\mathcal{E} = \mathcal{E} \cup (features, \alpha^\star)$
  learn H from $\mathcal{E}$
  decrease β

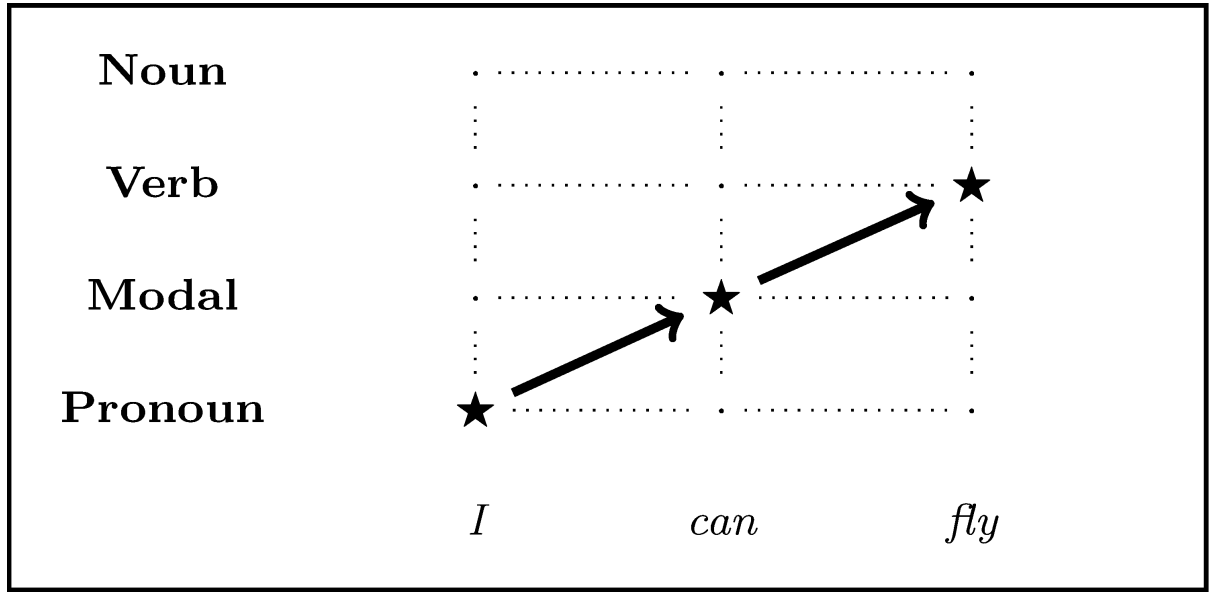
DAgger algorithm

Proposed by Ross et al. (2011) (http://www.cs.cmu.edu/~sross1/publications/Ross-AIStats11-NoRegret.pdf) motivated by robotics

- first iteration is standard classification training
- task loss and gold standard are implicitly considered via the expert
- DAgger: the Datasets in each iteration are Aggregated

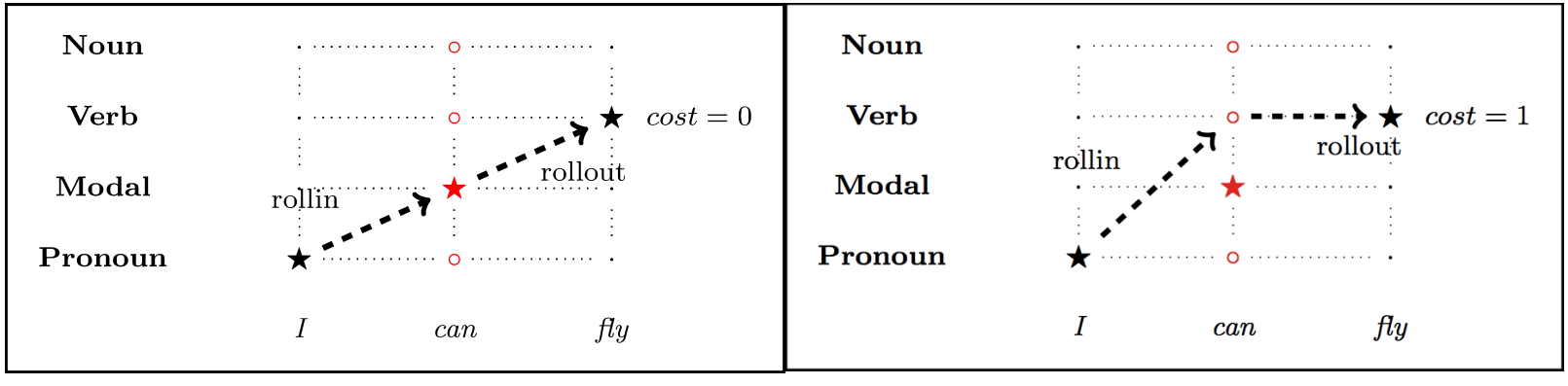
So far we looked at how to overcome previous errors. What about anticipating (and avoiding) the future ones?

Training labels as costs



Pronoun	Modal	Verb	Noun	features
0	1	1	1	token=I, prev=**NULL**...
1	0	1	1	token=can, prev=**Pronoun**...
1	1	0	1	token=fly, prev=**Modal**...

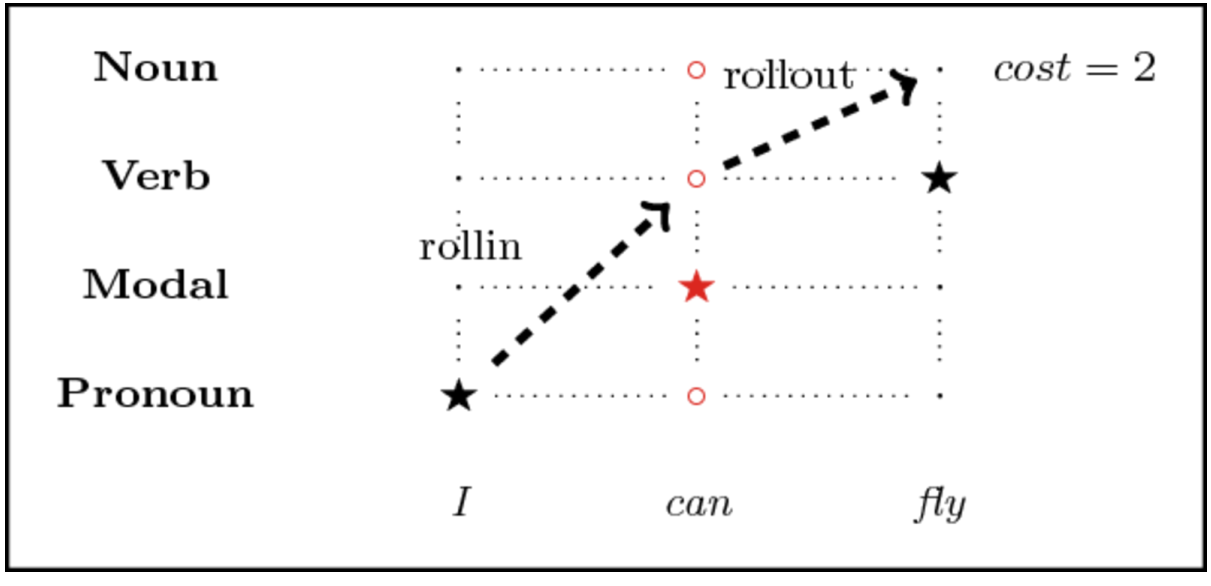
Cost break down



- **roll-in** to a point in the sentence
- try each possible label and **rollout** till the end
- evaluate the complete output with the task loss
- If **roll-out** with expert only, correct action has 0 cost, incorrect 1.

Mixed roll-outs

Rolling out with the classifier allows us to see future mistak es



Pronoun	Modal	Verb	Noun	features
1	0	2	1	token=can, prev=**Pronoun**...

DAgger with roll-outs

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$, *expert* $\pi^\star$, *loss function* L
Output: *classifier* H
training examples $\mathcal{E} = \emptyset$, *expert probability* $\beta = 1$
while termination condition not reached **do**
  set *rollin/out policy* $\pi^{in/out} = \beta H + (1 - \beta)\pi^\star$
  for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
    rollin to predict $\hat{\alpha}_1 \ldots \hat{\alpha}_T = \pi^{in/out}(\mathbf{x}, \mathbf{y})$
    for $\hat{\alpha}_t \in \hat{\alpha}_1 \ldots \hat{\alpha}_T$ **do**
      for $\alpha \in \mathcal{A}$ **do**
        rollout $S_{final} = \pi^{in/out}(S_{t-1}, \alpha, \mathbf{x})$
        cost $c_\alpha = L(S_{final}, \mathbf{y})$
      extract $features = \phi(\mathbf{x}, S_{t-1})$
      $\mathcal{E} = \mathcal{E} \cup (features, \mathbf{c})$
  learn H from $\mathcal{E}$
  decrease β

Roll-outs

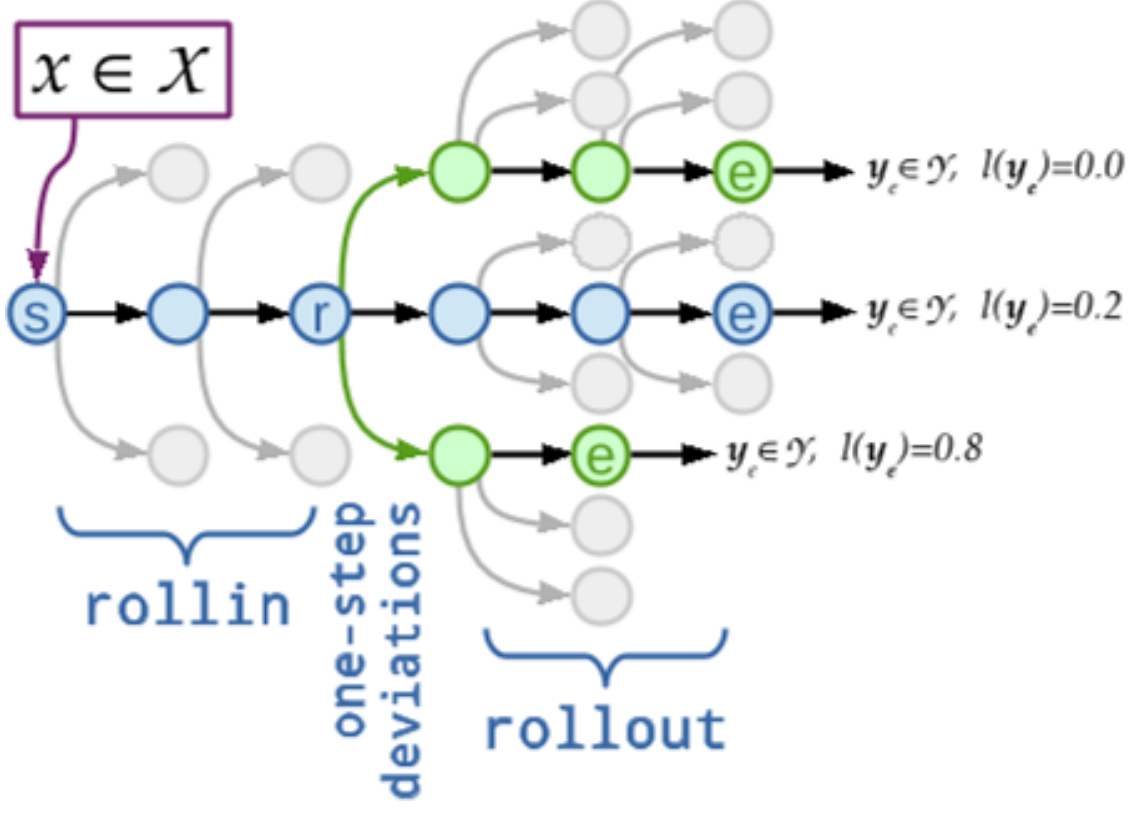
- can learn with non-decomposable losses
- can learn with sub-optimal experts
- expensive when there are many actions and long sequences to complete outputs

Some history:

- first proposed in SEARN (Daumé III et al., 2009 (http://hunch.net/~jl/projects/reductions/searn/searn.pdf))
- used to hybridise DAgger by Vlachos and Clark (http://www.aclweb.org/anthology/Q14-1042), referred to later as V-DAgger (Goodman et al. 2016 (http://aclweb.org/anthology/P16-1001))
- also proposed as look-aheads (Tsuruoka et al. 2011 (http://www.anthology.aclweb.org/W/W11/W11-0328.pdf))

LoLS

Locally Optimal Learning to Search (Chang et al., 2015 (https://arxiv.org/pdf/1502.02206.pdf))



- rollin always with the classifier
- each rollout uses either the expert or the classifier exclusively

Generic imitation learning

Input: $D_{train} = \{(\mathbf{x}^1, \mathbf{y}^1) \ldots (\mathbf{x}^M, \mathbf{y}^M)\}$, *expert* $\pi^\star$, *loss function* L
Output: *classifier* H
training examples $\mathcal{E} = \emptyset$
while termination condition not reached **do**
  set *rollin policy* $\pi^{in} = mix(H, \pi^\star)$
  set *rollout policy* $\pi^{out} = mix(H, \pi^\star)$
  for $(\mathbf{x}, \mathbf{y}) \in D_{train}$ **do**
    rollin to predict $\hat{\alpha}_1 \ldots \hat{\alpha}_T = \pi^{in}(\mathbf{x}, \mathbf{y})$
    for $\hat{\alpha}_t \in \hat{\alpha}_1 \ldots \hat{\alpha}_T$ **do**
      rollout to obtain costs c for all possible actions using L
      extract features $f = \phi(\mathbf{x}, S_{t-1})$
      $\mathcal{E} = \mathcal{E} \cup (f, c)$
  learn H from $\mathcal{E}$

Summary so far

- basic intuition behind IL
- rollin and the DAgger algorithm
- rollouts and V-DAgger and LoLS
- generic imitation learning recipe