Learning to Search for Structured Prediction

Kovalenko Boris

CS NRU HSE

Bayesian Methods in Machine Learning (seminar)

Talk outline

1. Introduction
2. Imitation learning and Structured prediction with DAgger
3. Structured prediction with Learning to Search / Searn
4. Learning to Search / Searn example

Introduction

Invent new algorithm for every problem, which "standard" algorithms do not support? $\rightarrow$ Code this algorithm from scratch.

or

Create meta algorithm that "reduces" new problem to standard problem? $\rightarrow$ Reuse existing state of the art implementations of base algorithms to implement your meta algorithm.

Introduction

Examples of reduction:

1. Multiclass classification $\rightarrow$ Binary Classification
2. Importance weighted classification $\rightarrow$ Binary Classification
3. Ranking $\rightarrow$ Classification
4. **Structured prediction $\rightarrow$ Binary Classification**
5. *More reductions here*

Imitation learning and Structured prediction with DAgger

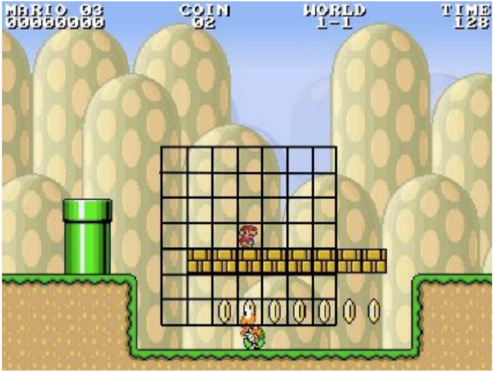

Figure: Super Mario Bros / 2009 AI competition

$\{0, 1\}$ actions: Jump, Right, Left, Speed. *Video*

Imitation learning and Structured prediction with DAgger

Core definitions:

1. Π - class of policies learner is considering
2. T - task horizon
3. d_π^t - distribution of states if learner executed π from step 1 to t-1. Average distribution of states - $d_\pi = \frac{1}{T}\sum_{t=1}^{T} d_\pi^t$
4. $C(s,a) \in [0,1]$ - immediate cost of performing action a at state s.
5. $C_\pi(s) = \mathbb{E}_{a\sim\pi}[C(s,a)]$ - expected immediate cost π in s
6. $J(\pi) = \sum_{t=1}^{T} \mathbb{E}_{s\sim d_\pi^t}[C_\pi(s)] = T\mathbb{E}_{s\sim d_\pi}[C_\pi(s)]$ - total cost of executing π for T steps
7. $\ell(s,\pi)$ - observed surrogate loss of π with respect to π^*

goal:

$$\hat{\pi} = \operatorname*{argmin}_{\pi\in\Pi} \mathbb{E}_{s\sim d_\pi}[\ell(s,\pi)]$$

Imitation learning and Structured prediction with DAgger

Vanilla supervised approach:

1. Collect states encountered by expert d_{π^*}
2. Extract features for each state, along with expert decision
3. Fit supervised model to get policy

$$\hat{\pi}_{sup} = \operatorname*{argmin}_{\pi \in \Pi} \mathbb{E}_{s \sim d_{\pi^*}}[\ell(s, \pi)]$$

Assuming $\ell(s, \pi)$ is $0 - 1$ loss (or upper bound on the $0 - 1$ loss), the following performance guarantees obtained, with respect to any cost function C, bounded in $[0, 1]$

Theorem 1 (Ross and Bagnel, 2010):
Let $\mathbb{E}_{s \sim d_{\pi^*}}[l(s, \pi)] = \epsilon$, then $J(\pi) \leq J(\pi^*) + T^2\epsilon$.

Imitation learning and Structured prediction with DAgger

Dataset Aggregation algorithm:

Initialize $D \leftarrow \emptyset$
Initialise $\hat{\pi}_i$ to any policy in Π
for $i = 1$ *to* N *do*
Let $\pi_i = \beta_i \pi^* + (1 - \beta_i)\hat{\pi}_i$
Sample T step trajectories using π_i
Get dataset $D_i = \{(s, \pi^*(s))\}$ of visited states by π_i
and actions given by expert
Aggregate datasets: $D \leftarrow D \bigcup D_i$
Train classifier $\hat{\pi}_{i+1}$ on D
end for
Return best $\hat{\pi}_i$ on validation

requirement: $\beta_N = \frac{1}{N} \sum_i^N \beta_i \rightarrow 0$ *as* $N \rightarrow 0$

Imitation learning and Structured prediction with DAgger

Performance guaranties for the algorithm:

$\pi_{1..N}$ - sequence of policies
Assume l is strongly convex and bounded over Π
Let $\beta_i \leq (1-\alpha)^{i-1}$ for all i, for some constant α independent of T
Let $\epsilon = min_{\pi \in \Pi} \frac{1}{N} \sum_{i=1}^{N} \mathbb{E}_{s \sim d_{\pi_i}} [\ell(s, \pi)]$ - true loss of best policy
Number of trajectories is infinite for each iteration

Then best policy π in sequence $\pi_{1..N}$ guarantees:

$$J(\pi) \leq T(\epsilon_N + \gamma_N) + O(\frac{T}{N})$$

for strongly convex loss and $(N = TlogT)$:

$$J(\pi) \leq T(\epsilon_N) + O(1)$$

Imitation learning and Structured prediction with DAgger

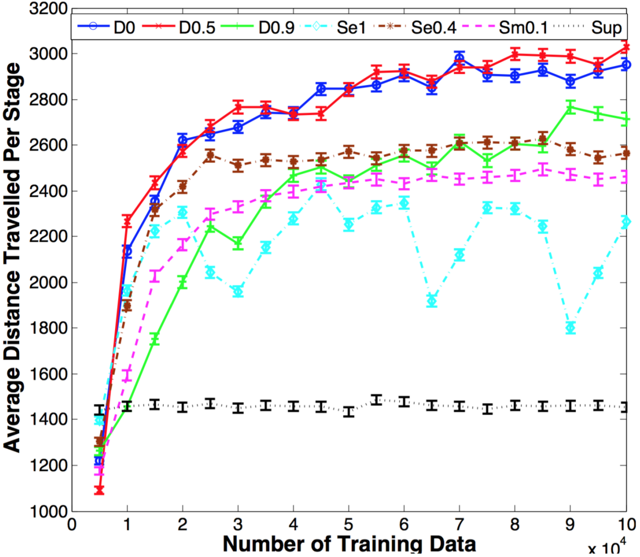

Figure: Average travelled distance as a function of data

Imitation learning and Structured prediction with DAgger

- Simple iterative meta algorithm
- Notion of states space exploration for states which learned policy may encounter on test set
- Good bound on error growth with time
- Can be applied to structured prediction problems

Structured prediction with Learning to Search

Core definitions:

Structured prediction problem D - cost sensitive classification problem where $\mathbb{Y}$ has structure and $y \in \mathbb{Y}$ decompose into variable length vectors $(y_1, y_2, ..., y_T)$. D is a distribution over inputs $x \in \mathbb{X}$ and cost vectors c, where $|c|$ is a variable in 2^T.

Goal - find $h : \mathbb{X} \to \mathbb{Y}$, which minimizes loss:

$$L(D, h) = \mathbb{E}_{(x,c)\sim D}[c_{h(x)}]$$

Assumption: $y \in \mathbb{Y}$ can be produced, by predicting each component $(y_1, y_2, ..., y_T)$ in turn

Structured prediction with Learning to Search

Core definitions:

Search space - space of all possible vectors $y \in \mathbb{Y}$, which is explored in an iterative fashion. Example: part of speech of each individual word in a sentence.
Cost sensitive learning algorithm - multi class cost sensitive classifier. Can be reduced to binary classification (Beygelzimer et al. 2005)
Known loss function - must be computable for any sequence of predictions
Good initial policy - should achieve low loss on training data.

Structured prediction with Learning to Search / Searn

The idea of L2S:

- At each iteration it uses known policy to create new cost sensitive classification examples
- These are used to fit new classifier, which is interpreted as new policy
- New policy is interpolated with the old policy and the process repeats

Structured prediction with Learning to Search / Searn

Algorithm (S^{SP}, π, A):

Initialize $h \leftarrow \pi$

while h has dependence on π do:

  initialize the set of cost-sensitive examples $S \leftarrow \emptyset$

    for $(x, y) \in S^{SP}$ do:

      Compute predictions under current policy $\hat{y} \sim x, h$

      for $t = 1 ... T_x$ do

        Compute features $\Phi = \Phi(s_t)$ for $s_t = (x, y_1, ..., y_t)$

        Initialise a cost vector $\mathbb{c}=<>$

        for each possible action a do:

          Let cost l_a for example x,c at state s be $l_h(c, a, a)$

        end for

        add cost sensitive example (Φ, l) to S

      end for

    end for

  Fit classifier on S: $h \leftarrow A(S)$

  Interpolate $h \leftarrow \beta h' + (1 - \beta) h$

end while

Structured prediction with Learning to Search / Searn

Obtaining cost sensitive examples:
Current policy is running on each training example. For each state one cost sensitive example is created. Cost (or regret) for action is calculated by running policy to the final state and subtracting minimum loss:

$$\ell(c, s, a) = \mathbb{E}_{\hat{y} \sim (s,a,h)} c_{\hat{y}} - \operatorname*{argmin}_{a`} \mathbb{E}_{\hat{y} \sim (s,a`,h)} c_{\hat{y}}$$

Computing features:
Step is arbitrary, however the performance of classification algorithm depends on good choice of features. The feature vector $\Phi(s_t)$ may depend on any aspect of the input x and any past decision.

Structured prediction with Learning to Search / Searn

Lemma 1 (Policy degradation):

Given a policy h with loss $L(D, h)$, apply a single iteration of Searn to learn a classifier h with cost-sensitive loss $\ell_h^{CS}(h)$. Create a new policy h^{new} by acting according to h with probability $\beta \in (0, \frac{1}{T})$ and otherwise acting according to h at each step. Then, for all D, with $c_{max} = \mathbb{E}_{(x,c)\sim D} max_i c_i$:

$$L(D, h^{new}) \leq L(D, h) + T\beta\ell_h^{CS}(h) + \frac{1}{2}\beta^2 T^2 c_{max}$$

Lemma 2 (Iteration):

For all D, for all learned h, after $\frac{C}{\beta}$ iterations of Searn beginning with a policy π with loss $L(D, \pi)$, and average learned losses, the loss of the final learned policy h (without the optimal policy component) is bounded by:

$$L(D, h_{last}) \leq L(D, \pi) + CT\ell_{avg} + c_{max}(\frac{1}{2}CT^2\beta + Texp[-C])$$

Structured prediction with Learning to Search / Searn

Theorem 2 (Loss bound):
For all D with $c_{max} = \mathbb{E}_{(x,c)\sim D} max_y c_y(with(x,c)$, for all learned cost sensitive classifiers h , Searn with $\beta = \frac{1}{T^3}$ and $2T^3 lnT$ iterations, outputs a learned policy with loss bounded by:

$$L(D, h_{last}) \leq L(D, \pi) + 2T\ell_{avg} lnT + (1 + lnT)\frac{c_{max}}{T}$$

Structured prediction with Learning to Search / Searn

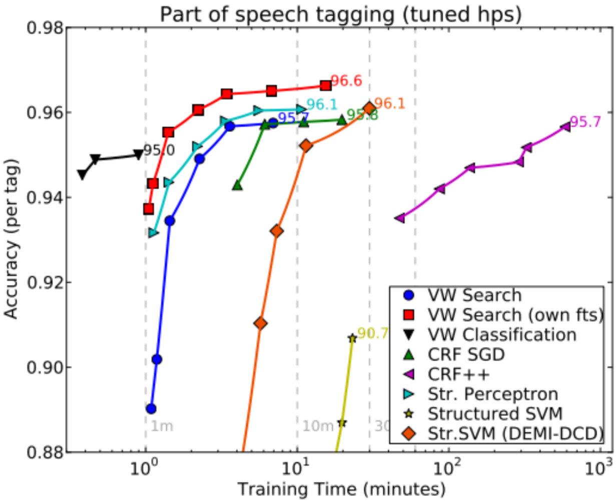

Figure: POS Algorithms comparison

Structured prediction with Learning to Search / Searn

Part of speech tagging, VW code example:

In [20]:
```
DET  = 1
NOUN = 2
VERB = 3
ADJ  = 4
my_dataset = [ [(DET , 'the'),
                (NOUN, 'monster'),
                (VERB, 'ate'),
                (DET , 'a'),
                (ADJ , 'big'),
                (NOUN, 'sandwich')],
               [(DET , 'the'),
                (NOUN, 'sandwich'),
                (VERB, 'was'),
                (ADJ , 'tasty')],
               [(NOUN, 'it'),
                (VERB, 'ate'),
                (NOUN, 'it'),
                (ADJ , 'all')] ]
print my_dataset[2]
```

Figure: Toy dataset

Structured prediction with Learning to Search / Searn

Part of speech tagging, VW code example:

```
In [21]: class SequenceLabeler(pyvw.SearchTask):
             def __init__(self, vw, sch, num_actions):
                 # you must must must initialize the parent class
                 # this will automatically store self.sch <- sch, self.vw <- vw
                 pyvw.SearchTask.__init__(self, vw, sch, num_actions)

                 # set whatever options you want
                 sch.set_options( sch.AUTO_HAMMING_LOSS | sch.AUTO_CONDITION_FEATURES )

             def _run(self, sentence):   # it's called _run to remind you that you shouldn't call it directly!
                 output = []
                 for n in range(len(sentence)):
                     pos,word = sentence[n]
                     # use "with...as..." to guarantee that the example is finished properly
                     with self.vw.example({'w': [word]}) as ex:
                         pred = self.sch.predict(examples=ex, my_tag=n+1, oracle=pos, condition=[(n,'p'), (n-1,
                         output.append(pred)
                 return output
```

Figure: POS code using VW

More examples here

References

1. A Reduction of Imitation Learning and Structured Prediction to No-Regret Online Learning. Link Video
2. Search-based structured prediction Link Video
3. Machine learning reductions Link Video
4. Recent developments Link