Search-based Structured Prediction

Hal Daume III, John Langford, Daniel Marcu
Machine Learning Journal, 2009

Presented By : Sudip & Michael

Motivation

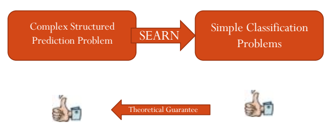
- Complex Structured Prediction Problems
  - Natural Language Processing
  - Speech
  - Computational Biology
  - Vision
- Current Algorithms
  - Decomposition of loss function
  - Decomposition of feature functions

What SEARN can do ?

- Structured Prediction Algorithm
- Not limited to bounded tree-width ??
- Applicable to any loss function
- Can Handle Arbitrary Features
- Can cope with imperfect data

SEARN - Overview

- Integrating SEARch and lEARNing
- Meta-algorithm



Definition: Structured Prediction Problem

- **Definition:** A ***structured prediction problem*** *D is a cost-sensitive classification problem where Y has structure: elements y* $\in$ *Y decomposes into variable-length vectors* $(y_1, \ldots, y_T)$. *D is a distribution over inputs x* $\in$ *X and cost vectors c, where |c| is a variable is* 2^T.
- Example: Parsing Problem under F1-loss
  - x = input sequence
  - y = parse tree of x
  - D = distribution over (x,**c**) where, c_y is the F1 loss of y on "true" output
  - |c| = number of trees with |x|-many leaves
- Goal : find $h : X \to Y$
  - Minimize loss $L(D,h) = E_{(x,c)\sim D}\{c_{h(x)}\}$
- SEARN : $y \in Y$ can be produced by predicting each component $(y_1, y_2, \ldots, y_T)$

SEARN algorithm ingredients

Ingredients	Purpose
Search Space	Decomposing the prediction problem
Cost-sensitive Learning Algorithm	Return multiclass classifier h(s) given cost sensitive training data
Labeled Structured Prediction Training data	SEARN converts them into cost-sensitive training data
Loss function	Used to calculate "regret" for an action
Good Initial Policy	Starting point of iteration

Background

- Reinforcement Learning
- Cost Sensitive Training Data
- Multi-class Cost-sensitive Learner

Background Concepts

- Reinforcement Learning
  - set of states S
  - set of actions A
  - set of scalar rewards

- Find "policy" that maps states to actions such that performing the action results in maximum reward

Cost Sensitive Classification

- Cost sensitive example - each input sample is associated with costs
- Notation: $(x, \mathbf{c})$

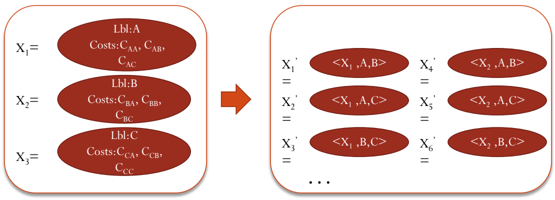
where $\mathbf{c} = < c_1, \ldots, c_k >$ and c_i is the cost of labeling x with class I

- Thus we are looking for a function minimizing

$$h_D = E_{(x,\mathbf{c}) \sim D}\{c_{h(x)}\}$$

Reduction to Binary Classification (Weighted All Pairs)

Step 1: Create new training set, including all pairs of classes



Each Weighted Pair contains information regarding which class is better

Step 2: Learn binary classifier h(<x,i,j>) on new training set

Reduction to Binary Classification

- **Step 3**
  - For a test example x, we say that class i is better than class j if $i < j$ and $h(<x,i,j>) = 1$ or $i > j$ and $h(<x,i,j>) = 0$
- **Step 4**
  - Calculate relation (i beats j) for all (i,j)
  - Use as prediction:

$$h_{new}(x) = \arg\max_i |\{ j \mid i \text{ beats } j \}|$$

Reduction to Binary Classification (Complete Algorithm)

- Define

$$L(t) = |\{j \mid k_j \leq t\}| \quad v_i = \int_0^{k_i} 1/L(t)\,dt.$$

1 WAP-Train (Set of r-class cost sensitive examples S, importance weighed binary classifier learning algorithm B)

Set $S' = \emptyset$.
for all examples $(x, k_1, \ldots, k_r)$ in S **do**
  for all pairs (i, j) with $1 \leq i < j \leq r$ **do**
    Add an importance weighted example $(\langle x, i, j\rangle, I(k_i < k_j), |v_j - v_i|)$ to S'.
  end for
end for
Return $h = B(S')$.

2 WAP-Test (classifier h, example x)

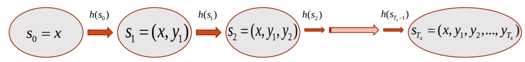
for all pairs (i, j) with $1 \leq i < j \leq r$ **do**
  Evaluate $h(\langle x, i, j\rangle)$
end for
Output $\text{argmax}_i |\{j \mid i \text{ beats } j\}|$.

Error Rate of WAP

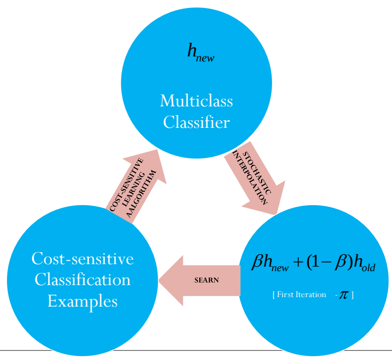
Theorem 2.3 (WAP error efficiency; (Beygelzimer et al., 2005)). *For all cost-sensitive problems $\mathcal{D}$, if the base importance weighted classifier has loss rate c, then WAP has loss rate at most $2c$.*

- Error Rate of WAP is bounded by the error of the binary classifier

SEARN - Testing



SEARN - Training



Complete SEARN Algorithm

Algorithm SEARN(S^{SP}, π^*, Learn)

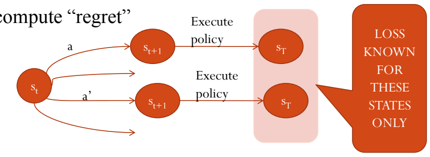
```
Initialize policy h^(0) ← π*
for I = 1 ... do
  Initialize the set of cost-sensitive examples S_I ← ∅
  for n = 1 ... N do
    Compute path under the current policy ⟨s_1, ..., s_{T_n}⟩ ← pth(x_n, h^(I−1), ∅)
    for t = 1 ... T_n do
      Compute features Φ = Φ(x_n, s_t) for input x_n and state s_t
      Initialize a cost vector c = ⟨⟩
      for each possible action a do
        Compute the cost of a: ℓ_a = ℓ^{h^(I−1)}_{s_t ⊕ a} (Eq (3.2))
        Append ℓ to c: c ← c ⊕ ℓ_a
      end for
      Add cost-sensitive example (Φ, c) to S_I
    end for
  end for
  Learn a classifier on S_I: h' ← Learn(S_I)
  Interpolate: h^(I) ← βh' + (1 − β)h^(I−1)
end for
return h^(last) without π*
```

Feature Computations

- Compute feature vector Φ on the basis of state s_t
- Good choice of features => Good Prediction
- $\Phi(s_t)$ may depend on any aspect of input x and any past decision
- Example : Part of speech tagging
  - $\Phi(s_t)$: zeros everywhere except for "interesting" aspects of input (such as features corresponding to x_{t+1} and y_t)

Construction of Cost-sensitive Examples

- Use policy *h* to construct cost-sensitive multiclass classification examples
- One path per structured training example
- Single cost-sensitive example for each state on each path
- Use loss to compute "regret"



$$\ell_h(c, s, a) = \mathbb{E}_{\hat{y} \sim (s,a,h)} c_{\hat{y}} - \min_{a'} \mathbb{E}_{\hat{y} \sim (s,a',h)} c_{\hat{y}}$$

Initial Policy

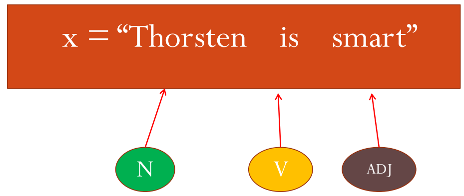
- *Definition*: For an input x and a cost vector **c**, and a state $s = x \times (y_1, y_2, \ldots, y_t)$ in the search space, the initial policy π(s,c) is

$$\arg\min_{y_{t+1}} \min_{y_{t+2},\ldots,y_T} c_{<y_1,\ldots,y_T>}$$

- That is, π chooses the action (i.e., value for y_{t+1}) that minimizes the corresponding cost, assuming that all future decisions are also made optimally.
- Requirements:
  - "Good" (not necessarily optimal)
  - Efficiently Computable

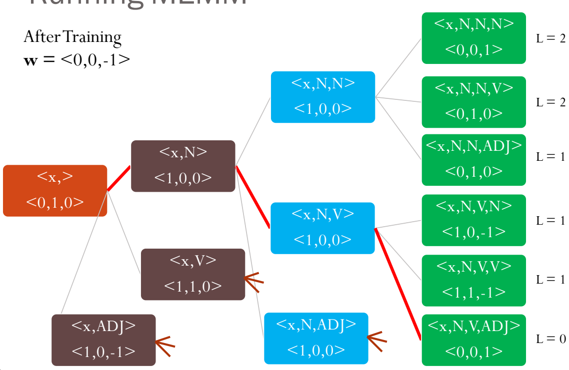
Running Example

- Part of Speech Tagging
- Single Training Point



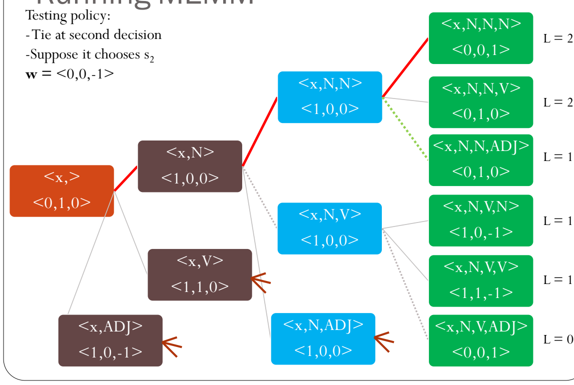
Running MEMM

After Training
w = <0,0,-1>



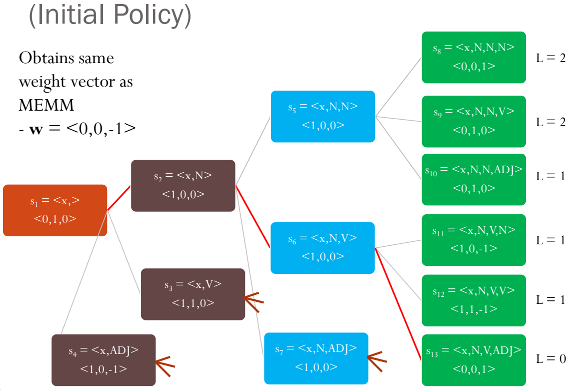
Running MEMM

Testing policy:
-Tie at second decision
-Suppose it chooses s_2
w = <0,0,-1>



Running SEARN (Initial Policy)

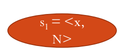
Obtains same weight vector as MEMM
- **w** = <0,0,-1>



Running SEARN Cost-sensitive Examples

- For each component of the prediction, add a cost sensitive example
- Cost is calculated by

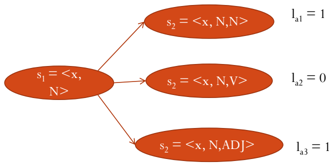
$$l_a^{\pi} = E_{y \sim search(x_n,\pi,a)} c_y - \min_{a'} l_{a'}^{\pi}$$



Running SEARN
Cost-sensitive Examples

- For each component of the prediction, add a cost sensitive example to S_i
- Cost is calculated by

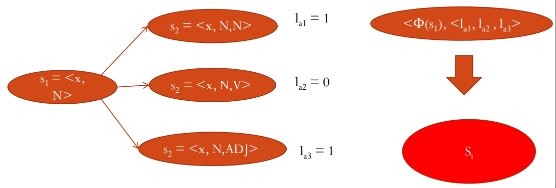
$$l_a^{\pi} = E_{y \sim search(x_n,\pi,a)} c_y - \min_{a'} l_{a'}^{\pi}$$



Running SEARN
Cost-sensitive Examples

- For each component of the prediction, add a cost sensitive example to S_i
- Cost is calculated by

$$l_a^{\pi} = E_{y \sim search(x_n,\pi,a)} c_y - \min_{a'} l_{a'}^{\pi}$$



Running SEARN
Cost-sensitive Examples

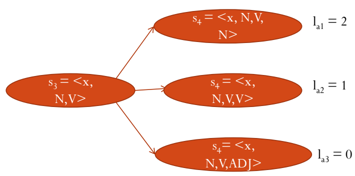
- For each component of the prediction, add a cost sensitive example to S_i
- Cost is calculated by

$$l_a^{\pi} = E_{y \sim search(x_n,\pi,a)} c_y - \min_{a'} l_{a'}^{\pi}$$

Running SEARN
Cost-sensitive Examples

- For each component of the prediction, add a cost sensitive example to S_i
- Cost is calculated by

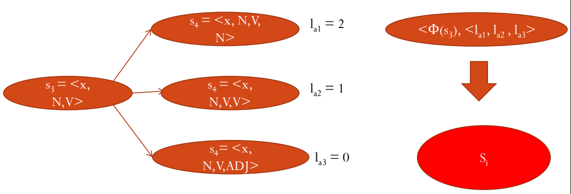
$$l_a^{\pi} = E_{y \sim search(x_n,\pi,a)} c_y - \min_{a'} l_{a'}^{\pi}$$



Running SEARN
Cost-sensitive Examples

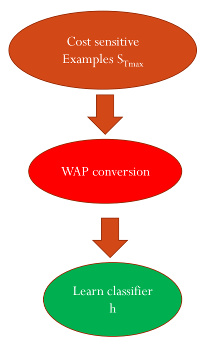
- For each component of the prediction, add a cost sensitive example to S_i
- Cost is calculated by

$$l_a^{\pi} = E_{y \sim search(x_n,\pi,a)} c_y - \min_{a'} l_{a'}^{\pi}$$



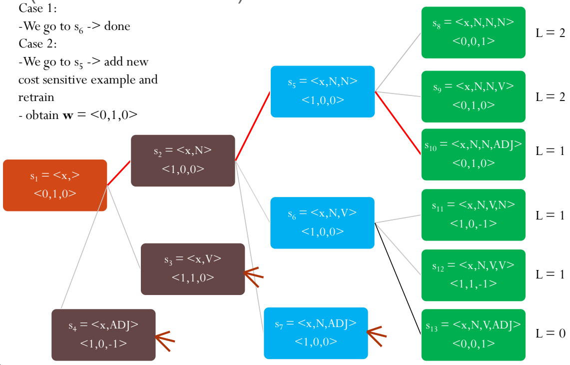
Running SEARN
Binary Classifier

- Take set of cost sensitive examples
- Use Weighted All Pairs to reduce problem into binary classification
- Output learned classifier h, set as new policy



Running SEARN (second iteration)

Case 1:
-We go to s_6 -> done
Case 2:
-We go to s_5 -> add new cost sensitive example and retrain
- obtain **w** = <0,1,0>



MEMM vs. SEARN

- MEMM
  - Expected Loss is much higher than it should be
  - Weight vector is only trained on optimal paths
- SEARN
  - Starts with same weight vector
  - SEARN does additional training
  - also goes through suboptimal paths

Computability of the Initial Policy

- Simple Problems under standard loss function => Good policy in constant time
  - Sequence Labeling
- SEARN can learn under strictly more complex structures and loss functions than "other techniques"
  - Comparison to M3N (apparently most powerful generic framework)
  - Loss augmented minimization:
    - (*)

$$opt(\mathcal{Y}_x, y, \boldsymbol{w}) = \arg\max_{\hat{y}\in\mathcal{Y}_x} \boldsymbol{w}^\top \Phi(x,\hat{y}) - l(y,\hat{y})$$

  - If (*) is computable in time T(x); then the optimal policy is computable in time O(T(x)). Further, there exist problems for which the optimal policy is computable in constant time and for which (*) is an NP-hard computation
    - Intuition: Keep increasing Markov Order
- Search Based Policies
  - No optimality requirement => Use search to create initial policy
  - Instead of a good policy we now require efficient approximate search

Theoretical Analysis

- *Away* from initial policy, *Toward* a fully learned policy
- Each iteration *degrades* current policy
- Learned policy not much worse than
  - Starting policy + average cost-sensitive loss + f(maximum cost sensitive loss)

Theorem 2 *For all $\mathcal{D}$ with $c_{max} = \mathbb{E}_{(x,\mathbf{c})\sim\mathcal{D}} \max_{\mathbf{y}} c_{\mathbf{y}}$ (with $(x,\mathbf{c})$ as in Def 1), for all learned cost sensitive classifiers h',* SEARN *with $\beta = 1/T^3$ and $2T^3 \ln T$ iterations, outputs a learned policy with loss bounded by:*

$$L(\mathcal{D}, h_{last}) \le L(\mathcal{D}, \pi) + 2T\ell_{avg} \ln T + (1+\ln T)c_{max}/T$$

Comparing Alternative Techniques (arg max)

- Attempts to solve :

$$\hat{y} = \arg\max_{y\in Y_x} F(y \mid x, \theta)$$

- Tractable only for certain structured problems
- Difficult ones boil down to NP-hard search problems
- Inspired the modeling of search in SEARN

Comparing Alternative Techniques (perceptron style)

- Learn a weight vector by updating

$$w \longleftarrow w + \Phi(x_n, y_n) - \Phi(x_n, \hat{y}_n)$$

- Essentially a search based structured prediction
- Cannot handle different loss functions
- Pro: Efficient, easy to implement
- Con: Cannot handle different loss functions

Comparing Alternative Techniques (global prediction)

- CRF, M^3N
- Models are limited to linear chains with Markov features
- SEARN can be used for more general model features with weaker assumptions
- Information is shared at test time (Viterbi), in SEARN it is shared at training.
- Adv: Large margin principle, tractible on more problems
- Con: slow. Limited to Hamming Loss

Comparing Alternative Techniques (SVM-struct)

- Also need to compute loss-augmented search problem
- Allows non decomposable loss functions

$$S(x,y) = \arg\max_{\hat{y}\in\mathcal{Y}} \left[\boldsymbol{w}^\top \Phi(x,\hat{y})\right] l(x,y,\hat{y})$$

- Pro: More loss functions available, maximizes margin, Often maximum constraint cannot be found
- Con: Intractable loss-augmented search procedure

Comparing Alternative Techniques (MEMM)

- Uses "state given observation" probabilities

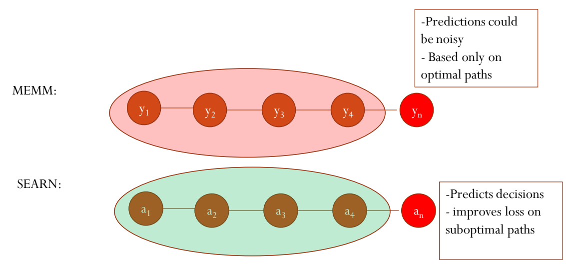
$$p\left(y_n \mid x, y_{n-1}; \boldsymbol{w}\right) = \frac{1}{Z_{x,y_{n-1};\boldsymbol{w}}} \exp\left[\boldsymbol{w}^\top \Phi(x, y_n, y_{n-1})\right]$$

$$Z_{x,y_{n-1};\boldsymbol{w}} = \sum_{y'\in\mathcal{Y}^n} \exp\left[\boldsymbol{w}^\top \Phi(x, y', y_{n-1})\right]$$

- Traces true output sequences, using true y_{n-1} labels to generate training examples

Comparing Alternative Techniques (MEMM)

- SEARN is most similar to MEMM prediction setting



Summary of Algorithms

	Loss			Features			Efficient	Easy to Implement
	0/1	Hamming	Any	argmax and sum	argmax only	Neither		
Structured Perceptron	√				√		√	√
Conditional Random Field	√			√				–
Max-margin Markov Network	√	√			√			
SVM for Structured Outputs	√	√			√			
Reranking	√	√	√			√	–	–

Relation to Reinforcement Learning

- Can map structured prediction to degenerate Reinforcement Learning problem
  - actions <=> indexed predictions
  - observation states: x at the beginning, then empty
  - r = 0 except at the end, where r is loss function
- "training wheels" analogy

Sequence Labeling Tests

- Simplest nontrivial structure
- Performed on 4 tasks:
  - Handwriting Recognition
  - Named Entity Recognition
  - Syntactic Chunking
  - Joint Chunking, POS tagging

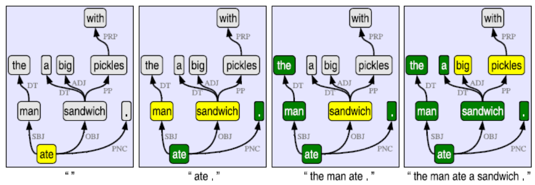
Experimental Results

ALGORITHM	Handwriting Small	Handwriting Large	NER Small	NER Large	Chunk	C+T
CLASSIFICATION						
Perceptron	65.56	70.05	91.11	94.37	83.12	87.88
Log Reg	68.65	72.10	93.62	96.09	85.40	90.39
SVM-Lin	75.75	82.42	93.74	97.31	86.09	93.94
SVM-Quad	82.63	82.52	85.49	85.49	∼	∼
STRUCTURED						
Str. Perc.	69.74	74.12	93.18	95.32	92.44	93.12
CRF	–	–	94.94	∼	94.77	96.48
SVM^{struct}	–	–	94.90	∼	–	–
M^3N-Lin	81.00	∼	–	–	–	–
M^3N-Quad	87.00	∼	–	–	–	–
SEARN						
Perceptron	70.17	76.88	95.01	97.67	94.36	96.81
Log Reg	73.81	79.28	95.90	98.17	94.47	96.95
SVM-Lin	82.12	90.58	95.91	98.11	94.44	96.98
SVM-Quad	87.55	90.91	89.31	90.01	∼	∼

Automatic Document Summarization

- Given a document collection and user query (topic), create a summary of the documents about the topic
- Typical approach is greedy sentence extraction
- Want short summaries with document compression
- SEARN uses vine-growth model

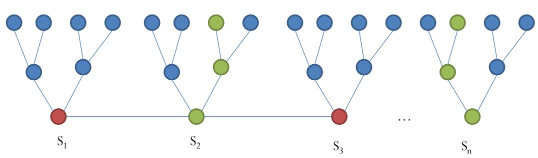
Vine Growth Model



- Models dependency structure
- If word w is added, all ancestors of w are also added
- Takes preference to shorter, grammatically correct sentences

Application to SEARN

- Search space and actions are the growth of the trees
- Incrementally grow summary by beginning a sentence or growing an existing one
- Frontier nodes – head of each sentence (red)
- Summary nodes – initialized to empty set (green)



SEARN in practice

- Data – DUC 2005 data, 50 collections, 25 documents each, each collection has a topic
- Metric – “Rouge 2” metric, uses evenly weighted bigram overlaps between summaries
- Initial Policy – use search to approximate total cost
- Features – includes aspects of current summary set and input document
  - e.g Word identity, stem, POS of w, location of s, length of document,…

Experimental Results

	ORACLE		SEARN		BAYESUM			
	Vine	Extr	Vine	Extr	D05	D03	Base	Best
100 w	.0729	.0362	.0415	.0345	.0340	.0316	.0181	-
250 w	.1351	.0809	.0824	.0767	.0762	.0698	.0403	.0725

- Oracle – system that returns summary given *true* output
- BAYESUM – achieved highest human scores in DUC 05
- Other structured prediction not listed- intractable

Discussion

- SEARN
  - Solves complex structured prediction problem
  - Minimal assumptions about structure and loss function
- Limitation
  - Overfitting
  - No optimal policy for noisy data
- Overall idea : proper local learning leads to good global performance
- Method for integrating search and learning
- SEARN : between global learning and local learning

Thank You

Questions