Complex Prediction Problems

A novel approach to multiple Structured Output Prediction

Yasemin Altun

Max-Planck Institute

ECML HLIE08

Information Extraction

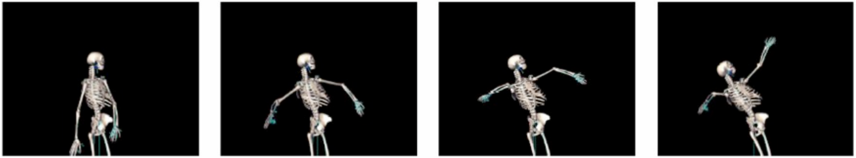
- Extract structured information from unstructured data
- Typical subtasks
  - Named Entity Recognition: person, location, organization names
  - Coreference Identification: noun phrases refering to the same object
  - Relation extraction: eg. Person works for Organization
- Ultimate tasks
  - Document Summarization
  - Question Answering

Complex Prediction Problems

- *Complex tasks* consisting of multiple *structured* subtasks
- Real world problems too complicated for solving at once
- Ubiquitous in many domains
    - Natural Language Processing
    - Computational Biology
    - Computational Vision

Complex Prediction Example

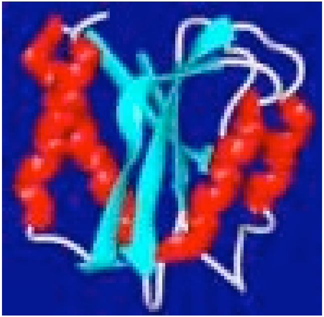
- Motion Tracking in Computational Vision
- Subtask: Identify joint angles of human body



Complex Prediction Example

- 3-D protein structure prediction in Computational Biology
- Subtask: Identify secondary structured Prediction from amino-acid sequence

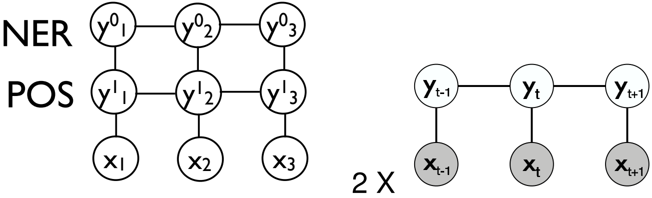
AAYKSHGSGDYGDHDVGHPTPGDPWVEPDYGINVYHSDTYSGQW
AAYKSHGSGDYGDHDVGHPTPGDPWVEPDYGINVYHSDTYSGQW



Standard Approach to Complex Prediction

Pipeline Approach

- Define intermediate/sub-tasks
- Solve them individually or in a cascaded manner
- Use output of subtasks as *features* (input) for target task



where for POS and for NER where x:x+POS tags

- **Problems:**
  - Error propagation
  - No learning across tasks

New Approach to Complex Prediction

Proposed approach:

- Solve tasks jointly discriminatively
  - Decompose multiple *structured* tasks
  - Use methods from multitask learning
    - Good predictors are it smooth
    - Restrict the search space for smooth functions of all tasks
  - Device targeted approximation methods
    - Standard approximation algorithms do not capture specifics
    - Dependencies within tasks are stronger than dependencies across tasks
  - **Advantages**
    - Less/no error propagation
    - Enables learning across tasks

Structured Output (SO) Prediction

- Supervised Learning
  - Given input/output pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$

$$\mathcal{Y} = \{0, \ldots, m\}, \mathcal{Y} = \Re$$

  - Data from unknown/fixed distribution D over $\mathcal{X} \times \mathcal{Y}$
  - Goal: Learn a mapping $h : \mathcal{X} \to \mathcal{Y}$
  - State-of-the art are discriminative, eg. SVMs, Boosting
- In *Structured Output* prediction,
  - Multivariate response variable with structural dependency. $|\mathcal{Y}|$: exponential in number of variables
  - Sequences, tree, hierarchical classification, ranking

SO Prediction

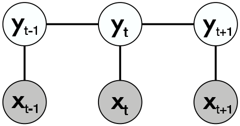
- Generative framework: Model $P(x, y)$
  - Advantages: Efficient learning and inference algorithms
  - Disadvantages: Harder problem, Questionable independence assumption, Limited representation
- Local approaches: eg. [Roth, 2001]
  - Advantages: Efficient algorithms
  - Disadvantages: Ignore/problematic long range dependencies
- Discriminative learning
  - Advantages: Richer representation via kernels, capture dependencies
  - Disadvantages: Expensive computation (SO prediction involves iteratively computing marginals or best labeling during training)

Formal Setting

- Given $S = ((x_1, y_1), \ldots, (x_l, y_n))$
- Find $h : \mathcal{X} \rightarrow \mathcal{Y}, h(x) = \mathrm{argmax}_y F(x, y)$
- Linear discriminant function $F : \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}$

$$F_w(x, y) = \langle \psi(x, y), w \rangle$$

- Cost function: $\Delta(y, y') \geq 0$ eg. 0-1 loss, Hamming loss
- Canonical example: Label sequence learning, where both x and y are sequences

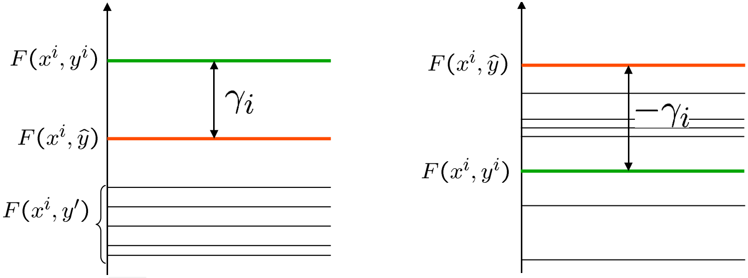


Maximum Margin Learning [Altun et al 03]

- Define separation margin [Crammer & Singer 01]

$$\gamma_i = F_w(x_i, y_i) - \max_{y \neq y_i} F_w(x_i, y)$$

- Maximize $\min_i \gamma_i$ with small $\|w\|$



- Minimize $\sum_i \max_{y \neq y_i} (1 + F_w(x_i, y) - F_w(x_i, y_i))_+ + \lambda \|w\|_2^2$

Max-Margin Learning (cont.)

$$\sum_i \max_{y \neq y_i} (1 + F_w(x_i, y) - F_w(x_i, y_i))_+ + \lambda \|w\|_2^2$$

- A convex **non**-quadratic program

$$\min_{w,\xi} \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_i \xi_i$$

$$s.t. \langle w, \psi(x_i, y_i)\rangle - \max_{y \neq y} \langle w, \psi(x_i, y)\rangle \geq 1 - \xi_i, \quad \forall i$$

Max-Margin Learning (cont.)

$$\sum_i \max_{y \neq y_i} (1 + F_w(x_i, y) - F_w(x_i, y_i))_+ + \lambda \|w\|_2^2$$

- A convex **quadratic** program

$$\min_{w,\xi} \frac{1}{2}\|w\|_2^2 + \frac{C}{n}\sum_i \xi_i$$

$$\text{s.t.}\ \langle w, \psi(x_i, y_i)\rangle - \langle w, \psi(x_i, y)\rangle \geq 1 - \xi_i, \quad \forall i, \forall y \neq y_i$$

- Number of constraints exponential
- Sparsity: Only a few of the constraints will be active

Max-Margin Dual Problem

- Using Lagrangian techniques, the dual:

$$\max -\frac{1}{2}\sum_{i,j,y,y'} \alpha_i(y)\alpha_j(y')\delta\psi(x_i,y)\delta\psi(x_j,y') + \sum_{i,y}\alpha_i(y)$$

$$\text{s.t. } 0 \leq \alpha_i(y), \quad \sum_{y \neq y_i} \alpha_i(y) \leq \frac{C}{n}, \quad \forall i$$

  where $\delta\psi(x_i, y) = \psi(x_i, y_i) - \psi(x_i, y)$
- Use the structure of equality constraints
- Replace the inner product with a kernel for implicit non-linear mapping

Max-Margin Optimization

- Exploit sparseness and the structure of constraints by incrementally adding constraints (cutting plane algorithm)
- Maintain a working set $\mathcal{Y}_i \subseteq \mathcal{Y}$ for each training instance
- Iterate over training instance
- Incrementally augment (or shrink) working set $\mathcal{Y}_i$

$$\hat{y} = \operatorname*{argmax}_{y \in \mathcal{Y} - y_i} F(x_i, y) \text{ via Dynamic Programming}$$

$$F(x_i, y_i) - F(x_i, \hat{y}) \leq 1 - \epsilon?$$

- Optimize over Lagrange multipliers α_i of $\mathcal{Y}_i$

Max-Margin Cost Sensitivity

- Cost function $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \Re$
  - Multiclass 0/1 loss
  - Sequences Hamming loss
  - Parsing (1-F1)
- Extend max-margin framework for cost sensitivity
  - (Taskar et.al. 2004)

$$\max_{y \neq y_i} (\Delta(y_i, y) + F_w(x_i, y) - F_w(x_i, y_i))_+$$

  - (Tsochantaridis et.al. 2004)

$$\max_{y \neq y_i} \Delta(y_i, y)(1 + F_w(x_i, y) - F_w(x_i, y_i))_+$$

Example: Sequences

$$\Lambda(2) = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} , \begin{pmatrix} 0 \\ x \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} = \Phi(x, 2)$$

- Viterbi decoding for argmax operation
- Decompose features into time
  $\psi(x, y) = \sum_t (\psi(x_t, y_t) + \psi(y_t, y_{t-1}))$
- Two types of features
  - Observation-label: $\psi(x_t, y_t) = \phi(x_t) \otimes \Lambda(y_t)$
  - Label-label: $\psi(y_t, y_{t-1}) = \Lambda(y_t) \otimes \Lambda(y_{t-1})$

Example: Sequences (cont.)

- Inner product between two features separately

$$
\begin{aligned}
&\langle \psi(x,y), \psi(\bar{x},\bar{y}) \rangle \\
&= \sum_{s,t} \langle \phi(x_t), \phi(\bar{x}_s) \rangle \delta(y_t, \bar{y}_s) + \delta(y_t, \bar{y}_s) \delta(y_{t-1}, \bar{y}_{s-1}) \\
&= \sum_{s,t} k((x_t, y_t), (\bar{x}_s, \bar{y}_s)) + \tilde{k}((y_t, y_{t-1}), (\bar{y}_s, \bar{y}_{s-1}))
\end{aligned}
$$

- Arbitrary kernels on x
- Linear kernels on y

Other SO Prediction Methods

- Find w to minimize expected loss $E_{(x,y)\sim D}[\Delta(y, h_f(x))]$

$$w^* = \operatorname*{argmin}_w \sum_{i=1}^{l} \mathcal{L}(x_i, y_i, w) + \lambda \|w\|^2$$

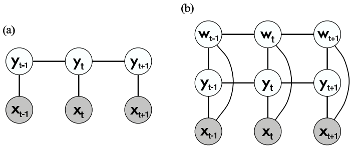
- Loss functions
  - Hinge loss
  - Log-loss: **CRF** [Lafferty et al 2001]

$$\mathcal{L}(x, y, f) = -F(x, y) + \log \sum_{\hat{y} \in \mathcal{Y}} \exp(F(x, \hat{y}))$$

  - Exp-loss: **Structured Boosting** [Altun et al 2002]

$$\mathcal{L}(x, y, f) = \sum_{\hat{y} \in \mathcal{Y}} \exp(F(x, \hat{y}) - F(x, y))$$

Complex Prediction via SO Prediction



- **Possible Solution:** Treat complex prediction as a loopy graph and use standard approximation methods
- **Shortcomings:**
  - No knowledge of graph structure
  - No knowledge that tasks defined over same input space
- **Solution:**
  - Dependencies within tasks more important than dependencies across tasks. Use this for approximation method
  - Restrict function class for each task via learning across tasks

Joint Learning of Multiple SO prediction [Altun 2008]

- Tasks $1, \ldots, m$
- Learn a discriminative function $T : \mathcal{X} \to \mathcal{Y}^1 \times \ldots \mathcal{Y}^m$

$$T(x, y; w, \bar{w}) = \sum_{\ell} \left[F^{\ell}(x, y^{\ell}; w_{\ell}) + \sum_{\ell'} F^{\ell\ell'}(y^{\ell}, y^{\ell'}; w, \bar{w}) \right].$$

where w_ℓ capture dependencies within individual tasks

$\bar{w}_{\ell,\ell'}$ capture dependencies across tasks

- F^{ℓ} defined as before
- $F^{\ell\ell'}$ linear functions wrt cliques assignments of tasks ℓ, ℓ'

Joint Learning of Multiple SO prediction

- Assume a low dimensional representation Θ shared across all tasks [Argyriou et al 2007]

$$F^\ell(x, y^\ell; w_\ell, \Theta) = \left\langle w_{\ell\sigma}, \Theta^T \psi(x, y^\ell) \right\rangle$$

- Find T by discovering Θ and learning $w, \bar{w}$

$$\min_{\Theta, w, \bar{w}} \hat{r}(\Theta) + r(w) + \bar{r}(\bar{w}) + \sum_{\ell=1}^{m} \sum_{i=1}^{n} \mathcal{L}^\ell(x_i, y_i^\ell; w, \bar{w}, \Theta),$$

- $r, \bar{r}$ regularization, eg. $L2$ norm
- $\hat{r}$, eg. Frobenius norm, trace norm
- L loss function, eg. Log-loss, hinge-loss
- Optimization is **not** jointly convex over Θ and $w, \bar{w}$

Joint Learning of Multiple SO prediction

- Via a reformulation, we get a jointly convex optimization

$$\min_{A,D,\bar{w}} \sum_{\ell\sigma} \langle A_{\ell\sigma}, D^{+} A_{\ell\sigma} \rangle + \bar{r}(\bar{w}) + \sum_{\ell=1}^{m} \sum_{i=1}^{n} \mathcal{L}^{\ell}(x_i, y_i^{\ell}; A, \bar{w}).$$

- Optimize iteratively wrt $A, \bar{w}$ and D
- Closed form solution for D

Optimization

- A and $\bar{w}$ decomposes into tasks parameters
- Optimize wrt each tasks parameters iteratively
- **Problem**: $F^{\ell,\ell'}$ is function of all other tasks
- **Solution**: Loopy Belief Propagation like algorithm where each clique assignment is approximated wrt current parameters iteratively
- Run DP for all other tasks, fix clique assignment values, optimize wrt current task

Algorithm

Algorithm 1 Joint Learning of Multiple Structure Prediction Tasks

repeat
    for each task ℓ **do**
        compute $\hat{a}_\ell = \operatorname{argmin}_a \sum_i \mathcal{L}(\mathbf{x}_i, \mathbf{y}^\ell; a) + \langle a, D^+ a \rangle$ via computing ψ functions for each $\mathbf{x}_i$ with dynamic programming
    end for
    compute $D = \frac{(AA^T)^{\frac{1}{2}}}{\|A\|_T}$ and D^+
until convergence

Experiments

- Task1: POS tagging evaluated with accuracy
- Task2: NER evaluated with F1 score
- Data: 2000 sentences from CONLL03 English corpus
- Structure: Sequence
- Loss: Log-loss (CRF)

	Cascaded			Joint (no Θ)	MT-Joint
POS	92.63			93.21	93.67
NER	58.77(noP)	67.42(predP)	69.75 (trueP)	68.51	70.01

Table 1: Comparison of cascaded model and joint optimization for POS tagging and NER

Conclusion

- IE involves complex tasks, ie. multiple structured prediction tasks
- Structured prediction methods include CRFs, Max-Margin SO
- Proposed a novel approach to joint prediction of multiple SO problems
  - Using a special approximation algorithm
  - Using multi-task methods
- More experimental evaluation is required