30 Second Overview
- A structural SVM for solving sequence problems such as
- Finding basic blocks of code in binary executables (our task)
- Some kinds of scheduling problems
- What’s so special about these problems?
- A linear time inference algorithm
- Two appropriate loss functions allowing fast training (and
  why Hamming loss is inappropriate)
- Experiments comparing against SVM_hmm, a
discriminatively trained HMM

Our Task
- Segment binary executable into blocks of code (the
  actual instructions), and data (everything else), without
  running the program.
- Instructions have no demarcations.
- Similar problem with URLs such as www.whorerepresents.com.
  Similar with 我们要生活得有意义

Isn’t This Just Sequence Labeling/Segmentation?
- It’s a weighted interval scheduling problem
- Data blocks cannot start in arbitrary positions
- Code blocks can be really long
- Given a starting position, the span of a code block is
deterministic.
- Hamming loss inappropriate. Text analogy: parsing
  “driverballeretarace” as “<junk>, verbal, letter, race”. Good
  overlap, but misses all words.

Inference
\[
\Delta(s, y) = 4 \ (y \text{ misses } 3 \text{ (blue), uses } 1 \text{ extra (orange)})
\]

Block Loss
\[
\Delta(s, y) = 6 \ (y \text{ misses } 4 \text{ (light blue), uses } 2 \text{ extra (yellow)})
\]

Instruction Loss
\[
\text{Solved with cutting plane algorithm.}
\]

Mandatory Panel: SVM Formulation
\[
\min \frac{1}{2} ||w||^2 + C \sum_{i=1}^n \xi_i \\
\text{s.t. } \forall (x, y) : w^T \Psi(x, y) - w^T \Psi(x', y') \geq \Delta(y, y') - \xi_i
\]
where \(\Psi(x, y) = \sum_{(u, v) \in E} \Phi(x, u, v)\)

Loss Augmented Inference (Constraint Generation)
Find \(\arg \max_{x} w^T \Psi(x, y) + \Delta(y, y')\)
Fast: loss functions decompose over the vertices.
Inference on modified graph.

Experiments
200 binary executables from a typical Windows machine.

<table>
<thead>
<tr>
<th>Features: unigrams and bigrams around each position, histograms of instructions for long edges</th>
</tr>
</thead>
</table>
| \(\Delta_H: \text{Hamming loss for reference} \)
| \(\Delta_N: \text{normalized version } \Delta_N(x, y) = \frac{\Delta(x, y)}{||y||} \)
| \(L, I, B: \text{ average length, instructions, blocks} \) |

<table>
<thead>
<tr>
<th></th>
<th>(\Delta_H)</th>
<th>(\Delta_N)</th>
<th>(\Delta_I)</th>
<th>(\Delta_N)</th>
<th>(\Delta_B)</th>
<th>(\Delta)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Greedy</td>
<td>1623.6</td>
<td>1916.6</td>
<td>2164.3</td>
<td>7045.2</td>
<td>1564.9</td>
<td>4747.2</td>
</tr>
<tr>
<td>SVM(\text{hmm})</td>
<td>236.2</td>
<td>201.3</td>
<td>-</td>
<td>-</td>
<td>45.1</td>
<td>46.9</td>
</tr>
<tr>
<td>SVM(\text{wis})</td>
<td>98.8</td>
<td>115.6</td>
<td>44.6</td>
<td>98.0</td>
<td>26.1</td>
<td>41.1</td>
</tr>
<tr>
<td>SVM(\text{wis})(\Delta)</td>
<td>104.3</td>
<td>103.7</td>
<td>45.5</td>
<td>79.7</td>
<td>30.5</td>
<td>35.5</td>
</tr>
<tr>
<td>SVM(\text{wis})(\Delta)</td>
<td>86.5</td>
<td>98.2</td>
<td>39.6</td>
<td>80.2</td>
<td>21.5</td>
<td>32.1</td>
</tr>
<tr>
<td>SVM(\text{wis})(\Delta)</td>
<td>85.2</td>
<td>87.2</td>
<td>40.6</td>
<td>75.4</td>
<td>23.4</td>
<td>29.8</td>
</tr>
</tbody>
</table>

Code & data available at www.cs.cornell.edu/~nk/svmwis

---

Email: nk@cs.cornell.edu

WWW: http://www.cs.cornell.edu