Structured Prediction

- **binary classification**: output is binary
- **multiclass classification**: output is a number (small # of classes)
- **structured classification**: output is a structure (seq., tree, graph)
  - part-of-speech tagging, parsing, summarization, translation
  - exponentially many classes: search (inference) efficiency is crucial!
Generic Perceptron

- online-learning: one example at a time
- learning by doing
  - find the best output under the current weights
  - update weights at mistakes

![](image)
Perceptron: from binary to structured classification

**binary classification**

- 2 classes
- $x$ and $y = \pm 1$

**trivial**

- Exact inference
- Update weights if $y \neq z$

**easy**

- Exact inference
- Constant number of classes
- Update weights if $y \neq z$

**structured classification**

- Exponential number of classes
- Complex inference
- Update weights if $y \neq z$

**multiclass classification**

- Constant number of classes
- Update weights if $y \neq z$
From Perceptron to SVM

- 1959: Rosenblatt invention
  - 1962: Novikoff proof
  - 1964: Vapnik Chervonenkis
    - 1995: Cortes/Vapnik SVM
    - 2001: Lafferty+ CRF
    - 2002: Collins structured
    - 2003: Taskar M3N
  - 1999: Freund/Schapire voted/avg: revived
    - 2003: Crammer/Singer MIRA
    - 2004: Tsochantaridis struct. SVM
    - 2005: McDonald+ structured MIRA
  - 2006: Singer group aggressive
    - 2007-2010: Singer group Pegasos
- 1995: Cortes/Vapnik SVM
  - 1999: Freund/Schapire voted/avg: revived
    - 2003: Crammer/Singer MIRA
    - 2004: Tsochantaridis struct. SVM
    - 2005: McDonald+ structured MIRA
- 2002: Collins structured
  - 2003: Taskar M3N
  - 2004: Tsochantaridis struct. SVM
- 2005: McDonald+ structured MIRA
  - 2006: Singer group aggressive
    - 2007-2010: Singer group Pegasos
- 1964: Vapnik Chervonenkis
  - 1966: Aizerman+ 
- 1959: Rosenblatt invention
- 1962: Novikoff proof
- 1964: Aizerman+
- Multinomial logistic regression (max. entropy)
- Online
  - Conservative updates
  - Inseparable case
- Batch
  - Online approx. max margin
  - Subgradient descent
  - Minibatch
- AT&T Research
  - Ex-AT&T and students
  - Fall of USSR
  - Same journal!
Multiclass Classification

- one weight vector ("prototype") for each class:
  \[ w = (w^{(1)}, w^{(2)}, \ldots, w^{(M)}) \],

- multiclass decision rule:
  \[ \hat{y} = \text{argmax}_{z \in 1 \ldots M} w^{(z)} \cdot x \] (best agreement w/ prototype)

Q1: what about 2-class?
Q2: do we still need augmented space?
Multiclass Perceptron

- on an error, penalize the weight for the wrong class, and reward the weight for the true class
Convergence of Multiclass

update rule: 
\[ w \leftarrow w + \Delta \Phi(x, y, z) \]

separability: 
\[ \exists u, \text{ s.t. } \forall (x, y) \in D, z \neq y \]
\[ u \cdot \Delta \Phi(x, y, z) \geq \delta \]

where \( w^{(i)} \) is used to calculate the functional margin for training example with label \( i \);

for a given training example \( x \) and a label \( y \), we define feature map function \( \Phi \) as
\[ \Phi(x, y) = (0^{(1)}, \ldots, 0^{(y-1)}, x, 0^{(y+1)}, \ldots, 0^{(M)}). \]
such that \( w \cdot \Phi(x, y) = w^{(y)} \cdot x \).

We also define that, with a given training example \( x \), the difference between two feature vectors for labels \( y \) and \( z \) as \( \Delta \Phi \):
\[ \Delta \Phi(x, y, z) = \Phi(x, y) - \Phi(x, z). \]
Example: POS Tagging

- **gold-standard:**

  \[ \text{DT} \quad \text{NN} \quad \text{VBD} \quad \text{DT} \quad \text{NN} \quad y \]

  the \quad man \quad bit \quad the \quad dog \quad x

- **current output:**

  \[ \text{DT} \quad \text{NN} \quad \text{NN} \quad \text{DT} \quad \text{NN} \quad z \]

  the \quad man \quad bit \quad the \quad dog \quad x

- **assume only two feature classes**

- **tag bigrams**

- **word/tag pairs**

- **weights ++:**
  
  - (NN, VBD)
  - (VBD, DT)
  - (VBD, bit)

  \[ \Phi(x) - \Phi(x, z) \]

- **weights --:**
  
  - (NN, NN)
  - (NN, DT)
  - (NN, bit)
Structured Perceptron

Inputs: Training set \((x_i, y_i)\) for \(i = 1 \ldots n\)

Initialization: \(W = 0\)

Define: 
\[
F(x) = \text{argmax}_{y \in \text{GEN}(x)} \Phi(x, y) \cdot W
\]

Algorithm: For \(t = 1 \ldots T, i = 1 \ldots n\)
\[
z_i = F(x_i) \\
\text{If } (z_i \neq y_i) \quad W \leftarrow W + \Phi(x_i, y_i) - \Phi(x_i, z_i)
\]

Output: Parameters \(W\)
Inference: Dynamic Programming

exact inference

update weights if \( y \neq z \)

\[
\begin{align*}
\text{x(t - 1)} & \rightarrow \text{x(t)} & \rightarrow \text{x(t + 1)} \\
\text{y(t - 1)} & \rightarrow \text{y(t)} & \rightarrow \text{y(t + 1)}
\end{align*}
\]
Complete this Python code implementing the Viterbi algorithm for part-of-speech tagging. It should print a list of word/tag pairs, e.g. [('a', 'D'), ('can', 'N'), ('can', 'A'), ('can', 'V'), ('a', 'D'), ('can', 'N')].

```python
from collections import defaultdict

best = defaultdict(lambda: defaultdict(float))
best[0]['<s>'] = 1
back = defaultdict(dict)

words = '<s> a can can can a can </s>'.split()

tags = {'a': ['D'], 'can': ['N', 'A', 'V'], '</s>': ['</s>']}  # possible tags for each word
ptag = {'D': {'N': 1}, 'V': {'</s>': 0.5, 'D': 0.5}, ...}  # ptag[x][y] = p(y | x)
pword = {'D': {'a': 0.5, 'N': {'can': 0.1}, ...}  # pword[x][w] = p(w | x)

for i, word in enumerate(words[1:], 1):
    for tag in tags[word]:
        for prev in best[i-1]:
            if tag in ptag[prev]:
                score = best[i-1][prev] * ptag[prev][tag] * pword[tag][word]
                if score > best[i][tag]:
                    best[i][tag] = score
                    back[i][tag] = prev

    def backtrack(i, tag):
        if i == 0:
            return []
        return backtrack(i-1, back[i][tag]) + [(words[i], tag)]

print backtrack(len(words)-1, '</s>')[::-1]
```

Q: what about top-down recursive + memoization?
Efficiency vs. Expressiveness

- the inference (argmax) must be efficient
  - either the search space $\text{GEN}(x)$ is small, or factored
  - features must be local to $y$ (but can be global to $x$)
    - e.g. bigram tagger, but look at all input words (cf. CRFs)

$$
\arg\max_{y \in \text{GEN}(x)} y_i
\quad \text{inference}
\quad x_i
\quad z_i
\quad y_i
\quad \text{update weights}
\quad w
$$

$$(x, y)$$

Structured Prediction
Averaged Perceptron

Inputs: Training set \((x_i, y_i)\) for \(i = 1 \ldots n\)

Initialization: \(W_0 = 0\)

Define: \(F(x) = \arg\max_{y \in \text{GEN}(x)} \Phi(x, y) \cdot W\)

Algorithm: For \(t = 1 \ldots T, i = 1 \ldots n\)
\[
z_i = F(x_i)
\]
If \((z_i \neq y_i)\)
\[
W_{j+1} \leftarrow W_j + \Phi(x_i, y_i) - \Phi(x_i, z_i)
\]

Output: Parameters \(W = \sum_j W_j\)

- more stable and accurate results
- approximation of voted perceptron (Freund & Schapire, 1999)
### Averaging Tricks

- **Daume (2006, PhD thesis)**

---

**Algorithm: AVERAGED STRUCTURED PERCEPTRON**

```
1: \( w_0 \leftarrow \langle 0, \ldots, 0 \rangle \)
2: \( w_a \leftarrow \langle 0, \ldots, 0 \rangle \)
3: \( c \leftarrow 1 \)
4: \textbf{for} \( i = 1 \ldots I \) \textbf{do}
5: \textbf{for} \( n = 1 \ldots N \) \textbf{do}
6: \( \hat{y}_n \leftarrow \arg \max_{y \in Y} w_0^\top \Phi(x_n, y_n) \)
7: \textbf{if} \( y_n \neq \hat{y}_n \) \textbf{then}
8: \( w_0 \leftarrow w_0 + \Phi(x_n, y_n) - \Phi(x_n, \hat{y}_n) \)
9: \( w_a \leftarrow w_a + c \Phi(x_n, y_n) - c \Phi(x_n, \hat{y}_n) \)
10: \textbf{end if}
11: \( c \leftarrow c + 1 \)
12: \textbf{end for}
13: \textbf{end for}
14: \textbf{return} \( w_0 - w_a/c \)
```

---

**Figure 2.3:** The averaged structured perceptron learning algorithm.

---

**sparse vector: defaultdict**
Do we need smoothing?

- smoothing is much easier in discriminative models
- just make sure for each feature template, its subset templates are also included
  - e.g., to include \((t_0 \cdot w_0 \cdot w_{-1})\) you must also include
  - \((t_0 \cdot w_0) \cdot (t_0 \cdot w_{-1}) \cdot (w_0 \cdot w_{-1})\)
  - and maybe also \((t_0 \cdot t_{-1})\) because \(t\) is less sparse than \(w\)
Convergence with Exact Search

- linear classification: converges iff. data is separable
- structured: converges iff. data separable & search exact
  - there is an oracle vector that correctly labels all examples
  - one vs the rest (correct label better than all incorrect labels)
- theorem: if separable, then $\# \text{ of updates} \leq \frac{R^2}{\delta^2}$
  
- R: diameter

$y=-1$  $y=+1$

Rosenblatt $\Rightarrow$ Collins
1957        2002

$z \neq y_{100}$
Geometry of Convergence Proof pt 1

1: repeat
2: for each example \((x, y)\) in \(D\) do
3: \(z \leftarrow \text{EXACT}(x, w)\)
4: if \(z \neq y\) then
5: \(w \leftarrow w + \Delta \Phi(x, y, z)\)
6: until converged

1-best

exact inference

update weights if \(y \neq z\)

perceptron update:

\[
 w^{(k+1)} = w^{(k)} + \Delta \Phi(x, y, z)
\]

\[
 u \cdot w^{(k+1)} = u \cdot w^{(k)} + u \cdot \Delta \Phi(x, y, z) \geq \delta \quad \text{margin}
\]

(by induction)

\[
 u \cdot w^{(k+1)} \geq k \delta
\]

\[
 \|u\| \|w^{(k+1)}\| \geq u \cdot w^{(k+1)} \geq k \delta
\]

(part 1: upperbound)
Geometry of Convergence Proof \textit{pt 2}

1: repeat
2: for each example \((x, y)\) in \(D\) do
3: \(z \leftarrow \text{EXACT}(x, w)\)
4: if \(z \neq y\) then
5: \(w \leftarrow w + \Delta \Phi(x, y, z)\)
6: until converged

update weights if \(y \neq z\)

violation: incorrect label scored higher

perceptron update:

\[
\begin{align*}
w^{(k+1)} &= w^{(k)} + \Delta \Phi(x, y, z) \\
\|w^{(k+1)}\|^2 &= \|w^{(k)} + \Delta \Phi(x, y, z)\|^2 \\
&= \|w^{(k)}\|^2 + \|\Delta \Phi(x, y, z)\|^2 + 2 w^{(k)} \cdot \Delta \Phi(x, y, z) \\
&\leq R^2
\end{align*}
\]

by induction: \(\|w^{(k+1)}\|^2 \leq kR^2\) (part 2: upperbound)

parts 1+2 \Rightarrow update bounds:

\[
k \leq \frac{R^2}{\delta^2}
\]
Experiments
Experiments: Tagging

- (almost) identical features from (Ratnaparkhi, 1996)
- trigram tagger: current tag $t_i$, previous tags $t_{i-1}$, $t_{i-2}$
- current word $w_i$ and its spelling features
- surrounding words $w_{i-1} \ldots w_{i+2}$

<table>
<thead>
<tr>
<th>Method</th>
<th>Error rate/%</th>
<th>Numits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perc, avg, cc=0</td>
<td>2.93</td>
<td>10</td>
</tr>
<tr>
<td>Perc, noavg, cc=0</td>
<td>3.68</td>
<td>20</td>
</tr>
<tr>
<td>Perc, avg, cc=5</td>
<td>3.03</td>
<td>6</td>
</tr>
<tr>
<td>Perc, noavg, cc=5</td>
<td>4.04</td>
<td>17</td>
</tr>
<tr>
<td>ME, cc=0</td>
<td>3.4</td>
<td>100</td>
</tr>
<tr>
<td>ME, cc=5</td>
<td>3.28</td>
<td>200</td>
</tr>
</tbody>
</table>
Experiments: NP Chunking

- **B-I-O scheme**
- Rockwell International Corp.
- 's Tulsa unit said it signed
- a tentative agreement...

- **features:**
  - unigram model
  - surrounding words and POS tags

<table>
<thead>
<tr>
<th>Current word</th>
<th>$w_i$ &amp; $t_i$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Previous word</td>
<td>$w_{i-1}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Word two back</td>
<td>$w_{i-2}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Next word</td>
<td>$w_{i+1}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Word two ahead</td>
<td>$w_{i+2}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Bigram features</td>
<td>$w_{i-2}, w_{i-1}$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$w_{i-1}, w_i$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$w_i, w_{i+1}$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$w_{i+1}, w_{i+2}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Current tag</td>
<td>$p_i$ &amp; $t_i$</td>
</tr>
<tr>
<td>Previous tag</td>
<td>$p_{i-1}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Tag two back</td>
<td>$p_{i-2}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Next tag</td>
<td>$p_{i+1}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Tag two ahead</td>
<td>$p_{i+2}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Bigram tag features</td>
<td>$p_{i-2}, p_{i-1}$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$p_{i-1}, p_i$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$p_i, p_{i+1}$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$p_{i+1}, p_{i+2}$ &amp; $t_i$</td>
</tr>
<tr>
<td>Trigram tag features</td>
<td>$p_{i-2}, p_{i-1}, p_i$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$p_{i-1}, p_i, p_{i+1}$ &amp; $t_i$</td>
</tr>
<tr>
<td></td>
<td>$p_i, p_{i+1}, p_{i+2}$ &amp; $t_i$</td>
</tr>
</tbody>
</table>
Experiments: NP Chunking

- results

<table>
<thead>
<tr>
<th>Method</th>
<th>F-Measure</th>
<th>Numits</th>
</tr>
</thead>
<tbody>
<tr>
<td>Perceptron, avg, cc=0</td>
<td>93.53</td>
<td>13</td>
</tr>
<tr>
<td>Perceptron, noavg, cc=0</td>
<td>93.04</td>
<td>35</td>
</tr>
<tr>
<td>Perceptron, avg, cc=5</td>
<td>93.33</td>
<td>9</td>
</tr>
<tr>
<td>Perceptron, noavg, cc=5</td>
<td>91.88</td>
<td>39</td>
</tr>
<tr>
<td>Max-ent, cc=0</td>
<td>92.34</td>
<td>900</td>
</tr>
<tr>
<td>Max-ent, cc=5</td>
<td>92.65</td>
<td>200</td>
</tr>
</tbody>
</table>

- (Sha and Pereira, 2003) trigram tagger
  - voted perceptron: 94.09% vs. CRF: 94.38%
Structured SVM

- structured perceptron: \( w \cdot \Delta \phi(x,y,z) > 0 \)
- SVM: for all \((x,y)\), functional margin \( y(w \cdot x) \geq 1 \)
- structured SVM version 1: simple loss
  - for all \((x,y)\), for all \(z \neq y\), margin \( w \cdot \Delta \phi(x,y,z) \geq 1 \)
  - correct \(y\) has to score higher than any wrong \(z\) by 1
- structured SVM version 2: structured loss
  - for all \((x,y)\), for all \(z \neq y\), margin \( w \cdot \Delta \phi(x,y,z) \geq \ell(y,z) \)
  - correct \(y\) has to score higher than any wrong \(z\) by \(\ell(y,z)\), a distance metric such as hamming loss
Loss-Augmented Decoding

- want for all $z$: $w \cdot \phi(x, y) \geq w \cdot \phi(x, z) + \ell(y, z)$
- same as: $w \cdot \phi(x, y) \geq \max_z w \cdot \phi(x, z) + \ell(y, z)$
- loss-augmented decoding: $\arg\max_z w \cdot \phi(x, z) + \ell(y, z)$
- if $\ell(y, z)$ factors in $z$ (e.g. hamming), just modify DP

\[
\text{Algorithm 41: } \text{StochSubGradStructSVM}(D, \text{MaxIter}, \lambda, \ell) \\
1: w \leftarrow 0 \\
2: \text{for } \text{iter} = 1 \ldots \text{MaxIter} \text{ do} \\
3: \quad \text{for all } (x, y) \in D \text{ do} \\
4: \quad \quad \hat{y} \leftarrow \arg\max_{\tilde{y} \in Y(x)} w \cdot \phi(x, \tilde{y}) + \ell(y, \tilde{y}) \quad \text{// loss-augmented prediction} \\
5: \quad \quad \text{if } \hat{y} \neq y \text{ then} \\
6: \quad \quad \quad w \leftarrow w + \phi(x, y) - \phi(x, \hat{y}) \quad \text{// update weights} \\
7: \quad \quad \text{end if} \\
8: \quad w \leftarrow w - \frac{\lambda}{N} w \quad \text{// shrink weights due to regularizer} \\
9: \quad \text{end for} \\
10: \text{end for} \\
11: \text{return } w \\
\]

very similar to Pegasos; but should use Pegasos framework instead
Correct Version following Pegasos

- want for all $z$: $w \cdot \phi(x,y) \geq w \cdot \phi(x,z) + \ell(y,z)$
- same as: $w \cdot \phi(x,y) \geq \max_z w \cdot \phi(x,z) + \ell(y,z)$
- loss-augmented decoding: $\arg\max_z w \cdot \phi(x,z) + \ell(y,z)$
- if $\ell(y,z)$ factors in $z$ (e.g. hamming), just modify DP

\begin{algorithm}
\caption{StochSubGradStructSVM($D$, MaxIter, $\lambda$, $\ell$)}
\begin{algorithmic}[1]
\State $w \leftarrow 0$ // initialize weights
\For{$iter = 1 \ldots$ MaxIter}
\ForAll{$(x,y) \in D$}
\State $w \leftarrow w - 1/t w$ // shrink weights due to regularizer
\State $\hat{y} \leftarrow \arg\max_{\hat{y} \in \mathcal{Y}(x)} w \cdot \phi(x,\hat{y}) + \ell(y,\hat{y})$ // loss-augmented prediction
\If{$\hat{y} \neq y$}
\State $w \leftarrow w + \frac{NC}{2t} (\phi(x,y) - \phi(x,\hat{y}))$ // update weights
\EndIf
\EndFor
\EndFor
\State return $w$ // return learned weights
\end{algorithmic}
\end{algorithm}

$N=|D|$, $C$ is from SVM, $t += 1$ for each example
Struct. Perceptron vs Struct. SVM

- tagging, ATIS (train: 488 sent); SVM < avg perc << perc

**perceptron**

epoch 1 updates 102, $|W|=291$, train_err 3.90%, dev_err 9.36% avg_err 6.14%
epoch 2 updates 91, $|W|=334$, train_err 3.33%, dev_err 8.19% avg_err 4.97%
epoch 3 updates 78, $|W|=347$, train_err 2.92%, dev_err 5.85% avg_err 4.97%
epoch 4 updates 81, $|W|=368$, train_err 3.11%, dev_err 6.73% avg_err 5.85%
epoch 5 updates 78, $|W|=378$, train_err 2.70%, dev_err 6.14% avg_err 5.56%
epoch 6 updates 63, $|W|=385$, train_err 2.26%, dev_err 6.14% avg_err 5.56%
epoch 7 updates 69, $|W|=385$, train_err 2.43%, dev_err 7.02% avg_err 5.56%
epoch 8 updates 60, $|W|=388$, train_err 2.15%, dev_err 6.73% avg_err 5.56%
epoch 9 updates 59, $|W|=390$, train_err 2.04%, dev_err 6.14% avg_err 5.56%
epoch 10 updates 64, $|W|=394$, train_err 2.15%, dev_err 5.85% avg_err 5.26%

**SVM C=1**

epoch 1 updates 116, $|W|=311$, train_err 4.55%, dev_err 5.85%
epoch 2 updates 82, $|W|=328$, train_err 3.05%, dev_err 4.97%
epoch 3 updates 78, $|W|=334$, train_err 2.92%, dev_err 5.56%
epoch 4 updates 77, $|W|=339$, train_err 2.92%, dev_err 5.26%
epoch 5 updates 80, $|W|=344$, train_err 2.94%, dev_err 5.56%
epoch 6 updates 73, $|W|=345$, train_err 2.75%, dev_err 4.68%
epoch 7 updates 72, $|W|=347$, train_err 2.75%, dev_err 4.97%
epoch 8 updates 75, $|W|=352$, train_err 2.86%, dev_err 4.97%
epoch 9 updates 74, $|W|=353$, train_err 2.78%, dev_err 4.97%
epoch 10 updates 72, $|W|=354$, train_err 2.78%, dev_err 4.97%