Today: latent variable SVMstruct - CCCP
dep deep learning - RNN

**Latent variables**

motivating: semantic segmentation

segmentation is expensive $\Rightarrow Z$ "latent variable"
p: perhaps only have class labels $\Rightarrow Y$
also: [Felzenszwalb et al. TPAMI 2010]

"deformable part models" for object recognition
$\Rightarrow Z$ there was an object part configuration

before, we had $s(x,y;w) = \langle w, \ell(x,y) \rangle$
now, consider $s(x,y,Z;w) = \langle w, \ell(x,y,Z) \rangle$

As before, could predict with argmax $s(x,y,Z;w) \triangleright h_w(x)$

CRF approach $\Rightarrow$ hidden CRF $\mathcal{P}(y,z|x)$

simply to latent variable modeling

$\Rightarrow$ marginalize $Z$ out

ML $\Rightarrow$ EM (expectation-maximization)

**Latent SVMstruct**

\[ l(y,(\hat{y},\hat{z})) \]

generalized structure huge loss:

\[ s(x,y;w) = \max_{\hat{z}} \langle w, \ell(x,y,\hat{z}) \rangle + l(y,\hat{z}) \]

(gradient-based minimization procedure $\Rightarrow$ like EM)
\[ f(x,y,w) = \max_{y,z} <w, \ell(x,y,z)> + \ell(y, (y,z)) \]

\[ \geq \max_{y,z} <w, \ell(x,y,z)> + \ell(y, (y,z)) \]

\[ \geq v(w) \]

Here \( f(x,y,w) = u(w) - v(w) \) where \( u \) \& \( v \) are convex functions of \( w \).

"difference of convex functions"

\( \text{CCCP procedure is to approximate minimizing this} \)

**CCCP procedure:**

- Minimize \( v(w) \) at \( w \) to get an upper bound
- \( w_{k+1} \) is obtained by minimizing upper bound
- Repeat \( k \) for a majorization-minimization procedure

\[ \frac{\partial v(w)}{\partial w} \]

\[ w_{k+1} = \arg \min_{w} g_t(w) \]

Properties of procedure:

- Like \( EM \), descent procedure is.
  \[ f(w_{k+1}) \leq f(w_k) \]

\[ f(w_k) = \frac{\partial v(w_k)}{\partial w} \]

**majorization property**

- Local linear convergence to a stationary pt. for latent SVMs

*CCCP for SVMs:

\[ v(w) = \max_{z} \in <w, \ell(x,y,z)> \]

\[ \frac{\partial v(w_k)}{\partial w} = \ell(x,y, z) \]

\[ \Rightarrow g_t(w) = \max_{z} <w, \ell(x,y, z)> + \ell(y, (y,z)) - <w, \ell(x,y, z)> \]

= the SVM loss obj.
Deep learning:

go from \( \langle w, y(x, y) \rangle \) to \( \langle w, y(x, y, \theta) \rangle \)

I) **plug-in "deep learning" features in a structured prediction model**

- **Examples:** OCR, image analysis
- **Example:** [Un et al. IJCV 2015] used convolutional CNNs for text recognition

\[
\mathcal{L}_t(x_t, y_t) = \begin{cases} 0 & \text{if } y_t \text{ is correct} \\ 1 & \text{otherwise} \end{cases}
\]

instead of \( \mathcal{L}_t(x_t, y_t) = \begin{cases} 0 & \text{if } y_t \text{ is correct} \\ \| \text{NN}(x_t) - y_t \| & \text{otherwise} \end{cases} \)

II) **"end-to-end" training**

**Structured prediction energy networks (SPENs)**

\[
y \xrightarrow{N} \mathcal{L}_n(x_t, y_t) \xrightarrow{N} S(x, y, w, \theta)
\]

III) **recurrent neural networks (RNN)**

- **Motivation:**
  \[
  p(y|x) = \prod_{t=1}^{T} p(y_t | y_{1:t-1}, x)
  \]
- **Graphical model approach**
  \[
  \mathbb{P}(y_t | y_{1:t-1}, x)
  \]

- **RNN with no causal independence assumptions**

\[
h_{t+1} \triangleq f(h_t, x, y_t, w)
\]
\[ h_t = f(f(...f(h_2, x, y_i, w), x, y_i, w), ..., x, y_t-1, w) \]

define \( p(y_t | y_{1:t-1}, x) \) \( \propto \exp (c_h y_t h_t) \) as 

- model embedding 
- in LSTM and can be fine-tuned 

standard learning: using ML 

\[ \text{arg min}_{w, \tilde{w}} \frac{1}{n} \sum_{i=1}^{n} \log p(y^{(i)}_t | x^{(i)}) \]

\[ \leq \frac{1}{n} \sum_{i=1}^{n} \log p(y^{(i)}_t | y^{(i)}_{1:t-1}, x^{(i)}; w, \tilde{w}) \]

output of a deep NN

"teacher forcing" 

\[ \text{for ML, do SGD on objective} \]

\[ \text{gradient of } \frac{1}{n} \sum_{i=1}^{n} \log p(y^{(i)}_t | y^{(i)}_{1:t-1}, x^{(i)}; w, \tilde{w}) \]

"exposure problem" 

\[ \text{not shown} \]

"don't know" 

\[ p(y_t | y^{\uparrow}_{1:t-1}, x) \]

for decoding 

\[ \text{categorical } \leq \log p(y_t | y^{\uparrow}_{1:t-1}, x) \] 

\[ \Rightarrow \text{NP hard?} \]

\[ \Rightarrow \text{need approximation} \]

\[ \Rightarrow \text{greedy decoding: } \hat{y}_t = \text{argmax}_{y_t} \log p(y_t | y^{\uparrow}_{1:t-1}, x) \]

\[ \Rightarrow \text{beam search} \]

\[ \Rightarrow \text{greedy decoding with memory of size } k \]

\[ \Rightarrow \text{size of beam} \]

beam search: construct \( \hat{y}_1, ..., \hat{y}_t \) 

beam of size \( L \) (memory) 

- at step \( t \), you have \( L \) candidate solutions \( y^{(i)}_{1:t} \)

- expand possible next choices \( |X^{\uparrow}_{t+1}| \cdot L \)

- score \( \log p(y_{t+1} | \hat{y}_t^{(i)}, x) + \log p(y^{(i)}_{t+1} | x) \)

\[ \Rightarrow \] beam size \( L \) candidates as \( \hat{y}^{(i)}_{t+1} \).
\[ \text{score train} = \log p(Y_t | y_{1:t}, x) + \log p(y_{1:t}, x) \]

then keep top \( k \) candidates as \( y^{(e)}_{1:t} \)

V.S. Viterbi alg which does "backtracking" to correct past mistakes

**Seq2Seq algo: encoder/decoder architecture**

A useful way to get \( p(y_t | y_{1:t-1}, x) \) for a RNN when \( x \) has variable length.

```
       h^2
      /  \
     /    \
    h^1  x_{1:T}
      |  |
    y_{1:T}
```

**Issues:**

a) variable length output? --- end-of-sequence character

b) how to handle long sequence \( x \)?

   problem: need to summarize input sentence in one context vector of fixed length

   solution: "attention mechanism"

   c) vanishing gradient?

      * LSTM
      * gated recurrent unit (GRU)
      * etc.

   d) sequential processing

      \( \rightarrow \) fixed by "transformer architecture"