latent variables

- motivation: semantic segmentation
- segmentation is expensive $\rightarrow z$ "latent variable"
- perhaps only have class labels $\rightarrow y$
- also: Felzenszwalb & al. PAMI 2007
  "deformable part models" for object recognition
  $\rightarrow z$ there was an object part configuration

before, we had $s(x, y; w) = \langle w, \varphi(x, y) \rangle$

now, consider $s(x, y, z; w) = \langle w, \varphi(x, y, z) \rangle$

as before, could predict with argmax $s(x, y, z; w)$

$y \in \mathcal{Y}, z \in \mathcal{Z}$

learning $\rightarrow$ CAF approach $\leftarrow$ hidden CAF $f(y, z|x)$

similarity to latent variable modeling with graph model
  $\rightarrow$ marginalize $z$ out

$ML \rightarrow EM$ (expectation-maximization)

analog for latent-SVMstruct & CCCP

latent-SVMstruct

$\ell(y, (\hat{y}, \hat{z}))$

generalize structural hinge loss

$\ell(x, y, w) \leq \max_{\hat{y}, \hat{z}} \langle w, \varphi(x, \hat{y}, \hat{z}) \rangle + \ell(y, (\hat{y}, \hat{z})) - \max_{\hat{z} \in \mathcal{Z}} \langle w, \varphi(x, y, \hat{z}) \rangle \geq \ell(y, \hat{y}, \hat{z})$ (best score for ground truth)
Here $g(x, y, w) = u(w) - v(w)$ where $u$ and $v$ are convex functions.

"Difference of convex functions"

→ CCCP procedure is to approx. minimize this

**CCCP procedure:**

- Line up $v(w)$ at $x^*$ to get an upper bound
- $w_{t+1}$ is obtained by minimizing this upper bound
- Repeat a majorization-minimization procedure
  \( (EM\ is\ another\ example) \)

\[
\begin{align*}
\bar{S}_t(w) &= u(w) - \langle \nabla v(w), w - w_t \rangle \quad \text{for subgradient} \\
\bar{w}_{t+1} &= \text{argmin}_w \bar{S}_t(w)
\end{align*}
\]

Properties of procedure:
- like EM, descent procedure i.e. $S(k_{t+1}) < S(k_t)$
  \( \bar{S}_t(w) = S_t(w) \geq S_t(w_{t+1}) \geq S_t(w_{t}) \)

- local linear convergence to a stationary point
  \[ \text{for latent-SVM shrink} \]  
  \[ \text{[see Nemirovskii OPT cost function]} \]

* CCCP for 

\[
\begin{align*}
\bar{V}(w) &= \max_{\tilde{x}} < w, (x, y, \tilde{x}) > \\
&\quad \rightarrow \max_{\tilde{x}} \langle u_t(y, x, \tilde{x}) > + \langle v_t(y, \tilde{x}) > \\
&\quad \text{opt.}
\end{align*}
\]

\[
\bar{V}(w) = \max_{\tilde{x}} < w, (x, y, \tilde{x}) > + \langle v_t(y, \tilde{x}) > + \text{opt.}
\]

\[ \quad \text{like SVM shrink objective} \]

CCCP algorithm for latent SVM shrink:

- repeat:
  - * fill in $\bar{V}(w)$ for all ground truth $y^{(i)}$ using $w_t$
  - solve a standard SVM shrink in cost $\bar{V}$...
Deep Learning

I) Plug in 'deep learning' features in a structured prediction model

Example: OCR

so far \( C_{\theta}(x_t, y_t') = o(x_t) - y_{t}^{th} \)

instead \( y_t(x_t, y_t) = (o(NM(x_t)))/y_{t}^{th} \)

II) "End-to-end" training

structured prediction energy networks (SPENs)

III) Recurrent neural networks (RNN)

Motivation: \( p(y|x) \sim \prod_{t=1}^{T} p(y_t|y_{t-1}, x) \)

graphical modeling

\( y_t \perp y_{t+1}, y_{t+2}, \ldots \) (cond. ind.)

RNN: "structured parameterization" with no cond. indep. assumptions

\[ h_{t+1} = f(h_t, x_t, y_t, w) \]

\[ h_t = f(f(\ldots f(h_1, x_1, y_1, w), x_1, y_1, \ldots), x_t, y_{t-1}, w) \]

define \( p(y_t|y_{t-1}, x) \) graph \( c(y_t) \) (HMM)

\[ h_t \rightarrow h_{t+1} \rightarrow h_{t+2} \rightarrow \ldots \rightarrow h_T \]

\[ y_t \rightarrow y_{t+1} \rightarrow \ldots \rightarrow y_T \]

\[ p(y_t|u, \ldots, x) \]
Standard learning: using ML

\[ \min_{W} \frac{1}{n} \sum_{i=1}^{n} \log p(y(i) | x(i); W) \]

\[ \frac{\sum \log p(y(i) | y_{i+1:i+n}, x(i); W, \hat{W})}{t} \]

Output of a deep NN

"Teacher forcing"

Exposure problem

ie. dont know \[ p(y_t | y_{1:t-1}, x) \]

\[ p(y_t | y_{1:t-1}, x) \]

for ML, do SGD gradient

\[ \frac{\partial \log p(y(i) | y_{1:i+n}, x(i); W, \hat{W})}{\partial W} \]

we backpropagation

Decoding: \[ \max_{y \in \mathbb{Y}} \sum_{t=1}^{T} \log p(y_t | y_{i+1:i+n}, x(i)) \]

\[ \sum_{t=1}^{T} \log p(y_t | y_{i+1:i+n}, x(i)) \] \to NP hard

-- need approximation

Greedy decoding: \[ \hat{y}_t = \arg \max_{y \in \mathbb{Y}} p(y_t | y_{1:t-1}, x) \]

Beam search: construct \( \hat{y}_1, \ldots, \hat{y}_T \)

Beam of size L (memory)

at step t, you have L candidate solution prefixes \( y_{1:t} \)

\( y_{1:t} \)

Expend possible next choice \( y_{t+1:1} \cdot L \)

Score from (e.g. \( \log p(y_{t+1} | y_{1:t+1}, x) + \log p(y_{t+1} | x) \))

down keeps top L candidates as \( \hat{y}_{1:t} \)

\[ \forall s, \]

IFT6132 Page 4
Vitioli d.g. which does "backtracking" to correct past mistakes.

**Seq2Seq a.k.a. encoder/decoder architecture**

A useful way to get $p(y | y_{1:i-1}, x)$ for a RNN.

When $x$ has a variable length.

```
encoder RNN
```

```
\[ h_1 \rightarrow h_2 \rightarrow \ldots \rightarrow h_i \rightarrow h \]
```

```
encoder RNN
```

```
\[ x \rightarrow \ldots \rightarrow x \]
```

```
fixed size representation
```

```
decoder RNN
```

```
\[ h_1 \rightarrow h_2 \rightarrow \ldots \rightarrow h_i \rightarrow h \]
```

```
\[ y_1 \rightarrow y_2 \rightarrow \ldots \rightarrow y_i \rightarrow y \]
```

---

**Issues:**

a) Variable length output? → end-of-sequence character
   how to handle
b) Long input 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..