today: latent variable SVMstruct - CCCP
  * deep learning

\[ \text{prox}_{\lambda} (\mathbf{w}_t - \lambda \left[ \mathbf{D}_i^t (\mathbf{w}_t) - \mathbf{g}_i^t + \frac{1}{n} \sum_{j=1}^{n} \mathbf{g}_j^t \right]) \]

→ what is used by default in Scikit-learn for lasso

latent variables

motivation: semantic segmentation → fluid boundary of different objects

\[ \text{segmentation is expensive} \rightarrow z \text{ "latent variable"} \]

perhaps only have class labels \( y \)

also: [Felzenszwalb et al., TPAMI 2010]

"deformable part models" for object recognition

\( z \) there was an object
before, we had \( S(x, y, w) = \langle w, \ell(x, y) \rangle \)

now, consider \( S(x, y, z, w) = \langle w, \ell(z, y, z) \rangle \)

as before, we define \( S(x, y, z, w) \) as before, we could predict with argmax \( S(x, y, z, w) \)

\[ y \in y, z \in Z \]

* CRF \( (p(y|x)) \) --- hidden CRF \( p(y,z|x) \)

similar to latent variable modeling with graph model

ML \( \rightarrow \) expectation-maximization (EM)

analog for latent sumstruct is CCCP

Latent Sumstruct

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

generalization of structured hinge loss:

\[ l(x, y, w) \triangleq \max_{\tilde{y}, \tilde{z}} \langle w, \ell(x, \tilde{y}, \tilde{z}) \rangle + \ell(y, (\tilde{y}, \tilde{z})) - \max_{\tilde{z} \in Z} \langle w, \rho(x, y, \tilde{z}) \rangle \geq l(y, h_0(x)) \]

here \( l(y, z, w) = \ell(w) - \ell(w) \) where \( u, v \) are convex sets of \( w \)
"Difference of convex functions"

CCCP procedure

- linearize \( v(w) \) at \( w_t \) to get an upper bound
- minimizing the upper bound
- repeat

\[ s_t(w) = u(w) - \langle \nabla u(w), w - w_t \rangle + \frac{1}{2} \langle \nabla^2 u(w)(w - w_t), w - w_t \rangle \]

(\( w_{t+1} = \underset{w}{\text{argmin}} \ s_t(w) \))

- a majorization-minimization procedure
  
  (EM is another example)

Properties of this procedure:

- Like EM, descent procedure i.e. \( s_t(w_{t+1}) \leq s_t(w_t) \)
  \[ s_t(w_t) = s_t(w_t) \geq s_t(w_{t+1}) = s_{t+1}(w_{t+1}) \]

- Local linear convergence to a stationary point [See NIPS 2012 paper]

* for SUMS

\[ v(w) = \max \langle w, y(u) \rangle \]
\[ v(w) = \max_{z} \langle w, \gamma(z, y) \rangle \]

\[ \Theta(w_t) = \{ \gamma(x, y, z(x, w_t)) \} \]

\[ \Rightarrow \delta_t(z(y, w)) = \max_{\phi \gamma \delta} \langle \phi, \gamma(y, \delta) \rangle + \text{cost}. \]

\[ \text{The sumshift objective} \]

CCCP for latent sumshift: repeat: 
1. solve standard sumshift to get \( w_t \)
2. repeat

\[ \text{Deep learning} \]

\[ \text{go from } \langle w, \gamma(z, y) \rangle \text{ to } \langle w, \gamma(z, y, \delta) \rangle \]

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

Example: OCR

\[ \text{images} \rightarrow \gamma(z, y, \delta) \text{ instead of } \gamma(z, y) \]

\[ \text{example } [\text{Vu et al., ICCV 2015}] \]

"context-aware cues for pose and head detection"

learned on images e.g.
1) "end-to-end" training: structured prediction energy network (SPEN)
2) recurrent neural networks (RNN)

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

Using a NN:

\[ h_{t+1} = \sigma(h_t, x, y_t, W) \]

\[ h_t = \sigma(\sigma(\ldots (h_0, \ldots ), x, y_{t-1}, W) \]

Algorithm: \[ p(y_t|y_{1:t-1}, x) \propto \exp(c(y_t)^T W h_t) \]

Standard learning: use maximum likelihood

\[ \min_w \sum_i \log p(y_i|x_i) \]

"teacher's forcing"
Clear SGD on this

Chain rule $\Rightarrow$ backpropagation

Decoding: $\operatorname{argmax}_{y_1:y_T} \log p(y|\mathbf{x}) \Rightarrow$ NP hard

Need approximation $\Rightarrow$ greedy decoding $\hat{y}_t = \operatorname{argmax}_{y_t \in \mathcal{Y}} p(y_t|\hat{y}_{1:t-1}, \mathbf{x})$

Beam search

Beam search $\Rightarrow$ construct $y_1, \ldots, y_T$

Beam of size $L$ (memory)

At step $t$, you have $L$ candidate solution prefixes $y_1: t$

Expand possible next choice: $L \cdot |\mathcal{Y}_{t+1}|$

Score them (e.g. $p(y_{t+1} | y_1:t)$)

Then keep top $L$ candidates as $rac{1}{2} \sum_{y_{t+1} \in \mathcal{Y}_{t+1}} \log p(y_{t+1} | y_1:t)$
VS. Viterbi algo. which does "backtracking" to correct past mistakes

Seq2seq | encoder/decoder

A useful way to get $p(y_t | x_{t-1:t})$ for a RNN

When $x$ can be variable length

Issues:

a) Variable length output?
   - end of sequence special character

b) Long input sequence $x$?
problem: needs to be summarized in fixed-length context vector

solution: "attention mechanism"

c) vanishing gradient?

- LSTM
- gate recurrent unit (GRU)
- etc...