# Optimizing Weighted Lower Linear Envelope Potentials Within Latent-SVM Framework

Chang Li

11. August 2016

## 1 Algorithm

The higher-order energy function is:

$$\begin{aligned} E^c(y_c, z) &= a_1 W_c(y_c) + b_1 + \sum_{k=1}^{K-1} z_k((a_{k+1} - a_k)W_c(y_c) + b_{k+1} - b_k) \\ &= a_1 W_c(y_c) + \sum_{k=1}^{K-1} (a_{k+1} - a_k) z_k W_c(y_c) + \sum_{k=1}^{K-1} (b_{k+1} - b_k) z_k \end{aligned}$$

It can be written as $E^c(y_c, z) = \theta^h \phi^h$ where

$$\theta_i^h = \begin{cases} a_1 & \text{for } i = 1 \\ a_i - a_{i-1} & \text{for } 1 < i \leq K \\ b_{i+1-K} - b_{i-K} & \text{for } K < i \leq 2K - 1 \end{cases} \tag{1}$$

$$\phi_i^h = \begin{cases} W(\mathbf{y}) & \text{for } i = 1 \\ W(\mathbf{y}) \left[\left[ i - 1 \leq k^* \right]\right] & \text{for } 1 < i \leq K \\ \left[\left[ i - K \leq k^* \right]\right] & \text{for } K < i \leq 2K - 1 \end{cases}$$

Therefore, the energy function (higher order features together with unary and pairwise features) are:

$$E^{all}(y, z) = \begin{bmatrix} \theta^h \\ \theta^{unary} \\ \theta^{pairwise} \end{bmatrix}^T \cdot \begin{bmatrix} \phi^h \\ \phi^{unary} \\ \phi^{pairwise} \end{bmatrix} = \theta^{all\mathrm{T}} \cdot \phi^{all} \tag{2}$$

where $\theta^{all}, \phi^{all} \in R^{2K+1}$. Therefore, we have the graph-cut method for inference latent variable:

$$\Delta((y_i, h_i^*(w)), (\hat{y}_i(w), \hat{h}_i(w))) \leq \left( \max_{(\hat{y}, \hat{h}) \in \mathcal{Y} \times \mathcal{H}} [w \cdot \phi(x_i, \hat{y}, \hat{h}) + \Delta(y_i, \hat{y}, \hat{h})] \right) - \max_{h \in \mathcal{H}} w \cdot \phi(x_i, y_i, h)$$

Let $\mathbf{z}_i^*(w) = \arg\max_{\mathbf{z}\in\mathcal{Z}} w \cdot \Psi(\mathbf{y}_i, \mathbf{z})$

$$(\hat{\mathbf{y}}_i(w), \hat{\mathbf{z}}_i(w)) = \underset{(\mathbf{y}\times\mathbf{z})\in\mathcal{Y}\times\mathcal{Z}}{\arg\max}\; w \cdot \Psi(\mathbf{y}, \mathbf{z})$$

**Algorithm 1** MRF-LSSVM (CCCP-Cutting Plane)

*Outer Loop*:<br>
$i \leftarrow patlen$<br>
*top*:<br>
**if** $i > stringlen$ **then return** false<br>
$j \leftarrow patlen$<br>
*loop*:<br>
**if** $string(i) = path(j)$ **then**<br>
    $j \leftarrow j - 1$.<br>
    $i \leftarrow i - 1$.<br>
    **goto** *loop*.<br>
    **close**;<br>
$i \leftarrow i + \max(delta_1(string(i)), delta_2(j))$.<br>
**goto** *top*.