EN.601.783: Vision as Bayesian Inference
Homework 3

Ha Bui
hbui13@jhu.edu

Spring 2023

Exponential Models with Hidden Variables. 30 points

1. What is an exponential model with hidden variables? What are the potentials, the parameters, and the normalization term Z?. What are the derivatives of Z with respect to the parameters of the exponential model? How can the EM algorithm be used for doing maximum likelihood estimation for an exponential model with hidden variables?

Solution.

- An exponential model with hidden variables is an exponential distribution representing probability distributions

$$\mathbb{P}(d, h \mid \lambda) = \frac{1}{Z[\lambda]} \exp\{\lambda \cdot \phi(d, h)\}.$$

- The potentials sufficient statistics of hidden variable h and observed variable d are $\phi(h, d)$, the parameters are λ, and the normalizing term is $Z[\lambda] = \sum_{d,h} \exp\{\lambda \cdot \phi(d, h)\}$.
- The derivatives of Z w.r.t. parameters λ are

$$\frac{\partial \log Z[\lambda]}{\partial \lambda} = \sum_{d,h} \phi(d, h) p(d, h \mid \lambda).$$

- The EM algorithm can be used for doing MLE to lean $\mathbb{P}(d, h \mid \lambda)$ by the following steps:

- Step 1 (Expectation): Compute $p(d \mid \lambda) = \sum_h \mathbb{P}(d, h \mid \lambda)$. This can be computed by Dynamic-Programming (DP) (sum). If closed loops, approximated Belief Propagation (BP). (sum-product).
- Step 2 (Maximization): Compute $\hat{h} = \arg\max_h \mathbb{P}(d, h \mid \lambda)$. this can be computed by DP (max). If closed loops, approximated by BP (max-product).
- Step 3 (Learning λ): Given data $D = \{d^m : m = 1, \cdots, M\}$, compute

$$\hat{\lambda} = \arg\max_{\lambda} \Pi_{m=1}^{M} \mathbb{P}(d^m \mid \lambda) = \arg\max_{\lambda} \Pi_{m=1}^{M} \sum_{h^m} \mathbb{P}(d^m, h^m \mid \lambda).$$

2. Now suppose that the exponential model can be expressed as a probability distribution on variables defined on a graph. How can the EM computations be done if the graph has no closed loops? What if the graph has closed loops? Does the EM algorithm attempt to make the statistics of the data equal to the expected statistics of the model?

Solution.

- For the variables $h = (h_1, \cdots, h_n)$ and $d = (d_1, \cdots, h_n)$ defined by a graph with no closed loops (e.g., Figure 1), the EM follows the above EM algorithm in the previous solution w.r.t. the statistics following

$$\phi(d, h) = \begin{Bmatrix} \phi(d_i, h_i) : i = 1, \cdots, N, \\ \psi(h_i, h_{i+1}) : i = 1, \cdots, N-1 \end{Bmatrix}$$

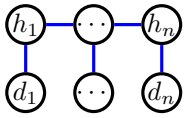

Figure 1: Example of a graph has no closed loops.

and the parameter follows

$$\lambda = \begin{Bmatrix} \lambda_i : i = 1, \cdots, N, \\ \mu_i : i = 1, \cdots, N-1 \end{Bmatrix}.$$

And therefore, we do EM algorithm to do MLE to learn the exponential distribution

$$\mathbb{P}(d, h \mid \lambda) = \frac{\exp\left(\sum_{i=1}^{N} \lambda_i \cdot \phi(d_i, h_i) + \sum_{i=1}^{N-1} \mu_i \cdot \psi(h_i, h_{i+1})\right)}{\sum_{d,h} \exp\left(\sum_{i=1}^{N} \lambda_i \cdot \phi(d_i, h_i) + \sum_{i=1}^{N-1} \mu_i \cdot \psi(h_i, h_{i+1})\right)}.$$

- If the graph has closed loops, then the normalization factor $Z[\lambda]$ is intractable, as a result, we can not use DP to compute $p(d \mid \lambda)$ at the Expectation and $\hat{h}$ at the Maximization step. And we will need to approximate these by sampling techniques such as BP.
- Yes, the EM algorithm attempt to make the statistics of the data equal to the expected statistic of the model because it does MLE for $p(d, h \mid \lambda)$, i.e.,

$$\hat{\lambda} = \arg\max_{\lambda} \Pi_{m=1}^{M} \mathbb{P}(d^m \mid \lambda) = \arg\max_{\lambda} \Pi_{m=1}^{M} \sum_{h^m} \mathbb{P}(d^m, h^m \mid \lambda),$$

which is equivalent to minimizing the KL divergence between empirical data $\mathbb{P}_{emp}(D)$ and model distribution $\mathbb{P}_{\lambda}(D)$ for $D = \{d^m : m = 1, \cdots, M\}$.

3. Describe how a Hidden Markov Model (HMM) can be expressed as an exponential model with hidden variables? Does this correspond to a graph with closed loops? What inference algorithms are used?

Solution.

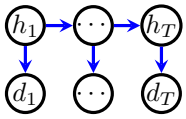

Figure 2: Example of a graph with HMM.

- The HMM can be expressed by following the hidden variables h and observed variables d in the previous solution w.r.t the sequence from $1 \to T$ of the hidden

$$h = (h_1, \cdots, h_T)$$

and the observed sequence

$$d = (d_1, \cdots, d_T)$$

from the graph 2. As a result, the exponential model with these hidden variables become

$$\mathbb{P}(d, h \mid \lambda) = \frac{\exp\left(\sum_{t=1}^{T} \lambda_t \cdot \phi(d_t, h_t) + \sum_{t=1}^{T-1} \mu_t \cdot \psi(h_t, h_{t+1})\right)}{\sum_{d,h} \exp\left(\sum_{t=1}^{T} \lambda_t \cdot \phi(d_t, h_t) + \sum_{t=1}^{T-1} \mu_t \cdot \psi(h_t, h_{t+1})\right)},$$

with the statistic $\psi(h_t, h_{t+1}) = I(h_t^k = s_i \text{ AND } h_{t+1}^k = s_j)$ with the state $s_i, s_j, i, j = 1, \cdots, N$ and the statistic $\phi(d_t, h_t) = I(h_t = s_i \text{ AND } d_t = v_m)$ with the observation v_m.
- No, the graph 2 correspond to a graph with no closed loops.
- With a no closed loops graph, we can use Viterbi Algorithm (Dynamic Programming) for inference by using $\psi_t(j)$ to keep track of the state that maximizes $\mathbb{P}(d_1, \cdots, d_t = s_i, h_1, \cdots, h_t \mid \lambda)$ at time $t-1$, we also can use Baum-Welch algorithm (EM) algorithm in the previous solution to recursively solve $\hat{h}$ at each step t, and finally use $\mathbb{P}(d \mid \lambda)$ for model selection.

Lighting Models: 30 points

1. What is the Lambertian lighting model? What is the albedo? What is the Generalized Bas Relief (GBR) ambiguity? How does it relate to the ambiguity between convex and concave shapes? What are cast and attached shadows? Does the GBR hold for cast and attached shadows?

Solution.

- The Lambertian reflectance model is a way to model how images are generated from three-dimensional objects illuminated by various light sources, which is formalized by

$$I(\vec{x}) = a(\vec{x})\vec{n}(\vec{x}) \cdot \vec{s},$$

where $I(\vec{x})$ is the image, $a(\vec{x})$ is the albedo, $\vec{n}(\vec{x})$ is the surface normal, $\vec{s}$ is the light source.
- The albedo $a(\vec{x})$ reflects radiance from a perfectly diffuse surface $\vec{n}$ when lit uniformly by the light $\vec{s}$ of unit radiance.
- The Generalized Bas Relief (GBR) ambiguity is the effect that corresponds to a transformation on the surface of the form:

$$z(x, y) \mapsto \lambda z(x, y) + \mu x + \nu y,$$

where $\lambda z(x, y)$ is a bas relief transformation which is known to sculptures with surface $z(x, y)$, point on z is $(x, y, z(x, y))$, and $\mu x + \nu y$ correspond to adding a plane.
- It relates to the ambiguity between a convex object lit from above and a concave object lit from below with multiple light sources (including attached shadows).
- The cast shadows $I(\vec{x}) = \sum_\mu \max\left\{\vec{b}(\vec{x}) \cdot \vec{s}_\mu, 0\right\}$ are surface patches caused by the blockage of light from a light source by the object. The attached shadows $I(\vec{x}) = \sum_\mu \min\left\{a(\vec{x})\vec{n}(\vec{x}) \cdot \vec{s}_\mu, 0\right\}$ are surface patches facing away from the light source.
- The GBR holds for attached shadows by handling convex, Lambertian objects with multiple light sources. However, it does not model cast shadows by do not handle the inter-reflections.

2. What is photometric stereo? How can it be formulated in terms of Singular Value Decomposition? What are the ambiguities?

Solution.

- Photometric stereo is a technique to estimate the surface normals of objects by observing several images of the Lambertian object under varying lighting.
- It can be formulated in terms of Singular Value Decomposition (SVD) by assuming a single directional source $M = LS$ where M is the set of images and L is the corresponding light sources and S is the nine basis vectors of the surface. Then we can apply SVD to find $M = U\Sigma V^\top$ (U, V are unitary matrixes, Σ is the diagonal matrix with singular values of M) by approximating a rank-3

$$\hat{M} = (\hat{L}A^{-1})(A\hat{S}),$$

where $\hat{L} = U\sqrt{\Sigma_3}$, $\hat{S} = \sqrt{\Sigma_3}V^\top$, and $\Sigma_3 = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$ with $\sigma_1, \sigma_2, \sigma_3$ are the largest singular values of M.
- The ambiguities are impossible things to distinguish from the geometry of the object being viewed like between a convex object lit from above and a concave object lit from below.

3. How many bases are required to describe the image of an object if shadows are ignored? How can principal component analysis (PCA) be used to estimate the number of bases? What theory predicts the number of bases for convex objects? And how many bases are predicted?

Solution.

- If shadows are ignored, then we only need three bases to describe the image of an object by $I(\vec{x}) = \sum_{i=1}^{3} \alpha e_i(\vec{x})$ such a the model can help for recognizing objects and for tracking an object when the lighting varies.
- To estimate the number of bases $\{I^\mu(\vec{x})\}$, we can use PCA to calculate the eigenvectors and eigenvalues

$$\sum_{\vec{x}'} K(\vec{x}, \vec{x}')e(\vec{x}') = \lambda e(\vec{x}),$$

where the correlation matrix $K(\vec{x}, \vec{x}') = \frac{1}{N}\sum_{\mu=1}^{N} I^\mu(\vec{x})I^\mu(\vec{x}')$.
- For convex objects, the Illumination cone theory predicts the number of bases for convex objects by forming the convex cone in $\mathbb{R}^p$, where p is the number of pixels.
- For convex objects, it predicts nine eigenvectors which captured much of the intensity variations and, if the lighting was restricted to come from the frontal hemisphere then only five were needed.

AdaBoost: 30 points

1. What is a weak classifier? What is a strong classifier?

Solution.

- In the context of AdaBoost, given labeled data $\mathcal{X} = \{(x^i, y^i) : i = 1, \cdots, N\}$ with $y^i \in \{\pm 1\}$, a weak classifier is $\phi_\mu(x)$ which is in the set of weak classifier $\{\phi_\mu(x) : \mu = 1, \cdots, M\}$.
- The strong classifier is a plane in feature space, i.e., $S(x) = sign(\sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(x))$, where $\{\lambda_i\}$ are weights to be learned.

2. What function does AdaBoost minimize? What is its relationship to the loss function for binary classification? How does the update rule correspond to weighting missclassified samples more highly? Can a weak classifier be selected twice by AdaBoost?

Solution.

- AdaBoost algorithm minimizes the objective function

$$Z[\lambda_1, \cdots, \lambda_M] = \sum_{i=1}^{N} \exp\left\{-y^i \sum_{\mu=1}^{M} \lambda_\mu \phi_\mu(x^i)\right\}. \tag{1}$$

- Since function $Z[\lambda_1, \cdots, \lambda_M]$ in Equation 1 is the convex upper bound of the empirical risk of the strong classifier $S(.)$

$$R_{emp} = \sum_{i=1}^{N} \left\{1 - I(S(x^i) = y^i)\right\}.$$

As a result, this will relate to the binary classification by the indicator term $I(\cdot)$, i.e., terms in the empirical risk take value 1, if $S(x^i) \neq y^i$ and value 0 if $S(x^i) = y^i$.

- To weighting missclassified samples more highly, the algorithm updates $\lambda_{\hat{\mu}}^{t+1} = \hat{\lambda}_\mu$, $\lambda_\mu^{t+1} = \lambda_\mu^t$, for all $\mu \neq \hat{\mu}$, where $\hat{\lambda}_\mu$ is the solution of $\frac{\partial Z}{\partial \lambda_\mu} = 0$ for each μ and $\hat{\mu}$ is the maximal decrease in Z, i.e.,

$$\hat{\mu} = \arg\min_{\mu} Z\left[\lambda_1^t, \cdots, \hat{\lambda}_\mu, \cdots, \lambda_M^t\right].$$

- Yes, the weak classifier can be selected twice by AdaBoost because of the fact that once one weak classifier is selected at step t, it can be selected again in later steps $t+$.

3. What are cascades? How are they used for face and text detection? What types of image features and weak classifiers are used by AdaBoost to perform face detection? What types of features are used by AdaBoost to perform text detection?

Solution.

- Cascades are classifier functions in ensemble learning (bootstrapping statistics), that are trained with several "positive" sample views of a particular object and arbitrary "negative" images of the same size.
- Face and text detectors use these cascades to construct a cascade of homogeneous classifiers, which can reject most of the negative examples at the early stages of processing thereby significantly reducing computation time.
- AdaBoost uses integral image representation and simple rectangular features (two-rectangle feature with horizontal & vertical, three-rectangle feature, and four-rectangle feature). And, it uses cascading weak classifiers to perform face detection.
- AdaBoost uses unconstrained scene features and selects informative features (invariant for text regions and discriminating between text & non-text regions) with consideration of computation cost to perform text detection.

Support Vector Machines: 30 points

1. What is the margin of an SVM? How does SVM deal with non-separable data? What is the primal formulation of SVM? How does the SVM objective function relate to the empirical risk? What term helps prevent over-fitting to the training data? What is the hinge loss? How does learning an SVM differ from learning Gaussian distributions for the positive and negative data examples and then applying the log-likelihood rule?

Solution.

- The margin C of an SVM is the distance between the hyperplane $\left\langle \vec{x} : \vec{a} \cdot \vec{x} + \vec{b} = 0 \right\rangle$, $|\vec{a}| = 1$ and the observations closest $(\vec{x}_\mu, y_\mu) \in \{(\vec{x}_\mu, y_\mu) : \mu = 1 \to N\}$ to the hyperplane (support vectors).
- For non-separable data, SVM can use kernel tricks to map non-separable to a higher separable dimension. Besides that, if we allow some data points to be misclassified, SVM can use slack variables $\{z_1, \cdots, z_n\}$ to allow data points to move in direction $\vec{a}$ so that they are on the right side of the margin, i.e.,

$$y_u \{(\vec{x}_\mu + C z_\mu \vec{a} \cdot \vec{a} + b)\} \geq C.$$

- The primal formulation of SVM is to find minimizer primal variable $\vec{a}, z$ and maximizer dual variables α, τ of the Lagrange multipliers:

$$L_p(\vec{a}, b, z; \alpha, \tau) = \frac{1}{2}\vec{a} \cdot \vec{a} + \gamma \sum_\mu z_\mu - \sum_\mu \alpha_\mu \{y_\mu(\vec{x}_\mu \cdot \vec{a} + b) - (1 - z_\mu)\} - \sum_\mu \tau_\mu z_\mu,$$

where $\{\alpha_\mu\}$ and $\{\tau_\mu\}$ are Lagrange parameters needed to enforce the inequality constraints $\alpha_\mu \geq 0, \tau_\mu \geq 0, \forall \mu$.
- The SVM objective function, i.e., max-margin criterion

$$\min \frac{1}{2} \sum \vec{a} \cdot \vec{a} + \gamma \sum_\mu z_\mu \text{ s.t. } y_\mu(\vec{x}_\mu \cdot \vec{a} + b) \geq 1 - z_\mu, \forall \mu z_\mu \geq 0$$

can be re-expressed as the sum of the empirical risk plus a term $\frac{1}{2}|\vec{a}|^2$, multiplied by a constant $\frac{1}{\gamma N}$, i.e.,

$$\frac{L_p}{\gamma N} = \frac{1}{2\gamma N}|\vec{a}|^2 + \frac{1}{N}\sum_{\mu=1}^{N} \max\{0, 1 - y_\mu(\vec{x}_\mu \cdot \vec{a} + b)\}.$$

- The first term $\frac{1}{2\gamma N}|\vec{a}|^2$ prevent over-fitting to the training data because it penalizes decision rules $\hat{y}(\vec{x}) = \text{sign}(\vec{x} \cdot \vec{a} + b)$ which have large $|\vec{a}|$, as a result, it restricts the set of rules and are more likely to generalize to new data.
- Let $\phi(x) = x$ is the optimal hyperplane and $\Delta(y_i, \hat{y}_i(\lambda))$ is the error function measure distance between the true solution y_i & the estimate $\hat{y}_i(\lambda) = \arg\max_y \lambda \cdot \phi(x_i, y)$ with learnable parameter λ. Then for the Binary case, i.e., $y_i \in \{-1, 1\}$, we have the hinge loss is

$$\Delta(y_i, \hat{y}_i(\lambda)) = \max\langle 0, 1 - y_i \lambda \cdot \phi(x_i)\rangle$$

in $R(\lambda) = \frac{1}{2}||\lambda||^2 + c\sum_{i=1}^{M} \Delta(y_i, \hat{y}_i(\lambda))$ where $\phi(x, y) = y\phi(x)$.
- When compared to applying Gaussian distributions to lean positive and negative data examples with the log-likelihood rule, the SVM algorithm differs by it maximizes the margin C between the closest support vectors, which is based on metric learning and are deterministic approach. Meanwhile, applying the log-likelihood ratio with two Gaussian belongs to the probabilistic approach.

2. What is a kernel? How does it relate to feature vectors? What type of kernel makes an SVM behave like a nearest neighbor classifier?

Solution.
- The kernel $k(x, x') = \langle \phi(x), \phi(x')\rangle$ is a mapping such that transforms linearly inseparable data of x, x' into linearly separable data, thus finding an optimal boundary λ for possible outputs.
- From equation $k(x, x') = \langle \phi(x), \phi(x')\rangle$, we can see that kernel is the inner product of features vector $\phi(x)$ and $\phi(x')$.
- The Radial basis function kernel

$$K(x, x') = \exp\left(-\frac{||x - x'||^2}{2\sigma^2}\right)$$

performs as the nearest neighbor classifier because we can see the classifier $\sum_\mu \alpha_\mu y_\mu e^{||x_\mu - x||^2} + \hat{b}$ behave similarly to finding the squared Euclidean distance $||x_\mu - x||^2$ in nearest neighbor techniques.

3. The primal formulation is given by $L_p(\vec{a}, b, \{z_i\}; \{\alpha_i, \mu_i\}) = (1/2)|\vec{a}|^2 + \gamma\sum_{i=1}^m z_i - \sum_{i=1}^m \alpha_i\{y_i(\vec{a} \cdot \vec{x}_i + b) - (1 - z_i)\} - \sum_{i=1}^m \mu_i z_i$. Explain the meaning of all the terms and variables in this equation. What constraints do the variables satisfy? Calculate the form of the solution $\vec{a}$ by minimizing L_p with respect

to $\vec{a}$. What are the support vectors? How can the dual formulation be obtained by eliminating $\vec{a}, b, \{z_i\}$ from L_p. How can the primal problem be solved by exploiting the dual formulation?

Solution.

- In the case of primal formulation follow

$$L_p(\vec{a}, b, \{z_i\}; \{\alpha_i, \mu_i\}) = (1/2)|\vec{a}|^2 + \gamma \sum_{i=1}^{m} z_i - \sum_{i=1}^{m} \alpha_i \{y_i(\vec{a} \cdot \vec{x}_i + b) - (1 - z_i)\} - \sum_{i=1}^{m} \mu_i z_i.$$

We have $\vec{a}, \{z_i\}_{i=1}^m$ are primal variables that L_p should be minimized by the two first terms, i.e., $(1/2)|\vec{a}|^2 + \gamma \sum_{i=1}^m z_i$. Meanwhile $\{\alpha_i, \mu_i\}$ are dual variables that L_p should be maximized by the last two terms, i.e., $\sum_{i=1}^m \alpha_i \{y_i(\vec{a} \cdot \vec{x}_i + b) - (1 - z_i)\}$ and $\sum_{i=1}^m \mu_i z_i$.

- The constraints for these variable includes $y_i(\vec{a} \cdot \vec{x}_i + b) \geq 1 - z_i$, $z_i \geq 0$, $\mu_i \geq 0$, and $\alpha_i \geq 0$ for all $i = 1 \rightarrow m$.

- To minimize L_p w.r.t. $\vec{\alpha}$, take partial derivative w.r.t. $\vec{\alpha}$ and we get

$$\frac{\partial L_p}{\partial \vec{a}} = \vec{a} - \sum_{i=1}^{m} \alpha_i y_i \cdot \vec{x}_i.$$

Therefore, L_p is minimized when $\frac{\partial L_p}{\partial \vec{a}} = 0$, i.e.,

$$\hat{\vec{a}} = \sum_{i=1}^{m} \alpha_i y_i \cdot \vec{x}_i. \tag{2}$$

- The support vectors are $\vec{x}_i$ for which $\alpha_i \neq 0$ in Equation 2 because this Equation shows that the solution of $\frac{\partial L_p}{\partial \vec{a}} = 0$ depends only on these vectors.

- The dual formulation can be obtained by eliminating $\vec{a}, b, \{z_i\}$ from L_p by re-formalize L_p by

$$L_p = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j \text{ s.t. } 0 \leq \alpha_i \leq \mu_i, \sum_{i=1}^{m} \alpha_i y_i = 0.$$

By knowing $\{\hat{\alpha}_i\}$, it will give us the solution $\hat{\vec{a}} = \sum_{i=1}^m \hat{\alpha}_i y_i \vec{x}_i$.

- The primal problem be solved by exploiting the dual formulation by re-write L_p as

$$L_p = -(1/2)|\vec{a}|^2 + \sum_{i}^{m} \alpha_i + \vec{a} \cdot (\vec{a} - \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i) + \sum_{i=1}^{m} z_i(\gamma - \mu_i - \alpha_i) - b \sum_{i=1}^{m} \alpha_i y_i.$$

Extremize w.r.t. $\vec{a}, b, \{z_i\}$, we obtain

$$\hat{\vec{a}} = \sum_{i=1}^{m} \alpha_i y_i \vec{x}_i, \sum_{i=1}^{m} \alpha_i y_i = 0, \gamma - \mu_i - \alpha_i = 0.$$

Substituting back into L_p gives:

$$L_p = -\frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} \alpha_i \alpha_j y_i y_j \vec{x}_i \vec{x}_j + \sum_{i=1}^{m} \alpha_i$$

which has to be maximized w.r.t. $\{\alpha_i\}$.

Structured Latent Support Vector Machine 30 points

1. Describe how SVM's can be extended to learning the parameters for a multi-class classification problem where the output is of form $\vec{y}^* = \arg\max_{\vec{y}} \lambda \cdot \phi(\vec{d}, \vec{y})$, where $\vec{d}$ is the input and $\vec{y}$ is the output.

Solution.

- In the case of the vector output $\vec{y} \in \mathbb{R}^n$ and the decision rule has the form $\vec{y}^* = \arg\max_{\vec{y}} \lambda \cdot \phi(\vec{d}, \vec{y})$, we can extend the learning process as minimizing the risk

$$R(\lambda) = \frac{1}{2}||\lambda||^2 + c \sum_{i=1}^{M} \Delta(\vec{y}_i, \vec{y}_i^*),$$

where $\Delta(\vec{y}, \vec{y}^*)$ is is any measure of distance between the true solution $\vec{y}_i$ and the estimate $\vec{y}_i^* = \arg\max_{\vec{y}} \lambda \cdot \phi(\vec{d}_i, \vec{y})$. To reduce the complications of $\Delta(\vec{y}_i, \vec{y}_i^*)$, the learning reduce to minimizes convex upper bound $\hat{R}(\lambda)$

$$\hat{R}(\lambda) = \frac{1}{2}||\lambda||^2 + c\sum_{i=1}^{M} \max\{\Delta(\vec{y}_i, \vec{y}^*) + \lambda \cdot \phi(\vec{d}_i, \vec{y}^*) - \lambda \cdot \phi(\vec{d}_i, \vec{y}_i)\}.$$

2. Now consider an SVM learnt over variable defined on a graph structure (e.g., like an HMM). How does the potential terms $\phi(\vec{d}, \vec{y})$ determine the connections/edges in the graph structure? How does the form of $\lambda \cdot \phi(\vec{d}, \vec{y})$ affect the difficulty of the inference and learning.

Solution.

- In the case of SVM learned over variable defined on a graph structure $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with vertexes $\mathcal{V}$ and edges $\mathcal{E}$, the potential term $\phi(\vec{d}, \vec{y}) \in \mathcal{E}$ determine the connections/edges between $\vec{d}, \vec{v} \in \mathcal{V}$.
- The form $\lambda \cdot \phi(\vec{d}, \vec{y})$ to compute $\vec{y}_i^* = \arg\max_{\vec{y}} \lambda \cdot \phi(\vec{d}_i, \vec{y})$ in inference and learning depends on whether this is a graph with closed loops or not. If no closed loops, we can use DP. If closed loops we would need an approximate algorithm like mean field theory or BP.

3. Describe how to learn latent SVMs where some output variables $\vec{y}$ are observed, some variables $\vec{h}$ are hidden/latent, and the input data is $\vec{d}$. How can this be expressed in terms of minimizing an energy function which is composed of a convex and a concave term? How can CCCP be used to perform the learning?

Solution.

- Given output variables $\vec{y}$ are observed, some variables $\vec{h}$ are hidden/latent, and the input data is $\vec{d}$, we can formalize the learning latent as minimizing the loss $\Delta(\vec{y}_i, \vec{y}_i^*(\lambda), \vec{h}_i^*(\lambda))$ from estimator $\vec{y}_i^*(\lambda), \vec{h}_i^*(\lambda)$ of the risk

$$R(\lambda) = \frac{1}{2}||\lambda||^2 + c\sum_{i=1}^{M} \Delta(\vec{y}_i, \vec{y}_i^*(\lambda), \vec{h}_i^*(\lambda)).$$

Similar to the above solution, this optimization is reduced to minimize the convex upper bound $\hat{R}(\lambda)$

$$\hat{R}(\lambda) = \frac{1}{2}||\lambda||^2 + c\sum_{i=1}^{M} \max_{(\hat{\vec{y}}, \hat{\vec{h}})} (\Delta(\vec{y}_i, \vec{y}^*, \vec{h}^*) + \lambda\phi(\vec{d}_i, \vec{y}^*, \vec{h}^*) - \max_{\vec{h}} \lambda \cdot \phi(\vec{d}_i, \vec{y}_i, \vec{h}). \tag{3}$$

- We can decompose the energy function in Equation 3 to obtain

$$f(\lambda) = \max_{(\vec{y}^*, \vec{h}^*)} (\Delta(\vec{y}_i, \vec{y}^*, \vec{h}^*) + \lambda \cdot \phi(\vec{d}_i, \vec{y}^*, \vec{h}^*))$$

as a convex function. And,

$$g(\lambda) = -\max_{\vec{h}} \lambda \cdot \phi(\vec{d}_i, \vec{y}_i, \vec{h})$$

as a concave function.

- The CCCP can be used to perform the learning by the iterative process following:

1. At step t: Estimating the hidden state $\vec{h}_i^*$

$$\frac{\partial g(\lambda^t)}{\partial \lambda} = -\phi(\vec{d}_i, \vec{y}_i, \vec{h}^*),$$

where $\vec{h}^* = \arg\max_{\vec{h}} \lambda^t \phi(\vec{d}_i, \vec{y}_i, \vec{h})$, λ^t is the current estimate of λ. This reduces to a modified SVM with the known state:

$$\min_{\lambda} \frac{1}{2}||\lambda||^2 + c\sum_{i=1}^{M} \max_{(\vec{y}, \vec{h})} \{\lambda \cdot \phi(\vec{d}_i, \vec{y}_i, \vec{h}) + \Delta(\vec{y}_i, \vec{y}, \vec{h})\} - c\sum_{i=1}^{M} \lambda \cdot \phi(\vec{d}_i, \vec{y}_i, \vec{h}_i^*).$$

2. Estimate λ when $\vec{h}$ is known.

3. Repeat steps 1 and 2 until convergence.