CS 542: Machine Learning **Spring 2016**

Problem Set 5

Lecturer: Prof. Peter Chin *Due: May 4, 2016*

- ◇ Please hand in the written part in class on the due date. For the programming part, please email your code and report to `fcakir@bu.edu` by 23:59PM on the due date.
- ◇ Late policy: there will be a penalty of 10% per day, up to three days late. After that no credit will be given.

1. (40 points) Written Problems

(a) (10 points) Bishop 8.3

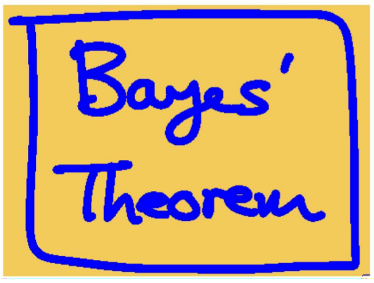
(b) (10 points) Bishop 8.4

(c) (10 points) Bishop 8.11

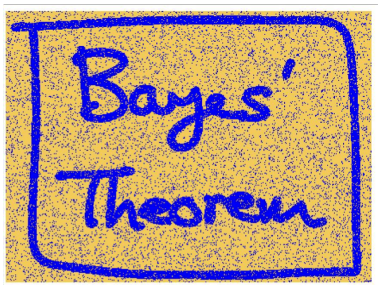
(d) (10 points) Bishop 8.14

2. (60 points) Programming

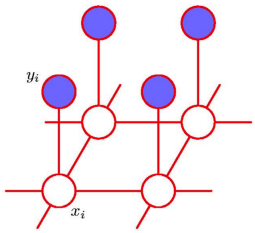
(a) (30 points)



(a) Clean image (to restore)



(b) Noise contaminated image



(c) MRF

The goal of this problem is to recover the clean image (a) from a noisy input (b). Of course, we cannot recover (a) exactly since information has been lost in the noise. The graphical model we use is a pairwise MRF as shown in (c). A pixel has four neighbors. We are going to use Markov Random Frields (MRFs) to model the distribution of natural images and restore the clean image given an noisy input image.

Observed image $y_i \in \{-1, +1\}$, and $i = 1, \cdots, D$ indexes pixels in the lattice. The original noise free image is $x_i \in \{-1, +1\}$. The noisy image y_i is obtained by randomly flipping the sign of pixels with some probability.

There are two types of cliques in this MRF. For $\{x_i, y_i\}$, we define the energy function as $-\eta x_i y_i$ ($\eta > 0$). Lower energy is achieved when x_i and y_i have the same sign and a higher

energy when they have the opposite sign. For a pair of variables $\{x_i, x_j\}$ where i and j are indices of neighboring pixels, we want the energy to be lower when the same sign than when they have opposite sign. So the energy function is $-\beta x_i x_j$ ($\beta > 0$). Lastly, we have an energy term hx_i for each pixel i to bias the model towards one particular sign (either + or −).

The final energy function for the model takes the form

$$E(\mathbf{x}, \mathbf{y}) = h \sum_i x_i - \beta \sum_{\{i,j\}} x_i x_j - \eta \sum_i x_i y_i$$

which defines a joint distribution over $\mathbf{x}$ and $\mathbf{y}$ given by

$$p(\mathbf{x}, \mathbf{y}) = \frac{1}{Z} \exp\{-E(\mathbf{x}, \mathbf{y})\}$$

Now implement Coordinate-descent algorithm as below on this:

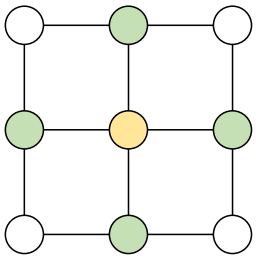
1. Initialize $\{x_i\}$ ($x_i = y_i$)
2. Loop over $\{x_i\}$. For each x_i, fix the neighborhood and see whether $-x_i$ would decrease the energy. If so, then flip x_i; otherwise, continue.
3. Stop when no changes can be made for $\mathbf{x}$.

Now make some initial guess for the parameters h, β, η so that the above algorithm converges and then adjust them till you can get up to 96% recovery or better. Record your parameter values and the number of iterations when you acheived this recovery. Save the recovered image and submit it along with your code. The clean image and noisy image are included: `Bayes.png`, `Bayes-noise.png`

(b) (30 points)

(a) Clean image (to restore)

(b) Noise contaminated image



(c) MRF

The goal of this problem is to once again recover the clean image (a) from a noisy input (b), now in the famous picture of Lena, which is used to benchmark many image processing algorithms. Of course, we cannot recover (a) exactly since information has

been lost in the noise. The graphical model we use is a pairwise MRF as shown in (c). A pixel has four neighbors.

The model we are going to use is a simple extension of the chain model we discussed on the class. A sample code for the message passing algorithm on a chain is included : `ChainMessagePassing.m`, `saveHHistogram.m`. Note that the sample code is a sum-product algorithm, which needs to be changed for max-sum algorithm. The clean image and noisy image are included: `Lena.png`, `Lena-noise.png`

Our graphical model is a simple pairwise MRF defined as

$$p(\mathbf{X}|\mathbf{Y}, \lambda_d, \lambda_d) = \frac{1}{Z} \exp\{-\lambda_d \sum_i \rho(x_i - y_i) - \lambda_s \sum_{(i,j)\in\varepsilon} \rho(x_i - x_j)\}$$

where $\mathbf{X} = \{x_i\}$ is the set of all the pixels in the image to restore, $\mathbf{Y} = \{y_i\}$ denotes the input noisy image, i indexes the image lattice, ε is the set of all the edges (only adjacent pixels are considered), λ_d and λ_s are coefficients weighing the data term and smoothness term, respectively. $\rho(\cdot)$ is a penalty function. You can use $\rho(z) = |z|$ (L1 norm) and $\rho(z) = z^2$ (L2 norm).

Use the max-product (or max-sum) algorithm to find the MAP solution for $\mathbf{X}$. Treat the number of state for each pixel as a parameter. Start from a small number such as 32, and then increase it to 64, 128, or 256. Try both L1 and L2 norms. You may also play with λ_d and λ_s to see which combination gives the best restoration result.