Assignment 5

Machine Learning COMS 4771

Spring 2017, Itsik Pe’er

Assigned: March 25th Due: Wednesday, April 5th, 1:10pm

Submission: Courseworks.

1) **PCA/clustering/EM:** Suppose each document is represented by a *D*-dimensional integer vector of counts of each of the *D* respective words in the dictionary. The *n*-th document is represented by $x_n = [x_n(1), \dots, x_n(D)]^T$, and its length is $l_n = \sum_{\Delta=1}^{D} x_n(\Delta)$
   a) You want to partition *N* input documents into *K* clusters in an unsupervised way. You are assuming each cluster *i*=1,…,*K* to be characterized by a multinomial distribution with (unknown) parameters $p^i(1), \dots, p^i(D)$. Devise an E-M clustering algorithm. Write update equations explicitly.
   [20 points]
   b) You want to reduce the dimensionality of the problem to $d \ll D$ dimensions, so each document x_n would be represented by a nonnegative real vector $u_n = [u_n(1), \dots, u_n(d)]^T$ of coefficients. You are assuming there exist *d* nonnegative real vectors $v_1, \dots, v_d \in \boldsymbol{R}^D$ in D dimensions, such that each x_n is approximated by $\hat{x}_n = \sum_{\delta=1}^{d} u_n(\delta) v_\delta$. More specifically, you assume x_n is a (vector) random variable whose expectation is $\hat{x}_n$. What is the marginal distribution of $x_n(\Delta)$ for each word $\Delta = 1, \dots, D$? What is a good approximation of this distribution if l_n is very large and $\frac{\hat{x}_n}{l_n}$ is very small?
   [10 points]
   c) How is the situation in (b) similar to PCA and how is it different from PCA?
   [20 points]
   d) Devise an EM algorithm for (b), to find the optimal $v_1, \dots, v_d$ with $\{u_n\}$ as the hidden variables.
   [20 points]

2) **Kernel SVM:** In this problem, you are going to explore the file *dataset1.csv* that contains ~50k labeled letter images (1st column: the letter; next 128 columns: 16 rows of 8 single-bit pixels). You may use a *python* library *Pillow* to generate the image for each datapoint and explore. Your task in this problem is to develop your own kernel function and use *sklearn*’s custom kernel SVM function to classify the letter for the data. Implement a *python* function *predictSVM* that receives one input ($M \times 128$), and returns one output ($M \times 1$). A detailed description is included in the *predict.py* template file provided. *predictSVM* predicts a string output given input. Similar to your midterm, your SVM model should be pre-trained, and you must use the template. In your write-up, explain your design of the kernel function, and include the custom kernel SVM’s classification error. Your grade of this problem will consist of two parts: the classification error in our test dataset ($M \times 128$) and your explanation.
   [25pts each]

Figure 1 The image of the first point in dataset1.csv

3) **Neural Networks:** You are building a neural network to classify input images, each with $D \times D$ real-valued pixels, i.e. the input is $X = \{X_n\}_{n=1}^{N} \subset \boldsymbol{R}^{D\times D}$, where each pixel is indexed with coordinates from zero to $D-1$, with labels $\{y_n\}_{n=1}^{N}$. Denote, for convenience, $D = d2^q + 2$. The first layer is a convolutional layer with a 3×3 filter matrix W is indexed with {-1,0,+1} horizontally and vertically: $W = \begin{bmatrix} w_{-1,-1} & w_{-1,0} & w_{-1,1} \\ w_{0,-1} & w_{0,0} & w_{0,1} \\ w_{1,-1} & w_{1,0} & w_{1,1} \end{bmatrix}$. The input to layer nodes is $a_{i,j}^{1} = \sum_{k,l\in\{0,\pm1\}} w_{k,l} x_{i+k,j+l}$ for $i,j = 1,\dots,D-2$. These nodes apply a logistic function. The next q layers are down-sampling layers, taking a soft maximum (log-of-sum-of exponents) of 4 previous-layer outputs without any coefficients:

$z_{i,j}^{r+1} = \log\left(\exp\left(z_{2i-1,2j-1}^{r}\right) + \exp\left(z_{2i-1,2j}^{r}\right) + \exp\left(z_{2i,2j-1}^{r}\right) + \exp\left(z_{2i,2j}^{r}\right)\right)$ for $r = 1,\dots,q$.

The last two layers are fully connected logistic layers, one between $d \times d$ input nodes and $d \times d$ output nodes, parametrized by a matrix U, the other between $d \times d$ input nodes and a single output node of the entire network, parametrized by a vector V. Write update equations.

[40 points]