CS 475 Machine Learning: Homework 4
EM and Clustering
Due: Thursday Nov 3, 2016, 11:59pm

Jonathan Liu
jliu118

1. Semi-supervised EM algorithm (10 points) We consider a d-dimensional mixture model with the following probability density function

$$f(x;\theta_1,...,\theta_k)=\sum_{i=1}^{K}\pi_i f_i(x;\theta_i),$$

where $f_i(x;\theta_i)$ is a probability density function. Suppose that we observe $n+m$ samples independently generated from the above mixture model, and m of the observed samples are labelled. Specifically, we have $x_1,...,x_n\in\mathbb{R}^d$ and $x_{n+1},...,x_{n+m}\in\mathbb{R}^d$. Meanwhile, we know the labels corresponding to $x_{n+1},...,x_{n+m}$, i.e., $y_{n+1},...,y_{n+m}\in\{1,...,K\}$. Design an EM algorithm to cluster the data.

[Hint: For $x_{n+1},...,x_{n+m}$, the corresponding labels are no longer missing values]

(a) Write the likelihood objective for this model.

(b) Write the update rules in each iteration.

Jon's Answer:

(a) $\log(p(x;\theta_1,...,\theta_k))=\sum_{i=1}^{m}\log\left(p(y_i;\theta)p(x_i;y_i,\theta)\right)+\sum_{i=m+1}^{m+n}\log\left(\sum_{y=0}^{K}p(y;\theta)p(x_i;y,\theta)\right)$

(b) Initialize the parameters using the m labeled examples.

Then perform the expectation step where we find the label y s.t. $p(y|x,\theta)$ is maximized for each of the n unlabeled examples.

Once these clusters are obtained, and all the examples are labeled, use these newly labeled examples to update the parameters.

2) Deep Neural Networks (15 points)

(a) Consider a 2-layer neural network, with M input nodes, Z nodes in the hidden layer and K nodes in the output layer. The network is fully connected, i.e. every node in the $n-1$th layer is connected to every node in the nth layer. However, for your application of interest, you suspect that only some of the nodes in the input are relevant. How would you modify the objective function to reflect this belief?

(b) Consider a N layer neural network. We could (a) train the entire network at once using back-propagation or (b) pre-train each layer individually, and then tune the final network with back-propagation. Will (a) and (b) converge to the same solution? Why would we favor strategy (a) vs. strategy (b)?

(c) Consider a $N \geq 2$ layer neural network with a single node in the output layer. We wish to train this network for binary classification. Rather than use a cross entropy objective, we want to take a max-margin approach and ensure a margin of $\gamma = 1$. Describe the structure of the last layer of the network, including the final activation function, and the training objective function that implements a max-margin neural network. What are the benefits of this network compared to one trained with cross entropy? Will a max-margin trained neural network learn the same decision boundary as an SVM?

Jon's Answer:

(a) Since some of the nodes are not relevant, when calculating the objective function we can set the weights corresponding to the nodes that aren't relevant to 0 or something much lower than the other nodes/features so the nodes that are less relevant will have less impact on the objective function.

(b) No they will not converge to the same solution because there is no closed form solution so they could converge to different solutions. Method (b) is favorable because if we use backpropagation we can get stuck in local optima because the objective is non-convex, also it takes longer and requires a ton of labeled data.

3) Neural Networks (15 points) Suppose we have two inputs: x_1 and x_2. Both x_1 and x_2 are real numbers and their values are restricted such that $-1 \leq x_1 \leq 1$ and $-1 \leq x_2 \leq 1$. You may use activation functions of the form:

$$\theta_r(z) = \begin{cases} 1 & \text{if } z \leq r \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where r determines the activation function. You may select a different r for each node in an artificial neural network. Create artificial neural networks for the following functions.

(a) Create a multi-layer network to recognize when $x_2 \geq \max(x_1, 1 - x_1)$.

(b) Create a multi-layer network to recognize when $|x_2| + |x_1| \leq 1$.

(c) Suppose we wanted to build a multi-layer network that approximates (with some error) the decision $x_1^2 + x_2^2 \leq 1$. Explain such an approximation.

For each network, describe the network structure, the value of each weight, and the activation function for each node. You may do this in words, or by drawing a picture. Please keep your answers clear and concise, i.e. we do not need a long explanation of how the network works; we only need the network definition.

Jon's Answer:

(a)

$$\begin{tikzcd}
 & {} \arrow[d] & \\
 & r=-2 & \\
r=-1 \arrow[ur, "-1"] & & r=0 \arrow[ul, "-1"'] \\
x_1 \arrow[u, "-1"] \arrow[urr, "1"' near start] & & x_2 \arrow[u, "-1"'] \arrow[ull, "-1" near start]
\end{tikzcd}$$