Korea Advanced Institute of Science and Technology
School of Electrical Engineering
EE488 Intro. to Machine Learning, Spring 2018

Issued: Apr. 30, 2018
Due: May. 23, 2018

Assignment III - part I

Policy

Group study is encouraged; however, assignment that you hand-in must be of your own work. Anyone suspected of copying others will be penalized. The homework will take considerable amount of time so start early.

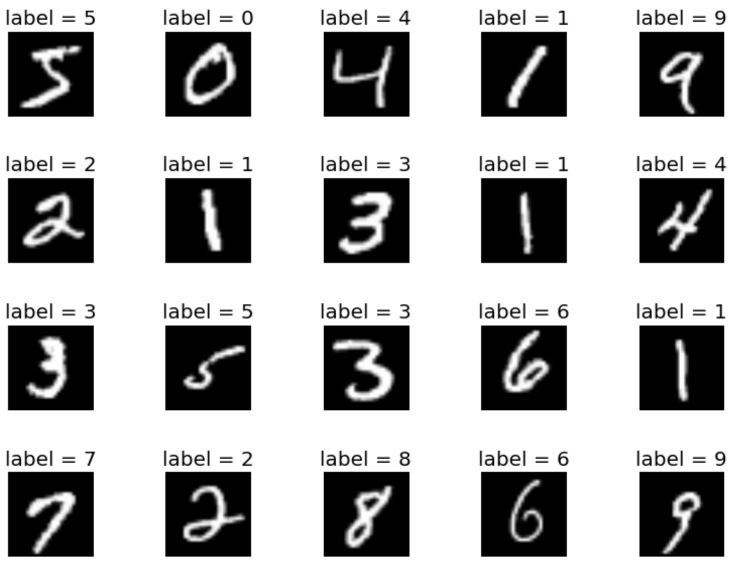
1. **K-NN** For a given test sample $\mathbf{x}$ with nearest neighbor $\mathbf{x}_n$, error occurs when the label associated with $\mathbf{x}_n$ denoted as $\theta_n \in \Omega$ does not equal the true label of $\mathbf{x}$ denoted as $\theta \in \Omega$. Thus the error rate given both the $\mathbf{x}$ and its nearest neighbor $\mathbf{x}_n$ is $P(\theta \neq \theta_n \big| \mathbf{x}, \mathbf{x}_n)$. Assume finite label set of C classes such that $\Omega = \{\omega_1, \omega_2, \ldots, \omega_C\}$.
   - (i) Given $\mathbf{x}$ and nearest neighbor $\mathbf{x}_n$, express error rate in terms of accuracy (when $\theta = \theta_n = \omega_l$ where $l \in \{1, \ldots, C\}$).
   - (ii) When $P(\omega_m \big| \mathbf{x}) = S$, then $\sum_{i=1}^{C} P^2(\omega_i \big| \mathbf{x})$ is minimum when $P(\omega_i \big| \mathbf{x}) = \frac{1-S}{C-1}$ for $i = 1, \ldots, m-1, m+1, \ldots, C$. Explain why this might be so.
   - (iii) Assume a dense training set in other words training data of infinite size, then the nearest neighbor of $\mathbf{x}$ will be itself such that $\mathbf{x}_n = \mathbf{x}$. Assume Bayes error rate defined as $P^*(e \big| \mathbf{x}) = 1 - \max_{\theta \in \Omega} P(\theta | \mathbf{x}) = 1 - P(\omega_m | \mathbf{x})$. Lower bound $\sum_{i=1}^{C} P^2(\omega_i | \mathbf{x})$ in terms of $P^*(e \big| \mathbf{x})$.
   - (iv) Upper bound error rate in terms of Bayes error rate.

2. **Bayesian Statistics**
   - (i) Given observation data $D = \{0, 0, 1, 0, 1, 1, 1\}$, use beta prior distribution with parameter (α_0, β_0) to determine $P(x = 1 | D)$ in terms of (α_0, β_0).
   - (ii) Given observation samples $D = \{x_1, x_2, \ldots, x_n\}$ from a Gaussian distribution with mean μ and known variance σ^2. Using prior distribution for μ to be a Gaussian with mean μ_0 and variance $\frac{1}{r_0}$, determine $P(\mu \big| D)$.
   - (iii) Given observation samples $D = \{x_1, x_2, \ldots, x_n\}$ from a Gaussian distribution with mean μ and precision γ. Using prior distribution for (μ, γ) a Normal-gamma distribution, determine $P(\mu, \gamma \big| D)$.
   - (iv) Express poisson, multinomial distribution, Laplace distribution, Dirichlet distribution, and gamma distribution in exponential form. Determine the conjugate prior for each of the distribution.

3. **Decision Theory**: Consider a binary classification problem where binary data x is generated from two different Bernoulli distribution where each is choosen depending on the binary value of y which follows a Bernoulli distribution, $y \sim Bernoulli(\frac{1}{2})$. If $y = 1$, then $x \sim Bernoulli(p)$ and otherwise, $x \sim Bernoulli(q)$. Assume that $p > q$. What is the Bayes optimal classifier and what is its expect loss or risk based on 0-1 loss?

4. **Implementing Backpropagation algorithm(Matlab Programming Assignment)**

   Write a matlab code for classifying the handwritten digits included in the MNIST database. Examples of hand written digits are shown below. The MNIST dataset includes 60,000 training and 10,000 testing. The size of each image is 28×28, and the each image can be classified into 10 different classes (0,1,...,9)



   For implementation, a skeleton code is provided in the "nn" folder. The main function `main_nn.m` calls the following 5 functions:(1) `[x_train,y_train,x_test,y_test] = createDataset()`, (2) `net = initialize_network(num_neuron, init)`, (3) `[net] = feed_forward(data_input, net)`,(4) `net_update = back_propagation(net, data_output)`, (5) `net = weight_update(net, net_update, training)`.

   (i) `[x_train,y_train,x_test,y_test] = createDataset()` loads both the training and test dataset respectively.

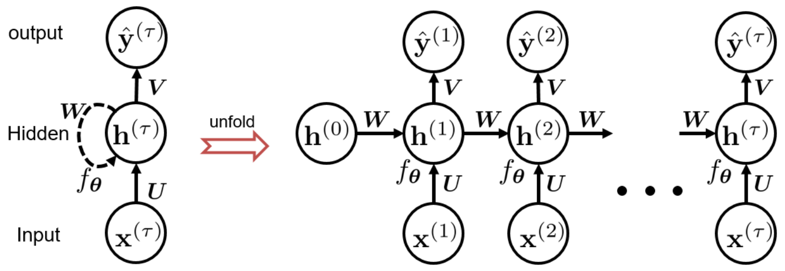
   (ii) `net = initialize_network(num_neuron, init)` takes vector `num_neuron` which specifies how many hidden nodes to use per each layer and `init` which has mean and stdev of weight parameters while doing the initialization. The output of this function gives struct architecture `net` that saves weight parameters and biases.

   (iii) `[net, pred] = feed_forward(data_input, net)` takes mini-batch of training set {`x_train,y_train`} where `x_train` is the data array while `y_train` is the corresponding label array of training dataset and mini-batch of training set is represented as {`data_input, data_output`}. The outputs of this function are `net` which saves all the forward propagation informations and `pred` which estimates the label of corresponding inputs.

   (iv) `net_update = back_propagation(net, data_output)` takes `net` and `data_output`. The output of this function is `net_update` which holds the gradients of each parameters.

   (v) `net = weight_update(net, net_update, l_rate)` takes `net`, `net_update` and learning rate `l_rate`. The output of this functions is the updated weight parameters `net`.

5. **Recurrent Neural Networks**:



An RNN is shown in the above figure. Input sequence $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(\tau)}$ is fed to the hidden units $\mathbf{h}^{(\tau)} = f_{\boldsymbol{\theta}}(\mathbf{h}^{(\tau-1)}, \mathbf{x}^{(\tau)}) = \tanh(\boldsymbol{U}\mathbf{x}^{\tau} + \boldsymbol{W}\mathbf{h}^{(\tau-1)} + \boldsymbol{b})$ which hold information of past inputs. The output of the hidden units are fed to the output unit $\hat{\mathbf{y}}^{(\tau)} = softmax(\boldsymbol{V}\mathbf{h}^{(\tau)} + \boldsymbol{c})$. The RNN is trained to minimize the cross entropy given below for a set of given input/target sequence pair $(\mathbf{x}^{(\tau)}, \mathbf{y}^{(\tau)})$. The cross entropy is defined as $L(\mathbf{y}, \hat{\mathbf{y}}) = -\sum_{\tau}\sum_{\imath} y_i^{(\tau)} \log \hat{y}_i^{(\tau)}$.

(i) Express $\boldsymbol{h}^{(\tau)}$ in terms of $f_{\boldsymbol{\theta}}, \boldsymbol{h}^{(\tau-1)}$ and $\mathbf{x}^{(1)}, \cdots, \mathbf{x}^{(\tau)}$. (hint: unfolding $f_{\boldsymbol{\theta}}$ from $\mathbf{x}^{(\tau)}$ to $\mathbf{x}^{(1)}$)

(ii) Please provide expression for $\frac{\partial L}{\partial \boldsymbol{V}}$ and $\frac{\partial L}{\partial \boldsymbol{W}}$ in terms of values $\mathbf{o}^{(\tau)} = \boldsymbol{V}\mathbf{h}^{(\tau)} + \boldsymbol{c}$, $\hat{\mathbf{y}}$ and $\mathbf{h}^{(\tau)}$.

(iii) Express $f_{\boldsymbol{\theta}}^{(\tau)}(\cdot,\cdot)$ in terms of $\mathbf{x}^{(t)}$ for $t = 1, \ldots, \tau$.

Submit Instructions for Programming Assignment

Please submit in .zip file to KLMS named "ee488_assignment3_student#.zip" , for example, "ee488_assignment3_20181234.zip".
This file should contain two folders and one document file for plotted result and explanation of the result.
In matlab code, the comment explaining your code **must be** included, or you will not get a full grade even if your code works fine. Please also include all the files that are required to run the code in the zip file. Do not change the name of the folder and comments should be written in English. Additionally submitting unexecutable code will receive no points. Using other libraries such as 'scipy' are not allowed.