CSE 253 Assignment 2

Puneeth BommiReddy
A53093725
pbommire@eng.ucsd.edu

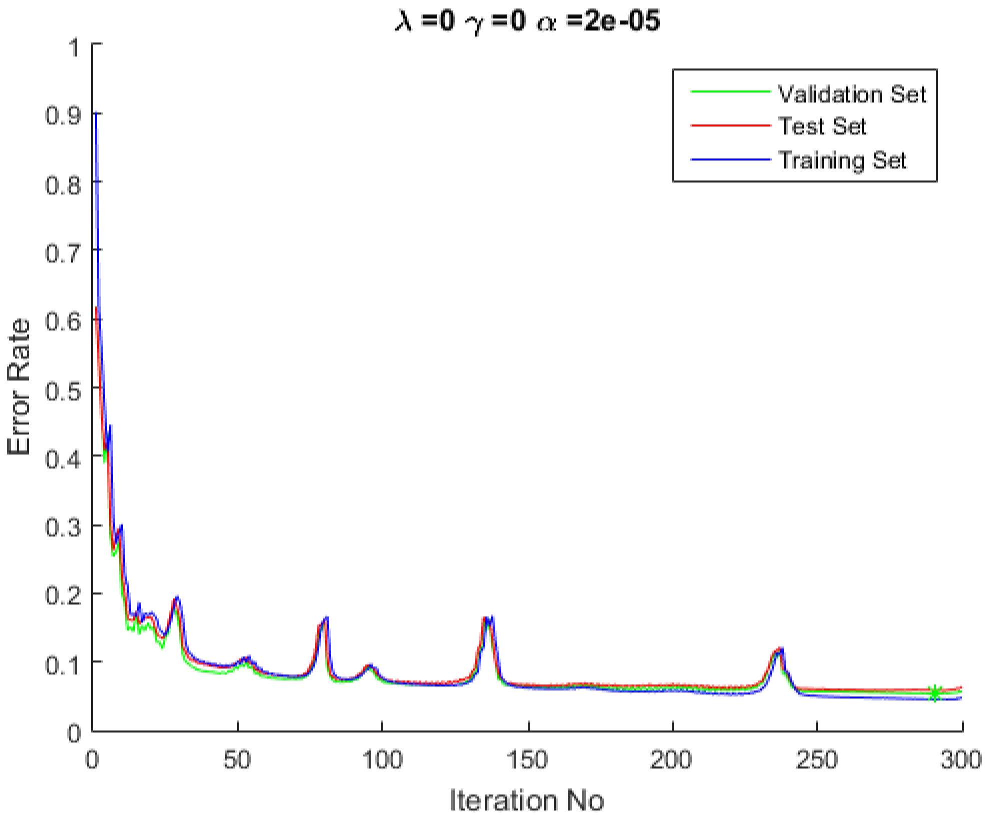
Note: Italics refer to variables in the code. The green star marks the location of the minimum error in the validation set. Training was done over the entire batch in all cases (not stochastic). Notes are attached at the end.

Question 2

For parts (d) to (f), the following description holds:

- The relevant script file is cse253_assgn2.m
- No of training images (*n_train*) = 50,000. No of validation images *(n_vldtn)*= 10,000.
- Number of Hidden Units (*NHU*) = 20 + 1(bias unit)
- α = 1/*n_train* = 2x10^-5
- Activation function is tanh
- No of iterations (*n_iter*) = 300
- The weights are initialized according to the formula U[- $\sqrt{\frac{6}{fan_{in} + fan_{out}}}$,+ $\sqrt{\frac{6}{fan_{in} + fan_{out}}}$] (Uniform distribution in that interval). This works well for tanh purportedly.
- If h(x) = tanh(x), $\frac{dh(x)}{dx} = 1 - h(x)^2$. This can be seen in act.m
- The reported minimum error rate for the validation set is close to the actual minimum as can be seen from the oscillations of error around the min value. Decreasing the learning rate adaptively will yield a better solution. But from visual inspection it seemed like the improvement was marginal (~1.5% for 1000 iterations).

d. iii.



As can be seen, in the absence of momentum the error oscillates quite a bit. Minimum validation error(*mvl*) = 5.36% at the 291th iteration (*loc*). The test error at the 291th iteration was 5.87%. The maximum absolute value of WIH (*max(abs(WIH(:)))*) = 0.5298, that of WHO (*max(abs(WHO(:)))*) = 1.5099. It is important to notice that the weights are very small. Thus in the subsequent L2 regularization step, the weights will not be affected in the least, as they are already small. However, L1 regularization would have helped remove some unnecessary small weights.

ii. After running the 300 iterations using tanh activation, cse253_error_aux.m is run to verify numerically the value of $\frac{dE(w)}{dw}$ using central differences. 1000 training images were used to compute this.

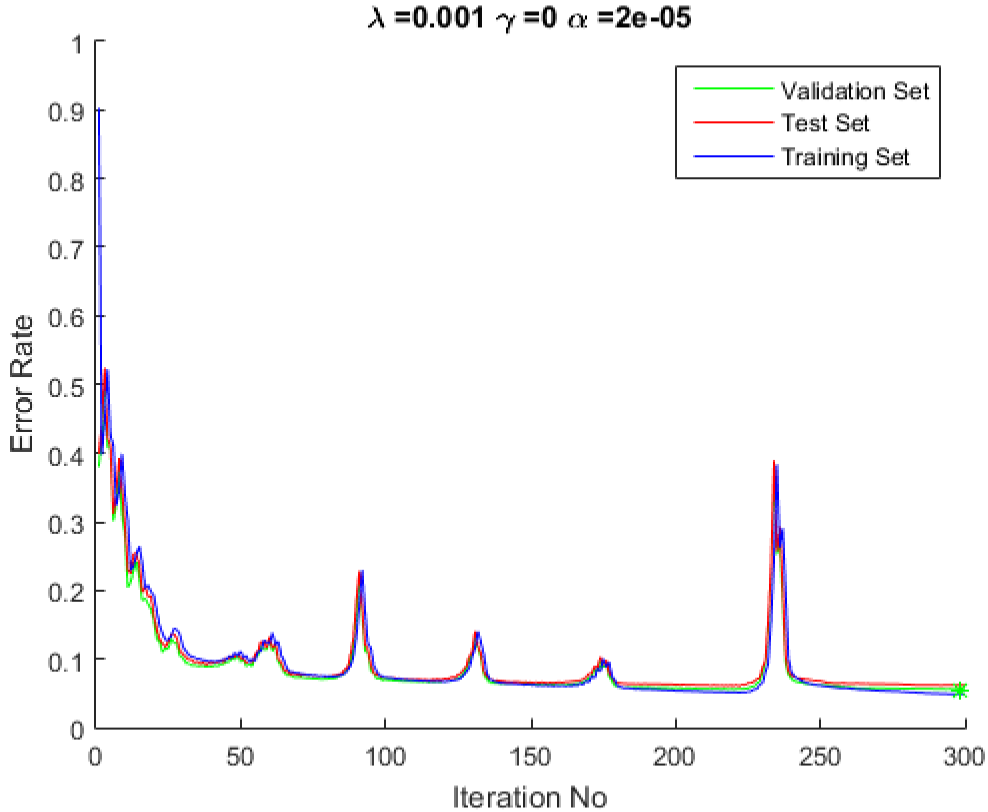
The perturbation (*eps*) = 10^{-5}. Thus the expected error is of the order of eps^2 = 10^{-10} (from the taylor series expansion about eps).

The mean absolute error (over all weights) between the numerical and theoretical errors for:

- WIH (*WIHErr*) = 4.6767e-10
- WHO (*WHOErr*) = 1.4412e-10

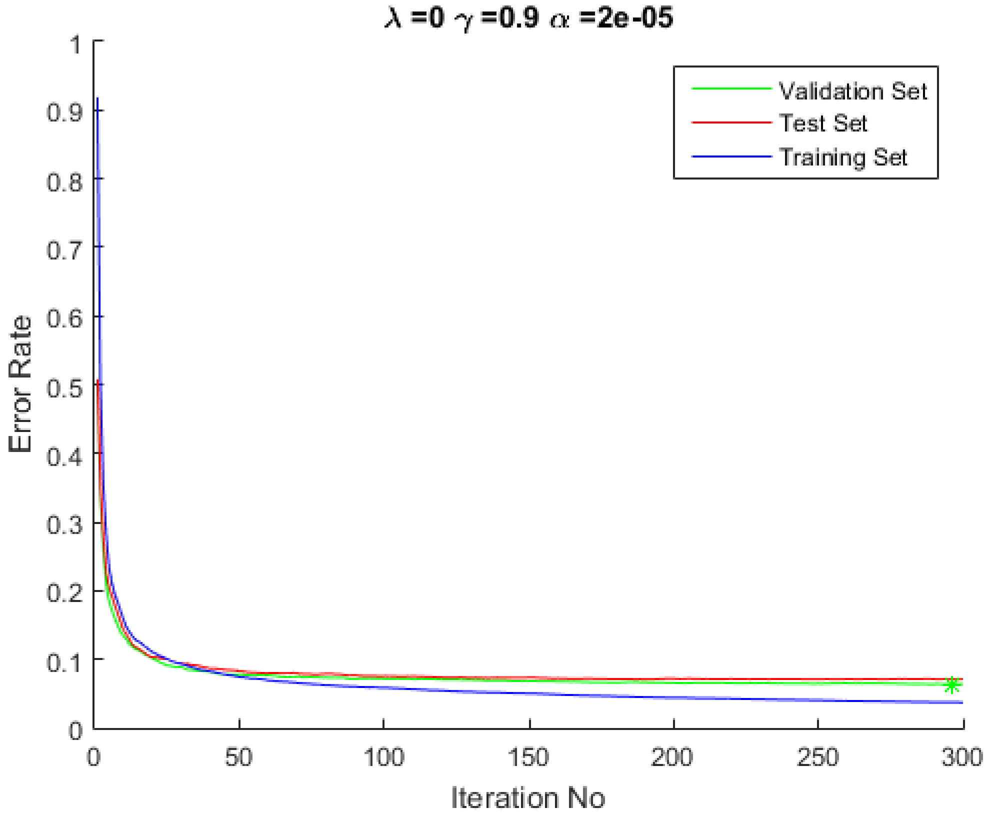
Which are ~ 10^{-10} as expected.

e. Regularization

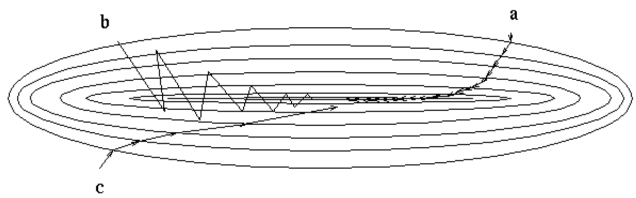


As anticipated in 2.d.iii, the error and weights are not affected significantly. Minimum validation error(*mvl*) = 5.53% at the 298th iteration (*loc*). The test error at the 298th iteration was 6.19%. The maximum absolute value of WIH (*max(abs(WIH(:)))*) = 0.4634, that of WHO (*max(abs(WHO(:)))*) = 1.5063. Indicating the weights were not affected much.

f. Momentum



As compared to 2.d.iii, the error curve is much smoother. This is because the momentum term prevents oscillation about a trench in the error surface (peaks in 2.d.iii). Also for a given number of iterations, momentum should lead to faster convergence.



The peaks in 2.d.iii correspond to (b) in illustration on the left. (a) corresponds to 2.f with momentum.

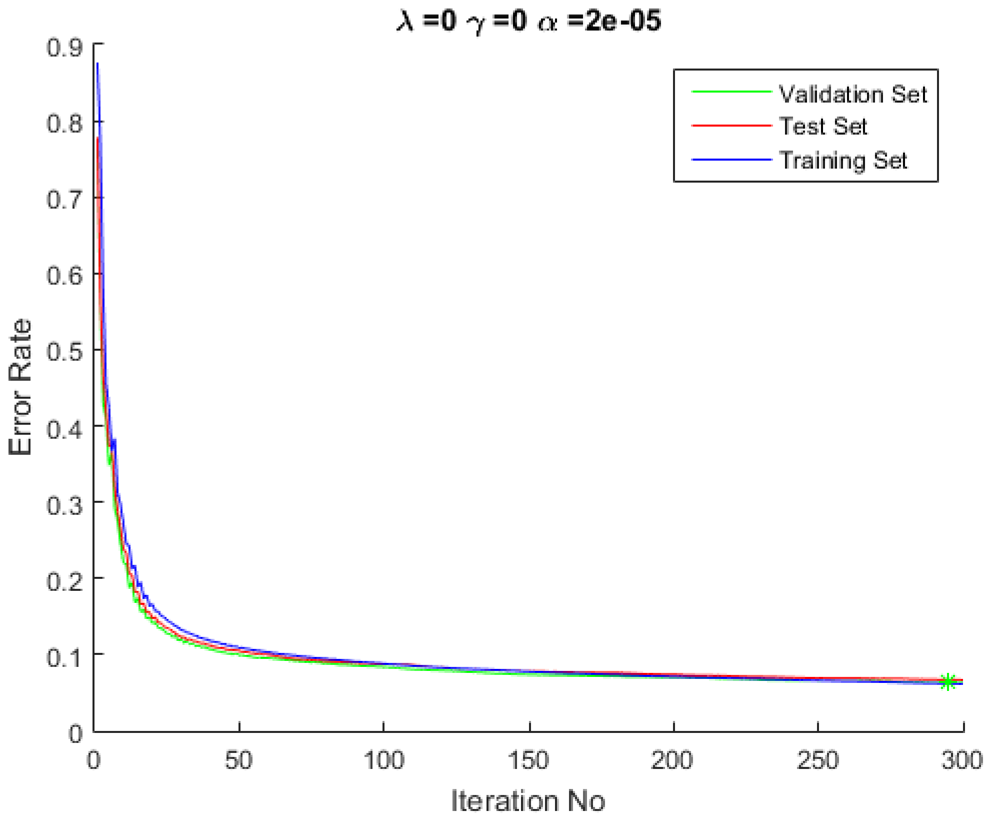
Minimum validation error (*mvl*) = 6.36% at the 298th iteration (*loc*). The test error at the 298th iteration was 7.11%. The increased error rate as compared to 2.d.iii is surprising. The error is higher probably because the net has not fully converged yet. However the marginal increase in performance has deterred me from running more iterations :P.

g. Activation functions

The relevant code is in act.m

A. Sigmoid (See attached notes for derivation)

$$h(z) = \frac{1}{1+\ e^{-z}};\ h'(z) = h(z)(1-h(z))$$

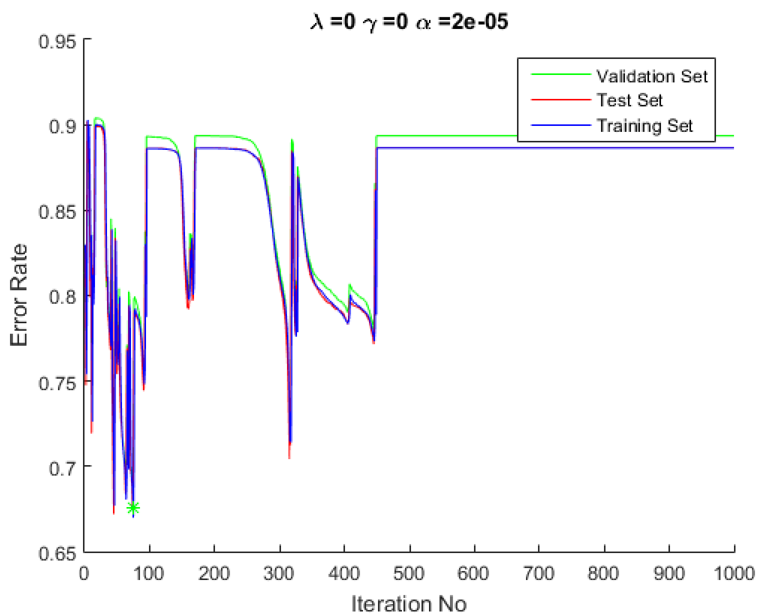


Minimum validation error(*mvl*) = 6.3% at the 295th iteration (*loc*). The test error at the 295th iteration was 6.73%.

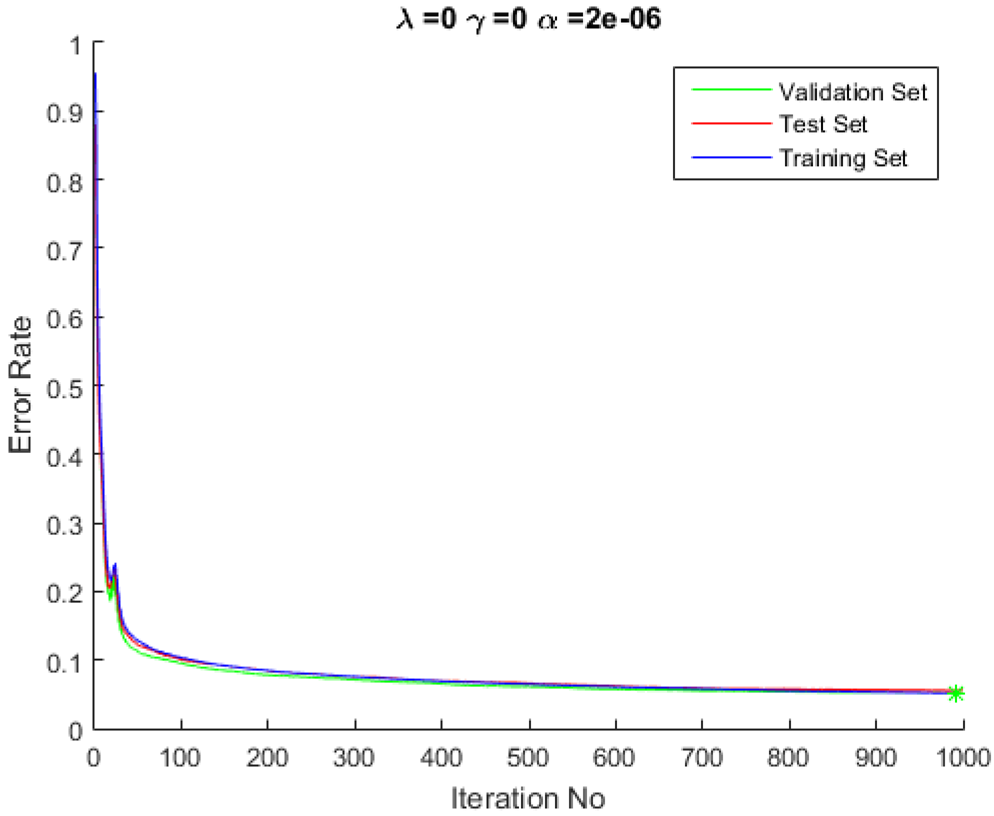
B. Tanh (already done)

C. ReLU

$$h(z) = \max(0, z)\,; h'(z) = \begin{cases} 1 & z > 0 \\ 0.5 & z = 0 \; (average\ of\ left\ and\ right\ limits) \\ 0 & z < 0 \end{cases}$$



Whoops is the learning rate too high? α_{batch} = 2e-0.5 corresponds to $\alpha_{stochatic} = \alpha_{batch} * n\_train$ = 1. Reducing it by a factor of 10 helps loads.

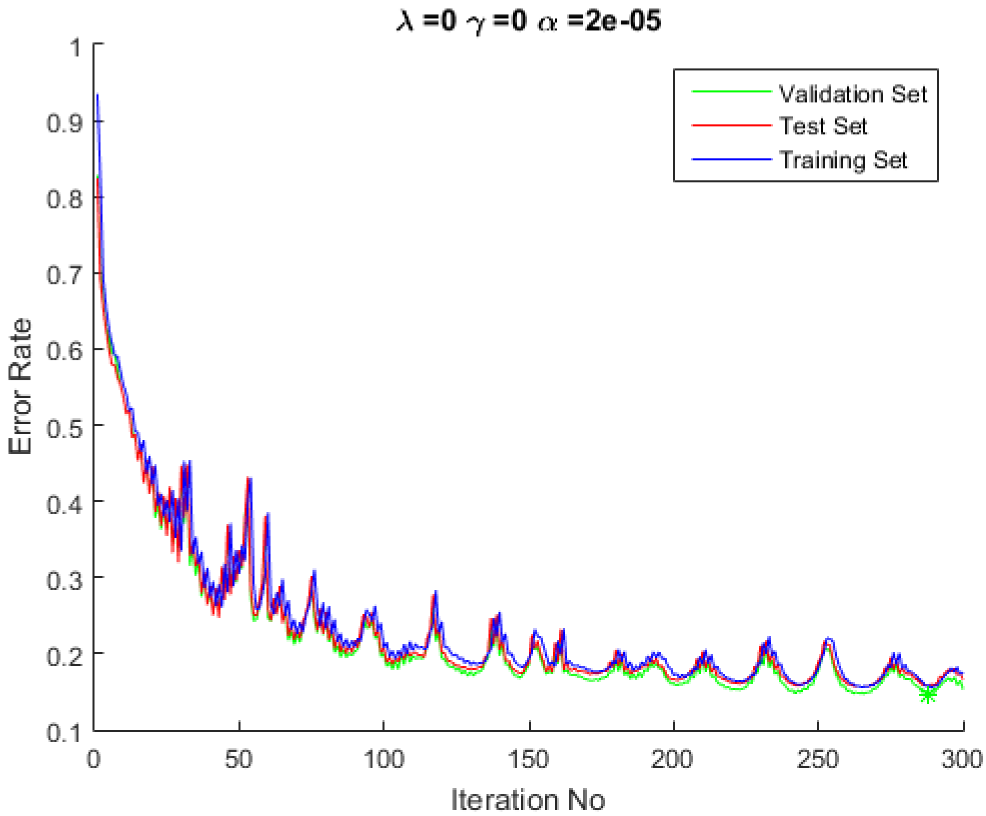


The error for RELU seems to decrease significantly (probably because α is small and not adaptive, and the linear activation for positive inputs must cause high gradients). This is why I ran it for 1000 iterations.

Minimum validation error(*mvl*) = 5.15% at the 991st iteration (*loc*). The test error at the 991st iteration was 5.55%.

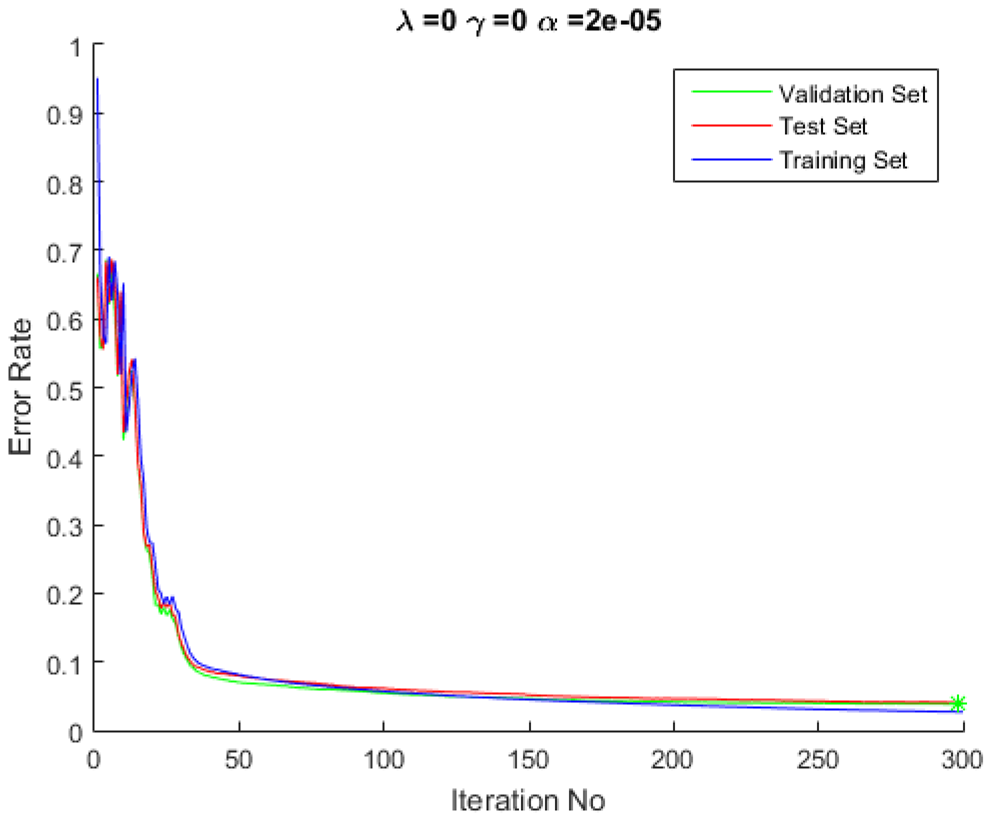
h. Network Topology

i. For effect, I reduced the number of hidden units by 4 instead of 2. Thus 5 + 1(bias) units are in the input layer. Again tanh is the activation function.



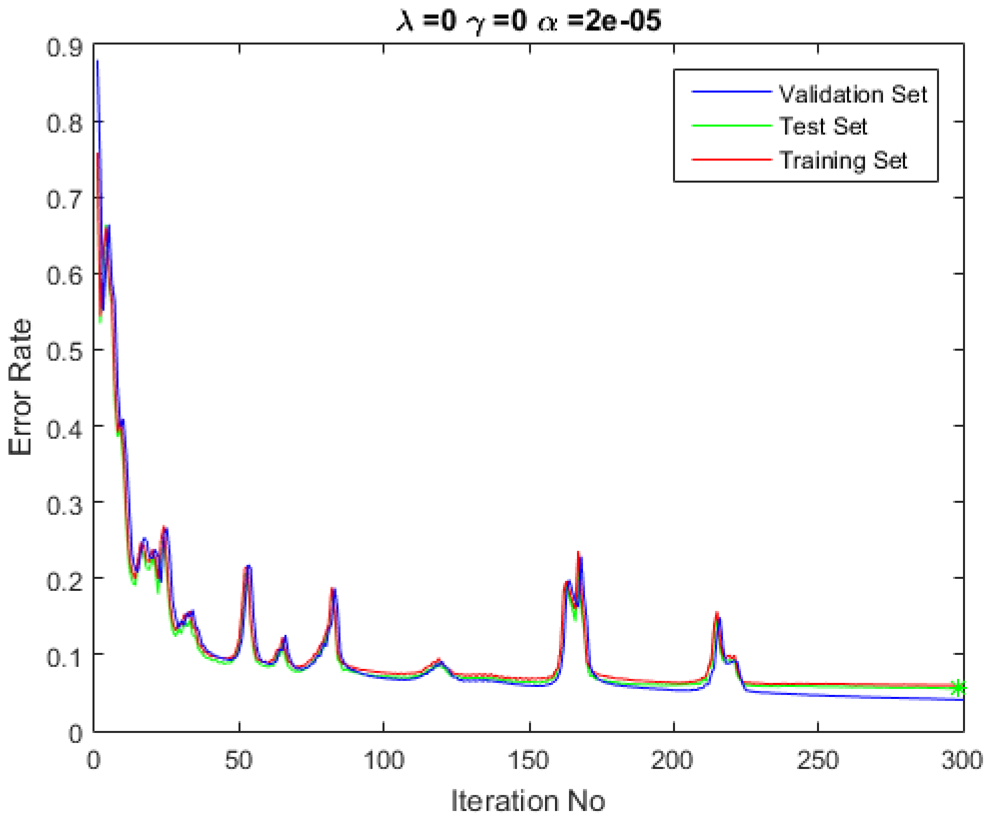
It is clear the error rate is much higher than 2.d.iii (upwards of 10%). This is because the reduced hidden units mean a reduced number of ‘features’ are learnt in the hidden layer leading to poor generalization.

Again for effect, I increased the number of hidden units by 4. Thus 80 + 1(bias) units are present.



Minimum validation error(*mvl*) = 3.86% at the 298th iteration (*loc*). The test error at the 298th iteration was 4.06%. Surprisingly the error rate decreased significantly! Must be that there are many more features to learn. However, I would not count of much better performance with further increase in hidden units.

ii. Two Hidden Layers



Minimum validation error(*mvl*) = 5.56% at the 298th iteration (*loc*). The test error at the 298th iteration was 5.98%. The performance is very similar to the single hidden layer case 2.d.iii. Probably suggesting that 2 hidden layers are overkill

CODE

Activation function (act.m)

```
function [Z,H_] = act(A,str)
    switch str
        case 'tanh'
            Z = tanh(A);
            H_ = 1 - Z.^2;
        case 'sigmoid'
            Z = 1./(1 + exp(-A));
            H_ = Z.*(1-Z);
        otherwise
            Z = (A>0).*A;
            H_ = A > 0 + 0.5*(A==0);
    end
end
```

Single hidden layer MLP (cse253_assgn2.m)

```
%% Loading Images
trainl_images = zscore(loadMNISTImages('train-images.idx3-ubyte'));
```

```matlab
trainl_labels = loadMNISTLabels('train-labels.idx1-ubyte');

n_train = 50000; n_vldtn = 10000;n_test = 10000;
X = [ones(1,n_train);trainl_images(:,1:n_train)]; %X = train_images
train_labels = trainl_labels(1:n_train);
T = zeros(10,n_train); %Target matrix.
for sample = 1:n_train
    T(train_labels(sample)+1,sample) = 1;% Each column has a '1' in the
location of the true class of that image
end

vldtn_images = [ones(1,n_vldtn);trainl_images(:,50001:60000)];
vldtn_labels = trainl_labels(50001:60000);
clear trainl_images trainl_labels

test_images = zscore(loadMNISTImages('t10k-images.idx3-ubyte'));
test_labels = loadMNISTLabels('t10k-labels.idx1-ubyte');
test_images = [ones(1,n_test);test_images];
%%
% Initialization
n_iter = 1000;
NI = 785; NHU = 5; NO = 10; alpha = 1/n_train; lambda = 0; gamma = 0; v1 =
0;v2 = 0;
WIH = 2*sqrt(6/(NI + NHU))*(rand(NI,NHU) - 0.5); WHO = 2*sqrt(6/(NHU+1 +
NO))*(rand(NHU+1,NO) - 0.5);
actfun = 'tanh';
train_error = zeros(n_iter,1);
vldtn_error = zeros(n_iter,1);
test_error = zeros(n_iter,1);
% Training
tic;
for iter = 1:n_iter
    A = (WIH')*X;
    [Z,H_] = act(A,actfun);
    Z = [ones(1,n_train);Z]; Y = exp((WHO')*Z);
    for sample = 1:n_train %Normalizing activations
        Y(:,sample) = Y(:,sample)/sum(Y(:,sample));
    end
    Del_k = T - Y;
    v1 = gamma*v1 - alpha*X*((H_.*(WHO(2:NHU+1,:)*Del_k))');
    v2 = gamma*v2 - alpha*Z*(Del_k');
    WIH = (1-alpha*lambda)*WIH - v1;
    WHO = (1-alpha*lambda)*WHO - v2;
    %Calculating training error
    [~,I] = max(Y); train_output = (I-1)';
    train_error(iter) = sum(1-(train_output == train_labels))/n_train;
    %Calculating validation error
    [~,I] = max((WHO')*[ones(1,n_vldtn);act((WIH')*vldtn_images,actfun)]);
    vldtn_output =(I-1)'; vldtn_error(iter) = sum(1-(vldtn_output ==
vldtn_labels))/n_vldtn;
    %Calculating test error
    [~,I] = max((WHO')*[ones(1,n_test);act((WIH')*test_images,actfun)]);
    test_output = (I-1)'; test_error(iter) = sum(1-(test_output ==
test_labels))/n_test;
end
toc;
hold on;
plot(vldtn_error,'g');xlabel('Iteration No');ylabel('Error Rate');
plot(test_error,'r');
plot(train_error,'b');legend('Validation Set','Test Set','Training Set');
[mvl,loc] = min(vldtn_error);
```

```
plot(loc,mvl,'g*');
title(['\lambda =',num2str(lambda),' \gamma =',num2str(gamma),' \alpha
=',num2str(alpha)]);
hold off;
[mvl,loc,vldtn_error(300),test_error(300),test_error(loc),max(abs(WIH(:))),
max(abs(WHO(:)))]
```

Error Verification (cse253_error_aux.m)

```
eps = 10^-5; n = 1000; X = X(:,1:n); T = T(:,1:n);
EIH = zeros(NI,NHU);EHO = zeros(NHU+1,NO);
A = (WIH')*X;[Z,H_] = act(A,'tanh');Z = [ones(1,n);Z]; Y = exp((WHO')*Z);
for sample = 1:n
    Y(:,sample) = Y(:,sample)/sum(Y(:,sample));
end
Del_k = T - Y;
EIHr = -X*((H_.*(WHO(2:NHU+1,:)*Del_k))');
EHOr = -Z*(Del_k');
for i = 1:NI
    for j = 1:NHU
        WIH(i,j) = WIH(i,j) + eps;
        Yf = exp((WHO')*[ones(1,n);act((WIH')*X,'tanh')]);
        WIH(i,j) = WIH(i,j) - 2*eps;
        Yb = exp((WHO')*[ones(1,n);act((WIH')*X,'tanh')]);
        for sample = 1:n
            Yf(:,sample) = Yf(:,sample)/sum(Yf(:,sample));
            Yb(:,sample) = Yb(:,sample)/sum(Yb(:,sample));
        end
        EIH(i,j) = -sum(sum(T.*(log(Yf) - log(Yb))))/(2*eps);
        WIH(i,j) = WIH(i,j) + eps;
    end
end
for i = 1:NHU + 1
    for j = 1:NO
        WHO(i,j) = WHO(i,j) + eps;
        Yf = exp((WHO')*[ones(1,n);act((WIH')*X,'tanh')]);
        WHO(i,j) = WHO(i,j) - 2*eps;
        Yb = exp((WHO')*[ones(1,n);act((WIH')*X,'tanh')]);
        for sample = 1:n
            Yf(:,sample) = Yf(:,sample)/sum(Yf(:,sample));
            Yb(:,sample) = Yb(:,sample)/sum(Yb(:,sample));
        end
        EHO(i,j) = -sum(sum(T.*(log(Yf) - log(Yb))))/(2*eps);
        WHO(i,j) = WHO(i,j) + eps;
    end
end
WIHErr = sum(sum(abs(EIH - EIHr)))/(NI*NHU);
WHOErr = sum(sum(abs(EHO - EHOr)))/(NO*(NHU+1));
```

Two hidden layers (cse253_assgn2_2.m)

```
%% Loading Images
trainl_images = zscore(loadMNISTImages('train-images.idx3-ubyte'));
trainl_labels = loadMNISTLabels('train-labels.idx1-ubyte');

n_train = 50000; n_vldtn = 10000;n_test = 10000;
```

```
X = [ones(1,n_train);trainl_images(:,1:n_train)]; %X = train_images
train_labels = trainl_labels(1:n_train);
T = zeros(10,n_train); %Target matrix.
for sample = 1:n_train
    T(train_labels(sample)+1,sample) = 1;% Each column has a '1' in the
location of the true class of that image
end

vldtn_images = [ones(1,n_vldtn);trainl_images(:,50001:60000)];
vldtn_labels = trainl_labels(50001:60000);
clear trainl_images trainl_labels

test_images = zscore(loadMNISTImages('t10k-images.idx3-ubyte'));
test_labels = loadMNISTLabels('t10k-labels.idx1-ubyte');
test_images = [ones(1,n_test);test_images];
%% Initialization
n_iter = 1000;
NI = 785; NH1U = 20; NH2U = 20; NO = 10; alpha = 0.001/NI; lambda = 0;
gamma = 0.5; v1 = 0;v2 = 0; v3 = 0;
WIH1 = sqrt(6/(NI + NH1U))*rand(NI,NH1U);
WH1H2 = sqrt(6/(NH1U+NH2U+1))*rand(NH1U+1,NH2U);
WH2O = sqrt(6/(NH2U+1 + NO))*rand(NH2U+1,NO);

train_error = zeros(n_iter,1);
vldtn_error = zeros(n_iter,1);
test_error = zeros(n_iter,1);
% Training
tic;
for iter = 1:n_iter
    A1 = (WIH1')*X; [Z1,H1_] = act(A1,'tanh'); Z1 = [ones(1,n_train);Z1];
    A2 = (WH1H2')*Z1; [Z2,H2_] = act(A2,'tanh'); Z2 = [ones(1,n_train);Z2];
    Y = exp((WH2O')*Z2);
    for sample = 1:n_train %Normalizing activations
        Y(:,sample) = Y(:,sample)/sum(Y(:,sample));
    end
    Del_O = T - Y;
    v3 = gamma*v3 - alpha*Z2*(Del_O');
    Del_H2 = H2_.*(WH2O(2:NH2U+1,:)*Del_O);
    v2 = gamma*v2 - alpha*Z1*(Del_H2');
    Del_H1 = H1_.*(WH1H2(2:NH1U+1,:)*Del_H2);
    v1 = gamma*v1 - alpha*X*(Del_H1');

    WIH1 = (1-alpha*lambda)*WIH1 - v1;
    WH1H2 = (1-alpha*lambda)*WH1H2 - v2;
    WH2O = (1-alpha*lambda)*WH2O - v3;
    %Calculating training error
    [~,I] = max(Y); train_output = (I-1)';
    train_error(iter) = sum(1-(train_output == train_labels))/n_train;
    %Calculating validation error
    A1 = (WIH1')*vldtn_images; Z1 = act(A1,'tanh'); Z1 =
[ones(1,n_vldtn);Z1];
    A2 = (WH1H2')*Z1; Z2 = act(A2,'tanh'); Z2 = [ones(1,n_vldtn);Z2];
    [~,I] = max((WH2O')*Z2); vldtn_output =(I-1)';
    vldtn_error(iter) = sum(1-(vldtn_output == vldtn_labels))/n_vldtn;
    %Calculating test error
    A1 = (WIH1')*test_images; Z1 = act(A1,'tanh'); Z1 =
[ones(1,n_test);Z1];
    A2 = (WH1H2')*Z1; Z2 = act(A2,'tanh'); Z2 = [ones(1,n_test);Z2];
    [~,I] = max((WH2O')*Z2); test_output = (I-1)';
    test_error(iter) = sum(1-(test_output == test_labels))/n_test;
```

```
end
toc;
hold on;
plot(vldtn_error,'g');xlabel('Iteration No');ylabel('Error Rate');
plot(test_error,'r');
plot(train_error,'b');legend('Validation Set','Test Set','Training Set');
[mvl,loc] = min(vldtn_error);
plot(loc,mvl,'g*');
title(['\lambda =',num2str(lambda),' \gamma =',num2str(gamma),' \alpha
=',num2str(alpha)]);
hold off;
```

Date

CSE 253 ASSIGNMENT 2

Puneeth Bomoi Reddy

A53043725

1. Solution is the same as that in Bishop's PRML chapter 3.

$t = y(\tilde{x}, \theta) + \epsilon$

where $\tilde{x} = [1, x]$ and $y = \sum x_k \theta_k = \theta^T x$

$\therefore p(t | x, \theta, \sigma^2) = \mathcal{N}(t | y(x, \theta), \sigma^2)$ σ^2 is assumed variance.

The joint distribution is given by:

$$p(\vec{t} | X, \theta, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(t^n | \theta^T \tilde{x}^n, \sigma^2) \quad [\vec{t} = t_1 \dots t_n$$

$$\Rightarrow \ln p = \sum_{n=1}^{N} \ln \mathcal{N}(t^n | \theta^T \tilde{x}^n, \sigma^2)$$

$$= -\frac{N}{2} \ln \sigma^2 - \frac{N}{2} \ln 2\pi - \frac{1}{2\sigma^2} \sum_{n=1}^{N} (t^n - \theta^T \tilde{x}^n)^2$$

By the ML rule, $\theta^* = \arg\max_\theta \ln p$

$$= \arg\min \sum_{n=1}^{N} (t^n - \theta^T \tilde{x}^n)^2 \quad \left[\text{Other terms are independent of } \theta\right]$$

$$= \arg\min \sum_{i=1}^{N} (t^i - f((1, x^i), \theta))^2 \; //$$

$\uparrow$

changed summation index to i, substituted $\tilde{x}^i = [1, x^i]$ and $f(x, \theta) = \theta^T x$.

Date

CSE 253 ASSIGNMENT 2

Puneeth BommiReddy

A53093725

2(a) $E = -\sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\ln y_{nk}$ sample; kth output

Output Layer:

$$-\frac{\partial E}{\partial a_i} = \sum_{n=1}^{N}\sum_{k=1}^{K} t_{nk}\frac{\partial}{\partial a_i}(\ln y_{nk})$$

but $y_{nk} = \frac{e^{a_{nk}}}{\sum e^{a_{nk}}}$

$$= \sum\sum \frac{t_{nk}}{y_{nk}}\frac{\partial y_{nk}}{\partial a_i}$$

$\delta_{ij} = 1$ iff $i=j$, $=0$ if $i \neq j$

But $$\frac{\partial y_{nk}}{\partial a_i} = \frac{\partial}{\partial a_i}\left[\frac{e^{a_{nk}}}{\sum_k e^{a_{nk}}}\right] = \frac{e^{a_{nk}}}{\sum_k e^{a_{nk}}}\delta_{ki} - \frac{1\cdot e^{a_{nk}}e^{a_{ni}}}{\left(\sum e^{a_{nk}}\right)^2}$$

$$\therefore -\frac{\partial E}{\partial a_i} = \sum_{n=1}^{N}\sum_{k=1}^{K}\frac{t_{nk}}{\cancel{y_{nk}}}\cancel{y_{nk}}\left[\delta_{ki} - \frac{e^{a_{ni}}}{\sum_k e^{a_{nk}}}\right]$$

$$= \sum_n\sum_k t_{nk}\left[\delta_{ki} - y_{ni}\right]$$

$$= \sum_n\left[t_{ni} - y_{ni}\sum_k t_{nk}\right]$$ but $\sum_k t_{nk} = 1$

$$= \sum_n\left(t_{ni} - y_{ni}\right)$$

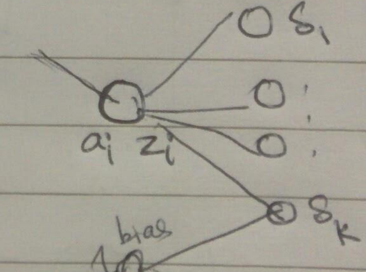
Per sample:

$$\delta_k = -\frac{\partial E_n}{\partial a_k} = \underline{\underline{(t_k - y_k)}}$$

Hidden Layer:

$$\delta_j = -\frac{\partial E_n}{\partial a_j} = -\sum_k \frac{\partial E_n}{\partial a_k}\frac{\partial a_k}{\partial a_j}$$

a_k are from the output layer



But $$\frac{\partial a_k}{\partial a_j} = \frac{\partial a_k}{\partial z_j}\frac{\partial z_j}{\partial a_j}$$

where $z_j = h(a_j)$ ↑ activation

$$= w_{jk}\,h'(a_j)$$

$$-\frac{\partial E_n}{\partial a_j} = +\sum_k \delta_k w_{jk} h'(a_j) = \underline{\underline{+h'(a_j)\sum \delta_k w_{jk}}}$$

Date

$\omega_{jk} = \omega_{jk} + \alpha \delta_k z_j$

(b) i. $\dfrac{\partial E}{\partial \omega_{jk}} = \dfrac{\partial E}{\partial a_k}\dfrac{\partial a_k}{\partial \omega_{jk}} = -\delta_k z_j$; $\omega_{jk} = \omega_{jk} + \alpha \delta_k z_j$

ii. $\dfrac{\partial E}{\partial \omega_{ij}} = \dfrac{\partial E}{\partial a_j}\dfrac{\partial a_j}{\partial \omega_{ij}} = -\delta_j x_i$; $\omega_{ij} = \omega_{ij} + \alpha \delta_j x_i$

(c) (i) $\omega_{jk} = \omega_{jk} + \alpha \delta_k z_j$

$W_{HO} = W_{HO} + \alpha Z (t-y)^T$ $\quad t = [t_1 .. t_O]^T$; $y = [y_1 .. y_O]^t$ $z = [z_1 .. z_H]^t$

but $a_j = \sum x_i w_{ij}$

$\vec{a} = (x^T W_{IH})^T = W_{IH}^T x \Rightarrow z = h(\vec{a}) = h(W_{IH}^T x)$

$\therefore W_{HO} = W_{HO} + \alpha [h(W_{IH}^T x)](t-y)^T$

(ii) $\omega_{ij} = \omega_{ij} + \alpha \delta_j x_i$

but $\delta_j = h'(a_j) \sum \delta_k \omega_{jk} = h'(a_j) \sum_k (t_k - y_k)\omega_{jk}$ $\Rightarrow \vec{\delta}_j = [W_{HO}(t-y)] .* h'(a_j)$

$\Rightarrow \vec{\delta}_j = h'(a_j) .* W_{HO}(t-y)$

$\therefore W_{IH} = W_{IH} + \alpha x [h'(W_{IH}^T x) .* W_{HO}(t-y)]^T$

Procedure: [For stochastic descent]

- Initialization:

$\vec{x} = [x_1 \dots x_I]^T$; $\vec{t} = [t_1 \dots t_O]^T$; $W_{IH} = (W_{IH})_{init}$ $W_{HO} = (W_{HO})_{init}$

- Update:

$\vec{a}_{H1} = W_{IH}^T x$; $\vec{h}'_{H1} = h'(a_{H1})$; $\vec{Z}_{H1} = h(a_{H1})$

$\vec{y} = \dfrac{e^{W_{HO}^T Z_{H1}}}{\sum_k (W_{HO}^T Z_{H1})_k}$

$(W_{IH})_{new} = W_{IH} + \alpha \vec{x} [\overbrace{\vec{h}' \cdot W_{HO}(\vec{t}-\vec{y})}^{b}]^T$

$(W_{HO})_{new} = W_{HO} + \alpha \vec{Z} (t - \vec{y})^T$

For batch descent:

Init: $X = [\vec{x}_1 \dots \vec{x}_n]$; $T = [\vec{t}_1, \dots \vec{t}_n]$

$W_{IH} \sim U\left[0, \frac{\sqrt{6}}{\text{fanin} + \text{fanout}}\right] \sim W_{HO}$

Update: $A_{Hn} = W_{IH}^T X_{IH}$ $\quad H'_{Hn} = h'(A_{Hn})$ $\quad Z_{Hn} = h(A_{Hn})$

$Y_{ON} = e^{W_{HO}^T Z_{Hn}}$ ← normalize.

$W_{IH} = W_{IH} + \alpha X \left[H' \cdot W_{HO} (T - Y) \right]^T$

$W_{HO} = W_{HO} + \alpha Z \left[T - Y \right]^T$

Note: Z will have additional bias terms appended as:

$Z = \left[1_{N1}^T ; h(A) \right]$

2g. i $f(z) = 1/1+e^{-z}$ $\quad \frac{d}{dz} f(z) = \frac{-1}{(1+e^{-z})^2} - e^{-z} = \frac{e^{-z}}{(1+e^{-z})^2} = \frac{1}{1+e^{-z}} \frac{e^{-z}}{1+e^{-z}}$

$= f(z)(1-f(z))$

ii $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$ $\quad \frac{d}{dx}\left(\frac{e^x - e^{-x}}{e^x + e^{-x}}\right) = \frac{e^x + e^{-x}}{e^x + e^{-x}} - \frac{(e^x - e^{-x})(e^x - e^{-x})}{(e^x + e^{-x})^2}$

$= 1 - f(x)^2$

iii $f(z) = \begin{cases} z & z > 0 \\ 0 & \text{elsewhere} \end{cases}$ $\quad \frac{d}{dz} f(z) = \begin{cases} 1 & z > 0 \\ 1/2 & \text{numerical approx at } z=0 \\ 0 & z < 0 \end{cases}$ //

$\lim_{z \to 0^+} f(z) = 1$ $\quad \lim_{z \to 0^-} f(z) = 0$

$\therefore \frac{1}{2}\left[\lim_{z \to 0^+} f(z) + \lim_{z \to 0^-} f(z) \right] = 1/2$ //