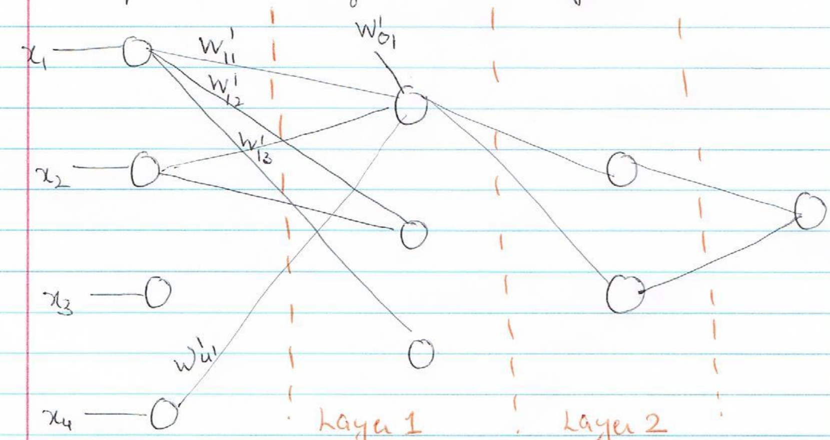
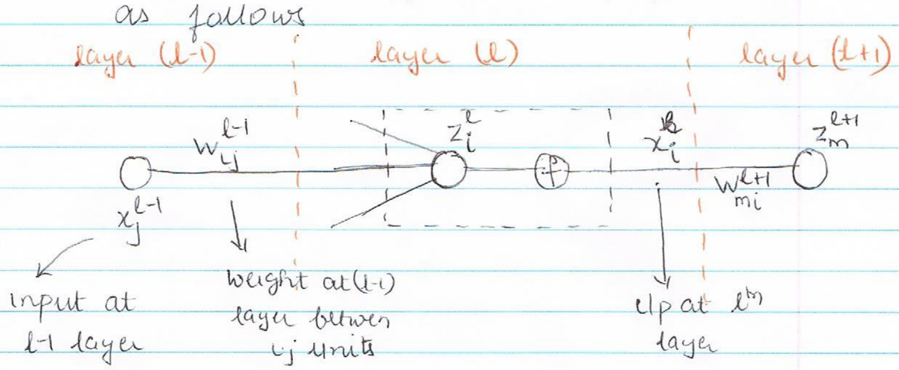
HW3

1. Q1

Q1 A representation of the multilayered n/w



This can be simplified to be represented as follows



- From here we get that input at layer l is the output of layer (l-1) & n/w feeds forward

Per the gradient descent update

$$w^{t+1} = w^t - \eta G(w^t)$$

From the diagram we can infer

$$Z_i^{l} = W_{ij}^{(l-1)} \cdot x_j^{(l-1)} \quad — (1)$$

hence $\cancel{\partial x_i}$ $\dfrac{\partial z_i^l}{\partial w_{ij}} = (x_j^{l-1}) \quad — (2)$

Also the assumption here is that
the transfer function $= \phi(z) = \cancel{\phi} \dfrac{1}{1+e^{-z}} \quad — (3)$

Let the output of the unit be $f_n(\underline{x})$

Error $=$ $E_{in} = \frac{1}{2}[y_{in} - f_n(x_i)]^2$

For the l^{th} layer

$$\frac{\partial E_{in}}{\partial z_i} = \frac{\partial}{\partial z_i}\frac{1}{2}[y_{in} - f_n(x_i)]^2$$

$$= \frac{1}{2}\frac{\partial}{\partial z_i}[y_{in} - \phi(z_i)]^2$$

$$= [y_{in} - \phi(z_i)]\left[\frac{\partial y_{in}}{\partial z_i} - \frac{\partial}{\partial z_i}\phi(z_i)\right]$$

if $\phi(z) = \dfrac{1}{1+e^{-z}}$ then $\phi'(z) = \dfrac{1}{(1+e^{-z})} \times \dfrac{e^{-z}}{(1+e^{-z})} = \phi(z)(1-\phi(z))$

hence

$$\frac{\partial E_{in}}{\partial z_i} = -[y_{in} - \phi(z_i)][\phi(z_i)][1-\phi(z_i)] \quad — (4)$$

$$\frac{\partial E_{in}}{\partial z_i} = -\delta_{in} \quad — \quad (5)$$

when $\delta_{in} = -[y_{in} - \phi(z_i)][\phi(z_i)][1-\phi(z_i)]$

Now

$$w^{t+1} = w^t - \eta G(w^t) \quad — \quad (6)$$

from (5) & (6) & (2)

$$w^{t+1} = w^t \mp \eta \left[\frac{\partial E_{in}}{\partial z_i} \cdot \frac{\partial z_i}{\partial w_{ij}} \right]$$

$$w^{t+1} = w^t + \eta \, \delta_{in} \cdot x_{ij} \quad — \quad (7)$$

Now consider the $(l+1)^{th}$ layer. The updates can be calculated using chain rule

$$\frac{\partial E_{in}}{\partial z_m} = \frac{\partial}{\partial z_m} [y_{in} - \phi(z_i)]^2$$

$$= -[y_{in} - \phi(z_i)] \cdot \frac{\partial \phi(z_m)}{\partial z_m}$$

$$= (1-\phi(z_m))(\phi(z_m) \cdot \delta_{in} w_{ij} \quad \text{from } (5)$$

$$\frac{\delta E_{in}}{\delta z_m} \simeq \delta_{mn} \quad — \quad (8)$$

$$\frac{\partial E_{in}}{\partial z_i} = \delta_{in}$$

from (5) & (8)

$$\frac{\partial E_{in}}{\partial z_m} = \delta_{im}$$

We can continue to apply chain rule for each hidden layer to get ~~&~~ δ's of the same form as above

Referring back to the diagram

(1) Calculate the feed forward $\phi(z_i)$

(2) Calculate the δ coefficient

$$\delta_{in} = \phi(z_i)(1-\phi(z_i)) \sum \delta_{jn} w_{ji}$$

(for all j's that are in layers downstream)

The update then is

$$w^{t+1} = w^t + \eta\, \delta_{in} x_{ij}$$

which can be recursively calculated

2. Q2

(1)

(Q2) $Q(\underline{w}) = \sum_{i=1}^{N} q_i(w)$ where $q_i(\underline{w})$ = error vector

$$q_i(w) = \frac{1}{2}\left[y_i - F(x_i)\right]^2$$

$$F(x_i) = a_0 + \sum_{m=1}^{M} a_m B(\underline{x} \mid u_m, \sigma_m)$$

$$= a_0 + \sum_{m=1}^{M} a_m \, e^{-\frac{1}{2\sigma_m^2} \sum_{i=1}^{n} (x_i - u_{im})^2]}$$

Parameters are a_m, σ_m & u_m

gradient w.r.t a_m

$$-\frac{\partial q_i(W)}{\partial a_m} = -\frac{\partial}{\partial a_m} \frac{1}{2}\left[y_i - a_0 - \sum_{m=1}^{M} a_m \, e^{-\frac{1}{2\sigma_m^2} \sum (x_i - u_{im})^2}\right]$$

$$= -[y_i - F(x_i)] \cdot \frac{\partial}{\partial a_m}\left[y_i - a_0 - \sum_{m=1}^{M} a_m \, e^{-\frac{1}{2\sigma_m^2} \sum_{i=1}^{n} (x_i - u_{im})^2}\right]$$

$$= -[y_i - F(x_i)]\left[0 - 0 - \sum_{m=1}^{M} e^{-\frac{1}{2\sigma_m^2} \sum_{i=1}^{n} (x_i - u_{im})^2}\right]$$

$$= -[y_i - F(x_i)]\, B(\underline{x} \mid u_m, \sigma_m) \quad \text{—— (1)}$$

gradient w.r.t u_m

$$-\frac{\partial q_i(w)}{\partial u_m} = \frac{-1}{2}\frac{\partial}{\partial u_m}\left[y_i - a_0 - \sum_{m=1}^{M} a_m e^{-\frac{1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_{lm})^2} \right]$$

$$= -[y_i - F(x_i)] \cdot \frac{\partial}{\partial u_m}\left[y_i - a_0 - \sum_{m=1}^{M} a_m e^{-\frac{1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_{lm})^2} \right]$$

$$= -[y_i - F(x_i)]\left[0 - 0 - \frac{\partial}{\partial u_m}\sum_{m=1}^{M} a_m e^{-\frac{1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_{lm})^2}\right]$$

$$= -[y_i - F(x_i)]\left[a_m e^{-\frac{1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_{lm})^2}\right] \cdot \frac{\partial}{\partial u_m}\left[\frac{-1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_{lm})^2\right]$$

$$= -[y_i - F(x_i)]\left[a_m \cdot e^{-\frac{1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_{lm})^2}\right)\left(\frac{-1}{2\sigma_m^2}\right)\left(2(x_i - u_{lm})(-1)\right)$$

$$= \frac{-a_m[y_i - F(x_i)]\left[e^{-\frac{1}{2\sigma_m^2}\sum_{l=1}^{n}(x_i - u_m)^2}\right]}{\sigma_m^2}[x_i - u_m] \quad — (2)$$

gradient w.r.t σ_m

$$-\frac{\partial q_i(w)}{\partial \sigma_m} = \frac{-1}{2}\frac{\partial}{\partial \sigma_m}\left[y_i - a_0 - \cancel{a_m}\sum_{m=1}^{M} a_m e^{-\frac{1}{2\sigma_m^2}\sum_{i=1}^{n}(x_i-u_m)^2}\right]$$

$$= -\frac{2}{2}[y_i - F(x_i)]\left[0 - 0 - \frac{\partial}{\partial \sigma_m}\sum_{m=1}^{M} a_m e^{-\frac{1}{2\sigma_m^2}\sum_{i=1}^{n}(x_i-u_m)^2}\right]$$

$$= -[y_i - F(x_i)]\left[-a_m e^{-\frac{1}{2\sigma_m^2}\sum_{i=1}^{n}(x_i-u_i)^2}\cdot\frac{\partial}{\partial \sigma_m}\left(-\frac{1}{2\sigma_m^2}\sum_{i=1}^{n}(x_i-u_m)^2\right)\right]$$

$$= -a_m[y_i - F(x_i)]\left[e^{-\frac{1}{2\sigma_m^2}\sum_{i=1}^{n}(x_i-u_m)^2}\cdot\sum(x_i-u_m)^2\frac{(-1)(-2)}{2\sigma_m^3}\right]$$

$$= -a_m[y_i - F(x_i)]\frac{\left[e^{-\frac{1}{2\sigma_m^2}\sum_{i=1}^{n}(x_i-u_m)^2}\right]\left[\sum_{i=1}^{n}(x_i-u_m)^2\right]}{\sigma_m^3} \quad -(3)$$

(1), (2), (3) show the derivative of gradient

3. Q3

Q3

$$B(X \mid u_m, \Sigma) = e^{-\frac{1}{2}[(x-u_m)^T \Sigma (x-u_m)]}$$

$\Sigma =$ positive definite and symmetric matrix

hence $\Sigma = \Sigma^{1/2} \Sigma^{1/2}$ — ①

Consider an input vector $\underline{x}$ that is pre-multiplied with $\Sigma^{1/2}$

i.e. i/p $= \Sigma^{-1/2} \underline{x} = \hat{\underline{x}}$

$$B(\hat{\underline{X}} \mid \hat{u}_m, \Sigma) = e^{-\frac{1}{2}[\Sigma^{1/2}(\hat{x}-\hat{u}_m)^T \Sigma \Sigma^{1/2}(\hat{x}-\hat{u}_m)}$$

$$= e^{-\frac{1}{2}[(\hat{x}-\hat{u}_m) \cdot \Sigma^{-1/2} \cdot \Sigma \cdot \Sigma^{-1/2}(\hat{x}-u_m)}$$

using ①

$$= e^{-\frac{1}{2}[(\hat{x}-\hat{u}_m)^T [I] (\hat{x}-\hat{u}_m)]}$$

$$B(\hat{\underline{X}} \mid \hat{u}_m, \Sigma) = e^{-\frac{1}{2}[(\hat{x}-\hat{u}_m)^T (x-u_m)]} \quad \text{— ②}$$

In problem ② if $\sigma_m = 1$

then $B(\underline{X} \mid u_m, \sigma_m) = e^{-\frac{1}{2}[x-u_m]^2}$ — ③

~~Since~~ ② & ③ are equivalent
hence proved

4. Q4

Q4

$$B(\underline{x} \mid \mu_m \Sigma_m) = e^{-\frac{1}{2}\left[(x-\mu_m)^t \Sigma_m (x-\mu_m)\right]}$$

→ In the std neural n/w the transfer function vary only in 1 direction of i/p space

→ Radial basis functions, vary in all directions equally

→ Σ_m controls the direction of variation

→ If there exists a vector $\bar{x}$ such that $\Sigma_m \underline{x} = \lambda \underline{x}$

ie if the $\bar{x}$ = eigen vector of Σ_m then the direction of variedness is just along that vector direction

5. Q5.

- 5a.

```
library(nnet)

inspam=read.csv("Spam_Train.txt")
spname<-c ("make", "address", "all", "3d", "our", "over", "remove",
              "internet","order", "mail", "receive", "will",
              "people", "report", "addresses","free", "business",
              "email", "you", "credit", "your", "font","000","money",
              "hp", "hpl", "george", "650", "lab", "labs",
              "telnet", "857", "data", "415", "85", "technology", "1999",
              "parts","pm", "direct", "cs", "meeting", "original", "project",
              "re","edu", "table", "conference", ";", "(", "[", "!", "$", "#",
              "CAPAVE", "CAPMAX", "CAPTOT","type")
colnames(inspam)=spname
x=inspam
colnames(x)=spname
x$type=as.factor(inspam$type)
x[,1:57]=scale(x[,1:57], center=TRUE, scale=TRUE)
x=data.frame(x)
colnames(x)=spname


inspamtest=read.csv("Spam.Test.txt")
colnames(inspamtest)=spname
w=inspamtest
colnames(w)=spname
w$type=as.factor(as.factor(inspamtest$type))
w[,1:57]=scale(w[,1:57],center=TRUE, scale=TRUE)
w=data.frame(w)
colnames(w)=spname
w.type=w$type

#Find the size with minimum error
set.seed(1)
for(i in 1:10){
  nn1=nnet(type~.,data=x, size=i, maxit=5000, decay=0.0, rang=0.5, trace=F)
  nn1.predict=predict(nn1, newdata = w,type="class")
  nn1.out=nn1.predict
  u=matrix(data=0,2,2)
  u=table(nn1.out, w.type)
  err=sum(nn1.out!=w.type)/(length(nn1.out))
  print(paste0("# of hidden nodes = ",i," and error = ", err))
}

## [1] "# of hidden nodes = 1 and error = 0.0730593607305936"
## [1] "# of hidden nodes = 2 and error = 0.0743639921722113"
## [1] "# of hidden nodes = 3 and error = 0.0652315720808871"
## [1] "# of hidden nodes = 4 and error = 0.060665362035225"
## [1] "# of hidden nodes = 5 and error = 0.0639269406392694"
## [1] "# of hidden nodes = 6 and error = 0.0574037834311807"
```

```
## [1] "# of hidden nodes = 7 and error = 0.0600130463144162"
## [1] "# of hidden nodes = 8 and error = 0.0547945205479452"
## [1] "# of hidden nodes = 9 and error = 0.0678408349641226"
## [1] "# of hidden nodes = 10 and error = 0.108936725375082"
```

```
print(paste0("Minimum Error is with # of hidden nodes = ", 8))
```

```
## [1] "Minimum Error is with # of hidden nodes = 8"
```

```
  + By using a # of hidden nodes as 8, we get  overall error rate ~4-5%
```

- 5b.

```
set.seed(1)
res=matrix(NA, length(nn1.out),11)
ii=1;
for(j in seq(0,1,0.1)){
  nn1=nnet(type~.,data=x, size=8, maxit=5000, decay=j, rang=-0.5, trace=F)
  nn1.predict=predict(nn1, newdata = w[,1:57],type="class")
  nn1.out=nn1.predict
  res[,ii]=nn1.out
  ii=ii+1;
  err=sum(nn1.out!=w.type)/(length(nn1.out))
  print(paste0("Decay = ",j," and error = ", err))
}
```

```
## [1] "Decay = 0 and error = 0.065883887801696"
## [1] "Decay = 0.1 and error = 0.0482713633398565"
## [1] "Decay = 0.2 and error = 0.0547945205479452"
## [1] "Decay = 0.3 and error = 0.0430528375733855"
## [1] "Decay = 0.4 and error = 0.0437051532941944"
## [1] "Decay = 0.5 and error = 0.0476190476190476"
## [1] "Decay = 0.6 and error = 0.0521852576647097"
## [1] "Decay = 0.7 and error = 0.0476190476190476"
## [1] "Decay = 0.8 and error = 0.0528375733855186"
## [1] "Decay = 0.9 and error = 0.0476190476190476"
## [1] "Decay = 1 and error = 0.0547945205479452"
```

```
print(paste0("Minimum Error is with value of decay as = ", 0.6))
```

```
## [1] "Minimum Error is with value of decay as = 0.6"
```

```
nn1=nnet(type~.,data=x, size=8, maxit=5000, decay=0.6, rang=-0.5, trace=F)
nn1.best=predict(nn1, newdata = w[,1:57],type="class")

#Finding the class through majority vote:
vote=rep(NA,length(nn1.out))
for (i in 1:length(nn1.out)){
  if(sum(res[i,]==1)>sum(res[i,]==0))
  {vote[i]=1}
  else{
    vote[i]=0
  }
}

#Calcualte error
err=sum(vote!=w.type)/(length(nn1.out))
print(paste0("Error using the majority of votes is  ", err))
```

```
## [1] "Error using the majority of votes is  0.0430528375733855"
```

```
  + Best Model:
    + Decay: 0.6
    + Number of hidden units: 8
    + Error:~3-4%



  + By using an ensemble, where we find majority of votes for a class, the error is about
4%
```

- 5c.

```
set.seed(1)
for(k in seq(0,1,0.1)){
      nn1=nnet(type~.,data=x, size=8, maxit=5000, decay=0.6, trace=F)
      nn1.predict=predict(nn1, newdata = w,type="raw")
      nn1.out=rep(0,length(nn1.predict))
      nn1.out[nn1.predict>k]=1
      u=matrix(data=0,2,2)
      u=table(w.type,nn1.out)

      print(paste0("Threshold= ", k , " Proportion of good mails misclassified is: ", u[3
][1]/(u[1][1]+u[3][1]) ))
}
```

```
## [1] "Threshold= 0 Proportion of good mails misclassified is: 0.998908296943231"
## [1] "Threshold= 0.1 Proportion of good mails misclassified is: 0.140829694323144"
## [1] "Threshold= 0.2 Proportion of good mails misclassified is: 0.0927947598253275"
## [1] "Threshold= 0.3 Proportion of good mails misclassified is: 0.0676855895196507"
## [1] "Threshold= 0.4 Proportion of good mails misclassified is: 0.0524017467248908"
## [1] "Threshold= 0.5 Proportion of good mails misclassified is: 0.0305676855895196"
## [1] "Threshold= 0.6 Proportion of good mails misclassified is: 0.0316593886462882"
## [1] "Threshold= 0.7 Proportion of good mails misclassified is: 0.0196506550218341"
## [1] "Threshold= 0.8 Proportion of good mails misclassified is: 0.0163755458515284"
## [1] "Threshold= 0.9 Proportion of good mails misclassified is: 0.00655021834061135"
## [1] "Threshold= 1 Proportion of good mails misclassified is: NA"
```

```
  + By using a Threshold of 90% of classifiying an email as spam, we can get misclassific
ation of good email as spam down to <1% error rate.
```