Tutorial 10

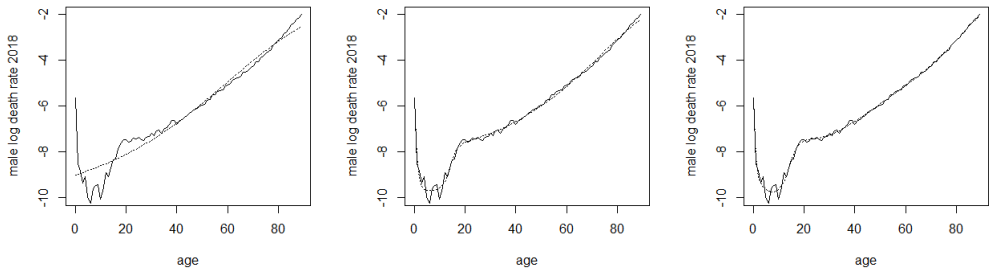
10.1

```
raw<-read.table("Australia Data.txt",header=TRUE)
age=90; year=49
d<-array(NA,c(age,year))
e<-array(NA,c(age,year))
m<-array(NA,c(age,year))
h=1
for (t in 1:year) {
for (x in 1:age) {
d[x,t]=raw$dxMale[h]
e[x,t]=raw$exMale[h]
m[x,t]=d[x,t]/e[x,t]
h=h+1 }
h=h+21 }

x=c(1:90)/100
mx=max(log(m[,year]))
mn=min(log(m[,year]))
t=(log(m[,year])-mn)/(mx-mn)*(0.9-0.1)+0.1
```

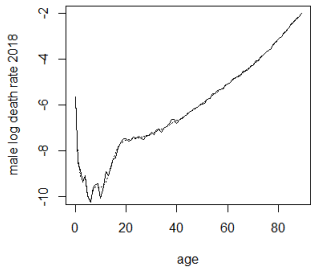
Then use the codes in the lecture slides and perform trial and error

```
plot(c(0:89),(z-0.1)/0.8*(mx-mn)+mn,xlab="age",ylab="male log death rate
2018",ylim=c(-10,-2),type="l",lty=3)
lines(c(0:89),(t-0.1)/0.8*(mx-mn)+mn)
```

hids = 1, 3, 5; epoch = 100000



hids = 30; epoch = 1000000



It appears that 5 to 10 hidden nodes would be sufficient for describing the entire mortality curve and allowing for the major patterns (i.e. infant mortality drop, accident hump, old-age exponential growth). Using too few hidden nodes does not provide adequate modelling capacity. On the other hand, using 30 hidden nodes (i.e. 91 weights) reduces the fitting error further but clearly leads to overfitting and so poor generalisation.

10.2

```
raw<-read.table("Individual Claims.txt",header=TRUE)

x<-array(NA,c(2,100))
t<-numeric()
for (k in 1:100) {
t[k]=raw$Line[k]
x[1,k]=raw$Claim[k]/60000
x[2,k]=raw$Time[k]/5 }

n=100; hids=5; epoch=100000
y<-array(NA,c(hids,n))
z<-numeric()

logistic<-function(s) {
if (s>700) { logistic=1 }
if (s<=700) { logistic=exp(s)/(1+exp(s)) }
logistic }
```

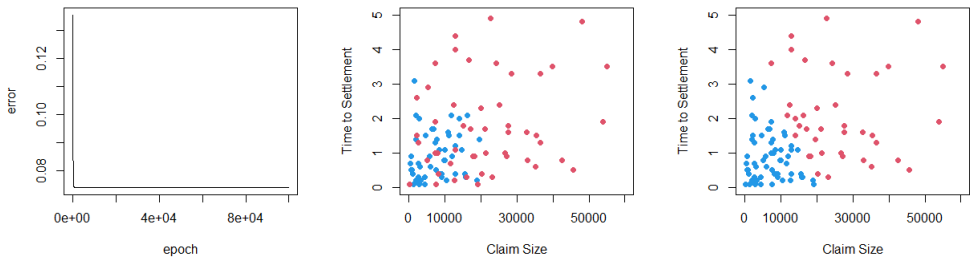
Note: $y_{j,k} = f\,(a_{0,j} + a_{1,j}\,x_{1,k} + a_{2,j}\,x_{2,k})$

```
e<-numeric()
a<-array(NA,c(3,hids))
for (i in 1:3) { for (j in 1:hids) {
a[i,j]=runif(1,-1,1) } }
b=-1+rbinom(hids+1,1,0.5)*2

epsilona<-array(0.5,c(3,hids))
epsilonb=rep(0.5,hids+1)
lambda=0.1; phi=0.5; theta=0.7

for (k in 1:n) {
for (j in 1:hids) {
y[j,k]=logistic(a[3,j]+a[1,j]*x[1,k]+a[2,j]*x[2,k]) }
z[k]=logistic(b[hids+1]+sum(b[1:hids]*y[1:hids,k])) }

for (h in 1:epoch) {
da<-array(0,c(3,hids))
db<-rep(0,hids+1)

for (k in 1:n) {

p=(z[k]-t[k])*z[k]*(1-z[k])/n
for (j in 1:hids) { db[j]=db[j]+p*y[j,k] }
db[hids+1]=db[hids+1]+p

for (j in 1:hids) {
q=(z[k]-t[k])*z[k]*(1-z[k])*b[j]*y[j,k]*(1-y[j,k])/n
da[1,j]=da[1,j]+q*x[1,k]
da[2,j]=da[2,j]+q*x[2,k]
da[3,j]=da[3,j]+q }

}
```

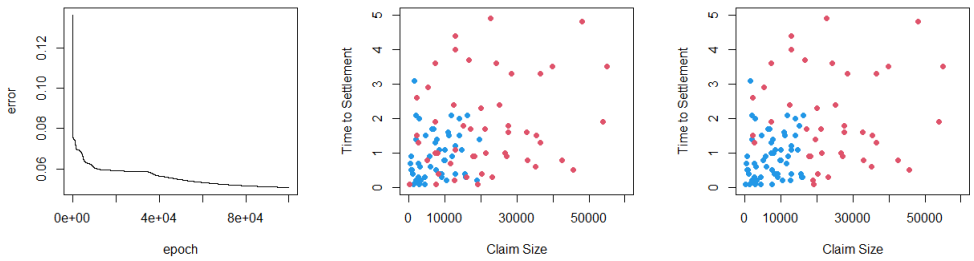
```
if (h>1) {
for (i in 1:3) { for (j in 1:hids) {
if (ga[i,j]*da[i,j]>0) {
epsilona[i,j]=epsilona[i,j]+lambda }
if (ga[i,j]*da[i,j]<=0) {
epsilona[i,j]=epsilona[i,j]*phi }
}}
for (j in 1:(hids+1)) {
if (gb[j]*db[j]>0) {
epsilonb[j]=epsilonb[j]+lambda }
if (gb[j]*db[j]<=0) {
epsilonb[j]=epsilonb[j]*phi }
}}

a=a-epsilona*da
b=b-epsilonb*db

if (h==1) { ga=da; gb=db }
if (h>1) {
ga=theta*ga+(1-theta)*da
gb=theta*gb+(1-theta)*db }

for (k in 1:n) {
for (j in 1:hids) {
y[j,k]=logistic(a[3,j]+a[1,j]*x[1,k]+a[2,j]*x[2,k]) }
z[k]=logistic(b[hids+1]+sum(b[1:hids]*y[1:hids,k])) }

e[h]=0.5*sum((z-t)^2)/n
}

plot(c(1:epoch),e,xlab="epoch",ylab="error",type="l")

plot(x[1,1:50]*60000,x[2,1:50]*5,xlab="Claim Size",ylab="Time to
Settlement",xlim=c(0,60000),ylim=c(0,5),pch=16,col=4)
points(x[1,51:100]*60000,x[2,51:100]*5,pch=16,col=2)
```

```
temp1<-numeric()
temp2<-numeric()
temp3<-numeric()
temp4<-numeric()
for (k in 1:100) {
if (z[k]<0.5) {
temp1=c(temp1,x[1,k])
temp2=c(temp2,x[2,k]) }
if (z[k]>0.5) {
temp3=c(temp3,x[1,k])
temp4=c(temp4,x[2,k]) }
}

plot(temp1*60000,temp2*5,xlab="Claim Size",ylab="Time to Settlement",
xlim=c(0,60000),ylim=c(0,5),pch=16,col=4)
points(temp3*60000,temp4*5,pch=16,col=2)
```

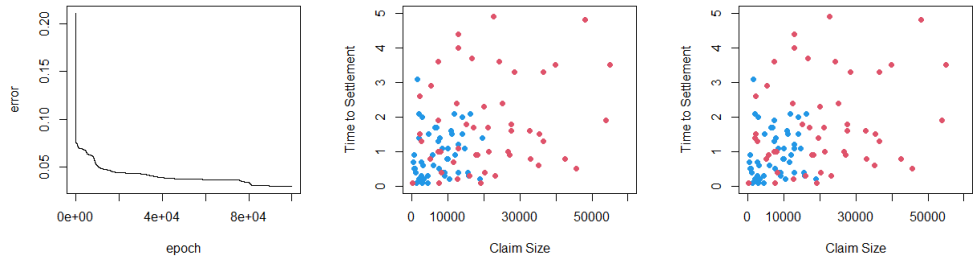
hids = 1; epoch = 100000



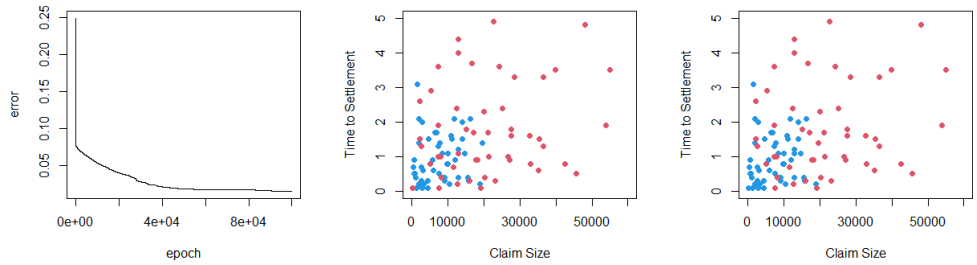
hids = 5; epoch = 100000



hids = 10; epoch = 100000



hids = 25; epoch = 100000



It is difficult to select the optimal number of hidden nodes here, as there is no extra information about the usual patterns of these claims (unlike what we know about human mortality). It seems that the claims in Line 1 tend to have a larger claim size and longer time to settlement, but there are also some mixed cases. Using too few hidden nodes leads to a simple (but not very accurate) split between the two groups, while using too many hidden nodes generates a (almost) perfect classification, which means overfitting. Those rather mixed cases can be either merely noises or explained by extra features (if available).

If the sample size is larger, the data can be split randomly by 60% / 20% / 20%, and validation and testing can be performed to identify the optimal model.