Glasgow Short Course, 12 June 1996

Neural Networks Practical

Start up S-Plus, and type

```
library(nnet.gla)
```

These should be used every time you start. It will bring up a Trellis graphics window, and add a Neural Nets item to the S-Plus menu.

1 Demonstration example

The demonstration in the lectures used Trellis, but it is rather complicated to set this up, so I've hidden it in functions. You can run the lecture demonstrations by using the items from the Neural Nets drop-down menu in turn. But for those who want to know how, the direct S-Plus commands follow.

```
#  EITHER generate the data
x <- 1:250 *pi/250
set.seed(111)
f <- 4.26*(exp(-x) - 4 * exp(-2*x) + 3 * exp(-3*x))
y <- rnorm(250,f, 0.2)
vx <- matrix(x, ncol=1)
# OR just
attach(glademo)

# plot it
plot(x, y, xlab="", ylab="", type="n")
points(x, y, cex=0.5, col=2)
lines(x, f, lty=2)

# plot the polynomials
lines(x, lm(y ~ poly(x,6))$fitted, col=5)
lines(x, lm(y ~ poly(x,10))$fitted, col=3)
lines(x, lm(y ~ poly(x,16))$fitted, col=4)
# OR
do250poly()  # look at this if you like, by typing do250poly

# do some NN fits
sp.nn2 <- nnet(x, y, linout=T, size=2, maxit=1000)
plt.nn(sp.nn2,"2 hidden units")
sp.nn4 <- nnet(x, y, linout=T, size=4, maxit=1000)
plt.nn(sp.nn4,"4 hidden units")
sp.nn8 <- nnet(x, y, linout=T, size=8, maxit=1000)
plt.nn(sp.nn8,"8 hidden units")
```



```
# and again
lines.nn(nnet(x, y, linout=T, size=8, maxit=1000), col=3)
# and with weight decay
lines.nn(nnet(x, y, linout=T, size=8, decay=0.01, maxit=1000), col=4)
```

Now try other numbers of hidden units and weight decay (up to $\lambda = 0.1$?).

2 Cushing's syndrome

Here we always use skip-layer connections, that is direct connections from input to the soft-max output layer. This is almost always a good idea.

```
# plot it
trellis.device()
plt("Cushing's syndome & logistic")

# Logistic regression
cush.nn <- nnet(cush, tp,  skip=T, softmax=T, size=0, maxit=1000)
bndry(cush.nn)

# Neural nets
plt()
cush.nn <- nnet(cush, tp,  skip=T, softmax=T, size=2, maxit=1000)
bndry(cush.nn, col=2)
```

and so on. To average over fits:

```
plt("Many local minima")
Z <- matrix(0, nrow(cushT), ncol(tp))
for(iter in 1:20) {
    set.seed(iter)
    cush.nn <- nnet(cush, tp,  skip=T, softmax=T, size=3,
        decay=0.01, maxit=1000, trace=F)
    Z <- Z + predict(cush.nn, cushT)
    cat("final value", format(round(cush.nn$value,3)), "\n")
    bndry(cush.nn, col=2)
}
b1(Z, col=8)
plt("Averaging over local minima")
b1(Z, col=8)
```

3 Forensic Glass

Details of this dataset are in the paper Ripley (1996b) which is a handout (and also in Ripley, 1996).

```
attach(fgl)
# Look at the variables in Trellis
fglbw <- data.frame(type = rep(type, 9), y = as.vector(as.matrix(fgl0)),
meas = factor(rep(1:9, rep(214,9)), labels = names(fgl0)))
bwplot(type ~ y | meas, data=fglbw, scales=list(x="free"))
detach()

# do logistic regression
set.seed(777)
fgl.nn <- nnet(fglA, fglY, softmax=T, skip=T, size=0,
                maxit=250, decay=0.001)
summary(fgl.nn)
con(fgl$type, factor(apply(predict(fgl.nn, fglA),1, which.is.max),
    labels=levels(fgl$type)))

# 10-fold CV partition
rand <- sample(10, nrow(fgl), replace=T)
con(fgl$type, CV.nn(size=0, maxit=250, decay=1e-3) )
# and slowly:
con(fgl$type, CV.nn(size=4, maxit=250, decay=1e-3) )
```

The last two commands do a 10-fold cross-validatory assessment of performance, for a logistic regression and a 4-hidden-unit neural network respectively. (Four was chosen from a larger set of trials.)