Department of Mathematics and Computer Science
University of Southern Denmark, Odense

February 13, 2013
Marco Chiarandini

DM825 - Introduction to Machine Learning

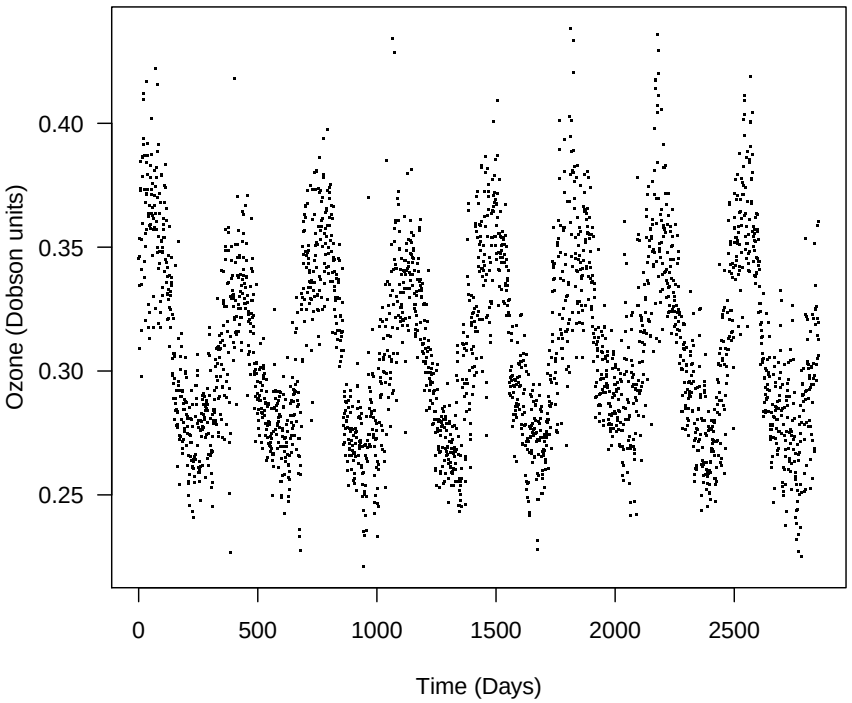
Sheet 6, Spring 2013

Exercise 1 – Neural Networks for Time Series Prediction

A common data analysis task is time series prediction, where we have a set of data that show something varying over time, and we want to predict how the data will vary in the future. Examples are stock markets, river levels and house prices.

The data set PNoz.dat contains the daily measurement of the thickness of the ozone layer above Palmerston North in New Zealand between 1996 and 2004. Ozone thickness is measured in Dobson units, which are 0.01 mm thickness at 0 degree Celsius and 1 atmosphere pressure. The reduction in stratosferic ozone is partly responsible for global warming and the increased incidence of skin cancer. The thickness of the ozone varies naturally over the year, as you can see from the plot. (There are four fields in the data, and the ozone level is the third).

```
PNoz <- read.table("PNoz.dat")
names(PNoz) <- c("year","day","ozone.level","sulphur.dioxide.level")
plot(PNoz$ozone.level,xlab="Time␣(Days)",ylab="Ozone␣(Dobson␣units)",pch="
    .",cex=1.5)
```



Solution

Your task is to use the multi-layer perceptron to predict the ozone levels into the future and see if you can detect an overall drop in the mean ozone level. Plot 400 predicted values together with the actual value.

The following is a reminder of the steps to carry out in the analysis:

- Select inputs and outputs for your problem and consequently the input and output nodes for the network.
- Normalize the data by rescaling.
- Split the data into training, validation and test (use the rule 50/25/25 if enough data or use cross validation with little data).
- Identify the main parameters to configure, e.g., the network architecture and others.
- Train the network and compare for different parameters
- Assess the performance on the test data.
- Analyse the bias and variance trade off.

Solution We wish to detect regularities in the time series. Th eproblem is that it can appear over many different scales. How many data points should we use and how far apart in time should we space those? Let $h(\cdot)$ be the function that represents the neural network:

$$y = x(t+\tau) = h(x(t), x(t-\tau), \ldots, x(t-\tau))$$

Let's prepare the data.

```
> tau <- 2
> D <- 3
> N <- dim(PNoz)[1]
> Ntest <- 800
```

Preprocessing the input data can help in improving the performance.

```
> # Normalise data
> PNoz$ozone.norm <- PNoz[,3]-mean(PNoz[,3])
> PNoz$ozone.norm = PNoz$ozone.norm/max(PNoz$ozone.norm)
```

We arrange the data in matrix format and we split the data into two parts, one part will be used for the assessment of the model:

```
> lastPoint = N-tau*(D)
> inputs = matrix(0, nrow=lastPoint, ncol=D)
> targets = matrix(0, nrow=lastPoint, ncol=1)
> for (i in 1:lastPoint) {
   inputs[i,] = PNoz[seq(i,i+tau*(D-1),tau),"ozone.norm"]
   targets[i] = PNoz[i+tau*D,"ozone.norm"]
 }
> # test
> idx <- lastPoint:(lastPoint-Ntest)
> test <- inputs[idx,]
```

```
> testtargets = targets[idx]
> # training
> inputs <- inputs[-idx,]
> targets <- targets[-idx]
```

It is a good idea to randomize the order of the data in the training set:

```
> # Randomly order the data
> mask <- sample(1:dim(inputs)[1])
> inputs.s <- inputs[mask,]
> targets.s <- targets[mask]
```

We further divide th edata in training and validation. The validation set will be used to tune the parameter D.

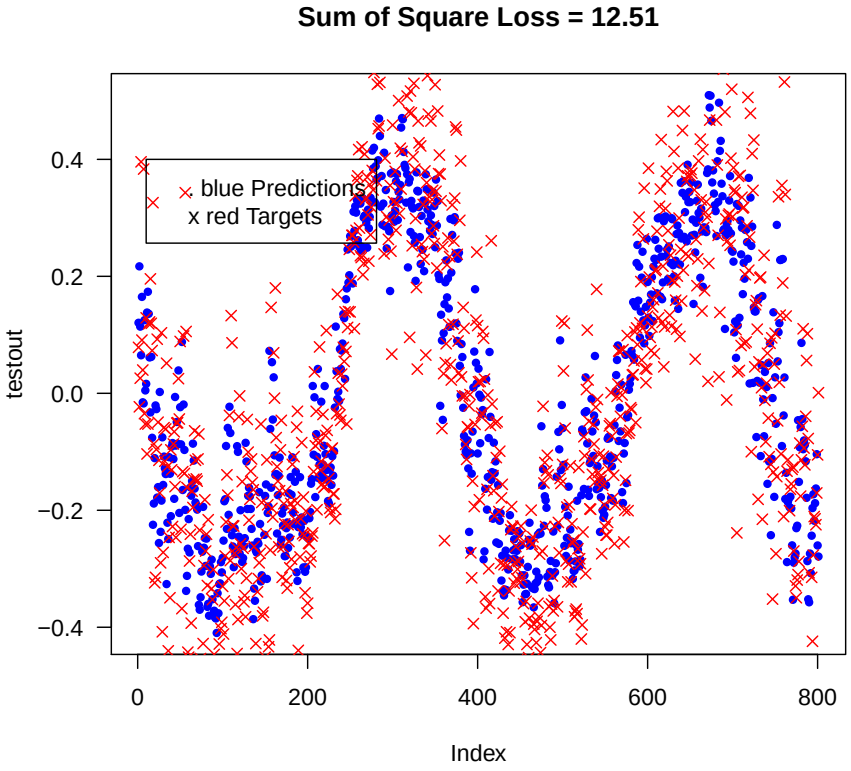
```
> # Let's divide training and validation
> train <- inputs.s[seq(1,length(mask),2),]
> traintargets <- targets.s[seq(1, length(mask), 2)]
> valid <- inputs.s[seq(2, length(mask), 2),]
> validtargets <- targets[seq(2, length(mask), 2)]
```

We are then ready to train the network

```
> # Train the network
> library(nnet)
> M <- D
> net <- nnet(train,traintargets,size=M,linout=TRUE)
> testout <- predict(net,test)
> error <- 0.5*sum((testtargets-testout)^2)
```

Let's have a visual inspection on how things went:

```
> plot(testout,pch=20,col="blue")
> points(testtargets,pch=4,col="red")
> legend(10,0.4,c(". blue Predictions","x red Targets"))
> title(paste("Sum of Square Loss =",round(error,2)))
```

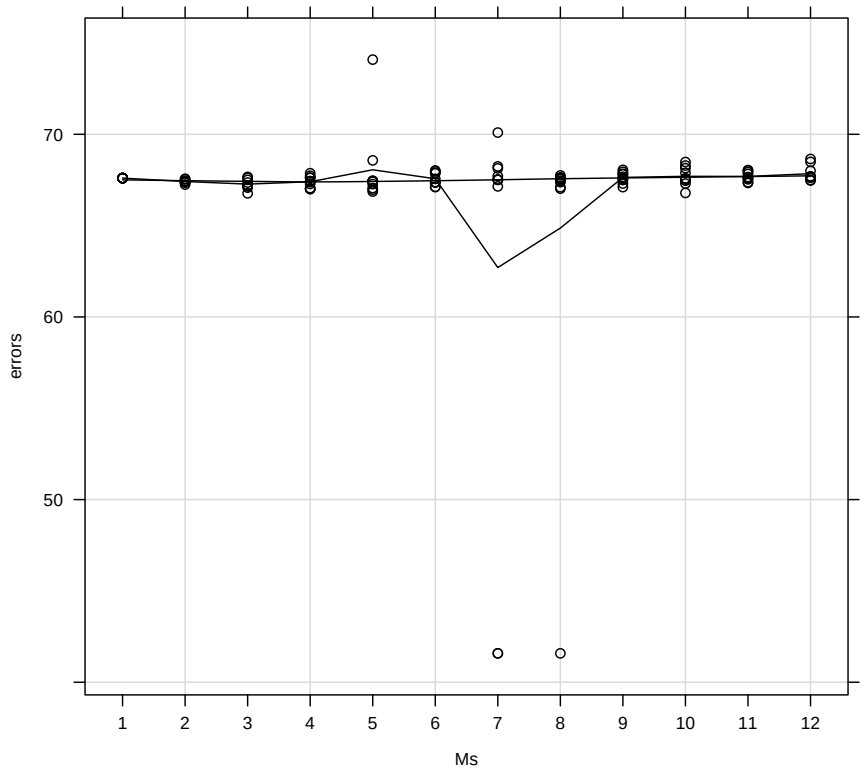


We may wish to use the validation set to improve the value of D.

```
> evaluate.model <- function(M) {
   net <- nnet(train,traintargets,size=M,linout=TRUE,trace=FALSE)
   validout <- predict(net, valid)
   0.5*sum((validtargets-validout)^2)
 }
> Ms <- (1):(4*D)
> errors <- matrix(0,ncol=10,nrow=length(Ms))
> for (i in 1:length(Ms)) {
   errors[i,] <- sapply(1:10,function(x) evaluate.model(Ms[i]))
 }
```

and we plot the bias variance trade off trend:

```
> wide <- as.data.frame(errors)
> long <- reshape(wide,direction="long",timevar="trial",
                  idvar="Ms",varying=list(1:10),v.names="errors")
> long$Ms <- factor(long$Ms,levels=Ms)
> print(
   xyplot(errors ~ Ms,data=long,type=c("p","a","smooth","g"))
   )
```



A similar analysis should be done for τ.

Exercise 2 – Prediction of count outcomes

The software engineers of an online shopping portal wish to predict the number of clicks that a new product will receive in their online price search engine. New products are those that are not yet in the data base of the portal and for which there is no historical data available. The engineers would like to charge the vendors of the products on the basis of these predictions.

When a vendor inserts its product in the data base, the products are classified as belonging to an existing category or initiating a new category. In the following, we will assume that the category of a new product is known and it comprises products with similar names and characteristics.

The engineers collected historical data including the following variables:

- offer_id: a numerical identifier of the entry
- product_name: the name of the product (this field is at the moment problematic as it exhibits no formalism)
- category_id: a numerical identifier for the category.
- price: the price of the product in euro (-1 if not given)
- deliver_price: the price for delivery (-1 if not given)
- total_price: the sum of the previous two prices (-1 if not given)
- availability: -1 see merchant site, 0 not available, 1 available, 2 available soon, 3 limited availability

- merchant: an encrypted name of the vendor
- number_reviews: the number of reviewers who rated the vendor
- average_rating: the average rating of the reviews
- number_clicks: the response variable indicating the number of clicks.

The historical data are relative to one single day and the prediction is asked on the same unit of time.
A useful nonlinear regression model when the outcome is a count, with large-count outcomes being rare events, is the Poisson model. The Poisson probability distribution is given by

$$p(x) = \frac{\lambda^x e^{-\lambda}}{x!} \quad x = 0, 1, 2, \ldots$$

Your tasks:

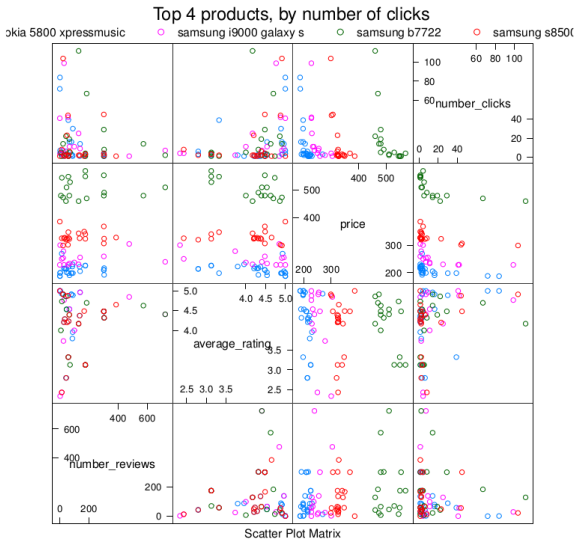
(a) The data set is confidential and it has been made available to you from the BlackBoard system. Retrieve the data set, load it in R via `load("clicks.rda")` and examine its content via `str(clicks)` and `head(clicks)`. You may explore possible relations between variables with the aid of some graphical representation like those in the example of the `splom` function of the package `lattice`. Use the obtained insights to preprocess the data and identify a subset of variables to use as predictors. Report the most significant insights eventually also with plots.[1]

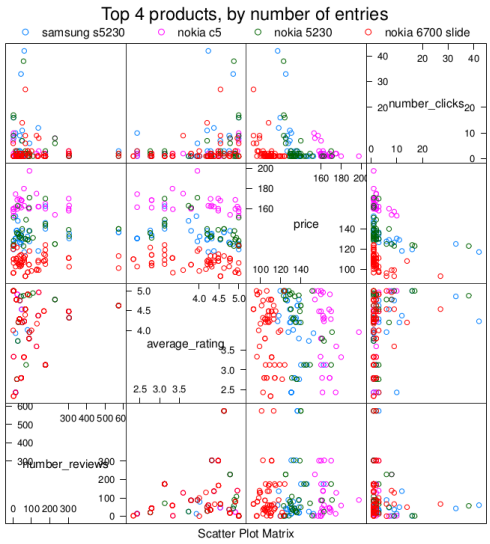
Solution

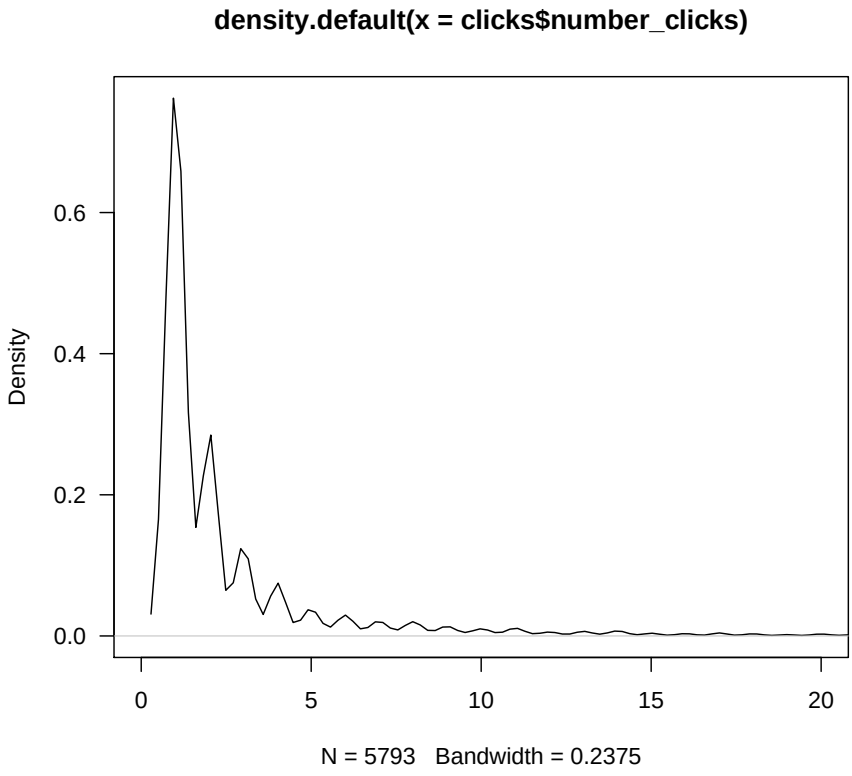
Looking at the density plot of the number of clicks variable, it seems reasonable to assume that it is Poisson-distributed:

```
> set.seed(1050)
> load("clicks.rda")
> str(clicks)
> head(clicks)
> plot(density(x = clicks$number_clicks),xlim=c(0,20))
```

[1] Note that for plotting to a graphical device like pdf with the functions from the lattice package, you need to enclose the call of the function in print.







(b) Model the prediction task by means of generalized linear regression using the Poisson distribution. More specifically, indicate the consequent link function for the GLM.

Solution The density function of the Poisson distribution is:

$$p(x \mid \mu) = \frac{e^{-\mu}\mu^x}{x!} = \frac{e^{(\log \mu)x - \mu}}{x!}$$

To rewrite this in the form of the exponential family:

$$p(x \mid \eta) = h(x)g(\eta)\exp(\eta T u(x))$$

we set $\eta = log\mu$, $g(\eta) = \exp(-exp(\eta))$, $u(x) = x, h(x) = 1$

The link funciton g is $\mu = exp(\eta) = g^{-1}(\eta) \Rightarrow g(\eta) = \log(\eta)$. The canonical response is $g^{-1}(x) = \exp(\eta)$.

(c) Indicate an appropriate measure for a quantitative assessment of the predictor.

(d) Which other predictive model studied in class can be applied to this task?