Boston

Beate and Oliver

1/16/2020

Goal of this script

We want to implement linear transformation models in NN and compare the achieved NLL and estimated ceofficients with the MLT results.

We fit a transformation function $h : (y|x) \to (z|x)$ with the property $(z|x) = h(y|x) \sim N(0, 1)$

In a **linear** transformation model the transformation function has the special form: $h_Y(y) - \sum_i \beta_i x_i$

Then we know, that.

- $F_{Y|X=x}(y) = F_z(h_Y(y) - \sum_i \beta_i x_i)$

Importing the required packages

```
library(MASS)
library(ggplot2)
library(mlt)
```

```
## Loading required package: basefun
```

```
## Loading required package: variables
```

```
##
## Attaching package: 'variables'
```

```
## The following object is masked from 'package:ggplot2':
##
##     unit
```

```
library(basefun)
library(keras)
library(tensorflow)
library(tfprobability)
```

```
## Warning: package 'tfprobability' was built under R version 3.6.2
```

```
T_STEPS = 2000
```

Source functions h and h_dash in w and w/o batch magic

```
# source("mlt_utils.R")  # eg scaling fct
# # preparing eval_h an eval_h_dash, fct implemented in tfp
# source("mlt_utils_keras_v2.R")  # causes error when knittering
#source('https://raw.githubusercontent.com/tensorchiefs/dl_playr/master/mlt/bern_utils.R')
#source('~/Documents/workspace/dl_playr/mlt/bern_utils.R')
```

```
source('bern_utils.R')

source('data.R')
```

Loading the data

We scale the y-varible to [0,1]

```
xy_dat = get_data_boston()
```

```
##  [1] "Names in X : crim"    "Names in X : zn"      "Names in X : indus"
##  [4] "Names in X : chas"    "Names in X : nox"     "Names in X : rm"
##  [7] "Names in X : age"     "Names in X : dis"     "Names in X : rad"
## [10] "Names in X : tax"     "Names in X : ptratio" "Names in X : b"
## [13] "Names in X : lstat"
```

```
dat = xy_dat$dat
sum(dat$y**2) # 299626.3 to compare with BH data in paper
```

```
## [1] 97.90634
```

```
dat$y_obs = dat$y
dat$y = NULL
y_range = xy_dat$scale
dat$y_scale = dat$y_obs
dat$y_obs = NULL
x = xy_dat$x
y = xy_dat$y
```

Defining the model

We set up the formula for the model:

```
fm_large = (y_scale ~ crim + zn + indus + chas + nox + rm + age + dis + rad + tax + ptratio + b + lstat)
#fm_small = (y_scale ~  rm + lstat) #lm log lik 346
#fm_uni = (y_scale ~  rm)
(fm = fm_large)
```

```
## y_scale ~ crim + zn + indus + chas + nox + rm + age + dis + rad +
##     tax + ptratio + b + lstat
```

```
is_univariate = TRUE
sum(dat$rm**2)  # 20234.6 to compare with BH data in paper
```

```
## [1] 20234.6
```

Baseline Linear Model

```
fit_lm = lm(fm, data=dat)
fit_lm$coef
```

```
##   (Intercept)          crim            zn         indus          chas
##  6.990997e-01 -2.400252e-03  1.031566e-03  4.568584e-04  5.970520e-02
##           nox            rm           age           dis           rad
```

```
## -3.948136e-01  8.466367e-02  1.538277e-05 -3.279037e-02  6.801100e-03
##           tax       ptratio             b         lstat
## -2.741021e-04 -2.117216e-02  2.069263e-04 -1.166130e-02
```

```
(logLik_lm=logLik(fit_lm) )/nrow(dat) + log(y_range)# the larger the better
```

```
## 'log Lik.' 4.651261 (df=15)
```

MLT fit and results

Variable and Model definition and fit

```
nb = 8  # order defining the Number of Bernstein fct in polynom
len_theta = nb+1
# specify a numeric variable with data in [0,1] and principle bounds [0,Inf]
var_y <- numeric_var("y_scale", support = c(0, 1), bounds = c(-Inf, Inf), add = c(0,0))
# what is done with the bound information (default bounds c(-INF, INF)

# set up monoton increasing polynomial of order nb with Bernstein basis function
bb <- Bernstein_basis(var_y, order=nb, ui="increasing")

# set up grid in interval supp+add -> gives data.frame with col y_scale
y_grid <- as.data.frame(mkgrid(bb, n = 500))

# set up model for mlt
ctm = ctm(bb, shift=fm[-2L], data=dat, todistr="Normal")
#~-1 + crim
#ctm = ctm(bb, shift = ~ b + crim - 1, data=dat, todistr="Normal")
# fm[-2L] defnes the basis function for the shift term h_y(y) in h(y|x)=h_y(y)+h_x(x)
# the intercept is included in the baseline-trafo h_y(y) (not in linear predictor h_x(x))
```

Fit of the model:

```
# fit the mlt model
mlt_fit <- mlt(ctm, data = dat, verbose=TRUE)
```

logLik with MLT

```
(logLik_mlt = logLik(mlt_fit))  #  df = nr-theta + nr-beta
```

```
## 'log Lik.' 567.4294 (df=22)
```

```
# compare to logLik of the baseline model - the larger the better
NLL_MLT = -logLik_mlt / nrow(dat) + log(y_range)
```

Estimated coefficients with MLT

Get the coefficients of the trafo h from the mlt fit:

```
( mlt_fit$coef )
```

```
##  Bs1(y_scale)  Bs2(y_scale)  Bs3(y_scale)  Bs4(y_scale)  Bs5(y_scale)
## -12.949148883  -9.935969361  -9.926925173  -4.953969841  -4.099484102
```

```
##  Bs6(y_scale)  Bs7(y_scale)  Bs8(y_scale)  Bs9(y_scale)          crim
##  -3.552333582  -3.543251848  -3.539084940  -2.253324090   0.044629187
##            zn         indus          chas           nox            rm
##  -0.006412394  -0.010904406  -0.583180972   4.724995436  -0.467408838
##           age           dis           rad           tax       ptratio
##   0.002101093   0.293249769  -0.079827513   0.003507470   0.225942673
##             b         lstat
##  -0.002572815   0.161517358
```

```
( theta = mlt_fit$coef[1:(nb+1)] )
```

```
## Bs1(y_scale) Bs2(y_scale) Bs3(y_scale) Bs4(y_scale) Bs5(y_scale)
##   -12.949149    -9.935969    -9.926925    -4.953970    -4.099484
## Bs6(y_scale) Bs7(y_scale) Bs8(y_scale) Bs9(y_scale)
##    -3.552334    -3.543252    -3.539085    -2.253324
```

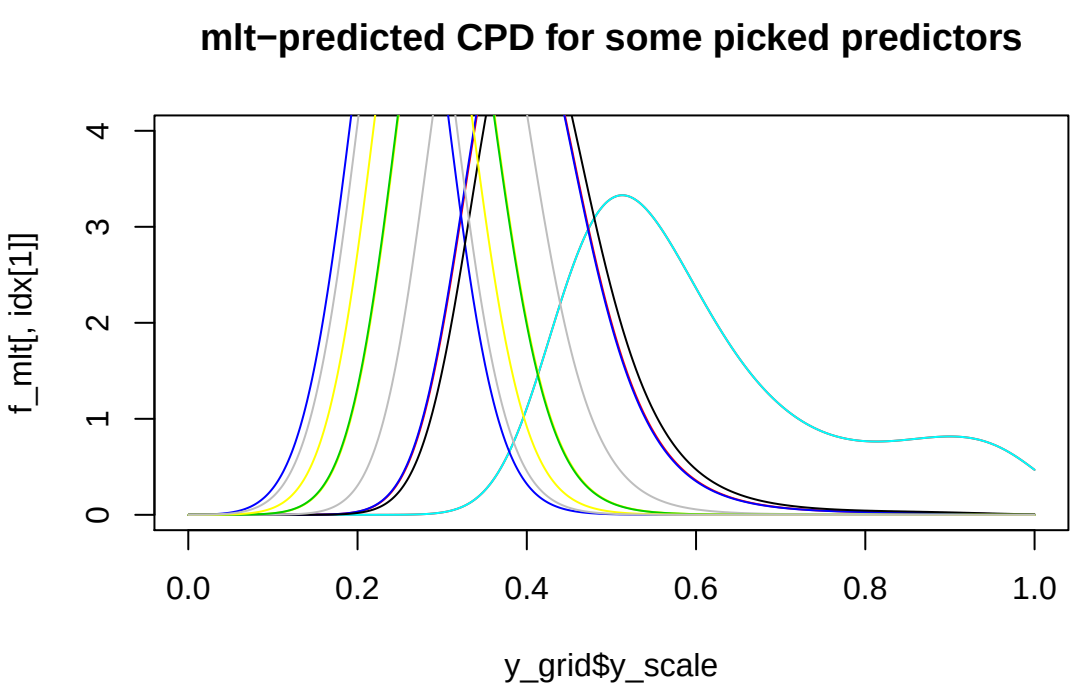
```
( beta = mlt_fit$coef[(nb+2):length(mlt_fit$coef)] )
```

```
##         crim           zn        indus         chas          nox
##  0.044629187 -0.006412394 -0.010904406 -0.583180972  4.724995436
##           rm          age          dis          rad          tax
## -0.467408838  0.002101093  0.293249769 -0.079827513  0.003507470
##      ptratio            b        lstat
##  0.225942673 -0.002572815  0.161517358
```

The conditional PDF for some observations

```
  f_mlt = predict(mlt_fit, newdata=dat,  q=y_grid$y_scale, type='density')

  q_mlt = predict(mlt_fit, newdata=dat,
                  prob=c(0.025,0.25,0.5, 0.75,0.975), type='quantile')
  q_mlt = t(q_mlt)
  #q_mlt = matrix(q_mlt$exact, ncol = 5, byrow = TRUE)
  set.seed(3)
  idx = sample(1:ncol(f_mlt))[1:10]
  m = max(f_mlt[,idx])
  plot(y_grid$y_scale, f_mlt[,idx[1]], type='l',col='red', ylim=c(0,4),
       main="mlt-predicted CPD for some picked predictors")
  for (i in idx){
    lines(y_grid$y_scale, f_mlt[,i], col=i)
  }
```


mlt−predicted CPD for some picked predictors
f_mlt[, idx[1]]
0
1
2
3
4
0.0
0.2
0.4
0.6
0.8
1.0
y_grid$y_scale

NN

NN approach for a linear shift model, modeled with NN

Fitting means to find the *nb* coefficients *theta* for the Bernsteinpolynom which approximaties the transformation function with *nb* being set to:

```
nb
```

```
## [1] 8
```

Preparing input and output

```
y = tf$Variable(as.matrix(dat$y_scale)[,drop=FALSE], dtype='float32')
y$shape # has to be (#y,1)
```

```
## (506, 1)
```

```
# conditional - we give the rm-variables as input to the NN
#x = tf$Variable(as.matrix(dat$rm)[,drop=FALSE], dtype='float32')

#x = tf$Variable(as.matrix(dat[,c('rm','lstat'),drop=FALSE]), dtype='float32')

#dat$chas = as.numeric(as.character(dat$chas))
x = tf$Variable(x, dtype='float32') #all
x$shape  # has to be (#y,1) for a univariate model
```

```
## (506, 13)
```

```
source('model_3.R')
source('bern_utils.R')
source("model_utils.R")
x_dim = as.integer(dim(x)[2])
model_3 = new_model_3(len_theta = as.integer(len_theta), x_dim = x_dim, y_range=y_range)
```

```
## Error in on_load() :
##   TensorFlow Probability has to be used with the TensorFlow Keras implementation.
```

```
T_OUT = 100
run = 1
history = model_train(model_3, make_hist(), x_train = x, y_train = y,
                      x_test = x, y_test = y, T_STEPS=15000)
```

```
## [1] "100 model_3: likelihood (in optimize)  9.13162422180176 likelihood (in test)  9.12455940246582"
## [1] "200 model_3: likelihood (in optimize)  8.47586822509766 likelihood (in test)  8.46978950500488"
## [1] "300 model_3: likelihood (in optimize)  7.90890216827393 likelihood (in test)  7.90362119674683"
## [1] "400 model_3: likelihood (in optimize)  7.41430187225342 likelihood (in test)  7.40967702865601"
## [1] "500 model_3: likelihood (in optimize)  6.97982311248779 likelihood (in test)  6.97574901580811"
## [1] "600 model_3: likelihood (in optimize)  6.60198593139648 likelihood (in test)  6.5984845161438"
## [1] "700 model_3: likelihood (in optimize)  6.27120971679688 likelihood (in test)  6.26809215545654"
## [1] "800 model_3: likelihood (in optimize)  5.97671508789062 likelihood (in test)  5.97393989562988"
## [1] "900 model_3: likelihood (in optimize)  5.71455430984497 likelihood (in test)  5.71208381652832"
## [1] "1000 model_3: likelihood (in optimize)  5.4813289642334 likelihood (in test)  5.47913217544556"
## [1] "1100 model_3: likelihood (in optimize)  5.27407360076904 likelihood (in test)  5.27212285995483"
## [1] "1200 model_3: likelihood (in optimize)  5.09016704559326 likelihood (in test)  5.0884370803833"
## [1] "1300 model_3: likelihood (in optimize)  4.92725944519043 likelihood (in test)  4.9257287979126"
```

```
## [1] "1400 model_3: likelihood (in optimize)  4.78322505950928 likelihood (in test)  4.78187322616577
## [1] "1500 model_3: likelihood (in optimize)  4.65612506866455 likelihood (in test)  4.6549334526062"
## [1] "1600 model_3: likelihood (in optimize)  4.5441780090332 likelihood (in test)  4.54312944412231"
## [1] "1700 model_3: likelihood (in optimize)  4.44573640823364 likelihood (in test)  4.44481515884399
## [1] "1800 model_3: likelihood (in optimize)  4.35927486419678 likelihood (in test)  4.35846567153931
## [1] "1900 model_3: likelihood (in optimize)  4.28337907791138 likelihood (in test)  4.28266906738281
## [1] "2000 model_3: likelihood (in optimize)  4.21674108505249 likelihood (in test)  4.21611738204956
## [1] "2100 model_3: likelihood (in optimize)  4.15815258026123 likelihood (in test)  4.15760374069214
## [1] "2200 model_3: likelihood (in optimize)  4.10650539398193 likelihood (in test)  4.10602045059204
## [1] "2300 model_3: likelihood (in optimize)  4.06078815460205 likelihood (in test)  4.06035804748535
## [1] "2400 model_3: likelihood (in optimize)  4.02009057998657 likelihood (in test)  4.01970624923706
## [1] "2500 model_3: likelihood (in optimize)  3.9835991859436 likelihood (in test)  3.98325300216675"
## [1] "2600 model_3: likelihood (in optimize)  3.95059728622437 likelihood (in test)  3.95028257369995
## [1] "2700 model_3: likelihood (in optimize)  3.92046308517456 likelihood (in test)  3.92017436027527
## [1] "2800 model_3: likelihood (in optimize)  3.89266538619995 likelihood (in test)  3.89239764213562
## [1] "2900 model_3: likelihood (in optimize)  3.8667573928833 likelihood (in test)  3.86650657653809"
## [1] "3000 model_3: likelihood (in optimize)  3.84236931800842 likelihood (in test)  3.84213209152222
## [1] "3100 model_3: likelihood (in optimize)  3.81919741630554 likelihood (in test)  3.81897115707397
## [1] "3200 model_3: likelihood (in optimize)  3.79706025123596 likelihood (in test)  3.79684352874756
## [1] "3300 model_3: likelihood (in optimize)  3.77574634552002 likelihood (in test)  3.77553653717041
## [1] "3400 model_3: likelihood (in optimize)  3.75504922866821 likelihood (in test)  3.75484490394592
## [1] "3500 model_3: likelihood (in optimize)  3.73484039306641 likelihood (in test)  3.73464059829712
## [1] "3600 model_3: likelihood (in optimize)  3.71501755714417 likelihood (in test)  3.71482086181641
## [1] "3700 model_3: likelihood (in optimize)  3.69549655914307 likelihood (in test)  3.69530248641968
## [1] "3800 model_3: likelihood (in optimize)  3.67620992660522 likelihood (in test)  3.67601823806763
## [1] "3900 model_3: likelihood (in optimize)  3.6571033000946 likelihood (in test)  3.65691304206848"
## [1] "4000 model_3: likelihood (in optimize)  3.63813281059265 likelihood (in test)  3.63794374465942
## [1] "4100 model_3: likelihood (in optimize)  3.61926412582397 likelihood (in test)  3.61907601356506
## [1] "4200 model_3: likelihood (in optimize)  3.60047078132629 likelihood (in test)  3.60028314590454
## [1] "4300 model_3: likelihood (in optimize)  3.58173251152039 likelihood (in test)  3.58154535293579
## [1] "4400 model_3: likelihood (in optimize)  3.56303548812866 likelihood (in test)  3.56284856796265
## [1] "4500 model_3: likelihood (in optimize)  3.54437017440796 likelihood (in test)  3.5441837310791"
## [1] "4600 model_3: likelihood (in optimize)  3.52573156356812 likelihood (in test)  3.52554535865784
## [1] "4700 model_3: likelihood (in optimize)  3.50711679458618 likelihood (in test)  3.50693082809448
## [1] "4800 model_3: likelihood (in optimize)  3.4885265827179 likelihood (in test)  3.48834085464478"
## [1] "4900 model_3: likelihood (in optimize)  3.46996307373047 likelihood (in test)  3.46977758407593
## [1] "5000 model_3: likelihood (in optimize)  3.45143032073975 likelihood (in test)  3.45124506950378
## [1] "5100 model_3: likelihood (in optimize)  3.43293285369873 likelihood (in test)  3.43274807929993
## [1] "5200 model_3: likelihood (in optimize)  3.41447687149048 likelihood (in test)  3.41429257392883
## [1] "5300 model_3: likelihood (in optimize)  3.39611577987671 likelihood (in test)  3.39593815803528
## [1] "5400 model_3: likelihood (in optimize)  3.37836313247681 likelihood (in test)  3.37818574905396
## [1] "5500 model_3: likelihood (in optimize)  3.36065149307251 likelihood (in test)  3.3604748249054"
## [1] "5600 model_3: likelihood (in optimize)  3.34299182891846 likelihood (in test)  3.34281539916992
## [1] "5700 model_3: likelihood (in optimize)  3.32539486885071 likelihood (in test)  3.32521915435791
## [1] "5800 model_3: likelihood (in optimize)  3.30787253379822 likelihood (in test)  3.30769777297974
## [1] "5900 model_3: likelihood (in optimize)  3.29043626785278 likelihood (in test)  3.29026222229004
## [1] "6000 model_3: likelihood (in optimize)  3.27309823036194 likelihood (in test)  3.27292537689209
## [1] "6100 model_3: likelihood (in optimize)  3.25587058067322 likelihood (in test)  3.25569891929626
## [1] "6200 model_3: likelihood (in optimize)  3.23876595497131 likelihood (in test)  3.23859548568726
## [1] "6300 model_3: likelihood (in optimize)  3.22179698944092 likelihood (in test)  3.22162818908691
## [1] "6400 model_3: likelihood (in optimize)  3.20497751235962 likelihood (in test)  3.20481014251709
## [1] "6500 model_3: likelihood (in optimize)  3.18832159042358 likelihood (in test)  3.18815612792969
## [1] "6600 model_3: likelihood (in optimize)  3.17184376716614 likelihood (in test)  3.17167997360229
## [1] "6700 model_3: likelihood (in optimize)  3.15555953979492 likelihood (in test)  3.15539765357971
```

```
## [1] "6800 model_3: likelihood (in optimize)  3.13948512077332 likelihood (in test)  3.13932538032532
## [1] "6900 model_3: likelihood (in optimize)  3.12363743782043 likelihood (in test)  3.12348008155823
## [1] "7000 model_3: likelihood (in optimize)  3.10803365707397 likelihood (in test)  3.10787892341614
## [1] "7100 model_3: likelihood (in optimize)  3.09269165992737 likelihood (in test)  3.09253978729248
## [1] "7200 model_3: likelihood (in optimize)  3.07762932777405 likelihood (in test)  3.07748007774353
## [1] "7300 model_3: likelihood (in optimize)  3.06286358833313 likelihood (in test)  3.06271743774414
## [1] "7400 model_3: likelihood (in optimize)  3.04841089248657 likelihood (in test)  3.04826807975769
## [1] "7500 model_3: likelihood (in optimize)  3.03428649902344 likelihood (in test)  3.03414678573608
## [1] "7600 model_3: likelihood (in optimize)  3.02050423622131 likelihood (in test)  3.02036809921265
## [1] "7700 model_3: likelihood (in optimize)  3.00707602500916 likelihood (in test)  3.00694370269775
## [1] "7800 model_3: likelihood (in optimize)  2.99401259422302 likelihood (in test)  2.99388384819031
## [1] "7900 model_3: likelihood (in optimize)  2.98132300376892 likelihood (in test)  2.98119807243347
## [1] "8000 model_3: likelihood (in optimize)  2.96901512145996 likelihood (in test)  2.96889400482178
## [1] "8100 model_3: likelihood (in optimize)  2.95709538459778 likelihood (in test)  2.95697832107544
## [1] "8200 model_3: likelihood (in optimize)  2.94556951522827 likelihood (in test)  2.9454562664032"
## [1] "8300 model_3: likelihood (in optimize)  2.93444132804871 likelihood (in test)  2.93433237075806
## [1] "8400 model_3: likelihood (in optimize)  2.92371463775635 likelihood (in test)  2.92360949516296
## [1] "8500 model_3: likelihood (in optimize)  2.91339087486267 likelihood (in test)  2.91328954696655
## [1] "8600 model_3: likelihood (in optimize)  2.9034698009491 likelihood (in test)  2.9033727645874"
## [1] "8700 model_3: likelihood (in optimize)  2.89395070075989 likelihood (in test)  2.89385747909546
## [1] "8800 model_3: likelihood (in optimize)  2.88482999801636 likelihood (in test)  2.88474082946777
## [1] "8900 model_3: likelihood (in optimize)  2.87610340118408 likelihood (in test)  2.87601804733276
## [1] "9000 model_3: likelihood (in optimize)  2.86776471138 likelihood (in test)  2.86768317222595"
## [1] "9100 model_3: likelihood (in optimize)  2.85980558395386 likelihood (in test)  2.85972833633423
## [1] "9200 model_3: likelihood (in optimize)  2.85221791267395 likelihood (in test)  2.85214400291443
## [1] "9300 model_3: likelihood (in optimize)  2.84499096870422 likelihood (in test)  2.84492039680481
## [1] "9400 model_3: likelihood (in optimize)  2.83811330795288 likelihood (in test)  2.83804607391357
## [1] "9500 model_3: likelihood (in optimize)  2.83157348632812 likelihood (in test)  2.83150959014893
## [1] "9600 model_3: likelihood (in optimize)  2.82535934448242 likelihood (in test)  2.82529854774475
## [1] "9700 model_3: likelihood (in optimize)  2.81945729255676 likelihood (in test)  2.81940007209778
## [1] "9800 model_3: likelihood (in optimize)  2.81385564804077 likelihood (in test)  2.81380081176758
## [1] "9900 model_3: likelihood (in optimize)  2.80854034423828 likelihood (in test)  2.80848860740662
## [1] "10000 model_3: likelihood (in optimize)  2.80349922180176 likelihood (in test)  2.8034501075744
## [1] "10100 model_3: likelihood (in optimize)  2.79871869087219 likelihood (in test)  2.7986721992492
## [1] "10200 model_3: likelihood (in optimize)  2.79418683052063 likelihood (in test)  2.7941427230835
## [1] "10300 model_3: likelihood (in optimize)  2.7898907661438 likelihood (in test)  2.78984880447388
## [1] "10400 model_3: likelihood (in optimize)  2.78581857681274 likelihood (in test)  2.7857789932861
## [1] "10500 model_3: likelihood (in optimize)  2.78195810317993 likelihood (in test)  2.7819206714630
## [1] "10600 model_3: likelihood (in optimize)  2.77829837799072 likelihood (in test)  2.7782626152038
## [1] "10700 model_3: likelihood (in optimize)  2.77482748031616 likelihood (in test)  2.7747936248779
## [1] "10800 model_3: likelihood (in optimize)  2.77153444290161 likelihood (in test)  2.7715024948120
## [1] "10900 model_3: likelihood (in optimize)  2.76840877532959 likelihood (in test)  2.7683784961700
## [1] "11000 model_3: likelihood (in optimize)  2.76544046401978 likelihood (in test)  2.7654113769531
## [1] "11100 model_3: likelihood (in optimize)  2.76261901855469 likelihood (in test)  2.7625916004180
## [1] "11200 model_3: likelihood (in optimize)  2.75993514060974 likelihood (in test)  2.7599089145660
## [1] "11300 model_3: likelihood (in optimize)  2.75737953186035 likelihood (in test)  2.7573547363281
## [1] "11400 model_3: likelihood (in optimize)  2.75494337081909 likelihood (in test)  2.7549195289611
## [1] "11500 model_3: likelihood (in optimize)  2.75261831283569 likelihood (in test)  2.7525956630706
## [1] "11600 model_3: likelihood (in optimize)  2.75039625167847 likelihood (in test)  2.7503747940063
## [1] "11700 model_3: likelihood (in optimize)  2.74827003479004 likelihood (in test)  2.7482490539550
## [1] "11800 model_3: likelihood (in optimize)  2.74623203277588 likelihood (in test)  2.7462120056152
## [1] "11900 model_3: likelihood (in optimize)  2.74427604675293 likelihood (in test)  2.7442567348480
## [1] "12000 model_3: likelihood (in optimize)  2.74239587783813 likelihood (in test)  2.7423775196075
## [1] "12100 model_3: likelihood (in optimize)  2.7405858039856 likelihood (in test)  2.74056816101074
```

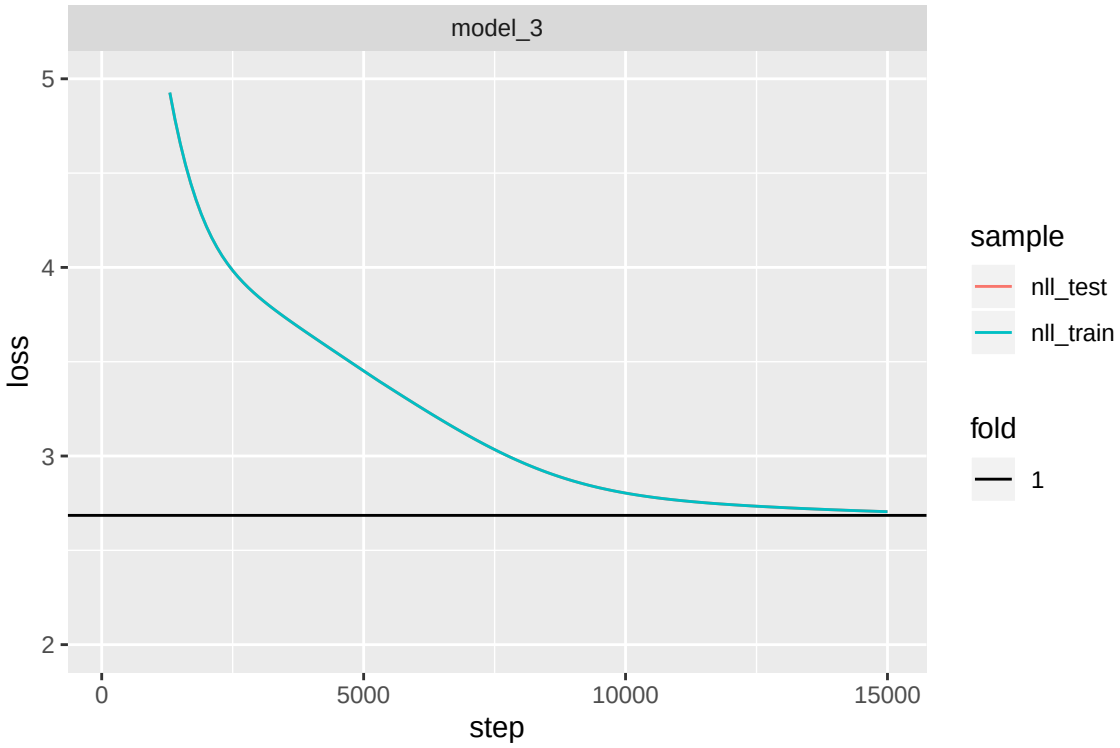
```
## [1] "12200 model_3: likelihood (in optimize)  2.73884105682373 likelihood (in test)  2.7388238906860
## [1] "12300 model_3: likelihood (in optimize)  2.73715615272522 likelihood (in test)  2.7371397018432
## [1] "12400 model_3: likelihood (in optimize)  2.7355272769928 likelihood (in test)  2.735511302948"
## [1] "12500 model_3: likelihood (in optimize)  2.73394989967346 likelihood (in test)  2.7339346408844
## [1] "12600 model_3: likelihood (in optimize)  2.73242092132568 likelihood (in test)  2.7324056625366
## [1] "12700 model_3: likelihood (in optimize)  2.7309365272522 likelihood (in test)  2.73092198371887
## [1] "12800 model_3: likelihood (in optimize)  2.72949361801147 likelihood (in test)  2.7294793128967
## [1] "12900 model_3: likelihood (in optimize)  2.7280900478363 likelihood (in test)  2.72807598114014
## [1] "13000 model_3: likelihood (in optimize)  2.72672271728516 likelihood (in test)  2.7267088890075
## [1] "13100 model_3: likelihood (in optimize)  2.72538900375366 likelihood (in test)  2.7253756523132
## [1] "13200 model_3: likelihood (in optimize)  2.72408723831177 likelihood (in test)  2.7240743637085
## [1] "13300 model_3: likelihood (in optimize)  2.722815990448 likelihood (in test)  2.72280311584473"
## [1] "13400 model_3: likelihood (in optimize)  2.7215723991394 likelihood (in test)  2.72156000137329
## [1] "13500 model_3: likelihood (in optimize)  2.72035598754883 likelihood (in test)  2.7203440666198
## [1] "13600 model_3: likelihood (in optimize)  2.71916532516479 likelihood (in test)  2.719153881073"
## [1] "13700 model_3: likelihood (in optimize)  2.71799921989441 likelihood (in test)  2.7179877758026
## [1] "13800 model_3: likelihood (in optimize)  2.71685695648193 likelihood (in test)  2.7168455123901
## [1] "13900 model_3: likelihood (in optimize)  2.71573781967163 likelihood (in test)  2.7157268524169
## [1] "14000 model_3: likelihood (in optimize)  2.71464109420776 likelihood (in test)  2.7146303653717
## [1] "14100 model_3: likelihood (in optimize)  2.71356678009033 likelihood (in test)  2.7135562896728
## [1] "14200 model_3: likelihood (in optimize)  2.71251440048218 likelihood (in test)  2.7125039100647
## [1] "14300 model_3: likelihood (in optimize)  2.7114839553833 likelihood (in test)  2.71147394180298
## [1] "14400 model_3: likelihood (in optimize)  2.7104754447937 likelihood (in test)  2.71046543121338
## [1] "14500 model_3: likelihood (in optimize)  2.70948839187622 likelihood (in test)  2.7094788551330
## [1] "14600 model_3: likelihood (in optimize)  2.70852422714233 likelihood (in test)  2.7085146903991
## [1] "14700 model_3: likelihood (in optimize)  2.70758152008057 likelihood (in test)  2.7075719833374
## [1] "14800 model_3: likelihood (in optimize)  2.70666074752808 likelihood (in test)  2.7066516876220
## [1] "14900 model_3: likelihood (in optimize)  2.70576238632202 likelihood (in test)  2.7057535648345
## [1] "15000 model_3: likelihood (in optimize)  2.70488619804382 likelihood (in test)  2.7048776149749
```

```
history$step = as.integer(history$step)
history$fold = as.integer(history$fold)
history$nll_train = as.numeric(history$nll_train)
history$nll_test = as.numeric(history$nll_test)
history$OK = NULL# = as.numeric(history$OK)

library(tidyr)
h = gather(history, 'sample', 'loss', nll_train:nll_test)
h$loss = as.numeric(h$loss)
h$sample = as.factor(h$sample)
h$fold = as.factor(h$fold)
hh =h[!is.na(h$loss),]

ggplot(hh, aes(x=step,y=loss, color=sample, linetype=fold)) +
ylim(2,5) + geom_hline(yintercept=NLL_MLT)+  geom_line() + facet_grid(. ~ method)
```

```
## Warning: Removed 24 rows containing missing values (geom_path).
```



Compare NN model to MLT model

Get beta coefficients

```
( beta_nn = as.numeric( model_3$model_beta$get_weights()[[1]]) )
```

```
##  [1] -0.3802913  0.1378509  0.0367732  0.1499210 -0.5380754  0.4655110
##  [7] -0.1609397 -0.6141492  0.6341144 -0.5436301 -0.4742308  0.2526142
## [13] -0.7920753
```

```
 ( beta_mlt = mlt_fit$coef[(len_theta+1):(len_theta+ncol(x))] )
```

```
##         crim           zn        indus         chas          nox
##  0.044629187 -0.006412394 -0.010904406 -0.583180972  4.724995436
##           rm          age          dis          rad          tax
## -0.467408838  0.002101093  0.293249769 -0.079827513  0.003507470
##      ptratio            b        lstat
##  0.225942673 -0.002572815  0.161517358
```

```
  beta_nn/beta_mlt
```

```
##         crim           zn        indus         chas          nox
##   -8.5211333  -21.4975638   -3.3723255   -0.2570746   -0.1138785
##           rm          age          dis          rad          tax
##   -0.9959396  -76.5980852   -2.0942869   -7.9435575 -154.9920947
##      ptratio            b        lstat
##   -2.0988986  -98.1859115   -4.9039642
```

Get theta coefficients

```
  one = tf$ones(shape = c(1,1))
  ( theta_nn = to_theta(model_3$model_hy(one)) )
```

```
## tf.Tensor(
## [[-5.531165  -3.3423092 -2.0974975  0.7311082  3.006028   3.1547186
##    3.1911764  3.2068603  4.420996 ]], shape=(1, 9), dtype=float32)
```

```
  ( theta_mlt = mlt_fit$coef[1:len_theta] )
```

```
## Bs1(y_scale) Bs2(y_scale) Bs3(y_scale) Bs4(y_scale) Bs5(y_scale)
##   -12.949149    -9.935969    -9.926925    -4.953970    -4.099484
## Bs6(y_scale) Bs7(y_scale) Bs8(y_scale) Bs9(y_scale)
##    -3.552334    -3.543252    -3.539085    -2.253324
```

```
  theta_nn$numpy()/theta_mlt
```

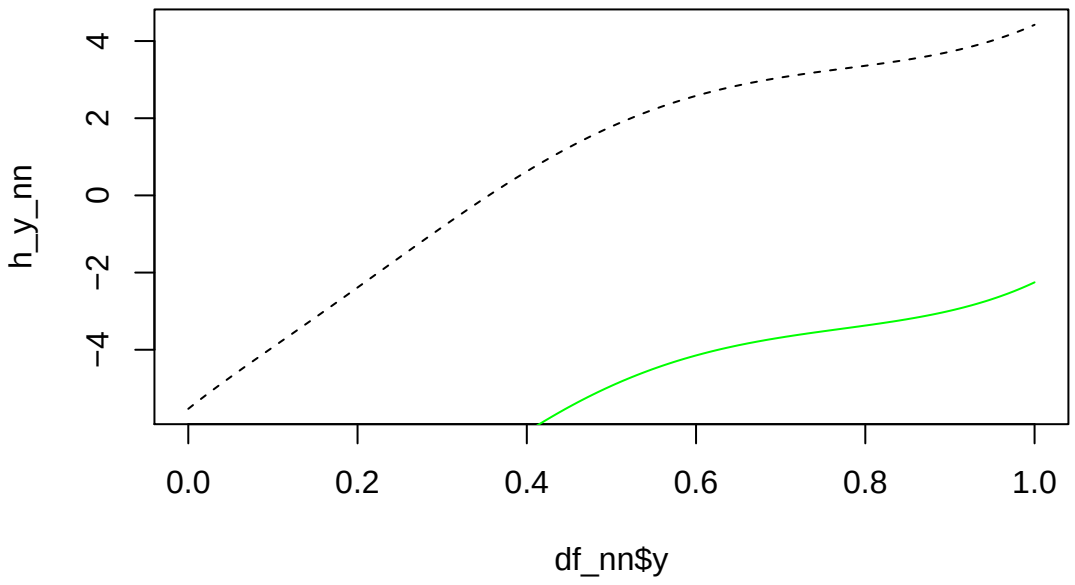
```
##           [,1]      [,2]      [,3]       [,4]       [,5]       [,6]
## [1,] 0.4271451 0.3363848 0.2112938 -0.1475803 -0.7332698 -0.8880694
##            [,7]      [,8]      [,9]
## [1,] -0.9006349 -0.906127 -1.961989
```

predict baseline trafo (first part of trafo w/o shift)

```
#nn
out_row = model_3$model_hy(one) #Pick row and compute CPD
df_nn = bernp.p_y_h(model_3$bernp, out_row, from = 0, to = 1, length.out = length(y_grid$y_scale))
h_y_nn = df_nn$h

# mlt
h_y_mlt = predict(bb, newdata = y_grid, coef = theta_mlt, type='trafo')

plot(df_nn$y, h_y_nn,  type='l', lty=2)
lines(y_grid$y_scale, h_y_mlt, type='l',col='green')
```

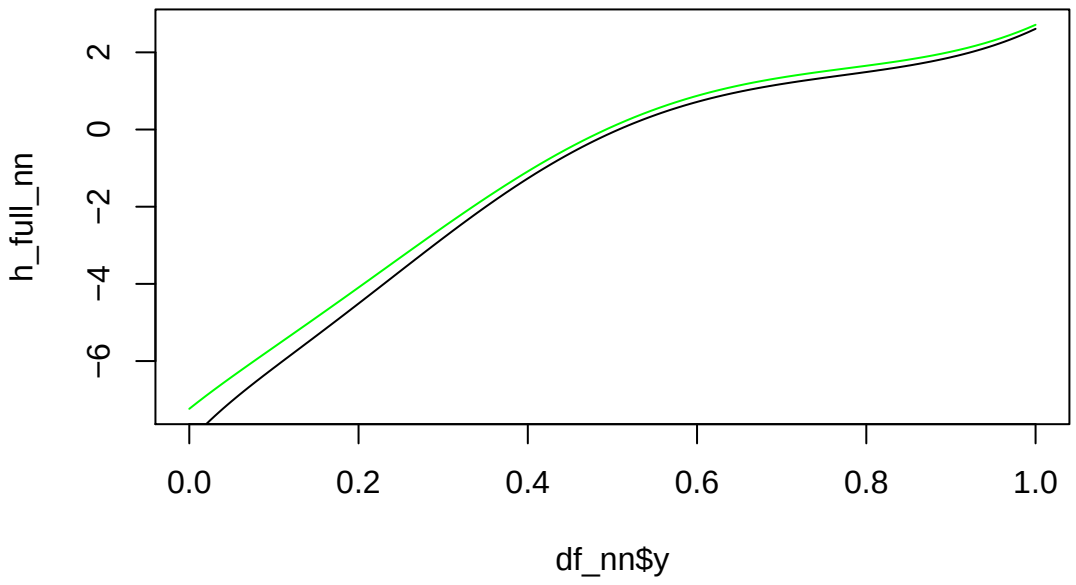


Predict full trafo (all parts of trafo inclusive shift) for picked observation

```
# nn
pick_idx = 1
shift = beta_nn %*% x[pick_idx,1:length(beta_nn)]$numpy()
out_row = model_3$model_hy(one) #Pick row and compute CPD
df_nn = bernp.p_y_h(model_3$bernp, out_row, from = 0, to = 1, length.out = 100, out_eta = shift)
h_full_nn = df_nn$h

# mlt:
h_full_mlt = predict(mlt_fit, newdata = dat[pick_idx,], q=y_grid$y_scale, type='trafo')

plot(df_nn$y, h_full_nn, type='l', col='green')
lines(y_grid$y_scale, h_full_mlt)
```



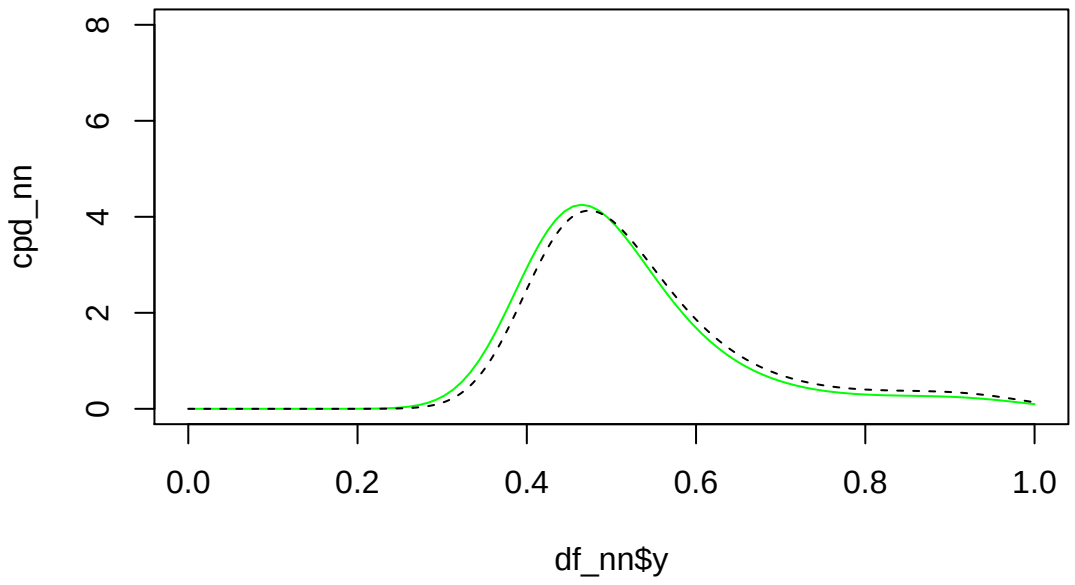
Predict CPD for picked observations

```
# NN

cpd_nn = df_nn$p_y

# mlt:
cpd_mlt = predict(mlt_fit, newdata = dat[pick_idx,], q=y_grid$y_scale, type='density')


plot(df_nn$y, cpd_nn, ylim=c(0,8), type="l", col='green')
lines(y_grid$y_scale, cpd_mlt, lty=2)
```


cpd_nn
0
2
4
6
8
0.0
0.2
0.4
0.6
0.8
1.0
df_nn$y