Beta Distributions

Fitting Bernsteinpolynomials

First importing custom functions

```
source('bern_utils.R')
```

Imports

```
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)
tf_version()
```

```
## [1] '2.0'
```

```
tf_probability()
```

```
## Module(tensorflow_probability)
```

Sourcing Utilities

```
# preparing h an h_dash, fct implemented in plain R
#source("mlt_utils.R")
# preparing eval_h an eval_h_dash, fct implemented in tfp
#source("mlt_utils_keras_v2.R")  # causes error when knittering
```

Basic idea (parameterizing the transformation function)

We parameterize the transformation function $h_Y(y)$ from the data to the noise z, which we usually assume Gaussian $z \sim N(0,1)$.

Fitting means to find the `length_theta` $=$ `M` $+$ `1` coefficients $\vartheta(x_i)$ for the Bernsteinpolynom which approximaties the transformation function $h_Y(y_i)$, where the index i is for the (mini-batch) example. The function

$h_Y(y)$ transforms from the data space into the latent space z where the data is assumed to be e.g. Gaussian distributed. We assume that $0 \leq y \leq 1$ (otherwise rescale)

$$h_Y(y) = \frac{1}{M+1} \sum_{m=0}^{M} \vartheta_m(x_i) \underbrace{f_{m+1,M-m+1}(\tilde{y})}_{f_m(\tilde{y}_i)} = \frac{1}{M+1} \sum_{m=0}^{M} \vartheta_m(x_i) f_m(y)$$

Here $f_{m+1,M-m+1}(\tilde{y})$ is the pdf of the Betadistribution. See also equation on page 12 of Most Likely Transformation paper. The basic idea of the MLT is that the parameters are controlled by the network to depend on x. Further the parameters must be ordered i.e. $\vartheta_m < \vartheta_{m+1}$. An insteresing observation is that, for all $m = 0, \ldots, M$ the sum of the parameters are $m + 1 + M - m + 1 = M + 2$. This is probably called a Bernstein basis.

Calculation of $h'(y)$

Simple example M=1

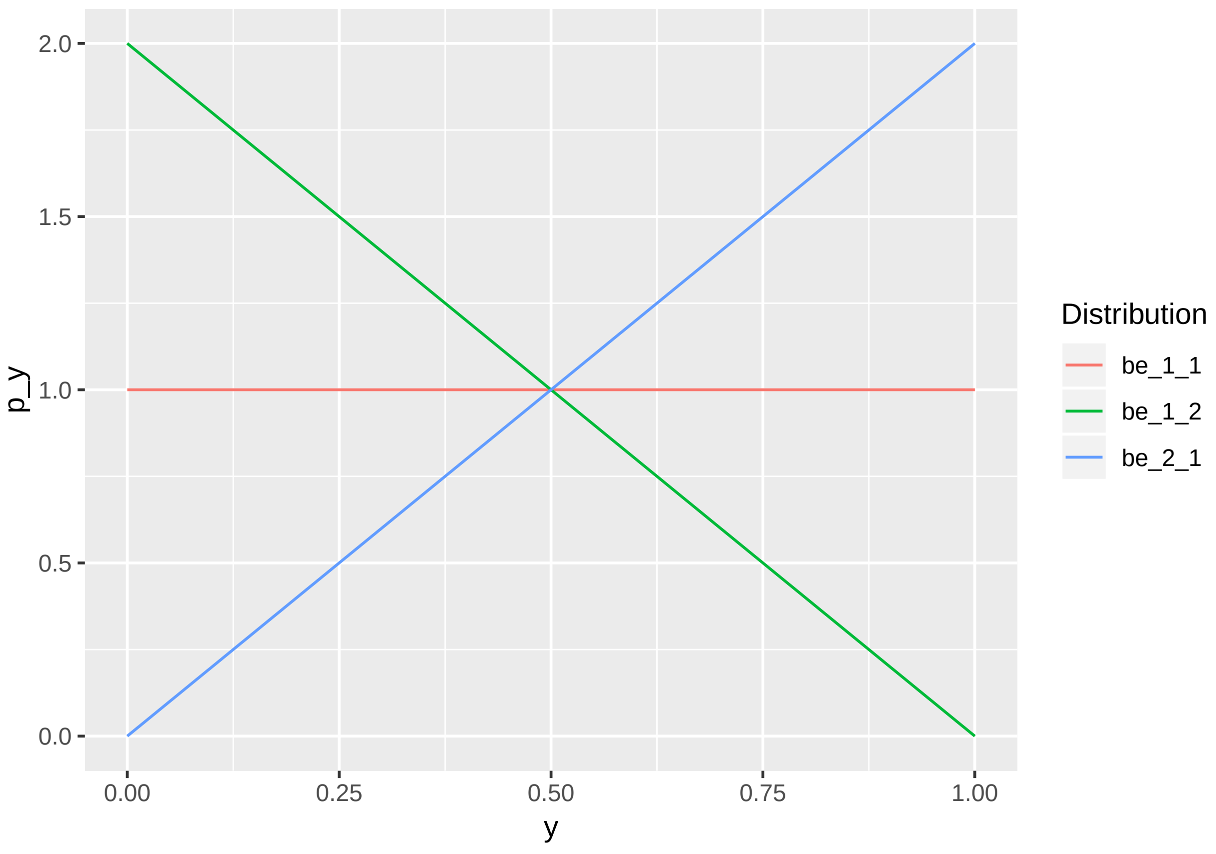
For a simple case with only two coefficients $M = 1$, we can caluclate $h_Y(y)$ by hand. Let's look at the pdfs of the Beta distributions, we assume $\vartheta_m(x) = \vartheta_m$ and choose $M = 1$.

$$h_Y(y) = \frac{1}{M+1} \sum_{m=0}^{M} \vartheta_m f_{(m+1,M-m+1)}(y) = \frac{1}{2} \left(\vartheta_0 f_{1,2}(y) + \vartheta_1 f_{2,1}(y)\right)$$

The sum of the parameters is 3 in that case.

For the derivative, we need the derivative of the pdfs of the Betadistributions $f_{1,2}(y)$ w.t.r. to y.

```
#(1*dbeta(0.5,1,2) + 4*dbeta(0.5,2,1))/2
library(ggplot2)
library(tidyr)
yy = seq(0,1,0.001)
dat = data.frame(y = yy, be_1_1 = dbeta(yy,1,1), be_2_1 = dbeta(yy,2,1), be_1_2 = dbeta(yy,1,2))
dat2 = gather(dat, "Distribution", "p_y", 2:4)
ggplot(dat2) + geom_line(aes(x=y, y=p_y, col=Distribution))
```



From the plot it's obious that:

$$\frac{\partial}{\partial y} f_{1,2}(y) = -2 = -2 f_{1,1}(y) \text{and}, \frac{\partial}{\partial y} f_{2,1}(y) = 2 = 2 f_{1,1}(y)$$

Therefore, we have for $h'(y)$ and $M = 1$

$$h'_Y(y) = \frac{1}{M+1} \left(\vartheta_0 f'_{1,2}(y) + \vartheta_1 f'_{2,1}(y)\right) = \frac{1}{M+1} (\vartheta_1 - \vartheta_0) \cdot (M+1) f_{1,1}(y)$$

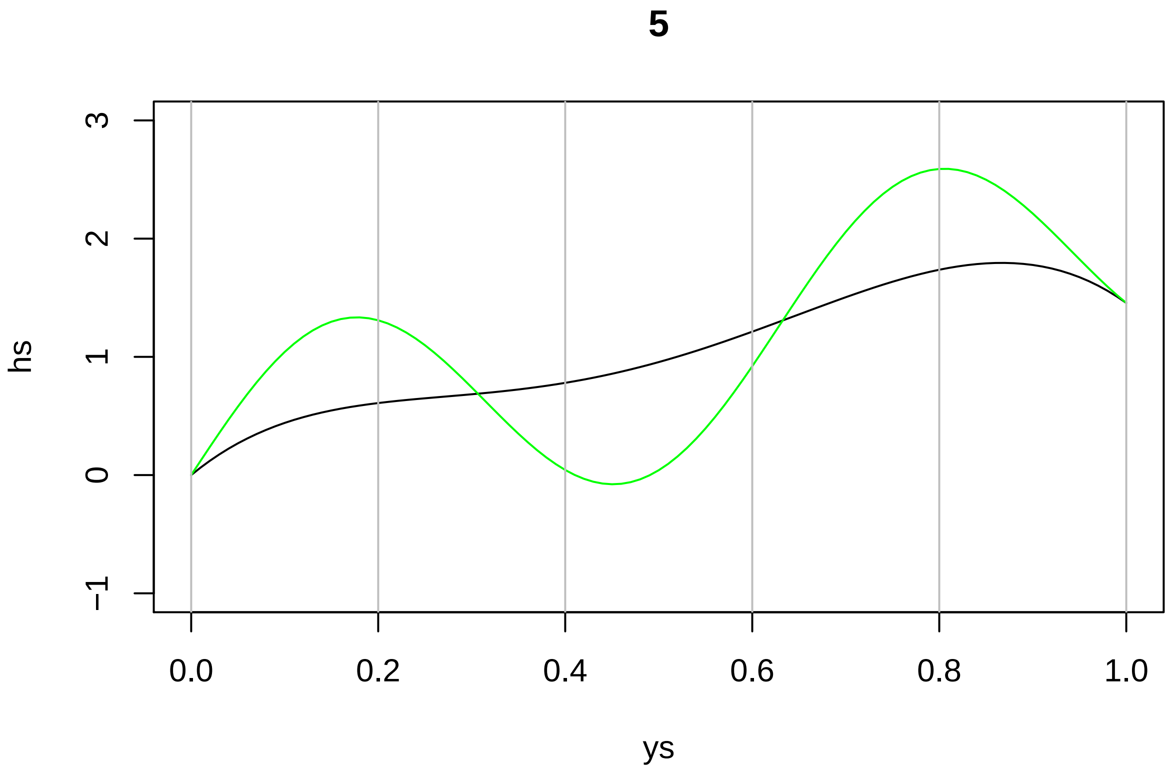
## Bernstein polynomials approximating a real-valued function Often when approximating a real-valued function $g(y)$ in the range $[0, 1]$ the following parameterization is used:

$$h(y) = \sum_{m=0}^{M} g(y = \frac{M}{m}) \binom{M}{m} y^m \cdot (1-y)^{M-m}$$

With this parameterisation $h_y \to g(y)$ for $M \to \infty$.

```
M = 5
h = function(y,g=sin){
  m = 0:M
  return(sum(
      g(m/M)
      *choose(M,m)
      *y**m*(1-y)**(M-m)
      ))
}
ys = seq(0,1,0.01)
```

```
hs = seq(0,1,0.01)
g = function(x){x*2+sin(x*10)}
for (i in 1:length(hs)) {
  hs[i] = h(ys[i],g)
}
plot(ys, hs, type='l', main = M, ylim=c(-1,3))
lines(ys,g(ys), col='green')
m = 0:M
abline(v=m/M,pch=4, col='grey')
```



Note that for finite M it's not the case that at $h(y = M/m)$ $h(y)$ is exactly $g(y)$. This seem to be the case at zero and one.

Implementation Details

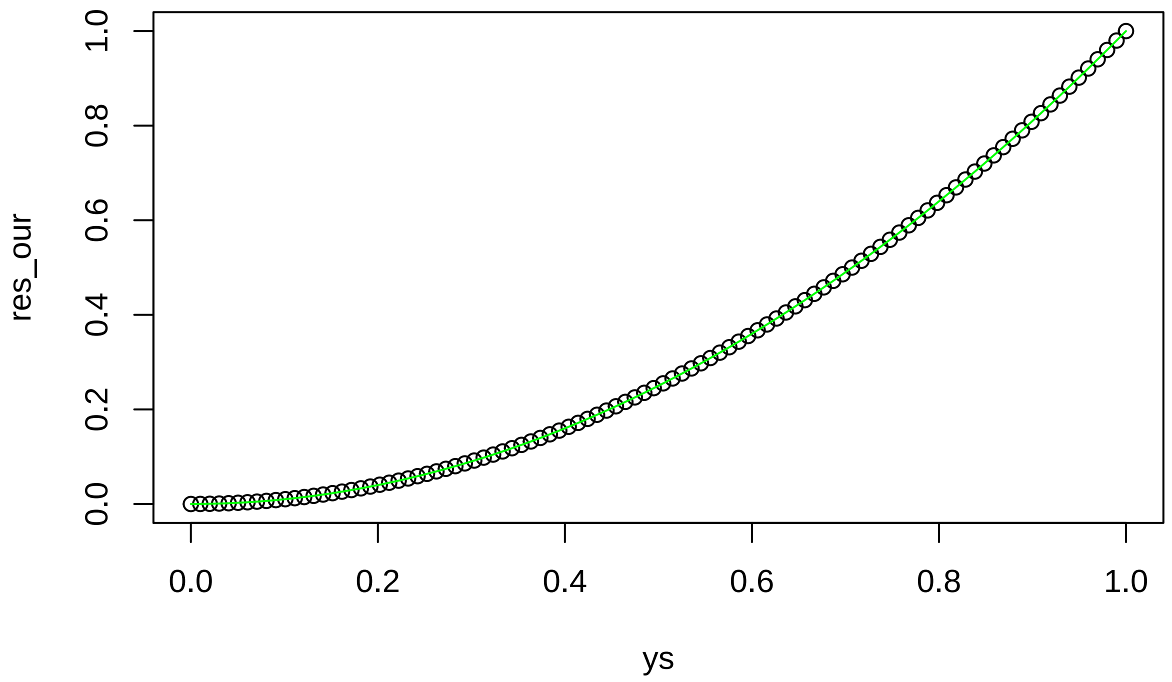
We have a look at an example $\vartheta_0 = 0, \vartheta_1 = 0, \vartheta_2 = 1$ in this case: $(M = 2)$ and the formulae taken from wikipedia

$$h_y(y) = \frac{1}{3} f_{1,3}(y) = \frac{1}{3} \frac{1}{Beta(p,q)} y^{p-1}(1-y)^{q-1} = \frac{1}{3} \frac{1}{Beta(3,1)} y^2(1-y)^0 = \frac{1}{3} 3 \cdot y^2 = y^2$$

Our Implementation:

```
try(bernp(3L))
my_bernd = bernp(3L)
ys=seq(0,1,length.out = 100)
y_cont = keras_array(matrix(ys, nrow=100,ncol=1))
res_our = as.numeric(eval_h(beta_dist_h = my_bernd$beta_dist_h,y_i= y_cont,theta_im = c(0.0,0.0,1.0)))
```

```r
plot(ys, res_our)
lines(x=ys, y=ys**2,col='green')
```

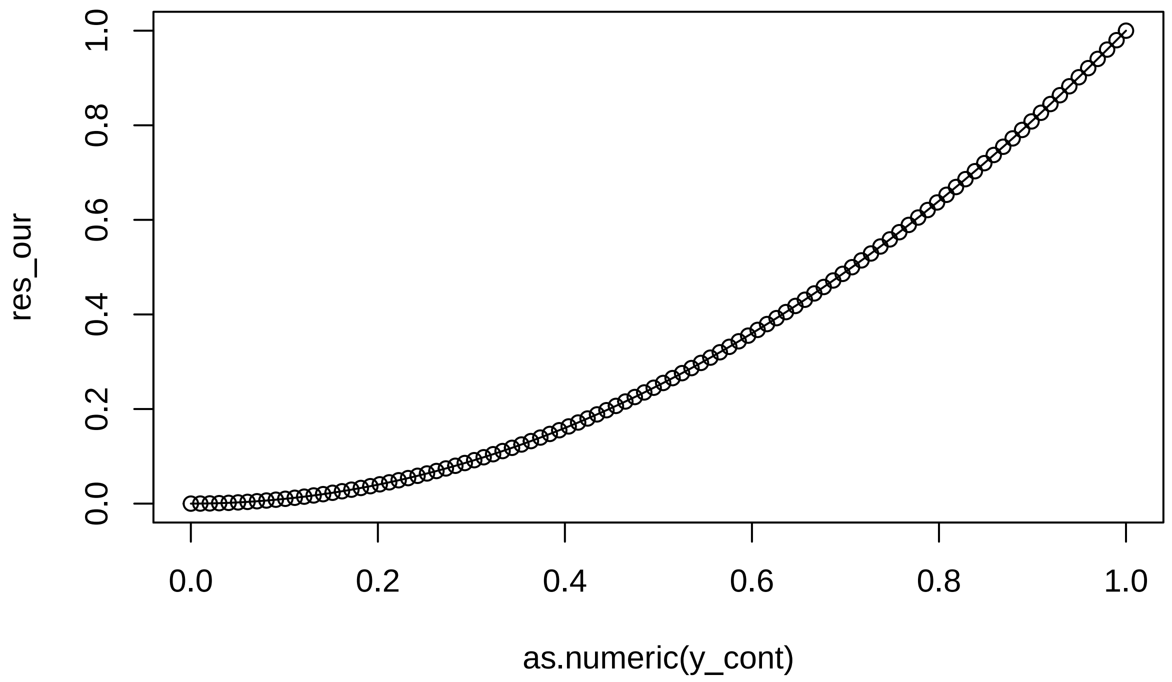


Implementation for MLT:

```r
  var_y = numeric_var("y", support = c(0, 1))
  bb <- Bernstein_basis(var_y, order=2, ui="increasing")
  y <- as.data.frame(mkgrid(bb, n = 100))
  cf = c(0,0,1)
  plot(as.numeric(y_cont), res_our, title='Our implementation (circles) vs MLT (dots)') #our implementa
```

```
## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter
```

```
lines(y$y, predict(bb, newdata = y, coef = cf))
```



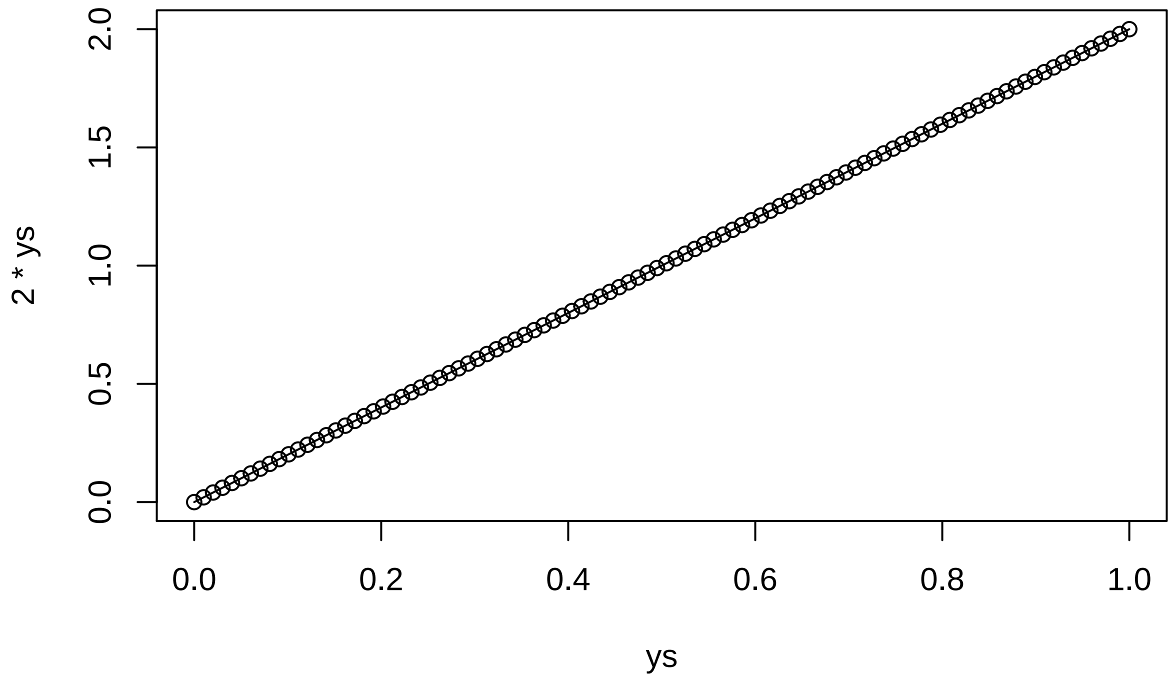
###

Implementation for MLT:

```
plot(ys, 2*ys, title='derivation analytical (lines) MLT (circles)')
```

```
## Warning in plot.window(...): "title" is not a graphical parameter

## Warning in plot.xy(xy, type, ...): "title" is not a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in axis(side = side, at = at, labels = labels, ...): "title" is not
## a graphical parameter

## Warning in box(...): "title" is not a graphical parameter

## Warning in title(...): "title" is not a graphical parameter
```

```
lines(y$y, predict(bb, newdata = y, coef = cf, deriv = c(y=1)))
```



Conclusion: It seems that we and MLT (at least the Bernstein_basis) uses the same coefficients. Also the derivations are the same.