Airbnb Case Study

Introduction

Case Study

```
url <- "https://github.com/davidruegamer/airbnb/raw/main/munich_clean_text.RDS"
destfile <- file.path(getwd(), "munich.RDS")
download.file(url, destfile, mode = "wb")
airbnb <- readRDS("munich.RDS")
airbnb$days_since_last_review <- as.numeric(difftime(airbnb$date, airbnb$last_review))
```

R Package

Core Functions

Formula Interface

```
y = log(airbnb$price)
```

```
list_of_formulas = list(
  loc = ~ 1 + te(latitude, longitude, df = 5),
  scale = ~ 1
)
```

Network Initialization

```
mod_simple <- deepregression(
  y = y,
  data = airbnb,
  list_of_formulas = list_of_formulas,
  list_of_deep_models = NULL
)
```

```
## Preparing additive formula(e)... Done.
```

```
class(mod_simple$model)
```

```
## [1] "keras.engine.functional.Functional"
## [2] "keras.engine.training.Model"
## [3] "keras.engine.base_layer.Layer"
## [4] "tensorflow.python.module.module.Module"
## [5] "tensorflow.python.training.tracking.tracking.AutoTrackable"
## [6] "tensorflow.python.training.tracking.base.Trackable"
## [7] "keras.utils.version_utils.LayerVersionSelector"
## [8] "keras.utils.version_utils.ModelVersionSelector"
## [9] "python.builtin.object"
```

Pre-processing for Structured Non-linear Layers

```
str(mod_simple$init_params$parsed_formulas_contents$loc,1)
```

```
## List of 2
##  $ :List of 12
##  $ :List of 9
```

```
sapply(mod_simple$init_params$parsed_formulas_contents$loc, "[[", "term")
```

```
## [1] "te(latitude, longitude, df = 5)"
## [2] "(Intercept)"
```

Specification of DNNs

```
deep_model <- function(x)
{
  x %>%
    layer_dense(units = 5, activation = "relu", use_bias = FALSE) %>%
    layer_dropout(rate = 0.2) %>%
    layer_dense(units = 3, activation = "relu") %>%
    layer_dropout(rate = 0.2) %>%
    layer_dense(units = 1, activation = "linear")
}
```

```
mod <- deepregression(
  y = y,
  data = airbnb,
  list_of_formulas = list(
    location = ~ 1 + beds + s(accommodates, bs = "ps") +
      s(days_since_last_review, bs = "tp") +
      deep(review_scores_rating, reviews_per_month),
    scale = ~1
  ),
  list_of_deep_models = list(deep = deep_model)
)
```

```
## Preparing additive formula(e)... Done.
```

Model Fitting and Tuning

Model Fitting

```
mod %>% fit(
  epochs = 100,
  verbose = FALSE,
  view_metrics = FALSE,
  validation_split = 0.2
  )
```

```
fitted_vals <- mod %>% fitted()
cor(fitted_vals, y)
```

```
##           [,1]
## [1,] 0.5762063
```

Model Tuning

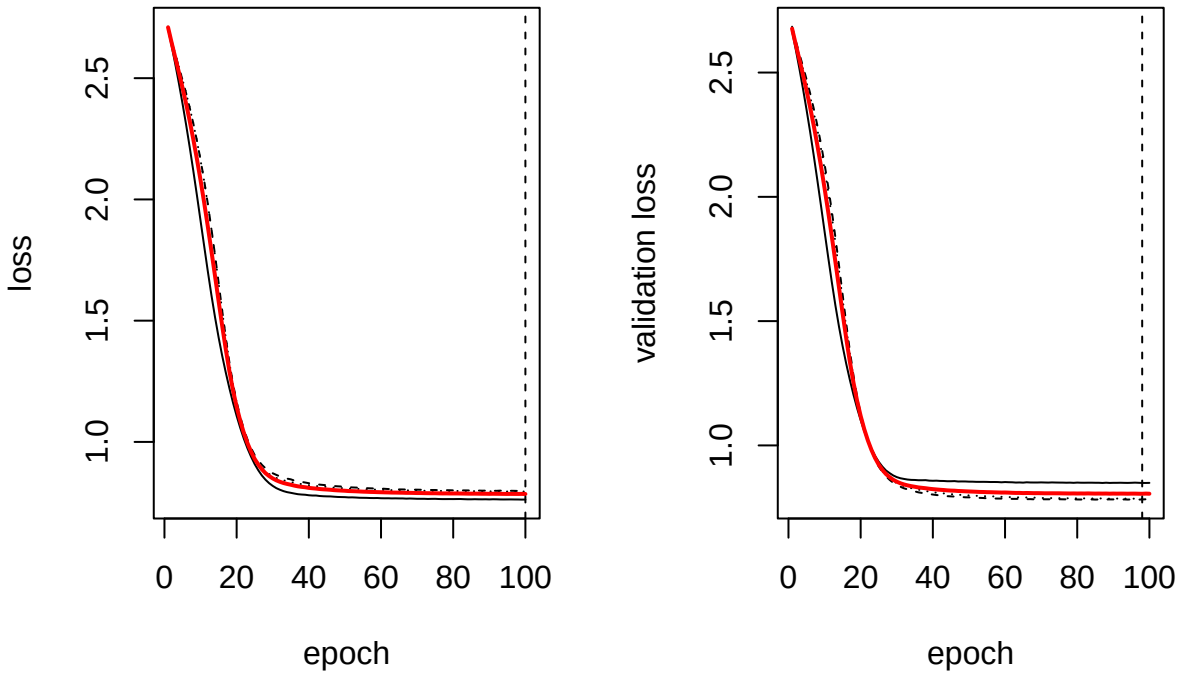
```
mod <- deepregression(
  y = y,
  data = airbnb,
  list_of_formulas = list(
    location = ~ 1 + beds + s(accommodates, bs = "ps") +
      s(days_since_last_review, bs = "tp") +
      deep(review_scores_rating, reviews_per_month),
    scale = ~1
  ),
  list_of_deep_models = list(deep = deep_model)
)
```

```
## Preparing additive formula(e)... Done.
```

```
res_cv <- mod %>% cv(
  plot = FALSE,
  cv_folds = 3,
  epochs = 100
  )
```

```
## Fitting Fold  1  ...
## Done in 11.00476  secs
## Fitting Fold  2  ...
## Done in 12.6492  secs
## Fitting Fold  3  ...
## Done in 12.14557  secs
```

```
plot_cv(res_cv)
```

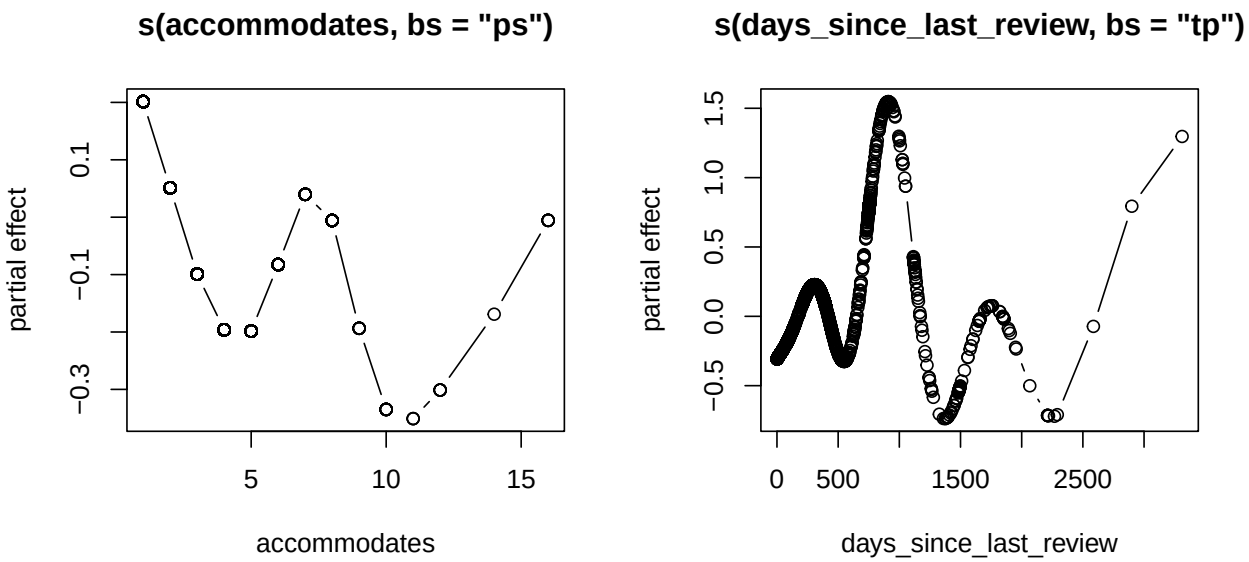


Other methods

```
coef(mod, type="linear")
```

```
## $beds
##               [,1]
##  [1,]  0.17008710
##  [2,] -0.19914058
##  [3,]  0.25847572
##  [4,] -0.15911233
##  [5,] -0.30516365
##  [6,]  0.50295949
##  [7,]  0.44622040
##  [8,]  0.28341854
##  [9,] -0.37227929
## [10,]  0.18978155
## [11,] -0.12972522
## [12,] -0.25176799
## [13,]  0.48208570
## [14,] -0.32038897
## [15,] -0.06026596
## [16,]  0.06977308
## [17,] -0.01016855
##
## $`(Intercept)`
##          [,1]
## [1,] 1.211872
```

```
par(mfrow=c(1,2))
plot(mod)
```




```
dist <- mod %>% get_distribution()
str(dist, 1)
```

```
## tfp.distributions.Normal("model_2_distribution_lambda_2_Normal", batch_shape=[3504, 1], event_shape=
```

```
first_obs_airbnb <- as.data.frame(airbnb)[1,,drop=F]
dist1 <- mod %>% get_distribution(first_obs_airbnb)
meanval <- mod %>% mean(first_obs_airbnb) %>% c
q05 <- mod %>% quant(data = first_obs_airbnb, probs = 0.05) %>% c
q95 <- mod %>% quant(data = first_obs_airbnb, probs = 0.95) %>% c
xseq <- seq(q05-1, q95+1, l=1000)
plot(xseq, sapply(xseq, function(x) c(as.matrix(dist1$prob(x)))), type="l",
     ylab = "density(price)", xlab = "price")
abline(v = c(q05, meanval, q95), col="red", lty=2)
```

Penalties

```
str(mod_simple$model$trainable_weights, 1)
```

```
## List of 3
##  $ :<tf.Variable 'te_latitude_longitude_df_5__1/kernel:0' shape=(25, 1) dtype=float32, numpy=
## array([[ 0.14152104],
##        [-0.16569498],
##        [ 0.21506482],
##        [-0.13238949],
##        [-0.25391153],
##        [ 0.4184876 ],
##        [ 0.37127787],
##        [ 0.23581845],
##        [-0.3097551 ],
##        [ 0.15790778],
##        [-0.1079379 ],
##        [-0.20948365],
##        [ 0.4011196 ],
##        [-0.26657975],
##        [-0.05014431],
##        [ 0.05805475],
##        [-0.00846076],
##        [-0.1293256 ],
##        [-0.40326533],
##        [-0.09151155],
##        [-0.21883169],
##        [ 0.2237323 ],
##        [ 0.19343877],
```

```
##         [ 0.07454377],
##         [ 0.17846191]], dtype=float32)>
##  $ :<tf.Variable '1_1/kernel:0' shape=(1, 1) dtype=float32, numpy=array([[0.01442194]], dtype=float3
##  $ :<tf.Variable '1_2/kernel:0' shape=(1, 1) dtype=float32, numpy=array([[-0.76927215]], dtype=float
```

```
lambda <- 0.5
addpen <- function(x) lambda * tf$reduce_sum(tf$abs(mod_simple$model$trainable_weights[[1]]))
mod_simple_pen <- deepregression(
  y = y,
  data = airbnb,
  list_of_formulas = list_of_formulas,
  list_of_deep_models = NULL,
  additional_penalty = addpen
)
```

```
## Preparing additive formula(e)... Done.
```

Smoothing Penalties

```
form_df_3 <- list(loc = ~ 1 + s(days_since_last_review, df = 3), scale = ~ 1)
form_df_6 <- list(loc = ~ 1 + s(days_since_last_review, df = 6), scale = ~ 1)
form_df_10 <- list(loc = ~ 1 + s(days_since_last_review, df = 10), scale = ~ 1)
args <- list(y = y, data = airbnb,
             list_of_deep_models = NULL)

mod_df_low <- do.call("deepregression", c(args, list(list_of_formulas = form_df_3)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_df_med <- do.call("deepregression", c(args, list(list_of_formulas = form_df_6)))
```

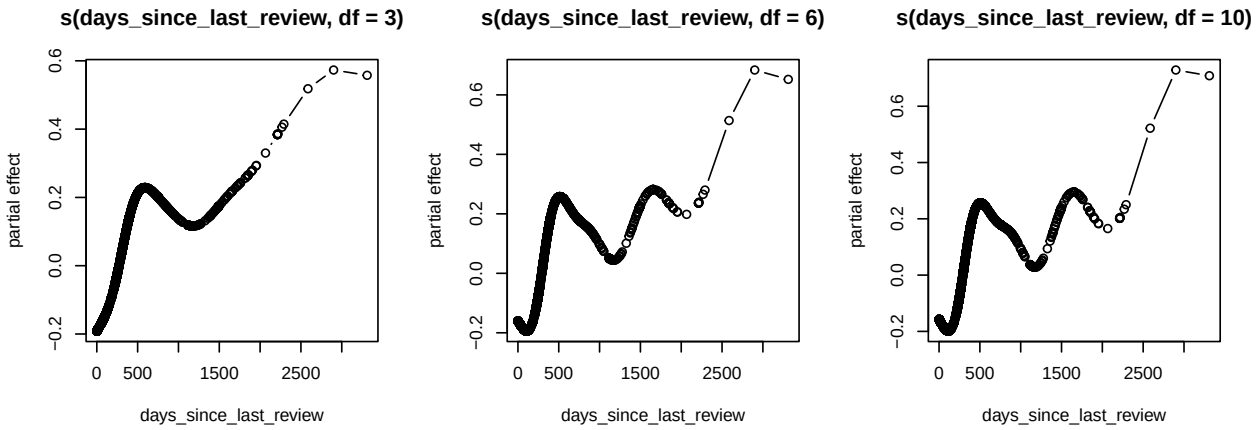
```
## Preparing additive formula(e)... Done.
```

```
mod_df_max <- do.call("deepregression", c(args, list(list_of_formulas = form_df_10)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_df_low %>% fit(epochs = 1000, early_stopping = TRUE, verbose = FALSE)
mod_df_med %>% fit(epochs = 1000, early_stopping = TRUE, verbose = FALSE)
mod_df_max %>% fit(epochs = 1000, early_stopping = TRUE, verbose = FALSE)

par(mfrow=c(1,3))
plot(mod_df_low)
plot(mod_df_med)
plot(mod_df_max)
```



Neural Network Settings

Shared DNN

```
embd_mod <- function(x) x %>%
  layer_embedding(input_dim = 1000,
                  output_dim = 100) %>%
  layer_lambda(f = function(x) k_mean(x, axis = 2)) %>%
  layer_dense(20, activation = "tanh") %>%
  layer_dropout(0.3) %>%
  layer_dense(2)
```

```
mod <- deepregression(
  y = y,
  list_of_formulas =
    list(
      location = ~ 1,
      scale = ~ 1,
      both = ~ 0 + embd_mod(texts)
    ),
  list_of_deep_models =
    list(embd_mod = embd_mod),
  mapping = list(1,2,1:2),
  data = airbnb
)
```

```
## Preparing additive formula(e)... Done.
```

```
embd_mod <- function(x) x %>%
  layer_embedding(input_dim = 1000,
                  output_dim = 100) %>%
  layer_lambda(f = function(x) k_mean(x, axis = 2)) %>%
  layer_dense(20, activation = "tanh") %>%
  layer_dropout(0.3) %>%
  layer_dense(1)

form_lists <- list(
```

```
  location = ~ 1 + embd_mod(texts),
  scale = ~ 1
)

mod <- deepregression(
  y = y,
  list_of_formulas = form_lists,
  list_of_deep_models =
    list(embd_mod = embd_mod),
  data = airbnb
  )
```

```
## Preparing additive formula(e)... Done.
```

Orthogonalization

```
set.seed(42)
n <- 1000
toyX <- rnorm(n)
toyY <- 2*toyX + rnorm(n)
```

```
deep_model <- function(x)
{
  x %>%
    layer_dense(units = 100, activation = "relu", use_bias = FALSE) %>%
    layer_dropout(rate = 0.2) %>%
    layer_dense(units = 50, activation = "relu") %>%
    layer_dropout(rate = 0.2) %>%
    layer_dense(units = 1, activation = "linear")
}
```

```
forms <- list(loc = ~ -1 + toyX + deep_model(toyX), scale = ~ 1)
args <- list(
  y = toyY,
  data = data.frame(toyX = toyX),
  list_of_formulas = forms,
  list_of_deep_models = list(deep_model = deep_model)
)

w_oz <- orthog_control(orthogonalize = TRUE)
wo_oz <- orthog_control(orthogonalize = FALSE)

mod_w_oz <- do.call("deepregression", c(args, list(orthog_options = w_oz)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_wo_oz <- do.call("deepregression", c(args, list(orthog_options = wo_oz)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_w_oz %>% fit(epochs = 1000, early_stopping = TRUE, batch_size = 50, verbose = FALSE)
mod_wo_oz %>% fit(epochs = 1000, early_stopping = TRUE, batch_size = 50, verbose = FALSE)

cbind(
  with = c(coef(mod_w_oz, which_param = 1)[[1]]),
  without = c(coef(mod_wo_oz, which_param = 1)[[1]]),
  linmod = coef(lm(toyY ~ 0 + toyX))
)
```

```
##          with   without   linmod
## toyX 2.002293 0.9065553 2.009948
```

Advanced Usage

Custom Distribution Function

```
function(x){
  do.call(your_tfd_dist,
    lapply(1:ncol(x)[[1]],
      function(i) your_trafo_list_on_inputs[[i]](
        x[,i,drop=FALSE])
    )
  )
}
```

```
## function(x){
##   do.call(your_tfd_dist,
##     lapply(1:ncol(x)[[1]],
##       function(i) your_trafo_list_on_inputs[[i]](
##         x[,i,drop=FALSE])
##     )
##   )
## }
```

Custom Orthogonalization

```
toyXinDisguise <- toyX
```

```
form_known <- list(loc = ~ -1 + toyX + deep_model(toyX), scale = ~ 1)
form_unknown <- list(loc = ~ -1 + toyX + deep_model(toyXinDisguise), scale = ~ 1)
form_manual <- list(loc = ~ -1 + toyX + deep_model(toyXinDisguise) %OZ% (toyXinDisguise), scale = ~ 1)
args <- list(
  y = toyY,
  data = data.frame(toyX = toyX, toyXinDisguise = toyXinDisguise),
  list_of_deep_models = list(deep_model = deep_model)
)

mod_known <- do.call("deepregression", c(args, list(list_of_formulas = form_known)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_unknown <- do.call("deepregression", c(args, list(list_of_formulas = form_unknown)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_manual <- do.call("deepregression", c(args, list(list_of_formulas = form_manual)))
```

```
## Preparing additive formula(e)... Done.
```

```
mod_known %>% fit(epochs = 1000, early_stopping = FALSE, verbose = FALSE)
mod_unknown %>% fit(epochs = 1000, early_stopping = FALSE, verbose = FALSE)
mod_manual %>% fit(epochs = 1000, early_stopping = FALSE, verbose = FALSE)

cbind(
  known = coef(mod_known, which_param = 1)[[1]],
  unknown = coef(mod_unknown, which_param = 1)[[1]],
  manual = coef(mod_manual, which_param = 1)[[1]]
)
```

```
##          [,1]     [,2]     [,3]
## [1,] 2.011471 1.946971 2.011512
```

Working with Images

```
airbnb$image <- paste0("/home/david/airbnb/airbnb/data/pictures/32/",
                       airbnb$id, ".jpg")
```

```
cnn_block <- function(filters, kernel_size, pool_size, rate, input_shape = NULL){
    function(x){
      x %>%
        layer_conv_2d(filters, kernel_size, padding="same", input_shape = input_shape) %>%
        layer_activation(activation = "relu") %>%
        layer_batch_normalization() %>%
        layer_max_pooling_2d(pool_size = pool_size) %>%
        layer_dropout(rate = rate)
    }
  }
```

```
cnn <- cnn_block(filters = 16, kernel_size = c(3,3), pool_size = c(3,3), rate = 0.25,
                 shape(200, 200, 3))
deep_model_cnn <- function(x){
    x %>%
    cnn() %>%
    layer_flatten() %>%
    layer_dense(32) %>%
    layer_activation(activation = "relu") %>%
    layer_batch_normalization() %>%
    layer_dropout(rate = 0.5) %>%
    layer_dense(1)
}
```

```
mod_cnn <- deepregression(
  y = y,
  list_of_formulas = list(
    ~1 + room_type + bedrooms + beds +
      deep_model_cnn(image),
    ~1 + room_type),
  data = airbnb,
  list_of_deep_models = list(deep_model_cnn = list(deep_model_cnn, c(200,200,3))),
  optimizer = optimizer_adam(lr = 0.0001)
)
```

```
## Preparing additive formula(e)... Done.
```

```
mod_cnn %>% fit(
  epochs = 100,
  early_stopping = TRUE,
  patience = 5,
  verbose = FALSE)
```

```
coef(mod_cnn)
```

```
## $room_type
##            [,1]
## [1,]  0.3227177
## [2,] -0.3778427
## [3,]  0.4904232
## [4,] -0.3018944
##
## $bedrooms
##              [,1]
##  [1,]  0.00614953
##  [2,] -0.41926220
##  [3,] -0.35590124
##  [4,]  0.21725744
##  [5,] -0.40246907
##  [6,] -0.31668970
##  [7,] -0.16164863
##  [8,] -0.05593127
##  [9,]  0.36044568
## [10,] -0.04302275
##
## $beds
##              [,1]
##  [1,] -0.25642404
##  [2,]  0.31782037
##  [3,]  0.15808105
##  [4,]  0.48887384
##  [5,]  0.50282443
##  [6,]  0.02143788
##  [7,] -0.45789778
##  [8,] -0.34924501
##  [9,]  0.27046627
```

```
## [10,] -0.36034852
## [11,]  0.52438724
## [12,]  0.02861530
## [13,] -0.33916539
## [14,] -0.18039402
## [15,]  0.17045712
## [16,]  0.39523315
## [17,]  0.41156256
##
## $`(Intercept)`
##          [,1]
## [1,] 1.211872
```