

GPTree Vignette

General application

In this vignette, we explore the GPTreeO package using the modified 2d toy data set from Higdon (2002):

```
library(GPTreeO)

set.seed(42)

X <- matrix(runif(200), ncol = 2)
colnames(X) <- c("X1", "X2")
target <- function(x) list(y = mean(sin(2*pi*x) + 0.2*sin(8*pi*x) + 0.1*rnorm(1)),
                           y_var = 0.1**2)
```

Next, we initialize a tree with up to 50 points per local GP and default values otherwise by using `GPTree$new`:

```
gptree <- GPTree$new(Nbar = 50)
```

For the purpose of this example, we simulate the data stream through a simple for loop. In actual applications, the input stream comes from e.g. a differential evolutionary scanner. Note that GPTreeO only trains on the data stream and does not influence it in any way.

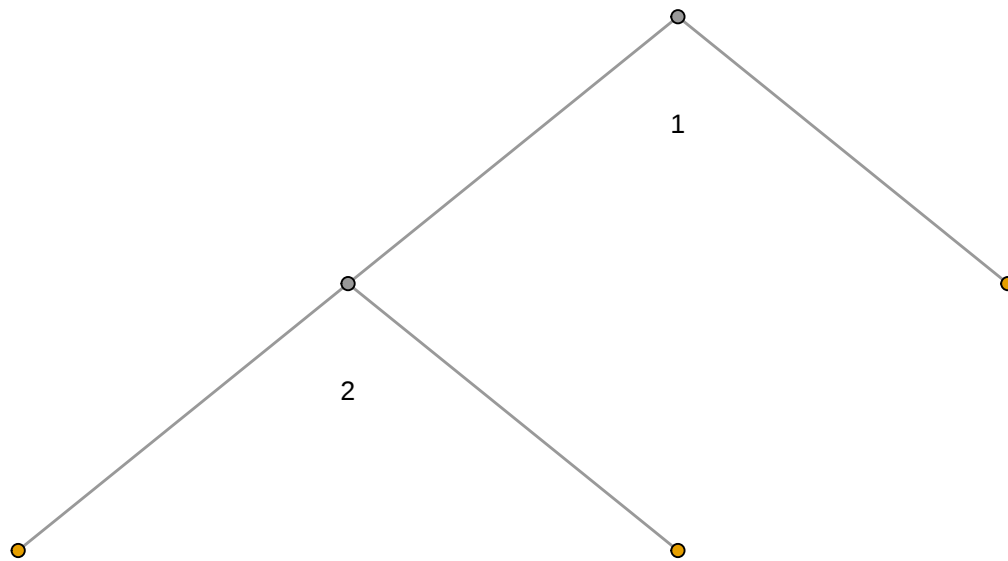
We follow the procedure in the associated paper, thus letting the tree make a prediction first using `joint_prediction`, before we update the tree with the point with `update`:

```
predictions <- NULL

for (i in 1:nrow(X)) {
  x <- X[i,]
  predictions <- rbind(gptree$joint_prediction(x), predictions)
  target_output <- target(x)
  gptree$update(x, target_output$y, target_output$y_var)
}
```

Next, we can plot the tree structure using `plot_tree_structure`. The orange vertices are leafs which contain local GPs, and the grey vertices are (empty) nodes. The splitting dimensions is noted below each vertex.

```
gptree$plot_tree_structure(include_shared = TRUE)
```

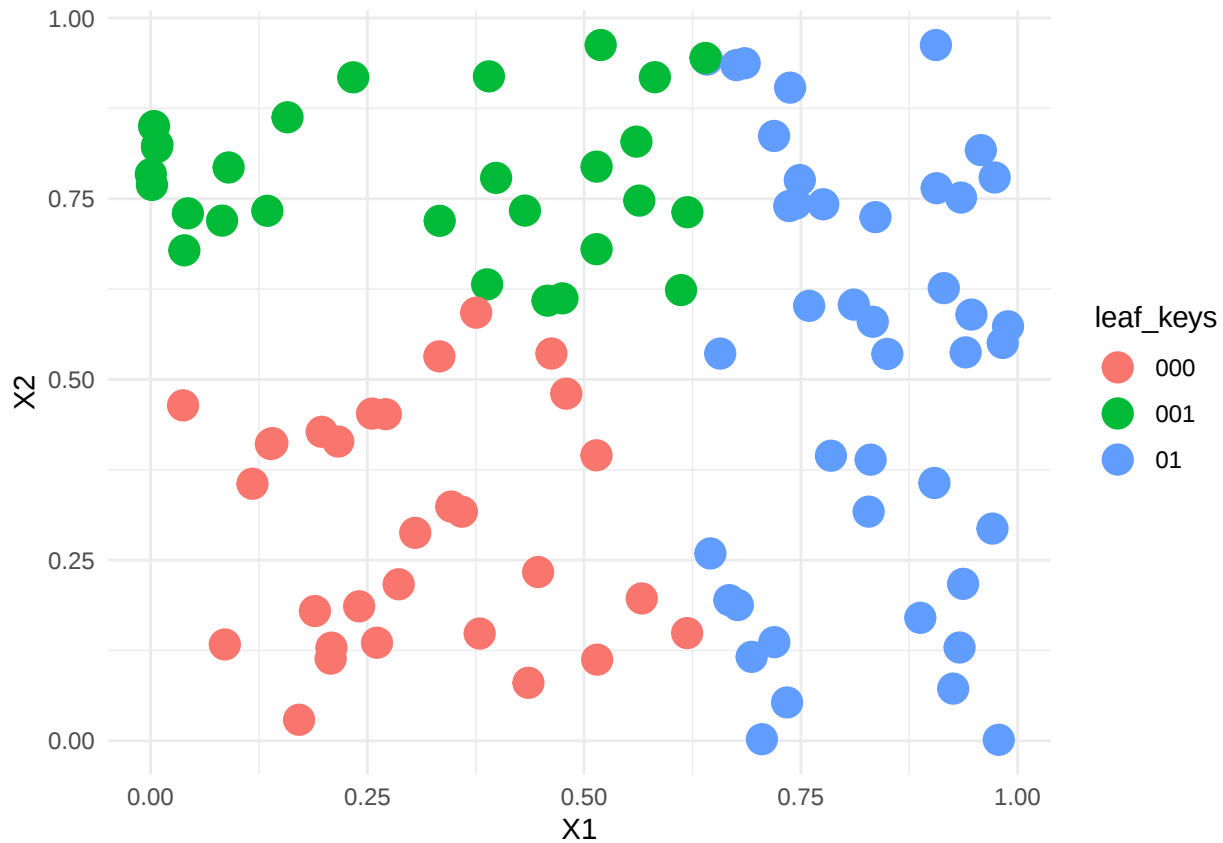


We can also visualize the split by using `get_leaf_key_from_input_point` which allows us to visualize to which leaves a training point belongs:

```
library(ggplot2)
library(tidyr)
library(dplyr)

leaf_keys <- apply(X, 1, gptree$get_leaf_key_from_input_point, include_shared = FALSE)
train_data_with_keys <- data.frame(X, leaf_keys, check.names = FALSE)

ggplot(data = train_data_with_keys, aes(x = X1, y = X2, colour = leaf_keys)) +
  geom_point(size = 5) +
  theme_minimal()
```



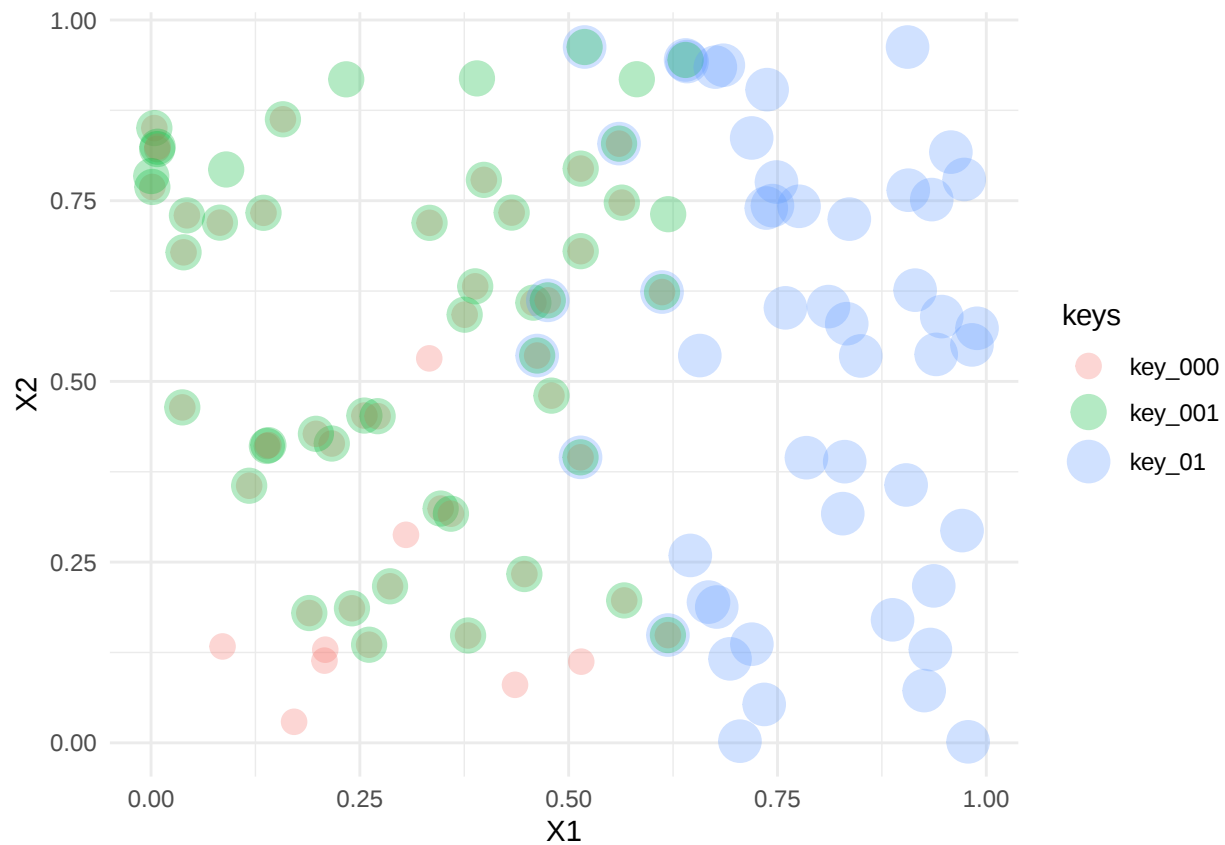
We can see the same structure as when we plotted the tree: There is one single leaf on the right hand side, and two split-up leaves on the left-hand side.

As we use gradual splitting by default, many of the points are shared across leaves:

```
leaf_keys <- t(apply(X, 1, gptree$get_leaf_key_from_input_point, include_shared = TRUE))
colnames(leaf_keys) <- paste0("key_", gptree$leaf_keys)

train_data_with_keys <- data.frame(X, leaf_keys, check.names = FALSE) %>%
  pivot_longer(cols = starts_with("key"),
               names_to = "keys") %>%
  filter(value == TRUE) %>%
  select(-value)

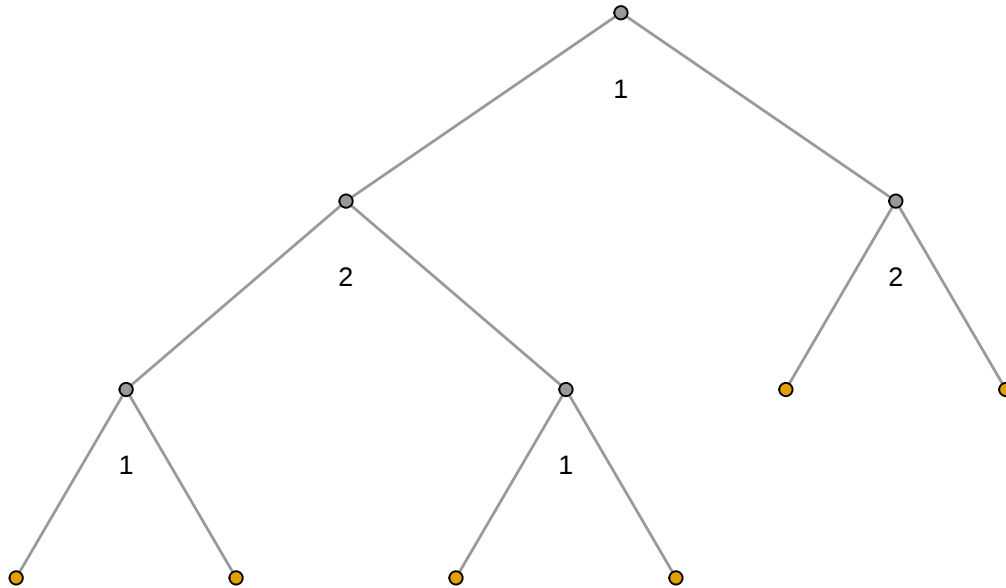
ggplot(data = train_data_with_keys, aes(x = X1, y = X2, colour = keys, size = keys)) +
  geom_point(alpha = 0.3) +
  scale_size_discrete(range = c(4, 7)) +
  theme_minimal()
```



Different tree parameter settings

Now that we have covered general usage, we dive deeper into tree parameters and their effects. First, let's consider a tree with more leafs. We can achieve this by reducing Nbar:

```
gptree <- GPTree$new(Nbar = 25)
for (i in 1:nrow(X)) {
  x <- X[i,]
  target_output <- target(x)
  gptree$update(x, target_output$y, target_output$y_var)
}
gptree$plot_tree_structure(include_shared = TRUE)
```



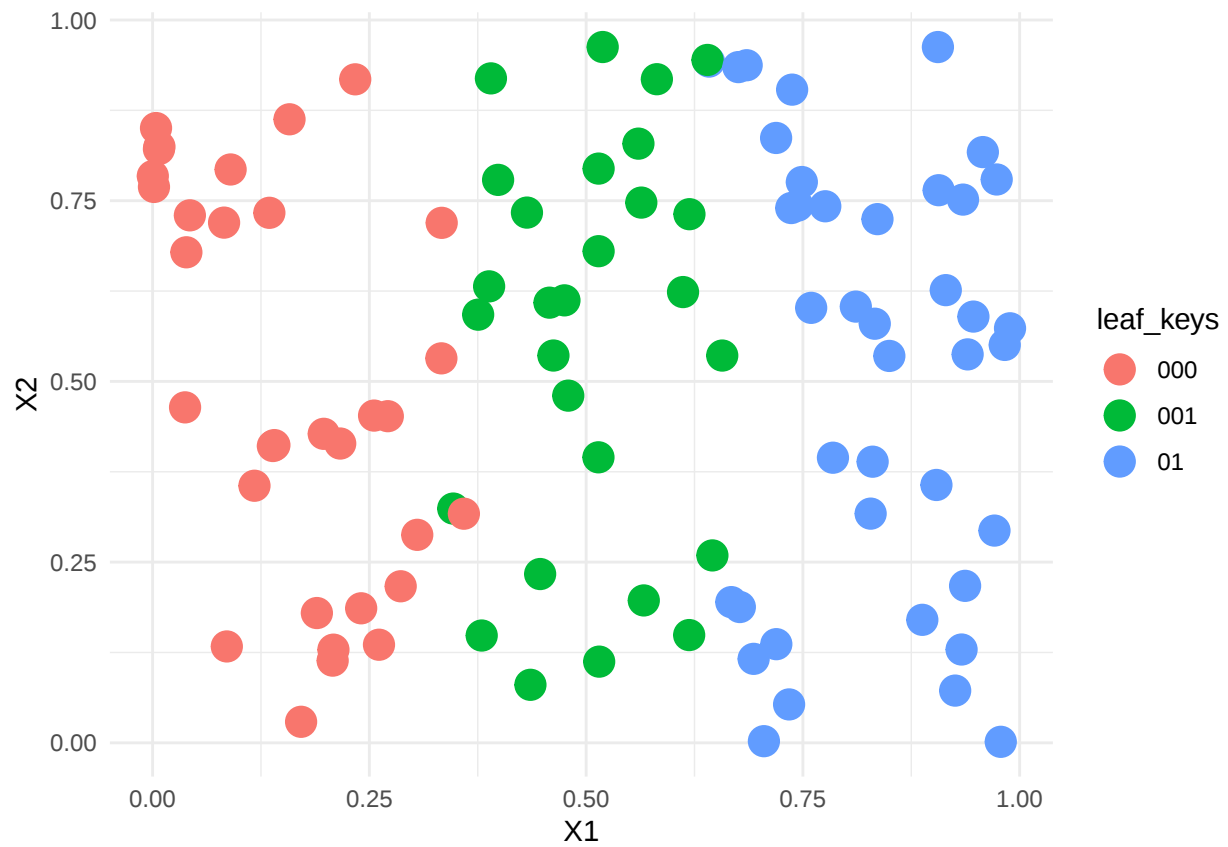
Next, we consider a tree without gradual splitting and with an overlap of 10 % between leafs and `Nbar = 50`:

```

gptree <- GPTree$new(Nbar = 50, theta = 0.1, gradual_split = FALSE)
for (i in 1:nrow(X)) {
  x <- X[i,]
  target_output <- target(x)
  gptree$update(x, target_output$y, target_output$y_var)
}
leaf_keys <- apply(X, 1, gptree$get_leaf_key_from_input_point, include_shared = FALSE)
train_data_with_keys <- data.frame(X, leaf_keys, check.names = FALSE)

ggplot(data = train_data_with_keys, aes(x = X1, y = X2, colour = leaf_keys)) +
  geom_point(size = 5) +
  theme_minimal()

```



Now, we can see that there are some points that lie outside of the typical region of their respective region.