03.Rozszerzanie MLR

Dodajemy prostą metodę - `foo`

Metoda konstruuje model postaci:

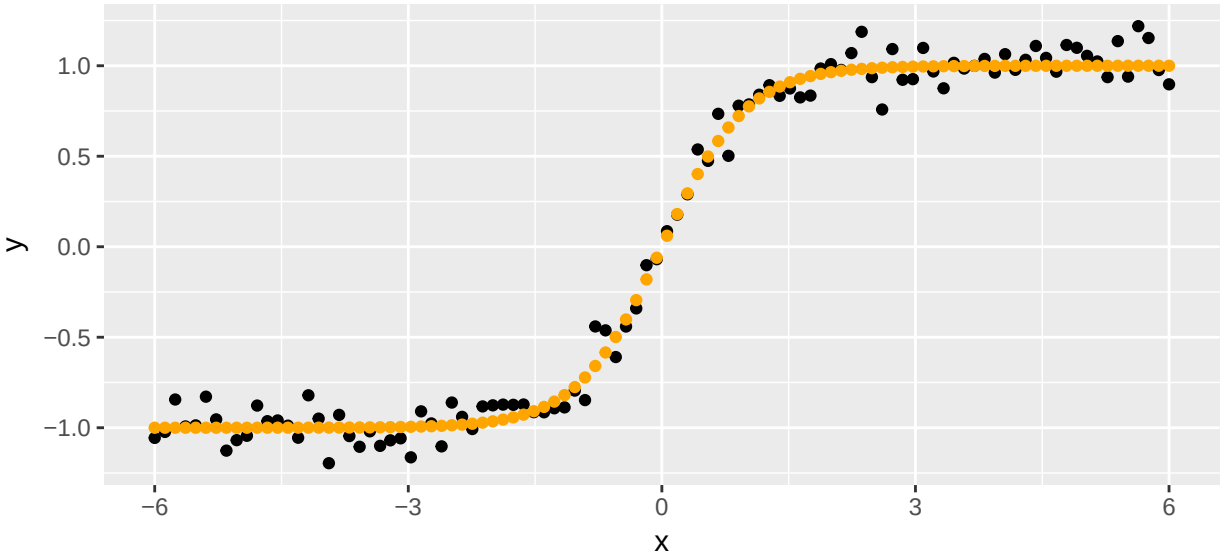
$$y = \tanh(\mathbf{wx})$$

```
## Obiekt z opisem metody
makeRLearner.regr.foo <- function() {
  makeRLearnerRegr(
    cl = "regr.foo",
    package = "stats",
    par.set = makeParamSet(
        makeIntegerLearnerParam("maxit", lower=10, upper = Inf)
    ),
    properties = c("numerics"),
    name = "Simple foo learner",
    short.name = "foo",
    note = ""
  )
}

fooPredict <- function(x, w){
  tanh(x %*% w) %>% as.vector()
}

## Uczenie
trainLearner.regr.foo = function(.learner, .task, .subset, .weights = NULL, ...) {
  trainData <- getTaskData(.task,.subset, target.extra = TRUE)
  res <- optim(rnorm(ncol(trainData$data)),
        function(w){
          mean((fooPredict(as.matrix(trainData$data), w) - trainData$target)^2)
        }, control = list(trace = 0, ...))

  list(w = res$par)
}

## Predykcja przy pomocy gotowego modelu
predictLearner.regr.foo = function(.learner, .model, .newdata, ...) {
  model <- .model$learner.model

  fooPredict(as.matrix(.newdata), model$w)
}
```

Przykładowe użycie

```
x <- seq(-6,6,length.out=100); y <- tanh(x) + rnorm(100, sd=0.1)
trainingData <- data.frame(x,y)
```

```
tsk <- makeRegrTask(data=trainingData, target = "y")
m <- train(makeLearner('regr.foo', maxit = 50), tsk)

preds <- predict(m, tsk)
ggplot(trainingData, aes(x,y)) + geom_point() + geom_point(y=getPredictionResponse(preds),color="orange
```



Rejestracja metody

```
registerS3method("makeRLearner", "regr.foo", makeRLearner.regr.foo)
registerS3method("trainLearner", "regr.foo", trainLearner.regr.foo)
registerS3method("predictLearner", "regr.foo", predictLearner.regr.foo)
listLearners('regr') %>% dplyr::filter(class=='regr.foo')
```

```
##      class               name short.name package note type installed
## 1 regr.foo Simple foo learner        foo   stats      regr      TRUE
##   numerics factors ordered missings weights  prob oneclass twoclass
## 1     TRUE   FALSE   FALSE    FALSE   FALSE FALSE    FALSE    FALSE
##   multiclass class.weights featimp oobpreds    se lcens rcens icens
## 1      FALSE         FALSE   FALSE    FALSE FALSE FALSE FALSE FALSE
```

Ćwiczenia:

1. rozszerzenie metody do modelu postaci: $y = w_0 + \texttt{tanh}(\mathbf{wx})$,
2. z modelu można wyciągnąć informacje (w tym przypadku wagi) przydatne podczas selekcji: `getFeatureImportance(m)` - trzeba tylko nieznacznie rozszerzyć metodę.
3. a jak poradzi sobie nasza metoda na zadaniu z poprzednich ćwiczeń? (możemy użyć do porównania wyników benchmarku:

```
previousBenchmark <- readRDS('data/02_benchmark.RDS')
```