Gaussian Linear Model

Priyanshu Tiwari

2025-03-18

Loading Torch

```r
library(torch)
torch_manual_seed(1) # setting seed for reproducibility
```

Creating a Gaussian Linear Model

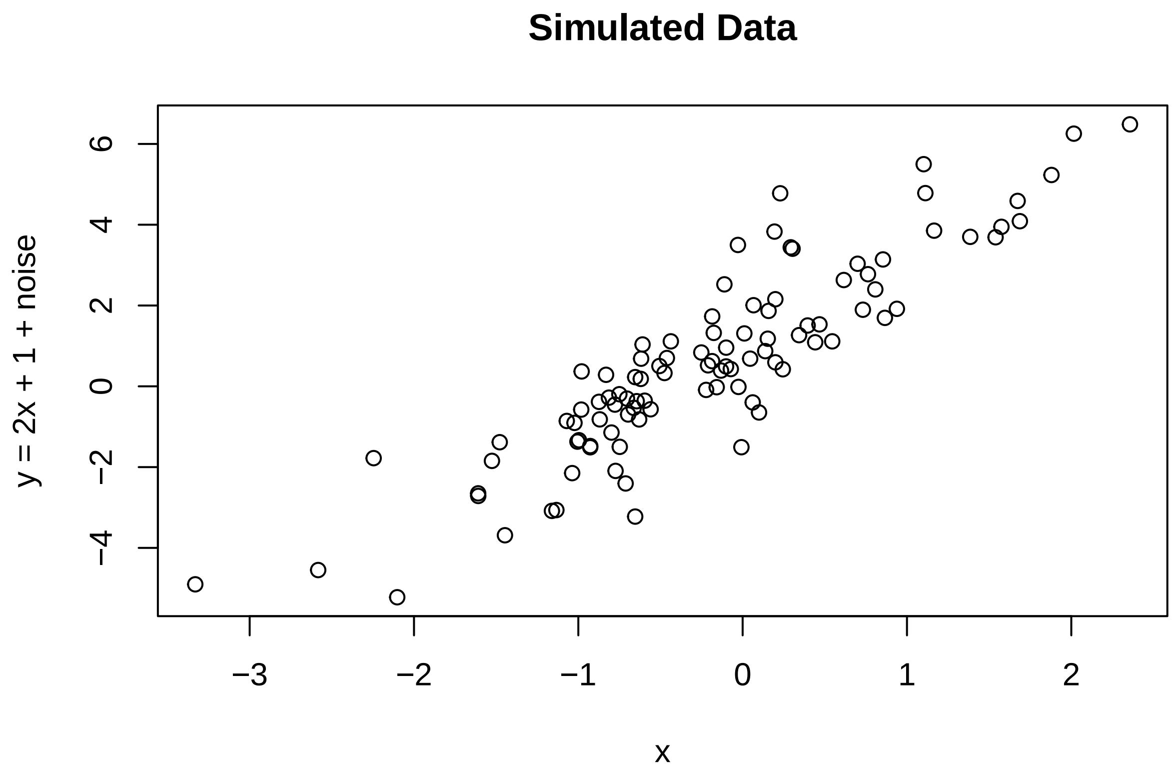
Taking example from the distributions vignette to create a Gaussian linear model.

First, simulate some data:

```r
x <- torch_randn(100, 1)
y <- 2*x + 1 + torch_randn(100, 1)
```

Visualize this data

```r
plot(as.numeric(x), as.numeric(y),
     main = "Simulated Data",
     xlab = "x", ylab = "y = 2x + 1 + noise")
```



Defining the model

```
GaussianLinear <- nn_module(
  initialize = function() {
    # this linear predictor will estimate the mean of the normal distribution
    self$linear <- nn_linear(1, 1)
    # this parameter will hold the estimate of the variability
    self$scale <- nn_parameter(torch_ones(1))
  },
  forward = function(x) {
    # we estimate the mean
    loc <- self$linear(x)
    # return a normal distribution
    distr_normal(loc, self$scale)
  }
)

model <- GaussianLinear()
```

Training the model

```
opt <- optim_sgd(model$parameters, lr = 0.1)

for (i in 1:100) {
  opt$zero_grad()
  d <- model(x)
```

```r
  loss <- torch_mean(-d$log_prob(y))
  loss$backward()
  opt$step()
  if (i %% 10 == 0)
    cat("iter: ", i, " loss: ", loss$item(), "\n")
}
#> iter:  10  loss:  1.798009
#> iter:  20  loss:  1.657071
#> iter:  30  loss:  1.556822
#> iter:  40  loss:  1.499135
#> iter:  50  loss:  1.477929
#> iter:  60  loss:  1.472361
#> iter:  70  loss:  1.471061
#> iter:  80  loss:  1.470764
#> iter:  90  loss:  1.470697
#> iter:  100  loss:  1.470681
```

Parameter estimates

```r
model$parameters
#> $linear.weight
#> torch_tensor
#>  2.1752
#> [ CPUFloatType{1,1} ][ requires_grad = TRUE ]
#>
#> $linear.bias
#> torch_tensor
#>  1.0343
#> [ CPUFloatType{1} ][ requires_grad = TRUE ]
#>
#> $scale
#> torch_tensor
#>  1.0531
#> [ CPUFloatType{1} ][ requires_grad = TRUE ]
```

Comparing with the glm function

```r
summary(glm(as.numeric(y) ~ as.numeric(x)))
#>
#> Call:
#> glm(formula = as.numeric(y) ~ as.numeric(x))
#>
#> Coefficients:
#>               Estimate Std. Error t value Pr(>|t|)
#> (Intercept)     1.0366     0.1087    9.54 1.21e-15 ***
#> as.numeric(x)   2.1775     0.1093   19.93  < 2e-16 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#> (Dispersion parameter for gaussian family taken to be 1.131653)
#>
#>     Null deviance: 560.25  on 99  degrees of freedom
#> Residual deviance: 110.90  on 98  degrees of freedom
```

```
#> AIC: 300.14
#>
#> Number of Fisher Scoring iterations: 2
```

Visualizing the model results

Plot showing data and the fitted line:

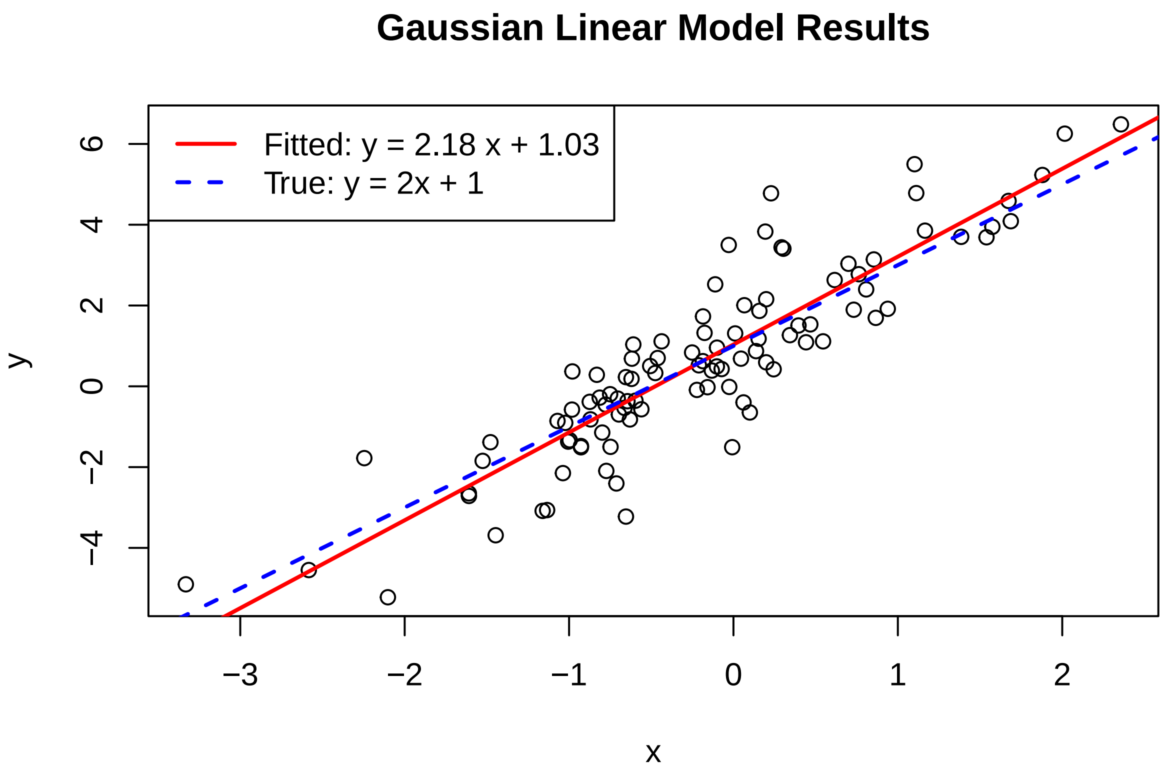
```r
# Get the model parameters
params <- model$parameters
weight <- as.numeric(params[[1]])
bias <- as.numeric(params[[2]])

# Create a plot
plot(as.numeric(x), as.numeric(y),
     main = "Gaussian Linear Model Results",
     xlab = "x", ylab = "y")

# Add the fitted line
abline(a = bias, b = weight, col = "red", lwd = 2)

# Add true line
abline(a = 1, b = 2, col = "blue", lwd = 2, lty = 2)

# Legend
legend("topleft",
       legend = c(
         paste("Fitted: y =", round(weight, 2), "x +", round(bias, 2)),
         "True: y = 2x + 1"
       ),
       col = c("red", "blue"),
       lwd = 2,
       lty = c(1, 2))
```


Gaussian Linear Model Results
Fitted: y = 2.18 x + 1.03
True: y = 2x + 1
6
4
2
0
−2
−4
y
−3
−2
−1
0
1
2
x