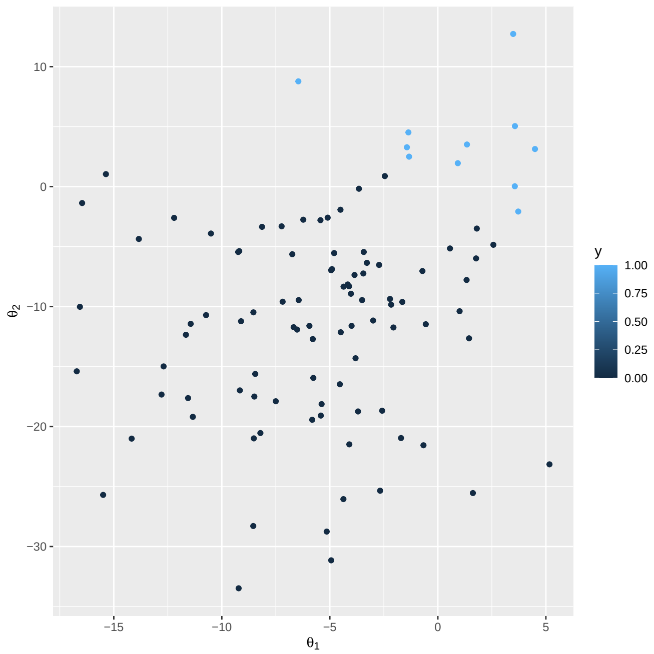
Multivariate Optimization 1

Exercise 1: Gradient Descent

You are given the following data situation:

```
library(ggplot2)

set.seed(314)
n <- 100
X = cbind(rnorm(n, -5, 5),
  rnorm(n, -10, 10))
X_design = cbind(1, X)

z <- 2*X[,1] + 3*X[,2]
pr <- 1/(1+exp(-z))
y <- as.integer(pr > 0.5)
df <- data.frame(X = X, y = y)

ggplot(df) +
  geom_point(aes(x = X.1, y = X.2, color=y)) +
  xlab(expression(theta[1])) +
  ylab(expression(theta[2]))
```



In the following we want to estimate a logistic regression without intercept via gradient descent[1].

(a) First consider the derivative of $g : \mathbb{R} \to \mathbb{R}, z \mapsto \log(1 + \exp(z)) - z$, i.e.,

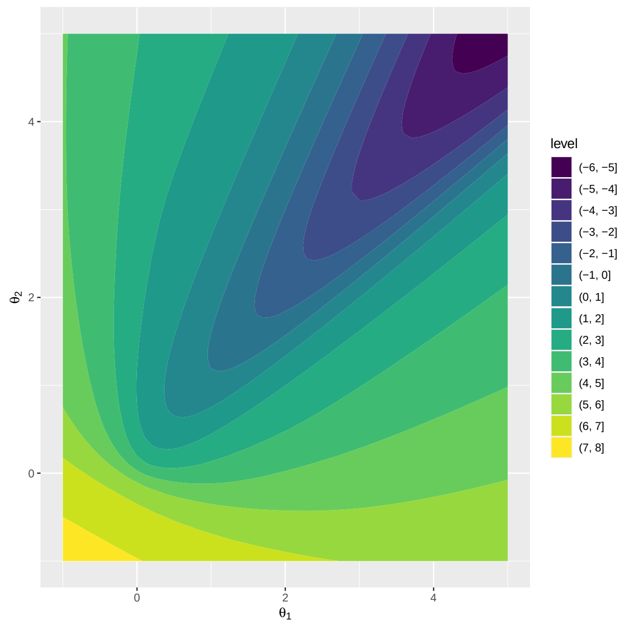
$$g'(z) = \underbrace{\frac{\exp(z)}{1 + \exp(z)}}_{<1} - 1 < 0 \Rightarrow g \text{ is monotonically decreasing} \Rightarrow g(z) > g(\alpha z) \quad \forall z > 0 \text{ and } \alpha > 1.$$

[1] We chose this algorithm for educational purposes; in practice, we typically use second order algorithms.

Second consider the derivative of $h : \mathbb{R} \to \mathbb{R}, z \mapsto \log(1 + \exp(-z))$, i.e.,
$h'(z) = -\underbrace{\frac{\exp(-z)}{1+\exp(-z)}}_{>0} < 0 \Rightarrow h$ is monotonically decreasing $\Rightarrow h(z) > h(\alpha z) \quad \forall z > 0$ and $\alpha > 1$.

With this we get for $\alpha > 1$
$\mathcal{R}_{\text{emp}}(\tilde{\boldsymbol{\theta}}) = \sum_{i=1}^{n} \log(1 + \exp(\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)})) - y^{(i)}\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)} =$
$\sum_{i=1}^{n} \mathbb{1}_{y^{(i)}=1}(\log(1 + \exp(|\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)}|)) - |\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)}|) + \mathbb{1}_{y^{(i)}=0}(\log(1 + \exp(-|\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)}|)) >$
$\sum_{i=1}^{n} \mathbb{1}_{y^{(i)}=1}(\log(1 + \exp(\alpha|\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)}|)) - \alpha|\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)}|) + \mathbb{1}_{y^{(i)}=0}(\log(1 + \exp(-\alpha|\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)}|)) =$
$\sum_{i=1}^{n} \log(1 + \exp(\alpha\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)})) - y^{(i)}\alpha\tilde{\boldsymbol{\theta}}^\top \mathbf{x}^{(i)} =$
$\mathcal{R}_{\text{emp}}(\alpha\tilde{\boldsymbol{\theta}})$ since $\tilde{\boldsymbol{\theta}}$ perfectly seperates the data.

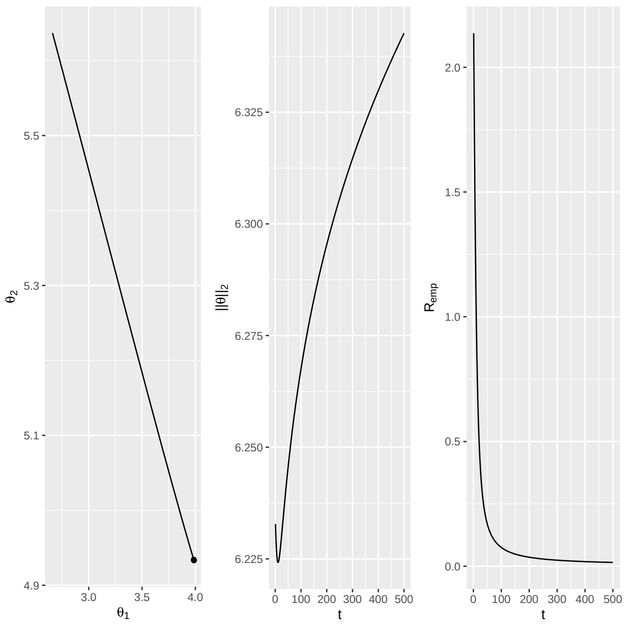
(b)
```
lambda = 0

f <- function(theta, lambda) lambda * theta %*% theta +
  sum(-y * X %*% theta + log(1 + exp(X %*% theta)))

x = seq(-1, 5, by=0.1)
xx = expand.grid(X1 = x, X2 = x)

fxx = log(apply(xx, 1, function(t) f(t, lambda)))
df = data.frame(xx = xx, fxx = fxx)

ggplot() +
    geom_contour_filled(data = df, aes(x = xx.X1, y = xx.X2, z = fxx)) +
    xlab(expression(theta[1])) +
    ylab(expression(theta[2]))
```



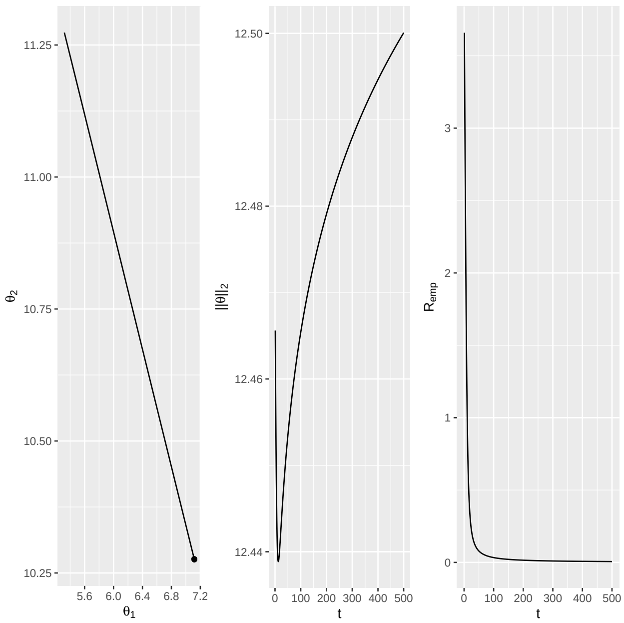
(c) $\frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{R}_{\text{emp}} = \sum_{i=1}^{n} \frac{\exp(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})}{1+\exp(\boldsymbol{\theta}^\top \mathbf{x}^{(i)})} \mathbf{x}^{(i)\top} - y^{(i)} \mathbf{x}^{(i)\top}$

(d)
```
library(gridExtra)

plot_fun <- function(gd_fun, lambda){
  theta = c(0,0)
  thetas = NULL
  thetas_norm = NULL
  fs = NULL
```

```
  for(i in 1:500){
    theta = gd_fun(theta)
    thetas_norm = rbind(thetas_norm, sqrt(theta %*% theta))
    thetas = rbind(thetas, theta)
    fs = rbind(fs, f(theta, lambda))
  }

  df_trace = as.data.frame(thetas)
  trace_plot = ggplot() +
    geom_line(data = df_trace, aes(x=V1, y=V2)) +
    xlab(expression(theta[1])) +
    ylab(expression(theta[2])) +
    geom_point(data = tail(df_trace), aes(x=V1, y=V2))
  norm_plot = ggplot(data.frame(norms = thetas_norm, t = 1:nrow(thetas_norm))) +
    geom_line(aes(x = t, y = norms)) + ylab(expression(paste("||", theta, "||"[2])))
  remp_plot = ggplot(data.frame(f = fs, t = 1:nrow(thetas_norm))) +
    geom_line(aes(x = t, y = f)) + ylab(expression(R[emp]))
  grid.arrange(trace_plot, norm_plot, remp_plot, ncol=3)
}

df_t <- function(theta, lambda) lambda * t(theta) -(t(y) %*% X) +
  t(1/(1 + exp(-X %*% theta))) %*% X

gd_step <- function(theta, alpha, lambda) return(theta - alpha * df_t(theta, lambda)[1,])

## Alpha = 0.01
gd_fun <- function(theta) return(gd_step(theta, 0.01, lambda))
plot_fun(gd_fun, 0)
```

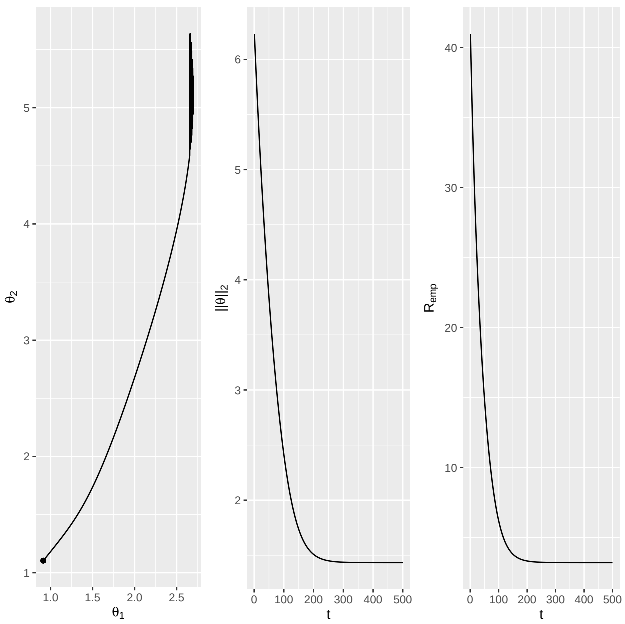



```
## Alpha = 0.02
gd_fun <- function(theta) return(gd_step(theta, 0.02, lambda))
plot_fun(gd_fun, 0)
```

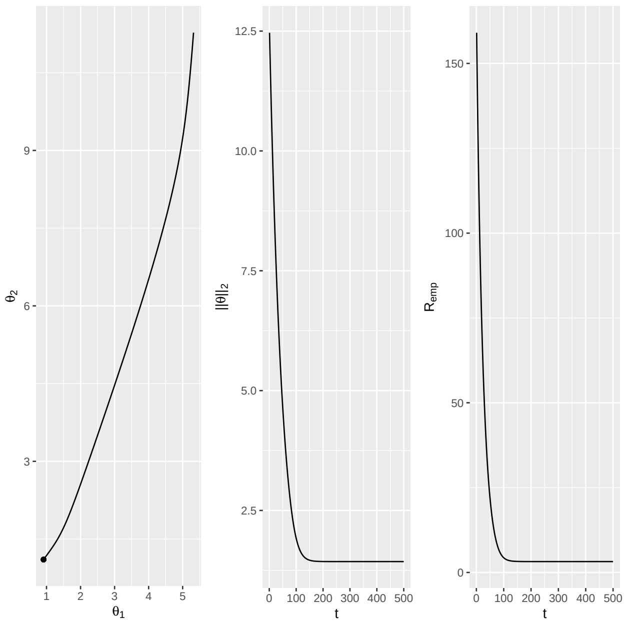


Gradient descent will in theory not converge since $\mathcal{R}_{\text{emp}}$ has no minimum (a)

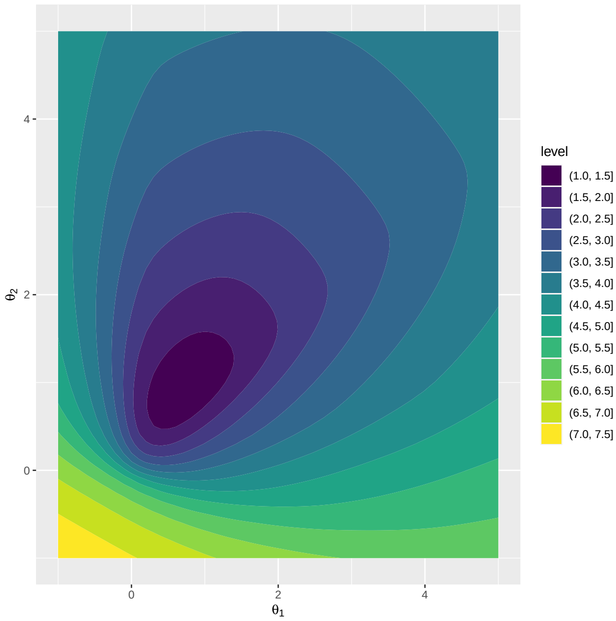
(e)
```r
## Lambda = 1, alpha = 0.01
gd_fun <- function(theta) return(gd_step(theta, 0.01, 1))
plot_fun(gd_fun, 1)
```




```r
## Lambda = 1, alpha = 0.02
gd_fun <- function(theta) return(gd_step(theta, 0.02, 1))
plot_fun(gd_fun, 1)
```

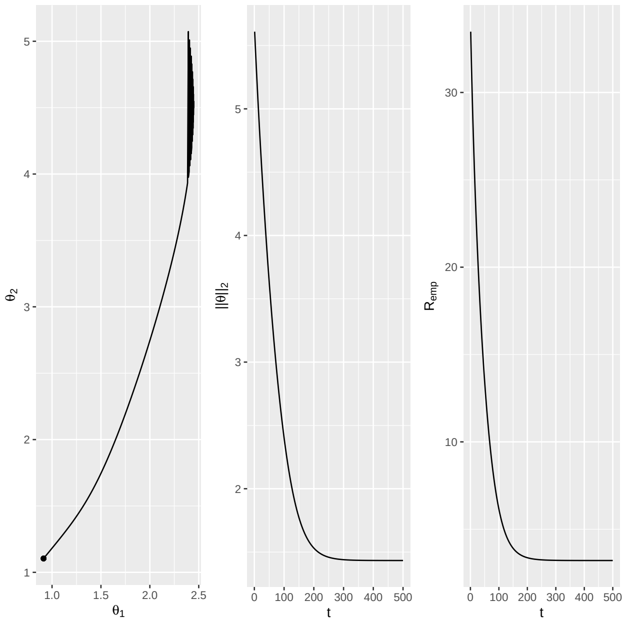


(f)
```
lambda = 1

fxx_reg = log(apply(xx, 1, function(t) f(t, lambda)))
df_reg = data.frame(xx = xx, fxx = fxx_reg)

ggplot() +
    geom_contour_filled(data = df_reg, aes(x = xx.X1, y = xx.X2, z = fxx)) +
    xlab(expression(theta[1])) +
    ylab(expression(theta[2]))
```



(g)
```
gd_backtracking_step <- function(theta, alpha, gamma, tau, lambda){
    ftheta = f(theta, lambda)
    for(i in 1:1000){
      theta_prop = theta - alpha * gamma * df_t(theta, lambda)[1,]
      if(f(theta_prop, lambda) < ftheta){
        return(theta_prop)
```

```r
      }else{
        alpha = tau * alpha
      }
    }
    return(theta)
}

## Lambda = 1, alpha = 0.01
gd_fun <- function(theta) return(gd_backtracking_step(theta, 0.01, 0.9, 0.5, 1))
plot_fun(gd_fun, 1)
```




```r
## Lambda = 1, alpha = 0.02
gd_fun <- function(theta) return(gd_backtracking_step(theta, 0.02, 0.9, 0.5, 1))
plot_fun(gd_fun, 1)
```