Lab/HW 8: Optimization; Sampling

Your lab/homework must be submitted in with two files: (1) R Markdown format file; (2) a pdf or html file, unless otherwise stated. Other formats will not be accepted. Your responses must be supported by both textual explanations and the code you generate to produce your result.

Continue from Homework 7: Part III – Maximize Likelihood

In Part III, we continue to use the data on the heart weight of cats from Part II.

1. [Done in HW7] Review Part II. Fit a gamma distribution to the cats' heart weights by maximum likelihood using `optim()`. Remember, the parameters for `optim` are in vector form (the `par` argument). You'll need to write a function that returns a the value of the likelihood that accepts the parameters as a vector rather than two scalars. Verify that the gradient of the negative log-likelihood at the maximum likelihood estimate is close to zero. Did you get better results compared to the method of moments estimates?
2. [25 pt] Perform the fit using the gradient descent function from class. Remember, the parameters are in vector form (the `theta0` argument). You'll need to write a function that returns a the value of the likelihood that accepts the parameters as a vector rather than two scalars. Did you get better results compared to the method of moments estimates? Compared to `optim`?
3. [5 pt Extra Credit] create a dense grid of shape and scale values around the estimates you found and see if you can further improve the result.

Visualizing the Gradient Descent algorithm

1. [50 pt] Modify the code for the function that implements the gradient descent algorithm so that it includes an additional logical input `visualize`. When `visualize = TRUE` please add lines to your code that will show a contour plot of the function you are optimizing, as well as the successive values of `theta` and lines connecting those. For the latter, you can use the `points()` and `segments()` functions.

Please demonstrate your code by reproducing the following:

The function we optimize is the density of a bivariate normal density:

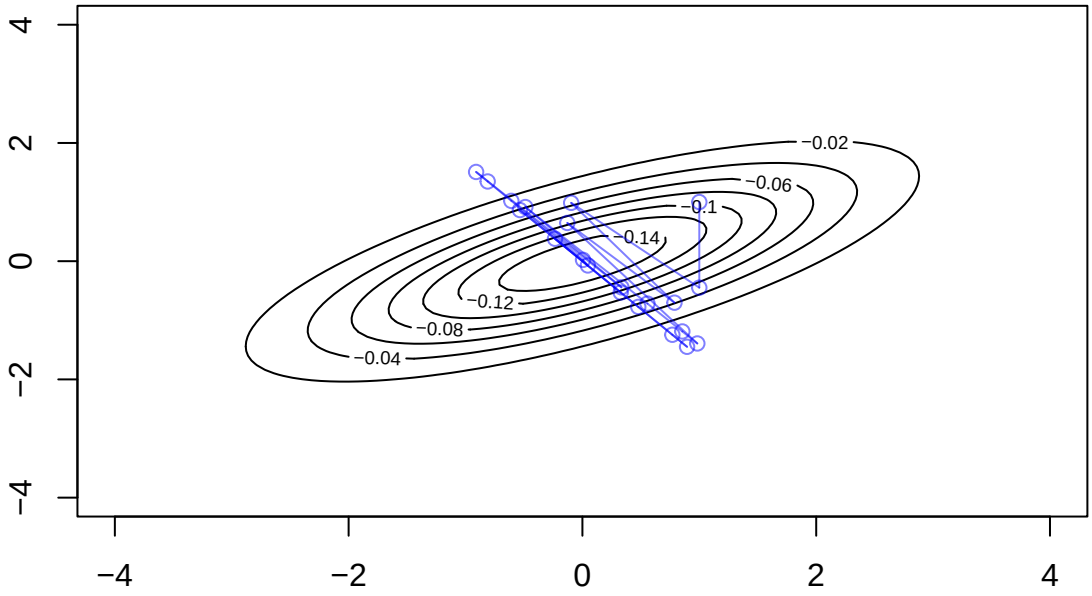
```
f <- function(x) {
  return( - mvtnorm::dmvnorm(x, sigma = matrix(c(2, 1, 1, 1), nrow = 2)) )
}
```

When `step_scale` is big, the `theta` values may hover around the minimum point without converging:

```
step_scale_too_big <-
  gradient_descent(
    f,
    theta0 = c(1, 1),
    step_scale = 15,
    stopping_deriv = 1e-6,
    max_iter = 20,
    visualize = TRUE
    )
```

```
## Warning in gradient_descent(f, theta0 = c(1, 1), step_scale = 15,
```

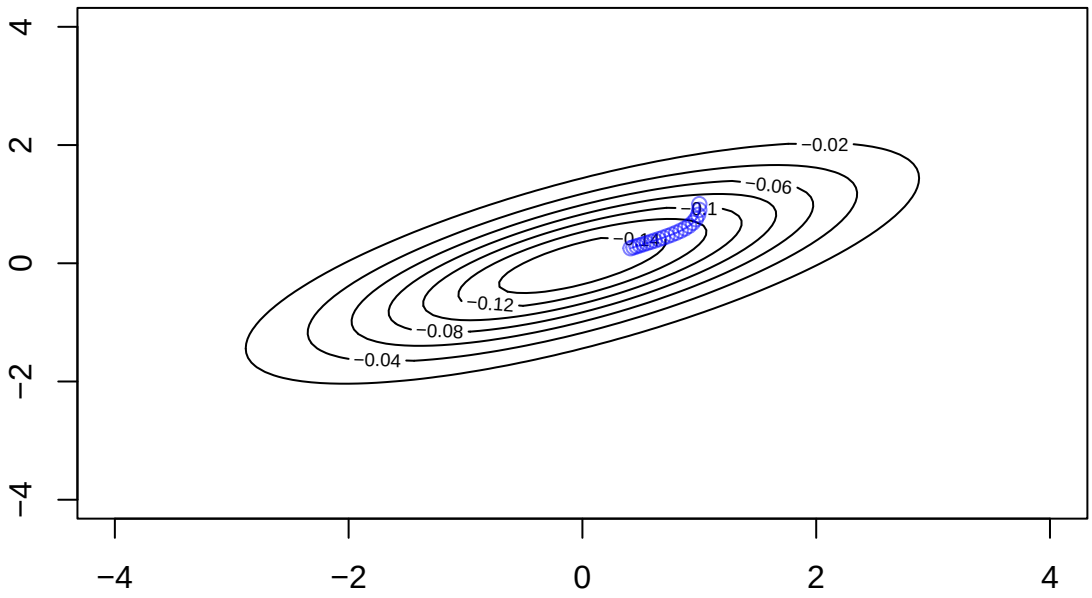
```
## stopping_deriv = 1e-06, : Maximal number of iterations reached, convergence not
## assured
```



When `step_scale` is small, the `theta` values may converge slowly:

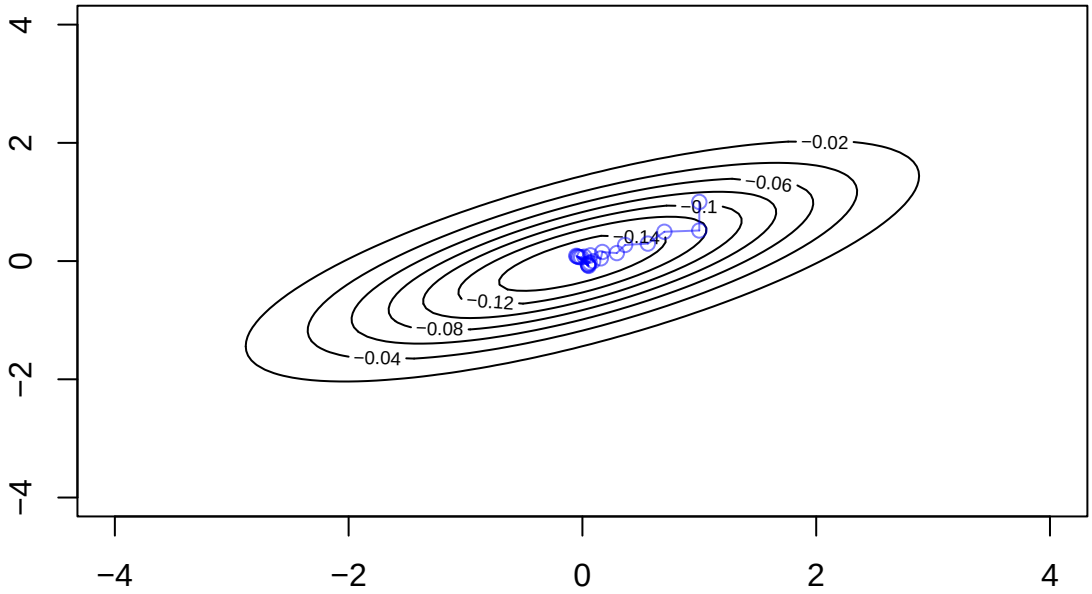
```
step_scale_too_small <-
  gradient_descent(
    f,
    theta0 = c(1, 1),
    step_scale = 1,
    stopping_deriv = 1e-6,
    max_iter = 20,
    visualize = TRUE
    )
```

```
## Warning in gradient_descent(f, theta0 = c(1, 1), step_scale = 1, stopping_deriv
## = 1e-06, : Maximal number of iterations reached, convergence not assured
```



```
step_scale_quite_right <-
  gradient_descent(
    f,
    theta0 = c(1, 1),
    step_scale = 5,
    stopping_deriv = 1e-6,
    max_iter = 20,
    visualize = TRUE
    )
```

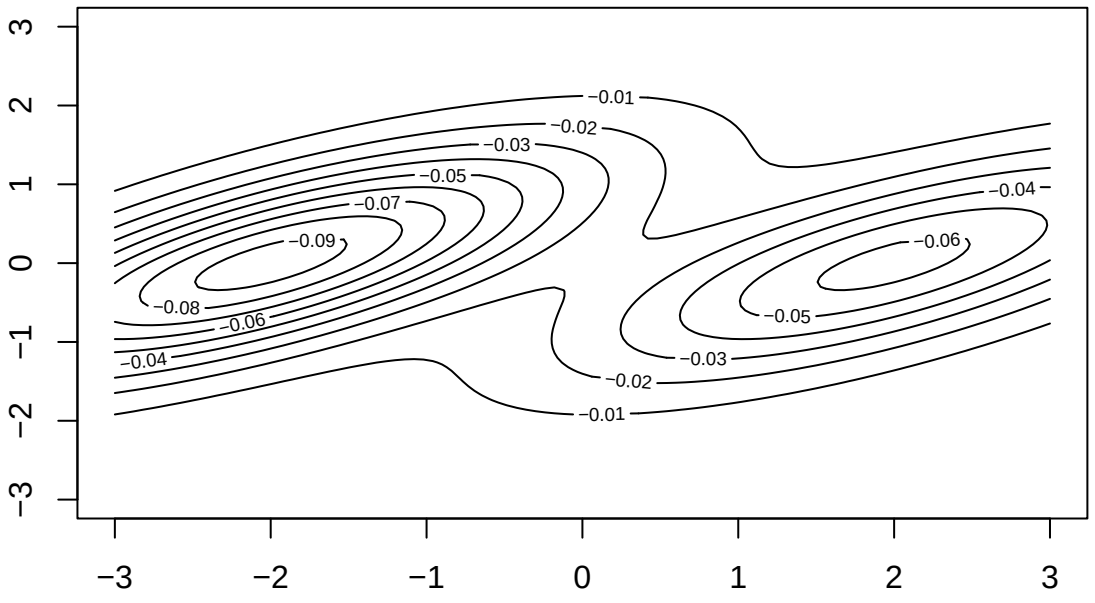
```
## Warning in gradient_descent(f, theta0 = c(1, 1), step_scale = 5, stopping_deriv
## = 1e-06, : Maximal number of iterations reached, convergence not assured
```



2. [25 pt] Generate 3 interesting visualizations of the convergence of the gradient descent algorithm for a function that is the density of a mixture model of two normal distributions:

- The first distribution is a bivariate normal, $N\left(\begin{bmatrix}-2\\0\end{bmatrix}, \begin{bmatrix}2 & 1\\1 & 1\end{bmatrix}\right)$
- The second distribution is a bivariate normal, $N\left(\begin{bmatrix}2\\0\end{bmatrix}, \begin{bmatrix}2 & 1\\1 & 1\end{bmatrix}\right)$
- The density of a mixture model with two components is defined as: $f_{mix}(x) = p \cdot f_1(x) + (1-p) \cdot f_2(x)$

The contour plot of the mixture model should look like this:


−0.01
−0.02
−0.03
−0.05
−0.07
−0.09
−0.08
−0.06
−0.04
−0.04
−0.06
−0.05
−0.03
−0.02
−0.01
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3