TMA4180 Optimization 1
Project 1: Unconstrained Optimization

Student numbers 767894, 767938, 767924.

February 2018

1 Introduction

In this project we will study a problem of fitting two different models to a set of data by applying two different optimization algorithms. Our focus will be on understanding the models and implementing the optimization algorithms effectively.

2 Answers to questions

Question 1

To solve this problem, we make use of the composition $(g \circ h_i)(\boldsymbol{x})$ where $h_i(\boldsymbol{x}) = \alpha_i(A, c)$, where $\alpha_i(A, c) = (z_i - c)^T A(z_i - c) - 1$ and

$$g(x) = \begin{cases} x^2 & : x > 0 \quad w_i > 0 \\ -x^2 & : x < 0 \quad w_i > 0 \\ 0 & : \text{otherwise} \end{cases} \tag{1}$$

This way, we get

$$f(\boldsymbol{x}) = \sum_{i=1}^{m} (g \circ h_i)(\boldsymbol{x}) = \sum_{i=1}^{m} g(h_i(\boldsymbol{x})) = \sum_{i=1}^{m} \begin{cases} \alpha_i(A, c)^2 & : \alpha_i(A, c) > 0, \quad w_i > 0 \\ -\alpha_i(A, c)^2 & : \alpha_i(A, c) < 0, \quad w_i < 0 \\ 0 & : \text{otherwise} \end{cases} \tag{2}$$

We are interested in the derivative, $\nabla f(\boldsymbol{x})$, and we want to show that this is continuous. Now $\nabla f(\boldsymbol{x}) = (df/dx_1, df/dx_2, \cdots, df/dx_{x(x+1)/2+n})$, which is continuous if df/dx_j is continuous for all $i = 1, 2, \cdots, n(n+1)/2 + n$. We therefore investigate df/dx_j, by performing the differentiation and applying the chain rule. We also introduce the variable change $h_i(\boldsymbol{x}) = \xi_i$:

$$\frac{df(\boldsymbol{x})}{dx_j} = \sum_{i=1}^{m} \frac{d}{dx_j} g(h_i(\boldsymbol{x})) = \sum_{i=1}^{m} \frac{dg(\xi_i)}{d\xi_i} \frac{d\xi_i}{dx_j} = \sum_{i=1}^{m} (g' \circ h_i)(\boldsymbol{x}) \cdot \frac{dh_i(\boldsymbol{x})}{dx_j} \tag{3}$$

We know that a sum of continuous functions is continuous, and products of continuous functions are also continuous. We therefore investigate the composition $(g' \circ h_i)(\boldsymbol{x})$ and $\frac{dh_i}{x_j}(\boldsymbol{x})$. We have

$$g'(x) = \begin{cases} 2x & : x > 0 \quad w_i > 0 \\ -2x & : x < 0 \quad w_i > 0 \\ 0 & : \text{otherwise} \end{cases} \tag{4}$$

Now, $2x$, $-2x$ and 0 are obviously continuous, and since we have $\lim_{x \to 0^+} 2x = \lim_{x \to 0^+} -2x = 0$ we see that $g'(\boldsymbol{x})$ is a continuous function for all x. We now take a closer look at the expression $(z_i - c)^T A(z_i - c)$. We use the notation $y_j = (z_{ij} - c_j)$, and get

$$y^T A y = [y_1, \ldots, y_n] \begin{bmatrix} a_{11} & \ldots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \sum_{j=1}^{n} y_j \sum_{k=1}^{n} y_k a_{jk}, \tag{5}$$

which is a polynomial of degree 1 with respect to a_{jk}, and a polynomial of degree 2 with respect to y_j, which implies that it is a polynomial of degree 2 with respect to $c_j = (z_{ij} - y_j)$. We know polynomials are continuous and infinitely contunuously differentiable, which means that $\alpha_i(A, c) = (y^T Ay - 1)$ is a continuous function with respect to A, c. Furthermore, we know $h_i(\boldsymbol{x})$ is a polynomial of degree 1 or 2 with respect to x_j, which means that both $h_i(\boldsymbol{x})$ and $\frac{dh_i(\boldsymbol{x})}{dx_j}$ are continuous functions. Also, we know that a composition of continuous functions are continuous, which means that $(g' \circ h_i)(\boldsymbol{x})$ is continuous. Thus, we see that $df(\boldsymbol{x})/dx_j = \sum_{i=1}^{n} (g' \circ h_i)(\boldsymbol{x}) \cdot \frac{dh_i(\boldsymbol{x})}{dx_j}$ is a continuous function, which is what we wanted to show. We also want to show that $\hat{f}(\boldsymbol{x})$, the function for model 2, also is once continuously differentiable. This proof is very similar. We use the same composition as before with $\hat{g}(x) = g(x)$ and $\hat{h}_i(\boldsymbol{x}) = \hat{\alpha}_i(A, b)$ where $\hat{\alpha}_i(A, b) = z^T Az + b^T z - 1$. Since $\hat{g}(x) = g(x)$, we must also have $\hat{g}'(x) = g'(x)$, which is a continuous function. Also, if we take a closer look at $z^T Az + b^T z - 1$, we get

$$\begin{aligned} z_i^T Az_i + b^T z_i &= [z_{i1}, \dots, z_{in}] \begin{bmatrix} a_{11} & \dots & a_{1n} \\ \vdots & \ddots & \vdots \\ a_{n1} & \dots & a_{nn} \end{bmatrix} \begin{bmatrix} z_{i1} \\ \vdots \\ z_{in} \end{bmatrix} + [b_1, \dots, b_n] \begin{bmatrix} z_{i1} \\ \vdots \\ z_{in} \end{bmatrix} \\ &= \sum_{j=1}^{n} z_{ij} \sum_{k_1}^{n} z_{ik} a_{jk} + \sum_{j=1}^{n} b_j z_{ij}. \end{aligned} \tag{6}$$

This is a polynomial of degree 1 with respect to a_{jk} and b_j, which means that $\hat{\alpha}_i(A, b)$ is a continuous function, implying that $\hat{h}_i(\boldsymbol{x}) = \hat{\alpha}_i(A, b)$ is a continuous function. Also, $h_i(\boldsymbol{x})$ is a polynomial of degree 1 with respect to x_j. Therefore $h_i(\boldsymbol{x})$ and $dh_i(\boldsymbol{x})/dx_j$ must also be continuous functions. Since the composition of continuous functions $(\hat{g}' \circ h_i)(\boldsymbol{x})$ must be continuous, we can conclude that also the function $d\hat{f}(\boldsymbol{x})/dx_j = \sum_{i=1}^{n} (\hat{g}' \circ h_i)(\boldsymbol{x}) \cdot \frac{d\hat{h}_i(\boldsymbol{x})}{dx_j}$ is continouous.

We further want to investigate whether the double derivative of f and $\hat{f}$ is continuous as well. We then get the Hessian matrix

$$\nabla^2 f = \begin{bmatrix} \frac{d^2 f}{dx_1^2} & \frac{d^2 f}{dx_1 dx_2} & \cdots & \frac{d^2 f}{dx_1 dx_n} \\ \frac{d^2 f}{dx_2 dx_1} & \frac{d^2 f}{dx_2^2} & \cdots & \frac{d^2 f}{dx_2 dx_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{d^2 f}{dx_n dx_1} & \frac{d^2 f}{dx_n dx_2} & \cdots & \frac{d^2 f}{dx_n^2} \end{bmatrix} \tag{7}$$

For this to be continuous, all entries must be continuous. We use the same change of variable, $h_i(\boldsymbol{x}) = \xi_i$, together with the chain rule, to get

$$\begin{aligned} \frac{d^2 f(\boldsymbol{x})}{dx_j dx_k} &= \sum_{i=1}^{n} \frac{d}{dx_k} \left(\frac{dg(\xi_i)}{d\xi_i} \cdot \frac{d\xi_i}{dx_j} \right) = \sum_{i=1}^{n} \left(\frac{d^2 g(\xi_i)}{d\xi_i^2} \cdot \frac{d\xi_i}{dx_k} \frac{d\xi_i}{dx_j} + \frac{d^2 \xi_i}{dx_j dx_k} \cdot \frac{dg(\xi_i)}{d\xi_i} \right) \\ &= \sum_{i=1}^{n} \left((g'' \circ h_i)(\boldsymbol{x}) \cdot \frac{dh_i(\boldsymbol{x})}{x_j} \cdot \frac{dh_i(\boldsymbol{x})}{x_k} + \frac{d^2 h_i(\boldsymbol{x})}{x_j x_k} \cdot (g' \circ h_i)(\boldsymbol{x}) \right) \end{aligned} \tag{8}$$

Now we expect derivatives of $h(\boldsymbol{x})$ with respect to x_j to be continuous functions, but our problem is with $g''(x)$. This is given by

$$g''(x) = \begin{cases} 2 & : x > 0 \quad w_i > 0 \\ -2 & : x < 0 \quad w_i > 0 \\ 0 & : \text{otherwise}, \end{cases} \tag{9}$$

which is not a continuous function. Thus, we can not expect the composition $(g'' \circ h_i)(\boldsymbol{x})$ to be continuous, and in general, as a result, we can not expect $f(\boldsymbol{x})$ to be twice continuously differentiable. For $\hat{f}(\boldsymbol{x})$, we get a similar equation to equation (8), and since we have $\hat{g}(x) = g(x)$, we get a function $\hat{g}''(x) = g''(x)$ which is not continuous. Therefore, we can't expect $\hat{f}(\boldsymbol{x})$ to be twice continuously differentiable. Finally, we want to show that $\hat{f}(\boldsymbol{x})$ has a gradient which is lipschitz continuous.

Generally, a function has a lipschitz continuous gradient if

$$||\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y})|| \leq L||\boldsymbol{x} - \boldsymbol{y}||. \tag{10}$$

We have from equation (3)

$$\begin{aligned}\nabla f(\boldsymbol{x}) - \nabla f(\boldsymbol{y}) &= \sum_{i=1}^{m} \nabla(\hat{g} \circ \hat{h}_i)(x) - \nabla(\hat{g} \circ \hat{h}_i)(y) \\ &= \sum_{i=1}^{m} (\hat{g}' \circ \hat{h}_i)(x) \cdot \begin{bmatrix} \frac{dh_i(\boldsymbol{x})}{x_1} \\ \vdots \\ \frac{dh_i(\boldsymbol{x})}{x_{n(n+1)/2+n}} \end{bmatrix} - (\hat{g}' \circ \hat{h}_i)(y) \cdot \begin{bmatrix} \frac{dh_i(\boldsymbol{y})}{y_1} \\ \vdots \\ \frac{dh_i(\boldsymbol{y})}{y_{n(n+1)/2+n}} \end{bmatrix}\end{aligned} \tag{11}$$

We know from equation (6), and the discussion afterwards, that $dh_i(\boldsymbol{x})/dx_j$ is a linear function with respect to x_j. Thus, $dh_i'(\boldsymbol{x})/dx_j$ is a constant, and the vector in the expression above only contains constant elements. The vectors as functions of $\boldsymbol{x}$ and $\boldsymbol{y}$ are therefore identical, and we denote this vector by $\boldsymbol{K}_i$. Also, we know the function $\hat{g}(x)$ is lipschitz since $g'(x) - g'(y) = 2(x - y)$. We also know that $\hat{h}_i(\boldsymbol{x})$ is a linear function with respect to x_j. There must therefore exist a number T so that $|\hat{h}(\boldsymbol{x}) - \hat{h}(\boldsymbol{y})| \leq T|\boldsymbol{x} - \boldsymbol{y}|$. Thus, the composition $(\hat{g}' \circ \hat{h}_i)(\boldsymbol{x})$, is a composition of two lipschitz functions, which implies that the composition is lipschitz as well. Thus we get

$$\begin{aligned}||\nabla \hat{f}(\boldsymbol{x}) - \nabla \hat{f}(\boldsymbol{y})|| &= ||\sum_{i=1}^{m} \boldsymbol{K}_i\big((\hat{g}' \circ \hat{h}_i)(x) - (\hat{g}' \circ \hat{h}_i)(y)\big)|| \\ &\leq \sum_{i=1}^{m} ||\boldsymbol{K}_i\big((\hat{g}' \circ \hat{h}_i)(x) - (\hat{g}' \circ \hat{h}_i)(y)\big)|| \\ &\leq m \cdot \max_i \Big(||\boldsymbol{K}_i|| \cdot ||(\hat{g}' \circ \hat{h}_i)(x) - (\hat{g}' \circ \hat{h}_i)(y)|| \Big) \\ &\leq L||(\boldsymbol{x} - \boldsymbol{y}||\end{aligned} \tag{12}$$

Thus we have show that $\hat{f}(\boldsymbol{x})$ has a lipschitz continuous gradient.

Question 2

There will certainly exist a globally optimal solution that is not unique if there is a way to separate the points 100% correctly. One way to achieve this is if there exists an x such that $f(x) = 0$ for Model 1 and $\hat{f}(x) = 0$ for Model 2. Now imagine a 2D case with two points: One point with positive weight at $(0, 0)$, and another point at $(1, 0)$ with negative weight, as shown in figure 1.

In this case, there are several ellipses that include the first point and omit the second (for example all circles centered around Origo with radius less than 1), so the globally optimal solution exists but is not unique. This example holds for both models, since Model 2 can choose ellipses centered around Origo, and Model 1 can choose any ellipse.

A function $f : \mathbb{R}^n \to \mathbb{R}$ is coercive if for every sequence $(x_k)_{k \in \mathbb{N}}$

$$\lim_{\|x_k\| \to \infty} f(x_k) \to \infty.$$

So for a non-coercive function, the function value will not diverge to infinity when $\|x_k\| \to \infty$.

In our case, a non-coercive function corresponds to a sequence in x (i.e. the elements in matrix A and vector $\boldsymbol{b}$ or $\boldsymbol{c}$ respectively, depending on the model), where the norm of the sequence diverges, but the sum of the squared residuals converges. In our optimization task, we might want the residuals go to zero to show a non-coercive example. Our aim here is to explore all the ways in which the norm of x might diverge.

If we let all x_i in (x_k) corresponding to the diagonal in the A-matrix describing the ellipsoid go to infinity and keep the other $x_i = 0$, we get the following equation for the value of the residuals:

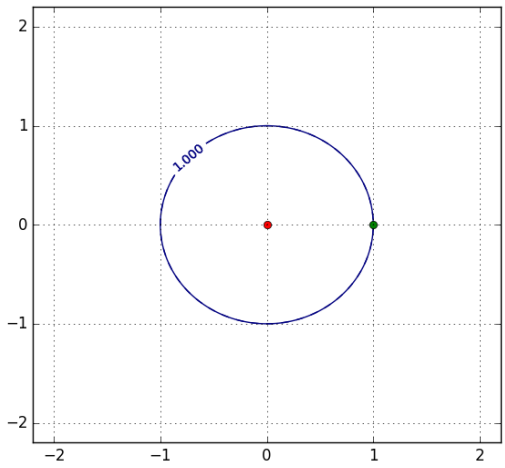

Figure 1: Not unique globally optimal solution for positive and negative point.

$$r_i(A, \mathbf{0}) = \begin{cases} \max(_{\lim \|a\| \to \infty} \sum_{j=1}^{n} z_{ij}^2 a - 1, 0) & : w_i > 0 \\ \max(1 - _{\lim \|a\| \to \infty} \sum_{j=1}^{n} z_{ij}^2 a, 0) & : w_i \leq 0, \end{cases} \tag{13}$$

where a is the value of the diagonals in A.

For (13) to be zero for points with positive weights, we need to have that all the points are in the centre, $z_i = \mathbf{0}$, otherwise the sum diverges to infinity and so do the residuals.

For the points with negative weights, either being in the centre or outside the centre, they will never make r_i diverge. If the points are outside of centre, the residuals will be zero, while being in the centre results in $r_i(A, \mathbf{0}) = 1$, for each point.

The same results therefore hold for model 2, with $\hat{r}_i = 0$ for points with positive weights placed in the centre. For any value of $\boldsymbol{b}$, the centre will still be included inside the ellipsoid and the residuals must be zero, since inserting $z_i = 0$ into $\hat{\alpha}_i(A, b)$ yields $max(0 + 0 - 1, 0) = 0$. Taking a look at the geometric interpretation of this, increasing the diagonal of A results in making a smaller ellipsoid. In the previous part described and the task in general, we had a given dataset with m points, where we changed A and $\boldsymbol{c}$ or $\boldsymbol{b}$, i.e. $\boldsymbol{x}$, to minimise the value of the residuals.

For the ellipsoid described by

$$\boldsymbol{y}^T A \boldsymbol{y} = 1,$$

where A is fixed and $\boldsymbol{y}$ is the variable, we see the connection between the increase of values of the diagonals in A and the ellipsoid shrinking.

For $\text{diag}(A) = a$ and the off-diagonals to be zero, we get

$$\begin{aligned} \boldsymbol{y}^T A \boldsymbol{y} &= 1 \\ \sum_{j=1}^{n} y_j^2 a &= 1 \\ y_1^2 + y_2^2 + ... + y_n^2 &= 1/a, \end{aligned}$$

which gives $\boldsymbol{y}$ laying on a circle with radius $\sqrt{1/a}$, centred in 0. Letting a go to infinity, the radius approaches zero, and we get as expected

Question 3

If we define $\hat{f}(x)$ to be a composition of two functions, where the first function is convex and the second is affine, then it is possible to show that $\hat{f}(x)$ itself is convex too. Let $\hat{f}(x) = \sum_{i=1}^{m} g(h(x))$, where

$g(x) = \max(h(x), 0)^2$ and $h(x) = \hat{\alpha}_i(A, b)$ for $w_i > 0$ and $h(x) = \hat{\beta}_i(A, b)$ for $w_i < 0$. To show that $g(x)$ is convex, (14) must be fulfilled with $\lambda \in [0, 1]$.

$$g(\lambda x_1 + (1-\lambda)x_2) \leq \lambda g(x_1) + (1-\lambda)g(x_2) \tag{14}$$

For the left side of the equation,

$$\begin{aligned} g(\lambda x_1 + (1-\lambda)x_2) &= \max(\lambda x_1 + (1-\lambda)x_2, 0)^2 && (15)\\ &= \max((\lambda x_1 + (1-\lambda)x_2)^2, 0) && (16)\\ &= (\lambda x_1 + (1-\lambda)x_2)^2 && (17) \end{aligned}$$

where the last equation holds since $(\lambda x_1 + (1-\lambda)x_2)^2$ always will be positive for $x \in \mathbb{R}^{n(n+1)/2+n}$. For the right side,

$$\begin{aligned} \lambda g(x_1) + (1-\lambda)g(x_2) &= \lambda \max(x_1, 0)^2 + (1-\lambda)\max(x_2, 0)^2 && (18)\\ &= \lambda \max(x_1^2, 0) + (1-\lambda)\max(x_2^2, 0) && (19)\\ &= \lambda x_1^2 + (1-\lambda)x_2^2, && (20) \end{aligned}$$

We now want to check that (17) $\leq$ (20).

$$\begin{aligned} (\lambda x_1 + (1-\lambda)x_2)^2 &\leq \lambda x_1^2 + (1-\lambda)x_2^2\\ 0 &\leq -\lambda^2 x_1^2 - 2\lambda(1-\lambda)x_1 x_2 - (1-\lambda)^2 x_2 + \lambda x_1^2 + (1-\lambda)x_2^2\\ 0 &\leq x_1^2(\lambda - \lambda^2) - 2\lambda(1-\lambda)x_1 x_2 + x_2^2((1-\lambda) - (1-\lambda)^2)\\ 0 &\leq x_1^2\lambda(1-\lambda) - 2\lambda(1-\lambda)x_1 x_2 + x_2^2\lambda(1-\lambda)\\ 0 &\leq x_1^2 - 2x_1 x_2 + x_2^2\\ 0 &\leq (x_1 - x_2)^2, \end{aligned}$$

which is always true for $\boldsymbol{x} \in \mathbb{R}$. We have now shown that $g(x) = \max(h(x), 0)^2$ is convex.

The next part is to show that $h(x)$ is affine, where an affine function only consists of linear or constant parts. Combining (5) and $\boldsymbol{b}^T z_i = \sum_{j=1}^{n} b_j z_{ij}$:

$$\hat{h}(\boldsymbol{x}) = \begin{cases} \hat{\alpha}_i(\boldsymbol{x}) = \sum_{j=1}^{n}\sum_{k=1}^{n} z_{ij} z_{ik} a_{jk} + \sum_{j=1}^{n} b_j z_{ij} - 1 & : w_i > 0 \\ \hat{\beta}_i(\boldsymbol{x}) = 1 - \sum_{j=1}^{n}\sum_{k=1}^{n} z_{ij} z_{ik} a_{jk} - \sum_{j=1}^{n} b_j z_{ij} & : w_i \leq 0, \end{cases} \tag{21}$$

with only a_{jk} and bj dependent on $\boldsymbol{x}$. Thus we have shown that $h(\boldsymbol{x})$ is affine.

Now, since we know that $g(\boldsymbol{x})$ is convex and $h(\boldsymbol{x})$ is affine, we only need to show that these two combined gives $\hat{f}(\boldsymbol{x})$ convex, which means fulfilling (22):

$$f(\lambda x_1 + (1-\lambda)x_2) \leq \lambda f(x_1) + (1-\lambda)f(x_2) \tag{22}$$

Using (14), (21), and introducing γ_i and k as constants in $h(\boldsymbol{x})$,

$$\begin{aligned} f(\lambda x_1 + (1-\lambda)x_2) &= \sum_{i=1}^{n} g(\lambda h(x_1) + (1-\lambda)h(x_2))\\ &= \sum_{i=1}^{n} g(\lambda(k_1 + \sum_{j=1}^{n}\gamma_{ij}x_1) + (1-\lambda)(k_2 + \sum_{j=1}^{n}\gamma_{ij}x_1))\\ &= \sum_{i=1}^{n} \lambda g(k_1 + \sum_{j=1}^{n}\gamma_{ij}x_1) + \sum_{i=1}^{n}(1-\lambda)g(k_2 + \sum_{j=1}^{n}\gamma_{ij}x_1)\\ &= \lambda f(x_1) + (1-\lambda)f(x_2) \end{aligned}$$

Thus, $\hat{f}(\boldsymbol{x})$ is convex.

Question 4

The code evaluating the function values and gradients was implemented and tested against a finite-element approximation. The following code output shows the steplength *ep* used in the finite-element approximation and the relative error of the directional derivative computed by our function compared to the finite-element approximation with the given steplength, for model 1.

```
ep = 1.000000e-01, error = 1.076066e+00
ep = 1.000000e-02, error = 1.036672e-01
ep = 1.000000e-03, error = 1.031714e-02
ep = 1.000000e-04, error = 1.031208e-03
ep = 1.000000e-05, error = 1.031157e-04
ep = 1.000000e-06, error = 1.031175e-05
ep = 1.000000e-07, error = 1.027675e-06
ep = 1.000000e-08, error = 1.598112e-08
ep = 1.000000e-09, error = 7.996875e-07
ep = 1.000000e-10, error = 9.349212e-06
ep = 1.000000e-11, error = 2.943334e-04
ep = 1.000000e-12, error = 1.130587e-03
```

We see the error converges to zero as the finite element approximation uses smaller steps. This indicates that the functon that calculates the function and derivative for model 1 is correct, since the approximated solution converges to the calculated value for $\nabla f(\boldsymbol{x})$. We also see when the finite-element approximation becomes even more exact, the error eventually starts to grow. Since the approximated solution involves very small numbers, which are less exact for float-numbers. When we in addition divide by these very small numbers, some numerical phenomenon might occur. The following listing shows the same test for the functions corresponding to model 2.

```
ep = 1.000000e-01, error = 2.093010e-02
ep = 1.000000e-02, error = 2.093010e-03
ep = 1.000000e-03, error = 2.093010e-04
ep = 1.000000e-04, error = 2.093010e-05
ep = 1.000000e-05, error = 2.092979e-06
ep = 1.000000e-06, error = 2.089854e-07
ep = 1.000000e-07, error = 2.112508e-08
ep = 1.000000e-08, error = 6.339614e-08
ep = 1.000000e-09, error = 4.744874e-08
ep = 1.000000e-10, error = 7.499213e-07
ep = 1.000000e-11, error = 2.955014e-05
ep = 1.000000e-12, error = 2.209189e-04
```

Again, we see a very similar result, indicating that the functions for model 2 also are correct.

Question 5

To solve the least square problems we implemented one Conjugate Gradient Method (Fletcher Reeves) and one Quasi-Newton Method (BFGS). First, let's look at plots where the function value to be minimized is plotted for each iteration.

Figures 2 and 3 show the BFGS iteration scheme for 100 different sets of points z, for Model 1 and Model 2 respectively, where the points z are classified by an ellipse so that there exists at least one solution with 100% correct classification. Similarly, figures 4 and 5 show 100 realizations of the Fletcher Reeves iteration for both models, again classifying the points by an ellipse so that there exists at least one solution with 100% correct classification. With either model, and for both iteration schemes, the iteration usually converges within 10-12 iterations, and the performance of the algorithms does not vary much between different sets of points. The tolerance used here is 10^{-3} for the norm of the gradient in the stopping point.

Next, let's look at how the number of iterations scale with the number of points in the model. The figures 6, 7 and 8 show how the models perform for different number of data points z_i, averaged over respectively 100, 50 and 20 realizations, for points classified respectively by ellipse, rectangle, and ellipse with 5% misclassification.

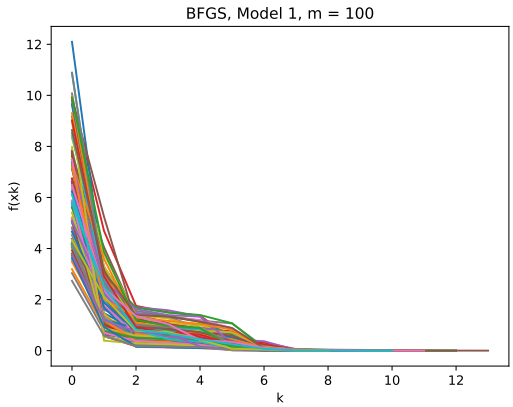

Figure 2: BFGS using Model 1 on 100 different sets of points classified by ellipse.

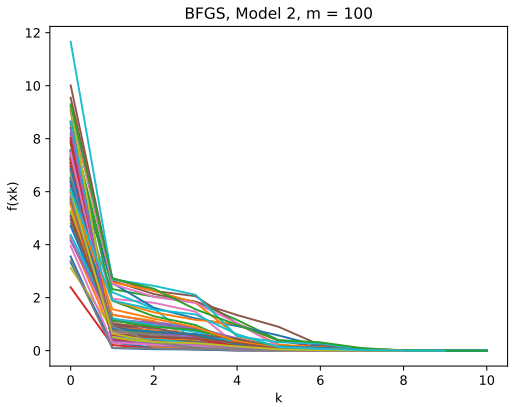

Figure 3: BFGS using Model 2 on 100 different sets of points classified by ellipse.

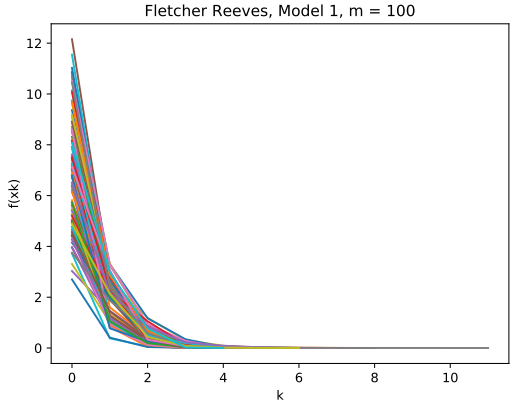

Figure 4: Fletcher Reeves using Model 1 on 100 different sets of points classified by ellipse.

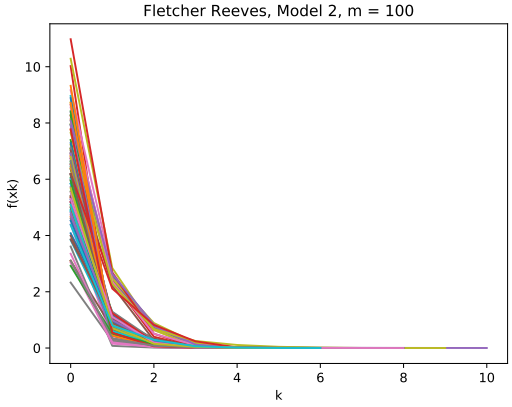

Figure 5: Fletcher Reeves using Model 2 on 100 different sets of points classified by ellipse.

For ellipse classified points, Fletcher Reeves seems to converge faster than BFGS. For points classified by a rectangle Fletcher Reeves still looks to be scoring marginally better. But for points classified by ellipses with 5% misclassification, BFGS converges considerably faster, and therefore looks like the more robust algorithms in harder cases. Here, we are mystified as to what is happening for m around 50, where especially Fletcher Reeves needs way more iterations than elsewhere. This plot shows the number of iterations averaged over 20 simulation (the low number due to a long runtime), so the peak might be due to one particular set of points generated for this m value. The abnormal runtime is still disconcerting.

From figure 8 we deduce that Model 2 converges faster than Model 1 for points with partial misclassification, independent of which optimization algorithm is being used. The optimization algorithms needed considerably more time to solve the cases where the z points were classified by an ellipse.

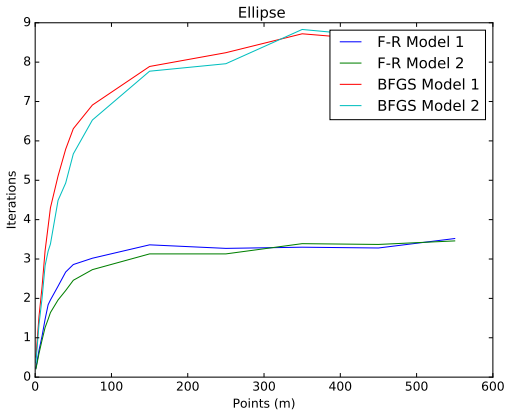

Figure 6: Number of iterations plotted against number of points m, where the dataset is classified by an ellipse.

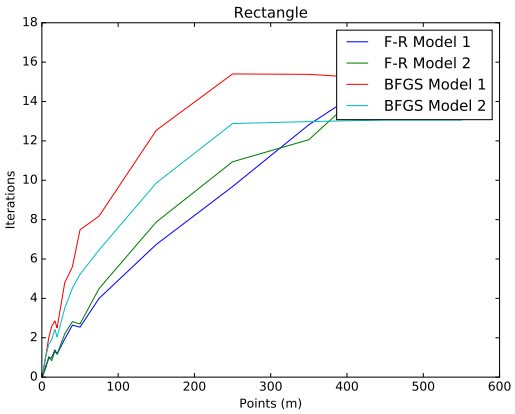

Figure 7: Number of iterations plotted against number of points m, where the dataset is classified by rectangle.

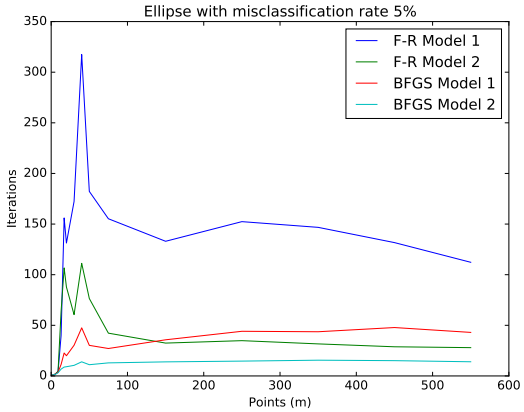

Figure 8: Number of iterations plotted against number of points m, where the dataset is classified by an ellipse with misclassification rate 0.05.

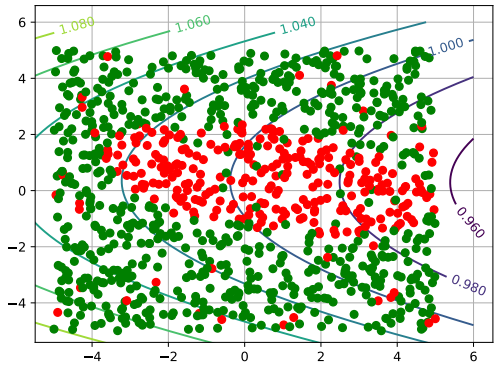

Figure 9: Plot showing BFGS converging to an alternative solution. Since most points are on a contour close to 1, $\alpha_i \sim 0$ for all i.

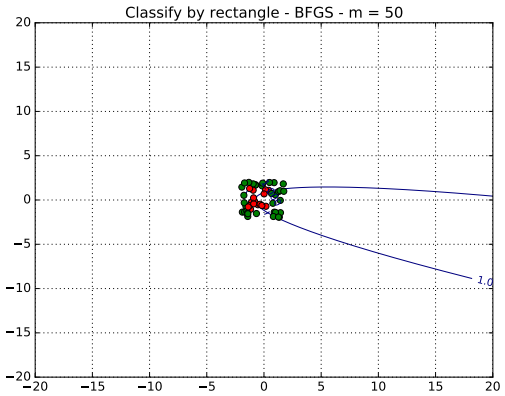

Figure 10: Classifying by rectangle.

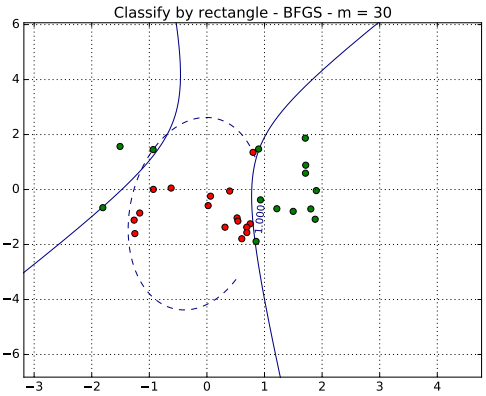

Figure 11: Classifying by rectangle.

3 Test problems

To easily check whether the algorithms are doing great or not, test problems can be made. We made three test cases, the first constructing data points from an existing ellipse where the location of the z_i determines it's weight, w_i. Then, the algorithm runs on the dataset, and we see if the algorithm produces a good solution, an x-vector corresponding to A-matrix and **c**for model 1 and **b** for model 2. This is plotted in figure 12 and 13. We have also used rectangles for classifying the weights. Here, the solutions are shown in figure 14 and 15. We see that all algorithms produce good solutions here too, but perfect solutions might not be posssible now, which results in some misclassifications. Finally, we have used ellipsoids with a probability of misclassification to determine weights. This is shown in figure 16 and 17. An interesting observation here is that the misclassified points have a quite large impact on the solution, resulting in many misclassified points. The reason for this is probably because the residuals are squared, resulting in larger error for points further away. Thus, more points with smaller error results in a better solution than few points with bigger error.

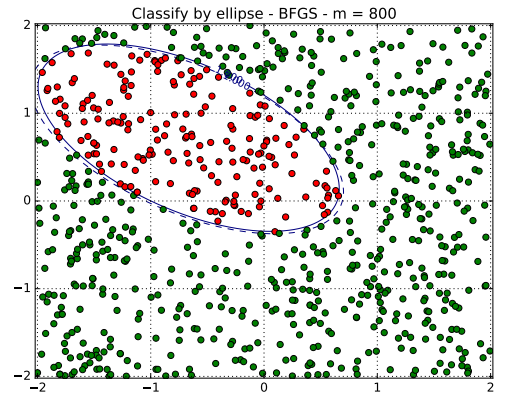

Figure 12: Plotting the countourline = 1. BFGS with ellipse.

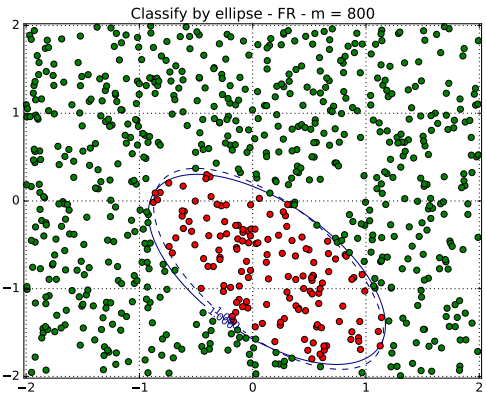

Figure 13: Plotting the countourline=1. Fletcher Reeves with ellipse

Notes on large ill-fitting ellipses with Model 1

Convexity was only proved for Model 2. When analysing the performance of model 1, we encountered large run times for some initial guesses for x. We have not looked fully into this, but want to make some notes on our thoughts.

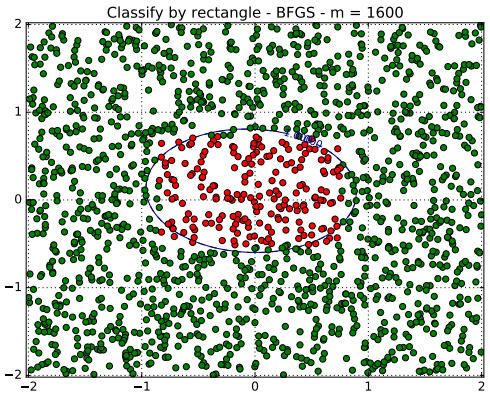

Figure 14: Plotting the countourline=1. BFGS with rectangle.

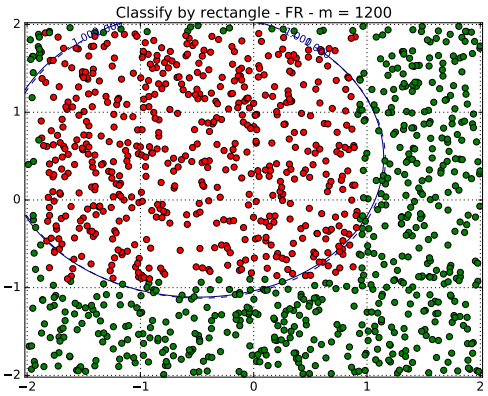

Figure 15: Plotting the countourline=1. Fletcher Reeves with rectangle.

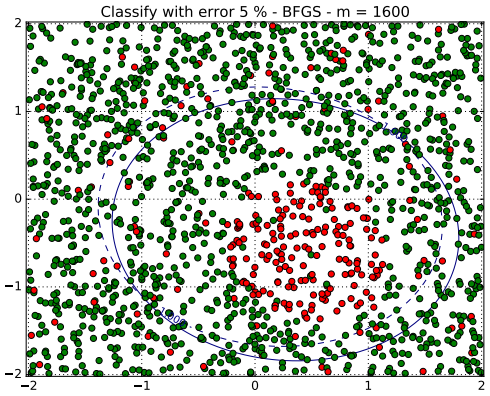

Figure 16: Plotting the countourline=1. BFGS with misclassification rate 0.05

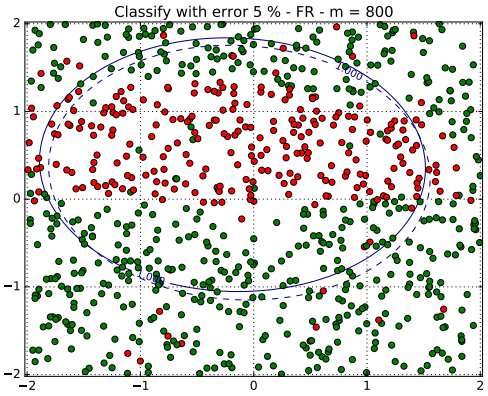

Figure 17: Plotting the countourline=1. Fletcher Reeves with misclassification rate 0.05

The equation in model 1 describes a scalar field for example in two dimensions. We are using the contour line where the scalar field equals 1 to decide wether we include or exclude points in our set. However, what we are trying to minimize is not the sum of the distance of misclassified points to our contour line, but but rather the difference of the value of the scalar field at misclassified points compared to the contour line. The distance between the contours grow bigger when the size of the ellipse grows large. Therefore, a large ellipse with centre far away from Origo, but contour line with value 1 running through Origo, will have small difference in the value of the scalar field around Origo, where our points are located. Figure 9 illustrates this. (We would like to note that for Method 2, the contour lines end up being close to each other around zero anyway even though we shift the ellipse so that Origo is close to its edge. In all cases it seems like model 2 is unable to get smaller residuals this way, so we dont get this effect with model 2. We have not looked too deep into this). As a result, the optimization algorithm might want to make the elements in the matrix A are very small, and make the elements in c grow large, since then all the z points will be placed on contours fairly close to 1. As a result, $y_i^T A y_i \sim 1$, giving small residuals, which in theory can converge to zero. Thus, if there is no exact solution for a given set of points, the local solution we are interested in will not be the best one. The value of f will therefore be small, quite independent of whether the points are classified correctly or not. So $f(x)$ may keep diminishing as the entries of A decrease, sending the iteration away from a good classification. So the iteration may move towards some large ellipse with low $f(x)$ but rather bad classification. With initial guesses closer to a "good" solution we can avoid this, but this is not explored any further here. Figure 10 and 11 shows other cases where the initial guess is poor. This might result in odd solutions.