STAT 548 REPORT 2

“Self-concordance for empirical likelihood”

Art B. Owen, July 2012

Submitted to
DR. JIAHUA CHEN
December 1, 2017

SAIFUDDIN SYED
85258144

Contents

1 Introduction

The likelihood method for parameter estimation and inference lies at the foundation of statistics. We assume our data is $X_1, X_2, \dots$ is drawn from $F \in \mathcal{F}$, where $\mathcal{F}$ is a family of distributions parametrized by some Θ. The likelihood function, $L(X|\theta)$ is a measure of how likely we are to observing our data given the samples are drawn form $\theta \in \Theta$. We can then use this function to generate point estimates such as the MLE, perform inference, generate confidence regions and perform hypothesis testing. Asymptotically these methods have some really properties that make them very useful.

The problem however with these methods is that we must specify a parametric model and the quality of our analysis is very dependent on how good $\mathcal{F}$ is at representing our data. If our model is not a good approximation of our data, then we run into the trouble of making potentially very inaccurate statistic claims.

Owen [Owe88] proposed a non-parametric alternative to the tradition likelihood method by introducing the empirical likelihood (EL) function. This method, allows us to perform inference and retain many of the cherished theoretical asymptotic properties of the traditional likelihood method without having to specify a specific model for our data. Since his enaugaral paper, emperical likelihood methods have become very prevalent in many applied areas such as economics and finance, (e.g. [Bra07] [TAOT14]) and has become an active area of research. However, unlike the simplicity of the computation of the parametric likelihood function, the emperical likelihood function is defined implicitly via a constrained optimization problem and requires some fineness to compute, when defined.

For the remainder of this report, we will give an overview of the empirical likelihood (EL) method. We will then give explore the new method proposed by Owen in [Owe13] to compute the emperical likelihood using a property called self-concordance to achieve a provably bound on the sub-optimalitity of the procedure. We will then compare compare and contrast to the alternative method proposed by Chen in [CVA08], called the adjusted emperical likelihood (AEL) that approximates the EL by adjusting the EL constraints that ensures it is globally well defined and retains the asymptotic properties of the EL. Finally we test both methods on some simulated data.

2 Emperical Likelihood

Suppose we have an i.i.d sample of data $X_1, \dots, X_n \in \mathbb{R}^d$ that is drawn from true distribution F_0. Our goal is to perform a discrete approximate F_0 by some distribution F supported on the observed values $X_1, \dots, X_n$.

Let $p_i \equiv P_F(x_i)$ be the probability of observing $X_i = x_i$ under F, since F is supported on our data $p_i \geq 0$ and $\sum_{i=1}^n p_i = 1$. We define the empirical likelihood of F as

$$L(F) = \prod_{i=1}^n p_i,$$

where p_i is the probability under F of observing X_i. It will be convenient to work with the log-empirical likelihood function $\ell(F)$ as

$$\ell(F) = \sum_{i=1}^{n} \log p_i.$$

So far the naive thing to do would be to find F such that $L(F)$ or equivalently $\ell(F)$ is optimize. This turns out to be short sighted, as ℓ is trivially optimized by the empirical distribution F_n, that assigned each observation equal probability, i.e., $P_{F_n}(x_i) = 1/n$. Indeed, this can be seen by a simple consequence of Jensens inequality. Suppose F is any distribution supported on our data.

$$\begin{aligned}
\log \frac{L(F)}{L(F_n)}) &= \log \prod_{i=1}^{n} \frac{P_F(x_i)}{P_{F_n}(x_i)} \\
&= \sum_{i=1}^{n} \log(np_i) \\
&= n \sum_{i=1}^{n} \frac{1}{n} \log(np_i) \\
&\leq n \log \left(\sum_{i=1}^{n} \frac{1}{n} np_i \right) \\
&= n \log \left(\sum_{i=1}^{n} p_i \right) \\
&= 0.
\end{aligned}$$

The inequality was using Jensen's inequality on the concave function log. Thus we have $L(F) \leq L(F_n)$. This naturally leads us to define the empirical likelihood ratio $R(F)$ to be the ratio $L(F)$ relative to maximum of L. I.e.

$$R(F) = \frac{L(F)}{L(F_n)} = \prod_{i=1}^{n} np_i.$$

We have $0 \leq R(F) \leq 1$, is a measure of how how likely our approximation F is to F_0 compared to the optimal F_n. Up until this point R, is not very useful beyond telling us how poor our model is compared to the trivial F_n.

2.1 Profile Empirical Likelihood

In practice we often have parameters $\theta \in \Theta \in \mathbb{R}^p$, where the true parameter θ_0 is known to satisfy

$$\mathbb{E}_{F_0}(m(X, \theta_0)) = 0$$

where $m : \mathbb{R}^d \times \mathbb{R}^p \to \mathbb{R}^q$ is the called the estimating equations of our model. $m(X, \theta)$ represents the constraints in our model that we wish our data to satisfy, at least on average. Given $\theta \in \Theta$ we will define the profile empirical likelihood function $\mathcal{R}(\theta)$ as the maximum of the ELR given our F satisfies the estimating equation where the expectation is taken with respect to F.

$$\begin{aligned}\mathcal{R}(\theta) &= \sup\{R(F) : \mathbb{E}_F(m(X,\theta)) = 0\} \\ &= \sup\left\{\prod_{i=1}^n np_i : p_i \geq 0, \sum_{i=1}^n p_i = 1, \sum_{i=1}^n p_i m(x_i,\theta) = 0\right\}.\end{aligned}$$

Although, the following analysis will hold true for general m, we will focus our attention to the case where $p = d$ and $m(X,\theta) = X - \theta$. I.e. we are interested in the profile empirical likelihood function for the mean. Given $\mu \in \mathbb{R}^d$, define the profile empirical likelihood function for the mean as

$$\mathcal{R}(\mu) = \sup\left\{\prod_{i=1}^n np_i : p_i \geq 0, \sum_{i=1}^n p_i = 1, \sum_{i=1}^n p_i x_i = \mu\right\}.$$

As before it will be useful to work with the $\log \mathcal{R}(\mu)$,

$$\log \mathcal{R}(\mu) = \sup\left\{\sum_{i=1}^n \log(np_i) : p_i \geq 0, \sum_{i=1}^n p_i = 1, \sum_{i=1}^n p_i x_i = \mu\right\}. \tag{1}$$

Note we can take the log inside the supremum since log is monotone. We will now discuss how to solve this convex optimization problem.

Remark 2.1. Before we begin computing $\log \mathcal{R}(\mu)$, note that the third constraint in (1) is only true if and only if μ can be written as a convex combination of our data, if and only if μ is in the convex hull generated by $X_1,\ldots,X_n$, we will denote as $\mathrm{Conv}(X)$.

If $\mu \in \partial\mathrm{Conv}(X)$, then there must be an i such that $p_i = 0$, and thus $\mathcal{R} = 0$. If μ is in the interior of the convex hull, denoted by $\mathrm{int}(\mathrm{Conv}(X))$, there exists $p_i > 0$ for all i such that $\sum_{i=1}^n p_i X_i = \mu$. Therefore a finite solution exists to (2) if and only if $\mu \in \mathrm{int}(\mathrm{Conv}(X))$, in which case the maximizing p_i must be non-zero. Therefore we will now assume that $\mu \in \mathrm{int}(\mathrm{Conv}(X))$ and $p_i > 0$.

2.2 Computation of the Profile log Empirical Likelihood

We will now go over the standard procedure for solving computing $\log \mathcal{R}(\mu)$ as outlined in [Owe01], [Owe13] by reformulating the (1) as the optimization problem,

$$\begin{aligned}\text{maximize:} \quad & \sum_{i=1}^n \log(np_i) \\ \text{subject to:} \quad & p_i > 0, \\ & \sum_{i=1}^n p_i = 1, \\ & \sum_{i=1}^n p_i X_i = \mu, \quad \mu \in \mathrm{int}(\mathrm{Conv}(X))\end{aligned} \tag{2}$$

To solve this constrained optimization problem, we employ the method of Lagrange multipliers. We define the Lagrangian for $G(p, \lambda, \delta)$ as,

$$G(p, \lambda, \delta) = \sum_{i=1}^{n} \log(np_i) + \delta(\sum_{i=1}^{n} p_i - 1) - \lambda^T \sum_{i=1}^{n} p_i Z_i, \tag{3}$$

where $-n\lambda \in \mathbb{R}^d$ and $\delta \in \mathbb{R}$ are Lagrange multipliers, and $Z_i = X_i - \mu$. We find the critical point of G by taking the gradient of (3) with respect to p to get

$$\frac{\partial G}{\partial p_i} = \frac{1}{p_i} + \delta - n\lambda^T Z_i.$$

We define $\hat{p}_i, \hat{\delta}, \hat{\lambda}$ to be the solution to $\nabla G = 0$. This leads to the following system of equations.

$$\frac{1}{\hat{p}_i} + \hat{\delta} - n\hat{\lambda}^T Z_i = 0. \tag{4}$$

We now multiply (4) by $\hat{p}_i$ and sum over i to get.

$$\begin{aligned} 0 &= \sum_{i=1}^{n} (1 + \hat{\delta}\hat{p}_i - n\hat{\lambda}^T \hat{p}_i Z_i) \\ &= n + \hat{\delta} \sum_{i=1}^{n} p_i - n\hat{\lambda}^t \sum_{i=1}^{n} \hat{p}_i Z_i \\ &= n + \hat{\delta} \end{aligned}$$

The last line used the constraints in (1). Thus we have $\hat{\delta} = -n$. Substituting this into (4), we get,

$$0 = \frac{1}{\hat{p}_i} - n - n\hat{\lambda}^t Z_i,$$

or equivalently,

$$\hat{p}_i = \frac{1}{n} \frac{1}{1 + \hat{\lambda}^T Z_i}. \tag{5}$$

Substituting (5) in to $\sum_{i=1}^{n} \hat{p}_i Z_i = 0$, we get $\hat{\lambda}$ must solve,

$$0 = \sum_{i=1}^{n} \frac{Z_i}{1 + \hat{\lambda}^T Z_i}. \tag{6}$$

Recall by Remark 2.1 $\mu \in \text{int}(\text{Conv}(X)$, if and only if $\hat{p}_i$ are strictly positive and sum to 1, we must have $p_i < 1$ for all i. This along with (5) implies that $\hat{\lambda}$ must satisfy.

$$1 + \hat{\lambda}^T Z_i > \frac{1}{n}. \tag{7}$$

2.2.1 Duality

We define the function $f : \mathbb{R}^d \to \mathbb{R}$ as,

$$f(\lambda) = -\sum_{i=1}^{n} \log(1 + \lambda^T Z_i).$$

We first claim that f is convex on its domain $\mathcal{D}(f) = \{\lambda : 1 + \lambda^T Z_i > 0, 1 \leq i \leq n\}$. To see this, we note that $g = -\log(\lambda)$ is convex on $\mathcal{D}(g) = \{\lambda : \lambda \geq 0\}$. Let us define the affine transformation, $A_i : \lambda \to 1 + \lambda^T Z_i$. Since convex function applied to an affine transformation preserves convexity, we have $g \circ A_i$ are convex functions with convex domain $\mathcal{D}(g \circ A_i) = \{\lambda : A(\lambda) \geq 0\}$. Since sums preserve convexity, we have

$$f(\lambda) = \sum_{i=1}^{n} g \circ A_i(\lambda)$$

is convex. Also the intersection of convex sets is convex, thus $\mathcal{D}(f) = \bigcap_{i=1}^{n} \mathcal{D}(g \circ A)$ is convex.

Therefore we have f is convex function over a convex domain. Also note that, the gradient of f with respect to λ is given by

$$\nabla f(\lambda) = -\sum_{i=1}^{n} \frac{Z_i}{1 + \lambda^T Z_i}.$$

So if $\hat{\lambda} \in \mathcal{D}(f)$ is a critical point of f, then we have $\hat{\lambda}$ satisfies (6) and thus the corresponding of $\hat{p}_i$ defined by (5) is an optimal weight our original problem (2).

Since f is a convex function, defined on a convex set, $\inf_\lambda f(\lambda)$ is either attained at the critical point $\hat{\lambda}$ or on the boundary. If $\mu \in \operatorname{int}(\operatorname{Conv}(X))$, we showed that and the end of the pervious section that if $\hat{\lambda}$ and $\hat{p}_i$ satisfy (6) and (5) respectively then $1 + \hat{\lambda} Z_i > 1/n$ for all i. Thus $\hat{\lambda} \in \operatorname{int}(\mathcal{D}(f))$ and is a global minimizer.

Thus when $\mu \in \operatorname{int}(\operatorname{Conv}(X))$ we have reduced our original n-dimensional optimization problem (2), into the d-dimensional problem,

$$\begin{array}{ll} \text{minimize:} & f(\lambda) = -\sum_{i=1}^{n} \log(1 + \lambda^T Z_i) \\ \text{subject to:} & 1 + \lambda^T Z_i > 0, \quad 1 \leq i \leq n \end{array}. \tag{8}$$

2.3 Traditional Empirical Likelihood

We we are only interested in the of the optimal solution, which occurs when $1 + \lambda^T Z_i > 1/n$ for all i, we can modify (16) into an unconstrained optimization problem by replacing log with $\log_\star : \mathbb{R} \to \mathbb{R}$, which replaces $\log(x)$ with the second order Taylor approximation to log for $x \leq 0$. I.e.

$$\log_\star(x) = \begin{cases} \log x & x > \frac{1}{n} \\ \log(1/n) - 3/2 + 2nx - (nx)^2/2 & x \leq \frac{1}{n} \end{cases}.$$

We have $f_\star(\lambda) = -\sum_{i=1}^n \log_\star(1+\lambda^T Z_i)$ is convex on all of $\mathbb{R}$ and has the same minimizer as f when $\mu \in \text{int}(\text{Conv}(X))$. Thus for $\mu \in \text{int}(\text{Conv}(X))$, (2) is equivalent to the optimization problem,

$$\text{minimize:} \quad f_\star(\lambda) = -\sum_{i=1}^n \log_\star(1+\lambda^T Z_i) \quad . \tag{9}$$

Since $f_\star$ is in $C^2(\mathbb{R}^d)$ and convex, with gradient and Hessian given by,

$$\nabla f_\star(\lambda) = -\sum_{i=1}^n \log_\star'(1+\lambda^T Z_i) Z_i$$
$$\nabla^2 f_\star(\lambda) = -\sum_{i=1}^n \log_\star''(1+\lambda^T Z_i) Z_i Z_i^T.$$

We can solve (9) iteratively via our favourite first or second order unconstrained convex optimization method such as a damped Newton's method.

When $\lambda \notin \text{Conv}(X)$, we have the gradient of $f_\star$ is

$$\nabla f_\star(\lambda) = -\sum_{i=1}^n \log_\star'(1+\lambda^T Z_i) Z_i \equiv -\sum_{i=1}^n W_i(X_i - \mu).$$

Where $W_i = \log_\star'(1+\lambda^T Z_i) > 0$ since $\log_\star'(x) > 0$ for all x. If $w_i = W_i / \sum_{j=1}^n W_j$ and $\nabla f_\star(\lambda) = 0$ then,

$$\sum_{i=1}^n w_i(X_i - \mu) = 0.$$

But $w_i > 0$ and sum to 1, which contradicts the fact that $\lambda \notin \text{Conv}(X)$. Therefore $f_\star$ does not have any critical points. Our optimization leads to $\|\nabla f_\star(\lambda)\| \to 0$ which is only possible if $\log_\star'(1+\lambda^T Z_i) \to 0$ for all i or equivalently $\|\lambda\| \to \infty$.

In summary, if the resulting minimizer $\hat{\lambda}$ of (9) satisfies (7), then we have $\mu \in \text{int}(\text{Conv}(X))$ and we can recover our optimal $\hat{p}_i$ and we will denote

$$\log \mathcal{R}(\mu) = \sum_{i=1}^n \log(n\hat{p}_i) = \sum_{i=1}^n \log(1+\hat{\lambda}^T Z_i).$$

If $\|\lambda\| \to \infty$, and $p_i = (n(1+\lambda_i^T Z_i))^{-1} \to 0$ for all i, we can conclude $\mu \notin \text{Conv}(X)$. We will call the process of computing $\mathcal{R}(\mu)$ via solving (9) the "traditional" approach.

3 Self-Concordance for Empirical Likelihood

The traditional method for computing the profile empirical likelihood given by solving (9) performs well enough in practice, but convergence is only guaranteed assuming our objective function, is sufficiently well behaved. In the case of Newton's method, we need the Hessian to be both well conditioned and Lipschitz continuous as shown in section 9.5.3 of [BV04]. Although these conditions are often true, but they are not immediately guaranteed and can be tedious/computationally expensive to check in practice so in general we don't.

More over the analysis of Newton's method gives sub-optimality bounds on $f(\lambda) - \inf_\lambda f(\lambda)$ that depend on the condition number and Lipschitz constants, and how close the initialization was to the optimal value. Again, these are not known in practice, therefore we cannot exploit these bounds to create a usable stopping criterion.

This is where self-concordance comes to the rescue. Ensuring our objective function is self-concordant not only guarantees convergence, it also offers a viable easy to compute sub-optimality bound that does not depend on unwieldy constants. The main contribution in [Owe13] was realizing that one mirror the analysis above and replacing log in (16) with a globally defined convex, self-concordant function instead of the merely globally defined convex function $\log_\star$ in the traditional method.

3.1 Self Concordance

The crux of the work done in [Owe13] exploits the notion of self-concordance. So let us define and build some intuition about what self-concordance is.

Definition 3.1. A convex function $f : \mathcal{D}(f) \to \mathbb{R}$, defined on an open convex $\mathcal{D}(f) \subset \mathbb{R}$ is self-concordant if for all $x \in \mathcal{D}(f)$,

$$|f'''(x)| \leq 2f''(x)^{3/2}, \tag{10}$$

Remark 3.2. In the definition the 2 is merely there to make the math more convenient. In general if

$$|f'''(x)| \leq Cf''(x)^{3/2},$$

for some $C \in (0, \infty)$, then it is immediate consequence that $\tilde{f} = \frac{C^2}{4} f$ is self-concordant.

We can now naturally extend the notion of self-concordance to multivariate functions, by forcing self concordance along every line in the domain.

Definition 3.3. A convex function $g : \mathcal{D}(g) \to \mathbb{R}$, defined on an open convex $\mathcal{D}(g) \subset \mathbb{R}^d$ is self-concordant if and only if it g is self concordant on every line in its domain. I.e, for all $x \in \mathcal{D}(g)$ and $v \in \mathbb{R}^d$, we have $f(t) = g(x + tv)$ is self-concordant function of one variable.

It is trivial to verify that self-concordance is preserved under summation and composition is affine transformations. I.e. if f, g are self concordance functions on $\mathbb{R}^d$ and $A : \mathbb{R}^n \to \mathbb{R}^d$ is an affine function, then $f + g$ and $f \circ A$ are also self-concordant. In particular, if $g : \mathcal{D}(g) \to \mathbb{R}$ is a self concordant function for convex $\mathcal{D}(g) \subset \mathbb{R}$, then we have $f : \mathcal{D}(g) \to \mathbb{R}$

$$f(\lambda) = \sum_{i=1}^{n} g(1 + \lambda^T Z_i), \tag{11}$$

is a self concordant function for convex $\mathcal{D}(g) \subset \mathbb{R}^d$. For an example of self-concordance, note that $g(x) = -\log x$ satisfies (10) with equality on for all $x > 0$ and is thus, self-concordant. This along with (11) tells us our objective function for (16) given by

$$f(\lambda) = -\sum_{i=1}^{n} \log(1 + \lambda^T Z_i),$$

is self-concordant on $\mathcal{D}(f) \subset \mathbb{R}^d$.

The self-concordance condition (10) puts a bound on the growth rate of f''', which essentially says that $f''(x)$ can't change too rapidly. This can be seen more clearly in the following way. Then (10) is equivalent to the following,

$$\begin{aligned}
& |f'''(x)| \leq 2f''(x)^{3/2} \\
\iff & \left| -\frac{f'''(x)}{2f''(x)^{3/2}} \right| \leq 1 \\
\iff & \left| \frac{d}{dx} f''(x)^{-1/2} \right| \leq 1 \\
\iff & -1 \leq \frac{d}{dx} f''(x)^{-1/2} \leq 1 \\
\iff & -\Delta x \leq \int_x^{x+\Delta x} \frac{d}{dt} f''(t)^{-1/2} dt \leq \Delta x \\
\iff & -\Delta x \leq f''(x+\Delta x)^{-1/2} - f''(x)^{-1/2} \leq \Delta x
\end{aligned}$$

Rearranging the last line gives tells us f is self-concordant if and only if

$$\frac{f''(x)}{(1+\Delta x f''(x)^{1/2})^2} \leq f''(x+\Delta x) \leq \frac{f''(x)}{(1-\Delta x f''(x)^{1/2})^2}.$$

The key point is that knowing a function is self-concordant gives us control on the second order fluctuation without needing to invoke strong convexity or forcing our function to be Lipschitz. In particular we get the control for free without needing to introduce hard to compute constants. This allows us to analyse our Hessian and the convergence of Newton's method very cleanly.

3.2 Newton decrement

Let us suppose that f is a convex function on a convex $\mathcal{D}(f) \subset \mathbb{R}^d$ and let f^* denote the solution to the optimization problem.

$$\begin{aligned}
\text{minimize:} \quad & f(x) \\
\text{subject to:} \quad & x \in \mathcal{D}(f)
\end{aligned} \tag{12}$$

Given $x \in \mathcal{D}(f)$, we define the Newton's step at x as the the $\Delta x_{nt} \in \mathbb{R}^d$ that minimizes the quadratic approximation $Q_{f,x}(v)$ of f at x. I.e.,

$$\begin{aligned}
\Delta x_{nt} &= \operatorname{argmin}_v Q_{f,x}(v) \\
&= \operatorname{argmin}_v f(x) + \nabla f^T(x) v + \frac{1}{2} v^T \nabla^2 f(x) v \\
&= \operatorname{argmin}_v f(x) + \nabla f^T(x) v + \frac{1}{2} \|v\|^2_{\nabla^2 f(x)}
\end{aligned}$$

Where $\|v\|_{\nabla^2 f(x)} = (v^T \nabla^2 f(x) v)^{1/2}$ is the Hessian norm of v with respect to the Hessian, which is well defined since the Hessian is positive definite. This is a measure of how big the change v is in the natural scale of our Hessian.

It is a routine exercise by taking derivatives to show that $\Delta x_{nt} = -(\nabla^2 f(x))^{-1} \nabla f(x)$. We will denote the Newton decrement $\nu(x)$ as the Hessian norm of our Newton step Δx_{nt}.

$$\begin{aligned}\nu(x) &= \|\Delta x_{nt}\|_{\nabla^2 f(x)} \\ &= (\Delta x_{nt}^T \nabla^2 f(x) \Delta x_{nt})^{1/2} \\ &= ((\nabla f(x))^T (\nabla^2 f(x))^{-1} \nabla f(x))^{1/2}\end{aligned}$$

Intuitively $\nu(x)$ is a natural measure how far x is from the optimal minimizer of quadratic approximation given by $x + \Delta x_{nt}$.

The difference between f and minimum of the quadratic approximation is given by

$$f(x) - Q_{f,x}(\Delta x_{nt}) = \frac{1}{2}(\nabla f(x))^T (\nabla^2 f(x))^{-1} \nabla f(x) = \frac{1}{2}\nu(x)^2$$

$$\begin{aligned}f(x) - Q_{f,x}(\Delta x_{nt}) = &\ -\nabla f^T(x) \Delta x_{nt} - \frac{1}{2}\|\Delta x_{nt}\|^2_{\nabla^2 f(x)} \\ = &\ (\nabla f(x))^T (\nabla^2 f(x))^{-1} \nabla f(x) - \frac{1}{2}\nu(x)^2 \\ = &\ \frac{1}{2}\nu(x)^2\end{aligned}$$

Therefore $\nu^2/2$ measures the sub-optimality of $f(x)$ compared to the minimum of its quadratic approximation at x.

Heuristically, we can use the Newton decrement to derive a sub-optimality bound on how far $f(x)$ is from the true minimum f^* in the case where f is self-concordant by using the the quadratic approximation as an intermediary. This was done in a detailed rigorous way in Section 9.6.3 in [BV04]. The proof was very clear and not missing any steps, so we will avoid the details and give the general ideas involved in the proof.

Indeed, self-concordance gives us a bound on how quickly the second derivative can change, and consequently gives a bound on the error between f and the quadratic approximation.The newton decrement gives us the error between f and the and minimum of of the quadratic approximation. Combing the two we can get an error between $f(x)$ and f^*.

Theorem 3.4. Let $f : \mathcal{D}(f) \to \mathbb{R}$ for convex $\mathcal{D}(f) \subset \mathbb{R}^d$ be self-concordant with minimum f^*. Then if $0 < \varepsilon < 0.68^2$ and $\nu(x)^2 \leq \varepsilon$, then

$$f(x) - f^* \leq \varepsilon.$$

Therefore in the case of self-concordant functions, we have a natural stopping criterion for our optimization problem. When $\nu(x)^2$ is less than our desired accuracy ε, we are guaranteed to be within ε of the global minimum.

3.3 Self-Concordant Empirical Likelihood

We now return to the problem of solving (16). In the traditional approach we showed that it was equivalent to solving the unconstrained convex optimization problem (9) by replacing log with a globally defined convex approximation $\log_\star$, and applying Newton's method. We will follow the same recipe as the traditional approach but will chose a globally defined convex, and self-concordant approximation to replace log.

Recall we arrived at $\log_\star(x)$ by restricting to $\log x$ when $x \geq 1/n$ and using the second order Taylor expansion of log for $x \leq 1/n$, that $\log_\star \in C^2(\mathbb{R})$ and is convex. We also have $\log_\star$ is self concordant for $x > 1/n$ since $\log x$ is and similarly when $x < 1/n$, we have $\log_\star$ is self concordant since $\log'''(x) = 0$. The only point $\log_\star$ fails self-concordance is at $x = 1/n$ since the third derivative is not defined. We will fix this by replacing the second order approximation at $x < 1/n$ with a higher order one.

Define for $a > 0$, we define $L_{k,a} : \mathbb{R} \to \mathbb{R}$ as

$$L_{k,a}(x) = \begin{cases} \log(x) & x > a \\ \sum_{i=0}^{k} \log^{(i)}(a)\frac{(x-a)^i}{i!} & x \leq a \end{cases},$$

where $\log^{(i)}(a)$ is the i-th derivative of log at a. Note that $\log_\star = L_{2,\frac{1}{n}}$. We want to find a k that ensures self-concordance. When $k = 3$, we have $L_{3,a}$ is not even convex, but we find $-L_{4,a}$ is both globally convex and self-concordant. The proof of self-concordance was a simple calculation done in great detail and clarity in Theorem 1 in [Owe13] and thus we will omit it here. Although they showed self concordance for a general $a > 0$, we are really only interested in $a \leq 1/n$ since our analysis in section 2 showed that the optimal solution to (16) only occurs when $1 + \hat{\lambda}^T Z_i > 1/n$ for all i. For the sake of convenience, we will denote $L_{4,\frac{1}{n}}$ as $\log_\heartsuit$.

We now define $f_\heartsuit : \mathbb{R}^d \to \mathbb{R}$ by

$$f_\heartsuit(\lambda) = -\sum_{i=1}^{n} \log_\heartsuit(1 + \lambda^T Z_i).$$

From (11), we know that $f_\heartsuit$ is self-concordant. We now examine the following unconstrained convex optimization problem

$$\text{minimize:} \quad f_\heartsuit(\lambda) = -\sum_{i=1}^{n} \log_\heartsuit(1 + \lambda^T Z_i) \quad . \tag{13}$$

The analysis of the solution to the traditional problem(9) is identical to that of the self-concordant one given by (13). Indeed, we can now restate Section 2.3 verbatim replacing $\log_\star$ with $\log_\heartsuit$, and everything would still hold true. So we will avoid that and just go straight to the punch line.

The gradient and Hessian of $f_\heartsuit$ are given by,

$$\begin{aligned}\nabla f_\heartsuit(\lambda) &= -\sum_{i=1}^n \log'_\heartsuit(1+\lambda^T Z_i)Z_i\\ \nabla^2 f_\heartsuit(\lambda) &= -\sum_{i=1}^n \log''_\heartsuit(1+\lambda^T Z_i)Z_i\end{aligned}$$

From this we can run our Newton's method to achieve accuracy ε, with the stopping criterion that we stop when

$$\nu(x)^2 = (\nabla f_\heartsuit(x))^T(\nabla^2 f_\heartsuit(x))^{-1}\nabla f_\heartsuit(x) \le \varepsilon.$$

Just as with the traditional approach, when $\mu \in \operatorname{int}(\operatorname{Conv}(X))$, then the minimizer $\hat{\lambda}$ and the corresponding $\hat{p}_i = (n(1+\hat{\lambda}^T Z_i))^{-1}$ are solutions to (16) and (2) respectively, and can be used to compute $\mathcal{R}(\mu)$ by the formula,

$$\log \mathcal{R}(\mu) = \sum_{i=1}^n \log(n\hat{p}_i) = \sum_{i=1}^n \log(1+\hat{\lambda}^T Z_i).$$

If $\|\lambda\| \to \infty$ and $p_i = (n(1+\lambda^T Z_i))^{-1} \to 0$ for all i, then we can conclude $\mu \notin \operatorname{Conv}(X)$. We will call the method for computing $\mathcal{R}(\mu)$ via solving via (13), the self-concordant approach .

4 Adjusted Empirical Likelihood

A lot of the difficulty that comes with computing for $\mathcal{R}(\mu)$, comes from the fact that our constraints in (2) may not always satisfied and thus $\mathcal{R}(\mu)$ may not be defined. As stated in [Owe01] and [CVA08], under certain mild moment condition on X, we have the true mean μ_0 will be contianed in the convex hull of the data, with as $n \to \infty$ almost surely. However, when μ is not close to μ_0 or n is not very large, we will have that there is a good chance that $\mu \notin \operatorname{Conv}(X)$.

Chen, Variyath, and Abraham in [CVA08] circumvented this issue by defining a new function function called the adjusted empirical likelihood (AEL) function, which we will denote at $\mathcal{R}_{AEL} : \mathbb{R}^d \to \mathbb{R}$, well-defined on all of $\mathbb{R}^d$, that approximates the $\mathcal{R}$ when μ is close to μ_0, and has has the same asymptotic properties as $\mathbb{R}$.

First suppose $\mu \in \mathbb{R}^d$ is fixed. Let $X = \{X_1, \dots, X_n\}$. We want to introduce new centered pseudo-sample X_{n+1} into our data that by defining

$$(X_{n+1} - \mu) = -\frac{a_n}{n}\sum_{i=1}^n (X_i - \mu) = -a_n(\bar{X} - \mu) \tag{14}$$

for $a_n > 0$ specified by the user. We will let $\tilde{X} = X \cup \{X_{n+1}\}$ be the new augmented data. We now define the adjusted empirical likelihood function $\mathcal{R}_{AEL}(\mu)$ as the profile likelihood of $\tilde{X}$. I.e.

$$\mathcal{R}_{AEL}(\mu; a_n) = \sup\left\{\prod_{i=1}^{n+1}(n+1)p_i : p_i \ge 0, \sum_{i=1}^{n+1} p_i = 1, \sum_{i=1}^{n+1} p_i X_i = \mu\right\} \tag{15}$$

Since $\mathcal{R}_{AEL}(\mu)$ is the just the profile empirical likelihood for the data $\tilde{X}$, we can compute $\mathcal{R}_{AEL}(\mu)$ using the same strategy via duality outlined in Section 2

4.1 Properties of AEL

First of all note that if $p_i = a_n/(na_n + n)$ for $i = 1 \dots n$ and $p_{n+1} = n/(na_n + n)$, then, $p_i \geq 0$, sum to 1, and

$$\begin{aligned}\sum_{i=1}^{n+1} p_i(X_i - \mu) &= \frac{a_n}{na_n + n}\sum_{i=1}^{n}(X_i - \mu) + \frac{n}{na_n + n}(X_{n+1} - \mu) \\ &= \frac{na_n}{na_n + n}\frac{1}{n}\sum_{i=1}^{n}(X_i - \mu) - \frac{na_n}{na_n + n}(\bar{X}_n - \mu) \\ &= 0.\end{aligned}$$

Thus we have $\mu \in \mathrm{int}(\mathrm{Conv}(\tilde{X}))$ is which implies $\mathcal{R}_{AEL}(\mu; a_n)$ is defined on all of $\mathbb{R}^d$. A stronger consequence of the above computation is that $\log \mathcal{R}_{AEL}(\mu; a_n)$ is actually bounded from below.

$$\begin{aligned}\log \mathcal{R}_{AEL}(\mu; a_n) &\geq \sum_{i=1}^{n+1} \log(np_i) \\ &= \sum_{i=1}^{n} \log\left(n\frac{a_n}{na_n + n}\right) + \log\left(n\frac{n}{na_n + n}\right) \\ &= n\log\left(\frac{a_n}{a_n + 1}\right) + \log\left(\frac{n}{a_n + 1}\right).\end{aligned}$$

This is in contrast to $\log \mathcal{R}(\mu)$ that approaches $-\infty$ as μ approaches $\partial\mathrm{Conv}(X)$ from the interior and is undefined outside of $\mathrm{Conv}(X)$.

Since this is not a report on the properties of the adjusted empirical likelihood, we will focus on developing some heuristic intuition without getting bogged down in details.

The addition of X_{n+1} is the is equivalent to adding in a_n new pseudo-samples whose direction opposes the difference between μ and the true mean μ_0. This can be seen heuristically (14) and the law of large numbers implies for n large enough

$$X_{n+1} - \mu = a_n(\bar{X}_n - \mu) \approx -a_n(\mu_0 - \mu)$$

Thus we are adding X_{n+1} into the data so that we expand out convex hull approximately in the direction of $\mu - \mu_0$ and the a_n controls the rate of expansion.

Also note that when n is large and μ is close to μ_0, then $X_{n+1} \approx 0$, and thus will not contribute much to our constraint in (15) $\mathcal{R}_{AEL}(\mu; a_n) \approx \mathcal{R}(\mu)$. This motivates that at μ_0, $\mathcal{R}_{AEL}(\mu_0; a_n)$ should have similar asymptotic behaviour as $\mathcal{R}(\mu_0)$ for a_n not growing too quickly. If a_n grows too fast, X_n will dominate over the other samples in the optimization. Indeed it was shown in [CVA08] that if a_n is $o_p(n^{2/3})$ then $\log \mathcal{R}_{AEL}(\mu_0; a_n)$ is equivalent to $\log \mathcal{R}(\mu_0)$ asymptotically, upto first order.

Overall the pros of the $\mathcal{R}_{AEL}$ are not insignificant. It is well defined and bounded from below on all of $\mathbb{R}^d$, the constraints are always satisfies, it shares the same nice asymptotic properties at $\mathcal{R}$ when μ is near μ_0, and outputs reasonable values when μ is far away from μ_0. However it does come with it's own challenges. The performance, and asymptotics are greatly dependant on our choice of a_n, which there is no natural methodical way of choosing. Furthermore, when n is small, the asymptotic fail to hold and it's the values are not as meaningful. Finally, the empirical likelihood was defined in a more natural way than the adjusted empirical, and is thus more interpretable and easy to explain to non-experts.

5 Confidence region

One of the most important properties about the empirical likelihood method is that asymptotically the profile empirical likelihood ratio shares the same limiting distribution as the parametric likelihood ratio, and admits its own non-parametric version of Wilk's theorem called the empirical likelihood theorem (ELT).

Theorem 5.1 (ELT)**.** [Owe01] Let $X_1, \dots, X_n$ be i.i.d. random vector in $\mathbb{R}^d$ with mean μ_0 and variance matrix V of rank r. Let $\mathcal{R}$ be the profile likelihood defined in Section 2. Then we have

$$-2\log \mathcal{R}(\mu_0) \xrightarrow[n\to\infty]{d} \chi^2_{(r)}.$$

A similar result was shown in [CVA08] for the adjusted empirical likelihood ratio, which we will call the adjusted empirical likelihood theorem (AELT)

Theorem 5.2 (AELT)**.** Let $X_1, \dots, X_n$ be i.i.d. random vector in $\mathbb{R}^d$ with mean μ_0 and variance matrix V of rank d, and $a_n = o_p(n^{2/3})$. Let $\mathcal{R}$ be the profile likelihood defined in Section 2. Then we have

$$-2\log \mathcal{R}_{AEL}(\mu_0; a_n) \xrightarrow[n\to\infty]{d} \chi^2_{(d)}.$$

Theorem 5.1 and 5.2 suggests that assuming the variance matrix of X has full rank and $a_n = o_p(n^{2/3})$, then

$$\begin{aligned} C_\alpha^{EL} &= \{\mu : -2\log \mathcal{R}(\mu) \le \chi^2_{(d)}(1-\alpha)\} \\ C_\alpha^{AEL} &= \{\mu : -2\log \mathcal{R}_{AEL}(\mu; a_n) \le \chi^2_{(d)}(1-\alpha)\} \end{aligned}$$

are a good approximation for a $100(1-\alpha)\%$ confidence region for the mean, where $\chi^2_{(d)}(1-\alpha)$ is the $1-\alpha$ quantile of the chi-square distribution with d degrees of freedom. The following lemmas will be when computing C_α^{EL}.

Lemma 5.3. For all $0 < \alpha \le 1$, we have $\bar{X}_n \in C_\alpha^{EL}$, and $\tilde{X}_n = \frac{1}{n}\sum_{i=1}^{n+1} X_i \in C_\alpha^{AEL}$.

Proof. Let us compute the empirical likelihood ratio.

$$\mathcal{R}(\bar{X}) = \sup\{\sum_{i=1}^n \log(np_i) : p_i \ge 0, \sum_{i=1}^n p_i = 1, \sum_{i=1}^n p_i X_i = \bar{X}_n\}.$$

Then we note that when $p_i = 1/n$, we have the constraints are satisfied and $\sum_{i=1}^n \log(np_i) = 0$. Since $\mathcal{R}(\mu) \le 0$ whenever defined, we have $\mathcal{R}(\bar{X}_n) = 0$. Thus $\bar{X}_n \in C_\alpha$ for all α.

An analogous argument works to show $\tilde{X}_n \in C_\alpha^{AEL}$. $\square$

Lemma 5.4. $\mathcal{R}$ is a concave function on it's domain, and C_α^{EL} is convex.

Proof. Let $\mu_1, \mu_2 \in \mathcal{D}(\mathcal{R})$, and $t \in [0,1]$, then let $\mu = t\mu_1 + (1-t)\mu_2$. Suppose $r_i, s_i \geq 0$, $\sum_{i=1}^n r_i = \sum_{i=1}^n s_i = 1$ and $\sum_{i=1}^n r_i X_i = \mu_1, \sum_{i=1}^n s_i X_i = \mu_2$. Then,

$$\begin{aligned} t\sum_{i=1}^n \log(nr_i) + (1-t)\sum_{i=1}^n \log(ns_i) &= \sum_{i=1}^n (t\log(nr_i) + (1-t)\log(ns_i)) \\ &\leq \sum_{i=1}^n \log(n(tr_i + (1-t)s_i)). \end{aligned}$$

The last line used the concavity of log. Now note that $tr_i+(1-t)s_i \geq 0$,$\sum_{i=1}^n tr_i+(1-t)s_i = 1$ and $\sum_{i=1}^n (tr_i + (1-t)s_i)X_i = t\mu_1 + (1-t)\mu_2 = \mu$. Thus we have

$$t\sum_{i=1}^n \log(nr_i) + (1-t)\sum_{i=1}^n \log(ns_i) \leq \mathcal{R}(\mu).$$

Now taking the supremum on the left hand side over all possible r_i, s_i that satisfy our constraints, gives us

$$t\mathcal{R}(\mu_1) + (1-t)\mathcal{R}(\mu_2) \leq \mathcal{R}(\mu).$$

Thus we have shown $\mathcal{R}$ is concave.

Since $\mathcal{R}$ is concave, we have $-2\mathcal{R}$ is convex. The sub-level sets of a convex function are convex, therefore C_α is convex. □

Remark 5.5. It was also shown in [CH13] that the confidence regions of for the adjusted empirical likelihood were are convex.

6 Implementation

We will now discuss how to numerically compute $\mathcal{R}(\mu)$ and $\mathcal{R}_{AEL}(\mu)$. We will compute these quantities by solving the dual problem (16) associated with them using Newton's method. Owen suggested in [Owe13] to use a damped Newton's method given by algorithm 9.5 in [BV04], and Chen et. al. recommended using a modified Newton method outlined in [CVA08].

6.1 Outline of optimization methods

Suppose we have the following convex optimization problem, we wish to solve.

$$\begin{array}{ll} \text{minimize:} & f(x) \\ \text{subject to:} & x \in \mathcal{D}(f) \end{array}. \tag{16}$$

Traditional newton method involves moving iteratively from $x^k \to x^{k+1} = x^k + \Delta x^k$, where

$$\Delta x^k = -(\nabla^2 f(x))^{-1} \nabla f(x).$$

In the damped Newton method our update is still in the direction of the Newton step, but is of the form $x^k + t\Delta x^k$. We perform a backtracking line search on t until our objective function decreases sufficiently. Let $\beta \in (0,1), \alpha \in (0,1/2)$ and initialize $t_k = 1$, and we keep replacing t^k by βt_k until

$$f(x^k + t_k \Delta x^k) \leq f(x^k) + \alpha t_k \nabla^T f(x^k) \Delta x^k).$$

Then define $x^{k+1} = x^k + t_k \Delta x^k$. We stop our algorithm when the newton decrement $\nu(x)^2/2 \leq \varepsilon$ for our desired accuracy ε. In the case where f is self-concordant we can stop when $\nu^2 \leq \varepsilon$.

The modified Newton method of Chen is similar to the damped newton method where we compute the Newton step and then backtrack. Our update is of the form $x^k + \delta_k \Delta x^k$, where we initialize $\delta_k = (1+k)^{-1/2}$, we then replace δ_k by δ_2 until $x^k + \delta_k \Delta x^k \in \mathcal{D}(f)$ and

$$f(x^k + \delta_k \Delta x^k) \leq f(x^k).$$

Then define $x^{k+1} = x^k + \delta \Delta x^k$, and we stop when $\|\Delta x^k\| \leq \varepsilon$.

Damped Newton method assumes we are working in the unconstrained setting whereas Chen's method is proven to work for empirical likelihood calculation [CSW02].

6.1.1 Computation of $\mathcal{R}(\mu)$ and $\mathcal{R}_{AEL}(\mu; a_n)$

We can solve for $\log \mathcal{R}(\mu)$ by solving (16) using Chen's Newton method, we will call the output of this $\log \mathcal{R}^{Chen}$. We can also use the damped Newton's method to solve for $\log \mathcal{R}(\mu)$ using the traditional and self-concordant method. We will denote the solution to (9), (13) as $\log \mathcal{R}^{trad}$ and $\log \mathcal{R}^{SC}$ respectively. If our final output for $\hat{\lambda}$ does not satisfy the constraints $1 + \lambda^T Z_i \geq 1/n$ for all i, we will define $\log \mathcal{R}(\mu) = -\infty$.

Similarly we can solve for $\log \mathcal{R}_{AEL}(\mu; a_n)$ by augmenting $X_{n+1} = -a_n(\bar{X} - \mu) + \mu$ to our data and then solving the associated (16) problem using Chen's Newton method, we will call the output of this $\log \mathcal{R}_{AEL}^{Chen}$. We can also use the damped Newton's method to solve for $\log \mathcal{R}_{AEL}(\mu; a_n)$ using the traditional and self-concordant method. We will only look at the self-concordant method, since global convergence is guaranteed and we will denote this as $log \mathcal{R}_{AEL}^{SC}$.

In each of these cases we initialize $\lambda^0 = 0$ as it is guaranteed to be in the domain of our objective function. For damped Newton method we set $\alpha = 0.3$ and $\beta = 0.8$ as recommended by [Owe13] in his code. Our desired accuracy will be $\varepsilon = 10^{-8}$ in all cases. In case of the AEL, we will use $a_n = \max(1, \log n/2)$ as suggested in [CVA08].

6.2 Comparison of Methods

To test the performance, on random samples generated from a Poison(3) and Normal$(0, \text{diag}(1, 10))$ of size 30. The goal of these experiments is to test to see how well the 5 different methods perform for different values of μ as opposed to doing inference. We chose these two distributions since it gives up a chance to see how the algorithms handle data in both the discrete and continuous setting in 1 and 2 dimensions for various values of μ. Poisson is also a good choice since it is asymmetric about the mean.

We choose size $n = 30$, to ensure the n is large enough for the asymptotic approximations for the confidence regions to be valid, but also don't want n so large so that the ELR concentrates at the mean, thus making it difficult to see differences between AEL and EL confidence regions.

We generated the data in R using the seed 072892.

6.2.1 Results: Poisson Data

We generated 30 samples with mean 3 and sample mean $\mu_0 = 2.96$ and range

$$[x_{(1)}, x_{(30)}] = [1, 7]$$

. A histogram of the data is plotted below,

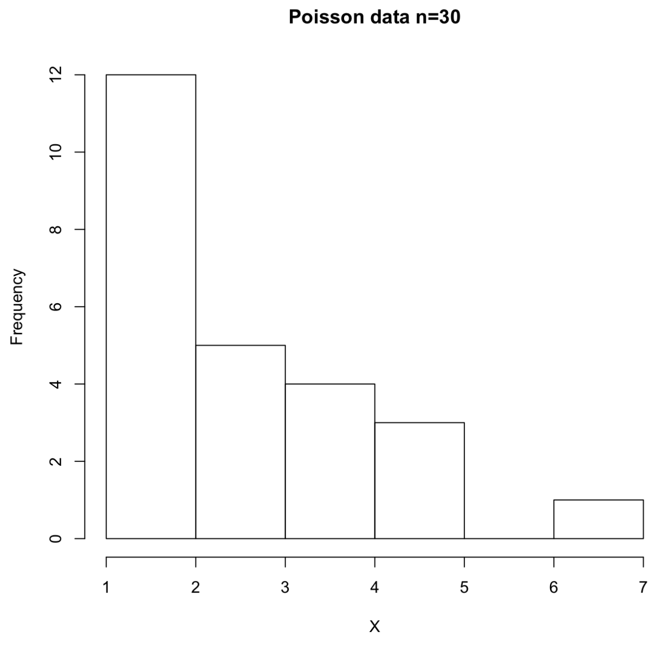

Figure 1: Histogram of the Poisson data

We now compute the EL and AEL near μ_0, near the edge and some where in between. We list the results of $mu = 3, 6, 1.0001$ in Table 1, 2 and 3 respectively. Th

Table 1: Poisson Data, $n = 30, \mu_0 = 3$

	$\mu = 3$		
	Value	Iterations	$\hat{\lambda}$
$\mathcal{R}^{Chen}(\mu)$	-0.009294779	5	-0.01844364
$\mathcal{R}^{trad}(\mu)$	-0.009294779	2	-0.01844389
$\mathcal{R}^{SC}(\mu)$	-0.009294779	2	-0.01844389
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-0.008144133	5	-0.01728203
$\mathcal{R}^{SC}_{AEL}(\mu)$	-0.008144133	2	-0.01728222

Table 2: Poisson Data, $n = 30, \mu_0 = 3$

	$\mu = 6$		
	Value	Iterations	$\hat{\lambda}$
$\mathcal{R}^{Chen}(\mu)$	-29.37578	44	-0.94555
$\mathcal{R}^{trad}(\mu)$	-29.37578	6	-0.5896481
$\mathcal{R}^{SC}(\mu)$	-29.37578	12	-0.5896476
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-8.3734	50	-0.1824219
$\mathcal{R}^{SC}_{AEL}(\mu)$	-8.3734	10	0.04849103

Table 3: Poisson Data, $n = 30, \mu_0 = 3$

	$\mu = 1.0001$		
	Value	Iterations	$\hat{\lambda}$
$\mathcal{R}^{Chen}(\mu)$	-207.2654	100	8799.667
$\mathcal{R}^{trad}(\mu)$	-207.2654	20	8089.754
$\mathcal{R}^{SC}(\mu)$	-207.2654	21	8089.753
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-7.976862	29	0.2810761
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-7.976862	4	0.2810761

6.2.2 Results: Multivariate Gaussian Data

We generated 30 samples with $\mu_0 = 0$ and $\Sigma = \text{diag}(2, 10)$. The data is plotted below in Figure 2.T he sample mean $\bar{x} = (0.050019460.47875169)$ We now compute the EL and AEL near μ_0, near the boundary of the convex hull and some where in between We list the results of $mu = (0, 0, (3, 6), (-2.5, 3.1)$ in table 4, 5, 6 respectively.

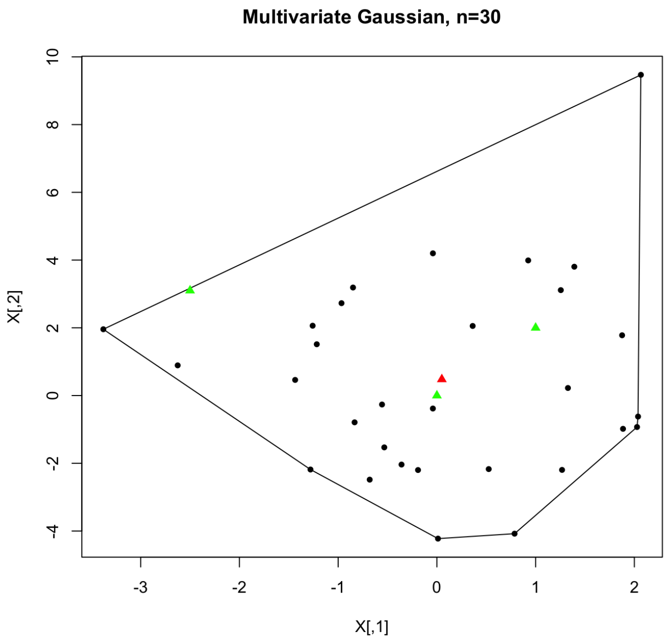

Figure 2: Plot of the multivariate Gaussian data. The red point is the sample mean, and the green points are the μ we are testing.

Table 4: Normal Data, $n = 30, \mu_0 = 0, \Sigma = \text{diag}(2, 10)$

	$\mu = (0, 0)$		
	Value	Iterations	$\hat{\lambda}$
$\mathcal{R}^{Chen}(\mu)$	-0.4689824	20	(0.02499763 0.06551925)
$\mathcal{R}^{trad}(\mu)$	-0.4689824	3	(0.02499767 0.06551926)
$\mathcal{R}^{SC}(\mu)$	-0.4689824	3	(0.02499767 0.06551926)
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-0.4138172	20	(0.02296238 0.06131776)
$\mathcal{R}^{SC}_{AEL}(\mu)$	-0.4138172	3	(0.02296241 0.06131777)

Table 5: Normal Data, $n = 30, \mu_0 = 0, \Sigma = \mathrm{diag}(2, 10)$

	$\mu = (1, 2)$		
	Value	Iterations	$\hat{\lambda}$
$\mathcal{R}^{Chen}(\mu)$	-8.629085	55	(-0.59216340 -0.02584358)
$\mathcal{R}^{trad}(\mu)$	-8.629085	7	(-0.536938901 -0.006552615)
$\mathcal{R}^{SC}(\mu)$	-8.629085	9	(-0.492759316 0.008880111)
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-6.57169	49	(-0.41747434 -0.03753552)
$\mathcal{R}^{SC}_{AEL}(\mu)$	-6.57169	6	(-0.41747433 -0.03753553)

Table 6: Normal Data, $n = 30, \mu_0 = 0, \Sigma = \mathrm{diag}(2, 10)$

	$\mu = (-2.1, 3.5)$		
	Value	Iterations	$\hat{\lambda}$
$\mathcal{R}^{Chen}(\mu)$	-117.7297	100	(18.57244, -13.43548)
$\mathcal{R}^{trad}(\mu)$	-117.7297	13	(9.357252, -6.587463)
$\mathcal{R}^{SC}(\mu)$	-117.7297	19	(9.357246 -6.587458)
$\mathcal{R}^{Chen}_{AEL}(\mu)$	-10.02464	48	(0.15980045 -0.04820261)
$\mathcal{R}^{SC}_{AEL}(\mu)$	-10.02464	12	(-0.06062915 0.01168551)

6.2.3 Analysis

First point of note amongst the results above is that the three methods for computing the EL and the 2 methods for computing the AEL both gave nearly identical results, for the log profile (adjusted) empirical likelihood function at the μ we chose to compare. Also when μ was close to the sample mean, we found that the AEL and EL gave very similar results. The real divergence came the closer we got the the boundary of the convex hull, which again is to be expected since the log AEL profile is bounded from below and the log EL profile diverges to infinity near the boundary. Both the EL and AEL took significantly more iterations for the algorithm to terminate as μ deviated from the sample mean.

In all scenarios, we found that the damped Newton's method performed significantly better that the modified Newton's method of Chen. Although it gave the same output at the damped version, it took significantly more iterations to terminate. This was the case for both the EL and AEL.

Comparing the traditional approach to the self-concordant approach, we found surprisingly that the traditional approach performed the best. The profile value was outputted were identical and $\hat{\lambda}$ only deviated from of the self-concordant output but was on the same order of magnitude. However, we found that the traditional approach consistently terminated in fewer iterations than the self-concordant approach and the difference became more apparent as μ deviated from the sample mean. This suggests that although the self-concordant method has provable convergence properties that the traditional approach lacks, it is not as effective in practice as the traditional approach.

6.3 Comparison of Confidence Regions

We will now outline how to construct confidence regions for μ. Recall that the confidence regions of μ of signifincant $100(1-\alpha)\%$ is well approximated by

$$C_\alpha^{EL} = \{\mu : -2\log\mathcal{R}(\mu) \le \chi^2_{(d)}(1-\alpha)\}$$
$$C_\alpha^{AEL} = \{\mu : -2\log\mathcal{R}_{AEL}(\mu) \le \chi^2_{(d)}(1-\alpha)\}$$

By Lemma 5.3 we know that the sample mean $\bar{x} \in \operatorname{int}(C_\alpha^{EL})$ and it maximizes $\mathcal{R}$. By Lemma 5.4 we know that $-2\mathcal{R}$ and C_α^{EL} are convex, and thus for each $x \in \partial C_\alpha^{EL}$, there is a line connecting $\bar{x}$ to x and $\mathcal{R}$ is monotonic along this line. So we can do a line search in every direction around $\bar{x}$ until we find the μ on the line such that $-2\mathcal{R}(\mu) = \chi^2_{(d)}(1-\alpha)$. If we do this for enough equally spaced lines, it will let us generate an approximation to the boundary of C_α^{EL}. An identical argument allows us to construct regions for the AEL.

In the case of $d = 1$, we only need to perform 2 line searches; one to the left and right of $\bar{x}$ respectively. We did this for our Poisson data for to get 50% and 95% confidence regions given by,

$$C_{0.5}^{EL}=[2.771,\ 3.165] \qquad C_{0.05}^{EL}=[2.400,3.611]$$

$$C_{0.5}^{AEL}=[2.757,\ 3.179] \quad C_{0.05}^{AEL}=[2.397,\ 3.663]$$

In the case of $d = 2$, we can generate N equally spaced lines along the unit circles centred at $\bar{x}$ and perform a line search on each. We did this for $N = 200$. We plot the 50% and 95% confidence regions below.

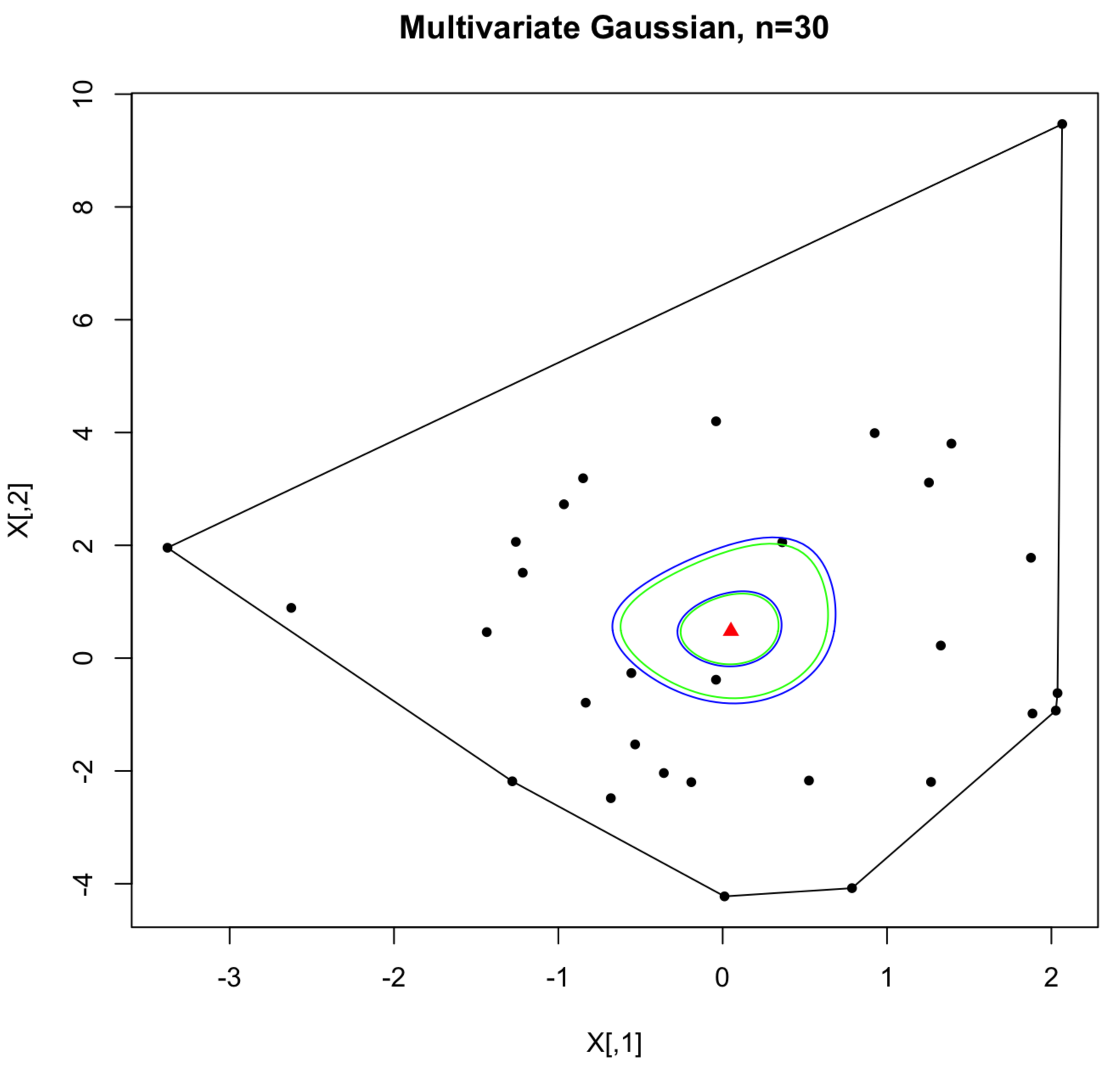

Figure 3: The red point is the sample mean, and the green contours outline the 50% and 95% coverage regions for the EL. The green contours outline the 50% and 95% coverage regions for the AEL.

The confidence regions take on the shape of our data, as opposed to the ellipses that arise from the classical Gaussian approach. Another remark worth making is that the confidence regions of AEL seems to always be large than the EL in both our data sets.

7 References

[Bra07] Francesco Bravo. Empirical likelihood methods with application to econometrics. 2007.

[BV04] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004.

[CH13] Jiahua Chen and Yi Huang. Finite-sample properties of the adjusted empirical likelihood. *Journal of Nonparametric Statistics*, 25(1):147–159, 2013.

[CSW02] J Chen, RR Sitter, and C Wu. Using empirical likelihood methods to obtain range restricted weights in regression estimators for surveys. *Biometrika*, 89(1):230–237, 2002.

[CVA08] Jiahua Chen, Asokan Mulayath Variyath, and Bovas Abraham. Adjusted empirical likelihood and its properties. *Journal of Computational and Graphical Statistics*, 17(2):426–443, 2008.

[Owe88] Art B Owen. Empirical likelihood ratio confidence intervals for a single functional. *Biometrika*, 75(2):237–249, 1988.

[Owe01] Art B Owen. *Empirical likelihood*. Wiley Online Library, 2001.

[Owe13] Art B Owen. Self-concordance for empirical likelihood. *Canadian Journal of Statistics*, 41(3):387–397, 2013.

[TAOT14] Masanobu Taniguchi, Tomoyuki Amano, Hiroaki Ogata, and Hiroyuki Taniai. *Empirical Likelihood Approaches for Financial Returns*, pages 41–64. Springer International Publishing, Cham, 2014.

8 Code

8.1 Computing EL and AEL

Below is the R code for used to compute $\mathcal{R}$ and $\mathcal{R}_{AEL}$ in the various ways.

```
library(mvtnorm)
rm(list=ls())

## Compute L_ord(x,a)

logEL <- function(x,a,ord){
  if(x>a){
    return(log(x))
  }
  else{
    sum <- log(a)
    for (i in 1:ord){
      sum <- sum + (-1)^(i+1)/(a^i*i)*(x-a)^i
    }
    return(sum)
  }
}


dlogEL <- function(x,a,ord){
  if(x>a){
    return(1/x)
  }
  else{
    sum <- 0
    for(i in 1:ord){
      sum <- sum + (-1)^(i+1)/a^i*(x-a)^(i-1)
    }
    return(sum)
  }
}


ddlogEL <- function(x,a,ord){
  if(x>a){
    return(-1/x^2)
  }
  else{
    sum <- 0
    for(i in 2:ord){
      sum <- sum + (-1)^(i+1)*(i-1)/a^i*(x-a)^(i-2)
    }
    return(sum)
  }
}


## Implementation of damped Newton method

logELRdamp <- function(X, # Data matrix
                 mu, # Desired mean
                 max_iter, # Maximum number of iterations
                 tol, # desired tolerance
                 ord # ord = 2 -> logELRtrad, logELRnewt -> logELRdamp
                 ){

  n <- nrow(X)
  d <- ncol(X)
```

```
# Center data, Z[i,]=X[i,]-mu
Z <- t(t(X) - mu)

# Initialize
lambda <- rep(0, d)
lam_z <- rep(1, n) # lam_z[i] = 1+lambda^t(Z[i,]) corresponding to lambda =
    rep(0,d)

# Line search parameters
alpha <- 0.3
beta <- 0.8
t <- 1
k <- 0
converged <- F
repeat{
  # If too many iterations
  if (k >= max_iter) {
    weights <- (n * lam_z)^-1
    if (any(lam_z<1/n)){
      logELRdamp <- -Inf}
    else{
      logELRdamp <- -sum(sapply(lam_z, logEL, a = 1/n, ord)) }
    return(list(logELRdamp = logELRdamp,
                weights = weights,
                sum = sum(weights),
                lambda = lambda,
                converged = converged,
                iter = k))
  }

  # Compute gradient, Hessian, Newton step, Newton decremenet
  grad <- -colSums(Z*sapply(lam_z, dlogEL,a = 1/n, ord))
  hess <- -t(Z) %*% (Z *sapply(lam_z, ddlogEL,a = 1/n, ord))
  delta_lam <- -solve(hess, grad)
  dec <- sqrt(-t(grad)%*%delta_lam)
  if (dec^2<tol){
    break
  }

  # Initialize line search
  line_iter <- 0
  logELRdamp <- -sum(sapply(lam_z, logEL, a = 1/n, ord))
  z_delta_lam <- as.vector(Z %*% delta_lam)

  # Line Search
  repeat {
      new_lam_z <- lam_z + z_delta_lam
      new_logELRdamp <- -sum(sapply(new_lam_z,logEL, a = 1/n, ord))
      if (new_logELRdamp > logELRdamp + alpha*t*grad%*%delta_lam & line_iter <
         max_iter) {
        t <- beta*t
        delta_lam <- beta*delta_lam
        z_delta_lam <- z_delta_lam
        line_iter <- line_iter + 1
      }
      else {
        break
      }
    }

    lambda <- lambda + delta_lam
    lam_z <- new_lam_z
    k <- k + 1
    t <- 1
  }
  logELRdamp <- -sum(sapply(lam_z, logEL, a = 1/n, ord))
  weights <- (n * lam_z)^-1
  converged <- T
  return(list(logELRdamp = logELRdamp,
              weights = weights,
              sum = sum(weights),
              lambda = lambda,
              converged = converged,
              iter = k))
}

## Traditional logELR using log_star

logELRtrad <- function(X, # Data matrix
                mu, # Desired mean
                max_iter, # Maximum number of iterations
                tol # desired tolerance
){
  z <- logELRdamp(X, mu, max_iter, tol/2, ord = 2)
  return(list(logELRtrad = z$logELRdamp,
              weights = z$weights,
              sum = z$sum,
              lambda = z$lambda,
              converged = z$converged,
              iter = z$iter))
}

## Compute self concordant logELR
```

```
logELRsc <- function(X, # Data matrix
                mu, # Desired mean
                max_iter, # Maximum number of iterations
                tol # desired tolerance
){
  z <- logELRdamp(X, mu, max_iter, tol, ord = 4)
  return(list(logELRsc = z$logELRdamp,
             weights = z$weights,
             sum = z$sum,
             lambda = z$lambda,
             converged = z$converged,
             iter = z$iter))
}

# Compute logELR using algorithm proposed in Chen (2008)

logELRchen <- function(X, # Data matrix
                  mu, # Desired mean
                  max_iter, # Maximum number of iterations
                  tol # desired tolerance
){
  n <- nrow(X)
  d <- ncol(X)

  # Center data, Z[i,]=X[i,]-mu
  Z <- t(t(X) - mu)

  # Initialize
  lambda <- rep(0, d)
  lam_z <- rep(1, n) # lam_z[i] = 1+lambda^t(Z[i,]) corresponding to lambda =
      rep(0,d)
  gamma <- 1
  k <- 0
  converged <- F
  repeat{
    # If too many iterations
    if (k >= max_iter) {
      weights <- (n * lam_z)^-1
      if (any(lam_z<1/n)){
        logELRchen <- -Inf
      }
      else{
        logELRchen <- -sum(log(lam_z)) }
      return(list(logELRchen = logELRchen,
                 weights = weights,
                 sum = sum(weights),
                    lambda = lambda,
                    converged = converged,
                    iter = k))
    }

    # Compute gradient, Hessian, Newton step
    grad <- -colSums(Z*1/lam_z)
    hess <- -t(Z) %*% (-Z /lam_z^2)
    delta_lam <- -gamma * solve(hess, grad)

    # Initialize line search
    line_iter <- 0
    logELRchen <- -sum(log(lam_z))
    z_delta_lam <- as.vector(Z %*% delta_lam)

    # If step is too small stop
    if (sqrt(sum(delta_lam*delta_lam)) < tol) {
      break
    }

    # Line Search
    repeat {
      if (line_iter >= max_iter) {
        stop("Cannot find the next feasible step.")
        break
      }

      new_lam_z <- lam_z + z_delta_lam
      new_logELRchen <- -sum(log(new_lam_z))

      if (any(new_lam_z <= 0) | new_logELRchen > logELRchen) {
        delta_lam <- delta_lam / 2
        z_delta_lam <- z_delta_lam / 2
        line_iter <- line_iter + 1
      }
      else {
        break
      }
    }

    lambda <- lambda + delta_lam
    lam_z <- new_lam_z
    k <- k + 1
    gamma <- k^-0.5
  }
  logELRchen <- -sum(log(lam_z))
  weights <- (n * lam_z)^-1
```

```
  converged <- T
  return(list(logELRchen = logELRchen,
              weights = weights,
              sum = sum(weights),
              lambda = lambda,
              converged = converged,
              iter = k))
}


#### Adjusted emperical likelihood Computation

# Compute logAELR using the algorithm proposed in Chen (2008).
logAELRchen <- function(X, # Data matrix
                     mu, # Desired mean
                     max_iter, # Maximum number of iterations
                     tol # desired tolerance
){
  n <- nrow(X)
  d <- ncol(X)
  a_n <- max(1,log(n)/2)
  # Center data, Z[i,]=X[i,]-mu
  Z <- t(t(X) - mu)
  z_new <- -a_n*colMeans(Z)
  Z <- rbind(Z,z_new)

  # Uncenter data
  X <- t(t(Z)+mu)
  z <- logELRchen(X, mu, max_iter, tol)
  return(list(logAELRchen = z$logELRchen,
              weights = z$weights,
              sum = z$sum,
              lambda = z$lambda,
              converged = z$converged,
              iter = z$iter))
}

# Computed logAELR using a damped Newton method
logAELRsc <- function(X, # Data matrix
                    mu, # Desired mean
                    max_iter, # Maximum number of iterations
                    tol # desired tolerance
){
  n <- nrow(X)
  d <- ncol(X)
  a_n <- max(1,log(n)/2)
  # Center data, Z[i,]=X[i,]-mu
  Z <- t(t(X) - mu)
  z_new <- -a_n*colMeans(Z)
  Z <- rbind(Z,z_new)

  # Uncenter data
  X <- t(t(Z)+mu)
  z <- logELRdamp(X, mu, max_iter, tol, ord = 4)
  return(list(logAELRdamp = z$logELRdamp,
              weights = z$weights,
              sum = z$sum,
              lambda = z$lambda,
              converged = z$converged,
              iter = z$iter))
}
```

8.2 Computing confidence regions

Below is the R code for used to compute the confidence regions for 1 and 2 dimensional data.

```
## Computes the point x = colmean(X)+ t*dir such that f(x)=alpha,
level_dir <- function(f, # convex function such -2ELR ir -2AELR
                    X, # Data matrix
                    alpha, # Desired alpha
                    tol, # Tolerance
                    dir # Direction vector
                    ){
    b <- colMeans(X)
    repeat{
      b <- b + dir
      f_b <- f(b)
      if (f_b > alpha){
        break
      }
    }

    a <- b-dir
    k <- 0
    while(2^-k> tol){
      c <- (a+b)/2
      f_c <- f(c)
      if(f_c>alpha){
        b <- c
      }
      else{
        a <- c
      }
      k <- k+1
    }
  return(c)
}


## Compute confidence interval of significance alpha

conv_int <- function(f, # convex function such -2ELR ir -2AELR
                     X, # Data matrix
                     alpha, # Desired confidence
                     tol # Tolerance
){
  upper <- level_dir(f,X,qchisq(alpha, df=1), tol, 1)
  lower <- level_dir(f,X,qchisq(alpha, df=1), tol, -1)
  return(c(lower,upper))
}


## Computes the points on the boundary of the confidence region of significance
    alpha

level <- function(f, # convex function such -2ELR ir -2AELR
                  X, # Data matrix
                  alpha, # Desired confidence
                  tol, # Tolerance
                  num_dir # Number of direction
                  ){
  # Create direction vectors
  theta <- seq(from = 0, 2*pi, length.out = num_dir)
  theta <- c(theta,0)
  v <- matrix(NA, nrow = num_dir+1, ncol = 2)
  for (i in 1:(num_dir+1)){
    v[i,] <- c(cos(theta[i]),sin(theta[i]))
  }
  levelset <- matrix(NA, nrow = num_dir+1, ncol = 2)
  for(i in 1:(num_dir+1)){
    levelset[i,] <- level_dir(f,X,qchisq(alpha, df=2), tol, v[i,])
  }
  return(levelset)
}

## Plots the convex hull of a set of 2 dimensional points
Plot_ConvexHull<-function(X, lcolor){
  hpts <- chull(x = X[,1], y = X[,2])
  hpts <- c(hpts, hpts[1])
  lines(X[hpts,1], X[hpts,2], col = lcolor)
```

```
}


ELR <- function(mu){
  return(-2*logELRsc(X,mu,100,1E-8)$logELRsc)
}

AELR <- function(mu){
  return(-2*logAELRdamp(X,mu,100,1E-8)$logAELRdamp)
}
```

8.3 tests

Below is the R code for used to conduct out tests.

```
## 1- dimension Poission mean 3
set.seed(072892)
X <- as.matrix(rpois(25, 3),ncol=1, nrow=20)
mu <- mean(X)
hist(X,main="Poisson data n=30")
range(X)

# mu = 3
logELRchen(X,c(3),100,1E-8)
logELRtrad(X,c(3),100,1E-8)
logELRsc(X,c(3),100,1E-8)
logAELRchen(X,c(3),100,1E-8)
logAELRsc(X,c(3),100,1E-8)

# mu = 6
logELRchen(X,c(6),100,1E-8)
logELRtrad(X,c(6),100,1E-8)
logELRsc(X,c(6),100,1E-8)
logAELRchen(X,c(6),100,1E-8)
logAELRsc(X,c(6),100,1E-8)

# mu = 1.0001
logELRchen(X,c(1.0001),100,1E-8)
logELRtrad(X,c(1.0001),100,1E-8)
logELRsc(X,c(1.0001),100,1E-8)
logAELRchen(X,c(1.0001),100,1E-8)
logAELRsc(X,c(1.0001),100,1E-8)

conv_int(ELR,X,0.95 ,1E-8)
conv_int(AELR,X,0.95 ,1E-8)

conv_int(ELR,X,0.5 ,1E-8)
conv_int(AELR,X,0.5 ,1E-8)


## 2-dimensional gaussian
set.seed(072892)
X <- rmvnorm(30, mean= c(0,0), sigma = diag(c(2,10)))
plot(X, pch=20, main = "Multivariate Gaussian, n=30")
mu <- colMeans(X)
mu
Plot_ConvexHull(X, lcolor = "black")
points(colMeans(X)[1],colMeans(X)[2],col= "red", pch=17)
points(0,0,col= "green", pch=17)
points(1,2,col= "green", pch=17)
points(-2.5,3.1,col= "green", pch=17)

# mu = c(0,0)
logELRchen(X,c(0,0),100,1E-8)
logELRtrad(X,c(0,0),100,1E-8)
logELRsc(X,c(0,0),100,1E-8)
logAELRchen(X,c(0,0),100,1E-8)
logAELRsc(X,c(0,0),100,1E-8)


# mu = c(1,2)
logELRchen(X,c(1,2),100,1E-8)
logELRtrad(X,c(1,2),100,1E-8)
logELRsc(X,c(1,2),100,1E-8)
logAELRchen(X,c(1,2),100,1E-8)
logAELRsc(X,c(1,2),100,1E-8)


# mu = c(-2.5,3.1)
logELRchen(X,c(-2.5,3.1),100,1E-8)
logELRtrad(X,c(-2.5,3.1),100,1E-8)
logELRsc(X,c(-2.5,3.1),100,1E-8)
logAELRchen(X,c(-2.5,3.1),100,1E-8)
logAELRsc(X,c(-2.5,3.1),100,1E-8)



z <- level(ELR,X,0.5 ,1E-8,200)
lines(z,type = "l",col= "green")
z <- level(ELR,X,0.95,1E-8,200)
lines(z,type = "l",col= "green")
```

```
z <- level(AELR,X,0.5,1E-8,200)
lines(z,type = "l",col= "blue")
z <- level(AELR,X,0.95,1E-8,200)
lines(z,type = "l",col= "blue")
```