Preprint SMU-HEP-10-14

Multi-dimensional Ellipsoidal Fitting

Bridget Bertoni
Department of Physics, Southern Methodist University, and
Department of Physics, University of Washington, Seattle
(Dated: August 20, 2010)

The problem of fitting ellipsoids occurs in many areas of science. It is useful in pattern recognition, particle physics, computer graphics, medical imaging of organs, and statistical error analysis. We describe an algorithm for finding an equation for a multi-dimensional ellipsoid fit to a distribution of discrete data points. In general, we find that the algorithm works well for fitting ellipsoids with a known center, and that additional work is needed when the center of the ellipsoid is unknown.

I. INTRODUCTION

The approach to ellipsoidal fitting used in this write-up, based primarily on the algorithms described in [1] and [2], was motivated by the application of ellipsoidal fitting to error analysis. In particular, it could be used in the analysis of parton distribution functions, along the lines of the analysis described in [3]. Since the emphasis here is not on 3D visualization, to the author's knowledge, this is the first implementation of d-dimensional ellipsoid fitting for an arbitrarily large number of dimensions.

This implementation of ellipsoid fitting reconstructs the d-dimensional ellipsoid from its two-dimensional ellipse projections. As shown in [1] and [2], ellipsoid fitting can be expressed simply and concisely in terms of linear algebra. This will be explained in detail in sections II, III, IV, and V. Section VI contains a brief overview of the Fortran 77 version of the ellipsoid fitting code, Section VII covers its usage in Fortran. and its performance is assessed in section VIII.

II. MATHEMATICAL REPRESENTATION OF AN ELLIPSOID

A d-dimensional ellipsoid is a d-dimensional surface–the extension of a two-dimensional ellipse. It can be thought of as an affine map (linear transformation plus a translation) of a d-dimensional sphere. Hence, as a set of points, an ellipsoid can be most generally expressed in terms of the following set:

$$\{\mathbf{A}\mathbf{x}+\mathbf{b} \ : \ \mathbf{x}\in S_1(0)\} \ , \tag{1}$$

where $\mathbf{A}$ is a d x d matrix, $\mathbf{b}$ is a d-dimensional translation vector, and $S_1(0)$ is the surface of a unit ball centered at the origin in d Euclidean dimensions:

$$S_1(0)=\left\{\mathbf{y}\in\Re^d \ : \ ||\mathbf{y}||=\mathbf{y}^T\mathbf{y}=1\right\}. \tag{2}$$

Note that $\mathbf{b}$ defines the center of the ellipsoid. Using the change of variables, $\mathbf{z}=A\mathbf{x}+\mathbf{b}$, in (1), we find that an ellipsoid can be represented by a set of points

$$\{\mathbf{z} \ : \ (\mathbf{z}-\mathbf{b})^T\mathbf{C}(\mathbf{z}-\mathbf{b})=1, z\in\Re^d\} \ , \tag{3}$$

where $\mathbf{C}=(\mathbf{A}\mathbf{A}^T)^{-1}$, and $\mathbf{C}$ is both symmetric and positive definite (for a non-degenerate d-dimensional ellipsoid in d dimensions). A degenerate $\bar{d}$-dimensional ellipsoid in d-dimensions (with $\bar{d}<d$) would have $\mathbf{C}$ as a positive semi-definite (PSD) matrix.

Since $\mathbf{C}$ is symmetric, we can rewrite (3) as

$$\{\mathbf{z} \ : \ \mathbf{z}^T\bar{\mathbf{C}}\mathbf{z} + \bar{\mathbf{b}}^T\mathbf{z} = 1, z \in \Re^d\} \ , \tag{4}$$

where

$$\bar{\mathbf{C}} = \frac{\mathbf{C}}{1 - \mathbf{b}^T\mathbf{C}\mathbf{b}} \quad \text{and} \quad \bar{\mathbf{b}}^T = \frac{-2\mathbf{b}^T\mathbf{C}}{1 - \mathbf{b}^T\mathbf{C}\mathbf{b}}. \tag{5}$$

Note that the term linear in $\mathbf{z}$ disappears for an ellipsoid centered at the origin. The equality sign in (3) emphasizes that we are only dealing with the surface of an ellipsoid. If it were replaced by a less than or equal to sign, we would have the equation for a filled ellipsoid.

Further, we know that a centered ellipsoid aligned with the coordinate axes obeys the formula

$$\sum_{i=1}^{d} \frac{z_i^2}{R_i^2} = 1 \ , \tag{6}$$

where R_i are the semi-axis lengths of the ellipsoid. By analogy to this, we can see that the eigenvectors of $\bar{\mathbf{C}}$ are the axes of the ellipsoid and if c_i are the eigenvalues of $\bar{\mathbf{C}}$, then the semi-axis lengths of the ellipsoid are given by $1/\sqrt{c_i}$.

III. TWO-DIMENSIONAL ELLIPSE FITTING

Our goal is to take two-dimensional projections of data points on a d-dimensional ellipsoid and then to reconstruct the full ellipsoid using the projections, we need a method to find the elliptic boundary of the projected points. The boundary of a set of data points can be found by using well-established convex hull algorithms, such as the one in [4] or those in the FastGEO library (available at http://www.partow.net/projects/fastgeo/index.html).

Once the boundary is found, it can be fit to a two-dimensional ellipse. The algorithm we use is the least squares minimization method described by Fitzgibbon *et al.* in [1]. They start by defining the "algebraic distance" of a point (x, y) to the surface of a general conic by

$$F(\mathbf{a}, \mathbf{x}) = \mathbf{a} \cdot \mathbf{x} = ax^2 + bxy + cy^2 + dx + ey + f \ , \tag{7}$$

where $\mathbf{a} = (a \ b \ c \ d \ e \ f)^T$, and $\mathbf{x} = (x^2 \ xy \ y^2 \ x \ y \ 1)^T$. The conic is defined by $F(\mathbf{a}, \mathbf{x}) = 0$, and so a conic may be fit to a set of points through a least squares minimization of algebraic distances.

In order to ensure that the best fit conic found is an ellipse, one only has to impose the constraint that $b^2 - 4ac < 0$. This can be done by noticing that (7) allows for arbitrary rescaling of all the coefficients. Hence one can instead impose the constraint $4ac - b^2 = 1$, which can be expressed as $\mathbf{a}^T\mathbf{C}\mathbf{a} = 1$, where

$$C = \begin{pmatrix} 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}. \tag{8}$$

Introducing a matrix $\mathbf{D}$ defined by $\mathbf{D} = (\mathbf{x}_1 \ \mathbf{x}_2 \ \ldots \ \mathbf{x}_N)$, where N is the total number of points, the problem becomes an optimization problem of minimizing $\sum_{i=1}^{N} F(\mathbf{a}, \mathbf{x}_i)^2 = ||\mathbf{D}\mathbf{a}||^2$, subject to the constraint $\mathbf{a}^T\mathbf{C}\mathbf{a} = 1$.

This can be solved using a Lagrange multiplier approach. Thus introducing a Lagrange multiplier, λ, we need to minimize the quantity

$$||\mathbf{Da}||^2 - \lambda \mathbf{a}^T \mathbf{Ca}. \tag{9}$$

Differentiating (9) with respect to $\mathbf{a}$, we see that we just need to find the vector $\mathbf{a}$ that satisfies

$$\begin{cases} \mathbf{Sa} = \lambda \mathbf{Ca} \ , \\ \mathbf{a}^T \mathbf{Ca} = 1 \ , \end{cases} \tag{10}$$

where $\mathbf{S} = \mathbf{D}^T \mathbf{D}$. The equation for the solution ellipse is then given by $F(\mathbf{a}, \mathbf{x}) = 0$ (where $\mathbf{x}$ is *not* a point on the ellipse, but is simply the coordinate vector given below (7)). The solution $\mathbf{a}$ is one of six generalized eigenvectors obtained from solving the first equation in (10). In [1], Fitzgibbon *et al.* further show that of the six eigenvectors, only one will be associated with a positive eigenvalue, and that this eigenvector is the desired solution.

However it is important to notice that when all of the N data points lie on the ellipse, $\mathbf{Sa} = 0$ and so the solution $\mathbf{a}$ is associated with a zero eigenvalue and not a positive eigenvalue. In fact, as pointed out in [5], due to numerical error, it is possible that the eigenvalue that labels the solution eigenvector is even a small negative number. Appropriate numerical precautions should be taken.

One can handle this situation by separating it into three cases: #1 rank($\mathbf{S}$) = 6, #2 rank($\mathbf{S}$) = 5, and #3 rank($\mathbf{S}$) < 5. (Of course here the equality and inequality symbols are not strict and are used loosely to mean 'within numerical error.') The first case is the ideal one in which one may simply find the generalized eigenvalues of (10) and pick the eigenvector associated with the positive eigenvalue.

In the second case, either only five points were given, or all input points lie on the ellipse, and numerical rounding may make it difficult to choose the correct eigenvector every time. Since $\mathbf{S}$ is symmetric, one can instead calculate the singular value decompostion (SVD) of $\mathbf{S}$, $\mathbf{S} = \mathbf{O}^T \Sigma \mathbf{O}$ where $\mathbf{O}$ is orthogonal and Σ is a diagonal matrix with the singular values of $\mathbf{S}$ along its diagonal. Since rank($\mathbf{S}$) = 5, one of the diagonal entries of Σ is zero–without loss of generality, say that $\Sigma_{66} = 0$. Also since

$$\mathbf{OSa} = \mathbf{OSO}^T \mathbf{Oa} = \Sigma \mathbf{Oa} \equiv \Sigma \mathbf{u} = 0 \ , \tag{11}$$

$\mathbf{u} = (0\ 0\ 0\ 0\ 0\ u_6)^T$, and u_6 may be found explicitly from $\mathbf{a}^T \mathbf{Ca} = 1$. Also,

$$\mathbf{a} = \mathbf{O}^T \mathbf{Oa} = \mathbf{O}^T \mathbf{u} \ , \tag{12}$$

and the solution vector, $\mathbf{a}$, can be taken to as equal to the sixth column of $\mathbf{O}^T$ (since the solution may be arbitrarily scaled). This method is preferred since finding the SVD of $\mathbf{S}$ is both computationally less expensive and less ambiguous than finding the eigenvalues and eigenvectors of $\mathbf{S}$.

The third case of rank($\mathbf{S}$) < 5 occurs either when too few points have been specified or when the data points are arranged in such a way that numerically, rank((S)) < 5. In this case there is no unique solution to (10). Note that at least five points are needed to uniquely determine a two-dimensional ellipse. In general, (4) shows that for an ellipsoid in d dimensions, $d(d+3)/2$ points are needed to uniquely determine the ellipsoid.

IV. FITTING A MULTI-DIMENSIONAL ELLIPSOID WITH A KNOWN CENTER

For this part, the algorithm is the one developed by Karl in [2]. The goal of Karl's algorithm is to fit a d-dimensional ellipsoid to d-dimensional data points using lower dimensional projections of the ellipsoid, provided that the center is known. If the center of the ellipsoid is known, it is easy to translate the ellipsoid to the origin where the math is cleaner, so all calculations are carried out as though the ellipsoid were centered at the origin. In this case, the ellipsoid can be completely represented by a symmetric, PSD matrix.

The setup of the problem is as follows. Let $\mathbf{X}$ be a d x d symmetric, PSD matrix representing the d-dimensional ellipsoid and let $\mathbf{P}$ be a d x m non-degenerate projection matrix whose m columns form an orthonormal basis for an m-dimensional subspace of the d-dimensional space. The m x m symmetric, PSD matrix, $\mathbf{Y}$, representing the m-dimensional projection of the ellipsoid is then given by

$$\mathbf{Y} = \mathbf{P}^T \mathbf{X} \mathbf{P}. \tag{13}$$

Hence there is a nice, linear relationship between the PSD representation of the d-dimensional ellipsoid, $\mathbf{X}$, and its m-dimensional projection, $\mathbf{Y}$. Even so, Karl introduces a more convenient and non-redundant way to express this relationship, exploiting the symmetry of the $\mathbf{X}$ and $\mathbf{Y}$ matrices. Karl introduces a symmetric matrix inner product space, in which d-dimensional symmetric matrices are represented by a $d(d+1)/2$-dimensional vectors, and the inner product between two symmetric matrices $\mathbf{A}$ and $\mathbf{B}$ is given by $\langle \mathbf{A}, \mathbf{B} \rangle = \mathrm{tr}(\mathbf{A}^T \mathbf{B})$.

Let $\mathbf{x}$ and $\mathbf{y}$ be the symmetric space vector representations of the ellipsoids in terms of the symmetric matrices $\mathbf{X}$ and $\mathbf{Y}$ in the orthonormal bases $\mathbf{M}_j^{(d)}$ and $\mathbf{M}_i^{(m)}$. Then (13) is expressed simply as

$$\mathbf{y} = \tilde{\mathbf{P}} \mathbf{x} \ , \tag{14}$$

where the components of $\mathbf{x}$, $\mathbf{y}$, and $\tilde{\mathbf{P}}$ are given by

$$\begin{aligned} x_j &= \langle \mathbf{X}, \mathbf{M}_j^{(d)} \rangle \\ y_i &= \langle \mathbf{Y}, \mathbf{M}_i^{(m)} \rangle \\ \tilde{P}_{ij} &= \langle \mathbf{M}_i^{(m)}, \mathbf{P}^T \mathbf{M}_j^{(d)} \mathbf{P} \rangle \end{aligned} \tag{15}$$

A simple form to use for the orthonormal basis is the set of matrices which contain all zeros except for either a 1 in a single location along the diagonal or a $1/\sqrt{2}$ in two locations, symmetric about the diagonal (e.g. $\mathbf{M}_{kl}^{(d)} = \mathbf{M}_{lk}^{(d)} = 1/\sqrt{2}$ for $1 < l,\ k < d$ and $l \neq k$).

If there are q ellipsoid projections, then the solution vector $\mathbf{x}$ is obtained by stacking individual projection equations as in (14) and solving them simultaneously:

$$\left(\begin{array}{c} \mathbf{y}_1 \\ \hline \mathbf{y}_2 \\ \hline \vdots \\ \hline \mathbf{y}_q \end{array} \right) = \left(\begin{array}{c} \tilde{\mathbf{P}}_1 \\ \hline \tilde{\mathbf{P}}_2 \\ \hline \vdots \\ \hline \tilde{\mathbf{P}}_q \end{array} \right) \mathbf{x}. \tag{16}$$

For simplicity we rewrite this as

$$\mathbf{y}_{\text{stack}} = \tilde{\mathbf{P}}_{\text{stack}} \mathbf{x}. \tag{17}$$

A unique solution for $\mathbf{x}$ exists if and only if the matrix $\tilde{\mathbf{P}}_{\text{stack}}$ has full column rank. In this case, the solution is given by

$$\mathbf{x} = \tilde{\mathbf{P}}_{\text{stack}}^{+} \mathbf{y}_{\text{stack}} = \left[\left(\tilde{\mathbf{P}}_{\text{stack}}^T \tilde{\mathbf{P}}_{\text{stack}} \right)^{-1} \tilde{\mathbf{P}}_{\text{stack}}^T \right] \mathbf{y}_{\text{stack}} \ , \tag{18}$$

where $\tilde{\mathbf{P}}_{\text{stack}}^{+}$ is the full column rank Moore-Penrose inverse of $\tilde{\mathbf{P}}_{\text{stack}}$. The Moore-Penrose inverse is a pseudoinverse which provides the least squares solution to an over- or under-determined problem.

Note that the final solution for $\mathbf{x}$ does not necessarily represent a PSD matrix. Hence the final solution is not guarenteed to be an ellipsoid, and additional constraints must be employed to attain this. Karl suggests methods for adding a PSD constraint to the final solution in [2]. An analytical form for computing the nearest (in the Frobenius norm) symmetric PSD matrix to an arbitrary real matrix is also given in [6].

V. FINDING THE CENTER OF A D-DIMENSIONAL ELLIPSOID

The center of a d-dimensional ellipsoid may be found in a manner along the lines of that introduced in the previous section. Recalling the equation given for an ellipsoid with an arbitrary center given by (4), consider the equivalent stacked matrix equation

$$\left(\frac{\mathbf{z}}{\sqrt{\mathbf{z}}}\right)^T \left(\begin{array}{c|c} \bar{\mathbf{C}} & \mathbf{0} \\ \hline \mathbf{0} & \mathrm{Diag}[\bar{\mathbf{b}}] \end{array}\right) \left(\frac{\mathbf{z}}{\sqrt{\mathbf{z}}}\right) = 1 \ , \tag{19}$$

where extra minus signs obscured by taking the square root have been absorbed into $\mathrm{Diag}[\bar{\mathbf{b}}]$. Finding $\bar{\mathbf{C}}$ and $\bar{\mathbf{b}}$ is then possible by using an additionally stacked version of the algorithm described in the previous section.

Let $\mathbf{c}$ denote the vector describing the center of the d-dimensional ellipsoid. The center of the ellipsoid can then be obtained from $\bar{\mathbf{C}}$ and $\bar{\mathbf{b}}$ by solving the following matrix equation:

$$\mathbf{c} = (-2\bar{\mathbf{C}})^{-1}\bar{\mathbf{b}}. \tag{20}$$

VI. THE CODE

The code, written to fit a d-dimensional ellipsoid to d-dimensional data points, takes two-dimensional projections of the data, fits ellipses to the two-dimensional projections using the method given in Section III, and reconstructs the d-dimensional ellipsoid from the two-dimensional projections using the method given in Section IV.

In particular, all possible two-dimensional projections of the ellipsoid onto planes defined by two of the coordinate axes are used. All $d(d-1)/2$ projections are needed to ensure that $\tilde{\mathbf{P}}$ has full column rank. Additionally, when using these projections, the calculation of the $\tilde{\mathbf{P}}_i$ is simplified to the point where they can be reduced to a few if-then statements rather than having the code create the bases and do all of the matrix multiplications.

VII. USAGE IN FORTRAN

The ellipsoid fitting program uses the LAPACK (with BLAS) linear algebra library (available at http://www.netlib.org/lapack/) which needs to be installed in order for it to work. The program can be called as a Fortran subroutine:

subroutine Ellipsoid_fit(*inputdata*, *ndim*, *maxpoints*, *needcenter*, *printinfo*,
FinalMatrix, *eigenvecs*, *eigenvals*, *center*).

Input parameters:

- *inputdata* is a double precision array; *inputdata*(*ndim*, *maxpoints*). It contains all (*maxpoints*) of the *ndim*-dimensional points. The first index (or row) of the array *inputdata* labels the axis, i.e. the i^{th} coordinate of the *ndim*-dimensional point.
- *ndim* is an integer parameter which specifies the number of dimensions of the to-be-reconstructed ellipsoid.
- *maxpoints* is an integer parameter which is equal to the number of data points inputted.
- *needcenter* is an integer.
  - *needcenter* = 0 forces the ellipse to have a center at the origin, and
  - *needcenter* = 1 tells the subroutine to find the center of the ellipse.
- *printinfo* is an integer.

- *printinfo* = 0 prints nothing, and
- *printinfo* = 1 prints out the singular values and rank of the matrix S, which is used to fit the two-dimensional ellipse projections. It also prints out the least squares error in each two-dimensional fit.
- *printinfo* = 2 prints everything that *printinfo* = 1 prints. It also prints out the ellipse axes and the corresponding axis lengths.

Output parameters:

- *FinalMatrix* is a double precision array; *FinalMatrix*(*ndim*, *ndim*) which contains the positive semi-definite (PSD) matrix representation of the centered ellipse.
- *eigenvecs* is a double precision array; *eigenves*(*ndim*, *ndim*) which contains the *ndim*-dimensional axes of the reconstructed ellipse side by side in its columns (second index). These are the eigenvectors of *FinalMatrix*.
- *eigenvals* is a double precision array; *eigenvals*(*ndim*) which contains the *ndim* axis lengths of the ellipse. They are listed in the same order as the axes of the ellipse in the columns of *eigenvecs*. These are the eigenvectors of *FinalMatrix*.
- *center* is a double precision array; *center*(*ndim*) which is a vector representing the center of the ellipse.

VIII. CODE PERFORMANCE

The code is fast and in general, it performs well. Several examples are shown below in Fig. 1.

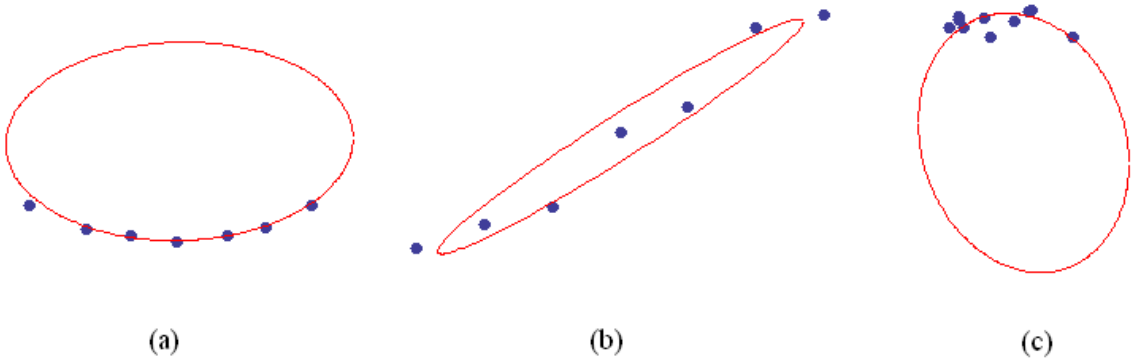

FIG. 1: (a) shows a two-dimensional ellipse fit to slightly noisy elliptic data, (b) shows a two-dimensional ellipse fit to noisy linear data, and (c) shows a two-dimensional projection of a three-dimensional ellipsoid fit to noisy data. The ellipse in (c) does not look like the best fit ellipse since it is just a projection of a best fit ellipsoid and in this case, the ellipsoid fit was forced to be centered at the origin.

There are some caveats to this approach. The method of finding the center of the ellipsoid is unstable and does not work very well with noisy input data points. Additionally, though the code guarantees that for each two-dimensional projection, the data will be fit to an ellipse, there is (currently) no positive semi-definite (PSD) constraint on the final matrix representing the *d*-dimensional ellipsoid, hence the final solution is not guaranteed to be an ellipsoid. Yet using simulated data with a known center, a final ellipsoid is almost always obtained. See [2] for a quantitative description of the amount of noise in the input data which will still guarantee a solution which is PSD. Adding a final PSD constraint and improving the method of finding the equation of a non-centered, *d*-dimensional ellipsoid are two paths for further work on this program.

Acknowledgements

This work was done as a part of a 2010 summer research project in the Physics Department at Southern Methodist University in Dallas, TX. It was supported by P. Nadolsky's start-up grant. The author thanks P. Nadolsky for his guidance in this work.

[1] A. Fitzgibbon, M. Pilu, and R. Fisher, Direct least square fitting of ellipses. IEEE Trans. PAMI, 21 (5), 1999

[2] W. C. Karl, Reconstructing objects from projections. PhD Thesis, MIT, Dept. of Electrical Engineering and Computer Science, 1991.

[3] H. L. Lai, J. Huston, Z. Li, P. Nadolsky, J. Pumplin, D. Stump, and C.-P. Yuan, Uncertainty induced by QCD coupling in CTEQ-TEA global analysis of parton distributions, arXiv:1004.4624v2 [hep-ph].

[4] C. Barber, D. Dobkin, and H. Huhdanpaa. The Quickhull algorithm for convex hulls. ACM Trans. on Mathematical Software, 22:469-483, 1997.

[5] R. Halíř, J. Flusser, Numerically stable direct least squares fitting of ellipses. In Skala, V., ed.: Proc. Int. Conf. in Central Europe on Computer Graphics, Visualization and Interactive Digital Media (WSCG98), 1998.

[6] N. J. Higham, Computing a nearest symmetric positive semi-definite matrix, Linear Algebra Appl., 103, pp. 103118, 1988.