Nonlinear Parameter Estimation

Q550: Models in Cognitive Science

- We want to estimate the optimum parameters for our cognitive model using an objective function. Then we can quantitatively compare our models.

Some Issues in Parameter Estimation:

1. *Identifiability and limits*
2. *Linear vs. nonlinear models*
3. *Data considerations/Fitting approaches*

Parameter Identification

- Is it possible to obtain a unique optimal solution?
  - Must have more data points than parameters
  - Can we arbitrarily set a parameter value and another parameter will compensate for it?

We originally assumed that output activations were the same for all individuals

$$P[A \mid x(t)] = \frac{e^{b \cdot o_A \cdot \gamma^t}}{e^{b \cdot o_A \cdot \gamma^t} + e^{b \cdot o_B \cdot \gamma^t}} = \frac{1}{1 + e^{-b(o_A - o_B)\gamma^t}}$$

It is not possible to estimate these parameters

b has no meaning in the presence of (o_C - o_I)

Parameter Identification

- The bias parameter is not identifiable in the presence of the output difference parameter (and vise versa)
- We could make these parameters identifiable by manipulating them in an experiment (instructions for speed vs. accuracy before or after learning).
- Gamma is identifiable: an arbitrary gamma selection **cannot** be compensated for by fitting other parameters

$$P\left[A \mid x(t)\right] = \frac{e^{b \cdot o_A \cdot \gamma^t}}{e^{b \cdot o_A \cdot \gamma^t} + e^{b \cdot o_B \cdot \gamma^t}} = \frac{1}{1 + e^{-b(o_A - o_B)\gamma^t}}$$

Linear vs. Nonlinear Models

- Estimating parameters in a cognitive model is similar to estimating regression coefficients
- However, regression models belong to the general linear class, whereas cognitive models are usually nonlinear…this effects our search for parameters
- **Linear model:** Unknown parameters are multiplied by known values, and these products are summed to yield the prediction: $\hat{Y} = \sum \beta_i x_i$
- It doesn't matter what the *x* values are
- Linear models satisfy a special condition: the average of the predictions from two different sets of parameters = the prediction produced by the average of the two sets of parameters. Nonlinear models do not satisfy this property

Linear vs. Nonlinear Models

- Estimating the parameters of a linear model can usually be done with a single-step algorithm that is guaranteed to produce an optimal solution
- There are no single-step solutions for estimating the parameters of a nonlinear model: we must search the (often complex) parameter space for the optimal solution
- There is no guarantee that the optimal solution will be found (but we can still try)

Data Considerations

- Before estimating parameters, we need to be clear about the data to be used.
- Assume that we have 10 participants per group, 11 delay conditions, and 200 observations per delay point

1. Aggregate modeling
2. Individual modeling
3. Hierarchical modeling

1. Aggregate Modeling

- Easiest approach: one set of parameters estimated for normal group; another set for the drunk group
- However, this assumes **no individual differences** within group…e.g., all individuals in the normal condition have the same decay rates
- Importance of individual differences (see Estes; Ashby & Maddox, etc.)
- A bad example from early learning theory: incremental strength learning models vs. all-or-none models. What if AON was generating model w/ individual differences on amount of training required to find solution to problem
- Cf. Exponential law of practice (Brown, Heathcote, & Mewhort, 2000, *PBR*)

2. Individual Modeling

- Estimate parameters for each individual separately
- Separate parameters (and even separate models) may be best fit for different individuals
- Requires a large amount of data from each subject
- Preferred method if parameters represent cognitive factors you expect to vary across subjects
- Compute the parameters for each group, then compare the parameter distributions across groups for significant differences…does the mean decay rate differ?

3. Hierarchical Modeling

- A compromise between the other two (see Van Zandt)
- Fit a single mixture model to all participants in each group…we assume a bivariate distribution of parameters across individuals in each group. We estimate the parameters within each group
- Requires a large number of participants
- Advantage over aggregate approach: Allows variability in parameters across individual differences
- Advantage over indiv approach: avoids fitting separate parameters to each person…if our assumption is true, provides more precise estimate of distribution of parameters over indiv approach

Comparing Approaches

- **Aggregate approach** is appropriate only if we have small number of subjects, and low variability in data
- **Indiv approach** is appropriate for experiments w/ a few subjects and large amount of data per subject
- **Hierarchical approach** is appropriate for experiments w/ a large number of participants but small amount of data per subject (e.g., educational testing)
- Lots of people, few data from each = poor esitmation of parameters by indiv model; Few people, lots of data = poor estimation of parameter distributions by hierarchical approach
- Bonus: Individual/aggregate approaches make no assumptions about the distribution of parameters

Estimating Parameters

- **Minimizing/maximizing objective functions**
- **The "Art" of hand-fitting**
- **Grid search, exhaustive enumeration** (example with our retention model of classification)
  - Grid search is impractical, and it's tough to visualize greater than 3-D parameter spaces
- **Some smarter ways of non-exhaustive searching in complex parameter spaces:**
  1. Gradient Descent
  2. Simulated Annealing
  3. Nelder-Meade Simplex
  4. Genetic Algorithms

Gradient Descent Algorithms

- Take steps downhill to determine parameter values that minimize the objective function

LOOP

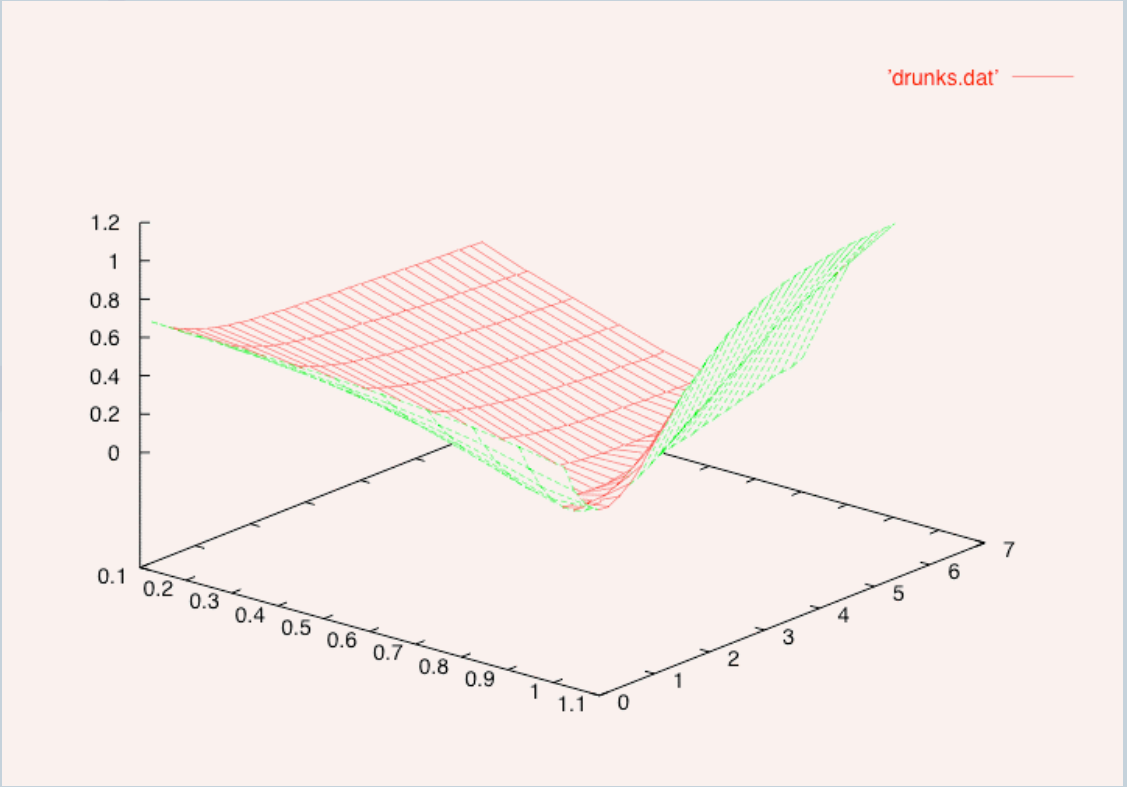
- Determine starting point--take a step to all possible adjacent grid points, and compute change in SSE
- Determine which point produces the steepest descent, and take a step in that direction
- We now have a new set of parameter values that are better starting values

UNTIL (Flat)

- Continue process until we reach a point where a move in any direction leads uphill…this is the set of parameters that minimizes the objective function
- Most programming languages and statistical packages (even graphing programs) have intrinsic functions for gradient descent minimization

Matlab: **fminsearch**()

R: **Optim**()



Constraints on Parameters:

- Remember to constrain your parameter search space to values that fall within theoretical boundaries

  e.g.: $0 \le \gamma \le 1$

  $0 \le b$

- Intrinsic functions may search outside the theoretical boundaries when trying to optimize. You may need to force the parameter space to stay within the theoretical boundaries

- Alternatively, you can recast your parameters so that they have no constraints. E.g., if decay needs to be constrained from 0-1, we could define decay as:

$$\gamma = \frac{1}{1 + e^{-v}}$$

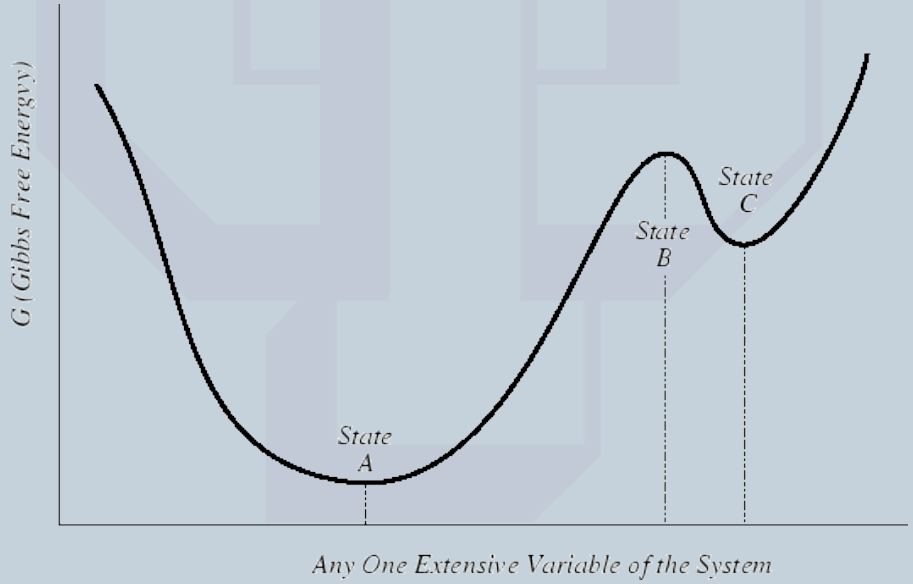
Algorithm searches for *v* values which are inserted into γ in the model

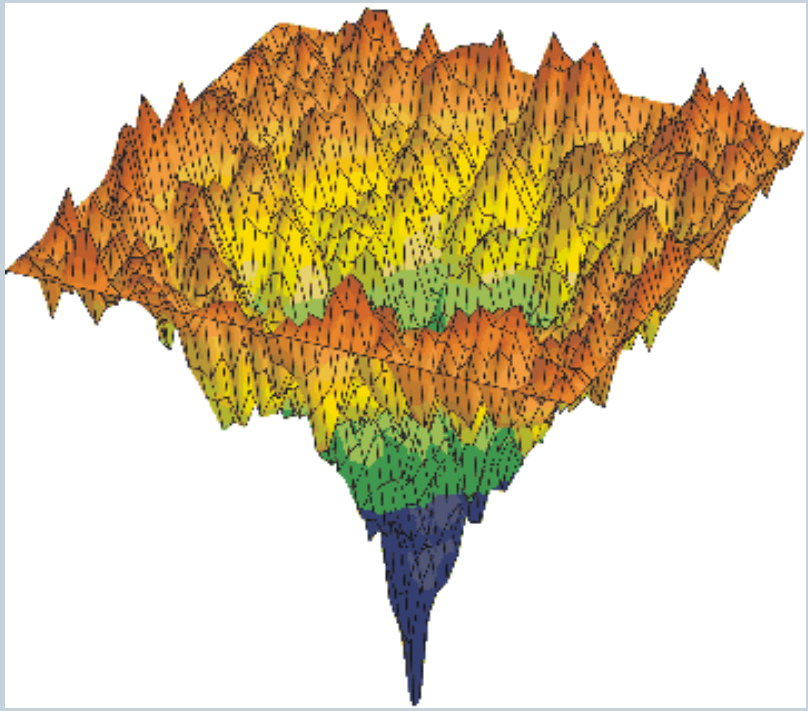
Flat Minimum Problem:

- If the parameter space is has a flat region, the search process may terminate prematurely b/c changed in the objective function are too small to detect improvements
- Near the minimum point, changes in one parameter can be compensated for by changes in another
- Flatness near the minimum produces parameter estimates with large standard errors

Local Minima Problem:

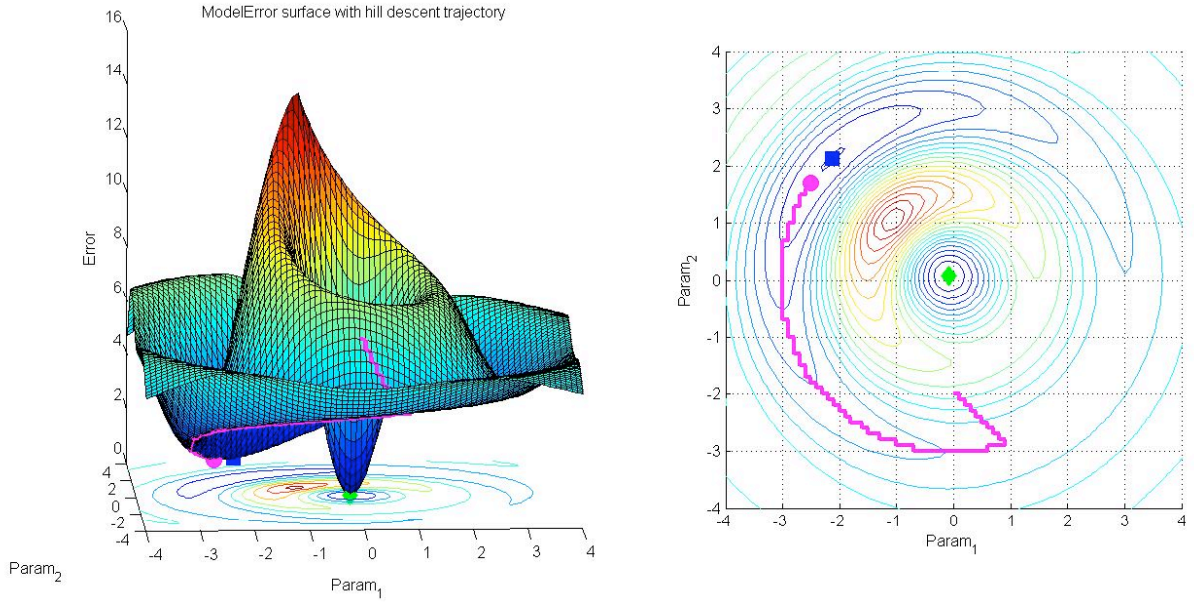
- Our examples have always had one minimum….this is common w/ a small # of parameters, but is certainly not guaranteed
- With multiple minima, gradient descent algorithms won't necessarily find the global minimum…it depends on where you start





Local Minima Problem:

- One way to avoid getting stuck in a local minimum is to try several starting positions
- As the # of parameters increases, the possibility of more local minima increases



Simulated Annealing

- Useful in highly nonlinear spaces with large # of parms
- Based on annealing in metallurgy: heating/cooling of a material to increase the size of its crystals and reduce their defects
- If you were fitting 100 parameters with a grid search method, you *could* find the global minimum in several million years of computation
  - Simulated annealing will quickly find a “pretty good” solution
  - Cf. Particle swarm algorithm

Simulated Annealing

- Select a starting position

LOOP

- Randomly select a new position and evaluate it compared to old position
- With probability *p*, select superior; w/ probability (1-*p*) select inferior
- Increment *p*

UNTIL (Flat)

Early in search, bounces up and down a lot to escape local minima; later in search, converges toward global minimum

Nelder-Meade Simplex

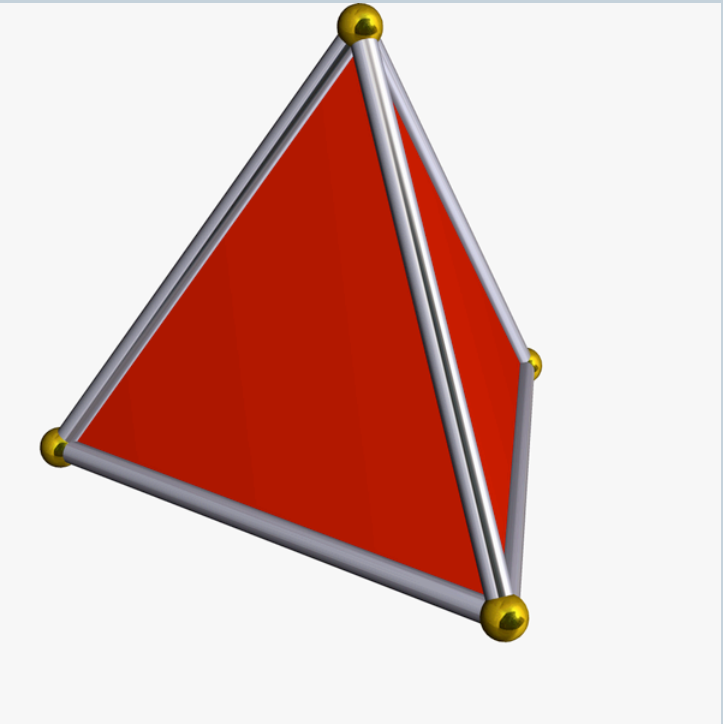
- Gradient descent requires smooth parameter spaces to compute gradient--otherwise, use a non-derivative based search
- **Simplex:** minimize the area of a tetrahedron in n-dimensional space (aka: tripod search)

Tetrahedron (3-simplex)

Petachoron (4-simplex)

Hexa-5-tope (5-simplex)

…etc.



Nelder-Meade Simplex

- Select 3 random points for the simplex

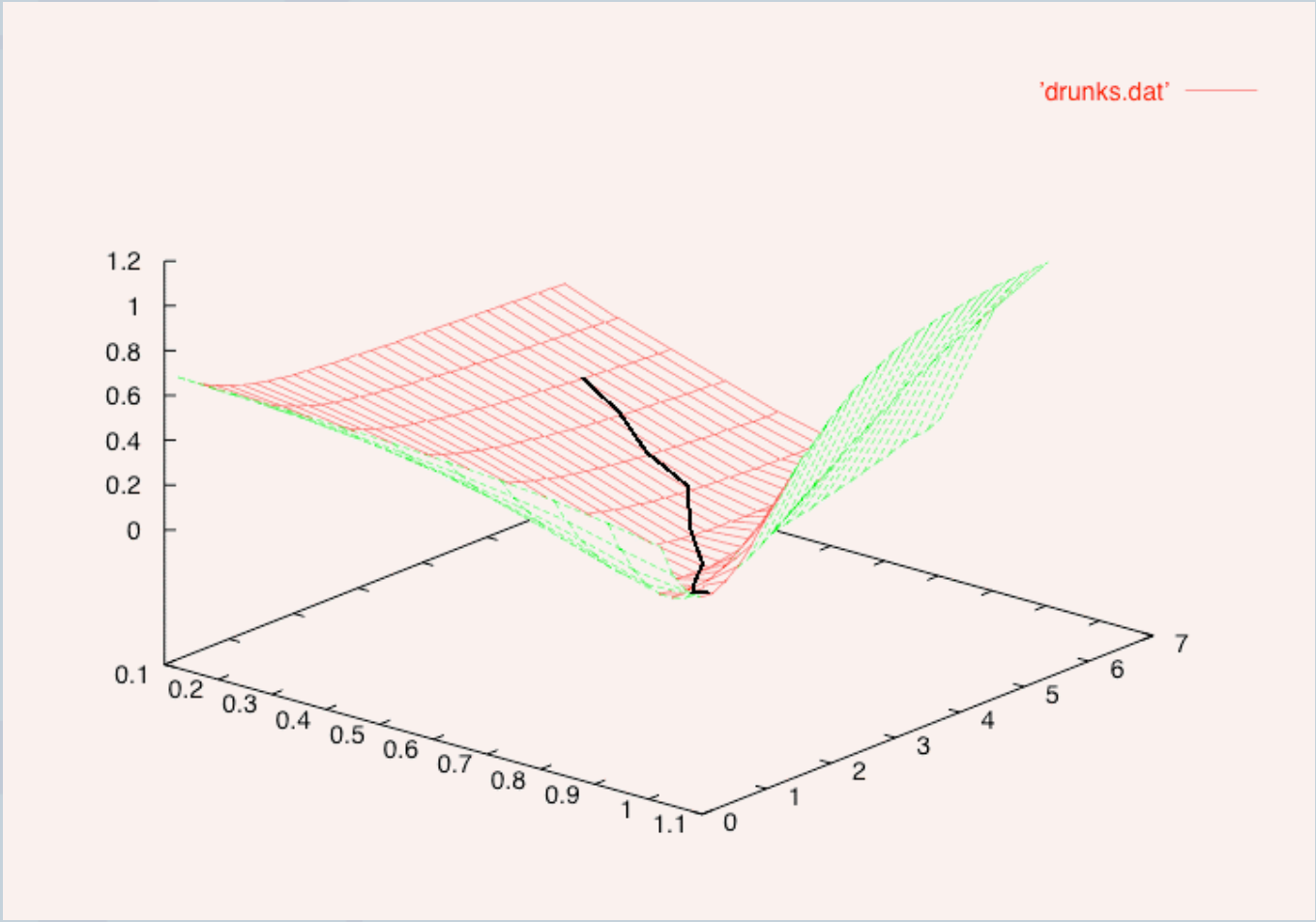
LOOP

- Select a 4th point within the simplex, or at an edge
- For each point, compute the objective function, and retain the best 3 of 4 points (step towards optimum solution)

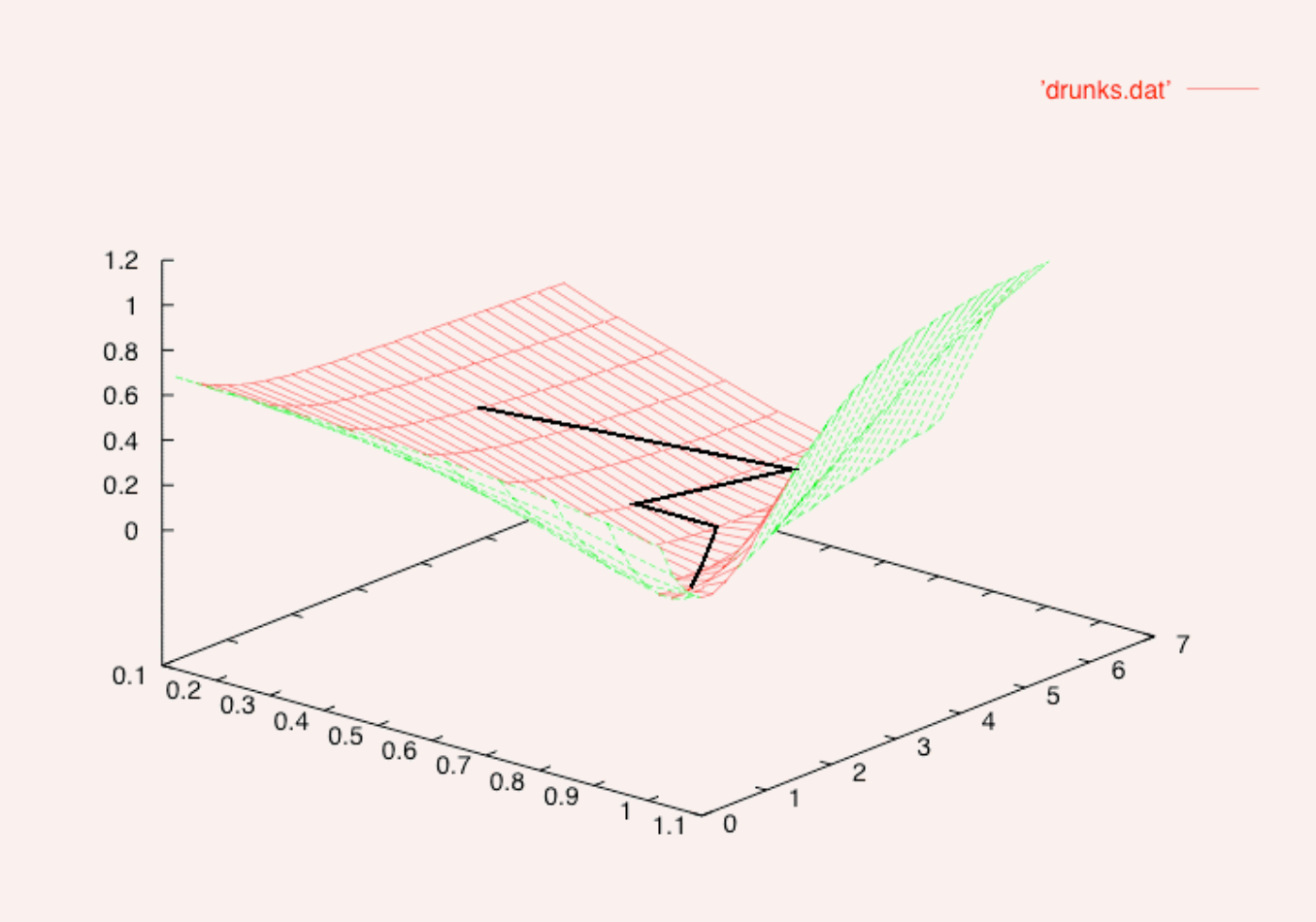
UNTIL (no further improvement)

Many optimal simplex algorithms exist…plenty of code online. It is one of the most popular parameter estimation algorithms b/c of its efficiency and reliability

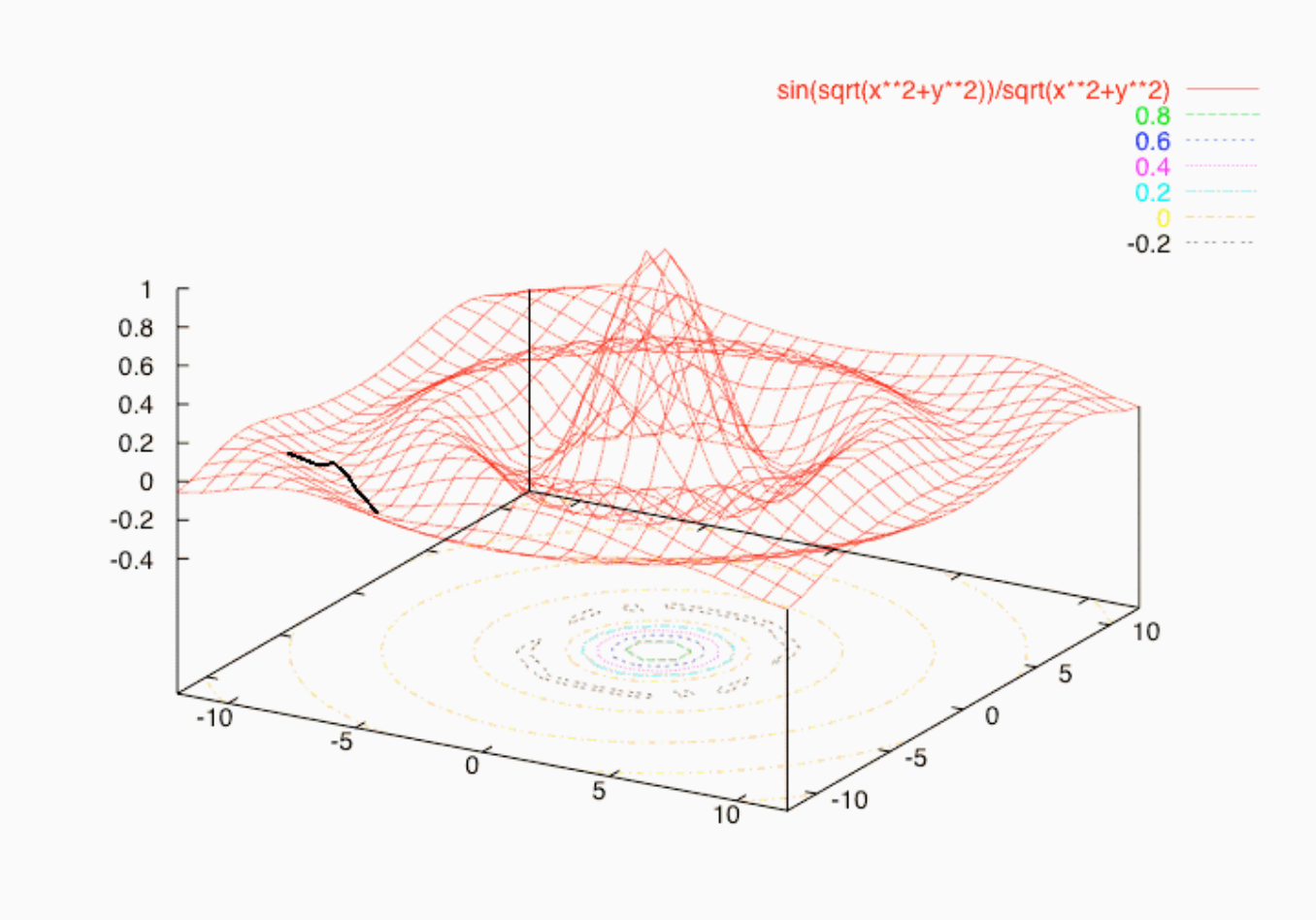
Downhill


'drunks.dat'
1.2
1
0.8
0.6
0.4
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
0
1
2
3
4
5
6
7

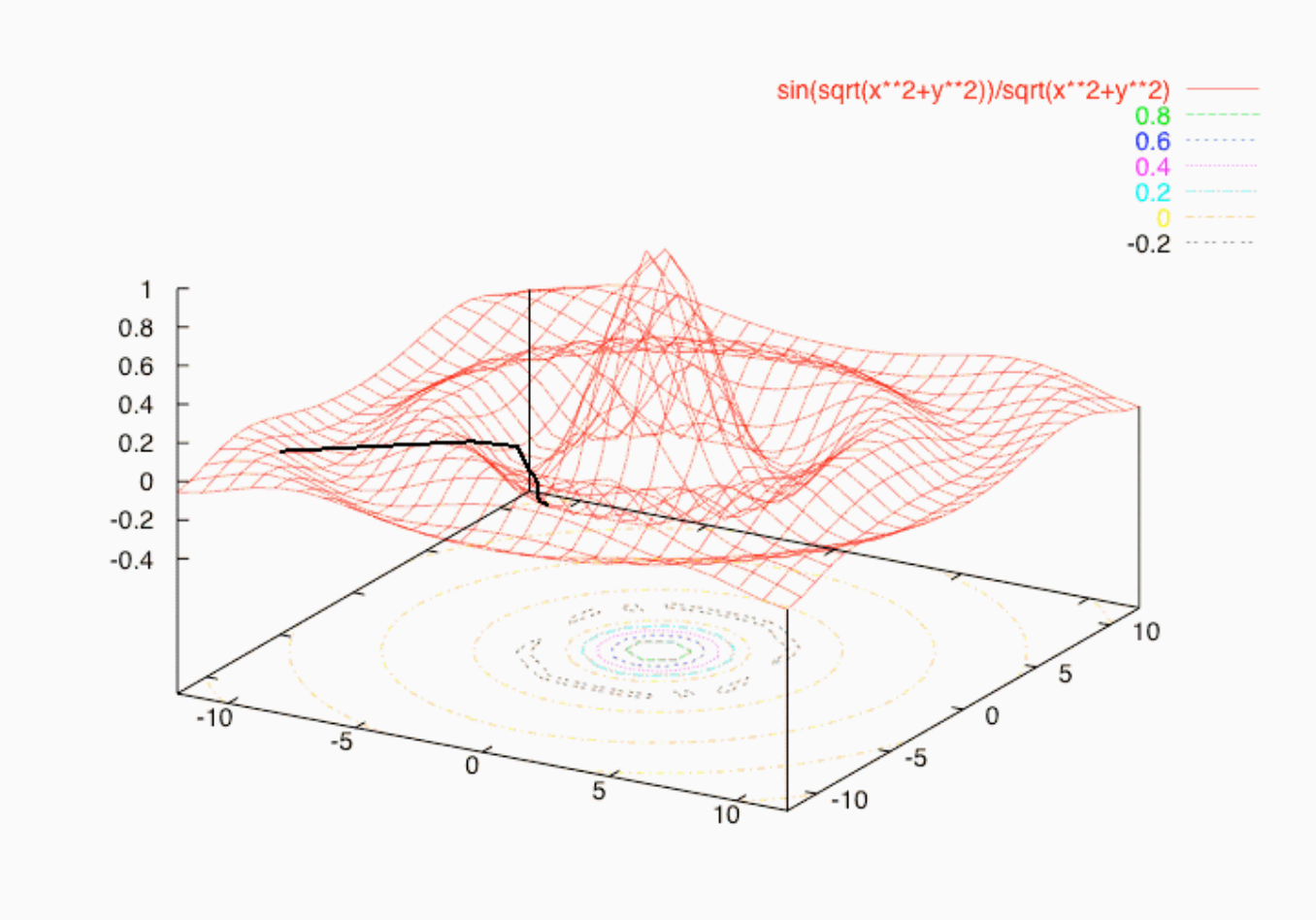
Simplex


'drunks.dat'
1.2
1
0.8
0.6
0.4
0.2
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
1.1
0
1
2
3
4
5
6
7

Downhill


sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
0.8
0.6
0.4
0.2
0
-0.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
-10
-5
0
5
10

Simplex


sin(sqrt(x**2+y**2))/sqrt(x**2+y**2)
0.8
0.6
0.4
0.2
0
-0.2
1
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-10
-5
0
5
10
-10
-5
0
5
10