Physics 403

Numerical Methods,
Maximum Likelihood, and Least Squares

Segev BenZvi

Department of Physics and Astronomy
University of Rochester

Table of Contents

Last Time

- The quadratic approximation of the PDF in more than one dimension:

$$p(\boldsymbol{x}|D, I) \propto \exp\left[(\boldsymbol{x} - \hat{\boldsymbol{x}})^\top \boldsymbol{H}(\hat{\boldsymbol{x}})(\boldsymbol{x} - \hat{\boldsymbol{x}})\right]$$

- The Hessian matrix $\boldsymbol{H}(\hat{\boldsymbol{x}})$ is an $N \times N$ symmetric matrix with components

$$H_{ij} = \left.\frac{\partial^2 L}{\partial x_i \partial x_j}\right|_{\hat{x}_i, \hat{x}_j}$$

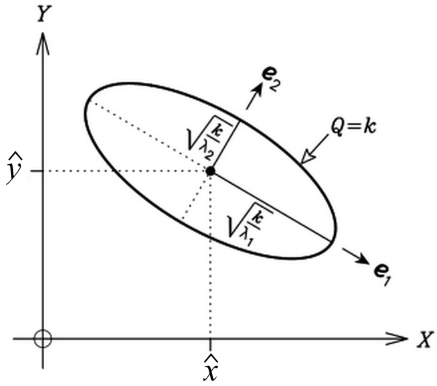
where

$$L = \ln p$$

- The covariance matrix $\boldsymbol{\Sigma}$ is related to the negative of the inverse Hessian matrix:

$$[\boldsymbol{\Sigma}]_{ij} = [-\boldsymbol{H}^{-1}]_{ij}$$

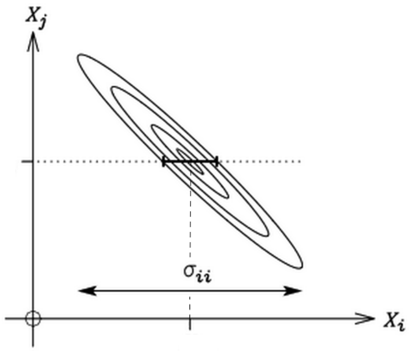
Geometric Intuition



- The uncertainty contours (in 2D) define an ellipse whose principal axes are the eigenvectors of $\boldsymbol{H}$
- When the covariance $\mathrm{cov}\,(x_i, x_j) = 0$, the ellipse is aligned with x_i and x_j
- When the covariance is nonzero the ellipse is tilted. In this case, a rotation can remove the covariances; there exists an orthogonal matrix of the eigenvectors of $\boldsymbol{H}$ which diagonalizes the Hessian:

$$\boldsymbol{D} = \boldsymbol{O}^\top \boldsymbol{H}\boldsymbol{O},$$
$$\boldsymbol{O} = \begin{pmatrix} \boldsymbol{e}_1 & \boldsymbol{e}_2 & \dots & \boldsymbol{e}_N \end{pmatrix}$$

Geometric Intuition



- When working from a joint distribution $p(x, y, \ldots | D, I)$, the uncertainty on the estimator x (for example) requires you to calculate

$$p(x|D,I) = \int p(x,y,z,\ldots|D,I)\; dy\; dz\; \ldots$$

- Remember that this is different from calculating the width of the joint distribution at the maximum
- The width of the contour at the maximum will underestimate the width of $p(x|D,I)$

Estimating μ if μ and σ are Unknown

Student-t Distribution

- If we have Gaussian data with unknown μ and σ, the resulting marginal distribution for μ is

$$p(\mu|D,I) \propto \left[\sum_{i=1}^{N}(x_i-\mu)^2\right]^{-(N-1)/2}$$

  if we use a uniform prior for σ. If we use a Jeffreys prior,

$$p(\mu|D,I) \propto \left[\sum_{i=1}^{N}(x_i-\mu)^2\right]^{-N/2}$$

- The width estimator is the usual sample variance

$$s^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\hat{\mu})^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i-\bar{x})^2$$

  for the uniform prior, and narrower ($\propto 1/N$) if using Jeffreys prior

Estimating σ if μ and σ are Unknown

χ^2 Distribution

- If we have Gaussian data with unknown μ and σ, the resulting marginal distribution for σ is

$$p(\sigma|D,I) \propto \sigma^{-(N-1)} \exp\left(-\frac{V}{2\sigma^2}\right), \qquad V = \sum_{i=1}^{N}(x_i - \bar{x})^2$$

  if we use a uniform prior for σ. If we use a Jeffreys prior,

$$p(\sigma|D,I) \propto \sigma^{-N} \exp\left(-\frac{V}{2\sigma^2}\right)$$

- $\hat{\sigma}^2 = s^2$, and the reliability of the width estimator is

$$\sigma = \hat{\sigma} \pm \frac{\hat{\sigma}}{\sqrt{2(N-1)}}.$$

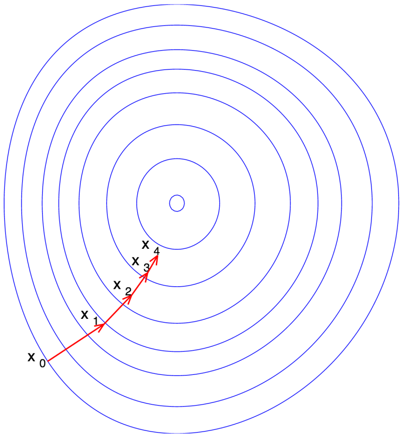
  The marginal PDF is equivalent to the $\chi^2_{2(N-1)}$ distribution.

Table of Contents

Methods for Automatic Minimization

- Getting the best estimate of a PDF means calculating its maximum. Sometimes this cannot be done analytically
- **Brute force approach**: just plot the PDF on a grid of points and visually pick out the maximum
- Unfortunately, this becomes impractical as the dimensionality of the problem grows
- Issue 1: visualizing a maximum in more than 2D is hard
- Issue 2: computational expense. For a problem with N dimensions, evaluating 10 points on each axis requires 10^N calculations
- Issue 3: a regular grid could miss narrow features in the PDF
- So we need other methods to find the maximum of a function. Most popular methods linearize the problem

Method of Steepest Descent



- How do we automatically minimize a multivariable function $f(\boldsymbol{x})$, or maximize $-f(\boldsymbol{x})$?
- Steepest Descent: given a point $\boldsymbol{a}$, $f(\boldsymbol{x})$ decreases fastest in the direction

$$-\nabla f(\boldsymbol{a})$$

- Start with a guess $\boldsymbol{x}_0$ and update:

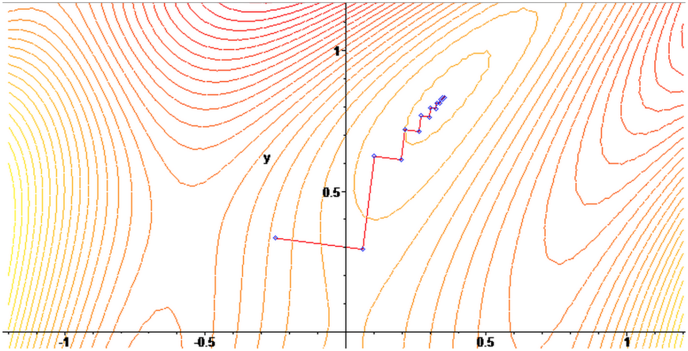
$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma_n \nabla f(\boldsymbol{x}_n), \ \ n \geq 0$$

- Control the step size with γ_n
- Keep iterating until (hopefully) $\boldsymbol{x}_n$ converges to a local minimum

Method of Steepest Descent

Known Issues

- There are several known issues with the steepest descent algorithm
- For example, if the sequence steps into a "valley" along the minimum it can start zig-zagging along the walls

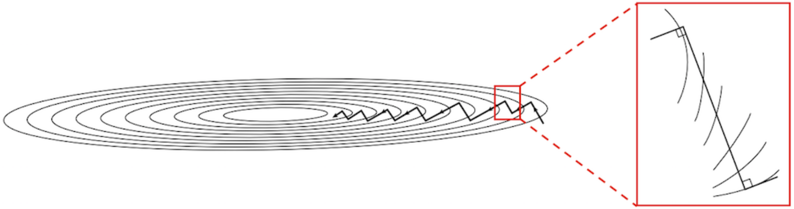


- This can make the algorithm quite slow as it approaches the minimum

Method of Steepest Descent

Behavior in the Valley

- The figure below shows why the steepest descent algorithm oscillates back and forth when you enter a valley [1]



- A step starts off in the local gradient direction perpendicular to the contour lines
- The step traverses a straight line until a local minimum is reached, where the traverse is parallel to the local contour lines
- Next update is perpendicular to the last direction. Result: S-L-O-W

Quadratic Approximation

- Suppose we Taylor-expand our function $f(\boldsymbol{x})$ about some arbitrary point $\boldsymbol{x}'$, so that

$$\begin{aligned} f(\boldsymbol{x}) &= f(\boldsymbol{x}') + (\boldsymbol{x}-\boldsymbol{x}')^\top \nabla f(\boldsymbol{x}') + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\top \nabla\nabla f(\boldsymbol{x}')(\boldsymbol{x}-\boldsymbol{x}') + \ldots \\ &\approx f(\boldsymbol{x}') + (\boldsymbol{x}-\boldsymbol{x}')^\top \nabla f(\boldsymbol{x}') + \frac{1}{2}(\boldsymbol{x}-\boldsymbol{x}')^\top \boldsymbol{H}(\boldsymbol{x}')(\boldsymbol{x}-\boldsymbol{x}') \end{aligned}$$

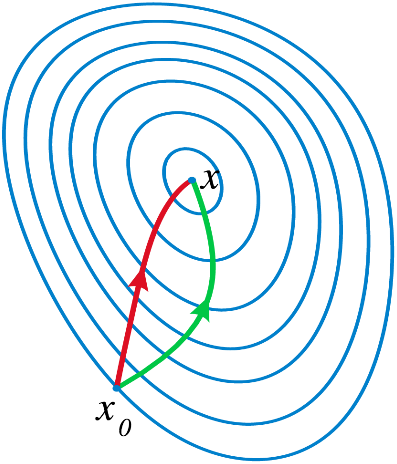
  where $\boldsymbol{H}(\boldsymbol{x}') = \nabla\nabla f(\boldsymbol{x}')$ is the Hessian matrix of f
- Differentiating f with respect to the $\{x_i\}$ gives

$$\nabla f(\boldsymbol{x}) \approx \nabla f(\boldsymbol{x}') + \boldsymbol{H}(\boldsymbol{x}')(\boldsymbol{x}-\boldsymbol{x}')$$

- If we demand $\nabla f(\hat{\boldsymbol{x}}) = 0$, since we're at an extremum, we obtain

$$\hat{\boldsymbol{x}} \approx \boldsymbol{x}' - [\boldsymbol{H}(\boldsymbol{x}')]^{-1}\nabla f(\boldsymbol{x}')$$

Newton's Method



- This expression suggests an iterative scheme for approaching a minimum:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - [\mathbf{H}(\mathbf{x}_n)]^{-1}\nabla f(\mathbf{x}_n), \quad n \geq 0$$

- Intuition: each iteration approximates $f(\mathbf{x})$ by a quadratic function and takes a step toward the minimum of the function
- If $f(\mathbf{x})$ is quadratic, the extremum will be found in exactly one step
- When the quadratic approximation is reasonable, this method will converge to the minimum much faster than the steepest descent algorithm

Newton's Method

Computational Tricks

- The stability of the iterations can be improved by reducing the step size by some positive factor $\gamma < 1$:

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n - \gamma[\boldsymbol{H}(\boldsymbol{x}_n)]^{-1}\nabla f(\boldsymbol{x}_n), \quad n \geq 0$$

- **Note**: in N dimensions, inverting $\boldsymbol{H}$ takes $\mathcal{O}(N^3)$ operations
- Instead of inverting, calculate the vector $\boldsymbol{p}_n = [\boldsymbol{H}(\boldsymbol{x}_n)]^{-1}\nabla f(\boldsymbol{x}_n)$ as the solution to the system of linear equations

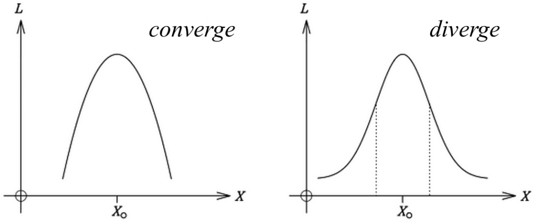
$$\boldsymbol{H}(\boldsymbol{x_n}) \cdot \boldsymbol{p}_n = \nabla f(\boldsymbol{x}_n)$$

- Methods to solve this equation, like the conjugate gradient (CG) technique [1], require $\boldsymbol{u}^\top \boldsymbol{H}(\boldsymbol{x}_n)\boldsymbol{u} > 0$ for any real nonzero vector $\boldsymbol{u}$.
- Jargon: the Hessian must be positive definite. This is a useful diagnostic, e.g., it tells you if the iteration converged to a saddle point

Newton's Method

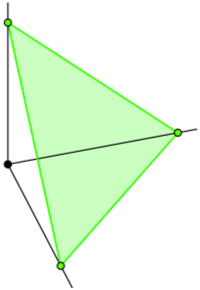
Known Issues

- Because $\nabla f(\hat{\boldsymbol{x}}) = 0$ is just the condition for a stationary point, Newton's method can diverge if $\boldsymbol{x}_0$ is far from the optimal solution



- In the figure (left) we want to find the maximum of the PDF. It's roughly quadratic so Newton's method converges rapidly
- On the right, if we start out in the tails of the function (outside the dotted lines) the algorithm will not converge to the maximum
- **Solution**: start with a good first guess. Can use an algorithm that doesn't depend on the gradient, like simplex minimization

Definition of a Simplex



- A simplex is basically a "hyper-triangle" in n dimensions.
- E.g., the n-simplex Δ^n is the subset of $\mathbb{R}^{n+1}$ such that

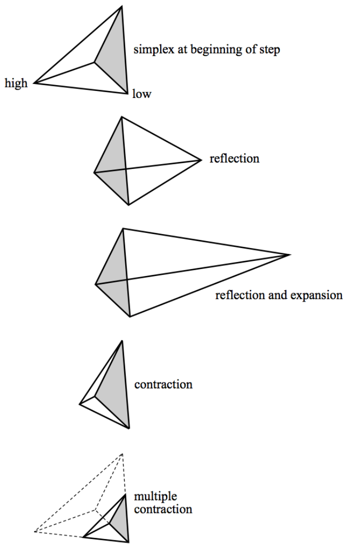
$$\Delta^n = \{(t_0, \cdots, t_n) \in \mathbb{R}^{n+1} \mid \sum_{i=0}^{n} t_i = 1 \text{ and } t_i \geq 0 \text{ for all } i\}$$

- Simplex/Nelder-Mead Technique [2]: start with $N+1$ points $\boldsymbol{p}_0$ and $\boldsymbol{p}_i$ $(i = 1 \ldots N)$ such that

$$\boldsymbol{p}_i = \boldsymbol{p}_0 + \lambda \boldsymbol{e}_i$$

- The points define a simplex for your N-dimensional parameter space. Try to move the simplex around and shrink/expand it until it contains the optimal point

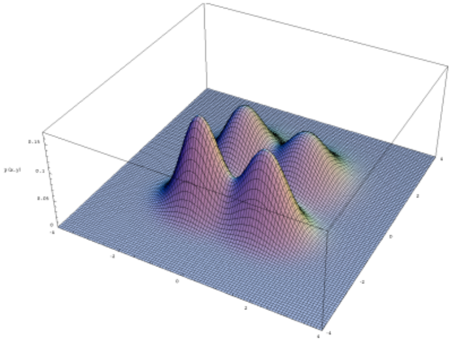
Downhill Simplex (Nelder-Mead) Algorithm



- Define the starting point for the simplex
- Pick out the point in the simplex where $f(\boldsymbol{x})$ is largest
- Reflect this point through the opposite face of the simplex to a lower point
- Shrink or expand the simplex to conserve its volume
- The simplex will crawl, amoeba-like, toward the minimum
- **Advantage**: no need to calculate the gradient. Use result as a starting point for Newton's method
- **Disadvantage**: convergence issues if initial simplex is too small

Difficult Problem: Multimodal Parameter Space

- Often you'll find that your parameter space is complex, with multiple minima and maxima



- The algorithms we have discussed so far will run as quickly as possible to the nearest minimum
- There is no way for you to guarantee that you have gotten to the global minimum rather than a local minimum

Simulated Annealing

- Starting from $\boldsymbol{x}_n$, randomly generate a new point

$$\boldsymbol{x}_{n+1} = \boldsymbol{x}_n + \Delta\boldsymbol{x}$$

- Calculate a probability

$$p = \exp\left\{-\frac{f(\boldsymbol{x}_{n+1}) - f(\boldsymbol{x}_n)}{kT}\right\}$$

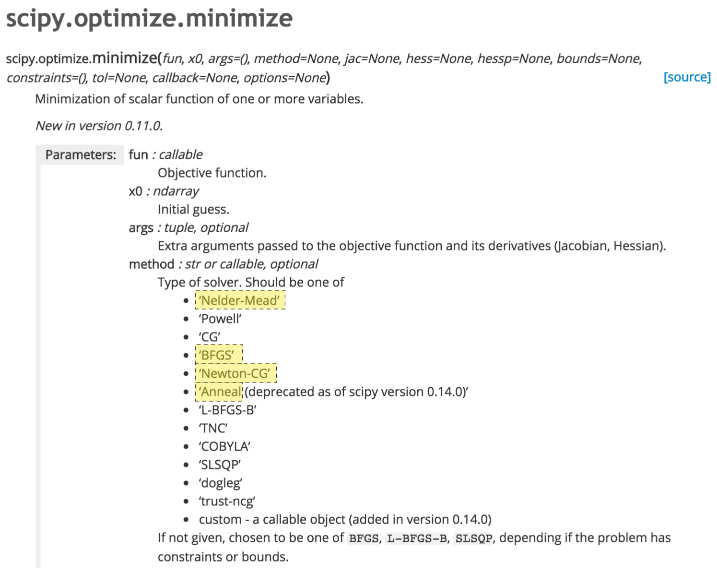
  for keeping the point, and generate a random number $u \in [0, 1]$. If $u < p$, move to $\boldsymbol{x}_{n+1}$. Otherwise, stay at $\boldsymbol{x}_n$.
- For large T, the probability of accepting new points (even "bad" moves) is high. For small T, the probability to accept new points is low
- Idea: start with a high T to help you jump out of local minima, then slowly reduce the temperature. Slow cooling helps you find the global minimum energy state, like annealing a piece of metal [3]

Markov Chain Monte Carlo

- The technique of choosing p to sample states of a thermodynamic system is called the Metropolis-Hastings algorithm [4]
- Simulated annealing depends on an annealing schedule for moving $T \rightarrow 0$, which you have to tune. Also, there is no guarantee of convergence to the global minimum in a finite time
- Another approach: run a large number of simulations at different temperatures, letting each one randomly walk through the parameter space
- This technique is called Markov Chain Monte Carlo (MCMC), and can be used to simulate exploration of all important parts of a parameter space
- MCMC methods have become central to Bayesian analysis. We'll talk about how and why in a couple of weeks

Popular Libraries

`scipy.optimize`



scipy.optimize.minimize

scipy.optimize.**minimize**(*fun, x0, args=(), method=None, jac=None, hess=None, hessp=None, bounds=None, constraints=(), tol=None, callback=None, options=None*) [source]

Minimization of scalar function of one or more variables.

New in version 0.11.0.

Parameters: fun : *callable*

Objective function.

x0 : *ndarray*

Initial guess.

args : *tuple, optional*

Extra arguments passed to the objective function and its derivatives (Jacobian, Hessian).

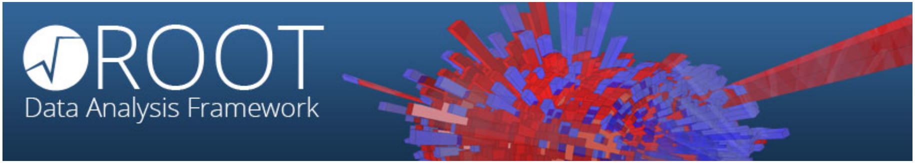
method : *str or callable, optional*

Type of solver. Should be one of

- 'Nelder-Mead'
- 'Powell'
- 'CG'
- 'BFGS'
- 'Newton-CG'
- 'Anneal (deprecated as of scipy version 0.14.0)'
- 'L-BFGS-B'
- 'TNC'
- 'COBYLA'
- 'SLSQP'
- 'dogleg'
- 'trust-ncg'
- custom - a callable object (added in version 0.14.0)

If not given, chosen to be one of `BFGS`, `L-BFGS-B`, `SLSQP`, depending if the problem has constraints or bounds.

Popular Libraries
ROOT `TMinuit`



ROOT has a C++ version of the "popular" MINUIT non-linear function minimizer. Three minimization algorithms are available:

1. Steepest descent (**MIGRAD**): evaluates gradient and second derivatives (Hessian) numerically. Assumes symmetric Gaussian errors
2. **MINOS**: relaxes error assumption, allows asymmetric error bars
3. **Simplex**: does not require evaluation of derivatives

If you've ever used this before, you know it requires a lot of hand-tuning. The going gets very rough in high-D if the parameter space is bumpy

Table of Contents

Maximum Likelihood Technique

- The method of maximum likelihood is an extremely important technique used in frequentist statistics
- There is no mystery to it. Here is the connection to the Bayesian view: given parameters $\boldsymbol{x}$ and data $\boldsymbol{D}$, Bayes' Theorem tells us that

$$p(\boldsymbol{x}|\boldsymbol{D}, I) \propto p(\boldsymbol{D}|\boldsymbol{x}, I)\ p(\boldsymbol{x}|I)$$

  where we ignore the marginal evidence $p(\boldsymbol{D}|I)$
- Suppose $p(\boldsymbol{x}|I) =$ constant for all $\boldsymbol{x}$. Then

$$p(\boldsymbol{x}|\boldsymbol{D}, I) \propto p(\boldsymbol{D}|\boldsymbol{x}, I)$$

  and the best estimator $\hat{\boldsymbol{x}}$ is simply the value that maximizes the likelihood $p(\boldsymbol{D}|\boldsymbol{x}, I)$
- So the method of maximum likelihood for a frequentist is equivalent to maximizing the posterior $p(\boldsymbol{x}|\boldsymbol{D}, I)$ with uniform priors on the $\{x_i\}$.

Connection to χ^2

- Suppose our data $\boldsymbol{D}$ are identical independent measurements with Gaussian uncertainties. Then the likelihood is

$$p(D_i|\boldsymbol{x}, I) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left[-\frac{(F_i - D_i)^2}{2\sigma_i^2}\right], \quad p(\boldsymbol{D}|\boldsymbol{x}, I) = \prod_{i=1}^{N} p(D_i|\boldsymbol{x}, I),$$

  where we defined the functional relationship between $\boldsymbol{x}$ and the ideal (noiseless) data $\boldsymbol{F}$ as

$$F_i = f(\boldsymbol{x}, i)$$

- If we define χ^2 as the sum of the squares of the normalized residuals $(F_i - D_i)/\sigma_i$, then

$$\chi^2 = \sum_{i=1}^{N} \frac{(F_i - D_i)^2}{\sigma_i^2} \implies p(\boldsymbol{D}|\boldsymbol{x}, I) \propto \exp\left(-\frac{\chi^2}{2}\right)$$

Maximum Likelihood and Least Squares

- With a uniform prior on $\boldsymbol{x}$, the logarithm of the posterior PDF is

$$L = \ln p(\boldsymbol{x}|\boldsymbol{D}, I) = \text{constant} - \frac{\chi^2}{2}$$

- The maximum of the posterior (and likelihood) will occur when χ^2 is a minimum. Hence, the optimal solution $\hat{\boldsymbol{x}}$ is called the least squares estimate
- Least squares/maximum likelihood is used all the time in data analysis, but...
- **Note**: there is nothing mysterious or even fundamental about this; least squares is what Bayes' Theorem reduces to if:
  1. Your prior on your parameters is uniform
  2. The uncertainties on your data are Gaussian
- If one of these conditions isn't met, then use Bayes' Theorem to derive something else

Maximum Likelihood: Poisson Case

- Suppose that our data aren't Gaussian, but a set of Poisson counts $\boldsymbol{n}$ with expectation values $\boldsymbol{\nu}$. E.g., we are dealing with binned data in a histogram. Then the likelihood becomes

$$p(\boldsymbol{n}|\boldsymbol{\nu}, I) = \prod_{i=1}^{N} \frac{\nu_i^{n_i} e^{-\nu_i}}{n_i!}$$

- In the limit $N \to$ large, this becomes

$$p(n_i|\nu_i, I) \propto \exp\left[-\sum_{i=1}^{N} \frac{(n_i - \nu_i)^2}{2\nu_i}\right]$$

- The corresponding χ^2 statistic is given by

$$\chi^2 = \sum_{i=1}^{N} \frac{(n_i - \nu_i)^2}{\nu_i}$$

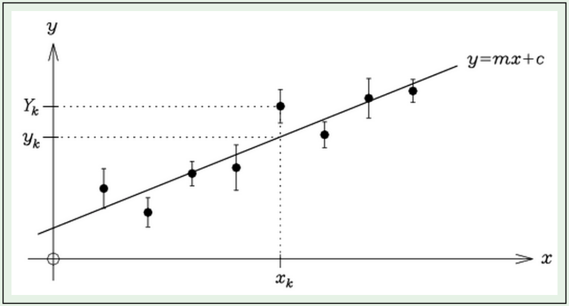
Justifications for Using Least Squares

- Nice property: as $N \to \infty$, the χ^2 statistic asymptotically approaches the value
$$\chi^2_{N-m},$$
where N is the number of data points and m is the number of parameters in $\boldsymbol{x}$.
- I.e., the statistic approximates a χ^2 distribution with $N - m$ degrees of freedom... if the uncertainties in the data are Gaussian
- Our definition of χ^2 as the quadrature sum (or l_2-norm) of the residuals makes a lot of calculations easy, but it isn't particularly robust
- **Note**: the l_1-norm
$$l_1\text{-norm} = \sum_{i=1}^{N} \left| \frac{F_i - D_i}{\sigma_i} \right|$$
is much more robust against outliers in the data

Application: Fitting a Straight Line to Data

Example

Suppose we have N measurements y_i with Gaussian uncertainties σ_i measured at positions x_i.



Given the straight line model $y_i = mx_i + b$, what are the best estimators of the parameters m and b?

Minimize the χ^2

Letting $F_i = mx_i + b$ and $D_i = y_i$, the χ^2 is

$$\chi^2 = \sum_{i=1}^{N} \frac{(mx_i + b - y_i)2}{\sigma_i^2}$$

Minimizing χ^2 as a function of the parameters gives

$$\frac{\partial \chi^2}{\partial m} = \sum_{i=1}^{N} \frac{2(mx_i + b - y_i)x_i}{\sigma_i^2} \quad \text{and} \quad \frac{\partial \chi^2}{\partial b} = \sum_{i=1}^{N} \frac{2(mx_i + b - y_i)}{\sigma_i^2}$$

Rewritten as a matrix equation, this becomes

$$\nabla\chi^2 = \begin{pmatrix} A & C \\ C & B \end{pmatrix} \begin{pmatrix} m \\ b \end{pmatrix} - \begin{pmatrix} p \\ q \end{pmatrix} = 0$$

$$A = \sum x_i^2 w_i,\ B = \sum w_i,\ C = \sum x_i w_i,\ p = \sum x_i y_i w_i,\ q = \sum y_i w_i$$

Best Estimators of a Linear Function

- Inverting the matrix, we find that

$$\hat{m} = \frac{Bp - Cq}{AB - C^2} \quad \text{and} \quad \hat{b} = \frac{Aq - Cp}{AB - C^2}$$

- The covariance matrix is found by evaluating $[2\nabla\nabla\chi^2]^{-1}$:

$$\begin{pmatrix} \sigma_m^2 & \sigma_{mb} \\ \sigma_{mb} & \sigma_b^2 \end{pmatrix} = 2 \begin{pmatrix} A & C \\ C & B \end{pmatrix}^{-1} = \frac{2}{AB - C^2} \begin{pmatrix} B & -C \\ -C & A \end{pmatrix}$$

- We note that even though the data $\{y_i\}$ are independent, the parameters $\hat{m}$ and $\hat{b}$ end up anticorrelated due to the off-diagonal terms in the covariance matrix
- This makes a lot of sense, actually; wiggling the slope of the line m clearly changes the y-intercept b

Summary

- You will often find the need to maximize a likelihood (or minimize a χ^2 or negative log likelihood) automatically
- Various algorithms available (simplex, Newton, etc.) with trade offs between speed and accuracy
- All algorithms are sensitive, to some degree or another, to the starting position of the minimization
- **Maximum likelihood**: same as maximizing a posterior PDF when the priors on the parameters are uniform
- Maximizing the likelihood is the same as minimizing χ^2 in the case where the uncertainties on the data are Gaussian
- In case of Gaussian uncertainties, there is asymptotic convergence of the maximum likelihood to the χ^2 distribution:

$$\chi^2 = -2\ln L \sim \chi^2_{N-m}$$

References I

[1] W. Press et al. *Numerical Recipes in C*. New York: Cambridge University Press, 1992. URL: http://www.nr.com.

[2] J.A. Nelder and R. Mead. “A Simplex Method for Function Minimization”. In: *Comp. J.* 7 (1965), pp. 308–313.

[3] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. “Optimization by Simulated Annealing”. In: *Science* 220.4598 (1983), pp. 671–680.

[4] N. Metropolis et al. “Equation of State Calculations by Fast Computing Machines”. In: *J. Chem. Phys.* 21 (1953), p. 1087.