18. Nonlinear least squares

Outline

Nonlinear equations and least squares

Examples

Levenberg–Marquardt algorithm

Nonlinear model fitting

Nonlinear least squares classification

Nonlinear equations

- set of m nonlinear equations in n unknowns $x_1, \ldots, x_n$:

$$f_i(x_1, \ldots, x_n) = 0, \quad i = 1, \ldots, m$$

- $f_i(x) = 0$ is the ith equation; $f_i(x)$ is the ith residual
- n-vector of unknowns $x = (x_1, \ldots, x_n)$
- write as vector equation $f(x) = 0$ where $f : \mathbf{R}^n \to \mathbf{R}^m$,

$$f(x) = (f_1(x), \ldots, f_m(x))$$

- when f is affine, reduces to set of m linear equations
- over-determined if $m > n$, under-determined if $m < n$, square if $m = n$

Nonlinear least squares

- find $\hat{x}$ that minimizes

$$\|f(x)\|^2 = f_1(x)^2 + \cdots + f_m(x)^2$$

- includes problem of solving equations $f(x) = 0$ as special case
- like (linear) least squares, super useful on its own

Optimality condition

- optimality condition: $\nabla\|f(\hat{x})\|^2 = 0$
- any optimal point satisfies this
- points can satisfy this and not be optimal
- can be expressed as $2Df(\hat{x})^T f(\hat{x}) = 0$
- $Df(\hat{x})$ is the $m \times n$ derivative or Jacobian matrix,

$$Df(\hat{x})_{ij} = \frac{\partial f_i}{\partial x_j}(\hat{x}), \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

- optimality condition reduces to normal equations when f is affine

Difficulty of solving nonlinear least squares problem

- solving nonlinear equations or nonlinear least squares problem is (in general) *much harder* than solving linear equations
- even determining if a solution exists is hard
- so we will use *heuristic* algorithms:
  - not guaranteed to always work
  - but often work well in practice

  (like k-means)

Outline

Nonlinear equations and least squares

Examples

Levenberg–Marquardt algorithm

Nonlinear model fitting

Nonlinear least squares classification

Computing equilibrium points

- *equilibrium prices:* find n-vector of prices p for which $S(p) = D(p)$
  - $S(p)$ is supply of n goods as function of prices
  - $D(p)$ is demand for n goods as function of prices
  - take $f(p) = S(p) - D(p)$
- *chemical equilibrium:* find n-vector of concentrations c so $C(c) = G(c)$
  - $C(c)$ is consumption of species as function of c
  - $G(c)$ is generation of species as function of c
  - take $f(c) = C(c) - G(c)$

Location from range measurements

- 3-vector x is position in 3-D, which we will estimate
- *range* measurements give (noisy) distance to known locations

$$\rho_i = \|x - a_i\| + v_i, \quad i = 1, \ldots, m$$

  a_i are known locations, v_i are noises
- least squares location estimation: choose $\hat{x}$ that minimizes

$$\sum_{i=1}^{m} (\|x - a_i\| - \rho_i)^2$$

- GPS works like this

Outline

Nonlinear equations and least squares

Examples

Levenberg–Marquardt algorithm

Nonlinear model fitting

Nonlinear least squares classification

The basic idea

- at any point z we can form the affine approximation

$$\hat{f}(x; z) = f(z) + Df(z)(x - z)$$

- $\hat{f}(x; z) \approx f(x)$ *provided* x is near z
- we can minimize $\|\hat{f}(x; z)\|^2$ using linear least squares
- we'll iterate, with z the current iterate

Levenberg–Marquardt algorithm

- iterates $x^{(1)}, x^{(2)}, \ldots$
- at iteration k, form affine approximation of f at $x^{(k)}$:

$$\hat{f}(x; x^{(k)}) = f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})$$

- choose $x^{(k+1)}$ as minimizer of

$$\|\hat{f}(x; x^{(k)})\|^2 + \lambda^{(k)}\|x - x^{(k)}\|^2 \qquad \text{(where } \lambda^{(k)} > 0\text{)}$$

- we want $\|\hat{f}(x; x^{(k)})\|^2$ small, but we don't want to move too far from $x^{(k)}$, where $\hat{f}(x; x^{(k)}) \approx f(x)$ no longer holds

Levenberg–Marquardt iteration

- $x^{(k+1)}$ is solution of least squares problem

$$\text{minimize} \quad \|f(x^{(k)}) + Df(x^{(k)})(x - x^{(k)})\|^2 + \lambda^{(k)}\|x - x^{(k)}\|^2$$

- solution is

$$x^{(k+1)} = x^{(k)} - \left(Df(x^{(k)})^T Df(x^{(k)}) + \lambda^{(k)} I\right)^{-1} Df(x^{(k)})^T f(x^{(k)})$$

- inverse always exists (since $\lambda^{(k)} > 0$)
- $x^{(k+1)} = x^{(k)}$ only if $Df(x^{(k)})^T f(x^{(k)}) = 0$, *i.e.*, optimality condition holds

Adjusting $\lambda^{(k)}$

idea:

- if $\lambda^{(k)}$ is too big, $x^{(k+1)}$ is too close to $x^{(k)}$, and progress is slow
- if too small, $x^{(k+1)}$ may be far from $x^{(k)}$ and affine approximation is poor

update mechanism:

- if $\|f(x^{(k+1)})\|^2 < \|f(x^{(k)})\|^2$, accept new x and reduce λ

$$\lambda^{(k+1)} = 0.8\lambda^{(k)}$$

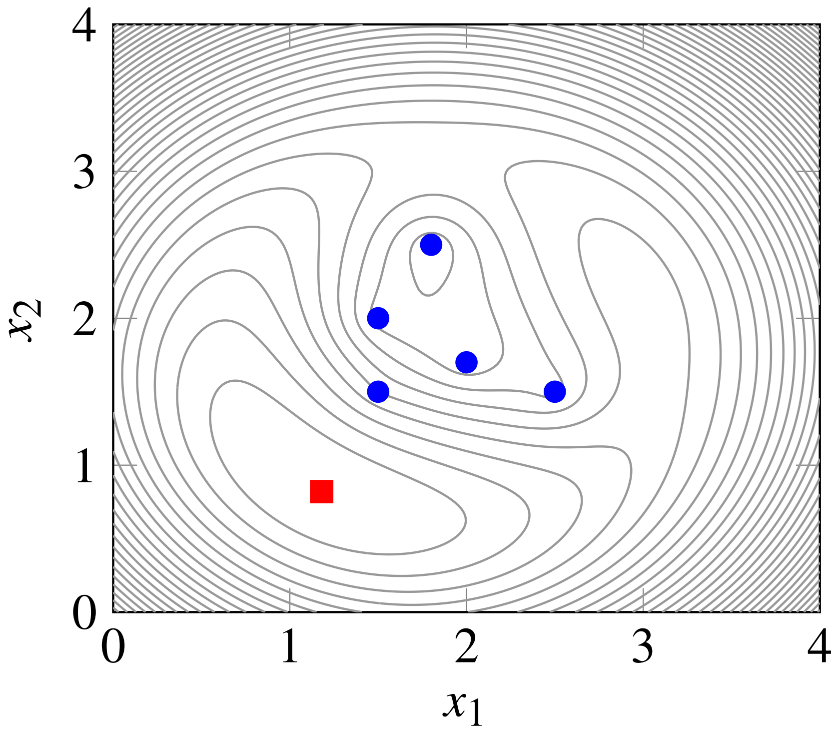
- otherwise, increase λ and do not update x:

$$\lambda^{(k+1)} = 2\lambda^{(k)}, \qquad x^{(k+1)} = x^{(k)}$$

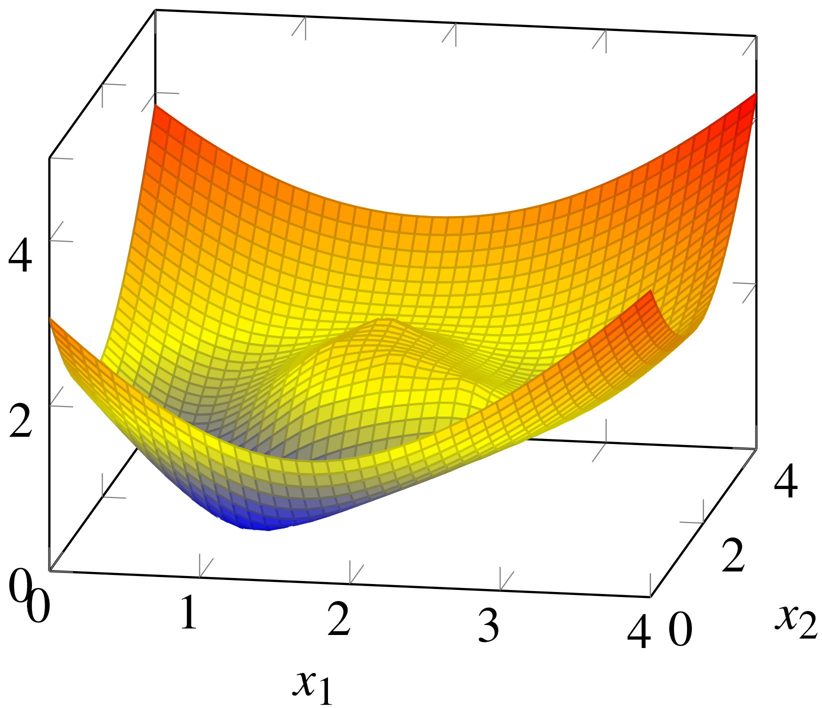
Example: Location from range measurements

- range to 5 points (blue circles)
- red square shows $\hat{x}$

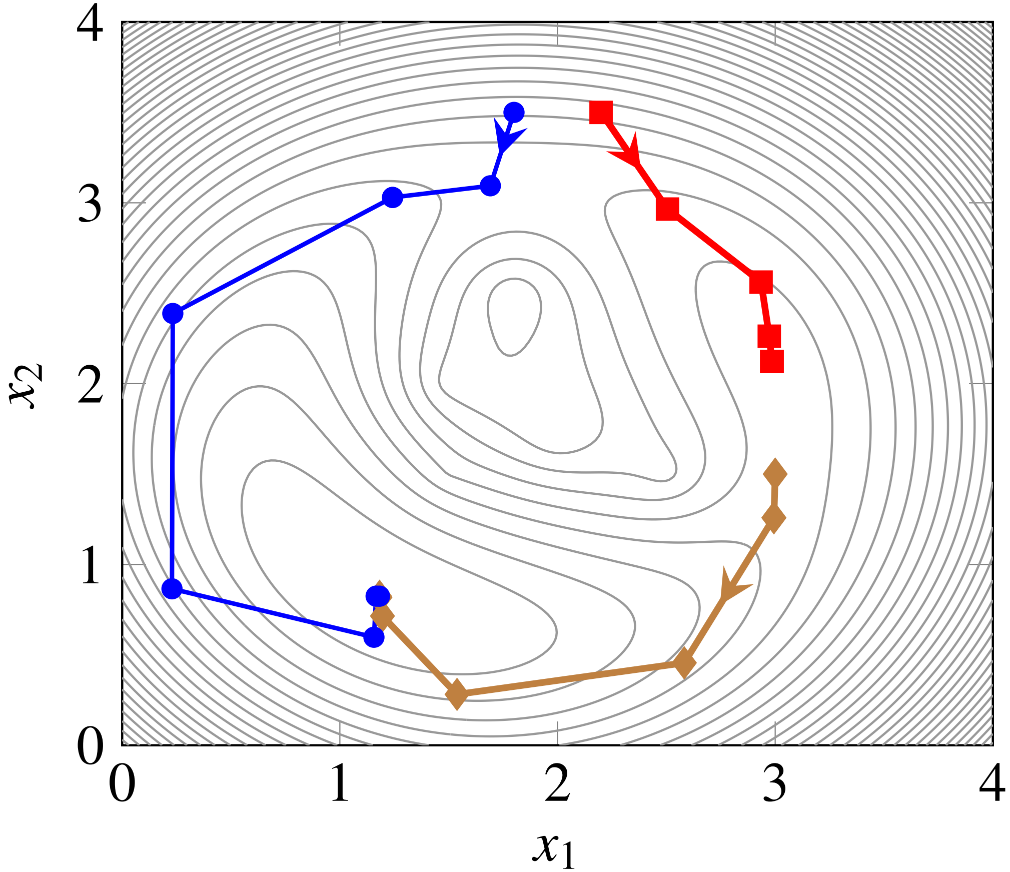
contour lines of $\|f(x)\|^2$



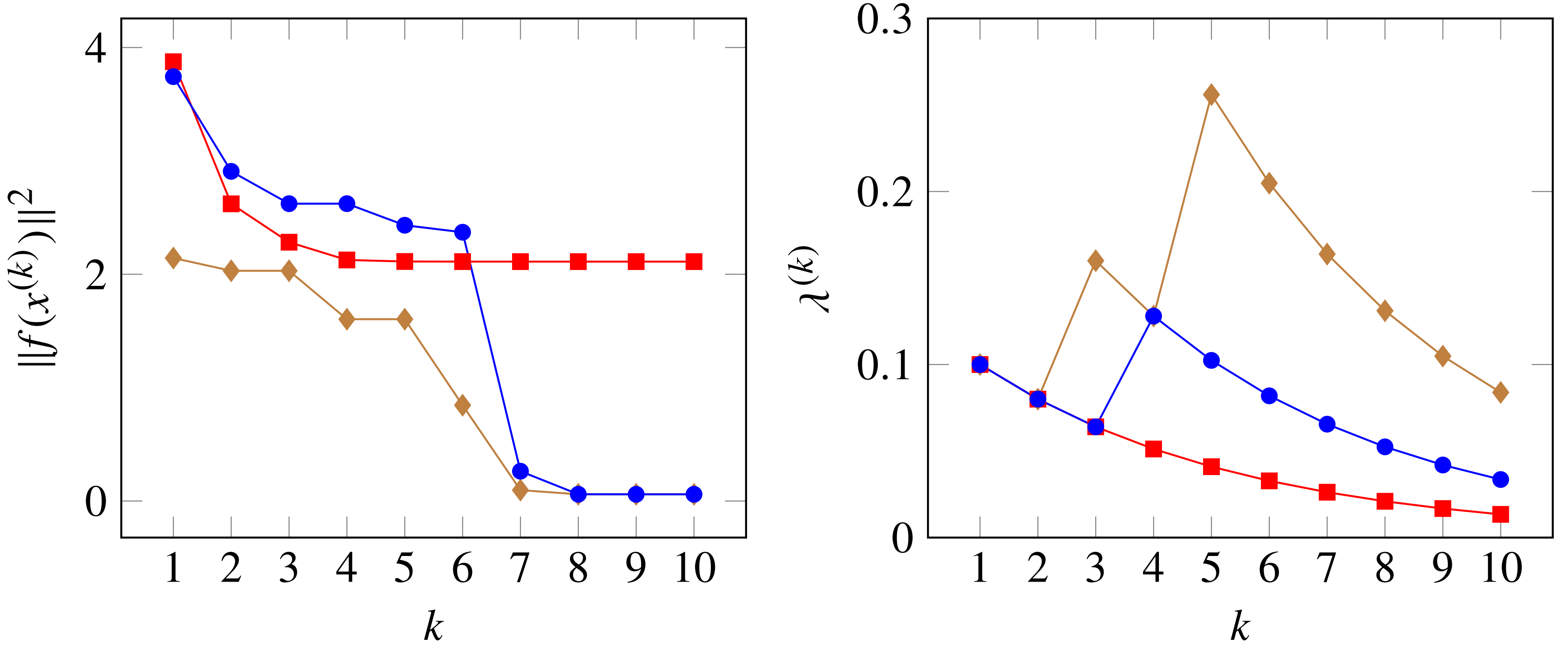
graph of $\|f(x)\|$



Levenberg–Marquardt from three initial points



Levenberg–Marquardt from three initial points



Outline

Nonlinear equations and least squares

Examples

Levenberg–Marquardt algorithm

Nonlinear model fitting

Nonlinear least squares classification

Nonlinear model fitting

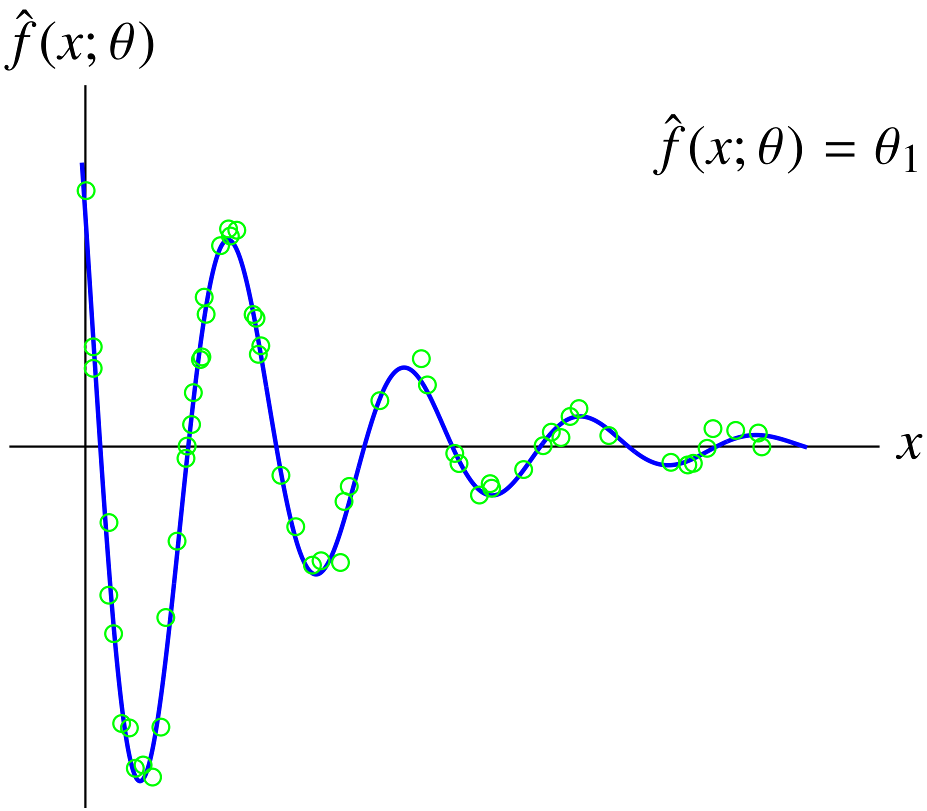
$$\text{minimize} \quad \sum_{i=1}^{N} (\hat{f}(x^{(i)};\theta) - y^{(i)})^2$$

- $x^{(1)},\ldots,x^{(N)}$ are feature vectors
- $y^{(1)},\ldots,y^{(N)}$ are associated outcomes
- model $\hat{f}(x;\theta)$ is parameterized by parameters $\theta_1,\ldots,\theta_p$
- this generalizes the *linear in parameters model*

$$\hat{f}(x;\theta) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$$

- here we allow $\hat{f}(x,\theta)$ to be a nonlinear function of θ
- the minimization is over the model parameters θ

Example



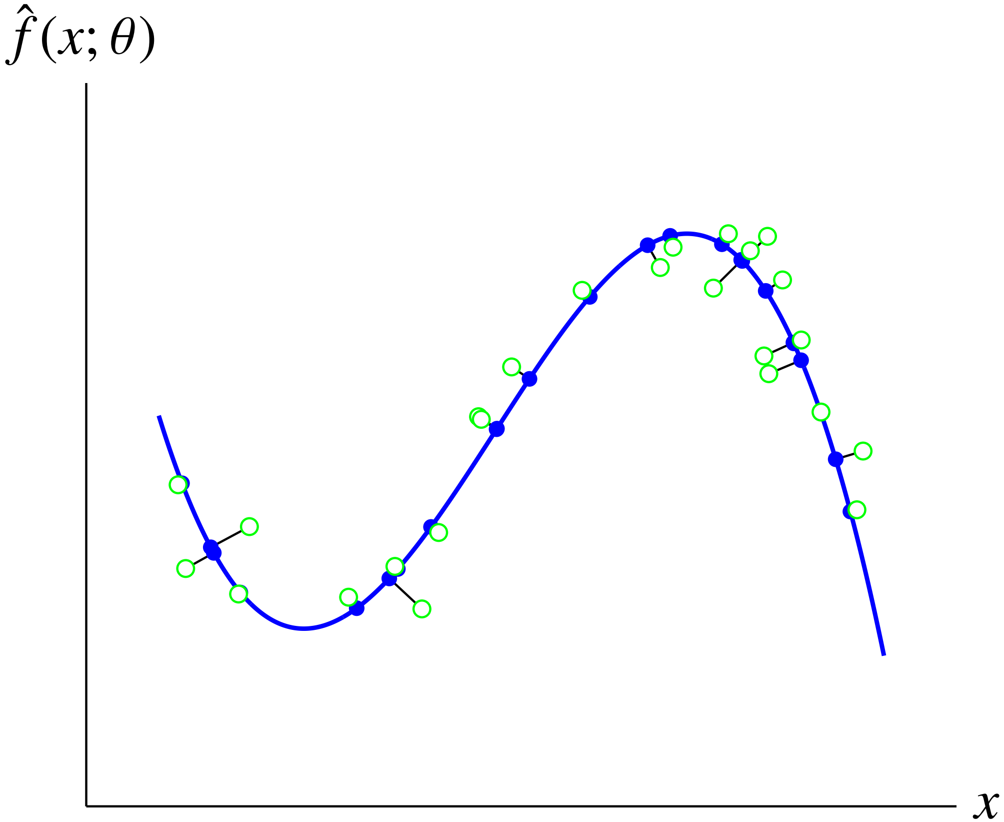
$$\hat{f}(x;\theta) = \theta_1 \exp(\theta_2 x) \cos(\theta_3 x + \theta_4)$$

a nonlinear least squares problem with four variables θ_1, θ_2, θ_3, θ_4:

$$\text{minimize} \quad \sum_{i=1}^{N} \left(\theta_1 e^{\theta_2 x^{(i)}} \cos(\theta_3 x^{(i)} + \theta_4) - y^{(i)}\right)^2$$

Orthogonal distance regression

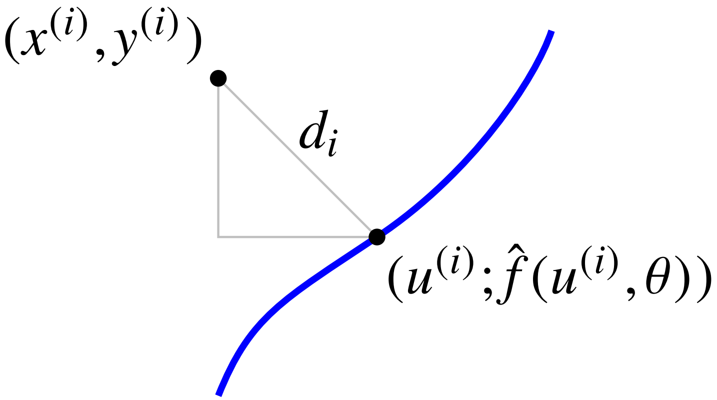
- to fit model, minimize sum square distance of data points to graph
- example: orthogonal distance regression to cubic polynomial



Nonlinear least squares formulation

$$\text{minimize} \quad \sum_{i=1}^{N} \left((\hat{f}(u^{(i)};\theta) - y^{(i)})^2 + \|u^{(i)} - x^{(i)}\|^2 \right)$$

- optimization variables are model parameters θ and N points $u^{(i)}$
- ith term is squared distance of data point $(x^{(i)}, y^{(i)})$ to point $(u^{(i)}, \hat{f}(u^{(i)}, \theta))$



$$d_i^2 = (\hat{f}(u^{(i)};\theta) - y^{(i)})^2 + \|u^{(i)} - x^{(i)}\|^2$$

- minimizing over $u^{(i)}$ gives squared distance of $(x^{(i)}, y^{(i)})$ to graph
- minimizing over $u^{(1)}, \ldots, u^{(N)}$ *and* θ minimizes the sum square distance

Outline

Nonlinear equations and least squares

Examples

Levenberg–Marquardt algorithm

Nonlinear model fitting

Nonlinear least squares classification

Nonlinear least squares classification

linear least squares classifier:

- classifier is $\hat{f}(x) = \mathbf{sign}\,(\tilde{f}(x))$ where $\tilde{f}(x) = \theta_1 f_1(x) + \cdots + \theta_p f_p(x)$
- θ is chosen by minimizing $\sum_{i=1}^{N}(\tilde{f}(x_i) - y_i)^2$ (plus optionally regularization)

nonlinear least squares classifier:

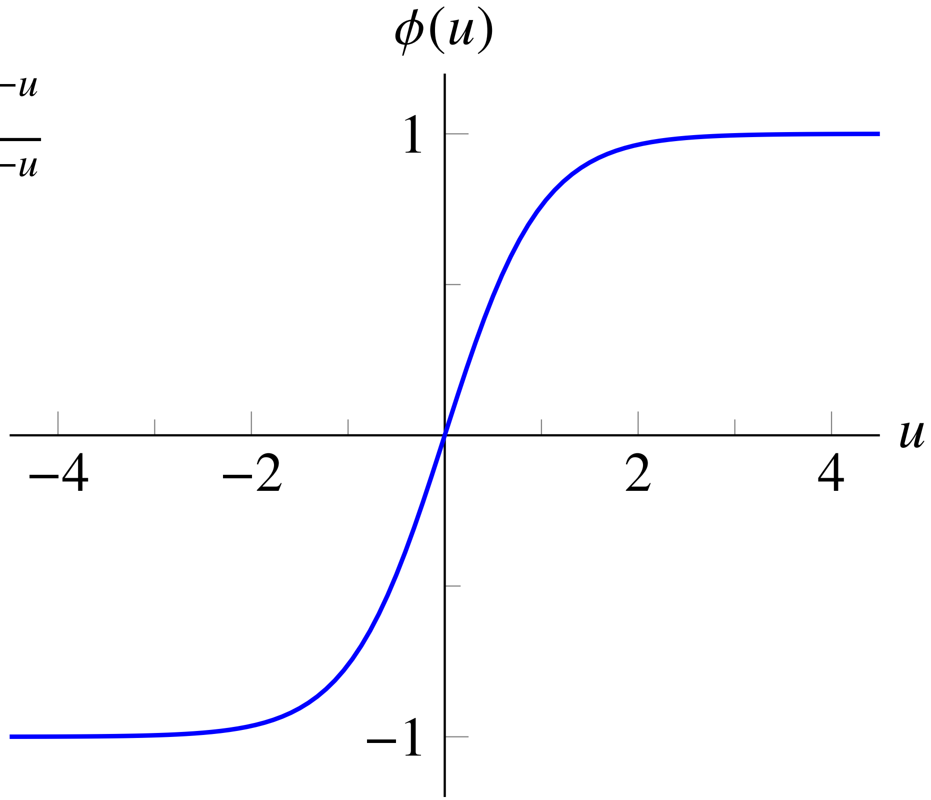
- choose θ to minimize

$$\sum_{i=1}^{N}(\mathbf{sign}(\tilde{f}(x_i)) - y_i)^2 = 4 \times \text{number of errors}$$

- replace **sign** function with smooth approximation ϕ, *e.g.*, sigmoid function
- use Levenberg–Marquardt to minimize $\sum_{i=1}^{N}(\phi(\tilde{f}(x_i)) - y_i)^2$

Sigmoid function

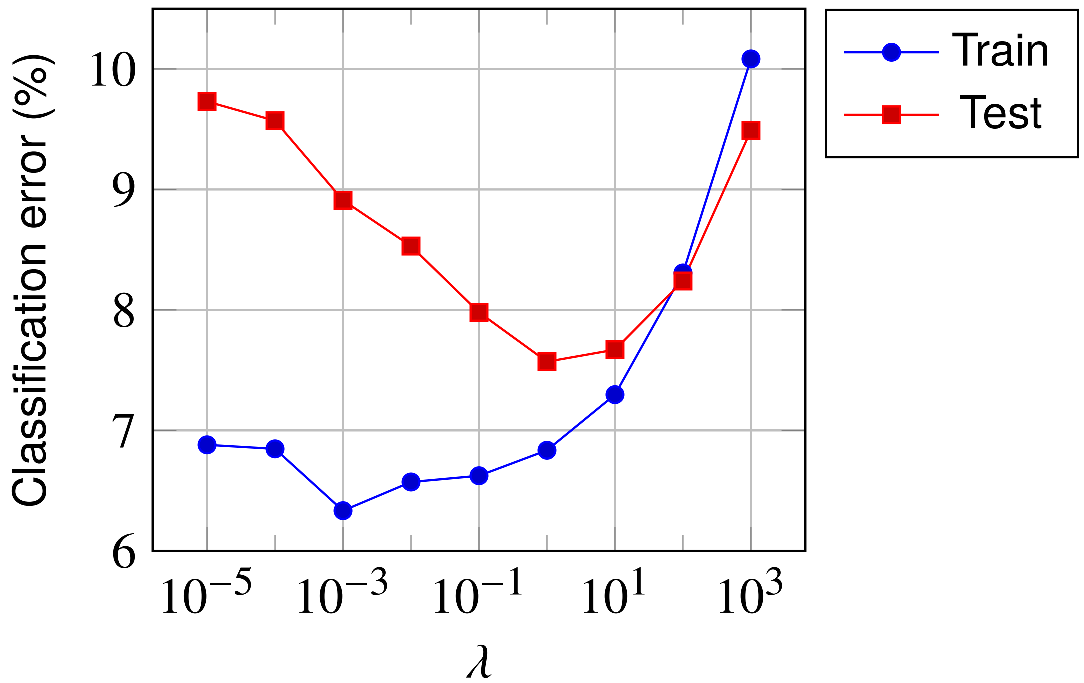
$$\phi(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$$



Example

- MNIST data set; feature vector x is 493-vector of pixel intensities
- nonlinear least squares 10-way multi-class classifier: 7.5% test error
- Boolean classifiers computed by solving nonlinear least squares problems

$$\text{minimize} \quad \sum_{i=1}^{N} (\phi((x^{(i)})^T \beta + v) - y^{(i)})^2 + \lambda \|\beta\|^2$$



Feature engineering

- add 5000 random features as before
- test set error drops to 2%
- this matches human performance
- with more feature engineering, can substantially beat human performance