Levenberg-Marquardt
(V. Rabaud)

Levenberg-Marquardt

- General Math
- General problem and obvious solutions
  - Gradient descent
  - Gauss-Newton
- Levenberg-Marquardt
- Limitations
- Applications

General math (1/5)

- Jacobian matrix: $f:\begin{cases} \mathbb{R}^n \rightarrow \mathbb{R}^n \\ x \rightarrow \begin{bmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{bmatrix} \end{cases}$

$$J(x)=\begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$$

- Hessian: Jacobian of the derivative

$$Hf(x)=\begin{bmatrix} \frac{\partial^2 f_1}{\partial x_1^2} & \cdots & \frac{\partial^2 f_1}{\partial x_1 \partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f_n}{\partial x_1 \partial x_n} & \cdots & \frac{\partial^2 f_n}{\partial x_n^2} \end{bmatrix}$$

$f:\mathbb{R}^n \rightarrow \mathbb{R}$

General math (2/5)

Hessian is like the Fisher information matrix

$X(x)$ a random variable, $f_X(x)$ a probability distribution on X:

$$F = E(\nabla_\theta \ln(p_{r/\theta}(r)) \nabla_\theta \ln(p_{r/\theta}(r))^T)$$

$$F = -E(H(\ln(p_{r/\theta}(r))))$$

General math (3/5)

- Quadratic form: $Q(x) = x^T A x$

  Q is symmetric and $Q: \mathbb{R}^n \rightarrow \mathbb{R}$
- Q is said to be positive definite iff $Q(x) > 0 \; \forall \; x \neq 0$

  iff all the eigen values are >0
- Q is said to be positive semi-definite iff $Q(x) \geq 0$

  then $Q = LDL^T$

General math (4/5)

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

If $\nabla f(x) = \vec{0}$

- if $H(x)$ is positive definite, x is a strict local minimum
- if $H(x)$ is negative definite, x is a strict local maximum
- if $H(x)$ is indefinite, x is a non-degenerate saddle point

General math (5/5)

$$f : \mathbb{R}^n \rightarrow \mathbb{R}$$

$$Q = LDL^T$$

Diagonal elements of H are related to the curvature

$$\kappa = \frac{\dfrac{\partial^2 y}{\partial x^2}}{\left(1 + \left(\dfrac{\partial y}{\partial x}\right)^2\right)^{3/2}}$$

General Problem (1/4)

- Non-linear least squares minimization:

  solve for the minimum of the differentiable function:

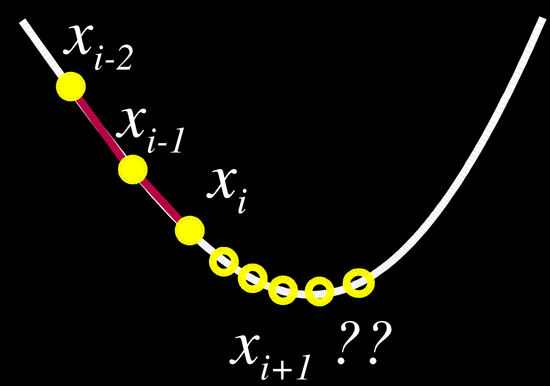
$$f : \begin{array}{l} \mathbb{R}^n \to \mathbb{R} \\ x \to \frac{1}{2}\sum_{j=1}^{m} r_j(x)^2 \end{array}$$

- Rewritten as: $f(x)=\frac{1}{2}\|r(x)\|^2$ with

$$r(x)=(r_1(x), r_2(x), \ldots, r_m(x))^T$$

General Problem (2/4)

<u>General method:</u>



$$x_{i+1} = x_i + \lambda_i \cdot d_i$$

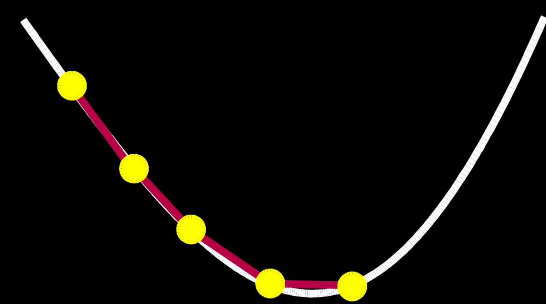
λ: intensity of the displacement

d: direction of the displacement

Values: constant, depend on ∇f, $\nabla^2 f$, depend on λ_k or d_k

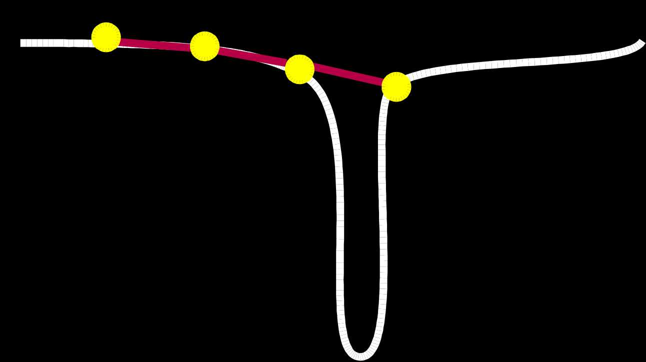
General Problem (3/4)

- Gradient descent:



$$x_{i+1} = x_i - \lambda \cdot \nabla f$$

- Problem:



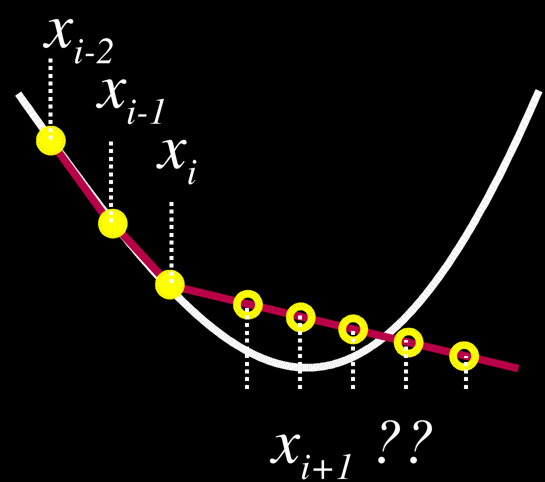
General Problem (4/4)

$$\boxed{x_{i+1} = x_i + \lambda_i \cdot d_i}$$

- <u>Line search:</u>

  d_i fixed and

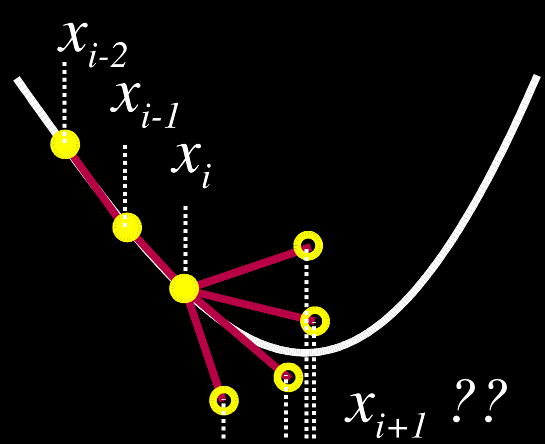
  $$\lambda_i = argmin_\lambda \| f(x_i + \lambda \cdot d_i) \|$$



- <u>Trust-region search:</u>

  λ_i fixed and

  $$d_i = argmin_d \| f(x_i + \lambda_i \cdot d) \|$$

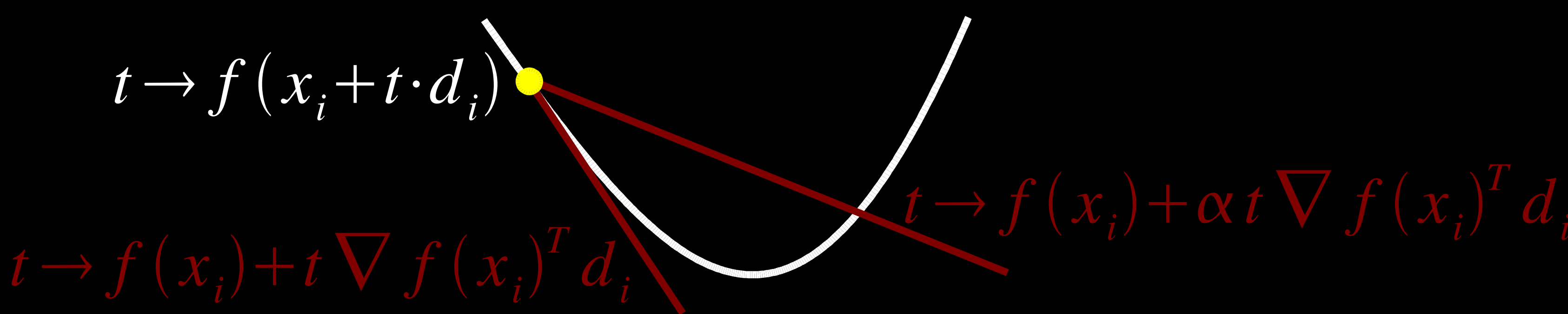


Gradient Descent (1/6)

$$\boxed{x_{i+1} = x_i + \lambda \cdot d_i}$$

- <u>Exact line search:</u> $\lambda_i = argmin_{\lambda} \| f(x_i + \lambda \cdot d_i) \|$
- <u>Backtracking line search:</u> $\alpha \in [0, 0.5], \beta \in [0, 1], t = 1$

While $f(x_i + t \cdot d_i) > f(x_i) + \alpha t \nabla f(x_i)^T d_i, t \leftarrow \beta t$



(BOARD)

Gradient Descent (2/6)

$$\boxed{x_{i+1} = x_i + \lambda \cdot d_i}$$

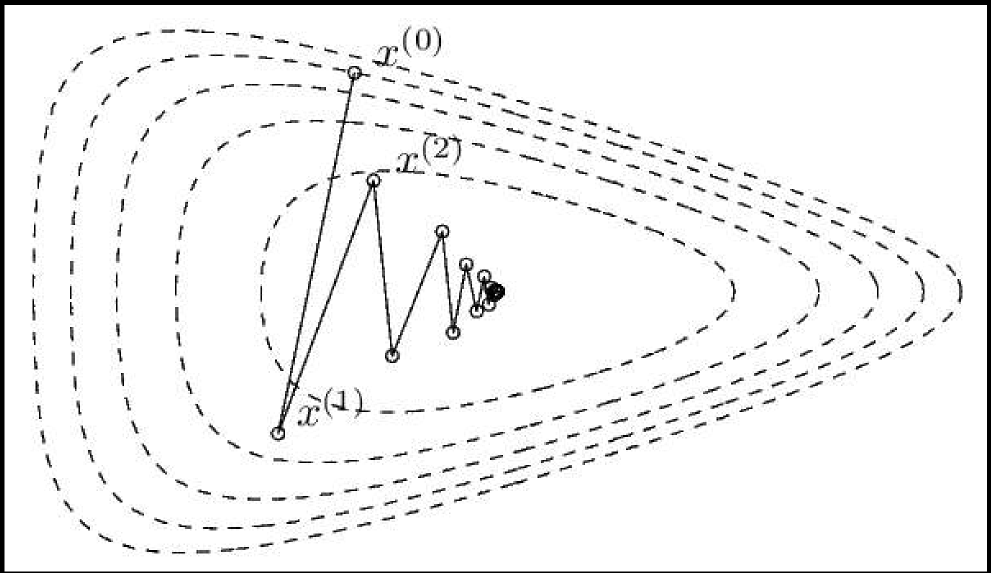
- Gradient descent: $x_{i+1} = x_i - \lambda \cdot \nabla f(x_i)$

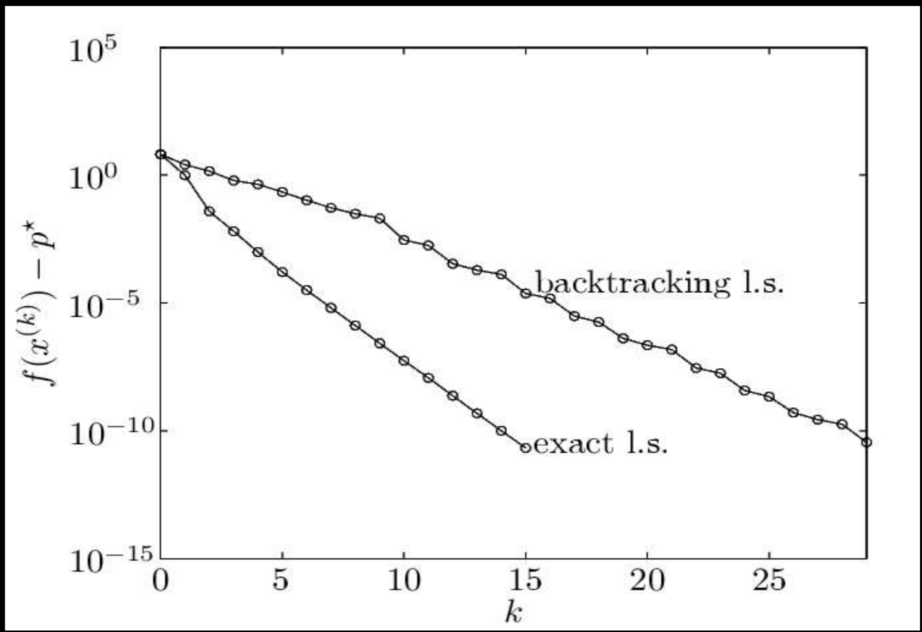
- Complexity: $\|f(x_i) - p'\| < \epsilon$ in $\dfrac{\log(\|f(x_i) - p'\|/\epsilon)}{\log(1/c)}$

  where $mI \leq H \leq MI$ and $c = 1 - \dfrac{m}{M}$

Gradient Descent (3/6)

Example: $$f(x_1, x_2) = e^{x_1 + 3x_2 - 0.1} + e^{x_1 - 3x_2 - 0.1} + e^{-x_1 - 0.1}$$





Gradient Descent (4/6)

Conclusion on the gradient descent:

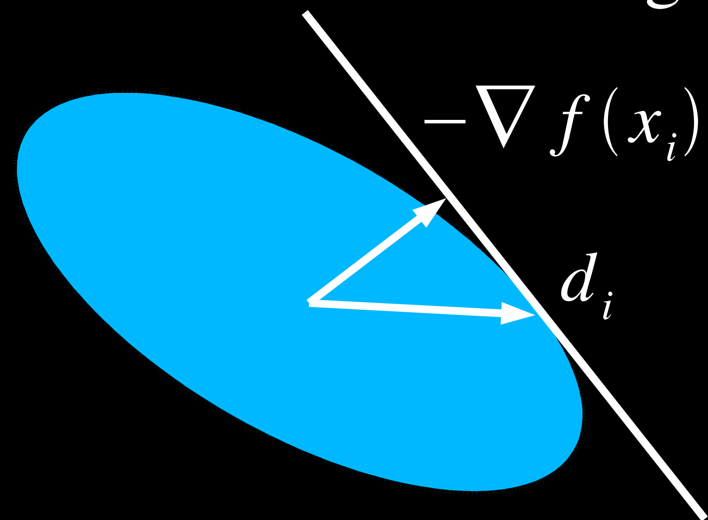
- linear convergence
- backtracking parameters α and β have a slight influence.

Gradient Descent (5/6)

$$\boxed{x_{i+1} = x_i + \lambda \cdot d_i}$$

- <u>Taylor series:</u> $f(x_i + d) \approx f(x) + \nabla f^T(x_i) d$
- <u>Steepest descent:</u> $d_i = argmin_d \{\nabla f^T(x_i) \cdot d / \|d\| = 1\}$

  Then, line search or backtracking line search



Gradient Descent (6/6)

$$\boxed{x_{i+1} = x_i + \lambda \cdot d_i}$$

$$d_i = argmin_d \{ \nabla f^T(x_i) \cdot d \,/\, \|d\| = 1 \}$$

Importance of the norm in Steepest descent:

- Euclidian: gradient descent
- l_1,l_2,quadratic

Gauss-Newton (1/6)

$$\boxed{x_{i+1} = x_i + \lambda_i \cdot d_i}$$

- <u>Gradient descent:</u> $x_{i+1} = x_i + \lambda \cdot \nabla f(x_i)$

- <u>Gauss-Newton:</u>

  Taylor series: $\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0) + o(\|x\|^2)$

  Iteration scheme: $x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \cdot \nabla f(x_i)$

- <u>Damped Gauss-Newton:</u>

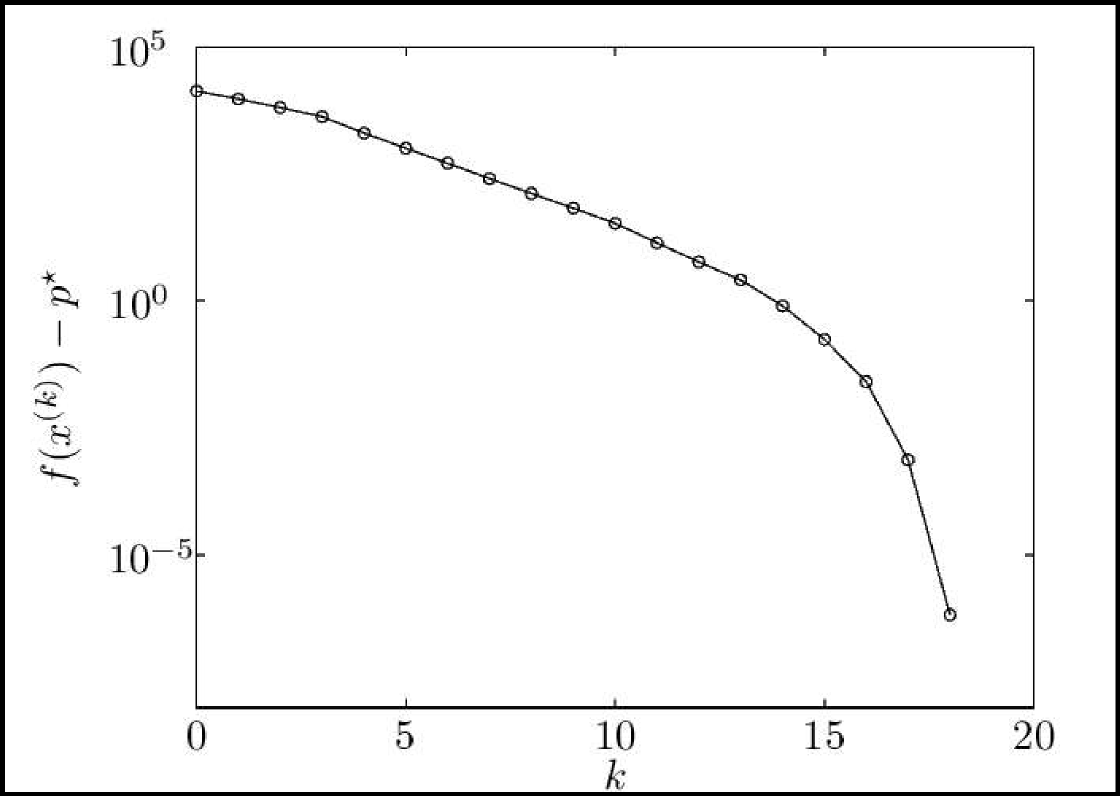
  Iteration scheme: $x_{i+1} = x_i - \alpha \cdot (\nabla^2 f(x_i))^{-1} \cdot \nabla f(x_i)$

  where α is chosen to minimize: $f(x_{i+1})$

Gauss-Newton (2/6)

$$\boxed{x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \cdot \nabla f(x_i)}$$

- <u>Example:</u> $-\sum \log(1 - x_i^2) - \sum \log(b_i - a_i^T x)$



Gauss-Newton (3/6)

$$\boxed{x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \cdot \nabla f(x_i)}$$

<u>Summary on Gauss-Newton:</u>

- fast convergence (quadratic close to the minimum
- scales well with problem size
- not dependent on the choice of parameters

Gauss-Newton (4/6)

<u>If the minimum is 0:</u>

if the residuals are too big

- the Newton method can be even faster and more accurate.
- the Gauss-Newton method can even not converge

Gauss-Newton (5/6)

- Linear least squares minimization: $\boxed{f : x \rightarrow \frac{1}{2}\sum_{j=1}^{m} r_j(x)^2}$

  Case where the residuals $r_i(x)$ is linear $r_i(x) = A_i \cdot x - b_i$

- Using the Jacobian $J = \frac{\partial r_j}{\partial x_i}$ we can rewrite f as :

$$f = \frac{1}{2}\|J \cdot x + r(0)\|^2$$

We seek for x / $\nabla f = J^T(J \cdot x + r(0)) = 0$

Solution: $\boxed{x_{min} = -(J^T J)^{-1} J^T \cdot r(0)}$

Gauss-Newton (6/6)

$$\boxed{x_{i+1} = x_i - (\nabla^2 f(x_i))^{-1} \cdot \nabla f(x_i)}$$

- Exactly:

$$\nabla f = \sum_{j=1}^{m} r_j \cdot \nabla r_j = J \cdot r$$

$$\nabla^2 f = J^T J + \sum_{j=1}^{m} r_j \cdot \nabla^2 r_j$$

- Using the linear approximation: Hessian matrix

$$H = \nabla^2 f \approx J^T J$$

Levenberg-Marquardt (1/4)

- Gauss-Newton: fast convergence but sensitive to the starting location.
- Gradient Descent: the opposite.
- <u>Levenberg algorithm:</u> combining both

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \cdot \nabla f(x_i)$$

  - if error goes down, reduce λ
  - else augment λ

Levenberg-Marquardt (2/4)

- Example of evolution:
  - if error goes down, reduce λ

$$\lambda_{k+1} = \frac{\lambda_k}{(1+\alpha)}$$

  - else augment λ

$$\lambda_{k+1} = \frac{min_k f - (J(x_k) \cdot d_k + f(x_k))}{\alpha}$$

Levenberg-Marquardt (3/4)

- Going faster when the gradient is low
- Hessian H $\propto$ curvature

$$\boxed{x_{i+1} = x_i - (H + \lambda\, diag[H])^{-1} \cdot \nabla f(x_i)}$$

Levenberg-Marquardt (4/4)

LM is actually also a region-search method:

the problem is equivalent to solve for :

$$p' = argmin_{\|p\| \leq \Delta} f(x_i) + \nabla f(x_i) \cdot p + \frac{1}{2} p^T H p$$

iteration : $x_{i+1} = x_i + p'$

Implementation (1/1)

$H+\lambda\, diag[H]$ or $H+\lambda I$ can be forced to be definite positive. A Cholesky decomposition is possible.

$$H+\lambda I = L L^T$$

We want $(H+\lambda I)\cdot d = -\nabla f$

- First, we compute by forward substitution *w*/ $L w = -\nabla f$
- Then, we compute by backward substitution *d*/ $L^T d = w$

Implementation (1/2)

- $H+\lambda\, diag[H]$ or $H+\lambda I$ is usually sparse. A sparse Cholesky decomposition is possible.

$$H+\lambda I = PL\,L^T P^T$$

- We have $H+\lambda I = D + A^T H_0 A$

Limitations (1/7)

- Can be slow: we have to invert a matrix: $H_i + \lambda_i$

$$x_{i+1} = x_i - H_i^{-1} \cdot \nabla f(x_i)$$

- Fundamental theorem of integral calculus

$$\left\{ \int_0^1 \nabla^2 f(x_i + t\, s_i)\, dt \right\} s_i = y_i \quad \text{with} \quad \begin{array}{c} s_i = x_{i+1} - x_i \\ y_i = \nabla f(x_{i+1}) - \nabla f(x_i) \end{array}$$

average of the Hessian on $[x_i, x_{i+1}]$

H_{i+1}: Force positive definite and $H_{i+1} \cdot s_i = y_i$

Limitations (2/7)

- Update of the Hessian:

$$\boxed{H_{i+1} = H_i - \frac{H_i \cdot s_i (H_i \cdot s_i)^T}{s_i^T \cdot H_i \cdot s_i} + \frac{y_i y_i^T}{y_i^T s_i} + \Phi_i (s_i^T \cdot H_i \cdot s_i) v_i v_i^T}$$

with

$$v_i = \frac{y_i}{y_i^T s_i} - \frac{H_i \cdot s_i}{s_i^T H_i \cdot s_i}$$

Φ_i=0 Broyden-Fletcher-Goldfarb-Shanno update

Φ_i=1 Davidon-Fletcher-Powell update

Limitations (3/7)

- Usually $H_i+\lambda_i$ is not updated directly. Its inverse or its Cholesky decomposition is.
- Renders steepest-descent methods obsolete.

Limitations (4/7)

- A big matrix needs to be stored
- To reduce the memory use:

$$H = J^T J = [J_1 J_2]^T \begin{bmatrix} J_1 \\ J_2 \end{bmatrix} = J_1^T J_1 + J_2^T J_2$$

Limitations (5/7)

If there are two many variables, H is hard to invert

<u>Truncated Newton methods</u>: before line search, find d_k /

$$\|\nabla^2 f(x_k) \cdot d_k + \nabla f(x_k)\| \leqslant \eta_k \|\nabla f(x_k)\|$$

Limitations (6/7)

Perturbation sensitivity:

$$J = USV^T$$

$$f(x_0 + d) \approx f(x_0) + J(x_0)\, d$$

$$f(x_0 + d) \approx f(x_0) + u_1 s_1 v_1^T d + \ldots + u_n s_n v_n^T d$$

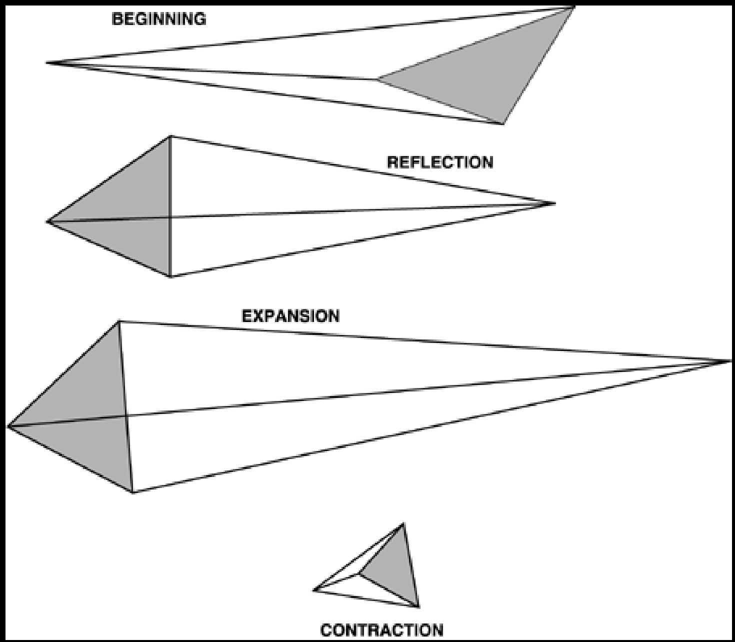
residual values are more sensitive to changes in the u_1 direction

Limitations (7/7)

If ∇f is too hard to compute: <u>Non-linear Simplex</u>

(Downhill Simplex method, Nelder-Mead, 1965))

for an n-dimensional problem, an n+1 simplex is used and distorted in order to find a minimizer

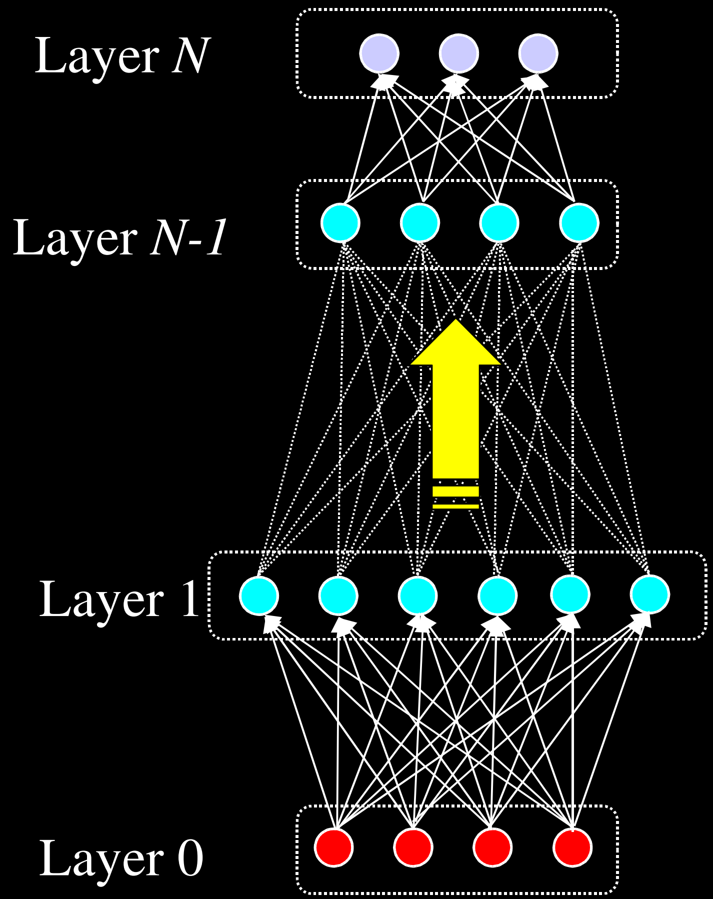


Application (1/8)

- Has a quadratic rate of convergence: good rate of convergence
- $H+\lambda\, diag[H]$ or $H+\lambda I$ can be forced to be invertible (when the Jacobian is not full rank and/or the pseudoinverse does not exist)

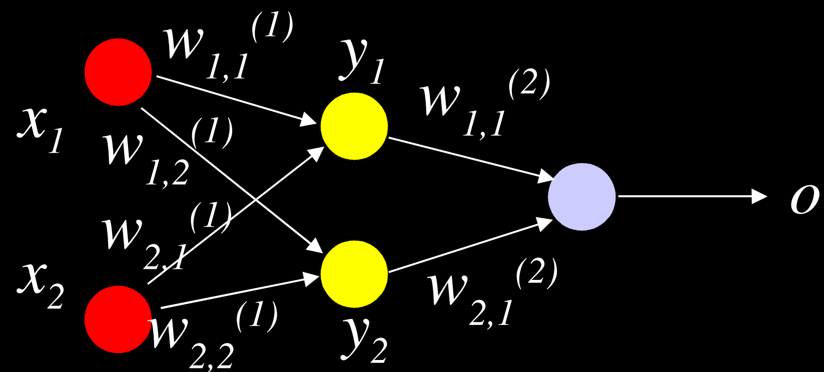
Applications (2/8)

Neural Networks: fastest method for training moderate-sized feedforward neural networks



Applications (3/8)

Basic Adaline



Layer 1

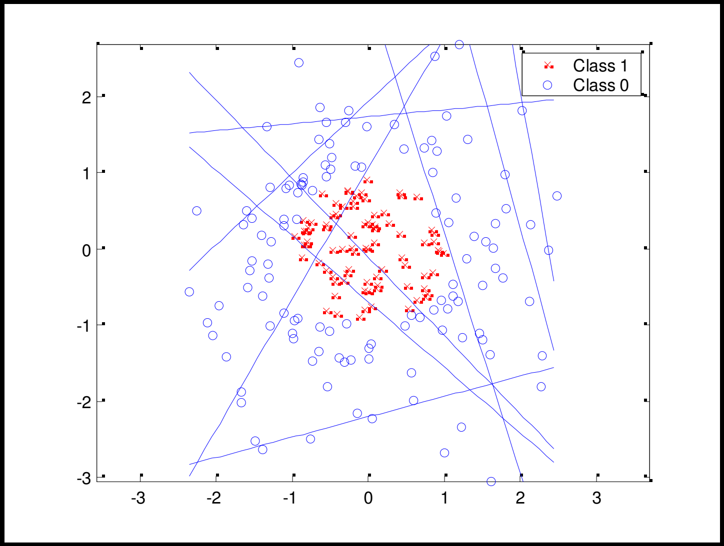
$$y_1 = f\,(w_{1,1}^{(1)} x_1 + w_{1,2}^{(1)} x_2 + \theta_1^{(1)})$$
$$y_2 = f\,(w_{2,1}^{(1)} x_1 + w_{2,2}^{(1)} x_2 + \theta_2^{(1)})$$

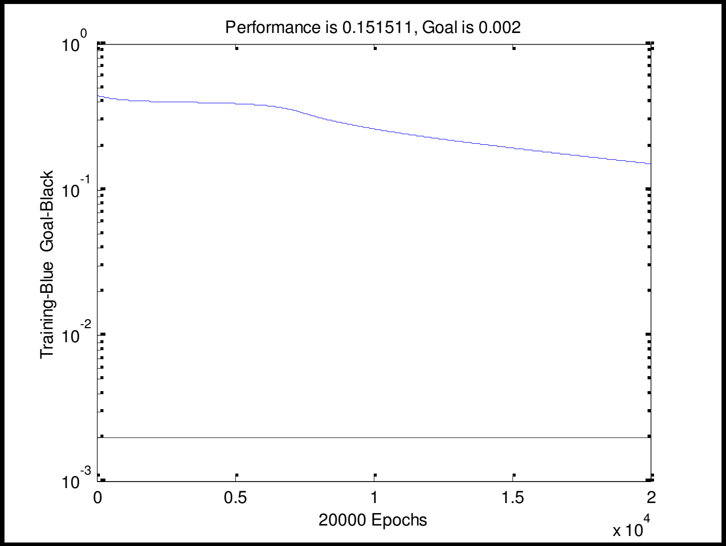
Layer 2

$$o = f\,(w_{1,1}^{(2)} y_1 + w_{1,2}^{(2)} y_2 + \theta_1^{(2)})$$

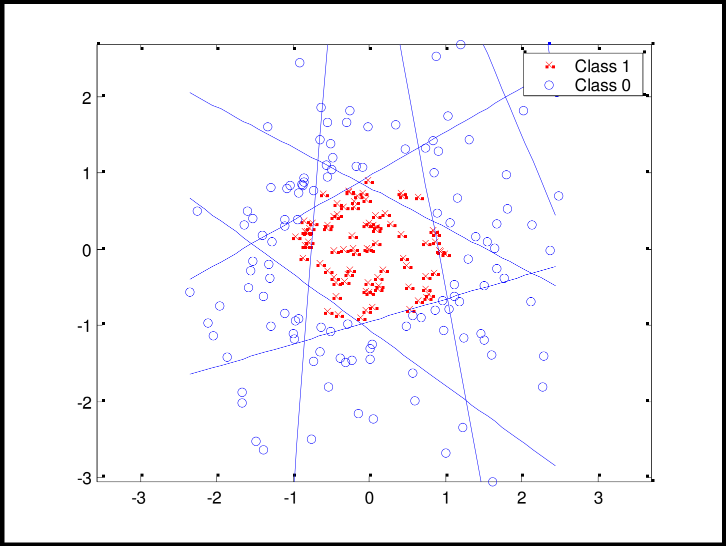
Applications (4/8)

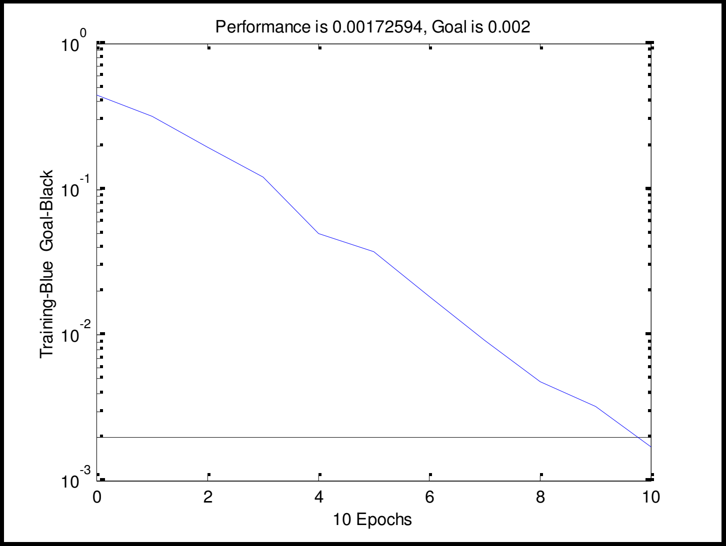
gradient descent





LM





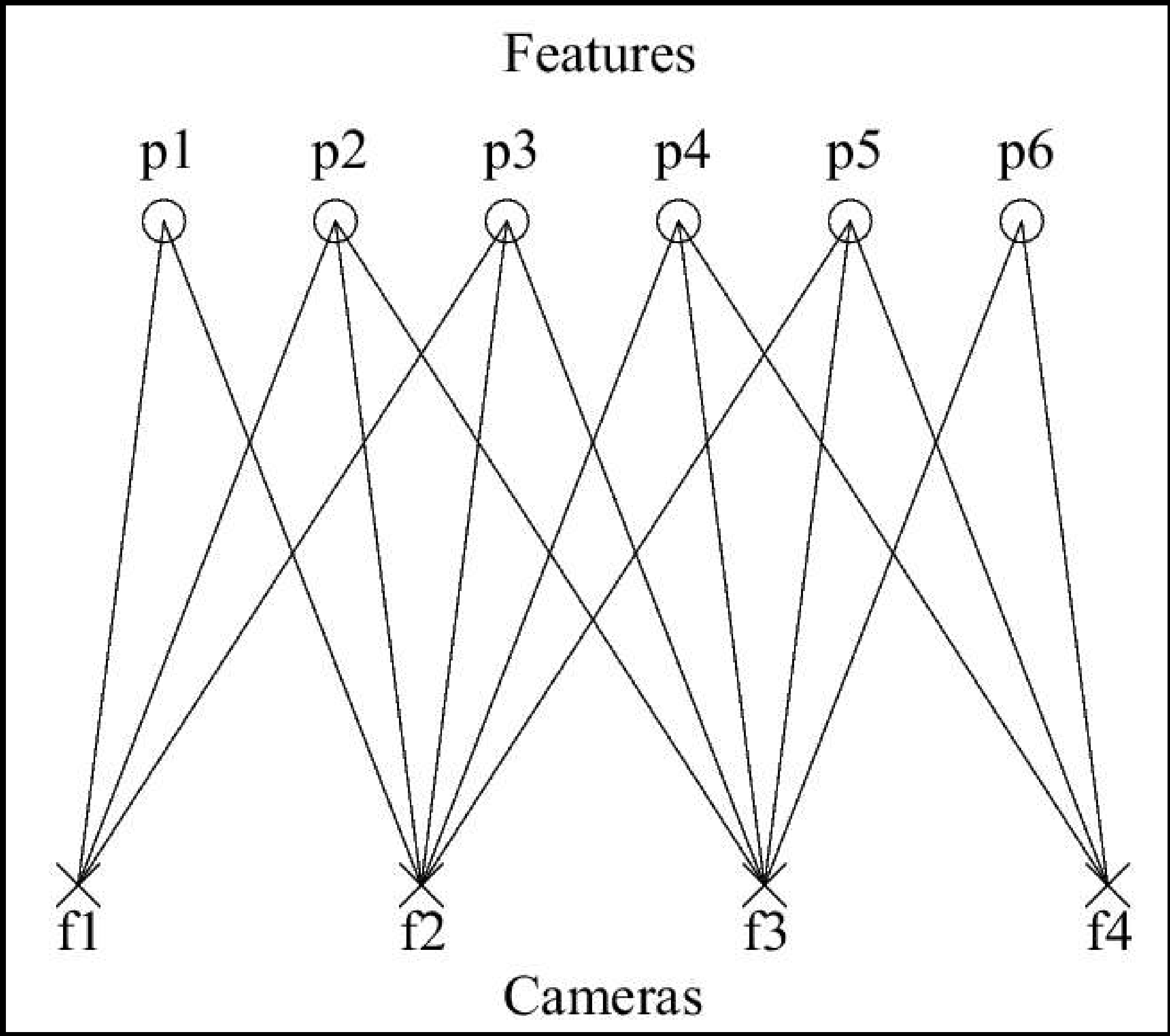
Applications (5/8)

Small number of weights: stabilized Newton and Gauss-Newton algorithms, Levenberg-Marquardt, trust-region algorithms. Memory: $O(N_w^2)$

Moderate number of weights: quasi-Newton algorithms are efficient. Memory: $O(N_w^2)$

Large number of weights: conjugate-gradient. Memory: $O(N_w)$

Applications (6/8)

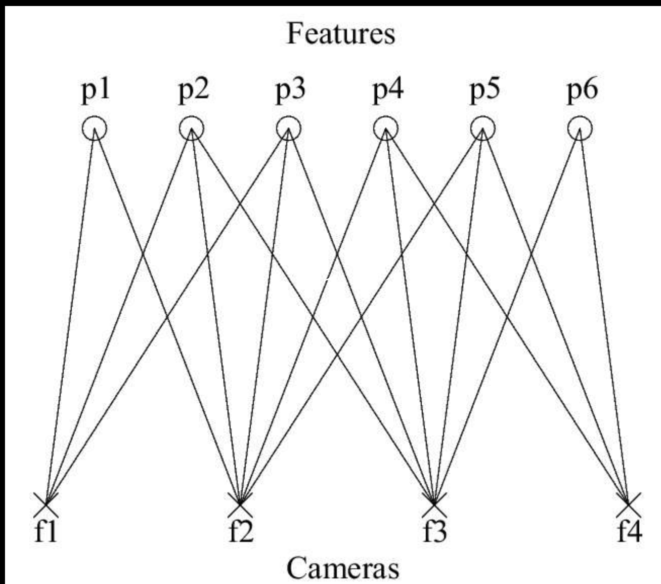


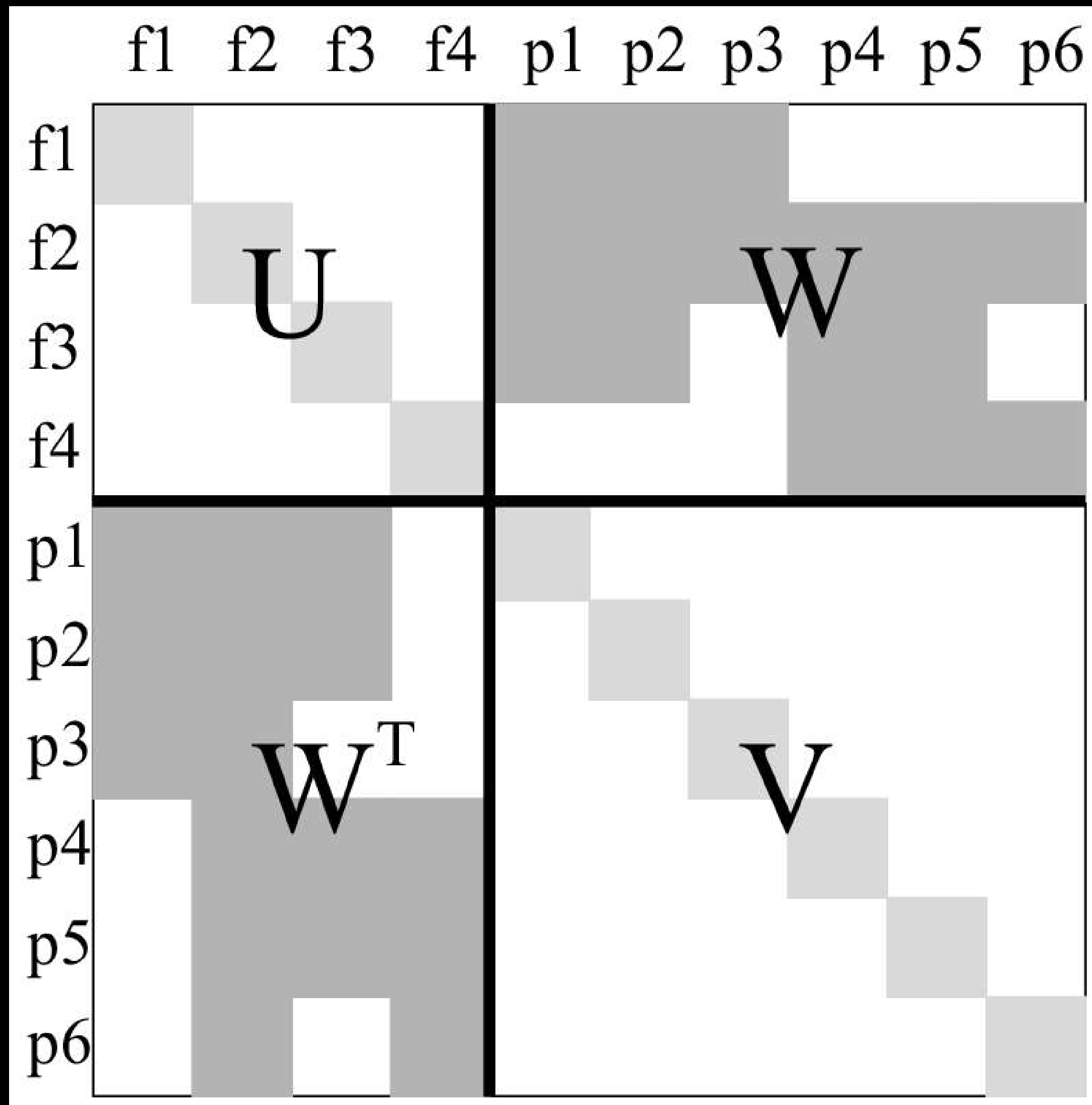
Applications (7/8)

- Objective function

$$\chi^2(theta) = \sum_{c \in C} \sum_{p \in P_c} \left\| \pi(\theta_c, \theta_p) - u_{p,c} \right\|^2$$

Applications (8/8)





Conclusion

- Levenberg-Marquardt has the best compromise between complexity and speed
- It works for many cases as it is at the border line:
  - between line-search and region-search
  - between Gauss-Newton and gradient descent
- Many other cases Hessian can have a different solution (faster, more accurate)
- Better methods (conjugate gradient...) but price to pay

References

- Downhill Simplex method, Nelder-Mead, 1965
- The Levenberg-Marquardt Algorithm, Ananth Ranganathan
- Multiple View Geometry in Computer Vision, Hartley & Zisserman
- Convex Optimization, S. Boyd, L. Vandenberghe
- An adaptative Nonlinear Least-Squares Algorithm, J. Dennis & D. Gay & R Welsh
- http://user.it.uu.se/~matsh/opt/
- http://www-fp.mcs.anl.gov/otc/Guide/OptWeb/continuous/unconstrained/
- Structure From Motion on Extended Sequences, D. Steedly