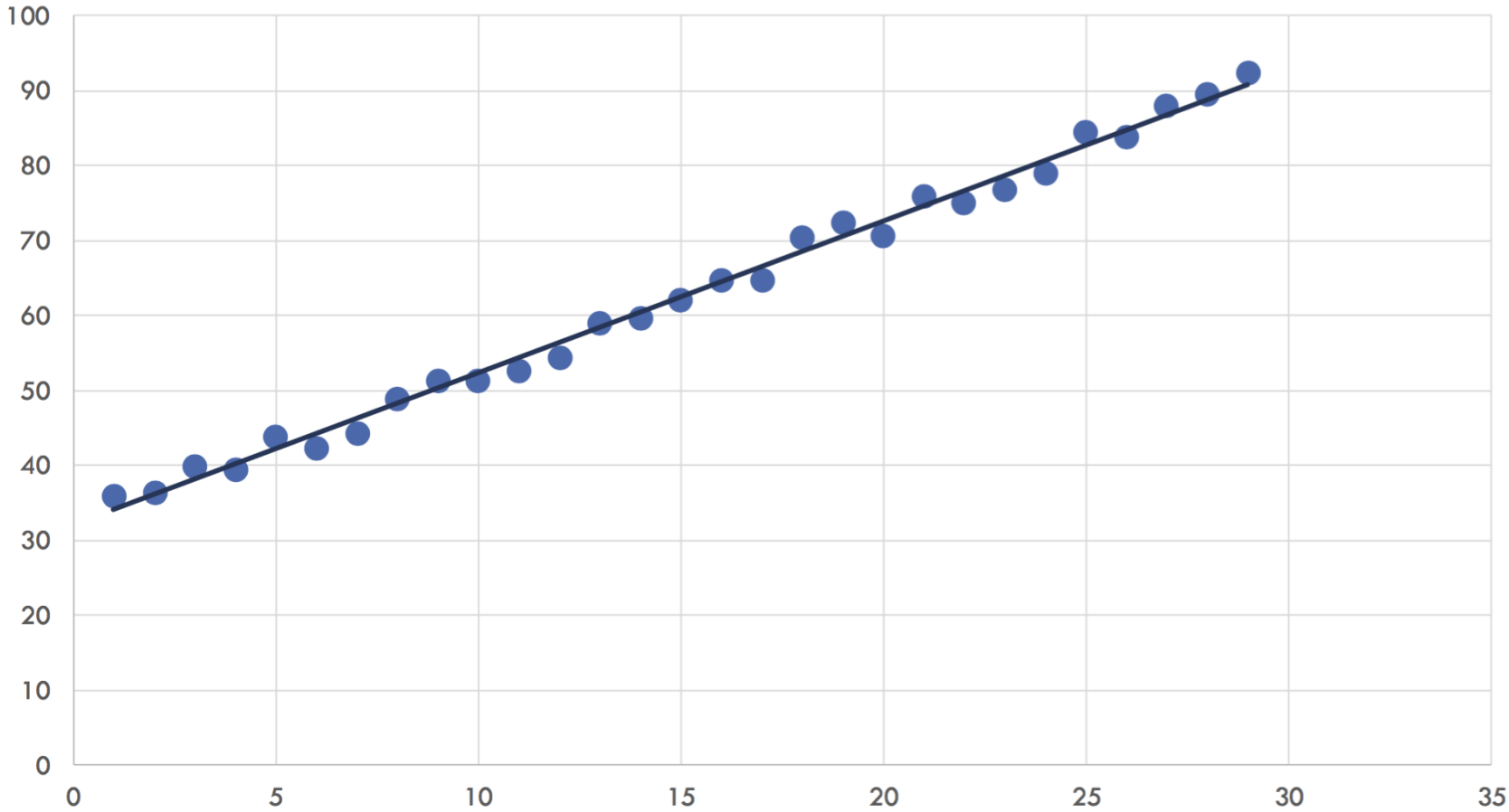
Convex Optimization

Hu SiXing, Hakki Can Karaimer, Pan An, Philipp Keck

National University of Singapore

Jan 20th, 2016

Linear Regression Example



Ordinary Least Squares

Input: points (x_i, y_i)

Regression line: $y = mx + b$

Objective:

$$\min_{m,b} \sum_i (y_i - mx_i - b)^2$$

$(\vec{x_i}, y_i)$

$y = \vec{w} \cdot \vec{x} + b$

$$\min_{\vec{w}} \sum_i (y_i - \vec{w} \cdot \vec{x_i} - b)^2$$

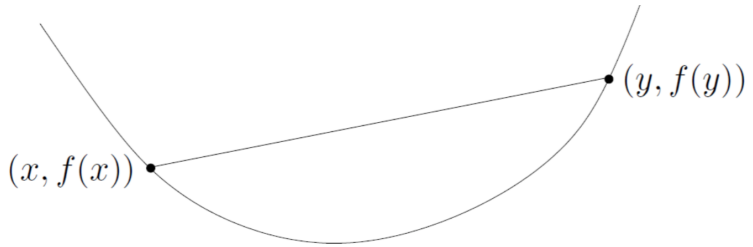
- Easily Solved: $\vec{w}^*(X^T X) - X^T \vec{y}$
- But what if $\dim \vec{x}$ is large?
- What about other similar regressions?

Convex Optimization Problems

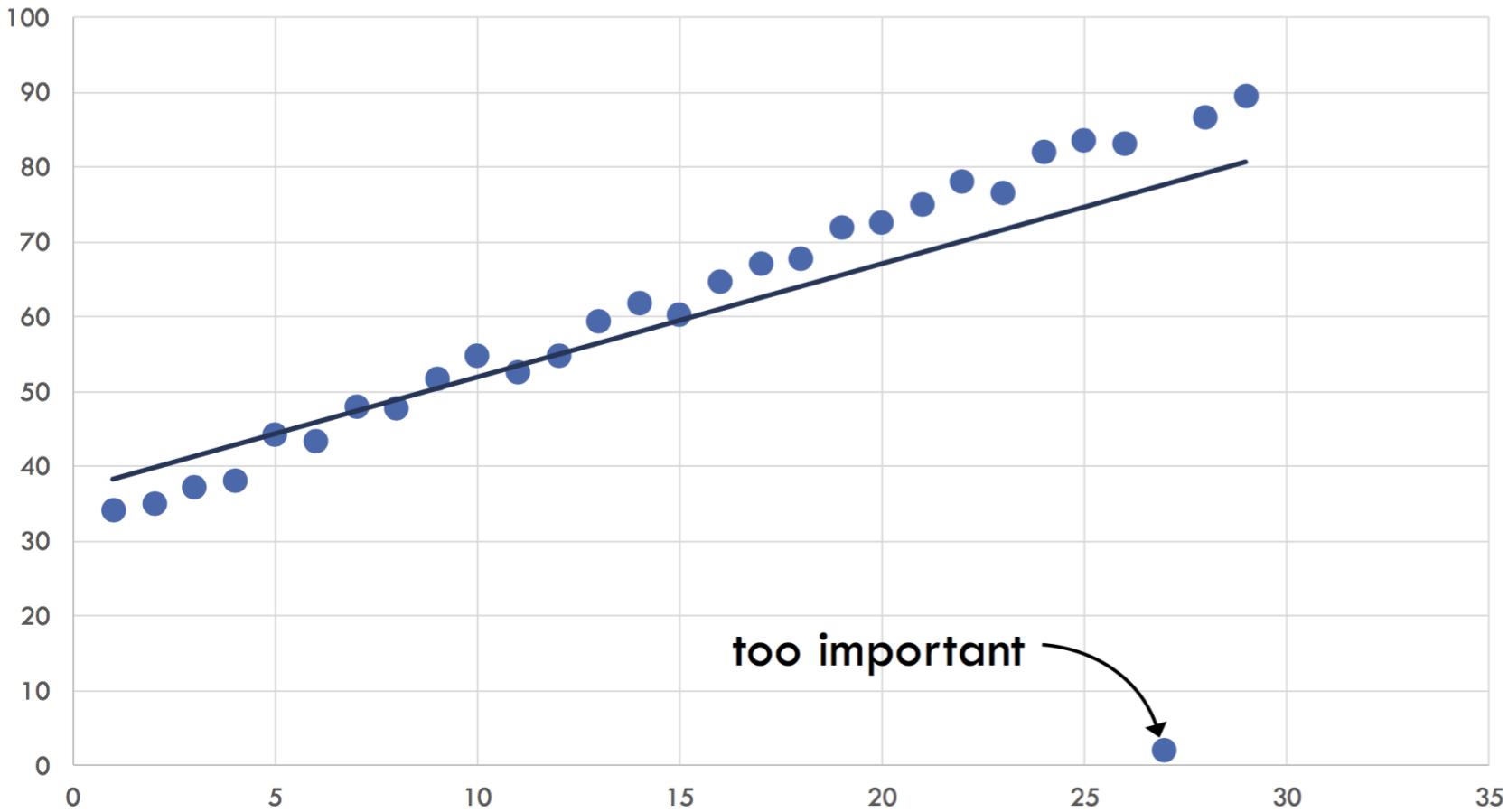
- OrdinaryLinearRegression: $\min_{\vec{w}} \sum_i (y_i - \vec{w} \cdot \vec{x_i})^2$
- General: $\min_x f(x)$ where $f(x)$ is convex
- Set C is convex $\iff \forall x, y \in C, 0 \leqslant t \leqslant 1 : tx + (1-t)y \in C$
- Function $f : \mathbb{R}^n \to \mathbb{R}$ is convex if $\mathbf{dom}\, f$ is convex and $\forall x, y \in \mathbf{dom}\, f, 0 \leqslant t \leqslant 1 :$

$$f(tx + (1-t)y) \leqslant tf(x) + (1-t)f(y)$$

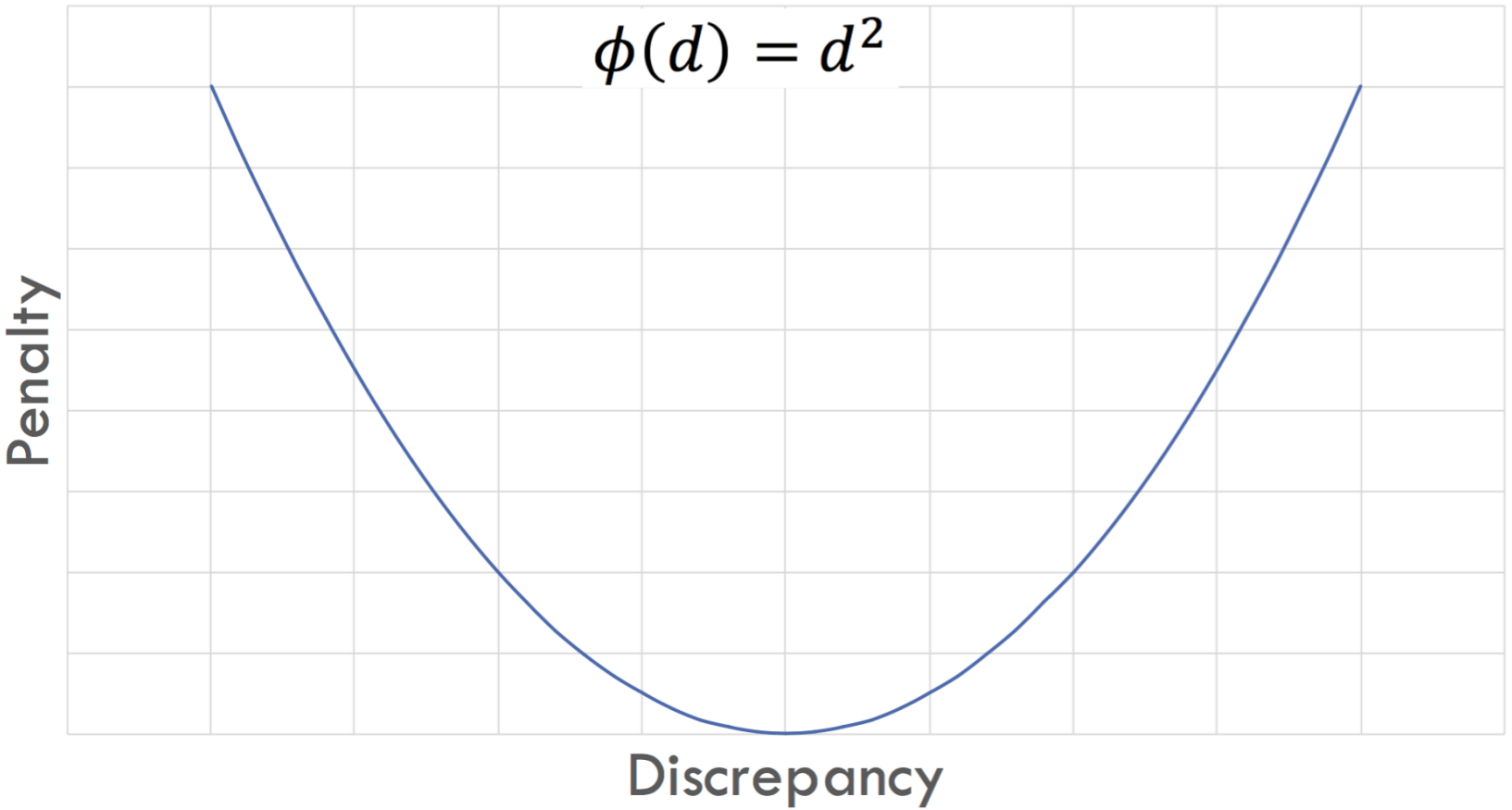
- Unconstrained.



Outliers

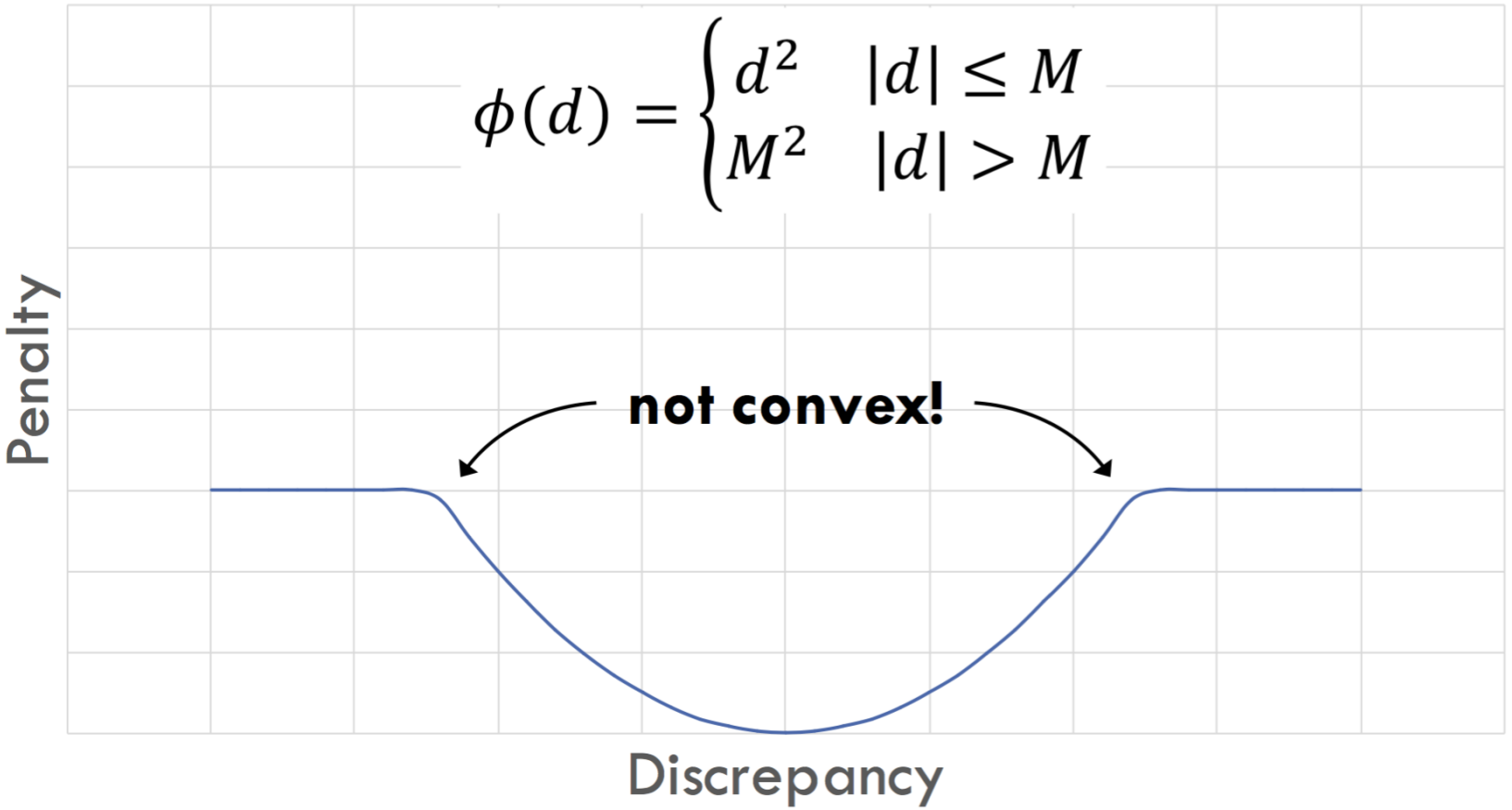


Outlier Penalty



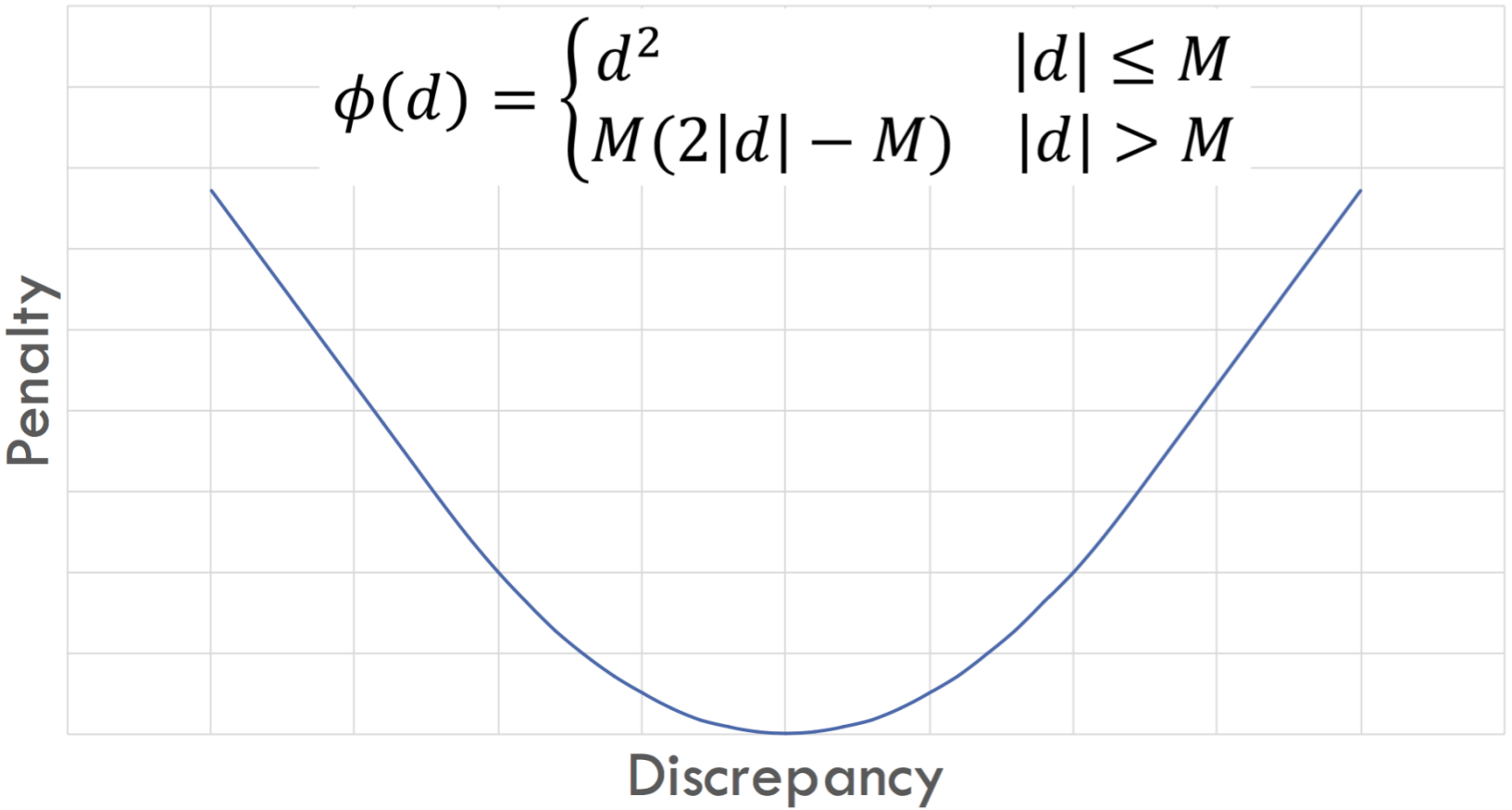
Capped Penalty

$$\phi(d) = \begin{cases} d^2 & |d| \leq M \\ M^2 & |d| > M \end{cases}$$



Huber Penalty Function

$$\phi(d) = \begin{cases} d^2 & |d| \leq M \\ M(2|d| - M) & |d| > M \end{cases}$$



Unconstrained Optimization

- Minimize $f(x)$;
- Where $f : \mathbb{R}^n \to \mathbb{R}$ is convex and twice differentiable;
- No additional constraints;
- Assume that unique minimum x^* exists.

General Principle

- Objective: minimize $f(x)$
- Necessary and sufficient condition: $\nabla f(x^*) = 0$
  - Solve analytically
  - Iterative algorithms

Iterative Algorithm:

$$x^{(0)}, x^{(1)}, \ldots \in \mathbf{dom}\, f$$

$$k \to \infty, f(x^{(k)}) \to f(x^*)$$

Descent Method:

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)}, \mathbf{s.t.} f(x^{(k+1)}) < f(x^{(k)})$$

General Descent Method

Descent Method:

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, \mathbf{s.t.} f(x^{(k+1)}) < f(x^{(k)}) \quad (1)$$

Algorithm:

Given $x^{(0)} \in \mathbf{dom}\, f$;
repeat
    Determine a descent direction Δx;
    Choose a step size $t > 0$;
    Update $x := x + t\Delta x$;
until stopping criterion is satisfied;

Therefore, Δx must satisfy:

$$\nabla f(x^{(k)})^T \Delta x^{(k)} < 0 \quad (2)$$

General Descent Method

Descent Method:

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, \mathbf{s.t.} f(x^{(k+1)}) < f(x^{(k)}) \quad (1)$$

Theorem

For a continuously differentiable function f:

$$f \text{ is convex} \Leftrightarrow f(x) \geqslant f(y) + f'(y)(x-y)$$

Based on the Theorem and Equation (1):

$$f(x^{(k)}) + \nabla f(x^{(k)})^T \Delta x^{(k)} \leqslant f(x^{(k+1)})$$
$$\nabla f(x^{(k)})^T \Delta x^{(k)} \leqslant f(x^{(k+1)}) - f(x^{(k)}) < 0$$

Therefore, Δx must satisfy:

$$\nabla f(x^{(k)})^T \Delta x^{(k)} < 0 \quad (2)$$

General Descent Method

Descent Method:

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, \mathbf{s.t.} f(x^{(k+1)}) < f(x^{(k)}) \quad (1)$$

Algorithm:

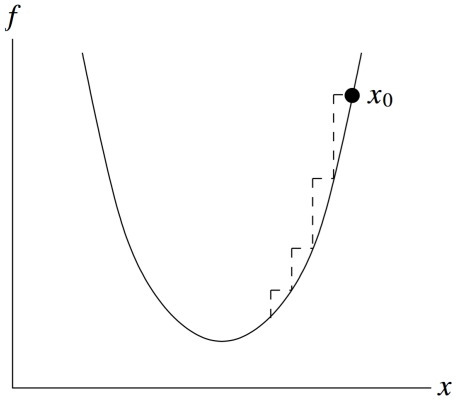
Given $x^{(0)} \in \mathbf{dom}\, f$;
repeat
  Determine a descent direction $\Delta x \Rightarrow$ Gradient/SteepestDescent;
  Choose a step size $t > 0 \Rightarrow$ LineSearchAlgo ;
  Update $x := x + t\Delta x$;
until stopping criterion is satisfied;

Therefore, Δx must satisfy:

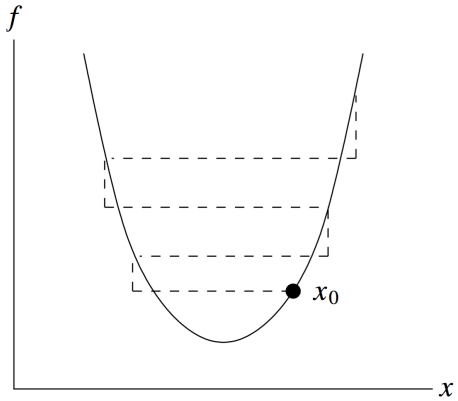
$$\nabla f(x^{(k)})^T \Delta x^{(k)} < 0 \quad (2)$$

Line Search

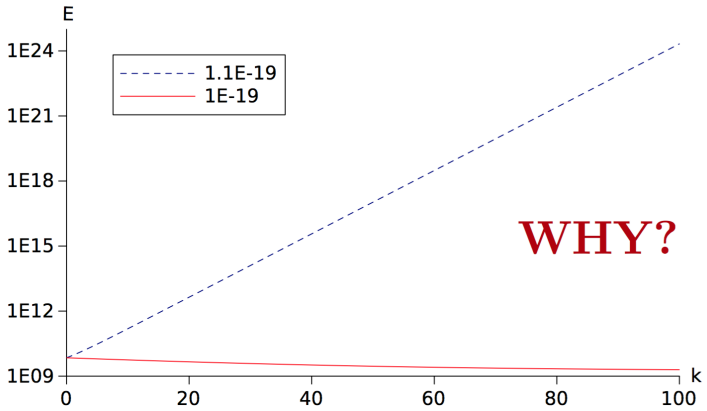
$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, f(x^{(k+1)}) \leftarrow f(x^{(k)})$$

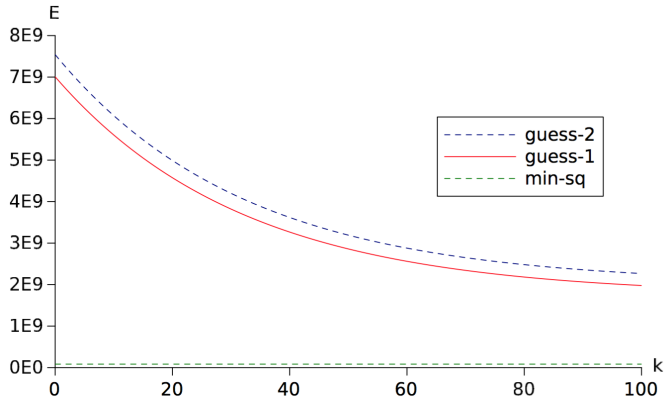


When Step Size t is Appropriate.



When Step Size t is Inappropriate.





Line Search

- Armijo

$$f(x^{(k)} + t\Delta x^{(k)}) \leqslant f(x^{(k)}) + \alpha_1 t \nabla f(x^{(k)})^T \Delta x^{(k)}, \alpha_1 > 0$$

- Wolfe Conditions (Including Armijo Condition):

$$\nabla f(x^{(k)} + t\Delta x^{(k)})^T \Delta x^{(k)} \geqslant \alpha_2 \nabla f(x^{(k)})^T \Delta x^{(k)}, 0 < \alpha_1 < \alpha_2 < 1$$

Where Δx is the step direction.

Theorem:

Gradient descent will find local minimum if step size t satisfies Wolfe conditions.

Line Search

- Armijo Condition:

$$f(x^{(k)} + t\Delta x^{(k)}) \leqslant f(x^{(k)}) + \alpha t \nabla f(x^{(k)})^T \Delta x^{(k)}, \alpha > 0$$

Backtracking Line Search:

Given a descent direction Δx for f at
$x \in \mathbf{dom}\, f, \alpha \in (0, 0.5), \beta \in (0, 1), t := 1;$
while $f(x + t\Delta x) > f(x) + \alpha t \nabla f(x)^T \Delta x$ do
  $t := \beta t;$
end

Exact Line Search Method:

$$t = \underset{s \geqslant 0}{\operatorname{argmin}} \{f(x + s\Delta x)\}$$

General Descent Method

- Gradient Descent Method
- Steepest Descent Method

Δx satisfies:

$$\nabla f(x^{(k)})^T \Delta x^{(k)} < 0$$

Gradient Descent Method

$\Delta x = -\nabla f(x)$

Given $x^{(0)} \in \mathbf{dom}\, f$;
repeat
    $\Delta x = -\nabla f(x)$;
    Choose a step size $t > 0$, [LineSearch];
    Update $x := x + t\Delta x$;
until stopping criterion is satisfied;

Steepest Descent Method

$\Delta x = \Delta x_{sd}$

•Taylor Series:

$$f(\mathbf{x} + \Delta\mathbf{x}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \Delta\mathbf{x} + \frac{1}{2}\Delta\mathbf{x}\nabla^2 f(\mathbf{x})\Delta\mathbf{x} + ...$$

$$f(\mathbf{x} + \mathbf{v}) \approx \hat{f}(\mathbf{x} + \mathbf{v}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{v}$$

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)}, \mathbf{s.t.} f(x^{(k+1)}) < f(x^{(k)})$$

Where $\mathbf{v}$ is a descent direction if $\nabla f(\mathbf{x})^T \mathbf{v} < 0$

Steepest Descent Method

$$f(\mathbf{x}+\mathbf{v}) \approx \hat{f}(\mathbf{x}+\mathbf{v}) = f(\mathbf{x}) + \nabla f(\mathbf{x})^T \mathbf{v}$$

- Normalized Steepest Descent Direction:

$$\begin{aligned} \Delta \mathbf{x}_{nsd} &= \operatorname{argmin}\{\nabla f(\mathbf{x})^T \mathbf{v} \mid \|\mathbf{v}\| = 1\} \\ &= \operatorname{argmin}\{\nabla f(\mathbf{x})^T \mathbf{v} \mid \|\mathbf{v}\| \leqslant 1\} \end{aligned} \tag{3}$$

Steepest Descent Method

Dual Norm, denoted $\|\cdot\|_*$, is defined as:

$$\|z\|_* = \sup\{z^T x \mid \|x\| \leqslant 1\}$$

Unnormalized Steepest Descent Direction:

$$\Delta x = \|\nabla f(x)\|_* \cdot \Delta x_{nsd}$$

$$\begin{aligned}\nabla f(x)^T v &= \nabla f(x)^T \Delta x_{sd} \\ &= \|\nabla f(x)\|_* \nabla f(x)^T \Delta x_{nsd} \\ &= -\|\nabla f(x)\|_*^2\end{aligned}$$

Proof

$$\begin{aligned}\Delta x_{nsd} &= \operatorname{argmin}\{\nabla f(x)^T v \mid \|v\| \leqslant 1\} \\ &= -\operatorname{argmax}\{\nabla f(x)^T v \mid \|v\| \leqslant 1\}\end{aligned}$$

$$\begin{aligned}\|\nabla f(x)\|_* &= \sup\{\nabla f(x)^T v \mid \|v\| \leqslant 1\} \\ \Rightarrow \|\nabla f(x)\|_* &= -\nabla f(x)^T \Delta x_{nsd}\end{aligned}$$

Steepest Descent Method

$$\Delta x_{nsd} = \operatorname{argmin}\{\nabla f(x)^T v \mid \|v\| \leqslant 1\}$$
$$\Delta x_{sd} = \|\nabla f(x)\|_* \, \Delta x_{nsd}$$

Steepest Descent Method

Given $x \in \operatorname{dom} f$;
repeat
  Compute steepest descent direction Δx_{sd};
  Choose a step size $t > 0$, [LineSearch];
  Update $x := x + t\Delta x_{sd}$;
until stopping criterion is satisfied;

When exact line search is used, scale factors in the descent direction have no effect.

Descent Method

General

Given $x \in \operatorname{dom} f$;
repeat
  Determine a descent direction Δx;
  Choose a step size $t > 0$;
  Update $x := x + t\Delta x$;
until stopping criterion is satisfied;

Gradient Descent

Given $x \in \operatorname{dom} f$;
repeat
  $\Delta x = -\nabla f(x)$;
  Choose a step size $t > 0$, [LineSearch];
  Update $x := x + t\Delta x$;
until stable stopping criterion is satisfied;

Steepest Descent

Given $x \in \operatorname{dom} f$;
repeat
  Compute steepest descent direction Δx_{sd};
  Choose a step size $t > 0$, [LineSearch];
  Update $x := x + t\Delta x_{sd}$;
until stable stable stopping criterion is satisfied;

Descent Method

General

- $\Delta \mathbf{x}_{nsd} = \text{argmin}\{\nabla f(\mathbf{x})^T \mathbf{v} \mid \|\mathbf{v}\| \leqslant 1\}$
- $\Delta \mathbf{x}_{sd} = \|\nabla f(x)\|_* \cdot \Delta \mathbf{x}_{nsd}$
- If the norm $\|\cdot\|$ is Euclidean norm, $\Delta \mathbf{x} = -\nabla f(\mathbf{x})$, which means that Gradient Descent and Steepest Descent become the same.

Gradient Descent

Given $x \in \text{dom}\, f$;
repeat
  $\Delta x = -\nabla f(x)$;
  Choose a step size $t > 0$, [LineSearch];
  Update $x := x + t\Delta x$;
until stable stopping criterion is satisfied;

Steepest Descent

Given $x \in \text{dom}\, f$;
repeat
  Compute steepest descent direction Δx_{sd};
  Choose a step size $t > 0$, [LineSearch];
  Update $x := x + t\Delta x_{sd}$;
until stable stable stopping criterion is satisfied;

Convex Domain

•Linear Regression method is applicable only if nonlinear function is linear in terms of function parameters:

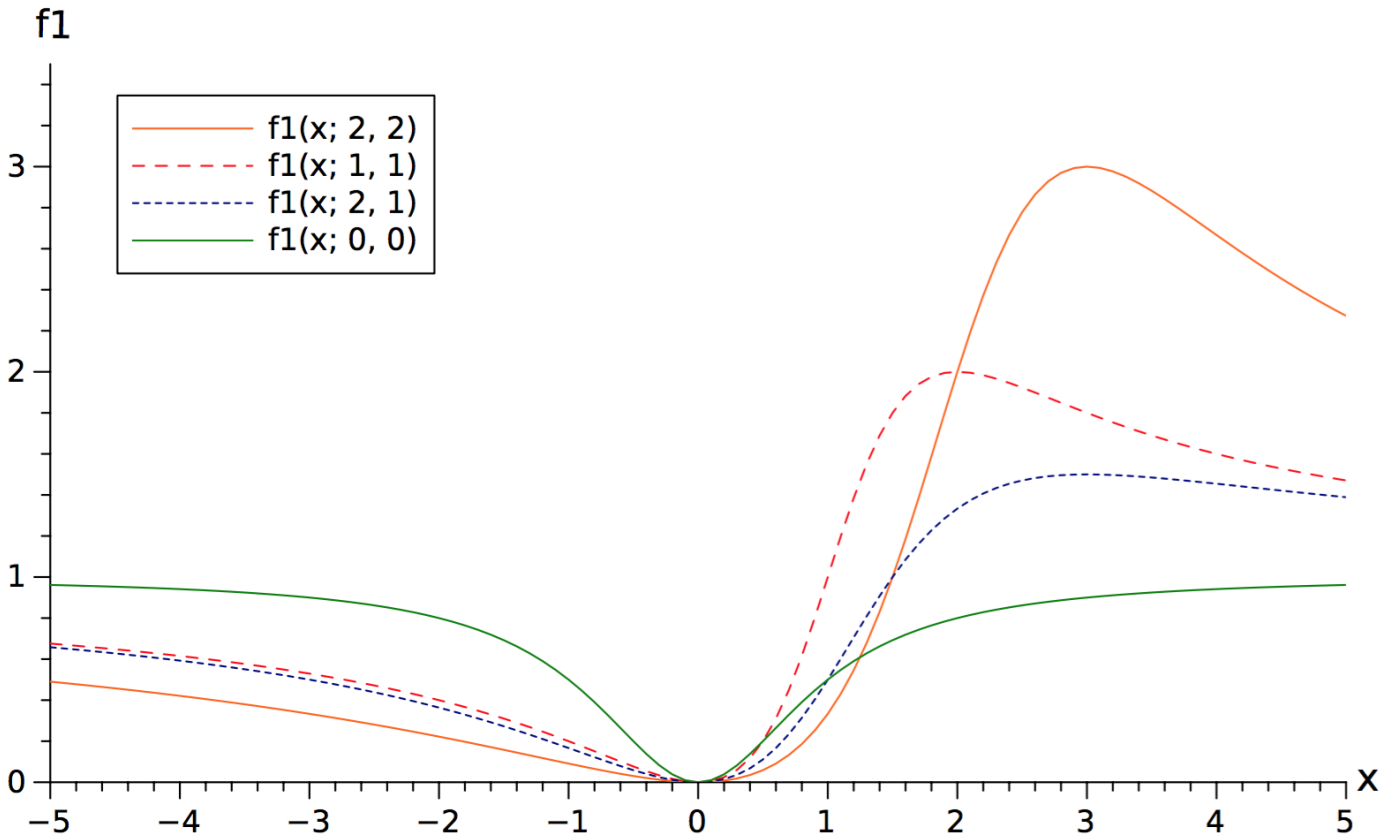
$$f(x; a) = \sum_{k=1}^{m} a_k h_k(x)$$

•Many nonlinear functions are not like that, for example:

$$f_1(x) = \frac{x^2}{a_1 + (x - a_2)}$$

$$f_2(x, y, z) = \frac{x^2}{a_1 + x^2} + \frac{y^2}{a_2 + y^2} + \frac{z^2}{a_3 + z^2}$$

Convex Domain



To minimize the error, we need iterative optimization.

Advantages – Disadvantages – Limitations

•If step length is appropriate, f always decreases: converge. (well conditioned)

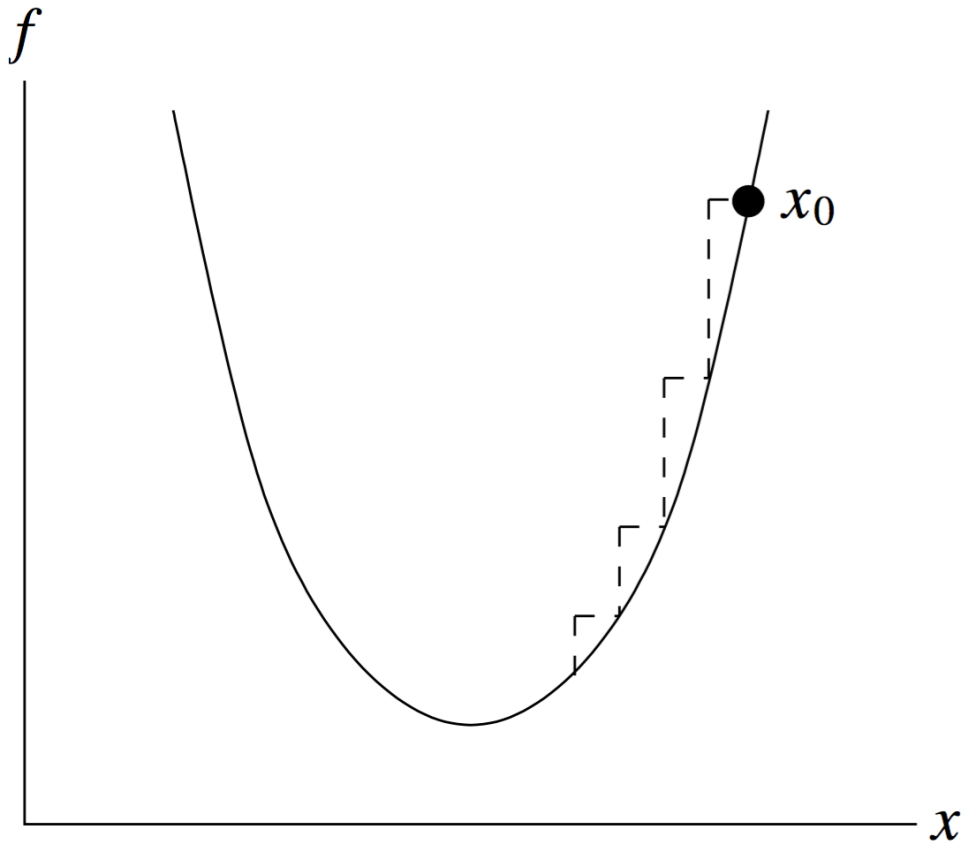


Figure: Well Conditioned

Advantages – Disadvantages – Limitations

•If step length is too large, f can increase: diverge. (ill condition)

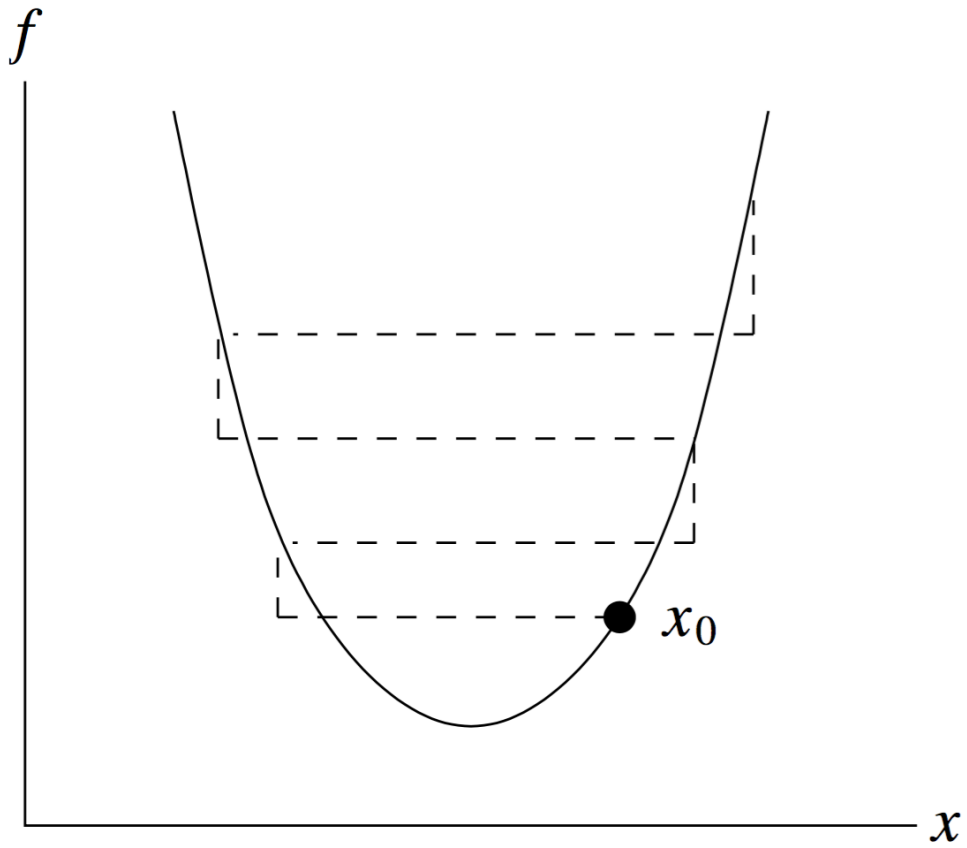


Figure: Ill Conditioned.

Advantages – Disadvantages – Limitations

- If parameters of f affect error equally,

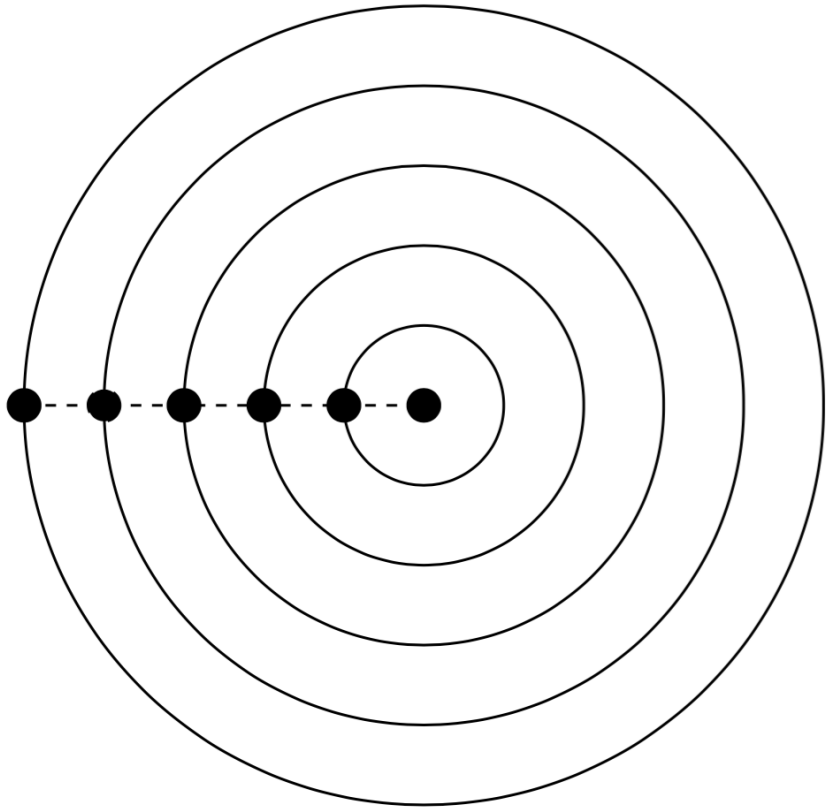


Figure: Straight path, fast convergence. (well condition)

Advantages – Disadvantages – Limitations

- If parameters of f affect error unequally,

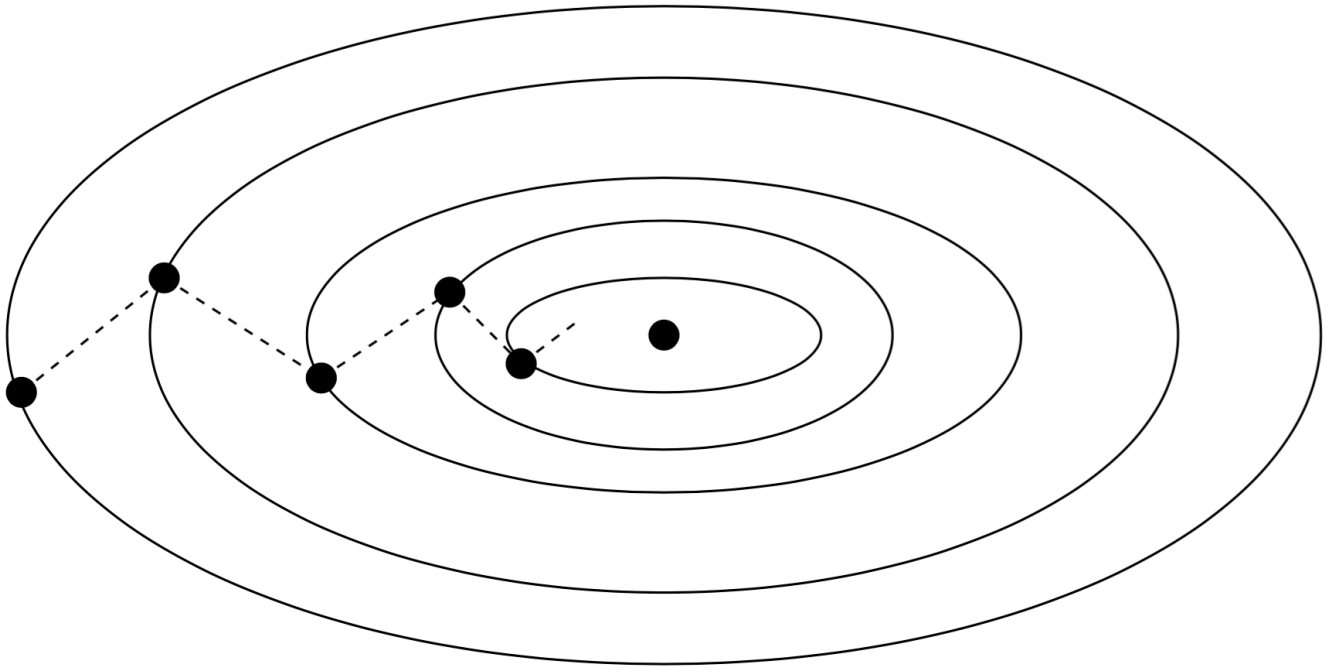


Figure: Jagged path, slow convergence. (ill condition)

Advantages – Disadvantages – Limitations

- If parameters of f affect error very unequally,

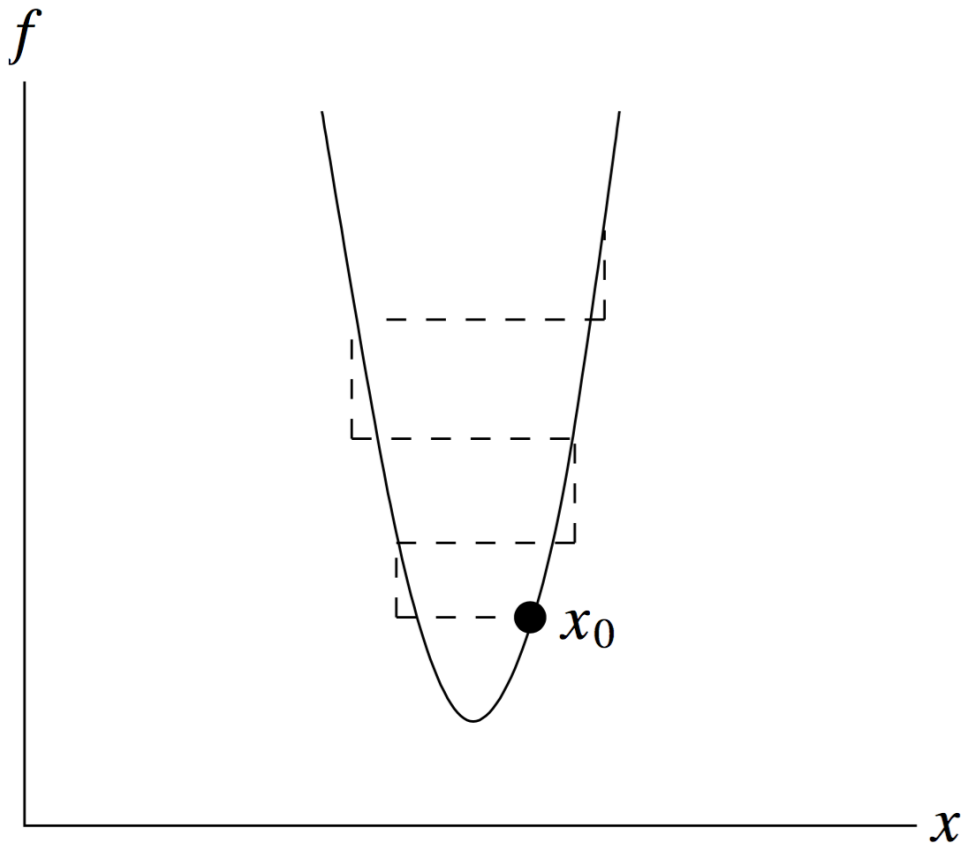
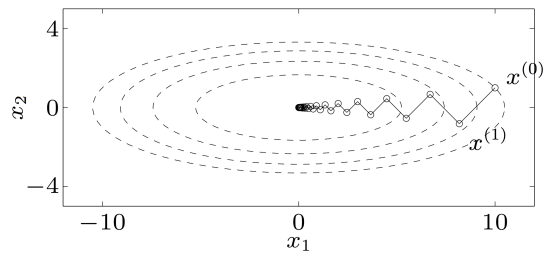


Figure: Small step length can also cause divergence. (ill condition)

Condition Number

- The condition number of C gives a measure of its anisotropy or eccentricity.
- If the condition number of a set C is small (say, near one) it means that the set has approximately the same width in all directions, i.e., it is nearly spherical.
- If the condition number is large, it means that the set is far wider in some directions than in others.
- $\mathbf{cond}(\mathrm{f}) = \frac{\lambda_{max}(\mathrm{f})}{\lambda_{min}(\mathrm{f})}$
- λ_{max} and λ_{min} describes minimum and maximum eigenvalues in 2D.



Example Quadratic Problem in $\mathbb{R}^2$

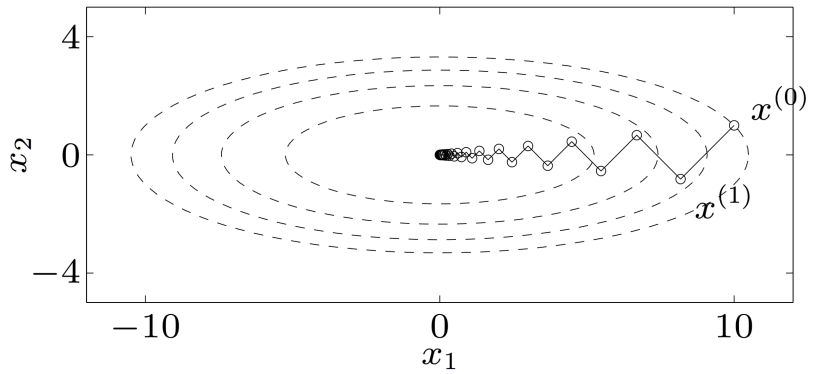
$$f(x) = \frac{1}{2}(x_1^2 + \gamma x_2^2) \quad \gamma > 0 \tag{3}$$

with exact line search, starting at $x^{(0)} = (\gamma, 1)$

$$x_1^{(k)} = \gamma(\frac{\gamma-1}{\gamma+1})^k, \quad x_2^{(k)} = \gamma(-\frac{\gamma-1}{\gamma+1})^k$$

- Hessian of f has eigenvalues 1 and γ. And $m = \min\{1, \gamma\}$,and $M = \max\{1, \gamma\}$
- In particular,$f(x_k)$ converges to optimal value p^*, at least as fast as a geometric series with an exponent that depends (at least in part) on the condition number bound $\frac{M}{m}$.

- Very slow if $\gamma > 1$ or $\gamma < 1$
- Useless if $\gamma > 20$.
- Example for $\gamma = 10$.



Advantages – Disadvantages – Limitations

- Left: Number of iterations of the gradient method as a function of γ which can be thought of as amount of diagonal scaling.
- Right: Condition number of the Hessian of the function at its minimum as a function of γ.
- We see that the condition number has a very strong influence on convergence rate.

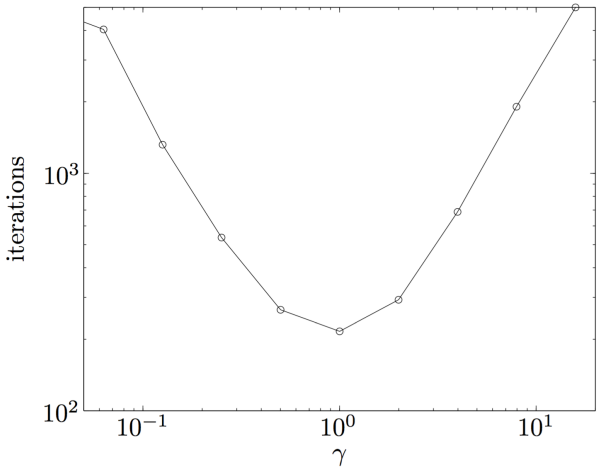


Figure: The vertical axis shows the number of iterations required to find the optimum. The horizontal axis shows γ, which is the parameter that controls the amount of diagonal scaling. We use a backtracking line search with $\alpha = 0.3, \beta = 0.7$.

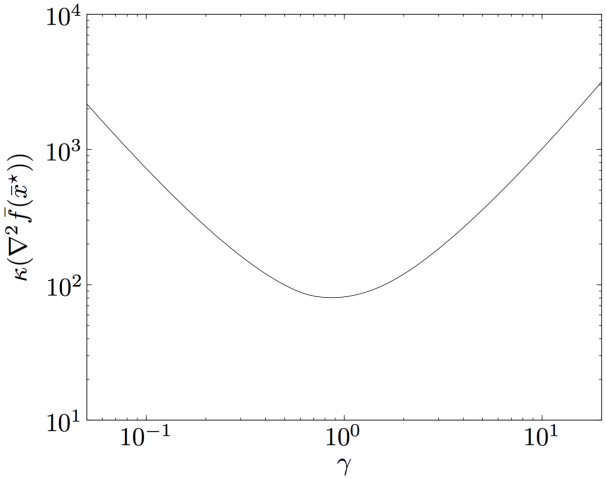
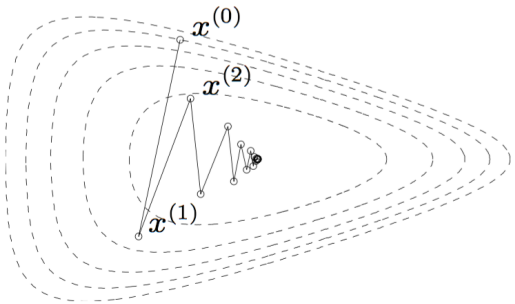


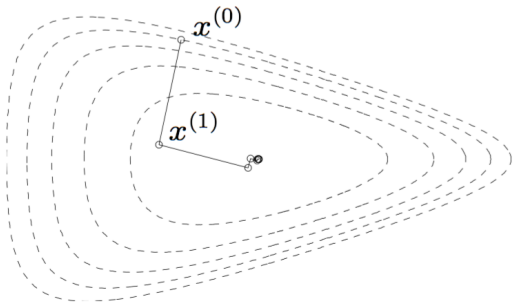
Figure: Condition number of the Hessian of the function at its minimum, as a function of γ. By comparing this plot with the one in the left figure, we see that the condition number has a very strong influence on convergence rate.

Exact Line Search VS. Backtracking Line Search with Non-Quadratic Example

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

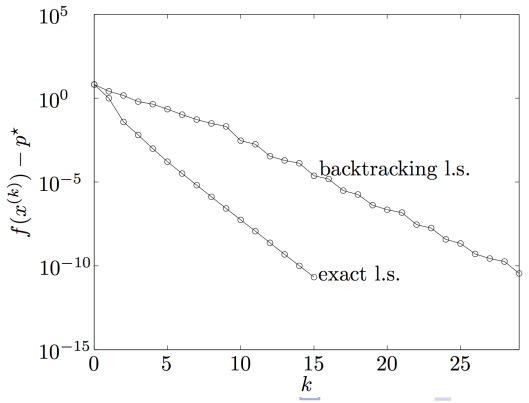


backtracking line search



exact line search

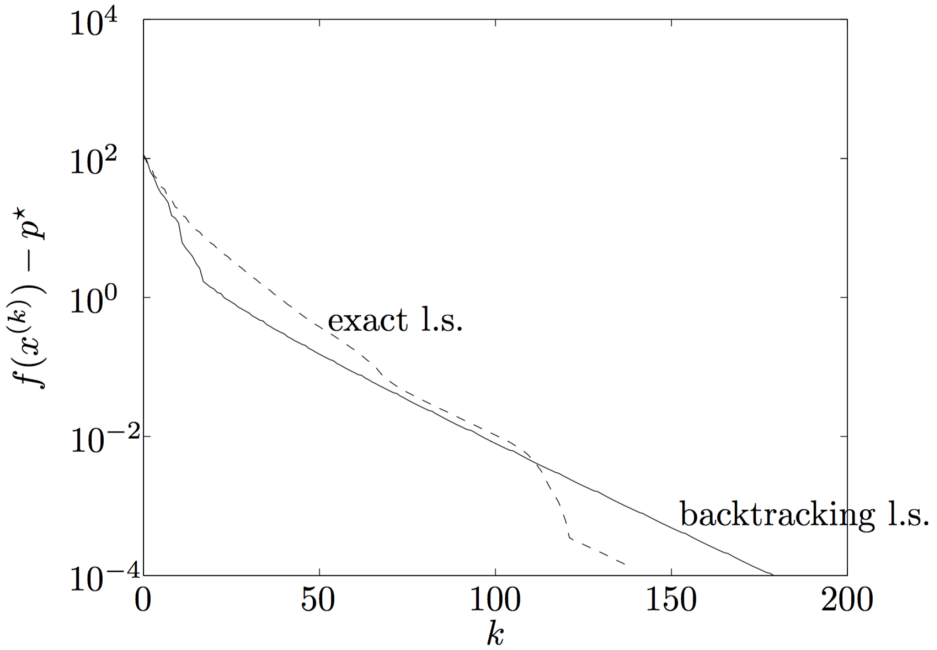
- With backtracking line search, the error is reduced about 10^{-11} in 15 iterations, i.e., a reduction by a factor of about $10^{-\frac{11}{15}} \approx 0.2$ per iteration.
- With exact line search, the error is reduced by about 10^{-8} in 20 iterations, i.e., a reduction by a factor of about $10^{-\frac{8}{20}} \approx 0.4$ per iteration.



Exact Line Search VS. Backtracking Line Search with a Problem in R^{100}

$$f(x) = c^T x - \sum_{i=1}^{500} \log(b_i - a_i^T x)$$

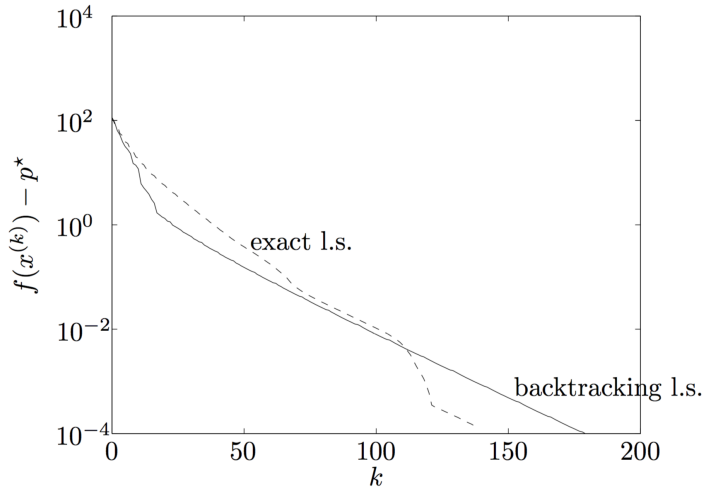
A larger example of this form with $m = 500$ terms and $n = 100$ variables.



Exact Line Search VS. Backtracking Line Search with a Problem in R^{100}

- The progress of the gradient method with backtracking line search, with parameters $\alpha = 0.1, \beta = 0.5$
- Average error reduction is $10^{-\frac{6}{175}} \approx 0.92$ per iteration.

- In the case of the gradient method with exact line search, average error reduction is $10^{-\frac{6}{140}} \approx 0.91$ per iteration. A bit faster than the gradient method with backtracking line search.



Exact Line Search VS. Backtracking Line Search with a Problem in R^{100}

•These experiments, done by the book authors, show that the effect of the backtracking parameters on the convergence is not large.
•Experiment 1: (effect of the choice of α): Fix $\beta = 0.5$, and vary α. This experiment suggests that the gradient method works better with fairly large α, in the range $(0.2, 0.5)$.
•Experiment 2: (effect of the choice of β): Fix $\alpha = 0.1$, and vary β. This experiment suggests that $\beta \approx 0.5$ is a good choice.

Advantages – Disadvantages – Limitations (Gradient Descent)

To summarize here:

- The gradient descent often shows linear convergence., i.e. error converges to zero as a geometric series.
- Choice of the two parameters α, β (backtracking parameters) has a noticeable effect on convergence, but not dramatic.
- Exact line search sometimes improves the convergence of the gradient method, slightly. However implementation is troublesome.
- Convergence rate depends on the condition number of the Hessian, and this is the main disadvantage.
- The main advantage of gradient method is its simplicity.

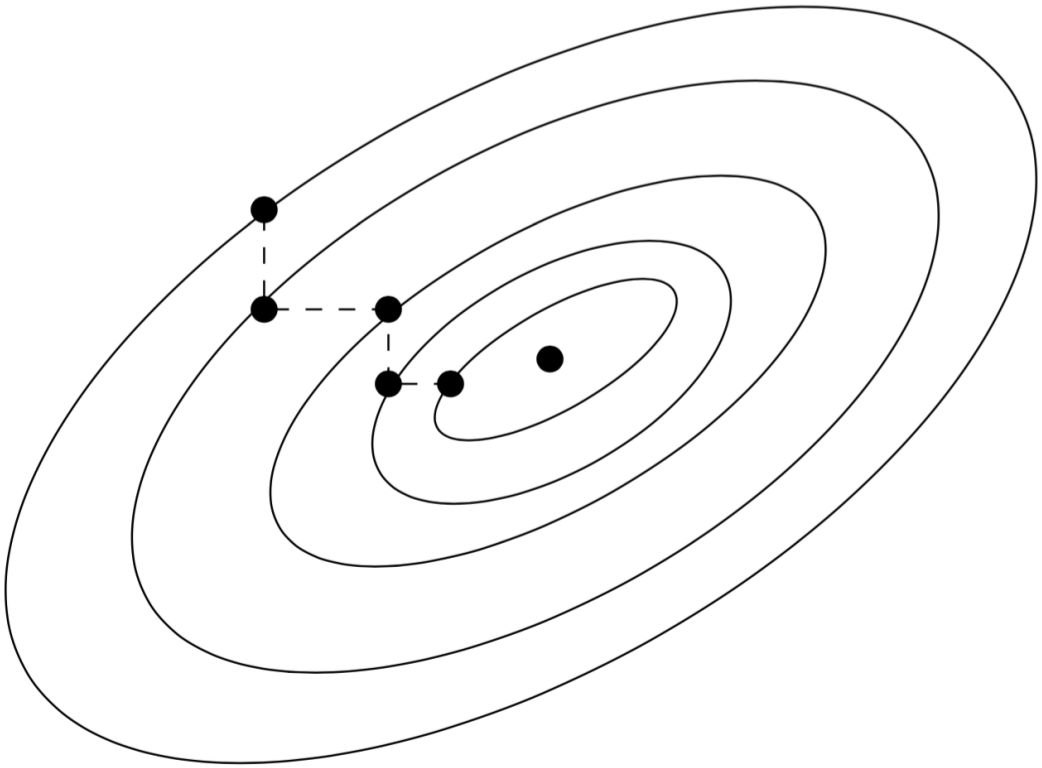
Advantages – Disadvantages – Limitations (Steepest Descent)

- For steepest descent, choice of norm is critical. When Euclidian norm is used the algorithm coincides with the gradient descent method. When l_1 norm is chosen, the algorithm is called Coordinate Descent.
- The idea is to descend along each coordinate direction iteratively.

Let $e_1, ..., e_m$ denote the unit vectors along coordinates $1, ..., m$.
Choose initial x_0;
repeat
  For $i = 1, ..., m$: ;
    Find x_{k+1} along e_j that minimizes $f(x_k)$. Or find x_{k+1} along e_j using line search to reduce $f(x_k)$ sufficiently.
until convergence;

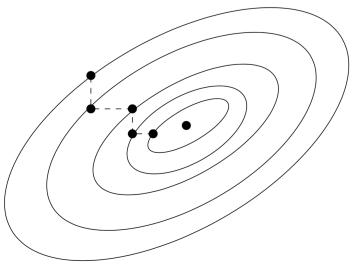
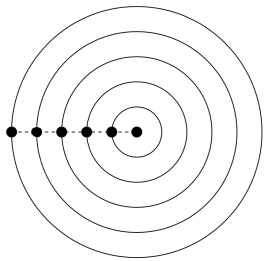
Advantages – Disadvantages – Limitations (Steepest Descent)

Coordinate descent can be slow. It can also iterate around the minimum, never approaching it.



Advantages – Disadvantages – Limitations (Steepest Descent)

- Steepest descent method with the quadratic P-norm $\|\cdot\|_P$ can be thought of as the gradient method applied to the problem after the change of coordinates $\overline{x} = P^{\frac{1}{2}}x$.
- In the case of steepest descent with quadratic P-norm, choice of P is important.
- For example: $P = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix}$ Choice of P helps to transform the problem:

from  to 

Advantages – Disadvantages – Limitations (Steepest Descent)

Gradient method works well when the condition numbers are moderate, and works poorly when the condition numbers are large.

- When we change the coordinates, as $\overline{x} = P^{\frac{1}{2}}x$, the function is moderately conditioned, the steepest descent method will work well.
- P should be chosen so that f, transformed by $P^{-\frac{1}{2}}$ to $\tilde{f}$, is well conditioned.
- If approximation $\hat{H}$ of the Hessian at the optimal point $\hat{H}(x^*)$ were known, $P = \hat{H}$ would be a good choice, since Hessian at the optimum:

$$\hat{H}^{-\frac{1}{2}} \nabla^2 f(x^*) \hat{H}^{-\frac{1}{2}} \approx I$$

is likely to have a low condition number.

Advantages – Disadvantages – Limitations (Steepest Descent)

- In summary, we can say that the steepest descent method works well in cases where we can identify a matrix P for which the transformed problem has moderate condition number.
- Comparison of two P-norms below with the previous nonquadratic problem in $\mathbb{R}^2$, using backtracking line search parameters $\alpha = 0.1$ and $\beta = 0.7$.

$$P_1 = \begin{pmatrix} 2 & 0 \\ 0 & 8 \end{pmatrix} \quad \text{and} \quad P_2 = \begin{pmatrix} 8 & 0 \\ 0 & 2 \end{pmatrix}$$

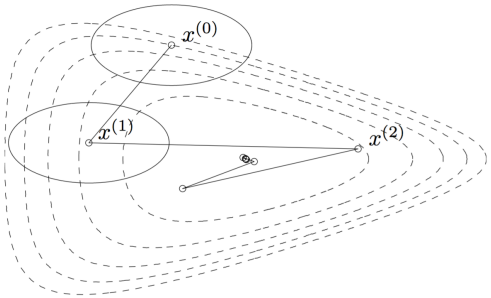


Figure: Steepest descent method, with quadratic norm $\|\cdot\|_{P_1}$. The ellipses are the boundaries of the norm balls $\{x \mid \|x - x^{(k)}\|_{P_1} \leqslant 1\}$ at $x^{(0)}$ and $x^{(1)}$.

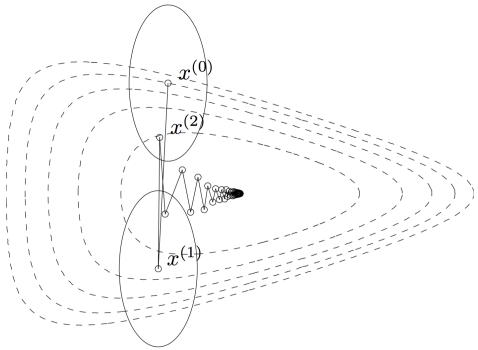
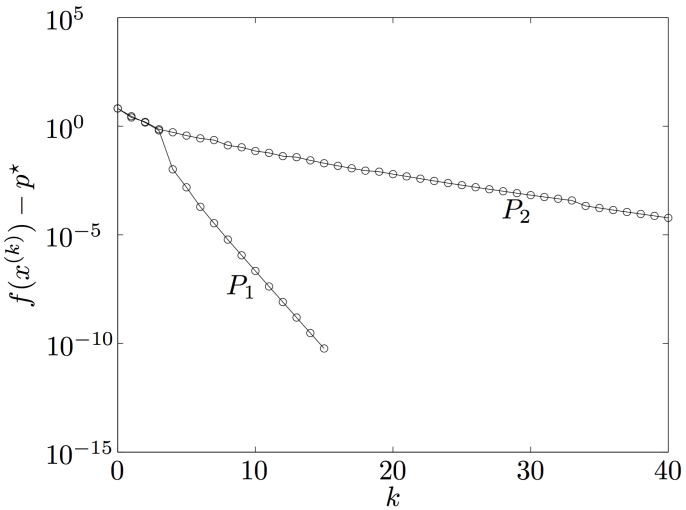


Figure: Steepest descent method, with quadratic norm $\|\cdot\|_{P_2}$.

Advantages – Disadvantages – Limitations (Steepest Descent)

- Figure shows the error vs. iteration differences of the two norms.
- With the norm $\|\cdot\|_{P_1}$, convergence is a bit more rapid than the gradient method, whereas with the norm $\|\cdot\|_{P_2}$, convergence is far slower.



Advantages – Disadvantages – Limitations (Steepest Descent)

•If we change the coordinates $\bar{x} = P_1^{\frac{1}{2}} x$, $\bar{x} = P_2^{\frac{1}{2}} x$ respectively we can get the following results in transformed coordinates.

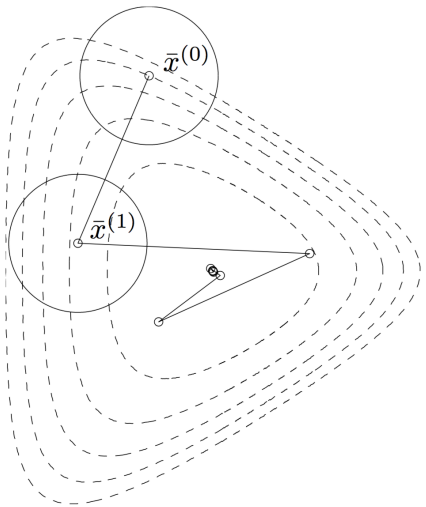


Figure: The iterates of steepest descent with norm $\|\cdot\|_{P_1}$, after the change of coordinates. This change of coordinates reduces the condition number of the sublevel sets, and so speeds up convergence.

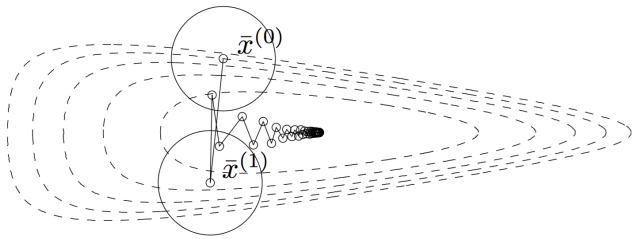
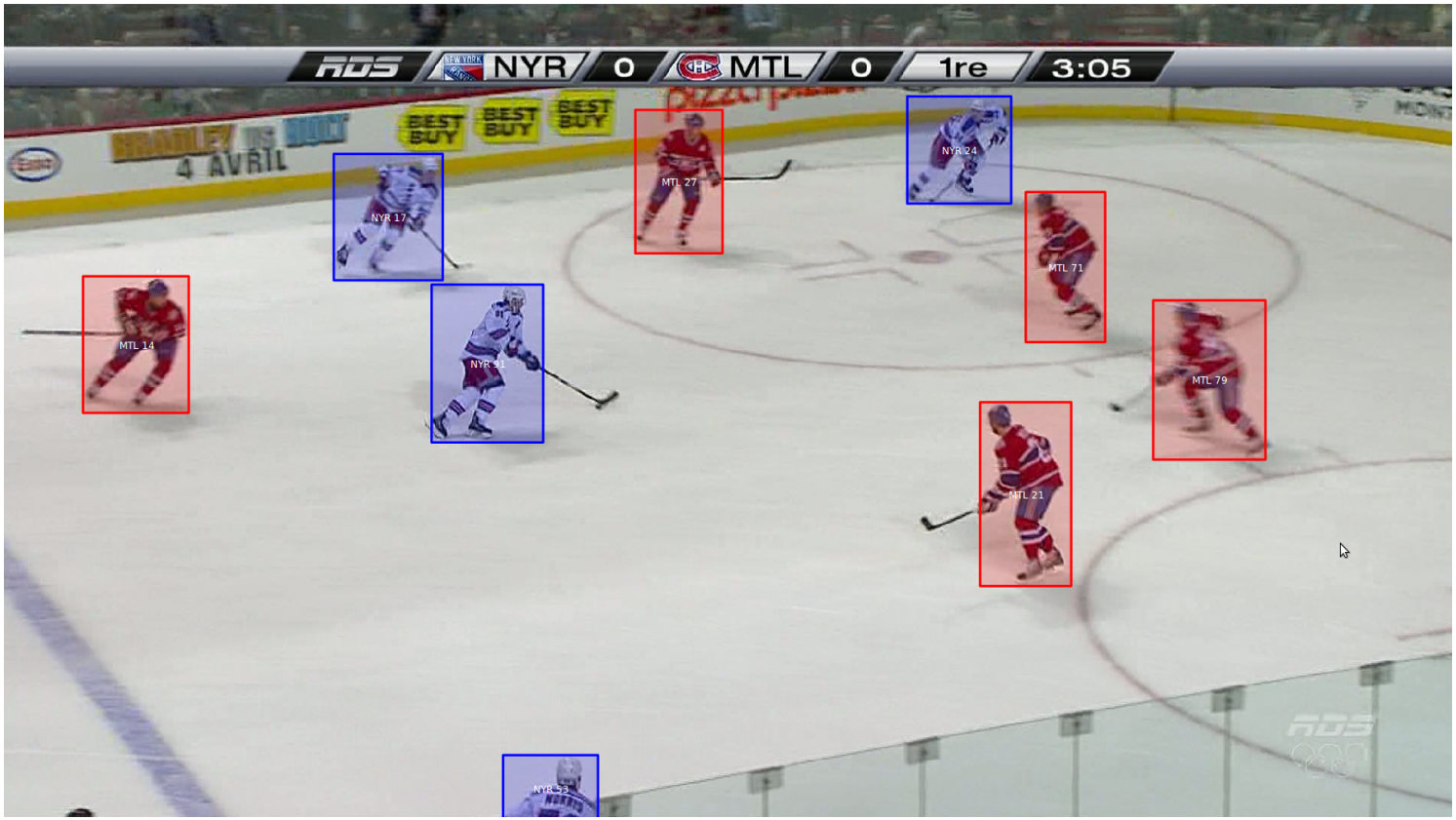


Figure: The iterates of steepest descent with norm $\|\cdot\|_{P_2}$, after the change of coordinates. This change of coordinates increases the condition number of the sublevel sets, and so slows down convergence.

Image Processing — Lucas-Kanade

Classic examples are optical flow techniques like Lucas-Kanade (VideoTracking), Horn-Schunck.



Lucas-Kanade

Goal of Lucas-Kanade

Minimize the sum of squared error between two images.

Assumption

The displacement of the image contents between two nearby instants (frames) is small and approximately constant within a neighborhood of the point $\mathbf{p}$ under consideration.

Lucas-Kanade

Optical Flow Equation (2 Dementional)

For a pixel location (x, y, t), the intensity has moved by $\Delta x, \Delta y, \Delta t$, the basic assumption can be represented as:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

Lucas-Kanade

Optical Flow Effect:

For all pixels within a window centered at p:

$$I_x(q_i)V_x + I_y(q_i)V_y = -I_t(q_i)$$

Where $i = 1, 2, 3...n$.

Abbreviations:

$$A = [I_x(q_i)^T, I_y(qi)^T]$$

$$V = [v_x, v_y]^T$$

$$b = [-I_t(q_i)]^T$$

Lucas-Kanade

Lucas-Kanade Method Abstraction:

LK method tries to solve 2×2 system:

$$A^T A V = A^T b$$

A.K.A:

$$V = (A^T A)^{-1} A^T b$$

Notice:

$V = [v_x, v_y]^T$ is variable. Which means that the system does not know the actual velocity of the system.

Lucas-Kanade

Goal of Lucas-Kanade Method:

To minimize $\|A^T V - b\|^2$.

Basic LK Derivation for Models(Stuff to be Tracked):

$$E[v_x, v_y] = \Sigma[I(x + v_x, y + v_y) - T(x, y)]^2$$

Where v_x, v_y is the hypothesized location of the model(s) to be tracked, and $T(x, y)$ model.

Lucas-Kanade

Key Step for Implementation of GD (Step 1):

Generalizing LK approach by introducing warp function W:

$$E[v_x, v_y] = \Sigma[I(W(x, y); P) - T(x, y)]^2$$

Generalizing is used to solve the problem where the constant flow of larger picture frames for a long time is a total waste of calculation power. Warp function examples are Affine and Projective.
The warping function are the convergence factor for steepest descent algorithm.

Lucas-Kanade

Key Step for Implementation of GD (Step 2):

The key to the derivation is Taylor series approximation:

$$I(W(x,y);P+\Delta P) \approx I(W([x,y];P)) + \nabla I \frac{\partial W}{\partial P} \Delta P$$

- The approximation equation is actually the abstract of the basic assumption of optical flow described in the slides before.
- Derivation of this equation can be discussed in forum (Too long for slides).

Lucas-Kanade

Some Explainations:

- Gradient image ∇I
- Image error $I_E = T(x, y) - I(W[x, y]; P)$
- Jacobian matrix $\frac{\partial W}{\partial P}$
- Steepest image $I_S = \nabla I \frac{\partial W}{\partial P}$
- Hessian Matrix $\Sigma (\nabla I \frac{\partial W}{\partial P})^T (\nabla I \frac{\partial W}{\partial P})$
- Iteration step $\Delta P = \Sigma I_S^T I_E$

Lucas-Kanade

Algorithms:

- Warp image and get $I(W[x, y]; P)$;
- Get image error I_E;
- Warp gradient image ∇I;
- Evaluate Jacobian;
- Compute steepest descent image $I_S = \nabla I \frac{\partial W}{\partial P}$;
- Compute Hessian matrix $\Sigma I_S^T I_S$;
- Get warping step $\Delta P = I_S I_E$;
- Update warping parameter $P = P + \Delta P$;
- Repeat until ΔP is negligible.

APPLICATIONS – MACHINE LEARNING

Generalized Utilization of Convex: Delta Rule

- The delta rule is derived by attempting to minimize the error in the output of the neural network through gradient descent.
- Gradient Descent optimization is the most basic principle for training neurons even with different activation functions.
- Delta rule, can also be modified, if possible, with steepest descent method.

APPLICATIONS – MACHINE LEARNING

Delta Rule:

$$\Delta w_{ji} = \alpha(t_j - y_j)g'(h_j)x_j$$

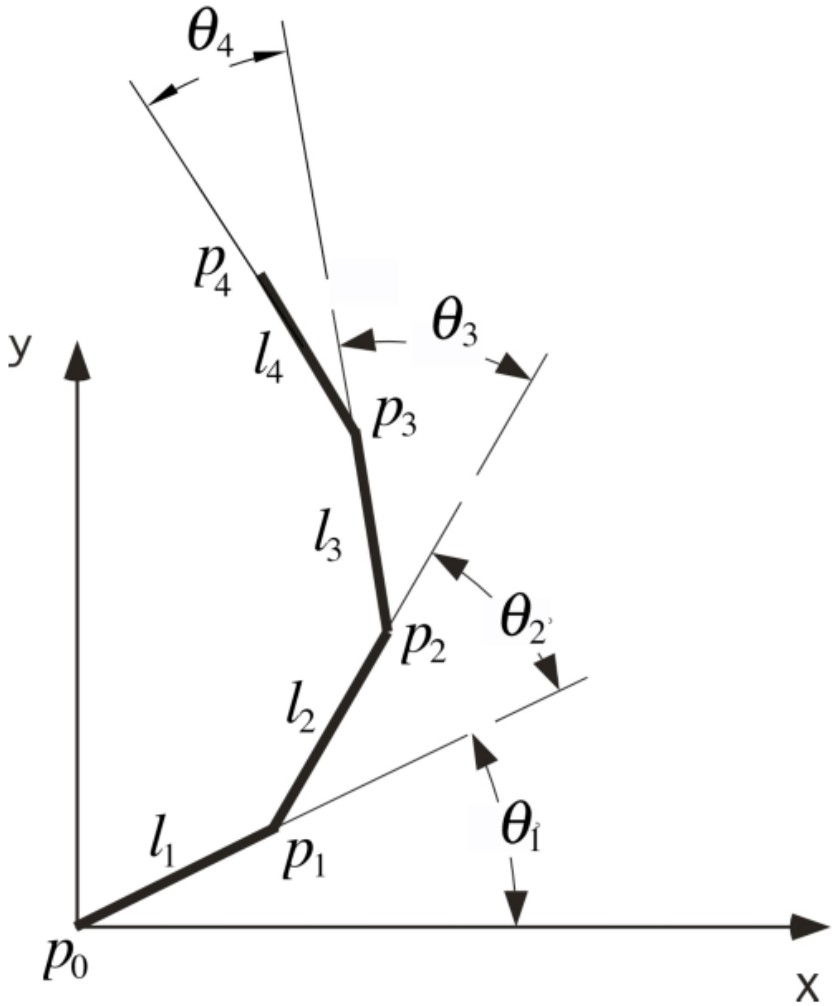
Where α is the learning rate, $g(x)$ is the neuron's activation function. t_j and y_j is the target and actual output of the neuron. h_j is the weighted sum of the neuron's inputs. And x_i is the i_{th} input.

The above equation holds the following:

$$h_j = \Sigma x_i w_{ji}$$

$$y_j = g(h_j)$$

APPLICATIONS – INVERSE KINEMATICS



APPLICATIONS – INVERSE KINEMATICS

Goal of Inverse Kinematics

Given a position in the space, calculate a way for a robot hand to reach a place.

Problem Abstract:

$$\vec{e} = R_1 T_1 R_2 T_2 R_3 T_3 R_4 T_4 \vec{e_0}$$

Where T_i is a series of translation transformation and R_i is a series of rotation translation.

APPLICATIONS – INVERSE KINEMATICS

Abstraction for Convex Optimization:

$$\Delta\vec{\theta} = \alpha J^{\top}\vec{e}$$

.

The target for the optimization is to achieve $|\vec{e_p} - \vec{e_t}| = 0$, where $\vec{e_p}$ is th original position of the tip of the robotic arm and $\vec{e_t}$ is the target position. J is the jacobian matrix in terms of $\vec{\theta}$, which is the vector of all the spatial angles of all joints. α is the convergence rate and $\vec{e}$ is the position derivation (step size).

APPLICATIONS – INVERSE KINEMATICS

About Inverse Kinematics

- Jacobian transpose is the implementation of gradient descent in the real physical world.
- It can actually achieve near linear solution for robotic arms with a fast convergence rate.

References

- Boyd, S. and Vandenberghe, L. (2004).
  Convex optimization.
  Cambridge university press.
- Buss, S. R. (2004).
  Introduction to inverse kinematics with jacobian transpose, pseudoinverse and damped least squares methods.
  IEEE Journal of Robotics and Automation, 17(1-19):16.
- Kheng, L. W.
  Nus - soc - cs 5240 lecture notes.
  http://www.comp.nus.edu.sg/~cs5240/index.html.
  Accessed: 2016-01-15.
- Lucas, B. D., Kanade, T., et al. (1981).
  An iterative image registration technique with an application to stereo vision.
  In IJCAI, volume 81, pages 674–679.
- Stone, G. O. (1986).
  An analysis of the delta rule and the learning of statistical associations.
  Parallel distributed processing: Explorations in the microstructure of cognition, 1:444–459.