Gradient Methods I: Introduction

Ruoyu Sun

Questions for the Last Lectures

- **Q1:** Why do we study optimality condition first?
- **Q2:** We learned optimality conditions for _____ problems last time.
- **Q3:** Have we learned any sufficient condition for global optimality last time?
- **Q4:** Judge: If there is a unique stationary point, then it is a global minimum or a global maximum.

Questions for the Last Lectures

- **Q1:** Why should we study optimality condition first?
  What other benefits?
  Answer: Reduce the range of search from the whole space to the solution set of the optimality conditions.
  Foundation for algorithm design.

- **Q2:** We learned optimality conditions for unconstrained (differentiable) problems last time.

- **Q3:** Have we learned any sufficient condition for global optimality last time?
  Yes. If the function is convex, and a point is stationary, then it is globally optimal.

- **Q4:** Judge: If there is a unique local minimum, then it is a global minimum. or a global maximum.
  False. Simplest counter-example: $x^3 = 0$.

Summary of the Last Lectures

- Necessary condition

$$\nabla f(x^*) = 0, \quad \text{(first-order condition)},$$
$$\nabla^2 f(x^*) \succeq 0, \quad \text{(second-order condition)}.$$

- Sufficient condition

$$\nabla f(x^*) = 0, \quad \text{(first-order condition)},$$
$$\nabla^2 f(x^*) \succ 0, \quad \text{(second-order condition)}.$$

- How to use optimality condition to directly find (local) minima, for simple problems
- Existence of global-min: coercive; bounded level set
- Convexity implies “every local-min is global-min”

Today

- Starting from today: algorithms for unconstrained optimization
- Introduction of Gradient Descent method
- After today's course, you will be able to
  - explain to your friends what is gradient descent (GD)

    Provide 3-4 ways to introduce GD
  - list different forms of iterative descent methods

    understand pros and cons of each form
  - **Advanced**: understand why applying GD to Deep Learning and AI is a remarkable achievement (worth Turing award)

Outline

1st Derivation: Best Direction of Hill Climbing


Figure: Hill Climbing[1]

- Which route is fast for climbing? (if foggy)
- **Greedy approach** (conceptual)
  - climb 1m along various directions;
  - pick the one with the biggest increase in height

Gradient Descent

- Issue: cannot check all directions
- **One solution**: instead of 1m, if climb ϵ meter along each direction, the best direction is approximately gradient
- **Gradient ascent**: when maximizing a function f, update iterates as

$$\mathbf{x} \leftarrow \mathbf{x} + \alpha \nabla f(\mathbf{x}).$$

- **Gradient descent**: when minimizing a function f, update iterates as

$$\mathbf{x} \leftarrow \mathbf{x} - \alpha \nabla f(\mathbf{x}).$$

Gradient Descent: Visualization

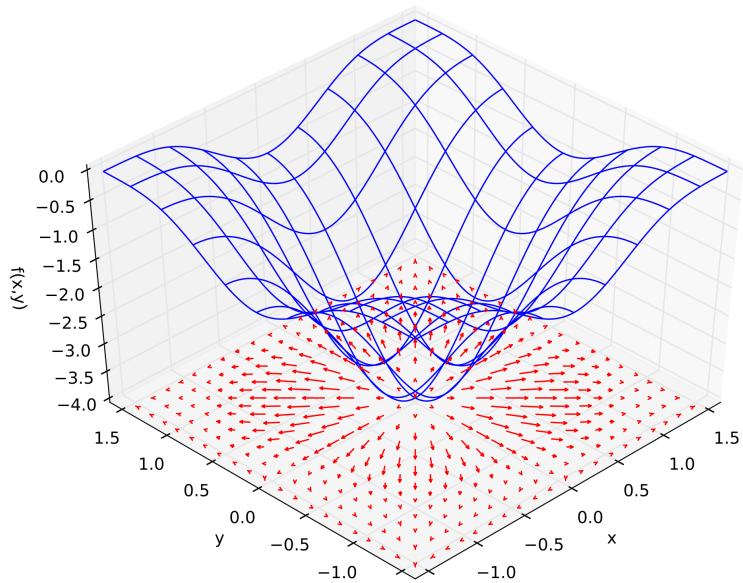


Figure: The gradients of a function (Wikipedia: Gradient)

2nd Interpretation: quadratic approximation

Minimizing a quadratic function is simple: closed-form solution.

Minimizing a general function can be hard.

Idea: successive quadratic approximation

- More general idea: reducing to easier problem
- GD performs quadratic approximation around $\mathbf{x}^r$

$$f(\mathbf{x}) \approx f(\mathbf{x}^r) + \langle \nabla f(\mathbf{x}^r), \mathbf{x} - \mathbf{x}^r \rangle + \frac{L}{2}(\mathbf{x} - \mathbf{x}^r)^T(\mathbf{x} - \mathbf{x}^r)$$

Minimize RHS to get

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \frac{1}{L}\nabla f(\mathbf{x}^r).$$

3rd Interpretation: fixed point algorithm

- Recall: A stationary point x^* satisfies

$$\nabla f(x^*) = 0, \quad \text{(first-order condition)}.$$

  Solving this equation is Step 1 of the method studied before.

- A simple way to derive GD: rewrite the equation as $x^* = x^* - \alpha \nabla f(x^*)$, and modify it to

$$x_{r+1} \quad = x_r - \alpha \nabla f(x_r \ \ )$$

  Example? Solving $x^5 + x + 1 = 0$ by $x_{r+1} = x_r - \alpha(x_r^5 + x_r + 1)$.

Does GD Work?

Applicable range: GD can be used to solve any unconstrained differentiable optimization problem

- Cannot be *directly* used to solve constrained problems
- Cannot be *directly* used to solve nonsmooth problems

Theoretical guarantee (informal): for minimizing an unconstrained differentiable function, GD (with proper stepsize) converges to stationary points

- See precise versions in the next lecture

Further theoretical guarantee (informal): If the problem is convex, then GD converges to global minima.

- If the problem is not convex, then GD may converge to sub-optimal local minima

Expressions on “can be used to solve”

Pay attention to the difference between

- (I) “can be used to solve xx problem”,
- (II) “can solve xx problem”
- (III) “can solve the problem to global optima”

(I) is often about **application range**:

- Simplex method can be used to solve LP (linear programming). It cannot be used to solve unconstrained optimization problem.
- GD can be used to solve unconstrained differentiable problem, but cannot be directly used to solve constrained problems like LP
  NOT defined for constrained problems!
- Exhaustive search can be used to solve any optimization problems (though NOT efficient)

(III) is about the **property of the algorithm**:

- Informally, simplex method can solve LP to global optima
- Informally, GD can solve unconstrained convex problems to global optima

(II) Is vague, as readers do not know whether it means (I) or (III), thus we often avoid such expressions.

Story on Using GD

As a senior (math student), I (R) was talking to a friend (F) doing machine learning.

He said: we are solving an optimization problem for classification.

I: great! Learned a lot of algorithms; can help you!

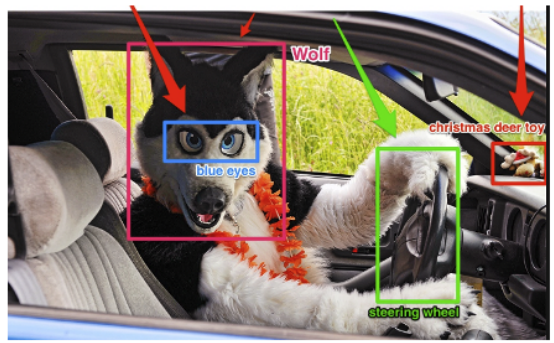
He said: OK! Here is the problem ...blablabla... (actually it is SVM, supporting vector machine)

I was struggling: shall I use GD, Newton, augmented Lagrangian, or what?

Long time later, I realize: SVM is a constrained problem, cannot be solved by GD!!!

Outline

Deep Learning and AI



Object detection

Machine translation

AI for games

Figure: Wide Application of Deep Learning

Recall: Neural Networks (Deep Learning)

Data set $\{\mathbf{a}_i, b_i\}_{i=1}^n$, $\mathbf{a}_i \in \mathbb{R}^d$, $b_i \in \mathcal{Y}$.

Define a neural-net function
$f(\theta; x) = W^L \phi(W^{L-1} \phi(W^{L-2} \cdots \phi(W^1 x) \dots))$, where $\phi : \mathbb{R} \to \mathbb{R}$ is a certain nonlinear function.

Optimization variable: $\theta = (W^L, \dots, W^1)$.

Optimization problem: $\min_\theta \sum_{i=1}^n \ell(f(\theta; \mathbf{a}_i), b_i)$, where ℓ is a certain loss function.

- Quadratic loss: $\ell(y, \hat{y}) = (y - \hat{y})^2$
- BCE loss: $\ell(y, \hat{y}) = \log(1 + \exp(-y\hat{y}))$

Can we use GD to solve this deep learning problem?
Yes!! (and indeed variants of GD are the major algorithms for AI)

- In 1980s, called backpropagation; it is just (variant of) GD

2018 Turing Award Partially Due to Using GD for AI

`https://awards.acm.org/about/2018-turing` wrote: the technical achievements of this year's Turing Laureates include ...

Geoffrey Hinton
Backpropagation: In a 1986 paper, "Learning Internal Representations by Error Propagation," co-authored with David Rumelhart and Ronald Williams, Hinton demonstrated that the backpropagation algorithm allowed neural nets to discover their own internal representations of data, making it possible to use neural nets to solve problems that had previously been thought to be beyond their reach. The backpropagation algorithm is standard in most neural networks today.

. . . . . .

Yann LeCun
Improving backpropagation algorithms: LeCun proposed an early version of the backpropagation algorithm (backprop), and gave a clean derivation of it based on variational principles. His work to speed up backpropagation algorithms included describing two simple methods to accelerate learning time.

. . . . . .

4th Interpretation: Improving Objective From Feedback

p is the basket position, w is the force, aw is the position of the ball after using force w.

Solve $\min 0.5(p - aw)^2$. ($aw = p$ means "on target")

- Start from arbitrary force w_0. The ball falls at position aw_0; error $aw_0 - p$.
- If $aw_0 - p > 0$, i.e., too far; shall reduce force w.
- If $aw_0 - p < 0$, i.e., too close; shall increase force w.
- Unify as $w_1 = w_0 - (aw_0 - p)$; here $aw_0 - p$ happens to be the gradient.

Using GD for Basketball Shooting Learning: Framework

Let's revisit this example.

Goal: To find the best shooting force in basketball.

Naive "algorithm":(exhaustive search) try all kinds of force (randomly), until finally works.

Modified framework:

- **Requirement 1**: there is an objective function

  minimize the shooting position and the basket position

- **Requirement 2**: know how the change of decision affects the outcome

  Q: will $f(w)$ increase or decrease, if we change w_r to $w_r + d_r$?

  Answer: $f(w)$ decreases if $d_r = -\nabla f(w_r)$ (will prove later)

- **Requirement 3**: can collect data necessary to compute d_r

  this example: $-\nabla f(w_r)$ depends on the current error, so need to know the current error

- **Algorithm**: adjust the decision w to improve outcome iteratively

Comparing 1st and 4th Interpretation

1st Interpretation: **Hill Climbing**

- A problem where it's easy to improve objective function
- **Nontrivial part**: pick the "best" direction negative gradient

4th interpretation: **Improving objective from feedback**

- A problem where how to improve objective is not clear
- **Nontrivial part**: there exists a "descent" direction $-\nabla f$; require collecting feedback
- This interpretation is from a higher level (modeling; environment)

Why mentioning 4th interpretation?

helps understand the evolution of deep learning and AI

Comparing 1st and 4th Interpretation

1st Interpretation: **Hill Climbing**

- A problem where it's easy to improve objective function
- **Nontrivial part**: pick the "best" direction negative gradient

4th interpretation: **Improving objective from feedback**

- A problem where how to improve objective is not clear
- **Nontrivial part**: there exists a "descent" direction $-\nabla f$; require collecting feedback
- This interpretation is from a higher level (modeling; environment)

Why mentioning 4th interpretation?
helps understand the evolution of deep learning and AI

Using GD for AI: Framework

Why can GD be used in AI?

Goal: To find a way to map images to labels "cat" or "dog"

Naive "algorithm" for decades (feature engineering): find human-designed features

Modified framework:

- **Requirement 1**: there is an objective function

  minimize the predicted label and the true label (as a function of neural-net parameter w)

- **Requirement 2**: know how the change of decision affects the outcome

  Q: will $f(w)$ increase or decrease, if we change w_r to $w_r + d_r$?

  Answer: $f(w)$ decreases if $d_r = -\nabla f(w_r)$ (will prove later)

- **Requirement 3**: can collect data necessary to compute d_r

  this example: $-\nabla f(w_r)$ depends on current error $\ell(y_i, \hat{y}_i)$, so need to know true labels y_i

- **Algorithm**: adjust w to improve $f(w)$ iteratively

Closed-loop Framework: Which Parts are Hard?

LeCun (2018 Turing award; AI guru) wrote a book: When the Machine Learns.

Most people in the area in 1970-2000:

- do NOT view learning as optimization problem (do not set up the objective that way)
- do NOT collect data (until Feifei Li, former UIUC and now Stanford, collected ImageNet around 2010)
- do NOT believe "hill climbing" can work

LeCun is very different (as undergrad from ranked-100+ univ in France):

- He is among the first (with Hinton, etc.) to advocate BackPropagation (which is GD)
- He strongly believes: with this objective function, with GD, then it must work

My comment: LeCun has an amazing faith in the power of math, optimization (and GD)

Outline

Gradient Descent is a Descent Method

If $\nabla f(\mathbf{x}) \neq 0$, then $-\nabla f(x)$ is a descent direction;
i.e., there is an interval $(0, \delta)$ of stepsizes such that

$$f(\mathbf{x} - \alpha \nabla f(\mathbf{x})) < f(\mathbf{x}),\ \forall\ \alpha \in (0, \delta).$$

Intuition: 1st order approximation $f(\mathbf{y}) \approx f(\mathbf{x}) + \nabla f(\mathbf{x})^T(\mathbf{y} - \mathbf{x})$, for $\mathbf{y}$ close enough to $\mathbf{x}$,

Thus for α small enough, we have

$$\begin{aligned} f(\mathbf{x} - \alpha \nabla f(\mathbf{x})) &\approx f(\mathbf{x}) - \langle \nabla f(\mathbf{x}), -\alpha \nabla f(\mathbf{x}) \rangle \\ &= f(\mathbf{x}) - \alpha \|\nabla f(\mathbf{x})\|^2 \\ &< f(\mathbf{x}). \end{aligned}$$

GD is a Descent Method: Proof of 1D Case (reading)

We provide a rigorous proof for $f : \mathbb{R} \to \mathbb{R}$.
Proof of 1D case: Denote $x_\alpha = x - \alpha \nabla f(x)$. By Taylor theorem (see Wikipedia `https://en.wikipedia.org/wiki/Taylor%27s_theorem`),

$$f(x_\alpha) = f(x) + f'(x)(x_\alpha - x) + h_\alpha(x)(x_\alpha - x). \tag{1}$$

where $\lim_{\alpha \to 0} h_\alpha(x) = 0$.
Plugging $x_\alpha - x = -\alpha f'(x)$ into (1), we obtain

$$\begin{aligned} f(x_\alpha) - f(x) &= -\alpha |f'(x)|^2 - h_\alpha(x) \alpha f'(x) \\ &= -\alpha (|f'(x)|^2 + h_\alpha(x) f'(x)) \end{aligned}$$

Since $|f'(x)|^2 > 0$ and $\lim_{\alpha \to 0} h_\alpha(x) = 0$, for α small enough, we have $|h_\alpha(x) f'(x)| < |f'(x)|^2$, which implies $|f'(x)|^2 + h_\alpha(x) f'(x) > 0$. Thus for α small enough, $f(x_\alpha) - f(x) < 0$. $\square$

Iterative Descent Method

More generally, if a direction $\mathbf{d}$ and $\nabla f(\mathbf{x})$ form an obtuse angle, i.e.,

$$\langle \nabla f(\mathbf{x}), \mathbf{d} \rangle < 0$$

then there is an interval $(0, \delta)$ of stepsizes such that

$$f(\mathbf{x} + \alpha \mathbf{d}) < f(\mathbf{x}), \ \forall \ \alpha \in (0, \delta).$$

Intuition: for α small enough, we have

$$\begin{aligned} f(\mathbf{x} + \alpha \mathbf{d}) &\approx f(\mathbf{x}) + \langle \nabla f(\mathbf{x}), \alpha \mathbf{d} \rangle \\ &< f(\mathbf{x}). \end{aligned}$$

Iterative Descent Method: Illustration

- Contour of function f;
- gradient direction $\nabla f(\mathbf{x})$;
- if $\mathbf{d}$ forms an obtuse angle with $\nabla f(\mathbf{x})$, then it is a descent direction

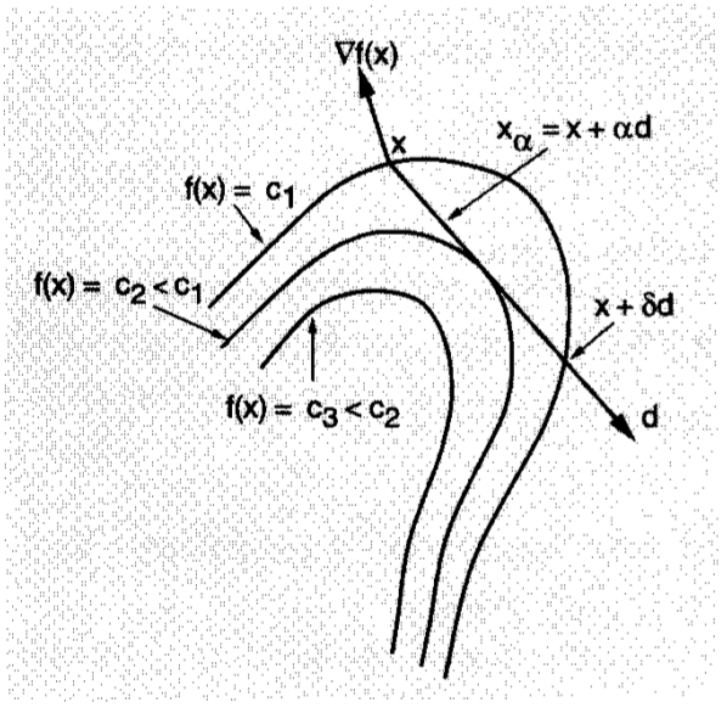


Figure: descent method (textbook Fig 1.2.3)

Iterative Descent Methods

$$\mathbf{x}^{r+1} = \mathbf{x}^r + \alpha_r \mathbf{d}^r, \ r = 0, 1, \cdots$$

Here, the direction $\mathbf{d}^r$ satisfies $\nabla f(\mathbf{x}^r)\mathbf{d}^r < 0$ when $\nabla f(\mathbf{x}^r) \neq 0$, and α^r is a positive stepsize.

Two major choices:

- **Direction choice**: how to pick $\mathbf{d}^r$?
- **Stepsize choice**: how to pick α_r?

Alternative form:

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \alpha_r \mathbf{D}^r \nabla f(\mathbf{x}^r), \ r = 0, 1, \cdots$$

where $\mathbf{D}^r$ is a positive definite matrix called scaling matrix

Choice of Update Direction

Special case I: **Steepest descent** (pick $\mathbf{D}^r = \alpha_r I$)

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \alpha_r \nabla f(\mathbf{x}^r),\ r = 0, 1, \cdots$$

- Traditionally, called steepest descent
- Nowadays, people often call it GD (especially in non-optimization world)

Special case II: **Newton's method** for strongly convex functions:

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \alpha_r \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1} \nabla f(\mathbf{x}^r),\ r = 0, 1, \cdots$$

- Here $\mathbf{D}^r = \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1}$; it is PD when f is strongly convex

There are many more variants of iterative descent methods...

Newton's Method: Derivation and Comments

- **Newton's method**:

  1. It treats the objective (locally) as a quadratic problem around $\mathbf{x}^r$

  $$f(\mathbf{x}) \approx f(\mathbf{x}^r) + \langle \nabla f(\mathbf{x}^r), \mathbf{x} - \mathbf{x}^r \rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}^r)^T \nabla^2 f(\mathbf{x}^r)(\mathbf{x} - \mathbf{x}^r)$$

  Minimizer of RHS: $\mathbf{x}^{r+1} = \mathbf{x}^r - \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1} \nabla f(\mathbf{x}^r)$,

  2. Pros: how many iterations does it take for Newton method to minimize a quadratic function f?

  3. Cons 1: not directly applicable to non-convex functions

  4. Cons 2: difficult to make it numerically stable

  5. Cons 3: each iteration is time-consuming

Zigzag behavior of GD

- GD can be very slow sometimes (zigzag behavior)
  - Practical performance?
  - implement GD for a 2-D convex quadratic problem.

    Show the convergence plot on the contour of the function

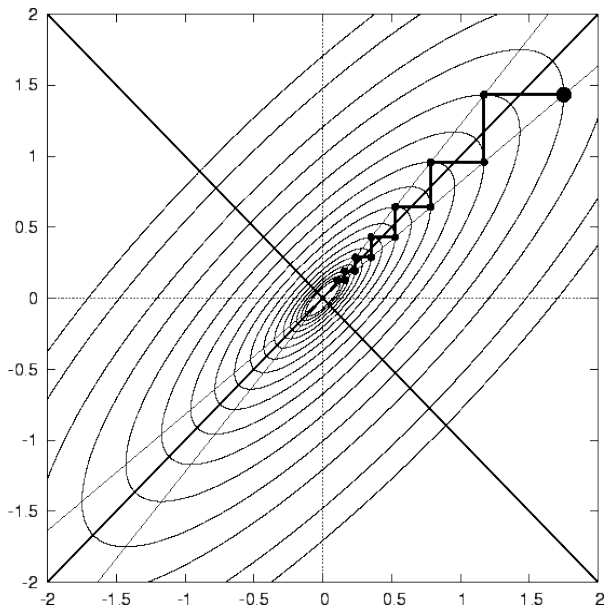


Figure: GD in Practice (Komarix.org)

First Order v.s. Second Order Methods

Both GD and Newton's methods have their pros and cons.

There is a whole spectrum of methods lying between pure 1st order methods (GD) and 2nd order methods.

- BB, BFGS, Adam, etc. Discuss later.

Deeper Understanding of Iterative Descent Methods

Algorithm design principle: it's better to let the update direction form obtuse angle with gradient direction.

Example: if f is non-convex function, and we want to use Hessian. Shall we still use

$$\mathbf{x}^{r+1} = \mathbf{x}^r - \alpha_r \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1} \nabla f(\mathbf{x}^r),\ r = 0, 1, \cdots ?$$

If $\nabla^2 f(\mathbf{x}^r)$ contains negative eigenvalues:

- then $\langle \mathbf{d}^r, \nabla f(\mathbf{x}^r) \rangle = -\langle \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1} \nabla f(\mathbf{x}^r), \nabla f(\mathbf{x}^r) \rangle$ can be positive!
- the function value may increase!

Newton's method with Hessian modification: find $\mathbf{D}^r$ that is PD and somewhat close to $\left(\nabla^2 f(\mathbf{x}^r)\right)^{-1}$.

- e.g. let $\mathbf{D}^r = \left(\nabla^2 f(\mathbf{x}^r) + \tau I\right)^{-1}$ for large enough τ
- e.g. let $\mathbf{d}^r = -c\nabla f(\mathbf{x}^r) - \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1} \nabla f(\mathbf{x}^r)$; or equivalently, let $\mathbf{D}^r = \left(\nabla^2 f(\mathbf{x}^r)\right)^{-1} + \tau I$

Check Sec. 3.4 of Nocedal & Wright's "Numerical Optimization" for more details.

Choice of Stepsize

- **Constant Stepsize**:

$$\alpha_r = \alpha$$

Comment: practically used often, but what's the constant?

- **Minimization Rule**: Pick α_r such that

$$\alpha_r = \arg\min_{\alpha \geq 0} f(\mathbf{x}^r + \alpha \mathbf{d}^r)$$

Comment: maximum reduction, but ... time-consuming

- **Limited Minimization Rule**: Pick α_r such that

$$\alpha_r = \arg\min_{\alpha \in [0,\, s]} f(\mathbf{x}^r + \alpha \mathbf{d}^r)$$

Choice of Stepsize (Cont.)

- **Diminishing Stepsize**: useful in practice, but cannot diminish too fast? Why??

$$\alpha_r \to 0, \quad \sum_{r=1}^{\infty} \alpha_r = \infty$$

Example: $\alpha_r = 1/r$; $\alpha_r = 1/r^{0.8}$; $\alpha_r = 1/r^{0.1}$;
Non-example: $\alpha_r = 1/r^2$; $\alpha_r = 1/2^r$

- **Armijo rule**: Let $\sigma \in (0, \frac{1}{2})$. Fix s as a constant, and $0 < \beta < 1$ as a constant. Keep shrinking α by $s, \beta s, \beta^2 s, \cdots$ until the following is satisfied

$$f(\mathbf{x}^r + \alpha \mathbf{d}^r) - f(\mathbf{x}^r) \leq \sigma \alpha \langle \nabla f(\mathbf{x}^r), \mathbf{d}^r \rangle$$

Comment: achieves descent, but need to test many times

The Armijo Rule

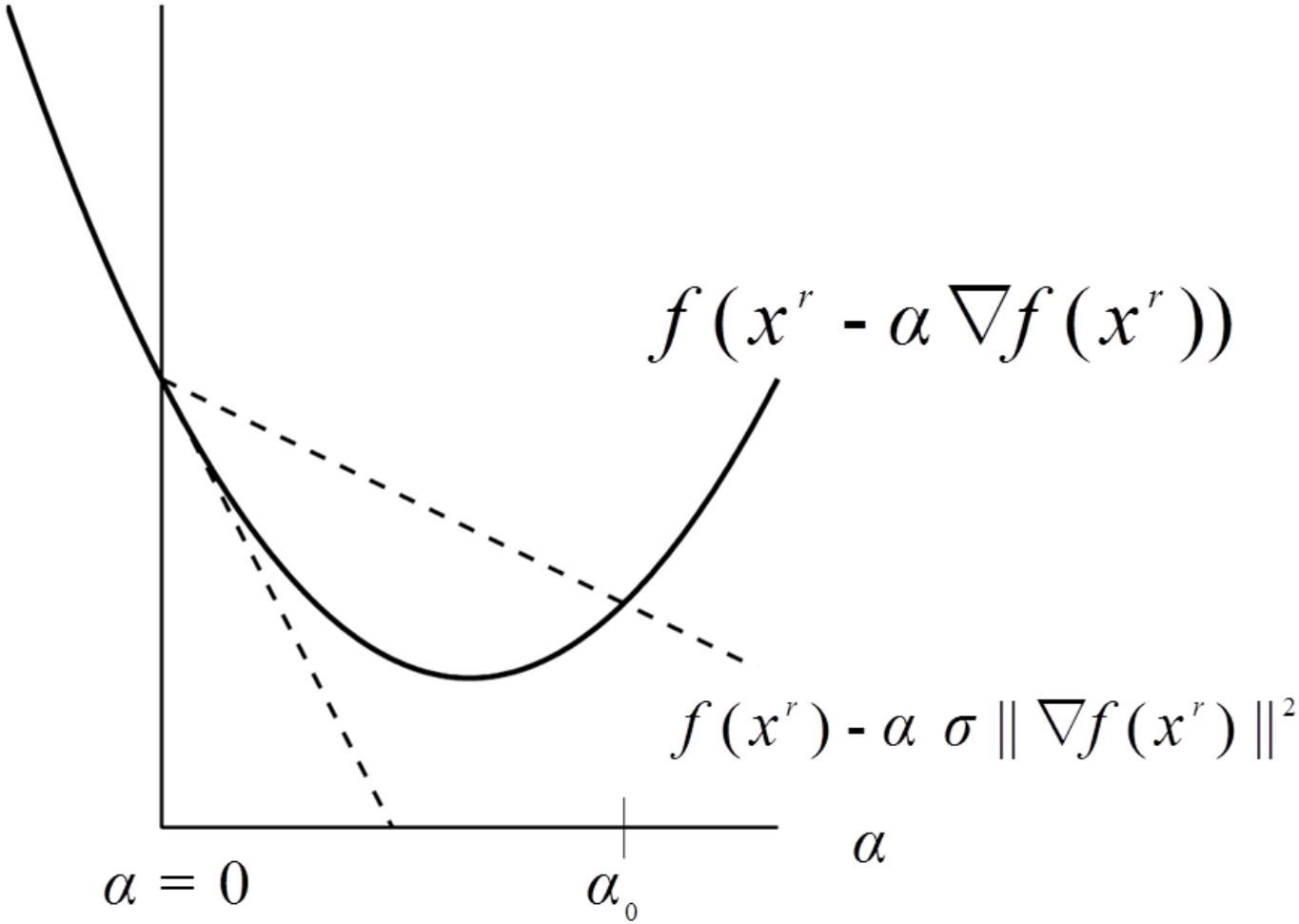


Figure: The Armijo Stepsize Selection

The Overall Strategy

- No matter what strategy we choose, there should be sufficient descent in the objective at each step
- The objective function $f(\mathbf{x})$ serves as a "potential" to guide the optimization process
- These methods are called "descent" methods, for precisely this reason
- Basically a "good" stepsize and a "good" direction is all that is required to find the (local) optimal solutions
- Next time: theoretical analysis of descent methods

Summary

- Gradient descent (also called steepest descent) has the form $x \to x - \alpha \nabla f(x)$.
  - **Application range**: any unconstrained differentiable problem
- **Three ways to motivate**
  - Iterative descent; hill climbing
  - Successive quadratic approximation (with identity 2nd order term)
  - Fixed point algorithm
- **Two key ingredients** of iterative descent methods:
  - **Direction**: $-\nabla f(x)$, $-\nabla^2 f(x)^{-1} \nabla f(x)$, general $D \nabla f(x)$
  - **Stepsize**
- Stepsize rules:
  - Pre-fixed: constant, diminishing (require sum to be infinity)
  - Line search: exact/limited minimization, Armijo rule
- GD is the powerhorse of deep learning and AI now.
  - scalable to huge problems (v.s., e.g., Newton methods)
  - Most ppl in AI did not believe GD is good before, except Hinton, LeCun, etc. Require belief in optimization paradigm; require data