Announcement: No lecture next week.

ZJU-UIUC INSTITUTE
Zhejiang University/University of Illinois at Urbana-Champaign Institute

CS450: Numerical Analysis
→ Optimization

Howard Hao Yang

Assistant Professor, ZJU-UIUC Institute

01/12/2023

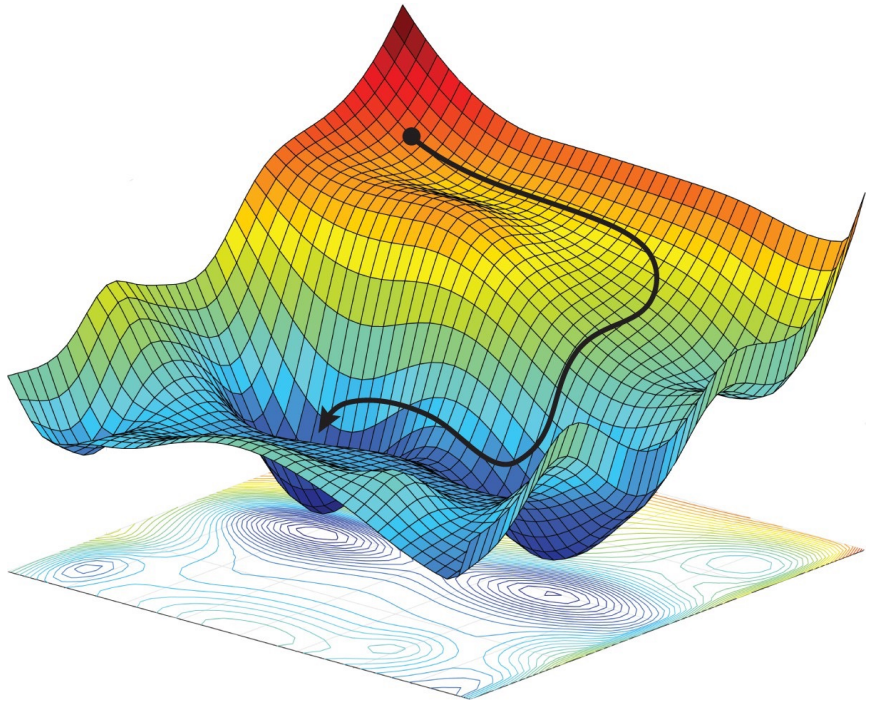
Today's Agenda

ZJU-UIUC INSTITUTE
Zhejiang University/University of Illinois at Urbana-Champaign Institute

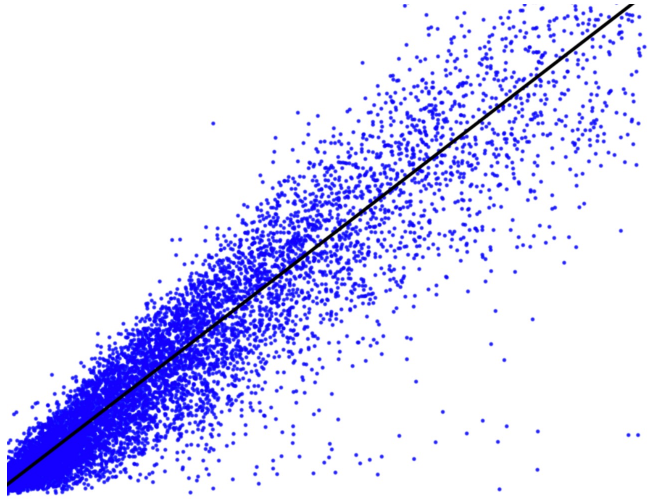
- Optimization: a variety of applications
- General forms & existence conditions
- Convexity & convex functions
- Gradient descent

Optimization

- Optimization in everyday life.
- Optimization problems underlie nearly everything we do in Machine Learning and Statistics.



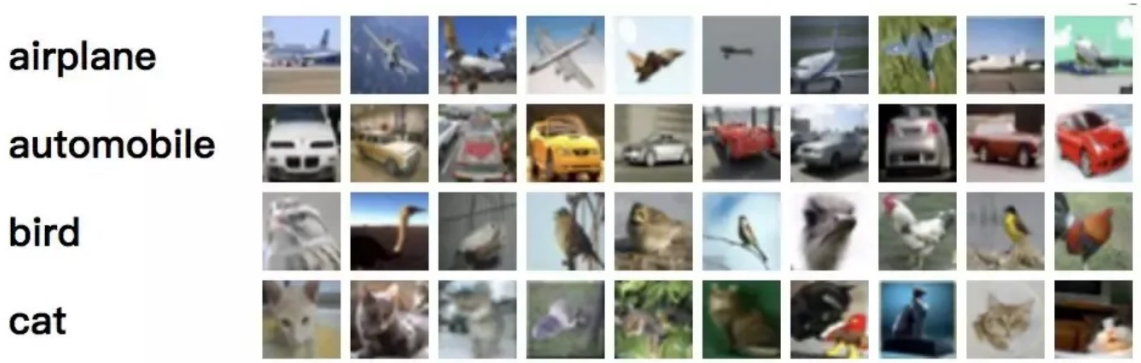
Optimization in Machine Learning



- Example 1 (Linear Regression): Given a set of input training data $(\boldsymbol{x}_1, y_1)$, $(\boldsymbol{x}_2, y_2)$, …, $(\boldsymbol{x}_N, y_N)$, find a vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$ so as to minimize the following

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} \left(y_i - \sum_{j=1}^{d} x_{ij} w_j \right)^2$$

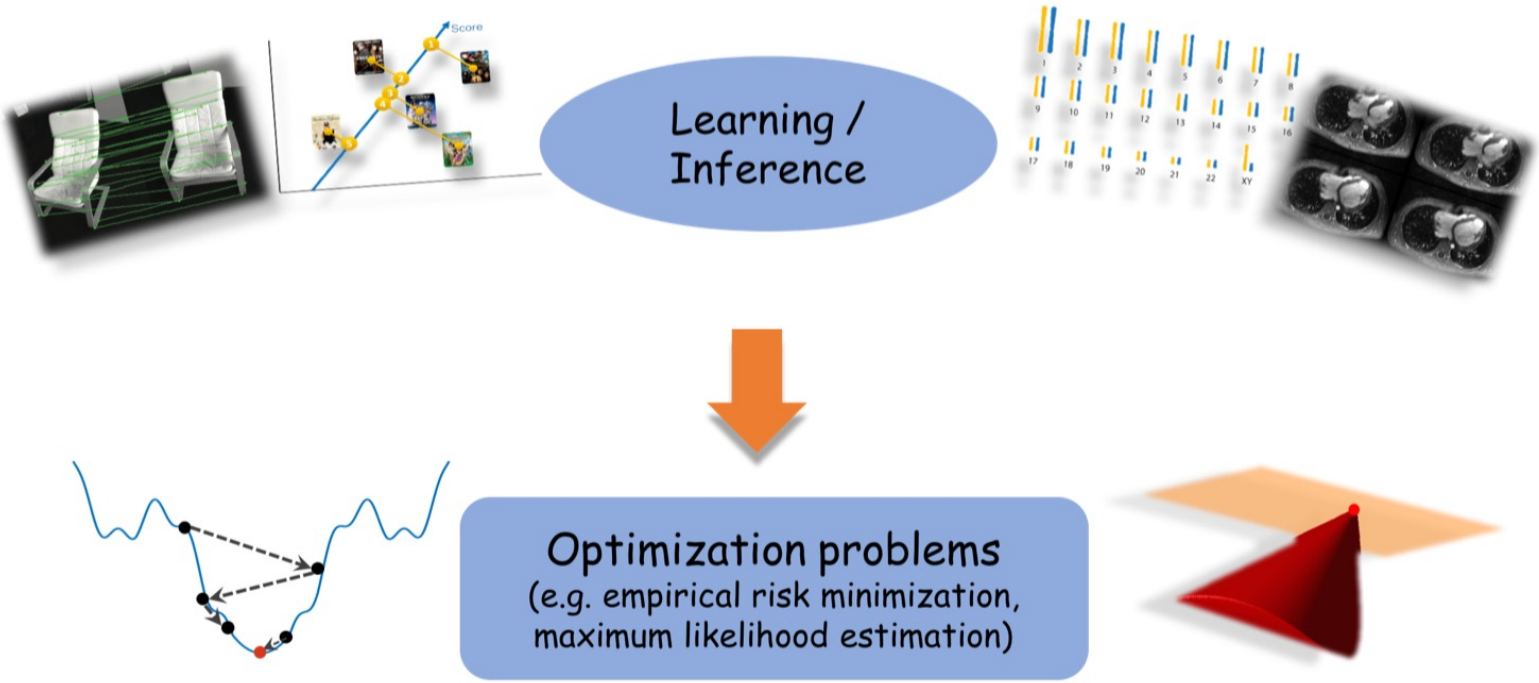
Optimization in Machine Learning (Cont'd)



- Example 2 (Classification): Given a set of input training data $(\boldsymbol{x}_1, y_1)$, $(\boldsymbol{x}_2, y_2)$, …, $(\boldsymbol{x}_N, y_N)$, where $y_i \in \{0, 1\}$. Find a vector $\boldsymbol{w} = (w_1, w_2, \ldots, w_d)$ so as to minimize the following

$$\mathrm{RSS}(\boldsymbol{w}) = \sum_{i=1}^{N} \left\{ y_i \left(\sum_{j=1}^{d} x_{ij} w_j \right) + (1 - y_i) \log \left(1 + \mathrm{e}^{\sum_{j=1}^{d} x_{ij} w_j} \right) \right\}$$

Optimization & Algorithm Design



Optimization: Problem Statement

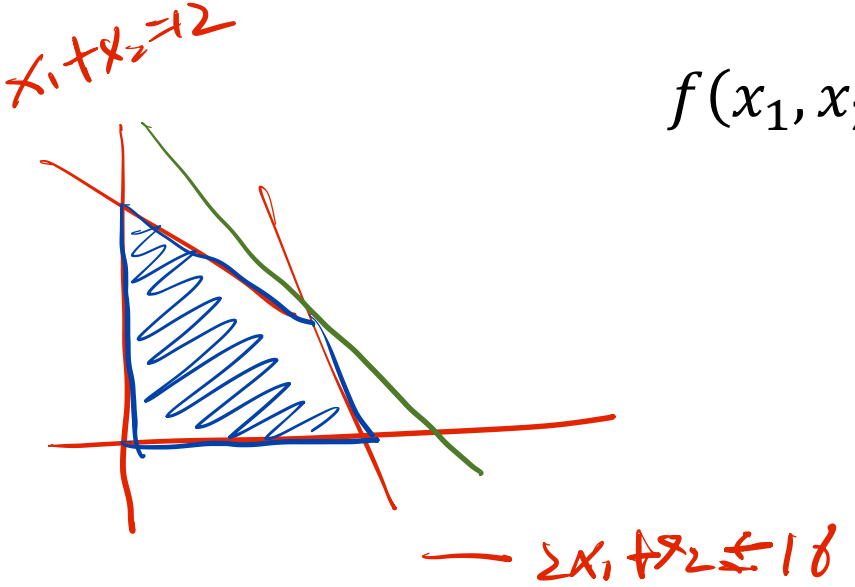
- What we have: an objective function $f: R^n \rightarrow R$
- What we want: a minimizer $\boldsymbol{x}^* \in R^n$ such that

$$f(\boldsymbol{x}^*) = \min_{\boldsymbol{x}} f(\boldsymbol{x}) \quad \text{subject to} \quad \boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0} \quad \text{and} \quad \boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}$$

  - $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$ and $\boldsymbol{h}(\boldsymbol{x}) \leq \boldsymbol{0}$ are termed constraints. They define the set of feasible points for which $\boldsymbol{x} \in S \subset R^n$.
  - If $\boldsymbol{g}$ and $\boldsymbol{h}$ are present, this is constrained optimization. Otherwise, it is unconstrained optimization.
  - If $\boldsymbol{f}$, $\boldsymbol{g}$, and $\boldsymbol{h}$ are linear, this is termed linear programming. Otherwise, it is nonlinear programming.

Examples (1)

- Consider minimizing the following
- An objective function $f: R^2 \to R$, as $f(x_1, x_2) = 40x_1 + 30x_2$
- Problem setup



$$f(x_1, x_2) = \min_{x_1, x_2} \quad 40x_1 + 30x_2$$

$$\text{s.t. } x_1 + x_2 \leq 12$$

$$2x_1 + x_2 \leq 16$$

$$x_1 \geq 0, x_2 \geq 0$$

Examples (2)

- Consider minimizing the following
- An objective function $f: R^2 \to R$, as $f(x_1, x_2) = 40x_1^2 + 30x_2^2$
- Problem setup

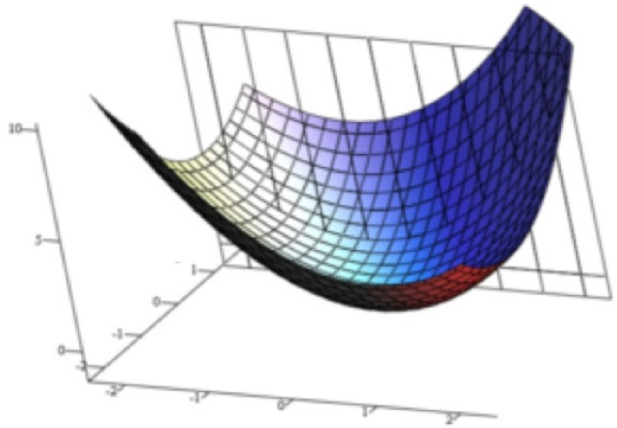
$$f(x_1, x_2) = \min_{x_1, x_2} \quad 40x_1^2 + 30x_2^2$$

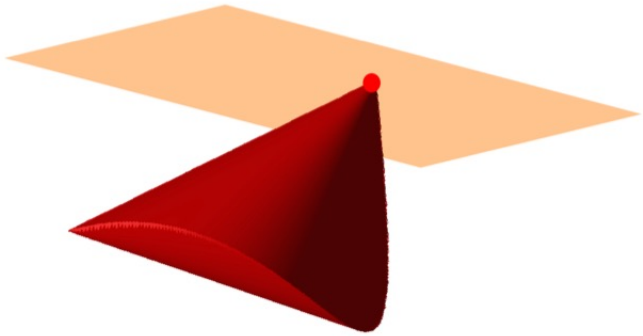
$$\text{s.t. } x_1 + x_2 \leq 12$$

$$2x_1 + x_2 \leq 16$$

$$x_1 \geq 0, x_2 \geq 0$$

Solvability/tractability





"... the great watershed in optimization isn't between linearity and nonlinearity, but convexity and nonconvexity"

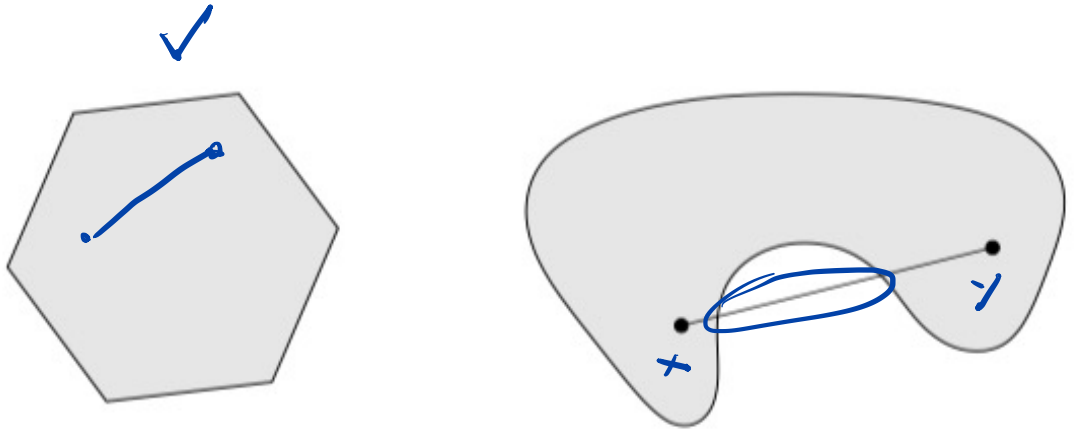
— R. Rockafellar '1993

Convexity

- A set $S \subset R^n$ is called convex if for all $\boldsymbol{x}, \boldsymbol{y} \in S$ and $0 \leq \alpha \leq 1$, there is

$$\alpha \boldsymbol{x} + (1 - \alpha)\, \boldsymbol{y} \in S$$

- Given any two points in a set, the line segment connecting these points lies within the set

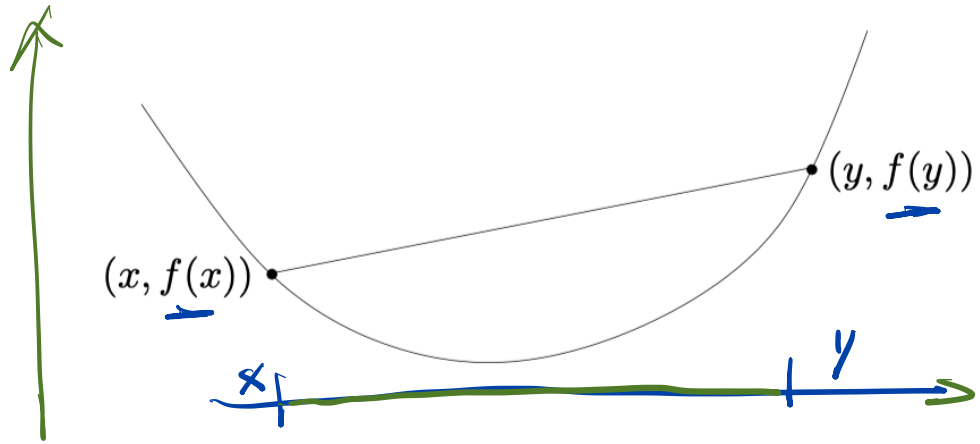


Convexity (Cont'd)

- A function $f: S \to R$ is called convex on $S \subset R^n$ if for all $\boldsymbol{x}, \boldsymbol{y} \in S$ and $0 \leq \alpha \leq 1$, there is

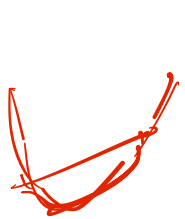
$$f(\alpha \boldsymbol{x} + (1-\alpha)\boldsymbol{y}) \leq \alpha f(\boldsymbol{x}) + (1-\alpha) f(\boldsymbol{y})$$

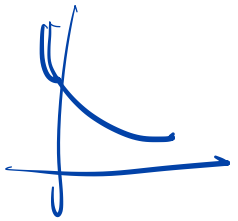
  - With '<': strictly convex
  - Q: Give an example of a convex but not strictly convex function

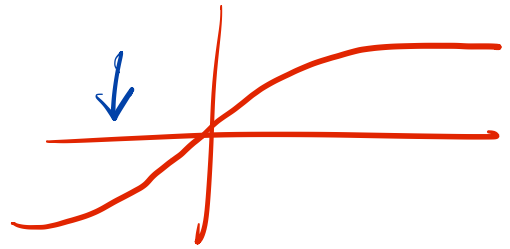


Examples/Exercise (1)

- Are the following functions convex?
  - Linear function: $f(x) = 2x + 3$
  - Quadratic function: $f(x) = x^2$
  - Exponential function: $f(x) = e^x$
  - Absolute value function: $f(x) = |x|$
  - Logarithm function: $f(x) = -\log(x)$ for $x > 0$
  - Sigmoid function: $f(x) = \frac{1}{1+e^{-x}}$
  - Sinusoidal function: $f(x) = \cos(x)$

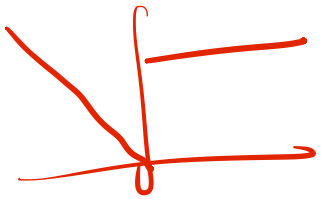






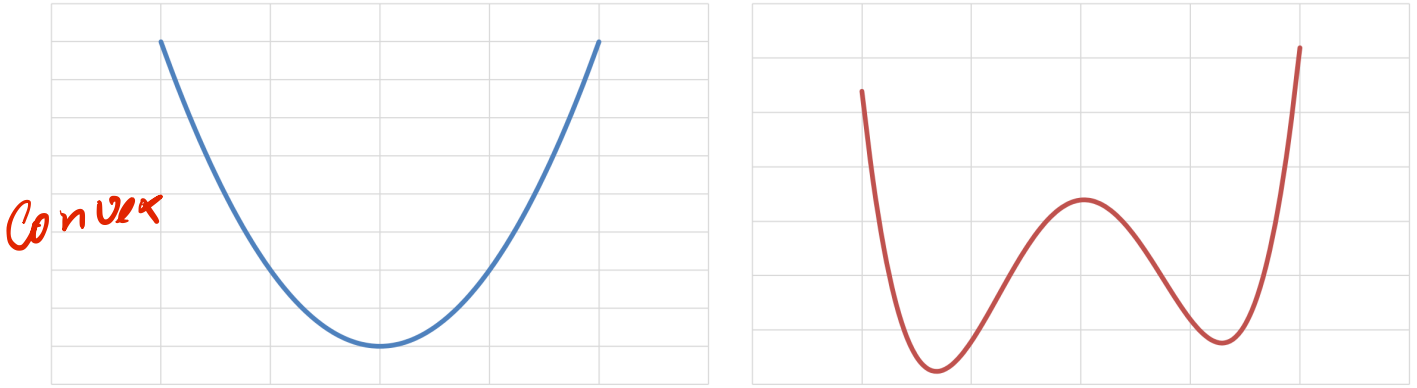
Examples/Exercise (2)

- Are the following claims correct?
  - If function $f(x)$ is continuous, it is differentiable. ✗ $f(x)=|x|$ at $x=0$
  - If function $f(x)$ is continuous, it is integrable. ✓
  - If function $f(x)$ is integrable, it is continuous. ✗
  - If function $f(x)$ is convex, it is continuous. ✓



Convexity (Cont'd)

- If a function $f: S \to R$ is convex, then
  - f is continuous at interior points (Why? What if there is a jump?)
  - A local minimum is a global minimum
- If a function $f: S \to R$ is strictly convex, then
  - A local minimum is a unique global minimum



Motivating Example

$f(x_i) = a_i x_i^2 + b_i x_i$ $\Rightarrow \frac{df}{dx_i} = 0$

- How to minimize a (high-dimension) convex function?
- An objective function $f: R^{2000} \to R$, as

$f(x_1, \cdots, x_{2000})$

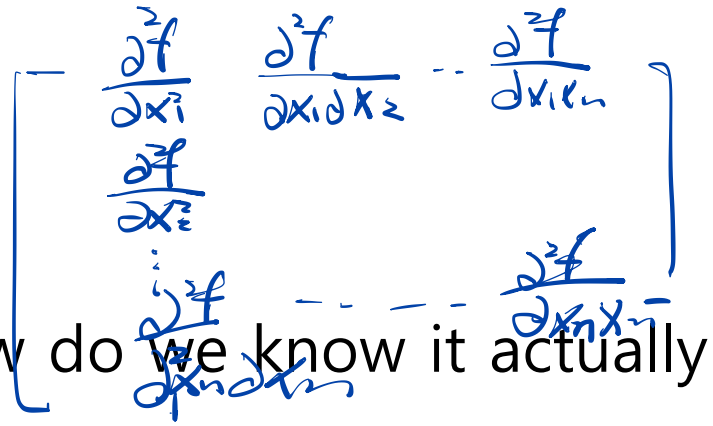
min

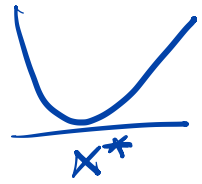
$$f(x_1, x_2, \cdots, x_{2000}) = a_1 x_1^2 + a_2 x_2^2 + \cdots + a_{2000} x_{2000}^2 + \sum_{1 \le i \ne j \le 2000} a_{i,j}\, x_i x_j$$

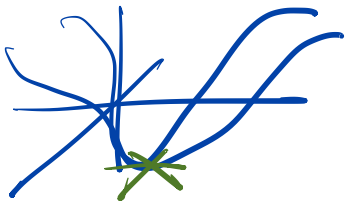
$$\begin{cases} \frac{\partial f}{\partial x_1} = 0 \\ \frac{\partial f}{\partial x_2} = 0 \\ \vdots \\ \frac{\partial f}{\partial x_{2000}} = 0 \end{cases} \Rightarrow \begin{bmatrix} & & \\ & A & \\ & & \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_{2000} \end{bmatrix} = 0$$

Optimality Conditions

- Suppose we found a candidate $\boldsymbol{x}^*$, how do we know it actually is (the) one?
- One dimension case:
  - Necessary: $f'(x^*) = 0$ (i.e., x^* is an extremal point)
  - Sufficient: $f'(x^*) = 0$ and $f''(x^*) > 0$ (implies x^* is a local minimum)
- n-dimension case:
  - Necessary: $\nabla f(\boldsymbol{x}^*) = 0$ (i.e., $\boldsymbol{x}^*$ is an extremal point)
  - Sufficient: $\nabla f(\boldsymbol{x}^*) = 0$ and $H_f(\boldsymbol{x}^*)$ is positive semidefinite (i.e., $\boldsymbol{x}^*$ is a local minimum), where $H_f(\boldsymbol{x}^*)$ denotes the Hessian matrix







Optimization: Observations

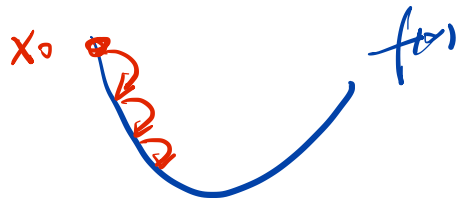
ZJU-UIUC INSTITUTE
Zhejiang University/University of Illinois at Urbana-Champaign Institute

- Can we convert the optimization problem into solving nonlinear equation systems as $\nabla f(\boldsymbol{x}) = \mathbf{0}$, and then use techniques from last week to solve it?
- Ideally, yes. Practically, not necessary:

  computational cost = iteration complexity x cost per iteration

- For large-scale problems, it calls for methods with cheap iterations
- **First-order methods**: methods that exploit only information on function values and (sub)gradients (without resorting to Hessian information)

Iterative Descent Algorithms



- Suppose we have an unconstrained optimization problem,

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \qquad \text{subject to} \quad \boldsymbol{x} \in R^n$$

- Iterative descent algorithms
  - Start with an initial point $\boldsymbol{x}_0$ and construct a sequence $\{\boldsymbol{x}_t\}$ such that

$$f(\boldsymbol{x}_{t+1}) < f(\boldsymbol{x}_t), \qquad t = 0,1, \ldots.$$

  - Which direction to go? $\boldsymbol{d}$ is called a descent direction at $\boldsymbol{x}$ if

$$\lim_{s \to 0^+} \frac{f(\boldsymbol{x} + s\boldsymbol{d}) - f(\boldsymbol{x})}{s} = \nabla f(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{d} = \|\nabla f(\boldsymbol{x})\| \cdot \|\boldsymbol{d}\| \cdot \cos(\theta)$$

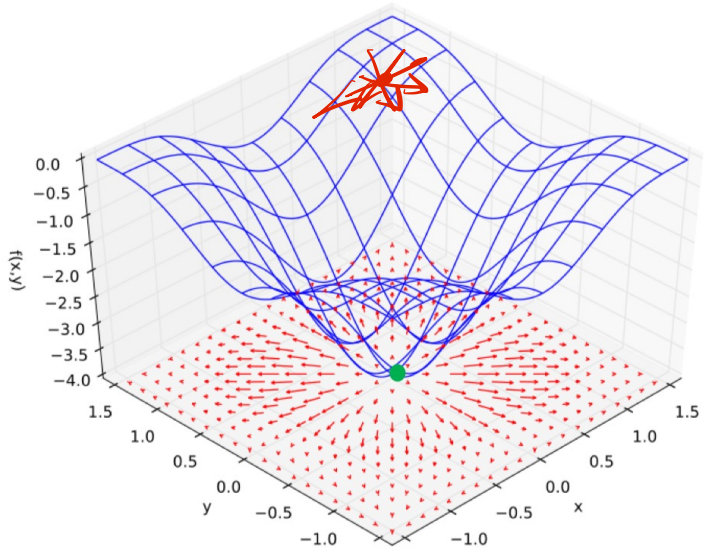
tiny progress; current position x; d

$\theta \in [0, \pi]$

$\theta = 0, \theta = \pi$

$\left(\frac{\partial f(x)}{\partial x_1}, \frac{\partial f}{\partial x_2}, \ldots, \frac{\partial f}{\partial x_n} \right)^T$

Iterative Descent Algorithms (Cont'd)



- Iterative descent algorithms
  - Start with an initial point $\boldsymbol{x}_0$
  - In each iteration, search in descent direction as follows

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t + \eta_t \boldsymbol{d}_t$$

where $\boldsymbol{d}_t$ is the descent direction, and η_t is the stepsize (How to realize algorithmically?)

Gradient Descent (GD)

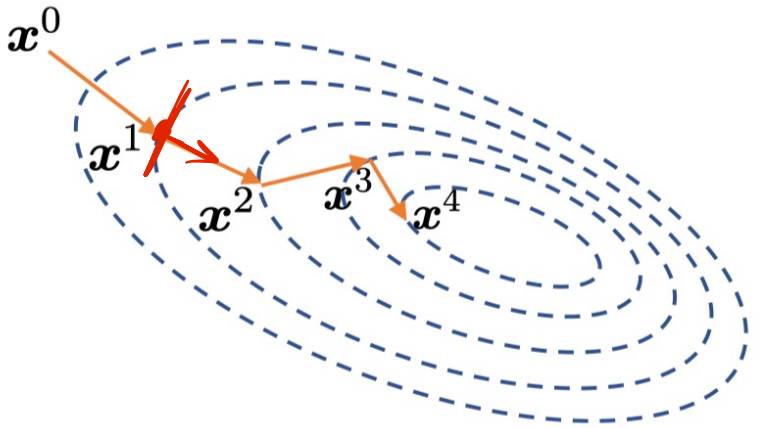
- An important example of iterative descent algorithms: Gradient descent
  - Start with an initial point $\boldsymbol{x}_0$ and update as follows

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t)$$

  - Descent direction: $\boldsymbol{d}_t = -\nabla f(\boldsymbol{x}_t)$, a.k.a., steepest descent -- why?

$$\text{argmin}_{\boldsymbol{d}:\|\boldsymbol{d}\|_2 \le 1} \nabla f(\boldsymbol{x})^{\mathrm{T}} \boldsymbol{d} = -\|\nabla f(\boldsymbol{x})\|$$

Cauchy



Illustrative Example

- Given an unconstrained optimization problem,

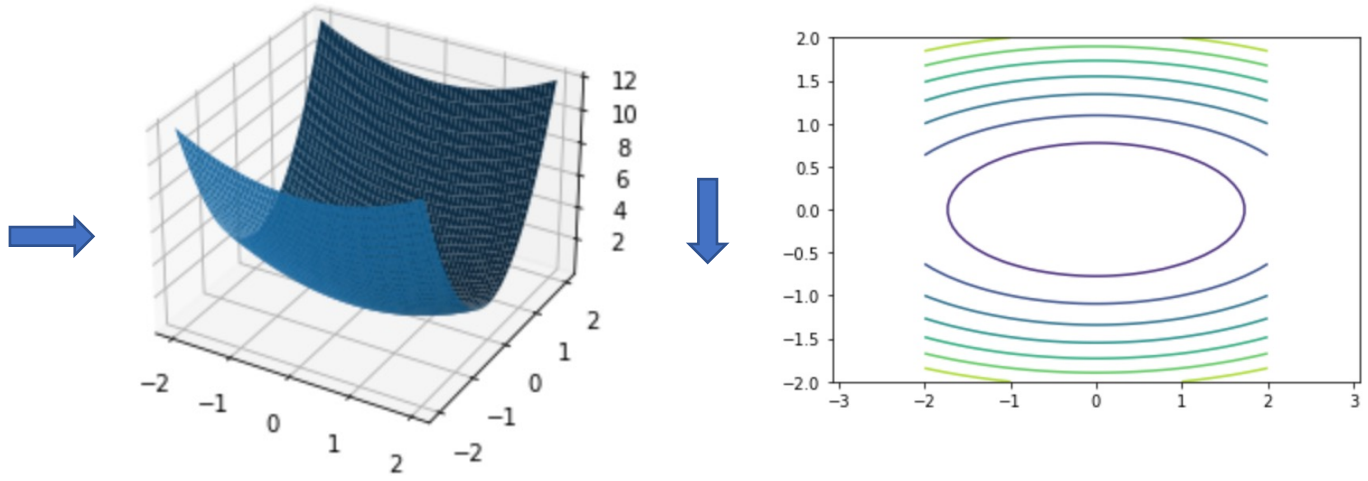
$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \boldsymbol{x}^*)^T \boldsymbol{A}(\boldsymbol{x} - \boldsymbol{x}^*)$$

```python
def f(x):
    return 0.5*x[0]**2 + 2.5*x[1]**2

def df(x):
    return np.array([x[0], 5*x[1]])
```

```python
fig = pt.figure()
ax = fig.gca(projection="3d")

xmesh, ymesh = np.mgrid[-2:2:50j,-2:2:50j]
fmesh = f(np.array([xmesh, ymesh]))
ax.plot_surface(xmesh, ymesh, fmesh)
```



Illustrative Example (Cont'd)

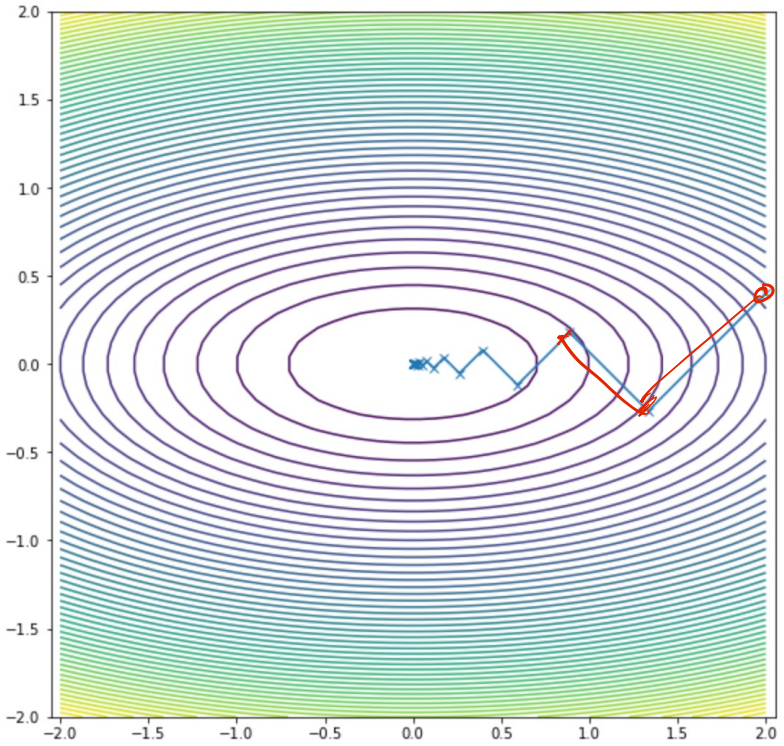
- Steepest descent

```
#clear
x = guesses[-1]
s = -df(x)

def f1d(alpha):
    return f(x + alpha*s)

alpha_opt = sopt.golden(f1d)
next_guess = x + alpha_opt * s
guesses.append(next_guess)

print(next_guess)
```




```
for i, guess in enumerate(guesses):
    print(i, la.norm(guess, 2))
```

```
0 2.039607805437114
1 1.3597385362818037
2 0.9064923598953564
3 0.6043282354184075
4 0.402885493365651
5 0.268590327422494
6 0.1790602196535349
7 0.11937347850578087
8 0.07958231961357493
9 0.05305487971599619
10 0.03536991972494859
11 0.023579946499640153
12 0.0157199643742649
13 0.01047997619145283
14 0.006986650869603477
15 0.004657767182597052
16 0.00310517817085647
17 0.0020701187313361103
18 0.001380079194276325 7
19 0.0009200527639282007
20 0.0006133685351837453
21 0.0004089123383293087
22 0.00027260823928739256
23 0.00018173881650443178
24 0.000121159217718125 8
25 8.077280684766319e-05
26 5.3848541322305716e-05
27 3.589902516077526e-05
28 2.393268515749757e-05
29 1.595512232752962e-05
30 1.0636748894671809e-05
31 7.0911654939472895e-06
32 4.727443954112351e-06
33 3.1516290931456465e-06
34 2.101086219381249e-06
35 1.4007240373068642e-06
36 9.33816096573883e-07
37 6.22544015961566e-07
```

Gradient Descent (Cont'd)

- Convergence rate:
  - The error we reduce in one-step iteration

$$\begin{aligned}\boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}^* &= \boldsymbol{\beta}_t - \eta_t\left(2\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\boldsymbol{\beta}_t - 2\boldsymbol{X}^T\boldsymbol{y}\right) - \boldsymbol{\beta}^* \\ &= (\boldsymbol{I} - 2\eta_t\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X})(\boldsymbol{\beta}_t - \boldsymbol{\beta}^*)\end{aligned}$$

  - As such, absolute value in the error is bounded by the following

$$\| \boldsymbol{\beta}_{t+1} - \boldsymbol{\beta}^* \| \leq \|\boldsymbol{I} - 2\eta_t\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\| \cdot \|\boldsymbol{\beta}_t - \boldsymbol{\beta}^*\|$$

  - Intermedia calculation gives

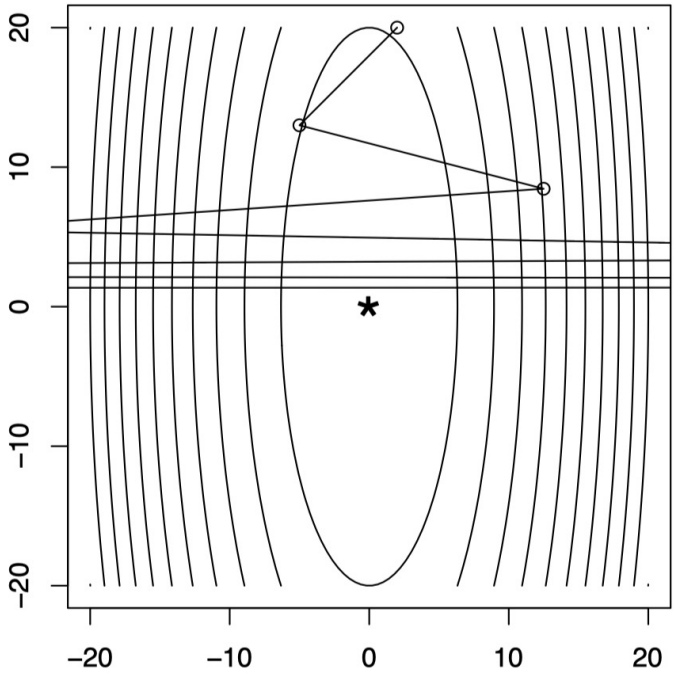
$$\left\|\boldsymbol{I} - 2\eta_t\boldsymbol{X}^{\mathrm{T}}\boldsymbol{X}\right\| \leq \max\{|1 - \eta_t\lambda_1|, |1 - \eta_t\lambda_n|\} = \frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}$$

  - Therefore, (linear convergence rate)

$$\| \boldsymbol{\beta}_t - \boldsymbol{\beta}^* \| \leq \left(\frac{\lambda_1 - \lambda_n}{\lambda_1 + \lambda_n}\right)^t \| \boldsymbol{\beta}_0 - \boldsymbol{\beta}^* \|$$

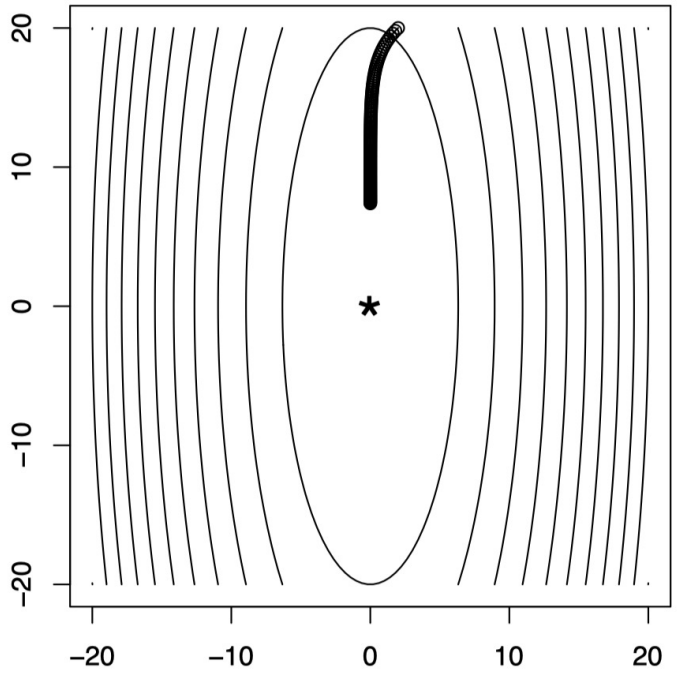
GD: How to choose stepsize?

- Simply assigning $\eta_t = \eta$ for $t = 1,2,\ldots$ might diverge if η is too large
- Example: consider $f(x_1, x_2) = (10x_1^2 + x_2^2)/2$, and use gradient descent. After 8 steps



GD: How to choose stepsize? (Cont'd)

- The convergence might be too slow if η is too small
- Same example, after 100 steps



GD: How to choose stepsize? (Cont'd)

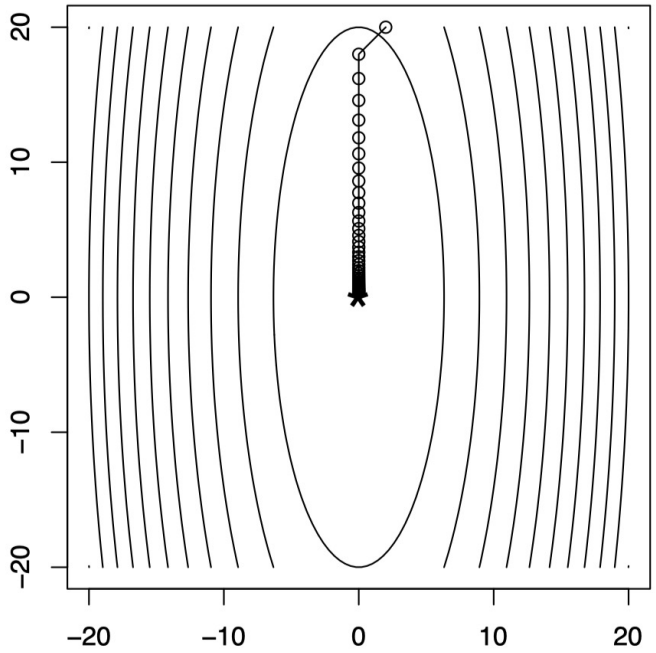
- Idea from playing golf

Strike hard when you are far away from the hole

Perform subtle improvements when you are closed

GD: How to choose stepsize? (Cont'd)

- Converges nicely when the stepsize is chosen "just right"
- Same example, after 40 steps



Backtracking Line Search

- Adaptively choosing the step size
  - Fix parameters $0 < \beta < 1$ and $0 < \alpha \leq 1/2$
  - Initialize the step size as η_0, at iteration t, if

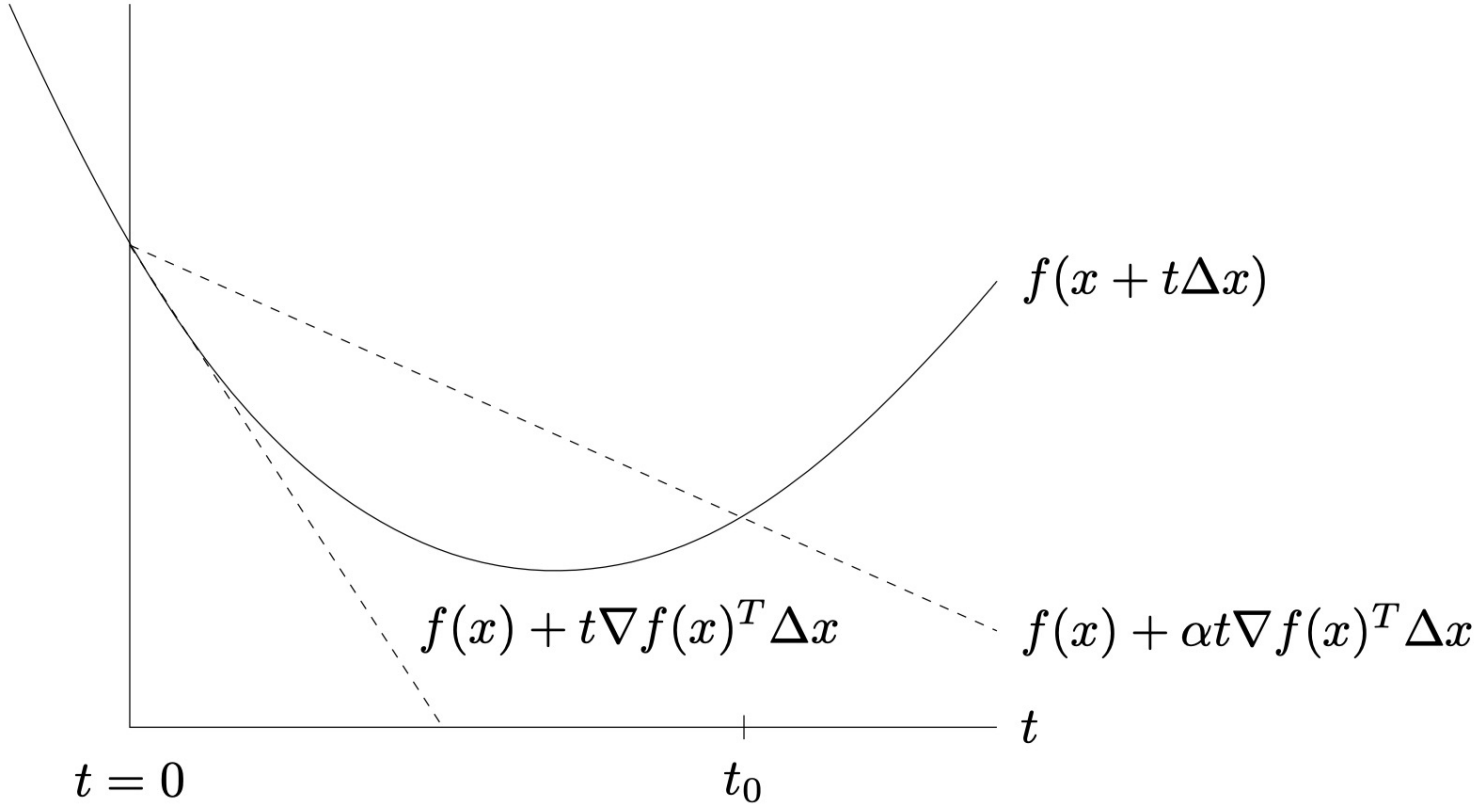
$$f\left(\boldsymbol{x} - \eta_t \nabla f(\boldsymbol{x})\right) > f(\boldsymbol{x}) - \alpha \eta_t \|\nabla f(\boldsymbol{x})\|^2$$

shrink step size as $\eta_{t+1} = \beta \cdot \eta_t$; otherwise, keep η_t unchanged and progress with GD

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - \eta_t \nabla f(\boldsymbol{x}_t)$$

Interpretation

- In the following, $\Delta x = -\nabla f(x)$



Exact Line Search

- We could also choose step to do the best we can along the direction of negative gradient, called exact line search

$$\eta_t = \min_{\eta \geq 0} f\left(\boldsymbol{x} - \eta \nabla f(\boldsymbol{x})\right)$$

- But this is not commonly used (why?)
- Approximation to exact line search is not as efficient as backtracking, and it is typically not worthy

Learning Objectives

ZJU-UIUC INSTITUTE
Zhejiang University/University of Illinois at Urbana-Champaign Institute

- Optimization is everywhere, and is particularly relevant to machine learning
- Convexity is a borderline between “easy” and “hard” optimization approaches
- Iterative descent approach for reaching optimality
- Gradient descent