Northeastern

EECE5645
Parallel Processing for Data Analytics

Lecture 8: Unconstrained Minimization & Gradient Descent

Outline

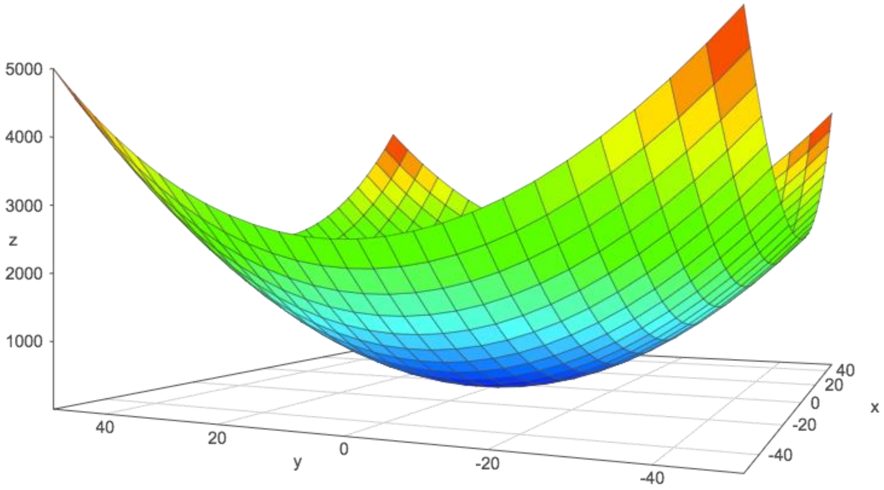
- ❑Unconstrained Optimization
- ❑Gradient Descent
- ❑Newton's Method
- ❑Parallelizing Computations

Outline

- ❑Unconstrained Optimization
- ❑Gradient Descent
- ❑Newton's Method
- ❑Parallelizing Computations

Unconstrained Minimization

$$\min_{x \in \mathbb{R}^d} f(x)$$



Where the **objective function**

$$f : \mathbb{R}^d \to \mathbb{R}$$

is:

- ❑ convex
- ❑ twice continuously differentiable (i.e., Hessian exists and is continuous)

Northeastern

Optimality Condition

Let

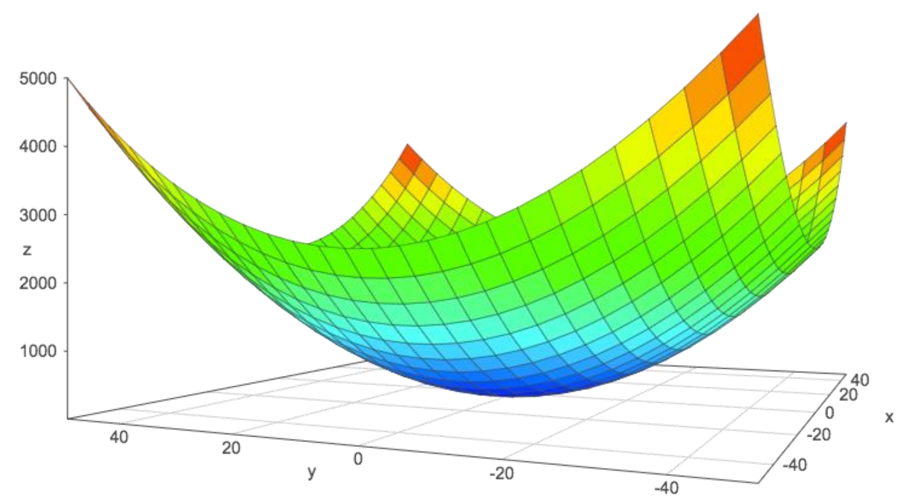
$$x^* = \arg\min_{x \in \mathbb{R}^d} f(x)$$

be an **optimal or minimum point**, and assume that this is **attained** (i.e., is in $\mathbb{R}^d$)

Then,

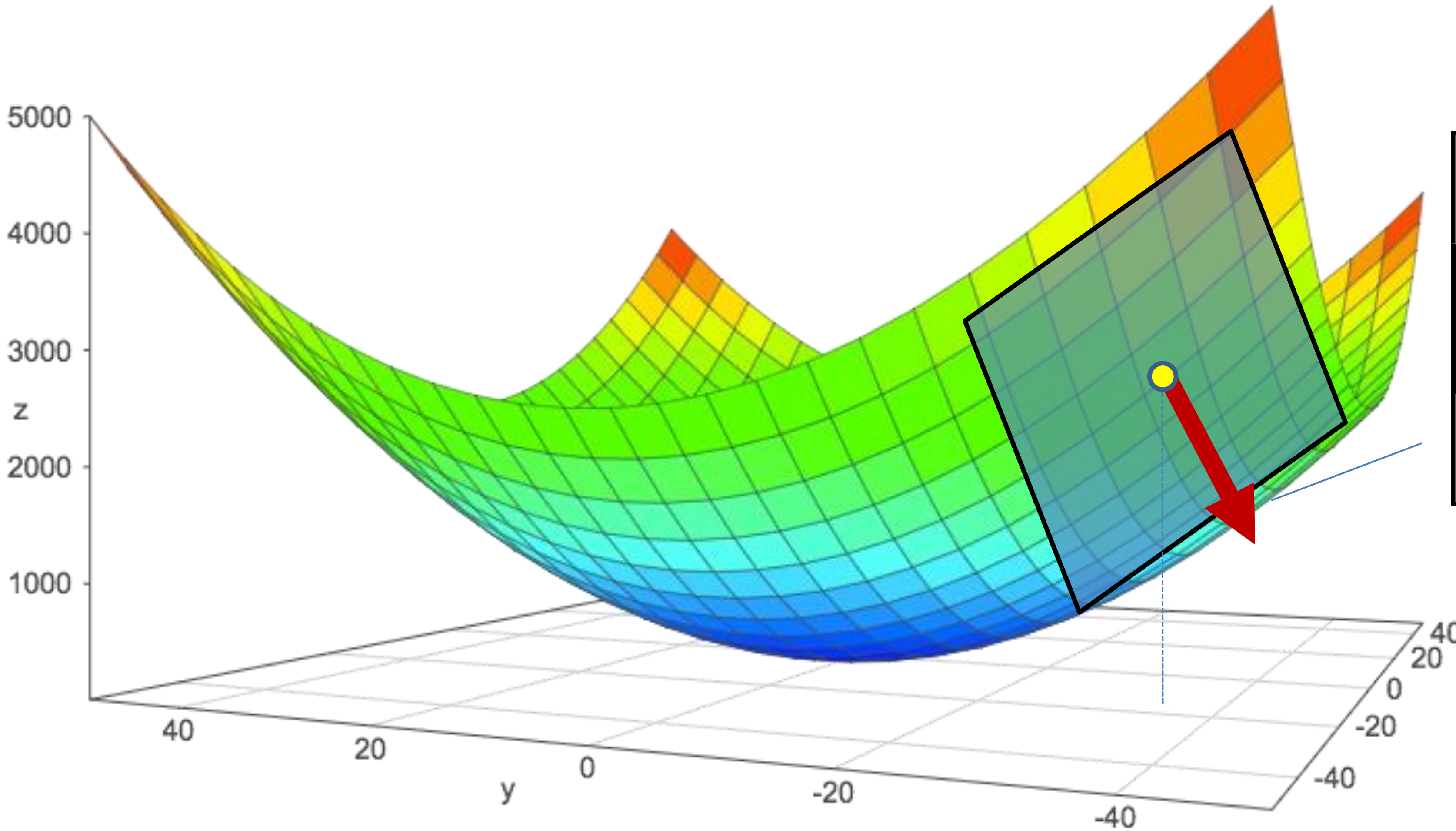
$$\nabla F(x^*) = 0$$

Certificate of optimality!!!



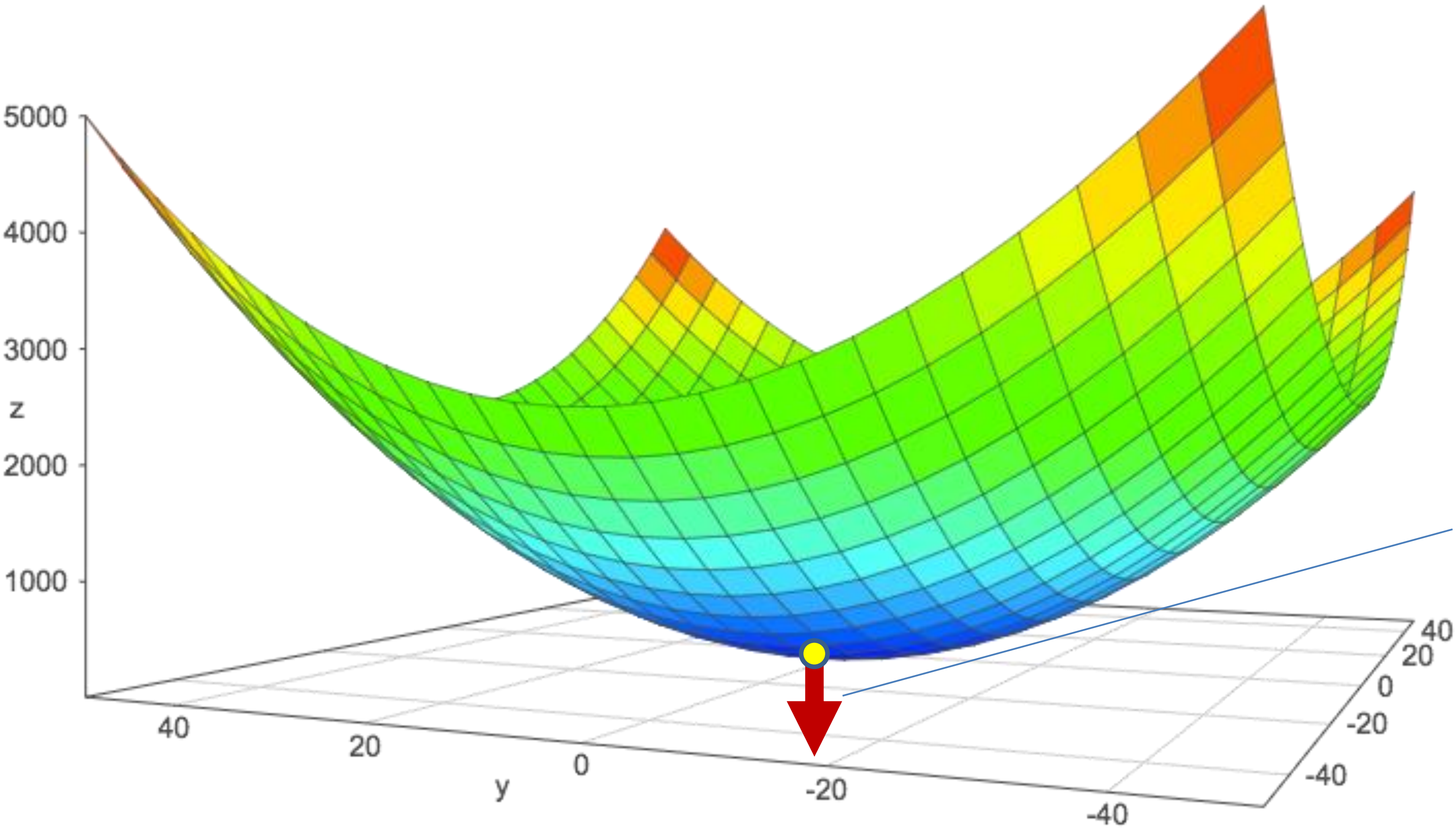
Northeastern

Optimality Condition



$$\begin{bmatrix} -1 \\ \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Optimality Condition



$$\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

Optimality Condition: Necessary and Sufficient

Finding optimal point:

$$x^* = \arg\min_{x \in \mathbb{R}^d} f(x)$$

$$\Leftrightarrow$$

Find point such that:

$$\nabla F(x^*) = 0$$

Optimality Condition: Necessary and Sufficient

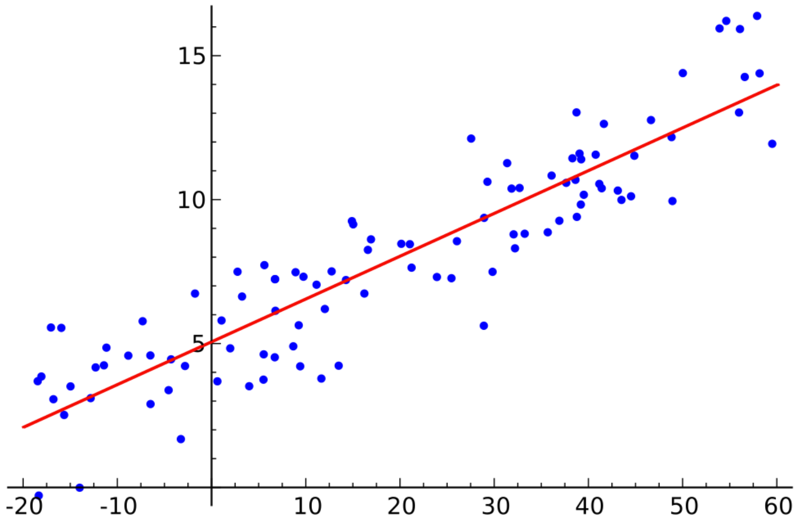
Sometimes,

$$\nabla F(x^*) = 0$$

has a closed form solution.

Example: Linear Regression

$$\min_{\beta \in \mathbb{R}^d} F(\beta)$$



where

$$F(\beta) = \|X\beta - y\|_2^2$$

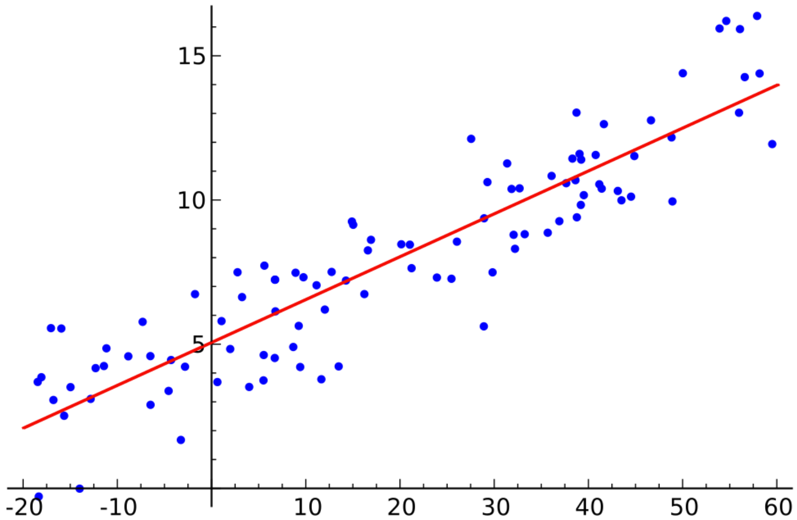
$$\nabla F(\beta) = 2(X^\top X\beta - X^\top y)$$

$$\nabla F(\beta^*) = 0 \quad \Leftrightarrow \quad \beta^* = (X^\top X)^{-1} X^\top y$$

Northeastern

Example: Linear Regression

$$\min_{\beta \in \mathbb{R}^d} F(\beta)$$



where

$$F(\beta) = \|X\beta - y\|_2^2$$

$$\nabla F(\beta) = 2(X^\top X\beta - X^\top \beta)$$

Solve linear system of dimension d x d

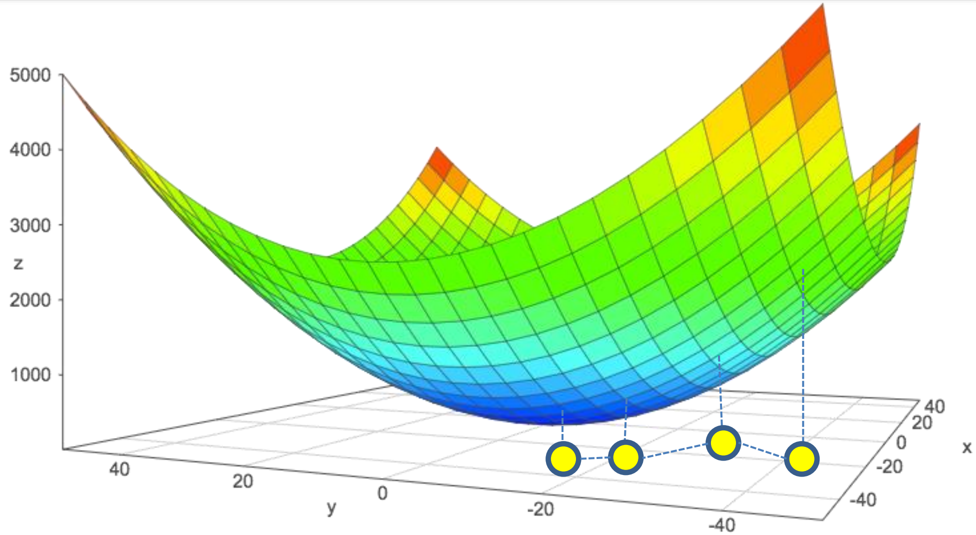
$$\nabla F(\beta^*) = 0 \quad \Leftrightarrow \quad \beta^* = (X^\top X)^{-1} X^\top y$$

Outline

- ❑ Unconstrained Optimization
- ❑ **Gradient Descent**
- ❑ Newton's Method
- ❑ Parallelizing Computations

Gradient Descent

$$x^* = \arg\min_{x \in \mathbb{R}^d} f(x)$$



- produce sequence of points $x^{(k)} \in \mathbf{dom}\, f$, $k = 0, 1, \ldots$ with

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

- can be interpreted as iterative methods for solving optimality condition

$$\nabla F(x^*) = 0$$

General Descent Methods

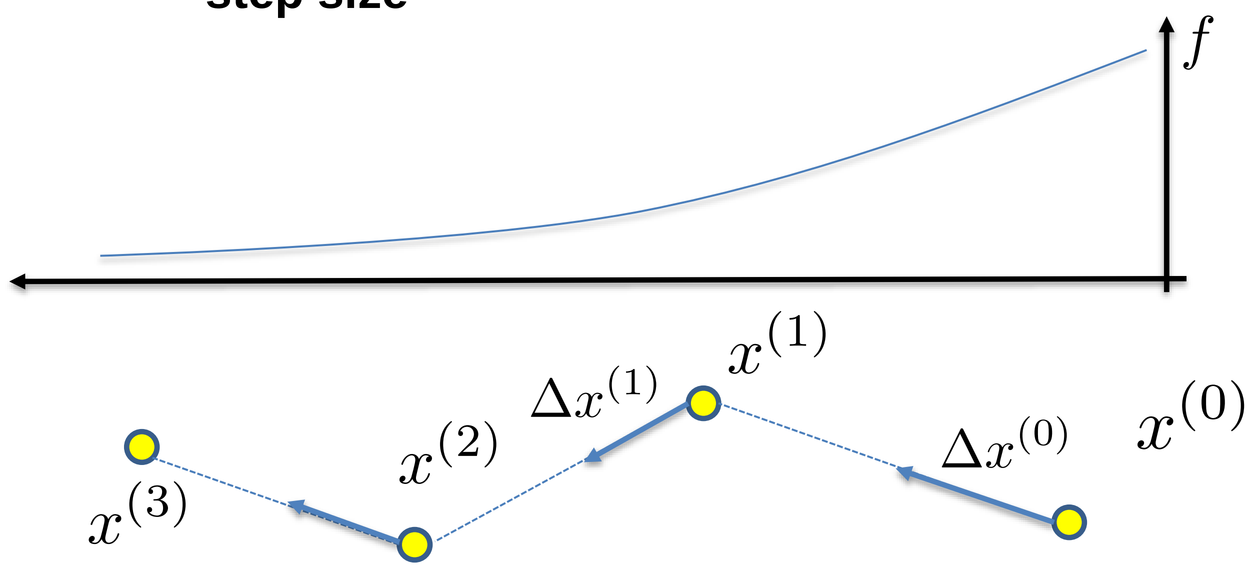
$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

≥ 0

gain or **step size**

$\in \mathbb{R}^d$

descent direction



General Descent Methods

$$x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)} \quad \text{with } f(x^{(k+1)}) < f(x^{(k)})$$

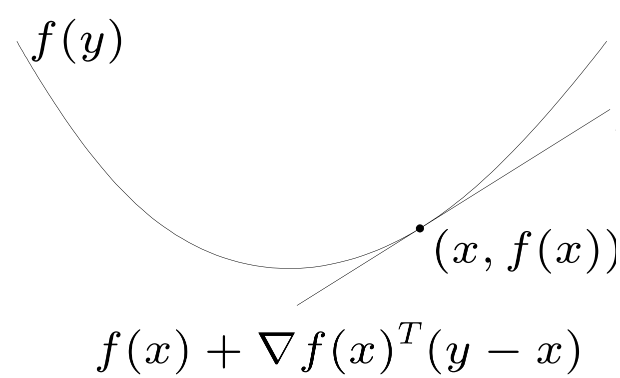
By convexity, if $f(x^{(k+1)}) < f(x^{(k)})$, then it must be that $\nabla f(x^{(k)})^T \Delta x^{(k)} < 0$

Proof: Suppose that $\nabla f(x^{(k)})^T \Delta x^{(k)} \geq 0$

Then,

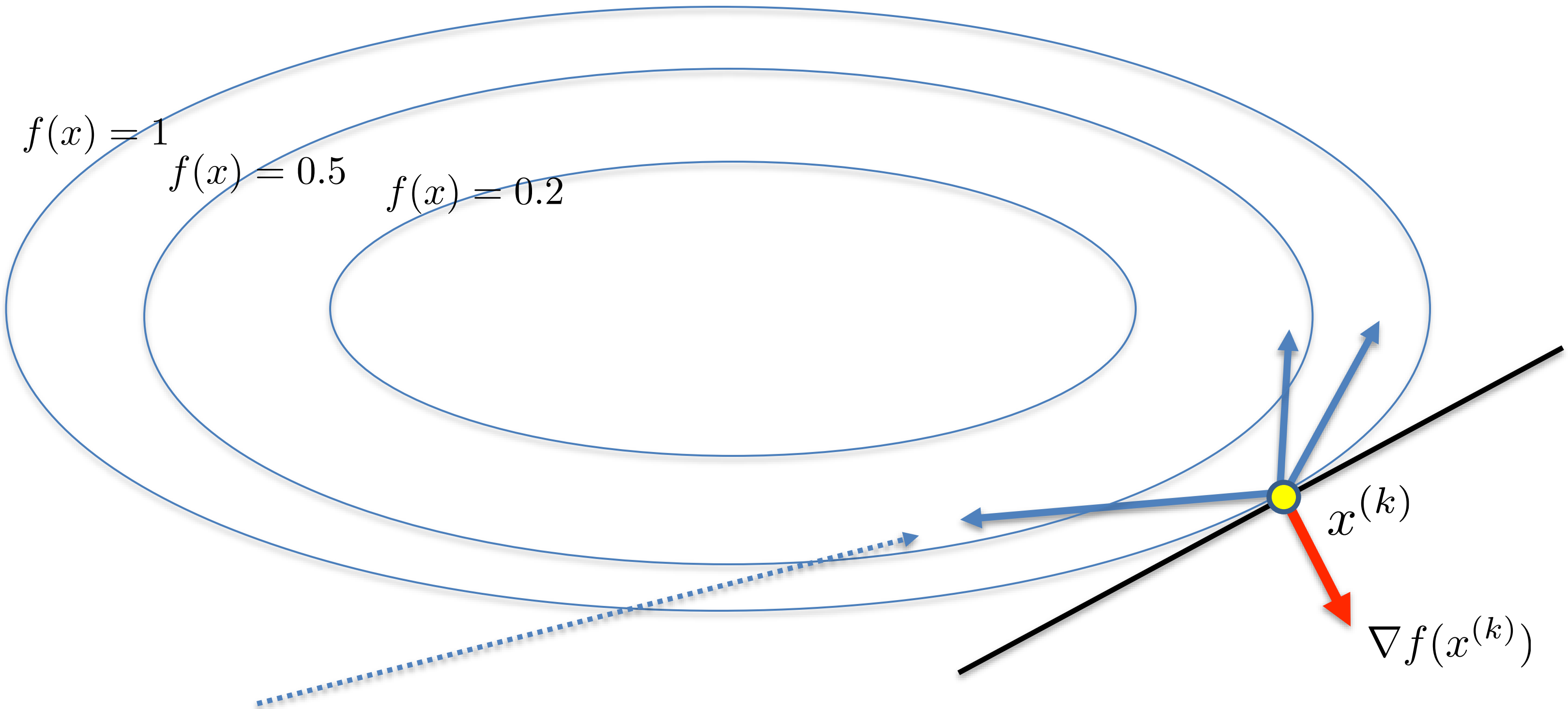
$$\begin{aligned} f(x^{(k+1)}) &\geq f(x^{(k)}) + \nabla f(x^{(k)})^T (x^{(k+1)} - x^{(k)}) \\ &= f(x^{(k)}) + \underbrace{t^{(k)} \cdot \nabla f(x^{(k)})^T \Delta x^{(k)}}_{\geq 0} \\ &\geq f(x^{(k)}) \end{aligned}$$

First-order condition



Descent Direction: View Through Contours

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



all are descent directions

Northeastern

General Descent Method Algorithm

General descent method.

given a starting point $x \in \mathbf{dom}\, f$.
repeat

1. Determine a descent direction Δx.
2. *Line search.* Choose a step size $t > 0$.
3. *Update.* $x := x + t\Delta x$.

until stopping criterion is satisfied.

Q: How to choose descent direction Δx?

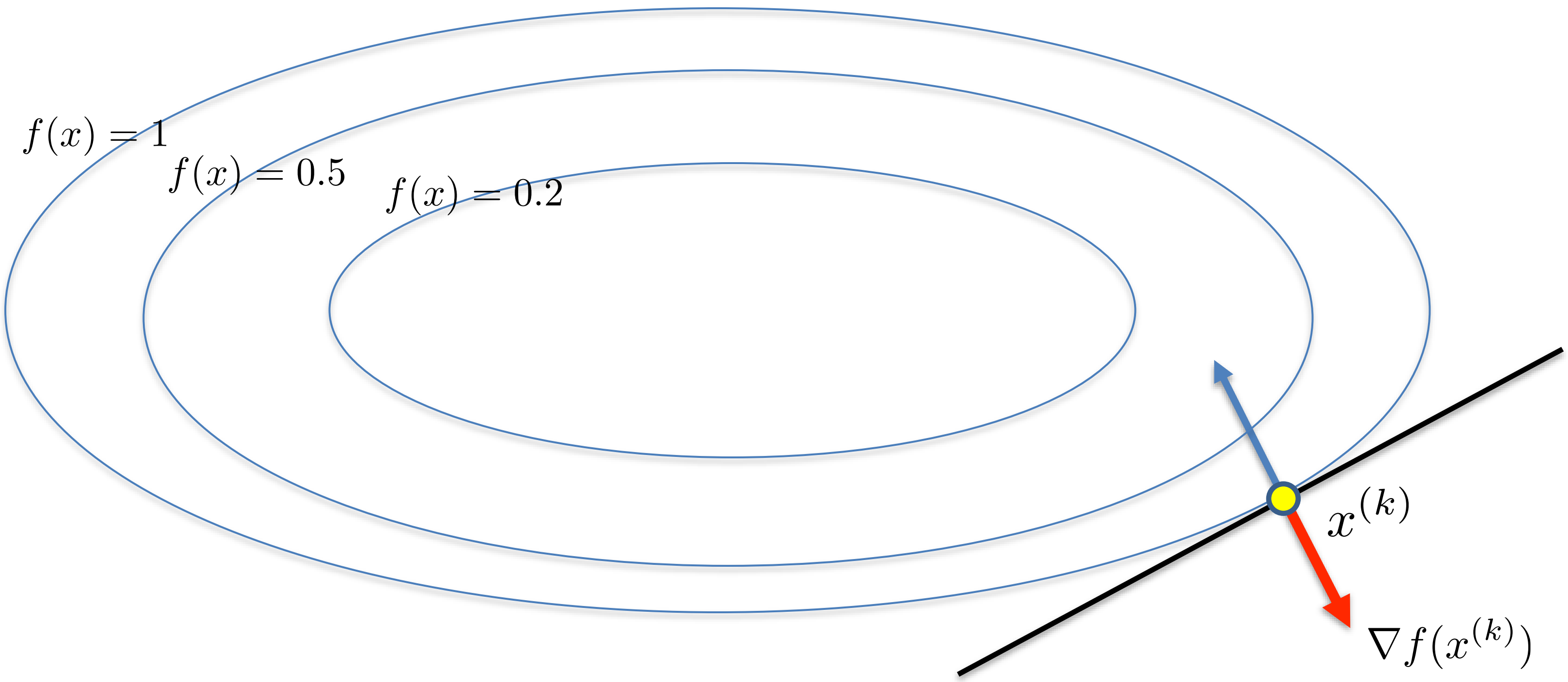
Q: How to choose step size t?

Q: How to choose stopping criterion?

Gradient Descent

Choose descent direction

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



Constant Step Size

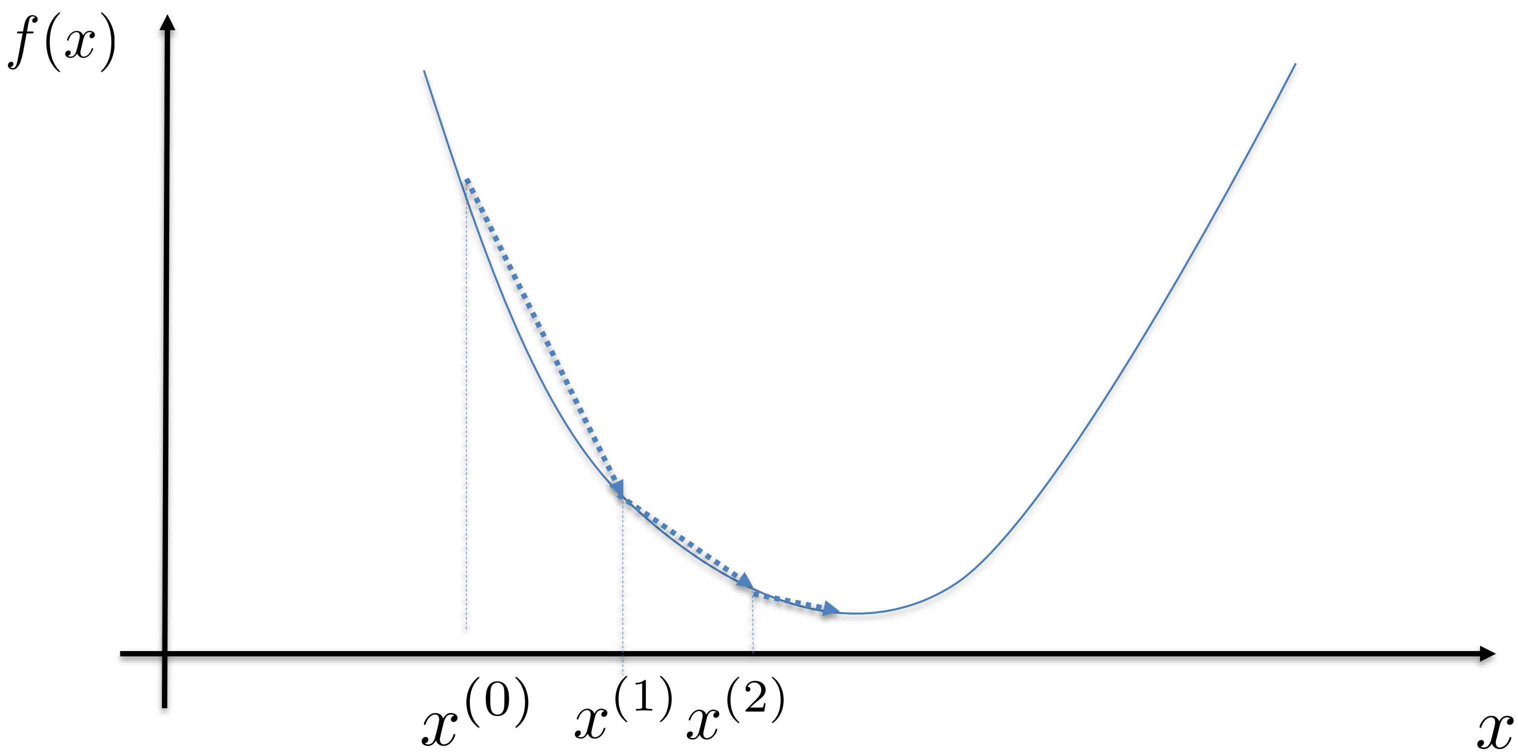
$$x := x + t\Delta x \qquad\qquad t = 0.1$$

- Simple
- May not converge!

Constant Step Size

$$x := x + t\Delta x \qquad t = 0.1$$

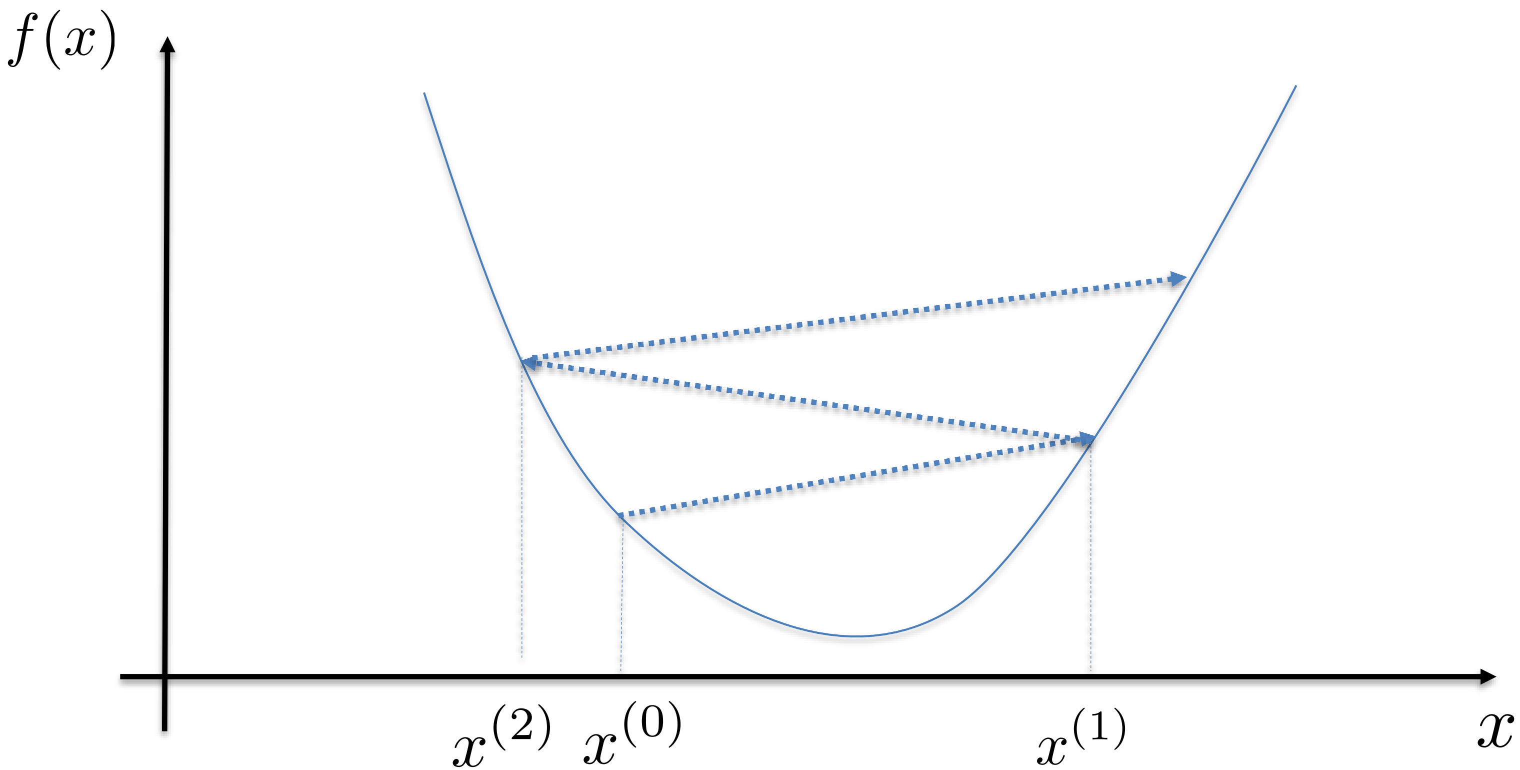
Good case



Constant Step Size

$$t = 0.1$$

Bad case

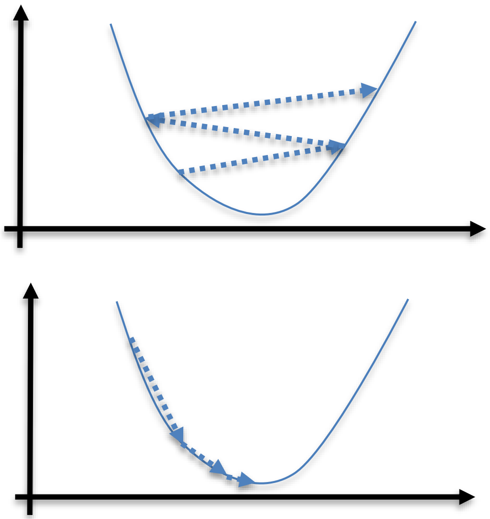


Northeastern

Decreasing Step Size

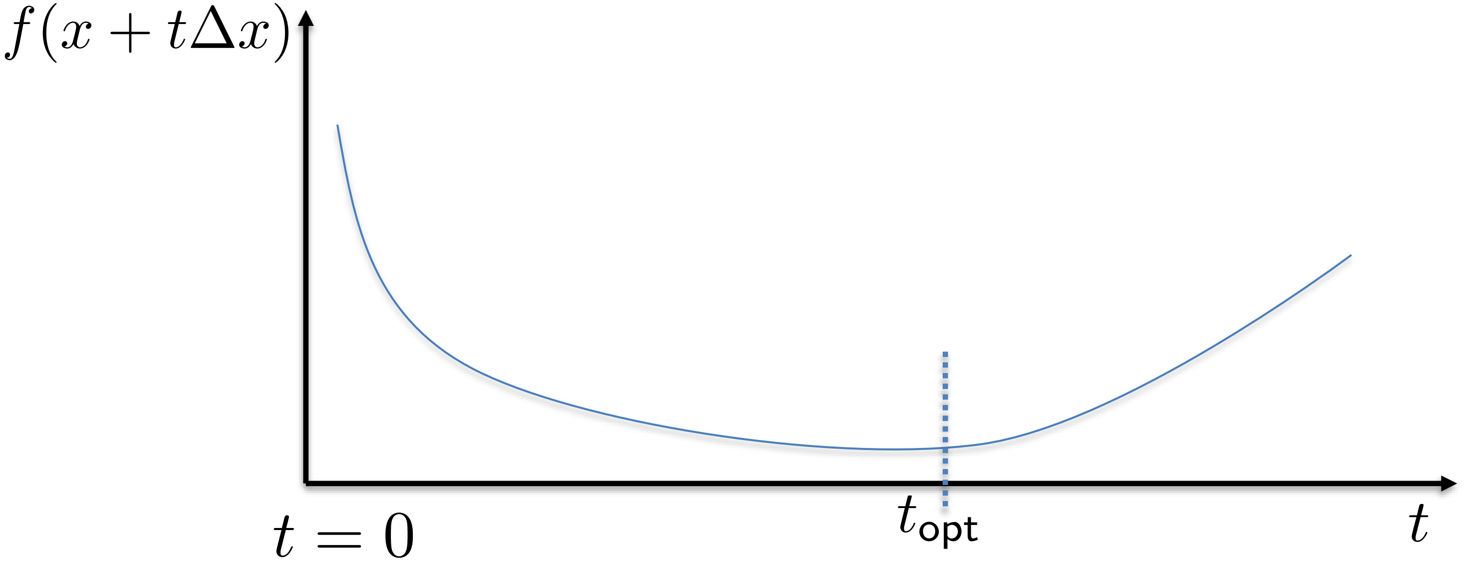
E.g. $x^{(k+1)} = x^{(k)} + t^{(k)}\Delta x^{(k)} \quad t^k = \dfrac{1}{k+1}$

- ❑Simple
- ❑Will eventually converge
- ❑May be slow
    - ❑Can start "bad" before becoming "good"
    - ❑Not guaranteed descent with every step



Exact Line Search

$$t_{\text{opt}} = \text{argmin}_{t>0}\, f(x + t\Delta x)$$



- ❑ Guaranteed descent
- ❑ May be hard to compute: optimization in one variable!

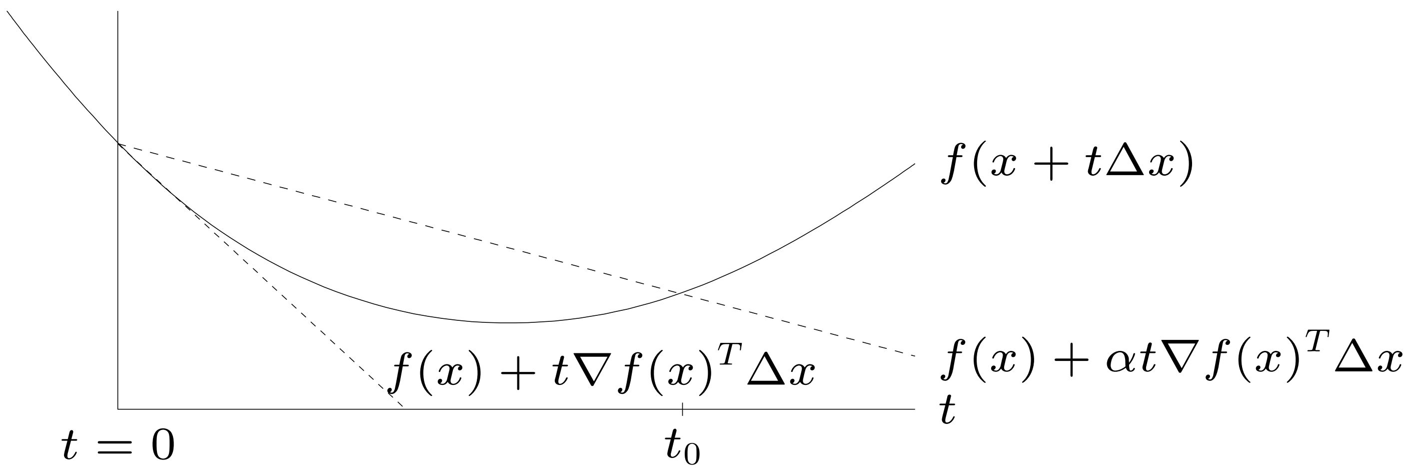
Backtracking Line Search

backtracking line search (with parameters $\alpha \in (0, 1/2)$, $\beta \in (0, 1)$)

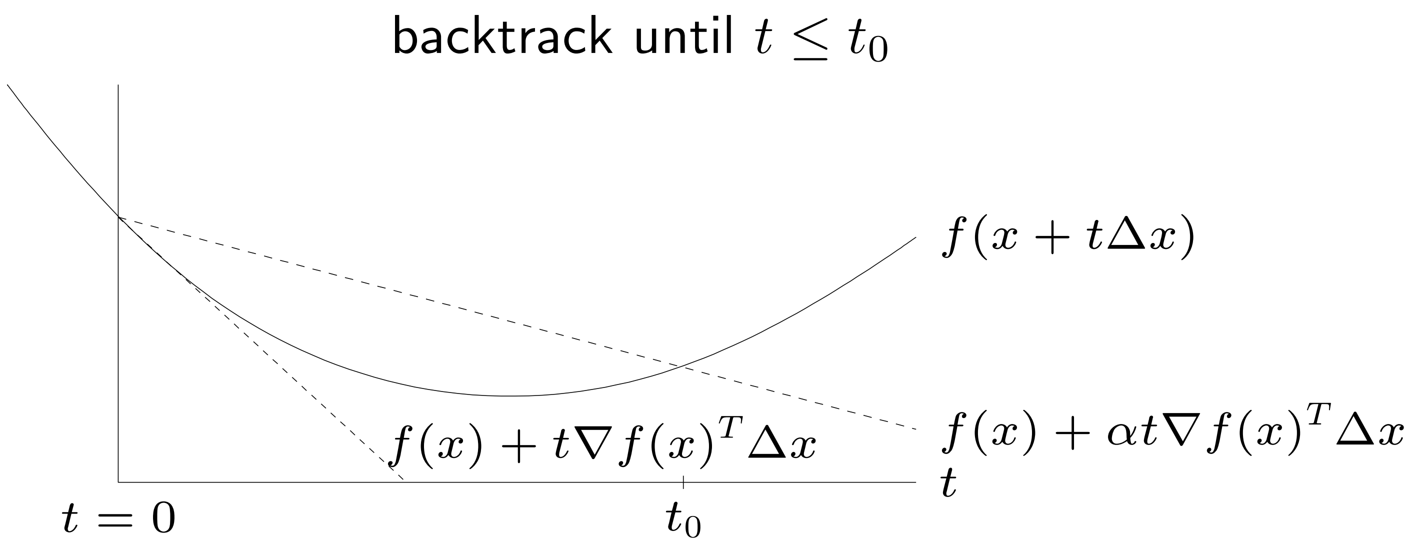
- starting at $t = 1$, repeat $t := \beta t$ until

$$f(x + t\Delta x) < f(x) + \alpha t \nabla f(x)^T \Delta x$$

- graphical interpretation: backtrack until $t \leq t_0$



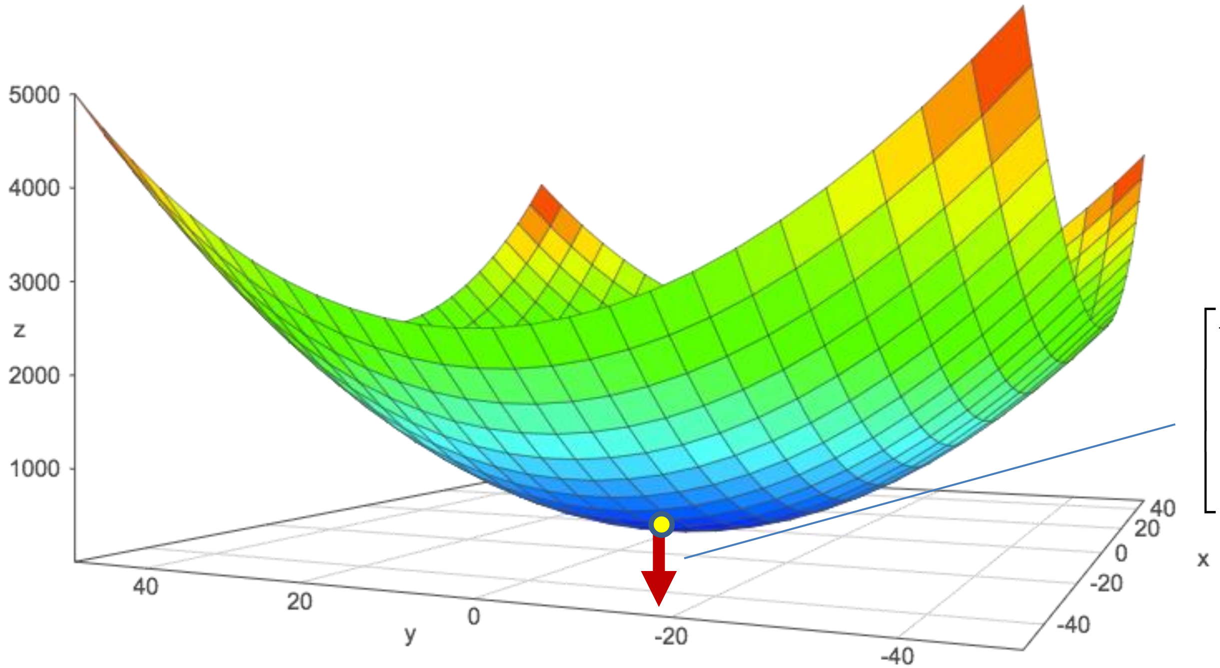
Backtracking Line Search



- ❑ Guaranteed descent
- ❑ Requires multiple **function** calls but **only one gradient** call

Stopping Criterion

$$\|\nabla f(x)\|_2 \leq \epsilon$$



$$\begin{bmatrix} -1 \\ 0 \\ 0 \end{bmatrix}$$

Gradient Descent: Putting Everything Together

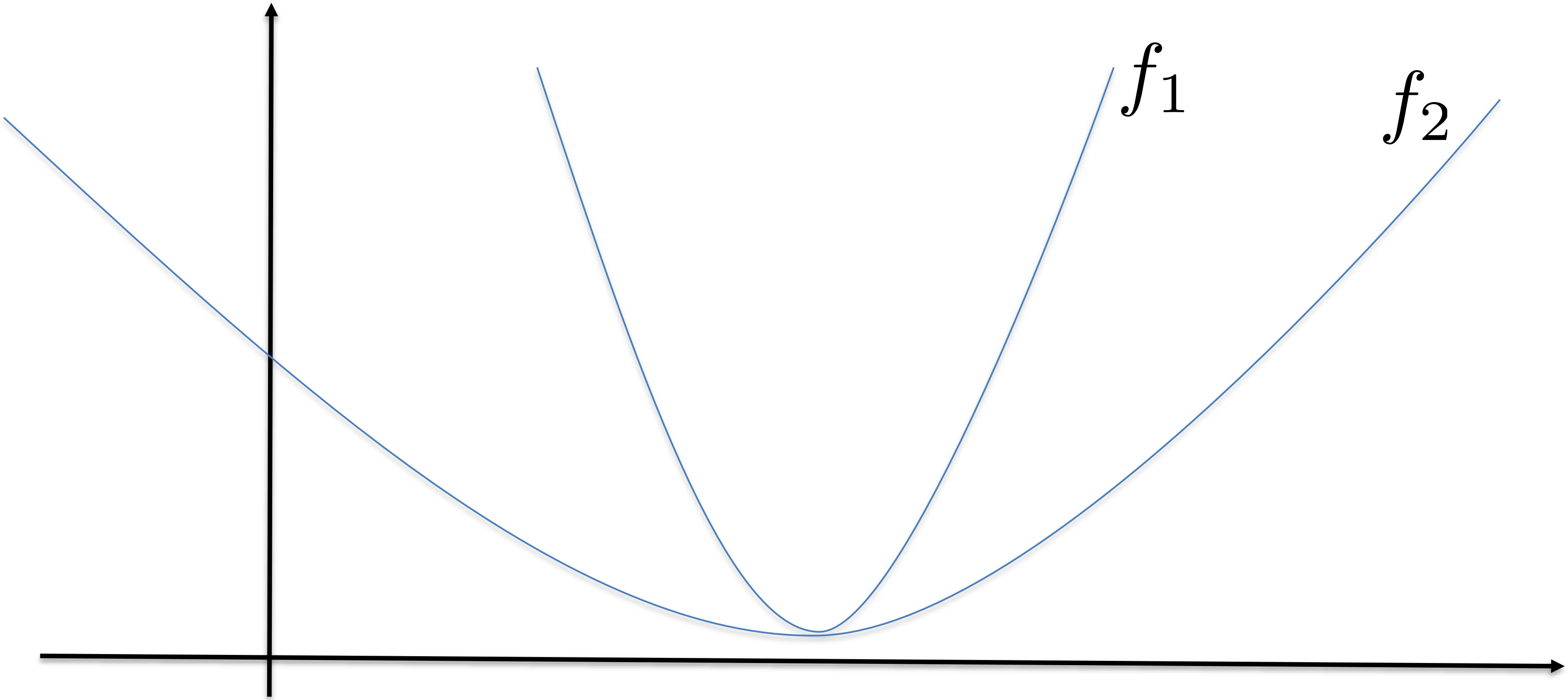
---

given a starting point $x \in \textbf{dom}\, f$.
repeat

1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until $\|\nabla f(x)\|_2 \leq \epsilon$

Convergence Properties



The steeper f is, the faster the convergence

Strong Convexity

- For $A, B \succeq 0$, we write

$$A \succeq B$$

if

$$A - B \succeq 0$$

Strong Convexity

- Convex:

$$\nabla^2 f(x) \succeq 0$$

- *Strictly* convex:

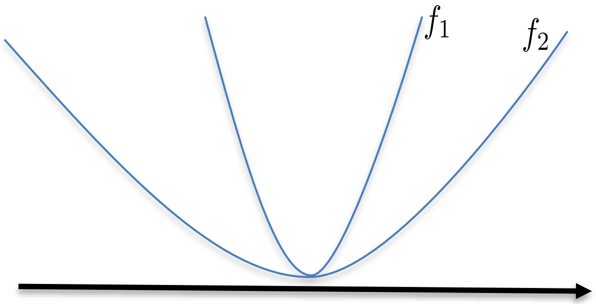
$$\nabla^2 f(x) \succ 0$$

- *Strongly* convex:

$$\nabla^2 f(x) \succeq mI, \quad \text{where } m > 0$$

Eigenvalues bounded away from zero!

Strong Convexity



f is strongly convex on S if there exists an $m > 0$ such that

$$\nabla^2 f(x) \succeq mI \qquad \text{for all } x \in S$$

implications

- for $x, y \in S$,

$$f(y) \geq f(x) + \nabla f(x)^T (y - x) + \frac{m}{2}\|x - y\|_2^2$$

Convergence of Gradient Descent

---

given a starting point $x \in \mathbf{dom}\, f$.
repeat
1. $\Delta x := -\nabla f(x)$.
2. *Line search.* Choose step size t via exact or backtracking line search.
3. *Update.* $x := x + t\Delta x$.

until $\|\nabla f(x)\|_2 \leq \epsilon$

- convergence result: for strongly convex f,

$$f(x^{(k)}) - f(x^*) \leq c^k \left(f(x^{(0)}) - f(x^*) \right)$$

$c \in (0, 1)$ depends on m, $x^{(0)}$, line search type

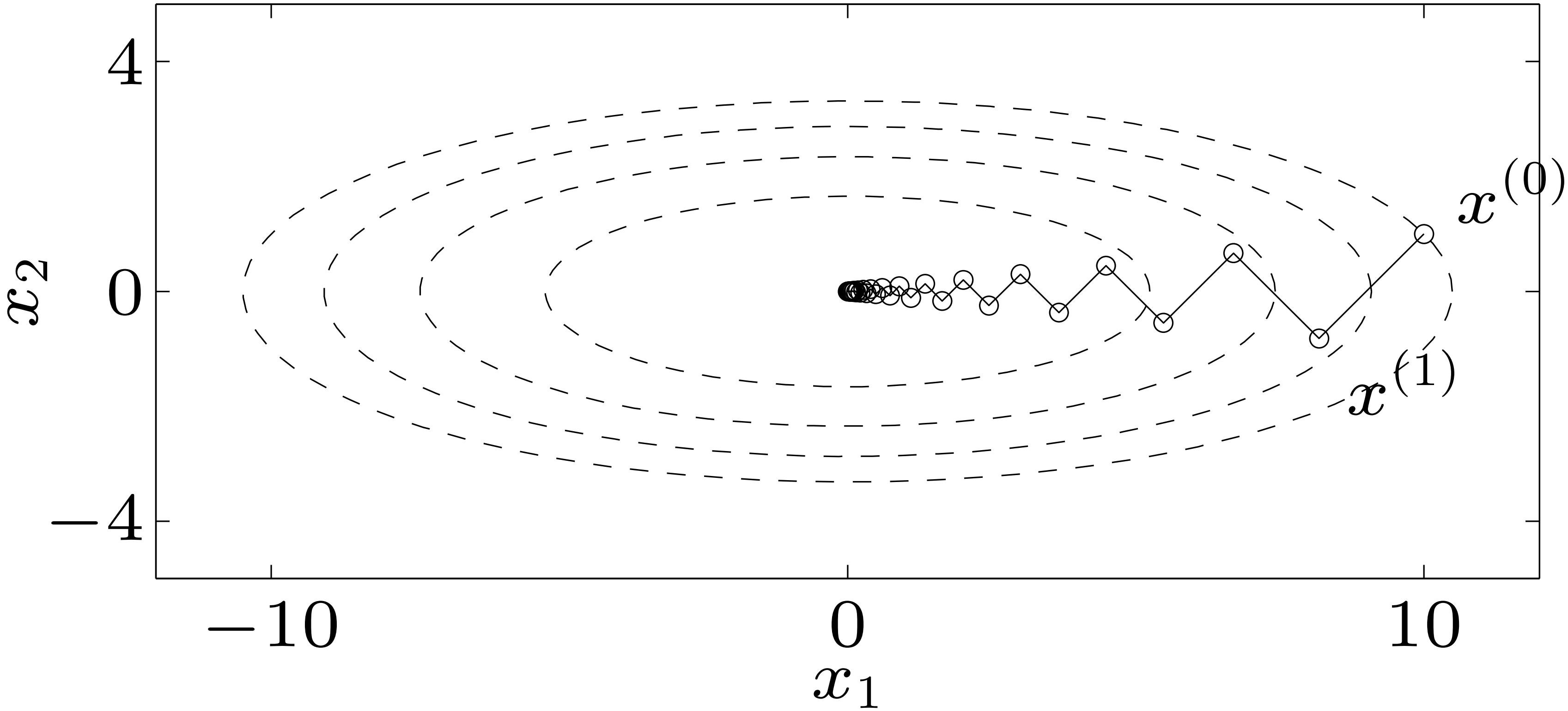
Strong Convexity and Quadratic Penalty

- Suppose that $f : \mathbb{R}^d \to \mathbb{R}$ is convex. Then

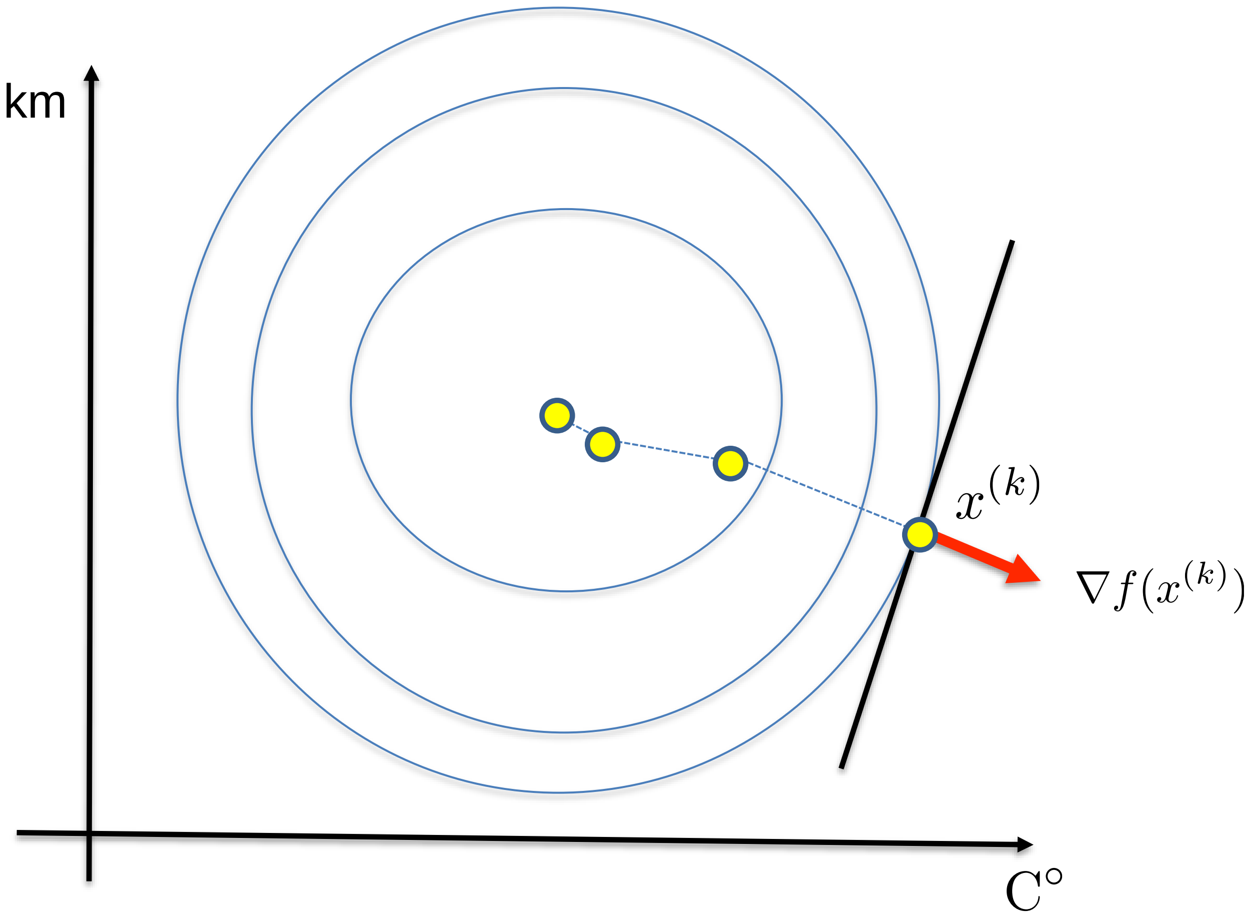
$$g(x) = f(x) + \lambda \|x\|_2^2$$

where $\lambda > 0$ is *strongly convex*.

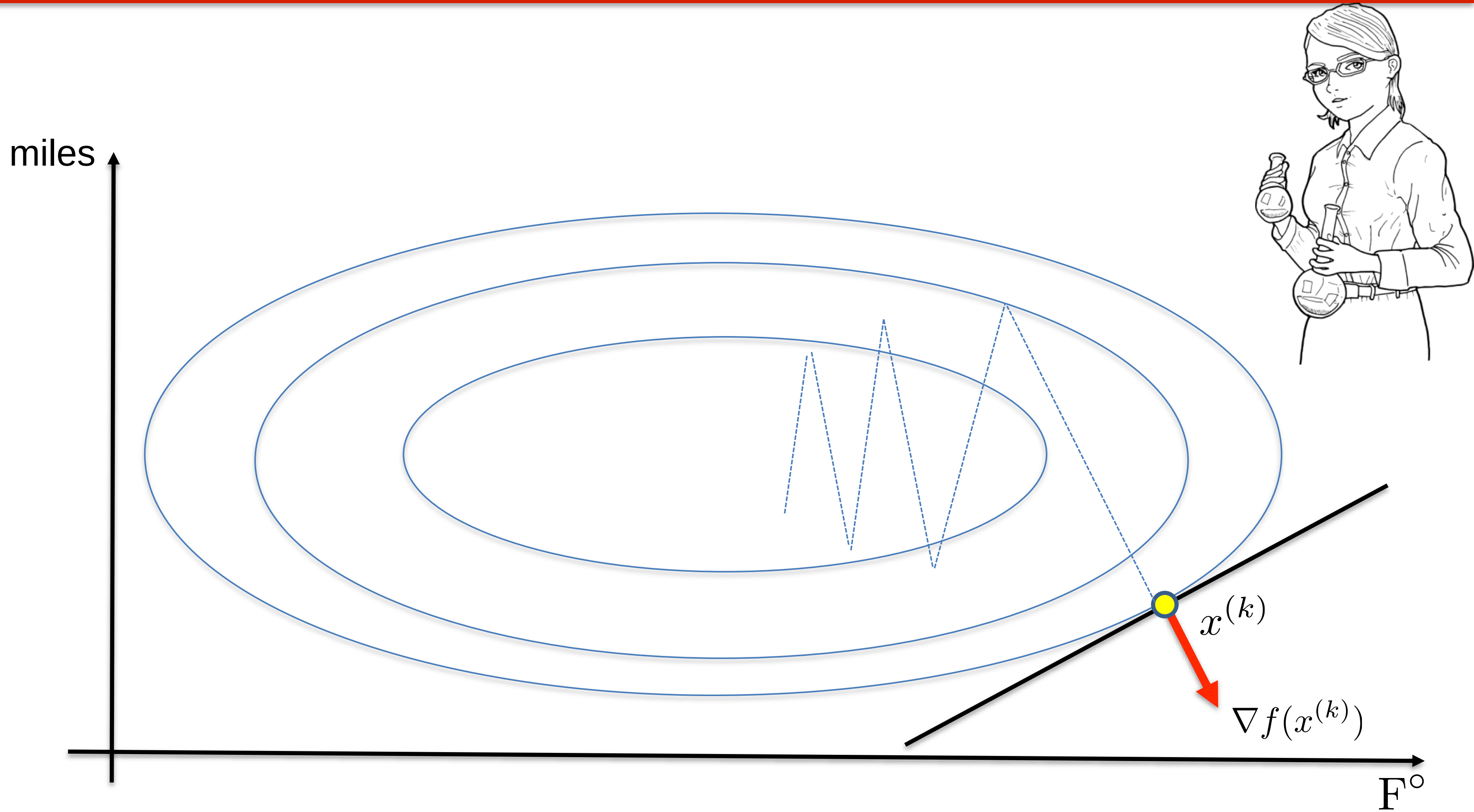
An Example of Slow Convergence



Why is this a problem?



Why is this a problem?



Outline

- ❑Unconstrained Optimization
- ❑Gradient Descent
- ❑Newton's Method
- ❑Parallelizing Computations

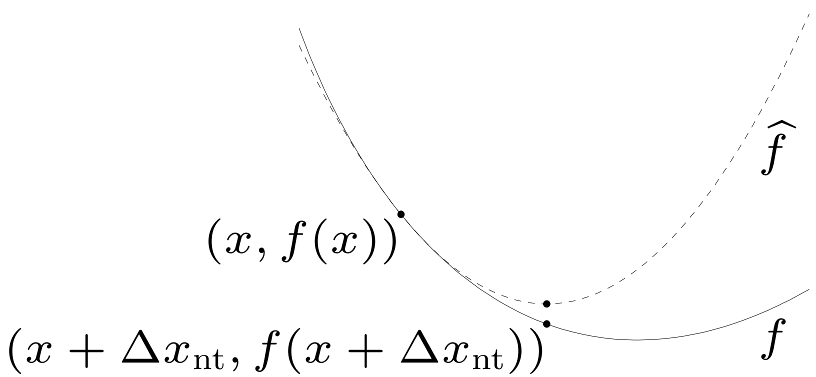
Newton's Method

$$\Delta x_{\rm nt} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

interpretation

$x + \Delta x_{\rm nt}$ minimizes second order approximation

$$\widehat{f}(x+v) = f(x) + \nabla f(x)^T v + \frac{1}{2} v^T \nabla^2 f(x) v$$



Northeastern

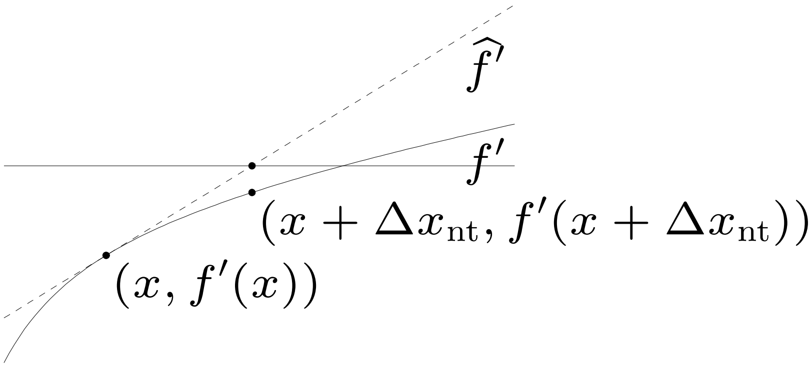
Newton's Method

$$\Delta x_{\mathrm{nt}} = -\nabla^2 f(x)^{-1} \nabla f(x)$$

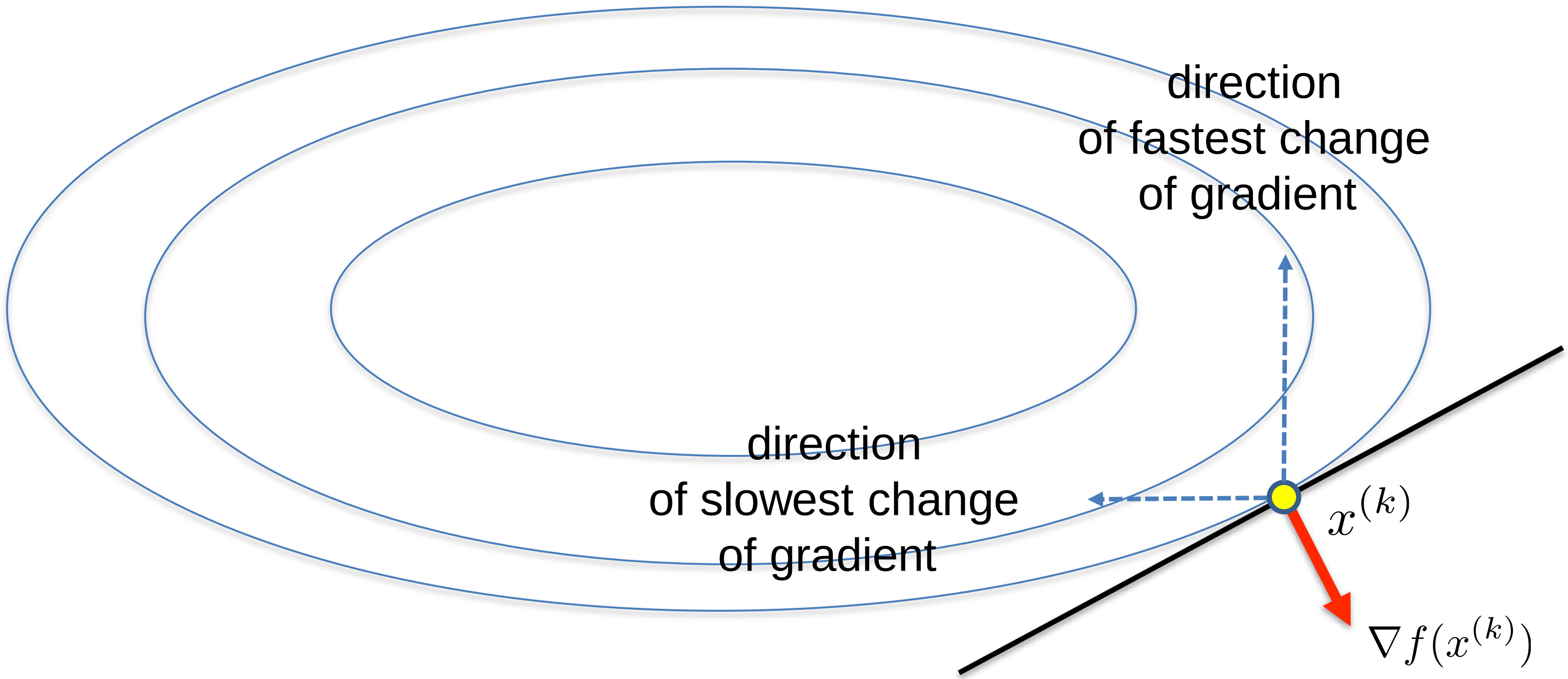
interpretation

$x + \Delta x_{\mathrm{nt}}$ solves linearized optimality condition

$$\nabla f(x+v) \approx \nabla \widehat{f}(x+v) = \nabla f(x) + \nabla^2 f(x) v = 0$$

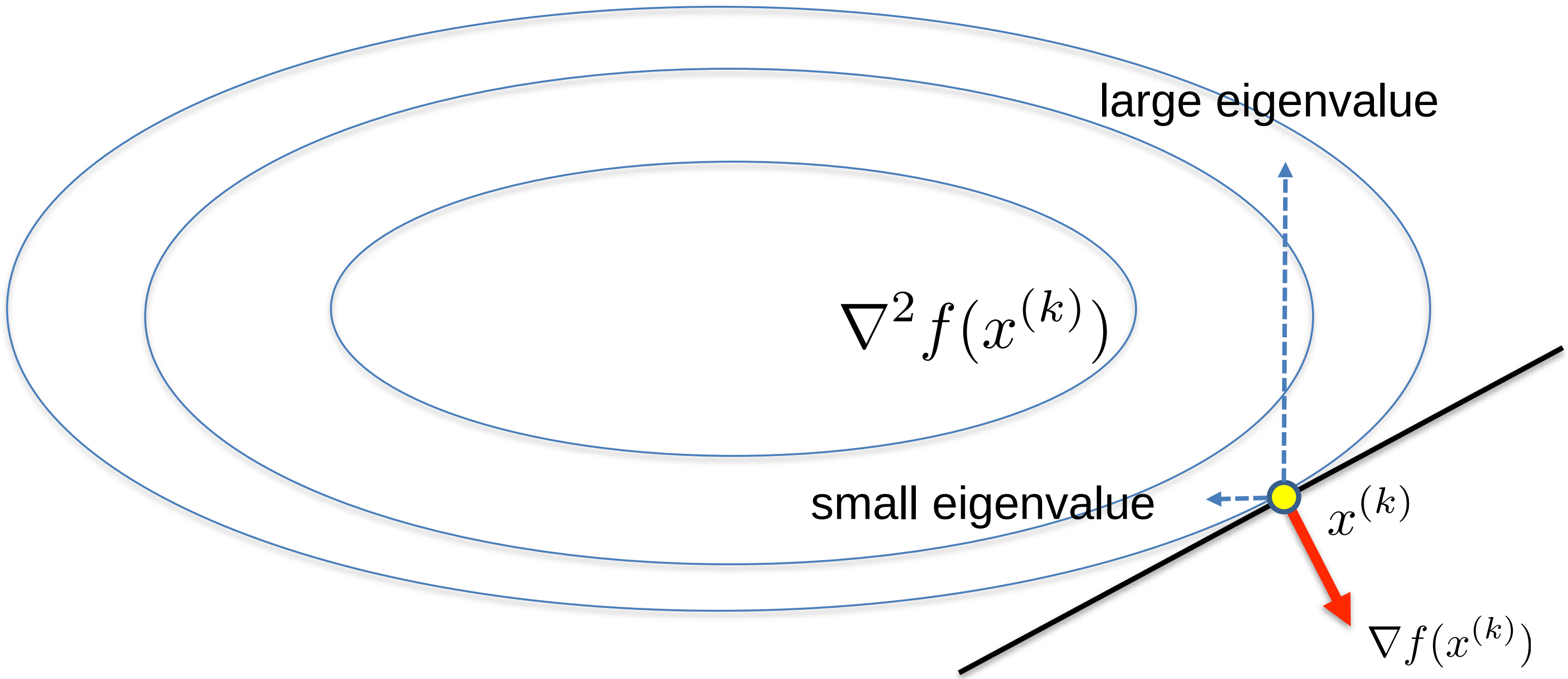


Newton's Method: Geometric Interpretation

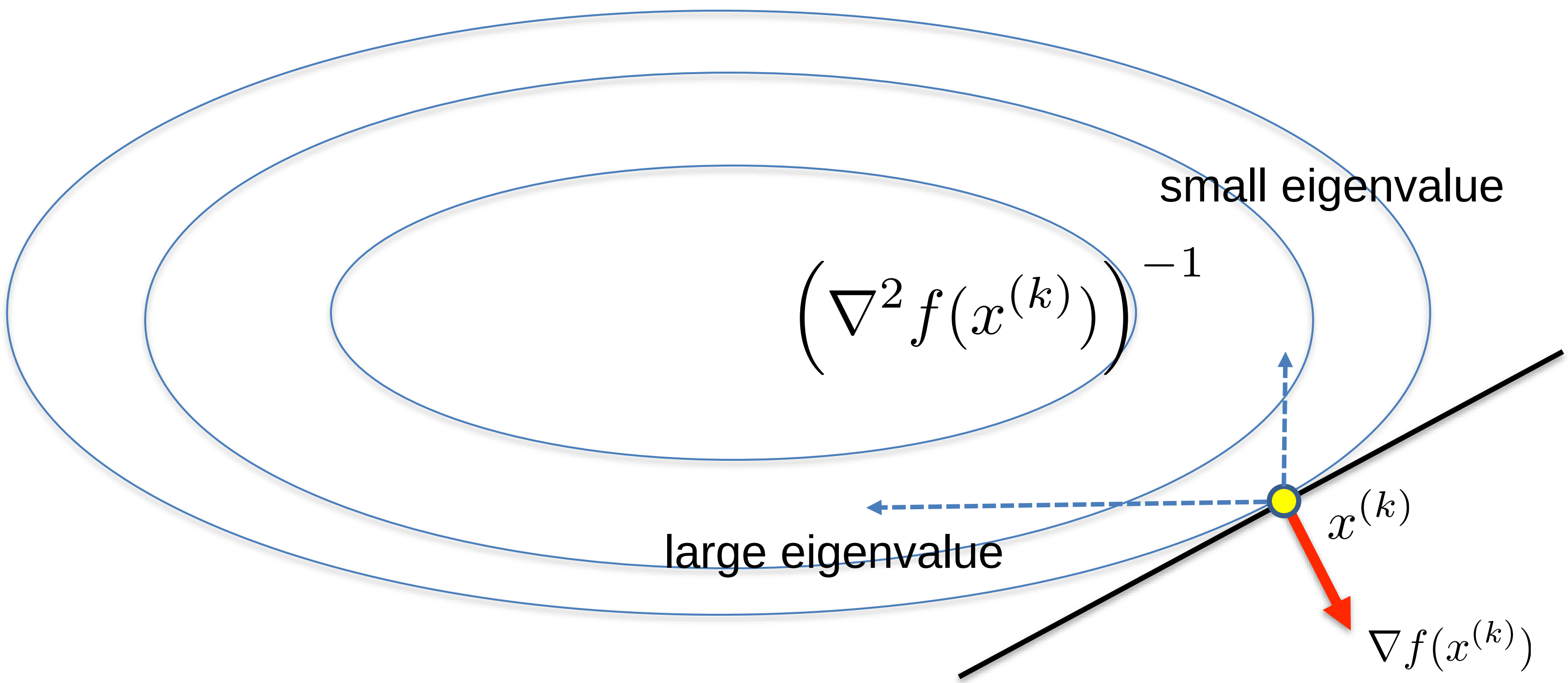


Northeastern

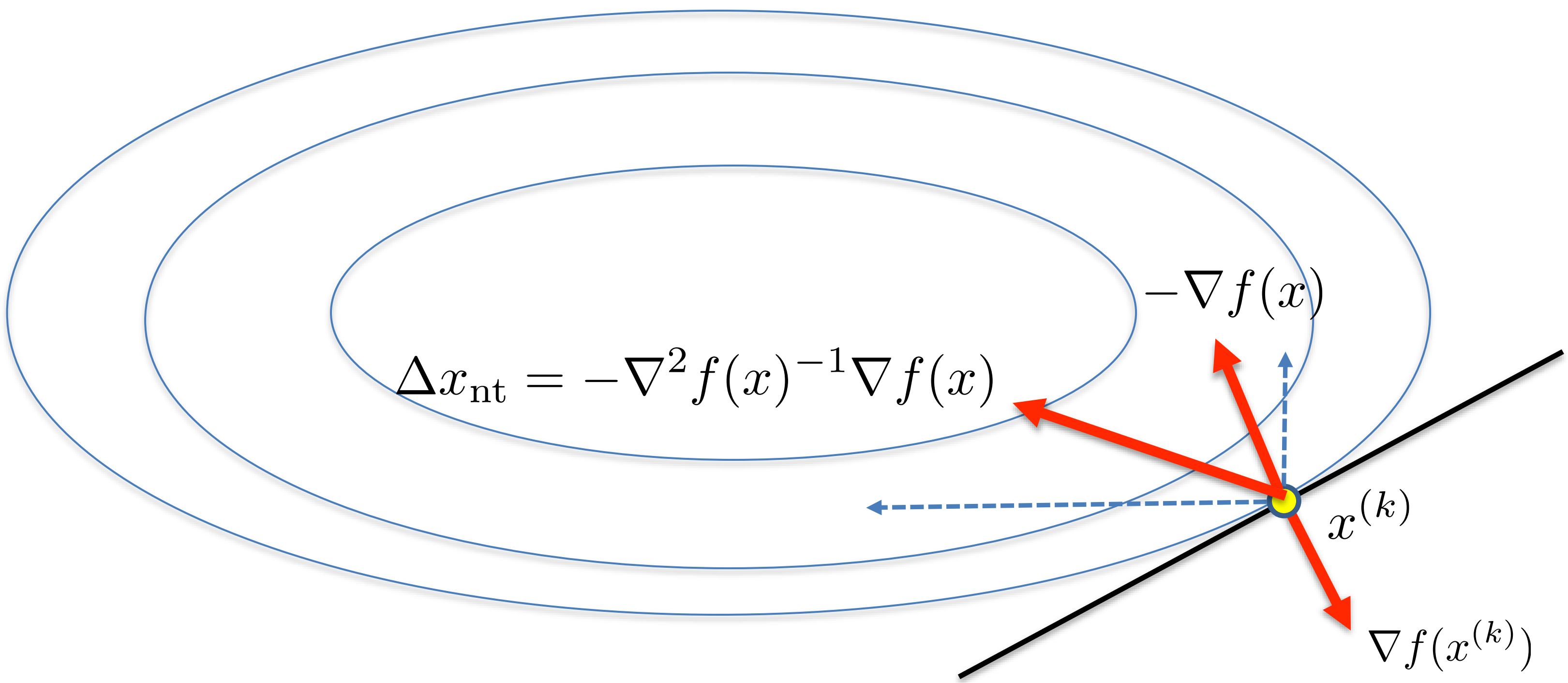
Newton's Method: Geometric Interpretation



Newton's Method: Geometric Interpretation



Newton's Method: Geometric Interpretation



Newton's Method

- Faster convergence
- Scale-free!
- Computationally intensive

$$f : \mathbb{R}^d \to \mathbb{R}$$

$$\nabla f : \mathbb{R}^d \to \mathbb{R}^d$$

$$\nabla^2 f : \mathbb{R}^d \to \mathbb{R}^{d \times d}$$

Increasing Complexity

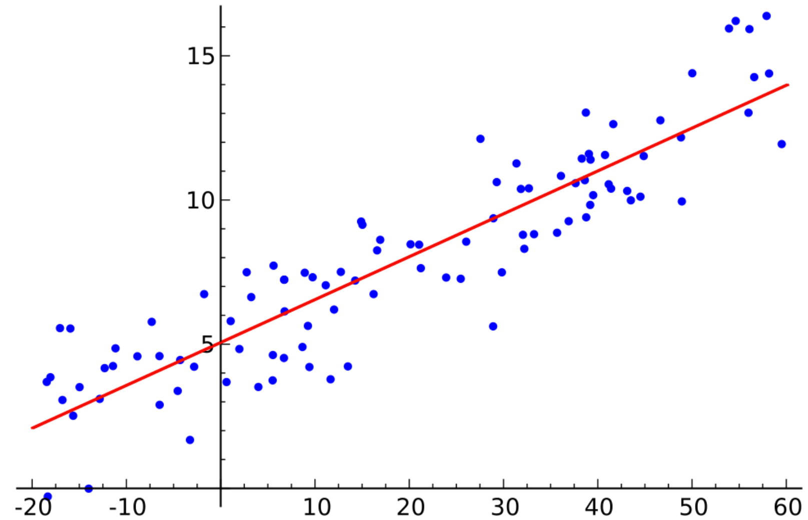
Outline

- ❑Unconstrained Optimization
- ❑Gradient Descent
- ❑Newton's Method
- ❑Parallelizing Computations

Loss Functions

$$\min_{\beta \in \mathbb{R}^d} F(\beta)$$

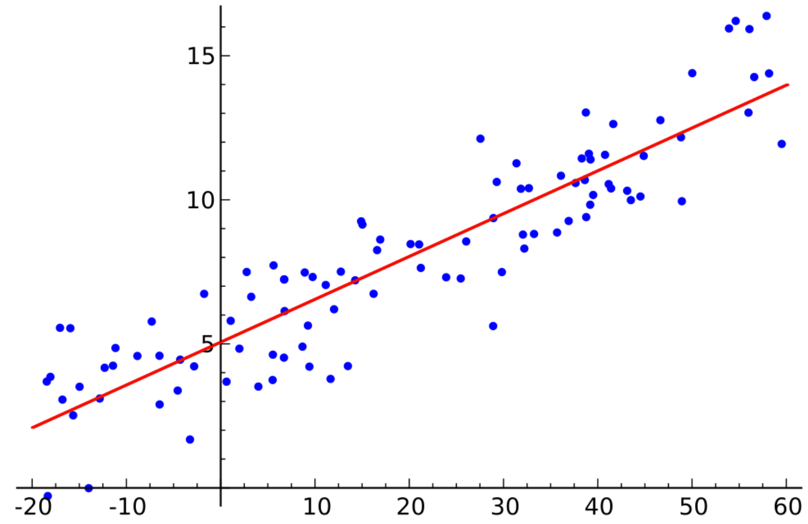
$$F(\beta) = \|X\beta - y\|_2^2$$
$$= \sum_{i=1}^{n} (y_i - \beta^\top x_i)^2$$



Loss Functions

$$\min_{\beta \in \mathbb{R}^d} F(\beta)$$

$$F(\beta) = \sum_{i=1}^{n} \ell(\beta; x_i, y_i)$$



- If ℓ is convex, so is $F(\beta)$
- Examples:
  - Squared loss $\ell(\beta; x, y) = (y - \beta^\top x)^2$
  - Logistic $\ell(\beta; x, y) = \log(1 + \exp(-y\beta^\top x))$
  - Hinge $\ell(\beta; x, y) = \max(0, 1 - y\beta^\top x)$

Parallel Computation

$$F(\beta) = \sum_{i=1}^{n} \ell(\beta; x_i, y_i)$$

- ❑ Track Objective
- ❑ Backtracking Line Search

$$\nabla F(\beta) = \sum_{i=1}^{n} \nabla_\beta \ell(\beta; x_i, y_i)$$

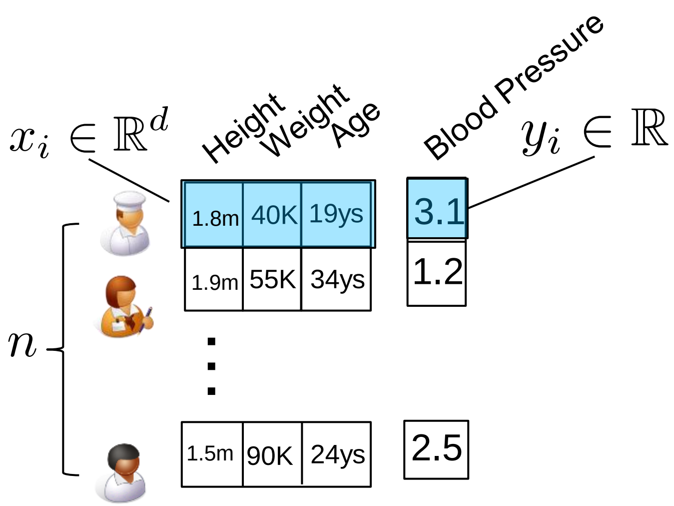
- ❑ GD step
- ❑ Convergence Criterion

$$\nabla^2 F(\beta) = \sum_{i=1}^{n} \nabla_\beta^2 \ell(\beta; x_i, y_i)$$

- ❑ Newton Step

Parallel Computation

$$F(\beta) = \sum_{i=1}^{n} \ell(\beta; x_i, y_i)$$



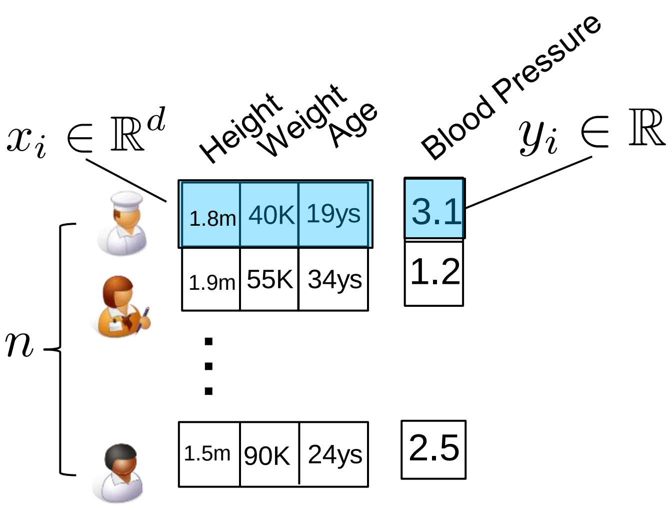

```
rdd = [(x1,y1),
       (x2,y2),
        ...
       (xn,yn)]

beta = np.array([0.1,0.4,-2.0])

rdd.map( lambda (x,y):
                 loss(beta,x,y))\
        .reduce(add)
```

- ❑ **Broadcasts O(d) sized vector**
- ❑ **Reduction msg size O(1)**

Northeastern

Parallel Computation

$$\nabla F(\beta) = \sum_{i=1}^{n} \nabla_\beta \ell(\beta; x_i, y_i)$$

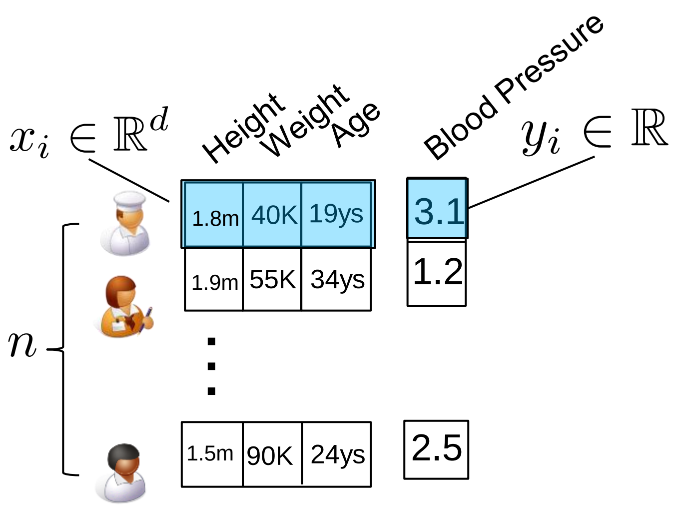



```
rdd = [(x1,y1),
       (x2,y2),
        …
       (xn,yn)]

beta = np.array([0.1,0.4,-2.0])

rdd.map( lambda (x,y):
                gradLoss(beta,x,y))\
     .reduce(add)
```

- ❑ **Broadcasts O(d) sized vector**
- ❑ **Reduction msg size O(d)**

Parallel Computation

$$\nabla^2 F(\beta) = \sum_{i=1}^{n} \nabla_\beta^2 \ell(\beta; x_i, y_i)$$

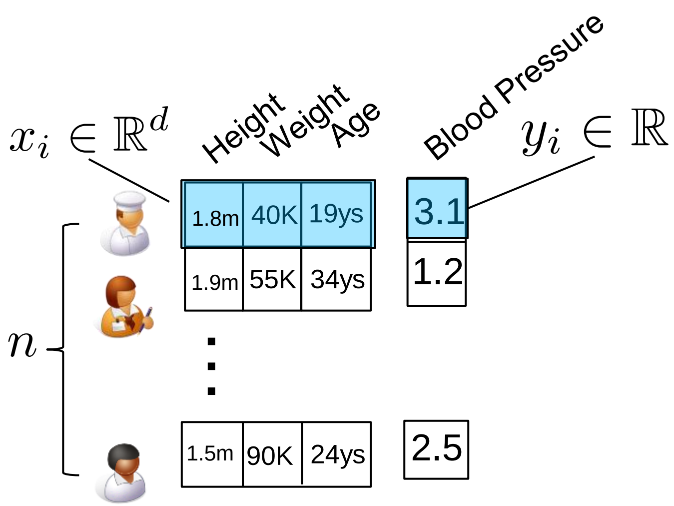



```
rdd = [(x1,y1),
       (x2,y2),
        …
       (xn,yn)]

beta = np.array([0.1,0.4,-2.0])

rdd.map( lambda (x,y):
                hessLoss(beta,x,y))\
      .reduce(add)
```

- ❑ **Broadcasts O(d) sized vector**
- ❑ **Reduction msg size O(d^2)**

Parallel Computation

$$F(\beta) = \sum_{i=1}^{n} \ell(\beta; x_i, y_i) \qquad \nabla F(\beta) = \sum_{i=1}^{n} \nabla_\beta \ell(\beta; x_i, y_i)$$



- Gradient descent parallelizable when n is big and d is small
- What about n>>1 **and** d>>1?
- Leverage **sparsity** (HW3 & future lectures)