Ch 4: Numerical Computation

Deep Learning Textbook Study Group, SF

Safak Ozkan
March 13, 2017

Overflow and Underflow

- **Rounding errors**

$$\text{softmax}(\boldsymbol{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^{n} \exp(x_j)}.$$

The exponentiation can

- **underflow** when the argument is a **small** number.
- **overflow** when it is a **large** number.

Remedy:

- Transform input variables $z_i = x_i - max(x_i)$
- Operate on $\text{softmax}(z)$

Overflow and Underflow

```
x = 1e9
eps = 1e-6

for _ in xrange(int(1e6)):
    x += eps

print x
```

```
In [11]: %run main.py
1000000000.95
```

```
x = 1e9
eps = 1e-6

for _ in xrange(int(1e6)):
    x += eps

for _ in xrange(int(1e6)):
    x -= eps

print x
```

```
In [14]: %run main.py
1000000000.0
```

Condition Number

- **Definition:**
  The ratio of relative error in x to relative error in b. → $\kappa(A) = \dfrac{\frac{||u||}{||x||}}{\frac{||\epsilon||}{||b||}}$
- $Ax = b$
  $A(x+u) = b + \epsilon$
  where ϵ : error in b, u : error in x
- $\boxed{\kappa(A) = \left|\dfrac{\lambda_{max}}{\lambda_{min}}\right|}$

- $Av = \lambda v$ … **eigenvalue equation**
- $\epsilon \approx v_{min}$
  $b \approx v_{max}$

Condition Number

- **Connection to ML:**
  Multicollinearity

$$M = \begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}_{m\times n}, \quad y = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}_{m\times 1}$$

$$Mx = y$$

$$M^T M x = M^T y$$

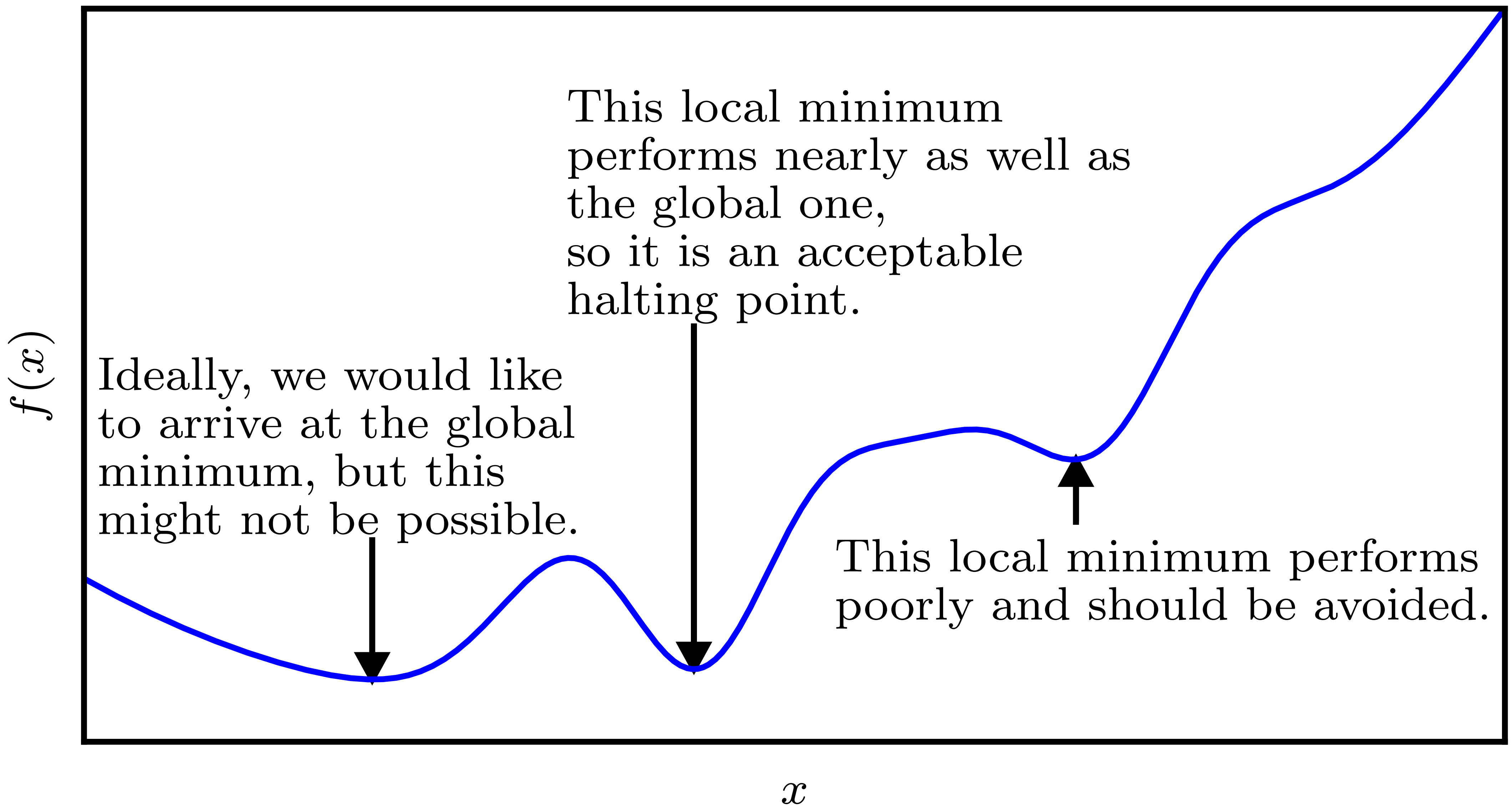
$$Ax = b \quad (A_{n\times n},\ x_{n\times 1},\ b_{n\times 1})$$

$$x = A^{-1} b$$

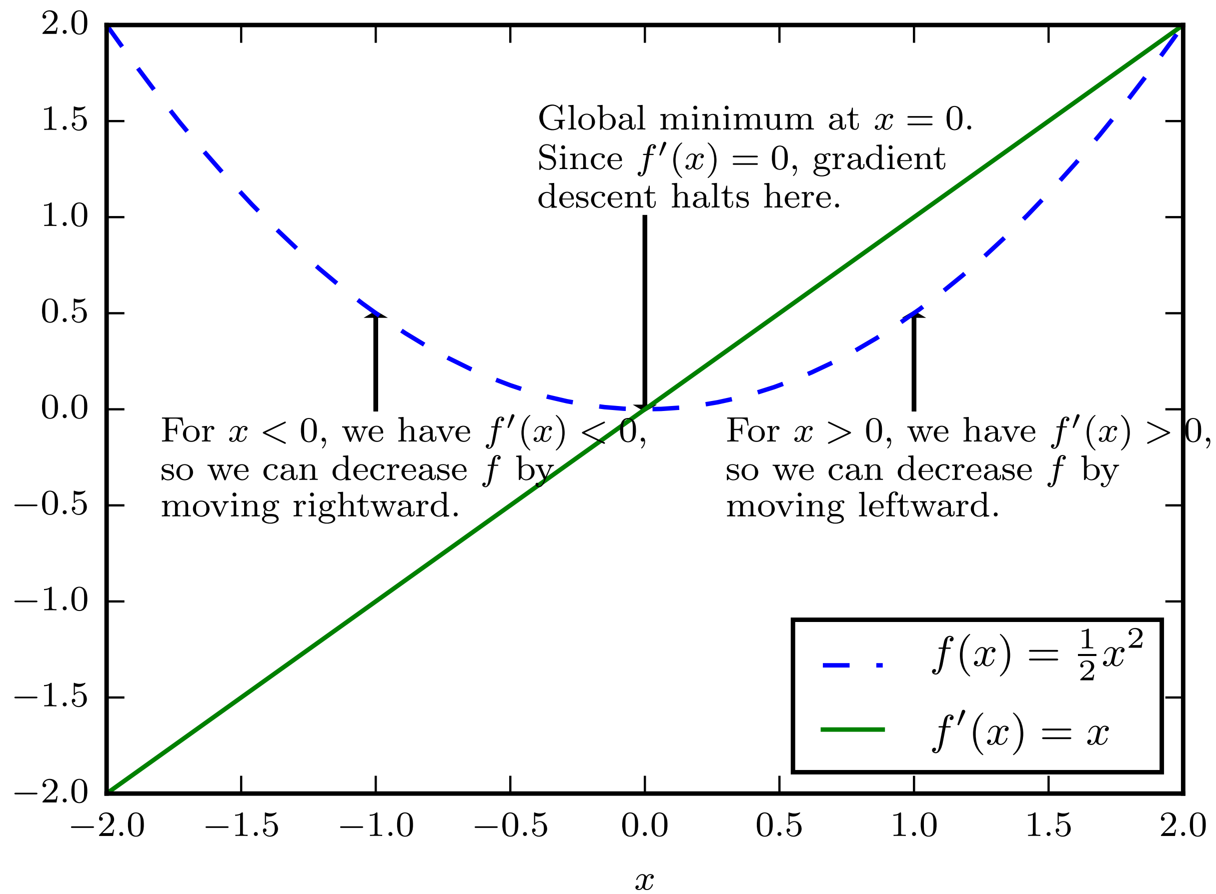
- $$\boxed{\kappa(A) = \left|\frac{\lambda_{max}}{\lambda_{min}}\right|}$$

- $\kappa(A) \gg O(1)$, ill-conditioned.
- $\kappa(A) = O(1)$, well-conditioned.

Optimization



Gradient Descent

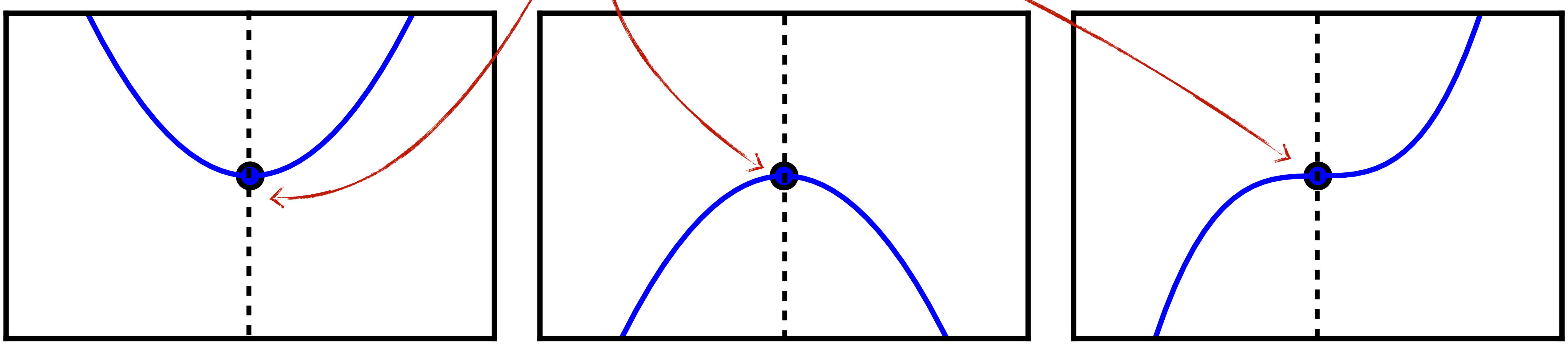


$$x' = x - \epsilon \nabla f(x)\,, \qquad \nabla f(x) = \left\{ \begin{matrix} \partial f / \partial x_1 \\ \partial f / \partial x_2 \\ \cdot \\ \cdot \\ \partial f / \partial x_n \end{matrix} \right\}$$

Critical Points

$$f'(x) = 0$$

Minimum | Maximum | Saddle point

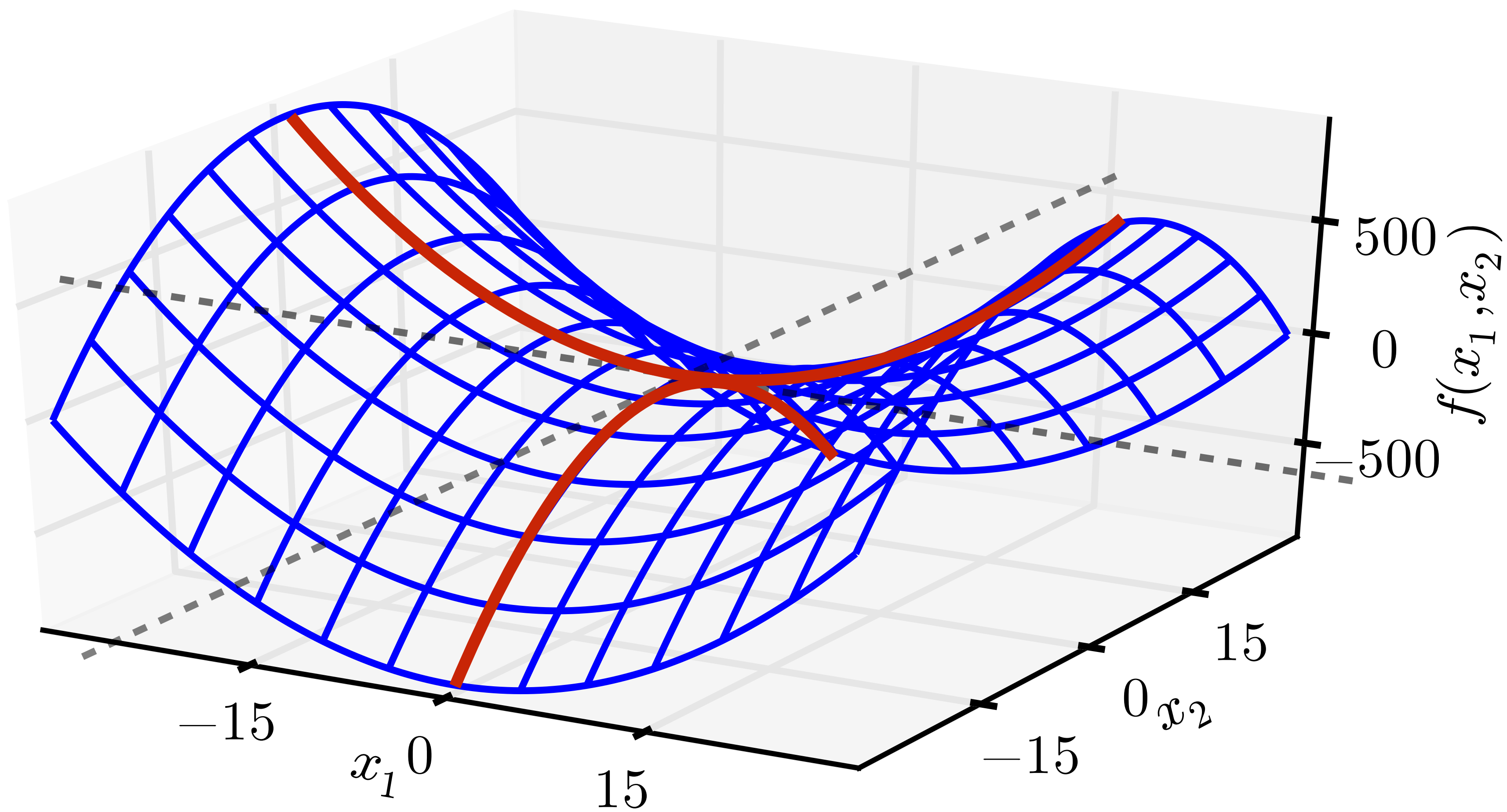


$x = x_o$ | $x = x_o$ | $x = x_o$

$x_o - \epsilon < x < x_o$ | $x_o < x < x_o + \epsilon$

$f''(x) > 0$ | $f''(x) < 0$ | $f''(x) < 0$ | $f''(x) > 0$

Saddle Points



<u>At point $(0,0)$:</u>

$$\frac{\partial f}{\partial x_1} = 0 \qquad \frac{\partial^2 f}{\partial x_1^2} > 0$$

$$\frac{\partial f}{\partial x_2} = 0 \qquad \frac{\partial^2 f}{\partial x_2^2} < 0$$

Jacobian Matrix

- **A system of equations:**

$$
\begin{aligned}
&f(x_1, x_2, ..., x_n) = b_1 \\
&f(x_1, x_2, ..., x_n) = b_2 \\
&... \\
&... \\
&f(x_1, x_2, ..., x_n) = b_n
\end{aligned}
$$

- **If equations are linear**

$$
\left.
\begin{aligned}
&a_{11}x_1 + a_{12}x_2 + ... + a_{1n}x_n = b_1 \\
&a_{21}x_1 + a_{22}x_2 + ... + a_{2n}x_n = b_2 \\
&... \\
&... \\
&a_{n1}x_1 + a_{n2}x_2 + ... + a_{nn}x_n = b_n
\end{aligned}
\right\} \Rightarrow \quad Ax = b
$$

Jacobian Matrix

- **non-linear equation**

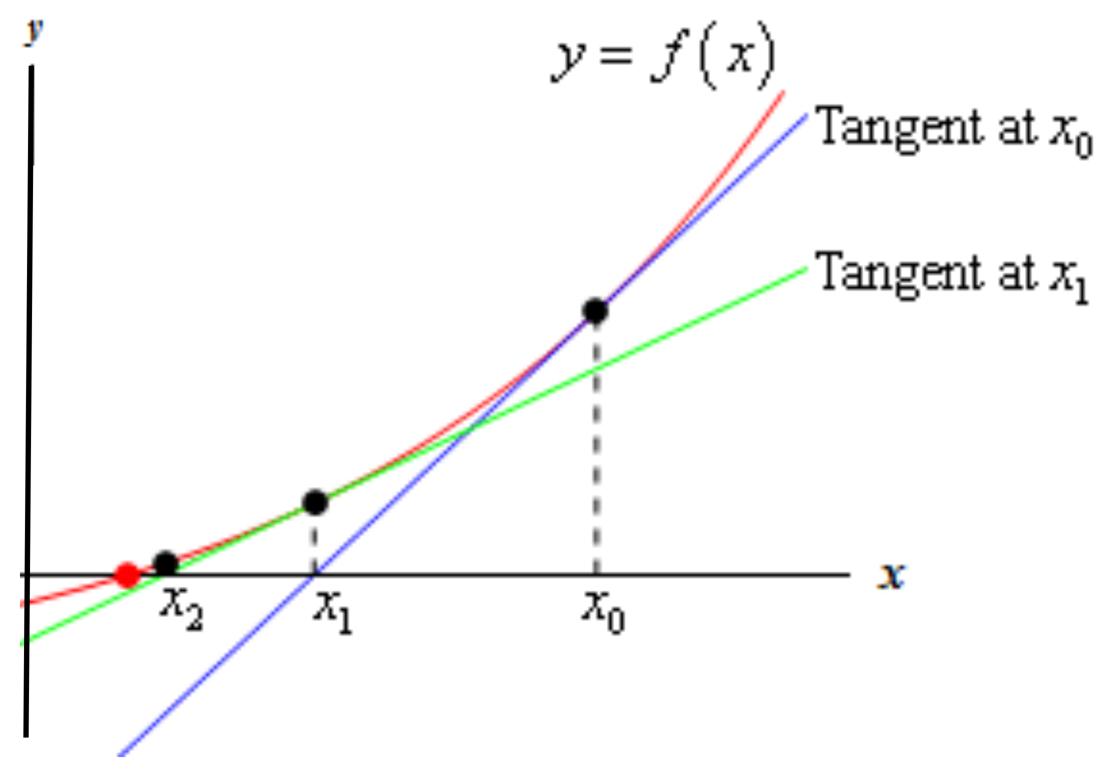
$$f(x) = b$$

- **Newton's Method (root finding) (n = 1)**

Solve for $f(x) = 0$

Solution schema:

1. initialize x
2. $\frac{\partial f}{\partial x}\Delta x = -f(x)$
3. $\boxed{\Delta x = -\frac{f(x)}{\partial f / \partial x}}$



Jacobian Matrix

- A system of non-linear equations:

$$
\begin{aligned}
&f(x_1, x_2, ..., x_n) = b_1 \\
&f(x_1, x_2, ..., x_n) = b_2 \\
&... \\
&... \\
&f(x_1, x_2, ..., x_n) = b_n
\end{aligned}
$$

- Newton's Method (root finding) (n > 1)

Solve for $f(\mathbf{x}) = 0$

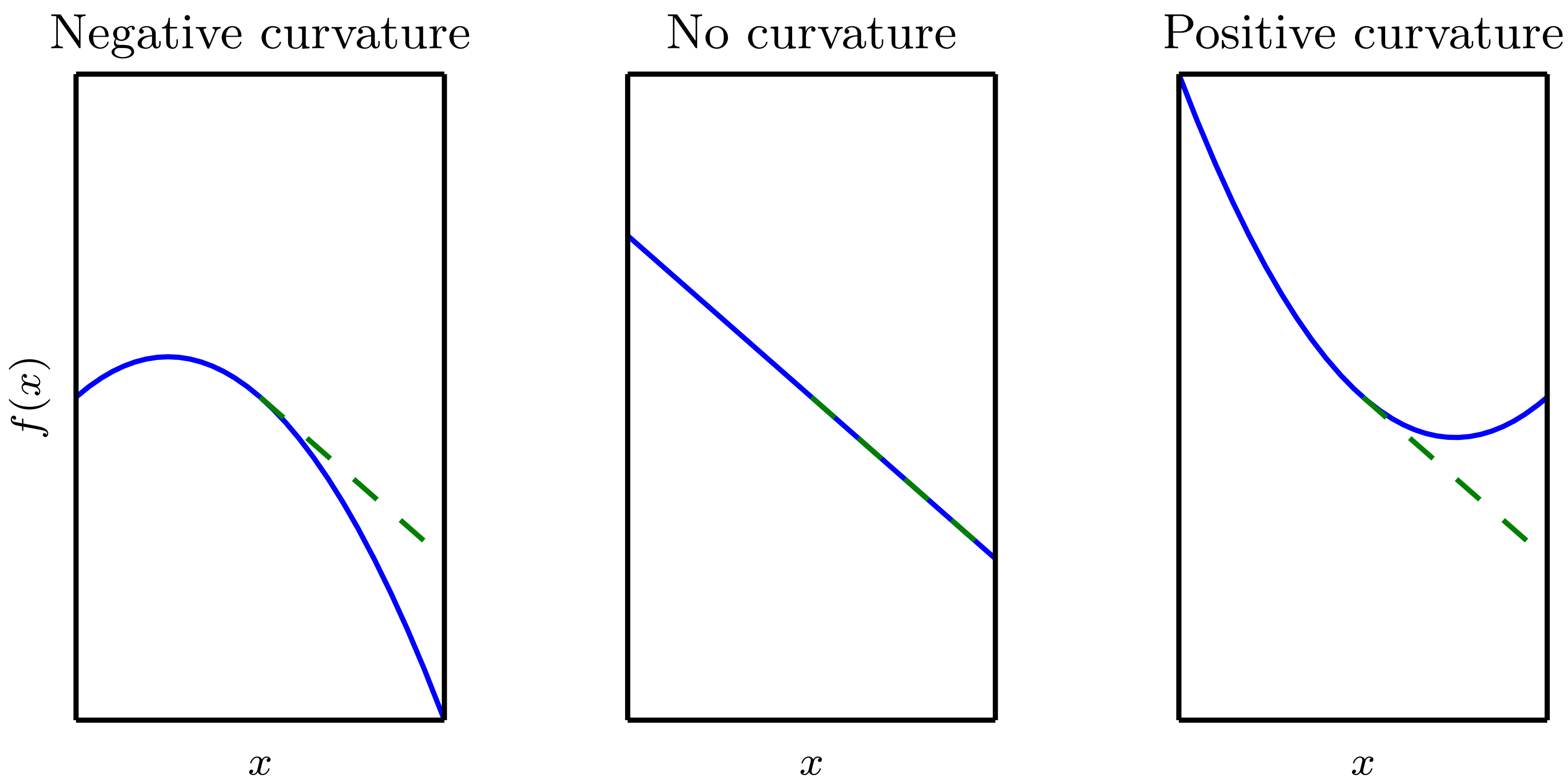
Solution schema:

1. initialize $\mathbf{x}$
2. $J\Delta\mathbf{x} = -f(\mathbf{x})$
3. $\boxed{\Delta\mathbf{x} = -J^{-1}f(\mathbf{x})}$

- $J = \left[\dfrac{\partial f_i}{\partial x_j}\right] = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & . & . & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_2} & \frac{\partial f_2}{\partial x_2} & . & . & \frac{\partial f_2}{\partial x_n} \\ . & . & . & . & . \\ \frac{\partial f_n}{\partial x_1} & \frac{\partial f_n}{\partial x_2} & . & . & \frac{\partial f_n}{\partial x_n} \end{bmatrix}$

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

Curvature



$f''(x) < 0$ $\qquad$ $f''(x) = 0$ $\qquad$ $f''(x) > 0$

- **1st order Taylor Series**

$$f(x) = f(x_o) + \Delta x f'(x_o) + O(\Delta x^2)$$

For n-dim input, $\mathbf{x}$:

$$f(\mathbf{x}_o + \Delta \mathbf{x}) = f(\mathbf{x_o}) + \mathbf{\Delta x}^T \nabla f(\mathbf{x_o}) + O(|\mathbf{\Delta x}|^2)$$

Hessian Matrix

- $f : \mathbb{R}^n \to \mathbb{R}$
- **Jacobian of the Gradient Operator**

$$H(\mathbf{x}) = J(\nabla f(\mathbf{x}))$$

$$\boxed{H(\mathbf{x}) = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x})}$$

- **Newton's Method (for optimization)**

Solve for $f'(x) = 0$, or $\nabla f(\mathbf{x}) = 0$

<u>Solution schema:</u>

1. initialize $\mathbf{x} = \mathbf{x}_o$
2. $H(\mathbf{x_o})\Delta\mathbf{x} = -\nabla f(\mathbf{x_o})$
3. $\boxed{\Delta\mathbf{x} = -\mathbf{H}^{-1}(\mathbf{x}_o)\ \nabla f(\mathbf{x}_o).}$

Hessian Matrix

- **2^{nd} order Taylor Series**

1-dim input, x:

$$f(x_o + \Delta x) \approx f(x_o) \; + \; \Delta x f'(x_o) \quad + \; \tfrac{1}{2}\, \Delta x^{\,2} f''(x_o)$$

n-dim input, $\mathbf{x}$:

$$f(\mathbf{x}_o + \Delta\mathbf{x}) \approx \; f(\mathbf{x}_o) + \; \Delta\mathbf{x}\nabla f(\mathbf{x}_o) \; + \; \tfrac{1}{2}\, \Delta\mathbf{x}^{\,T} H(\mathbf{x}_o)\Delta\mathbf{x}$$

Solve for $\Delta\mathbf{x}$, in $\nabla f(\mathbf{x}) = 0$

$$\boxed{\Delta\mathbf{x} = -\mathbf{H}^{-1}(\mathbf{x}_o)\; \nabla f(\mathbf{x}_o).}$$

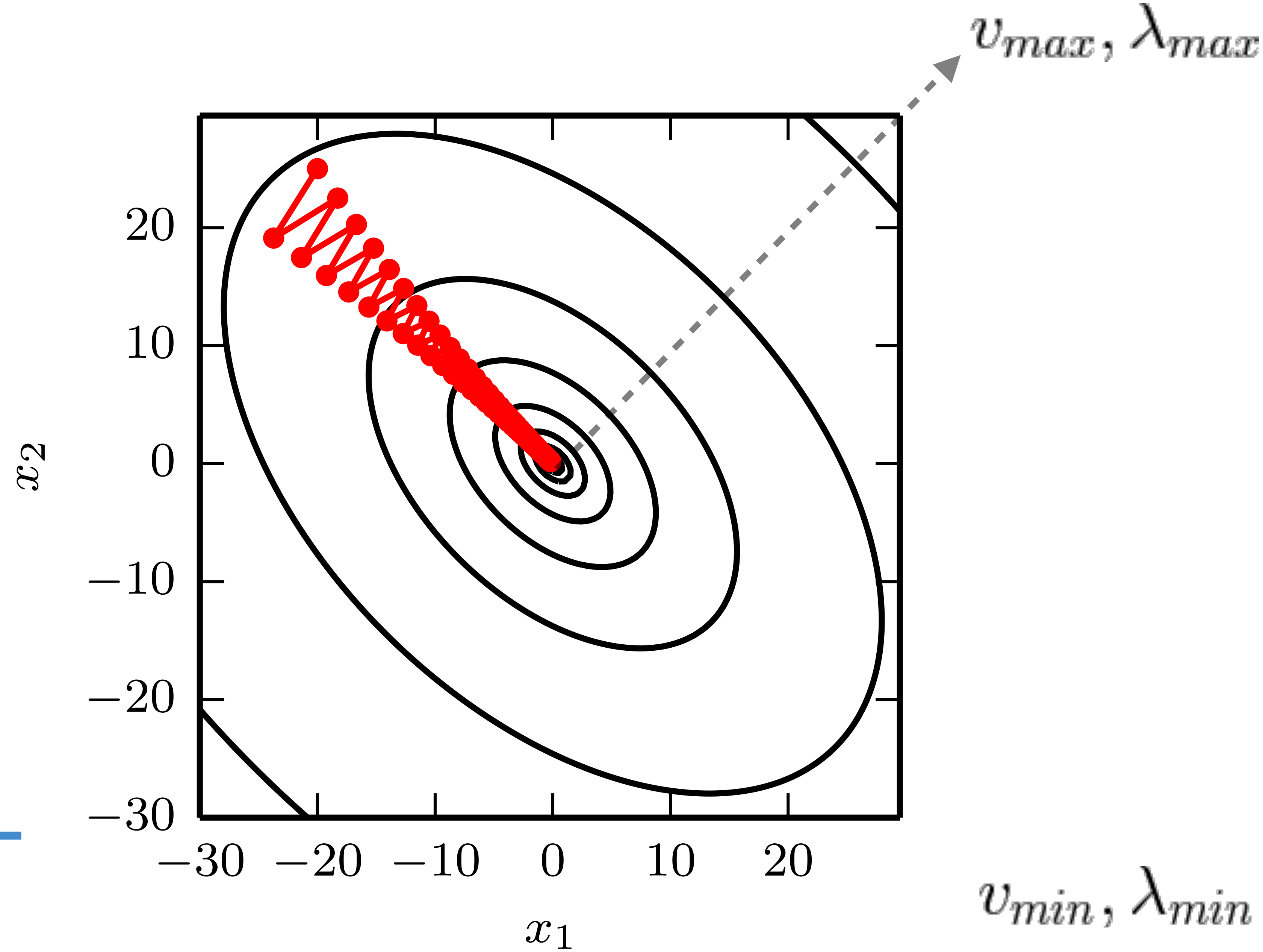
- **Optimum step-size in Gradient Descent**

$$f(\mathbf{x_o} - \epsilon\nabla f(\mathbf{x_o})) \approx f(\mathbf{x_o}) - \epsilon\nabla^T f(\mathbf{x_o})\nabla f(\mathbf{x_o}) + \frac{1}{2}\epsilon^2\nabla^T f(\mathbf{x_o}) H \nabla f(\mathbf{x_o})$$

$$\boxed{\epsilon^* = \frac{\nabla f(\mathbf{x}_o)^T\; \nabla f(\mathbf{x}_o)}{\nabla f(\mathbf{x}_o)^T\; H(\mathbf{x}_o)\; \nabla f(\mathbf{x}_o)}}$$

Gradient Descent and Poor Conditioning

$$Hv = \lambda v$$



- 2nd derivative in direction of $\mathbf{d}$.

$$\mathbf{d}^T H \ \mathbf{d}$$

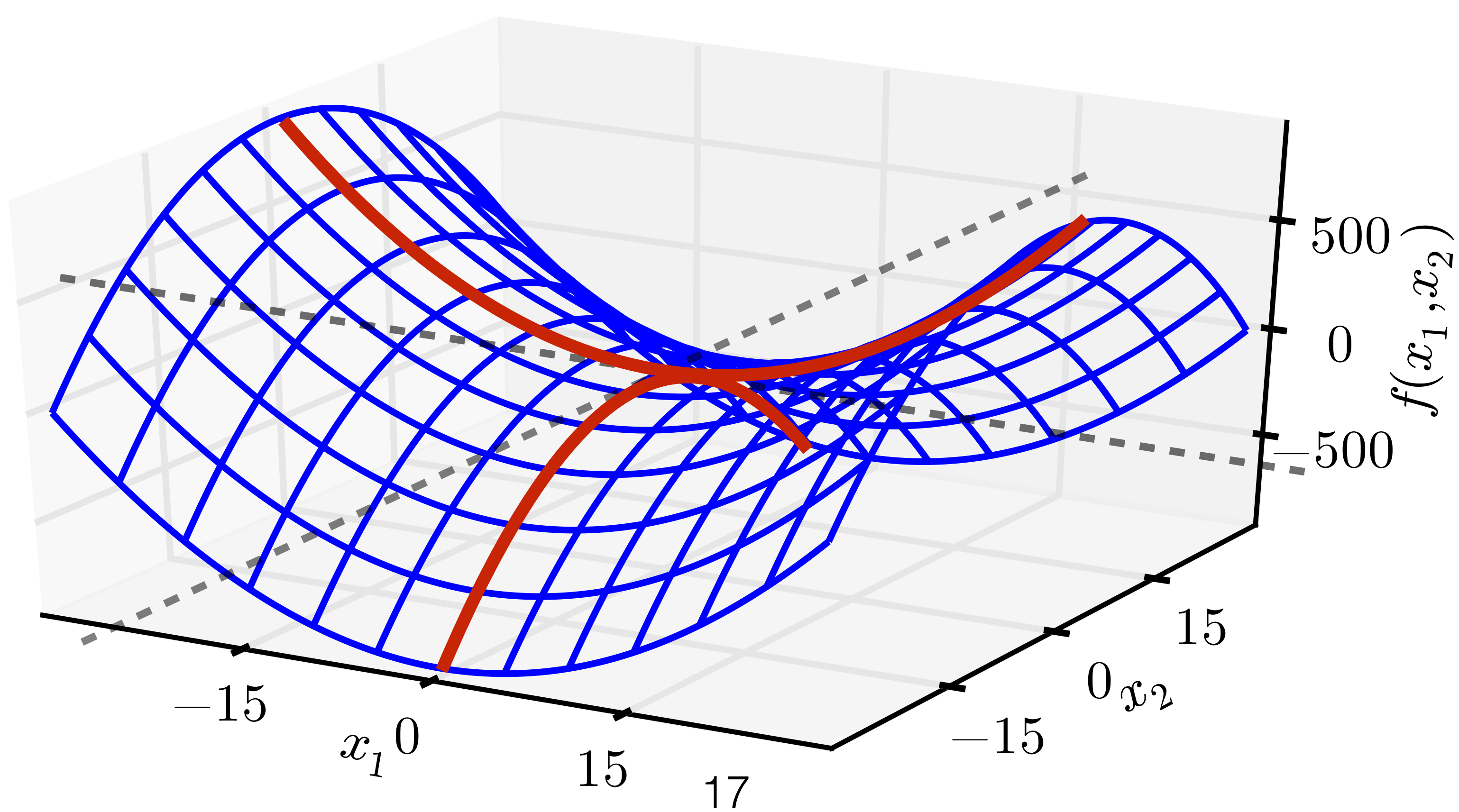
Safak Ozkan

Gradient Descent and Poor Conditioning

- $H(\mathbf{x}) = \frac{\partial^2}{\partial x_i \partial x_j} f(\mathbf{x})$

- $Hv = \lambda v$

  $\mathbf{x}^T H \mathbf{x} > 0$ ... H positive semi-definite
  $\mathbf{x}^T H \mathbf{x} < 0$ ... H negative semi-definite



Safak Ozkan

Optimization with Constraints and KKT Multipliers

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{\lambda}} \max_{\boldsymbol{\alpha}, \boldsymbol{\alpha} \geq 0} -f(\boldsymbol{x}) + \sum_i \lambda_i g^{(i)}(\boldsymbol{x}) + \sum_j \alpha_j h^{(j)}(\boldsymbol{x}).$$

Safak Ozkan