

Introduction to Optimization

Itthi Chatnuntawech

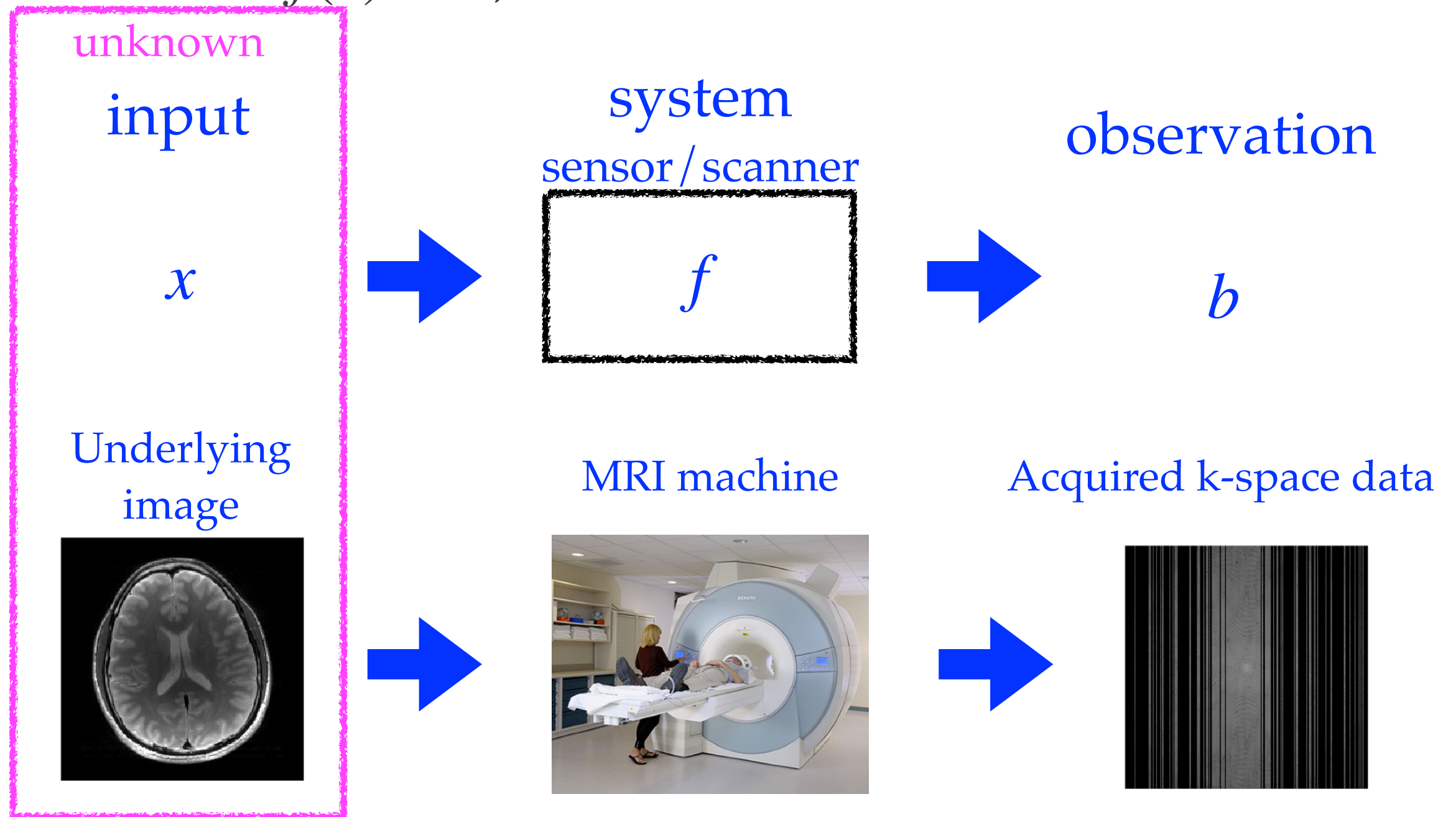
Nanoinformatics and Artificial Intelligence (NAI)

January 29, 2023

<https://github.com/ichatnun/MRI-simple-optimization-workshop>

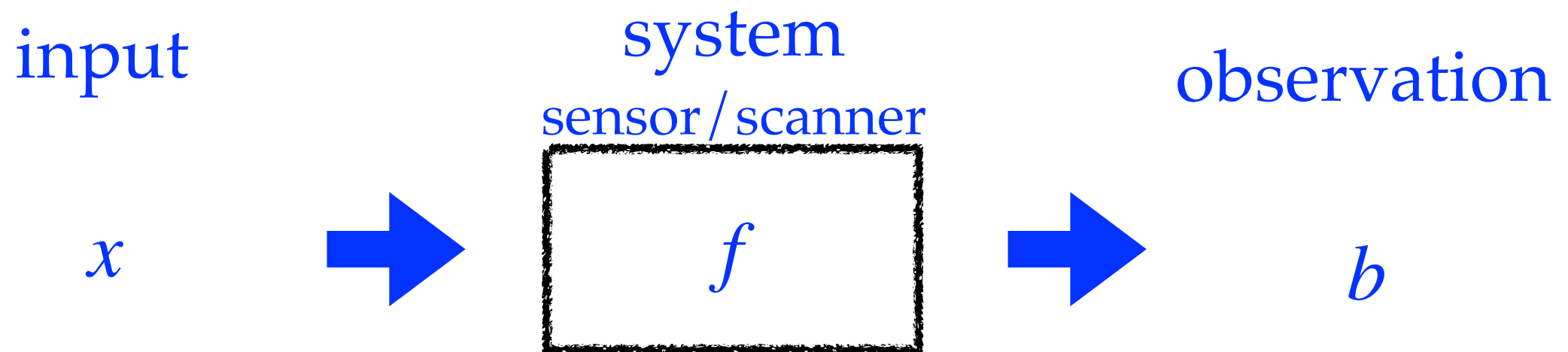
Optimization

- ❖ Goal: Given $f(x) = b$, recover x from b



Optimization

- ❖ Goal: Given $f(x) = b$, recover x from b



Idea 1: Randomly guess lots of x 's and pick the best one

What does it mean to be “the best one”?

For simplicity, we will use the Euclidean distance between $f(x)$ and b . Specifically, we want to minimize the following **loss function**

$$L(x) = ||f(x) - b||_2^2$$

Optimization

$$L(x) = ||f(x) - b||_2^2$$

Try x_1 $\xrightarrow{\text{apply } f \text{ to it}}$ $f(x_1)$

$\begin{bmatrix} 1 \\ 3 \\ 1 \end{bmatrix}$ $\begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}$ acquired / observed data

$$L(x_1) = ||f(x_1) - b||_2^2 = (1 - 2)^2 + (3 - 3)^2 + (1 - 0)^2 = 2$$

Optimization

$$L(x) = ||f(x) - b||_2^2$$

$$\begin{bmatrix} 1 \\ 3 \\ 0 \end{bmatrix} \quad \begin{bmatrix} 2 \\ 3 \\ 0 \end{bmatrix}$$

acquired / observed data

Try
 x_2

apply f to it



$f(x_2)$

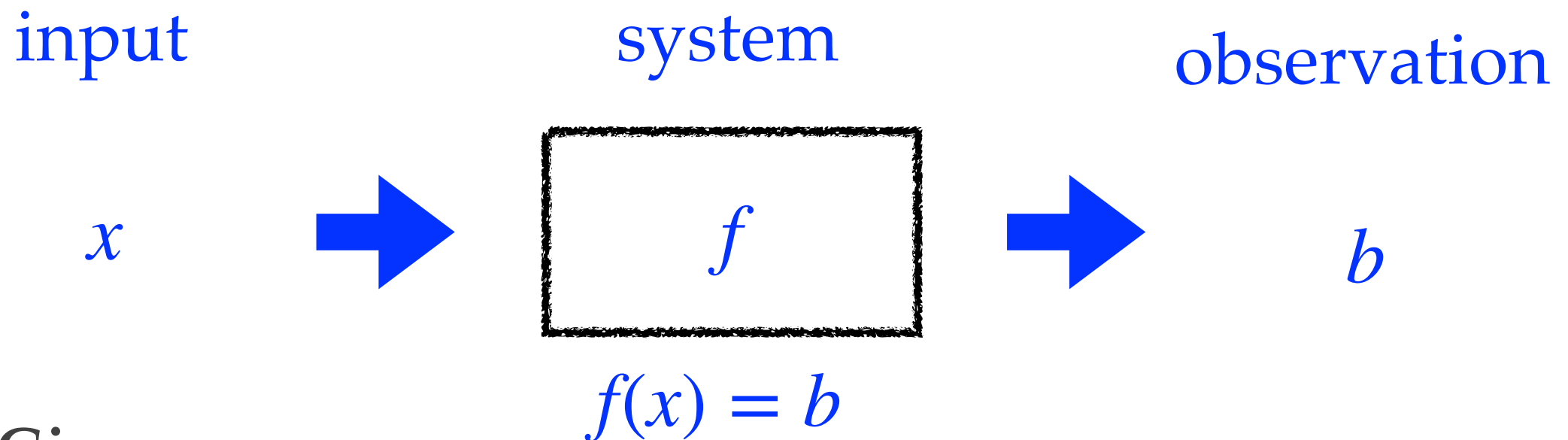
$$L(x_1) = ||f(x_1) - b||_2^2 = (1 - 2)^2 + (3 - 3)^2 + (1 - 0)^2 = 2$$

$$L(x_2) = ||f(x_2) - b||_2^2 = (1 - 2)^2 + (3 - 3)^2 + (0 - 0)^2 = 1$$



Better guess

Example 1: Random guess



Given

- ❖ The observation b
- ❖ The `applyF()` function, which computes $f(x)$ whenever an x is given
 $applyF(x) \rightarrow f(x)$
- ❖ The `loss()` function, which computes the difference between the given two vectors
 $loss(b_1, b_2) \rightarrow ||b_1 - b_2||_2^2$

Goal: Try to recover x using the code provided in `ex1.m`

Hint: Each entry of the true x has the value between 0 and 1

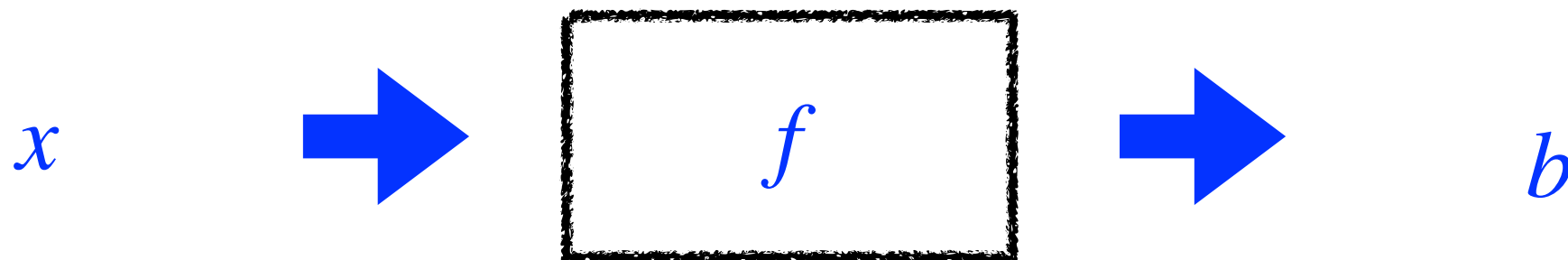
Optimization

- ❖ Goal: Given $f(x) = b$, recover x from b

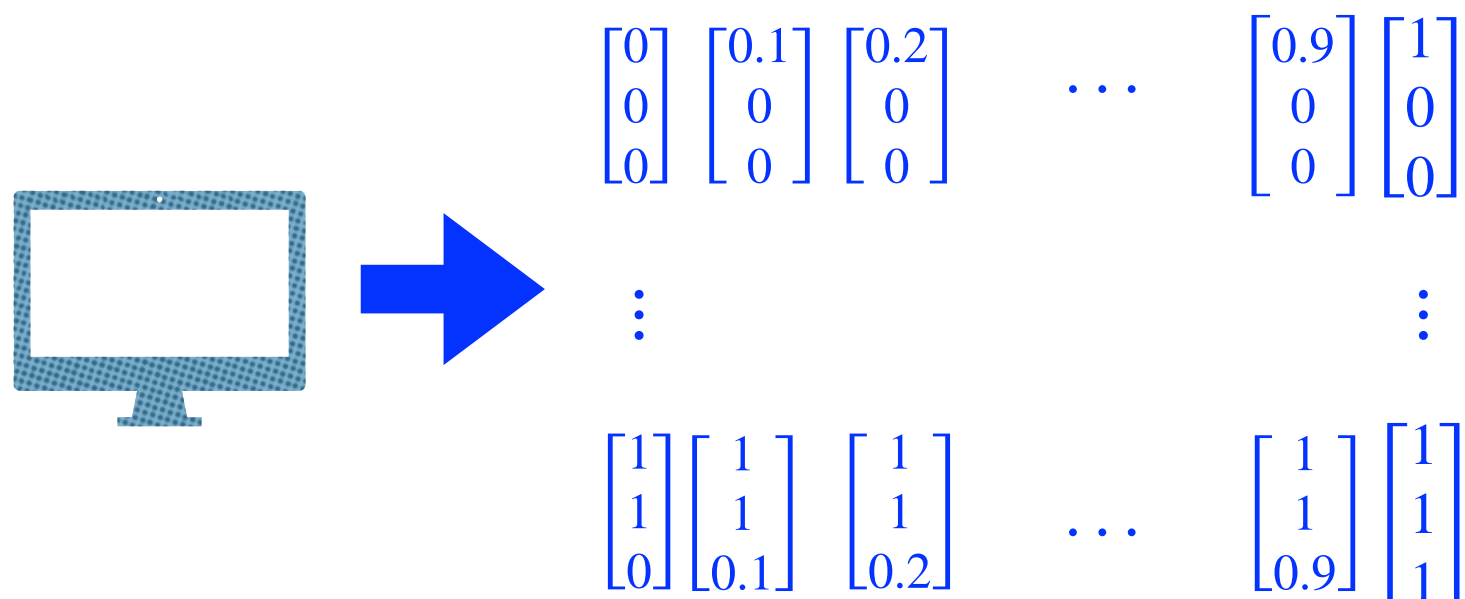
input

system

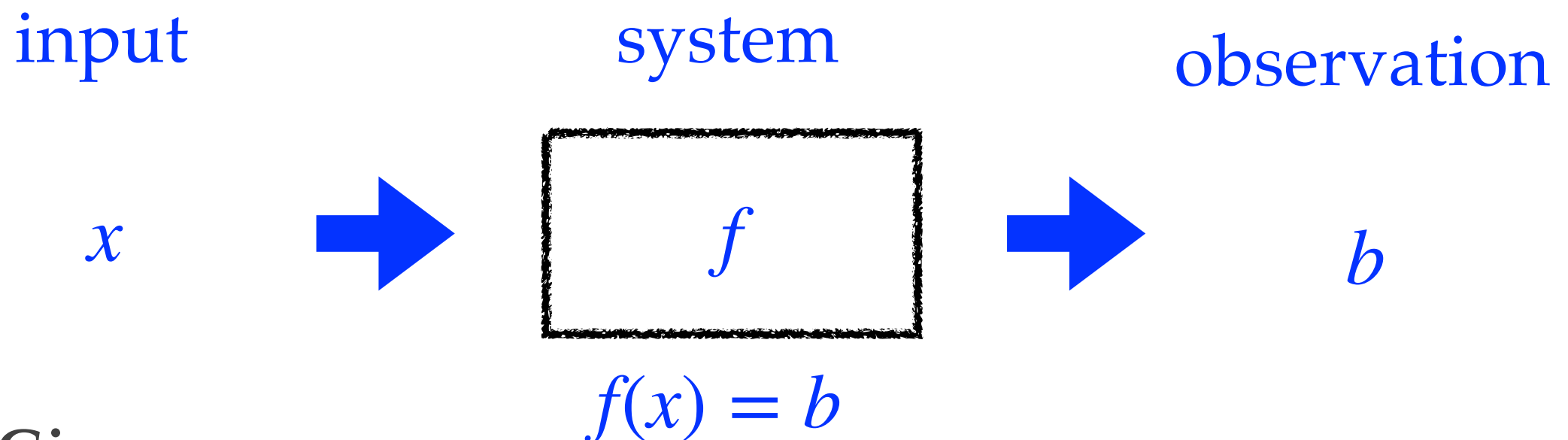
observation



Idea 2: Write a MATLAB program to generate lots of x 's in a grid search manner and automatically pick the best one



Example 2: Grid Search



Given

- ❖ The observation b
- ❖ The `applyF()` function, which computes $f(x)$ whenever an x is given
 $applyF(x) \rightarrow f(x)$
- ❖ The `loss()` function, which computes the difference between the given two vectors
 $loss(b_1, b_2) \rightarrow ||b_1 - b_2||_2^2$

Goal: Try to recover x using the code provided in `ex2.m`

Hint: Each entry of the true x has the value between 0 and 1

Optimization

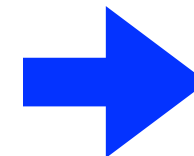
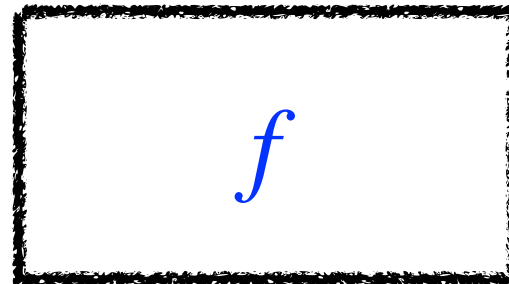
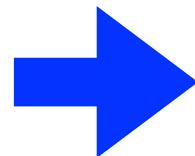
- ❖ Goal: Given $f(x) = b$, recover x from b

input

system

observation

x



b

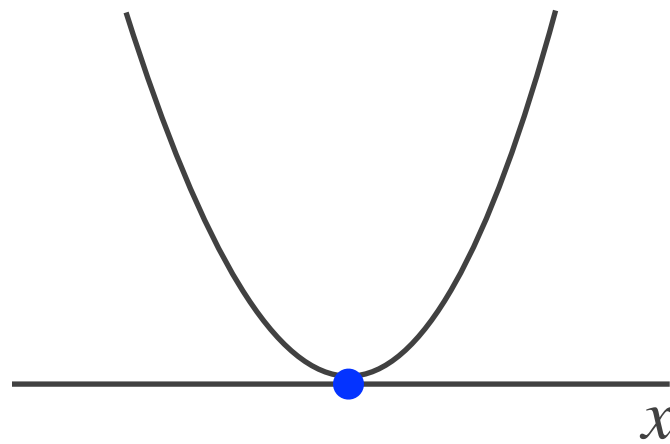
Guessing the solution is not very efficient...

Idea 3: Gradient descent

Gradient Descent

- ❖ Goal: Find x that minimizes the following loss function

$$L(x) = x^2$$



Method 1: Compute the derivative and set it to 0

Gradient $\frac{dL(x)}{dx} = \frac{dx^2}{dx} = 2x = 0 \quad \rightarrow \quad x = 0$

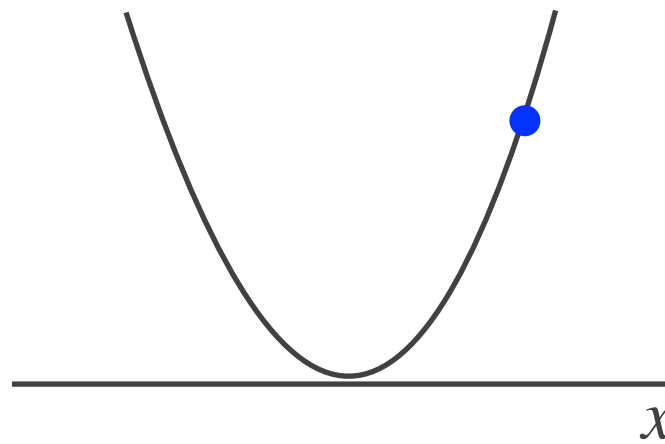
In many cases, it is not simple to compute the gradient with respect to x .

Even when we have a way to compute the gradient and manage to set it to zero, we might not be able to get a closed-form solution for it.

Gradient Descent

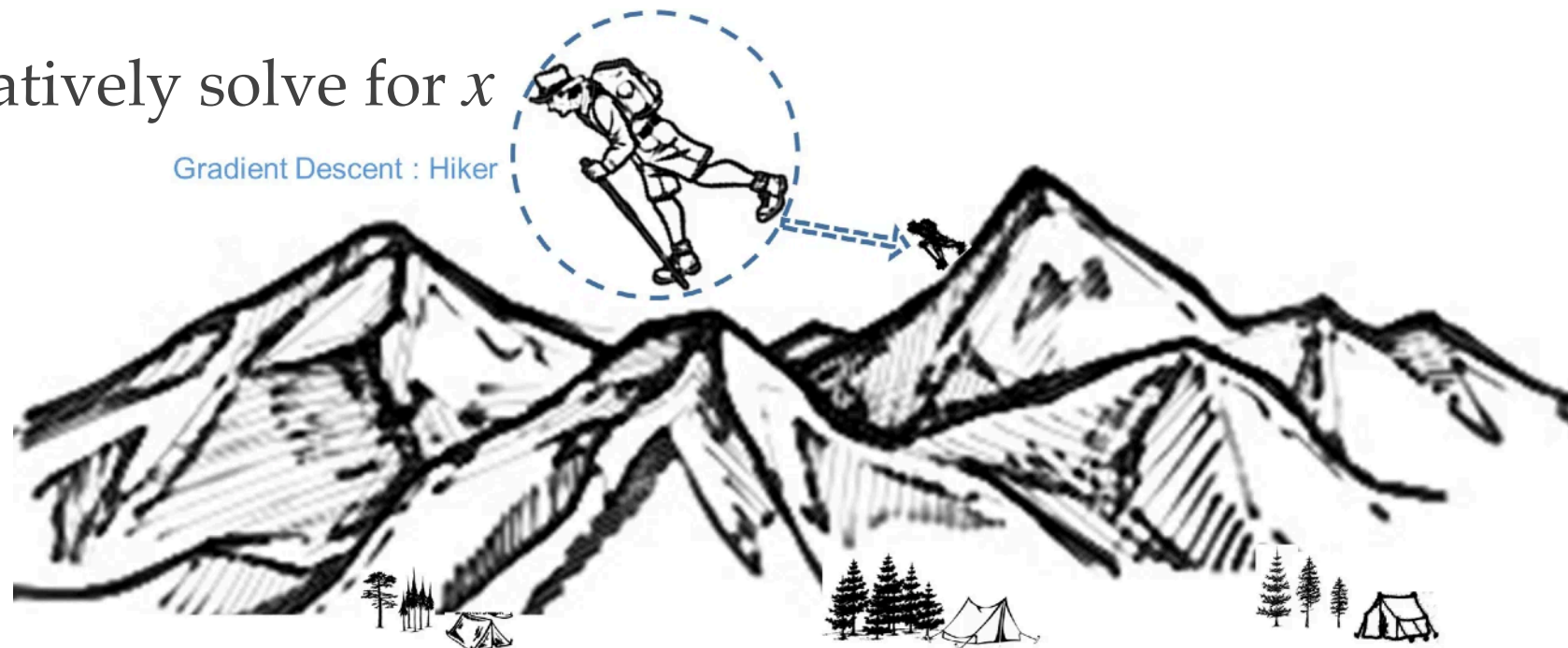
❖ Goal: Find x that minimizes the following loss function

$$L(x) = x^2$$



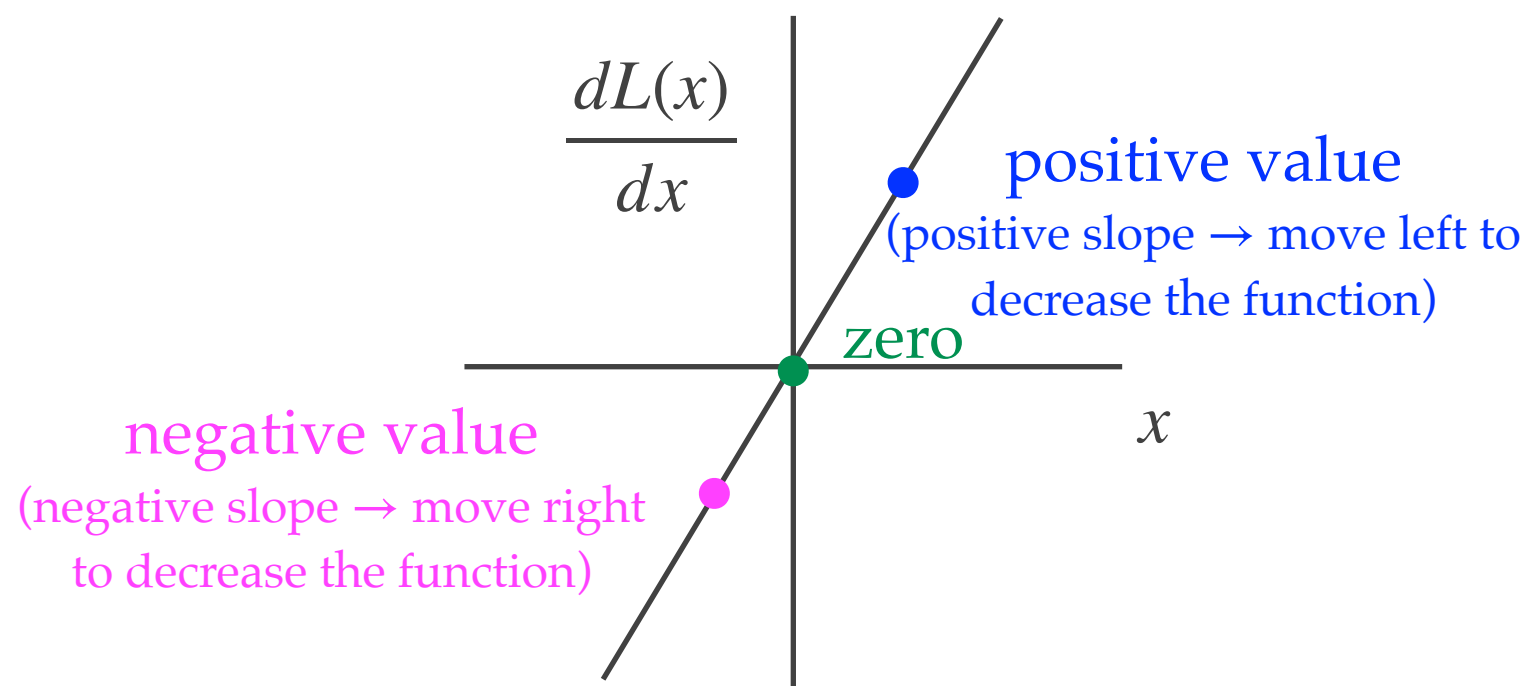
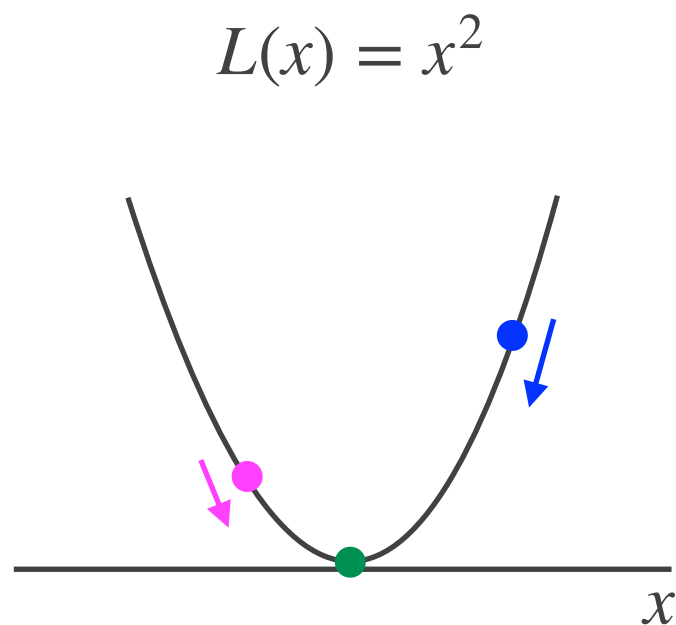
- Randomly start somewhere
- Gradually move to the minimum of the function

Method 2: Iteratively solve for x



Gradient Descent

❖ Goal: Find x that minimizes the following loss function



Method 2: Iteratively solve for x

1. Guess an initial solution x_0 and pick α
 For $k = 0, 1, 2, \dots$

2. Compute $\frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = 2x \Big|_{x=x_{(k)}} = 2x_{(k)}$

3. Compute a new solution $x_{(k+1)} := x_{(k)} - \alpha \frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = x_{(k)} - \alpha * 2x_{(k)}$

4. Repeat step 2 and 3 until converge

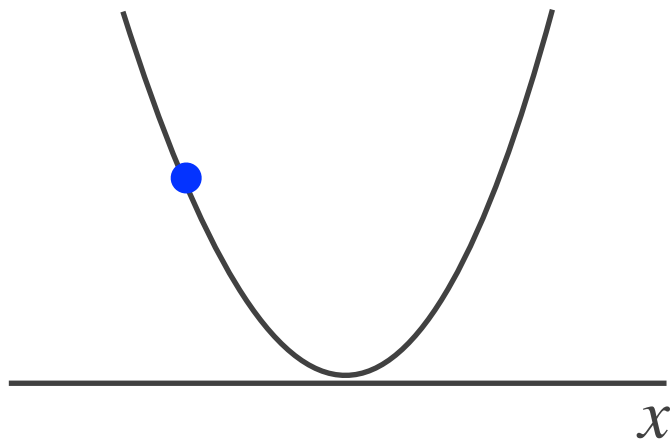
The negative of the gradient $-\frac{dL(x)}{dx}$ tells us the direction we should move to decrease the function

step size/learning rate α indicates how far we want to move

Gradient Descent

❖ Goal: Find w that minimizes the following loss function

$$L(x) = x^2$$



$$x_{(0)} = -2, \alpha = 0.5$$

$$x_{(1)} = x_{(0)} - \alpha * 2x_{(0)} = -2 - 0.5 * 2 * -2 = 0$$

$$x_{(2)} = x_{(1)} - \alpha * 2x_{(1)} = 0$$

$$x_{(3)} = x_{(2)} - \alpha * 2x_{(2)} = 0$$

Method 2: Iteratively solve for x

1. Guess an initial solution x_0 and pick α

For $k = 0, 1, 2, \dots$

2. Compute $\frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = 2x \Big|_{x=x_{(k)}} = 2x_{(k)}$

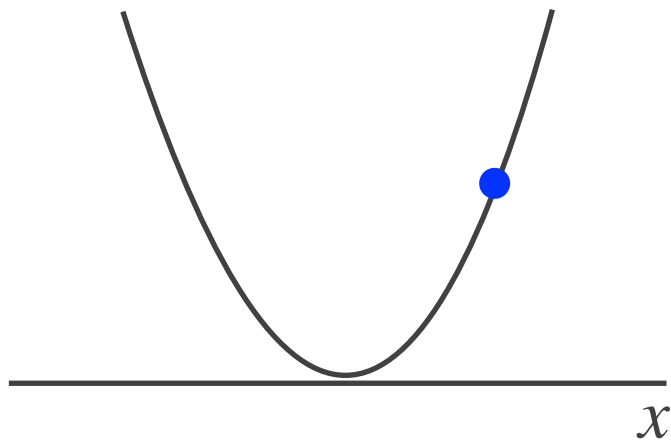
3. Compute a new solution $x_{(k+1)} := x_{(k)} - \alpha \frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = x_{(k)} - \alpha * 2x_{(k)}$

4. Repeat step 2 and 3 until converge

Gradient Descent

❖ Goal: Find w that minimizes the following loss function

$$L(x) = x^2$$



$$x_{(0)} = 2, \alpha = 0.5$$

$$x_{(1)} = x_{(0)} - \alpha * 2x_{(0)} = 2 - 0.5 * 2 * 2 = 0$$

$$x_{(2)} = x_{(1)} - \alpha * 2x_{(1)} = 0$$

$$x_{(3)} = x_{(2)} - \alpha * 2x_{(2)} = 0$$

Method 2: Iteratively solve for x

1. Guess an initial solution x_0 and pick α

For $k = 0, 1, 2, \dots$

2. Compute $\frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = 2x \Big|_{x=x_{(k)}} = 2x_{(k)}$

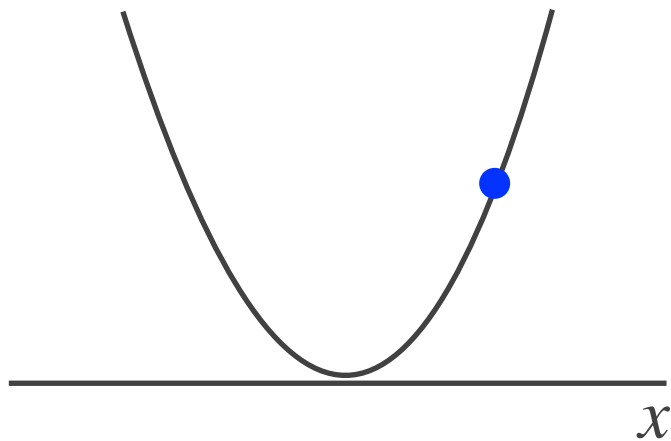
3. Compute a new solution $x_{(k+1)} := x_{(k)} - \alpha \frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = x_{(k)} - \alpha * 2x_{(k)}$

4. Repeat step 2 and 3 until converge

Gradient Descent

❖ Goal: Find w that minimizes the following loss function

$$L(x) = x^2$$



$$x_{(0)} = 2, \alpha = 0.25$$

$$x_{(1)} = x_{(0)} - \alpha * 2x_{(0)} = 2 - 0.25 * 2 * 2 = 1$$

$$x_{(2)} = x_{(1)} - \alpha * 2x_{(1)} = 1 - 0.25 * 2 * 1 = 0.5$$

$$x_{(3)} = x_{(2)} - \alpha * 2x_{(2)} = 0.5 - 0.25 * 2 * 0.5 = 0.25$$

$$x_{(4)} = x_{(3)} - \alpha * 2x_{(3)} = 0.125$$

$$x_{(5)} = x_{(4)} - \alpha * 2x_{(4)} = 0.0625$$

Method 2: Iteratively solve for x

1. Guess an initial solution x_0 and pick α

For $k = 0, 1, 2, \dots$

2. Compute $\frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = 2x \Big|_{x=x_{(k)}} = 2x_{(k)}$

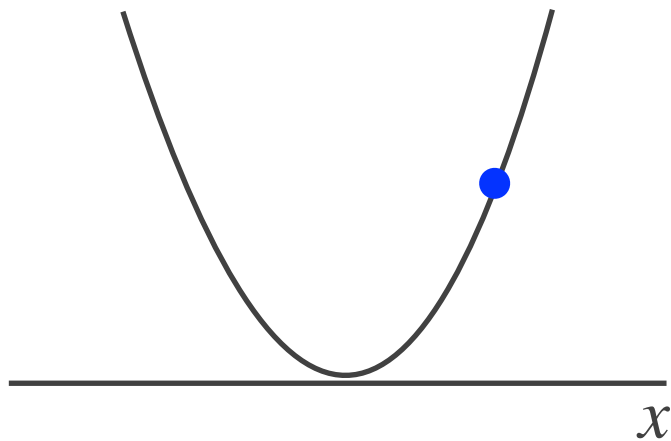
3. Compute a new solution $x_{(k+1)} := x_{(k)} - \alpha \frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = x_{(k)} - \alpha * 2x_{(k)}$

4. Repeat step 2 and 3 until converge

Gradient Descent

❖ Goal: Find w that minimizes the following loss function

$$L(x) = x^2$$



$$x_{(0)} = 2, \alpha = 1$$

$$x_{(1)} = x_{(0)} - \alpha * 2x_{(0)} = 2 - 1 * 2 * 2 = -2$$

$$x_{(2)} = x_{(1)} - \alpha * 2x_{(1)} = -2 - 1 * 2 * -2 = 2$$

$$x_{(3)} = x_{(2)} - \alpha * 2x_{(2)} = -2$$

$$x_{(4)} = x_{(3)} - \alpha * 2x_{(3)} = 2$$

$$x_{(5)} = x_{(4)} - \alpha * 2x_{(4)} = -2$$

Method 2: Iteratively solve for x

1. Guess an initial solution x_0 and pick α

For $k = 0, 1, 2, \dots$

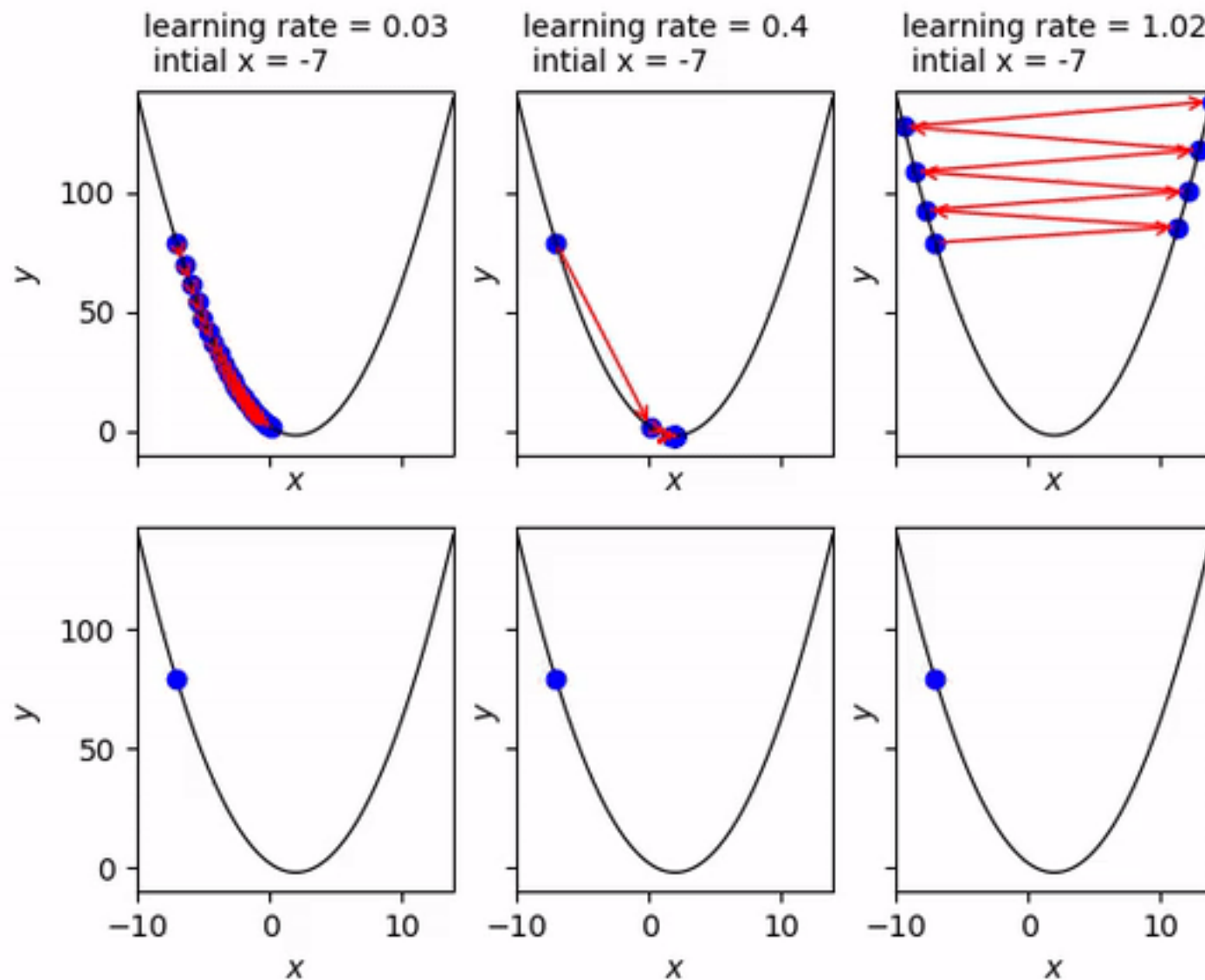
2. Compute $\frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = 2x \Big|_{x=x_{(k)}} = 2x_{(k)}$

3. Compute a new solution $x_{(k+1)} := x_{(k)} - \alpha \frac{dL(x)}{dx} \Big|_{x=x_{(k)}} = x_{(k)} - \alpha * 2x_{(k)}$

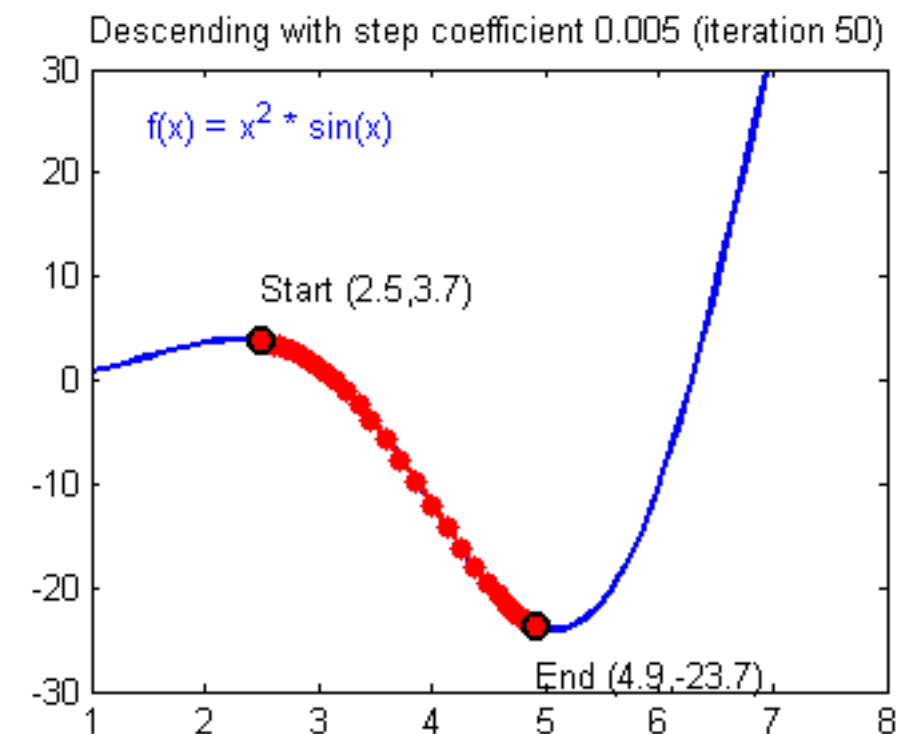
4. Repeat step 2 and 3 until converge

α : learning rate

Gradient Descent: Learning Rate



Fixed learning rate

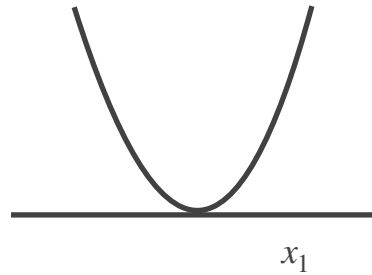


Adaptive learning rate

Gradient

1-dimensional case

$$L(x_1) = x_1^2$$



Derivative of $L(x_1)$

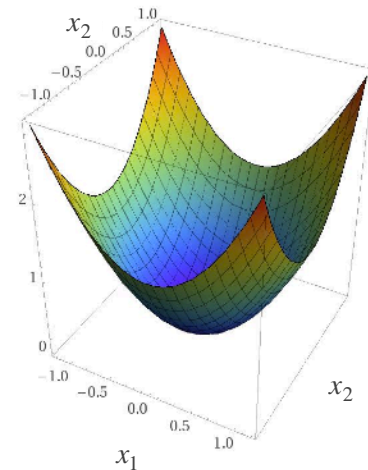
$$\frac{dL(x_1)}{dx_1} = \frac{dx_1^2}{dx_1} = 2x_1$$

Weight update

$$\begin{aligned} x_{1,(k+1)} &:= x_{1,(k)} - \alpha \frac{dL(x_1)}{dx_1} \Big|_{x_1=x_{1,(k)}} \\ &= x_{1,(k)} - 2x_{1,(k)} \end{aligned}$$

2-dimensional case

$$L(x_1, x_2) = x_1^2 + x_2^2$$



Partial derivatives of $L(x_1, x_2)$

$$\frac{\partial L(x_1, x_2)}{\partial x_1} = \frac{\partial}{\partial x_1}(x_1^2 + x_2^2) = 2x_1$$

$$\frac{\partial L(x_1, x_2)}{\partial x_2} = \frac{\partial}{\partial x_2}(x_1^2 + x_2^2) = 2x_2$$

Gradient $\nabla_x L(x_1, x_2)$

Weight update

$$\begin{aligned} \begin{bmatrix} x_{1,(k+1)} \\ x_{2,(k+1)} \end{bmatrix} &= \begin{bmatrix} x_{1,(k)} \\ x_{2,(k)} \end{bmatrix} - \alpha \begin{bmatrix} \frac{\partial L(x_1, x_2)}{\partial x_1} \Big|_{x_1=x_{1,(k)}} \\ \frac{\partial L(x_1, x_2)}{\partial x_2} \Big|_{x_2=x_{2,(k)}} \end{bmatrix} \\ &= \begin{bmatrix} x_{1,(k)} \\ x_{2,(k)} \end{bmatrix} - \alpha \begin{bmatrix} 2x_{1,(k)} \\ 2x_{2,(k)} \end{bmatrix} \end{aligned}$$

Optimization

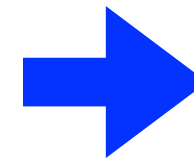
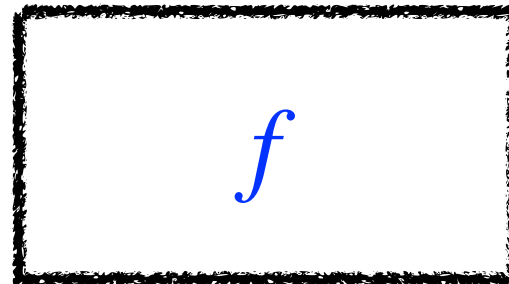
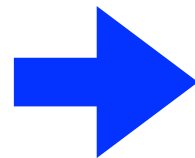
- ❖ Goal: Given $f(x) = b$, recover x from b

input

system

observation

x



b

Guessing the solution is not very efficient...

Idea 3: Gradient descent

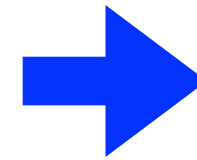
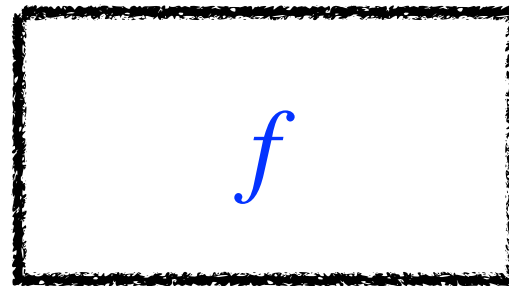
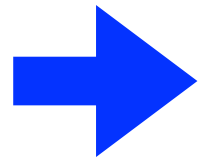
Example 3: Gradient Descent

input

system

observation

x



b

Given

- ❖ The observation b
- ❖ The `applyF()` function, which computes $f(x)$ whenever an x is given
- ❖ The `loss()` function, which computes the difference between the given two vectors
- ❖ The `computeGradient()` function, which computes the gradient of the loss function: $L(x) = ||f(x) - b||_2^2$

Goal: Try to recover x using the code provided in [ex3.m](#)

Hints

- ❖ If we take a look at the code, we will see that $f(x) = Ax$
- ❖ The gradient of the loss function has a nice form:
$$\nabla_x L(x) = \nabla_x ||f(x) - b||_2^2 = \nabla_x ||Ax - b||_2^2 = 2A^T(Ax - b)$$
- ❖ The update equation becomes

$$x_{(k+1)} := x_{(k)} - \alpha \times 2A^T(Ax_{(k)} - b)$$

Linear System

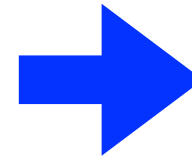
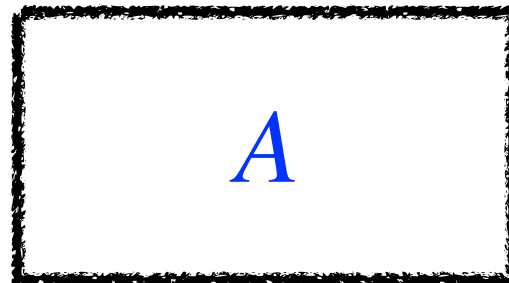
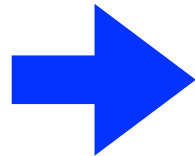
- ❖ Goal: Given $Ax = b$, recover x from b

input

system

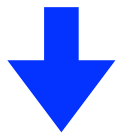
observation

x



b

$$\begin{aligned} 1.3 &= x_1 + 0 \cdot x_2 + 4x_3 \\ 1.5 &= 0.2x_1 + 3x_2 + x_3 \\ 0.4 &= 0 \cdot x_1 + x_2 + 0 \cdot x_3 \end{aligned}$$



$$\begin{bmatrix} 1.3 \\ 1.5 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 4 \\ 0.2 & 3 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\downarrow \min_x \|Ax - b\|_2^2$$

Unique solution

$$x = A^{-1}b = \begin{bmatrix} 0.5 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Underdetermined system

$$\begin{aligned} 1.3 &= x_1 + 0 \cdot x_2 + 4x_3 \\ 1.5 &= 0.2x_1 + 3x_2 + x_3 \\ 0.4 &= 0 \cdot x_1 + x_2 + 0 \cdot x_3 \end{aligned}$$



$$[1.3] = [1 \ 0 \ 4] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\downarrow \min_x \|Ax - b\|_2^2$$

Possible solutions

$$\begin{bmatrix} 0.5 \\ 0.4 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 0.0765 \\ 0.0 \\ 0.3509 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0.325 \end{bmatrix}, \dots$$

Example 4: Confirm that the results shown here are accurate using [ex4.m](#)

$$\begin{aligned} 1.3 &= x_1 + 0 \cdot x_2 + 4x_3 \\ 1.5 &= 0.2x_1 + 3x_2 + x_3 \\ 0.4 &= 0 \cdot x_1 + x_2 + 0 \cdot x_3 \end{aligned}$$



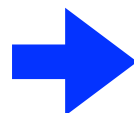
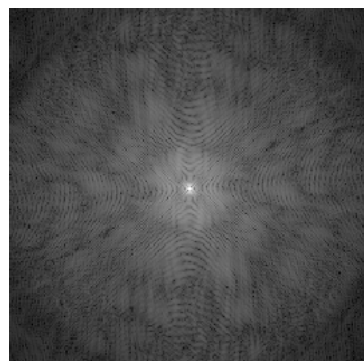
$$\begin{bmatrix} 1.3 \\ 1.5 \\ 0.4 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 4 \\ 0.2 & 3 & 1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\downarrow \min_x \|Ax - b\|_2^2$$

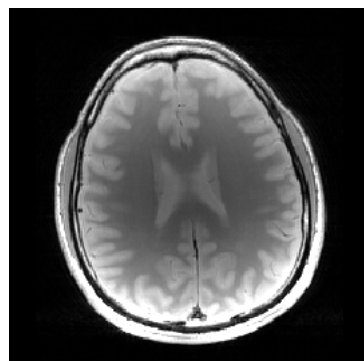
Unique solution

$$x = A^{-1}b = \begin{bmatrix} 0.5 \\ 0.4 \\ 0.2 \end{bmatrix}$$

Fully-sampled

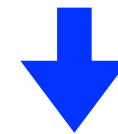


Unique solution



Underdetermined system

$$\begin{aligned} 1.3 &= x_1 + 0 \cdot x_2 + 4x_3 \\ 1.5 &= 0.2x_1 + 3x_2 + x_3 \\ 0.4 &= 0 \cdot x_1 + x_2 + 0 \cdot x_3 \end{aligned}$$



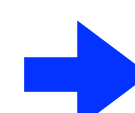
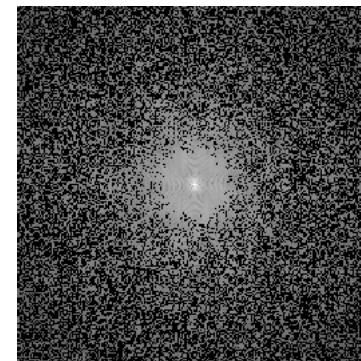
$$[1.3] = [1 \ 0 \ 4] \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

$$\downarrow \min_x \|Ax - b\|_2^2$$

Possible solutions

$$\begin{bmatrix} 0.5 \\ 0.4 \\ 0.2 \end{bmatrix}, \begin{bmatrix} 0.0765 \\ 0.0 \\ 0.3509 \end{bmatrix}, \begin{bmatrix} 0 \\ 0 \\ 0.325 \end{bmatrix}, \dots$$

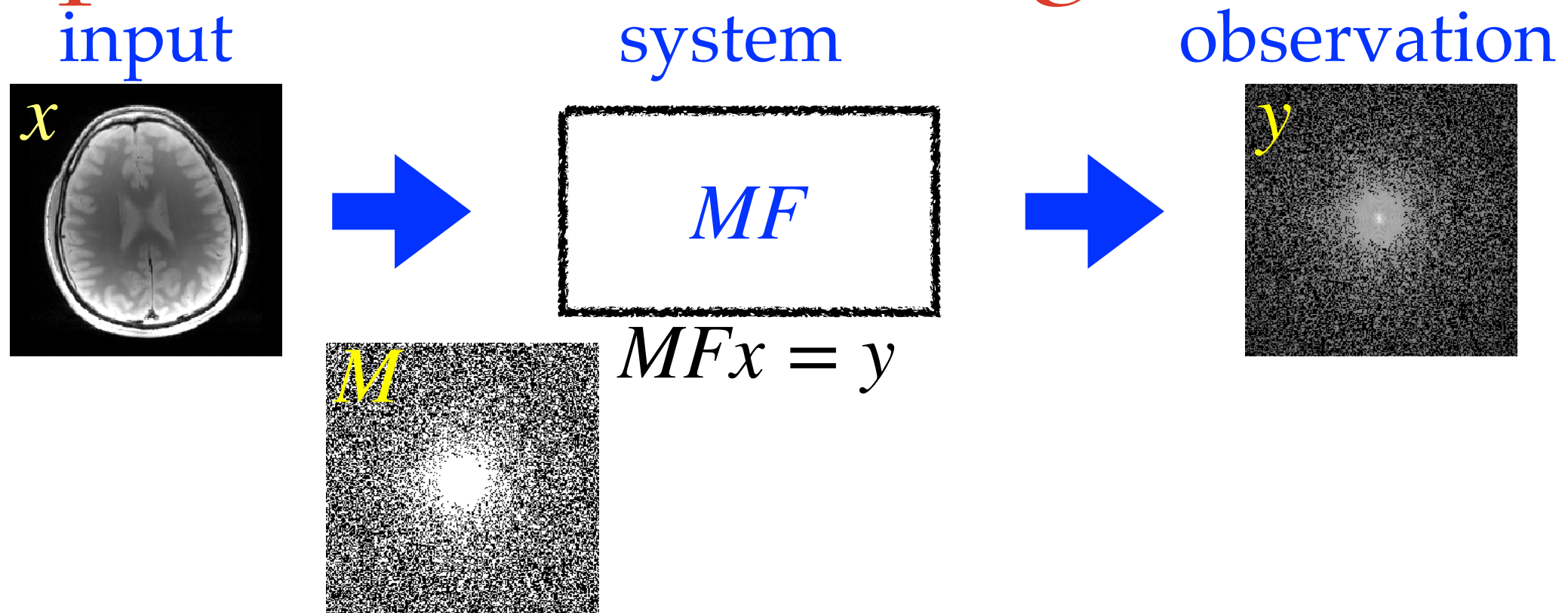
Undersampled



one possible solution



Example 5: Tikhonov Regularization



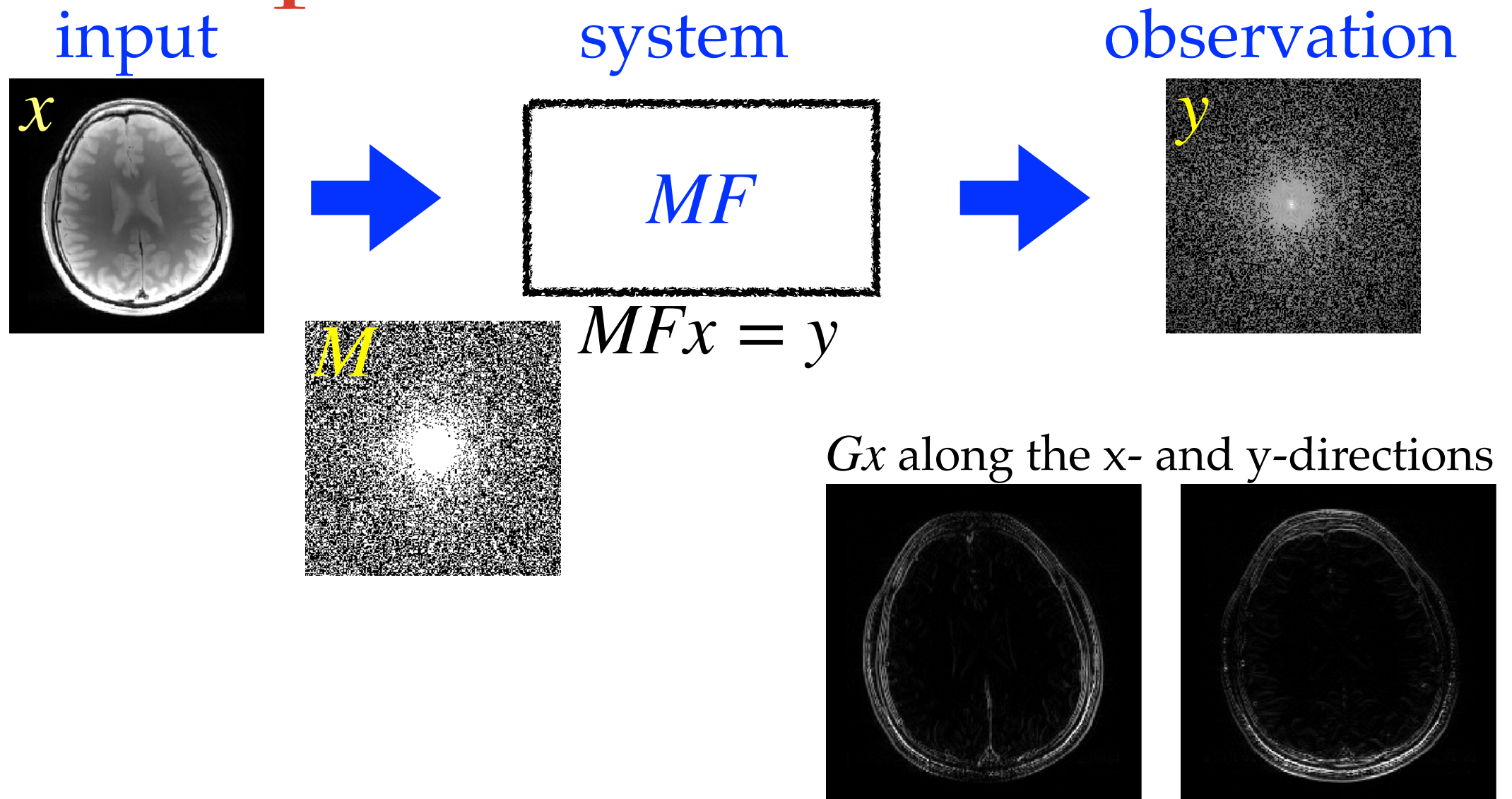
- ❖ Approach: Solve $\min_x L(x) = \min_x \|MFx - y\|_2^2 + \lambda \|x\|_2^2$ using gradient descent (ex4.m), enforcing the L2-norm on the image
- ❖ The update equation:

$$x_{(k+1)} := x_{(k)} - \alpha \times \underbrace{[2F^T M^T (MFx_{(k)} - y) + 2\lambda x_{(k)}]}_{\text{gradient of } L(x)}$$

Notes

- ❖ Try adjusting the parameters in ex5.m such as
 - ❖ The regularization parameter λ
 - ❖ $\lambda = 0 \rightarrow$ No regularization (under-determined system of equations)
 - ❖ The learning rate α
 - ❖ The reduction factor R

Example 6: Total Variation



- ❖ Approach: Solve $\min_x L(x) = \min_x \|MFx - y\|_2^2 + \lambda \|Gx\|_1$ using (sub)gradient descent ([ex6.m](#)), enforcing sparsity on the image in the finite difference domain (Gx)
- ❖ The update equation:

$$x_{(k+1)} := x_{(k)} - \alpha \times [2F^T M^T (MFx_{(k)} - y) + 2G^T \text{sign}(Gx_{(k)})]$$