FUNCTION APPROXIMATION

GOALS FOR TODAY

AND WHAT YOU SHOULD LEARN

1. What is function approximation
2. Universal function approximation Theorem(s)
3. What are and why basis functions are needed
4. How to do linear function approximation with a basis function

FUNCTION APPROXIMATION

WHAT IS IT?

For any function f_*, e.g., $f_*(x) = ax^2 + bx + c$, we want an approximation $f(x)$

$$\forall x, \; |f(x) - f_*(x)| < \epsilon$$

FUNCTION APPROXIMATION

WHAT IS IT?

For any function f_*, e.g., $f_*(x) = ax^2 + bx + c$, we want an approximation $f(x)$

$$\forall x, \ |f(x) - f_*(x)| < \epsilon$$

Why?

- $f_*(x)$ might be expensive (or impossible) to compute — even if we know f_*
  - $sin(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \ldots$ ==> Computers approximate this

FUNCTION APPROXIMATION

WHAT IS IT?

For any function f_*, e.g., $f_*(x) = ax^2 + bx + c$, we want an approximation $f(x)$

$$\forall x,\ |f(x) - f_*(x)| < \epsilon$$

Why?

- We don't know f_*
  - supervised learning: samples of $\big(x, y = f_*(x)\big)$ pairs

FUNCTION APPROXIMATION

WHAT IS IT?

For any function f_*, e.g., $f_*(x) = ax^2 + bx + c$, we want an approximation $f(x)$

$$\forall x,\ |f(x) - f_*(x)| < \epsilon$$

Why?

- Sometimes both
  - Create a big model f to approximate f_* from a data set
  - Compress f into a faster version f'

FUNCTION APPROXIMATION

HOW

$f_* : \mathbb{R} \to \mathbb{R}$

Real numbers may be too large to approximate

Consider some bounded set $\mathcal{X} = [-1,1]$

$f : \mathcal{X} \to \mathbb{R}$

$$\forall x \in \mathcal{X},\ |f(x) - f_*(x)| < \epsilon$$

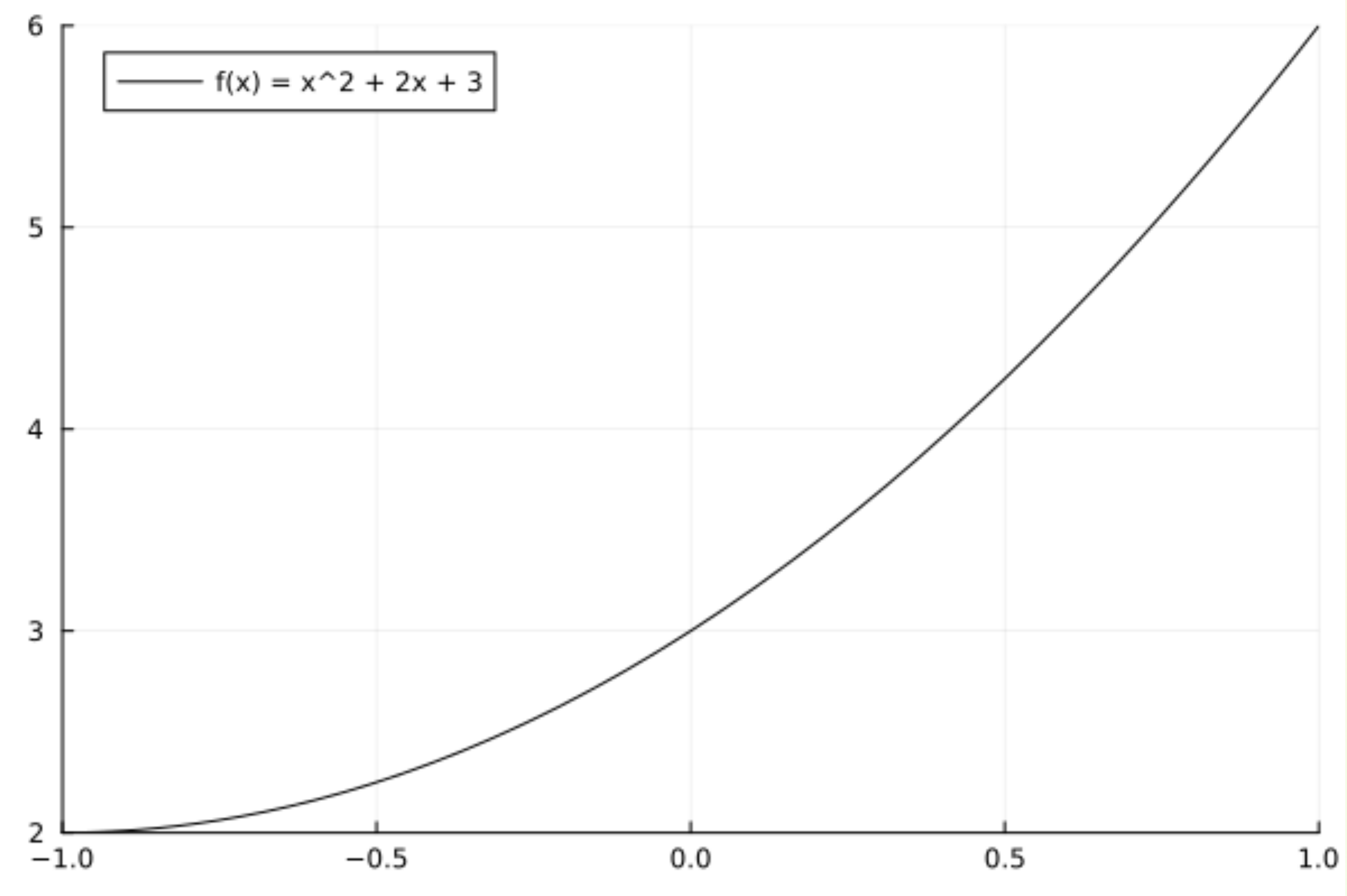
FUNCTION APPROXIMATION

EXAMPLE

We know the functional form: $f_*(x) = ax^2 + bx + c$

$$f_*(x) = x^2 + 2x + 3$$

$$\mathscr{X} = [-1,1]$$



FUNCTION APPROXIMATION

EXAMPLE

We know the functional form: $f_*(x) = ax^2 + bx + c$

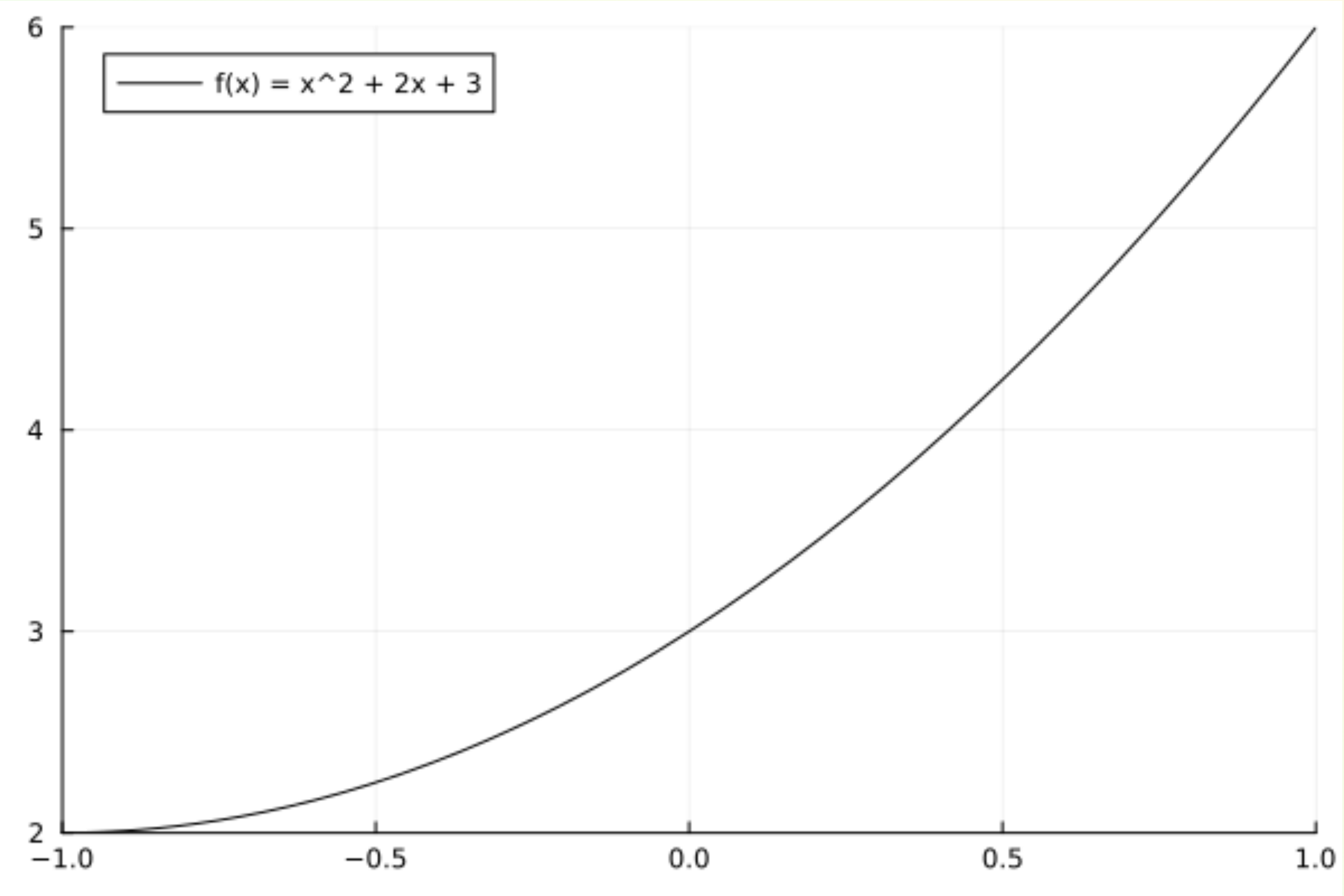
$$f_*(x) = x^2 + 2x + 3$$

$$\mathcal{X} = [-1,1]$$

$$f(x, a, b, c) = ax^2 + bx + c$$

Need to find a, b, c to minimize the error

$$|f(x, a, b, c) - f_*(x)|$$



FUNCTION APPROXIMATION

EXAMPLE

Have 3 unknown variables: a, b, c

Need 3 evaluations of f at unique x, i.e., $f_*(x_1), f_*(x_2), f_*(x_3)$

FUNCTION APPROXIMATION

EXAMPLE

Have 3 unknown variables: a, b, c

Need 3 evaluations of f at unique x, i.e., $f_*(x_1), f_*(x_2), f_*(x_3)$

Create a system of equations:

$$
\begin{aligned}
f_*(x_1) &= ax_1^2 + bx_1 + c \\
f_*(x_2) &= ax_2^2 + bx_2 + c \\
f_*(x_3) &= ax_3^2 + bx_3 + c
\end{aligned}
$$

Solve

FUNCTION APPROXIMATION

EXAMPLE

$$
\begin{aligned}
f_*(x_1) &= \; ax_1^2 + bx_1 + c \\
f_*(x_2) &= \; ax_2^2 + bx_2 + c \\
f_*(x_3) &= \; ax_3^2 + bx_3 + c
\end{aligned}
$$

FUNCTION APPROXIMATION

EXAMPLE

$$\underbrace{\begin{bmatrix} f_*(x_1) \\ f_*(x_2) \\ f_*(x_3) \end{bmatrix}}_{=b} = \underbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w}$$

FUNCTION APPROXIMATION

EXAMPLE

$$\underbrace{\begin{bmatrix} f_*(x_1) \\ f_*(x_2) \\ f_*(x_3) \end{bmatrix}}_{=b} = \underbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w}$$

$b = Aw$

FUNCTION APPROXIMATION

EXAMPLE

$$\underbrace{\begin{bmatrix} f_*(x_1) \\ f_*(x_2) \\ f_*(x_3) \end{bmatrix}}_{=b} = \underbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w}$$

$b = Aw$

$A^{-1}A = I, Iw = w$

FUNCTION APPROXIMATION

EXAMPLE

$$\underbrace{\begin{bmatrix} f_*(x_1) \\ f_*(x_2) \\ f_*(x_3) \end{bmatrix}}_{=b} = \underbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w}$$

$b = Aw$

$A^{-1}A = I, Iw = w$

$A^{-1}b = w$

FUNCTION APPROXIMATION

EXAMPLE

$$\underbrace{\begin{bmatrix} f_*(x_1) \\ f_*(x_2) \\ f_*(x_3) \end{bmatrix}}_{=b} = \underbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w}$$

$$b = Aw$$

$$A^{-1}A = I, Iw = w$$

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

FUNCTION APPROXIMATION

EXAMPLE

$$\underbrace{\begin{bmatrix} f_*(x_1) \\ f_*(x_2) \\ f_*(x_3) \end{bmatrix}}_{=b} = \underbrace{\begin{bmatrix} x_1^2 & x_1 & 1 \\ x_2^2 & x_2 & 1 \\ x_3^2 & x_3 & 1 \end{bmatrix}}_{=A} \underbrace{\begin{bmatrix} a \\ b \\ c \end{bmatrix}}_{w}$$

$b = Aw$

$A^{-1}A = I, Iw = w$

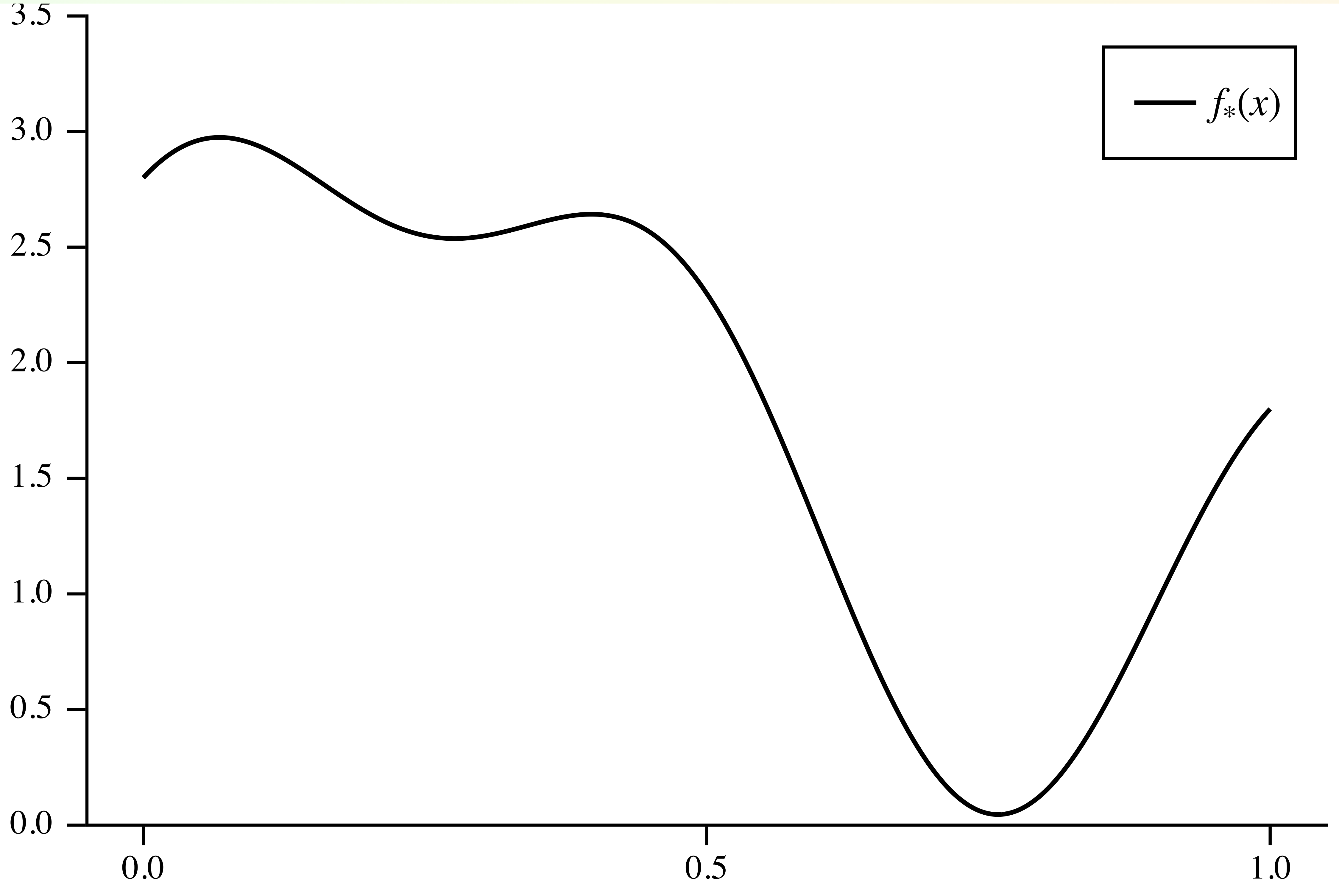
$w = A^{-1}b$

A **must be invertible**

FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f_*

Need to find $f(x) \approx f_*(x)$



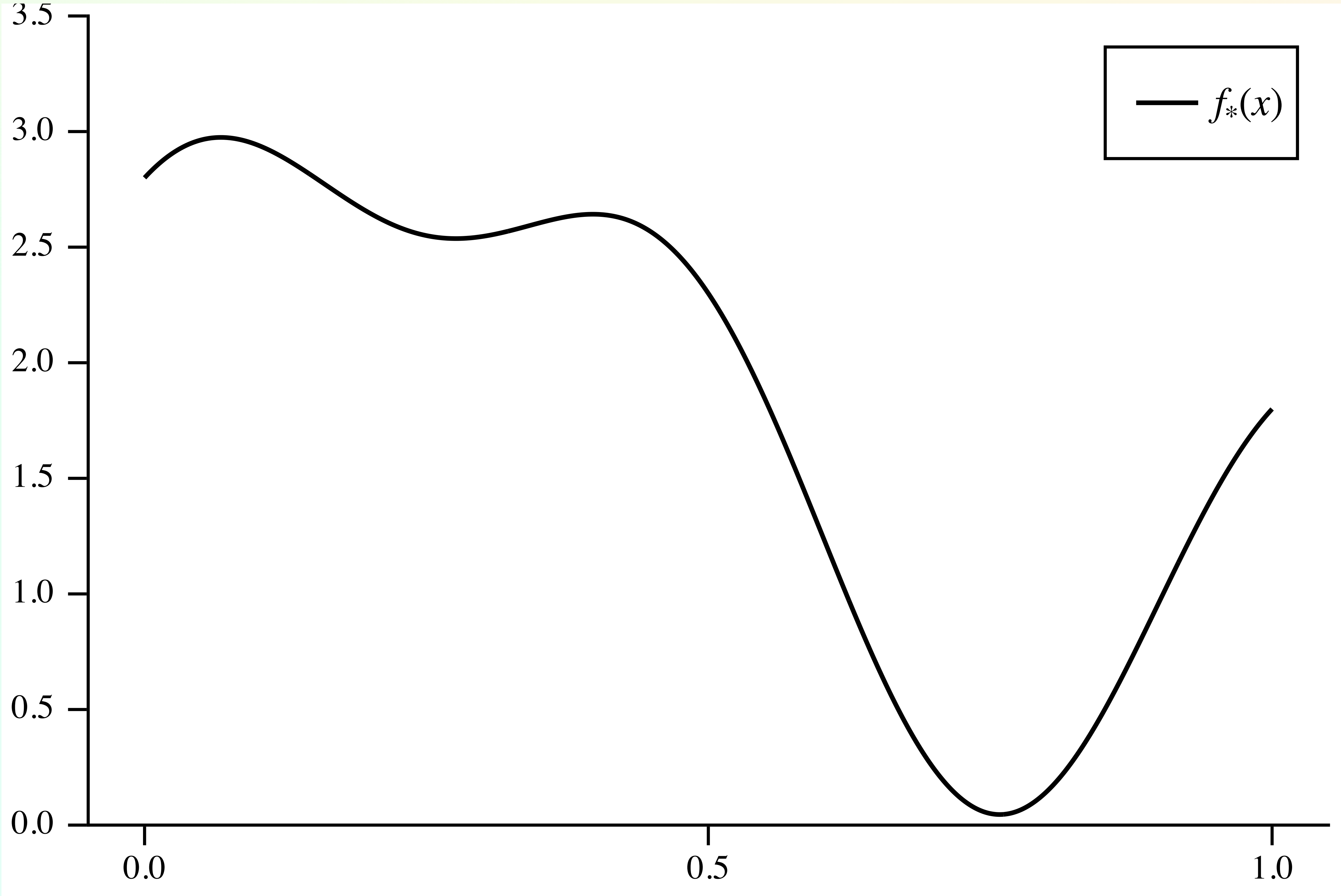
FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f_*

Need to find $f(x) \approx f_*(x)$

Discrete approximation (binning)

- Split x into bins
- $f(x)$ is the average height of the bin



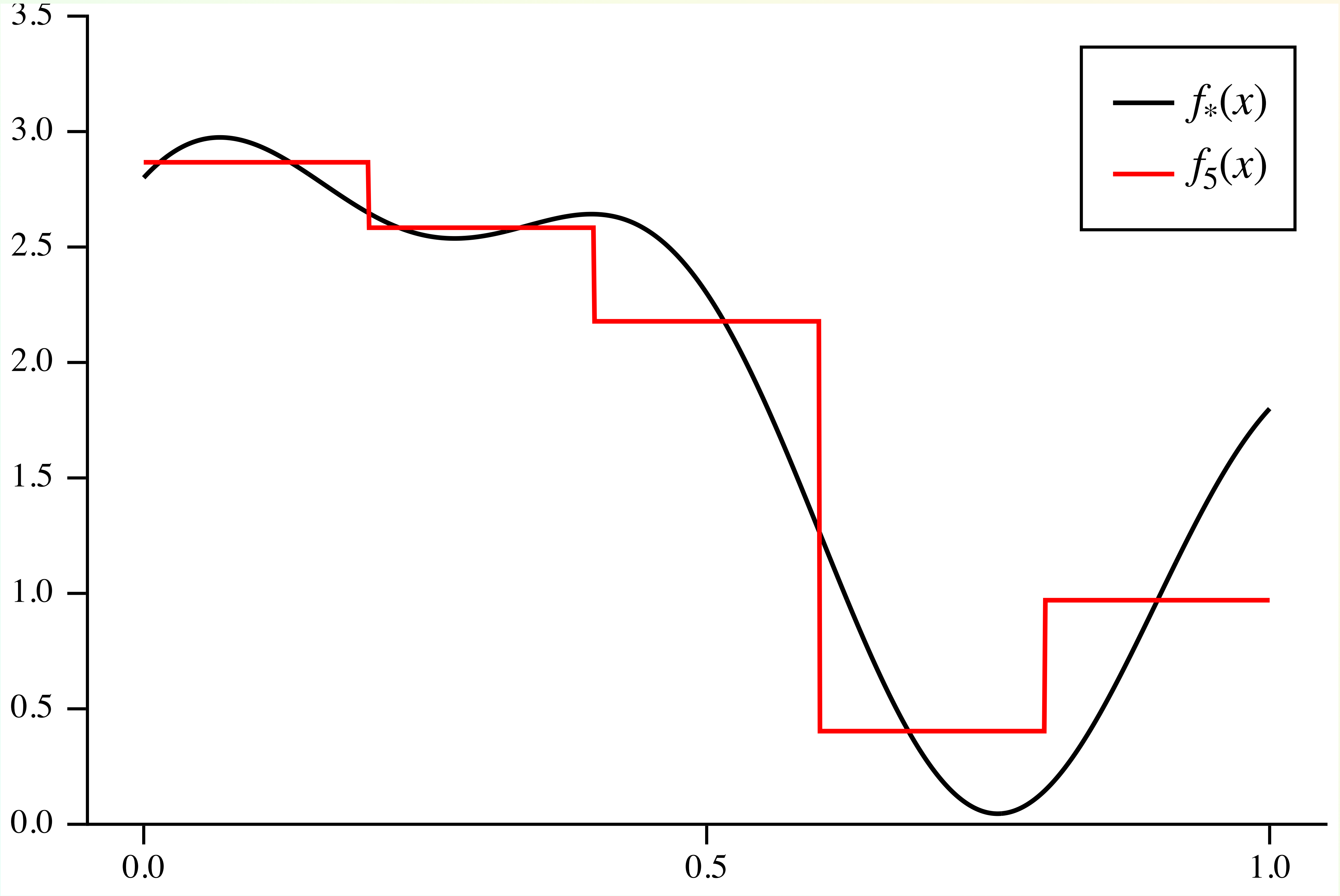
FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f_*

Need to find $f(x) \approx f_*(x)$

Discrete approximation (binning)

- Split x into bins
- $f(x)$ is the average height of the bin



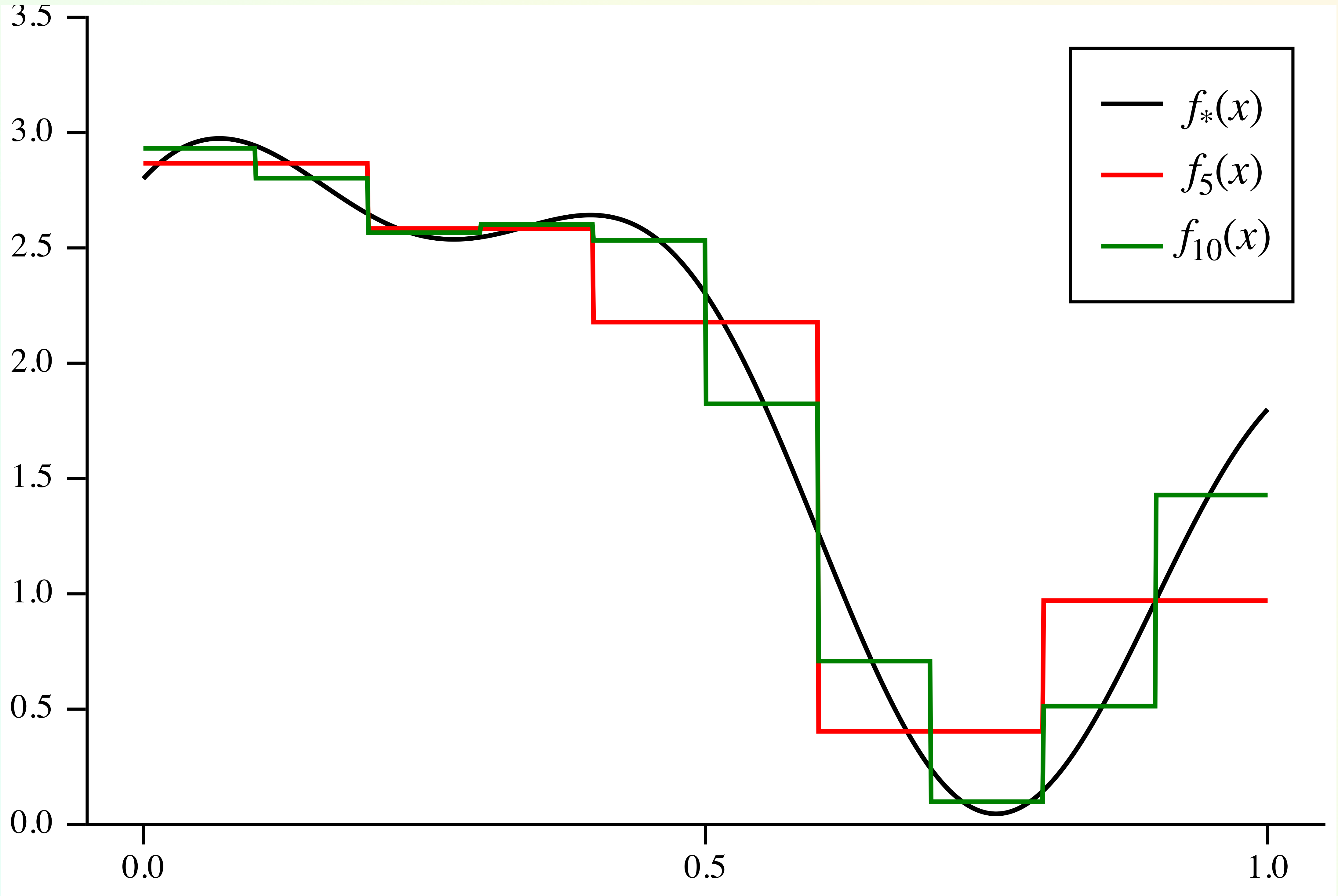
FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f_*

Need to find $f(x) \approx f_*(x)$

Discrete approximation (binning)

- Split x into bins
- $f(x)$ is the average height of the bin



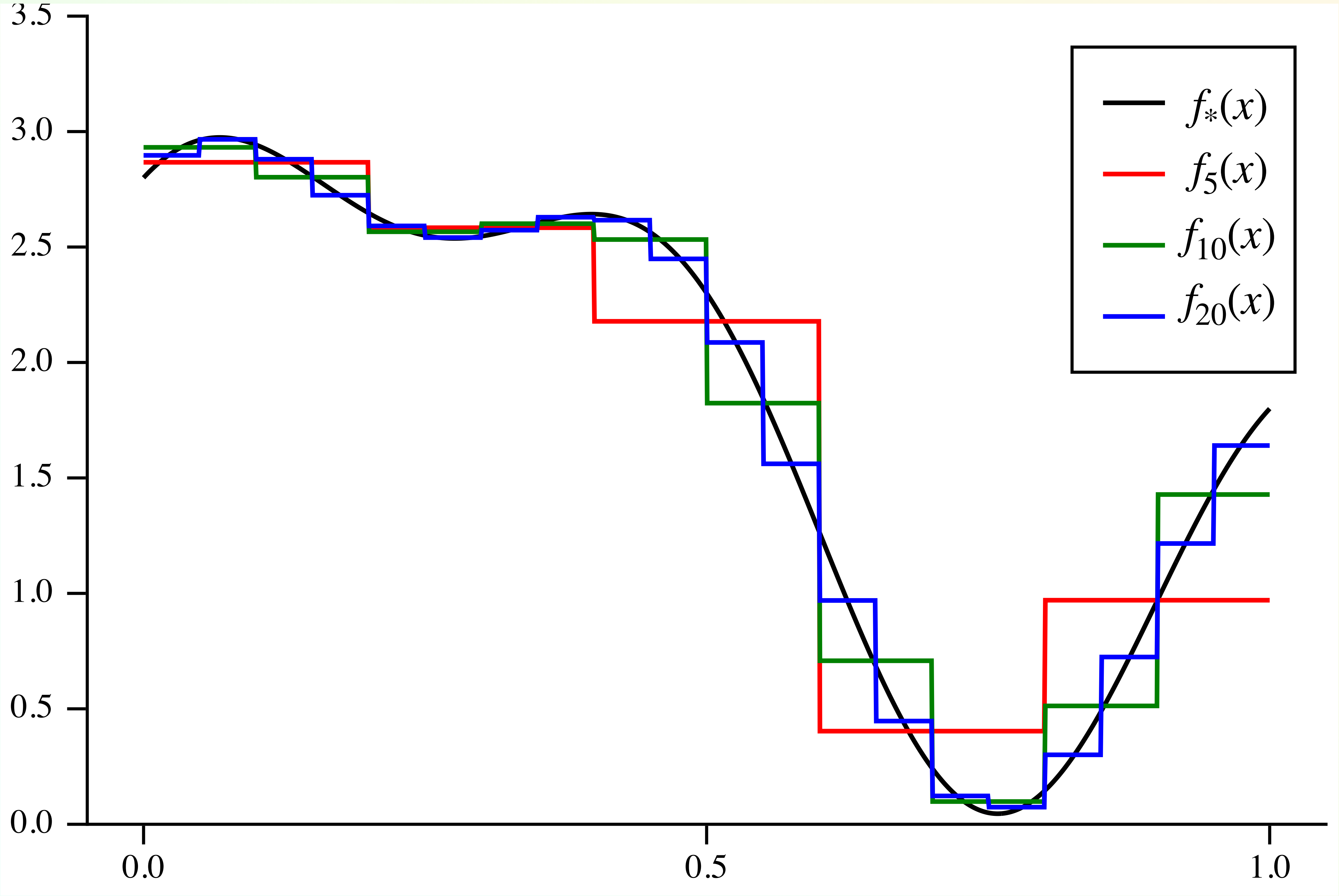
FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f_*

Need to find $f(x) \approx f_*(x)$

Discrete approximation (binning)

- Split x into bins
- $f(x)$ is the average height of the bin



FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f_*

Represent as a linear function with a basis function

$\phi: \mathbb{R} \to \mathbb{R}^n$ is the basis function

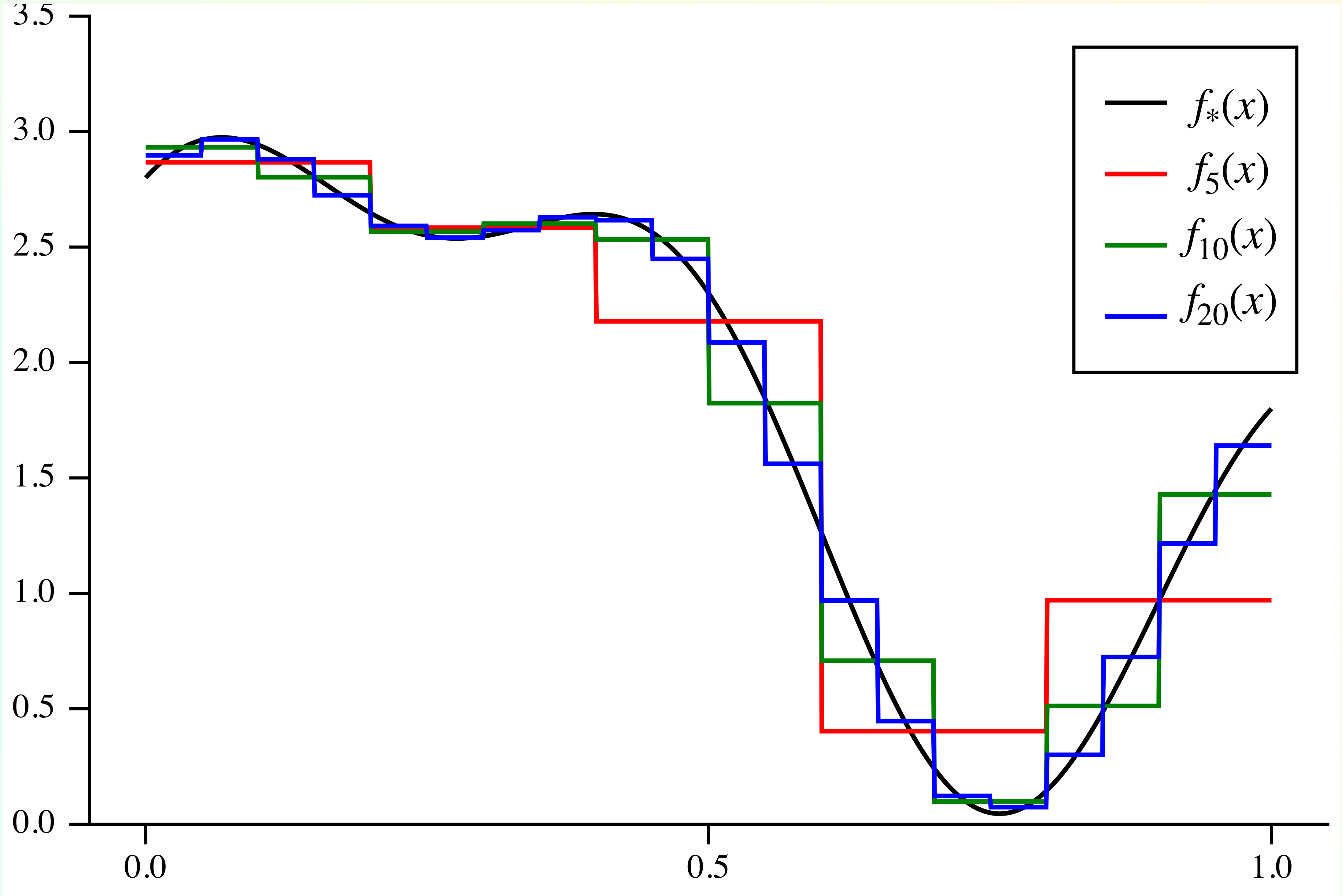
$\phi_i(x)$ is the output of i^{th} the basis function

For discrete approximation

$$\phi_i(x) = \begin{cases} 1 & x \in \text{bin}_i \\ 0 & \text{otherwise} \end{cases}$$

$\phi(x) = [0,0,1,0,0]^\top$ — one-hot vector

$$f(x, w) = w^\top \phi(x) = \sum_{i=1}^{n} w_i \phi_i(x)$$



FUNCTION APPROXIMATION

APPROACHES DON'T KNOW PARAMETRIC FORM FOR f

$$f(x, w) = w^\top \phi(x) = \sum_{i=1}^{n} w_i \phi_i(x)$$

$$\forall x \in \mathscr{X},\ |f(x, w) - f_*(x)| < \epsilon$$

1. For a fixed number of bins, we cannot guarantee this approximation
2. Assume f is smooth on $\mathscr{X}$, then we could upper-bound the number of bins necessary for ϵ error.
3. Practice: use only as many bins as you need and hope it is enough :(
   - If you can compute $f_*(x)$ then it is possible to get machine precision accurate $f(x, w)$

FUNCTION APPROXIMATION

BASIS FUNCTIONS

Polynomial:

$$\phi_i(x) = x^{(i-1)}, \phi(x) = [x^4, x^3, x^2, x, 1]$$

FUNCTION APPROXIMATION

BASIS FUNCTIONS

Polynomial:

$$\phi_i(x) = x^{(i-1)}, \phi(x) = [x^4, x^3, x^2, x, 1]$$

FUNCTION APPROXIMATION

BASIS FUNCTIONS

Polynomial:

$$\phi_i(x) = x^{(i-1)}, \phi(x) = [x^4, x^3, x^2, x, 1]$$

Create new features by going to higher powers

$$\phi(x) = \left[x^n, x^{n-1}, \cdots, x^4, x^3, x^2, x, 1\right]$$

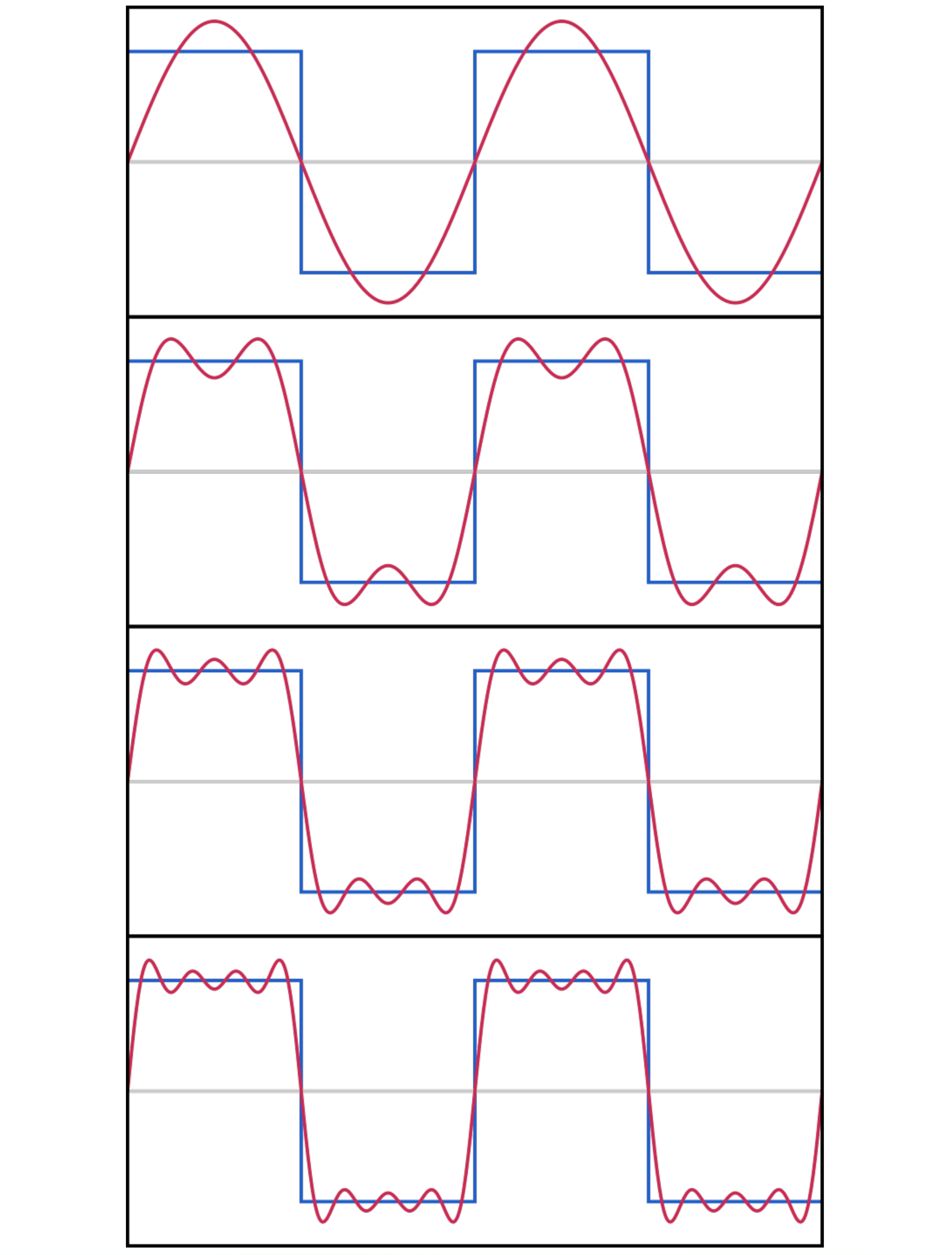
$$w \in \mathbb{R}^{n+1}$$

$$f(x, w) = w^\top \phi(x, w) = w_1 x^n + w_2 x^{n-1} + \cdots + w_n x + w_{n+1}$$

FUNCTION APPROXIMATION

BASIS FUNCTIONS

Fourier Features

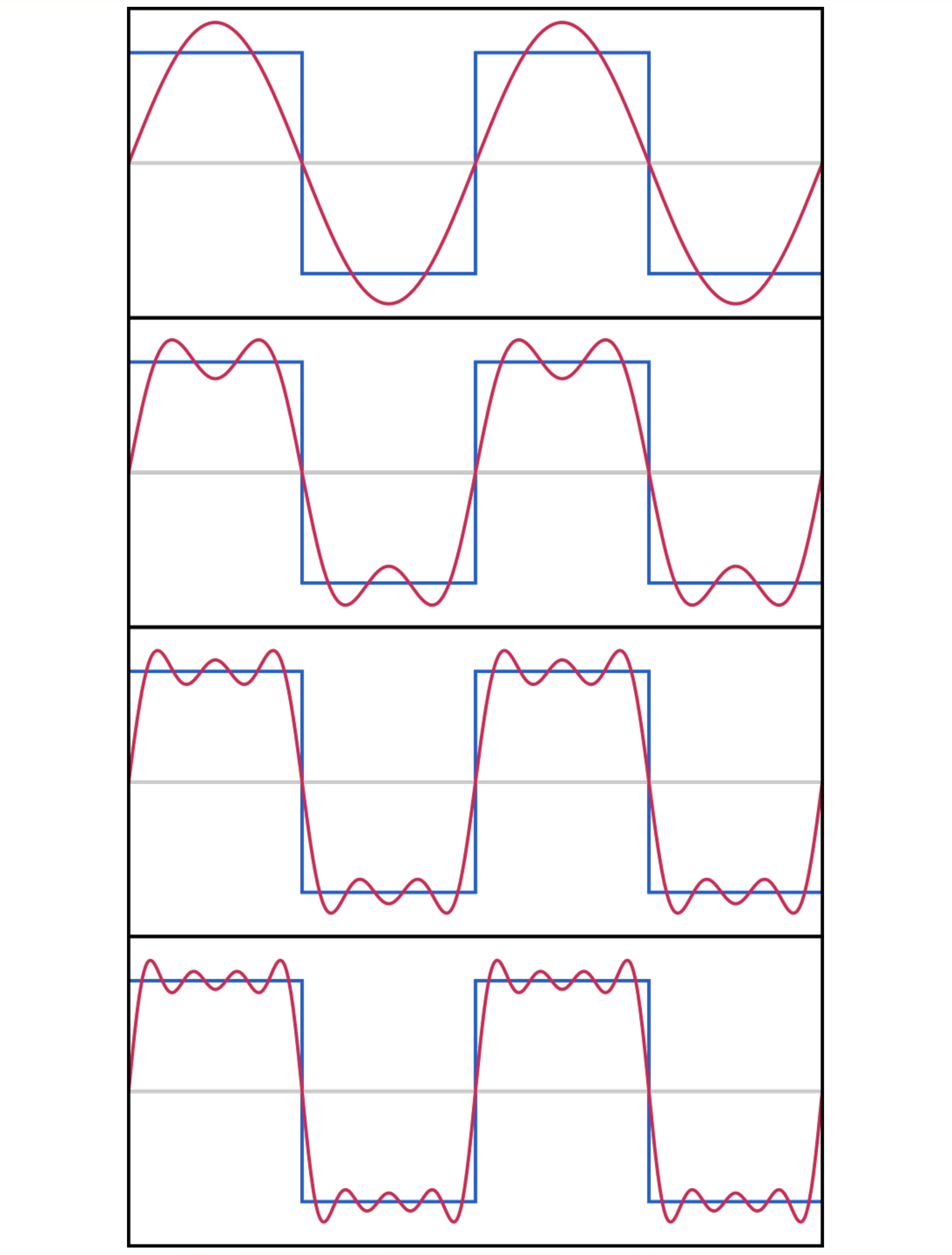


FUNCTION APPROXIMATION

BASIS FUNCTIONS

Fourier Features

$$\phi_i(x) = \sin((i-1)\pi x),\ \phi_{i+n}(x) = \cos((i-1)\pi x)$$



FUNCTION APPROXIMATION

BASIS FUNCTIONS

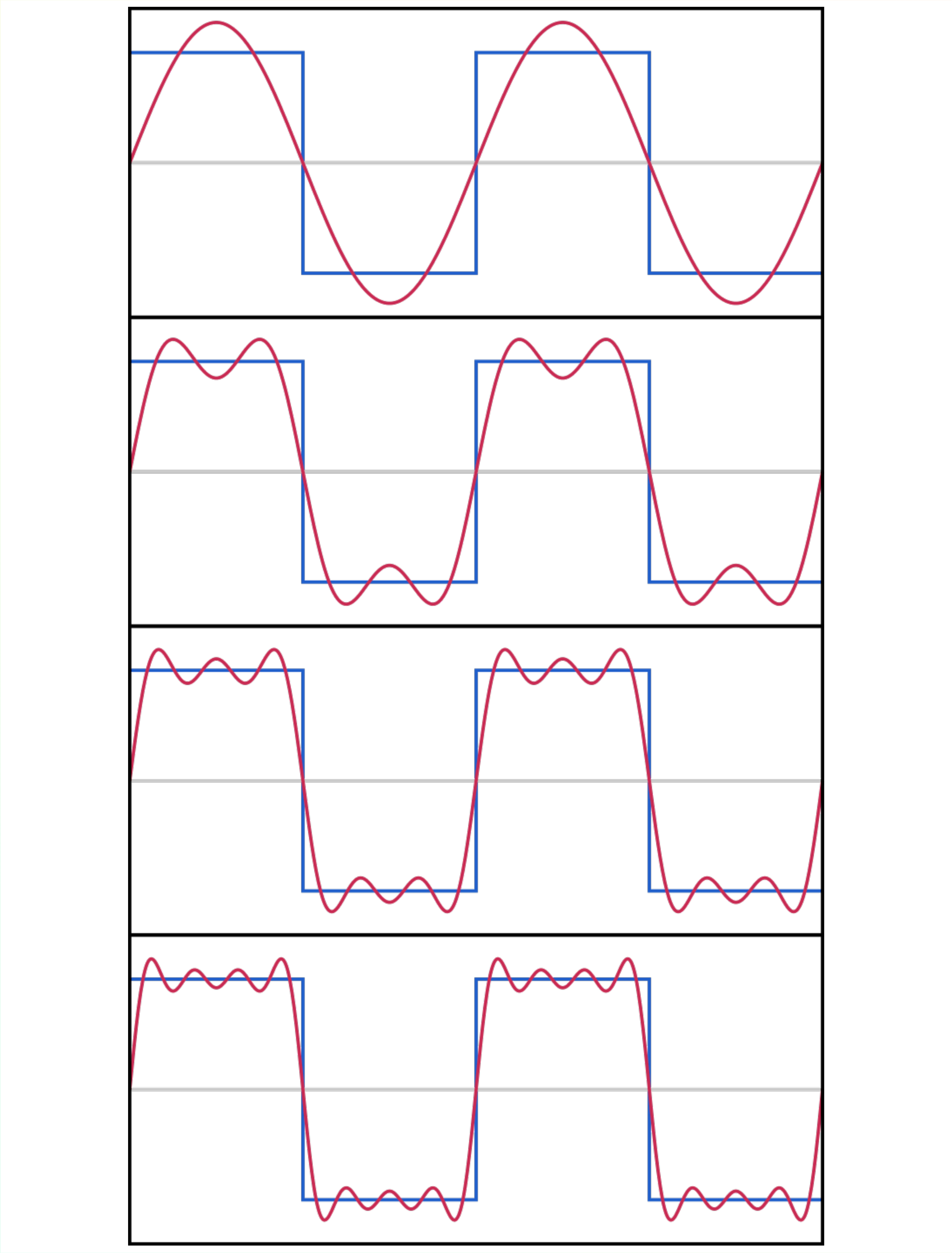
Fourier Features

$$\phi_i(x) = \sin((i-1)\pi x),\ \phi_{i+n}(x) = \cos((i-1)\pi x)$$

$$\phi(x) = [\sin(0), \sin(\pi x), \sin(2\pi x), \cos(0), \cos(\pi x), \cos(2\pi x)]$$

Increase the frequency to get a more accurate approximation

Works well for functions that smooth and don't have discrete jumps



FUNCTION APPROXIMATION

BASIS FUNCTIONS

Binning: $\phi(x) = [0,0,1,0,0]^\top$

It is good for functions with discrete jumps. However, there is no interpolation between bins.

Polynomial: $\phi_i(x) = x^{(i-1)}$, $\phi(x) = [x^4, x^3, x^2, x, 1]$

Don't usually work that well unless the function is a polynomial function

Fourier Features: $\phi(x) = [\sin(0), \sin(\pi x), \sin(2\pi x), \cos(0), \cos(\pi x), \cos(2\pi x)]$

Good for smooth functions (no jumps)

Many other basis functions

Could use any features we think will be helpful in approximating $f_*(x)$

UNIVERSAL FUNCTION APPROXIMATION

LIMIT OF INFINITE FEATURES

Want to be able always to get a better approximation with more features

Universal function approximators satisfy the following

$$\lim_{n\to\infty} \sup_{x\in[a,b]} |f_*(x) - f_n(x, w^*)| \to 0$$

w^* are the weights for the best fit to $f_*(x)$

Theoretical statements are a bit different, but this is the basic idea

UNIVERSAL FUNCTION APPROXIMATION

LIMIT OF INFINITE FEATURES

Some approximations that are guaranteed to represent the function with infinite features:

- Polynomial basis
- Fourier basis
- Infinite width neural network
- Many more

FUNCTION APPROXIMATION

NOISY SAMPLES FROM f

$Y = f_*(x) + \xi$

ξ is noised sampled from some distribution

Often assume $\xi \sim \mathcal{N}(0, \sigma^2)$

We will never know $f_*(x)$ and cannot measure $|f(x, w) - f_*(x)|$

We need a different measure of the approximation error in f

FUNCTION APPROXIMATION

NOISY SAMPLES FROM f

Let X be a random variable representing an a draw of x from $\mathcal{X}$

Let $Y = f_*(X) + \xi$

We can measure the approximation error in terms of mean squared error

$$\mathbf{E}\left[\left(f(X, w) - Y\right)^2\right]$$

"On average, how far away is the estimate from the samples Y"

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$w^* \in \arg\min_w l(w)$$

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$w^* \in \arg\min_{w} l(w)$$

If $\frac{\partial}{\partial w}\mathbf{E}\left[\left(f(X,w) - Y\right)^2\right] = 0$, then w is a minimizer

Solve for w

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$\frac{\partial}{\partial w}\mathbf{E}\left[\left(f(X,w)-Y\right)^2\right] = ?$$

PREREQ REMINDER

RULES

Addition: $\frac{d}{dx}(ax + bx) = \frac{d}{dx}ax + \frac{d}{dx}bx$

Powers: $\frac{d}{dx}x^2 = 2x$

Chain Rule: $\frac{d}{dx}g(f(x)) = \frac{d}{dy}g(y)\Big|_{y=f(x)}\frac{d}{dx}f(x)$

$$\mathbf{E}\left[f(X)\right] = \sum_x \Pr(X = x)f(x)$$

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$
\begin{aligned}
\frac{\partial}{\partial w}\mathbf{E}\left[\left(f(X,w)-Y\right)^2\right] &= \mathbf{E}\left[\frac{\partial}{\partial w}\left(f(X,w)-Y\right)^2\right] \\
&= 2\mathbf{E}\left[\left(f(X,w)-Y\right)\frac{\partial}{\partial w}\left(f(X,w)-Y\right)\right] \\
&= 2\mathbf{E}\left[\left(f(X,w)-Y\right)\left(\frac{\partial}{\partial w}f(X,w)-\frac{\partial}{\partial w}Y\right)\right] \\
&= 2\mathbf{E}\left[\left(f(X,w)-Y\right)\frac{\partial}{\partial w}f(X,w)\right]
\end{aligned}
$$

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$\frac{\partial}{\partial w} f(X, w) = \frac{\partial}{\partial w} \phi(x)^\top w = \phi(x)$$

$$2\mathbf{E}\left[\left(f(X, w) - Y\right) \frac{\partial}{\partial w} f(X, w)\right] = 2\mathbf{E}\left[\left(f(X, w) - Y\right) \phi(X)\right]$$

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$
\begin{aligned}
2\mathbf{E}\left[\left(f(X,w) - Y\right)\phi(X)\right] &= 0 \\
\mathbf{E}\left[f(X,w)\phi(X)\right] - \mathbf{E}\left[Y\phi(X)\right] &= 0 \\
\mathbf{E}\left[f(X,w)\phi(X)\right] &= \mathbf{E}\left[Y\phi(X)\right] \\
\mathbf{E}\left[\phi(X)^\top w\phi(X)\right] &= \mathbf{E}\left[Y\phi(X)\right] \\
\mathbf{E}\left[\phi(X)\phi(X)^\top w\right] &= \mathbf{E}\left[Y\phi(X)\right] \\
\mathbf{E}\left[\phi(X)\phi(X)^\top\right] w &= \mathbf{E}\left[Y\phi(X)\right] \\
w &= \mathbf{E}\left[\phi(X)\phi(X)^\top\right]^{-1}\mathbf{E}\left[Y\phi(X)\right]
\end{aligned}
$$

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR

$$
\begin{aligned}
w &= \underbrace{\mathbf{E}\left[\phi(X)\phi(X)^\top\right]^{-1}}_{=A} \underbrace{\mathbf{E}\left[Y\phi(X)\right]}_{=b} \\
&= A^{-1}b
\end{aligned}
$$

FUNCTION APPROXIMATION

MINIMUM MEAN SQUARED ERROR — IN PRACTICE

A finite number of samples m, $(x_1, y_1), (x_2, y_2), \cdots, (x_m, y_m)$

$y = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_m \end{bmatrix}, A = \begin{bmatrix} \phi(x_1) \\ \phi(x_2) \\ \vdots \\ \phi(x_m) \end{bmatrix}$ A is a $m \times n$ matrix

$w = A^+ y$

A^+ is the pseudo inverse

FINDING THE BEST FIT

GRADIENT DESCENT

Assume we have a data set of inputs and outputs:

$$x_1, x_2, \ldots, x_m, \;\; y_1 = f(x_1), y_2 = f(x_2), \ldots, y_m = f(x_m)$$

Loss function for weights w (how bad the approximation is)

$$l(w) = \frac{1}{m} \sum_{i=1}^{m} \left(f(x_i, w) - y_i\right)^2$$

GRADIENT DESCENT ON $l(w)$

STOCHASTIC GRADIENT DESCENT

$$l(w) = \frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}\left(f(x_i, w) - y_i\right)^2$$

$$\nabla l(w) = \frac{\partial}{\partial w}\frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}\left(y_i - f(x_i, w)\right)^2$$

$$= \frac{1}{2}\frac{1}{m}\sum_{i=1}^{m}\frac{\partial}{\partial w}\left(f(x_i, w) - y_i\right)^2 = \frac{1}{m}\sum_{i=1}^{m}\left(f(x_i, w) - y_i\right)\frac{\partial f(x_i, w)}{\partial w}$$

Idea: move in the direction of the sample estimate of the gradient

$$w \leftarrow w - \eta \nabla l(w)$$

NEXT CLASS

Next Class — Classification Problem