最小二乗法の信頼区間と予測区間

黒木玄

2020-06-11, 2020-10-26

- Jupyterノートブック版 (https://nbviewer.jupyter.org/github/genkuroki/Statistics/blob/master/Least%20squares%20estimate
- PDF版 (https://genkuroki.github.io/documents/Statistics/Least%20squares%20estimates.pdf)

以下の説明では直交射影が本質的な役割を果たす.

最小二乗法は直交射影の言い換えに過ぎない.

1 解説

一次独立な函数系 $f_1, \dots, f_r$ による

$$y = b_1 f_1(x) + \cdots + b_r f_r(x) + \varepsilon(x), \quad \varepsilon(x) \sim \mathrm{Normal}(0, \sigma)$$

型のモデルによるフィッティングは以下のように実現される. ($\varepsilon(x)$ 達は互いに独立だと仮定する.)

1.1 データ

(x_i, y_i) $(i = 1, \dots, n)$ をデータとする.

1.2 行列 X と縦ベクトル b, y の定義

$n \times r$ 行列 X と r 次元列(縦)ベクトル b と n 次元列ベクトル y を以下のように定める:

$$X = \begin{bmatrix} f_1(x_1) & \cdots & f_r(x_1) \\ \vdots & & \vdots \\ f_1(x_n) & \cdots & f_r(x_n) \end{bmatrix}, \quad b = \begin{bmatrix} b_1 \\ \vdots \\ b_r \end{bmatrix}, \quad y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix}.$$

X 内の r 本の列(縦)ベクトル達は一次独立であると仮定する. その仮定は $r \times r$ 行列 $X^{\mathrm{T}}X$ は可逆になることや, X のランクが r になることと同値である.

1.3 直交射影

$\mathrm{Im}\, X = X\mathbb{R}^r = \{\, Xb \mid b \in \mathbb{R}^r \,\}$ への y の直交射影を $\hat{y}$ と書く:

$$\hat{y} = X\hat{b}, \quad \hat{b} = (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y.$$

$\hat{y}$ は y の直交射影なので $\|y - Xb\|^2$ を最小にする b は $\hat{b}$ になる. すなわち $\hat{b}$ は最小二乗法の解になっている. ($\|y - Xb\|^2$ が最小化されている.)

証明: $\hat{y}$ を y の $\mathrm{Im}\, X$ への直交射影とすると, $\hat{y} = X\hat{b}$, $\hat{b} \in \mathbb{R}^r$ と書くことができ, X のすべての列と $y - \hat{y}$ は直交し, $X^{\mathrm{T}}(y - \hat{y}) = 0$ すなわち $X^{\mathrm{T}}y = X^{\mathrm{T}}X\hat{b}$ が成立している. $X^{\mathrm{T}}X$ は可逆なので $\hat{b} == (X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$ が成立する. q.e.d.

$\hat{y} = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$ は y の直交射影なので, $X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$ は二乗しても不変である(直交射影の結果を直交射影しても変化しない):

$$(X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})^2 = X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}.$$

これより

$$(E - X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})^2 = E - X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$$

が成立することもわかる. ここで E は $n \times n$ の単位行列である.

例: $r = 2, f_1(x) = 1, f_2(x) = x$ の場合には

$$X = \begin{bmatrix} 1 & x_1 \\ \vdots & \vdots \\ 1 & x_n \end{bmatrix}, \quad X^{\mathrm{T}}X = n \begin{bmatrix} 1 & \bar{x} \\ \bar{x} & \overline{x^2} \end{bmatrix}, \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \overline{x^2} = \frac{1}{n}\sum_{i=1}^{n} x_i^2.$$

これは $y = a + bx + \varepsilon$ 型の線形回帰の場合である. □

1.4 最小二乗法

$\hat{y}$ は y の $\mathrm{Im}\, X$ への直交射影なので, $y - \hat{y}$ と $Xb - \hat{y} \in \mathrm{Im}\, X$ は直交する. ゆえに

$$\|y - Xb\|^2 = \|(y - \hat{y}) - (Xb - \hat{y})\|^2 = \|y - \hat{y}\|^2 + \|Xb - \hat{y}\|^2.$$

そして, $\hat{y}$ による y の近似の二乗誤差は

$$\|y - \hat{y}\|^2 = \|y - X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y\|^2 = \|(E - X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})y\|^2 = y^{\mathrm{T}}(E - X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})y.$$

実はこれを n で割ったものが σ^2 の最尤法による推定結果に一致することを示せる. ただし, それは不偏推定量ではない. σ^2 の不偏推定量を得るためには $n - r$ で割らなければいけない. σ^2 の不偏推定量を

$$\hat{u}^2 = \frac{1}{n-r}\|y - \hat{y}\|^2, \quad \hat{u} > 0$$

と書くことにする.

例: $r = 1, f_1(x) = 1$ の場合は x_i 達の情報は無用になり, y_i 達のデータの正規分布によるフィッティングになる. この場合には $\hat{y}$ の成分はすべて y_i 達の平均

$$\bar{y} = \frac{1}{n}\sum_{j=1}^{n} y_j$$

に等しくなるので,

$$\|y - \hat{y}\|^2 = \sum_{i=1}^{n}(y_i - \bar{y})^2$$

となる. 分散の不偏推定量はこれを $n-1$ で割ることによって得られるのであった. この場合には $r=1$ なので $n-r=n-1$ になっていることに注意せよ. $r=1$ でない場合にも $\|y-\hat{y}\|^2$ を $n-r$ で割れば σ^2 の不偏推定量が得られる. □

1.5 σ^2 の不偏推定量の構成の詳細

$y=Xb+\varepsilon$ における ε の成分達 $\varepsilon_i=\varepsilon(x_i)$ $(i=1,\dots,n)$ の分布が独立で, 各々の ε_i は平均 0 と同一の分散 σ^2 を持つと仮定する. このとき,

$$\hat{y}=X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}y$$

なので,

$$y-\hat{y}=(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})y=(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})(Xb+\varepsilon)=(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})\varepsilon.$$

ゆえに, $y-\hat{y}$ の成分達の分散共分散行列は $X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$ が $\mathrm{Im}\,X$ (r 次元)への直交射影変換の表現行列であることより, $E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}$ は $\mathrm{Im}\,X$ の直交補空間($n-r$ 次元)への直交射影変換の表現行列になるので,

$$\begin{aligned}\mathbb{E}[(y-\hat{y})(y-\hat{y})^{\mathrm{T}}]&=\mathbb{E}[(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})\varepsilon\varepsilon^{\mathrm{T}}(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})]\\&=(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})\mathbb{E}[\varepsilon\varepsilon^{\mathrm{T}}](E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})\\&=\sigma^2(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}}).\end{aligned}$$

となることがわかる. ここで $\mathbb{E}[\ ]$ は確率変数にその期待値を与える汎函数である. ゆえに

$$\mathbb{E}[\|y-\hat{y}\|^2]=\mathrm{tr}\,\mathbb{E}[(y-\hat{y})(y-\hat{y})^{\mathrm{T}}]=(n-r)\sigma^2.$$

これより,

$$\hat{u}^2=\frac{1}{n-r}\|y-\hat{y}\|^2,\quad \hat{u}>0$$

が σ^2 の不偏推定量になっていることがわかる. □

1.6 データが正規分布に従うという仮定のもとでの結論

https://twitter.com/genkuroki/status/1264548714590253056
(https://twitter.com/genkuroki/status/1264548714590253056)

データ (x_i,y_i) $(i=1,\dots,n)$ が, 与えられた f_j 達と b_j 達と x_i 達と $\sigma>0$ を使って,

$$y_i=b_1f_1(x_i)+\cdots+b_rf_r(x_i)+\varepsilon_i,\quad \varepsilon_i\sim\mathrm{Normal}(0,\sigma)$$

によってランダムに生成されていると仮定する. ε_i 達は独立な確率変数であるとする.

このとき, 成分が y_i の n 次元列(縦)ベクトル y は多変量正規分布に従う確率変数になり,
$\hat{b}=(X^{\mathrm{T}}T)^{-1}X^{\mathrm{T}}y$ と $\hat{y}=X\hat{b}$ も多変量正規分布に従う確率変数になる.

ベクトル値の確率変数 $y-\hat{y}=y-X\hat{b}$ は平均が0で分散共分散行列がランク $n-r$ の $n\times n$ 行列

$$\sigma^2(E-X(X^{\mathrm{T}}X)^{-1}X^{\mathrm{T}})$$

の多変量正規分布に従い, $\hat{y}$ の各成分と $y-X\hat{b}$ の各成分の共分散は0になる. (このことは前節の計算よりわかる.)

1.7 サンプルを生成した未知の回帰曲線上の値の信頼区間

$x_1, \ldots, x_n$ の次の x_* が与えられていると仮定し,

$$f(x_*) = \begin{bmatrix} f_1(x_*) \\ \vdots \\ f_r(x_*) \end{bmatrix}$$

と r 次元列(縦)ベクトル $f(x_*)$ を定める.

以上の設定において,

$$f(x_*)^{\mathrm{T}} b - f(x_*)^{\mathrm{T}} \hat{b} = \sum_{j=1}^{r} b_j f_j(x_{n+1}) - \sum_{j=1}^{r} \hat{b}_j f_j(x_{n+1})$$

は期待値0分散 $\sigma^2 f(x_*)^{\mathrm{T}} (X^T X)^{-1} f(x_*)$ の正規分布に従うので、

$$\frac{f(x_*)^{\mathrm{T}} b - f(x_*)^{\mathrm{T}} \hat{b}}{\hat{u}\sqrt{f(x_*)^{\mathrm{T}} (X^{\mathrm{T}} X)^{-1} f(x_*)}}$$

は自由度 $n - r$ の t 分布に従う.

ゆえに,サンプルの生成に使われた回帰曲線の x_* における値

$$f(x_*)^{\mathrm{T}} b = b_1 f_1(x_*) + \cdots + b_r f_r(x_*)$$

の信頼度 $1 - \alpha$ の信頼区間を

$$f(x_*)^{\mathrm{T}} \hat{b} - t_{n-r}(\alpha/2)\hat{u}\sqrt{f(x_*)^{\mathrm{T}} (X^{\mathrm{T}} X)^{-1} f(x_*)} \leqq f(x_*)^{\mathrm{T}} b \leqq f(x_*)^{\mathrm{T}} \hat{b} + t_{n-r}(\alpha/2)\hat{u}\sqrt{f(x_*)^{\mathrm{T}} (X^{\mathrm{T}} X)^{-1} f(}$$

と定義できる. ここで, 自由度 $n - r$ の t 分布の累積分布函数を F と書いたときの $F^{-1}(1 - \alpha/2)$ を $t_{n-r}(\alpha/2)$ と書いた.

多くの環境で累積分布函数の逆函数は `quantile` という名前の函数として実装されている.

1.8 予測値の予測区間

さらに, $\varepsilon_1, \ldots, \varepsilon_n$ と独立な確率変数 ε_* で平均0分散 σ^2 に従うものを用意して,

$$y_* = f(x_*) b + \varepsilon_* = b_1 f_1(x_*) + \cdots + b_r f_r(x_*) + \varepsilon_*$$

とおく. このとき,

$$\frac{y_* - f(x_*)\hat{b}}{\hat{u}\sqrt{1 + f(x_*)(X^T X)^{-1} f(x_*)^T}}$$

は自由度 $n - r$ の t 分布に従う. 前節の式と比較すると分母の平方根内部に 1 が増えているが, その 1 は y_* を定義するときに用いた ε_* から来ている.

ゆえに, x_* における予測値 y_* の信頼度 $1 - \alpha$ の予測区間を

$$f(x_*)\hat{b} - t_{n-r}(\alpha/2)\hat{u}\sqrt{1 + f(x_*)(X^{\mathrm{T}} X)^{-1} f(x_*)^{\mathrm{T}}} \leqq y_* \leqq f(x_*)\hat{b} + t_{n-r}(\alpha/2)\hat{u}\sqrt{1 + f(x_*)(X^{\mathrm{T}} X)^{-1} f(}$$

と定義できる.

1.9 正規分布の仮定に注意せよ

以上で説明した信頼区間と予測区間はデータの残差の分布が独立同分布な正規分布に従うという仮定のもとで構成されている. ゆえに, 現実のデータの残差が独立同分布な正規分布に従っていないように見える場合には以上で定義した信頼区間と予測区間の使用は不適切になる.

信頼区間や予測区間はモデルの設定に依存して決まる. モデルが現実で妥当でなければそのモデルを使って計算した信頼区間や予測区間は信頼できないものになる.

次のセルは以上を素直にプログラムに翻訳したものである.

In [1]:

```
using LinearAlgebra, Distributions

norm2(x) = dot(x, x)

X_matrix(F, x) = hcat((f.(x) for f in F)...)

function orthogonal_projection(X, y)
    # b̂ = inv(X'X)*X'y
    # b̂ = (X'X)\X'y # is equivalent to b̂ = inv(X'X)*X'y
    b̂ = X\y # equivalent to b̂ = (X'X)\X'y
    ŷ = X*b̂
    Ŝ² = norm2(y - ŷ)
    b̂, ŷ, Ŝ²
end

function linear_regression(F, x, y)
    X = X_matrix(F, x)
    b̂, ŷ, Ŝ² = orthogonal_projection(X, y)
    n, r = size(X)
    û² = Ŝ²/(n - r)
    return b̂, √û², X
end

function confidence_interval_functions(F, X, b̂, û; α=0.05)
    n, r = size(X)
    t = quantile(TDist(n-r), 1-α/2)
    f(xstar) = (φ -> φ(xstar)).(F)
    m(xstar) = f(xstar)'b̂
    # d(xstar) = û*√(f(xstar)'/(X'X)*f(xstar))
    # Modify X'X to √eps()*I + X'X in order to prevent Singular
Exception.
    d(xstar) = û*√(f(xstar)'/(√eps()*I + X'X)*f(xstar))
    g₋(xstar) = m(xstar) - t*d(xstar)
    g₊(xstar) = m(xstar) + t*d(xstar)
    g₋, g₊
end

function prediction_interval_functions(F, X, b̂, û; α=0.05)
    n, r = size(X)
    t = quantile(TDist(n-r), 1-α/2)
    f(xstar) = (φ -> φ(xstar)).(F)
    m(xstar) = f(xstar)'b̂
    # d(xstar) = û*√(1 + f(xstar)'/(X'X)*f(xstar))
    # Modify X'X to √eps()*I + X'X in order to prevent Singular
Exception.
    d(xstar) = û*√(1 + f(xstar)'/(√eps()*I + X'X)*f(xstar))
    h₋(xstar) = m(xstar) - t*d(xstar)
    h₊(xstar) = m(xstar) + t*d(xstar)
    h₋, h₊
end
```

Out[1]: prediction_interval_functions (generic function with 1 method)

In [2]:

```
using Plots
pyplot(fmt = :svg)

using Random: seed!

rd(x, d=3) = round(x; digits=d)

reg_func(F, b) = (x -> (φ -> φ(x)).(F)'b)

function plot_linear_regression(F, x, f_true;
        n = length(x),
        y = f_true.(x) + σ*randn(n),
        α = 0.05,
        b = nothing,
        σ = nothing,
        xs = nothing,
        ylim = nothing
    )

    b̂, û, X = linear_regression(F, x, y)
    f_fit = reg_func(F, b̂)
    g₋, g₊ = confidence_interval_functions(F, X, b̂, û, α=α)
    h₋, h₊ = prediction_interval_functions(F, X, b̂, û, α=α)

    !isnothing(b) && @show b
    !isnothing(σ) && @show σ
    !isnothing(b) && !isnothing(σ) && println()
    @show b̂
    @show û

    isnothing(xs) && (xs = range(minimum(x), maximum(x),
length=600))
    P = plot()
    isnothing(ylim) || plot!(ylim = ylim)
    plot!(xs, f_fit.(xs); label="fitting curve", color=:red, lw=2)
    plot!(xs, g₋.(xs); color=:red, ls=:dot, label="$(100(1-α))%
conf. int.")
    plot!(xs, g₊.(xs); color=:red, ls=:dot, label="")
    plot!(xs, h₋.(xs); color=:red, ls=:dash, label="$(100(1-α))%
pred. int.")
    plot!(xs, h₊.(xs); color=:red, ls=:dash, label="")
    plot!(xs, f_true.(xs); label="true curve", color=:blue, lw=1.4,
alpha=0.85)
    scatter!(x, y; label="sample", color=:blue, msc=:blue,
alpha=0.5)
end
```

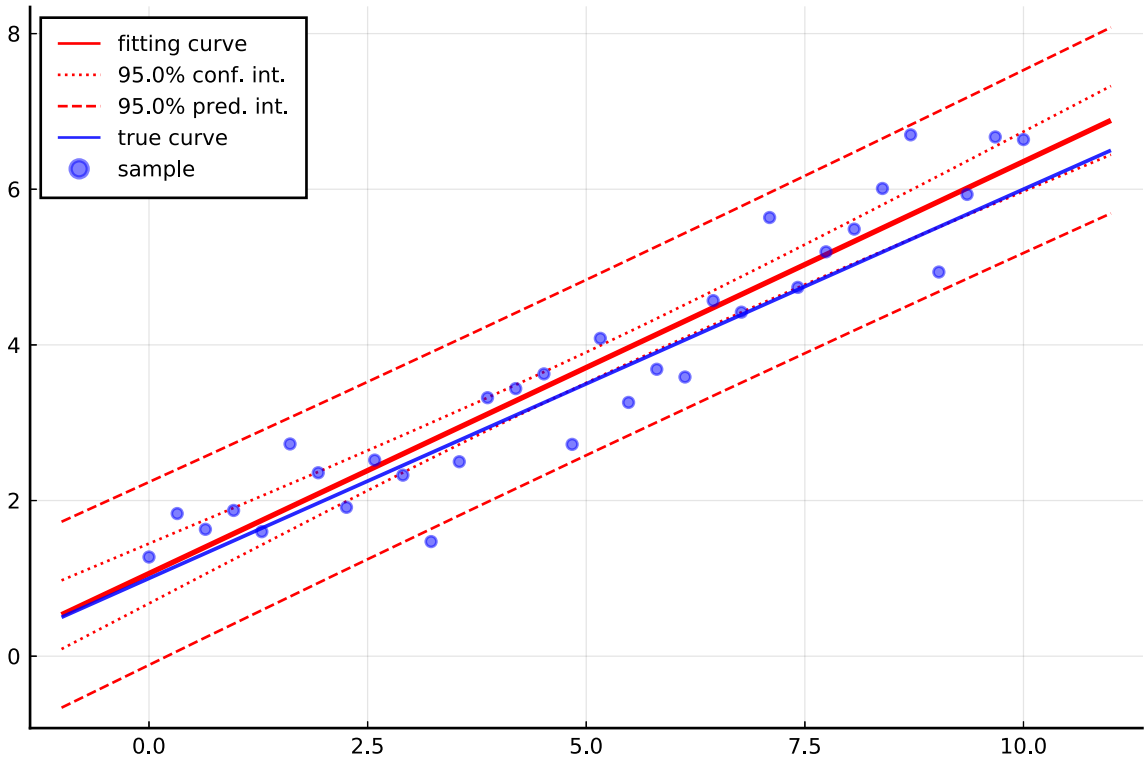
Out[2]: plot_linear_regression (generic function with 1 method)

In [3]:

```
n = 2^5
x = range(0, 10, length=n)
F = [one, identity]
b = [1, 0.5]
σ = 0.5
f_true = reg_func(F, b)

seed!(4649)
plot_linear_regression(F, x, f_true; xs=range(-1, 11; length=400))
```

```
b̂ = [1.0623231871378152, 0.5293426742494878]
û = 0.543556130053012
```

Out[3]:



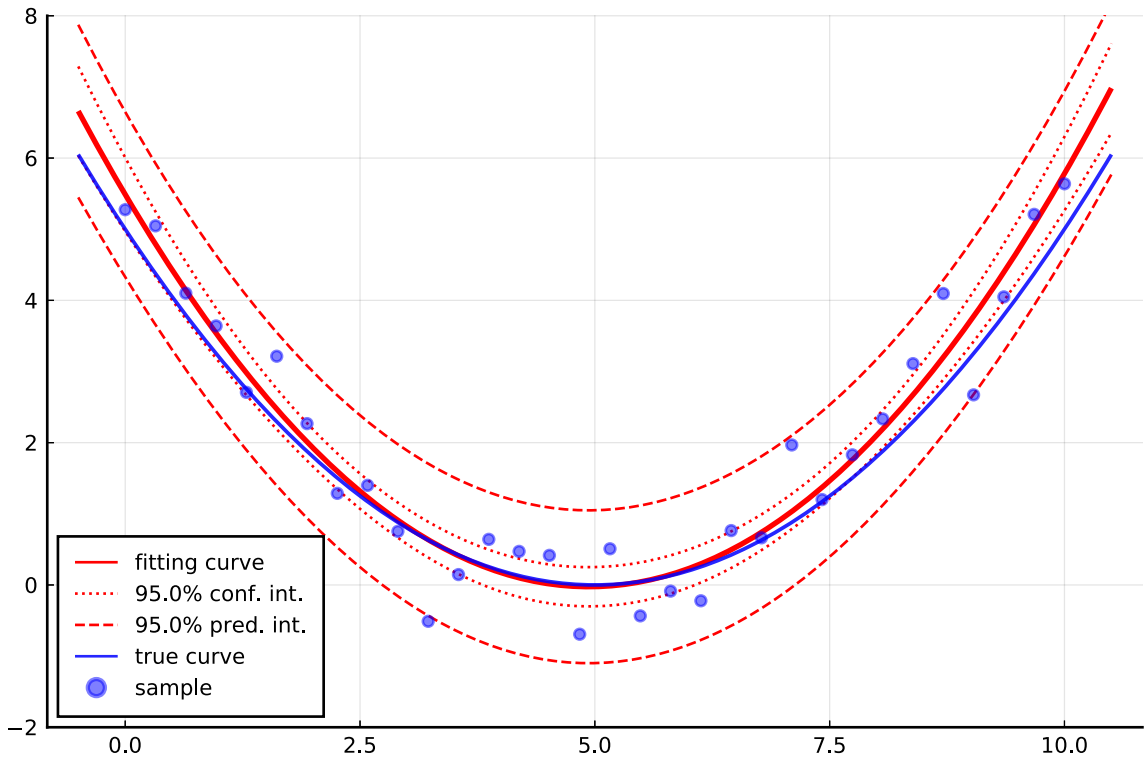
/

In [4]:

```
n = 2^5
x = range(0, 10, length=n)
F = [one, identity, x->x^2]
b = [5, -2, 0.2]
σ = 0.5
f_true = reg_func(F, b)

seed!(4649)
y = f_true.(x) + σ*randn(n)

plot_linear_regression(F, x, f_true; b=b, σ=σ, y=y, xs=range(-0.5,
10.5, length=400), ylim=(-2, 8))
```

```
b = [5.0, -2.0, 0.2]
σ = 0.5

b̂ = [5.486856844970879, -2.233868193607013, 0.226321086785650 08]
û = 0.5075974138073638
```

Out[4]:



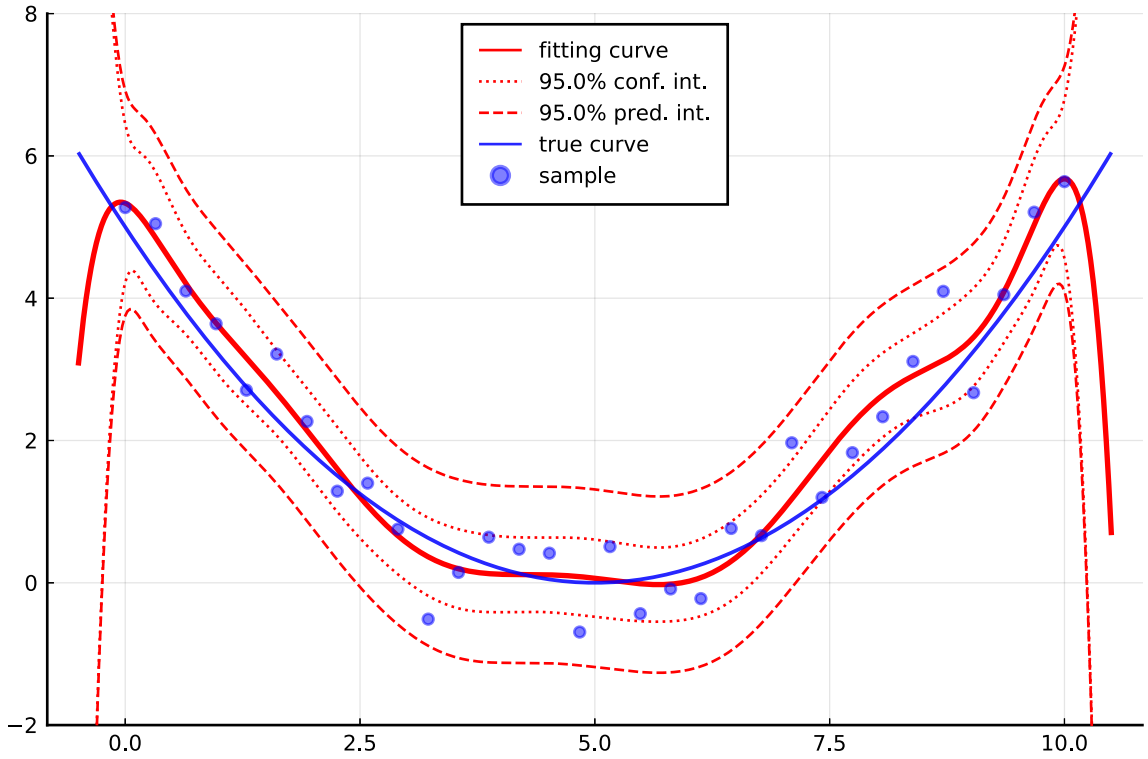
/

In [5]:

```
G = [one, identity, x->x^2, [x->x^k for k in 3:10]...]
plot_linear_regression(G, x, f_true; b=b, σ=σ, y=y, xs=range(-0.5,
10.5, length=400), ylim=(-2, 8))
```

```
b = [5.0, -2.0, 0.2]
σ = 0.5

b̂ = [5.33668986461604, -0.5338812683011727, -4.871744833070539, 7.3085706
15940025, -5.416405452144519, 2.2571387538141496, -0.5598990987201361, 0.
08441719974829395, -0.007584268101515659, 0.0003731377363355991, -7.73493
6121594632e-6]
û = 0.5404023143393804
```

Out[5]:



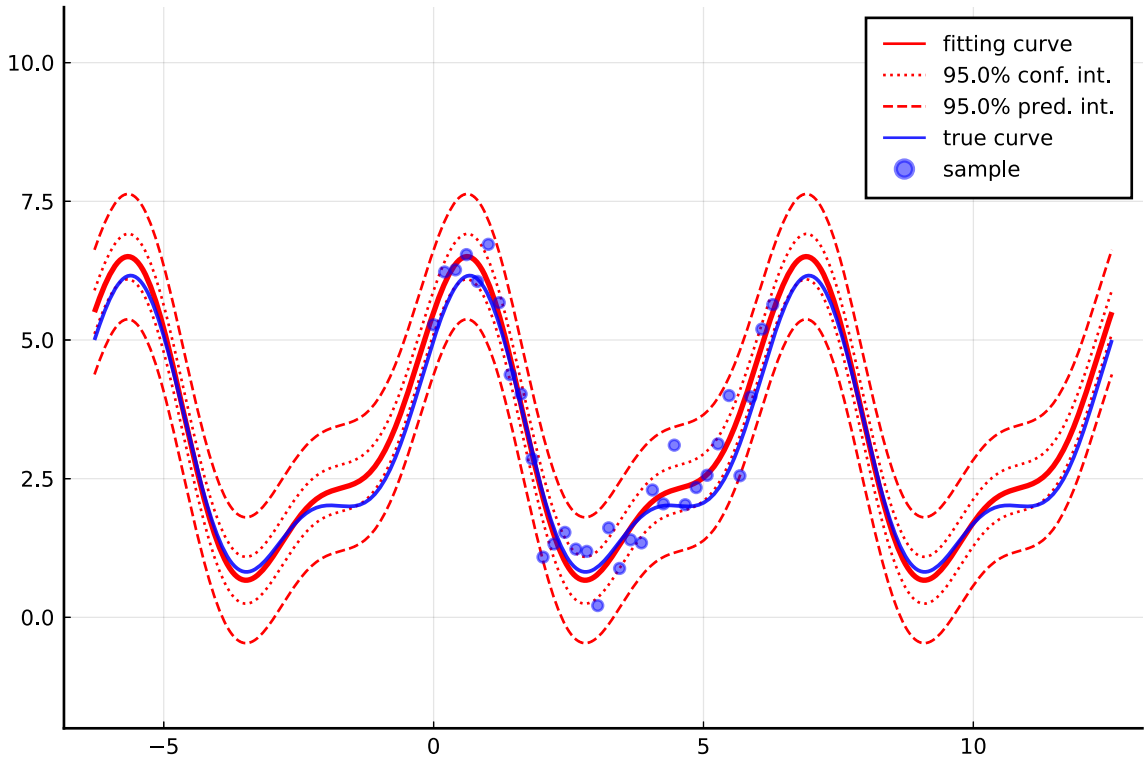
/

In [6]:

```
n = 2^5
x = range(0, 2π, length=n)
F = [one, cos, sin, x->cos(2x), x->sin(2x)]
b = Float64[3; 2; 1; 0; 1]
σ = 0.5
f_true = reg_func(F, b)

seed!(4649)
y = f_true.(x) + σ*randn(n)

plot_linear_regression(F, x, f_true; b=b, σ=σ, y=y, xs=range(-2π,
4π, length=400), ylim=(-2, 11))
```

```
b = [3.0, 2.0, 1.0, 0.0, 1.0]
σ = 0.5

b̂ = [3.1995778060952005, 2.3109966330394434, 0.8370849374587023, -0.00831
6559757708682, 0.9939991872831458]
û = 0.5130552224456036
```

Out[6]:

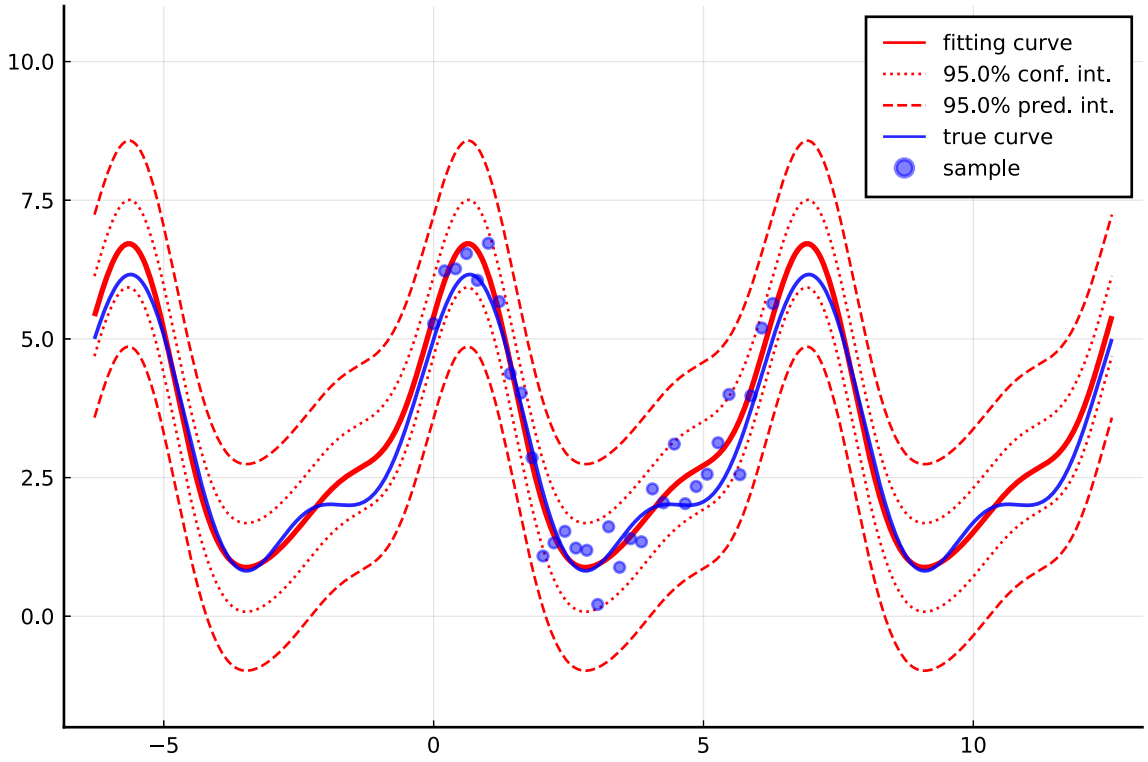


In [7]:

```
1  G = [one; cos; sin; x->cos(2x); x->sin(2x); [[x->cos(3x), x->sin(3x)] for k in 3:10]...]
2  plot_linear_regression(G, x, f_true; b=b, σ=σ, y=y, xs=range(-2π, 4π, length=400), ylim=(-2, 11))
```

```
b = [3.0, 2.0, 1.0, 0.0, 1.0]
σ = 0.5

b̂ = [3.202517756360168, 2.316876533569378, 0.8370849374587022, -0.0024366592277740933, 0.993999187283146, -0.013229776192353014, 0.02253901549258018, -0.013229776192352742, 0.02253901549258017, -0.013229776192352742, 0.02253901549258017, -0.013229776192352742, 0.02253901549258017, -0.013229776192352742, 0.02253901549258017, -0.013229776192352742, 0.02253901549258017, -0.013229776192352742, 0.02253901549258016, -0.013229776192352742, 0.02253901549258013, -0.01322977619235287, 0.02253901549258013]
û = 0.7639536515803761
```

Out[7]:

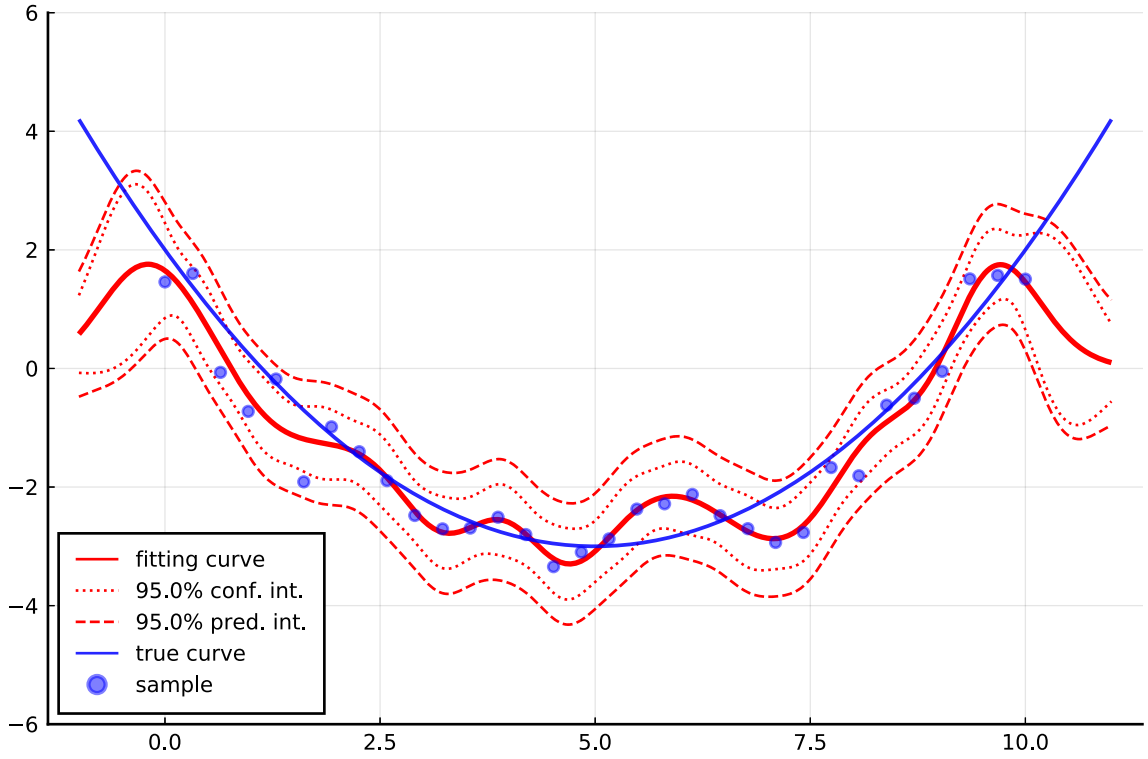


In [8]:

```
n = 2^5
x = range(0, 10, length=n)
F = [x -> exp(-(x-μ)^2/(2*0.5^2)) for μ in range(minimum(x)-0.3,
maximum(x)+0.3, length=n÷2)]
f_true = (x -> 2 - 2x + 0.2x^2)
y = f_true.(x) + 0.5randn(n)

plot_linear_regression(F, x, f_true; y=y, xs=range(-1, 11,
length=400), ylim=(-6, 6))
```

```
b̂ = [1.5118375436966138, 0.6234362385784558, -0.699258801355383, -0.72456
03654914766, -0.6632613569275949, -2.1622845357208265, -0.819667273619387
7, -2.4829103865547117, -1.2092123634620573, -1.055323774357349, -1.64418
44761647606, -1.863504146956927, -0.18314801288770247, -0.795916930409831
7, 1.9272753733479218, 0.16446459124198962]
û = 0.39112463183254637
```

Out[8]:



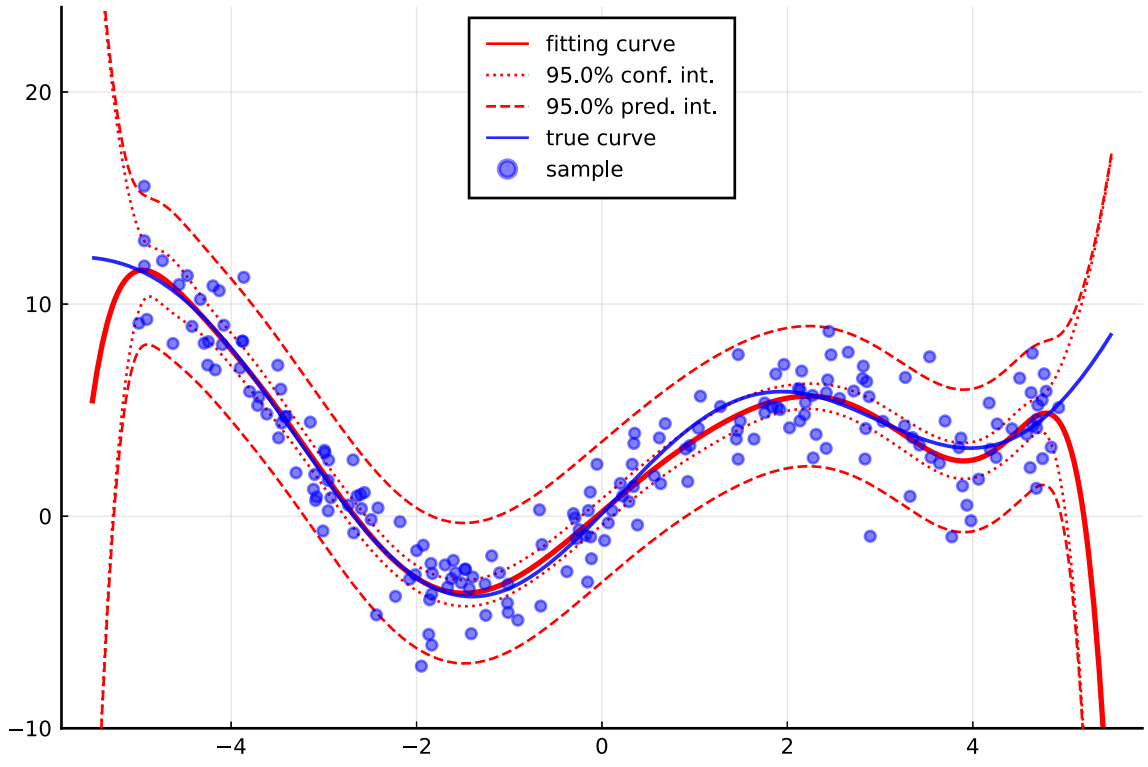
/

In [9]:

```
n = 200

b = [0.12, 0.23, 0.34, 4.32]
σ = 1.5
f_true = (x ->  b[1] + b[2]*x + b[3]*x^2 + b[4]*sin(x))

seed!(102)
x = rand(Uniform(-5, 5), n)
y = f_true.(x) + σ*randn(n)

F = [x -> x^k for k in 0:10]
plot_linear_regression(F, x, f_true; y=y, xs=range(-5.5, 5.5,
length=400), ylim=(-10, 24))
```

```
b̂ = [0.20778205316103399, 3.7278268225910036, -0.07283288009344126, -0.39
117838960872936, 0.14626759325482752, -0.008041165605993737, -0.017797091
59464472, 0.0014974050119800778, 0.0008739417985579831, -3.35062255543331
2e-5, -1.488657181502442e-5]
û = 1.6487068513596623
```

Out[9]:

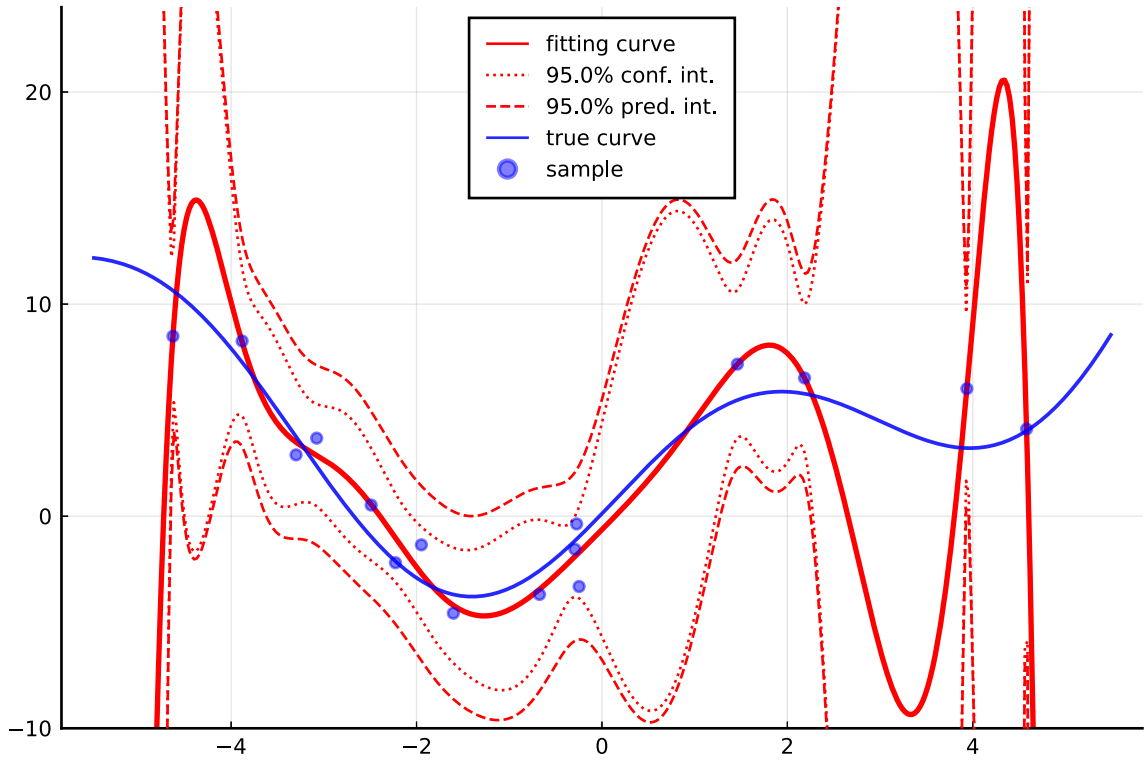


In [10]:

```
n = 16

b = [0.12, 0.23, 0.34, 4.32]
σ = 1.5
f_true = (x ->  b[1] + b[2]*x + b[3]*x^2 + b[4]*sin(x))

seed!(102)
x = rand(Uniform(-5, 5), n)
y = f_true.(x) + σ*randn(n)

F = [x -> x^k for k in 0:10]
plot_linear_regression(F, x, f_true; y=y, xs=range(-5.5, 5.5,
length=400), ylim=(-10, 24))
```

```
b̂ = [-0.6360301895154846, 4.417028707038097, 0.24209604165307497, 0.32558
59803291357, 0.6006383962282419, -0.29639295579868763, -0.158866237142891
03, 0.02680240754901261, 0.012358089093270274, -0.000666093706055679, -0.
00029238834600292903]
σ̂ = 1.357400058296853
```

Out[10]:



In [ ]:

```
1
```