知的システム論第 7 回レポート

37186305
航空宇宙工学専攻修士一年
荒居秀尚

2018 年 11 月 20 日

1　宿題 1

Julia 1.0.0 により実装した。

```
using Plots
using LinearAlgebra
using StatsBase
using IterTools
using Printf

function rmse(yobs, ypred)
   sqrt(sum((yobs - ypred).^2))
end

function kernelLinear(x, y, h, l)
    n = size(x)[1]
    x2 = x.^2
    k = exp.(-(repeat(x2, 1, n) + repeat(x2', n, 1) - 2 .* (x * x')) ./ (2 * h^2))
    t = inv(k^2 + l .* I) * (k * y)
end

function calcValidation(X, x, t, h)
    X2 = X .^ 2
    x2 = x .^ 2
    hh = 2 * h .^ 2
    K = exp.(-(repeat(X2, 1, length(x)) + repeat(x2', length(X), 1) - 2 .* (X * x')) ./ hh)
    F = K * t
end

function getCVIndice(maxIndice, numCV)
    x = collect(1:maxIndice)
    train = []
    remains = copy(x)
    nsample = Int(maxIndice / numCV)
    for i in 1:numCV
        s = sample(remains, nsample, replace=false)
        append!(train, [s])
        remains = setdiff(remains, s)
    end
    valid = [setdiff(x, t) for t in train]
    indices = [(t, v) for (t, v) in zip(train, valid)]
end

function CV(X, Y, h, l, numCV)
    len = size(X)[1]
    indices = getCVIndice(len, numCV)
    loss_array = []
    for (train_indices, valid_indices) in indices
        x = X[train_indices]
        Xnew = X[valid_indices]
        y = Y[train_indices]
        Ynew = Y[valid_indices]
        t = kernelLinear(x, y, h, l)
        F = calcValidation(Xnew, x, t, h)
        loss = rmse(Ynew, F)
        append!(loss_array, loss)
    end
    sum(loss_array) / size(loss_array)[1]
end

N = 1000
X = range(-3, stop=3, length=N)
piX = pi .* X
Y = sin.(piX) ./ (piX) + 0.1 .* X + 0.2 .* randn(N, 1)
ytrue = sin.(piX) ./ (piX) + 0.1 .* X

h = 0.3
l = 0.3
l_mean = CV(X, Y, h, l, 10)

h_array = collect(range(0.01, stop=1.0, length=100))
l_array = collect(range(0.01, stop=1.0, length=100))
loss_array = []
param_array = [(h, l) for (h, l) in Iterators.product(h_array, l_array)]
for (h, l) in Iterators.product(h_array, l_array)
    l_mean = CV(X, Y, h, l, 10)
    append!(loss_array, l_mean)
end

min_idx = argmin(loss_array)
best = param_array[min_idx]
```

```julia

println("Best Parameters h: $(best[1]) l: $(best[2])")

h_sample = [0.1, 0.74, 2.0]
l_sample = [0.001, 0.11, 5.0]
plots_array = []
l_array
for (h, l) in Iterators.product(h_sample, l_sample)
    l_mean = CV(X, Y, h, l, 10)
    n = length(X)
    s = sample(1:n, 100, replace=false)
    x = X[s]
    y = Y[s]

    t = kernelLinear(x, y, h, l)
    F = calcValidation(X, x, t, h)
    str = @sprintf "Loss: %.3f" l_mean
    lstr = @sprintf "l: %.3f" l
    hstr = @sprintf "h: %.3f" h
    p = scatter(
        X, Y,
        label="observed",
        color=RGB(150/255, 150/255, 220/255),
        legend=:top,
        size=(360, 240),
        legendfontsize=6,
        annotations=[
            (2, 1.25, text("$(str)", 6)),
            (2, 1.15, text("$(lstr)", 6)),
            (2, 1.05, text("$(hstr)", 6))
        ]
    )
    plot!(p, X, F, xlabel="X", ylabel="Y", label="predicted", linewidth=3,
       color=:red)
    plot!(p, X, ytrue, label="actual", linewidth=3, color=:green)
    push!(plots_array, p)
    push!(l_array, l_mean)
end


p2 = plot(
    plots_array[1],
    plots_array[2],
    plots_array[3],
    plots_array[4],
    plots_array[5],
    plots_array[6],
    plots_array[7],
    plots_array[8],
    plots_array[9],
    size=(1080, 720)
)
png("fitting_")
```

結果は図 1.1 のようになった。乱数のシード値を固定していなかったため、ややばらつくが、$h : 0.74$、$l : 0.11$ の時が最適であった。

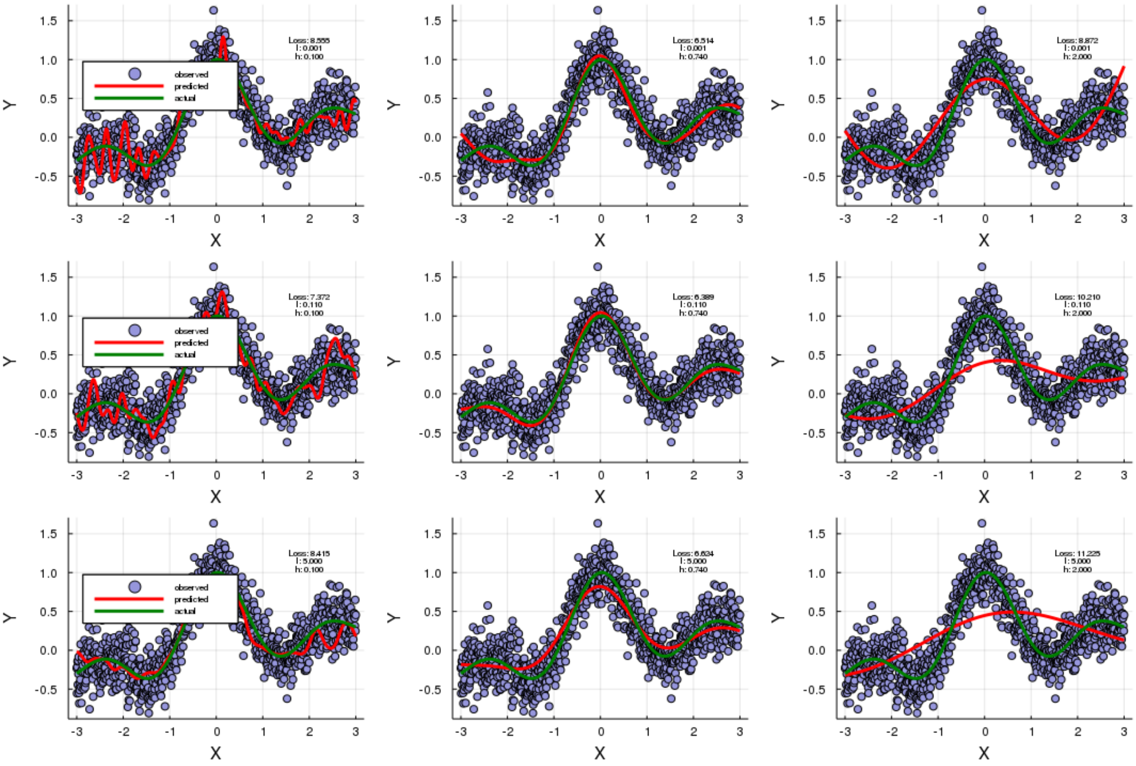

図 1.1: パラメータを変化させた結果

2　宿題 2

Julia 1.0.0 で実装した。

```
using Plots


mutable struct SVMResult
    n::Int64
    epsilon::Float64
    h::Float64
    C::Float64
    theta::VecOrMat{Float64}
    x::VecOrMat{Float64}
    y::VecOrMat{Float64}
    k::Matrix{Float64}
    converged::Bool
end

function SVM(
    x::Matrix{T},
    y::VecOrMat{T},
    C::Float64,
    h::Float64;
    epsilon::Float64=0.01,
    maxiter::Int64=10000,
    tol::Float64=1e-6) where T<:AbstractFloat

    n = Int(length(x) / 2)
    @assert n == length(y)
    xx = x[:, 1]
    xy = x[:, 2]
    xx2 = xx .^ 2
    xy2 = xy .^ 2
    k = exp.(
        -(
            (repeat(xx2, 1, n) + repeat(xx2', n, 1) - 2 .* (xx * xx')) +
            (repeat(xy2, 1, n) + repeat(xy2', n, 1) - 2 .* (xy * xy'))
        ) ./ (2 * h^2)
    )
    theta = rand(n, 1)
    converged = false
    cnt = 1
    while !converged && cnt < maxiter
        delThetaj = ifelse.(
            ones(n, 1) - (k * theta) .* y .> 0,
            - y .* k,
            0
        )
        sum_delTheta = sum(delThetaj, dims=1)'
        new_theta = theta - epsilon .* (C .* sum_delTheta + 2 .* (k * theta))
        if sqrt(sum((theta - new_theta).^2)) > tol
            theta = new_theta
            cnt += 1
        else
            theta = new_theta
            converged = true
        end
    end
    SVMResult(n, epsilon, h, C, theta, x, y, k, converged)
end

n = 200
```

```julia
a = collect(range(0, stop=4 * pi, length=Int(n/2)))
u = append!(a .* cos.(a), (a .+ pi) .* cos.(a)) + rand(n, 1)
v = append!(a .* sin.(a), (a .+ pi) .* sin.(a)) + rand(n, 1)
x = [u v]
y = append!(reshape(ones(1, Int(n/2)), (100,)), reshape(-ones(1, Int(n/2)), (100,)))

svm = SVM(x, y, 0.1, 0.3)
m = 100
X = collect(range(-15, stop=15, length=m))
X2 = X .^ 2
U=exp.(-(repeat(u.^2,1,m)+repeat(X2',n,1)-2 .* (u*X'))/ (2 * svm.h^2))
V=exp.(-(repeat(v.^2,1,m)+repeat(X2',n,1)-2 .* (v*X'))/ (2 * svm.h^2))
p = contour(X, X, sign.(V' * (U .* repeat(svm.theta, 1, m))),
    fill=true,
    fillcolor=:viridis,
    fillalpha=0.1,
    xlabel="X",
    ylabel="Y",
    xlims=(-15, 15),
    ylims=(-15, 15)
)
scatter!(p, u[1:100], v[1:100], y[1:100], color=:yellow, label="Class1", ticks=false)
scatter!(p, u[101:200], v[101:200], y[101:200], color=:red, label="Class2", ticks=false)
png("svm")
```

結果は、図2.1のようになった。

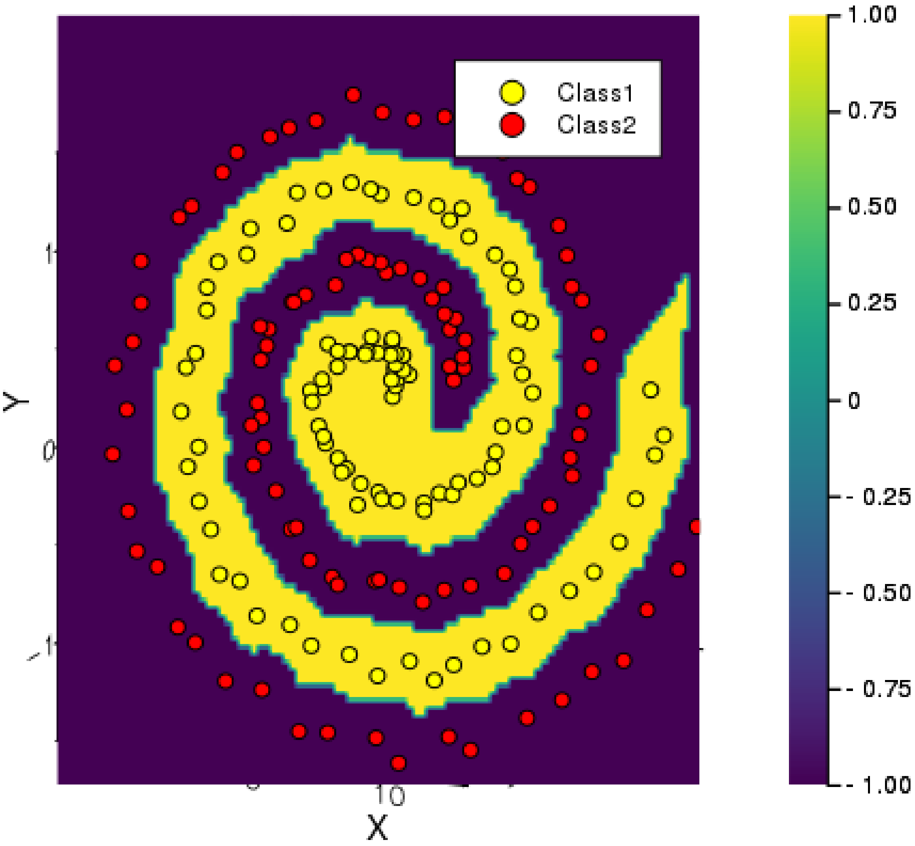

図 2.1: SVM による分類