DATA 1010
Problem Set 10
Due 16 November 2018 at 11 PM

Problem 1

Suppose that for each $i = 1, 2, \ldots, n$, the random vector $\mathbf{X}_i$ is chosen from a distribution on $\mathbb{R}^p$ and then Y_i is chosen from the Bernoulli distribution with probability $r(\mathbf{X}_i)$, where $r : \mathbb{R}^p \to [0, 1]$ is some function.

(a) Given $\mathbf{y} \in \{0, 1\}^n$, find $\mathbb{P}((Y_1, \ldots, Y_n) = \mathbf{y} \mid \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$.

(b) Suppose that multiple candidate r's are proposed. Show that finding the one which maximizes your answer to (a) is the same as finding the one that minimizes

$$\sum_{i=1}^{n} \left[y_i \log \frac{1}{r(X_i)} + (1 - y_i) \log \frac{1}{1 - r(X_i)} \right].$$

Solution

(a) For each i, the conditional probability given $\mathbf{X}_i$ of the event $\{Y_i = y_i\}$ is equal to $r(\mathbf{X}_i)$ if $y_i = 1$ or $1 - r(\mathbf{X}_i)$ if $y_i = 0$. Therefore,

$$\mathbb{P}((Y_1, \ldots, Y_n) = \mathbf{y} | \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n) = \prod_{y_i=1} r(\mathbf{X}_i) \prod_{i=0} (1 - r(\mathbf{X}_i)). \tag{1.1}$$

(b) Minimizing a quantity is the same as minimizing its logarithm, so we can replace (1.1) with

$$\sum_{y_i=1} \log r(\mathbf{X}_i) + \sum_{y_i=0} \log(1 - r(\mathbf{X}_i))$$

without changing the argmax. We can multiply all the terms in the first sum by y_i and in the second by $1 - y_i$ so that both sums can be written over all i values. Multiplying by -1 to change the maximization problem to a minimization problem yields

$$\sum_{i=1}^{n} \left[y_i \log \frac{1}{r(X_i)} + (1 - y_i) \log \frac{1}{1 - r(X_i)} \right],$$

as desired.

Problem 2

Given $f : \mathbb{R}^{m \times n} \to \mathbb{R}$, we define

$$\frac{\partial}{\partial A} f(A) = \begin{bmatrix} \frac{\partial f}{\partial a_{1,1}} & \cdots & \frac{\partial f}{\partial a_{1,n}} \\ \vdots & \ddots & \vdots \\ \frac{\partial f}{\partial a_{m,1}} & \cdots & \frac{\partial f}{\partial a_{m,n}} \end{bmatrix},$$

where $a_{i,j}$ is the entry in the ith row and jth column of A. Suppose that $\mathbf{u}$ is a $1 \times m$ row vector and $\mathbf{v}$ is an $n \times 1$ column vector. Show that

$$\frac{\partial}{\partial A} (\mathbf{u} A \mathbf{v}) = \mathbf{u}' \mathbf{v}'.$$

Solution

We have

$$\mathbf{u} A \mathbf{v} = a_{1,1} u_1 v_1 + a_{1,2} u_1 v_2 + \cdots + a_{m,n} u_m v_n.$$

Therefore, the derivative with respect to $a_{i,j}$ is $u_i v_j$. This is also the (i, j)th entry of $\mathbf{u}' \mathbf{v}'$, so the purported equality holds.

Problem 3

Train a QDA classifier for identifying a car as American or Japanese based on its weight and MPG rating. Use the `cars` dataset from the `VegaDatasets` package. Show the classification regions in different colors.

```
using VegaDatasets, Plots, DataFrames
D = DataFrame(dataset("cars"))
cars = [((x,y),c) for (x,y,c) in zip(D[:Miles_per_Gallon],
                                     D[:Weight_in_lbs],
                                     D[:Origin])
          if !any(ismissing.([x,y,c])) && c ≠ "Europe"]
x₁s = [x₁ for ((x₁,x₂),y) in cars]
x₂s = [x₂ for ((x₁,x₂),y) in cars]
ys = [y for ((x₁,x₂),y) in cars]
scatter(x₁s,x₂s,group=ys)
```

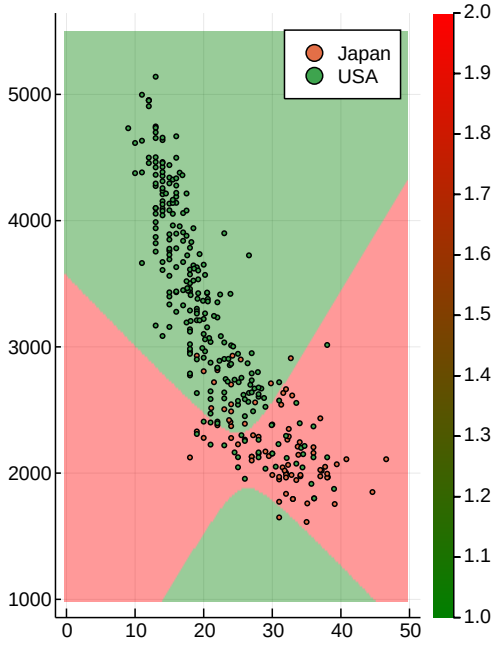
Solution

We follow the flowers example from the text. We begin by defining our own `Normal` struct and multivariate normal estimator.

```
using Plots, StatsBase
gr(match_dimensions=true,size=(350,440))
struct Normal
    μ::Vector
    Σ::Array
end
f(x,N::Normal) = 1/(2π*sqrt(det(N.Σ))) * exp(-1/2*((x-N.μ)'*inv(N.Σ)*(x-N.μ)))
function mvn_estimate(cars,country)
    cars_subset = [[x₁,x₂] for ((x₁,x₂),y) in cars if y == country]
    m̂  = mean(cars_subset)
    Ŝ  = mean([(X - m̂ )*(X - m̂ )' for X in cars_subset])
    Normal(m̂ ,Ŝ )
end
```

Next we define a function to compute the classifications and a function to plot them.

```
classify(x,ps,Ns) = argmax([p*f(x,N) for (p,N) in zip(ps,Ns)])
xgrid = range(0,stop=50,length=256)
ygrid = range(1000,stop=5500,length=256)
function classificationplot(cars,ps,Ns)
    P = heatmap(xgrid,
                ygrid,
                [classify([x,y],ps,Ns) for x=xgrid,y=ygrid],
                fillcolor=cgrad([:green,:red]),
                opacity=0.4)
    scatter!(P,[(x₁,x₂) for ((x₁,x₂),y) in cars],
             group=ys,
             markersize=2)
    P
end
countries = ["USA","Japan"]
countrycounts = countmap([y for ((x₁,x₂),y) in cars])
p̂ s = [countrycounts[y]/length(cars) for y in countries]
N̂ s = [mvn_estimate(cars,y) for y in countries]
classificationplot(cars,p̂ s,N̂ s)
```



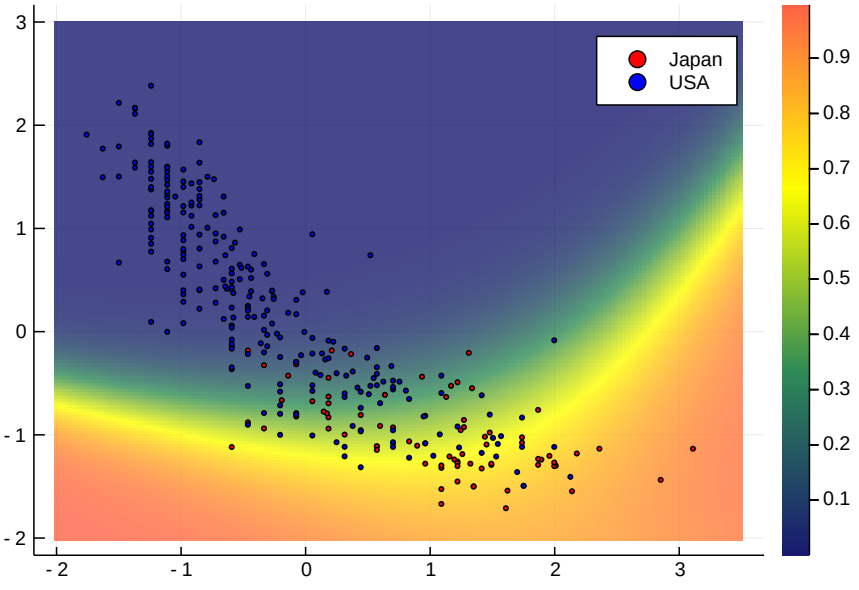
Problem 4

Train a logistic regression classifier for identifying a car as American or Japanese based on its weight and MPG rating. Include quadratic combinations of the regressors.

Solution

We first z-score the data, and then we use the code we developed in class. We insert the derived regressors in the definition of the function r.

```
using Optim
μ₁, σ₁ = mean(x₁s), std(x₁s)
x̃₁s = [(x₁-μ₁)/σ₁ for x₁ in x₁s]
μ₂, σ₂ = mean(x₂s), std(x₂s)
x̃₂s = [(x₂-μ₂)/σ₂ for x₂ in x₂s]
ỹs = [y == "USA" ? 0 : 1 for y in ys]
σ(u) = 1/(1 + exp(-u))
r(β,x) = σ(β⋅[1; x; x.^2; x[1]*x[2]])
C(β,xᵢ,yᵢ) = yᵢ*log(1/r(β,xᵢ))+(1-yᵢ)*log(1/(1-r(β,xᵢ)))
L(β) = sum(C(β,[x₁,x₂],y) for (x₁,x₂,y) in zip(x̃₁s,x̃₂s,ỹs))
β̂= optimize(L,ones(6),BFGS()).minimizer
xgrid = -2:1/2^5:3.5
ygrid = -2:1/2^5:3
r̂s = [r(β̂,[x,y]) for x=xgrid,y=ygrid]
mycgrad = cgrad([:MidnightBlue,:SeaGreen,:Yellow,:Tomato])
heatmap(xgrid,ygrid,r̂s,fillcolor=mycgrad,match_dimensions=true)
scatter!(x̃₁s,x̃₂s,group=ys,color=[:red :blue],markersize=2)
```



Problem 5

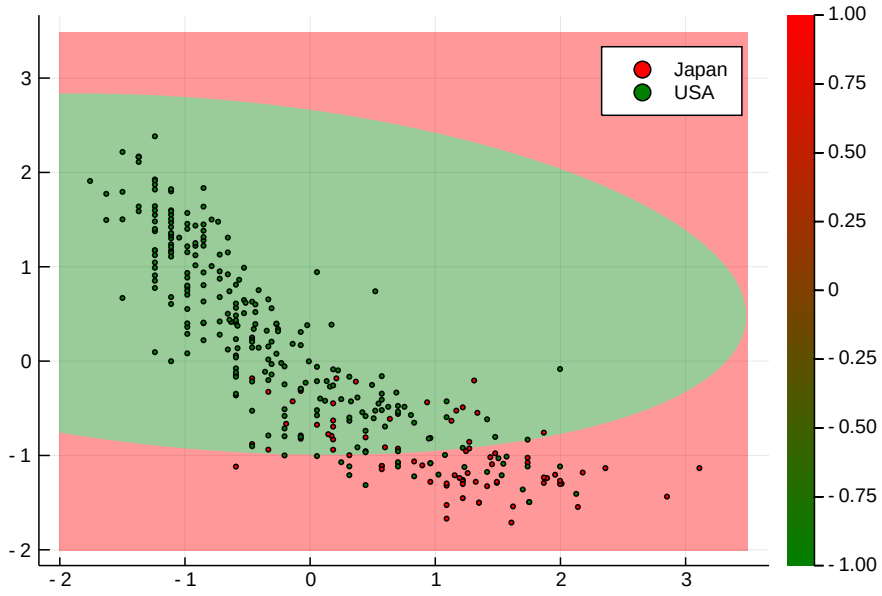
Train a support vector machine classifier for identifying a car as American or Japanese based on its weight and MPG rating. Include quadratic combinations of the regressors.

Solution

We follow the example from class. However, instead of optimizing over λ numerically (which takes a long time), inspect a variety of λ values and discover that we get the same misclassification rate (around 18%) for essentially all small λ values. The resulting graph is shown below.

Julia

```julia
using Optim
ỹs = [y == "USA" ? -1 : 1 for y in ys] # change the class encoding
samples = [([x₁,x₂,x₁^2,x₂^2,x₁*x₂],y) for (x₁,x₂,y) in zip(x̃₁s,x̃₂s,ỹs)];
L(λ,β,α,samples) = λ*norm(β)^2 + mean(max(0,1-y*(β⋅x - α)) for (x,y) in samples)
L(λ,params,samples) = L(λ,params[1:end-1],params[end],samples)
function SVM(λ,samples)
    params = optimize(params->L(λ,params,samples),
                      ones(1+length(first(first(samples)))),
                      BFGS()).minimizer
    params[1:end-1], params[end]
end
β,α = SVM(1e-9,samples)
heatmap(xgrid,
        ygrid,
        [sign(β⋅[x₁,x₂,x₁^2,x₂^2,x₁*x₂] - α) for x₁=xgrid, x₂=ygrid],
        match_dimensions=true, fillcolor = mycgrad, opacity=0.5)
scatter!(x̃₁s,x̃₂s,group=ys,color=[:red :blue],markersize=2)
```



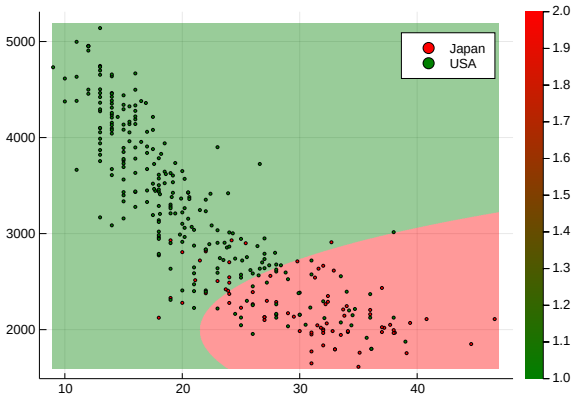
Problem 6

Train a Naive Bayes classifier for identifying a car as American or Japanese based on its weight and MPG rating. For estimating the marginal densities of each class, you may use kernel density estimation or assume that the distributions are Gaussian.

Solution

We begin by selecting the samples from each class, and we calculate the mean and covariance matrices for each component of each class separately. To simplify the code a little, we approximate the class covariance matrices, and then we set the diagonal entries to zero.

```julia
using Distributions
USA = hcat(x₁s,x₂s)[ys .== "USA",:]
Japan = hcat(x₁s,x₂s)[ys .== "Japan",:]
μ_USA, μ_Japan = [vec(mean(country,dims=1)) for country in (USA, Japan)]
Σ_USA, Σ_Japan = [cov(country) for country in (USA, Japan)]
for M in (Σ_USA, Σ_Japan)
    M[1,2] = 0
    M[2,1] = 0
end
mesh = 800
xgrid = range(9,stop=47,length=mesh)
ygrid = range(1600,stop=5200,length=mesh)
p_USA = mean(ys .== "USA")
p_Japan = mean(ys .== "Japan")
f_USA(x,y) = pdf(MvNormal(μ_USA, Σ_USA),[x,y])
f_Japan(x,y) = pdf(MvNormal(μ_Japan, Σ_Japan),[x,y])
heatmap(xgrid, ygrid,
    [argmax([p_USA * f_USA(x,y), p_Japan * f_Japan(x,y)]) for x=xgrid, y=ygrid],
    match_dimensions = true, fillcolor = cgrad([:green,:red]), opacity=0.4)
scatter!(x₁s,x₂s,group=ys,color=[:red :green],markersize=2)
```



Problem 7

Come up with your very own machine learning model. All 38 responses should be unique.

You should (a) specify your class $\mathcal{H}$ of admissible prediction functions, (b) specify your loss function, and (c) implement your model with some simulated data. Discuss how your model attempts to manage the bias-variance tradeoff. Feel free to restrict the problem type (classification vs. regression) and the input and output spaces $\mathcal{X}$ and $\mathcal{Y}$.