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 $\mathbb{P}((Y_1, \ldots, Y_n) = \mathbf{y} \,|\, \mathbf{X}_1, \mathbf{X}_2, \ldots, \mathbf{X}_n)$.

(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

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

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

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

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.

```julia
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)
```

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.

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.

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.

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 $\mathcal{Y}$.

Solution

Suppose $\mathcal{X} = [0,1]$ and $\mathcal{Y} = \mathbb{R}$. To solve the regression problem, we propose a piecewise linear model which penalizes the number of pieces: the class $\mathcal{H}$ of admissible functions is defined to be the set of continuous, piecewise linear functions from $[0,1]$ to $\mathbb{R}$. The loss functional $L(h)$ is defined to be

$$\mathbb{E}[(Y - h(X))^2] + \lambda N(h),$$

where $N(h)$ is the number of pieces in the definition of h (more precisely, one plus the number points where h is not differentiable), and λ is a parameter of the model. The empirical risk minimizer is the continuous piecewise linear function h which minimizes residual sum of squares plus $\lambda n N(h)$, where n is the number of samples.

Without the $N(h)$ term, this model would badly overfit, since we could achieve an empirical risk of zero by zigzagging through the samples (assuming no two of them have the same x-coordinate). Thus this model attempts to navigate the bias-variance tradeoff both through the choice of $\mathcal{H}$ *and* through the loss functional L.

We begin by simulating some data with a nonlinear regression function.

```julia
using Plots, JuMP, NLopt
r(x) = 1 - x^2 + x^6
xᵢs = [rand() for i=1:1000]
yᵢs = [r(xᵢ) + 0.05randn() for xᵢ in xᵢs]
```

Next we define a type for our piecewise linear functions. Such a function is specified by its starting point, its sequence of slopes, and the x-coordinates where it changes slope. We also define a call method for our `PiecewiseLinear` type (so that we can treat it like a function).

```julia
struct PiecewiseLinear
    slopes
    splitpoints
    yintercept
end
function (P::PiecewiseLinear)(x)
    xs = P.splitpoints
    value = P.yintercept
    for i=1:length(P.slopes)
        if x ≤ xs[i+1]
            return value + P.slopes[i]*(x-xs[i])
        else
            value += P.slopes[i]*(xs[i+1]-xs[i])
        end
    end
end
```

We will use a method which fixes the number of pieces n and finds the best fit for that number of pieces. This is a *constrained* optimization problem, since the order of the split points must be preserved. Therefore, we use `JuMP` instead of `Optim`.

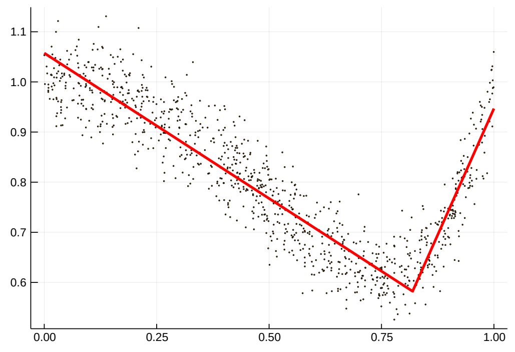
```
  function piecewisefit(xᵢs,yᵢs,n)
    m = Model(solver=NLoptSolver(algorithm=:LD_MMA))
    @variable(m,β[1:n],start=0) # define a variable for slopes
    @variable(m,x[i=1:n-1],start=i/n) # split points
    @variable(m,α,start=mean(yᵢs)) # y-intercept
    for i=1:n-2
        @constraint(m, x[i+1] ≥ x[i]) # enforce split point order
    end
    function RSS(v...) # take [β;x;α] together and compute RSS
        a = (length(v)-2)÷2
        h = PiecewiseLinear(v[1:a],v[a+1:end-1],v[end])
        sum((yᵢ - h(xᵢ))^2  for (xᵢ,yᵢ) in zip(xᵢs,yᵢs))
    end
    ex = Expr(:call,:RSS,
              [β[i] for i=1:n]...,
              0,[x[i] for i=1:n-1]...,1,
              α) # make an expression with all the variables
    JuMP.register(m,:RSS,2n+2,RSS,autodiff=true) # register RSS with JuMP
    JuMP.setNLobjective(m, :Min, ex) # set the objective function
    status = solve(m)
    status == :Optimal || error("Solver did not converge")
    r̂  = PiecewiseLinear(getvalue(β),[0;getvalue(x);1],getvalue(α))
end
```

We minimize our loss functional by stepping through the piece numbers n and stopping once we find that the empiricial risk has begun to increase. (This assumes that the minimum is unique, so it might not always identify the true minimum. Likewise, the optimization problem above is not convex, so finding the global minimum is not guaranteed). Note that we are taking $\lambda = \frac{1}{1000}$ for this example.

```
function piecewisefit(xᵢs,yᵢs)
    RSSes = []
    r̂s = []
    n = 0
    while true
        n += 1
        push!(r̂s,piecewisefit(xᵢs,yᵢs,n))
        push!(RSSes,sum((yᵢ - r̂s[end](xᵢ))^2  for (xᵢ,yᵢ) in zip(xᵢs,yᵢs)))
        if n > 1 && RSSes[end] + n > RSSes[end-1] + n-1
            return r̂s[end-1]
        end
    end
end
```

Finally, we plot our results for the simulated data.

```
function piecewiseplot(xᵢs,yᵢs,r̂ )
    xs = 0:1/2^10:1
    P = scatter(xᵢs,yᵢs,mark=1,color=:orange)
    plot!(xs,r̂ .(xs),color=:red,linewidth=3,legend=false)
    P
end
piecewiseplot(xᵢs,yᵢs,piecewisefit(xᵢs,yᵢs))
```



1.1
1.0
0.9
0.8
0.7
0.6
0.00
0.25
0.50
0.75
1.00