DATA 1010
Problem Set 9
Due 09 November 2018 at 11 PM

Problem 1

The *Epanechnikov* kernel is defined by

$$D(u) = \frac{3}{4}(1-u^2)\mathbf{1}_{|u|\leq 1}.$$

(a) Is D continuous? Is it differentiable? Is it twice differentiable?

(b) Is the tri-cube weight function continuous? Is it differentiable? Is it twice differentiable?

Feel free to use technology to perform the symbolic differentiation in this problem.

Solution

We calculate derivatives of the polynomial expressions in the two functions:

```
using SymPy
@vars u
subs(diff(1-u^2,u),u=>1)
subs(diff((1-u^3)^3,u),u=>1)
subs(diff((1-u^3)^3,(u,2)),u=>1)
```

We find that the Epanechnikov kernel is continuous but not differentiable at 1, since its derivative from the right is zero and its derivative from the left is negative. The tri-cube weight function is twice differentiable, since its first and second derivatives from the left are zero at 1.

Problem 2

Consider two random variables X and Y whose joint distribution has probability mass of $\frac{1}{n}$ at each of the n points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$ in $\mathbb{R}^2$. Show that the covariance matrix of X and Y is equal to

$$\frac{1}{n}\sum_{i=1}^{n}\begin{bmatrix} x_i - \overline{x} \\ y_i - \overline{y}\end{bmatrix}\begin{bmatrix} x_i - \overline{x} & y_i - \overline{y}\end{bmatrix}.$$

where $\overline{x} = (x_1 + \cdots + x_n)/n$ and $\overline{y} = (y_1 + \cdots + y_n)/n$.

Solution

The off-diagonal entries of the covariance matrix are each equal to

$$\mathbb{E}[(X-\mu_X)(Y-\mu_Y)]$$

where μ_X and μ_Y are the expected values of X and Y. Then using the formula $\mathbb{E}[g(X,Y)] = \sum_{(x,y)\in\mathbb{R}^2} g(x,y)m_{X,Y}(x,y)$, we find that

$$\mathbb{E}[XY] = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}),$$

Meanwhile, the off-diagonal entry of $\frac{1}{n}\sum_{i=1}^{n}[x_i - \overline{x}, y_i - \overline{y}][x_i - \overline{x}, y_i - \overline{y}]'$ is

$$\frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}),$$

which is indeed equal to the expression we found for $\mathbb{E}[XY]$. Similar analysis applies to the diagonal entries.

Problem 3

Suppose that the distribution of (X, Y) is uniform on the union of the rectangles $[0,3] \times [0,1]$ and $[0,3] \times [2,3]$.

(a) Find the regression function $r(x) = \mathbb{E}[Y \mid X = x]$.

(b) Generate 1000 samples from this distribution.

(c) Using the samples you generated, estimate the regression function $r(x)$ using a kernel density estimator with bandwidth λ selected by cross-validation.

(d) Find the Nadaraya-Watson estimator of $r(x)$, with λ selected by RSS cross-validation.

Solution

(a) Any vertical slice of the density yields a density which is symmetric about $\frac{3}{2}$. Therefore, the regression function is $r(x) = \frac{3}{2}$.

(b) We can generate a sample from the distribution of Y via `rand([0,2]) + rand()`, and we can generate 1000 samples from the joint distribution on X and Y via

```
samples = [(rand(),rand([0,2]) + rand()) for i=1:1000]
```

(c) We use our KDE code from class:

```
xs = 0:1/2^6:3
ys = 0:1/2^6:3

D(u) = abs(u) < 1 ? 70/81*(1-abs(u)^3)^3 : 0 # tri-cube function
D(λ,u) = 1/λ*D(u/λ) # scaled tri-cube
K(λ,x,y) = D(λ,x) * D(λ,y) # kernel
kde(λ,x,y,samples) = sum(K(λ,x-Xi,y-Yi) for (Xi,Yi) in samples)/length(samples)

function kdeCVterm(λ,i,samples)
    x,y = samples[i]
    newsamples = copy(samples)
    deleteat!(newsamples,i)
    kde(λ,x,y,newsamples)
end

function λCV(samples;λ₀=1.0)
    J(λ) = sum([kde(λ,x,y,samples)^2 for x=xs,y=ys])*step(xs)*step(ys) -
        2/length(samples)*sum(kdeCVterm(λ,i,samples) for i=1:length(samples))
    optimize(λ->J(first(λ)),[λ₀],BFGS()).minimizer[1]
end

λCV(samples,λ₀=0.25)
```

(d) Finally, we use cross-validation on RSS directly

```
function rssCV(λ,samples)
    sum((yᵢ - r̂(λ,xᵢ,[samples[1:k-1];samples[k+1:end]]))^2
        for (k,(xᵢ,yᵢ)) in enumerate(samples))
end
optimize(λ -> rssCV(first(λ),samples), [1.0], BFGS())
```

We find that the optimal value of λ is much larger for RSS cross-validation than for kernel density estimation. Since the regression function is constant, taking λ to be quite large helps average over more points and therefore produce a more accurate estimate of the regression function. Even though a large λ value does a poor job of estimating the original density, the RSS cross-validation is not checking for that.

Problem 4

The value of the Nadaraya-Watson estimator $\widehat{r}(x)$ can be thought of as the constant function which minimizes the weighted residual sum of squares, with the weight applied to each sample according to its horizontal distance from x. This optimization problem must be solved on a per-x basis, since the weights change for different values of x.

(a) Using the exam scores example, adjust this procedure by fitting a *line* for each point. In other words, for each $x \in [0, 20]$, find β_0 and β_1 such that

$$\sum_{i=1}^{n} D_\lambda(x - x_i)(y_i - \beta_0 - \beta_1 x_i)^2$$

is minimized (note that you are finding a new β_0 and β_1 for each value of x). Then set $\widehat{r}(x) = \beta_0 + \beta_1 x$. You may do this optimization using the `Optim` package.

(b) Plot the new estimator. Does it curl up at the ends of the interval like the Nadaraya-Watson estimator? Explain your intuition for why this is the case.

Solution

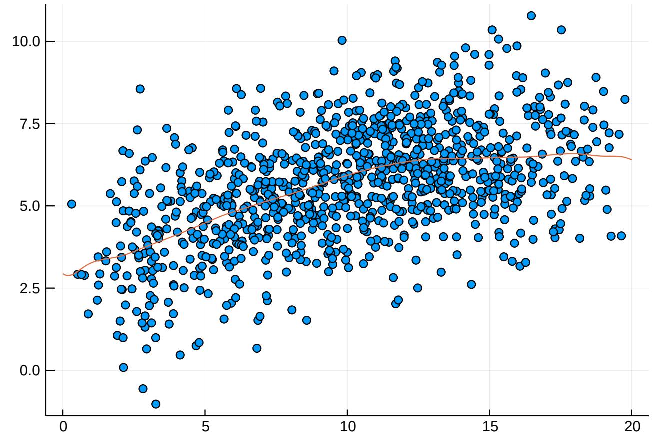
We begin by generating the samples using the code from class.

```
using Roots, Random; Random.seed!(1234)
n = 1000; r(x) = 2 + 1/50*x*(30-x); σ = 3/2
f(x,y) = 3/4000 * 1/sqrt(2π*σ^2) * x*(20-x)*exp(-1/(2σ^2)*(y-r(x))^2)
xs = 0:1/2^3:20; ys = 0:1/2^3:10
F(t) = -t^3/4000 + 3t^2/400
function sampleX()
    U = rand()
    find_zero(t->F(t)-U,(0,20),Bisection())
end
function sampleXY(r,σ)
    X = sampleX()
    Y = r(X) + σ*randn()
    (X,Y)
end
samples = [sampleXY(r,σ) for i=1:n]
```

Next we run the propose optimization problem for each x value:

```
D(u) = abs(u) < 1 ? 70/81*(1-abs(u)^3)^3 : 0 # tri-cube function
D(λ,u) = 1/λ*D(u/λ) # scaled tri-cube
weightedRSS(β,λ,x,samples) =
    sum(D(λ,x-xᵢ)*(yᵢ - β⋅[1,xᵢ])^2 for (xᵢ,yᵢ) in samples)
xs = 0:1/2^4:20
βmin(λ,x) = optimize(β -> weightedRSS(β,λ,x,samples), ones(2), BFGS()).minimizer
r̂s = [βmin(3.0,x) ⋅ [1,x] for x in xs]
scatter(samples)
plot!(xs,r̂s,legend=false)
```

If we experiment with a few different λ values, we see that the curve sometimes curls up at the ends and sometimes down. When λ is reasonably large, the curve is quite smooth and does not curl at the ends.



Problem 5

(a) Find the variance of the uniform distribution on the interval $[0, 10]$.

(b) Generate 10 independent samples from the uniform distribution, calculate the average $\overline{X}$ for those samples, and estimate the variance as $\widehat{V} = \frac{1}{n}\sum_{i=1}^{10}(X_i - \overline{X})^2$. Package this whole process as a function, and call it a million times to find the mean of $\widehat{V}$.

(c) Which is larger, the answer to (a) or the answer to (b)? Calculate the percent error.

Solution

We package the sampling procedure as a function and call it M times for $M = 10^6$:

```
function variance_estimate(n)
    X = [10*rand() for i=1:n]
    X̄  = mean(X)
    1/n * sum((x - X̄ )^2 for x in X)
end
M = 10^6
variance_estimate_mean = mean(variance_estimate(10) for i=1:M)
true_variance = 10^2 / 12
perc_error = (variance_estimate_mean - true_variance)/true_variance
```

We find that the variance estimate is lower than the true variance, with approximately 10% error. If we repeat for other values of M, we find that this percent error is not diminishing.

Problem 6

In this problem, we will implement a classifier for the flower data based on kernel density estimation.

(a) For each color, find the cross-validation kernel density estimator for the set of flowers of that color.

(b) Substitute your estimates into

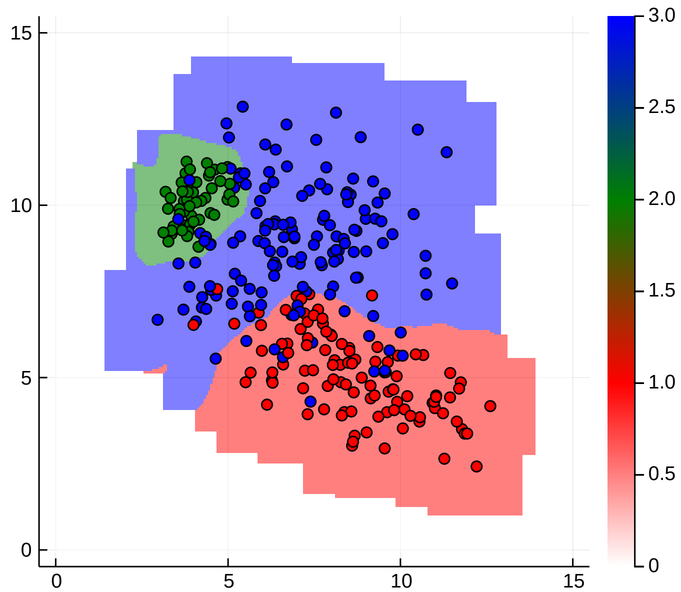
$$m_{(X_1,X_2)=(x_1,x_2)}(c) = \frac{p_c f_c(x_1, x_2)}{\sum_{d\in\{\mathrm{R,G,B}\}} p_d f_d(x_1, x_2)}, \tag{1.1.4}$$

to obtain a classifier (in the form of a Julia function).

(c) Make a plot similar to Figure 1.13 for this classifier.

Solution

```
using Plots, StatsBase, Random; Random.seed!(1234)
struct Normal
    μ::Vector
    Σ::Array
end
struct Flower
    X::Vector
    color::String
end
# density function for the normal distribution N
f(x,N::Normal) = 1/(2π*sqrt(det(N.Σ))) * exp(-1/2*((x-N.μ)'*inv(N.Σ)*(x-N.μ)))
xs = 0:1/2^4:15
ys = 0:1/2^4:15
As = [[1.5 -1; 0 1],[1/2 1/4; 0 1/2], [2 0; 0 2]]
μs = [[9,5],[4,10],[7,9]]
Ns = [Normal(μ,A*A') for (μ,A) in zip(μs,As)]
p = Weights([1/3,1/6,1/2])
colors = ["red","green","blue"]
function randflower(μs,As)
    i = sample(p)
    Flower(As[i]*randn(2)+μs[i],colors[i])
end
flowers = [randflower(μs,As) for i=1:300]
subsets = [[F.X for F in flowers if F.color == c] for c in colors]
# Find the cross-validated bandwidth for each color separately:
bestλs = [λCV(subset) for subset in subsets]
# Return a separate indicator if all the estimated densities are zero:
myargmax(x) = all( x .== 0 ) ? 0 : argmax(x)
classify(x,y) = myargmax([kde(λ,x,y,subset) for (λ,subset) in zip(bestλs,subsets)])
# Store the predicted classes in an array and plot
classes = [classify(x,y) for x=xs,y=ys]
heatmap(xs,ys,classes,color=cgrad([:white,:red,:green,:blue]),opacity=0.5,match_dimensions=true)
scatter!([(F.X[1],F.X[2]) for F in flowers],
         color=[F.color for F in flowers],aspect_ratio=:equal,legend=false)
```



Problem 7

(a) Show that if f_1, f_2 are different multivariate normal densities on $\mathbb{R}^2$, then the set of points (x, y) for which $f_1(x, y) = f_2(x, y)$ is a line or a conic section (in other words, it is the solution set of a linear or quadratic equation).

(b) Show that if the covariance matrices for the two densities are the same, then the solution set of $f_1(x, y) = f_2(x, y)$ is a line.

You might find this code block helpful.

```julia
using SymPy
@vars x y μ₁ μ₂ a b c real=true
Σ⁻¹ = [a c; c b]
v = [x - μ₁, y - μ₂]
expand(v' * Σ⁻¹ * v)
```

Solution

(a) The equation $f_1(x, y) = f_2(x, y)$ expands to

$$\frac{1}{\sqrt{\det \Sigma_1}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_1)'\Sigma_1^{-1}(\mathbf{x}-\boldsymbol{\mu}_1)} = \frac{1}{\sqrt{\det \Sigma_2}} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_2)'\Sigma_2^{-1}(\mathbf{x}-\boldsymbol{\mu}_2)},$$

where $\mathbf{x} = [x, y]$. Taking the log of both sides and combining, we find that the equation takes the form

$$(\mathbf{x} - \boldsymbol{\mu}_1)'\Sigma_1^{-1}(\mathbf{x} - \boldsymbol{\mu}_1) + (\mathbf{x} - \boldsymbol{\mu}_2)'\Sigma_2^{-1}(\mathbf{x} - \boldsymbol{\mu}_2) = c, \tag{7.1}$$

where c is some constant. The left-hand side of this equation is a polynomial in x and y of degree 2 or less, so its solution set is either a conic section or a line.

(b) Running the suggested code block and inspecting the result, we find that none of the quadratic terms involve μ_1 or μ_2. Therefore, if $\Sigma_1 = \Sigma_2$, then all of the quadratic terms in (7.1) cancel and we're left with a linear equation in x and y.

Problem 8

Consider a binary classification problem where conditional density of class 0 is uniform on the left half of the unit square and the conditional density of class 1 is uniform on the right half of the unit square. Devise a learner which is **maximally overfit** in the sense that its training error is zero and its generalization error is maximal (that is, the learner gets every classification wrong, with probability 1).

Solution

The learner must return class 0 for every class-0 training sample, and likewise for class 1. If we return class 1 for every other point in the left half of the unit square and class 0 for every other point in the right half of the unit square, then the learner will classify every test point incorrectly, since the probability that a test point coincides with one of the training points is zero.