# 16831 Statistical Techniques, Fall 2014: Problem Set 4

**Name:**

**Due:** Tuesday, December 9, 11:59 pm EST (Email)

## 1 RKHS

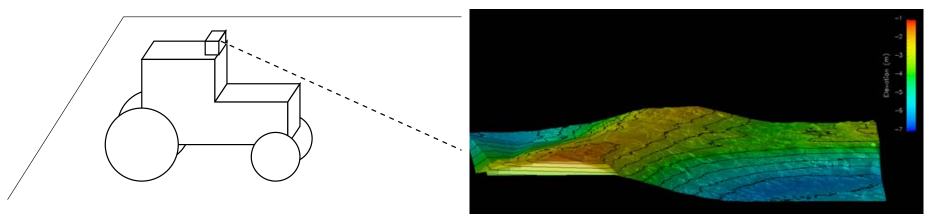

Figure 1: Left: Tractor with ladar used to detect and upper bound the terrain supporting surface. (image from Wellington and Stentz, 04). Right: A 3D representation of resulting terrain using a kernel algorithm.

Let's explore using a kernel function (like that above) for modeling terrain supporting surface as an alternative to the use of a factor graph model for ground plane estimation. We'll model terrain height as a function in a RKHS, that is $f([x, y]) = \sum_i \alpha_i k([x, y]_i, [x, y])$. Assume we get a ray from a laser that contacts the ground supporting surface. That provides us two pieces of information about the surfaces:

a) First, suppose the beam returns from the ground at point $[x_i, y_i]_{\text{end}}$. Write a loss function (for just this point) on the terrain surface that captures this information. $l_t =$. (Hint: Make it simple– you'll need to use it as part of an update later.)

b) Second, suppose the beam starts at point $[x_i, y_i]_{\text{start}}$. Given that the terrain supporting surface for the robot is a function, we know that besides telling us a point where the surface is, the beam provides an upper bound on where the terrain could be. Can you write down a simple loss function that penalizes the terrain surface that lies above the ray $([x_i, y_i]_{\text{start}}, [x_i, y_i]_{\text{end}})$? (Hint: consider only the **largest** violation of the constraint that the surface lies below the beam.)

c) Let's write a kernel gradient descent algorithm that minimizes these loss functions. What is the functional gradient of the sum of (a) and (b) above? (Hint: Remember that in general, the functional gradient will look like $\nabla_f l_t(f) = l'_t(f([x_i, y_i]_t))k([x_i, y_i], \cdot)$ where the prime symbol means the regular derivative.
If you are confused by the functional gradient method, assume instead that the terrain supporting surface can be written as a linear combination of features/basis $b$ (perhaps fourier functions or wavelets) as $f([x_i, y_i]) = w^T b([x_i, y_i])$. Then compute the standard sub-gradient with respect to $w$ instead.

d) Explain intuitively how such an algorithm works.

## 2 Gaussian Processes

Stranded on a desert island, the key to your survival is one last regression problem. Fortunately, you brought your Gaussian Processes reading on your ill-fated flight, and even more fortunately the plane did not have in-flight entertainment so you studied in in great detail this time.
Given the data $x = \{-2, 1, 2\}$ and $y = \{0.5, -2, 1\}$, you need to come up with a mean estimate for arbitrary locations $x^* \in [-8 \ldots 8]$. You may plot variance estimates if you wish, but are not required to. Assume a prior mean of 0 on all y values, and assume some noise.
*Note*: We are not looking for exact answers, your survival depends on your ability to reason about and understand kernel functions, not invert 3x3 matrices (trust us). You are welcome to write a computer program to verify your intuition (not vice versa).

a) But before plotting, answer this one theoretical question: To be a covariance function, or equivalently a kernel function, $k(x, y)$ must be positive definite. What does this mean?

b) RBF/Squared Exponential Kernel: $k(x, x') = \exp\left(-\frac{||x-x'||^2}{\ell^2}\right)$ for the length-scales 1.0, 0.2, and 5.0 (please write each length scale on its plot).

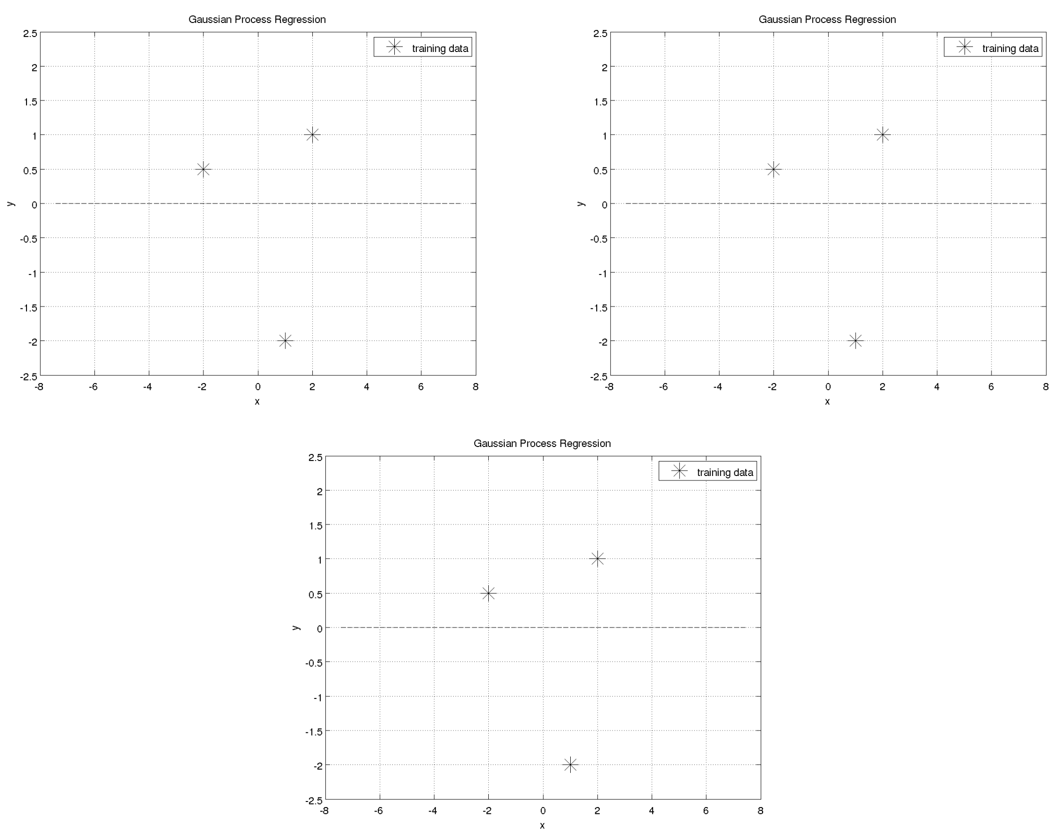


Figure 2: Squared Exponential / RBF Kernel

c) $k(x, x') = x^T x'$:

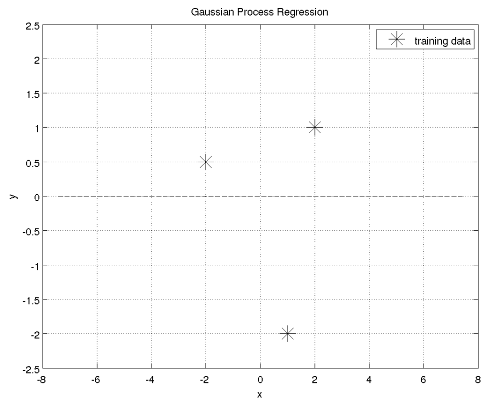


Figure 3: Linear Kernel

d) $k(x, x') = 1$:

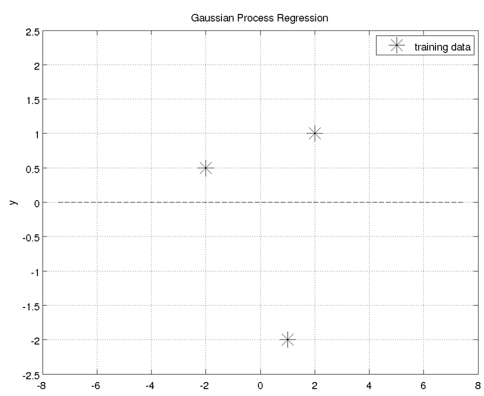


Figure 4: Constant Kernel

e) $k(x, x') = \mathbf{1}(x == x')$:

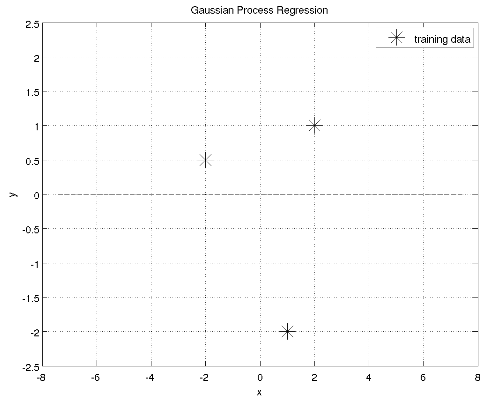


Figure 5: Indicator Kernel

Here are spare plots. If you use these, clearly mark which kernel you are plotting.

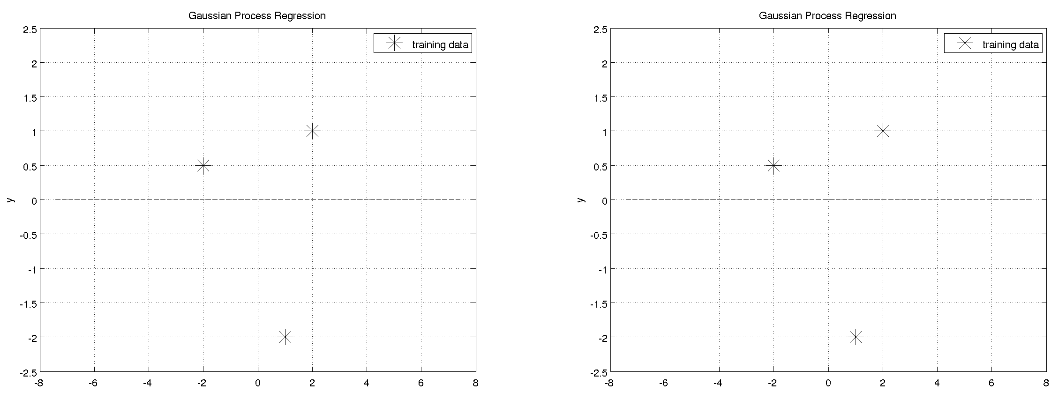


Figure 6: Spare plots.

f) What would happen if you tried to parameterize a Gaussian Process in the "natural" parameterization. Can you make something like this work? Why or why not?