A1: Conceptual Questions

A.1.a

Decraese σ. This makes the function $\exp\left(-\frac{\|u-v\|_2^2}{2\sigma^2}\right)$ thinner, make the inner product between different points more distinct.

A.1.b

True. It's an non-convex objective function, assuming thta deep means more than one hidden layer of course and assuming that activation function is not linear.

A.1.c

A.1.d

Flase, the what gives non linear decision boundary is the number of layers and the number of neurons in the layers.

A.1.e

A.2: Kernels And Bootstrap

Give na vector whose n th components paramterized by n is given by:

$$\frac{1}{\sqrt{n!}}\exp\left(\frac{-x^2}{2}\right)x^n$$

where x is one dimensional, and the feature mapping function $\phi(x)$ is an infinite dimensional function. Then:

$$\begin{aligned}
\langle\phi(x),\phi(y)\rangle &= \sum_{n=1}^{\infty}\phi_n(x)\phi_n y \qquad \text{(A.2.1)}\\
&= \sum_{n=1}^{\infty}\frac{1}{\sqrt{n!}}\exp\left(-\frac{x^2}{2}\right)x^n\frac{1}{\sqrt{n!}}\exp\left(-\frac{y^2}{2}\right)y^n\\
&= \sum_{n=1}^{\infty}\frac{(xy)^n}{n!}\exp\left(-\frac{x^2+y^2}{2}\right)\\
&= \exp\left(-\frac{x^2+y^2}{2}\right)\sum_{n=1}^{\infty}\frac{(xy)^n}{n!}\\
&= \exp\left(-\frac{x^2+y^2}{2}\right)\exp(xy)\\
&= \exp\left(-\frac{x^2+y^2}{2}+\frac{2xy}{2}\right)\\
&= \exp\left(\frac{-(x-y)^2}{2}\right)
\end{aligned}$$

And this is the RBF kernel for a scalar, in the 1d case.

A.3: Kernel Ridge Regression

A.3.a

To implement, we will need to take care of the process of solving for the best parameter α, and we will also need to careful about the offset. So what we are going to train is on the zero mean data, and then the prediction made from the model will have to add back the offset from the training set of course.
Here is basically what we had from the sections:

$$
\begin{aligned}
\frac{1}{2}\nabla_x[\|K\alpha - y\|_2^2 + \lambda\alpha^T K\alpha] &= 0 \\
\implies K^T(K\alpha - y) + \lambda K\alpha &= 0 \\
K(K\alpha - y) + \lambda K\alpha &= 0 \\
KK\alpha - Ky + \lambda K\alpha &= 0 \\
KK\alpha + \lambda K\alpha &= Ky \\
K\alpha + \lambda\alpha &= y \\
\alpha &= (K + \lambda I)^{-1}y
\end{aligned} \tag{A.3.a}
$$

Where, K and y are from the training set. And in this case, the predictor can be computed via: $K_{\text{test,train}}\alpha$ where the $K_{\text{test,train}}$ is computed via:

$$K_{i,j} = \langle\phi(X_{\text{test}}[i,:]), \phi(X_{\text{train}}[:,j])\rangle$$

A.3.b

To train the model and get the best hyperparameter, here are some of my strategies:

1. Use "sklearn.metrics.pairwise" to compute the kernel matrix. Cause I am lazy and stupid so I just want smart people to write the code for me.
2. Use "sklearn.modelselection" to do cross val cause I am lazy.
3. Use exactly the same training and validation for all hyper parameter traning (n = 30). And there is no test set.
4. Linear search on the degree of the poly kenel, bruteforcing out the one with minimal error across the validation set.
5. Section search on Gamma for the gaussian kernel. Partition a range into fine little sections, and use scipy optimizer to look for the minimal on all tiny little section. And then bruteforce out the optimal by comparing the best results from each section. Here we make the assumption that the crossval error function is at least, smooth and all of its derivatives are bonded.

A.3.c

A.3.d

A.3.e