2. a) Gaussian Kernel for one-dimensional inputs $x,\ x' \in \mathbb{R}$

$$K(x, x') = \exp\left(\frac{-(x-x')^2}{2\sigma^2}\right) \qquad \alpha = \frac{1}{\sqrt{2\sigma^2}}$$

Taylor Series expansion of exponential func:

$$e^{-u} = \sum_{n=0}^{\infty} \frac{(-u)^n}{n!}$$

Substituting $u$ with $\alpha^2(x-x')^2 \rightarrow$ series expansion for the Gaussian Kernel

Mapping function $\phi(x)$ in space $\mathcal{H}$: $\phi(x) = (1, \alpha x, \alpha^2 x^2, \alpha^3 x^3, \dots)$

Each term $(\alpha x)^n$ corresponds to the term in the taylor series expansion of the Gaussian Kernel

b) multi dimensional inputs $x,\ x' \in \mathbb{R}^d$ with $\|x\|^2 = \|x'\|^2 = 1$

Gaussion Kernel:

$$K(x, x') = \exp\left(\frac{-\|x-x'\|^2}{2\sigma^2}\right)$$

$$\|x - x'\|^2 = 2 - 2x^T x' \leftarrow \|x\|^2 = \|x'\|^2 = 1$$

using $\alpha = \frac{1}{\sqrt{2\sigma^2}}$ & Taylor exponsion:

$$K(x, x') = \sum_{n=0}^{\infty} \frac{1}{n!}\left(-\alpha^2(2 - 2x^T x')\right)^n$$

Mapping Func $\phi(x)$ in space $\mathcal{H}$:

$$\phi(x) = (1, \alpha^2(2-2x^T x), \alpha^4(2-2x^T x)^2, \alpha^6(2-2x^T x)^3, \dots)$$

Each term $(\alpha^2(2-2x^T x))^n$ corresponds to the termin Taylor series expansion of the Gaussian kernel, reflecting the kernel's generalization to higher dimensions while keeping the norm of $x$ & $x' = 1$.

In both cases, the mapping func $\phi(x)$ translates the input data into an infinite-dimensional space where the dot product corresponds to the Gaussian kernel function.

3. a) $f(w)=\frac{1}{2}\|Xw-y\|^2+\frac{\lambda}{2}\|w\|^2$ $\lambda>0,\ X\in R^{n\times d},\ w\in R^d,\ y\in R^n$

$\nabla_w f(w)=X^T(Xw-y)+\lambda w=0 \rightarrow X^TXw+\lambda Iw=X^Ty$

$w=(X^TX+\lambda I)^{-1}X^Ty$

$X^TX+\lambda I$ is invertible for $\lambda>0$ bc if $X=U\Sigma V^T$ is the SVD of X, then $X^TX=V\Sigma^T\Sigma V^T$, & all eigenvalues of $X^TX$ are nonnegative. Adding $\lambda I$, where $I$: Identity matrix & $\lambda>0$, increases each eigenvalue by $\lambda$, making sure that all eigenvalues are positive which then makes sure that the matrix is invertible

b) rewriting the normal equation:

$X^TXw+\lambda Iw=X^Ty$

$\lambda w=X^Ty-X^TXw$

$w=\frac{1}{\lambda}X^Ty-\frac{1}{\lambda}X^TXw$

we can express w as $w=X^T\alpha$

where $\alpha\in R^n$ is a vector of coefficients, substituting back into eq, we get:

$\alpha=\frac{1}{\lambda}y-\frac{1}{\lambda}XX^T\alpha$

c) saying that w is in the span of the data means that w can be expressed as a linear combinaton of the columns of X, which are the features of the training data. Since $w=X^T\alpha$, this is the case

d) Substituting $w=X^T\alpha$ into the normal equation & solving for $\alpha$ we get

$\alpha=(\lambda I+XX^T)^{-1}y$

$XX^T=$ Gram (kernel) matrix for the standard vector dot product which is a $n\times n$ matrix

e) predicted values on the training points: $Xw=XX^T\alpha$

substituting a from d): $Xw=XX^T(\lambda I+XX^T)^{-1}y$

f) for a test sample x not in the training set, $w^Tx=\alpha^TXx$

since a can be expressed using kernel matrix & target values, we get

$w^Tx=y^T(\lambda I+XX^T)^{-1}Xx$

g) In Kernelized Ridge regression, the prediction for a new sample x uses the kernel function $K(x,x')=x^Tx'$ to compute the inner products btwn x & the training samples within the kernel matrix $K=XX^T$.

The predicted value is given by:

$w^Tx=y^T(\lambda I+K)^{-1}K(X,x)$

- $(\lambda I+K)^{-1}$ is the inverse of the kernel matrix regularized w/ $\lambda$
- $K(X,x)$ is the vector of kernel evaluations btwn x & training samples

4. a) $\alpha_i = \frac{1}{2} \log\left(\frac{1-\epsilon_i}{\epsilon_i}\right)$ $\epsilon_i$ = weighted training error of classifier $f_i$

misclassified = D2, 9, 10, 12, 13, 14

$\epsilon_1 = \frac{6}{16}$,

$\alpha_1 = \frac{1}{2} \log\left(\frac{1-6/16}{6/16}\right) = 0.2554$

b)

error of first decision stump $f_1$: $\epsilon_1 = 6/16$

weight of $f_1$: $a_1 = \frac{1}{2} \log\left(\frac{1-6/16}{6/16}\right)$

updates:

incorrectly classified instances: $W_{new, incorrect} = \frac{1}{16} e^{\alpha_1}$

correctly classified instances: $W_{new, correct} = \frac{1}{16} e^{-\alpha_1}$

Normalize weights:

- compute sum of all updated weights
- divide each weight by sum to normalize, ensuring that the total weight accross all instances equals 1

c) Technically the iterations might contine but it's better to stop boosting when a weak classifier achieves a 0 error rate on weighted training data. This prevents overfitting & maintains a balance btwn bias & variance, which is needed for good generalization