A. Kernels

1. Show that the dimension of the feature space associated to the polynomial kernel of degree $d$, $K : \mathbb{R}^N \times \mathbb{R}^N \rightarrow \mathbb{R}, K(x, x') = (x \cdot x' + c)^d$, with $c > 0$, is

\[
\binom{N + d}{d}.
\] (1)

It is clear that $K$ can be written as a linear combinations of all monomials $x_1^{k_1} \cdots x_N^{k_N}$ with $\sum_{j=1}^{N} k_i \leq d$. The dimension of the feature space is thus the number of such monomials, $f(N, d)$, that is the number of ways of adding $N$ non-negative integers to obtain a sum of at most $d$. Note that any sum of $N - 1$ integers less than or equal to $d$ can be uniquely completely to be equal to $d$ by adding one more term. This defines in fact a bijection between sums of $N - 1$ integers with value at most $d$ and sums of $N$ integers with value equal to $d$. Thus, the number of sums of $N$ integers exactly equal to $d$ is $f(N - 1, d)$.

Now, since a sum of $N$ terms less than or equal to $d$ is either equal to $d$ or less than or equal to $d - 1$,

\[
f(N, d) = f(N - 1, d) + f(N, d - 1).
\]

The result then follows by induction on $N + d$, using $f(1, 0) = f(0, 1) = 1$.

Write $K$ in terms of kernels $k_i : (x, x') \mapsto (x \cdot x')^i$, $i \in [0, d]$. What is the weight assigned to each $k_i$ in that expression? How does it vary as a function of $c$.

By the binomial identity, $K(x, x') = (x \cdot x' + c)^d = \sum_{i=0}^{d} \binom{d}{i} c^{d-i} (x \cdot x')^i$. The weight assigned to each $k_i$, $i \leq d$, is thus $\binom{d}{i} c^{d-i}$. Increasing $c$ decreases the weight of $k_i$, particularly that of $k_i$s with larger $i$.

2. For $\alpha \geq 0$, the kernel $K_\alpha : (x, x') \mapsto \sum_{k=1}^{N} \min(|x_k|^\alpha, |x'_k|^\alpha)$ over $\mathbb{R}^N \times \mathbb{R}^N$ is used in image classification. Show that $K_\alpha$ is PDS. To do that, you can proceed as follows.

(a) Use the fact that $(f, g) \mapsto \int_0^\infty f(t)g(t)dt$ is an inner product over the set of measurable functions over $[0, +\infty)$ to show that $(x, x') \mapsto \sum_{k=1}^{N} \min(|x_k|^\alpha, |x'_k|^\alpha)$ is an inner product over the set of measurable functions over $[0, +\infty)$.
\( \min(x, x') \) is a PDS kernel (hint: associate an indicator function to \( x \) and another one to \( x' \)).

Observe that \( \min(|u|^\alpha, |u'|^\alpha) = \int_0^{\infty} 1_{t \in [0, |u|^\alpha]} 1_{t \in [0, |u'|^\alpha]} dt \), which shows that \( (u, u') \mapsto \min(|u|^\alpha, |u'|^\alpha) \) is PDS.

(b) Use the previous question to show that \( K_1 \) is PDS and similarly \( K_\alpha \) with other values of \( \alpha \).

Since \( K_\alpha(x, x') = \sum_{k=1}^N \min(|x_k|^\alpha, |x'_k|^\alpha) \), \( K_\alpha \) is PDS as a sum of \( N \) PDS kernels.

B. Boosting

1. Assume that the main weak learner assumption of AdaBoost holds. Let \( h_t \) be the base learner selected at round \( t \). Show that the base learner \( h_{t+1} \) selected at time \( t \) must be different from \( h_t \).

By the weak learning assumption, there exists a hypothesis \( h \in H \) whose \( D_{t+1} \)-error is less than half. Examine the empirical error of \( h_t \) for the distribution \( D_{t+1} \). Since \( Z_t = 2 \sqrt{\epsilon_t (1 - \epsilon_t)} \) and \( \alpha_t = \frac{1}{2} \log \frac{1 - \epsilon_t}{\epsilon_t} \),

\[
\hat{R}_{D_{t+1}}(h_t) = \sum_{i=1}^m \frac{D_t(i)e^{-\alpha_t y_i h_t(x_i)}}{Z_t} 1_{y_i h_t(x_i) < 0}
\]

\[
= \sum_{y_i h_t(x_i) < 0} \frac{D_t(i)e^{\alpha_t}}{Z_t}
\]

\[
= \frac{e^{\alpha_t}}{Z_t} \sum_{y_i h_t(x_i) < 0} D_t(i)
\]

\[
= \frac{\sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}}{2 \sqrt{\epsilon_t (1 - \epsilon_t)}} \epsilon_t = \frac{1}{2}.
\]

This shows that \( h_t \) cannot be selected at round \( t + 1 \).

2. Let the training sample be \( S = ((x_1, y_1), \ldots, (x_m, y_m)) \). Suppose we wish to penalize differently errors made on \( x_i \) versus \( x_j \). To do that, we associate some non-negative importance weight \( w_i \) to each point \( x_i \) and define the objective function \( F(\alpha) = \sum_{i=1}^m w_i e^{-y_i f(x_i)} \), where \( f = \sum \alpha_i h_i \) with the notation already used in class. Show that this function is convex and differentiable and use it to derive a boosting-type algorithm (give a clear description of the algorithm similar to that of AdaBoost presented in class).

For all \( \alpha, F(\alpha) = \sum_{i=1}^m w_i e^{-y_i f(x_i)} = \sum_{i=1}^m w_i e^{-y_i \sum_{t=1}^T \alpha_t h_t(x_i)} \), with \( w_i \geq 0 \). \( F \) is convex as a non-negative linear combination of convex functions and is
clearly differentiable. Applying coordinate descent to this function leads to the same algorithm as AdaBoost with the only difference that

$$D_1(i) \leftarrow \frac{w_i}{\sum_{i=1}^{m} w_i}. \quad (2)$$

C. Perceptron

1. The margin bound on the maximum number of updates presented in class for the perceptron algorithm was given for the special case \( \eta = 1 \). Consider now the general perceptron update \( w_{t+1} \leftarrow w_t + \eta y_t x_t \), where \( \eta > 0 \). With the same assumptions as for the theorem presented in class, prove a bound on the maximum number of mistakes. How does \( \eta \) affect the bound?

The bound is unaffected as shown by the following using the same definitions and steps as in the lecture slides:

$$M \rho \leq \frac{v \cdot \sum_{t \in I} y_t x_t}{\|v\|} = \frac{v \cdot \sum_{t \in I} (w_{t+1} - w_t) / \eta}{\|v\|} \quad \text{(definition of updates)}$$

$$= \frac{v \cdot w_{T+1}}{\eta \|v\|} \leq \frac{\|w_{T+1}\| / \eta}{(t_m \text{ largest } t \in I)}$$

$$= \frac{\|w_{t_m} + \eta y_{t_m} x_{t_m}\| / \eta}{(t_m \text{ largest } t \in I)}$$

$$= \left[ \left( \|w_{t_m}\|^2 + \eta^2 \|x_{t_m}\|^2 + \eta y_{t_m} w_{t_m} \cdot x_{t_m} \right) \right]^{1/2} / \eta$$

$$\leq \left[ \left( \|w_{t_m}\|^2 + \eta^2 R^2 \right) \right]^{1/2} / \eta \leq 0$$

$$\leq \left[ M \rho^2 R^2 \right]^{1/2} / \eta = \sqrt{MR} \quad \text{(applying the same to previous } t \text{ in } I).$$

2. Suppose each input vector \( x_t, t \in [1, T] \), coincides with the \( t \)th unit vector of \( \mathbb{R}^T \). How many updates does it take the perceptron algorithm to converge? How does the number of updates (or mistakes) compare with the margin bound?

Clearly, it takes \( T \) updates and leads to \( w = \sum_{t=1}^{T} y_t x_t \). Let \( u \in \mathbb{R}^T \) be a vector of norm 1 defining a separating hyperplane, thus \( y_t u \cdot x_t = y_t u_t \geq 0 \) for all \( t \in [1, T] \). To obtain the maximum margin \( \rho \), we seek a vector \( u \) maximizing the minimum of \( y_t u_t \) with \( y_t u_t \geq 0 \) for all \( t \) and \( \|u\| = 1 \). By symmetry, all \( y_t u_t \)s are equal, thus \( u_t = y_t / \sqrt{T} \) for all \( t \in [1, T] \) and \( \rho = 1 / \sqrt{T} \). Thus, Novikoff’s bound gives \( R^2 / \rho^2 = 1 / (1 / T) = T \).