1. We generalise the Perceptron Algorithm by introducing a learning rate $\eta > 0$. The update becomes

$$w_{t+1} = w_t + \eta \sigma_t y_t x_t.$$ 

Further, we start the algorithm with $w_1 = w_{\text{init}}$ where the initial weights need not be zero. (Note that if we have $w_{\text{init}} = 0$ then the learning rate does not affect the predictions $\text{sign}(w_t \cdot x_t)$.) Assume that $\|x_t\|_2 \leq X$ for some $X > 0$, and some $u \in \mathbb{R}^d$ satisfies $y_t u \cdot x_t \geq 1$ for all $t$. Modify the proof for the Perceptron Convergence Theorem by using

$$P_t = \frac{1}{2} \|u - w_t\|_2^2$$

as the potential function. The result should be that

$$\sum_{t=1}^{T} \sigma_t \leq \|u - w_{\text{init}}\|_2^2 X^2$$

for a suitable choice of $\eta$. Thus, if we start the algorithm close to the target, we get a smaller mistake bound.

Hint: This is a fairly straightforward modification of the proof in the lecture notes. Instead of $c$ and $\gamma$, the learning rate $\eta$ will appear in some terms of the potential estimate.

2. As with the all subsets kernel (Example 2.19, page 110), define for $A \subseteq \{1, \ldots, n\}$ the feature

$$\psi_A(x) = \prod_{i \in A} x_i.$$ 

The degree $q$ ANOVA feature map has the $\binom{n}{q}$ features $\psi_A$ where $|A| = q$. (Thus the all subsets feature map combines the ANOVA features for $q = 0, \ldots, n$.) Let $k_q$ be the kernel of this feature map. There is no nice closed form for this kernel, but given $x, z \in \mathbb{R}^n$ we can still compute the value

$$k_q(x, z) = \sum_{|A| = q} \psi_A(x)\psi_A(z)$$

much more efficiently than the naive $O(n^q)$. Give an algorithm to do this.

Hint: Express $k_q((x_1, \ldots, x_n), (z_1, \ldots, z_n))$ in terms of $k_{q-1}((x_1, \ldots, x_{n-1}), (z_1, \ldots, z_{n-1}))$ and $k_q((x_1, \ldots, x_{n-1}), (z_1, \ldots, z_{n-1}))$. You can save computation effort by dynamic programming.

Continues on the next page!
3. Consider online linear regression, where now \( \hat{y}_t \) and \( y_t \) can both be arbitrary real numbers. The analogue of the Perceptron algorithm is the Least Mean Squares algorithm (LMS, also known as Widrow-Hoff):

\[
\text{Initialise } w_1 = 0.
\]

Repeat for \( t = 1, \ldots, T \):

1. Get \( x_t \in \mathbb{R}^n \).
2. Predict \( \hat{y}_t = w_t \cdot x_t \).
3. Receive the correct answer \( y_t \).
4. Update \( w_{t+1} = w_t - \eta(\hat{y}_t - y_t)x_t \).

Here \( \eta > 0 \) is a learning rate parameter.

Assume that there are some \( u \in \mathbb{R}^n \) and \( X > 0 \) such that \( y_t = u \cdot x_t \) and \( \|x_t\|_2 \leq X \) for all \( t \).

Show that the square loss of the LMS algorithm can be bounded as

\[
\sum_{t=1}^{T} (y_t - \hat{y}_t)^2 \leq \|u\|_2^2 X^2.
\]

For extra credit (worth one regular problem), generalise this to the “agnostic” case where we do not assume \( u \cdot x_t = y_t \).

Hint: For the basic case, show that

\[
\frac{1}{2} \|u - w_t\|_2^2 - \frac{1}{2} \|u - w_{t+1}\|_2^2 = \left( \eta - \frac{1}{2} \eta X^2 \right) (y_t - \hat{y}_t)^2.
\]

Optimise \( \eta \) and sum over \( t \).

For the agnostic case, show that

\[
\frac{1}{2} \|u - w_t\|_2^2 - \frac{1}{2} \|u - w_{t+1}\|_2^2 = a(y_t - \hat{y}_t)^2 - b(y_t - u \cdot x_t)^2
\]

for some \( a, b > 0 \) that depend on \( X \) and \( \eta \). You do not need to find the optimal \( \eta \) for this case.

4. Consider the linear classifier \( f(x) = \text{sign}(w \cdot x) \) for \( x \in \mathbb{R}^d \) where \( w_1 = w_2 = 1 \) and \( w_i = 0 \) for \( i = 3, \ldots, d \).

We generate a random sample as follows. First, we draw a large number of instances \( x_t \) from the uniform distribution over the cube \([-1, 1]^d \). Then we classify the instances using the above classifier \( f \). Finally, we discard from the sample the points where the margin is below some value \( \gamma \) we decide in advance. Therefore we get a sample that is linearly separable with margin \( \gamma \) by the classifier \( f \).

Implement the sampling method and the Perceptron algorithm. Study how the number of mistakes made by the algorithm changes when you

- keep dimension \( d \) fixed but let the margin \( \gamma \) vary
- keep the margin \( \gamma \) fixed but let dimension \( d \) vary.

Is the behaviour of the algorithm similar to what you would expect from the Perceptron Convergence Theorem?

Your solution should consist of a brief explanation of the observations you made, a couple of representative plots to support this, and a printout of your program code.