CSE 250B: Homework 5 Solutions

Problem 1

1. Suppose $K(x, z) = \langle \phi(x), \phi(z) \rangle$ for some feature map $\phi$, and let $\phi'(x) = \sqrt{c} \phi(x)$. Then, for all $x$ and $z$,

$$K'(x, z) = cK(x, z) = c\langle \phi(x), \phi(z) \rangle = \langle \sqrt{c} \phi(x), \sqrt{c} \phi(z) \rangle$$

Therefore $K'(x, z)$ is a kernel corresponding to the feature map $\phi'$.

2. Suppose $x_0$ is the $x$ for which $K(x, x) > 0$. Consider the $1 \times 1$ kernel matrix $K' = K'(x_0, x_0)$ for the kernel $K'$ and the data point $x_0$. Then, $K' = cK(x_0, x_0)$. If $z = 1$, then $z^\top K' z = cK(x_0, x_0) < 0$, which violates the kernel Positive Semi Definiteness (PSD) property. Thus $K'$ is not a kernel.

3. Suppose $K_1(x, z) = \langle \phi^1(x), \phi^1(z) \rangle$ and $K_2(x, z) = \langle \phi^2(x), \phi^2(z) \rangle$. Then, for all $x$ and $z$,

$$K'(x, z) = c_1 \langle \phi^1(x), \phi^1(z) \rangle + c_2 \langle \phi^2(x), \phi^2(z) \rangle = \langle \sqrt{c_1} \phi^1(x), \sqrt{c_1} \phi^1(z) \rangle + \langle \sqrt{c_2} \phi^2(x), \sqrt{c_2} \phi^2(z) \rangle$$

where $\phi'(x)$ is a concatenation of the feature maps $\sqrt{c_1} \phi^1(x)$ and $\sqrt{c_2} \phi^2(x)$. In other words, if the feature maps $\phi^1$ and $\phi^2$ have $m_1$ and $m_2$ coordinates respectively, then $\phi'$ has $m_1 + m_2$ coordinates; for any $x$, the first $m_1$ coordinates of $\phi'(x)$ are $\sqrt{c_1} \phi^1_1(x), \sqrt{c_1} \phi^1_2(x), \ldots, \sqrt{c_1} \phi^1_{m_1}(x)$ and the remaining $m_2$ coordinates of $\phi'(x)$ are $\sqrt{c_2} \phi^2_1(x), \sqrt{c_2} \phi^2_2(x), \ldots, \sqrt{c_2} \phi^2_{m_2}(x)$. Therefore $K'(x, z)$ is a kernel corresponding to the feature map $\phi'$.

4. Suppose $K_1(x, z) = \langle \phi^1(x), \phi^1(z) \rangle$ and $K_2(x, z) = \langle \phi^2(x), \phi^2(z) \rangle$. If $x$ and $z$ are $d$-dimensional vectors, then, for all $x$ and $z$,

$$K'(x, z) = K_1(x, z)K_2(x, z) = \langle \phi^1(x), \phi^1(z) \rangle \cdot \langle \phi^2(x), \phi^2(z) \rangle$$

$$= \left( \sum_i \phi^1_i(x) \phi^1_i(z) \right) \cdot \left( \sum_j \phi^2_j(x) \phi^2_j(z) \right) = \sum_{i,j=1}^{d} (\phi^1_i(x) \phi^2_j(x)) \cdot (\phi^1_i(z) \phi^2_j(z)) = \langle \phi'(x), \phi'(z) \rangle$$

where

$$\phi'(x) = \begin{bmatrix}
\phi^1_1(x) & \phi^2_1(x) \\
\phi^1_2(x) & \phi^2_2(x) \\
\phi^1_3(x) & \phi^2_3(x) \\
\vdots & \vdots
\end{bmatrix}$$

(1)

That is, $\phi'$ is a $d^2 \times 1$ feature map, which has a coordinate $\phi_{(i,j)}(\cdot)$ corresponding to each pair $(i, j)$, $1 \leq i, j \leq d$, where $\phi_{(i,j)}(x) = \phi^1_i(x) \phi^2_j(x)$. Thus $K'(x, z)$ is a kernel corresponding to the feature map $\phi'$.

Problem 2

1. $K(x, z)$ is not a kernel.

   For $x = [1, -1]$, we have $K(x, x) = 1 \times -1 = -1$. The corresponding kernel matrix $K = -1$. For $v = 1$, $v^\top K v = -1 < 0$, which violates the PSD property. Thus $K$ is not a kernel.
2. \( K(x, z) \) is not a kernel.
   For \( x = [2, 2, \cdots] \), we have \( K(x, x) = 1 - \langle x, x \rangle = 1 - 4d \). The corresponding kernel matrix \( K = 1 - 4d \).
   For \( v = 1 \), \( v^T K v = 1 - 4d < 0 \), which violates the kernel PSD property for \( d > 0 \). Thus \( K \) is not a kernel.

3. \( K(x, z) \) is not a kernel.
   One way to prove that \( K \) is not a kernel is to show a counterexample to the PSD property. Pick 
   \( x = [1, 0, \ldots, 0], z = [2, 0, \ldots, 0], v = [1, -1]^\top \). Then the kernel matrix
   \[
   A = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}
   \]
   and \( v^T A v = -2 < 0 \), which violates positivity.
   A nice, second way to prove this is through contradiction. Suppose \( K \) a kernel, such that
   \( K(x, z) = \langle \phi(x), \phi(z) \rangle \). Recall the Cauchy-Schwarz Inequality for inner product, that we discussed in
   Lecture 2:
   \[
   \langle \phi(x), \phi(z) \rangle^2 \leq \langle \phi(x), \phi(x) \rangle \cdot \langle \phi(z), \phi(z) \rangle
   \]
   From this inequality,
   \[
   K(x, z)^2 \leq K(x, x) \cdot K(z, z)
   \]
   Suppose \( x \) is any vector with norm 1 and let \( z = 2x \). By the definition of \( K \), we have
   \( K(x, x) = \|x - x\|^2 = 0 \) and \( K(x, x) \cdot K(z, z) = 0 \). However \( K(x, z) = \|x - 2x\|^2 = 1 > K(x, x) \cdot K(z, z) \),
   which leads to a contradiction! Thus \( K \) is not a kernel.

4. \( K(x, z) \) is a kernel corresponding to the feature map \( \phi(x) = f(x_1, x_2) \).

5. \( K(x, z) \) is a kernel.
   Recall that \( a^2 - b^2 = (a - b) \cdot (a + b) \)
   Hence, we have
   \[
   \frac{1 - \langle x, z \rangle^2}{1 - \langle x, z \rangle} = 1 + \langle x, z \rangle
   \]
   (4)
   In the above equation, we can rewrite 1 as \( \langle x, z \rangle^0 \)
   Thus, we can now write, \( K(x, z) = K_0(x, z) + K_1(x, z) \). In Problem 1, we saw that the sum or product
   of two kernels is also a kernel. We know that \( K_0(x, z) \) and \( K_1(x, z) \) are both kernels.
   The feature map \( \phi_0(x) \) corresponding \( K_0(x, z) \) is
   \[
   \phi_0(x) = 1
   \]
   (5)
   \( K_1(x, z) \) corresponds to the feature map
   \[
   \phi_1(x) = x
   \]
   (6)
   Using Problem 1 Part 3, \( K(x, z) = K_0(x, z) + K_1(x, z) \) is a kernel corresponding to the feature map \( \phi' \),
   where for any \( x \), \( \phi'(x) \) is a concatenation of the feature maps \( \phi_0(x) \) and \( \phi_1(x) \).

6. \( K(x, z) \) is a kernel.
   Let \( K_i(x, z) = \min(x_i, z_i) \). From Problem 1, we know that the sum of two kernels \( K_1 \) and \( K_2 \) is also a
   kernel whose corresponding feature map is the concatenation of the feature maps corresponding to \( K_1 \)
   and \( K_2 \). Thus if we can find the feature maps for all \( K_i(x, z) \), then we can get the feature map for
   \( K(x, z) \) by concatenating these maps. Consider following feature map:
   \[
   \phi_i(x) = [f_1(x_1), f_2(x_1), \ldots, f_{100}(x_i)]^\top
   \]
   (7)
where \( f_k(t) = I(t \geq k) = \begin{cases} 1 & t \geq k \\ 0 & t < k \end{cases} \). Without loss of generality, suppose that \( x_i \leq z_i \). Then \( \phi_i(x) = [1, \ldots, 1, 0, \ldots, 0]^\top \) where only the first \( x_i \) entries are 1. Analogously, \( \phi_i(z) = [1, \ldots, 1, 0, \ldots, 0]^\top \) where only the first \( z_i \) entries are 1. Then

\[
\langle \phi_i(x), \phi_i(z) \rangle = \sum_{i=1}^{x_i} 1 \cdot 1 + \sum_{i=x_i+1}^{z_i} 0 \cdot 1 + \sum_{i=z_i+1}^{100} 0 \cdot 0 = x_i = \min(x_i, z_i)
\]

Therefore \( K_i(x, z) \) is a kernel corresponding to the feature map \( \phi_i(x) = [f_1(x_i), f_2(x_i), \ldots, f_{100}(x_i)]^\top \), and \( K(x, z) \) is a kernel corresponding to the feature map \( \phi(x) \) which is a concatenation of the feature maps \( \phi_1(x), \phi_2(x), \ldots, \phi_d(x) \).

7. \( K(x, z) \) is a kernel.

Let \( K_i(x, z) = 1 + x_i z_i \), then \( K(x, z) = \prod_{i=0}^{d} K_i(x) \). From Problem 1, we know that the product of two kernels is also a kernel. Since \( K_i(x, z) \) is a kernel corresponding to the feature map \( \phi_i(x) = [1, x_i]^\top \), \( K(x, z) \) is also a kernel. More specifically, \( K(x, z) \) is a kernel corresponding to the feature map \( \phi(x) \), where for any \( x \), \( \phi(x) \) has \( 2^d \) coordinates, one corresponding to each subset \( S \) of \( \{1, 2, \ldots, d\} \). \( \phi_S(x) \), the coordinate of \( \phi(x) \) corresponding to the set \( S \) is \( \prod_{i \in S} x_i \). This kernel is called the All Subsets kernel.

8. \( K(x, z) \) is not a kernel.

One way to prove this is by showing a violation of the PSD property. Let \( x = [0, \ldots, 0], z = [1, 0, \ldots, 0] \) and \( v = [1, -1]^\top \). Then the kernel matrix

\[
K = \begin{bmatrix} K(x, x) & K(x, z) \\ K(z, x) & K(z, z) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 1 & 1 \end{bmatrix}
\]

Thus, \( v^\top A v = -1 < 0 \), which violates positivity.

Another nice way is through a violation of the Cauchy-Schwartz inequality. Consider \( x = [0, \ldots, 0] \) and \( z = [1, 0, \ldots, 0] \). Then \( K(x, x) = 0 \), \( K(x, z) = K(z, z) = 1 \), which violates Cauchy-Schwarz inequality – that is \( K(x, z)^2 \geq K(x, x) \cdot K(z, z) \).

Problem 3

1. First, we can compute the marginal distributions of \( Y \) and \( Z \) as follows,

<table>
<thead>
<tr>
<th>( y )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(Y = y) )</td>
<td>( \frac{2}{5} )</td>
<td>( \frac{3}{5} )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( z )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(Z = z) )</td>
<td>( \frac{9}{20} )</td>
<td>( \frac{11}{20} )</td>
</tr>
</tbody>
</table>

Then, by definition of conditional probability, i.e. \( P(X = x | Y = y) = \frac{P(X = x, Y = y)}{P(Y = y)} \), we can get the conditional distributions of \( X | Y \) as follows.

<table>
<thead>
<tr>
<th>( x )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(X = x</td>
<td>Y = 0) )</td>
<td>( \frac{1}{7} )</td>
</tr>
<tr>
<td>( P(X = x</td>
<td>Y = 1) )</td>
<td>( \frac{1}{5} )</td>
</tr>
</tbody>
</table>

Similarly we have the conditional distributions of \( X | Z \) as follows,

<table>
<thead>
<tr>
<th>( x )</th>
<th>0</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( P(X = x</td>
<td>Z = 0) )</td>
<td>( \frac{1}{5} )</td>
</tr>
<tr>
<td>( P(X = x</td>
<td>Z = 1) )</td>
<td>( \frac{3}{11} )</td>
</tr>
</tbody>
</table>
2. By the definition of conditional entropy, \( H(X|Y) = P(Y = 0)H(X|Y = 0) + P(Y = 1)H(X|Y = 1) \).

\[
H(X|Y = 0) = -P(X = 0|Y = 0) \log P(X = 0|Y = 0) - P(X = 1|Y = 0) \log P(X = 1|Y = 0)
= -\frac{1}{2} \log \frac{1}{2} - \frac{1}{2} \log \frac{1}{2}
= \log 2
\]

Similarly we have

\[
H(X|Y = 1) = -P(X = 0|Y = 1) \log P(X = 0|Y = 1) - P(X = 1|Y = 1) \log P(X = 1|Y = 1)
= -\frac{1}{6} \log \frac{1}{6} - \frac{5}{6} \log \frac{5}{6}
= \log 6 - \frac{5}{6} \log 5
\]

Thus

\[
H(X|Y) = P(Y = 0)H(X|Y = 0) + P(Y = 1)H(X|Y = 1)
= \frac{2}{5} \log 2 + \frac{3}{5} \left( \log 6 - \frac{5}{6} \log 5 \right)
= \frac{2}{5} \log 2 + \frac{3}{5} \log 6 - \frac{1}{2} \log 5
\]

For \( H(X|Z) \), we can get

\[
H(X|Z = 0) = -P(X = 0|Z = 0) \log P(X = 0|Z = 0) - P(X = 1|Z = 0) \log P(X = 1|Z = 0)
= -\frac{1}{3} \log \frac{1}{3} - \frac{2}{3} \log \frac{2}{3}
= \log 3 - \frac{2}{3} \log 2
\]

Similarly we have

\[
H(X|Z = 1) = -P(X = 0|Z = 1) \log P(X = 0|Z = 1) - P(X = 1|Z = 1) \log P(X = 1|Z = 1)
= -\frac{3}{11} \log \frac{3}{11} - \frac{8}{11} \log \frac{8}{11}
= \log 11 - \frac{3}{11} \log 3 - \frac{8}{11} \log 8
\]

Thus

\[
H(X|Z) = P(Z = 0)H(X|Z = 0) + P(Z = 1)H(X|Z = 1)
= \frac{9}{20} \left( \log 3 - \frac{2}{3} \log 2 \right) + \frac{11}{20} \left( \log 11 - \frac{3}{11} \log 3 - \frac{8}{11} \log 8 \right)
= -\frac{3}{2} \log 2 + \frac{3}{10} \log 3 + \frac{11}{20} \log 11
\]

Using natural logarithm, the numerical values are shown as follows.

| \( H(X|Y) = 0 \) | 0.693147180560 |
| \( H(X|Y) = 1 \) | 0.450561208866 |
| \( H(X|Y) \) | 0.547595597544 |
| \( H(X|Z = 0) \) | 0.63651416829 |
| \( H(X|Z = 1) \) | 0.5859526183 |
| \( H(X|Z) \) | 0.6087053158 |

3. From the table above, \( H(X|Y) < H(X|Z) \). This suggests that there is less uncertainty in \( X \) when given \( Y \) than when given \( Z \). Therefore gene A is more informative about the cancer.
Problem 4

1. False.
   Counterexample: Consider a classifier for data which uses one feature (called Feature1).

   Figure 1: Two Decision Trees which are equal (see definition in question) but have different structures

   Feature1 < 0.75
   Yes
   Predict 0
   No
   Predict 1

   Feature1 < 0.5
   Yes
   Predict 0
   No
   Predict 1

2. False.
   If \( T \) and \( T' \) produce zero error on the same training set \( S \subseteq \mathcal{X} \), then, \( \forall x \in S, T(x) = T'(x) \). However, the training set typically does not include all elements in feature space \( \mathcal{X} \). Thus, there exist such \( x_0 \in \mathcal{X} - S \) that \( T(x_0) \neq T'(x_0) \). For example, consider the following training set:

<table>
<thead>
<tr>
<th>Feature 1</th>
<th>Feature 2</th>
<th>Label</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

   For training set above, the two decision trees shown in Figure 2 both produce zero error. However, for the point \( x_1 = (0, 1) \) or the point \( x_2 = (1, 0) \), these two trees would give different predictions. Hence they are not equal.

   Figure 2: Two Decision Trees with Zero Error on \( S \)

Problem 5

For this question, we assume that ties are always broken in a consistent manner for both the \( k \)-NN and ID3 decision tree algorithms.

\( k \)-NN. We will obtain equal \( k \)-NN classifiers before and after a space transform for an arbitrary data set, if and only if, the following relative distance condition holds: for any three points \( x, x_p \) and \( x_q \) in the original space, \( d(x, x_p) \geq d(x, x_q) \) implies \( d(z, z_p) \geq d(z, z_q) \), where \( z, z_p \) and \( z_q \) are the points after rescaling. In other words, we need to ensure that in all cases, the nearest neighbors of a point in the original space are still the nearest neighbors in the rescaled space.

In the case of a uniform scaling factor (all \( \alpha^j = \alpha \)), the distance between any two points \( z_1 \) and \( z_2 \) in the rescaled space is,

\[
d(z_1, z_2) = \sqrt{\sum_j (z_1^j - z_2^j)^2} = \sqrt{\sum_j (\alpha x_1^j - \alpha x_2^j)^2} = \alpha \sqrt{\sum_j (x_1^j - x_2^j)^2} = \alpha d(x_1, x_2),
\]
This is simply the distance in the original space scaled by a constant $\alpha$. Clearly the relative distance condition holds. In particular, this means that the training points that are the nearest neighbors of $x$ in the original space remain the nearest neighbors of $z$ in the rescaled space, therefore prediction for $x$ remains the same as the prediction for $z$.

For nonuniform scaling factors, the relative distance condition does not necessarily hold. One extreme example is if $\alpha^1 = 1, \alpha^2 = 0.0001$ (a very small quantity). The transform now essentially projects each point to the $x$-axis (although the point will be not be exactly on the axis). Consider the training points (1,0) with label 0 and (0,1) with label 1, and a test example (0.1, 0). In the original space, (0.1, 0) is closer to (1,0) than (0,1) and will be assigned label 0; in the rescaled space however, it will be rescaled to be (0.1, 0), will be closer to (0,1) (now rescaled as (0, 0.001)), and thus will be assigned label 1 by the 1-NN classifier. Therefore in this case, we are not guaranteed to get the same $k$-NN classifier.

**ID3 Decision Tree.** The decision trees produced by the ID3 algorithm will be equal in both cases, assuming that ties are broken in a consistent manner. We can show this by induction. In what follows, we will say that a splitting rule $(j, t)$ in the original space is equal to a splitting rule $(j, \alpha^t t)$ in the rescaled space.

We run the ID3 algorithm on $S$ and $S'$ simultaneously, and maintain the following invariants at each step of the algorithm. If $T$ and $T'$ are the trees built based on $S$ and $S'$ respectively, then, (a) $T$ and $T'$ have the same structure, (b) for each internal node $v$ in $T$, the splitting rule at $v$ is equal to the splitting rule at the corresponding internal node $v'$ in $T'$ and (c) if $D$ is the dataset associated with a leaf node in $T$, then the dataset associated with the corresponding leaf node in $T'$ is the rescaled version of points in $D$.

The invariant holds at the beginning of the algorithm, as the only (leaf) node is the root, which is associated with $S$ in $T$ and $S'$ in $T'$. Suppose the invariant holds at step $t$ of the algorithm, and at step $t + 1$ we split a node $v$ in $T$ such that the dataset associated with $v$ is $D$. If the splitting rule used is $(j, t)$, then, this splitting rule has the highest information gain among all the possible splitting rules. Observe that as the corresponding node $v'$ in $T'$ is associated with a scaled version $D'$ of $D$, for any $j$ and $t$, the information gain of a splitting rule $(j, \alpha^t t)$ at $v'$ is equal to the information gain of the splitting rule $(j, t)$ in $v$. Thus, assuming that ties are broken consistently, we will pick the splitting rule $(j, \alpha^t t)$ to split node $v'$. Thus invariants (a) and (b) are maintained after step $t + 1$. Finally, invariant (c) is also maintained as the subset of $D$ for which feature $j$ is $\leq t$ is exactly equal to the subset of $D'$ for which feature $j$ is $\leq \alpha^t t$.

Thus, at the end of the ID3 decision tree algorithm, we arrive at two trees $T$ and $T'$ which have exactly the same structure, where the corresponding nodes $v$ and $v'$ have equal splitting rules. Thus if a test example $x$ follows a path $P$ in $T$ from the root to the leaf, its rescaled version $z$ will follow exactly the same path in $T'$ from root to leaf and will be classified the same way. Therefore the two decision trees will be equal.

**Perceptron.** For a uniform scaling factor $\alpha$, we claim that at any step, if the hyperplane normal in the original space is $w$, then the hyperplane normal in the rescaled space must be $\alpha w$. If this claim is true, then the classifiers in the two spaces will be equal, because as $\alpha > 0$, $\text{sign}(\langle w, x_t \rangle) = \text{sign}(\langle \alpha w, z_t \rangle) = \text{sign}(\langle \alpha w, \alpha x_t \rangle)$.

We prove this by induction. The base case is trivial because $w$ is initialized to 0 in both spaces. Then suppose our claim is true for step $t - 1$, we show that the claim still holds at step $t$. At step $t$ the algorithm predicts the label for the training data $(x_t, y_t)$ in the original space and training data $(z_t = \alpha x_t, y_t)$ in the rescaled space. It is easy to see that the prediction result is the same for the classifiers in both spaces as $\alpha > 0$. If the result is correct, then no change is made to either $w$. If the result is wrong, the normal in the original space is updated to $w + y_t x_t$, while in the rescaled space, the normal is updated to $\alpha w + y_t z_t = \alpha (w + y_t x_t)$. Thus the claim still holds at this step. Therefore the Perceptron algorithm produces equal classifiers in both spaces.
For non-uniform $\alpha$’s, the two classifiers are not equal. One counter-example is given below. There is only one positive training data $(2, -2)$, which becomes $(1, -2)$ in the rescaled space. Consider the test data $(2, 1)$, and the rescaled version $(1, 1)$. The resulting classifier classifies them into different labels.

**Behavior under scaling transformations.** In case of uniform scaling transformations (same $\alpha^i$) across all features/dimensions, all the 3 algorithms are equally robust. However, in case of non-uniform scaling transformations (different $\alpha^i$), ID3 Decision Trees are more robust compared to $k$-NN and Perceptrons.