CAP 5610
Assignment #3 Solution

March 17, 2025

Arman Sayan

---

1 Kernel Computation Cost [30 points]

1. [10 points] Consider we have a two-dimensional input space such that the input vector is $x = (x_1, x_2)^T$. Define the feature mapping $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$. What is the corresponding kernel function, i.e., $K(x, z)$? Do not leave $\phi(x)$ in your final answer.

Ans:
To find the kernel function $K(x, z)$, we use the definition of the kernel function as follows:

$$K(x, z) = \phi(x)^T \phi(z)$$

where $\phi(x)$ represents feature mapping.
Given that $\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T$, for another input vector $z = (z_1, z_2)^T$, the feature mapping is $\phi(z) = (z_1^2, \sqrt{2}z_1z_2, z_2^2)^T$.
Now, we can compute the dot product of the feature mappings as follows:

$$\begin{aligned} K(x, z) &= \phi(x)^T \phi(z) \\ &= \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} z_1^2 \\ \sqrt{2}z_1z_2 \\ z_2^2 \end{bmatrix} \\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \end{aligned}$$

With a deduction from the above calculations, we can observe the final expression as

$$K(x, z) = (x_1z_1 + x_2z_2)^2$$

Since $x_1z_1 + x_2z_2$ is the dot product of the input vectors x and z, namely x^Tz, the corresponding kernel function is

$$\boxed{\boldsymbol{K(x, z) = (x^Tz)^2}}$$

2. [20 points] Suppose we want to compute the value of the kernel function $K(x, z)$ from the previous question on two vectors $x, z \in \mathbb{R}^2$. How many additions and multiplications are needed if you
   i. [10 points] Map the input vector to the feature space and then perform the dot product on the mapped features?

Ans:
Using the feature mappings
$$\phi(x) = (x_1^2, \sqrt{2}x_1x_2, x_2^2)^T \text{ and } \phi(z) = (z_1^2, \sqrt{2}z_1z_2, z_2^2)^T,$$
the kernel function $K(x, z)$ can be computed as

$$\begin{aligned} K(x,z) &= \phi(x)^T\phi(z) \\ &= \begin{bmatrix} x_1^2 & \sqrt{2}x_1x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} z_1^2 \\ \sqrt{2}z_1z_2 \\ z_2^2 \end{bmatrix} \\ &= x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2 \end{aligned}$$

For the feature mapping of the vector x, we need to compute terms x_1^2, x_2^2, and $\sqrt{2}x_1x_2$. To calculate these terms, we need 3 multiplications.
Since we have 2 input vectors x and z, we need to perform 3 multiplications for each vector, which results in 6 multiplications for mapping the vectors to the feature space.
After mapping the vectors to the feature space, we need to compute the dot product of the feature mappings as
$$K(x, z) = x_1^2z_1^2 + 2x_1x_2z_1z_2 + x_2^2z_2^2$$
This requires 3 multiplications for each term in the summation and 2 additions to sum three terms.
In total, we need **9 multiplications and 2 additions** to compute the value of $K(x, z)$.

ii. [10 points] Compute through the kernel function you derived in question 1?

Ans:
The kernel function derived in question 1 is $K(x, z) = (x^T z)^2$.
To compute the value of the kernel function $K(x, z)$ using this kernel function, we need to calculate the dot product of the input vectors x and z, namely $x^T z$.
The dot product of two vectors x and z is calculated as

$$x^T z = x_1 z_1 + x_2 z_2$$

This requires 2 multiplications for terms $x_1 z_1$ and $x_2 z_2$ and 1 addition to sum those terms.
After computing the dot product, we need to square the result to obtain the value of the kernel function $K(x, z)$.
Squaring the result requires 1 multiplication.
In total, we need **3 multiplications and 1 addition** to compute the value of $K(x, z)$.

2 Activation Functions and Loss Functions [30 points]

1. [20 points] For this assignment, you are encouraged to consult Dr. GOOGLE and Dr. ChatGPT. But please explain things in your own language while you write the answer. For each of the following activation functions, briefly describe the type of non-linearity (if any) it introduces and discuss their pros and cons.
   (a) Linear Activation Function
   (b) Sigmoid Activation Function
   (c) Tanh Activation Function
   (d) ReLU (Rectified Linear Unit) Activation Function

Ans:

(a) **Linear Activation Function:**

$$f(x) = x$$

Linear activation function introduces a linear relationship between the input and output where the output is directly proportional to the input. It is a simple activation function that does not introduce any non-linearity to the model.

It is simple and computationally efficient, making it useful for regression problems. However, it does not introduce any non-linearity, meaning it cannot capture complex relationships in data, and stacking multiple linear layers does not increase the model's learning capacity.

(b) **Sigmoid Activation Function:**

$$f(x) = \frac{1}{1 + e^{-x}}$$

Sigmoid activation function introduces a non-linearity that squashes the input values between 0 and 1 in a S-shaped curve.

It is useful for binary classification problems where the output is required to be in the range of 0 and 1. Furthermore, it provides a smooth gradient that helps in training the model. However, it suffers from the vanishing gradient problem, which makes training deep neural networks difficult. Also, it is not zero-centered, namely the output is always positive, which can lead to convergence issues.

(c) **Tanh Activation Function:**

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

Tanh activation function introduces a non-linearity that squashes the input values in a S-shaped curve, similar to sigmoid activation function, but output values between -1 and 1.

It is zero-centered, which helps in faster convergence during training. Also, it provides stronger gradients for hidden layers compared to sigmoid activation function. However, just like sigmoid activation function, it also suffers from the vanishing gradient problem, making it difficult to train deep neural networks. Furthermore, it is not suitable for sparse data. Lastly, it is computationally expensive due to the exponential operations.

(d) **ReLU (Rectified Linear Unit) Activation Function:**

$$f(x) = \max(0, x)$$

ReLU activation function introduces a non-linearity that outputs the input as it is if it is positive, and zero otherwise, making computations simple.

It is computationally efficient and helps in faster convergence during training compared to sigmoid and tanh. Also, it does not suffer from the vanishing gradient problem, making it suitable for training deep neural networks. Lastly, it remains one of the most widely used activation functions due to its strong performance in deep networks. However, it suffers from the dying ReLU problem where neurons can die during training and stop learning when inputs become negative and ReLU outputs 0. Also, it is not zero-centered, which can lead to convergence issues. Lastly, it is not suitable for small input values as it ignores negative information.

2. [10 points] For each of the following loss functions, briefly describe their mathematical formula and discuss for which type of learning task (classification, regression, etc.) they are most appropriate.
   (a) Mean Squared Error Loss
   (b) Binary Cross Entropy Loss
   (c) Hinge Loss
   (d) Softmax Cross Entropy Loss

Ans:

(a) **Mean Squared Error Loss:**

Mean Squared Error (MSE) loss is a regression loss function that measures the average squared difference between the predicted values $\hat{y}_i$ and actual values y_i.

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

where N is the number of samples in the dataset, y_i is the actual target value, and $\hat{y}_i$ is the predicted value.

The squaring ensures that large errors are penalized more heavily than small errors.

It is most appropriate for regression tasks where the model is required to predict continuous values.

(b) **Binary Cross Entropy Loss:**

Binary Cross Entropy loss is a classification loss function that measures the difference between the predicted probability $\hat{y}_i$ and actual class labels for binary classification tasks.

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

where N is the number of samples in the dataset, y_i is the actual class label such as 0 or 1, and $\hat{y}_i$ is the predicted probability between 0 and 1.

The logarithm makes wrong confident predictions highly penalized.

It is most appropriate for binary classification tasks where the model is required to predict probabilities for two classes.

(c) **Hinge Loss:**

Hinge loss is a classification loss function that measures the margin of the predicted class label $\hat{y}_i$ from the decision boundary for binary classification tasks.

$$\text{Hinge Loss} = \frac{1}{N}\sum_{i=1}^{N}\max(0, 1 - y_i\hat{y}_i)$$

where N is the number of samples in the dataset, y_i is the actual class label such as -1 or 1, and $\hat{y}_i$ is the predicted class label.

If the true label y_i and predicted label $\hat{y}_i$ have the same sign and a large margin greater than 1, the loss is 0. If that is not the case, it penalizes the prediction proportionally, encouraging a large decision margin between different classes.

It is most appropriate for binary classification tasks where the model is required to predict class labels, like in SVMs.

(d) **Softmax Cross Entropy Loss:**

Softmax Cross Entropy loss is a classification loss function that measures the difference between the predicted and actual class labels for multi-class classification tasks.

$$\text{Softmax Cross Entropy} = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C} y_{i,c}\log\left(\frac{e^{\hat{y}_{i,c}}}{\sum_{k=1}^{C} e^{\hat{y}_{i,k}}}\right)$$

where N is the number of samples in the dataset, C is the number of classes, $y_{i,c}$ is 1 if the true class is c and 0 otherwise, and $\hat{y}_{i,c}$ is the raw predicted probability for class c before applying softmax.

The softmax function converts raw model outputs into probabilities, and encourages high confidence in the correct class while penalizing incorrect ones. The cross-entropy term compares the predicted probability of the correct class to true label.

It is most appropriate for multi-class classification tasks where each sample belongs to one of many classes and the model is required to predict probabilities for multiple classes.

3 Linear SVM Implementation [40 points]

Solution for Q3:

Ans:

Please check the source code and outputs included in the appendix named as

CAP_5610_Assignment_3_Solution_Arman_Sayan.ipynb

for the solution.