10. Kernel Methods

Project due Mar 15, 2023 08:59 -03 Past due

As you can see, implementing a direct mapping to the high-dimensional features is a lo an even higher dimensional feature mapping.) This is where the kernel trick becomes u

Recall the kernel perceptron algorithm we learned in the lecture. The weights θ can be combination of features:

$$\theta = \sum_{i=1}^{n} \alpha^{(i)} y^{(i)} \phi\left(x^{(i)}\right)$$

In the softmax regression fomulation, we can also apply this representation of the weig

$$\theta_j = \sum_{i=1}^{n} \alpha_j^{(i)} \phi\left(x^{(i)}\right).$$

$$h\left(x\right) = \frac{1}{\sum_{j=1}^{k} e^{[\theta_j \cdot \phi(x)/\tau] - c}} \begin{bmatrix} e^{[\theta_1 \cdot \phi(x)/\tau] - c} \\ e^{[\theta_2 \cdot \phi(x)/\tau] - c} \\ \vdots \\ e^{[\theta_k \cdot \phi(x)/\tau] - c} \end{bmatrix}$$

$$h\left(x\right) = \frac{1}{\sum_{j=1}^{k} e^{[\sum_{i=1}^{n} \alpha_j^{(i)} y^{(i)} \phi(x^{(i)}) \cdot \phi(x)/\tau] - c}} \begin{bmatrix} e^{[\sum_{i=1}^{n} \alpha_1^{(i)} y^{(i)} \phi(x^{(i)}) \cdot \phi(x)/\tau] - c} \\ e^{[\sum_{i=1}^{n} \alpha_2^{(i)} y^{(i)} \phi(x^{(i)}) \cdot \phi(x)/\tau] - c} \\ \vdots \\ e^{[\sum_{i=1}^{n} \alpha_k^{(i)} y^{(i)} \phi(x^{(i)}) \cdot \phi(x)/\tau] - c} \end{bmatrix}$$

We actually do not need the real mapping $\phi\left(x\right)$, but the inner product between two fe $\phi\left(x_i\right) \cdot \phi\left(x\right)$, where x_i is a point in the training set and x is the new data point for wh the probability. If we can create a kernel function $K\left(x, y\right) = \phi\left(x\right) \cdot \phi\left(y\right)$, for any tw then kernelize our softmax regression algorithm.

You will be working in the files `part1/main.py` and `part1/kernel.py` in this proble

```
        Compute the polynomial kernel between two matrices X and Y::
            K(x, y) = (<x, y> + c)^p
        for each pair of rows x in X and y in Y.

        Args:
            X - (n, d) NumPy array (n datapoints each with d features)
            Y - (m, d) NumPy array (m datapoints each with d features)
            c - a coefficient to trade off high-order and low-order terms
            p - the degree of the polynomial kernel

        Returns:
            kernel_matrix - (n, m) Numpy array containing the kernel matri
```

Press ESC then TAB or click outside of the code editor to exit

Unanswered

Submit You have used 0 of 25 attempts

Gaussian RBF Kernel

1 point possible (graded)

Another commonly used kernel is the Gaussian RBF kenel. Similarly, write a function r
two matrices and and computes the RBF kernel for every pair of rows

Available Functions: You have access to the NumPy python library as np

```

def rbf_kernel(X, Y, gamma):
    """
        Compute the Gaussian RBF kernel between two matrices X and Y::
            K(x, y) = exp(-gamma ||x-y||^2)
        for each pair of rows x in X and y in Y.

        Args:
            X - (n, d) NumPy array (n datapoints each with d features)
            Y - (m, d) NumPy array (m datapoints each with d features)
            gamma - the gamma parameter of gaussian function (scalar)

        Returns:
            kernel_matrix - (n, m) Numpy array containing the kernel matri
    """
```

Press ESC then TAB or click outside of the code editor to exit

Unanswered

Submit You have used 0 of 25 attempts

kernelized features.

In the next project, you will apply neural networks to this task.

Previous Next

Discussion