

TASK 9: Quantum SVM

Aim:

To compute a quantum kernel matrix for a dataset using a quantum feature map.

Algorithm:

- Generate synthetic 2D data for two classes.
- Map classical features to quantum feature space.
- Compute pairwise kernel values using dot product.
- Visualize dataset with class labels.

Program:

```
print("\n" + "="*50)
print("TASK 9: QUANTUM SVM ")
print("="*50)

def quantum_kernel(x1, x2):
    """Simplified quantum kernel function"""
    # Quantum kernel based on feature map overlap
    phi_x1 = np.array([np.cos(x1[0]), np.sin(x1[0]),
np.cos(x1[1]), np.sin(x1[1])])
    phi_x2 = np.array([np.cos(x2[0]), np.sin(x2[0]),
np.cos(x2[1]), np.sin(x2[1])])
    return np.abs(np.dot(phi_x1, phi_x2))**2

# Generate simple 2D dataset (simplified Iris)
np.random.seed(42)
class_0 = np.random.normal([0, 0], 0.5, (10, 2))
class_1 = np.random.normal([2, 2], 0.5, (10, 2))
X = np.vstack([class_0, class_1])
y = np.array([0]*10 + [1]*10)
```

```
print("Generated simplified 2D dataset:")
print(f"Class 0 samples: {len(class_0)}")
print(f"Class 1 samples: {len(class_1)}")

# Compute quantum kernel matrix
n_samples = len(X)
K = np.zeros((n_samples, n_samples))
for i in range(n_samples):
    for j in range(n_samples):
        K[i, j] = quantum_kernel(X[i], X[j])
print(f"\nQuantum kernel matrix computed: {K.shape}")
print("QSVM training would use this kernel matrix")

# Visualize dataset
plt.figure(figsize=(8, 6))
plt.scatter(class_0[:, 0], class_0[:, 1], c='blue', label='Class 0',
            alpha=0.7)
plt.scatter(class_1[:, 0], class_1[:, 1], c='red', label='Class 1',
            alpha=0.7)
plt.xlabel('Feature 1')
plt.ylabel('Feature 2')
plt.title('Quantum SVM Dataset')
plt.legend()
plt.show()
```

Result:

Quantum kernel matrix for the dataset was computed and visualized successfully.

