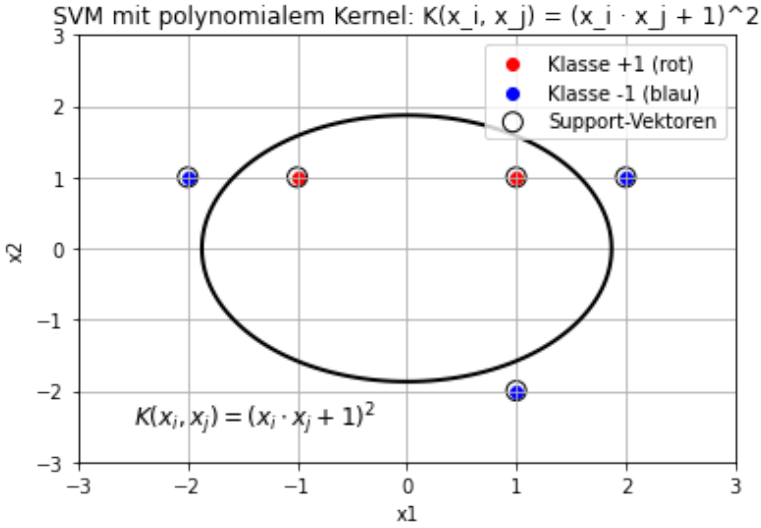
In [1]:

```
# Import der benötigten Bibliotheken
import numpy as np
import matplotlib.pyplot as plt
from sklearn import svm

# Definieren der Datenpunkte
# Klasse +1: rot (A und B)
X_pos = np.array([[1, 1], [-1, 1]])  # A(1, 1) und B(-1, 1)
y_pos = np.ones(X_pos.shape[0])

# Klasse -1: blau (C, D und E)
X_neg = np.array([[2, 1], [1, -2], [-2, 1]])  # C(2, 1), D(1, -2) und E(-2,
1)
y_neg = -1 * np.ones(X_neg.shape[0])

# Zusammenfügen der Datenpunkte und Klassenlabels
X = np.vstack((X_pos, X_neg))
y = np.hstack((y_pos, y_neg))

# SVM-Modell mit polynomialem Kernel (Grad 2) erstellen
model = svm.SVC(kernel='poly', degree=2, C=1e5)  # C ist der Regularisierungs
parameter
model.fit(X, y)

# Erstellen eines Mesh-Grids für die Entscheidungsebene
xx, yy = np.meshgrid(np.linspace(-3, 3, 500), np.linspace(-3, 3, 500))
Z = model.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Visualisieren der Datenpunkte, Trennlinie und Support-Vektoren
plt.scatter(X_pos[:, 0], X_pos[:, 1], color='red', label='Klasse +1 (rot)')
plt.scatter(X_neg[:, 0], X_neg[:, 1], color='blue', label='Klasse -1 (blau)')

# Trennlinie plotten (Entscheidungsfunktion == 0)
plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='black')

# Support-Vektoren plotten
plt.scatter(model.support_vectors_[:, 0], model.support_vectors_[:, 1], s=10
0,
            facecolors='none', edgecolors='black', label='Support-Vektoren')

# Plot-Details
plt.title("SVM mit polynomialem Kernel: K(x_i, x_j) = (x_i · x_j + 1)^2")
plt.xlabel("x1")
plt.ylabel("x2")
plt.legend()
plt.grid(True)
plt.xlim(-3, 3)
plt.ylim(-3, 3)
plt.text(-2.5, -2.5, r'$K(x_i, x_j) = (x_i \cdot x_j + 1)^2$', fontsize=12, c
olor='black')

plt.show()
```


SVM mit polynomialem Kernel: K(x_i, x_j) = (x_i · x_j + 1)^2
Klasse +1 (rot)
Klasse -1 (blau)
Support-Vektoren
$K(x_i, x_j) = (x_i \cdot x_j + 1)^2$
x1
x2
3
2
1
0
−1
−2
−3
−3
−2
−1
0
1
2
3