Vergleich Polynomischer Kernel mit RBF Kernel

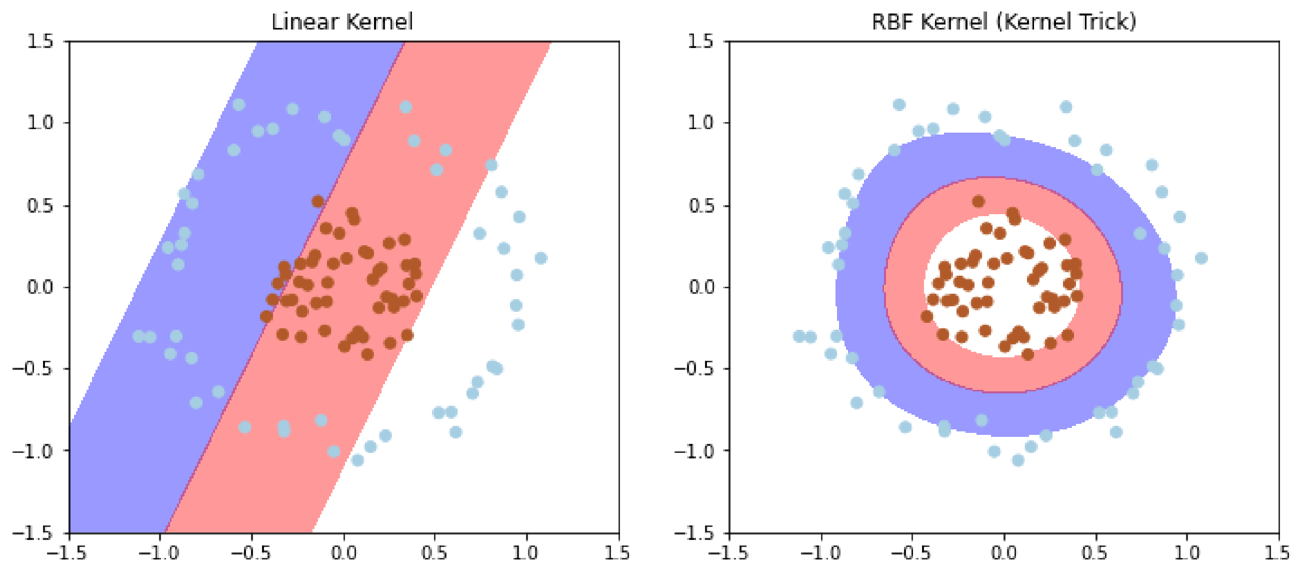
Author/Datum: Hermann Völlinger, DHBW Stuttgart / 13.10.2023

1. SVM with RBF kernel

In this example, we create a dataset of points that form concentric circles, which is not linearly separable in the original feature space. We then use two SVM models: one with a linear kernel and another with an RBF (Radial Basis Function) kernel. The RBF kernel effectively applies the kernel trick by transforming the data into a higher-dimensional space where it becomes linearly separable. The decision boundaries and data points are plotted to visualize the difference between the two kernels.

The RBF kernel is commonly used in practice to handle non-linearly separable data, and it is one of the key applications of the kernel trick in SVM.

In [1]:
```
# Import necessary libraries
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets
from sklearn.svm import SVC

# Create a simple dataset that is not linearly separable
X, y = datasets.make_circles(n_samples=100, factor=0.3, noise=0.1)

# Create an SVM model with a linear kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X, y)

# Create an SVM model with an RBF (Radial Basis Function) kernel (kernel trick)
svm_rbf = SVC(kernel='rbf', gamma='scale')
svm_rbf.fit(X, y)

# Create a meshgrid to visualize the decision boundaries
xx, yy = np.meshgrid(np.linspace(-1.5, 1.5, 200), np.linspace(-1.5, 1.5, 200))
Z_linear = svm_linear.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z_linear = Z_linear.reshape(xx.shape)
Z_rbf = svm_rbf.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z_rbf = Z_rbf.reshape(xx.shape)

# Plot the data points and decision boundaries
plt.figure(figsize=(12, 5))
plt.subplot(121)
plt.contourf(xx, yy, Z_linear, levels=[-1, 0, 1], colors=['blue', 'red'], alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.title('Linear Kernel')

plt.subplot(122)
plt.contourf(xx, yy, Z_rbf, levels=[-1, 0, 1], colors=['blue', 'red'], alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.title('RBF Kernel (Kernel Trick)')

plt.show()
```

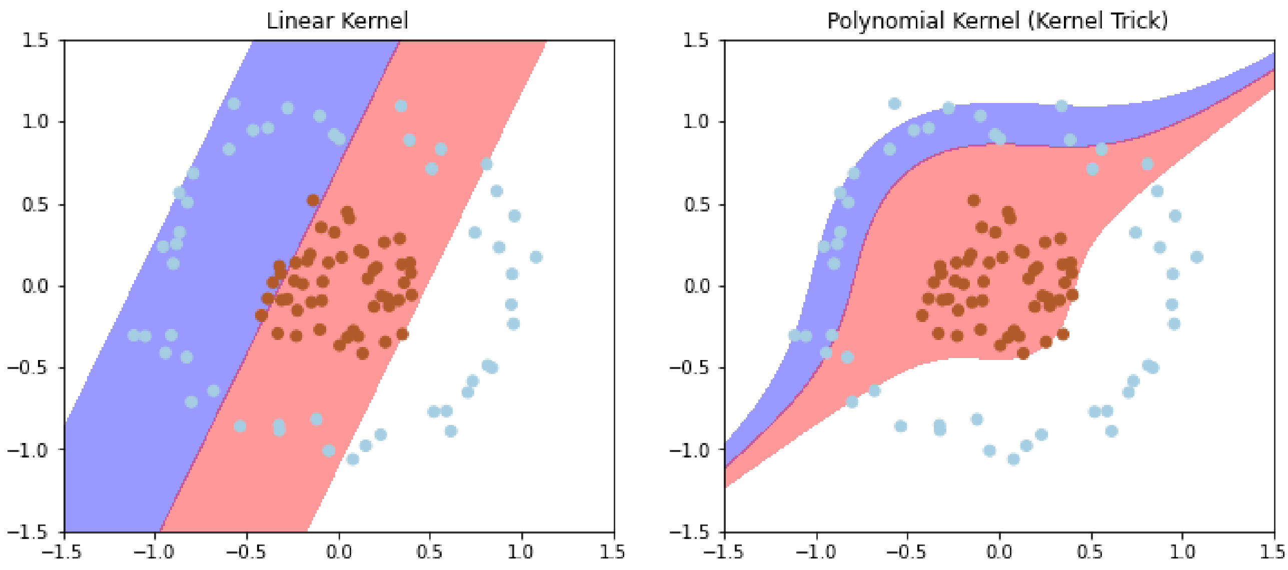


2. SVM with Polynominal Kernel

In this example, we again use a dataset of points that form concentric circles, which is not linearly separable. We create two SVM models: one with a linear kernel and another with a polynomial kernel. The polynomial kernel uses a third-degree polynomial transformation (you can adjust the degree parameter) to make the data linearly separable in a higher-dimensional space.

The decision boundaries and data points are plotted, illustrating the difference between the linear kernel and the polynomial kernel, which applies the kernel trick to transform the data.

In [2]:
```
# Create an SVM model with a linear kernel
svm_linear = SVC(kernel='linear')
svm_linear.fit(X, y)

# Create an SVM model with a polynomial kernel (kernel trick)
svm_poly = SVC(kernel='poly', degree=3)  # Using a third-degree polynomial kernel
svm_poly.fit(X, y)

# Create a meshgrid to visualize the decision boundaries
xx, yy = np.meshgrid(np.linspace(-1.5, 1.5, 200), np.linspace(-1.5, 1.5, 200))
Z_linear = svm_linear.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z_linear = Z_linear.reshape(xx.shape)
Z_poly = svm_poly.decision_function(np.c_[xx.ravel(), yy.ravel()])
Z_poly = Z_poly.reshape(xx.shape)

# Plot the data points and decision boundaries
plt.figure(figsize=(12, 5))
plt.subplot(121)
plt.contourf(xx, yy, Z_linear, levels=[-1, 0, 1], colors=['blue', 'red'], alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.title('Linear Kernel')

plt.subplot(122)
plt.contourf(xx, yy, Z_poly, levels=[-1, 0, 1], colors=['blue', 'red'], alpha=0.4)
plt.scatter(X[:, 0], X[:, 1], c=y, cmap=plt.cm.Paired)
plt.title('Polynomial Kernel (Kernel Trick)')

plt.show()
```


Linear Kernel
Polynomial Kernel (Kernel Trick)