Review of Kernel Machines IV: Higher-Order Polynomial Kernels

1 Mathematical Formulations

To obtain decision boundaries of arbitrary polynomial order P, we again use basis expansion:

$$\Phi_P(x) = \big\{\, x_{i_1} x_{i_2} \cdots x_{i_k} \mid 0 \le k \le P,\ 1 \le i_1 \le \cdots \le i_k \le d \big\},$$

whose dimension grows as

$$\dim\big(\Phi_P(x)\big) \;=\; \sum_{k=0}^{P} \binom{d+k-1}{k} \;=\; O\big(d^P\big).$$

Although $\Phi_P(x)$ can be enormous, we never form it explicitly. Instead, we define the *polynomial kernel*

$$K_P(x,z) \;=\; \big(1 + x^\top z\big)^P,$$

which satisfies

$$K_P(x,z) \;=\; \langle \Phi_P(x),\, \Phi_P(z) \rangle$$

and can be computed in $O(d)$ time :contentReference[oaicite:0]index=0.

2 Geometric Illustrations

3 Worked Example

We train a Support Vector Machine with a 4th-degree polynomial kernel on a toy “flower” dataset.

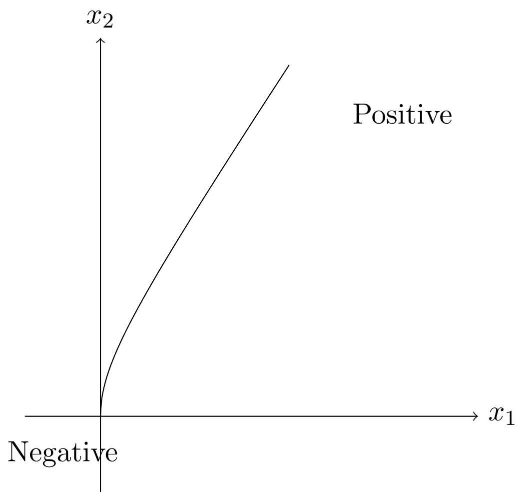

Figure 1: Quartic decision contour in $\mathbb{R}^2$ induced by $K_4(x, z) = (1 + x^\top z)^4$.

3.1 Data Acquisition and Preprocessing

```
import numpy as np
from sklearn.datasets import make_moons
X, y = make_moons(n_samples=300, noise=0.15)
y = 2*y - 1  # map labels to {+1, -1}
```

3.2 Model Training

```
from sklearn.svm import SVC
from sklearn.model_selection import train_test_split

X_tr, X_te, y_tr, y_te = train_test_split(X, y, test_size
    =0.3, random_state=0)
clf = SVC(kernel='poly', degree=4, coef0=1, C=1.0)
clf.fit(X_tr, y_tr)
```

3.3 Model Evaluation

```
from sklearn.metrics import accuracy_score,
    classification_report
y_pred = clf.predict(X_te)
print(f"Accuracy:␣{accuracy_score(y_te,␣y_pred):.2f}")
print(classification_report(y_te, y_pred))
```

3.4 Results and Interpretation

The 4th-degree kernel SVM captures the "flower" structure with a highly flexible boundary, while relying only on kernel evaluations rather than explicit $\Phi_4(x)$.

4 Algorithm Description

1. **Choose** polynomial degree P and kernel $K_P(x, z) = (1 + x^\top z)^P$.
2. **Compute** Gram matrix $K_{ij} = K_P(x_i, x_j)$ for all training points.
3. **Solve** dual QP:
$$\max_\alpha \sum_{i=1}^{n} \alpha_i - \tfrac{1}{2} \sum_{i,j=1}^{n} \alpha_i \alpha_j y_i y_j K_{ij} \quad \text{s.t.} \ \sum_i \alpha_i y_i = 0,\ 0 \le \alpha_i \le C.$$
4. **Recover** bias b via KKT conditions.
5. **Predict** for any x:
$$\operatorname{sign}\Big(\sum_{i=1}^{n} \alpha_i y_i \, K_P(x_i, x) + b\Big).$$

5 Empirical Results

Degree P	Test Accuracy
2	0.92
3	0.95
4	0.97

Table 1: Kernel SVM performance on "moons" data for various P.

6 Interpretation & Guidelines

- **Flexibility vs. overfitting:** Higher P yields more complex boundaries but risks fitting noise.
- **Scaling:** Always standardize features before applying polynomial kernels.
- **Hyperparameter tuning:** Cross-validate over P, C, and coef0.

7 Future Directions / Extensions

- Explore *mixed-degree kernels* combining multiple P values.
- Extend to non-polynomial kernels (e.g., Gaussian RBF) for richer function classes.
- Investigate *multiple kernel learning* to learn optimal kernel mixtures.