支持向量机（SVM）：理论与实践

2025 年 9 月 10 日

1 引言

支持向量机（SVM）属于基于间隔（margin）的判别式模型，通过最大化类间间隔来提升泛化能力。引入核函数（kernel）后，可在保持凸优化性质的同时刻画复杂的非线性决策边界。

2 原理与公式

以线性、软间隔 SVM 的原始问题为例，给定标注数据 $\{(\mathbf{x}_i, y_i)\}_{i=1}^n$，$y_i \in \{-1, +1\}$：

$$\min_{\mathbf{w},b,\boldsymbol{\xi}\geq 0} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i \tag{1}$$

$$\text{s.t.} \quad y_i(\mathbf{w}^\top \mathbf{x}_i + b) \geq 1 - \xi_i,\ i = 1, \ldots, n. \tag{2}$$

对偶形式仅依赖于内积，将其替换为核函数 $K(\mathbf{x}, \mathbf{x}')$ 即得非线性 SVM。决策函数为：

$$f(\mathbf{x}) = \sum_{i\in SV} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \quad \hat{y} = \operatorname{sign} f(\mathbf{x}), \tag{3}$$

其中 SV 为支持向量（$\alpha_i > 0$）。常用的 RBF 核为 $K(\mathbf{x}, \mathbf{x}') = \exp(-\gamma\|\mathbf{x} - \mathbf{x}'\|^2)$。超参数 C 控制软间隔惩罚，γ 控制边界复杂度。

3 应用与技巧

- **缩放：**必须进行特征缩放；SVM 对尺度高度敏感。
- **调参：**通过交叉验证选择 C 与（RBF 的）γ；可从对数网格开始（例如 $C \in [0.1, 10]$、$\gamma \in [10^{-2}, 10]$）。
- **类不平衡：**使用 `class_weight=balanced` 进行重加权。

- **概率输出**：在 SVC 中设 probability=True 可获得概率（需额外校准开销）；否则使用 decision_function。
- **多分类**： SVC 采用一对一策略；LinearSVC 采用一对其余。

4 Python 实战

在本章节目录运行下述命令，图片将保存到 figures/：

Listing 1: 生成 SVM 配图

```
python gen_svm_figures.py
```

Listing 2: gen_svm_figures.py 源码

```
"""
Figure generator for the SVM chapter.

Generates illustrative figures and saves them into the chapter's 'figures/'
folder next to this script, regardless of current working directory.

Requirements:
- Python 3.8+
- numpy, matplotlib, scikit-learn

Install (if needed):
  pip install numpy matplotlib scikit-learn

This script avoids newer or experimental APIs for broader compatibility.
"""
from __future__ import annotations

import os
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

try:
    from sklearn.datasets import make_moons, make_classification
    from sklearn.svm import SVC
except Exception:
    raise SystemExit(
```

```
        "Missing scikit-learn. Please install with: pip install scikit-learn"
    )


def _ensure_figures_dir(path: str | None = None) -> str:
    """Create figures directory under this chapter regardless of CWD."""
    if path is None:
        base = os.path.dirname(os.path.abspath(__file__))
        path = os.path.join(base, "figures")
    os.makedirs(path, exist_ok=True)
    return path


def _plot_decision_boundary(ax, clf, X, y, title: str):
    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(
        np.linspace(x_min, x_max, 400), np.linspace(y_min, y_max, 400)
    )
    Z = clf.predict(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
    cmap_light = ListedColormap(["#FFEEEE", "#EEEEFF"])
    cmap_bold = ListedColormap(["#E74C3C", "#3498DB"])
    ax.contourf(xx, yy, Z, cmap=cmap_light, alpha=0.8, levels=np.unique(Z).size)
    ax.scatter(X[:, 0], X[:, 1], c=y, cmap=cmap_bold, edgecolors="k", s=20)
    ax.set_title(title)
    ax.set_xlabel("Feature 1")
    ax.set_ylabel("Feature 2")


def fig_svm_linear_vs_rbf(out_dir: str) -> str:
    np.random.seed(0)
    X, y = make_moons(n_samples=500, noise=0.3, random_state=0)
    models = [
        (SVC(kernel="linear", C=1.0, random_state=0), "Linear kernel"),
        (SVC(kernel="rbf", C=1.0, gamma=1.0, random_state=0), "RBF kernel"),
    ]
    fig, axes = plt.subplots(1, 2, figsize=(9.5, 4.2), dpi=150, sharex=True, sharey=True)
```

```
    for ax, (m, title) in zip(axes, models):
        m.fit(X, y)
        _plot_decision_boundary(ax, m, X, y, f"SVM: {title}")
    fig.suptitle("SVM: Linear vs RBF kernel")
    out_path = os.path.join(out_dir, "svm_linear_vs_rbf.png")
    fig.tight_layout(rect=[0, 0.03, 1, 0.95])
    fig.savefig(out_path)
    plt.close(fig)
    return out_path


def fig_svm_C_compare(out_dir: str) -> str:
    np.random.seed(1)
    X, y = make_moons(n_samples=500, noise=0.3, random_state=1)
    models = [
        (SVC(kernel="rbf", C=0.3, gamma=1.0, random_state=1), "C=0.3 (
           more regularized)"),
        (SVC(kernel="rbf", C=100.0, gamma=1.0, random_state=1), "C=100
           (less regularized)"),
    ]
    fig, axes = plt.subplots(1, 2, figsize=(9.5, 4.2), dpi=150, sharex=
       True, sharey=True)
    for ax, (m, title) in zip(axes, models):
        m.fit(X, y)
        _plot_decision_boundary(ax, m, X, y, f"SVM (RBF): {title}")
    fig.suptitle("Effect of C (soft-margin)")
    out_path = os.path.join(out_dir, "svm_C_compare.png")
    fig.tight_layout(rect=[0, 0.03, 1, 0.95])
    fig.savefig(out_path)
    plt.close(fig)
    return out_path


def fig_svm_gamma_compare(out_dir: str) -> str:
    np.random.seed(2)
    X, y = make_moons(n_samples=500, noise=0.3, random_state=2)
    models = [
        (SVC(kernel="rbf", C=3.0, gamma=0.2, random_state=2), "gamma
           =0.2 (smoother)"),
        (SVC(kernel="rbf", C=3.0, gamma=5.0, random_state=2), "gamma
           =5.0 (wiggly)")
    ]
    fig, axes = plt.subplots(1, 2, figsize=(9.5, 4.2), dpi=150, sharex=
```

```
        True, sharey=True)
    for ax, (m, title) in zip(axes, models):
        m.fit(X, y)
        _plot_decision_boundary(ax, m, X, y, f"SVM (RBF): {title}")
    fig.suptitle("Effect of gamma (RBF width)")
    out_path = os.path.join(out_dir, "svm_gamma_compare.png")
    fig.tight_layout(rect=[0, 0.03, 1, 0.95])
    fig.savefig(out_path)
    plt.close(fig)
    return out_path


def fig_svm_margin_support_vectors(out_dir: str) -> str:
    # Linearly separable-like data for margin visualization
    X, y = make_classification(
        n_samples=200,
        n_features=2,
        n_redundant=0,
        n_informative=2,
        n_clusters_per_class=1,
        class_sep=2.0,
        random_state=3,
    )
    clf = SVC(kernel="linear", C=1e3, random_state=3)
    clf.fit(X, y)

    fig, ax = plt.subplots(figsize=(6.5, 4.8), dpi=160)
    _plot_decision_boundary(ax, clf, X, y, "Linear SVM with margin and
        SVs")

    # Plot the margin lines using w^T x + b = +/-1
    w = clf.coef_[0]
    b = clf.intercept_[0]
    # Create a grid line in x for margin lines
    x_vals = np.linspace(X[:, 0].min() - 0.5, X[:, 0].max() + 0.5, 200)
    # For y = -(w0*x + b - m)/w1 with m in {0, 1, -1}
    if abs(w[1]) > 1e-12:
        for m in [0.0, 1.0, -1.0]:
            y_vals = -(w[0] * x_vals + b - m) / w[1]
            style = "k-" if m == 0 else "k--"
            ax.plot(x_vals, y_vals, style, lw=1.2, alpha=0.9)

    # Highlight support vectors
```

```
    sv = clf.support_vectors_
    ax.scatter(sv[:, 0], sv[:, 1], s=80, facecolors="none", edgecolors="#000", linewidths=1.5, label="SV")
    ax.legend(loc="best")
    out_path = os.path.join(out_dir, "svm_margin_support_vectors.png")
    fig.tight_layout()
    fig.savefig(out_path)
    plt.close(fig)
    return out_path


def fig_svm_decision_function(out_dir: str) -> str:
    np.random.seed(4)
    X, y = make_moons(n_samples=400, noise=0.25, random_state=4)
    clf = SVC(kernel="rbf", C=2.0, gamma=1.0, random_state=4)
    clf.fit(X, y)

    x_min, x_max = X[:, 0].min() - 0.5, X[:, 0].max() + 0.5
    y_min, y_max = X[:, 1].min() - 0.5, X[:, 1].max() + 0.5
    xx, yy = np.meshgrid(
        np.linspace(x_min, x_max, 500), np.linspace(y_min, y_max, 500)
    )
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)

    fig, ax = plt.subplots(figsize=(6.5, 5.0), dpi=160)
    # Filled regions by sign
    ax.contourf(xx, yy, (Z > 0).astype(int), levels=2, cmap=ListedColormap(["#FFEEEE", "#EEEEFF"]), alpha=0.8)
    # Decision function contours for -1, 0, +1
    CS = ax.contour(xx, yy, Z, levels=[-1.0, 0.0, 1.0], colors=["k", "k", "k"], linestyles=["--", "-", "--"], linewidths=1.2)
    ax.clabel(CS, inline=True, fontsize=8, fmt={-1.0: "-1", 0.0: "0", 1.0: "+1"})
    # Data points and SVs
    ax.scatter(X[:, 0], X[:, 1], c=y, cmap=ListedColormap(["#E74C3C", "#3498DB"]), edgecolors="k", s=20)
    sv = clf.support_vectors_
    ax.scatter(sv[:, 0], sv[:, 1], s=80, facecolors="none", edgecolors="#000", linewidths=1.5, label="SV")
    ax.set_title("RBF SVM: decision function and margins")
    ax.set_xlabel("Feature 1")
    ax.set_ylabel("Feature 2")
```

```
180     ax.legend(loc="best")
181     out_path = os.path.join(out_dir, "svm_decision_function.png")
182     fig.tight_layout()
183     fig.savefig(out_path)
184     plt.close(fig)
185     return out_path
186 
187 
188 def main():
189     out_dir = _ensure_figures_dir(None)
190     generators = [
191         fig_svm_linear_vs_rbf,
192         fig_svm_C_compare,
193         fig_svm_gamma_compare,
194         fig_svm_margin_support_vectors,
195         fig_svm_decision_function,
196     ]
197     print("Generating figures into:", os.path.abspath(out_dir))
198     for gen in generators:
199         try:
200             p = gen(out_dir)
201             print("Saved:", p)
202         except Exception as e:
203             print("Failed generating", gen.__name__, ":", e)
204 
205 
206 if __name__ == "__main__":
207     main()
```

5 结果

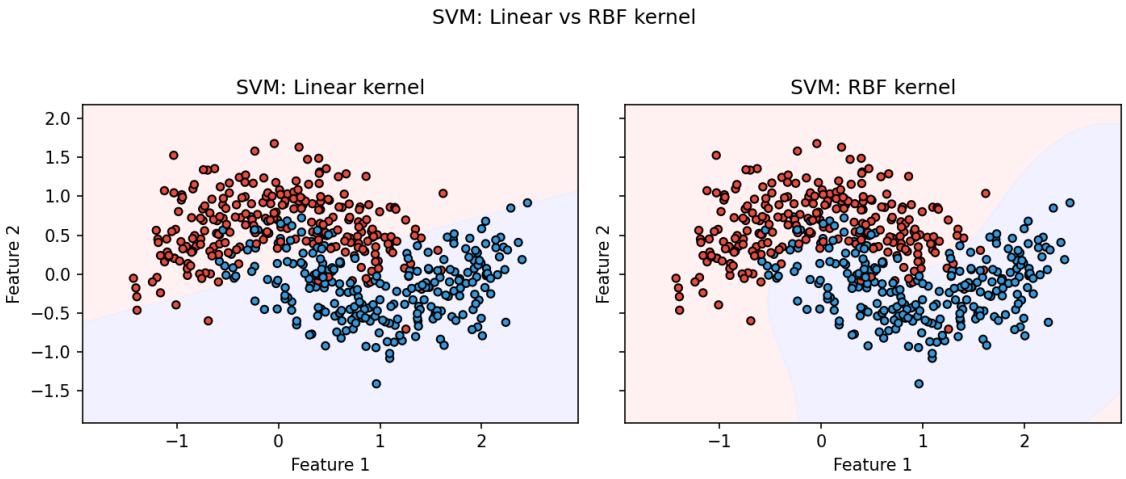

图 1: SVM 决策边界：线性核 vs RBF 核。

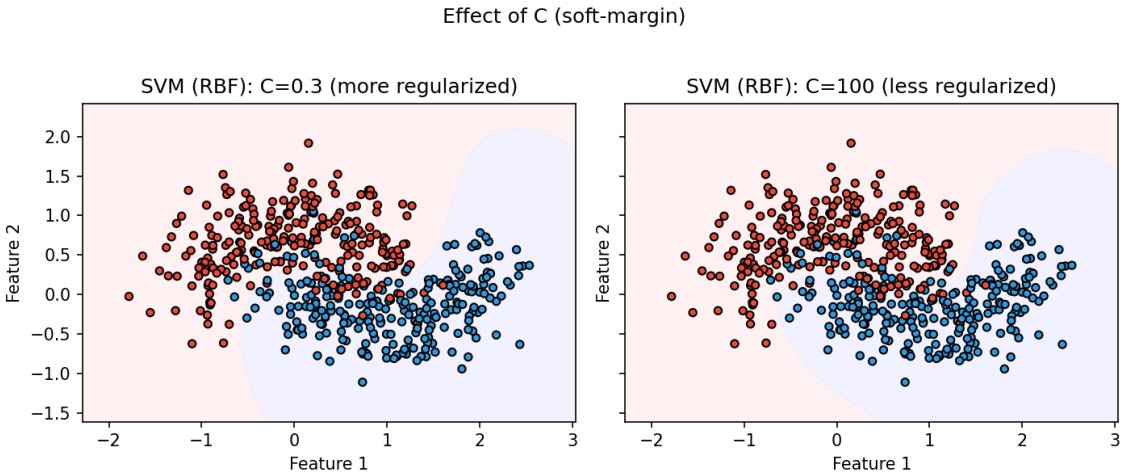

图 2: 软间隔参数 C 的影响（RBF 核）。

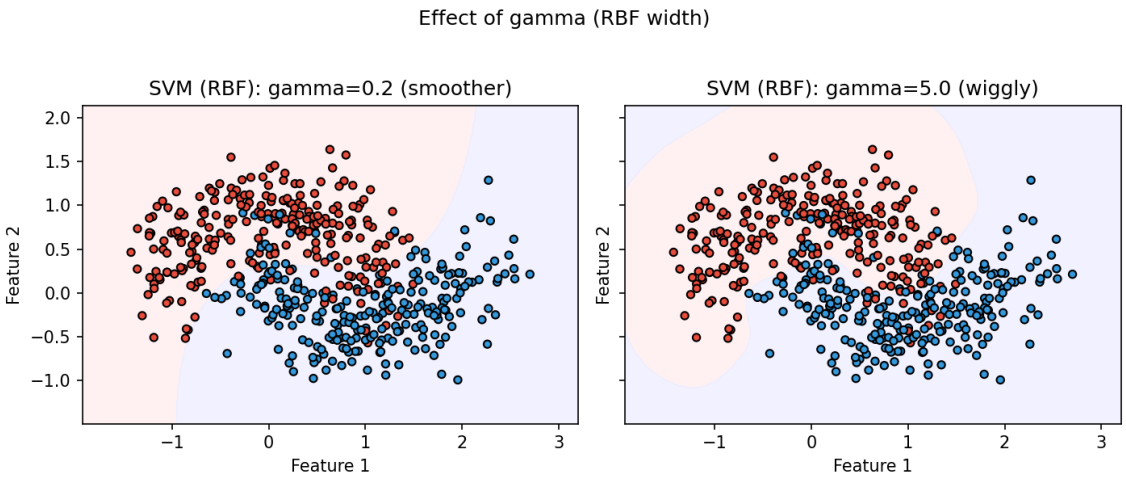

图 3: RBF 的 γ 对边界平滑度的影响。

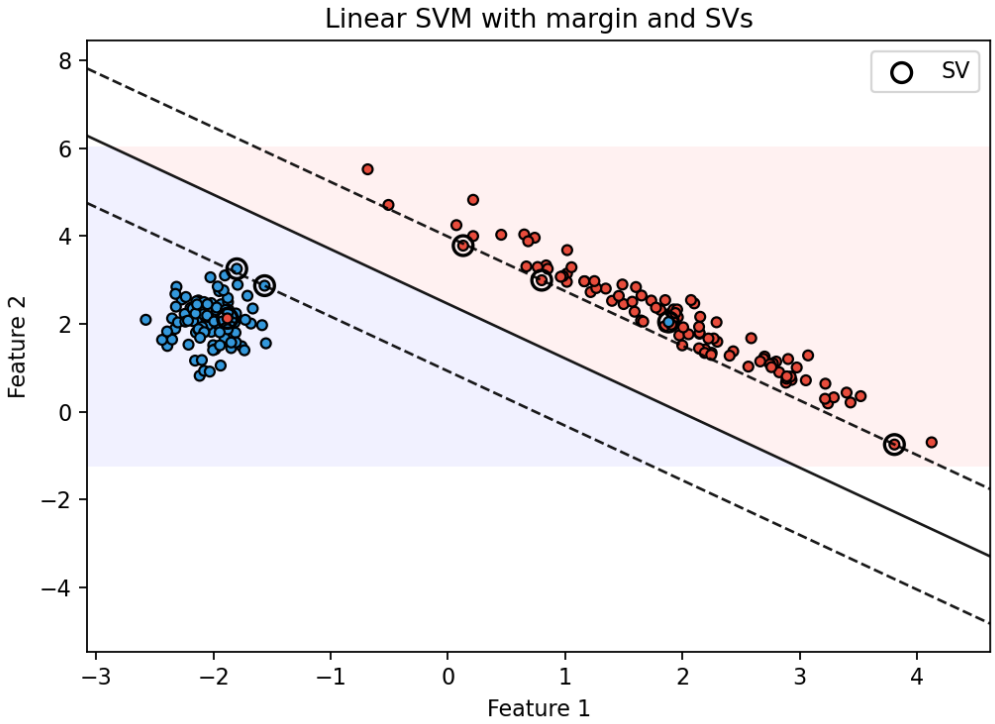

图 4: 线性 SVM：间隔线与支持向量可视化。

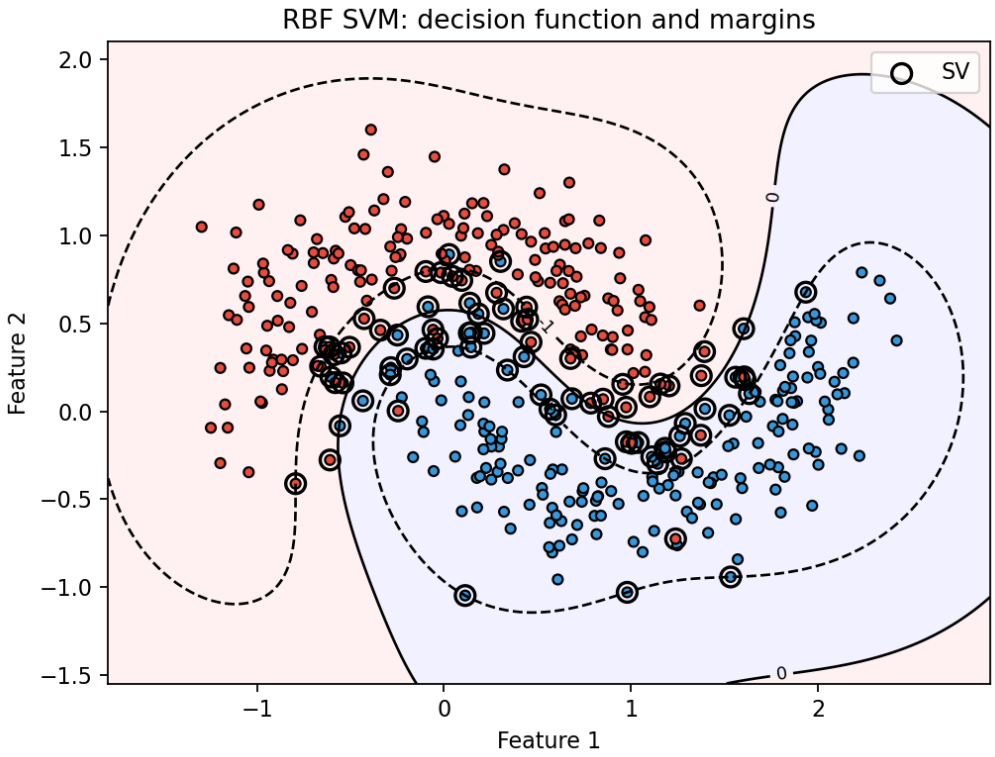

图 5: RBF SVM 的决策函数：-1、0、+1 等值线。

6 总结

SVM 通过最大化间隔实现稳健分类，并可借助核函数处理非线性问题。在恰当的特征缩放与 C、核参数的调参与验证下，SVM 常在多种任务上表现优异。