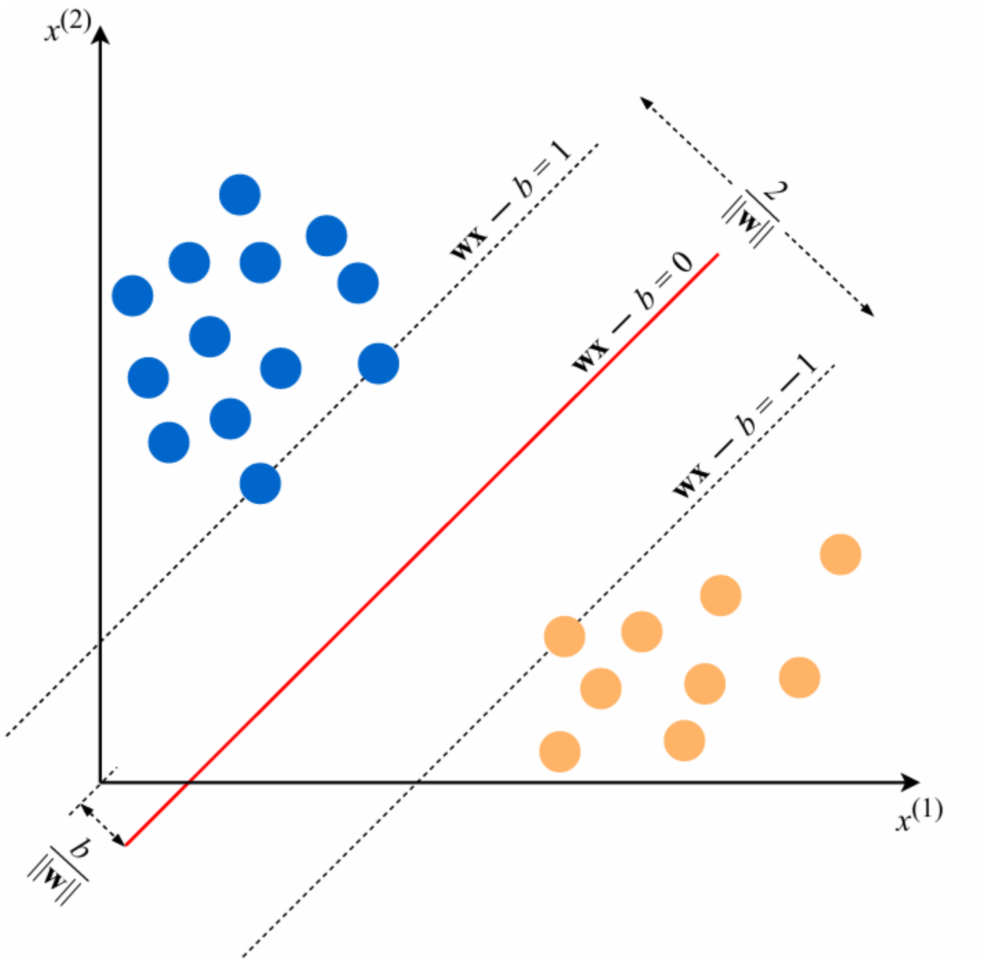
Chapter 3 : Fundamental Algorithms

Support Vector Machine (SVM)



SVM Optimization Review

Linear Separation

Support Vectors

SVM Optimization Review

Original SVM Constraints:

- $\mathbf{w}^T\mathbf{x}_i - b \geq +1$ if $y_i = +1$
- $\mathbf{w}^T\mathbf{x}_i - b \leq -1$ if $y_i = -1$

Objective: Minimize $\frac{1}{2}||\mathbf{w}||^2$

Combined Constraint: $y_i(\mathbf{w}^T\mathbf{x}_i - b) \geq 1$

Optimization Problem:

$$\min \frac{1}{2}||\mathbf{w}||^2 \text{ subject to } y_i(\mathbf{x}_i^T\mathbf{w} - b) - 1 \geq 0$$

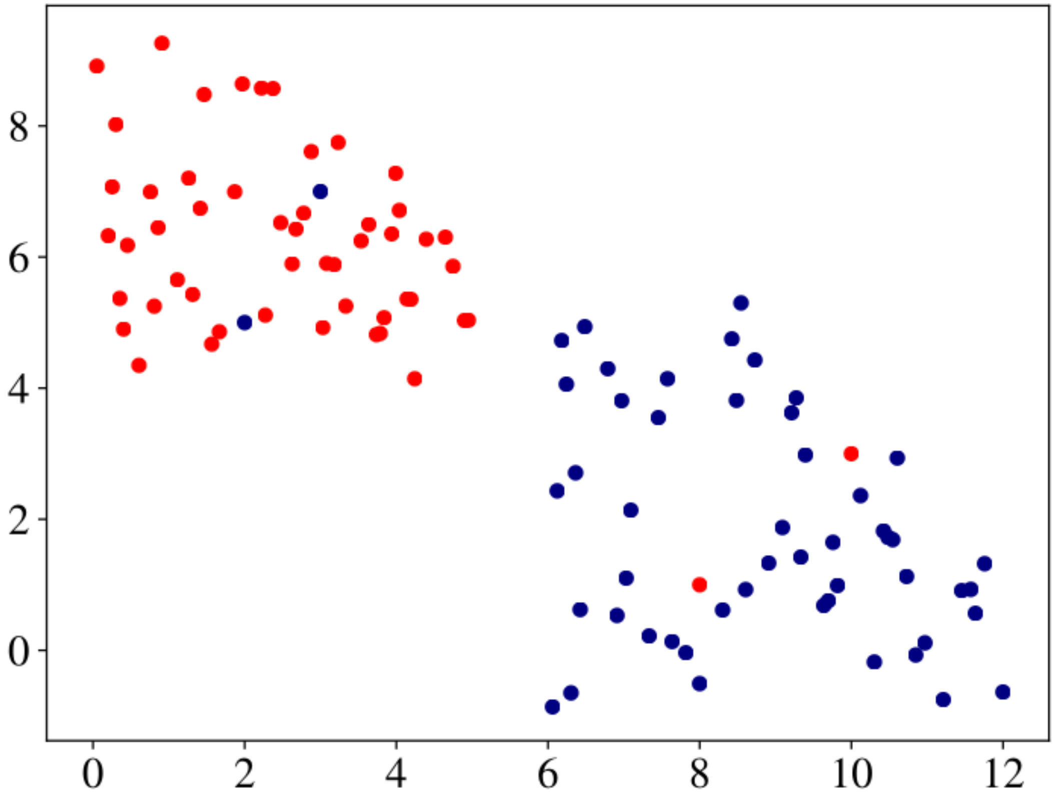
SVM: Two Critical Questions

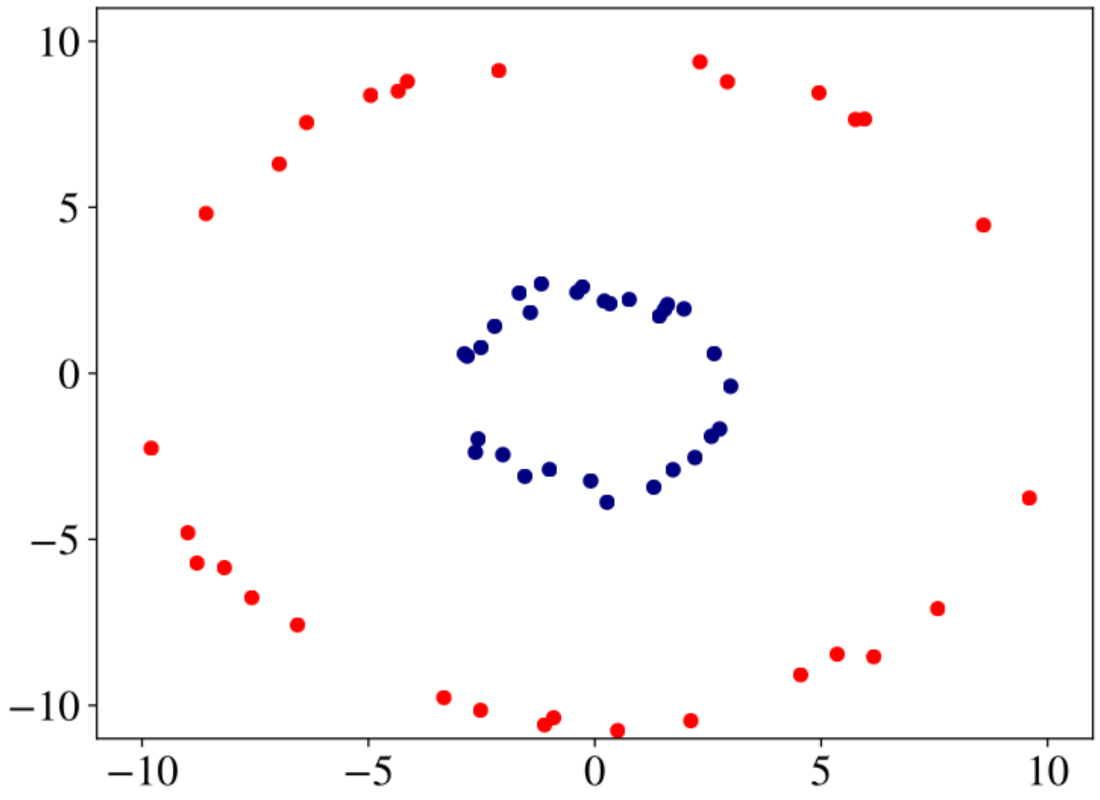
We already considered SVM basics, so this section fills important gaps:

1. **Noisy Data Problem:**
   - What if no hyperplane can perfectly separate positive from negative examples?
   - How to handle outliers and mislabeled examples?
2. **Non-Linear Data Problem:**
   - What if data cannot be separated by a straight line?
   - How to handle circular or curved decision boundaries?

Real World: Most data has both challenges!

SVM: Two Critical Questions





The Problem Illustrated

Left: Noisy Data

- Data could be separated by straight line
- But outliers prevent perfect separation
- Need flexibility to handle noise

Right: Non-Linear Data

- Decision boundary is circular, not linear
- Linear hyperplane cannot solve this
- Need transformation to higher dimensions

Goal: Extend SVM to handle both scenarios

Problem 1: Dealing with Noise

Solution: Introduce **Hinge Loss Function**

Hinge Loss:

$$\ell_{\text{hinge}}(y, f(\mathbf{x})) = \max(0, 1 - y \cdot f(\mathbf{x}))$$

Properties:

- **Zero loss** if constraints satisfied (correct side of boundary)
- **Linear penalty** proportional to distance from boundary
- **Robust** to outliers compared to squared loss

Interpretation:

- If $y \cdot f(\mathbf{x}) \geq 1$: No penalty (correct classification with margin)
- If $y \cdot f(\mathbf{x}) < 1$: Linear penalty increases with distance

Soft-Margin SVM

New Cost Function:

$$C||\mathbf{w}||^2 + \sum_{i=1}^{N} \max(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - b))$$

Two Competing Objectives:

1. **Maximize margin:** Minimize $||\mathbf{w}||^2$
2. **Minimize errors:** Minimize hinge loss

Hyperparameter C controls trade-off:

- **High C:** Focus on classification accuracy (small margin)
- **Low C:** Focus on large margin (allow some errors)

Understanding Parameter C

High C (Focus on Accuracy):

- Second term dominates
- Algorithm tries to classify all training points correctly
- May lead to overfitting, Smaller margin

Low C (Focus on Generalization):

- First term dominates
- Algorithm allows some misclassification
- Larger margin for better generalization, More robust to noise

Sweet Spot: Balance between training accuracy and generalization

Problem 2: Non-Linear Data

Challenge: Data cannot be separated by hyperplane in original space

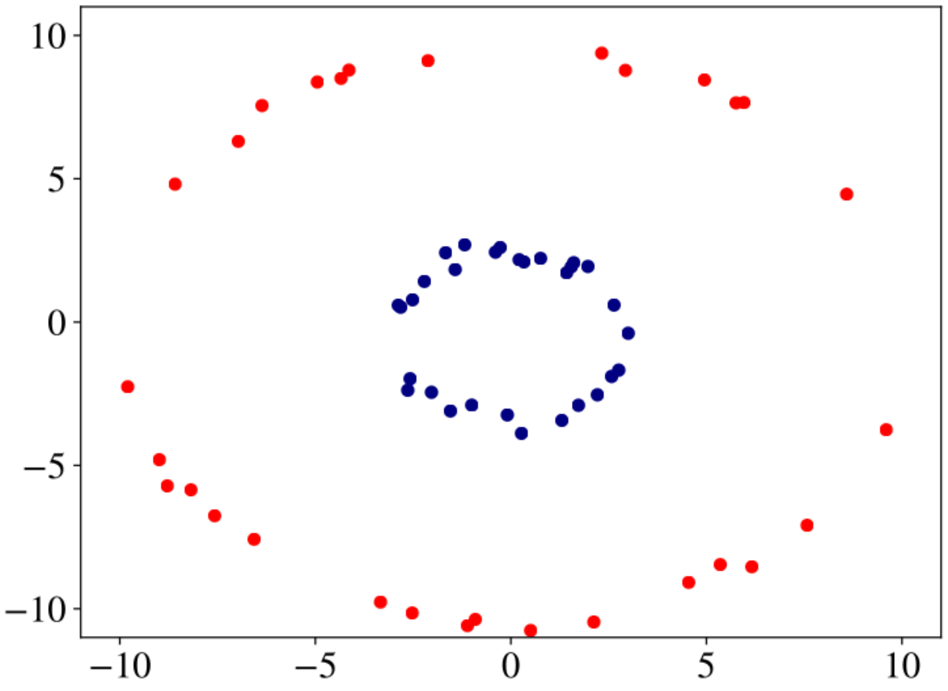
Solution: Transform to higher-dimensional space where linear separation is possible

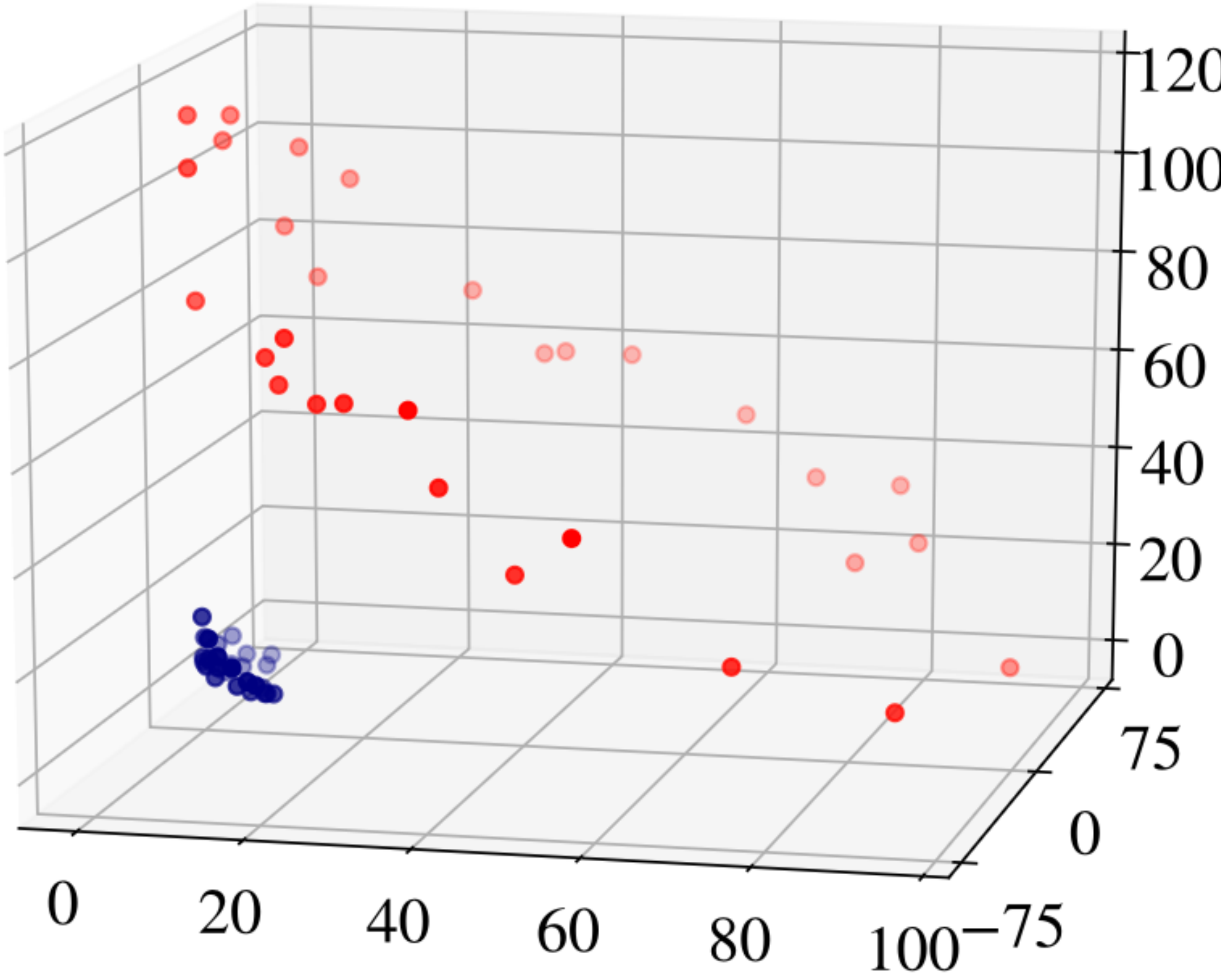
Key Insight: Many non-linear problems become linear in higher dimensions

Example Transformation:

- **2D input:** $\mathbf{x} = [q, p]$
- **3D mapping:** $\phi([q, p]) = [q^2, \sqrt{2}qp, p^2]$
- **Result:** Circular boundary becomes linear in 3D

Dealing with Non Linear Data





The Kernel Trick

Problem with Explicit Transformation:

- Don't know which mapping ϕ will work
- High-dimensional transformations are computationally expensive
- Need to try many different mappings

Kernel Trick Solution:

- Work in high-dimensional space without explicit transformation
- Only need dot products $\phi(\mathbf{x}_i) \cdot \phi(\mathbf{x}_j)$
- Replace with kernel function $k(\mathbf{x}_i, \mathbf{x}_j)$

Magic: Get same result as high-dimensional dot product using simple operation on original vectors

SVM Dual Formulation

Lagrange Multipliers Transform:

Original problem becomes:

$$\max_{\alpha_1 \ldots \alpha_N} \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{k=1}^{N} y_i \alpha_i (\mathbf{x}_i \cdot \mathbf{x}_k) y_k \alpha_k$$

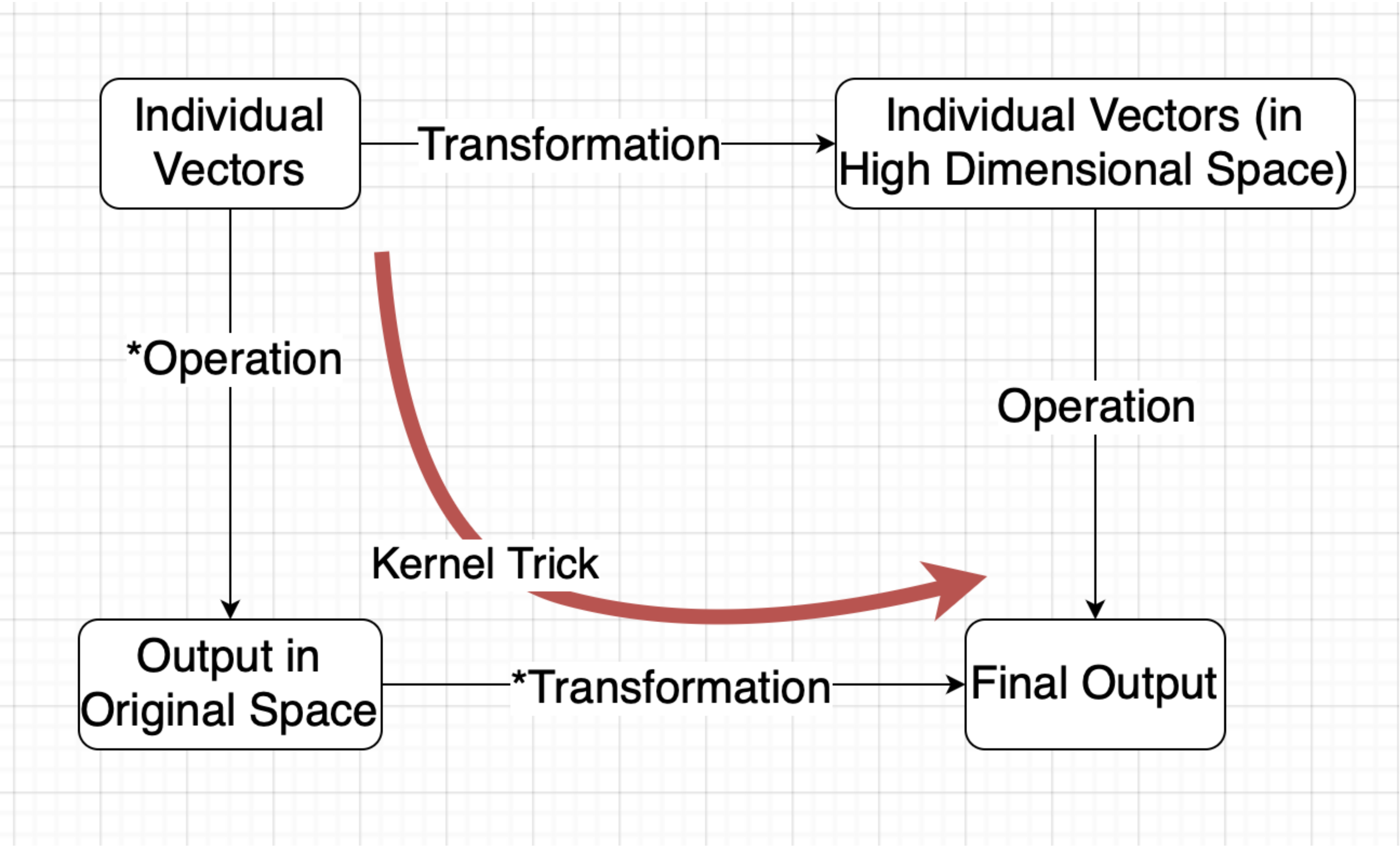
Subject to:

$$\sum_{i=1}^{N} \alpha_i y_i = 0 \text{ and } \alpha_i \geq 0$$

Key Observation: Only uses dot products $\mathbf{x}_i \cdot \mathbf{x}_k$

Kernel Substitution: Replace $\mathbf{x}_i \cdot \mathbf{x}_k$ with $k(\mathbf{x}_i, \mathbf{x}_k)$

Kernel Trick



Kernel Function Examples

Linear Kernel:

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{x}^T \mathbf{x}'$$

Polynomial Kernel:

$$k(\mathbf{x}, \mathbf{x}') = (\mathbf{x}^T \mathbf{x}' + 1)^d$$

Quadratic Example:

- Instead of transforming $(q_1, p_1) \to (q_1^2, \sqrt{2}q_1p_1, p_1^2)$
- Simply compute $(q_1q_2 + p_1p_2)^2$
- **Same result, much faster!**

Kernel Trick: Mathematical Insight

Key Insight: We only need dot-products, not explicit transformations!

Example Transformation:

- **Original vectors:** (q_1, p_1) and (q_2, p_2)
- **Explicit transformation:**
  - $(q_1, p_1) \rightarrow (q_1^2, \sqrt{2}q_1p_1, p_1^2)$
  - $(q_2, p_2) \rightarrow (q_2^2, \sqrt{2}q_2p_2, p_2^2)$
- **Dot-product result:** $(q_1^2q_2^2 + 2q_1q_2p_1p_2 + p_1^2p_2^2)$

Kernel Trick Alternative:

- **Simple operation:** $(q_1q_2 + p_1p_2)^2$
- **Same result:** $(q_1^2q_2^2 + 2q_1q_2p_1p_2 + p_1^2p_2^2)$

RBF (Gaussian) Kernel

Most Popular Kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

Properties:

- **Infinite-dimensional** feature space
- **Smooth decision boundaries**
- **Local influence:** Similar points have high kernel values

RBF (Gaussian) Kernel

Most Popular Kernel:

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2\sigma^2}\right)$$

Hyperparameter σ:

- **Small σ:** Curvy, complex boundaries (high variance)
- **Large σ:** Smooth, simple boundaries (high bias)

Euclidean Distance:

$$||\mathbf{x} - \mathbf{x}'||^2 = \sum_{j=1}^{D}(x_j - x'_j)^2$$

SVM Decision Function

Final Prediction:

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{N} \alpha_i y_i k(\mathbf{x}_i, \mathbf{x}) + b\right)$$

Support Vectors:

- Training points with $\alpha_i > 0$
- Only these points affect the decision boundary
- Typically small subset of training data

Sparsity: Most $\alpha_i = 0$, leading to efficient predictions

SVM Hyperparameter Summary

C (Regularization Parameter):

- Controls margin vs accuracy trade-off
- Higher C → More complex model, Lower C → Simpler model

Kernel Choice:

- **Linear:** For linearly separable data
- **Polynomial:** For moderate non-linearity
- **RBF:** For complex non-linear patterns

RBF Parameter σ:

- Controls smoothness of decision boundary
- Cross-validation typically used for selection

SVM: Strengths and Limitations

Strengths:

- **Effective** in high-dimensional spaces
- **Memory efficient** (uses only support vectors)
- **Versatile** (different kernels for different data)
- **Works well** with small datasets

Limitations:

- **No probability estimates** (only classifications), **Sensitive to feature scaling**
- **Slow on large datasets** (quadratic in training size)
- **Choice of kernel and parameters** can be tricky

Best Use Cases: Medium-sized datasets with complex decision boundaries

SVM Implementation

Check the implementation on Notebook