Non-Linear Problems

Problem:
- some tasks have non-linear structure
- no hyperplane is sufficiently accurate
How can SVMs learn non-linear classification rules?

Example

- Input Space: \( x = (x_1, x_2) \) (2 attributes)
- Feature Space: \( \Phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, \sqrt{2}x_1x_2, 1) \) (6 attributes)

Extending the Hypothesis Space

Idea: add more features

- Kernels can make high-dimensional spaces tractable
- Kernels can make non-vectorial data tractable
- Transform a linear learner into a non-linear learner

Example:

- The separating hyperplane in feature space is degree two polynomial in input space.

Dual (Batch) Perceptron Algorithm

Inputs: \( X = \{(x_m, y_m)\}, \quad y_m \in \{-1, 1\}, \quad m \in [1, 2, \ldots] \)

Dual Algorithm:
- \( k \in [1, m] \) : \( \alpha_k = 0 \)
- \( \epsilon = \frac{\alpha}{\sum \alpha_k} \)
- \( \alpha_{m+1} = -1 \)

- \( \text{FOR} \ m = 1 \ to \ n \)
- \( \text{IF} \ y_m (\sum \alpha_k y_k x_m \cdot x_k) \leq 0 \)
- \( \alpha_m \leftarrow \alpha_m + 1 \)
- ENDIF
- ENDFOR

- until \( \epsilon \) iterations reached

Primal Algorithm:

- \( \text{FOR} \ m = 1 \ to \ n \)
- \( \text{IF} \ y_m (\sum \alpha_k y_k x_m \cdot x_k) \leq 0 \)
- \( \alpha_m \leftarrow \alpha_m + 1 \)
- ENDIF
- ENDFOR

- until \( \epsilon \) iterations reached
**Kernels**

**Problem:** Very many Parameters! Polynomials of degree \( p \) over \( N \) attributes in input space lead to attributes in feature space!

**Solution:** [Boser et al.] The dual OP depends only on inner products \( \Rightarrow \) Kernel Functions

\[
K(\boldsymbol{x}, \boldsymbol{y}) = \mathbf{\Phi}(\boldsymbol{x}) \cdot \mathbf{\Phi}(\boldsymbol{y})
\]

**Example:** For \( \mathbf{\phi}(\boldsymbol{x}) = (x_1^2, x_2^2, \sqrt{2}x_1, \sqrt{2}x_2, x_1x_2, 1) \) calculating \( K(\boldsymbol{x}, \boldsymbol{y}) = [\boldsymbol{x} \cdot \boldsymbol{y} + 1]^2 \) computes inner product in feature space.

\( \Rightarrow \) no need to represent feature space explicitly.

**Properties of SVMs with Kernels**

- **Expressiveness**
  - Can represent any boolean function (for appropriate choice of kernel)
  - Can represent any sufficiently "smooth" function to arbitrary accuracy (for appropriate choice of kernel)

- **Computational**
  - Objective function has no local optima (only one global)
  - Independent of dimensionality of feature space

- **Design decisions**
  - Kernel type and parameters
  - Value of \( C \)

**Examples of Kernels**

- **Polynomial**
  \[
  K(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x} \cdot \boldsymbol{y} + 1)^d
  \]

- **Radial Basis Function**
  \[
  K(\boldsymbol{x}, \boldsymbol{y}) = \exp(-\|\boldsymbol{x} - \boldsymbol{y}\|^2)
  \]

**Kernels for Non-Vectorial Data**

- **Applications with Non-Vectorial Input Data**
  - Classify non-vectorial objects
    - Protein classification (\( x \) is string of amino acids)
    - Drug activity prediction (\( x \) is molecule structure)
    - Information extraction (\( x \) is sentence of words)
    - Etc.

- **Applications with Non-Vectorial Output Data**
  - Predict non-vectorial objects
    - Natural Language Parsing (\( y \) is parse tree)
    - Noun-Phrase Co-reference Resolution (\( y \) is clustering)
    - Search engines (\( y \) is ranking)

\( \Rightarrow \) Kernels can compute inner products efficiently!