Modern Machine Learning — Support Vector Machines and Kernels

Kenneth E. Barner
Department of Electrical and Computer Engineering
University of Delaware

Motivation: Support Vector classifiers determine linear boundaries in the observation/feature space. How do we make the procedure more flexible and enable nonlinear boundaries?

Solution Options: Enlarge the feature space using basis expansion, such as polynomials or splines.

Determine appropriate basis functions: $h_m(\boldsymbol{x}), m = 1, \ldots, M$

Determine the SV classifier on the transformed input features:

$$h(\boldsymbol{x}_i) = (h_1(\boldsymbol{x}_i), h_2(\boldsymbol{x}_i), \ldots, h_M(\boldsymbol{x}_i))$$

The SV establishes linear decision boundaries in the transformed feature space: $h(\boldsymbol{x})$

The function $\hat{f}(\boldsymbol{x}) = h(\boldsymbol{x})^T \widehat{\boldsymbol{\beta}} + \hat{\beta}_0$ is nonlinear in the original observation/feature space: $\boldsymbol{x}$

Resulting classifier is: $\hat{G}(\boldsymbol{x}) = \text{sign}(\hat{f}(\boldsymbol{x}))$

Support Vector Machines: SVM classifiers are an extension of this approach, where the dimension of the expanded space is allowed to be very large, or even infinite in some cases

Recall the SV Lagrangian dual function:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j\, y_i y_j \mathbf{x}_i^T \mathbf{x}_j$$

Applying the transformations $h(\cdot)$ to the observations

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j\, y_i y_j h(\mathbf{x}_i)^T h(\mathbf{x}_j)$$

Notice that L_D depends only on the inner product $\langle h(\mathbf{x}_i), h(\mathbf{x}_j) \rangle = h(\mathbf{x}_i)^T h(\mathbf{x}_j)$

Thus

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j\, y_i y_j \langle h(\mathbf{x}_i), h(\mathbf{x}_j) \rangle$$

Also, the solution $\boldsymbol{\beta}$ for is now:

$$\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i\, y_i h(\boldsymbol{x}_i)$$

Thus utilizing $\boldsymbol{\beta} = \sum_{i=1}^{n} \alpha_i\, y_i h(\boldsymbol{x}_i)$, the solution function can be written as:

$$\begin{aligned} f(\boldsymbol{x}) &= h(\boldsymbol{x})^T \boldsymbol{\beta} + \beta_0 \\ &= \sum_{i=1}^{n} \alpha_i\, y_i \langle h(\boldsymbol{x}), h(\mathbf{x}_i)\rangle + \beta_0 \end{aligned}$$

Observations:

- L_D and $f(\boldsymbol{x})$ involve $h(\boldsymbol{x})$ only through the inner product
- In fact, the transformation $h(\boldsymbol{x})$ need not be explicitly known
- All that is required is knowledge of the Kernel Function:

  $$K(\boldsymbol{x}, \boldsymbol{x}') = \langle h(\boldsymbol{x}), h(\boldsymbol{x}')\rangle$$

  which computes the inner product in the transformed space

Definition: Formally, a function $K: \mathbb{R}^p \times \mathbb{R}^p \to \mathbb{R}$ is a kernel if:

- K is symmetric: $K(\boldsymbol{x}, \boldsymbol{x}') = K(\boldsymbol{x}', \boldsymbol{x})$
- K is positive semi-definite, i.e., define the matrix $\boldsymbol{K}$ to have elements $K_{ij} = K(\boldsymbol{x}_i, \boldsymbol{x}_j)$ for any $\boldsymbol{x}_1, \dots, \boldsymbol{x}_n \in \mathbb{R}^p$, then it must hold that
$$\boldsymbol{a}^T \boldsymbol{K} \boldsymbol{a} \geq 0, \forall \boldsymbol{a} \mathbb{R}^n$$

Commonly utilized SVM kernels include:

$$d\text{th} - \text{Degree polynomial:}\, K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d$$
$$\text{Radial basis:}\, K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$$
$$\text{Neural network:}\, K(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\kappa_1 \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \kappa_2)$$

Support Vector Machines & Kernels

Thus we can replace $h(\cdot)$ with any symmetric positive semidefinite kernel K

$$\begin{aligned} L_D &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j\, y_i y_j h(\mathbf{x}_i)^T h\big(\mathbf{x}_j\big) \\ &= \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j\, y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned}$$

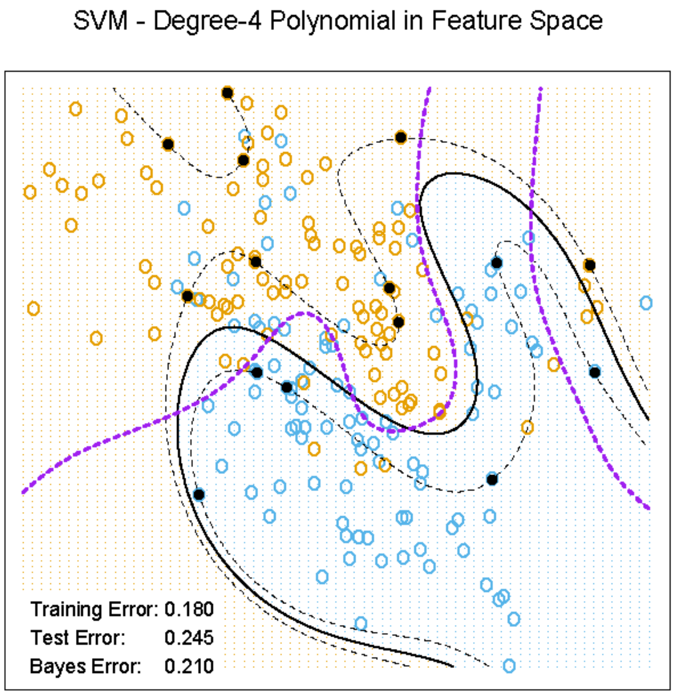
Quadratic programming optimization is employed, taking into account the standard constraints, following which

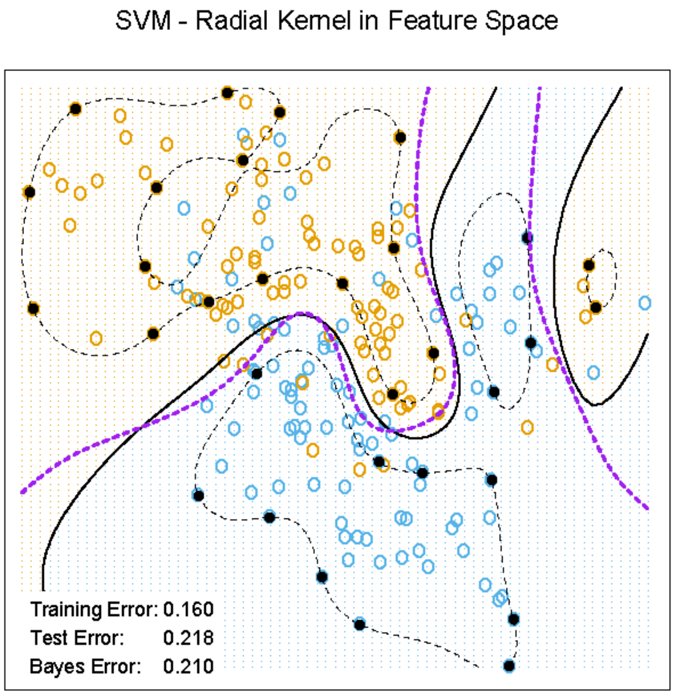
$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i\, y_i \boldsymbol{x}_i, \qquad \text{and} \qquad \hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i\, K(\boldsymbol{x}, \boldsymbol{x}_i) + \beta_0$$

Comments:

- The effect of the parameter C is more obvious given the enlarged, transformed feature space
- Perfect (linear) separation is often achievable in the high-dimension, transformed feature space
- Large C suppresses the slack variables (small ξ_i), resulting in over fitting in the original feature space
- Small C allows for larger slack variables and encourages a smaller $||\boldsymbol{\beta}||$, resulting in a smoother boundary (less over fitting)

Support Vector Machines & Kernels





Example: Mixture data with overlapping classes. LEFT: 4th degree polynomial kernel. RIGHT: radial basis kernel with $\gamma = 1$. C was tuned to achieve the best test error performance; $C = 1$ worked well in both cases. Radial basis kernel performed best (close to the optimal Bayes decision boundary, purple line), which is expected since data is from a Gaussian mixture. Dark dots indicate support vectors on the margin.

Summary:

- Support Vector Machines map the observation/feature space to a higher dimensional space via a transformation $h(\boldsymbol{x})$
- Support Vector Machines establish linear boundaries between classes in the high-dimension transformed space; the decision boundaries in the original observation/feature space are nonlinear
- The transformation $h(\boldsymbol{x})$ only comes into play as an inner product that can be represented as a kernel function
- Valid kernel functions must be symmetric and positive semi-definite
- Commonly employed SVM kernels include:

$$d\text{th} - \text{Degree polynomial: } K(\boldsymbol{x}, \boldsymbol{x}') = (1 + \langle \boldsymbol{x}, \boldsymbol{x}' \rangle)^d$$
$$\text{Radial basis: } K(\boldsymbol{x}, \boldsymbol{x}') = \exp(-\gamma \|\boldsymbol{x} - \boldsymbol{x}'\|_2^2)$$
$$\text{Neural network: } K(\boldsymbol{x}, \boldsymbol{x}') = \tanh(\kappa_1 \langle \boldsymbol{x}, \boldsymbol{x}' \rangle + \kappa_2)$$

- The Wolf dual SVM optimization (to be maximized) is:

$$L_D = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j \, y_i y_j K(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

subject to $0 \leq \alpha_i \leq C$ and $\sum_{i=1}^{n} \alpha_i \, y_i = 0$

- Quadratic programming optimization is employed, the results of which are used to set:

$$\widehat{\boldsymbol{\beta}} = \sum_{i=1}^{n} \hat{\alpha}_i \, y_i \boldsymbol{x}_i, \qquad \text{and} \qquad \hat{f}(\boldsymbol{x}) = \sum_{i=1}^{n} \hat{\alpha}_i y_i \, K(\boldsymbol{x}, \boldsymbol{x}_i) + \beta_0$$

- Classification is determined as $G(\boldsymbol{x}) = \operatorname{sgn}\left(\hat{f}(\boldsymbol{x})\right)$
- The tuning parameter C controls the margin; because of the high dimensionality of the transformed space, the effects are more pronounced: Small $C \Longrightarrow$ a larger margin & less over fitting, while large $C \Longrightarrow$ a smaller margin & more over fitting