Tufts

Class #10:
Support Vector Machines (SVMs) and Kernel Functions, II

Machine Learning (COMP 135)

1

Review: Support Vector Machines (SVMs)

1. Start with labeled data-set:

$$\{(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_n, y_n)\} \qquad [\forall i,\ y_i \in \{+1, -1\}]$$

2. Solve constrained quadratic optimization problem:

$$\text{Maximize:} \quad W(\alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i\, \alpha_j\, y_i\, y_j (\mathbf{x}_i \cdot \mathbf{x}_j)$$

while satisfying constraints:

$$\forall i,\ \alpha_i \geq 0$$

$$\sum_i \alpha_i\, y_i = 0$$

3. Derive necessary classification weights ***when and if*** needed; typically, instead, use dual form to compute the classification hypothesis:

$$\mathbf{w} \cdot \mathbf{x}_i + b = \sum_j \alpha_j y_j (\mathbf{x}_i \cdot \mathbf{x}_j) + b$$

2

Retaining the Support Vectors

- After computing the various optimizing α values, the SVM typically ends up with:
  1. A large number of data points $\mathbf{x}_i$ with $\alpha_i = 0$
  2. A few special data points $\mathbf{x}_j$ with $\alpha_j \neq 0$
- These special points, the support vectors, can be used by themselves to compute necessary weights and biases
  - Often, the SVM keeps a list of these vectors, for computation of later classification functions, rather than the weights defining the classification boundary directly

3

Pros and Cons of SVMs

- **[+]** Compared to linear classifiers like logistic regression, SVMs:
  1. Are insensitive to outliers in the data (extreme class examples)
  2. Give a robust boundary for separable classes
  3. Can handle high-dimensional data, via transformation
  4. Can find optimal α-values, with no local maxima
- **[–]** Compared to linear classifiers like logistic regression, SVMs:
  1. Are less applicable in multi-class ($c > 2$) instances
  2. Require more complex tuning, via hyper-parameter selection
  3. May require some deep thinking or experimentation in order to select the appropriate kernel functions

4

Kernel Functions for SVMs

- SVMs are often used with kernel functions that:
  1. Transform the data
  2. Compute necessary dot-products of points

$$k(\mathbf{x},\,\mathbf{z}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{z}) \qquad (\varphi : \mathbb{R}^n \to \mathbb{R}^m)$$

- The kernel is then used to compute the classification over the transformed data:

$$\mathbf{w} \cdot \mathbf{x}_i + b = \sum_j \alpha_j y_j (\varphi(\mathbf{x}_i) \cdot \varphi(\mathbf{x}_j)) + b$$

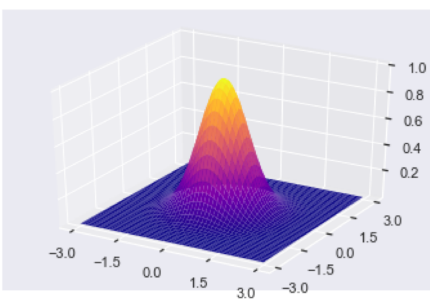
$$= \sum_j \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j) + b$$

Kernel Functions and Computation

$$\mathbf{w} \cdot \sigma(\mathbf{x}_i) + b = \sum_j \alpha_j y_j k(\mathbf{x}_i, \mathbf{x}_j) + b$$

- Some useful kernels take n-dimensional data and give results ***equivalent*** to what would happen if we:
  1. Use a function σ to transform it to m dimensions ($n << m$)
  2. Applied m weights to ***that*** data
- Storing original, smaller n-dimensional data and support vectors, and computing the kernel function when needed (RHS above), can be much more efficient than working with the larger m-dimensional data and weights (LHS above)
  - Especially true in cases where $m = \infty$ (!!)

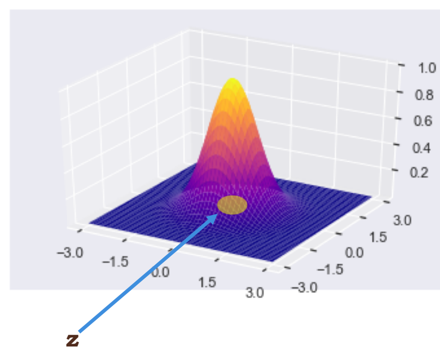
Gaussian Radial Basis Function (RBF)



- A popular kernel with many uses is the Gaussian RBF

$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

Gaussian Radial Basis Function

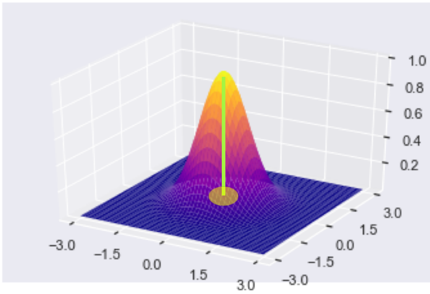


$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

- The RBF is based on a distance from a central focal point, $\mathbf{z}$
- This distance can be measured in a variety of ways, but is often Euclidean (L_2 norm):

$$||\mathbf{x} - \mathbf{z}|| = \sqrt{\sum_{i=1}^{n} (x_i - z_i)^2}$$

Gaussian Radial Basis Function



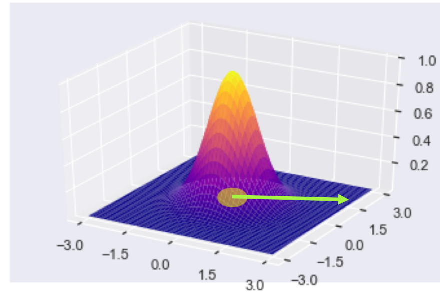
$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

$$||\mathbf{x} - \mathbf{z}|| = 0$$
$$k(\mathbf{x},\, \mathbf{z}) = e^0 = 1$$

- The value of the function is highest at point $\mathbf{z}$ itself

12

Gaussian Radial Basis Function



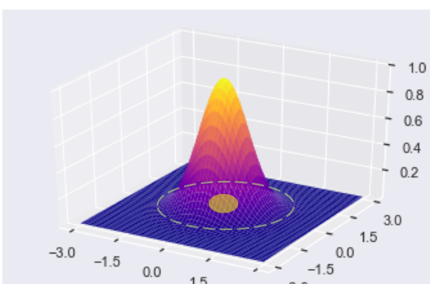
$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

$$||\mathbf{x} - \mathbf{z}|| \to \infty$$
$$k(\mathbf{x},\, \mathbf{z}) \to e^{-\infty} = 0$$

- The value drops toward 0 as we get further from $\mathbf{z}$

13

Gaussian Radial Basis Function



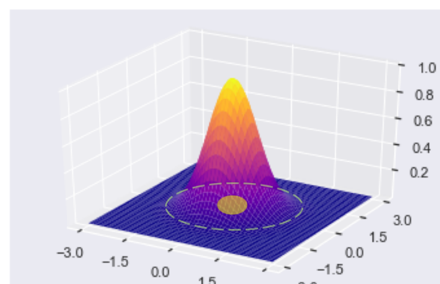
$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

Tuning Parameter

- σ controls the diameter of the non-zero area

14

Gaussian Radial Basis Function



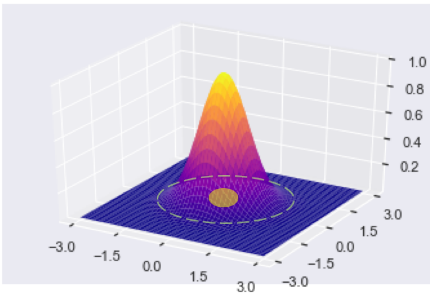
$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

$$\sigma \to \infty$$
$$k(\mathbf{x},\, \mathbf{z}) \to e^0 = 1$$

- If σ gets ***larger***, the non-0 area will become wider

15

Gaussian Radial Basis Function

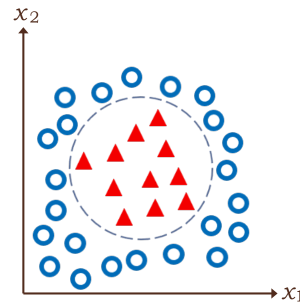


$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

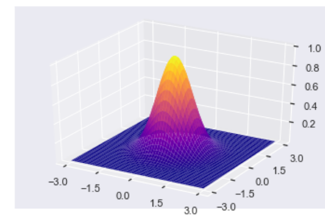
$$\sigma \to 0$$
$$k(\mathbf{x},\ \mathbf{z}) \to e^{-\infty} = 0$$

- If σ gets ***smaller***, non-0 area will become narrower

Gaussian Radial Basis Function

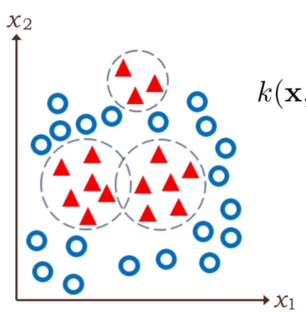


$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$



- The radius around the focal point $\mathbf{z}$ at which the function becomes 0 corresponds to the decision boundary in our data

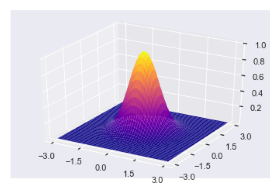
Gaussian Radial Basis Function



$$k(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) = \sum_{j=1}^{3} e^{-\frac{||\mathbf{x}-\mathbf{z}_j||^2}{2\sigma^2}}$$

- We can deal with multiple clusters in the data by using a combination of multiple RBFs

Dimensionality-Equivalence of the RBF



$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$

- Computing the RBF is pretty straightforward overall:
  1. The distance metric depends upon the dimensionality of the data:

$$- \mathbf{z}||^2 = (x_1 - z_1)^2 + (x_2 - z_2)^2 + \cdots + (x_n - z_n)^2$$

  2. Everything else is just constant-time arithmetic, and doesn't depend upon the data dimensionality
- The RBF output, however, is ***equivalent*** to an operation on data-vectors that are ***infinite*** in dimension
  - Something that ***cannot*** be computed exactly with any finite set of linear weights applied to a finite number of input features
  - Thus, the SVM will use its support vectors, its kernel function, and the associated α-parameters, and ***never*** try to translate into weights and offsets