Tufts

Class #14:
SVMs & Kernel Functions, II

Machine Learning (COMP 135): M. Allen, 04 Mar. 20

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

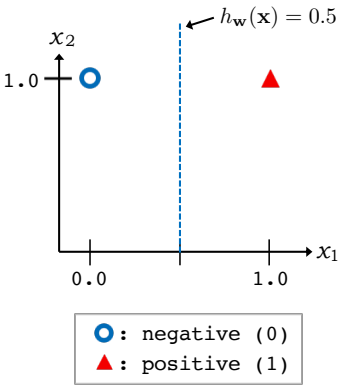
3

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

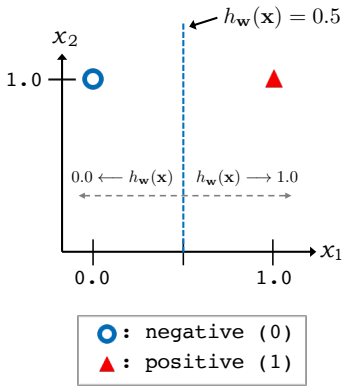
A Question: Isn't Logistic Regression ***Already*** a "Large-Margin" Classifier?



- Sometimes, our intuitions about how various models work can mislead us
- We might think the "coin-flip" threshold—where the logistic function is 0.5—lies equidistant between data at (0, 1) and (1,1)

5

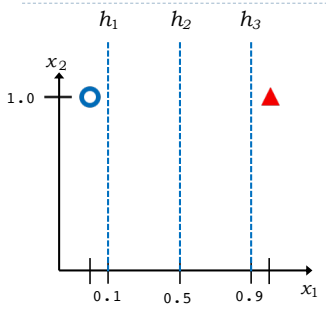
A Question: Isn't Logistic Regression ***Already*** a "Large-Margin" Classifier?



- Centered separator ***does*** work to classify the data as we expect
  - ***But***: this is ***not*** what the logistic classifier will actually choose
- ***Reason***: this separator does ***not*** minimize the logistic loss, which is what logistic regression will attempt to minimize

6

Logistic Regression ≠ "Large-Margin"



h_i	$x_1 \mid h_i(x_1, x_2) = 0.5$
h_1	0.1
h_2	0.5
h_3	0.9

- Consider 3 possible points to place the logistic regression threshold
- Now, we compute:

1. The probabilities assigned to the data-points
2. The resulting logistic loss on the data

h_i	$h_i(0,1)$	$h_i(1,1)$	$\mathcal{L}_i$
h_1	0.475	0.711	0.647
h_2	0.378	0.622	0.663
h_3	0.289	0.525	0.647

Log-loss is minimized when threshold is set ***closer*** to each data-point, ***not*** at the center between them (unlike SVM)

The function is ***symmetric***, meaning the classifier sometimes puts the separator close to one, and sometimes to another

7

Kernel Functions for SVMs

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

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

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

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

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

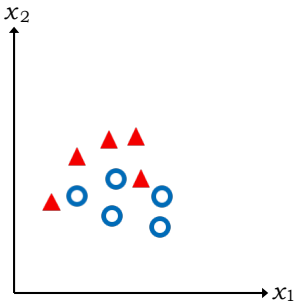
8

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 $m = \infty$ (!!)

Transforming Non-Separable Data



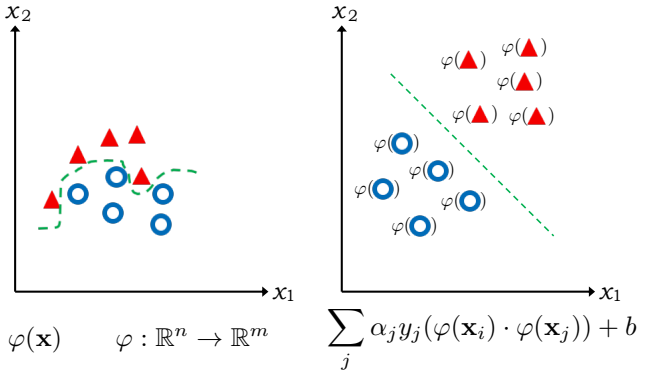
A transformation function:

$\varphi(\mathbf{x})$ $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m$

maps data-vectors to new vectors, of either the same dimensionality ($m = n$) or a different one ($m \neq n$)

- If data that is not linearly separable, we can transform it
  - We ***change*** features used to represent our data
  - Really, we ***don't care*** what the data feature are, so long as we can get classification to work

Transforming Non-Separable Data

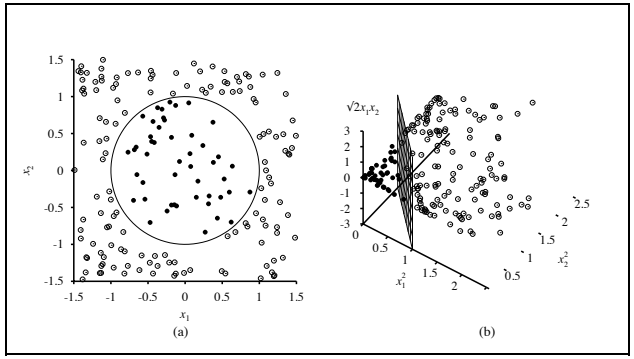


$\varphi(\mathbf{x})$ $\varphi : \mathbb{R}^n \rightarrow \mathbb{R}^m$

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

The "Kernel Trick"

Image source: Russel & Norvig, *AI: A Modern Approach* (Prentice Hal, 2010)



$$\varphi(x_1, x_2) = (x_1^2, x_2^2, \sqrt{2}x_1x_2)$$

Simplifying the Transformation Function

- We can derive a simpler (2-dimensional) equation, ***equivalent to*** the cross-product needed when doing SVM computations in the transformed (3-dimensional) space:

$$\varphi(\mathbf{x}) \cdot \varphi(\mathbf{z}) = (x_1^2,\ x_2^2,\ \sqrt{2}x_1x_2) \cdot (z_1^2,\ z_2^2,\ \sqrt{2}z_1z_2)$$

Needed

$$= x_1^2z_1^2 + x_2^2z_2^2 + \sqrt{2}x_1x_2\sqrt{2}z_1z_2$$

$$= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2$$

10 multiplications
2 additions

$$= (x_1z_1 + x_2z_2)^2$$

3 multiplications
1 addition

$$= (\mathbf{x} \cdot \mathbf{z})^2$$

Used instead

13

The Kernel Function

$$k(\mathbf{x},\ \mathbf{z}) = \varphi(\mathbf{x}) \cdot \varphi(\mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2$$

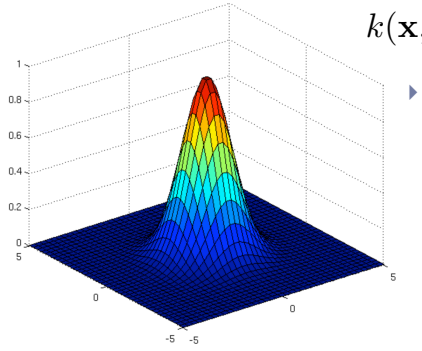
- This final function (right side) is what the SVM will actually use to compute dot-products in its equations
- This is called the kernel function
- To make SVMs really useful we look for a kernel that:

1. Separates the data usefully
2. Is relatively efficient to calculate

14

Gaussian Radial Basis Function (RBF)

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



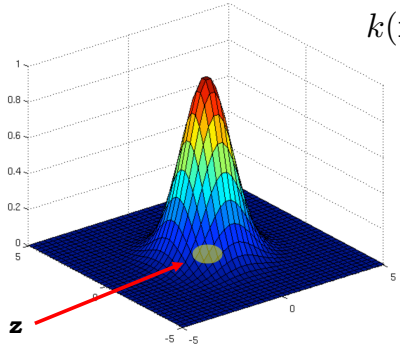
- A popular kernel with many uses is the Gaussian RBF

Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

15

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, $\boldsymbol{z}$
- This distance can be measured in a variety of ways, but is often Euclidean (L_1 norm):

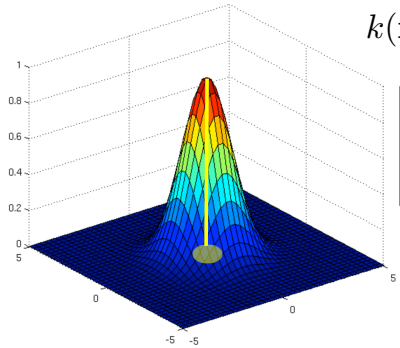
$$||\mathbf{x} - \mathbf{z}|| = \sqrt{\sum_{i=1}^{n}(x_i - z_i)^2}$$

Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

16

Gaussian Radial Basis Function

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



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

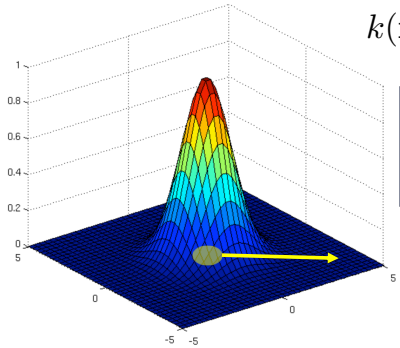
- The value of the function is highest at point $\mathbf{z}$ itself

Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

17

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 to 0 as we get further from $\mathbf{z}$

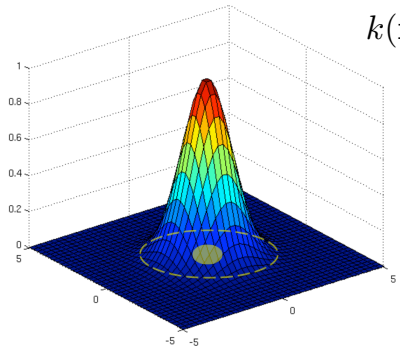
Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

18

Gaussian Radial Basis Function

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

Tuning Parameter



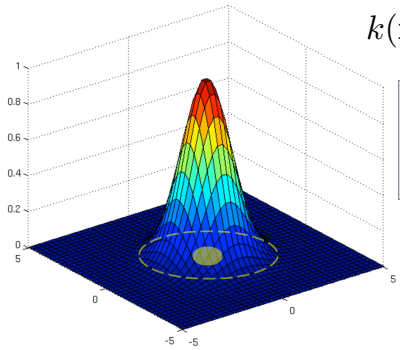
- σ controls the diameter of the non-zero area

Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

19

Gaussian Radial Basis Function

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



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

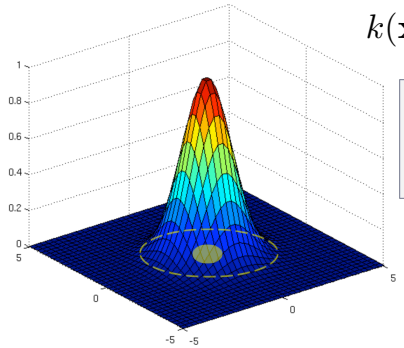
- If σ gets larger, the non-0 area will become wider

Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

20

Gaussian Radial Basis Function

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



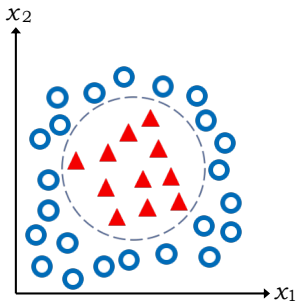
$$\sigma \to 0$$
$$k(\mathbf{x}, \mathbf{z}) \to e^{-\infty} = 0$$

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

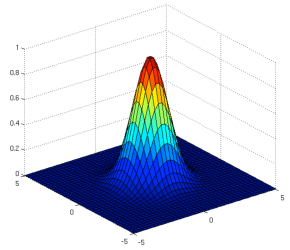
Image source: https://www.cs.toronto.edu/~duvenaud/cookbook/

21

Gaussian Radial Basis Function



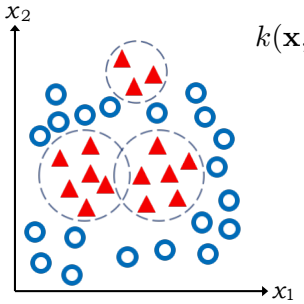
$$k(\mathbf{x}, \mathbf{z}) = e^{-\frac{||\mathbf{x}-\mathbf{z}||^2}{2\sigma^2}}$$



- The radius around the focal point **z** at which the function becomes 0 corresponds to the decision boundary in our data

22

Gaussian Radial Basis Function

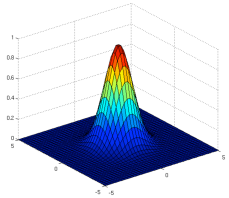


$$k(\mathbf{x}, \mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3) = \sum_{j=1}^{3} e^{-\frac{||\mathbf{x}-\mathbf{z}_j||^2}{2\sigma^2}}$$

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

23

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:

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

  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

24

This Week and Next

- **Topics**: SVMs and kernels
  - Readings linked from class website schedule page.
- **Assignments**:
  - Project 01: due Monday, 09 March, 5:00 PM
    - Feature engineering and classification for image data
  - Midterm Exam: Wednesday, 11 March
    - Practice exam distributed by end of this week
    - Review session in class, Monday, 09 March
- **Office Hours**: 237 Halligan
  - Monday, 10:30 AM – Noon
  - Tuesday, 9:00 AM – 1:00 PM
  - TA hours can be found on class website