Non-Linear Models

Weizi Li

Department of Computer Science
University of Memphis

Outline

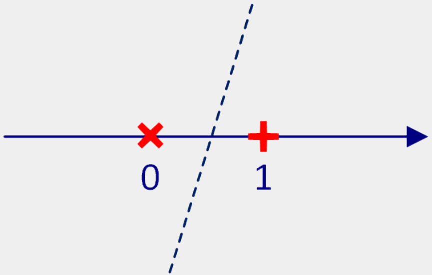
- Feature Mapping
- Kernels

Feature Mapping

- Consider the following 1-D example with 2 classes.
- We can separate the 2 classes using a linear classifier:

$$h(x; \theta, \theta_0) = sign(\theta x + \theta_0)$$

(θ and x are scalars)



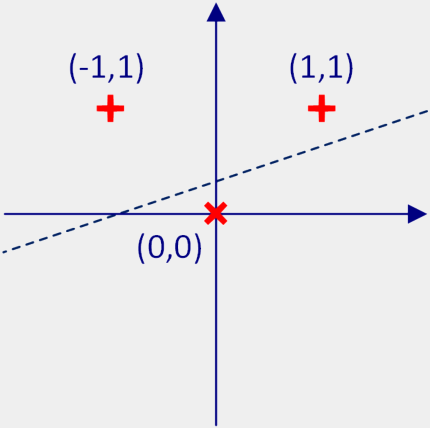
- Consider the following 1D example with 2 classes.
- We cannot separate the 2 classes using a linear classifier. Why? data are not linearly separable.

- Feature mapping: $x \rightarrow \phi(x) = [x, x^2]^T$
- Accordingly more parameter to learn: $\theta = [\theta_1, \theta_2]^T$
- Now we have:

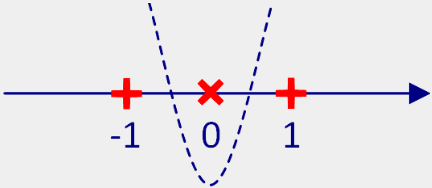
$$h(x; \theta, \theta_0) = sign(\theta\phi(x) + \theta_0) = sign(\theta_1 x + \theta_2 x^2 + \theta_0)$$

- $\theta_1 x + \theta_2 x^2 + \theta_0$ is non-linear (quadratic function, parabola)

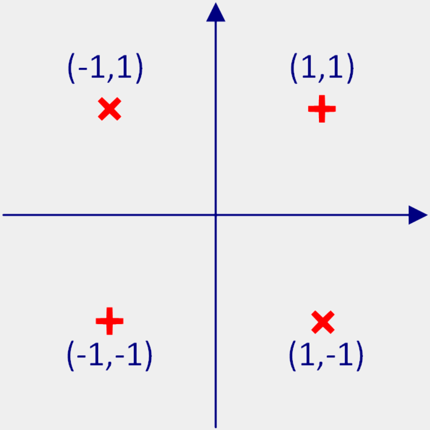
- After feature mapping, the 2 classes can now be separated using a linear classifier.



- Linear classifier in the new feature representation corresponds to non-linear classifier in the original feature representation



- Consider the following 2-D example with 2 classes.
- We cannot separate the 2 classes using a linear classifier.



- Feature mapping: $x = [x_1, x_2]^T \rightarrow \phi(x) = [x_1, x_2, x_1x_2]^T$
- $h(x; \theta, \theta_0) = sign(\theta\phi(x) + \theta_0) = sign(\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_0)$
- $\theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_0$ is non-linear
- Now we can separate the 2 classes using a linear classifier (a 2-D plane)

- We can design infinitely many "new features" and achieve more and more expressive feature representation.
- E.g., $[x_1, x_2]^T \rightarrow [x_1, x_2, x_1x_2, x_1^2, x_2^2, \ldots]^T$

- Pros: feature representation can be expressive
- Cons: requires manual design

Kernels

- Consider the following 2 feature vectors.
- $x = [x_1, x_2]^T \rightarrow \phi(x) = [x_1, x_2, x_1^2, x_2^2, \sqrt{2}x_1x_2]^T$
- $z = [z_1, z_2]^T \rightarrow \phi(z) = [z_1, z_2, z_1^2, z_2^2, \sqrt{2}z_1z_2]^T$
- $\phi(x) \cdot \phi(z) = x_1z_1 + x_2z_2 + x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2$

- Consider the following equation:

$$(x \cdot z) + (x \cdot z)^2 = x_1 z_1 + x_2 z_2 + x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2$$

- Achieved the same feature representation without explicit feature mapping
- That means we can operate high-dimensional features without manually design them

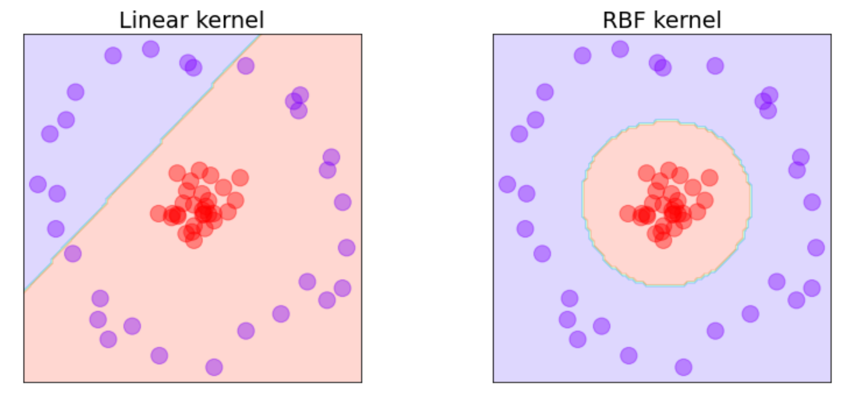
- A function that is constructed using the original feature representation
- E.g., $k(x, z) = (x \cdot z) + (x \cdot z)^2$

- Constant: $k(x, z) = 1$
- Addition: $k(x, z) = k_1(x, z) + k_2(x, z)$
- Multiplication: $k(x, z) = k_1(x, z) k_2(x, z)$
- Multiplication w/ other functions: $k(x, z) = f(x) k_1(x, z) f(z)$

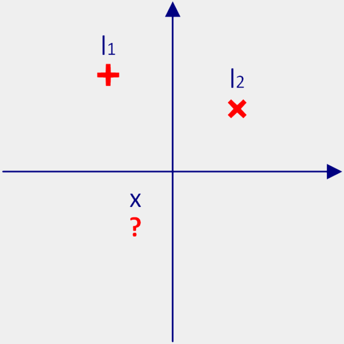
- A kernel function can be infinitely expressive while easy to compute

- RBF kernel: $k(x, z) = exp(-\gamma \|x - z\|^2)$
  - γ: spread factor
  - $\|x - z\|$: Euclidean distance
- Why infinite expressive?
  - Gaussian pdf: $f(x) = \frac{1}{\sigma\sqrt{2\pi}} exp^{-\frac{1}{2}(\frac{x-\mu}{\sigma})^2}$
  - $e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \dots$

- RBF kernel: $k(x, z) = exp(-\gamma \|x - z\|^2)$



- Treat each training example as a landmark l
- Given an unknown data point x, we can compute its distance to all landmarks: $f_i = similarity(x, l_i) = exp(-\gamma\|x - l_i\|^2), \forall i$



- If x is near a specific l_i:

$$f_i \approx exp(-\gamma \times 0) \approx 1$$

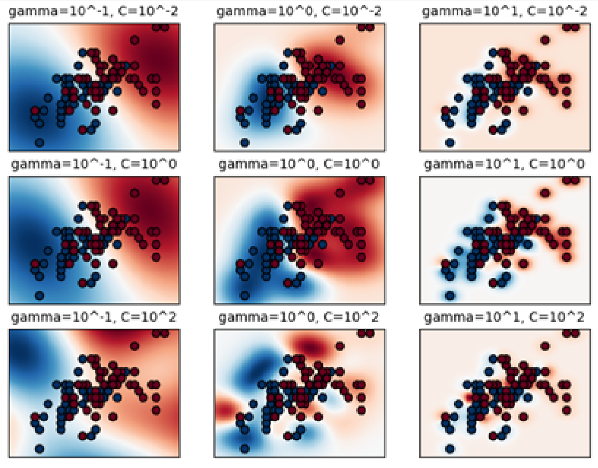
- If x is far away from a specific l_i:

$$f_i \approx exp(-\gamma \times \text{large number}) \approx 0$$

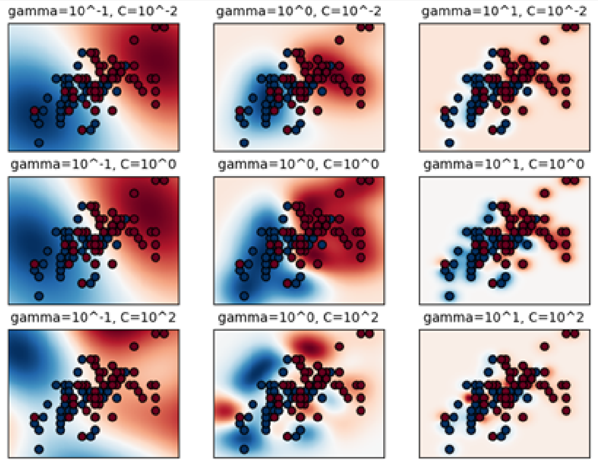
- Using all computed f_i, we can use a linear classifier $\sum_i \theta_i f_i$ to predict the label of the unknown data point x
- E.g., if $\sum_i \theta_i f_i \geq 0$, x is $+$, otherwise x is $-$

- RBF kernel is the driving force for Support Vector Machine (SVM)—one of the best ML models before deep learning era
- γ is a hyperparameter to be tuned (in many implementations of SVM, people usually tune $C = \frac{1}{\gamma}$)

- Small γ: high bias, low variance (top left, f_i spread out more)



- Large γ: low bias, high variance (bottom right, f_i spread out less)



- raw data $\rightarrow$ feature engineering (and kernel function design) via human experts $\rightarrow$ model training and tuning $\rightarrow$ outcome