RBF : Radial Basis Function (RBF)

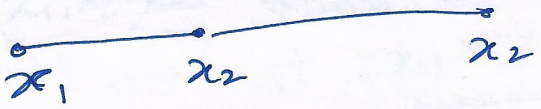
Most popular / general-purpose kernel

$$K_{RBF}(x_1, x_2) = \exp\left(\frac{-\|x_1 - x_2\|^2}{2\sigma^2}\right) = \exp\left(\frac{-d_{12}^2}{2\sigma^2}\right)$$

$$\|x_1 - x_2\|^2 = d_{12}^2$$

$\sigma \rightarrow$ hyperparameter

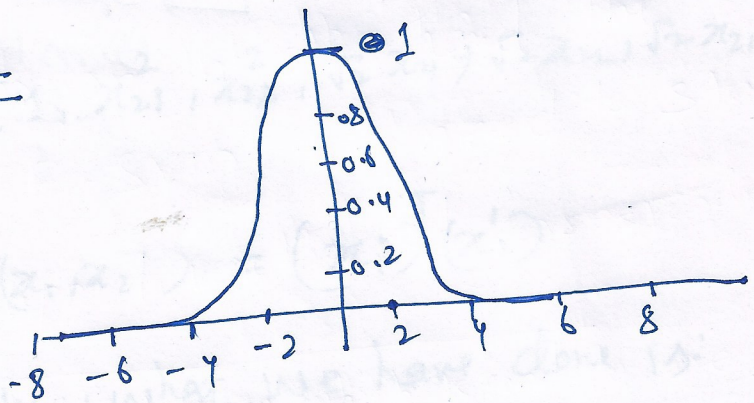
$\therefore$ (1) $d_{12} \uparrow$, $K(x_1, x_2) \downarrow$ (exponentially)



$\therefore K(x_1, x_2) > K(x$

(2) Effect of σ

$\sigma = 1$



As point going farther - farther away. We are making kernel value to zero.

Kernels are like 'similarity function'

∴ If kernel = 0 ⇒ Similarity = 0

⇒ Dis-~~similarity~~ similarity is very large.

Sim = 0 ⇒ distance = ∞ i.e. ~~very~~ very large.

∴ at $\sigma = 1$, the RBF behaves ~ Gaussian curve.

Let $\sigma = 0.1$ ⇒ $\sigma^2 = 0.01$

→ see the Plot in Google.

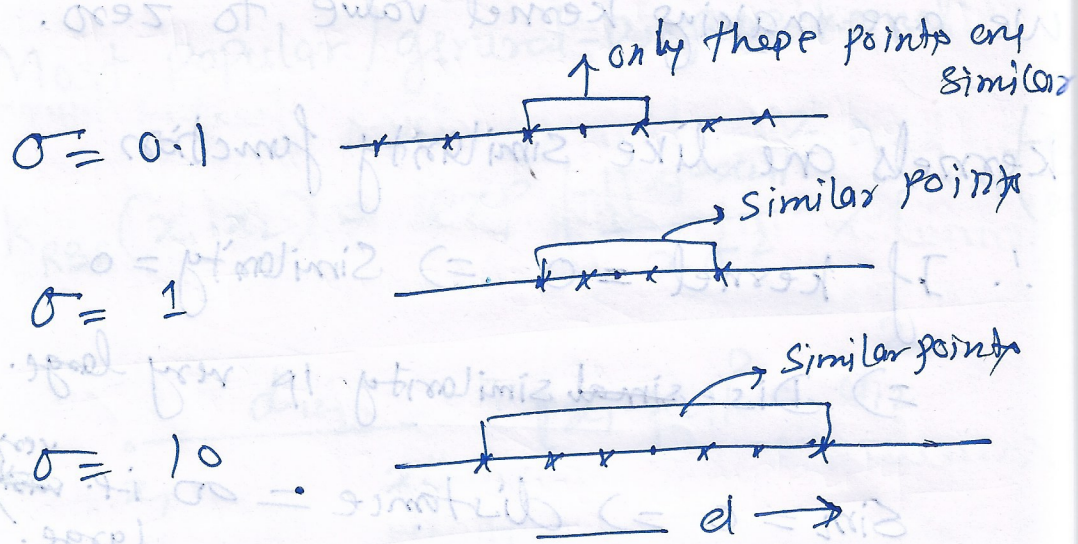
$d > 1$, $k = 0$

This distribution very-very peaked.

$\sigma = 10$ ⇒ $\sigma^2 = 100$.

$d > 10$, $k = 0$

∴ As $\sigma \uparrow$; the similarity ↑



KNN and RBF

As $\sigma \uparrow$ in RBF ⇒ $k \uparrow$ in KNN

RBF SVM ~ K-NN

The only issue with KNN, stores all the k-points ∴ ↑ space complexity ↑ d(nd)

However on the other hand the in case of RBF-SVM, we need to store just the support vectors and corresponding α_i'

If you don't know which kernel to use, use RB

RBF kernel has 2 hyperparameter.

C → Softmargin

σ → RBF

C
σ

do the grid search on both C, σ
or
Random search

* RBF kernel is very-very similar to K-NN

Domain specific kernels :

In SVM, there are lot of specialized kernel for specific task.

- String kernel ⇒ Text classification
- Genome kernel ⇒ Genome prediction problem. Bio-informatics
- Graph kernel ⇒ Graph theory problem

So, given the problem you want to solve, research for the specific kernel. If it is available use it. Otherwise :

As we know that feature transformation is the hard part of ML. You will pe- that it is partially replaced by findi the right kernel.

Train and Run time Complexity —

We can always train SVM using

- SGD
- specialized algorithm (dual problem)
  - sequential minimal optimiz (smo)

libsvm => one of the best open source libraries for train SVM

You can also use sklearn, however they are not that much great.

- The training time complexity ~ $O(n^2)$ for kernel SVM

- More optimized algorithm published in 2007 has time complexity $O(nd^2)$ if $d < n$

If n is large $\Rightarrow$ $O(n^2)$ ↑↑ very-very large.

↳ Typically, we don't use SVM if the value of n is large.

Runtime Complexity —

$$f(x_q) = \sum_{i=1}^{n} \alpha_i y_i K(x_i, x_q) + b$$

$\alpha_i = 0$ for non SVs

So runtime complexity given $x_q \longrightarrow y_q$ depends upon the # of support vectors.

If # SVs = k, then time complexity $O(kd)$

If # SVs is small, the run time complexity will be small.