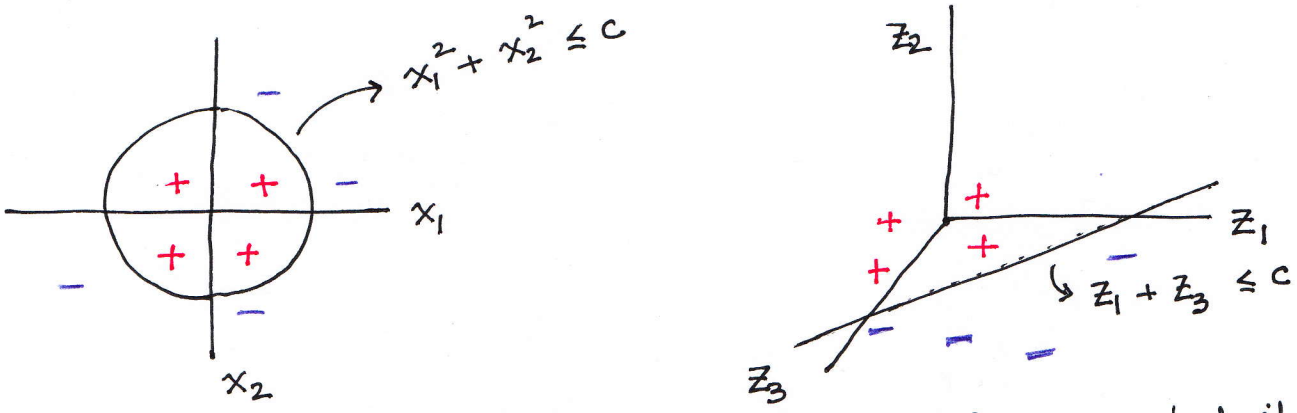
Lecture 6: Kernels



Data is not linearly separable in (x_1, x_2)-space, but if

$$z_1 = x_1^2, \quad z_2 = x_1 x_2, \quad z_3 = x_2^2$$

then data is linearly separable in (z_1, z_2, z_3) space.

Sometimes, data which is not linearly separable may be linearly separable in a different <u>feature space</u>.

For a vector x, we use $\Phi(x)$ to denote a <u>feature map</u>.

$$\Phi : \mathbb{R}^d \to \mathbb{R}^m$$

- $\mathbb{R}^d$: input is a d-dimensional vector
- $\mathbb{R}^m$: output is a m dimensional vector.

<u>Example</u>: Suppose $x = [x_1, x_2]$. $\Phi(x) = [x_1^2, x_1 x_2, x_2^2]$

So if data is linearly separable in the feature space of $\Phi(x)$, we often want to run perceptron, or SVM, or other algorithms in the feature space $\Phi(x)$.

<u>Kernel</u>: For a feature map Φ, we define the corresponding Kernel as:

$$K(x, z) = \langle \Phi(x), \Phi(z) \rangle$$

Kernel Trick:

Writing down and computing with long feature vectors is often cumbersome. However, sometimes the kernels $K(x, z)$ are much faster to compute. (examples later)

The kernel trick is to write down algorithms, such as, perceptron in terms of the dot-products, and then compute them fast (without writing down the full feature vectors). This cannot always be done, but can be done for many common algorithms, including perceptron.

Examples of Kernels:

1. $K(x, z) = (\langle x, z \rangle)^2$.

What is the feature space?

* Suppose x, z are 2-d vectors, $x = [x_1, x_2]$, $z = [z_1, z_2]$

$$K(x, z) = (x_1 z_1 + x_2 z_2)^2$$

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

If $\Phi(x) = [x_1^2, x_2^2, \sqrt{2} x_1 x_2]$, then,

$$K(x, z) = \langle \Phi(x), \Phi(z) \rangle$$

* If x, z are d-dimensional vectors, $x = [x_1, \cdots, x_d]$, $z = [z_1, \cdots, z_d]$,

then, $K(x, z) = (x_1 z_1 + \cdots + x_d z_d)^2$

$$= \sum_{i=1}^{d} x_i^2 z_i^2 + 2 \sum_{\substack{i,j \\ i \neq j}} x_i z_i x_j z_j$$

If $\Phi(x) = [\underbrace{x_1^2, \cdots, x_d^2}_{d \text{ terms}}, \underbrace{\sqrt{2} x_1 x_2, \sqrt{2} x_1 x_3, \cdots}_{\binom{d}{2} \text{ terms.}}]$

then $K(x, z) = \langle \Phi(x), \Phi(z) \rangle$

* Time taken to write down $\Phi(x) = \Theta(d^2)$
* Space taken to store $\Phi(x) = \Theta(d^2)$
* Time taken to compute $K(x, z) = (\langle x, z \rangle)^2$ is $\Theta(d)$ [$\Theta(d)$ time to compute $\langle x, z \rangle$, $O(1)$ time to square]

2 $K(x, z) = (\langle x, z \rangle + c)^2 \qquad c \geq 0.$

What is the corresponding feature map?

$$K(x,z) = \langle x,z\rangle^2 + 2c\langle x,z\rangle + c^2$$
$$= \left(\sum_{i=1}^{d} x_i z_i\right)^2 + 2c\sum_{i=1}^{d} x_i z_i + c^2$$
$$= \sum_{i=1}^{d} x_i^2 z_i^2 + 2\sum_{i\neq j} x_i x_j z_i z_j + 2c\sum_{i=1}^{d} x_i z_i + c^2$$

If $\Phi(x) = [\underbrace{x_1^2, .., x_d^2}_{d \text{ terms}}, \underbrace{\sqrt{2}x_1x_2, \sqrt{2}x_1x_3, \ldots}_{\binom{d}{2} \text{ terms}}]$ is the feature map from example 1, and if $\Psi(x) = [\Phi(x), \sqrt{2c}\,x, c]$, then $K(x, z) = \langle \Psi(x), \Psi(z) \rangle$

Note: Length of $\Phi(x)$ is again $\Theta(d^2)$, but $K(x, z)$ takes $O(d)$ time to compute.

3 $K(x, z) = (\langle x, z \rangle + c)^k$, $c \geq 0$, k = integer (Polynomial Kernel)

Corresponding feature map has $d^{O(k)}$ terms, but $K(x, z)$ can be computed in $O(d+k)$ time

($\Theta(d)$ to compute $\langle x, z \rangle$, $O(1)$ to add c, $O(k)$ to raise to k-th power)

4 $K(x, z) = e^{-\|x-z\|^2/c^2}$ (called the Gaussian Kernel)

Corresponds to an <u>infinite dimensional feature map</u>.

5 String Kernels.

Let s and t be strings over an alphabet Σ, p = an integer > 0.

$K(s,t)$ = # of common substrings of length p in s and t.

eg. for $p=1$, $K(s,t)$ = # common letters in s and t.

so, if s = "asdf", t = "gpsd", then $K(s,t) = 2$ for $p=1$.

What is the corresponding feature map?

$\Phi(s)$ has a coordinate for every substring of length p over Σ. $\Phi_u(s) = 1$ if u is a substring of s,

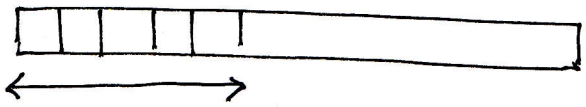
$= 0$, o/w.

Then; $K(s,t) = \sum_{u \in \Sigma^p} \Phi_u(s)\,\Phi_u(t) = \langle \Phi(s), \Phi(t) \rangle$

Length of $\Phi(s) = |\Sigma|^p$. (Space to store $\Phi(s)$, time to write it down)

But $K(s,t)$ can be computed in $O(stp)$ time, which is usually much less for large p.

How to compute $K(s,t)$?



- For $i = 1, .., |s| - p + 1$
  - Let $v = s[i]\, s[i+1] \ldots s[i+p-1]$
  - Check if v is a substring of t. (Be careful of duplicates and double counting. - also check if v appeared earlier in s)

- # vs to check : $O(|s|)$
- For each v, need to check it against $O(|t|) + O(|s|)$ possible substrings, each of length p
- Total time = $O(p|s|(|s|+|t|))$ (Can be even less with a cleverer algorithm)

6 Graph Kernels.

Let G_1, G_2 be two graphs, $p \geq 2$ an integer.

$\Phi(G)$ has a coordinate for every kind of subgraph of size p. eg. for $p=3$, possible subgraphs are:

$K(G_1, G_2) = \langle \Phi(G_1), \Phi(G_2) \rangle$

7 Time Series Kernels, etc.

Kernel Properties:

Not all functions $K(x, z)$ are kernels.

Conditions for a function to be a kernel:

1. Symmetry: For all x, z, $K(x, z) = K(z, x)$
2. Positive Semi Definiteness: For a set of points $x^1, .., x^m$, define kernel matrix as:

$$K_{m \times m}, \quad K_{ij} = K(x^i, x^j)$$

For all $x^1, .., x^m$, the kernel matrix is PSD.

These are necessary and sufficient conditions.

How to show a function $K(x, z)$ is a kernel?

1. Either find a feature map ϕ s.t. $K(x, z) = \langle \phi(x), \phi(z) \rangle$
2. Or show conditions (1) and (2) hold (usually harder).

Example: Let $K(x, z) = \langle x, z \rangle^2$.

- (1) holds as $K(x, z) = K(z, x)$
- Let t : $m \times 1$ vector $= [t_1, .., t_m]^T$, $x^1 \ldots x^m$ any m vectors.

$$t^T K t = \sum_{i=1}^{m} \sum_{j=1}^{m} t_i t_j K_{ij} = \sum_{i,j} t_i t_j \langle x^i, x^j \rangle^2$$

$$\downarrow$$

$$\text{Kernel matrix } K = \sum_{i,j} t_i t_j \left(\sum_{\ell=1}^{d} x_\ell^i x_\ell^j \right)^2 = \sum_{i,j=1}^{m} t_i t_j \sum_{\ell,\ell'=1}^{d} x_\ell^i x_\ell^j x_{\ell'}^i x_{\ell'}^j$$

For each pair (ℓ, ℓ') we can take the summation over this pair out; for a fixed (ℓ, ℓ') we sum over all (i, j) pairs.

So we get:

$$\sum_{l,l'=1}^{d} \sum_{i=1}^{m} \sum_{j=1}^{m} x_l^i x_l^j x_{l'}^i x_{l'}^j t_i t_j$$

$$= \sum_{l,l'=1}^{d} \left(\sum_{i=1}^{m} x_l^i x_{l'}^i t_i \right) \left(\sum_{j=1}^{m} x_l^j x_{l'}^j t_j \right)$$

[The two parenthesized terms are the same, with different indices

$$= \sum_{l,l'=1}^{d} \left(\sum_{i=1}^{m} x_l^i x_{l'}^i t_i \right)^2 \geq 0$$

How to show a function $K(x,z)$ is NOT a kernel?

Show a counterexample to (1) or (2).

Example: $K(x,z) = -\langle x,z \rangle$ is NOT a kernel. Why?

Pick x = any vector. Kernel matrix of x is a single scalar:

($x \neq 0$)

$$K = [-\langle x,x \rangle] = [-\|x\|^2]$$

K is not PSD. Why? For any 1×1 t, s.t. $t \neq 0$,

$$t^T K t = -t^2 \|x\|^2 < 0$$

So Condition (2) is violated.

Tip: If you suspect $K(x,z)$ is not a kernel, try to find a small (1×1 or 2×2) matrix that is a counterexample to (2).

How to Kernelize Perceptron:

1. Initially, $w_1 = 0$.
2. For $t = 1, 2, 3, \ldots$

   If $y_t \langle w_t, \Phi(x_t) \rangle \leq 0$ then

   $w_{t+1} = w_t + y_t \Phi(x_t)$

   Else

   $w_{t+1} = w_t$

Perceptron in the Φ-space. (if we were to maintain w_t directly)

- How to avoid maintaining w_t explicitly?

* Observe that w_t is a linear combination of $\Phi(x_i)$'s where a mistake has been made in the past.
* We just store these x_i's and the corresponding y_i's.
* Also observe: we only need to take dot products of the form $\langle w_t, \Phi(x_t) \rangle$ during training, and $\langle w_t, \Phi(x) \rangle$ for test examples x during testing.

If $w_t = y_{i1} \Phi(x_{i1}) + y_{i2} \Phi(x_{i2}) + \cdots + y_{ik} \Phi(x_{ik})$

(so we have made mistakes on $(x_{i1}, y_{i1}), \ldots, (x_{ik}, y_{ik})$ during training)

then, $\langle w_t, \Phi(x) \rangle$

$= \langle y_{i1} \Phi(x_{i1}) + \cdots + y_{ik} \Phi(x_{ik}), \Phi(x) \rangle$

$= y_{i1} \langle \Phi(x_{i1}), \Phi(x) \rangle + y_{i2} \langle \Phi(x_{i2}), \Phi(x) \rangle + \cdots$

$\quad + y_{ik} \langle \Phi(x_{ik}), \Phi(x) \rangle$

$= y_{i1} K(x_{i1}, x) + \cdots + y_{ik} K(x_{ik}, x)$

Kernels to Distances:

Given a kernel function $K(x, z)$, define a distance function:

$$D_K(x, z) = \sqrt{K(x, x) + K(z, z) - 2K(x, z)}$$

If $K(x, z) = \langle \Phi(x), \Phi(z) \rangle$ then:

$$\begin{aligned} D_K^2(x, z) &= \langle \Phi(x), \Phi(x) \rangle + \langle \Phi(z), \Phi(z) \rangle - 2 \langle \Phi(x), \Phi(z) \rangle \\ &= \langle \Phi(x) - \Phi(z), \Phi(x) - \Phi(z) \rangle \\ &= \| \Phi(x) - \Phi(z) \|^2 \end{aligned}$$

$D_K(x, z)$ can also be computed easily, if $K(x, z)$ can be computed easily.

Any algorithm which is based on dot products and Euclidean distances can be kernelized. e.g. k-NN can be kernelized if you $D_K(x, z)$ instead of the usual Euclidean distance.