Feature Mapping

- Consider the following mapping ϕ for an example $\mathbf{x} = \{x_1, \ldots, x_D\}$

$$\phi : \mathbf{x} \rightarrow \{x_1^2, x_2^2, \ldots, x_D^2, , x_1 x_2, x_1 x_2, \ldots, x_1 x_D, \ldots \ldots, x_{D-1} x_D\}$$

- It's an example of a quadratic mapping
  - Each new feature uses a pair of the original features
- Problem: Mapping usually leads to the number of features blow up!
  - Computing the mapping itself can be inefficient in such cases
  - Moreover, *using* the mapped representation could be inefficient too
    - e.g., imagine computing the similarity between two examples: $\phi(\mathbf{x})^\top \phi(\mathbf{z})$
- Thankfully, Kernels help us avoid both these issues!
  - The mapping doesn't have to be explicitly computed
  - Computations with the mapped features remain efficient

Kernels as High Dimensional Feature Mapping

- Consider two examples $\mathbf{x} = \{x_1, x_2\}$ and $\mathbf{z} = \{z_1, z_2\}$
- Let's assume we are given a function k (kernel) that takes as inputs $\mathbf{x}$ and $\mathbf{z}$

$$
\begin{aligned}
k(\mathbf{x}, \mathbf{z}) &= (\mathbf{x}^\top \mathbf{z})^2 \\
&= (x_1 z_1 + x_2 z_2)^2 \\
&= x_1^2 z_1^2 + x_2^2 z_2^2 + 2 x_1 x_2 z_1 z_2 \\
&= (x_1^2, \sqrt{2} x_1 x_2, x_2^2)^\top (z_1^2, \sqrt{2} z_1 z_2, z_2^2) \\
&= \phi(\mathbf{x})^\top \phi(\mathbf{z})
\end{aligned}
$$

- The above k implicitly defines a mapping ϕ to a higher dimensional space

$$\phi(\mathbf{x}) = \{x_1^2, \sqrt{2} x_1 x_2, x_2^2\}$$

- Note that we didn't have to define/compute this mapping
- Simply defining the kernel a certain way gives a higher dim. mapping ϕ
- Moreover the kernel $k(\mathbf{x}, \mathbf{z})$ also computes the dot product $\phi(\mathbf{x})^\top \phi(\mathbf{z})$
  - $\phi(\mathbf{x})^\top \phi(\mathbf{z})$ would otherwise be much more expensive to compute explicitly
- All kernel functions have these properties

Kernels: Formally Defined

- Recall: Each kernel k has an associated feature mapping ϕ
- ϕ takes input $\mathbf{x} \in \mathcal{X}$ (input space) and maps it to $\mathcal{F}$ ("feature space")
- Kernel $k(\mathbf{x}, \mathbf{z})$ takes two inputs and gives their similarity in $\mathcal{F}$ space

$$
\begin{aligned}
\phi &: \mathcal{X} \to \mathcal{F} \\
k &: \mathcal{X} \times \mathcal{X} \to \mathbb{R}, \quad k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})
\end{aligned}
$$

- $\mathcal{F}$ needs to be a *vector space* with a *dot product* defined on it
  - Also called a *Hilbert Space*
- Can just *any* function be used as a kernel function?
  - No. It must satisfy **Mercer's Condition**

Mercer's Condition

- For k to be a kernel function
  - There must exist a Hilbert Space $\mathcal{F}$ for which k defines a dot product
  - The above is true if K is a positive definite function

$$\int d\mathbf{x} \int d\mathbf{z} f(\mathbf{x}) k(\mathbf{x}, \mathbf{z}) f(\mathbf{z}) > 0 \quad (\forall f \in L_2)$$

  - This is Mercer's Condition
- Let k_1, k_2 be two kernel functions then the following are as well:
  - $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) + k_2(\mathbf{x}, \mathbf{z})$: direct sum
  - $k(\mathbf{x}, \mathbf{z}) = \alpha k_1(\mathbf{x}, \mathbf{z})$: scalar product
  - $k(\mathbf{x}, \mathbf{z}) = k_1(\mathbf{x}, \mathbf{z}) k_2(\mathbf{x}, \mathbf{z})$: direct product
  - Kernels can also be constructed by composing these rules

The Kernel Matrix

- The kernel function k also defines the Kernel Matrix $\mathbf{K}$ over the data
- Given N examples $\{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$, the (i,j)-th entry of $\mathbf{K}$ is defined as:

$$K_{ij} = k(\mathbf{x}_i, \mathbf{x}_j) = \phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j)$$

- K_{ij}: Similarity between the i-th and j-th example in the feature space $\mathcal{F}$
- $\mathbf{K}$: $N \times N$ matrix of pairwise similarities between examples in $\mathcal{F}$ space
- $\mathbf{K}$ is a symmetric matrix
- $\mathbf{K}$ is a positive definite matrix (except for a few exceptions)
- For a P.D. matrix: $\mathbf{z}^\top \mathbf{K}\mathbf{z} > 0, \quad \forall \mathbf{z} \in \mathbb{R}^N$ (also, all eigenvalues positive)
- The Kernel Matrix $\mathbf{K}$ is also known as the Gram Matrix

Some Examples of Kernels

The following are the most popular kernels for real-valued vector inputs

- Linear (trivial) Kernel:

$$k(\mathbf{x}, \mathbf{z}) = \mathbf{x}^\top \mathbf{z} \text{ (mapping function } \phi \text{ is identity - no mapping)}$$

- Quadratic Kernel:

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^2 \quad \text{or} \quad (1 + \mathbf{x}^\top \mathbf{z})^2$$

- Polynomial Kernel (of degree d):

$$k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z})^d \quad \text{or} \quad (1 + \mathbf{x}^\top \mathbf{z})^d$$

- Radial Basis Function (RBF) Kernel:

$$k(\mathbf{x}, \mathbf{z}) = \exp[-\gamma ||\mathbf{x} - \mathbf{z}||^2]$$

  - γ is a hyperparameter (also called the kernel bandwidth)
  - The RBF kernel corresponds to an infinite dimensional feature space $\mathcal{F}$ (i.e., you can't actually write down the vector $\phi(\mathbf{x})$)

Note: Kernel hyperparameters (e.g., d, γ) chosen via cross-validation

Using Kernels

- Kernels can turn a linear model into a nonlinear one
- Recall: Kernel $k(\mathbf{x}, \mathbf{z})$ represents a dot product in some high dimensional feature space $\mathcal{F}$
- Any learning algorithm in which examples only appear as dot products $(\mathbf{x}_i^\top \mathbf{x}_j)$ can be kernelized (i.e., non-linearlized)
    - .. by replacing the $\mathbf{x}_i^\top \mathbf{x}_j$ terms by $\phi(\mathbf{x}_i)^\top \phi(\mathbf{x}_j) = k(\mathbf{x}_i, \mathbf{x}_j)$
- Most learning algorithms are like that
    - Perceptron, SVM, linear regression, etc.
    - Many of the unsupervised learning algorithms too can be kernelized (e.g., K-means clustering, Principal Component Analysis, etc.)