7.7 Kernel Methods

[Shaw-Taylor & Cristiani, 2004]

Linear Regression

$$\mathbf{x} = (x_1, x_2, \ldots, x_I)$$

$$D = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}_{n=1}^{N}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{I} w_i x_i$$

$$\mathbf{xw} = \mathbf{y}$$

$$\varepsilon = (y - g(\mathbf{x}))$$

$$\boldsymbol{\varepsilon} = \mathbf{y} - \mathbf{xw}$$

$$\begin{pmatrix} \overbrace{\boxed{x_1^{(1)} \quad x_2^{(1)} \quad \cdots \quad x_n^{(1)}}}^{\mathbf{x}^{(1)}} \\ x_1^{(2)} \quad x_2^{(2)} \quad \cdots \quad x_n^{(2)} \\ \vdots \quad \vdots \quad \quad \vdots \\ \underbrace{\boxed{x_1^{(N)} \quad x_2^{(N)} \quad \cdots \quad x_n^{(N)}}}_{\mathbf{x}^{(I)}} \end{pmatrix} \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_n \end{pmatrix} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_N \end{pmatrix}$$

$$L(\mathbf{w}, D) = \|\mathbf{e}\|^2 = (\mathbf{y} - \mathbf{xw})'(\mathbf{y} - \mathbf{xw}) = \mathbf{y}^2 - \mathbf{y}'\mathbf{xw} - \mathbf{w}'\mathbf{x}'\mathbf{y} + \mathbf{w}'\mathbf{x}'\mathbf{xw}$$

$$\frac{\partial L(\mathbf{w}, D)}{\partial \mathbf{w}} = -2\mathbf{x}'\mathbf{y} + 2\mathbf{x}'\mathbf{xw} = 0$$

$\mathbf{x}'\mathbf{xw} = \mathbf{x}'\mathbf{y}$: normal equations

$$\mathbf{w} = (\mathbf{x}'\mathbf{x})^{-1}\mathbf{x}'\mathbf{y}$$

$$= (\mathbf{x}'\mathbf{x})(\mathbf{x}'\mathbf{x})^{-2}\mathbf{x}'\mathbf{y} = \mathbf{x}'\alpha$$

$$\mathbf{w} = \sum_{n=1}^{N} \alpha_n \mathbf{x}_n$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$

$$= \left\langle \sum_{n=1}^{N} \alpha_n \mathbf{x}_n, \mathbf{x} \right\rangle$$

$$= \sum_{n=1}^{N} \alpha_n \langle \mathbf{x}_n, \mathbf{x} \rangle \qquad \text{(sum is over the } N \text{ examples)}$$

$$= \sum_{n=1}^{N} \alpha_n k \langle \mathbf{x}_n, \mathbf{x} \rangle$$

Note: $g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{w}'\mathbf{x} = \sum_{i=1}^{I} w_i x_i$ (sum is over the I weights)

Ridge Regression

$$\min_{\mathbf{w}} L_\lambda(\mathbf{w}, D) = \min_{\mathbf{w}} \lambda \|\mathbf{w}\|^2 + \sum_{n=1}^{N} (y_n - g(\mathbf{x}_n))^2$$

$$\frac{\partial L_\lambda(\mathbf{w}, D)}{\partial \mathbf{w}} = 2\lambda \mathbf{w} - 2\mathbf{x}'\mathbf{y} + 2\mathbf{x}'\mathbf{x}\mathbf{w} = 0$$

$$\mathbf{x}'\mathbf{x}\mathbf{w} + \lambda \mathbf{w} = (\mathbf{x}'\mathbf{x} + \lambda \mathbf{I}_I)\mathbf{w} = \mathbf{x}'\mathbf{y} \qquad (*)$$

$$\mathbf{w} = (\mathbf{x}'\mathbf{x} + \lambda \mathbf{I}_I)^{-1}\mathbf{x}'\mathbf{y}$$

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle = \mathbf{y}'\mathbf{x}(\mathbf{x}'\mathbf{x} + \lambda \mathbf{I}_I)^{-1}\mathbf{x}$$

From (*)

$$\mathbf{w} = \lambda^{-1}\mathbf{x}'(\mathbf{y} - \mathbf{x}\mathbf{w}) = \mathbf{x}'\boldsymbol{\alpha}$$

$$= \sum_{n=1}^{N} \alpha_n \mathbf{x}_n \quad \text{with } \boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{x}\mathbf{w})$$

$$\lambda \mathbf{w} = \mathbf{x}'\mathbf{y} - \mathbf{x}'\mathbf{x}\mathbf{w}$$
$$= \mathbf{x}'(\mathbf{y} - \mathbf{x}\mathbf{w})$$
$$\mathbf{w} = \lambda^{-1}\mathbf{x}'(\mathbf{y} - \mathbf{x}\mathbf{w})$$

$$\boldsymbol{\alpha} = \lambda^{-1}(\mathbf{y} - \mathbf{x}\mathbf{w})$$
$$\Rightarrow \lambda \boldsymbol{\alpha} = (\mathbf{y} - \mathbf{x}\mathbf{x}'\boldsymbol{\alpha})$$
$$\Rightarrow (\mathbf{x}\mathbf{x}' + \lambda \mathbf{I}_N)\boldsymbol{\alpha} = \mathbf{y}$$
$$\Rightarrow \boldsymbol{\alpha} = (\mathbf{G} + \lambda \mathbf{I}_N)^{-1}\mathbf{y} \qquad \text{where } \mathbf{G} = \mathbf{x}\mathbf{x}' \text{ or } \mathbf{G}_{ij} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

The resulting prediction function

$$g(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$$
$$= \left\langle \sum_{n=1}^{N} \alpha_n \mathbf{x}_n, \mathbf{x} \right\rangle$$
$$= \sum_{n=1}^{N} \alpha_n \langle \mathbf{x}_n, \mathbf{x} \rangle$$
$$= \mathbf{y}'(\mathbf{G} + \lambda \mathbf{I}_N)^{-1}\mathbf{k} \qquad \text{with } k_n = \langle \mathbf{x}_n, \mathbf{x} \rangle$$

Kernel Functions: Definition

$$\phi : \mathbf{x} \in \mathbb{R}^n \mapsto \phi(\mathbf{x}) \in F \subseteq \mathbb{R}^N$$

$$\hat{D} = \{(\phi(\mathbf{x}_1), y_1), \ldots, (\phi(\mathbf{x}_n), y_n)\}$$

$$f((\mathbf{x}, y)) = |y - g(\mathbf{x})| = |y - \langle \mathbf{w}, \phi(\mathbf{x}) \rangle| = |\xi|$$

$$g(\mathbf{x}) = \mathbf{y}'(\mathbf{G} + \lambda \mathbf{I}_l)^{-1}\mathbf{k}$$

$$\mathbf{G}_{ij} = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}_j) \rangle$$ Gram matrix or kernel matrix

$$k_i = \langle \phi(\mathbf{x}_i), \phi(\mathbf{x}) \rangle$$

Definition: A kernel is a function k that for all $\mathbf{x}, \mathbf{z} \in X$ satisfies

$$k(\mathbf{x}, \mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle,$$

where ϕ is a mapping from X to an (inner product) feature space F, i.e. $\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) \in F$. In other words, a function that returns the inner product between the images of two inputs in some feature space is called a kernel function. Thus the kernel functions can be used to measure the (dis)similarities of data points in a feature space with feature map ϕ without explicitly computing their coordinates.

Example 2.9: $x \in \mathbb{R}^2$: 2-dimensional input space

$$\phi : x = (x_1, x_2) \mapsto \phi(x) = (x_1^2, x_2^2, \sqrt{2}x_1x_2) \in F = \mathbb{R}^3$$ (F: feature space)

$g(\mathbf{x}) = w_{11}x_1^2 + w_{22}x_2^2 + w_{12}\sqrt{2}x_1x_2$: hypothesis space of linear functions in F

$$\begin{aligned} \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \langle (x_1^2, x_2^2, \sqrt{2}x_1x_2), (z_1^2, z_2^2, \sqrt{2}z_1z_2) \rangle \\ &= x_1^2z_1^2 + x_2^2z_2^2 + 2x_1x_2z_1z_2 \\ &= (x_1z_1 + x_2z_2)^2 = \langle \mathbf{x}, \mathbf{z} \rangle^2 \end{aligned}$$

$\therefore\ k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ is a kernel function with F its corresponding feature space. The same kernel computes the inner products corresponding to the 4-D feature map.

Note:: $\phi : x = (x_1, x_2) \mapsto \phi(x) = (x_1^2, x_2^2, x_1x_2, x_2x_1) \in F = \mathbb{R}^4$, showing that the feature space is not uniquely determined by the kernel function.

Example 2.10: $k(\mathbf{x}, \mathbf{z}) = \langle \mathbf{x}, \mathbf{z} \rangle^2$ is a kernel function for the feature map

$$\phi : x \mapsto \phi(x) = (x_ix_j)_{i,j=1}^n \in F = \mathbb{R}^{n^2},$$

since we have that

$$\begin{aligned}\langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle &= \left\langle (x_i x_j)_{i,j=1}^n, (z_i z_j)_{i,j=1}^n \right\rangle \\ &= \sum_{i,j=1}^n x_i x_j z_i z_j = \sum_{i=1}^n x_i z_i \sum_{j=1}^n x_j z_j \\ &= \langle \mathbf{x}, \mathbf{z} \rangle^2\end{aligned}$$

Feature Maps and Kernel Functions

$$k(\mathbf{x}_t, \mathbf{x}_{t'}) = \phi(\mathbf{x}_t)^T \phi(\mathbf{x}_{t'}) = \langle \phi(\mathbf{x}_t), \phi(\mathbf{x}_{t'}) \rangle$$

$\phi(\mathbf{x}) = k(\cdot, \mathbf{x})$: like a delta function

$\langle f, k(\cdot, \mathbf{x}) \rangle = \sum_{i=1}^N \alpha_i k(\mathbf{x}_i, \mathbf{x}) = f(\mathbf{x})$: reproducing property of the kernel

$$\langle \phi(\mathbf{x}_t), \phi(\mathbf{x}_{t'}) \rangle = \langle k(\cdot, \mathbf{x}_t), k(\cdot, \mathbf{x}_{t'}) \rangle = k(\mathbf{x}_t, \mathbf{x}_{t'})$$

$$\phi(\mathbf{x}) = [1; \mathbf{x}; \mathbf{x}^2; \ldots, \mathbf{x}^p]$$

Kinds of Kernels

- Gaussian kernels

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

$$\phi : \mathbf{x} \mapsto \phi(\mathbf{x}) = k(\mathbf{x}, \cdot) = \exp\left(-\frac{\| \mathbf{x} - \cdot \|^2}{2\sigma^2} \right)$$

$$\left\langle \sum_{i=1}^l \alpha_i k(\mathbf{x}_i, \cdot), \sum_{j=1}^l \beta_i k(\mathbf{x}_j, \cdot) \right\rangle = \sum_{i=1}^l \sum_{j=1}^l \alpha_i \beta_j k(\mathbf{x}_i, \mathbf{x}_j)$$

- Polynomial kernels

$$
\begin{aligned}
k(\mathbf{x},\mathbf{z}) &= (<\mathbf{x},\mathbf{z}>+R)^d \\
&= \sum_{s=0}^{d} \binom{d}{s} R^{d-s} \left\langle \mathbf{x},\mathbf{z} \right\rangle^s \\
&= \sum_{s=0}^{d} a_s \hat{k}_s(\mathbf{x},\mathbf{z})
\end{aligned}
$$

where $a_s = \binom{d}{s} R^{d-s}$ and $\hat{k}_s(\mathbf{x},\mathbf{z}) = \left\langle \mathbf{x},\mathbf{z} \right\rangle^s$

$\phi_{\mathbf{i}}(\mathbf{x}) = k_{\mathbf{i}}(\mathbf{x},\cdot) = \mathbf{x}^{\mathbf{i}} = x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}$

$\mathbf{i} = (i_1,\ldots,i_n) \in \mathbb{N}^n$ satisfies $\sum_{j=1}^{n} i_j = s$.

The features corresponding to $k_d(\mathbf{x},\mathbf{z})$ are all functions of the form $\phi_i(\mathbf{x})$ for $\mathbf{i}$ satisfying

$$\sum_{j=1}^{n} i_j \le d .$$

We shall see other examples of kernels in the next section. These include all-subsets kernels and ANOVA kernels along with their randomized versions, which can naturally be realized in hypernetworks.

7.8 Hypernetworks as Generalized Kernel Machines

[Shaw-Taylor & Cristiani, 2004, Chap. 9]

In this section we show that the hypernetworks are natural representations for kernel machines. To do this we review the kernel types presented in [Shaw-Taylor & Cristiani, 2004, Chap. 9] and demonstrate how they can be realized in the hypernetworks. The basic idea underlying all these approaches is that a hyperedge $E_i \subset X$ can be defined to represent an arbitrary embedding map $\phi_{E_i}(\mathbf{x})$. We shall focus on the class of subset kernels where the various subsets (combinations) of a given set X of variables are considered. The subset kernels match perfectly with the concept of hyperedges as subsets of the vertex set. Also, from the computational point of view, the kernel function, i.e., the inner product $<\phi(\mathbf{x}),\phi(\mathbf{z})>$ of two images in the feature space can be very naturally computed by matching the two corresponding hyperedges.

Varioius Kernel Types and Their Hypernetwork Representations

- **All-subsets kernels**

Consider a space with a feature ϕ_A for each subset $A \subseteq \{1,2,...,n\}$ of the input features, including the empty subset. Equivalently we can represent them as features

$$\phi_{\mathbf{i}}(\mathbf{x}) = x_1^{i_1} x_2^{i_2} ... x_n^{i_n},$$

with the restriction that $\mathbf{i} = (i_1,...,i_n) \in \{0,1\}^n$. The feature is given by multiplying together the input features for all the elements of the subset

$$\phi_A(\mathbf{x}) = k_A(\mathbf{x},\cdot) = \prod_{i \in A} x_i.$$

This generates all monomial features for all combinations of up to *n* different indicies. Note that, unlike in the polynomial case, each factor in the monomial has degree 1.

Definition: The all-subsets kernel is defined by the embedding

$$\phi : \mathbf{x} \mapsto (\phi_A(\mathbf{x}))_{A \subseteq \{1,...,n\}}$$

with the corresponding kernel $k_{\subseteq}(\mathbf{x},\mathbf{z})$ given by

$$k_{\subseteq}(\mathbf{x},\mathbf{z}) = \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle .$$

Computation of the kernel:

$$\begin{aligned} k_{\subseteq}(\mathbf{x},\mathbf{z}) &= \langle \phi(\mathbf{x}), \phi(\mathbf{z}) \rangle \\ &= \sum_{A \subseteq \{1,2,\ldots,n\}} \phi_A(\mathbf{x}), \phi_A(\mathbf{z}) \\ &= \sum_{A \subseteq \{1,2,\ldots,n\}} \prod_{i \in A} x_i z_i \qquad \left(= \prod_{i=1}^{n} (1 + x_i z_i) \right) \end{aligned}$$

Note that the kernel is expressed as a sum of products and this is a very natural structure a hypernetwork can represent. Indeed, a complete hypernetwork $H_{\subseteq}(\mathbf{x},\mathbf{z})$ consisting of all possible hyperedges $E_i \equiv A$ above represents the all-subsets kernel, i.e. $H_{\subseteq}(\mathbf{x},\mathbf{z}) \equiv k_{\subseteq}(\mathbf{x},\mathbf{z})$.

Note also that each subset in this feature space is assigned equal weight unlike the variable weightings characteristic of the polynomial kernel. We will see below that the same kernel can be formulated in a recursive way, giving rise to a class known as the ANOVA kernels. They build on a theme already apparent in the above where we see, but computed as a product of sums.

- **ANOVA kernel of degree *d***

The ANOVA kernel of degree *d* is like the all-subsets kernel except that it is restricted to subsets of the given cardinality *d*. That is,

$$\sum_{j=1}^{n} i_j = d .$$

The embedding of the ANOVA kernel of degree *d* is given by

$$\phi_d : \mathbf{x} \mapsto (\phi_A(\mathbf{x}))_{|A|=d}$$

where for each subset A the feature is given by

$$\phi_A(\mathbf{x}) = \prod_{i \in A} x_i = \mathbf{x}^{\mathbf{i}_A}$$

and $\mathbf{i}_A$ is the indicator function of the set A.

The dimension of the resulting embedding is $\binom{n}{d}$, since this is the number of such subsets. The resulting inner product is given by

$$k_d(\mathbf{x},\mathbf{z}) = \langle \phi_d(\mathbf{x}), \phi_d(\mathbf{z}) \rangle = \sum_{|A|=d} \phi_A(\mathbf{x}) \phi_A(\mathbf{z})$$

$$= \sum_{1 \le i_1 < i_2 < \cdots < i_d \le n} (x_{i_1} z_{i_1})(x_{i_2} z_{i_2}) \cdots (x_{i_d} z_{i_d})$$

$$= \sum_{1 \le i_1 < i_2 < \cdots < i_d \le n} \prod_{j=1}^{d} x_{i_j} z_{i_j}$$

Note that the ANOVA kernel $k_d(\mathbf{x}, \mathbf{z})$ of degree d is represented as a d-hypernetwork, i.e. a hypernetwork consisting only of hyperedges of cardinality d. Here a hyperedge $E_i \equiv A$ represents the feature map $\phi_A(\cdot)$ and the match of the images $\phi_A(\mathbf{x})$ and $\phi_A(\mathbf{z})$ of the two vectors **x** and **z** are computed by the match of two hyperedges $\phi_{E_i}(\mathbf{x})$ and $\phi_{E_i}(\mathbf{z})$, i.e. $\phi_{E_i}(\mathbf{x})\phi_{E_i}(\mathbf{z}) = \prod_{j=1}^{d} x_{i_j} z_{i_j}$.

- **ANOVA kernel of all degrees up to and including *d***

  (i.e. kernels using different degrees)

$$k_{\le d}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} k_s(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{d} \sum_{1 \le i_1 < i_2 < i_d \le n} \prod_{j=1}^{s} x_{i_j} z_{i_j}$$

If $d=n$, we recover the all-subsets kernel

$$k_{\subseteq}(\mathbf{x}, \mathbf{z}) = k_{\le n}(\mathbf{x}, \mathbf{z}) = \sum_{s=0}^{n} k_s(\mathbf{x}, \mathbf{z})$$

(computation: $k_{\subseteq}(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} (1 + x_i z_i)$)

Note that $(1 \ldots d)$-hypernetwork $H^{(1 \ldots d)}(\mathbf{x}, \mathbf{z})$ is an exact implementation of the kernel $k_{\le d}(\mathbf{x}, \mathbf{z})$.

Uneven term weighting. We can downplay some features and emphasize others by simply introducing a weighting factor $a_i \ge 0$, thus the all-subsets kernel becomes

$$k_{\subseteq}(\mathbf{x}, \mathbf{z}) = \prod_{i=1}^{n} (1 + a_i x_i z_i)$$

Note that the concept of hypernetwork (as a generalization of the hypegraph) already embodies the term weighting in terms of the weight values of the hyperedges.

- **General ANOVA kernel**

We can view the individual components $x_i z_i$ in the ANOVA kernel as a base kernel

$$k_i(\mathbf{x}, \mathbf{z}) = x_i z_i .$$

The reweighting scheme can be seen as changing this base kernel to

$$k_i(\mathbf{x}, \mathbf{z}) = a_i x_i z_i .$$

More generally, we can simply replace the n coordinates by n general base kernels that provide different methods of comparing the two objects. Generalizing further, we do not require that the ith kernel depends only on the ith coordinate. It might depend on a set of coordinates that could overlap with the coordinates affecting other features. In general, all of the base kernels could depend on all of the input coordinates. Given a sequence $k_1(\mathbf{x},\mathbf{z}),\ldots,k_n(\mathbf{x},\mathbf{z})$ of base kernels, this more general version of the ANOVA kernel (of degree d) therefore has the following form

$$k_d^A(\mathbf{x},\mathbf{z}) = \sum_{1\le i_1<i_2\cdots<i_d\le n}\prod_{j=1}^{d} k_{i_j}(\mathbf{x},\mathbf{z}).$$

Note that each hyperedge E_i in a hypernetwork represents a *microkernel*

$$h_i(\mathbf{x},\mathbf{z}) = \prod_{j=1}^{d} k_{i_j}(\mathbf{x},\mathbf{z})$$

which is a product of a sequence of base kernels. Since the microkernel is constructed by the hyperedge it will also be referred to as a hyperedge kernel. It is easy to verify that a d-hypernetwork represents the general ANOVA kernel $k_d^A(\mathbf{x},\mathbf{z})$.

- **Randomized kernels**

There are cases of kernels where the feature space is explicitly constructed by a vector

$$\phi(\mathbf{x}) = (\phi_i(\mathbf{x}))_{i=I},$$

and each of the individual feature maps $\phi_i(\mathbf{x})$ is simple to compute, but the size of the set I makes complete evaluation of the feature vector prohibitively expensive. If the feature space inner product is given by

$$\begin{aligned} k(\mathbf{x},\mathbf{z}) &= \left\langle (\phi_i(\mathbf{x}))_{i\in I}, (\phi_i(\mathbf{z}))_{i\in I} \right\rangle \\ &= \sum_{i\in I} \mu_i \phi_i(\mathbf{x})\phi_i(\mathbf{z}) \end{aligned}$$

with $\sum_i \mu_i = 1$ then we can estimate the value of the inner product by sampling indicies according to μ and forming an empirical estimate of the inner product through an average of N randomly drawn features

$$\hat{k}(\mathbf{x},\mathbf{z}) = \frac{1}{R}\sum_{i\sim\mu}\phi_i(\mathbf{x})\phi_i(\mathbf{z}) = \frac{1}{R}\sum_{i\sim\mu} h_i(\mathbf{x},\mathbf{z})$$

Where $h_i(\mathbf{x},\mathbf{z}) = \phi_i(\mathbf{x})\phi_i(\mathbf{z})$ is a random hyperedge kernel.

Note that a hypernetwork constructed by random sampling of hyperedges E_i corresponds to a randomized kernel. In this case, the number $|E|$ of hyperedges sampled is equal to the number N of random features drawn. Since the hyperedges E_i are sampled from the training data set, the indicies reflect

the distribution μ_i.

Hypernetworks as Randomized Kernel Machines

The kernel machines predicts its output by

$$\hat{y}(\mathbf{x}_q) = \text{sign}\left(\sum_{\mathbf{x}^{(n)} \in D} \lambda_n y_n k(\mathbf{x}_q, \mathbf{x}^{(n)}) + b \right)$$

Depending on the form of kernel functions adopted, the specific forms are different. Typical examples include the Gaussian kernels defined as

$$k(\mathbf{x}_q, \mathbf{x}^{(n)}) = \exp\left(-\frac{1}{2\sigma^2} \left\| \mathbf{x}_q - \mathbf{x}^{(n)} \right\|^2 \right)$$

and the polynomial kernels defined as

$$k(\mathbf{x}_q, \mathbf{x}^{(n)}) = (\mathbf{x}_q^T \mathbf{x}^{(n)} + 1)^p .$$

In case of the hypernetworks, the kernel is defined as

$$k(\mathbf{x}_q, \mathbf{x}^{(n)}) = \frac{1}{R} \sum_{i \sim \mu} \prod_{j=1}^{d} k_{i_j}(\mathbf{x}_q, \mathbf{x}^{(n)}) .$$

More specifically, the hypernetwork consists of the set E of random hyperedges E_i and the kernel function is computed by

$$\begin{aligned} k(\mathbf{x}_q, \mathbf{x}^{(n)}) &= \left\langle (\phi_{E_i}(\mathbf{x}_q))_{E_i \in E}, (\phi_{E_i}(\mathbf{x}^{(n)}))_{E_i \in E} \right\rangle \\ &= \sum_{E_i \in E} \mu_{E_i} \phi_{E_i}(\mathbf{x}_q) \phi_{E_i}(\mathbf{x}^{(n)}) \\ &= \sum_{E_i \in E} \mu_{E_i} \left\langle \phi_{E_i}(\mathbf{x}_q), \phi_{E_i}(\mathbf{x}^{(n)}) \right\rangle \\ &= \sum_{E_i \in E} \mu_{E_i} k_{E_i}(\mathbf{x}_q, \mathbf{x}^{(n)}) \\ &= \sum_{E_i \in E} \mu_{E_i} h_i(\mathbf{x}_q, \mathbf{x}^{(n)}) \\ &= \frac{1}{R} \sum_{i \sim \mu} \prod_{j=1}^{d} k_{i_j}(\mathbf{x}_q, \mathbf{x}^{(n)}) \end{aligned}$$

where the hyperedge kernels $h_i(\mathbf{x}_q, \mathbf{x}^{(n)}) = \left\langle \phi_{E_i}(\mathbf{x}_q), \phi_{E_i}(\mathbf{x}^{(n)}) \right\rangle = \prod_{j=1}^{d} k_{i_j}(\mathbf{x}_q, \mathbf{x}^{(n)})$ represent randomized "microkernels" corresponding to the hyperedges E_i and $k_{i_j}(\mathbf{x}_q, \mathbf{x}^{(n)})$ are the base kernels for the i_jth coordinate.

Note that the random hyperedges represent the feature space:

$$\phi(\mathbf{x}^{(n)}) = (\phi_{E_i}(\mathbf{x}^{(n)}))_{E_i \in E} = (k_{E_i}(\cdot, \mathbf{x}^{(n)}))_{E_i \in E}$$

A large number of these hyperedges in the hypernetwork build an associative memory consisting of the random *partial* patterns *sampled* from the training data.

Given a query pattern x_q, the hypernetwork $H(\mathbf{x}_q, \cdot)$ computes its output as follows:

$$\begin{aligned}
\hat{y}(\mathbf{x}_q) &= \text{sign}\left(\sum_{\mathbf{x}^{(n)} \in D} \lambda_n y_n k(\mathbf{x}_q, \mathbf{x}^{(n)}) + b \right) \\
&= \text{sign}\left(\sum_{\mathbf{x}^{(n)} \in D} \lambda_n y_n \sum_{E_i \in E} \mu_{E_i} \left\langle \phi_{E_i}(\mathbf{x}_q), \phi_{E_i}(\mathbf{x}^{(n)}) \right\rangle + b \right) \\
&= \text{sign}\left(\sum_{\mathbf{x}^{(n)} \in D} \lambda_n y_n \frac{1}{R} \sum_{i \sim \mu_n} \prod_{j=1}^{d} k_{i_j}(\mathbf{x}_q, \mathbf{x}^{(n)}) + b \right) \\
&= \text{sign}\left(y_i \frac{1}{N \cdot R} \sum_{i \sim \mu} \prod_{j=1}^{d} k_{i_j}(\mathbf{x}_q, \mathbf{x}) + b \right)
\end{aligned}$$

where $N = |D|$ and R is the number of hyperedges sampled from $\mathbf{x}^{(n)}$. μ_n is the distribution of the indices for the nth data point and μ is the distribution of the indices for the whole data set. For the query pattern $\mathbf{x}_q$, every hyperedge computes $\left\langle \phi_{E_i}(\mathbf{x}_q), \phi_{E_i}(\mathbf{x}^{(n)}) \right\rangle$ to which its label y_i is multiplied (note that each hyperedge has its label associated with it for pattern classification problems).

Relation to *k*-NN

If the hypernetwork contains N hyperedges of cardinality n (the number of features in the example) only and the hypernetwork chooses the best k hyperedges for voting, this is equivalent to a k-nearest neighbor classifier.

Relation to locally weighted k-NN with partial distance: Since the hyperedges are of varying cardinality with local weights and the variables in the hyperedges are sampled, the hypernetwork is more like a locally weighted k-NN where the (partial) distance is measured on $k < n$ attributes.

Question: Can you now explain why the k-hypernetworks without any training show competitive performance to k-NN in digit image recognition?

Regression using a hypernetwork of randomized polynomial kernels of degrees up to *d*

$$
\begin{aligned}
k_d^{poly}(\mathbf{x},\mathbf{z}) &= \sum_{s=0}^{d} \binom{d}{s} R^{d-s} \langle \mathbf{x},\mathbf{z} \rangle^s \\
&= \sum_{s=0}^{d} a_s \hat{k}_s(\mathbf{x},\mathbf{z}) \\
&= \sum_{s=0}^{d} a_s \left\langle (\phi_{\mathbf{i}}^{(s)}(\mathbf{x}))_{\mathbf{i}\in\mathbf{i}_s}, (\phi_{\mathbf{i}}^{s}(\mathbf{z}))_{\mathbf{i}\in\mathbf{i}_s} \right\rangle \\
&= \sum_{s=0}^{d} a_s \sum_{\mathbf{i}\in\mathbf{i}_s} \tilde{\mu}_{\mathbf{i}} \phi_{\mathbf{i}}^{(s)}(\mathbf{x}) \phi_{\mathbf{i}}^{(s)}(\mathbf{z}) \\
&= \sum_{s=0}^{d} \sum_{\mathbf{i}\in\mathbf{i}_s} \mu_{\mathbf{i}} \phi_{\mathbf{i}}^{(s)}(\mathbf{x}) \phi_{\mathbf{i}}^{(s)}(\mathbf{z})
\end{aligned}
$$

$$
\phi_{\mathbf{i}}^{(s)}(\mathbf{x}) = k_{\mathbf{i}}^{(s)}(\mathbf{x},\cdot) = \mathbf{x}^{\mathbf{i}_s} = x_1^{i_1} x_2^{i_2} \cdots x_n^{i_n}
$$

$$
\mathbf{i}_s \in (i_1,\cdots,i_n) \in \mathbb{N}^n \quad \text{and} \quad \sum_{j=1}^{n} i_j = s
$$

$$
k_d^{\sim poly}(\mathbf{x},\mathbf{z}) = \sum_{s=0}^{d} \frac{1}{N} \left(\sum_{\mathbf{i}\in\mu_i} \phi_{\mathbf{i}}^{(s)}(\mathbf{x}) \phi_{\mathbf{i}}^{(s)}(\mathbf{z}) \right)
$$

Discussion: Hypernetworks as Localist Models

- In hypernetworks, the kernel function computes the similarity of two *n*-dim vectors in terms of $k < n$ subdimensions.
- A hypernetwork makes use of a large number of simple random microkernels which correspond to the hyperedges.
- Since the hyperedges are randomly sampled from the training examples, a large number of hyperedges can effectively memorize the training data, which is a property of localist models.
- But, this memorization results in generalization since the original *n*-dim data is represented in $k < n$ subspaces (dimension reduction), which differs from other lazy learning methods such as *k*-NN.
- Despite the local nature of the hyperedges, a hypernetwork achieve global approximation though a linear combination of a large number of locally weighted microkernels.
- If the hypernetwork is constructed *k*-hyperedges of mixed *k* values, large *k*-hyperedges tend to realize local approximations and small *k*-hyperedges tend to realize global approximations
- Therefore, the hypernetwork models have the properties of both localist and globalist aspects of memory.
- One of the main issues of learning in hypernetworks is to find the best mixture of globalist and locaslist

hyperedges for modeling the data at hand (and those for the future).

- Generally, the criteria for the goodness of the models in machine learning is either the prediction performance or the explanation capability or both.
- In terms of explanation capability, the hyperedges (or microkernels) may shed light on the regularity underlying the data. This is important for data mining applications, such as for scientific data (e.g., bioinformatics) and business data (e.g., market analysis and customer service). For example, the hyperedges might correspond to the set of genes or modules that activate each other in a genetic network.
- The prediction performance can also be measured in a dynamical environment in which the training data comes in an on-line stream. In this case, the problem of balancing the short-term performance (adaptation) and the long-term performance (persistence or survival) is an important issue. Reference [Zhang, FOCI-2006] addresses this issue.

Summary and the road ahead:

We have seen the connection of hypernetwork models to the *localist* memory models, such as *k*-NN, RBF, Gaussian processes (GP), and kernel machines. We note that by specifying the specific form of hyperedges (and the associated potential functions), a hypernetwork can represent a different localist model. In this sense the hypernetworks are rather a framework which provides a uniform basis on which different learning and memory models can be compared and/or synthesized. In this chapter we have seen that the hypernetworks can make global approximation by combining a large number of local approximators or microkernels. In the next chapter (Chapter 8) we examine the relationship of the hypernetworks to the more explicit *globalist* models. These include the probabilistic graphical models such as Boltzmann machines, Bayesian networks, and other models. Based on this analysis, the extensions and improvements of the hypernetwork models and their learning algorithms are described in Chapter 9.