Lecture 13 - *Reproducing Kernel Hilbert Spaces*

OBJECTIVE: In this section, we derive one of the most fundamental theorems in machine learning. The theorem establishes a formal connection between kernels and dot products of feature vectors in a (possibly infinite) feature space.

Symmetric Positive Definite Matrices

If $\mathbf{A} = \mathbf{A}^\mathbf{T}$ (so $\mathbf{A}$ has to be square!) then $\mathbf{A}$ is said to be *symmetric*. If $\mathbf{A}$ is symmetric, its eigenvalues are real.

A matrix $\mathbf{A}$ is *symmetric positive (semi)definite* (SPD) if it is symmetric and

$$\mathbf{x}^\mathbf{T}\mathbf{A}\mathbf{x} \geq 0, \qquad \forall \mathbf{x} \neq \mathbf{0}.$$

If $\mathbf{A}$ is SPD, its eigenvalues are positive.

Kernels as Measures of Similarity

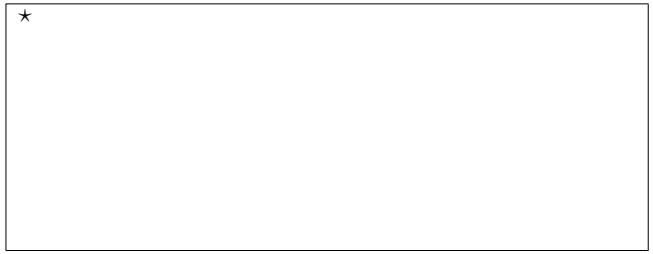
Given the data $\{\mathbf{x}_{1:n}, y_{1:n}\}$, where $\mathbf{x}_i \in \mathcal{X}$ is an object and y its label, we measure the similarity between objects with the following function (called kernel):

$$k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$$

The matrix of kernels $\mathbf{K}$ for a given dataset is called the **Gram matrix**. A kernel is SPD if the induced Gram matrix for any dataset is SPD.

Kernels allow us to map objects to function spaces:

$$\mathbf{x}_i \mapsto k(\cdot, \mathbf{x}_i)$$



Example: Kernel PCA

We can project items in high-dimensions to low dimensions, while preserving their topology. For example we can ensure that if two objects are far apart in high dimensions, then they will be far apart in low dimensions.

Initially, we have n data points in a d dimensional space in the matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$. We map this data to the funtion (feature) space "matrix" $\mathbf{\Phi} \in \mathbb{R}^{n \times \infty}$ and consider its SVD: $\mathbf{\Phi} = \mathbf{U}\Sigma\mathbf{V}^T$. Of course, we can't compute the SVD of a a matrix with an infinite number of columns. However, if we use the kernel trick: $\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T$, then we have:

$$\mathbf{K} = \mathbf{\Phi}\mathbf{\Phi}^T = \mathbf{U}\Sigma^2\mathbf{U}^T$$

So we can compute the eigenvalue decomposition of the n by n Gram matrix and then the k pricipal components are:

$$\mathbf{U}_k\Sigma_k = \mathbf{K}\mathbf{U}_k\Sigma_k^{-1}$$

Theorem 6 *A function $k : \mathcal{X} \times \mathcal{X} \mapsto \mathbb{R}$, which is either continuous or has finite domain, can be decomposed as a dot product of features (in a Hilbert space):*

$$k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle = \boldsymbol{\phi}(\mathbf{x}_i)\boldsymbol{\phi}(\mathbf{x}_j)^T$$

if and only if it is SPD.

Proof: First we prove the IF statement. That is, we begin by proving that if k admits the decomposition $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$, then k is SPD.

Next we prove the ONLY IF state. That is, if k is SPD, then it can be expressed as the dot product of feature mappings. The proof consits of the following three parts:

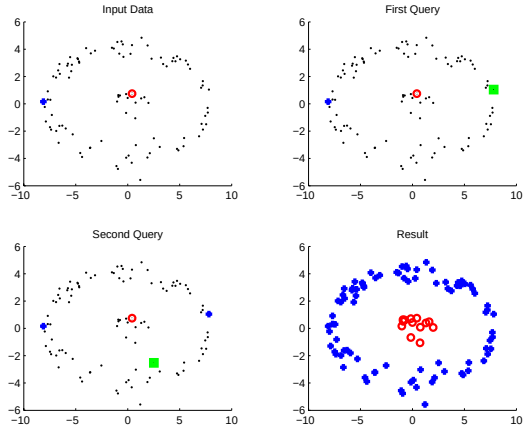
1. Construct a vector (linear) space containing the images of $\mathbf{x}$ under $\boldsymbol{\phi}$.
2. Define a dot product in this space and prove that it is valid.
3. Show that the dot product satisfies $k(\mathbf{x}_i, \mathbf{x}_j) = \langle \boldsymbol{\phi}(\mathbf{x}_i), \boldsymbol{\phi}(\mathbf{x}_j) \rangle$.

⋆ Step 1:

⋆ Step 2:

⋆ Step 3:

Lecture 14 - *Active Learning*



The top-left plot shows the initial data $\mathbf{x}$*, where only two points have been labelled. By running the active learning algorithm, the computer asks the user to enter the label for a point that could minimize the Bayes risk the most (the square in the top-right plot). The process is then repeated in the bottom-left plot. The final classification using only these four labels is perfect.*

One can use Bayesian decision theory to extend the semi-supervised learning approach to the active learning domain. In this approach, the class posterior probabilities $p(y_i^u|\mathbf{y}_l, \mathbf{x})$ are approximated with the estimates $0 \leq y_i^u \leq 1$, yielding the following expression for the posterior risk of the Bayes classifier:

$$R(\mathbf{y}_u) = \sum_{i=1}^{M} \min(y_i^u, 1 - y_i^u).$$

⋆ Proof:

To decide which unlabelled point $\mathbf{x}_j$ requires a label y_j, we first solve the linear system to obtain $\mathbf{y}_u$. Adding the new label y_j to the training set forces us to have to solve the linear system again. Fortunately, we can recompute the labels efficiently using the matrix inversion lemma.

Let the field obtained by adding the point $(\mathbf{x}_j, y_j)$ be denoted by $\mathbf{y}_u^{+(\mathbf{x}_j, y_j)}$. Then, the posterior risk is:

$$R(\mathbf{y}_u^{+(\mathbf{x}_j, y_j)}) = \sum_{i=1}^{M-1} \min\left(y_i^{+(\mathbf{x}_j, y_j)}, 1 - y_i^{+(\mathbf{x}_j, y_j)}\right).$$

Of course, before querying the user, we do not know the label y_j, so we have to use the expected posterior risk:

$$\mathbb{E}[R(\mathbf{y}_u^{+(\mathbf{x}_j, y_j)})] = (1 - y_i^u) R(\mathbf{y}_u^{+(\mathbf{x}_j, 0)}) + y_i^u R(\mathbf{y}_u^{+(\mathbf{x}_j, 1)})$$

After computing this expression, we pick the index j that minimizes it and ask the user to enter a label for $\mathbf{x}_j$. The pseudo-code follows.

Input: L, U and $\mathbf{W}$

WHILE more labels are required:

- Compute $\mathbf{y}_u$ using (1).
- Find the best query j using the expected posterior risk.
- Query point $\mathbf{x}_j$ and receive answer y_j.
- Add $(\mathbf{x}_j, y_j)$ to L and remove $\mathbf{x}_j$ from U.

The Learning Challenges

- Active Learning.
- Learning and Game Theory.
- Exciting new data.
- Function spaces and first order probabilistic inference.
- Solving complex probabilistic models in high dimensions.
- Combining all the bricks to build the cathedral.