CNEURO 2022
Neural networks: HW 2

Kameron Decker Harris

August 15, 2022

This is the theory section, but I also include a coding exercise. Some of these exercises will be difficult, depending on your background. Do what is useful for you and don't despair if you aren't sure how to get started; ask for help! Most of these exercises are to build your chops working with kernel function spaces.

Those of you with analysis backgrounds may notice I leave out some of the technical requirements for various limits to hold. If you care, you can think about those requirements, but I don't mind if you ignore them and assume everything just "works," physicist-style.

1. (Hebbian learning) This learning rule—"fire together, wire together"—applies to a linear network whose output $y(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$ with weights $\mathbf{w}$ and input $\mathbf{x}$. A sequence of inputs $(\mathbf{x}_i)_{i=1}^n$ are assumed to be presented to the network, and with every input i at time t the weights update as $\mathbf{w}_{t+1} = \mathbf{w}_t + \eta y(\mathbf{x}_i)\mathbf{x}_i$, where $\eta > 0$ is some small parameter.

Show that in the limit of many iterations over the dataset, the weights tend to align as $\mathbf{w}_t \propto \mathbf{v}$, where $\mathbf{v}$ is the leading eigenvector of the data covariance matrix. For simplicity, assume the inputs are centered, i.e. $\sum_{i=1}^n \mathbf{x}_i = 0$, and the covariance $C = \sum_{i=1}^n \mathbf{x}_i\mathbf{x}_i^T$. *(Complete the sketch of the proof we started in lecture 1.)*

2. (Random feature network kernels are kernels) In random feature networks, feature layer neuron i has activity given by $\phi_i(\mathbf{x}) = h(\mathbf{w}_i^T\mathbf{x})$ for an input $\mathbf{x}$ and some nonlinearity h. The weight vector $\mathbf{w}_i$ is drawn independently and identically distributed (iid) from some distribution μ for all $i = 1, \ldots, m$ feature neurons.

The kernel associated with the network is

$$k(\mathbf{x}, \mathbf{x}') = \mathbb{E}[h(\mathbf{w}^T\mathbf{x})h(\mathbf{w}^T\mathbf{x}')] = \lim_{m\to\infty} \frac{1}{m}\sum_{i=1}^m \phi_i(\mathbf{x})\phi_i(\mathbf{x}'). \tag{1}$$

For k to be a "real" kernel, we need these properties:

1. **Symmetry:** For any two inputs, $k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')$.
2. **Definiteness:** For any set of n datapoints, the $n \times n$ kernel matrix $\mathbf{K} = (k(\mathbf{x}_i, \mathbf{x}_j))_{ij}$ satisfies $\mathbf{c}^T\mathbf{K}\mathbf{c} \geq 0$ for any $\mathbf{c} \in \mathbb{R}^n$. In other words, the kernel matrix is positive definite.

Show that k is a symmetric, positive definite kernel. Are there any edge cases where you need to assume something about h or μ for this to be true?

3. (Mercer kernels) Assume we have a kernel which can be written as

$$k(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^\infty \lambda_i e_i(\mathbf{x})e_i(\mathbf{x}'), \tag{2}$$

where the eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots \geq 0$, e_i is an orthonormal basis of functions, and the series (2) converges absolutely and uniformly. The kernel is positive definite and defines a reproducing kernel Hilbert space (RKHS). We have some function $f(\mathbf{x}) = \sum_{i=1}^{\infty} c_i e_i(\mathbf{x})$ in this same basis, where

$$\sum_{i=1}^{\infty} \frac{c_i^2}{\lambda_i} < \infty. \tag{3}$$

(3.1) Verify that, for another function $g(\mathbf{x}) = \sum_{i=1}^{\infty} d_i e_i(\mathbf{x})$ in the same basis with (3) holding for the coefficients d_i,

$$\langle f, g \rangle_{\mathcal{H}} = \sum_{i=1}^{\infty} \frac{c_i d_i}{\lambda_i} \tag{4}$$

is an inner product.

(3.2) Give a formula for the Hilbert space norm associated with $\langle \cdot, \cdot \rangle_{\mathcal{H}}$.

(3.3) (Harder) Prove that $\mathcal{H}$ = the space of all functions f that satisfy (3) equipped with the inner product (4) is the RKHS of the kernel k. The last property to verify is the reproducing property:

- **Reproducing property:** $\delta_{\mathbf{x}}(\cdot) = k(\cdot, \mathbf{x})$ is a member of the function space as a function of the unspecified argument, and $\langle \delta_{\mathbf{x}}, f \rangle_{\mathcal{H}} = f(\mathbf{x})$.

(Why this problem is here: The ease of learning a function with a given kernel can be described in terms of the RKHS norm of that function. So, how hard it is for a network to perform a learning task depends on the target function's norm in the network RKHS. Also, this gives you a good reminder of the inner-product definitions.)

4. (RKHS norms, hard) Consider the function $f(\mathbf{x}) = \frac{1}{\sqrt{\pi}} \cos(x_1)$ for $\mathbf{x} \in [-\pi, \pi]^d$. Let

$$e_n(x) = \frac{1}{\sqrt{\pi}} \cos(nx), \qquad e_0(x) = \frac{1}{\sqrt{2\pi}} \tag{5}$$

be an orthonormal basis for all even functions on $[-\pi, \pi]$ with $n \in \mathbb{Z}$. We can construct an orthonormal basis for all even functions in d-dimensions by taking products:

$$e_{\mathbf{n}}^{\text{prod}}(\mathbf{x}) = \prod_{i=1}^{d} e_{n_i}(x_i) \tag{6}$$

where $\mathbf{n} \in \mathbb{Z}^d$. (If you aren't sure this is orthonormal, go ahead and verify for yourself.)

Let $\{\lambda_i\}_{i \in \mathbb{Z}}$ be some sequence of eigenvalues $\lambda_1 \geq \lambda_2 \geq \ldots$ which is summable.

Consider two kernels:

$$K_{\text{prod}}(\mathbf{x}, \mathbf{x}') = \sum_{\mathbf{n} \in \mathbb{Z}^d} \left(\prod_{i=1}^{d} \lambda_{n_i} \right) e_{\mathbf{n}}^{\text{prod}}(\mathbf{x}) \tag{7}$$

$$K_{\text{add}}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} \sum_{n=0}^{\infty} \lambda_n e_n(x_i) \tag{8}$$

Then $\{e_{\mathbf{n}}^{\text{prod}}(\mathbf{x})\}_{\mathbf{n} \in \mathbb{Z}^d}$ is an orthonormal basis for the RKHS $\mathcal{H}_{\text{prod}}$ and $\oplus_{i=1}^{d} \{e_n(x_i)\}_{n \in \mathbb{Z}}$ (the orthogonal direct sum of the bases for each coordinate, i.e. the union of the sets of single-variable basis functions (5) for all variables x_i from $i = 1, \ldots, d$) for the RKHS $\mathcal{H}_{\text{add}}$.

Show that $\|f\|_{\mathcal{H}}$ is exponentially larger (as a factor of d) in the RKHS under K_{prod} compared to K_{add}. Hints: You may want to start by thinking about the edge cases of $d = 1, 2$ and go from there. If you want, you may pick an eigenvalue sequence.

5. (Kernels versus networks) Use the same "circles" dataset that we used in the first part of the homework, problem 2. We are going to compare two methods. First, we will make a random feature network. Each feature will use a cosine encoding:

$$\phi_i(\mathbf{x}) = \sqrt{\frac{2}{m}} \cos(\mathbf{w}_i^T \mathbf{x} + b), \tag{9}$$

where $\mathbf{w}_i \sim \mathcal{N}(0, \sigma^2)$ (normal distribution with mean 0 and standard deviation σ) and $b \sim U[0, 2\pi]$ and $i = 1, \ldots, m$. Train the network to learn to classify the circles problem using a linear SVM readout applied to the feature vector $\boldsymbol{\phi}$.

Second, compare this to the performance of the kernel SVM using the Gaussian kernel

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\sigma^2}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right). \tag{10}$$

Compare the training and testing errors on this problem for the two methods as a function of σ for a few different m. A good range would be $10^{-2} \leq \sigma \leq 10^2$ and $m = 100, 200, 400, 600, 800, 2000$. Make sure the SVM regularization is matched between the methods. Scikit-learn has a nice implementation of SVM and kernel SVM that will work, and the default regularization is fine.

Do the methods perform similarly? When do you get deviations? Is double-descent possible here? Take note of the number of training points that were used. Plot training and testing error versus σ as well as error versus m for a fixed $\sigma = 1$.