Supervised Learning

November 2024

1 3.1 Questions

9. Kernel Modification

(a) To determine for which values of $c \in R$ the kernel

$$K_c(\mathbf{x}, \mathbf{z}) = c + \sum_{i=1}^{n} x_i z_i$$

is positive semi-definite, we note that a kernel K is positive semi-definite if, for any set of points $\{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_m\} \subset R^n$, the matrix $[K(\mathbf{x}_i, \mathbf{x}_j)]$ is positive semi-definite. This means that for any vector $\mathbf{a} \in R^m$,

$$\sum_{i=1}^{m} \sum_{j=1}^{m} a_i a_j K(\mathbf{x}_i, \mathbf{x}_j) \geq 0.$$

We can rewrite $K_c(\mathbf{x}, \mathbf{z})$ as:

$$K_c(\mathbf{x}, \mathbf{z}) = c + \mathbf{x} \cdot \mathbf{z},$$

where $\mathbf{x} \cdot \mathbf{z}$ is the standard inner product, a known positive semi-definite kernel.

The constant c adds a uniform shift to the kernel matrix. If $c \geq 0$, this preserves the positive semi-definiteness of the inner product kernel. If $c < 0$, the kernel matrix may gain negative eigenvalues, violating positive semi-definiteness.

Therefore, K_c remains positive semi-definite if $c \geq 0$.

(b) When we use $K_c(\mathbf{x}, \mathbf{z}) := c + \mathbf{x} \cdot \mathbf{z}$ in linear regression, c affects the solution by uniformly shifting all values in the kernel matrix K. Each entry is $K_{ij} = c + \mathbf{x}_i \cdot \mathbf{x}_j$, so increasing c raises the similarity scores between all data points equally.

For small values of c, this shift keeps the original structure of the kernel matrix largely intact, preserving distinctions between points and adding slight regularisation by increasing stability.

If c becomes very large, it overwhelms the dot product values, making all entries in K nearly equal. This effectively treats all data points as equally similar, leading to underfitting, as the model can no longer distinguish between points and fails to capture meaningful patterns.

In summary, c influences the balance between stability and sensitivity: a small positive c stabilises the model, while a large c leads to underfitting by erasing differences between data points.

This is somewhat analogous to introducing a soft margin in SVMs, where allowing slight violations of the margin constraint can improve generalisation. Similarly, adding c provides regularisation, making the model less sensitive to specific data variations and balancing between a strict fit and better generalisation.

10. Gaussian Kernel Classifier

To enable a Gaussian kernel-based linear classifier to simulate a 1-Nearest Neighbour (1-NN) classifier, we choose a large value of β in the Gaussian kernel $K_\beta(x,t) = \exp(-\beta\|x-t\|^2)$. As $\beta \to \infty$, the kernel value $K_\beta(x_i,t)$ decays rapidly with increasing distance between x_i and t. For sufficiently large β, this decay ensures that $K_\beta(x_i,t)$ is close to zero for all points x_i except the nearest point x_k to t. Consequently, the function

$$f(t) = \sum_{i=1}^{m} \alpha_i K_\beta(x_i,t)$$

is dominated by the term involving the nearest neighbour x_k, making $f(t) \approx \alpha_k K_\beta(x_k,t)$. The classifier then outputs $sign(f(t))$, which, for large β, primarily reflects the label y_k of the nearest neighbour. In summary, selecting β to be sufficiently large effectively isolates the influence of the nearest neighbour on $f(t)$, enabling the classifier to behave like a 1-NN classifier by basing its decision solely on the label of the closest point.

11. No Free Lunch Theorem

(a) Let $C \subset \mathcal{X}$ be a subset of the input space with cardinality $|C| = 2n$. Denote by $\mathcal{Y}^C = \{f_1, \ldots, f_T\}$ the set of all possible functions from C to $\mathcal{Y} = \{-1, 1\}$. There are $T = |\mathcal{Y}^C| = 2^{2n}$ such functions. For each function $f_i \in \mathcal{Y}^C$, we define a distribution ρ_i over $C \times \mathcal{Y}$ such that $\rho_i(\{x,y\}) = \frac{1}{2n} if y = f_i(x), and 0 otherwise$. This distribution assigns a probability of $\frac{1}{2n}$ to each pair (x,y) where $y = f_i(x)$, and zero otherwise.

The expected loss $E_{\rho_i}(f_i)$ is then calculated as follows:

$$E_{\rho_i}(f_i) = \sum_{x \in C} \sum_{y \in \{-1,1\}} \mathbf{1}_{\{f_i(x) \neq y\}} \rho_i(\{x,y\}).$$

Since $\rho_i(\{x,y\})$ is non-zero only when $y = f_i(x)$, the indicator function $\mathbf{1}_{\{f_i(x)\neq y\}}$ is always zero. Therefore, we conclude that $E_{\rho_i}(f_i) = 0$. This establishes that

$$\inf_{f:\mathcal{X}\to\mathcal{Y}} E_\rho(f) = 0.$$

(b) Let C^n denote the set of all possible datasets of size n consisting of points in C, i.e., $C^n = \{S_1, \ldots, S_k\}$ where $k = |C^n| = (2n)^n$. For each $j = 1, \ldots, k$ and for each input set $S_j = \{x_1, \ldots, x_n\} \in C^n$, define the corresponding dataset for function f_i as

$$S_j^i = \{(x_1, f_i(x_1)), \ldots, (x_n, f_i(x_n))\}.$$

This dataset S_j^i is constructed by associating each $x \in S_j$ with its output under f_i.

We seek to prove that

$$\max_{i=1,\ldots,T} E_{S\sim\rho_i^m} E_{\rho_i}(A(S)) \geq \min_{j=1,\ldots,k} \frac{1}{T} \sum_{i=1}^{T} E_{\rho_i}(A(S_j^i)).$$

Using the hint, we establish that for any $i = 1, \ldots, T$,

$$E_{S\sim\rho_i^m} E_{\rho_i}(A(S)) = \frac{1}{2n} \sum_{j=1}^{k} E_{\rho_i}(A(S_j^i)).$$

By averaging over all functions f_i and datasets S_j, we obtain the inequality

$$\max_{i=1,\dots,T} E_{S\sim\rho_i^m} E_{\rho_i}(A(S)) \geq \min_{j=1,\dots,k} \frac{1}{T}\sum_{i=1}^{T} E_{\rho_i}(A(S_j^i)),$$

(c) To establish a lower bound on the risk, we introduce a subset $S_j' \subset C$ such that S_j' contains elements not present in S_j. Specifically, for each dataset S_j, let $S_j' = \{v_1, \dots, v_p\} \subset C \setminus S_j$, where $p \geq m$. This subset S_j' is chosen so that it consists of points disjoint from S_j, meaning the learning algorithm A has not encountered any elements of S_j' during training on S_j.

Now, consider the expected risk of the algorithm over the points in S_j', with respect to the function f_i. We define the expected error over S_j' as:

$$\frac{1}{T}\sum_{i=1}^{T} E_{\rho_i}(A(S_j^i)) = \frac{1}{T}\sum_{i=1}^{T}\frac{1}{p}\sum_{v\in S_j'} \mathbf{1}_{\{A(S_j)(v)\neq f_i(v)\}}.$$

Using the hint provided, we apply the inequality

$$\frac{1}{p}\sum_{v\in S_j'} \mathbf{1}_{\{A(S_j)(v)\neq f_i(v)\}} \geq \min_{v\in S_j'} \mathbf{1}_{\{A(S_j)(v)\neq f_i(v)\}},$$

which gives us a lower bound on the risk over S_j' by focusing on the minimum error over points in S_j'. This allows us to bound the average risk from below as:

$$\frac{1}{T}\sum_{i=1}^{T} E_{\rho_i}(A(S_j^i)) \geq \min_{j=1,\dots,k} \frac{1}{T}\sum_{i=1}^{T} E_{\rho_i}(A(S_j^i)).$$

Thus, we conclude that the expected risk over the subset S_j' establishes a lower bound for the overall risk of the learning algorithm A by considering errors on points that the algorithm has not seen during training. This bound on the expected error across unseen data highlights that there exists a minimum achievable risk, supporting the "No Free Lunch" theorem in machine learning.

(d) With the same assumptions as above, we now aim to show that for any $v \in R_j$,

$$\frac{1}{T}\sum_{i=1}^{T} \mathbf{1}_{\{A(S_j^i)(v)\neq f_i(v)\}} = \frac{1}{2}.$$

To achieve this, we utilize a partition of the function set $\mathcal{Y}^C$ into $T/2$ pairs of functions $(f_i, f_{i'})$ such that for each pair $(f_i, f_{i'})$, the functions differ only on a single input point $x = v$, i.e., $f_i(x) \neq f_{i'}(x)$ if and only if $x = v$. Thus, for each pair $(f_i, f_{i'})$,

$$\mathbf{1}_{\{A(S_j^i)(v)\neq f_i(v)\}} + \mathbf{1}_{\{A(S_j^{i'})(v)\neq f_{i'}(v)\}} = 1,$$

ensuring that across all pairs, the error indicator $\mathbf{1}_{\{A(S_j^i)(v)\neq f_i(v)\}}$ is $\frac{1}{2}$ on average. Therefore,

$$\frac{1}{T}\sum_{i=1}^{T} \mathbf{1}_{\{A(S_j^i)(v)\neq f_i(v)\}} = \frac{1}{2}.$$

(e) We now prove an auxiliary result. Let Z be a random variable with values in $[0, 1]$ and expected value $E[Z] = \mu$. We wish to show that for any $a \in (0, 1)$,

$$P(Z > 1 - a) \geq \frac{\mu - (1 - a)}{a}.$$

To derive this result, we apply *Markov's inequality*. Let $W = 1 - Z$, so $W \geq 0$ and $W \leq 1$. Then $Z > 1 - a$ is equivalent to $W < a$. Since $E[W] = 1 - \mu$, Markov's inequality gives

$$P(W \geq a) \leq \frac{E[W]}{a} = \frac{1-\mu}{a}.$$

Therefore,

$$P(Z > 1 - a) = 1 - P(W \geq a) \geq 1 - \frac{1-\mu}{a} = \frac{\mu - (1-a)}{a}.$$

(f) We now combine the results of the previous steps to show that for any algorithm A and any integer n, there exists a distribution ρ over $\mathcal{X} \times \mathcal{Y}$ such that

$$P_{S \sim \rho^m}\left(E_\rho(A(S)) > \frac{1}{8}\right) \geq \frac{1}{7}.$$

Let $Z = E_\rho(A(S))$, representing the expected error of the algorithm A on a dataset S. From the previous steps, we have established that $E_{S\sim\rho^m}[Z] \geq \frac{1}{4}$. Applying the auxiliary result from part (e) with $\mu = \frac{1}{4}$ and $a = \frac{7}{8}$, we obtain

$$P\left(Z > 1 - \frac{7}{8}\right) = P\left(Z > \frac{1}{8}\right) \geq \frac{\frac{1}{4} - \left(1 - \frac{7}{8}\right)}{\frac{7}{8}}.$$

Simplifying this expression, we get

$$P\left(Z > \frac{1}{8}\right) \geq \frac{\frac{1}{8}}{\frac{7}{8}} = \frac{1}{7}.$$

Thus, we have shown that there exists a distribution ρ for which the probability that the expected error of A exceeds $\frac{1}{8}$ is at least $\frac{1}{7}$, completing the proof.