Indeed there can be applied an existing $k$-d tree method for density estimation and it will speed up the computation of the objective function: $\sum_i p(c_i|\mathbf{x}_i)$. What I am arguing is that for the gradient computation is not that simple to apply the same ideas.

From what I understand, for a query point $\mathbf{x}_q$ and a group of points $G$, there are basically two types of approximations that can be done:

- If the points $\mathbf{x}_i$ in $G$ are very close together we can approximate each individual contribution $k_A(\mathbf{x}_q, \mathbf{x}_i)$ with some sort of an average contribution, let's say $k_A(\mathbf{x}_q, \mu_G)$.
- If the points $\mathbf{x}_i$ in $G$ are very far from the query point $\mathbf{x}_q$, we can ignore their contribution: $k_A(\mathbf{x}_i, \mathbf{x}_q) = 0, \forall \mathbf{x}_i \in G$.

I think that the first type of approximation won't give any improvements for the gradient computation.

The derivative consists of sums of the following form: $S_i = \sum_i k_A(\mathbf{x}_q, \mathbf{x}_i)\mathbf{x}_{qi}\mathbf{x}_{qi}^{\mathrm{T}}$, where $\mathbf{x}_{qi}$ are $D$-dimensional column vectors. If I try to use the first type of pruning $S_i \approx k_A(\mathbf{x}_q, \mu_G)\sum_i \mathbf{x}_{qi}\mathbf{x}_{qi}^{\mathrm{T}}$ then, because I cannot really cache those matrices, I will still have to compute their values for each $q$ and $i$ and for each iteration; the complexity will remain $O(ND^2)$. An idea might be to also approximate $\mathbf{x}_{qi}$ with an average version of the group $\mathbf{x}_{qG}$, but this is not really correct, because the points $\mathbf{x}_i$ form a node of the $k$-d tree in the projected space, whereas $\mathbf{x}_{qi}$ refer to the points in the original space.

On the other hand, the second type of approximation, that excludes some of the points, will give some speed-ups for the gradient because we don't have to compute all the statistics $\mathbf{x}_{qi}\mathbf{x}_{qi}^{\mathrm{T}}$. Also lower and upper bounds can be introduced to get a sense of the maximum error that can result. Moreover, we can accumulate weights in a best-first search and stop when the sum doesn't change significantly.

For convenience, here are the equations of the objective function and the gradient:

$$\begin{aligned} f(A) &= \frac{1}{N}\sum_{i=1}^{N} p_i = \frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\omega_i} p_{ij} \\ &= \frac{1}{N}\sum_{i=1}^{N}\sum_{j\in\omega_i} \frac{e^{-d_{ij}^2}}{\sum_{k\neq i} e^{-d_{ij}^2}}, \end{aligned} \tag{1}$$

where $d_{ij}^2 = \|A\mathbf{x}_i - A\mathbf{x}_j\|^2$.

By differentiating with respect to $A$, we obtain the gradient:

$$\frac{\partial f}{\partial A} = 2A \sum_{i=1}^{N} \left( p_i \sum_{k=1}^{N} p_{ik} \mathbf{x}_{ik} \mathbf{x}_{ik}^{\mathrm{T}} - \sum_{j \in \omega_i} p_{ij} \mathbf{x}_{ij} \mathbf{x}_{ij}^{\mathrm{T}} \right), \tag{2}$$

where $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$.