Interpretable Multiple-Kernel Prototype Learning for Discriminative Representation and Feature Selection

1 Supplementary Materials

This document provides supplementary material regarding the technical aspects of the paper and additional insights to the experiments and will be available in the online public repository along with the code of the algorithm[1].

1.1 Proposition 2 and its proof

Proposition 1. *The objective $\mathcal{J}_{dis}$ in Eq. (paper-5) has its minimum if* $\forall \vec{x}_i$, $\hat{\Phi}(\vec{x}_i) \approx \hat{\Phi}(\mathbf{X})\mathbf{U}\vec{\gamma}_i$ *s.t.* $\forall t : \gamma_{ti} \neq 0, \forall s : u_{st} \neq 0$, $\vec{l}_i = \vec{l}_s$ *and* $\|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 \approx 0$.

Proof. The objective term $\mathcal{J}_{dis}$ is constructed upon summation and multiplication of non-negative elements. Hence, its global minima would lie where $\mathcal{J}_{dis}(\mathbf{U}, \mathbf{\Gamma}) = 0$ holds. This condition can be fulfilled if for each $\vec{\gamma}_i$:

$$[\sum_{s=1}^{N} \vec{u}^s (\vec{l}_i^\top \vec{l}_s \|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 + \|\vec{l}_i - \vec{l}_s\|_2^2)]\vec{\gamma}_i = 0.$$

Since the trivial solution $\vec{\gamma}_i = 0$ is avoided due to $\mathcal{J}_{rec}$ in Eq. (paper-4), we can find a set $\mathcal{I}$ s.t. $\forall t \in \mathcal{I}, \gamma_{ti} \neq 0$ holds. Therefore, $\forall t \in \mathcal{I}, \sum_{s=1}^{N} u_{st}\Omega_{si} = 0$, where

$$\Omega_{si} = \vec{l}_i^\top \vec{l}_s \|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 + \|\vec{l}_i - \vec{l}_s\|_2^2.$$

It is clear that

$$\Omega_{si} = \begin{cases} 2 & \vec{l}_i \neq \vec{l}_s \\ \|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 & \vec{l}_i = \vec{l}_s, \end{cases}$$

which means that $\forall s$, $u_{st}\Omega_{si} = 0$ holds in either of the following cases:

1. $u_{st} = 0$, meaning that the data point $\vec{x}_s$ does not contribute to the t-th prototype (e.g., consider the squares in Figure 1-b which are not a part of $\vec{u}_1$) .
2. $\vec{u}_t$ uses $\vec{x}_s$ that lies in the same class as $\vec{x}_i$ (e.g., the circles in Figure 1-b as the main constituents of $\vec{u}_1$).

Putting all the above conditions together, $\mathcal{J}_{dis} = 0$ happens only if in case of the condition described by the proposition. □

[1] https://github.com/XXX

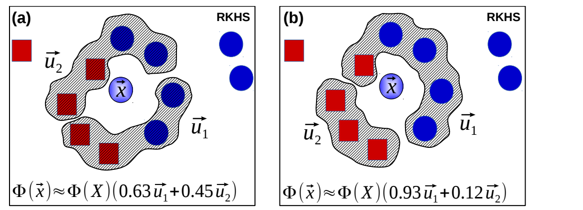

Figure 1: The effect of $\mathcal{J}_{dis}$ in Eq. (paper-4). (a). When $\lambda = 0$, the prototypes $(\vec{u}_1, \vec{u}_2)$ (the hatched selections) are shaped and reconstruct $\hat{\Phi}(\vec{x})$ by its neighboring samples from both of the classes (circles and squares). (b). When $\lambda \neq 0$, these prototypes are formed s.t. $\hat{\Phi}(\vec{x})$ is approximately represented by $\vec{u}_1$ mostly using its local, same-class neighbors (circles).

Although Proposition 1 describes the ideal situations, in practice, it is common to observe $\|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 < \epsilon$ for a small non-negative ϵ when $\vec{x}_s$ is among the neighboring points of $\vec{x}_i$. This condition results in a small non-zero minima for $\mathcal{J}_{dis}$. Besides, for a given $\vec{x}_i$, if its cross-class neighbors lie closer to its same-class neighbors, Ω_{si} obtains higher values by choosing $\vec{x}_s$ s.t. $\vec{x}_s \neq \vec{l}_i$ in favor of betterh minimizing $\mathcal{J}_{rec}$ (e.g., the squares in Figure 1-b which is a part of $\vec{u}_1$).

1.2 Derivation of the Optimization Problems in Sec. 4

Based on the dot-product relationship in Eq. (paper-2), we rewrite $\mathcal{J}_{rec}$ from Eq. (paper-4) as

$$\mathcal{J}_{rec} = \mathrm{Tr}(\hat{\mathcal{K}} + \mathbf{\Gamma}^\top \mathbf{U}^\top \hat{\mathcal{K}} \mathbf{U}\mathbf{\Gamma} - 2\hat{\mathcal{K}}\mathbf{U}\mathbf{\Gamma}), \quad (1)$$

where $\mathrm{Tr}(.)$ is the trace operator. In addition, the $\mathcal{J}_{dis}$ term from Eq. (paper-5) can be reformulated as

$$\begin{aligned} \mathcal{J}_{dis} = \; & \tfrac{1}{2}\sum_{i=1}^{N}\sum_{s=1}^{N} \vec{u}^s \vec{\gamma}_i \|\hat{\Phi}(\vec{x}_i) - \hat{\Phi}(\vec{x}_s)\|_2^2 (\vec{l}_i^\top \vec{l}_s) \\ & + \tfrac{1}{2}\sum_{i=1}^{N}\sum_{s=1}^{N} \vec{u}^s \vec{\gamma}_i \|\vec{l}_i - \vec{l}_s\|_2^2 \\ & = \mathrm{Tr}(\mathcal{L}(\mathbf{U}\mathbf{\Gamma}\mathbf{L}^\top\mathbf{L})\hat{\Phi}(\mathbf{X})^\top \hat{\Phi}(\mathbf{X})) \\ & \quad + \mathrm{Tr}(\mathcal{L}(\mathbf{U}\mathbf{\Gamma})\mathbf{L}^\top\mathbf{L}) \\ & = \mathrm{Tr}[(\mathcal{K}_{\mathbf{L}} - \mathcal{K}_{\mathbf{L}} \odot \mathcal{K})\mathbf{U}\mathbf{\Gamma}] + \mathrm{Tr}[(\mathbf{1} - \mathcal{K}_{\mathbf{L}})\mathbf{U}\mathbf{\Gamma}] \\ & = \mathrm{Tr}[(\mathbf{1} - \mathcal{K}_{\mathbf{L}} \odot \mathcal{K})\mathbf{U}\mathbf{\Gamma}], \end{aligned} \quad (2)$$

where $\mathcal{K}_{\mathbf{L}} = \mathbf{L}^\top\mathbf{L}$, and $\mathcal{L}(.)$ is the Laplacian operator [Von. Luxburg, 2007]. Regarding $\mathcal{J}_{ls}(\vec{\beta})$, we formulate it as

$$\begin{aligned}&\mathcal{J}_{ls}(\vec{\beta}) = \\ &\sum_{i=1}^{N}\big[\textstyle\sum_{s\in\mathcal{N}_i^k}\hat{\mathcal{K}}(\vec{x}_i,\vec{x}_i)+\hat{\mathcal{K}}(\vec{x}_s,\vec{x}_s)-2\hat{\mathcal{K}}(\vec{x}_i,\vec{x}_s) \\ &\qquad+\textstyle\sum_{s\in\overline{\mathcal{N}_i^k}}\hat{\mathcal{K}}(\vec{x}_i,\vec{x}_s)\big].\end{aligned} \tag{3}$$

Also, since the base kernels are normalized beforehand and $\|\vec{\beta}\|_1 = 1, \forall i, \hat{\mathcal{K}}(\vec{x}_i,\vec{x}_i) = 1$. Hence,

$$\begin{aligned}&\mathcal{J}_{ls}(\vec{\beta}) = \\ &\sum_{i=1}^{N}\sum_{s\in\mathcal{N}_i^k}[2-2\hat{\mathcal{K}}(\vec{x}_i,\vec{x}_s)]+\sum_{s\in\overline{\mathcal{N}_i^k}}\hat{\mathcal{K}}(\vec{x}_i,\vec{x}_s).\end{aligned} \tag{4}$$

Considering the Eqs. (1)-(4), we can derive the optimization problem of $\vec{\gamma}_i$ (Eq. (paper-7)) by decomposing Eqs. (1),(2) in terms of $\vec{\gamma}_i$.

Regarding the optimization of $\vec{u}_i$, we decompose

$$\mathcal{J}_{rec} = \mathrm{Tr}(\vec{\gamma}^{i^\top}\vec{u}_i^\top\hat{\mathcal{K}}\vec{u}_i\vec{\gamma}^i)+\mathrm{Tr}(\mathbf{E}_i^\top\hat{\mathcal{K}}\mathbf{E}_i)-\mathrm{Tr}(2\mathbf{E}_i^\top\hat{\mathcal{K}}\vec{u}_i\vec{\gamma}^i),$$

where $\vec{\gamma}^i\vec{\gamma}^{i^\top} \in \mathbb{R}$. Therefore, also by decomposing $\|\mathbf{LU}\|_1$ and Eq. (2) in terms of $\vec{u}_i$ we can derive the update formulation of $\vec{u}_i$ in Eq. (paper-9).

Based on the provided decompositions in Eqs. (1)-(4), derivation of the parts in Eq. (paper-11) is straightforward.

1.3 Public Repository of the Selected Datasets

We have noticed the main feature selection repository that we used to obtain the vectorial datasets in our experiments (`CLL_SUB_111`,`TOX_171`, and `Isolet`) is currently inactive. Therefore we refer to their mirror repository to obtain those datasets. [2]

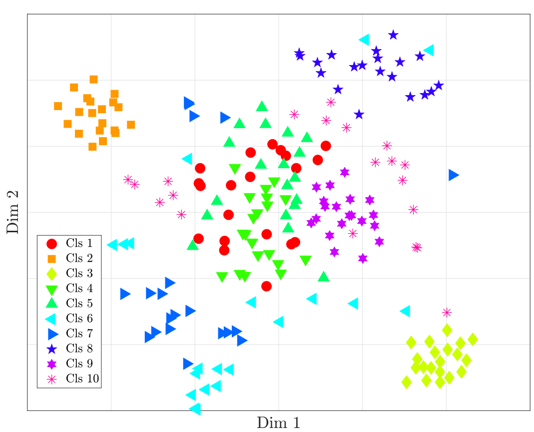

Figure 2: 2-dimensional embedding of the `UTKinect` dataset (based on the average-kernel) which visualizes the relative overlapping of the classes.

[2]https://github.com/jundongl/scikit-feature/tree/master/skfeature/data

Classes	1	2	3	4	5	
Names	walk	sit down	stand up	pick up	carry	
Prototypes	7	2	2	8	8	
Classes	6	7	8	9	10	Total
Names	throw	push	pull	wave	clap	
Prototypes	6	5	4	3	5	50

Table 1: Number of prototypes assigned to each class of the `UTKinect` dataset.

1.4 Detailed Analysis of the Prototypes

Although we decide on total number of the prototypes to learn for each dataset by choosing $c = pT_0$, the IMKPL automatically assigns proper number of prototypes to each class of data. As reported in Table 1, we examined the frequency of prototypes per class on the `UTKinect` dataset, which shows a notable variation among them. Also, by considering the 2-dimensional embedding of the `UTKinect` dataset (using the t-SNE algorithm [van der Maaten and Hinton, 2008]) in Figure 2, it is clear that IMKPL assigns more prototypes to classes which suffer from significant overlapping (e.g., *pick up* and *carry*) and fewer representatives to the more condensed classes (e.g, *sit own* and *stand up*).

Synthetic Dataset

In order to illustrate the feature selection performance of the IMKPL algorithm, a 4-classes dataset of multi-variant time-series is designed using variations of simple 1-dimensional curves. As depicted in Fig.3, the first 5 features (rows) in the i-th data exemplar (i-th column) follow a specific pattern related to class i. However for each class, f_7 is the replicate of f_6 with slight variations, and the last two features (f_8, f_9) are identical through all the samples.

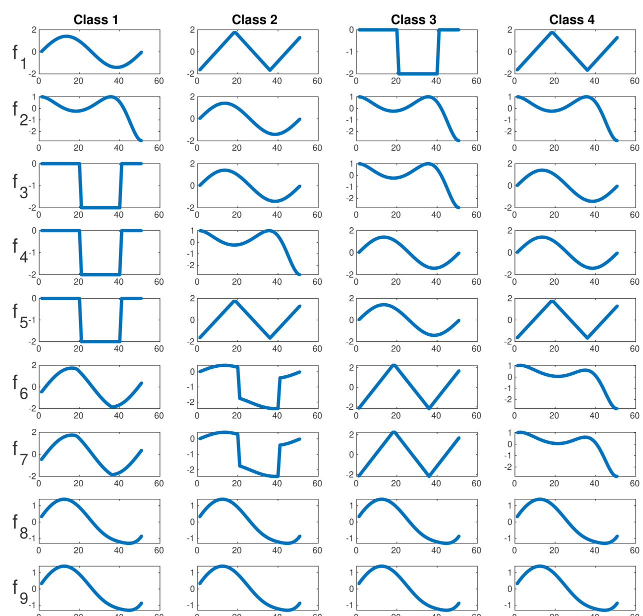

Figure 3: Four classes of synthetic multi-variant time-series. Column i: A time-series exemplar from the i-th class. Row i: The i-th feature of the time-series.

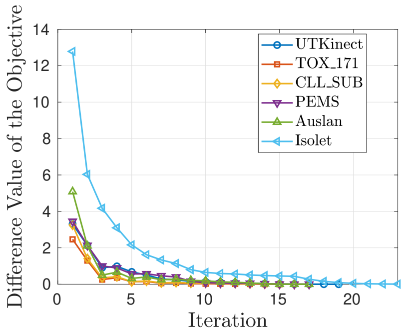

Figure 4: Convergence of IMKPL on the selected datasets.

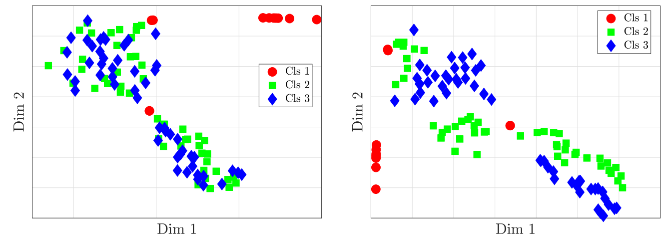

Figure 5: Visualization of class overlapping for the `CLL_SUB` dataset based on the average-kernel combination (left) and the optimized $\vec{\beta}$-combined embedding (right). Clearly, the kernel weightings reduced the overlapping between the classes.

As a result, 9 individual kernel functions $\mathcal{K}_i$ are computed related to each feature f_i. Despite the simplicity of the dataset for the classification task, we are interested to study the performance of IMKPL regarding final feature weightings.

After application of IMKPL, the data is classified with 100% accuracy with the following $\vec{\beta}$:

$$\vec{\beta} = [0,\ 0,\ 0.68,\ 0.50,\ 0,\ 0.54,\ 0,\ 0,\ 0]^\top$$

As a result, the last two identical features $\{f_8, f_9\}$ were ruled out as they were totally irrelevant to the discriminative LMK objective. However, these two features could be ideal choices to have a small reconstruction term $\mathcal{J}_{rec}$ in Eq. (4). Similarly, the weight of f_7 is 0 in $\vec{\beta}$ as its kernel function is similar to $\mathcal{K}_6$, however, $\mathcal{K}_6$ along with 2 other features $\{f_3, f_4\}$ are remained as they are determined to be useful for the discriminative representation. Features $\{f_1, f_2\}$ are removed due to the redundancy of $\{\mathcal{K}_1, \mathcal{K}_2\}$ comparing to others in providing a locally-separable representation. This experiment demonstrates how IMKPL can decide on the importance of the features based on their role in having a discriminative representation.

Convergence Curve of IMKPL

In Figure 4, we plot the the difference value of the whole objective function in Eq. (paper-4) as the difference between the objective value in each iteration and its value in the previous iteration. Based on this figure, the Algorithm (paper-1) is considered converged when the above value becomes relatively small, which occurs rapidly on all the selected datasets in the experiments (less than 20 iterations).

Visualization of the Learned Kernel

To visualize the effect of the learned kernel weights ($\vec{\beta}$) on the distribution of classes, we visualized the 2-dimensional embeddings of the `CLL_SUB` dataset in Figure 5 (using the t-SNE method). Clearly, the optimized $\vec{\beta}$ has lead to better local separation of the classes in the resulted RKHS (Figure 5-left) compared to the average-kernel representation of the data ($\vec{\beta} = \vec{1}/f$) in Figure 5-right. This observation complies with the role of $\mathcal{J}_{ls}$ in Eq. (paper-4).

1.5 Non-negative Quadratic Pursuit

The Non-negative Quadratic Pursuit algorithm (NQP) is a part of an under-review journal article. In the following parts, we explain this algorithm and discuss its complexity and convergence.

Consider a quadratic function $f(\vec{\gamma}) := \frac{1}{2}\vec{\gamma}^\top \mathbf{Q}\vec{\gamma} + \vec{c}^\top\vec{\gamma}$, in which $\vec{\gamma} \in \mathbb{R}^n$, $\vec{c} \in \mathbb{R}^n$, and $\mathbf{Q} \in \mathbb{R}^{n\times n}$ is a hermitian positive semidefinite matrix. Non-negative quadratic pursuit algorithm (NQP) is an extended form of the Matching Pursuit problem [Aharon *et al.*, 2006] and is inspired from by [Lee *et al.*, 2006]. Its objective is to approximately minimize $f(\vec{\gamma})$ in an NP-hard optimization problem similar to

$$\begin{aligned} \vec{\gamma} = \quad & \underset{\vec{\gamma}}{\arg\min} \tfrac{1}{2}\vec{\gamma}^\top \mathbf{Q}\vec{\gamma} + \vec{c}^\top\vec{\gamma} \\ & s.t. \quad \|\vec{\gamma}\|_0 \leq T_0\ ,\ \gamma_i \geq 0\ \ \forall i \end{aligned} \tag{5}$$

where at most $T_0 \ll n$ elements from $\vec{\gamma}$ are permitted to be positive while all other elements are forced to be zero.

As presented in Algorithm 1, at each iteration of NQP we compute $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ to guess about the next promising dimension of $\vec{\gamma}$ (denoted as γ_j) which may lead to the biggest decrease in the current value of $f(\vec{\gamma}_{\mathcal{I}})$; where $\mathcal{I}$ denotes the set of currently chosen dimensions of $\vec{\gamma}$ based on the previous iterations. We look for $\vec{\gamma} \geq 0$ solutions, and also the current value of $\vec{\gamma}$ entries for new dimensions are zero; therefore, similar to the Gauss-Southwell rule in coordinate descent optimization [Nesterov, 2012] we choose the dimension j which is related to the smallest negative entry of $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ as

$$j = \underset{j\in S}{\arg\min}\ \vec{q}_j^\top\vec{\gamma} + c_j \qquad s.t.\ \vec{q}_j^\top\vec{\gamma} + c_j < 0 \tag{6}$$

where $\vec{q}_j$ is the j-th column of $\mathbf{Q}$. Then by adding j to $\mathcal{I}$, the resulting unconstrained quadratic problem will be solved using the closed form solution $\vec{\gamma}_{\mathcal{I}} = -{\mathbf{Q}_{\mathcal{I}\mathcal{I}}}^{-1}\vec{c}_{\mathcal{I}}$, and generally we repeat this process until reaching $\|\vec{\gamma}\|_0 = T_0$ criterion. Notation $\mathbf{Q}_{\mathcal{I}\mathcal{I}}$ and $\vec{c}_{\mathcal{I}}$ denote the principal submatrix of $\mathbf{Q}$ and the subvector of $\vec{c}$ respectively corresponding to the set $\mathcal{I}$. In order to preserve non-negativity of the solution $\vec{\gamma}$ in each iteration t of NQP, in case of having a negative entry in $\vec{\gamma}^t_{\mathcal{I}}$, a simple line search is performed between $\vec{\gamma}^t_{\mathcal{I}}$ and $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$. The line search chooses the nearest zero-crossing point to $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$ on the connecting line between $\vec{\gamma}^{(t-1)}_{\mathcal{I}}$ and $\vec{\gamma}^t_{\mathcal{I}}$.

In addition, to reduce the computational cost, we use the Cholesky factorization $\mathbf{Q}_{\mathcal{I}\mathcal{I}} = \mathbf{L}\mathbf{L}^\top$ [Van Loan, 1996] to compute $\vec{\gamma}$ with a back-substitution process.

Furthermore, because matrix $\mathbf{Q}$ in equations (5) is PSD, its principal sub-matrix $\mathbf{Q}_{\mathcal{I}\mathcal{I}}$ should be either PD or PSD theoretically [Johnson and Robinson, 1981], where the first case is a requirement for the Cholesky factorization. However, in practice by choosing $T_0 << rank(Q)$ we have never confronted a singular condition. Nevertheless, to avoid such rare

Algorithm 1 Non-negative Quadratic Pursuit

Parameters: T_0, ϵ : stopping threshold.
Input: $\mathbf{Q} \in \mathbb{R}^{n\times n}, c \in \mathbb{R}^n$ when $f(\vec{\gamma}) = \frac{1}{2}\vec{\gamma}^\top \mathbf{Q}\vec{\gamma} + \vec{c}^\top\vec{\gamma}$.
output: An approximate solution $\vec{\gamma}$.
Initialization: $\vec{\gamma} = 0$, $\mathcal{I} = \{\}$, $\mathcal{S} = \{1, ..., n\}$, $t = 1$.
repeat
  $j = \arg\min_{j\in S} \vec{q}_j^\top\vec{\gamma} + c_j \qquad s.t.\ \vec{q}_j^\top\vec{\gamma} + c_j < 0$
  if $j = \varnothing$ **then** Convergence.
  $\mathcal{I} := \mathcal{I} \cap j$;
  $\vec{q}_{\mathcal{I}j}$:= created via selecting rows $\mathcal{I}$ and column j of matrix $\mathbf{Q}$.
  $\vec{c}_{\mathcal{I}}$:= a subvector of c based on selecting entries $\mathcal{I}$ of vector $\vec{c}$.
  if $t > 1$ **then**
    $v :=$ Solve for v $\{\mathbf{L}v = \vec{q}_{\mathcal{I}j}\}$;
    $\mathbf{L} := \begin{bmatrix} \mathbf{L} & 0 \\ v^\top & \sqrt{q_{jj} - v^\top v} \end{bmatrix}$
  else
    $\mathbf{L} = q_{jj}$
  end if
  $\vec{\gamma}^t_{\mathcal{I}} :=$ Solve for x $\{\mathbf{L}\mathbf{L}^\top x = \vec{c}_{\mathcal{I}}\}$;
  if $\exists j \in \mathbb{N};\ (\gamma_j^t < 0)$ **then**
    $\vec{\gamma}^t_{\mathcal{I}}$:= the nearest zero-crossing to $\vec{\gamma}_{\mathcal{I}}^{(t-1)}$ via a line search.
    $\mathcal{S} := \mathcal{S} - \{$zeros entries in $\vec{\gamma}^t_{\mathcal{I}}\}$
  end if
  $\mathcal{S} := \mathcal{S} - j$
  $t = t + 1$
until $(\mathcal{S} = \{\}) \ \vee\ (\|\vec{\gamma}\|_0 = T_0) \ \vee\ (\frac{1}{2}\vec{\gamma}^\top Q\vec{\gamma} + c^\top\vec{\gamma} < \epsilon)$
Convergence.

conditions, we do a non-singularity check for the selected dimension j which is to have $q_{jj} \neq v^\top v$ right after obtaining v (1st Cholesky step in Algorithm 1). In case the resulted v does not fulfill that condition, we choose another j based on Eq. (6)

The Convergence of NQP

NQP does not guarantee the global optimum as it is a greedy selection of rows/columns of matrix $\mathbf{Q}$ to provide a sparse approximation of the NP-hard problem in Eq. (5); nevertheless, its convergence to a local optimum point is guaranteed. The algorithm consists of 3 main parts:

1. Gradient-based dimension selection
2. Closed form solution
3. Non-negative line search and updating $\mathcal{I}$.

It is clear that the closed-form solution $\vec{\gamma}$ via selecting a negative direction of the gradient $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ always reduces the current value of $f(\vec{\gamma}^t)$ as $\vec{\gamma}^t$ has to be non-negative and initially $\gamma_j = 0$. In addition, The zero-crossing line search in iteration t can guarantee to strictly reduce the value of $f(\vec{\gamma}^{(t-1)})$. It finds a non-negative $\vec{\gamma}^t_{new}$ between the line connecting $\vec{\gamma}_{\mathcal{I}}^{(t-1)}$ to $\vec{\gamma}^t_{\mathcal{I}}$, and since $f(\vec{\gamma})$ is convex, $f(\vec{\gamma}^t_{new}) < f(\vec{\gamma}_{\mathcal{I}}^{(t-1)})$

Consequently, each of the steps guarantees a monotonic decrease in the value of $f(\vec{\gamma})$, therefore if $\|\vec{\gamma}^{(t+i)}\|_0 > \|\vec{\gamma}^{(t)}\|_0 \implies f(\vec{\gamma}^{(t+i)}) < f(\vec{\gamma}^{(t)})$. Also the algorithm structure guarantees that in any iteration t, $\mathcal{I}_t \neq \mathcal{I}_i\ \forall i < t$ meaning that NQP never gets trapped into a loop of repeated dimension selections. Furthermore, we have $\|\vec{\gamma}\|_0 \leq nT$, meaning that the total number of possible selections in $\mathcal{I}$ is bounded. Concluding from the above, the NQP algorithm converges in a limited number of iterations.

The Computational Complexity of NQP

We can calculate computational complexity of NQP by considering its individual steps. Iteration t contains computing $\mathbf{Q}\vec{\gamma} + \vec{c}$ ($nt + t$ operation), finding minimum of $\nabla_{\vec{\gamma}} f(\vec{\gamma})$ w.r.t the negative constraint ($2n$ operations), computing v (t^2 operation for the $t \times t$ back-substitution), computing $\vec{\gamma}^t_{\mathcal{I}}$ (two back-substitutions resulting in $2t^2$ operation), and checking negativity of entries of $\vec{\gamma}^t_{\mathcal{I}}$ along with the probable line-search which has $3t$ operations in total. Computing for all T_0 iterations, the total runtime of NQP is obtained as

$$\mathbf{T}_{NQP} = (n + 4)T_0^2 + 2T_0 n + T_0^3$$

Considering that in practice $T_0 \ll n$, the algorithm's computational complexity is $\mathcal{O}(nT_0^2)$.

References

[Aharon *et al.*, 2006] M. Aharon, M. Elad, and A. Bruckstein. K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[Johnson and Robinson, 1981] Charles R Johnson and Herbert A Robinson. Eigenvalue inequalities for principal submatrices. *Linear Algebra and its Applications*, 37:11–22, 1981.

[Lee *et al.*, 2006] H. Lee, A. Battle, R. Raina, and A. Y. Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[Nesterov, 2012] Yu Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22(2):341–362, 2012.

[van der Maaten and Hinton, 2008] L.J.P. van der Maaten and G.E. Hinton. Visualizing high-dimensional data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008.

[Van Loan, 1996] Charles F Van Loan. Matrix computations (johns hopkins studies in mathematical sciences), 1996.

[Von. Luxburg, 2007] Ulrike Von. Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.