

MOKPE: Drug–target interaction prediction via manifold optimization based kernel preserving embedding

— Supplementary material —

Oğuz C. Binatlı¹ and Mehmet Gönen^{2, 3}

¹Graduate School of Science and Engineering, Koç University, 34450 İstanbul, Turkey

²Department of Industrial Engineering, College of Engineering, Koç University, 34450 İstanbul, Turkey

³School of Medicine, Koç University, 34450 İstanbul, Turkey

S1 Manifold Optimization based Kernel Preserving Embedding

Our data come from two different domains, namely, $\mathcal{D}$ and $\mathcal{T}$, and we are given two sets of objects $\mathbf{D} = \{\mathbf{d}_i \in \mathcal{D}\}_{i=1}^{N_d}$ and $\mathbf{T} = \{\mathbf{t}_i \in \mathcal{T}\}_{i=1}^{N_t}$, corresponding to drugs and targets, respectively. Usually, these objects have vectorial representations (i.e., $\mathcal{D}$ and $\mathcal{T}$ are Euclidean spaces) in practical applications, however, the domains in our work contain non-vectorial structures, such as protein sequence strings for targets. Aiming for having a general notation for both vectorial and non-vectorial data, we assume that the drug–target interactions and drug–drug, target–target similarities are provided with three different scoring functions:

- i. $s_{c,j}^i: \mathcal{D} \times \mathcal{T} \rightarrow \mathbb{R}$ gives the drug–target interaction score between $\mathbf{d}_i$ and $\mathbf{t}_j$,
- ii. $s_{d,j}^i: \mathcal{D} \times \mathcal{D} \rightarrow \mathbb{R}$ gives the drug–drug similarity score between drugs $\mathbf{d}_i$ and $\mathbf{d}_j$,
- iii. $s_{t,j}^i: \mathcal{T} \times \mathcal{T} \rightarrow \mathbb{R}$ gives the target–target similarity score between targets $\mathbf{t}_i$ and $\mathbf{t}_j$.

Hence, we can use the within-domain similarity functions to represent the objects in a vectorial form, which is known as *empirical kernel map* [1, Chapter 2]:

$$\begin{aligned}\mathbf{d}_i &= [s_{d,1}^i \quad s_{d,2}^i \quad \dots \quad s_{d,N_d}^i]^\top \quad \forall i \\ \mathbf{t}_i &= [s_{t,1}^i \quad s_{t,2}^i \quad \dots \quad s_{t,N_t}^i]^\top \quad \forall i.\end{aligned}$$

This technique allows us to introduce non-linearity into the embedding part [2, Chapter 2] even for the models where the objects have corresponding vectorial representations.

We also introduce three index sets, namely, $\mathcal{I}_c = \{(i, j): s_{c,j}^i \text{ is known}\}$, $\mathcal{I}_d = \{(i, j): s_{d,j}^i \text{ is known}\}$, and $\mathcal{I}_t = \{(i, j): s_{t,j}^i \text{ is known}\}$, to represent known information coming from these scoring functions.

We map objects from two different domains into a unified embedding space. The objects in $\mathbf{D}$ and $\mathbf{T}$ are converted into R -dimensional vectors of an Euclidean space, namely, $\mathbf{E}_d = \{\mathbf{e}_{d,i} \in \mathbb{R}^R\}_{i=1}^{N_d}$ and $\mathbf{E}_t = \{\mathbf{e}_{t,i} \in \mathbb{R}^R\}_{i=1}^{N_t}$. However, to be able to do out-of-sample embedding, we can assume linear projections from the input domains to the embedding domain and optimize the model with respect to the projection matrices. The embedding coordinates are formulated as

$$\mathbf{e}_{d,i} = \mathbf{V}_d^\top \mathbf{d}_i \quad \forall i$$

$$\mathbf{e}_{\mathbf{t},i} = \mathbf{V}_{\mathbf{t}}^\top \mathbf{t}_i \quad \forall i,$$

where we assume that both of the objects have vectorial representations (i.e., $\mathcal{D} \in \mathbb{R}^{D_{\mathbf{d}}}$ and $\mathcal{T} \in \mathbb{R}^{D_{\mathbf{t}}}$, where $D_{\mathbf{d}}$ and $D_{\mathbf{t}}$ are subsets of $N_{\mathbf{d}}$ and $N_{\mathbf{t}}$, respectively).

We want to approximate the scoring functions $s_{\mathbf{c},j}^i$, $s_{\mathbf{d},j}^i$, and $s_{\mathbf{t},j}^i$ by three kernel functions, $k_{\mathbf{c},j}^i: \mathbb{R}^R \times \mathbb{R}^R \rightarrow \mathbb{R}$, $k_{\mathbf{d},j}^i: \mathbb{R}^R \times \mathbb{R}^R \rightarrow \mathbb{R}$, and $k_{\mathbf{t},j}^i: \mathbb{R}^R \times \mathbb{R}^R \rightarrow \mathbb{R}$. These three kernel functions in the embedding space have to be differentiable with respect to the projected coordinates for calculating the gradients required for the optimization step. We use the Gaussian kernel (also called the *radial basis function* kernel) in the linear projections from the input domains to the embedding space to capture the local neighborhood information coming from the cross-domain interactions and within-domain similarities. The kernel functions in the embedding space can be written as

$$\begin{aligned} k_{\mathbf{c},j}^i &= \exp\left(-\frac{\|\mathbf{e}_{\mathbf{d},i} - \mathbf{e}_{\mathbf{c},j}\|_2^2}{\sigma_{\mathbf{e}}^2}\right) = \exp(\mathcal{V}_{\mathbf{c},j}^i) \quad \forall(i, j) \\ k_{\mathbf{d},j}^i &= \exp\left(-\frac{\|\mathbf{e}_{\mathbf{d},i} - \mathbf{e}_{\mathbf{d},j}\|_2^2}{\sigma_{\mathbf{e}}^2}\right) = \exp(\mathcal{V}_{\mathbf{d},j}^i) \quad \forall(i, j) \\ k_{\mathbf{t},j}^i &= \exp\left(-\frac{\|\mathbf{e}_{\mathbf{t},i} - \mathbf{e}_{\mathbf{t},j}\|_2^2}{\sigma_{\mathbf{e}}^2}\right) = \exp(\mathcal{V}_{\mathbf{t},j}^i) \quad \forall(i, j), \end{aligned}$$

where $\sigma_{\mathbf{e}} \in \mathbb{R}_{++}$ is the kernel width, and the auxiliary variables, namely, $\mathcal{V}_{\mathbf{c},j}^i$, $\mathcal{V}_{\mathbf{d},j}^i$, and $\mathcal{V}_{\mathbf{t},j}^i$, are used for simplicity.

We propose to preserve the interaction and similarity scores together using a compound loss function:

$$\mathcal{L} = \frac{\lambda_{\mathbf{c}}}{|\mathcal{I}_{\mathbf{c}}|} \sum_{\mathcal{I}_{\mathbf{c}}} (k_{\mathbf{c},j}^i - s_{\mathbf{c},j}^i)^2 + \frac{\lambda_{\mathbf{d}}}{|\mathcal{I}_{\mathbf{d}}|} \sum_{\mathcal{I}_{\mathbf{d}}} (k_{\mathbf{d},j}^i - s_{\mathbf{d},j}^i)^2 + \frac{\lambda_{\mathbf{t}}}{|\mathcal{I}_{\mathbf{t}}|} \sum_{\mathcal{I}_{\mathbf{t}}} (k_{\mathbf{t},j}^i - s_{\mathbf{t},j}^i)^2,$$

where $|\cdot|$ gives the cardinality of the input set. We have separate mean squared error terms, and separate regularization hyperparameters, namely, $\lambda_{\mathbf{c}} \in \mathbb{R}_+$, $\lambda_{\mathbf{d}} \in \mathbb{R}_+$, and $\lambda_{\mathbf{t}} \in \mathbb{R}_+$, to adjust the weights.

The corresponding optimization problem is

$$\begin{aligned} &\text{minimize } \mathcal{L} \\ &\text{with respect to } \mathbf{V}_{\mathbf{d}} \in \mathbb{R}^{D_{\mathbf{d}} \times R}, \mathbf{V}_{\mathbf{t}} \in \mathbb{R}^{D_{\mathbf{t}} \times R}, \sigma_{\mathbf{e}} \in \mathbb{R}_{++} \\ &\text{subject to } \mathbf{V}_{\mathbf{d}}^\top \mathbf{V}_{\mathbf{d}} = \mathbf{I}_R, \mathbf{V}_{\mathbf{t}}^\top \mathbf{V}_{\mathbf{t}} = \mathbf{I}_R, \end{aligned}$$

where we assume orthonormality of the embedding dimensions in each domain separately. With this assumption, we prevent the scaling ambiguity caused by heterogeneous objects, and we are able to extract extrapolated information in each dimension of the embedding space. The objective function of the optimization problem is non-convex due to the non-linearity introduced by the Gaussian kernels and global optimization is not possible with known methods. Therefore, we employ a Quasi-Newton method based framework to find a locally optimal solution. In our method, we need to satisfy the orthonormality constraints on the embedding space and the non-negativity constraint on the kernel width.

We can write the gradients of $\mathcal{L}$ with respect to the projection matrices as:

$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial \mathbf{v}_{\mathbf{d},p}} &= 2 \frac{\lambda_{\mathbf{c}}}{|\mathcal{I}_{\mathbf{c}}|} \sum_{\mathcal{I}_{\mathbf{c}}} k_{\mathbf{c},j}^i (k_{\mathbf{c},j}^i - s_{\mathbf{c},j}^i) \frac{\partial \mathcal{V}_{\mathbf{c},j}^i}{\partial \mathbf{v}_{\mathbf{d},p}} + 2 \frac{\lambda_{\mathbf{d}}}{|\mathcal{I}_{\mathbf{d}}|} \sum_{\mathcal{I}_{\mathbf{d}}} k_{\mathbf{d},j}^i (k_{\mathbf{d},j}^i - s_{\mathbf{d},j}^i) \frac{\partial \mathcal{V}_{\mathbf{d},j}^i}{\partial \mathbf{v}_{\mathbf{d},p}} \quad \forall p \\ \frac{\partial \mathcal{L}}{\partial \mathbf{v}_{\mathbf{t},p}} &= 2 \frac{\lambda_{\mathbf{c}}}{|\mathcal{I}_{\mathbf{c}}|} \sum_{\mathcal{I}_{\mathbf{c}}} k_{\mathbf{c},j}^i (k_{\mathbf{c},j}^i - s_{\mathbf{c},j}^i) \frac{\partial \mathcal{V}_{\mathbf{c},j}^i}{\partial \mathbf{v}_{\mathbf{t},p}} + 2 \frac{\lambda_{\mathbf{t}}}{|\mathcal{I}_{\mathbf{t}}|} \sum_{\mathcal{I}_{\mathbf{t}}} k_{\mathbf{t},j}^i (k_{\mathbf{t},j}^i - s_{\mathbf{t},j}^i) \frac{\partial \mathcal{V}_{\mathbf{t},j}^i}{\partial \mathbf{v}_{\mathbf{t},p}} \quad \forall p, \end{aligned}$$

where the gradients of the auxiliary variables are found as

$$\begin{aligned}
\frac{\partial \mathcal{V}_{c,j}^i}{\partial \mathbf{v}_{d,p}} &= -\frac{2\mathbf{d}_i(\mathbf{v}_{d,p}^\top \mathbf{d}_i - \mathbf{v}_{t,p}^\top \mathbf{t}_j)}{\sigma_e^2} & \forall p \\
\frac{\partial \mathcal{V}_{d,j}^i}{\partial \mathbf{v}_{d,p}} &= -\frac{2(\mathbf{d}_i - \mathbf{d}_j)(\mathbf{v}_{d,p}^\top \mathbf{d}_i - \mathbf{v}_{d,p}^\top \mathbf{d}_j)}{\sigma_e^2} & \forall p \\
\frac{\partial \mathcal{V}_{c,j}^i}{\partial \mathbf{v}_{t,p}} &= -\frac{2\mathbf{t}_j(\mathbf{v}_{t,p}^\top \mathbf{t}_j - \mathbf{v}_{d,p}^\top \mathbf{d}_i)}{\sigma_e^2} & \forall p \\
\frac{\partial \mathcal{V}_{t,j}^i}{\partial \mathbf{v}_{t,p}} &= -\frac{2(\mathbf{t}_i - \mathbf{t}_j)(\mathbf{v}_{t,p}^\top \mathbf{t}_i - \mathbf{v}_{t,p}^\top \mathbf{t}_j)}{\sigma_e^2} & \forall p.
\end{aligned}$$

Because of the orthonormality constraints, the embedding coordinates of each domain are defined on a Stiefel manifold (i.e., $\mathcal{S}(R, N) = \{\mathbf{E} \in \mathbb{R}^{R \times N} : \mathbf{E}\mathbf{E}^\top = \mathbf{I}_R\}$), which is a Riemannian manifold [3]. To satisfy these orthonormality constraints, we need to use a method proved efficient for optimization on Stiefel manifolds to update the embedding coordinates [4].

When learning the kernel width, we apply the *change of variable* technique to work on the logarithmic scale to satisfy the non-negativity constraint. We define a new variable for the logarithm of the kernel width ($\eta_e = \log \sigma_e$) and use an optimization method on this variable. The gradient of $\mathcal{L}$ with respect to η_e are

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \eta_e} &= 2\frac{\lambda_c}{|\mathcal{I}_c|} \sum_{\mathcal{I}_c} k_{c,j}^i (k_{c,j}^i - s_{c,j}^i) \frac{\partial \mathcal{V}_{c,j}^i}{\partial \eta_e} \\
&\quad + 2\frac{\lambda_d}{|\mathcal{I}_d|} \sum_{\mathcal{I}_d} k_{d,j}^i (k_{d,j}^i - s_{d,j}^i) \frac{\partial \mathcal{V}_{d,j}^i}{\partial \eta_e} \\
&\quad + 2\frac{\lambda_t}{|\mathcal{I}_t|} \sum_{\mathcal{I}_t} k_{t,j}^i (k_{t,j}^i - s_{t,j}^i) \frac{\partial \mathcal{V}_{t,j}^i}{\partial \eta_e},
\end{aligned}$$

where the gradients of the auxiliary variables are found as

$$\begin{aligned}
\frac{\partial \mathcal{V}_{c,j}^i}{\partial \eta_e} &= \frac{2\|\mathbf{e}_{d,i} - \mathbf{e}_{t,j}\|_2^2}{\sigma_e^2} \\
\frac{\partial \mathcal{V}_{d,j}^i}{\partial \eta_e} &= \frac{2\|\mathbf{e}_{d,i} - \mathbf{e}_{d,j}\|_2^2}{\sigma_e^2} \\
\frac{\partial \mathcal{V}_{t,j}^i}{\partial \eta_e} &= \frac{2\|\mathbf{e}_{t,i} - \mathbf{e}_{t,j}\|_2^2}{\sigma_e^2}.
\end{aligned}$$

Our complete algorithm is an alternating optimization scheme consisting of three main steps:

- i. update $\mathbf{V}_d$ given $\mathbf{V}_t$ and σ_e ,
- ii. update $\mathbf{V}_t$ given $\mathbf{V}_d$ and σ_e , and
- iii. update σ_e given $\mathbf{V}_d$ and $\mathbf{V}_t$.

The optimization procedure sequentially updates the decision variables until the stopping criteria are satisfied. Algorithm 1 summarizes the training procedure of our proposed framework.

Algorithm 1 Manifold Optimization based Kernel Preserving Embedding

Input: The scoring functions $s_{c,j}^i$, $s_{d,j}^i$, and $s_{t,j}^i$, the index sets, $\mathcal{I}_c$, $\mathcal{I}_d$, and $\mathcal{I}_t$, subspace dimensionality R , regularization parameters $\lambda_c, \lambda_d, \lambda_t$, relative tolerance parameters ϵ_d, ϵ_t

- 1: Initialize $\mathbf{V}_d \in \text{St}(D_d, R), \mathbf{V}_t \in \text{St}(D_t, R)$ randomly with QR factorization, set $\sigma_e = \sqrt{R}$
 - 2: Calculate the gradients of $\mathcal{L}$ with respect to the projection matrices and logarithm of the kernel width: $\frac{\partial \mathcal{L}}{\partial \mathbf{v}_{d,p}}, \frac{\partial \mathcal{L}}{\partial \mathbf{v}_{t,p}}, \frac{\partial \mathcal{L}}{\partial \eta_e}$
 - 3: **repeat**
 - 4: **repeat**
 - 5: update $\mathbf{V}_d$ by using LRBFGS solver
 - 6: compute kernel functions $k_{c,j}^i$ and $k_{d,j}^i$
 - 7: **until** stopping criterion is met
 - 8: **repeat**
 - 9: update $\mathbf{V}_t$ by using LRBFGS solver
 - 10: compute kernel functions $k_{c,j}^i$ and $k_{t,j}^i$
 - 11: **until** stopping criterion is met
 - 12: **repeat**
 - 13: update σ_e by using $\frac{\partial \mathcal{L}}{\partial \eta_e}$
 - 14: compute kernel functions $k_{c,j}^i, k_{d,j}^i$ and $k_{t,j}^i$
 - 15: **until** stopping criterion is met
 - 16: **until** stopping criterion is met
 - 17: **return** $\mathbf{V}_d, \mathbf{V}_t, \sigma_e$
-

S2 Comparison of MOKPE against the Steepest Descent Method

We compare our algorithm against MKPE [5], a steepest descent based method with Armijo-type line search. We demonstrate the performance of out-of-sample embedding in predicting interactions for unseen drugs. For NR, GPCR, and IC data sets, we apply ten replications of ten-fold cross-validation over drugs to obtain robust results. Since Enzyme data set is much bigger compared to others, achieving convergence within a reasonable timeframe may not be possible for both methods, especially when the subspace dimensionality is larger (i.e., 20 and 25), therefore, it is not included in the comparison. We convert the Matlab implementation of MKPE provided by [5] to R implementation and use it with its default parameters. The code is publicly available at <https://github.com/ocbinatli/mkpe>. We obtain the results of both methods by training them with changing subspace dimensionality parameters taken from $\{5, 10, 15, 20, 25\}$. We use the manifold optimization library **ManifoldOptim** (version 1.0.1) [6] for employing the algorithm LRBFGS in R programming language (version 4.0.2) [7]. We set the regularization parameters $(\lambda_c, \lambda_x, \lambda_z)$ to $(1, 0.1, 0.1)$. For MOKPE, we set relative improvement tolerance parameters (ϵ_x, ϵ_z) to $(10^{-3}, 10^{-3})$ as stopping criteria. We use the default values for all other input parameters. We train both of the methods for equal time amount linearly increasing with respect to the subspace dimensionality. Table S1 shows the maximum allowed training time per fold in seconds. All the experiments are run in the same computer cluster family.

Figure S1 gives the average AUROC (area under the receiver operating characteristic curve) values for MKPE and MOKPE. When the subspace dimensionality is smaller than twenty five, we see that MOKPE achieves significantly higher average AUROC values on all the data sets.

Figure S2 shows the objective function value over time for both methods in training data

Table S1: Training Time per fold (in seconds)

	R (subspace dimensionality)				
Data Set	5	10	15	20	25
NR	10	20	30	40	50
GPCR	300	600	900	1200	1500
IC	400	800	1200	1600	2000

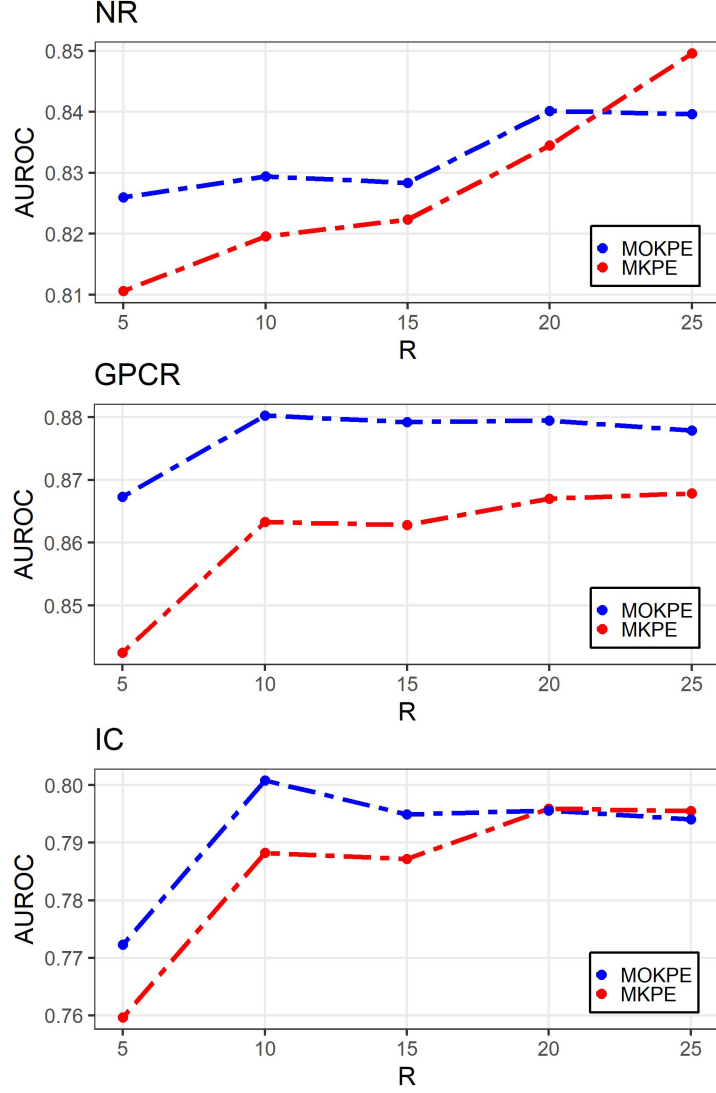


Figure S1: The prediction performances of MKPE and MOKPE with changing subspace dimensionality on the NR, GPCR and IC data sets in terms of average AUROC values.

sets. Each smooth curve is a LOESS (locally estimated scatterplot smoothing) fitted curve. MKPE starts better but more likely to “get stuck in a local optima” later on. Also, the iterations are less costly time-wise for MKPE, which provides an advantage over MOKPE at the starting phase.

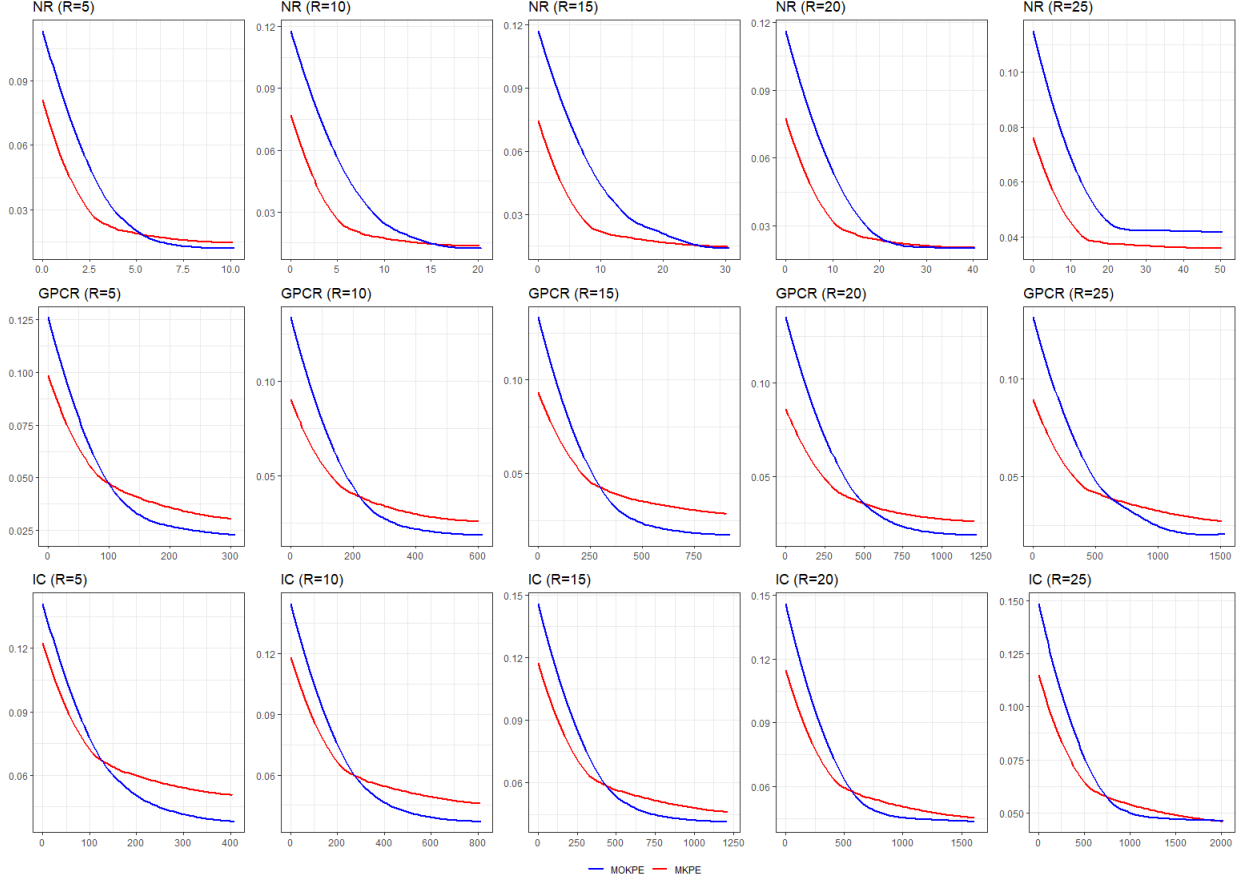


Figure S2: Objective function values vs. processing time (in seconds) for the training sets on MKPE and MOKPE with changing subspace dimensionality on the NR, GPCR and IC data sets.

S3 Parameter Settings

When running the compared methods, we used the reported best hyper-parameters from the sources [8, 9, 10]. The hyper-parameters are as follows:

- LapRLS: $\beta_d = 0.3, \beta_p = 0.3, \gamma_{d1} = 1, \gamma_{p1} = 1, \gamma_{d2} = 0.01, \gamma_{p2} = 0.01$
- DLapRLS: NR data set: regularization terms $\lambda_1 = 0.9, \lambda_2 = 0.9$, number of iterations = 10. GPCR data set: $\lambda_1 = 0.95, \lambda_2 = 0.9$, number of iterations = 10. IC data set: $\lambda_1 = 0.9, \lambda_2 = 1$, number of iterations = 5. Enzyme data set: $\lambda_1 = 0.9, \lambda_2 = 0.9$, number of iterations = 5.
- RLS-WNN: NR data set: regularization term $\sigma = 0.5$, weight $\alpha = 0.5$, kernel bandwidth $\eta = 0.6$. GPCR data set: $\sigma = 0.5, \alpha = 1, \eta = 0.7$. IC data set: $\sigma = 0.25, \alpha = 1, \eta = 0.7$. Enzyme data set: $\sigma = 0.5, \alpha = 1, \eta = 0.7$.
- KBMF2K: NR data set: subspace dimensionality $R = 50$. GPCR data set: $R = 100$. IC data set: $R = 100$. Enzyme data set: $R = 200$.
- CMF: number of iterations = 2, dimensionality of the feature space $k = 100$. NR data set: weight parameters $\lambda_l = 1, \lambda_d = 4, \lambda_t = 32$. GPCR data set: $\lambda_l = 2, \lambda_d = 8, \lambda_t = 0.125$. IC data set: $\lambda_l = 4, \lambda_d = 32, \lambda_t = 0.25$. Enzyme data set: $\lambda_l = 2, \lambda_d = 64, \lambda_t = 0.125$.
- WGRMF: number of iterations = 2, $k = 100$. NR data set: regularization terms $\lambda_l = 0.0625, \lambda_d = 0.05, \lambda_t = 0.1$, number of neighbors $p = 2$. GPCR data set: $\lambda_l = 0.0625,$

$\lambda_d = 0.05$, $\lambda_t = 0.01$, $p = 7$. IC dataset: $\lambda_l = 0.25$, $\lambda_d = 0.15$, $\lambda_t = 0.01$, $p = 6$. Enzyme dataset: $\lambda_l = 0.25$, $\lambda_d = 0.2$, $\lambda_t = 0.2$, $p = 4$.

- GRGMF: number of iterations = 100, the dimension of the subspace expanded by self-representing vectors $K = 150$, number of neighbors $k = 5$, decay term $\eta = 0.5$, matrix regularization parameters $\lambda = 0.0333$, $\beta = 4$, graph regularization parameters $r_1 = 0.5$, $r_2 = 1$, the weight assigned to known link $c = 5$.
- HGBI: the decay factor $\alpha = 0.5$.

References

- [1] Schölkopf B, Tsuda K, Vert JP. Kernel methods in computational biology. MIT press; 2004.
- [2] Schölkopf B, Smola AJ, Bach F, et al. Learning with kernels: support vector machines, regularization, optimization, and beyond. MIT press; 2002.
- [3] Chakraborty R, Vemuri BC. Statistics on the Stiefel manifold: Theory and applications. The Annals of Statistics. 2019;47(1):415–438.
- [4] Boumal N. An introduction to optimization on smooth manifolds. Cambridge University Press; 2023. Available from: <https://www.nicolasboumal.net/book>.
- [5] Gönen M. Embedding heterogeneous data by preserving multiple kernels. In: 21st European Conference on Artificial Intelligence, ECAI 2014. IOS Press; 2014. p. 381–386.
- [6] Martin S, Raim AM, Huang W, Adraghi KP. ManifoldOptim: An R Interface to the ROPTLIB Library for Riemannian Manifold Optimization. Journal of Statistical Software. 2020;93:1–32.
- [7] R Core Team. R: A Language and Environment for Statistical Computing. Vienna, Austria; 2021. Available from: <https://www.R-project.org/>.
- [8] Ezzat A, Wu M, Li XL, Kwok CK. Computational prediction of drug–target interactions using chemogenomic approaches: an empirical survey. Briefings in bioinformatics. 2019;20(4):1337–1357.
- [9] Ding Y, Tang J, Guo F. Identification of drug–target interactions via dual laplacian regularized least squares with multiple kernel fusion. Knowledge-Based Systems. 2020;204:106254.
- [10] Zhang ZC, Zhang XF, Wu M, Ou-Yang L, Zhao XM, Li XL. A graph regularized generalized matrix factorization model for predicting links in biomedical bipartite networks. Bioinformatics. 2020;36(11):3474–3481.