This folder contains the source codes of multi-kernel representation learning in MAMK model, which aims to select representative kernels from the base kernel collection by kernel distance learning and multi-kernel reconstruction for estimating kernel correlation matrix so as to obtain the optimal kernel combination and reformulate multi-kernel attribute representations of adverse drug pairs, which are fed into the designed deep neural network for multi-attribute multi-kernel ADDI prediction. In the experiments, we employ 17 different types of kernels and generate 372 kernels for multi-kernel representation learning. The 17 different types of kernels are Linear, Polynomial, Gaussian, Exponential, Laplacian, Sigmoid, Rational Quadratic, Multi-Quadric, Inverse Multi-Quadric, Log\_1, Log\_2, Cauchy, Chi Square, Histogram, Generalized Histogram, Generalized T-Student 1, and Generalized T-Student 2. The details of adopted 17 types of kernels with their formulations and parameter settings are given in Extended Data Table 1 of the manuscript.

Except for two subfolders .idea and \_\_pycache\_\_ automatically generated by source codes, this folder consists of 24 subfolders, source code files named ‘**KernelFunctionLearning.py**’ and ‘**MultiAttributeRepresentationLearning.py**’, and an immediate result file named ‘**ADDIPredictionResults.txt**’. The detailed information of the files in this folder are described in the following.

Subfolder **AdverseInteraction** stores the ADDI dataset used for experiments. There are a total of 25,495 adverse drug pairs extracted from Twosides associated with 752 drugs and 1307 adverse interactions. Therefore, the number of drugs in file DrugSet.txt is 752 (i.e., 752 lines in **DrugSet.txt**), The number of adverse interactions in file AdverseInteractionSet.txt is 1307 (i.e., 1307 lines in **AdverseInteractionSet.txt**), and the number of adverse drug pairs associated with adverse interactions is 25,495 (i.e., 25,495 lines in **AdverseInteractionSetfromTWOSIDES.txt**). In particular, **AdverseInteractionSetfromTWOSIDES.txt** contains the ADDI dataset presented by the indexes of drugs from 0-751 (the first column in DrugSet.txt) and the vector of adverse interactions with the dimension of 1307, such that if the adverse drug pair can result in the *i*-th adverse interaction (the index of the adverse interaction corresponds to the first column in **AdverseInteractionSet.txt**), then the *i*-th element of the adverse interaction vector in the third column is 1; otherwise, the *i*-th element of the adverse interaction vector is 0. The details descriptions for the files included in this subfolder can be referred to **./ Experimental Dataset\ADDI Data\ReadMe.docx**.

Subfolder **SharedSpecificAttributeRepresentations** stores the shared and specific attribute representations of drugs generated by multi-attribute representation learning. To be specific, P\_CommonAttributeMatrix.npy are the shared attribute representations of drugs, and **Q\_MolecularStructure.npy, Q\_Target.npy, Q\_Pathway.npy, Q\_SideEffect.npy,** **Q\_Phenotype.npy, Q\_Gene.npy,** and **Q\_Disease.npy** are the specific attribute representations of molecular structure, target, pathway, side effect, phenotype, gene, and disease, respectively. These eight files are generated by **MultiAttributeRepresentationLearning.py** in .\Source Codes of MAMK\Multi-Attribute Representation Learning and can be copied from the Subfolder of

.\Source Codes of MAMK\Multi-Attribute Representation Learning \SharedSpecificAttributeRepresentations. However, we only copy the files with suffix ‘.npy’ for the convenience of the implementation for the source codes.

17 subfolders (i.e., **LinearKernelFunction**, **PolynomialKernelFunction**, **GaussianKernelFunction, ExponentialKernelFunction, LaplacianKernelFunction, SigmoidKernelFunction, RationalQuadraticKernelFunction, MultiquadricKernelFunction, InverseMultiquadricKernelFunction, LogKernelFunction1, LogKernelFunction2, CauchyKernelFunction, ChiSquareKernelFunction, HistogramIntersectionKernelFunction, GeneralizedHistogramIntersectionKernelFunction, GeneralizedTStudentKernel1Function, GeneralizedTStudentKernel2Function**) record the kernel matrices for the Kernels of Linear, Polynomial, Gaussian, Exponential, Laplacian, Sigmoid, Rational Quadratic, Multi-Quadric, Inverse Multi-Quadric, Log\_1, Log\_2, Cauchy, Chi Square, Histogram, Generalized Histogram, Generalized T-Student 1, and Generalized T-Student 2, respectively. By feeding the shared attribute representations of an adverse drug pair into the kernel with the corresponding kernel parameters, we can obtain the instantiation of the kernel in terms of shared attribute representation. And by feeding the specific attribute representations of an adverse drug pair into the kernel with the corresponding kernel parameters, we can obtain the instantiation of the kernel in terms of specific attribute representation. Taking Linear Kernel as an example, for each adverse drug pair (*di*, *dj*), we feed the shared attribute representations **P***i*. and **P***j*.into the Linear kernel and calculate the similarly between **P***i*. and **P***j*., and feed the specific attribute representations **Q***i*. and **Q***j*.into the Linear kernel and calculate the similarly between **Q***i*. and **Q***j*. (**Q***i*. and **Q***j*. are the concatenated specific attribute representation of the seven attributes). Thus, we can gain the Linear Kernel matrix in terms of shared and specific attribute representations by respectively feeding the shared attribute representations and specific attribute representations of all adverse drug pairs into Linear Kernel. As a result, LinearKernelFunction contains four files, in which LinearKernelMatrixP.txt and LinearKernelMatrixP.npy are the Kernel matrices of Linear Kernel in terms of shared attribute representations, LinearKernelMatrixQ.txt and LinearKernelMatrixQ.npy are the Kernel matrix of linear kernel in terms of specific attribute representations. The files with suffix ‘.txt’ that can be opened and viewed by notepad the files with suffix ‘.npy’ cannot be opened and viewed directly but only can be loaded by Python codes for the convenience of the implementation for the source codes. Taking Polynomial kernel as another example, as Polynomial kernel contains three parameters, i.e., a, b, and c, where a∈{1, 2, 3, 4, 5}, b∈{0, 1, 2, 3, 4, 5}, and c∈{0.5, 1.0, 1.5, 2.0} (See Extended Data Table 1 in the main text), for each parameter combination (a,b,c), we can generate four files of kernel matrices, i.e., two files (a file with suffix ‘.txt’ and a file with suffix ‘.nyp’) for kernel matrices of Polynomial Kernel in terms of shared attribute representations and two file (a file with suffix ‘.txt’ and a file with suffix ‘.nyp’) for kernel matrices of Polynomial Kernel in terms of specific attribute representations. As there are 5×5×4=120 parameter combinations (a,b,c), we can derive 480 files in Subfolder **PolynomialKernelFunction**.

However, due to that the Kernel matrices for different Kernels occupy so many storage space that we cannot upload all Kernel matrices for 17 different types of Kernels to Github (it is estimated that there are over 13GB for all Kernel matrices for 17 different types). Meanwhile, in the implementation of multi-kernel representation learning, we only need the kernel matrices with the suffix of ‘.npy’ for the convenience of further implementation. With the aim of uploading executable source codes into Github, we calculate the Kernel matrices in terms of 17 types of Kernels with suffix of ‘.npy’ beforehand **Interested readers can implement lines 1620-1686 of KernelFunctionLearning.py** for generating 372 Kernel Matrices of 17 types of kernels with the suffix of ‘txt’, which may take several hours. The ways of generating Kernel Matrices of each type of Kernel are given in the ‘**Note!.txt**’ at the Subfolder of each Kernel.

Subfolder GMatrixListP stores the matrices , *l*=1,2,…,372 for model optimization, where is a symmetric matrix for the *l*-th kernel in terms of shared attribute representation with its entry , where is kernel matrix of the *l*-th kernel in terms of shared attribute representation. Therefore, there are overall 372 matrices with suffix of ‘.npy’ in this Subfolder. Due to that the calculation of matrices , *l*=1,2,…,372 requires several hours, we calculated matrices in advance and only load these matrices in **KernelFunctionLearning.py**. Lines 1723 in **KernelFunctionLearning.py** are the source codes for loading the matrices *l*=1,2,…,372 from Subfolder GMatrixListP. **Interested readers can implement lines 1708-1712 of KernelFunctionLearning.py** for calculating matrices , *l*=1,2,…,372.

Subfolder GMatrixListQ stores the matrices , *l*=1,2,…,372 for model optimization, where is a symmetric matrix for the *l*-th kernel in terms of specific attribute representation with its entry , where is kernel matrix of the *l*-th kernel in terms of specific attribute representation. There are overall 372 matrices with suffix of ‘.npy’ in this Subfolder. Due to that the calculation of matrices , *l*=1,2,…,372 also requires several hours, we calculated matrices beforehand and load these matrices in **KernelFunctionLearning.py**. Lines 1724 in **KernelFunctionLearning.py** are the source codes for loading the matrices *l*=1,2,…,372 from Subfolder GMatrixListQ. **Interested readers can implement lines 1714-1716 of KernelFunctionLearning.py** for calculating matrices , *l*=1,2,…,372.

Subfolder KernelMatricesSimilarity stores the distance among 372 Kernels in terms of shared and specific attribute representations **DP**, **DQ**, which can be calculated by Eq. (8) in the main text. In particular, KernelMatricesPSimilarity.npy and KernelMatricesPSimilarity.txt represent matrix **DQ** that record the distances among 372 Kernels in terms of shared attribute representations, and KernelMatricesQSimilarity.npy and KernelMatricesQSimilarity.txt represent matrix **DQ** that record the distances among 372 kernels in terms of specific attribute representations. Similarly, it also takes several hours to output these four files. Therefore, we also calculated these four matrices beforehand and store them in this Subfolder and load them in **KernelFunctionLearning.py**. Lines 1699-1705 in **KernelFunctionLearning.py** are the source codes for loading the matrices **DP**, **DQ. Interested readers can implement lines 1695-1696 of KernelFunctionLearning.py** for calculating matrices **DP**, **DQ.**

Subfolder LambdaMuAndY\_Matrix stores the estimated kernel correlation matrix **Y** in terms of parameters Lambda (λ) and Mu (µ), where λ is a balance factor regulating the weight of kernel distance learning, and µ controls the row sparsity constraint for matrix **Y.** λ and µ are set from {10-4, 10-3, 10-2, 10-1, 10-0, 101, 102, 103, 104}, and we find that the model achieves the best results when λ=10-3 and µ=10. Thus, we output the corresponding kernel correlation matrix **Y** in terms of λ=10-3 and µ=10, i.e., Lambda=0.001Mu=10Y\_Matrix3.npy and Lambda=0.001Mu=10Y\_Matrix3.txt in this Subfolder. In particular, Lambda=0.001Mu=10Y\_Matrix3.npy is used for selecting representative kernels and as the input for further implementation of the source codes in **KernelFunctionLearning.py.**

Subfolder **DeepNeuralNetwork** stores the trained deep neural network for ADDI prediction. As kernel correlation matrix **Y** is determined by parameters Lambda (λ) and Mu (µ), based on the estimated matrix **Y,** we select the representative kernels by the weights of each kernel. And the selected representative kernels integrated with their probabilities as the representative of all base kernels (i.e., the corresponding rows for the indexes of representative kernels in **Y**) are used to derive the optimal kernel combination for the reformulation of multi-kernel attribute representations shown in Eq. (12) in the supplementary information of this paper. The reformulation of multi-kernel attribute representations can be considered as the inputs of the designed deep neural network for multi-attribute multi-kernel ADDI prediction. In this way, the final deep neural network is not only controlled by Lambda (λ) and Mu (µ), but also determined by the number of kernels selected the representative ones (i.e., variable UsedKernel in **KernelFunctionLearning.py**). Therefore, the deep neural networks with different parameter combinations (Lambda, Mu, UsedKernel) are stored in this Subfolder. We find that the model achieves the best results when the number of representative kernel is 66 (UsedKernel=66) in the condition of Matrix **Y** of Lambda=0.001Mu=10Y\_Matrix3.npy. The best deep neural network with Lambda=10-3, Mu=10, and UsedKernel=66 is stored in this Subfolder for the convenience of further directly loading the network for model testing. In other word, for a testing adverse drug pair, we can directly load this deep neural network to explore its adverse interaction vector based on its reformulation of multi-kernel attribute representations. **Interested readers can further discuss the effects of different parameter combinations**  (Lambda, Mu, UsedKernel) on the deep neural network, which may also take several hours with a large amount of memory to store the trained neural networks.

ADDIPredictionResults.txt records the evaluation metrics of different parameter combination (Lambda, Mu, UsedKernel) in terms of Specificity, Precision, F\_score, Accuracy, AUC, and AUPR. Here, we only store the metrics in terms of Lambda=10-3, Mu=10, and UsedKernel=66

**DeepNeuralNetworkforADDIPredicton.py** is the source codes of the designed deep neural network for multi-attribute multi-kernel ADDI prediction. we utilize an eight-layer full connection neural network, in which the number of neurons for the eight hidden layers is set as {2L, 2L, 1024, 1024, 2048, 2048, K, K} with L and K being the number of base kernels and adverse interactions, respectively.

**KernelFunctionLearning.py** is the source codes of multi-Kernel representation learning for representative kernel selection from the base kernel collection by kernel distance learning and multi-kernel reconstruction for estimating kernel correlation matrix Y so as to obtain the optimal kernel combination and reformulate multi-kernel attribute representations of adverse drug pairs. This file can be implemented directly by ‘Ctrl+B’ in the software of Sublime Text 3. It takes approximately nearly half of an hour to provide the results of the model, as shown in the following:

