Supplementary Material: CA-NN

Hsiang Hsu
Harvard University
hsianghsu@g.harvard.edu

Salman Salamatian
Massachuestts Institute of Technology.
salmansa@mit.edu

Flavio P.Calmon
Harvard University
flavio@seas.harvard.edu

In this supplementary material, we provide details on the experiments setup and on algorithms mentioned in the main text, as well as additional experiments on multi-modal Gaussian data and proofs omitted in the main text.

1 Experimental Details

1.1 Discrete Synthetic Data: Binary Symmetric Channels

Explicit calculation of PICs between two given random variables is challenging in general; however, for some simple cases, e.g. $P_{Y|X}$ given by a so-called discrete memoryless Binary Symmetric Channel (BSC), the PICs can be derived exactly (Calmon et al., 2017, Section 3.5) or (O'Donnell, 2014, Section 2.4). Let X be a binary string of length n, and consider a binary string Y of the same length, where each bit is flipped independently with probability δ. The parameter δ, called the *crossover probability*, captures how noisy the mapping from X to Y is. By symmetry it is sufficient to let $\delta \leq 1/2$. The PICs between X and Y are characterized below: there are $\binom{n}{k}$ PICs of value $(1-2\delta)^k$. For example, for $n=5$ and $\delta = 0.1$, there are $\binom{5}{0} = 1$ PIC of value $(1-0.2)^0 = 1$, $\binom{5}{1} = 5$ PICs of value $(1-0.2)^1 = 0.8$, $\binom{5}{2} = 10$ PICs of value $(1-0.2)^2 = 0.64$, and so on.

For this experiment, we randomly generate 15000 binary strings for training and 1500 strings for testing. The CA-NN is composed of simple neural nets with two hidden layers with ReLU activation, and 32 units per hidden layer. We train over the entire training set for 2000 epochs using a gradient descent optimizer with learning rate 0.01. The approximated PICs for the training and test set, along with the PICs values obtained analytically from theory are shown in Figure 1. The approximated PICs are close to the theoretical values, verifying that the CA-NN is valid in this example.

We also show the factoring planes under different crossover probability δ in Figure 2. When $\delta = 0.1$, most bits are identical between X and Y, while when $\delta = 0.9$ most of the bits are flipped.

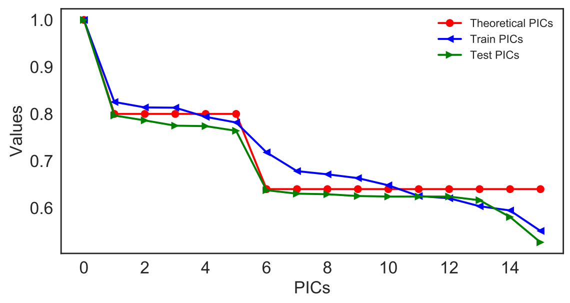


Figure 1: Theoretical and approximated PICs between inputs and outputs of a BSC.

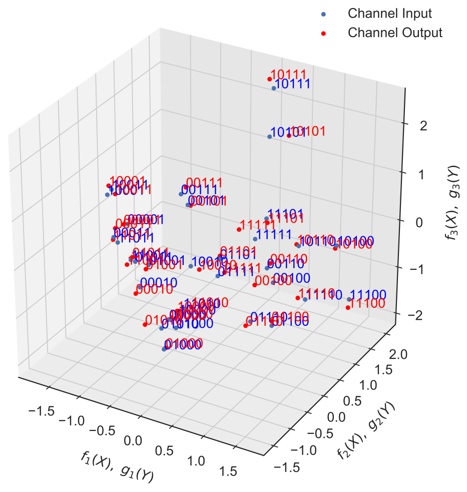
1.2 Gaussian Synthetic Data and Hermite Polynomials

When $\mathcal{X} = \mathcal{Y} = \mathbb{R}$, $X \sim \mathcal{N}(0, \sigma_1)$, $Z \sim \mathcal{N}(0, \sigma_2)$ and $Y = X + Z$, the set of functions $\mathcal{F}$ and $\mathcal{G}$ that give the PICs are the Hermite polynomials (Abbe and Zheng, 2012), where for $x \in \mathbb{R}$, the Hermite polynomial $H_i(x)$ of degree $i \geq 0$ is defined as

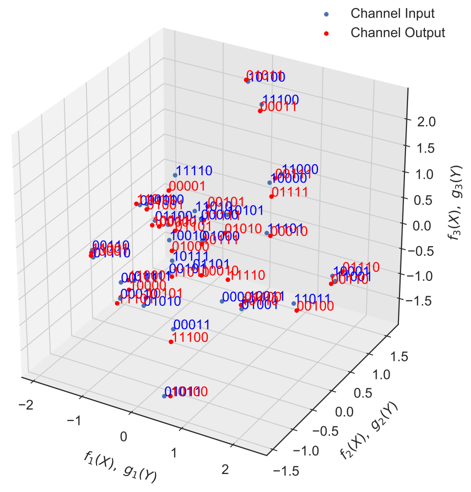
$$H_i(x) \triangleq (-1)^i e^{\frac{x^2}{2}} \frac{d^i}{dx^i} e^{-\frac{x^2}{2}}. \tag{1}$$

More precisely, the i^{th} principal functions f_i and g_i are $H_i^{(\sigma_1)}$ and $H_i^{(\sigma_1+\sigma_2)}$ respectively, where $H_i^{(r)}$ denotes the generalized Hermite polynomial, defined as $H_i^{(r)}(x) = \frac{1}{\sqrt{i!}} H_i(\frac{x}{\sqrt{r}})$, of degree i with respect to the Gaussian distribution $\mathcal{N}(0, r)$, for $r \in (0, \infty)$. The PICs will then be given by the associated inner product $\mathbb{E}[H_i^{(\sigma_1)}(X) H_i^{(\sigma_1+\sigma_2)}(Y)]$.

We pick $\sigma_1 = \sigma_2 = 1$, and generate 5000 training samples for X and Y according to the Gaussian distribution and 1000 test samples. The CA-NN is composed of two hidden layers with hyperbolic tangent activation, 30 units per hidden layer. We train over the entire training set for 8000 epochs using a gradient descent optimizer with learning rate 0.01.



(a) Crossover probability $\delta = 0.1$



(b) Crossover probability $\delta = 0.9$

Figure 2: Three-dimensional factoring planes for the BSC with uniform inputs with different crossover probability δ.

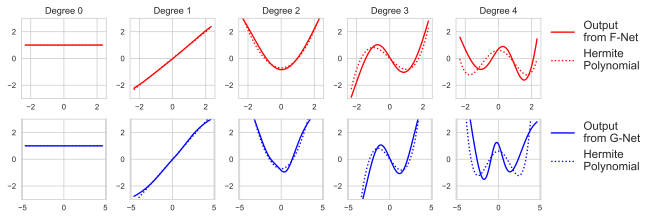


Figure 3: Hermite polynomials of degree 0 to 4 and outputs of the CA-NN that approximate the 0th to 4th principal functions.

In Figure 3, we show the Hermite polynomials of degrees 0 to 4 and the outputs of the CA-NN that approximate the 0th to 4th principal functions. The output of the CA-NN closely recovers the Hermite polynomials; this can be further verified by computing the mean square difference between the approximated principal functions and the Hermite polynomials, i.e.

$$\mathsf{MSE}_f \triangleq \mathbb{E}[(f_i(X) - H_i^{(\sigma_1)}(X))^2], \quad (2)$$

$$\mathsf{MSE}_g \triangleq \mathbb{E}[(g_i(Y) - H_i^{(\sigma_1+\sigma_2)}(Y))^2]. \quad (3)$$

Table 1 provides the mean square difference, as well as the theoretical and estimated PICs. Since the CA-NN approximates the Hermite polynomials, the estimated PICs are also close to their theoretical values.

Table 1: The MSE when using the FG-Net to approximate the principal functions (Hermite polynomials)

	1st	2nd	3rd	4th
MSE_f	0.0001	0.0042	0.0213	0.0522
MSE_g	0.0053	0.0197	0.0238	0.0583
True PICs	0.6977	0.4675	0.2979	0.2113
CorrA-NN	0.7007	0.4938	0.3376	0.2037

1.3 Noisy MNIST Dataset

The noisy MNIST dataset (Wang et al., 2015) consists of 28×28 grayscale handwritten digits, with 60K/10K images for training/testing. Each image is rotated at angles uniformly sampled from $[-\pi/4, \pi/4]$, and random noise uniformly sampled from $[0, 1]$ is added. We let X be those images and Y be the ture labels.

The CA-NN is composed of two neural nets with different structures. Since the inputs of the encoder F-Net are images, we use two convolutional layers with output sizes 32 and 64 with filter dimension 5×5 and max pooling, a fully-connected layer with $1,024$ units, and a readout layer with output size 10. For the G-Net, the inputs are the one-hot encoded labels, and we use two hidden layer with output size 128 and 64, respecively, and a readout layer with output size 10. We adopt ReLU activation for all hidden layers in the

CA-NN.

We train for 200 epochs on the training set with a batch size of 2048 using a gradient descent optimizer with a learning rate of 0.01. To avoid numerical instability, we clip the outputs of the F-Net to the interval $[-10000, 10000]$. Moreover, when back-propagating the objective in (2), we compute $\mathbf{C}_{fg}^{\intercal}(\mathbf{C}_f^{-1} + \epsilon\mathbf{I}_d)\mathbf{C}_{fg}$ instead of $\mathbf{C}_f^{-1/2}\mathbf{C}_{fg}$, where $\epsilon = 0.001$ to avoid an invalid matrix inverse. Using the reconstitution formula (3), we reconstruct the likelihood $p_{Y|X}$ for classification, and obtain an accuracy of 99.76% on the training set, 96.77% on the test set.

The PICs are reported in Table 4, and the factoring planes drawn with the nine principal functions extracted from training and test set are shown in Figure 4 and Figure 5 respectively.

1.4 CIFAR-10 Images

The CIFAR-10 dataset contains 32×32 colored images, each with three channels representing the RGB color model, along with a label representing one of 10 categories. We let X be the images and Y be the labels. In this experiment, the CA-NN is composed of two neural nets with different structures. For the F-Net, we use five convolutional layers with max pooling, two fully-connected layers, and a readout layer. The convolutional layers have output size 128, and the filter dimension is 3×3; the two fully-connected layers have output sizes 384 and 192. The G-Net has the same architecture as the one we use for training over the noisy MNIST, see the previous Section 1.3. We train for 200 epochs with a batch size of 256 using a gradient descent optimizer with learning rate 0.001. The accuracy, once again obtained via classification using the likelihood given by the reconstitution formula in (3), is 93.41% on the training set and 89.75% on the test set. The PICs are reported in Table 5, and the factoring planes of the nine principal functions extracted from training and test set are shown in Figure 6 and Figure 7 respectively, where again each colored point corresponds to an image (X) differentiated by color for each class, and the black point corresponds to the labels (Y).

1.5 Kaggle What's Cooking Recipe Data

We first describe how we pre-processed this dataset. Originally the Kaggle What's Cooking Recipe data contains a list of detailed ingredients for each recipe, along with the type of cuisine the dish corresponds to. We parse the descriptions using Natural Language Toolkit (NLTK) in Python (Bird and Loper, 2004) to tokenize the descriptions into a vector of ingredients

Before	romaine lettuce, black olives, grape tomatoes, garlic, pepper, purple onion, seasoning, garbanzo beans, feta cheese crumbles
After	onion, garlic, pepper, tomato, lettuce, bean

Table 2: Effect of the pre-processing and removal of ingredients on a greek recipe.

for each recipe. Next, we keep only the top 146 most common ingredients and discard the others. This is done for visualization purposes on the factorial planes. The output of this process for an example recipe is shown in Table 2.

The CA-NN is composed of two simple neural nets with 3 hidden layers, with 30 units per hidden layers. Both neural nets adopt hyperpolic tangent activation functions. We train the whole dataset for 20000 epochs by gradient descent optimizer with learning rate 0.005. In addition to the first factoring plane shown in the main text, we illustrate the following two factoring planes in Figure 8 and Figure 9 respectively. Since the PICs of this dataset are large in general, the second and third factoring planes also contain some amount of information. In particular the third principal function allows to separate Indian cuisine from Asian and Western cuisine. Moroccan cuisine is between Indian and Western cuisine on this axis. The fourth principal function separates Asian cuisines into, on one hand Vietnamese and Thai cuisine, and on the other Chinese, Korean and Japanese cuisine. Note that, in this case, there are no signature ingredient, instead it is the entire recipe which helps determining which family of Asian cuisine a dish belongs to.

1.6 UCI Wine Quality Data

The CA-NN is composed of two neural nets with different structures. For the F-Net, we use a simple neural nets with 3 hidden layers, where the numbers of units at each layer are 500, 100, and 30. For the G-Net, we use a simple neural nets with 3 hidden layers, where the numbers of units at each layer are 10, 5, and 3. Both neural nets adopt hyperbolic tangent activation functions. We train the whole dataset for 1000 epochs using an Adam optimizer (Kingma and Ba, 2014) with learning rate 0.001.

The PICs are reported in Table 6, and we illustrate the first two and following two factoring planes in Figure 10 and Figure 11 respectively. Moreover, we plot the minimum and maximum values of the 11 features. In Figure 10, since we have an additional second factoring plane, we observe that the interpolation path

Table 3: Estimating the PICs with different configurations of the CA-NN.

Discrete PICs

	1st PIC	2nd PIC	3rd PIC	4th PIC
Analytic value	0.8000	0.8000	0.8000	0.8000
30-30-25	0.8011	0.7942	0.7918	0.7883
30-30-30-25	0.8272	0.8217	0.8144	0.7926
20-20-15	0.8259	0.8201	0.8195	0.8075
40-30-20-15	0.8363	0.8274	0.8182	0.8020
50-50-30	0.8260	0.8199	0.8193	0.8001
60-50-40-30-20	0.8226	0.8179	0.8079	0.7972

of a low quality and high quality wines does not actually pass through the cluster of medium quality wines. Since there are only two significant PICs in Table 6, we can see that the third and fourth factoring planes in Figure 11 contain barely any information.

1.7 Influence of the Encoder Net Depths

We investigate the influence of different configurations of the encoders F and G Nets on the estimation of the PICs. Specifically, we adopt the experiment setting in Section 4.1.1, and vary neural network configurations including depth and number of neurons. In Table 3, we summarize the estimation of the principal inertia components and different configurations of the encoders F and G Nets. As we can see deeper encoders are prone to overfit the PICs, while shorter and wider encoders are likely to give more accurate estimations of the PICs.

Algorithm 1 Recovering $\mathbf{F}_n(\mathbf{x}_n)$ and $\mathbf{G}_n(\mathbf{y}_n)$ from $\widetilde{\mathbf{F}}_n(\mathbf{x}_n)$ and $\widetilde{\mathbf{G}}_n(\mathbf{y}_n)$, the output of the FG-Nets.

```
Input: $\widetilde{\mathbf{F}}_n(\mathbf{x}_n)$ and $\widetilde{\mathbf{G}}_n(\mathbf{y}_n)$
Output: Principal functions $\mathbf{F}_n(\mathbf{x}_n)$ and $\mathbf{G}_n(\mathbf{y}_n)$
1: $\widetilde{\mathbf{F}}_n(\mathbf{x}_n) \leftarrow \widetilde{\mathbf{F}}_n(\mathbf{x}_n) - \mathbb{E}\left[\widetilde{\mathbf{F}}_n(\mathbf{x}_n)\right]$,
   $\widetilde{\mathbf{G}}_n(\mathbf{y}_n) \leftarrow \widetilde{\mathbf{G}}_n(\mathbf{y}_n) - \mathbb{E}\left[\widetilde{\mathbf{G}}_n(\mathbf{y}_n)\right]$   ▷ (Remove mean)
2: $\mathbf{U}_f, S_f, \mathbf{V}_f \leftarrow$ SVD of $\frac{1}{n}\widetilde{\mathbf{F}}_n(\mathbf{x}_n)\widetilde{\mathbf{F}}_n(\mathbf{x}_n)^\intercal$,
   $\mathbf{U}_g, S_g, \mathbf{V}_g \leftarrow$ SVD of $\frac{1}{n}\widetilde{\mathbf{G}}_n(\mathbf{y}_n)\widetilde{\mathbf{G}}_n(\mathbf{y}_n)^\intercal$
3: $\mathbf{C}_f^{-1/2} \leftarrow \mathbf{U}_f S_f^{-1/2}\mathbf{V}_f^\intercal$,
   $\mathbf{C}_g^{-1/2} \leftarrow \mathbf{U}_g S_g^{-1/2}\mathbf{V}_g^\intercal$   ▷ (Find inverse)
4: $\mathbf{L} = \frac{1}{n}(\mathbf{C}_f^{-1/2}\widetilde{\mathbf{F}}_n(\mathbf{x}_n))(\mathbf{C}_g^{-1/2}\widetilde{\mathbf{G}}_n(\mathbf{x}_n))^\intercal$
5: $\mathbf{U}, S, \mathbf{V} \leftarrow$ SVD of $\mathbf{L}$   ▷ (Find singular vectors)
6: $\mathbf{A} = \mathbf{U}^\intercal\mathbf{C}_f^{-1/2}$, $\mathbf{B} = \mathbf{V}^\intercal\mathbf{C}_g^{-1/2}$
7: return $\mathbf{A}\widetilde{\mathbf{F}}_n(\mathbf{x}_n)$, $\mathbf{B}\widetilde{\mathbf{G}}_n(\mathbf{y}_n)$
```

2 Algorithms

Algorithm 1 summarizes how to convert the outputs $\widetilde{\mathbf{F}}_n(\mathbf{x}_n)$ and $\widetilde{\mathbf{G}}_n(\mathbf{y}_n)$ of the CA-NN to the principal functions by the whitening processing.

Table 4: The PICs of training and test sets for noisy MNIST.

PICs	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
Training	0.989	0.987	0.987	0.985	0.982	0.981	0.979	0.978	0.976
Test	0.957	0.945	0.944	0.927	0.925	0.924	0.921	0.917	0.903

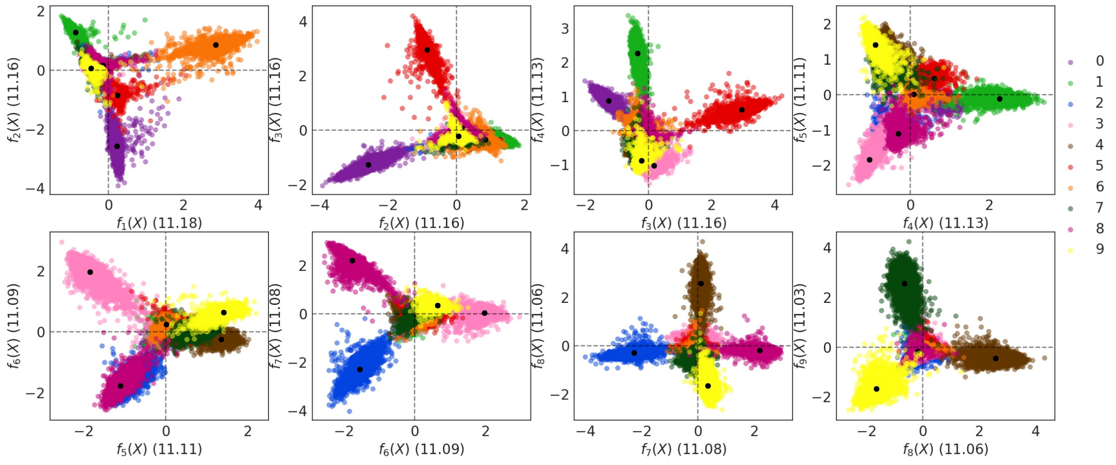


Figure 4: Factoring planes of noisy MNIST on training set.

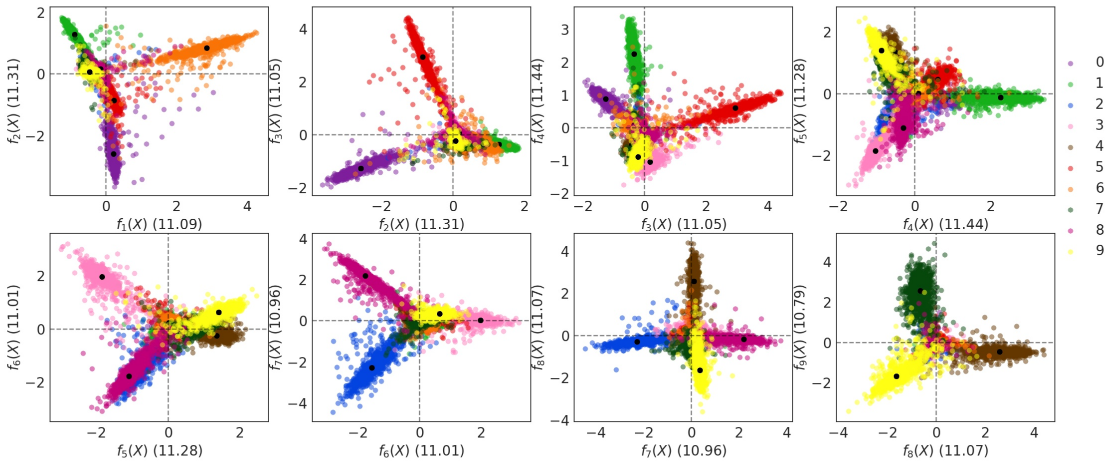


Figure 5: Factoring planes of noisy MNIST on test set.

Table 5: The PICs of training and test sets for CIFAR-10.

PICs	1st	2nd	3rd	4th	5th	6th	7th	8th	9th
Training	0.996	0.996	0.996	0.995	0.995	0.994	0.994	0.994	0.993
Test	0.837	0.800	0.752	0.746	0.739	0.722	0.584	0.562	0.487

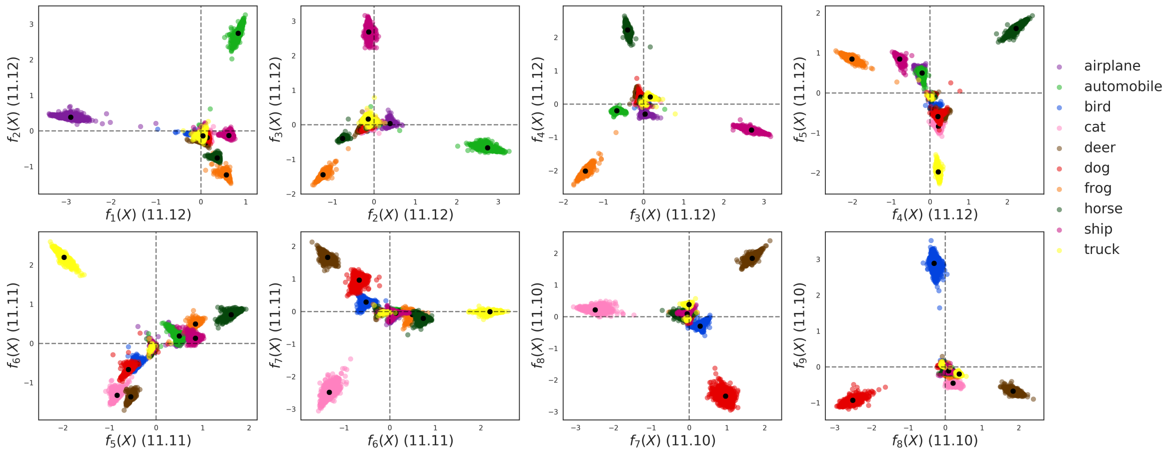


Figure 6: Factoring planes of CIFAR-10 on training set.

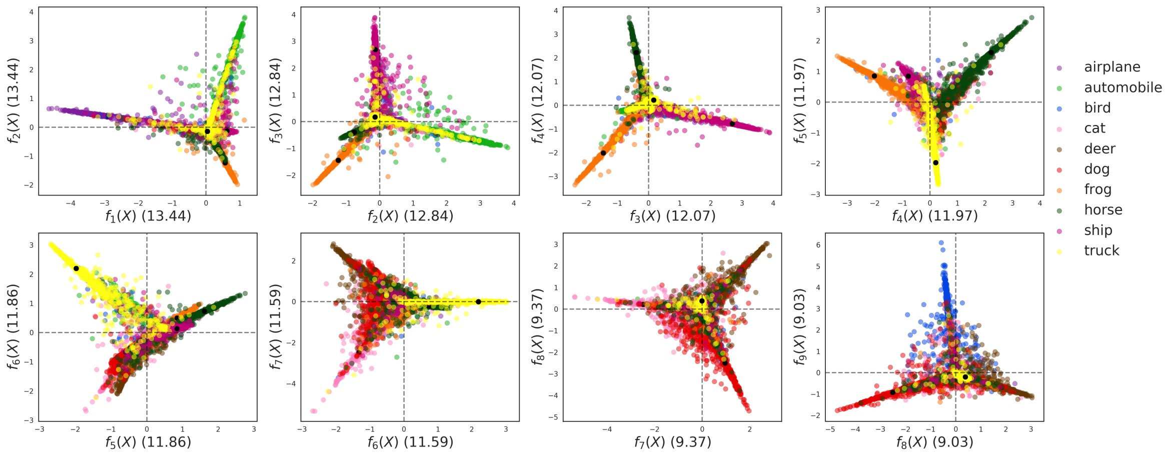


Figure 7: Factoring planes of CIFAR-10 on test set.

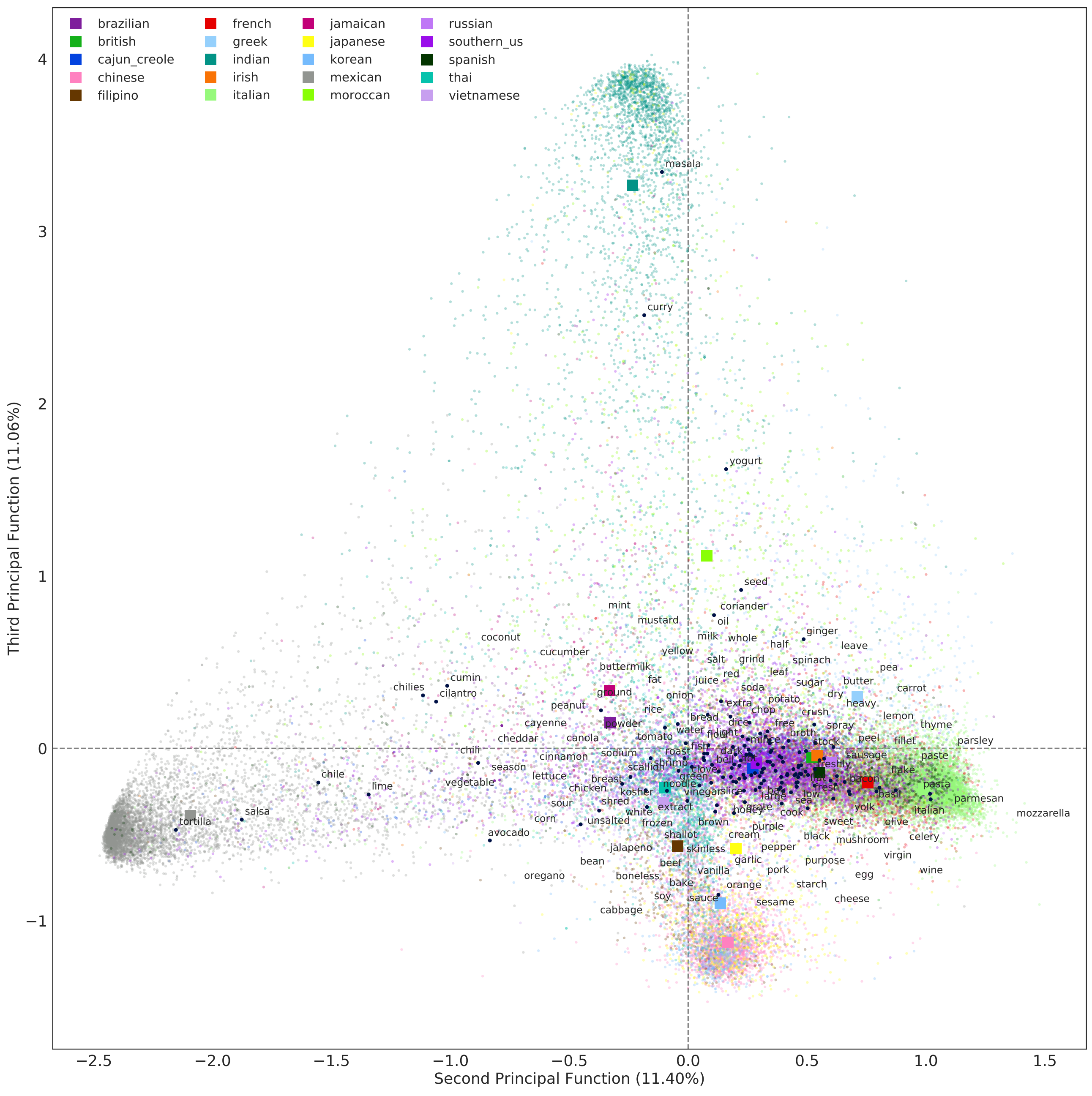


Figure 8: The second factoring plane of CA on Kaggle What's cooking dataset (Colored dots: recipe, dark blue: ingredient).

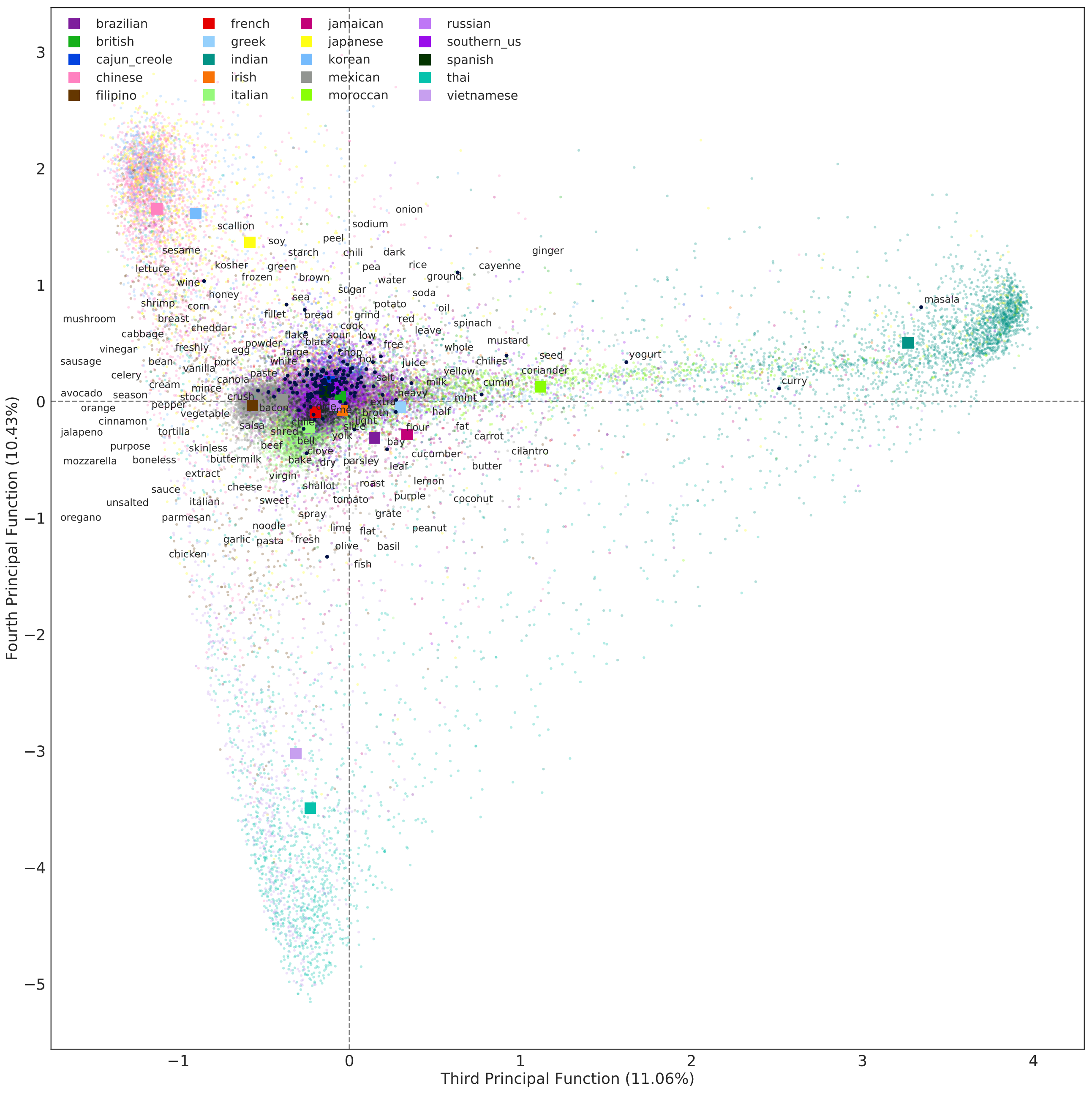


Figure 9: The third factoring plane of CA on Kaggle What's cooking dataset (Colored dots: recipe, dark blue: ingredient).

Table 6: The PICs of training and test sets for UCI Wine Quality Data.

PICs	1st	2nd	3rd	4th	5th	6th
Training	$9.9815e-01$	$9.9353e-01$	$5.6861e-02$	$2.6282e-04$	$2.0870e-06$	$1.9238e-27$
Test	$9.9984e-01$	$6.1934e-01$	$8.8158e-02$	$2.8603e-04$	$7.7783e-08$	$1.4357e-15$

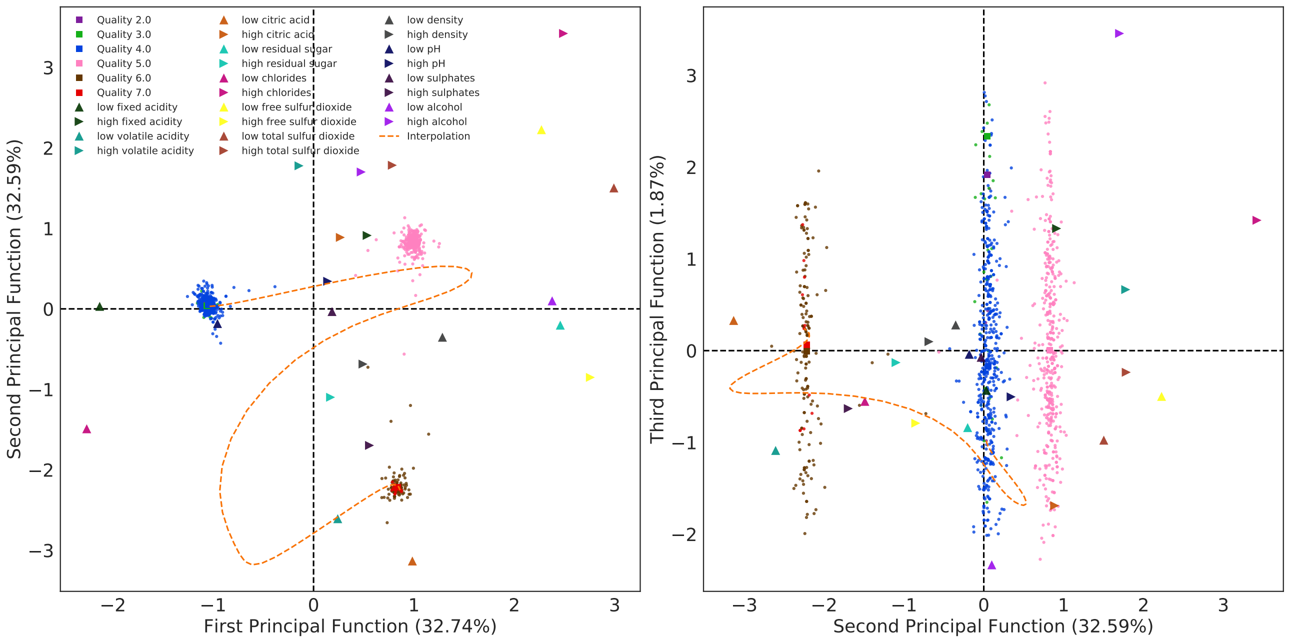


Figure 10: The first (left) and second (right) factoring plane of CA on UCI wine quality dataset.

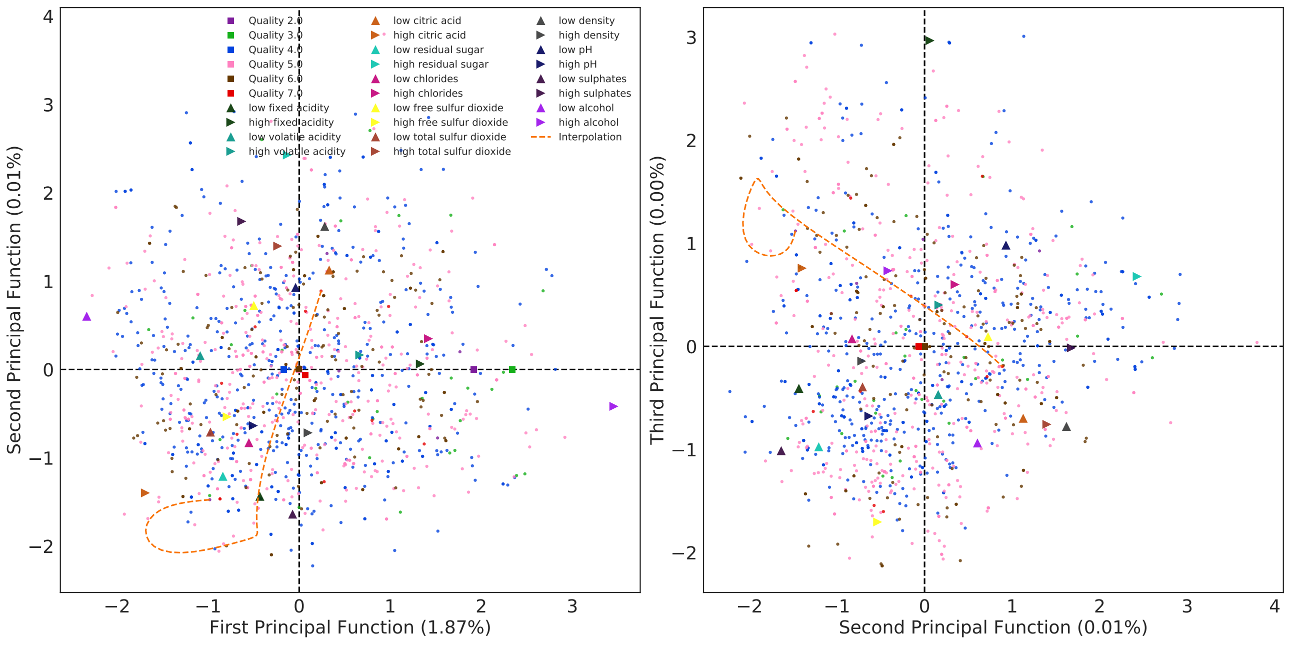


Figure 11: The third (left) and fourth (right) factoring plane of CA on UCI wine quality dataset.

3 Additional Experiment - Multi-Modal Gaussian

As a final set of experiments on synthetic data, we consider mixtures of Gaussian (or multi-modal Gaussian) random variables. More precisely, for $\mu_i \in \mathbb{R}^2, i = 0, 1$, we let $(X, Y) = \mathbf{1}(B = 0)\mathcal{N}(\mu_0, \Sigma) + \mathbf{1}(B = 1)\mathcal{N}(\mu_1, \Sigma)$, where $B \sim \text{Ber}(p)$, and $\mathcal{N}(\mu_i, \Sigma)$ are 2-dimensional multivariate Gaussian random variables with mean μ_i and covariance matrix Σ independent of B. In this experiment, we demonstrate the power of the PICs as a fine representation of the relationship between X and Y. In particular, letting Σ have diagonal elements 1 and off-diagonal elements .7, and letting $\mu_i = (-1)^i[5, 5]^T$, we obtain two modes, one at $[-5, -5]$ and the other at $[5, 5]$. First, note that a general measure of dependence such as Mutual information, would be unable to capture the existence of two modes. In fact, one can verify that the mean-zero jointly Gaussian pair $(\widetilde{X}, \widetilde{Y})$ which has correlation .93 satisfy $I(X; Y) = I(\widetilde{X}, \widetilde{Y}) \approx 1.03$ nats. Despite this, the relationship between X and Y is different from the relationship between $\widetilde{X}$ and $\widetilde{Y}$, as exhibited by the principal functions Fig. 12. Specifically, note that the first principal function distinguishes between the two modes. The second and third principal functions capture the two dimensional space of piece wise linear-function, where each mode follows a separate linear function. When it comes to the value of the PICs, we see that the top PIC is very close to 1, while the top PIC of $(\widetilde{X}, \widetilde{Y})$ is given by the correlation, i.e. .93. However, when it comes to estimating linear functions, one can perform better inference over $(\widetilde{X}, \widetilde{Y})$, since the PIC for this family of function is of about .7 in the multi-modal gaussian.

4 Proofs

4.1 Proposition 2

If we write (3) in the main text into matrix form and following the definitions in Section 3.2 in the main text, we have

$$\begin{aligned}
\mathbf{F\Lambda G}^\intercal &= \mathbf{D}_X^{-1}\mathbf{P}_{X,Y}\mathbf{D}_Y^{-1} - \mathbf{1}_{|\mathcal{X}|}\mathbf{1}_{|\mathcal{Y}|}^\intercal && (4)\\
&= \mathbf{D}_X^{-1}(\mathbf{P}_{X,Y} - \mathbf{p}_X\mathbf{p}_Y^\intercal)\mathbf{D}_Y^{-1} && (5)\\
&= \mathbf{D}_X^{-1/2}\mathbf{Q}\mathbf{D}_Y^{-1/2} && (6)\\
&= \mathbf{D}_X^{-1/2}\mathbf{U\Sigma V}^\intercal\mathbf{D}_Y^{-1/2} && (7)\\
&= \mathbf{L\Sigma R}^\intercal, && (8)
\end{aligned}$$

where $[\mathbf{F}]_{i,j} = f_j(i)$, $[\mathbf{G}]_{i,j} = g_j(i)$ and $\mathbf{\Lambda} = \mathsf{diag}(\lambda_0, \cdots, \lambda_d)$. Eq. (4) shows that in discrete case, the principal functions $\mathbf{F}$ and $\mathbf{G}$ are equivalent to the orthogonal factors $\mathbf{L}$ and $\mathbf{R}$ in the CA, and the factoring scores $\mathbf{\Sigma}$ are the same as the PICs $\mathbf{\Lambda}$. The reconstitution formula in (3) actually connects the PICs and correspondence analysis, and enables us to generalize correspondence analysis to continuous variables (Hirschfeld, 1935; Gebelein, 1941).

4.2 Proposition 3

Since the objective (6) in the main text can be expressed as

$$\begin{aligned}
\mathbb{E}[\|\mathbf{A}\widetilde{\mathbf{f}}(X) - \widetilde{\mathbf{g}}(Y)\|_2^2] &= \text{tr}\left(\mathbf{A}\mathbb{E}[\widetilde{\mathbf{f}}(X)\widetilde{\mathbf{f}}(X)^\intercal]\mathbf{A}^\intercal\right)\\
&-2\text{tr}\left(\mathbf{A}\mathbb{E}[\widetilde{\mathbf{f}}(X)\widetilde{\mathbf{g}}(Y)^\intercal]\right) + \left(\mathbb{E}[\|\widetilde{\mathbf{g}}(Y)\|_2^2]\right), \qquad (9)
\end{aligned}$$

we have

$$\mathbb{E}[\|\mathbf{A}\widetilde{\mathbf{f}}(X) - \widetilde{\mathbf{g}}(Y)\|_2^2] = d - 2\text{tr}\left(\mathbf{AC}_{fg}\right) + \mathbb{E}[\|\widetilde{\mathbf{g}}(Y)\|_2^2], \qquad (10)$$

where the last equation comes from the fact that $\text{tr}\left(\mathbf{A}\mathbb{E}[\widetilde{\mathbf{f}}(X)\widetilde{\mathbf{f}}(X)^\intercal]\mathbf{A}^\intercal\right) = \text{tr}\left(\mathbf{I}_d\right) = d$. Since $\mathbf{C}_f$ is positive-definite, $C_f^{-\frac{1}{2}}$ exists, and so does $\mathbf{A} = \widetilde{\mathbf{A}}\mathbf{C}_f^{-\frac{1}{2}}$, and (9) can be alternatively expressed as

$$\begin{aligned}
&\min_{\mathbf{A}\in\mathbb{R}^{d\times d}, \widetilde{\mathbf{f}}, \widetilde{\mathbf{g}}} \; - 2\text{tr}(\widetilde{\mathbf{A}}\mathbf{B}) + \mathbb{E}[\|\widetilde{\mathbf{g}}(Y)\|_2^2]\\
&\text{subject to } \widetilde{\mathbf{A}}\widetilde{\mathbf{A}}^\intercal = \mathbf{I}_d,
\end{aligned} \qquad (11)$$

where $\mathbf{B} = \mathbf{C}_f^{-\frac{1}{2}}\mathbf{C}_{fg}$. The term $\text{tr}(\widetilde{\mathbf{A}}\mathbf{B})$ can be upper bounded by the Von Neumann's trace inequality (Mirsky, 1975),

$$\text{tr}(\widetilde{\mathbf{A}}\mathbf{B}) \leq \sum_{i=1}^{d} \sigma_{\widetilde{\mathbf{A}},i}\sigma_{\mathbf{B},i}, \qquad (12)$$

where $\sigma_{\widetilde{\mathbf{A}},i}$'s and $\sigma_{\mathbf{B},i}$'s are the singular values for $\widetilde{\mathbf{A}}$ and $\mathbf{B}$ respectively. Moreover, the upper bounded can be achieved by solving the orthogonal Procrustes problem (Gower and Dijksterhuis, 2004), and the optimizer is $\widetilde{\mathbf{A}}^* = \mathbf{VU}^\intercal$, where $\mathbf{V}$ and $\mathbf{U}$ are given by the SVD of $\mathbf{B} = \mathbf{U\Sigma_B V}^\intercal$. Therefore,

$$\text{tr}(\widetilde{\mathbf{A}}^*\mathbf{B}) = \text{tr}(\mathbf{VU}^\intercal\mathbf{U\Sigma_B V}^\intercal) = \sum_{i=1}^{d}\sigma_{\mathbf{B},i} \qquad (13)$$

which is the d-th Ky-Fan norm of $\mathbf{B}$. The desired result then follows by simple substitution.

References

Abbe, E. and Zheng, L. (2012). A coordinate system for gaussian networks. *IEEE Transactions on Information Theory*, 58(2):721–733.

Bird, S. and Loper, E. (2004). Nltk: the natural language toolkit. In *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, page 31. Association for Computational Linguistics.

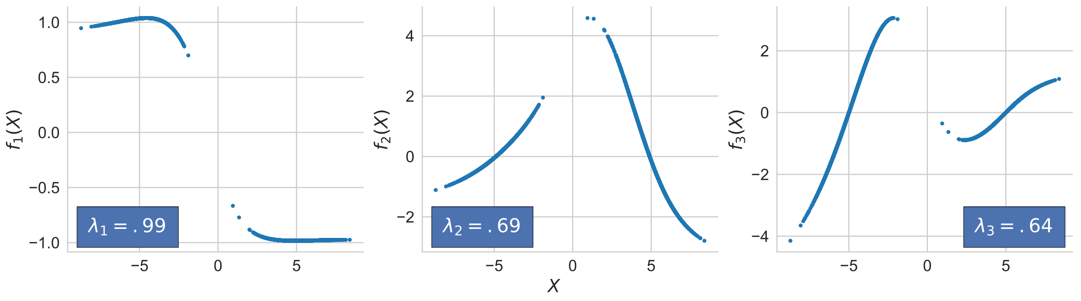


Figure 12: First three principal functions of a multimodal Gaussian, along with the associated PIC values.

Calmon, F. P., Makhdoumi, A., Médard, M., Varia, M., Christiansen, M., and Duffy, K. R. (2017). Principal inertia components and applications. *IEEE Transactions on Information Theory*, 63(8):5011–5038.

Gebelein, H. (1941). Das statistische problem der korrelation als variations-und eigenwertproblem und sein zusammenhang mit der ausgleichsrechnung. *ZAMM-Journal of Applied Mathematics and Mechanics/Zeitschrift für Angewandte Mathematik und Mechanik*, 21(6):364–379.

Gower, J. C. and Dijksterhuis, G. B. (2004). *Procrustes problems*, volume 30. Oxford University Press on Demand.

Hirschfeld, H. O. (1935). A connection between correlation and contingency. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 31, pages 520–524. Cambridge University Press.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Mirsky, L. (1975). A trace inequality of john von neumann. *Monatshefte für mathematik*, 79(4):303–306.

O'Donnell, R. (2014). *Analysis of boolean functions*. Cambridge University Press.

Wang, W., Arora, R., Livescu, K., and Bilmes, J. (2015). On deep multi-view representation learning. In *International Conference on Machine Learning*, pages 1083–1092.