Data Integration on High-Difficulty Binary Classification

Julia Finch, Jesse Hellemn, Kentaro Hoffman, Zhe Zhang

May 1, 2017

Abstract

This paper explores the effectiveness of the data integration technique Generalized Multiple Kernel Learning (GMKL) on high-difficuly binary classification data. GMKL that integrates learned kernels from two disparate data sources is systematically compared to performing GMKL on the two sources naively concatenated as well as standard classification algorithms that are performed on the concatenated sources. The variables that are systematically varied are the number of informative dimensions in each source, the relative information provided by each source, and the number of useless noise dimensions added to each source. It is found that GMKL consistently outperforms its competitors. This paper provides evidence that data integration techniques, specifically GMKL, have the ability to drastically improve upon the performance of naive concatenation.

Introduction

For classification tasks in many fields, especially in medicine and the social sciences, it is increasingly common for data to come from multiple disparate data sources. For example, a sociologist might be interested in predicting future income using two different sources, one on family environment and one on school environment (we will use "source" to refer to a single coherent dataset). For the best analysis and classification results, all data sources should be taken into account. However, most current machine learning classification techniques have been developed for only single dataset inputs, and it is not obvious how to best adapt these techniques to multi-source data.

A naive approach is to concatenate the sources together into one large feature matrix, essentially treating all of the data as a single incoherent source. Although simple, this method throws away information about the separate sources and forces data analysis and classification techniques to treat all of the sources in the same way. Various data integration techniques have been proposed to more cleverly and effectively combine multiple sources. Unfortunately, these techniques have been poorly tested, and there has been no systematic evaluation of their effectiveness.

This paper evaluates one such specialized data integration technique, Generalized Multiple Kernel Learning (GMKL), against traditional classifiers that use concatenated data. We simulate many 2-source datasets with a variety of properties for these comparative tests. We will use the term "classifier" to refer to both specialized data integration techniques such as GMKL as well as naive methods that first concatenate all sources together.

Data Generation

This paper follows the systematic philosophy of simulation studies advocated in [NBM14].

Criteria of Good Simulated Data

In order to understand the behavior of the classifiers on 2-source data as properties of the data are varied, we systematically created data to satisfy all of the criteria detailed below. These criteria allow us to test all the classifiers fairly against each other on consistent benchmarks.

Criteria of Simulated Data

1. The dataset consists of two separate sources.
2. The two classes are separated by a true decision boundary that is known and calculable.
   - or the two classes can be specified to overlap with percentage p, where $p = 0$ leads to no overlap and $p = 50$ leads to complete overlap
3. The decision boundary's complexity (and thus difficulty) can be parametrized and controlled.
4. The reliability of each source can be specified.
5. The noisiness of each source can be specified.
   - extra meaningless dimensions can be added to each source

Explanation for Data Criteria

Suppose that a classifier C only obtains 60% classification accuracy on a dataset D (with datapoints evenly split amongst 2 classes). This could be attributable to either:

- The classifier is not well suited to certain properties of dataset D.
- Eighty percent of both classes overlap with each other. The best possible strategy in this area of overlap is to guess the class with 50% accuracy. An optimal classifier will then guess 40% of the datapoints correctly and also classify the 20% of non-overlapping datapoints perfectly.

Criterion 2 ensures that the latter case does not occur, so that classifier performance is attributable solely to its efficacy on certain types of data.

Criteria 5 is more difficult than it at first seems. In order to create a source with N_{useful} useful dimensions and N_{noisy} meaningless dimensions, we needed to both 1) make N_{noisy} dimensions of pure noise and 2) make N_{useful} dimensions, all of which are always useful. With a random coefficient linear model, it is impossible to verify that all N_{useful} dimensions are actually useful.

Data Generation Models

Random Coefficient Linear Model

A simple, common way to simulate data is to use a linear model with random coefficients. This model generates data by:

1. Specify the number of true latent variables K along with the number of visible variables N
2. Specify two distribution D_j^0 and D_j^1 of each latent variable z_j, $1 \leq j \leq K$, where D_j^0 is the distribution of z_j for negative classes and D_j^1 is the distribution of z_j for positive classes
3. Specify how each visible variable x_j is generated from the latent variables z_i with a formula of the form

$$x_l = \sum_{i=1}^{K} \beta_i z_i + \sum_{i=1}^{K} \sum_{j=i}^{K} \beta_{i,j} z_i z_k + \text{higher-order-interactions}$$

   of linear combinations of arbitrary functions of the latent variables
4. Specify every β in the above formula
5. For every datapoint

(a) Pick which class the datapoint belongs to

(b) Sample each z_j from its respective distribution for this class

(c) Generate each visible variable x_j from its formula

This model has significant shortcomings

- It is not known how to systematically make the classification problem more or less difficult
- It is hard to know if the generated data overlaps
- It is hard to pick the βs to ensure that all of the criteria in are satisfied
- There are many distributions and formulas to specify arbitrarily
- It is hard to know how many of the generated dimensions are useful

Feed-forward Network Model

In order to satisfy all of the criteria in , we created a feed-forward conditional network (Figure 1). This network's process is given in detail Algorithm 1. The network model was inspired by Bayesian Networks; the output of each node is sampled conditionally after that node's inputs ahave all been calculated. The model is highly extensible. For this particular network, each sources reliability, difficulty, noisiness, and relative amount of noise to useful information could all be systematically varied independently of each other. separately.

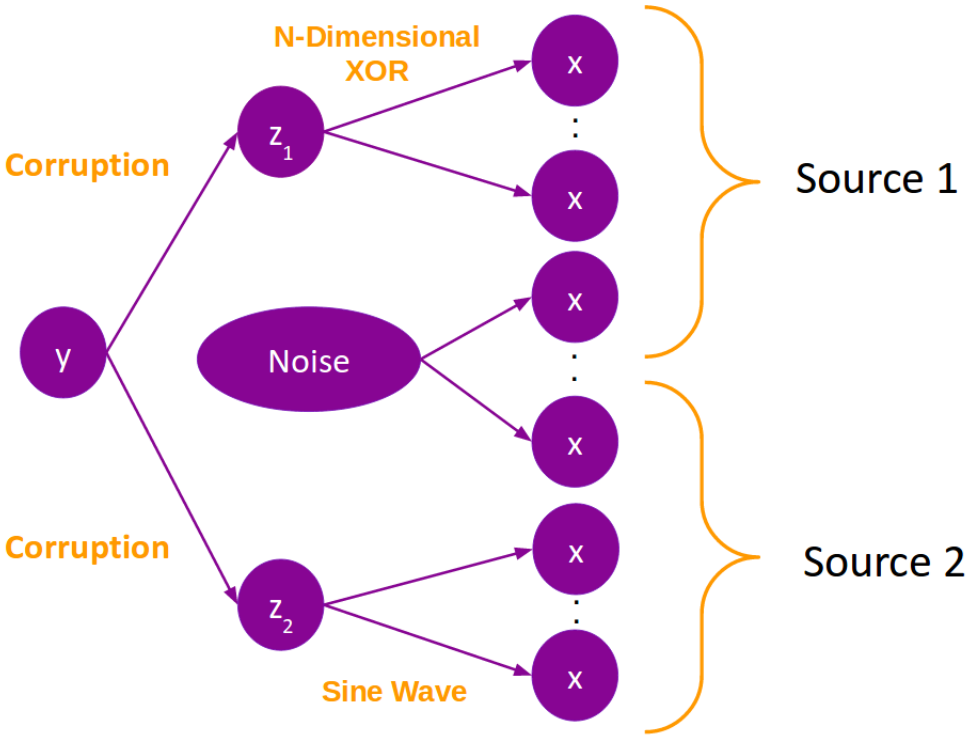

Figure 1: *General schematic of the network data generation model. Every circle is a "node" with takes in an input from the previous layer and outputs a new layer. For example, z_1 will produce a N-dimensional XOR, where the parity depends on the input from y.*

The above model creates two sources of data with very different types of decision boundaries.

The first source is an N-dimensional XOR, which consists of clusters of points at every corner of an N-dimensional binary hypercube, where each corner belongs to a different class than all of its $N-1$ closest

Algorithm 1 Data generation process for the network model

▷ Sample the y layer

$y \leftarrow \text{Bernoulli}(p)$

▷ Sample the z layer

for all $z_i \in \{z_1, z_2\}$ **do**

  $c \leftarrow \text{Bernoulli}(p_i)$ ▷ p_i chance to corrupt source i

  $z_i \leftarrow c * (1 - y) + (1 - c) * y$ ▷ If corrupting, z_i will be 0 if y is 1 and 1 if y is 0

end for

▷ Sample the x layer of source 1, an N_1-dimensional XOR

if z_1 is even **then**

  $x_1^{(1)}...x_{N_1}^{(1)} \leftarrow N_1$-dimensional binary vector of even parity

else

  $x_1^{(1)}...x_{N_1}^{(1)} \leftarrow N_1$-dimensional binary vector of odd parity

end if

▷ Sample the x layer of source 2, a k_2-period sine wave

$x_1^{(2)} \leftarrow \text{Uniform}(-k_2\pi,\, k_2\pi)$

$x_2^{(2)} \leftarrow \text{Uniform}(-k_2\pi,\, k_2\pi)$

$x_3^{(2)} \leftarrow (x_1^{(2)} + x_2^{(2)})sin(x_1^{(2)}) + mz_2$

▷ Add noise

$x^{(1)} \leftarrow x^{(1)} + \text{Normal}(0,\, \sigma_1)$ ▷ Gaussian noise added to each dimension independently for XOR sources

$x_3^{(2)} \leftarrow x_3^{(2)} + \text{Normal}(0,\, \sigma_2)$ ▷ Gaussian noise only added in direction of margin for sine wave sources

neighboring corners. When the input to z_1 is a 1 (representing a positive example), then the output is sampled from all binary 0-1 N-dimensional vectors with an even numbers of ones, also known as an even-parity XOR. Figure 2 shows a 3D example. The complexity of the decision boundary is increases with the number of clusters, and the number of clusters increases exponentially with the dimension of the XOR. Noise is added to an XOR by adding sampling from Normal(0, σ_1) independently for every dimension.

The second source is a sine wave in 3D space (Figure 3); the complexity of this decision boundary is directly proportional to the period of the wave and inversely proportional to the margin m between the two classes. Noise is added to the sine wave only in the direction of the margin (points can only be moved "up" or "down").

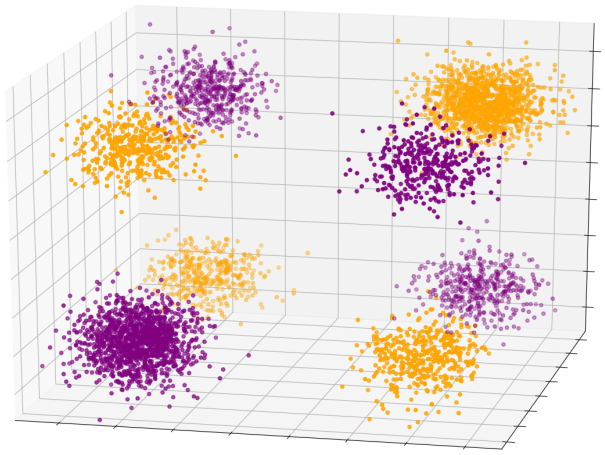

Figure 2: *A 3D-XOR with equal class size and $\sigma_1 = 0.2$ noise. The noise for the XORs created by this model are independent Normal(0, σ_1) added to every dimension.*

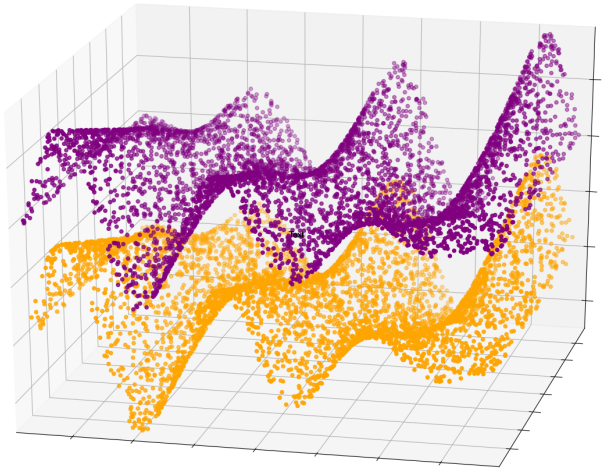

Figure 3: *A 3 period sine wave with $\sigma = 0$ noise and a margin of $m = 10$. The equation of this wave in xyz coordinates is $z = (x + y)sine(x) + mc + Normal(0, \sigma_2)$, where m is the margin and c is the 1 or 0 class label.*

This model has several nice properties:

- For both the N-dimensional XOR and the sine wave (with small enough margin m), every dimension is necessary for perfect classification.
- The complexity of the decision boundary for each source can be easily controlled.
- The reliability of each source can be varied independently of the other.
- The data can be made more noisy (to an extent) without compromising the separability of the two classes.
- Extra meaningless dimensions can easily be added to either source.

The reliability of each source is controlled by corrupting its input signal by a speficied percentage. For example, to decrease the reliability of source 1 to 80%, p_1 is set to 0.1. Then z_1 will only equal y 90% of the time, and will give the wrong signal 10% of the time. Note that 0% reliability is achieved when the signal is corrupted 50% of the time, since the signal is a binary 0 or 1.

The first experiments in this paper use a similar model to the one above, except with another XOR as the second source instead of the sine wave. We refer to this model as a "Double XOR." The final experiment uses the exact model described above.

Classifiers

The data integration method GMKL was tested against standard machine learning classifiers Gaussian Naive Bayes, Random Forest, and an SVM using a RBF kernel. The machine learning classifiers were fed concatenated data, and so ignored separate sources. Additionally, standard GMKL was compared against GMKL that was fed concatenated data, to isolate the effects of using a separate kernel for each data source. A brief overview of every classifier and of GMKL follow.

Where possible, Python's sklearn implementations were used. Classifiers with hyper parameters, such as the SVM, were trained using cross-validation over a suite of possible parameters, as is industry standard.

Notation

In describing different classification techniques, the following notation will be used:

- (x, y) pair denotes a feature vector x $\in \mathbb{R}^m$ and its corresponding target value $\in \{0, 1\}$.
- x_j denotes the jth feature in x.
- there are N training examples and ith training pair is represented by (x^i, y^i).

Classifiers Using Concatenated Data

Gaussian Naive Bayes

Model: Gaussian Naive Bayes assumes that every feature (dimension) x_j is generated independently conditional on the value of y i.e. $P(x|y) = \prod_{j=1}^{m} P(x_j|y)$. In addition it also assumes that each $P(x_j|y) \sim \mathbf{N}(\mu_{\mathbf{j},\mathbf{y}}, \sigma_{\mathbf{y}})$. To predict $\hat{y}$ from a given x, the model computes the posterior probability $P(y|x)$ and pick the more likely case. This is not a very powerful model.

Properties: The formulation is clean and simple. Naive Bayes is very fast to train and is somewhat resistant to extra noise dimensions. However, the assumption of independence between dimensions is very strong, and means that only 2nd degree decision boundaries can be fit. It is expected to perform very poorly on the complex, very non-linear Double XOR dataset.

Implementation: Gaussian Naive Bayes does not have any hyper parameters and did not need any parameter tuning.

Random Forest

Model: Random Forest uses an ensemble of deep decision trees. Each decision tree is based by a bootstrap sample of the training data, and the splitting nodes are constrained to a random subset of all features. In this manner, overfitting of individual decision tree is controlled by ensembling.

Properties: Random Forest is known to be very resistant to extra noise dimensions, since uninformative features are simply never used to split decision tree. Random Forests make no assumptions about the input data, and are unaffected by scaling or large differences in scale between features.

Implementation: We used cross validation to select the optimal number of trees $\bar{t} \in \{10, 100\}$. Other parameters were the defaults in the Python sklearn implementation: the gini index was used as the measure quality of the tree, up to $\sqrt{num\_features}$ features were used for each tree, and trees are split until all nodes are pure or have ≤ 2 points.

K-Nearest Neighbors (KNN)

Model: KNN labels a given point x as the majority vote label of its K-closest points (in the training data) i.e.

$$f(x) = sign(\sum_{x^i \in \text{K closest points}} y^i)$$

Properties: KNN is able to fit extremely complex decision boundaries. However, it often suffers from the curse of high dimensionality; the distance between points is distorted in high dimensions. Importantly, KNN is very affected by meaningless noisy features, as these features factor into the Euclidian distance between points just as much as the useful features.

Implementation: We used cross validation to select the best number of neighbors $\bar{k} \in \{1, 2, 10\}$

Support Vector Machine (SVM) with a Radial-Basis-Function (RBF) Kernel

Model: SVMs find the largest-margin between two classes, using a kernel function to first project the data into a higher-dimensional space. Similar to KNN, SVMs label data points considering the weighted voting of a set of similar data points

$$f(x) = sign(\sum_{i=1}^{N} y^i d^i K(x, x^i)) \text{ where K(x, } x^i) = exp(-\gamma|x - x^i|^2)$$

where d^i is learned from data by finding the largest-margin separating hyperplane in the projected space.

Properties: SVMs can fit very complex decision boundaries, and don't suffer from the curse high dimensionality as much as KNN does. The complexity of the decision boundary learned by the SVM can be controlled with two hyper parameters γ and C (kernel width and regularization strength respectively). These characteristics make SVM the go-to algorithm for many complex classification problems. However, just like KNN, it is unable to effectively discriminate against extra noise dimensions. SVMs also require lots of data points to learn complex decision boundaries.

Implementation: Cross-validation was performed to search for the best regularization parameter $\bar{C} \in \{0.1, 1, 10, 100\}$ and kernel width $\bar{\gamma} \in \{.01, .1, 1, 10\}$.

Generalized Multiple Kernel Learning (GMKL)

GMKL learns separate kernels for each data source before integrating the separate kernels in an optimization step that's based on a global error measure; the resulting combined kernel is then fed into an SVM. Although this method uses a SVM for actual classification, the separate kernels allow each source to have its own representation. A brief description of the algorithm is described below, with a general flowchart of the process shown in Figure 4 and a more detailed explanation of the optimization part in Algorithm 2. A full explanation of the GMKL algorithm can be found in [YXY$^+$11].

Model: The final trained model is almost the same as those from SVM, except that the final kernel function is actually learned from data. More specifically, the final kernel is a convex combination of a set of predefined kernel functions:

$$K(x, x^i) = \sum_{q=1}^{Q} \theta_q * K_q(x, x^i) \text{where } \theta_q \text{is learned from data.}$$

Formulation as Optimization Problem

$$\begin{aligned} &\underset{\theta,v,b}{\text{minimize}} && C\frac{1}{N}\sum_{i=1}^{N} L(f_{\theta,w,b}(x^i), y^i) + 1/2\sum_{q=1}^{Q} |\frac{v_q|^2}{\theta_q} \\ &\text{subject to} && \beta|\theta|_2^2 + (1-\beta)|\theta|_1 \leq 1 \end{aligned}$$

Where C is the regularization parameter, β is the elastic net parameter, L is the hinge loss function and

$$f_{\theta,w,b}(x^i) = \sum_{q=1}^{Q} w_q \phi_q(x^i)\sqrt{\theta_q} + b$$

Now if we define v by $v_q := \frac{w_q}{\sqrt{\theta_q}}$, then we have a convex optimization problem:

$$\begin{aligned} &\underset{\theta,v,b}{\text{minimize}} && C\frac{1}{N}\sum_{i=1}^{N} L(f_{\theta,w,b}(x^i), y^i) + 1/2|w|^2 \\ &\text{subject to} && \beta|\theta|_2^2 + (1-\beta)|\theta|_1 \leq 1 \\ &\text{Where} && f_{\theta,v,b}(x^i) = \sum_{q=1}^{Q} v_q \phi_q(x^i) + b \end{aligned}$$

Observe that the elastic net constraint ensures sparsity and grouping effect for the set of kernels chosen for the final model.

Optimization Algorithm

Algorithm 2 Level method for the MKL[1]

▷ Initialization

1: $t \leftarrow 0$

2: Let θ be uniformly initialized subject to the elastic net constraint

3: **while** difference between the upper bound and lower bound is less than ϵ **do**

4: $\alpha^t = \underset{\alpha}{argmax} D(\theta^{t-1}, \alpha)$ ▷ Solve dual problem

5: $h^t(\theta) = \underset{1\leq i\leq t}{max} D(\theta, \alpha^i)$ ▷ Construct a cutting plane model

6: Calculate a lower bound and an upper bound for the optimal solution $\overline{D_t}, \underline{D_t}$ and an improvement set level set $L = \{\theta : h^t(\theta) \leq$ some convex combination of $\overline{D_t}, \underline{D_t}\}$

7: Project θ^{t-1} to L to obtain θ^t

8: **end while**

Implementation For each 'independent' data source, we constructed 10 RBF kernels with the width $\gamma \in \{2^{-3}, 2^{-2}, 2^{-1}..., 2^6\}$. Then we trained our model with the regularization parameter C fixed to 100.

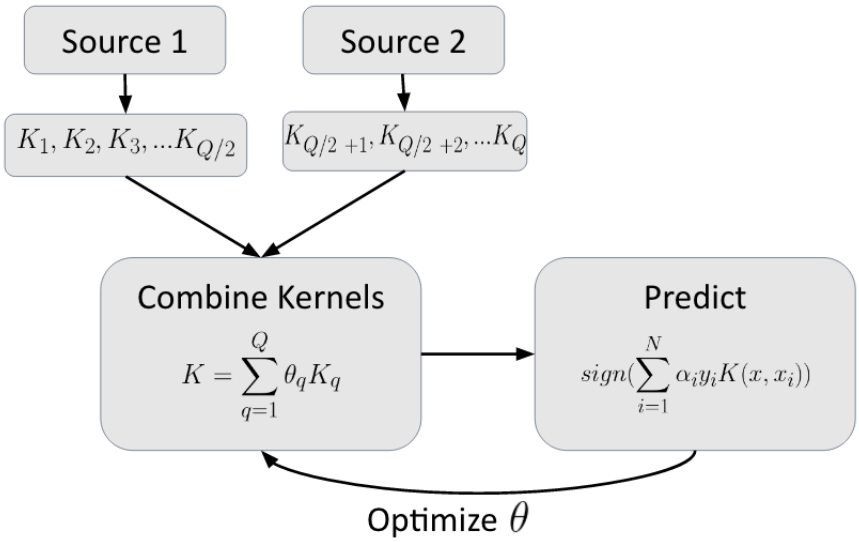

Figure 4: *Flow Chart of Our Model*

Concatenated GMKL To isolate the effect of data integration, we also applied GMKL to concatenated data for comparison. More specifically, we constructed only 10 RBF kernels with $\gamma \in \{2^{-3}, 2^{-2}, 2^{-1}..., 2^{6}\}$ for the concatenated dataset and then trained GMKL algorithm with C fixed at 100.

Experiments

Experiment 1: Data Dimension Scaling

The first experiment used the network data generation model shown in Figure 1, using an XOR for both sources. The two sources always had the same dimension as each other. This first experiment tested GMKL, Concatenated GMKL, SVM, KNN, and Random Forest as the dimensionality of the XOR (and thus the difficulty of the classification task) increased. This test increases to a maximum of 7 dimensions per source. At first glance this may seem like a very small problem, but recall that the XOR has a very complex decision boundary, as evidenced by every classifier's poor performance at even this low XOR dimensionality.

Experiment 1 Parameters

$M_{i,useful}$	$M_{i,noisy}$	N	T	p_1, p_2	σ_i	C	k
[2,3,4,5,6,7]	0	5000	1:2	0,0	0.2	1:1	NA

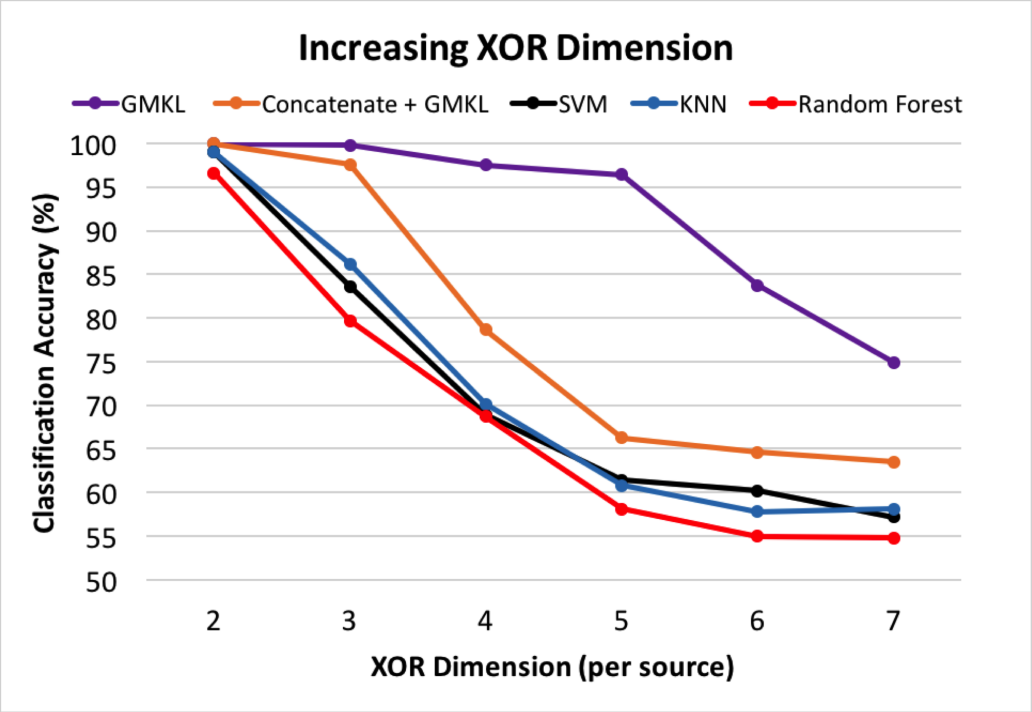

Figure 5: *An illustration of how resistant GMKL is to the affect of an increasingly complex XOR decision boundary compared to the standard classifiers.*

From Figure 5 we can see that as the number of XOR dimensions increases, the classification accuracy of all of the classifiers decreases. This makes sense because as the XOR dimension increases, the decision boundary becomes more complex causing classification to become more difficult. However, while all classifiers have decreasing accuracy, we can see that GMKL performs much better than the other classifiers with nearly a 35 percent better classification accuracy compared to the classical SVM at five dimensions. In fact, there is a very intriguing pattern on display here as the the GMKL at $2n$ dimensions seems to be performing about as well as the classical SVM at n dimensions. This seems to indicate that doing classification on the data sources as separate entities is highly desirable for complicated classification problems.

Experiment 2: Corrupted XOR Sources

Another variable that we decided to vary is the relative importance of the disparate sources. The purpose of this experiment is to evaluate how the various classifiers perform when one source is more important than the other. We want to see which classifiers are able to identify the important sources/features.

Experiment 2 Parameters

$M_{i,useful}$	$M_{i,noisy}$	N	T	p_1, p_2	σ_i	C	k
[3,5,7]	0	5000	1:2	[0.0,0.1,0.2,0.3,0.4,0.5],[0.0,0.1,0.2,0.3,0.4,0.5]	0.2	1:1	NA

In order to test how the classifiers perform with different level of corruption on the Double XOR data, we varied the probability of corruption for each XOR source in increments of 0.1 from 0 to 0.5. We performed this experiment three times, holding the dimension of each source static at three, five, and seven. The results of this experiment for GMKL and Concatenated GMKL performed on five dimensional XOR are displayed in the figures below. See the appendix for the results from all classifiers tested on five dimensional XOR.

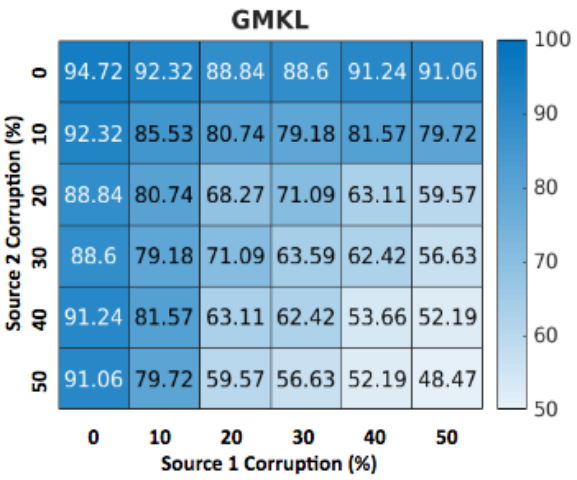

Figure 6: *An illustration of the ability of GMKL to ignore useless data sources.*

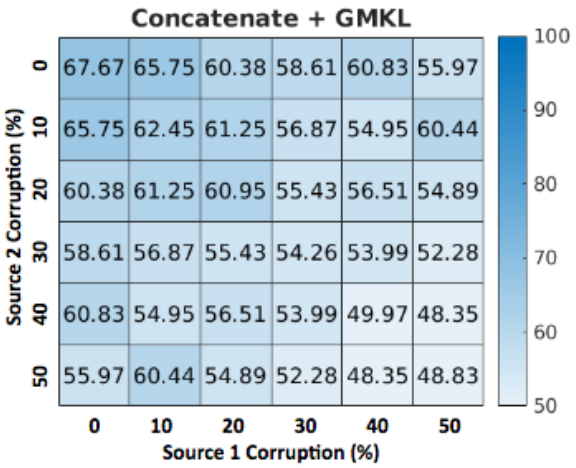

Figure 7: *An illustration that demonstrates Concatenated GMKL is vulnerable to unimportant sources.*

We see that the results of Experiment 2 are consistent with the results of Experiment 1 in that GMKL consistently outperforms Concatenated GMKL. This indicates that the data integration part of GMKL is extremely useful as running GMKL on the concatenated data does not provide nearly as good classification accuracy as the GMKL that treats sources separately. We also see that GMKL is the least affected by one of the two sources becoming corrupted. This is evidence that GMKL effectively identifies which sources are useful and relies primarily on those sources.

Experiment 3: Noise Dimensions

Another variable to explore is the number of noise dimensions. This experiment explores how each classifier is able to handle additional dimensions that do not provide useful information regarding the class each datapoint belongs to. Ideally, the classifiers would be able to identify them as noise dimensions and ignore these dimensions in their predictive model.

For this experiment, we added the same number of noise dimensions to each XOR source. We performed this experiment three times, when there were three, five, and seven XOR dimensions per source. We varied the proportion of dimensions that were noise dimensions. The proportion varied from no noise dimensions, to a quarter, third, half, and then finally two thirds noise dimensions. The purpose of measuring the proportion of noise variables as opposed to the number of noise variables is so that the three experiments over different XOR dimensions can be accurately compared.

Experiment 3 Parameters

$M_{i,useful}$	$M_{i,noisy}$	N	T	p_1, p_2	σ_i	C	k
3	[0,1,2,3,6]	5000	1:2	0,0	0.2	1:1	NA
5	[0,1,3,5,10]	5000	1:2	0,0	0.2	1:1	NA
7	[0,1,4,7,14]	5000	1:2	0,0	0.2	1:1	NA

The results from the three dimensional Double XOR experiment are represented in Figure 8. The results from the five dimensional and seven dimensional Double XOR can be found in the appendix.

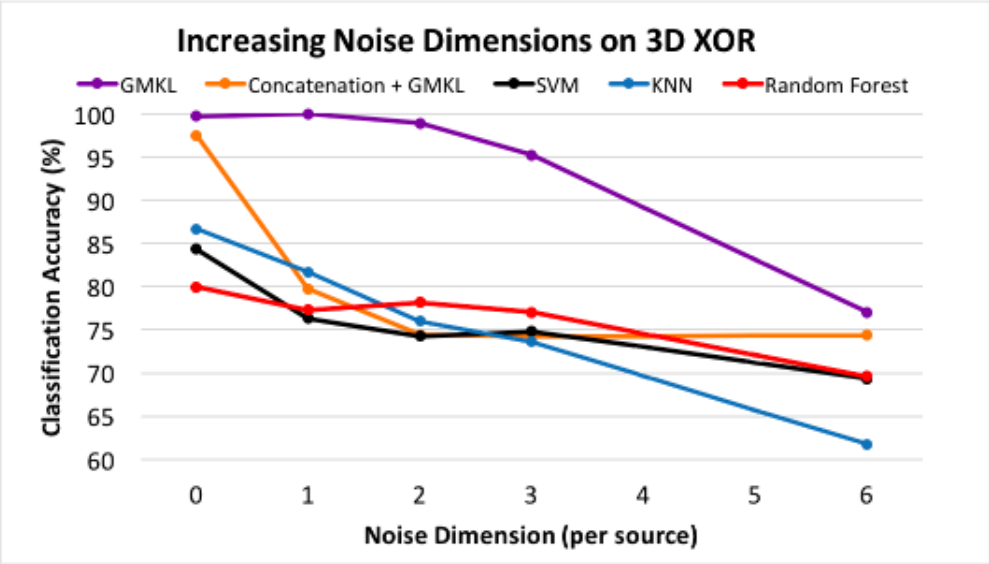

Figure 8: *An illustration of how resistant GMKL is to the affect of noise dimensions compared to the standard classifiers.*

The purple line in the plot above shows how noise resistant GMKL is while the accuracy of Concatenated GMKL falls when only a single noise dimension is added. When there are the same number of noise dimensions as there are useful dimensions, the accuracy of GMKL is still greater than 95 percent. This tells us that GMKL effectively ignores noise dimensions. It is worthy to note the Random Forest classifier is considered noise resistant and is even less affected by noise than GMKL. However, GMKL is initially so much more accurate than Random Forest, that as far as we tested, the accuracy of GMKL remains superior to the Random Forest classifier.

Experiment 4: Corrupted Sine and XOR Sources

This experiment follows the same structure as Experiment 2 in terms of varying the corruption probabilities to each source. The difference is that instead of using two sources with the XOR structure, one XOR source was used and one Sine source was used. We also varied the corruption probabilities in different increments. The corruption probability on each source ranged from 0 to 0.45 in increments of 0.15. We executed this experiment on three dimensional XOR and Sine data with a period of two. The results are displayed below.

Experiment 4 Parameters

$M_{i,useful}$	$M_{i,noisy}$	N	T	p_1, p_2	σ_i	C	k
3	0	0	1:2	[0.0,0.15,0.3,0.45],[0.0,0.15,0.3,0.45]	0.2	1:1	[1,2,3]

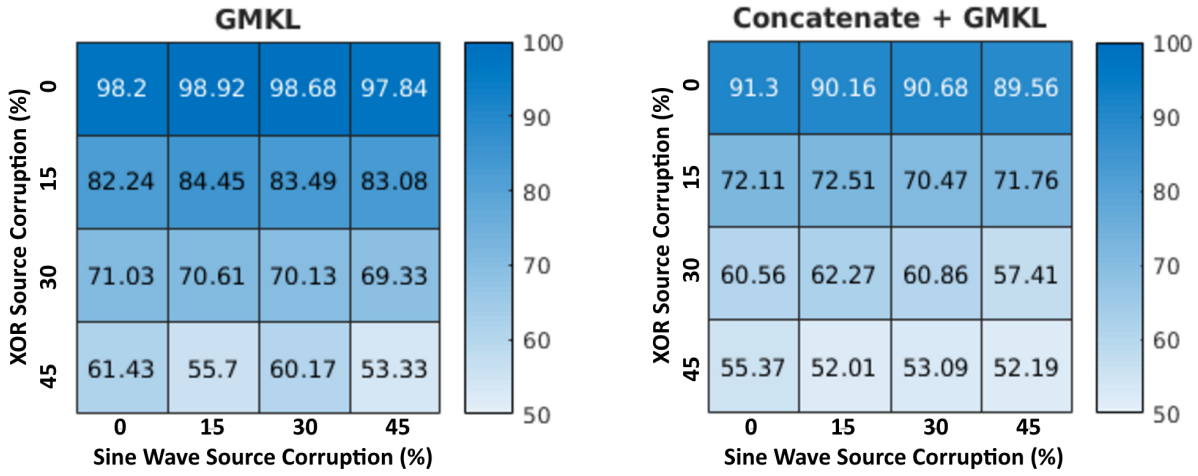

Figure 9: *An illustration that shows GMKL is unable to classify accurately using solely a Sine source.*

Again we can see that GMKL outperforms Concatenated GMKL overall. From the heat maps above, we can observe that GMKL does not handle Sine data well. When the XOR source is heavily corrupted, the prediction accuracy of GMKL falls to barely better than a coin flip. On the opposite end of the heat map, we can note that GMKL performs very well when the Sine source is heavily corrupted as long as the XOR source remains intact. This means that GMKL is insensitive to unreliable data. GMKL is successfully able to identify the source that does not contribute to classification accuracy and ignore it.

Conclusion

From these experiments we have seen not only the usefulness of GMKL, but also the issues that traditional classifiers face in data integration. Highly complex decision boundaries, variably useful data sources, and noisy dimensions have all been shown to negatively affect the classification accuracy of traditional classifiers such as KNN, SVM and Random Forest. Although these aspects negatively affect the accuracy of GMKL as well, we saw that GMKL was much more resistant to these factors than the standard classifiers. Not only does this illustrate the usefulness of GMKL as a tool for classification and data integration, but the way in which GMKL generates kernels separately indicates the importance of treating your data sources as separate entities and the damage you will do to your classification accuracy if you concatenate it all together without any thought. The drastic improvements seen by using an intelligent data integration technique over naive concatenation illustrates that data integration is extremely useful and is worth developing.

Future Work

Data integration, data simulation, and benchmarking of machine learning techniques are all fields in need of further investigation. In particular, the behavior of machine learning and data integration techniques as functions of properties of the data and its decision boundaries is very poorly understood. Some fruitful further questions for research include:

- **Shared Information Across Sources:** An important component of real world multi-source datasets is that information is shared across sources. That is, the sources aren't completely independent of each other, but both give insights into similar underlying hidden variables. The current data generation network model presented above does not have a way to systematically vary the information that is shared between both sources, but it would be very interesting to investigate if and how this property of the data affected the performance of classifiers.

- **Neural Networks:** Artificial neural networks are very powerful classifiers that are growing in popularity. They are an extremely extensible and flexible framework, with specialized versions existing for specific problems within the machine learning. They were not included in these experiments because they are too flexible. Neural networks have many more parameters than most popular classifiers, so it was not known how to pick an appropriate network architecture and topology for comparison against the other classifiers.
- **Theoretical Framework of Decision Boundaries:** We saw in Experiment 4 that GMKL performed well on the XOR dataset and not well on the Sine dataset. We would like to develop a theoretical framework that characterizes the aspects of decision boundaries that enable or disable a particular data integration technique from performing well on a dataset. We would like to come up with a concrete method for determining the best choice of data integration technique for a specific decision boundary.
- **More Computationally Expensive Testing:** All of the experiments in this paper ran with $N = 5000$ data points. For really thorough results, it'd be better to run these experiments for several values of $N \in \{1000, 5000, 1000, 2000, 50000, 10000\}$ as well as for more values of noise, more dimensions, more corruption levels, more width γ and regularization C of the SVMs, etc.
- **More varied simulated data:** This paper only used two types of decision boundaries, that of the sine wave and that of the N-dimensional XOR. There are many types of decision boundaries that are not represented here and many possible properties of data that aren't present here. Possible unimplemented ideas were to create separable data in $m < N$ dimensions and then to use (mostly) monotonic transformations to map the m dimensions into N-dimensional space, to create N dimensional separable data.

Appendix

Variable	Description
$M_{i,useful}$	Number of useful Data Dimensions for Data Source i
$M_{i,noisy}$	Number of Noisy Dimensions for Data Source i
N	Number of Data Points Generated (training and testing)
T	Ratio of Training to Testing
p_1, p_2	Probability of Data Source Corruption
σ_i	Variance of Gaussian Noise
C	Proportion of Data points in each class
k	Period of Sine Curve

Experiment 1: Data Dimension Scaling

XOR Dimensions (per source)	KNN	Random Forest	SVM	Concatenate + GMKL	GMKL
2	99.0	96.6	99.0	99.9	99.9
3	86.2	79.7	83.6	97.5	99.8
4	70.1	68.7	68.9	78.6	97.5
5	60.8	58.1	61.4	66.2	96.4
6	57.8	55.0	60.2	64.6	83.7
7	58.1	54.8	57.2	63.5	74.9

Experiment 2: Corrupted XOR Sources

p_1	p_2	KNN	Random Forest	SVM	Concatenate + GMKL	GMKL
0.0	0.0	61.5	58.6	62.6	67.7	94.7
0.0	0.1	61.2	56.8	62.0	65.8	92.3
0.0	0.2	57.6	58.3	59.5	60.4	88.8
0.0	0.3	55.2	54.9	57.0	58.6	88.6
0.0	0.4	57.3	56.7	59.7	60.8	91.2
0.0	0.5	57.3	57.4	57.4	56.0	91.1
0.1	0.1	58.2	55.7	59.5	62.5	85.5
0.1	0.2	54.7	53.2	57.3	61.3	80.7
0.1	0.3	54.3	53.9	55.1	56.9	79.2
0.1	0.4	54.8	53.5	54.7	55.0	81.6
0.1	0.5	54.9	54.3	53.8	60.4	79.2
0.2	0.2	55.3	50.8	55.4	61.0	68.3
0.2	0.3	54.2	51.6	55.6	55.4	71.1
0.2	0.4	50.9	51.5	49.7	56.5	63.1
0.2	0.5	50.4	50.7	52.5	54.9	59.6
0.3	0.3	52.4	48.1	50.9	54.3	63.6
0.3	0.4	52.2	51.7	51.2	54.0	62.4
0.3	0.5	53.3	50.0	48.6	52.3	56.6
0.4	0.4	51.0	52.4	53.1	50.0	53.7
0.4	0.5	50.7	48.4	50.8	48.4	52.2
0.5	0.5	49.1	49.7	49.0	48.8	48.5

Experiment 3: Noise Dimensions

XOR Dimensions (per source)	Noise Dimensions (per source)	KNN	Random Forest	SVM	Concatenate + GMKL	GMKL
3	0	86.7	80.0	84.3	97.5	99.8
3	1	81.7	77.3	76.3	79.7	100.0
3	2	76.0	78.2	74.3	74.5	98.9
3	3	73.6	77.0	74.8	74.2	95.3
3	6	61.7	69.6	69.4	74.4	77.0
5	0	61.0	60.4	63.5	66.2	96.4
5	1	58.4	56.6	61.4	66.7	72.3
5	3	57.5	54.2	59.5	67.2	70.0
5	5	57.0	54.2	58.4	66.3	65.6
5	10	54.7	54.3	58.3	64.0	65.1
7	0	57.7	54.9	60.2	63.5	74.9
7	1	52.4	52.1	53.8	64.2	66.2
7	4	55.3	53.8	56.5	59.8	62.1
7	7	54.8	54.1	56.5	59.7	59.9
7	14	51.6	52.1	54.5	61.9	59.4

References

[NBM14] Elias Chaibub Neto, J Christopher Bare, and Adam A Margolin. Simulation studies as designed experiments: the comparison of penalized regression models in the large p, small n setting. *PloS one*, 9(10):e107957, 2014.

[YXY+11] H. Yang, Z. Xu, J. Ye, I. King, and M. R. Lyu. Efficient sparse generalized multiple kernel learning. *IEEE Transactions on Neural Networks*, 22(3):433–446, March 2011.