Brenna Gibbons (brennamg@stanford.edu)
Cooper Elsworth (coopere@stanford.edu)

Optimizing Downsampling in Variable Density Experimental Data: Predicting Metallic Glasses

Category: Physical Sciences

Abstract: Metallic glasses remain an elusive material type, due to the lack of analytical formulae or empirical models for their prediction. However, recent improvements in experimental methods have provided high-density alloy data, making it possible to apply machine learning techniques for prediction of metallic glasses. In this paper, we consider a variety of machine learning algorithms for predicting metallic glasses and investigate the most efficient way to incorporate high-density experimental data into existing sparse datasets.

Introduction

Metallic glasses are a unique class of materials that combine many of the desirable properties of crystalline metals, such as good electrical conductivity, with the advantages of amorphous glasses, such as ease of processing and high resistance to corrosion. These properties make metallic glasses very attractive for applications including high-efficiency electrical transformers [1] and artificial joints [2]. Unfortunately, there is no analytical formula for determining whether an arbitrary alloy composition is capable of forming a metallic glass, and such predictions are difficult using empirical models [3]. Furthermore, the possible search space is too large to cover experimentally - to capture information for every 2% variation in composition for likely three-element alloys, called ternary alloys, would require 1326 measurements for each of the 2024 possible ternaries. We will therefore explore the capability of machine learning algorithms to accurately predict the glass-forming ability of ternary alloys.

The input to our algorithm is a ternary alloy composition (e.g. $Al_{80}Ni_{15}Zr_5$). We then use a variety of binary classifiers (SVM, Logistic Regression, Random Forest and Neural Network) to output a predicted binary structure (crystalline or glass). After determining the best model for this dataset, we consider the addition of a high-density dataset available on only a few ternaries and quantify its influence on our predictions. This is of particular interest for the experimental material research community, where it remains an open question on how to optimize the collection of high-density ternary data to improve predictions of metallic glasses.

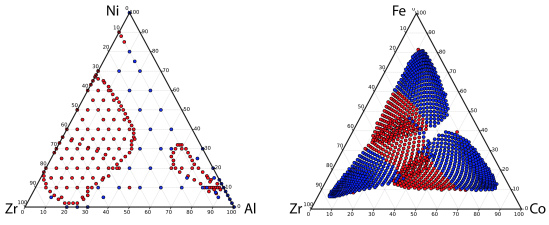

Figure 1: Ternaries in the sparse (left) and dense (right) datasets.

Related Work

The only true way to discover a metallic glass is to synthesize and test it. However, this process can be very time consuming - often, fewer than 1 in 1000 tested compositions results in a newly discovered glass, and it has been estimated that it would take 4000 years to process all possible combinations using traditional methods [4] [5]. Much of the recent focus on metallic glass discovery has gone to improving the experimental throughput rate. Ren et al. use high-throughput x-ray diffraction to evaluate glasses, while Ding et al. use parallel blow forming [6] [7].

In addition to scientific intuition, there are several analytical models that have been used to guide experimental searches, including the dense random packing, stereochemically defined, and efficient cluster packing (ECP) models [4] [8]. These models can guide researchers in choosing a ternary to test, but have poor specific predictive ability. Ferry et al. recently reported a predictive structural model that builds on

the ECP model; while no accuracy or validation metrics were reported, the authors were able to experimentally validate 35/44 new predicted glasses [4].

In recent years, there have been several attempts to use machine learning to predict metallic glasses. Li et al. use correlations and what is effectively a single decision tree to estimate the number of metallic glasses, though their predictive accuracy for metallic glasses is poor [3]. Sun et al. used an SVM model with two input features (difference in liquidus temperature and fictive liquidus temperature), and though they do not report any common validation metrics, they claim good prediction efficiency [9]. In 2016, Ward et al. used a random forest algorithm trained on the Landolt-Bornstein database featurized using the Magpie library, and achieved a predictive accuracy of 90.1% [10]. The same authors refined their model by including high throughput observations from the CoVZr ternary, improving the area under the receiver operating characteristic (ROC) curve from 0.65 to 0.8 [11]. These last two results were the inspiration behind our project, and we hope to further these findings by comparing additional algorithms, exploring the effects of downsampling, and including as-yet-unpublished data.

Data & Features

The data that we use is based on the Landold-Bornstein dataset, which contains ~ 5700 data points over ~ 300 ternary alloys [10]. This sparse dataset is a result of the previous generation experimental methods and contains relatively few experimental results for each ternary. In fact, the median number of experimental datapoints per ternary is 6 and the most dense ternary has 203 datapoints, shown in Figure 1. We also consider supplementing this sparse dataset with high-density data consisting of 9 ternaries with ~ 1300 points each [12], shown in Figure 1.

The raw data from these datasets contain a string representation of a composition (e.g. "$Al_{80}Ni_{15}Zr_5$") and a binary class definition - "0" for amorphous, or glassy, and "1" for crystalline and amorphous-crystalline. The data was featurized using Magpie, the Materials-Agnostic Platform for Informatics and Exploration [13]. 144 features capture the weighted average, minimum, maximum, and standard deviation of various physical parameters (e.g. the mean electronegativity, weighted by the composition). It should be noted that many of these features are correlated - for example, the mean covalent radius and the mean ground state atomic volume will be highly correlated. We preprocessed these features to ensure zero-mean and unit-variance.

The methodology we used to split training and dev sets was motivated by the material science application of predicting glass content on unknown ternaries. Therefore, we split the data by ternary, such that individual ternaries were either only in the training set or dev set. In practice, this means that the training and dev sets are sampling different distributions leading to higher bias, compared to splitting the data randomly, agnostic to the ternaries. For training and dev sets we randomly split the sparse ternaries to achieve a ratio of data points of 80/20, while always maintaining the AlNiZr ternary in the dev set to allow for direct comparison between methods on a single ternary. When including the high-density data, we maintained the 80/20 split for the sparse dataset and separated the high-density such that 6 high-density ternaries were in the training set and 3 were in the test set.

Methods

Methods: We implemented a variety of binary classifiers using the scikit-learn library [14] to compare the performance of different machine-learning methods for this data. These methods are outlined as follows:

Logistic regression was implemented through scikit-learn's `LogisiticRegression()` function, an unweighted binary classification with L_2-regularization. This uses the gradient descent update,

$$\theta_j \leftarrow \theta_j + \alpha(y - \sigma(x))x_j \quad \forall j$$

The support vector machine classifier was implemented through scikit-learn's `SVC()` function with a radial basis function (RBF) kernel and regularization controlling the smoothness of the RBF. In the most basic sense, this classifier finds the separating hyperplane that maximizes the distance to the nearest points and solves,

$$\underset{w,w_0}{\operatorname{argmin}} \frac{1}{2} w^T w + \gamma \|w\|_2 \qquad \text{subject to} \qquad y_n(w^T x_n + w_0) \geq 1 \quad \forall n.$$

The random forest classifier was implemented as an ensemble of decision trees using sklearn's `RandomForestClassifier()` function. This classifier contained the most parameters that had direct effects on the performance of the classification, no maximum depth, 500 trees, and $\sqrt{n}$ features considered at each split.

The neural network classifier was constructed as a simple network with one hidden layer of 100 neurons and a sigmoid ($\sigma(z) = 1/(1+e^{-z})$) activation function using scikit-learn's `MLPClassifier()` function.

For logistic regression, SVM, and neural network classifiers, we consider different amounts of regularization to manage overfitting. For each classifier we considered how the probability threshold separating the classes affects the results. For many applications the probability threshold is set at 50%, but in applications where false positives are detrimental, the threshold may be increased. Probability thresholds are also increased when the model is systematically over-predicting a class due to an unbalanced training data set.

Metrics: We consider a variety of metrics to quantify the performance of the preceding classifiers. Firstly, we consider the accuracy metric, defined as the percentage of correct classifications in the dataset. For predicting glasses, the false positive rate is especially important, as the experimental validation of a prediction can be quite time consuming. Therefore, a receiver operating characteristic (ROC) curve and log-loss are particularly relevant metrics. The ROC curve is quantified by computing the area underneath the ROC curve (AUROC), with 1.0 signifying perfect classification. Finally, to quantify the precision and recall of the methods, we implement the F_1-score, defined as the harmonic mean of precision and recall. We compute each of these metrics for the training and dev sets to compare the methods directly, but to also understand how much bias (i.e. overfitting) is occurring in each model by comparing the relative size of these metrics between the training and dev sets.

Results

Comparing classifiers on the sparse dataset:
In order to determine the most suitable classification method for this dataset, we first tried a variety of algorithms on the sparse dataset. The methods and metrics are outlined in the previous section and results are summarized in Table 1. We observe that the random forest classifier performs the best across metrics, and the neural network classifier has the second best performance. The ROC curves for classifiers is shown in Figure 2. It shows that the random forest and neural network classifiers are particularly effective at

Table 1: Performance metrics for each of the classification methods.

	Accuracy		AUROC		Log-loss		F_1-score	
	Train	Dev	Train	Dev	Train	Dev	Train	Dev
Logistic Regression	82.4%	85.7%	0.877	0.836	0.398	0.384	0.689	0.639
SVM	84.1%	85.8%	0.907	0.859	0.364	0.372	0.703	0.590
Random Forest	100%	88.3%	1.000	0.915	0.064	0.313	1.000	0.684
Neural Network	86.6%	84.7%	0.941	0.876	0.294	0.336	0.842	0.665

predicting without false positives, though all of the models are reasonable. For the random forest classifier, we find a classification accuracy of approximately 90%, similar to published results [10].

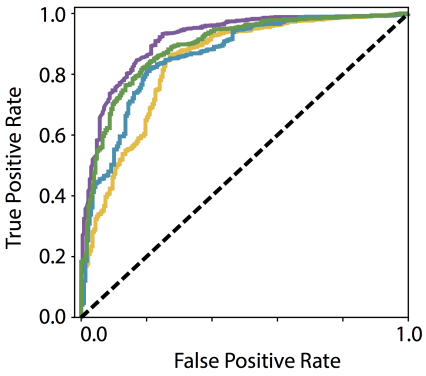

Figure 2: Receiver operating characteristic (ROC) curve for the logistic regression (yellow), SVM (blue), random forest (purple), and neural network (green) classifiers.

One issue with this data is that it has relatively few datapoints relative to the number of features. In addition, there are likely a large percentage of features that are correlated to other features or not relevant to the classification. These two factors make classifiers prone to overfitting the training distribution. Feature importances were determined by averaging the results of sklearn's random forest classifier feature importance over 20 runs, and it was clear that ~ 20 of the features carry a disproportionate importance. We ran all of these methods using only the 20 most important features, but the results varied only marginally.

We observe a relatively large disparity in metrics between the training and dev sets across classifiers. This suggests a systematic bias that could result from overfitting of the classifiers on training data or a difference in distributions between the training and dev sets. The bias is not reduced by tweaking the classifier parameters or introducing more regularization into the SVM, LR, or NN classifiers. This means that overfitting is likely not the cause of high bias, rather we expect that the bias arises from differences in distributions between the training and dev sets. This makes sense, since we separate the data by ternaries, which means that the training and dev sets sample different distributions, especially with the relatively small size of this dataset.

Incorporating the dense dataset: The question of how additional experimental data can help improve metallic glass predictions is of significant material science interest. Therefore, we consider adding the dense dataset to the best performing classifier, the random forest. Figure 4 shows the difference in RF classification for the sparse training set (top) and when we include 6 high-density ternaries to the training set. The introduction of the dense data improves the classification significantly.

The fact that these experiments are costly and time-intensive to perform motivates the question - how much high-density data is necessary to observe an improvement in classifications? To investigate this question, we uniformly downsample the dense dataset at different rates. Figure 3 shows the effect of downsampling on the accuracy and log-loss of the random forest classifier for both the dev set and several selected ternaries. Clearly, it only takes a small amount of additional data to improve predictions, with 2% of the dense data showing significant improvements in classification for the selected ternaries. The increase in log-loss and decrease in accuracy arises from outlier experimental results disagreeing with the increasingly confident classification (i.e. red points in FeNbTi ternary, Figure 4). The predictions from the model with 5% of the dense data are shown in Figure 4, which recovers a similar accuracy to the model trained using the full dense dataset. With limited resources, this suggests that experimentalists should collect fewer data points on more ternaries to improve the predictive capabilities of these models.

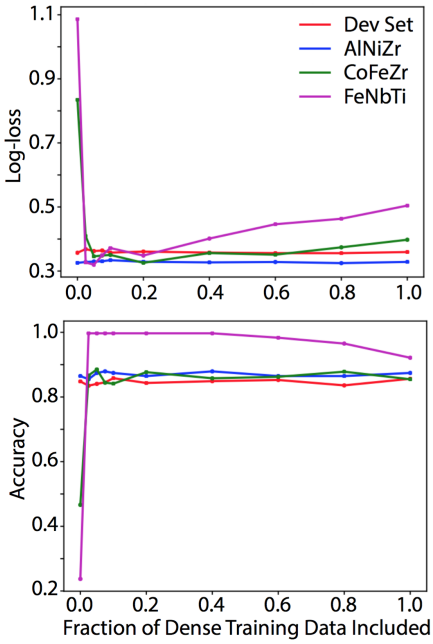

Figure 3: Random forest classifier log-loss (top) and accuracy (bottom) with different levels of downsampling of the dense dataset.

One factor that likely contributed to the increased accuracy in the dense test ternaries CoFeZr and FeNbTi when using the dense training data is an improvement in similarity between the distributions in these ternaries versus the training set. The sparse dataset is biased towards glass with a 71:29 glass-crystalline

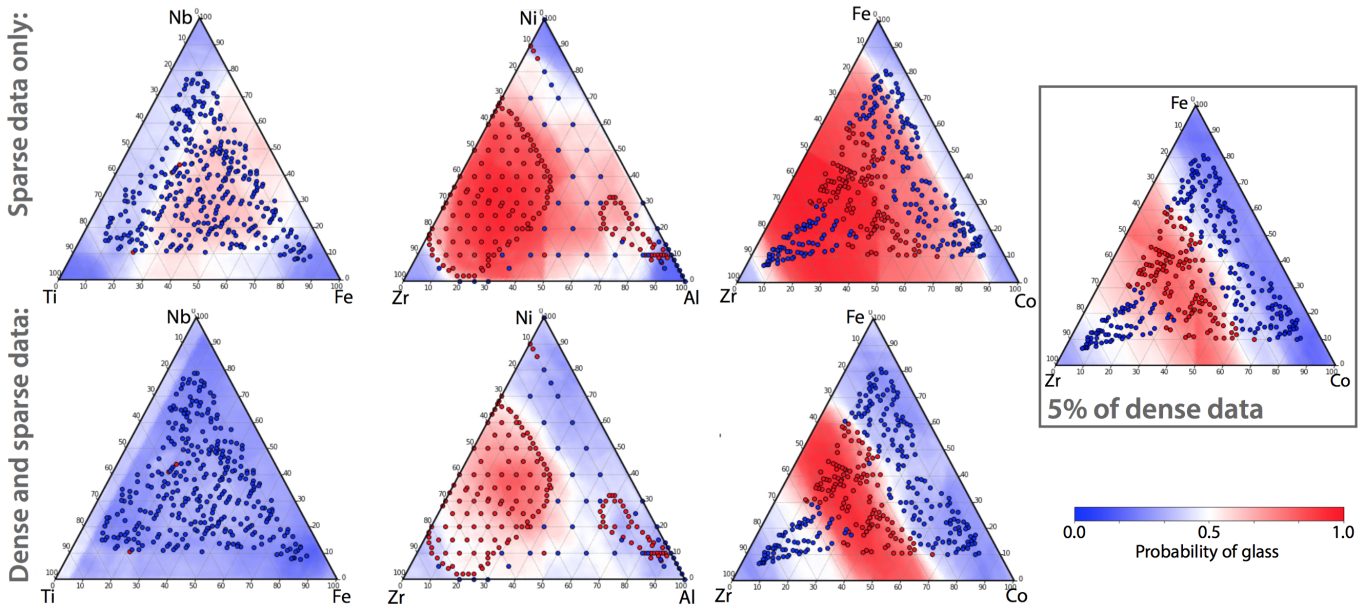

Figure 4: High-density ternaries showing the experimental results (dots) subsampled by a factor of four and random forest predictions (background color). The top row uses only the sparse data in the training set, while the bottom row includes 6 of the dense data ternaries as well. On the right, we shows the predictions for CoFeZr using the sparse dataset and 5% of the data from the same 6 dense ternaries; it is qualitatively clear that much of the improvement from including the high density data is captured by this first 5%.

ratio, while the dense data is 79% crystalline (over all possible ternary alloys, the ratio is closer to 1:1 million [3]). This difference arises from an inherent bias in the academic community towards reporting only positive results (i.e. hard to find metallic glasses). We therefore expect the classifier to overpredict for glass, so we tried raising the probability threshold, but the results did not improve appreciably. Another solution would be to include more ternaries with null-results (all crystalline) in the training set.

Conclusions & Future Work

We tested several variations of models to predict the likelihood of forming a glass vs crystalline structure from the composition and associated chemical features of ternary alloys. Random forest and neural network classifiers performed the best, with similar performances to the paper in which we found the data [10]. We hypothesize that logistic regression did relatively poorly because of the complex nature of this problem - after decades of study, there are no simplified analytical relations that estimate glass forming ability well. SVM may be better able to capture this complexity with additional effort to tune the hyperparameters, but our relatively simple SVM may also fail to adequately capture the complexity of the problem. Similarly, with additional tuning and resources, a neural network may be able to outperform our random forest model, which was quite robust out of the box. The most obvious way to improve all of these models would be to add more data, particularly data from ternaries that arent represented here; however, it may not be possible to reach perfect accuracy, since the experimental results occasionally disagree. An additional avenue of investigation could be exploring new features, such as those used in analytical models from literature.

One surprising result of our investigation was how little of the dense data is required to achieve a noticeable improvement in performance when predicting on specific ternaries. This result is notable because these experiments are performed at SLAC, where time is extremely limited, so any reduction in data collection time is valuable. Moving forward, we hope that these models can be used to guide experimental data collection and ultimately result in the discovery of new metallic glasses.

Acknowledgements: We would like to thank Apurva Mehta, Jae Hattrick-Simpers, and Logan Ward for graciously allowing us to use unpublished data for this study.

Contributions: The workload was distributed evenly among Brenna Gibbons (BG) and Cooper Elsworth (CE). BG took the lead on generating the feature set with Magpie. CE implemented the first working example of the Random Forest algorithm and tailored the methodology to this dataset. BG and CE worked together to devise and implement the experiments. BG and CE both wrote up the results and designed the poster.

References

[1] Kennedy, Barry (1998). "Energy Efficient Transformers," *McGraw-Hill*.

[2] Zberg, B., Uggowitzer, P.J., and Lffler, J F. (2009). "MgZnCa glasses without clinically observable hydrogen evolution for biodegradable implants," *Nature materials*, 8(11), 887-891.

[3] Li, Y., Zhao, S., Liu, Y., Gong, P., and Schroers, J. (2017). "How Many Bulk Metallic Glasses Are There?," *ACS Combinatorial Science*.

[4] Laws, K. J., Miracle, D.B., and Ferry, M. (2015). "A predictive structural model for bulk metallic glasses," *Nature Communications*, 6.

[5] Gerson, E. "Glasses strong as steel: A fast way to find the best" *Yale News* https://news.yale.edu/2014/04/13/glasses-strong-steel-fast-way-find-best

[6] Ren, F., Williams, T., Hattrick-Simpers, J., and Mehta, A. (2017). "On-the-fly segmentation approaches for x-ray diffraction datasets for metallic glasses," *MRS Communications*, 7(3), 613-620.

[7] Ding, S., Liu, Y., Li, Y., Liu, Z., Sohn, S., Walker, F. J., and Schroers, J. (2014). "Combinatorial development of bulk metallic glasses," *Nature materials*, 13(5), 494-500.

[8] Miracle, D. B. (2004). "A structural model for metallic glasses," *Nature materials*, 3(10), 697-702.

[9] Sun, Y. T., Bai, H. Y., Li, M. Z., and Wang, W. H. (2017). "Machine Learning Approach for Prediction and Understanding of Glass-Forming Ability," *The journal of physical chemistry letters*, 8(14), 3434-3439.

[10] Ward, L., Agrawal, A., Choudhary, A., and Wolverton, C. (2016). "A general-purpose machine learning framework for predicting properties of inorganic materials," *npj Computational Materials 2*.

[11] Ren, F., Ward, L., Williams, T., Laws, K.J., Wolverton, C.M., Hattrick-Simpers, J., Mehta, A. submitted

[12] Ren, F., Pandolfi, R., Van Campen, D., Hexemer, A., and Mehta, A. (2017). "On-the-fly Data Assessment for High Throughput X-ray Diffraction Measurement," *ACS Combinatorial Science*.

[13] "Magpie, the Materials-Agnostic Platform for Informatics and Exploration," https://bitbucket.org/wolverton/magpie

[14] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., and Vanderplas, J. (2011). "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, 12(Oct), pp.2825-2830.