Midterm Report: Optimizing Machine Learning Model Memory Usage With Minimal Statistical Performance Degradation and Maximal Interoperability

By: Saharat Rodjanamongkol, Pakin Wirojwatanakul, and Eliot Seo Shekhtman

1. Introduction

The performance of a machine learning model can be broken down into statistical performance, hardware performance, and interoperability. In practice, oftentimes improving the statistical performance comes at the expense of a degradation in hardware performance or interoperability. For example, creating new features via the process of feature engineering can improve the statistical performance, but degrade the hardware performance because the new transformed data will have higher memory requirements.

In this project we seek to explore various dimension reduction methods to determine which methods can maximally conserve memory (optimizing hardware performance) and maximize interpretability while minimally degrading statistical performance of the models. At this point in the project, we have identified multiple dimension reduction experiments. We have also identified, analyzed, and processed the data sets in which to test our experiments. Section 2 goes over the data exploratory analysis and processing for each of our datasets. Section 3 goes over the various dimension reduction experiments. Section 4 goes over the future works.

2. Data Exploratory Analysis and Processing

2.1 Wine Quality Dataset

The red wine dataset has 11 features related to the chemical properties of wine, which is used to predict its quality. The data set is already pretty clean. It does not have any NaN values and no noticeable outliers. The first quartile of all the features are all non-zero, which indicates that the dataset is dense.

2.2 AUS weather Dataset

The AUS weather dataset is a dataset consisting of 23 original features describing meteorological conditions such as cloudiness, wind speed/direction, temperature, date, location etc which seeks to predict given a set of initial conditions for the day before, whether it'll rain tomorrow. These features are rather comprehensive, and we chose this dataset with the hopes that some of them will be correlated with each other and so their removal might not detriment prediction efforts significantly.

Unfortunately, this dataset came in a very messy state. Every feature, including the dependent feature on which we seek to predict, had NaN values in it to varying degrees. Most features had less than 10% values missing; however, two features (Pressure9am and Pressure3pm) had just over 10%, and four more (Evaporation, Sunshine, Cloud9am, and Cloud3pm) had between 35%-50% values as missing. To fix this, we first removed rows where the more present features (those where the missing rate was <10%) had at least one missing value, and observed that the the overall number of observations decreased from 145,460 to 121,790. This indicates that most data will have all observed values. We then took the other six columns and imputed the mean value for the missing entries, reasoning being that too many values were missing to be able to impute based on previous observations, and if we decided that this method was too noisy we could simply drop the columns later; however, we haven't found this to be the case in testing so far.

Once this was done we converted the string-encoded boolean values ("Yes" rain and "No" rain) to floats (1.0 and 0.0) and converted cardinal directions. Cardinal directions were converted by taking the counts of 'N', 'S', 'E', 'W' in a string and scaling the sum until it equalled six. This was because we observed that cardinal directions were either given as a single value ('N', 'W'), two values ('NE', 'SW'), or three values where one was repeated twice to give emphasis ('NWN', 'SSE'), so six was the least common multiple such that the magnitude of the directions would always be equal. After this, all entries had non-missing float values.

It is important to note that for this dataset, the labels are heavily imbalanced with around 80% of days having no rain and only 20% having rain. This plays a role in the performance of classification models.

2.3 Adult Dataset

The target of the adult dataset is to classify whether the income is above or below 50K. About 23% of the adults have income above 50K. The features of workclass, occupation, and native_country have "?" as unknown values. We decide to keep the "?" as its own category because the number is relatively small and could possibly be an indicator of lower income. The main outliers in the dataset are the capital_gain and capital_loss. Most groups have 0 capital_gain and capital_loss while a few have very high values. We left the capital_gain and capital_loss data as-is because they could be a strong income indicator. We dropped the ordinal education feature because education_num already captures the ordinality. We one-hot encode all the categorical features and normalized continuous features. After processing the data we went from 48842 samples with 15 features to 93 features. The cleaned up data is fairly sparse due to the one-hot encoding.

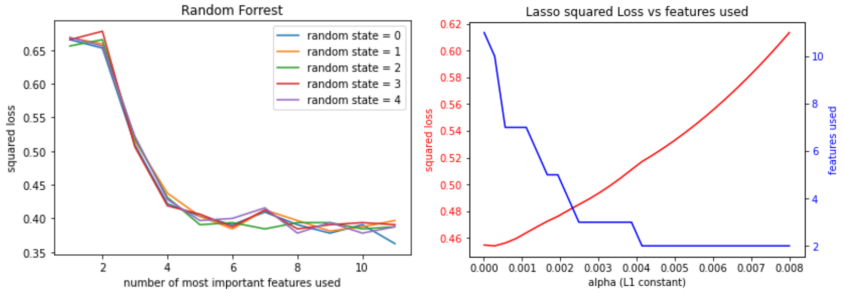
2.4 Titanic Dataset

The target of the Titanic dataset is to classify whether the passenger survived. About 38% of the passengers survived. The Age, Fare, Cabin, and Embarked features have nan in them. We imputed the missing continuous Age and Fare values with the mean. Embarked was imputed with the most common category. The Cabin was processed by taking the first character and leaving nan as-is because different cabins could have different survival rates. The main outliers in the dataset are the SibSp and Parch, the number of siblings, spouses, parents, and children. Most passengers board the ship alone, while some travel with 10 family members. We process the Name by turning it to boolean whether the person has a special title, e.g. Major and Rev. We dropped the Ticket feature because we couldn't find a good feature to extract. We one-hot encode all the categorical features and normalized continuous features. After processing the data we went from 1309 samples with 12 features to 21 features.

3. Experiments

3.1 Selecting Features with Random Forest, Lasso, and Control Burn

The first experiment involves dimension reduction techniques that select features to keep based on their importance. Because the reduced feature set is just a subset of the old features, this family of dimension reduction techniques has a very high degree of interoperability. The methods that we explored include Random Forest, LASSO, and ControlBurn. Based on our preliminary experiments on the wine dataset, we observe that we can use Random Forest and LASSO to select the most important features to keep without much increase in the squared loss.



3.2 Dimension Reduction with PCA and Kernel PCA

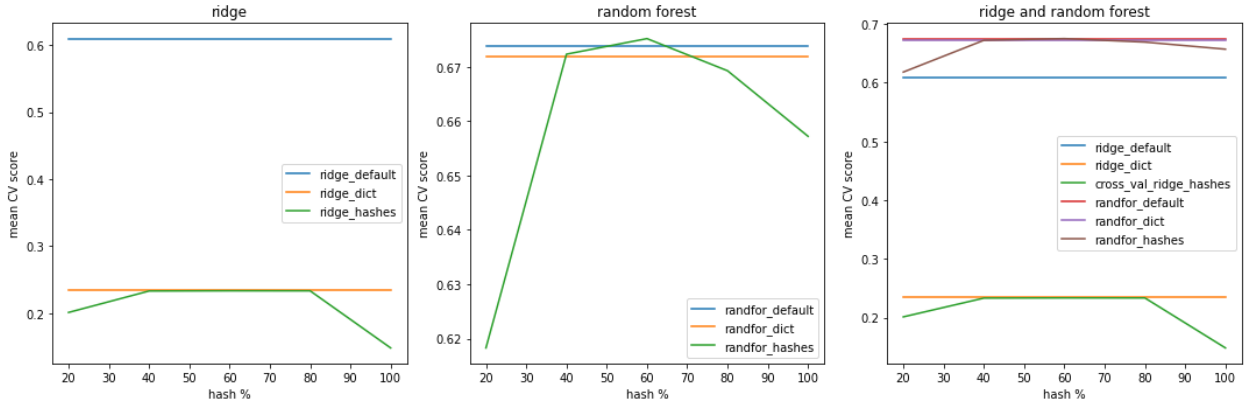
The second experiment was concerning PCA's effectiveness on the AUS weather dataset. The dataset was first taken as two types, a standard form (as cleaned) and a normalized form (where each feature's mean was zeroed). A preliminary SVD analysis on the standard form indicated that while the data wasn't rank deficient and each singular value was >1, some vastly outweighed the others with the first singular value being a multiple of 10^5.

PCA with a downstream logistic regression classifier using ridge regularization was applied repeatedly under increased numbers of output features on both the standard and normalized data, in order to gauge how many features seemed to be of great significance to the data. Ridge regularization was chosen as to prevent double feature selection, with logistic regression being one of the most baseline methods of binary classification. The confusion matrices for all evaluations were collected alongside raw scores, and the scores were plotted for interpretation purposes.

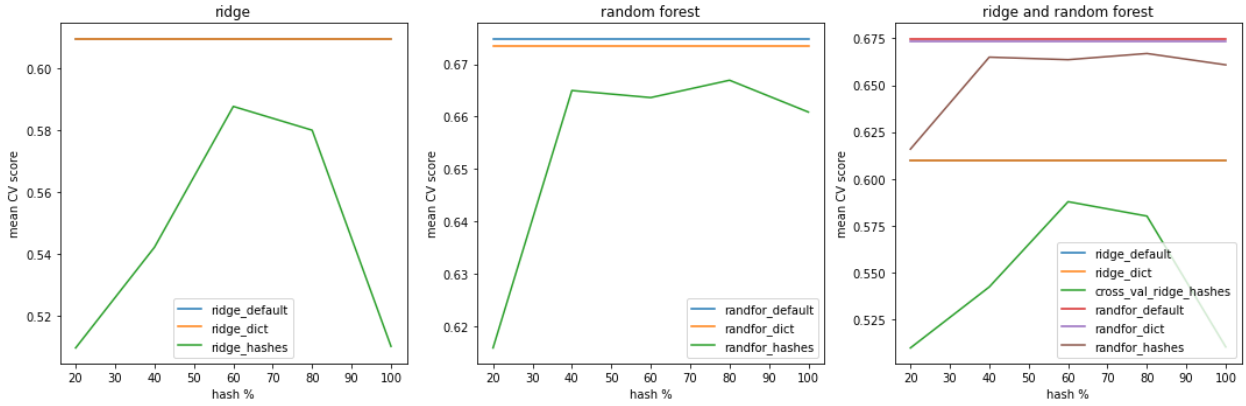
3.3 Dimension Reduction with Feature Hashing

We compared FeatureHasher, DictVectorizer and default plain data as features to train RidgeClassifier and RandomForestClassifier. FeatureHasher hashes the features to a smaller bucket, shown as a percentage of the original feature length on the x-axis below. We evaluated the features using 5 fold cross validation and F1 score. The unnormalized features from DictVectorizer and FeatureHasher perform poorly on RidgeClassifier but perform well on the random forest. This is likely due to random forests being insensitive to features' scale. The random forest also has good performance even when hashing down to around 40% of the original feature size, likely due to data sparsity. While the random forest might perform better with smaller data size, it takes longer to train, around 10-20 seconds compared to ridge taking less than a second. The main drawback of hashing features is that the hashed features are uninterpretable. DictVectorizer saves some memory because it's a sparse matrix and performs similarly to the default plain data in normalized data and in random forest. However it doesn't provide any feature reduction and takes very long to train in a random forest, about a minute.

Unnormalized adult mean F1 score vs hash percent



Normalized adult mean F1 score vs hash percent



4. Future Work

Each experiment was conducted in parallel by each of the group members, thus for the final report we would have to standardize the performance metrics for each of the dimension reduction methods. The system performance will be measured by memory usage, which can be measured by determining how many features we can reduce by. The statistical performance will be measured by the Accuracy, the F1 Score, and an analysis of the confusion matrix. The interoperability will be measured by how understandable the reduced features are.