ECE 236A: Final Project, Part II

Robustness to Data Corruption in a 1-Norm Support Vector Machine

David Martinez & Peter Racioppo

Introduction

We now consider multi-class linear classification in the presence of data corruption. Our goal is to build a linear classification algorithm which is robust against erasure of testing features. We assume that the input features in the data set are erased independently of the others, with probability *p.* We first consider the case of unknown erasure probabilities (Classifier 3) and then consider how knowledge of *p* can be used during training to improve the classifier (Classifier 4).

Data corruption removes one or more of our features and thus invalidates the hyperplane boundaries drawn in the space of all of the features. This problem can be corrected by training our multidimensional voting algorithm on every possible feature combination and testing using the voting algorithm that corresponds to the set of testing features. However, there are $2^n - 1$ possible combinations of n features, so this approach would require training an exponential number of SVMs (as a function of the features).

Classifier 3 (v2)

Our first version of Classifier 3 is a modification of Classifier 2, in which we replace the missing features with weighted-averages for these features. In particular, we replace the missing feature with the average value of this feature for each class, divided by the number of elements of each class, summed over all classes. The reasoning here is that a feature of average value should be equally similar to every class, and thus should have no effect on its classification.

Adding noise to the training data has been found in the literature to improve performance in the presence of noise in the testing data. Intuitively, adding noise to the data causes it to spread out and forces an SVM to draw hyperplanes more conservatively. We thus added Gaussian noise in Classifier 3 (independently to each feature). During training, we performed the combinatorial k-fold cross validation method described in Part I, which improves robustness by preventing overfitting. We also randomly corrupted the *k*-fold testing batch, using the average-replacement method described in the previous paragraph. Since the probability of corruption was unknown, we used *p* values sampled from a Poisson distribution. As in Part I, we also included a 1-norm term in the cost function and performed a line search on the value of λ, thus improving robustness by maximizing the size of the margin and preventing overfitting.

Classifier 3 (improved version)

We next implemented a second version of Classifier 3, using the *k*-nearest neighbors (*k*-NN) algorithm to replace corrupted features. If a sample contains an NaN value during testing, the feature vector is passed to the function f(). This function detects which feature indices are

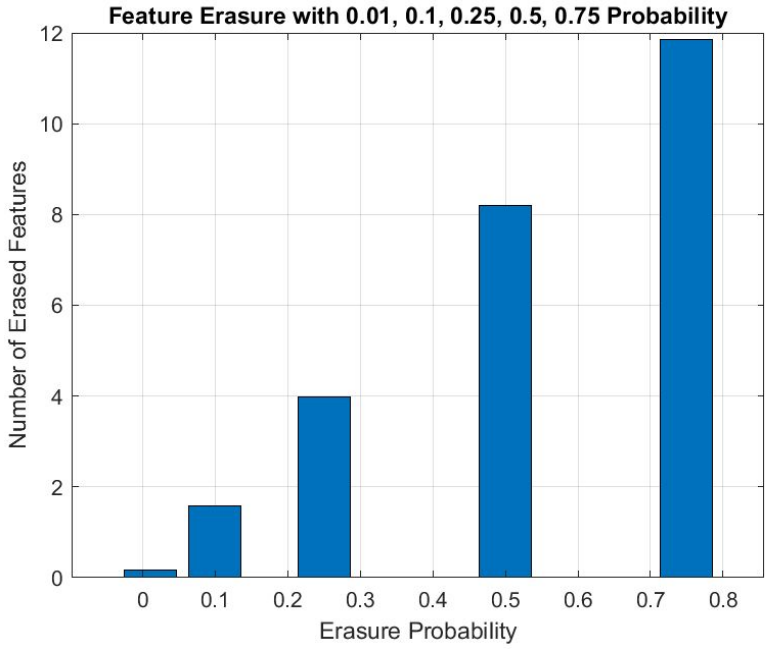
missing and completely removes the dimensions of missing features, in order to do distance operations at a lower dimensionality. We next perform a broadcasting subtraction operation (test_vector - training_data) and then an L1 norm along the sample dimension of the reduced-dimension feature matrix. Whereas the 2-norm is typically used for a distance metric in *k*-NN algorithms, we chose to use a 1-norm metric, which significantly improved execution time. This operation outputs a minimum index, corresponding to the sample that is most closely related to our incomplete data vector. The NaNs from the original vector are then replaced with the feature values of the similar sample vector from our training data. The patched feature vector is then input into the same classification algorithm as in Classifier 1.

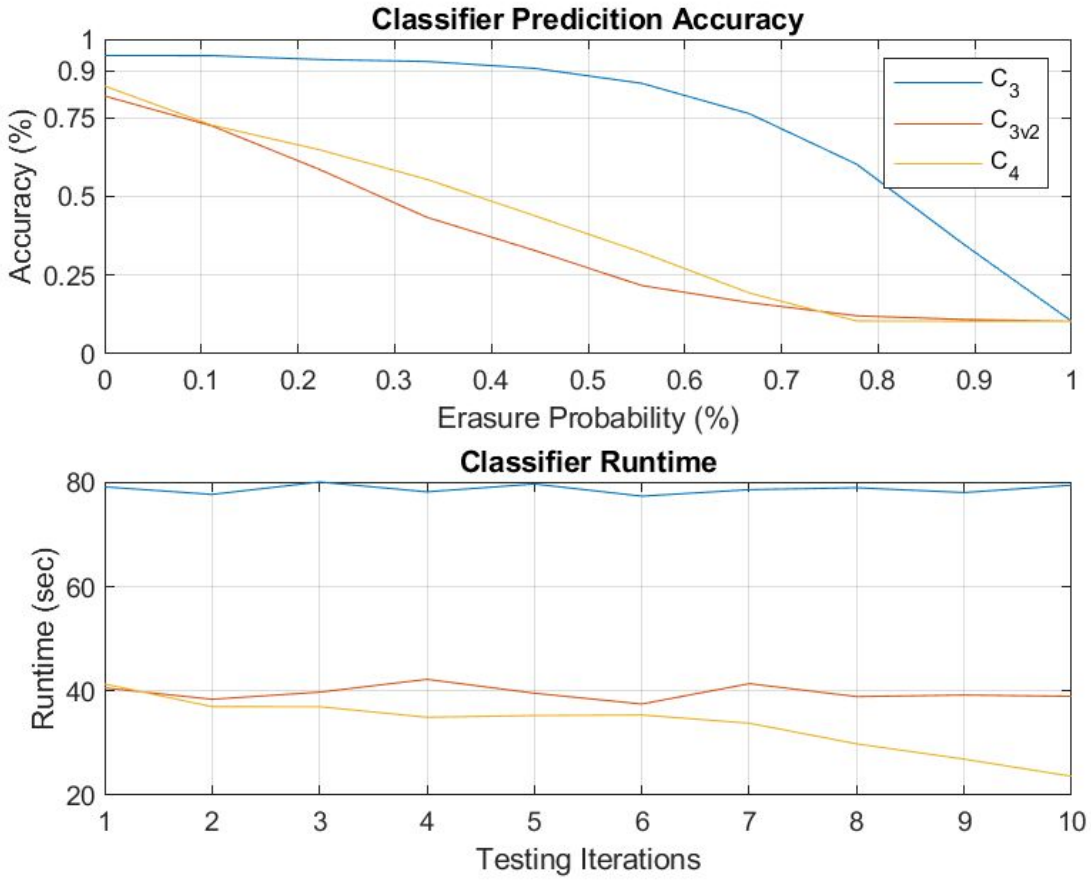
Classifier 4

Classifier 4 uses the same architecture as Classifier 3v2 (the first version), except that it corrupts the training data using the known probabilities of erasure.

Results and Discussion

Our improved version of Classifier 3, which uses the *k*-nearest neighbors algorithm to patch corrupted features before running Classifier 1 on the corrected data, dramatically outperforms our earlier versions of Classifiers 3 and 4, which rely on replacing corrupted values with weighted averages, and adding noise and corruption to the training data. Whereas the prediction accuracies of these earlier models fall off worse-than-linearly with increasing corruption, the *k*-NN based approach displays great robustness, not falling below 90% prediction accuracy until the probability of corruption reaches about 45%. Of the earlier versions of Classifiers 3 and 4, the latter's knowledge of the corruption probability gives it a slight edge between erasure probabilities of about 10% and 70%.





Conclusion

Classifier 3's $O(n^2)$ hyperplane voting system, combined with a k-NN algorithm for patching corrupted features, proved highly robust again increasing erasure probability, maintaining a 90% average testing accuracy until an approximately 0.45 probability of erasure. Introducing a 1-norm into the cost function improves robustness and prevents overfitting, as compared to a vanilla linear classification algorithm, and improves efficiency and sparseness characteristics, as compared to a 2-norm SVM. Future work should focus on the use of the kernel trick to map the data to higher dimensions during the training process.