BE175
Final Project

“Sign-to-speech translation using machine-learning-assisted stretchable sensor arrays”

By: Clara Kang & Rachel Yu

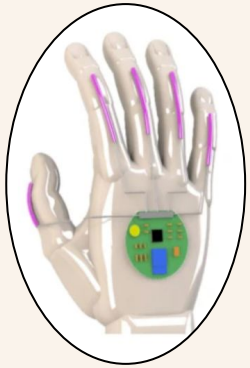
Introduction/Motivation

- Problem: Although sign language plays a crucial role in bridging the conversational gap with those with speech impediments, it is limited in its breadth to those who can understand it.
- Proposed Solution: Researchers have been looking into ways to use wearable technology to translate sign language into audible speech, lowering the barrier to communication.

With the use of machine learning to assist these stretchable sensor assays, we are able to shrink the separation of disabilities and open the world to a whole new audience of listeners, a topic we found both inspiring and intriguing.

Role of Data Analytics: How can we use data analytics/machine learning to identify important data points, classify by labels, and predict a future interpretation of new data.

Methods



01

Signal Collection
Collecting the finger electrode signals for each gesture

02

Data Preprocessing
Sample selection, labels, trial concatenation

03

Dimensionality Reduction
PCA vs. LDA

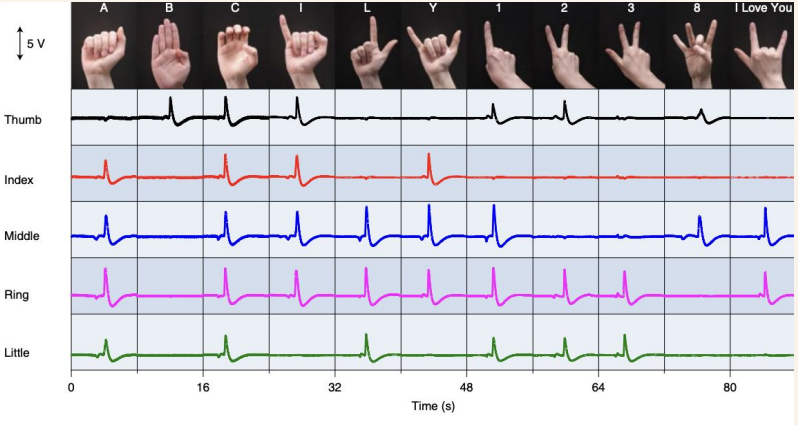
04

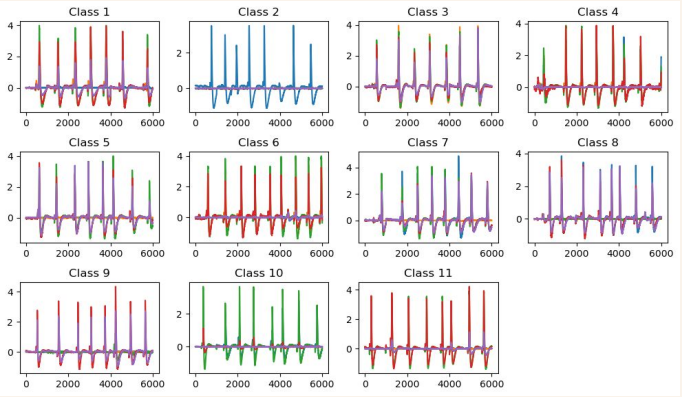
Classification
SVMs, kernel and hyperparameter optimization

05

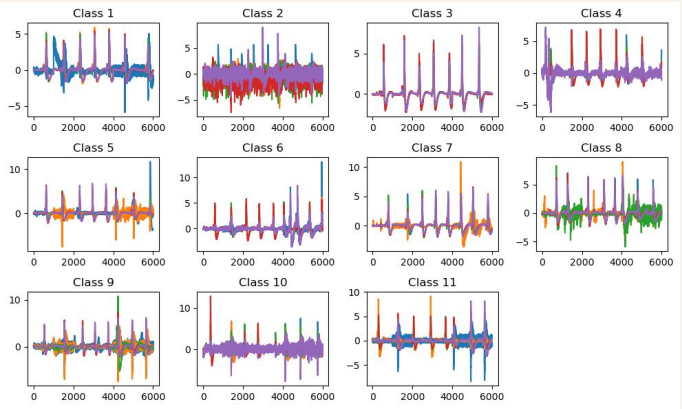
Validation & Testing
K-fold CV
LOO CV

Signal Collection & Data Preprocessing





Normalize?



Initial:
11 classes (for each gesture)
2 trials for each class
Total concatenated size: 263, 215 data points

Method:
3000 samples from each trial
6000 samples total for each class (gesture)
added corresponding class labels

Dimensionality Reduction

PCA

- Unsupervised
- Maximizes variance in dataset, regardless of class labels
- Only requires features of dataset
- Reduces dimensionality by capturing most significant variance in dataset
- Ouptut: PCs that are linear combinations of original features

```
from sklearn.decomposition import PCA
pca = PCA(n_components=0.95)
X_train_pca = pca.fit_transform(X_train)
X_test_pca = pca.transform(X_test)
```

vs.

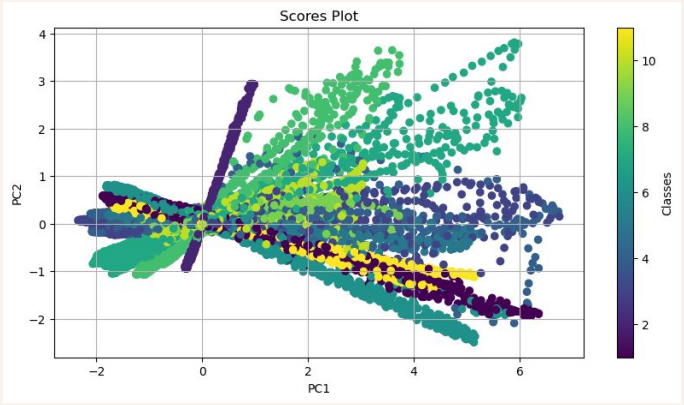
LDA

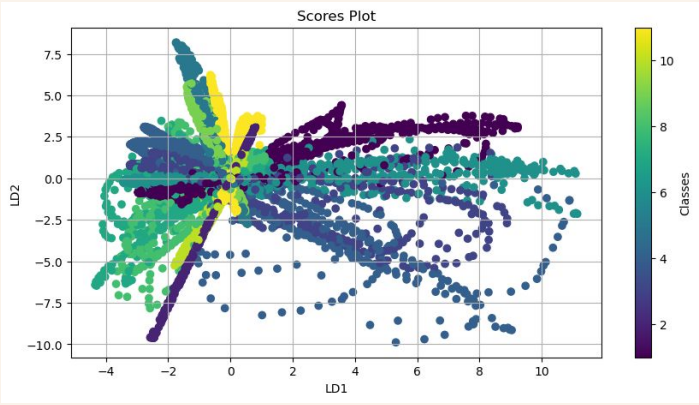
- Supervised
- Maximizes separation between classes in dataset
- Requires features and corresponding class labels
- Reduces dimensionality by maximizing sparability b/t classes
- Ouput: linear discriminants that optimize class separation

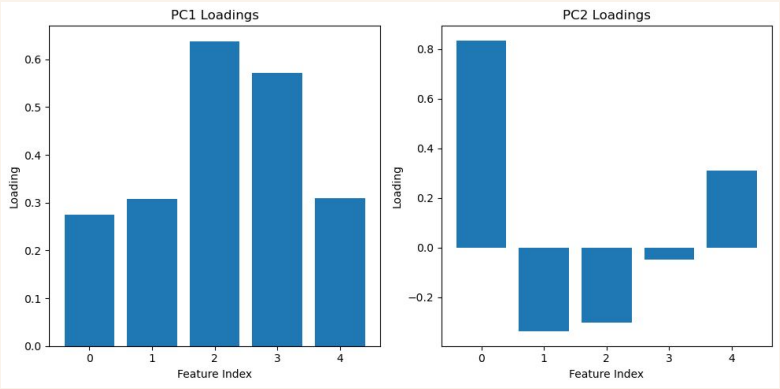
```
from sklearn.discriminant_analysis import
LinearDiscriminantAnalysis as LDA
lda = LDA()
X_train_lda = lda.fit_transform(X_train, y_train)
X_test_lda = lda.transform(X_test)
```

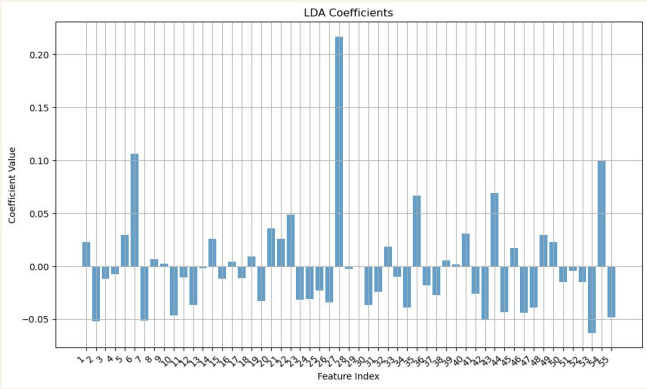
Dimensionality Reduction Results

PCA vs. **LDA**









Classification via SVMs

- SVMs excel in high-dimensional spaces, effectively separating classes using optimal hyperplanes
- Handle non-linear relationships through kernel functions

```
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score

svm classifier = SVC(kernel='rbf', C=1)
svm classifier.fit(X train lda, y train)
y pred = svm classifier.predict(X test lda)
print(classification_report(y_test, y_pred))
```

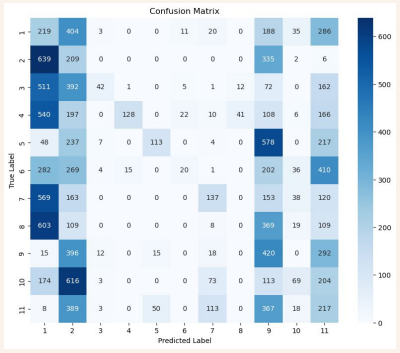
Goal: kernel and hyperparameter optimization

- Kernel: linear, polynomial, radial basis function (rbf), sigmoid
- C: Regularization parameter
- Gamma: Influence of individual training samples
  *higher gamma = smaller range of influence for each training sample → more complex decision boundaries

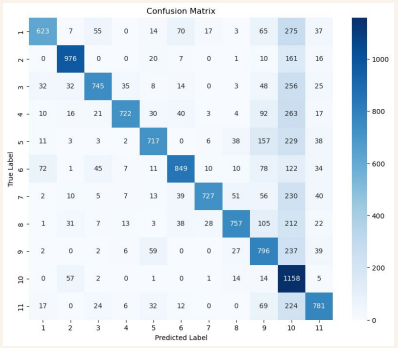
Validation & Testing

```
from sklearn.model_selection import cross_val_score, LeaveOneOut

cv_scores = cross_val_score(svm_classifier1, X_train_lda, y_train, cv=5)

loo = LeaveOneOut()
loo_scores = cross_val_score(svm_classifier1, X_train_lda, y_train, cv=loo)
```

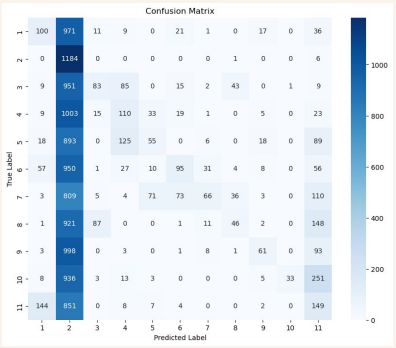
k-fold CV vs. **Leave-One-Out (LOO) CV**



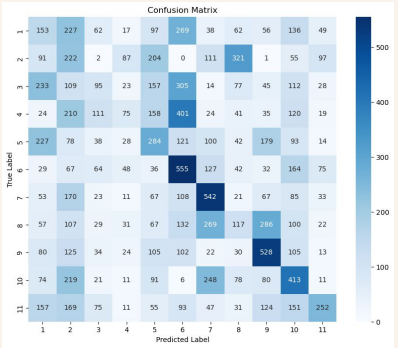
Linear



RBF



Polynomial



Sigmoid

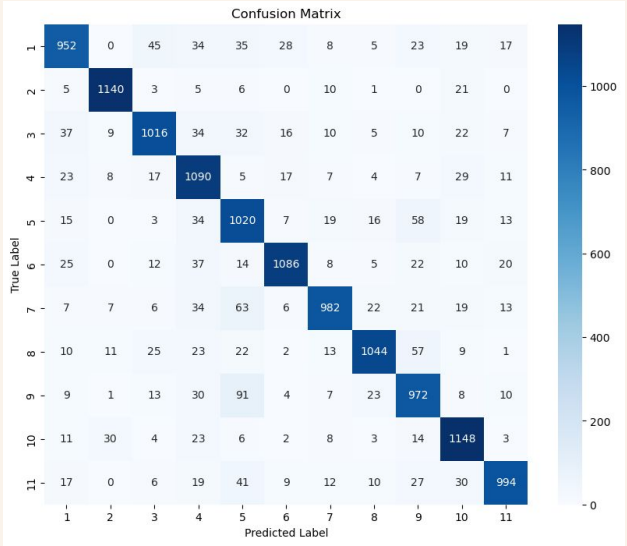
Final Result

```
from sklearn.model_selection import GridSearchCV

param_grid = {
    'C': [0.001, 0.01, 0.1, 1, 10, 100, 1000],
    'gamma': [0.0001, 0.001, 0.01, 0.1, 1, 10],
}
svm_classifier = SVC(kernel='rbf')
grid_search = GridSearchCV(estimator=svm_classifier,
param_grid=param_grid, cv=5, scoring='accuracy')
grid_search.fit(X_train_lda, y_train)
```

Optimized hyperparams: kernel='rbf', C=1000, gamma=10

Overall Accuracy: 0.87
(87% of the instances were correctly classified)

Average Accuracy (k-fold cross-validation): 0.8633



Thank you!