**California State University, Long Beach**
**Department of Computer Engineering and Computer Science**
**CECS 553 Sec 02 11792 (Machine Vision) – Fall 2022**
**Assignment 08 – Tuesday, 11/08/2022**

# Support Vector Machine vs Vanilla Linear Classifier

## Table of Contents

We will be classifying the popular handwritten data set which we can find in the sklearn library and comparing the results of the logistic regression and SVM. In the Sklearn library, there are several ways to use logistic regression for multiclass applications; in this lab, we will use the `multinomial` option; this is like Softmax function we discussed before

- Plotting an Image
- Preprocess data for Logistic Regression
- Logistic Regression with SkLearn
- SVM for Image Classification with SkLearn

Estimated Time Needed: **60 min**

---

## Load Important Libraries and Digit Dataset

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn import datasets, svm, metrics, model_selection
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
```

In [ ]:
```
digits = datasets.load_digits()
```

In [ ]:
```
target = digits.target
flatten_digits = digits.images.reshape((len(digits.images), -1))
```

## Visualize Some Handwritten Images in the Dataset

In [ ]:
```
_, axes = plt.subplots(nrows=1, ncols=5, figsize=(10, 4))
for ax, image, label in zip(axes, digits.images, target):
    ax.set_axis_off()
    ax.imshow(image, cmap=plt.cm.gray_r, interpolation='nearest')
    ax.set_title('%i' % label)
```

## Divide Images into Training and Test Set

I have set the test size to 20% of the total dataset

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(flatten_digits, target,
                                                    test_size=0.2)
```

## Hand-written classification with Logistic Regression

Standardize the dataset to put all the features of the variables on the same scale

In [ ]:
```
scaler = StandardScaler()
X_train_logistic = scaler.fit_transform(X_train)
X_test_logistic = scaler.transform(X_test)
```

Create the logistic regression and fit the logistic regression and use the `l1` penalty. Note here that since this is a multiclass problem the Logistic Regression parameter `multi_class` is set to `multinomial`.

In [ ]:
```
logit = LogisticRegression(C=0.01, penalty='l1', solver='saga', tol=0.1,
                           multi_class='multinomial')
```

In [ ]:
```
logit.fit(X_train_logistic, y_train)
```

In [ ]:
```
y_pred_logistic = logit.predict(X_test_logistic)
```

Get the accuracy of the logistic regression

In [ ]:
```
print("Accuracy: "+str(logit.score(X_test_logistic, y_test)))
```

Lets plot out the confusion matrix, each row of the matrix represents the instances in a predicted class, while each column represents the instances in an actual class.

In [ ]:
```
label_names = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
cmx = confusion_matrix(y_test, y_pred_logistic, labels=label_names)
```

Accuracy is fine and above 80% but we can see some heavily misclassified values, The classifier had a hard time classifying `8`

In [ ]:
```
df_cm = pd.DataFrame(cmx)
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size
title = "Confusion Matrix for SVM results"
plt.title(title)
plt.show()
```

# Hand-Written Classification with SVM

Create and fit the SVM model

In [ ]:
```
svm_classifier = svm.SVC(gamma='scale')
```

In [ ]:
```
svm_classifier.fit(X_train, y_train)
```

Predict for our test set

In [ ]:
```
y_pred_svm = svm_classifier.predict(X_test)
```

Get accuracy for the SVM model, we can see we have a nearly perfect model

In [ ]:
```
print("Accuracy: "+str(accuracy_score(y_test, y_pred_svm)))
```

Let's take a look at the confusion matrix for SVM, we can see a nearly perfect model with SVM

In [ ]:
```
label_names = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
cmx = confusion_matrix(y_test, y_pred_svm, labels=label_names)
```

In [ ]:
```
df_cm = pd.DataFrame(cmx)
# plt.figure(figsize=(10,7))
sns.set(font_scale=1.4) # for label size
sns.heatmap(df_cm, annot=True, annot_kws={"size": 16}) # font size
title = "Confusion Matrix for SVM results"
plt.title(title)
plt.show()
```

## Comparing both SVM and Logistic Regression with K-Fold Cross Validation

k-fold Cross validation is used when there are limited samples, the handwritten dataset contains about 1800 samples, this will give an opportunity for all the data to be in the training and test set at different given times. We will add `l2` regularization to visualize how well they both do against SVM.

In [ ]:
```
algorithm = []
algorithm.append(('SVM', svm_classifier))
algorithm.append(('Logistic_L1', logit))
algorithm.append(('Logistic_L2', LogisticRegression(C=0.01, penalty='l2',
                    solver='saga', tol=0.1, multi_class='multinomial')))


results = []
names = []
y = digits.target
for name, algo in algorithm:
    k_fold = model_selection.KFold(n_splits=10, random_state=10)
    if name == 'SVM':
        X = flatten_digits
        cv_results = model_selection.cross_val_score(algo, X, y, cv=k_fold,
                                                     scoring='accuracy')
    else:
        scaler = StandardScaler()
        X = scaler.fit_transform(flatten_digits)
        cv_results = model_selection.cross_val_score(algo, X, y, cv=k_fold,
                                                     scoring='accuracy')

    results.append(cv_results)
    names.append(name)
```

We plot and we can see that SVM performs better all the time even with k-fold cross validation and it is better than both Logistic regressions on average

In [ ]:

```
fig = plt.figure()
fig.suptitle('Compare Logistic and SVM results')
ax = fig.add_subplot()
plt.boxplot(results)
plt.ylabel('Accuracy')
ax.set_xticklabels(names)
plt.show()
```