Übungsangabe 2 (Zweitantritt)

Sie haben in den letzten beiden Modulen die Grundlagen der Bildverarbeitung mit angewandter Statisik kennenglernt. In dieser Übung sollen Sie diese Anwenden. Lösen Sie bitte folgendes Problem in C++ unter Linux unter der zuhilfenahme von OpenCV 3.4.0.

Schreiben Sie bitte ein C++ Programm welches zwei Buchstaben aus dem EMNIST TEST Datensatz unterscheiden kann (bitte selbstständig herunterladen). Nutzen Sie dazu die PCA und SVM. Sie können die OpenCV PCA und SVM Klasse nutzen. Nutzen Sie eine ν-SVM (nu-SVM) und einen RBF Kernel. Optimieren Sie die Parameter der SVM in einem grid search (siehe Bishoph's machine learning Buch für theoretische Erklärungen).

Nutzen Sie die ersten 1000 Zeilen des Datensatzen zum für den Trainingsprozess. Nutzen Sie weitere 5000 nicht im Trainingsdatensatz erhaltene Zeilen für den Testprozess. Gehen Sie folgendermaßen vor:

- Laden Sie den Datensatz
- Nutzen Sie die beiden letzten unterschiedlichen Buchstaben Ihres Nachnamens für den Datensatz. Sie sollen für diese einen Classifier schreiben.
- Preprocessing
  - Extrahieren Sie die nutzbaren Zeilen
  - Standartisieren Sie den Trainingsdatensatz. Das bedeutet, dass jede Spalte den Erwartungswert 0 und die Standartabweichung 1 haben muss. (siehe Tipps) Diese Werte müssen Sie natürlich aus dem Trainignsdatensatz berechnen.
  - Reduzieren Sie den standartisierten Trainingsdatensatz mit der PCA auf eine adäquate Größe. Argumentieren Sie warum Sie sich für wie viele latente Dimensionen entschieden haben
  - Wenden Sie die vorverarbeitung (vorhergehenden beiden Punkte) auch auf den Testdatensatz an (Skalierung und PCA)
- Training SVM
  - Parameter optimieren
  - Performance berechnen (accuracy, ACC)
- Formatierung der Ausgabe
  - Formatieren Sie die ausgabe am Bildschirm so, dass **ausschließlich**: "Iteration n: ACC" pro Iteration erkennbar ist (sie müssen für n und ACC die tatsächlichen Werte einsetzen)

Ihre muss aus Ihrer C++ Quelldatei und einem Makefile bestehen. Sollte das Programm nicht kompilieren/linken bekommen Sie 0 Punkte auf die Abgabe. Jede Abgabe wird einem Plagiatstest unterzogen. Sie bekommen Punkte für Codequalität (sauberes C++ programmieren, siehe MMR1 MPK) Lesbarkeit Ihrer Abgabe, Funktionsfähigkeit und Richtigkeit der Implementierung. Bei Unregelmäßigkeiten und Fragen kann es zu einem Abgabegespräch kommen. Ihre Punkte basieren dann auf diesem Abgabegespräch.

Laden des Datensatzes

Sie können folgende Codezeile zum Laden des Datensatzes nutzen:

```
cv::Ptr< cv::ml::TrainData > tdata = cv::ml::TrainData::loadFromCSV( "./mnist_test.csv", 0, 0, 1 ); // First col is the target as a float
cv::Mat samples = tdata->getTrainSamples( );                                                          // Get design matrix
cv::Mat target = tdata->getTrainResponses( );                                                         // Get target values
```

Dies ladet den gesamten MNIST TEST Datensatz in die Matritzen.

Umgang mit der PCA

Sie können folgende Codezeile für die Implementierung der PCA nutzen:

```
cv::PCA pca_analysis( trainX, cv::Mat( ), cv::PCA::DATA_AS_ROW, PCADim ); // Estimate PCA
```

Die *project* methode projiziert Vektoren in den latenten Raum.

Skalierung/Vorverarbeitugn der Merkmale

Hier finden Sie **nicht funktionsfähigen** Code als Ausgangslage für die Standartisierung. Sie können auch integrierte OpenCV Funktionen verwenden:

```
mu = cv::Mat( X.cols, 1, CV_32F );  // Memory for the means
var = cv::Mat( X.cols, 1, CV_32F ); // Memory for the scale
for( int i = 0; i < X.cols; i++ ) {
    cv::Mat col = X.col( i );                          // Get column
    cv::Mat M, S_, S;                                  // Memory for mean and scale/standard deviation
    cv::meanStdDev( col, M, S_ );                      // Get scale/Standard deviation
    mu.at< float >( i ) = static_cast<float>(M.at< double >( 0 ));         // Store mean
    cv::meanStdDev( col - mu.at< float >( i ), M, S ); // Get scale/Standard deviation
    ... Get scale = std. dev. in var
    /* FANGEN SIE HIER NUMERISCHE PROBLEME von S AB (zu klein) - SONST FUNKTIONIERT DIE STANDARTISIERUNG NICHT*/
}
...
void transform( cv::Mat& X, cv::Mat& X_skaliert ) {...}
```

Achtung: Beachten Sie Ihre Datentypen!

Tipps

- Achten Sie auf OpenCV Datentypen (32: float, 64: double) - Sie müssen den selben Datentyp selber überall gleich definieren
- Das Vorgehen bei der Standartisierung ist: (1) Mittelwerte pro Spalte berechnen, (2) korrekten Mittelwert von den Spalten abziehen, (3) Stadardabweichung der Spalten berechnen, (4) Spalten durch die Standardabweichung dividieren. Sie können dazu die *cv::meanStdDev* verwenden.

Zuletzt geändert: Freitag, 4. Februar 2022, 15:53