OpenCV and machine learning

- Reading: Kaehler and Bradski, Learning OpenCV3, parts of ch. 21
- Installing OpenCV, project properties
- Classification and decision trees in OpenCV
- Basic training, prediction

Installing OpenCV

- Self-extracting archive
- Environment variables:
  - OPENCV_DIR = {installation folder}\build\{arch}\{version of Visual C++}
    - Ex: C:\Users\Dell\Downloads\opencv\build\x64\vc14
  - Add “bin” subfolder of OPENCV_DIR to PATH
    - Note: there is another bin folder in the distribution under the “build” folder – that has different libraries

Project settings

- Make sure target platform is consistent with installed libraries (for preceding slide, use “x64” target)
- Under C/C++, add “…build\include” as additional include folder (for header files)
- Under Linker, add the same folder you added to your PATH
  - Documentation mentions a lot of library files to add as Input’s, but it seems they have been replaced by one combined library file
    - opencv_world331.lib or opencv_world331d.lib, depending on release or debug target

Project settings

- For Visual Studio, Property Manager can store settings
  - View > Other Windows > Property Manager
  - Saves in “.props” file
  - can “Add existing property sheet” to a new target
- See: Working with Project Properties

OpenCV classification

- OpenCV offers many other options for classification, including Support Vector Machines, Naïve Bayes, neural networks
  - All use a similar setup

OpenCV decision trees

- OpenCV decision trees are a bit different from the description in the text
  - Only binary trees are supported
    - Have to adjust how features of data are handled accordingly
  - Choice of feature at each node based on *gini impurity*
    - = How likely a random item would be misclassified if assigned a class based only on the overall frequency of each class
    - = $\Sigma\ p_i\ (1 - p_i)$ for all classes i

Some comments on OpenCV types

- OpenCV has its own String class (https://stackoverflow.com/questions/45856079/cvstring-and-stdstring-when-to-use-which-one-and-necessity-to-use-both -- OpenCV started as a C library, and it's old)
- OpenCV has a Ptr class that manages its own memory
  - C++ now has a similar "smart" pointer, but again, too late for OpenCV to easily change to it

Some comments on OpenCV types

- CV_32F: 32 bit floating point values
  - To standardize size of float values (standard C++ now has these, too)
- InputArray: matrix
  - Can generate this from OpenCV's Mat_, Matx types, as well as from std::vector's
- Feature (attribute) values are stored as floats
  - If only uniqueness matters, just casting to CV_32F should work ok
  - Alternative for categorical data: replace feature with N categories by N binary features
    - Ex: for (main) type of food at a restaurant, replace Type by IsItalian, IsThai, IsEthiopian, …

Algorithm class

- Provides general methods for loading, saving, clearing data in models
- virtual void save(const String & filename) const;
- template<typename _Tp> static Ptr<_Tp> load(const String & filename, const String & objname = String());
  - Ex: from API docs: Ptr<SVM> svm = Algorithm::load<SVM>("my_svm_model.xml");
- virtual void clear();

StatModel class

- Child class of Algorithm
- Defines general methods for all machine learning algorithms
- General steps:
  - Train the model
  - Predict responses for a given set of data

Training

- static Ptr<TrainData> TrainData::create(InputArray samples, int layout, InputArray responses [, 4 optional parameters]);
  - Samples = matrix of data point values (CV_32F)
  - Layout
    - ROW_SAMPLES = each row is a data point (feature vector), each column for values of one feature
    - COL_SAMPLES = each column is a data point
  - Responses: depends on which machine learning algorithm
    - For decision trees, categorical responses must be CV_32SC1 (32-bit signed int)

Training

- static Ptr<TrainData> TrainData::loadFromCSV(const String & filename, int headerLineCount, responseStartIdx = -1, responseEndIdx = -1, varTypeSpec=String(), char delimiter=‘,’, char missch=‘?’)
  - Filename – regular string is fine, check IDE for default folder
    - Ex: Visual Studio: project folder
  - headerLineCount = number of lines of header information
  - responseStartIdx = column for correct response, -1 = last
  - responseEndIdx = 1 + last column for response
  - varTypeSpec = “ord[<list of columns with ordered data>]cat[<list of columns with categorical data]” – ex: “ord[0-2,4]cat[3,5-9]”
  - missch = character for missing feature value

Examining training data

- TrainData supports getTrainSamples() which returns a Mat of values
- Mat supports at<*type*>(row, col) method to access data as a *type*

Data format

- *Channels* refer to multiple, independent streams of data
  - in image processing and vision, refers to situations like RGB format providing red, green and blue channels
  - For machine learning, not used

DTrees model

- Model stored in object of type Ptr<DTrees>
  - Besides supporting decision tree model, parent class of other machine learning methods which may use a forest of trees as a model
  - Construct and then set properties
  - train(Ptr<TrainData>) to do the training

Prediction

- float predict(InputArray sample [, 2 optional parameters])
  - Sample is same format as training data
  - For single row, predict() returns result (for decision tree, this is the label/class)
  - For multiple rows, predict() returns results in 2nd parameter (type OutputArray)