50.039 – Deep Learning

Week 01: Discriminative ML - Quick Intro

[The following notes are compiled from various sources such as textbooks, lecture materials, Web resources and are shared for academic purposes only, intended for use by students registered for a specific course. In the interest of brevity, every source is not cited. The compiler of these notes gratefully acknowledges all such sources.]

Run a small ML project. The goal is to let you do a bit recap on how to run a machine learning project in principle.

- Download from eDimension the **imagefeatures.zip**, the set of all labels in **label_info.txt** and the image labels **gtlabels.txt.**
- Split the images into **65** percent per class for training, **15** percent per class for validation, **20** percent per class for final test.
- Use a linear SVM (e.g. scikit-learn) with Python3 interface. Besides using the linear kernel, try another kernel also, to compare its result with that of the linear SVM and give your analysis (once we have some experimental results, we often need to give some comparison and analysis).
- For this first experiment, take only the features from those images who have as labels either **Spring** or **Summer** or **Autumn**, i.e., using a sub-set (***1145*** samples) of these samples.
- That results in a multi-class dataset with mutually exclusive labels. Thus you can train **3 binary SVMs**, one for each class in one-vs-all manner. Each SVM is trained on the training dataset using all the training data. Do not directly use some high-level one-vs-rest wrapper.

- At test time you predict the class-label as the index of the svm with the highest prediction score.
- This method has one free parameter - the regularization constant. Find the best regularization constant from the set 0.01, 0.1, $0.1^{0.5}$, 1, $10^{0.5}$, 10, 100 by repeatedly training on the training set and measuring performance on the validation set. Use as performance measure the **class-wise accuracy averaged** over all **3** classes.
- Once you have the best value for C, train on *train + validation*, then report performance of that classifier on the test set.
- Submit your code (python3), including the code for splitting, and your saved train val test splits (.npy),
- Submit a short report showing the **two performance measures** on validation and test sets, and give your analysis about the purpose of using two measures here:
  (1) Vanilla Accuracy

$$\frac{1}{n}\sum_{i=1}^{n} 1[f(x_i) == y_i]$$

  (2) Class-wise averaged accuracy:

$$\begin{aligned} A =& \frac{1}{C}\sum_{c=1}^{C} a_c \\ a_c =& \frac{1}{\sum_{i=1}^{n} 1[y_i == c]} \sum_{i=1}^{n} 1[y_i == c] 1[f(x_i) == c] \\ =& \frac{1}{\sum_{(x_i,y_i):y_i=c} 1} \sum_{(x_i,y_i):y_i=c} 1[f(x_i) == c] \end{aligned}$$

Useful things in python:

File handling:

```
os.path.basename(...)
os.path.join(...)
os.path.isdirectory(...)
os.makedirs()
```

for reading one line of the labels file trainset_gt_annotations.txt:

```
line.rstrip().split()
```

returns you a list where the first entry is the filename, and the following 99 entries are the labels as strings. they need still to be converted into integers

also helps to extract parts from a filename:

```
position=pythonstring.find(someotherstring)
```

loading of numpy arrays:

```
np.save(...)
np.load(...)
```

```
sklearn.svm.*
yourwhateversvm.predict(...) does not help you here,
```

you need the real valued scores, not the label from one one-vs-all SVM

Offtopic: you can try other sets of labels. Some do not deliver good performance when measured by class-wise accuracy.