1 Overview

dataloader.py implements a general purpose data loader for python non-sequential machine learning tasks. Given a directory path the data loader will read data files (in commonly found formats) and return a python object containing the data in the form of numpy matrices, along with some supporting functions for manipulating the data. The data loader supports the loading of data sets containing multiple feature inputs, and multiple target labels.
2 Data Directory Structure

The directory at the top level can be named whatever. There must be three directories below the top level data directory named 'train', 'dev', and 'test'. If the 'train', 'dev', and 'test' directories are not present an error will be raised during loading. The top level directory may contain other files besides these three directories.

$N$ is the number of feature sets. Not to be confused with the number of elements in a feature vector for a particular feature set.

$Q$ is the number of label sets. Not to be confused with the number of elements in a label vector for a particular label set.

$F_{\text{trn}}_n$ is a file containing the $n$th set of training features, and should be named

features\_descriptor.ext

where descriptor is a placeholder for some appropriate brief name for the feature set and ext is a placeholder for some extension corresponding to one of the supported formats.

$L_{\text{trn}}_n$ is a file containing the $n$th set of training labels, and should be named

labels\_descriptor.ext

where descriptor is a placeholder for some appropriate brief name for the label set and ext is a placeholder for some extension corresponding to one of the supported formats.

$F_{\text{dev}}_n$, $L_{\text{dev}}_n$, $F_{\text{tst}}_n$, and $L_{\text{tst}}_n$ are files for corresponding features and labels in the dev and test folders and have naming conventions as above.

2.1 Notes:

Rows of feature and data matrices should correspond to individual data points as opposed to the transpose. There should be the same number of data points in each file of the train directory, and the same is true for the dev and test directories. The number of data points can of course vary between directories.

3 DataSet Class

- Attributes:

  features: A hash map (python dictionary) of feature matrices. The key to a matrix for a data file named:
features_descriptor.extension

is the string 'descriptor'.

labels: A hash map (python dictionary) of label matrices. The key to a matrix for a data file named:

labels_descriptor.extension

is the string 'descriptor'.

- Methods:

  next_batch(num_examples) returns a DataSet object containing the next num_examples rows of the feature matrices and label matrices. If num_examples is greater than the number of data points in the the data stream a python assert fails and the loader stops. If num_examples is greater than the number of examples left in the epoch then all the matrices in the data stream are shuffled and a DataSet object containing the first num_examples rows of the shuffled feature matrices and label matrices is returned.

4  Loader Function

read_data_sets(<directory>) takes as input a string of the data directory path where your data is located. It will return an record of DataSet objects which can be accessed for a record called someData by

someData.train to access the data read from the train folder,

someData.dev to access the data read from the dev folder, and

someData.test to access the data read from the test folder.

4.1  Supported Formats

.mat files

.sparsetxt files are plain text files where lines correspond to an entry in a matrix and have the form i j k, so that the matrix A is such that \( A_{ij} = k \). Tokens must be whitespace delimited.

.densetxt files are plain text files with a matrix represented in standard form. Tokens must be whitespace delimited.

.sparse files are like .sparsetxt files but written in binary (no delimiters) to save disk space and speed file i/o. Matrix dimensions are contained in the first line of the file.

.binary files are like .densetxt files but written in binary (no delimiters) to save disk space and speed file i/o. Matrix dimensions are contained in the first line of the file.
4.2 Exceptions

**Bad_directory_structure_error** is raised when the data directory specified does not contain subfolders named ‘train’, ‘dev’, and ‘test’. Any of these directories could be empty and the loader will hand back a DataSet object containing no data which corresponds to the empty folder.

**Unsupported_format_error** is raised when a file with name beginning ‘labels’ or ‘features’ is encountered without one of the accepted file extensions listed above. It is okay to have other files in your directory as long as their names don’t begin with ‘labels’ or ‘features’.

**Missing_data_error** is raised if there is not at least one features file and one labels file found in the directory.

**Mat_format_error** is raised if the .mat file being read does not contain a variable named ‘data’.

**Sparse_format_error** is raised when reading a plain text file with .sparsetxt extension and there are not three entries per line.

**Mismatched_data_error** is raised if there is a mismatch in the number of rows between two matrices of a DataSet object. The number of rows is the number of data points, or examples, and this loader assumes that each example will have each feature set and each label set. If you have missing labels or missing features for a particular example they may be substituted with some appropriate sentinel value.

5 Possible Extensions

- Data transformations:
  - Mean Cancellation, KL-Expansion, Covariance Equalization, Data Whitening, shift labels to avoid asymptotes of logistic.

- Feed:
  - Advanced shuffling, variable batch size.

- Data sets:
  - Support for sequential and tensor data.

6 datatest.py

A utility that was made for testing the dataloader datatest.py also works well for testing the correctness of your processed data. datatest.py calls read_data_sets from the dataloader.py module and prints the dimensions and keys of the feature and label matrices in your dataset. To run datatest.py on a correctly formatted folder give the folder path as an argument. Example usage:
5

train:
121451 data points
features:
  tfidf: (121451, 7669)
  row_one_hots: (121451, 109315)
  word_freq: (121451, 7669)
  col_one_hots: (121451, 5892)
labels:
  values: (121451, 1)

dev:
8562 data points
features:
  tfidf: (8562, 7669)
  row_one_hots: (8562, 109315)
  word_freq: (8562, 7669)
  col_one_hots: (8562, 5892)
labels:
  values: (8562, 1)

test:
8562 data points
features:
  tfidf: (8562, 7669)
  row_one_hots: (8562, 109315)
  word_freq: (8562, 7669)
  col_one_hots: (8562, 5892)
labels:
  values: (8562, 1)