PROGRAMMING IN PYTHON II

Data Analysis and Preprocessing

Andreas Schörgenhumer

Institute for Machine Learning

JOHANNES KEPLER UNIVERSITY LINZ

Institute for Machine Learning

Contact

Andreas Schörgenhumer

Institute for Machine Learning
Johannes Kepler University
Altenberger Str. 69
A-4040 Linz

E-Mail: `schoergenhumer@ml.jku.at`
Write mails only for personal questions
Institute ML Homepage

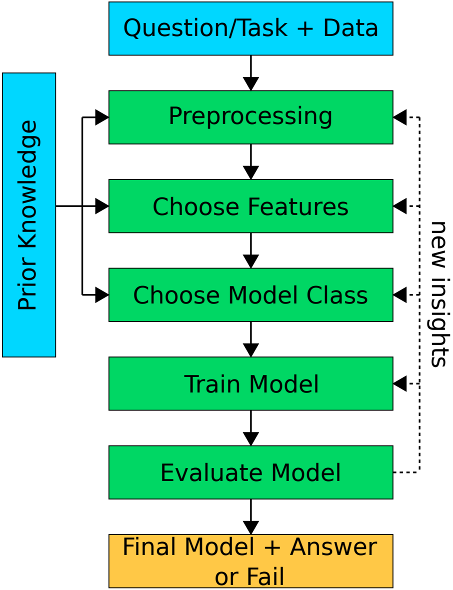
Copyright Statement

This material, no matter whether in printed or electronic form, may be used for personal and non-commercial educational use only. Any reproduction of this material, no matter whether as a whole or in parts, no matter whether in printed or in electronic form, requires explicit prior acceptance of the authors.

Terminology

- **Model**: parameterized function/method with specific parameter values (e.g., a trained neural network)
- **Model class**: the class of models in which we search for the model (e.g., neural networks, SVMs, ...)
- **Parameters**: what is adjusted during training (e.g., network weights)
- **Hyperparameters**: settings controlling model complexity or the training procedure (e.g., network learning rate)
- **Model selection/training**: process of finding a model (optimal parameters) from the model class

Basic Data Analysis Workflow



Motivation

- We want to train a machine learning (ML) model such that we get a “good” or even the “best” model
- How do we get the “best” model?
  1. How does our model perform on our data?
     $\rightarrow$ **Loss function**
  2. How will it perform on (unseen) future data?
     $\rightarrow$ **Generalization**

Generalization – Theory

- We train our model on a subset of data points (e.g., to predict labels)
  - We use **Empirical Risk Minimization (ERM)**
  - This subset of data points is called **training set**
- We want this trained model to also work on (e.g., correctly predict) unknown/future data
  - Problem: We might fit our parameters to noise specific to our training data set (=**overfitting**)
  - We can use a separate subset of samples to estimate the (true) risk on unknown data (=how well our model **generalizes**)
  - This separate subset of data points is called **test set**

Generalization – Assumptions

- Of course, there is a price to pay: The theory comes with assumptions:
  1. Strong law of large numbers: Our **subset of data points has to be large enough**
  2. Our data points have to be **independently and identically distributed (i.i.d.)**
- What does i.i.d. mean?
  - Each sample has the same probability distribution as the others and all are mutually independent.

i.i.d. With Respect to Our ML Project

- What we want:
  - We want our model to perform image depixelation on all kinds of pictures within certain restrictions (size, color)
  - The distribution our pictures are sampled from should be that of all possible pictures within these restrictions
  - The pictures should be sampled randomly from this distribution of all possible pictures
- What we have:
  - We collected 100 pictures per student
  - $\rightarrow$ The 100 pictures per student are probably not mutually independent
  - The pictures are not sampled from the distribution of all possible pictures but from another distribution (European setting, ML-students, $\ldots$)
  - $\rightarrow$ Pictures are not randomly drawn from the true distribution of all possible pictures!

Working With What We Have – Theory

- We need to consider the violations of i.i.d. properties in our data
- Training set and test set splitting must reflect this consideration
  - Test set must be drawn independently from training set (or as independently as possible) to get a good estimate of true risk
  - Preprocessing must not violate test and training set split
  - Data analysis done on complete set of data points cannot be used for training

Working With What We Have – Practice

- Example: our ML project
  - Random assignment of samples to training and test set will not be sufficient! (reason: not independently sampled!)
  - Better: Assign samples of one set of students to the training set and those of other students to the test set
  - In general: Assign samples of one set of clusters to the training set and those of other clusters to the test set if you want an estimate for generalization between clusters!
- Even then, we will not get rid of the problem that we did not sample correctly from the true distribution of all possible pictures
  - We do not know how well our model performs on this true distribution of all possible pictures
- Keeping that in mind, let's try our luck and get started!

Cleaning Up

- Never assume the data is valid or correctly formatted
- Typical problems:
  - Empty or corrupted files
  - Wrong filetypes
  - Duplicated datapoints
  - Missing datapoints
  - Outlier values
  - Inconsistent filenames/sample names
  - Inconsistent label names
  - Incorrect labeling
  - ...

First Analysis

- Check mean/standard deviation of data points
- Check number of valid samples
- Check number of classes and valid labels
- If applicable, visualize (parts of) the data set

Data Preprocessing

- What violates the training and test split?
  - Do not compute global values for the whole data set for normalization!
  - Do not perform feature-selection on the whole data set!
- What preprocessing should be done once and saved and what should be done on-the-fly?

Normalization

- Many ML methods profit from normalized data
  - Make data more homogeneous
  - Reduce chances to overfit
  - Some methods require a specific normalization
- Different normalization schemes for different settings/tasks
  - Typical for NN: Mean=0, Variance=1 (often referred to as standardization)
- Clustering and down-projection methods also benefit from normalized data

Normalization – Common Approaches

- Normalization per sample
  - Mean and variance computed and normalized per sample
  - Does not violate data set splits
  - Removes offsets of samples (e.g., brightness in images)
- Normalization per data set
  - Mean and variance computed over all samples in data set and then used for normalization
  - Violates data set splits! $\rightarrow$ Mean and variance need to be computed on training set and these values should be used for other sets too!
  - Keeps offsets of samples

Normalization – Other Approaches

- Other approaches for normalization or scaling
  - Scaling values to range $[0, 1]$ for each sample or complete data set
  - Scaling values to range $[-1, 1]$ for each sample or complete data set
- Best normalization/scaling depends on the data set, method and task

Normalization – Tips and Tricks

- Check the publication of the method you are applying for theory/recommendations
- You can evaluate different normalization schemes on another separated set (=**validation set**)
- Using pretrained models? If your data is similar, you can often keep the normalization constants from the pretraining

Optimization

- Prepare data such that we do not need to convert it before feeding it to our models
- Load data set in RAM if possible to decrease loading time
- Compress data set to save disk space
  - Max. number of files per directory, max. size per file, max. length of file paths depend on file system/OS

Clustering and Down-Projection (1)

- After normalization, look into clustering and down-projection methods
  - They often give us valuable insights in the data
  - If you use such clusters to create test and training splits, verify them manually! (Do not trust clustering methods.)
- Popular clustering methods: k-means, DBSCAN, ...
- Popular down-projection methods: PCA, t-SNE, ...
- Our raw data might be incompatible with these methods
  - Too many feature values
  - Datapoints in odd feature space
  - $\rightarrow$ We need to be creative

Clustering and Down-Projection (2)

- For our ML project, for instance, we would like to inspect whether there are certain clusters/patterns in our data
- What we want:
  - Small suitable feature space
  - Constant number of features
- What we have:
  - Huge feature space (number of pixels)
  - Odd feature space (pixel space)
  - Images of different size (different number of features)
- Possible solution:
  - Down-project images into better feature space before clustering and/or visualization (e.g, using pretrained CNN features)

  $\rightarrow$ Fewer features, constant number of features, better feature-space (hopefully)