PROGRAMMING IN PYTHON II

Data Analysis and Preprocessing

Michael Widrich
Institute for Machine Learning

JOHANNES KEPLER
UNIVERSITY LINZ

Institute for
Machine Learning

Outline

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, ...)

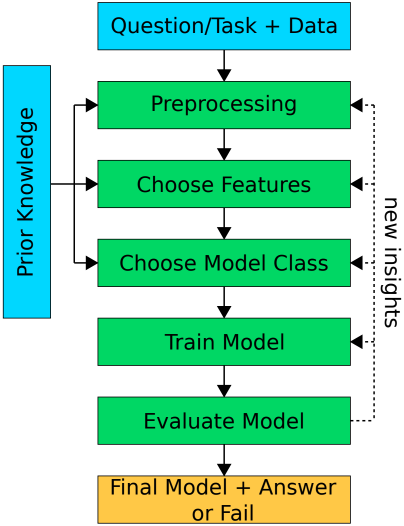
Parameters: representations of concrete models inside the given model class (e.g. network weights)

Hyperparameters: parameters controlling model complexity or the training procedure (e.g. network learning rate)

Model selection/training: process of finding a model from the model class

Institute for Machine Learning

Basic data analysis workflow



Motivation

- We want to train a 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? – Loss function
    2. How will it perform on (unseen) future data? – Generalization

Generalization – Theory

- Generalization is something humans hope for every day

  . . . but sometimes fail at
- Generalizationin a nut-shell:
  - 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 dataset (= over-fitting)
    - We can use 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 in practice? (1)

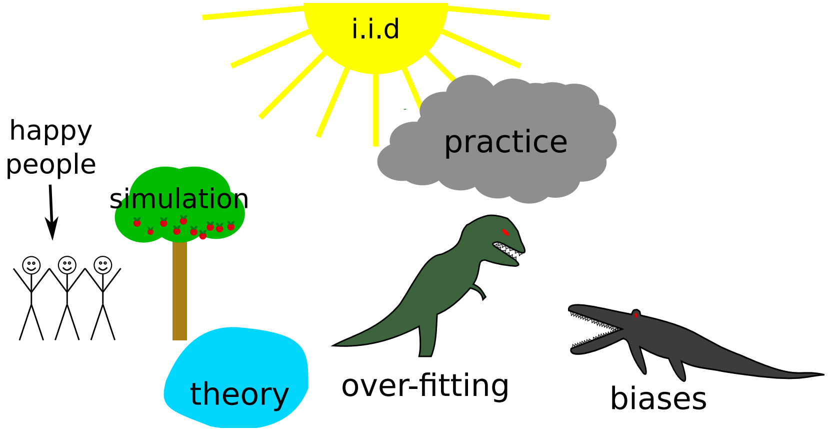
- Independently and identically distributed (i.i.d.):

  Each sample has the same probability distribution as the others and all are mutually independent.

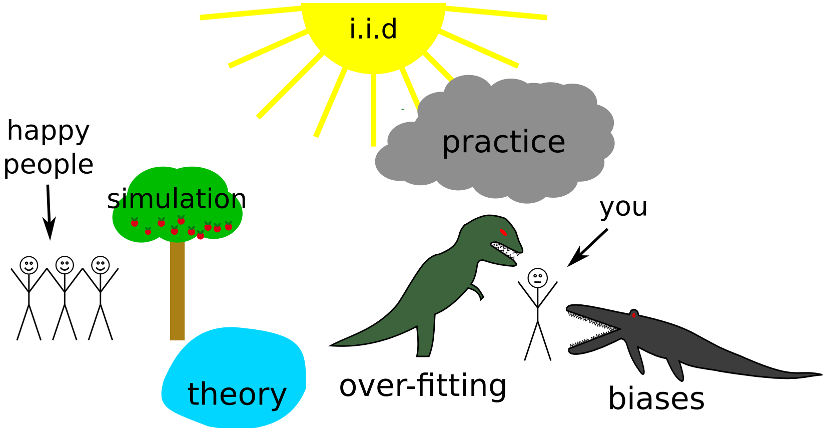
What does i.i.d. mean in practice? (2)

- Example: Our ML project
  - What we want
    - We want our model to perform data imputation 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 within these restrictions
  - What we have
    - We collected 100 pictures per student
    - $\rightarrow$ The 100 pictures per student are probably not mutually independent
    - Pictures not sampled from distribution of all possible pictures but other distribution (European setting, ML-students, . . .)
    - $\rightarrow$ Pictures are not randomly drawn from the true distribution of all possible pictures!

You and i.i.d. in practice



You and i.i.d. in practice



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! (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 estimate for generalization between clusters!
- Even then we will not get rid of the problem that we did not sample correctly from distribution of all possible pictures
  - $\rightarrow$ We do not know how well our model performs on distribution of all possible pictures
- Keeping that in mind, let's try our luck and get started!

Cleaning up

- Never trust in that the data is valid or correctly formatted
- Typical problems
  1. Empty or corrupted files
  2. Wrong filetypes
  3. Duplicated datapoints
  4. Inconsistent filenames/samplenames
  5. Inconsistent label names

$\rightarrow$ Exercise 2 will be on cleaning up the data

First analysis

- Check mean/standard deviation of data points
- Check number of valid samples
- Check number of classes and valid labels
- Our ML project: ~27k valid samples from ~275 students

Data preprocessing

- What violates the training and test split?
  - Do not compute global values for the whole dataset for normalization!
  - Do not perform feature-selection on the whole dataset!
- 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 over-fit
  - Some methods require a specific normalization
- Different normalization schemes for different setting/tasks
  - Typical for NN: Mean=0, Variance=1
- Clustering and down-projection methods also benefit from normalized data
- Exercise 3 will be on normalizing the data

Normalization – Common approaches

- Normalization per sample
  - Mean and variance computed and normalized per sample
  - Does not violate dataset splits
  - Removes offsets of samples (e.g. brightness in images)
- Normalization per dataset
  - Mean and variance computed over all samples in dataset and then used for normalization
  - Violates dataset splits! – 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 dataset
  - Scaling values to range $[-1, 1]$ for each sample or complete dataset
- Best normalization/scaling depends on the dataset, method, and task
- Sometimes fancy normalization/scaling is used to implicitly scale network activations and/or learning rates

Normalization – Tipps 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 pre-trained models? If your data is similar you can often keep the normalization constants from the pre-training
- Calculation of standard deviation can lead to numerical inaccuracy when using fewer bits

Optimization

- Prepare data such that we do not need to convert it before feeding it to our models
- Compress dataset to save disk space
- Load dataset in RAM if possible to decrease loading time
- Use folders to structure data files or load them into one container, e.g. hdf5
    - Max. number of files per directory, max. size per file, max. length of file paths depending on filesystem/OS

Clustering and Down-Projection (1)

- After normalization, look into clustering and down-projection methods
  - Give us valuable insights in the data
  - If you use such clusters to create test and training split, verify them manually! (Do not trust clustering methods.)
- Popular methods: PCA, ICA, t-SNE, UMAP, ...
- Our raw data might be incompatible with these methods
  - Too many feature values
  - Datapoints in odd feature-space
  - → We need to be creative

Clustering and Down-Projection (2)

Example: Our ML project

- We would like to use t-SNE or UMAP for clustering
- 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)

Clustering and Down-Projection (3)

- Possible solution:
  - ☐ Down-project images into better feature-space before clustering using pretrained CNN features
  - $\rightarrow$ Less features, constant number of features, better feature-space (hopefully)
  - ☐ Alternative: PCA or ICA

Outlook

- Next time: Implementing and optimizing data loading using PyTorch