Statistical Learning

Due on Moodle 36 hours before the discussion day

Final Homework

A) Data and Goal

What and where

For this final homework you will work on a reduced version of the dataset I prepared for last year SL-Hacka. It is already divided in train and test and can be downloaded from here. Although not huge, to import everything in `R` I strongly suggest to have a look at the `data.table` package and in particular to the function `fread()`.

The Task: Tempo Estimation

If you ever tried to play around with vinyl records and classic heavy-duty turntables, or even with lighter, shiny and more contemporary DJ-gears, the bare minimum you had to learn was how to sync the beat of two songs and come up with a smooth, silky transition between them: master that and you're halfway closer to success, as simple as that!

The musical aspects of tempo, beat, and rhythm play a fundamental role for the understanding of and the interaction with music. It is the beat, the steady pulse that drives music forward and provides the temporal framework of a piece of music. Intuitively, the beat can be described as a sequence of perceived pulses that are regularly spaced in time and correspond to the pulse a human taps along when listening to the music. The term tempo then refers to the rate of the pulse. Musical pulses typically go along with note onsets or percussive events. Locating such events within a given signal constitutes a fundamental task, which is often referred to as onset detection.

While many different **tempo estimation** techniques have been proposed, some comparative studies suggest that there has been a relatively small improvement in the state of the art recently. Current approaches for tempo estimation focus on the analysis of mainstream popular music with clear and stable rhythm and percussion instruments, which facilitates this task. These approaches mainly consider the periodicity of intensity descriptors to locate the beats, and then to estimate the tempo. Nevertheless, they usually fail when they are analyzing other music genres like classical music, because this type of music often exhibits tempo variations.

Contrary to many existing systems, which typically first identify beats and then derive a tempo, here you will try to estimate the tempo directly and blindly (read "ignorantly") from generic spectral signatures extracted from 8 seconds long song-snippets

Evaluation

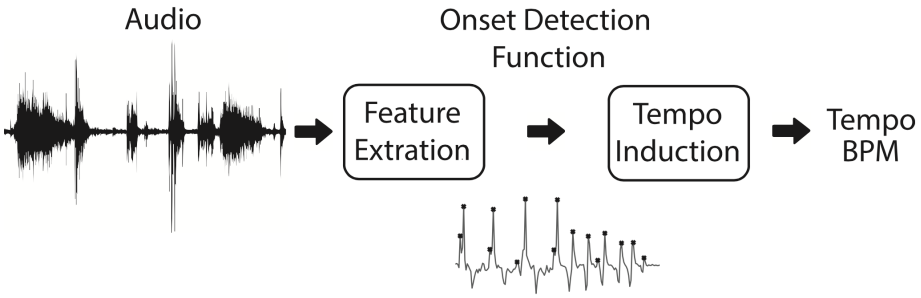
Tempo estimation can be casted as a **regression** or as a **multi-class classification** problem. Although you can "internally" work with the latter, to make things simple and familiar your submission will be scored via **RMSE**, as in a regression problem. This choice is clearly a suboptimal in this context since, as you may imagine, many musical factors can easily affect your prediction.

For the sake of completeness I will mention the two most commonly used metrics:

- `Accuracy1`: it measures if an estimated tempo is within +/- 4% of the ground truth.
- `Accuracy2`: the same as `Accuracy1` but considering octave errors (multiple of the ground truth) as correct.

Data Description

The general scheme for automatic tempo estimation methods is depicted below. It represents the main steps to estimate the tempo of an audio signal.



This scheme includes a **feature list creation block** and a **tempo induction block**: I give you (mainly) spectral features extracted from 8 seconds long song-snippets, you build the other block! I will not bother you with the details, but I'll provide here a sketch of what's going on in the first box.

In this feature creation step (that I performed for you) we transform the audio waveform into a series of features representing (hopefully) predominant rhythmic information. Often, given a music recording, a short-time Fourier transform (STFT) is used to obtain a spectrogram. One simple, yet important step, which is often applied in the processing of music signals, is referred to as *logarithmic compression* and it consists in applying a logarithm to the magnitude *spectrogram* or *energy* of the signal. Such a compression step not only accounts for the logarithmic sensation of human sound intensity, but also balances out the dynamic range of the signal (for more details, see MEL-frequency cepstrum).

As you will see, the train dataset has 7042 columns/features (a bit too-much, don't you think?) that can be broken down in the following way:

1. From 1 to $171 \times 40 = 6840$: Mel-frequency cepstral coefficients of an 8 secs song-snippet sampled at 11025 (windows size = 1024 samples, overlap = 50%, number of mel-bands = 40). The result is a matrix with 171 rows (time instants) and 40 cols (MEL-frequencies) that has been column-vectorized (i.e. by stacking its 171-long columns one after the other).
2. From 6841 to 7011: Tracking of the dominant frequency (the frequency of highest amplitude) over 171 time instants.
3. From 7012 to 7019: 8 summary statistics extracted from the STFT (`time.P1` = the time initial percentile, `time.M` = the time median, `time.P2` = the time terminal percentile, `time.IPR` = the time interpercentile range, `freq.P1` = the frequency initial percentile, `freq.M` = the frequency median, `freq.P2` = the frequency terminal percentile, `freq.IPR` = the frequency interpercentile range).
4. From 7020 to 7033: 14 statistical properties of a frequency spectrum (`mean` = mean frequency, `sd` = standard deviation of the mean, `sem` = standard error of the mean, `median` = median frequency, `mode` = modal frequency, `Q25` = first quartile, `Q75` = third quartile, `IQR` = interquartile range, `cent`= centroid, `skewness` = skewness, `kurtosis` = kurtosis, `sfm` = spectral flatness measure, `sh` = spectral entropy, `prec.x` = frequency precision of the spectrum).
5. From 7034 to 7039: 6 additional statistical properties of the signal (e.g. `roughness`).
6. From 7040 to 7042: 3 annotations: `id`, `genre` and `tempo`, the target variable (not available in `test`)

⇝ Your job (A) ⇜

Pick any algorithm/model/black-box/predictor you like (LASSO/Elastic Net, SVM, Random Forest, Boosting, Convolutional nets, etc), to predict the tempo and comment the results. Please notice:

1. I expect you to upload a well commented working code that covers the entire pipeline: from data loading/pre-processing and feature engineering/dim-reduction to model fit and prediction on `test`.
2. You will also upload a `.csv` file containing the prediction on `test` that I will then score (RMSE).
3. ...uhmmmmmm...can you beat last year champions!?!? Just keep in mind they had only 48 hours!

B) Predicting with Confidence

Introduction

When doing estimation, we usually provide confidence intervals in addition to point estimates. Is there a similar notion for **predictions**? The answer is yes: we provide *prediction sets* or *set-valued predictions*. Given data $\mathsf{D}_n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$ we construct a set-valued function C_n, depending on D_n such that $\Pr\big(Y_{n+1} \in C_n(\boldsymbol{X}_{n+1})\big) \geqslant 1 - \alpha$.

The approach we consider here is **conformal prediction**. The idea is due to Vovk, Gammerman and Shafer (2005). The statistical theory was developed in Lei, Robins and Wasserman (2013), Lei and Wasserman (2014), Lei, G'Sell, Rinaldo, Tibshirani and Wasserman (2017), Sadinle, Lei and Wasserman (2018). More recent variations on this idea can be found in Angelopoulos et al. (2021), Bates et al. (2021) and Romano, Sesia and Candès (2020).

To give you an idea of what *conformal prediction* is, let's start from the **unsupervised case**: we observe $\mathsf{D}_n = \{Y_1, \ldots, Y_n\}$ and we want to predict Y_{n+1}. The basic algorithm is as follows:

Naive Algorithm

1. Define a permutation invariant *residual function* (or *conformity score*) $R = \phi(z, \mathsf{D}_{n+1})$ where $\mathsf{D}_{n+1} = \{Y_1, \ldots, Y_n, y\}$ is *any* **augmented dataset** of size $(n+1)$.
2. For each y we pick, do the following:
   - Set $Y_{n+1} = y$ and form the corresponding augmented dataset $\mathsf{D}_{n+1} = \{Y_1, \ldots, Y_n, Y_{n+1}\}$
   - Let $R_i = \phi(Y_i, \mathsf{D}_{n+1})$ for $i \in \{1, \ldots, n+1\}$
   - Test the hypothesis $H_0 : Y_{n+1} = y$ by computing the p-value $\pi(y) = \frac{1}{n+1} \sum_{i=1}^{n+1} \mathbb{1}(R_i \geqslant R_{n+1})$
3. Invert the test to get the confidence set $C_n = \{y : \pi(y) \geqslant \alpha\}$ for a pre-specified level $\alpha \in (0, 1)$.

Remarks

- It can be shown that $\Pr\big(Y_{n+1} \in C_n\big) \geqslant 1 - \alpha$ and this result is distribution-free and holds for all finite samples.
- The coverage validity does **not** depend on the choice of residual. But a poor choice can lead to large prediction sets.
- The algorithm above requires that we test $H_0 : Y_{n+1} = y$ for every y. In practice, we only consider a **grid of values** for y. But this can be slow. The **split conformal** method below is much faster but can result in larger prediction sets (and it depends on the particular split). The steps are:

  1. Randomly split D_n in two equal-sized subsets $\mathsf{D}^{(1)}$ and $\mathsf{D}^{(2)}$
  2. Compute the residuals on $D^{(1)}$: $R_i = \phi(Y_i, D^{(1)})$ for $Y_i \in D^{(1)}$
  3. Let q be the $1 - \alpha$ quantile of the residuals $\{R_i\}_i$
  4. Return $C_n = \big\{y : \phi(y, D^{(1)}) \leqslant q\big\}$

Regression

The extension to regression is straightforward. All the relevant information and algorithms can be found in Section 2.1 and 2.2 of Lei et al. (2017) and I strongly suggest you to read them. The following is their **Algorithm 2** at page 10:

Split Conformal Prediction for Regression

- *Input*: Data $\mathsf{D}_n = \{(\boldsymbol{X}_1, Y_1), \ldots, (\boldsymbol{X}_n, Y_n)\}$, miscoverage level $\alpha \in (0, 1)$, a generic regression algorithm $\mathcal{A}$
- *Output*: Prediction band over $\boldsymbol{x} \in \mathbb{R}^p$

1. Randomly split D_n in two equal-sized subsets $\mathsf{D}^{(1)}$ and $\mathsf{D}^{(2)}$
2. Train on $\mathsf{D}^{(1)}$ to get: $\widehat{f} = \mathcal{A}\big(\mathsf{D}^{(1)}\big)$
3. Predict and evaluate the residuals on $\mathsf{D}^{(2)}$: $R_i = \big|Y_i - \widehat{f}(\boldsymbol{X}_i)\big|$ for all $(\boldsymbol{X}_i, Y_i) \in \mathsf{D}^{(2)}$
4. Let $d =$ the $k^{\texttt{th}}$ smallest value of $\{R_i\}_i$ where $k = \lceil (n/2 + 1)(1 - \alpha) \rceil$
5. For any $\boldsymbol{x} \in \mathbb{R}^p$ you choose, return $C_{\texttt{split}}(\boldsymbol{x}) = \big[\widehat{f}(\boldsymbol{x}) - d, \widehat{f}(\boldsymbol{x}) + d\big]$

Also in this case it can be shown that, if $(\boldsymbol{X}_{n+1}, Y_{n+1})$ is a new datapoint, then $\Pr\big(Y_{n+1} \in C_{\texttt{split}}(\boldsymbol{X}_{n+1})\big) \geqslant 1 - \alpha$

⇝ Your job (B) ⇜

1. Starting from the model used in Part A), implement the *Split Conformal Prediction for Regression* algorithm.
2. Randomly pick $m = 100$ observations from the `test` set and build their predictive sets.
3. Provide a suitable visualization of the results and comment.

C) Variable Importance (Bonus if you're discussing in June)

Introduction

Being able to quantify the importance of a covariate in predicting a response of interest is crucial in most real applications... even more so nowadays, when the use of very complex, nonlinear, overparametrized models is the rule rather than the exception.

Despite this, the very idea of "importance" is slippery and need to be precisely framed, defined and handled (... yes, even when talking about linear models!). Here we'll focus on a very simple and general technique.

The LOCO

At page 32 of their paper, Lei and coauthors proposed a simple, general and, essentially assumptions–free idea for measuring variable importance, called **leave-one-covariate-out** (LOCO) inference. The algorithm is extremely simple

1. Randomly split the sample into two, non overlapping, parts: $\mathsf{D}_n = \mathsf{D}_{n_1}^{(1)} \cup \mathsf{D}_{n_2}^{(2)}$ with $n_1 + n_2 = n$.
2. Run any algorithm you like to compute an estimate $\widehat{f}_{n_1}(\cdot)$ on first part $\mathsf{D}_{n_1}^{(1)}$.
3. Select some variable $\boldsymbol{X}[j]$ of interest to you, and recompute $\widehat{f}_{n_1}^{-j}(\cdot)$ on $\mathsf{D}_{n_1}^{(1)}$ again (rerun algorithm without access to variable $\boldsymbol{X}[j]$).
4. Use $\mathsf{D}_{n_2}^{(2)}$ to construct finite-sample, distribution-free confidence interval (e.g., use non-parametric bootstrap or sign-test or Wilcoxon-test) for the following new (population) parameter:

$$\theta_j\big(\mathsf{D}_{n_1}^{(1)}\big) = \text{median}_{(Y,\boldsymbol{X})}\Big(\big|Y - \widehat{f}_{n_1}^{-j}(\boldsymbol{X})\big| - \big|Y - \widehat{f}_{n_1}(\boldsymbol{X})\big| \,\Big|\, \mathsf{D}_{n_1}^{(1)}\Big), \quad j \in \{1, \dots, p\}. \tag{1}$$

   Since you're using the same dataset to build more than one confidence-interval, apply any reasonable correction to adjust for multiplicity (e.g. Bonferroni or Benjamini-Hochberg FDR).

θ_j has a very clear interpretation without resorting to linearity or any other *uncheckable* model assumption: it's just how much extra prediction error you would pay by not having access to variable $\boldsymbol{X}[j]$.

In addition, from a more technical point of view, this parameter is "smooth" enough (Hadamard differentiable) to guarantee the success of resampling techniques like the nonparametric bootstrap. In other words, confidence intervals, tests, etc for θ_j are easy to obtain no matter what is the underlying predictive model you picked (LASSO, LASSO + CV, Random Forest, Boosting, Deep nets, etc).

Of course there are also problems with the LOCO. More specifically:

1. It is **not** on an intuitive scale but we could fix this by rescaling.
2. Results are tied to our choice of the algorithm. In theory we could use a "meta–cross–validation" scheme that cycles over different candidate predictor classes (computational expensive but trivially parallelizable).
3. Results are also sensitive to the ratio of training to test set sizes.
4. It is conditional on $\mathsf{D}_{n_1}^{(1)}$, meaning that it measures "*how important is variable X_j, to our algorithm's estimates on $\mathsf{D}_{n_1}^{(1)}$?*"...not obvious at all how to fix this...

For more information, comments and examples, please take a look at the following set of slides.

⇝ Your job (C) ⇜

1. Starting from the model used in Part A), evaluate variable importance via LOCO for a subset of 5 features. More in particular:
   - If the technique you picked comes with a *default* method to evaluate variable importance (e.g. Random Forest), compare the top-5 in this ranking with their LOCO scores.
   - On the other hand, if the technique you picked does not come with a default method to evaluate variable importance, pick 5 features completely at random or based on any other reasoning you may come up with.
2. Provide a suitable visualization of the results and comment.