Dynamic Programming for Beat Tracking

MUMT 621 Presentation 3. February 23, 2021. Sevag Hanssian, 260398537

Summary

In 1950, Richard Bellman coined the term “dynamic programming” to refer to a type of algorithm he designed for solving multistage decision processes (Dreyfus 2002). Programming here refers to scheduling or planning, not computer programming. In the creation story, the adjective “dynamic” was chosen because it both describes the time-varying and multistage nature of the algorithm, and because it generally isn’t used in a negative or prejorative sense. The goal was to give the algorithm a vague and positive name, so that the anti-research Secretary of Defense in charge of funding could find no reason to tank the project.

Skiena (2008) and Cormen et al. (2001) describe dynamic programming as follows. It is an algorithm to solve computer science optimization problems by finding the best solution that maximizes an objective cost function. It involves breaking the problem down into subproblems and storing the results of each subproblem for efficiency. Dynamic programming leads to a globally optimal solution by searching the entire solution space, and the cost of repeatedly recomputing subproblems is traded off for the storage cost of computing them once and storing the solution. A similar type of algorithm is the divide-and-conquer recursive algorithm, which decomposes a problem into subproblems (divide), calls itself to solve the subproblems (conquer), and combines the solutions. Adding storage of the recursively-computed subproblems brings one most of the way towards a dynamic programming solution.

Ellis (2007) formulated the task of beat tracking in music information retrieval as a two-way optimization problem. The algorithm starts with computing onset strengths, $O(t)$, for every time t (discrete samples) in the input signal, also known as the onset strength envelope. Onsets refer to the start of musical notes or events (Bello et al. 2005). Autocorrelation (a measure of signal self-similarity) is then applied on the onset strength envelope to find the dominant onset periodicity. Some further perceptual post-processing is done to take into account the natural human bias towards 120bpm. Finally, the most dominant periodicity is chosen as the global tempo estimate for the entire song. An assumption (and limitation) of the algorithm is that there is a single stable tempo throughout the song.

Once the onset strength envelope and global tempo are known, the beat tracking problem becomes this:

$$C(\{t_i\}) = \sum_{i=1}^{N} O(t_i) + \alpha \sum_{i=2}^{N} F(t_i - t_{i-1}, \tau_p)$$

The final beat times returned by the beat tracker are a set $\{t_i\}$ that maximize the above cost function. O, the first term, is the onset strength envelope which weighs strong onsets as more likely to be beats. The second term $F(\Delta t, \tau) = -\left(\log \frac{\Delta t}{\tau} \right)^2$ is the penalty term to ensure that the selected beat has an inter-beat interval ($\Delta t = t_i - t_{i-1}$) that is consistent both with the previously selected beat and that fits the global tempo τ_p. Finally, α controls how much each term is weighted in the objective score.

The problem as stated involves an exponential search across all possible combinations of discrete time points $\{t_i\} \in [0, N]$ in an audio signal of length N samples (Ellis 2013), which is computationally very expensive. Dynamic programming allows the same problem to be restated in a recursive manner:

$$C^*(t) = O(t) + \max_{\tau=0\ldots t}\{\alpha F(t - \tau, \tau_p) + C^*(\tau)\}$$

$$P^*(t) = \mathrm{argmax}_{\tau=0\ldots t}\{\alpha F(t - \tau, \tau_p) + C^*(\tau)\}$$

The best score C^* for time t is the local onset strength, plus the best score to the preceding beat time τ, and P^* stores the corresponding previous beat time. This optimization is possible because future beats $\in [t, N]$ are not considered to influence the correctness of past or present beats $\in [0, t]$. To get beats for an input signal, first C^* and P^* are computed for every time point $t_i \in [0, N]$. Then, the maximum C^* is chosen from within τ_p of the end of the song, and the chain of the best preceding beat is followed through P^* to generate the final optimal sequence of beats $\{t_i\}$.

The algorithm is computationally efficient and scored reasonably well in MIREX 2006.[1] It is the implementation of the beat tracking algorithm available in librosa (McFee et al. 2015), a popular modern Python library for music information retrieval.

1. https://www.music-ir.org/mirex/wiki/2006:Audio_Beat_Tracking

Annotated bibliography

Bello, Juan, Laurent Daudet, Samer Abdallah, Chris Duxbury, Mike Davies, and Mark Sandler. 2005. "A Tutorial on Onset Detection in Music Signals." *Speech and Audio Processing, IEEE Transactions on* 13 (October): 1035–1047. https://doi.org/10.1109/TSA.2005.851998. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.332.989&rep=rep1&type=pdf.

A paper summarizing different definitions of onsets and the task of onset detection and onset strength.

Cormen, Thomas H., Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms.* 2nd. 28, 323, 339. The MIT Press. ISBN: 0262032937.

A legendary computer science textbook, from where I took definitions for dynamic programming and recursion.

Dreyfus, S.E. 2002. "Richard Bellman on the Birth of Dynamic Programming." *Operations Research* 50 (February): 48–51. https://doi.org/10.1287/opre.50.1.48.17791. http://www.cas.mcmaster.ca/~se3c03/journal_papers/dy_birth.pdf.

A biographical history of Richard Bellman's creation of dynamic programming.

Ellis, Daniel. 2007. "Beat Tracking by Dynamic Programming." *Journal of New Music Research* 36 (March): 51–60. https://doi.org/10.1080/09298210701653344. http://www.music.mcgill.ca/~ich/classes/mumt621_09/presentations/wingate/27406228.pdf.

Daniel Ellis' paper presenting his algorithm for beat tracking with dynamic programming. Diagrams and formulas were all taken from this paper.

———. 2013. "Lecture 10: Beat Tracking." *ELEN E4896 Music Signal Processing* (April). https://www.ee.columbia.edu/~dpwe/e4896/lectures/E4896-L10.pdf.

Lecture slides from Daniel Ellis for a course he teaches where he describes and expands on onset detection and the dynamic programming beat tracking algorithm.

McFee, Brian, Colin Raffel, Dawen Liang, Daniel Ellis, Matt Mcvicar, Eric Battenberg, and Oriol Nieto. 2015. "librosa: Audio and Music Signal Analysis in Python," 18–24. January. https://doi.org/10.25080/Majora-7b98e3ed-003.

The official citation for librosa, a Python library for MIR.

Skiena, Steven S. 2008. *The Algorithm Design Manual.* 31–40, 273–278. London: Springer. ISBN: 9781848000704 1848000707 9781848000698 1848000693. https://doi.org/10.1007/978-1-84800-070-4.

Another legendary computer science textbook, from where I took definitions for dynamic programming, and the hypothetical computer for analyzing time complexity, Big-Oh analysis, and the Fibonacci example in the presentation.