CS5112: Algorithms and Data Structures for Applications

Lecture 14: Exponential decay; convolution

Ramin Zabih

Some content from: Piotr Indyk; Wikipedia/Google image search;
J. Leskovec, A. Rajaraman, J. Ullman: Mining of Massive Datasets, http://www.mmds.org

Cornell University

CORNELL TECH

Administrivia

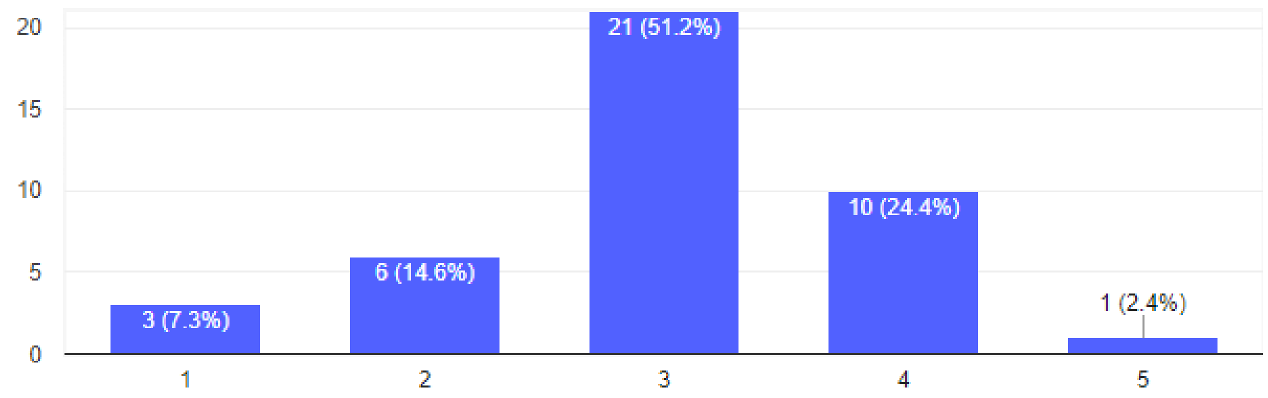
- Q7 delayed due to Columbus day holiday
- HW3 out but short, can do HW in groups of 3 from now on
- Class 10/23 will be prelim review for 10/25
  - Greg lectures on dynamic programming next week
- Anonymous survey out, please respond!
- Automatic grading apology

Survey feedback

Lecture

1. How do you find the pace of lectures? (3 = just right)

41 responses



CORNELL TECH

Automatic grading apology

- In general students should not have a grade revised downward
  - Main exception is regrade requests
- Automatic grading means that this sometimes happened
- At the end, we decided that the priority was to assign HW grades based on how correct the code was
- Going forward, please treat HW grades as tentative grades
  - We will announce when those grades are finalized
  - After, they will only be changed under exceptional circumstances

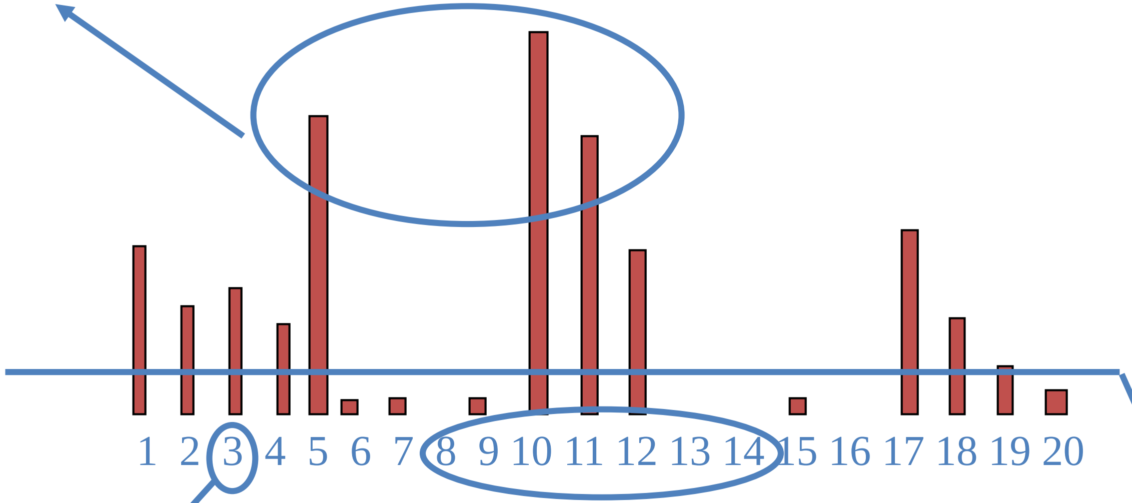
CORNELL TECH

Today

- Two streaming algorithms
- Convolution

Some natural histogram queries

Top-k most frequent elements



Find all elements with frequency > 0.1%

What is the frequency of element 3?

What is the total frequency of elements between 8 and 14?

How many elements have non-zero frequency?

CORNELL TECH

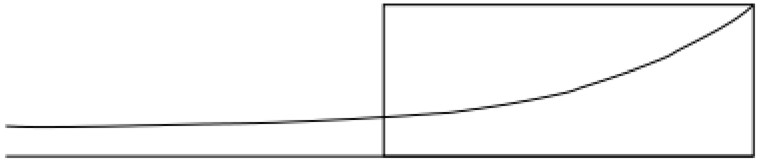
Streaming algorithms (recap)

1. Boyer-Moore majority algorithm
2. Misra-Gries frequent items
3. Find popular recent items (box filter)
4. Find popular recent items (exponential window)
5. Flajolet-Martin number of items

Weighted average in a sliding window

- Computing the average of the last k inputs can be viewed as a dot product with a constant vector $v = \left[\frac{1}{k}, \frac{1}{k}, \ldots, \frac{1}{k}\right]$
- Sometimes called a box filter
  - Easy to visualize
- This is also a natural way to smooth, e.g., a histogram
  - To average together adjacent bins, $v = [\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$
- This kind of weighted average has a famous name: convolution

Decaying windows



- Let our input at time t be $\{a_1, a_2, \ldots, a_t\}$
- With a box filter over all of these elements we computed

$$\frac{1}{t}\sum_{i=0}^{t-1} a_{t-i}$$

- Instead let us pick a small constant c and compute

$$\hat{a}_t = \sum_{i=0}^{t-1} a_{t-i}(1-c)^i$$

Easy to update this

- Update rule is simple, let the current dot product be $\hat{a}_t$
$$\hat{a}_{t+1} = (1 - c)\hat{a}_t + a_{t+1}$$
- This downscales the previous elements correctly, and the new element is scaled by $(1 - c)^0 = 1$
- This avoids falling off the edge
- Gives us an easy way to find popular items

4. Popular items with decaying windows

- We keep a small number of weighted sum counters
- When a new item arrives for which we already have a counter, we update it using decaying windows, and update all counters
- How do we avoid getting an unbounded number of counters?
- We set a threshold, say ½, and if any counter goes below that value we throw it away
- The number of counters is bounded by $\frac{2}{c}$

CORNELL TECH

How many distinct items are there?

- This tells you the size of the histogram, among other things
- To solve this problem exactly requires space that is linear in the size of the input stream
  - Impractical for many applications
- Instead we will compute an efficient estimate via hashing

5. Flajolet-Martin algorithm

- Basic idea: the more different elements we see, the more different hash values we will see
  - We will pick a hash function that spreads out the input elements
  - Typically uses universal hashing

Flajolet-Martin algorithm

- Pick a hash function h that maps each of the n elements to at least $\log_2 n$ bits
- For input a, let $r(a)$ be the number of trailing 0s in $h(a)$
  - $r(a)$ = position of first 1 counting from the right
  - E.g., say $h(a) = 12$, then 12 is 1100 in binary, so $r(a) = 2$
- Record R = the maximum $r(a)$ seen
- Estimated number of distinct elements = 2^R
  - Anyone see the problem here?

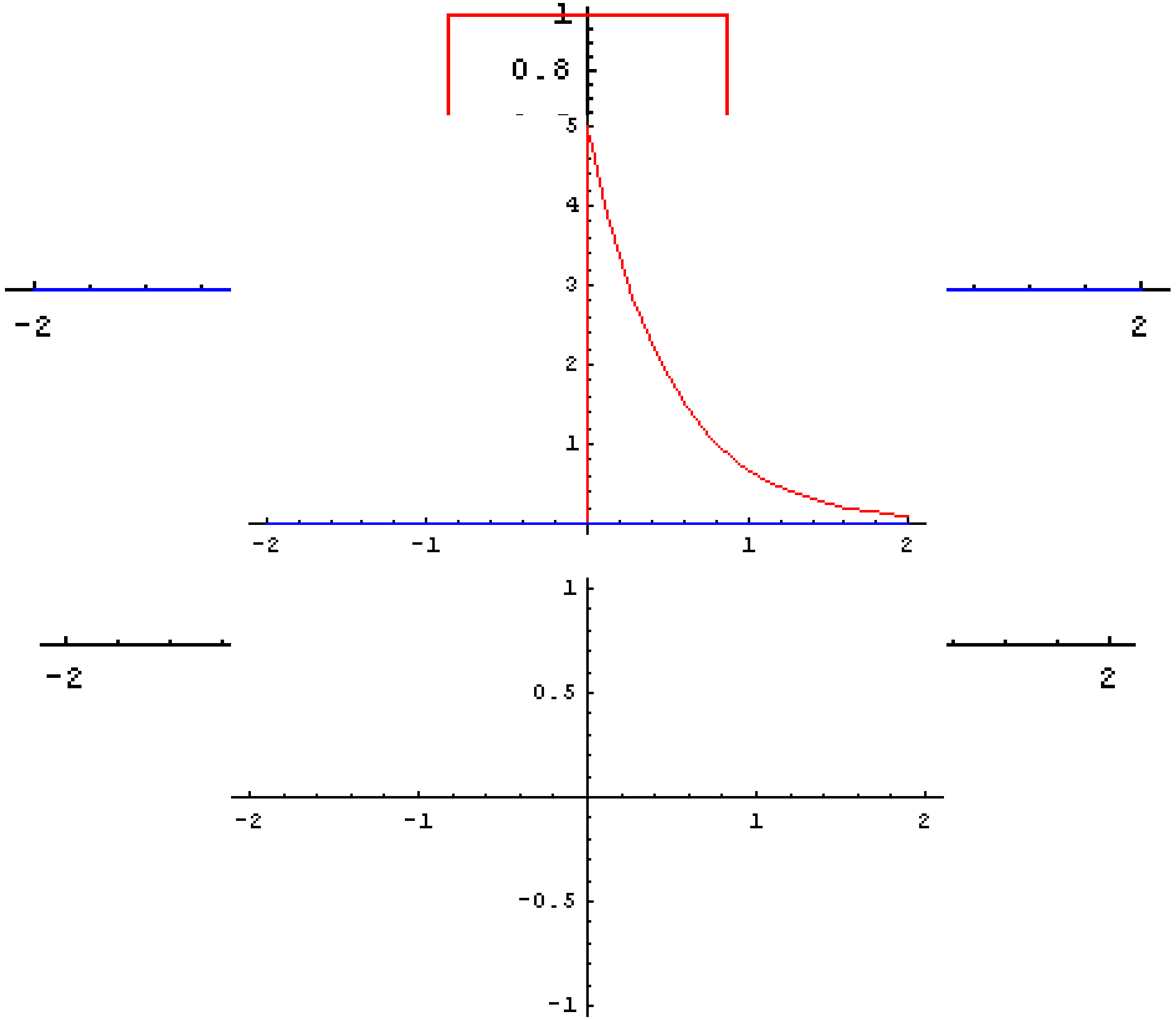
Why It Works: Intuition

- Very rough intuition why Flajolet-Martin works:
  - $h(a)$ hashes a with equal probability to any of n values
  - Sequence of $(\log_2 n)$ bits; 2^{-r} fraction of $a's$ have tail of r zeros
    - About 50% hash to ***0
    - About 25% hash to **00
    - So, if we saw the longest tail of r=2 (i.e., item hash ending *100) then we have probably seen about 4 distinct items so far
  - Hash about 2^r items before we see one with zero-suffix of length r

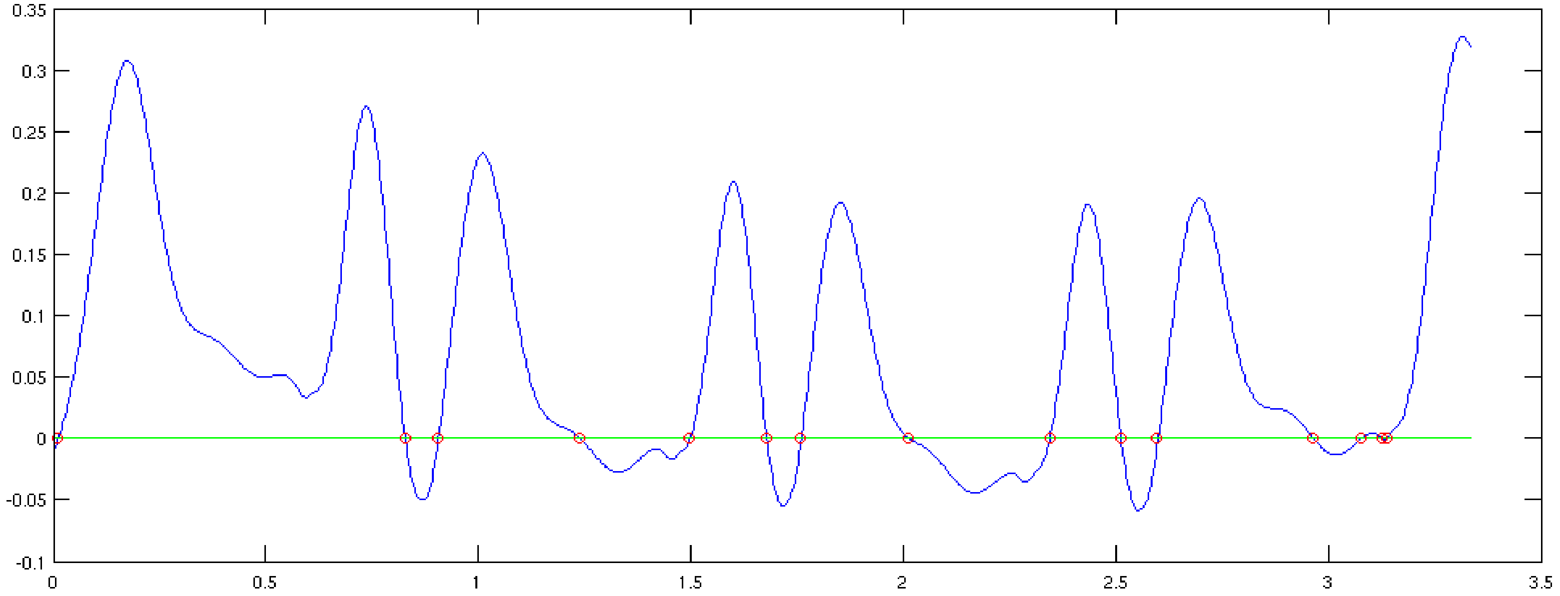
Convolution

- Weighted average with a mask/stencil/template
  - Dot product of vectors
- Many important properties and applications
- Symmetric in the inputs
- Equivalent to linear shift-invariant systems
  - “Well behaved”, in a certain precise sense
- Primary uses are smoothing and matching
- This is the “C” in “CNN”

Local averaging in action



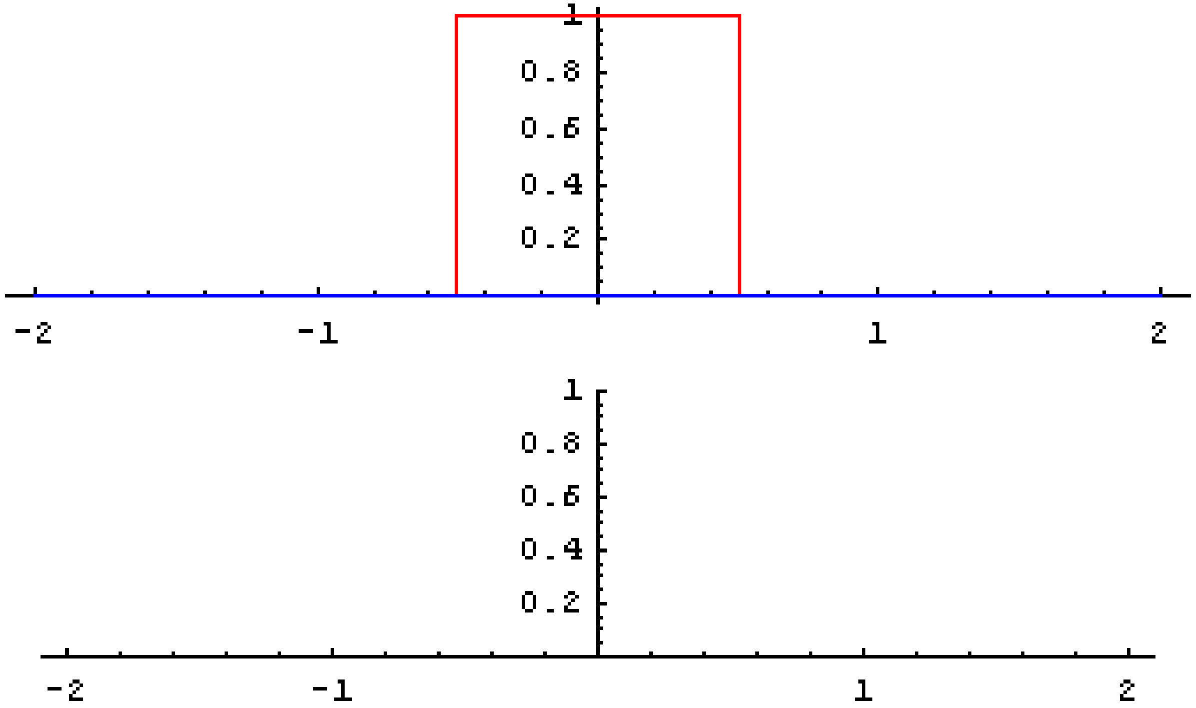
Smoothing parameter effects



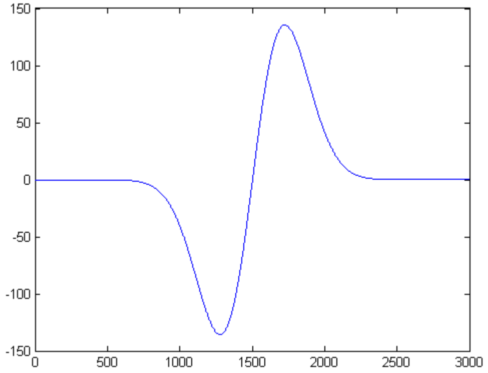
Matched filters

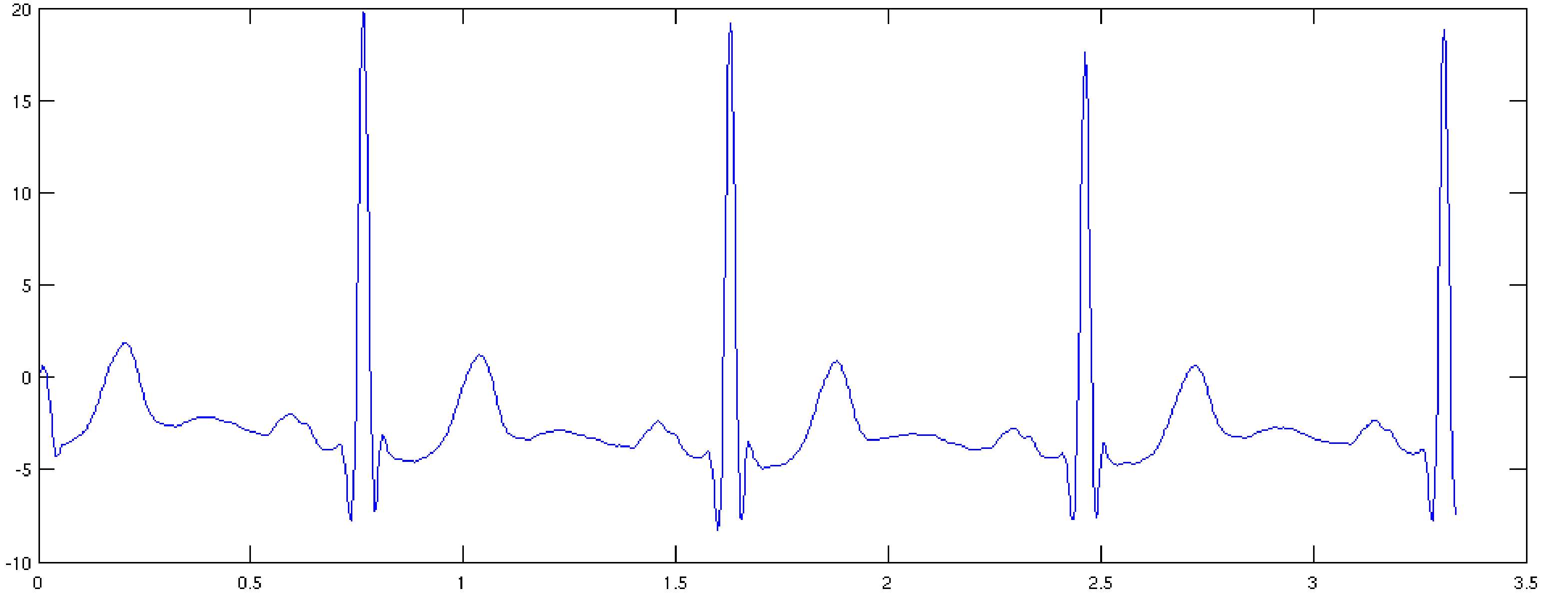
- Convolution can be used to find pulses
  - This is actually closely related to smoothing
- How do we find a known pulse in a signal? Convolve the signal with our template!
  - E.g. to find something in the signal that looks like [1 6 -10] we convolve with [1 6 -10]
- Question: what sense does this make?
  - Anecdotally it worked for finding boxes

Box finding example



Pulse finding example



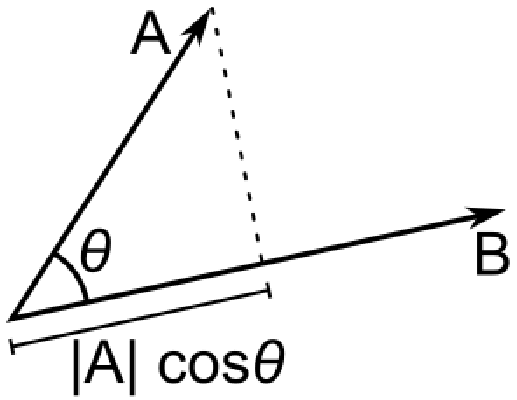


Why does this work?

- Some nice optimality properties, but the way I described it, the algorithm fails
- Idea: the [1 6 -10] template gives biggest response when signal is [… 1 6 -10 …]
  - Value is 137 at this point
- But is this actually correct?
  - You actually need both the template and the input to have a zero mean and unit energy (sum of squares)
    - Easily accomplished: subtract -1, then divide by 137, get 1/137 * [2 7 -9]

Geometric intuition

- Taking the dot product of two vectors
  - Recall $[a\ b\ c] \cdot [e\ f\ g] = ae + bf + cg$
  - Basically the projection of a vector on another



- The normalized vector with the biggest projection on x is, of course: x!