So what does this have to do with speech recognition??

Remember we found that the duration of a word can be quite variable.

How can we evaluate the similarity of a test word and a template when they are of different lengths?

How do we align them to maximize their similarity, or minimize their dissimilarity (or distance or cost)?

Distance can be calculated using the Inner Product (1/IP) or other similar techniques.
Evaluating similarity when words differ temporally

- Spectrograms:
  - “You can **shoot** at the ship or do nothing.”
  - vs.
  - “**shoot**” template from a word list
Evaluating similarity when words differ temporally

- Divide both into 50 ms frames.
- Would be shorter in a real recognizer, and will be shorter later today.
Evaluating similarity when words differ temporally

▶ Add labels indicating which segment is going on in each frame.
▶ Recognizer won’t have those labels, just spectra at those frames
▶ so... “sh” stands for a sh-like spectrum
▶ “uw” stands for an uw-like spectrum
Evaluating similarity when words differ temporally

- Matrix or grid of Template frames X test frames
- For every row (frames of test) we need to choose which column frame or frames (template) to best match it against to calculate distance.
- Overall distance of test to template is the sum of all the distances in those frames.
- Which frames do we choose?
Each successive test frame is matched to the next template frame.

Path has the fewest frames (diagonal), but is non-optimal.

It compares a frame of UW in the test with a frame of SH in the template.

It compares frames of T in the test with frames of UW in the template.

The overall distance (cost) will be high.
An even worse path (blue)

Path crosses even more frames, but still matches incompatible frames of test and template.
The global distance between **template** and **test** is minimized.

**How do we find it?**
Finding the path

- Turn matrix on its side and we can see that this is a network of paths
- We start at the first frame of both and end at the last frame of both,
- What path do we take?
From each square (node), there are three possible paths.

- Test progresses in time (A).
- Template progresses in time (B).
- Both test and template progress in time (C).

Cost of taking any path is the distance between the test and template there.
Every choice leads to 3 more possible paths.

Each cell is like a stage in our earlier networks.

So the number of stages is the number of rows ($R$) times the number of columns ($C$).

So there are many, many possible paths (roughly on the order of $3^{R\times C}$ or $3^{11\times11}$ in this case. A huge number.

How do we find the optimal (least cost) path, the one that minimizes distance of test to template?
For any cell (for example, E), what is the minimum cost of getting there ($V_{AE}$)?

- We don’t know what it is.
- But there are only 3 possible nodes that lead to E (B,C,D).
- If we knew the minimum cost of getting to each of those from A (we don’t yet know), then we would know the minimum cost of getting to E.
  
  - $V_{AE} = distance(E) + \min(V_{AB}, V_{AC}, V_{AD})$

- But we can apply the same logic to B, C, and D ...
The application of dynamic programming to this problem is called **Dynamic Time Warping**

Let's illustrate with a simpler problem.

Align two vectors \((a, b)\) so as to minimize their total distance.

Let distance is equal to the absolute value of the difference of the paired \(a\) and \(b\) values.

The optimal alignment is clearly to align the 5’s with the 6’s and the 10’s with the 11’s.

How do we find that?
First calculate the distance between every individual element of \( a \) with every element of \( b \).

Colors are “heat map.” Hottest color (white) means small distance (very similar). Coldest color (black) means large distance (very dissimilar).
Now begin to calculate the minimum distance path from the start point (lower left) to each cell in the matrix.

For cells in the first column, this is an easy problem.

Since we cannot go backwards in time, there is only one path, so the minimum distance from the start point to each cell is the sum of the distances in the cells along the (single) path to that cell.
The same is true along the first row.
Now we can calculate the minimum path to cell D.

It is equal to the distance at that cell + the minimum of the path lengths from start to A,B,C (the only cells from which we can get to D.)
Value at ? = 1 + \min(2,2,1)
Now we can move up the column.

Value at $\ ? = 4 + \min(6,2,2)$
- We can continue the process, until we arrive at the end node, at the upper right.
- The length of the minimum path is 8.
- For some purposes we are done. We know the length of the minimum path.
- But if we want to know which path (or paths) have that minimum length, we have to go backwards, choosing at each cell, which of the three possible choices has the minimum path length from the start.
From the end, we choose which of the adjacent nodes has the minimum path length (8,7,7).

There is a tie, so we can choose either of the paths whose distance is 7.

There is not one single optimal path.
Continue until we get back to the starting node.
If we draw the path on top of our original matrix with the local distance between pairs of values, it goes through the cells that have the small distances.

5’s are paired with 6’s and 10’s are paired with 11’s as we desired.
We can apply exactly the same algorithm to comparing two spectrograms.

The spectrum is taken every 20 milliseconds, so in this example there are 30 frames in both the test and the template.

So our matrix will have 900 cells.

The only difference is that the distance measure for each cell is not a simple difference.

It must compare the dissimilarity of two vectors, which represent the spectrum of the test and template.

Distance is calculated in a way similar to the inner product (IP).
Results: “shoot” from sentence with shoot template

Distance = 46
Results: “shoot” from sentence with shoot template

Distance = 46
We can also compare the word “shoot” cut out of a sentence with a “sip” template. We expect that the total minimum distance from begin to end will be longer than when we compare it with the “shoot” template.
Results: “shoot” from sentence with sip template

Distance = 62.4