ScattLab Architecture Document

April 16, 2013

1 Filters

1.1 Filter Structure

Filters are defined by a signal size `[N,M]`, a filter type (Morlet, Gabor, spline), and wavelet-specific parameters. For one-dimensional signals, $M = 1$.

Filter parameters are specified in a parameters structure, `fparam`, containing the following fields:

- `fparam.filter_type`: The wavelet type, such as 'morlet_1d', 'gabor_2d', 'spline_1d', for example.
- `fparam.precision`: The numeric precision of the filters. Either `'double'` or `'single'`.

In addition, the `fparam` structure would contain parameters specific to the wavelet type chosen (see below).

Once filter parameters are entered, the `filter_bank` function is called to generate the filter bank:

```
filters = filter_bank([N M], fparam);
```

This function will then call the appropriate filter bank function (`morlet_1d_filter_bank`, `gabor_2d_filter_bank`, etc.) depending on the value of `fparam.filter_type` and put it in a cell array.

If one of the parameters is an array (except for `fparam.filter_type` and `fparam.precision`, which have to be cell arrays), `filter_bank` will create multiple filter banks and output them. For example, if `fparam.filter_type` equals {`'gabor_1d'`,`'spline_1d'`}, `filter_bank` will output a cell array of two filter banks, one with Gabor wavelets and one with spline wavelets. If parameter fields are of different sizes, the shorter ones are extended by concatenating the last value the required number of times.

The returned structure, `filters`, contains the filters ψ and ϕ that form the filter bank, as well as meta information. Specifically, the fields are:

- `filters.psi`: A set of wavelet filters ψ_λ (for definition, see below)
- `filters.phi`: A set of lowpass filter(s) ϕ (for definition, see below)
- The parameters given in `fparam` and the signal size `[N,M]`. For example, `filter.filter_type` gives the type of filters in `filters.psi` and `filters.phi`.

Each filter set (be it `filters.phi` or `filters.psi`), is a structure `fset` containing the following:

- `fset.filter`: A cell array of the actual filter coefficients. These coefficients are implementation-dependent and can encode the filter spatially, in the Fourier domain, at different resolutions, etc.
- `fset.meta`: Contains meta information on the filters. Specifically, it has two fields: `fset.meta.k`, which is the scale indices, and `fset.meta.theta`, which is the angle indices (for two-dimensional filters). Both `fset.meta.k` and `fset.meta.theta` are of the same length as `fset.filter`.

The scale and angle indices are non-negative integers. The scale index rises with increasing scale, while the angle index rises with increasing angle (counterclockwise).

1.2 Morlet/Gabor filter bank

In addition to the parameters listed above, the Morlet/Gabor filter bank has the following options:

- `fparam.V`: The number of wavelets per octave.
- `fparam.J`: The number of wavelet scales.
- `fparam.sigma_psi`: The standard deviation of the mother wavelet in space.
- `fparam.sigma_phi`: The standard deviation of the scaling function in space.
- `fparam.slant`: The slant of the mother wavelet ellipse in frequency.
- `fparam.nb_angle`: The number of wavelet angles.

The maximal wavelet bandwidth (in space) is determined by $2^{J/V}$ times the bandwidth of the mother wavelet, which is proportional to `sigma_psi`. If `sigma_psi` is smaller than a certain threshold, a number of constant-bandwidth filters are added, linearly spaced, to cover the low frequencies.

Again, we can specify different filter banks by setting `fparam.V` and `fparam.J`, etc. to arrays instead of scalars. This is often useful if the nature of the signal is different at different orders, which is usually the case in audio.

1.3 Spline filter bank

In addition to the parameters listed above, the spline filter bank has the following options:

- `fparam.J`: The number of wavelet scales.
- `fparam.spline_order`: The order of the splines. Only linear (spline order 1) and cubic (spline order 3) are supported.

The maximal bandwidth is specified here by 2^J.

2 Wavelet Modulus and Scattering Transforms

2.1 Wavelet Modulus Transform

The wavelet modulus transform takes a layer of coefficients and calculates the next. This layer has the fields:

- `layer.signal`: A cell array of signals.
- `layer.meta`: The meta information on the signals, such as their path, their resolutions, etc.

The signals are one-dimensional or two-dimensional arrays while `meta` contains the fields `meta.k` and `meta.theta`, which are two-dimensional arrays. The first dimension has length corresponding `layer.signal` while the second dimension as length corresponding to the order of the coefficients. The path of the lth coefficient is thus encoded in `meta.k(l,:)` and `meta.theta(l,:)`, respectively.

A wavelet modulus transform (of which there are multiple, such as `wavemod_1d`, `wavemod_2d`, etc.) is a function that takes a layer `U{m}`, a filter bank (see previous section), an options structure, and returns the smoothed output of this layer `S{m}` as well as the next layer `U{m+1}`. Specifically, for a wavelet modulus transform `wavemod`, we have:

```
[S{m}, U{m+1}] = wavemod(U{m}, ...
    filters, options);
```

Here, `U` and `S` are cell arrays of layers, as described above. This wavelet modulus can be a one-dimensional wavelet modulus transform (`wavemod_1d`), a two-dimensional wavelet modulus transform (`wavemod_2d`), a joint wavelet modulus transform, etc. It only has to satisfy the above input/output conditions.

2.2 Scattering Transform

By stacking multiple wavelet modulus transforms together, we obtain the scattering transform. Specifically, the `scatt` function, takes a signal, an options structure, a cell array of wavelet modulus transforms

(with filters fixed), and returns the scattering coefficients (or wavelet modulus coefficients, if desired). The scattering coefficients are output as a cell array of layers `S`.

The scattering transform could be used like the following

```
fparam.filter_type = {'gabor_1d', ...
    'morlet_1d'};
fparam.V = [8 1];
fparam.J = [80 13];
fparam.sigma_psi = [8 1];
fparam.sigma_phi = [8 0.5];

filters = filter_bank(N,fparam);

wavemod{1} = @(x)(wavemod_1d(x, ...
    filters{1}, options));
wavemod{2} = @(x)(wavemod_1d(x, ...
    filters{2}, options));
wavemod{3} = @(x)(wavemod_1d(x, ...
    filters{2}, options));

S = scatt(x,wavemod);
```

The above code will compute a filter bank for signals of length N, with the first filter bank consisting of 8 filters per octave, for $80/8 = 10$ octaves, with a wavelet bandwidth corresponding to about 1/8th of an octave. The second filter bank will only have one wavelet per octave, with the corresponding bandwidth, and 13 octaves of wavelets. Since the mother wavelet for the first filter bank has bandwidth 8 while that of the second has bandwidth 1, their maximal bandwidths $8 \cdot 2^{80/10} = 8192$ and $2^{13} = 8192$ will coincide.

Note that for the filter bank for which $V = 1$, the lowpass filter is half as wide in space compared to the largest wavelet. This is necessary to properly tile the frequency domain when $V = 1$. For larger Vs, we can set `sigma_phi` equal to `sigma_psi`.

These filters are then fed into wavelet modulus transform functions which are concatenated into a cell array. They are called with the options structure `opt` fixed, which contains various options for the wavelet modulus transform. Finally, the scattering transform is called, yielding the result `S` from the signal `x`.

Another this to note is that the number of wavelet modulus is 3, which means that `scatt` will produce zeroth-, first- and second-order coefficients. The last `wavemod` function will only serve to smooth the second-order wavelet modulus coefficients to give the second-order scattering coefficients. Thus to obtain scattering coefficients of maximal order M, you have to supply $M + 1$ `wavemod` functions.

To obtain the wavelet modulus coefficients `U`, `scatt` is called for two outputs, as in

```
[S, U] = scatt(x,wavemod);
```

The structure of `U` follows that of `S`. That is it consists of a cell array representing each layer of wavelet modulus coefficients. The first layer corresponds to the zeroth-order wavelet modulus coefficients, which is the original signal. The second layer contains the first-order wavelet modulus coefficients, which are the signal convolved with the filters with the modulus applied, and so on. The coefficients in `S{m+1}` are thus the mth-order scattering coefficients obtained from smoothing the coefficients in `U{m+1}`.

2.3 `wavemod` Factory

To simplify the construction of wavelet modulus transforms, several wavelet modulus factory functions are provided. The one corresponding to `wavemod_1d` is `wavemod_1d_factory`. These take as input the filter parameters along with options passed on to the wavelet modulus transforms. In addition, the desired order `M` of the scattering transform is specified.

The code from the previous section then becomes:

```
fparam.filter_type = {'gabor_1d', ...
    'morlet_1d'};
fparam.V = [8 1];
fparam.J = [80 13];
fparam.sigma_psi = [8 1];
fparam.sigma_phi = [8 0.5];

wavemod = wavemod_1d_factory(fparam, ...
    options, 2);

S = scatt(x,wavemod);
```

This is the recommended way of defining a scattering transform. The previous method of creating a cell array of `wavemod` functions may not always be supported.

3 Manipulating, Displaying, Formatting

3.1 Renormalization and Logarithm

Often, second- and higher-order coefficients will reproduce information from their parent coefficients. That is, they will be highly correlated. To reduce this, the `renorm_scatt` function renormalizes each coefficient by dividing it by its parent coefficient. Similarly, the `log_scatt` function calculates the logarithm of each coefficient.

These functions both act on the output of `scatt`, and so can be called on the scattering transform `S` like:

```
Sr = renorm_scatt(S);
Srl = log_scatt(S);
```

3.2 Display

Two functions are available to display scattering coefficients, `display_slice` and `display_multifractal`. They both take as input a scattering transform `S` and a time point `t`.

3.3 Formatting

In order to use the scattering coefficients for classification, they need to be in a vector format. This is obtained using the function `format_scatt`, which arranges scattering coefficients into a two-dimensional table, the first dimension corresponding to a scattering coefficient index and the second dimension corresponding to time/space. It also returns a meta structure that specifies the order, scale, etc. of each scattering coefficient. The following example illustrates its usage:

```
[t,meta] = format_scatt(S);

% plot the 2nd-order coefficients
% corresponding to scale (3, 7)
ind = find(meta.order==2 & ...
    meta.scale(:,1)==3 & ...
    meta.scale(:,2)==7);
plot(t(ind,:));

% calculate the energy of 1st-order
% coefficients
ind = find(meta.order==1);
E2 = norm(t(ind,:),'fro')^2;
```

Note that if two or more filter banks are used when calculating `S`, formatting is only possible if the low-pass filters ϕ have the same bandwidth, since otherwise scattering coefficients of different orders will have different resolutions and so cannot be fitted into the same matrix without resampling.

4 Database Computation

5 Affine Classifiers

6 Support Vector Classifiers