ScatNet Implementation Document

June 6, 2013

1 Introduction

2 The Scattering Transform

The scattering transform is implemented in ScatNet using the `scat` function. This function can be used to calculate several types of scattering transforms by varying the linear operators supplied to it.

2.1 The `scat` Function

The `scat` function takes as input a signal `x` and a set of linear operators `Wop` and outputs a scattering transform `S` and intermediate modulus coefficients `U`. A call to the `scat` thus looks like

```
[S, U] = scat(x, Wop);
```

Often, the modulus coefficients `U` are not necessary and so can be left out.

Outputs `S` and `U` are cell arrays, each element corresponding to a layer of the scattering transform. The format of these layers is described in the next subsection.

Each element of the `Wop` is a function handle, with signature

```
[A, V] = Wop{m+1}(X);
```

Note that operators are indexed starting at $m = 0$, so an offset of 1 is necessary for compatibility with MATLAB. Here, `X`, `A` and `V` are all in the network layer format described in the next subsection. The linear operator `Wop{m+1}` transforms the signals in the `X` into two new layers: an invariant layer `A` (average) and a covariant layer `V` (variations). In the case of a wavelet transform, `A` corresponds to the averaging of the lowpass filter ϕ while `V` consists of the wavelet coefficients obtained by convolving with ψ_j.

Initializing `U1` using the input signal, the following loop is then executed in `scat`

```
for m = 0:numel(Wop)-1
    if (m < numel(Wop)-1)
        [S{m+1}, V] = Wop{m+1}(U{m+1});
        U{m+2} = modulus_layer(V);
    else
        S{m+1} = Wop{m+1}(U{m+1});
    end
end
```

For each intermediate layer `U{m+1}`, we thus apply the linear operator `Wop{m+1}`, assigning the invariant output to the mth-order scattering layer `S{m+1}`, and computing the modulus of the covariant part to obtain the $(m+1)$th order intermediate coefficients `+2Um`. For the last layer, we do not need the intermediate coefficients, and so only compute the scattering coefficients.

2.2 Network Layers `S{m+1}` and Linear Operators `Wop{m+1}`

As mentioned earlier, each element of `S` and `U` are in the network layer format. These consist of two fields, `signal` and `meta`. The former is a cell array of signals and the latter is a structure containing information related to each signal.

Specifically, each of the fields in `meta` is an array with the same number of columns as the length of `signal`. For example, the `meta.resolution` field

has one row, indicating the extent the signal has been subsampled with respect to the original length. In the case of wavelet transforms used as linear operators, an important field is `meta.j`, which has a variable number of rows, describing the scales of the wavelets used to compute the coefficient. For a first-order coefficient $|x \star \psi_{j_1}| \star \phi(t)$, this will simply be one row containing j_1. For a second-order coefficient $||x \star \psi_{j_1}| \star \psi_{j_2}| \star \phi(t)$, the first row will contain j_1 while the second will contain j_2.

Let us consider the example output `S` from the `scat` function using wavelet transforms as linear operators. In this case, `S{m+1}` will contain the mth-order scattering coefficients, with `S{m+1}.signal{p}` being the pth signal among these. Its scales $(j_1, j_2, \ldots, j_m)$ are specified in `S{m+1}.meta.j(:,p)`.

As mentioned earlier, the `Wop{m+1}` function handles take as an input a network layer and outputs two layers, one invariant and one covariant. Any function with these inputs and outputs can be used as an element of `Wop`. Two basic functions can be used for this purpose: `wavelet_layer_1d` and `wavelet_layer_2d`, which define wavelet transforms on network layers. For example, supposing we have defined a filter bank `filters` (see next section), we can create an accompanying 1D wavelet transform by defining

```
1  Wop{m+1} = @(U)(wavelet_layer_1d(U, ...
       filters));
```