Wavelet Density Estimator Optimization

Yixin Lin, Glizela Taino, Mark Moyou, and Dr. Adrian Peter

July 7, 2016

Using Wavelet Density Estimation in shape matching was explored by Dr. Adrian M. Peter and Dr. Anand Rangarajan in [**?**]. It estimates probability density functions of sample data whose function are unknown. It does this by using Wavelets theory and applying it to density estimation. It's used in applications such as 1D, 2D, and 3D shape matching. The coefficients that are used to scale the basis functions act as feature representations for a shape. These feature representations are constrained to one which conveniently map them onto a unit hypersphere. The geometry of the hypersphere can be easily manipulated for shape retrieval, in effectively recognizing similar and dissimilar shapes based on distance between coefficient vectors. The Wavelet Density Estimator code they developed is essential in extracting shape densities as features representations for various shapes. Computation involved for a sample point set can be very complicated. Therefore, the code is designed to execute the necessary computation. However, the issue with this code is the time it takes to process large databases. Databases such as Brown, Swedish Leaf, and MPEG7 have 99, 1125, an 1400 shapes, respectively, and can take days to process even with a relatively low resolution. This is due to the design and structure of the implementation. The code is designed to perform superfluous and redundant computations to solve for basis functions\ values for every sample point individually. As a solution to this problem we only calculated the basis functions for relevant sample points under specified basis functions with the hope to cut computation and time considerably. As wavelet density coefficients prove to be distinctly representative of their shape and can be useful in shape retrieval, the computational time to extract these coefficients become ever more important. If we wish to see timely growth in this area of research, hours for coefficient extraction for a database will not help in achieving this outlook. If we wish to see wavelet density coefficients used more commonly as represention, the wavelet density estimator must work more efficiently. In this report we will get a basic understanding of wavelets and how it's used in density estimation. Then we will delve into the code to expand its inefficiency, explain our solution to the problem, and lastly we will go over the very promising results from the optimization.

Wavelet Density Estimation

In order to create our feature representation we have to estimate the probabilityy density function of sample points that make up a shape. Wavelets act as the basis function of the function space L^2. An L^2 function is a function that consists of real numbers that is squared integrable, or finite. Probability density functions fall under this function space, therefore allowing us to use wavelets for density estimation using a linear combination of these basis functions. The equation for 1D density estimation is

$$p(x) = \sum_{j_o,k} \alpha_{j_o,k}\phi_{j_o,k}(x) + \sum_{j \geq j_o,k}^{j_1} \beta_{j,k}\psi_{j,k}(x) \tag{1}$$

where $\phi(x)$ and $\psi(x)$ are the scaling and wavelet basis functions, also known as the "father" and "mother" wavelets, respectively. The functions can also be set to some starting resolution level that determines its dilation and contraction. To extend to multiple resolutions, we incorporate the wavelet basis function with a starting, j_o, and stopping, j, resolution level. In order to form a basis, we need to make multiple copies of the function over different translates, k. These functions are scaled by their scaling and wavelet basis function coefficients, $\alpha_{j_o,k}$ and $\beta_{j,k}$. To retain properties of true densities, such as non-negativity and should integrate to one, we compute the square root of the probability density function

$$\sqrt{p(x)} = \sum_{j_o,k} \alpha_{j_o,k}\phi_{j_o,k}(x) + \sum_{j \geq j_o,k}^{j_1} \beta_{j,k}\psi_{j,k}(x) \tag{2}$$

Since we optimized for multi-resolution 2D wavelet density estimation, we consider the second dimension with $\mathbf{x} = (x_1, x_2) \in \mathbb{R}^2$ and $\mathbf{k} = (k_1, k_2) \in \mathbb{Z}^2$. We also extend the equation to

$$\sqrt{p(\mathbf{x})} = \sum_{j_o,\mathbf{k}} \alpha_{j_o,\mathbf{k}}\phi_{j_o,\mathbf{k}}(\mathbf{x}) + \sum_{j \geq j_o,\mathbf{k}}^{j_1} \sum_{w=1}^{3} \beta^w_{j,\mathbf{k}}\psi^w_{j,\mathbf{k}}(\mathbf{x}) \tag{3}$$

where we calculate for the basis functions by using the tensor product method

$$\begin{aligned}
\phi_{j_o,\mathbf{k}}(\mathbf{x}) &= 2^{j_o}\phi(2^{j_o}x_1 - k_1)\phi(2^{j_o}x_2 - k_2) \\
\psi^1_{j,\mathbf{k}}(\mathbf{x}) &= 2^{j}\phi(2^{j}x_1 - k_1)\psi(2^{j}x_2 - k_2) \\
\psi^2_{j,\mathbf{k}}(\mathbf{x}) &= 2^{j}\psi(2^{j}x_1 - k_1)\phi(2^{j}x_2 - k_2) \\
\psi^3_{j,\mathbf{k}}(\mathbf{x}) &= 2^{j}\psi(2^{j}x_1 - k_1)\psi(2^{j}x_2 - k_2)
\end{aligned} \tag{4}$$

A mathematical advantage in using wavelets is its compact support, meaning they are nonzero on a small domain. Any point falling out of the basis function's support does not contribute to the basis function value at that translation. Intuitively, the scaling and wavelet basis function coefficients determine the

height of its basis function. Therefore, our goal is to obtain the coefficients which uniquely define the probability density function of a shape and use as a feature representation in shape retrieval. In order to calculate coefficients we use the following equations

$$\alpha_{j_o,\mathbf{k}} = \left(\frac{1}{N}\right)\sum_{i=1}^{N}\frac{\phi_{j_o,\mathbf{k}}(\mathbf{x})}{\sqrt{p(\mathbf{x})}} \tag{5}$$

$$\beta_{j_o,k} = \left(\frac{1}{N}\right)\sum_{i=1}^{N}\frac{\psi_{j,\mathbf{k}(\mathbf{x})}}{\sqrt{p(\mathbf{x})}} \tag{6}$$

As we estimate the coefficients of our probability density function, we use a loss function to determine whether our coefficients are as close to optimal as possible. To do this we try to minimize a negative loglikelihood objective function with respect to the coefficients

$$-\log p(X;\{\alpha_{j_o,k}\beta_{j,k}\}) = -\frac{1}{N}\sum_{i=1}^{N}\log\left[\sum_{j_o,k}\alpha_{j_o,k}\phi_{j_o,k}(x_i) + \sum_{j\geq j_o,k}^{j_1}\beta_{j,k}\psi_{j.k}(x_i)\right]^2 \tag{7}$$

where $X = \{x_i\}_{i=1}^{N}$. And in order to determine the direction of the gradient descent we compute the following

$$-2\left(\frac{1}{N}\right)\sum\frac{\phi_{j_o,\mathbf{k}}(\mathbf{x})}{\sum\alpha_{j_o,\mathbf{k}}\phi_{j_o,\mathbf{k}}(\mathbf{x})} \tag{8}$$

In order for maintain essential properties of density estimation, we constrain the coefficients to one

$$\sum_{j_o,k}\alpha_{j_o,k}^2 + \sum_{j\geq j_o,k}^{j_1}\beta_{j,k}^2 = 1 \tag{9}$$

This constraint conveniently maps our feature representation into a unit hypersphere which allows us to take advantage of its geometric properties to signify similarity and dissimilarity between shapes in shape retrieval.

Wavelet Density Estimation Optimization

Now that we have an overview of wavelet density estimation and the equations used, we can delve into our optimization. Our ultimate goal is to estimate a 2D density function on a given point set shape representation. We then use the coefficients that uniquely characterize the density function as our feature vector. Originally, for a data base such as MPEG7 made up of 1400 shapes with a domain of [-0.5,0.5], resolution level 2 to 3, and 1344 translates it would take roughly 16.8 hours. This is a particularly low resolution and if raised we would expect an even slower runtime. We want to optimize this code for speed. The bottleneck occurs when calculating the initial coefficients and negative log likelihood cost value with its gradient.

Initializing Coefficients

This function sets out to calculate equations (5) and (6) to compute for estimated initial coefficients that will be updated later in the code for the wavelet density estimation. The time spent for this calculation with the same parameters mentioned above is roughly around 1.8 of the 16.8 hours.

The reason for this bottleneck is because of the way the code executes computation. Given a point set shape representation, it covers the shape with basis functions at each translation. Each function has a coefficient that needs to be calculated. The problem is that the code computes equations (4) for a *single point over all translations* using a MATLAB function called Kronecker Tensor Product [CITE]. This means that if we have 1344 translations, it would perform 1344 operations. If a shape is made up of 4007 sample points, then it would be performing a total of 5,385,408 operations for a single shape. This calls for an excess amount of redundant computations since most of the scaling and wavelet basis functions for a single point over all translations return the value of zero. It would return zero because a specified observed sample point does not contribute to most basis functions, or does not exist under the compact support of the basis function at specific translations. Therefore, our solution was to find an approach that would require less computation and, overall, work more effectively and efficiently.

Instead of looping through each point and finding which basis functions it falls under, we looped through basis functions along the horizontal direction, evaluated its basis function value (4) based on the points fall under its support, and placed them into a matrix that holds these values. We then store this matrix to be used for later optimized computation (for the negative log likelihood cost value and gradient). Once we have this matrix of values, this allows us to evaluate the basis function coefficients with the relevant points under each translation using equation (5) and (6).

Negative Log Likelihood

The second area where the bottleneck occurs is in computing for the negative log likelihood cost value (7) and the function's gradient (8). We use these equations to check whether we have found the most optimal coefficient values and which direction in which to descend. The original time spent for this computation was around 2 of the 16.8 hours.

The problem with the code was similar to the execution of initialize coefficients. It would take a single point and calculate the basis function over all translations resulting in needless computation. It does this to attain a matrix of the basis function values to perform the appropriate operations to solve for the cost value and gradient.

Since we already performed most of the heavy computation to attain this matrix in initializing coefficients, our solution was to simply pass in the needed matrix of basis function values for each translation and perform the proper computations. This effectively solves for the negative log likelihood cost value

and gradient while ultimately eliminating loops.

Results

As mentioned above, our goal is to estimate a density function of a 2D shape using wavelets. This allows us to extract the coefficients as our feature representation to be used in shape retrieval on a unit hypersphere. However, with MPEG7 containing 1400 shapes, domain [-0.5,0.5], resolution level 2 to 3, and 1344 translates it would take about 16.8 hours to calculate the coefficients. After our optimization implementation to the two areas where the code bottlenecks , we yield fantastic results.

The original time it took the code to calculate for only the initial coefficients was about 1.8 hours. After our optimization, the run time to compute the initial coefficients cut all the way down to 0.13 hours, or about eight minutes. Ultimately, the computation runs 92.8% faster. As for calculations for the negative log likelihood cost value and gradient, our optimization shaved the lines of code from 54 lines to 21 lines, ridding it of loops. We essentially do away with 61% of lines of code. The run time was promising as well. To calculate the cost value and gradient for seven iterations, computation time went from 13.9 hours to 0.089 hours, or around 5 minutes for the entire dataset. The optimized computation runs 99% faster. Overall, to estimate the wavelet densities of a database of 2D shapes with the optimized code under the specified parameters would take 0.3 hours, or 17.8 minutes, running 98% faster.

With this optimization, we hope that we can spend less time waiting for the wavelet density coefficient values to be estimated and more time in developing new and better methods of shape retrieval using wavelet density coefficients as feature vectors.

References