Image Denoising with Kernel Regression from the Bayesian Approach

Novi Wang, Tong Wu, Xinyu Zhang
Washington University in St. Louis

Abstract

In this project, we compare some previous works on image denoising by kernel regression with our proposed solution, which is guided under the Bayesian framework. The crux of kernel regression is to define the associated weights. Differing from most of the previous works, which assign the weights by considering the data relative location, we optimize the kernel weights purely based on the image intensity values. We formulate the problem in a way such that the regression model becomes the mean of the predictive distribution. By *maximizing a posterior* (MAP) using gradient descent approach, we can therefore do inference by model averaging to assign predicted intensity value to each point. We make comparison between our solution and the classical Gaussian filtering algorithm. The results suggest that our solution can achieve even a better result than the best tuning Gaussian filtering.

Introduction

High quality images are essential prerequisite in many areas such as medical image analysis, criminal suspects detection, and autonomous cars image recognition. With such demands, different image denoising methods, like the frequency domain filtering, are proposed and applied. As suggested by many previous works, one can view the image denosing as a regression problem in a sense that the query point is inferred based on how the training samples look like. Among all the regression strategies that have been proposed, the kernel regression is a non-parametric approach, which relies on the data itself and generally does not assume any statistical distribution about the image noise. Therefore, it is an effective and efficient solution and is also widely used.

The key idea of the kernel regression is to have a kernel which underscores the distance between points. Such a distance estimation implies the weights in a regression model and therefore becomes the central part of the entire algorithm. Many previous works give their own approaches of estimating the weights. In this paper, we firstly review two widely used kernel regression in image denosing. Based on these two approaches, we propose our solution, which is guided by the Bayesian framework.

Related Work

In order to have a better understanding of the kernel regression and be able to compare the result of separate image denoising algorithms, we first replicate two kernel regression methods, which are highly related to our work. The first regression method is simple and also denoted as the Gaussian filtering in many context. This approach focuses the spatial relations between pixels in the kernel. It estimates the query point by averaging all the points with different weights, which are given by the Gaussian kernel.

$$\hat{r}(x) = \frac{\sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right) \cdot y_i}{\sum_{i=1}^{n} K\left(\frac{x_i - x}{h}\right)} \tag{1}$$

$$K(x_1, x_2) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{x_1 - x_2}{\sigma}\right)^2} \tag{2}$$

Eq. (1) is also well known as the Nadaraya Watson estimator. However, such an approach does not consider the input data itself. Therefore, the Gaussian filtering performs well when the image texture is smooth but badly at the sharp edges.
On the other hand, we have found that some regression methods take the advantage of basis functions. In the paper, (Takeda, Farsiu, and Milanfar 2007) presents the approach of using a linear combination of basis function to present the regression function.

$$y(x) = \mathbf{w}^T \phi(x) + w_0 \tag{3}$$

The weights of each basis function play a key role in this formula. Takeda proceeds by suggesting the usage of equivalent kernel. Readers may refer to the original work for detailed information. But it is worth to notice that this approach still takes the relative location of pixels in the kernel as its input. Therefore, in this paper, we try to formulate an image denoising method, which uses the kernel regression but purely based on the intensity values as its input.

Bayesian Image Denoising

In this work, we assume that a Gaussian noise, which has unknown variance σ^2, is added to the original clean image. The observed intensity is from the following distribution of form:

$$t_i \sim N\left(y_i, \sigma^2\right) \tag{4}$$

where y_i is the output of the Bayesian Regression and t_i is the observed intensity in the noisy picture. In this case, we first formulate the prior and likelihood of one kernel and then generate the algorithm to the whole image. According to our assumption, the likelihood of the data in a given kernel is formulated to:

$$p(D|M) = \prod_{i=1}^{K} \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(t_i - y_i)^2}{2\sigma^2}\right) \tag{5}$$

where D denotes all observed data $t_1, t_2 ... t_k$ in the kernel, M is the regression model and K is the kernel size.
Same as (Tipping 2001)'s work, we present our regression function, $y_i(x_i; \mathbf{w})$, as a linear combination of basis function, with the weights of basis functions being $\mathbf{w}$. We use nonlinear transformation (kernel) to map the input vector x_i to $\phi(x_i)$. Then, we can formulate y_i as following:

$$\begin{aligned} &y_i(x_i; \mathbf{w}) = \mathbf{w}^T \phi(x_i) + w_0 \ \ , where \\ &\phi\left(x_i\right) = \left[\overline{K}\left(x_i, x_1\right), \overline{K}\left(x_i, x_2\right), \ldots, \overline{K}\left(x_i, x_k\right)\right]^{\mathrm{T}} \\ &\overline{K}(x_i, x') = \exp\left(-\frac{\left\|x_i - x'\right\|^2}{2r^2}\right) \end{aligned} \tag{6}$$

where $\overline{K}$ denotes the nonlinear transformation kernel for x_i. We choose the RBF kernel here since it is widely used in kernel regression models. Inside the RBF kernel equation, r is a width of the Gaussian functions and we set it as $r = \frac{\sqrt{K}}{2.5}$. (We will discuss this choice in Future Work Section).

Here, we use a Bayesian perspective and obey a prior probability distribution for our model, namely $\mathbf{w}$ and w_0. For weight $\mathbf{w}$, we choose zero mean Gaussian distribution for each w_i inside $\mathbf{w}$, since it always works for smoother functions. In addition, we also select Gaussian distribution as the prior of w_0. Instead of a zero mean distribution, the mean value of w_0 should be the the mean intensity value m in the kernel. Hence, we get $w_i \sim N\left(0, \sigma_i^2\right)$, and $w_0 \sim N\left(m, \sigma_0^2\right)$. Then we define the corresponding precision parameter (inverted variance) α for both Gaussian distributions, where $\alpha_i = 1/\sigma_i^2$ and $\alpha_0 = 1/\sigma_0^2$. Therefore, the whole prior distribution of the model $P(M)$ after calculation is given by:

$$\frac{1}{(2\pi)^K} \prod_{i=1}^{K} \alpha_i^{1/2} \exp(-\frac{\alpha_i w_i^2}{2}) \alpha_0^{1/2} \exp(-\frac{\alpha_0 (w_0 - m)^2}{2}) \tag{7}$$

Having defined the prior and likelihood, according to the Bayesian rule, we then come up the posterior over all unknowns given the data.

$$p(M|D) = \frac{p(D|M)p(M)}{\int p(D|M)p(M)dM} \tag{8}$$

Note that the denominator in this formula is a constant which we don't need to worry about when optimizing the posterior probability (applying MAP). Then we derive the model (weight) by:

$$M^* = \arg\max_M p(D|M)p(M) \tag{9}$$

This is equivalent to maximize the log of $p(D|M)p(M)$, then we derive the formula as following:

$$\begin{aligned} &L\left(\mathbf{w}, \alpha_i, w_0, \alpha_0, \sigma\right) = -K \ln(\sigma) - \textstyle\sum_{i=1}^{K} \frac{(t_i - y_i)^2}{2\sigma^2} \\ &- \textstyle\sum_{i=1}^{K} w_i^2 \frac{\alpha_i}{2} + \sum_{i=1}^{K} \frac{1}{2} \ln\left(\alpha_i\right) - \frac{(w_0 - m)^2}{2} \alpha_0 + \frac{1}{2} \ln\left(\alpha_0\right) \end{aligned} \tag{10}$$

Here, L is the log of posterior distribution by substituting the Eq. (5), Eq. (7) and Eq. (9). We then calculate the gradient of L with respect to $w_k, w_0, \alpha_k, \alpha_0$ similar with (Cohen and Ben-Ari)'s method, since we want to find out the argmax of the L. Therefore, the gradients are represented as following:

$$\frac{\partial L}{\partial w_k} = -\frac{\sum_{i=1}^{K} \left(y_i - t_i\right) \phi_k\left(t_i\right)}{\sigma^2} - w_k \alpha_k \tag{11}$$

$$\frac{\partial L}{\partial w_0} = -\frac{\sum_{i=1}^{K} \left(y_i - t_i\right)}{\sigma^2} - \left(w_0 - m\right) \alpha_0 \tag{12}$$

$$\frac{\partial L}{\partial \alpha_k} = \frac{-w_k^2}{2} + \frac{1}{2\alpha_k} = 0 \tag{13}$$

$$\frac{\partial L}{\partial \alpha_0} = \frac{1}{2\alpha_0} - \frac{\left(w_0 - m\right)^2}{2} = 0 \tag{14}$$

For the estimation of noise standard deviation σ, we use the none biased estimate as follows:

$$\sigma^2 = \frac{\sum_{i=1}^{K} \left(y_i - t_i\right)^2}{K - 1} \tag{15}$$

The gradient of L with respect to w_k, w_0 showed on Eq. (11) and Eq. (12) cannot be calculated by the closed form. Hence, we use iterative gradient algorithm to achieve the best result. For most time, after several iterations, the L will be converged. The whole algorithm is showed on Algorithm 1.

Bayesian Inference

In this part, we introduce how to use result y to do the Bayesian inference of each point. Our previous algorithm is basically focused on single kernel, which gives us the optimal model and unknown noise standard deviation σ. To make prediction, we consider total number of K kernels, each of which includes the pixel to be inferred. However, recall that each model is not only giving predictions on the inference point but instead all points in that kernel. Therefore, we modify the Bayesian model averaging strategy by firstly using each model to inference that point, then averaging all the predictions on this point to output the final decision. The equation to make prediction is defined as follows:

$$\begin{aligned} &p\left(t_* | \mathbf{t}, \alpha_{\mathrm{MP}}, \sigma_{\mathrm{MP}}^2\right) = \\ &\int p\left(t_* | \mathbf{w}, \sigma_{\mathrm{MP}}^2\right) p\left(\mathbf{w} | \mathbf{t}, \alpha_{\mathrm{MP}}, \sigma_{\mathrm{MP}}^2\right) d\mathbf{w} \end{aligned} \tag{16}$$

Algorithm 1 Image Denoising: Learning *MAP*

Input:
Kernel Intensity: $T : t_1, t_2, ..., t_k$
Total iterations: N
Tolerance: ϵ
Learning rate: η
Output:
Weights: $w_k, w_0, \alpha_k, \alpha_0$ as well as σ^2

1.Initialize $w_k, w_0, \alpha_k, \alpha_0$ using our assumption
2.**for** m **in** 0 to N **do**:
3. $w_k^{m+1} = w_k^m + \eta * \frac{\partial L}{\partial w_k}$ from Eq.(11)
4. $w_0^{m+1} = w_0^m + \eta * \frac{\partial L}{\partial w_0}$ from Eq.(12)
5. Update the α_k, α_0 from Eq.(13) and Eq.(14)
6. Update the noise estimation σ from Eq.(15)
7. **if** $\frac{\partial L}{\partial w_k} \leq \epsilon$:
8. **break**
9. **end for**

,where the above equation can be formed as the following (Tipping 2001), which takes advantage of the property of Gaussian distribution:

$$p\left(t_*|\mathbf{t}, \alpha_{\mathrm{MP}}, \sigma^2_{\mathrm{MP}}\right) = \mathcal{N}\left(t_*|y_*, \sigma_*^2\right) \tag{17}$$

Algorithm 2 Image Denoising: making prediction

Input:
For each pixel:
Optimal Models: $M_1, ..., M_K$ each contains the inference point
Optimal Variance: $\sigma_1^2, ..., \sigma_K^2$
Output:
For each pixel:
The final decision of the intensity value t_*

1.**for** each pixel **in** in input image **do**:
2. **for** each model **in** optimal models **do**:
3. Predict y_i
4. **end for**
5. Compute $\overline{\sigma^2} = \frac{\sum_{k=1}^{K} \sigma_k^2}{K}$
6. Inference the current pixel, which is weighted by $N\left(t_*; 0, \overline{\sigma^2}\right)$
9. **end for**

Experiment

In this section, we discuss the performance of our proposed solution. The classical Gaussian filtering algorithm is implemented and used for comparison. For the test design, we used the standard "Lena" image and manually added a Gaussian noise. Notice that this noise is unknown to the image denoising algorithm. Our goal is to test the performance of our algorithm by varying the standard deviation of this Gaussian noise. The evaluation is based on the mean square error (MSE) between the original image and the denoised image. On the other hand, we varied the size of kernel. It is well known that the denoised image will be blurred too much if the kernel size is too large in the Gaussian filtering. Then, this test is to check if our algorithm can preserve more image information than the Gaussian filtering does.

Figure 1: "Lena" clean image

Figure 2: "Lena" with Gaussian Noise, $\sigma = 30$

Figure 3: "Lena" with Gaussian Noise, $\sigma = 100$

For the input data, specifically, the standard "Lena" image with size $512X512$ was used and shown on Figure 1. We separately added two Gaussian noises, with standard deviation $\sigma = \{30, 100\}$, on this image. Then, each of Figure 2 and 3 was the input of the image denoising algorithm.
Test 1 restricted the kernel size to be $3X3$. To make the comparison valid, we explicitly set the hyperparameter bandwidth in the Gaussian Filtering to be 0.5 based on some empirical suggestions:

$$Radius = 2 * bandwidth$$

,where $ceil(2 * radius) = kernelwidth$. For our algorithm, we firstly padded the image by $kernelwidth/2$. The entire process was performed pixel-wise. For each pixel, we performed the Algorithm 1, where the learning rate η is set to be 0.01, and then we performed the Algorithm 2 for inference. The results are displayed in Figure 4, 5, 6, 7.

Visually speaking, these results almost have no difference between each other. This can be confirmed by the evaluated mean square error. However, recall that in order to make the Gaussian filtering to have such a great performance, users have to explicitly set up the bandwidth in the Gaussian kernel. Without knowing the benchmark, tuning this hyperpa-

Figure 4: Proposed solution, noise $\sigma = 30$

Figure 5: Gaussian filtering, noise $\sigma = 30$

Figure 6: Proposed solution, noise $\sigma = 100$

Figure 7: Gaussian Filtering, noise $\sigma = 100$

	$\sigma = 30$	$\sigma = 100$
Bayesian Denoising Algorithm	106.9082	544.3075
Gaussian Filtering	115.793	429.9099

Table 1: Mean Square Error between the original image and denoised image.

rameter takes extra effort. Instead, without tuning any hyperparameter, our algorithm gives a performance which is almost identical to the best result given by the classical Gaussian filtering.

For test 2, we used Figure 3 as the input of denoising algorithm. This time, we specified the kernel size to be $5X5$. Similar to what we did in the test 1, we explicitly set the hyperparameter in the Gaussian filtering to be 1 based on the empirical suggestion. And again, there was no hyperparameter required for our proposed solution.

	$\sigma = 100, 5X5$
Bayesian Denoising Algorithm	388.2368
Gaussian Filtering	416.1532

Table 2: Mean Square Error between the original image and denoised image. Kernel size $= 5X5$

The results of test 2 achieves our expectation as the follows. Firstly, when the kernel size enlarges to $5X5$, based on

Figure 8: Proposed solution, kernel size $5X5$

Figure 9: Gaussian Filtering, kernel size $5X5$

the MSE value, the proposed solution is able to perform as well as the best Gaussian filtering algorithm. Our algorithm even has a lower MSE value, which means its denoised image is closer to the original clean image. As we carefully examine the details on this pair of images, we see that our solution is closer to the clean image because it maintains more texture information. For those parts, such as hair, hat edge, etc., our solution is more detailed and lighter than the other solution. In short, with the same kernel size, the classical approach blurs the image too much, while our solution is able to preserve details, resulting in a lower MSE.

Future Work

Although we get promising results for image denoising described in previous section, there are still lots of works could be done in the future. For our kernel regression model, setting the hyperparameter as $r = \frac{\sqrt{K}}{2.5}$ is an arbitrary way. It is better for us to tune r a little bit or use Bayesian method and find out the best result. Other than that, RBF kernel is just one of the common kernels used in research and more kernels such as Squared Exponential Kernel, Rational Quadratic Kernel can be applied in our algorithm. With that, embedding all the models in the prediction of regression is another way to improve our algorithm.

For Bayesian Inference, instead of averaging the predictions, (Tipping 2001) also provides another method to derive the final restored image, using the following formula:

$$\begin{aligned} & p\left(t_* | \mathbf{t}, \alpha_{\mathrm{MP}}, \sigma_{\mathrm{MP}}^2\right) = \mathcal{N}\left(t_* | y_*, \sigma_*^2\right)) \\ & y_* = \mu^{\mathrm{T}} \phi\left(\mathbf{x}_*\right) \\ & \sigma_*^2 = \sigma^2 + \phi\left(\mathbf{x}_*\right)^{\mathrm{T}} \Sigma \phi\left(\mathbf{x}_*\right) \end{aligned} \tag{18}$$

In practice, setting all weights to be fixed μ is a common way to do the point estimation of t_*.

Conclusion

In this work, We formulate image denoising as a simple kernel regression problem. Using Bayesian approach (MAP), we can therefore come up the weights of model and get the intensity inference of each point. We clearly show that our approach is much better than the baseline Gaussian approach in preserving image details. In the future, we can improve

our method via careful tuning parameter, combining different kernel models, and using different inference approach.

References

[Cohen and Ben-Ari] Cohen, S., and Ben-Ari, R. Image denoising by bayesian regression.

[Takeda, Farsiu, and Milanfar 2007] Takeda, H.; Farsiu, S.; and Milanfar, P. 2007. Kernel regression for image processing and reconstruction. *IEEE Transactions on Image Processing* 16(2):349366.

[Tipping 2001] Tipping, M. E. 2001. Sparse bayesian learning and the relevance vector machine. *J. Mach. Learn. Res.* 1:211–244.

Matlab Code*

`https://github.com/AlanZhang0817/BayesianKernelRegression.git`