KTH Electrical Engineering

EQ2415 Machine Learning and Data Science Project Report

YUE SONG, YIZHAN YANG

Stockholm September 2019

1 Assignment I: Regularized Linear Regression

The original *linear regression* is formulated as:

$$\mathbf{y} = \mathbf{w}^T \phi(\mathbf{x}) \tag{1}$$

where $\mathbf{w}$ is the parameters of interest. Take the *weight decay* into consideration, the cost function then can be expressed as:

$$\mathcal{J}(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}_n\mathbf{x}_n\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \tag{2}$$

where t is the target output, λ is the regularization wight which controls the relative importance between $fitness$ and *weight decay*. Setting the derivative of cost $\frac{\partial \mathcal{J}(\mathbf{w})}{\partial \mathbf{w}}$ to zero, we can obtain the optimal solution:

$$\mathbf{w} = (\lambda\mathbf{I} + \mathbf{\Phi}^T\mathbf{\Phi})^{-1}\mathbf{\Phi}^T\mathbf{t} \tag{3}$$

In experiments, we set the value $\lambda = \{0.05, 0.1, 0.5, 1\}$ on several datasets and use cross-validation to manually choose the best value based on training accuracy.

Dataset	Test Accuracy(%)	Best λ	Training NME	Testing NME	Training Time(s)
Vowel	27.7±4.7	0.5	-0.47	-0.72	0.00
Extended YaleB	95.6±1.2	0.05	-1.29	-0.98	0.31
AR	92.4±1.2	0.05	-0.63	-0.53	0.39
Satimage	78.4±1.4	0.5	-1.51e+3	-1.64e+3	0.01
Scene15	94.3±0.7	0.5	-27.71	-27.57	47.48
Caltech101	61.2±1.8	0.1	-325.87	-306.56	59.85

Table 1: Experimental results using Regularized Linear Regression

2 Assignment II: Gaussian Kernel Regression

Many linear parametric models can be re-cast into an equivalent 'dual representation' in which the predictions are also based on linear combinations of a *kernel function* evaluated at the training points. For models which are based on a fixed nonlinear *feature space* mapping $\phi(\mathbf{x})$, the kernel function is a symmetric function given by

$$k(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}', \mathbf{x}) = \phi(\mathbf{x})^T\phi(\mathbf{x}') \tag{4}$$

We can reformulate the cost function of a *ridge regression*:

$$\begin{aligned} \mathcal{J}(\mathbf{w}) &= \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^T\mathbf{w} \\ \mathbf{w} &= -\frac{1}{\lambda}\sum_{n=1}^{N}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\}\phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) = \mathbf{\Phi^T a} \end{aligned} \tag{5}$$

where $\mathbf{\Phi}$ is the design matrix and $\mathbf{a}$ is the vector defined as

$$a_n = -\frac{1}{\lambda}\{\mathbf{w}^T\phi(\mathbf{x}_n) - t_n\} \tag{6}$$

If we substitute $\mathbf{w} = \mathbf{\Phi^T a}$ and *Gram* matrix $\mathbf{K} = \mathbf{\Phi\Phi^T}$ into $\mathcal{J}(\mathbf{w})$, we can obtain the *dual representation* as:

$$\mathcal{J}(\mathbf{a}) = \frac{1}{2}\mathbf{a^T K K a} - \mathbf{a^T K t} + \frac{1}{2}\mathbf{t^T t} + \frac{\lambda}{2}\mathbf{a^T K a} \tag{7}$$

Setting the derivative of $\mathcal{J}(\mathbf{a})$ to zero, we can obtain

$$\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I_N})^{-1}\mathbf{t} \tag{8}$$

Substitute this back into original regression model, then we get:

$$y(x) = \mathbf{w}^T\phi(x) = \mathbf{a^T \Phi}\phi(x) = \mathbf{k}(x)(\mathbf{K} + \lambda \mathbf{I_N})^{-1}\mathbf{t} \tag{9}$$

We choose Gaussian kernels as our knernel function, which can be defined as

$$k(x, x') = -\frac{exp(-\|x - x'\|^2)}{2\sigma^2} \tag{10}$$

Here we first normalize all the features and choose $\sigma = 0.5$ and $\lambda = 0.05$ manually.

Dataset	Test Accuracy(%)	Training NME	Testing NME	Training Time(s)
Vowel	61.3±1.6	-2.79	-49.92	1.69
Extended YaleB	94.0±1.1	-0.08	-0.05	47.15
AR	94.9±0.5	-0.11	-0.06	61.51
Satimage	82.9±1.0	-1.94e-4	-5.91e-5	147.14
Scene15	51.3±1.5	-0.87	-0.98	1126.60
Caltech101	69.0±0.5	-0.42	-0.44	4439.51

Table 2: Experimental results using Gaussian Kernel Regression

3 Assignment III: Gaussian Process

In the Gaussian process viewpoint, we dispense with parametric model and instead define a prior probability distribution over function $y(\mathbf{x}|\mathbf{w})$. For a linear regression model, we have:

$$y(x) = \mathbf{w}^T\phi(x) \tag{11}$$

Consider a prior distribution of $\mathbf{w}$ by an isotropic Gaussian of form:

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|0, \alpha\mathbf{I}^{-1}) \tag{12}$$

where α governs the precision of distribution. Therefore we can find the mean and variance of $\mathbf{y}$:

$$\begin{aligned} \mathrm{E}[\mathbf{y}] &= \Phi\mathrm{E}[\mathbf{w}] = 0 \\ cov[\mathbf{y}] = \mathrm{E}[\mathbf{y}\mathbf{y}^\mathbf{T}] &= \mathbf{\Phi}\mathrm{E}[\mathbf{w}\mathbf{w}^\mathbf{T}]\mathbf{\Phi}^T = \frac{1}{\alpha}\Phi\Phi^T = \mathbf{K} \end{aligned} \tag{13}$$

where $\mathbf{K}$ is the Gram matrix ($K_{nm} = k(x_n, x_m)$)) and $k(x, x')$ is the kernel function. If we take into account of the noise on the observed target value as

$$t_n = y_n + \epsilon_n \tag{14}$$

If we further assume the noise ϵ_n is independently Gaussian distributed, the expression can be denoted as

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1}) \tag{15}$$

The conditional distribution of $\mathbf{t}$ and $\mathbf{y}$ are also given by isotropic Gaussian form

$$\begin{aligned} p(\mathbf{t}|\mathbf{y}) &= \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}\mathbf{I}_N) \\ p(\mathbf{y}) &= \mathcal{N}(\mathbf{y}|0, \mathbf{K}) \end{aligned} \tag{16}$$

The marginal distribution of t is then given by

$$p(\mathbf{t}) = \int p(\mathbf{t}|\mathbf{y})p(\mathbf{y})d\mathbf{y} = \mathcal{N}(\mathbf{t}|\mathbf{0}, \mathbf{C}) \tag{17}$$

where the covariance matrix $\mathbf{C}$ has elements

$$C(x_n, x_m) = k(x_n, x_m) + \beta\delta_{nm} \tag{18}$$

The joint distribution of $\mathbf{t}_{N+1}$ is given by

$$p(\mathbf{t_{N+1}}) = \mathcal{N}(\mathbf{t_{N+1}}|\mathbf{0}, \mathbf{C_{N+1}}) \tag{19}$$

where $\mathbf{C_{N+1}}$ can be partitioned by

$$\mathbf{C_{N+1}} = \begin{pmatrix} \mathbf{C} & \mathbf{k} \\ \mathbf{k}^T & k(x_{N+1}, x_{N+1}) + \beta^{-1} \end{pmatrix} \tag{20}$$

The conditional distribution $p(t_{N+1})$ is a Gaussian distribution with mean and variance by

$$\begin{aligned} m(\mathbf{x}_{N+1}) &= \mathbf{k}^T\mathbf{C_N^{-1}}\mathbf{t} \\ \sigma^2(\mathbf{x}_{N+1}) &= k(x_{N+1}, x_{N+1}) + \beta^{-1} - \mathbf{k}^T\mathbf{C_N^{-1}}\mathbf{k} \end{aligned} \tag{21}$$

We use the mean value as the prediction of Gaussian process regression.

Dataset	Test Accuracy(%)	Training NME	Testing NME	Training Time(s)
Vowel	39.2±2.0	-0.30	-0.26	14.52
Extended YaleB	66.7±0.6	37.59	-40.17	450.12
AR	73.2±1.1	-196.23	-223.56	870.93
Satimage	83.4±0.7	-0.63	-0.81	1948.76
Scene15	47.3±1.2	-18.03	-26.56	19548.28
Caltech101	54.11±0.6	-483.24	-495.73	71226.46

Table 3: Experimental results using Gaussian Kernel Process

4 Assignment IV: Support Vector Machine

For a two-class classification problem using linear model, the problem can be stated as

$$\mathbf{y} = \mathbf{w}^T\phi(\mathbf{x}) + b \tag{22}$$

where $\phi(\mathbf{x})$ denotes a fixed feature-space transformation. The training dataset comprises N input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$, with corresponding target values $t_n \in \{-1, 1\}$, and new data points are classified according to the sign of $y(\mathbf{x})$. If

the decision surface is decided by $y(\mathbf{x}) = 0$, the distance of a point $\mathbf{x}_n$ to the decision surface is given by

$$\frac{t_n y_{x_n}}{\mathbf{w}} = \frac{t_n(\mathbf{w}^T\phi(x_n) + b)}{\mathbf{w}} \tag{23}$$

The margin is given by the perpendicular distance to the closest point $\mathbf{x}_n$ from the dataset, thus the maximum margin solution is found by solving:

$$\arg\max_{\mathbf{w},b}\{\frac{1}{\|\mathbf{w}\|}\min_n[\mathbf{w}^T\phi(x_n) + b]\} \tag{24}$$

If we rescale $\mathbf{w}$ and b, the distance from any point $\mathbf{x}_n$ to the decision surface is unchanged. Thus, we can use this freedom to set closest point

$$t_n(\mathbf{w}^T\phi(x_n) + b) = 1 \tag{25}$$

For any other points in the dataset, they should satisfy

$$t_n(\mathbf{w}^T\phi(x_n) + b) \geq 1,\ n = 1, \ldots, N. \tag{26}$$

This is known as the *canonical representation* of decision hyperplane. Then the optimization problem becomes minimizing $\|\mathbf{w}\|$. By introducing Lagrange multipliers $a_n \geq 0$, we have the Lagrange function as

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}\phi(x_n) + b) - 1\} \tag{27}$$

where $\mathbf{a} = (a_1, \ldots, a_N)^T$. Setting the derivative of $L(\mathbf{w}, b, \mathbf{a})$ with regards to $\mathbf{w}$ and b to zero, we obtain the following two conditions

$$\begin{aligned} \mathbf{w} &= \sum_{n=1}^{N} a_n t_n \phi(x_n) \\ \sum_{n=1}^{N} a_n t_n &= 0 \end{aligned} \tag{28}$$

If the training data points are not linearly separable in the feature space, we can introduce *slack variable* $\xi \geq 0$ to allow some points to be misclassified. The classification constraint can be expressed as

$$t_n y(x_n) \geq 1 - \xi_n,\ n = 1, \ldots, N \tag{29}$$

Introducing another multiplier $\mu_n \geq 0$, the corresponding Lagragian function becomes

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{n=1}^{N}\xi_n - \sum_{n=1}^{N} a_n\{t_n(\mathbf{w}\phi(x_n) + b) - 1 + \xi_n\} - \sum_{n=1}^{N}\mu_n\xi_n \tag{30}$$

Setting the derivative with regards to ξ_n to zero, we can obtain another condition

$$a_n = C - \mu_n \tag{31}$$

Eliminating $\mathbf{w}$ and b from $L(\mathbf{w}, b, \mathbf{a})$ gives the dual representation of the maximum margin problem in which we maximize

$$\widetilde{L}(\mathbf{a}) = \sum_{n=1}^{N} a_n - \frac{1}{2} \sum_{n=1}^{N} \sum_{m=1}^{N} a_n a_m t_n t_m k(x_n, x_m)$$
$$\text{subject to } 0 \leq a_n \leq C,\ n = 1, \ldots, N \text{ and } \sum_{n=1}^{N} a_n t_n = 0 \tag{32}$$

In order to classify new data, substitute $\mathbf{a}$ and the kernel function for $\mathbf{w}$ and we get

$$y(\mathbf{x}) = \sum_{n=1}^{N} a_n t_n k(\mathbf{x}, x_n) + b \tag{33}$$

We build soft-margin classifier using Gaussian kernel for binary classification and test our model on the first two classes of each dataset.

Dataset	Test Accuracy(%)	Training Time(s)
Vowel	80.95	0.89
Extended YaleB	77.94	0.97
AR	76.47	0.77
Satimage	65.93	104.11
Scene15	70.09	32.26
Caltech101	75.16	107.07

Table 4: Experimental results using Support Vector Machine (SVM)

5 Assignment V: Extreme Learning Machine

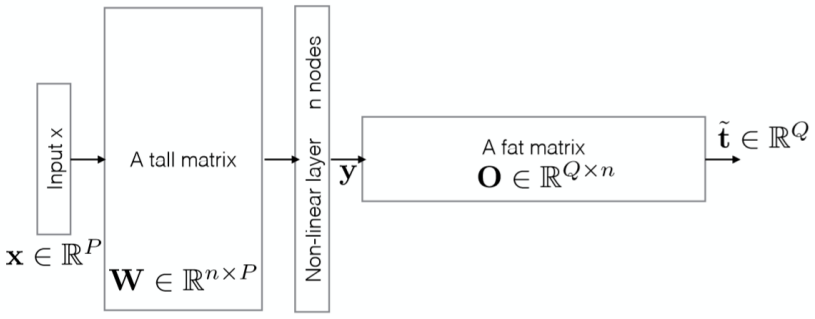

Figure 1: The architecture of Extreme Learning Machine (ELM)

As displayed in Fig. 1, the 1 layer ELM workflow follows:

$$\begin{aligned} \mathbf{z} &= \mathbf{w}\mathbf{x} + b \\ \mathbf{y} &= g(\mathbf{z}) \\ \hat{\mathbf{T}} &= \mathbf{O}\mathbf{y} \end{aligned} \tag{34}$$

where $g(\cdot)$ is a *nonlinear* transform. Our goal can be expressed as

$$\min_{\mathbf{O}} \left\| \mathbf{T} - \hat{\mathbf{T}} \right\|_2^2 \ s.t. \ \|\mathbf{O}\|_F^2 \leq \epsilon \tag{35}$$

The cost function can be expressed as

$$\mathcal{J}(\mathbf{O}) = \left\| \mathbf{T} - \hat{\mathbf{T}} \right\|_F^2 + \lambda \|\mathbf{O}\|_F^2 \tag{36}$$

Setting the derivative of cost function to zero, we can obtain the optimal $\mathbf{O}$

$$\mathbf{O}^* = \mathbf{T}\mathbf{y}^T(\mathbf{y}\mathbf{y}^T + \lambda\mathbf{I})^{-1} \tag{37}$$

We use radial basis transfer function $(exp(-x_n^2))$ as our nonlinear transform. We test the accuracy versus the number of nodes of both 1 layer ELM and 3 layer ELM respectively, then present the result in Fig 2. The 1 layer ELM tends to stablize when nodes is larger than number of features while 3-layer ELM still has the trend of increasing accuracy. Accordingly, the number of nodes is set as the same number of features for 1 layer ELM.

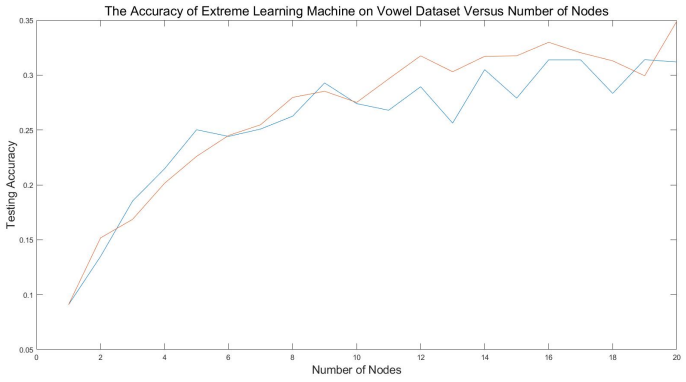

Figure 2: The Accuracy of ELM versus the number of nodes of 1layer ELM and 3-layer ELM. The blue line indicates the accuracy of 1 layer ELM while the red indicates that of 3 layer ELM.

Dataset	Test Accuracy(%)	Test NMSE	Training Time(s)
Vowel	31.92	-5.88	0.29
Extended YaleB	71.40	-35.82	0.63
AR	53.26	-96.75	0.65
Satimage	82.74	-4.78	0.32
Scene15	62.73	-13.16	23.39
Caltech101	31.58	-98.45	33.29

Table 5: Experimental results using 1-layer Extreme Learning Machine (ELM).

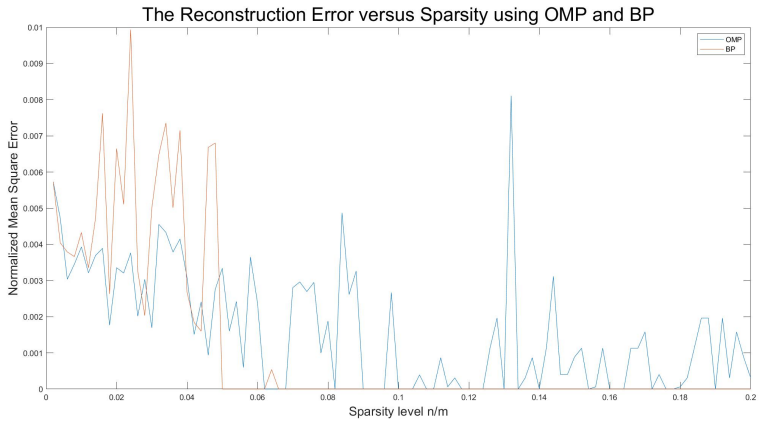

Figure 3: The Normalized Mean Square Error(NMSE) using OMP and BP versus varing sparsity

6 Assignment V: Sparse Representation

Orthogonal Matching Pursuit (OMP) an Subspace Pursuit(SP) aim at solving P_0 problem, which can be denoted as:

$$(P_0) : \arg \min_{x \in \mathbb{R}^m} \|x\|_0 \ \ subject\ to\ \mathbf{b} = \mathbf{A}\mathbf{x} \tag{38}$$

By smoothing penalty, we can solve the problem by Basis Pursuit (BP) using l_1 norm. Then the P_0 Problem can be reformulated as P_1 problem as:

$$(P_1) : \arg \min_{x \in \mathbb{R}^m} \|x\|_1 \ \ subject\ to\ \mathbf{b} = \mathbf{A}\mathbf{x} \tag{39}$$

As we all know, P_1 problem can reduce to linear program (LP) by introducing new variable $\mathbf{t}$

$$\begin{gathered} \min \mathbf{c^T t} \quad subject\ to\ \mathbf{Ix} - \mathbf{It} \leq \mathbf{0},\ \mathbf{Ix} + \mathbf{It} \geq \mathbf{0}\ and\ \mathbf{Ax} = \mathbf{b} \\ where\ \mathbf{c} = [1, 1, \ldots, 1]^T \end{gathered} \tag{40}$$

Hence, we can solve BP problem by using LP algorithm.

To evaluate these algorithms, we set $\mathbf{x} \in \mathcal{R}^m$ as a sparse vector with sparsity $k = \|x\|_0 << m$, and increase value of n to generate random matrix $\mathbf{A} \in \mathcal{R}^{n \times m}$. The sparsity k is set 5, the input dimension m is set 500 and the output dimension n is increased from 1 to 100. We set the tolerance for LP as $1e-4$.

As shown in Fig. 3, there is a sudden phase transition for BP when the value of ratio $\frac{n}{m}$ is larger than 5%. The BP algorithm consistently outperforms OMP in terms of reconstruction NMSE when the ratio is larger than the threshold. Although there is some spikes for both OMP and BP, the peak NMSE of spike is at an acceptable level.

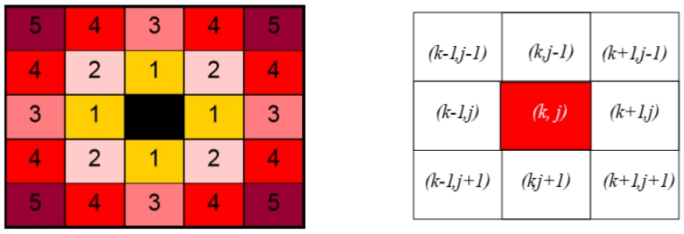

Figure 4: The neighborhood system (left) and 2_{nd} neighborhood of image (right).

7 Assignment VI: Image De-noising Using Markov Random Field

The neighborhood concept is fundamental in Markov Random Fields (MRF), a neighborhood system depends on the order of spatial dependencies. Here we adopt 2_{nd} clique C_2 as shown in Fig. 4.

Let $\mathbf{x}$ denotes the hidden image and $\mathbf{y}$ denotes the noisy image with additive white Gaussian noise ϵ which can be expressed as

$$\epsilon \sim \mathcal{N}(0, \sigma) \tag{41}$$

The energy function of each pixel is defined as

$$E(\mathbf{x}) = \sum_{i,j}(x_{i,j} - y_{i,j})^2 + \lambda \sum_{m,n \in C_1}(x_{m,n} - x_{i,j})^2 + \beta \sum_{m,n \in C_2}(x_{m,n} - x_{i,j})^2 \tag{42}$$

where the first term is the difference of observation and hidden value, second term and third term model the neighborhood pixel correlation respectively. λ and β controls the relative weight of 1_{st} clique and 2_{nd} clique where β is typically smaller. If we set the derivative of $E(\mathbf{x})$ with respect to $x_{i,j}$ to zero, we can get the solution

$$x_{i,j} = \frac{1}{1 + 4\lambda + 4\beta}\{y_{i,j} + \lambda \sum_{m,n \in C_1} x_{m,n} + \beta \sum_{m,n \in C_2} x_{m,n}\} \tag{43}$$

The Eq. 43 is much less computational demanding and like using a filter to do convolution with the image. Furthermore it is free of the problem of iterations or convergence. Alternatively we can directly set the pixel using this Eq. 43. The two above methods adopt quadratic potentials. However, the quadratic potentials are not robust to *outliers* and hence they will over-smooth edges. Instead we try robust *Lorentzian* error function $p(\cdot)$:

$$p(z, \sigma) = log(1 + \frac{1}{2}(\frac{z}{\sigma})^2) \tag{44}$$

Therefore the energy function can be expressed as

$$E(\mathbf{x}) = \sum_{i,j} p(x_{i,j} - y_{i,j}, \sigma_1) + \lambda \sum_{m,n \in C_1} p(x_{m,n} - x_{i,j}, \sigma_2) + \beta \sum_{m,n \in C_2} p(x_{m,n} - x_{i,j}, \sigma_2) \tag{45}$$

Figure 5: The comparison between different de-noising approaches. The top left is the original lena image; top middle is the noisy image; top right is result using MRF1 energy function in Eq. 42 ; bottom left is the output using MRF2 energy function in Eq. 43; bottom middle corresponds to robust potential function MRF3 setting in Eq. **??**; bottom right uses SP/MS algorithm. The noise variance is 0.01 and All the images are using original 8 bits.

where σ_1 can be related to noise variance, σ_2 can be related to the inter-pixel variance of the image. We compare the above three algorithms and present the result in Fig. 5. For energy function defined in Eq. 42 and Eq. 45, we adopt Iterated Condition Modes (ICM) for 10 iterations. The robust potential function encourages piecewise constant result and may fall into local minima.

Approaches	**SSIM**	**WPSNR**	**PSNR**	**NMSE**
MRF1 in Eq. 42	**0.69**	**86.63 dB**	75.81 dB	0.0017
MRF2 in Eq. 43	0.63	85.07 dB	**76.25 dB**	**0.0015**
MRF3 in Eq. 45	0.68	75.32 dB	69.28 dB	0.0077
SP/MS	0.53	84.13 dB	72.14 dB	0.0034
Noisy image	0.24	83.30dB	68.15 dB	0.0100

Table 6: Evaluation of each method compared with noisy image.

Table. 6 summarized our evaluation of three models using a comprehensive assessment system, namely Structural Similarity (SSIM) to measure the similarity between origin image and output, Peak Signal-to-Noise Ratio (PSNR), Weighted Peak Signal-to-Noise Ratio (WPSNR), and Normalized Mean Square Error. The SSIM index is calculated on various windows of an image. The measure between two windows x and y of common size $N \times N$ is:

$$SSIM(\mathbf{x}, \mathbf{y}) = \frac{(2\mu_x\mu_y + \mathbf{c}_1)(2\sigma_{xy} + \mathbf{c}_2)}{(\mu_x^2 + \mu_y^2 + \mathbf{c}_1)(\sigma_x^2 + \sigma_y^2 + \mathbf{c}_2)} \tag{46}$$

where $\mathbf{c_1} = (\mathbf{k_1 L})^\mathbf{2}$ and $\mathbf{c_2} = (\mathbf{k_2 L})^\mathbf{2}$ are two variables to stabilize the division with weak denominator.
For given MSE of an image, bit depth **BD**, width **W** and height **H** the corresponding PSNR is defined by

$$PSNR = 10\log_{10}(\frac{(2BD-1)^2}{MSE}) \tag{47}$$

Weighted PSNR is defined through the weighted MSE as:

$$WPSNR = 10\log_{10}(\frac{(2BD-1)^2}{MSE_w}) \tag{48}$$

8 Assignment VII: Image De-noising Using the SP/MS Algorithm

For the factor graph of image, each pixel is connected to the observed pixel and the neighboring pixels in both horizontal and vertical directions. All the connections are represented by a factor node. The message between node and factor can be represented as:

$$\begin{aligned} \mu_{f\to x} &= \sum_{x_1}\cdots\sum_{x_m} f(x, x_1, \ldots, x_m)\prod_m \mu_{x_m\to f}(x_m) \\ \mu_{x\to f} &= \prod_{l\in ne(x)/f} \mu_{f_l\to x}(x) \end{aligned} \tag{49}$$

The sum-product algorithm sometimes causes the problem of underflow, so the max-sum is proposed for numerical stability. The message propagation can be reformulated as

$$\begin{aligned} \mu_{f\to x} &= \max_{x_1,\ldots,x_m}[\log f(x, x_1, \ldots, x_m) + \sum_{m\in ne(f_s)/x} \mu_{x_m\to f}(x_m)] \\ \mu_{x\to f} &= \sum_{l\in ne(x)/f} \mu_{f_l\to x}(x) \end{aligned} \tag{50}$$

Then finding the maximum joint distribution of $\mathbf{x}$ and $\mathbf{y}$ is expressed as:

$$\begin{aligned} \max_{\mathbf{x}} \log p(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{x}} \sum_{i=1}^{W}\sum_{j=1}^{H} \phi(y_{i,j}, x_{i,j}) + \\ \sum_{i=1}^{W}\sum_{j=1}^{H} \log\psi(x_{i,j}, x_{i,j+1}) + \log\psi(x_{i,j}, x_{i+1,j})) \end{aligned} \tag{51}$$

where $\phi(\cdot)$ and $\psi(\cdot)$ represents the probability function between hidden and observed pixel, neighboring pixels respectively. Here we assume the image follows Gaussian distribution and we also add additive Gaussian white noise to the image. The functions are denoted as follows:

$$\begin{aligned} \phi(x, y) &= \mathcal{N}(x - y|0, \sigma_1) \\ \psi(x, y) &= \mathcal{N}(x - y|0, \sigma_2) \end{aligned} \tag{52}$$

where σ_1 is the deviation of noise we add and σ_2 is the inter-pixel standard deviation. We can backtrack and keep track of the maximization of each node.

$$\begin{aligned} \max_{\mathbf{x}} \log p(\mathbf{x}, \mathbf{y}) &= \max_{x_1}[\max_{x_2} \dots \log p(\mathbf{x}, \mathbf{y})] \\ \max_{x_i} \log p(\mathbf{x}, \mathbf{y}) &= \max_{x_i}[\sum_{s \in ne(x_i)} \mu_{f_s \to x_i}(x_i)] \end{aligned} \tag{53}$$

We process the same noisy image in MRF and present the result in Table. 6 and Fig. 5.

9 Assignment VIII: EM Mixture Model

In this case, we can view noisy image as Gaussian Mixture Model (GMM): Gaussian noise and pixels that follow Gaussian distribution. The original Gaussian mixture distribution can be denotes as

$$p(\mathbf{y}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{y}|\mathbf{u}_k, \mathbf{\Sigma_k}) \tag{54}$$

where the parameter π_k satisfies both $0 \leq \pi_k \leq 1$ and $\sum_1^K \pi_k = 1$. In the *Expectation* step, the responsibilities of each cluster are evaluated.

$$\begin{aligned} \gamma(z_{nk}) &= \frac{\pi_k \mathcal{N}(\mathbf{y}_n|\mathbf{u}_k, \mathbf{\Sigma}_k)}{\sum_{j=1}^{K} \mathcal{N}(\mathbf{y}_n|\mathbf{u}_j, \mathbf{\Sigma}_j)} \\ N_k &= \sum_{n=1}^{N} \gamma(z_{nk}) \\ \pi_k^{new} &= \frac{N_k}{N} \end{aligned} \tag{55}$$

Then we re-estimate the mean, variance and mixing coefficients in the *Maximization* step.

$$\begin{aligned} \mu_k^{new} &= \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{y}_n \\ \mathbf{\Sigma}_k &= \frac{1}{N_k} \sum_{n=1}^{N} (\mathbf{y}_n - \mu_k^{new})(\mathbf{y}_n - \mu_k^{new})^T \end{aligned} \tag{56}$$

We set $k = 1$ for image cluster and $k = 2$ for noise cluster. After iterations or convergence (μ for image converges), we can cluster the noisy image by using the responsibilities $\gamma(z_{n,k=1})$.

$$k(\mathbf{x}_n) = \begin{cases} 1 & \gamma(z_{n,k=1}) > \gamma(z_{n,k=2}) \\ 2 & \gamma(z_{n,k=1}) < \gamma(z_{n,k=2}) \end{cases} \tag{57}$$

The resulting noise cluster is displayed in Fig. 5 and noise pixel is indicated using white. As the variance of noise increases, the pattern is more clear to distinguish. After clustering, we can restore the missing image pixel by either MRF or SP/MS algorithm.

If we use MRF, pixels in the image cluster are viewed as hidden pixel while the noise pixel is taken as observable value and its hidden value is modelled by MRF. We present result using different MRF equations in Table. 7.

As for SP/MS algorithm, the image pixel is connected with neighboring pixel while noise pixel is further connected with hidden pixel. we set result variance of two clusters as the variance in probability function $\phi(\cdot)$ and $\psi(\cdot)$, then perform denoising targeting noise pixels.

Approaches	**SSIM**	**WPSNR**	**PSNR**	**NMSE**
EM+MRF1	0.85	76.71 dB	66.11 dB	0.0149
EM+MRF2	0.85	76.55 dB	66.09 dB	0.0160
EM+MRF3	0.85	76.25 dB	65.81 dB	0.0170
EM+SP/MS	**0.86**	**79.34 dB**	**67.89 dB**	**0.0132**
Noisy image	0.84	79.06 dB	64.96 dB	0.0207

Table 7: Evaluation of jointly denoising using EM and each method compared with noisy image.

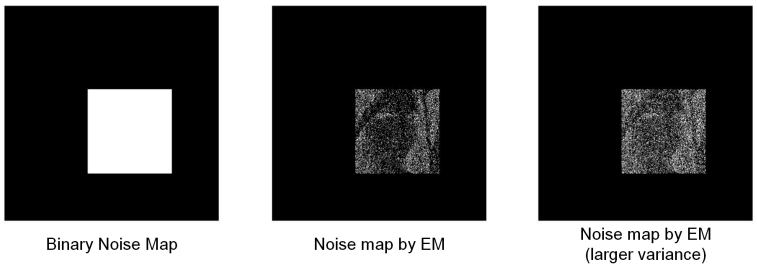

Figure 6: The EM clustering image and its combination with MRF and SP/MS algorithm.