Image Super-resolution with Sparse Image Representation using Dictionary Learning

Thesis to be submitted in partial fulfillment
of the requirements of the degree of

Master of Technology(Hons.)

in

Electronics & Electrical Communication Engineering

with specialization in

Visual Information and Embedded Systems

by

Bishal Santra
Roll No. 12EC35001

Under the supervision of

Prof. Priyadip Ray
Prof. Saswat Chakrabarti
(G S Sanyal School of Telecommunications)

Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology, Kharagpur
Kharagpur-721302, India

Contents

Abbreviation **v**

Symbols **vi**

Abstract **1**

1 Introduction **2**
1.1 Problem Description 2
1.1.1 What is Dictionary Learning? 2
1.1.2 What is Sparse Signal Representation/Compressive Sensing? 3
1.1.3 What is Super-Resolution? 3
1.2 Motivation 3
1.3 Organization of this Thesis 4

2 Literature Review **5**

3 Prior Distributions and Conjugate Pairs **7**
3.1 Gaussian Distribution 7
3.2 Multivariate Gaussian Distribution 7
3.3 Gamma Distribution 8
3.4 Bernoulli Distribution 9
3.5 Beta Distribution 9
3.6 Conjugate Priors 9
3.6.1 Normal-Gamma Conjugate Pair 9
3.6.2 Beta-Bernoulli Conjugate Pair 10

4 Bayesian Inference and Gibbs Sampling **12**
4.1 Bayesian Inference 12
4.2 Markov Chain Monte Carlo 12
4.3 Gibbs Sampling 12

5 Super Resolution using Sparse Image Representation **14**
5.1 Sparse Image Representation using Dictionary Learning 14
5.1.1 Graphical Models 14
5.1.2 Dictionary Learning for synthetic data 15
5.1.3 Dictionary Learning - Model I 19

5.1.4 Dictionary Learning using Beta-Bernoulli Process - Model 2 22
5.2 Super-Resolution using Sparse Image Representation 25
5.2.1 Super-Resolution: A Formal Definition 25
5.2.2 Objectives of Super-Resolution Method 26
5.2.3 Joint Dictionary Learning - Shared Latent Space Model . . . 26
5.2.4 Problem Formulation 27
5.2.5 Applying the model to RGB images 28

6 Experimental Results **30**
6.1 Sparse Image Representation using Dictionary Learning 30
6.1.1 Gibbs Sampling from a Joint Posterior Distribution 30
6.1.2 Preprocessing of Image Data 34
6.1.3 Dataset 35
6.1.4 Learning Features for Images 35
6.1.5 Output Plots 37
6.2 Super-Resolution using Sparse Image Representation 40
6.2.1 Training: Learning the Joint-Dictionaries 40
6.2.2 Testing 43
6.2.3 Super-Resolution Results 43

7 Conclusion **47**
7.1 Future Work 47
7.1.1 Deep Network using (Multilevel) Dictionary Learning 47
7.1.2 Multimodal Dictionary Learning 48

References **50**

List of Tables

6.1 Gibbs Sampling Example: Coefficient Estimation 31
6.2 Results: 2x Super-resolution 43
6.3 Results: 3x Super-resolution 45

List of Figures

3.1 A. PDF of Gaussian Distributed Random Variable, B. PDF of a bivariate normal distribution 8
3.2 Plot of PDF of Gamma Distribution 8
3.3 Marginal distribution of $\mathrm{P(X = x)}$ $(X, \gamma) \sim NormalGamma(\gamma, \alpha, \beta)$ 11

5.1 Graphical Model 1 : Simple Dictionary Learning 15
5.2 Graphical Model 2 : Dictionary Learning using Beta-Bernoulli Process 15

6.1 Performance of Bayesian Method in Polynomial Regression for different noise variance 33
6.2 Performance of Bayesian Method in Polynomial Regression for different noise variance 34
6.3 Generation of Training Data Matrix from Images 35
6.4 MSE in approximation as a function of Gibbs Iterations 37
6.5 MSE in approximation as a function of Gibbs Iterations 37
6.6 Reconstructed image from coefficients and features estimated by the algorithm 38
6.7 Image Reconstruction Example 1, A. Actual Image B. Reconstructed Image 38
6.8 Image Reconstruction Example 2, A. Actual Image B. Reconstructed Image 39
6.9 Image Reconstruction Example 3, A. Actual Image B. Reconstructed Image 39
6.10 Reconstruction of Images using single dictionary elements 39
6.11 Reconstruction of Images using single dictionary elements 40
6.12 Dictionary Atoms learned by the algorithm 40
6.13 Reconstruction of Images using single dictionary elements 40
6.14 Jointly Learned Dictionaries 42
6.15 Learned Probabilities for Joint Dictionary Atoms 42
6.16 Super-Resolution(2x magnification) Examples 43
6.17 Super-Resolution(2x magnification) Examples 44
6.18 Super-Resolution(2x magnification) Examples 44
6.19 Super-Resolution(3x magnification) Examples 45
6.20 Super-Resolution(3x magnification) Examples 46
6.21 Super-Resolution(3x magnification) Examples 46

7.1 Multilayer Dictionary Learning Model 48

Abbreviation

DL - Dictionary Learning

LR - Low Resolution

HR - High Resolution

SampleMVN - Sample from Multi-Variate Normal Distribution

SampleBeta - Sample from Beta Distribution

SampleBernoulli - Sample from Bernoulli Distribution

SampleGamma - Sample from Gamma Distribution

Symbols

D - Dictionary Matrix

S - Weight Matrix

B - Selection Matrix

η - Bias

ϵ - Gaussian Noise

γ - Precision Parameter for Gaussian Distribution

α - Shape Parameter of Gamma Distribution

β - Shape Parameter of Gamma Distribution

$*_h$ - Parameter related to model of high resolution image

$*_l$ - Parameter related to model of low resolution image

Abstract

Superresolution imaging techniques are used for enhancing resolution of images. It has applications in various domains e.g. enhancement of images from survellience cameras, medical diagnosis, astronomical observations, satelite images etc. In this thesis we present a dictionary learning based approach for superresolution using sparse image representation. By using pairs of low resolution(LR) and high resolution(HR) images we first learn joint dictioanries for LR and HR images in a way that the sparse representation is shared between them. We call this learning a "Shared Latent Space" for related data. The model is learnt on patched image data thus making it more robust against any kind of textures. To verify this we have tested our model on different kinds of images (natural images, facial images, etc.) and got equally good results in all the cases.

Keywords: Dictionary Learning, Gibbs Sampling, Sparse Representation, Superresolution, Shared Latent Space, Bayesian Inference, Beta-Bernoulli Process, Normal-Gamma Process

Chapter 1

Introduction

1.1 Problem Description

There has been a significant increase of interest in Bayesian inference aligned with the fact that in the last two decades we have seen a large growth in availability of computing memory and processing power. Because of this MCMC simulations, Gibbs sampling methods are now easily realizable. Many advancements have been done in image processing domain regarding the use of Deep Neural Networks, RBMs, CNNs etc. All these models eventually learns some hidden representation of the training dataset. These models are hierarchical / multilayer in nature in the sense that they build features of different scale and abstraction in a bottom up fashion. For each layer the feature activation map of the last layer is the input to the next layer. All these features has proven to be very accomplished in tasks like classification / prediction problems.

Now the model proposed by us is based on the concept of Sparse (and over-complete) Dictionary Learning. The outputs (sparse representations for images) from our model are particularly similar to the outputs from first layer of a convolutional neural network. Since we used patches generated by sliding window the model thus generated are also shift equivariant which means that our model captures any feature irrespective of its location in the image (just like CNNs).

For the super-resolution task we use the sparse representation of the LR image to reconstruct the HR image with help of a parallely trained HR dictionary. We learn this HR dictionary by sharing the latent space for LR and HR images' sparse representation learning during training phase.

1.1.1 What is Dictionary Learning?

Dictionary Learning is a signal representation method where the dictionary, a set of predefined/learned basis vectors, is used to represent a given signal y as a linear

combination of those fixed set of basis vectors. These basis vectors are also called atoms. Usually in it's most basic form it is expressed as,

$$y = \sum_{k=1}^{K} s_k \Phi_k + \epsilon \tag{1.1}$$

where, $\Phi = \{\Phi_k | k = 1, 2, ..., K\}$ is the dictionary and Φ_k's are called the atoms. s_k is the weight value for k^{th} atom and ϵ is the approximation error.

1.1.2 What is Sparse Signal Representation/Compressive Sensing?

A sparse signal representation is representing a signal with sparse vectors in some traformed vector space. In the context of dictionary learning the learned representation is called sparse when most of the values in weight matrix, S are nearly (or exactly) zero. This is also known as Sparse Approximation problem. There exists different algorithms for solving this sparse approximation problem most of them being optimization based. We, in this thesis, explore a probabilistic model of sparse dictionary learning and use Bayeian Inference to solve the problem of super-resolution.

1.1.3 What is Super-Resolution?

Super-resolution is a technique for reconstructing a high resolution image from one or more low resolution image(s)[1]. In literature there are two types of super resolution algorithms single-image and multi-image based. Even though multi-image based methods produces better results due to availability of more information about the neighborhood of the missing pixel but multiple images may not be always available. In this thesis we propose a method for single-image super-resolution.

1.2 Motivation

In many scenarios an HR imaging device may not be available due to price, size of device or other factors. But in many applications like medical imaging of internal organs or obtaining HR images of astronomical objects detailed images are very much desired and important. Even though there have been quite a few approaches to solve this problem of super-resolution, most of the methods only work for a particular type of images e.g. images with mostly high frequency content(multicolored textures) or low frequency content as in facial images. To tackle this problem we want explore Bayesian method of solving this problem.

1.3 Organization of this Thesis

The rest of the thesis is organized as follows. In 2 we discuss a few existing works on sparse image representation and super-resolution. We review the concept of priors and conjugate distributions in 3 which is a crucial component of building the Gibbs Samplers used in this thesis. 4.3 introduces Bayesian Inference and the Gibbs Learning algorithm. 5.2 formalizes the super-resolution task and describe about the preliminaries regarding the proposed approach. 5.1.4 details about the non-parametric bayesian approach for solving the dictionary learning problem. In 5.2.3, we extend the sparse image representation model and discuss how we build the Probabilistic Graphical Model for the single image super-resolution problem. In 6 we experiment the performance of our model under various parameters and settings.

Chapter 2

Literature Review

An introduction and working of Markov Chain Monte Carlo (MCMC) methods are given in [2]. It gives several examples in order to explain how to do inference based on a Bayesian model. Further for a Naive Bayes model for text document labelling (as in NLP) where an analytical expression for the pdf of posterior distribution isn't available it shows how to derive a Gibbs sampler for generating random samples from such a distribution. Importance of selection of proper priors and conjugate pairs are also well explained in this article.

Various optimization methods for solving Sparse Dictionary Learning problems are summarized in [3]. Comprehensive analysis of Method of Optimized Directions / MOD also know as Iterative Least squares dictionary learning algorithm (ILS-DLA), KSVD and other standard methods are documented in this paper. For ILS-DLA the name itself explains the reason behind such nomenclature. It considers the well known least square solution to the approximation problem. KSVD in each iteration alternatively update the dictionary atoms and the weight matrix.

An alternative method for imposing sparsity using Beta process priors for a Bayesian Dictionary Learning model is proposed in [4]. The generative model suggested in this paper takes the form $x_i = \Phi(z_i \cdot w_i) + \epsilon_i$. z_i is an indicator matrix denoting whether a certain atom in dictionary Φ will participate in reconstruction of i^{th} sample vector. z_i is modeled as N draws of Bernoulli process with success probability π drawn from a Beta distribution. Our final Dictionary Learning model is a slight variation of the factor analysis model proposed in this paper.

Another similar model which tries to mimic the behavior of a convolutional neural network or CNN by replacing the linear combination step by a convolution step is shown in [5]. Here in the Bayesian inference step what is learnt are small spatially localized canonical elements as the atoms, d_i, and a spatial activation map, W, which through the convolution represents the contribution of an atom in each small block of the image. Results similar to this are obtained when we considered patched version of images by sliding window method for training data.

The models presented in [6, 7, 8] along with the one that we are going to use in this project are fully unsupervised in nature. [9] shows that the features learned using these algorithms are actually visually important in case of image processing. They resembles standard features used in image processing like edges, primitive shapes, objects etc. As shown further in [10] stacked versions of CNN / Boltzman Machine can actually learn features with much higher level of abstraction. When such networks from different modalities are connected and learned together these unsupervised algorithms learn new cross-modality features.

There are quite a few works on super-resolution using Dictionary Learning. One similar approach for super-resolution using dictionary learning has been taken in [11], where optimization based methods are used for learning the dictionary. Other approaches towards single image super-resolution includes [12], where the authors have used convolutional neural networks to generate a fitting texture from the low resolution image. Though this model sometime fails when there is a sudden change in the textures in the image. Another notable work is [13], where the authors, in their model, estimates different distortion parameters for the LR image, eg. translation, rotation etc. They model these parameters as random variables and solves the estimation problem using variation approximation. Although the work in [13] infers HR image from a set of degraded LR images. [14] uses the state of the art Machine Learning models like Deep CNN for this task. They have also shown in their paper how sparse coding methods can be viewed as Deep convolutional networks.

Chapter 3

Prior Distributions and Conjugate Pairs

We discuss below a few standard distributions and the kind of prior belief on parameters for which they have been used for.

3.1 Gaussian Distribution

If x is a Gaussian distributed random variable with mean and variance μ and σ^2, respectively, then probability density function of x is given by,

$$f(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-\frac{1}{2\sigma^2}(x-\mu)^2) \tag{3.1}$$

A Gaussian Distributed random variable, x, can take any value between $(-\infty, \infty)$. By changing the variance, σ^2, we can control the how probable it is for x to take values closer to the mean, μ. A high variance would mean flat prior or a non-informative prior.

3.2 Multivariate Gaussian Distribution

Multivariate Gaussian Distribution is an generalization of univariate Gaussian distribution to higher dimensions.
If X is a Multivariate Gaussian distributed **random vector** with mean-vector and covariance matrix μ and Σ, respectively, then probability density function of X is given by,

$$P(X|\mu, \Sigma) = \frac{1}{\sqrt{2\pi|\Sigma|}} \exp(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)) \tag{3.2}$$

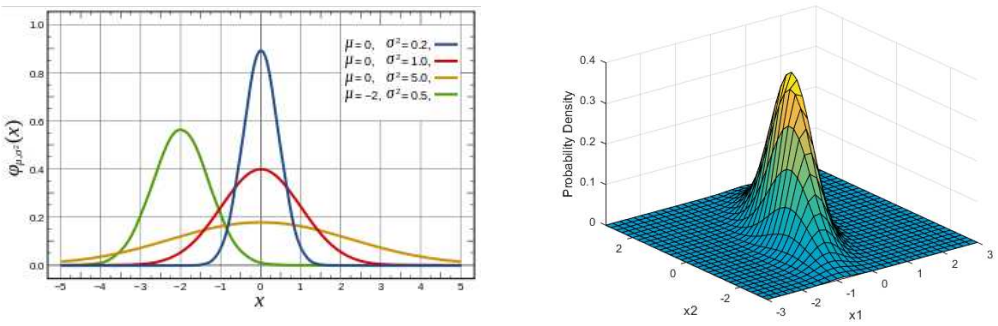


Figure 3.1: A. PDF of Gaussian Distributed Random Variable, B. PDF of a bivariate normal distribution

3.3 Gamma Distribution

PDF of a random variable, r, following Gamma Distribution, is

$$f(r|\alpha,\beta) = \frac{\beta^{\alpha} x^{\alpha-1} e^{-x\beta}}{\Gamma(\alpha)} \tag{3.3}$$

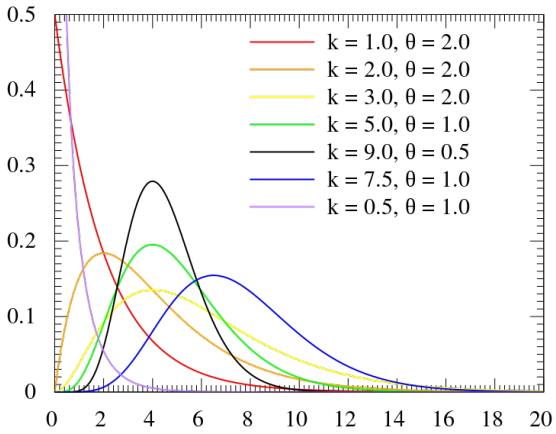


Figure 3.2: Plot of PDF of Gamma Distribution

A Gamma distributed random variable, x, can take value in the range $[0, \infty)$. Thus this distribution is suitable for modelling random variables that take only non-negative values e.g. variance of a distribution.

3.4 Bernoulli Distribution

A discrete random variable X is called Bernoulli Distributed if it takes on value 1 with probability p, and the value 0 with probability (1-p).

$$Pr(X = 1) = \pi \tag{3.4}$$
$$Pr(X = 0) = 1 - \pi \tag{3.5}$$
$$\tag{3.6}$$

The Probablity Mass Function can also be expressed as,

$$P(X = k) = (1 - \pi)^{1-k}\pi^k \tag{3.7}$$
$$where,\ k \in \{0, 1\} \tag{3.8}$$

3.5 Beta Distribution

PDF of a random variable, b, following Beta Distribution, is

$$P(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1}(1 - x)^{\beta-1} \tag{3.9}$$
$$where,\ B(\alpha, \beta) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha + \beta)} \tag{3.10}$$

A Beta distributed random variable, x, takes values only between $[0, 1]$. Parameters like probability p of a Bernoulli distributed random variable can be modelled using this parameter.

3.6 Conjugate Priors

A very important concept that we would use quite frequently in the rest of this thesis is **conjugate Priors**. In Bayesian Probability theory, if the posterior distribution, $P(\theta|X)$, is in the same family as the prior probability distribution, $P(\theta)$, then the prior and the posterior are called **conjugate distributions** and the prior is called **conjugate prior** for the likelihood function.
Notably, all the distributions in exponential family have conjugate priors. Following are the conjugate pairs used for deriving the dictionary learning model used in this project.

3.6.1 Normal-Gamma Conjugate Pair

Gamma distribution is the conjugate prior of Normal distribution with unknown precision (if we choose variance to be the hyperparameter in that case conjugate prior would be inverse-gamma distribution).

$$X|\gamma \sim \mathcal{N}(0, 1/\gamma) \tag{3.11}$$

$$\gamma|\alpha, \beta \sim Gamma(\alpha, \beta) \tag{3.12}$$

Also the joint distribution, (X, γ), is then a Normal-Gamma distribution,

$$(X, \gamma) \sim NormalGamma(\gamma, \alpha, \beta) \tag{3.13}$$

For various values of α & β, the nature of the marginalized distribution P(X = x) is shown in the following figures. Note depending on the values of α & β the size of the main lobe and the length of the tails can be controlled. This property is used to obtain sparseness in one of our models.

Following plots in 3.3 were generated in Matlab as histograms.

3.6.2 Beta-Bernoulli Conjugate Pair

Suppose X is a discrete random variable and can take on values 0 or 1. The porbability, Pr(X = 1) = p, which is unknown and in turn modeled as another random variable. If p follows beta distribution with parameters a, b then (X, p) is called a Beta-Bernoulli process.

$$P(P = p) = \frac{p^{a-1}(1-p)^{b-1}}{B(a, b)} \tag{3.14}$$

$$P(X = 1|P = p) = p; \tag{3.15}$$

By the property of conjugate priors, the posterior distribution, $P(P = p|X = x)$ follows Beta distribution also. N such beta-bernoulli process together forms a beta-binomial process whose support is the set of non-negative integers upto n.

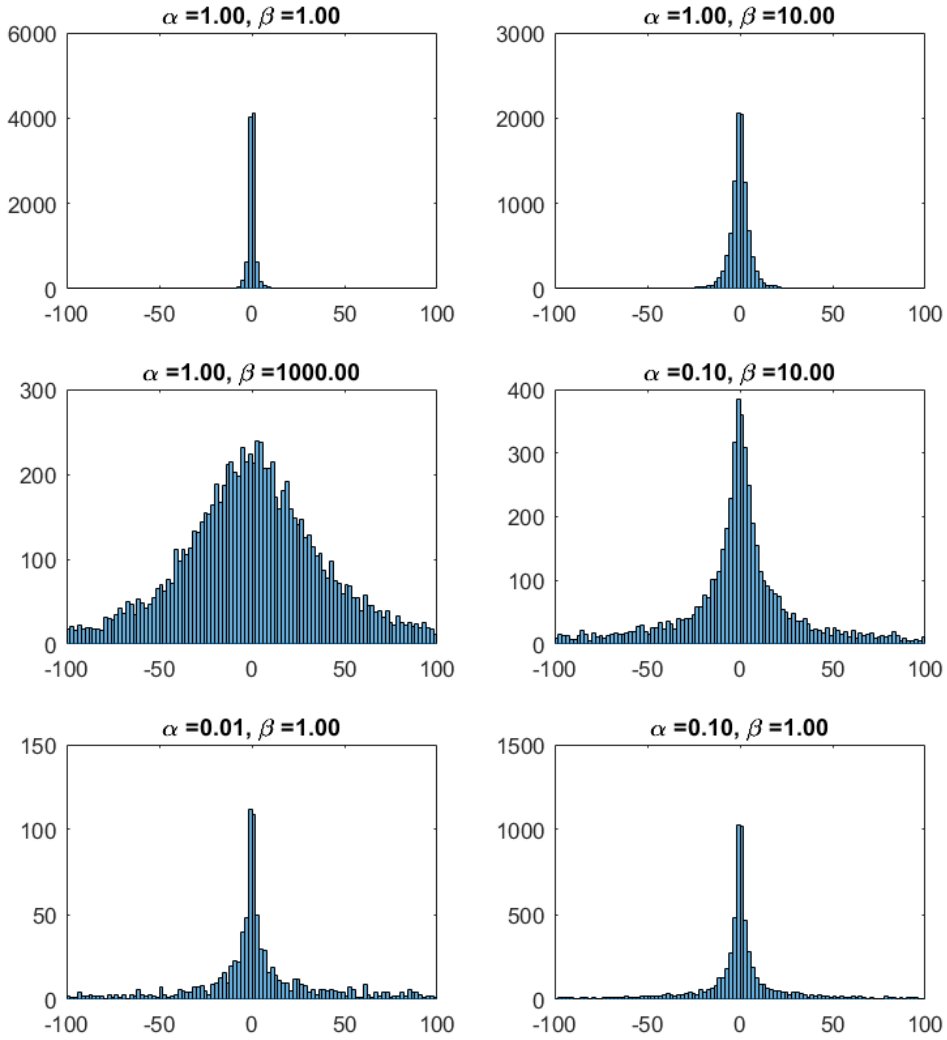


Figure 3.3: Marginal distribution of P(X = x)
$(X, \gamma) \sim NormalGamma(\gamma, \alpha, \beta)$

Chapter 4

Bayesian Inference and Gibbs Sampling

4.1 Bayesian Inference

Bayesian inference is a method of statistical inference based on Bayes theorem. As more and more evidence or information becomes available we update our hypothesis accordingly using the Bayes theorem.

$$P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$$

where, X and Y are events and $P(X) \neq 0$.

4.2 Markov Chain Monte Carlo

A class of methods known as Markov Chain Monte Carlo or MCMC are used to sample from a desired distribution for which direct sampling sampling isn't possible. It does so by constructing a Markov chain whose steady state is the desired distribution.
One such MCMC method, **Gibbs Sampling** is described below.

4.3 Gibbs Sampling

Gibbs sampling is one such Markov chain Monte Carlo algorithm for sampling a sequence from a joint probability distribution. Though for one to be able to use Gibbs Sampling all the conditional distributions has to fully defined preferebly in the form of standard probability distributions. This makes sampling from the desired distribution, efficiently, using a computer program easier.
As we saw in the polynomial regression example, since the multivariate posterior distribution doesn't follow any standard probability distribution direct sampling is not possible. This is where Gibbs sampling comes into picture.

For sampling from the joint posterior distribution in Gibbs sampling method we instead sample from the conditional posterior distribution of the system hyper-parameters sequentially which in steady state approximately resembles the actual joint distribution.
Mathematically, from a set of collected data samples we want to estimate the hyperparameters, θ, of a proposed model. θ is the set of all the hyperparameters involved in our model. So, when $P(\theta)$ is written it means the joint distribution of (independent subsets of) hyperparameters, $P(\theta_1, \theta_2, \ldots, \theta_k)$[1].

$$\begin{aligned}
\theta &= \{\theta_1, \theta_2, \ldots, \theta_k\} \\
\theta_{\sim j} &= \theta - \theta_j \\
P(\theta|X) &= P(X|\theta_1, \theta_2, \ldots, \theta_k)P(\theta_1, \theta_2, \ldots, \theta_k)/P(X) \\
P(\theta|X) &\propto P(X|\theta_1, \theta_2, \ldots, \theta_k)P(\theta_1, \theta_2, \ldots, \theta_k) \\
P(\theta_j|\theta_{\sim j}, X) &= P(\theta|X)/P(\theta_{\sim j}) \\
&\propto P(\theta|X) \\
&= P(X|\theta_1, \theta_2, \ldots, \theta_k)P(\theta_1, \theta_2, \ldots, \theta_k)
\end{aligned}$$

The RHS of the proportionality relation has two terms in it, namely, likelihood of the data multiplied by the joint prior probability of the system hyperparameters. Next we make use of the assumption of independence of the hyperparameters. So, the expressions above can be simplified even further.

$$\begin{aligned}
P(\theta_j|\theta_{\sim j}, X) &\propto P(X|\theta_1, \theta_2, \ldots, \theta_k)P(\theta_1, \theta_2, \ldots, \theta_k) \\
&= P(X|\theta_1, \theta_2, \ldots, \theta_k)P(\theta_1)P(\theta_2)\ldots P(\theta_k) \\
&\propto P(X|\theta_1, \theta_2, \ldots, \theta_k)P(\theta_j)
\end{aligned}$$

In the Gibbs sampling algorithm we are required to sample from these conditional expressions, $P(\theta_j|\theta_{\sim j}, X)$. Now we can use a conjugate pair for the likelihood and the prior. This way the type of distribution of $P(\theta_j|\theta_{\sim j}, X)$ will be directly determined as that of the prior.

[1]Note that θ_i doesn't have to be a scalar parameter it can be a vector of size more than one too.

Chapter 5

Super Resolution using Sparse Image Representation

5.1 Sparse Image Representation using Dictionary Learning

Dictionary Learning is a family of algorithms in the domain of Signal Processing, which can be used to obtain sparse representation of signals. Dictionary is a collection of basis vectors also known as atoms. Linear combination of these basis vectors are used to represent a particular signal observation. These atoms are different for different kind of applications. A sparse representation of a vector X is usually denoted by,

$$X_a = DS \tag{5.1}$$

where, X_a is the approximated representation of the vector X as a weighted linear combination of columns of D.

5.1.1 Graphical Models

This is a graph used to represent the direct conditional dependencies between the random variables and the hypperparameters. The nodes in the graph represents the random variables and hyperparameters involved in the model. The directed edges represents (one way) dependency between the nodes. In the following illustrations for graphical models, the nodes with dark borders are the parameters of the model, and rest are the hyperparameters.

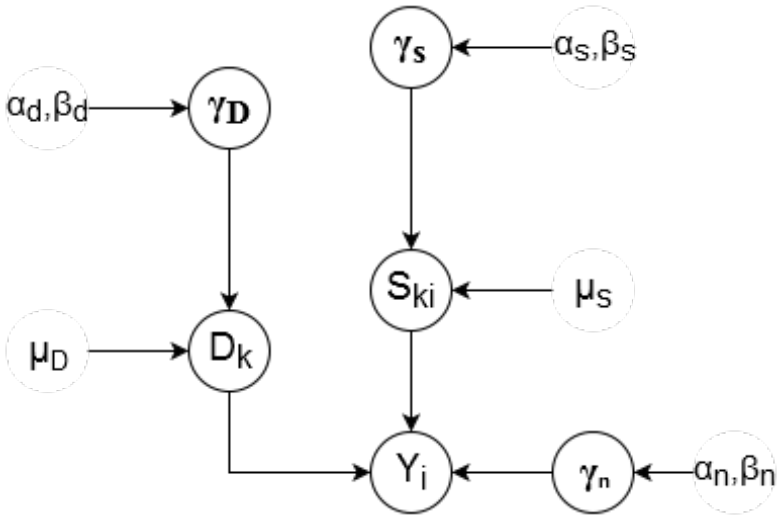


Figure 5.1: Graphical Model 1 : Simple Dictionary Learning

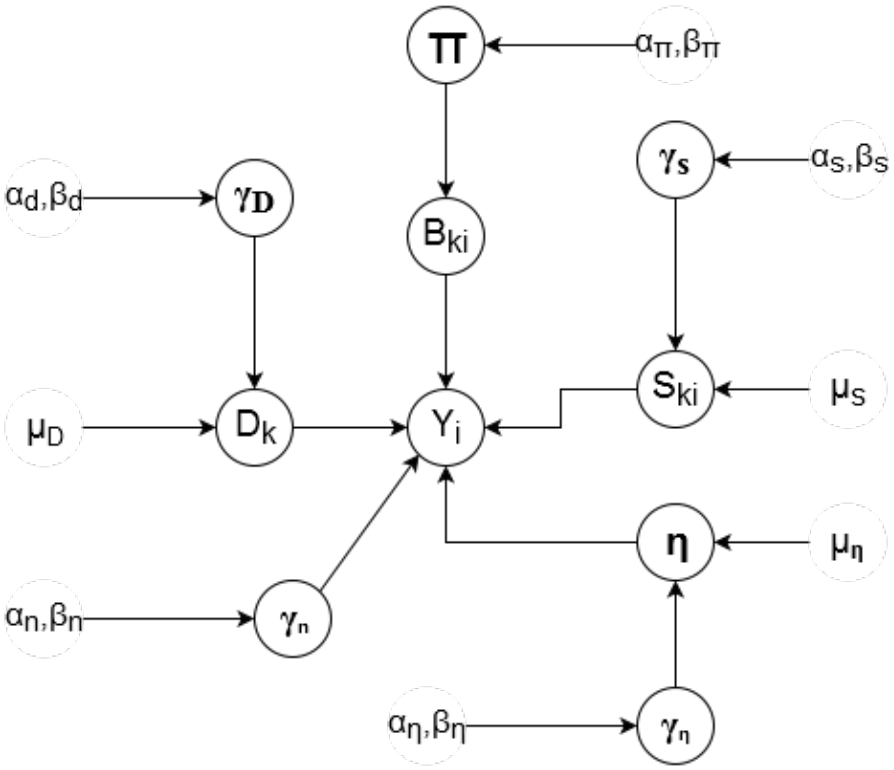


Figure 5.2: Graphical Model 2 : Dictionary Learning using Beta-Bernoulli Process

Both of the models above are explained in full details in section 5.1.3 and 5.1.4.

5.1.2 Dictionary Learning for synthetic data

Polynomial Regression using Bayesian Method

We picked the problem of Polynomial Regression first to illustrate and validate the working of the Gibbs sampling implementation for the Dictionary Learning Model.

Consider the following polynomial,

$$y_0 = a_3x^3 + a_2x^2 + a_1x + a_0 \tag{5.2}$$

Suppose we are receiving samples from above system but the output is affected by Gaussian noise, i.e. the data that we have are of the form,

$$y = y_0 + \epsilon \tag{5.3}$$

where, ϵ denotes additive Gaussian noise.
Say, we have N pairs of input and noisy output data with us. We want to estimate the scalar coefficients, $\{a_i | i = 0, 1, ..., 3\}$ of the system. Let's denote the data as,

$$\begin{aligned}
Y &= [y_0, y_1, \cdots, y_{N-1}]' \\
x &= [x_0, x_1, \cdots, x_{N-1}]' \\
a &= [a_0, a_1, a_2, a_3]' \\
n &= [\epsilon_0, \epsilon_1, \ldots, \epsilon_{N-1}]' \\
X &= \begin{bmatrix} 1 & x_0 & x_0^2 & x_0^3 \\ 1 & x_1 & x_1^2 & x_1^3 \\ \vdots & \vdots & \vdots & \vdots \\ 1 & x_{N-1} & x_{N-1}^2 & x_{N-1}^3 \end{bmatrix} \\
Y &= Xa + n
\end{aligned} \tag{5.4}$$

Algebraic Solution:

The solution of the above problem obtained by minimizing the squared error with respect to the polynomial coefficients, a, is given by,

$$\begin{aligned}
\hat{a} &= argmin_a \|Y - Xa\|^2 \\
&= (X^TX)^{-1}X^TY
\end{aligned} \tag{5.5}$$

Bayesian Method:

Now let's see the Bayesian Method for solving the same problem,

1. Let us first solve a simpler version of the problem. The prior distribution for the polynomial coefficients, a, and the random noise, n[1], are as defined below. Assume

[1]Lowercase 'n' is used for denoting noise, whereas, uppercase 'N' is the number of data

that σ^2 (noise variance) and γ (precision of the prior Gaussian Distribution for a) are known to us.

$$\begin{aligned} a_i &\propto \mathcal{N}(0, \gamma^{-1} I) \\ \forall\ i &\in 0, 1, 2, 3 \\ n &\propto \mathcal{N}(0, \sigma^2 I) \end{aligned} \tag{5.6}$$

Assuming all a_i's are independent,

$$P(a) = \left(\frac{\gamma}{\sqrt{2\pi}}\right)^4 \exp(-\frac{\gamma}{2} a^T a) \tag{5.7}$$

The likelihood of the observed data is given by,

$$P(Y|a, X) = \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^N \exp\left(-\frac{1}{2\sigma^2}(Y - Xa)^T (Y - Xa)\right) \tag{5.8}$$

Now, the expression for the PDF of posterior $P(a|Y, X)$ using Bayes rule is given by,

$$\begin{aligned} P(a|Y, X) &= \frac{P(Y|a, X)P(a)}{P(Y)} \\ &\propto P(Y|a, X)P(a) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(Y - Xa)^T (Y - Xa)\right) \exp(-\frac{\gamma}{2} a^T a) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}[-2Y^T Xa + a^T (X^T X + \gamma\sigma^2 I)a]\right) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(a - M^{-1} X^T Y)^T (a - M^{-1} X^T Y)\right) \end{aligned} \tag{5.9}$$

where, $M = (X^T X + \gamma\sigma^2 I)$. Note that the terms which doesn't depend on a has been removed to obtain a proportionality relation to keep the derivation short.

As we can see the posterior distribution of the polynomial coefficients a follows a Multivariate Gaussian Distribution with mean and co-variance matrix,

$$\begin{aligned} \mu_a &= (X^T X + \gamma\sigma^2 I) X^T Y \\ \Sigma_a &= \sigma^2 (X^T X + \gamma\sigma^2 I)^{-1} \end{aligned} \tag{5.10}$$

samples

2. In part 1 we saw that the posterior distribution of coefficients, a, follows multivariate normal distribution. Since we had only one model parameter a we can directly use μ_a (it's the MMSE estimate of a) as the estimator for a. But the drawback of above method is that we have to have prior knowledge about noise variance and precision of prior distribution of a.

In this section we would see how to model even the *hyperparameters*, in this case *noise variance*(σ^2) and precision of prior distribution of $a(\gamma)$, as random variables.

The prior distributions for the hyperparameters are as follows,

$$\begin{aligned} a|\gamma_0 &\sim \mathcal{N}(a|0, \gamma_0^{-1} I_N) \\ \gamma_0 &\sim Gamma(\alpha_0, \beta_0) \\ n|\gamma_n &\sim \mathcal{N}(0, \gamma_n^{-1} I_N) \\ \gamma_n &\sim Gamma(\alpha_n, \beta_n) \end{aligned} \tag{5.11}$$

The PDFs for the above hyperparameters,

$$\begin{aligned} P(a|\gamma_0) &= \left(\frac{\gamma_0}{2\pi}\right)^{3/2} \exp(-\frac{\gamma_0}{2} a^T a) \\ P(\gamma_0) &= \frac{\beta_0^{\alpha_0}}{\Gamma(\alpha_0)} \gamma_0^{\alpha_0} \exp(-\beta_0 \gamma_0) \\ P(\gamma_n) &= \frac{\beta_n^{\alpha_n}}{\Gamma(\alpha_n)} \gamma_n^{\alpha_n} \exp(-\beta_n \gamma_n) \end{aligned} \tag{5.12}$$

And the likelihood of the data matrix,

$$P(Y|a, \gamma_n, X) = \frac{\gamma_n}{2\pi}^{N/2} \exp(-\frac{\gamma_n}{2}(Y - Xa)^T (Y - Xa)) \tag{5.13}$$

Now with the help of Bayes Rule, the expression of the joint posterior distribution is given by,

$$\begin{aligned} &P(a, \gamma_0, \gamma_n|Y, X) \propto P(Y|a, \gamma_0, \gamma_n) P(a|\gamma_0) P(\gamma_0) P(\gamma_n) \\ &\propto (\gamma_n)^{N/2} \exp\left(-\frac{\gamma_n}{2}\|Y - Xa\|^2\right) \gamma_0^{3/2} \exp(-\frac{\gamma_0}{2}\|a\|^2) \gamma_0^{\alpha_0 - 1} \exp(-\beta_0 \gamma_0) \gamma_n^{\alpha_n - 1} \exp(-\beta_n \gamma_n) \end{aligned} \tag{5.14}$$

The individual conditional expression for Gibbs sampling of a, γ_n *and* γ_0 given everything else is constant (denoted as $P(\cdot|_\sim)$) are as follows,

$$
\begin{aligned}
P(a|_\sim) &\propto \exp(-\frac{\gamma_n}{2}\|Y - Xa\|^2)\exp(-\frac{\gamma_0}{2}\|a\|^2) \\
&\propto \mathcal{N}\left((X^TX + \gamma\sigma^2 I)X^TY, \sigma^2(X^TX + \gamma\sigma^2 I)^{-1}\right)
\end{aligned} \tag{5.15}
$$

$$
\begin{aligned}
P(\gamma_0|_\sim) &\propto \gamma_0^{3/2}\exp(-\frac{\gamma_0}{2}\|a\|^2)\gamma_0^{\alpha_0 - 1}\exp(-\beta_0\gamma_0) \\
&\propto Gamma\left(\gamma_0|\alpha_0 + 3/2, \beta_0 + \frac{\|a\|^2}{2}\right)
\end{aligned} \tag{5.16}
$$

$$
\begin{aligned}
P(\gamma_n|_\sim) &\propto \gamma_n^{N/2}\exp(-\frac{\gamma_n}{2}\|Y - Xa\|^2)\gamma_n^{\alpha_n - 1}\exp(-\beta_n\gamma_n) \\
&\propto Gamma\left(\gamma_n|\alpha_n + N/2, \beta_n + \frac{\|Y - Xa\|^2}{2}\right)
\end{aligned} \tag{5.17}
$$

5.1.3 Dictionary Learning - Model I

The most basic mathematical representation of a Dictionary Learning problem is given by, $Y = DS + \epsilon$, where the NXK matrix D consists of K basis vectors, which may or may not be known to us in prior. If, we have a D matrix with us, this problem can easily be solved for S, by minimizing the squared error between Y and DS. Using Lagranges Multiplier method the solution is found to be,

$$
\begin{aligned}
\hat{S} &= D^\dagger Y \\
D^\dagger &= (D^TD)^{-1}D^T
\end{aligned} \tag{5.18}
$$

where, $D^\dagger$ is also called the **Pseudo-Inverse** of matrix D.
But in the more generalized form of this problem we only have the dataset of observations Y. From there itself we want to learn the dictionary D. There are quite a few algorithms for learning both the dictionaries and the appropriate representation matrix or the weight matrix S. There are iterative methods for solving this problem, where some cost function D and S are updated to obtain a local or global minima. e.g. MOD(Methods of Direction), K-SVD, Matching Pursuit etc. Here we would discuss the **Bayesian Method** for estimating the matrices D and S.

In Bayesian Method, we first try to find out the joint posterior distribution, $P(\theta|Y)$, where, θ is the complete set of system parameters / hyperparameters (discussed in the following paragraph). Later we can take the mean(also known as MMSE estimate), median or the mode of this distribution as the estimate for the system parameters. As discussed earlier in section 4.3 that an analytical solution

to $P(\theta|Y)$ for such systems is intractable. We use Gibbs sampling to draw samples from the posterior distribution.

The first step to solving this problem is to choose a likelihood function and assign priors to the model parameters. We assume a the noise present is Additive White Gaussian Noise or AWGN, with a inverse variance $\gamma_n (= \frac{1}{\sigma^2})$. Hence the likelihood function is,

$$P(Y|D,S) = \mathcal{N}(Y|DS, \frac{1}{\gamma_n}) \tag{5.19}$$

Up next we need to assign priors for D and S. The support for the elements d_{ik} *and* s_{ki} in matrices D and S, respectively, are complete real line in both the cases. i.e. $-\infty \leq d_{ik} \leq \infty$ *and* $-\infty \leq s_{ki} \leq \infty$. We also want to impose sparsity on the matrices to help obtain a sparse representation of the data. If we choose the Normal-Gamma conjugate pair would be appropriate for this case as it satisfies the first condition and as shown in 3.6.1 we can control the sparsity by changing the parameters α and β.

For the noise variance we choose a Gamma prior with parameters α_n *and* β_n.

Likelihood and Prior Distributions:

$$
\begin{aligned}
y_i &= \sum_{k=1}^{K} d_k s_{ki} + \epsilon_i, \ \forall\ i \in 1, \cdots, N \\
Y_{CXN} &= D_{(CXK)} S_{(KXN)} + \epsilon_{(CXN)} \\
D &= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ d_1 & \cdot & \cdot & \cdot & d_K \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \\
S &= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ s_1 & \cdot & \cdot & \cdot & s_N \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \\
B &= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ b_1 & \cdot & \cdot & \cdot & b_N \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}
\end{aligned} \tag{5.20}
$$

$$
\begin{aligned}
d_{ij} &\sim \mathcal{N}(0, \gamma_d^{-1}) \\
\gamma_d &\sim Gamma(\alpha_d, \beta_d) \\
s_{ij} &\sim \mathbf{N}(0, \gamma_s^{-1}) \\
\gamma_s &\sim Gamma(\alpha_s, \beta_s) \\
\epsilon &\sim \mathbf{N}(0, \gamma_n^{-1})
\end{aligned} \tag{5.21}
$$

Now we would present[2] the conditional distributions for the parameters given the data for the model parameters/hyperparameters.

For sampling d_{uv} :

$$
\begin{aligned}
P(d_{uv}|_{\sim}) &= \mathcal{N}\left(d_{uv}|\mu'_{duv}, \gamma'_{duv}\right) \\
where,\ \mu'_{duv} &= \frac{\sum_{j=1}^{N} \gamma_n s_{vj}(y_{uj} - \sum_{t \neq v} d_{ut} s_{tj})}{\gamma_d + \gamma_n \sum_{j=1}^{N} s_{vj}^2} \\
\gamma'_{duv} &= \gamma_d + \gamma_n \sum_{j=1}^{N} s_{vj}^2
\end{aligned} \tag{5.22}
$$

[2]Derivations are skipped as they are quite lengthy

For sampling s_{pq} :

$$\begin{aligned} P(s_{pq}|_{\sim}) &= \mathcal{N}\left(s_{pq}|\mu'_{spq}, \gamma'_{sqp}\right) \\ where,\ \mu'_{sqp} &= \frac{\sum_{i=1}^{M}(y_{iq} - \sum_{\tau \neq p} d_{i\tau}s_{\tau q})\gamma_n d_{ip}}{\gamma_{sqp} + \gamma_n \sum_{i=1}^{M} d_{ip}^2} \\ \gamma'_{spq} &= \gamma_{sqp} + \gamma_n \sum_{i=1}^{M} d_{ip}^2 \end{aligned} \tag{5.23}$$

For sampling γ_d :

$$\begin{aligned} P(\gamma_d|_{\sim}) &= Gamma(\alpha'_d, \beta'_d) \\ \alpha'_d &= \alpha_d + \frac{MK}{2} \\ \beta'_d &= \beta_d + \frac{\|D\|_F^2}{2} \end{aligned} \tag{5.24}$$

where, $\|\cdot\|_F$ denotes the Frobenius Norm.

$$\|D\|_F^2 = \sum_{u=1}^{M}\sum_{v=1}^{N} d_{uv}^2$$

For sampling γ_s :

$$\begin{aligned} P(\gamma_s|_{\sim}) &= Gamma(\alpha'_s, \beta'_s) \\ \alpha'_s &= \alpha_s + \frac{KN}{2} \\ \beta'_s &= \beta_s + \frac{\|S\|_F^2}{2} \end{aligned} \tag{5.25}$$

For sampling γ_n :

$$\begin{aligned} P(\gamma_n|_{\sim}) &= Gamma(\alpha'_n, \beta'_n) \\ \alpha'_n &= \alpha_n + \frac{MN}{2} \\ \beta'_n &= \beta_n + \frac{\|Y - DS\|_F^2}{2} \end{aligned} \tag{5.26}$$

5.1.4 Dictionary Learning using Beta-Bernoulli Process - Model 2

Sparsity

In this model rather than forcing sparsity in D and S matrices we introduce another parameter b_{ki}, a discrete valued random variable, to denote whether dictionary atom, d_k is used in encoding the data vector, Y_i. (b_{ki}, π_k) is modeled as Beta-Bernoulli process i.e. $\pi_k = Prob.(b_{ki} = 1)$ is unknown and also modeled as random variable. We keep K sufficiently large and modify the parameters of the parameters of the beta-Bernoulli process to make the B matrix very sparse. This

inturn ensures the sparsity of the model without us having to set the hyperparameters for the D and S matrices. We can now put flat priors on the elements of D and S matrices.

$$
\begin{aligned}
Y &= D(S \odot B) + \eta + \epsilon \\
\eta &= \begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ \eta_0 & \eta_0 & \cdot & \cdot & \eta_0 \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix}_{CXN}
\end{aligned} \tag{5.27}
$$

B is a binary matrix of same dimesions as S matrix. Another bias term η was introduced to handle cases where the columns of the Y matrix may have non-zero mean. (η, γ_η) is also modeled as Normal-Gamma conjugate pair as was done for d_{ik} and s_{ki}. Likelihood and prior distributions are noted down as follows,

$$
\begin{aligned}
(Y_i|D, S_i, B_i, \eta) &\sim \mathcal{N}\left(Y_i|D(S \odot B) + \eta, \gamma_n^{-1} I_M\right) \\
d_k &\sim \mathcal{N}(\mu_d, \gamma_d^{-1} I_M) \\
\gamma_d &\sim Gamma(\alpha_d, \beta_d) \\
s_i &\sim \mathcal{N}(\mu_s, \gamma_s^{-1} I_K) \\
\gamma_s &\sim Gamma(\alpha_s, \beta_s) \\
(B_{ki} = 1|\pi_k) &\sim Bernoulli(\pi_k) \\
\pi_k &\sim Beta(\alpha_\pi, \beta_\pi) \\
\eta_0 &\sim \mathcal{N}(\mu_\eta, \gamma_\eta^{-1} I_M)
\end{aligned} \tag{5.28}
$$

Following are the conditional expression that we found and were required for the Gibbs sampling algorithm.

For sampling d_k:

$$
\begin{aligned}
P(d_k|_\sim) &= \mathcal{N}(d_k|\mu'_{dk}, \gamma'_{dk} I_M) \\
where,\ \gamma'_{dk} &= \gamma_d + \gamma_n \sum_{i=1}^{N} B_{ki} S_{ki}^2 \\
\mu'_{dk} &= \left(\frac{1}{\gamma'_{dk}}\right) [\gamma_d \mu_d + \gamma_n \Delta X_{\sim k}(S_k \odot B_k)] \\
\Delta X_{\sim k} &= Y - D_{\sim k}(S_{\sim k} \odot B_{\sim k})
\end{aligned} \tag{5.29}
$$

For sampling γ_d:

$$\begin{aligned} P(\gamma_d|_\sim) &= Gamma(\alpha'_d, \beta'_d) \\ where, \alpha'_d &= \frac{KM}{2} + \alpha_d \\ \beta'_d &= \frac{\|D\|_F^2}{2} + \beta_d \end{aligned} \tag{5.30}$$

For sampling s_i:

$$\begin{aligned} P(s_i|_\sim) &= \mathcal{N}(s_i|\mu'_{si}, \Gamma'_{si}) \\ where, \Gamma'_{si} &= \gamma_n (DB_i)^T (DB_i) + \gamma_s I_K \\ \mu'_{si} &= \Gamma'^{-1}_{si} (\gamma_n (DB_i)^T (Y_i - \eta) + \gamma_s \mu_s) \end{aligned}$$

For sampling γ_s:

$$\begin{aligned} P(\gamma_s|_\sim) &= Gamma(\alpha'_s, \beta'_s) \\ where, \alpha'_s &= \frac{KN}{2} + \alpha_s \\ \beta'_s &= \frac{\|S\|_F^2}{2} + \beta_s \end{aligned} \tag{5.31}$$

For sampling γ_n:

$$\begin{aligned} P(\gamma_n|_\sim) &= Gamma(\alpha'_n, \beta'_n) \\ where, \alpha'_n &= \frac{MN}{2} + \alpha_n \\ \beta'_s &= \frac{\|X - D(S \odot B)\|_F^2}{2} + \beta_n \end{aligned} \tag{5.32}$$

For sampling B_{ki}:

We normalize the RHS of the following to relation to obtain the conditional probability of $P(B_{ki} = 1|_{\sim})$ or $\pi_i k'$. This probability value is later used in the multilayer model in 7.1.1 as the data for second layer.

$$\begin{aligned} P(B_{ki} = 1|_{\sim}) &\propto \pi_k \left[exp(-\frac{\gamma_n}{2}(s_{ki}^2 d_k^T d_k - 2 s_{ki} d_k^T \Delta X_{i_{\sim k}})) \right] \\ P(B_{ki} = 0|_{\sim}) &\propto (1 - \pi_k) \end{aligned} \tag{5.33}$$

For sampling π_k:

$$\begin{aligned} P(\pi_k|_{\sim}) &= Beta(\alpha'_{\pi}, \beta'_{\pi}) \\ where, \alpha'_{\pi} &= \alpha_{\pi} + \sum_{i=1}^{N} B_{ki} \\ \beta'_{\pi} &= \beta_{\pi} + N - \sum_{i=1}^{N} B_{ki} \end{aligned} \tag{5.34}$$

For sampling η:

$$\begin{aligned} P(\eta|_{\sim}) &= \mathcal{N}(\eta|\mu'_{\eta}, \Gamma_{\eta}^{-1}) \\ where, \Gamma'_{\eta} &= N\gamma_n + \gamma_{\eta} \\ \mu_{\eta} &= \Gamma_{\eta}'^{-1} \left[(Y - D(S \odot B))(\gamma_n \bar{1_N}) \right] \end{aligned} \tag{5.35}$$

5.2 Super-Resolution using Sparse Image Representation

5.2.1 Super-Resolution: A Formal Definition

Super-Resolution is a family of models/algorithms that are used for increasing the resolution of images captured by various imaging devices. The simples of the algorithms in this family are interpolation methods, where missing pixels in the image is replaced by some linear/quadratic/n-degree polynomial function of it's neighbouring pixels. But this method can't create new information; it's like puting the average pixel intensity of the neighbouring pixels. In most of the cases it creates a blurry magnified image and lacks sharpness like in a real higher resolution image. Here we present a statistical model to probabilistically infer the missing pixels by drawing samples from a distribution learned using pretrained dictionaries and the original low resolution image. These dictionaries are initially learned from a dataset of real low and high resolution image pairs.

5.2.2 Objectives of Super-Resolution Method

Sparsity Condition

In the actual problem of single image representation in 5, the sparsity constraint was kept to enforce learning of features that resembles the actual structures that are observed in the images rather than some sinusoidal basis functions like Fourier basis or DCT basis. Though learning such basis is possible only because of the fact that any 256x256 matrix of pixel intensities doesn't necessarily represent a valid image. Real images tend to stay together in a much lower dimensional space within the $256^{N\times N}$ d space, for a image of size $N \times N$. This is as well observed as in various kinds of representation models and also in case of other measurable signals (e.g. speech). This means that if we consider very small patches of size $p \times p$ of a larger image, those pathces can easily represented as a linear combination of K number of patches where $K << 256^{p\times p}$. For example, in our experiments, setting K = 400 proves sufficient to reconstruct any image nearly perfectly with patching windows size of 4(i.e. total possible patches $= 256^{16} = 3.4 \times 10^{38}$). The probability values learned by the model is shown in 6.15. We can see that the probability of selection of a dictioanry atom is nearly zero for the atoms towards the right hand side of the plot. (The atoms are sorted as per the probability values.)

When this constraint was used for the super-resolution problem not only did this helped in learning sparse representations, as explained above, but also helped in learning corresponding feature pairs for the high resolution and low resolution dictionaries. This correspondence between the high and low resolution dictionary atom pairs enables us to recover the high resolution image from the sparse encoding of the downsampled image.

5.2.3 Joint Dictionary Learning - Shared Latent Space Model

In 5 we saw how an image can be represented as a sparse linear combination of a set of small basis images; where each small patch in the image was representated as a linear combination of fixed dictionary / code book of patches. In this section we extend the model to solve the problem of **Image Super-Resolution**, where from a LR image we have to obtain an approximated HR image as close as possible to the ground truth HR image. Usual approaches taken towards Image Super-Resolution uses, a set of low resolution images, differing by small pixel-shifts and rotations, to predict the high resolution image. But single image super resolution task is a much harder as the search space for the high resolution image is much larger (less constrained) compared to that of multiple image super resolution task.

Our approach towards the problem is by learning a shared latent space for the selection matrices as in the dictionary model, namely S and B, for both low resolution and high resolution images. While learning, by using a shared latent space for selection matrices, we force the model to learn dictionary atoms for both resolutions such that corresponding pairs of atoms in the joint dictioanries should be related to each other by a downsampling equivalent operation. This can be observed in the results section 6.14.

5.2.4 Problem Formulation

We first represent the low resolution and high resolution images individually, using the sparse dictionary model as we did in earlier sections for learning a sparse representations for single images.

$$Y_h = D_h S_h \odot B_h + \eta_h + \epsilon_h$$
$$Y_l = D_l S_l \odot B_l + \eta_l + \epsilon_l$$

Now the main idea behind sharing the latent space is that in the sparse decompositions of these two images, the higher resolution image decomposition should ideally be the exact same linear combination as the low resolution image decomposition except that the high resolution atoms being replaced with the lower resolution ones. (D_h replaced by D_l) We achieve this in our model by forcing $S_h = S_l$ and $B_h = B_l$ and solving the two dictionary learning problem simultaneously. The combined model is thus written as,

$$Y_h = D_h S \odot B + \eta_h + \epsilon_h$$
$$Y_l = D_l S \odot B + \eta_l + \epsilon_l$$

We define the sets of hyperparameters, Γ_h and Γ_l,

$$\Gamma_h = \{\gamma_{nh}, \gamma_{dh}, \gamma_s\}$$
$$\Gamma_l = \{\gamma_{nl}, \gamma_{dl}, \gamma_s\}$$

The changes in this model compared to the earlier one (for sparse image representation) are in sampling distributions of S and B, and in rest of the parameters as there would be now two sets of parameters one for each of LR and HR image data matrices. The formulation of the posterior for the vectors of weight matrix, S_i is shown below.

$$
\begin{aligned}
P(S_i|\sim) \propto & P(Y_h|D_h,S,B,\eta_h,\Gamma_h)P(Y_l|D_l,S,B,\eta_l,\Gamma_l)P(S_i|\gamma_s) \\
& \propto \left(\frac{\gamma_{nh}}{2\pi}\right)^{M_h/2} \exp\left(-\frac{\gamma_{nh}}{2}\|Y_{hi}-D_hS_iB_i\|^2\right) \\
& \quad \times \left(\frac{\gamma_{nl}}{2\pi}\right)^{M_l/2} \exp\left(-\frac{\gamma_{nl}}{2}\|Y_{li}-D_lS_iB_i\|^2\right) P(S_i|\gamma_s) \\
& \propto \exp\left(-\frac{\gamma_{nh}}{2}[-2Y_{hi}^T(DB_{hi})S_i + S_i^T(DB_{hi})^T(DB_{hi})S_i]\right) \\
& \quad \times \exp\left(-\frac{\gamma_{nl}}{2}[-2Y_{li}^T(DB_{li})S_i + S_i^T(DB_{li})^T(DB_{li})S_i]\right) P(S_i|\gamma_s)
\end{aligned}
$$

Simplifying further,

$$
\begin{aligned}
P(S_i|\sim) &\propto \mathcal{N}(S_i|\mu'_{si},\Gamma'_{si}) \\
\Gamma'_{si} &= \gamma_{nl}(DB_{li})^T(DB_{li}) + \gamma_{nh}(DB_{hi})^T(DB_{hi}) + \gamma_s I_K \\
\mu'_{si} &= \Gamma'^{-1}_{si}\left[\gamma_{nh}(DB_{hi})^T(Y_{hi}-\eta_h) + \gamma_{nl}(DB_{li})^T(Y_{li}-\eta_l) + \gamma_s\mu_s\right]
\end{aligned}
$$

where, $DB_{hi} = D_h diag(B_i)$

The new conditional posterior distribution for B is given by,

$$
\begin{aligned}
P(B_{ki}=1|\sim) \propto \pi_k & \exp\left(-\frac{\gamma_{nh}}{2}(S_{ki}^2D_{hk}^TD_{hk} - 2S_{ki}D_{hk}^T\Delta Y_{i\sim k}^h)\right) \\
& \times \exp\left(-\frac{\gamma_{nl}}{2}(S_{ki}^2D_{lk}^TD_{lk} - 2S_{ki}D_{lk}^T\Delta Y_{i\sim k}^l)\right) \\
P(B_{ki}=0|\sim) &\propto (1-\pi_k)
\end{aligned}
$$

5.2.5 Applying the model to RGB images

To apply the model on RGB images we first convert them to YCbCr space. Then we apply super-resolution only to the Y-channel of image and for other two channels, Cb and Cr, we use simple bicubic interpolation as the Cb and Cr channel mostly contains low-frequency contents.

The overall training algorithm is given below.

Algorithm 1: Gibbs Sampler for Super-resolution

$Y_h : p_h \times N$ *data matrix*
$Y_l \leftarrow L_f Y_h$
$\Gamma : \{\gamma_{dh}, \gamma_{dl}, \gamma_s, \gamma_{nh}, \gamma_{nl}, \gamma_{biash}, \gamma_{biasl}\}$
Super-Resolution Pseudo-Code:
$\alpha : \{d : 1, s : 1 \times 10^{-1}, bias : 1 \times 10^{-1}, n : 1 \times 10^{-3}\}$
$\beta : \{d : 1, s : 1 \times 10^{-1}, bias : 1 \times 10^{-1}, n : 1 \times 10^{-3}\}$
Initial Samples:
$\{D_h, D_l, S, \Pi, B, \eta_h, \eta_l, \Gamma\} \leftarrow InitialSampleFromPriors()$
Total Gibbs Iterations: gsamples $\leftarrow 5000$
lastSamples = Queue(size = 500)
Initialilze Gibbs Sequence:
$gm_{0s}(1) \leftarrow gm_0$
$gm_{ns}(1) \leftarrow gm_n$
$as(:, 1) \leftarrow a_{new}$
while *Gibbs iterations done ¡ gsamples* **do**
  $Y_{happrox} \leftarrow D_h S \odot B + \eta_h$
  $Y_{lapprox} \leftarrow D_l S \odot B + \eta_l$
  if *lastSamples.length.isfull()* **then**
    lastSamples.removeFrontElement()
  end
  lastSamples.Queue.push($[Y_{happrox}, Y_{lapprox}]$)
  $D_h \leftarrow SampleMVN()$
  $D_l \leftarrow SampleMVN()$
  $S \leftarrow SampleMVN()$
  $\Pi \leftarrow SampleBeta()$
  $B \leftarrow SampleBernoulli()$
  $\eta_l \leftarrow SampleMVN()$
  $\eta_h \leftarrow SampleMVN()$
  $\Gamma \leftarrow SampleGamma(\alpha, 1/\beta)$
end
$a_{approx} \leftarrow mean(as(:, 500 : gsamples), 2)$
$y_{approx} \leftarrow X_{dat} a_{approx}$
Reconstruct_and_plot(mean($lastSamples.Y_{happrox}$))
Reconstruct_and_plot(mean($lastSamples.Y_{lapprox}$))

Chapter 6

Experimental Results

6.1 Sparse Image Representation using Dictionary Learning

6.1.1 Gibbs Sampling from a Joint Posterior Distribution

As we cannot directly draw samples from full posterior distribution, $P(a, \gamma_0, \gamma_n, |Y, X)$, we iteratively draw samples from $P(a|_\sim)$, $P(\gamma_0|_\sim)$ and $P(\gamma_n|_\sim)$ (derived in 5.1.2, note that they follow easily implementable distributions like Normal and Gamma distribution) and use that sequence[1] of observations to approximate the joint distribution.

We ran a simulation for approximation of polynomial coeffcients using Gibbs sampling as explained above. We used the normrnd and gamrnd functions in MATLAB for sampling from the above distributions. Given are the results from the simulation,

Algorithm 2: Gibbs Sampler Pseudo Code - Part 1

Result: MMSE Estimate of Polynmial Coefficients
Initialization:

N $\leftarrow$ 1000;
$x_s \leftarrow$ N random points [Input to the system];
a $\leftarrow$ [1.2;1;2]; //$[a_0, a_1, a_2]$;
X $\leftarrow$ [1 x_s^1 x_s^2];
system_order $\leftarrow$ 3;
$y_0 \leftarrow$ X*a;
Noise Variance $\leftarrow$ 25;
y $\leftarrow$ y_0 + AWGN;

[1] First few hundred samples of this sequence are dropped. This part of the sampling is called burn-in period.

First samples were generated from the polynomial $y_0(x) = a_2x^2 + a_1x + a_0$, then AWGN noise, ϵ was added to generate input to the model.

'SampleGammaRV', 'SampleMVNormal' etc. functions are available in Matlab Statistical Signal processing package. In the second last line of the pseudo code above we discard the first 500 samples to get samples only after mixing is done(Burn-in period). There's no mathematical way to find out the correct burn-in period of the algorithm. We can find this out experimentally only.

System Parameters:

$$\begin{aligned} y &= y_0 + \epsilon \\ [a_0 a_1 a_2] &= [1.2\ 1\ 2] \\ \epsilon &\sim \mathcal{N}(0, \sigma^2)\ where,\ \sigma^2 = 25 \end{aligned}$$

Estimated Coefficients:

σ	a_0	a_1	a_2
1	1.17	0.9946	1.9973
10	1.52	0.83	1.998
25	3.142	0.992	1.969

Table 6.1: Gibbs Sampling Example: Coefficient Estimation

We see from the results that the higher order coefficients are estimated with quite high accuracy. This model is very similar to the actual Dictionary learning model as we explained earlier. In this case the dictionary equivalent matrix X_{dat} was kept fixed the weight matrix(rather a vector) equivalent a was estimated using Bayesian inference.

In the following plots we can see the performance of the proposed Dictionary learning model. Even under noise variance as high as 25 the model is able to estimate the coefficients with a maximum of 1% and 1.5% errors in the estimated

coefficients of linear and quadratic terms, respectively.

Algorithm 3: Gibbs Sampler Pseudo Code - Part 2

Set Hyperparameters:

$\alpha_0 \leftarrow$ 1e-1;
$\beta_0 \leftarrow$ 1e-1;
$\alpha_n \leftarrow$ 1e-1;
$\beta_n \leftarrow$ 1e-1;

Initial Samples: $gm_0 \leftarrow Sample\ from\ Gamma\ Distribution(\alpha_0, 1/\beta_0)$;
$\mu \leftarrow zeros(systemorder, 1)$;
$\Sigma \leftarrow 1/gm_0 I_{system\_order}$;
$a_{new} \leftarrow SamplefromMultiVariateNormal(\mu, \Sigma)$;
$gm_n \leftarrow gamrnd(\alpha_n, 1/\beta_n)$;
[Total Gibbs Iterations] gsamples $\leftarrow$ 5000;

$X_{dat} \leftarrow X$;
$xtx \leftarrow X_{dat}^T X_{dat}$;

Initialilze Gibbs Sequence:

$gm_{0s}(1) \leftarrow gm_0$;
$gm_{ns}(1) \leftarrow gm_n$;
$as(:, 1) \leftarrow a_{new}$;
while *gsamples Gibbs iterations not complete* **do**
  $it \leftarrow CurrentGibbsIteration$;
  $M \leftarrow xtx + gm_{0s}(it-1)/gm_{ns}(it-1) I_{new\_order}$;
  $u \leftarrow M^{-1} X_{dat}^T y$;
  $sig \leftarrow (1/gm_{ns}(it-1)) M^{-1}$;
  $as(:, it) \leftarrow SampleMVNormalRV(\mu = u, \Sigma = sig)$;
  $\alpha \leftarrow \alpha_0 + new\_order/2$;
  $\beta \leftarrow \beta_0 + as(:, it)^T as(:, it)/2$;
  $gm_{0s}(it) \leftarrow SampleGammaRV(\alpha, \beta)$;
  $\alpha \leftarrow \alpha_n + gsamples/2$;
  $\beta \leftarrow \beta_n + norm(y - X_{dat} as(:, it))^2/2$;
  $gm_{ns}(it) \leftarrow SampleGammaRV(\alpha, \beta)$;
end
$a_{approx} \leftarrow mean(as(:, 500 : gsamples), 2)$;
$y_{approx} \leftarrow X_{dat} a_{approx}$;

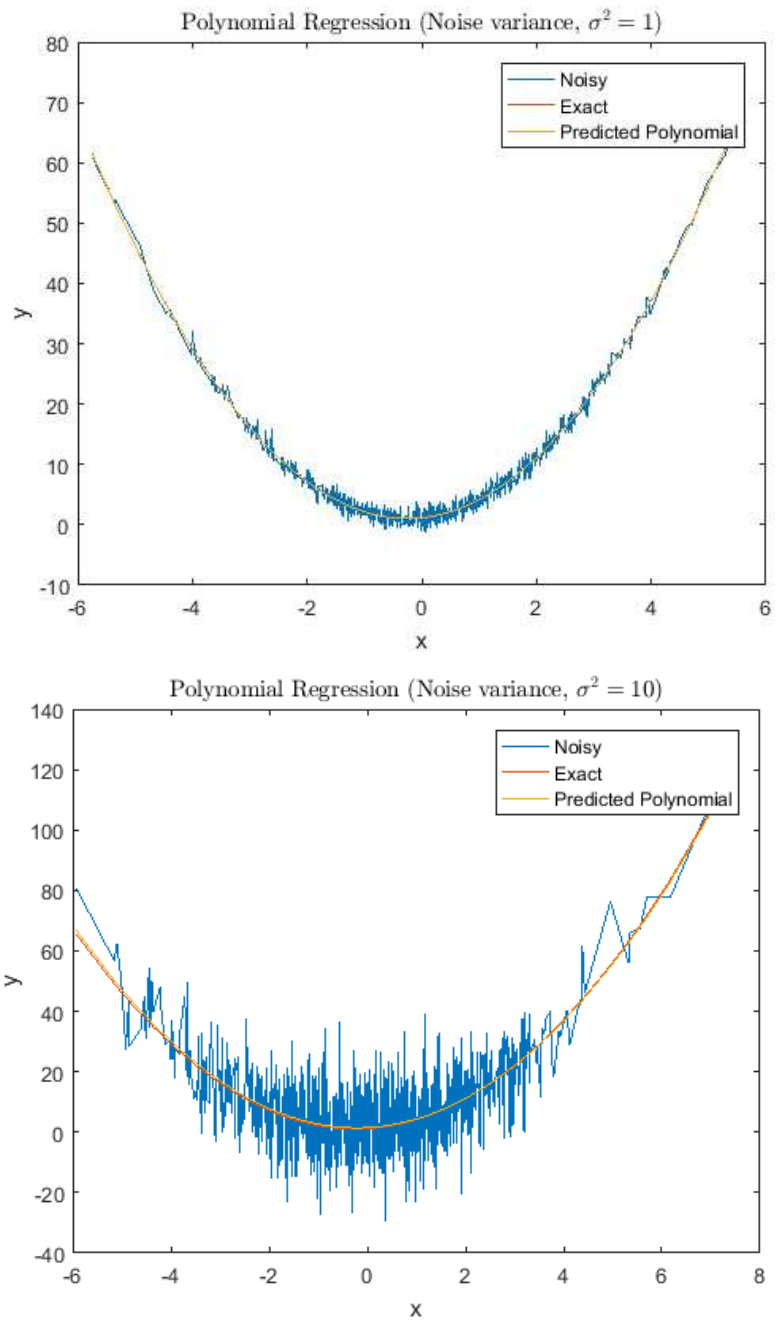


Figure 6.1: Performance of Bayesian Method in Polynomial Regression for different noise variance

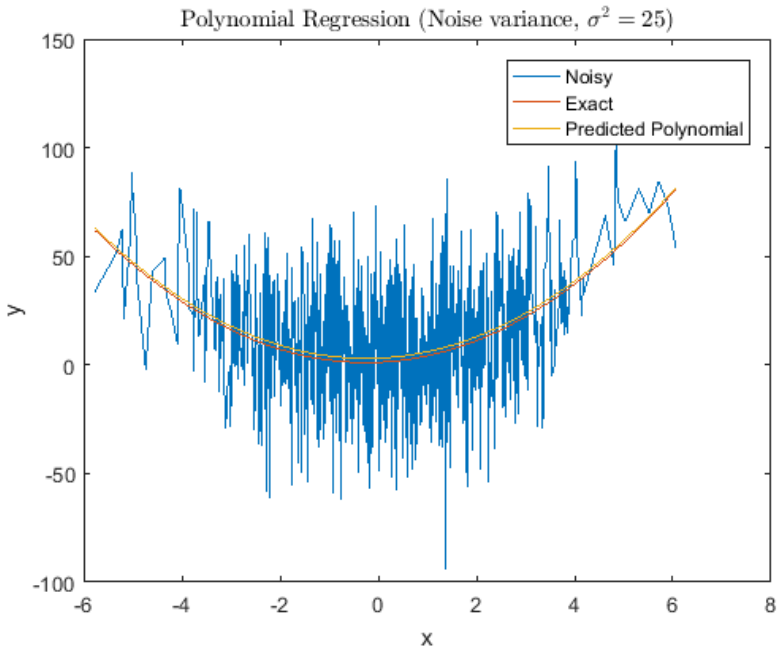


Figure 6.2: Performance of Bayesian Method in Polynomial Regression for different noise variance

6.1.2 Preprocessing of Image Data

System Setup:

For verification and performance evaluations of our models, the algorithms were implemented using MATLAB and tested on various standard datasets of images. The models were trained on two machines a Xeon server (12 cores and 128 GB RAM) and a Core-i7 (8 Cores and 16 GB RAM with NVidia GTX960m GPU). GPU was required for learning of the joint dictionaries in the second part of the problem, i.e. super-resolution, to decrease the training time of the algorithm.

Generating the Data Matrix:

Images with the category 'Faces_easy' from Caltech10 dataset were used as the training dataset. In the preprocessing step the images were first downsized to 128x128 or 256x256 dimensions and converted to grayscale images. After that using a window of 8x8 or 16x16 overlapping patches were generated, each of which was vectorized and finally put together as the training data matrix. All the matrices generated from different images are concatenated together to create a single global matrix. Location of each image inside the matrix was stored for reconstruction and future references.

For the super-resolution task various images were chosen from all 3 datasets with appropriate amount of details for training and testing. In this part also data matrices were generated from images using the same method as explained above.

Though in the training section of the super-resolution algorithm there were two seperate data matrices Y_h and Y_l for high and low resolution images, respectively.

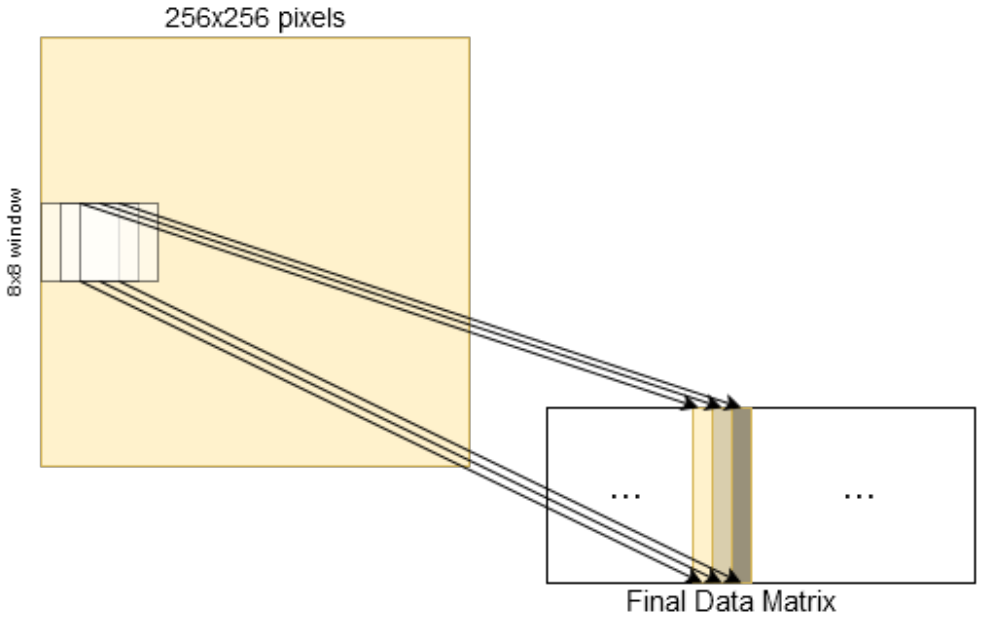


Figure 6.3: Generation of Training Data Matrix from Images

6.1.3 Dataset

For testing the algorithm following datasets were used,

- **DEAP Dataset:** DEAP [18] or Database for Emotion Analysis using Physiological Signals is a multimodal dataset with recordings of 32 channel EEG signals and 8 other peripheral physiological signals. This dataset has recordings from 32 volunteers.
- **Caltech 101:** This is a categorized image database with more than 40 sample images for each category. This database is very suitable for training statistical or machine learning models.
- **Set 5:** Dataset used for Super-resolution Evaluation [23]
- **Set 14:** Dataset used for Super-resolution Evaluation [23]

6.1.4 Learning Features for Images

Setting the Priors

Following were the values set for the parameters of the final model. Note that precision γ_d and γ_s has nearly flat priors (by setting a high variance).

```
%% Initialize
K1 = 40;
K2 = 80;

Alpha1 = {};
Beta1 = {};

% Params for gamma distro
Alpha1.d = 1e-1;
Beta1.d = 1e-1;
Alpha1.s = 1e-1;
Beta1.s = 1e-1;
Alpha1.bias = 1e-1;
Beta1.bias = 1e-1;
Alpha1.n = 1e-3;
Beta1.n = 1e-3;

% Params for beta distro : Near to zero, sparse
Alpha1.pi = 1;
Beta1.pi = 1800;

Alpha2 = Alpha1;
Beta2 = Beta1;

Alpha2.pi = 1;
Beta2.pi = 2000;
```

Let's caculate the mean and variance of γ_d and γ_s,

$$mean(\gamma_d) = \frac{\alpha_d}{\beta_d} = 1$$
$$var(\gamma_d) = \frac{\alpha_d}{\beta_d^2} = 10$$

$$mean(\gamma_s) = \frac{\alpha_s}{\beta_s} = 1$$
$$var(\gamma_s) = \frac{\alpha_s}{\beta_s^2} = 10$$

$$mean(\gamma_n) = \frac{\alpha_n}{\beta_n} = 1$$
$$var(\gamma_n) = \frac{\alpha_n}{\beta_n^2} = 1000$$

The parameter π_k follows beta distribution. So,

$$mean(\pi_k) = \frac{\alpha}{\alpha+\beta} = 5.52 \times 10^{-4}$$
$$var(\pi_k) = \frac{\alpha\beta}{(\alpha+\beta)^2(\alpha+\beta+1)} = 3.08 \times 10^{-7}$$

6.1.5 Output Plots

MSE vs. Gibbs Iterations

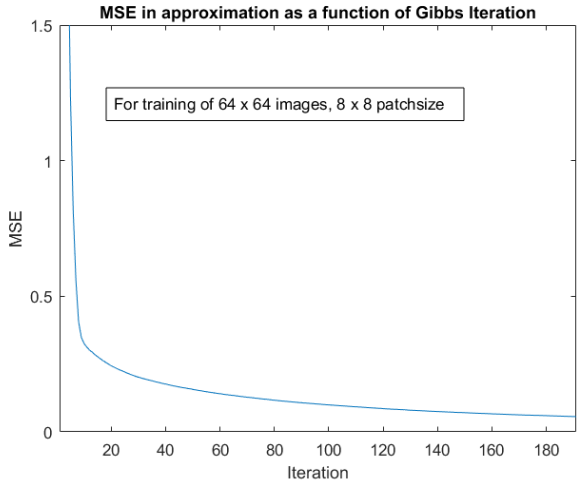


Figure 6.4: MSE in approximation as a function of Gibbs Iterations

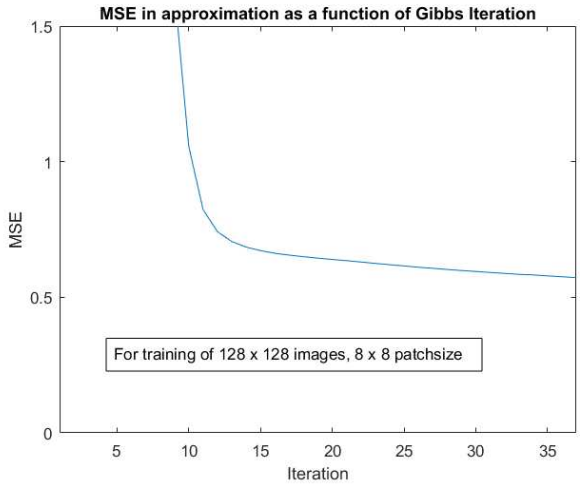


Figure 6.5: MSE in approximation as a function of Gibbs Iterations

Reconstruction

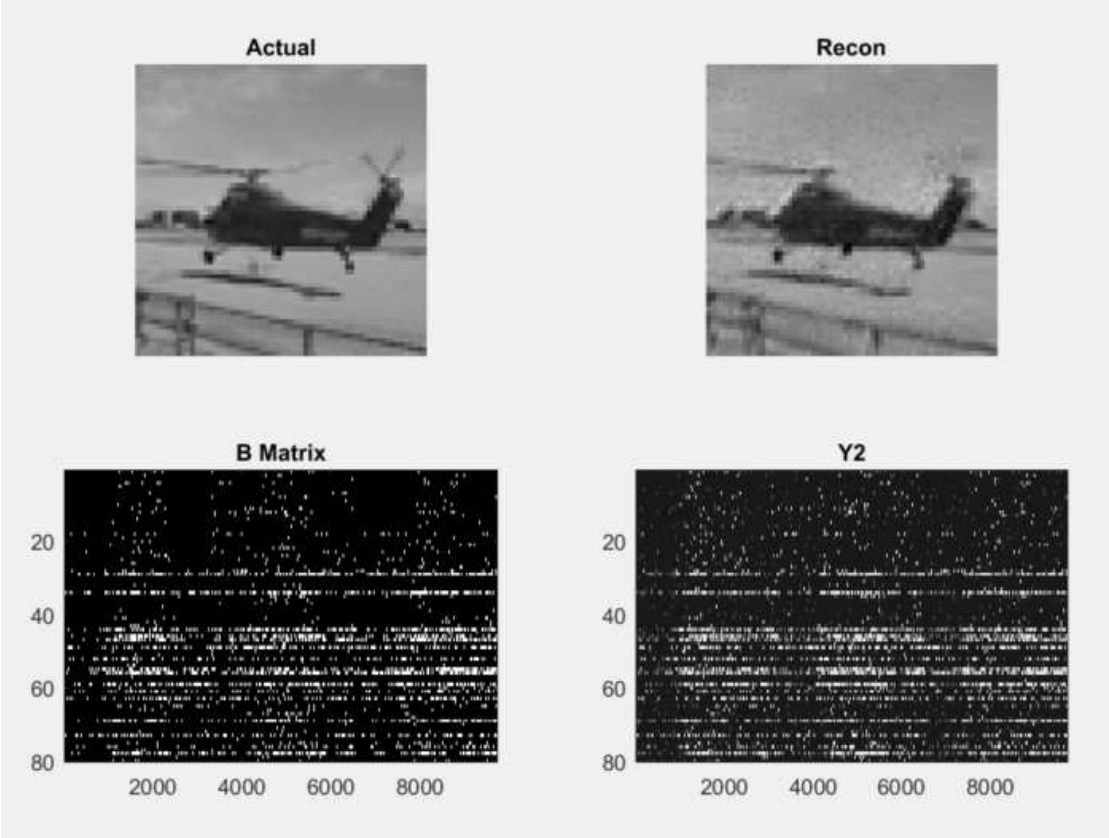


Figure 6.6: Reconstructed image from coefficients and features estimated by the algorithm

Figure 6.7: Image Reconstruction Example 1, A. Actual Image B. Reconstructed Image

Figure 6.8: Image Reconstruction Example 2, A. Actual Image B. Reconstructed Image

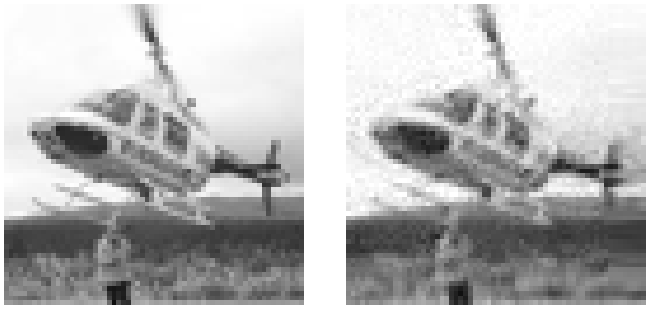


Figure 6.9: Image Reconstruction Example 3, A. Actual Image B. Reconstructed Image

Learned Features

Features Learned with Sparse Priors:

After learning was complete we reconstructed the images using single dictionary atoms and the output were similar to edge filtered version of those images. Thus our model in a completely unsupervised manner was able to learn features which are known for their usefulness in various image processing tasks.

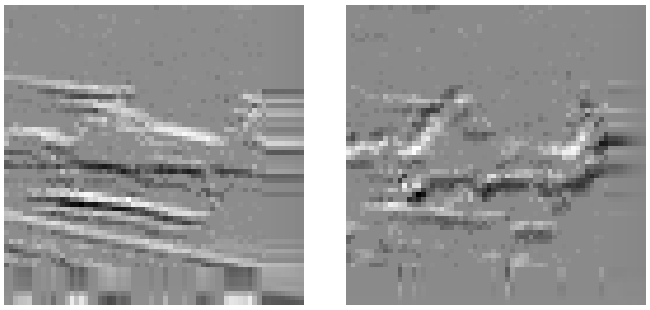


Figure 6.10: Reconstruction of Images using single dictionary elements

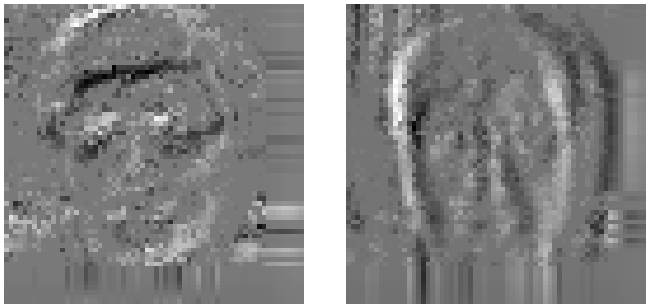


Figure 6.11: Reconstruction of Images using single dictionary elements

Dictionary Columns Learned with Sparse Priors:

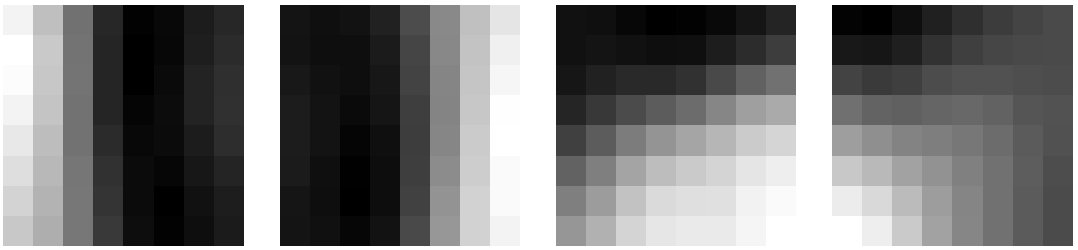


Figure 6.12: Dictionary Atoms learned by the algorithm

Features Learned with Non-Sparse Priors:

We also validated our model by taking away the sparsity settings which ended up learning noisy features because of overfitting. Hence our prior settings also acted as a regularization method for the algorithm.

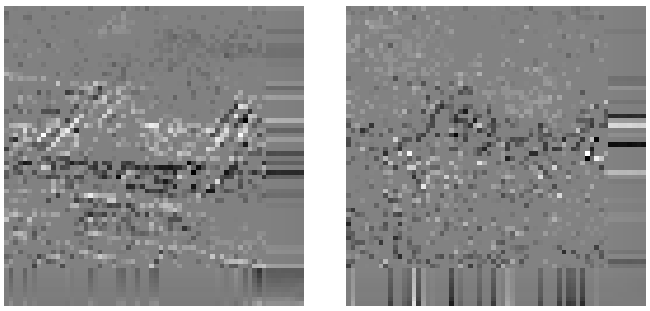


Figure 6.13: Reconstruction of Images using single dictionary elements

6.2 Super-Resolution using Sparse Image Representation

6.2.1 Training: Learning the Joint-Dictionaries

The Gibbs Sampler for learning of the joint dictionary for the super-resolution problem was very similar to the one used in the earlier sections for sparse image representations of single images. Except for the conditional posterior distributions

for sampling of the weight matrix, S, and selection matrix, B, rest of the samplers were same as the ones derived for the sparse image representation problem. Although nature of the posterior distributions were still the same for both the problems. The details of the sampling distributions are explained in 5.2.3.

First step of the training algorithm was to prepare the training data for the images. How the training data matrices were generated are explained in the section 5. For applying this algorithm to color images, the RGB format images are first converted to HSV format and then super-resolution algorithm is only applied to H-channel as data S and V-channel are relatively low frequency; so using bicubic interpolation is sufficient for those two channels.

The pseudo code below illustrates the Gibbs Sampler for learning the joint dictionary pair. Each of the sampling function wrapper used in the pseudo code generates samples as per the conditional posterior distributions of the parameters in the graphical model derived in the earlier section 5.2.3. Mean of the samples generated in last I (from 100 to 500) iterations were used to reconstruct the magnified high resolution images.

During iterations of Gibbs sampling, Y_h and Y_l was used to sample D_h and D_l independently based the same S and B matrices thus enabling sharing of latent space. As indicated by the new sampling equations in 5.2.3 and quite intuitively the sampling of S and B requires both high resolution and low resolution data matrices and dictionaries.

The joint dictionaries learned by the model is shown in 6.14. The dictionaries are plotted in order of their selection probabilities. This is done by taking rowwise mean of the B matrix. Note that how most of the LR dictionary atoms are just downsampled version of corresponding HR dictionary atoms. In 6.15 we have plotted the sorted selection probabilities of the dictionary atoms. From these we can see that the model doesn't even need all the 400 atoms for sparse approximation of the images. As expected it was observed that the required number of dictionary atoms increases with increasing patch size.

Hyperparameter settings:

Due to huge size of data matrices doing hyperparameter optimization based on cross-validation is computationally intractable. So the same hyperparameter settings, obtained during sparse image representation, is used for the joint dictionary learning for the super-resolution problem also.

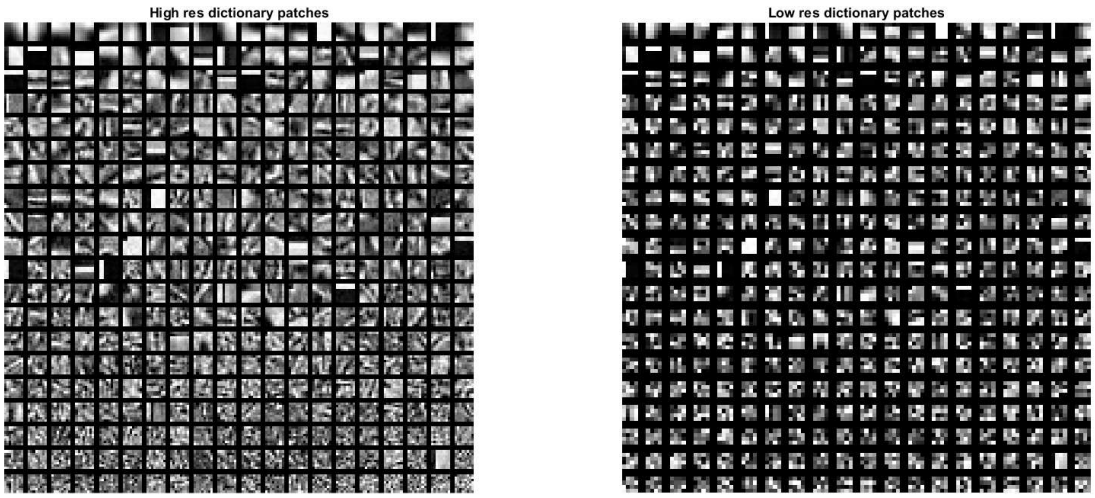


Figure 6.14: Jointly Learned Dictionaries

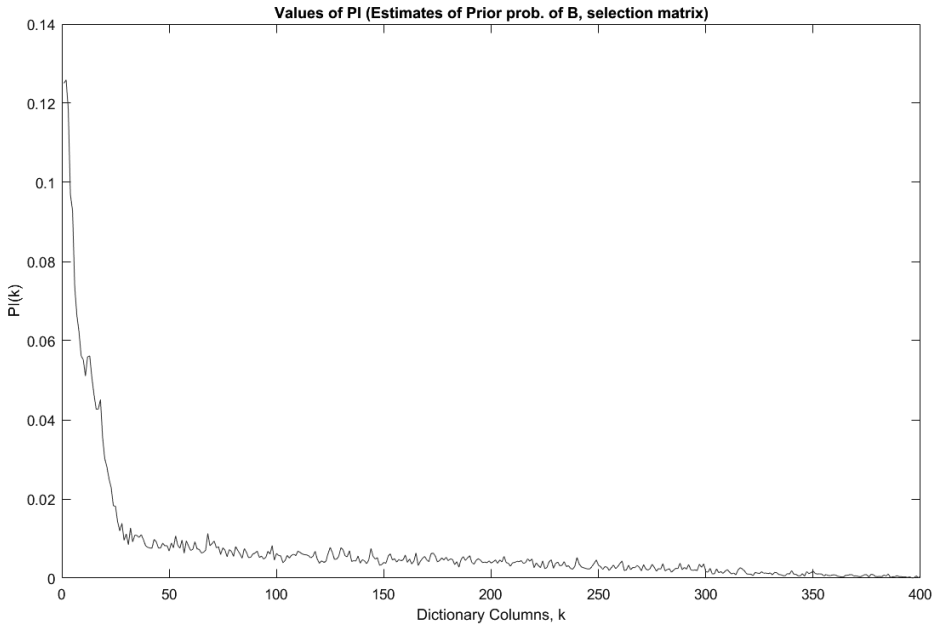


Figure 6.15: Learned Probabilities for Joint Dictionary Atoms

6.2.2 Testing

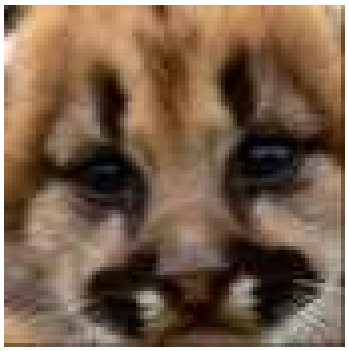
After training is done, the joint dictioanries, D_h and D_l are saved along with other model parameters like Π, Γ_n etc as there is no need to sample those parameters again, which would add a time overhead per iteration during testing. For testing the model, the dictionaries learned in the training section are fixed and S and B matrices are inferred using the same Gibbs sampler as the one used for sparse image representation with the dictionary fixed to D_h.

6.2.3 Super-Resolution Results

RMSE of magnified image w.r.t. original High Resolution Image:

	PSNR		RMSE	
	Bicubic	DL	Bicubic	DL
Racoon	27.64	28.76	10.58	9.3
Butterfly	22.66	23.86	18.76	16.35
Girl Face	30.16	32.05	7.91	6.37

Table 6.2: Results: 2x Super-resolution



(a) Low Res

(b) Bicubic

(c) High Res

(d) Model Super-resolution

Figure 6.16: Super-Resolution(2x magnification) Examples

The PSNR for DL model based HR reconstruction may be just slightly better from the ones produced by bicubic interpolation but PSNR values doesn't really

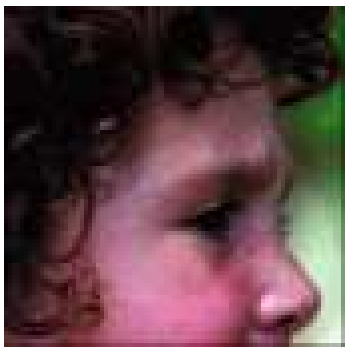
(a) Low Res

(b) Bicubic

(c) High Res

(d) Model Super-resolution

Figure 6.17: Super-Resolution(2x magnification) Examples



(a) Low Res

(b) Bicubic

(c) High Res

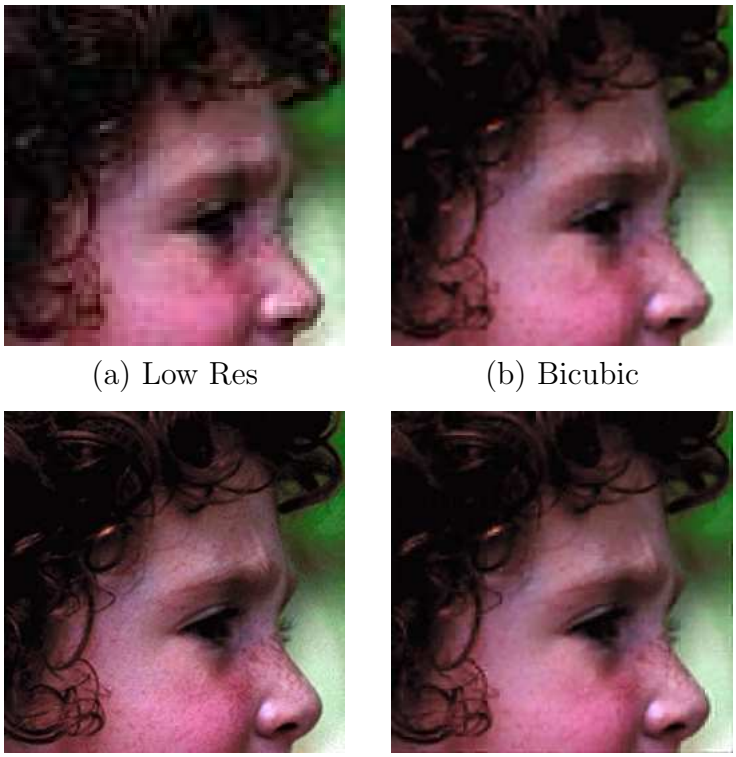
(d) Model Super-resolution

Figure 6.18: Super-Resolution(2x magnification) Examples

	PSNR		RMSE	
	Bicubic	DL	Bicubic	DL
Racoon	25.99	27.98	12.78	10.17
Girl Face	29.96	31.53	8.09	6.76
Butterfly	19.94	22.78	25.66	18.5

Table 6.3: Results: 3x Super-resolution

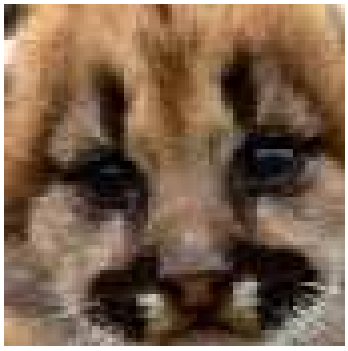
infer much about visual qualities of a image. From the sample outputs given here it can be seen that the reconstructions produced by our model have more details and are of better quality in terms of blurriness.



(a) Low Res (b) Bicubic

(c) High Res (d) Model Super-resolution

Figure 6.19: Super-Resolution(3x magnification) Examples



(a) Low Res

(b) Bicubic

(c) High Res

(d) Model Super-resolution

Figure 6.20: Super-Resolution(3x magnification) Examples

(a) Low Res

(b) Bicubic

(c) High Res

(d) Model Super-resolution

Figure 6.21: Super-Resolution(3x magnification) Examples

Chapter 7

Conclusion

In this work we showed how Bayesian Inference can be used to design a Generative Model for superresolution. The problem is formulated as a joint dictionary learning task. The model hyperparameters were first tuned for the sparse image representation task. Later the same model have been used for the superresolution task by just replacing two sampling distribution in the Gibbs sampler. Among the two dictionary models the one leveraging the Beta-Bernoulli process for realizing sparsity learns much faster and better than the other. As a non-parametric model, theoretically, we should have kept size of the dictionary much higher that what have been used but suggested by the learnt selection matrix only a fourth of the dictionary columns have reasonable non-negligible probability meaning that adding any further dictionary elements may just learn some noisy details corresponding to some residual patches. How the model process a image by first patching and then learning a weight corresponding to dictionary atom is very much like what a layer in Convolutional Neural Network / CNN does. Thus the model presented here can also be further modified to have several layers of dictionary learner just like a CNN which may be able to learn more abstract and domain specific features (e.g eyes, lips or nose for face images) of an image. Though learning such a model would pose a high computational complexity.

7.1 Future Work

7.1.1 Deep Network using (Multilevel) Dictionary Learning

We saw that the first layer of our model learns useful features like edges/corners. But if we now combine these features to learn new features, they should have much more compicated structure which resembles the shape of actual object more. This can be done in either same scale of a larger scale by increasing the patch window size in the second layer. We have already tested version of this which got partially correct results but due to computational issues we haven't been able to produce full results to be presented at the moment.

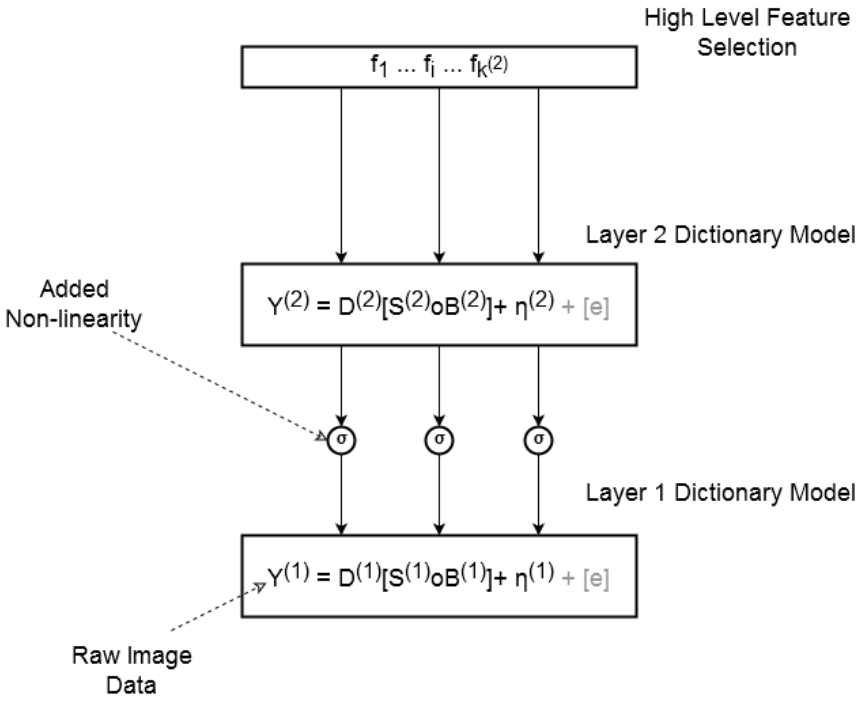
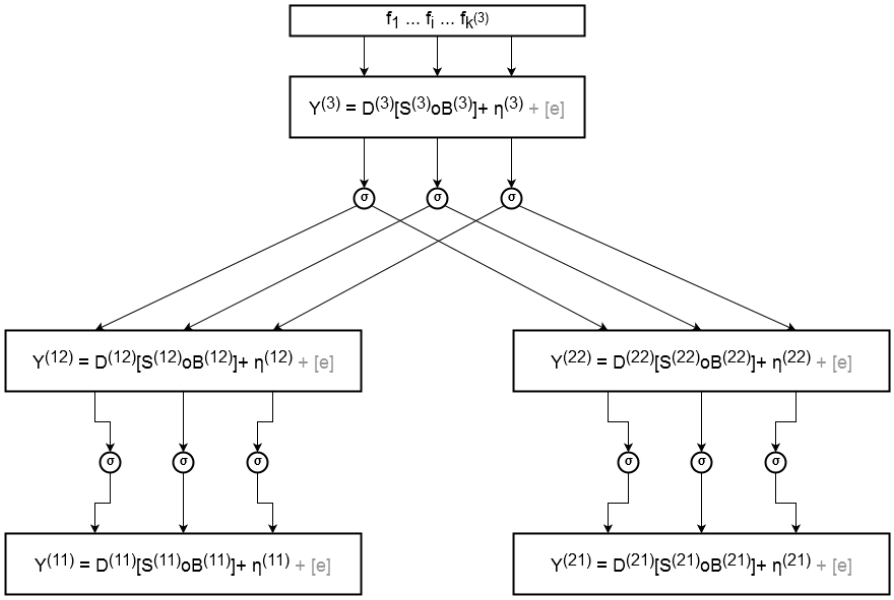


Figure 7.1: Multilayer Dictionary Learning Model

7.1.2 Multimodal Dictionary Learning



As shown in [7] and [10] multilayer RBMs and Deep networks can be used successfully learn unsupervised features not only for a single modality but also for multiple modalities when trained using related data across modalities.

Following the same ideology we also propose a multimodal architecture for dictionary learning. Given that multilayer networks learn abstract features in the higher layers, two such networks working on different modalities (e.g. audio, im-

age, text etc.) if merged together at the higher levels should be able to learn related cross-modality features.

Bibliography

[1] Sung Cheol Park, Min Kyu Park, and Moon Gi Kang. Super-resolution image reconstruction: a technical overview. *IEEE signal processing magazine*, 20(3):21–36, 2003.

[2] Philip Resnik and Eric Hardisty. Gibbs sampling for the uninitiated. Technical report, DTIC Document, 2010.

[3] Karl Skretting. Dictionary learning tools for matlab. Technical report, University of Stavanger, 2014.

[4] John Paisley and Lawrence Carin. Nonparametric factor analysis with beta process priors. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 777–784. ACM, 2009.

[5] Bo Chen, Gungor Polatkan, Guillermo Sapiro, Lawrence Carin, and David B Dunson. The hierarchical beta process for convolutional factor analysis and deep learning. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 361–368, 2011.

[6] Bo Chen, Gungor Polatkan, Guillermo Sapiro, David Blei, David Dunson, and Lawrence Carin. Deep learning with hierarchical convolutional factor analysis. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1887–1901, 2013.

[7] Nitish Srivastava and Ruslan R Salakhutdinov. Multimodal learning with deep boltzmann machines. In *Advances in neural information processing systems*, pages 2222–2230, 2012.

[8] Li Fei-Fei and Pietro Perona. A bayesian hierarchical model for learning natural scene categories. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 524–531. IEEE, 2005.

[9] Honglak Lee, Roger Grosse, Rajesh Ranganath, and Andrew Y Ng. Convolutional deep belief networks for scalable unsupervised learning of hierarchical representations. In *Proceedings of the 26th annual international conference on machine learning*, pages 609–616. ACM, 2009.

[10] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y Ng. Multimodal deep learning. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 689–696, 2011.

[11] Jianchao Yang, John Wright, Thomas S Huang, and Yi Ma. Image super-resolution via sparse representation. *IEEE transactions on image processing*, 19(11):2861–2873, 2010.

[12] Mehdi SM Sajjadi, Bernhard Schölkopf, and Michael Hirsch. Enhancenet: Single image super-resolution through automated texture synthesis. *arXiv preprint arXiv:1612.07919*, 2016.

[13] S Derin Babacan, Rafael Molina, and Aggelos K Katsaggelos. Variational bayesian super resolution. *IEEE Transactions on Image Processing*, 20(4):984–999, 2011.

[14] Chao Dong, Chen Change Loy, Kaiming He, and Xiaoou Tang. Image super-resolution using deep convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 38(2):295–307, 2016.

[15] Honglak Lee, Alexis Battle, Rajat Raina, and Andrew Y Ng. Efficient sparse coding algorithms. In *Advances in neural information processing systems*, pages 801–808, 2006.

[16] Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137, 2015.

[17] Zoubin Ghahramani and Thomas L Griffiths. Infinite latent feature models and the indian buffet process. In *Advances in neural information processing systems*, pages 475–482, 2005.

[18] Sander Koelstra, Christian Muhl, Mohammad Soleymani, Jong-Seok Lee, Ashkan Yazdani, Touradj Ebrahimi, Thierry Pun, Anton Nijholt, and Ioannis Patras. Deap: A database for emotion analysis; using physiological signals. *IEEE Transactions on Affective Computing*, 3(1):18–31, 2012.

[19] R. Fergus L. Fei-Fei and P. Perona. Learning generative visual models from few training examples: an incremental bayesian approach tested on 101 object categories. *CVPR, Workshop on Generative-Model Based Vision*, 2004.

[20] Wikipedia. Bayesian inference — wikipedia, the free encyclopedia, 2016. [Online; accessed 21-November-2016].

[21] Wikipedia. Conjugate prior — wikipedia, the free encyclopedia, 2016. [Online; accessed 1-November-2016].

[22] Wikipedia. Sparse dictionary learning — wikipedia, the free encyclopedia, 2016. [Online; accessed 19-November-2016].

[23] Jia-Bin Huang, Abhishek Singh, and Narendra Ahuja. Single image super-resolution from transformed self-exemplars. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5197–5206, 2015.