# Drosophila Stage Annotation using Sparse Learning Method

by

Cheng Pan

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved November 2012 by the
Graduate Supervisory Committee:

Jieping Ye, Chair
Baoxin Li
Gerald Farin

ARIZONA STATE UNIVERSITY

December 2012

ABSTRACT

*Drosophila melanogaster*, as an important model organism, is used to explore the mechanism which governs cell differentiation and embryonic development. Understanding the mechanism will help to reveal the effects of genes on other species or even human beings. Currently, digital camera techniques make high quality *Drosophila* gene expression imaging possible. On the other hand, due to the advances in biology, gene expression images which can reveal spatiotemporal patterns are generated in a high-throughput pace. Thus, an automated and efficient system that can analyze gene expression will become a necessary tool for investigating the gene functions, interactions and developmental processes. One investigation method is to compare the expression patterns of different developmental stages. Recently, however, the expression patterns are manually annotated with rough stage ranges. The work of annotation requires professional knowledge from experienced biologists. Hence, how to transfer the domain knowledge in biology into an automated system which can automatically annotate the patterns provides a challenging problem for computer scientists. In this thesis, the problem of stage annotation for *Drosophila* embryo is modeled in the machine learning framework. Three sparse learning algorithms and one ensemble algorithm are used to attack the problem. The sparse algorithms are Lasso, group Lasso and sparse group Lasso. The ensemble algorithm is based on a voting method. Besides that the proposed algorithms can annotate the patterns to stages instead of stage ranges with high accuracy; the decimal stage annotation algorithm presents a novel way to annotate the patterns to decimal stages. In addition, some analysis on the algorithm performance are made and corresponding explanations are given. Finally, with the proposed system, all the lateral view BDGP and FlyFish images are annotated and several interesting applications of decimal stage value are revealed.

ACKNOWLEDGEMENTS

I would like to thank all the students and professors who helped or taught me during my two and a half years of graduate study at Arizona State University. I would like to thank Dr. Jieping Ye for his help in academia and research and for providing such an interesting and challenging problem to me. I would like to thank Dr. Baoxin Li and Dr. Gerald Farin for being a part of my thesis committee and giving me instructions on my thesis. I would like to thank Lei for giving me a lot of helpful suggestions during the process of algorithm design and performance evaluation. I would like to thank Qian for her review and helpful suggestions to my thesis. Last but not least, I would like to thank my parents, my aunt and my uncle. It is you who supported my study at ASU and gave me the courage to pursue my master's degree.

TABLE OF CONTENTS

Page

LIST OF TABLES . . . . . vi

LIST OF FIGURES . . . . . viii

LIST OF SYMBOLS/NOMENCLATURE . . . . . xi

CHAPTER

1 INTRODUCTION . . . . . 1

1.1 Background . . . . . 1

1.2 Challenges . . . . . 2

Large Data Size Challenge . . . . . 3

High Feature Dimension Challenge . . . . . 3

Application Specific Challenges . . . . . 4

1.3 Problem Definition . . . . . 4

1.4 Thesis Organization . . . . . 5

2 RELATED WORKS . . . . . 6

2.1 LdaPath . . . . . 6

2.2 Bag of Words . . . . . 7

3 OVERVIEW OF LINEAR CLASSIFIERS . . . . . 8

3.1 Support Vector Machine . . . . . 8

3.2 Logistic Regression . . . . . 8

3.3 Ridge Regression . . . . . 9

3.4 Lasso . . . . . 10

4 DATA SETS AND PREPROCESSING . . . . . 12

4.1 Two Data Sets . . . . . 12

4.2 Gabor Feature Extraction . . . . . 13

4.3 Training Data Sets Generation . . . . . 14

CHAPTER Page

4.4 Genomewide-Expression-Map . . . . . 14
5 ANNOTATION BY LEARNING SPARSE STRUCTURE . . . . . 17
5.1 Algorithms . . . . . 17
Lasso . . . . . 18
Group Lasso . . . . . 18
Sparse Group Lasso . . . . . 19
5.2 Annotation System . . . . . 20
5.3 Evaluation Setup . . . . . 23
5.4 Results and Analysis . . . . . 23
Evaluation on Accuracy . . . . . 23
Evaluation on Sparsity . . . . . 24
6 ANNOTATION BY ENSEMBLE . . . . . 27
6.1 Algorithm . . . . . 27
Model Pool Construction . . . . . 28
Stage Annotation by Majority Voting . . . . . 29
6.2 Ensemble Annotation System . . . . . 30
6.3 Results and Analysis . . . . . 30
7 ANNOTATION AS DECIMAL STAGE . . . . . 32
7.1 Expression Pattern Transition as a Continuous Process . . . . . 33
7.2 Without-range Problem . . . . . 34
7.3 Algorithm . . . . . 36
7.4 Decimal Stage Annotation System . . . . . 37
7.5 Applications of Decimal Stage . . . . . 38
Image Sorting . . . . . 38
Sub-stage Annotation . . . . . 39

CHAPTER Page

Improving Similar Expression Pattern Retrieval . . . . . 42

7.6 Results and Analysis . . . . . 43

With-range Problem vs. Without-range Problem . . . . . 44

Independent Evaluation . . . . . 45

8 CONCLUSIONS . . . . . 48

REFERENCES . . . . . 51

APPENDIX

A FLYFISH GEMS OVERLAID WITH ATLAS . . . . . 54

B SUB-STAGE GEMS FOR FLYFISH . . . . . 56

LIST OF TABLES

Table Page

4.1 Number of BDGP images within the same stage range. . . . . 16

4.2 Number of FlyFish images within the same stage range. . . . . 16

5.1 $\lambda$ values used in cross validation for Lasso and group Lasso. . . . . 22

5.2 $\lambda$ value pairs used in cross validation for sparse group Lasso. . . . . 22

5.3 Algorithm accuracies evaluated on BDGP data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios. . . . . 24

5.4 Algorithm accuracies evaluated on FlyFish data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios. . . . . 24

7.1 Number of images in each stage for the BDGP data set. All the 36802 images are annotated by solving the with-range problem. . . . . 38

7.2 Number of images in each stage for the FlyFish data set. All the 33572 images are annotated by solving the with-range problem. . . . . 39

7.3 Without-range algorithm accuracies evaluated on the BDGP data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios. . . . . 44

7.4 Without-range algorithm accuracies evaluated on the FlyFish data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios. . . . . 44

Table Page

7.5 Independent performance evaluation results of the annotation system (stage-range information is used). . . . . 46

7.6 Independent performance evaluation results of the annotation system (stage-range information is not used). . . . . 46

7.7 Independent performance evaluation results of our annotation system (stage-range information is used). Only SVM models are used in ensemble method. . . . . 47

7.8 Independent performance evaluation results of our annotation system (stage-range information is not used). Only SVM models are used in ensemble method. . . . . 47

7.9 Independent performance evaluation results of our annotation system (stage-range information is used). 6 sparse models are used in ensemble method. . . . . 47

7.10 Independent performance evaluation results of our annotation system (stage-range information is not used). 6 sparse models are used in ensemble method. . . . . 47

LIST OF FIGURES

Figure Page

1.1 Sample images for BDGP data sets. There are 6 stage ranges for BDGP data. Each image is a representative for the corresponding stage range. . 2

1.2 Sample images for FlyFish data sets. There are 4 stage ranges for FlyFish data. Each image is a representative for the corresponding stage range. . . . . 2

4.1 Data preprocessing flow. The original image insitu5449 is fetched from FlyExpress database. The binary image is generated by thresholding the original image. The feature vector is extracted from original image by Gabor filters. . . . . 13

4.2 GEMs for BDGP. The numbers below the images are stage values. Each GEM is generated by overlapping the binary expression images of the same stage. . . . . 15

4.3 GEMs for FlyFish. The numbers below the images are stage values. Each GEM is generated by overlapping the binary expression images of the same stage. . . . . 15

5.1 Annotation algorithm flow for BDGP data set. . . . . 21

5.2 Sparsity for each proposed algorithms on BDGP data set. . . . . 25

5.3 Sparsity for each proposed algorithms on FlyFish data set. . . . . 25

6.1 Model pool high-level illustration. The left side shows the split ratios and algorithms which are used to construct the model pool. Each bin contains 30 models. Hence, there are 1050 models in the model pool. . . 28

6.2 Ensemble annotation system overview. The upper component is consisted of a set of stage annotation algorithms. The lower component uses the voting method to combine the results of the annotation algorithms. . . . . 30

6.3 Generated BDGP GEMs overlapped with Atlas. . . . . 31

7.1 Without-range stage annotation algorithm flow. . . . . 35

7.2 Final version of stage annotation system overview. Each module is an algorithm that are posed in previous chapters. . . . . 37

7.3 Stage 4 - 6 BDGP GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles. . . . . 40

7.4 Stage 7 - 12 BDGP GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles. . . . . 41

7.5 Stage 13 - 16 BDGP GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles. . . . . 42

7.6 BDGP GEMs generated using 3 sub-stages. The left column shows GEMs overlapped by stages. The middle column shows GEMs overlapped by 1 sub-stage. The right column shows GEMs overlapped by 3 sub-stages. Here stage 7 and 8 are used as illustrative examples. . . . . 43

A.1 Generated FlyFish GEMs overlapped with Atlas. . . . . 55

B.1 Stage 4 - 5 FlyFish GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles. . . . . 57

B.2 Stage 6 - 7 FlyFish GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles. . . . . 57

B.3 Stage 8 - 9 FlyFish GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles. . . . . 57

B.4 FlyFish GEMs generated using 3 sub-stages. The left column shows GEMs overlapped by stages. The middle column shows GEMs overlapped by 1 sub-stage. The right column shows GEMs overlapped by 3 sub-stages. Here stage 6 and 7 are used as illustrative example. . . . . 58

LIST OF SYMBOLS


$n$ Data size . . . . . 11

$d$ Feature vector size . . . . . 4

$\mathbf{x}$ Feature vector . . . . . 4

$y$ Label for $\mathbf{x}$ . . . . . 5

$\ell$ Loss function . . . . . 5

$Reg$ Regularization function . . . . . 5

$\mathbb{R}$ Real number set . . . . . 4

$\mathbb{Z}$ Integer number set . . . . . 5

$T$ Training data set . . . . . 5

$\mathbf{X}$ Data matrix . . . . . 17

$\mathbf{Y}$ Label vector . . . . . 17

# Chapter 1

# INTRODUCTION

## 1.1 Background

*Drosophila melanogaster* (the fruit fly), as one of the model organisms of invertebrates, is widely used for biological research in studies of genetics, physiology, microbial pathogenesis and life evolution. It is typically used because it is a species that is easy to care for, breeds quickly, and lays many eggs [20]. A large number of the key genes involved in fruit fly development are found in the genomes of species in diverse animal phyla and their proteins show extensive sequence similarities [14, 21, 24].

Hence, study of interactions and functions of fruit fly genes is quite meaningful to decipher the mechanisms which govern cell differentiation and embryonic development. The genetic analysis of spatiotemporal information of fruit fly genes involves visualization of the presence of certain gene products (mRNA) at given stage. One way of visualization is the RNA *in situ* hybridization (ISH) technique. It is a gene-specific probe which can illuminate the spatiotemporal pattern of gene expression precisely. The raw data produced from ISH is in the form of digital documented images which reveal certain *Drosophila* gene expression in a specific color.

Recently, BDGP [23] and FlyFish [13] are two popular datasets for *Drosophila* embryos. For the high-throughput characteristic of these two datasets, each image is assigned to a range of stages, according to *Drosophila* embryonic development process, rather than to a specific stage. Thus, the interplay of genes in different stages of development can be studied through comparative analysis of the spatial overlap of gene expression patterns (images).

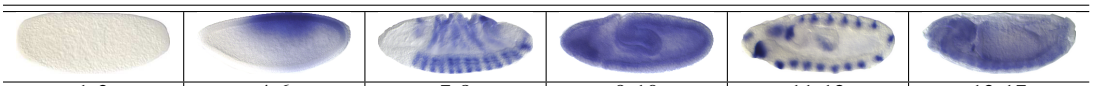


Figure 1.1: Sample images for BDGP data sets. There are 6 stage ranges for BDGP data. Each image is a representative for the corresponding stage range.

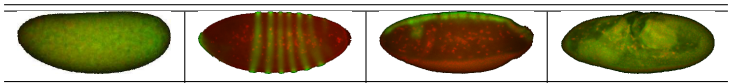


Figure 1.2: Sample images for FlyFish data sets. There are 4 stage ranges for FlyFish data. Each image is a representative for the corresponding stage range.

Advances in ISH technique results in a high-throughput pace of generating images; however, it is a waste of time and resources or even impossible for biologists to do the comparison and stage annotation manually. Moreover, integer stage value is sometimes not enough in the embryonic development stage analysis. A decimal stage value can give more detailed information on embryos' developmental stage. Hence, how to analyze the spatiotemporal pattern automatically and comprehensively and how to annotate the stages as decimal values with only stage range information pose challenging problems to computer scientists. The analytical results will give novel insight into the mechanism of genes. Hence, an automated and efficient system that can analyze the gene expression will become a necessary tool for investigating gene functions, interactions and developmental processes.

## 1.2 Challenges

The specific biological application background and data attributes determine the challenging nature of the stage annotation problem. In the following two sections, the large data size challenge, high feature dimension challenge and application specific challenges are discussed separately. The first two challenges are about the image data which are provided in the two data sets. The last one is about the

challenges that are caused by different requirements of specific applications such as high-resolved stage annotation or binary gene expression image retrieval.

*Large Data Size Challenge*

As described in the *Background* section, because of the advances in biological techniques, fruit fly embryo gene expression image are generated at a high-throughput pace. The data amount to process or annotate is very large. For the example of our application, there are 36,802 images in the BDGP data set as well as 33,572 images in the FlyFish data set. The total data size is about 10GB on a hard drive. On the other hand, due to the fact that embryogenesis is a stochastic process and no two embryos develop in the same morphological pattern, the problem is embedded with more uncertain factors. Hence, how to annotate all the images efficiently and accurately is a challenging problem to our system.

*High Feature Dimension Challenge*

Since the data to be annotated is image data, the feature extracted from this kind of data is commonly in the form of high dimension vectors. The dimension size of the vector depends on the algorithm which is used to extract the feature, and feature dimension size is large most of the time, which causes the curse of dimensionality problem. To deal with this problem, algorithms are required to be more efficient in order to process the large amount of features. It can be even worse when the feature size is larger than the sample size, which is our case. Also, a lot of traditional machine learning classifiers such as Gaussian Mixture Model (GMM) and K Nearest Neighbor (KNN) are not applicable to our problem. Therefore, efficient and state-of-the-art models are critical in solving the problem. I will discuss the techniques that we used in the following chapters.

*Application Specific Challenges*

With different levels of stage granularity posed in the stage annotation problem, the application specific challenges vary accordingly. For the discrete stage annotation problem, in which the annotated stage values are required to be integers, there are only a small portion (about 3K) of data that are manually annotated and about 30K+ data do not have stage information, which makes it hard for us to evaluate the algorithm performance. Also, with more and more new data generated and stored in database in the future, a general model that can annotate unseen data is essential to our system. For the decimal stage annotation problem, according to the previous requirement on annotation granularity, our system should further refine the discrete stage value to be decimal one. If the decimal stage value can be calculated, the analytical comparison of the images will no longer be limited in stage ranges or stages. Biologists will be able to be compare those images in stage level, or even further in any high-resolved level that they can imagine. Hence, how to design a stable and decimal-stage-annotating system poses another two challenges to our system.

## 1.3 Problem Definition

Generally, in machine learning framework, *Drosophila* embryo stage annotation problem is defined as: given a feature vector $\mathbf{x} \in \mathbb{R}^d$ which is extracted from the original image data, find a function $f$ that maps $\mathbf{x}$ to y, in which y is the annotated stage value. The choice of the function should be an optimal function which achieves optimal classification performance. Mathematically, the problem is formulated as follows:

$$\arg\min_{f \in \mathscr{F}} \sum_{\mathbf{x} \in T} \ell(f(\mathbf{x}), y),$$

in which y is the true stage value for $\mathbf{x}$, $\ell$ is the loss function, T is the training data set and $\mathscr{F}$ is the function set.

Specifically, with different requirements, the problem formulation varies slightly. For discrete stage annotation problem, function $f$ is required to be a mapping from $\mathbb{R}^d$ to $\mathbb{Z}$. But for decimal stage annotation problem, function $f$ should be a mapping from $\mathbb{R}^d$ to $\mathbb{R}$.

## 1.4 Thesis Organization

In the following chapters, some related works are discussed in chapter *RELATED WORKS*, a high-level illustration of linear classifier and some well-known classifiers are presented in chapter *OVERVIEW OF LINEAR CLASSIFIERS* and some introductions to the data sets that our application focuses on and how data are pre-processed are described in chapter *DATA SETS AND PREPROCESSING*. Next, detailed descriptions on the algorithms design and the annotation system design are discussed. How to exploit the underlying sparse structure to solve the problem is revealed in chapter *ANNOTATION BY LEARNING SPARSE STRUCTURE*, on which chapter *ANNOTATION BY ENSEMBLE* is based to provide a stable model that can annotate unseen data in high accuracy. With the results in previous two chapters, decimal stage annotation algorithm is illustrated in chapter *ANNOTATION AS DECIMAL STAGE*. Finally, a brief conclusion and some vision to the future works is discussed in chapter *CONCLUSIONS*.

# Chapter 2

# RELATED WORKS

In this chapter, two recent related works are discussed. I will introduce what kind of stage annotation problems are posed and how are they solved. Followed by these, brief summaries are given to each work separately.

## 2.1 LdaPath

In LdaPath paper [25], the stage annotation problem is modeled as a classification problem with 3 classes involved. The 3 classes include the early stages (1-3, 4-6 and 7-8) of embryo development from BDGP data set. For each image from the stage range, Gabor filters [5] are applied to extract textural feature at sub-block level. After the problem is modeled as multi-class classification problem. Multi-class Linear Discriminant Analysis (LDA) is used to do the classification as well as feature extraction at the same time. LDA is a well-known technique that projects high-dimensional feature into low-dimension space where data achieves maximum class separability [2].

Following LDA, in the paper, Ye. et al [25] proved the generalized equivalence relationship between LDA and least square for multi-class case under a mild condition. Based on the equivalence relationship, $\ell_1$ regularization term is added to the least square formulation to force sparsity on the selected feature. And simultaneous feature selection and feature extraction is achieved by solving the optimization problem in LdaPath. The advantages of LdaPath include its high efficiency in computing the entire solution path with the same cost as fitting one LDA model, and its feature sparsity is enforced by $\ell_1$ norm regularization. But the algorithm is still modeled on small data size of 3 stage ranges. The rest of the data remain untouched. And the stage can only be annotated to stage ranges, which is in coarse

granularity. On the other hand, even through the sparsity is enforced on the selected feature, the sparse structure underlies the image data needs further exploring.

## 2.2 Bag of Words

S.Ji et al [8] handled the problem by the approach of bag-of-words. In their approach, invariant features are first extracted through scale-invariant feature transform (SIFT)[19]. And a visual codebook is constructed by applying k-means clustering on SIFT features. After each image is represented as a group of bag-of-words, multi-label classification method [9] is used. The advantages are that images from different views (eg., lateral and dorsal) are used to construct the bag-of-words and control vocabulary [23] is used. The proposed algorithm can annotate images to the level of stages instead of stage range, which is one step further than the previous work of LdaPath [25]. However, the drawback of the approach is that only local descriptive feature, SIFT, is used and global spatial information in the original image is not exploited thoroughly. And the constructed bag-of-word is a global description to the image which can hardly express local differences. Also, the algorithm only focuses on small part (about 3K+ images) of the BDGP data set, the large scale annotation problem is not mentioned and decimal stage annotation is still untouched.

# Chapter 3

# OVERVIEW OF LINEAR CLASSIFIERS

In order to attack the large data size challenge, we used linear classifier as the building block of our algorithms. For multi-classification problem, one-against-rest method [1] is used to convert the original problem into a series of binary-classification problems. Linear models are widely used for its simplicity and high efficiency in data prediction. The general idea for linear model is that it models the decision boundary as a hyperplane in the feature space (Euclidean space). For instance, in 2-D space, the hyperplane is a line; and in 3-D space, the hyperplane is a planar. Upon the linear classifier, other complex classifiers such as AdaBoost and Ensemble can be constructed. Some well-known linear classifiers are introduced as follow.

## 3.1 Support Vector Machine

SVM is short for Support Vector Machine and is inspired from statistical learning theory from Vladimir Vapnik's work in 1996. In SVM, the hyperplane is modeled as the one that represents the largest separation (geometric margin) between the two classes. So the hyperplane is constructed so that the distance from it to the nearest data point on each side is maximized. Hence, the hyperplane is known as the maximum-margin hyperplane and the linear classifier it defines is known as a maximum margin classifier. Also, SVM is considered as a special case of Tikhonov regularization [7].

## 3.2 Logistic Regression

Logistic regression, as a generalized linear model, is used for prediction of the probability of occurrence of an event by fitting data to a logistic sigmoid

function $\sigma(a)$ [1]:

$$\sigma(a) = \frac{1}{1+exp(-a)} \tag{3.1}$$

For two-class classification problem, $\sigma(a)$ models the posterior probability of class $C_1$, and $a$ is a linear function of the feature vector $\mathbf{x}$ so that [1]

$$p(C_1|\mathbf{x}) = y(\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}) \tag{3.2}$$

Maximum likelihood is used to determine the parameters of the logistic regression model. The likelihood function is as follows:

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}, \tag{3.3}$$

in which

$$t_n = \begin{cases} 1, if\ n \in C_1 \\ 0, if\ n \in C_2 \end{cases}, \tag{3.4}$$

$$\mathbf{t} = (t_1, t_2, ..., t_n)^T, \tag{3.5}$$

$$y_n = p(C_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x}). \tag{3.6}$$

Then the parameters are obtained by minimizing the cross-entropy function [1]

$$E(\mathbf{w}) = -lnp(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N}\{t_n ln(y_n) + (1-t_n)ln(1-y_n)\}, \tag{3.7}$$

which provides a new kind of loss function called logistic loss and a general way to look at linear classifier.

## 3.3 Ridge Regression

Ridge regression is also known as Tikhonov regularization which is one of the commonly used methods of regularization of ill-posed problem. In the area of regression, regression problem is formulated as follows:

$$\mathbf{A}x = b$$

But when that problem is not well posed (either because of non-existence or non-uniqueness of data matrix $\mathbf{X}$), standard approaches such as least square may not be able to address the problem well. So, in order to give preference to a particular solution with desirable properties, a regularization term is included in the formulation of least square as follows:

$$min_x 1/2\|\mathbf{A}x - b\|_2^2 + \lambda/2\|x\|_q^2, \tag{3.8}$$

in which $\|\cdot\|_q$ is the q norm of vector $x$ and $\lambda$ is the parameter to control the importance of the regularization term. When $q = 2$, the problem (3.8) becomes ridge regression problem.

## 3.4 Lasso

As another kind of regularization, Lasso is becoming more and more popular for these years. For the problem (3.8), if $q = 1$, then the original becomes Lasso problem. One appealing property of the result hyperplane $x$ of Lasso is that, if $\lambda$ is sufficiently large, most of the entries of $x$ will be driven to zero, which is called sparsity. Exploiting the sparse structure is quite useful to address problem in which feature dimension is much larger than the number of data sample. The main applications include data mining and biological data analysis.

As a summary to the four linear classifiers, there are three commonly used loss functions, which are square loss, hinge loss and logistic loss. Hinge loss is inspired from the loss function in SVM formulation. It is shown in Equation (3.9):

$$\ell(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \max\{0, 1 - y_i \mathbf{w}^T \mathbf{x}_i\} \tag{3.9}$$

Used in ridge regression and Lasso, least squares loss is shown in Equation (3.10):

$$\ell(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = \frac{1}{2} \sum_{i=1}^{n} (\mathbf{w}^T \mathbf{x}_i - y_i)^2 \tag{3.10}$$

Logistic loss is inspired from logistic regression, and it is shown in Equation (3.11) which can be proved to be equivalent to Equation (3.7).

$$\ell(\mathbf{w}, \mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{n} \log(1 + exp(y_i \mathbf{w}^T \mathbf{x}_i)) \tag{3.11}$$

In the three formulations, $x_i$ is the feature vector, $y_i$ is the label value and $w$ is the weight vector which is supposed to be estimated.

# Chapter 4

# DATA SETS AND PREPROCESSING

This chapter mainly describes the data set that our stage annotation system orients on and how these data images are preprocessed to be fed into the machine learning framework properly. First, Berkeley *Drosophila* Genome project (BDGP) and FlyFish data sets are introduced in detail. Then, the methods that used to generate binary expression patterns and feature vectors are illustrated. Finally, how the training data sets are obtained is revealed. And the calculated training data are used in model training in following chapters.

## 4.1 Two Data Sets

BDGP and FlyFish are two databases that store fruit fly gene expression images. Since BDGP was created earlier than FlyFish, the image resolution of FlyFish is higher than BDGP. Due to advanced ISH technique [13] used to generate FlyFish data, the expression pattern in FlyFish database is more clear than those in BDGP database and the color is quite different between the two. Though image data was provided in these previous works, it is not ready to be fed into the system, because the embryos in images are different in sizes and orientations. Each image contains a lot of embryos, which makes it hard to compare images with each other directly. Kumar et al. [12] applied the image standardization procedure to segment each single embryo as an image and align the images into the same size ($128 pixels \times 320 pixels$) and orientation. The standardized images are stored in the FlyExpress database [11].

In Figure 4.1, the left original BDGP gene expression *in situ5449* is an example of a standardized image. The right above binary image is generated by thresholding. There are three thresholds (from low to high) which are used to gen-

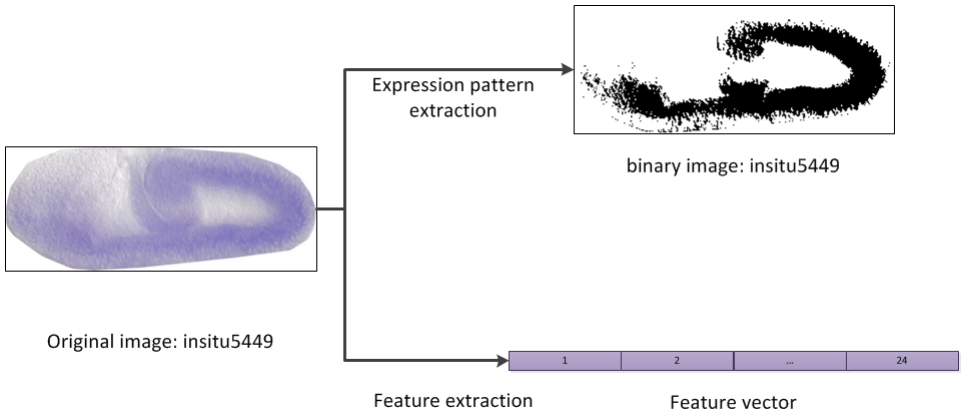


Figure 4.1: Data preprocessing flow. The original image insitu5449 is fetched from FlyExpress database. The binary image is generated by thresholding the original image. The feature vector is extracted from original image by Gabor filters.

erate the binary images. When the lowest threshold is used, the most pixels are reserved in the binary expression image. In the system, the middle threshold is used to extract binary expressions. The right bottom figure is the feature vector which is extracted by a series of Gabor filters.

## 4.2 Gabor Feature Extraction

In order to build an efficient and effective automated annotation system, extracting effective features that contain discriminative information is the first step. In this thesis, we use Log Gabor filter to extract features from original images. Log Gabor filters [5] have been shown to offer the best simultaneous localization of spatial and frequency information with arbitrary bandwidth. In the frequency domain, the Log Gabor function can be described as:

$$G(f) = exp\{-[log(f/f_0)]^2/2[log(\sigma/f_0)]^2\}, \tag{4.1}$$

in which $f_0$ is the filter's center frequency.

With the Gabor filter, the procedures to generate feature vector are as follows. First, the RGB image is converted to a gray scale image. Then Log Gabor filters with 4 different wavelet scales and 6 different filter orientations are used to

extract the texture information. As a result, a combination of 24 images of size $128 pixels \times 320 pixels$ are obtained. Next, each image is down-sampled by a block of $8 pixels \times 8 pixels$. Finally, the down-sampled images are converted into vectors in the same order and the vectors are concatenated to form the feature vector of which the size is 15360.

### 4.3 Training Data Sets Generation

After Log Gabor features were extracted, we built databases for the BDGP and FlyFish images separately with stage information and feature vectors for lateral view images. The stage information is manually annotated by biologists with their domain knowledge. Thus, the databases (training data sets) can provide critical knowledge information for us to transfer into the stage annotation system. The training data size for the BDGP and FlyFish data set are 3721 and 1166 separately, which is almost 10 percent of the original data size. In the databases, the stage value for the BDGP data set ranges from 3 to 17; the stage value for the FlyFish data set ranges from 3 to 10. The difference is because the images in the BDGP data set extend a longer time period than those in the FlyFish data set. The difference between stages 1-3 is just different number of nuclei in each stage, which is not quite morphologically different. Hence, stages 1-3 are considered as one stage (stage 3) in the system.

### 4.4 Genomewide-Expression-Map

Genomewide-Expression-Map (GEM) is a color map that can be used to visualize overlapped binary expression images. When binary expression images with the same stage range, stage or even high-resolved stage are overlapped, the gene expression can be enforced to show the expression pattern of that specific stage resolution. Moreover, within the same stage resolution, binary images of the same gene can be overlapped to show how different genes interact with each other. Along

with manually annotated stage information in the training data sets, the GEMs for the BDGP and FlyFish data set are generated and shown in Figure 4.2 and 4.3.

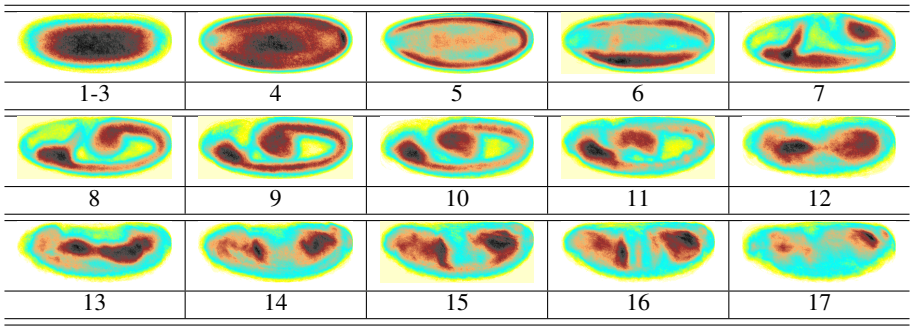


Figure 4.2: GEMs for BDGP. The numbers below the images are stage values. Each GEM is generated by overlapping the binary expression images of the same stage.

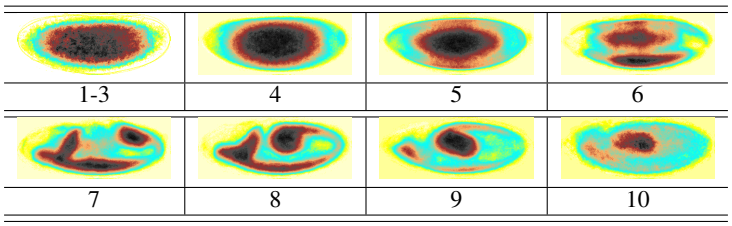


Figure 4.3: GEMs for FlyFish. The numbers below the images are stage values. Each GEM is generated by overlapping the binary expression images of the same stage.

In the two figures, Figure 4.2 shows GEMs for BDGP and Figure 4.3 shows the ones for FlyFish. We can see that the expression patterns are highlighted and quite smooth after the images are overlapped and colored. However, in the data set, the number of images of the same stage range is still large. As shown in Table 4.1 and Table 4.2, the image number is greater than several thousands or even ten thousands, which means the number of images of the same stage is at least over a thousand if the images are evenly distributed. Hence, the GEMs which are shown in Figure 4.2 and 4.3 are still a coarse view of the gene expressions. Also it should be noted that the images which are used to generate the GEMs are just manually

Table 4.1: Number of BDGP images within the same stage range.

| Range | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| Stage | 1-3 | 4-6 | 7-8 | 9-10 | 11-12 | 13-17 |
| Image number | 4204 | 7341 | 3663 | 3765 | 9244 | 8585 |

Table 4.2: Number of FlyFish images within the same stage range.

| Range | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Stage | 1-3 | 4-5 | 6-7 | 8-9 | 10 |
| Image number | 9759 | 13175 | 4355 | 5280 | 1003 |

annotated images in the training data sets. The number of these images is just about 10% of the original size. These facts, again, show the importance of our stage annotation system. It will enable biologists to generate GEMs for all the expression images not only at stage level but also at further refined stages (eg., stage 3.5-4 or stage 4.25-4.5). The details will be discussed in the following chapters.

# Chapter 5

# ANNOTATION BY LEARNING SPARSE STRUCTURE

The idea of sparse learning is to exploit the embedded sparse structure underlying the problem. It is one of techniques that are used to attack the high feature dimension challenge. Instead of projecting all the features into lower dimension space where data are maximum separable, sparse learning assumes that only part of the features affect the classification/regression results and other features should be ignored. One example of a machine learning algorithm that exploits the sparse structure is Lasso. Other state-of-the-art algorithms include group Lasso [26], fused Lasso [18] and overlapping group Lasso [16].

In this chapter, I mainly describe three sparse learning algorithms that are used to exploit the sparse structure underlying the gene expression image data. The algorithms are arranged from the traditional (Lasso) to the state-of-the-art (sparse group Lasso). Each algorithm has different way of modeling the sparse structure. Following the algorithms, model training process, regularization parameters setup, and annotation system construction are illustrated. Finally, the performance of each algorithm is evaluated and compared with the others.

## 5.1 Algorithms

Before we proceed to the details of this chapter, I will give some introductions to the notations that are used in the formulations. We define $\mathbf{X} \in \mathbb{R}^{n \times d}$ as the data matrix and $\mathbf{X} = [\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_n]^T$, in which $\mathbf{x}_i \in \mathbb{R}^d$ ( $1 \leq i \leq n$) and define $\mathbf{Y} \in \mathbb{Z}^n$ as the label vector (annotated stages) and $\mathbf{Y} = [y_1, y_2, ..., y_n]^T$, in which $y_i \in \{3, 4, ..., 17\}$ ( $1 \leq i \leq n$) for BDGP data set or $y_i \in \{3, 4, ..., 10\}$ ( $1 \leq i \leq n$) for FlyFish data set. In the definition, $n$ is the number of training data and $d$ is the feature size (15360 in our case). The following algorithm models are based on the

general form of the optimization problem as follows:

$$\mathbf{w}^* = \arg\min_{\mathbf{w}} \ell(\mathbf{w}, \mathbf{X}, \mathbf{Y}) + Reg(\mathbf{w}), \tag{5.1}$$

in which $\ell(\mathbf{w}, \mathbf{X}, \mathbf{Y})$ is the loss function and $Reg(\mathbf{w})$ is the regularization function. For the following algorithms, different $Reg(\mathbf{w})$ functions are used. In high-dimensional but small size data (e.g., $n < d$ in our case), $Reg(\mathbf{w})$ plays an important role in preventing over-fitting and improving algorithm generalization performance. For each $Reg(\mathbf{w})$ function, we use two kinds of loss functions $\ell(\mathbf{w}, \mathbf{X}, \mathbf{Y})$ which are least squares loss (3.10) and logistic loss (3.11).

*Lasso*

The intuitive way to exploit the sparse structure and to do feature selection is to assume that some entries of the feature vector can be ignored in the process of prediction. The optimization problem should be formulated such that the selected features are decided by the optimal solution to the problem. This inspired us to use $\ell_1$ regularization [22]:

$$Reg(\mathbf{w}) = \lambda \sum_{j=1}^{d} |\mathbf{w}_j| = \lambda \|\mathbf{w}\|_1 \tag{5.2}$$

The $\ell_1$ regularization can enforce some entries of the weight vector $\mathbf{w}^*$ to be zero. In the regularization term, $\lambda$, is a trade-off parameter to control the balance between the loss function and the $Reg(\mathbf{w})$ function. In the weight vector $\mathbf{w}$, the corresponding non-zero entries are the selected features.

*Group Lasso*

Simply letting the solution to decide the selected features may not fully exploit the sparse structure of the image data. Since the feature vector is constructed by concatenating 24 vectors of the Gabor images, the same entry of Gabor image vectors comes from the same $8 \times 8$ block of the original image. As a result, all the features

which come from the same $8 \times 8$ block in the original image should be selected or ignored at the same time. This motivated us to use group Lasso [26]. In group Lasso, features are divided into S disjoint sets. We define the index set of the feature vector as $G_i(\ 1 \leq i \leq S)$. Then, the following equation is satisfied:

$$G_1 \bigcup G_2 \bigcup ... \bigcup G_S = \{1, 2, ..., d\}, \tag{5.3}$$

in which $d$ is the size of the feature vector. The $Reg(\mathbf{w})$ function for group Lasso is formulated as

$$Reg(\mathbf{w}) = \lambda \sum_{i=1}^{S} \|\mathbf{w}_{G_i}\|_2, \tag{5.4}$$

in which $\mathbf{w}_{G_i}$ is the weight vector for the features of i-th group. Similar to Lasso, $\lambda$ is the trade-off parameter to control the balance between the loss function and the $Reg(\mathbf{w})$ function. In the formulation, we define the index set for each group as

$$G_i = \bigcup_{j=0}^{23} \{i + S * j\}, (1 \leq i \leq S) \tag{5.5}$$

Since each Gabor image is down-sampled to $16 pixel \times 40 pixle$ image, the feature vector is partitioned into 640 groups. Hence, the group number S = 640 and $|G_i| = 24$. Then, in the optimal weight vector $\mathbf{w}^*$, entries of the same group will be enforced to be either zeros (ignored) or non-zeros (selected) at the same time.

*Sparse Group Lasso*

Even if the group structure is enforced on the solution to the problem, we need more flexibility with feature selection. Instead of selecting the whole group of features, we formulated sparse group Lasso [17, 4] as a combined solution based on previous two algorithms to attack the stage annotation problem. The $Reg(\mathbf{w})$ function for sparse group Lasso is formulated as

$$Reg(\mathbf{w}) = \lambda_1 \|\mathbf{w}\|_1 + \lambda_2 \sum_{i=1}^{S} \|\mathbf{w}_{G_i}\|_2, \tag{5.6}$$

in which $\lambda_1$ and $\lambda_2$ are regularization parameters. However, it should be noted that there are two parameters in sparse group Lasso formulation and they are used together to control the trade-off between the loss function, the $\ell_1$ regularization term and the $\ell_{2,1}$ regularization term. Index sets $G_i(1 \leq i \leq S)$ for each group are set the same as the ones in Equation (5.5). In the formulation, we can see that sparse group Lasso is a combination of Lasso and group Lasso. Hence, it can enforce "between group" sparsity and "within group" sparsity on the weight vector $\mathbf{w}^*$ simultaneously. This means the weight vector $\mathbf{w}^*$ not only has entries which are selected in groups but also has within-group entries which are selected sparsely. Thus, the sparse group Lasso algorithm can provide more flexibility with feature selection.

## 5.2 Annotation System

As are shown in Table 4.1 and 4.2, each image from the two databases is annotated in the form of stage range ahead of time. What our system is going to do is give a high-resolution annotation. Hence, in order to take the advantages of the available information, we design the annotation as a stage-range-wise annotation process. The algorithm flow for the system is shown in Figure 5.1.

In Figure 5.1, the stage-range-wise annotation process for the BDGP data set is illustrated. The process for the FlyFish data set is similar. We name the problem with regard to stage-range-wise annotation as the with-range problem. This is named in comparison to without-range problem which I will discuss in chapter 7. As shown in Figure 5.1, the images of different stage ranges are annotated separately. Therefore, our system is constructed by a series of models which perform the annotation on images of some certain stage ranges.

During the training phase, two kinds of classification problems (two-class problem and multi-class problem) are considered according to the specific stage

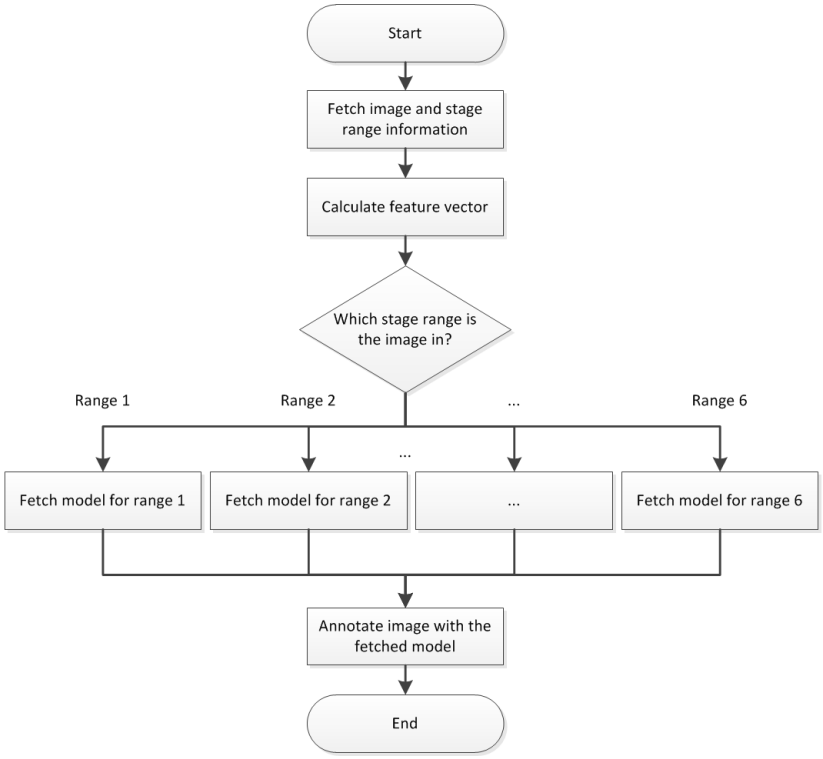


Figure 5.1: Annotation algorithm flow for BDGP data set.

range that is being annotated. In Table 4.1 and 4.2, we can see that different stage ranges have different numbers of stages. For instance, in the BDGP data set, there are 3 stages in range 2 and 2 stages in range 3. In the case of 2 stages, the binary classification model is trained upon the training data. In the case of more than 2 stages within a range, the multi-classification model is trained. For multi-classification model, the one-against-rest method [1] is used to construct the model by a series of binary-classification models. The binary-classification model is obtained by solving the optimization problem which was formulated in previous sections. For each of the proposed sparse formulations, least squares loss (3.10) and logistic loss (3.11) are used.

Before solving the optimization problem, the data matrix $\mathbf{X}$ is obtained by normalizing the feature vectors to be satisfied to $\mathcal{N}(0,1)$ Gaussian distribution.

Table 5.1: $\lambda$ values used in cross validation for Lasso and group Lasso.

| Algorithm | $\lambda$ | | | | |
|---|---|---|---|---|---|
| Lasso | 0.005 | 0.01 | 0.02 | 0.05 | 0.1 |
| Group Lasso | 0.005 | 0.01 | 0.02 | 0.05 | 0.1 |

Table 5.2: $\lambda$ value pairs used in cross validation for sparse group Lasso.

| Algorithm | $[\lambda_1, \lambda_2]$ | | |
|---|---|---|---|
| Sparse Group Lasso | [0.01,0.02] | [0.02,0.02] | [0.05,0.02] |
| | [0.01,0.05] | [0.02,0.05] | [0.05,0.05] |

The trade-off parameter $\lambda$ (or $\lambda_1$ and $\lambda_2$ in sparse group Lasso) is obtained by a 3 fold cross validation [7], in which F1 is used as the criteria to evaluate the performance of the models. The $\lambda$ values that are cross validated are shown in Table 5.1 and 5.2. When both the data matrix and trade-off parameter are ready, the Sparse Learning with Efficient Projections (SLEP) package [15] is used to solve the optimization problem.

The solution $\mathbf{w}$ to the optimization problem is the model parameter that we store for later use in the annotation period. During the annotation period, after the weight vector $\mathbf{w}$ is fetched, two kinds of classification problems are considered. When the problem is a binary-classification problem, the stage of value the image to be annotated is decided by

$$y = sgn(\mathbf{w}^T \mathbf{x}), \tag{5.7}$$

in which $\mathbf{x}$ is the normalized feature vector for the given image, $y \in \{1, -1\}$ and $sgn(\cdot)$ is the sign function. When the problem is a multi-classification problem, the stage value is decided by

$$y = \arg\max_i \{\mathbf{w}_i^T \mathbf{x}\}, \ (1 \leq i \leq R), \tag{5.8}$$

in which $R$ is the number of stages in the stage range, $y \in \{1, 2, ..., R\}$ and $\mathbf{x}$ is the normalized feature vector for the given image.

## 5.3 Evaluation Setup

In order to evaluate the performance of our algorithms, the training data set is randomly split to model training set T and model validation set V. The model training set is used to train the model and model validation set is used to evaluate the performance of the trained model. In our evaluation setup, 5 split ratios are used: 0.5, 0.6, 0.7, 0.8, 0.9. The split ratio is calculated by

$$ratio = \frac{|T|}{|T| + |V|}, \tag{5.9}$$

in which $|\cdot|$ is the size of the set. For each split ratio, the model training set and the model validation set T are randomly sampled 30 times. Hence, there are 30 (T, V) pairs. The algorithm performance for some certain split ratio is given by averaging through the 30 models' performance, which gives a more stable estimation of the algorithm performance.

## 5.4 Results and Analysis

### *Evaluation on Accuracy*

The proposed 6 (3 Reg functions $\times$ 2 loss functions) algorithms are evaluated on the model validation sets. Also, SVM is used as the reference algorithm. In our evaluation, SVM with linear kernel from the LIBLINEAR [3] package is used. The algorithm performance is evaluated in terms of accuracy. The accuracy is obtained by comparing the annotated stage values with the manually annotated ones and calculating the percentage of correctly annotated data. Table 5.3 and 5.4 show the evaluated accuracies of the 6 proposed algorithms against 5 split ratios.

From both tables, we can see that all the proposed algorithms perform almost the same accuracy as SVM while our algorithms can achieve feature selection simultaneously. For the same algorithm with different split ratios, we can see that, with the increase of split ratio, accuracies also increase. This is in accordance with

Table 5.3: Algorithm accuracies evaluated on BDGP data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios.

| Alg. / Ratios | lasso(least) | glasso(least) | sglasso(least) | lasso(log) | glasso(log) | sglasso(log) | SVM |
|---|---|---|---|---|---|---|---|
| 0.5 | 0.84778 | 0.85552 | 0.85531 | 0.85335 | 0.85437 | 0.85462 | 0.86265 |
| 0.6 | 0.85179 | 0.85846 | 0.85901 | 0.85601 | 0.85851 | 0.85716 | 0.86331 |
| 0.7 | 0.85818 | 0.86058 | 0.8601 | 0.85719 | 0.85991 | 0.85972 | 0.86494 |
| 0.8 | 0.86336 | 0.86494 | 0.86609 | 0.86293 | 0.86053 | 0.86317 | 0.86882 |
| 0.9 | 0.86407 | 0.86474 | 0.86879 | 0.8657 | 0.86455 | 0.86869 | 0.8683 |

Table 5.4: Algorithm accuracies evaluated on FlyFish data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios.

| Alg. / Ratios | lasso(least) | glasso(least) | sglasso(least) | lasso(log) | glasso(log) | sglasso(log) | SVM |
|---|---|---|---|---|---|---|---|
| 0.5 | 0.94608 | 0.95055 | 0.94909 | 0.94708 | 0.9416 | 0.93705 | 0.96127 |
| 0.6 | 0.94682 | 0.95039 | 0.95193 | 0.94615 | 0.94547 | 0.93873 | 0.9631 |
| 0.7 | 0.94713 | 0.94995 | 0.95457 | 0.95149 | 0.94931 | 0.94674 | 0.96265 |
| 0.8 | 0.94008 | 0.94566 | 0.94836 | 0.94913 | 0.94412 | 0.93892 | 0.95954 |
| 0.9 | 0.94393 | 0.9639 | 0.96006 | 0.95661 | 0.95392 | 0.947 | 0.96659 |

our common sense that more training samples will provide more information to the algorithm, thus increasing the algorithm performance.

Comparing the accuracies between two data sets, we can see that all the algorithms achieve accuracy over 94% on FlyFish data set, which is much higher than the ones (around 85%) on BDGP data set. This may be caused by the fact that FlyFish data set has a higher image resolution than BDGP data set.

*Evaluation on Sparsity*

In sparse learning algorithm evaluation methods, except for accuracy, another way of evaluation is to calculate the sparsity of the weight vector $\mathbf{w}$ which is obtained by solving the optimization problem. The formula to calculate the sparsity of the trained model is as follows:

$$sparsity = \frac{|\{\arg_i \mathbf{w}_i \neq 0\}|}{d}, \tag{5.10}$$

in which $d$ is the size of feature vector. The sparsity stands for the percentage of non-zero entries in the weight vector. On the other hand, it means the percentage of

selected features in the feature vector. Similarly, in order to get a stable performance evaluation on the sparsity of the algorithms, the algorithm sparsity is calculated by averaging through all the 30 sparsity values under the same split ratio. The sparsities for the proposed algorithms are shown in Figure 5.2 and 5.3.

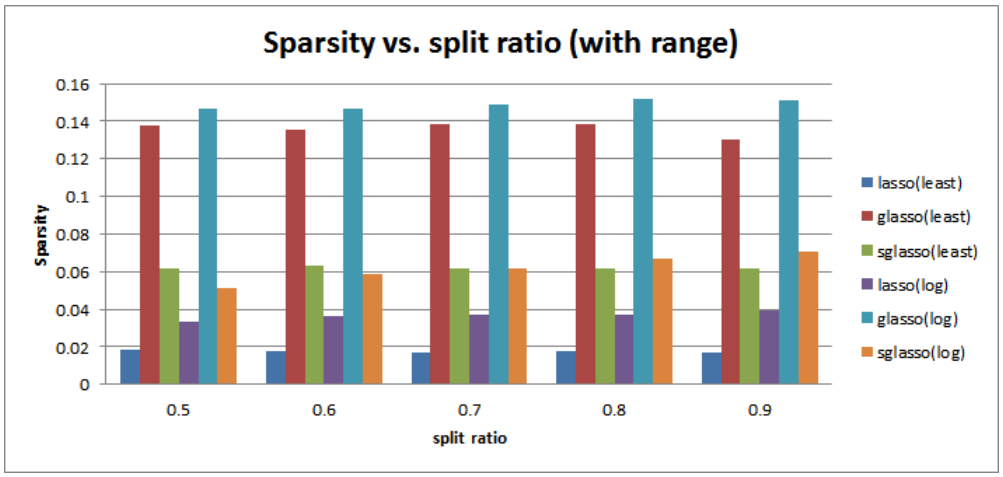


Figure 5.2: Sparsity for each proposed algorithms on BDGP data set.

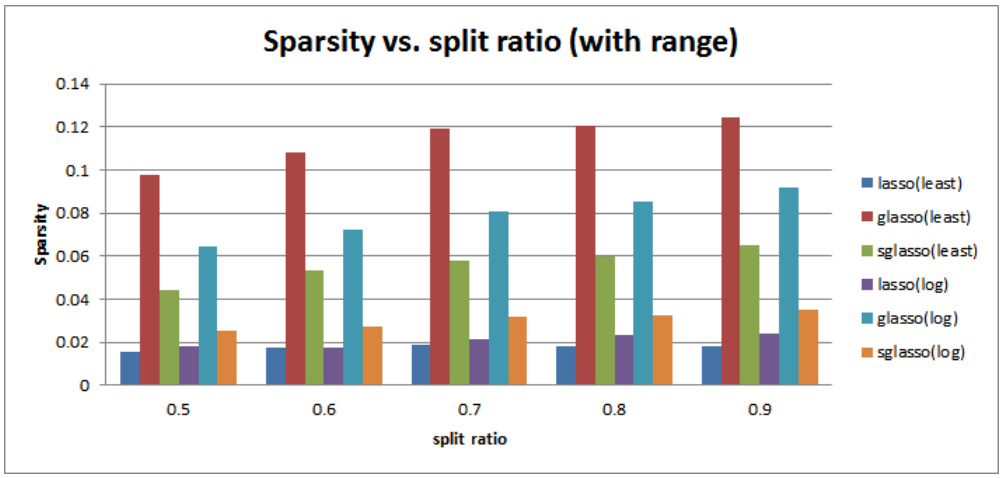


Figure 5.3: Sparsity for each proposed algorithms on FlyFish data set.

From the two figures, we can see that Lasso has the lowest sparsity, group Lasso has the second lowest sparsity and sparse group Lasso sparsity is the third. This explained the differences between sparse structures which are exploited by the three $Reg(\mathbf{w})$ functions. Since Lasso selects features freely, it has the lowest

sparsity. On the other hand, group Lasso selects features under the group restriction, its sparsity is the highest. Therefore, as a combination of Lasso and group Lasso, sparse group Lasso has the sparsity laying between. Also, comparing the sparsity difference between different loss functions, we can see that loss function does not have much effect on sparsity (at most 2% difference). Similarly, comparing the sparsity between different split ratios, we can see that split ratio does not have much effect on sparsity too.

# Chapter 6

# ANNOTATION BY ENSEMBLE

Even though sparse structures are exploited to attack the high feature dimension challenge, the large data size challenge still remain unsolved. Perviously, the algorithm performances are all evaluated on the training data set which only has about 3K data. However, the total data size (about 30K) is 10 times larger than the training data set. Hence, how to generalize our algorithm performance to annotate the remaining data and how to evaluate the proposed algorithm based on the whole data set are the focuses of my next step.

In this chapter, I will introduce how the previously trained models are used to construct the ensemble model which is used as the solution in annotating the whole data sets (both the BDGP and FlyFish data set). The idea of ensemble learning is to build a prediction model by combining the strengths of a collection of simpler base models [7]. In the following sections, first, the ensemble learning algorithm is introduced. Then, the proposed algorithm is used to construct a new ensemble annotation system. By using our system, all 36802 lateral images in the BDGP data set and 33572 lateral images in the FlyFish data set are automatically annotated. Finally, since there are no ground truth for images out of the training set, the algorithm performance is evaluated in a general way by visualizing GEMs.

## 6.1 Algorithm

In order to annotate the images in a stable performance, firstly, a pool of models is constructed. Then, following the algorithm in Figure 5.1, each model in the model pool is used to annotate the fetched image. Next, a vote histogram is calculated by majority voting. At last, the fetch image is annotated as the stage value with the highest votes. In the majority voting method, two ways of calculating

the voting number is proposed. One is increasing the votes by 1 according to the output of each model. The other one is increasing the votes by the model accuracy according to both the model output and the specific model being used.

*Model Pool Construction*

An essential factor to build a annotation system by ensemble method is to contain various models in the model pool. In the previous chapter, we have proposed 6 sparse algorithms. Plus SVM for reference, we totally have 7 algorithms. And for each algorithm, there are 5 split ratios; and for each split ratio, there are 30 randomly sampled model training data. As a result, 1050 ($7 \times 5 \times 30$) models are obtained. All those models are used to build the model pool in our study. Figure 6.1 shows the high-level illustration of our model pool.

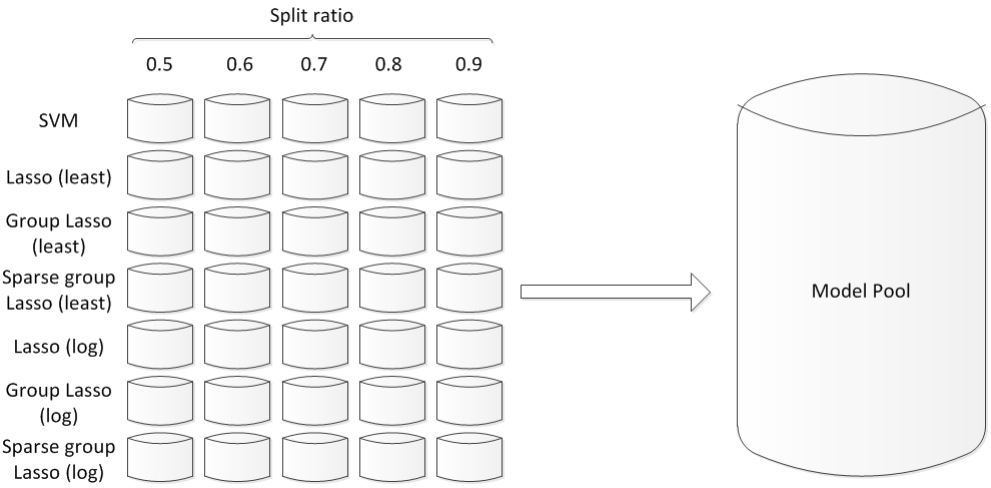


Figure 6.1: Model pool high-level illustration. The left side shows the split ratios and algorithms which are used to construct the model pool. Each bin contains 30 models. Hence, there are 1050 models in the model pool.

In the left side of Figure 6.1, each row represents the models of the same algorithm and each column represents the models which are trained by the same split ratio on the training data. Each bin contains 30 models which are trained under 30 different model training sets.

*Stage Annotation by Majority Voting*

In the model pool, every model in the pool can give an annotation with the information it learned from the training data. Then, majority voting is used to annotate the stage value for each image. Majority voting considers the model pool as a voting committee. It counts the annotation (vote) of each model (committee member) for the fetched image. Then the image is annotated as the stage value with the highest votes. Define $\mathscr{M}$ as the model pool. Then $\forall f \in \mathscr{M}, f : \mathbb{R}^d \to \mathbb{Z}$. The algorithm is shown in algorithm 1.

**Algorithm 1** Stage Annotation Algorithm by Majority Voting

**Require:** Feature vector $\mathbf{x} \in \mathbb{R}^d \wedge \mathscr{M} \neq \emptyset$

**function** STAGE-ANNOTATION-ENSEMBLE($\mathbf{x}$)
    Set voting vector $\mathbf{v} = \mathbf{0} \in \mathbb{R}^s$, in which $s$ is the total number of stage.
    **for all** $model f \in \mathscr{M}$ **do**
        $y = f(\mathbf{x})$
        $\mathbf{v}[y] \leftarrow \mathbf{v}[y] + 1$
    **end for**
    Find index $y^*$ with the highest votes in $\mathbf{v}$.
    **Return** $y^*$
**end function**

In algorithm 1, the vote $\mathbf{v}[y]$ is increase by 1 when a model annotates the image as in stage $y$. However, this does not take the creditability of each model into account. For those models which have a high accuracies, their annotation results should be given more credits than those with lower accuracies. Given this assumption, we proposed an improved version of majority voting which takes the accuracy of each model into the calculation of the voting table. The updating function $\mathbf{v}[y] \leftarrow \mathbf{v}[y] + 1$ in algorithm 1 is modified as $\mathbf{v}[y] \leftarrow \mathbf{v}[y] + Acc_f$, in which $Acc_f$ is the accuracy of each model. The accuracy is calculated in the evaluation part of chapter 5. The improved version of ensemble algorithm is shown in algorithm 2.

**Algorithm 2** Stage Annotation Algorithm by Majority Voting with Accuracy

**Require:** Feature vector $\mathbf{x} \in \mathbb{R}^d \wedge \mathscr{M} \neq \emptyset$

**function** STAGE-ANNOTATION-ENSEMBLE($\mathbf{x}$)
  Set voting vector $\mathbf{v} = \mathbf{0} \in \mathbb{R}^s$, in which $s$ is the total number of stage.
  **for all** $model\,f \in \mathscr{M}$ **do**
    $y = f(\mathbf{x})$
    $\mathbf{v}[y] \leftarrow \mathbf{v}[y] + Acc_f$
  **end for**
  Find index $y^*$ with the highest votes in $\mathbf{v}$.
  **Return** $y^*$
**end function**

## 6.2 Ensemble Annotation System

Since all the models in model pool follows the same annotation process as Figure 5.1, our ensemble annotation system is built upon the previous system. In this case, system in Figure 5.1 is considered as a sub-system to this ensemble system. The overview of our ensemble annotation system is shown in Figure 6.2.

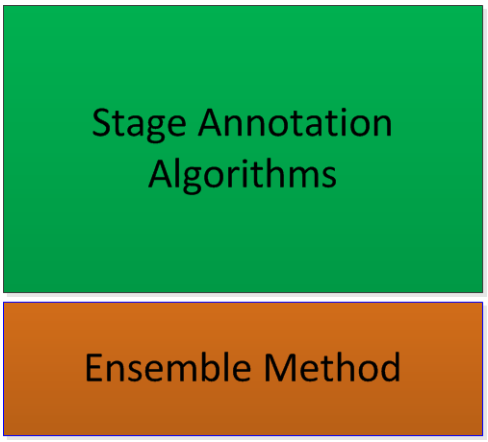


Figure 6.2: Ensemble annotation system overview. The upper component is consisted of a set of stage annotation algorithms. The lower component uses the voting method to combine the results of the annotation algorithms.

## 6.3 Results and Analysis

With the ensemble annotation system, we annotated all the images in the BDGP and FlyFish data set. The GEMs of each stage is generated by overlapping

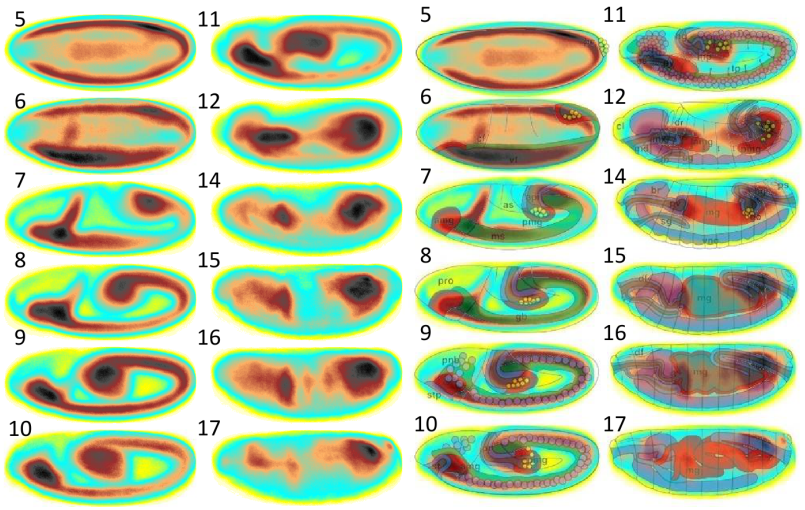


Figure 6.3: Generated BDGP GEMs overlapped with Atlas.

the binary expression images of the same stage value. Then we overlay the GEMs with the stage overview images by Hartenstein [6]. The results for the BDGP data set are shown in Figure 6.3. For the results of the FlyFish data set, please refer to APPENDIX A.

From Figure 6.3, we can see that the GEMs match perfectly with the atlas of *Drosophila* development. The expression patterns (brown part) develop morphologically all most the same as the ones by Hartenstein. From this observation, we believe that our automatic annotation system can generate very reasonable results in general.

# Chapter 7

# ANNOTATION AS DECIMAL STAGE

In the previous two chapters, sparse structures are exploited in our algorithms to attack the large data size challenge and an ensemble method is used to attack the high feature dimension challenge. Up to now, the system can annotate *Drosophila* gene expression images not only efficiently with linear sparse learning models but also accurately with the ensemble method. The system has successfully applied machine learning techniques to handle discrete stage annotation problem. However, the annotated stage value by the system is still coarse-grained (discrete stage value). Therefore, how to enhance the ability of the system to annotate the images as decimal stage values remains an application specific challenge. The automated decimal stage annotation system will be an essential tool for biologists to explore the gene interactions in a finer resolution.

In this chapter, I will mainly introduce a fine-grained stage annotation algorithm that can annotate expression images as decimal stage values. The algorithm is based on the key observation that the *Drosophila* gene expression development process is continuous. First, under the observation, without-range stage annotation problem is posed. Similar sparse algorithm and ensemble method in previous chapters are applied to this problem. Then the voting table of without-range problem is constructed to give a whole-stage voting distribution which is used to calculate the decimal stage value. The idea of calculating the decimal stage value is giving a displacement to the annotated discrete stage value. The magnitude of the displacement is based on the votes of the voting distribution. With the decimal stage annotation algorithm, the annotation system has a new enhancement based on the system in chapter 6. Next, several applications of the decimal stage value are posed. Those

are just limited to our imagination from the point of computer science. The real biological extension is still waiting for biologists to explore. Finally, I will show some of the results generated by our system and give some analysis to them.

## 7.1 Expression Pattern Transition as a Continuous Process

Although the annotation by ensemble algorithm can annotate all the BDGP and FlyFish gene expression images as discrete stage values with precise accuracy. It is still coarse-grained to express the embryo development process in the term of stages (eg., stage 3 or stage 4). From Figure 4.2 and 4.3, we can see that the transition of gene expression in GEMs is smooth. The patterns are changed from one stage to the next in a continuous manner. Based on this key observation, we assume that the binary gene expression pattern images are all subjected to a continuous transition process. Hence, moreover, we assume the original images which are used to generate binary expression images are also subjected to the same process. Even though it is intuitive to think about the embryogenesis as a continuous process, it is interesting to notice that genes which control the embryo development still function continuously on different part of the embryo. Based on this assumption, our decimal stage annotation algorithm is constructed and several exciting applications of the decimal stage value are illustrated in the follow sections.

Given the previous assumption, it will be more useful and reasonable to annotate the expression images as some stage value that is between two integer stage values. Since it is more likely that some of them are in the early state of that stage and the rest of them are in the late state, we used decimal stage value to model the extent of early state and the late state. Also, this is in correspondence with the requirement that the annotated stage values should be decimal values.

However, in the with-range problem, the annotation results fall in the values that are restricted by the stage ranges. For the images which are in the early state

of the first stage or in the late state of the last stage of some stage range, the votes for previous stage of the first stage or the votes for the next stage of the last stage cannot be obtained. For example, if a voting table is obtained for an BDGP image of range 2 (stage 4 - 6) by using the ensemble method in chapter 6. The votes will be limited in the stage 4 - 6. The votes for stage 3 or stage 7 are zeros. Hence, if an image falls in the boundary (early stage 4 or late stage 6) of range 2, it is hard to infer how early or late that image is by using the nearby votes of that stage.

Therefore, in order to annotate the image as decimal stage values and let the system to be tolerant to images which fall in in the boundary of the stage ranges (in the early period of the first stage of that range or in the late period of the last stage of that range), we defined without-range stage annotation problem. The definition and our solution are shown in the next section.

## 7.2 Without-range Problem

Even though the stage ranges are provided in the original data sets, it is important to define a kind of problem that is lack of stage range information and find out a way of generalizing our system to handle the problem. For the stage annotation problem in which stage range information is not provided as a prior knowledge, we name it as without-range stage annotation problem which is short for without-range problem. Without range problem assumes the stage range information of each image is not given ahead of time. For each image, the system is supposed to generate the annotation only with the given feature vector.

Following the idea in chapter 5 and chapter 6, we designed a new system component to fulfill the without-range annotation task. The component consists of a set of algorithms and an ensemble method. The algorithms exploit the sparse structure to generate annotation in the case of lacking stage range information. The algorithm flow is shown in the following figure. The ensemble method is the same

as the one in chapter 6.Since stage range information is not provided in without-range problem, the annotation algorithm is slightly different from the one in with-range problem (Figure 5.1). The modified algorithm flow is shown in Figure 7.1.

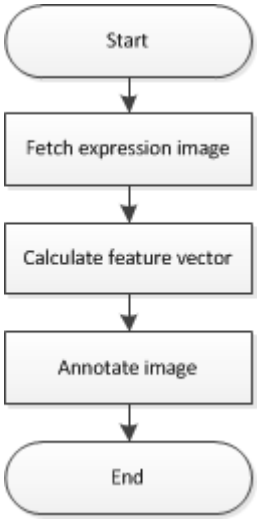


Figure 7.1: Without-range stage annotation algorithm flow.

In Figure 7.1, we can see that the algorithm fetches the image and then annotate it with the trained model. The model can annotate images as any discrete stage value of the data set. During the model training phase, we follow the same setup in chapter 5. The same training data set is used. However, the difference is that there are only one model for the same split with some certain split ratio. In the with-range problem, the number of models is equal to the number of stage ranges of the given data set. For algorithm performance evaluation, the same model training sets and model validation sets as chapter 7 are used. The detailed results and analysis are shown in the next *Results and Analysis* section. We applied the same ensemble method which is shown in chapter 6 to annotate all the data in the two data sets. What is different is that the models used in ensemble method are the without-range models which are used in the algorithm flow in Figure 7.1.

## 7.3 Algorithm

From the voting distribution obtained from without-range problem, we noticed that, in most of the cases, the votes for stages which are beside the stage of the highest votes (annotated stage) are the second and third highest. Hence, the votes indicate the trend of that image to be in early stage or late stage. The votes ratios of the highest and the second highest votes between different images of the same stage indicate the relative order of those images. This motivated us to designed the decimal stage annotation algorithm.

**Algorithm 3** Decimal Stage Annotation Algorithm

**Require:** Voting vector $\mathbf{V} \in \mathbb{R}^S$

**function** DECIMAL-STAGE-ANNOTATION(**x**)
  $y$ = Stage-Annotation-Ensemble(**x**)
  **if** $2 \leq y \leq S-1$ **then**
    **if** $\mathbf{V}[y-1] > \mathbf{V}[y+1]$ **then**
      status = -1
      $vote = \mathbf{V}[y-1]$
    **else**
      status = 1
      $vote = \mathbf{V}[y+1]$
    **end if**
  **else if** $y == 1$ **then**
    status = 1
    $vote = \mathbf{V}[2]$
  **else if** $y == S$ **then**
    status = -1
    $vote = \mathbf{V}[S-1]$
  **end if**
  **Return** $y^* = y + 0.49 * status * \frac{vote}{vote + \mathbf{V}[y]}$
**end function**

We define $y$ as the annotated integer stage value obtained by solving with-range problem; and $\mathbf{V}$ as the voting vector obtained from without-range problem, in which $\mathbf{V} \in \mathbb{R}^S$ and $S$ is the number of stages in the given data set.

In algorithm 3, *status* stands for the status of the image with regard to the annotated stage value. 1 stands for that the image is in the early status and -1 stands for that the image is in the late status. *vote* is the value of second highest votes. The vote ratio $vote/(vote+\mathbf{V}[y])$ is used to describe how early or late the image is with regard to the annotated stage value $y$. The ratio increases with the increasing of the *vote*. Hence, the higher the ratio, the later or earlier the stage is. Here coefficient 0.49 is used to scale the ratio to the range [0,0.49], so that the late stage value of stage $y$ will not be overlapped with the early status value of stage $y+1$ or with the late status value of stage $y-1$. Then $y^*$ is the annotated decimal stage value.

## 7.4 Decimal Stage Annotation System

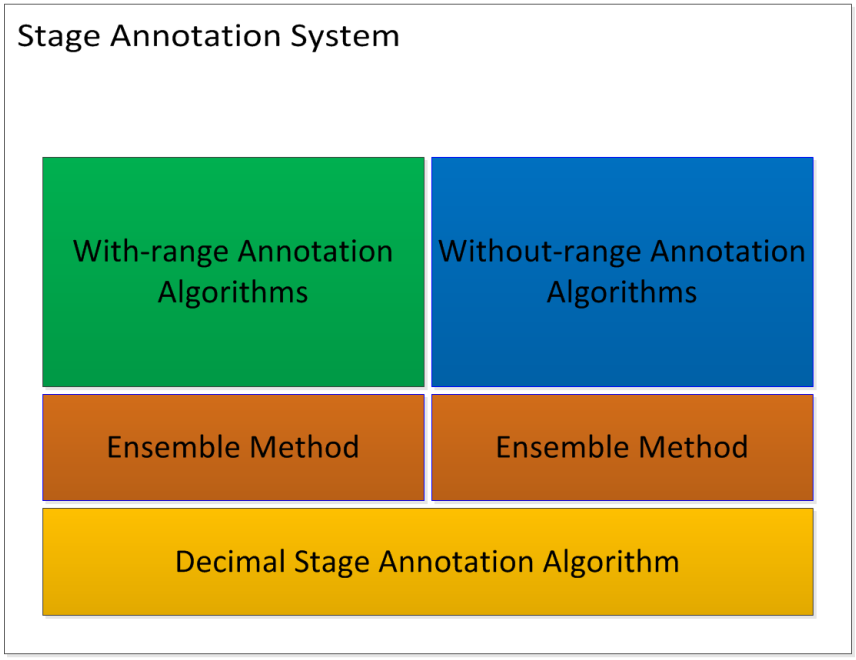


Figure 7.2: Final version of stage annotation system overview. Each module is an algorithm that are posed in previous chapters.

With decimal stage annotation algorithm, the final version of our system is constructed as shown in Figure 7.2. Each module is consisted of one or several proposed algorithms. The left side handles the with-range problem and the right side handles the without-range problem. Both subsystems share the same ensemble

algorithm. The outputs of them are used as the inputs to decimal stage annotation algorithm.

## 7.5 Applications of Decimal Stage

Decimal stage value provides a further refined resolution to categorize the gene expression images. It broadens our imagination of analytically comparing of those expression images. In the following, I will mainly introduce three main applications of the decimal stage value. These are just limited to our understanding to the problem of stage annotation. The wider range of its applications are waiting biologists to explore.

### *Image Sorting*

Previously, even though the ensemble algorithm can give an annotation to each images, the annotated stage values are just discrete values. As a result, there are still a lot of images being annotated to the same stages. Using the results obtained by solving the with-range problem, we count the numbers of images in each stages for the BDGP and FlyFish data set. The results are shown in the following tables. Compared to Table 4.1 and 4.2, they are finer-grained statistics of image stage distribution.

Table 7.1: Number of images in each stage for the BDGP data set. All the 36802 images are annotated by solving the with-range problem.

| Stage | 3 | 4 | 5 | 6 | 7 |
|---|---|---|---|---|---|
| Image # | 4204 | 2544 | 3373 | 1424 | 1683 |
| Stage | 8 | 9 | 10 | 11 | 12 |
| Image # | 1980 | 2588 | 1177 | 5673 | 3571 |
| Stage | 13 | 14 | 15 | 16 | 17 |
| Image # | 2319 | 1604 | 1211 | 1780 | 1671 |

In Table 7.1 and 7.2, we can see that there are thousands of images in each stages. Hence, even though we can annotate the images with high accuracy, it is still

Table 7.2: Number of images in each stage for the FlyFish data set. All the 33572 images are annotated by solving the with-range problem.

| Stage | 3 | 4 | 5 | 6 |
|---|---|---|---|---|
| Image # | 9759 | 4927 | 8248 | 3076 |
| Stage | 7 | 8 | 9 | 10 |
| Image # | 1279 | 2415 | 2865 | 1003 |

hard to compare so many images of the same stage. However, with decimal stage value, we get an order for each image. Moreover, we can use this order information (decimal stage value) to sort all images in an increasing order. With the sorted images, each image can be easily compared to the others under any give ranges. For instance, each image can be compared to the most nearest image or the nearest 10 images. For the decimal value, it should be noted that only the relation between two values matters and the exact value do not have any meanings. For instance, we have 4 images A, B, C and D with decimal values as 5.0, 5.5, 6.0 and 6.7. We can say that image B is in later stage than image A. So as image C and D. But it is incorrect to say that, D developed more against C than B against A.

*Sub-stage Annotation*

In the original data sets, the stage range gives a rough categorization to the developmental process of each embryo. The annotated discrete stage value gives a finer categorization. With decimal stage value, we further refined the stage categorization into sub-stage. We named the process of this further refinement as sub-stage annotation which gives a high-resolved stage categorization. The resolution granularity can be adjusted by sub-stage number. Define $S$ as the number of stages for each data set, and $n$ as the number of sub-stages. The sub-stage is annotated in the following procedures:

1. Calculate the decimal stage value for all the image in the data sets.
2. For some given stage $y(1 \leq y \leq S)$, fetch all the images within its early

and late state.

3. Sort all the fetched images by their decimal stage values in ascending order.

4. For the given sub-stage number $n(n \leq 1)$, evenly divide the sorted early and late state images into $n$ disjoint groups.

5. For each group $i(1 \leq i \leq n)$, the annotated stage is calculated by the following formula.

$$y_i = \begin{cases} (y + \frac{i-1-n}{2n}, y + \frac{i-n}{2n}), \text{i is in early state} \\ (y + \frac{i-1}{2n}, y + \frac{i}{2n}), \text{i is in late state} \end{cases} \tag{7.1}$$

In equation 7.1, we can see that the sub-stage $y_i$ is annotated as an interval. The concept of interval is more precise to express our idea that the embryogenesis is a continuous process and it is more reasonable to annotate the development of each embryo to some certain range instead of some specific value.

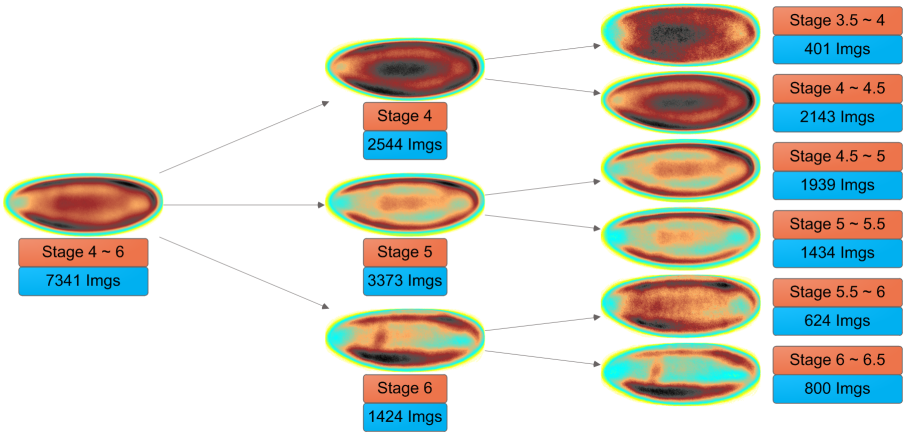


Figure 7.3: Stage 4 - 6 BDGP GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles.

With the annotated sub-stage, we overlapped the binary gene expression images of the same sub-stage by weighting the number of different genes. The

GEMs are shown as follows. Since there are a lot of generated GEMs. Only GEMs for the BDGP data set is shown here. For the ones of the FlyFish data set, please refer to APPENDIX B.

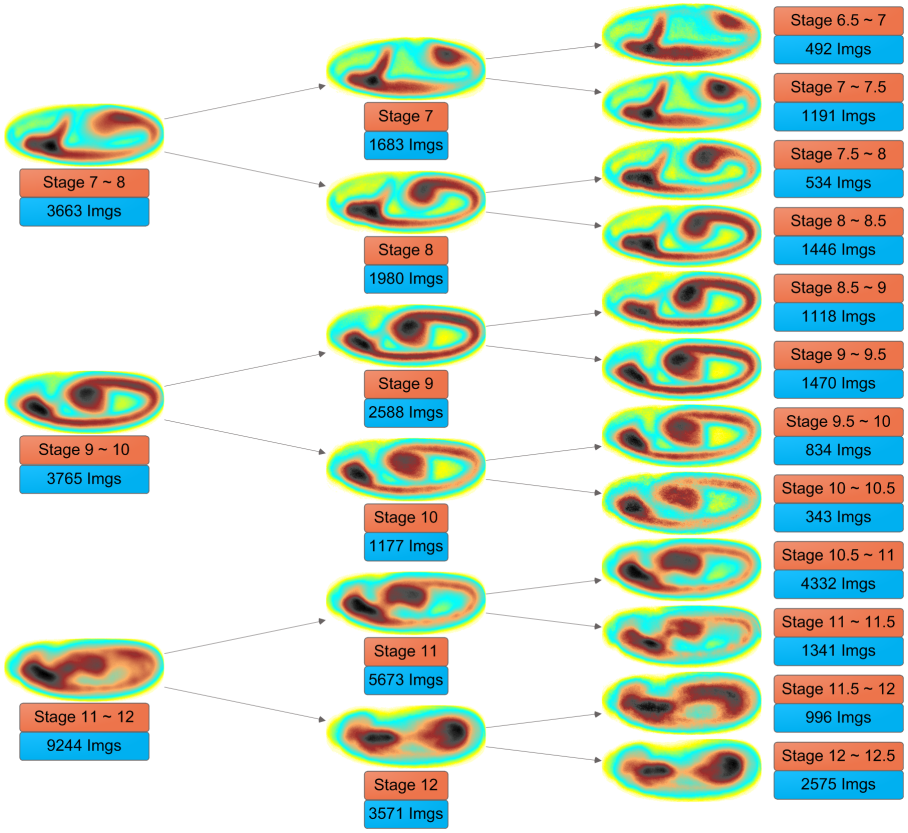


Figure 7.4: Stage 7 - 12 BDGP GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles.

Figure 7.3, 7.4 and 7.5 show the GEMs generated by setting sub-stage number to 1. Figure 7.6 shows the GEMs generated by setting sub-stage number to 3. In Figure 7.6, image number of the right column is not shown, because the images are evenly grouped from sorted late or early state images. Comparing the GEMs of different stage resolution, we can see that the continuous gene development process is revealed in more fine-grained detail using the results of sub-stage annotation. The

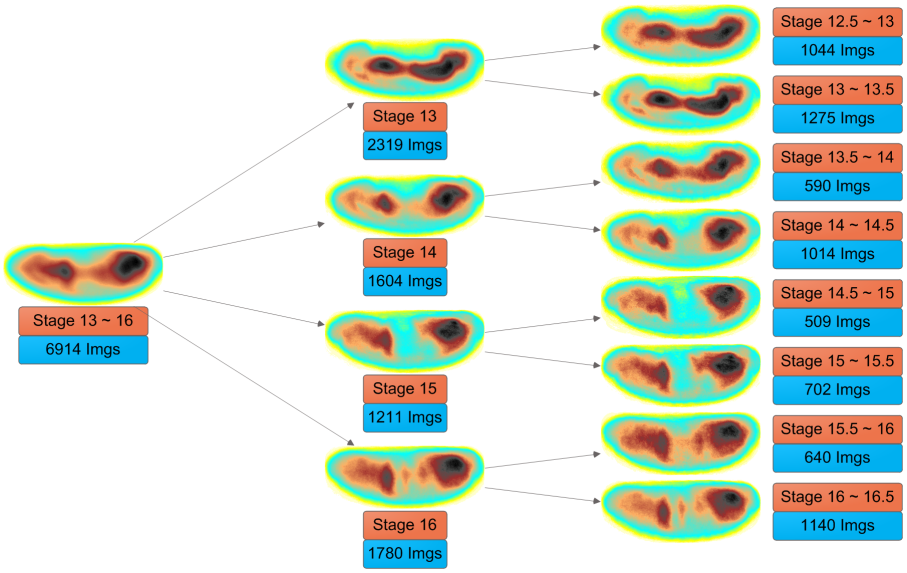


Figure 7.5: Stage 13 - 16 BDGP GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles.

transition from one stage to the next turns out to be more smooth then the GEMs generated by using stage values.

*Improving Similar Expression Pattern Retrieval*

The FlyExpress [11] system previously uses the similarity measurement [10] of the two binary expression images to do the image retrieving. The images are returned by the system in the order from highest similarity to the lowest. The similarity measurement $S_{QD}$ is calculated by the following equation:

$$S_{QD} = \frac{|Q \bigcap D|}{|Q \bigcup D|}, \tag{7.2}$$

where $|Q \bigcap D|$ is the size of the intersection of expression between image $Q$ and $D$ and $|Q \bigcup D|$ is the size of the union between image $Q$ and $D$ [10]. However, this way of measurement can not capture the developmental similarities. For example, two images have high $S_{QD}$, but they are actually in different stages or even different

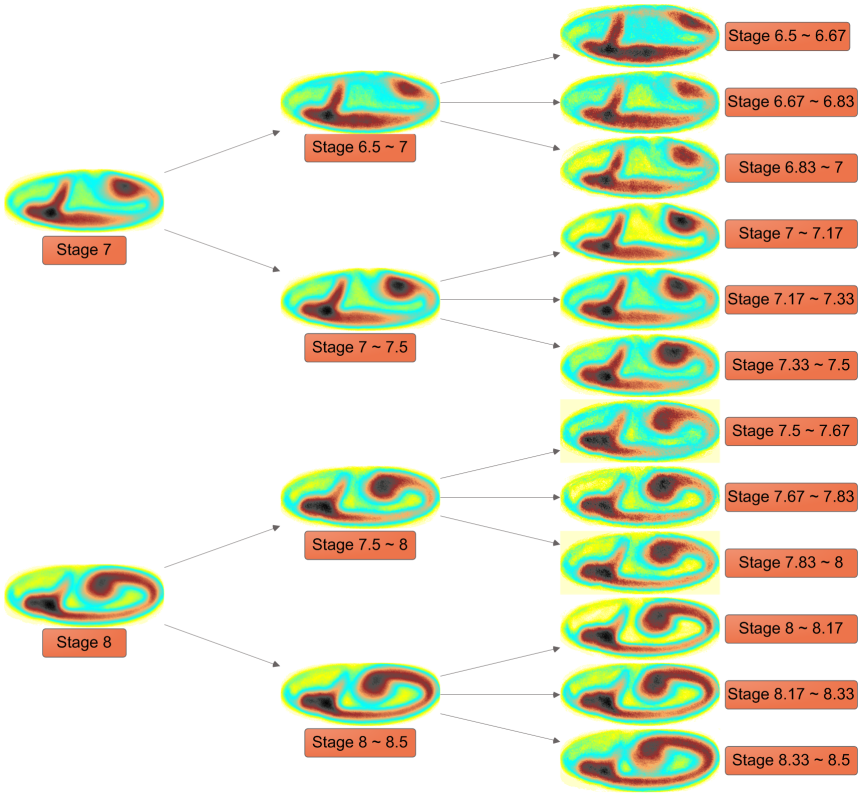


Figure 7.6: BDGP GEMs generated using 3 sub-stages. The left column shows GEMs overlapped by stages. The middle column shows GEMs overlapped by 1 sub-stage. The right column shows GEMs overlapped by 3 sub-stages. Here stage 7 and 8 are used as illustrative examples.

stage ranges. So it is necessary to figure out a way of restricting the search domain within the same stage or sub-stage.

Here comes another application of our sub-stage annotation. With the help of annotated stage or sub-stage information, we can further refine the retrieving within some certain stage or sub-stage. Therefore, the images which are not in the same stage or sub-stage will not be retrieved.

## 7.6 Results and Analysis

We evaluated the performance of sparse learning algorithms in solving the without-range problem. Then the performance under the with-range and without-range problem is compared with each other.

*With-range Problem vs. Without-range Problem*

In order to evaluate how do the 6 sparse algorithms perform on the without-range problem, we use the same setup as in chapter 5. The obtained accuracies are shown in Figure 7.3 and 7.4.

Table 7.3: Without-range algorithm accuracies evaluated on the BDGP data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios.

| Alg. / Ratios | lasso(least) | glasso(least) | sglasso(least) | lasso(log) | glasso(log) | sglasso(log) | SVM |
|---|---|---|---|---|---|---|---|
| 0.5 | 0.73639 | 0.74724 | 0.75295 | 0.7834 | 0.78609 | 0.78363 | 0.79122 |
| 0.6 | 0.74891 | 0.7508 | 0.76328 | 0.78932 | 0.78927 | 0.78814 | 0.79105 |
| 0.7 | 0.75853 | 0.76222 | 0.77019 | 0.79273 | 0.79401 | 0.79177 | 0.7889 |
| 0.8 | 0.77333 | 0.77775 | 0.77947 | 0.79549 | 0.79683 | 0.79492 | 0.78795 |
| 0.9 | 0.78189 | 0.77948 | 0.77697 | 0.79788 | 0.79875 | 0.79817 | 0.7707 |

Table 7.4: Without-range algorithm accuracies evaluated on the FlyFish data set. *glasso* stands for group Lasso. *sglasso* stands for sparse group Lasso. *least* stands for least square loss. *log* stands for logistic loss. The first row is the split ratios.

| Alg. / Ratios | lasso(least) | glasso(least) | sglasso(least) | lasso(log) | glasso(log) | sglasso(log) | SVM |
|---|---|---|---|---|---|---|---|
| 0.5 | 0.8569 | 0.87591 | 0.87683 | 0.90162 | 0.90488 | 0.89532 | 0.91771 |
| 0.6 | 0.86931 | 0.87804 | 0.88469 | 0.90966 | 0.90916 | 0.90637 | 0.9176 |
| 0.7 | 0.87924 | 0.8821 | 0.88896 | 0.9086 | 0.91365 | 0.9065 | 0.9187 |
| 0.8 | 0.8887 | 0.89127 | 0.89814 | 0.91302 | 0.91803 | 0.9133 | 0.92132 |
| 0.9 | 0.89726 | 0.88784 | 0.89384 | 0.91039 | 0.91981 | 0.91467 | 0.91781 |

Comparing the accuracies of the same data set between different problems, we can see that, in the BDGP data set, the algorithms have a better performance in with-range problem (about 85%) than in without-range problem (about 77%). Similar results can be observed in the FlyFish data set. This is in consistent with our intuition that, the more information (stage-range information) provided, the better the algorithms perform. However, it should be noted that, even through the problem is harder in the case of without-range, the algorithms just perform at most 6% lower than SVM while achieving feature selection.

*Independent Evaluation*

Since, for the images which are not in the training set, we do not have stage information. In order to evaluate the performance of our annotation system, we setup an independent evaluation experiment in which 140 stage 4 - 17 BDGP images are randomly selected from the data set. During the selection, 10 images are selected for each stage. Within each stage, 5 images are in early state and late state separately. Then we ask a domain expert to annotate these images in the term of 1 sub-stage (e.g. 7 early as $6.5 \sim 7$, 7 late as $7 \sim 7.5$). The results are considered as the ground truth in independent evaluation. Among the 140 images, the ones which are improper to manually annotate are eliminated first. The improper images includes images that are mis-labeled as lateral view image or the images that are out of focus. For the proper images, the following three criteria are used in our evaluation.

1. **Sub-stage Accuracy**. The annotation result is considered as correct if the sub-stage annotation result is the same as the ground truth. For example, a manually annotated "stage $6.5 \sim 7$" image is considered accurate only if it is annotated as "stage $6.5 \sim 7$" by the system.

2. **Stage Accuracy**. The annotation result is considered as correct if the stage annotation result is in either early or late state of stage in the ground truth. For example, a manually annotated "stage $6.5 \sim 7$" image is considered accurate if it is annotated as "stage $6.5 \sim 7$" or "stage $7 \sim 7.5$" by the system.

3. **Plus-Minus-Half Accuracy**. The images that are annotated at most "half stage away" from the manually annotated side-stage are considered accurate. For example, a manually annotated "stage $6.5 \sim 7$" image is considered accurate if it is annotated as "stage $6 \sim 6.5$", "stage $6.5 \sim 7$" or "stage $7 \sim 7.5$" by the system.

We evaluated the system performance on both with-range and without-range problem. The results are shown in table 7.5 and 7.6.

Table 7.5: Independent performance evaluation results of the annotation system (stage-range information is used).

| Total Images | Proper Images | Sub-stage Accuracy | Stage Accuracy | Plus-Minus-Half Accuracy |
|---|---|---|---|---|
| 140 | 117 | 76.07% | 86.32% | 94.87% |

Table 7.6: Independent performance evaluation results of the annotation system (stage-range information is not used).

| Total Images | Proper Images | Sub-stage Accuracy | Stage Accuracy | Plus-Minus-Half Accuracy |
|---|---|---|---|---|
| 140 | 117 | 75.21% | 81.20% | 93.16% |

From the tables, we can see that, for the system performance in with-range problem, 76.7% accuracy is achieved in sub-stage accuracy while 86.32% accuracy is achieved in stage accuracy. It should be noted that, during the training phase, only the stage information is provided to the algorithms. The system generalized the ability to do sub-stage annotation only in a reduction of 10% accuracy. Also the achieved 86.02% accuracy is almost the same as the accuracies evaluated in chapter 5. This indicates that our system can potentially annotate the unseen images in a high accuracy. This, on the other hand, shows the high accuracy of our ensemble system in chapter 6. Last but not least, our system achieves an amazing accuracies of 94.87% in plus-minus-half accuracy. The means that even if our system fails to annotate the image to the correct sub-stage, it is quite likely that the annotation is very close to the ground truth.

For the evaluation on the without-range problem, we obtained the same results as in with-range problem. This again indicates that our system has a very well generalization performance. However, what is interesting is that, even if range information is not provided, the drops of accuracies are very small. This shows that our system has robust performance in the case where stage range information is not available.

We also evaluated the system performance when difference algorithms are used in ensemble algorithm. In the evaluation, two setups are evaluated. The first one is only using SVM models in the ensemble algorithm while the second one is using the 6 sparse algorithms. The results are shown in table 7.7, 7.8, 7.9 and 7.10. In the results, both with-range performance and without-range performance are listed.

Table 7.7: Independent performance evaluation results of our annotation system (stage-range information is used). Only SVM models are used in ensemble method.

| Total Images | Proper Images | Sub-stage Accuracy | Stage Accuracy | Plus-Minus-Half Accuracy |
|---|---|---|---|---|
| 140 | 117 | 54.70% | 80.34% | 92.31% |

Table 7.8: Independent performance evaluation results of our annotation system (stage-range information is not used). Only SVM models are used in ensemble method.

| Total Images | Proper Images | Sub-stage Accuracy | Stage Accuracy | Plus-Minus-Half Accuracy |
|---|---|---|---|---|
| 140 | 117 | 50.43% | 76.07% | 91.45% |

Table 7.9: Independent performance evaluation results of our annotation system (stage-range information is used). 6 sparse models are used in ensemble method.

| Total Images | Proper Images | Sub-stage Accuracy | Stage Accuracy | Plus-Minus-Half Accuracy |
|---|---|---|---|---|
| 140 | 117 | 67.52% | 85.47% | 94.87% |

Table 7.10: Independent performance evaluation results of our annotation system (stage-range information is not used). 6 sparse models are used in ensemble method.

| Total Images | Proper Images | Sub-stage Accuracy | Stage Accuracy | Plus-Minus-Half Accuracy |
|---|---|---|---|---|
| 140 | 117 | 65.81% | 82.05% | 94.02% |

From the evaluated results, we can see that most accuracies in both of the two setups are lower than the ones obtained by using all the models. Only the accuracies where the 6 sparse models are used have similar values to the results obtained by using all the models. This, on the other hand, shows the reasonableness of our system which uses all the models in the ensemble algorithm.

# Chapter 8

# CONCLUSIONS

In this thesis, a stage annotation system is constructed to solve the stage annotation problem by transferring the domain knowledge in biology into a machine learning framework. During the construction of the automatic stage annotation system, different algorithms are used to annotate the *Drosophila* gene expression images in a granularity from coarse to fine. Using the sparse learning algorithms, the system succeeds to annotate the high dimensional data with high accuracy and efficiency. Using the ensemble algorithm, the system annotates all the images in two data sets to attack the large data size challenge. An independent evaluation shows that the system can still achieve a high accuracy when unseen images are provided. Using the decimal stage annotation algorithm, the system fulfills the task to annotate the images as decimal values. With annotated decimal stages, we illustrate a series of interesting applications such as image sorting, sub-stage annotation and improving similar expression pattern retrieval. The final version of the stage annotation system is constructed from bottom to up by combining the three algorithms as three system components.

In the part of algorithm results analysis, different evaluation methods are used. In evaluation of the sparse learning algorithms, accuracy and sparsity are used. In the analysis of the ensemble algorithm, a way of visual evaluation is posed by showing GEMs. On the other hand, due to the lack of ground truth of the whole data set, independent evaluation is used to analysis the performance of the proposed ensemble annotation algorithm and decimal annotation algorithm.

As a summary, sparse learning algorithm is a efficient way to attack large data size challenge, ensemble algorithm can provide robust performance in pre-

dicting large amount of unseen data with just a small percentage of training data (about 10% in our case) and voting table is a useful statistical result to derive decimal prediction value. Moreover, the process of decimal stage annotation can be generalized. In the decimal stage annotation algorithm, we solve the with-range and without-range problem separately. The output of the with-range problem gives a tight stage bound in annotation results while the one of without-range problem gives a loose bound in annotation results. However, the loose bound in the without-range annotation problem may results in some votes to be far from the actual stage range. For example, if we are annotating the image of stages 4 to 6, some without-range votes may in stage 10. These votes will not be used in the calculation of decimal stage. Hence, when annotating the image of stages 4 to 6, we just need to train another model that can annotate the image to stage 3 to 7 instead of stage 3 to 17. By doing this, the system will must provide a more accurate estimation on decimal stage value. Also, the application domain of the proposed system can be extended to other decimal value prediction problem. Only if the data are subject to some continuous transition process, the system framework can be applied to give a decimal value prediction with only a few training samples.

Future works can be divided into three parts. The first part is about system robustness and accurateness which include designing an aforementioned generalized decimal stage annotation algorithm and improving the accuracy of basic learning algorithm. The second part is about system extensibility which includes extending our system to annotate image of other views such as dorsal view and extending our system to be able to process unstandardized or our-of-focus images. The third part may be system applications which include exploring more applications of the decimal stage value or extending our system framework to the prediction of other continuous signals.

REFERENCES


[1] Christopher M. Bishop. *Pattern recognition and machine learning*. Springer, 1st ed. 2006. corr. 2nd printing edition, October 2006.

[2] Richard O. Duda, David G. Stork, and Peter E. Hart. *Pattern classification*. John Wiley, 2 edition, November 2000.

[3] Rong E. Fan, Kai W. Chang, Cho J. Hsieh, Xiang R. Wang, and Chih J. Lin. LIBLINEAR: A Library for Large Linear Classification. *J. Mach. Learn. Res.*, 9:1871–1874, June 2008.

[4] J. Friedman, T. Hastie, and R. Tibshirani. A note on the group lasso and a sparse group lasso. *Arxiv preprint arXiv:10010736*, 2010.

[5] DAUGMAN J. G. Complete discrete 2-d gabor transform by neural networks for image analysis and compression. *IEEE Trans. Acoust. Speech & Signal Process*, 36:1169–1179, 1988.

[6] Volker Hartenstein. *Atlas of Drosophila Development*. Cold Spring Harbor, NY: Cold Spring Harbor Laboratory Press, 1993.

[7] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer, corrected edition, August 2003.

[8] Shuiwang Ji, Ying-Xin Li, Zhi-Hua Zhou, and Sudhir Kumar. A bag-of-words approach for drosophila gene expression pattern annotation. *BMC Bioinformatics*, 10(119), April 2008=9.

[9] Shuiwang Ji, Lei Tang, Shipeng Yu, and Jieping Ye. Extracting shared subspace for multi-label classification. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 381–389, New York, NY, USA, 2008. ACM.

[10] Charlotte E. Konikoff, Timothy L. Karr, Michael McCutchan, Stuart J. Newfeld, and Sudhir Kumar. Comparison of embryonic expression within multigene families using the flyexpress discovery platform reveals more spatial than temporal divergence. *Developmental dynamics : an official publication of the American Association of Anatomists*, September 2011.

[11] Sudhir Kumar. Fly express database. http://www.flyexpress.net.

[12] Sudhir Kumar, Karthik Jayaraman, Sethuraman Panchanathan, Rajalakshmi Gurunathan, Ana Marti-Subirana, and Stuart J. Newfeld. BEST: A Novel Computational Approach for Comparing Gene Expression Patterns From Early Stages of Drosophila melanogaster Development. *Genetics*, 162(4):2037–2047, December 2002.

[13] Eric Lécuyer, Hideki Yoshida, Neela Parthasarathy, Christina Alm, Tomas Babak, Tanja Cerovina, Timothy R. Hughes, Pavel Tomancak, and Henry M. Krause. Global analysis of mRNA localization reveals a prominent role in organizing cellular architecture and function. *Cell*, 131(1):174–187, October 2007.

[14] Michael Levine and Eric H. Davidson. Gene regulatory networks for development. *Proceedings of the National Academy of Sciences of the United States of America*, 102(14):4936–4942, April 2005.

[15] Jun Liu, Shuiwang Ji, and Jieping Ye. *SLEP: Sparse Learning with Efficient Projections.*, 2009.

[16] Jun Liu and Jieping Ye. Fast overlapping group lasso. *arXiv:1009.0306v1*, 2010.

[17] Jun Liu and Jieping Ye. Moreau-yosida regularization for grouped tree structure learning. *Advances in Neural Information Processing Systems*, 23:1459–1467, 2010.

[18] Jun Liu, Lei Yuan, and Jieping Ye. An efficient algorithm for a class of fused lasso problems. *ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2010.

[19] David G. Lowe. Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110, 2004. 10.1023/B:VISI.0000029664.99615.94.

[20] E. Reeve. *Encyclopedia of Genetics*. Taylor & Francis, 2001.

[21] P. Simpson. Evolution of development in closely related species of flies and worms. *Nat Rev Genet*, 3(12):907–917, December 2002.

[22] Robert Tibshirani. Regression Shrinkage and Selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[23] Pavel Tomancak, Amy Beaton, Richard Weiszmann, Elaine Kwan, ShengQiang Shu, Suzanna Lewis, Stephen Richards, Michael Ashburner, Volker Hartenstein, Susan Celniker, and Gerald Rubin. Systematic determination of patterns of gene expression during Drosophila embryogenesis. *Genome Biology*, 3(12), 2002.

[24] K. M. Weiss. The phenogenetic logic of life. *Nat. Rev. Genet.*, 6(1):36–45, 2005.

[25] Jieping Ye, Jianhui Chen, Ravi Janardan, and Sudhir Kumar. Developmental Stage Annotation of Drosophila Gene Expression Pattern Images via an Entire Solution Path for LDA. *ACM transactions on knowledge discovery from data*, 2(1), March 2008.

[26] Ming Yuan and Yi Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 68(1):49–67, February 2006.

# APPENDIX A

## FLYFISH GEMS OVERLAID WITH ATLAS

Since, for the FlyFish data set, the developmental stage starts from stage 3 to stage 10, and stages overview in Atlas start from stage 5, there are only GEMs of stage 5 through stage 10 are shown here.

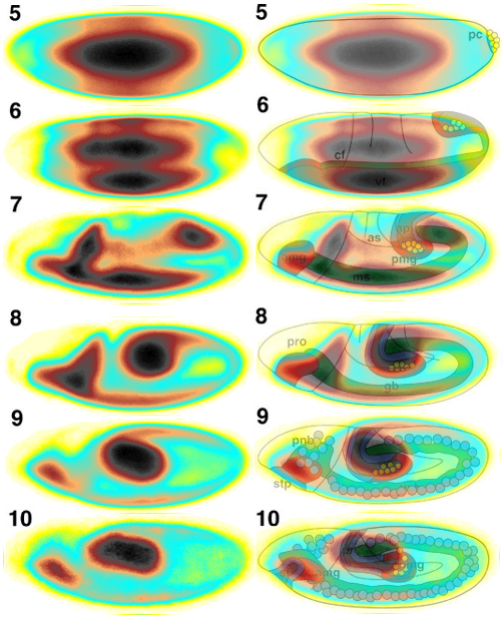


Figure A.1: Generated FlyFish GEMs overlapped with Atlas.

# APPENDIX B

## SUB-STAGE GEMS FOR FLYFISH

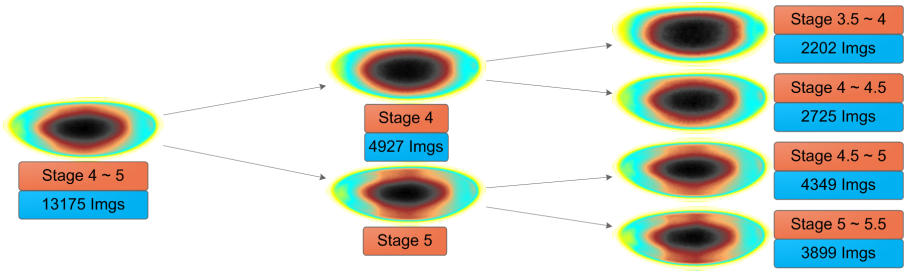


Figure B.1: Stage 4 - 5 FlyFish GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles.

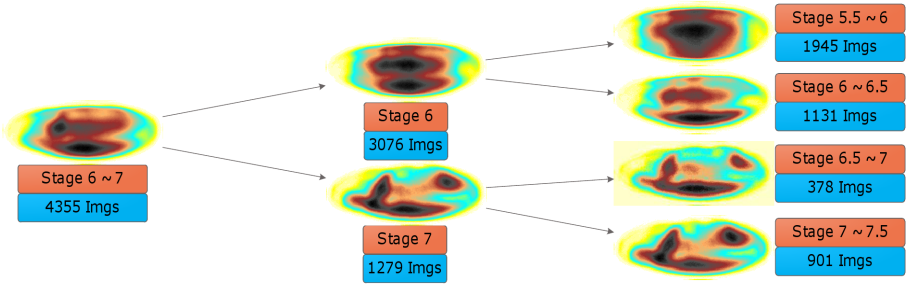


Figure B.2: Stage 6 - 7 FlyFish GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles.

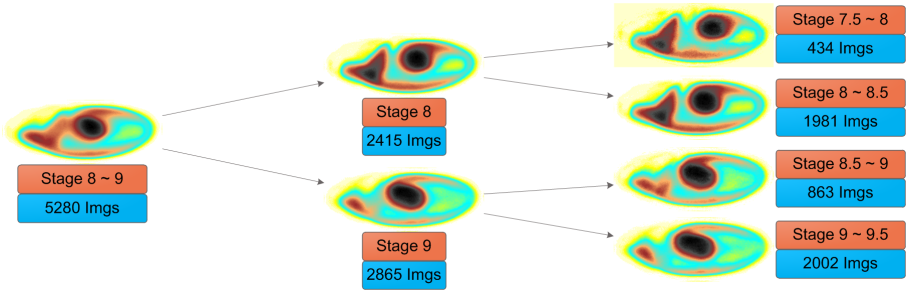


Figure B.3: Stage 8 - 9 FlyFish GEMs generated by using only the stage range information (left column), using the predicted stage information (middle column) and using the sub-stage information (right column, sub-stage is 1). The total numbers of images used for creating each individual GEM are shown in blue rectangles.

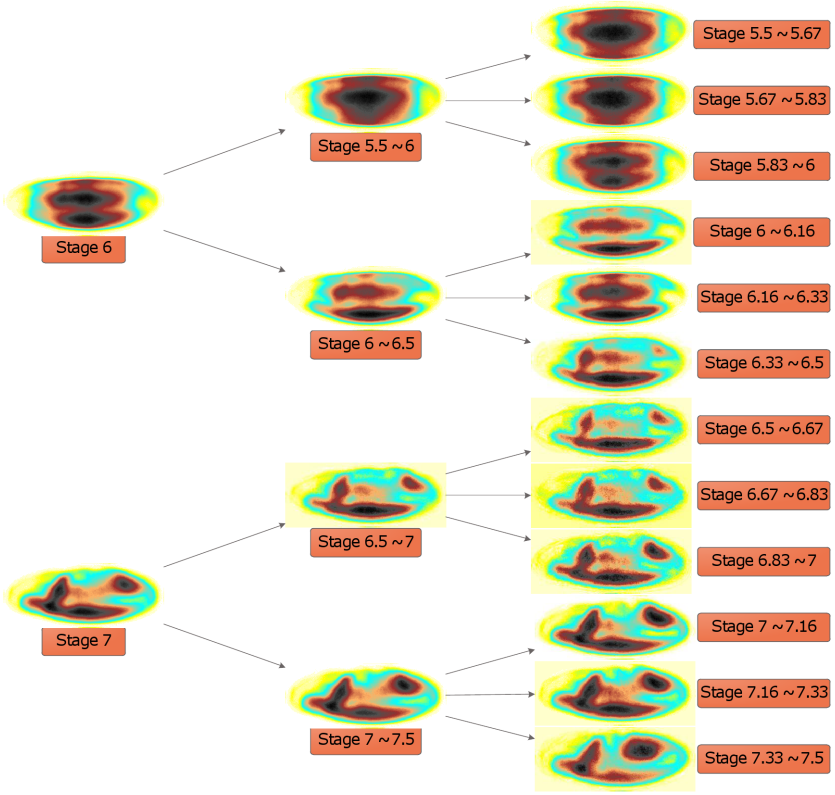


Figure B.4: FlyFish GEMs generated using 3 sub-stages. The left column shows GEMs overlapped by stages. The middle column shows GEMs overlapped by 1 sub-stage. The right column shows GEMs overlapped by 3 sub-stages. Here stage 6 and 7 are used as illustrative example.